WDV - Intro to data analysis and visualization in R

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Before we start

- Open RStudio
- Under the File menu, click on New project, choose New directory, then New project
- Enter a name for this new folder (or "directory"), and choose a convenient location for it. This will be your working directory for the rest of the day (e.g., /WDV-R)
- Click on Create project
- Under the Files tab on the right of the screen, click on New Folder and create a folder named data within your newly created working directory (e.g., /WDV-R/data)

Notes on R R is case senstive, so animal_sex is treated differently from Animal_Sex

Introduction

R is a language and environment for statistical computing and graphics. **RStudio** is currently a very popular way to not only write your R scripts but also to interact with the R software. To function correctly, RStudio needs R and therefore both need to be installed on your computer.

Outlines:

- Why R nd RStudio
- Navigate RStudio
- Organize your files
- Use the built-in RStudio help interface to search for more information on * R functions.
- Basics of R -
  a. Create objects and and assign values to them.
  b. Use comments to inform script.
  c. Do simple arithmetic operations in R using values and objects.
  d. Call functions and use arguments to change their default options.
- Getting data to R
- Manipulating data frames
  a. What is data frames
  b. factors and vectors
  c. maipulating data frames
  d. What are levels of categorical data. How to change the name of a level
- Ploting in R

Learning Objectives

- Describe the purpose of the RStudio Script, Console, Environment, and Plots panes.
- Organize files and directories for a set of analyses as an R Project, and understand the purpose of the working directory.
- Use the built-in RStudio help interface to search for more information on R functions.
- Demonstrate how to provide sufficient information for troubleshooting with the R user community.
- Describe what a data frame is.
- Load external data from a .csv file into a data frame in R.

- Summarize the contents of a data frame in R.
- Manipulate categorical data in R.

Navigate RStudio

RStudio is divided into 4 “Panes”: the Source for your scripts and documents (top-left, in the default layout), the R Console (bottom-left), your Environment/History (top-right), and your Files/Plots/Packages/Help/Viewer (bottom-right). The placement of these panes and their content can be customized (see menu, Tools -> Global Options -> Pane Layout).

Introduction to R

R is great with mathematical operations and statistical analysis. You can use use as a calculator (which means doing simple tasks like adding two numbers) till doing sophisticated analysis on the data.

```
#Type this in the consol and run it
3 + 5 * 2
```

```
## [1] 13
```

```
#Note that just like python this is comment  Anything that follows after the hash (or octothorpe) symbo
```

Variables and assignment We can store values in variables using the assignment operator <-, like this:b <- 5

```
# assign 3 to a
a <- 3
# assign 5 to b
b <- 5

# what now is a
a
```

```
## [1] 3
```

```
# what now is b
b
```

```
## [1] 5
```

```
#Add a and b
a + b
```

```
## [1] 8
```

Mathematical functions

R has many built in mathematical functions. To call a function, we simply type its name, followed by open and closing parentheses. Anything we type inside the parentheses is called the function’s arguments.

```
#functions
sqrt(4)
```

```
## [1] 2
```

```
round(3.149847848)
```

```
## [1] 3
```

```r
round(3.149847848,2)
```

```
## [1] 3.15
```

```r
log(1)
```

```
## [1] 0
```

Don't worry about trying to remember every function in R. You can simply look them up on Google, or if you can remember the start of the function's name, use the tab completion in RStudio.

R, and every package, provide help files for functions. The general syntax to search for help on any function, "function_name", from a specific function that is in a package loaded into your namespace (your interactive R session): ?function_name help(function_name) This will load up a help page in RStudio (or as plain text in R by itself).

Each help page is broken down into sections:

- Description: An extended description of what the function does.
- Usage: The arguments of the function and their default values.
- Arguments: An explanation of the data each argument is expecting.
- Details: Any important details to be aware of.
- Value: The data the function returns.
- See Also: Any related functions you might find useful.
- Examples: Some examples for how to use the function.
- Different functions might have different sections, but these are the main ones you should be aware of.

```r
#args(round)

#If you need help with a specific function, let's say sqrt(), you can type:
?sqrt
#?barplot
#If you just need to remind yourself of the names of the arguments, you can use:
args(lm)
```

```
## function (formula, data, subset, weights, na.action, method = "qr",
##     model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
##     contrasts = NULL, offset, ...)
## NULL
```

```r
#If the function is part of a package that is installed on your computer but don't remember which one,
??geom_point
#If you are looking for a function to do a particular task
help.search("kruskal")
```

Installong packages

```r
#install.packages("dplyr")
#dplyr is a package for making tabular data manipulation easier
```

Presentation of the Survey Data

We will study the species and weight of animals data. The dataset is stored as a comma separated value (CSV) file. Each row holds information for a single animal, and the columns represent.

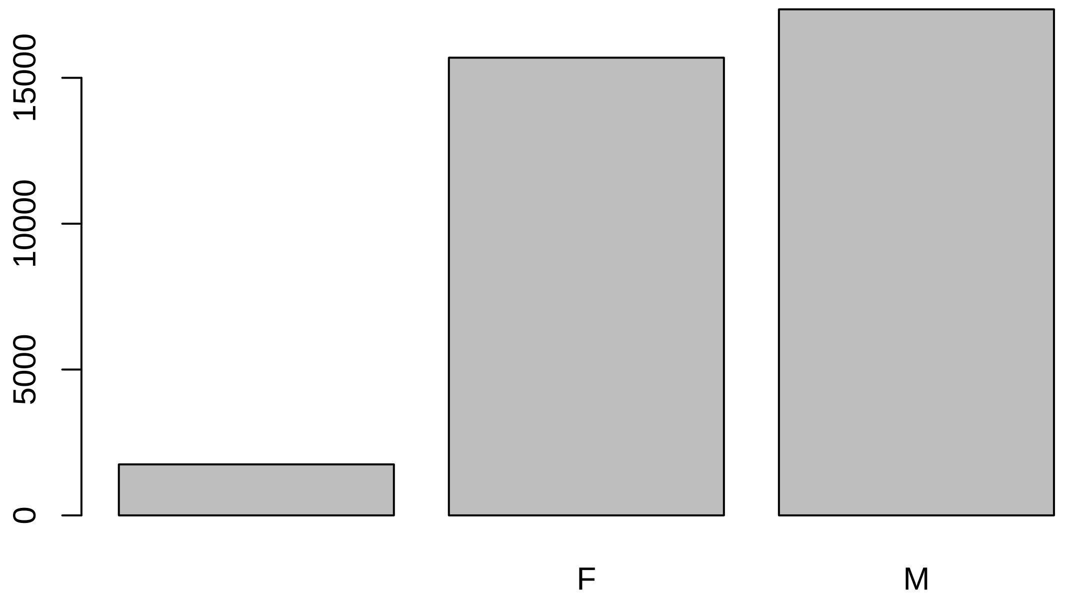
Download and Read Data from the Web

```r
#We are going to use the R function download.file() to download the CSV file that contains the survey d

#download.file("https://ndownloader.figshare.com/files/2292169", "data/survey_data.csv")  #the function
#we will use read.csv() to load into memory the content of the CSV file as an object of class data.fram
#Read the data and assign it to a data frame
survey_data <- read.csv('data/survey_data.csv')
#survey_data
str(survey_data)
```

```
## 'data.frame':    34786 obs. of  13 variables:
##  $ record_id      : int  1 72 224 266 349 363 435 506 588 661 ...
##  $ month          : int  7 8 9 10 11 11 12 1 2 3 ...
##  $ day            : int  16 19 13 16 12 12 10 8 18 11 ...
##  $ year           : int  1977 1977 1977 1977 1977 1977 1977 1978 1978 1978 ...
##  $ plot_id        : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ species_id     : Factor w/ 48 levels "AB","AH","AS",..: 16 16 16 16 16 16 16 16 16 16 ...
##  $ sex            : Factor w/ 3 levels "","F","M": 3 3 1 1 1 1 1 1 3 1 ...
##  $ hindfoot_length: int  32 31 NA NA NA NA NA NA NA NA ...
##  $ weight         : int  NA NA NA NA NA NA NA NA 218 NA ...
##  $ genus          : Factor w/ 26 levels "Ammodramus","Ammospermophilus",..: 13 13 13 13 13 13 13 13 
##  $ species        : Factor w/ 40 levels "albigula","audubonii",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ taxa           : Factor w/ 4 levels "Bird","Rabbit",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ plot_type      : Factor w/ 5 levels "Control","Long-term Krat Exclosure",..: 1 1 1 1 1 1 1 1 1 1
```

Download and Read Data from the Web then plot animal gender

```r
#We are going to use the R function download.file() to download the CSV file that contains the survey d

#download.file("https://ndownloader.figshare.com/files/2292169", "data/portal_data_joined.csv")

#we will use read.csv() to load into memory the content of the CSV file as an object of class data.fram
#Read the data and assign it to a data frame
survey_data <- read.csv('data/portal_data_joined.csv')
#Plot animal gender
plot(survey_data$sex)
```



Data frames

What are data frames? Data frames are the de facto data structure for most tabular data, and what we use for statistics and plotting. A data frame is the representation of data in the format of a table where the columns are **vectors** that all have the same length. Because the column are vectors, they all contain the same type of data (e.g., characters, integers, factors).

Operations on data frame

```r
#Summary about the data
str(survey_data)
```

```
## 'data.frame':    34786 obs. of  13 variables:
##  $ record_id      : int  1 72 224 266 349 363 435 506 588 661 ...
##  $ month          : int  7 8 9 10 11 11 12 1 2 3 ...
##  $ day            : int  16 19 13 16 12 12 10 8 18 11 ...
##  $ year           : int  1977 1977 1977 1977 1977 1977 1977 1978 1978 1978 ...
##  $ plot_id        : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ species_id     : Factor w/ 48 levels "AB","AH","AS",..: 16 16 16 16 16 16 16 16 16 16 ...
##  $ sex            : Factor w/ 3 levels "","F","M": 3 3 1 1 1 1 1 1 3 1 ...
##  $ hindfoot_length: int  32 31 NA NA NA NA NA NA NA NA ...
##  $ weight         : int  NA NA NA NA NA NA NA NA 218 NA ...
##  $ genus          : Factor w/ 26 levels "Ammodramus","Ammospermophilus",..: 13 13 13 13 13 13 13 13
##  $ species        : Factor w/ 40 levels "albigula","audubonii",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ taxa           : Factor w/ 4 levels "Bird","Rabbit",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ plot_type      : Factor w/ 5 levels "Control","Long-term Krat Exclosure",..: 1 1 1 1 1 1 1 1 1 1
```

```r
#size of the data
dim(survey_data)
```

```
## [1] 34786    13
```

```r
nrow(survey_data)
```

```
## [1] 34786
```

```r
ncol(survey_data)
```

```
## [1] 13
```

```r
#content
head(survey_data) #shows the first 6 rows
```

```
##   record_id month day year plot_id species_id sex hindfoot_length weight
## 1         1     7  16 1977       2         NL   M              32     NA
## 2        72     8  19 1977       2         NL   M              31     NA
## 3       224     9  13 1977       2         NL                  NA     NA
## 4       266    10  16 1977       2         NL                  NA     NA
## 5       349    11  12 1977       2         NL                  NA     NA
## 6       363    11  12 1977       2         NL                  NA     NA
##     genus  species   taxa plot_type
## 1 Neotoma albigula Rodent   Control
## 2 Neotoma albigula Rodent   Control
## 3 Neotoma albigula Rodent   Control
## 4 Neotoma albigula Rodent   Control
## 5 Neotoma albigula Rodent   Control
## 6 Neotoma albigula Rodent   Control
```

```r
tail(survey_data) #shows the last 6 rows
```

```
##       record_id month day year plot_id species_id sex hindfoot_length
## 34781     26787     9  27 1997       7         PL   F              21
## 34782     26966    10  25 1997       7         PL   M              20
## 34783     27185    11  22 1997       7         PL   F              21
## 34784     27792     5   2 1998       7         PL   F              20
## 34785     28806    11  21 1998       7         PX                  NA
## 34786     30986     7   1 2000       7         PX                  NA
##       weight       genus  species   taxa        plot_type
## 34781     16  Peromyscus leucopus Rodent Rodent Exclosure
## 34782     16  Peromyscus leucopus Rodent Rodent Exclosure
## 34783     22  Peromyscus leucopus Rodent Rodent Exclosure
## 34784      8  Peromyscus leucopus Rodent Rodent Exclosure
## 34785     NA Chaetodipus      sp. Rodent Rodent Exclosure
## 34786     NA Chaetodipus      sp. Rodent Rodent Exclosure
```

```r
#column name
names(survey_data)
```

```
##  [1] "record_id"       "month"           "day"
##  [4] "year"            "plot_id"         "species_id"
##  [7] "sex"             "hindfoot_length" "weight"
## [10] "genus"           "species"         "taxa"
## [13] "plot_type"
```

Sometimes you want to retrieve or present a specific column or row of your data frame. For example, I collected data and I only want to work with the age column//vector for now, how I can do this in R. In excel you select the column and plot or do operations, in R you call the function that select a column for you.

```r
#Select a column
head(survey_data$sex)
```

```
## [1] M M
## Levels:  F M
```

```r
#select more than one column. You can do this using the square brackets.
head(survey_data[, c("species_id", "sex")])
```

```
##   species_id sex
## 1         NL   M
## 2         NL   M
## 3         NL
## 4         NL
## 5         NL
## 6         NL
```

```r
#you also can call/retrieve specific subset of the data, specific element or column or row using the in
#[row, column]

survey_data[1, 2]   # first element in the 2nd column of the data frame
```

```
## [1] 7
```

```r
survey_data[1, 6]   # first element in the 6th column
```

```
## [1] NL
## 48 Levels: AB AH AS BA CB CM CQ CS CT CU CV DM DO DS DX NL OL OT OX ... ZL
```

```r
survey_data[1:3, 7] # first three elements in the 7th column
```

```
## [1] M M
## Levels:  F M
```

```r
survey_data[3, ]    # the 3rd element for all columns
```

```
##   record_id month day year plot_id species_id sex hindfoot_length weight
## 3       224     9  13 1977       2         NL                  NA     NA
##     genus  species   taxa plot_type
## 3 Neotoma albigula Rodent   Control
```

```r
#survey_data[, 7]    # the entire 7th column
#head_survey_data <- survey_data[1:6, ] # metadata[1:6, ] is equivalent to head(metadata)


#You can even access columns by column name and select specific rows of interest. For example, if we wa

head(survey_data[4:7, c("species_id", "sex")])
```

```
##   species_id sex
## 4         NL
## 5         NL
## 6         NL
## 7         NL
```

When the data of a vector is categorical data, the class type of this vector is called **factor**. Once created, factors can only contain a pre-defined set of values, known as levels. By default, R always sorts levels in alphabetical order. For instance, if you have a factor with 2 levels:

```r
#sex <- factor(c("male", "female", "female", "male"))

levels(survey_data$sex)
```

```
## [1] ""  "F" "M"
```

```r
nlevels(survey_data$sex)
```

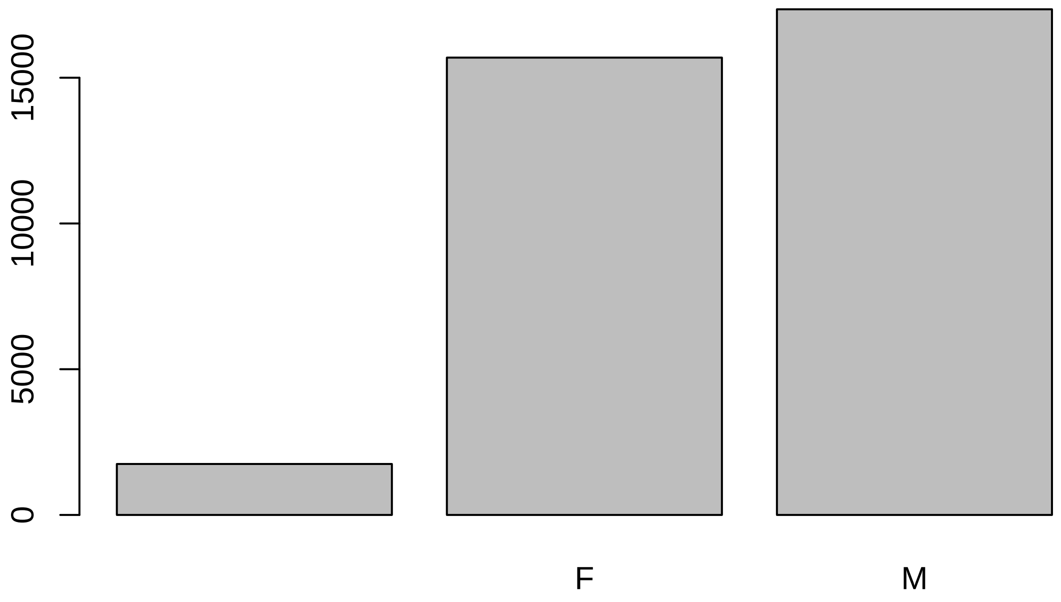
```
## [1] 3
```

Plotting in R

Renaming factors

When your data is stored as a factor, you can use the plot() function to get a quick glance at the number of observations represented by each factor level. Let's look at the number of males and females captured over the course of the experiment:

```r
plot(survey_data$sex)
```



In addition to males and females, there are about 1700 individuals for which the sex information hasn't been recorded. Additionally, for these individuals, there is no label to indicate that the information is missing. Let's rename this label to something more meaningful. Before doing that, we're going to pull out the data on sex and work with that data, so we're not modifying the working copy of the data frame:

```r
levels(survey_data$sex) # print the categorical data of the animal sex
```

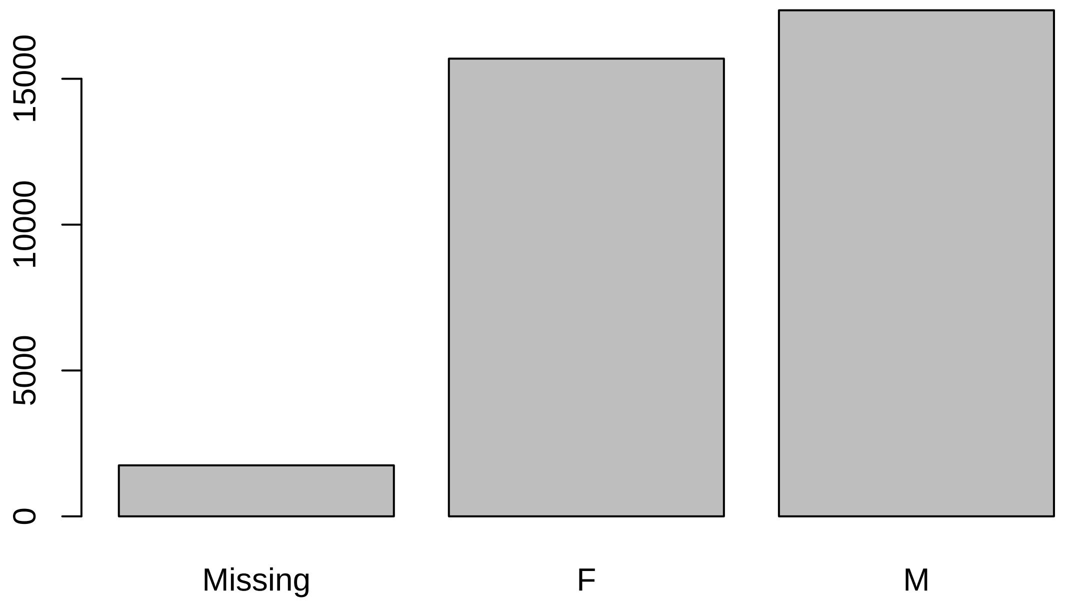
```
## [1] ""  "F" "M"
```

```r
levels(survey_data$sex)[1] <- "Missing"  # For the first item in the levels vector, rename it to missin

# print the categorical data of the animal sex again
levels(survey_data$sex)
```

```
## [1] "Missing" "F"       "M"
```

Now let us plot the survey sex of animal species again

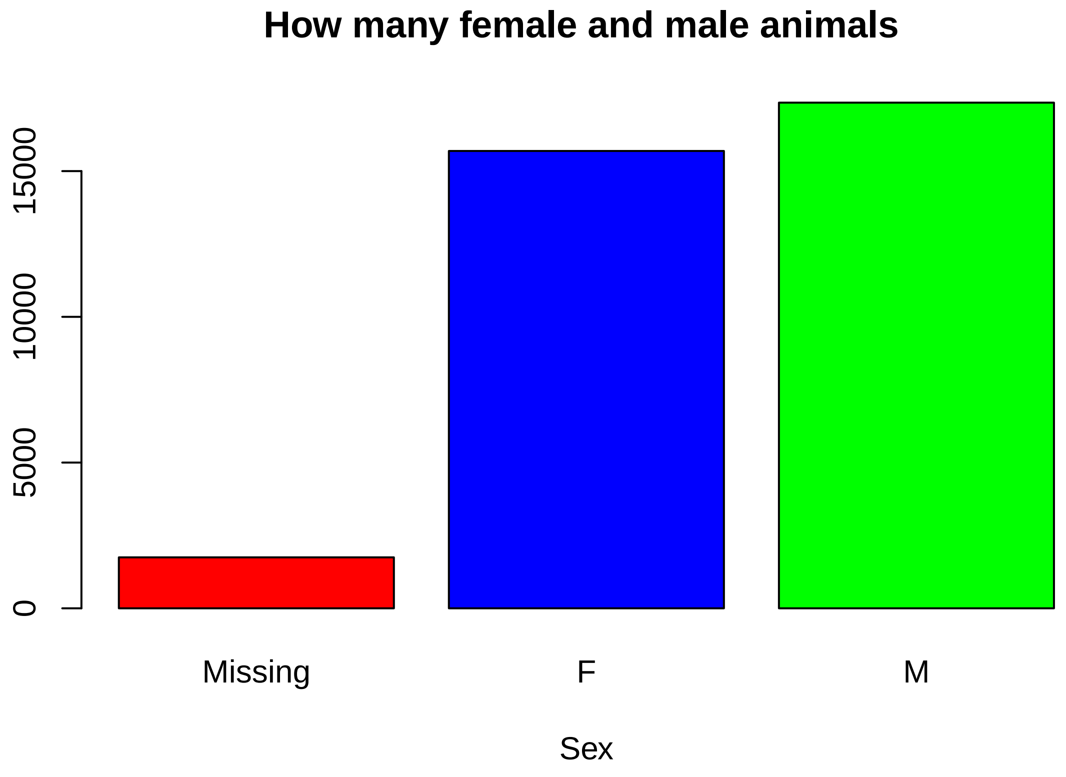
```r
plot(survey_data$sex)
```



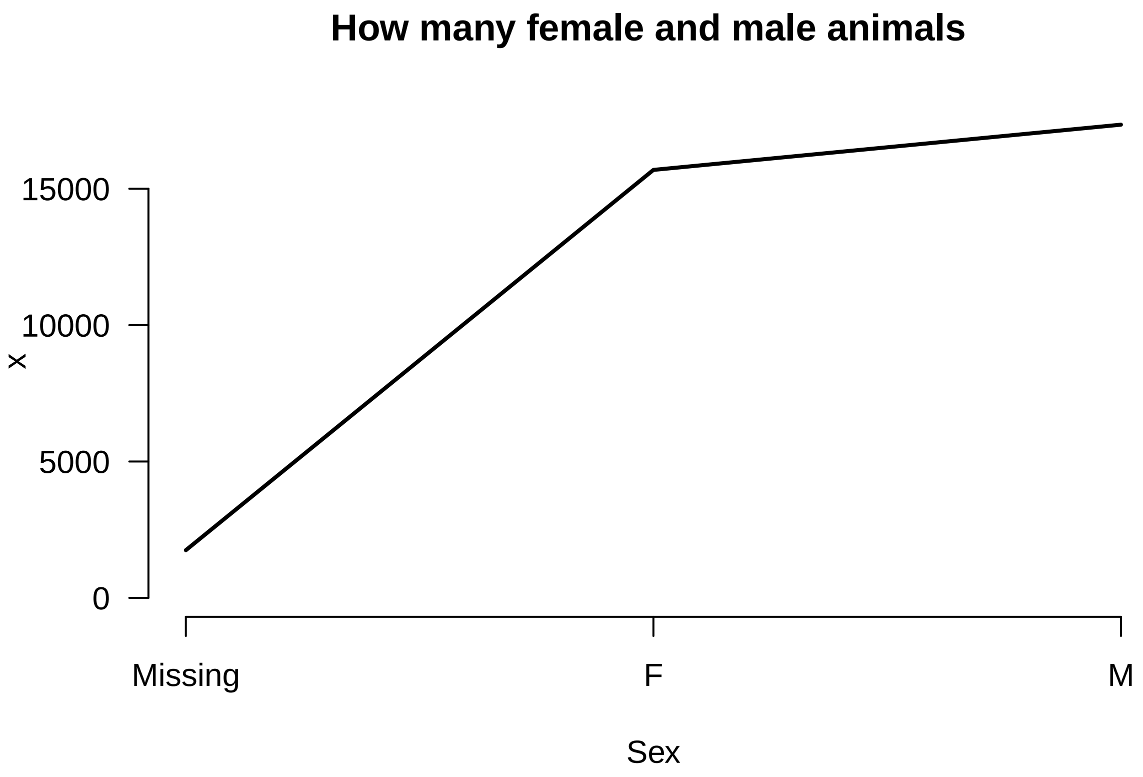
Question: resname F to be Female and M to be Male.

```r
#Add your answer here
```

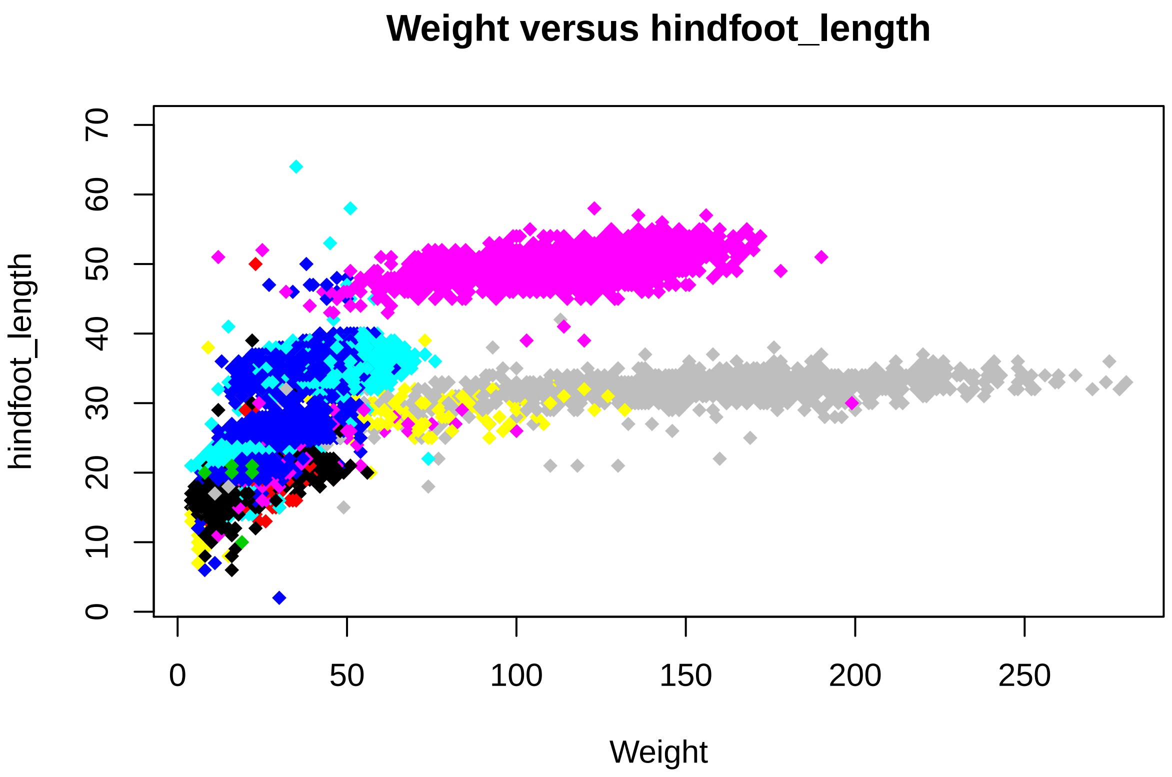
```r
#Let us pass some arguments: change labels for the axes and title with xlab and ylab
#main is for setting a title for the plot
#col is a function that takes colors to be assigned to each category of the data


plot(survey_data$sex, xlab = "Sex", main="How many female and male animals ", col=c("red", "blue", "gre
```



```r
#Change the plot type
x <- table(survey_data$sex)
plot(x, xlab = "Sex", main="How many female and male animals ",type="l" , las=1)
```



```r
plot(survey_data$weight , survey_data$hindfoot_length, xlab = "Weight", ylab="hindfoot_length", main="We
```



```
#change the symbols used (pch)
```

Save files in R

```
pdf("figures/animals_gender.pdf")

plot(survey_data$sex, xlab = "Sex", main="How many female and male animals ", type="o" )
```

```
## Warning in plot.window(xlim, ylim, log = log, ...): graphical parameter
## "type" is obsolete

## Warning in axis(if (horiz) 2 else 1, at = at.l, labels = names.arg, lty =
## axis.lty, : graphical parameter "type" is obsolete

## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## graphical parameter "type" is obsolete

## Warning in axis(if (horiz) 1 else 2, cex.axis = cex.axis, ...): graphical
## parameter "type" is obsolete
```

```
dev.off()
```

```
## pdf
##   2
```

Aggregating and analyzing data with dplyr

Add new columns to a data frame that are functions of existing columns with mutate.

```
#install.packages("dplyr")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#Selecting columns and filtering rows

#select(survey_data, plot_id, species_id, weight)
#filter(survey_data, year == 1995)


#Pipes - Select certain rows in a data frame according to filtering conditions with the dplyr function 

survey_data %>%
  filter(weight < 5) %>%
  select(species_id, sex, weight) %>%
  head
```

```
##   species_id     sex weight
## 1         PF       F      4
## 2         PF       F      4
## 3         PF       M      4
## 4         RM       F      4
## 5         RM       M      4
## 6         PF Missing      4
```

```
#------------------#

#In the above, we use the pipe to send the surveys dataset first through filter() to keep rows where we

#If we wanted to create a new object with this smaller version of the data we could do so by assigning

surveys_sml <- survey_data %>%
  filter(weight < 5) %>%
  select(species_id, sex, weight) %>%
  head

#surveys_sml



#------------------#
#Mutate
#Link the output of one dplyr function to the input of another function with the 'pipe' operator %>%.
#To create a new column of weight in kg:

survey_data %>%
  mutate(weight_kg = weight / 1000) %>%
  head
```

```
##   record_id month day year plot_id species_id     sex hindfoot_length
## 1         1     7  16 1977       2         NL       M              32
## 2        72     8  19 1977       2         NL       M              31
## 3       224     9  13 1977       2         NL Missing              NA
## 4       266    10  16 1977       2         NL Missing              NA
## 5       349    11  12 1977       2         NL Missing              NA
## 6       363    11  12 1977       2         NL Missing              NA
##   weight   genus  species   taxa plot_type weight_kg
## 1     NA Neotoma albigula Rodent   Control        NA
## 2     NA Neotoma albigula Rodent   Control        NA
## 3     NA Neotoma albigula Rodent   Control        NA
## 4     NA Neotoma albigula Rodent   Control        NA
## 5     NA Neotoma albigula Rodent   Control        NA
## 6     NA Neotoma albigula Rodent   Control        NA
```

```
#you can use a pipe to view the head()
survey_data %>%
  mutate(weight_kg = weight / 1000) %>%
  head
```

```
##   record_id month day year plot_id species_id     sex hindfoot_length
## 1         1     7  16 1977       2         NL       M              32
## 2        72     8  19 1977       2         NL       M              31
## 3       224     9  13 1977       2         NL Missing              NA
## 4       266    10  16 1977       2         NL Missing              NA
## 5       349    11  12 1977       2         NL Missing              NA
## 6       363    11  12 1977       2         NL Missing              NA
##   weight   genus  species   taxa plot_type weight_kg
## 1     NA Neotoma albigula Rodent   Control        NA
## 2     NA Neotoma albigula Rodent   Control        NA
## 3     NA Neotoma albigula Rodent   Control        NA
## 4     NA Neotoma albigula Rodent   Control        NA
## 5     NA Neotoma albigula Rodent   Control        NA
## 6     NA Neotoma albigula Rodent   Control        NA
```

```
#The first few rows of the output are full of NAs, so if we wanted to remove those we could insert a fi

survey_data %>%
  filter(!is.na(weight)) %>%
  mutate(weight_kg = weight / 1000) %>%
  head
```

```
##   record_id month day year plot_id species_id sex hindfoot_length weight
## 1       588     2  18 1978       2         NL   M              NA    218
## 2       845     5   6 1978       2         NL   M              32    204
## 3       990     6   9 1978       2         NL   M              NA    200
## 4      1164     8   5 1978       2         NL   M              34    199
## 5      1261     9   4 1978       2         NL   M              32    197
## 6      1453    11   5 1978       2         NL   M              NA    218
##     genus  species   taxa plot_type weight_kg
## 1 Neotoma albigula Rodent   Control     0.218
## 2 Neotoma albigula Rodent   Control     0.204
## 3 Neotoma albigula Rodent   Control     0.200
## 4 Neotoma albigula Rodent   Control     0.199
## 5 Neotoma albigula Rodent   Control     0.197
## 6 Neotoma albigula Rodent   Control     0.218
```

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).