Understanding Game Balance with Quantitative Methods

Alexander Jaffe

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2013

Reading Committee:

James R. Lee, Chair

Zoran Popović, Chair

Anna Karlin

Program Authorized to Offer Degree:
UW Computer Science & Engineering

University of Washington

Abstract

Understanding Game Balance with Quantitative Methods

Alexander Jaffe

Co-Chairs of the Supervisory Committee:
Professor James R. Lee
CSE

Professor Zoran Popović
CSE

Game balancing is the fine-tuning phase in which a functioning game is adjusted to be deep, fair, and interesting. Balancing is difficult and time-consuming, as designers must repeatedly tweak parameters and run lengthy playtests to evaluate the effects of these changes. Only recently has computer science played a role in balancing, through *quantitative balance analysis*. Such methods take two forms: analytics for repositories of real gameplay, and the study of simulated players. In this work I rectify a deficiency of prior work: largely ignoring the players themselves. I argue that variety among players is the main source of depth in many games, and that analysis should be contextualized by the behavioral properties of players. Concretely, I present a formalization of diverse forms of game balance. This formulation, called 'restricted play', reveals the connection between balancing concerns, by effectively reducing them to the fairness of games with restricted players.

Using restricted play as a foundation, I contribute four novel methods of quantitative balance analysis. I first show how game balance be estimated without players, using simulated agents under algorithmic restrictions. I then present a set of guidelines for using domain-specific models to guide data exploration, with a case study of my design work on a major competitive video game. I extend my work on this game with novel data visualization techniques, which overcome limitations of existing work by decomposing data in terms of player skill. I finally present an advanced formulation of fairness in games - the first to take into account a game's metagame, or player community. These contributions are supported by a detailed exploration of common understandings of game balance, a survey of prior work in quantitative balance analysis, a discussion of the social benefit of this work, and a vision of future games that quantitative balance analysis might one day make possible.

TABLE OF CONTENTS

LIST OF FIGURES

DEDICATION

to my singular grandparents: the heart, the hands, the face, and the soul of my family.

Chapter 1

INTRODUCTION

Among creative artifacts, games have the rare distinction of being largely unobservable by their creators. A painter or writer, upon finishing a work, can step back and observe exactly what it is she has created. Games are different. As an interactive system, a game is characterized by a space of countless explicit experiences, rather than just one. These systems are often defined by a very small set of mechanics, from which gameplay arises emergently. Indeed, many of the games which we consider the *deepest* - such as *Go* - are defined by a single page of rules, yet have many more potential play-throughs than atoms in the universe. This phenomenon is particularly evident in competitive games, which often aim to remain fresh and interesting to their players for many years, without requiring years of developer content creation.

1.1 The Problem of Balancing

If designers must contemplate the effect of each rule change on 10^{170} possible play-throughs, we might ask how it is that one designs games at all. The answer: roughly, slowly, and expensively. Much of this work comes, of course, from the initial design and extensive content creation. Yet a large amount of work goes into fine-tuning the gameplay, *after* the game has reached a playable state. This period of subtle adjustments is known as balancing, and it is perhaps the most technically difficult portion of the design process. This is where a game is massaged into the form the designer envisions for it. At this point, the basic feel of the game and its interaction modalities have for the most part already been fixed. However, the global consequences of actions and the relationships between strategies are emergent, and hence difficult for the designer to plan for or predict. How useful are various actions? How important is unpredictable play? How varied is the pace of the game? These are among the questions a designer attempts to understand and manipulate in balancing.

Traditionally, balancing is thought of as something of a black art. In its simplest form, the process consists of a play-analyze-iterate loop. Designers play or observe the game, attempt to generalize their observations to a high-level understanding of what the game is like to play, then adjust parameters and features in a way that they believe will move the

gameplay in a direction they envision. This process has been practiced for centuries, even in a distributed fashion in the evolution of games like *Chess*, *Go*, and *Soccer*.

Despite the ad-hoc process of balancing, much of the nature of balance is fundamentally mathematical. It is concerned with the high-level structure and properties of large systems, protocols, graphs, state machines, and random processes. It seems that computer science and statistics should be able to offer some path to make the process less difficult. The challenge is one of problem formulation. Games are a diverse medium, and attempts to formalize normative goals for their design have failed, if only for the existence of countless counter-examples. We should no sooner attempt to define a 'good' game as we would a 'good' painting. Each designer has her own vision, and each player has her own preferences. Yet in this diversity we can find a more realistic goal: simply helping a designer *understand* her game. A good designer knows what the 'good' in her game can be; the difficulty is in understanding if and how the game has achieved that good. The goal, in other words, is to make game design more like painting and writing: to reveal games to their designers, through the properties that define their aggregate experience. Efforts to this end, which I call *quantitative balance analysis* (QBA), have taken two distinct forms: analysis of human gameplay, and analysis of simulated gameplay.

1.2 Introducing Quantitative Balance Analysis

I define quantitative balance analysis as the use of quantitative methods in formulating, measuring, and understanding game balance. This definition is quite broad, yet in practice most feasible contributions come in one of two forms. These two forms of QBA serve to decompose both prior work and much of the work that I will discuss in this dissertation. I briefly discuss each form here.

Instrumented Gameplay Analysis This older form of QBA is concerned with maximizing the insight derived from human playtests. As data storage and processing has become more convenient, game developers have begun recording gameplay in a number of fashions, including video, biometrics such as heart rate and skin conductance, and automated attitudinal surveys. The most prevalent method is telemetry: the instrumentation of code to record players' actions and their effects on the game world. In a typical development cycle, a developer can play or observe at best thousands of play-throughs. With telemetry, it is possible to investigate millions of play-throughs at once, provided one has access to sufficient players, and is satisfied to examine coarser signals than full play-throughs, such as the average number of moves of a given piece. The difficult question is *how* to derive actionable insights from so much rich, heterogenous data. Dozens of papers address this topic, which I survey in Chapter 3.1. For example, one common approach utilizes heatmaps

to visualize the movements and actions of players on a game map or state space.

Automated Analysis This more nascent form of QBA attempts to evaluate games without the use of any human players at all. Gameplay instrumentation requires a good deal of play-throughs, typically from a variety of players. Yet it often takes a deep competitive game hundreds of iterations to converge on a successful design. As a consequence, many games must rely disproportionately on designer intuition, and may end up being released in a severely unbalanced state. It should be possible to reason computationally about certain properties of a game without using any players at all. The real challenge here is in formulating design properties rigorously enough to productively analyze them. A small collection of papers have pursued this challenge. The most common approach is to study instrumented gameplay of simulated players. I survey this work in Chapter 3.2.

1.3 The Challenge of Competitive Games

The existing work on gameplay analysis has proven valuable in the development of many real games. Yet there is a deficiency it its approach, which is highlighted by the case of competitive games. In competitive games, one might say that *the players are the levels.* Although many competitive games offer a variety of scenarios for players to compete in, the primary source of variety emerges from opponents themselves. *Depth* comes from players, where a game is deep if there are many ways to play; if actions have meaningful consequences; if it never ceases to reveal nuances to experienced players.[1] After all, playing against a new Player 2 creates a fundamentally new state machine from Player 1's perspective, just as if the level itself had changed. Games such as *Chess* and *Go* offer limitless new experiences with a single starting condition. Even games that offer multiple possible starting conditions - such as *Street Fighter* and *Starcraft* - contain this same breadth of experience within any one of these conditions.

If we accept the premise that depth often comes from players, the natural response is to analyze games in terms of the space of real and possible players. To capture at a high level the experience of playing, say, *Halo* multiplayer, I argue that we should first know how strong and weak players play, how offensive and defensive players play, how melee and ranged players play, and the effects of each of these playstyles on the game's outcome. Yet existing research in gameplay analysis misses this opportunity. Little to none of the games research to date on analysis of player data decomposes or contextualizes the data by the high-level properties of the players themselves. In the language of Smith et al. [82], researchers often employ 'universal' player models. The need for player contextualization is

[1] It is sometimes said that the depth of a game can be measured by the number of tiers of skill it affords.

great. It is highlighted by competitive games, but it is by no means restricted to them.

My thesis, below, addresses this need.

Thesis

Game balance can be understood through the effectiveness of explicitly modeled players.

This thesis statement gets at the heart of what distinguishes game balance as a special case of game design. I claim that when designers and players speak of balance, their concern is not the totality of player actions and game events, but rather those actions and events that *matter*. At first this claim seems to beg the question. Yet in reality, in many games the source of meaning is clear: winning and losing. Meaning flows outward from a player's goals toward every aspect of the game. If a mechanic has no impact on a player's goals, there is some sense in which it simply does not matter to gameplay. *Of course* the goal-independent aspects of a game (be they superfluous bits of mechanical entertainment, aesthetic properties, or narrative) are crucial to a player's holistic experience. But these aspects are not typically meaningful to the gameplay - to the choices and actions of the player. For the purposes of this dissertation, I will take it for granted that such aspects sit outside the purview of what we call game balance.[2] I will focus on games in which we take players' goals to be clear, whether these goals are designer-specified or otherwise. Questions of how to best capture these goals are left open for future consideration.

Concretely, I will frame games as having concrete notions of winning and losing. I will take this one step further and focus much of my exposition on multiplayer competitive zero-sum games, in which we assume each player's primary goal is to win. Such simplifying assumptions somewhat limit the immediate applicability of the work, but are essential to making initial progress in truly understanding what game balance is and how to measure it.

Of course, a designer's concern is not simply whether players win or lose, but *how* they win or lose. The goal of a designer is to create an experience for players; a game's mechanics and goals form the system of channels that guides players through this experience as they act freely. Each player will navigate these channels differently, and it is this diversity - this freedom - that makes evaluating game balance difficult, particularly for deep, challenging games. If we are to capture balance in a truly informative way, our characterization must not just describe the form of gameplay, but specify for *whom* the gameplay takes that form. This is where player models come in. A player model is nothing more than a mode of play,

[2]By no means do I wish to define what is or is not a game; the medium benefits from diversity. I am content to call sandbox games like *Sim City* or *Minecraft* games, and even games of pure exploration like *Proteus*. In such games the goal is not specified for players, although players do often impose their own goals as they play. Yet as goals become less clear, game balance becomes less meaningful as well.

with certain biases, tendencies, knowledge, and desires. Player models, in combination with effectiveness (the impact of a game dynamic on a players ability to win), form the two key facets of my thesis statement. This thesis statement is largely philosophical, but one of my primary contributions is to make it concrete, using a framework we call restricted play. It is from this formulation that all of my work on balance evaluation arises.

1.4 Restricted Play

The premise of restricted play is very simple. I claim that many forms of game balance can be reframed as the success of some restricted mode or method of play. This empowers us to formally, quantitatively define these forms of balance, which until now have primarily been framed as purely intuitive concepts. Specifically, I argue that many traditional forms of game balance attempt to capture the magnitude of the role played some game dynamic, where a dynamic is a facet of gameplay as specific as an individual mechanic, or as abstract and high-level as pace or strategy. Then in order to measure the impact of some dynamic D, we measure the win probability of a player who is restricted in its use of D in some way.

For example, to measure the impact of a particular player action, such as moving a certain piece in *Chess*, we consider a player who is restricted from moving that piece more than k times in a game (or, alternately, fewer than k times), and we ask the probability that the player wins against an opponent without the restriction. To measure the role of unpredictable play, we consider a player who is restricted from playing a high-entropy strategy distribution, and ask the probability that this player wins. As we will see, a wide variety of disparate balance questions can be framed in this way. Most importantly, each question framed in this way is placed in a common quantitative framework: a probability, between 0 and 1. This allows game features to be compared against one another, across time, and across games.

This basic description of restricted play does not entirely capture its motivation and details, which are elaborated in chapter 4. However, the description is sufficient to understand broadly the QBA work that it facilitates, which form my primary contributions.

1.5 Contributions

In this dissertation, I develop a player-centric approach to quantitative balance analysis, through novel analytical methods and an improved understanding of game balance itself. This document serves as a blueprint for a powerful, efficient balancing methodology. All of my contributions can be framed in terms of restricted play, although some of the contributions can be easily understood without explicit reference to restricted play. Many of

these ideas will require years of further development. Yet already in this dissertation we can begin to see the promise of new and refined techniques.

The core of my work consists of advancements in each of the two forms of QBA discussed above: instrumented gameplay and automated analysis. Although they differ, both topics rely on a clear understanding of the problems, goals, and methods of game balancing. Deep understanding of (or at least consensus on) the nature of the process remains sparse. As such, a sizable portion of my thesis is dedicated to exploring characterizations of game balance, including traditional processes, related work on quantitative balance analysis, and my own attempt at taming the problem using restricted play. My dissertation is thus divided broadly into two parts. The first part seeks to characterize game balance, and the second utilizes these characterizations to measure it and inform the game design process. I now overview the major ideas and contributions of each of these parts.

1.5.1 Characterizing Game Balance

Balance is a topic for which few can agree on a definition, let alone an ideal process. Yet even within the confines of this dissertation, I decline to fix a concrete definition. Balance is a multifaceted concept for good reasons, and I would be remiss to rule any of them out. Instead, I dedicate Part I of my thesis to painting a picture of what game balance is, why it is important, and how it is achieved. In this way the reader can be brought up to speed about the space of topics under discussion, even without prior familiarity with game design.

In the first chapter of Part I I aim to capture the meaning and methods of balance through the words of three prominent game designers and educators. I summarize each of their thoughts on game balance, contextualized by their individual perspective, then attempt to synthesize a working understanding of balance. I also explore the relevance of balance beyond its traditional context, by discussing the potential positive societal benefits of work in balancing.

The second chapter of Part I surveys related work in quantitative balance analysis. This chapter is itself separated into related work on the two forms of quantitate balance analysis: instrumented gameplay and automated analysis. The majority of this related work is not situated toward balance in particular, but rather analyzing or automating game design in general. Nonetheless, much of it applies, just as some of the work in this thesis could apply to more general design concepts without much modification. I choose to concentrate on balancing to keep the problems I face tightly focused on some of their most difficult and concrete forms.

The third chapter of Part I outlines the principles of restricted play, which will serve as the conceptual grounding for the rest of my dissertation. Restricted play is simple to

describe, but I also take pains to contrast it with prior methods of quantitative balance analysis, and motivate its use. To give some sense of the framework's broad applicability, I describe several varied forms of game balance that can be captured through restricted play.

1.5.2 Measuring Game Balance

Having established a plausible understanding of balance that supports quantitative measurement, I proceed to describe a number of projects that utilize this understanding. These projects are quite diverse, rely on differing sources of information, and in fact are suited for distinct phases of the game design process. Nonetheless, under the surface they share the common thread that they exploit the principle of restricted play.

Automated Analysis The first chapter of Part II focuses on the early-phase balancing that might be possible without any players at all. A focus on player models may seem counter-intuitive in a methodology intended to function without players. Yet if gameplay does vary by player, then insights from simulated gameplay can have little meaning without specifying the players to whom they apply. This raises the question of how we can learn about human gameplay despite the fact that for most games, no AI can play in a human manner. The existing work that *does* contextualize its simulation using player models typically provides behavioral statistics of its agents, which in many circumstances are unlikely to behave as real players do.

We address this challenge with the insight that agents that do not play like humans may nonetheless play *as well as* humans. We utilize this observation by expressing the impact of design features in terms of the win rate of specially restricted AI players, as recommended by the principle of restricted play. Such a system can be thought of as a design early-warning system. We can use automated balance analysis to exclude all but the most promising variants of a game from human playtesting, reducing the chance that human time will be wasted on a broken iteration. The validity of this idea is explored through formulations of several diverse kinds of balance, a prototype balance evaluation tool, and a case study in which we use the tool to balance an educational card game. To extend the work, I present a detailed proposal for the use of *Monte Carlo tree search* (MCTS). It is very adaptive, powerful, and in its purest form does not rely on human-built heuristics. I argue that it can make restricted play-based balancing a reality, even for real complex games. I do so by describing how to calibrate instantiations of MCTS to carefully chosen 'target players', who can then be simulated to gain early understanding of a game's balance.

Causal Models The second chapter of Part II is a case study that seeks to establish a stronger connection between data exploration and data modeling for games. This case study comes from several months of iterated data analysis and design as a technical game designer on *Playstation All-Stars Battle Royale* (*PSABR*), a multiplayer brawler for the PlayStation 3, developed by SuperBot Entertainment, and published by Sony Computer Entertainment of America. I helped guide the balancing process of *PSABR* with data analysis and mathematical modeling. As it happened, *PSABR* was a model subject for progressing the state of the art in game analytics. The game has deep complexity, through substantial asymmetry, emergent strategy, and rich game theory, yet is regular enough to facilitate validation of new methods: there is no character progression, character customization, or user-created content, so individual scenarios repeat frequently.

Like many games, *PSABR* allows for a diversity of data sources and visualizations, but it can be difficult to turn designer questions into actionable answers in the face of such diversity, when so many statistics have subtle emergent connections to any given topic. This is where data modeling comes into play. Rather than explore the relevant data in an ad hoc way, or arbitrarily limit oneself to a small set of potential visualizations, I argue that it is essential to maintain working mental models of the causal relationships between gameplay statistics. Moreover, these models should hand-designed using domain knowledge of the game, rather than being inferred automatically and risking spurious correlations. These models can then direct viewing of data, through a cascading series of causal relationships.

Player Skill Analysis Delving deeper into the analytical methods of *PSABR*, the third chapter of Part II presents a simple collection of techniques that I argue should be the cornerstone of player-centric data analysis: skill-based analysis. Measures of player skill have, for the most part, been ignored in previously published work on game data analysis. Yet I show that player skill serves as an essential axis by which to decompose gameplay data. A competitive game that does not change with skill is often one that fails fundamentally. In essence, I apply restricted play retroactively, by measuring player skill after-the-fact. Other axes of analysis are also crucial, but without decomposing them by player skill, the most important signals are often averaged out and lost.

There is certainly difficulty in utilizing player skill for analysis, the greatest problem being the inherent fuzziness of the concept of skill. Yet as with balancing itself, this does not stop us from reasoning about it and exploiting it. We are fortunate to have an approximate signal for skill readily available, through skill-based matchmaking. *PSABR*, like many competitive online games, features a skill-based matchmaking system based on Elo ratings, the classic system for rating chess players. In my work these Elo ratings serve - unconventionally - as a powerful source of analytical player information, despite their in-

accuracies. I show that Elo is vital as a value by which to plot data, filter data, and sort data. I also explore the space of alternative skill-evaluation systems and their relevance to data analysis. As a whole, this chapter serves as an example of the relevance of my thesis to concrete problems in game development.

Metagame Analysis The fourth and final chapter of Part II focuses in detail on one aspect of game balance that is often overlooked: the metagame. Metagame is as broad a concept as balance itself; it concerns the habits and tendencies of players in a game's *community*, and their effects on the low-level balance of a game. The notion of metagame puts into question even traditional notions of balance such as fairness. I explore this concept through a formalization of some of its more basic problems, redefining fairness in a new, richer context, through the principles of restricted play. Through this formalization I show how to gain new insight into a game's balance, using the fighting game *Street Fighter 4* as a concrete example. Such metagame-sensitive measures of game balance are particularly relevant for predicting the potential balance states to which a game can converge in the long term. Hence these methods are most applicable for live, released games concerned with longevity, (an ever-increasing concern as the game industry transitions from products to services).

1.6 Overview

In this thesis, I present a case for the role of computation in facilitating the creative process of game balancing, by providing designers deep insight into the complexities of their designs. By making the balancing process easier, commercial games can be made more evergreen, while the development of educational, serious, and independent games can be made more accessible than ever. Chapter 2 presents an overview understanding of balance and its consequences. Chapter 3 surveys prior work in quantitative balance analysis. Chapter 4 presents restricted play as a principled approach to quantifying some forms of game balance. Chapter 5 introduces my analytics work on *Playstation All-Stars Battle Royale* and presents a case study of the importance of modeling for data exploration. Chapter 6 presents techniques for the use of player skill in contextualizing data analysis. Chapter 7 presents a method for automatically evaluating game balance without players, and a prototype tool (with case study) for doing so, as well as a proposal for extending these methods to more complex games. Chapter 8 describes the idea of a metagame, and builds a formal model for understanding it. Chapter 9 concludes by reflecting on these contributions, presenting a vision of how these techniques may function and work together several years into the future, should they continue to mature.

Part I

CHARACTERIZING GAME BALANCE

Chapter 2

GAME BALANCING BACKGROUND

To make progress in a discussion of computation's role in the balancing process, we must first explore the ingrained methods and traditions of balancing. Attempts at understanding so fluid a phenomenon are common, with varying levels of success. I will describe the process of balancing through previous insight on balance from game designers and academics. Some of this discussion will concern the process and goals of design at large. I will also provide some broader context for game balance, discussing the unexpected positive contributions it can have on society.

2.1 Three Perspectives on Game Balance

Game balance is an amorphous topic, one whose many definitions often seem to fall at odds with one another. Yet there certainly exists common threads, which I will attempt to tease out. To do so I will summarize the views of three prominent game designers on balance. I begin with David Sirlin, a fighting game and board game designer. I move on to Jaime Griesemer, a designer of first-person shooters and action games. I finally describe the views of Jesse Schell, a designer and educator. In some sense, these designers' views are presented in order of 'breadth'. Jesse Schell has an extremely flexible view of what game balance *is*, and this is the view I have adopted for the majority of this thesis. In fact, Schell never even defines game balance as a property of games, opting instead to define it operationally. According to Schell, game balanc*ing* is "adjusting the elements of the game until they deliver the experience you want".

My own view is perhaps slightly more concrete. As that my primary interest in games is gameplay, I offer the following provisional definition of game balance:

Game balance is meaningful diversity of gameplay experiences.

This definition is intentionally vague, as it leaves unspecified the aspect of the experience to which we are referring. We may think of each aspect of the experience (be it strategy, fairness, win conditions, duration, randomness, etc.) as defining an 'axis' of game balance, often orthogonal. Meanwhile, the world 'meaningful' is itself vague; I use it to refer to the

player's choices and consequences; meaningfully diverse experiences should be noticeably different in terms of choices and outcomes, not purely aesthetics. Under this definition, a certain kind of balance is not necessarily a desirable property of a game. It is simply a property to which designer attention should be paid. These ideas will become clearer as I discuss each designer's view below, when I define formal models of game balance in Chapter 4, and in general throughout this dissertation.

David Sirlin David Sirlin is perhaps the most public figure specifically associated with competitive game balancing. Once a tournament fighting game player, he has written and spoken at length about game balance, and designed several highly asymmetric competitive games. Most notably, he designed a popular re-balanced version of *Super Street Fighter 2 Turbo* - already a famously well-balanced game - fourteen years after its initial release.

Sirlin published a handout [79] for his 2009 Game Developer's Conference talk "Balancing Multiplayer Competitive Games", which probably serves as the most succinct description of his definitions and views of game balance. He goes so far as to offer a precise definition of balance, which is more restrictive than the breadth of ways that I and others use the term, but is still very helpful.

> "A multiplayer game is balanced if a reasonably large number of options available to the player are viable – especially, but not limited to, during high-level play by expert players."

The core of this definition concerns *options*. This is clearly more broad than a single notion that a game should be 'fair'. Fairness is just one way to say that there are multiple viable options. My own understanding of balance concerns variety of play experiences. If we interpret 'player options' more loosely than Sirlin probably intends it - by including not just traditional starting and strategic choices but also choices about the pace of game, win conditions, randomness, and other more abstract qualities - then this definition does in fact gel rather well with my own.

In fact, Sirlin is careful to distinguish two varieties of balance, which I would just consider two of many. He defines

> "Viable Options (during a game): The player must have many meaningful choices throughout a game. The choices must be materially different from each other, not worthless, and not dominated by other choices. To make the game deeper, the choices should not be exactly equal in value at all times"

and

> "Fairness (options before a game starts): Players of equal skill should have an equal chance at winning even though they might start the game with different sets of options / moves / characters / resources / etc."

Some of the content of this handout is concerned specifically with games of simultaneous choice or hidden information, and thus is too specific to be mentioned here. Other content focuses more on balancing *technique* rather than characterizing balance itself. However, one key point on fairness should be mentioned. Sirlin discusses the notion of 'tiers': broad strength classes into which starting conditions can be categorized. Tiers, he says, are acceptable and mostly unavoidable. But two tiers must be stricken altogether. First the so-called 'god tier' (also known as 'S tier' in some communities) must be weakened. These are characters who are so powerful that picking them before the game begins is a near-dominant strategy. Their presence effectively removes all other characters from your game.[1] Next the 'garbage tier' should be strengthened. These are characters who are so weak as to be unviable in almost any game, who have no chance of, e.g., winning a tournament. Their weakness will not disrupt the balance of the rest of the game, but it is a waste of content and a frustration to players. Sirlin says that balancing such asymmetric starting conditions is more vital than balancing in-game choices, because players are 'locked into' these selections. I will discuss this notion in Chapter 8, but the distinction is really more of a continuum.

A final noteworthy message from Sirlin is this:

> "Game balance is so complex as to be inherently unsolvable. If it were solvable, your players will solve it and stop playing. Intuition, not math, is the best tool to navigate high-complexity problems".

Unsurprisingly, this is where my view diverges from Sirlin's. There is a nugget of truth to it: most strategically interesting games are indeed not feasible to solve exactly. Yet this in no way prevents us from applying mathematical reasoning to game balancing. This thesis presents two very different approaches for doing just that. The first approach aims to statistically measure and decompose the empirical gameplay data that a designer would normally observe anecdotally, thus improving the scale and diversity of gameplay that can be understood. The second approach is to gain early *approximate* insight into a game's balance by *approximately* solving it computationally. This description obscures much of the subtlety, but the key idea to counter Sirlin's claim is that even moderately skilled play

[1] Such a phenomenon is not just theoretical. Games such as *Super Smash Bros. Brawl* have been released with a single character who was accidentally made so powerful that tournaments which do not ban him are dominated by that single character out thirty-five. [39]

can be somewhat informative toward a game's balance. I do not claim that balance as a whole can be formalized, but rather that formal measures can deeply inform a designer's understanding of balance.

Nonetheless, in both approaches I do rely on a designer's intuition and knowledge of a game to make sense of the results and put them into action with design changes. This is all the more important due what Sirlin calls the 'slow feedback loop' of game balance. "Creating a game and seeing how its balance turns out takes years" This is because players' own views of a game develop slowly over time, and the 'true' balance of a game is always contingent on the players. I will expand on this dilemma in Chapter 8.

Jaime Griesemer Jaime Griesemer is a game designer who served as a lead on *Halo 3*, one of the most popular competitive first-person shooters of its time. In a 2010 talk at the Game Developer's Conference [44], he captured some of the intricate challenge of the balance process, anchored by the story of a single balancing decision.

In leading up to his discussion of balancing *Halo 3*'s sniper rifle, Griesemer describes many facets of the balancing process. Much as I do in Chapter 4, he treats balance as a kind of variety in high-level play, using it to refer to several different phenomena. In particular, he goes to lengths to disabuse his audience of the idea that balance is just fairness. In balanced games, each strategic element should have a distinct 'role'. Such a role specifies not just its mechanics, but also the aesthetic feeling of using it. "Roles require real differences." he says, by which he means that the value of a strategy must change with context. Much of the challenge of a game comes from estimating the payoff matrix at any moment, to know in what manner one's strategy should deviate from the uniform distribution. After all, a symmetric game is perfectly fair, but often uninteresting. As Griesemer says, "Making a game fair by making it homogenous will wreck longevity", since the variety of experiences is reduced. In fact he goes further, identifying randomness itself as an axis of balance, as we do in Chapter 4. A game should strike a "balance between predictability and randomness", where a game is too predictable if there is "only one choice" in a given situation, and too random if there is "no basis for choices".

Griesemer's talk emphasizes the importance of frequent playtesting with rapid iteration. Moreover, an insightful segment concerns the variety of players who might playtest a game, and how designers should take different forms of insight from each of them. This segment is directly in keeping with my thesis, and in particular the premise of restricted play. He defines six example kinds of players, and the most useful kind of takeaway each can provide. I list them below, though the names are altered.

- Optimizers: by 'playing to win', these players will find discover dominant or strong

strategies.

- Skeptics: these players' complaints will direct designers toward the most dissatisfying, unfair, or frustrating aspects.

- Specialists: these players will play in a particular fashion regardless of its effectiveness, thus providing an unbiased view of the strength of core components.

- Novices: bad or new players reveal difficult or confusing components.

- Griefers: just as optimizers seek the best way to win, these players seek the best way to frustrate other players, thus mapping the least pleasant edges of the design space.

- Professionals: tournament-level players tend to value predictability, and thus these players will point designers toward segments of play that feel too unpredictable.

Griesemer eventually discusses the logic that went into their balance choices for *Halo 3*'s sniper rifle. In playtesting they discovered that it was overpowered in a particularly problematic way: it subsumed other basic weapons, and players would use it nearly exclusively. They knew that they had to change the gun to stop this phenomenon, without simply giving the gun artificial weaknesses or removing its core strength. There were a large variety of variables that could be tuned to accomplish this, including reload time, clip size, zoom time, max ammo, and time between shots. In each case but one, playtesting or reason ruled one option out. Only through extensive iteration and a focused design plan is it possible to let each strategic choice operate in synergy.

Jesse Schell Jesse Schell is a game designer and educator, writer of "The Art of Game Design" [77], a critically acclaimed guide to many facets of game design. Schell's book features a prominent chapter on game balancing, which presents balance in its broadest interpretation. Indeed, the chapter is titled "The Twelve most Common Types of Game Balance", and all he says to unify them is the following: "balancing a game is nothing more than adjusting the elements of the game until they deliver the experience you want". Even though one of my contributions is a formalization of the commonalities of several forms of balancing, I feel that this broad process-oriented definition of balancing is the most productive.

Much of my formative motivation came from [77]'s balancing chapter, and several of the types of balance we capture in Chapter 4 map directly on to types from the chapter. As such, I will briefly outline all twelve of the types of balance described by Schell, in an effort

to convey the breadth of concerns designers face while balancing. Note that in most cases, Schell is careful to describe these axes of balance as features that designers should observe and control, but for which there is no right answer that fits all games.

1. Fairness: this variety concerns the advantages of asymmetrical starting conditions, as discussed previously.

2. Challenge vs. Success: this form of balance describes difficulty in the absence of other players, and as such is more pertinent to single-player and cooperative games. Still, there is a helpful analogue to our approach of framing balance in terms of win rate. Schell says "As a designer, it makes sense to ask yourself 'What percentage of players do I want to be able to complete this game?' and then design for that". We substantially generalize this sort of question in Chapter 4.

3. Meaningful Choices: this type of balance also echoes comments by Sirlin and Griesemer, that a balanced game gives players choices which themselves are interesting to make, because of various forms of measured ambiguity. The problem is that "Many designers fall into the trap of offering the player meaningless choices; for example, in a racing game, you might have 50 vehicles to choose from, but if they all drive the same way, it is like having no choice at all. Other designers fall into a different trap – offering choices that no one would want. You might offer a soldier ten guns, all different, but if one of them is clearly better than the rest, again it is like having no choice at all." The implication is that meaningful choices should be situations in which there are a variety of somewhat right answers, and the best answer is difficult to discern. I would add to that: meaningful choices are choices that impact the way you must subsequently play to be successful. As a facet of meaningful choices, Schell also describes the degree of risk available to players, and the available reward for risk. This question is probably fundamental enough to deserve its own category.

4. Skill vs. Chance: this type of balance refers to the degree of inherent randomness in a game, such as through card draws, die rolls, and random number generators. Schell does not include the randomness of probabilistic strategies in this category, but it potentially should be included.

5. Head vs. Hands: here head refers to strategy, and hands refers to dexterity. Most video games feature some element of both requirements, and striking the right balance largely depends on your audience. Oftentimes it can be difficult to assess the relative role of 'head' and 'hands'. As players' understanding of a fighting game shifts,

post-release, the perceived dependence on execution of challenging inputs can shift dramatically. A certain difficult playstyle may rise to the top, but when an easy-to-execute counter to that style is discovered, the game may shift back to a strategic focus.

6. Competition vs. Cooperation: in most games this axis is trivial to reason about - either it is a single-player game, or a team game, or a competitive game. But in free-for-all games such as *PSABR*, this can be a genuinely complex form of balance to control for. To be effective, players must fluidly form implicit, makeshift alliances with other players, in a way that seems to benefit the others but benefits themselves more in the long run. A similar process can be observed in the classic board games *Risk* and *Diplomacy*.

7. Short vs. Long: pacing is again a subjective decision. But it is not at all straightforward to evaluate; length of game can depend fundamentally on player decisions, and it may be the extent to which players can control pace that a designer wishes to change. Moreover, attention must be paid to the duration of a game following the point at which the outcome becomes clear.

8. Rewards: Schell refers to a broad variety of rewards, from aesthetic, to narrative, to mechanical. For the purposes of my discourse, rewards that concern the mechanical state of the game are more relevant. There are a variety of such rewards, including 'Prolonged Play', 'Powers', 'Resources', and 'Completion'. The question of how much of each such reward to give, and under which circumstances, directly affects both a player's experience of the game and the strategy of the game itself.

9. Punishment: this is essentially the inverse of reward, and can also be broken into aesthetic and mechanical choices. The mechanical choices Schell lists are 'Shortened Play', 'Terminated Play', 'Setback', 'Removal of Powers', and 'Resource Depletion'. This is essentially just a psychologically distinct way of framing a lack of a rewards.

10. Freedom vs. Controlled Experience: this axis is not particularly relevant to the balance of a game as I see it, as it in some sense does not impact the details of gameplay. All games have elements that are out of the player's control, but these elements can essentially be factored out when considering the choices and consequences that a player faces.

11. Simple vs. Complex: Schell identifies two distinct forms of complexity. There is 'Innate Complexity', which in effect describes the variety of actions available and the complexity of determining the immediate consequences of actions. (We could model these as branching factor of a game tree, and something like Kolmogorov Complexity of the state transition function, respectively.) Then there is 'Emergent Complexity', which concerns strategy and depth. As Schell says, many game designers explicitly value emergent complexity arising from innate complexity, but this is indeed a subjective preference.

12. Detail vs. Imagination: by this Schell means that some games tell a player everything there is to know, and others force them to use their imaginations and infer something. This axis is surprisingly important to the playability of a game. The states of a given state space can be represented in countless distinct schemes, some of which give the player substantially more information than others. For instance, one could imagine a chess state representation that also includes a statement of the optimal move; such a game is trivial. Conversely, we could create a chess game in which each board is solely represented by an integer, and a complex but computable series of rules determines the set of immediately reachable integers (child states). This game would likely be impossible for a human to play effectively (though potentially somewhat tractable for a computer). The right relationship between state and representation and the ability to formulate simple heuristics from a game's representation are paramount to a game's playability.

Although some of the above list may seem like non-sequiturs, I believe that Schell takes the right approach by listing them together. A key uniting factor is that all of them must be considered *at once*, during the balancing process. Mechanical changes affect all twelve of the above axes, and design decisions must trade off between all of them simultaneously so as to capture the intended experience. This kind of nuanced experience is exactly what balance is all about.

Common Threads What can we take away from these three diverse characterizations of balance? Most importantly, we find that it is indeed a difficult concept to pin down to a single definition. Most notably, we can see that each of the three designers' understanding of balance is broader than the last. Sirlin seems to present it as a rather similar concept to depth, which he himself defines as follows: "A multiplayer game is deep if it is still strategically interesting to play after expert players have studied and practiced it for years, decades or centuries". After all, his definition of a in-game balance begins "The player must

have many meaningful choices throughout a game.", which is surely intrinsically connected to depth as he defines it. In contrast, Griesemer identifies some features that seem to go beyond depth (like randomness), and Schell's broad classes of balance are often orthogonal to the definition given by Sirlin, and at times at odds with it! Schell certainly describes 'meaningful choices' as a positive thing overall, but nonetheless treats it as a dimension that can distinguish equally legitimate games. In other words, Sirlin says that a game is balanced if it has a large value on a certain axis; Schell simply identifies this axis as one that designers might control for when balancing, along with eleven others!

What is to explain for this difference in perspective between two eminent game designers? I claim that it is largely a cultural difference, and that Sirlin's view reflects a more specialized view of the space of games. As a one-time tournament chess and fighting game player, Sirlin's values for game design align directly with strategy and *mastery*. To Sirlin, a good game is one that is challenging and engaging, through which players can grow and learn for a long time. This leads him naturally to competitive games with strategy, including his own design. He has spoken out publicly against games like *World of Warcraft*, which he claims incentivize players to spend meaningless time improving their character, rather than challenging effort improving themselves.

Schell, on the other hand, has designed more varied games, including a pirate ship simulator to be played at Walt Disney World. A game in that context must first and foremost engage *new* players, giving them a burst of visceral fun and excitement, with a secondary objective being replayability and depth. Schell has no value judgment against games in all their forms, and the many axes of balancing he describes reflect this. In truth, when any designer balances a game, these axes must be considered, to shape the game to the intended experience. It is just that when designing the sort of game Sirlin is interested in, many of these axes are already prescribed. The form of the experience is implicit, and subject to that form, the primary balancing goal is indeed to create depth. Similarly, Griesemer is somewhat more prescriptive than Schell, but as a designer of competitive games that are explicitly intended to be more accessible than fighting games, Griesemer is somewhat more open-minded about depth than Sirlin.

Throughout this thesis, I will explicitly avoid taking a normative perspective on games, instead choosing to align with Schell's view that the goal of balancing is to create a certain experience: whichever experience the designer intends. That said, Sirlin's perspective is still paramount. Depth is often the most *difficult* form of balance to achieve, and thus deserves the lion's share of the attention. Many of the methods I discuss will be focused particularly on depth, both within a game and during selection of asymmetric starting conditions. This leads me to talk mostly about competitive games, the class in which depth most easily

emerges, due to the adversarial escalation of multiple motivated humans. I nonetheless will attempt to do while maintaining attention to the variety of other concerns a designer must address during the balancing process.

2.2 Game Balance and Society

The role of balance is most paramount in the context of competitive games, which will be a major theme of this thesis. Here I will briefly discuss some of the positive societal outcome of work on balancing competitive games. The connection between competitive gaming and serious games might seem thin at first blush. The demographics of competitive games currently skew toward young adult males who have played major commercial games for years. This is exactly the audience that serious games tend to overlook, favoring instead players with fewer preconceptions about games' functions and purposes. I believe this disconnect is likely to fade. When effective crafted and supported, competitive games engage the mind and will in a way done by few others. They have the potential to reap benefits for the players themselves, or when integrated with real problems, for society at large. What's more, the somewhat narrow audience of competitive games has the potential to expand over the coming years, just as the reach of sports have across gender and age. Competitive gaming communities are beginning to recognize their sometimes hostile environments, and are taking steps to correct them [11]. Such steps may be inevitable, due to the economic inviability of ignore such a large potential audience.

In past centuries, sports have served a purpose beyond recreation: they act as a motivational framework for maintaining physical health, teaching teamwork and dedication, and building the prowess and skills necessary for the labor market. In a growing information economy, where too few individuals are trained with the information skills for many modern jobs, competitive video games can serve a parallel role. I discuss a few such phenomena below.

2.2.1 Fold.It and Competitive Science

The value of competition in encouraging excellence may not be well understood, but we certainly have reason to believe in it. Performance in many sports events strictly improves year-over-year (albeit toward a limit) [34], and it is true that advances in health and training partially explain this phenomenon. However, it is widely believed that competition with rivals also fuels this rise, partially due to the existence proof that certain feats are possible. Competition is a powerful force, and leveraging it effectively seems to have the potential to create societal benefit, if individuals can be properly motivated. When economic profit

is the goal, we see competition driving higher quality products and services, to the extent that many anti-competitive practices are illegal. However, not all desirable ends can be effectively monetized, either for practical, psychological or budgetary reasons.

Games offer the possibility of motivated competition toward a goal, without any intrinsic reward on the line. The Center for Game Science's *Foldit* may be the best example of this potential. *Foldit* is a video game designed with the express purpose of crowdsourcing scientific work. Players manipulate a three-dimensional representations of real proteins in a puzzle-game-like framework, attempting to find stable configurations. The hope is that human pattern-recognition and problem-solving might facilitate better solutions than even supercomputers can, and in multiple cases this had already proven a reality. In one notable case [40], *Foldit* players solved a years-standing open problem seen as a potential building block in the fight against AIDS - a problem which has already received significant computational resources.

Crucial to *Foldit*'s success has been its reliance on a thriving community of players who actively communicate, collaborate, and compete with another. Players can play either as an individual or in a team, according to their preference. Leaderboards display each player or team's level of progress on current problems, thus motivating and encouraging other players. Players in a team often collaborate to share strategies, and even build macros to capture effective strategies in a single action. What's more, players sometimes share these macros and other insight publicly in *Foldit*'s forums. Despite each player's motivation to win, there is a complex manifold of collaboration and competition taking place, as discussed in [52]. The authors observe that a large breakthrough by a single team on one puzzle set off 'a chain reaction' of improvements by other teams. This may partially be explained by information-sharing online, but is almost certainly also a function of the motivation of competition. It seems unlikely that *Foldit* could succeed without the interaction between players that emerges around it.

It is clear that some players do play *Foldit* for due to an extrinsic motivation: the desire to contribute to science, to learn something, or even to potentially help a sick loved one. However, we have every reason to believe that game-based competition serves as an excellent motivator even without such clear extrinsic benefits. In my own experience with the fighting game community, it is remarkable how many players take time to share their knowledge and insight with other players. Web sites like *shoryuken.com* publish several user-created videos every day (among news articles and features), each a laboriously built, polished work that serves to teach other players, analyze gameplay, or even scientifically explore the properties of the game system. Competition is the engine that drives this activity - it gives purpose to this work, as many players delight in influencing the most competitive players, or bringing

up new players, even if they themselves do not compete.

The remarkable thing about this interplay of competition and collaboration is that it may be an effective way of harnessing human problem-solving power in an agglomerative way. Within fighting games, this can be seen in the ubiquitous 'combo video'. Some explanation is in order. Nearly all fighting games share a feature wherein certain attacks can be chained into other attacks in a way which - due to the emergent dynamics of the combat system - prevent opponents from reacting between the attacks. Depending on a game's basic systems, these combos can be extremely elaborate and subtle, requiring players to execute dozens of actions in a row, at a precision of 1/30th or 1/60th of a second, in an order that is in no way suggested by the game. Such combos are sometimes essential for effective gameplay, but are also significant sources of pride for many players. This explains the prevalence of 'combo videos', in which players show off what can only be described as 'discoveries' (or even 'inventions'), often building off principles from other players and previous videos.

What has yet gone unnoticed is the potential, however remote, of harnessing the practice of combo discovery for problem solving. Given the variety and complexity of combat systems in fighting games, it would be extremely unlikely that some such system (or a near variant) could not be shown to be NP-Hard. By this I mean that it should be possible to encode a given instance of, say, the Maximum 3-SAT problem as a fixed character in a fighting game, whereby each combo for that character maps to a partial solution to the SAT problem. The existence of a sufficiently long (or damaging) combo could then provide a solution of unprecedented value to this instance. Admittedly, such a plan may not be practical, for three reasons. The first reason is that an entire character would have to be generated just for the solution of a single problem, although if this instance is an open problem in, for example, software verification, it may be worth it. The second reason is that a character designed via an arbitrary reduction is unlikely to be fun to play, not to mention easy to implement. However, it is possible that manual adjustment of such a character could result in only a somewhat decreased chance of combos mapping to true solutions to the problem instance. The third reason is the most damning: it seems unlikely that one could design a gadget that reduces genuinely NP-Hard problem instances to characters who are not impossibly complex to play. Nonetheless, this is a thought experiment worth further investigation, as some variant of a fighting game at the least may prove practical. After all, identifying the kinds of games that already support thriving competitive cultures may be the most effective way to develop games that solve problems.

2.2.2 Learning Science and Engineering Fundamentals

The complexities of game balance that are discussed in this thesis are not solely the purview of game designers. As players of any game become more competitive, it becomes more important for them to understand the deeper subtleties of the game system, and this work can sometimes be as challenging as a designer's work. The player's practice of analytically and experimentally maximizing their own effectiveness is known as *theorycrafting*, and it has become commonplace throughout several game communities, such as *Starcraft 2*, *World of Warcraft*, *League of Legends*, and more. I will speak mostly of *World of Warcraft* in this subsection, as it has a particularly thriving theorycrafting community, partially as a function of its genre. Some broad discussion of the role theorycrafting is playing in changing games can be found in [71].

Theorycrafting generally takes two forms, though the lines between them are blurry. One of these forms serves as a parallel to engineering in the real world, where players attempt to engineer strategic solutions that maximize key gameplay statistics. The other parallels natural science, where players experiment with the basic mechanics of an opaque game world. In the best case, these practices can create a context in which science and engineering are accessible and relevant to players, potentially building broader interest or even skills in STEM. One prominent research paper [83] examined the discussions on a *World of Warcraft* forum, and categorized them according to various scientific behaviors, finding that the majority of posts were in some sense doing real participatory science work. Another paper [6] discusses the basic forms of economics embedded within *World of Warcraft*, and the potential for such games to teach players understanding of economic concepts and practices. In this section I will explore the behaviors of engineering and science in *World of Warcraft* in some depth, including major barriers. Many of the principles I discuss are relevant across a variety of games; the very term 'theorycraft' comes from an earlier game of a different genre, *Starcraft*. However, massively multiplayer online games such as *World of Warcraft* probably offer more rich opportunities for theorycraft than any other genre, with their huge variety of atomic actions, items, states, characters, and environments. While reading this discussion of theorycrafting, consider the role that balance plays in facilitating the possibility of theorycrafting. A game that does not feature a diverse variety of experience, for which choices are not simultaneously meaningful and difficult, is a game in which theorycrafting is either impossible or irrelevant.

Engineering The first form of theorycrafting aims to identify concrete but high-level strategic rules for maximizing player success, or some proxy for player success. In some sense, this practice is as old as games themselves; books on *Chess* strategy can be seen as

an early form of codified theorycraft. Yet there is something different about modern games - certain video games in particular - that makes this process somewhat more concrete and accessible. Unlike board games, video games automatically handle the bookkeeping of updating variables and evaluating rule consequences. This fact allows for the creation of more rich and quantitative rule sets than would typically be seen in non-digital games (with partial exceptions such as tabletop role-playing games and war-games.). Quantitatively complex games often have the feature that player strategy can be approximately evaluated through a hierarchical decomposition of statistics. In other words, such games often have more accurate quantitative heuristics for success, and these statistics themselves have good heuristics for predicting them, and so on. Without such heuristics, these statistics could be quite overwhelming, in contrast to a game like *Go*, which has weak heuristics but inherent rule simplicity. I will describe the idea of statistic decompositions in detail in Chapter 6, and show its utility for balancing *PSABR*. However, for the time being, I will illustrate it though a brief example in *World of Warcraft*.

In *World of Warcraft* (and many other MMOs) players participate in a variety of disparate challenges, however one form known as a 'raid' occupies much of the time of most advanced players. Raids are challenging dungeons for groups of up to 25 players. They typically feature difficult boss fights, in which players must plan ahead, coordinate, and play to their characters' specific strengths in order to prevail. Much of the work done by theorycrafters is focused on maximizing a character's effectiveness in raid-like contexts. Each raid is different, as is each character class, hence it can be difficult to meaningfully measure a player's effectiveness to winning or losing, particularly since raids are time-consuming and so repeated trials are not very feasible. Instead, many theorycrafters rely on the proxy of 'damage per second' DPS output for estimating a character's effectiveness. For certain classes (such as healers) this metric is clearly uninformative. However, for other classes DPS is used as a metric very broadly, despite capturing only the most important single feature of that class's contribution to the battle. The preponderance of this metric is not surprising: it provides a single number by which choices can be evaluated, experiences can be compared, and even balance patches can be distinguished. That said, when reasoning about DPS players do often speak about the tradeoffs to other statistics in the process.

Determining a player's DPS is a very complex process, as it is a function of several factors: that player's 'build' (abilities and items that have been acquired over time), the sequence and timing of abilities used by that player, and the context (teammates, enemies, stage) in which play occurs. Nonetheless, theorycrafters have amassed an impressive collection of techniques for estimating DPS. DPS can be estimated in four ways, at varying levels of fidelity.

- For marginal accuracy, players can utilize a system of formulas that explain how the game's mechanics determine the relationship between the many variables impacting DPS. These calculations are sometimes built into spreadsheets, which are provided to less enterprising players so that they can understand their own character's potential in various contexts. Unfortunately, such calculations are always significant simplifications, as they require idealized models of the many emergent components in gameplay.

- A second level of DPS evaluation seeks to overcome the many simplified assumptions of collections of formulae by performing repeated monte carlo simulations, in which all players, enemies, and statistics are modeled individually, and updated over time. Simulation engines (such as EhnSim and SimulationCraft) are made available as open-source software, and updated regularly to match the newest knowledge of the calculations inherent in *World of Warcraft*. They are treated as very reliable sources of knowledge by many players, but still rely on models of the game's code and the actions of players and enemies. As such there is a fundamental disconnect from the realities of gameplay.

- For more reliable estimates, players sometimes turn to in-game experiments. The developers of *World of Warcraft* include a 'target dummy' in player cities for just this cause. The target dummy acts as an inert enemy on which players can try out different builds and strategies, evaluating their damage and other effects. The game client provides some visual quantitative feedback on players' performance, but many players use client modifications that act as makeshift telemetry systems. Such software instruments the *World of Warcraft* client where possible, and records key statistics such as damage as they are updated. This data can then be collected in various ways, shared, and analyzed in order to support further optimization of DPS. Unfortunately, target dummies themselves provide a simplified environment for analysis, despite their accuracy; other players are typically not there to influence the outcome, nor are the effects of environment or enemy actions consider.

- The data that is arguably the highest fidelity is logs from real raids. Using the same instrumentation tools, groups of players record their every action and outcome. Online tools such as World of Logs (http://www.worldoflogs.com/) then support easy visualization and analysis of these logs, for individuals or groups. This allows players to optimize their own performance, but also to synthesize high-level guidelines for effectiveness in practice, outside of any simulating assumptions. DPS itself is often evaluated in these contexts, and can vary from the data of the other three methods.

That said, this method has plenty of its own downsides. Data collection is time-consuming. Controlled experiments are rare, as players who take the time to raid typically do so to win, not solely to learn. Moreover, player error is always present, and can serve as a source of systematic bias that is hard to measure.

The above four techniques constitute the primary tools of the trade of theorycrafters who seek to optimize play. Note that these techniques only describe one facet of the optimization process: evaluation. The optimization itself can be quite difficult, and much of it is performed by hand or intuitively. The many features of gameplay make computational optimization somewhat prohibitive. In this sense, players are practicing genuine engineering. They operate in an environment of relatively well-understood rules (analogous to physics), use their own ingenuity to find solutions that accomplish a certain goal, and use simulation and experimentation to evaluate carefully chosen metrics of practical success. One would be hard-pressed to argue that those who practice theorycraft are not learning or employing genuine engineering skills that might transfer to the outside world. Perhaps the greatest critique of this argument is that theorycrafting may be a practice of the elite in games, and may remain very inaccessible to the average player - that same player who might struggle or lack motivation in engineering classes. Learning the truth of this critique will take time and study. We do know one thing: the work of theorycrafters spreads broadly through the player community, and players of all stripes are known to use their insights. The real question is the extent to which players understand the results they are using, let alone the methods of their derivation.

Science All the theorycraft work I have discussed thus far focused on problem solving, and thus serves as a metaphor for engineering in the real world. Yet there is another form of theorycraft made uniquely possible (or necessary, rather) in games such as *World of Warcraft*. Blizzard, the developer of *World of Warcraft*, does not publish a rulebook in which its many mechanics are precisely detailed. Nor are these mechanics fully described in the game, on a website, or in any other context. In many ways, players must approach the world of *World of Warcraft* as a black box - one with its own rules, phenomena, and cause and effect - essentially a distinct physics. Some basic mechanics are described explicitly, and others match intuition so blatantly as to be self-evident. Yet a player seeking to perform engineering of the form I have discussed finds herself in an interesting situation. To make progress in engineering, she must first construct a deep, accurate model of the world itself. Experiments are necessary here, not to evaluate the emergent properties of certain mechanics, but to evaluate the mechanics themselves. Here no simulated model is useful; all that matters is data, as the core mechanics are not derived from anywhere. They are

basic atoms.

Quite appropriately, we can think of this work as a form of scientific process. Players live in the world, from their observations and goals they form questions and hypotheses, and to test these hypotheses they devise experiments. This is not to say that the mechanics of *World of Warcraft* begin to approach the complexity of those of the physical world. What's more, observation and experimentation are both made substantially easier through the presence of logging and automation. Nonetheless, players are engaging in the same kinds of reasoning as early physical scientists. If anything, it is the presence of low-hanging fruit of knowledge that makes the environment an interesting one for learning. Even in the best experiential science classes, students are aware that their work is merely an exercise - they are rediscovering long-known facts, and their findings will have no outcome in the world. In contrast, theorycrafters experience the joy of discovering 'new' knowledge and sharing it with an invested community. Because these virtual worlds are ever-changing (and new ones continue to come into existence) there is a never-ending source of real problems of discovery. These problems fit the skills of any participant, unlike natural sciences in which years of expertise and costly tools are typically necessary to contribute.

It should be noted that the 'scientific community' of theorycraft is still in its infancy, at least for *World of Warcraft*. Hearsay and appeal to authority still play a tremendous role. A single sub-community called 'Elitist Jerks' (http://elitistjerks.com) dominates the discussion. These theorycrafters post lengthy, detailed FAQs in which every facet of mechanics and strategy is explored. Yet claims are infrequently substantiated by references to explicit data. Anecdotally, it seems that most claims in these FAQs are indeed derived from experiments, yet these experiments may be quite ad hoc, and players asking for concrete evidence are sometimes shunned. An extremely simple example can be found in the FAQ for the hunter character class in the *Mists of Pandaria* expansion [73].

> "Q) Does mastery round down? A) No. Although the tooltip in game makes it look like mastery gets rounded down to whole points, testing indicates that you do get the full value of the mastery, so 16.99 mastery is definitely better than 16.01 mastery for instance."

The question here is simple: a certain statistic is reported in rounded form in some parts of the game's user interface, suggesting that the statistic's effect may be based on its rounded value. The answer is simple as well: 'testing' indicates that this is not the case. Yet no reference to specific tests is given, forcing the reader to rely on faith in the moderators of the forum. Threads addressing this concern (about this single mechanic out of thousands) can be found on multiple forums across the internet. In them, players are

berated for questioning the claim, sometimes being told that "it is well-known". Other negative responses [54] shed a little more light:

> "The fact that it is so impossibly easy to test with many classes and many specs makes it very obvious. When you can definitively test something in less than 5 minutes the only people questioning it are those who don't actually understand game mechanics. Thus why it's not a hot topic on [Elitist Jerks]. They have pretty strict rules about posting about things of that nature."

Indeed, one poster in that thread eventually gives in and runs an experiment to justify the claim, providing the limited numbers for at least one example.

> "Way easier to test than you're making out. Get a Light Crossbow and remove all proc based items. Alter Mastery without going up or down more than 1%, repeat test. I did this by switching 67 Mastery. My Cobra Shot went from 5,773 to 5,753. This is going from 693 to 626 Mastery Rating, or 11.87 to 11.49. This is almost exactly the +.38% you would expect from that gain in Mastery."

In general, it seems that in the major forums more complex questions might have their answers better justified. Indeed, the details of experiments are often published on Elitist Jerks, and serve as a source of pride. However, these results are often then taken for granted, and go uncited in future references to their them. This level of unprofessionalism is somewhat disheartening, but is also not surprising. There are no formalized institutions, and no funding agencies to regulate behavior. Players participate in theorycraft solely for the enjoyment, intellectual stimulation, pride, and possibility to improve their play.

Some help - though also some danger - may come from the designers of the games themselves. Whether originally intended or not, the designers of *World of Warcraft* have come to recognize that the obscurity of the game's mechanics actually drives some its appeal. Just as exploration of the landscape is a key feature of the game, they see exploration of the system's affordances as an enjoyable feature to be supported for a different subset of the players. In one notable forum thread a lead designer named Greg Street (known as Ghostcrawler) elaborates on this idea [32].

A player had posted claiming that the values for a statistic known as 'armor penetration' were being misreported by the client, because their total damage dealt was not as expected. After some deliberation, Ghostcrawler posted a lengthy explanation of the game's formula for calculating damage as a function of armor penetration. This formula turned out to

be substantially more complex than thought by the player community. The formula involved a *min* function in an intermediate calculation, creating a nonlinearity that made effective backward inference from data very difficult. The players could tell that they might have taken a long time to derive this rule, and appreciated the contribution, but also felt somewhat disempowered. One poster commented

> "Uhh... [Ghostcrawler] came in and stomped all over my last 5 hours of testing against a paladin in [Player versus Player] by showing quite a few assumptions I'd thought were safe were wrong."

In response, Ghostcrawler made clear the design team's opinion on their ideal role in theorycrafting.

> "This is honestly one of the reasons we don't do this more often. There is a risk players will stop experimenting and theorycrafting if they think we will eventually just dump all of the answers on them. We like for players to experiment with gear, talents and the like. Having black boxes adds depth and a sense of exploration to the game. When everything is known with certainty, you can do things like definitively know the best choice in every situation. Theorycrafting is dead. However, in this case, the system was both complicated and difficult to test so we figured we'd just open the kimono."

It is apparent that the relationship between designers and players is becoming increasingly difficult to navigate as games transition from boxed products to live services with active communities. Designers build challenges into their game by definition, and the right time to step in and assist, be it with information or a patch, is extremely ambiguous. In this case, it seems that the goal of *World of Warcraft* designers is to design a game that, wherever possible, is feasible to reverse-engineer. If they perform this job well, treating scientific discovery as a game itself, they can build a system that encourages strong community and practices to form. When they fail and must step in like God with a tablet of physical laws, this has the danger of making the entire enterprise of theorycraft feel futile or manipulative. This removes a form of entertainment from their game (albeit a divisive one). It also, as Ghostcrawler points out, removes a valuable source of balance information, a sort of free crowdsourced labor from players, who can explore the game to a much greater depth than the designers themselves can. These issues will no doubt become more common in coming years, as player dedication continues to increase. Hopefully it will move toward an environment that embodies the best principles of scientific research and motivates - if not trains - future scientists.

2.2.3 Competitive Gameplay Analysis as a Testing Ground for Economic and Government Analysis

Chess acted as a focusing lens for artificial intelligence in its infancy and beyond. In a similar way, I believe analysis of game design has the potential to advance the methods of data science as a whole, particularly toward behavioral fields such as economics, sociology, and political science. Game design offers a strong parallel to these fields: a designer attempts to build a set of simple mechanisms and rules that, when engaged by a group of humans, generate an emergent experience with certain desirable properties. Yet games simplify this process tremendously. Instrumentation can be near-perfect, allowing a far richer source of ground truth than any real-world system. Perhaps more importantly, competitive games have a unique benefit: player *utility* is largely known. Even if we discount irrationality or limited rationality, real-world data analysis is necessarily confused by the multitude of goals and preferences underlying each human's decisions. In competitive games, with occasional known exceptions, it is safe to assume that each player has a single, simple, common goal: to win.[2] This assumption makes the process of reasoning about players' decisions, knowledge, and reasoning immeasurably easier. By limiting our unknowns from two to one (player ability), we can build out analytical tools and better understand their usefulness. This puts us in a better position to start from when extending these tools to real-world circumstances with greater fuzziness and further unknowns.

.

[2]The greatest exception is a preference for certain play-styles or game features. However, these preferences can often be discovered through the data, and normalized out. What's more, such preferences become decreasingly relevant with increased skill, as players focus further on winning as the singular goal.

Chapter 3

RELATED WORK IN QUANTITATIVE BALANCE ANALYSIS

Having discussed several views of balance itself, it is time to turn to related work in quantitative balance analysis. Most of this work is somewhat new, as it builds both on new ideas about game design and new innovations in data mining, visualization, and artificial intelligence. The goals of the work are similar to mine. However, much of it omits the player-centric perspective that I put forward in this thesis, and in several areas go into lesser depth. The work generally fits into one of the two categories into which I have separated quantitative balance analysis: instrumented gameplay and automated analysis. I will look at each of these categories in turn.

3.1 Instrumented Gameplay

To the present day, nearly all balancing insights come from evaluation of human play. , be it from designers, internal playtests, or public betas and released games. In recent years, this process has evolved substantially, moving from ad-hoc observation to rich quantitative analysis of heavily-instrumented play. Companies including Bioware [98], EA [64], and Valve [2] record detailed events as players play at home or in the studio, from which they derive design and implementation insights. Microsoft Game Studios' user research group has run countless focus tests over ten years, with over 100,000 players participating [58]. Even small independent development studios have found it practical and useful to instrument their games, some single-handedly [72], and some using the libraries of distribution platforms such as Playtomic and MochiBot. In the extreme case, Zynga has based their primary design strategy on the analysis of millions of players' actions and thousands of A/B tests [76] [85]. This is to say nothing of the many ways in which players themselves consume gameplay data [63].

In this section I give an overview of research in modern gameplay capture and analysis techniques, with an emphasis on the methods' use for balancing. The formalization of these processes has had the added benefit of making the process more communicable, which will enable us to aim our balancing work in the most practical and beneficial direction.

One of the best-documented examples of instrumented gameplay analysis is Microsoft Game Studios' Tracking Real-Time User Experience (TRUE) [53]. It involves a collection of tools, so may be more appropriate to think of as a methodology than a concrete system. [53] describes two typical deployment conditions through two use cases. To evaluate the single-player campaign of *Halo 2*, Microsoft ran extensive in-house user studies. 44 players played through the entire campaign (10-12 hours) in two groups. Each of these play-throughs was instrumented to the fullest extent possible.

- Video: All gameplay visuals and audio were recorded, timestamped, and indexed by level, allowing designers to easily view whole sessions or drill down to specific segments of interest.

- Attitudinal Surveys: Gameplay was paused every three minutes, so that players could respond to a single-question digital survey, asking about their perception of the current difficulty.

- Gameplay Telemetry: Detailed events were logged by the game, statistically tracking each player's actions, and their effects on the game world.

In contrast, to evaluate the beta version of *Shadowrun*, Microsoft used only gameplay telemetry. The tradeoff of eschewing video and surveys is the scale increase it affords: the *Shadowrun* beta was played by 10,000 players in their own homes for months, and all of this gameplay was collected and analyzed. Video would typically be impractical due to bandwidth constraints, and surveys would likely be onerous to players who are playing recreationally, unlike the paid participants in a small-scale user study.

Tychsen and Canossa [91] give a related decomposition of the forms of playtesting: "metrics analysis can inform how the user is playing the game, where playability testing can inform if the user has a good experience, and usability testing can inform if the user can operate the controls".

In my own work at SuperBot Entertainment, we employed all of the above methods, but our primary use of external players was telemetry. The scale this allows is of particular importance to competitive games, as I describe in Chapter 6. Thus I will focus primarily on work with telemetry data.

Research in the use of player telemetry data partitions naturally into Visualization and Data Exploration, and Data Mining. I will look at each of these topics in turn, emphasizing the first topic, as the most immediately practical source of value, and the closest to my work at SuperBot.

Visualization and Data Exploration Aggregate visualization remains the most accessible method of exploiting telemetry data. Visualizations give designers an opportunity to consume data in terms they can relate to: individual maps, characters, time periods, etc. Indeed, a large fraction of gameplay visualization work concerns the spatial mapping of player movement and actions. Early work by Hoobler et al. [47] was built for spectators, rather than designers. Nonetheless, the objective is the same: expanding a designer's view of gameplay. They present a real-time visualization of a first-person shooter. Player movements and actions are overlaid on a top-down view of the map, with visual encodings of character information, movement paths, firing paths, or visibility. They also create early examples of the now-ubiquitous gameplay heat map, showing intensity of player location and actions on the map. Moura et al. [65] also build maps to support viewing entire games, though here the focus is on aggregating over time. They plot a single player's entire progression through a large map in the role-playing game *Dragon Age: Origins*, using color-coded iconography to indicate interactions with level features, characters spoken to, etc.

In contrast to [47] and [65], most telemetry visualizations extend a designer's perception from a single game to many games. Chittaro et al. [21] present an extension of [47] that can aggregate behavior over many games, using heatmaps of time spent, but also of richer measures like variety of players in an area; time spent looking at an area; degree of simultaneous travel through an area; and average direction through an area (using vector imagery). Gagné et al. [37] build a spatial visualization for analyzing a real-time strategy game. They point out this genre's unique needs: players control not a single avatar, but dozens of units. Nonetheless, their solutions closely resemble the other work in this section. Drachen and Canossa [29] examine the use of Geographic Information Systems for visualizing spatial gameplay data. These are off-the-shelf visualization systems typically used for real-world location data. Drachen and Canossa show that such systems (although expensive) can easily generate similar visualizations to those of the above papers, and support arbitrary on-the-fly weighting of semantic annotations. The most canonical instance of spatial gameplay analysis are the death heatmaps for the *Halo* series. Romero's 2008 GDC talk [75] and [53] show examples of these heatmaps. They are used in drill-downs, where designers can click on charts of deaths per level to see a heatmap for that level, and click on individual locations to see videos of those deaths.

Interestingly, [37] points to a limitation in spatial mapping work: it only applies to games in which players control an avatar situated in a spatial environment. [37] extends this slightly to games involving multiple units, but the fact remains that many games (most puzzle games, card games, simulation games, and more) do not involve navigating a virtual environment at all. Andersen et al. [3] overcome this limitation by visualizing

player movement not through a virtual environment, but through the game's *state space*. Their tool, Playtracer, visualizes many plays through a game by mapping the states onto a 2d plane using Multidimensional Scaling, and visually emphasizing the most-visited nodes and edges. Playtracer is used to 'debug' the level designs of *Refraction*, an educational puzzle game, by finding popular misconceptions in players who failed, for example. This substantially extends the applicability of spatial mapping work, as nearly every game can be thought of as a state machine. Unfortunately, these state machines can be very complex, which Liu et al. [59] resolves using simplified projections of games' actual state spaces. They show how - with designer direction - traces from three distinct complex games can be viewed effectively with Playtracer. Wallner and Kriglstein [94] present a similar extension to Playtracer. They require a distance metric on states, which is used to cluster the state space, and thusly collapse it. In abstract games, the tool acts much as [59]; in spatial games, they cluster on the game's map, making the visualization a subtle variation on the traditional heatmap. In Chapter 6, I will show how we can move yet forward, beyond games in which state space is a useful abstraction.

Bowman et al. [13] present a broad taxonomy of game visualization work, which covers many possible uses and forms. Some of their discussion concerns non-spatial analyses, but none address the player-contextualized analyses I discuss in Chapter 6. Perhaps the existing work with the most non-spatial analysis is Medler et al.'s Data Cracker [64], a custom-built analytics tool for visualizing telemetry data from *Dead Space 2*'s team multiplayer mode. Medler's PhD dissertation [63] is also an excellent source of information on game data visualizations, though its focus is on the uses of said visualizations by players, rather than developers. An emphasis of Data Cracker is to support analytics at multiple levels of aggregation, to address the needs of varying developers. They may show hours played per day for general use, hours played per region for production, and hours played per character for design. They also argue heavily for the use of time as a partitioning feature of data.[1]

Data Mining A number of papers apply machine learning techniques to telemetry data. Some of these papers utilize this data to predict player's in-game actions, (notably predicting high-level strategy in Starcraft [96] [86] and *Civilization IV* [60]) and are not germane to this thesis. More relevant is work that predict players retention from gameplay. Nozhin describes his efforts on the Massively-Multiplayer Online RPG *Aion* to predict the day that new players would play their last game [69], and the week that veteran players would play their first low-activity week [70]. These models were used primarily for marketing. In

[1] In contrast, we have found that gameplay data (within a given build) tends to be somewhat static or change arbitrarily over time, due to the changing user base as new players enter the system.

contrast, Weber et al [95] found gameplay features in *Madden NFL 2011* that are predictive of long play times, such as play variety and win/loss ratio, and made recommendations to the designers for future iterations. Bauckchage et al. [8] present results derived from several distinct games. They find that five action games had player retention distributions that were well-fit by a Weibull distribution, and they propose this as a general model.

I see promise in systems that integrate data mining with visualization. Data mining can facilitate the selection of meaningful features, and visualization can present the data 'directly' to designers, allowing them to memorably observe simple patterns themselves. Ferreira de Oliveira and Levkowitz [36] survey such integrated processes outside of games. The only games research I am aware of in this vein seeks to cluster players according to their play-styles.[2] [31], [30], and [88] offer treatments of player clustering, across four genres of games, utilizing Multi-Dimensional Scaling, Self-Organizing Maps, and Simplex Volume Maximization, respectively. This work is not yet well-supported, however. Establishing ground truth for unsupervised learning is difficult; what's more, none of these papers describe actionable design insights that were utilized or even proposed. Clusters can be a reductionist mechanism for describing people. When possible, I prefer to view players directly as points in a high-dimensional space, displaying game statistics on a given axis at a time.

One anomalous paper is Normoyle et al's [68] use of machine learning in the data *collection* process. They argue that subjecting players to fixed gameplay encounters may not be the most efficient way to collect data, and propose an adaptive system that chooses levels for players to play according to the features for which we are the most uncertain. Unfortunately, this approach is somewhat impractical: any playtest featuring enough players for such a system is unlikely to be paid, and in such playtests a high priority is placed on providing an enjoyable experience for the players, rather than one which simply gathers the most useful data.

3.2 Automated Analysis

In the past five years, research has begun on helping designers evaluate (and often modify) their games without human playtests. Some projects share with my work the analysis of traces of explicitly-restricted simulated agents. However, none of them note the generality of this approach, nor exploit it to measure multiple balance terms in comparable quantitative terms, as we do.

---

[2]Much of this work is inspired by Bartle's seminal article "Hearts, clubs, diamonds, spades: Players who suit MUDs" [7], which anecdotally presents a taxonomy of player types in Multi-User Dungeons and beyond.

A Requirements Analysis. Some of the ideas developed in my work are proposed and discussed by Nelson and Mateas [67]. This paper is an exploratory look at the possible forms of a design tool for games. The paper takes two approaches to explore this space: they survey the history of traditional Computer-Aided Design tools, and they interview three game designers about their process and needs. This interview process is supplemented by interactions with partial prototypes, to help the designers understand what kinds of tools might be possible.

Hunicke et al. [48] partition a game's core elements into *mechanics*, *dynamics*, and *aesthetics*. Mechanics are the concrete rules defining individual state transitions and legal moves, whereas dynamics are the high-level emergent processes of these mechanics. Our work and [67] are both concerned primarily with elucidating the degree to which various dynamics play a role in gameplay.

The goals of our work and [67] can be further specified, using a CAD taxonomy described by [57]. This taxonomy divides the roles of a CAD system into five categories: *learner, informer, critic, collaborator, and initiator*. Of these, the roles of learner, collaborator, and initiator are most dynamic, and require the most agency on the part of the tool. Indeed, the history of CAD tools is littered with highly ambitious projects, which seek to make the computer an equal partner in the design process, making proposals, learning about the designer, evaluating her processes, and even communicating with natural language. In contrast, our work is much less ambitious, and intentionally so. We choose to iterate on a single piece of functionality, to ensure that it is accurate, meaningful, and easy to understand. Namely, our system should evaluate balance measures of an existing game design, quantitatively. This function is an example of the informer role, which is concerned with answering designer questions. However, it is more accurate to characterize our system by the critic role, which is concerned with *finding* flaws in a design. We will see that our system attempts to succeed through quantity, rather than quality, of metrics. It is best to think of the tool as providing a bevy of design 'sanity checks', which each iteration will be subjected to, with extreme values being surfaced to the designer.

[67] also finds informer and critic to be the most natural roles for a game design tool, largely due to their interactions with game designers. The designers suggested a number of features such a tool could have, most of which match these roles, and indeed the goals of our own project. It is not clear to what extent these suggestions came from priming by the authors, who freely admit to a very two-way interview process. Nonetheless, it is reassuring that designers would find such tools interesting. For the most part, the designers found the most potential in the late phase of the design process, when the high-level mechanics are crystallized in a working prototyping, and the remaining concern is 'design debugging'.

They wished to ask questions about topics such as: the relative usefulness of each unit; the existence of dominant strategies; the convergence of seemingly distinct strategies; the performance of various play-styles. Although the objectives set forth by [67] greatly resemble ours, the approach differs substantially. They propose to answer logical questions, about the possibility or impossibility of certain outcomes. To answer these questions, they consider both static analysis of the game rules, and generation of uniform play traces. However, in interviews, the authors found that designers were more interested in the statistical likelihood of outcomes than their logical possibility. Possibility is well suited for puzzle games, whereas competitive games must be treated differently. Such games must offer a consistently fresh experience, hence much of the dynamics are emergent from a variety of player choices. When so many playtraces are possible, only statistical likelihood offers a meaningful insight into the players' experience. Fortunately, this is exactly the tack that our work takes - we analyze a gameplay feature by its statistical effect on the likelihood of a player winning or losing.

Ludi *Ludi* [18], [17] is a system for describing and automatically playing general games. Browne's objective, like ours, is to generate player-free quantitative features which describe the design properties of a game. His primary motivation is somewhat different; he sees such features as a guide for procedural content generation algorithms, allowing designers to specify quantitative goals that an algorithm could push a game towards using, for instance, evolutionary computation. Although this application is interesting, my focus remains on human interpretation of design features. This seems a natural first step to develop meaningful measures.

Moreover, being skeptical of the reliability of such measures, I wish to keep humans in the design loop, using automated evaluation only as a first step to precede human evaluation. In fact, Browne states several times that certain design features, such as depth or uncertainty, are 'desirable' or 'correct'. In contrast, I maintain a non-normative stance that design features can inform designers, but that games are an expressive medium in which any number of unusual properties may be desirable. Admittedly, Browne is concerned only with *combinatorial* games, the subset of board games that is discrete, zero-sum, turn-taking, deterministic, and perfect information. These games may indeed admit more general principles for the design of enjoyable games, at least.

Browne defines a number of quantitative game design measures, which generally take two forms: structural and observational. Structural measures are concerned with the definitions of the rules themselves, or the (in principle) directly measurable properties of the game tree; for instance, the Kolmogorov complexity of the ruleset, or the branching factor of the game tree. Observational properties relate to how the game is, can, should, or must be

played, and are measured statistically using simulated play; for example, the typical gap in apparent value of sibling nodes, or the rate at which the winner becomes clear. Most observational properties are estimated using Monte-Carlo Tree Search. (In fact, many of the structural measures are as well, for game trees that are too large to analyze directly.)

Our work implicitly responds to two potential critiques of his approach. The first critique is that many of these measures are very ad hoc, particularly those meant to capture subjective design properties, such as depth and elegance, in [17]. These measures typically take an average, minimum, or maximum of other terms, with little concern for the relative interplay of these features. For example. the 'efficiency' of a game (the extent to which few pieces result in large emergent complexity) is defined as the mean of the piece usage, and the board usage, terms which themselves are defined rather arbitrarily. Of course, these measures are only intended as approximate guides, but it nonetheless raises concerns. Perhaps this cannot be helped with subjective features, but the problem occurs even with somewhat more objective features, such as the 'game tree complexity'. This term is meant to estimate the number of leaves of the game tree; this itself is reductionist, as it ignores the feasibility of these nodes; many may never be reached even by the simplest algorithms. Furthermore, the term is in practice defined by the simplification $\log_{10}\left(\prod_{d=0}^{L(g)-1} BF(g,d)\right)/50$, where $BF(g,d)$ is the observed average branching factor at depth d, and $L(g)$ is the observed average depth of the game tree.

The second critique is that it largely relies on circumstantial observations of agents. For instance, in [18], Browne defines a measure of 'control', which captures the extent to which an agent's actions happen to limit the opponent's quantity of actions. I will argue in Chapter 4 that such observations can be misleading, as it may have been possible to have played as well without exhibiting the properties observed. I will argue for measurements that solely observe the win rate of carefully restricted agents. In actuality, among the many design measures proposed in [18], there are four that measure the win rate of a restricted agent. These measure agents who search at a limited depth, play randomly, or optimize only over one player's pieces. However, Browne does not make a distinction, or address the unique advantages of these measures over those based on observation of AI behavior. Admittedly, behavioral measures may be more appropriate for Ludi, because its task is generative game design, not balancing. Browne does claim that combinatorial games do not typical admit the subtle balancing adjustments of modern video games. I would call this claim into question however, given the number of small tweaks that have occurred even to games such as *Chess* and *Go*, for which there is tremendous cultural inertia to remain the same, and for which centuries of refinement have already taken place.

Other Automated Evaluation Techniques [62] aim to actually design 'balanced'

board games, where 'balanced' means that the game's starting conditions are fair. They measure this property by playing AI agents against themselves and checking that one side does not win a disproportionately large amount of the time, a special case of our framework.

Several projects described below illuminate game designs using simulated players with custom properties. However, all focus primarily on producing example play traces, in favor of quantitative formulation of balance features. One outlier is Bauer-Popovic [9], which explores and visualizes an avatar's range of motion in a level automatically, using Rapidly-Exploring Random Trees and subsequent clustering. This work bears some resemblance to the spatial maps of Chapter 3.1, without the need for human playtests.

Nelson [66] conceptually explores a variety of automatically-generated game metrics, including (briefly) studying the behavior of agents from interesting simple families. He briefly proposes and discusses studying the behavior of agents belonging to interesting simple families. This idea is in essence restricted-play. However, Nelson's focus is not on quantitatively expressive features, which are essential for comparisons across balancing iterations. Dormans presents Machinations [28], a diagramming tool for describing in-game resource dependencies. It is intended to help designers reason about resources through examination, but also through simulation. Users can write scripts to simulate interesting player strategies, and observe their play in the model. Denzinger et al. [20, 27] have studied similar questions in a series of papers, but are concerned only with the special case of discovering short but powerful action sequences. For instance, they discover that in *FIFA* one can consistently score with corner kicks. Smith et al. [80, 81] built *BIPED*, a game prototyping system, on top of their logical engine, *Ludocore*. *BIPED* generates play traces obeying designer-specified constraints. These traces can be examined to gain early qualitative insight into a design. This is complementary to our emphasis on the performance of constrained agents.

Chapter 4

INTRODUCING RESTRICTED PLAY

In this chapter I present the conceptual underpinning of my dissertation, called 'restricted play'. This simple guideline for gameplay analysis strikes a distinct contrast to many existing techniques, yet helps answer similar questions and more. In this chapter I will briefly discuss a motivating problem with much existing work, outline the high-level idea of restricted play, then present some concrete instantiations of quantitative balance measures based on restricted play. The rest of my dissertation will be devoted to applications of this principle, in which we evaluate game balance in disparate domains and phases of the design process.

4.1 The Problem with Observational Quantitative Balance Analysis

Imagine the following hypothetical scenario: a designer wishes to understand the balance of an existing game, such as *Chess*. Such understanding is crucial to creating a variant of the game, but is also informative for designing new games. In particular, she is interested in the extent to which various dynamics play a role, from the use of a given piece, to the possession of a certain part of the board, to the employment of aggressive fast-paced play. Let us assume that she has access to a source of skilled play, be it human beings or AI players.

Many existing game projects find themselves in a situation something like this, be they academic or industry projects, as discussed in Chapter 3. These projects are more commonly aimed at the analysis of a game under development, but this distinction is not important. What is important is the methodology that these projects take. The most common approach is also the most natural: observation. Have your players play a large number of games, and simply observe what happens. The frequencies of occurrence of various dynamics can then act as a measure of those dynamic's impact. If players move knights 20% more than pawns, this might be taken as evidence that knights are in some sense 20% more important or powerful than pawns. If long games are twice as common as short games, this may seem to say something fundamental about the pace of the game. If the central square are 50% more

likely to be occupied than the outer squares, this may say something about the importance of these space. In the extreme cases, perhaps we could say that a game is imbalanced if some phenomenon is substantially more common than its alternative, and this disparity does not match the intended vision of the game.

At first blush the above program sounds reasonable. However, it begins to break down by looking at another simple example. If we were to measure the frequency of movement of knights and compare it to the frequency of movement of kings, we would find a tremendous disparity. Yet we should by no means say that knight moves are substantially more important, valuable, or powerful than king moves. The key observation here is that the *freedom* to move the king is a defining dynamic of *Chess*, even if it rarely happens. We cannot hope to capture the true impact of dynamics if we simply compare their frequencies.

A second problem with the observational approach is that the frequency of events may be incidental. There could be many viable strategies for a game, which use, say, knights to varying extents. The fact that a given collection of observations places a high frequency on some dynamic seems like it says little fundamentally about the game. When working with large databases of human gameplay this is not so problematic, as designers are concerned with the human experience, though even here gameplay is at risk of changing with time. The real problem arises when working with automated agents, which may be substantially biased toward certain strategies. Even optimal strategies to zero-sum games can be quite distinct.

With the flaws of the observational approach clear, the solution is no longer so difficult to see.

4.2 The Alternative: Restricted Play

The criticisms levied at the observational approach in the previous section rest on the notion of 'impact'. In examining the designer views on balancing in Chapter 2, it is clear that impact is indeed a key concern of balancing. After all, what distinguishes balancing from game design at large? Let us consider elements of game design that I would not consider relevant to balance: surprise achievements, finishing moves, and narrative, for example. These elements absolutely affect the player's holistic experience, but they do not impact the gameplay per se. In particular, they do not affect a player's ability to achieve her goal. This is what I consider impact, and indeed a scan of the patch notes for any popular competitive game will reveal that the changes categorized as balance in some way affect a player's ability to win. For example, if a designer is concerned that players are jumping too much in her game, I claim that this is only a balancing concern if that jumping has any impact on their ability to play effectively, or the manner in which they must play effectively.

If it does not, the jumping itself reduces to an aesthetic concern![1]

There is at least one form of game balance in which characterization by player effectiveness is the norm: fairness of asymmetric starting conditions. For games in which players must choose one of multiple 'modes of play' (such as a race in a real-time strategy game or character in a fighting game) before beginning, fairness is often seen as an utmost priority. A game with one clearly superior starting condition quickly reduces to a game of only that starting condition; a game with clearly inferior starting conditions sees those modes wasted and rarely played. Both players and designers are more satisfied, in most cases, when an asymmetric game is fair. In fact, fairness is the property most often associated with the term balance, with some designers and players recognizing fairness as the *only* meaning of balance.

Despite the centrality of fairness as a form of balance, it is by no means the only form, as is made clear by the broad variety of forms described by Jesse Schell, (and paraphrased in Chapter 2) from 'skill vs. randomness' to 'short vs. long'. Is there any connection between these diverse forms? I claim that there is. In fact, I claim that many varieties of balance can be *reduced* to that central form, fairness. The form of the reduction is simple: to understand the balance of some dynamic, we may frame it as the fairness of a match between two players, one of whom is restricted in a way that highlights that dynamic. In other words, we hallucinate a player (realistic or otherwise) whose behavior captures that dynamic or the lack thereof. This is what we call restricted play.

Concretely, to measure the impact of some dynamic, we measure the win rate of a player who is restricted in their access to that dynamic, when facing a player without that restriction (in most cases, similarly skilled). For the time being we will not concern ourselves with the mechanical constraints of imposing such a restriction, and rather attend to the definition itself. Even impractical restrictions can potential provide a satisfactory definition of previously vague forms of balance, thus providing a concrete basis for discussion, at the very least.

In the following section I present a collection of restricted play balance measures, which take existing concerns of balancing and frame them as the win rate of asymmetric agents. This list is nowhere near exhaustive; in many cases the relevant measures will be specific to a single game, but can be easily specified by the game's designers. The list below also skips over subtleties like the intersection of restrictions (which can be a valuable way of measuring the effect of some dynamic on the manner in which a game must be played)

---

[1]This example is not purely hypothetical; a subset of players are known to repeatedly jump throughout platformer games, even in portions in which the jump has no impact on their speed, access or combat. It is essentially a 'tick', yet one that players can be frustrated to lose in games such as *Legend of Zelda: The Ocarina of Time*, where jumps are restricted to circumstances in which they would be meaningful.

and relaxed versions of restrictions, which capture a one-dimensional rather than zero-dimensional notion of a dynamic's impact. Nonetheless, the list serves as an effective starting place.

4.3 Restricted Play Balance Measures

For simplicity, we consider zero-sum two-player games. A game is given by a set of states (some of which are win or tie states), a set of actions available to each player at each state, and a transition function mapping each state and pair of actions to a child state, or distribution over child states.

Let $\mathcal{R}$ be some behavior restriction - for instance, never playing a certain action, or avoiding certain outcome states. For some player or agent A, we let $A_{\mathcal{R}}$ be A modified with the restriction $\mathcal{R}$. If we consider some game element that $\mathcal{R}$ avoids, then we measure the impact of that game element by the probability that $A_{\mathcal{R}}$ beats A.[2] (Technically, we work with the *value* of the game [93], by treating a tie as a 50% chance of a win.) We assume that A has full knowledge of its opponent's restriction, and is thus able to exploit it.

In [49], we describe several initial categories of balance features, and the restricted behaviors $\mathcal{R}$ to measure their impact. I summarize these features here, to complement the measure of strategy described in Chapter 5.1.

How important is playing unpredictably? In real-time games such as fighting games, optimal play often requires players to randomize their strategies. However, the cost of playing predictably can vary across games. We measure the importance of unpredictability with a restricted player who must play low-entropy mixed strategies. In practice, this can be approximated by a player who can play any distribution, as long it has small support. $\mathcal{R}$: $|\textbf{Support}| \leq \textbf{k}$.

To what extent must players react to the current state of the game? Real-time strategy games often allow for fixed 'build orders', which are long-term strategies suggesting some actions to take at each timestep, roughly independently of the other player's actions [15]. If a build-order is rigid yet effective, it may result in a strategically uninteresting game. We capture the importance of adaptation with an *oblivious* player: one who for the first k rounds knows only her own actions. She cannot query the current state of the game or her opponent's actions, and instead must simulate her opponent. $\mathcal{R}$: **Oblivious-until-round-k**.

---

[2]The randomness in the outcome of a game can come from a number of sources. In games of simultaneous action, such as *Rock-Paper-Scissors*, optimal play requires randomness. However, even for non-simultaneous games, where randomness is not necessary, many heuristic agents use randomized strategies [19]. Finally, a game may itself have random transitions.

How powerful is a given action or combination of actions? In strategy games such as *Chess*, each piece is meant to act in unison: each should be essential to the game. In contrast, first-person shooters typically allow players to play according to their preference of weapon and play-style: no one weapon should be essential. We capture the importance of an action with a restricted player who is forbidden from using that action ever (or more than k times). Also informative is a restricted player who *must* use that action the first k times it is available. For action $\mathbf{a}$, $\mathcal{R} : |\mathbf{Plays(a)}| \leq \mathbf{k}$, or $\mathcal{R} : |\mathbf{Plays(a)}| \geq \mathbf{k}$.

How much long-term strategy is necessary? Actions in strategy games often have emergent consequences which are not observed until many turns later. In contrast, action and racing games, among others, tend to rely more on short-term tactical thinking. We capture the impact of long-term strategy with a player who is restricted to explore states at most k steps into the future. At one extreme, a player does not strategize at all, playing completely randomly. Depth-1 strategy produces a *greedy* player, who optimizes for some score function of the subsequent state, like health or piece count. At the other extreme, a player may search all the way to the leaves. The win rates of these varying search depths (against a strong, unrestricted opponent) gives a broad picture of the role of long-term strategy in the game. $\mathcal{R} : \mathbf{Depth} \leq \mathbf{k}$.

Is the outcome known long before the game's end? Some games (such as *Risk*) are infamous for a 'lame duck' phase, in which the likely winner is known, but players must continue for some time to finish the game. This can decrease tension, or force players to concede early. On the other hand, other games (such as *Mario Kart* with 'rubber banding' opponents) leave the outcome uncertain until the end, making the majority of the game feel meaningless. To capture the extent to which outcomes are apparent early in the game, we study players who assign an additive value bonus ϵ to 'low' states, those at most k moves from a leaf. If this player cannot win frequently, it indicates that the game will tend to drag on, even once the winner is somewhat certain. $\mathcal{R} : \epsilon$-**bonus-to-height-k**.

What is the effect of avoiding certain end states? Games that include multiple win conditions (such as *Civilization*) often do so to create an interesting strategic decision: players must hedge their goals, to prevent their opponent from acting to stop any one. Other games such as *Magic: The Gathering* feature multiple win conditions simply to let players choose their play style. To capture the importance of hedging goals, we study a restricted player with a modified payoff function, who treats certain win states as losses. (A related measure is for aggressive or passive players, who treat ties as losses or wins.) For a set of leaf states $\mathbf{S}$, $\mathcal{R} :$ **Avoids-S**.

Are the starting conditions of the game fair? This is the question most traditionally thought of as 'balance'. It is often essential that asymmetric starting conditions be

somewhat fair, as players are 'locked into' them [79]. For example, one player must play black in *Chess*, and if she were at a tremendous disadvantage, the game could be frustrating. That said, even unfairness can be intentional. The *Street Fighter* series features an intentionally weak character, who players sometimes play as a challenge, a handicap, or a joke. To measure the fairness of starting conditions, we use an unrestricted player who always chooses a given condition. For a starting condition s $\mathcal{R}$: **Chooses-s**.

4.4 Applying Restricted Play

The discussion in chapter has intentionally been kept fully abstract, even philosophical. Of course, in so doing, I have swept under the rug much of the challenge of building and utilizing a rigorous formulation. Fortunately, that will be the domain of the remainder of this dissertation. Part II is all about what can be done with restricted play. The contexts of balancing are varied, as our the resources available to designers. But the principle of restricted play is sufficiently broad that it can usefully capture and quantify balance in a variety of these scenarios. Much of the work I will present is in its infancy. Yet it will be invaluable even if it only serves to establish the promise of restricted play as a useful way of framing and working with the once-black art of game balance.

Part II
MEASURING GAME BALANCE

Chapter 5

EVALUATING GAME BALANCE WITHOUT PLAYERS

Thus far, I have sought to convey the importance of game balancing, and the challenge of even understanding the balance of a new or existing game. Historically, knowledge of game balance has comes primarily from gameplay by the designer herself, the studio at large, focus testers, beta testers, or live players.

This gameplay is only as informative as the observations taken from it. Each of the above contexts has its own advantages and disadvantages. A designer playing her own game can explore the relevant gameplay facets, and can understand how her own reasoning and ability led to the outcomes observed; on the other hand, a given designer is unlikely to be able to imagine the play-styles of the diversity of potential players, let alone simulate them. For this reason, developers have for decades brought playtesters into the office to play and be observed. These observations can be enlightening, but still occur at a limited scale, depend on some imagination of the players' experiences (since self-reporting is only so accurate), and are subject to the limited attention and confirmation biases of the designers. Video recording and telemetry get around these problems to some extent, and further allow for an increase in scale to players in their own homes, though with limited access to helpful additional information like attitudinal surveys. Such methods were discussed somewhat in Chapter 3, and will be the subject of much of the subsequent chapters' work.

Each of the phases and scales of playtesting and evaluation is essential. However, in this chapter, we are concerned with only the earliest phase of balancing, in which a designer would typically play a game herself for insight. This phase is not often examined in the academic literature, but is of the utmost importance. Designers spend a good deal of time rapidly iterating on parameters and scripts, in an effort to find a game variant that feels right and addresses the balance concerns of the moment. Some of this work is performed as a sort of hill-climbing, in which new parameters are encoded (such as duration of some attack), and the game is playtested by the designer herself, against an AI opponent, a nearby colleague, or even a second controller in her other hand. In other circumstances - particularly when player progressions or game economies are the subject - designers frequently work with spreadsheets containing hundreds of parameters. Simple models often guide the choice of

these parameters, such as rules of thumb that costs should progress according to a Fibonacci sequence [10], or the use of quadratic, linear, or feedback models.

Both rapid playtesting and spreadsheet-based design, while invaluable under the circumstances, are fundamentally lacking. Each approach makes iteration rather time-consuming. Many design problems are sufficiently complex that hill-climbing is of limited use, and guess-and-check to some extent necessary. In such a context, it is very difficult to iterate through a large enough number of variants to find an ideal candidate. What's more, the balance feedback of rapid playtesting and spreadsheet-based design can be quite misleading. Rapid playtesting naturally imposes the designer's own play biases; even if a designer is skilled at imagining and playing like a diversity of players, she often simply does not have the time. Games are huge possibility spaces that are difficult to single-handedly explore. Spreadsheet-based design can act as an excellent guideline, yet rests on so many assumptions (sometimes unstated) about the emergent properties of systems that at times it can do more harm than good. An assumption that some system (like virtual good prices) should grow linearly might rest on the assumption that some other statistic (like player virtual wealth) grows linearly. Few emergent systems behave quite so nicely, and without testing to confirm these phenomena, insights taken from balance spreadsheets must be approached skeptically.

As a consequence of these challenges, game candidates can be promoted to larger playtesting sessions with glaring balance flaws. Many such flaws are unavoidable, but others could be discovered early, if we had a balance early warning system that could somehow identify these flaws. This is where restricted play can rear its head, by providing a rigorous, *measurable* definition of certain forms of balance. In [49] we initiated a foundational investigation into viable methods to evaluate game balance without any players at all, which I overview below.

5.1 The Potential of AI-Based Balance Evaluation

A natural question is whether some portion of competitive game balance evaluation could be automated, through AI simulation by a backend reasoning tool [67] rather than playtesting. One major roadblock to this goal is that the standard approach to evaluating balance relies on observation of typical human play, yet AI play is not currently predictive of human play [46]. The diversity of viable strategies in complex games makes it unlikely for even a strong AI agent to behave like a human. A more basic roadblock is that many balance questions have not even been well-defined.

One balance question that *is* relatively well-defined is the fairness of starting conditions. For example, the fairness of *Chess* is characterized by the win rates of the white and black

players. Recall that the idea of restricted play is to pose other kinds of balance quantitatively by reducing them to the win rates of asymmetric players. When these players are artificial intelligences, this enables us to exploit similarities between human and AI *skill level* rather than behavior. Such similarities are now becoming a reality, as recently developed AI techniques, such as Monte-Carlo tree search, can play an unprecedented variety of games competitively with humans. This addresses the first roadblock above, by allowing us to ignore the specific behavior of agents.

For example, suppose we wish to automatically gauge whether a game is focused more on long-term strategy, like *Risk*, or on short-term tactics, like a first-person shooter. To do so, we measure the win rate of a *greedy* agent, who optimizes only for a simple 'score' function of the subsequent state, versus a standard, proficient opponent who exploits the restricted agent's greediness. In *Chess*, the score might be number of pieces; in a fighting game, it might be health. Each such restriction allows us to capture gameplay contextualized through some player behavior, by *forcing* the agent to play in some fashion (of common interest to game designers) and having it optimize subject to this restriction. This contrasts with most prior work, which would have an agent play as well as possible, and simply observe it for certain behavioral properties, which may circumstantially never occur.

We propose that such formulations facilitate an algorithmic approach to answering competitive game balance questions, since win rates can be estimated with repeated games between reasonably strong AI agents. It allows us to build an 'early warning system', that illuminates certain flagrant imbalances upfront, saving playtesting time for the most promising iterations. An imbalance can match a designer's goals, but she will always hope to be aware of it.

Consider a chess-like game, where one design goal is that all the pieces must work in unison. A designer might consider a rebalance in which the movement radius of one of the pieces p is extended. This has the potential to impact other aspects of the game indirectly, for example rendering useless some other piece q. This can be hard to predict using reason alone. Yet by observing that in the rebalance, restricting an agent from moving q no longer diminishes her win rate, the designer is immediately directed to the flaw. She could then use this information to refine her intuition, and go back to the drawing board without playtesting this iteration at all. If, on the other hand, the reasons for this result remain unclear, she always has the option of playtesting.

5.2 Exploratory Study

We built a prototype balance evaluation tool, to serve as an application of restricted play, and a testing ground for its validity. The tool measures the behavior of optimal (restricted)

agents playing two-player, perfect-information games in extensive form, with simultaneous moves and randomized transitions. Such games have optimal strategies that can be computed exactly in polynomial time [93]. It is straightforward to adapt such solvers to many restricted behaviors.

A System to Support Rapid Iteration I describe an experimental tool for supporting rapid design iteration. This tool can be provided with code embodying a parameterized game tree. It then facilitates the following interaction cycle. A designer inputs a set of game parameters into a web-based interface, (Figure 5.1, left side) and is presented with a set of balance measures for the specified game, derived by computing the value of several restricted behaviors. These measures are ranked by their intensity, so that the restricted behaviors that are most effective or ineffective are highlighted. (Figure 5.1, right side). The designer forms a hypothesis about how modifying the parameters could resolve the balance features she finds most problematic, and repeats.

Of the balance features in Chapter 4.3, our prototype tool implements all but two of the proposed balance measures.

- 'Omit **a**': $|\mathbf{Plays}(\mathbf{a})| \leq \mathbf{0}$.
- 'Prefer **a**': $|\mathbf{Plays}(\mathbf{a})| \geq \infty$.
- 'Random': $\mathbf{Depth} \leq \mathbf{0}$.
- 'Greedy': $\mathbf{Depth} \leq \mathbf{1}$.
- 'Agressive': **Avoids-ties**.
- 'Low-randomness': $|\mathbf{Support}| \leq \mathbf{2}$.
- 'No-randomness': $|\mathbf{Support}| \leq \mathbf{1}$.
- 'Player 1': **Chooses-first-player**.

Monsters Divided To evaluate the relevance of our tool, we used it to help balance *Monsters Divided* - a card game developed by our lab to help teach fractions to elementary school students. (Figure 5.2.) The game consists of *monster cards* and *power cards*. Each monster card has a fraction and a color: red (R), green (G), or blue (B). Each power card represents a function which modifies a fraction through either addition or multiplication. For example, a power card may be labeled $+1/3$ or $*2/3$. Each player is given three monster cards and two power cards, visible to both players.

	Power Cards	**Noteworthy Balance Measures**	**Interpretation**
A	$+1/2$	Omit Green: 13.54%, Omit Blue: 7.44%, Omit Red: 7.83%.	Green is too weak. (Green cannot beat Red, even by playing $+1/2$.)
B	$+2/3$	Omit Blue: 6.76%, Omit Red: 11.62%, Omit Green: 12.38%.	Blue is too strong. (Red cannot beat Blue, even by playing $+2/3$.)
C	$+2/1$	Omit Blue: 10.77%, Omit Red: 10.83%, Omit Green: 10.78%.	All monsters are about as strong. The slight differences point to interesting variety in how they are played.
D	$*1/2$	Random: 8.04%.	Random play is unreasonably effective.
E	$*2/3$	Omit $Self * 2/3$: 47.41%, Random: 11.99%.	Power card on self unhelpful; random play too good.
F	$*4/1$	Omit Blue: 6.76%, Omit Green: 12.38%, Omit Red: 11.62%, Random: 3.26%.	Blue is too strong, but now random performs sufficiently poorly.
G	$*2/3$, $*4/1$	Greedy: 3.41%, Prefer $Self * 4/1$: 3.83%.	Too harsh to simple players; version of $* 4/1$ is self-destructive.
H	$*2/3$, $+2/3$	Prefer $Self * 2/3$: 5.41%, Omit Blue: 4.31%.	Multiplier power is less self-destructive, but Blue is now too strong.
I	$*2/3$, $+1/2$	Prefer $Self * 2/3$: 23.73%, Greedy: 7.95%.	Values appear reasonable, as do variety among actions. Ready to playtest.

Table 5.1: Balancing iterations of the set of power cards in *Monsters Divided*. We give noteworthy balance measures for each iteration, and the interpretation that led to the next iteration. For example, in iteration E, playing the power card $*2/3$ on one's self is mostly useless, as the inability to do so barely hurts chances of winning. Also in E, playing randomly performs better than intended. We address these problems in F, with a larger power card that may have a stronger effect.

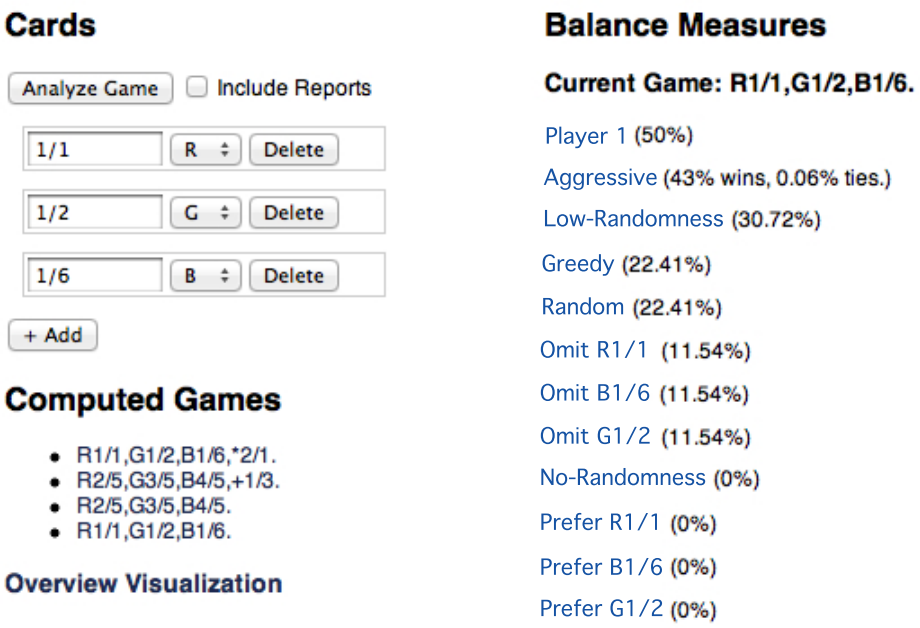

Figure 5.1: A basic analysis using our tool. The results show the performance of varieties of restricted play against a restriction-exploiting player. Players who never play Red 1/1, for example, win 11.54% of the time.

In each round, players simultaneously play a single monster card, trying for the 'best' fraction. The color of the played monster cards determine the 'color of the round'. Blue overrides red, red overrides green, and green overrides blue. *In a red round, the highest fraction wins. In a blue round, the lowest fraction wins. In a green round, the fraction closest to* $\frac{1}{2}$ *wins.* After playing monster cards, players can also choose to play a single power card, or pass. Power cards can be applied to either player's fraction. At the end of the round, the winning player gets a trophy of her monster's color. If there is a tie, both players get their monsters' respective trophies. The played power cards are discarded, and the monster cards are returned to the players. To win the game, a player needs either one trophy of each color or three trophies of one color. The game can end in a tie.

Balancing the Cards he rules above leave unspecified the cards used by each player. Hence our next design goal was to choose a small, balanced set of starting cards.

We knew that a simple set of monster cards (*Red 1/1*, *Green 1/2*, and *Blue 1/6*) created a reasonably interesting game of hedging against win states and prediction. To add complexity, we wished to add a pair of power cards to the game. Rather than generate a single plan and immediately playtest it, we used our tool to generate interactive feedback on many

Figure 5.2: Screenshot from *Monsters Divided*. $G1/1$ and $B5/6$ are played, with $+3/2$ and $+1/3$ power cards applied to them, respectively. G's rule is in play: closest to $1/2$ wins. $5/6 + 1/3$ beats $1/1 + 3/2$, so P2 wins a B trophy.

designs, build hypotheses to explain the feedback, and invent new variations to address the feedback. In this way we threw out many inarguably bad card sets, until a promising set could be promoted to playtesting.

To select a strong pair of power cards, we first investigated the properties of single power cards in isolation. We then moved on to select a pair of these cards. Note that at all stages, we relied on some intuition for the values of a balance measure that should be considered a success. This comes partly from common sense, but also from practice working with the tool.

An abridged history of the design session is found in Table 5.1. The variations shown constitute half of the evaluated variations - only the most informative. By applying the tool, we were able to quickly discover design flaws such as:

An overpowered or underpowered monster card. We intended all three monster cards to be roughly comparable in usefulness. This balance was frequently broken by power cards, and the tool immediately informed us, as in variations A, B, and F. For example, in variation A we see that restricting the player from playing Green has a smaller impact on the win-rate than restricting Red or Blue. This suggests Green as under-powered. We corrected such variations in two ways. Sometimes we reasoned about the ways a power card could help one monster more than others, and tried to weaken that effect. Other times, we simply searched the surrounding space of cards, evaluating the tool on several similar

variations. In this way we formed intuition for the effects of differing power cards.

Leniency or strictness toward simple players. We wished to create a strategic game that was nonetheless accessible to players with a basic understanding of the rules. Hence we were wary of variations in which greedy or random behaviors performed extremely well or poorly. Variations D and E suffered from the former problem, G from the latter problem. In variation G, a greedy player wins only 3.4% of games, which probably indicates too harsh a punishment for mediocre play. To address these problems, we focused on tempering the power card that was most effective or harmful when 'preferred' (played as soon as possible).

A self-destructive or irrelevant power card. In some variations, a power card was unhelpful, or self-destructive if used at the wrong time. These diagnoses came from observing that an 'omit' behavior performs nearly perfectly, while a 'prefer' behavior performs terribly. This occurred in variation G, where preferring the power card '$Self * 4/1$' often lead to losing the game, and in variation H. We decided that a 'decoy' strategy - which players must learn to avoid - was harmful to the accessibility of our game.

After twenty or so iterations, we arrived at a design which passed most of the quantitative tests we were concerned with. Not all values were perfect, but they were satisfactory enough to feel confident going into playtesting. The design problems we identified would have been definitively incompatible with our goals for the game. In using this tool, we saved the time and effort of up to 20 phases of manual evaluation, many of which would have required playtesting. Moreover, by tracking the graph of balance features across iterations (Figure 5.3) we gained a novel form of intuition for the dependencies of the game on parameter choices.

5.3 Monte-Carlo Tree Search and Heuristic-Free AI

My primary ambition in this chapter is to allow designers to automatically evaluate the balance of real games. The case study of Chapter 5.2 does this, but is lacking in three senses, all related to its use of optimal players. First, the game is substantially simpler than the majority of interesting games. Second, there are cases where a restriction might effect an optimal player to a much greater or lesser extent than a real player. In contrast, it is a realistic possibility that a strong heuristic algorithm would see a similar effect from restriction as a real player. Third, for many games (e.g. deterministic turn-taking games), optimal play discounts the possibility of any meaningful 'win probability' at all. In the remainder of this chapter I explore the use of heuristics to address these problems, first giving some background on our eventual heuristic of choice.

Monte-Carlo Tree Search (MCTS) has come to prominence quickly since its introduction in 2006. MCTS first originated in [24] as an approach to AI for *Go*, and in its more

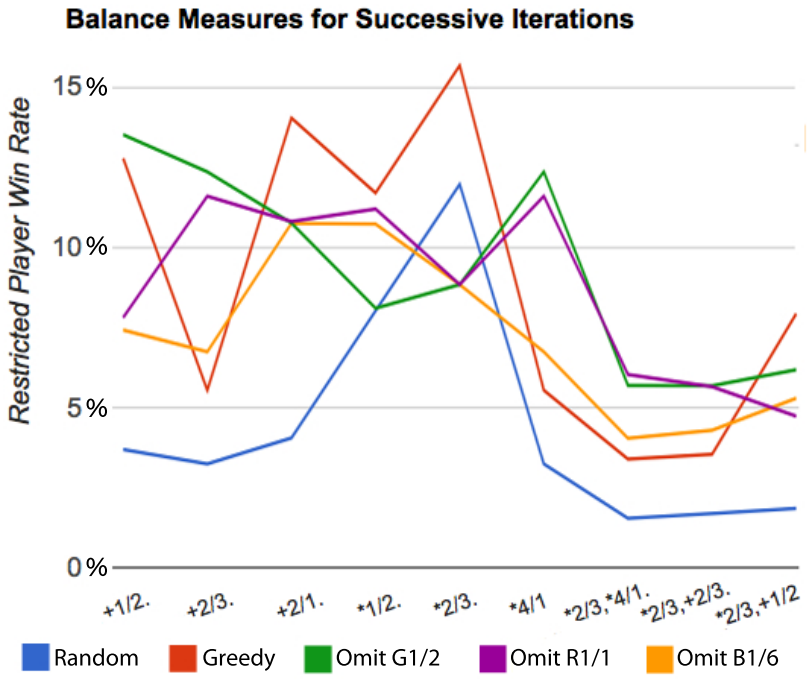

Figure 5.3: Balance parameters across variations. The gaps between *Greedy* and *Random* convey the relative roles of long-term strategy and short-term tactics. The designer can easily discover trends such as the power gap between G1/2 and R1/1 due to a 1/2 power card.

commonly used form in [55]. The application to *Go* was the source of much of the algorithms' success. Historically, tree search methods such as $\alpha\beta$-pruning had proved ineffective on *Go*, due to the lack of an accurate board evaluation heuristic. By eschewing heuristics as its primary mechanism, MCTS achieved the first computer successes in *Go* against professional players. It does so by relying on Monte-Carlo Evaluation (MCE) of internal states in place of heuristics, rolling out entire games naively to leaves, and using those leaf values to estimate the internal states' values.

At the time of MCTS's invention, MCE had already seen frequent use [1] [12] in approximately solving both game trees and Markov Decision Processes. The innovation of MCTS is to interleave its MCE with the tree search itself. Earlier algorithms would simply run breadth-first tree search or iterative deepening, and apply MCE at each 'leaf' of the search. In contrast, MCTS applies MCE to essentially every state that it encounters before further exploration of a state is considered. In this way the problem becomes one of refining a model for each visited state, beginning with naive MCE, continuing with deeper tree search, and repeatedly picking which model to refine in a way that maximizes the expected payoff.

Algorithm 1 RunMCTS

Input: A game tree T and the current decision state S.
Output: A child state S' to select for the next move.
1: $globalRoot \leftarrow S$
2: **while** $time < MAX\_TIME$ **do**
3: $\quad RefineMCTS(S)$
4: **end while**
5: **return** $\text{argmax}_{S' \in children(S)}\, value(S')$

Algorithm 2 RefineMCTS

Input: A game tree T and a state S from which we wish to run a single simulation.
Output: The achieved payoff of this playout.
1: $S' \leftarrow SelectChild(children(S))$
2: $payoff \leftarrow RefineMCTS(S')$
3: **if** $ShouldStoreState(S)$ **then**
4: $\quad$ Initialize numVisits(S) and value(S) to 0 if undefined
5: $\quad numVisits(S) \leftarrow numVisits(S) + 1$
6: $\quad value(S) \leftarrow value(S) + payoff$
7: **end if**
8: **return** $-payoff$

Algorithm 3 ShouldStoreState (UCT)

Input: A state S.
Output: True iff we should store stats for this node in memory.
1: **return** $numVisits(parent(S)) > 0$

At a high level, MCTS is a method for approximating the value of a game state by recursively simulating MCTS on each successor in an interleaved fashion. This interleaving allows MCTS to iteratively refine the results of each child simulation, concentrating its computational resources on the most promising children. A single 'step' of MCTS typically consists of following a path all the way to a terminal - using pre-existing knowledge or random selection as appropriate - and backing up the value of this terminal. There are many variants of MCTS, but all follow this rough template. This formulation as a recursive algorithm is novel, as far as I know, and helps facilitate the use of restricted play; pseudocode can be found in Algorithms 1 and 2.

RunMCTS is the top-level function, which is given the current state from which we wish to take an action. It is given a computation time-limit, and simply calls RefineMCTS repeatedly, until it runs out of time. Successive calls to RefineMCTS establish a progressively more accurate view of the values of the current state's children. It does so by recursively calling RefineMCTS on these children, in an order given by the SelectChild method. The visit-counts and value-estimates established at each node are only stored at those for which ShouldStoreState returns true, in order to reduce memory overhead. Without loss of generality, we assume that player order alternates.

Algorithm 4 SelectChild (UCT)

Input: A set of states $\mathcal{S}$.
Output: A single $S \in \mathcal{S}$ to further explore.
for $S \in \mathcal{S}$ **do**
  $exploit(S) \leftarrow \frac{val(S)}{numVisits(S)}$
  $explore(S) \leftarrow 2C\sqrt{\frac{2 \ln numVisits(globalRoot)}{numVisits(S)}}$
  if numVisits(S) $= 0$ **then**
    $UCB(S) = \infty$
  else
    $UCB(S) = exploit(S) + explore(S)$
  end if
end for
$choice \leftarrow \text{argmax}_{S \in \mathcal{S}} UCB(S)$ [Break ties randomly.]
return *choice*

The MCTS template itself does not specify the implementations of SelectChild and ShouldStoreState. I provide implementations of these methods for the most well-known variant of MCTS, called Upper-Confidence Trees (UCT). This pseudocode is found in Algorithms 3 and 4.

Upper-Confidence Trees (UCT) was established by Kocsis and Szepesvári in one of the two formative MCTS papers [55] [24]. The main insight of UCT is that selecting a child to further simulate is analogous to selecting an arm to pull in a Multi-Armed Bandit Problem. Though the analogy is not perfect, [55] show that Upper Confidence Bounds (UCB) - the best algorithm for minimizing the regret of Multi-Armed Bandit [5] - can be used in Monte-Carlo Tree Search. They show that recursively applying UCB to select children to visit results in eventual convergence to optimal play, and moreover that the probability of failure shrinks at a 'reasonable rate'.

The idea of UCB, as with any Multi-Armed Bandit solution, is to balance two conflicting goals: to exploit probably-high-paying arms, while exploring poorly measured but potentially-high-paying arms. UCB accomplishes this in a simple way: by choosing the arm that maximizes the sum of its average payoff and a carefully chosen 'exploration term'. This exploration term is asymptotically the square root of the log of the total number of pulls over the number of pulls of this arm. As we make more total pulls, we slowly become more likely to pull historically-less-successful arms, and we choose these inversely proportionally to their pulls so far.

[55] do manage to show that the bounds of [5] hold even when the distributions of the arms are non-stationary, provided their drift is bounded in a few key ways. Such an extension is necessary because the payoffs of simulating a given child evolve as we store more information about that subtree. [55] then show that these drift bounds hold inductively, for progressively higher nodes of the game tree, which treat subsequent simulations as arms. It

should be noted that the interpretation of simulation selection as arm selection is somewhat stranger than it appears at first glance. The greatest abnormality, as noted in [24], is that the goal of MCTS is not to maximize the sum of the value of several arm pulls, but rather to find the 'arm' with the greatest expectation, and choose that arm (or back up its value, if we are sitting at an intermediate node). This is a direct consequence of working with minimax game trees.

So why do we apply UCB, and in fact report the average value of the all child simulations, rather than just the best one? For one, the error bound of UCB ensures that we will not do much worse than we could have done by only simulating the best child repeatedly. Second, by averaging over all child simulations, we more accurately reflect our uncertainty of the true value of this node. If we knew with absolute certainty the value of a given child, then we could simply report that value, but if that value is sufficiently high, we would visit that node exclusively. We want the value that is associated with a given node (and that is backed up) to reflect our best estimate of the value that can be achieved from visiting that node; to be effective this model must be conservative. By being conservative in the particular manner mandated by the selection process of UCB, we can get approximate guarantees of the value of the parent node, provided the algorithm operates correctly inductively.

5.4 Benefits of MCTS for Balancing

MCTS is particularly well-suited to the needs of Resticted Play. In this section I outline many essential properties of a restricted play algorithm, which held by MCTS.

Heuristic-Free: By using Monte-Carlo samples, MCTS can eschew the use of all domain knowledge. This allows evaluation of new iterations for which, by its nature, effective strategies may not be not known. There are several methods for incorporating heuristics into MCTS to improve its performance. We can use heuristics to break ties [78], particularly during the random rollout phase. Methods like [38] learn heuristics from experience, without user intervention. It is true that the strongest uses of MCTS have incorporated heuristics, but this is not damning for our application: much can be learned from intermediate play; designers may be able to provide heuristics that function across iterations; and automatic learning of heuristics may be feasible.

Powerful: MCTS is the first algorithm to play at a tournament level in 9x9 *Go*. Full *Go* boards are more difficult, but I am satisfied to restrict my immediate goals to games as complex as 9x9 *Go*, which has over 10^{38} states. Future innovations in MCTS can be seamlessly applied to analyze more complex games. Alternately, games can be played at an abstracted level, with hard-coded subroutines for sub-phases of the game, as is being considered for complex games such as *Starcraft* [22].

Restrictable: MCTS uses a simple encoding of a game: it takes a successor function for states and a value function for terminals. By modifying these functions, we can create many restrictions of interest. To forbid strategies, remove them from the successor function. To 'prefer' certain states, simply modify the value function associated with them. To limit depth of search, simply cap the game tree at that depth, and treat inner states as terminals with value given by a heuristic.

Note that not every restriction of Chapter 4 is straightforwardly computable. The most computationally difficult restrictions are those involving some global optimization over all rounds of the game; for example, restricting players to use any single subset of actions for the entirety of the game. Such a restriction is very informative from a design perspective, as it reveals whether a game has a simplified sub-game embedded in it. Such a sub-game could be discovered by expert players then passed on to weaker players, making the game more accessible, but potentially less interesting for experts. For instance, beginning *Street Fighter* players are often told to use a small subset of their character's moves, and think only about the tradeoffs between these few moves. If such a subset is yet unknown, then this restriction is difficult to play with. Approaching it greedily requires an early 'commitment', meaning that the first few moves have a disproportionate effect on an agent's success. Running simulations with each action subset in turn is likely to be infeasible. The restriction will be difficult to study in any algorithmic technique: it is likely NP-Hard. The best we can hope to do is apply some heuristic to the subset-selection itself. For example, we can employ a hill-climbing approach in parallel with the MCTS computation. Each time MCTS seems to be converging, we can attempt to swap out a single action for any other, continue to run the MCTS for some time, and observe whether it moves in a positive or negative direction. I would like to experiment with this approach for a game like Chess, to discover the a sparse subset of the play space that performs reasonably well.

Restriction-Exploiting: It can be difficult to design agents that exploits an opponent's restriction, because many algorithms make the simplifying assumptions that the opponent behaves like the agent itself. MCTS is well-suited to overcome this challenge because it is built on a form of opponent modeling. MCTS evaluates child states by explicitly modeling its opponent as MCTS (with less total power) on each of these children. It is possible to run that model with restrictions. When computing opponent moves, apply the restriction, and when computing our own moves, choose moves unrestricted.

Such an approach succeeds under three very rough conditions: MCTS should be almost as effective as the opponent's real strategy, the current player must have a game-theoretic advantage, and the value function of the game must be somewhat 'smooth'. (I.e. adding slightly more computation should only reveal much better strategies with low probability.)

Alternatively, the current player can lack an advantage, but the opponent must be using an algorithm that is weaker than MCTS at the simulated level of power.

Given that MCTS already models its opponent, it is not hard to run that model with restrictions. When computing opponent moves, apply the restriction, and when computing our own moves, choose moves in an unrestricted way. There is one suspicious aspect of such an approach: the restricted opponent's model of us must be unrestricted. This is because we rely on the opponent's recursive simulation of us not just for predicting the opponent's move, but also for predicting the value of the subsequent states for us. I.e. we assume that the opponent's model of us is realistic. Avoiding this assumption is difficult, because we must assign separate computational resources to computing our opponent's prediction of us, as well as the 'actual' moves that we will make given their moves. Interleaving such computations may well be feasible, but has not been pursued in the literature as far as I know.

Note that in some cases it is natural to assume that the opponent is able to simulate us in an unrestricted way. But this falls apart in restrictions that are meant to test limited understanding or strength of a player. For example, imagine that our opponent is a greedy player, who can only search to depth one. It is clear that in exploiting this restriction, we must simulate our own deeper moves despite the fact that the opponent does not. This is an algorithmic problem that merits experimentation; a simple approach is to maintain a separate MCTS 'instance' for each player, and branch a single round of simulation-improvement based on the selections made by either players' model. For example: to update from state S, we make a selection, then to model our opponent's selection, we make a selection for their simulation of us, as well as our own simulation, which will backup an estimate of the true value of the successor state of the opponent's move. In the worst case this may result in exponentially larger simulation rounds, but only experimentation will determine whether such an approach is feasible.

Power-Scalable: Unlike many algorithms, MCTS supports fine-grained control of an agent's strength: simply run a single additional play-out. This allows us to experiment with agents that simulate a given 'realistic' level of power, provided we are not interested in truly expert players. Moreover, we can simulate the strength of a specific target player, which will help ground our balancing approach in an easily interpretable context.

Generic: Basic MCTS runs on any finite, discrete, turn-taking, perfect information game. It need not be zero-sum, as each player simulates the other. Nor must it be finite-horizon: much of the foundational work was developed for MDPs [51]. The game may be stochastic if the the transition distribution can be queried or efficiently sampled. Moreover, countless variations of MCTS [19] extend the domain of feasible games — e.g. to real-time

games [26] and continuous state spaces [97]. Simultaneous play is studied in [87], but is not yet satisfactorily understood. (I expect an eventual solution may involve interleaving of both the recursive simulation of MCTS, and the primal-dual refinement of an approximate solution to the linear program for computing nash equilibria at each state.) Imperfect information is addressed in [92], though not to the level of sophistication of Koller et al's. [56] work on exact solutions to poker, etc. Extra-strategic skill (i.e. execution) can be simulated by modeling execution as a stochastic transition, though this ignores the complexity of players' differing knowledge of their own abilities.

Plausibly Realistic: We are a long way from understanding the methods and behavior of real human players, so any attempt to characterize an algorithm as human-like suspect. Nonetheless, there is something intuitively appealing about MCTS, which prevents us from immediately *ruling it out* as human-like. It seems to capture some essence of that human ability to explore promising paths and ignore unproductive paths, while avoiding forever being trapped in local optima. What's more, it's hard to imagine what *else* a human could do in a truly unfamiliar game. Let us imagine a game whose emergent rules are sufficiently complex that a human cannot a priori predict the game tree's structure, and can only mentally simulate the successor function to navigate the game tree. In such a game a human has no choice but to explore the game tree with what are effectively Monte-Carlo simulations. The only question of MCTS's realism in this context is whether its exploration/exploitation tradeoff and memory are realistic.

Of course, this thought experiment obscures much of the true difference between human and computational agents. Most games do bear similarity to previous games, which humans can exploit; humans quickly learn heuristics for unfamiliar games; and the rules for a game's successor function are typically enlightening enough to provide humans some insight on effective strategies. That said, any attempt to build 'realistic' algorithms is well-served by first placing humans and machines on a level playing field, then backing off only once realism at the pure level of search is understood.

It is also possible that a future variant of MCTS would take advantage of background knowledge from previous games - essentially learning the biases and tendencies of *game designers*. Some strategic guideline are intuitive to humans because they capture the assumption that the game is made to be playable, rather than hopeless complex. For example, a system might learn a rule of thumb as simple as "in a given game, taking an opponent's piece is typically good, or is typically bad". In other words, one won't often find a game in which the capture of pieces does not serve as a weak heuristic for either success or failure. Of course exceptions are always possible, but such rules act as a useful prior assumption over strategies to explore, and are likely present in the minds of most human game play-

ers. In video games, such guidelines are yet more prevalent, where the logic of a particular genre may be obviously apparent to frequent game players, but utterly unintuitive to new players. For instance, a player picking up a new 2D platformer title knows immediately that she probably progresses by moving right, and that touching enemies side-to-side likely kills her, but that jumping on these enemies may kill them. These 'rules' gestated in early Atari games, were crystallized in the first *Super Mario Bros* game, and made inescapable in the countless 2D platforms that followed. Today, game designers find value in building on traditional mechanics rather than rewriting them. The advantage goes beyond saving time and exploiting player nostalgia and familiarity. It also exploits player *ability*, allowing them to build skills and experience the joy of proficiency and improvement while continuing to play new and varied games.

5.5 A Balancing System using Human-Calibrated MCTS

Recall that the insight of restricted play is to reduce a variety of balance questions to the most classical form of balance: the success of asymmetric agents. As that evaluating asymmetry is so fundamental to understanding balance, I will focus our initial investigation on this single question. Subsequently applying results to the more general goals of restricted play will not be difficult.

That is, I will explore the feasibility of building a system to evaluate the balance of asymmetric starting conditions of a game. Accomplishing this goal will immediately facilitate the evaluation of some other forms of balance, (though the evaluation of their validity and usefulness will have to be performed independently). Some other forms of balance will require further work - namely, those involving a restriction in which one player has a fundamental lack of understanding of computational power, as discussed in Chapter 5.4. For the time being, we will be satisfied to look only at asymmetrical starting conditions, as the existence of such a system could already effect a huge change on the game design process.

Our immediate goal is to more carefully understand the objectives of a designer wishing to evaluate the asymmetrical balance of her game. For the time being, let us consider a hypothetical fighting game with two characters, Adon and Balrog. We have already characterized the asymmetrical balance (or *fairness*) of such a game as the probability that Adon wins a match against Balrog. The question that remains is then "for what agents?"

The insight for *which* suboptimal agents to use comes from considering the role that we hope a balance tool would play. Our aim is conservative: we would like a balance tool to be the first line of defense in evaluating the balance of a new game variant. Hence, our goal is to select a heuristic that will be approximately predictive of the win rate of real human players, (potentially playing with restrictions themselves).

A more basic question is then "which humans would we like to predict"? This question gets to the core ambiguity of the objectives of balancing, even for a simple question like fairness. In a perfect world, perhaps the designer would like all pairs of equal-skill players to see a 50/50 win rate. Yet in most cases this goal is not compatible with the basic intended form of the gameplay. Substantively asymmetrical characters will naturally depend on skill in differing ways, so that the skills of an intermediate player give one character an advantage, whereas the skills of an expert give another an advantage.

For example, in *Street Fighter*, it can be difficult for beginning players to learn to execute the activation motions of some special attacks, like "Down, Down-Right, Right, Punch", or "Right, Down, Down-Right, Punch". Such players will tend to do better with characters lacking these motions, who rely instead on 'charge attacks', which require only holding a direction for two seconds, then pressing the opposite direction and an attack. Indeed, weak players with many charge characters will tend to beat equivalently skilled players playing complex-command characters, despite the fact that charge characters suffer in expert play.

The question becomes even more fuzzy when we consider that skill itself is not controlled on a single axis. Players can excel in their reaction time, their opponent prediction, their high-level strategy, their patience, and so on. There are correlations between these skills, as all tend to improve with play. But the multidimensional nature of skill means that defining 'equivalent skill' itself becomes difficult, so that the goal of character parity in all equivalent-skill matches becomes that much more unreachable. There are potential paths around this problem: averaging over players with a certain degree of experience; characterizing players by their demographic percentile (in terms of win rate) with their preferred character; ranking players based on their performance with some third, established character.

Fortunately, one of the primary goals of competitive games is to reward player investment and skill. In asymmetrical games, one of the strongest methods for doing so is to provide skill-dependent choices, which are challenging to utilize, creating a disadvantage to new players but an advantage with practice. Mastering such choices gives players a clear goal for practice, which will reap rewards in the long-term, and can be a point of pride. In *Starcraft*, the Protoss are notoriously complex to exploit, and were designed to be so. In *Street Fighter*, a character like Ibuki is somewhat infeasible for new players, but a top-tier choice for experts, due to her precisely executable options when opponents get up from being knocked down. In first-person shooters, playing a sniper is typically a rather specialized skill, but which facilitates very quick kills if used correctly.

5.5.1 Utilizing Target Players

Rather than wrestle with definitional questions of skill, our answer to the question "who should we predict" is a pragmatic one: we should simulate a real player of the designer's choosing. Designers already have a sense of who they are balancing for; the existing design process requires it. They have a player (or collection of players) in mind when making design decisions, and in playtesting they pay the greatest attention to representatives of their target audience. For instance, some games will focus on expert players, others on casual players. We exploit the typical existence of such a target audience, by asking the designer to choose a 'target player', one who she would like to examine thousands of games from if she could. This player cannot be an expert, due to practical limitations. But if this player is *somewhat* skilled, then we should be able to simulate her for a breadth of games. [1]

Let us now assume that the designer does have a clear target player in mind; Alice. If we could produce an agent that plays like Alice, we would make the design evaluation process easier, facilitating rapid iteration. Yet as argued Chapter 5, playing like Alice is not necessary; all we require is that the agent plays *as well* as Alice. For this we employ MCTS. I propose to predict Alice's ability by running a series of *calibration matches* between Alice and MCTS. To find an MCTS instantiation with Alice's ability, we seek one that beats her in a given game variant with probability 50%. Note that in a fundamentally asymmetrical game, we will have to alternate Alice's starting condition

A straightforward way to find the right power level is to perform binary search on the running time of MCTS. Have Alice play matches against a given instantiation until her win rate has approximately converged. Then choose a longer or shorter running time, and play several matches again. Continue this process, narrowing in on a win rate of 50%. Once we are within a tolerable error (5-10% perhaps), we take the final MCTS running time, and fix that as our Alice-simulator.

Fortunately, there is a less time-intensive solution. If we assume transitivity of win-rates, we can have Alice play only the first set of matches, against a single instantiation. Once her win rate has converged to some value[2], we fix her opponent as the *target opponent*.

[1] In practice, we could choose multiple target players to use with our tool, but for now we concentrate only on that first pass of balance evaluation - a small set of numbers a designer can sanity check with before beginning playtesting.

[2] One challenge is that Alice may learn the behavior of MCTS. I hope that the stochastic nature of MCTS will prevent it form exhibiting simple, exploitable strategies. This could be studied directly through an experiment to test human adaptivity to MCTS. We have players play several 'training' matches, followed by a number of 'challenge' matches against MCTS. By having one bucket play their training matches against MCTS, and another against human opponents, we can discern whether players gain an advantage against MCTS through repeated play, beyond general experience.

We then perform binary search again, but without Alice in the loop, instead searching for a running time of MCTS that beats the target opponent with the same rate that Alice beat it. For example, if Alice beats the target opponent with probability 0.2, we play MCTS instantiations against the target opponent until we find one that also beats it with probability 0.2. *This* implementation is our Alice simulator.

There is an assumption underlying this process: that Alice's skill in the variant of the game used for calibration will transfer to future variants of the game. Yet this assumption is present even in traditional playtesting. Fortunately, balancing changes tend not to vastly change the mechanics of a game, so players should be able to apply roughly the same sets of skills. That said, there is a learning component. When a designer imagines Alice performing equally well in the new variant, she usually imagines Alice after a process of some adaptation and learning of the new variant. If the speed of some attack increases by 30%, Alice will have to take time to learn the implications of this change, and understand how to exploit the move as well as she did previously. When we calibrate an MCTS implementation to Alice's skill level, that implementation already takes into account the benefits of Alice's experience; as such, we hope that it would accurately model how well Alice would play the new game, *following* her learning process. If this turns out to not be reliable, a possible solution would be to explicitly utilize a learning-based MCTS algorithm. We could calibrate this MCTS implementation to learn a quantitative amount that approximates Alice's improvement over the course of several matches of a new variant. This approach will necessarily be weak, as human learning occurs in fits and starts, and is extremely hard to model. We nonetheless could potentially see some improvement in our predictiveness.

5.6 Wrapping Up Automated Analysis

In this chapter I have provided a template for an automated approach to balance analysis. This approach is conservative, in so far as it relies on restricted play, eschewing observational statistics in favor of measures of the impact of game dynamics. In this way the results of such a system have the potential to be more trustworthy than prior work in automated balance analysis. Nonetheless, I have advocated an equally conservative use of such methods. Even as artificial intelligence and our understanding of balance improve, we will not be able to fully capture the behavior of human beings. Often, it is these strange quirks or surprising creativity that have the greatest impact on the balance of a game. For this reason, human playtesting will remain the crucial checkpoint of a game's quality. We had best get as much out of human playtesting as we can. This is where automated analysis may have a tremendous impact: in finding those imbalances that we *do not* need human beings to discover, leaving humans as much time as possible to express their uniqueness in the games

that we do playtest. Once we do move to human playtesting, the questions of ambiguity in data to some extent grow. This work is the subject of the remainder of my work.

Chapter 6

ANALYZING HUMAN GAMEPLAY WITH DOMAIN-SPECIFIC CAUSAL MODELS

The role that data plays in influencing a game's development is currently in a state of transition. Early uses of data for game development grew out of business analytics, so naturally skewed toward the business interests of games: sales, downloadable content purchases, referrals, subscriptions, retention, and so on. In time these analyses evolved to capture more of the player's immediate experience: levels completed, trophies earned, hours played, etc. The goal, more often than not, was to connect these high-level behavioral statistics to business analytics, thereby finding opportunities for marketing and improving the game to maximize profit.

Only recently (as described in Chapter 3.1) have a small collection of developers begun to analyze in depth the properties of gameplay itself. By analyzing gameplay I refer to visualizing and modeling the detailed statistical properties of the gameplay dynamics that emerge from game mechanics and player choices. I argue that making this connection is a fundamental step in the maturation of game development. Designer creativity and intuition remains the fundamental force by which games can be designed, and even balanced. Yet I believe that designers act best as the glue between two processes through which analytics are invaluable: gameplay analysis and behavioral analysis. A designer's gameplay vision provides the goal for development, but gameplay analysis helps refine the game until the actual gameplay matches that vision. On the other side, this gameplay vision is intended to elicit a certain experience (and possibly behavior) in players, and behavioral analysis can help determine whether players actually enjoy and respond to that gameplay as intended, and suggest potential changes to the gameplay vision. Although behavioral analysis is potentially as important, this thesis is only concerned with gameplay analysis, the less studied and more technical of the disciplines.

In this chapter I will describe an elementary use of restricted play in human gameplay analysis. Although restricted play acts only as a new interpretation of prior methods, this interpretation motivates a new, more conservative utilization of the methods, which will consume much of the chapter.

6.1 Retroactive Restrictions

To begin discussing the transition from analysis of simulated players to analysis of real humans, we must ask a natural question: is it even feasible to restrict human players? Practically, it may be hard to motivate players to play a game in large numbers if their form of play is restricted. Philosophically, it may even be impossible to impose certain restrictions, such as those that constrain the search or strategic properties of a player. Although some such explicit restrictions may be possible, in this dissertation, I take an alternative approach. Rather than explicitly restrict humans, I investigate those features of players that can be identified after the fact of play, acting as a sort of retroactive restriction. This practice must be approached carefully, however. The majority of gameplay features - such as a player's kill rate or time spent jumping - are not restrictions but rather measurements, which may themselves be influenced by other gameplay features. The goal of restricted play is to understand whether winning is *possible* under fixed conditions, rather than incidentally correlated with wins. For a feature like time spent jumping, making such a claim is effectively impossible, because a correlation between jumping and winning offers no causal evidence. For instance, a player who is losing may jump more due to a sense of panic, even if it has no impact either way on her performance. Alternatively, the best players may tend to use a strategy involving a great deal of jumping, even if there exist several other equally viable strategies.

In this chapter I discuss a natural legitimate retroactive restriction: asymmetric starting condition. Nearly every competitive game exhibits some asymmetry between players, and many modern video games strive to produce richly asymmetric forms of gameplay. I will call the various asymmetric conditions of a game 'characters', as they are called in fighting games. However, the principles are the same regardless of whether we speak of board games, fighting games, first-person shooters, or strategy games. Analysis of the relative strength of characters has traditionally been the foremost concern of balancing, with good reason. Players choosing a character are 'locked in' to the starting condition for the duration of a game. What's more, players spend substantial time building skills with a given character, and hence can be adverse to changing. Designers often work extremely hard to ensure that their characters are balanced with respect to one another. Fortunately, as a signal, win rate with a character is uniquely meaningful. Because players choose their character before the onset of the game, it is not at risk of being influenced by other in-game features. Of course, there are social contexts and games in which gameplay can influence future character choice, but I will ignore these for the time being. For the most part, we can assume that players playing a given character are working to maximize their win rate. Thus the sample of games in which players chose a given character is roughly equivalent to a dataset in which

players were *restricted* to playing that character. The major nuance ignored by this claim is that since players typically choose characters freely (rather than randomly), other player qualities may be correlated with certain character choices. Chapter 7 is all about overcoming that flaw in reasoning. That said, in looking at character win rate, we essentially take the standard practice of human gameplay analysis, with all its strengths and flaws.

The major advantage that this chapter gains over prior work in human gameplay analysis is the recognition that other statistical features may *not* be informative. Everything from jumping time, to kill rate, to control over a map region may be incidental to observed play, and not intrinsically connected to win rate, the primary concern of balance. Yet it is simply unacceptable to throw out the rich multitude of data features that can be observed during gameplay. Indeed, one of the primary benefits of transitioning from simulated gameplay to human gameplay is that we can 'trust' data beyond win rate. Even if players needn't play in a certain way, the fact that real humans *do* play that way is valuable information, as it would not be for simulated players. As such, we would very much like a principled method for exploring and making sense of the broad variety of data features. At the very least, we would hope to use ancillary statistics to inform the causes of a character's observed win rate. If one character wins 40% of games, and another wins 70% of games, this information is only of limited use without an understanding of how this difference arises, and, ideally, what can be done to address it.

6.1.1 The Value of Domain Knowledge

My contribution in this chapter is an argument for understanding the causes of win rates through a simple but controversial mechanism: hand-built models based on domain knowledge. Such models sit in sharp contrast to machine-learned regression, which may describe the current data effectively, but cannot step beyond the biases of the current data.[1]

This can best be explained by an analogy to the natural sciences. Newton's theory of gravitation is not simply a statement of correlation between velocities at various times. It is in fact a model of a cause and effect, describing the acceleration that will occur as a result of an object's location. Centuries of observations have approximately confirmed this theory, yet under general relativity the theory is known to be false, and indeed unpredictive in certain contexts. Yet despite being neither true not perfectly predictive, it remains valuable to treat Newtonian gravitation as a causal model, in the following crucial sense: it predicts the effects of an intervention. If I move an object away from the earth, it will be dragged back to it at a certain rate. Our willingness to treat the model as causal requires a

[1]It is possible to present this discussion in terms of strong priors on machine learning algorithms, but doing so would be both more complex and counter-intuitive than my approach.

certain leap of faith, combining measurement, reasoning, and some amount of philosophical induction.

Much of this dissertation treats balance evaluation as a sort of data science, but in this particular instance, I advocate treating balance evaluation as more of a traditional natural science, just as we do with Newtonian physics. We wish to understand the causes of a certain character's win rate, in terms of other observable variables. The emergent dynamics of the game are sufficiently complex that we can never hope to capture a true model in any intuitive way that can be mentally captured and used to inform design. On the other hand, we are privileged to knowi the precise underlying mechanics of the game - and it is from these mechanics that we can hypothesize a *simplified* model for the emergent dynamics. Through a combination of reasoning, induction, and verification of the model's predictive power, we can come to accept it as a useful, though imperfect, causal model.

This chapter is constituted of an exposition of just such a process. I decompose the forces at work in influencing a character's win rate in *PSABR* - simple though they may be - and build a series of progressively more accurate causal models. If such a model is believed, even taken with a grain of salt, it brings order to what would otherwise be a very messy, confusing process: navigating dozens of variables and visualizations to understand the aggregate forces at work in a game. In fact, the models I describe can be thought of as proxies for restricted play. They hypothesize the true effect of variables like kill rate and death rate on win rate, without ever restricting a player's use of those variables. Note that such a practice is only valid because the model is not derived primarily from the data (which can be greatly influenced by confounding factors), but from our domain knowledge of the game itself.

6.2 Overview of Playstation All-Stars Battle Royale

I spent the summer of 2012 (and subsequent consulting work) as a technical game designer on *PlayStation All-Stars Battle Royale* (*PSABR*). *PSABR* is published by Sony Computer Entertainment America, and developed by SuperBot Entertainment, an independent studio founded in 2009 and 'incubated' by Sony Santa Monica. *PSABR* is a 'multiplayer brawler' featuring characters from the three Sony PlayStation consoles. The multiplayer brawler is a variant of the fighting game (popularized by games such as *Street Fighter* and *Tekken*), in which up to four characters fight each other on a fixed stage. The presence of four characters makes certain design changes natural: multi-level battlefields, bidirectional blocking, a wider camera, and a fine-grained scoring system. These features can be observed across games in the genre, such as *Jump Superstars*, *Teenage Mutant Ninja Turtles: Smash-Up*, and most

notably the *Super Smash Bros.* series, which crystallized and popularized the genre.[2]

The most interesting gameplay distinction between *PSABR* and *Super Smash Bros* comes in the way kills are achieved. In *Super Smash Bros*, the goal is to knock your opponents off the battlefield; each subsequent attack makes opponents more physically responsive to attacks, increasing their knock-back distance and probability they will be knocked off the level. In contrast, the only way to defeat opponents in *PSABR* is to perform character-specific 'super attacks' on them. These attacks kill instantly, but can only be used by building up a 'super meter' (called 'AP') through landing regular attacks.

This change has surprisingly sweeping effects on the Nash Equilibria of the game, and hence on the entire experience of playing the game. The kill system in *Super Smash Bros*, (and indeed any multiplayer health system) can exhibit a 'Tragedy of the Commons': all players may hesitate to put themselves at risk to attack their opponents, since they can wait for their other opponents to do so, then 'steal their kill' when an opponent is near death. This can result either in passive play-styles, which tend to be less enjoyable in this variety of game, or player frustration when their kills are stolen. *PSABR* sidesteps this problem by causing players' attacks to increase their own power, rather than decrease a certain opponent's strength. This is in essence equivalent to giving each player a separate health bar for their opponents in aggregate, albeit one that must be followed up with a successful super move to result in a kill. Hence players are never discouraged from hitting opponents, resulting in a generally more active and aggressive game. (This has its tradeoffs: the lessened effects of target selection removes a certain source of strategy from the game.)

The standard competitive mode of *PSABR* is a timed four-player free-for-all match, taking place on one of fourteen randomly chosen maps. Over the course of three minutes, players attack each other to gain AP and spend that AP on supers to kill their opponents. At the end of three minutes, the player with the largest score wins, where score is equal to twice a player's number of kills minus their number of deaths. In case of a tie for first place, the game goes into successive one-minute overtime rounds with increased AP generation. A large fraction of play takes place in a monthly online tournament season, in which players gain and lose belts and leaderboard position according to their performance. Both belt and leaderboard are determined by wins and losses against other players, in a way that considers the opponent's current belt. Belt and leaderboard progression are determined by specific placement each game, so a second or even third place placement has the potential to raise a player's position.

All the graphs in this chapter and Chapter 7 come from real data at various phases of

---

[2]Many similar elements can be found in the older games from which *Super Smash Bros* draws, such as *Nekketsu Kakutou Densetsu*, *The Outfoxies*, and *Power Stone*.

testing of *PSABR*. For security, exact labels have at times been removed, but graphs begin at 0 unless otherwise noted, so that relative scale is clear.

6.3 Data-Guided Design at SuperBot Entertainment

Among other responsibilities, I was responsible for redesigning and employing *PSABR*'s gameplay telemetry system. The system and our analytic processes both undergo frequent iteration; as such I will describe the system at the time of writing. For each game played, a collection of events is uploaded to a server and stored in a relational database. These events generally take two forms: gameplay events, which represent individual occurrences during the course of the game, and meta-events, which record information about that particular game as a whole. Gameplay events include the use of a given attack, a player's death, the pick up of an item, etc. Meta-events provide background information for that game, such as the stage, player information, and game-mode, as well as summary results such as final scores, aggregate time in certain states, and network performance.

The uses of this data are broad, but here I am primarily concerned with its use for balancing. *PSABR* is balanced using a variety of sources of games, including designer play sessions, internal tournaments, outside focus tests, external private beta, external public beta, and the shipped game. Each of these entails differing analytical methods, but the lessons of this section primarily concern large-scale data analysis, so will be concentrated on the latter three domains. Our analytical methodology for this data consists of iterative question answering and refinement. As a member of the design team, I speak frequently with the other designers to understand their current balance questions and concerns. From these I attempt to formalize the most pressing yet tractable question, and formulate a visualization which I believe might answer this question. I generate this visualization using SQL and Tableau, and use domain knowledge and drill-downs to verify its correctness. I then look for signal in the data, and iterate on this graph until a message becomes clear. This graph is then shown to the design team and discussed, along with any previous graphs we have developed. Each of these is kept up-to-date, and watched by myself and two others for noteworthy changes, particularly when the game or playerbase changes. When the design team is considering major changes, I prepare detailed reports evaluating the gameplay surrounding this topic, and at times make concrete recommendations. At times I even work with the other designers directly to decide on balance changes, using my knowledge of the aggregate gameplay data to guide these decisions.

When I arrived at SuperBot, telemetry in *PSABR* was partially implemented, and my first task was to expand the design for which data points would be collected, in what contexts. From that point on, both the design and the actual implementation of the telemetry

system were constantly being expanded, with priorities shifting as the design questions we wished to investigate changed. Similarly, the questions investigated and the means of inquiry continuously expanded and improved, as design issues changed, analysis resources freed up, and even as gameplay itself changed, necessitating new data formats, queries, and analyses. It should be emphasized that the iterative nature of this data collection and analysis is fundamental to the process. Design is near-universally an iterative process, and here we have multiple layers of iteration: the analysis is being improved and iterated as its most useful qualities are identified, but the target of this analysis is changing as well, as the game is iterated upon. Fortunately, there is some predictability, as the analysis and game design form a dialogue: insights from the analysis help shape the direction of the game.

6.4 Case Study Part 1: Measuring Character Strength

The value of data is multitudinous, from business concerns, to debugging, to player preferences. My own analytics work at SuperBot was concerned primarily with balancing, taken as broadly as possible. In practice this means evaluating the player experience in aggregate, to learn what the game is like to play for various kinds of players: what experiences those players encounter, what choices they make, and how those choices impact the game. This characterization is almost vacuously general, and captures the way my work might map onto a very different game. In the case of a game like *PSABR*, it is possible to provide a more concrete description of our analysis goals, which still captures the principle of answering the question "Have you made the game you sought out to make?" where a game is not just a set of mechanics but also a set of emergent dynamics.

The questions we seek to answer in *PSABR* are diverse, and run the gamut from understanding the value of particular attacks, to the popularity of particular characters, to the role of experience in a particular play mode. However, in this section I will focus on the foremost question of analysis in many asymmetric games: the relative strength of the starting conditions. With 20 characters in the initial release of *PSABR*, it was essential that all characters be competitively viable, and more crucially, that no small set of characters be overpowered. This is all the more important in a mash-up game, where fans of a given character come to the game planning to play that character; even a single dramatically weak character could be taken very harshly.[3]

In this section I detail the process by which we analyze character strength (and much

[3]Evaluating character strength is made even more complex by the existence of a 2v2 mode: no *pair* of characters should be dramatically overpowered or underpowered, since a pair of friends may have their own characters they wish to play. However, in this thesis I focus on the problem of balancing Free-For-All play, in which each of the four players competes against all others simultaneously.

more) at SuperBot Entertainment. The methods are indicative of our general process for answering any broad balance question. The focus is on the decomposition of key variables, and attempts to approximate the causal relationships between them. The process is done by hand, as the goal is to achieve rich understanding of the game's forces, not diagnose, measure, or remedy any one element. Although in certain aspects the analysis could be automatically *guided*, at this time it would not bring a substantial benefit. The goal is understanding, and this is best achieved through manual examination of an array of visualizations. The challenge is in taking the broad variety of visualizations available, and knowing what to look at, when to look at it, and how to add it to a mental picture of the dynamics of some variable. In the subsequent section I will argue that this guidance is best acquired from an explicit model of the causal relationships between variables.

6.4.1 Gameplay Decomposition

Evaluating character strength involves layers of drilling down through the 'causes' of successive statistics. Constructing a mental picture of such forces is essential for any attempt at true understanding of a game as a system. Games have a variety of amenabilities to such a model. Basketball is famous for resisting efforts at quantification more than baseball has, and this is often said to be due to baseball's comparatively decomposable play structure [42]. In baseball, each potential run is somewhat independent from one another, whereas the attempts at a basket are substantially correlated. Moreover, each chance at a run is separated into four phases (pitching, batting, running, fielding), which can be studied roughly independently. There is no such decomposition in basketball, where actions overlap more fluidly and are hence further correlated.

PSABR benefits from a somewhat neat decomposition. A more traditional fighting game like *Street Fighter* is more difficult to decompose, as the role of nearly every action is simply to deal damage or prevent damage. And yet, such a comparison is somewhat disingenuous. At the heart of *PSABR* lies a system much like *Street Fighter*'s (landing basic attacks, blocking others); it is just that it is repeated and layered with other elements within the course of a single game. Does *PSABR*'s decomposability make it easier to analyze? Similarly, does baseball's decomposability make it easier to analyze than basketball, even though the dynamics of a single phase such as batting are rich and opaque, and must themselves be abstracted into simple numerical values? In fact, I would argue yes, to some extent.

For argument, take as an assumption that each game has core mechanical loops that resist analysis, whose complexity is so great that understanding of their dynamics will be limited. In this case, the game itself becomes more predictable and comprehensible by

repeating and layering these loops. For an overly simple example, imagine that a *Street Fighter* game consisted of a best 10-of-19 short rounds structure, rather than the traditional 2-of-3. This structure would smooth out the noise of individual player-and-character matchups, making the true win rates of the core loop clearer. It would also shift more of the cause of a given win or loss onto the interactions between the rounds, which are not truly independent, due to sharing 'super meter'. Assuming that the individual round gameplay is somewhat analytically impenetrable, this gives a designer a target to comprehend and focus on: the interactions between the rounds. Even leaving the fundamental gameplay intact, it is possible that the game could be balanced somewhat by solely tweaking these interactions.

In *PSABR*, we indeed take advantage of this decomposition. The components are by no means independent, but they can be tentatively examined one-at-a-time. We begin by looking at each character's win rate. Although wins are a coarse measure, which hide margin of victory, they have the advantage of reflecting the true object-of-concern, (winning) and of not being subject to changes in relative score. In fact, even measuring win rate is somewhat ambiguous in a 4-player game. One way to do so is to examine the rate of placement (1st, 2nd, 3rd, or 4th) of each character, as in Figure 6.1. Unfortunately, such a graph is somewhat hard to interpret as a general characterization of win rate, since it is hard to know whether fewer 1st places but comparatively more 2nd places should be considered stronger. Moreover, such a graph is agnostic toward ties; although by design a game of *PSABR* must have a single winner, it may have three 2nd-places, or two 2nd or 3rd-places.

6.4.2 Measuring Character Strength with 'Win Rate'

A more holistic measure is a character's 'win rate': the fraction of opponents whom she places higher than (where a tie is considered half a win). The win rate W_c for character c is defined simply by $W_c = E[1 - (R_c - 1)/3]$, where R_c is c's placement in a given match. A plot of each character's win rate can be seen in Figure 6.2. This formulation has the advantage of more directly capturing a competitive player's likely primary concern, since each opponent defeated awards the player points with which to advance the leaderboard. Yet individual rank probabilities are still valuable for more detailed analysis, because it shows that some characters have fundamentally different success dynamics: one character may be able to consistently get 2nd place, but have a hard time getting 1st place. This phenomenon is surprising at first, but is understandable due to the heavy interaction and assymmetry between characters. For instance, a given character may be better at siloing off and defeating whichever opponent is not currently being attacked by the player in the lead.

Note that the above treatment of character wins is totally uncontextualized. It specifies

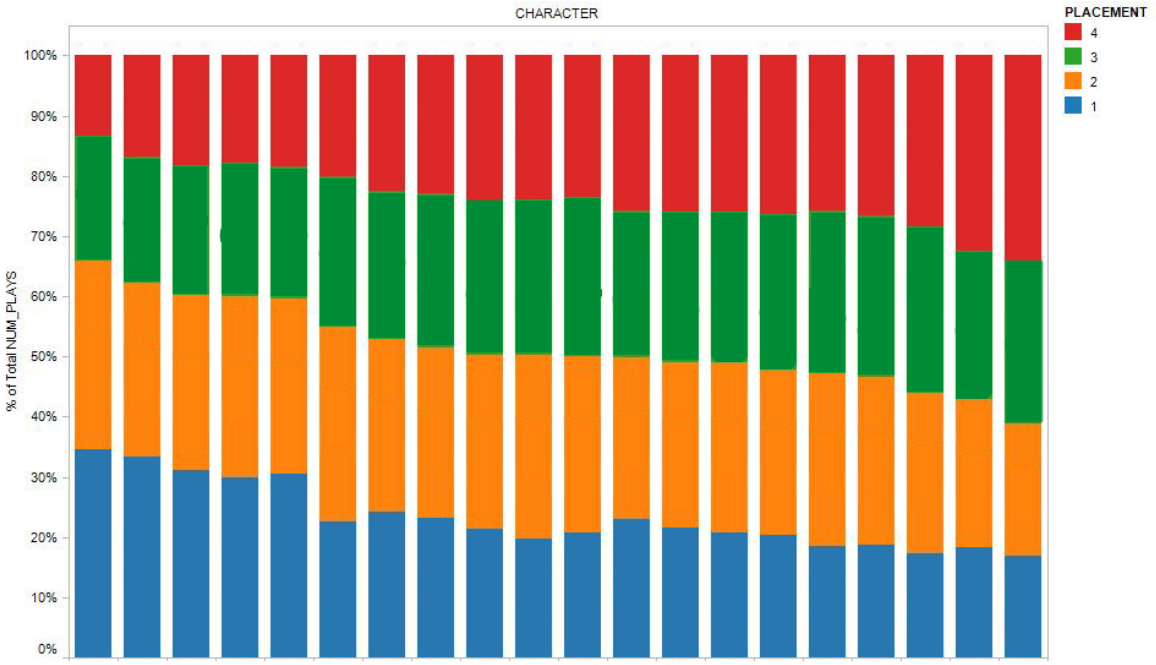

Figure 6.1: Placement percentages of each character. Each column represents one of the twenty characters; the four colors indicate the percentage of games in which they achieve first, second, third, or fourth place. The characters are ordered by their combined probability of getting first or second place. Observe that some characters are better at others than securing at least second place, even when another character may be better at getting first place. (This holds beyond the margin of error.)

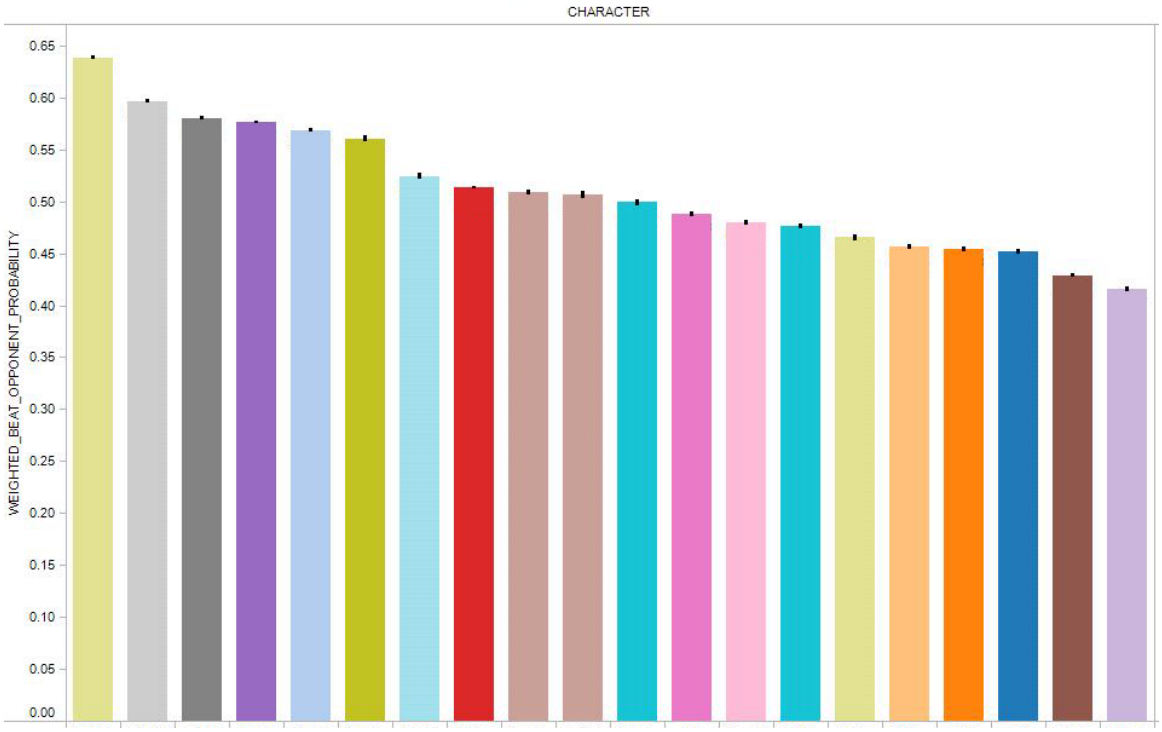

Figure 6.2: The average 'win rate' of each character, defined by $W_c = E[1 - (R_c - 1)/3]$, where R_c is c's placement in a given match. This aggregate statistic captures the probability that a character places a higher than a random given opponent.

nothing of the players playing these character, nor even of the characters played by these opponents. As seen in Chapter 8, in some cases it may be more appropriate to think of a character's strength as their potential frequency of play, rather than their win rate in the current metagame. Nonetheless, win rate is a good starting point that is easy to interpret for both the analyst and designers. It also often the most useful measure, since once a game has gone live the *current* metagame may be of the greatest concern.

Once win rates are somewhat understood, the next step is to understand the causes of these win rates. This is necessary to get some idea of how to alter a given character's win rate. However, even if the current win rates are thought to be acceptable, understanding their causes can give insight into the likely *stability* of the win rates. For instance, if a character is seen to be losing in a way that is easily rectifiable once players have discovered some trick, then the designers may choose to give the players time to hopefully discover this trick. Indeed, such a principle is at the heart of each drill down from some statistic into its constituent statistical causes. Formulating an accurate model for wins is the subject of the next section.

6.5 Case Study Part 2: Building Causal Models

I have argued that interpreting the results of character win data in a meaningful way requires an understanding of how those wins arise. In this section I build up a series of successively more useful models for the causes of wins. These models can then serve, first and foremost, as a *mental* mechanism for relating statistics to one another. The models essentially act as the glue that ties together a breadth of confusingly varied visualizations.

The limitation of these models to guiding intuition is somewhat fundamental. The models are approximate, and they are also almost certainly overfit. Yet the key realization is that they only seek to inform on a second-order effect. For all questions about actual data, designers can turn to just that: the data. We are modeling relationships between *equally observable* variables, such as wins and score. When the model is not accurate, the data will quickly overrule it. The goal of the models is to suggest an approximate casualty between the variables, thus establishing a more complete picture of the gameplay.

In Chapter 6.6 I will explore extending the models of this section to the point that they may have actual predictive power for unknown data. Yet in the process, the simplicity of intuitive models is lost, making the work of this section an equally important complement. I now begin by examining the relationship between wins and the most immediately related variable: score. All of the work here will take into account domain knowledge of the gameplay whenever possible, in as much generality as is feasible.

I present this section in high detail, to elaborate the process of domain-based mathe-

matical models for game analysis. The details of this chapter are not relevant except to a few genres of game, but the modes of reasoning are somewhat universal.

6.5.1 Modeling Win Rate with Score

Score is the ubiquitous metric by which a game's four players are ranked. The player with the highest score wins, and all others place according to their score, with the possibility of tying amongst each other. This is because games go into repeated overtime until some player has the unique high score, with overtimes modified so as to boost scores.

Scores are computed by the formula $s = 2k - d$, where k is the number of kills, and d is the number of deaths. In this way active, aggressive play is encouraged, but in a measured way. Even disregarding the the details of how score is computed, one can guess that as a raw metric it is informative for understanding wins, but not entirely so. This is because, as in nearly any multiplayer game, scores are correlated. Measuring a character by their score is misleading if the highest-scoring characters substantially bring up or down the score of their opponents by their presence. A high-scoring player may be a frequent loser if all other scores tend to go up as well. Even if scores had simple, predictable correlations, looking at means would might be insufficient, since scores beyond the highest or lowest opponent scores are irrelevant to wins. For this reason it is sometimes necessary to examine distributions of scores directly, or more rich distributional graphics such as box plots.

In other words, score is a partial source of information, and must be understood, but cannot be taken in isolation. When the ranking of win rates (or the disparity between them) does not correspond with the relative scores, we can directly examine the effect of a given character on its opponent's scores. In other words, we can measure the average score of players in games in which at least one opponent is a particular character. If this average is higher or lower than an unconditioned average, it will help shed light on the reasons for a gap between a character's win rate and score. Again it is possible that averages will be uninformative, but not likely. We have found that one character's kills of another tend to be controlled by the individual's kill rate and the collection of characters in a match. Hence the average score is mostly sufficient to describe the presence of one character on another.

Nonetheless, the first step in understanding the causes of win rates is to formulate a model, however weak, of the relationship between wins and scores. The idea is that this model serves as a guide for mutual interpretation of win data and score data. It lets us approximately understand why one character's wins take the value that they do *given* that their scores take the value that they do. As I have said, such causal relationships are crucial to understanding how a variable of interest might change with an evolving metagame, respond to upcoming design changes, or be intentionally manipulated directly.

The simplest method for establishing a relationship between scores and wins is to consider the mean of each variable, on a per-character basis. There will clearly be a correlation between these variables, though the extent is not clear. Figure 6.3 shows the strength of mean score as a regressor for win rate, for each of the game's twenty characters. Despite the broad variety of average scores, there is a near linear relationship, with $p_c \approx 0.286 + 0.072 \cdot s_c$, where p_c is character c's win probability, and s_c is c's mean score. The root mean squared error is 0.0127. In other words mean score already predicts the chance of having beaten a given opponent within a probability of around 1%. Of course, there are some characters who clearly deviate more than others. Upon investigation, these turned out to mostly be the characters who have an unusually number of those deaths. Those who die very frequently tend to have their win rate overestimated by their mean score, and those die infrequently have their win rates underestimated. In Chapter 6.5.2 I will discuss the causes for this.

We can further refine our understanding of the relationship between win rate and score by considering the fundamental cause for an imperfect correlation. If a character wins less often than another with the same mean win rate, there are only two possibilities. The first is that their score *distributions* have differing shapes, so that, for example, the first character's mean is partially explained by a few unnecessarily high values. If, on the other hand, their distributions are similar, then the only remaining possibility is that they have differing effects on the mean of opponents' scores. When a character's presence in a game alters the scores of other players, it changes the baseline which the player must meet to place, thus decreasing the overall win rate.

Fortunately, we have a very direct way of considering this possibility. We can measure the mean change in opponent scores in games in which a given character is present. This change can then be subtracted from that character's plotted score, representing a sort of normalized score. Concretely, if o_c is the mean score of all players having c as an opponent, and s is the mean score of all players, we define the opponent-normalized mean score of by $s_c^* = s_c - (o_c - s)$.

Even more than previously we are making assumptions about the uniform distribution of scores. We essentially assume that the average score difference given to each opponent by a character's presence is allocated to each opponent equally. Yet we are not attempting to build a realistic model of gameplay; if the values are predictive, this is sufficient. Indeed, this prediction is *very* effective. Figure 6.4 shows the relationship between so-called 'opponent-normalized' scores, and win rates. Adjusting characters' scores by their opponents' score differences shifts some characters back and others forward, so that they settle onto a much cleaner, albeit slightly less steep line. This regression is defined by $p_c \approx 0.329 + 0.058 \cdot s_c^*$, and root mean squared error 0.0017. The opponent-normalized score gives an 87% reduction

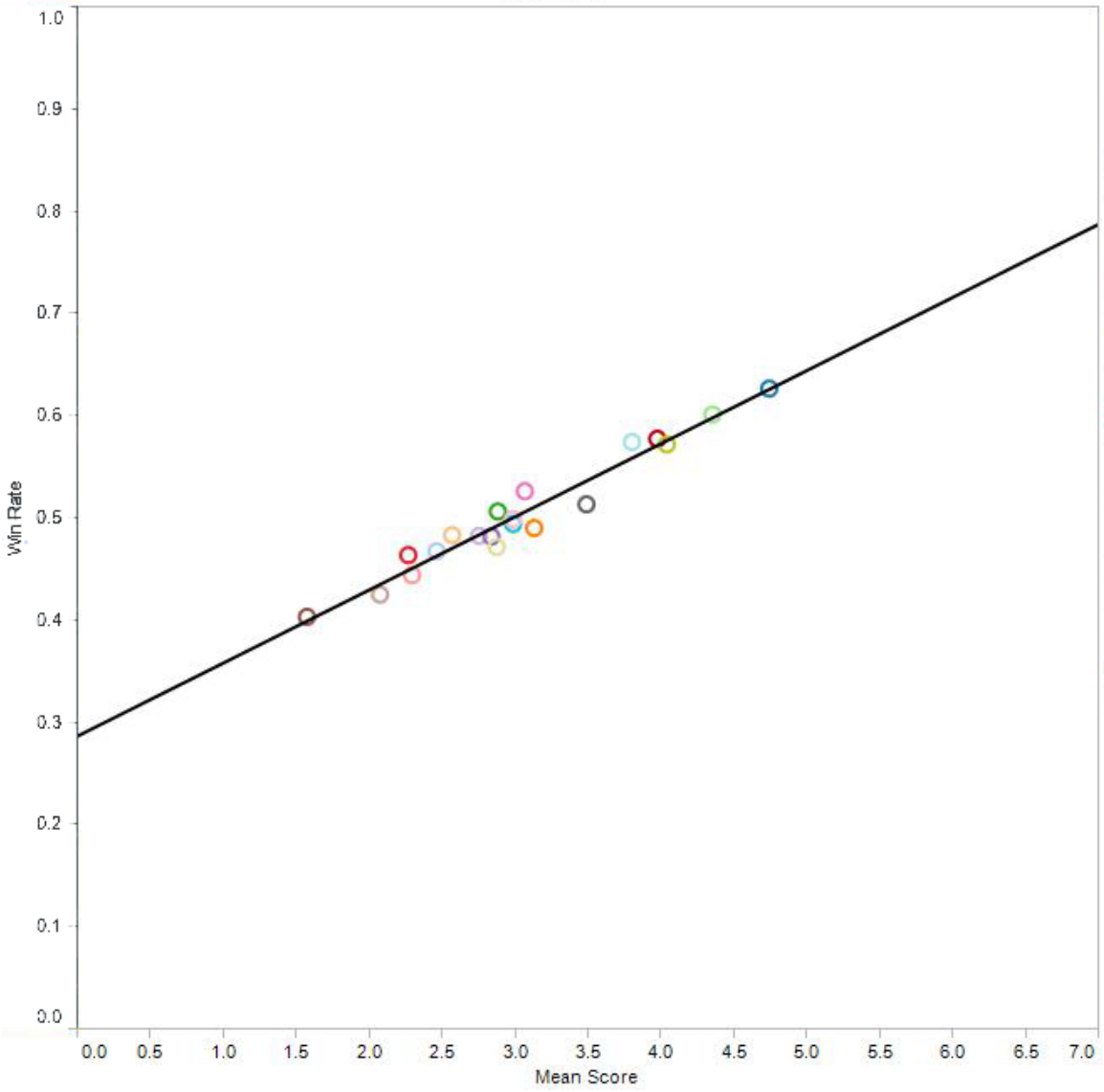

Figure 6.3: Scatter plot of all each of the 20 characters according to two statistics. The y-axis specifies a character's actual win rate. The x-axis specifies their mean score per game. The linear regression is somewhat accurate.

in RMSE, despite the simplifying assumptions of working with means.

Although there are two non-trivial deviations from the regression, it is more than sufficient as an expression of the relationship between scores and win rate. The very fact that win rate can be so simply predicted is surprising in itself, and helps designers reason about the determining factor for winning. Understand a character's scoring potential and their effect on opponents' scores, and you understand their winning potential. What's more, the relationship is so tight that - for some range of scores - it can be used to predict the win rates of future character modifications for which score is understood. It is to some extent possible that score could be understood without win rate, but this will be addressed in Chapter 6.5.2.

One important caveat is that this analysis was not performed with cross-validation. Each character participated in the 'training' for the predictor to which it is fit. Yet for a 2-dimensional linear function, the potential for overfitting is extremely minimal. The removal of any of the 20 characters from the regression process would have a tiny effect on the resulting function, and consequently on its predictiveness.

6.5.2 Modeling Score with Kills and Deaths

Although player and opponent scores turn out to be excellent predictors of win rates, there is a further challenge, given that our goal is to understand the *gameplay* causes of win rates. Scores themselves are somewhat difficult for designers to reason about. A given score conveys an entire 1-dimensional space of possible kill and death rates. Kills and deaths are the behavior that designers reason about and manipulate directly, whereas score is a derived expression. As such we have good reason to delve further, attempting to establish a direct relationship between kills/deaths, and scores. Doing so would be give a much more effective tool for balancing, as predicting a character's new kill rate is often quite feasible. For example, a 10% reduction in the cost of super attacks will allow 1/9 more super attacks to be performed, (modulo some error due to the discreteness of super attack costs) thus resulting in 1/9 more kills.

The first step in connecting kills and deaths to win rates is to attempt to replicate the result of the previous section, effectively by predicting the effect of kills and deaths on opponents' scores. From the beginning we know that the relative 'makeup' of a character's score (for example, many deaths and many kills, or few deaths and few kills) will have an effect on opponents' scores, since deaths benefit a given opponent more than they hurt the player, and kills benefit the player more than they hurt a given opponent. This is a fundamental property of almost any multiplayer scoring system in which players score points off of opponents. Hurting an opponent is not as beneficial as benefiting one's self;

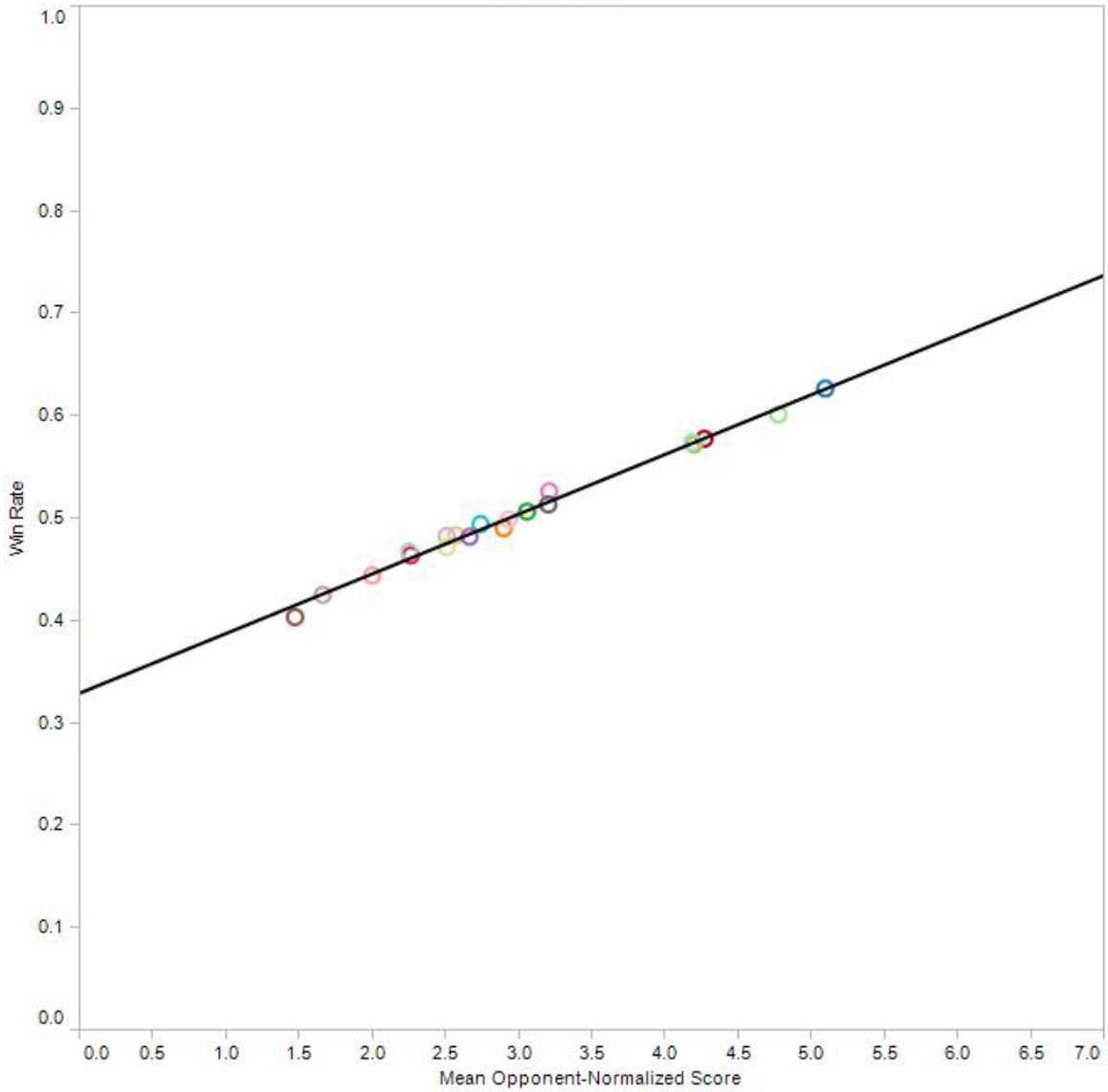

Figure 6.4: Extension of Figure 6.3, in which x-axis is now a modified score, adjusted by the average score difference this character's presence effects on each opponent. The best linear model is substantially more accurate than it is on raw scores.

individual player pairs' actions are not zero-sum. I will now quantitatively characterize this relationship, using only the definition of the rule set and some assumptions about play, rather than any data.

For greater generality, let us consider an n-player free-for-all game, with score defined by $s = \kappa k - \delta d$. Define s_c, k_c, d_c to be the mean score, kills, and deaths of character c, let $s_{o(c)}, k_{o(c)}, d_{o(c)}$ be the mean score, kills and deaths of c's opponents, and let s, k, d be the mean score, kills, and deaths overall. Then since c's kills are divided among $n-1$ opponents, and deaths are attributed to $n-1$ opponents, we have

$$s_o = \kappa k_o - \delta d_o = \kappa(\frac{n-2}{n-1}k + \frac{1}{n-1}d_c) - \delta(\frac{n-2}{n-1}d + \frac{1}{n-1}k_c),$$

by operating under the (not quite correct) assumption that c's presence has no effect on the kills between players other than c. Note that $k = d$, because for each kill there must be one death. So now we can estimate the opponent-normalized score of character c as

$$\begin{aligned} s_c + (s - s_o) &= \kappa k_c - \delta d_c + \kappa k - \delta d - \frac{1}{n-1}[(n-2)(\kappa-\delta)k + \kappa d_c - \delta k_c] \\ &= (\kappa + \frac{\delta}{n-1})k_c - (\delta + \frac{\kappa}{n-1})d_c + \frac{1}{n-1}(\kappa - \delta)k. \end{aligned}$$

In other words, we may estimate the opponent-normalized score by a linear function of a character's deaths and kills, and the average number of kills overall. This can be done *without* explicit reference to opponents' scores. But how well does this estimate predict the actual values of opponents' scores? Quite well. Instantiating the above formula with $n = 4$, $\kappa = 2$, $\delta = 1$ for *PSABR* gives $s_c + (s - s_o) = \frac{7}{3}k_c - \frac{5}{3}d_c + \frac{1}{3}k$. We can plot opponents' actual scores against these normalized scores, as in Figure 6.5. More aptly, we can plot the win rate against these estimated opponent-normalized scores, as in Figure 6.6. Both the regression ($p_c \approx 0.322 + 0.060(\frac{7}{3}k_c - \frac{5}{3}d_c + \frac{1}{3}k)$) and its quality (RMSE 0.002) are nearly identical to that of the true opponent-normalized scores. Yet the quality is not *quite* as good; there may be some explanation for opponent scores which is not yet captured.

6.5.3 Modeling Kills and Deaths with AP Generation

We are now in a position to ask why some characters' opponent scores are not fully determined by their own wins and losses. This seems to point to some mechanical facet of gameplay that directly affects opponents' killing and dying potential even outside of interactions with the character in consideration. Indeed, we know from the game's design that such phenomena should exist. One character, Parappa the Rapper, is able to generate orbs containing AP that any player can pick up. In principle, this move should bring up the

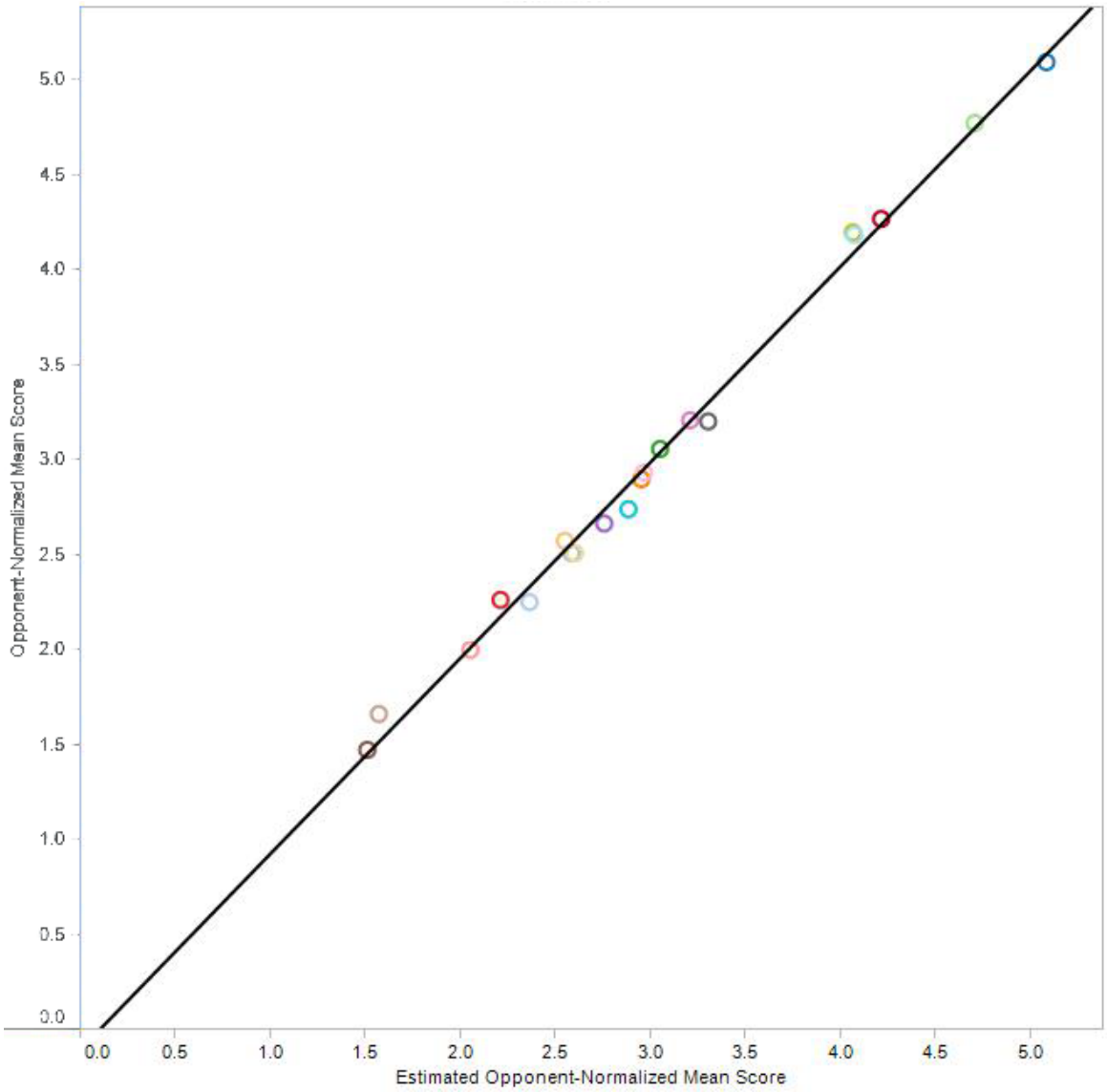

Figure 6.5: Plot of the relationship between average opponent scores for a character and estimated average opponent scores, solely as a function of a character's kills and deaths.

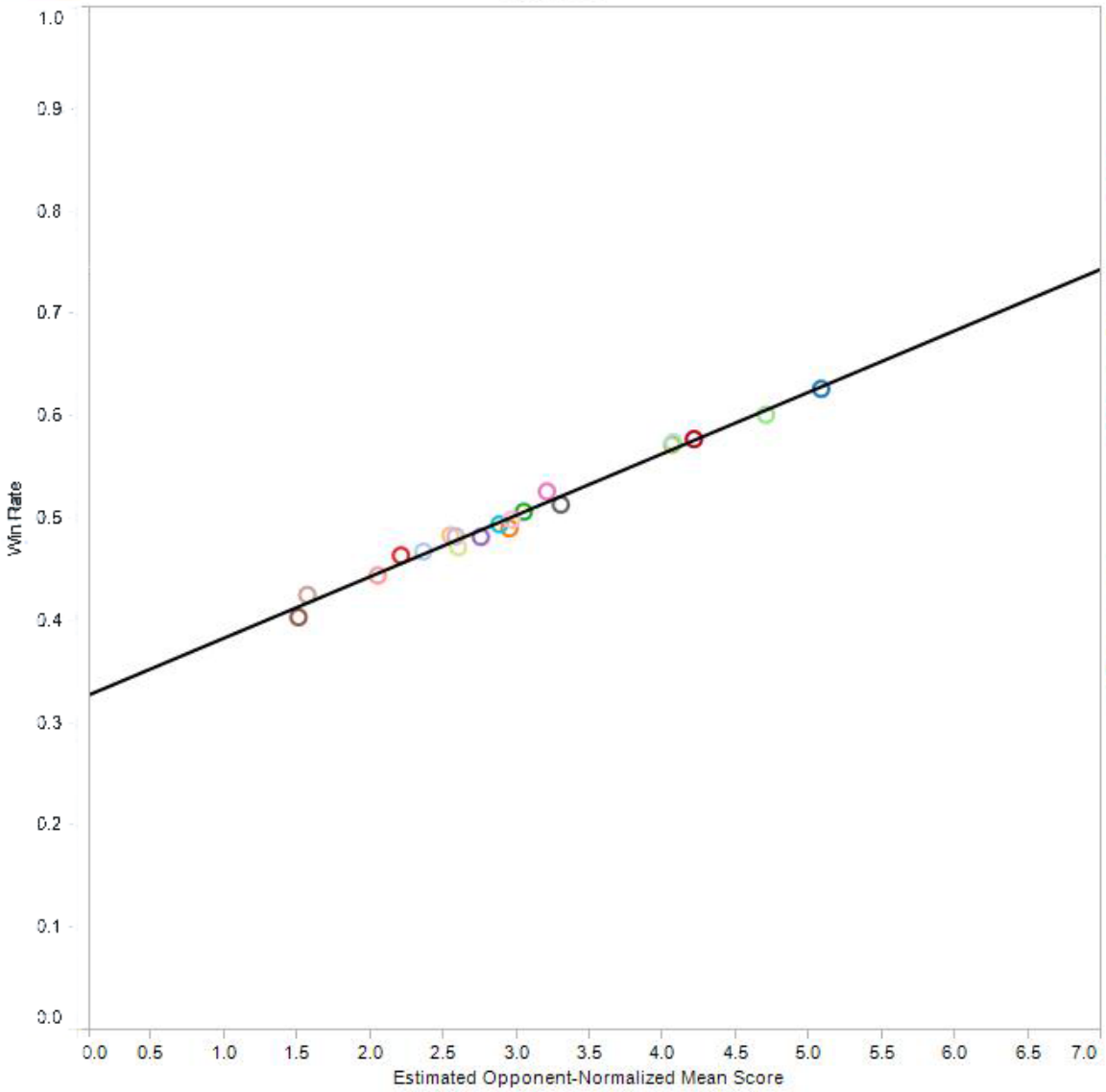

Figure 6.6: Extension of Figure 6.4, in which the x-axis is no longer shifted by actual opponent score differences, but by the estimated differences accounted for by this character's unique pattern of kills and deaths. It is potentially more interpretively useful as a 'direct' statistic of fundamental gameplay.

overall AP of all players in games with Parappa, and hence their kills. More subtly, any character who tends to get hit by regular attacks frequently (such as short-range characters) will award AP to opponents, which can then be spent on supers to kill all opponents. For this reason it seems likely that certain players' opponents should get more kills on *each other*.

We can now investigate whether such an effect does indeed exist, and if it helps explain the scores of opponents. Let a be the average AP generated by a player in a match, with a_c and $a_{o(c)}$ defined accordingly. By definition, an opponent of c generates $a - a_{o(c)}$ more AP than average. We again assume for simplicity that AP is spent equally on supers targeting each opponent. We are relaxing our assumption that c's presence does not affect kills between c's opponents, but nonetheless assume that the *only* affect comes through the increased or decreased generation of AP.

How many extra kills can we expect c's opponent to acquire in a match? The key observation is that any effect of additional AP on killing c is already factored into c's death rate. Thus we must only estimate the number of kills gained by the $\frac{n-2}{n-1}$ of the additional AP that is not spent on killing c. As we have assumed, the kill rate per AP remains static: it is k/a. Hence the additional kills of each opponent due to extra AP is $\frac{n-2}{n-1}(a-a_{o(c)})\frac{k}{a} = \frac{n-2}{n-1}(1-\frac{a_{o(c)}}{a})k$. Each extra opponent kill also results in an extra opponent death, thus bringing up each opponent score by $\frac{n-2}{n-1}(k-\frac{ka_{o(c)}}{a})(\kappa-\delta)$. Hence we may model c's new (potentially) more precise opponent-normalized score by

$$\begin{aligned} s_c + (s - s_o) &= (\kappa + \frac{\delta}{n-1})k_c - (\delta + \frac{\kappa}{n-1})d_c + \frac{1}{n-1}(\kappa-\delta)k & (6.1)\\ &\quad -\frac{n-2}{n-1}\left(1 - \frac{a_{o(c)}}{a}\right)(\kappa-\delta)k & (6.2)\\ &= (\kappa + \frac{\delta}{n-1})k_c - (\delta + \frac{\kappa}{n-1})d_c + \frac{\kappa-\delta}{(n-1)^2}\left(1 + \frac{a_{o(c)}}{a}(n-2)\right)k. & (6.3) \end{aligned}$$

Note that in moving to this refined model we are again relying on statistics about c's opponents, rather than just c himself. This can make the interpretation somewhat more complicated. But it is an optional addition for a more precise model, and it concerns only a statistics with which designers reason directly (AP generation potential) rather than a derived statistic like score.

Instantiating Equation 6.3 to *PSABR* gives $s_c + (s - s_o) = \frac{7}{3}k_c - \frac{5}{3}d_c + (\frac{1}{9} + \frac{2a_{o(c)}}{a})k$. We plot characters' win rates by this expression in Figure 6.7. The regression is defined by $p_c = 0.324 + 0.060(\frac{7}{3}k_c - \frac{5}{3}d_c + (\frac{1}{9} + \frac{2a_{o(c)}}{a})k)$, and the RMSE is 0.0017, the same value as is achieved by correcting for the true value of opponent score. In other words, the same accuracy can be achieved solely by considering the effects of player kills and deaths on

opponents scores with an adjustment for the additional AP gained by opponents. Note that the adjustments from the previous formulation are subtle, because characters' mean AP only vary by a few dozen AP. However, one can imagine a circumstance in which a character's presence affects opponents' AP drastically, and this correcting term would be all the more important.

6.5.4 Other Gameplay Factors

I have spent a great deal of space detailing the relationships between a few key variables in *PSABR*'s gameplay, and more importantly, our efforts for characterize these relationships. However, wins, score, kills, and deaths do not begin to capture the full dynamics of the game, nor does it suffice to stop there, from a practical standpoint. Just as it is not very useful to understand characters' win rate without capturing the underlying causes (kills and deaths), we must similarly dig deeper into the causes of kills and deaths themselves. This effort is essentially never-ending (it's 'turtles all the way down') for a system as complex as *PSABR*. This is why gameplay analysis must be an iterative process. We are always making dangerous assumptions about the relationships between factors; the more time we spend the more misunderstandings we can rule out, and the lower the chances of a catastrophic misjudgment.

For instance, a character's rate of kills can be broken up rather effectively into AP generation (the rate at which they perform successful regular attacks and build their super meter) and super efficiency (the average number of kills achieved per AP spent on supers). Naively it seems that this relationship is trivial to characterize: $k_c = a_c \cdot e_c$, where k_c, a_c, and e_c are character c's mean kills-per-game, AP-generated-per-game, and kills-per-AP-spent. There are a surprising number of factors that can disrupt this formula however, and it is a useful exercise simply to enumerate them, showing the level of due diligence that must be performed before an assumed model is trusted.

- The AP generated and super efficiency of *individual games* have positively and negatively correlated factors. For instance, high AP generation in a game is evidence of favorable conditions for the player (be it weak opponents or an advantaged stage), which would tend to increase super efficiency as well. Yet a player with high AP generation in a game must be spending much of her time performing regular attacks, thus giving her fewer opportunities to perform super attacks and get kills. What's more, killed opponents are unable to be attacked for several seconds, thus potentially bringing down AP generation. Surely, at least for a fixed player, one of these factors will win out so that AP generation and super efficiency are indeed positively *or* nega-

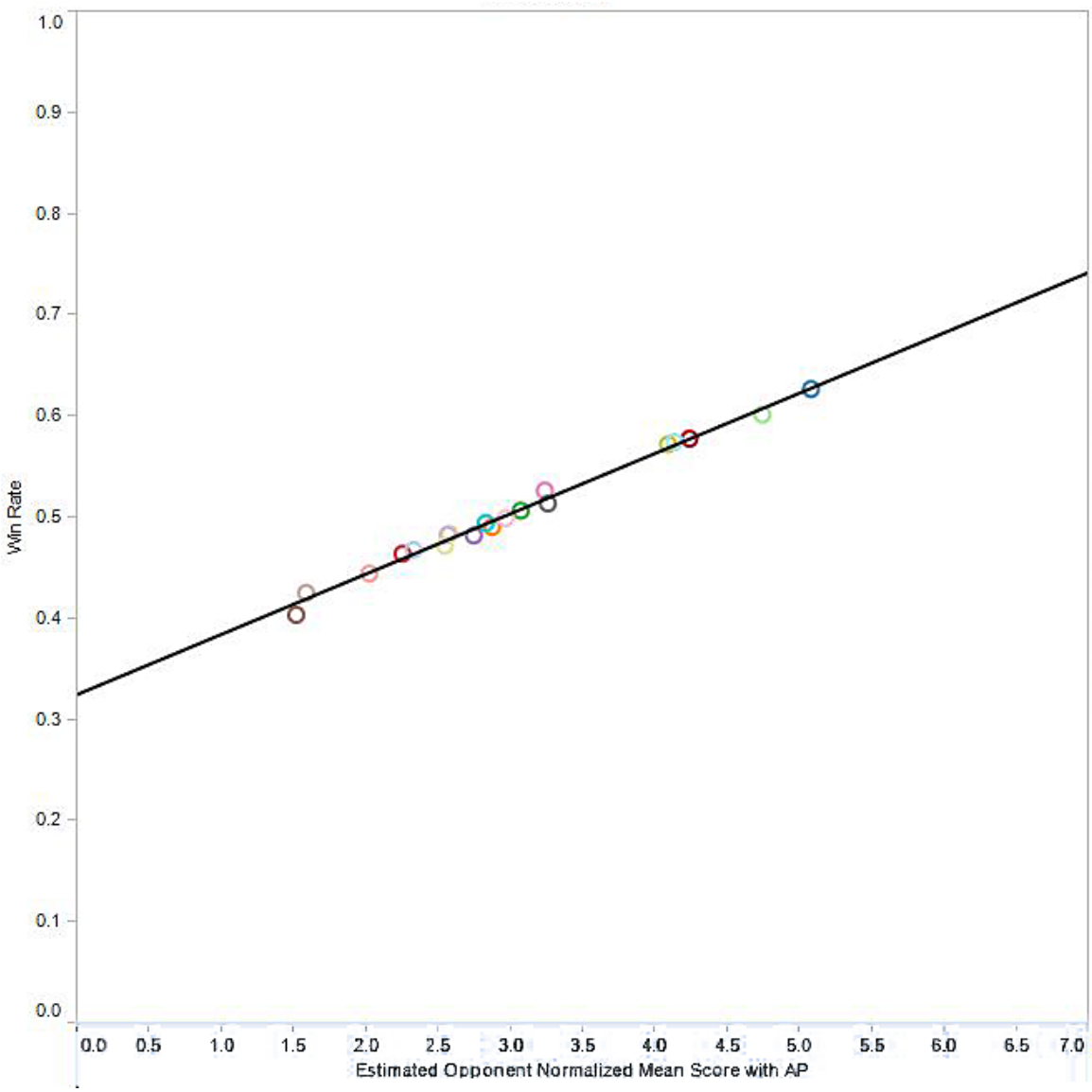

Figure 6.7: Slightly improved extension of Figure 6.6, with the addition of one more feature into the estimation. A character's effect on opponents' AP generation is factored into the predicted opponent score, by estimating opponents extra kills on each other through supers paid for with this AP.

tively correlated overall. In this case $a_c \cdot e_c$ must be an underestimate or overestimate of k_c. (Because, e.g., $(a_c - \epsilon) \cdot (s_c - \beta) + (a_c + \epsilon) \cdot (e_c + \beta) > a_c \cdot s_c + a_c \cdot e_c$.)

- More generally, the means of these three statistics capture a wide variety of conditions, players. Hence we will see correlations between the individual AP generations and super efficiencies across players as well, causing over and under estimates in the same way.

- Players have a maximum cap on the AP they can hold at any one time. This means that some AP earned is effectively wasted, and does not contribute to their overall kills.

- Players also waste AP by failing to spend it before the end of the round. This nearly always happens to some extent because of the large discrete costs of supers. Indeed, different characters experience this phenomenon to different extents, and so $a_c \cdot e_c$ acts as an overestimate of k_c to varying extents.

- Players can have AP removed from them by certain items and hazards, yet this is not necessarily captured in the record of their total AP *gained*.

Just as we needed a roughly approximate model for the relationship between wins, kills, and deaths, it is necessary to build an understanding of the relationships between kills, AP generation, and super efficiency. We can do so roughly by directly measuring the extent of each of the phenomena listed above; with more time we can incorporate compensatory mechanism into our primary metrics to address these phenomena.

The list goes on. AP generation is a function of the attacks selected, the effectiveness of those attacks, and the behavior of opponents. Super efficiency is a function of the kill rate of individual supers, the AP costs of supers, and player selection of supers. We have explored all these facets and more. We have even begun delving into the conditions of success for individual uses of attacks and supers, and the way these statistics can reach up to effect broader statistics in the hierarchy. Balancing a live game means this process never ends, and means delving ever deeper.

6.5.5 Using Gameplay Models for Data Exploration

Let us now step back for a moment and consider the motivation for this work. The aim is not as straightforward as prediction. If our ultimate goal were to take a character whose primary statistics are known and predict their win rate, we would most likely be better off

throwing a collection of potentially relevant features into a standard machine learning algorithm, and using whatever regression the system trains for. However, in this instance the priority is to build *understanding* of the relationships between variables, so that in designing or redesigning a character, designers can understand the effects of their decisions, approximately or otherwise. There is a parallel here to the way scientists build understanding of fundamental scientific laws. Such laws are generally more useful to the engineering process than would be an oracle that can opaquely produce predictions. The regressions generated by a more general machine learning algorithm might produce an easily interpretable function, and can even be influenced to do so. But such a process is not certain to succeed, and is often overkill when what we really want to do is confirm or deny our existing theories about relationships between certain variables.

Similarly, these 'rules' guide our efforts to examine the data as we receive it. To understand a character's strength, we first look at their win rate (often under varying conditions, as discussed in the subsequent chapter). We then must endeavor to understand *why* a character wins with that frequency, so that we can understand if the win rate is likely to change, if other choices might change it, and how we can manipulate it. Rather than explore countless visualizations and formulate theories as to the effects of certain data points on the win rate, we can immediately direct ourselves to the key statistics of interest: kill rate, death rate, and opponent AP generation. If any of these statistics is out-of-the-ordinary, we know that it can act as a causal factor for that character's win rate, and more importantly, we know *to what extent* it can do so. Before these relationships were derived, we knew in principle that a high death rate could cause a character to lose frequently, but we had essentially no idea how that factor balanced out with others factors.

The contextual value of these models must be emphasized. They are only a guideline, and will not function in all circumstances. Indeed, no linear model can ever produce a valid probability function. If we think of the relationship between performance statistics and win rate as sigmoidal (which is a fair first assumption) then the accuracy of these models shows that the 'near-linear' portion of this sigmoidal function is quite long. Nonetheless, it would be helpful to attempt to derive a more realistic model for win rate - one that produces an actual probability distribution and is based on an explicit process simulating wins and losses. The next subsection attempts to do so, again focusing on the creation of an intuitive model 'by hand', rather than inferring one through a generalized machine learning process.

6.6 Case Study Part 3: Extending Models for Prediction

I now set out to build a somewhat accurate probabilistic model of wins that depends solely on the means of a few basic statistics. Such a model allows for the prediction of win rates

of new characters and variants to existing characters, as well as selecting among basic rules like the scoring function. I show how both such applications would work in this section, and yet it is somewhat surprising that it should be possible. Much of the message of this thesis is that the relationships between gameplay mechanics can be highly chaotic, non-linear, and dynamic. They depend on player decisions and the precise state of the game at any time. Such models are only possible because they employ our domain knowledge about the rules of the game and the basic statistical properties that seem to emerge. We can think of this as a sort of advanced feature selection process, where, rather than place potentially meaningful features into a generic machine learning algorithm, we hypothesize a simple, fixed relationship between variables, and learn only the most basic parameters from data.

Concretely, I enter into this project working from the assumption (anecdotally confirmed from data) that a player's kills and deaths can be approximately modeled by a Poisson process. This model hides under the rug *many* essential dependencies across variables and time. Nonetheless, the poisson model for scoring has repeatedly been confirmed to be useful in analyzing sports, e.g. in [33]. Here I extend this model to far richer setting in which multiple players compete in a free-for-all, and scores are based on multiple weighted values. The model again turns out to be highly predictive of real data, and remains simple enough to easily explain, and can facilitate win probability calculations using Monte Carlo sampling.

6.6.1 A Probabilistic Model of Player Wins

My goal in this subsection is to define a model that probabilistically assigns wins to players according to a few parameters, while being somewhat in-keeping with the simplified models of the previous section. I do so by modeling the kills between each pair of players as a random process, and calculating the probability of the player obtaining a higher score than a given opponent.

Concretely, we treat the kills between each player as coming from a Poisson process. This assumption is mathematically convenient, but also makes intuitive sense for a multiplayer action game. Each use of a super attack can be thought of as a 'kill attempt', and each attempt's success is independent from the last. In reality there are many factors tying together the timing and effectiveness of subsequent kill attempts, but the premise is that such a simplification should be sufficient.

A Poisson distribution is defined by a single parameter λ, its mean and variance. Let $K_{i,j}$ be a random variable specifying the number of kills player i inflicts on player j. Let player 1 be the player whose win rate we wish to measure, playing character c. We make the simplifying assumption that each opponent is independent but identical to one another, and 'train' a Poisson distribution for them solely based on the mean. In other words, we

define $K_{i,j}$ $Pois(\frac{k}{n-1})\forall i,j \neq 1$. We let player 1's statistics control the Poisson distributions determining kills by or on him, so that $K_{1,i}$ $Pois(\frac{k_c}{n-1})$, and $K_{i,1}$ $Pois(\frac{d_c}{n-1})$.

Under this admittedly simplified model, we can approximate the probability that Player 1 places higher than a given opponent Player m. Doing so requires only that Player 1 achieves a higher score than Player m. However, the conditions under which this occurs in a real game (of *PSABR* at least) would be complex to simulate. Until there is a unique highest-scoring player, games go into successive rounds of overtime, with increased AP generation. Once that process ends, there is a unique winner, but other players may be tied for 2nd or 3rd place. For simplicity we do away with overtime in our model, and allow for players to tie arbitrarily. Once ties are possible, our real interest is in estimating the expected value of a given player's outcome. Equivalently, we treat each tie as a win with probability 0.5. Hence we need only halve the contribution to the final probability of the event that two scores are identical.

Define the random variables for a player's score, kills, and deaths by $S_i = \kappa K_i - \delta D_i$. Of course, $K_i = \sum_{j\neq i} K_{i,j}$ and $D_i = \sum_{j\neq i} K_{j,i}$. Then under this model the probability that player 1 beats player m is $p_1 = \Pr[S_1 > S_m] + \frac{\Pr[S_1=S_m]}{2}$.

I first address the first term above.

$$
\begin{aligned}
\Pr[S_1 > S_m] &= \Pr[\kappa K_1 - \delta D_1 > \kappa K_m - \delta D_m] && (6.4)\\
&= \kappa \sum_{j\neq 1} K_{1,j} - \delta \sum_{j\neq 1} K_{j,1} - \kappa \sum_{j\neq m} K_{m,j} + \delta \sum_{j\neq m} K_{j,m} && (6.5)\\
&= \kappa \sum_{j\neq 1,m} K_{1,j} - \delta \sum_{j\neq 1,m} K_{j,1} - \kappa \sum_{j\neq 1,m} K_{m,j} + \delta \sum_{j\neq 1,m} K_{j,m} && (6.6)\\
&+ \kappa K_{1,m} - \delta K_{m,1} - \kappa K_{m,1} + \delta K_{1,m} && (6.7)\\
&= \kappa \sum_{i=1}^{n-2} Pois(\frac{k_c}{n-1}) - \delta \sum_{i=1}^{n-2} Pois(\frac{d_c}{n-1}) && (6.8)\\
&- \kappa \sum_{i=1}^{n-2} Pois(\frac{k}{n-1}) + \delta \sum_{i=1}^{n-2} Pois(\frac{k}{n-1}) && (6.9)\\
&+ (\kappa+\delta) Pois(\frac{k_c}{n-1}) - (\kappa+\delta) Pois(\frac{d_c}{n-1}) && (6.10)\\
&= \kappa Pois\left(\frac{n-2}{n-1}\cdot k_c\right) - \delta Pois\left(\frac{n-2}{n-1}\cdot d_c\right) && (6.11)\\
&- \kappa Pois\left(\frac{n-2}{n-1}\cdot k\right) + \delta Pois\left(\frac{n-2}{n-1}\cdot k\right) && (6.12)\\
&+ (\kappa+\delta) Pois\left(\frac{k_c}{n-1}\right) - (\kappa+\delta) Pois\left(\frac{d_c}{n-1}\right), && (6.13)
\end{aligned}
$$

where the last step holds because the sum of independent poisson distributions is equal to a poison distribution with mean equal to the sum of the means.

Unfortunately, unless $\delta = \kappa$, no further straightforward simplification is possible. Linear combinations of Poisson distributions are not themselves Poisson. In fact, even a single Poisson distribution multiplied by a scalar is not Poisson; this is easy to observe because the support of the distribution is no longer the natural numbers.

6.6.2 Evaluating the Poisson Model

The difference of two Poisson distributions is a well-known entity known as the *Skellam distribution*. Unfortunately, even the Skellam distribution's probability mass function involves an infinite sum (as part of the Bessel function) that must be calculated numerically. In the same fashion, calculate probabilities for Equation 6.4 requires numerically approximating six levels of infinite sum. On the one hand, we are fortunate in that this calculation does not become more complex as the number of players increases or κ and δ change. On the other hand, it is already somewhat prohibitive.

As a consequence, I choose to calculate win probabilities under the Poisson model through Monte Carlo sampling. This makes it possible to plot the win rates of hypothetical characters having any kill and death rates. This allows us to confirm the coherence of this model to the linear model in the ranges of real data, and having done so, examine potential win rates outside these ranges, with some skepticism.

The first step is to instantiate Equation 6.4 with $n = 4, \kappa = 2, \delta = 1$. This gives us win probability.

$$2Pois\left(\frac{2k_c}{3}\right) - Pois\left(\frac{2d_c}{3}\right) - 2Pois\left(\frac{2k}{3}\right) + Pois\left(\frac{2k}{3}\right) + 3Pois\left(\frac{k_c}{3}\right) - 3Pois\left(\frac{d_c}{3}\right),$$

We will first visually inspect this model in comparison to the linear model. In Figure 6.8 I plot the win rate as a function of kill rate, with one pair of lines per fixed death rate. The pairs of lines represent deaths rates 1.5, 3.0, 4.5, 6.0, and 7.5, from top to bottom. Note that the Poisson model produces a valid probability distribution for any choice of death rate, whereas for any death rate the linear model breaks down for sufficiently low or high kill rates. More interestingly, the Poisson model has very good parity with the linear model for 'moderate' values of death rate and kill rate. It is not surprising that the models are inconsistent well outside the values of approximately three kills and deaths. The linear model is only trained for kill and death rates that have been observed in practice. In the release game of *PSABR*, characters are relatively well-balanced and hence do not have vastly different kill and death rates. Nonetheless, it is useful to have some idea of how such

characters perform. This Poisson model gives us a very strong initial idea.

In actuality, it is remarkable that the Poisson model is such an accurate predictor. It is trained exclusively with the mean kill and death rates of each character, and is clearly an oversimplification in some regards. For instance, it assumes that each opponent has the overall mean kill and death rate, whereas really these opponents are first sampled from such distribution. The presence of one weak opponent and two somewhat strong opponents could in principle produce very different win rates than three average opponents. As we can see, for the observed data this is essentially not the case. Moreover, there are clear interactions between characters kills and deaths; they are not fully independent. Yet the prediction is quite good as a first approximation.

We can quantitatively check the accuracy of the model by evaluating it on the concrete data points we have: the twenty characters. The root mean squared error between the actual win rate of each character and the predicted win rate by the Poisson model gives us a reasonable measure of its accuracy: **0.026**. As we are comparing probabilities, we can interpret this as roughly a 2.5% error in predicted probabilities. This is not perfect, but is completely sufficient for shifting characters into a healthy range of values.

6.6.3 Choosing a Scoring Function using the Poisson Model

A generalizing model of win rate opens up interesting possibilities beyond evaluating individual and potential characters. In particular, it allows us to explore the potential consequences of scoring functions themselves. This idea is a natural extension of the *PSABR* designers' choice to use a non-traditional scoring function. In the majority of games in which one player scores 'off' another, players' scores go up through their own actions exclusively. In some games, points are traded, where a kill by player A to player B adds to A's score and detracts from B's, thus encouraging defense. *PSABR* takes the unusual step of choosing a non-uniform 'trading' score function. A kill is worth two points, whereas a death is worse negative-one point, thus encouraging defense to a moderate extent.

This choice reveals an entire space of scoring functions for multiplayer games, which value offense and defense to varying extents, thus helping shape the intended play experience. Under normal circumstances, the choice of such a score function would be somewhat difficult. *PSABR*'s score function was chosen using intuition and playtesting; to a large extent its choice was also driven by its simplicity and ease of player understanding. Nonetheless, one could image that another scoring function might be more beneficial to the gameplay while nearly as simple.

In this passage I will not seek to rule on the correct value of offensive or defensive encouragement. Rather I will show how the Poisson model allows us to visualize this

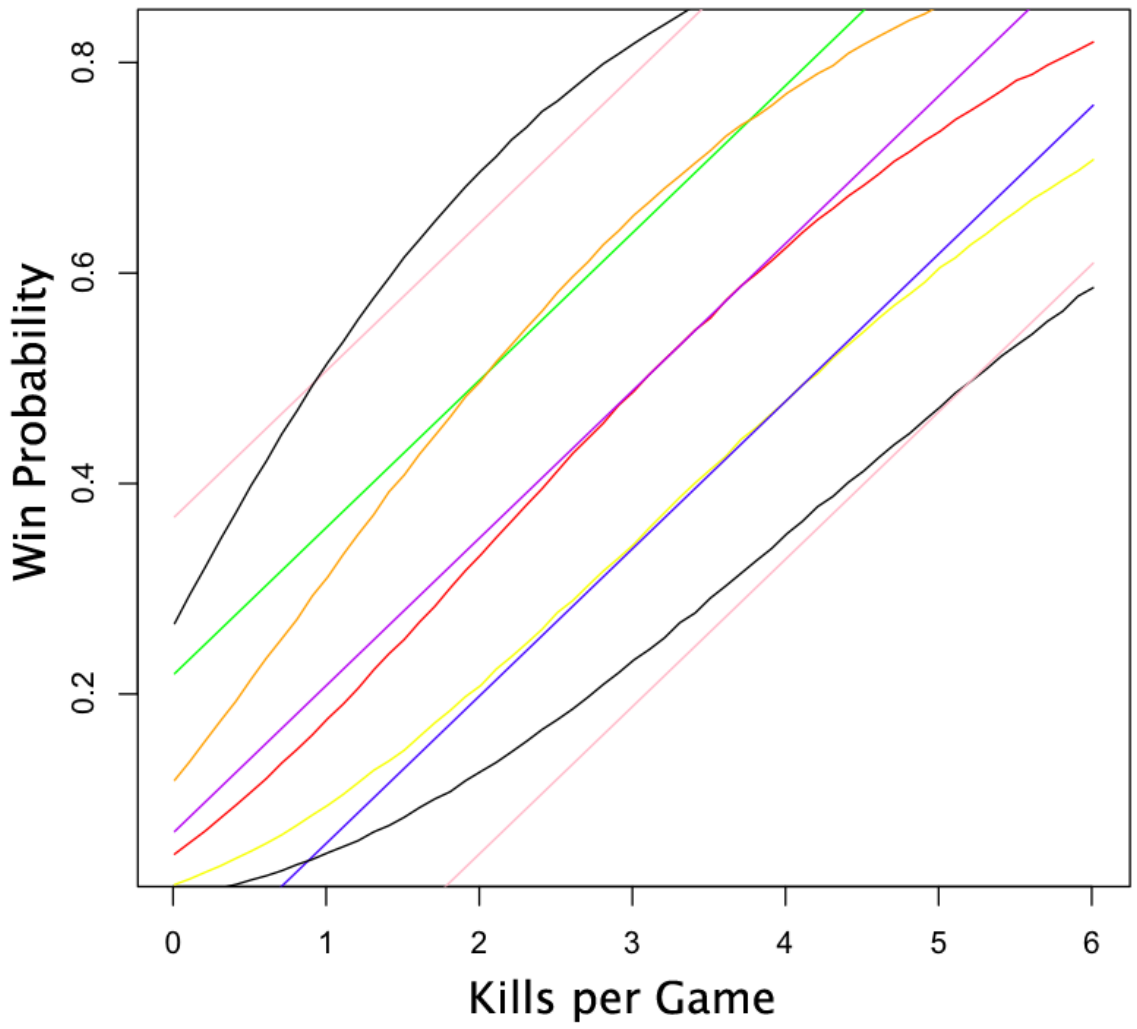

Figure 6.8: Comparison of linear model of win rate to Poisson model. Each pair of lines represents the prediction value of win rate as a function of kill rate, for some fixed value of the death rate. The five pairs of line represents death rates (from top to bottom) 1.5, 3.0, 4.5, 6.0, and 7.5 mean deaths per game.

tradeoff for any score function, and thus allow designers to pick the function that 'feels' best.

The first observation to make is that in choosing κ and δ, we need not vary δ. Scoring functions are scale-independent: from a gameplay perspective a scoring function of 3 times the number of kills minus 2 times the number of deaths is equivalent to $\frac{3}{2}$ times the number of kills minus the number of deaths. As such we may assume that $\delta = 1$ and define a scoring function by κ. In a final implementation a fractional scalar might not be desirable, but avoiding this simpler requires scaling both numbers by the least common multiple of their denominators, to make them integers.

Visualizing the effect of κ, kill rate, and death rate on win probability is a bit complex, as we have to deal with a three-dimensional function. Fortunately, by thinking about the application we can simplify the problem substantially. First, it is likely that designers will only be interested in a small number of candidate values for κ, which are easy to comprehend, so it should be sufficient to create a separate visualization for each κ. Second, we can contemplate the primary feature of interest of a scoring function. I claim that this is the range of possible kill and death rates at which a player can be moderately successful. For simplicity I define moderately successful as a win rate between 45% and 55%. In other words, for a given κ I am interested in which kill and death rates we will see in a pool of evenly matched players (or characters, for that matter). Third, I will assume that the average death rate remains fixed at 3.315 we have observed in current play. This assumption may not be exactly correct, because a change in scoring function can encourage more or less defensive behavior overall, thus changing the average kill rate by a small amount.

In Figures 6.9, 6.10, and 6.11 depict the range of moderately successful kill and death rates for $\kappa = 2$, 5, and 0.5 respectively. The x-axis of each graph specifies the death rate, and the y-axis specifies the kill rate. The space between the two lines in each graph is the valid set of death, kill pairs to achieve between a 45% and 55% win rate.

The first observation of note is that the relationship between kills and deaths for a fixed win rate and κ is indeed near-linear, despite the non-linearities of the Poisson model. In particular, κ controls for the slope of the relationship between kills and deaths, and to a lesser extent the width of the band of valid kill and death counts. For instance, $\kappa = 0.5$ specifies a looser relationship between kills and deaths when the number of deaths is high. These relationships would be difficult to reason about intuitively, as there are multiple factors at play. A high value of κ implies a stronger dependence on kills for determining score, and yet there is always some contribution of deaths to determining the winner. If δ is non-zero, deaths still determine the winner among two players who achieved no kills. Even if δ is zero, death rate still affects score, as it controls opponents' ability to kill a player and

raise their own skill.

These graphs allow us to derive the true relationship between kills and deaths for a given scoring function. The approximately similar width of the bands means that understanding the slope of the kill, death relationship is the most useful determining factor. For instance, $\kappa = 0.5$ has line roughly $k = 1.333d - 1.73$. $\kappa = 2$ has line roughly $k = 0.65d + 0.7$. And $\kappa = 5$ has line roughly $k = 0.425d + 1.4$. As a designer, how would we go about determining which of these lines describes a desirable gameplay scenario?

The most useful way to think about such a scoring function is that it tells us how many deaths a typical player should be willing to 'trade' to attain a kill. For example, with $\kappa = 0.5$ a player can maintain an average win rate in the limit as long as she gets 4/3 kills for each death she gets. (In reality, the constant factor -1.73 means that she can sustain a fixed 1.73 deaths on top of that.) In contrast, $\kappa = 5$ means that a player can trade $1/0.425 = 2.353$ deaths for each kill she attains, provided she attains 1.4 additional kills. These two settings reflect very different scenarios about how aggressively a player can afford to play. If a death is 'worth' $-4/3$ kills as when $\kappa = 0.5$, players are highly disincetivized from taking risks, and will typically run away from other players unless they have a substantial advantage, or can risk their own death for multiple kills. When $\kappa = 5$, a player can safely be killed almost 2.5 times for each time she successfully kills an opponent. Such play may be so aggressive as to discourage caution, potentially making the game less strategically interesting.

We can make this line of argumentation slightly clearer by a simplified model of the relationship between kills and deaths. Imagine that every so often a player has a chance to 'attempt a kill', at the potential risk of dying. At what risk level will a player consistently play aggressively? Let us say that a given character's attempted kill succeeds with probability p, and results in a death with probability $1 - p$. Then if $\kappa = 0.5$, a player should be willing to consistently make such attempts when $\frac{p}{1-p} > 4/3$, i.e. $p > 0.571$. If $\kappa = 5$, a player will make these attempts when $\frac{p}{1-p} > 0.425$, i.e. $p > 0.298$. These numbers provide a clear intuitive (if ad hoc) measure for the level of risk a given scoring function supports. Admittedly, this dependence between kills and deaths is not factored into the Poisson model, and yet the Poisson model has been found to be predictive of real gameplay. It would be worth considering a model in which kills correlate negatively with deaths, and determine whether it might also fit the real data but extend better to more extreme unobserved settings.

We have finally characterized the effect of a scoring function in a sufficiently intuitive form to pick a value: what confidence of killing should cause a player to be willing to risk death? This is, of course, a pure design decision, one driven by a designer's vision for a game. Let us posit that the designer desires a 1/3 risk value - that is, that players should be

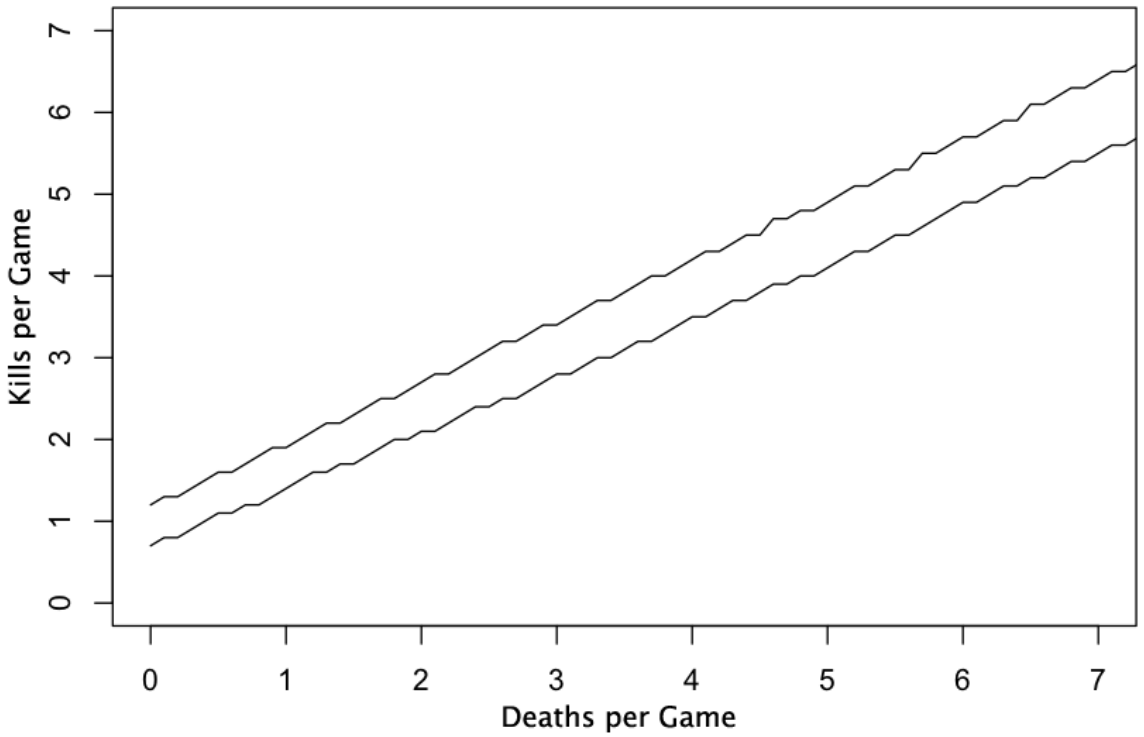

Figure 6.9: Range of kill and death rates at which a player can achieve between a 45% and 55% win rate, for $\kappa = 2, \delta = 1$. The x-axis represents the death rate, and the y-axis represents the kill rate. The space between the two lines represents the valid range of kill and death rates.

willing to attempt a kill even when there is only a 2/5 chance of success, and a 3/5 chance of death. This is a 'moderate' value for which players will in general play quite aggressively, but will back off when severely disadvantaged. What risk value does $\kappa = 2$ (the actual scoring function used in *PSABR*achieve? It requires $\frac{p}{1-p} > 0.65$, i.e. $p > 0.394$. This is nearly exactly the 2/5 risk value I proposed (prior to making this calculation). It seems that, at least in this case, designer intuition and observation combined with a desire for a simple system worked effectively to create a reasonable balance of risk. Admittedly, this calculation does not factor in the effect of the constant factor. Players must get 0.7 additional kills, so a play must occasionally take riskier bets. However, the 2/5 goal is simply a rough heuristic for success. What's more, it might easily be worth taking a simpler scoring function like $\kappa = 2$ (instead of a larger value) at the cost of less aggressive play than desired.

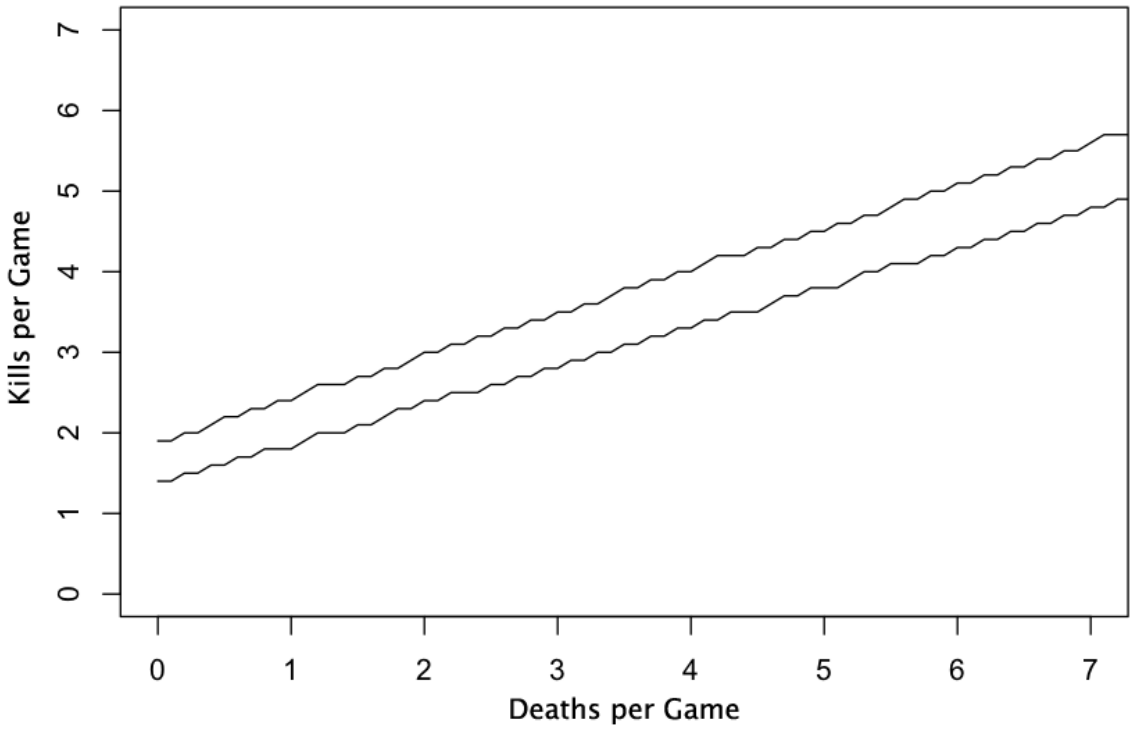

Figure 6.10: Range of kill and death rates at which a player can achieve between a 45% and 55% win rate, for $\kappa = 5, \delta = 1$. The x-axis represents the death rate, and the y-axis represents the kill rate. The space between the two lines represents the valid range of kill and death rates.

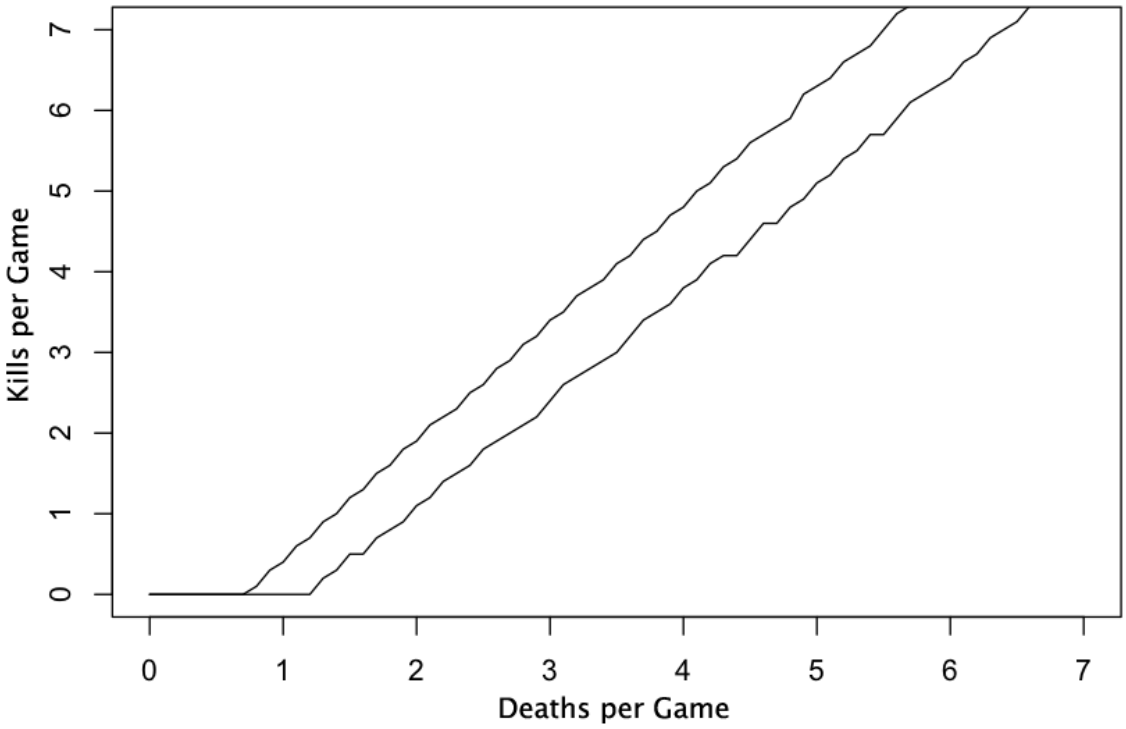

Figure 6.11: Range of kill and death rates at which a player can achieve between a 45% and 55% win rate, for $\kappa = 0.5, \delta = 1$. The x-axis represents the death rate, and the y-axis represents the kill rate. The space between the two lines represents the valid range of kill and death rates.

6.7 Wrapping Up Causal Models

In this section I have explored the utility of a coherent model of player wins, as a function of their kill and death rates. The process of building such a model must be iterative. The model I have proposed here might be sufficient to provide early insight into gameplay, and drive some small decisions. Evaluating the model's effectiveness at predicting scenarios outside of that for which it is built can then be evaluated, and the model can be extended or refined as necessary. Models such as this Poisson model should only be one tool in a multifaceted attempt to understand a game and its dynamics. When taken with a grain of salt, such a tool should surely offer a valuable perspective with which to move forward.

Chapter 7

CONTEXTUALIZING GAMEPLAY WITH PLAYER SKILL

In the introduction to this dissertation, I emphasized a single thesis statement: that game balance can be captured by the effectiveness of varying players. This is made concrete by restricted play, which explicitly restricts players and measures their win rate. In Chapter 6 I began to explore how these principles can be applied to the analysis of human data. However, the attempt was largely constrained by our inability to explicitly restrict human players (in most cases). I discussed the selection of characters as a feature that could be retroactively restricted without being influenced by other gameplay features. I also showed how it is at times possible measure other features without the use of explicit restrictions, when our domain knowledge is sufficiently strong to build a plausible causal model for connecting these features to win rate. In this chapter, I take the restrictions for human gameplay one step further, by recognizing an additional key feature of play that can be retroactively 'restricted': player skill. Like character selection, skill is essentially fixed before the beginning of a game. By assuming that players play to the greatest extent of their abilities (which is a sufficiently reasonable assumption in competitive games), skill is in fact entirely outside the control of the player. Despite changing over time, during the course of a given game it is a legitimate restriction, which we need only measure to utilize.

In prior work, other authors have attempted to understand the properties of players through clustering, typically on those players' actions or states, but the use of skill in particular has largely been overlooked. Unlike clusters, skill is a continuous measure, which allows for refined modifications of the visualizations of Chapter 6, by plotting, filtering, and sorting by skill. Moreover, skill is an (essentially) universally salient measure; it applies to nearly any game and almost always affects gameplay. Unlike automatically-discovered clusters, it has little risk of being a spurious concept. Indeed, we found that without looking at skill, many of the most basic statistics were very misleading.

On the other hand, defining and measuring skill is itself a huge challenge. Much of this chapter is dedicated to understanding skill sufficiently to provide a helpful signal. In the first section, I will discuss the classic Elo rating system, which is used in *PSABR*. The subsequent three short sections will describe some of the uses of skill in providing accurate

gameplay analysis for *PSABR*. Following those applications I will draw back the curtain and discuss the many weakness of of Elo as a skill metric. I will present a number of methods for working around these weaknesses, some of which exist (by myself and others), and some of which remain proposals by myself.

7.1 Background: Elo Ratings

Today Elo is the most ubiquitous system for rating game players. The Elo Ranking System [35] was developed in 1959 by its namesake Arpad Elo, a mathematics professor and Chairman of the United States Chess Federation (USCF). Elo's intention was to replace the existing collection of chess rating systems with a more principled technique, which would rely on a formal model of player skill and interaction. This system would serve both to rank players for status, and to match them up in certain contexts. To this extent he succeeded somewhat[1] , but it was a sufficient improvement that the method was quickly adopted by the USCF, and by 1970 the World Chess Federation (FIDE). Despite later major improvements, a slight variation of Elo remains the modern standard for rating (and consequently ranking) *Chess* players. It has also seen use in countless other competitive venues, including two-player sports such as *Tennis*, team sports such as *Football*, and competitive video games such as *PSABR* and *League of Legends*, the most popular competitive video game in play today [89]. Note that Elo's use in most sports is purely evaluative rather than actionable. Major sports have entrenched league and tournament systems of matching players and teams, which tend to be closer to a round robin format, and are not likely to be replaced by skill-based matchmaking soon. In contrast, the relative newness of video games allows them to use Elo natively, and both *PSABR* and *League of Legends* use Elo as the primary system for determining who plays whom.

7.1.1 The Elo Model

Elo maintains an integer rating for each player in an ecosystem. The rating reflects a single-dimensional model of skill which determines the outcome of games. Namely, a player's rating specifies their 'base' skill; in each game a player's 'momentary' performance is sampled from a distribution whose mean is given their skill. The winner is then determined by a simple rule: the player whose sampled performance is higher wins the game. In in its initial use, Elo assumed a Gaussian model of skill: player's performances are drawn from normal distribution with identical fixed variance, but unique means given by their Elo rating. In later years, it became common practice to adopt a logistic distribution in place of a Gaussian,

[1] I will discuss existing techniques that have largely accomplished this goal in Chapter 7.5.

as it is said to empirically model outcomes more accurately, at least for *Chess*. Specifically, if player i has rating r_i, and player j has rating r_j, then player i is said to win with probability

$$p_{ij} = \frac{e^{r_i/\psi}}{e^{r_i/\psi} + e^{r_j/\psi}},$$

where ψ is a parameter which has no impact but to determine the scale of ratings. In chess, it is hand-chosen to be $\psi = \ln(10)/400$, in order to approximately match the chess ratings of earlier systems.

This model for pairwise comparisons, known as the Bradley-Terry model [14] is older even than Elo itself. It actually reflects an extremely simple model of success or failure: each player throws a number of tokens of their color into an urn, and a single token at random to determine the winner. A player's skill is a model for how many such 'chances of success' they are given. In other words, a player i beats player j with probability $\frac{t_i}{t_i+t_j}$, where t_i and t_j specify each player's number of tokens. A player's Elo rating is a *logarithmic* measure of this expression of skill. Surely the use of the logarithm here is for mathematical and interpretive convenience. In principle skill ratings could be tracked at an exponential scale from how they are now, and it would make an intuitive sense of who will win a match easier, at the cost of making it more difficult to 'see' how scores will be updated.

7.1.2 Updating Elo Ratings

I have not yet discussed how Elo ratings are calculated and maintained, only how they are used to model the outcome of a game. Each player is assigned a default rating at the time of their first game, often 1000. Ratings are updated by thinking of them as a prediction function, and increasing or decreasing them according to the scale of error of this prediction. In a game in which only wins or losses are possible, we represent that game's true outcome by $s_{ij} \in \{0, 1\}$, where $s_{ij} = 1$ indicates a win for player i, and the difference $p_{ij} - s_{ij}$ as the gap between prediction and outcome. The policy of Elo is then simple: to update each player's rating by as a linear function of this gap. Player i's new rating is given by

$$r'_{ij} = r_{ij} - K(p_{ij} - s_{ij})$$

where K is an update parameter that determines the speed at which Elo ratings can converge to their 'true' values, but also the scale of fluctuations due to noise. Because Elo does not model confidence explicitly, K is chosen in an ad hoc way: for $\psi = \ln(10)/400$, a value of $K = 32$ is common; in Chess the value in fact is often varied according to the level of experience of players, so that new players' skill jumps to an accurate value more quickly,

where true changes in skill tend also tend to be more rapid.

7.1.3 Weaknesses of Elo

Already some of the weaknesses of Elo as a system are apparent: it does not handle ties appropriately, it lacks a principled way to choose the scale of updates, it does not accommodate for teams or games with more than two players, it cannot accommodate for metagame state outside of skill, and it does not capture the fundamentally multidimensional nature of skill. In Chapter 7.5 I will address some of these weaknesses, as well as discuss some methods for dealing with them. For the time being, Elo nonetheless serves as a very effective matchmaking system *and* evaluation metric for players. Note that in *PSABR*, we employ a simple but somewhat-flawed method for handling games with more than two players. Each game of four players is treated as a series of competitions between the $\binom{4}{2}$ pairs of opponents. This assumes that the simultaneous interactions between the four players are somewhat independent. This is not completely true, but true enough for Elo to serve as a strong estimator for wins.

7.2 Plotting by Skill

The most straightforward analytic use of player skill is to reveal an inherent pattern. In basic visualizations that integrate over skill, such a signal is lost. As such, plotting a key play statistic by its players skill at the time of play becomes a powerful, simple method for gaining insight.

In balancing *PSABR*, one of our primary concerns is to keep the strength of each character relatively in line. A key mechanisms for measuring character strength is the average number of kills per game earned by that character.[2]

The straightforward method for measuring kills per game is a simple bar graph, as seen in Figure 7.1. Labels are omitted for confidentiality, but the baseline of the graph indicates 0 kills. The initial interpretation of this graph is clear: the strongest characters have a substantial advantage. Provided that the green character does not die proportionally more often (likely), it seems that it should be weakened in balancing. However, a different picture is painted by plotting each of these averages against the Elo rating of the player controlling it, as in Figure 7.2. We find that most characters have a range of skill at which they perform

[2]This does not capture the whole picture - since the score that determines the winner is twice the number of kills minus the number of deaths. Even score is not fully predictive of winner, as a given character can alter the average score achieved by *opponents*. It is nonetheless an excellent starting point for measuring character strength, being easier to interpret than score, and more informative than win percentage, which does not capture the margin of the win.

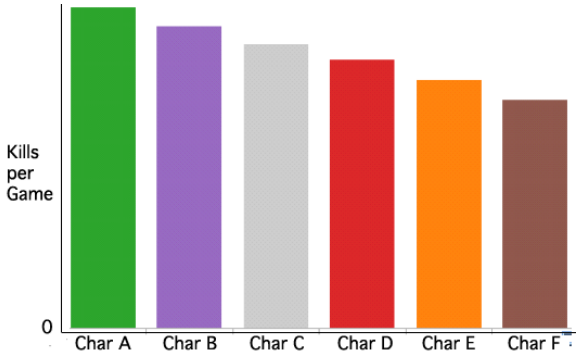

Figure 7.1: Average number of kills by each character.

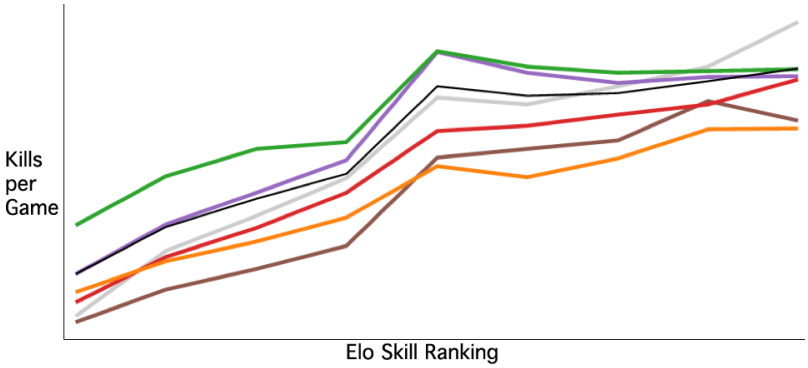

Figure 7.2: Average kills by a character for players of each Elo.

well. If anything, it is the challenging-to-play grey character who should be weakened, which performs well above the others in expert play. Players may be frustrated by a dominant character in early play, but can at least overcome it by improving; if a character dominates in expert play, a competitive player has no choice but to play it. Moreover, designers can target the techniques used by the top players and weaken those strategically.

Figure 7.1, outside the context of Figure 7.2, could have easily misled designers into weakening a perfectly acceptable character. Similar phenomena were observed across the measures that we study, making plotting-by-skill a crucial variant of almost any graph of interest. I will further detail these graphs, their motivation, and their use, in my dissertation.

7.3 Filtering by Skill

Just as plotting by skill serves to highlight phenomena that explicitly transition by skill, (of which there are many) filtering by skill facilitates more complex visualizations for a fixed skill region of interest, or excluding some skill outliers. This may be as simple as bucketing data into, for example, high skill players, then examining the usual collection of graphs in this context. However, in this section I present a slightly more subtle use of skill filtering, to ensure that the data examined has any relevance to a games 'true' balance. In essence, the problem is that poor matchmaking artificially alters skill-based measures.

Having observed the role that skill plays in kills per game, we sought to understand such relationships in general. We created graphs showing the correlation between skill and various features, including overall kills per match, shown as the orange line in Figure 7.3. We found, as some designers expected, that kills increased reliably with player skill. We interpreted this as the effect of increased facility with the game, despite other designers' expectations that defense improvements at high levels would decrease the rate of kills.

Cautiously, we drilled down to individual data points at the outliers, to learn about the sources of this data. What we learned by looking at individual games for high Elo players revealed a fundamental misunderstanding. The vast majority of the games with high Elo players featured at least one player of substantially lower skill. Of the players in our small private beta, there were rarely four players of very high skill available to play simultaneously. It seemed that high-skill players might be earning many kills not because it is a property of high-skill games, but simply due to their unfair advantage over weaker players. (In a larger population, players will typically be matched up with similarly skilled players, except at the *very* extreme points.)

To test this theory, we created two other views of the data. Figure 7.4 examines the effect of player skill difference on difference in kill count, and the blue line in Figure 7.3 looks at the average kill count when filtered to games in which all players are within a small skill window. Both graphs independently work to confirm the theory. The effect of skill on kills plateaus past average skill. Our matchmaking system simply was not operating effectively enough (given the population size) to blindly use data from every game. For the most meaningful view of what games would be like post-launch, we chose to restrict ourselves from then on to games of limited 'skill gap'. Doing so was counter-intuitive, as it cut out almost 3/4 of our dataset! Yet it seems clear that sacrificing statistical significance is the right choice, rather than be misled by unrepresentative data.

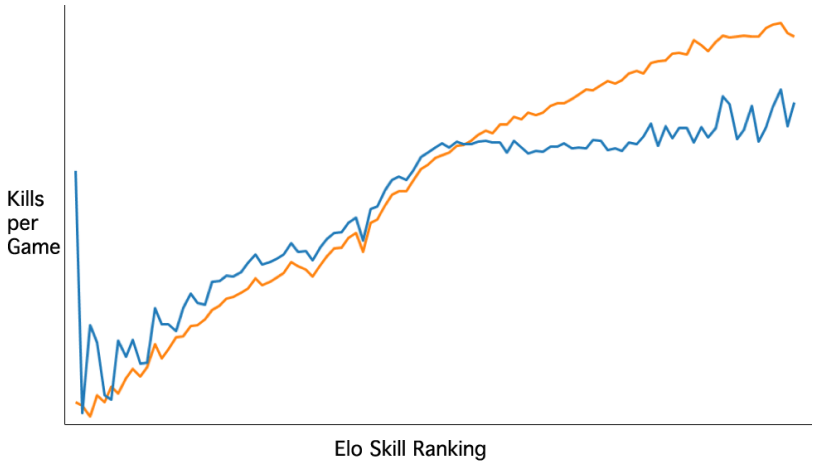

Figure 7.3: Average kills per game by players of each Elo. Orange show all games, and blue shows only games in which all players are within 100 Elo of each other.

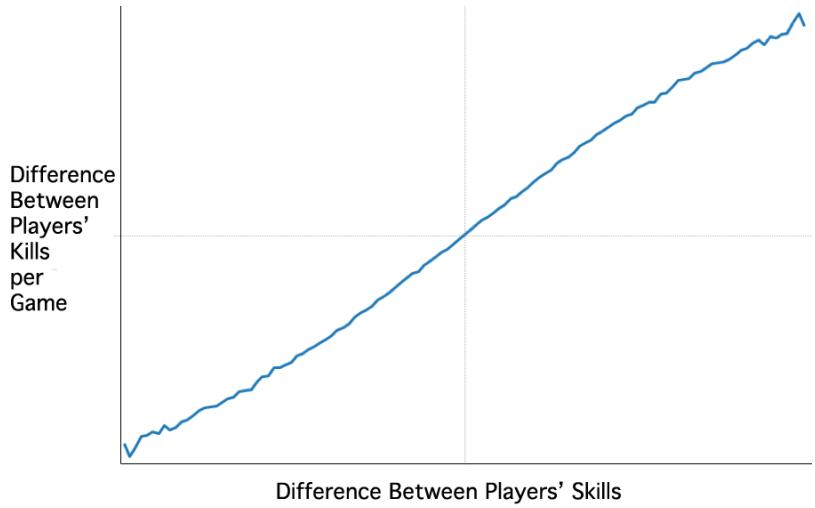

Figure 7.4: Average difference in kills between two players of varying Elo differences. Note that the graph is necessarily symmetrical.

7.4 Sorting by Skill

A further use of skill, which in some sense interpolates between plotting and filtering, is sorting. Gameplay data is sufficiently rich that sometimes aggregate statistics simply do not suffice for creating understanding of a game. It becomes necessary to view individual games, or even moments, either through video or through low-level statistics. In these circumstances, a small set of data points can be dangerous, as it is all to easy to create the illusion of knowledge of a game when the intuition gained may simply come from a few data points. Combating this illusion is one of the responsibilities of a data scientist. One way to do so is to present data in a format that emphasizes its plentitude, giving consumers an easy way to scan through it a rapid rate. Confirmation bias and other dangers still exist, but this method at least decreases the impact of noise in misinformation. What's more, such direct views of data are meant to serve as support for more rigorous analyses, by motivation further inquiry, or making sense of existing results.

One difficulty in presenting data to be 'scanned' is that the diversity of gameplay is so great. Even if a single 'feature' is under examination, its properties can swing so wildly that it can be difficult to see any trend or pattern. One way around this is to identify potential axes of change, and to sort the snippets of gameplay by this axis. Player skill excels again here, being perhaps the strongest correlative with changes in player behavior. I will discuss a concrete instance of such a problem, my solution to it, and finally the role that skill plays.

7.4.1 A Novel Visualization of Key Gameplay Snippets

As discussed in Chapter 6, a character's winning potential is based foremost on their ability to earn kills, and doing so partitions clearly into generating AP with regular attacks, and spending AP on super attacks. Regular attack data is extremely complex, but super attacks have the benefit of being infrequent and essential. Analyzing and tweaking the performance of these super attacks is the most accessible way to make progress in *PSABR* in particular. I have discussed some techniques for doing just this, but all of them suffer from abstractness. They give nearly no idea of the mechanical details of super attacks: when players are using them, why they are using them, when they are effective, and why they are effective. This is a perfect example of a circumstance in which concrete non-aggregate gameplay information becomes essential. In this instance, this concern was raised to me explicitly by Paul Edwards, *PSABR*'s lead combat designer. He asked me to build a system that would allow direct examination of the use of supers in the game, to complement the statistical data, giving causal *reasons* for the statistics we see.

The first step of such a project, and indeed the bulk of the work, was devising and

Figure 7.5: An example of Nariko's Level 1 Super Attack.

iterating a visualization for the use of an individual super. Super attacks in *PSABR* are extremely mechanically diverse. Each character has three distinct supers, of progressively greater cost and effectiveness. A typical Level 1 super tends to kills a single opponent. For example, Nariko's Level 1, shown in Figure 7.5, places a barrel of fireworks several feet in front of her, eliminating any opponents caught in its blast. A typical Level 2 tends to kill two opponents. For example, Kratos's Level 2, shown in Figure 7.6 generates a vertical tornado for about a second, killing any opponents that move through it. Finally, a typical Level 3 kills three or more opponents. For example, Big Daddy's Level 3, shown in Figure 7.7, floods the stage with water, and gives him several seconds in which to chase down and instantly kill swimming opponents.

It's difficult to imagine a visualization that could capture "what happens" during a single such super, let alone all of them. Given the variety of player trajectories, states, levels, and timing, I eventually decided it an error to attempt to visualize the aggregate use of even a single super in a single visualization, such as a heatmap. Rather I opted for a solution that allows rapid scans of individual uses of the super, as discussed. The straightforward way to do this would be to use video, creating a 'clip reel' of individual seconds of a given super's use. Although such a solution has great merit, we did not have the luxury[3], and as such were forced to invent a solution with advantages that would otherwise have been missed.

The data that *is* available to us (by design) is the full state of the game at the exact moment of each key event during a given super attack's use: the start of the super, each

[3]For bandwidth reasons, we do not collect video, nor full state or input sequences with which to reproduce video.

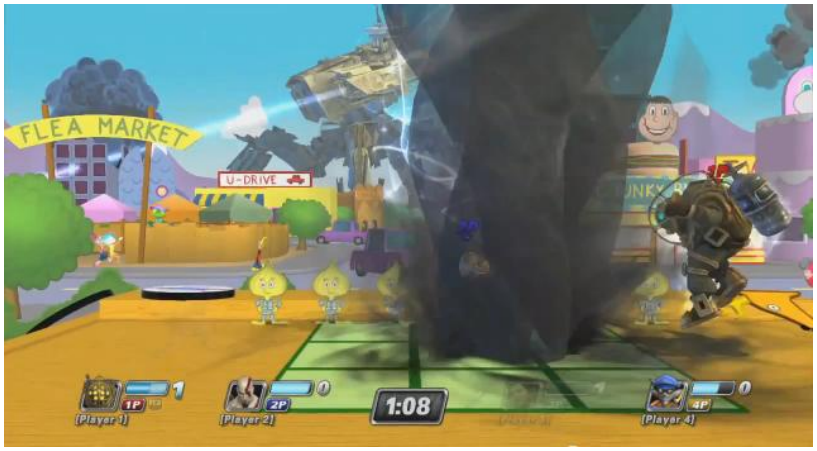

Figure 7.6: An example of Kratos's Level 2 Super Attack.

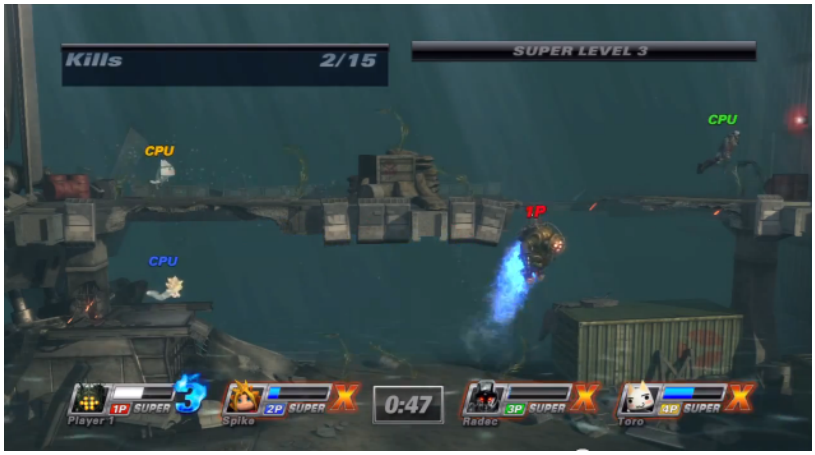

Figure 7.7: An example of Big Daddy's Level 3 Super Attack.

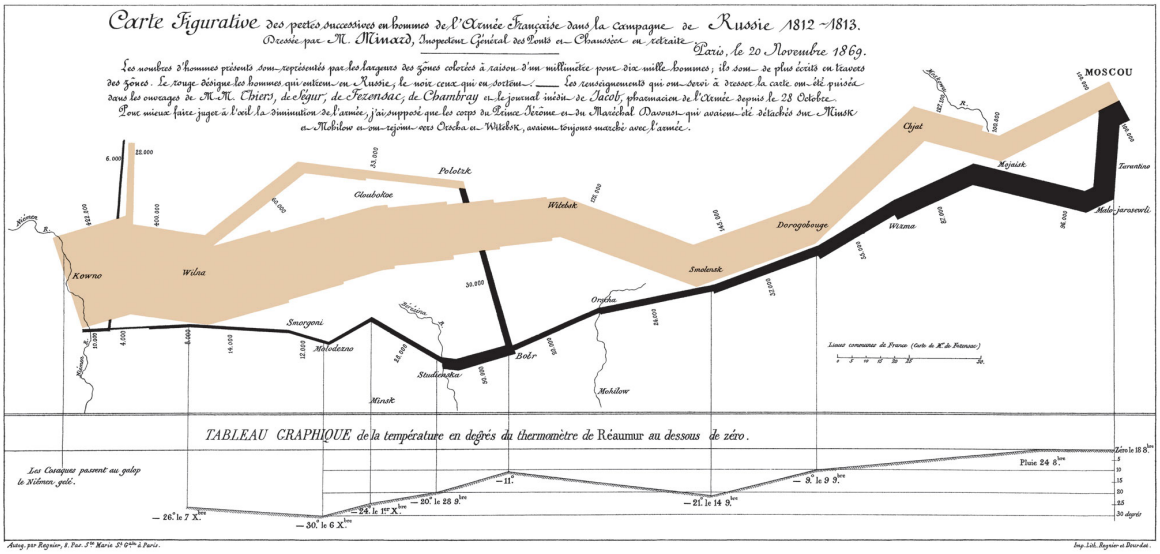

Figure 7.8: Charles Joseph Minard's *Carte figurative des pertes successives en hommes de l'Arme Franaise dans la campagne de Russie 1812-1813.* The figure depicts in a single graphic several features of Napoleon's Russian campaign of 1812. The x and y axes of the upper figure depict latitude and longitude. The thickness of the line at any point shows represents the size of Napoleon's remaining army at that point in *space*. The lower figure shows the weather at various times, corresponding to unique locations on the upper figure.

opponent's death, and the end of the super. Formulating a super in this way - as a short series of distinct subevents - motivates an alternative to video's linear presentation: collapsing time. I chose to present all of the key information of a given super's use, be it half a second or eight seconds, in a single 2-dimensional graphic. Similar visualizations were made famous by Tufte [90] (he calls them 'Narrative Graphic of Space and Time', and they are also known as 'flow maps') through his frequent praise for Charles Joseph Minard's visualization of Napoleon's March in the Russian Campaign of 1812, seen in Figure 7.8.

My own visualization was built solely with SQL and Tableau, though not without considerable effort. It is somewhat complicated to learn to read initially, but quickly becomes second nature, allowing for rapid scanning. Three examples are given in Figures 7.9, 7.10, and 7.11. Each image depicts the events surrounding a use of Kratos's Level 2 super attack, from Figure 7.6. Just as in Minard's graphic, both axes represent space, namely the x and y dimensions of the two-dimensional playspace. Each character's location at various critical times is encoded by an icon, the symbol itself specifying more detail. A series of contiguous lines connect the icons of a given player, stretching from their position at the start of the super to their position at the end of the super. Note that although in some sense these lines

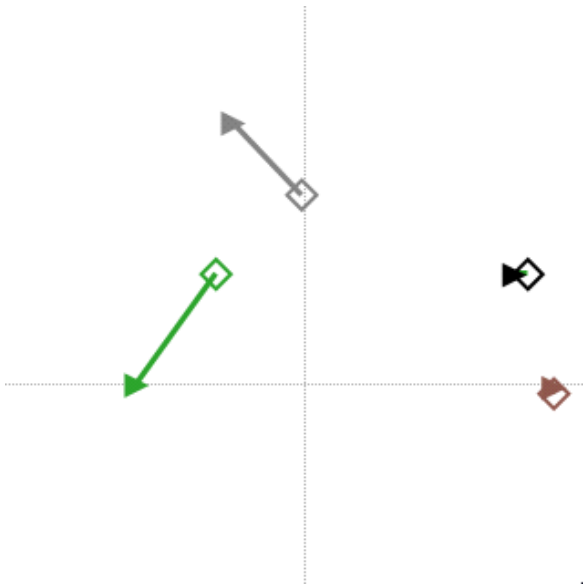

Figure 7.9: Visualization of a single usage of Kratos's Level 2 super attack, at low skill.

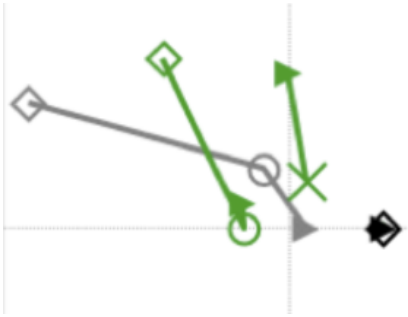

Figure 7.10: Visualization of a single usage of Kratos's Level 2 super attack, at medium skill.

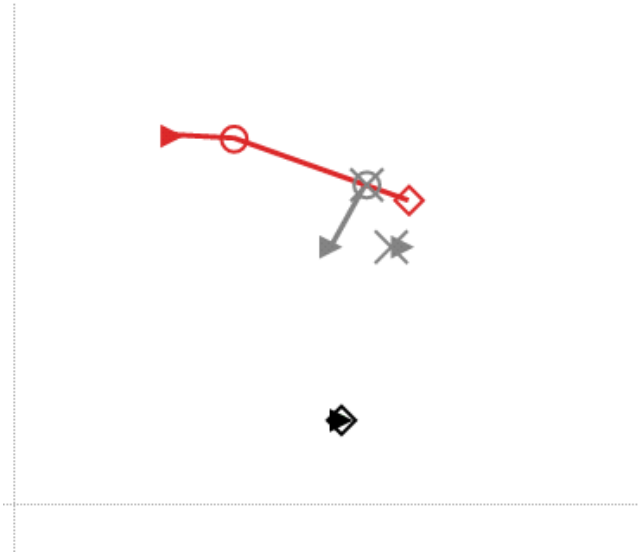

Figure 7.11: Visualization of a single usage of Kratos's Level 2 super attack, at high skill.

encode time, that time is not 'to scale'; a long line may have been traveled very quickly, or a short line slowly.

The icons themselves represent a character's position at four different states of the game:

- ▶: the start of the super.
- ×: this character's death.
- ◯: some other character's death.
- ◇: the end of the super.

Note that the order of deaths *is* implicit, because a given player's line sequence ends following their death.[4]

Note also that the color of a line sequence is determined by the character, rather than the player. In other words, it is possible to have a graphic in which all symbols and lines are the same color. Although this results in some loss of readability, it is made up for by the ability to recognize the characters playing at-a-glance, and understand how their unique

[4]Some supers allow for multiple kills of a given opponent, if the super's duration is long enough that the opponent can respawn while it is still active. In this circumstance we can display a second or third start triangle, along the same line sequence. Doing so does have the potential to ambiguate death order, which can be resolved with small numerals.

abilities might have resulted in this visualization. (All visualizations for the entire game use a common set of colors, so that an association between a given character and color can develop.)

In some sense we give up a great deal of fidelity through such a visualization; it is not known exactly how a player moves from one point in the sequence to the next, whether they performed any of their own attacks, etc. But for such a short duration, that information may be largely noise. The sparsity of this visualization emphasizes quantity over quality: the goal is to flip through dozens or hundreds of these images, to get an aggregate sense of the context in which players use a super attack, (where are they, and where are their opponents) and of when it is actually effective. The visualization targets combat designers who have a deep understanding of the workings of each character, and can for the most part understand immediately how a character may have gotten from one point to the next.

7.4.2 Sorting Gameplay Snippets by Skill

We are finally at the point where we may see the value of sorting by skill. Imagine scanning through two-hundred images like Figures 7.9, 7.10, and 7.11. It would be easy for the images to blend together, as they would change so drastically from one to the next. To begin to allay this, we sort the graphics by the Elo rating of the player executing the super, displaying that player's Elo prominently. Now, as a designer moves forward in the sequence of images, she moves from weak players, to typical players, to strong players. And in so doing, patterns *do* begin to emerge. For a super like Kratos's Level 2, we begin to see that players are using the attack when their opponents are closer to them, a good start. We then start to see more players using the attack when players are standing above them, a very good strategy for this attack. Finally, at the top level, we see players using the attack as opponents are actively jumping over them, an advanced technique that requires foresight and is more unpredictable. The *extent* to which these usage trends do appear informs designers as to whether players are using the attack as expected, and also can reveal other low-level strategies they might not have predicted. What's more, the number of locations of deaths answers the question of whether these strategies are indeed as successful as they are expected to be. Sorting by skill is essentially the only way to experientially observe these important trends.

This visualization is by no means the final word on visualizing even these particular moments of the game. An eventual next step would be to use machine learning to classify these same properties of state: in which states supers are used, and in which states supers are effective, across skills. But I argue that such an approach must be complemented, and indeed preceded, by a more direct visualization such as this. Understanding these graphics

gives insight into the all-important feature selection, for a given super. It provides concrete examples to help understand a machine learning algorithm's execution and output. And it serves as 'immediate' data, that is potentially more trustworthy and comprehensible to someone with little machine learning experience.

7.5 Alternate Formulations of Skill

As mentioned at the beginning of this chapter, Elo is but one way to formulate the skill of players. Elo is one of the simplest methods, which is both a blessing and a curse. It is easy to implement, and somewhat easy to reason about and test. But it simply does not capture much of the richness that occurs in human gameplay. No system can be a perfect simulation of player interactions, but it is sometimes worth looking at formulations that capture some orthogonal or greater aspect of skill, when relevant to our needs. I will discuss a few such methods in this section, some from the literature, and some nascent methods that I have devised. I will make an effort to explain the advantages that each such method gains over basic Elo. At a high level, the solutions I discuss all seek to address the problem of limited confidence in a player's rating. This lack of confidence can come from that player's limited experience with the game, changes in skill, or even limited experience with an individual starting condition such as character.

7.5.1 Experience Filtering

When I first presented the graph of character kills by skill I neglected to address one egregious anomaly: a hump near the center of each character's graph. During our early analysis, such humps were found in several plots of statistics against skill, and initially surprised us. Since kills are strongly correlated with wins, kills are expected to be monotonic in Elo, (being a measure of skill that is based on wins). However, if one thinks about Elo as a system rather than a true measure of skill, the explanation becomes clear. Elo (and any skill-based ranking) is a delayed signal, which takes time to converge to its true value. We plot kills according to a player's reported Elo at the time of that game, not their true Elo at that time. The hump occurs at the initial Elo value given to all players, because every player leaves a trail through this point, regardless of their actual skill at the time of their first games. Indeed, a player who joins with high or low skill 'pollutes' the measures of every other rating as she converges.

This problem is difficult to overcome, and potentially more damaging to analysis than to the matchmaking system itself. A player who plays a few poorly matched games may quickly forget the experience, but if a large fraction of players do so at the same apparent skill, it

can cause substantial statistical aberrations. This problem is considered a fundamental weakness of the Elo system, and there are three possible ways of addressing it.

TrueSkill The newest solution of choice is to use another rating system altogether. Microsoft researchers created the TrueSkill system in 2006 [45], and it has since been written about extensively and employed in major game titles inside and outside of Microsoft. TrueSkill is a generalization of Elo, one that addresses the problem of uncertainty by modeling it explicitly. It is a Bayesian method which models each player's skill not by a single value, but by a normal distribution of belief. Players are assigned a mean *and* a variance for their skill. As players win and lose games these beliefs are updated. Wins and losses shift a player's mean skill up and down; outcomes consistent with predictions decrease variance (increase confidence), whereas inconsistent outcomes increase variance.[5]

TrueSkill runs on top of a factor graph indicating the relationships between the probability distributions under examination. After each win or loss all marginal distributions are approximately updated using a message passing algorithm on this graph. In addition to supporting increased precision, TrueSkill aims to generalize Elo to games with teams of more than one player, and potentially more than two teams.

All of this technology increase the complexity of implementing TrueSkill substantially over Elo, and increases the likelihood of introducing errors. As such, *PSABR* opted to use Elo even before I arrived on the project. In principle, it would be possible to simulate TrueSkill ratings for all players, by 'replaying' all the games we have captured, since TrueSkill makes no assumptions about the matchmaking at the source of the games played. However, due to minor data loss, it would be difficult to simulate these ratings accurately in a way that could be compared to our measured Elo ratings. For this reason I opted to stick with Elo for analysis. Trueskill has not that I know of been employed for the analysis of gameplay data, which I hope to remedy in the future.

Retrospective Updates Even without explicitly modeling confidence in skill estimates, it should be possible to simply reason about the validity of these estimates. To do so we can take advantage of information from future games. A 1400-Elo player is not likely to have truly been a 1000-Elo player five games ago. Such retrospective reasoning will work in most cases, since players who are far from the starting Elo value drift away from it relatively quickly, and most players play at least a half-dozen games. The simplest conceivable way

[5]Note that TrueSkill bears a substantial resemblance to Mark Glickman's Glicko system [41], introduced as long ago as 1995. TrueSkill uses a different choice for the distribution of player wins, and models ties and multiplayer matches. However, it is based on the same fundamental idea of maintaining a Gaussian belief distribution over player skill.

to do this would be to plot games according to the *current* Elo of players, rather than their Elo at the time of play. The obvious problem with this method is that player skill legitimately improves over time, and this is hard to distinguish from perceived improvement by the system.

Instead we should hope to make an estimate about the actual skill at the time of previous games, given everything we know about current games. This is a sufficiently complex endeavor that it would likely be inadvisable without explicit modeling of skill confidence. Indeed, the authors of the TrueSkill system pursued exactly this line of reasoning in [25]. They present an extension to TrueSkill which is built to process retrospective data: 'TrueSkill through Time'. The system performs smoothing over time, propagating evidence of wins backward and forward through time, for several full iterations, until approximate convergence. The motivation for the paper was that foregoing the online approach of Elo and TrueSkill would allow for more accurate understanding of the skill rankings of past Chess masters, for comparing players throughout history. However, such a system is particularly well-suited for gameplay analysis, where we may have months of data available at once, and hope to understand the true progression of players' skill (as well as a more accurate relationship between skill and true statistics). This is an agenda very much worth pursuing.

Cleaning Elo Data As with other challenges here, my own solution to the problem of unclean data (that is, inaccurate Elo ratings) is simple: to clean it. Such simplicity is a virtue here and elsewhere due to severe time limitations. There are hundreds of questions genuinely worth asking at any given time. In most cases, two rough answers are more valuable than one reliable answer. In this case, we accomplish this by filtering. In the simplest case, we filter out the first k games by all players, where k estimates the average convergence time to actual current skill. We derive k by measuring the average skill change of a player's ith game *beyond* what would be predicted by a random walk, (whose magnitude is calculated empirically). We find a phase change after k games, in which non-random skill change per game becomes constant, potentially representing steady actual skill improvement. Such a method is ad hoc, but serves to smooth out spikes in the data, and remove the vast majority of 'lagging' skill data, since *most* games are by new players.

7.5.2 Character-Experience Filtering

The goal of TrueSkill is simple: predictive accuracy. It takes a well-defined problem (rating based on wins and losses), and gives a solution that extracts more detailed and precise knowledge of skill from data than does Elo. It assumes nearly the identical Bradley-Terry model to Elo. This model is obviously oversimplified in some ways for modeling *Chess*.

Yet for a game such as *PSABR*, it fails to capture reality on a more fundamental level, by ignoring crucial information: the 'configuration' of a game.

The Problem of Skill in Asymmetric Games Excepting the players and one bit of information (the side played by a player), each game of *Chess* has identical conditions to the last. That one bit of information is chosen randomly or alternatingly, so that its effects are averaged out and can be simply thought of as a feature of an individual game, rather than a condition of the game. Hence we may think of all games of *Chess* as identical, and treat each win or loss as an equally informative signal toward the single variable of interest skill.

In reality, a fighting game is nothing like this. The most important distinction is that players choose a character before the match. These characters affect a player's chances dramatically. Some characters are stronger than others, some pairs are particularly bad matchups, some characters are harder to play than others, and players have more experience or ability with some characters than others. Thus far we have gotten around this wrinkle with an unstated simplifying assumption. We imagine that the act of character selection is itself part of a player's skill. We essentially treat character selection as an intra-game decision, rather than a metagame decision. In this way we are able to simply evaluate a player's skill, rather than a player-character-combination's skill. If they are performing poorly due to a weak character or character with which they are inexperienced, then we say that their skill really isn't that high, *by definition*. If they were skilled, they would know to choose a different character!

This is a reasonable assumption in order to make headway in our analysis. Yet in fact this assumption obscures the true nature of the situation, more than it does to ignore, say, a player's strategic preferences in *Chess*. A choice of character in a fighting game (or for that matter army in a strategy game or class in a role-playing game, all of which I will refer to as 'character') typically exerts a tremendous influence on a player's experience, both mechanically and aesthetically. In some sense the choice to play one character over another is similar to the choice to play one *game* over another. Even in competitive scenes, where we generally assume that players 'play to win', it anecdotally seems partially untrue that players 'pick a character to win'.

There are a few factors involved here. Players may come to a game with existing aesthetic preferences, sometimes drawn to the game in the first place due to an affinity for some character. In a mashup game like *PSABR*, or a sequel, characters may even have prior experience with a character themselves. Playstyles themselves may draw in a player; some *Street Fighter* players only have interest in playing fast, aggressive characters, or slower, reactive characters; this may not even correspond with that player's optimal skillset. Some

players even go so far as to intuitionally choose supposedly-weak characters, either to prove wrong the assumptions of other players, or exhibit their own skill.

Beyond the above, there is a single most compelling reason that it might be incorrect to assume players 'choose characters to win'. As discussed in Chapter 8, a choice of character is often a long-term commitment. In any game with substantive asymmetry of starting conditions, that asymmetry implies the requirement of different skills and knowledge for different starting conditions. A player who has played the Zerg for five years in *Starcraft* has great inertia to stick with that race. Even if the game changes in a way that weakens the Zerg, or opponents learn how to exploit that player's Zerg strategies, the player will still typically stick with their choice for a long time. This is partially a matter of allegiance, but mostly simple practicality. Playing a new race would involve building a completely new set of skills to complement the shared skills, and, more psychologically problematically, 'throwing away' the sunk cost of years of work. This phenomenon becomes more literal in a game like *World of Warcraft*, where in addition to building proficiency with a given class, players must invest time into that character to mechanically power them up within the game. It is no surprise that players of a given *World of Warcraft* class are known to complain vocally when their class is weakened in a patch.

I have hopefully established that character preference, experience, and skill is a dangerous phenomenon to ignore when analyzing asymmetrical gameplay. As we have seen, my primary analysis for *PSABR* does just that. However, I have not yet mentioned that I complement this analysis with specialized analyses explicitly targeted at understanding the effects of character experience on our primary results. I will briefly outline these methods, but it should be understood that they are quite ad hoc - a basic attempt at confirming that these problems are not terribly disrupting our basic analysis. I will discuss a more principled method for addressing this problem in Chapter 7.5.3.

Modeling Character Experience My primary method for checking the effects of character selection on observed statistics is to study players' experience with their characters of choice. Such a problem manifests most blatantly when observing the high-skill play of an unpopular character. For instance, we observed that one character's popularity - call it C - was strongly negatively correlated with Elo. His play rate is shown in red in Figure 7.12. Among low Elo players C was a top character, whereas the top players played C in less than 0.5% of plays (compared to the 5% that would be seen from uniform play of the 20 characters). At the same time, C was well under-performing all other characters, particularly at high skill. His win rate is shown partitioned into skill tiers in Figure 7.13. As a somewhat advanced character, we were forced to ask whether this high-skill play might not be representative of high-skill C-play. Some amount of character experimentation is

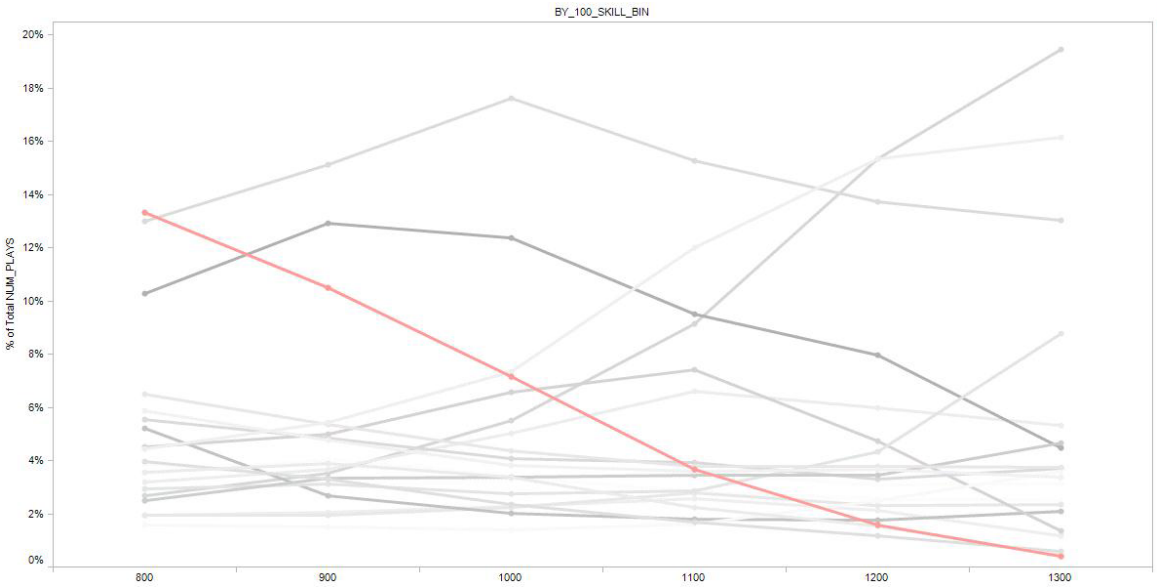

Figure 7.12: Graph of character popularity by skill. The x-axis represents a player's Elo at the time of play, and the y-axis represents the percentage of plays at that Elo represented by a given character, where each character is given one line. Character C is the colored line, in pink. We can see that weak players (mostly beginners) choose Character C for 14% of all plays. In contrast, skilled players choose Character C for less than 0.5% of plays.

expected at all skills, and it could be the case that the high skill players playing C were mostly one-offs by players without much C experience. Such a phenomenon, if true, would severely under-represent C's potential.

Once the problem is framed in this way, it is not hard to conceive of tests for this phenomenon and its possible effect. The simplest way to do so is to explicitly measure the average degree of experience with that character. Figure 7.14 shows plots the average number of past games with a character, at the time at which players of a given skill played that character. To clarify, let $c_{i,j,t}$ be the indicator variable for whether player i played player j in their t'th game, and $s_{i,k,t}$ be the indicator variable for whether Player i's Elo rating at the start of their t'th game was at least k. Then the red line for high skill (1150+ Elo) plots $\sum_i \sum_t c_{i,C,t} s_{i,1150,t} \sum_{t'<t} c_{i,C,t'}$.

Fortunately, Figure 7.14 answers our first concern somewhat confidently: high-Elo players of C have *more* experience with C on average than players of other characters. So it seems that we can tentatively rule out lack of character experience as the primary cause of this character's poor performance. Yet there is another major factor that might invalidate this conclusion. Some players are more difficult to play than others, and C in particular has

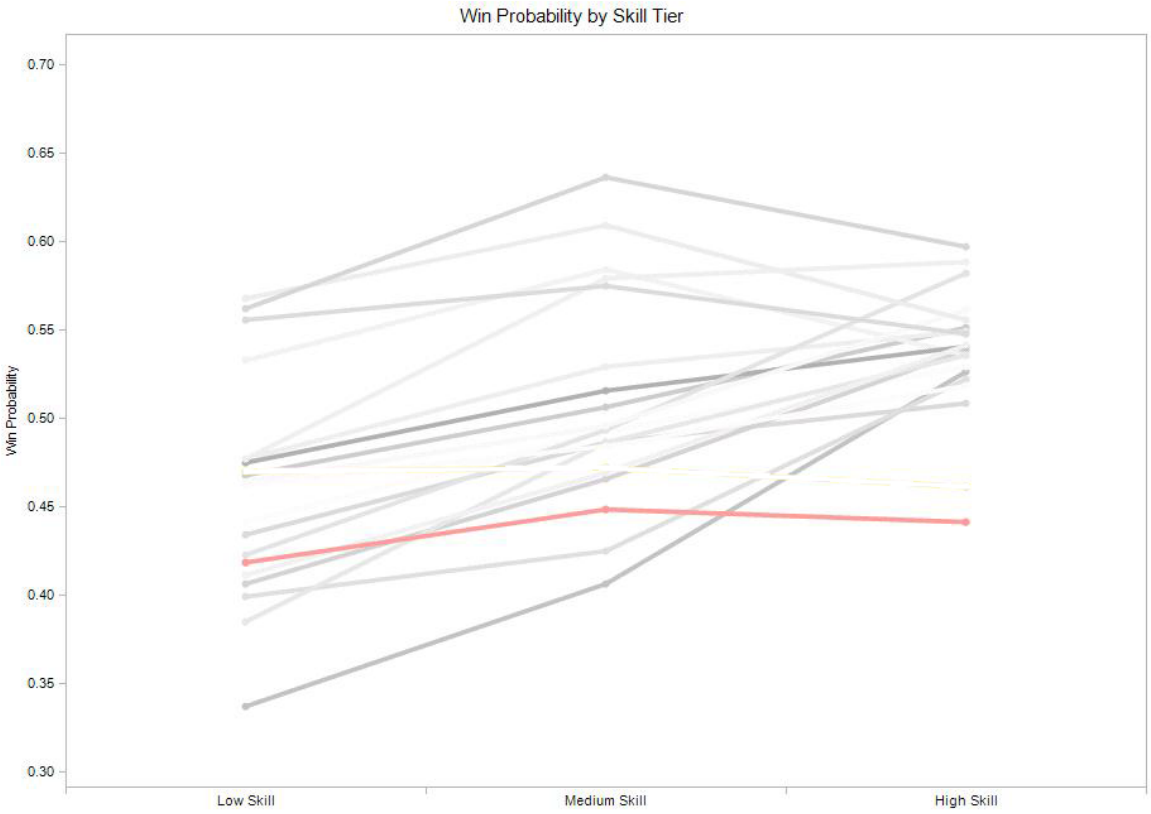

Figure 7.13: Graph of win rate by skill tier. Each line represents a character's frequency of wins (that is, the fraction of opponents whom they placed higher than), separated into three skill tiers: less than 950 Elo, 950 - 1150 Elo, and greater than 1150 Elo. Character C, shown in pink, is one of the least successful characters, and only gets weaker with skill.

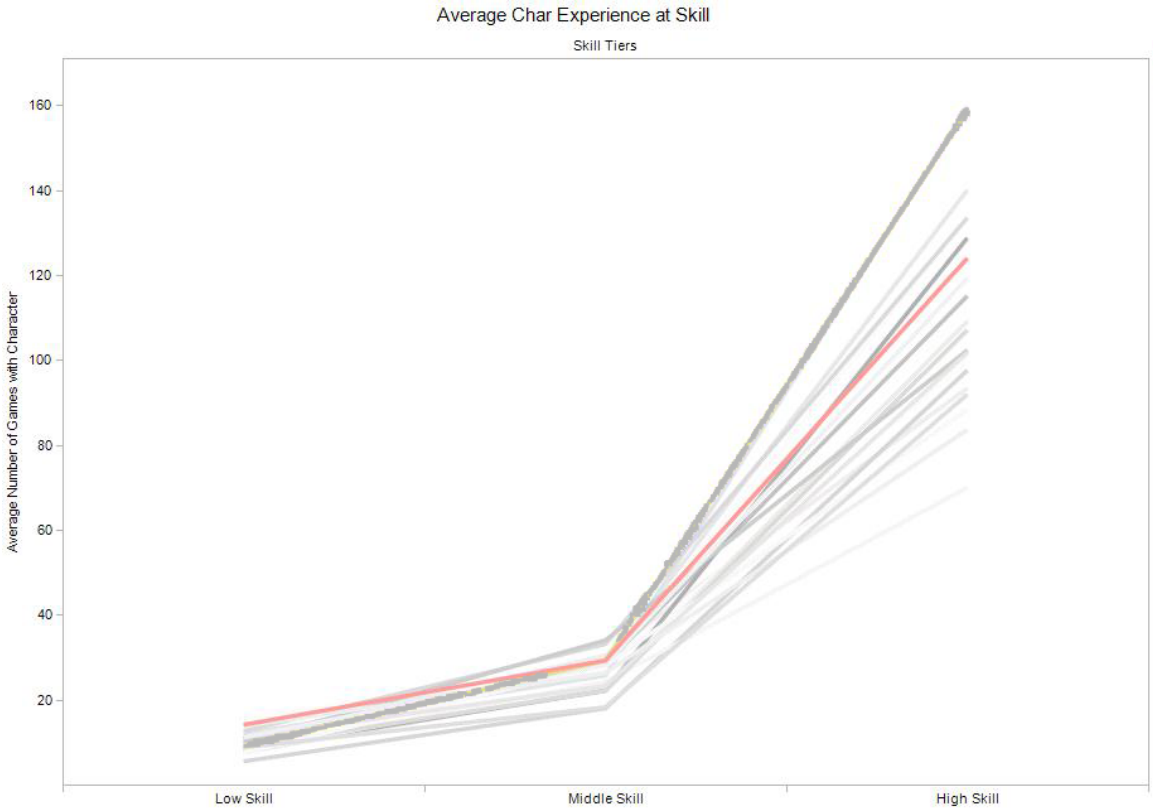

Figure 7.14: Average number of previous games played with a given character at the time of play, partitioned by skill at the time of play. Character C's numbers are shown in pink. For example, averaged over all games in which a middle skill (at the time of play) player played Character C, the typical number of previous games with Character C was just over 20. The key takeaway from this graph is that Character C players tend to have had more experience with him, and that high-skill players have had on average 120 games with him at the time of play.

advanced potential that could dramatically affect his performance. So the next question we hope to ask is how many games are required to 'become skilled' with C. This question is by no means well-defined, but we can start to get at it in the following way. We can plot the average performance of this character as a *function* of experience. In this way we are taking our primary method of plotting by skill, and shifting it to plot by a very approximate but relevant measure of skill, namely character experience. In fact, to avoid losing the gains of Elo-based analysis, we partition this data into three skill tiers, and plot all three lines simultaneously. Finally, to handle the noisiness of such a fine-grained partitioning, we plot not the average win rate at a given experience level, but the *cumulative* average win rate. This value is not informative to any true win rate at high experience, since the data is naturally biased toward low experience. What it does convey is the trend of our data: we can immediately find the experience level at which win rates converge (if one exists) by looking for a largely flat cumulative win rate line. The high skill graph for C, shown in green in Figure 7.15 does just this. Remarkably, the dependence on experience at high skill is very strong, just as we expected. Indeed, high skill play does not seem to converge to its maximum potential until players have had about 150 games with C, compared with the 120 games on average that a high-skill C player has had with C. So it seems possible that this gap might be negatively affecting C's performance, and that players who were a bit more committed to C would find greater potential.

Filtering is one way to determine just how great this potential performance improvement might be. We can produce any of the original graphs of this section, but now impose the constraint that the only games included are those from players who have already had many games with that character - say, 150. There will still be some natural bias from the tendency of some characters to be played by a given player more than others, but as that experience is not too drastic across players, this should not be too problematic. Upon producing such a graph we found that the potential improvement of experience with C is not in fact that great. Of course, this is only the improvement we have observed among C's most dedicated players; there is *always* the potential that a character can be played much more effectively than *any* existing player has yet found possible. For this instance though, we decided that the safest and most positive option was to take steps to improve C's performance.

I have taken further steps to determine the effects of character experience, both with C and other characters. But rather than continue detailing these methods, I will move on to my proposal for a more principled approach, which if effective should make all such distinct ad hoc analyses unnecessary.

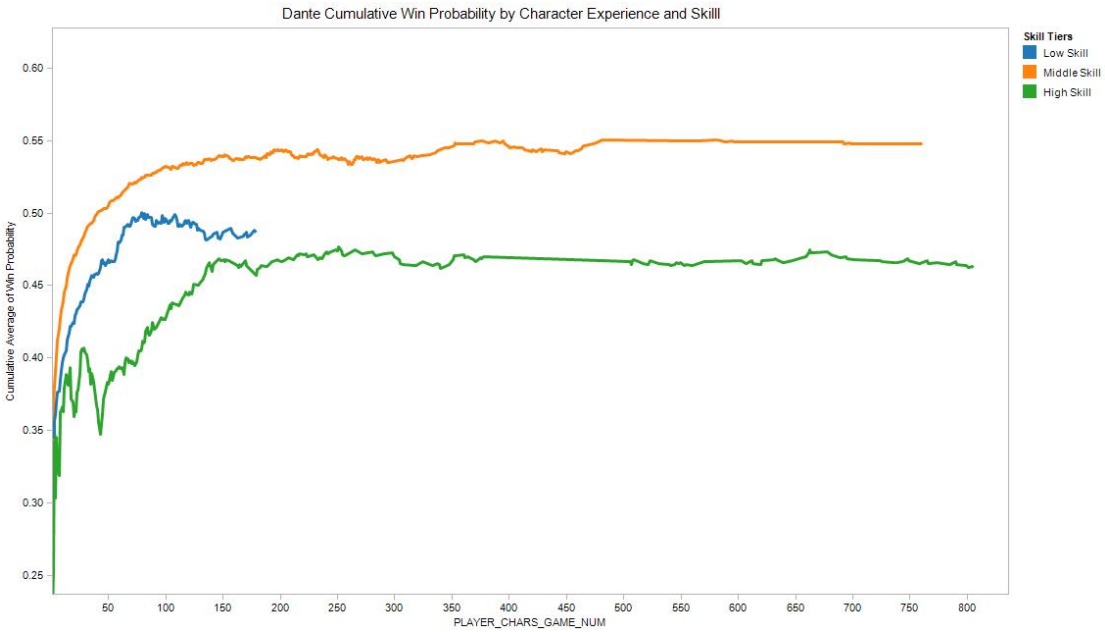

Figure 7.15: Graph of the approximate utility of previous experience with Character C. For instance, the green line represents all plays by Character C players at high skill. The x-axis divides the plays according to the number of previous games that player has had with Character C. So, for instance, the green value at x=1 is the average win rate by skilled players in their first game with Character C. As x grows, the y axis gives the cumulative average win rate of all values up until that point. We do so in order to dampen noise and make the graph readable; the primary value of this graph is to note the point of rough convergence, where the cumulative average loses any trend. Note that all this data is somewhat skewed by occasional data loss due to crashes, but this is unavoidable, and at worst will tend to shift the graphs laterally.

7.5.3 Multidimensional Elo for Character-Centric Skill Modeling

Just as TrueSkill seeks to solve the problem of inaccurate Elo ratings in a principled manner that does not discard data unnecessarily, I will propose a hypothetical principled method for reasoning about character-specific skill. The precise method will surely require iteration once tested experimentally. Nonetheless, the spirit of the method is sound, and I expect that a successful variant of it should be possible.

The method also has the potential to resolve problems with character-agnostic rating that I have not yet addressed. For example, some players of *PSABR* were found to be systematically underrated due to their repeated choice of a weak character; when switching to a strong character they would appear to be playing a fair game, when in actuality that would be exploiting weaker opponents.

The Relationship Between Multidimensional Skill and Character Selection The method I describe builds upon Elo rather than TrueSkill, which makes it simpler to discuss and experiment with, though somewhat less principled. This method is my own contribution, though has not yet been tested experimentally. Specifically, I will describe a generalization of the Bradley-Terry model, which tracks and accounts for players' own facility with each character in a game, as well as each character's unique strengths. It heuristically learns correlations between the skills required to utilize each character, and the skills possessed by each player, without explicit characterization of what those skills are.

The observation underlying this method is that Elo's failure to capture the varying skill of characters stems from its most basic limitation: the single-dimensional modeling of skill. Despite my claim that it is advisable to treat character choice as a metagame decision, for the purposes of this argument let us imagine it as a decision internal to gameplay, but one of the utmost importance. Then imagine that we had available to us some multi-dimensional generalization of Elo, which learns and tracks each player's level of skill with the varying facets of a game. Since differing characters must be played in significantly different fashions, relying on different basic skills, such a system would *implicitly* track and predict a player's performance with each character. A player who performs well will have the relevant skills increased, and a player who fails will have skills decreased. It may be that some of the implicit skills are unique to a single character, but if such a system worked correctly, it would capture even these axes.

Unfortunately, a multidimensional Elo system is no easy task. If all we are given is the wins and losses of a each match, then we have no information with which to determine what a player's constituent successes and failures were within a match. As such we would have no method with which to update each dimension of a player's skill vector.

My claim is that the characters played act as a higher-order bit set which can inform a multidimensional Elo system. They serve as the ground truth which distinguishes the performance of one match from another. I will propose a system in which each player is represented by a vector of skill values, and the way that this vector is updated after a win or loss depends on the character this player used. The difficulty is that we do not want to assume a priori that what the core skills of a game are, and we certainly do not want to assume what the overlap of skills are between characters. The solution I suggest is to incorporate character skill modeling into the player skill learning process itself.

Modeling Players and Characters with Skill Vectors I wish to assert that for each character there exists a set of skills which its players must utilize to varying extents, in order to perform well with this character. It is thus natural to model characters, as well as players, as vectors in a high-dimensional skill space. A player's vector represents the skills they have available, and the character's vector represents the skills they require to do well. Despite this asymmetry, such a model works excellently, because we can then naturally capture a player's skill using a character as the inner product of these two vectors. Such a model is surprisingly expressive, since we think of each player and character's global advantage as being represented by their vector's magnitude. For vectors α and β, the inner product $< \alpha, \beta >= \|alpha\| \cdot \|\beta\| \cdot cos(\theta)$, where θ is the angle between α and β. In other words, the inner product is a scale-sensitive measure of the angle between two vectors, i.e. their similarity. It captures exactly the extent to which a player's skills 'fit the needs' of a character.

Once skill vectors for each character and player are known, then prediction of win rate is straightforward. We take each player vector's inner product with the vector of the character played in that game. These values then serve as their respective base skills for this game. From here the system operates exactly like Elo: the probability of a given player winning is derived by the logistic function with these inner products as skills, or by sampling from normal distributions with these inner products as means.

The main question, finally, is how we are to infer character and player skill vectors, given no a priori information about a game's skills. The key insight is that the treatment of characters as skill vectors allows us to treat them as players within the Elo system. We can re-envision the taking of inner products for skill as a model of teamwork between two players. (It may indeed turn out to be well-suited to the problem of evaluating the skill of players on a team.) This models a game in which players succeed only if they play similarly – if their skills are well-aligned. Then each game we observe gives us information about 'all four players' in a game – that is, the two players and the two characters. Just as a loss by a player in Elo is treated as evidence that that player's skill is lower than

previously thought, a loss in this system is taken as evidence that the player's true vector has a smaller dot product with the character's true vector than previously thought. It is not known which of these values is incorrect, so *both* are modified. If one of the vectors was actually correct, then it will in time be pushed back to its original position, whereas the other will slowly move towards its correct position. We reduce the dot product of these vectors in two ways. We multiply each of them by a scalar less than one, reflecting our reduced confidence in their magnitude. We *also* increase the angle between the vectors. Again, we both increase angle and reduce magnitude due to lack of information as to the real cause, with the knowledge that future games will correct for our mistakes. Our method for increasing the angle between the vectors is to add a small random vector to each, in the halfspace of vectors having positive inner product with the vectors' difference. In other words, we move each vector away from the other in some random direction, acknowledging our belief that the vectors should be further apart, but in an unknown direction. Yet again we rely on future rounds to push each vector slowly in the right direction. This acts as a random walk in which 'most' steps move the vector in a direction having positive correlation with the correct direction.

Setting up the Model I have now fully described the model for maintaining multi-dimensional Elo ratings for players and characters, with the exception of specifying scale parameters and starting vectors. Scale parameters should be picked empirically so as to optimize predictive correctness. Initial vectors, I claim, can be chosen identically as a unit vector. This ensures that the system starts with the identical performance prediction for all players and characters. The random fluctuations then allow the vectors to spread out through the entire space over time, however many dimensions it is given. Note that characters should converge to their correct values substantially faster, since they participate in so many more games than a given player. Note also that the model is a true generalization of Elo, in the sense that running it with a single dimension of freedom is identical to Elo. For practice it is simplest to reason about in a two-dimensional space, and in fact large potential gains may still be had from such a model. Finally, note that such a system relies on players playing multiple characters, to gain information about their relative skills. But if players do not play multiple characters, the problem is that much simpler, and essentially reduces to measuring players outside the context of characters.

The real value of such a system is to confirmed or denied experimentation, just as it is for Elo. It is not likely that interesting worst-case guarantees on convergence can be proven, as such bounds are not even known for Elo. However, I plan to run experiments in the near future to evaluate the system's feasibility. It may allow for greater accuracy predictions for all players, and character-specific evaluations of player skill (the dot product) which will

make data analysis substantially richer and more accurate.

7.5.4 Non-Traditional uses of Rating Systems

One of my primary insights is to use player ratings as an analytical tool for understanding the design of games themselves, rather than simply a tool for matching players together. However, this is not the first alternative use of rating systems within games.

Coulom [23] applies Elo to the problem of 'pattern rating' in Computer Go. Many Go AI techniques complement their search methods with heuristic evaluations of board patterns, or basic high level features. These heuristics are typically learned from expert games using AI or machine learning methods, rather than explicitly provided expert knowledge. In the case of [23], Coulom proposes using a (pre-existing) team variant of Elo to rate patterns. In the metaphor of Elo, he treats each move selection as a game in itself between competing move options. Each possible move is modeled as a team whose members consist of its constituent patterns; the selection of a given move is treated as a win for that team, and hence evidence for those patterns' superiority over their competitors (those in the other available moves). The advantage to using Elo here is precisely this comparison to the available moves, in contrast to a system that rates patterns based on their absolute frequency. Coulom is not the first author to propose an Elo-like system; [84] and [4] do the same. However, Coulom's work achieves the strongest predictions of player behavior, by utilizing a minorization-maximization technique to find the maximum-likelihood global 'skills' of all patterns.

7.6 Wrapping Up Skill Analysis

I have argued in this chapter that skill is an invaluable facet by which we can make sense of and find patterns in deep games. It is one of the few features of gameplay that can be retroactively treated as a restriction on gameplay, and hence is particularly reliable as a form of balance measure. However, I have also offered caution at over-interpreting the saliency of formal skill measures, and have suggested some remedies for certain issues with the standard choice of Elo. Although these techniques for working with skill cater to a specific game, their insights should be generalizable and testable on other competitive games. Designers always implicitly contextualize their playtest observations in terms of the skill of the players; when we lump these playtests together into a small collection of numbers, it is all too easy to pass over this essential step.

Chapter 8

A FORMULATION OF METAGAME BALANCE

Our discussion of game balance so far has addressed several meanings of the term, yet it has still largely framed a multiplayer game as an artifact instantiated through the interaction of two or more players with the game system. In reality, there is a level of context beyond a single instantation of a game, which has a profound impact on its balance in practice. This context is the community, (or more correctly, overlapping collection of communities) that plays a game. The knowledge, preferences, and skills of the players in a game's community at any moment shape the experience of all the players within it. This phenomenon, and also the specific properties of the community, are known colloquially as the *Metagame* of a game. As a concept, metagame is even less concretely defined than balance itself, but understanding and shaping it (when possible) too is essential to a game's success. I will not attempt to define the notion of metagame, but will rather outline some of its core constituent concepts, and address those directly.

After outlining a variety topics captured by the notion of metagame, I will focus on one major facet: character selection. Here 'character selection' acts a proxy for a broader phenomenon: selecting from a finite set of choices prior to the beginning of a game. As previously discussed, such choices - as minimal as selecting a side in *Chess* or as extensive as selecting a loadout of weapons and abilities in *Call of Duty* - are subtly different from choices made in-game. The first difference is that players are 'locked in' to pre-game choices for the duration of a game. Although many in-game choices also have permanent effects, those that that occur prior to a game *tend* to more sweepingly determine the affordances of play. The second difference is that the mechanisms for allowing metagame choices vary tremendously, and depend both on design and social context. Tournaments offer some structure for determining who can play which 'character', as do leagues, teams, friendships, matchmaking systems, etc. Possibly for this reason, there are little to no existing examples of rigorous reasoning about a metagame.

My treatment of character selection in this chapter rests on a key observation: that measuring character strength as win rate is not completely well-posed, because win rate can depend on an opponent's character selection. In other words, all previous measurements of win rate have implicitly taken a game's metagame into account. If we observe that a

character wins 60% of the time, this really means that a character wins 60% of the time against the current distribution of opponent characters, (and even strategies). To establish a more refined, metagame-sensitive characterization of win rate, we must reason explicitly about the potential states of the metagame, and a character's effectiveness within them. I will discuss several approaches to doing so, finally converging to a solution that again relies on the principles of restricted play. In essence, I take the measurements of in-game effectiveness (discussed in previous sections) as a given, then measure the effectiveness of restricted players of the character selection metagame that surrounds that game. Doing so provides a novel measure of character strength, which captures the necessity and sufficiency of that character's presence in a distribution or community of characters.

In this chapter I show that a collection of natural 'character selection' problems can indeed be easily framed as zero-sum games. In Chapter 8.2 I consider character selection in a repeated game, in a tournament structure, and even as a skill-based investment when joining a community. In Chapter 8.3 I go on to show how such formalization facilitates a new form of reasoning about game balance, whereby the classic question of character strength can be evaluated through the potential states of a metagame. I analyze a major competitive fighting game to show how a richer notion of character strength than current standards can be developed. This model addresses precisely the concerns that many competitive players and analysts have with existing methods.

8.1 Examples of Metagame

Metagame was likely coined for a single concept: the selection of a character before the beginning of a competitive game. If selected simultaneously (as it sometimes is), this process is indeed a classical zero-sum game, whose outcome determines the details (and hence win probabilities) of the subsequent non-meta-game. In this context, the term metagame is perfectly appropriate, although it only has relevance in games with substantial asymmetry. However, the term has now come to refer to any processes and states that occur outside the existence of a single game, that affect the states and outcomes of actual games. It refers to concepts such as

- The fraction of players of a game who *typically* choose each starting condition, character, or race. This complicates the naive zero-sum game formulation, because players have experience with certain characters, natural biases, and partial knowledge of their opponents' experience and biases.

- The distribution of players' skill, commitment, degree of experience with a game. This

can have an amplifying effect, where a glut of skilled players turns away newer players, or vice versa. Communities can also fragment in this way, with so-called 'hardcore' players populating one mode of a game, and casual players populating another.

- The processes through which players are pitted together, from casual play, to tournaments, to random online matchmaking, to ranked online matchmaking.

- The effects of outside media such as blog posts, online videos, patch notes, and books on players' choices and knowledge.

- The popularity of various game modes and rules, and the effects these have on character popularity.

- The written and unwritten codes of conduct of communities, specifying appropriate play and house rules.

This chapter will primarily be conceded with the character selection, yet it is not hard to see how these phenomena overlap with one another. When appropriate, these other issues will be occasionally raised.

8.2 Reasoning about Metagame with Zero-Sum Game Theory

It is possible and valuable to consider balance outside the context of the metagame, but by considering the community, an evaluation of balance will nearly always be more relevant to players. There are numerous difficulties in doing so: communities may be disparate and hard to measure, humans are difficult to characterize quantitatively, metagame can change dramatically with time, and metagame may be effectively non-existent before the release of a game. Yet most balancing work in some way considers at least the possible forms of a metagame.

In this section, I build out the beginnings of a formalization of character selection, the most classical facet of metagame. I will start with a simple explication of the metagame of two skilled players, in the traditional zero-sum context. Although video games are sometimes analyzed in the context of zero-sum games, most examples of which I am aware analyze the in-game choices (such as which attack to perform). This is the first attempt I know of to analyze a metagame as a zero-sum game. For concreteness, I have instantiated this analysis with freely available approximate data from the fighting game with arguably the most active tournament community, *Street Fighter 4*. Here I consider the problem of selecting which of

the game's 39 characters to play.[1]. That said, this game is chosen solely for concreteness. The arguments of this section apply to any game with freely chosen asymmetric starting conditions – from *Starcraft*, to *Team Fortress*, to *PSABR*. Rather than work with the proprietary data of *Playstation All-Stars Battle Royale*, I chose *Street Fighter 4* to highlight the missed implications of a discourse that is already common in a game's public community. I then extend this theory to a richer context, in which we think of many players playing repeated games, and improving with individual characters.

8.2.1 Tier Lists and Matchups

The current version of *Street Fighter 4* (the fourth yearly iteration, *Super Street Fighter 4 Arcade Edition 2012*) is renowned for being extremely well-balanced. Anecdotally, what is often meant when players say this is that it is viable to win a tournament with many of the characters. This is indeed a variant of the most traditional form of balance: character strength. But it's also somewhat subtler than the slightly abstract notion of a character's win rate. A naive notion of character strength categorizes characters into a 'tier list', according to each character's win probability. Yet the phrasing "win a tournament" points to the metagame as a crucial feature of balance; to win a tournament you must defeat several opponents, and to do so must consider the collection of opponents you might face, in aggregate. These opponents might be characterized by their choice of character, their level of skill, or their play style.

For the time being, I will abstract away skill and play style, and focus only on character selection. To choose a character with whom to enter a tournament, (assuming you must keep the same character throughout) you must consider the collection of characters you will face, as characters have distinct pairwise interactions. There are two ways to frame this problem. The first is to imagine each player's choice of character as simultaneous upon entering the tournament. In our simplified model this choice will be driven entirely by the win percentage of pairs of characters. These win percentages can be characterized by a symmetric matrix, and indeed, often are. Figure 8.1 contains the community-built 'matchup tier list', from eventhubs.com [43]. A given entry gives the chance in 10 that the row character will defeat the column character.

There are some large caveats. The first is that this matrix is created by the Eventhubs community[2], and so may be subject to severe confirmation bias, among other biases. What's

[1]It is sometimes more appropriate to think of the game as having 78 characters, because each character has two variants determined by their 'ultra attack', chosen at the start of the game.

[2]Technically, the matrix is built by one or few individuals, but is loosely derived from votes by the website's users.

	SETH	VIPER	CAMMY	AKUMA	F.LONG	RUFUS	SAGAT	BALROG	ADON	IBUKI	ABEL	BLANKA	MAKOTO	BISON	RYU	KEN	YUN	ZANGIEF	DHALSIM	GUILE	SAKURA	CHUN-LI	DEE JAY	JURI	ROSE	GOUKEN	GUY	CODY	FUERTE	YANG	HONDA	GEN	VEGA	DUDLEY	ONI	E.RYU	HAKAN	T.HAWK	DAN	TOTAL
Seth	-	4	4	5	4	5	5	5	4	5	5	5	5	6	6	5	6	7	6	6	4	6	5	6	6	6	4	6	5	6	6	5	6	6	6	5	6	7	7	206
C. Viper	6	-	6	6	5	5	6	4	5	4	6	5	6	4	6	6	5	4	7	6	5	6	5	4	6	6	5	6	5	5	5	6	5	5	6	6	5	6	6	205
Cammy	6	4	-	6	4	5	6	4	5	6	6	6	6	4	6	5	5	4	6	4	5	5	4	6	6	6	5	6	6	5	6	6	6	5	6	6	6	6	6	205
Akuma	5	4	4	-	5	5	6	5	5	5	6	5	5	5	5	5	4	6	6	6	6	6	5	5	5	6	6	5	6	5	6	6	6	5	5	6	6	6	6	204
F. Long	6	5	6	5	-	5	6	4	5	5	5	6	5	5	6	5	5	6	6	5	5	6	6	6	4	5	5	5	5	5	6	5	5	5	6	5	6	6	6	203
Rufus	5	5	5	5	5	-	4	5	5	6	6	5	6	5	4	5	6	3	6	4	6	6	4	5	7	4	4	5	6	6	6	5	6	5	6	6	6	5	7	200
Sagat	5	4	4	4	4	6	-	6	5	4	5	5	5	5	5	6	6	7	5	5	6	4	5	4	6	5	6	6	5	6	6	5	6	4	5	6	6	7	6	200
Balrog	5	6	6	5	6	5	4	-	6	5	5	6	6	5	5	5	5	4	4	4	6	4	5	5	6	5	6	5	5	6	5	5	5	6	6	5	5	5	6	198
Adon	6	5	5	5	5	5	5	4	-	5	5	6	5	5	6	5	5	4	4	6	6	5	5	5	6	6	5	5	6	5	4	5	5	5	5	6	6	4	6	196
Ibuki	5	6	4	5	5	4	6	5	5	-	5	4	5	4	5	5	5	4	6	6	5	5	6	5	5	5	6	5	5	5	5	5	6	5	6	6	6	4	6	195
Abel	5	4	4	4	5	4	5	5	5	5	-	6	5	5	6	5	5	4	7	6	5	4	5	4	5	6	5	5	5	5	6	6	6	5	6	6	6	4	5	194
Blanka	5	5	4	5	4	5	5	4	4	6	4	-	5	4	5	5	5	6	4	6	5	5	6	6	6	6	6	6	6	4	5	5	4	5	5	5	4	8	6	194
Makoto	5	4	4	5	5	4	5	4	5	5	5	5	-	5	5	5	5	4	6	6	5	4	5	5	6	5	5	5	6	5	4	5	6	6	6	5	6	6	7	194
Bison	4	6	6	5	5	5	5	5	5	6	5	6	5	-	5	5	4	4	5	3	6	5	6	6	5	5	5	5	6	5	4	5	5	5	5	5	6	5	6	194
Ryu	4	4	4	5	4	6	5	5	4	5	4	5	5	5	-	5	6	6	4	5	6	5	5	5	4	5	5	5	5	5	6	5	6	5	6	6	6	6	7	194
Ken	5	4	5	5	5	5	4	5	5	5	5	5	5	5	5	-	6	5	4	4	5	6	6	5	4	5	5	6	5	6	5	5	6	5	5	5	6	5	6	193
Yun	4	5	5	6	5	4	4	5	5	5	5	5	5	6	4	4	-	4	7	6	5	6	5	5	6	4	5	5	5	5	6	6	6	5	5	5	5	4	6	193
Zangief	3	6	6	4	4	7	3	6	6	6	6	4	6	6	4	5	6	-	4	4	5	3	4	4	4	4	7	6	5	6	5	4	5	6	6	5	6	5	7	193
Dhalsim	4	3	4	4	4	4	5	6	6	4	3	6	4	5	6	6	3	6	-	6	4	6	6	4	5	6	5	6	5	4	7	5	5	6	5	6	5	6	7	192
Guile	4	4	6	4	5	6	5	6	4	4	4	4	4	7	5	6	4	6	4	-	5	6	5	6	4	5	4	5	4	5	6	5	5	6	5	5	6	7	6	192
Sakura	6	5	5	4	5	4	4	4	4	5	5	5	5	4	4	5	5	5	6	5	-	5	5	6	5	6	5	5	5	5	4	6	5	5	6	6	6	6	6	192
Chun-Li	4	4	5	4	4	4	6	6	5	5	6	5	6	5	5	4	4	7	4	4	5	-	5	5	5	5	5	5	6	5	6	5	5	5	5	5	5	6	6	191
Dee Jay	5	5	6	5	4	6	5	5	5	4	5	4	5	4	5	4	5	6	4	5	5	5	-	5	4	5	5	5	5	6	6	5	5	5	5	5	5	7	6	191
Juri	4	6	4	5	4	5	6	5	5	5	6	4	5	4	5	5	5	6	6	4	4	5	5	-	5	5	5	5	4	4	4	5	5	5	6	6	6	7	6	191
Rose	4	4	4	5	6	3	4	4	4	5	5	4	4	5	6	6	4	6	5	6	5	5	6	5	-	6	6	5	6	4	4	5	5	6	5	5	6	6	6	190
Gouken	4	4	4	4	5	6	5	5	4	5	4	4	5	5	5	5	6	6	4	5	4	5	5	5	4	-	4	4	6	6	6	5	5	6	6	6	5	6	6	189
Guy	6	5	5	4	5	6	4	4	5	4	5	4	5	5	5	5	5	3	5	6	5	5	5	5	4	6	-	4	5	6	5	5	5	5	6	6	6	4	6	189
Cody	4	4	4	5	5	5	4	5	5	5	5	4	5	5	5	4	5	4	4	4	5	5	5	5	5	5	6	6	-	4	4	5	5	5	6	6	6	6	6	188
Fuerte	5	5	4	4	5	4	5	5	4	5	5	4	4	4	5	5	5	5	6	5	4	5	6	4	4	5	6	-	6	5	5	5	5	6	6	5	6	5	187
Yang	4	5	5	5	5	4	4	4	4	5	5	5	6	5	5	5	4	5	4	6	5	5	5	4	6	6	4	4	6	4	-	5	6	5	5	5	5	6	4	6	187
Honda	4	5	4	4	4	4	4	5	6	5	4	5	6	6	4	5	4	5	3	4	6	4	4	6	6	4	5	5	5	5	-	6	6	5	5	5	6	6	6	186
Gen	5	4	4	4	5	5	5	5	5	5	4	5	5	5	5	4	4	6	5	5	4	5	5	5	5	5	5	5	5	4	4	-	5	5	5	5	5	7	6	185
Vega	4	5	4	4	5	4	4	5	5	4	4	6	4	5	4	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	-	5	5	5	6	6	6	183
Dudley	4	5	5	5	5	5	6	4	5	5	5	5	4	5	5	5	5	4	4	4	5	5	5	5	4	4	5	4	5	5	5	5	5	-	5	5	5	4	6	182
Oni	4	4	4	5	4	4	5	4	5	4	4	5	4	5	4	5	5	4	5	5	4	5	5	4	5	4	4	4	4	5	5	5	5	5	-	6	5	6	6	176
Evil Ryu	5	4	4	4	5	4	4	5	4	4	4	5	5	5	4	5	5	5	4	5	4	5	5	4	5	4	4	4	4	5	5	5	5	5	4	-	5	6	6	175
Hakan	4	5	4	4	4	4	4	5	4	4	4	6	4	4	4	4	5	4	5	4	4	4	5	5	4	4	5	4	4	5	4	4	5	4	5	5	-	5	6	169
T. Hawk	3	4	4	4	4	5	3	5	6	6	6	2	4	5	4	5	6	5	4	3	4	4	4	3	3	4	4	6	4	4	6	4	3	4	6	4	4	-	6	166
Dan	3	4	4	4	4	3	4	4	4	4	5	4	3	4	3	4	4	3	3	4	4	4	4	4	4	4	4	4	5	4	4	4	4	4	4	4	4	4	-	148

Figure 8.1: Eventhubs Community Tier List for Street Fighter 4 AE 2012, March 2012

more, the matrix does not specify concretely what level of skill at which it is meant to apply, an admittedly messy concept. However, it is loosely stated that the match ups are meant to reflect high-level tournament play. In other words, the data should be roughly predictive of an average tournament's outcome, subject to the amplification of repeated games. Finally, the data is revised every several months, which is more tacit admission of its subjectivity. The revisions are not just due to improved information. They are also due to changes in the metagame *beyond* character selection. Over time certain strategies and play styles become dominant across characters. These strategic changes are partially due to publicized strategic discoveries, and are partially themselves responses to previous changes in the strategic metagame. In addition, players have their own qualitative skills, some of which are more relevant to certain characters. So two players who may win with the same frequency overall may nonetheless perform poorly with each other's characters, even after practice.

In a blog post about the limitations of tier lists [61], fighting game theorist and designer Artavan Mkhikian (known as 'Maj') writes

> "If the community decides that the game is based on Low Fierce, then [in *Capcom vs SnK 2*] Sagat and Blanka find themselves at the top. If the community decides that the game is based on [Cable's strongest attack], then [in Marvel vs Capcom 2] Cable finds himself at the top. If the community decides that the game is based on Back+Fierce, then [in *Street Fighter 3: 3rd Strike*] Chun Li finds herself at the top. Everyone else's worth is determined by how well they can cope with the kings (or queens) of the hill."

In other words, tier lists and matchups are (correctly) determined by the fundamental strategic metagame – the perceived primary strategic principles of the game. It's not just that certain characters are chosen, but that characters are played differently, changing the value of each matchup.[3]

8.2.2 Character Selection in Varying Contexts

Despite the caveats about matchups, for the purpose of argument I will treat Figure 8.1 as fact, and explore the possible consequences. First consider a single game, in which two

[3]Note that the notion of a single metagame is misleading. Game players compose a patchwork of cultures, some with little overlap. In particular, play styles and (skill levels) are known to vary greatly across continents. In face-to-face games travel is prohibitive; in network games latency is prohibitive. So players within a continent (or country, or city) tend to play each other frequently, and in effect develop their own 'culture'.

players choose their characters simultaneously, based solely on the matchups. This selection process is a classical zero-sum game, in which a player's payoff is the probability that they will win the upcoming match. Framed differently, character selection is a stochastic game in which the expected payoff to the row player of choosing character i, when the column player chooses character j, is $1 \cdot \Pr[i \textit{ beats } j] - 1 \cdot \Pr[j \textit{ beats } i]$. For convenience we discount ties by treating them as randomly selecting a winner.

Indeed, to consider character selection for a full tournament, with character remaining fixed throughout[4] , we can show that the problem is equivalent to selecting a character for a single game.

Theorem 8.2.1. *Let $\rho = \{\rho_1, \ldots, \rho_n\}$ be an optimal mixed strategy over the pure strategies of a symmetric game whose matrix $M_{i,j}$ specifies 'win probabilities'. Then ρ is also a Nash equilibrium for selecting a fixed pure strategy to use throughout a single-elimination tournament with random seeding.*

Proof. Assume for contradiction that ρ is not a Nash equilibrium for the tournament. Then by definition there exists a mixed strategy ϕ by which some player p can achieve win probability strictly greater than 0.5 when all other players are playing ρ. So assume that all other players are indeed playing ρ. We show by induction that the marginal distribution of pure strategies player p encounters at each round is identical to ρ. Let $A_{r,p}$ be the random variable specifying the pure strategy played in round r by player p. Let $W_{r,p}$ be the event that player p plays and wins in round r. Let q_r be the random variable specifying player p's opponent in round r.

Claim: Assume all players but p play mixed strategy ρ for their initial choice of strategy, and maintain that pure strategy for the entire tournament. Then

$$\forall r, i: \ \Pr[A_{r,q_r} = i | W_{1,p}, \ldots, W_{r-1,p}] = \rho_i.$$

We prove the claim by induction over r. The claim holds trivially for the base case of $r = 1$. We thus assume that the claim holds for a given $r - 1$, and show that it holds for round r.

Assume that p won round $r - 1$, else the claim is vacuous. Now, the probability that q_r plays pure strategy i in round r is identical to the probability that q_r played strategy i in round $r - 1$, and won. This holds because the players' pure strategies are chosen

[4]Some tournaments do disallow character-switching, though the majority do not. This is nonetheless not a terrible approximation, as the amount of work to learn a character at a tournament level can be sufficiently prohibitive that most players stick with a single character most of the time. That said, there are numerous noteworthy examples of top players changing characters mid-tournament or mid-match.

independently at the start of the tournament, as are the seedings, so player p's strategy and her opponent's can have had no influence on one another, probabilistically, until now. In other words,

$$\Pr[A_{r,q_r} = i] = \sum_u \Pr[A_{r-1,u} \wedge W_{r-1,u}],$$

where the sum is over the two players u vying in round $r-1$ to become q_r. Then $\Pr[A_{r-1,u} = i \wedge W_{r-1,u}] = \Pr[W_{r-1,u}|A_{r-1,u} = i] \cdot \Pr[A_{r-1,u} = i]$. It is a property of symmetric zero-sum games that each pure strategy in the support of a Nash equilibrium has identical expected payoff (0, specifically) against that Nash equilibrium. By induction, both u and her opponent played round $r-1$ according to ρ, hence $\Pr[W_{r-1,u}|A_{r-1,u} = i] = 0.5$. What's more, $\Pr[A_{r-1,u} = i] = \rho_i$ by induction. Hence $\Pr[A_{r,q_r} = i] = \sum_u 0.5\rho_i = \rho_i$, completing the induction, and proving the claim.

Given the claim, the theorem follows easily. To receive an incentive for deviating from ρ, player p must have a probability strictly greater than 0.5 of winning at least one round. Yet in every round, her randomly distributed opponents play, in aggregate, according to ρ. Since ρ is a Nash equilibrium for the underlying game, this is impossible. Hence ρ is also a Nash equilibrium for the tournament.

□

Extending the proof to double-elimination tournaments should be straightforward.

Theorem 8.2.1 shows that it suffices to consider individual games when considering the distribution of players in a tournament-focused metagame. This is important, as tournaments are perhaps the harshest litmus test of a game's balance. Yet there is another wrinkle of complexity that even simultaneous choice in a tournament obscures. This is the fact that players don't really pick a random character at time of play or of a tournament. Rather they pick a single character early on, and largely stick with them through their play careers. Fortunately, there is a natural analog to simultaneous play, which comes from the primal-dual algorithm for solving zero-sum games. If we model all players as fixing a character forever, then a new (perfectly rational, perfectly informed) player entering the game picks a character as a response to the existing community of players. We can show that this community naturally balances itself out, and indeed although each player's decision is deterministic, its demographics approach exactly the Nash equilibrium of an individual game. This ensures that when playing a randomly matched opponent (as in a tournament) a player's odds of winning are 50%, just as they would be if all players picked characters simultaneously.

For the sake of simplicity, we assume that each player chooses a character to maximize her win rate against the community at the time of her choice, and does not account for

players who may join in the future. However, in the limit this assumption can be relaxed, as the future distribution of incoming players will match the current distribution, according to the theorem itself. Hence our model is that at time t, a player community C_t consists of $t-1$ players, and a new player joins the system, choosing a character who maximizes her win probability in a game against a randomly chosen player from C. Ties are broken arbitrarily. In reality such a player would be attempting to maximize the fraction of a large number of future games that will be won; by linearity of expectation this is equivalent.

Claim 8.2.2. *As t approaches infinity, the distribution of C_t approaches some Nash equilibrium ρ for the underlying game.*

Proof. The process of assembling C_t replicates exactly the algorithm of *fictitious play* [16] Fictitious play computes a Nash equilibrium in a two-player zero-sum game by playing a series of 'fictitious' rounds of the game. In each round, both players choose a strategy deterministically; namely, they choose the purse strategy that maximizes the expected payoff against their opponent's history. That is, each player optimizes under the assumption that the opponent will play a mixed strategy in which each pure strategy is played according to its historical frequency. [74] proves that in a finite zero-sum game, each player's historical distribution converges to an optimal strategy.

Now observe that in a symmetric zero-sum game, both players will play identically, assuming ties are broken identically. In other words, in a symmetric game it is sufficient to respond to one's own historical actions. Equivalently, we may think of a succession of new players playing one round, each playing the pure strategy that optimizes against the historical strategies of all past players. The community-assembling process described above models exactly this setting. Hence the distribution of characters chosen converges to a Nash Equilibrium over the underlying game. □

Fictitious play converges to a Nash equilibrium at a rate that is at worst exponential in the number of strategies, but is conjectured by Samuel Karlin to converge at a polynomial rate [50]. There is no doubt that real game communities do not match the exact distribution of a Nash equilibrium, nor could they with a finite number of members. Yet they would begin to approach it, in the simplified model I have put forward.

8.3 Estimating Fairness in a Metagame

It is now apparent that it is roughly appropriate to frame a community's character selections as picking a mixed strategy from the matchup matrix. (We have still brushed under the rug questions of distinct player skill, character change, learning, etc. but we can only grow our model so fast.)

Character	Probability
Balrog	0.327
Seth	0.184
Ibuki	0.142
Cammy	0.102
Akuma	0.102
Guile	0.081
Zangief	0.020
Sagat	0.020
C. Viper	0.020

Table 8.1: A Nash equilibrium for choosing characters in Super Street Fighter 4 AE 2012, according to the matchups of Figure 8.1. All unlisted characters are assigned probability 0.

8.3.1 Measuring Character Strength with Optimal Strategies

We can begin to test the zero-sum model of character selection (and the data itself) by simply computing the optimal strategy. Despite being entirely natural, I have never seen community character distribution framed as mixed strategy selection, nor have I ever seen the optimal strategy of these matchup matrices computed. Table 8.1 contains the first such optimal strategy.

From a technical perspective, Table 8.1 seems perfectly reasonable. The characters chosen with the highest probability are those who have few bad matchups, and are difficult to exploit. The less frequent characters 'cover the gaps' in the frequent characters. Yet a fan of competitive fighting games would likely be rather incredulous toward Table 8.1. For instance, it seems to say that Balrog is the dominant force of the game, and that Guile is a very strong character. Yet these characters are anecdotally thought of as strong-but-not-amazing, and middling, respectively. Indeed, one of the top American players, PRBalrog, was famous for vocally sticking with Balrog despite their supposed weakness, and finally abandoned him recently for Fei Long, a character more traditionally considered 'top tier'.

There are many reasons for Table 8.1's apparent mismatch with reality. One is that it should not necessarily be thought of as character strength – just the frequency with with a character should be played. But this interpretation is equally problematic, as Balrog players are nowhere near this common in competitive games. More obvious criticisms include inaccuracies in the matchup data, players' inherent bias toward some characters, partial information of the community, and nonuniform dependence on skill. (Some characters are thought of as more difficult to play 'to their full potential'.)

A more technical critique of Table 8.1 is also perhaps the most straightforward one.

Character	Probability
Balrog	0.333
Seth	0.154
Akuma	0.154
Ibuki	0.115
Cammy	0.077
Guile	0.05
Zangief	0.026
Sagat	0.026
C. Viper	0.020
Rufus	0.013

Table 8.2: An example alternative Nash equilibrium for choosing characters in Super Street Fighter 4 AE 2012.

Many zero-sum games do not have unique optimal strategies. In the case of the matchups in Figure 8.1, this is indeed the case. We can confirm that there exist different Nash equilibria for this game with a simple constraint: we lower bound the probability assigned to some unused character, like Rufus. Choosing 1% as an arbitrary threshold, we attain the alternative Nash equilibrium in 8.2. There could be countless very different equilibria to this game, and indeed as a zero-sum game, all the convex combinations in-between are also equilibria.

Yet the two given Nash Equilibria are quite similar; the KL-divergence of the second distribution is only 0.059 from the first. Further investigation reveals this to be a necessity! We can begin to frame the problem through restricted play, by restricting players of the metagame to play at most or at least some frequency of a given character. Indeed, constraining the most common characters to probabilities above over below their current values always results in a suboptimal strategy. For instance, constraining the probability of Balrog above 0.33962 or below 0.319444 results in suboptimality. A strong bias toward *Balrog* is required for optimality.

8.3.2 Generalizing Optimal Strategies for a Nuanced View of Character Strength

Fortunately for the applicability of the character selection model, optimality is an unnecessarily strong property about which to reason. Even if our model and data were perfect, players would never converge to a true Nash equilibrium. What's more, by reasoning about near-optimal strategies, we encode more tolerance for those inaccuracies in the model and

Character	Probability
Balrog	0.75
Akuma	0.25

Table 8.3: An near-optimal mixed strategy for Super Street Fighter 4 AE 2012, in which Balrog is constrained to be played with probability at least 0.75. This mixed strategy has win rate 45%.

Character	Probability
Ibuki	0.187
Fei Long	0.183
Akuma	0.143
Rufus	0.099
Guile	0.092
Ken	0.078
Bison	0.065
Yun	0.059
C. Viper	0.033
Sagat	0.031
Zangief	0.030

Table 8.4: An near-optimal mixed strategy for Super Street Fighter 4 AE 2012, in which Balrog is constrained to never be played. This mixed strategy has win rate 49.4%.

data. Perhaps good players will play any mixed strategy that ensures them some high non-optimal win probability. For the time being, let us arbitrarily posit 45% as the threshold win probability that 'appears fair' to a player. Then we can ask for mixed strategies that guarantee at least a 45% win rate. Considering Balrog again, we find that to win at least 45% of the time, a player can play Balrog in as much as 75% of games, and as little as not at all! Indeed, restricting playing Balrog with probability 0 results in a 49.4% win rate - barely distinguishable from optimal! These two mixed strategies can be seen in Table 8.3 and Table 8.4. Unlike Table 8.2, both of these mixed strategies differ substantially from Table 8.2, and understandably so. Table 8.3 includes a character whose matchups against Balrog's weak matchups are strong, but has few poor matchups himself. Table 8.4 includes a variety of characters with similar match ups to Balrog, in order to replace him. In particular, the second-most common character is Fei Long, generally thought to be similar to but *stronger* than Balrog.

I posit that for the purposes of a game designer interested in balancing her game, the

probability with which a character can be played and still achieve a high win rate is a useful overall measure of a character's strength.This is a richer example of a *restricted play* balance measure, where we are concerned with the overall win rate associated with a restriction, rather than the binary distinction between optimality and sub-optimality. Using such restrictions frames game balance in terms of the *potential* space of metagames. When a designer worries that a character may be overpowered, her genuine concern is probably that at some point, in some community, that character may be played exceptionally frequently. After all, few designers worry about overpowered characters that nobody will ever play.[5] This measure describes exactly the worst case that might arise among mostly rational, highly informed, highly skilled players, assuming good matchup data. It is approximate, but informative nonetheless. If a 45% win threshold is too arbitrary, we may pick any other threshold.

In fact, for richer information, we can draw the entire graph for a character, plotting the maximum win probability achievable while playing a given character with probability p. For example, Figure 8.2 shows this range of probability for Balrog. By necessity, the worst achievable win rate is 0.4, because Balrog's worst matchup is a 4, so playing him deterministically can only guarantee the opponent a 40% win rate by making a deterministic pick. Contrastingly, Figure 8.3 shows the range of achievable probabilities for a weak character, Hakan. Hakan is largely derided (with some exceptions) within the competitive fighting game community, and indeed is played with probability 0 in all the mixed strategy's I've shown so far. Looking at the graph, Hakan's effect on a community's viability is certainly more negative than Balrog's, yet not excessively so. Hakan too has no matchups worse than a 4, so the effect near saturation is similar. It is around the 50% play rate that they differ most drastically, with Balrog still near optimal, and Hakan harming win rates linearly as his play rate increases. Such graphs have promise for explicating the potential extent to which a character might be considered 'overpowered'. The fact that Hakan is considered weak may in itself be evidence that 45% is simply too low a win rate threshold.

Sub-optimal mixed strategies have a natural intepretation for an individual playing a simultaneous-character-selection game. But what do they mean in the context of a community's distribution of characters? When constraining one player's mixed strategy's in the linear program, the opponent fundamentally remains unconstrained. After all, the mechanism of the linear program is to evaluate a mixed strategy by its worst case performance against an opponent's fixed pure strategy. If we wish to interpret these constrained mixed strategies as saying something about potential metagames, we can think of them as com-

[5]It seems unlikely that a designer would be concerned with the minimum probability at which a character can be near-optimally played, as this simply says that a character will be played often in every possible metagame.

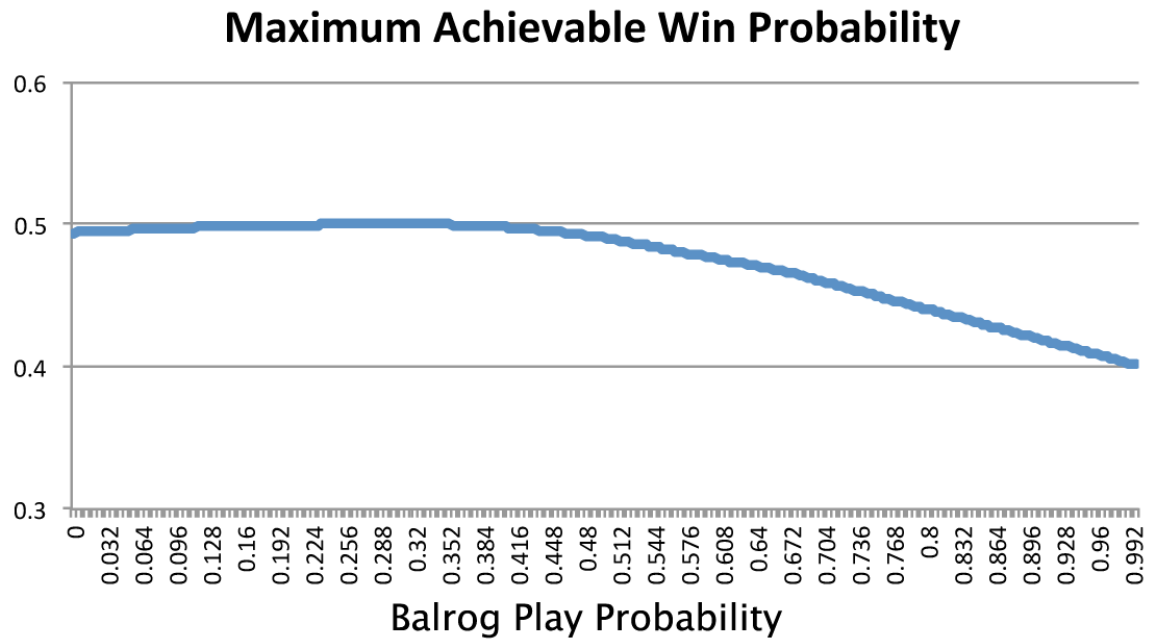

Figure 8.2: Achievable win probability with mixed strategies that play Balrog at a fixed percentage.

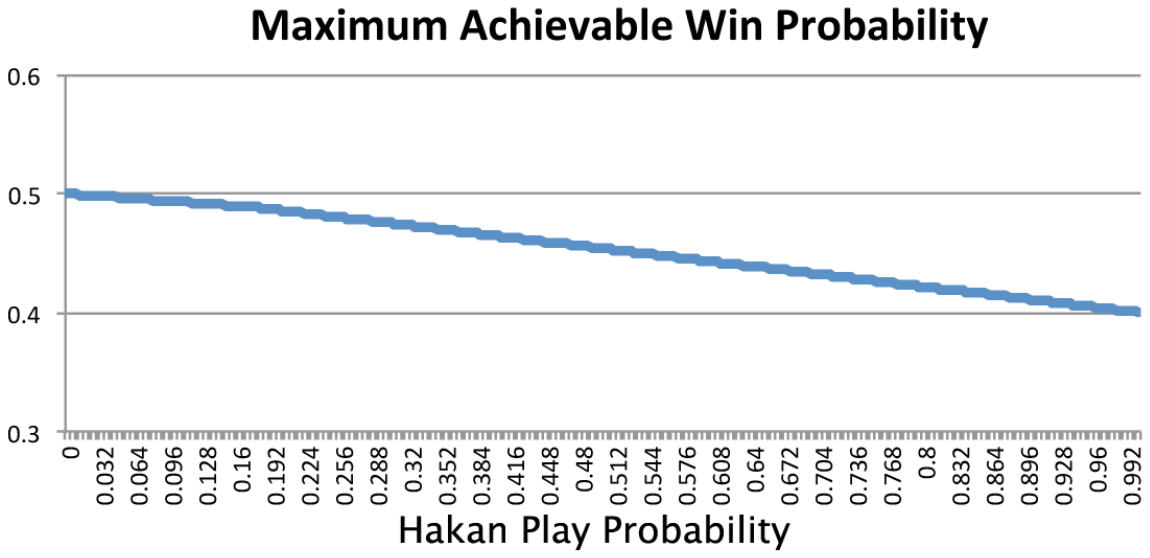

Figure 8.3: Achievable win probability with mixed strategies that play Hakan at a fixed percentage.

menting on the space of 'near-stable' metagames. That is, consider a community distributed according to a mixed strategy such as that of Table 8.3. A new player entering this community can at most achieve a win rate within the 'margin of acceptability' of 45-55%. As such, the community need not change to respond to these possible players, nor must these new players choose characters that change the distribution in the limit.

8.4 Wrapping Up Metagame Analysis

In this chapter I have shown how the fundamental metagame of a fighting game – and indeed any game with freely chosen asymmetric starting conditions – can be viewed as a zero-sum game. Moreover, I have shown how solutions of this game constrained by restricted play can be used to reason about the fundamental balance of these starting conditions. In other words, the principles of my thesis extend beyond gameplay itself, and can be applied at higher levels of abstraction. This potential is essential to the feasibility of balancing games in the long-term. As a live game matures, its community often begins to become the most apparent influence on its balance. Simultaneously, developers often become more conservative in their balance patching, as players become accustomed to the status quo. These two factors make it essential for designers to be able to reason about the long-term potential steady states of their metagame, and potentially address them *before* they become a problem. Restricted play analysis of metagame effectiveness is one step in this direction.

Chapter 9

CONCLUSION

In this dissertation, I have worked to progress the discourse and tools of game balancing. I have focused in particular on quantitative balance evaluation, the class of techniques for judging an existing game's balance using quantitative methods. My approach has been multi-faceted, as well it should be for a problem that can mean many things despite a common thread. I have no doubt opened more questions than I have resolved, but this will be a success provided it opens some eyes to the possibility of balancing games differently.

For the most part, the motivation for my work has arisen from my central thesis, which I restate below.

Thesis

Game balance can be understood through the effectiveness of explicitly modeled players.

This thesis statement was made concrete with the principle of restricted play: that we can measure balance by the win rates of restricted players. The consequences of restricted play within this dissertation have been varied. For automated analysis of games, restricted play can be applied directly to AI agents. When working with human players, a small set of properties (character, skill, etc.) can be measured and effectively restricted retroactively, yet these few properties are some of the most crucial to competitive games. When working with entire communities of human players, restricted play can be used to reason about the stability of a distribution of player habits, and thus predict possible future balance states.

Of course, there is much room for improvement of balancing practices outside the domain of this thesis statement. For instance, my technique for model-based data exploration in Chapter 6 does not in itself depend on deep understanding of players. However, as with so many aspects of gameplay analysis, it benefits greatly from a surrounding player-centric context. For instance, customizing the win model itself to player skill will produce more accurate and informative results.

In this chapter my dissertation comes to a close. I will first outline my contributions retrospectively. I will discuss a few avenues of potential future work, focusing on the unification of the projects discussed here and on their potential applications. I then conclude with some final words.

9.1 Contributions

This thesis has been a broad exploration of computational and statistical perspectives on game balancing. Aside from my thesis statement, another common theme has been this: that balancing is hard because it is technical, and that it is this technicality that admits rigorous and quantitative approaches, if we look deeply enough to the problems we are solving. I have focused my efforts on *evaluating* the balance of games, rather than adjusting it. This latter goal is no less important, and is indeed more ambitious. If I have made any progress in our understanding of the former problem, I believe it will have been a success for both.

The contributions of this dissertation include:

- An overview and synthesis of three prominent views on the meaning and role of game balance.
- A discussion of the legitimacy of game 'theorycraft' as a nascent form of science and engineering.
- A survey of existing work in quantitative balance evaluation, for both late-phase human gameplay analysis, and early-phase simulated gameplay analysis.
- A quantitative formulation of disparate balance measures in a single framework, using the win rates of restricted players.
- A example of automated balancing with restricted play, applying a prototype tool to an educational card game.
- A simplified recursive presentation of the Monte-Carlo tree search algorithm.
- A proposed technique for applying restricted to play to complex games using calibrated Monte-Carlo tree search agents.
- A case study of the actual uses of telemetry data in balancing a major competitive game.
- An approach to guiding data exploration through simplified causal models.
- A generalization of a traditional Poisson-based model for competitive game outcomes.

- A stable of techniques for utilizing player skill data to make sense of gameplay data.
- An ad-hoc method for measuring skill in asymmetric games, and a proposed 'principled' approach to extend it.
- A formal model of a typical form of metagame, and a method for evaluating possible convergent states of the metagame, instantiated with real numbers from a popular competitive game.

9.2 Future Work

There are countless ways in which the methods of this dissertation could be extended and improved. However, these will not be my primary concern in this section. Instead I will look at the potential consequences of my work, at the broadest level. I will first give a high-level picture of what balancing might look like in a world in which these methods have matured. I will then discuss a potential application that is not just made easier, but made *possible* by these methods.

9.2.1 A Picture of Future Game Design

Having described distinct categories of quantitative balance analysis, I will now propose a complete picture of how these methods might operate in unison. I will describe the entire pipeline of a possible future balancing process. Although I assume that these balancing technologies develop, the view I present is nonetheless somewhat conservative and traditional. It looks quite similar to a modern effective process today, but amplified to its full potential. I omit entire classes of methods that are somewhat at odds with traditional notions of balance, such as automated difficulty adjustment, online micro-tuning, crowd-sourced content, AB testing, and more. Such forward-looking methods are exciting, but essentially outside the bounds of this thesis. I would claim that they all can benefit from a more thorough understanding of the fundamentals of balancing, and that is what I have sought to establish.

I envision a model of game balancing that acknowledges games as living systems, where balancing decisions aim to create a holistically enjoyable experience rather than optimize some platonic ideal. Yet I also see the utility of platonic ideals for balance, as baselines to which player experiences can be compared. Successfully designing any user-centric system requires a back-and-forth process between a designer's intentions and the users' view of the product. A designer envisions an experience and crafts an artifact that she believes will encourage that experience. Users inevitably drift from that experience, and designers must

understand the reality well enough to either nudge users in the direction of their vision, or reshape their vision into a new local optimum around the current reality.

Regardless of a designer's commitment to her original vision, responding to a live system requires her to successfully navigate three challenges. The first is to deeply understand the breadth of experiences of the players (taking high-level player traits into account), and how this experience differs from their vision. The second is to devise modifications that shift the player experience in the intended direction without overly disrupting their experience. The third is to evaluate the effectiveness of such modifications without indiscriminately foisting it upon the player base. This thesis has attempted to address the first and third of these challenges, through data analysis and player simulation respectively. Assisting the second challenge with computation is absolutely a worthwhile goal; however, for the purpose of this thesis I have left the creativity up to human beings. If nothing else, the first and third steps are prerequisites for the second.

I will attempt to make concrete the process of addressing these challenges using hypothetical mature quantitative balance analysis techniques. For an example scenario I take a hypothetical recently-released competitive multiplayer game awaiting its first balance patch.[1] Such an environment calls upon the breadth of balancing techniques. This is also an environment with which I am personally familiar, having been intimately involved in the first balance patches for *PSABR*. (Of course, many of the tools I might have hoped for had not yet been created or refined.) The steps of the process, which will loop back on each other, are: collect data, analyze gameplay, identify goals, devise solutions, simulate gameplay, playtest locally, and release patch.

1. **Collect Data (and Watch for Outliers)**. During the first weeks of a game's release, the priority will typically be collecting data rather than analyzing it. Players are not yet familiar with the game, and even flagrantly negative trends - such as a single dominant character - may disappear quickly as players discover methods to respond to them. That said, major outliers should be investigated. Designers can employ their understanding of the strategy space to consider the potential longevity of these outliers, spurring further investigation and a possible rush patch. Although rush patches can endanger a game by potentially introducing more errors and reducing player faith in the designers, in extreme cases it is necessary to avoid turning off players permanently before they become invested.

[1] In reality, any release of a game deserves a similar process. However, as patches stack up, they often become more conservative, and so the analysis and revision process can become correspondingly conservative.

2. **Analyze Gameplay**. Once trends in usage and impact of gameplay features have begun to converge, deep data analysis may begin. This step is exemplified by the methods of Chapters 6 and 7. The goal is nothing less than a complete picture of the space of real and potential gameplay. The goal is fundamentally unreachable, but through visualization, data mining, and statistics it is possible to achieve successive approximations.

   Much like some branches of data-driven science, controlled experiments are possible but difficult. Instead, a wealth of data is collected, and scientists choose what to look at and which questions to ask, so as to enlighten the areas of greatest ignorance and facilitate specific engineering goals. For a competitive game, goals include understanding the interactions between components, the effectiveness of strategies and their combinations, the strength of starting conditions, the difficulty of challenges, and the preferences of players. Moreover, player progression (both in-game progression and advancement of skill) should be understood and correlated to these questions.

3. **Identify Goals** Once the state of a game is understood as well as time permits, it is up to designers to decide what constitutes a 'problem'. Emergent gameplay will never take on its exact intended form (nor is this intended form fully specified); certain deviations will be acceptable, while others will not. Some constraints may come from business necessities, such as the requirement that a certain character be used. Other constraints will come from players, who vocally demand that a certain play style perform effectively. Still other constraints will come from the designers themselves, who might feel confident that a certain change will produce a more enjoyable experience. In the end, a list of balancing goals may be formulated: throws should be made less frequent, character X should be made more powerful, game duration should decrease, a given attack should have more viable response options, etc. Some goals are fundamentally incompatible, and this will be the source of difficult decisions.

4. **Devise Solutions** The designers will now possess a list of balancing goals which concern the high-level dynamics of the game. The goal is then to devise low-level solutions that may bring about those dynamics. To take an example from above, how could one adjust a fighting game to make throws less common? The problem is not hypothetical: the developers of *Street Fighter x Tekken* had exactly this goal when release their first major balance patch, *Street Fighter x Tekken 2013*.

   There are diverse possible solutions to the simple goal of increasing throw frequency, each of which affects other aspects of balance and uniquely impacts the experience

of gameplay. For example, throws could be made more damaging; failed attempts at throws could be made less punishable; other moves could be made relatively less effective; the metagame could be manipulated to make throw-heavy characters more viable in other ways; counter-throws could be removed or made less effective; throws could be made to telegraph less obviously; execution of throws could be made less difficult; repetition of throws could be made to provide bonuses; secondary benefits could be added to throws. Making such decisions is technically difficult because each change can interact with each other. Yet the process is also the heart of creativity in balancing. The interesting choices made in balancing add spice and color to a game, engaging the minds and curiosities of players.

In the case of *Street Fighter x Tekken*, the designers opted (among other changes) to add secondary benefits to throws. Namely throws were modified to deplete a player's 'recoverable health' on success. Thus as a player inflicts recoverable damage, her incentive to throw the opponent increases, forcing both players to consistently re-evaluate the likelihood of a throw, on average bringing it up. This clever solution simultaneously increases the depth of strategy and improves throws without making them consistently overpowered. It would be very hard to build a machine that could devise such a solution any time in the near future.

Once a concrete change has been proposed, it must be implemented. In balancing this is often the easy part. So as to not be too disruptive, balancing changes tend to be small, and are often purely quantitative. For this reason I focus next on evaluation.

5. **Simulate Gameplay** In this picture of balancing, the first step to evaluating a change is rapid simulation. I assume that an agent has at this point been calibrated to play the original game with the skill of a moderate player, as discussed in Chapter 5.3. That agent is then pitted against itself in the modified game under a wide variety of restrictions. As described in Chapter 5, extreme win rates of any of these modified agents can inform the designer of potential red flags in the design: an action suddenly made useless, a win state made infeasible, an outcome made certain too early.

   Additional confidence in the validity of these win rates can be obtained by running the same agents on the original game, looking for *differences* in win rates between the versions. Nonetheless, these win rates will always be very approximate. They serve primarily as a method to shrewdly select variants for 'real' playtesting. For example, a designer may decide to reduce the startup time of some move, yet be uncertain whether she should reduce the startup by three, four, or five frames. This level of precision decision-making is at times simply beyond the intuition and knowledge of a

human. This is where restricted play simulation becomes valuable. Rather than begin playtesting with an arbitrary choice, she begins with the variant whose restricted play balance features have shifted the closest toward her goals. Alternatively, the red flags of these balance features might be so flagrant that the designer chooses to go back to the drawing board, rather than begin even minor human playtesting. In this case she moves back to step 4, and otherwise forward to Step 6.

6. **Playtest Locally** Once a patch has been developed and appears viable using restricted play, it is ready to be played by a small number of players. Sometimes even a single play-through by the designers can reveal a design flaw in a new variant. The goal of restricted play simulation is to skip many of these 'bad iterations', but surely some will still be present. More common are flaws that become clear after a small group of designers, testers, family members, or focus testers have spent time with the game, experimented with the changes, and explored possible consequences. If a problem persists from the old version, or a new problem becomes apparent (which it often will), the process is again sent back to Step 4. Occasionally, a goal may resist solutions for sufficiently long that it is given up on. Eventually, the game reaches a state that the designers feel confident is a sufficient improvement. It is ready for deployment.

7. **Release Patch (or Beta Test)** In the best case, a new patch can be released at large scale in a separate environment that will not disrupt the primary metagame. *World of Warcraft*, *League of Legends*, *Starcraft 2*, and other large competitive games employ exactly this strategy with beta environments. Fully predicting the effect of a balance patch on a game community is a literal impossibility. The strategy space of most interesting games will be too large to fully explore, and the addition of hundreds of thousands of eyes will simply find things that a modestly sized team or computational effort will not. Moreover, the human element cannot be ignored: certain 'intended' playstyles will never be discovered, others will be too difficult, and still others will be consciously ignored. Indeed, any given game has multiple possible semi-stable metagames, and knowing which if any it will reach is impossible.

For the above reasons, a beta release of a balance patch is greatly desirable. However, for technical, business, or social reasons it is not always possible. In such a situation, it is all the more important that each other step of this balancing process is approached vigilantly.

Note that nowhere in the above narrative is the work of Chapter 8 mentioned, in which the metagame-sensitive strength of characters is evaluated. This is a consequence of my

choice to explain the process of an early balance patch. Metagame-sensitive balance is less relevant early in a game's lifecycle, when there is rarely enough reliable, converged data to inform on the matchups of each pair of characters. Yet as time progresses, such an approach becomes increasingly relevant. For a game like *Street Fighter 4*, which has now been on the market for several years, the matchups and metagame are a primary concern of both designers and players. Individual matchups are well-known, and player can observe changes in the metagame begin to calcify to undesirably limited states. If developers were to evaluate the matchup data after the game was somewhat well-understood on a fine-grained basis, but before the metagame began to converge, they would be in a position to make subtle adjustments to keep it from doing so. I believe that such a process will also be a key feature of future balancing methodologies.

The history of game design is littered with severely imbalanced games, and even severely imbalanced balance patches. Yet other games have come out the door of release in a startlingly stable state. The ability of certain studios to do so somewhat reliably is a testament to the fact that balancing is not just a blind stab in the darkness, a submission to randomness. It is possible to do it well. The contributions and ideas of this thesis should make it possible to do it better.

9.2.2 One Step Toward Democratization of Design

The focus of this work has been on evolving the traditional balancing cycle, with an eye toward making the process more efficient (in terms of time spent) and more effective (in terms of game quality in the eyes of the designer). These are not the only potential benefits of improving the balancing process, however. As balancing techniques become more powerful a phase change may be possible, in which qualitatively new forms of game design open up. In particular, it may finally be feasible to create a game that features mechanically substantive player-created content.

The Challenge of Player-Created Content At a basic level, player-created content is already quite common. The last ten years in particular have seen a boom of player creativity as a core draw for games. Platformers such as *Little Big Planet* and *Soundshapes* allow players to create their own levels and share them in a public bazaar. Massively-Multiplayer Online (MMOs) games such as *Second Life* allow players to script essentially anything they imagine, and share it with or sell it to other players. Traditional MMOs such as *Everquest 2* allow players to craft homes for their avatars. Construction and exploration games such as *Minecraft* allow players to create entire worlds simply through the collection and placement of various types of cubes. Games of many genres allow players to release

their own modifications ('Mods') to a game's core code or scripts, essentially creating new games.

Despite the breadth of player-created content, there remains a gaping hole in the medium of games: player-created *mechanics* in a shared ecosystem. Every game (of which I am aware) that opens up creation to its players is conservative in one of the following two senses. Either players' creations are entirely superficial, so that they cannot affect the affordances of other players; or creations are siloed, so that they affect only the creator themselves or whichever players explicitly opt to interact with it.[2] I claim that there is a single primary cause at the root of these limitations: balance. Consider a multiplayer game in which players are free to create mechanically novel content and impose it on other players. It could dramatically disrupt those players' experience in any number of ways. It could make the game unfair, make certain regions or actions unviable, make games end abruptly, or create any number of other unacceptable circumstances, all of which can reasonably fall under the purview of balance. As I've argued throughout this thesis, such consequences can emerge as a result of nearly any mechanical change, be it a large one like the creation of a virtual meteor, or a small one like a 10% reduced duration of a single attack.

Certainly there are challenges outside of balance to integrating player-created mechanics into a single ecosystem. It can be difficult to give players the tools to create mechanics in a way that is expressive, that interact with each other seamlessly, and that have no security risks. Yet these challenges have been tackled quite successfully in previous games, when balance is not a concern. The next step is to address the problem of balance, ideally in a multiplayer competitive game. Creativity in a shared rule-driven environment is has the potential to amplify the human benefits of player creativity.

The question then is how to go about making player-created mechanics feasible. It seems clear that designers must still play a role in maintaining game balance. Indeed, there is nothing in principle stopping such a game's creation today. Players could create an item, system, character, or level, and simply submit it to the design team. If the designers feel it will not harm the game's balance, they could simply add it to the ecosystem. The problem is only one of resources. Balancing is a laborious and difficult process. Evaluating possibly thousands of contributions from player (potentially in combination with each other) would be prohibitively time-consuming. Perhaps more pointedly: balancing is a specialized craft with which few individuals have experience. The chances are low that an individual player's

[2]Technically *Second Life* breaks both these rules, but it is able to do so by not being a game in the traditional sense. There are no defined goals to reach nor challenges to overcome, so balance is a concept which has little to no meaning in this context. *Minecraft* gets away with breaking both rules because players largely play in shared worlds with their friends or associates, and implicitly negotiate shared guidelines for acceptable play.

contribution would be an acceptable (or desirable) change. Players who have experience in balancing, or have a very deep understanding of the game, or have undergone several rounds of iteration with designers might have a greater hope of producing creations that would be approved. But it is likely that designers would have to find those rare practical contributions buried under a haystack of infeasible changes.

In fact, the game that has come closest to this vision also abruptly stops short here. In 2010, the online first-person shooter *Team Fortress 2* introduced the *Steam Workshop*, an environment for players to craft their own items and weapons and submit them to the game's creators. This laudable system has attracted a good deal of press and adoration from players, as well as arguably enriched the game. There is one catch: player submissions are purely superficial. When a player creates a new gun, they build the 3D model for it, which may suggest quite clearly its functionality. Yet the game's designers build the functionality of their choice themselves, carefully reasoning about the new weapon's potential role in the game's already-crowded ecosystem. This limitation takes away what I believe to be the foremost potential of player-generated content: involving players in a game's *design*, and allowing them to feel their choices' impact when they play.

Enabling Player-Created Content Is there potential for something like a *Steam Workshop*, where players do not just model their weapons and items, but also design and script their functionality? I wholeheartedly believe there is, and that it will rest upon tools like those in this thesis. Just as I have partitioned quantitative balance analysis into two very different parts – early-phase and late-phase – this same partitioning applies to player-generated content.

The first technique to make player-created mechanics feasible is automated feedback. As designers scarcely have time to playtest their own carefully thought-out gameplay additions, it would be infeasible to even cursorily test the many contributions of mixed quality that players might produce. If players themselves could receive automated feedback on their designs using restricted play techniques like those of Chapter 5, it could help point out flaws that they could iterate on themselves. More importantly, it would give them the ability to predict – however roughly – their design's chances for approval. If desired, designs with sufficiently poor automated evaluation could be barred from submission altogether, or deprioritized for examination. In the process of using such a system, players may even learn about game balancing. To be an effective contributor, they would be typically forced to gain some intuitive understanding for the measures being evaluated, giving them basic knowledge of some of the fundamentals of game balancing, however reductionist. They would also learn through experience how to manipulate their design's balance, using the restricted play measures as guideposts to measure their progress. This information combined

with playtests by the contributors, alone or with friends, could slowly create game designers without them even realizing it.

The second technique for allowing player-created mechanics concerns the evaluation process. Just as the designer's own content should go through phases of evaluation, it is all the more important for player-created content. With their own content, designers know intimately the *intention* for an element. They have the option of ruling it out if it is not played in a particular way, and they can find effective strategies for for its use that were implicitly envisioned and may be hard to discover. Essentially, evaluating player-created content takes away all the advantages of being a designer, and makes the designers players. Of course, a player may provide a short design document to the designers, detailing the vision for a feature and its intended use. However, such a document can only convey so much nuance; the designers will always be at a disadvantage. This is where data becomes crucial. By releasing player contributions in limited environments such as beta tests, designers can collect the entire experience of using these features. Making *sense* of all this data would be harder than ever, and this is where refined gameplay analysis techniques like those of Chapters 6 and 7 become essential. There is nothing about these techniques that is particularly targeted at player-created content. Yet they will do the job very well, as they are focused focused on capturing the breadth of the gameplay experience, through decomposition of the primary causes of statistics and behavioral properties of players.

Building a true player-influenced shared world would not be easy, even with these techniques. The view I have presented is in many ways an oversimplification. For example, it is unlikely that user-submitted contributions would be approved or denied wholesale. Rather, designers may compromise with a small modification in order to bring a contribution in line with their more subtle design goals. Such issues would have to be worked out through iteration both broad and small.[3] However, the goal of a player-influenced shared world is admirable, in some sense inevitable, and will be much more likely to succeed with balancing tools that can support it.

9.3 Final Words

This work is a small step toward the larger goal of mechanizing technical aspects of game design, thus freeing designers to concentrate on the most creative aspects. I have largely

[3]It may be the case that a more aggressive methodology is necessary. For example, one can imagine a sufficiently advanced economic system that dynamically adjusts access to game resources and cost of features, so as to make every player contribution fair. A more advanced system might employ a sort of 'pyramid scheme' of designer-players, with the most knowledgeable players evaluating the best content and passing it up the line.

omitted several entire branches of research toward this goal, including procedural content generation, player experience modeling, game mechanic search, and dynamic difficulty adjustment. These areas, along with quantitative balance analysis, are exciting and vibrant new fields of research, still defining themselves and their boundaries. They currently operate largely independently, and have largely not been adopted by the industrial game development community.

Yet we should not be concerned by the fact that technical game design is in its infancy. As researchers we work with problems that are more difficult than we sometimes acknowledge. They push at the boundaries of our abilities and knowledge in many areas: psychology, assisted creativity and design, data science, algorithmic game theory, artificial intelligence, and game design itself. I genuinely believe that to improve the process of game designers is to learn fundamental things about humans and technology. These lessons will be invaluable in a changing world full of ever-more-complex systems, processes, and data sources. Of course, games themselves will benefit the most. Games are at once a nascent art form, a booming industry, a monolithically popular form of entertainment, and a potential source for good. Technologists are fortunate to have discovered a role for ourselves in pushing their potential further and further still.

ACKNOWLEDGMENTS

You are in a dark place. The only way is out.
> out
You are born. It's kind of intense.

Your parents bestow years of unconditional love and acceptance on you. Parents: "You can be anything you want to be when you grow up, even if it makes no sense to us." What do you want to be?
> mathematician, game designer, or ninja
Please select two options.
> mathematician and game designer?
Accepted. Please wait. What would you like to do in the meantime?
> play games
You play a lot of games. Actually, you mostly just talk about them incessantly, and your family wants to strangle you for it.
> look at family
Your father introduces you to computers, and supports you immensely. Your mother introduces you to performance, and helps you see who you are. Your brother Taliesin torments you mercilessly, and makes you an immeasurably better person. Your younger siblings Anika and Tibet are born, who are wonderful but force you to be some kind of an adult. Your friends the Landises are nuts, and show you how high you can aim. You get a good education for a while, then... eh.
> go to college
Roll D20...
20/20. That's better. You go to Berkeley and are immediately surrounded with incredible friends known as 115, who totally get you.

It is your first exam. Differentiate something easy like x^2.
> 2....x^2?
FAIL. You'll get the hang of it. College continues. Would you like to do an internship?
> y

You intern at Electronic Arts. Your boss, Jim Greer, is an excellent mentor, and continues to be one, even after you resist his ethically dubious solicitations to stay.

You return to college. You do research with Marc Davis, Mor Naaman, and Satish Rao. Marc and Mor are stunningly supportive. Satish tells you theory research is depressing, but in a fun way. Do you want to go to grad school or make games?

> ask friends

Eric Miles and Nimesh Singh: "Go to grad school like us!"

> go to grad school

Eric and Nimesh ditch. It's okay, they'll go later. You apply to grad school, and get into the University of Washington, a school you've loved since you were a teenager. You start school.

> meet the love of my life

Roll D100...

100/100. You meet Nodira Khoussainova. She works three doors down from you. She is brilliant, charming, beautiful, and unlike anyone you've met. Her family are unreasonably kind to you. She will accompany you through grad school and beyond. The rest of the world looks brighter now.

> start research

Your advisor is now James Lee. James teaches you how to think with discipline, how to write, how to believe in yourself, and how to cower in fear. Research is depressing, but in a fun way.

> freak out

Lindsay Michimoto appears. She tells you that self-doubt is natural, that you are doing great, and that she's rooting for you, as she has done many times before. You are no longer freaked out.

> make friends

You make incredibly close friends, more slowly this time. You level up many comrades; the following are among those with maximum XP: Dan, Saleema, Ben, Kayur, John, Travis, Eva, Liz, Ian, Ian, Paul, Abe, Alexis, and Emily. Your mentor is Yaw Anokwa. The Deus Ex Detective Agency does your detective work. Several others are very, very close to your heart.

Would you like to do more internships?

> y

You intern at Yahoo! Research with Malcolm Slaney, and at Microsoft Research with Thomas Moscibroda. They show you research outside your area, permanently increasing your breadth-of-interests.

You return to school, and work with other talented co-authors in several fields.
> i
Your CV contains: Mohammad Moharammi, Luis Ceze, Karin Strauss, Laura Effinger-Dean, Michael Piatek, Siddhartha Sen.
> switch areas
You sort of switch areas, sort of again. You join the Center for Game Science, with Zoran Popović. You collaborate with Erik Andersen, Yun-En Liu, and Anna Karlin. An undergrad, Alex Miller, joins your party. They help you with formative work that prepares you for your dissertation. Zoran and James support you in striking your own path in research, pursuing risky work that you love and believe in.

Would you like to do yet another internship?
> um, y
Your brother uses magic to meet and introduce you to Leif Johansen, who hires you at SuperBot Entertainment, jumpstarting your game design career. Leif and everyone at SuperBot are tremendously encouraging and open-minded to your unorthodox methods. It is really, really fun. Even though you lose at *Street Fighter* every night.

You return to school. What would you like to do?
> propose thesis. write thesis. graduate. follow nodira to san francisco. work as game designer.
Slow down.
> no
Fine. You crunch for six more months, nearly going crazy, were it not for the infinite patience and support of the wonderful people you've met along the way. Anup Rao, Michael Toomim, Cecilia Aragon, and Adam Smith jump in to save you, as well as much of CGS, and others you've met so far. Kayur, Eric, Ian and Nodira fly in to watch your defense. Friends celebrate with you. You feel as if an epic adventure has ended, made so much more meaningful by the care of others.

It was definitely worth it.
> ...
> ...
> begin next chapter

BIBLIOGRAPHY

[1] B. Abramson. Expected-outcome: a general model of static evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(2):182 –193, feb 1990.

[2] Mike Ambinder. Valve's approach to playtesting: The application of empiricism. Game Developer's Conference, March 2009.

[3] E. Andersen, Y.E. Liu, E. Apter, F. Boucher-Genesse, and Z. Popović. Gameplay analysis through state projection. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 1–8. ACM, 2010.

[4] Nobuo Araki, Kazuhiro Yoshida, Yoshimasa Tsuruoka, and Junichi Tsujii. Move prediction in go with the maximum entropy method. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 189–195. IEEE, 2007.

[5] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002.

[6] Joshua Barnett and Leanna Archambault. How massive multiplayer online games incorporate principles of economics. *TechTrends*, 54(6):29, 2010.

[7] R. Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1):19, 1996.

[8] C. Bauckhage, K. Kersting, R. Sifa, C. Thurau, A. Drachen, and A. Canossa. How players lose interest in playing a game: An empirical study based on distributions of total playing times. In M. S. El–Nasr S. Lucas, S.–B. Cho, editor, *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, Granada, Spain, Sept. 11–14 2012.

[9] A. Bauer and Z. Popovic. Rrt-based game level analysis, visualization, and visual refinement. In *Proceedings of the Eight Artificial Intelligence and Interactive Digital Entertainment Conference*, AIIDE '12, 2012.

[10] Mike Birkhead. Tips from a combat designer: Fibonacci game design. `http://www.gamasutra.com/view/news/40294/Tips_from_a_combat_designer_Fibonacci_game_design.php`, 2012.

[11] bitingpig. League of legends community forum - esports competition rulings: Suspensions of mouz envision and aaa linak. http://eune.leagueoflegends.com/board/showthread.php?p=5032725#post5032725, 2012.

[12] B. Bouzy and B. Helmstetter. Monte-carlo go developments. In *ACG. Volume 263 of IFIP., Kluwer (2003) 159174 5 Typically the*, pages 159–174. Kluwer Academic, 2003.

[13] B. Bowman, N. Elmqvist, and T. Jankun-Kelly. Toward visualization for games: Theory, design space, and patterns. In *Transactions on Visualization and Computer Graphics*. IEEE, 2012.

[14] Ralph A. Bradley and Milton E. Terry. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4), 1952.

[15] Louis Brandy. Using genetic algorithms to find starcraft 2 build orders. http://lbrandy.com/blog/2010/11/using-genetic-algorithms-to-find-starcraft-2-build-orders/, 2010.

[16] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[17] C. Browne. Elegance in game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[18] Cameron Bolitho Browne. *Automatic generation and evaluation of recombination games*. PhD thesis, Queensland University of Technology, 2008.

[19] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[20] B. Chan, J. Denzinger, D. Gates, K. Loose, and J. Buchanan. Evolutionary behavior testing of commercial computer games. In *Evolutionary Computation, 2004. CEC2004.*, 2004.

[21] L. Chittaro, R. Ranon, and L. Ieronutti. Vu-flow: a visualization tool for analyzing navigation in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on*, 12(6):1475–1485, 2006.

[22] D. Churchill and M. Buro. Incorporating search algorithms into rts game agents. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[23] Rémi Coulom. Computing elo ratings of move patterns in the game of go. In *Computer games workshop*, 2007.

[24] Rmi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Proceedings of Computers and Games 2006*. Springer-Verlag, 2006.

[25] Pierre Dangauthier, Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill through time: Revisiting the history of chess. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. Curran Associates, Inc., 2007.

[26] N.G.P. Den Teuling. Monte-carlo tree search for the simultaneous move game tron. *Univ. Maastricht, Netherlands, Tech. Rep*, 2011.

[27] Jörg Denzinger, Kevin Loose, Darryl Gates, and John Buchanan. Dealing with parameterized actions in behavior testing of commercial computer games. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG)*, pages 37–43, 2005.

[28] Joris Dormans. Simulating mechanics to study emergence in games. In *Workshop on Artificial Intelligence in the Game Design Process (IDP11) at the 7th Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE)*, 2011.

[29] A. Drachen and A. Canossa. Analyzing spatial user behavior in computer games using geographic information systems. In *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era*, pages 182–189. ACM, 2009.

[30] A. Drachen, A. Canossa, and G.N. Yannakakis. Player modeling using self-organization in tomb raider: underworld. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 1–8. IEEE, 2009.

[31] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *Proceedings of IEEE Computational Intelligence in Games, CIG 2012*, 2012.

[32] Dysent. 116 `http://urbad.net/blue/us/16473618356-116_Armor_Pen_only_giving_66`, 2009.

[33] David Dyte and Stephen R Clarke. A ratings based poisson model for world cup soccer simulation. *Journal of the Operational Research society*, pages 993–998, 2000.

[34] The Economist. Faster, higher, no longer. `http://www.economist.com/node/21559903`, 2012.

[35] A. E. Elo. *The rating of chessplayers, past and present*. Batsford, 1978.

[36] M.C. Ferreira de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: A survey. *Visualization and Computer Graphics, IEEE Transactions on*, 9(3):378–394, 2003.

[37] A. Gagné, M. El-Nasr, and C. Shaw. A deeper look at the use of telemetry for analysis of player behavior in rts games. *Entertainment Computing–ICEC 2011*, pages 247–257, 2011.

[38] S. Gelly and D. Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM, 2007.

[39] Richard George. Meta knight: Banned from super smash bros. brawl. `http://www.ign.com/articles/2011/10/03/meta-knight-banned-from-super-smash-bros-brawl`, 2011.

[40] Miroslaw Gilski, Maciej Kazmierczyk, Szymon Krzywda, Helena Zábranská, Seth Cooper, Zoran Popovic, Firas Khatib, Frank DiMaio, James Thompson, David Baker, et al. High-resolution structure of a retroviral protease folded as a monomer. *Acta Crystallographica Section D: Biological Crystallography*, 67(11):907–914, 2011.

[41] Mark E Glickman. The glicko system. *Boston University*, 1995.

[42] Kirk Goldsberry. The kobe assist. `http://www.grantland.com/story/_/id/8719297/how-kobe-bryant-missed-shots-translate-new-nba-statistic-kobe-assist`, 2012.

[43] Kirk Goldsberry. Tiers for super street fighter 4 arcade edition v2012 by the eventhubs community. `http://www.eventhubs.com/guides/2008/oct/17/street-fighter-4-tiers-character-rankings/`, 2012.

[44] Jaime Griesemer. Design in Detail: Changing the Time Between Shots for the Sniper Rifle from 0.5 to 0.7 Seconds for Halo 3. Game Developer's Conference, 2010.

[45] R. Herbrich, T. Minka, and T. Graepel. Trueskill: A bayesian skill rating system. *Advances in Neural Information Processing Systems*, 19:569, 2007.

[46] P Hingston. A turing test for computer game bots. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm? arnumber=5247069, 2009.

[47] N. Hoobler, G. Humphreys, and M. Agrawala. Visualizing competitive behaviors in multi-user virtual environments. In *Proceedings of the conference on Visualization'04*, pages 163–170. IEEE Computer Society, 2004.

[48] R. Hunicke, M. LeBlanc, and R. Zubek. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, pages 04–04, 2004.

[49] Alexander Jaffe, Alex Miller, Erik Andersen, Yun-En Liu, Anna Karlin, and Zoran Popović. Evaluating competitive game balance with restricted play. In *Proceedings of the Eight Artificial Intelligence and Interactive Digital Entertainment Conference*, AIIDE '12, 2012.

[50] Samuel Karlin. *Mathematical Methods and Theory in Games, Programming, and Economics: Two Volumes Bound as One*. Dover publications, 2003.

[51] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 49:193–208, 2002.

[52] Firas Khatib, Seth Cooper, Michael D Tyka, Kefan Xu, Ilya Makedon, Zoran Popović, David Baker, and Foldit Players. Algorithm discovery by protein folding game players. *Proceedings of the National Academy of Sciences*, 108(47):18949–18953, 2011.

[53] Jun H. Kim, Daniel V. Gunn, Eric Schuh, Bruce Phillips, Randy J. Pagulayan, and Dennis Wixon. Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, 2008.

[54] kipto. Does mastery round down? `http://www.mmo-champion.com/threads/817588-Does-mastery-round-down/`, 2010.

[55] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML) '06.*, pages 282–293. Springer, 2006.

[56] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2), 1996.

[57] Bryan Lawson and Shee Ming Loke. Computers, words and pictures. *Design Studies*, 18(2):171–183, 1997.

[58] Matt Leone. Data entry, risk management and tacos: Inside halo 4s playtest labs. `http://www.polygon.com/2012/10/24/3538296/data-entry-risk-management-and-tacos-inside-halo-4s-playtest-labs`, 2012.

[59] Y.E. Liu, E. Andersen, R. Snider, S. Cooper, and Z. Popović. Feature-based projections for effective playtrace analysis. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 69–76. ACM, 2011.

[60] Marlos C. Machado, Gisele L. Pappa, and Luiz Chaimowicz. A Binary Classification Approach for Automatic Preference Modeling of Virtual Agents in Civilization IV. In *IEEE Conference on Computational Intelligence and Games (CIG)*, september 2012.

[61] Artavan Mkhikian (Maj). Tiers and you: The survivors guide. `http://sonichurricane.com/?p=163`, 2012.

[62] Joe Marks and Vincent Hom. Automatic design of balanced board games. In Jonathan Schaeffer and Michael Mateas, editors, *AIIDE*, pages 25–30. The AAAI Press, 2007.

[63] B. Medler. *Play with data-an exploration of play analytics and its effect on player expereinces*. PhD thesis, Georgia Institute of Technology, 2012.

[64] B. Medler, M. John, and J. Lane. Data cracker: developing a visual game analytic tool for analyzing online gameplay. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2365–2374. ACM, 2011.

[65] D. Moura, M.S. el Nasr, and C.D. Shaw. Visualizing and understanding players' behavior in video games: discovering patterns and supporting aggregation and comparison. In *ACM SIGGRAPH 2011 Game Papers*, page 2. ACM, 2011.

[66] Mark Nelson. Game metrics without players: Strategies for understanding game artifacts. In *Workshop on Artificial Intelligence in the Game Design Process (IDP11) at the 7th Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE)*, 2011.

[67] Mark J. Nelson and Michael Mateas. A requirements analysis for videogame design support tools. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, pages 137–144, New York, NY, USA, 2009. ACM.

[68] A. Normoyle, J. Drake, M. Likhachev, and A. Safonova. Game-based data capture for player metrics, 2012.

[69] Dmitry Nozhnin. Predicting churn: Data-mining your game. `http://www.gamasutra.com/view/feature/170472/predicting_churn_datamining_your_.php`, 2012.

[70] Dmitry Nozhnin. Predicting churn: When do veterans quit? `http://www.gamasutra.com/view/feature/176747/predicting_churn_when_do_veterans_.php`, 2012.

[71] Christopher A Paul. Optimizing play: How theorycraft changes gameplay and design. *Game Studies*, 11(2), 2011.

[72] Chris Pruett. Hot failure: Tuning gameplay with simple player metrics. `http://www.gamasutra.com/view/feature/6155/hot_failure_tuning_gameplay_with_.php`, 2010.

[73] Rivkah. Hunter faq mop edition. `http://elitistjerks.com/f74/t130554-hunter_faq_mop_edition_read_before_asking_questions/`, 2013.

[74] Julia Robinson. An iterative method of solving a game. *Annals of mathematics*, pages 296–301, 1951.

[75] R. Romero. Successful instrumentation: Tracking attitudes and behviors to improve games. Game Developer's Conference, February 2008.

[76] K. Rudin. Actionable analytics at zynga: Leveraging big data to make online games more fun and social, 2011.

[77] Jesse Schell. *The Art of Game Design: A book of lenses*. Morgan Kaufmann, 2008.

[78] D. Silver and G. Tesauro. Monte-carlo simulation balancing. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 945–952. ACM, 2009.

[79] David Sirlin. Balancing multiplayer competitive games. Game Developer's Conference, 2009.

[80] Adam Smith, Mark Nelson, and Michael Mateas. Computational support for playtesting game sketches. In *Proceedings of the Fifth Artificial Intelligence and Interactive Digital Entertainment Conference*, AIIDE '09, 2009.

[81] Adam M. Smith, Mark J. Nelson, and Michael Mateas. Ludocore: A logical game engine for modeling videogames. In *IEEE Conference on Computational Intelligence and Games (CIG)*, 2010.

[82] A.M. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan. An inclusive taxonomy of player modeling. *University of California, Santa Cruz, Tech. Rep. UCSC-SOE-11-13*, 2011.

[83] Constance Steinkuehler and Sean Duncan. Scientific habits of mind in virtual worlds. *Journal of Science Education and Technology*, 17(6):530–543, 2008.

[84] David Stern, Ralf Herbrich, and Thore Graepel. Bayesian pattern ranking for move prediction in the game of go. In *Proceedings of the 23rd international conference on Machine learning*, pages 873–880. ACM, 2006.

[85] Keith Stuart. The metrics are the message: how analytics is shaping social games. `http://www.guardian.co.uk/technology/gamesblog/2011/jul/14/social-gaming-metrics`, 2011.

[86] G. Synnaeve and P. Bessiere. A bayesian model for opening prediction in rts games with application to starcraft. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 281–288. IEEE, 2011.

[87] O. Teytaud and S. Flory. Upper confidence trees with short term partial information. *Applications of Evolutionary Computation*, pages 153–162, 2011.

[88] R. Thawonmas and K. Iizuka. Visualization of online-game players based on their action behaviors. *International Journal of Computer Games Technology*, 2008:5, 2008.

[89] Tryndamere. League of legends community infographic. `http://na.leagueoflegends.com/news/league-legends-community-infographic`, 2012.

[90] E.R. Tufte and PR Graves-Morris. *The visual display of quantitative information*, volume 31. Graphics press Cheshire, CT, 1983.

[91] A. Tychsen and A. Canossa. Defining personas in games using metrics. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, pages 73–80. ACM, 2008.

[92] G. Van den Broeck, K. Driessens, and J. Ramon. Monte-carlo tree search in poker using expected reward distributions. *Advances in Machine Learning*, pages 367–381, 2009.

[93] John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[94] G. Wallner and S. Kriglstein. A spatiotemporal visualization approach for the analysis of gameplay data. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 1115–1124. ACM, 2012.

[95] B.G. Weber, M. John, M. Mateas, and A. Jhala. Modeling player retention in madden nfl 11. In *Proceedings of Innovative Applications of Artificial Intelligence*, 2011.

[96] B.G. Weber and M. Mateas. A data mining approach to strategy prediction. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 140–147. IEEE, 2009.

[97] A. Weinstein, C. Mansley, and M. Littman. Sample-based planning for continuous action markov decision processes. In *ICML 2010 Workshop on Reinforcement Learning and Search in Very Large Spaces*, 2010.

[98] G. Zoeller. Development telemetry in video games projects. Game Developers Conference, 2010.