An Introduction to R and RStudio for Exploratory Data Analysis

Jessica Minnier, PhD & Meike Niederhausen, PhD
OCTRI Biostatistics, Epidemiology, Research & Design (BERD) Workshop

Part 1: 2020/09/16 & Part 2: 2020/09/17

slides: bit.ly/berd_intro_part2
pdf: bit.ly/berd_intro_part2_pdf

An Introduction to R and RStudio for Exploratory Data Analysis (Part 2)

Instructors: Meike Niederhausen, PhD & Jessica Minnier, PhD
OCTRI Biostatistics, Epidemiology, Research & Design (BERD) Workshop

Do this now:

1. **Open html slides**: bit.ly/berd_intro_part2
2. **Open google doc** for asking questions: bit.ly/berd_doc
    - Helpers will be monitoring this, you can ask questions, copy code or screenshots.
3. **Open Rstudio with these steps:**
    - Open the folder from yesterday
    - Double click on the `.Rproj` file.
    - All your files should be there.

Working with data, we will use the pipe %>%

The pipe operator `%>%` is part of the tidyverse, and strings together commands to be performed sequentially

```
penguins %>% head(n=3)      # prounounce %>% as "then"
```

```
## # A tibble: 3 x 9
##      id species island bill_length_mm bill_depth_mm flipper_length_… body_mass_
##   <dbl> <chr>   <chr>           <dbl>         <dbl>             <dbl>       <dbl
## 1  1689 Adelie  Torge…           39.1          18.7               181         375
## 2  4274 Adelie  Torge…           NA            17.4               186         380
## 3  4539 Adelie  Torge…           40.3          18                 195         325
## # … with 2 more variables: sex <chr>, year <dbl>
```

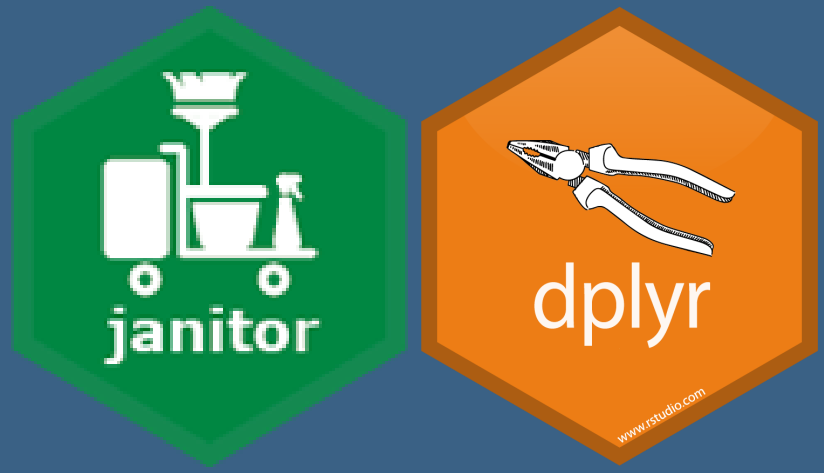
- Always *first list the tibble* that the commands are being applied to
- Can use **multiple pipes** to run multiple commands in sequence
  - What does the following code do?

```
penguins %>% head(n=2) %>% summary()
```

Quick tips on summarizing data

categorical data

numerical data



janitor, dplyr

Numerical data summaries: $ vs summarize()

We saw how to summarize a vector pulled with $, but there are easier ways to summarize multiple columns at once.

```
mean(penguins$body_mass_g)
```

```
## [1] 4201.754
```

```
median(penguins$body_mass_g)
```

```
## [1] 4050
```

```
penguins %>%
  summarize(mean(body_mass_g),
            median(body_mass_g))
```

```
## # A tibble: 1 x 2
##   `mean(body_mass_g)` `median(body_mass_g)`
##                 <dbl>                 <dbl>
## 1               4202.                  4050
```

`summarize()` with NA

- Don't forget `na.rm = TRUE` if you need it.
- You can also name these columns.

```
penguins %>%
  summarize(mean_mass = mean(body_mass_g),
            mean_len = mean(bill_length_mm, na.rm = TRUE))
```

```
## # A tibble: 1 x 2
##   mean_mass mean_len
##       <dbl>    <dbl>
## 1     4202.     44.0
```

By group `summarize()` (1/2)

- We can summarize data as a whole, or in groups with `group_by()`
- `group_by()` is very powerful, see data wrangling cheatsheet

```
# summary of all data as a whole
penguins %>%
  summarize(mass_mean =mean(body_mass_g),
            mass_sd = sd(body_mass_g),
            mass_cv = sd(body_mass_g)/mean(body_mass_g))
```

```
## # A tibble: 1 x 3
##   mass_mean mass_sd mass_cv
##       <dbl>   <dbl>   <dbl>
## 1     4202.    802.   0.191
```

By group `summarize()` (2/2)

- We can summarize data as a whole, or in groups with `group_by()`
- `group_by()` is very powerful, see data wrangling cheatsheet

```
# summary by group variable
penguins %>%
  group_by(species) %>%
  summarize(n_per_group = n(),
            mass_mean =mean(body_mass_g),
            mass_sd = sd(body_mass_g),
            mass_cv = sd(body_mass_g)/mean(body_mass_g))
```

```
## # A tibble: 3 x 5
##   species   n_per_group mass_mean mass_sd mass_cv
##   <chr>           <int>     <dbl>   <dbl>   <dbl>
## 1 Adelie            151     3701.    459.  0.124
## 2 Chinstrap          68     3733.    384.  0.103
## 3 Gentoo            123     5076.    504.  0.0993
```

Advanced `summarize(across())` (1/3)

- Can also use `across()` to summarize multiple variables (more examples)

```
penguins %>%
  summarize(across(c(body_mass_g, bill_depth_mm), mean))
```

```
## # A tibble: 1 x 2
##   body_mass_g bill_depth_mm
##         <dbl>         <dbl>
## 1       4202.          17.2
```

```
penguins %>%
  summarize(across(where(is.numeric), mean, na.rm=TRUE))
```

```
## # A tibble: 1 x 6
##      id bill_length_mm bill_depth_mm flipper_length_mm body_mass_g  year
##   <dbl>          <dbl>         <dbl>             <dbl>       <dbl> <dbl>
## 1 3031.           44.0          17.2              201.       4202. 2008.
```

Advanced `summarize(across())` (2/3)

- Can also use `across()` to summarize multiple variables *and* functions (more examples)

```
penguins %>%
  summarize(across(c(body_mass_g, bill_depth_mm),
                   c(m = mean, sd = sd)))
```

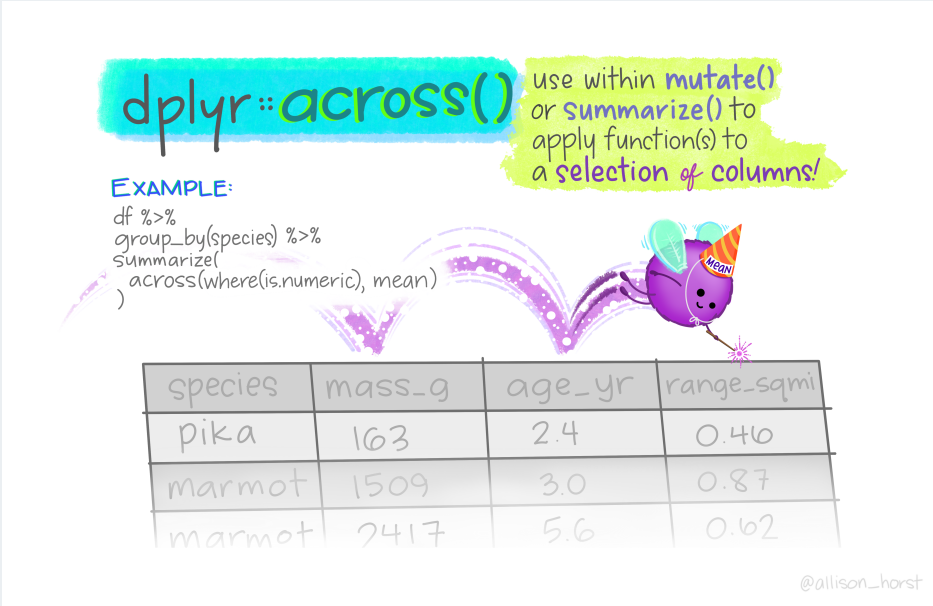
```
## # A tibble: 1 x 4
##   body_mass_g_m body_mass_g_sd bill_depth_mm_m bill_depth_mm_sd
##           <dbl>          <dbl>           <dbl>             <dbl>
## 1         4202.           802.            17.2              1.97
```

Advanced `summarize(across())` (3/3)

- Can also use `across()` to summarize based on true/false conditions (more examples)

```
penguins %>%
  summarize(
    across(where(is.character),
           n_distinct))
```

```
## # A tibble: 1 x 3
##   species island   sex
##     <int>  <int> <int>
## 1       3      3     3
```



Allison Horst

Frequency tables: simple `count()`

```
penguins %>% count(island)
```

```
## # A tibble: 3 x 2
##   island        n
##   <chr>     <int>
## 1 Biscoe      167
## 2 Dream       124
## 3 Torgersen    51
```

```
penguins %>% count(species, island)
```

```
## # A tibble: 5 x 3
##   species   island        n
##   <chr>     <chr>     <int>
## 1 Adelie    Biscoe       44
## 2 Adelie    Dream        56
## 3 Adelie    Torgersen    51
## 4 Chinstrap Dream        68
## 5 Gentoo    Biscoe      123
```

Fancier frequency tables: `janitor` package's `tabyl` function

```
# default table
penguins %>% tabyl(species)
```

```
##    species   n   percent
##     Adelie 151 0.4415205
##  Chinstrap  68 0.1988304
##     Gentoo 123 0.3596491
```

```
# output can be treated as tibble
penguins%>%tabyl(species)%>%select(-n)
```

```
##    species   percent
##     Adelie 0.4415205
##  Chinstrap 0.1988304
##     Gentoo 0.3596491
```

adorn_ your table!

```
penguins %>%
  tabyl(species) %>%
  adorn_totals("row") %>%
  adorn_pct_formatting(digits=2)
```

```
##    species   n percent
##     Adelie 151  44.15%
##  Chinstrap  68  19.88%
##     Gentoo 123  35.96%
##      Total 342 100.00%
```

2x2 tabyls

```
# default 2x2 table
penguins %>%
  tabyl(species, sex)
```

```
##    species female male NA_
##     Adelie     73   73   5
##  Chinstrap     34   34   0
##     Gentoo     58   61   4
```

```
penguins %>% tabyl(species, sex) %>%
  adorn_percentages(denominator = "col") %>%
  adorn_totals("row") %>%
  adorn_pct_formatting(digits = 1) %>%
  adorn_ns()
```

```
##    species       female          male
##     Adelie  44.2%  (73)  43.5%  (73)  55.6%
##  Chinstrap  20.6%  (34)  20.2%  (34)   0.0%
##     Gentoo  35.2%  (58)  36.3%  (61)  44.4%
##      Total 100.0% (165) 100.0% (168) 100.0%
```

What adornments does the tabyl to right have?

- Base R has a `table` function, but it is clunkier and the output is not a data frame.
- See the tabyl vignette for more information, adorn options, & 3-way `tabyls`

3 way tabyls are possible

```
penguins %>% tabyl(species, island, sex)
```

```
## $female
##    species Biscoe Dream Torgersen
##     Adelie     22    27        24
##  Chinstrap      0    34         0
##     Gentoo     58     0         0
##
## $male
##    species Biscoe Dream Torgersen
##     Adelie     22    28        23
##  Chinstrap      0    34         0
##     Gentoo     61     0         0
##
## $NA_
##    species Biscoe Dream Torgersen
##     Adelie      0     1         4
##  Chinstrap      0     0         0
##     Gentoo      4     0         0
```

Practice 3

1. Continue adding code chunks to your Rmd (or, start a new one! But remember to load the libraries and data at the top.)

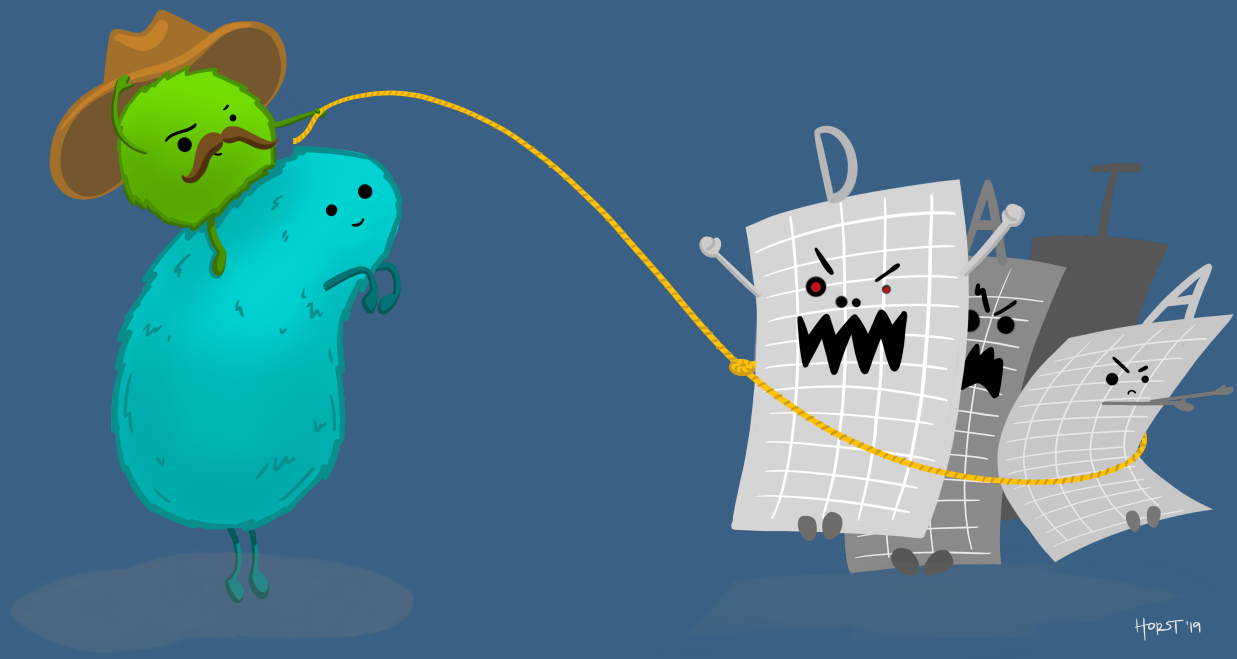
2. How many different years are in the data? (Hint: use `tabyl()` or `n_distinct()`)

3. Count the number of penguins measured each year.

4. Calculate the median body mass by each species and sex subgroup. Use `summarize()` and `group_by()` to do this.

5. Create a 2x2 table of number of penguins measured in each year by each island.

Data Wrangling



Allison Horst

Subsetting data

Subset Observations (Rows)

Subset Variables (Columns)

tidyverse data wrangling cheatsheet

`filter()` ~ rows

filter data based on rows

- math: >, <, >=, <=
- double = for "is equal to": ==
- & (and)
- | (or)
- != (not equal)

- `is.na()` to filter based on missing values
- `%in%` to filter based on group membership
- `!` in front negates the statement, as in
  - `!is.na(age)`
  - `!(grade %in% c("9th","10th"))`

```
mydata %>% filter(bmi > 20)
```

```
## # A tibble: 15 x 10
##         id age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##      <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
##  1 3.35e5 17 y… Fema… 10th  White  27.6      66.2 <NA>             <NA>
##  2 6.39e5 16 y… Fema… 9th   <NA>   29.3      84.8 <NA>             Yes
##  3 9.23e5 15 y… Male  9th   White  21.4      60.3 <NA>             Yes
##  4 9.26e5 16 y… Male  10th  All …  22.2      70.3 <NA>             No
##  5 9.34e5 16 y… Fema… 10th  All …  21.0      45.4 <NA>             Yes
##  6 1.10e6 15 y… Male  10th  All …  22.5      79.4 <NA>             <NA>
##  7 1.11e6 17 y… Fema… 9th   Blac…  26.6      68.0 <NA>             No
```

`filter()` practice

What do these commands do? Try them out:

```
mydata %>% filter(age == "14 years old")
mydata %>% filter(bmi/weight_kg < 0.5)    # can do math
mydata %>% filter((bmi < 15) | (bmi > 25))
mydata %>% filter(bmi < 20, weight_kg < 60, sex == "Female") # filter on multiple

mydata %>% filter(id == 923122)      # note the use of == instead of just =
mydata %>% filter(sex == "Female")
mydata %>% filter(!(grade == "9th"))
mydata %>% filter(grade %in% c("10th", "11th"))

mydata %>% filter(is.na(bmi))
mydata %>% filter(!is.na(bmi))
```

`select()` ~ columns

- select columns (variables)
- no quotes needed around variable names
- can be used to rearrange columns
- uses special syntax that is flexible and has many options

```
mydata %>% select(id, grade)
```

```
## # A tibble: 20 x 2
##         id grade
##      <dbl> <chr>
##  1  335340 10th
##  2  638618 9th
##  3  922382 9th
##  4  923122 9th
##  5  923963 10th
##  6  925603 10th
##  7  933724 10th
##  8  935435 12th
##  9 1096564 10th
## 10 1108114 9th
## 11 1306150 10th
## 12 1307481 12th
```

Column selection syntax options

There are many ways to select a set of variable names (columns):

- `var1:var20`: all columns from `var1` to `var20`
- `one_of(c("a", "b", "c"))`: all columns with names in the specified character vector of names
- **Removing columns**
  - `-var1`: remove the column`var1`
  - `-(var1:var20)`: remove all columns from `var1` to `var20`
- **Select using text within column names**
  - `contains("date")`, `contains("_")`: all variable names that contain the specified string
  - `starts_with("a")` or `ends_with("last")`: all variable names that start or end with the specificed string
- **Rearranging columns**
  - use `everything()` to select all columns not already named
  - example: `select(var1, var20, everything())` moves the column `var20` to the second position

See other examples in the data wrangling cheatsheet.

`select()` practice

Which columns are selected & in what order using these commands?
First guess and then try them out.

```
mydata %>% select(id:sex)
mydata %>% select(one_of(c("age","weight_kg")))

mydata %>% select(-grade,-sex)
mydata %>% select(-(id:sex))

mydata %>% select(contains("race"))
mydata %>% select(starts_with("r"))
mydata %>% select(-contains("r"))

mydata %>% select(id, race4, everything())
```

Changing the data

Alison Horst

Save a new data frame, or overwrite existing one

Use a new variable name on left side of <- assignment to save updated data frames:

```
mydata_new <- mydata %>% select(id:sex)
mydata_new
```

```
## # A tibble: 20 x 3
##         id age                sex
##      <dbl> <chr>              <chr>
##  1  335340 17 years old       Female
##  2  638618 16 years old       Female
##  3  922382 14 years old       Male
##  4  923122 15 years old       Male
##  5  923963 15 years old       Male
##  6  925603 16 years old       Male
##  7  933724 16 years old       Female
##  8  935435 17 years old       Female
##  9 1096564 15 years old       Male
## 10 1108114 17 years old       Female
## 11 1306150 16 years old       Male
## 12 1307481 17 years old       Male
## 13 1307872 17 years old       Male
```

`rename()` ~ columns

- renames column variables
- `%>% rename(new_name = old_name)`

Renames the column, just prints the output

```
# This does not save the new name
mydata %>% rename(record = id)
```

```
## # A tibble: 20 x 10
##    record age   sex   grade race4
##     <dbl> <chr> <chr> <chr> <chr> <
##  1 3.35e5 17 y… Fema… 10th  White
##  2 6.39e5 16 y… Fema… 9th   <NA>
##  3 9.22e5 14 y… Male  9th   White
##  4 9.23e5 15 y… Male  9th   White
##  5 9.24e5 15 y… Male  10th  Blac…
##  6 9.26e5 16 y… Male  10th  All …
##  7 9.34e5 16 y… Fema… 10th  All …
##  8 9.35e5 17 y… Fema… 12th  All …
##  9 1.10e6 15 y… Male  10th  All …
```

Renames the column *and* overwrites `mydata` with renamed column:

```
mydata <- mydata %>% rename(record = i
mydata
```

```
## # A tibble: 20 x 10
##    record age   sex   grade race4
##     <dbl> <chr> <chr> <chr> <chr> <
##  1 3.35e5 17 y… Fema… 10th  White
##  2 6.39e5 16 y… Fema… 9th   <NA>
##  3 9.22e5 14 y… Male  9th   White
##  4 9.23e5 15 y… Male  9th   White
##  5 9.24e5 15 y… Male  10th  Blac…
##  6 9.26e5 16 y… Male  10th  All …
##  7 9.34e5 16 y… Fema… 10th  All …
##  8 9.35e5 17 y… Fema… 12th  All …
```

Make new variables

tidyverse data wrangling cheatsheet

mutate()

Use `mutate()` to add new columns to a tibble

- many options in how to define new column of data

```
newdata <- mydata %>%
   mutate(height_m = sqrt(weight_kg / bmi))   # use = (not <- or ==) to define new

newdata %>% select(bmi, weight_kg, height_m)
```

```
## # A tibble: 20 x 3
##       bmi weight_kg height_m
##     <dbl>     <dbl>    <dbl>
##  1  27.6       66.2     1.55
##  2  29.3       84.8     1.70
##  3  18.2       57.6     1.78
##  4  21.4       60.3     1.68
##  5  19.6       63.5     1.80
##  6  22.2       70.3     1.78
##  7  21.0       45.4     1.47
##  8  17.5       43.1     1.57
##  9  22.5       79.4     1.88
## 10  26.6       68.0     1.60
```

`mutate()` practice

What do the following commands do?
First guess and then try them out.

```
mydata %>% mutate(bmi_high = (bmi > 30))

mydata %>% mutate(male = (sex == "Male"))
mydata %>% mutate(male = 1 * (sex == "Male"))

mydata %>% mutate(grade_num = as.numeric(str_remove(grade, "th")))
```

Practice 3

1. Continue adding code chunks to your Rmd (or, start a new one! But remember to load the libraries and data at the top.)

2. How many different years are in the data? (Hint: use `tabyl()` or `n_distinct()`)

3. Count the number of penguins measured each year.

4. Calculate the median body mass by each species and sex subgroup. Use `summarize()` and `group_by()` to do this.

5. Create a 2x2 table of number of penguins measured in each year by each island.

Save data frame

- Save **.RData** file: the standard R format, which is recommended if saving data for future use in R

```
save(mydata, file = "data/mydata.RData")  # saving mydata within the data folder
```

You can load .RData files using the load() command:

```
load("data/mydata.RData")
```

- Save **csv** file: comma-separated values

```
write.csv(mydata, file = "data/mydata.csv", col.names = TRUE, row.names = FALSE)
```

More commands to filter rows

Remove rows with missing data

`drop_na()` (or `na.omit()`) removes *all* rows with *any* missing (`NA`) values in *any* column

```
mydata %>% drop_na()
```

```
## # A tibble: 7 x 10
##   record age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##    <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
## 1 1.31e6 17 y… Male  12th  Hisp…  19.5      56.2 1 or 2 days      No
## 2 1.31e6 17 y… Male  11th  Hisp…  20.6      61.7 1 or 2 days      No
## 3 1.31e6 15 y… Fema… 10th  Hisp…  27.5      70.3 0 days           No
## 4 1.31e6 16 y… Fema… 11th  Hisp…  26.6      68.0 0 days           No
## 5 1.31e6 16 y… Fema… 11th  White  24.8      63.5 3 to 5 days      No
## 6 1.31e6 16 y… Fema… 10th  All …  25.0      76.7 0 days           No
## 7 1.32e6 18 y… Fema… 12th  Blac…  27.5      74.8 All 30 days      Yes
## # … with 1 more variable: bullied_past_12mo <lgl>
```

For more on dealing with missing data, see Tidyverse part 2 workshop

Remove rows with duplicated data

`distinct()` removes rows that are duplicates of other rows

```
data_dups <- tibble(
  name = c("Ana","Bob","Cara", "Ana"),
  race = c("Hispanic","Other", "White", "Hispanic")
)
```

```
data_dups
```

```
## # A tibble: 4 x 2
##   name  race
##   <chr> <chr>
## 1 Ana   Hispanic
## 2 Bob   Other
## 3 Cara  White
## 4 Ana   Hispanic
```

```
data_dups %>% distinct()
```

```
## # A tibble: 3 x 2
##   name  race
##   <chr> <chr>
## 1 Ana   Hispanic
## 2 Bob   Other
## 3 Cara  White
```

Order rows: `arrange()`

Use `arrange()` to order the rows by the values in specified columns

```
mydata %>% arrange(bmi, weight_kg) %>% head(n=3)
```

```
## # A tibble: 3 x 10
##   record age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##    <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
## 1 9.35e5 17 y… Fema… 12th  All …  17.5      43.1 <NA>             No
## 2 9.22e5 14 y… Male  9th   White  18.2      57.6 <NA>             Yes
## 3 1.31e6 17 y… Male  12th  Hisp…  19.5      56.2 1 or 2 days      No
## # … with 1 more variable: bullied_past_12mo <lgl>
```

```
mydata %>% arrange(desc(bmi), weight_kg) %>% head(n=3)
```

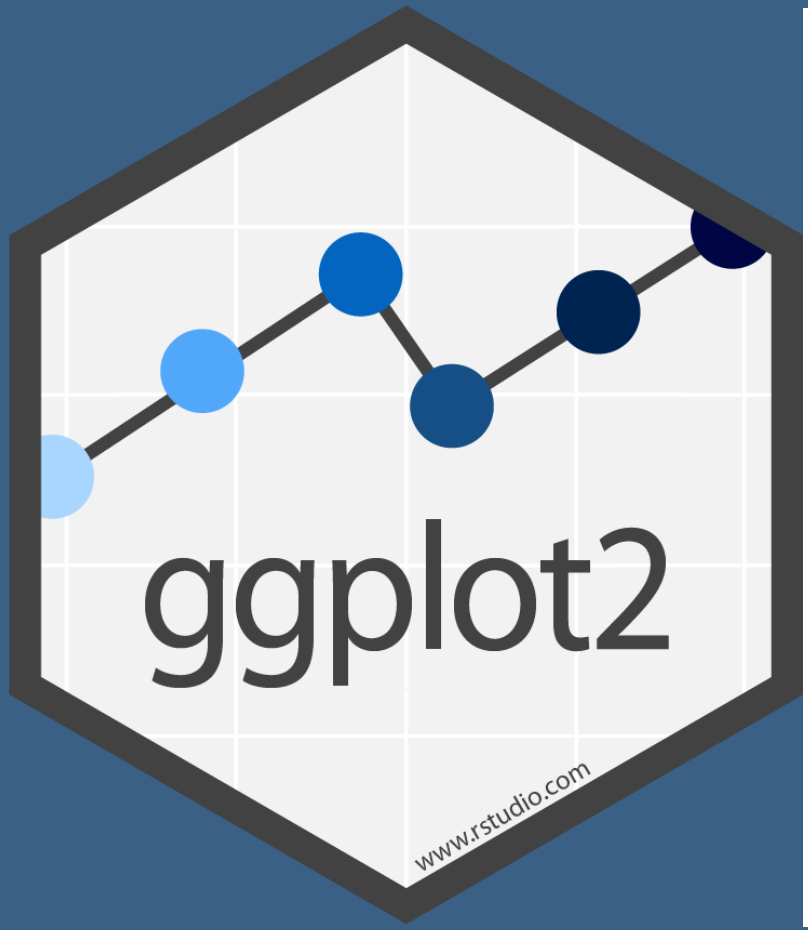
```
## # A tibble: 3 x 10
##   record age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##    <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
## 1 6.39e5 16 y… Fema… 9th   <NA>   29.3      84.8 <NA>             Yes
## 2 3.35e5 17 y… Fema… 10th  White  27.6      66.2 <NA>             <NA>
## 3 1.32e6 18 y… Fema… 12th  Blac…  27.5      74.8 All 30 days      Yes
```

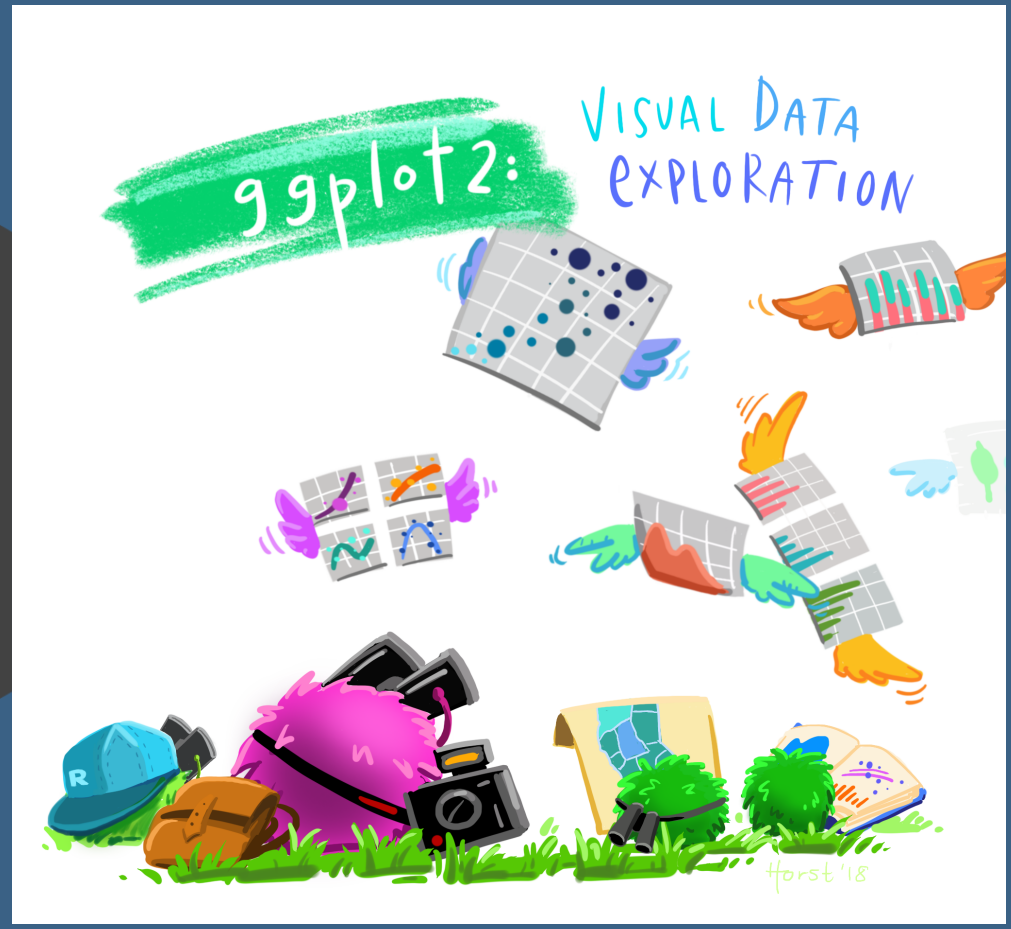
Combine `summarize()` with other tidy functions

```
mydata %>%
  group_by(grade) %>%
  summarize(n_per_group = n(),
            bmi_mean =mean(bmi),
            bmi_sd = sd(bmi, na.rm = TRUE)) %>%
  mutate(bmi_cv = bmi_sd/bmi_mean) %>%
  filter(bmi_cv > .2)
```

```
## # A tibble: 2 x 5
##   grade n_per_group bmi_mean bmi_sd bmi_cv
##   <chr>       <int>    <dbl>  <dbl>  <dbl>
## 1 12th            4     21.0   4.44  0.212
## 2 9th             4     23.9   5.03  0.211
```

Making prettier plots





Allison Horst

Basics of ggplot

- For a full treatment, watch our BERD workshop "Data Visualization with R and ggplot2"

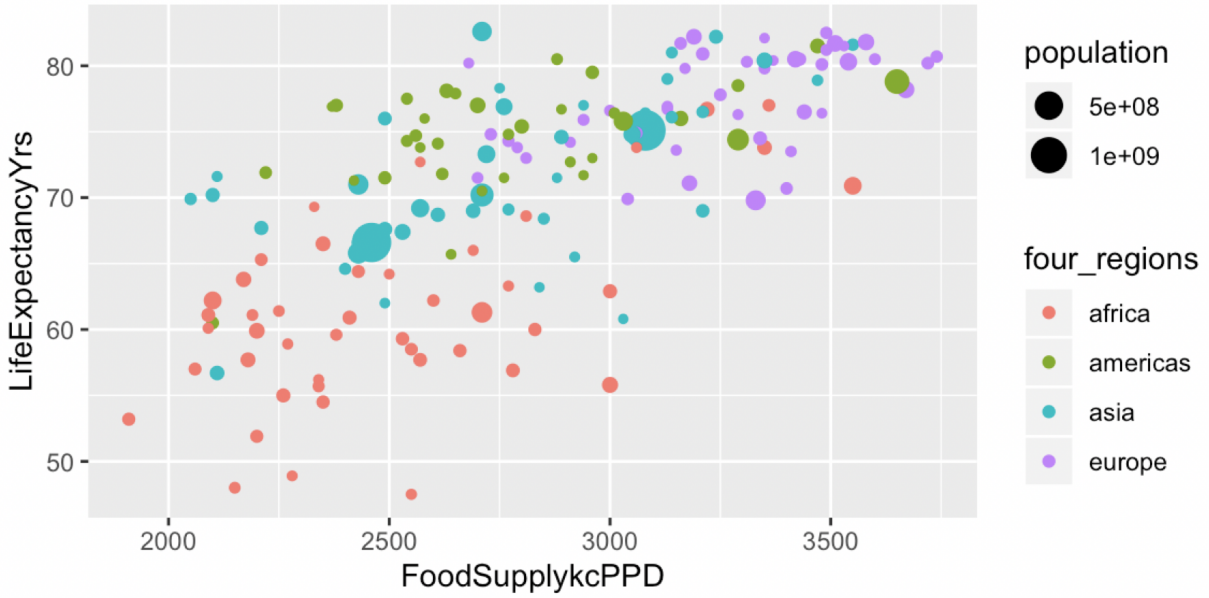
Function

Dataset

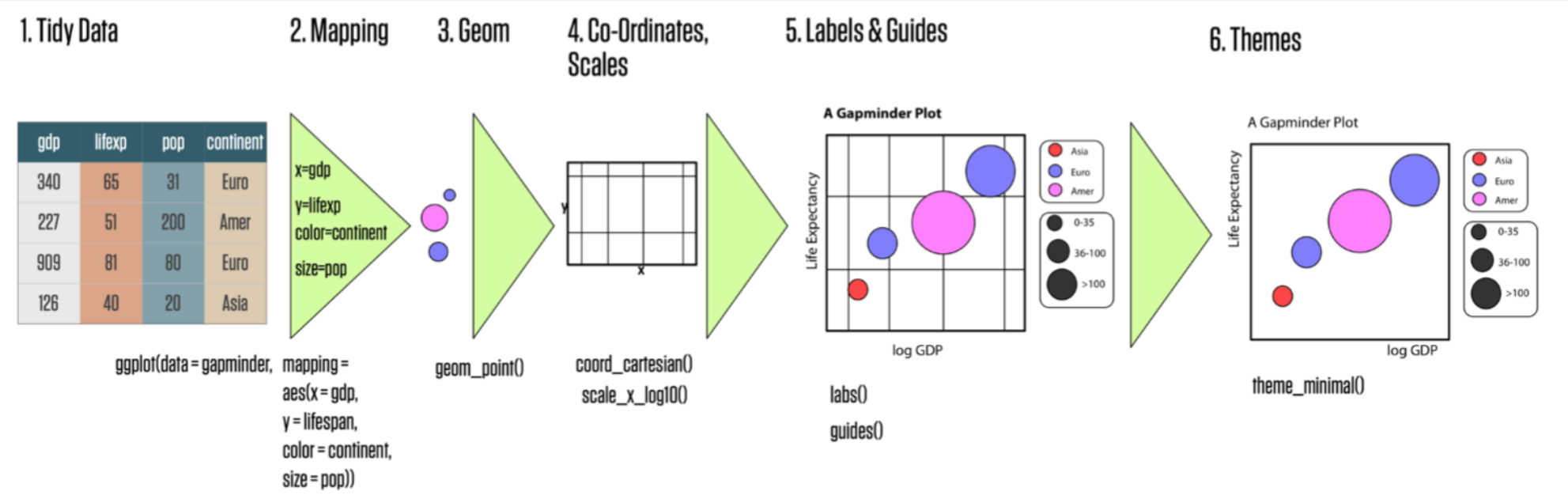
```
ggplot(data = gapminder2011,
       aes(x = FoodSupplykcPPD, y = LifeExpectancyYrs,
           color = four_regions, size = population)) +
       geom_point()
```

Which variables to plot

What kind of plot to make



Grammar of ggplot2



Kieran Healy

Back to ggplot code

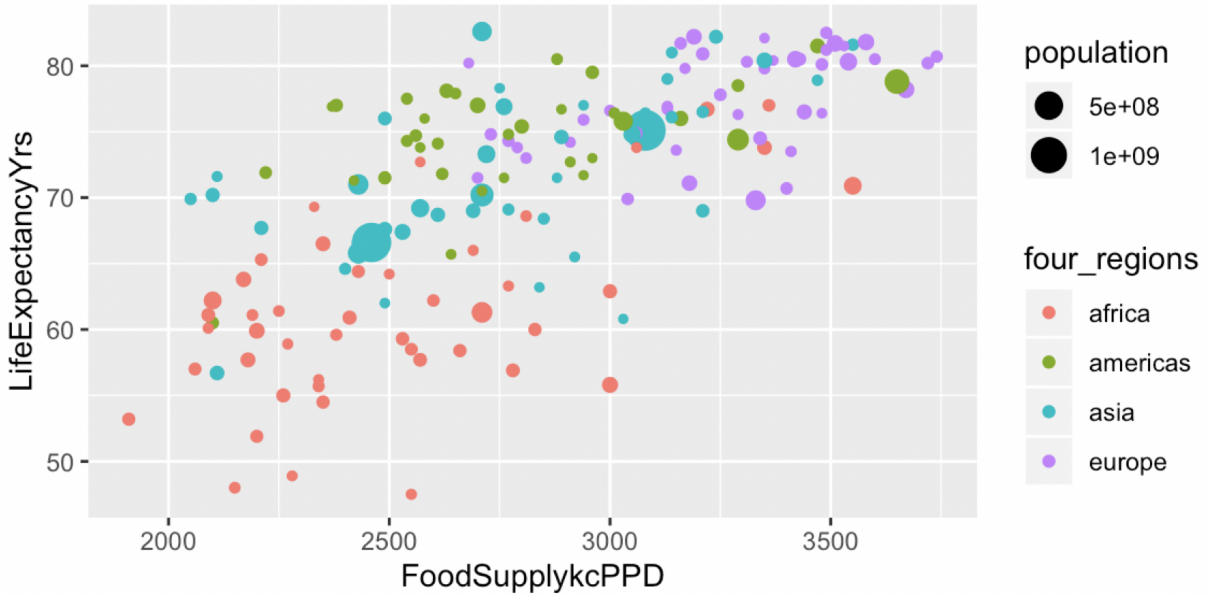
Function

Dataset

```
ggplot(data = gapminder2011,
       aes(x = FoodSupplykcPPD, y = LifeExpectancyYrs,
           color = four_regions, size = population)) +
       geom_point()
```

Which variables to plot

What kind of plot to make



Ggplot needs tidy data

What are **tidy** data?

1. Each variable forms a column
2. Each observation forms a row
3. Each value has its own cell

G. Grolemond & H. Wickham's R for Data Science

See BERD workshop Data Wrangling Part 1 slides for more info.

Is our data tidy?

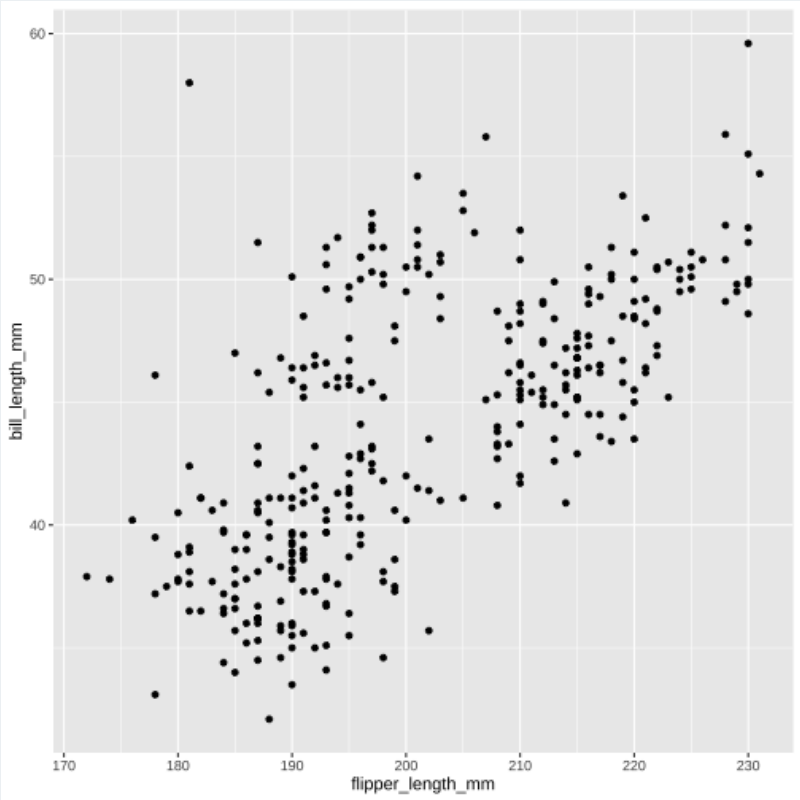
practice.Rmd × penguins ×

Filter

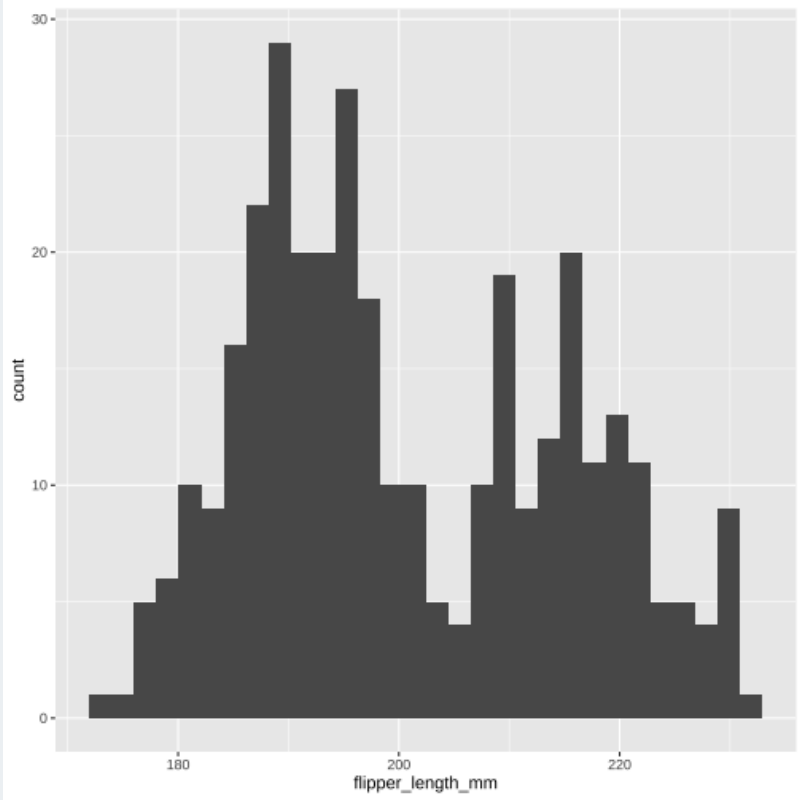
	id	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
1	1689	Adelie	Torgersen	39.1	18.7	181	3750	male	2007
2	4274	Adelie	Torgersen	*NA*	17.4	186	3800	female	2007
3	4539	Adelie	Torgersen	40.3	18.0	195	3250	female	2007
4	2435	Adelie	Torgersen	36.7	19.3	193	3450	female	2007
5	2326	Adelie	Torgersen	39.3	20.6	190	3650	male	2007
6	2637	Adelie	Torgersen	38.9	17.8	181	3625	female	2007
7	4443	Adelie	Torgersen	*NA*	19.6	195	4675	male	2007
8	2102	Adelie	Torgersen	34.1	18.1	193	3475	*NA*	2007
9	2975	Adelie	Torgersen	42.0	20.2	190	4250	*NA*	2007
10	3966	Adelie	Torgersen	37.8	17.1	186	3300	*NA*	2007
11	4648	Adelie	Torgersen	37.8	17.3	180	3700	*NA*	2007
12	2887	Adelie	Torgersen	41.1	17.6	182	3200	female	2007
13	4939	Adelie	Torgersen	38.6	21.2	191	3800	male	2007
14	2849	Adelie	Torgersen	34.6	21.1	198	4400	male	2007
15	4743	Adelie	Torgersen	36.6	17.8	185	3700	female	2007
16	4573	Adelie	Torgersen	38.7	19.0	195	3450	female	2007
17	1669	Adelie	Torgersen	42.5	20.7	197	4500	male	2007

Simple plots

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




```
ggplot(data = penguins,
       aes(x = flipper_length_mm)) +
  geom_histogram()
```

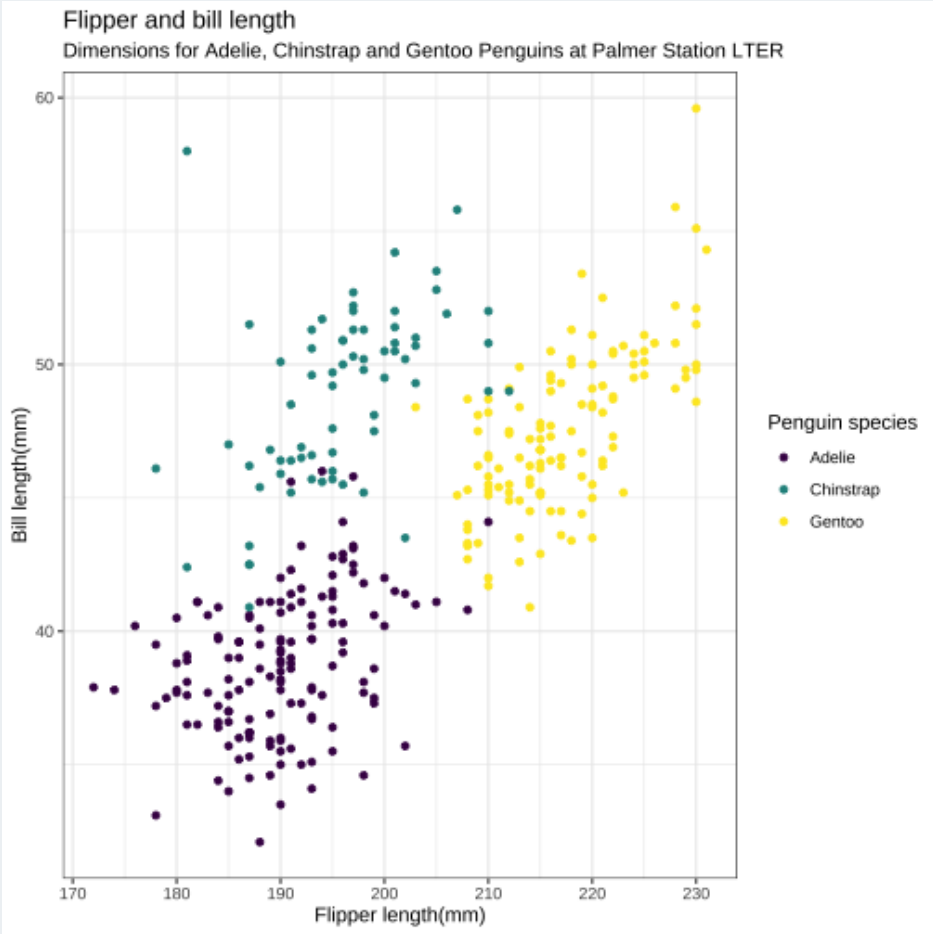


Improved plots - tips

- **Start with simple**, slowly add in additions/colors/etc
- You are building a plot layer by layer! ++++++
- At the beginning, **just copy and paste examples** that you want to edit until you understand what each function does
- It will take some trial and error!
- **Watch BERD ggplot video** for more instruction, and many customizations

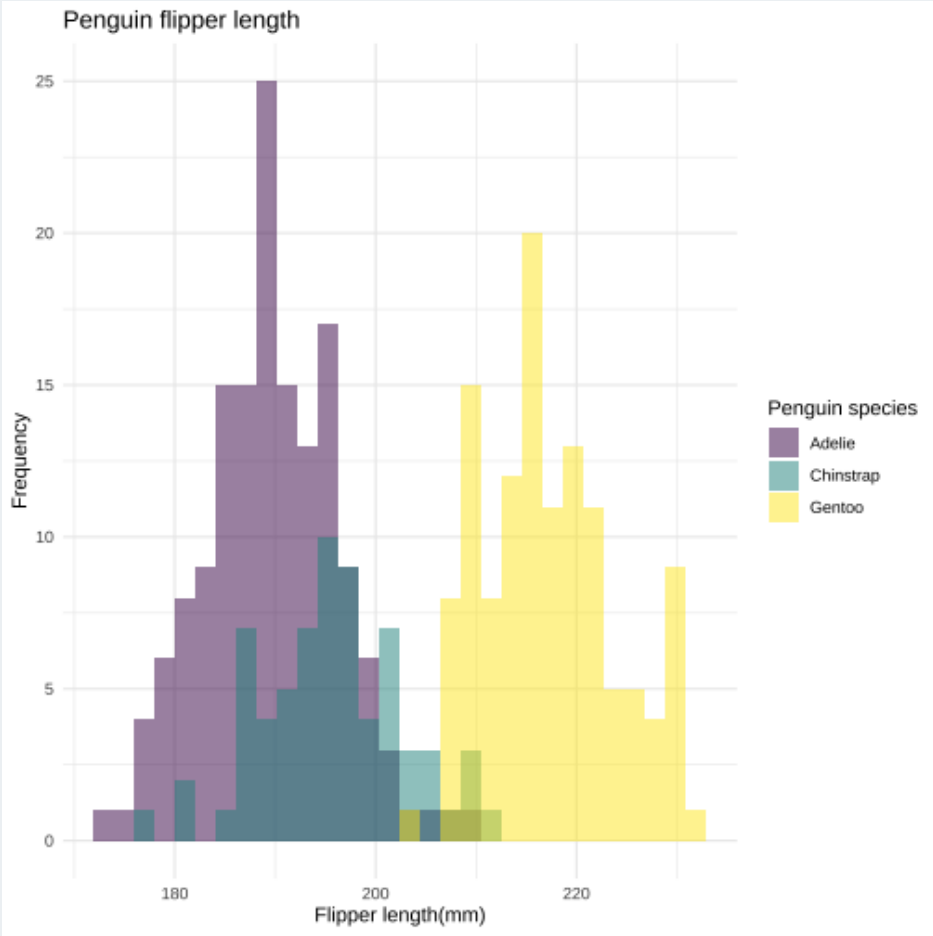
Improved scatterplot

```
ggplot(data = penguins,
       aes(x = flipper_length_mm,
           y = bill_length_mm,
           color = species)) +
  geom_point()+
  labs(
    title = "Flipper and bill length",
    subtitle = "Dimensions for Adelie,
    x = "Flipper length(mm)",
    y = "Bill length(mm)") +
  scale_color_viridis_d(name = "Pengui
  theme_bw()
```



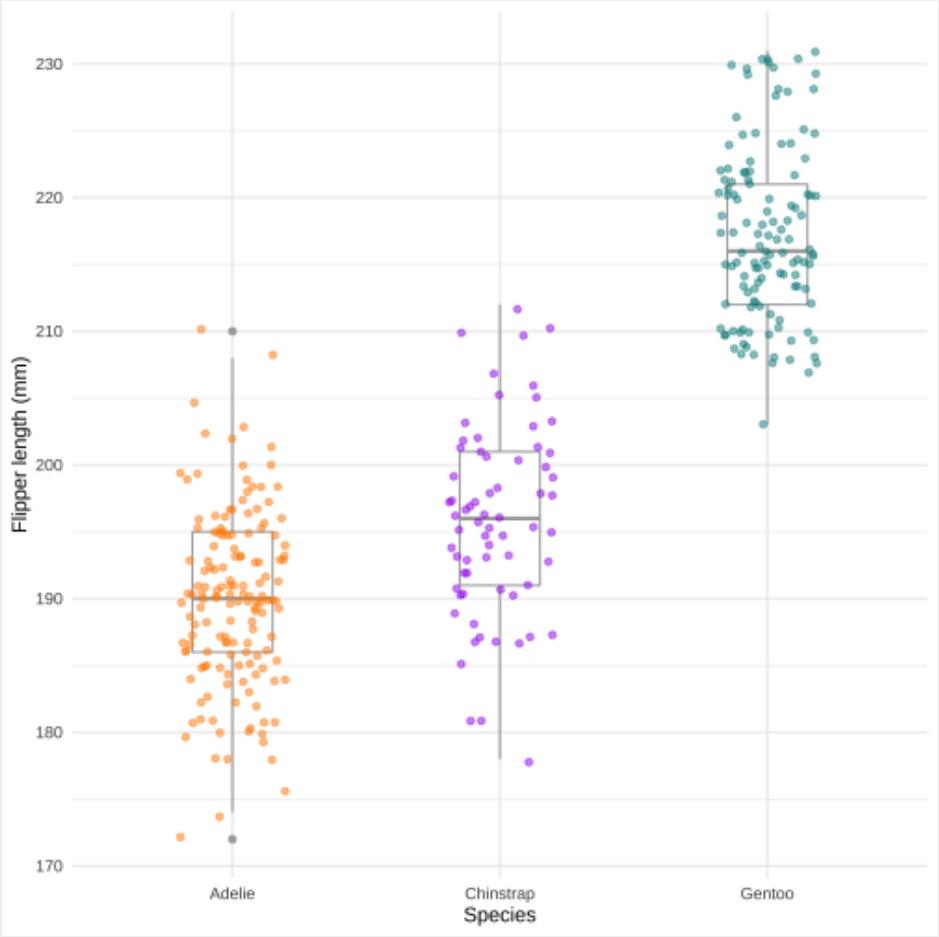
Improved histogram

```
ggplot(data = penguins,
       aes(x = flipper_length_mm,
           fill = species)) +
  geom_histogram(
    alpha = 0.5,
    position = "identity") +
  labs(title = "Penguin flipper length
       x = "Flipper length(mm)",
       y = "Frequency") +
  scale_fill_viridis_d(
    name = "Penguin species") +
  theme_minimal()
```



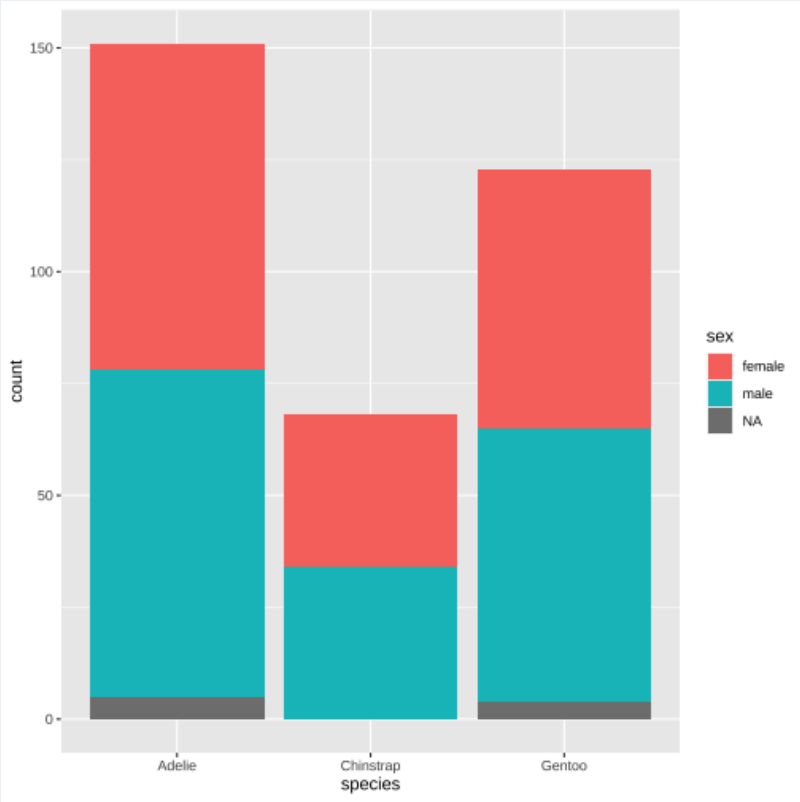
Boxplot + jitter

```
ggplot(data = penguins,
       aes(x = species,
           y = flipper_length_mm)) +
  geom_boxplot(color="darkgrey",
               width = 0.3,
               show.legend = FALSE) +
  geom_jitter(aes(color = species),
              alpha = 0.5,
              show.legend = FALSE,
              position = position_jitt
  scale_color_manual(values = c("darko
  theme_minimal() +
  labs(x = "Species",
       y = "Flipper length (mm)")
```

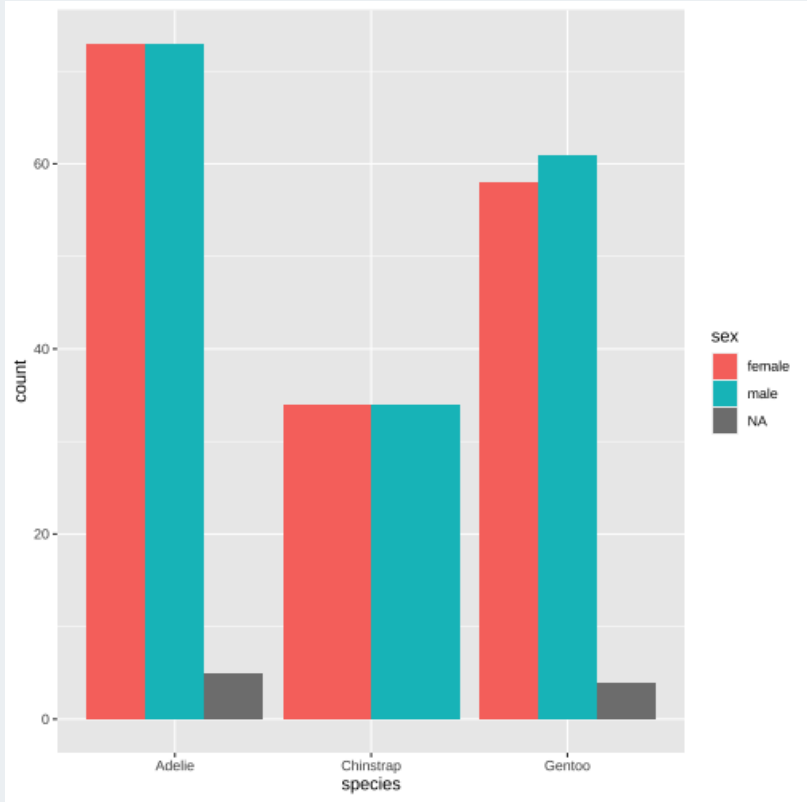


Bar plots - counts

```
ggplot(data = penguins,
       aes(x = species,
           fill = sex)) +
  geom_bar()
```




```
ggplot(data = penguins,
       aes(x = species,
           fill = sex)) +
  geom_bar(position = "dodge")
```

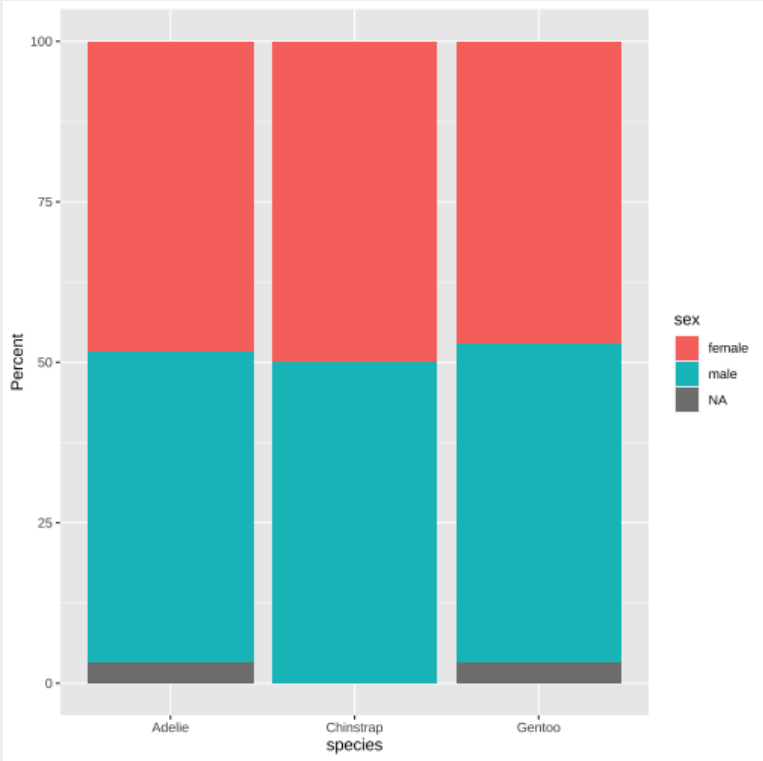


Bar plots - percentages

```
pct_data <- penguins %>%
  count(species, sex) %>%
  group_by(species) %>%
  mutate(pct = 100*n/sum(n))
pct_data
```

```
## # A tibble: 8 x 4
## # Groups:   species [3]
##   species   sex        n   pct
##   <chr>     <chr>  <int> <dbl>
## 1 Adelie    female    73 48.3
## 2 Adelie    male      73 48.3
## 3 Adelie    <NA>       5  3.31
## 4 Chinstrap female    34 50
## 5 Chinstrap male      34 50
## 6 Gentoo    female    58 47.2
## 7 Gentoo    male      61 49.6
## 8 Gentoo    <NA>       4  3.25
```

```
ggplot(data = pct_data,
       aes(x = species, y = pct,
           fill = sex)) +
  geom_col()+
  ylab("Percent")
```

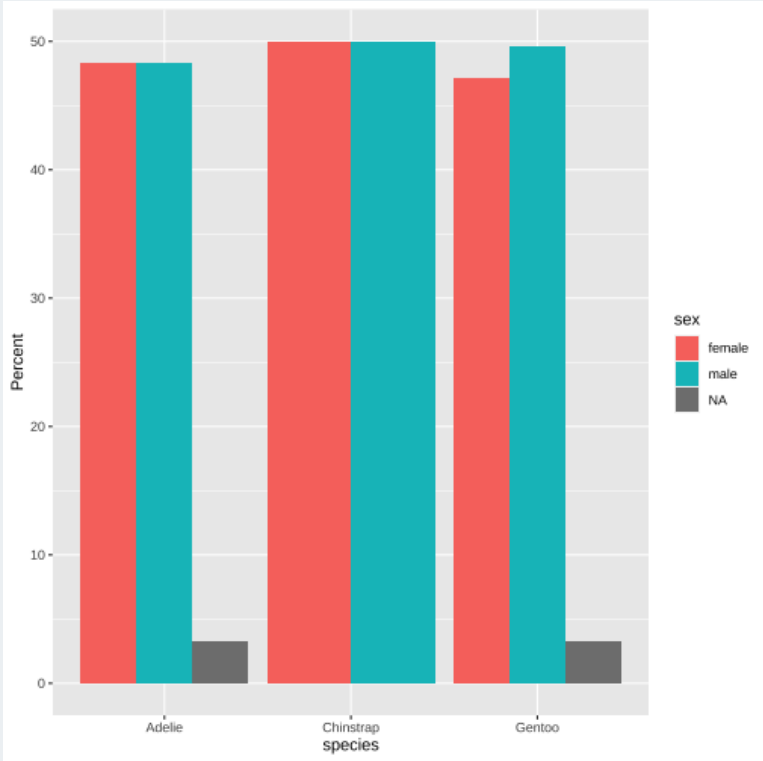


Bar plots - percentages

```
pct_data <- penguins %>%
  count(species, sex) %>%
  group_by(species) %>%
  mutate(pct = 100*n/sum(n))
pct_data
```

```
## # A tibble: 8 x 4
## # Groups:   species [3]
##   species   sex        n   pct
##   <chr>     <chr>  <int> <dbl>
## 1 Adelie    female    73 48.3
## 2 Adelie    male      73 48.3
## 3 Adelie    <NA>       5  3.31
## 4 Chinstrap female    34 50
## 5 Chinstrap male      34 50
## 6 Gentoo    female    58 47.2
## 7 Gentoo    male      61 49.6
## 8 Gentoo    <NA>       4  3.25
```

```
ggplot(data = pct_data,
       aes(x = species, y = pct,
           fill = sex)) +
  geom_col(position = "dodge") +
  ylab("Percent")
```



Factor data types

class: inverse, center, middle

The more you know

Installing and using packages

- Packages are to R like apps are to your phone/OS
- Packages contain additional functions and data
- Install packages with `install.packages()`
  - Also can use the "Packages" tab in Files/Plots/Packages/Help/Viewer window
  - *Only install once (unless you want to update)*
  - Installs from Comprehensive R Archive Network (CRAN) = package mothership

```
install.packages("dplyr")   # only do this ONCE, use quotes
```

- Load packages: At the top of your script include `library()` commands to load each required package *every* time you open Rstudio.

```
library(dplyr)    # run this every time you open Rstudio
```

- Use a function without loading the package with `::`

```
dplyr::arrange(mydata, bmi)
```

Installing packages from other places (i.e. github, URLs)

- Need to have remotes package installed first:

```
install.packages("remotes")
```

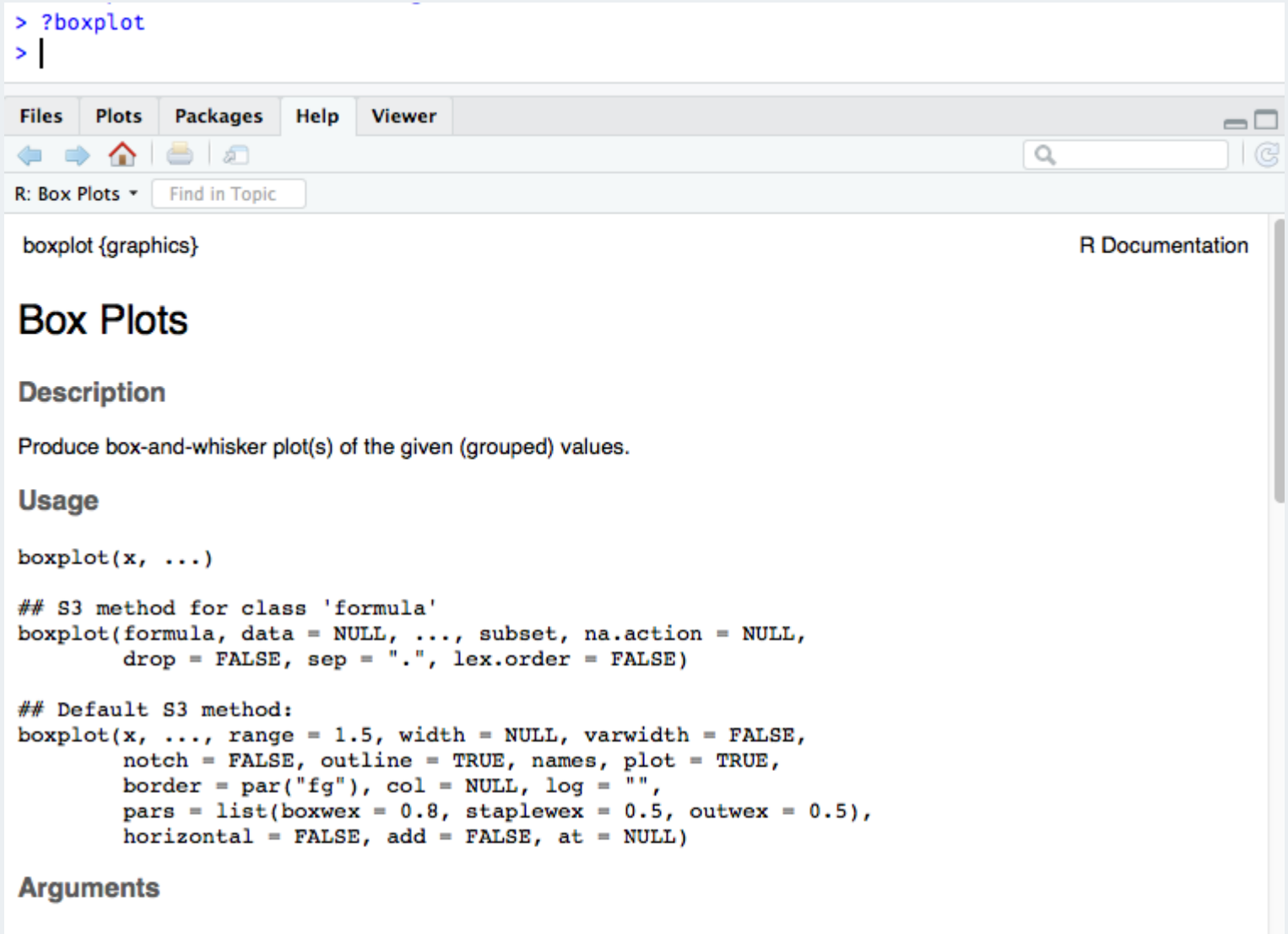
- To install a package from github (often in development) use `install_github()` from the remotes package

```
# https://github.com/hadley/yrbss
remotes::install_github("hadley/yrbss")

# Load it the same way
library(yrbss)
```

How to get help (1/2)

Use ? in front of function name in console. Try this:



How to get help (2/2)

- Use `??` (i.e `??dplyr` or `??read_csv`) for searching all documentation in installed packages (including unloaded packages)
- search Stack Overflow #r tag
- googlequestion + rcran or + r (i.e. "make a boxplot rcran" "make a boxplot r")
- google error in quotes (i.e. `"Evaluation error: invalid type (closure) for variable '***'"`)
- search github for your function name (to see examples) or error
- Rstudio community
- twitter #rstats

Resources

- Click on this List of resources for learning R
- Watch recordings of our other workshops
- **Highly recommend** *Data Wrangling in R with Tidyverse*

Getting started:

- RStudio IDE Cheatsheet
- Install R/RStudio help video
- Basic Basics

Some of this is drawn from materials in online books/lessons:

- Intro to R/RStudio by Emma Rand
- Modern Dive - An Introduction to Statistical and Data Sciences via R by Chester Ismay & Albert Kim
- Cookbook for R by Winston Chang

Local resources

- OHSU's BioData club + active slack channel
- Portland's R user meetup group + active slack channel
- R-ladies PDX meetup group
- Cascadia R Conf - May 31, 2020 in Eugene with workshops

Allison Horst

Contact info:

Jessica Minnier: *minnier@ohsu.edu*

Meike Niederhausen: *niederha@ohsu.edu*

This workshop info:

- Code for these slides on github: jminnier/berd_r_courses
- all the R code in an R script
- answers to practice problems can be found here: html, pdf
- The project folder of examples can be downloaded at github.com/jminnier/berd_intro_project & the solutions are in the `solns/` folder.