

# **Workspace**

# Table of contents

<b>Welcome</b>	<b>5</b>
<b>I Notebooks</b>	<b>6</b>
<b>1 R</b>	<b>8</b>
1.1 General Notes . . . . .	8
1.2 Data Types . . . . .	9
1.2.1 Mixing Data Types . . . . .	9
1.3 Data Structures . . . . .	9
1.4 Basics Operations . . . . .	10
1.5 Exploratory Operations . . . . .	11
1.5.1 Retrieve specific element/row/column . . . . .	14
1.5.2 Dealing with factor (categorical) columns . . . . .	14
1.5.3 Basic Plotting . . . . .	15
1.6 Data Manipulation . . . . .	17
1.6.1 Selection . . . . .	19
1.6.2 Piping . . . . .	21
1.6.3 Summary . . . . .	21
1.6.4 Count . . . . .	25
1.6.5 Reshaping . . . . .	26
1.6.6 Filtering . . . . .	28
1.6.7 Saving to disk . . . . .	28
1.7 Visualization . . . . .	28
1.7.1 Scatter plot . . . . .	29
1.7.2 Boxplot . . . . .	32
1.7.3 Time series data . . . . .	34
1.8 References . . . . .	39
<b>2 Quarto</b>	<b>40</b>
2.1 Render & Review . . . . .	40
2.2 Render w/o Review . . . . .	40
2.3 References . . . . .	40
<b>3 RStudio</b>	<b>41</b>
3.1 Keyboard Shortcuts . . . . .	41

<b>II R for Data Science (2e)</b>	<b>42</b>
<b>4 Data visualization</b>	<b>44</b>
4.1 Introduction . . . . .	44
4.2 Clear Workspace . . . . .	44
4.3 Packages . . . . .	44
4.3.1 List . . . . .	44
4.3.2 Install Missing . . . . .	44
4.3.3 Load . . . . .	45
4.4 Plot Creation Process . . . . .	46
4.4.1 Load Dataset . . . . .	46
4.4.2 Load Plotting Package . . . . .	48
4.4.3 Create <code>ggplot</code> object . . . . .	48
4.4.4 Link Dataset . . . . .	49
4.4.5 Map Two Variables . . . . .	50
4.4.6 Display Data . . . . .	51
4.4.7 Map Third Variables . . . . .	52
4.4.8 Display Three Trendlines . . . . .	53
4.4.9 Display One Trendline . . . . .	54
4.4.10 Map One Variable Twice . . . . .	55
4.4.11 Fix Labels . . . . .	56
4.4.12 Ensure Color-blind Safe . . . . .	58
4.4.13 Can Call Implicitly . . . . .	59
4.4.14 Use Pipe Operator . . . . .	60
4.5 Visualizing Distribution . . . . .	61
4.5.1 Categorical Variables . . . . .	61
4.5.2 Numerical Variables . . . . .	62
4.6 Visualizing Relationships . . . . .	63
4.6.1 One Categorical + One Numerical . . . . .	63
4.6.2 Two Categoricals . . . . .	63
4.6.3 Two Numerical . . . . .	64
4.6.4 Three or More Variables . . . . .	64
<b>III Practice</b>	<b>65</b>
<b>5 Topic Modeling in R</b>	<b>67</b>
5.1 Introduction . . . . .	67
5.2 Download Book . . . . .	67
5.3 Wrangle: Label Stories . . . . .	68
5.4 Wrangle: Put in Tidy Format . . . . .	69
5.5 Explore tf-idf . . . . .	69
5.6 Implement Topic Modeling . . . . .	71

5.7	Contribution of Words in Topics . . . . .	74
5.8	Distribution of Topics in Stories . . . . .	75
5.9	References . . . . .	77

# Welcome

This Quarto Book<sup>1</sup> is a workspace for the notes and projects of the programming/scripting languages that I am learning.

---

<sup>1</sup>[Quarto Books Documentation](#)

# **Part I**

# **Notebooks**

This section contains notes for the programming/scripting languages that I am learning.

# 1 R

## 1.1 General Notes

- When writing R code, create a project instead of a file which will enable saving the workspace settings
- An R package usually includes:
  - a. reusable functions
  - b. documentation describing how to use the function
  - c. sample data
- Before running a project, clear the objects in its workspace environment to avoid mixing up objects created in other files. This can be done either:
  - a. pragmatically as shown below OR
  - b. Environment window -> Broom icon

Clear the objects in its workspace environment to avoid mixing up objects created in other files

```
rm(list = ls())
```

List all the packages that will be used in this script.

```
packages = c("here")
```

Any missing package will be installed automatically

```
# Do NOT modify
install.packages(setdiff(packages, rownames(installed.packages())))
```

Load all packages

```
# Do NOT modify
lapply(packages, require, character.only = TRUE)
```

```
[[1]]
[1] TRUE
```

## 1.2 Data Types

- character
- numeric
- logical
- raw
- imaginary numbers

To know the datatype of an object, run the command:

```
class(x)      # give the data type of x
```

### 1.2.1 Mixing Data Types

- character + numeric → character
- numeric + logical → numeric
- numeric + character + logical → character

## 1.3 Data Structures

- **vector**: hold single type of data
- **matrix**: 2D vector
- **array**: nD vector
- **list**: generic vector, can hold mixed type of data, eg, one element can a character, another a list of integers, and the third could be a logical
- **data frame**: table where columns represent vectors
- **tibbles**: data frames, but slightly tweaked to work better with **tidyverse** package
- **factor**

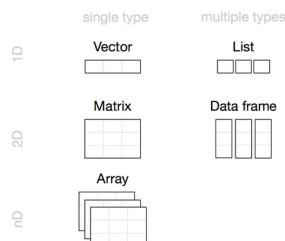


Figure 1.1: Common data structures in R (Source: [Grolemund, 2014](#))

To know the data structure and length of the object, run the command:

```
str(x)      # give the structure type of x  
length(x)   # length of structure
```

## 1.4 Basics Operations

Assignment

```
x <- 3      # assign 3 to x  
(x <- 3)    # assign 3 to x & print the result to console
```

```
[1] 3
```

Getting Help

```
args(round) # print the argument list of function  
?round       # show documentation of function in Help window
```

Dealing with Structure

```
# concatenate set of values to create vector  
weight_g <- c(50, 60, 3, 9)  
animals <- c("dog", "bat", "cat")  
  
# utilizing logical values to pull specific values  
weight_g[weight_g < 10 & weight_g > 60 | weight_g == 50]
```

```
[1] 50
```

```
# pull dog & cat records  
animals[animals %in% c("dog", "cat")]
```

```
[1] "dog" "cat"
```

```
animals[animals == "dog" | animals == "cat"]
```

```
[1] "dog" "cat"
```

Statistics

```
# signaling missing data using NA
heights <- c(2, 3, NA, 4)

# get mean while ignoring missing data
mean(heights, na.rm = TRUE)
```

```
[1] 3
```

```
# how to use mean
# ?mean
```

## 1.5 Exploratory Operations

The `here` package makes it easy to point to files starting from the project main directory.

```
library(here)
```

Loading file from repository and saving it locally on disk. It is always a good idea to structure the workspace—for more information, see [Best Practices for Scientific Computing](#) paper.

```
download.file(url = "https://ndownloader.figshare.com/files/2292169",
              destfile = here("data", "portal_data_joined.csv"))
```

Load file to R as data frame

```
surveys <- read.csv(here("data", "portal_data_joined.csv"))
```

Inspecting data frame

```
class(surveys) # data type
```

```
[1] "data.frame"
```

```
str(surveys) # structure
```

```
'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     : chr  "NL" "NL" "NL" "NL" ...  
 $ sex             : chr  "M" "M" "" "" ...  
 $ hindfoot_length: int  32 31 NA NA NA NA NA NA NA NA ...  
 $ weight          : int  NA NA NA NA NA NA NA 218 NA ...  
 $ genus           : chr  "Neotoma" "Neotoma" "Neotoma" "Neotoma" ...  
 $ species         : chr  "albigula" "albigula" "albigula" "albigula" ...  
 $ taxa            : chr  "Rodent" "Rodent" "Rodent" "Rodent" ...  
 $ plot_type       : chr  "Control" "Control" "Control" "Control" ...
```

```
dim(surveys)      # dimensions
```

```
[1] 34786    13
```

```
nrow(surveys)
```

```
[1] 34786
```

```
ncol(surveys)
```

```
[1] 13
```

```
summary(surveys)
```

	record_id	month	day	year	plot_id
Min.	: 1	Min. : 1.000	Min. : 1.0	Min. :1977	Min. : 1.00
1st Qu.	: 8964	1st Qu.: 4.000	1st Qu.: 9.0	1st Qu.:1984	1st Qu.: 5.00
Median	:17762	Median : 6.000	Median :16.0	Median :1990	Median :11.00
Mean	:17804	Mean : 6.474	Mean :16.1	Mean :1990	Mean :11.34
3rd Qu.	:26655	3rd Qu.:10.000	3rd Qu.:23.0	3rd Qu.:1997	3rd Qu.:17.00
Max.	:35548	Max. :12.000	Max. :31.0	Max. :2002	Max. :24.00

	species_id	sex	hindfoot_length	weight
Length:	34786	Length:34786	Min. : 2.00	Min. : 4.00

```

Class :character   Class :character   1st Qu.:21.00   1st Qu.: 20.00
Mode  :character   Mode  :character   Median :32.00   Median : 37.00
                           Mean   :29.29   Mean   : 42.67
                           3rd Qu.:36.00   3rd Qu.: 48.00
                           Max.   :70.00   Max.   :280.00
                           NA's    :3348   NA's    :2503

genus           species          taxa          plot_type
Length:34786    Length:34786    Length:34786    Length:34786
Class :character Class :character Class :character Class :character
Mode  :character Mode  :character Mode  :character Mode  :character

```

Show first/last few objects/records/rows

```
head(surveys)
```

	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					

```
tail(surveys)
```

	record_id	month	day	year	plot_id	species_id	sex	hindfoot_length	weight
34781	26787	9	27	1997	7	PL	F	21	16
34782	26966	10	25	1997	7	PL	M	20	16
34783	27185	11	22	1997	7	PL	F	21	22
34784	27792	5	2	1998	7	PL	F	20	8

```

34785      28806    11  21 1998       7       PX      NA      NA
34786      30986     7   1 2000       7       PX      NA      NA
            genus  species   taxa      plot_type
34781 Peromyscus leucopus Rodent Rodent Exclosure
34782 Peromyscus leucopus Rodent Rodent Exclosure
34783 Peromyscus leucopus Rodent Rodent Exclosure
34784 Peromyscus leucopus Rodent Rodent Exclosure
34785 Chaetodipus      sp. Rodent Rodent Exclosure
34786 Chaetodipus      sp. Rodent Rodent Exclosure

```

### 1.5.1 Retrieve specific element/row/column

```
surveys[1,1]      # element[1,1]
```

```
[1] 1
```

```
surveys[1, ]      # row 1
```

```

record_id month day year plot_id species_id sex hindfoot_length weight
1          1     7 1977        2       NL    M           32      NA
            genus  species   taxa      plot_type
1 Neotoma albicula Rodent Control

```

```
head(surveys[,1])  # column 1
```

```
[1] 1 72 224 266 349 363
```

```
head(surveys$sex)      # column by name
```

```
[1] "M" "M" ""   ""   ""   "
```

### 1.5.2 Dealing with factor (categorical) columns

R convert columns that contain characters to factors by default. Factors are treated as integer vectors. By default, R sorts levels in alphabetical order.

```
levels(surveys$sex)
```

NULL

```
nlevels(surveys$sex)
```

[1] 0

Reorder factors (to get better plots)

```
surveys$sex_ordered <- factor(surveys$sex, level=c("F", "M", ""))
str(surveys$sex_ordered)
```

Factor w/ 3 levels "F", "M", "": 2 2 3 3 3 3 3 3 2 3 ...

```
levels(surveys$sex_ordered)
```

[1] "F" "M" "

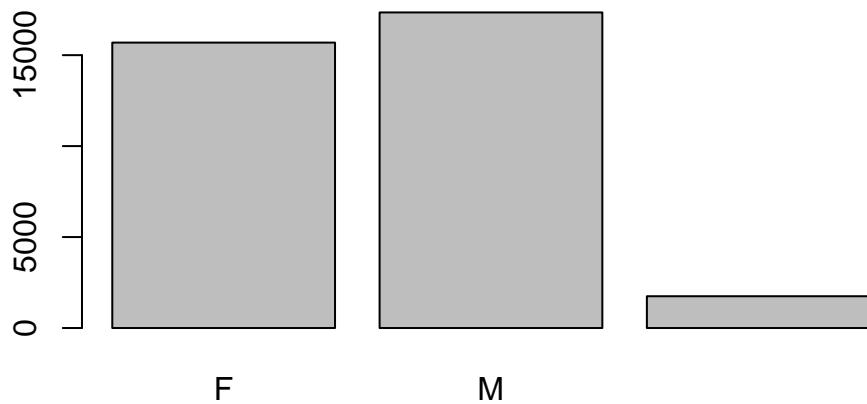
```
nlevels(surveys$sex_ordered)
```

[1] 3

### 1.5.3 Basic Plotting

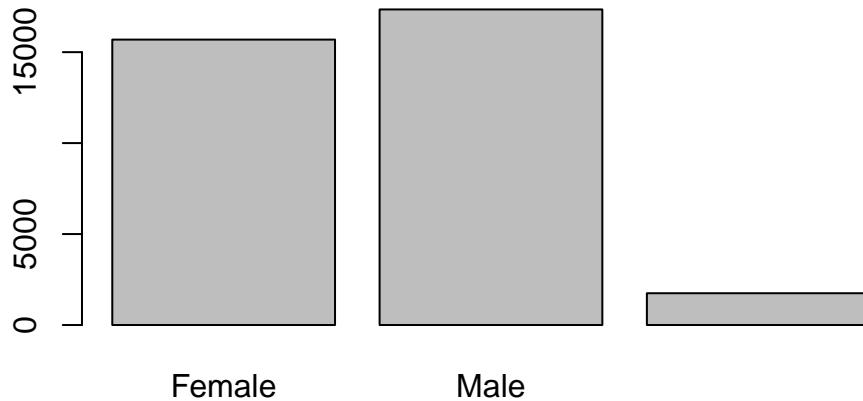
Histogram

```
# plot(surveys$sex) # not possible
plot(surveys$sex_ordered)
```



Enhance the plot

```
levels(surveys$sex_ordered)[1] <- "Female"  
levels(surveys$sex_ordered)[2] <- "Male"  
plot(surveys$sex_ordered)
```



## 1.6 Data Manipulation

- **tidyverse**
  - makes manipulation of data easier
  - built to work with data frames directly
  - can directly work with data stored in an external database which give the advantage of only bringing what we need to the memory to work on without having to bring the whole database
- **tidyverse**
  - allows to swiftly convert b/w different data formats for plotting & analysis in order to accommodate the different requirements by different functions
    - \* sometime we want one row per measurement
    - \* other times we want the data aggregated like when plotting

Before using **tidyverse** and **tidyverse**:

- Install **tidyverse** package: umbrella-package that install several packages (tidyverse, dplyr, ggplot2, tibble, magrittr, etc.)
- Load the package each session

Load packages

```
library("tidyverse")  
  
-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --  
v dplyr     1.1.4     v readr     2.1.5  
v forcats   1.0.0     v stringr   1.5.1  
v ggplot2   3.5.1     v tibble    3.2.1  
v lubridate 1.9.3     v tidyr    1.3.1  
v purrr    1.0.2  
-- Conflicts ----- tidyverse_conflicts() --  
x dplyr::filter() masks stats::filter()  
x dplyr::lag()   masks stats::lag()  
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become non-conflicting
```

Load & inspect data

```
# notice the '_' instead of '.' of basic R  
surveys <- read_csv(here("data", "portal_data_joined.csv"))
```

```
Rows: 34786 Columns: 13  
-- Column specification -----  
Delimiter: ","  
chr (6): species_id, sex, genus, species, taxa, plot_type  
dbl (7): record_id, month, day, year, plot_id, hindfoot_length, weight  
  
i Use `spec()` to retrieve the full column specification for this data.  
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
str(surveys) # structure:tbl_df (tibble)
```

```
spc_tbl_ [34,786 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)  
$ record_id      : num [1:34786] 1 72 224 266 349 363 435 506 588 661 ...  
$ month          : num [1:34786] 7 8 9 10 11 11 12 1 2 3 ...  
$ day            : num [1:34786] 16 19 13 16 12 12 10 8 18 11 ...  
$ year           : num [1:34786] 1977 1977 1977 1977 1977 1977 1977 1977 1977 1977 ...  
$ plot_id         : num [1:34786] 2 2 2 2 2 2 2 2 2 2 ...  
$ species_id     : chr [1:34786] "NL" "NL" "NL" "NL" ...  
$ sex             : chr [1:34786] "M" "M" NA NA ...  
$ hindfoot_length: num [1:34786] 32 31 NA NA NA NA NA NA NA NA ...
```

```

$ weight           : num [1:34786] NA NA NA NA NA NA NA NA 218 NA ...
$ genus            : chr [1:34786] "Neotoma" "Neotoma" "Neotoma" "Neotoma" ...
$ species          : chr [1:34786] "albigula" "albigula" "albigula" "albigula" ...
$ taxa              : chr [1:34786] "Rodent" "Rodent" "Rodent" "Rodent" ...
$ plot_type         : chr [1:34786] "Control" "Control" "Control" "Control" ...
- attr(*, "spec")=
.. cols(
..   record_id = col_double(),
..   month = col_double(),
..   day = col_double(),
..   year = col_double(),
..   plot_id = col_double(),
..   species_id = col_character(),
..   sex = col_character(),
..   hindfoot_length = col_double(),
..   weight = col_double(),
..   genus = col_character(),
..   species = col_character(),
..   taxa = col_character(),
..   plot_type = col_character()
.. )
- attr(*, "problems")=<externalptr>

```

```
# view(surveys) # preview in the viewer window, avoid when rendering
```

### 1.6.1 Selection

Select certain columns

```
select(surveys, plot_id, species_id, weight)
```

```
# A tibble: 34,786 x 3
  plot_id species_id weight
  <dbl> <chr>      <dbl>
1       2 NL          NA
2       2 NL          NA
3       2 NL          NA
4       2 NL          NA
5       2 NL          NA
6       2 NL          NA
7       2 NL          NA
```

```

8      2 NL      NA
9      2 NL      218
10     2 NL      NA
# i 34,776 more rows

```

Select all columns except ...

```
select(surveys, -sex)
```

```

# A tibble: 34,786 x 12
  record_id month   day   year plot_id species_id hindfoot_length weight genus
  <dbl>    <dbl> <dbl> <dbl>    <dbl>    <chr>            <dbl>    <dbl>    <chr>
1       1      7   16  1977      2 NL             32      NA Neotoma
2      72      8   19  1977      2 NL             31      NA Neotoma
3     224      9   13  1977      2 NL            NA      NA Neotoma
4     266     10   16  1977      2 NL            NA      NA Neotoma
5     349     11   12  1977      2 NL            NA      NA Neotoma
6     363     11   12  1977      2 NL            NA      NA Neotoma
7     435     12   10  1977      2 NL            NA      NA Neotoma
8     506      1     8  1978      2 NL            NA      NA Neotoma
9     588      2   18  1978      2 NL            NA      218 Neotoma
10    661      3   11  1978      2 NL            NA      NA Neotoma
# i 34,776 more rows
# i 3 more variables: species <chr>, taxa <chr>, plot_type <chr>

```

Select rows based on criteria

```
filter(surveys, year == 1995)
```

```

# A tibble: 1,180 x 13
  record_id month   day   year plot_id species_id sex   hindfoot_length weight
  <dbl>    <dbl> <dbl> <dbl>    <dbl>    <chr> <chr>            <dbl>    <dbl>
1     22314     6     7  1995      2 NL      M      34      NA
2     22728     9    23  1995      2 NL      F      32     165
3     22899    10    28  1995      2 NL      F      32     171
4     23032    12     2  1995      2 NL      F      33      NA
5     22003     1    11  1995      2 DM      M      37      41
6     22042     2     4  1995      2 DM      F      36      45
7     22044     2     4  1995      2 DM      M      37      46
8     22105     3     4  1995      2 DM      F      37      49
9     22109     3     4  1995      2 DM      M      37      46

```

```

10      22168      4      1  1995      2 DM      M      36      48
# i 1,170 more rows
# i 4 more variables: genus <chr>, species <chr>, taxa <chr>, plot_type <chr>

```

## 1.6.2 Piping

Sending the results of one function to another

```

# in multiple steps
survey_less5 <- filter(surveys, weight < 5)
survey_sml <- select(survey_less5, species_id, sex, weight)

# in one long step
survey_sml <- select(filter(surveys, weight < 5), species_id, sex, weight)

# using pipe %>% of magrittr package.  Use Ctrl + Shift + M to add
survey_sml <- surveys %>%
  filter(weight < 5) %>%
  select(species_id, sex, weight)

```

## 1.6.3 Summary

Summary of groups (1+ columns)

one factor

```

surveys %>%
  group_by(sex) %>%
  summarise(mean_weight = mean(weight, na.rm = TRUE))

```

```

# A tibble: 3 x 2
  sex    mean_weight
  <chr>     <dbl>
1 F          42.2
2 M          43.0
3 <NA>       64.7

```

two factors

```

surveys %>%
  group_by(sex, species) %>%
  summarise(mean_weight = mean(weight, na.rm = TRUE))

```

`summarise()` has grouped output by 'sex'. You can override using the ` `.groups` argument.

```

# A tibble: 81 x 3
# Groups:   sex [3]
  sex   species     mean_weight
  <chr> <chr>          <dbl>
1 F     albigula     154.
2 F     baileyi      30.2
3 F     eremicus     22.8
4 F     flavus       7.97
5 F     fulvescens    13.7
6 F     fulviventer  69
7 F     hispidus     69.0
8 F     leucogaster   31.1
9 F     leucopus      19.3
10 F    maniculatus   22.1
# i 71 more rows

```

```

surveys %>%
  group_by(species, sex) %>%
  summarise(mean_weight = mean(weight, na.rm = TRUE))

```

`summarise()` has grouped output by 'species'. You can override using the ` `.groups` argument.

```

# A tibble: 81 x 3
# Groups:   species [40]
  species     sex   mean_weight
  <chr> <chr>          <dbl>
1 albigula    F     154.
2 albigula    M     166.
3 albigula    <NA>   168.
4 audubonii   <NA>   NaN
5 baileyi     F     30.2
6 baileyi     M     33.8

```

```

7 baileyi      <NA>      30.6
8 bilineata    <NA>      NaN
9 brunneicapillus <NA>      NaN
10 chlorurus   <NA>      NaN
# i 71 more rows

```

to avoid using `na.rm = FALSE` each statistics

```

surveys %>%
  filter(!is.na(weight)) %>%
  group_by(species, sex) %>%
  summarise(mean_weight = mean(weight), sd_weight = sd(weight), sd_count = n())

```

``summarise()` has grouped output by 'species'. You can override using the  
.groups` argument.`

```

# A tibble: 59 x 5
# Groups:   species [22]
  species   sex   mean_weight   sd_weight   sd_count
  <chr>     <chr>     <dbl>       <dbl>       <int>
1 albigula   F      154.        39.2        652
2 albigula   M      166.        49.0        484
3 albigula   <NA>    168.        44.2        16
4 baileyi    F      30.2        5.27       1617
5 baileyi    M      33.8        8.27       1188
6 baileyi    <NA>   30.6        9.96        5
7 eremicus   F      22.8        4.57       568
8 eremicus   M      20.6        3.49       689
9 eremicus   <NA>   17.7        0.577       3
10 flavus    F      7.97        1.69       742
# i 49 more rows

```

arrange by mean weight

```

surveys %>%
  filter(!is.na(weight)) %>%
  group_by(species, sex) %>%
  summarise(mean_weight = mean(weight), sd_weight = sd(weight), sd_count = n()) %>%
  arrange(mean_weight)

```

```
`summarise()` has grouped output by 'species'. You can override using the
`.groups` argument.
```

```
# A tibble: 59 x 5
# Groups:   species [22]
  species   sex   mean_weight   sd_weight   sd_count
  <chr>     <chr>      <dbl>       <dbl>      <int>
1 flavus    <NA>        6          1.63         4
2 taylori   M           7.36        0.842        14
3 flavus    M           7.89        1.59        802
4 flavus    F           7.97        1.69        742
5 taylori   F           9.16        2.24         31
6 montanus  M           9.5         1.29         4
7 megalotis M           10.1        1.73       1339
8 montanus  F           11          2.16         4
9 megalotis <NA>        11.1        2.57         12
10 megalotis F          11.1        2.56       1184
# i 49 more rows
```

in descending order

```
surveys %>%
  filter(!is.na(weight)) %>%
  group_by(species, sex) %>%
  summarise(mean_weight = mean(weight), sd_weight = sd(weight), sd_count = n()) %>%
  arrange(desc(mean_weight))
```

```
`summarise()` has grouped output by 'species'. You can override using the
`.groups` argument.
```

```
# A tibble: 59 x 5
# Groups:   species [22]
  species   sex   mean_weight   sd_weight   sd_count
  <chr>     <chr>      <dbl>       <dbl>      <int>
1 albigula  <NA>        168.        44.2        16
2 albigula  M           166.        49.0       484
3 albigula  F           154.        39.2       652
4 hispidus  <NA>        130          NA          1
5 spilosoma M           130          NA          1
6 spectabilis M          122.        24.0      1220
7 spectabilis <NA>        120          18.5        18
```

```

8 spectabilis F      118.      21.5     1106
9 fulviventer F     69        37.8      16
10 hispidus   F     69.0      29.7     98
# i 49 more rows

```

by count

```

surveys %>%
  filter(!is.na(weight)) %>%
  group_by(species, sex) %>%
  summarise(mean_weight = mean(weight), sd_weight = sd(weight), sd_count = n()) %>%
  arrange(sd_count)

```

``summarise()` has grouped output by 'species'. You can override using the  
.groups` argument.`

```

# A tibble: 59 x 5
# Groups:   species [22]
  species    sex  mean_weight  sd_weight  sd_count
  <chr>     <chr>    <dbl>       <dbl>      <int>
1 hispidus   <NA>     130        NA         1
2 intermedius <NA>     18         NA         1
3 leucopus    <NA>     25         NA         1
4 spilosoma   F        57         NA         1
5 spilosoma   M        130        NA         1
6 fulviventer <NA>    40.5       6.36       2
7 leucogaster <NA>    29         11.3       2
8 eremicus    <NA>    17.7       0.577      3
9 ordii       <NA>    50.7       6.51       3
10 sp.        F        20.7       1.15       3
# i 49 more rows

```

#### 1.6.4 Count

Count of a categorical column

```

surveys %>%
  count(sex)

```

```
# A tibble: 3 x 2
  sex      n
  <chr> <int>
1 F      15690
2 M      17348
3 <NA>   1748
```

### 1.6.5 Reshaping

Using gather & spread

prepare the needed data first

```
surveys_gw <- surveys %>%
  filter(!is.na(weight)) %>%
  group_by(genus, plot_id) %>%
  summarise(mean_weight = mean(weight))
```

`summarise()` has grouped output by 'genus'. You can override using the `groups` argument.

creating a 2D table where each dimension represent a category the cell will represent a statistics

```
surveys_spread <- surveys_gw %>%
  spread(key = genus, value = mean_weight)
str(surveys_spread)
```

```
tibble [24 x 11] (S3: tbl_df/tbl/data.frame)
$ plot_id      : num [1:24] 1 2 3 4 5 6 7 8 9 10 ...
$ Baiomys      : num [1:24] 7 6 8.61 NA 7.75 ...
$ Chaetodipus  : num [1:24] 22.2 25.1 24.6 23 18 ...
$ Dipodomys    : num [1:24] 60.2 55.7 52 57.5 51.1 ...
$ Neotoma       : num [1:24] 156 169 158 164 190 ...
$ Onychomys    : num [1:24] 27.7 26.9 26 28.1 27 ...
$ Perognathus  : num [1:24] 9.62 6.95 7.51 7.82 8.66 ...
$ Peromyscus   : num [1:24] 22.2 22.3 21.4 22.6 21.2 ...
$ Reithrodontomys: num [1:24] 11.4 10.7 10.5 10.3 11.2 ...
$ Sigmodon     : num [1:24] NA 70.9 65.6 82 82.7 ...
$ Spermophilus : num [1:24] NA NA NA NA NA NA NA NA NA ...
```

```

head(surveys_spread)

# A tibble: 6 x 11
  plot_id Baiomys Chaetodipus Dipodomys Neotoma Onychomys Perognathus Peromyscus
  <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
1      1      7     22.2    60.2    156.    27.7     9.62    22.2
2      2      6     25.1    55.7    169.    26.9     6.95    22.3
3      3     8.61   24.6    52.0    158.    26.0     7.51    21.4
4      4     NA     23.0    57.5    164.    28.1     7.82    22.6
5      5     7.75   18.0    51.1    190.    27.0     8.66    21.2
6      6     NA     24.9    58.6    180.    25.9     7.81    21.8
# i 3 more variables: Reithrodontomys <dbl>, Sigmodon <dbl>, Spermophilus <dbl>

```

bring spread back

```

surveys_gw <- surveys_spread %>%
  gather(key = genus, value = mean_weight, -plot_id)
str(surveys_gw)

```

```

tibble [240 x 3] (S3: tbl_df/tbl/data.frame)
$ plot_id    : num [1:240] 1 2 3 4 5 6 7 8 9 10 ...
$ genus       : chr [1:240] "Baiomys" "Baiomys" "Baiomys" "Baiomys" ...
$ mean_weight: num [1:240] 7 6 8.61 NA 7.75 ...

```

```
head(surveys_gw)
```

```

# A tibble: 6 x 3
  plot_id genus  mean_weight
  <dbl> <chr>    <dbl>
1      1 Baiomys    7
2      2 Baiomys    6
3      3 Baiomys   8.61
4      4 Baiomys   NA
5      5 Baiomys   7.75
6      6 Baiomys   NA

```

## 1.6.6 Filtering

Remove missing data

```
survey_complete <- surveys %>%
  filter(!is.na(weight), !is.na(hindfoot_length), !is.na(sex))
```

Filter those that has sample greater than 50

```
species_counts <- survey_complete %>%
  count(species_id) %>%
  filter(n >= 50)
```

filter only those in the indicated category

```
surveys_com <- surveys %>%
  filter(species_id %in% c("albigula", "eremicus"))
```

## 1.6.7 Saving to disk

```
write_csv()
```

## 1.7 Visualization

- Help in making complex plots from data frames in simple steps
- ggplot graphics are built step by step by adding new elements; this makes it flexible as well as customization

Step 1: Bind the plot to specific data frame

```
surveys_plot <- ggplot(data = survey_complete,
  mapping = aes(x = weight, y = hindfoot_length))

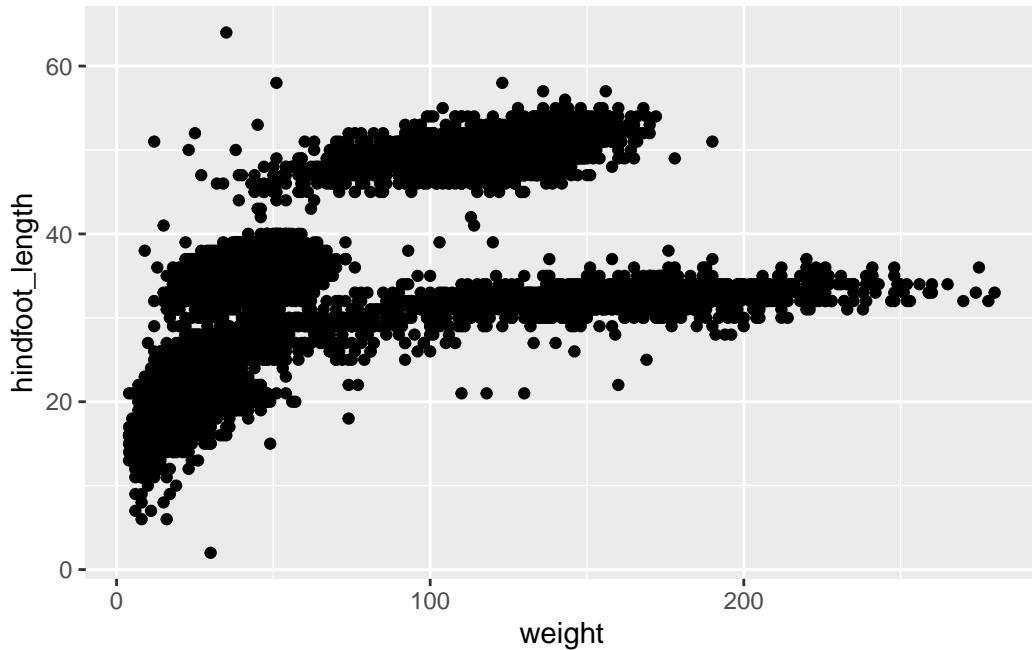
# Color for each group
surveys_plot <- ggplot(data = survey_complete,
  mapping = aes(x = weight, y = hindfoot_length),
  color=species_id)
```

Step 2: Select the type of the plot

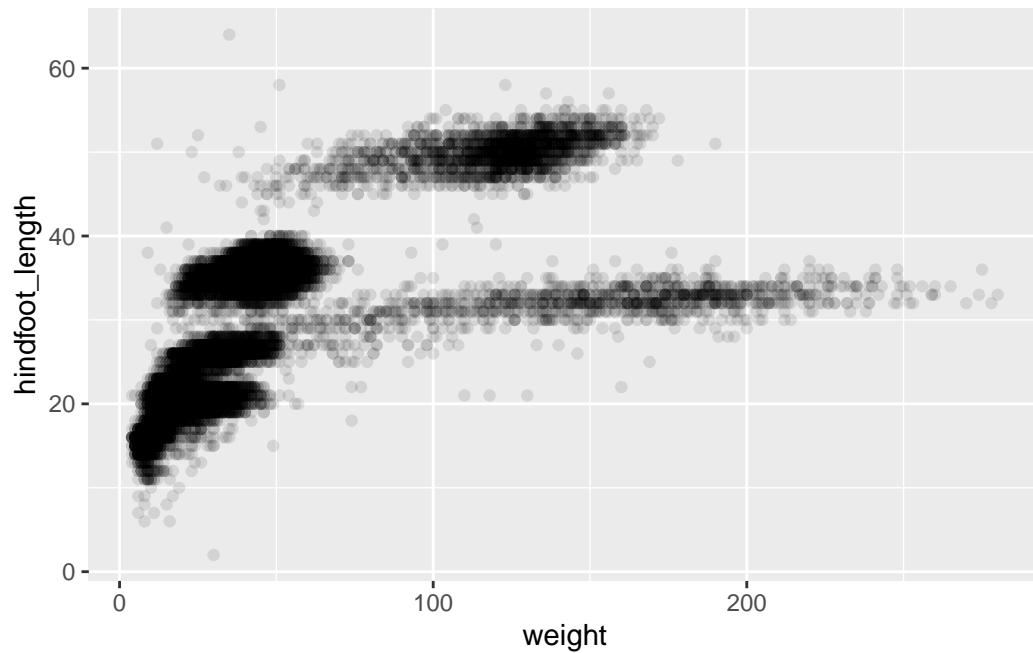
- scatter plot, dot plots, etc. > geom\_point()
- boxplots > geom\_boxplot()
- trend lines, time series, etc. > geom\_line()

### 1.7.1 Scatter plot

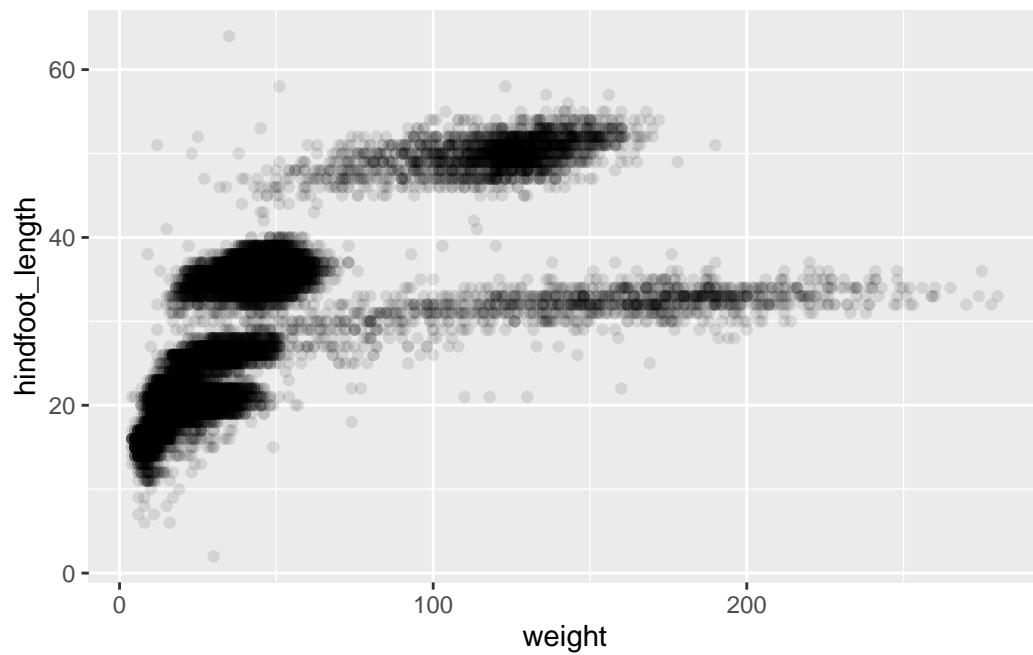
```
surveys_plot + geom_point()
```



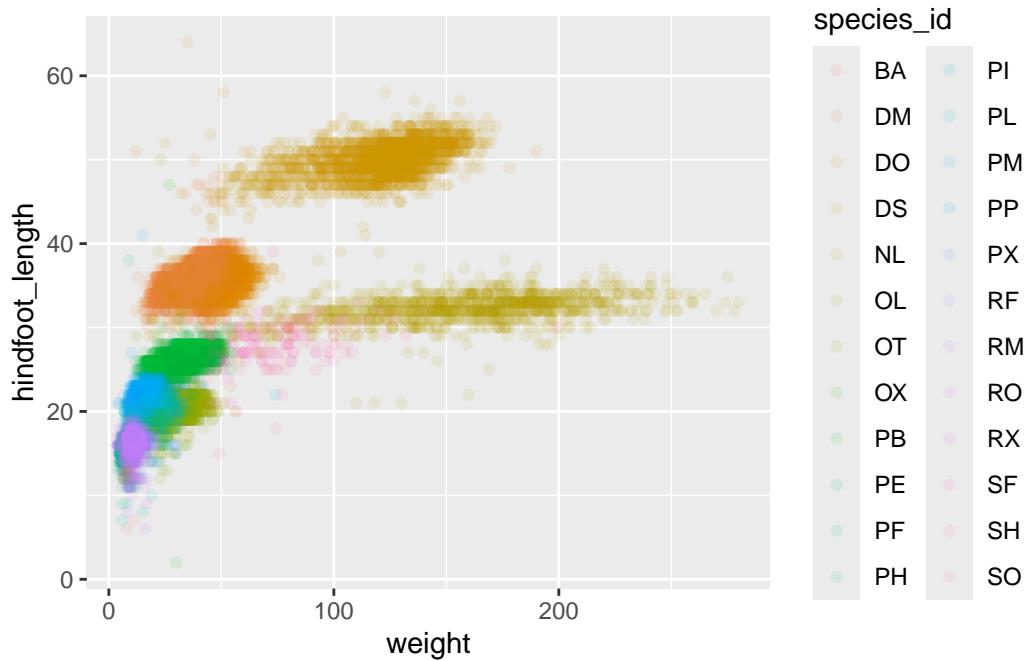
```
# add transparency
surveys_plot + geom_point(alpha = 0.1)
```



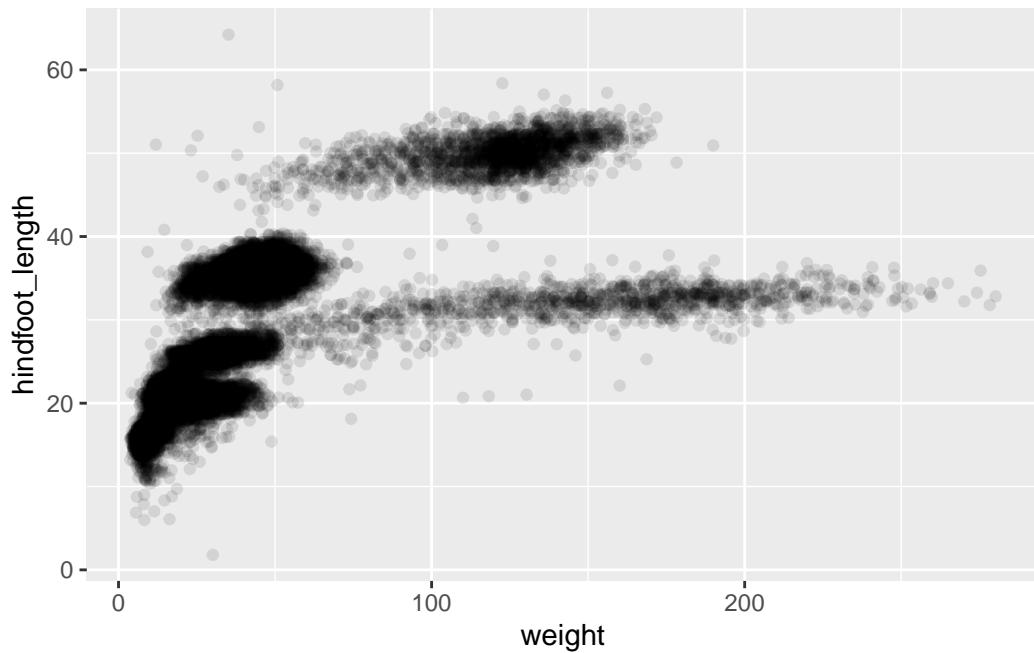
```
# color if not used in binding  
surveys_plot + geom_point(alpha = 0.1, color = "black")
```



```
# add color if not used in binding  
surveys_plot + geom_point(alpha = 0.1, aes(color = species_id))
```



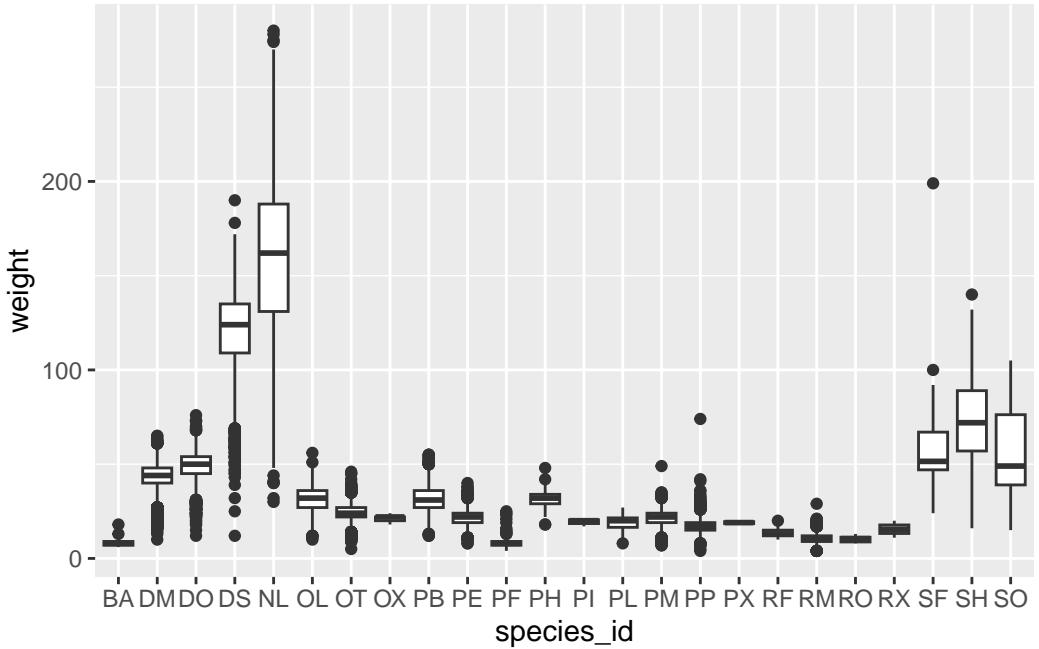
```
# make the color blend by introducing small random variation in points locations  
# used when having small data sets  
surveys_plot + geom_jitter(alpha = 0.1)
```



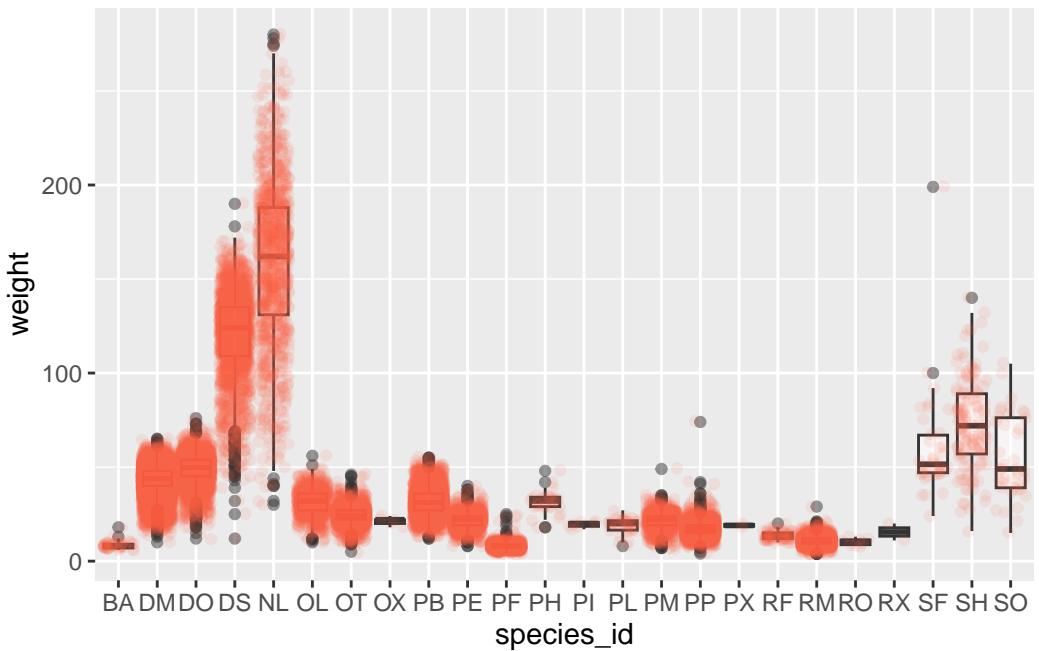
### 1.7.2 Boxplot

```
surveys_plot <- ggplot(data = survey_complete,
  mapping = aes(x = species_id, y = weight))

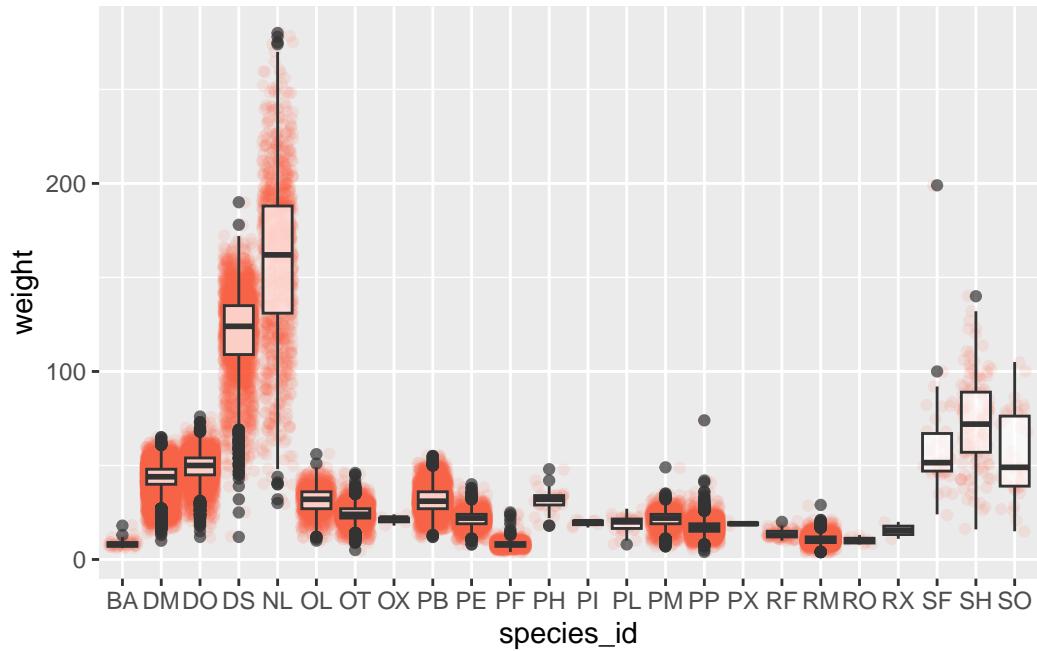
surveys_plot + geom_boxplot()
```



```
# show data
surveys_plot + geom_boxplot(alpha = 0.5) +
  geom_jitter(alpha = 0.1, color = "tomato")
```



```
# bring boxplot layer in front
surveys_plot + geom_jitter(alpha = 0.1, color = "tomato") +
  geom_boxplot(alpha = 0.7)
```

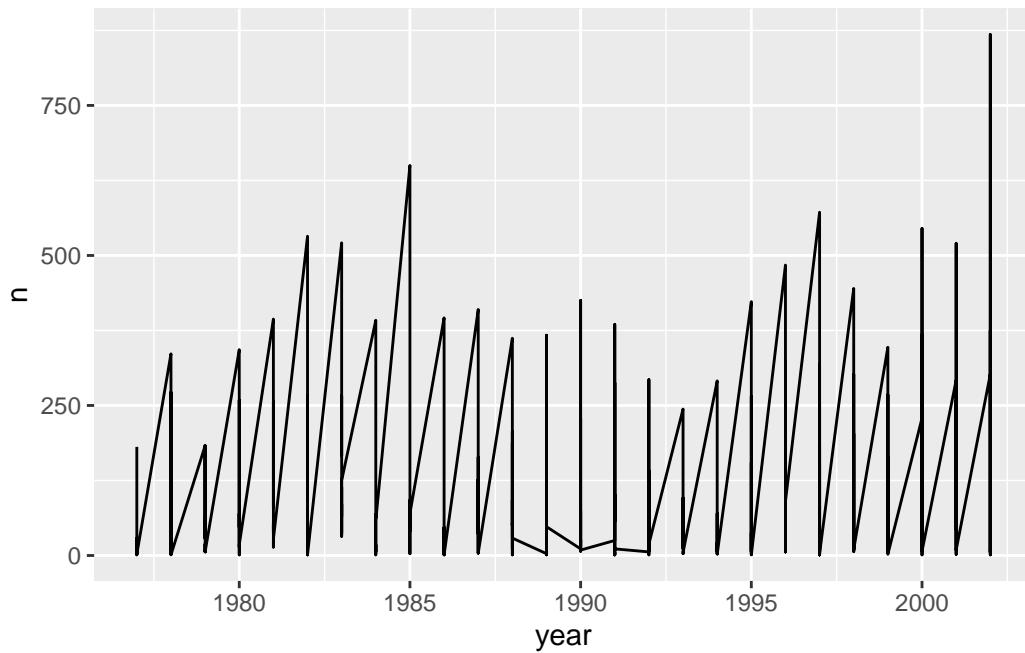


### 1.7.3 Time series data

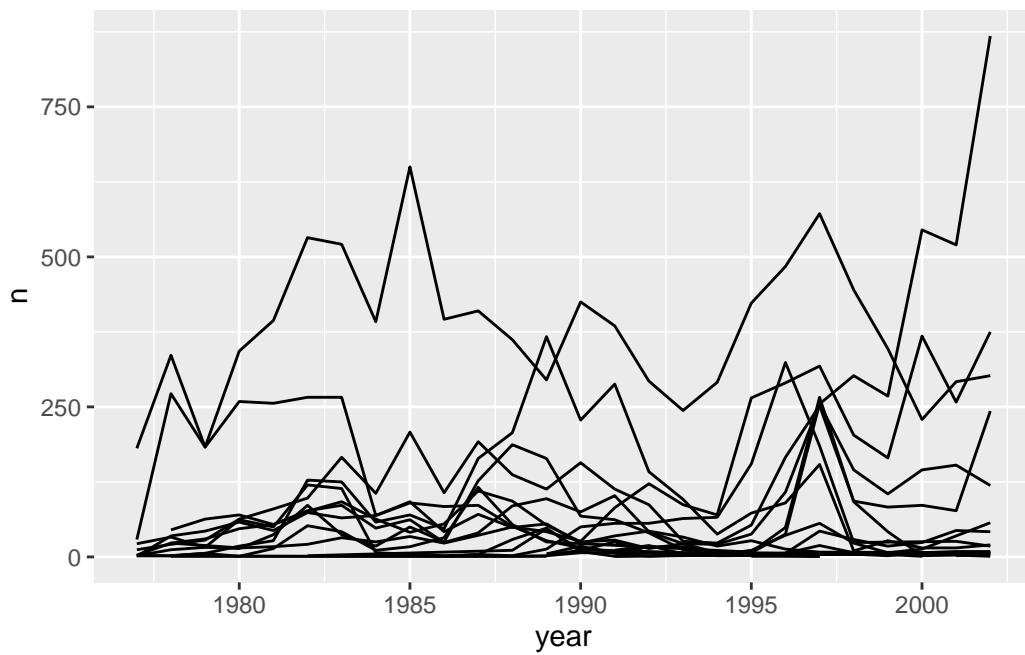
```
# create appropriate dataset
yearly_count <- survey_complete %>%
  count(year, species_id)

surveys_plot <- ggplot(data = yearly_count,
  mapping = aes(x = year, y = n))

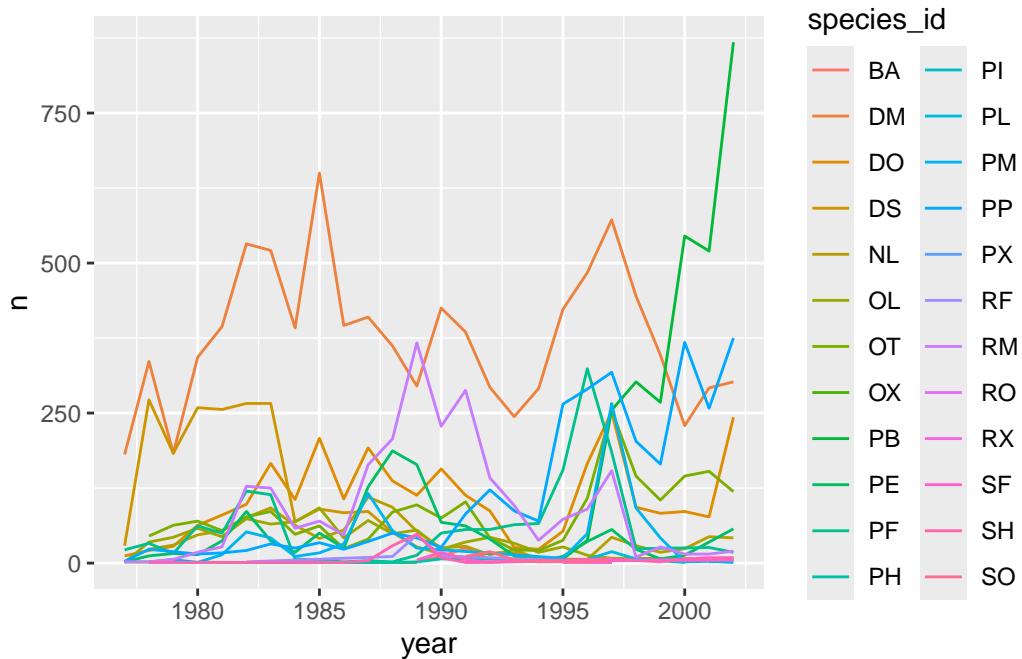
surveys_plot + geom_line()
```



```
# make it more meaningful by breaking it by category  
surveys_plot + geom_line(aes(group = species_id))
```

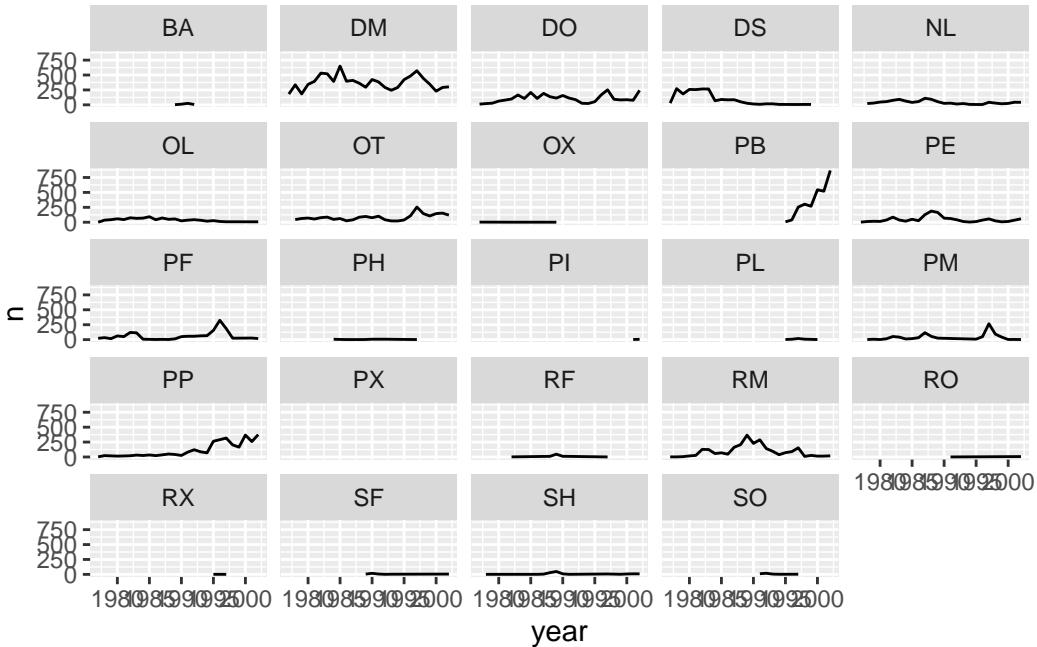


```
# make it more colorful  
surveys_plot + geom_line(aes(color = species_id))
```



```
# split into multiple plots  
surveys_plot + geom_line() + facet_wrap(~ species_id)
```

`geom\_line()`: Each group consists of only one observation.  
i Do you need to adjust the group aesthetic?

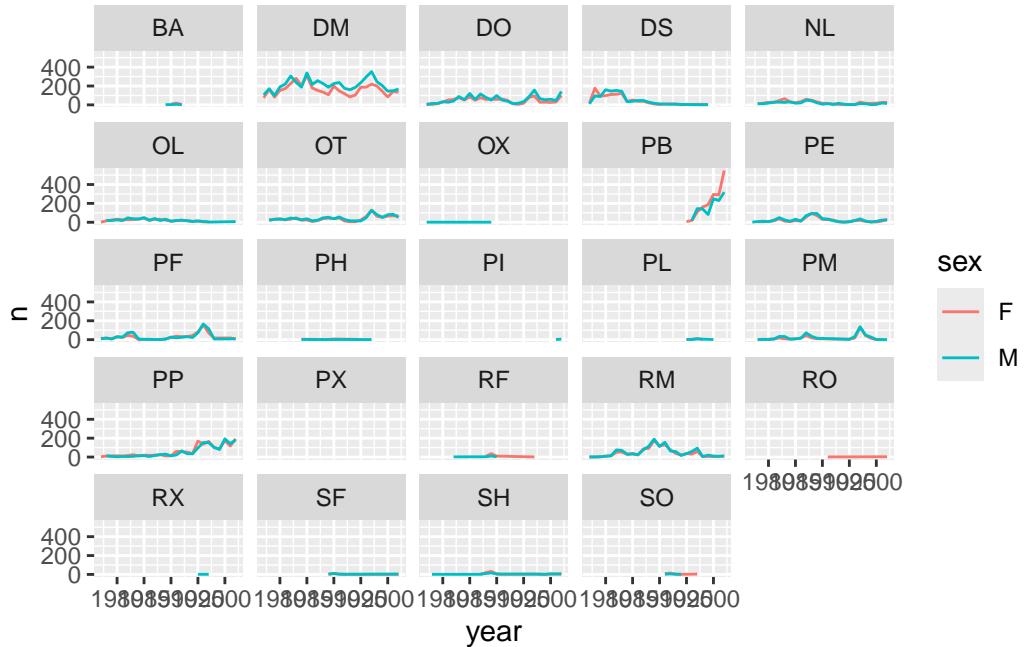


```
# split the line in each plot by sex
yearly_sex_counts <- survey_complete %>%
  count(year, species_id, sex)

surveys_plot <- ggplot(data = yearly_sex_counts,
  mapping = aes(x = year, y = n))

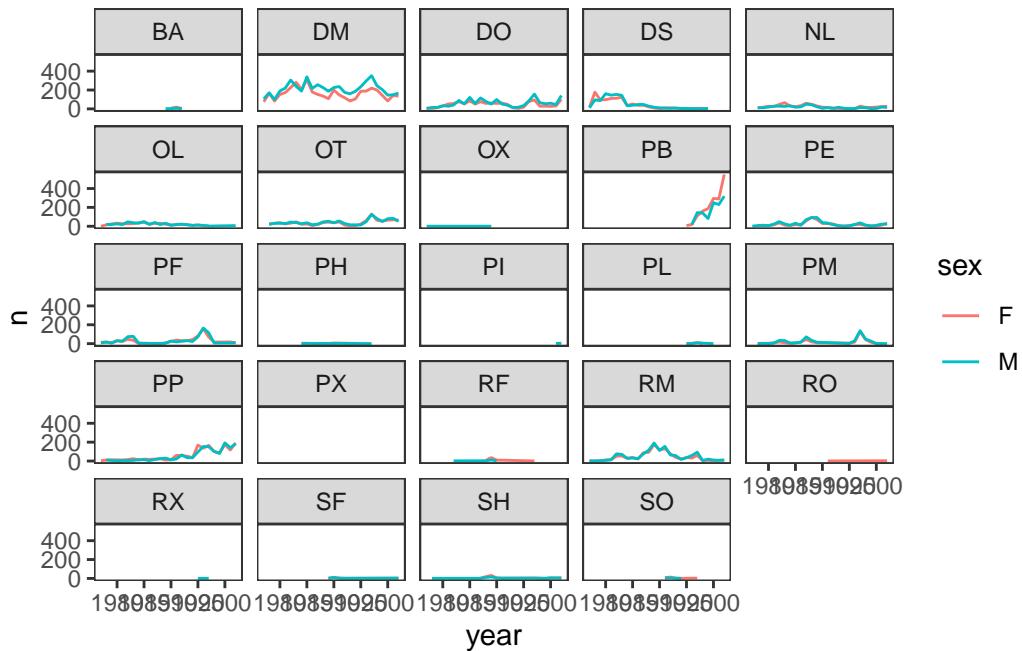
surveys_plot + geom_line(aes(color = sex)) +
  facet_wrap(~ species_id)
```

`geom\_line()`: Each group consists of only one observation.  
i Do you need to adjust the group aesthetic?



```
# remove background
surveys_plot + geom_line(aes(color = sex)) +
  facet_wrap(~ species_id) +
  theme_bw() +
  theme(panel.grid = element_blank())
```

`geom\_line()`: Each group consists of only one observation.  
i Do you need to adjust the group aesthetic?



## 1.8 References

- OU Software Carpentry Workshop (check other workshops [here](#))
  - [Main Tutorial](#)
  - [Data Carpentry with R](#)
  - [Software Carpentry with R](#)
  - [Etherpad](#)
  - [Google Doc](#)
- [Intro to ggplot by Allison Horst](#)
- [R for Data Science book by Garrett Grolemund and Hadley Wickham](#)
- [Best Practices for Scientific Computing paper](#)

## 2 Quarto

### 2.1 Render & Review

1. VSCode/RStudio -> *Render* button
2. Terminal -> `quarto preview`
3. Terminal -> `quarto preview help`

### 2.2 Render w/o Review

1. Terminal -> `quarto render`
2. Terminal -> `quarto render help`

### 2.3 References

- [Quarto Reference](#)

# 3 RStudio

## 3.1 Keyboard Shortcuts

Below is a set of helpful keyboard shortcut. The full list can be reached by clicking Tools -> Keyboard Shortcut Help (**Alt+Shift+K**) -> See All Shortcuts

- **Ctrl+Enter**: run line on which cursor is standing
- **Ctrl+Alt+I**: insert new code chunk
- **Ctrl+Shift+C**: comment/un-comment
- **Alt-↑**: move line up
- **Alt-↓**: move line down
- **Ctrl+D**: delete line
- **Ctrl+Shift+A**: format code
- **Ctrl+M**: add pipe |> operator

## **Part II**

# **R for Data Science (2e)**

This section contains replications for the examples used in the [R for Data Science \(2e\)](#) book, one Quarto file per chapter of the book.

# 4 Data visualization

## 4.1 Introduction

This page introduces the plot creation vocabulary step by step

## 4.2 Clear Workspace

Always start by clearing the workspace. This ensure objects created in other files are not used here.

```
rm(list = ls())
```

## 4.3 Packages

### 4.3.1 List

List all the packages that will be used in this script.

```
packages = c("palmerpenguins", "ggthemes", "ggplot2", "dplyr", "printr", "here", "knitr")
```

### 4.3.2 Install Missing

Any missing package will be installed automatically. This ensure smoother execution when run by others.

```
# Do NOT modify
install.packages(setdiff(packages, rownames(installed.packages()))))
```

### 4.3.3 Load

Load all packages

```
Loading required package: 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
```

Loading required package: printr

```
Registered S3 method overwritten by 'printr':
```

```
method           from  
knit_print.data.frame rmarkdown
```

Loading required package: here

```
here() starts at C:/Users/Amin Alhashim/Documents/GitHub/hashcx/notebooks
```

Loading required package: knitr

```
[[1]]  
[1] TRUE
```

```
[[2]]  
[1] TRUE
```

```
[[3]]  
[1] TRUE
```

```
[[4]]
```

```
[1] TRUE
```

```
[[5]]
```

```
[1] TRUE
```

```
[[6]]
```

```
[1] TRUE
```

```
[[7]]
```

```
[1] TRUE
```

## 4.4 Plot Creation Process

### 4.4.1 Load Dataset

The `penguins` dataset from the `palmerpenguins` package will be used for plotting. Typically, the package is loaded using the `library` function as shown in the code chunk below. However, a better approach is the one outlined in Section 4.3.

```
library(palmerpenguins)
```

Explore the dataset

```
help(penguins)
```

Size measurements for adult foraging penguins near Palmer Station,  
Antarctica

Description:

Includes measurements for penguin species, island in Palmer Archipelago, size (flipper length, body mass, bill dimensions), and sex. This is a subset of '`penguins_raw`'.

Usage:

```
penguins
```

Format:

A tibble with 344 rows and 8 variables:

species a factor denoting penguin species (Adélie, Chinstrap and Gentoo)

island a factor denoting island in Palmer Archipelago, Antarctica (Biscoe, Dream or Torgersen)

bill\_length\_mm a number denoting bill length (millimeters)

bill\_depth\_mm a number denoting bill depth (millimeters)

flipper\_length\_mm an integer denoting flipper length (millimeters)

body\_mass\_g an integer denoting body mass (grams)

sex a factor denoting penguin sex (female, male)

year an integer denoting the study year (2007, 2008, or 2009)

Source:

Adélie penguins: Palmer Station Antarctica LTER and K. Gorman. 2020. Structural size measurements and isotopic signatures of foraging among adult male and female Adélie penguins (*Pygoscelis adeliae*) nesting along the Palmer Archipelago near Palmer Station, 2007–2009 ver 5. Environmental Data Initiative. doi: 10.6073/pasta/98b16d7d563f265cb52372c8ca99e60f (URL: <https://doi.org/10.6073/pasta/98b16d7d563f265cb52372c8ca99e60f>)

Gentoo penguins: Palmer Station Antarctica LTER and K. Gorman. 2020. Structural size measurements and isotopic signatures of foraging among adult male and female Gentoo penguin (*Pygoscelis papua*) nesting along the Palmer Archipelago near Palmer Station, 2007–2009 ver 5. Environmental Data Initiative. doi: 10.6073/pasta/7fca67fb28d56ee2ffa3d9370ebda689 (URL: <https://doi.org/10.6073/pasta/7fca67fb28d56ee2ffa3d9370ebda689>)

Chinstrap penguins: Palmer Station Antarctica LTER and K. Gorman. 2020. Structural size measurements and isotopic signatures of foraging among adult male and female Chinstrap penguin (*Pygoscelis antarcticus*) nesting along the Palmer Archipelago near Palmer Station, 2007–2009 ver 6. Environmental Data Initiative. doi:

```
10.6073/pasta/c14dfcfada8ea13a17536e73eb6fbe9e (URL:  
https://doi.org/10.6073/pasta/c14dfcfada8ea13a17536e73eb6fbe9e)
```

Originally published in: Gorman KB, Williams TD, Fraser WR (2014) Ecological Sexual Dimorphism and Environmental Variability within a Community of Antarctic Penguins (Genus *Pygoscelis*). PLoS ONE 9(3): e90081. doi:10.1371/journal.pone.0090081

Explore the dataset differently

```
dplyr::glimpse(penguins)
```

```
Rows: 344  
Columns: 8  
$ species      <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie,  
$ island        <fct> Torgersen, Torgersen, Torgersen, Torgersen,  
$ bill_length_mm <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~  
$ bill_depth_mm  <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~  
$ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~  
$ body_mass_g    <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~  
$ sex            <fct> male, female, female, NA, female, male, female, male-  
$ year           <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

#### 4.4.2 Load Plotting Package

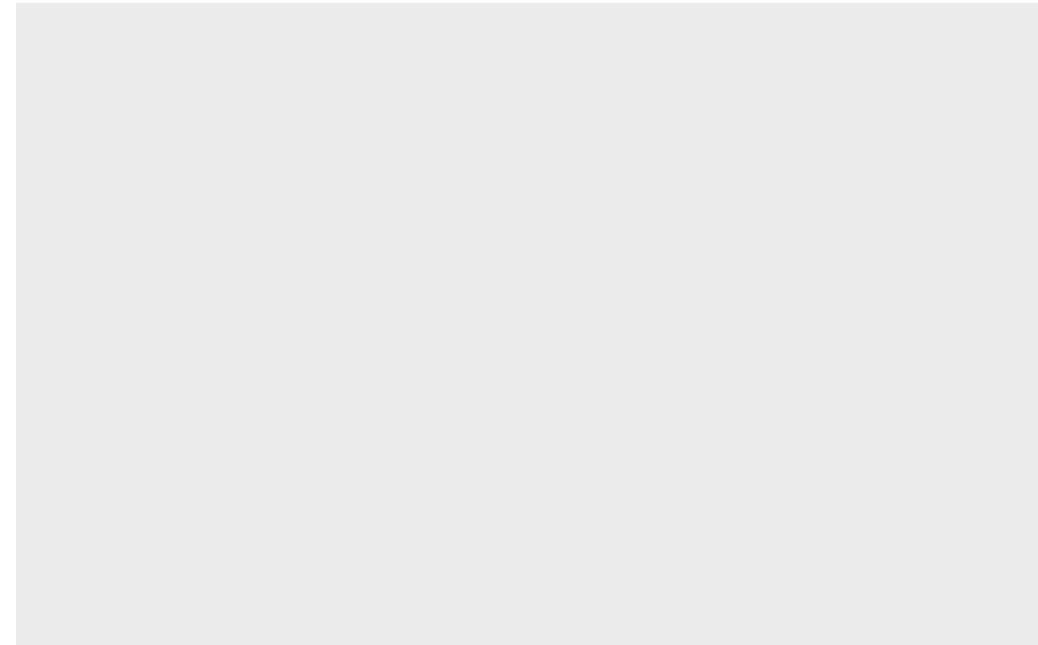
The `ggplot2` package will be used for plotting. The package is typically loaded using the `library` function as shown in the code chunk below. However, a better approach is the one outlined in Section 4.3.

```
library(ggplot2)
```

#### 4.4.3 Create ggplot object

Create an empty canvas by instantiating a `ggplot` object using the `ggplot()` function.

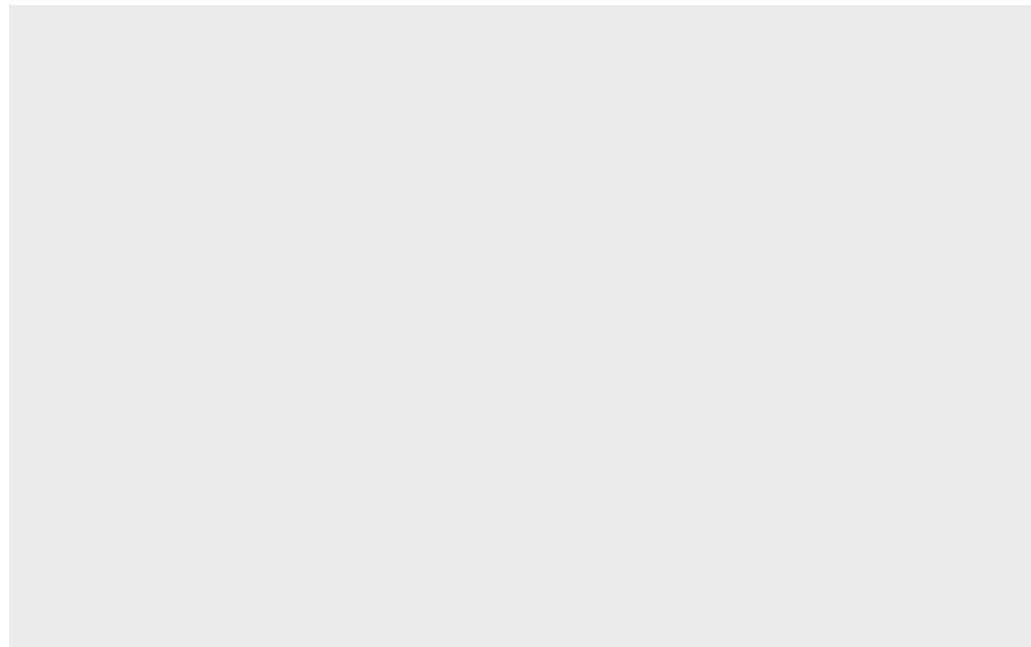
```
ggplot()
```



#### 4.4.4 Link Dataset

Link the dataset with the instantiated `ggplot` object using the `data` parameter.

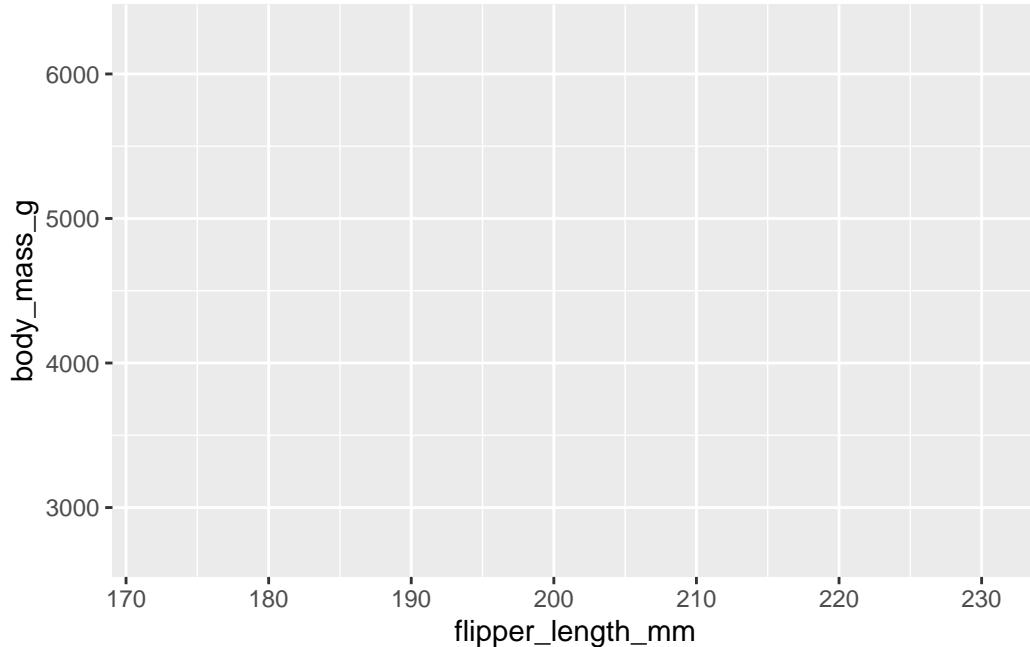
```
ggplot(data = penguins)
```



#### 4.4.5 Map Two Variables

Specify which of the variables in the dataset will be used as the plot aesthetics (visual properties) using the `mapping` argument done via the `aes()` function.

```
ggplot(data = penguins,  
       mapping = aes(x = flipper_length_mm, y = body_mass_g))
```

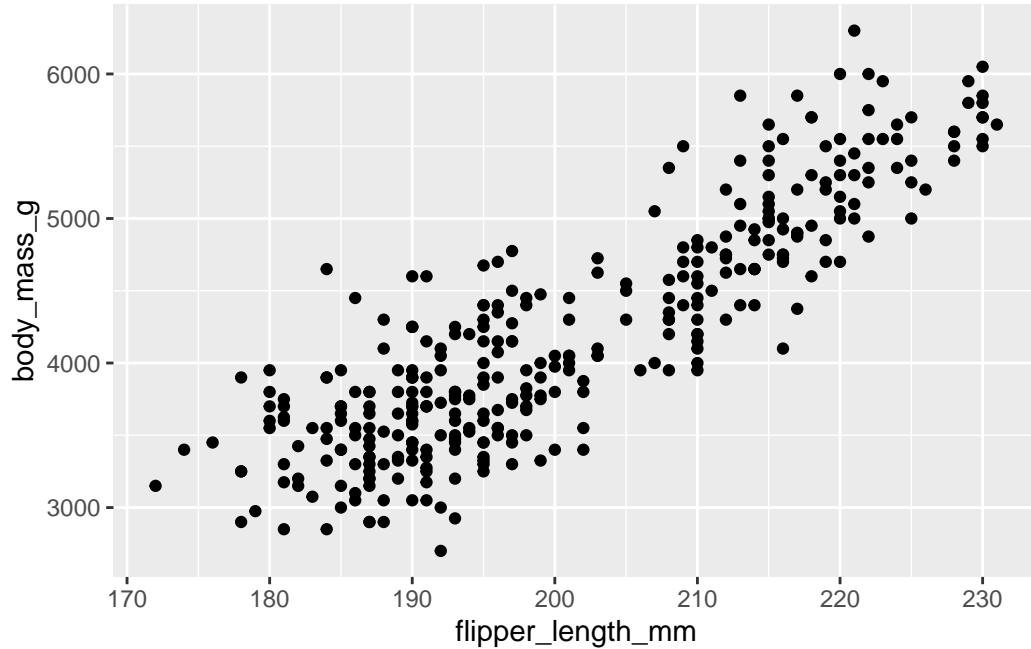


#### 4.4.6 Display Data

Specify how the data (observations) will be represented geometrically on the plot, eg, bars, points, or line. The functions starting with `geom_` is used for this purpose. These functions add layer of the selected geometric object to the plot.

```
ggplot(data = penguins,  
       mapping = aes(x = flipper_length_mm, y = body_mass_g)) +  
       geom_point()
```

Warning: Removed 2 rows containing missing values or values outside the scale range  
(`geom\_point()`).

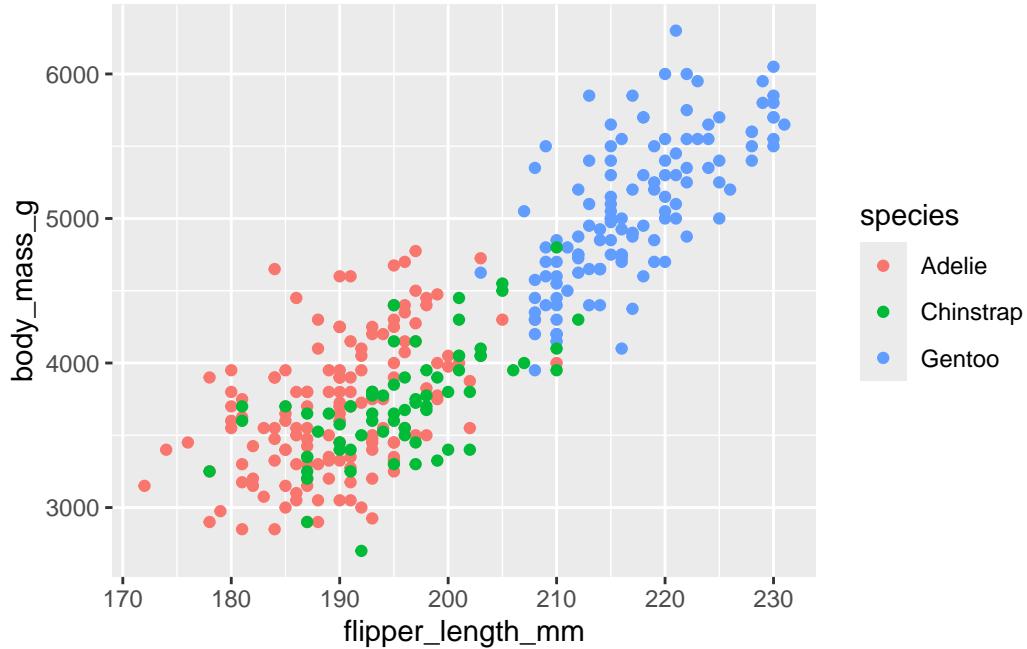


#### 4.4.7 Map Third Variables

Other variables in the dataset can be linked to plot aesthetics (visual properties) using the `mapping` argument done via the `aes()` function.

```
ggplot(  
  data = penguins,  
  mapping = aes(x = flipper_length_mm, y = body_mass_g, color = species)) +  
  geom_point()
```

Warning: Removed 2 rows containing missing values or values outside the scale range (`geom\_point()`).



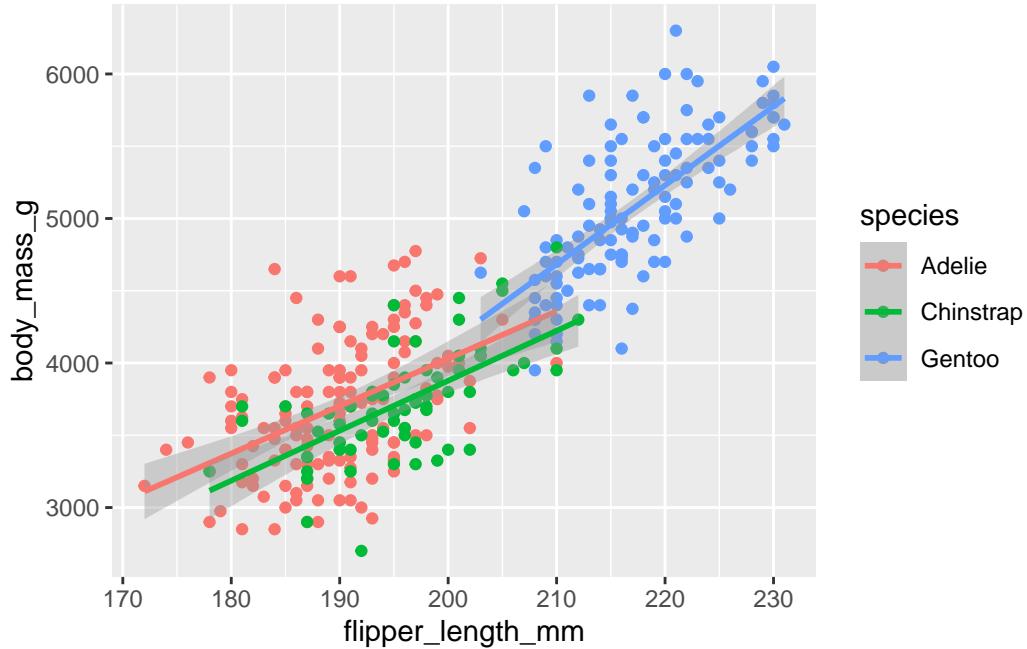
#### 4.4.8 Display Three Trendlines

More geometric representations for the data can be specified using the functions starting with `geom_` which will add layer of the selected geometric object to the plot.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g, color = species)
) +
  geom_point() +
  geom_smooth(method = "lm")`geom_smooth()` using formula = 'y ~ x'

Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_smooth()`).

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```



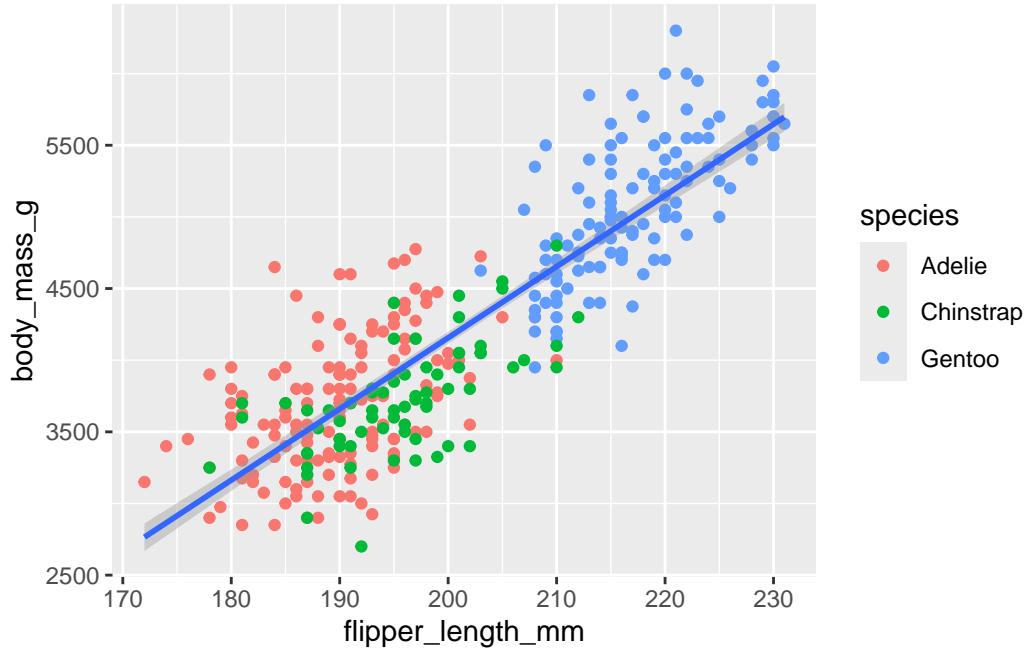
#### 4.4.9 Display One Trendline

The aesthetic mapping defined in the `ggplot()` function is *global* meaning that all the `geom_()` functions inherit it. However, the aesthetic mapping defined in the `geom_()` functions are *local*, ie, not shared with other `geom_()` functions.

```
ggplot(data = penguins,
       mapping = aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(mapping = aes(color = species)) +
  geom_smooth(method = "lm")`geom_smooth()` using formula = 'y ~ x'
```

Warning: Removed 2 rows containing non-finite outside the scale range  
(`stat\_smooth()`).

Warning: Removed 2 rows containing missing values or values outside the scale range  
(`geom\_point()`).



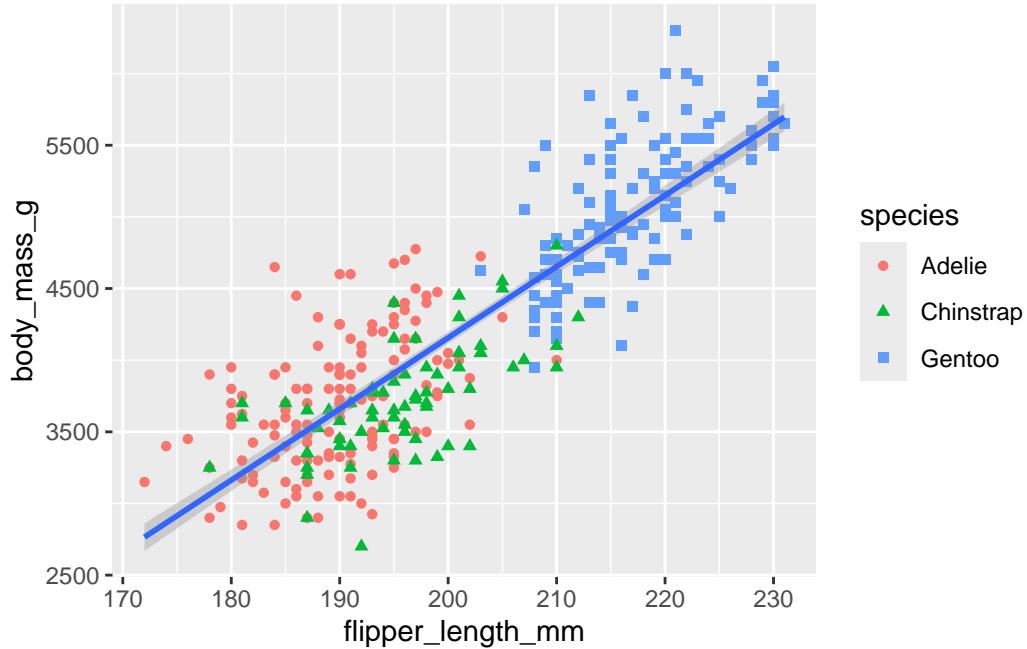
#### 4.4.10 Map One Variable Twice

We can link the same variable to multiple plot aesthetics (visual properties) using the `mapping` parameter done via the `aes()` function.

```
ggplot(data = penguins,
       mapping = aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(mapping = aes(color = species, shape = species)) +
  geom_smooth(method = "lm")`geom_smooth()` using formula = 'y ~ x'
```

Warning: Removed 2 rows containing non-finite outside the scale range  
(`stat\_smooth()`).

Warning: Removed 2 rows containing missing values or values outside the scale range  
(`geom\_point()`).



#### 4.4.11 Fix Labels

The `labs()` function can be used to make the plot more accessible. The function will add new layer to the plot and the following items can be added to the layer using the corresponding parameters

- a title using the `title` parameter
- a sub-title, if necessary, using the `subtitle` parameter
- x-axis title using the `x` parameter
- y-axis title using the `y` parameter
- data-series label or legend using the `color` and/or `shape` parameters

```
ggplot(data = penguins,
       mapping = aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(mapping = aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(
    title = 'Palmer Three Species Penguins',
    subtitle = 'The flipper length has a moderately strong positive linear relationship with',
    x = 'Flipper length (mm)',
    y = 'Body mass (g)',
    shape = 'Species',
```

```

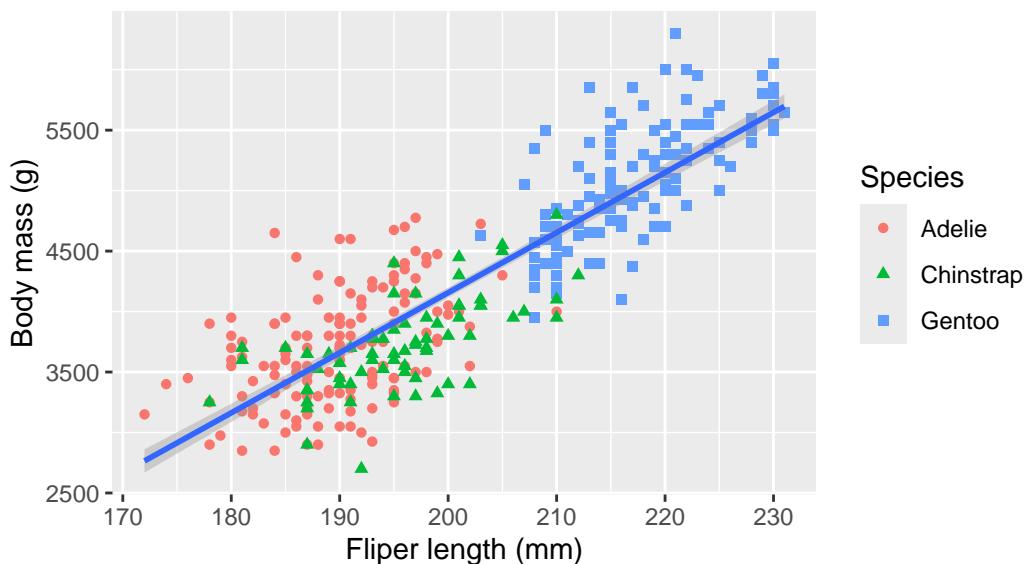
    color = 'Species'
)
`geom_smooth()` using formula = 'y ~ x'
```

Warning: Removed 2 rows containing non-finite outside the scale range  
(`stat\_smooth()`).

Warning: Removed 2 rows containing missing values or values outside the scale range  
(`geom\_point()`).

## Palmer Three Species Penguins

The flipper length has a moderately strong positive linear relationship with



Other types of texts can be added using other functions. The other types of texts are:

- x-axis label
- y-axis label
- data labels, if necessary
- annotation for interesting or important data, if exist

#### 4.4.12 Ensure Color-blind Safe

Make the plot more color-blind safe by using the `scale_color_colorblind()` function from the `ggthemes` package which will add new layer to the plot.

```
ggplot(data = penguins,
       mapping = aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(mapping = aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(
    title = 'Palmer Three Species Penguins',
    subtitle = 'The flipper length has a moderately strong positive linear relationship with',
    x = 'Flipper length (mm)',
    y = 'Body mass (g)',
    shape = 'Species',
    color = 'Species'
  ) +
  scale_color_colorblind()

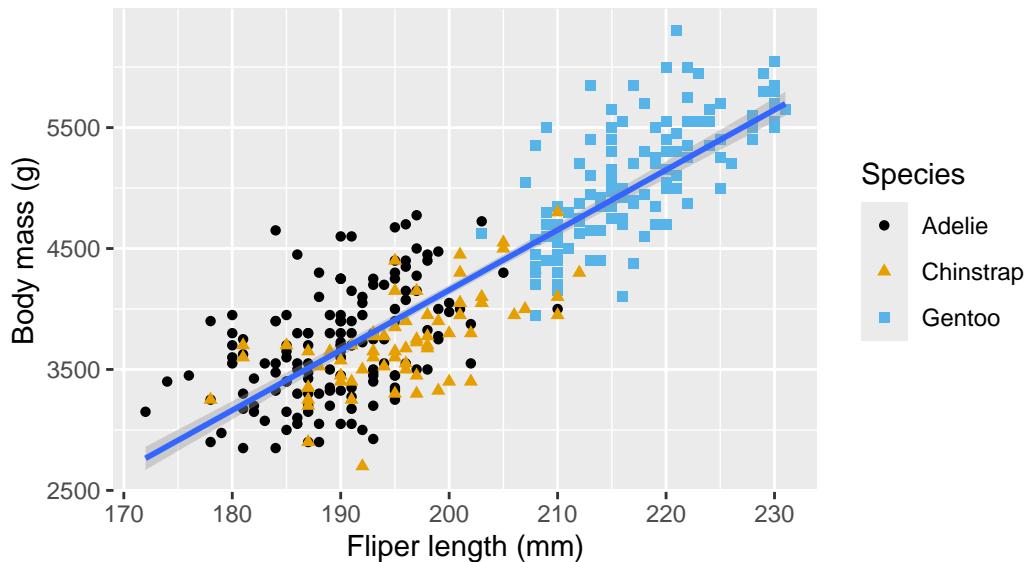
`geom_smooth()` using formula = 'y ~ x'
```

Warning: Removed 2 rows containing non-finite outside the scale range  
(`stat\_smooth()`).

Warning: Removed 2 rows containing missing values or values outside the scale range  
(`geom\_point()`).

## Palmer Three Species Penguins

The fliper length has a moderattly strong positive linear relationship with



### 4.4.13 Can Call Implicitly

The first one or two arguments of functions are so important that scientists should know them by heart. Hence, to save some typing, the name of these arguments are usually omitted and only the values assigned to them are kept, ie, the names becomes implicit and no more explicit. Hence, the above call can be written as follows—the arguments `data` and `mapping` were omitted.

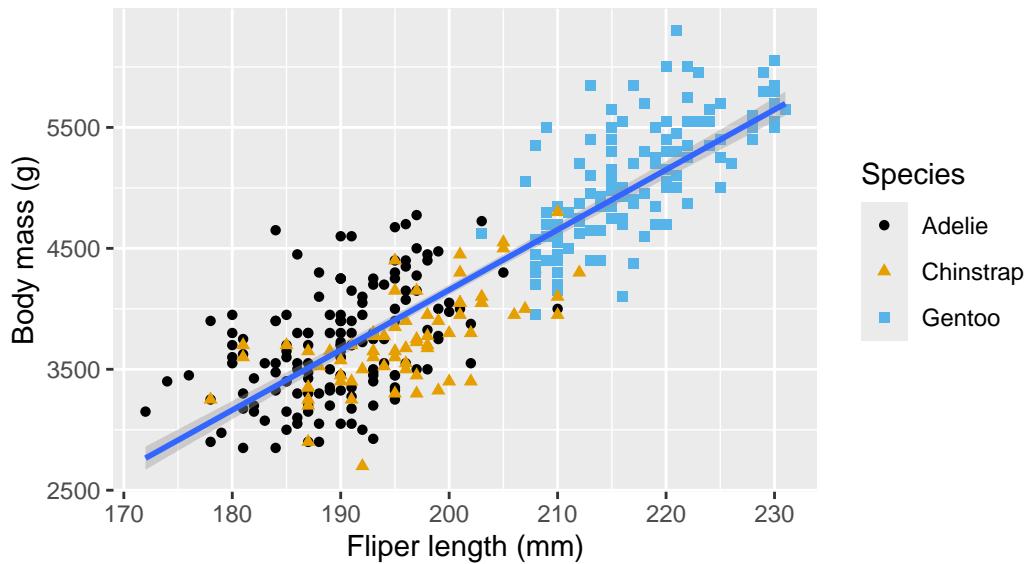
```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +  
  geom_point(aes(color = species, shape = species)) +  
  geom_smooth(method = "lm") +  
  labs(  
    title = 'Palmer Three Species Penguins',  
    subtitle = 'The fliper length has a moderattly strong positive linear relationship with  
    x = 'Fliper length (mm)',  
    y = 'Body mass (g)',  
    shape = 'Species',  
    color = 'Species'  
) +  
  scale_color_colorblind()  
  
`geom_smooth()` using formula = 'y ~ x'
```

```
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_smooth()`).
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```

## Palmer Three Species Penguins

The flipper length has a moderately strong positive linear relationship with



### 4.4.14 Use Pipe Operator

The pipe operator `\>` (shortcut: `Ctrl+M`) can be used to make the code tidy. The above code can be re-written as follow—notice the dataset was pulled before the call to the `ggplot()` function.

```
penguins |>
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(
    title = 'Palmer Three Species Penguins',
    subtitle = 'The flipper length has a moderately strong positive linear relationship with',
    x = 'Fliper length (mm)',
    y = 'Body mass (g)',
```

```

    shape = 'Species',
    color = 'Species'
) +
scale_color_colorblind()

`geom_smooth()` using formula = 'y ~ x'

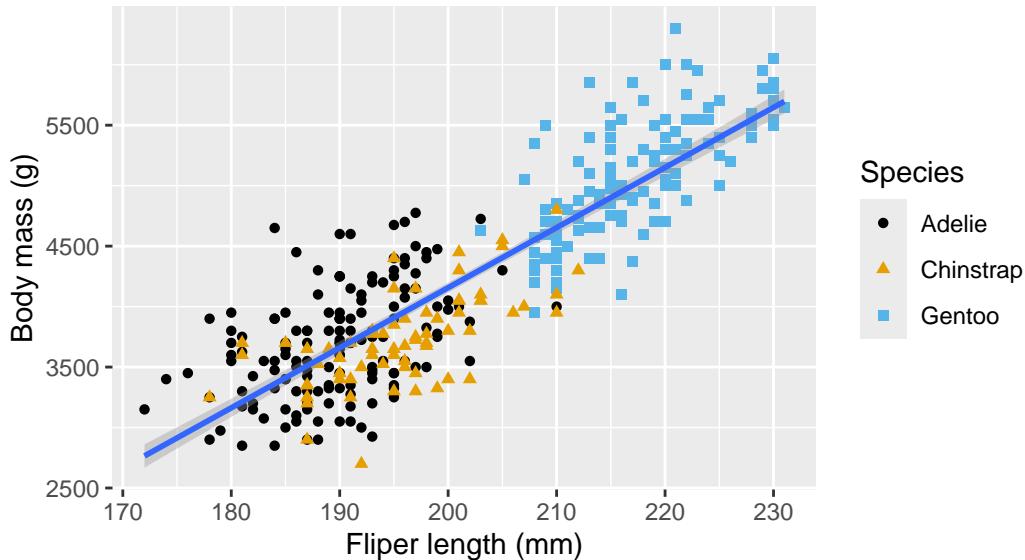
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_smooth()`).

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).

```

### Palmer Three Species Penguins

The fliper length has a moderately strong positive linear relationship with



## 4.5 Visualizing Distribution

### 4.5.1 Categorical Variables

Plot options to visualize how a categorical variable is distributed:

- bar chart, if the counts are not computed, using `geom_bar()` function
- column chart, if the counts are computed, `geom_col()` function

## 4.5.2 Numerical Variables

Plot options to visualize how a numerical (discrete or continuous) variable is distributed:

- histogram, using `geom_histogram()` function

### Histogram Bin Width

The bin width of the histogram is in the unit of the variable mapped to the plot `x` (or `y`) aesthetic (visual property)

- density plot, using `geom_density()` function
- boxplot, using `geom_boxplot()` function

### Boxplot Components

As described beautifully in [R4DS](#), a boxplot consists of:

1. A box that describes the range of the middle half of the data, a distance known as the interquartile range (IQR), stretching from the 25th percentile of the distribution to the 75th percentile.
2. A line in the middle of the box displaying the median, ie, the 50th percentile, of the distribution.
3. The box and the line give sense of the spread of the distribution and whether or not the distribution is symmetric about the median or skewed to one side
4. Visual points that display the observations that fall more than 1.5 time the IQR from either edge of the box. These outlying points (hence called outliers) are unusual so are plotted individually
5. A whisker that extend from each end of the box and goes to the farthest non-outlier point in the distribution

Below is the diagram from [R4DS](#) showing the above components and how the boxplot is created.

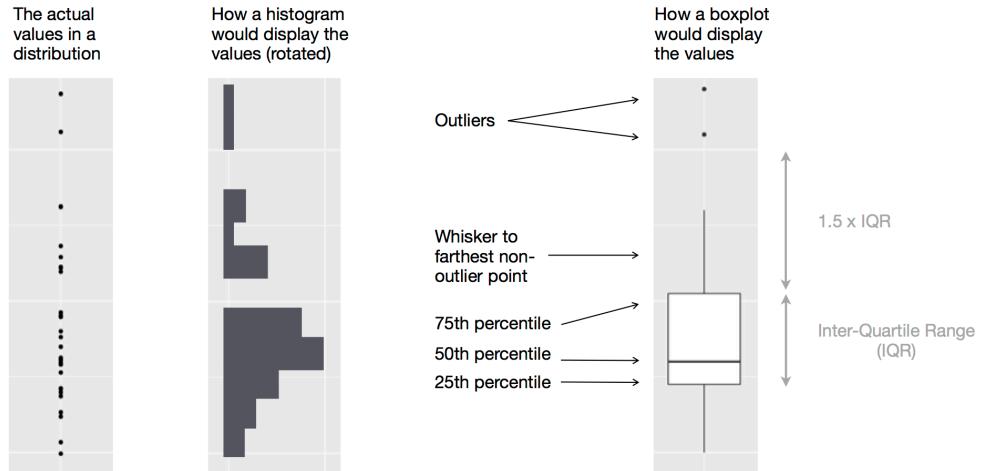


Figure 4.1: Boxplot Components (taken from [R4DS](#))

## 4.6 Visualizing Relationships

### 4.6.1 One Categorical + One Numerical

For each category of the categorical variable, We can use any of the plot options mentioned above for the numerical variables

### 4.6.2 Two Categoricals

Each category of one of the categorical variables will be placed on the x-axis (or the y-axis) by mapping it to the plot `x` (or `y`) aesthetic (visual property) of the `geom_bar()` and the distribution of the categories of the other categorical variables by mapping it to the plot `fill` aesthetic (visual property). The second variable can be shown as:

- pure counts (stacked bar chart), or
- percentages (percent stack bar chart) by setting the `position` attribute of the `geom_bar()` to `fill`.

### 4.6.3 Two Numerical

Plot options to show the relationship between two numerical variables are:

- Scatter plot using the `geom_point()` function
- trend line using `geom_smooth()` function
- line graph using `geom_line()` function if one of the variables is monotonic, eg, time or date.

### 4.6.4 Three or More Variables

To visualize 3+ variables, We can either

- map variables to other aesthetics of the plot, eg, `color`, `size`, and `shape`
- split plot into facets, subplots that each display one subset of the data, based on a categorical variable using `facet_wrap()` function where its first argument is a formula created using `~` followed by a (categorical) variable name.

# **Part III**

# **Practice**

This section contains practices for the programming/scripting languages that I am learning.

# 5 Topic Modeling in R

## 5.1 Introduction

An attempt to understand Sherlock Holmes short stories found in Adventures of Sherlock Holmes book by Arthur Conan Doyle. After inspecting the table of content, the book seems to have 12 stories, one story per chapter. The analysis is inspired by [Julia Silge's YouTube video Topic modeling with R and tidy data principles](#)

## 5.2 Download Book

```
library(gutenbergr) # download books from Project Gutenberg using book ID
library(tidyverse)

-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate  1.9.3     v tidyr    1.3.1
v purrr     1.0.2
-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```
library(tidytext)

# Download the book, each line of the book is read into a seperate row
sherlock_raw <- gutenberg_download(48320)
```

Determining mirror for Project Gutenberg from <https://www.gutenberg.org/robot/harvest>  
Using mirror <http://aleph.gutenberg.org>

```
dim(sherlock_raw)
```

```
[1] 12350      2
```

```
head(sherlock_raw)
```

```
# A tibble: 6 x 2
  gutenberg_id text
  <int> <chr>
1     48320 "ADVENTURES OF SHERLOCK HOLMES"
2     48320 ""
3     48320 ""
4     48320 ""
5     48320 ""
6     48320 "[Illustration:"
```

```
tail(sherlock_raw)
```

```
# A tibble: 6 x 2
  gutenberg_id text
  <int> <chr>
1     48320 "boisterous fashion, and on the whole _changed to_"
2     48320 "boisterous fashion, and on the whole"
3     48320 ""
4     48320 "Page 297"
5     48320 "wrapt in the peaceful beauty _changed to_"
6     48320 "rapt in the peaceful beauty"
```

## 5.3 Wrangle: Label Stories

```
sherlock <- sherlock_raw %>%
  # determine start of each story/chapter
  mutate(story = ifelse(str_detect(text, "^(A SCANDAL IN BOHEMIA|THE RED-HEADED LEAGUE|A CAS"))
```

# determine lines belonging to each story/chapter by  
# filling down the N/A rows of story column

```

fill(story) %>%
  # remove the part that does not belong to any story/chapter,
  # i.e, the introduction
  filter(!is.na(story)) %>%
  # convert story column to factor
  mutate(story = factor(story))

```

## 5.4 Wrangle: Put in Tidy Format

The row of `text` column contains multiple words/tokens. We want to put each word/token of each `text` row into a separate row. This makes the dataframe follows the tidy format and hence makes it easy to process.

```

tidy_sherlock <- sherlock %>%
  # number the rows
  mutate(line = row_number()) %>%
  # break the text column into multiple row where each row contain one token
  unnest_tokens(word, text) %>%
  # remove the stopwords--the rows where the word column is a stopword
  anti_join(stop_words) %>%
  # remove holmes rows which might affect our topic models
  filter(word != "holmes")

```

Joining with `by = join\_by(word)`

## 5.5 Explore tf-idf

- To see which words are important in each story/chapter, i.e., the words that appears many times in that story but few or none in the other stories.
- tf-idf (term frequency-inverse document frequency) is a great exploratory tool before starting with topic modeling

```

library(ggplot2)

tidy_sherlock %>%

# count number of occurrence of words in stories
count(story, word, sort = TRUE) %>%

# compute and add tf, idf, and tf_idf values for words
bind_tf_idf(word, story, n) %>%

# group by story
group_by(story) %>%

# take top 10 words of each story with highest tf_idf (last column)
top_n(10) %>%

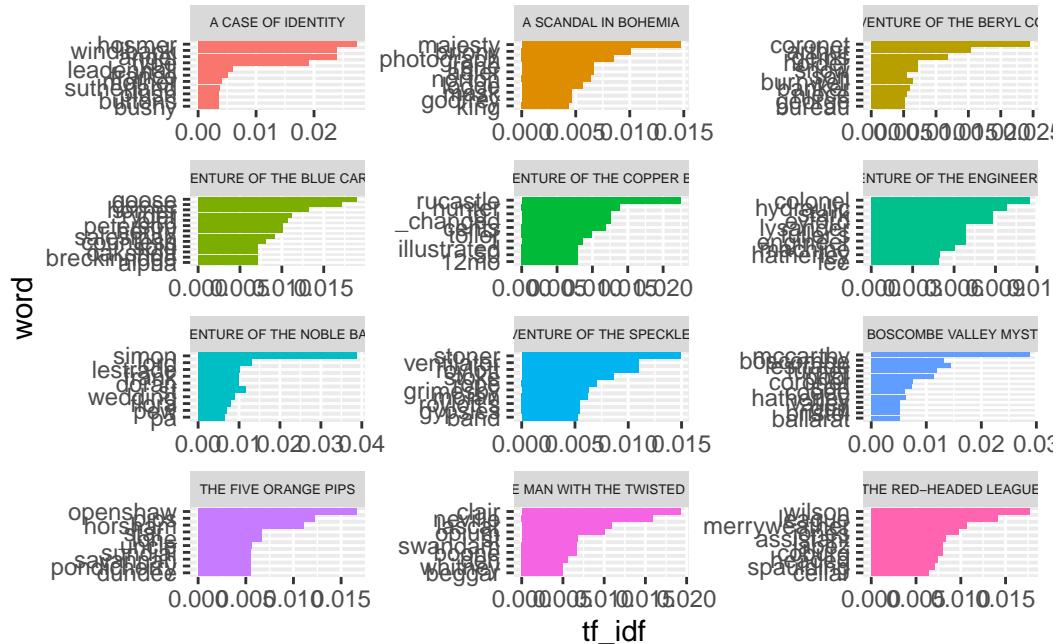
# unpack
ungroup() %>%

# turn words into factors and order them based on their tf_idf values
# NOTE: This will not affect order the dataframe rows which is can be
#   done via the arrange function
# NOTE: Recording the word column this way is for ggplot to visualize them
#   as desired from top tf_idf to lowest
mutate(word = reorder(word, tf_idf)) %>%

# plot
ggplot(aes(word, tf_idf, fill = story)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~story, scales = "free", ncol = 3) +
  theme(strip.text.x = element_text(size = 5)) +
  coord_flip()

```

Selecting by tf\_idf



## 5.6 Implement Topic Modeling

Training the model for the topics

```
library(stm)      # for do topic modeling
```

```
stm v1.3.7 successfully loaded. See ?stm for help.  
Papers, resources, and other materials at structuraltopicmodel.com
```

```
library(quanteda)  # great text mining, will be used to structure the input
```

```
Package version: 4.0.2  
Unicode version: 15.1  
ICU version: 74.1
```

```
Parallel computing: 8 of 8 threads used.
```

See <https://quanteda.io> for tutorials and examples.

```

#     to stm

# Convert from tidy form to quanteda form (document x term matrix)
sherlock_stm <- tidy_sherlock %>%
  count(story, word, sort = TRUE) %>%
  cast_dfm(story, word, n)

# Train the model
topic_model <- stm(sherlock_stm, K=6, init.type = "Spectral")

```

```

Beginning Spectral Initialization
  Calculating the gram matrix...
  Finding anchor words...
  .....
  Recovering initialization...
  .....
Initialization complete.
  .....
Completed E-Step (0 seconds).
Completed M-Step.
Completing Iteration 1 (approx. per word bound = -7.785)
  .....
Completed E-Step (0 seconds).
Completed M-Step.
Completing Iteration 2 (approx. per word bound = -7.593, relative change = 2.458e-02)
  .....
Completed E-Step (0 seconds).
Completed M-Step.
Completing Iteration 3 (approx. per word bound = -7.481, relative change = 1.473e-02)
  .....
Completed E-Step (0 seconds).
Completed M-Step.
Completing Iteration 4 (approx. per word bound = -7.455, relative change = 3.469e-03)
  .....
Completed E-Step (0 seconds).
Completed M-Step.
Completing Iteration 5 (approx. per word bound = -7.450, relative change = 7.612e-04)
Topic 1: st, simon, lord, day, lady
Topic 2: door, miss, house, rucastle, matter
Topic 3: hat, goose, stone, bird, geese
Topic 4: father, time, mccarthy, son, hand
Topic 5: house, time, night, door, heard

```

```

Topic 6: red, time, wilson, business, headed
.....
Completed E-Step (0 seconds).
Completed M-Step.
Completing Iteration 6 (approx. per word bound = -7.449, relative change = 1.233e-04)
.....
Completed E-Step (0 seconds).
Completed M-Step.
Completing Iteration 7 (approx. per word bound = -7.449, relative change = 1.168e-05)
.....
Completed E-Step (0 seconds).
Completed M-Step.
Model Converged

```

```
summary(topic_model)
```

A topic model with 6 topics, 12 documents and a 7709 word dictionary.

Topic 1 Top Words:

```

Highest Prob: st, simon, lord, day, lady, found, matter
FREX: simon, clair, neville, lascar, opium, doran, flora
Lift: aloysius, ceremony, doran, millar, 2_s, aberdeen, absurdly
Score: simon, st, clair, neville, _danseuse_, lestrade, doran

```

Topic 2 Top Words:

```

Highest Prob: door, miss, house, rucastle, matter, street, lady
FREX: rucastle, hosmer, hunter, angel, windibank, _changed, 1
Lift: advertised, angel, annoyance, brothers, employed, factor, fowler
Score: rucastle, hosmer, angel, windibank, hunter, type, 1

```

Topic 3 Top Words:

```

Highest Prob: hat, goose, stone, bird, geese, baker, sir
FREX: geese, horner, ryder, henry, peterson, salesman, countess
Lift: battered, bet, bred, brixton, cosmopolitan, covent, cream
Score: goose, geese, horner, _alias_, ryder, henry, peterson

```

Topic 4 Top Words:

```

Highest Prob: father, time, mccarthy, son, hand, lestrade, left
FREX: mccarthy, pool, boscombe, openshaw, pips, horsham, turner
Lift: bone, dundee, horsham, pondicherry, presumption, savannah, sundial
Score: mccarthy, pool, lestrade, boscombe, openshaw, _détour_, turner

```

Topic 5 Top Words:

```

Highest Prob: house, time, night, door, heard, hand, round
FREX: coronet, stoner, arthur, roylott, ventilator, gems, stoke
Lift: _absolute_, _all_, _en, 1100, 16a, 3d, 4000

```

```

Score: coronet, arthur, stoner, gems, 4000, roylott, ventilator
Topic 6 Top Words:
Highest Prob: red, time, wilson, business, headed, day, league
FREX: wilson, league, merryweather, jones, coburg, jabez, headed
Lift: daring, saturday, vincent, _employé_, _october, _partie, 17
Score: wilson, league, merryweather, _employé_, jones, headed, coburg

```

## 5.7 Contribution of Words in Topics

Looking at which words contribute the most in each topic.

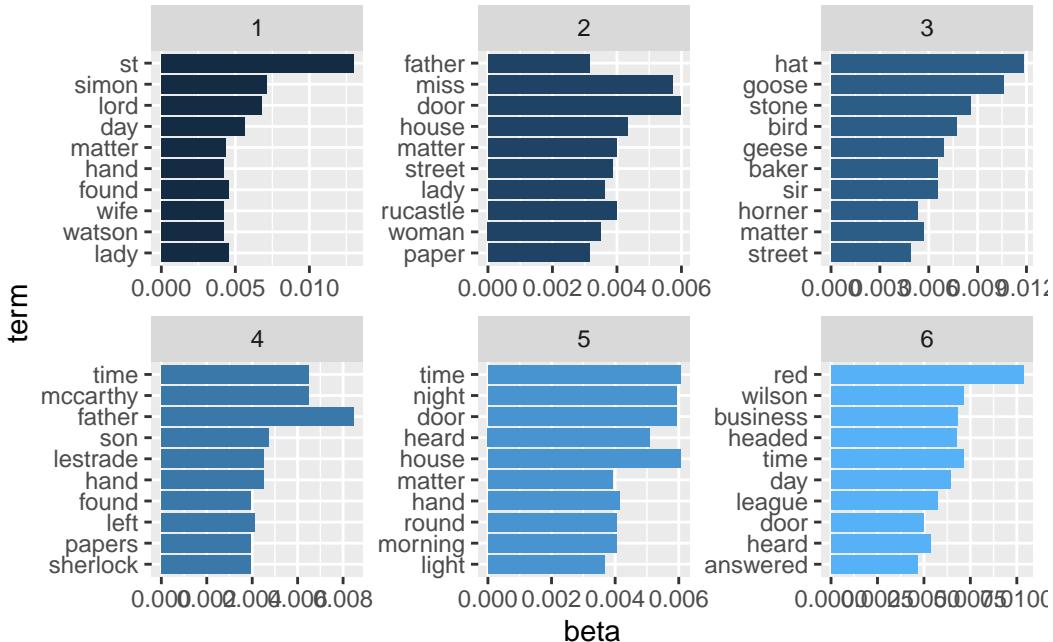
```

# Extracting betas and putting them in a tidy format
tm_beta <- tidy(topic_model)

# Visualizing the top words contributing to each topic
tm_beta %>%
  group_by(topic) %>%
  # top 10 word in each topic with higest beta (last column)
  top_n(10) %>%
  ungroup() %>%
  # turn words into factors and order them based on their tf_idf values
  # NOTE: This will not affect order the dataframe rows which is can be
  #       done via the arrange function
  # NOTE: Recording the word column this way is for ggplot to visualize them
  #       as desired from top tf_idf to lowest
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = topic)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~topic, scales = "free", ncol = 3) +
  coord_flip()

```

Selecting by beta



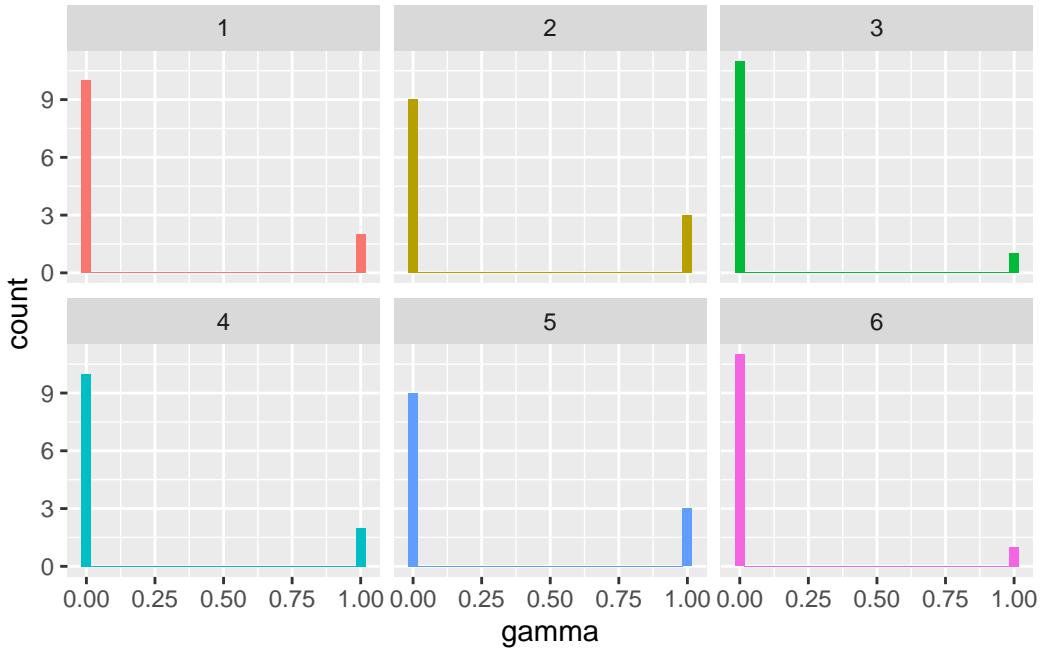
## 5.8 Distribution of Topics in Stories

Looking at how the stories are associated with each topic and how strong each association is.

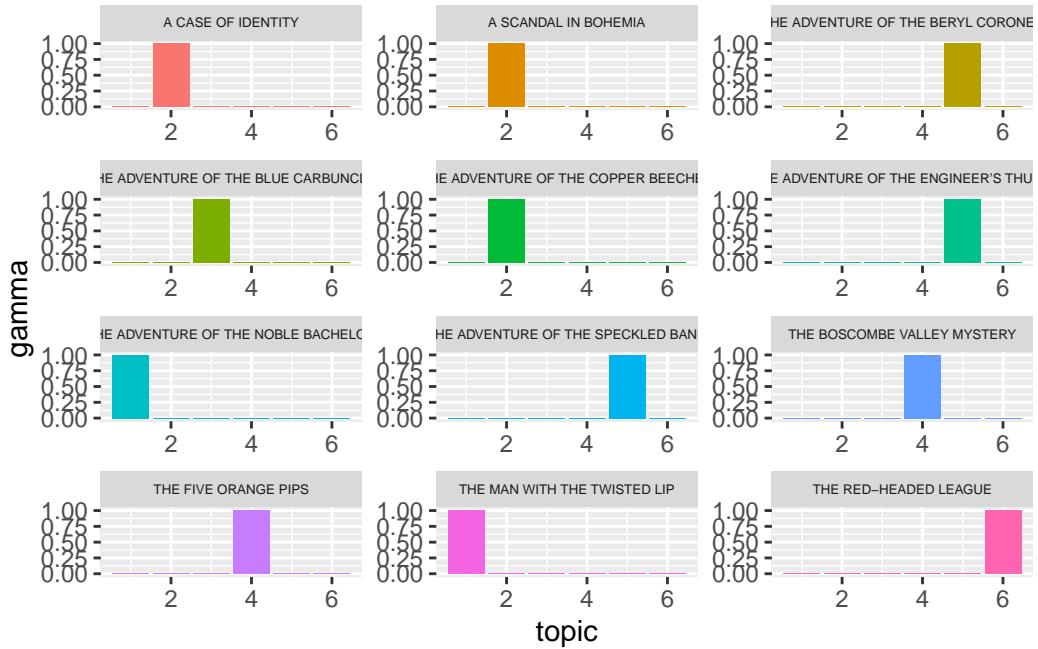
```
# Extracting gammas and putting them in a tidy format
tm_gamma <- tidy(topic_model, matrix = "gamma",
  # use the names of the stories instead of the default numbers
  document_names = rownames(sherlock_stm))

# Visualizing the number of stories belonging to each topics and the confidence
# of the belonging
tm_gamma %>%
  ggplot(aes(gamma, fill = as.factor(topic))) +
  geom_histogram(show.legend = FALSE) +
  facet_wrap(~topic, ncol = 3)

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



```
# Visualizing how much each topic appear in each story
tm_gamma %>%
  ggplot(aes(topic, gamma, fill = document)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~document, scales = "free", ncol = 3) +
  theme(strip.text.x = element_text(size = 5))
```



The model did an excellent job strongly associating the stories into one or more topics. This perfect association is rare in the world of topic modeling. The reason behind this perfect association here could be due to the small number of documents that we have.

## 5.9 References

- Adventures of Sherlock Holmes book by Arthur Conan Doyle on [Project Gutenberg](#)
- [Regular Expressions 101](#)