R Training 1

Rachel Kenny

3/25/2020

General notes

- It's important to first open the "project file" and then the rmarkdown file, because this allows a user to call data without linking it to a path on the computer
- To insert a new code chunk, press the "insert" button on the top right of the screen and then "R", or use the keyboard shortcut [ctrl + alt + i] PC / [cmd + alt + i] MAC
- To see how a document looks altogether, you can hit the "knit" button to knit to html, pdf, or word. In other words, exporting your code as a document

Formatiing

- The [#] symbol in the document section of the Rmarkdown defines headers. One [#] is the biggest header, two is slightly smaller, and so forth. If there is no "#" then it is treated as normal text. However, a [#] used in a code chunk invalidates that line of code, treating it like normal text.
- The [+] symbol in the document denotes a bullet point
- Using asterisks before and after text willl make it *bold*
- Using underscores before and after text willl make it *italic*

Libraries

```
# Packages contain functions that are used to wrangle, analyse, and visualize data. Packages are stored

library(tidyverse) # The tidyverse contains main functions that are useful for data wrangling and manip
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2019c.1.0/
## zoneinfo/America/New_York'
```

```
## -- Attaching packages ----------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## Warning: package 'readr' was built under R version 3.4.4
```

```
## Warning: package 'stringr' was built under R version 3.4.4
```

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

```
library(ggplot2) # The ggplot library is useful for visualizing data

library(rmarkdown) # The rmarkdown library contains code relevant to rmarkdown files, such as knitting

library(readr) # This library helps you read data

library(janitor) # This helps clean data. R does not like spaces and special characters in column names
```

```
## Warning: package 'janitor' was built under R version 3.4.4
```

Within a code chunk

- Use a [#] to tell R not to treat text as code when you are trying to make notes, or highlight the relevant text and hit [ctrl + alt + C] PC / [command + alt + C] MAC
- Character data must always be in quotes. For example if I were looking for "BACSAL" in the data it would need to be in quotes
- Using [<-] names an object (dataframe, plot, etc.) within the code. For example your data file likely has a long name but is easier to refer to as [mitigation_data].
- To run code click "run" in the upper right hand corner and select an option, or hit [ctrl + enter] PC / [cmd + enter] MAC to run one line or selected code
- Functions always use parentheses (). A common error is adding too many or too few

Loading data

```
# To load data, we give it a name, and then read it in according to file type (csv, excel, etc.)

oak_data_raw <- read_csv("Weir_Oak_Restoration_Data_winter19.csv")
```

```
## Parsed with column specification:
## cols(
##   `Short ID` = col_character(),
##   Survival = col_logical(),
##   Quantity = col_double(),
##   `Height (cm)` = col_double(),
##   `Open Closed` = col_character(),
##   `Location UML` = col_character(),
##   `Water Yes No` = col_character(),
##   `Sampling Group` = col_character()
## )
```

```
View(oak_data_raw)

# Pipe operators [%>%] are used to do additional functions without writing unnecessary lines of code. T

oak_data <- oak_data_raw %>%
  clean_names()

View(oak_data)
```

##Common operators + [*] multiply + [/] divide + [>] greater than + [>=] greater than or equal to + [<] less than + [<=] less than or equal to + [=] equal to + [==] find a match in the data (for example, when filtering)

Wrangling data

```
# To reveal column names, use the [names] function

names(oak_data)
```

```
## [1] "short_id"       "survival"       "quantity"       "height_cm"
## [5] "open_closed"    "location_uml"   "water_yes_no"   "sampling_group"
```

```
# To only retain certain columns, use the function [select]

oak_data_selection <- oak_data %>%
  select(survival, height_cm)

View(oak_data_selection)

# To filter data, use the function [filter]

oak_data_survived <- oak_data %>%
  filter(survival == "TRUE", height_cm >= 6)

View(oak_data_survived) # Now we only have seedlings that survived and are greater than or equal to 6 c
```

Visualizing data

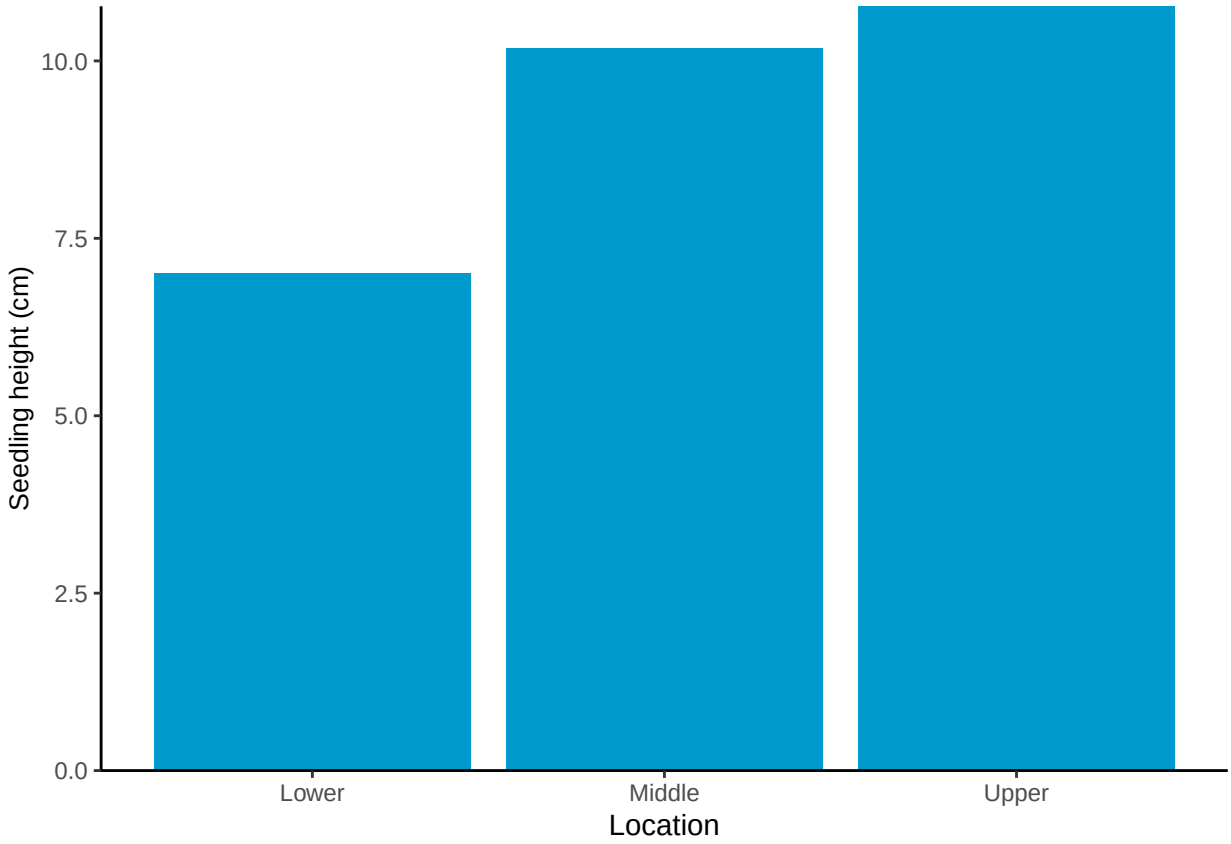
```
# ggplot allows us to visualize our data in a number of ways. Here's an example below.

# Differences in seedling height between acorns planted at three elevations: upper, middle, and lower
oak_plot <- ggplot(oak_data) +
  geom_bar(fill='deepskyblue3', aes(x = location_uml, y = height_cm),
           position = "dodge", stat = "summary", fun.y = "mean") +
  theme_classic() +
  xlab("Location")+
  ylab("Seedling height (cm)")+
  scale_x_discrete(expand=c(0.3,0))+
  scale_y_continuous(expand=c(0,0))+
  theme(axis.title.y = element_text(size = 10))
```

```
## Warning: Ignoring unknown parameters: fun.y
```

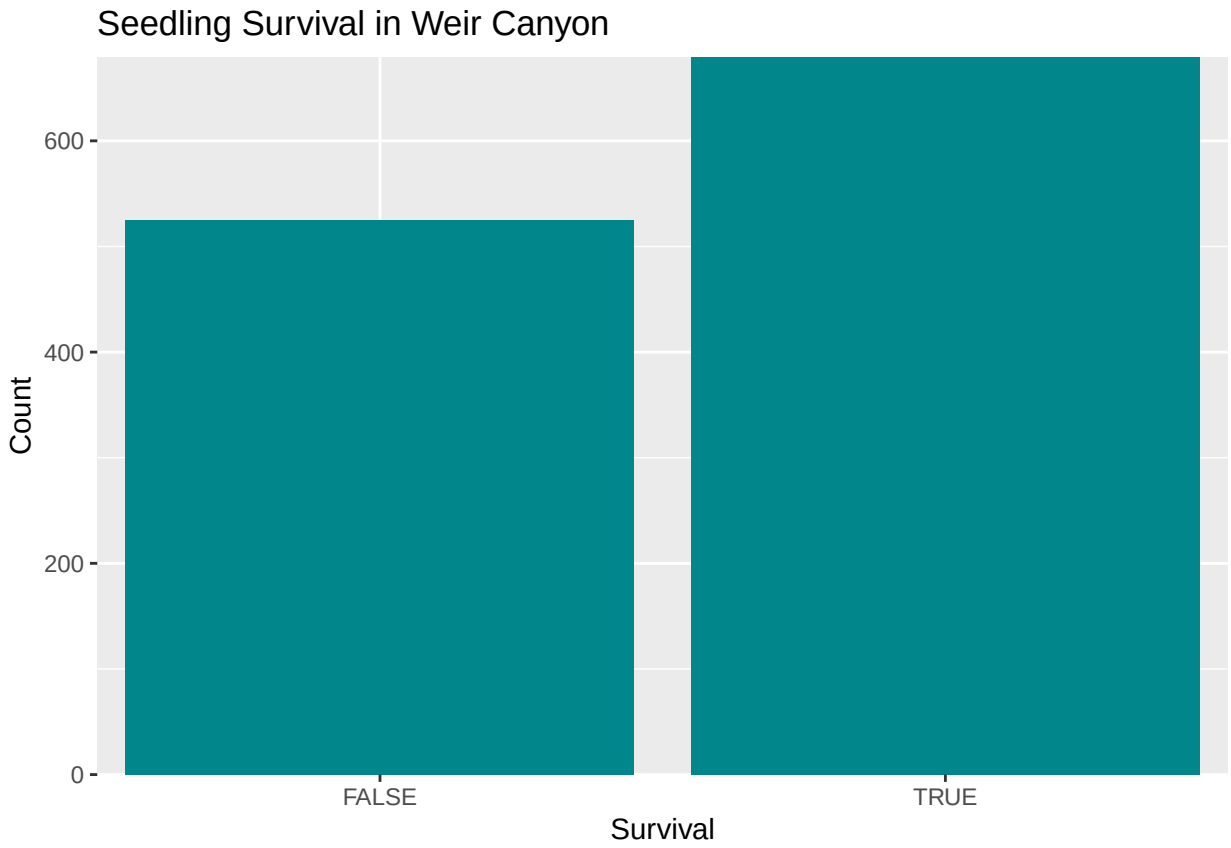
```
oak_plot
```

```
## Warning: Removed 1 rows containing non-finite values (stat_summary).
```

```
## No summary function supplied, defaulting to `mean_se()`
```




```
# This plot shows a count of the number of seedlings that survived vs didn't

oak_plot2 <- ggplot(oak_data, aes(survival)) +
  geom_bar(fill="turquoise4") +
  xlab("Survival") +
  ylab("Count") +
  ggtitle("Seedling Survival in Weir Canyon") +
  scale_x_discrete(expand=c(0.5,0))+
  scale_y_continuous(expand=c(0,0))

oak_plot2
```


Seedling Survival in Weir Canyon
600
400
200
0
Count
FALSE
TRUE
Survival