

Introduction to Coding in R

EKB

2024-09-19

2-Dimensional Data in R

Student Learning Outcomes

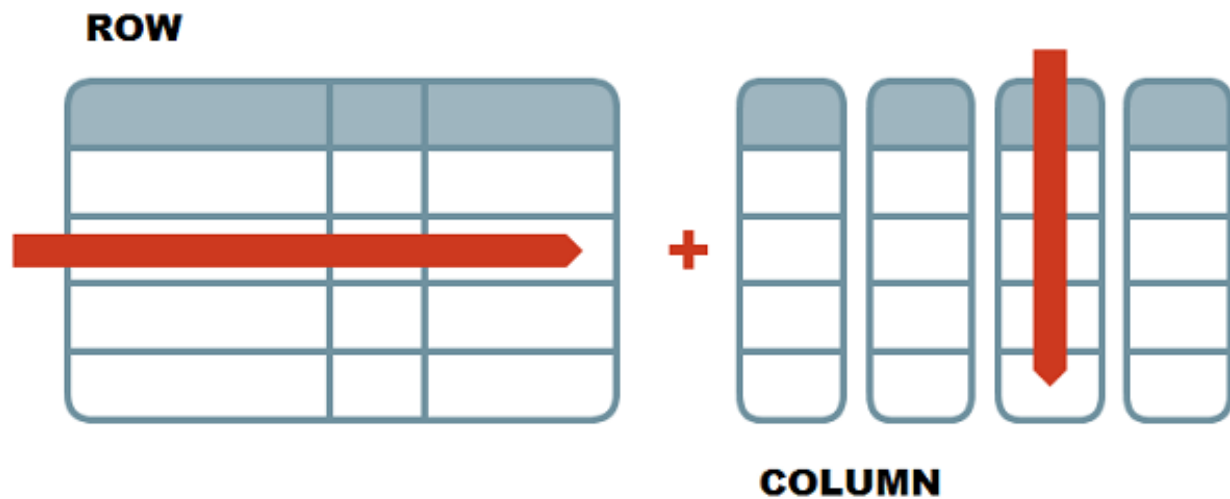
- Students will be able to describe the relationship between vectors and data frames
- Students will be able to do the following in the R language:
 - explore and describe data frames
 - filter specific values from data frames
 - calculate descriptive statistics from data frames
 - make histograms

Working with Data Frames

Most of the data we work with is two-dimensional, i.e., it has columns and rows. Its structure resembles a spreadsheet. Because a single data point needs to be referenced by two positions (which row and which column), we call it 2D.

As a friendly reminder:

- **rows** go side-to-side
- **columns** go up-and-down

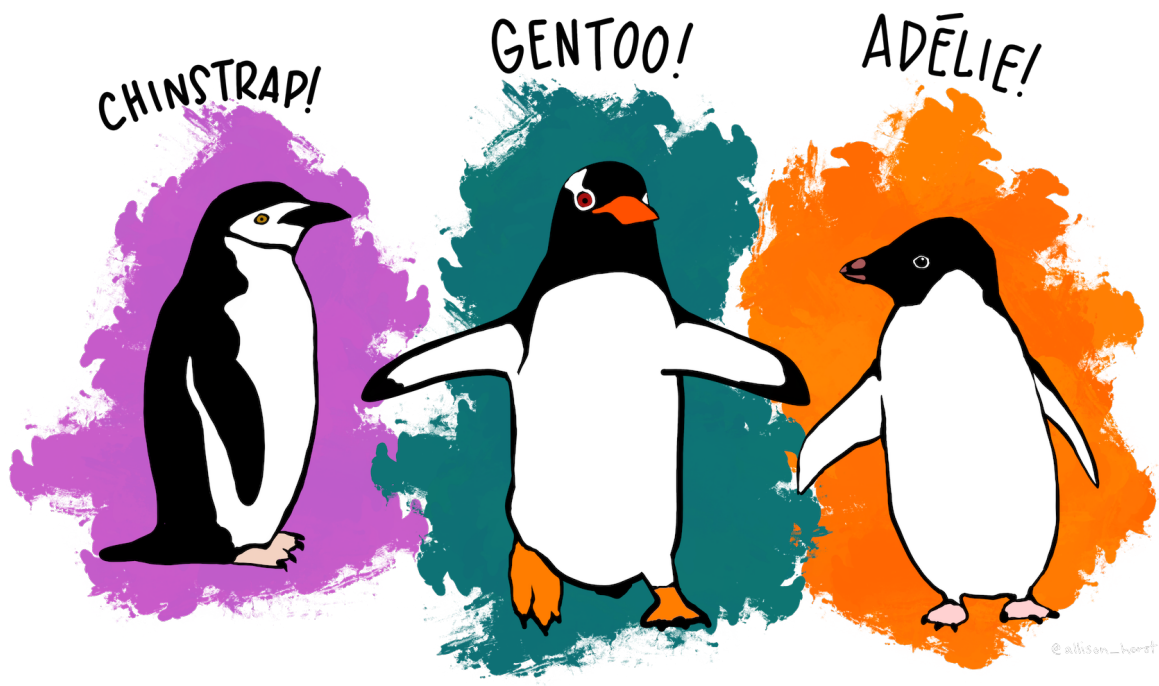


R is really good at working with these types of data. We call them *data frames*.

Data frames are made up of multiple vectors. Each vector becomes a column in a data frame.

To explore data frames, we are going to use a package called `palmerpenguins`.

A *package* is a bunch of pre-written code, often in the form of functions, which we can bring into R and use. In this case, we are using a data package, which loads data into R that we can use. It is real data from penguins in Antarctica! You can learn more about the `palmerpenguins` package and data [here](#).



Installing a Package

The first time that you want to use a certain package, you need to “install” the package, meaning download the contents of the package from the internet into your work space.

I have already installed the `palmerpenguins` package into this Posit Cloud project, so you do not need to install it. I’ve included the code here for future reference, if you need it.

```
# code for installing a package from the internet ONLY FOR FUTURE REFERENCE
# install.packages("palmerpenguins")
# to run the line of code above, remove the # symbol
```

Loading a Package

Although we have installed the package, we aren’t ready to use it yet. Every time (for us, every new project) we want to use something from a package, we need to tell RStudio that we want to use it. We will need to do that every time we open Posit Cloud.

We do this through a function called `library()`.

```
library(palmerpenguins)
```

Exploring the Penguin Data

Let's take a look at our data. The data we are using is in a data frame called `penguins`.

```
penguins
```

```
## # A tibble: 344 x 8
##   species island   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>         <dbl>         <dbl>         <int>         <int>
## 1 Adelie  Torgersen         39.1          18.7          181          3750
## 2 Adelie  Torgersen         39.5          17.4          186          3800
## 3 Adelie  Torgersen         40.3           18          195          3250
## 4 Adelie  Torgersen          NA           NA           NA           NA
## 5 Adelie  Torgersen         36.7          19.3          193          3450
## 6 Adelie  Torgersen         39.3          20.6          190          3650
## 7 Adelie  Torgersen         38.9          17.8          181          3625
## 8 Adelie  Torgersen         39.2          19.6          195          4675
## 9 Adelie  Torgersen         34.1          18.1          193          3475
## 10 Adelie Torgersen         42           20.2          190          4250
## # i 334 more rows
## # i 2 more variables: sex <fct>, year <int>
```

A quirk about using data that we've loaded in through a package instead of directly reading in data from a .csv file is that the data frame will not show up in our environment unless we specifically tell it to.

```
penguins <- penguins
```

Functions

As with vectors, there are many functions that are useful for taking a look at data frames. Many of the ones that work with vectors also work with data frames. Here are a few of the ones I find very helpful.

```
head(penguins) # first 6 lines
```

```
## # A tibble: 6 x 8
##   species island   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>         <dbl>         <dbl>         <int>         <int>
## 1 Adelie  Torgersen         39.1          18.7          181          3750
## 2 Adelie  Torgersen         39.5          17.4          186          3800
## 3 Adelie  Torgersen         40.3           18          195          3250
## 4 Adelie  Torgersen          NA           NA           NA           NA
## 5 Adelie  Torgersen         36.7          19.3          193          3450
## 6 Adelie  Torgersen         39.3          20.6          190          3650
## # i 2 more variables: sex <fct>, year <int>
```

```
head(penguins, 10) # can specify how many lines
```

```
## # A tibble: 10 x 8
##   species island   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>         <dbl>         <dbl>         <int>         <int>
## 1 Adelie  Torgersen         39.1          18.7          181          3750
## 2 Adelie  Torgersen         39.5          17.4          186          3800
## 3 Adelie  Torgersen         40.3           18          195          3250
## 4 Adelie  Torgersen          NA           NA           NA           NA
## 5 Adelie  Torgersen         36.7          19.3          193          3450
## 6 Adelie  Torgersen         39.3          20.6          190          3650
## 7 Adelie  Torgersen         38.9          17.8          181          3625
## 8 Adelie  Torgersen         39.2          19.6          195          4675
```

```
## 9 Adelie Torgersen      34.1      18.1      193      3475
## 10 Adelie Torgersen     42      20.2      190      4250
## # i 2 more variables: sex <fct>, year <int>
```

```
tail(penguins) # last 6 lines
```

```
## # A tibble: 6 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>    <fct>      <dbl>      <dbl>          <int>      <int>
## 1 Chinstrap Dream      45.7        17            195      3650
## 2 Chinstrap Dream      55.8        19.8           207      4000
## 3 Chinstrap Dream      43.5        18.1           202      3400
## 4 Chinstrap Dream      49.6        18.2           193      3775
## 5 Chinstrap Dream      50.8         19            210      4100
## 6 Chinstrap Dream      50.2        18.7           198      3775
## # i 2 more variables: sex <fct>, year <int>
```

```
str(penguins) # structure
```

```
## tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
## $ species      : Factor w/ 3 levels "Adelie","Chinstrap",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ island       : Factor w/ 3 levels "Biscoe","Dream",...: 3 3 3 3 3 3 3 3 3 3 ...
## $ bill_length_mm : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
## $ bill_depth_mm : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
## $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
## $ body_mass_g    : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
## $ sex           : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ...
## $ year          : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

```
nrow(penguins) # number of rows
```

```
## [1] 344
```

```
ncol(penguins) # number of columns
```

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

```
names(penguins) # same as colnames(penguins) in a df
```

```
## [1] "species"      "island"        "bill_length_mm"
## [4] "bill_depth_mm" "flipper_length_mm" "body_mass_g"
## [7] "sex"          "year"
```

Sub-setting using Indexing

When sub-setting data frames, we need to now specify 2 locations, the row and the column. In R, it is always row *then* column. Note that this is typically the opposite of spreadsheets.

```
# in vectors, only 1 dimension, so we only need to specify one location
# data frames are 2-dimensional, so we have to specify 2 different locations
```

```
penguins[1:10, c(2,3)]
```

```
## # A tibble: 10 x 2
##   island bill_length_mm
##   <fct>      <dbl>
## 1 Torgersen    39.1
## 2 Torgersen    39.5
## 3 Torgersen    40.3
```

```
## 4 Torgersen      NA
## 5 Torgersen      36.7
## 6 Torgersen      39.3
## 7 Torgersen      38.9
## 8 Torgersen      39.2
## 9 Torgersen      34.1
## 10 Torgersen     42
```

```
penguins[1:10, ]
```

```
## # A tibble: 10 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>         <dbl>         <dbl>         <int>         <int>
## 1 Adelie Torgersen      39.1           18.7           181          3750
## 2 Adelie Torgersen      39.5           17.4           186          3800
## 3 Adelie Torgersen      40.3            18           195          3250
## 4 Adelie Torgersen      NA            NA            NA            NA
## 5 Adelie Torgersen      36.7           19.3           193          3450
## 6 Adelie Torgersen      39.3           20.6           190          3650
## 7 Adelie Torgersen      38.9           17.8           181          3625
## 8 Adelie Torgersen      39.2           19.6           195          4675
## 9 Adelie Torgersen      34.1           18.1           193          3475
## 10 Adelie Torgersen      42            20.2           190          4250
## # i 2 more variables: sex <fct>, year <int>
```

```
penguins[ , c(1:4)]
```

```
## # A tibble: 344 x 4
##   species island bill_length_mm bill_depth_mm
##   <fct>   <fct>         <dbl>         <dbl>
## 1 Adelie Torgersen      39.1           18.7
## 2 Adelie Torgersen      39.5           17.4
## 3 Adelie Torgersen      40.3            18
## 4 Adelie Torgersen      NA            NA
## 5 Adelie Torgersen      36.7           19.3
## 6 Adelie Torgersen      39.3           20.6
## 7 Adelie Torgersen      38.9           17.8
## 8 Adelie Torgersen      39.2           19.6
## 9 Adelie Torgersen      34.1           18.1
## 10 Adelie Torgersen      42            20.2
## # i 334 more rows
```

Select Individual Columns

Often, we want to select a specific column to perform calculations on or to plot. We can do this via sub-setting, though the result is a data frame with 1 column, not a vector.

To pull out one column to treat as a vector, we can use the `$` operator.

```
# with subsetting by index
```

```
penguins[ ,1] # requires position and creates a df with 1 column
```

```
## # A tibble: 344 x 1
##   species
##   <fct>
## 1 Adelie
## 2 Adelie
```

```
## 3 Adelie
## 4 Adelie
## 5 Adelie
## 6 Adelie
## 7 Adelie
## 8 Adelie
## 9 Adelie
## 10 Adelie
## # i 334 more rows
```

```
# subsetting with $
penguins$species # pulling out 1 column by name, as a vector
```

```
## [1] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [8] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [15] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [22] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [29] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [36] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [43] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [50] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [57] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [64] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [71] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [78] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [85] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [92] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [99] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [106] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [113] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [120] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [127] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [134] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [141] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [148] Adelie Adelie Adelie Adelie Adelie Adelie Adelie
## [155] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [162] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [169] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [176] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [183] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [190] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [197] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [204] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [211] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [218] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [225] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [232] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [239] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [246] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [253] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [260] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [267] Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo Gentoo
## [274] Gentoo Gentoo Gentoo Chinstrap Chinstrap Chinstrap Chinstrap
## [281] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [288] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
```

```
## [295] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [302] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [309] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [316] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [323] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [330] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [337] Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap Chinstrap
## [344] Chinstrap
## Levels: Adelie Chinstrap Gentoo
```

```
unique(penguins$species) # we can then place the vector inside of a function
```

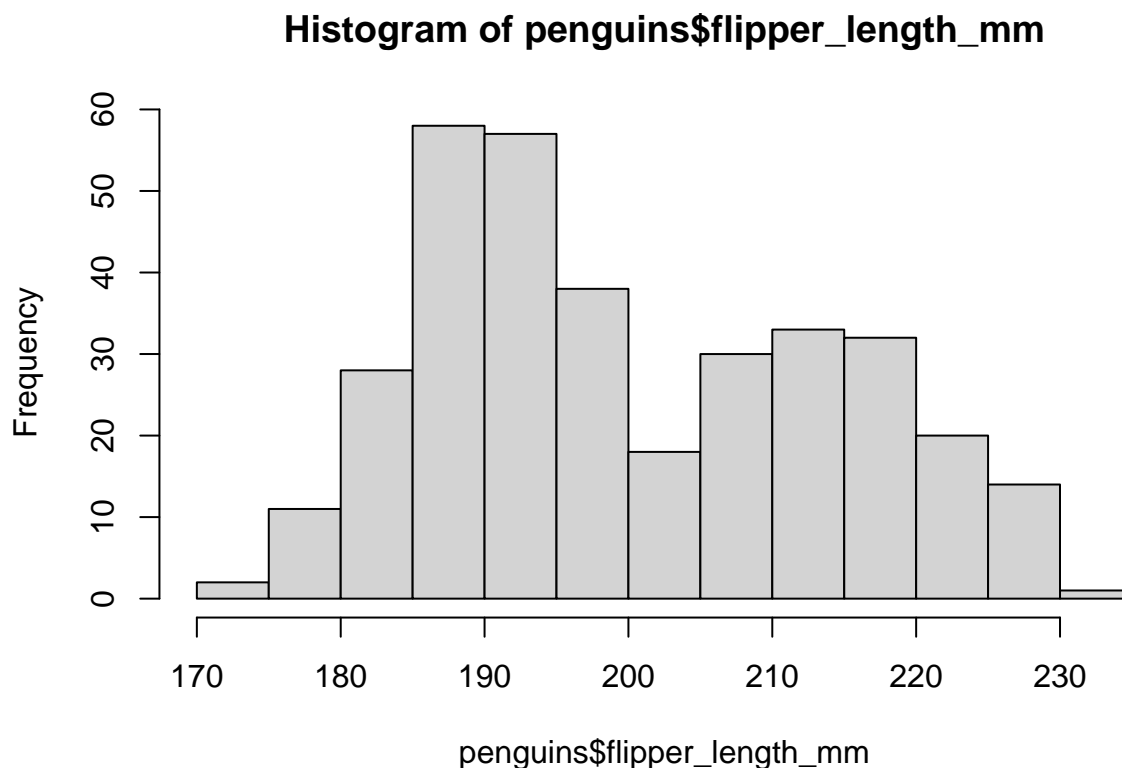
```
## [1] Adelie    Gentoo    Chinstrap
## Levels: Adelie Chinstrap Gentoo
```

```
# we can save single columns as vectors with the assignment operator
flipper_lenght_mm <- penguins$flipper_length_mm
```

Plot a Histogram

Let's plot a histogram with the flipper length data.

```
# Plot a histogram
hist(penguins$flipper_length_mm) # same as hist(flipper_length_mm)
```



We can also perform calculations on these vectors.

```
mean(penguins$flipper_length_mm)
```

```
## [1] NA
sd(penguins$flipper_lenght_mm)

## Warning: Unknown or uninitialised column: `flipper_lenght_mm`.
## [1] NA
# min, max, median, mode are other functions we might want to use
```

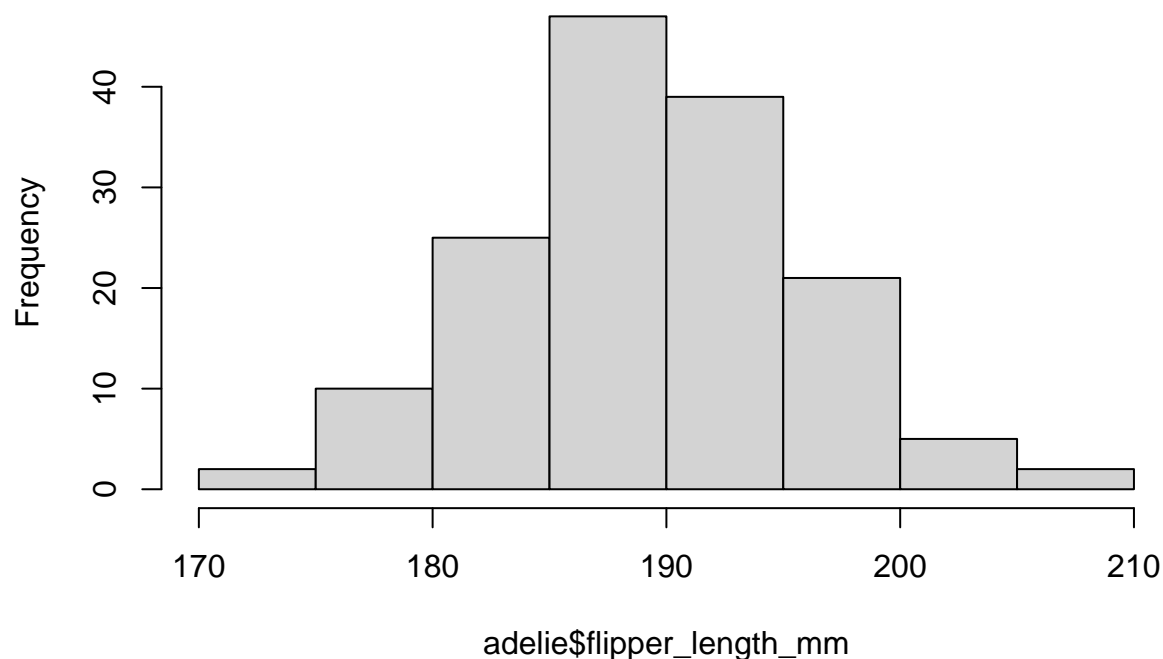
Conditional Subsetting

As with vectors, we can use conditional formatting to select specific observations (typically rows).

```
# create a new data frame with only adelic penguins
adelie <- penguins[penguins$species == 'Adelie', ]
adelie

## # A tibble: 152 x 8
##   species island   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>         <dbl>         <dbl>         <int>         <int>
## 1 Adelie  Torgersen         39.1          18.7          181          3750
## 2 Adelie  Torgersen         39.5          17.4          186          3800
## 3 Adelie  Torgersen         40.3           18          195          3250
## 4 Adelie  Torgersen          NA           NA           NA           NA
## 5 Adelie  Torgersen         36.7          19.3          193          3450
## 6 Adelie  Torgersen         39.3          20.6          190          3650
## 7 Adelie  Torgersen         38.9          17.8          181          3625
## 8 Adelie  Torgersen         39.2          19.6          195          4675
## 9 Adelie  Torgersen         34.1          18.1          193          3475
## 10 Adelie Torgersen         42           20.2          190          4250
## # i 142 more rows
## # i 2 more variables: sex <fct>, year <int>
hist(adelie$flipper_length_mm)
```


Histogram of adelie\$flipper_length_mm



```
# dealing with numeric columns with NA values  
mean(adelie$flipper_length_mm)
```

```
## [1] NA
```

```
mean(adelie$flipper_length_mm, na.rm = TRUE)
```

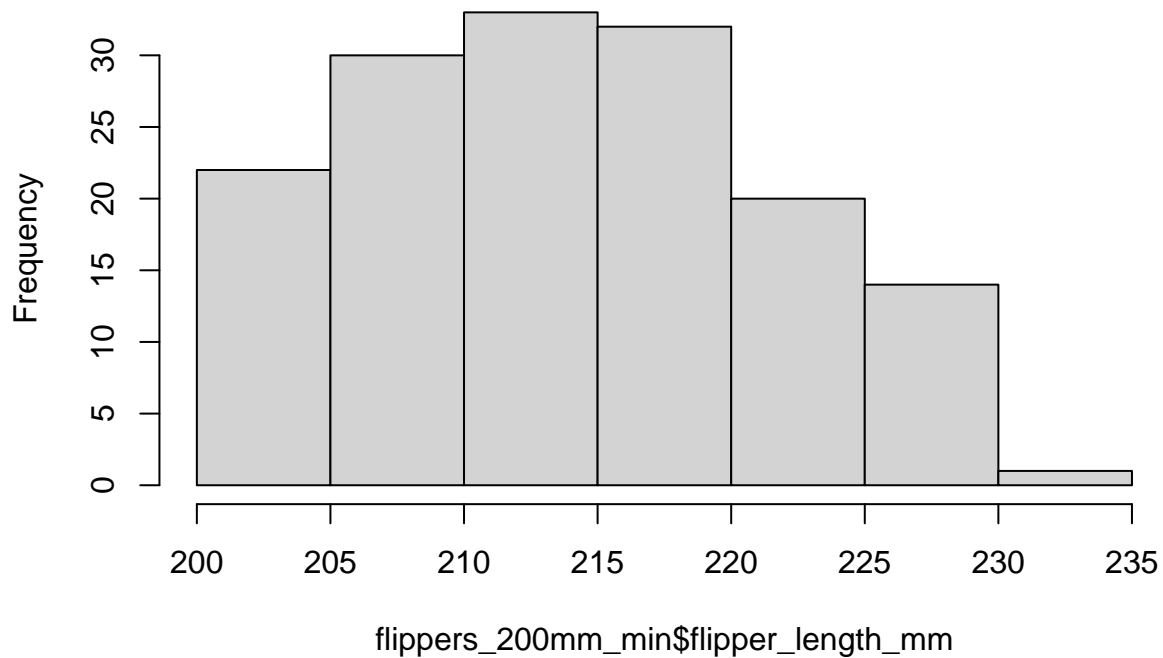
```
## [1] 189.9536
```

We can also use conditional formatting to filter rows based on numeric conditions.

```
# penguins with flippers greater than or equal to 200 mm  
flippers_200mm_min <- penguins[penguins$flipper_length_mm >= 200, ]
```

```
# create a histogram  
# hist(flippers_200mm_min) # why doesn't this work? We haven't specified a column  
hist(flippers_200mm_min$flipper_length_mm)
```

Histogram of flippers_200mm_min\$flipper_length_mm



Challenge

Write some lines of code to do the following: calculate the minimum (`min()`), maximum (`max()`), and the standard deviation (`sd()`) of the body mass values for Gentoo penguins. Remember the `na.rm` argument!

Then, plot a histogram of the Gentoo body mass data.

```
gentoo <- penguins[penguins$species == "Gentoo", ]  
min(gentoo$body_mass_g, na.rm = TRUE)
```

```
## [1] 3950
```

```
max(gentoo$body_mass_g, na.rm = TRUE)
```

```
## [1] 6300
```

```
hist(gentoo$body_mass_g)
```

Histogram of gentoo\$body_mass_g

