**A baseline deck**

Let’s consider a baseline setup for comparison.

* We have 3 energy to play cards per turn.
* We draw 5 cards per turn.
* Our deck has 16 cards: 1 card costs 0 energy, 12 cost 1 energy, 3 cost 2 energy, and 1 costs 3 energy.

(I made up this deck for this example.)

If we build our deck, we can find the average cost of our cards and simulate some draws by sampling without replacement.

library(magrittr)

set.seed(20210707)

costs <- c(0, rep(1, 12), rep(2, 3), 3)

costs

#> [1] 0 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3

mean(costs)

#> [1] 1.235294

# Simulate 3 hands

sample(costs, size = 5)

#> [1] 1 2 2 1 1

sample(costs, size = 5)

#> [1] 1 3 2 1 1

sample(costs, size = 5)

#> [1] 1 1 1 1 1

Suppose that we don’t really care about what the cards do. We want to maximize the number of cards that we can play per turn. We just want to know: **How many cards per turn can I expect to play on average?**

Let’s write a function that counts the number of playable cards in a hand given a certain energy cost. The basic logic is that we sort the costs, compute the cumulative sum (cumulative energy spent on each card), and count how many sums are less than or equal to the energy limit.

# A worked example

energy <- 3

hand <- sample(costs, size = 5)

hand

#> [1] 3 1 0 1 1

sort(hand)

#> [1] 0 1 1 1 3

cumsum(sort(hand))

#> [1] 0 1 2 3 6

cumsum(sort(hand)) <= energy

#> [1] TRUE TRUE TRUE TRUE FALSE

sum(cumsum(sort(hand)) <= energy)

#> [1] 4

count\_max\_playable <- function(hand, energy) {

sum(cumsum(sort(hand)) <= energy)

}

count\_max\_playable(hand, energy)

#> [1] 4

Now, we can do this procedure on several thousand hands and run summary statistics on those hands.

simulated\_cards\_played <- replicate(

10000,

sample(costs, size = 5) %>%

count\_max\_playable(energy = 3)

)

summary(simulated\_cards\_played)

#> Min. 1st Qu. Median Mean 3rd Qu. Max.

#> 2.000 3.000 3.000 3.173 3.000 4.000

table(simulated\_cards\_played)

#> simulated\_cards\_played

#> 2 3 4

#> 451 7364 2185

proportions(table(simulated\_cards\_played))

#> simulated\_cards\_played

#> 2 3 4

#> 0.0451 0.7364 0.2185

The expected number of playable cards per hand is 3.2. The dreaded (1, 2, 2, 2, 3) hand appears about 4.5% of the time, but the 0 card lets us play a fourth card about 21.8% of the time.

**Enter the Snecko**

Snecko Eye is probably the best relic in the game.

Now, let’s suppose we obtain the mighty [Snecko Eye](https://slay-the-spire.fandom.com/wiki/Snecko_Eye) relic. It says “Draw 2 additional cards each turn. Start each combat Confused.” Confused is a debuff that randomizes the costs of cards when we draw them. So now our setup is the following:

* We have 3 energy to play cards per turn.
* We draw 7 cards per turn.
* Our deck has 16 cards: the costs are random integers between 0 and 3 energy.

The average energy cost of any given card in our deck is now mean(0:3) = 1.5. In the baseline example, the average energy cost was 1.24. (One obvious strategy with Snecko Eye is to maximize the costs of new cards—that is, try to get as many as 2s and 3s as possible because the new expected cost is less than the original cost. But let’s ignore that dimension of gameplay for now.)

So here’s the puzzle, **how many cards per turn can I play with Snecko Eye?** We can run the same simulations as above.

snecko\_costs <- 0:3

simulated\_snecko\_cards\_played <- replicate(

10000,

sample(snecko\_costs, size = 7, replace = TRUE) %>%

count\_max\_playable(energy = 3)

)

summary(simulated\_snecko\_cards\_played)

#> Min. 1st Qu. Median Mean 3rd Qu. Max.

#> 1.000 3.000 4.000 3.826 5.000 7.000

table(simulated\_snecko\_cards\_played)

#> simulated\_snecko\_cards\_played

#> 1 2 3 4 5 6 7

#> 83 979 2911 3397 1945 621 64

proportions(table(simulated\_snecko\_cards\_played))

#> simulated\_snecko\_cards\_played

#> 1 2 3 4 5 6 7

#> 0.0083 0.0979 0.2911 0.3397 0.1945 0.0621 0.0064

Let us note that the dream—playing 7 cards in one turn—happened about 0.6% of the time and the nightmare—drawing only 2-cost and 3-cost cards—happened 0.8% of the time. Recall that in the baseline setup, we got to play 4 cards 21.8% of the time. With Snecko Eye, we can play 4 or more cards per turn 60.3% of the time. Snecko Eye simply lets us play more cards on average.

**Where does this power come from?**

Is the magic of Snecko Eye the card draw or the cost randomization? Well, let’s suppose that we are just confused and we draw only 5 cards (as in the baseline example.)

simulated\_confused\_cards\_played <- replicate(

10000,

sample(snecko\_costs, size = 5, replace = TRUE) %>%

count\_max\_playable(energy = 3)

)

summary(simulated\_confused\_cards\_played)

#> Min. 1st Qu. Median Mean 3rd Qu. Max.

#> 1.00 2.00 3.00 3.03 4.00 5.00

table(simulated\_confused\_cards\_played)

#> simulated\_confused\_cards\_played

#> 1 2 3 4 5

#> 358 2496 4162 2454 530

proportions(table(simulated\_confused\_cards\_played))

#> simulated\_confused\_cards\_played

#> 1 2 3 4 5

#> 0.0358 0.2496 0.4162 0.2454 0.0530

Here the average number of cards played is 3.0 and we play 4–5 cards per turn 29.8% of the time. This percentage is greater than the baseline case (21.8%), but the nightmare case is worse (1 card), occuring 3.6% of the time.

We can plot the three simulations side by side and observe the distributions. First, we package them together into a single dataframe suitable for plotting and plot a bar chart.

library(ggplot2)

sim1 <- data.frame(

set = "Baseline",

energy = 3,

cards = simulated\_cards\_played

)

sim2 <- data.frame(

set = "Snecko Eye",

energy = 3,

cards = simulated\_snecko\_cards\_played

)

sim3 <- data.frame(

set = "Confused",

energy = 3,

cards = simulated\_confused\_cards\_played

)

sims <- rbind(sim1, sim2, sim3)

ggplot(sims) +

aes(x = cards) +

geom\_bar(aes(y = stat(prop))) +

facet\_wrap("set") +

scale\_x\_continuous(breaks = 1:7, minor\_breaks = NULL) +

scale\_y\_continuous(labels = scales::label\_percent()) +

labs(

title = "Confusion increases variance. Card draw increases mean.",

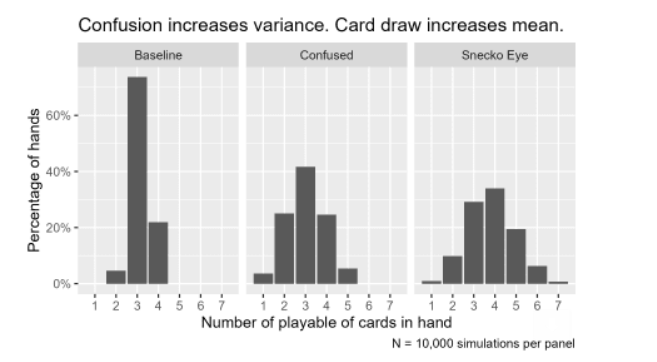
x = "Number of playable of cards in hand",

y = "Percentage of hands",

caption = "N = 10,000 simulations per panel"

) +

theme\_grey(base\_size = 12)



Both the confused and the Snecko Eye panels have increased variance. The bars are shorter and more spread out, compared to the Baseline panel. The peak (the mode) shifts from 3 to 4 cards from the Confused and Snecko Eye panels.

A more statistically niche technique would be plotting the [empirical cumulative distribution function](https://en.wikipedia.org/wiki/Empirical_distribution_function). Imagine taking the bars from the previous plot and summing them along the *x* axis so that they are cumulative percentages. These percentages would tell you about the percentage of cases less than or equal to that given value. In the plot below, I do that procedure on reversed *x* axis, so we can look at what proportion of simulations had at least 4 cards played. (I chose the reversed *x* axis to visually convey the advantage of Snecko Eye.)

library(dplyr)

props <- sims %>%

count(set, cards) %>%

# Fill in rows that would be n = 0

tidyr::complete(set, cards = 1:7, fill = list(n = 0)) %>%

# Compute ECDF in reverse order (dtarting at 7 cards)

arrange(set, desc(cards)) %>%

group\_by(set) %>%

mutate(

proportion = n / sum(n),

ecdf = cumsum(proportion)

) %>%

ungroup()

ggplot(props) +

aes(x = cards) +

geom\_step(

aes(y = ecdf , color = set, linetype = set),

direction = "mid"

) +

geom\_label(

aes(color = set, y = ecdf),

label = "Snecko can play 4 or more\ncards in 60% of hands",

data = . %>% filter(set == "Snecko Eye", cards == 4),

y = .65,

nudge\_x = -.25,

hjust = 1.0,

vjust = 0,

fill = scales::alpha("grey93", .6),

label.size = 0,

show.legend = FALSE,

size = 4.5,

) +

scale\_x\_reverse(breaks = 7:0, minor\_breaks = NULL) +

scale\_y\_continuous(labels = scales::label\_percent()) +

labs(

x = "At least X playable cards in hand",

y = "Percentage of hands",

caption = "N = 10,000 simulations per line",

color = NULL,

linetype = NULL

) +

theme\_grey(base\_size = 12) +

theme(

legend.position = "top",

legend.justification = "left"

)

**The advantage at higher energy**

During a run through the game, we can obtain up to two relics (along with Snecko Eye) that increase our energy per turn by 1 unit. Let’s see how these new energy budgets affect the simulations.

First, we run the simulations. We put the main code into functions so that we can build the dataframes more easily.

simulate\_decko <- function(n, energy, costs, size = 5) {

replicate(

n,

sample(costs, size = size) %>%

count\_max\_playable(energy = energy)

)

}

simulate\_snecko <- function(n, energy, size = 7) {

snecko\_costs <- 0:3

replicate(

n,

sample(snecko\_costs, size = size, replace = TRUE) %>%

count\_max\_playable(energy = energy)

)

}

additional\_sims <- rbind(

# include old results

sims,

data.frame(

set = "Baseline",

energy = 4,

cards = simulate\_decko(10000, 4, costs)

),

data.frame(

set = "Baseline",

energy = 5,

cards = simulate\_decko(10000, 5, costs)

),

data.frame(

set = "Snecko Eye",

energy = 4,

cards = simulate\_snecko(10000, 4)

),

data.frame(

set = "Snecko Eye",

energy = 5,

cards = simulate\_snecko(10000, 5)

),

data.frame(

set = "Confused",

energy = 4,

cards = simulate\_snecko(10000, 4, size = 5)

),

data.frame(

set = "Confused",

energy = 5,

cards = simulate\_snecko(10000, 5, size = 5)

)

)

We can make the same kind of plot as before. We see that the distribution with the highest mode (the peak that lands on the highest number of cards) in each row is Snecko Eye.

ggplot(additional\_sims %>% mutate(energy = paste0(energy, " energy"))) +

aes(x = cards) +

geom\_bar(aes(y = stat(prop))) +

facet\_grid(energy ~ set) +

scale\_x\_continuous(breaks = 1:7, minor\_breaks = NULL) +

scale\_y\_continuous(labels = scales::label\_percent()) +

labs(

x = "Number of playable of cards in hand",

y = "Percentage of hands",

caption = "N = 10,000 simulations per panel"

) +

theme\_grey(base\_size = 12)

One limitation of the other two non-Snecko sets becomes more obvious in the 5-energy row: They can never play 6 or 7 cards in a turn. They don’t draw that many cards. Their distributions are cut off at 5 cards.

If we look at numerical summaries, we get some sense that the benefit of Snecko diminishes as energy increases but we won’t explore this trend in any detail.

additional\_sims %>%

group\_by(set, energy) %>%

summarise(

mean = mean(cards),

sd = sd(cards),

median = median(cards),

.groups = "drop"

) %>%

tidyr::pivot\_longer(cols = c(mean, sd, median)) %>%

tidyr::pivot\_wider(

names\_from = energy,

values\_from = value,

names\_prefix = "Energy "

) %>%

rename(Set = set, Statistic = name) %>%

arrange(Statistic, Set) %>%

knitr::kable(digits = 2)

| **Set** | **Statistic** | **Energy 3** | **Energy 4** | **Energy 5** |
| --- | --- | --- | --- | --- |
| Baseline | mean | 3.17 | 3.81 | 4.27 |
| Confused | mean | 3.03 | 3.44 | 3.79 |
| Snecko Eye | mean | 3.83 | 4.30 | 4.72 |
| Baseline | median | 3.00 | 4.00 | 4.00 |
| Confused | median | 3.00 | 3.00 | 4.00 |
| Snecko Eye | median | 4.00 | 4.00 | 5.00 |
| Baseline | sd | 0.48 | 0.56 | 0.53 |
| Confused | sd | 0.92 | 0.89 | 0.84 |
| Snecko Eye | sd | 1.11 | 1.08 | 1.06 |