Simulation of Modifying Death Saving Throw Rules

ST 541 Project

Amy Ly

Background

Dungeons and Dragons (D&D) is a roleplaying and adventure game where players come together to create an exciting story. Each player creates a character to represent them and play with the guidance of a game master, who is responsible for narration and setting up scenarios for players to overcome. Gamemasters use a particular set of rules that helps them to narrate and decide on the outcomes of players decision. Whenever there is a potential for failure (such as attacking and enemy or performing a song), players must roll a die (including any bonuses or penalties a character has accumulated) and check that number (DC) against a number set by the game master. If the result is higher than the DC, players will succeed.

Characters who end up with 0 health points will be required to make a death saving throw. A death saving throw is a special saving throw that determines whether the player's character nears closer to death (fail) or life (success). The rules for determining the outcome of a death saving throw are as followed:

- 1 is considered as two failures
- 2 – 9 are considered as one failure
- 10 – 19 are considered as one success
- 20 automatically means you are stabilized and regain one health point

Additionally, death saving throws can be aided or hindered by spells or other features. Aid could come in the form of having advantage, where players could roll twice and then choose the higher value to check against the DC. Alternatively, hindrance could come in the form of having disadvantage, where players would roll twice and then choose the lower value.

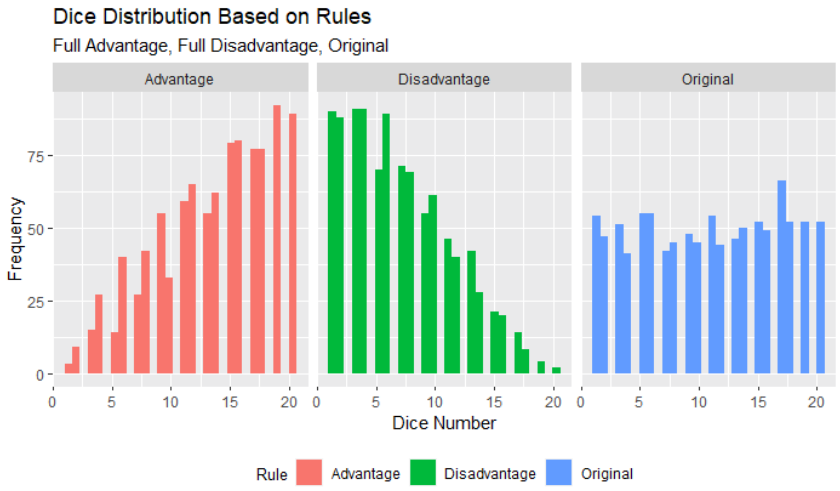
Research Questions

Not all D&D adventures come from official guides. Game masters may be interested in customizing rules or creating their own adventures. For example, a game master may want to introduce a trait that could give players advantage in death saving throws. However, when customizing or adding new rules, game masters must be careful not to unbalance the game. With the original rules, players have about 59.5% chance of survival. What are the chances of survival in the following scenarios?

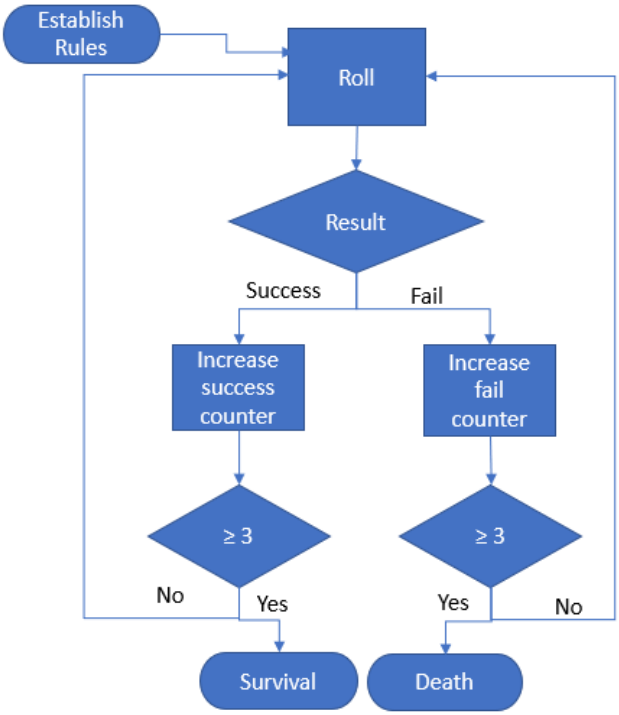
- Allowed an advantage after at least X number of successes
- Given an advantage or disadvantage on all rolls
- Given a disadvantage after at least X number of failures
- What happens if you increase the number of rerolls given in advantage and disadvantage?

Methods

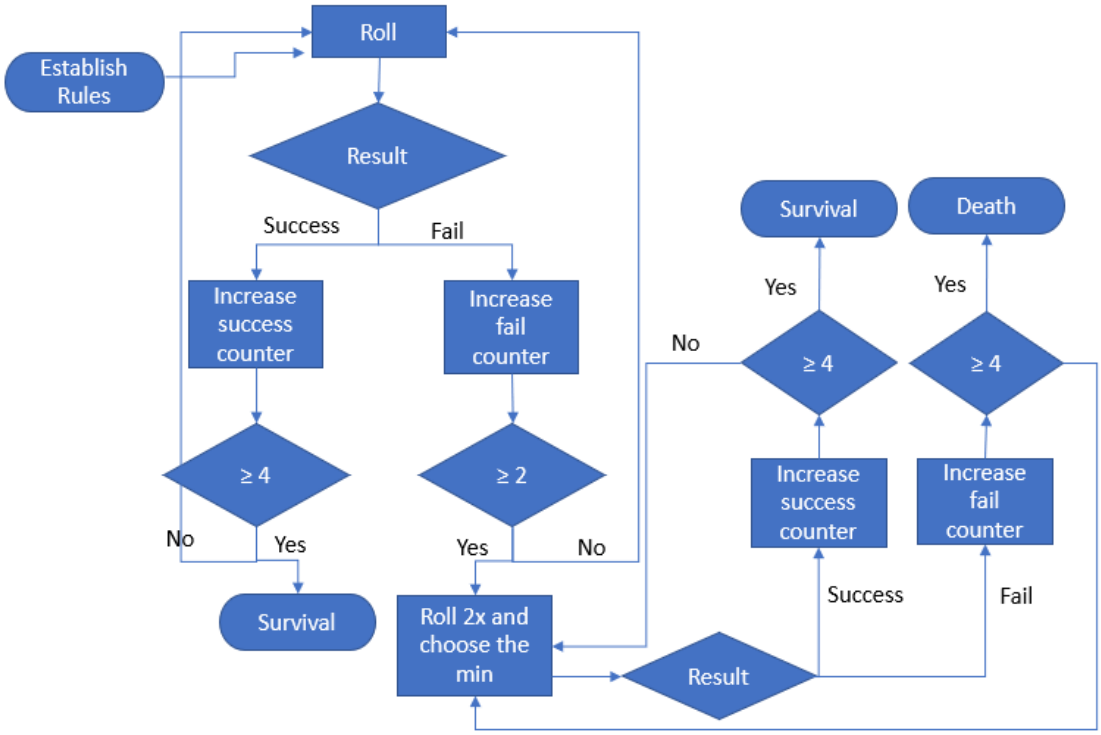
Dice events are independent, and each die is fair. Rolling dice is a discrete distribution. Depending on the rules, the distribution of numbers is approximately as shown below. With the original rules, players should expect an equal probability of rolling any of the numbers.



First, death save scenarios were simulated 1000 times according to the algorithm on the right. For example, I would establish that death saves are performed with advantage from the beginning and that three successful death saves are required for survival. The outcome of each roll will contribute to the success or fail counters. Death saving rolls are simulated until the number of saves are met. For example, the original rules requires that players succeed at least three times to be stabilized. If players instead fail at least three times, then their character dies. Survival and death are the only outcomes. For the ease of calculating survival probability, survival is considered as a 1 and death is considered as 0. To calculate the probability of survival, the mean of survivals is calculated. In addition, the standard deviation is calculated.

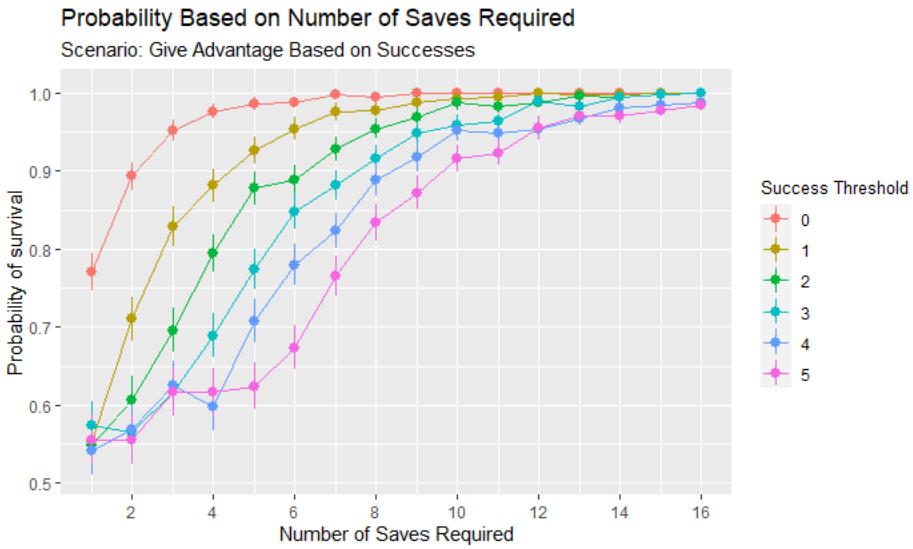


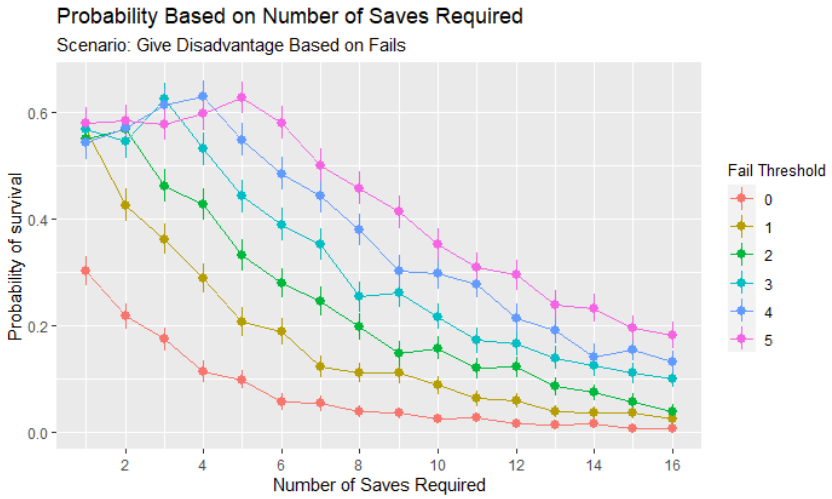
Next, the algorithm for death save scenarios were modified to include a condition for when advantage or disadvantage is given. For example, as shown in the algorithm below, I would first establish that death saves will be performed until there are four successful death saves and that if players have at least two failures, then disadvantage will then be applied to their rolls. Note that the first roll will always be done with original rules. The survival probability and its standard deviation will be calculated in the same way as mentioned before.



Results and Discussion

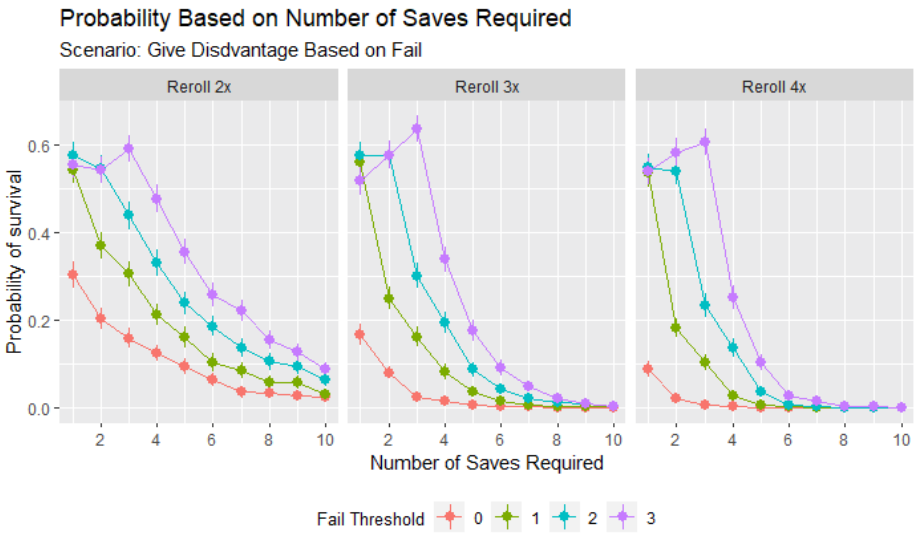
For the sake of the simulation study, I am simulating the probability of survival against the number of saves required. If you provide players with advantage from the beginning, then players will always survive if game masters modify the rules to be six saves or more are required. If you provide players with disadvantage from the beginning, then players chance of surviving seems to decrease exponentially. The probability of survival increases very gradually if the game master chooses to play with the original rules and increase the number of saves required.





If I simulate scenarios where advantage or disadvantage was given conditionally, the general trends are very similar. Note that the first roll will always be done with original rules. As the success threshold increase, survival probability increases more gradually and as the fail threshold increase, survival probability decrease more gradually. The standard error bars are more noticeable as threshold values increase, but the width of the error bars decreases as the number of saves required increase.

Error bars are much smaller in width when the success and fail threshold is at 0 because the outcomes are more deterministic. When advantage or disadvantage is given conditionally, there is more variation in the rolls so that as the number of saves required increase, the standard error bars are wider in comparison. However, as players roll more as the number of saves required increases, the variation in seems to decrease.



It is expected that if players were allowed to reroll more than twice, they would have a higher chance of achieving success or fails given advantage or disadvantage, respectively. The more rerolls allowed for giving disadvantage based on the number of fails, the more the probability of survival decreases.

Conclusion

Realistically, players would not enjoy rolling dice more than they need to. With the current rules of three saves, players could do up to five rolls before surviving or dying. Each time the number of saves required increases, the number of total rolls a player could make increases by two. This is without accounting for the rerolls game masters may provide players in a customized game.

Game masters would be more interested in providing advantages to help players. Based on the simulation results, it is best to only allow two rolls and provide advantage after players achieve two successes. Any higher than those thresholds would result in too much of an increase in survivability, making death saving throws trivial. Any lower than those thresholds would cause players feel as if there were no benefits gained.

Appendix: Important Functions

mod_roll()

This function will modify the roll based on the chosen rule, listed below:

- Original: Roll only once
- Advantage: Roll a number of times and choose the higher of the rolls
- Disadvantage: Roll a number of times and choose the lesser of the rolls

Inputs:

- rule [char]: "original", "advantage", or "disadvantage"
- roll_time [int]: number of times that players can roll before choosing the higher or lesser of the rolls when playing with custom rules

Output:

- roll [int]: a number

```r
# changes the way a roll is performed based on the custom rule chosen.

mod_roll <- function(rule = c("original", "advantage", "disadvantage"), roll_time) {
  rule <- match.arg(rule)

  if (rule == "advantage") {
    roll <- max(sample(1:20, size=roll_time, replace=TRUE)) #advantage
  }
  else if (rule == "disadvantage") {
    roll <- min(sample(1:20, size=roll_time, replace=TRUE)) #disadvantage
  }
  else {
    roll <- sample(1:20, size=1, replace=TRUE) #original
  }
  return(roll)

}
```

roll()

This function will simulate the roll based on several inputs.

Inputs:

- saves [int]: the number of successes or fails required before survival result is evaluated
- rule [char]: "original", "advantage", or "disadvantage"
- status [int]: a threshold number of success or fails before players can gain advantage or disadvantage.
- reason [char]: set "success" or "fail" to be the threshold for judging status
- roll_time [int]: number of times that players can roll before choosing the higher or lesser of the rolls when playing with custom rules

Output:

- survive [int]: 1 if the player has the required number of successes first or 0 if the player has the required number of fails first.

```r
roll <- function(saves, rule, status, reason = c("success", "fail"), roll_time) {
  reason <- match.arg(reason)
  success <- 0 #success counter
  fail <- 0 #failure counter
  status <- status
  if (status == 0 & rule %in% c("advantage", "disadvantage")){
    roll <- mod_roll(rule, roll_time)
  } else {
    roll <- sample(1:20, size=1, replace=TRUE)
  }

  while (success <= saves && fail <= saves) {
    #perform death save throw based on desired rules
    success <- success + sum(result(roll)=="success")
    fail <- fail + sum(result(roll) =="fail")

    if (reason == "success"){
      if (success >= status){
        roll <- mod_roll(rule, roll_time)
      } else {
        roll <- sample(1:20, size=1, replace=TRUE)
      }
    }

    if (reason == "fail"){
      if (fail >= status){
        roll <- mod_roll(rule, roll_time)
      } else {
        roll <- sample(1:20, size=1, replace=TRUE)
      }
    }

    if (success >= saves || fail >= saves) {
      break
    }
  }

  survive <- ifelse(success >=saves, 1, 0)
  return(survive)
}
```

result()

This function assesses whether the current roll is a success or fail (and how many successes or fails) based on the following thresholds.

- Roll a 1: 2 fails
- Roll 2 - 9: 1 fail
- Roll 10 - 19: success
- Roll 20: 3 successes

Input:

- roll [int]: a number randomly generated by a 20-sided die

Possible outputs [char]:

- success

- fail
- c(success, success, success)
- c(fail, fail)

```r
#function assess whether the current roll is a success or fail based on the following thresholds
result <- function(roll) {
  result <- c()

  if (roll<10 && roll >1) {
    result <-c("fail")
  } else if (roll >= 10 && roll <20){
    result <-c("success")
  } else if (roll==1){
    result <- c("fail", "fail")
  } else if (roll ==20) {
    result <- c("success", "success", "success")
  }

  return(result)
}
```

survival_prob()

This function will simulate B number of rolls and return the estimated survival probability and its standard deviation.

Inputs:

- saves [int]: the number of successes or fails required before survival result is evaluated
- B [int]: the desired number of simulated die rolls to make
- rule [char]: "original", "advantage", or "disadvantage"
- status [int]: a threshold number of success or fails before players can gain advantage or disadvantage.
- reason [char]: set "success" or "fail" to be the threshold for judging status
- roll_time [int]: number of times that players can roll before choosing the higher or lesser of the rolls when playing with custom rules

Output [list]:

- prob [num]: estimated survival probability
- sd [num]: standard deviation for the estimated survival probability

```r

survival_prob <- function(saves, B, rule, status, reason, roll_time){

  samp <- rerun(B, roll(saves, rule, status, reason, roll_time))
  prob <- mean(unlist(samp))
  sd <- sd(unlist(samp))
  return(list(prob = prob, sd = sd))

}
```

Appendix: Example of Using the Functions

Example 1:

The scenario of interest is:

- 3 successful death saves are required for a character to survive.
- The death save dice roll will be simulated 1000 times.
- Play with original rules, where there is no advantage or disadvantage.
- Since original rules are being played with, there is no threshold based on success or fail before the gamemaster can give out advantage/disadvantage
- Since original rules are being played with, players can only ever roll at most once.

```
# source all files within the Function folder containing the string '.R'
library(miceadds)
source.all("./R", ".R" )

set.seed(123)

survival_prob <- function(saves, B, rule, status, reason, roll_time){

  samp <- rerun(B, roll(saves, rule, status, reason, roll_time))
  prob <- mean(unlist(samp))
  sd <- sd(unlist(samp))
  return(list(prob = prob, sd = sd))
}


survival_prob(3, 1000, "original", 0, "fail", 1)
```

When the above code is run, you should expect the following outcome:

```
*** source mod_roll.R
*** source result.R
*** source roll.R
*** source survival_prob.R
$prob
[1] 0.606

$sd
[1] 0.4888793
```

Example 2:

I want to compare the trend of survival probabilities when players are under three different scenarios while varying the save threshold:

- Play by original rules
- Have advantage from the beginning
- Have disadvantage from the beginning

```r
library(miceadds)
source.all("./R", ".R")
library(tidyverse)

roll_time <- 2
B <- 1000

saves_list <- seq(from = 1, to = 16, by = 1)

rule_list <- c("original", "advantage", "disadvantage")

simulation_params <- list(
  saves = saves_list, rule = rule_list
)

df <- cross_df(simulation_params)

df_rules <- df %>%
  mutate(
    p_hat = unlist(map2(.x = saves, .y = rule,
                        ~ survival_prob(saves = .x, B, rule = .y, status = 0,
                                        reason = "success", roll_time)$prob)),
    sd = unlist(map2(.x = saves, .y = rule,
                     ~ survival_prob(saves = .x, B, rule = .y, status = 0,
                                     reason = "success", roll_time)$sd)),
    rule = as.factor(rule)
  )


ggplot(df_rules, aes(x=saves, y=p_hat, color = rule))+
  geom_point(size=0.5) +
  geom_line() +
  geom_pointrange(aes(
    ymin = p_hat - 1.96*sd/sqrt(B),
    ymax = p_hat + 1.96*sd/sqrt(B)))+
  scale_x_continuous(breaks=seq(0, 16, 2))+
  ggtitle("Probability Based on Rules",
          subtitle = "Full Advantage, Full Disadvantage, Original")+
  labs(y = "Probability of survival", x = "Number of Saves Required")+
  guides(color = guide_legend(title="Rule"))+
  theme(legend.title = element_text(size=10))
```

When the above code is run, you should expect a ggplot as seen in the main report.

Appendix: Summary Statistic for Conditional Advantage

status	saves	max_sd	prob	min_sd
0	1	0.421043082447957	0.817	0.421043082447957
0	2	0.290577761426155	0.897	0.290577761426155
0	3	0.218054001448145	0.947	0.218054001448145
0	4	0.162164414398774	0.975	0.162164414398774
0	5	0.113330683593826	0.981	0.113330683593826
0	6	0.0944877131056093	0.987	0.0944877131056093
0	7	0.0705689732104696	0.996	0.0705689732104696
0	8	0.0705689732104696	0.996	0.0705689732104696
0	9	0.0446989708829856	1	0.0446989708829856
0	10	0.054717401199198	1	0.054717401199198
0	11	0.0446989708829856	0.999	0.0446989708829856
0	12	0	1	0
0	13	0	0.999	0
0	14	0	1	0
0	15	0	1	0
0	16	0	1	0
1	1	0.497939699104163	0.549	0.497939699104163
1	2	0.45536618051669	0.714	0.45536618051669
1	3	0.394850486775573	0.83	0.394850486775573
1	4	0.315524255098646	0.896	0.315524255098646
1	5	0.253580960056751	0.908	0.253580960056751
1	6	0.207407854282373	0.956	0.207407854282373
1	7	0.186383311320351	0.973	0.186383311320351
1	8	0.129335717138491	0.979	0.129335717138491
1	9	0.117549214450024	0.989	0.117549214450024
1	10	0.0705689732104696	0.989	0.0705689732104696
1	11	0.0772655807586403	0.987	0.0772655807586403
1	12	0	0.999	0
1	13	0.0631505185092568	1	0.0631505185092568
1	14	0.0316227766016838	0.999	0.0316227766016838
1	15	0	1	0
1	16	0	0.999	0
2	1	0.49915945364198	0.535	0.49915945364198
2	2	0.496513368929302	0.561	0.496513368929302
2	3	0.467856698028242	0.688	0.467856698028242
2	4	0.397167145848875	0.79	0.397167145848875
2	5	0.377569276502417	0.854	0.377569276502417
2	6	0.317970289657863	0.874	0.317970289657863
2	7	0.251871871076227	0.931	0.251871871076227
2	8	0.202958597236462	0.946	0.202958597236462
2	9	0.207407854282373	0.972	0.207407854282373
2	10	0.159214789983058	0.983	0.159214789983058
2	11	0.1531255937048	0.987	0.1531255937048
2	12	0.104354635210372	0.991	0.104354635210372
2	13	0.0705689732104696	0.988	0.0705689732104696
2	14	0.0834143750078963	0.996	0.0834143750078963

2	15	0.0834143750078963	1	0.0834143750078963
2	16	0.0446989708829856	0.997	0.0446989708829856
3	1	0.496755237280661	0.542	0.496755237280661
3	2	0.494437527366672	0.558	0.494437527366672
3	3	0.489730370546728	0.602	0.489730370546728
3	4	0.446737702208549	0.711	0.446737702208549
3	5	0.394070160366516	0.778	0.394070160366516
3	6	0.368635174517967	0.835	0.368635174517967
3	7	0.305413289293831	0.883	0.305413289293831
3	8	0.284886223184204	0.906	0.284886223184204
3	9	0.231958627355839	0.938	0.231958627355839
3	10	0.193691808077979	0.951	0.193691808077979
3	11	0.173404351355637	0.971	0.173404351355637
3	12	0.136593036636783	0.987	0.136593036636783
3	13	0.104354635210372	0.988	0.104354635210372
3	14	0.0944877131056093	0.985	0.0944877131056093
3	15	0.0834143750078963	0.998	0.0834143750078963
3	16	0.0891288053546324	0.999	0.0891288053546324
4	1	0.498565509805449	0.55	0.498565509805449
4	2	0.496988931464271	0.549	0.496988931464271
4	3	0.490746198998863	0.604	0.490746198998863
4	4	0.485875575713654	0.604	0.485875575713654
4	5	0.469719742651548	0.691	0.469719742651548
4	6	0.416466618643613	0.764	0.416466618643613
4	7	0.375820756799357	0.83	0.375820756799357
4	8	0.323950527000565	0.865	0.323950527000565
4	9	0.284886223184204	0.917	0.284886223184204
4	10	0.255274685711618	0.936	0.255274685711618
4	11	0.235742602375079	0.965	0.235742602375079
4	12	0.202958597236462	0.959	0.202958597236462
4	13	0.125537881771837	0.966	0.125537881771837
4	14	0.133017644294634	0.981	0.133017644294634
4	15	0.113330683593826	0.988	0.113330683593826
4	16	0.0995485304256668	0.996	0.0995485304256668
5	1	0.496988931464271	0.558	0.496988931464271
5	2	0.49753747953651	0.554	0.49753747953651
5	3	0.4909429956981	0.6	0.4909429956981
5	4	0.492619197427053	0.618	0.492619197427053
5	5	0.480819287453392	0.64	0.480819287453392
5	6	0.45305767983918	0.728	0.45305767983918
5	7	0.41102791139282	0.785	0.41102791139282
5	8	0.384379692163553	0.829	0.384379692163553
5	9	0.330884397463048	0.867	0.330884397463048
5	10	0.294738648417324	0.895	0.294738648417324
5	11	0.253580960056751	0.932	0.253580960056751
5	12	0.226130792098589	0.956	0.226130792098589
5	13	0.16505527329729	0.958	0.16505527329729
5	14	0.1531255937048	0.971	0.1531255937048
5	15	0.159214789983058	0.979	0.159214789983058
5	16	0.117549214450024	0.988	0.117549214450024

Appendix: Summary Statistics for Conditional Disadvantage

status	saves	max_sd	prob	min_sd
0	1	0.461480188051409	0.3	0.461480188051409
0	2	0.397167145848875	0.197	0.397167145848875
0	3	0.372266816386557	0.154	0.372266816386557
0	4	0.302799748493435	0.103	0.302799748493435
0	5	0.302799748493435	0.09	0.302799748493435
0	6	0.250147103868051	0.054	0.250147103868051
0	7	0.218054001448145	0.047	0.218054001448145
0	8	0.18387171834953	0.046	0.18387171834953
0	9	0.188856206322871	0.041	0.188856206322871
0	10	0.170672578726429	0.029	0.170672578726429
0	11	0.170672578726429	0.018	0.170672578726429
0	12	0.140070052543788	0.022	0.140070052543788
0	13	0.108939744206914	0.013	0.108939744206914
0	14	0.0834143750078963	0.009	0.0834143750078963
0	15	0.0944877131056093	0.01	0.0944877131056093
0	16	0.104354635210372	0.008	0.104354635210372
1	1	0.498951854343636	0.551	0.498951854343636
1	2	0.494590355364538	0.389	0.494590355364538
1	3	0.472921404310119	0.326	0.472921404310119
1	4	0.444698500960771	0.294	0.444698500960771
1	5	0.414453582167861	0.205	0.414453582167861
1	6	0.393285740491152	0.178	0.393285740491152
1	7	0.359200535947575	0.144	0.359200535947575
1	8	0.340822769427396	0.137	0.340822769427396
1	9	0.296105857217784	0.105	0.296105857217784
1	10	0.281976081451088	0.089	0.281976081451088
1	11	0.237605674293474	0.07	0.237605674293474
1	12	0.230036772966665	0.056	0.230036772966665
1	13	0.207407854282373	0.042	0.207407854282373
1	14	0.173404351355637	0.032	0.173404351355637
1	15	0.170672578726429	0.037	0.170672578726429
1	16	0.140070052543788	0.012	0.140070052543788
2	1	0.498128630106855	0.554	0.498128630106855
2	2	0.496873105413125	0.563	0.496873105413125
2	3	0.498482180709085	0.484	0.498482180709085
2	4	0.49347696959384	0.395	0.49347696959384
2	5	0.473606696852255	0.349	0.473606696852255
2	6	0.450681832465115	0.298	0.450681832465115
2	7	0.439942666207257	0.253	0.439942666207257
2	8	0.387678080750375	0.205	0.387678080750375
2	9	0.404632830758612	0.18	0.404632830758612
2	10	0.341894562536306	0.146	0.341894562536306
2	11	0.330884397463048	0.127	0.330884397463048
2	12	0.294738648417324	0.125	0.294738648417324
2	13	0.263523138347365	0.091	0.263523138347365
2	14	0.246649216167082	0.071	0.246649216167082
2	15	0.239450070654273	0.068	0.239450070654273
2	16	0.222138122224316	0.051	0.222138122224316

3	1	0.497214463005877	0.59	0.497214463005877
3	2	0.498396829242923	0.554	0.498396829242923
3	3	0.493309592805932	0.589	0.493309592805932
3	4	0.50020116073556	0.513	0.50020116073556
3	5	0.499673567115245	0.439	0.499673567115245
3	6	0.491709037722287	0.418	0.491709037722287
3	7	0.477832946043201	0.376	0.477832946043201
3	8	0.454450795494436	0.273	0.454450795494436
3	9	0.426686138819931	0.266	0.426686138819931
3	10	0.392497202606661	0.23	0.392497202606661
3	11	0.372266816386557	0.19	0.372266816386557
3	12	0.360167628640777	0.154	0.360167628640777
3	13	0.347160655202344	0.127	0.347160655202344
3	14	0.340822769427396	0.11	0.340822769427396
3	15	0.315524255098646	0.113	0.315524255098646
3	16	0.286324958712967	0.087	0.286324958712967
4	1	0.496755237280661	0.535	0.496755237280661
4	2	0.497641082153239	0.565	0.497641082153239
4	3	0.487074164005074	0.604	0.487074164005074
4	4	0.484877413483146	0.617	0.484877413483146
4	5	0.499993993957921	0.531	0.499993993957921
4	6	0.499993993957921	0.5	0.499993993957921
4	7	0.494741112874975	0.459	0.494741112874975
4	8	0.489520817361867	0.386	0.489520817361867
4	9	0.478448443317473	0.319	0.478448443317473
4	10	0.4452125494797	0.277	0.4452125494797
4	11	0.441019839121211	0.233	0.441019839121211
4	12	0.426072221034812	0.227	0.426072221034812
4	13	0.400200150125109	0.192	0.400200150125109
4	14	0.37136627864268	0.166	0.37136627864268
4	15	0.349224795737585	0.152	0.349224795737585
4	16	0.338660140392534	0.142	0.338660140392534
5	1	0.498646817545895	0.566	0.498646817545895
5	2	0.492441365543654	0.55	0.492441365543654
5	3	0.495180981262613	0.604	0.495180981262613
5	4	0.479947236005532	0.622	0.479947236005532
5	5	0.477208264521611	0.66	0.477208264521611
5	6	0.493309592805932	0.569	0.493309592805932
5	7	0.499888876540466	0.538	0.499888876540466
5	8	0.499960959436795	0.458	0.499960959436795
5	9	0.492968577110162	0.423	0.492968577110162
5	10	0.477521755553637	0.361	0.477521755553637
5	11	0.465145526002734	0.321	0.465145526002734
5	12	0.4452125494797	0.287	0.4452125494797
5	13	0.44261372200476	0.269	0.44261372200476
5	14	0.415128090270137	0.219	0.415128090270137
5	15	0.41102791139282	0.207	0.41102791139282
5	16	0.373162498451077	0.173	0.373162498451077