Game Theoretical Insights for Incomplete Information Texas Hold‘em Poker Games

Max B. Philipp

Master of Business Intelligence and Process Management

January 24th, 2018

Berlin School of Economics and Law

First Supervisor: Prof. Dr. Markus Löcher

Second Supervisor: Prof. Dr. Michael Faustino Bauer

Immatriculation Number: 77210308278

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Contents

List of Tables

List of Figures

Sworn declaration

I hereby formally declare that I have written the submitted Master‘s thesis entirely by myself without anyone else’s assistance. Where I have drawn on literature or other sources, either in direct quotes, or in paraphrasing such material, I have referenced the original author or authors and the source in which it appeared.
I am aware that the use of quotations, or of close paraphrasing, from books, magazines, newspapers, the internet or other sources, which are not marked as such, will be considered as an attempt at deception, and that the thesis will be graded as a fail. In the event that I have submitted the dissertation - either in whole or in part - for examination within the framework of another examination, I have informed the examiners and the board of examiners of this fact.

Max Philipp
Berlin, January 24th 2018

Abstract

Poker is an interesting environment for artificial intelligence research. It is a game of imperfect information with multiple competing agents facing each other in a world of limited information, risks and deception. The observed average playrate is a commonly used indicator for holecard selection patterns of agents in Texas Hold´em poker. It serves as a fundamental feature in any modeling approach.
In this paper we discuss why the average playrate is a biased metric and derive an unbiased new metric, the “maximum willingness differential” from empirical data. Finally the metric is formalized mathematically in a way that one can derive it from the observed playrate. Further implementation of this new metric could enhance agent-modeling techniques and ahead/behind-research.

Chapter 1

Introduction

The very first decision of any poker hand is whether or not you are playing with the two cards you have been dealt, the so called "Holecards". Players tend to play only the strongest of hands, yet their threshold on which they decide to not play a hand is based on the players preferences. The selective behavior about Holecards is called "Hand Selection". One could classify a player by the number of hands he plays, the more selective he is, one could argue that his "Range" of playable hands is smaller than that of a player who plays more hands. Which Holecard combination is strong and which isn´t is common knowledge among poker players, so the argument is that one could list all possible combinations of the two Holecards and list them in descending order of their winrate. A player would always prefer a hand which is of higher winrate over another. Naturally, when considering this ranking of Holecards, there is a threshold of which a player decides to not play a hand.

Opponent Modeling in poker is one if not the most imporant aspects of the game. The ability to observe the opponents behavior and to adjust to it is crucial, so that (Billings et al. 1998) stated: "One cannot be a strong poker player without modeling your opponent´s play and adjusting to it." In his paper Billings introduced a poker playing algorithm called Loki (named after the nordish god of deceit), which has been state of the art at the time as well as the foundation of research on Artificial Intelligence of the University of Alberta and made them the leading innovators in the field of computer played poker.

An important differentiation when talking about poker playing algorithms are the two possible mindsets. An algorithm can either be optimal or maximising. Optimal play ultimately leads to a Nash-Equilibrium state in which all agents have found sets of strategies and behavior which minmax their play. For it is true that a optimal algorithm will outperform any other agent who is not playing within the bounderies of the Nash Equilibrium, it fails to exploit weaknesses. This leads to suboptimal choices when facing suboptimal play. A maximizing algorithm is set up to find and exploit weaknesses in suboptimal play, thereby performing better in a non-Nash-Equilibrium state. Nash-Equilibrium models are self-trained until convergence, which means until a set of rules is found which leads to equal min/maxed performance among all agents. One could question this approach, since in real world applications, there is no Nash-Equilibrium player, hence an optimised agent is in fact not optimised. Still in academic literature Nash-Equilibrium algorithms are the majority and we should understand why before we continue.

The Association for the Advancement of Artificial Intelligence (AAAI) is the host of the Annual Computer Poker Competition (ACPC) which is held during the „AAAI Workshop on AI for Imperfect-Information Games“. In the competition teams compete against each other to identify the best algorithm. The game: 1v1 Texas Hold´em poker. In each round the algorithms play 3000 hands against each other. Over multiple rounds of play, the algorithms gathers enough data to get themselves into a approximate-optimal playing state. In real life application, it is barely possible to play against a player for those number of hands. To illustrate, on fulltiltpoker.com, one of the leading online poker rooms, 60-100 hands are played per hour and table. Not just that getting 3000 observations of a single player on a single table would take between 30 and 50 hours, but also no human player is playing for those amounts of time (at least not in one session). Though it is possible to collect those numbers over time. To play accordingly against unknown agents, an algorithm has to utilise generalized, prior information.

Billings suggested six important factors for a”world class poker player“:

- Assessment of Hand Strength
- Assessment of Hand Potential
- Betting Strategies
- Bluffing
- Unpredictability
- Opponant Modeling

In the next section we will take a look at how some of the six aspects of poker are assessed by Billings.

Hand Strength (HS) can be calculated in a enumeric approach: First we calculate the rank of our Holecards and the Community cards. In R, we could use `holdem::handeval` to perform this operation. This returns a single numeric value which indicates the quality of the hand. To identify the HS, we compare this number with the quality-meassure of all other possible Holecards. Our agent´s hand is “ahead” if our value is bigger than those of our opponent, “tied” for the pot when the values equal each other and “behind” when we are below the opposing hand value. The HS is then calculated as the percentage of times we are ahead or tied over all possible outcomes. $HS = (ahead + tied/2)/(ahead + tied + behind)$

To illustrate a hands potential, here is an example: Consider Holecards like [6s 9s] on a [7h 8s 2s] Board (s, h, c and d represent the suits spades, hearts, clubs and diamonds). At this point the hand has zero HS, but with two cards to come, this hand can be very strong. Every 5 and every 10 gives the Straight and every Spade results in a Flush. There are 4 fives, 4 tens and 9 Spades left in the deck. Since one five and one ten is a spade, there are 15 cards in the deck left. We can calculate the probability of one of them occuring during the next two cards using the reversed probability (not getting one of them), which results in: $1 - (32/47 * 31/46) = 54.11\%$ chance of hitting on the next two cards.

Billings differentiates positive potential and negative potential: Positive potential is the chance that a hand which is behind on the flop will be ahead at the end of the game. Negative potential is the chance that a hand which is ahead will be behind at the end of the game. For both cases, a loop over all possibilities can calculate these numbers. The Effective Hand Strength (EHS) is calculated summing the winrate and positive potential of a hand: $EHS = HS_n + (1 - HS_n)xPpot$. Billings correctly recognized that enumerating over all possible Holecards is misleading, since hand selection will occur during the Pre-Flop phase, which means that not every hand combination is equally likely to being played by the opponent. Instead of weighting each observation in the Hand Strength calculation equally, it can be weighted by the possibility that a set of Holecards is played by an agent. This refers to our concept of Ranges. It is quite interesting to read the original paragraph by Billings:

> "The initial weights are based on the starting hands each opponent will play. The most important observed information is the frequency of folding, calling and raising before the flop."

This translates that from the chance of calling/raising versus folding we can deduct the Range of a player.

> "We deduce the mean (m, representing the median hand) and variance (σ, for uncertainty) of the threshold needed for each player’s observed action."

Here it becomes interesting: Calculating the median Holecard rank and its variance, requires perfect information in a game which is by definition a game of imperfect information. This may come from the fact that on poker bot versus poker bot level all hands are shown, but in reality the only information we can gather reliabily are counts of actions (fold, call, raise) and occasionally a hand we can see at a Showdown.

Billings takes the median hand rank (e.g. “income rate”) and calculates an interval of $\pm\sigma$ around it. A hand with an income rate above this threshold gets 100% possibility, Holecards below the boundries get a weight of 0.01 assigned to them. Within the boundaries Holecards are assigned linear decreasing values from 1 to 0.01 with the median at .5.

> "A weight of 0.01 is used for “low” probabilities to avoid labeling any hand as “impossible”. While this approach will not reveal certain opponentspecific tendencies, it does provide reasonable estimates for the probability of each possible starting hand."

Not just Billings but also many other authers are using the observed, average playrate as an indicator for the Hand Selection Range of a player. This thesis´ focus is to show that the observed average playrate is a biased metric and to introduce a new metric, called “maximum willingness differencial” (“mwd” or “M”), which can be a better representation of a players Range. mwd is explored from empiric data at first (“mwd”) and derived analytically later on (“M”).

Chapter 2

Additional Literature

The landscape of quality papers on poker is quite sparse, the following section will introduce a selection of papers which have proven to be relevant for this thesis.

Sklansky and Malmuth (1999) has been the foundation of modern poker theory (Rubin and Watson 2011). They developted a classification system, the so called "Sklansky Groups" of Holecards and wrote multiple other books about advanced poker mechanics.

Felix and Reis (2008) developed an opponent modeling algorithm and tested it successfully against multiple basic poker playing agents. Their approach was based on Billings et al. (1998): They estimated hand potential by adjusting the Range of an opponent depending on a player classification. The classification was based on the observed average playrate and a players aggressiveness.

Dalpasso and Lancia (2014) estimated hand strength according to Billings et al. (1998) based on linear programming techniques. They have been able to achive high prediction accuracy (average error of 4 percent) with a set of only 5 out of 73 features.

Billings et al. (2002) continuation of Loki, "Poki", "uses learning techniques to construct statistical models of each opponent, and dynamically adapts to exploit observed patterns and tendencies." The mentioned "learning technique" is a neural network. Still the work is based on full information in a computer versus computer setting.

Bowling et al. (2017) essentially solved heads-up Limit Texas Hold'em poker. Limit poker is a massive state reduction comparing to no-limit Hold´em. They developed a new algorithm named CFR+ and have been able to solve game-states three magnitudes larger than possible before.

Mehrabi and Haghighat (2009) are using the Apriori algorithm for association rule mining of pre-flop and flop situations to predict Ranges. They only found rules up to a maximum of 78.6 percent confidence.

Ranca (2013) investigates features for Bluff detection in no-limit Hold´em based on data of the ACPC. The features achived ~.8 AUC on a classification tree algorithm.

Ekmekci and Sirin (2013) build an ensemble learning model from a neural network, support vector machines and k-nearest neighbors to predict future actions of opponents and achived .83 percent average accuracy.

Ponsen et al. (2008) are suggesting a baysian approach to opponent modeling utilizing prior learning as the basis for a relational regression tree. Starting at 200 observed played hands, the model starts to perform better with prior learning than without.

Axelle Moreau and Chauchard (2016) and Rubin and Watson (2011) are topic related review papers.

Chapter 3

Poker Rules and Glossary

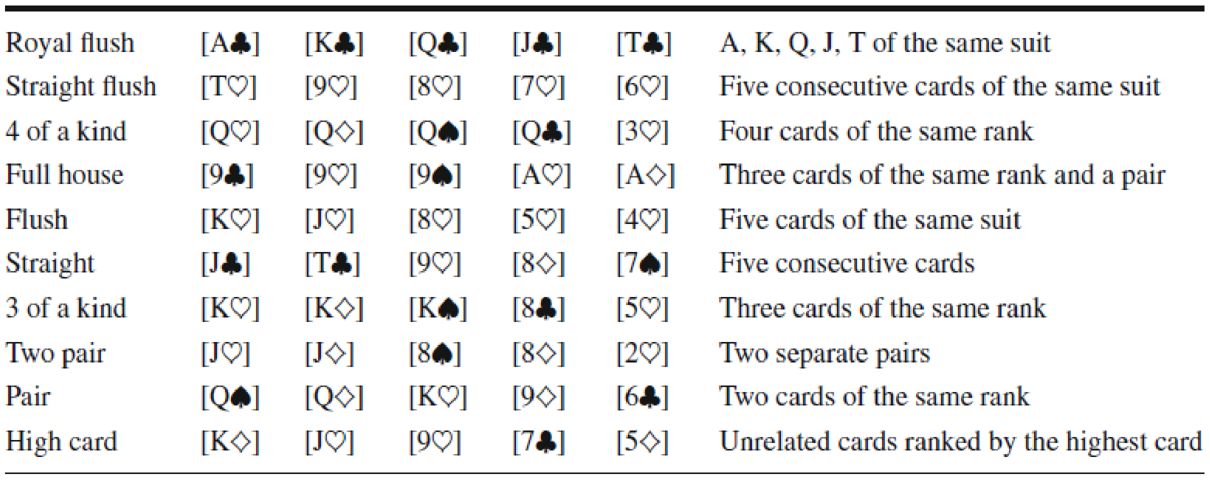
In this chapter we will cover the basic rules of No-Limit Texas Hold´em as well as commonly used keywords.

3.1 Glossary

The entries in this short glossary are taken from (Sklansky and Malmuth 1999).
Action: The betting in a particular hand or game.
Active player: A player still in the pot.
All-in: Having all one´s money in the pot.
Bet: To put money in the pot before anyone else on any given round.
Blind: A forced bet that one or more players must make to start the action on the first round of betting.
Board: The community cards in the center of the table.
Call: To put in the pot an amount of money equal to an opponent´s bet or raise.
Check: To decline to bet when it is your turn.
Fold: To drop out of the pot rather than call a bet or raise.
Heads-up: Playing against a single opponent.
1v1: Heads-up.
Loose: Playing more hands than the norm.
Pot: The total amount of money wagered at any point in a hand.
Raise: To bet an additional amount after someone else has bet.
Showdown: The turning up of all active players´cards at the end of the final round of betting to see who has the best hand.
Tight: Playing fewer hands than the norm.

3.2 Poker Rules

Poker is played with a deck of 52 cards (13 different values from 2 to Ace in 4 colors (so called “suits”) and is a game for up to 10 players. As in every poker game players bet on their hands, which are a combination of five cards. The player with the highest ranked combination wins. (Dalpasso and Lancia 2014) summarised possible hand combinations from best to worst (Figure 3.1).



Royal flush	[A♣]	[K♣]	[Q♣]	[J♣]	[T♣]	A, K, Q, J, T of the same suit
Straight flush	[T♡]	[9♡]	[8♡]	[7♡]	[6♡]	Five consecutive cards of the same suit
4 of a kind	[Q♡]	[Q♢]	[Q♠]	[Q♣]	[3♡]	Four cards of the same rank
Full house	[9♣]	[9♡]	[9♠]	[A♡]	[A♢]	Three cards of the same rank and a pair
Flush	[K♡]	[J♡]	[8♡]	[5♡]	[4♡]	Five cards of the same suit
Straight	[J♣]	[T♣]	[9♡]	[8♢]	[7♠]	Five consecutive cards
3 of a kind	[K♡]	[K♢]	[K♠]	[8♣]	[5♡]	Three cards of the same rank
Two pair	[J♡]	[J♢]	[8♠]	[8♢]	[2♡]	Two separate pairs
Pair	[Q♠]	[Q♢]	[K♡]	[9♢]	[6♣]	Two cards of the same rank
High card	[K♢]	[J♡]	[9♡]	[7♣]	[5♢]	Unrelated cards ranked by the highest card

Figure 3.1: Poker Hand Combinations Ranking (descending). Dalpasso & Lancia (2015)

In Texas Hold´em, every player gets two cards dealt at the beginning of the hand. Those so called “Hole Cards” belong to the player and cannot be seen by any other. Over the course of multiple betting rounds 5 cards are put onto the table from the deck, the so called “community cards”. Those cards belong to every player at the same time. The best hand is the best combination of 5 of the 7 cards. The community cards are revealed in 3 steps: the “flop”, which are the first 3 cards, the “turn” (the 4th card), and the “river” (the 5th card).

After getting the Hole Cards, and after each revealing of community cards, there is a betting round. The betting starts left of the dealer in a clockwise rotation. Every player who is still playing has three possible actions to choose from: Fold (getting out of the hand), bet/raise (bet if none bet before, or raise if it was bet before), check/call (match the previous bet/raise). A round of betting continues until every player has gone out of a hand, put in all of their chips, or matched the highest raise/bet put in by another active players.

Before the hole cards are dealt the two players left of the dealer have to place a forced commitment to the pot, the so called Blinds. The first player to the left of the dealer is placing the Small Blind, which is usually the half of what the player to his left has to place (the “Big Blind”). The size of the Blinds are giving before the play and denoted for example “1€/2€ No-Limit Texas Hold´em”.

Chapter 4

Essentials of Holecards

A regular poker deck consists of 52 cards, 13 cards with increasing values in 4 different colors, so called "suits". This leads to 52*51=2652 possible combinations in a mathematical sense. Since the order of the cards doesn't matter, this leaves us with 52*51/2=1326 combinations. Going along with the 1326 combinations is needed when considering post-flop action, but on a pre-flop state, some of these combinations are similar to each other.

In poker the color of the cards isn´t important when evaluating Holecards, the only differentiation which is important is whether or not the cards have the same color (they are "suited") or not (they are "off-suit"). e.g. imagine all possible hands on which one card is any king and the other card is any ace. There are 16 combinations, but only two in a poker-sense: Ace-King suited and Ace-King off-suit. Since color identity doesn´t matter in poker (colors are not ranked in any way), all possible combinations of two cards can be accounted for in this way:
Overall there are 13 pairs and 78 unique non-pair combinations of different values, we need to count them both for suited and offsuited, which leads to a total of 13*13=169 possible Holecards when only considering the pre-Flop state.

There are two conventions in poker literature to label Holecards. One is derived from the way you would identify a single card: The value of the card is symboled first, followed by a lowercase letter indicading the suit, e.g Ah for the Ace of Hearts, 7s for the seven of Spades. Conventionally pictured cards are displayed as JQKA and the 10 as an uppercase T. Displaying every card with Value and suit leads to Holecards in this format: $VsVs$, e.g. [Ah Ks], [7d 8d]. This format contains all information about the Holecards.

When talking about Holecards alone, without further look onto a board, it is common to label both values (descending order) and an indicating lowercase "s" if the cards are suited, a lowercase "o" if not or a blank when the cards are paired. VVs This format reduces all relevant information about Holecards to one of the 169 Holecards. [Ah Kd] would look like this: [AKo].

In order to analyse Holecards, we need to create a deck of cards from which we can draw cards.

##Data Engineering

```
deck <- data.frame(matrix(c(1:52,rep(c(2:14),4),rep(1:4,each=13)), ncol=3,
                          dimnames=list(NULL,c("Index","rank","suit"))))

getcards <- function(x){
  deck[deck$Index %in% x,]
}
```

The deck is represented in a 3-columns dataframe with a column for index, rank and suit.

Table 4.1: Deck of Cards (first and last 3 lines)

	Index	rank	suit
1	1	2	1
2	2	3	1
3	3	4	1
50	50	12	4
51	51	13	4
52	52	14	4

The function "getcards" allows one to get the corresponding rank and suit values of a given index. On this deck J, Q, K and A are considered 11, 12, 13 and 14 respectively.
The following chunk will create all possible hand combinations, but fails to get rid of duplicates right away, the problem will be solved later. Throughout this work we will use the `dplyr` package for data manipulation (Wickham et al. (2017)). We are using `this` font to indicate objects and functions. When talking about "the player" we will adress the player as masculine.

```
Starthands <- cbind.data.frame(
  Card1=rep(2:14,each=13),
  Card2=rep(2:14,13), #model all combinations of the 13*13 values, inkl pairs
  Suit="Offsuited",stringsAsFactors=FALSE) %>% #they are all offsuit
  bind_rows(
    cbind.data.frame(
      Card1=rep(2:14,each=13),
      Card2=rep(2:14,13),
      Suit="Suited",stringsAsFactors=FALSE) %>% #generate the suited hands
      filter(Card1!=Card2) #filter out pairs
  )
```

Table 4.2: Holecard Combinations (first 6 lines)

Card1	Card2	Suit
2	2	Offsuited
2	3	Offsuited
2	4	Offsuited
2	5	Offsuited
2	6	Offsuited
2	7	Offsuited

```
Starthands <- Starthands %>% mutate(CardId1=Card1-1,
                                    CardId2=ifelse(Suit=="Offsuited",Card2-1+13,Card2-1))
for(newcolumn in paste(1:9, sep="")){#add 9 new columns to save win rates
  Starthands[,newcolumn]<-0
}
```

Now that the Holecards and the deck are set up, we need a way to evaluate the strength of a specific holding. The following function gives exactly this information. `check.winrate` takes two arguments: a vector of two hand-indexes (based on the deck-data.frame) and the number of opponents on the table. The function is relatively straight forward: it deals every opponent two cards, reveals the 5 community cards and evaluates the winner of the hand by utilising the `holdem::handeval` function (Frederic Paik Schoenberg 2017). It will repeat this process 1000 times for demonstration purposes. In order to get the data, with which we will continue the work, we evaluated each starting hand 1.8 million times.

```r
check.winrate <- function(hand,players){
  set.seed(42)
  hand <- getcards(hand) #get rank and suit information from the index´s

  res <- array() #initialising result array to store data

  for(i in 1:1000){ #start loop, 1000 iterations
    board <- getcards(sample(deck$Index[!deck$Index %in% hand$Index],5,replace = F))
    #get board cards, which aren´t dealt as Holecards
    myhandandboard <- rbind(hand, board) #the players 7 cards
    allcards <- myhandandboard #all used cards so far
    otherhands <- list() #initiating an emtry list to store the opponent cards

    for(p in 1:players){ #for every opponent
      otherhands[[p]] <- getcards(sample(
        deck$Index[!deck$Index %in% allcards$Index],2,replace = F))
      #draw two cards from the deck, which havn´t been drawn before
      allcards <- rbind(allcards,otherhands[[p]])
      #add those cards to the array of drawn cards
    }

    scores <- array() #initialising a array to store the scores of every hand

    for(p in 1:players){
      opphandandboard <- rbind(otherhands[[p]],board)
      #combine hand and board cards for every opponent
      scores[p] <- holdem::handeval(opphandandboard$rank,opphandandboard$suit)
      #using handval to get a numeric value indicating the strength of the hand
    }

    res[i] <- holdem::handeval(myhandandboard$rank,myhandandboard$suit) > max(scores)
    #check if the players score is higher than the maximum of opponent scores
  }
  m <- mean(res) #generate the winrate

  return(m)
}
```

To demonstrate the function, here it will run with two aces in a heads-up, 1 versus 1 situation:

```r
set.seed(42)
check.winrate(c(13,26),1)
```

```
## [1] 0.841
```

Since two Aces are considered the best possible starting hand, the winrate of 84.1% isn´t surprising. This rather simple method is quite handy to get a first impression. In reality opposing Holecards are not random, because one can decide not to play a bad hand. Additionally only a minority of hands will be played until the showdown (on which hands are evaluated), most rounds of play end without ever showing Holecards. It is also quite rare to have all players on a table on a hand. On the next section we will use regression to get further insights on what makes Holecards successful.

The `winrates.rmd` in the github repository (Philipp (2018)) uses parallel processing (Analytics and Weston (2015);@foreach) to calculate an approximated winrate for every starting hand against 1 up to 9 opponents and creates a data.frame called `Starthands`. A total of 304.2 million hands have been generated to get the results for the 169 possible Holecards.

4.1 Feature Engineering on Holecards

At this point we are loading the `Starthands` data.frame.

```r
values <- c(2:9,"T","J","Q","K","A") #commonly used card names, starting with 2 up to Ace

Starthands.Pair.Desc <-Starthands %>%
  select(1:5) %>% # Card1 &2, CardId1 &2, Suit
  mutate(Index=1:325,
         Pair=ifelse(Card1==Card2,1,0), # 1 if a hand is a pair
         Symbol1=values[Card1-1],
         Symbol2=values[Card2-1],
         Description=ifelse(Card2>Card1, # adding VVs pattern to the DF
                            paste(Symbol2,Symbol1,
                                  ifelse(Pair==1,"",
                                         ifelse(Suit=="Suited","s","o")),sep=""),
                            paste(Symbol1,Symbol2,
                                  ifelse(Pair==1,"",
                                         ifelse(Suit=="Suited","s","o")),sep=""))) %>%
  bind_cols(Starthands[,6:14]) %>%
  group_by(Description) %>%
  filter(Index==max(Index)) %>%
  #get rid of duplicates, leaving the last observation of a hand in the DF
  select(-Index) %>%
  ungroup() %>%
  mutate(Connectors=ifelse(abs(Card1-Card2)<=4 & abs(Card1-Card2)>0 |
                             Card1==14 & Card2==2|
                             Card1==14 & Card2==3|
                             Card1==14 & Card2==4|
                             Card1==14 & Card2==5,1,0)) # 1 if a hand is a connector hand
```

Table 4.3: Holecards with Features (first 6 lines)

Description	Card1	Card2	Suit	Pair	Connectors	1
22	2	2	Offsuited	1	0	0.495055
32o	3	2	Offsuited	0	1	0.292580
33	3	3	Offsuited	1	0	0.527740
42o	4	2	Offsuited	0	1	0.301980
43o	4	3	Offsuited	0	1	0.320885
44	4	4	Offsuited	1	0	0.561370

On this feature engineering chunk we got rid of the duplicated Holecards in the `Starthand` data.frame and created some new columns:

- `Pair` is 1 if the hand is a pair, 0 otherwise.
- `Symbol1 & Symbol2` the Symbol (2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A) of the cards respectively.
- `Description` the VVs way to name Holecards.
- The columns labeled with the numbers 1 to 0 contain winrates of the Hoelcards againat 1 up to 9 opponents.
- `Connectors` is 1 if the cards are within +/- 4 Cards away from each other, leading to a straight

possibility, 0 if they are too far apart.

4.2 Regression on Holecards

In this part we will use our generated data to perform multiple analysis on it. First we set up the data in a way that every row contains details about the Holecards, the number of opponents and the winrate against this specific number of opponents (we melt the data.frame with the opponent-specific winrate as the measuring variable utilizing the `reshape2` package (Wickham (2007))). This leads to $169 * 9 = 1521$ rows, saved in a new data.frame called DF.

```
DF <- Starthands.Pair.Desc %>%
  ungroup() %>%
  select(Description,Card1,Card2,Suit,Pair,Connectors,10:18) %>%
  as.data.frame() %>%
  melt(measure.vars = c(7:15),variable.name="Other.Players",value.name = "Winchance") %>%
  as.data.frame.array() %>%
  mutate(Pair=as.factor(Pair),
         Connectors=as.factor(Connectors),
         Suit=as.factor(Suit),
         Other.Players=as.numeric(Other.Players))
```

Now the data is prepared and a simple linear regression can be built in order to identify what makes a hand strong, and what has little or no effect. What we would expect is that two high value cards are important as well as that those cards are suited.

```
fit.lm <- lm(Winchance ~.,data=DF[,-1])
summary(fit.lm)
```

```
##
## Call:
## lm(formula = Winchance ~ ., data = DF[, -1])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.09494 -0.04571 -0.01665  0.03444  0.23968
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.2294354  0.0078867   29.09  < 2e-16 ***
## Card1          0.0096128  0.0007663   12.54  < 2e-16 ***
## Card2          0.0093246  0.0007792   11.97  < 2e-16 ***
## SuitSuited     0.0376978  0.0031923   11.81  < 2e-16 ***
## Pair1          0.1572757  0.0080025   19.65  < 2e-16 ***
## Connectors1    0.0199434  0.0042612    4.68 3.12e-06 ***
## Other.Players -0.0419415  0.0005939  -70.62  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05981 on 1514 degrees of freedom
## Multiple R-squared:  0.8161, Adjusted R-squared:  0.8154
## F-statistic:  1120 on 6 and 1514 DF,  p-value: < 2.2e-16
```

First let us adress the quality of the model, otherwise all further conclusions are useless. The F statistic is highly significant, which means that the model is in fact able to represent reality. The Adjusted R-squared can be interpreted as the percentage of total variance explained by the observed features (81.54 %). We can also see that every feature is important to the model (low P-Values). Now that we know that the model is trustworthy, we can check out which story it is telling us.

4.3 Interpretation of the Linear Regression

We created this first model with an intercept, which can be interpreted as a base-winrate we add or substract from. The baseline winrate is valid for offsuited cards, which are not connected or paired.
High cards leading to a higher winrate. Remember that our cards start at value 2 up to 14 for an ace. The cards have been ordered in a way that Card1 is always the higher of the two cards. We can see that an increase in value of the higher card is leading to an increase of 0.96% in winrate, while the lower card only increases the winrate by 0.93%. A interpretation could be that high cards are valuable and there is evidence that higher cards can decide games more often than lower cards. This corresponds with the fact that there is a 1.99% winrate increment when cards are close to each other.
Paired cards seem reasonably strong, leading to 15.73% increase in winrate. This underlines the fact that the so called “pocket pairs” are in fact already “made hands” and can win a hand even without further improving through community cards. If cards are suited, the winrate goes up by 3.77%, which corresponds to the higher chance of getting a flush. Every opponent in the hand reduces the winchance by -4.2%.

Overall this result further emphasizes the common understanding of Holecards from a pseudo-empirical view. Based on the insights from the model, one can state that the best hand is a pair of Aces in a 1v1 and the worst hand are the two lowest, unconnected cards in a 10-way hand, which will be 72o, Seven & Deuce offsuit. We can verify this from our data as well:

Table 4.4: Worst and Best Holecards

	Description	Card1	Card2	Suit	Pair	Connectors	Other.Players	Winchance
91	AA	14	14	Offsuited	1	0	1	0.849575
1368	72o	7	2	Offsuited	0	0	9	0.040460

The quality of our model can be checked by comparing its residuals with the fitted values using the ggplot package (Wickham (2009)). For a perfect model, all those points would form a horizontal line with 0 residual. Yet we can observe two things:

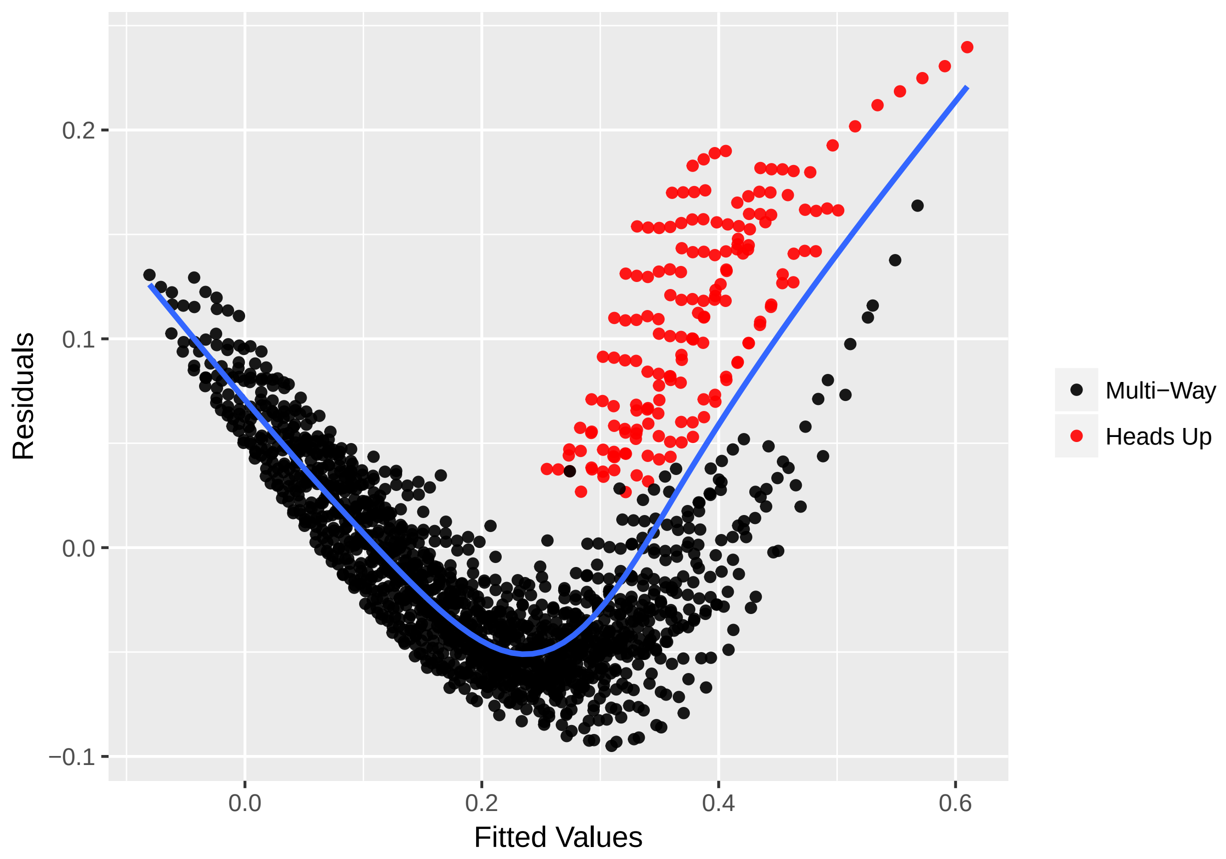


Figure 4.1: Predicted Winrate versus Residuals for a Linear Regression Model

First, we can see a non-linearity. Our fitted values tend to underestimate the winrate when forecasting low and high winrates and overestimate when predicting mediocre winrates. From this non-linearity we can conclude one of two possible conditions: There could be an unobserved variable or one or more of the variables behave in a non linear fashion. Secondly, and more important, there is a cloud of underestimated values. Exploration revealed that those residuals are the group of hands versus a single opponent, “heads up”.

The examination of the residual plots suggests that it may be appropriate to build two seperate models, one for 1v1 situations and one for all others, in order to get a better fit on the data.
Here is a model without heads-up hands:

```
fit.lm.2andmore <- lm(Winchance ~.,data=DF[DF$Other.Players!=1,-1])
summary(fit.lm.2andmore)
```

```
##
## Call:
## lm(formula = Winchance ~ ., data = DF[DF$Other.Players != 1,
##     -1])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.069503 -0.024083 -0.007183  0.017509  0.239165
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.1695400  0.0050786  33.383  < 2e-16 ***
## Card1          0.0078779  0.0004757  16.562  < 2e-16 ***
## Card2          0.0092248  0.0004837  19.070  < 2e-16 ***
## SuitSuited     0.0389672  0.0019817  19.664  < 2e-16 ***
## Pair1          0.1438665  0.0049677  28.961  < 2e-16 ***
## Connectors1    0.0197798  0.0026452   7.478 1.36e-13 ***
## Other.Players -0.0301470  0.0004155 -72.562  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.035 on 1345 degrees of freedom
## Multiple R-squared:  0.8714, Adjusted R-squared:  0.8708
## F-statistic:  1519 on 6 and 1345 DF,  p-value: < 2.2e-16
```

In the next model we included only heads-up hands:

```
fit.lm.headsup <- lm(Winchance ~.,data=DF[DF$Other.Players==1,-1])
summary(fit.lm.headsup)
```

```
##
## Call:
## lm(formula = Winchance ~ ., data = DF[DF$Other.Players == 1,
##     -1])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.033287 -0.009716 -0.000360  0.009533  0.043721
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1477023  0.0053358  27.682  < 2e-16 ***
## Card1         0.0234923  0.0005596  41.980  < 2e-16 ***
## Card2         0.0101228  0.0005691  17.788  < 2e-16 ***
## SuitSuited    0.0275424  0.0023313  11.814  < 2e-16 ***
## Pair1         0.2645492  0.0058442  45.267  < 2e-16 ***
## Connectors1   0.0212518  0.0031120   6.829 1.61e-10 ***
## Other.Players        NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01456 on 163 degrees of freedom
## Multiple R-squared:  0.9824, Adjusted R-squared:  0.9818
## F-statistic:  1816 on 5 and 163 DF,  p-value: < 2.2e-16
```

We can see a reasonable increase in Adjusted R squared, which means that there is a big increase in quality by deviding both models, which induces the assumption that the hand requirements in 1v1 versus other situations are quite different.

First of all, the overall interpretation stays intact, but there are three major differences:

- On multiplayer tables the lower card has more influence than the higher card, this could be explained by the fact that if there is a tie between players. e.g say both have a pair of Aces, the one with the next higher card wins. This is called the "Kicker". We can see its impact on the data.
- In 1v1 situations the second card rarely ever matters, since the higher card is winning the hand more often. This comes from the fact that in 1v1s most of the time the players will miss the board (the

highest card wins), or hit the board and win with the higher pair. Since there is less possibility for selecting your hand in 1v1s, higher cards are more impactful.

- As mentioned, players will miss the board a lot and a pocket pair is very strong in those situations, because it doesn´t have to improve.

Here are the residual plots for the two new models:

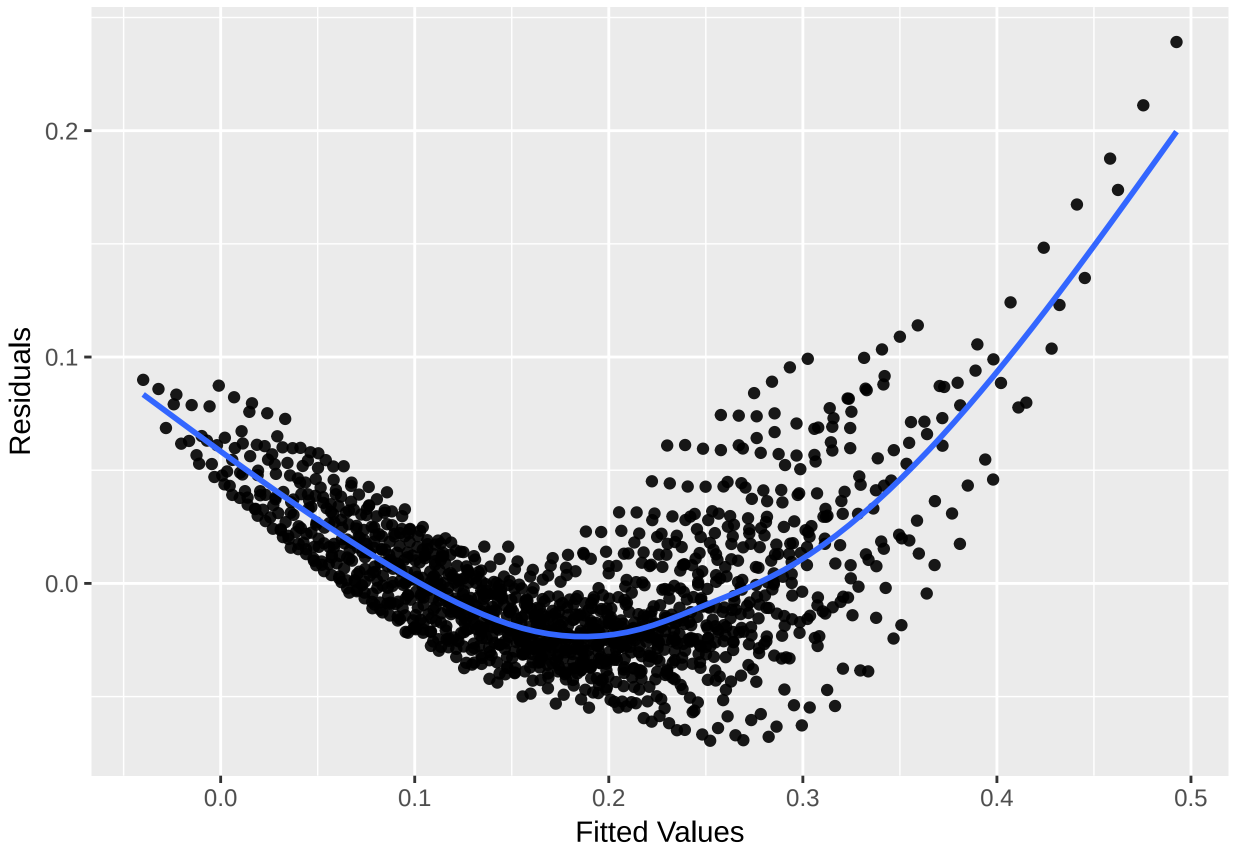
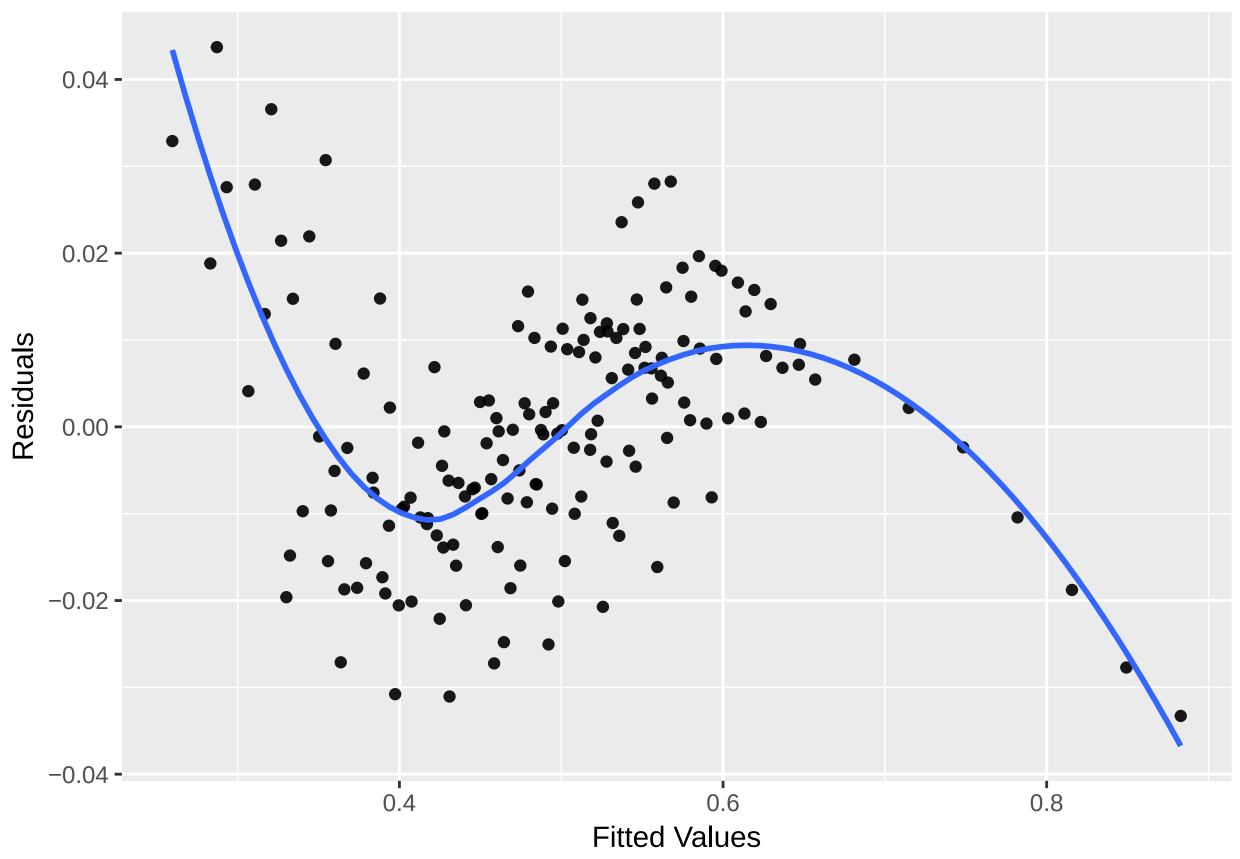


Figure 4.2: Predicted Winrate versus Residuals for a Linear Regression Model on Multiplayer Tables



In this chapter we derived winrates pseudo-empirically, created features and analysed the data by fitting an easy to understand linear model with the purpose to get common knowledge about Holecards. It was not the intension of this chapter to model perfectly, otherwise we would have used a logistic algorithm. In the following chapter, we continue our exploratory analysis on Holecards, this time on actual data.

Chapter 5

Logdata Engineering

This chapter will introduce the reader to the logdata we are using, as well as the code which is used to transform the logs into a data.frame structure. The code used can be found in the github repository (Philipp (2018)) as `DataEngineeringServer.rmd`.

```
PokerStars Hand #XXXXXX: Tournament #XXXXXX, $0.44+$0.06 USD Hold'em No Limit - Level I
(10/20) - yyyy/mm/dd 0:11:55 CET [yyyy/mm/dd 18:11:55 ET]
Table 'XXXXXX 1' 9-max Seat #2 is the button
Seat 1: Zero (1500 in chips)
Seat 2: Player_2 (1370 in chips)
Seat 3: Player_3 (1480 in chips)
Seat 4: Player_4 (1500 in chips)
Seat 5: Player_5 (1500 in chips)
Seat 6: Player_6 (1650 in chips)
Seat 7: Player_7 (1500 in chips)
Seat 8: Player_8 (1500 in chips)
Seat 9: Player_9 (1500 in chips)
Player_3: posts small blind 10
Player_4: posts big blind 20
*** HOLE CARDS ***
Dealt to Zero [2h 5s]
Player_5: folds
Player_6: folds
Player_7: folds
Player_8: folds
Player_9: folds
Zero: folds
Player_2: calls 20
Player_3: folds
Player_4: checks
*** FLOP *** [4d Kc Kd]
Player_4: checks
Player_2: checks
*** TURN *** [4d Kc Kd] [Ac]
Player_4: bets 40
Player_2: calls 40
*** RIVER *** [4d Kc Kd Ac] [4c]
Player_4: checks
Player_2: bets 200
```

```
Player_4: raises 400 to 600
Player_2: folds
Uncalled bet (400) returned to Player_4
Player_4 collected 530 from pot
Player_4: doesn't show hand
*** SUMMARY ***
Total pot 530 | Rake 0
Board [4d Kc Kd Ac 4c]
Seat 1: Zero folded before Flop (didn't bet)
Seat 2: Player_2 (button) folded on the River
Seat 3: Player_3 (small blind) folded before Flop
Seat 4: Player_4 (big blind) collected (530)
Seat 5: Player_5 folded before Flop (didn't bet)
Seat 6: Player_6 folded before Flop (didn't bet)
Seat 7: Player_7 folded before Flop (didn't bet)
Seat 8: Player_8 folded before Flop (didn't bet)
Seat 9: Player_9 folded before Flop (didn't bet)
```

As we can see in this anonymised example log, there is a lot of data to extract. To explain all of the 480 lines of our data engineering code would not be consistent with this thesis, yet we want to look at some important lines to create a better understanding of what data is available.

We will create two data.frames on which we can save data. `Handstats` is a data.frame which contains information about players in a hand granularity. Each row represents a player (PLID), the seat, the current amount of money (Stack), if the player placed a Blind and the Blindsize of the table. Additionally we have information about the hand-ID (PSID or GameID), the gametype (e.g. "0.44+$0.06 USD Hold'em No Limit"), the date (ET), tournament ID and tournament level.

```r
    for(k in 1:NoPlayers){
      PLID[k] <- sub("(Seat|Platz) .: (.*?) \\(.*", "\\2", y[[i]][k+2])
      Seat[k] <-  sub(".*? (.*?): .* in [Cc]hips.*", "\\1", y[[i]][k+2])

      Stack[k] <- sub(".*?: .*\\((\\$)?(.*?) in [Cc]hips.*", "\\2", y[[i]][k+2])

      Blind[k] <- ifelse(sum(grepl(paste(quotemeta(PLID[k]),
                                        ": .* [Ss]mall [Bb]lind(:)? ",sep=""),
                                  y[[i]]))>=1,"SB",
                        ifelse(sum(grepl(paste(quotemeta(PLID[k]),
                                              ": .* [Bb]ig [Bb]lind(:)? ",sep=""),
                                        y[[i]]))>=1,"BB",""))
      Blind.size[k] <- ifelse(Blind[k]=="SB",SB.amount,BB.amount)
    }

PSID <- sub(".*?PokerStars.+Hand (#)?(.*?): .*", "\\2", y[[i]])[1]
GameType <- sub(".*?: (.*?) - .*", "\\1", y[i])[1]
Date <- sub(".*( |\\[)((\\d|\\.|/)+ (\\d|\\:)+ ET).*","\\2",y[[i]][1])
TourID <- ""
    if(grepl("(Tournament|Turnier)",GameType)){
      TourID <- sub("(Tournament|Turnier) #([0-9]+), .*","\\2",GameType)
    }
    if(!grepl("(Tournament|Turnier)",GameType)){
      TourID <- sub("(Table|Tisch) \\'(.*?)\\' .*","\\2",y[[i]][2])
    }
tourLvl <- ""
    # Blinds Amount
    if(!grepl("(Tournament|Turnier)",GameType)){
      SB.amount <- as.numeric(sub(".*?((\\d|\\.)+)/.*", "\\1", GameType))
      BB.amount <- as.numeric(sub(".*?/(\\$)?((\\d|\\.)+)( .*)?\\)", "\\2", GameType))
    }
    if(grepl("(Tournament|Turnier)",GameType)){
      SB.amount <- sub(".* Level .* \\((.*?)/.*","\\1",y[[i]][1])
      BB.amount <- sub(".* Level .*/(.*?)\\).*","\\1",y[[i]][1])
      tourLvl <- sub(".*Level (\\w+) .*","\\1",y[[i]][1])
    }
```

`Logdata` is a data.frame with a single action granularity. Each row represents an action taken by a player in a given hand.

`Logdata` consist of an action such as “raises $0.05 to $0.07” and the corresponding action category “raise”. There are additional columns to save the currently highest overall bet (maxbet/betsize), the number of players remaining in the hand, the number of players on the table, the relative position, the Board, the winner of the hand and any cards which are shown during the game.

```
t <- sub("^(.*): (.*)", "\\2", prefloplines[k])
action.raw <- sub(" .*", "", substr(t,0,nchar(t)-1))
#extracting the first word of the action for easier matching
action <- str_trim(substr(t,0,nchar(t)-1),"right")
#keeping all relevant information
action.type <- get.action.type(action.raw)
betsize <- set.betsize(action,action.type)
PLremain <- PLremain - as.numeric(action.type=="fold")
relPos <- get.rel.position()
winner <- get.winner(y[[i]])
won <- ifelse(PLID==winner,1,0)
data.frame(action,action.type,betsize,PlayerID=PLID,
           PLremain,relPos,Round,maxbet,PSID,GameType,
           NoPlayers,prevBet,Pot,Board="Preflop",
           winner=won,shown.hand=hand,
           stringsAsFactors=FALSE)
```

We had to repeat some of the steps on multiple rounds (pre-flop, flop, turn and river), therefore it was easier to set up functions instead of writing the code multiple times. The interested reader is advised to take a look at the `DataEngineeringServer.rmd` to fully understand the coding work behind the transformation (Philipp (2018)). On the other hand, it is not required to understand how the data is transformed to follow the thesis.

Chapter 6

Analyzing Holecards from Logdata

For analysis purposes 71,000 poker hands have been collected between January 2016 and November 2017. Those hands have been played on pokerstars.com, the biggest online poker platform. In a first step, the data will be read in, cleaned of unwanted lines (such as player talking in chat) and converted into a list structure.

6.1 Read in Logdata

The logs are saved in multiple folders: year/day/session. The chunk above identifies all logs inside the mainfolders subfolders and reads them in one after another.

```r
setwd(parent.folder)

files <- character()
files.inner <- character()

  file.name.v <- list.files(parent.folder, pattern=".*\\.txt")

  for (f in file.name.v){
    file.read.v <- scan(paste(parent.folder, f, sep="/"),
                        what ="character",sep="\n",quiet = TRUE)
    files.inner <- c(files.inner,file.read.v)
  }
  files <- c(files,files.inner)

rm(f,file.name.v,file.read.v,files.inner,parent.folder)
setwd(github.folder)
```

The output is a large character with more than 3.2 million elements (lines of text). These lines have to be cleaned to not contain non-poker related information, such as connection status, time outs and players chat.

```r
  x <- files
  keywords <- c("setzt sich auf", "verlässt den Tisch","verl?sst den Tisch", "Time Out",
                "has timed out", "sagte","ist verbunden","ist nicht verbunden",
                "wurde von Tisch","setzt aus","joins the table",
                "has returned","connected","disconnect","leaves the"," said",
                "is sitting out","will be allowed","ist zurück", "ist zur?ck",
                "was removed","wurde entfernt")
  buzzwords <- paste(keywords, collapse = "|")
  x <- x[!grepl(buzzwords, x)]
```

At this point we still have one large character file, which we need to split to get a hand-by-hand list. The data can be cut at lines which include “PokerStars Hand” or close related terms depending on the language of the logs aswell as the game mode.

```
 cuts = grep(paste(c("PokerStars Hand ","PokerStars Zoom Hand ",
                     "PokerStars Zoom-Hand ",
                     "PokerStars Home Game Hand "),collapse="|"), x)
  y = list()
  for (i in 2:length(cuts)){
    j = cuts[i-1]:(cuts[i]-1)
    y[[i]] = x[j]
  }
  y <- y[-1]
rm(files,x,i,j,buzzwords,cuts,keywords)
```

Now we have a list containing 85353 hand logs. At the moment we are only interested in the Holecards of each hand. So we create a data.frame with the number of rows equal to the amount of hand logs we have to fill in the data.

```
Holecards <- as.data.frame(
  matrix("",ncol=3,nrow=length(y)),
    stringsAsFactors =FALSE)

colnames(Holecards) <- c("Index","Holecards","GameID")
```

Afterwards we can extract the Holecards by using Regular Expressions combined with the `stringr`package (Wickham (2017)) and the `qdapRegex` package (Rinker (2017)). Again, the Regex are written in a way that they can process German as well as English logs.

```
for(i in 1:length(y)){
  Holecards[i,1] <- as.character(i)
  Holecards[i,2] <- sub(".*\\[(.*)\\]","\\1",y[[i]][which(grepl(".*HOLE CARDS.*",y[[i]]))+1])
  Holecards[i,3] <- sub(".*?PokerStars.+Hand (#)?(.*?): .*", "\\2", y[[i]])[1]
}
```

6.2 Feature Engineering on real Holecards

An important information seems to be if the Holecards were played or not. To get this information we can manipulate the `Logdata`. A hand was played when the player has not folded the Holecards during the pre-flop phase.

Our data comes from two observed players (Hero and Zero). We don´t want the data to be biased by the fact that both players are on the same table, so we filter those hands out. The object `remove` contains the corresponding GameIDs.

```
Holecards <- Holecards %>% filter(!GameID %in% remove)
Logdata <- Logdata %>% filter(!PSID %in% remove)
Handstats <- Handstats%>% filter(!GameID %in% remove)
```

Now we can extract the required information (played or folded) from `Logdata` and join it with our `Holecards` data.frame.

```
PlayedHands <- Logdata %>%
  filter(Round=="Pre-Flop" & PlayerID %in% c("Hero","Zero")) %>% #Flop actions only
  group_by(PSID) %>% #by each GameID
  summarise(played=ifelse("fold" %in% action.type,0,1), #0 if folded, 1 if not
            NoPlayers=max(NoPlayers),
            PlayerID=max(PlayerID)) %>%
  rename(GameID=PSID) #rename PSID to GameID

Holecards <- Holecards %>% left_join(PlayedHands) #left join the new feature
```

Table 6.1: Played Holecards Extracted from the Logs (first 6 lines)

Index	Holecards	GameID	played	NoPlayers	PlayerID
1	Js 6d	146669522709	0	6	Hero
2	8h Td	146669552852	0	6	Hero
3	7d Qc	146669566568	0	6	Hero
4	7s Jd	146669578634	1	6	Hero
5	4c 3d	146669596721	0	5	Hero
6	2h 9d	146669604605	0	5	Hero

As you can see, the Holecards from the logs are written in the Vs Vs pattern, which makes sense, if we want to keep all information available. For this thesis and its purpose it is needed to transform the pattern into the VVs pattern, which fits our previous data. To transform any possible two cards from the Vs Vs scheme towards VVs, we wrote a function which first takes the string and extracts all letters and digits one by one. In a second step it rearrange them in a way that the higher value cards is named first. Afterward it is checked if the cards are suited or not and the lowercase s|o will be added accordingly.

```
extract.hand.class <- function(hand){
  #extract all symbols
  n1 <- substr(hand,1,1)
  n2 <- substr(hand,4,4)
  c1 <- substr(hand,2,2)
  c2 <- substr(hand,5,5)

  if(n1 %in% 1:9==TRUE |  n2 %in% 1:9==TRUE){ #if cards are numbers paste max and min
    cards <- paste(max(n1,n2),
                   min(n1,n2),sep="")}
  else{

    if("J" %in% c(n1,n2) & "K" %in% c(n1,n2)){ #if a hand is KJ paste max min
    cards <- paste(max(n1,n2),
                   min(n1,n2),sep="")
    } else{

      if("Q" %in% c(n1,n2) & "J" %in% c(n1,n2)){ #if a hand is QJ paste max min
      cards <- paste(max(n1,n2),
                     min(n1,n2),sep="")
      }else{
      cards <- paste(min(n1,n2),
#all other cases paste min max (to compensate for the alphabetical order)
                 max(n1,n2),sep="")
            }
          }
        }
  class <- paste(cards,
                 ifelse(n1==n2,"", #if values are the same, add an empty string
                        ifelse(c1==c2,"s","o")),sep="")
  #else add s when suits are equal, and o if not
  return(class)
}
```

```
tmp <- c("Qh Kh","As 9d","Js Jc","6h Ac", "2d Qd")
sapply(tmp,FUN="extract.hand.class")
```

```
## Qh Kh As 9d Js Jc 6h Ac 2d Qd
## "KQs" "A9o"  "JJ" "A6o" "Q2s"
```

As you can see in the examples above, the function works fine, even if the initial cards are not in the requested order.

6.3 Winrate Ranking

To further deepen our knowledge about winrates of Holecards, we will create a measurement of quality, so that we can say a player plays only “the top 15% of hands”. In poker literature, players are categorized by the percentage of hands they play (Sklansky and Malmuth (1999)). The less hands they play, the “tighter” the “Range” of cards they play. And vice versa: The more they play, the “looser” the Range of cards they play is.
So an assumption is, that if a player plays 20% of their hands, this is corresponding to the best 20% of the Holecards (according to the winrate ranking of Holecards). If he plays 30%, his Range is the top 30% of Holecards and so on.

Setting an opponent on a Range of cards he would play helps to identify his strength in a given situation. So if the assumption proves true, it may help to find a better understanding of your opponent and your own play. This is valid under the condition that the observed player knows fundamental poker mindsets (e.g. is not playing random hands). In the following chapters we will provide evidence that playrate alone is a misleading indicator of a players Range.
To illustrate this, we will take the winrates on a 10 player hand, because it represents winrates under the worst conditions of play. What can be identified as a misconception is the assumption that every hand contributes $1/169 = 0.6\%$ of probability. Someone may misinterprete “the top 20% of hands” as the top $20\% * 169 = 34$ hands. What is not thought about is that the probabilities of getting hands are quite different for some types of hands: The probability of getting a specific pair is $\frac{4}{52} * \frac{3}{51} = 0.45\%$ while the chance of getting a specific offsuited non-pair is $\frac{8}{52} * \frac{3}{51} = 0.90\%$ and a specific suited non-pair $\frac{8}{52} * \frac{1}{51} = 0.3\%$.

So what needs to be done is to get a new column with the probabilities to get the specific hand and a cumulative sum of those to indicate the “top 20%” not as an absolute number of 20% of 169 but as the “top Holecards you get 20% of the time”.

```
winrates <- Starthands.Pair.Desc[,c(9,3,6,18)]
colnames(winrates) <- c("Symbol","Suit","Pair","Rate")
winrates <- winrates %>%
  mutate(chance=ifelse(Pair==1,(4/52)*(3/51),
         ifelse(Suit=="Offsuited",(8/52)*(3/51),(8/52)*(1/51))),
         Rate=as.numeric(Rate)) %>%
  arrange(-Rate) %>%
  mutate(percentcumsum=cumsum(chance))
```

Table 6.2: Holecards, Winrates and Cumulated Chance (first 6 lines)

Symbol	Suit	Pair	Rate	chance	percentcumsum
AA	Offsuited	1	0.31093	0.0045249	0.0045249
KK	Offsuited	1	0.25882	0.0045249	0.0090498
QQ	Offsuited	1	0.21949	0.0045249	0.0135747
AKs	Suited	0	0.19999	0.0030166	0.0165913
JJ	Offsuited	1	0.19029	0.0045249	0.0211161
AQs	Suited	0	0.18491	0.0030166	0.0241327

Now when we check the top 20% based on the cumulative sum of probabilities we will find out that we are in fact talking about the top 45 hands instead of the top 34. Which is indicading a much looser Range than expected.

```
Holecards <- cbind(Holecards,Symbol=sapply(Holecards$Holecards,
                                          FUN="extract.hand.class"))

Holecards <- Holecards %>% left_join(winrates %>% select(Symbol,percentcumsum),
                                    by="Symbol")

Holecards <- Holecards %>% mutate(high=substr(Symbol,1,1),
                                  kicker=substr(Symbol,2,2),
                                  suit=substr(Symbol,3,3))
```

There are 13668 NAs, which all came from 1v1 hands on which the opponent folds the hand before our player has to act the first time, leading to no entry in the gamelog. Since we have no information if the player intended to play the hand, we are going to omit those lines.

```
Holecards <- na.omit(Holecards)
```

6.4 Hand Distribution

In this thesis the reader will frequently encounter heatmaps like the following. It is quite important to understand how to read it in poker context.

```
tmp <- Holecards %>%
  select(GameID,played,high,kicker,Symbol,suit,percentcumsum,PlayerID) %>%
  group_by(PlayerID,Symbol) %>%
  summarise(Playrate=mean(played),
            highcard=ifelse(max(suit)=="o",max(high),max(kicker)),
            kickercard=ifelse(max(suit)=="o",max(kicker),max(high))
            )%>%
  ungroup() %>%
  mutate(highcard=factor(highcard,levels=values),
                     kickercard=factor(kickercard,levels=values))
```

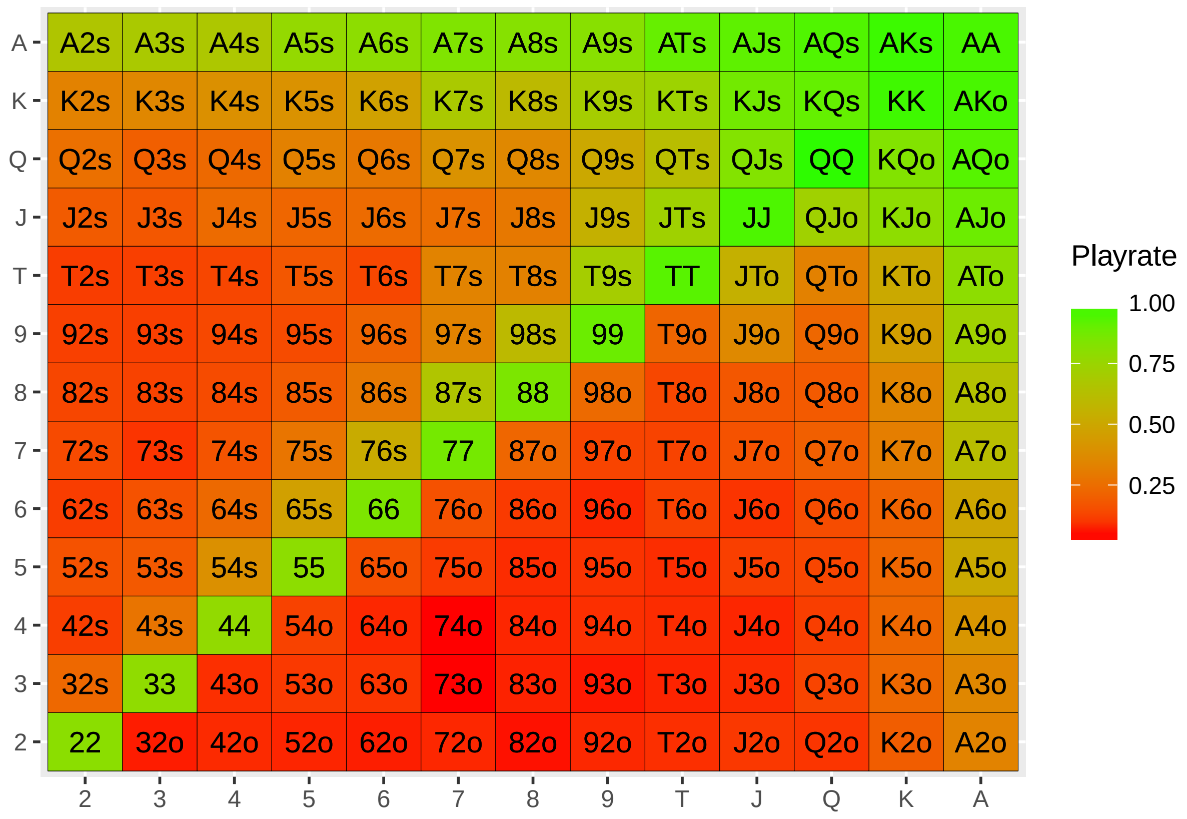


Figure 6.1: Unsubsetted Heatmap of Holecards Playrate Distribution

On this heatmap we can see every possible poker starting hand. On X and on Y axis we have the values of poker cards. Every tile represents the combination of those values. Paired values are building the diagonal of the heatmap. In the upper triangle we display suited card combinations. In the lower triangle we display offsuited combinations.
The gradient scaled fill of the tiles represents the average playrate of the Holecard, from green (100%) to dark red (0%). We observe arrow-shaped patterns (pointing to the upper right) which are more dense on the diagonal, on aces and on the right-top corner with a slightly more dense upper triangle. When facetting the heatmap for different subsets to see differences we will drop the text overlay to guarantee readability.

```r
tmp <- Holecards %>%
  select(GameID,played,high,kicker,Symbol,suit,percentcumsum, NoPlayers,PlayerID) %>%
  group_by(GameID) %>%
  mutate(NoPlayers=ifelse(max(NoPlayers)<=4,
                          "short-handed",ifelse(max(NoPlayers)>=7,
                                                "long-handed","mid-handed"))) %>%
  ungroup() %>%
  group_by(PlayerID,Symbol,NoPlayers) %>%
  summarise(Playrate=mean(played),
            highcard=ifelse(max(suit)=="o",max(high),max(kicker)),
            kickercard=ifelse(max(suit)=="o",max(kicker),max(high))
            )%>%
  ungroup() %>%
mutate(highcard=factor(highcard,levels=values),
                         kickercard=factor(kickercard,levels=values))
```

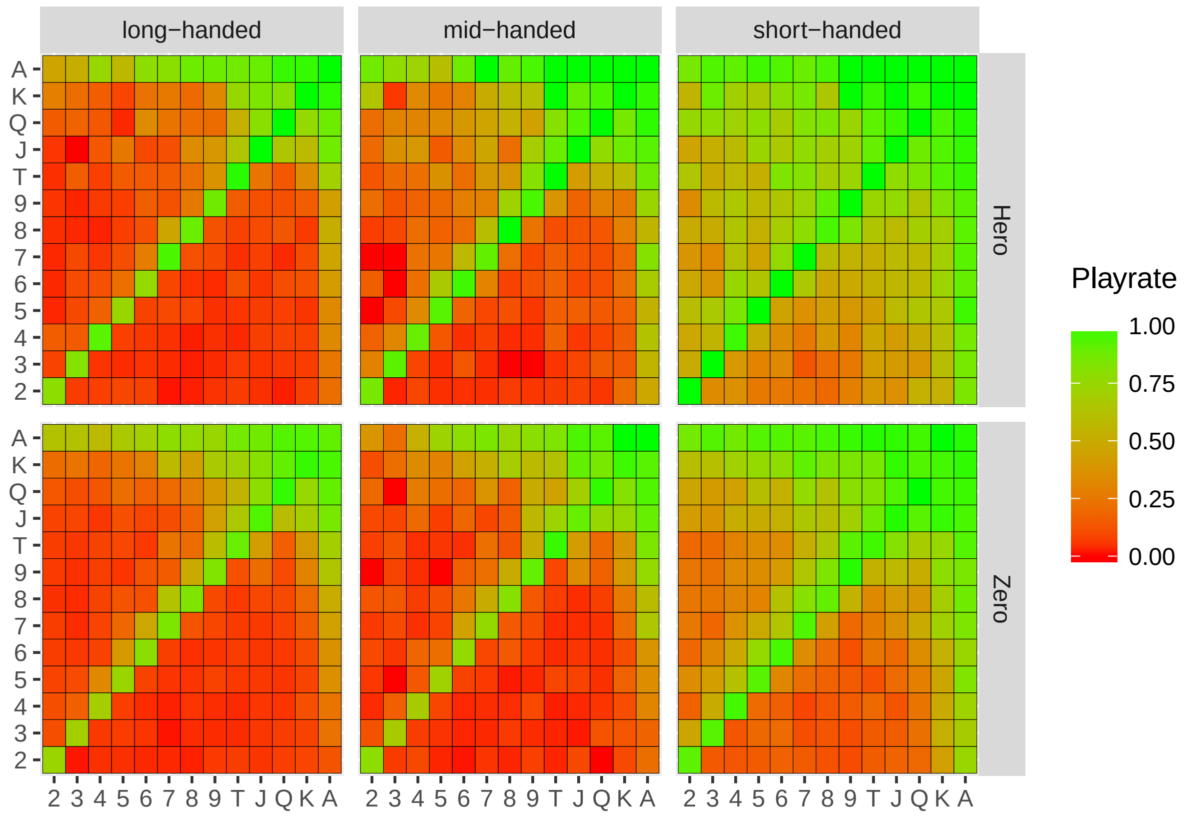


Figure 6.2: Playrate Distribution Heatmap by Tablesize and Player

Also we can observe that when there are less players on the table (the table is “short handed”), more Holecards are getting played spreading out from the right-top corner, as well as Holecards right above the diagonal, which are the so called “suited connectors”, hands which can lead to straights as well as flushes.

Chapter 7

The Observed Playrate versus the Range of a Player

As we can see, there are visual differences when it comes to Holecard selection in dependency on the number of players on the table. This comes from the fact that on a shorter handed table you have to pay the Big Blinds more frequently, so waiting for a good hand can become quite expensive. When you lower your Range you can compensate for this effect, knowing that the chances of being against top Holecards are dependent on the number of players aswell. The question remains: In which extend can we make conclusions from what we can observe on a live table (the observed average playrate) to the real Range of the player?

In the following subsection we will show that the observed average playrate is not a good indicator for the Range of a player. Furthermore we will use this inequality to derive a metric from it (px), which will help to identify the true Range of a player later on.

7.1 px and fx

If it is true that the observed average playrate (A) of a player represents his Range (R) of cards he plays, so that: $H0 : A = R$, We could say that for every hand (H) he played (p), this Holecards have to be element of the top A% of Holecards: $H|p \in A$. $px = \frac{\Sigma(H|p \in A)}{\Sigma(H|p)} \approx 1$.

We will calculate the numbers seperately for tables of different sizes, because hand selection and table size are negatively correlated. Including data from more than one tablesize would deminish the quality. To show that what we will find out is valid not only for one table size but for all of them, we will perform grouped operations.

```
A <-Holecards %>%
  group_by(PlayerID,NoPlayers) %>% #group by player and tablesize
  summarise(Playrate=round(mean(played),3))
```

Now that we now A for both our players, we can check if $A = R$. We expect values close to 1, which would indicate that all played hands belong to the top A% Range of hands.

```
Holecards.A <- Holecards %>% left_join(A,by=c("PlayerID","NoPlayers"))
kable(cbind(Holecards.A %>% group_by(NoPlayers) %>%
  filter(played==1 & PlayerID=="Hero") %>% #all played hands of Hero
  summarise(px.percent.Hero=mean(percentcumsum<=Playrate),
            #calculate how many of of the played cards are in the range
            Ahero=max(Playrate)) ,
```

```r
Holecards.A %>% group_by(NoPlayers) %>%
  filter(played==1 & PlayerID=="Zero") %>% #all played hands of Hero
  summarise(px.percent.Zero=mean(percentcumsum<=Playrate),
            #calculate how many of of the played cards are in the range
            Azero=max(Playrate))  )[-4],
caption="Observed Average Playrate and px by Player and Number of Opponents")
```

Table 7.1: Observed Average Playrate and px by Player and Number of Opponents

NoPlayers	px.percent.Hero	Ahero	px.percent.Zero	Azero
2	0.8466055	0.761	0.7958173	0.602
3	0.7614350	0.593	0.7231001	0.463
4	0.6803069	0.386	0.6404624	0.303
5	0.6454121	0.362	0.6531729	0.277
6	0.7041773	0.341	0.6411439	0.246
7	0.6781609	0.244	0.6507115	0.243
8	0.6616466	0.269	0.6676843	0.251
9	0.6516184	0.222	0.6597452	0.244

We can see that for both players and every subset of tablesizes $A \neq R$. The lower the number of players on the table (and the more % hands are played) the higher the percent of hands being in the A Range. $A = R$ only for a player who plays every hand ($A = 1$). One would have assumed that there is a clean threshold in a players preferences, something like a cutoff: "I don´t play hands worse than A3o" but we cannot observe this behavior with the assumption that the observed playrate equals the true Range.

The observed playrate might be biased by hands which will be played because its mathematically correct (many callers before or sitting on the Big Blind), even if they might not be within Range R.
The mathematical correct playing of Holecards which are outside of the Range of a play is what we call a "forced play". The Range should not reflect decisions which are forced. Contrairy to this a fold is always an active decision to not play. Therefore we can formalize a second metric: $fx = \frac{\Sigma(H|f \in A))}{\Sigma(H|f)} \approx 0$.
In other words, we expect a player to not fold Holecards when they are among the best A% of Holecards. A value of 0 would mean that once a player has Holecards from within A%, he plays it. We denote this value fx (% folded Holecards among the top %).

```r
kable(cbind(Holecards.A %>% group_by(NoPlayers) %>%
  filter(played==0 & PlayerID=="Hero") %>% #all played hands of Hero
  summarise(fx.percent.Hero=mean(percentcumsum<=Playrate),
            #calculate how many of of the folded cards are in the range
            Ahero=max(Playrate)) ,
Holecards.A %>% group_by(NoPlayers) %>%
  filter(played==0 & PlayerID=="Zero") %>% #all played hands of Hero
  summarise(fx.percent.Zero=mean(percentcumsum<=Playrate),
            #calculate how many of of the foldedcards are in the range
            Azero=max(Playrate))  )[-4],
caption="Observed Average Playrate and fx by Player and Number of Opponents")
```

Table 7.2: Observed Average Playrate and fx by Player and Number of Opponents

NoPlayers	fx.percent.Hero	Ahero	fx.percent.Zero	Azero
2	0.5005834	0.761	0.3125000	0.602
3	0.3514043	0.593	0.2409209	0.463
4	0.2238325	0.386	0.1368209	0.303
5	0.2037037	0.362	0.1371644	0.277
6	0.1636524	0.341	0.1109771	0.246
7	0.1048735	0.244	0.1029687	0.243
8	0.1256003	0.269	0.1087591	0.251
9	0.0904010	0.222	0.0978029	0.244

We observe low values when the number of players on the table is high, which means a low probability of folding a Hand from Range $R = A$. Still, we would expect those values to be very close to zero, if $R = A$ holds true. Therefore we can say that given our 71,000 real life observations the observed playrate is not a good representation of the actual Range of a player.

7.2 Going Beyond the Average Playrate

px and fx can be calculated not just for the A% threshold but for any number of intervals. We denote the observed threshold as x% or x, and state that px is a vector with the px values of $p|x$ from x=0 to x=1 in .01 increments. We increase x by a percent point at the time starting by 0 up to 1 resulting in 101 observations.

px can be rewritten as $P(x|p)$: Under the condition that the Holecards getting played, what is the chance that the Holecards are among the top x% Range (Rx?
From all played hands, how many have been among the top x% or better?
Intuitively px becomes 1 when x becomes 1, which would be the same as if asked "From all played hands, which percent of hands played belong to the best 100% of Holecards?" The answer is: all of them, 1. px has to increase monochronic. The more restricted the player is when choosing from Range (R), the more area under the curve. If he plays at random, the line becomes a diagonal.

fx can be rewritten as $P(x|f)$: From all the folded hands, how many of them have been top x% or better? This value is less effected by game situations (for example playing a hand for free on the Big Blind), since a fold is always an active decision not to play a hand. We expect low values until around 18-24%, which is a usual Range for a experienced player. The curve tend to break out after being close to zero for early percentages of x.

```r
tmp.cards <- Holecards %>%
  select(GameID,played,high,kicker,Symbol,suit,percentcumsum, NoPlayers,PlayerID)
tmp.df1 <- as.data.frame(matrix(0,ncol=2,nrow=length(seq(0.,1,0.01)),
                                dimnames = list(1:101,c("px","fx"))))

  tmp.df1[,"px"]<- unlist(sapply(seq(0,1,0.01),
               FUN=function(x){tmp.cards %>%
                   filter(played==1 & PlayerID=="Hero") %>%
                   summarise(m=mean(percentcumsum<=x))}
               ))

  tmp.df1[,"fx"]<- unlist(sapply(seq(0,1,0.01),
                FUN=function(x){tmp.cards %>%
                    filter(played==0 & PlayerID=="Hero") %>%
                summarise(m=mean(percentcumsum<=x))}
        ))
```

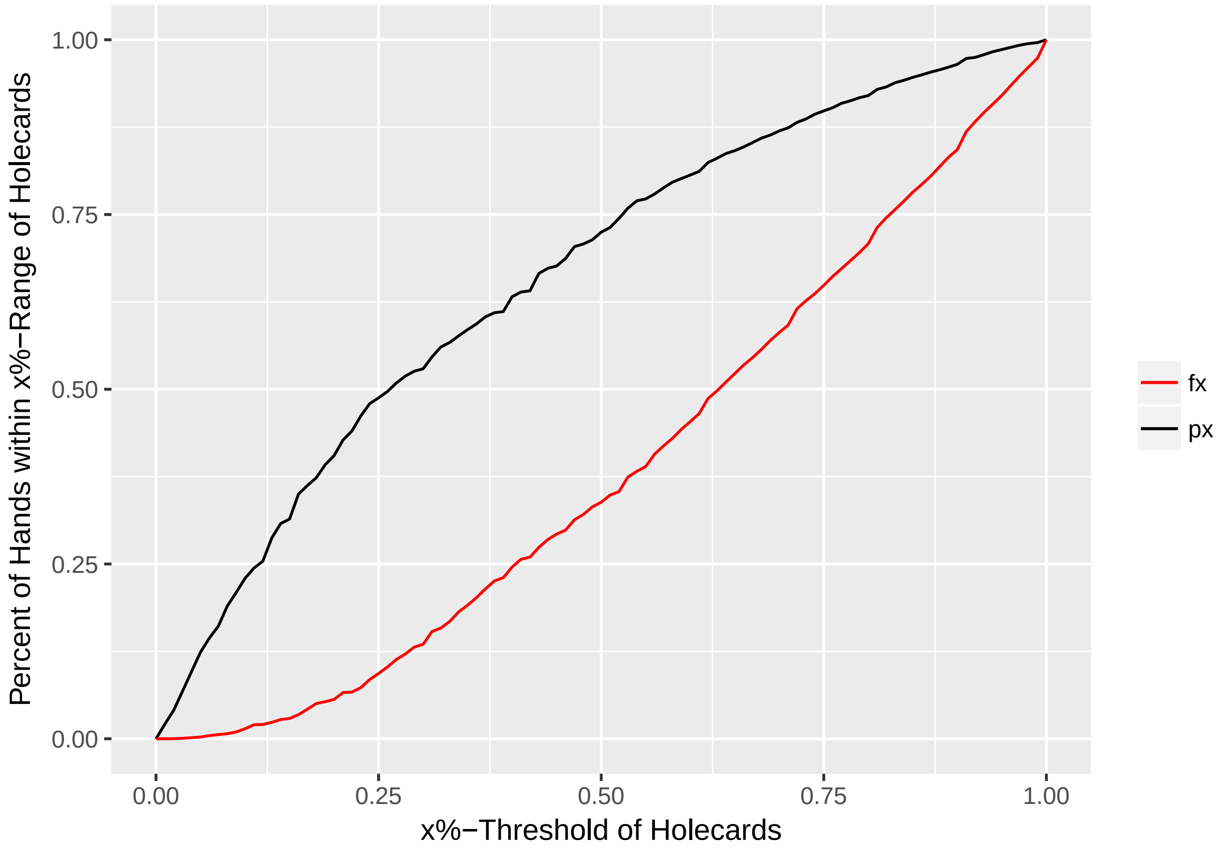


Figure 7.1: px and fx for Hero

To interpred the black curve (px) we look at the area under the curve. Imagine a very restricted player who plays only the top 10% of Holecards. In that case the curve would in fact be aproximately rectangular, e.g reaching 1 on the Y-axis within $\leq .10$ on the X-axis. The more area is covered by the black line, the smaller the overall Range of a player. Yet we cannot make a conclusion on how the Range is set up.
Interpreting the curve for the red line (fx): The red curve represents the willingness of the player to not play top hands. We would expect those values to be close to zero at lower top-x% and increasing as x% goes up. We need to discuss two aspects of fx:

- For how long the value stays close to zero (indicating strong willingness to play Holecards of the Rx),
- the bend on which is starts to increase more rapidly (around 0.2).

For our example of the player who plays only top 10% hands, fx would be 0 until .1x and increasing from this point constantly with rate i $i = 100/(100 - x)$.
The more convex the diviation of the fx from i is (starting at the bend), the more fluctuation there is in hand selection.

First we need a function which can calculate px and fx without having to rewrite the same lines when ever we want to calculate px and fx.

```
calc.pxfx <- function(data){
  data<-cbind.data.frame(x_threshold=seq(0,1,0.01),
  px=unlist(sapply(seq(0,1,0.01),
               FUN=function(x){data %>%
                   filter(played==1) %>%
                   summarise(px=mean(percentcumsum<=x))})),
  fx=unlist(sapply(seq(0,1,0.01),
               FUN=function(x){data %>%
                   filter(played==0) %>%
                   summarise(fx=mean(percentcumsum<=x))})),
  stringsAsFactors=FALSE
  )
  rownames(data)<-NULL
  return(data)
}
```

```
pxfx_by_size <- Holecards %>%
  select(GameID,played,percentcumsum, NoPlayers,PlayerID) %>%
  group_by(GameID) %>%
  mutate(NoPlayers=ifelse(max(NoPlayers)<=4,
                          "short-handed",ifelse(max(NoPlayers)>=7,
                                                "long-handed","mid-handed")),
         NoPlayers=factor(NoPlayers,
                          levels=c("long-handed","mid-handed","short-handed"))) %>%
  ungroup() %>%
  group_by(PlayerID,NoPlayers) %>%
  do(calc.pxfx(.))
```

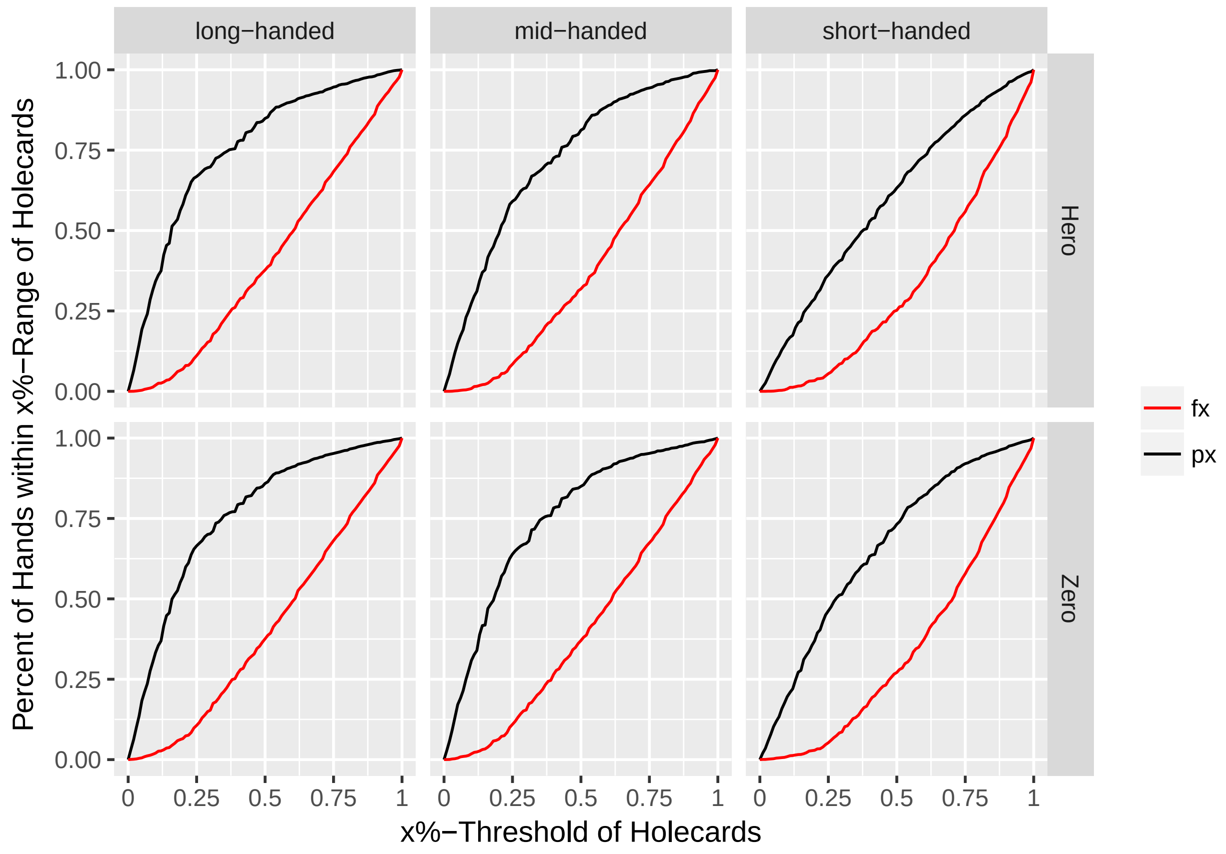


Figure 7.2: px and fx by Player and Tablesize

We can examine on long-handed tables (left diagrams) that Hero and Zero are very restricted outside of their comfortzone (the bend in fx), leading to continues increase of fx after the bend. When observing a short table (rhs), we examine more diviation and a more concave shape of fx.

When looking at px, we can see that the area under curve goes down, when the number of players are decreasing. On long-handed tables we have nice sharp edges on both players. On short handed tables both players show different playing styles: Hero (right top diagram) barely shows any selection in R, his resulting px is therefore nearly at a random level (diagonal), while Zero remains his hand selection, the area under curve is getting smaller nethertheless (bottom right).

With our basic understanding of table sizes and the negative correlation between the number of players and the Range which is played, we can prove that px and fx are able to show these effects on handselection.

7.3 px fx Differential

To get an insight about the actual Range of the player, we can substract the px and fx values from another: The vector px which is the willingness to play a hand better than the x% threshold, minus the willingness to fold Holecards better than the x% threshold. The blue line on the plot below shows the behavior of the difference between px and fx, namingly the difference between willingness to play and to fold.

```
pxfx_by_player <- Holecards %>%
  select(GameID,played,percentcumsum, NoPlayers,PlayerID) %>%
  group_by(PlayerID) %>% do(calc.pxfx(.))
```

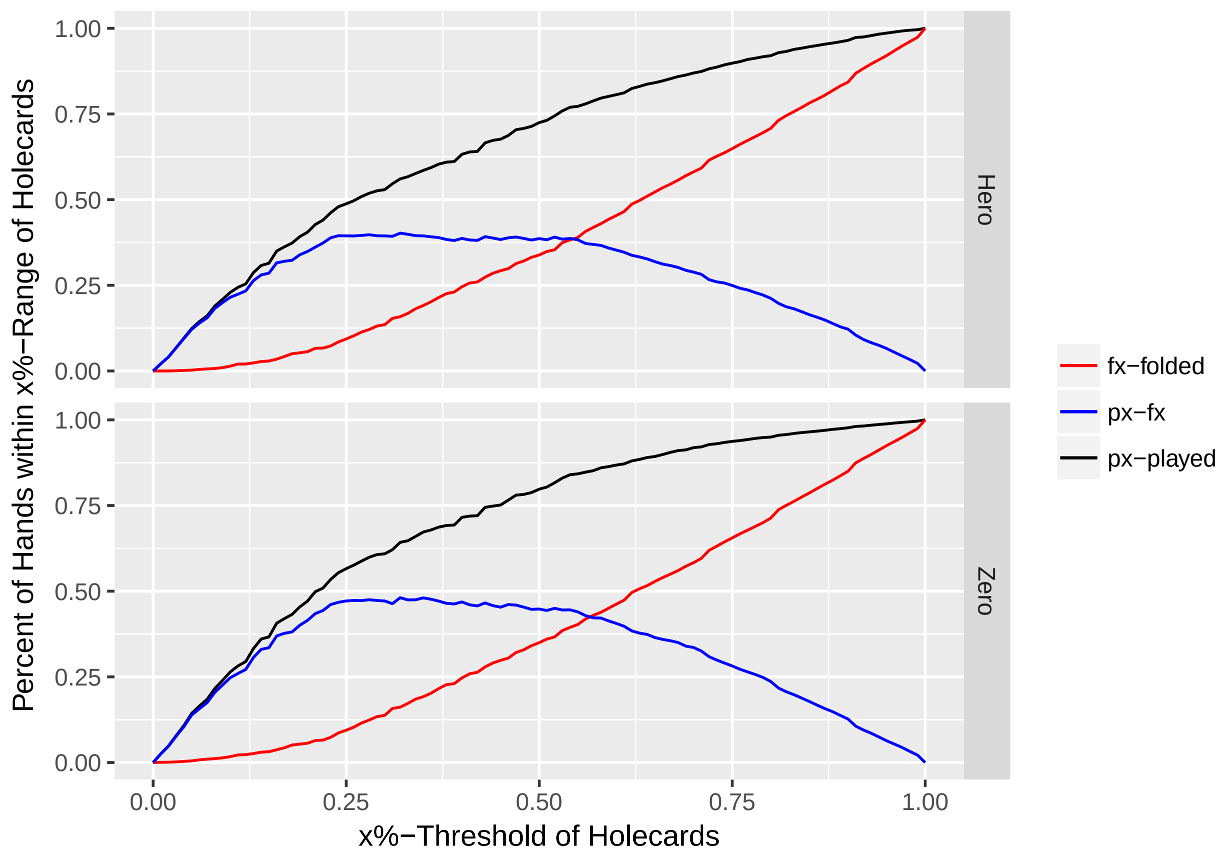


Figure 7.3: px, fx and px-fx by players

```
Ao_by_size <- Holecards %>% select(GameID,played,percentcumsum, NoPlayers,PlayerID) %>%
      group_by(GameID) %>%
      mutate(NoPlayers=ifelse(max(NoPlayers)<=4,"short-handed",
                            ifelse(max(NoPlayers)>=7,"long-handed","mid-handed")),
           NoPlayers=factor(NoPlayers,
                            levels=c("long-handed","mid-handed","short-handed"))) %>%
      ungroup() %>%
      group_by(PlayerID,NoPlayers) %>%
      summarise(Ao=mean(played))

maxima_by_size <- pxfx_by_size %>%
    mutate(dx=px-fx) %>%
    group_by(PlayerID,NoPlayers) %>%
    filter(dx==max(dx)) %>%
    select(mwd=x_threshold)
```

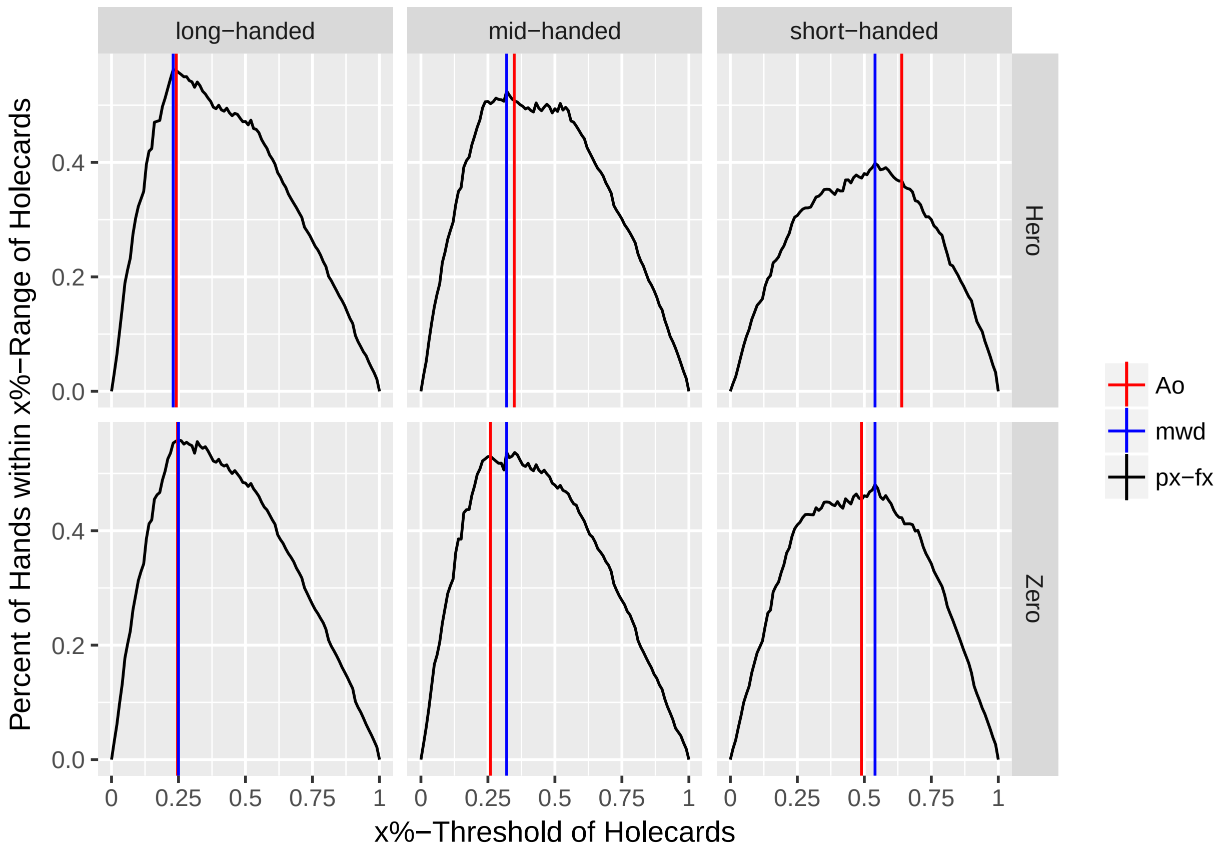


Figure 7.4: px-fx and mwd by Player and Tablesize

The result is a curve ($px - fx$) on which we can identify a maximum which represents the x% threshold on which the difference between willingness to play and willingness to fold a hand is maximised (blue vertical line). We call the maximum point mwd - maximum willingness differential.

"The Range (Rx%) threshold on which the difference between willingness to play and willingness to fold a hand is maximised" could also be a textbook definition of a players "Range". And as we can see on the tight, restricted long-handed tables, mwd and average playrate (red vertical line) are surprisingly close to each other. Even though mwd was calculated with a complete different mindset and information.
It is in the nature of long-handed tables to be very selective, here the average playrate can be a very good indicator. When the number of players goes up, A and mwd shifts to the right, indicating a looser handselection.

7.3.1 Gap b

What is actually interesting is the gap between the blue and red line, namingly the difference between mwd and the average playrate. Lets call this gap b. |b| seems to be negatively correlated to the number of players. As the number of players goes down, the absolut value of b (|b|) tends to increase. Thus |b| correlates positively with the overall tighness/looseness of the table. On tight tables, |b| tends to be small and vice versa.

The gap is explained by considering hands on the Big Blind which have been checked or played for relatively small amounts of money: They can be outside of the real Range, yet still be played. To deepen the understanding: When a player sits on the Big Blind he already put money into the pot before getting the

Holecards. So when the player has to act, he gets a better price for a call or he can check for free, when there was no raise before him.

Obviously, when the number of players goes down, players have to pay Blinds more often which leads to higher frequences of played hands outside of the players real Range. Yet, on an average playrate, this would not have been accounted for. Therefore there has to be a gap between the observed playrate and the real Range.
Blinds are also the reason why on shorter tables hand selection gets quite loose: There is more mathematical pressure to play. On a 3 player table, you pay the Blinds inbetween 3 hands. Since the Small Blind is the half of the Big Blind, you have to pay 1.5bb every 3 hands, which leads to 0.5bb investment per hand (no matter if you play it or not). On a long handed table with 10 players for example a hand only costs 0.15bb. Therefore you have to play more hands on short tables. Since waiting for the best hands costs more than they give you in return, you have to adjust your Range.

To illustrate how A, mwd and b behave we can look at this easy formular, where mwd is the approximate true Range of a player, A is the average playrate observed while playing and b is a effect of yet unknown size which represents forced plays.

$$\begin{aligned} mwd + b &= A \\ mwd &= A - b \\ b &= A - mwd \end{aligned} \tag{7.1}$$

It seems, that if we could quantify b, it could be possible to identify the true Range from the average playrate. Which would be a great advantage in reading your opponent and for all research on the topic.

Sidenote: On the buttom/middle diagram you can observe a negative value b for player Zero, the interpretation of negative values will be approached in the next chapter.

7.3.2 no subset mwd

Without subsetting the data for players, tables sizes or anything else, this is what the mwd metrics look like. We can still observe all points of interest:

- the area under curve for px,
- the bend at the .2 mark for fx and
- the steadiness of the increase of fx for x >=0.2.

Additionally we can see that on average without subsetting,

- mwd and A are close to each other with
- a slightly positive b over a total of more than 71,000 hands.

```
pxfxdx_unsubsetted <- calc.pxfx(Holecards) %>% mutate(dx=px-fx)
```

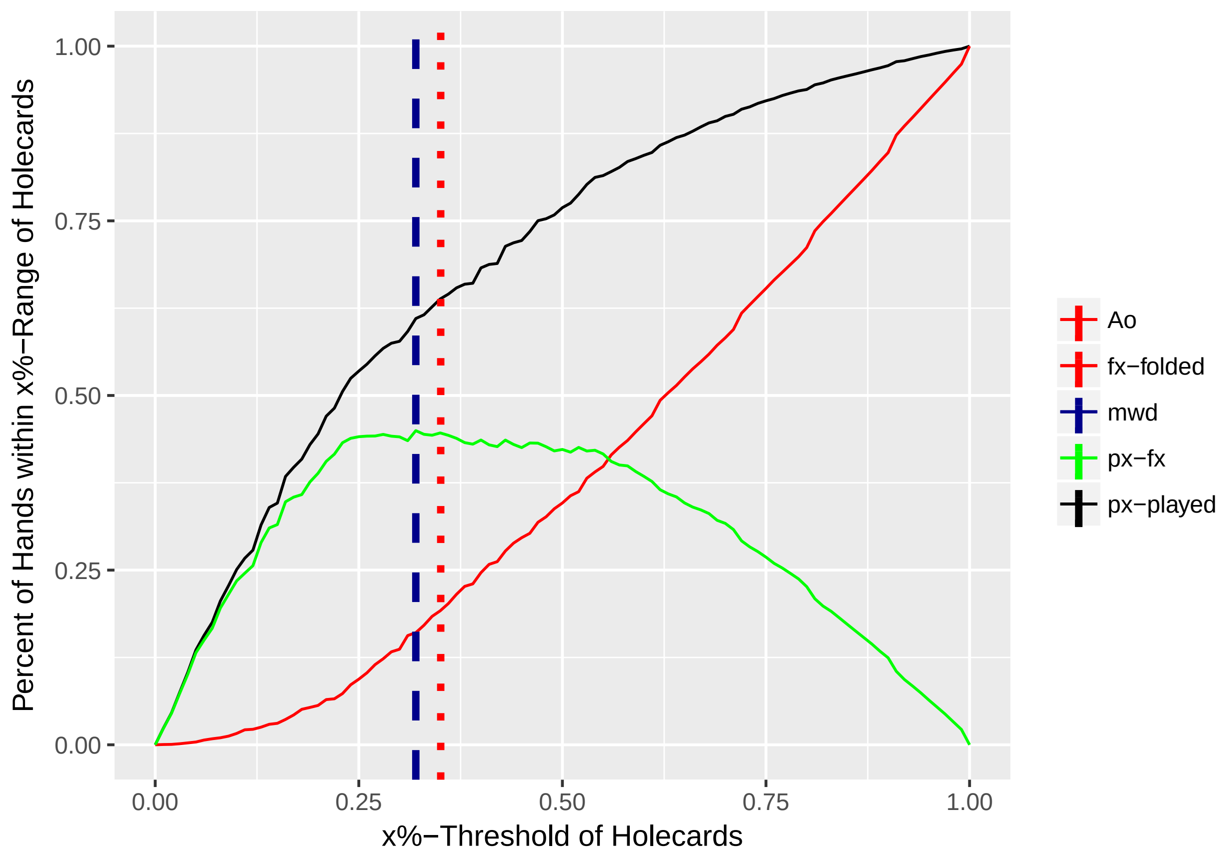


Figure 7.5: px, fx, px-fx, Ao and mwd unsubsetted

7.4 Holecard Selection versus Relative Position

An imporant factor when deciding if to play the Holecards you have been dealt is the position on the table. From under the gun (the first position after the Blinds) you have to act before you can get information about the players behind you, while on the button (the dealer, the last position) you have seen every other player except for the Blinds acting before you. From early positions the Range of the player is smaller to compensate for uncertainty added by all those players who have still to act. In late position decisions are easier. Overall players in late position have a lot easier time later in the game, since they always have to act after their opponents, which allows them to select their Holecards from a much looser Range.

Due to these effects positioning matters a lot when trying to read a player, since it determines the Range of possible hands. Now we take a look at how the position effects the playrates of Holecards. To do this, we stick with mid- to long-handed tables. On short-handed tables the position does not matter less, but the looseness on short-handed hands makes evaluating the effect of position rather complicated. The problem is: On short tables (2-4 players), it is difficult to classify a position to be “late” or “early”.
We start this chapter by engineering our data.frame and writing a function which can label the position based on two arguments: The number of players on the table and the position.

```
Holecards.pos <- Holecards %>%
  left_join(Logdata %>% filter(PlayerID %in% c("Hero","Zero") & Round=="Pre-Flop") %>%
            select(PLremain,relPos,prevBet,GameID=PSID,action.type,PlayerID) %>%
            group_by(GameID) %>%
            filter(prevBet==max(prevBet)))
```

First we need to get the information from the `Logdata`. We extend our `Holecards` dataframe by information about the number of players remaining in the hand, the number of previous bets, the relative position and the action type which has been choosen. What we need is a function which can evaluate a position and tell us if its either "early", "middle" or "late" position.

```
get.position <- function(Position,NoPlayers){
  tmp <- "Middle"
  if((Position==1 | Position==2) && NoPlayers>=7){tmp <- "Early"}
  if(Position>round(NoPlayers-2-NoPlayers/3)){tmp <- "Late"}
  if(NoPlayers-2==Position){tmp <-"Late"}#Dealer
  if(Position==NoPlayers-1){tmp <- "Blinds"} #SB
  if(Position==NoPlayers){tmp <- "Blinds"} #BB
  return(tmp)
}
```

```
sapply(1:10,FUN="get.position",NoPlayers=10)
```

```
##  [1] "Early"  "Early"  "Middle" "Middle" "Middle" "Late"   "Late"
##  [8] "Late"   "Blinds" "Blinds"
```

```
Holecards.pos <- Holecards.pos %>% na.omit() %>%
  group_by(GameID) %>%
  mutate(seat=get.position(relPos,NoPlayers)) %>%
  mutate(seat=factor(seat,levels=c("Blinds","Early","Middle","Late")))
```

There are 146 NAs, which will be omitted.

Table 7.3: Holecards Data.Frame (example columns and rows)

Symbol	played	NoPlayers	PlayerID	action.type	seat
J6o	0	6	Hero	fold	Late
T8o	0	6	Hero	fold	Middle
Q7o	0	6	Hero	fold	Middle
J7o	1	6	Hero	check	Blinds
43o	0	5	Hero	fold	Blinds
92o	0	5	Hero	fold	Late

The question at hand is: Does the position on the table have an influence on the cards our player is playing?

What we expect from a experienced player, is to be very narrow in his Range in early positions, loosening up the later he/she has to act. He will have the widest Range when in the Blinds, not just because he is already in the pot and gets good odds to call even with weaker hands, but also because he can sometimes play hands for free and has seen every other player act once in the hand.

```
tmp <- Holecards.pos %>%
  filter(NoPlayers>=4) %>%
  select(GameID,played,high,kicker,Symbol,suit,percentcumsum, seat,PlayerID) %>%
  group_by(Symbol,seat,PlayerID) %>%
  summarise(Playrate=mean(played),
          highcard=ifelse(max(suit)=="o",max(high),max(kicker)),
          kickercard=ifelse(max(suit)=="o",max(kicker),max(high)))%>%
  ungroup() %>%
  mutate(highcard=factor(highcard,levels=values),
                       kickercard=factor(kickercard,levels=values))
```

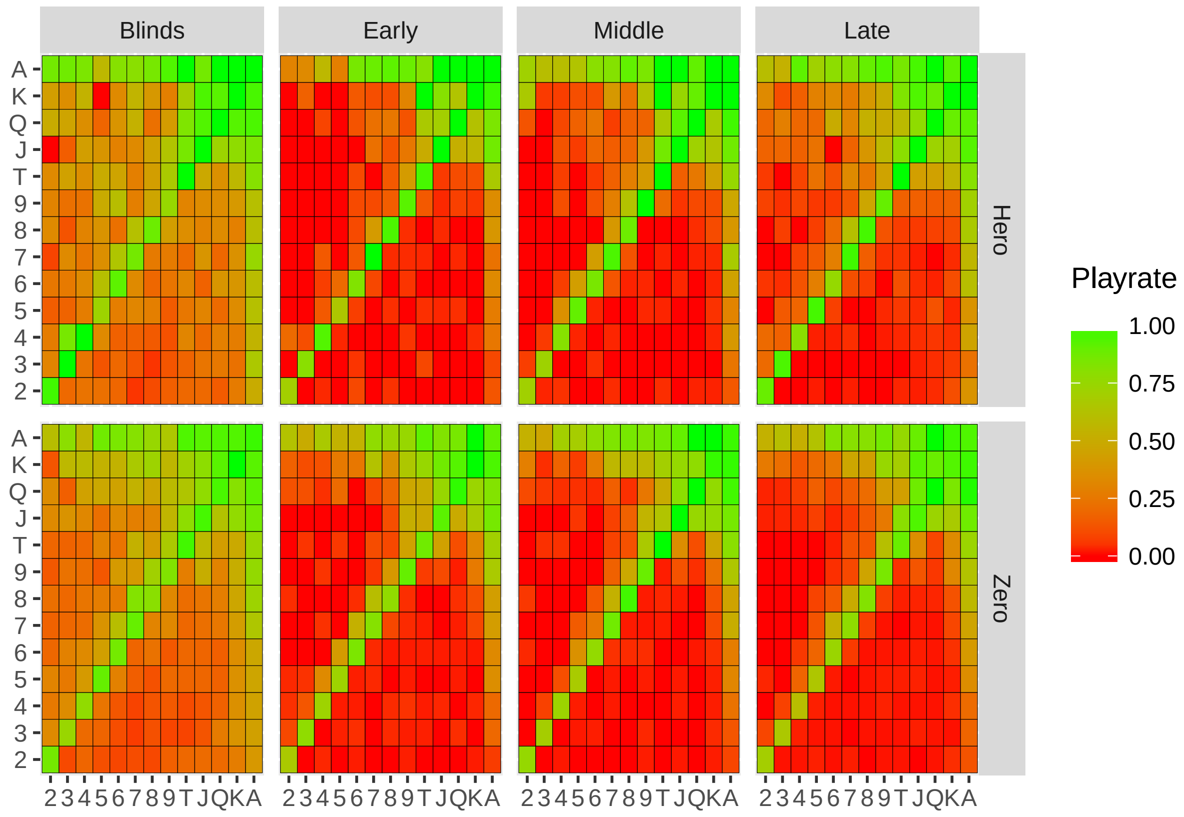


Figure 7.6: Holecard Playrate Distribution Heatmap by Position and Player

As we can see our players know about the concept of position and adjust their handselection accordingly. Again, as how we did it for the size of the table, we can calculate our mwd. If the metrics make sense again, we can be more sure about the interpretability of it.

```
pxfx_by_position <- Holecards.pos %>%
  select(GameID,played,percentcumsum, seat,PlayerID) %>%
  ungroup() %>%
  group_by(PlayerID,seat) %>%
  do(calc.pxfx(.))
```

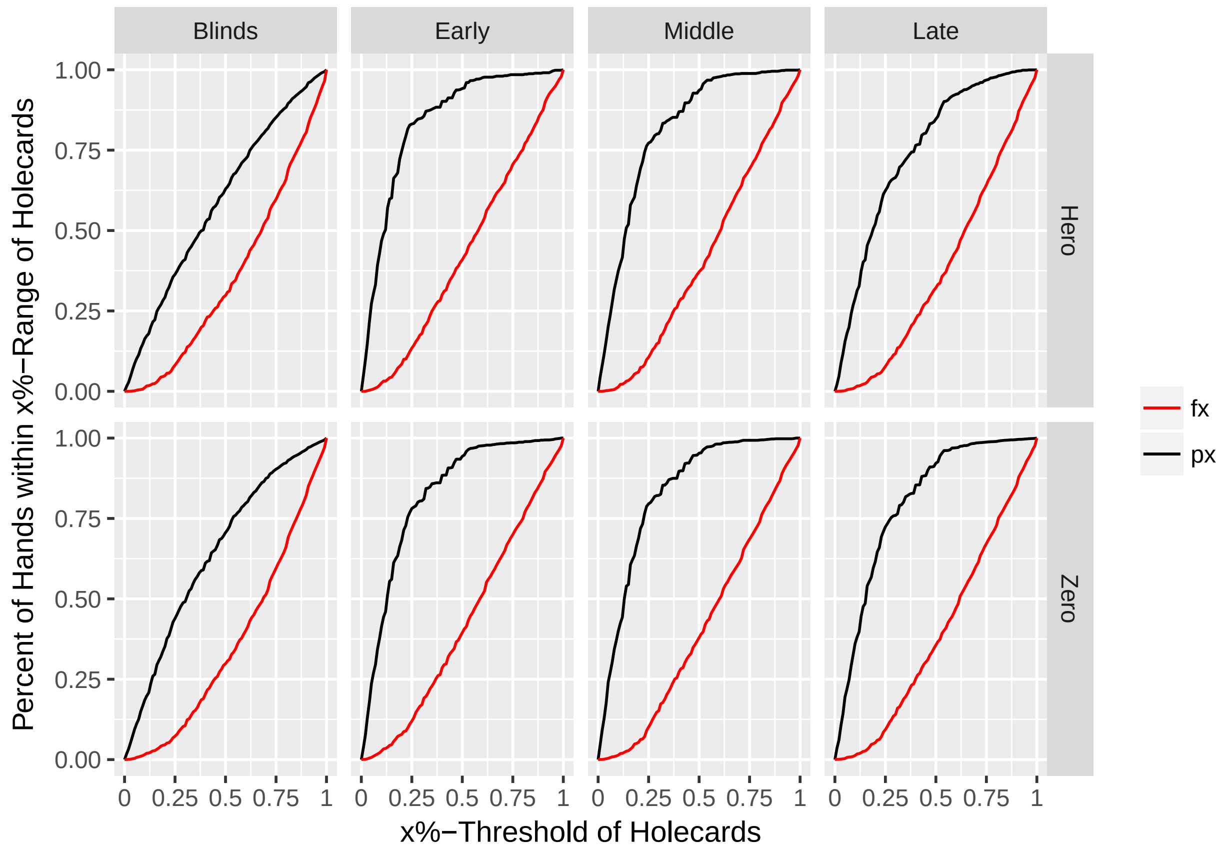


Figure 7.7: px and fx by Player and Position

We can observe very sharp edges in the px when looking at the early positions, which converge towards the diagonal as the position gets later. This indicates a looser Range selection as the number of players who still have to act behind the player goes down.

```
Ao_by_position <- Holecards.pos %>% select(GameID,played,percentcumsum, seat,PlayerID) %>%
      group_by(PlayerID,seat) %>%
      summarise(Ao=mean(played))

maxima_by_position <- pxfx_by_position %>%
    mutate(dx=px-fx) %>%
    group_by(PlayerID,seat) %>%
    filter(dx==max(dx)) %>%
    select(mwd=x_threshold)
```

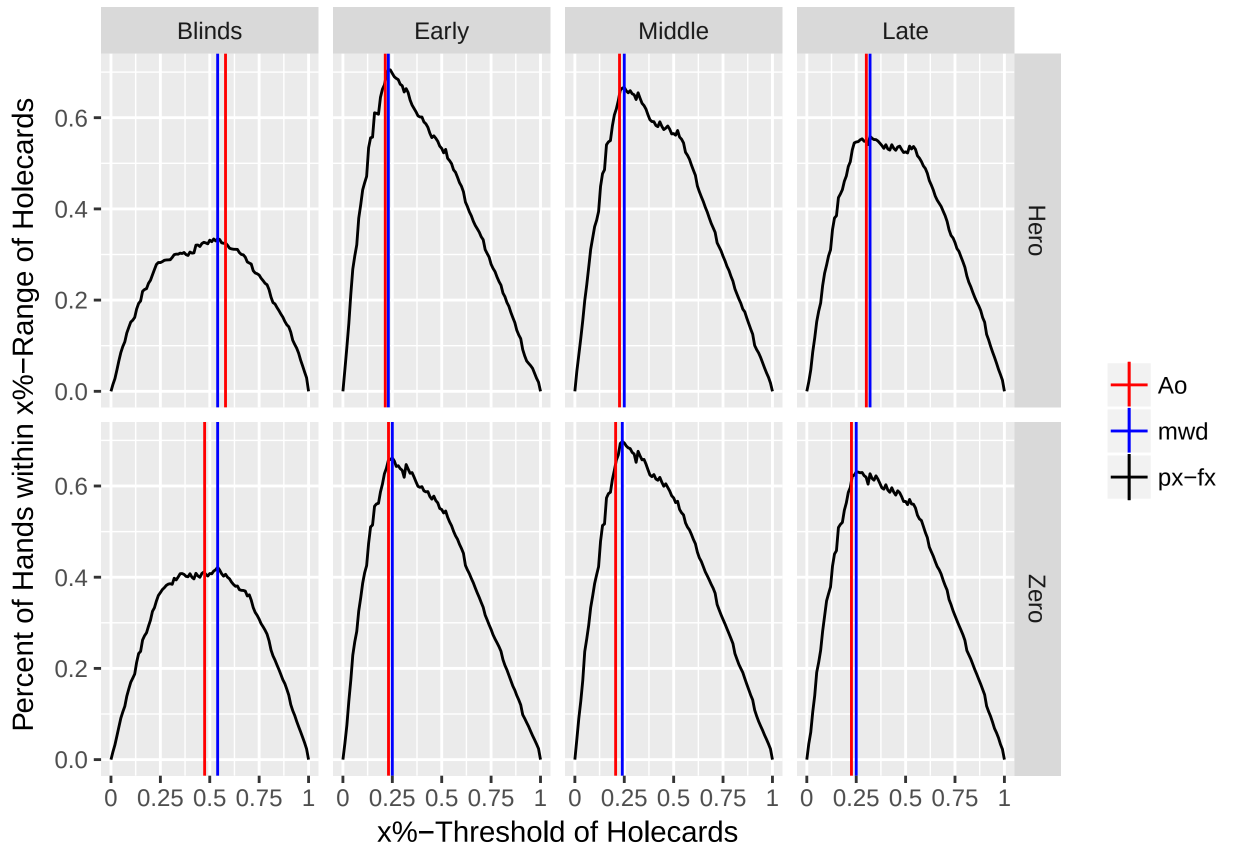


Figure 7.8: px-fx and mwd by Player and Tablesize

We can clearly see the same effect: the further to the right mwd is, the looser the hand selection. Interesting is: we interpreted the gap (b) between the blue line (mwd) and the red line (the average winrate) as an indicator of hands which are outside of the Range but played because of blind effects. We can further confirm this assumption now: On every position, which are not the Blinds, we have negative b values.
This complements to what we observed as we looked at the metrics seperated by size of the table. Here we argued that shorter tables have a higher portion of Blinds than long tables, resulting in a bigger (positive) gap b. For every table size the gap should be positive, because on every table, no matter the size, a player will always face Blinds. When investigating the metrics by position, we can observe a shift in this pattern: Only when sitting in the Blinds, the player can be effected by Big Blind forced plays, so when not sitting in the Blinds, the player is not forced to play outside his Range, therefore b is always negative.
A negative value b is indicating a percentage playrate which undercuts the played Range of the player. One would expect both values to be of approximately the same size. This is up to interpretation, a first impression would be: The higher the negative b, the more inconsistend the player is, which means that he plays the same cards on similar position differently, which widens his Range while reducing the average winrate.

7.4.1 Excursion to “Zero”´s mid-sized table hand selection anormaly

The interpretation on the negative b-values is also true for the negative b for Zeros mid-handed tables (Figure 7.4, bottom-middle diagram). There we could observe an unusual behavior of Rx with a local maximum shifted to the left of the actual maximum. Curiously the average playrate A was just on point on this local maximum. Here we can interpred this as a pattern in Zeros hand selection, he obviously chooses from a bigger

Range with a gap in between. A look at the Heatmap in the previous chapter reveals that he is refusing to play small aces (A5 to A2) as well medium ranged hands as QTs-Q6s and J9s-J7s. Still he plays hands which are worse than those, which leads to a gap in the Range and a local maximum.

7.5 Holecard Selection versus Previous Actions

Not just position and table size are important when deciding whether to play your Holecards: As 3rd and last major factor we will take a look at previous actions. First we need to clarify what we mean when talking about "much action" or "little action". As in nearly every situation in poker, a player has three lines of play to choose from:

- fold
- check/call
- bet/raise

We can translate those actions into three types: passive, neutral and aggressive actions. Given a table size and position, previous actions influence decision making the most. When talking about "little- or much action" we always refer to the number of non-passive actions. To get an idea of the effect, we take a look at a situation in which our player sits in late position on a long (8 player) table. The player under the gun (the first one to act) raises 3 Big Blinds. This is our base situation, 4 players will act before us, their decisions influence ours. Lets examine some possible events:

- folding towards the player. Imagine the actions are: fold, fold, fold, fold. If you assume that the Blinds will fold as well, you can think about this as a 1v1 hand. The opponent raised 3 Big Blinds, so the pot contains 4.5 bb at that point. You have to invest 3 bb to play. These are 1:1.5 odds which equals 40%. In other words, you have to win at least 40% of the time to break even. Keeping in mind that the opponent raised from out of position, we have to set him on a narrow Range, in order to get the required 40% winrate, our hand must come from an even smaller Range.
- multiple callers: The actions goes: fold, fold, call, fold. Now you have to invest 3 bb in a pot of 7.5, which lowers the requirement to your hand, you only have to win 28.6% of the hands to break even. So as more and more players call before you, the better the price for you to play the hand. Additionally, since for every caller the same effect occurs (they can lower their requirements on their Holecards), you can assume, that they are playing hands on the lower end of their Range most of the time. An early caller will hold better hands on average than a late caller. We know that this assumption holds true, since we checked it in the last chapter.
- multiple raises: Being the first player to call a raise requires a smaller Range than the initial raiser. Imagine a situation where it goes call, fold, raise, fold. You are to act and have 2 raises and a caller in front of you. The first caller has to be on a short Range to call the initial raise. On what narrow of a Range has the second raiser to be to raise in this situation? We should seriously narrow our Range a lot at this situation. The more action, the smaller the Range.

7.5.1 Data Engineering for Actions

We will generate 3 new features: the number of calls, raises and folds before the last action of our player. So if the player has to act twice or more times pre flop, we will count his previous actions as if they were taken by any other player. This is consistent with the fact that we looked at the final decision (played / folded) all the time, which is the result of his latest decision, not of the decisions he may have taken in previous betting rounds of the same hand.

```
actions <- Logdata %>%
  group_by(GameID=PSID) %>%
  filter(Round=="Pre-Flop") %>%
  mutate(rownum=row_number(),
         cutoff=ifelse(PlayerID %in% c("Hero","Zero"),rownum,0)) %>%
  filter(rownum<max(cutoff)) %>%
  summarise(calls=sum(action.type=="call" | action.type=="check"),
            raises=sum(action.type=="raise"),
            folds=sum(action.type=="fold"))

Holecards.act <- Holecards.pos %>% left_join(actions, by="GameID")
```

We joined calls and checks. Checks are only possible pre flop in two scenarios: It is called to the Big Blind, this player can check. We won´t have this data in our set, because we dropped every row after our players hand and our players last decision as well. So even if he sits on the Big Blind and checks, we won´t count it as previous action.
The second scenario in which a player is allowed to check is when he or she just joined the table. This player has the option to wait until the next Big Blind to play or to place a Big Blind right away, no matter of the actual position. This can lead to the player checking within the first betting round. In this rare event, a check is of the same value as a call, so we join them together. We see some NAs in our new features, they exist when our player was

- first to act and folded,
- first to act and called the Big Blind and wasn´t raised afterwards,
- first to act and raised and all players folded after him.

All three events are leading to 0 rows for the previous actions for a GameID, which leads to NA entries. Otherwise we can prove that the sum of all our three features is never 0.

```
Holecards.act %>%
        mutate(sum=calls+raises+folds) %>%
        ungroup() %>%
        filter(sum==0) %>%
        summarise(NumberOfZeros=n())
```

```
## # A tibble: 1 x 1
##   NumberOfZeros
##           <int>
## 1             0
```

We don´t have to omit the NAs in the data set, we can replace them with zeros, since we know in the three cases named above there was no previous action.

```
Holecards.act <- Holecards.act %>%
  group_by(GameID) %>%
  mutate(folds=ifelse(is.na(folds),0,folds),
         calls=ifelse(is.na(calls),0,calls),
         raises=ifelse(is.na(raises),0,raises))
```

7.5.2 Holecard Selection versus Multiple Calls

The more players are in a hand, the better the prize to call. This effect is loosening up the Ranges of all players as the number of calls grows. For example the first caller has a tighter Range than the caller behind and so on. We would expect the players to loosen up according to this scheme. In the following analysis we categorize “no calls”, “one call” and “more than one call”.

```
tmp <- Holecards.act %>%
  mutate(actions=calls,
         actions=ifelse(actions==0,"no calls",
                 ifelse(actions<=1,"one call", "more than one call")),
         actions= factor(actions,levels=c("no calls","one call","more than one call"))) %>%
  group_by(Symbol,actions,PlayerID) %>%
  summarise(Playrate=mean(played),
            highcard=ifelse(max(suit)=="o",max(high),max(kicker)),
            kickercard=ifelse(max(suit)=="o",max(kicker),max(high)))%>%
  ungroup() %>%
  mutate(highcard=factor(highcard,levels=values),
                       kickercard=factor(kickercard,levels=values))
```

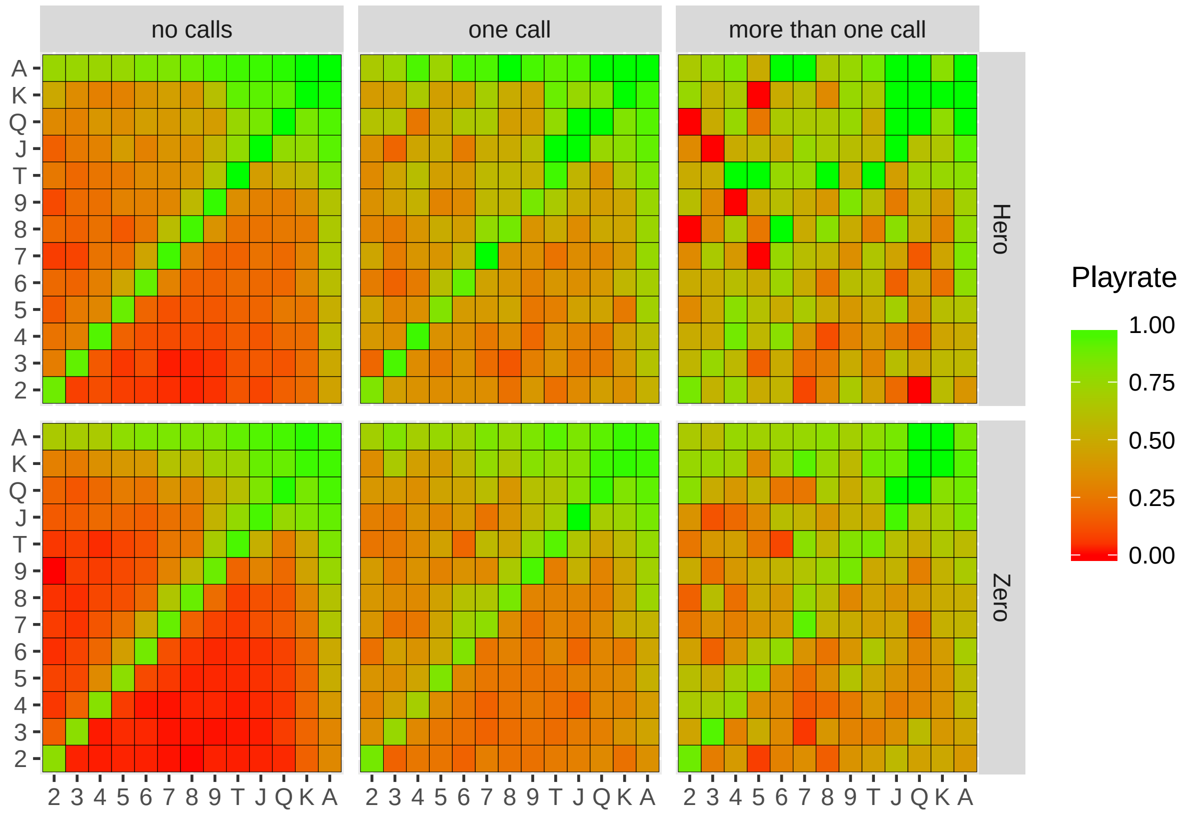


Figure 7.9: Holecard Playrate Distribution Heatmap versus Number of Calls

The results are as expected: tight Ranges when there was no call before and very loose selection when they were multiple calls.

```
pxfx_by_calls <- Holecards.act %>%
  select(GameID,NoPlayers,played,percentcumsum, calls,PlayerID) %>%
  filter(NoPlayers>=4)%>%
  mutate(actions=calls,
         actions=ifelse(actions==0,"no calls",
                 ifelse(actions<=1,"one call", "more than one call")),
         actions= factor(actions,levels=c("no calls","one call","more than one call"))) %>%
  ungroup() %>%
  select(GameID, played, percentcumsum, actions, PlayerID) %>%
  ungroup() %>%
  group_by(PlayerID,actions) %>%
  do(calc.pxfx(.))
```

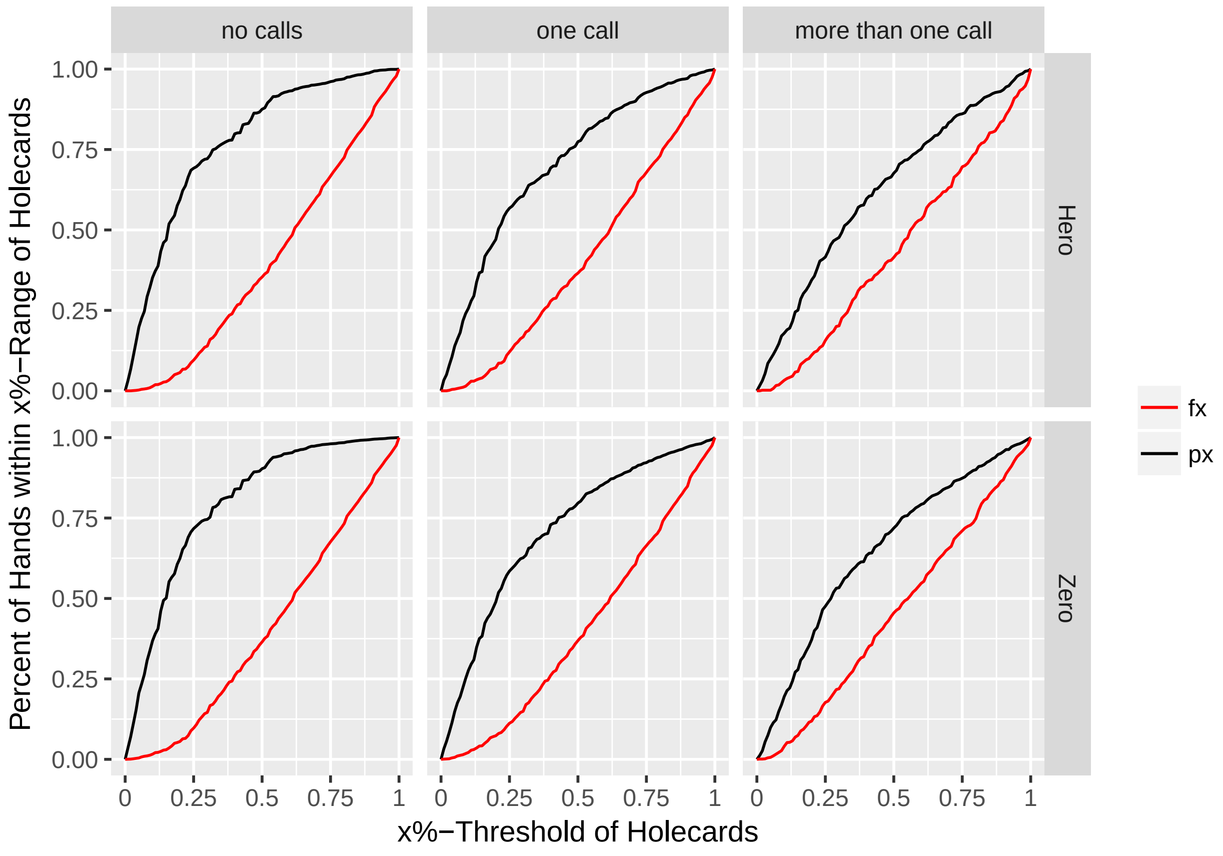


Figure 7.10: px and fx by Player and Number of Calls

We see high area under curves for “no calls” hands, and nearly diagonal px curves for multiple calls. What is worth observing is the bend in fx. We would expect it to be roughly around the .2 mark, like we see on the left hand side. When there are multiple callers, the fx bend is shifted far to the right, which makes fx almost diagonal. This is surprising but also highly reasonable, since the criterea for a Holecards has gone down as the potodds go up. We can interpred this behavior of fx as almost random folding criteria.

```
Ao_by_calls <-  Holecards.act %>%
  select(GameID,NoPlayers,played,percentcumsum, calls,PlayerID) %>%
  filter(NoPlayers>=4)%>%
  mutate(actions=calls,
        actions=ifelse(actions==0,"no calls",
                  ifelse(actions<=1,"one call", "more than one call")),
        actions= factor(actions,levels=c("no calls","one call","more than one call")))%>%
  ungroup() %>%
      group_by(PlayerID,actions) %>%
      summarise(Ao=mean(played))
```

```
maxima_by_calls <- pxfx_by_calls %>%
    mutate(dx=px-fx) %>%
    group_by(PlayerID,actions) %>%
    filter(dx==max(dx)) %>%
    select(mwd=x_threshold)
```

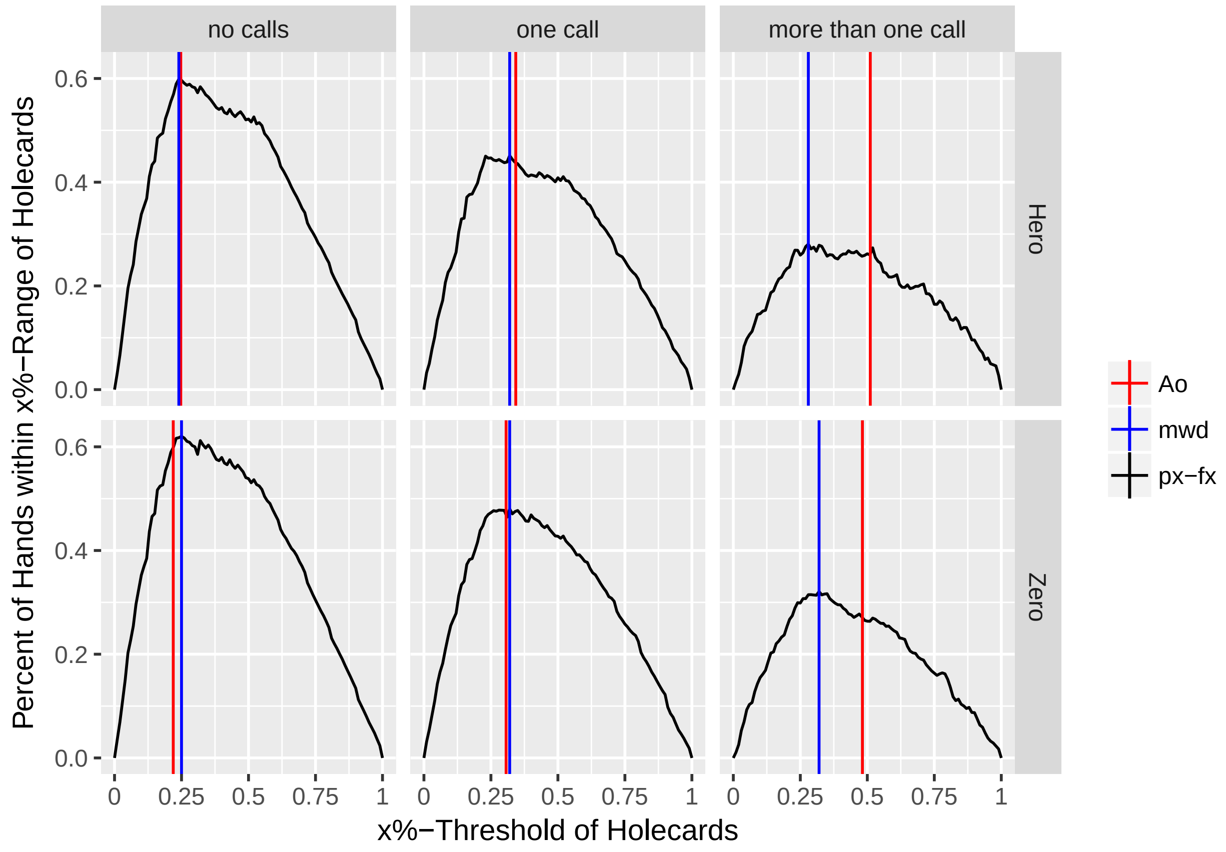


Figure 7.11: px-fx and mwd by Player and Number of Calls

We can see how the Y-value for the mwd is falling, as the looseness increases. Still we are able to observe mwd values which undercut the observed playrate against multiple calls (rhs). This is surprising because the missing fx bend indicated the absence of a Range. Still px-fx indicated a mwd value for both players which seems to identify a core-Range nethertheless.

7.5.3 Holecard Selection versus Multiple Raises

Against multiple raises one would expect a tighter selection of Holecards. Every raise of a player indicates a shorter Range this player may have choosen from. Poker is all about managing the risk and a raise of an opponent induces risk. When there are multiple opponents confident enough about their hand strength, a player has to choose from a tighter Range to not be behind from the pre-flop phase onwards.

```
tmp <- Holecards.act %>%
  mutate(actions=raises,
         actions=ifelse(actions==0,"no raises",
                 ifelse(actions<=1,"one raise", "more than one raise")),
         actions= factor(actions,
                         levels=c("no raises","one raise","more than one raise"))) %>%
  group_by(Symbol,actions,PlayerID) %>%
  summarise(Playrate=mean(played),
            highcard=ifelse(max(suit)=="o",max(high),max(kicker)),
            kickercard=ifelse(max(suit)=="o",max(kicker),max(high)))%>%
  ungroup() %>%
  mutate(highcard=factor(highcard,levels=values),
                        kickercard=factor(kickercard,levels=values))
```

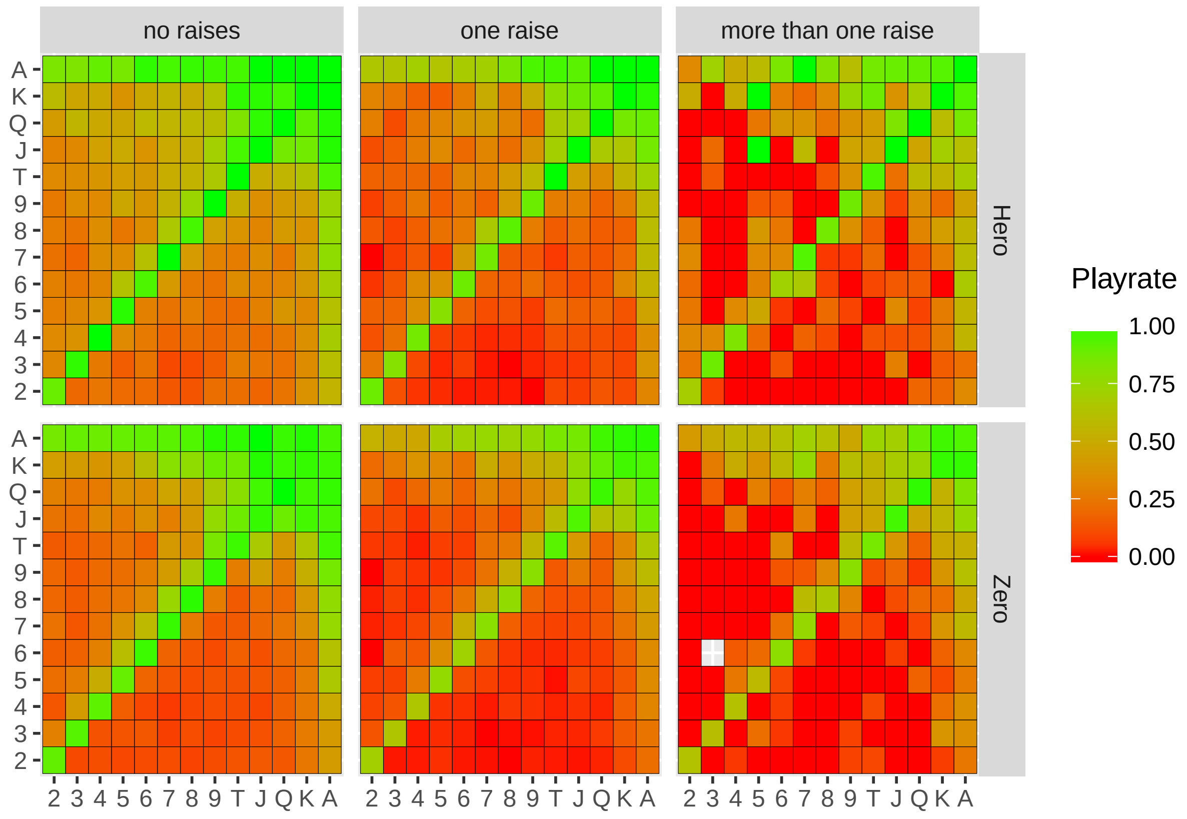


Figure 7.12: Holecards Playrate Distribution Heatmap by Player and Number of Raises

```
pxfx_by_raises <- Holecards.act %>%
  select(GameID,NoPlayers,played,percentcumsum, raises,PlayerID) %>%
  filter(NoPlayers>=4)%>%
  mutate(actions=raises,
         actions=ifelse(actions==0,"no raises",
                 ifelse(actions<=1,"one raise", "more than one raise")),
         actions= factor(actions,
                         levels=c("no raises","one raise","more than one raise"))) %>%
  ungroup() %>%
  select(GameID, played, percentcumsum, actions, PlayerID) %>%
  ungroup() %>%
  group_by(PlayerID,actions) %>%
  do(calc.pxfx(.))
```

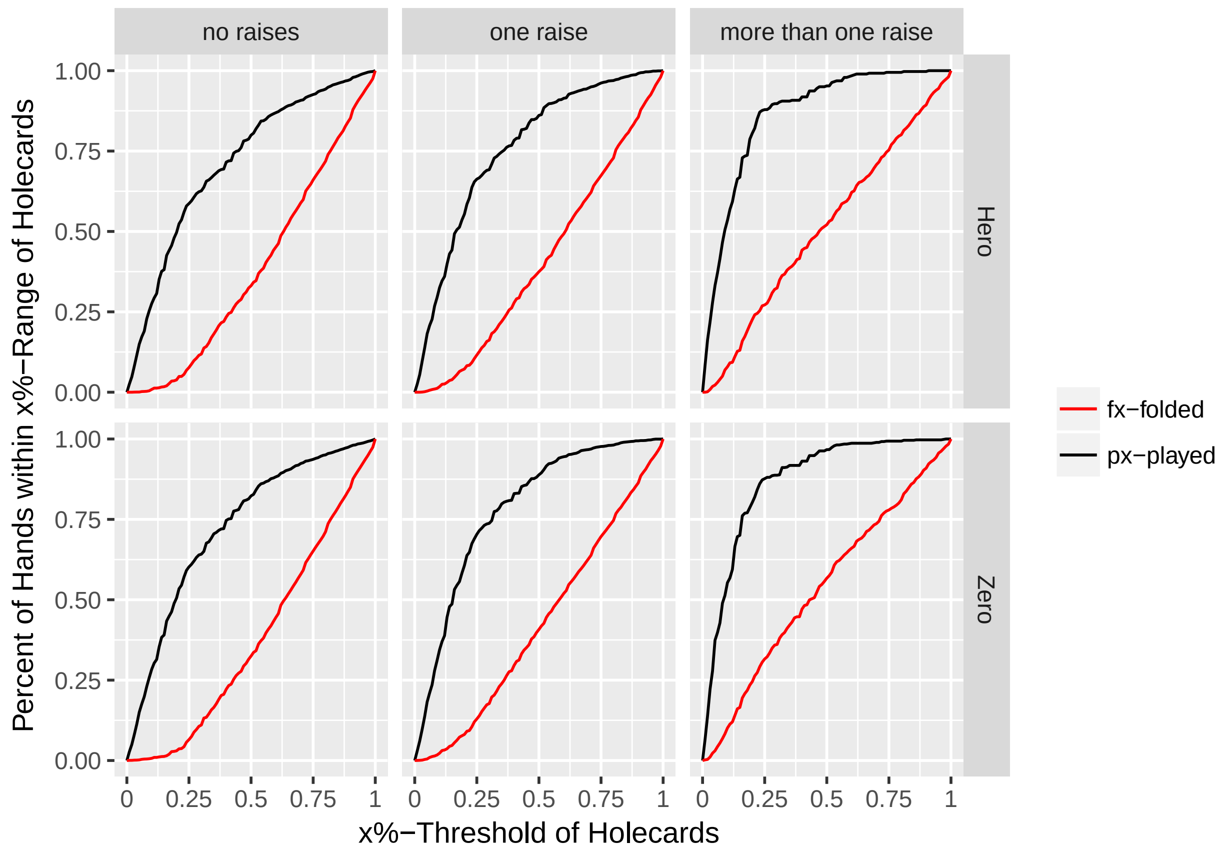


Figure 7.13: px and fx by Player and Number of Raises

We can see a very high area under curves for Holecards played against more than one raise. This nearly rectangular shape is what we predicted initially for a very restricted player. When we look from left to right we can see that fx goes steaper as the selection get more selective, this leads to a reverse in shape for fx against multiple raises. fx is now concave instead of convex, a behavior which we have never seen before, even for the early position raises. The fact that this behavior occurs for both players decreases the chance that this pattern occured by chance.

```
Ao_by_raises <-  Holecards.act %>%
  select(GameID,NoPlayers,played,percentcumsum, raises,PlayerID) %>%
  filter(NoPlayers>=4)%>%
  mutate(actions=raises,
         actions=ifelse(actions==0,"no raises",
                 ifelse(actions<=1,"one raise", "more than one raise")),
         actions= factor(actions,
                         levels=c("no raises","one raise","more than one raise"))) %>%
  ungroup() %>%
    group_by(PlayerID,actions) %>%
      summarise(Ao=mean(played))

maxima_by_raises <- pxfx_by_raises %>%
    mutate(dx=px-fx) %>%
    group_by(PlayerID,actions) %>%
    filter(dx==max(dx)) %>%
    select(mwd=x_threshold)
```

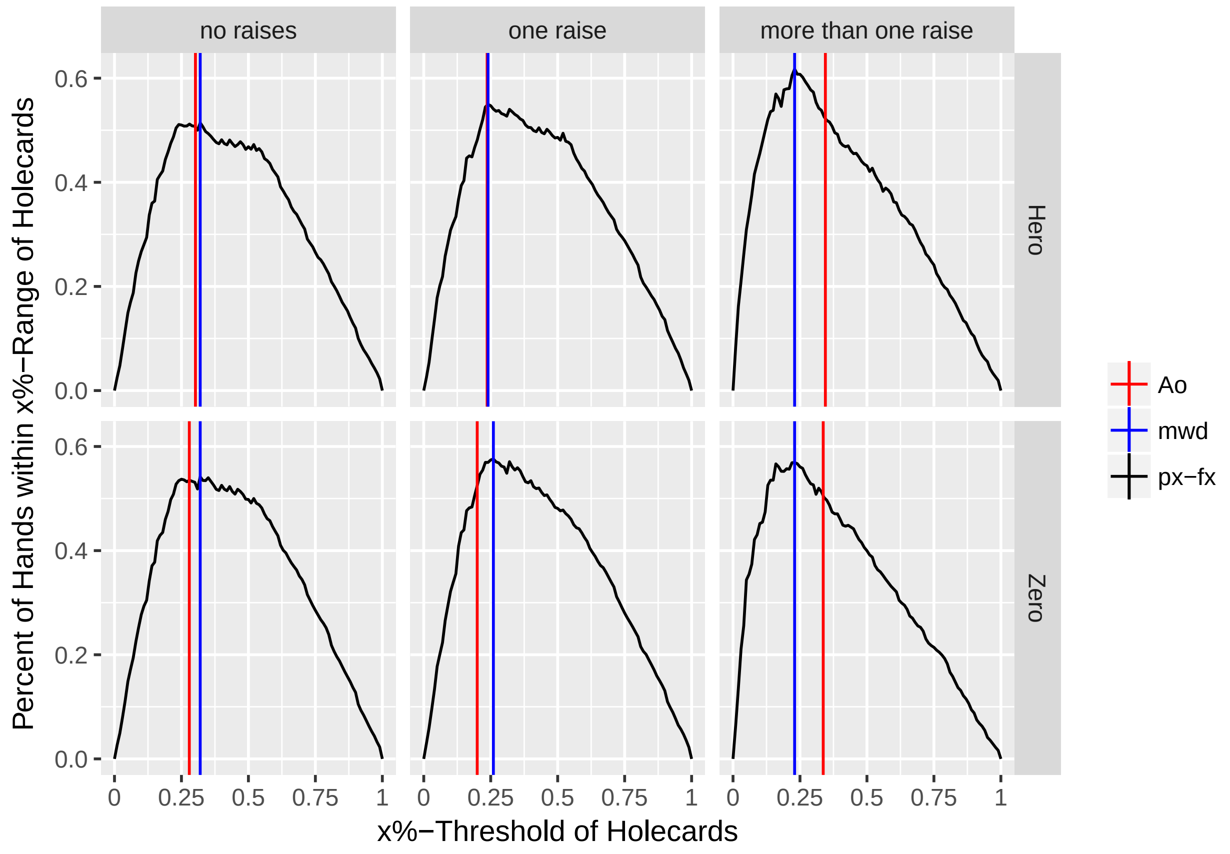


Figure 7.14: px-fx and mwd by Player and Number of Raises

Both, the bavavior against calls and against raises are biased metrics. Both fail to model the game state. Imagine you have call and raise counts of value 3: The situation a player finds himself in can vary a lot. Imagine a 3 player hand goes like this: raise, raise, raise, call, call, call.
Or the same hand goes call, call, raise, fold, raise, raise, call.
these are two complete different game states. “Raise, raise, raise, call, call, call” in a 3 player game is totally different as the same pattern occuring in a 6 player game.
But why do we see such clear patterns in the data? The number of calls and the number of raises converge into a metric called pot-odds. The next final subchapter introduces this metric and investigates Holecard selection against it.

7.5.4 Holecard Selection against Pot Odds

Pot-Odds are the ratio of money you have to spend (to call) against the amount of money you can win (the pot). You can use Pot-Odds to evaluate if a call is mathematically reasonable. Here is a example: You have a flush draw (you need one of the remaining cards of a suite to have 5 of them) and you are on the flop. In this situation you have 9 cards remaining in the deck which can help you win the hand, and you have two cards to come (turn and river). The chance of getting the flush is given by $1 - 38/47 * 37/46 = .35$ the odds are $1/.35 - 1 = 1.85to1$.
Let us assume an opponent raises all-in for 300 chips on a pot of 600 chips. We need to pay 300 to win 900 which is 3-to-1 compared to our 1.85-to-1 win chance. This is mathematically a good choice to call. If he would have raised 1200 on a pot of 600, we need to pay 1200 to win 1800, 1.5-to-1, which compared to our winrate 1.85-to-1 is a bad mathematical choice, because statistically we would lose more than we gain, if we play out this situation over and over again.

A raise increases the pot-size but also the amount one has to pay to call. A call increases the pot-size but does not increase the amount to call. Therefore it is quite appealing that the more raises there are, the worse our Pot-Odds get and the more restricted we have to be. The more calls, the better the odds, the looser our hand selection. To calculate the Pot-Odds for our players we need three numbers:

- the amount of money in the pot
- the amount of money already put into the pot by our player
- the betsize

```
Logdata <- Logdata %>%
  mutate(index=row_number()) %>%
  group_by(index) %>%
  mutate(raisesize=ifelse(action.type=="raise",
                          sub("raises (\\$)?((\\d|\\.)+) .*","\\2",action),""))
 pot.odds <- Logdata %>%
  group_by(GameID=PSID) %>%
  filter(Round=="Pre-Flop") %>%
  mutate(rownum=row_number(),
         cutoff=ifelse(PlayerID %in% c("Hero","Zero"),rownum,0)) %>%
  filter(rownum<=max(cutoff)) %>% ungroup() %>%
   mutate(bb=ifelse(relPos==NoPlayers,1,0),
          sb=ifelse(relPos==NoPlayers-1,1,0),
          betsize=as.numeric(betsize),
          raisesize=as.numeric(raisesize))
 Blindsizes <- pot.odds %>%
   group_by(PSID) %>%
   filter(row_number()==1) %>%
   mutate(blindsize=ifelse(action.type!="raise",betsize,betsize-raisesize)) %>%
   select(GameID=PSID,blindsize)
 pot.odds <- pot.odds %>%
   left_join(Blindsizes, by="GameID")

 get.commitment <- function(data){
   commitment <-0
   if(data[1,"bb"]==1){commitment <- data[1,"blindsize"]}

   if(data[1,"sb"]==1){commitment <- data[1,"blindsize"]/2}

   if(nrow(data)==1 &&
      data[nrow(data),"PlayerID"] %in% c("Hero","Zero") &&
      data[1,"bb"]!=1 && data[1,"sb"]!=1) {commitment <- 0}
   if(nrow(data)==1 &&
      data$action.type %in% c("call","raise","check") &&
      !data[nrow(data),"PlayerID"] %in% c("Hero","Zero")) {commitment <- data[1,"betsize"]}

   if(nrow(data)>1 &&
      data[nrow(data),"action.type"] %in% c("call","raise","check") &&
      data[nrow(data),"PlayerID"] %in% c("Hero","Zero")){
     commitment  <- data[nrow(data)-1,"betsize"]}

   if(nrow(data)>1 &&
      data[nrow(data),"action.type"] %in% c("call","raise","check") &&
      !data[nrow(data),"PlayerID"] %in% c("Hero","Zero")){
     commitment  <- data[nrow(data),"betsize"]}
   if(nrow(data)>1 &&
      data[nrow(data),"action.type"]=="fold"){commitment  <- data[nrow(data)-1,"betsize"]}

   return(as.data.frame(commitment))
 }


get.maxbet <- function(data){
  res<-array()
 for(i in 1:max(data[,"rownum"])){
   res[i] <- max(data[1:i,"betsize"])
 }
  return(cbind.data.frame(data$rownum,res))
}

betsizes <- pot.odds %>% group_by(GameID) %>% do(get.maxbet(.))
colnames(betsizes) <- c("GameID","rownum","betsize")

commited <- pot.odds %>%
  select(-betsize) %>%
  left_join(betsizes,by=c("GameID","rownum"))

commited <-  commited %>%
  group_by(GameID, PlayerID) %>%
  do(get.commitment(.)) %>%
  mutate(commitment=ifelse(is.na(commitment),max(betsize),commitment),
         commitment=ifelse(is.na(commitment),max(blindsize),commitment)) %>%
  select(commitment)
```

The chunk above calculates a data.frame `commited` which contains the GameID, PlayerID and the amount put in the pot at the moment when our player has to act. The sum of those commited values equals the amount of money in the pot, while the maximum of those numbers equals the highest bet. With the given numbers we can calculate the Pot-Odds.

In many cases our player has to act before the Blinds, therefore they are not included in the pot calculation.`unobserved_blinds` is a data.frame which contains information about if the Blinds are already included or not and gives a measurement "addtopot", which equals the Blinds which are not observed yet. We will normalise bet- and potsizes by deviding them by the Big Blind size.

```
unobserved_blinds <- pot.odds %>%
  group_by(GameID) %>%
  summarise(nobb=sum(bb)==0,
            nosb=sum(sb)==0,
            addtopot=as.numeric(nobb)+.5*as.numeric(nosb),
            blindsize=max(blindsize)) %>%
  select(GameID,addtopot,blindsize)
```

With this information we can successfully calculate the potsize, betsize, our players commitment and finally the odds.

```
odds <- commited %>%
  ungroup() %>%
  left_join(unobserved_blinds,by="GameID") %>%
  mutate(commitment=commitment/blindsize) %>% #normalise the commitment to bigblinds.
  group_by(GameID) %>%
  summarise(potsize=sum(commitment)+max(addtopot), #pot+unobserved, yet placed blinds
            betsize=max(commitment,1), #the current highest bet
            players.commitment=max(commitment[PlayerID %in% c("Hero","Zero")]),
            #already placed money
            odds.to.one=1/((betsize-players.commitment)/potsize)-1,odds.to.one=ifelse(is.nan(odds.to.one
            ) %>%
  ungroup()
```

Table 7.4: Calculated Odds by Game (first 6 lines)

GameID	potsize	betsize	players.commitment	odds.to.one
146669522709	5.0	3.5	0.0	0.4285714
146669552852	1.5	1.0	0.0	0.5000000
146669566568	1.5	1.0	0.0	0.5000000
146669578634	2.5	1.0	1.0	Inf
146669596721	4.0	2.5	0.5	1.0000000
146669604605	1.5	1.0	0.0	0.5000000

We can now add this information to our `Holecards` data.frame and visualise the hand selection according to the Pot Odds. Odds become infinite when the player can check.

```
tmp <- Holecards.act %>%
  ungroup() %>%
  left_join(odds,by="GameID") %>%
  mutate(odd.category=odds.to.one,
        odd.category=ifelse(odds.to.one==Inf,"free check",
                             ifelse(odds.to.one<=1,"bad odds (<=1)",
                                    ifelse(odds.to.one<=2,"good odds (>1, <=2)",
                                           "great odds (>2)"))),
        odd.category= factor(odd.category,
                              levels=c("bad odds (<=1)",
                                       "good odds (>1, <=2)",
                                       "great odds (>2)","free check"))) %>%
  group_by(Symbol,odd.category,PlayerID) %>%
  summarise(Playrate=mean(played),
            highcard=ifelse(max(suit)=="o",max(high),max(kicker)),
            kickercard=ifelse(max(suit)=="o",max(kicker),max(high)))%>%
  ungroup() %>%
  mutate(highcard=factor(highcard,levels=values),
                          kickercard=factor(kickercard,levels=values))
```

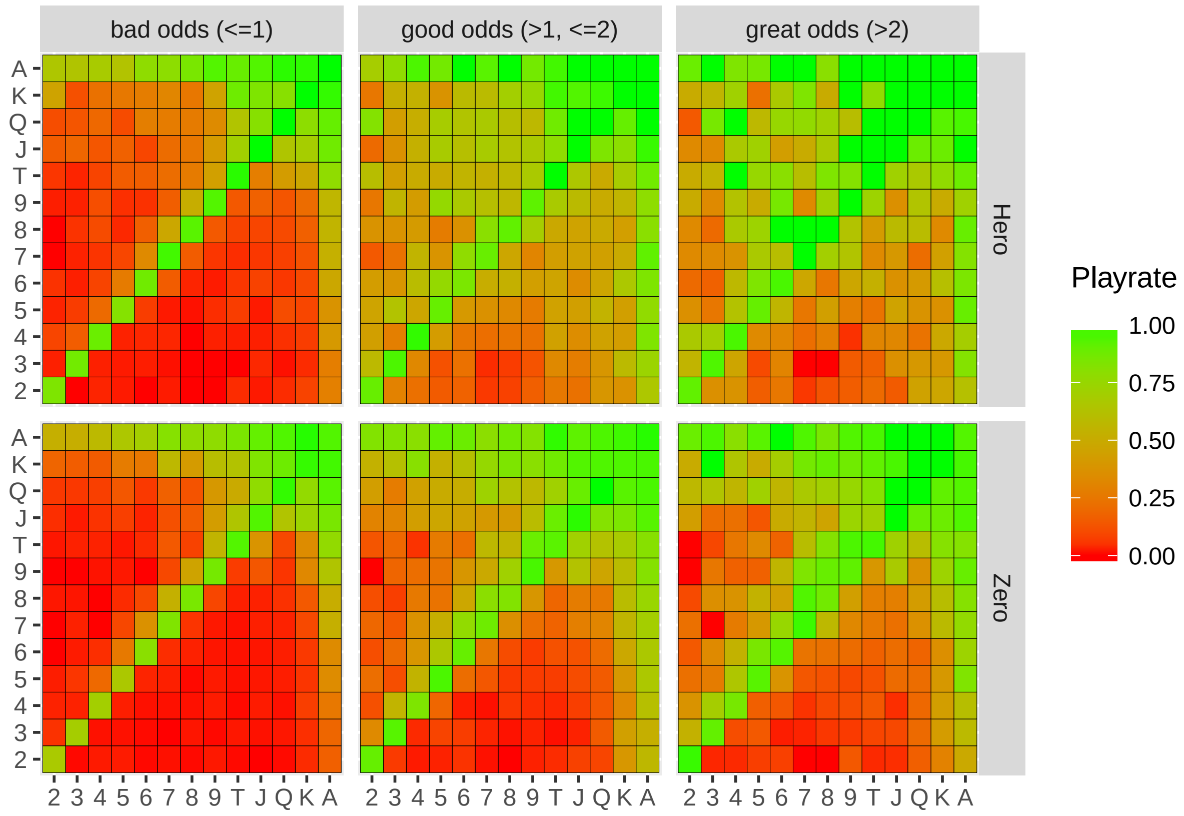


Figure 7.15: Holecards Playrate Distribution Heatmap by Player and Pot-Odds

Even though Pot-Odds are a good representation of the state of a game, it doesn´t model pre-flop situations perfectly:

- In early positions, the odds are bad most of the time because there is no money other than the Blinds in the pot.
- Sometimes Odds can be good but the amount to call them would put the player at risk (e.g. when playing a tournament).

Overall Pot-Odds are more useful when calculated in later situations of the game (e.g. post-flop to varify if you call with your flush draw). Later in this paper we will discuss what the effect of being in the Blinds has on the Pot-Odds.

```
pxfx_by_odds <- Holecards.act %>%
  ungroup() %>%
  left_join(odds,by="GameID") %>%
  mutate(odd.category=odds.to.one,
         odd.category=ifelse(odds.to.one==Inf,"free check",
                             ifelse(odds.to.one<=1,"bad odds (<=1)",
                                    ifelse(odds.to.one<=2,"good odds (>1, <=2)",
                                              "great odds (>2)"))),
         odd.category= factor(odd.category,
                                    levels=c("bad odds (<=1)",
                                             "good odds (>1, <=2)",
                                             "great odds (>2)",
                                             "free check"))) %>%
  select(GameID,NoPlayers,played,percentcumsum, odd.category, PlayerID) %>%
  ungroup() %>%
  group_by(PlayerID,odd.category) %>%
  do(calc.pxfx(.))
```

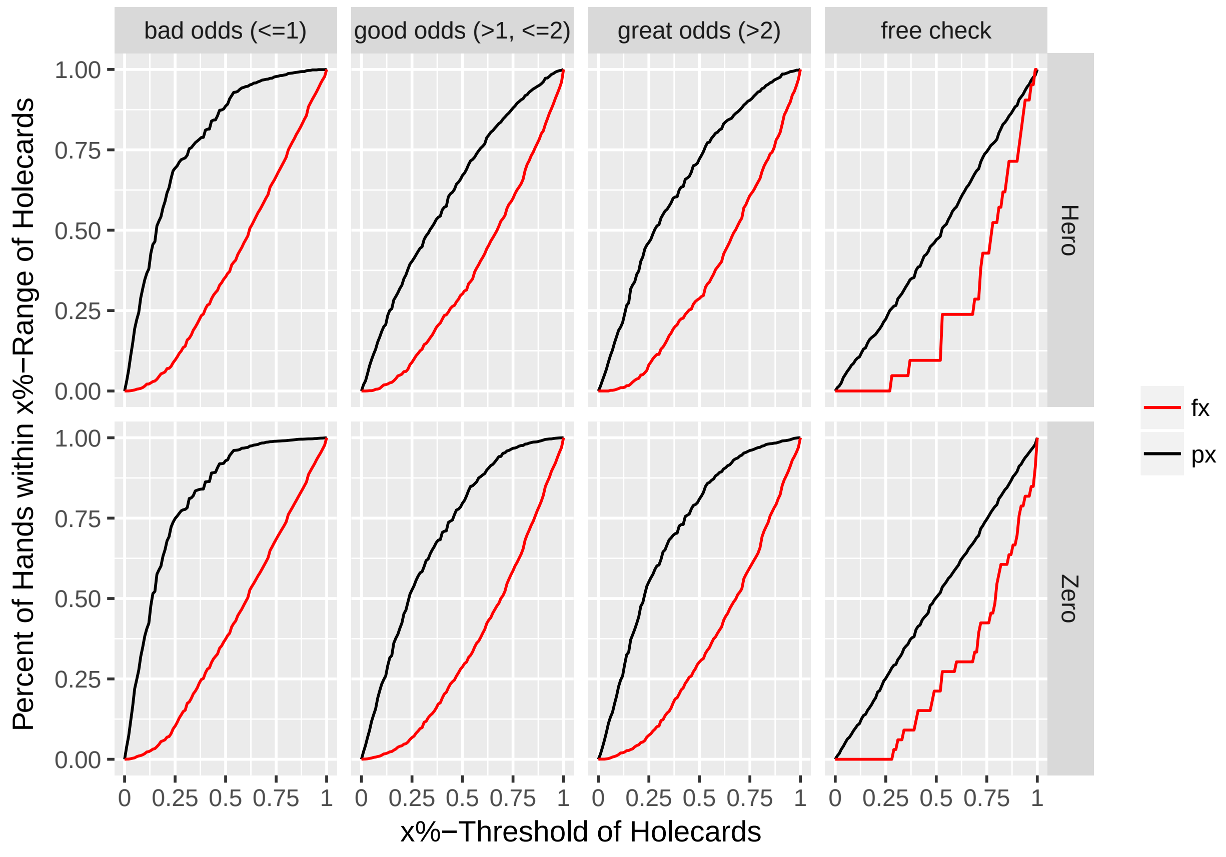


Figure 7.16: px and fx by Player and Pot-Odds

Finally we can observe a diagonal px, which we proposed theoretically earlier: If we can play a hand for free, the Holecards are random. The fx line at free checks are Holecards which have been folded even though a free check was possible. 54 hands have been folded this way.
We can observe our players loosening up, when the pot-odds are getting better, yet the difference between good and great odds is barely visible. This may come back to the point earlier, that Pot-Odds are not the best representation of a game state, or have a limited influence on the Range of a player. e.g. it does not matter if the odds are good or great, a player is not starting to play 72o from out of position.

```
Ao_by_odds <-  Holecards.act %>%
  ungroup() %>%
  left_join(odds,by="GameID") %>%
  mutate(odd.category=odds.to.one,
         odd.category=ifelse(odds.to.one==Inf,"free check",
                             ifelse(odds.to.one<=1,"bad odds (<=1)",
                                    ifelse(odds.to.one<=2,"good odds (>1, <=2)",
                                           "great odds (>2)"))),
         odd.category= factor(odd.category,
                              levels=c("bad odds (<=1)",
                                       "good odds (>1, <=2)",
                                       "great odds (>2)",
                                       "free check"))) %>%
  select(GameID,NoPlayers,played,percentcumsum, odd.category, PlayerID) %>%
      group_by(PlayerID,odd.category) %>%
      summarise(Ao=mean(played))

maxima_by_odds <- pxfx_by_odds %>%
    mutate(dx=px-fx) %>%
    group_by(PlayerID,odd.category) %>%
    filter(dx==max(dx)) %>%
    select(mwd=x_threshold)
```

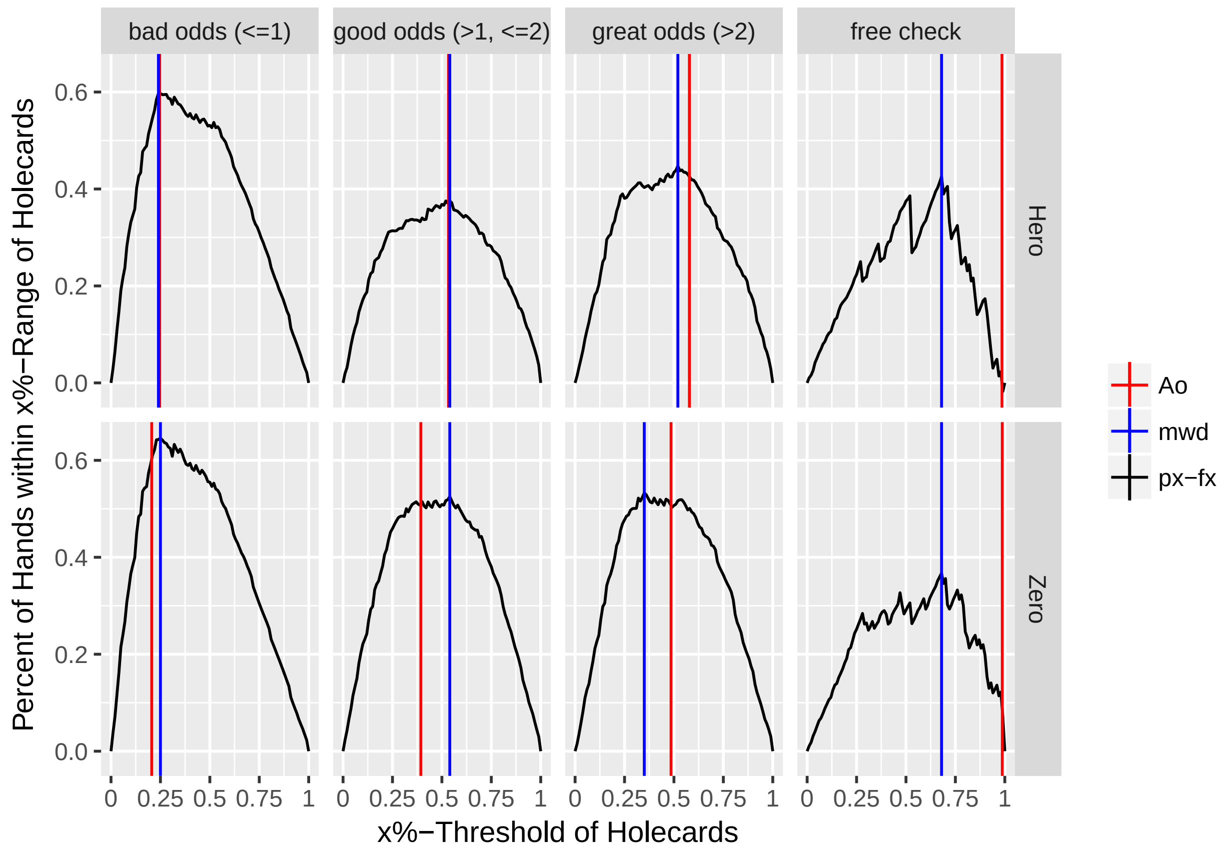


Figure 7.17: px-fx and mwd by Player and Pot-Odds

We can observe expected bahavior: There is a right shift of mwd as the odds get better, yet the gap b is hard to interpret.

Pot-Odds are a very situational tool, they are helpful to evaluate the mathematical correctness of a play, yet it is obviously not easy to generalise over Pot-Odds and interpret them.

Chapter 8

Deriving M analytically

In the previous chapters we discussed different scenarios in which mwd seems to be a valid alternative for the average playrate when it comes to Range identification. Since it it not possible to calculate mwd for every player (because we have to deal with imperfect information), this chapter is focussing on approximating mwd analytically from observable data (and call it "M" instead of "mwd" to make the difference clear). To do this we start with the assumption that we have gathered information about the average playrate (A) on a table of a specific number of players (o). Example: An average playrate of .23 on a 10-Player table.

8.1 Number of Players

The Big Blind is a handy unit to compare games of different Blind sizes. On every game the Small Blind is half of the Big Blind and every following bet has to be of a size which is at least 1 Big Blind (bb). Getting a hand dealt costs the player money, which sums up to 1.5 bb for c hands, where c is the number of players on the current table. The cost per hand decreases monochronically $\frac{1.5}{c}$. We can conclude that the R should change in a similar manner, so that there is a negative or positive differential when deriving from the observed number of players (o) or the current number of players (c).

$(\frac{1.5}{c} - \frac{1.5}{o})$ denotes the difference of costs-per-hand when going from o to c players on the table. Now we can use this differential as a non-relative reweighting factor on our reversed observed average playrate (1-Ao), and can approximate the current average playrate Ac from it. Therefore if we go down in number of players, we will increase the estimate for Ac and vice versa. We use the reversed probability instead of weighting Ao directly so Ac stays in the boundary of $0 \leq Ac \leq 1$.

$$Ac \approx \left(\frac{1.5}{c} - \frac{1.5}{o}\right) * (1 - Ao) + Ao \qquad (8.1)$$

In online poker you often face the same opponents on different sized tables. It would be misleading to think that $Ao = Ac$ when $o \neq c$. e.g. believe that an observed A on a 10 player table (Ao=.23) is true for a 5 player table. The above formula allows one to recalculate an approximated A for the current table, by inserting A, o and c in the formula, which gives an aproximated Ac for our example of 0.3455, which would be a reasonable adjustment to the smaller amount of players and the higher Blind frequency.
The formula works even if $Po < Pc$, let us assume $o = 6$ and $c = 8$ and an observed $A = 0.28$ which leads to an Ac of 0.235.

```
Ao=.28;o=6;c=8
(1.5/c-1.5/o)*(1-Ao)+Ao
```

```
## [1] 0.235
```

Additionally when $o = c$, the formula returns $Ac = Ao$, because there is no adjustment needed.
On the following chunk we calculate Ac based on our observation on a 10-player table $Ao|(o = 10)$. We compare this Ac with the actual observed playrates and check for correlation.

```
tmp.cards <- Holecards.pos %>%
  select(GameID,played,high,kicker,Symbol,suit,percentcumsum, NoPlayers,PlayerID) %>%
  group_by(GameID) %>%
  mutate(NoPlayers=factor(NoPlayers,levels=2:9)) %>%
  ungroup()

tmp.df1 <- as.data.frame(matrix(0,
                                ncol=length(2:9),
                                nrow=length(seq(0.,1,0.01)),
                                dimnames = list(1:101,levels(tmp.cards$NoPlayers))))
tmp.df2 <- as.data.frame(matrix(0,
                                ncol=length(2:9),
                                nrow=length(seq(0.,1,0.01)),
                                dimnames = list(1:101,levels(tmp.cards$NoPlayers))))

for(i in levels(tmp.cards$NoPlayers)){
  tmp.df1[,i]<- unlist(sapply(seq(0,1,0.01),
             FUN=function(x){tmp.cards %>%
                 filter(played==1 & NoPlayers==i &PlayerID=="Hero") %>%
                 summarise(m=mean(percentcumsum<=x))}
             )) # for played hands, how many are in the top 20%

  tmp.df2[,i]<- unlist(sapply(seq(0,1,0.01),
              FUN=function(x){tmp.cards %>%
                  filter(played==0 & NoPlayers==i &PlayerID=="Hero") %>%
                  summarise(m=mean(percentcumsum<=x))}
              )) # for not played cards, how many are in the top 20%
}

mwd <- array()
Ao <- array()
for(i in levels(tmp.cards$NoPlayers)[-1]){
  mwd <- seq(0,1,0.01)[which.max(tmp.df1[,i]-tmp.df2[,i])]
  Ao[i] <- tmp.cards %>%
    filter(NoPlayers==i) %>%
    summarise(A=mean(played)) %>%
    as.matrix() %>%
    as.array()
}

Ac<- (1.5/3:10-0.15)*(1-Ao[8])+Ao[8] #calc based on Ao|o=9
cor.test(Ac,Ao)
```

```
##
##  Pearson's product-moment correlation
##
## data:  Ac and Ao
## t = 5.8222, df = 5, p-value = 0.002111
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.607603 0.990360
## sample estimates:
##       cor
## 0.9335196
```

```
Abytablesize <- Ao #for later use
```

Table 8.1: Calculated Ac based on o=10 and actual Ao by Number of Opponents

	Ac	Ao
3	0.4082334	0.5146347
4	0.3509657	0.3255573
5	0.3127872	0.3078901
6	0.2855168	0.2881399
7	0.2650641	0.2441558
8	0.2491564	0.2568127
9	0.2364302	0.2364302

We can see that Ac and Ao have a high correlation and significance. Given there is no observed playrate for a table of a particular size, Ac is a valid alternative which can be calculated from any given observations.

8.2 Position

The relative position of a player influences the hand selection in a way that for each player who is still to come, the uncertainty of events increases. Therefore, at the first position, R is smallest and on late position R is largest. One exception are the Blind positions, in which players were forced to put in money even before they have seen their cards. As described in previous chapters, this leads to a better price for calling or enables checking (a free flop).

When we define position as the number of players still to act (a), we would find that the Blinds are the actual late positions (all players acted before them at least once).
In poker context we would look at position as if we were on the flop or later: After the first betting round the player left from the dealer starts the action (which is the player on the Small Blind). In this mindset the Blinds are not the latest, but the first positions. As we have observed before, the playrate goes up in the Blinds.
Therefore the effect of the already placed Blinds must overcome the negative effect of position manyfold. It is reasonable to continue with a metric "players still to act (a)", because as seen empirically, the average playrate follows exactly this kind of a distribution (R becomes bigger when a goes down).

Based on our observation of the average playrate Ao and our ability to derive Ac, we can see this playrate Ac as the average playrate above all positions on a given tablesize. The dealer and positions rotate after each hand, and even though players can be moved from and onto the table or sit out, in a short period of time all positions are played equally often from. With this given, we can think of Ac (or Ao) as the playrate on the average position. We can calculate the average position as: $\Sigma[(1:c)]/c$ where c denotes the number of players on the current table. For all possible positions on two to ten player tables, we can calculate the average position as: $\Sigma[(1:c)]/c = c/2 + 0.5$

In a first step we follow $\frac{1}{(9:1)}$ as a distribution and calculate the difference to our average position: $\frac{1}{(c/2+.5)}$. We need to add a 1 to a and c, so the equation can not become infinity. We add a 1 to the difference in playrates to create a weighting factor. We can now multiply the weighting factor directly with Ac to obtain the Apc (the average playrate at position and current number of players).

$$Apc \approx \left(\left(\frac{1}{a+1} - \frac{1}{(\frac{c+1}{2}+.5)}\right)+1\right) * Ac \tag{8.2}$$

For example, we assume Ac to be .23, c is 8, so the array over all Positions is 0.21, 0.22, 0.22, 0.23, 0.24, 0.26, 0.3, 0.41. We can see that around the average position, Apc~Ac. The later the position, the more hands the player is willing to play.

```
c=8;Ac=.23;a=7:0
((1/(a+1)-1/((c+1)/2+.5))+1)*Ac
```

```
## [1] 0.2127500 0.2168571 0.2223333 0.2300000 0.2415000 0.2606667 0.2990000
## [8] 0.4140000
```

In the next chunk we compare our approximated Apc for a ten player table with the actually observed playrates among all positions.

```
tmp.cards <- Holecards.pos %>%
  filter(NoPlayers==9) %>%
  select(GameID,played,percentcumsum,PlayerID, relPos) %>%
  group_by(GameID) %>%
  mutate(relPos=factor(relPos,levels=1:9)) %>%
  ungroup()



tmp.df1 <- as.data.frame(matrix(0,
                                ncol=length(1:9),
                                nrow=length(seq(0.,1,0.01)),
                                dimnames = list(1:101,levels(tmp.cards$relPos))))
tmp.df2 <- as.data.frame(matrix(0,
                                ncol=length(1:9),
                                nrow=length(seq(0.,1,0.01)),
                                dimnames = list(1:101,levels(tmp.cards$relPos))))

for(i in levels(tmp.cards$relPos)){
  tmp.df1[,i]<- unlist(sapply(seq(0,1,0.01),
               FUN=function(x){tmp.cards %>%
                   filter(played==1 & relPos==i & PlayerID=="Hero") %>%
                   summarise(m=mean(percentcumsum<=x))}
               )) # for played hands, how many are in the top 20%

  tmp.df2[,i]<- unlist(sapply(seq(0,1,0.01),
                FUN=function(x){tmp.cards %>%
                    filter(played==0 & relPos==i & PlayerID=="Hero") %>%
                    summarise(m=mean(percentcumsum<=x))}
                )) # for not played cards, how many are in the top 20%
}
mwd <- array()
Apo <- array()
for(i in levels(tmp.cards$relPos)){
  mwd[i] <- seq(0,1,0.01)[which.max(tmp.df1[,i]-tmp.df2[,i])]
  Apo[i] <- tmp.cards %>%
    filter(relPos==i) %>%
    summarise(Apo=mean(played)) %>%
    as.matrix() %>%
    as.array()
}

#playrate for Hero on a 10 player table is .23
Ac=0.23;c=10;a=9:0
Apc <- ((1/(a+1)-1/((c+1)/2+.5))+1)*Ac
cor.test(Apc,Apo)
```

```
##
##  Pearson's product-moment correlation
##
## data:  Apc and Apo
## t = 7.5194, df = 7, p-value = 0.0001351
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7474415 0.9882933
## sample estimates:
##       cor
## 0.943311
```

Table 8.2: Apc and Apo by Position on a 10 Player Table

Apc	Apo
0.2172222	0.2376658
0.2204167	0.1906250
0.2245238	0.1759197
0.2300000	0.1643098
0.2376667	0.1758827
0.2491667	0.1872887
0.2683333	0.2262626
0.3066667	0.3240924
0.4216667	0.4484154

We can observe that our function (8.2) can estimate the average playrate per position very well (correlation of .943 and a P-Value of <=0.0002).

Our approximation seems to be a solid representation of the reality. But we know that the Blind hand selection is not just based on the fact that the player is the last to act, the already placed money is also beneficial. Our Apc model fails to separate those two effects. The differentiation will not matter in general thinking, since we can model the reality very well with our easier model. Still, the next chapter tries to identify which portion of the Blind effects are belonging to money or position.

8.3 Effect of Action and Blinds

This chapter´s purpose is to identify why playrates on Blinds go up, even if the position on later betting rounds is a strong deminishing factor. We have seen in our Apc calculation, that "players still to act" (a) is enough to model all positions including the Blinds neatly. For the sake of completeness, we want to separate the two factors which influences hand selection in Blinds:

- the effect of position in the Blinds.
- the effect of the already placed money.

In the pre-flop state of the game, Blinds are in fact the late positions, at least when defining positions as “the numbers of players still to act” and ignoring that Blinds are always out of position at the flop and after. The position itself has a negative on R, therefore already placed money has to have a big impact.

The effect is best illustrated when considering Pot-Odds again. Here is an example: We sit in late position and a player from early position raised 3bb (Big Blinds), every other player folded. We have to invest 3bb in a pot of 4.5bb, which are 1.5-to-1 or 40%. We need to have Holecards which win in 40%+ of the cases to play the hand. If we sit in the Big Blind, we only have to invest 2bb to win 4.5bb (2.25-to-1, 30.7%), our requirement on our hand strength has gone down a significant amount.

For each additional caller before us, the ratio increases to 4.5+3p-to-2 odds. With p=3, so a initial raise and three callers before us, we are at 6.25-to-1 potodds, which requires a 12.9% winchance of our hand, which is a free invitation for every hand. In general this leads to better odds for the Blinds. It is not handy to model this problem with a possible slot for multiple raises and calls, but we can generalize potodds to $P + 1.5 - to - C - b$ where P is the money put into the pot by all players before, C is the amount which has to be called (or the amount of the highest raise minus money put into the pot on previous calls; C<=M) and b is element of (1,.5), 1 if we are on the Big Blind, .5 on the Small Blind respectively.

We denote the difference in Odd-percentages of the Pot-Odds on Blind versus non-Blind positions, *Db*. Given a fixed P, one can show that Db is minimized when P=C, which would be a single raise. If P=C increaes, Db becomes smaller. If P/C is growing, I is growing.The perks of being in the Big Blind loses its significance when P=C becomes big enough.

```r
odds <-expand.grid(pr=1:9,M=2:30) %>%
  # all combinations of 1:9 player remaining in the pot (pr) and Pots sizes until 30
  filter(pr<=M) %>% # pr has to be <= M, if all player just call the bb, pr=M.
  mutate(
    C=M/pr, #amount which has to be called
    Odds_normal=1/(((M+1.5)/C)+1), #odds from outside the blinds
    Odds_small=1/(((M+1.5)/(C-0.5))+1), #odds from small blind
    Odds_big=1/(((M+1.5)/(C-1))+1)) %>% #odds from big blind
  filter(C%%1==0) #filter for C modulus 1 being 0 (C is an integer)
```

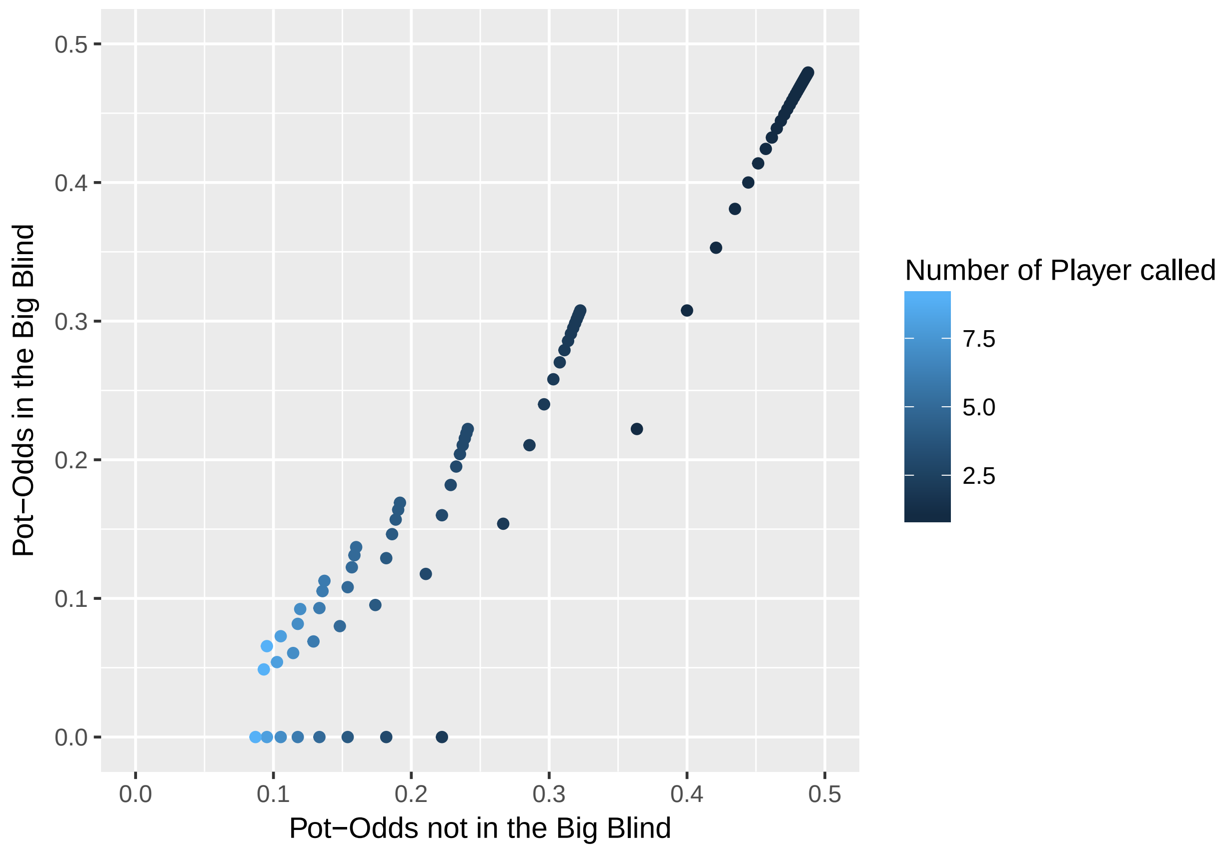


Figure 8.1: Difference in Pot-Odds when having placed a Big Blind

Table 8.3: Average Required Odd-Perccentage Nonblind, SB and BB

Normal_Odds	SB_Odds	BB_Odds
0.2950399	0.2738522	0.2504643

Over all relevant situations one could be in pre flop, the Pot-Odds are 0.07% better when on the small Blind and 0.15% better when on the Big Blind. This strongly depends on the situation the player is facing, but since we want to generalize, one could state that a player in the Blinds should adapt to these numbers.

How can we interpret this in a way that it is coherent with what we have done in the last chapter? Imagine we are at Ac=Ao=.24, that would set the Range to better or equal K6s, which are the top 53 Holecards. K6s has a winrate of 11.6% against 9 random hands. We can lower this threshold by 15% when we are in the Big Blind, which leads to $11.6 * .85 = 9.86$ required winrate, which is 53s or better. 53s upwards are represented by the top 32% winrate.

After this excursion we have developed a deeper understanding on how the Blinds effect the Range selection. To further identify why there are such high playrates would be a great effort for a paper on its own. For this thesis, we only wanted to show that there are a multitude of effects working at the Blinds.

8.4 Deriving M from A

So far we are able to model Apc. But this thesis would not be finished if we cannot make the long run and conclude from Ao to Apc to M and finally to Mpc. In the empirical part of the thesis, we stated formula (7.1) $M \approx Ao - b$: When sitting on a long table, mwd starts to equal Ao, so that we concluded that there is an effectsize b (for “Blind effects”). When we take our data and calculate mwd and Ao without including hands played from the Blinds, we expect a high correlation between mwd and Ao.

```
tmp.cards <- Holecards.pos %>%
  select(GameID,played,high,kicker,Symbol,suit,percentcumsum, NoPlayers,PlayerID,seat) %>%
  filter(seat!="Blinds") %>% #filtering out every hand which was played from the blinds
  group_by(GameID) %>%
  mutate(NoPlayers=factor(NoPlayers,levels=2:9)) %>%
  ungroup()


tmp.df1 <- as.data.frame(matrix(0,
                               ncol=length(2:9),
                               nrow=length(seq(0.,1,0.01)),
                               dimnames = list(1:101,levels(tmp.cards$NoPlayers))))
tmp.df2 <- as.data.frame(matrix(0,
                               ncol=length(2:9),
                               nrow=length(seq(0.,1,0.01)),
                               dimnames = list(1:101,levels(tmp.cards$NoPlayers))))

for(i in levels(tmp.cards$NoPlayers)){
  tmp.df1[,i]<- unlist(sapply(seq(0,1,0.01),
               FUN=function(x){tmp.cards %>%
                   filter(played==1 & NoPlayers==i &PlayerID=="Hero") %>%
                   summarise(m=mean(percentcumsum<=x))}
               )) # for played hands, how many are in the top 20%

  tmp.df2[,i]<- unlist(sapply(seq(0,1,0.01),
                FUN=function(x){tmp.cards %>%
                    filter(played==0 & NoPlayers==i &PlayerID=="Hero") %>%
                    summarise(m=mean(percentcumsum<=x))}
                )) # for not played cards, how many are in the top 20%
}

tmp.df1 <- tmp.df1[-1,-1]
tmp.df2 <- tmp.df2[-1,-1]

mwdnb <- array()
Anb <- array()
for(i in levels(tmp.cards$NoPlayers)[-1]){
  mwdnb[i] <- seq(0,1,0.01)[which.max(tmp.df1[,i]-tmp.df2[,i])]
  Anb[i] <- tmp.cards %>%
    filter(NoPlayers==i) %>%
    summarise(Anb=mean(played)) %>%
    as.matrix() %>%
    as.array()
}
cor.test(mwdnb,Anb)
```

```
##
##  Pearson's product-moment correlation
##
## data:  mwdnb and Anb
## t = 14.514, df = 5, p-value = 2.802e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9200663 0.9983492
## sample estimates:
##       cor
## 0.9883399
```

Table 8.4: mwd and Ao without Hands played from Blinds

	mwdnb	Anb
4	0.31	0.27
5	0.27	0.25
6	0.31	0.25
7	0.23	0.20
8	0.22	0.21
9	0.23	0.20

Without considering hands which have been played from the Blinds, mwd and A are correlated nearly perfectly, while with considering Blinds, they are only correlated on a .75 level. The Blinds are obviously what makes the difference.

8.4.1 Estimating b

Our Ac-formula (8.1) allows us to approximate Ac from Ao. To make the analytical way to approach M easier, we state that $Ac \approx Ao$.

With our Apc-formula (8.2) we have developed a model which can break an observed Playrate Ao down to each position at the table, including the Blinds. With this information we can identify the size of b and should be able to calculate M in a second step.

We know the approximated effectsize of being positioned in the Blinds, it is given by the weighting factor in the Apc formula (8.2),

$$\left(\frac{1}{a+1} - \frac{1}{(\frac{c+1}{2} + .5)}\right) + 1$$

with $a \in (0,1)$ for the Big Blind or Small Blind respectively, and c as the number of players on the table. The sum of both weighting factors w_b (for Small and Big Blind at a given tablesize) can be generalized to:

$$w_b = 3.5 - \frac{1}{(\frac{c+1}{2} + .5)} * 2 \tag{8.3}$$

The combined playrate on Small and Big Blind (Asb + Abb) is given by:

$$(Asb + Abb) = w_b * Ac$$

Given an $Ac = .25$ on a $c = 10$ player table:

```
Ac=.25;c=10
(3.5-(1/((c+1)/2+.5))*2)*Ac # (Asb + Abb) combined Playrate on Blinds
```

```
## [1] 0.7916667
```

Blinds effects are based on the Apc formula (8.2). Our aim is still to calculate M based on Ac minus Blindeffects as stated in (7.1). So we need to know to which extend Ac is increased by b. With our isolated Asb and Abb, we can calculate the "extend of blinds" (eb), to find out the extend of the two Blind playrates over the average playrate.

$$\begin{aligned}
eb &= w_b * Ac - Ac * 2 \\
eb &= (w_b - 2) * Ac \\
eb &= \left((3.5 - 2) - \frac{1}{(\frac{pc+1}{2} + .5)} * 2\right) * Ac \\
eb &= \left((1.5) - \frac{1}{(\frac{pc+1}{2} + .5)} * 2\right) * Ac \qquad (8.4)
\end{aligned}$$

```
Ac=.25; c=10
(1.5-(1/((c+1)/2+.5))*2)*Ac
```

```
## [1] 0.2916667
```

Knowing eb (8.4), we can calculate the blind-adjusted Ac (which we know as M), by substracting $1/(c-2) * b$ from Ac.

$$M \approx Ac - \frac{eb}{c-2} \qquad (8.5)$$

With M given, we can use the Apc-formula (8.2) and replace Ac with M to get the core Range, M, for any given position at a given tablesize.

$$Mpc \approx \left(\left(\frac{1}{a+1} - \frac{1}{(\frac{c+1}{2} + .5)}\right) + 1\right) * M \qquad (8.6)$$

```
Ao=.23;c=7;o=10;a=(c-1):0
Ac=((1.5/c-1.5/o)*(1-Ao)+Ao)
Apc= ((1/(a+1)-1/((c+1)/2+.5))+1)*Ac

M=Ac-(((3.5-(1/((c+1)/2+.5))*2)*Ac - Ac*2)/(c-2))
Mpc= ((1/(a+1)-1/((c+1)/2+.5))+1)*M
tmp <- cbind.data.frame(Value=c(((1/(a+1)-1/((c+1)/2+.5))+1)*((1.5/c-1.5/o)*(1-Ao)+Ao),
                                  Mpc),
                         Ident=c(rep("Apc",c),rep("Mpc",c)),
                         Position=c(c-a,c-a)
  )
```

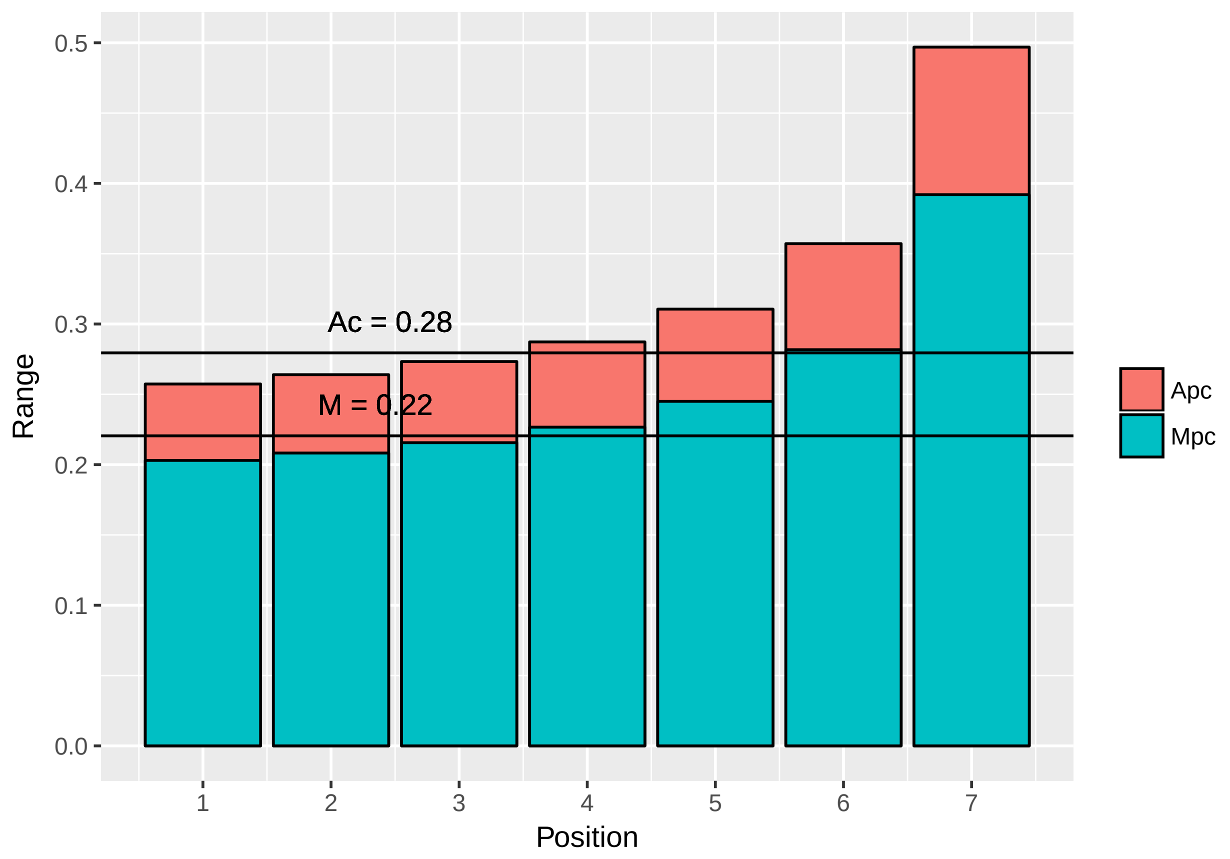


Figure 8.2: Apc, Mpc by Position, Ac and M for a 7-Player Table

As seen in the example above, coming from an observed average playrate Ao=.23 on a o=10 table, we want to calculate effective Ranges (M) on a c=7 player table. We see that Ac>Ao to adjust for the higher Blind frequency. Secondly, we can see Apc=Ac in approximately average position (3.5). Mpc is smaller than Ac when not sitting on the Blinds, which corresponds to the right shift we observed in earlier chapters. Mpc exceeds Ac when sitting in the Blinds, which again, we had observed before.

We can fill in the M formula to get:

$$Mpc \approx \left(\left(\frac{1}{a+1} - \frac{1}{(\frac{c+1}{2}+.5)}\right)+1\right) * \left(Ac - \frac{\left((1.5) - \frac{1}{(\frac{c+1}{2}+.5)} * 2\right) * Ac}{c-2}\right) \tag{8.7}$$

Further we can fit in our AC formula (8.1). Here is a full-formula example: Sitting on a c=10 player table in the Big Blind (a=0), while on a o=5 player table a average playrate of Ao=.35 has been observed.

```
Ao=.35;o=5;c=10;a=0
((1/(a+1)-1/((c+1)/2+.5))+1)*(((1.5/c-1.5/o)*(1-Ao)+Ao)-
                                  (1.5-(1/((c+1)/2+.5))*2)*((1.5/c-1.5/o)*(1-Ao)+Ao)/(c-2))
```

```
## [1] 0.395408
```

An Mpc of .395 suggests a rather tight hand selection, but is a valid approximation.

In a final step we can compare our actual observed mwd on a 9 player table with the results of our Mpc equation (8.7).

```
Ao=Abytablesize[8];o=10;c=10;a=9:0
Mpc=((1/(a+1)-1/((c+1)/2+.5))+1)*
  (((1.5/c-1.5/o)*(1-Ao)+Ao)-(1.5-(1/((c+1)/2+.5))*2)*((1.5/c-1.5/o)*(1-Ao)+Ao)/(c-2))
cor.test(Mpc,mwd)
```

```
##
##  Pearson's product-moment correlation
##
## data:  Mpc and mwd
## t = 6.0794, df = 7, p-value = 0.0005011
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6464961 0.9826590
## sample estimates:
##       cor
## 0.9169314
```

Table 8.5: Mpc and mwd on a 10 Player Table by Position

	Mpc	mwd
	0.1884874	NA
1	0.1907313	0.23
2	0.1935362	0.25
3	0.1971425	0.23
4	0.2019508	0.22
5	0.2086825	0.24
6	0.2187800	0.24
7	0.2356093	0.28
8	0.2692677	0.24
9	0.3702431	0.52

The observed correlation of .91 with a p-value of <=.0006 proves that $M \approx mwd$. Differences in those values come from the fact that M is an analytically derived “optimal” core Range, while we know our players are not playing optimally, e.g. playing too many hands from the Blinds and too many hands in early position.

8.4.2 Application of M and Mpc onto the existing literature

Billings et al. (1998) suggested something similar to a confidence interval around the average playrate. He does it while having full information about the Holecards of the opponent and can calculate the variance, which he called σ. His purpose is to assign weights to cards, how likely it is for a player to hold them.
Translated to our terminology, he suggest a Range with a confidence interval around them, to identify a core-Range (hands better than $R + \sigma$), where the probability approximates 100% that the hand is played. On the lower end of the convidence interval, the probability approximates 0. We derived M analytically to identify a “core Range”, where the willingness to play and fold deviates the most from each other. We could suggest using M as the upper bound convidence intervall. With Ac as the median. $Ac - M = b \approx \sigma$. Therefore one could suggest $Ac \pm b$ as a new interval, without needing to have full information.

Chapter 9

Summary

We explored real world data and found a metric (mwd) which at the first glance was more robust to the Blind effects than the average playrate (A). The metric mwd empirically was derived as “the maximum difference between the willingness to play and to fold Holecards of a given quality”. We further investigated the relationship between mwd and A in specific situations and came to the conclusion that mwd is indeed a valid aproximation for a players core-Range which appears to be free of bias. In a second step we have developed a set of formula which can derive good aproximations from going from the observed playrate on a table sized o to a table sized c (8.1). And a second formula which can approximate the effect of table position onto the playrate (8.2). Finally we found out empirically, that when Blinds are left out $Ac \approx M$. With this information we have been able to compute a model which is able to derive M from any given observed Ao by using information about table sizes (observed and current) (8.5) to estimate the effectsize of playing from the Blinds (8.4). With the calculated M, one is able to use the formula for position effects to extract Mpc (8.7), the core Range at a specific table position. Analytically derived we can call M “a blind-bias adjusted core average playrate”.

9.1 Problems of M and mwd

M only works if the player knows about concepts like hand-strength and position. M gives the optimal playrate for a player´s preferences, which means that M is always dependent on Ao, which is a good and a bad thing at the same time. It is able to adapt to any players playstyle, but fails to converge at a globally valid, specific optimal playrate. mwd is derived from actual playdata while the analytical M is derived from the mindset of an optimising player. Both could differenciate, but in a real world situation, we would never know since we cannot calculate the empirical M, without having perfect information.

9.2 Conclusion

The analytical M is a logic-based approximation of the empirical mwd and adjusts the observed playrate to be resistant against biases of various kinds. M should be used over A as an indicator for the Range of a player, because it can protect against inconsistencies of a player and gives a view of the “core”-playrate of a player.

Acknowledgement

I want to thank my supervisor Prof. Dr. Markus Loecher (Berlin School of Economics and Law) for his trust and support.

I´d want to thank my dear friend "Hero" for providing the data which I used in this thesis.

I want to thank my proofreaders Sara, Marc, Felix, and Anne.

Last and finally I want to thank my family for supporting me on my way, I know this thesis means as much to them as it means to me.
I love you!

This document was created in Rmarkdown (.rmd) using the `bookdown` package (Xie (2017)) as well as the `bibtex` package (Francois (2017)).

References

Analytics, Revolution, and Steve Weston. 2015. *DoParallel: Foreach Parallel Adaptor for the 'Parallel' Package*. https://CRAN.R-project.org/package=doParallel.

Axelle Moreau, Henri Chabrol, and Emeline Chauchard. 2016. "Psychopathology of Online Poker Players: Review of Literature." *Journal of Behavioral Addictions 5*, pp. 155–68.

Billings, Darse, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron. 2002. "The Challenge of Poker." *Artificial Intelligence 134*, 201–40.

Billings, Darse, Denis Papp, Jonathan Schaeffer, and Duane Szafron. 1998. "Opponent Modeling in Poker." In *AAAI-98 Proceedings*. AAAI.

Bowling, Michael, Neil Burch, Michael Johanson, and Oskari Tammelin. 2017. "Heads-up Limit Hold'em Poker Is Solved." *Communications of the ACM* 80.

Dalpasso, Marcello, and Giuseppe Lancia. 2014. "Estimating the Strength of Poker Hands by Integer Linear Programming Techniques." Springer-Verlag.

Ekmekci, Omer, and Volkan Sirin. 2013. "Learning Strategies for Opponent Modeling in Poker." *Computer Poker and Imperfect Information: Papers from the AAAI 2013 Workshop*. AAAI.

Felix, Dinis, and Luís Paulo Reis. 2008. "An Experimental Approach to Online Opponentmodeling in Texas Hold'em Poker." *Advances in Artificial Intelligence* SBIA 2008. Springer: pp. 83–92.

Francois, Romain. 2017. *Bibtex: Bibtex Parser*. https://CRAN.R-project.org/package=bibtex.

Frederic Paik Schoenberg, Yixuan Shao. 2017. *Holdem: Texas Holdem Simulator*. https://CRAN.R-project.org/package=holdem.

Mehrabi, A., and A. T. Haghighat. 2009. "Discovering the Rules in a Poker Player's Mind Based on the Association Rule Mining." In *2009 International Conference on Computer Technology and Development*.

Philipp, Max. 2018. "Max Philipp Github Repository." https://github.com/MPhilippHWR/MaxPhilippPoker.

Ponsen, Marc, Jan Ramon, Tom Croonenborghs, Kurt Driessens, and Karl Tuyls. 2008. "Bayes-Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker." In *Proceedings of the Twenty-Third Aaai Conference on Artificial Intelligence*. AAAI.

Ranca, Razvan. 2013. "Identifying Features for Bluff Detection in No-Limit Texas Hold'em." *Computer Poker and Imperfect Information: Papers from the AAAI 2013 Workshop*. AAAI.

Rinker, Tyler W. 2017. *qdapRegex: Regular Expression Removal, Extraction, and Replacement Tools*. Buffalo, New York: University at Buffalo/SUNY. http://github.com/trinker/qdapRegex.

Rubin, Jonathan, and Ian Watson. 2011. "Computer Poker: A Review." *Artificial Intelligence 175*, pp. 958–87.

Sklansky, David, and Mason Malmuth. 1999. *Holdem Poker for Advanced Players*. Third. Two Plus Two

Pub.

Wickham, Hadley. 2007. "Reshaping Data with the reshape Package." *Journal of Statistical Software* 21 (12): 1–20. http://www.jstatsoft.org/v21/i12/.

———. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. http://ggplot2.org.

———. 2017. *Stringr: Simple, Consistent Wrappers for Common String Operations*. https://CRAN.R-project.org/package=stringr.

Wickham, Hadley, Romain Francois, Lionel Henry, and Kirill Miller. 2017. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Xie, Yihui. 2017. *Bookdown: Authoring Books and Technical Documents with R Markdown*. https://github.com/rstudio/bookdown.