Pokemon Breeding Analysis

Roland Hentz

Abstract

Using a simulation method that depended upon open knowledge of a few of the mechanics of the video game Pokemon Sun, different methods of acquiring highly powerful Pokemon were compared. Graphical analysis gave insight into the benefit and cost of different acquisition methods. Final analysis allowed for the recommendation of the most efficient method of obtaining the most powerful Pokemon of a given species. The most efficient method to acquire Pokemon high individual values is to breed Pokemon starting with a Destiny Knot and with an initial parent chained to have five perfect IVs. If the player has selected a Pokemon whose attack style matters, then some time may be saved by ignoring the unused attack style.

Introduction

In the popular video game series Pokemon, players capture, train, and battle with digital monsters called Pokemon. Each of these monsters has six statistics, which the player can view, that communicate the effectiveness of that Pokemon in a battle. Those statistics are calculated using values innate to a species of that Pokemon (base statistics), values accumulated through training (effort values), and values assigned to a Pokemon at birth or capture (individual values). Base statistics are immutable, and effort values are easily assigned by a motivated player. The focus of this project is on a Pokemon's individual values. In this paper, individual values will be referred to as IVs.

Each of the Pokemon's six statistics has an IV between zero and thirty-one that is randomly assigned upon capture or birth. These cannot be changed at any time or in any way during the lifetime of the Pokemon. IVs have a rather high weight when used in calculating a Pokemon's visible statistics. In other words, IVs matter tremendously when a Pokemon makes it into a battle. While the mechanics are very similar across multiple generations of the game series, this paper has been written to match the mechanics of Pokemon Sun and Moon.

There are two main methods for acquiring Pokemon with high IVs. The first is breeding. By taking two Pokemon of breedable species to a Pokemon day-care center within the video game, the player will be given an egg of the female Pokemon's species. This new Pokemon will inherit three out of its six IVs from its parents. Using an item called the Destiny Knot, a bred Pokemon inherits five out of its six IVs from its parents. The parent each inherited IV comes from is random. When an egg Pokemon is born that has better IVs than its parent of that gender, the parent is replaced with the child to progressively increase the IVs of the parents.

The other method of acquiring high IV Pokemon is called ally-chaining. In this method, a player first engages a Pokemon of the desired species in combat. Next, the player weakens that Pokemon's health, causing it to summon another Pokemon as an "ally" to assist it in battle. After repeating this second step thirty times, the Pokemon summoned will have four perfect IVs. Only one Pokemon may be caught in each ally-chaining episode.

Each of these methods can be relatively time-consuming, and there are a handful of ways to modify each, such as using the Destiny Knot. Therefore, this paper will answer the question of which method of obtaining Pokemon with high IVs is the most efficient.

Materials and Methods

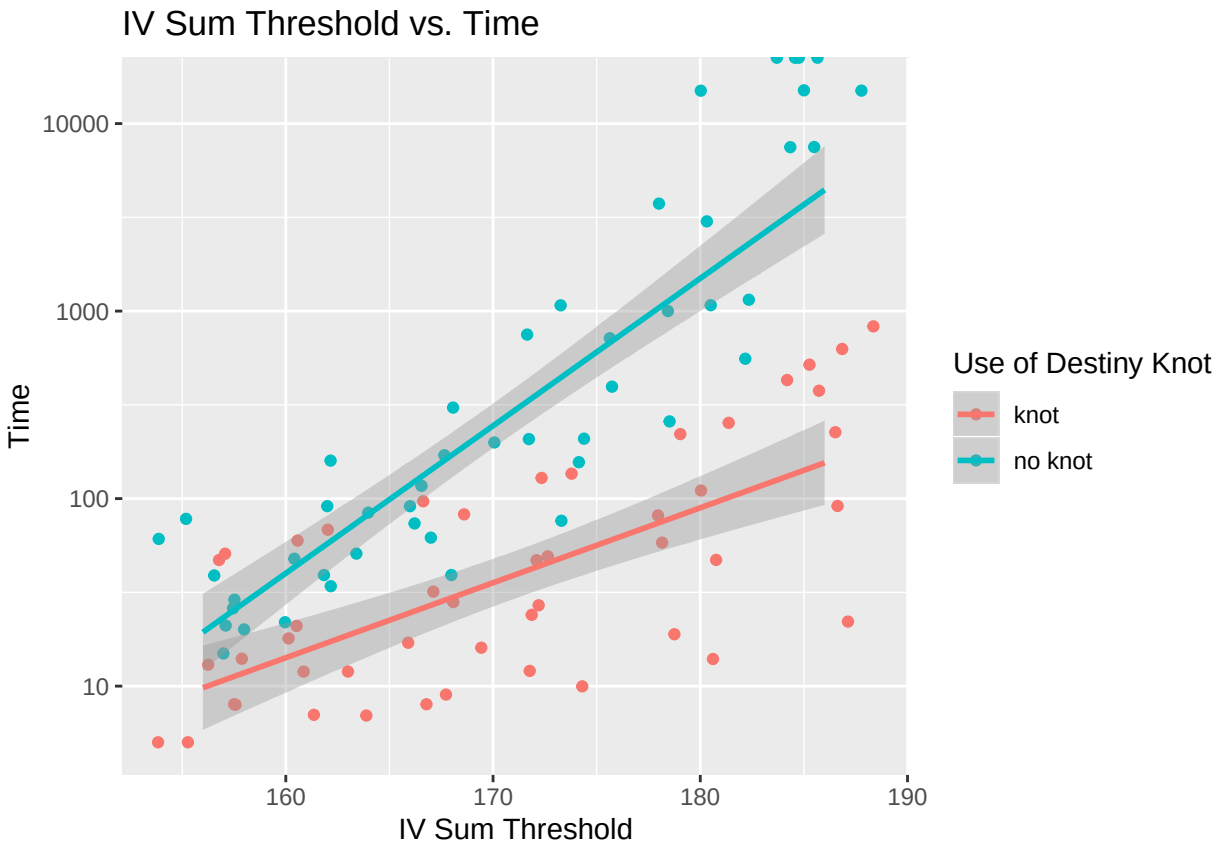
To solve this problem, a series of distributions were generated using random sampling and Pokemon game mechanics. Since most of the Pokemon game mechanics are quite-well known to enthusiasts and any internet-user with the time and interest, the distributions were generated using exactly the same rules as the video game series. As such, the R simulation perfectly emulates a player performing each of the methods analyzed.

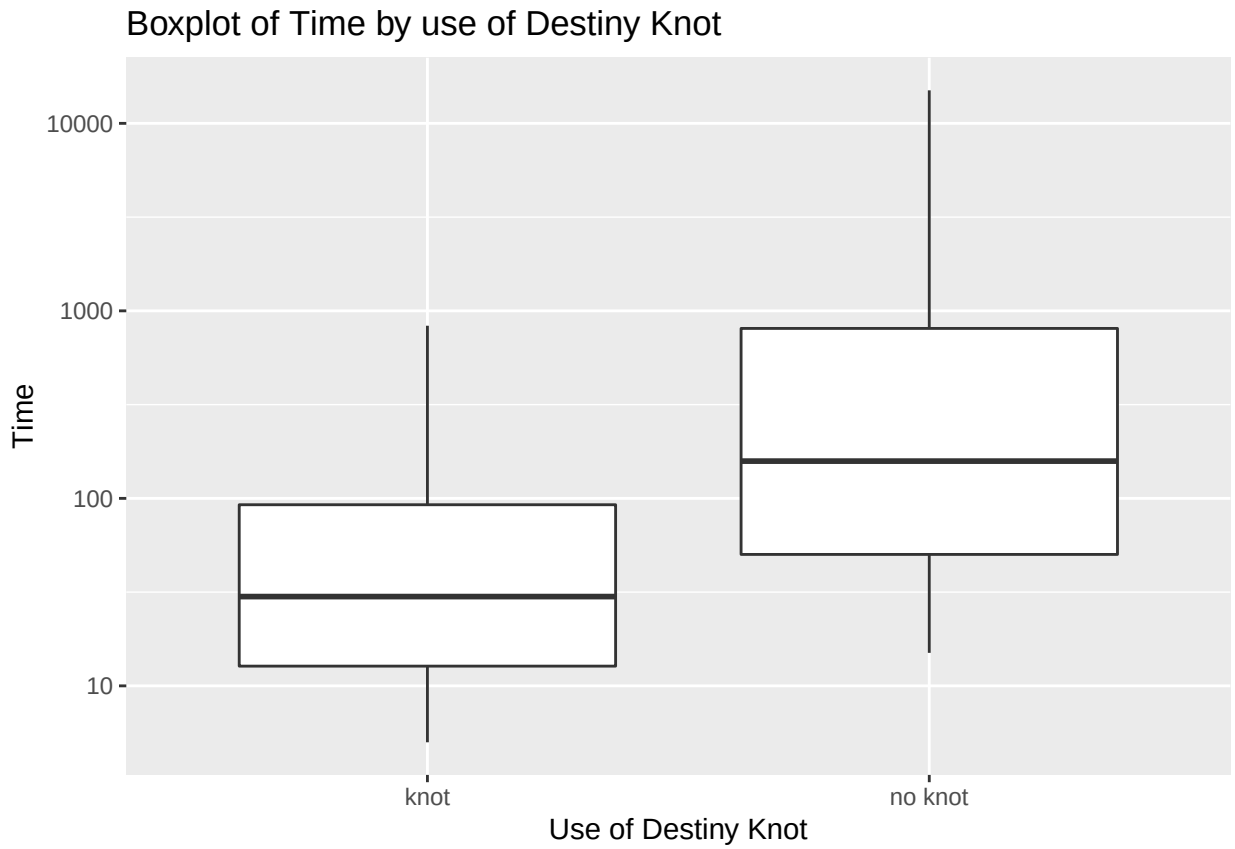
Two variables that require explanation right away are threshold and time. The threshold variable is an integer that represents the value that a Pokemon's sum of IVs must surpass in order to be considered a success. In breeding, that is the point at which breeding is started over from scratch again. In ally-chaining, the threshold simply measure which Pokemon are acceptable products of the method. The other variable to explain now is time. This variable measures time in minutes to acquire a Pokemon that meets the given threshold requirement.

Threshold values are 186 (all six IVs are perfect), 180, 174, 168, 162, and 156

Results

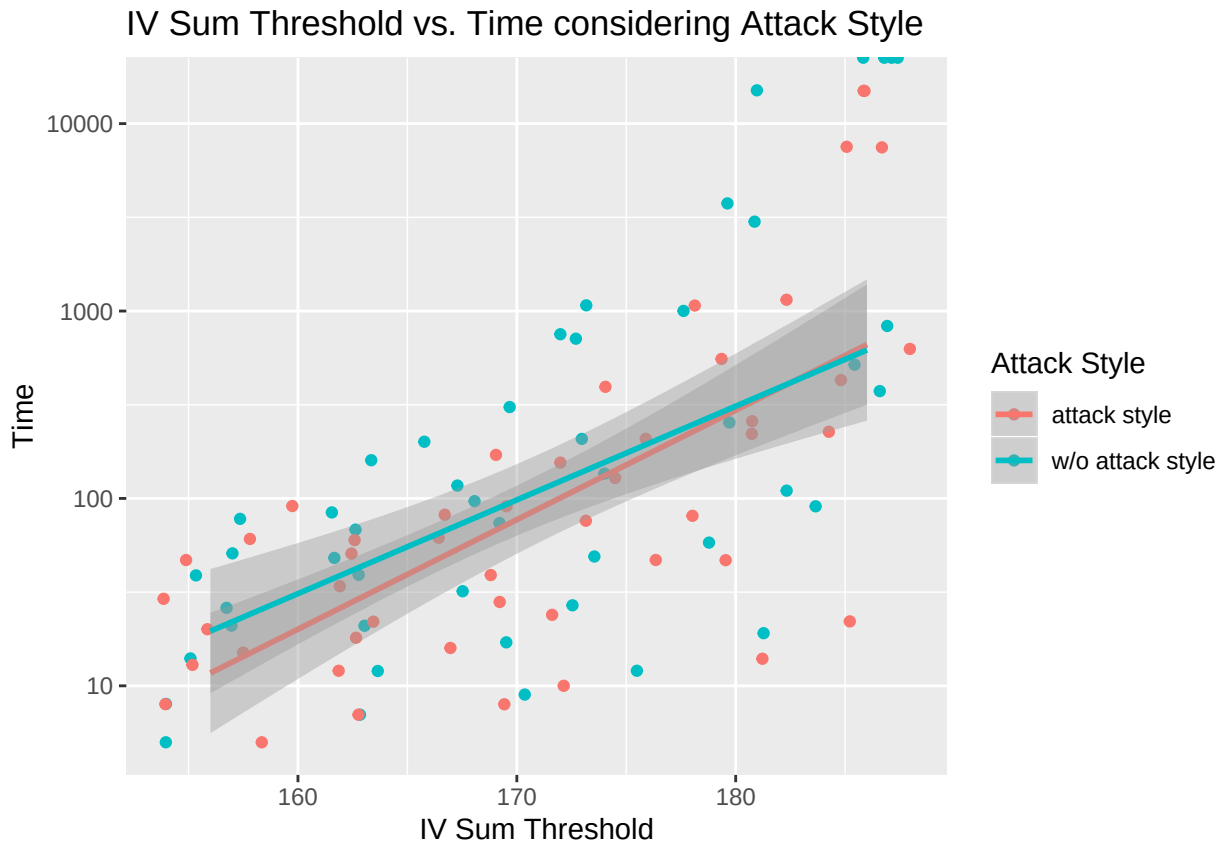
Each of the two previously mentioned methods of obtaining high-IV Pokemon have a few different cases to consider. We begin with considering breeding. The first variable to look at is the Destiny Knot. This item increases the number of IVs inherited from a bred Pokemon's parents from three to five. In the scatter plot and box plot, higher values of time, along the y-axis, are worse because they correspond to a greater amount of time needed to acquire an acceptable Pokemon using the given method and conditions. In the scatter plot, the blue points (cases not using the knot) appear to be clustered above the red points (cases using the knot). Overlaying a linear regression makes this observation more clear. At no point do either regression line fall inside the other line's confidence interval. This indicates that the use of a knot reduces the time spent breeding. In the box plot, the median of the knot usage group lies above the 3rd quartile for the group not using the knot. This further shows that, at each of the points of measurement used to generate the plot, not using a knot takes quite a bit longer than using one.

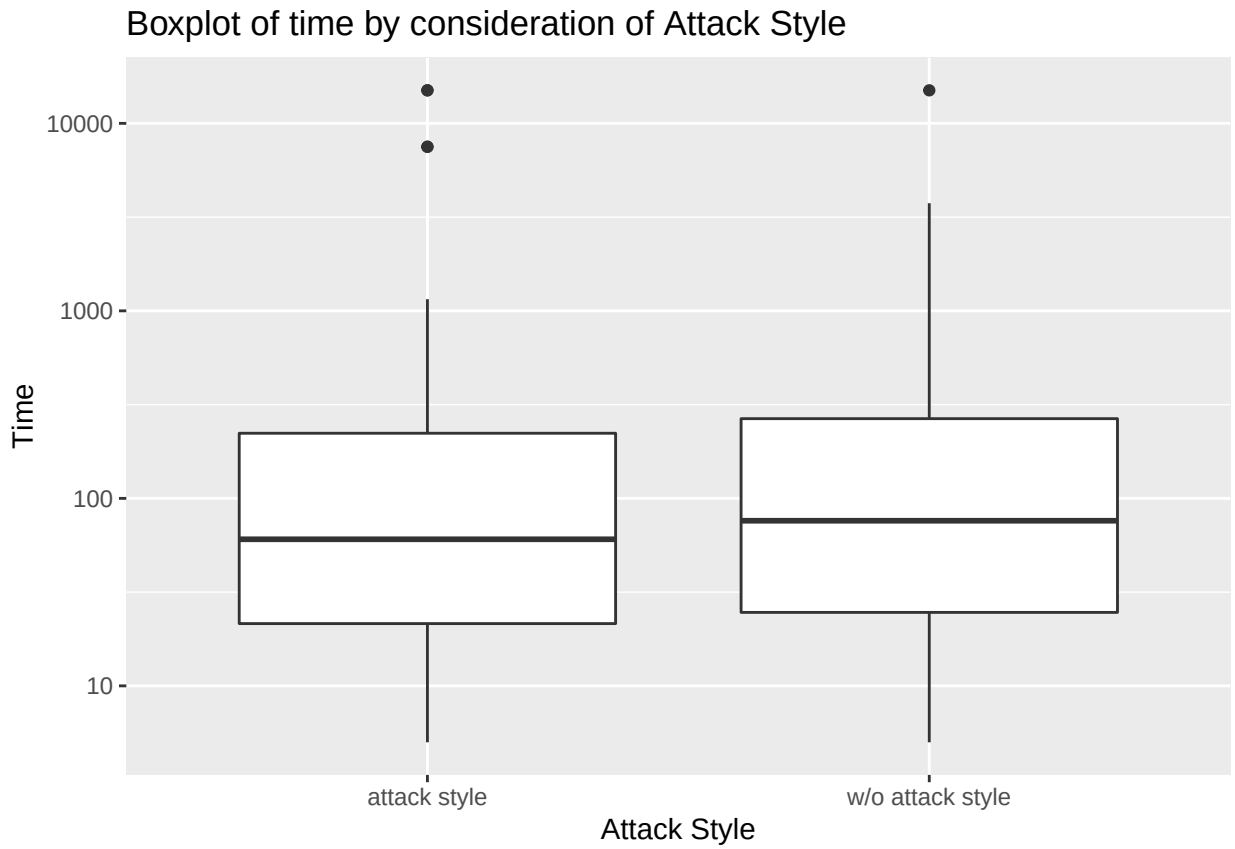

IV Sum Threshold vs. Time
Time
10000
1000
100
10
160
170
180
190
IV Sum Threshold
Use of Destiny Knot
knot
no knot



The next condition to consider is what we will refer to as attack style. For most Pokemon, only five out of its six IVs are truly important. This is because two of the IVs are used exclusively to one another to calculate attack damage, based on the style of attack used. Almost all Pokemon have a preferred attack style that depends on the Pokemon's base stats and available attacks. Whether or not attack style matters is generally inherent to the Pokemon the player wishes to use, meaning that it is not necessarily a variable the player chooses to manipulate. However, we will look at whether or not time is saved by ignoring the unused attack IV when breeding.

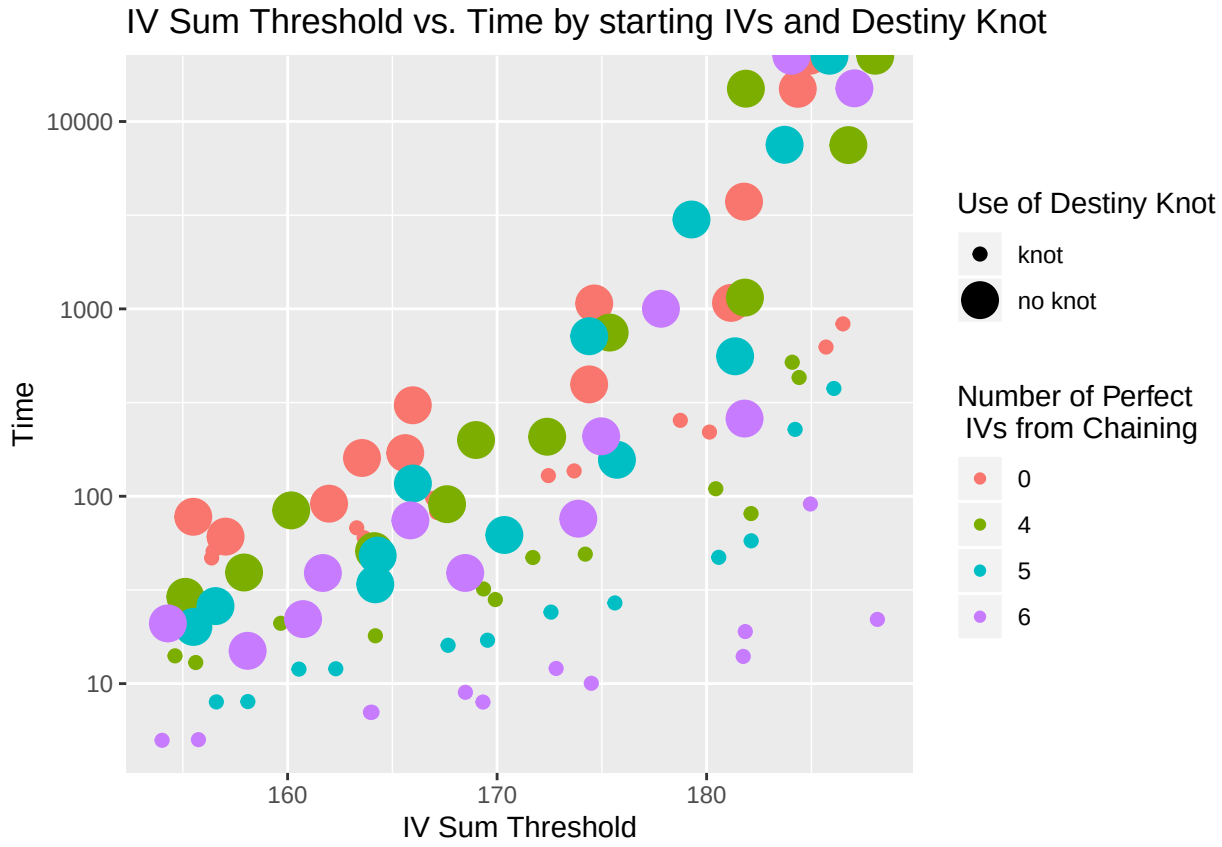
The scatter plot colored by whether attack style is considered does not suggest that attack style saves time. The linear regression lines overlap each other's confidence interval. The points appear to be distributed quite evenly. The box plot suggests a very modest amount of time saved by considering attack style. The quantiles appear similarly situated, suggesting that there is not much of a difference between the two cases.

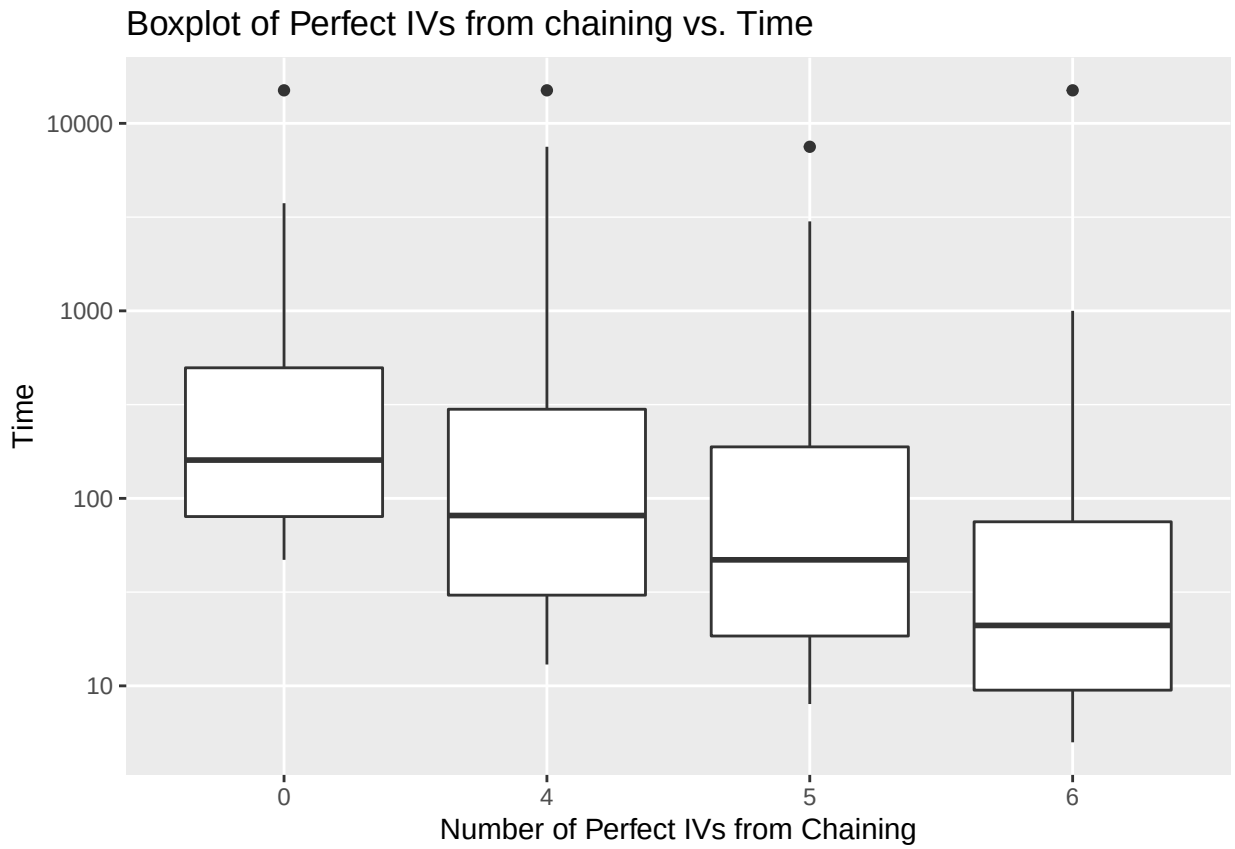

IV Sum Threshold vs. Time considering Attack Style
Time
10000
1000
100
10
160
170
180
IV Sum Threshold
Attack Style
attack style
w/o attack style

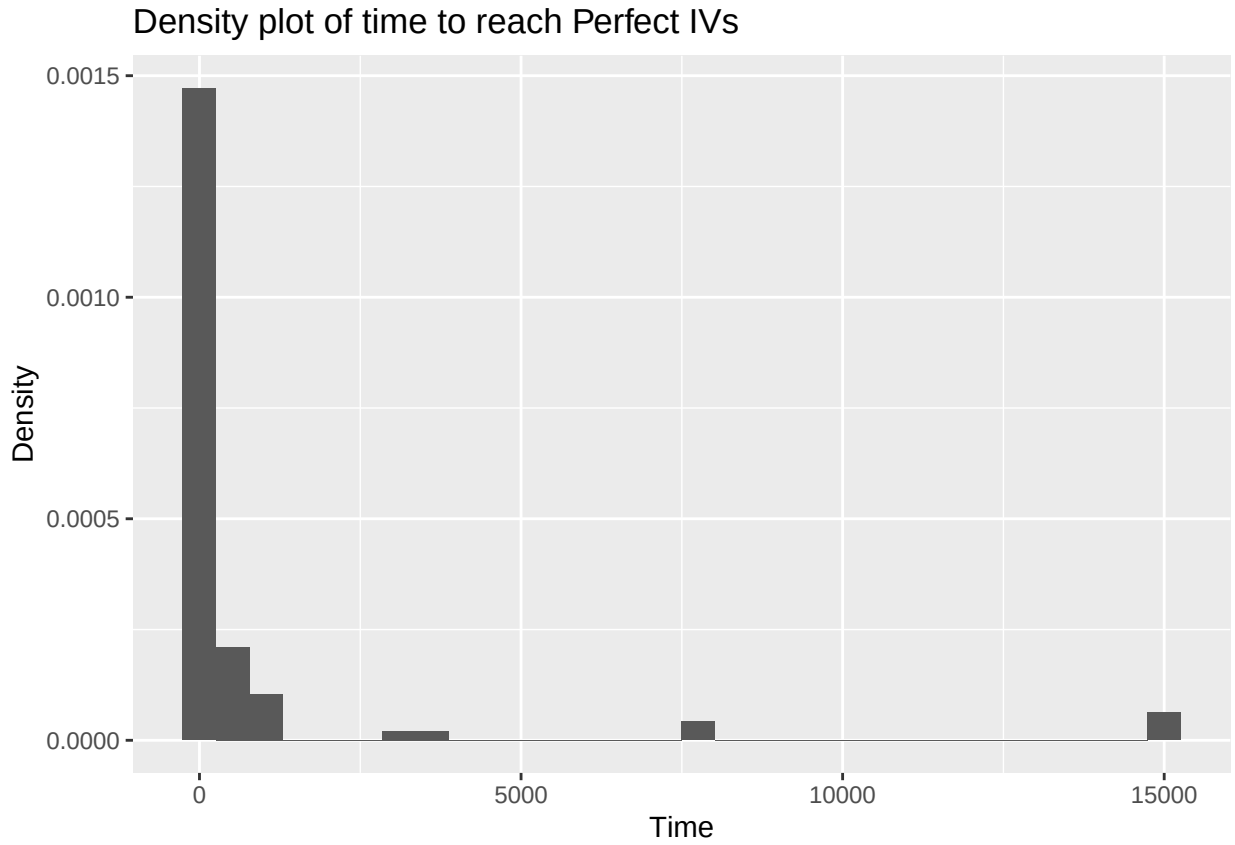


The next breeding condition to analyze is the IV sum of the initial parents in a breeding session. The player may re-use a Pokemon that has been chained to guarantee at least four perfect IVs across breeding sessions. While it only takes a single chaining session to acquire a Pokemon with four perfect IVs, a player may continue to chain Pokemon in order to catch one with five or even all six perfect IVs with better odds than without chaining. Now we will examine the cases of using a totally random wild Pokemon as the initial parent and parents chained to four, five, and six perfect IVs.

In the scatter plot, large circles show Pokemon that were bred with a Destiny Knot, small circles without. It appears quite clearly that across the cases of Destiny Knot usage, the more an initial parent is chained to perfect IVs, the faster breeding is. The box plot demonstrates this phenomenon quite well. What must also be taken into consideration beyond just time saved by chaining is how much time it would take to acquire a Pokemon with a given number of perfect IVs. Because the values for time are positively-skewed, and because some values for time are infinity (meaning a Pokemon meeting the threshold requirement was never acquired with that method), we will look at the median of time across cases of chaining the initial parent in breeding. Additionally, the median offers some interpretive benefit by allowing a player to see below or above which point they have a one in two chance of falling. The median for not chaining, case reported below.


IV Sum Threshold vs. Time by starting IVs and Destiny Knot
Time
10000
1000
100
10
160
170
180
IV Sum Threshold
Use of Destiny Knot
knot
no knot
Number of Perfect IVs from Chaining
0
4
5
6


Boxplot of Perfect IVs from chaining vs. Time
Time
10000
1000
100
10
0
4
5
6
Number of Perfect IVs from Chaining




```
## Median for chaining to  0  IVs:  165  minutes.
## Median for chaining to  4  IVs:  82.5  minutes.
## Median for chaining to  5  IVs:  47.5  minutes.
## Median for chaining to  6  IVs:  21.5  minutes.
```

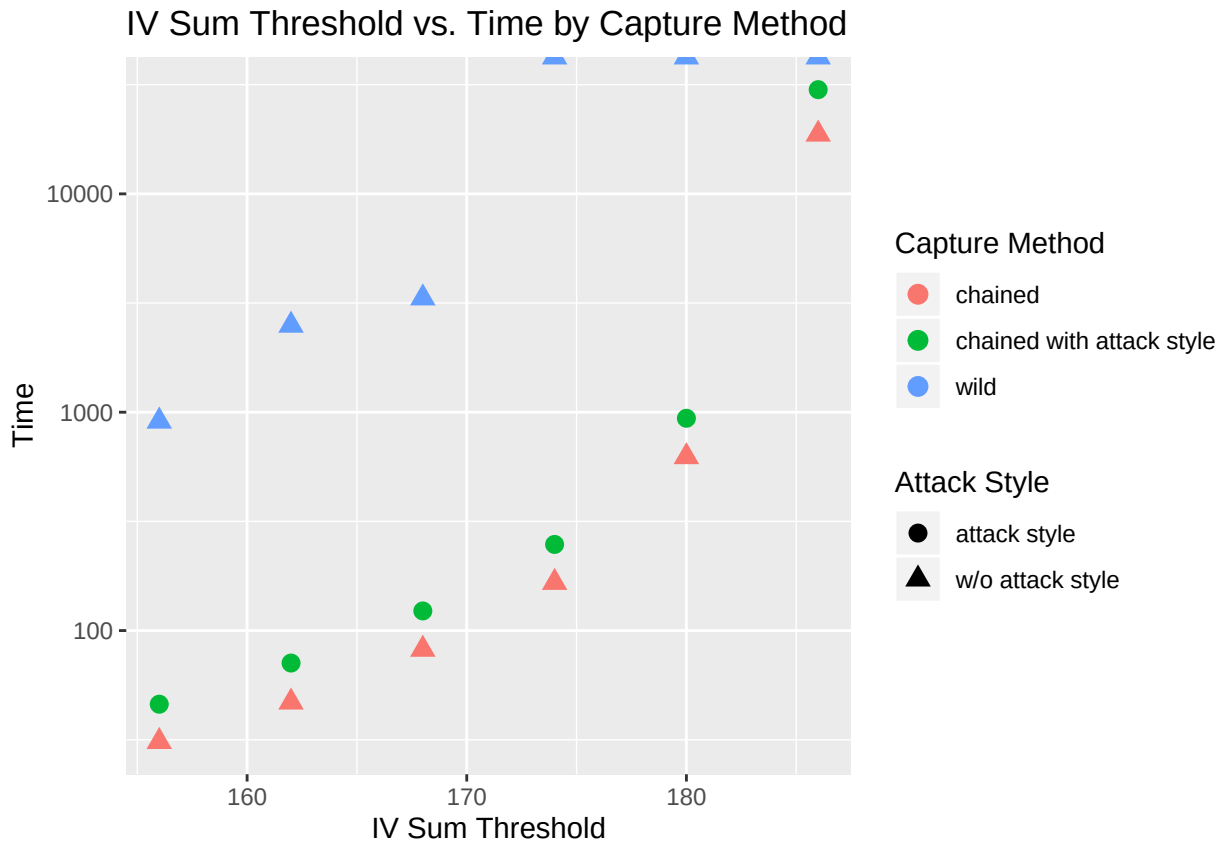
Next we must ask how much time it takes to acquire a five IV initial parent or a six IV initial parent. Remember that we need only capture one since we can re-use the initial parent for each breeding. The median time to capture a five IV parent is and the median time for a six IV parent are each reported below. Chaining up to a five IV parent will save the player time if the player intends to breed at a minimum five Pokemon. Chaining to a six IV parent would save time only if the player intended to breed around five-hundred Pokemon.

```
## Median time to capture a five IV Pokemon by chaining:  150  minutes.
```

```
## Median time to capture a six IV Pokemon by chaining:  9975  minutes.
```

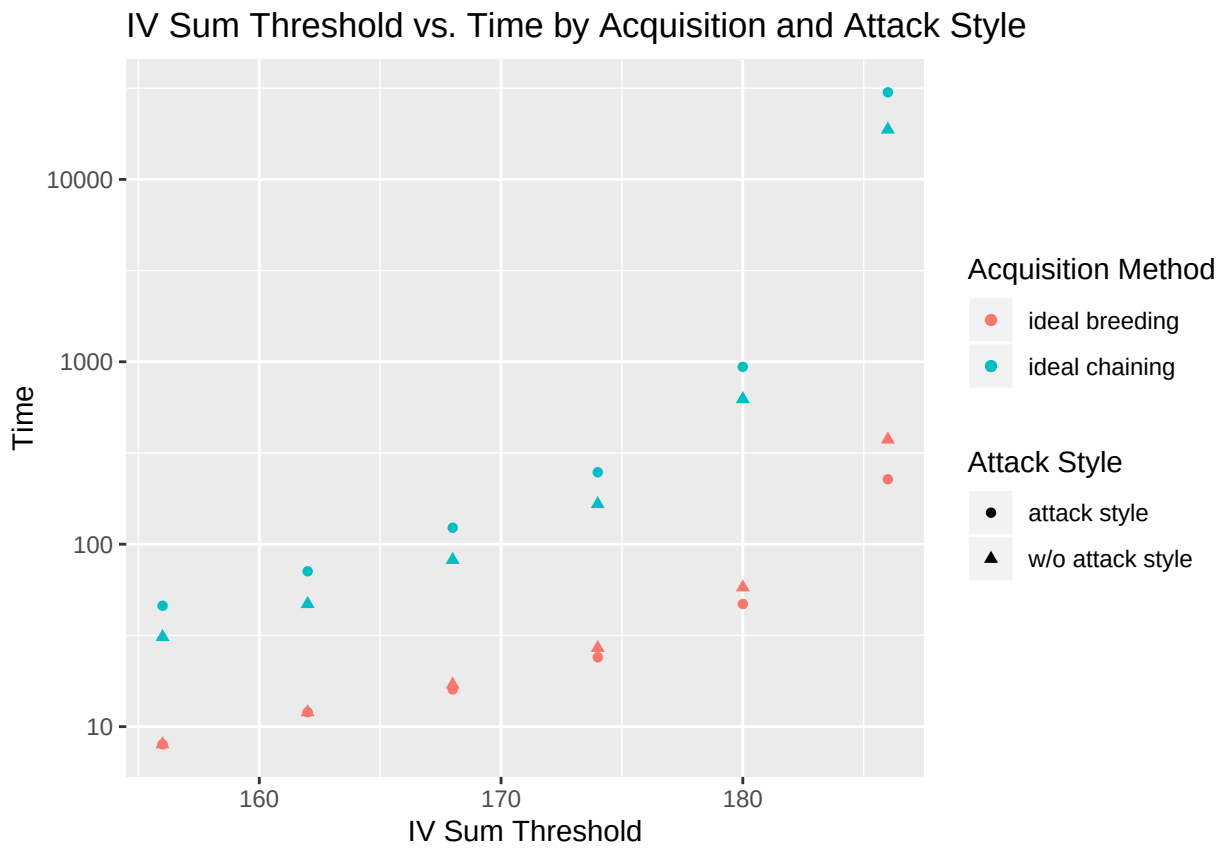
Now that we have seen all cases for breeding, we will look at capturing Pokemon. The three cases to consider are catching completely wild Pokemon with no other strategies implemented, ally-chaining, and ally-chaining considering attack style. These cases are coded by color and shape in the scatter plot below. Catching Pokemon completely wild appears to take dramatically longer than chaining. Chaining when considering attack style actually slows down the process of acquiring a high IV Pokemon in this case. This is because a Pokemon that matches the threshold when considering all six IVs may have to be rejected on the basis of much of its IV sum coming from its useless attack style. Chained Pokemon with the correct attack style are slightly less common.

To make the calculation for the attack style chained captures, the threshold value was multiplied by five-sixths.



Discussion

Across chaining and breeding, a set of ideal conditions can be selected. These are plotted below by color. Remember that attack style is generally chosen by the Pokemon, not by the player, but does have an impact on time. Ideal breeding conditions are to use the Destiny Knot and to start breeding with a five IV initial parent. Ideal chaining is essentially just regular chaining, as there are not optional techniques that are in question of saving time. Here we can see that the best method is to breed using the aforementioned techniques. If the player is lucky and chooses a Pokemon whose attack style matters, he or she will save just a little bit more time.



Literature Cited

https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
https://www.gamesradar.com/pokemon-sun-and-moon-breeding-guide/
https://pokemongohub.net/pokemon-sun-moon-advanced-s-o-s-chaining-guide/
https://samurai-gamers.com/pokemon-sun-and-moon/get-destiny-knot/
https://gamefaqs.gamespot.com/boards/187276-pokemon-sun/74677783
https://stackoverflow.com/questions/8214303/conditional-replacement-of-values-in-a-data-frame

Code Appendix

```
# Chunk 1

if(!require(ggplot2)) install.packages("ggplot2")
library(ggplot2)

wild_sim <- function(iter) {
  # Simulates distribution of wild-caught Pokemon.
  # Args:
  #   iter: The size of the distribution.
  # Returns:
  #   A vector containing the sums of the IVs.

  stats <- numeric(length = iter)
  x <- 0

  for (i in 1:iter) {
    for (j in 1:6){
      y <- sample(0:31, 6, replace=T)
      x <- sum(y)
    }
    stats[i] <- x
  }

  return(stats)
}


# Print quantiles of a distribution of wild-caught.
# for (k in c(.99,0.999,0.9999)) {
#   cat(k, quantile(wild_sim(100000),probs=k), "\t")
# }

chained_sim <- function(iv_slots) {
  # Simulates a single Pokemon, expected to be caught by ally-chaining.
  # Args:
  #   iv_slots: The number of perfect IVs possessed by the Pokemon.
  #     Normal chaining means that this value will be 4. Values of 0, 5, and
  #     6 are useful to simulate unique circumstances for the breeding function.
  # Returns:
  #   A vector of length 6 containing the IVs of the Pokemon simulated.

  # selects which IVs will be perfect
  log_v <- c(1,2,3,4,5,6)
  perfect_slots <- sample((iv_slots >= log_v),
            6, replace=F)
  chain_stats <- numeric(length = 6)

  for (l in c(1:6)) {
    if (perfect_slots[l]) {
      chain_stats[l] <- 31
    } else {
```

```
      chain_stats[l] <- sample(0:31, 1, replace=T)
      }
  }
  return(chain_stats)
}

chained_sim_mult <- function(iters) {
  # Simulates a distribution of Pokemon caught using chaining
  # Args:
  #   iters: The size of the distribution.
  # Returns:
  #   A vector containing the sums of the IVs.

  mons <- numeric(length = iters)
  for (m in c(1:iters)) {
    mons[m] <- sum(chained_sim(4))
  }
  return(mons)
}

breed_sim <- function(wknot,ch_ivs,atk_style,goal_iv,num_iter) {
  # Simulates Pokemon breeding.
  # Args:
  #   wknot: Boolean true if destiny knot in use, false if not.
  #   ch_ivs: The number of perfect IVs of one of the starting parents. 0 if
  #     starting from a wild Pokemon. 4, 5, and 6 refer to chain-caught Pokemon.
  #   atk_style: Boolean true if one of Atk of SpA does not matter in Pokemon
  #     being bred.
  #   goal_iv: The IV sum total that is considered the threshold for accepting
  #     an egg as perfect.
  #   num_iter: The size of the distribution.
  # Returns:
  #   A dataframe containing the following variables for each egg.
  #     num_eggs: The number of eggs since breeding was last started over.
  #     iv_sum: The sum of the IVs of the egg.
  #     father: The sum of the IVs of the father of the egg.
  #     mother: The sum of the IVs of the mother of the egg.
  #     num_perf: The number of perfect eggs that have been generated so far.

  # initializations
  m <- f <- numeric(length = 6)
  inheritance <- numeric(length = 6)
  gender <- c('M','F')
  child <- numeric(length = 6)
  perfect_egg <- FALSE
  perfect_num <- 0
  num_eggs <- iv_sum <- father <- mother <- num_perf <-
  numeric(length = num_iter)
  envis <- data.frame(num_eggs, iv_sum, father, mother, num_perf)


  # five IVs inherited when using knot, three otherwise
  if(wknot) {
```

```
  iv_sample <- c(TRUE, TRUE, TRUE, TRUE, TRUE, FALSE)
} else {
  iv_sample <- c(TRUE, TRUE, TRUE, FALSE, FALSE, FALSE)
  }

# breeding loop
for (l in c(1:num_iter)) {
  # generates parents and resets counter and flag if start of loop or reset
  #   necessary due to perfect egg being reached
  if (l == 1 | perfect_egg == TRUE) {
    m <- chained_sim(ch_ivs)
    f <- sample(0:31, 6, replace=T)
    eggs <- 0
    perfect_egg <- FALSE
  }

  # randomly assign which IVs will be inherited, gender of egg, and which IVs
  #   will come from which parent if inherited
  inheritance <- sample(iv_sample, 6, replace=F)
  child_gender <- sample(gender, 1)
  iv_genders <- sample(gender, 6, replace=T)
  eggs <- eggs + 1

  # generates egg based on inheritance, parents IVs, and random simulation
  for (k in c(1:6)) {
    if (inheritance[k]) {
      if (iv_genders[k] == 'M') {
        child[k] <- m[k]
      } else if (iv_genders[k] == 'F') {
        child[k] <- f[k]
      }
    } else {
      child[k] <- sample(0:31,1)
    }
  }

  # checks if egg is perfect; if true, triggers reset flag and increments the
  #   number of perfect eggs found
  # if egg not perfect, compares egg to parents and replaces parent if
  #   necessary
  # if attack style matters, only five IVs must be considered, and 5/6 of the
  #   IV sum threshold gives correct sum for testing
  if (atk_style) {
    if (sum(child[1:5]) >= goal_iv * (5 / 6)) {
      perfect_egg <- TRUE
      perfect_num <- perfect_num + 1
    } else if (child_gender == 'M') {
      if (sum(child[1:5]) > sum(m[1:5])) {
        m <- child
      }
    } else if (sum(child[1:5]) > sum(f[1:5])) {
      f <- child
    }
```

```
    } else {
      if (sum(child) >= goal_iv) {
        perfect_egg <- TRUE
        perfect_num <- perfect_num + 1
      } else if (child_gender == 'M') {
        if (sum(child) > sum(m)) {
          m <- child
        }
      } else if (sum(child) > sum(f)) {
        f <- child
      }
    }

    envis[l,1:5] <- c(eggs, sum(child), sum(m), sum(f), perfect_num)
  }
  return(envis)
}

frame_gen <- function(sim_size) {
  # Creates a dataframe containing data for each case in consideration
  # Args:
  #   sim_size: Distribution size for every case considered by the function.
  #     Function processes 114 cases.
  # Returns:
  #   A dataframe containing egg count data as well as variable flags for each
  #     case.
  #     bred: Wild caught and chain cases are 0; breed function is 1.
  #     chained: Perfect IV state 0, 4, 5, or 6 with equivalent meanings to the
  #       chain simulator.
  #     atkst: 1 if attack style matters, 0 otherwise.
  #     knot: 1 if using knot, 0 otherwise.
  #     thold: Threshold for a perfect egg.
  #     count: Number of perfect eggs.

  # initializations
  bred <- atkst <- knot <- logical(114)
  chained <- thold <- count <- numeric(114)
  pd <- data.frame(bred, knot, chained, atkst, thold, count)
  thr <- 186
  row <- 1

  # loops through all cases in consideration and fills dataframe
  while (thr > 155) {

    # wild-caught simulation
    a <- wild_sim(sim_size)
    acount <- length(subset(a, a>=thr))

    # chained simulation with attack style consideration
    b <- chained_sim_mult(sim_size)
    bcount <- length(subset(b, b>=thr))
    b2count <- floor(length(subset(b, b>=thr))*2/3)
```

```
    # filling dataframe
    pd[row, 1:6] <- c(FALSE, FALSE, 0, FALSE, thr, acount)
    row <- row + 1
    pd[row, 1:6] <- c(FALSE, FALSE, 4, FALSE, thr, bcount)
    row <- row + 1
    pd[row, 1:6] <- c(FALSE, FALSE, 4, TRUE, thr, b2count)
    row <- row + 1

    for (cval in c(0,4,5,6)) {
      # considers no chaining, regular chaining, five perfect IVs, and six
      #   perfect IVs
      for (clog in c(FALSE, TRUE)) {
        # without knot and with knot
        for (c2log in c(FALSE, TRUE)) {
          # attack style doesn't matter, does matter
          c <- breed_sim(clog, cval, c2log, thr, sim_size)
          ccount <- max(c$num_perf)
          pd[row, 1:6] <- c(TRUE, clog, cval, c2log, thr, ccount)
          row <- row + 1
        }
      }
    }
    thr <- thr - 6
  }
  return(pd)
}

reparam <- function(poke.frame,post) {
  # Edits dataframe variables to more suitable formats for plotting.
  # Args:
  #   poke.frame: Dataframe input.
  #   post: TRUE if conducting conclusion analysis. Allows for ideal catch.type
  #     labels.
  # Returns:
  #   poke.frame: Dataframe output, after variable changes.

  index <- poke.frame$bred == 1
  poke.frame$time[index] <- round(iter/poke.frame$count[index]*1.5, digits = 0)

  index <- poke.frame$bred == 0 & poke.frame$chained == 0
  poke.frame$time[index] <- round(iter/poke.frame$count[index], digits = 0)

  index <- poke.frame$bred == 0 & poke.frame$chained == 4
  poke.frame$time[index] <- round(iter/poke.frame$count[index]*15, digits = 0)

  index <- poke.frame$knot == 1
  poke.frame$knot[index] <- "knot"
  poke.frame$knot[!index] <- "no knot"

  index <- poke.frame$atkst == 1
  poke.frame$atkst[index] <- "attack style"
  poke.frame$atkst[!index] <- "w/o attack style"
```

```
  index <- poke.frame$bred == 1
  poke.frame$bred[index] <- "bred"
  poke.frame$bred[!index] <- "caught"

  index <- poke.frame$chained >= 0
  poke.frame$chained[index] <- as.character(poke.frame$chained[index])

  index <- poke.frame$bred == "caught" & poke.frame$chained == "0"
  poke.frame$catch.type[index] <- "wild"

  index <- poke.frame$bred == "caught" & poke.frame$chained == "4"
  poke.frame$catch.type[index] <- "chained"

  index <- poke.frame$bred == "caught" & poke.frame$chained == "4" &
    poke.frame$atkst == "attack style"
  poke.frame$catch.type[index] <- "chained with attack style"

  if(post) {
    index <- poke.frame$bred=="bred" & poke.frame$knot=="knot" &
      poke.frame$chained=="5"
    poke.frame$catch.type[index] <- "ideal breeding"

    index <- poke.frame$bred=="caught" & poke.frame$chained=="4"
    poke.frame$catch.type[index] <- "ideal chaining"
  }

  return(poke.frame)
}


# Chunk 2

# use the year Pokemon Ruby was released as the seed
set.seed(2003)
iter <- 10 ^ 4
alpha <- frame_gen(iter)


# Chunk 3

beta <- reparam(alpha, FALSE)
gamma <- subset(beta, beta$bred == "bred")

# knot is good
ggplot(gamma, aes(thold, time, color = knot)) +
  geom_jitter() +
  scale_y_log10() +
  geom_smooth(method = "lm") +
  labs(color = "Use of Destiny Knot", x = "IV Sum Threshold", y = "Time",
       title = "IV Sum Threshold vs. Time")

ggplot(gamma, aes(knot, time)) +
  geom_boxplot() +
```

```
  scale_y_log10() +
  labs(x = "Use of Destiny Knot", y = "Time",
       title = "Boxplot of Time by use of Destiny Knot")


# Chunk 4

# atk style matters, just not as much as the knot
# atk style also is generally chosen by the pokemon rather than the player
ggplot(gamma, aes(thold, time, color=atkst)) +
  geom_jitter() +
  scale_y_log10() +
  labs(color = "Attack Style", x = "IV Sum Threshold", y = "Time",
       title = "IV Sum Threshold vs. Time considering Attack Style") +
  geom_smooth(method = "lm")

ggplot(gamma, aes(atkst, time)) +
  geom_boxplot() +
  scale_y_log10() +
  labs(x = "Attack Style", y = "Time",
       title = "Boxplot of time by consideration of Attack Style")


# Chunk 5

# does chaining matter
ggplot(gamma, aes(thold, time, color = chained, size = knot)) +
  geom_jitter() +
  scale_y_log10() +
  labs(color = "Number of Perfect\n IVs from Chaining",
       size = "Use of Destiny Knot", x = "IV Sum Threshold", y = "Time",
       title = "IV Sum Threshold vs. Time by starting IVs and Destiny Knot")

ggplot(gamma, aes(chained, time)) +
  geom_boxplot() +
  scale_y_log10() +
  labs(x = "Number of Perfect IVs from Chaining", y = "Time",
       title = "Boxplot of Perfect IVs from chaining vs. Time")

ggplot(gamma,(aes(time, stat(density)))) +
  geom_histogram() +
  labs(x = "Time", y = "Density",
       title = "Density plot of time to reach Perfect IVs")

for (t in c("0", "4", "5", "6")) {
  cat("Median for chaining to ", t, " IVs: ",
      median(as.numeric(gamma$time[gamma$chained==t])), " minutes.\n")
  }
# how much time saved by chaining: a lot by a 4 chain, a little by a 5 chain


# Chunk 6
```

```r
# odds of 6iv ditto is on gamefaqs (out of the question)
# odds of 5 iv ditto P(A)=2/31
delta <- data.frame(val <- rgeom(10 ^ 5, 2 / 31))
cat("Median time to capture a five IV Pokemon by chaining: ",
    median(delta$val) * 15, " minutes.\n")

# test of the 6 iv ditto for fun
delta2 <- data.frame(val <- rgeom(10 ^ 5, 1 / (31 * 31)))
cat("Median time to capture a six IV Pokemon by chaining: ",
    median(delta2$val) * 15, " minutes.")
# chaining to 5 ditto worth it, but not for 6 ditto


# Chunk 7

# chain catching and wild-caught
epsilon <- subset(beta, beta$bred != "bred")

ggplot(epsilon, aes(thold, time, color = catch.type, shape = atkst)) +
  geom_point(size = 3) +
  scale_y_log10() +
  labs(color = "Capture Method", x = "IV Sum Threshold", y = "Time",
       title = "IV Sum Threshold vs. Time by Capture Method",
       shape = "Attack Style")


# Chunk 8

beta <- reparam(alpha, TRUE)

# now compare knot with attack style, 5 chain against chained with attack style
zeta <- subset(beta, catch.type == "ideal breeding" |
                 catch.type == "ideal chaining")
ggplot(zeta, aes(thold, time, color = catch.type, shape = atkst)) +
  geom_point() +
  scale_y_log10() +
  labs(color = "Acquisition Method", x = "IV Sum Threshold", y = "Time",
       title = "IV Sum Threshold vs. Time by Acquisition and Attack Style",
       shape = "Attack Style")

# conclusion: breed with knot and a 5 chain starting parent to w/e threshold
```