Lab 1

Sawyer Balint

This document is available at https://github.com/sjbalint/BI521/tree/main/scripts/labs

```
#import packages
library(tidyverse) #for data wrangling
library(here) #for filepath management
library(ggsci) #for colors

#custon graphing theme
theme <- list(
  theme_classic(),
  scale_fill_jama(),
  theme(legend.position="none")
)
```

1. "TASK: Estimate the size of your *in plastico* fish population."

```
#import data
plastico.df <- read.csv(here("raw/labs/lab1.csv"))

#estimate N
plastico.df <- plastico.df %>%
  mutate(N=m*K/r)

#calculate mean
plastico_estimate <- plastico.df$N %>%
  mean() %>%
  round(digits=1)
```

We conducted three rounds of mark and recapture. Our mean estimate was 23.7, which compared favorably to the actual population size of 21.

2. "TASK: Using code described above, generate an equivalent data set *in silico* to what you generated *in plastico*. Are the results similar? Decide on a way to plot them to make a graphical comparison."

```
#population size
N <- 21

#empty list to store results
result.list <- list()
```

```
#iterate three times
for (i in 1:3){

  #for each iteration...
  silico.df <- data.frame( #new dataframe
    m = round(runif(N)), #random m
    K = round(runif(N)) #random K
    ) %>%
    mutate(r = ifelse(m == 1 & K == 1, 1, 0)) %>% #calculate r
    summarize_all(sum) %>% #sum
    mutate(N=m*K/r, #calculate N
           iteration=i) #log iteration

  #append result
  result.list <- append(result.list, list(silico.df))

}

#final dataframe
silico.df <- bind_rows(result.list)

silico_estimate <- silico.df$N %>%
  mean() %>%
  round(digits=1)
```

Compared to my *in plastico* experiment, which had a result of 23.7, I got an estimate of 25.4. However, any variation between the two is likely due to the small sample size - I only did three replicates of each experiment.

3. "TASK: Now repeat the *in silico* experiment many more times (let's say five hundred). What is your estimate of the population size? You should get something noticably larger than the actual population size. Try it again! Same result? What gives!?"

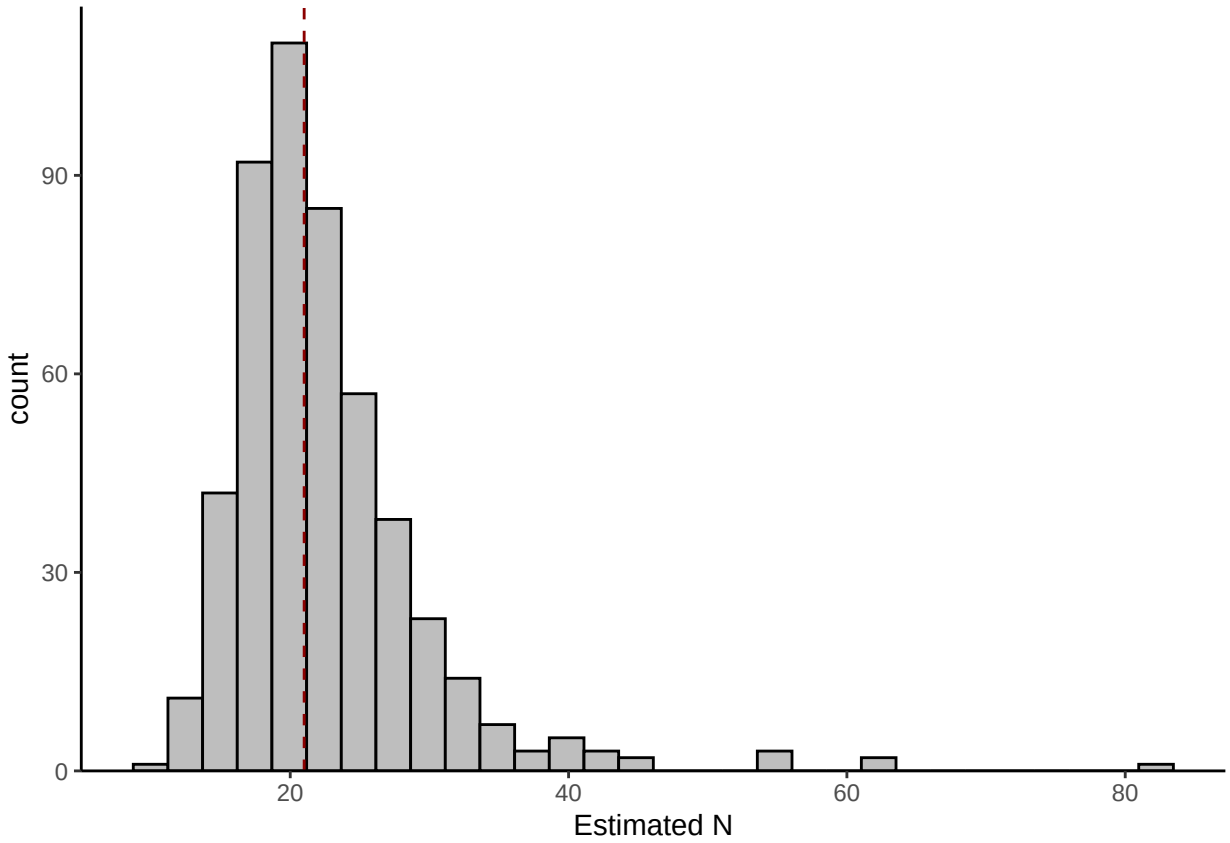
```
#empty list to store results
result.list <- list()

#iterate 500 times
for (i in 1:500){

  #for each iteration...
  silico.df <- data.frame( #new dataframe
    m = round(runif(N)), #random m
    K = round(runif(N)) #random K
    ) %>%
    mutate(r = ifelse(m == 1 & K == 1, 1, 0)) %>% #calculate r
    summarize_all(sum) %>% #sum
    mutate(N=m*K/r, #calculate N
           iteration=i) #log iteration

  #append result
  result.list <- append(result.list, list(silico.df))
```

```
}

#final dataframe
silico.df <- bind_rows(result.list)

#look at the max - probably infinite
max(silico.df$N)
```

```
## [1] Inf
```

```
#remove infinate values
silico.df <- silico.df %>%
  filter(r != 0)

#
new_silico_estimate <- silico.df$N %>%
  median() %>%
  round(digits=1)

#mak a graph
ggplot(silico.df, aes(N)) +
  theme+
  geom_histogram(fill="grey", color="black")+
  geom_vline(xintercept=N, color="darkred", linetype="dashed")+
  scale_y_continuous(expand=expansion(mult=c(0,0.05)))+
  labs(x="Estimated N")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



If no marked fish are recaptured (r = 0), then the estimated population size will be infinity. Thus, we need to exclude estimates for which there are no recaptures. There is still a positive bias in the mean, but if we then calculate the median of the distribution we end up with an estimate of 21 which is pretty accurate!

3. CHALLENGE

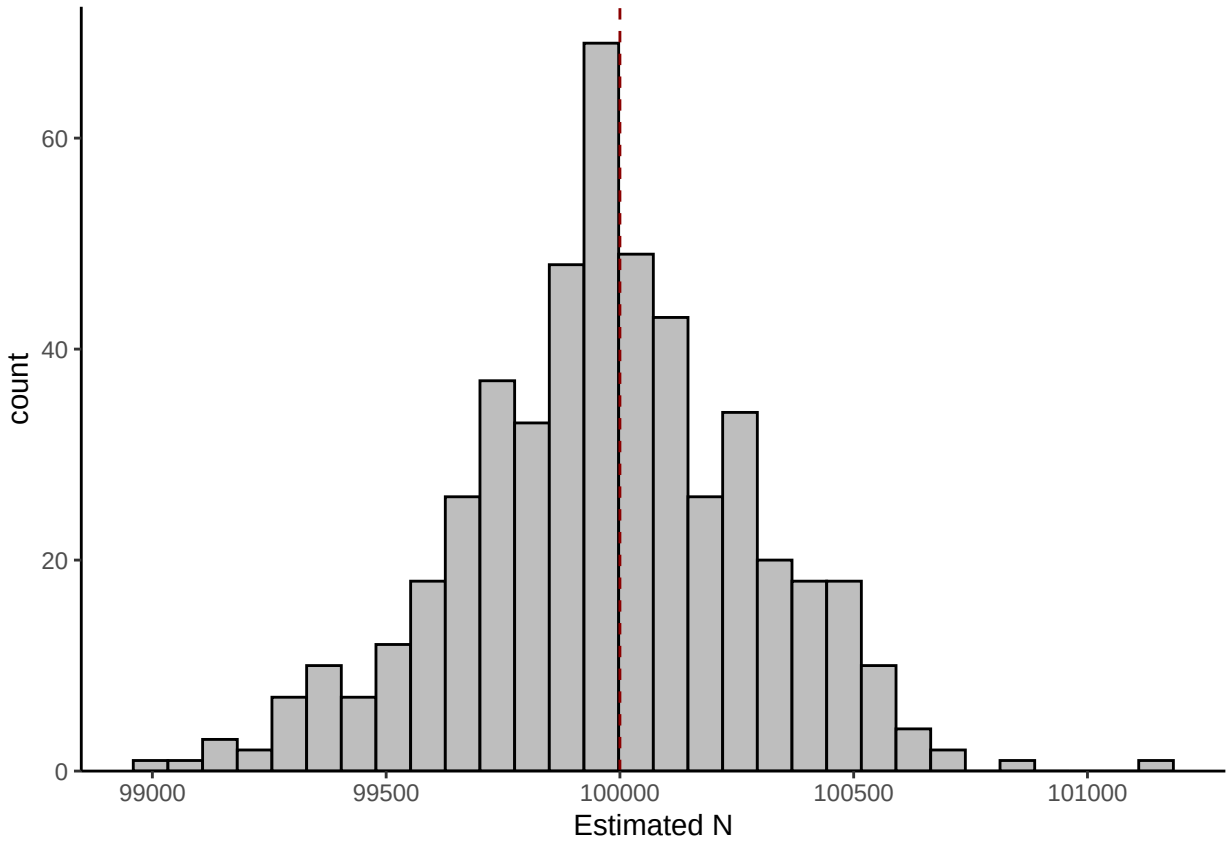
```
#pick a very large N to see if the bias goes away
N <- 100000

#empty list to store results
result.list <- list()

#iterate 500 times
#this takes a while to run with large N
for (i in 1:500){

  #for each iteration...
  silico.df <- data.frame( #new dataframe
    m = round(runif(N)), #random m
    K = round(runif(N)) #random K
    ) %>%
    mutate(r = ifelse(m == 1 & K == 1, 1, 0)) %>% #calculate r
    summarize_all(sum) %>% #sum
    mutate(N=m*K/r, #calculate N
           iteration=i) #log iteration
```

```
  #append result
  result.list <- append(result.list, list(silico.df))

}

#final dataframe
silico.df <- bind_rows(result.list)

#look at the max - this looks better
max(silico.df$N)
```

```
## [1] 101122
```

```
#make a graph
#this graph looks better
ggplot(silico.df, aes(N)) +
  theme+
  geom_histogram(fill="grey", color="black")+
  geom_vline(xintercept=N, color="darkred", linetype="dashed")+
  scale_y_continuous(expand=expansion(mult=c(0,0.05)))+
  labs(x="Estimated N")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

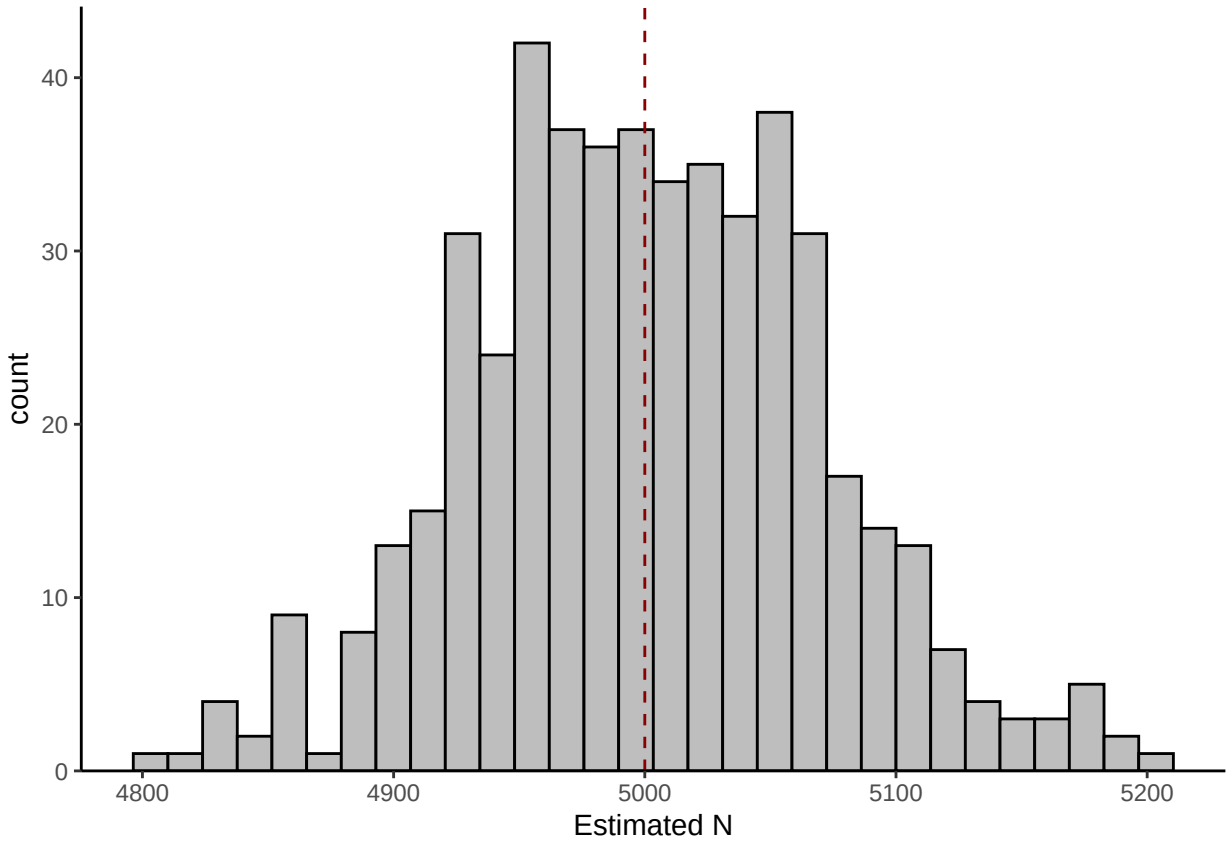


```r
#try again with N=5000, but bayesian derivation
N <- 5000

#empty list to store results
result.list <- list()

#iterate 500 times
#this takes a while to run with large N
for (i in 1:500){

  #for each iteration...
  silico.df <- data.frame( #new dataframe
    m = round(runif(N)), #random m
    K = round(runif(N)) #random K
    ) %>%
    mutate(r = ifelse(m == 1 & K == 1, 1, 0)) %>% #bayesian derivation
    summarize_all(sum) %>% #sum
    mutate(N=(m+1)*(K+1)/(r+1)-1, #calculate N
           iteration=i) #log iteration

  #append result
  result.list <- append(result.list, list(silico.df))

}

#final dataframe
silico.df <- bind_rows(result.list)

#look at the max - this looks better
max(silico.df$N)
```

```
## [1] 5200.958
```

```r
#make a graph
ggplot(silico.df, aes(N)) +
  theme+
  geom_histogram(fill="grey", color="black")+
  geom_vline(xintercept=N, color="darkred", linetype="dashed")+
  scale_y_continuous(expand=expansion(mult=c(0,0.05)))+
  labs(x="Estimated N")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

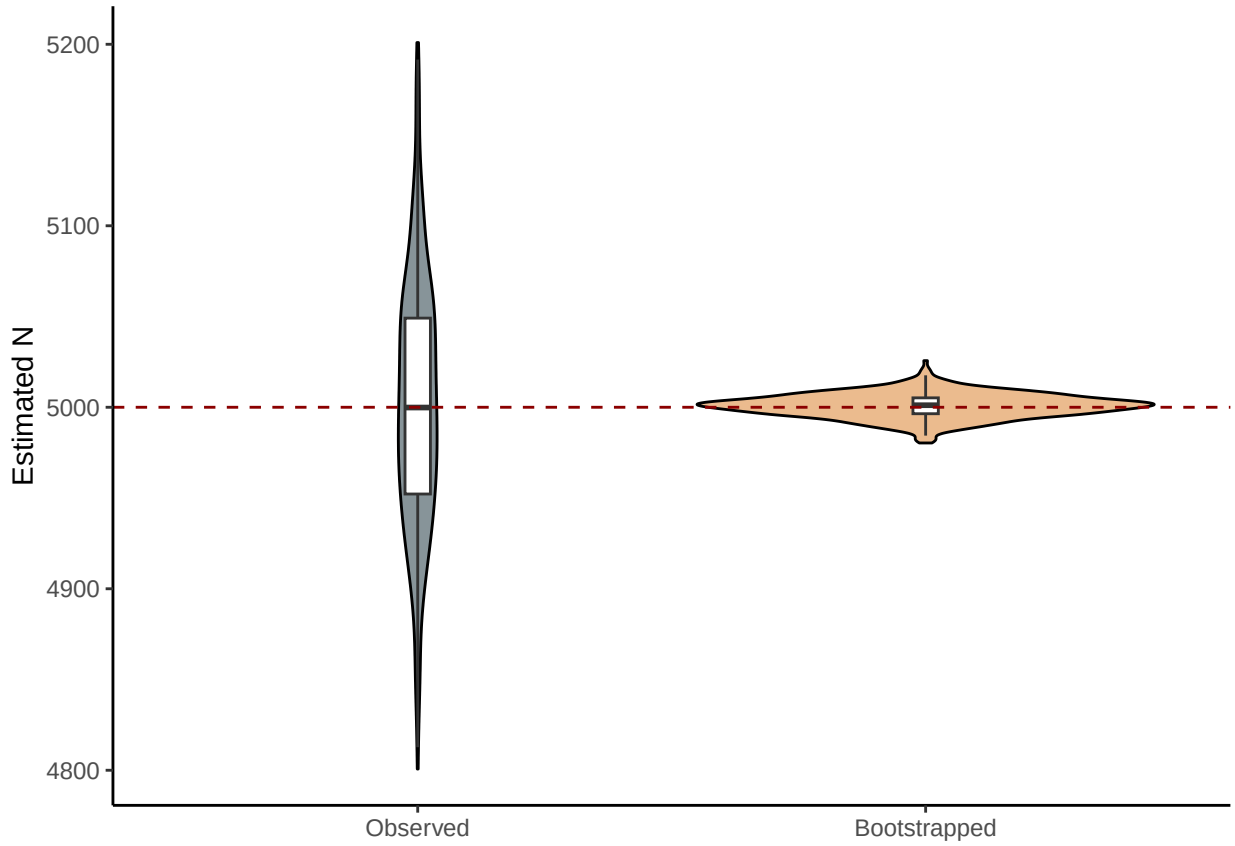


With a very large population size, it becomes nearly impossible for r to approximate zero, and thus the distribution of data is no longer skewed towards high values (divide by very low r). As the fraction of the population (re)captured decreases, this biases becomes more severe. The bayesian derivation aids with this problem.

4. "TASK: Compute estimates of the 0.025 and 0.975 quantiles for an *in silico* mark-recapture experiment"

```
#calculate CI from most recent mark-recapture (N=5000, bayesian derivation)

#empty list to store results
result.list <- list()

#perform bootstrapping
for (i in 1:nrow(silico.df)){
 bootstrap <- sample(silico.df$N, 100, replace=TRUE)
 result.list <- append(result.list, mean(bootstrap))
}

#compile results
bootstrap.list <- as.numeric(result.list)

#calculate quantiles from bootstrap
lower_CI <- quantile(bootstrap.list, probs=0.025) %>%
```

```
  round(digits=1)
upper_CI <- quantile(bootstrap.list, probs=0.975) %>%
  round(digits=1)

#new dataframe for plotting
plot.df <- silico.df %>%
  mutate(N_bootstrap = bootstrap.list) %>%
  select(N, N_bootstrap) %>%
  pivot_longer(cols=c("N", "N_bootstrap")) %>%
  mutate(name=factor(name, levels=c("N","N_bootstrap"),
                     labels=c("Observed","Bootstrapped")))

#visualize the results
ggplot(plot.df, aes(x=name, y=value, fill=name)) +
  theme+
  geom_violin(color="black", alpha=0.6)+
  geom_boxplot(fill="white", width=0.05, outlier.shape=NA)+
  geom_hline(yintercept=N, color="darkred", linetype="dashed")+
  labs(x=NULL, y="Estimated N")
```



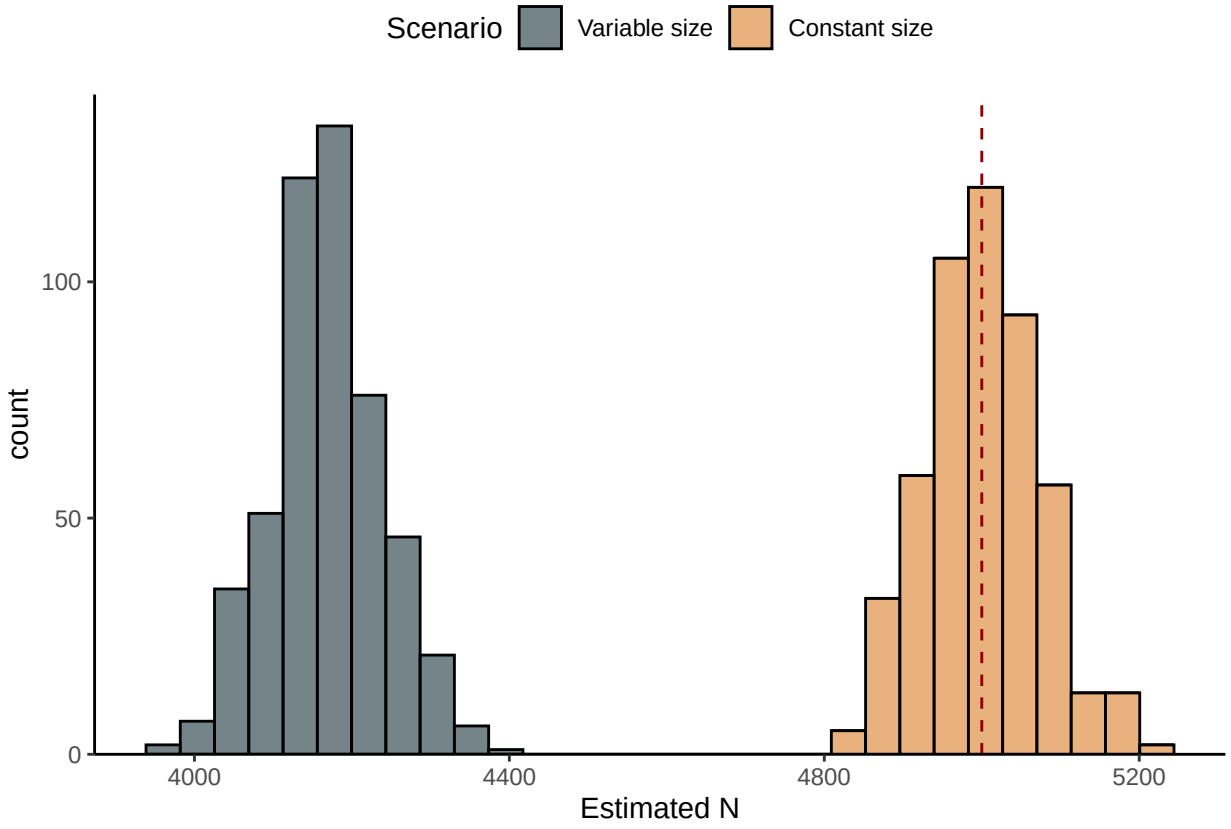
The 0.025 quantile is 4987.5, and the 0.975 quantile is 5014.2. The true population size of 5000 fits comfortably within the confidence intervals, in part because I was using the more accurate bayesian derivation. To further visualize the data, I compared the observed and bootstrapped data using violin plots with integrated boxplots. The population size (5000) is indicated by the dashed red line.

Extension

```
#i'm going to randomly assign fish size, and then bias catchability towards larger individuals
#somewhat larger N
N <- 5000

#empty list to store results
result.list <- list()

#iterate 500 times
#this takes a while to run with large N
for (i in 1:500){

  #for each iteration...
  silico.df <- data.frame( #new dataframe
    size1 = runif(N, min=0.4), #size ranging from 0-1
    m = round(runif(N)), #random m
    K = round(runif(N)) #random K
    ) %>%
    mutate(size2 = mean(size1), #all fish the same size
           m1 = round(m*size1), #scale with size
           K1 = round(K*size1), #scale with size
           m2 = round(m*size2), #size has no effect because it is greater than 0.5
           K2 = round(K*size2), #size has no effect because it is greater than 0.5
           r1 = ifelse(m1 == 1 & K1 == 1, 1, 0),
           r2 = ifelse(m2 == 1 & K2 == 1, 1, 0)) %>%
    summarize(across(c("m","m1","m2","K","K1","K2","r1","r2"), sum),
              across(c("size1","size2"), mean)) %>% #sum
    mutate(N1=(m1+1)*(K1+1)/(r1+1)-1, #bayesian derivation
           N2=(m2+1)*(K2+1)/(r2+1)-1, #bayesian derivation
           iteration=i) #record iteration

  #append result
  result.list <- append(result.list, list(silico.df))

}

#compiled dataframe
silico.df <- bind_rows(result.list)

#manually pivot longer because I'm lazy
#scenario with size varying across individuals
size1.df <- silico.df %>%
  select(size1,m1,K1,r1,N1) %>%
  mutate(scenario="Variable size")

colnames(size1.df) <- c("size","m","K","r","N","scenario")

#scenario with size constant throughout population
size2.df <- silico.df %>%
  select(size2,m2,K2,r2,N2)%>%
  mutate(scenario="Constant size")
```

```r
colnames(size2.df) <- c("size","m","K","r","N","scenario")

#combine together in a loger format
size.df <- bind_rows(size1.df, size2.df) %>%
  mutate(scenario=factor(scenario, levels=c("Variable size","Constant size")))

#plot histogram of results
ggplot(size.df, aes(N, fill=scenario))+
  theme+
  geom_histogram(color="black", alpha=0.7)+
  theme(legend.position="top")+
  geom_vline(xintercept=N, color="darkred",linetype="dashed")+
  scale_y_continuous(expand=expansion(mult=c(0,0.05)))+
  labs(fill="Scenario",
       x="Estimated N")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



In the model above, I tested two scenarios: one where the size of individuals in the population was a constant 0.7, and one where the size of individuals was uniformly distributed between 0.4 and 1.0. In both cases, only fish exceeding a size of 0.5 are able to be caught in the net. Thus, in scenario 1 all of the fish are able to be caught, while in the second scenario only a subset of the population can be caught.

In the figure above, we find that this bias in catchability decreases the estimated N from mark and recapture, because it is only modelling the population of *catchable* fish, rather than all fish. Clearly, a more sophisticated model is required to account for this bias.

REFLECTION QUESTIONS

1. Both *in plastico* and *in silico* models of mark and recapture are highly simplified. What is sorely missing? What is missing but won't necessarily matter?

Both models do not account for variations in catchability due to fish size, age, or location. As shown in my extension model above, catchability can have a large effect on estimated population size. Additionally, the models to not include the natural mortality rate of the fish - it assumes that all of the marked fish are present when the recapture occurs. In reality, some of the population probably dies/migrates out of the habitat/is caught between the two sampling events, and this is not being captured by the models.