Module 4_1

Keaton Wilson

4/1/2020

A new road - bypass the penguins please

The overarching goal of this study builds on the last. We're looking at constructing a new road that gives the fishing team access to some new sites that seem to be relatively leopard-seal free. However, the road crosses through some Gentoo penguin nesting grounds. We have data on a lot of different sites, and want to pick a route that minimizes the impact to the penguins (e.g. we want to pick sites that currently have low nesting success).

We're going to examine our first data set of the module: https://tinyurl.com/w9tol83

Let's explore this data set before moving on to the focus of this lesson.
Load in the data and use the tools we've learned to explore the data. Answer the following questions:
1. What does each column represent
2. What questions do you have about the data?

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------------------------------ tidy
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
penguins = read_csv("https://tinyurl.com/w9tol83")
```

```
## Parsed with column specification:
## cols(
##   chick_ID = col_double(),
##   site = col_character(),
##   avg_tussocks = col_double(),
##   dist_open_water = col_double()
## )
```

```
head(penguins)
```

```
## # A tibble: 6 x 4
##   chick_ID site       avg_tussocks dist_open_water
##      <dbl> <chr>             <dbl>           <dbl>
## 1       33 Cape Royds         7.29            8.46
## 2       35 Cape Royds         7.83           10.1
## 3       63 Cape Royds         7.58            9.70
## 4      104 Cape Royds         6.45            9.78
## 5      116 Cape Royds         7.17            9.95
## 6      127 Cape Royds         7.93            9.18
```

```
glimpse(penguins)
```

```
## Observations: 3,200
## Variables: 4
## $ chick_ID        <dbl> 33, 35, 63, 104, 116, 127, 152, 158, 167, 178, 179,...
## $ site            <chr> "Cape Royds", "Cape Royds", "Cape Royds", "Cape Roy...
## $ avg_tussocks    <dbl> 7.293552, 7.834570, 7.581946, 6.451924, 7.165165, 7...
## $ dist_open_water <dbl> 8.460861, 10.113899, 9.695033, 9.779135, 9.949675, ...
```

Bootstrapping

What would be nice is to have the average number of grass tussocks for each site. We can do this with the group_by and summarize tools we've been working all semester.

```
penguins %>%
  group_by(site) %>%
  summarize(avg_tussocks = mean(avg_tussocks),
            n = n())
```

```
## # A tibble: 20 x 3
##    site                   avg_tussocks     n
##    <chr>                         <dbl> <int>
##  1 Cape Royds                     7.00    170
##  2 Erebus East                   10.0     172
##  3 Erebus West                    2.99    167
##  4 Ferrar Coast                   0.959   171
##  5 Marble Point East              1.02    187
##  6 Marble Point North             7.99    139
##  7 Marble Point South            21.0     168
##  8 Marble Point West              8.96    153
##  9 McMurdo East                  18.0     164
## 10 McMurdo North                  9.99    140
## 11 McMurdo South                 11.0     159
## 12 McMurdo West                  11.1     142
## 13 Ross Island North             15.9     152
## 14 Ross Island Tip                6.07    157
## 15 Scott Base East                5.04    144
## 16 Scott Base North              16.0     166
## 17 Scott Base South              15.0     177
```

```
## 18 Scott Base West              13.0     145
## 19 Taylor Valley Entrance       11.0     153
## 20 Wright Valley Entrance        3.02    174
```

One big question though - how confident are we that these means represent the actual mean for a given site? E.g. we didn't sample every square meter of all of these sites. We took a **sample**, and we assume that the mean here is a good approximation of the actual mean. We can use confidence intervals to give us a little more information about the spread of this value and how confident we are.

Check in and ask about where we have used confidence intervals before

By the bootstraps

Bootstrapping is inherently about resampling without replacement. We take a small data set (all data sets are small, right?) that we assume is representative of a population of samples, and draw out samples with replacement, calculate some statistic (a mean, here), and do it over again. Many, many times.

We can use these samples to create a confidence interval of the statistic (the mean), or even a comparison of means like a t-test.

Video explanation

let's take ~10 minutes to watch the following video: https://www.youtube.com/watch?v=-YgeLJRZQYY. After we've finished, get into your groups and discuss the following questions (and designate a reported to report out to the class):
1. What is the utility of bootstrapping many many times (> 10k)?
2. Why do we sample with replacement?
3. How might we apply this to our data if we wanted to construct means (and confidence intervals) for each site?

Building a loop to do bootstrapping for us

Let's build some code to run a bootstrap for the overall mean number of tussocks across all sites

```
# length vector - how many samples do we have?
n = length(penguins$avg_tussocks)
# number of bootstraps
B = 1000

#what confidence interval do we want?
# here we'll do 95
# 0.95 = 4/2
# 0.80 = 2.23/2
a = 2
# Making an empty vector to pipe values into
result = rep(NA, B)

# looping
for (i in 1:B){
  boot.sample = sample(n, replace = TRUE)
  result[i] = mean(penguins$avg_tussocks[boot.sample])
}
```

```r
# calculating upper and lower bounds
lower = mean(penguins$avg_tussocks) + (-1*a*sd(result))
upper = mean(penguins$avg_tussocks) + (1*a*sd(result))
CI = c(lower, upper)

print(CI)
```

```
## [1] 9.470278 9.875028
```

Group challenge Every site! Expanding the bootstrap

Take the code above and modify/expand it to get a series of means and confidence intervals for each site. Hint: think about nesting one for loop inside of another.

```r
sites = unique(penguins$site)
summaries = list()
for(i in 1:length(sites)){
  df_tmp = penguins %>% filter(site == sites[[i]])
  # length vector - how many samples do we have?
  n = length(df_tmp$avg_tussocks)
  # number of bootstraps
  B = 1000
  # Making an empty vector to pipe values into
  result = rep(NA, B)

  # looping
  for (j in 1:B){
    boot.sample = sample(n, replace = TRUE)
    result[j] = mean(df_tmp$avg_tussocks[boot.sample])
  }
  # calculating upper and lower bounds
  lower = mean(df_tmp$avg_tussocks) + (-1*2*sd(result))
  upper = mean(df_tmp$avg_tussocks) + (1*2*sd(result))
  summary = data.frame(name = sites[i],
                       mean = mean(df_tmp$avg_tussocks),
                       lower = lower,
                       upper = upper)
  summaries[[i]] = summary
}

summary_df = bind_rows(summaries)
```

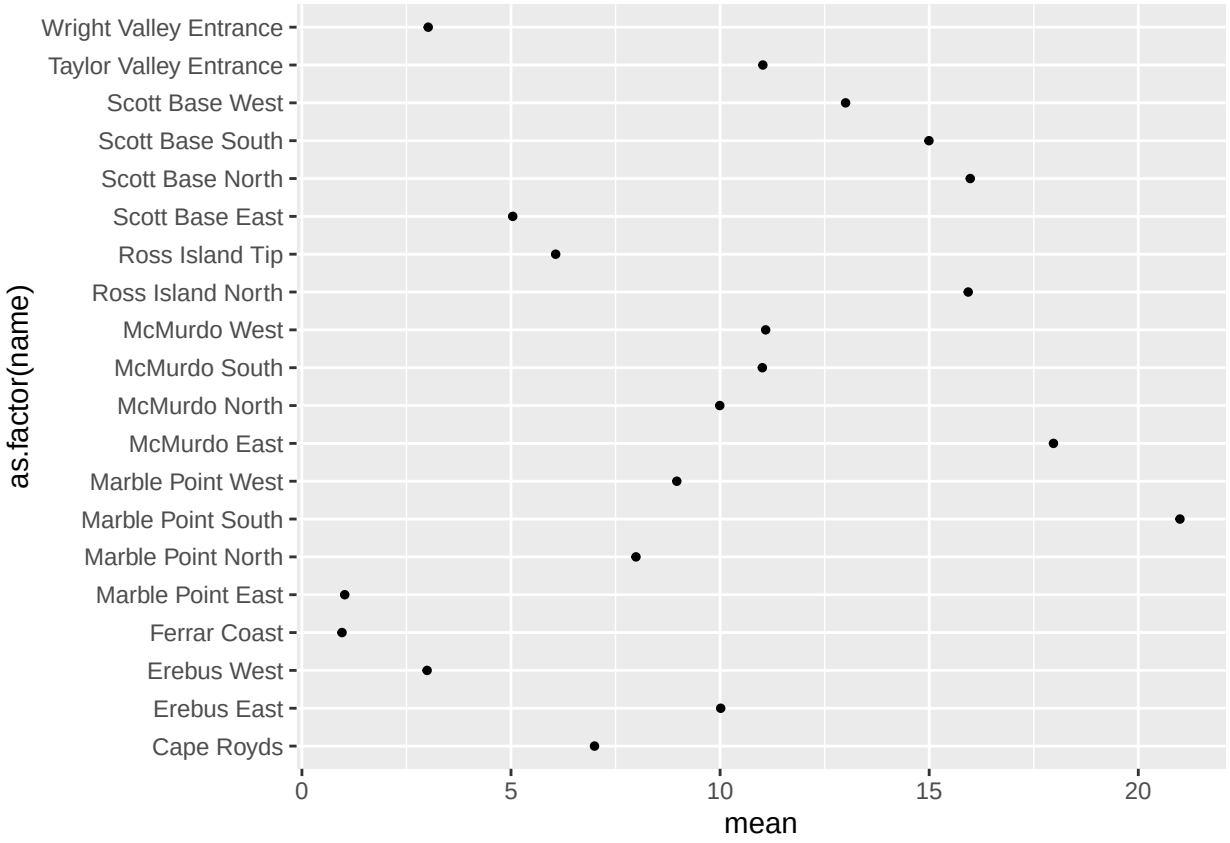
```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
```

```
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector
```

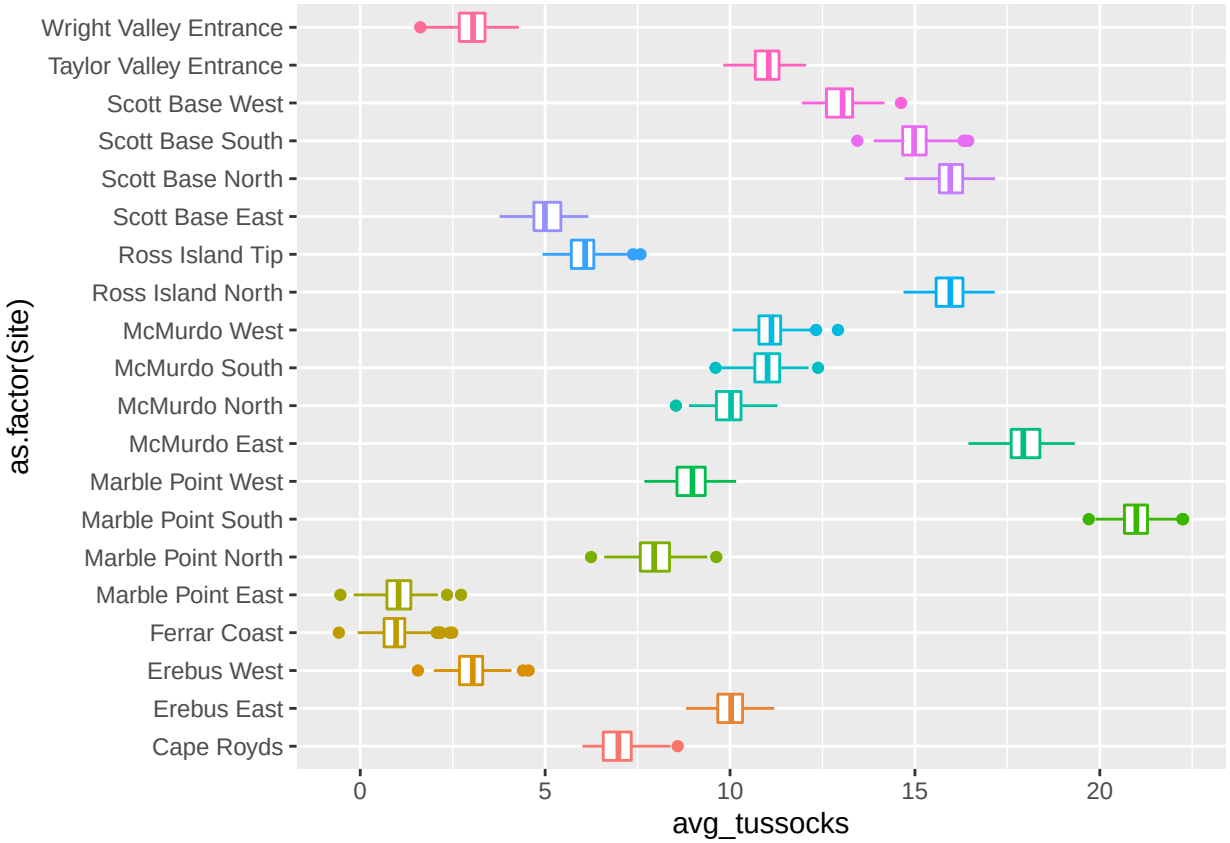
Visualizing these data

Let's finish this mini-project by plotting the means and bounds of our confidence intervals so visualize the set of differences among all the sites.

```
summary_df %>%
  ggplot(aes(y = mean, x = as.factor(name), group = as.factor(name))) +
  geom_point(size = 1) +
  geom_linerange(aes(ymax = upper, ymin = lower)) +
  coord_flip()
```




```
penguins %>%
  ggplot(aes(x = as.factor(site), y = avg_tussocks, color = as.factor(site))) +
  geom_boxplot() +
  coord_flip() +
  theme(legend.position = "none")
```


Wright Valley Entrance
Taylor Valley Entrance
Scott Base West
Scott Base South
Scott Base North
Scott Base East
Ross Island Tip
Ross Island North
McMurdo West
McMurdo South
McMurdo North
McMurdo East
Marble Point West
Marble Point South
Marble Point North
Marble Point East
Ferrar Coast
Erebus West
Erebus East
Cape Royds
as.factor(site)
0
5
10
15
20
avg_tussocks