Introduction to R for Biologists

Intro to R and the Tidyverse Ecosystem

Developed by Rachael Cox and Alex Lukasiewicz

Welcome to Class!!

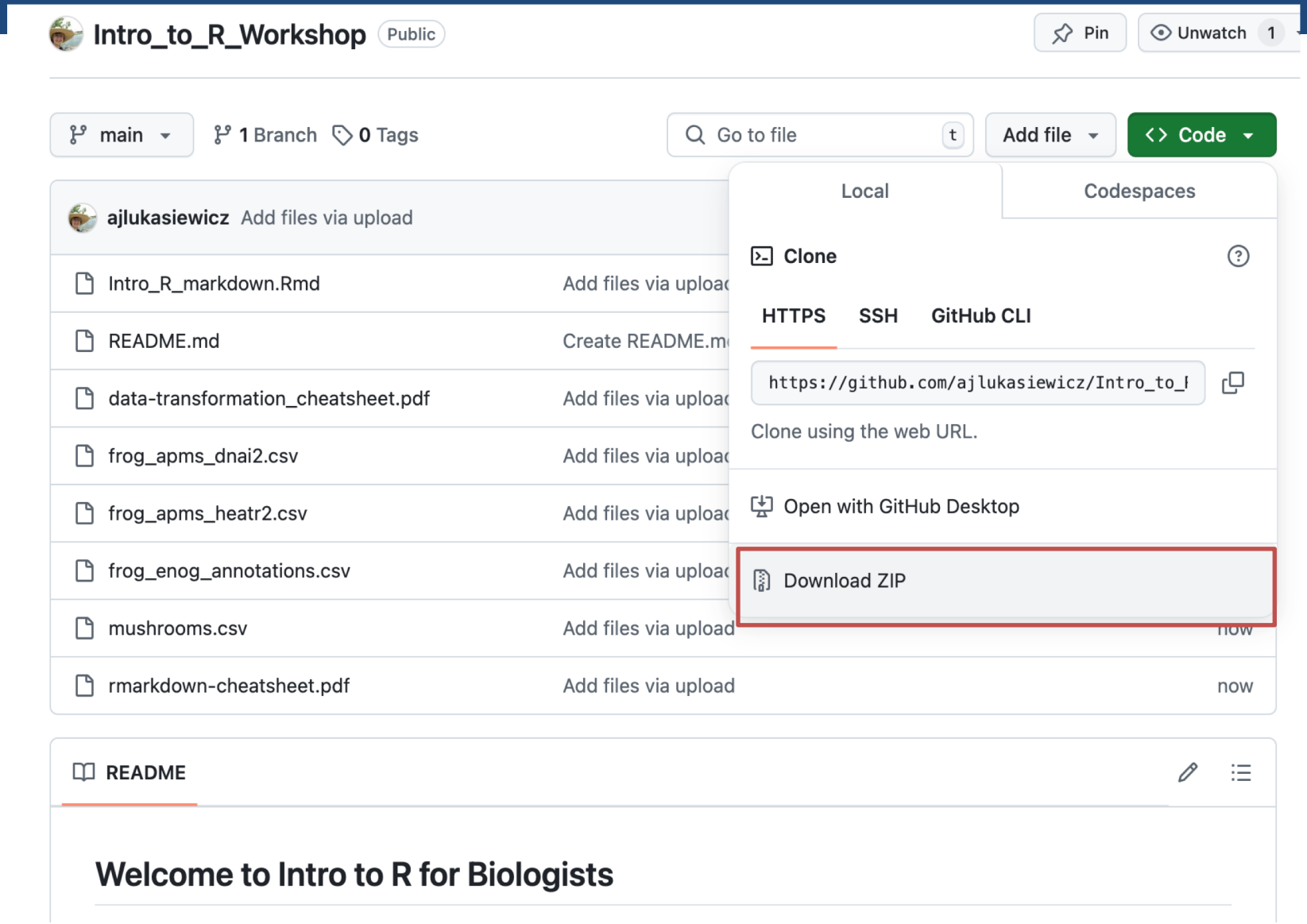
Alexandra Lukasiewicz
(they/them)
PhD Candidate
Contreras Lab

Course Format

Concept Slides and Code Along Activities

- **Lecture -> Coding -> Lecture -> Coding**
- Check email for link to my Github
  - Download zip file for this workshop
  - https://github.com/ajlukasiewicz/Intro_to_R_Workshop/

Course Format



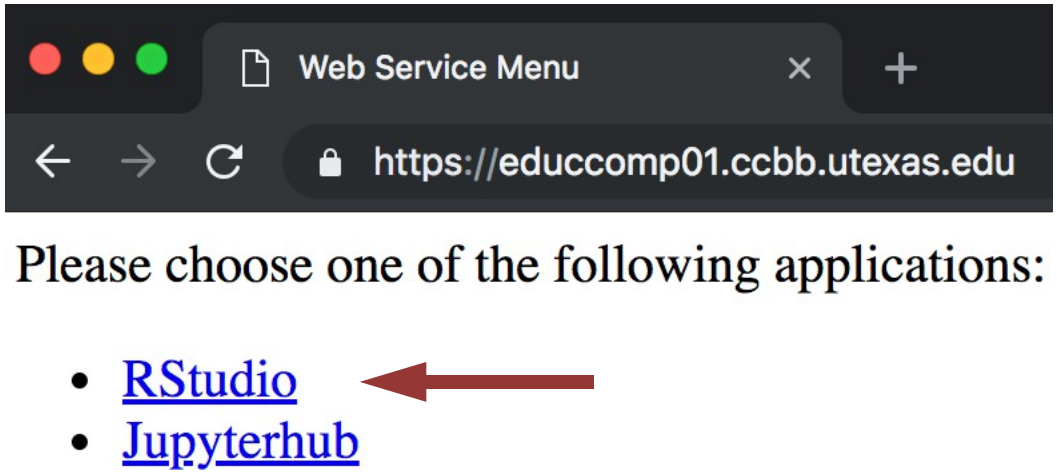
Workshop Outline

1. How to get set up using R (On the POD and Locally)
2. How and why to use RStudio & R Markdown (.Rmd)
3. Basics of programming
   - Data types
   - Functions
   - Troubleshooting
4. Intro to the Tidyverse
   - Tidy vs untidy data
   - Tidyverse-specific functions

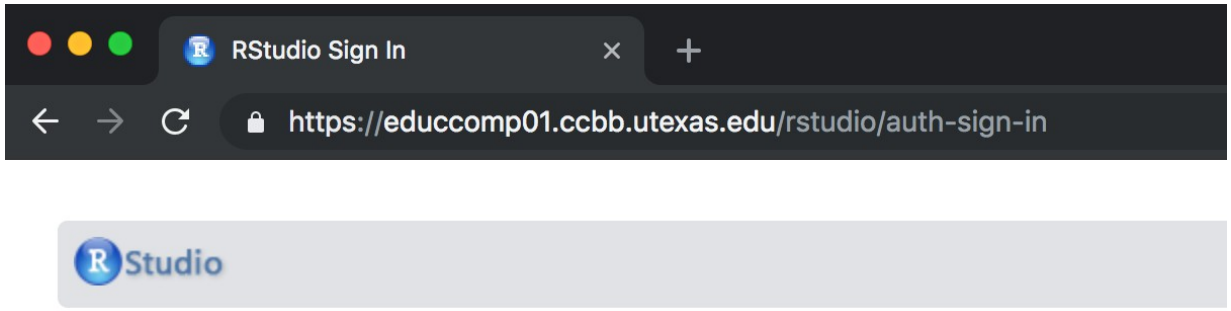
Access R Studio through your web browser

1. https://gsafcomp01.ccbb.utexas.edu/
2. https://gsafcomp02.ccbb.utexas.edu/

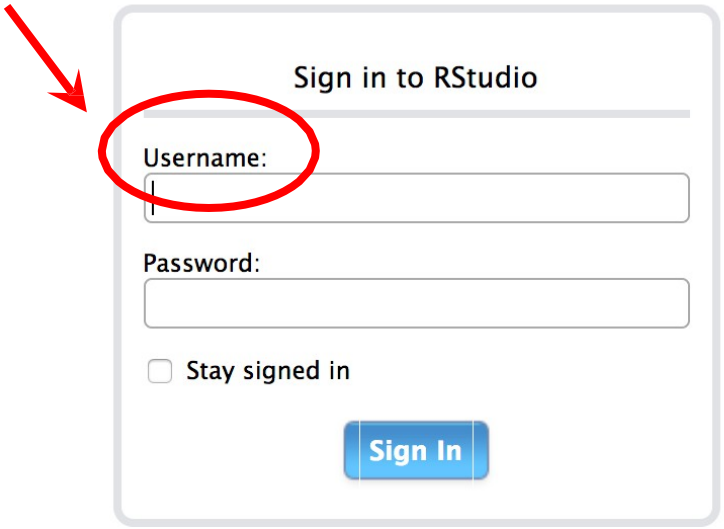
Select RStudio



Sign in with your student# and password



Refer to “Student Accounts” file for your username



R: The premier data analysis and visualization platform

Step 1: install R
https://cran.r-project.org/

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

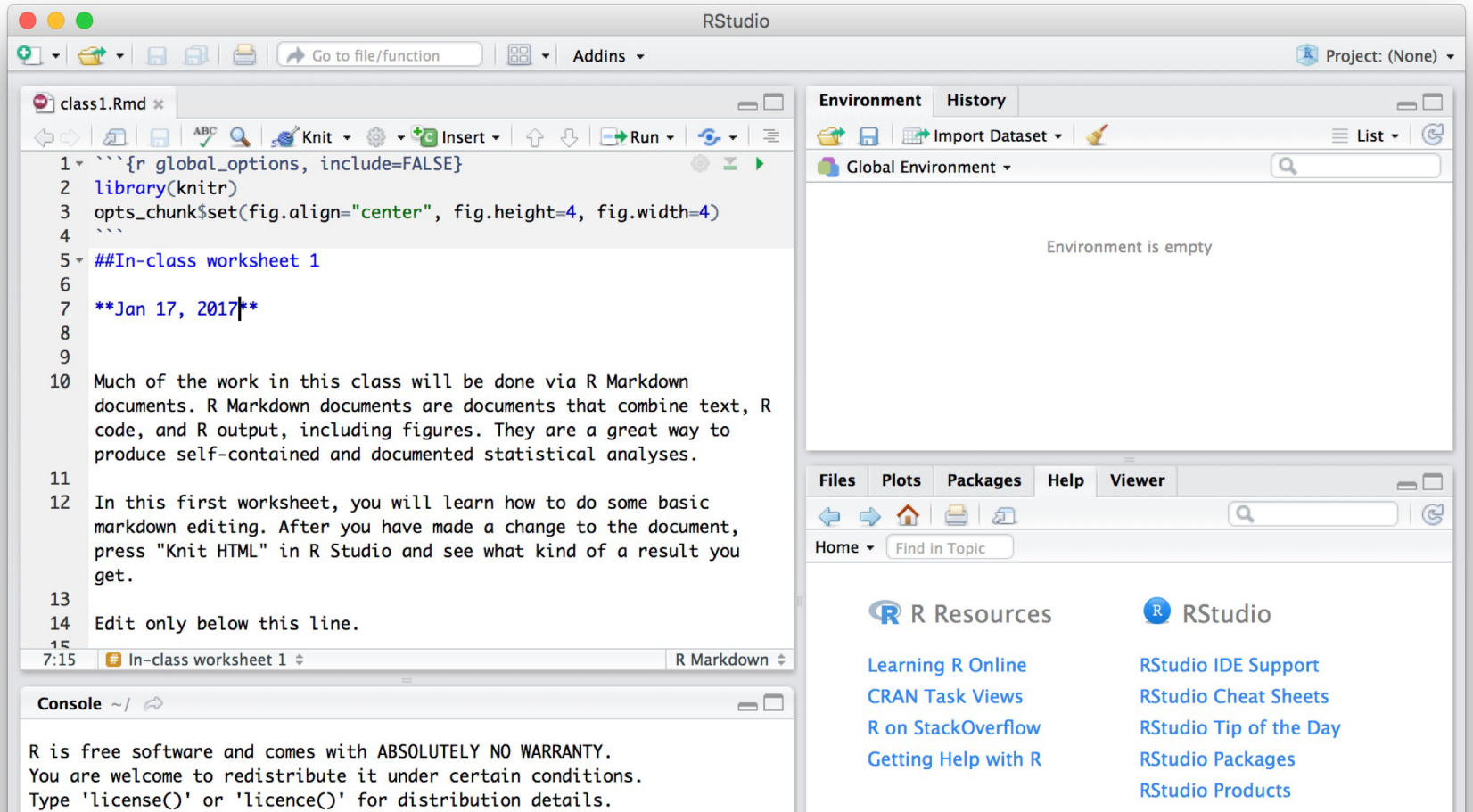
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper

R Studio:
A nice user interface for R

Step 2: Install RStudio
https://www.rstudio.com/products/rstudio/download/

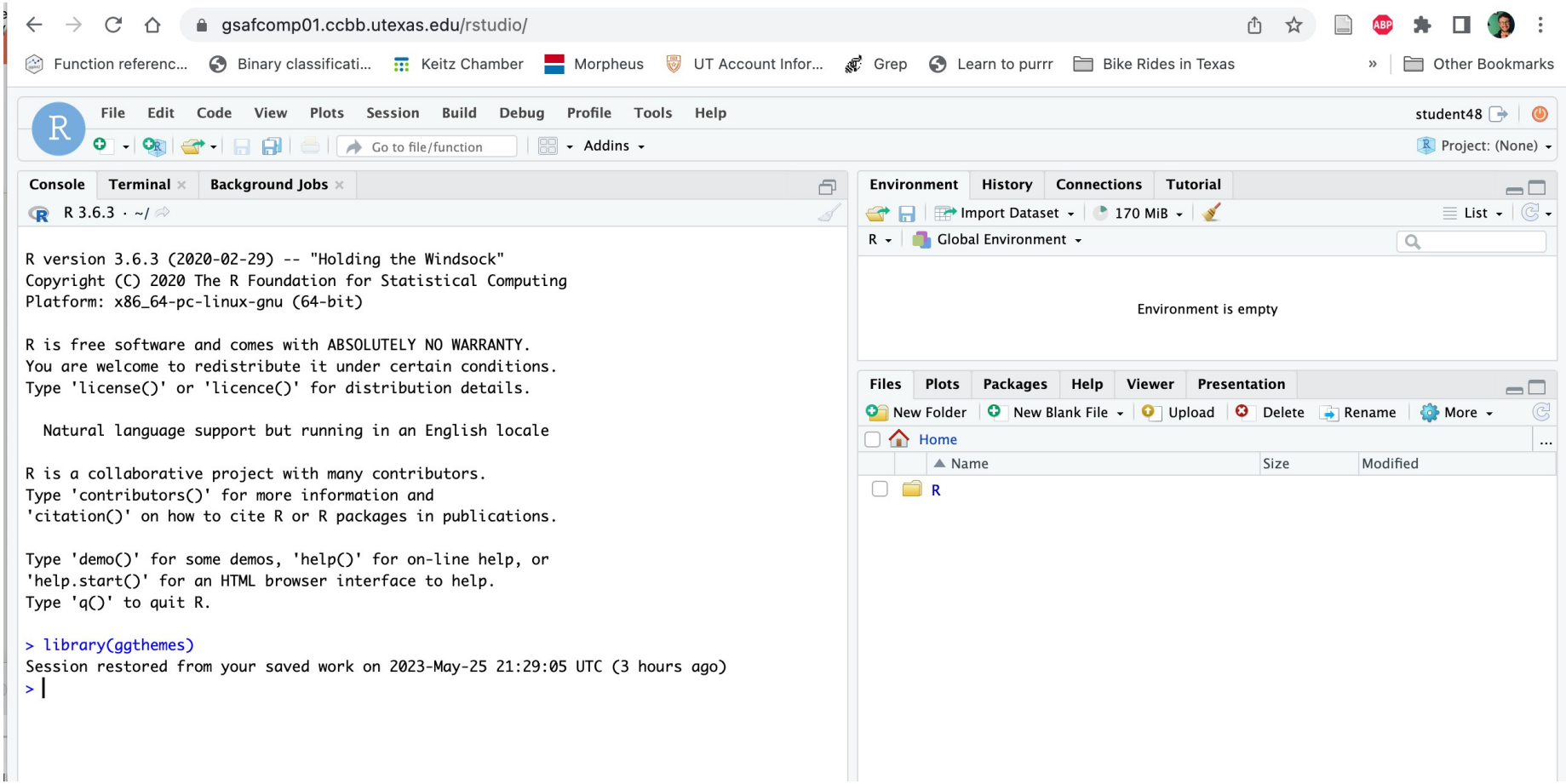


Note about local installation

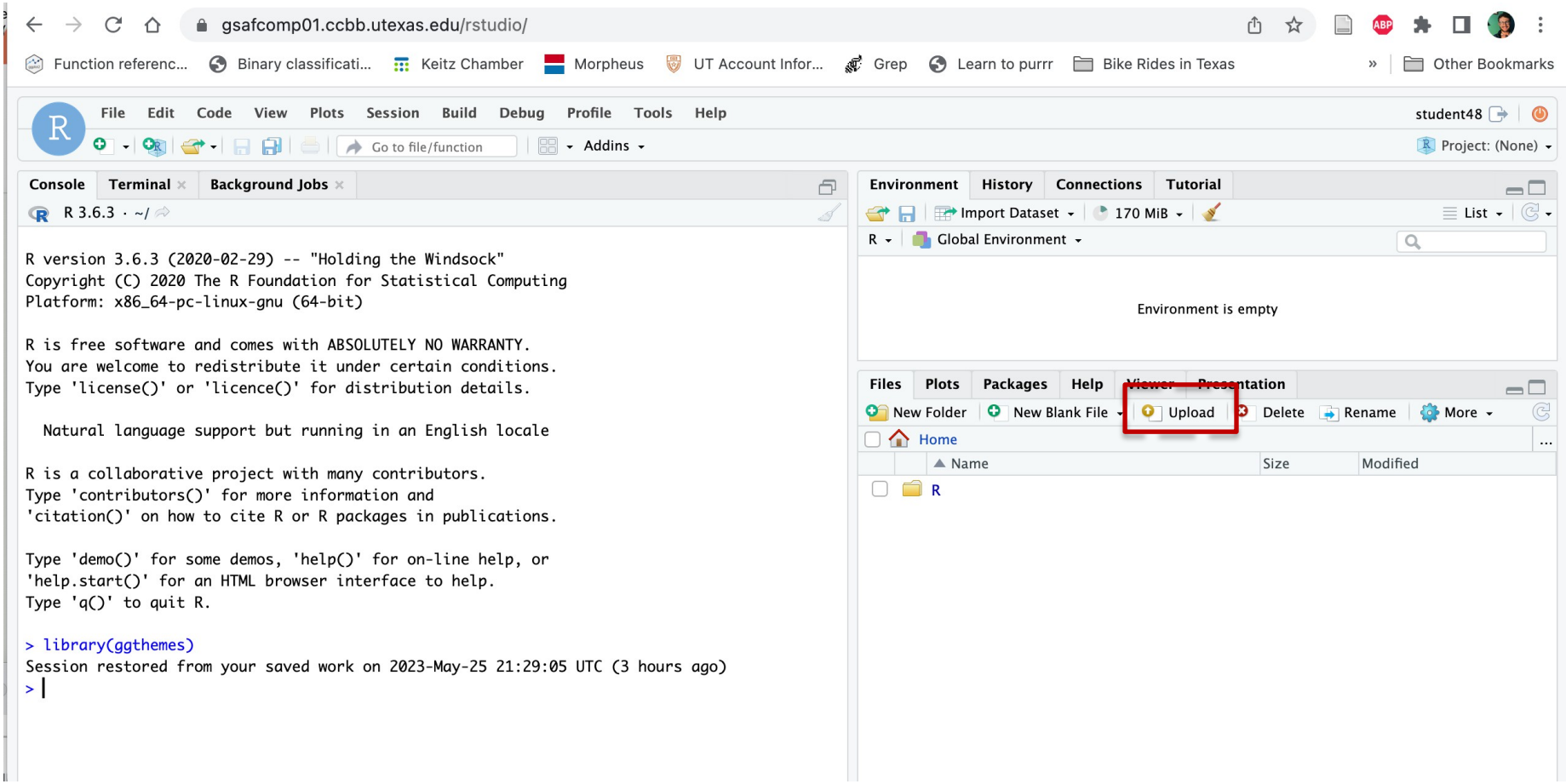
Installing R locally will require you to install additional packages, such as ”Tidyverse” and ”DESeq”
I will add the installation code for these in the markdown files

R Markdown

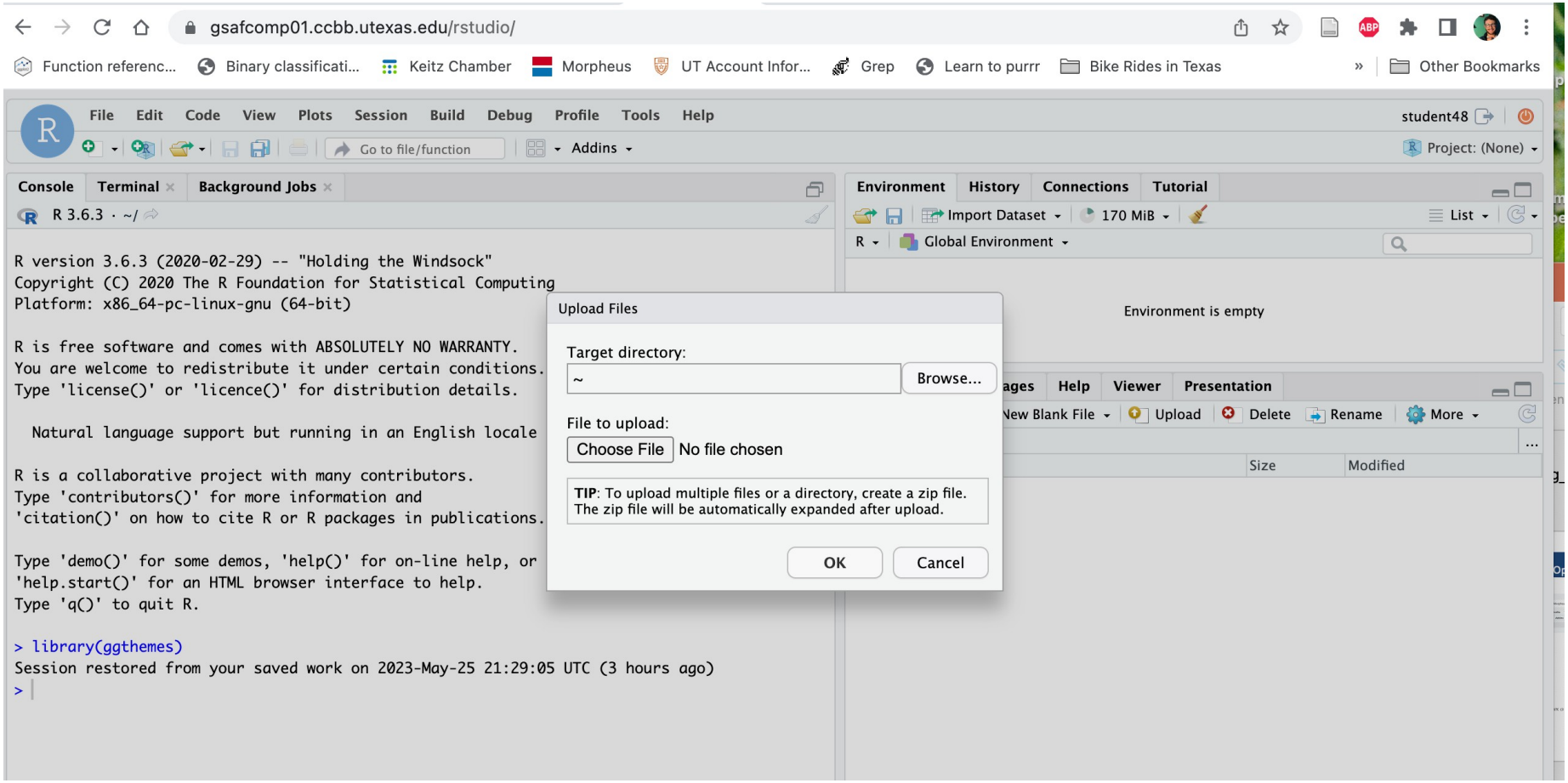
R Markdown:
Open the markdown



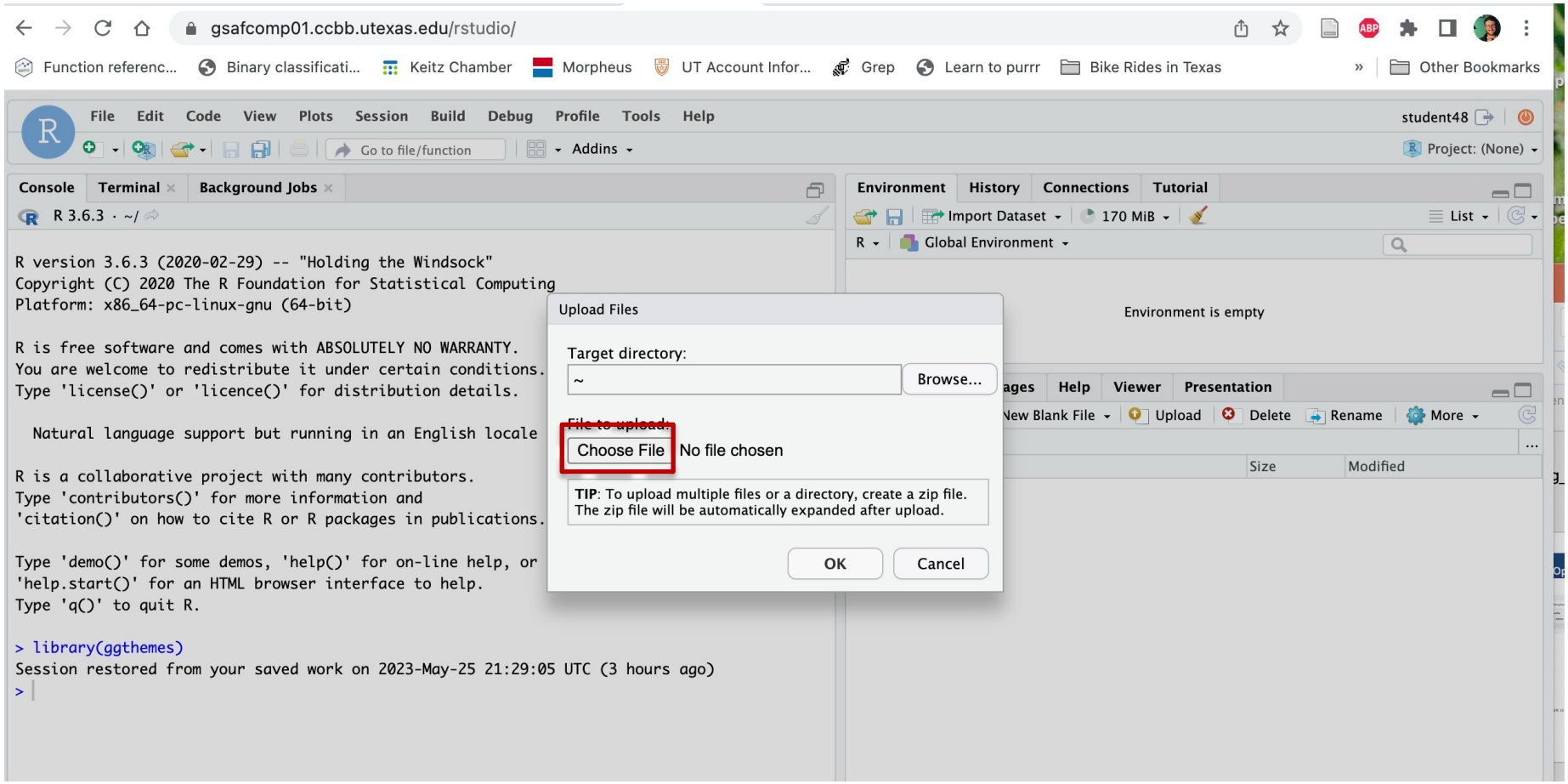
R Markdown:
Open the markdown



R Markdown:
Open the markdown

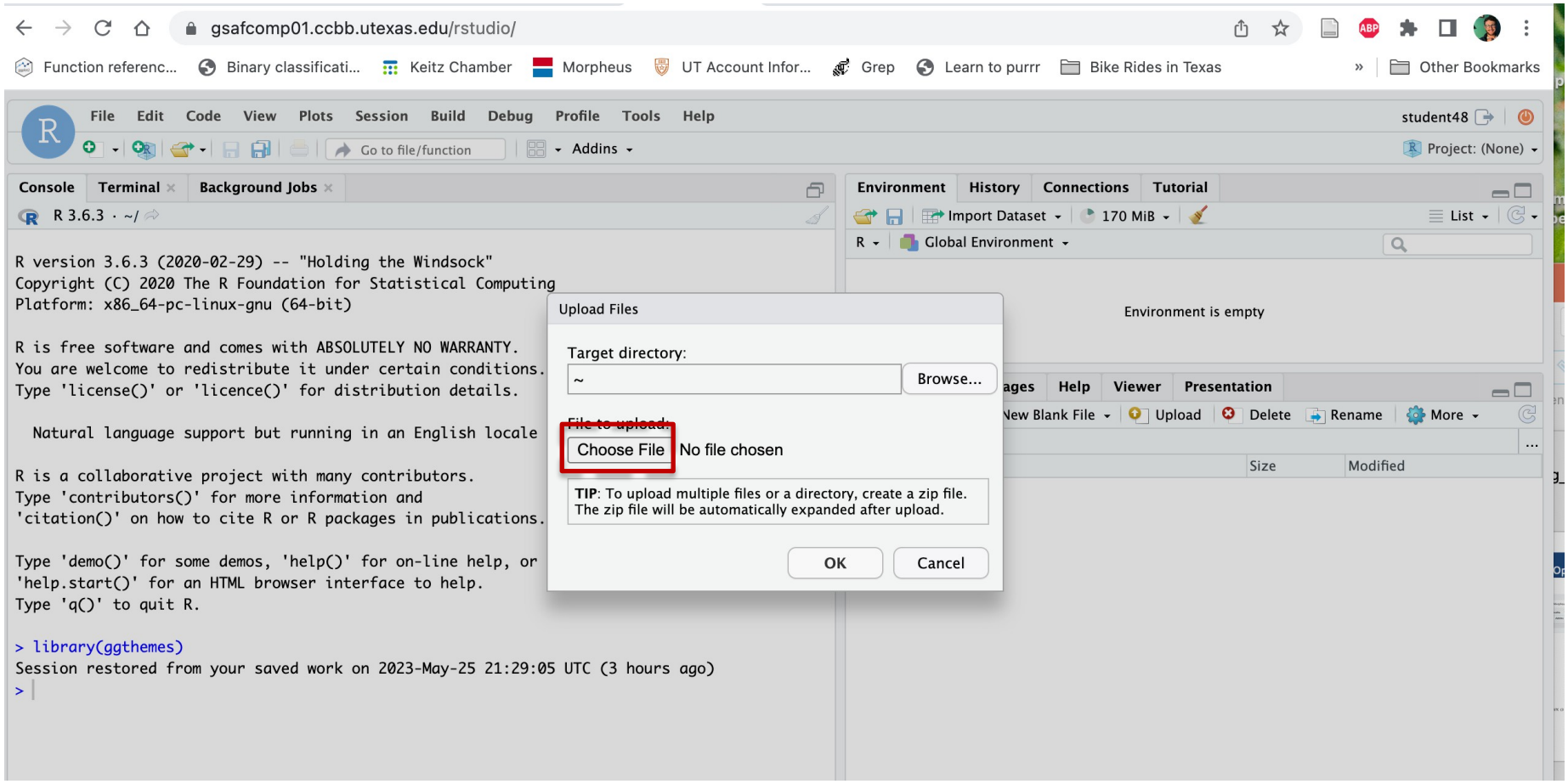


R Markdown:
Open the markdown

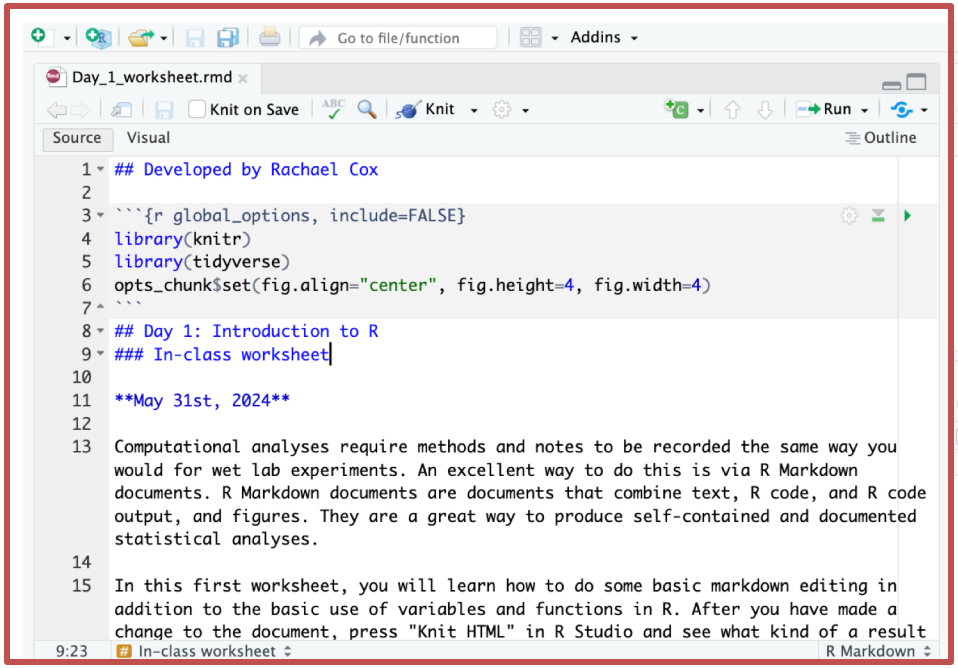


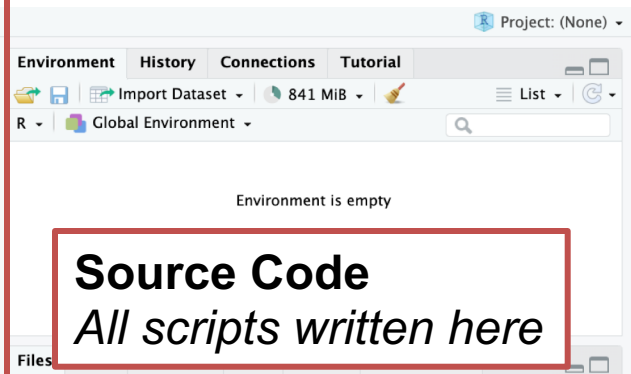
R Markdown:
Open the markdown

Demonstration Time!!

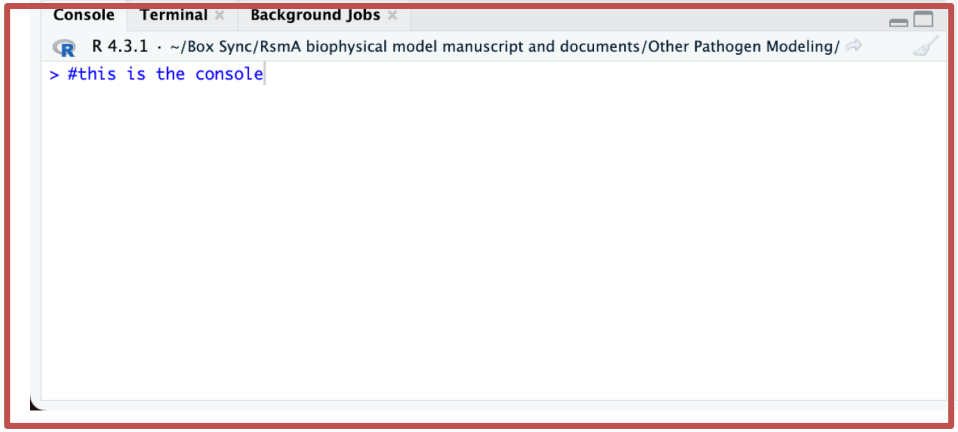


R Markdown: Relevant Sections



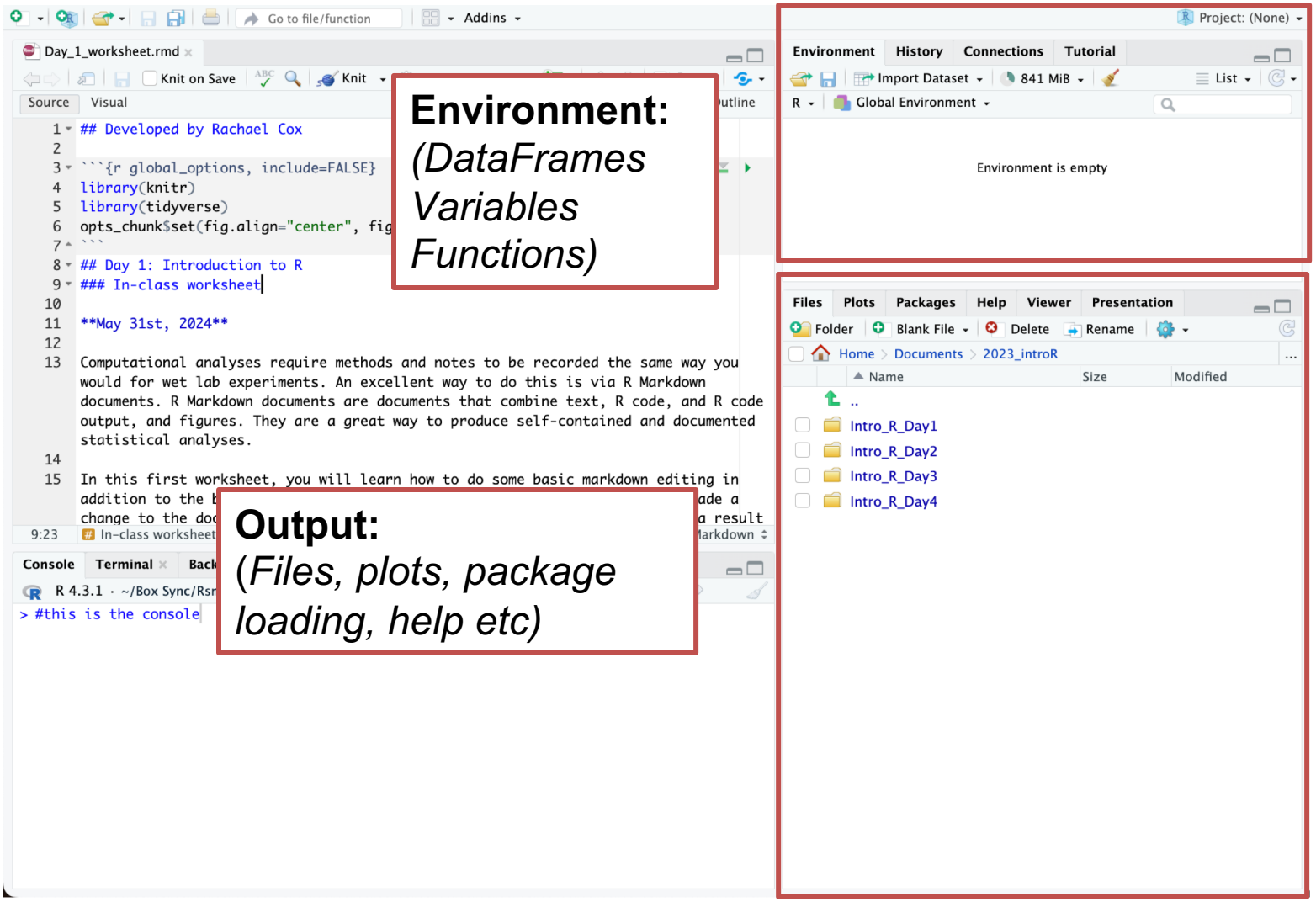


Source Code
All scripts written here



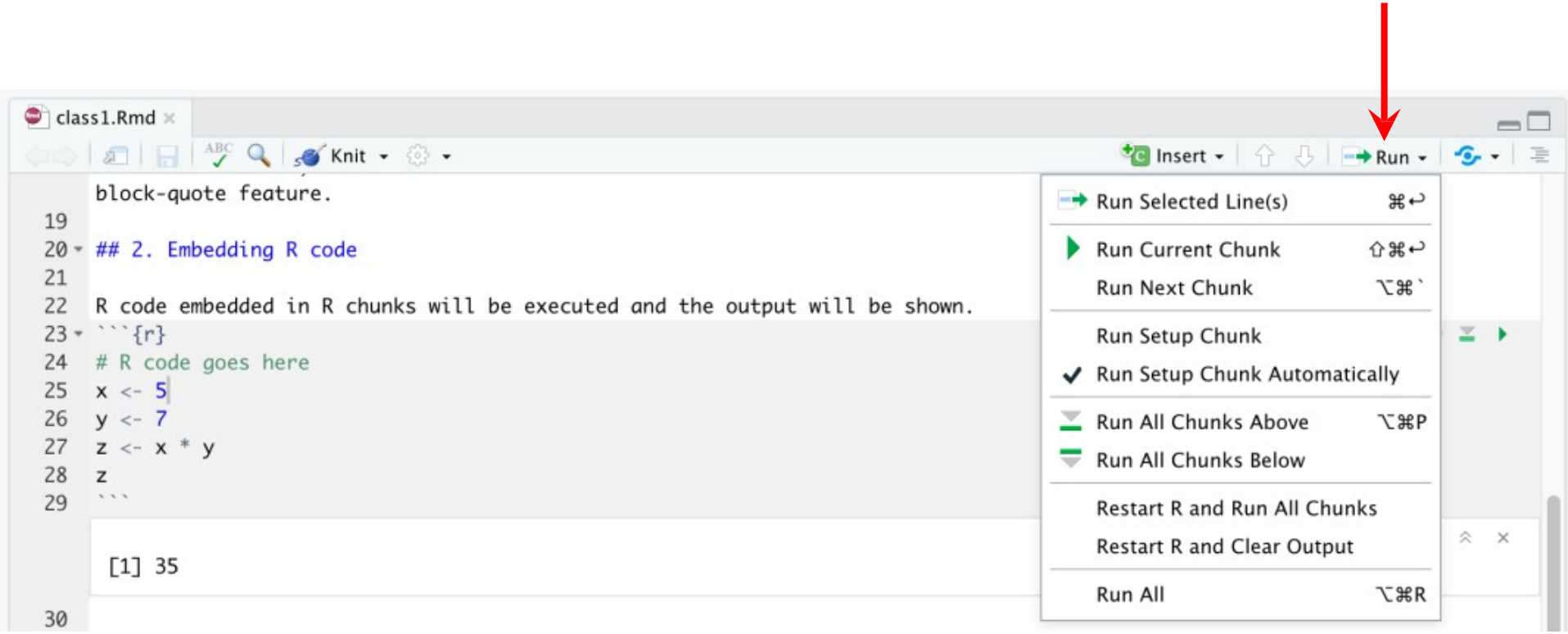
Console
Output and history of executed code

R Markdown: Relevant Sections



Different ways to execute code in RStudio

Option 1: Press the "Run" button



Option 2: Highlight code you want to execute and press ctrl+Enter (cmd+Enter on Macs)

R code embedded in R chunks will be executed and the output will be shown.

```
```{r}
# R code goes here
x <- 5
y <- 7
z <- x * y
z
```
```

Console | Terminal | Jobs

~/Desktop/projects/

```
> x <- 5
> y <- 7
> z <- x * y
> z
[1] 35
>
```

Option 3: Place pointer on line of code you want to execute, press ctrl+Enter (cmd+Enter on Macs)

R code embedded in R chunks will be executed and the output will be shown.

```
```{r}
# R code goes here
x <- 5
y <- 7
z <- x * y
z
```
```

Console | Terminal | Jobs

~/Desktop/projects/

```
> z <- x * y
>
```

Use ctrl+Shift+Enter (cmd+Shift+Enter on Macs) to execute an entire code chunk

R code embedded in R chunks will be executed and the output will be shown.

```
```{r}
# R code goes here
x <- 5
y <- 7
z <- x * y
z
```
```

Console | Terminal | Jobs

~/Desktop/projects/

```
> x <- 5
> y <- 7
> z <- x * y
> z
[1] 35
>
```

Shortcuts for coding

- **Ctrl+Shift+C** (Cmd+Shift+C on Macs) will comment/uncomment a line or multiple lines
- **Tab** and **Shift+Tab** will indent and un-indent lines, respectively
- **Ctrl+Shift+M** (Cmd+Shift+M on Macs) produces a pipe operator %>% (will be used within tidyverse)

R Programming Basics

Variable assignments and objects

```
> x <- 5
> x
[1] 5
```

Assign **number** 5 to **variable** x

```
> 5*x^2+7
[1] 132
```

Calculate $5*x^2+7$

```
> y <- c(1, 2, 3, 4, 5)
> y
[1] 1 2 3 4 5
```

Create object (a **vector**), assign to **variable** y

```
> x*y
[1]  5 10 15 20 25
```

Multiply each element in **vector** y with the **number** in x

Strings

A **string** contains text:

```
> name <- “Alex L"
> name
[1] “Alex L"
```

A **vector** of **strings**:

```
> animals <- c("cat", "mouse", "mouse",
"cat", "rabbit")
> animals
[1] "cat"    "mouse"  "mouse"  "cat"
"rabbit"
```

Factors

Factors keep track of distinct categories (levels) in a vector:

```
> animals
[1] "cat"    "mouse"  "mouse"  "cat"
"rabbit”

> factor(animals)
[1] cat    mouse  mouse  cat    rabbit
Levels: cat mouse rabbit
```

Data frames

We use **data frames** to store data sets with multiple variables:

```
> pets <- data.frame(
    family = c(1, 2, 3, 4, 5),
    pet = animals
  )
```

```
> pets
  family    pet
1      1    cat
2      2  mouse
3      3  mouse
4      4    cat
5      5 rabbit
```

Data frames

We access individual columns in a data frame with $ + the column name:

```
> pets$family
[1] 1 2 3 4 5


> pets$pet
[1] cat    mouse  mouse  cat    rabbit
Levels: cat mouse rabbit
```

Demonstration Time!

Work on Section #1

Data frames

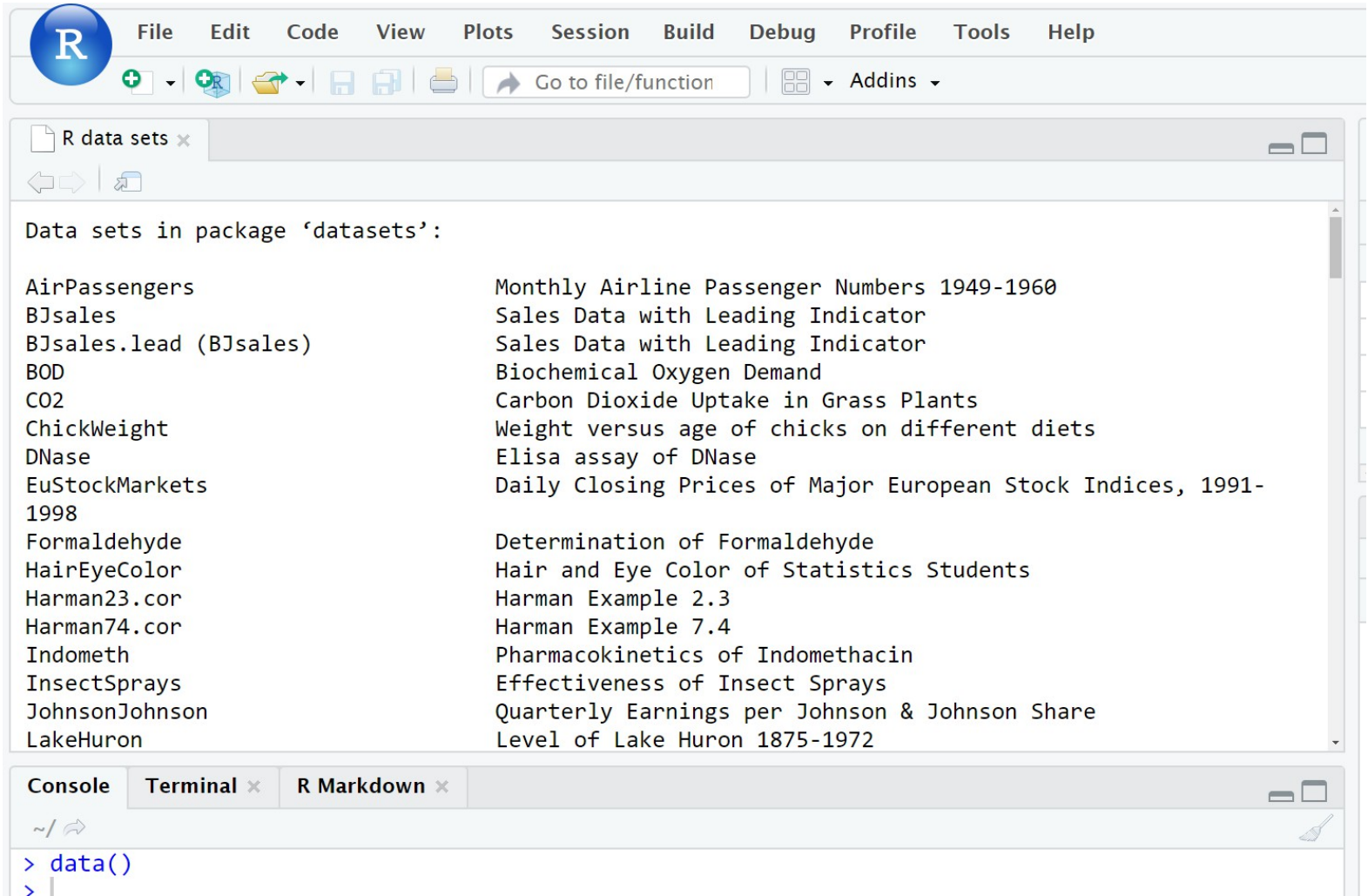
R has many built-in data frames:

```
> cars
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
7     10   18
8     10   26
9     10   34
```

```
> chickwts
    weight      feed
1      179 horsebean
2      160 horsebean
3      136 horsebean
4      227 horsebean
...    ...       ...
11     309   linseed
12     229   linseed
13     181   linseed
14     141   linseed
```

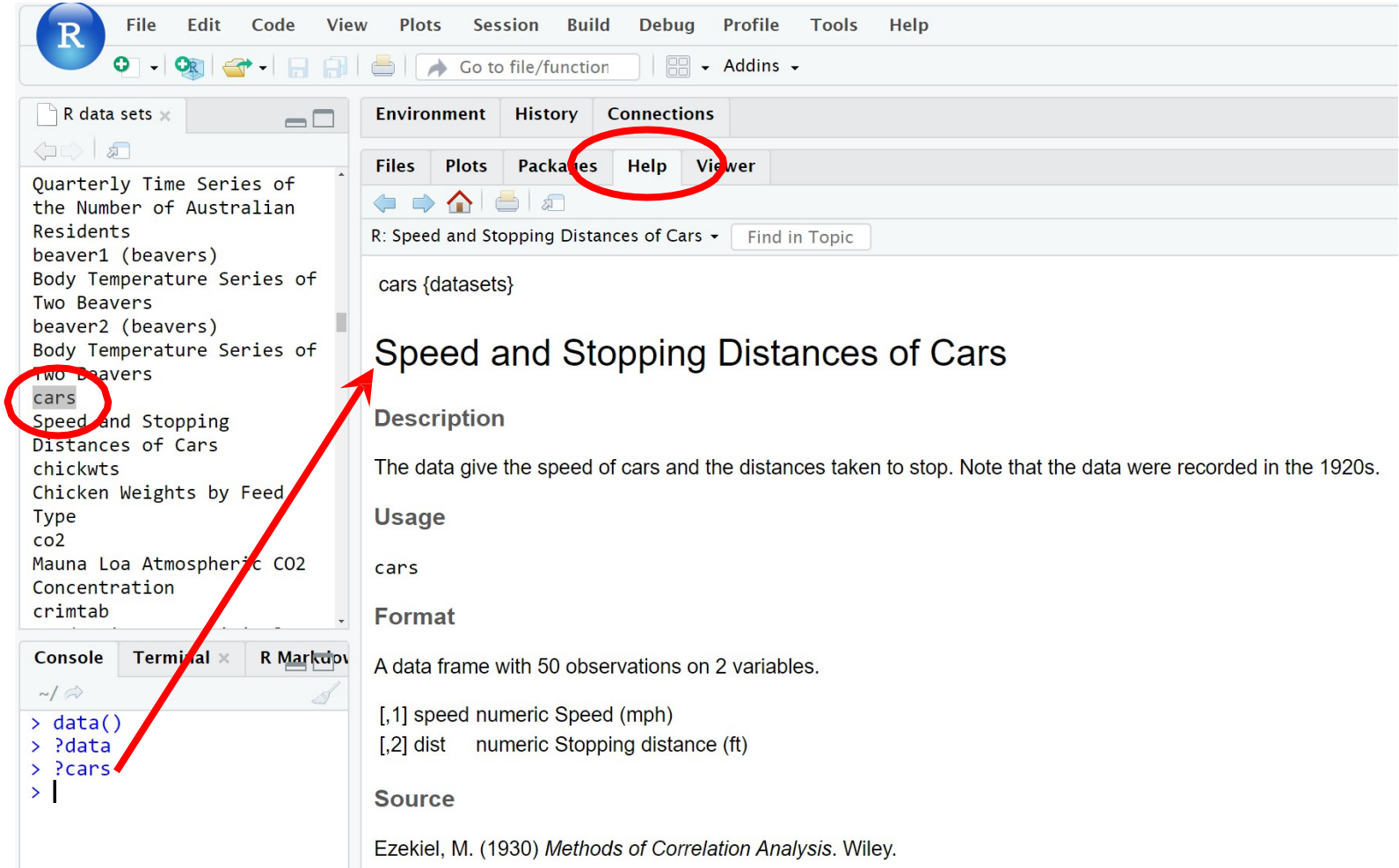
Data frames

Available built-in datasets can be accessed with `data()`



Data frames

Data set information can be accessed with `?dataset`



Data frames

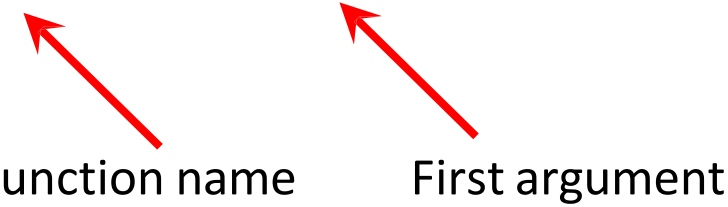
The `head()` function shows the first few lines of a data frame:

```
> head(cars)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
>
```

Functions

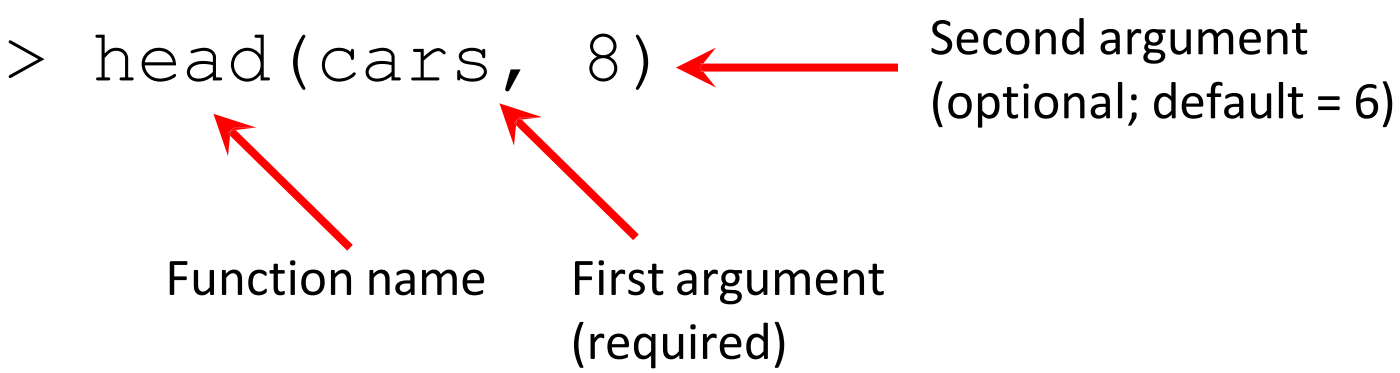
Functions are called in the format `function(argument)`

```
> head(cars)
```



Functions

Functions can have any number of required arguments or optional arguments



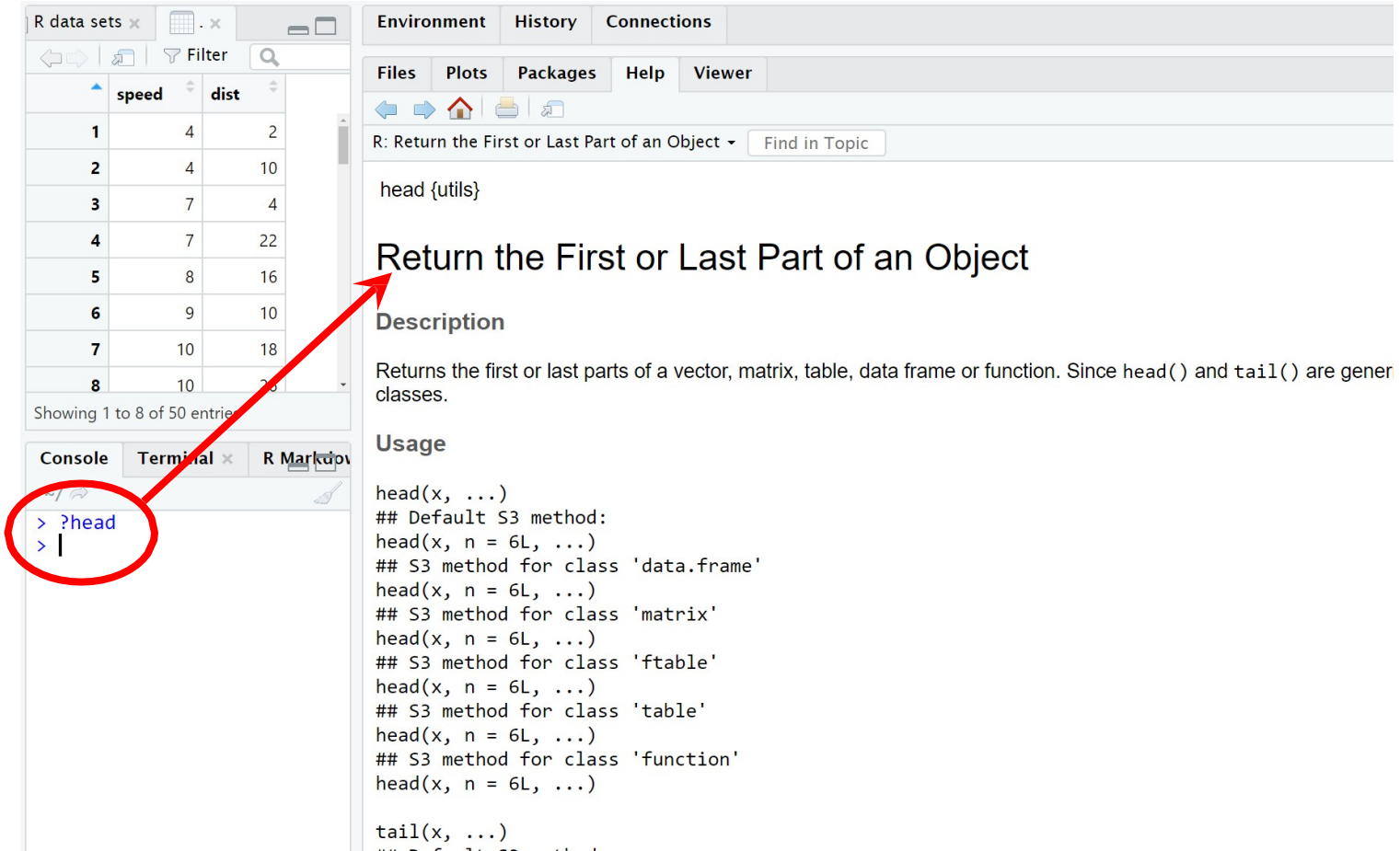
Functions

`head(cars, 8)` will show the first 8 lines of the data frame instead of the default 6:

```
> head(cars, 8)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
7    10   18
8    10   26
```

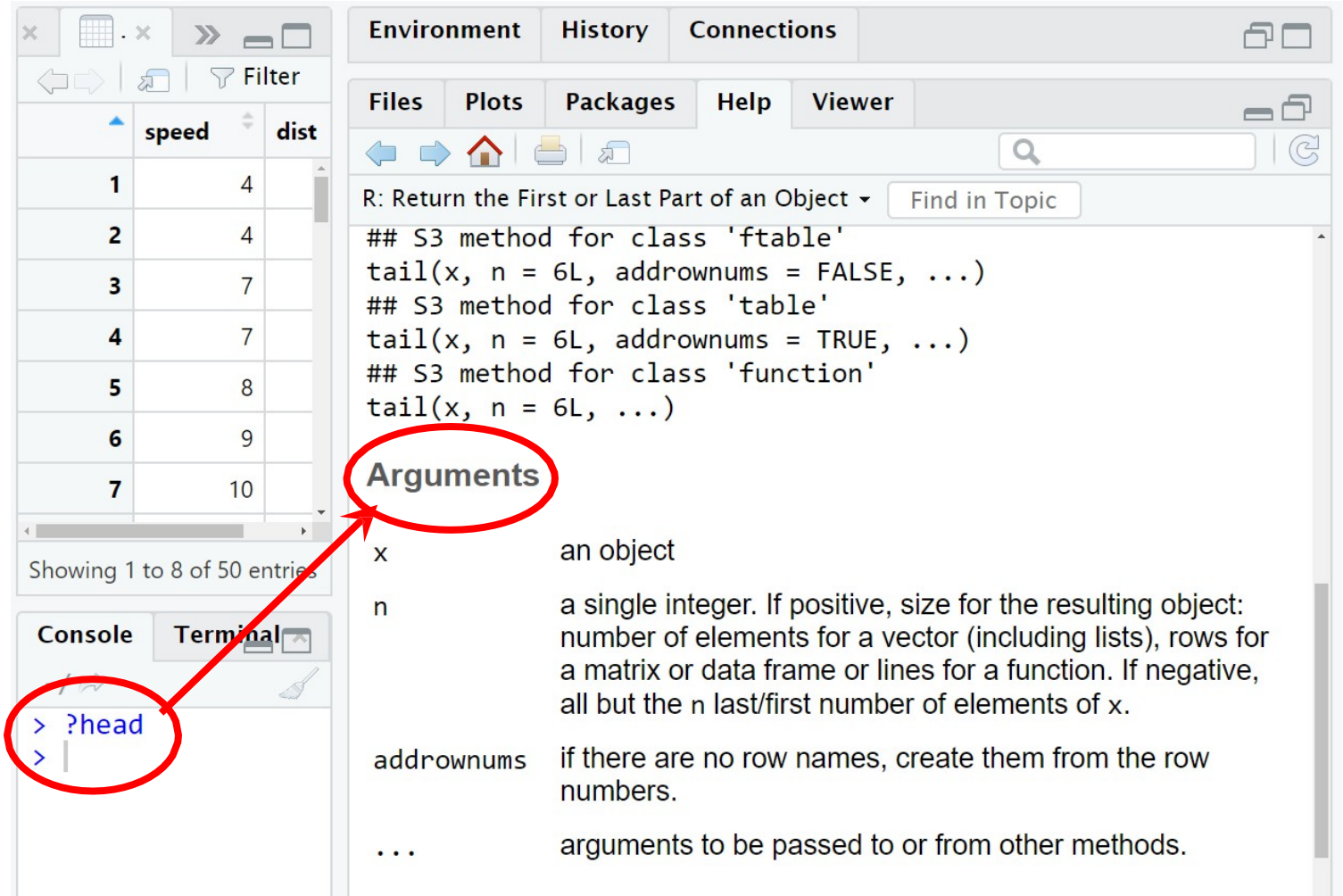
Functions

More information about what a function does and/or requires can be accessed with `?function`



Functions

`?function` has argument information



Functions

We can implicitly or explicitly pass arguments

```
> head(cars, 8)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
7    10   18
8    10   26
>
```

```
> head(x=cars, n=8)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
7    10   18
8    10   26
>
```

Reading and writing data

`write_csv(variable,"filename.csv")` is a package-specific function that allows you write R data frames to a comma-separated table

`read_csv("filename.csv")` is a package-specific function that allows you to import a file in your local environment:

```
> read_csv("mushrooms_tiny.csv")
```

*note: if you are running the workbook locally and have not installed tidyverse use: `write.csv()` and `read.csv()` with the same arguments

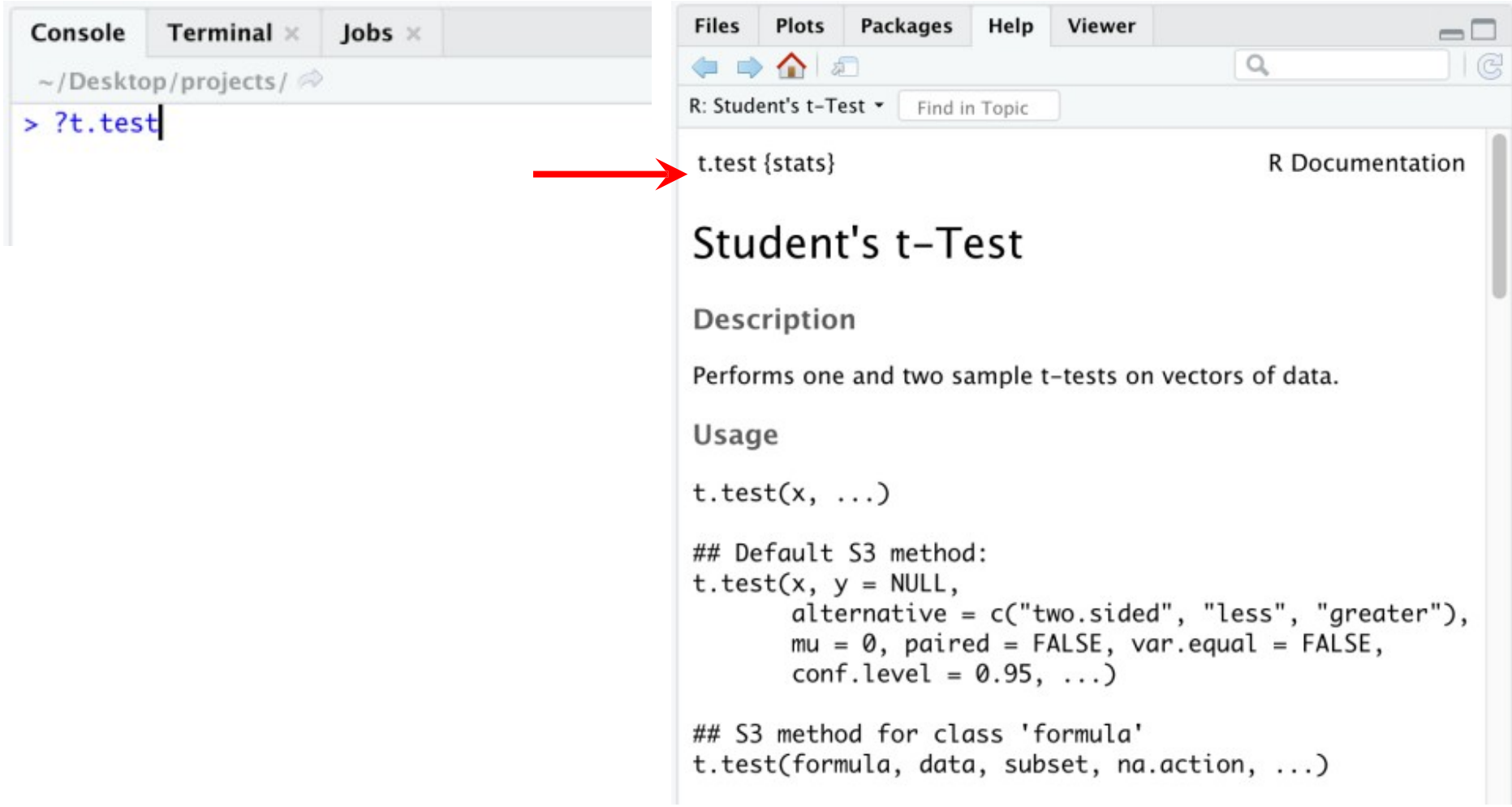
Demonstration Time!

Work on Sections 2 and 3

Troubleshooting

Ask RStudio for help

Type `?function` into console



Ask Google for help

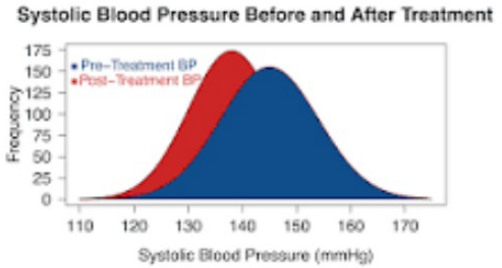
how do i run a t test in r

All Videos Images News Shopping More Settings Tools

About 2,780,000,000 results (0.64 seconds)

How to Perform T-**tests in R**. To conduct a one-sample t-**test in R**, we use the syntax t. **test**(y, mu = 0) where x is the name of our variable of interest and mu is set equal to the mean specified by the null hypothesis.

Aug 17, 2015



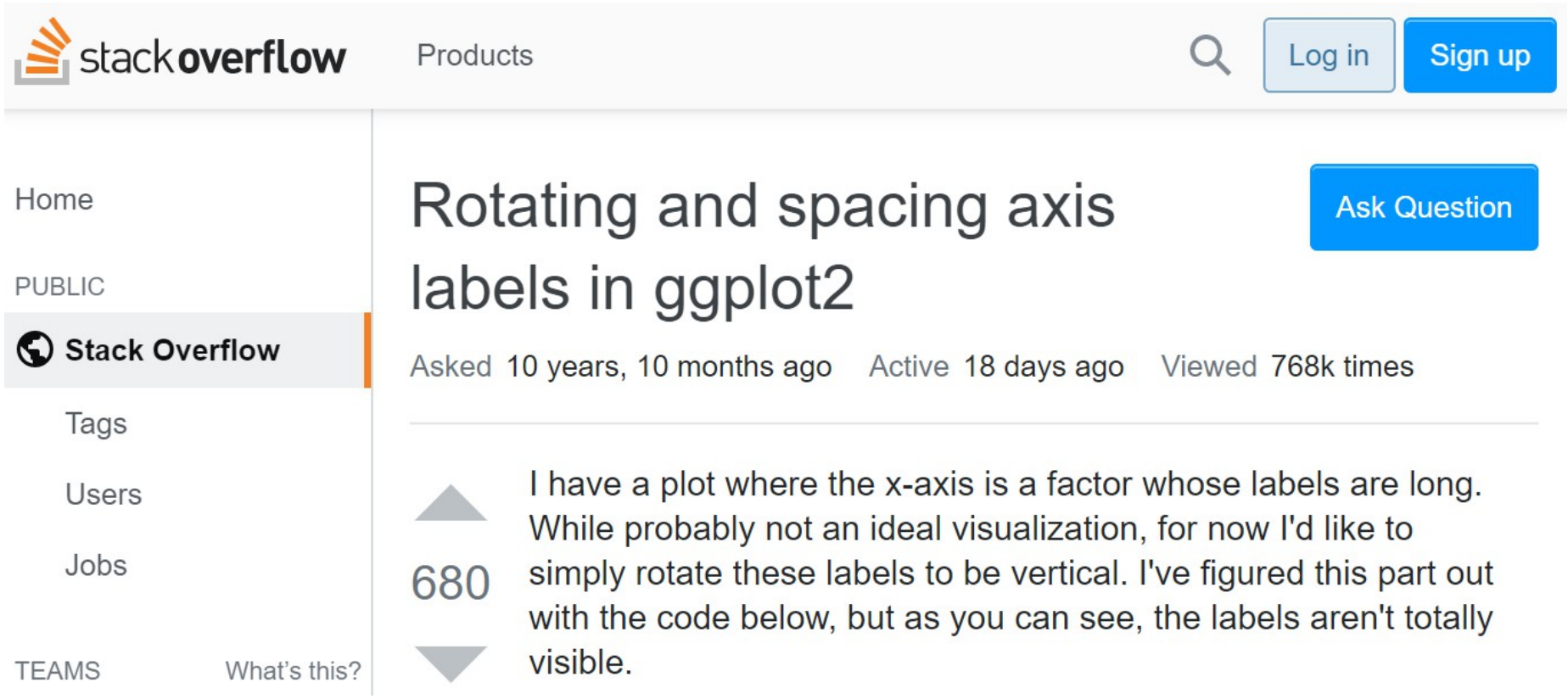
datascienceplus.com › t-tests

How to Perform T-tests in R | DataScience+

Error in t.test.default(x, y) : not enough 'x' observations

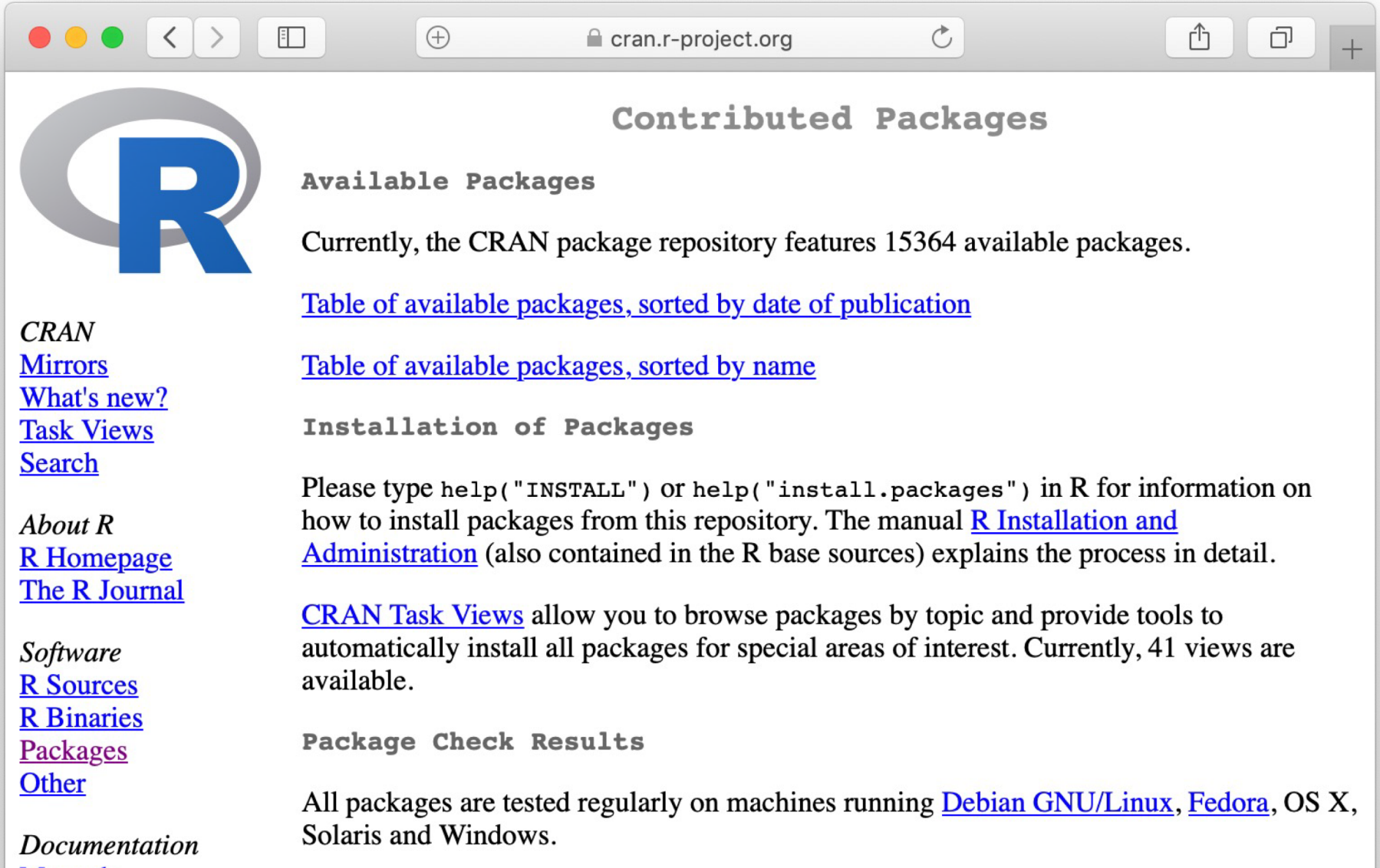
Error in t.test.default(x, y) : not enough 'x' observations - Google Search

Ask StackOverflow for help

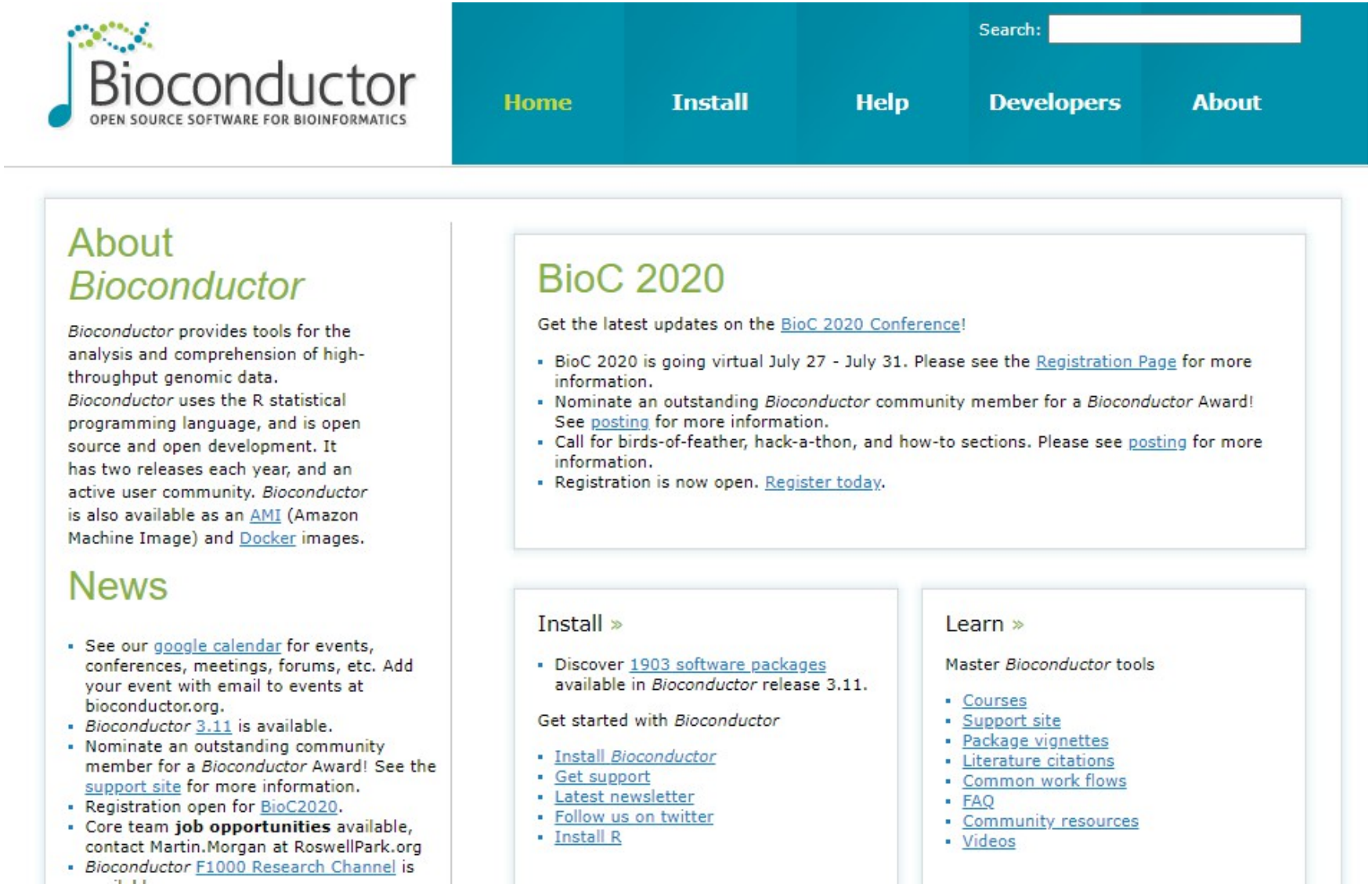


Extending R through packages: There's a package for everything

R packages are available on CRAN (Comprehensive R Archive Network)



Bio-specific R packages are available on Bioconductor



Break Time
Feel free to ask questions!

Workshop Outline

1. How to get set up using R
2. How and why to use RStudio & R Markdown (.Rmd)
3. Basics of programming
   - Data types
   - Functions
   - Troubleshooting
4. Intro to the Tidyverse
   - Tidy vs untidy data
   - Tidyverse-specific functions

Workshop Outline

1. How to get set up using R
2. How and why to use RStudio & R Markdown (.Rmd)
3. Basics of programming
4. Intro to the Tidyverse
   - Tidy vs untidy data
   - Tidyverse-specific functions

Especially in biology, we work with many different types of data which we collect, analyze, and communicate

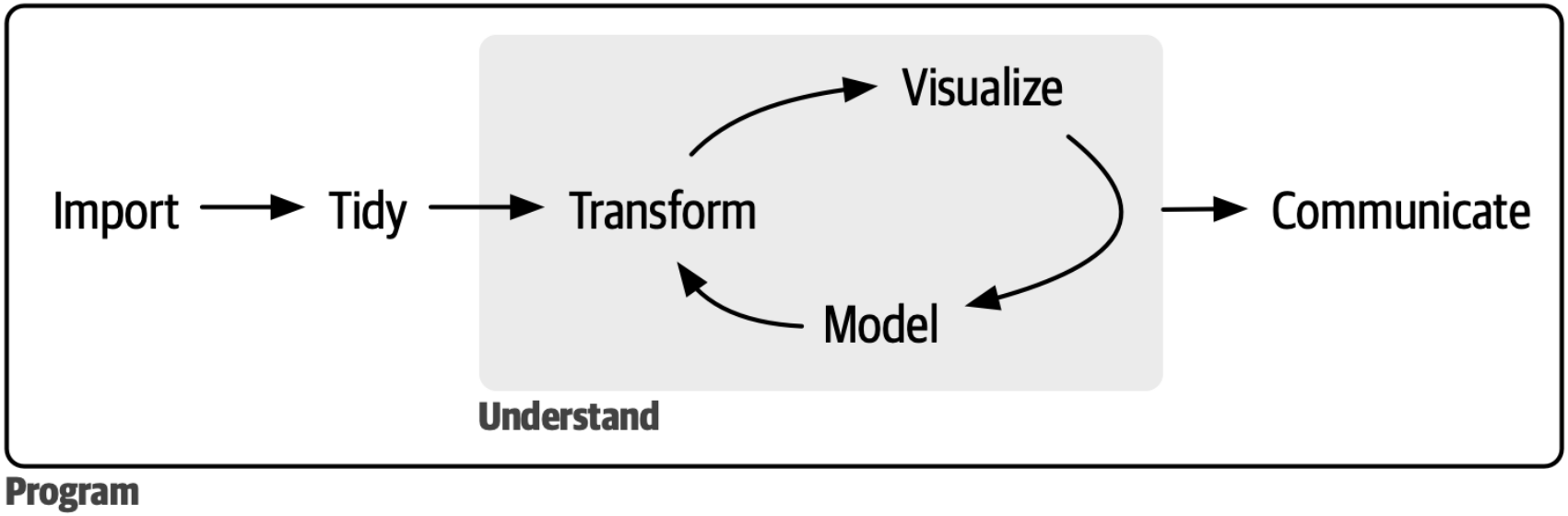


Figure 1, R for Data Science https://r4ds.hadley.nz/

In R, the collection of programs that make up the Tidyverse make this process easier and more reproducible

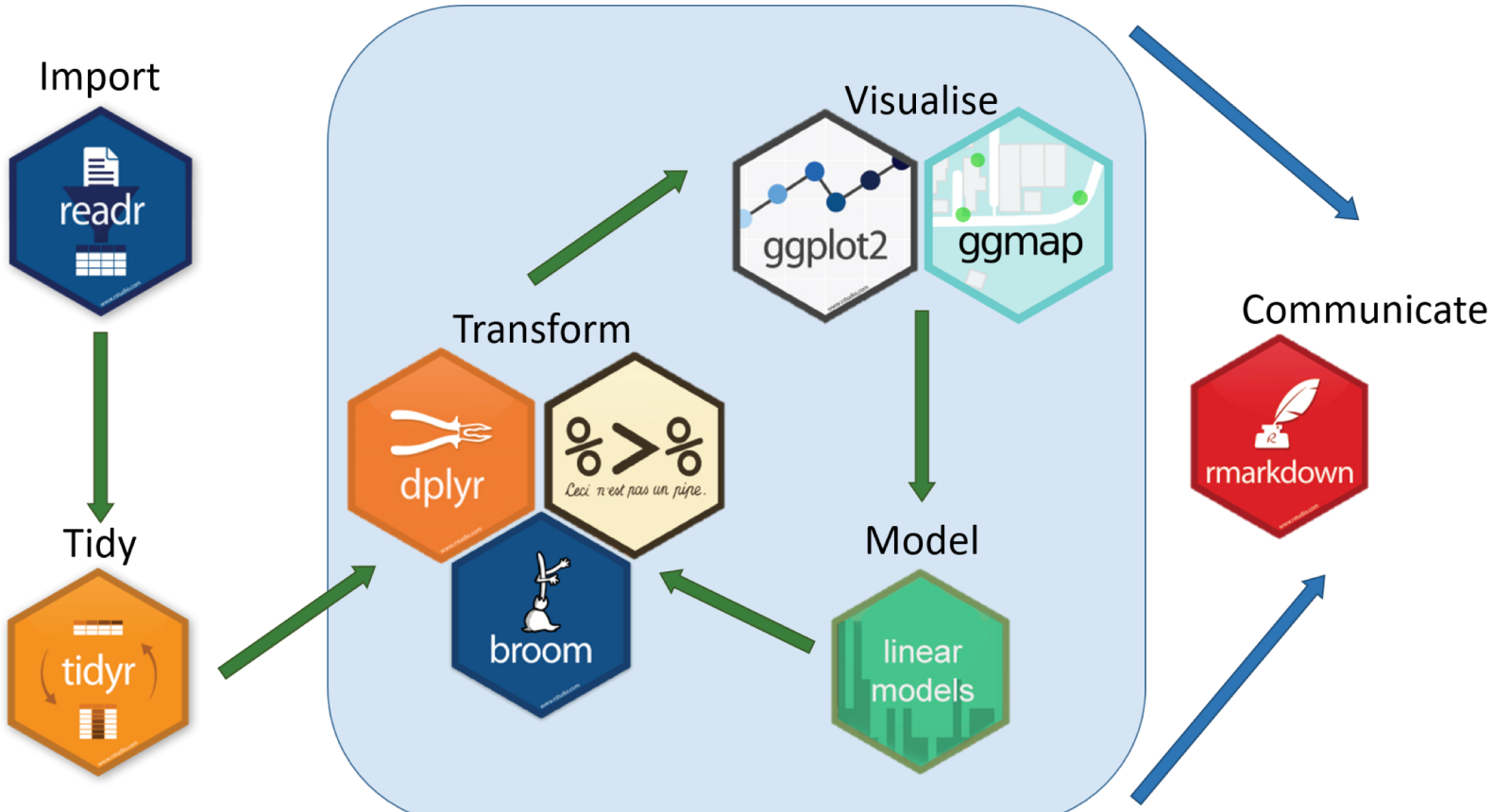


Image from: https://teachdatascience.com/tidyverse/

Tidy data

“Tidy datasets are all alike but every messy dataset is messy in its own way” — Hadley Wickham

Tidy data

Three rules:

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

Example: Contingency table

	survived	died
drug	15	3
placebo	4	12

not tidy

Example: Contingency table

	survived	**died**
drug	15	3
placebo	4	12

Y variable, outcome

not tidy

X variable, treatment

tidy

X **treatment**	Y **outcome**	**count**
drug	survived	15
drug	died	3
placebo	survived	4
placebo	died	12

Example: Contingency table, extended

	survived	died
drug	15	3
placebo	4	12

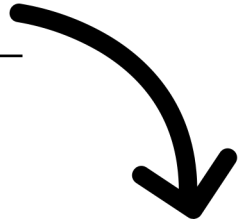
not tidy

tidy

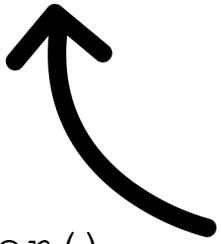
patient	treatment	outcome
1	drug	survived
2	drug	died
3	drug	survived
4	placebo	died
⋮	⋮	⋮

`tidyr` library provides functions for transforming tables

	survived	died
drug	15	3
placebo	4	12



`pivot_longer()`



`pivot_wider()`

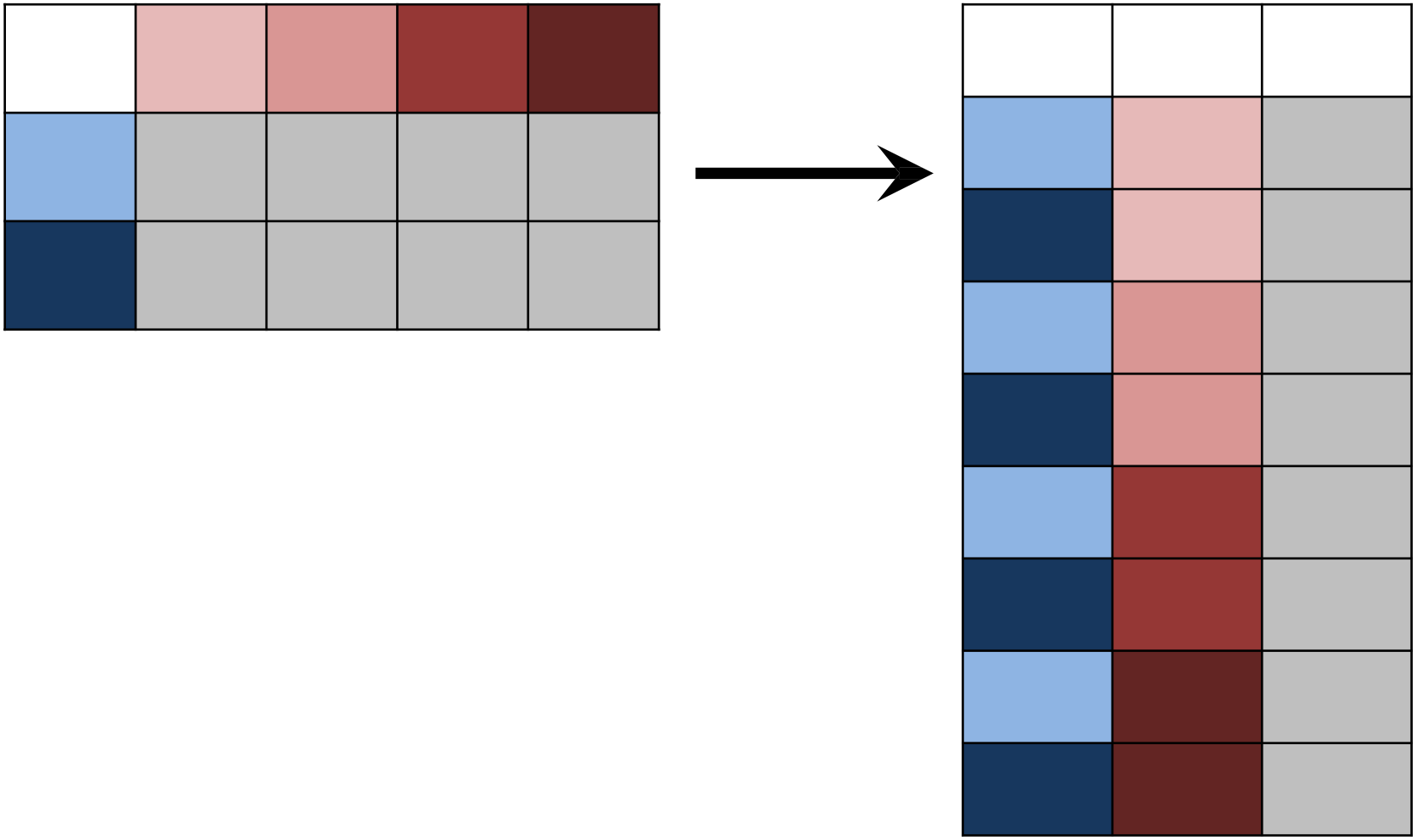
patient	treatment	outcome
1	drug	survived
2	drug	died
3	drug	survived
4	placebo	died
⋮		

Making data sets longer or wider

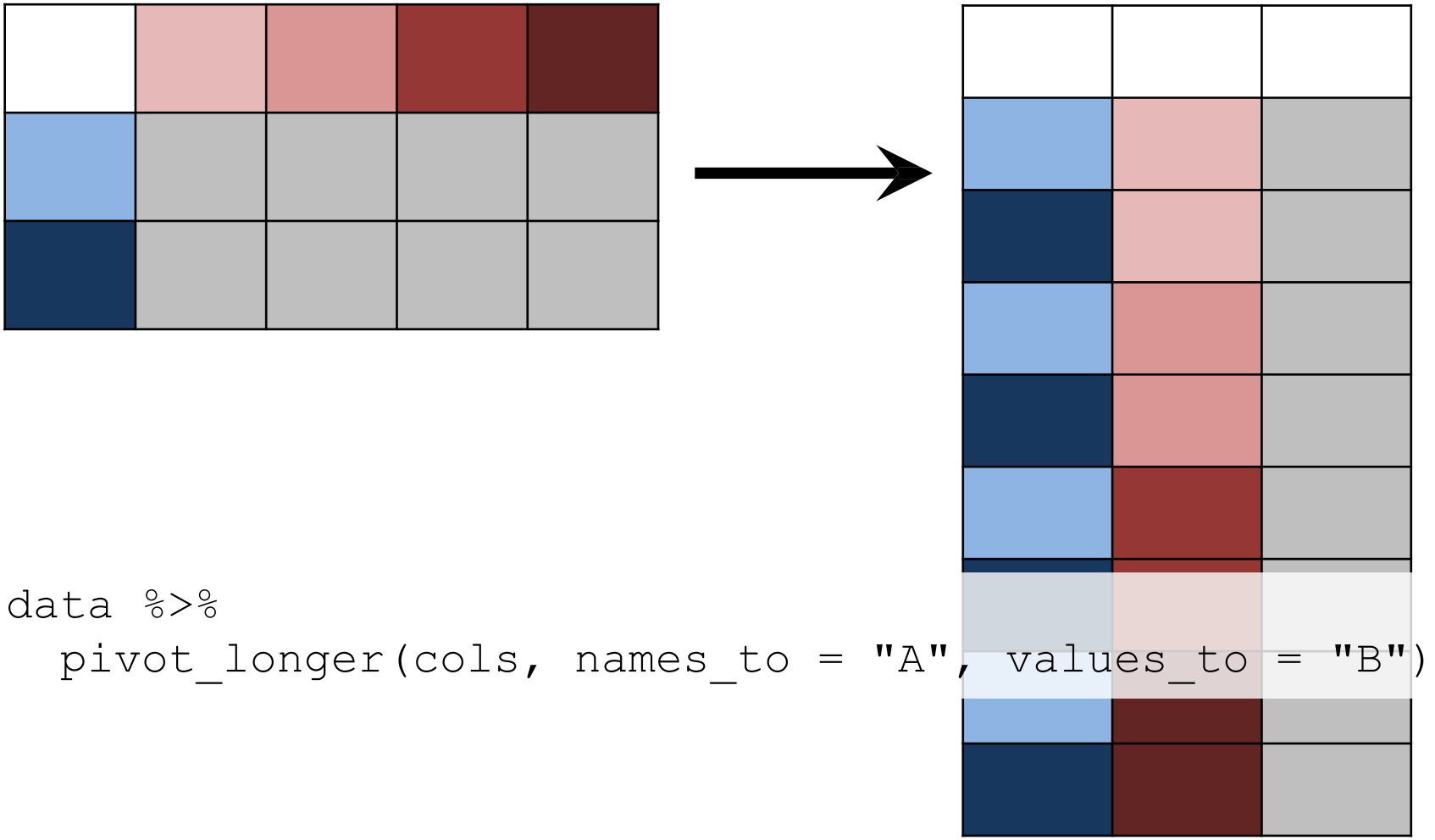
We'll be discussing two functions:

- `pivot_longer()` — make a wide table long
- `pivot_wider()` — make a long table wide

pivot_longer()

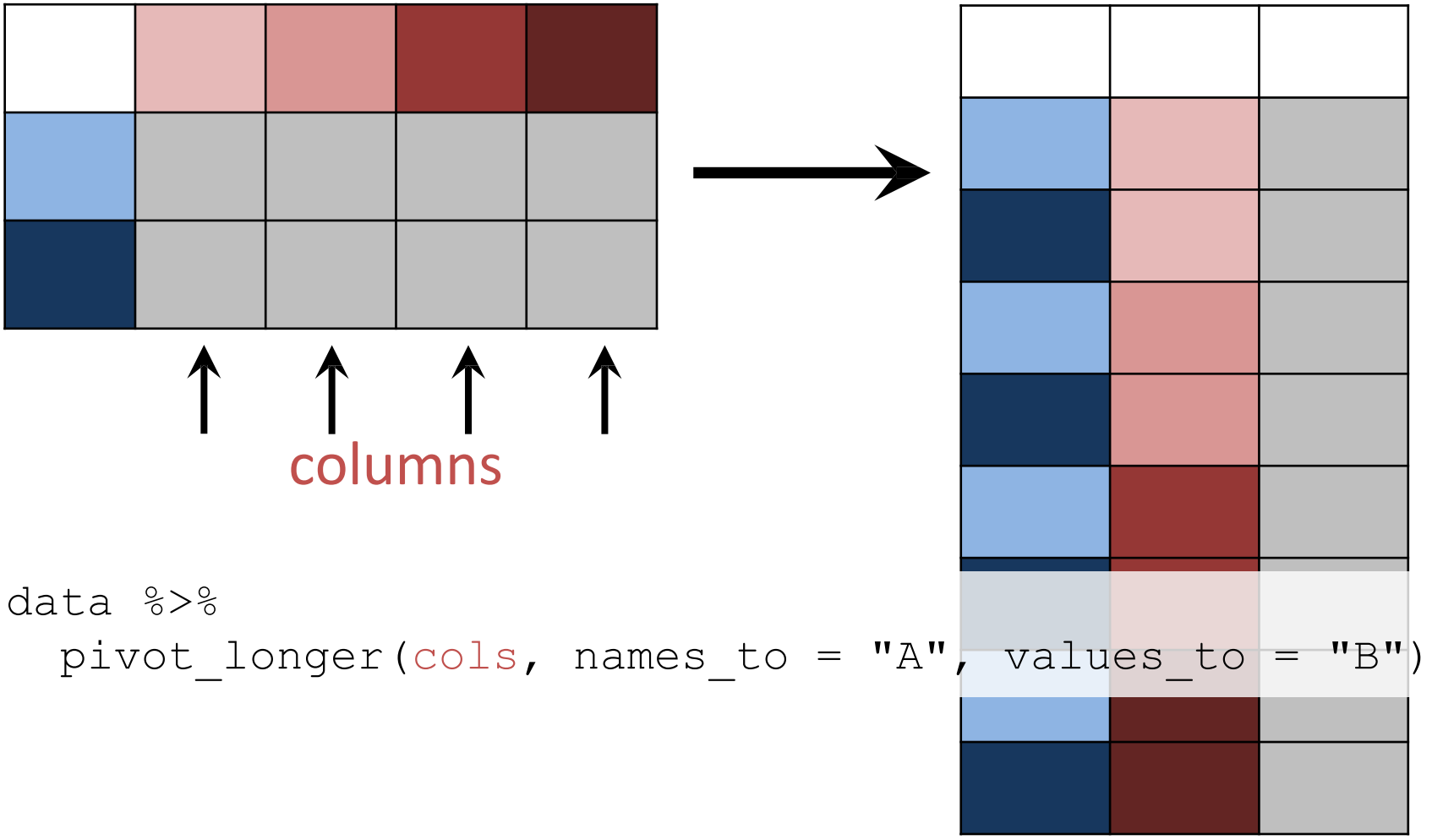


pivot_longer()



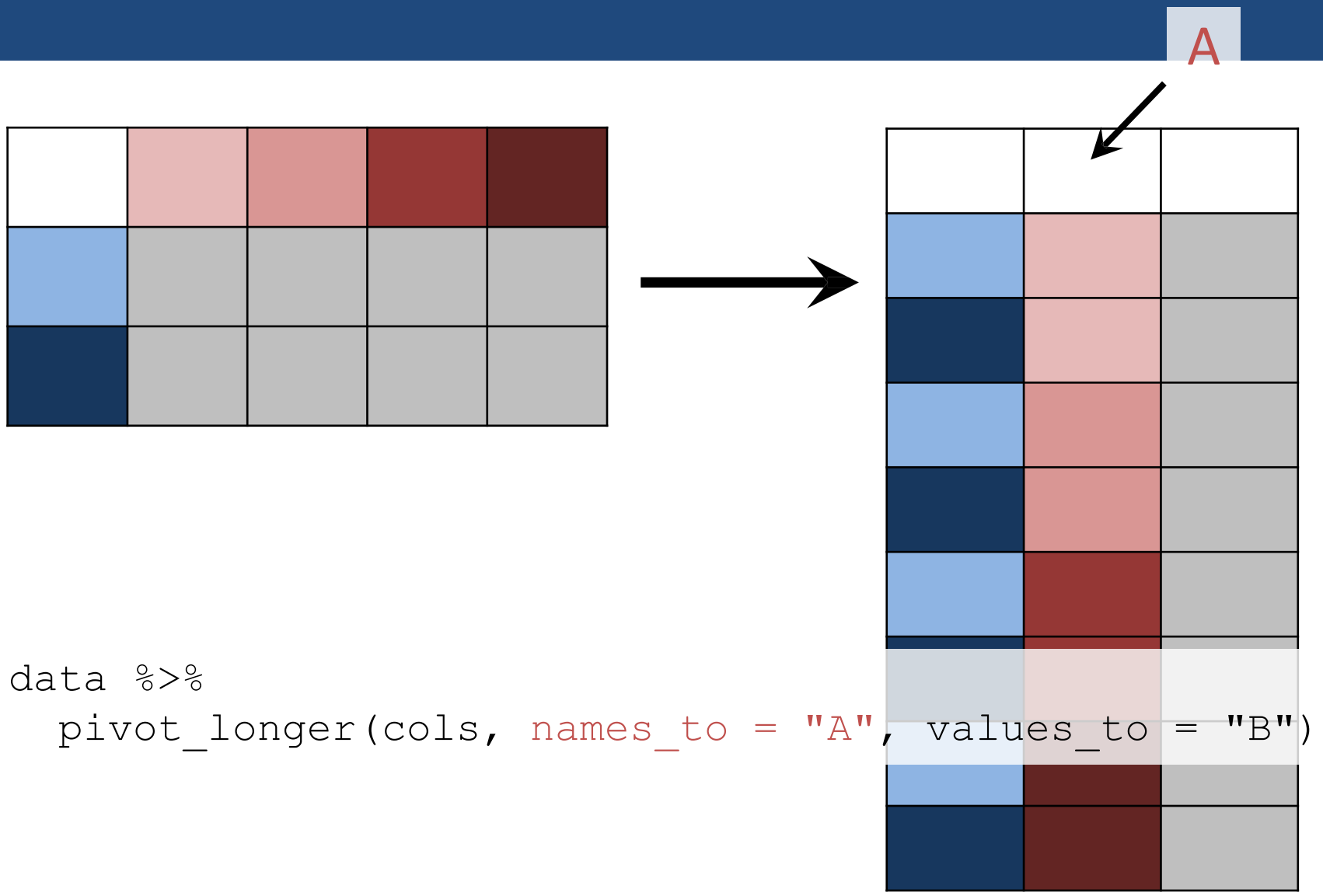
```
data %>%
  pivot_longer(cols, names_to = "A", values_to = "B")
```

pivot_longer()



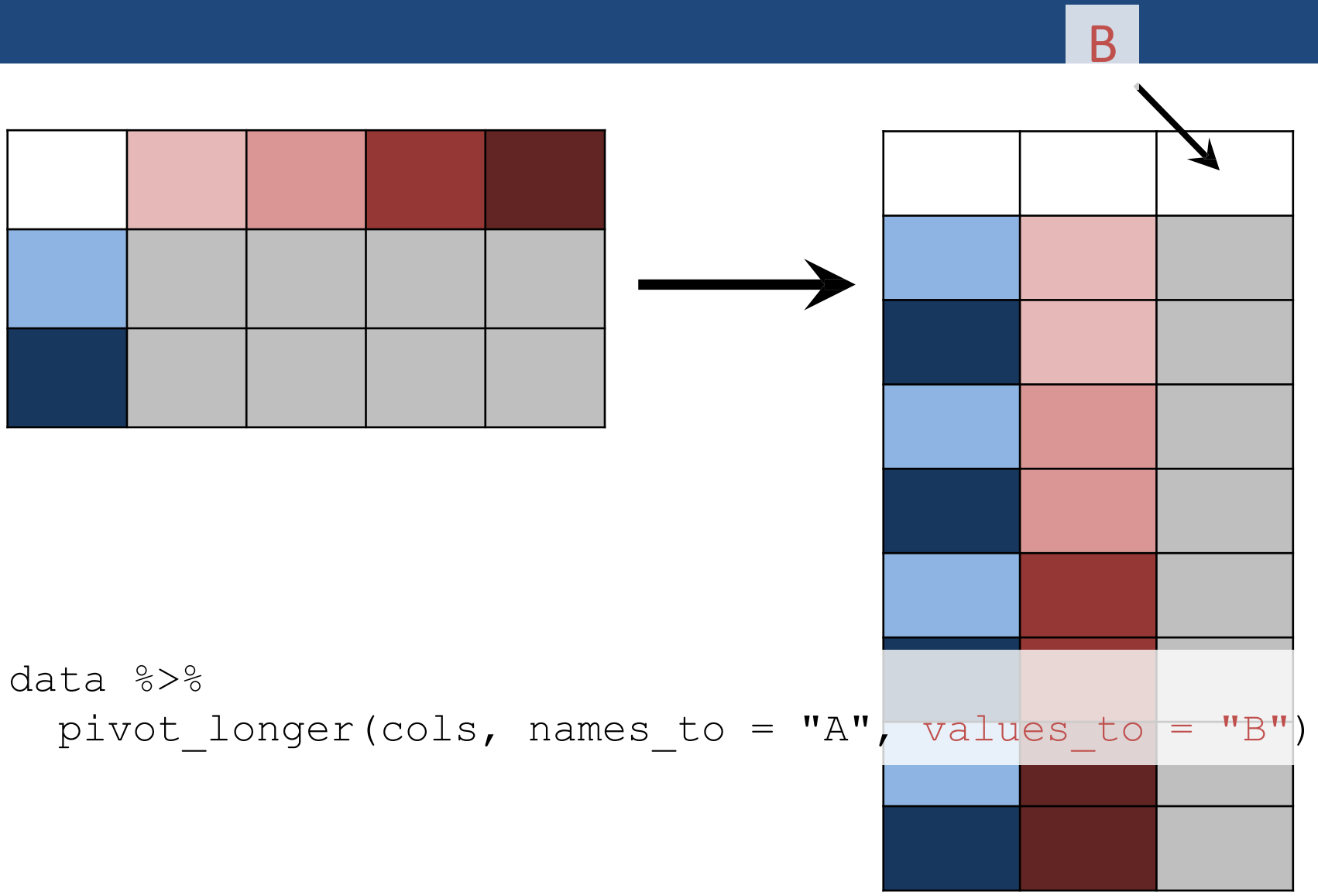

```
data %>%
  pivot_longer(cols, names_to = "A", values_to = "B")
```

pivot_longer()




```
data %>%
  pivot_longer(cols, names_to = "A", values_to = "B")
```

pivot_longer()




```
data %>%
  pivot_longer(cols, names_to = "A", values_to = "B")
```

Example dataset

```
> head(sitka_wide)
  tree treat t152 t174 t201 t227 t258
1    1 ozone 4.51 4.98 5.41 5.90 6.15
2    2 ozone 4.24 4.20 4.68 4.92 4.96
3    3 ozone 3.98 4.36 4.79 4.99 5.03
4    4 ozone 4.36 4.77 5.10 5.30 5.36
5    5 ozone 4.34 4.95 5.42 5.97 6.28
6    6 ozone 4.59 5.08 5.36 5.76 6.00
```

Is this data tidy? Why or why not?

Example: Tidying dataset using pivot_longer()

```
> head(sitka_wide)
  tree treat t152 t174 t201 t227 t258
1    1 ozone 4.51 4.98 5.41 5.90 6.15
2    2 ozone 4.24 4.20 4.68 4.92 4.96
3    3 ozone 3.98 4.36 4.79 4.99 5.03
4    4 ozone 4.36 4.77 5.10 5.30 5.36
5    5 ozone 4.34 4.95 5.42 5.97 6.28
6    6 ozone 4.59 5.08 5.36 5.76 6.00
```

Data Frame object

```
sitka_wide %>%
  pivot_longer(
    t152:t258, names_to = "time", values_to = "size"
  )
```

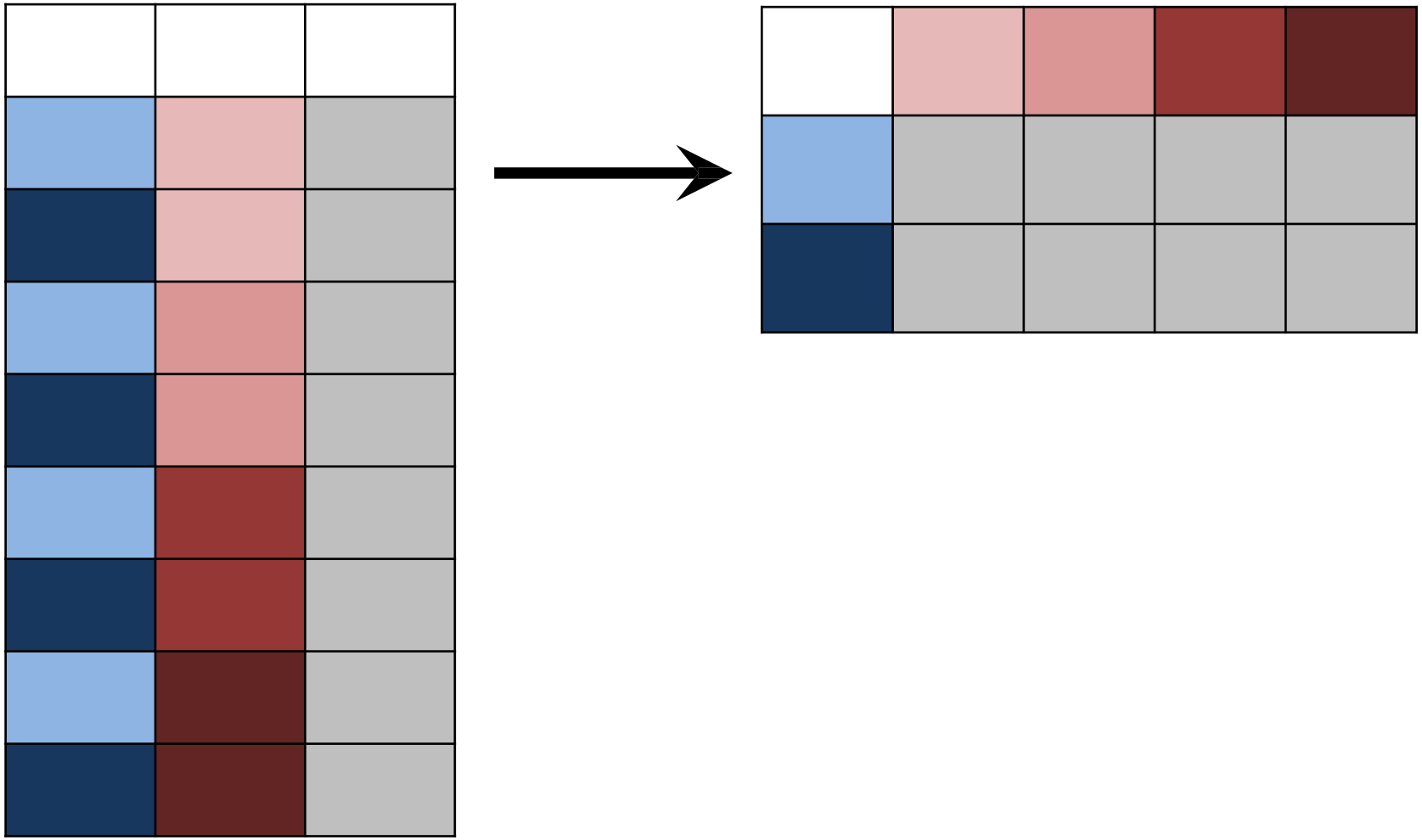
Pipe command (being "sent to")

`pivot_longer()` function

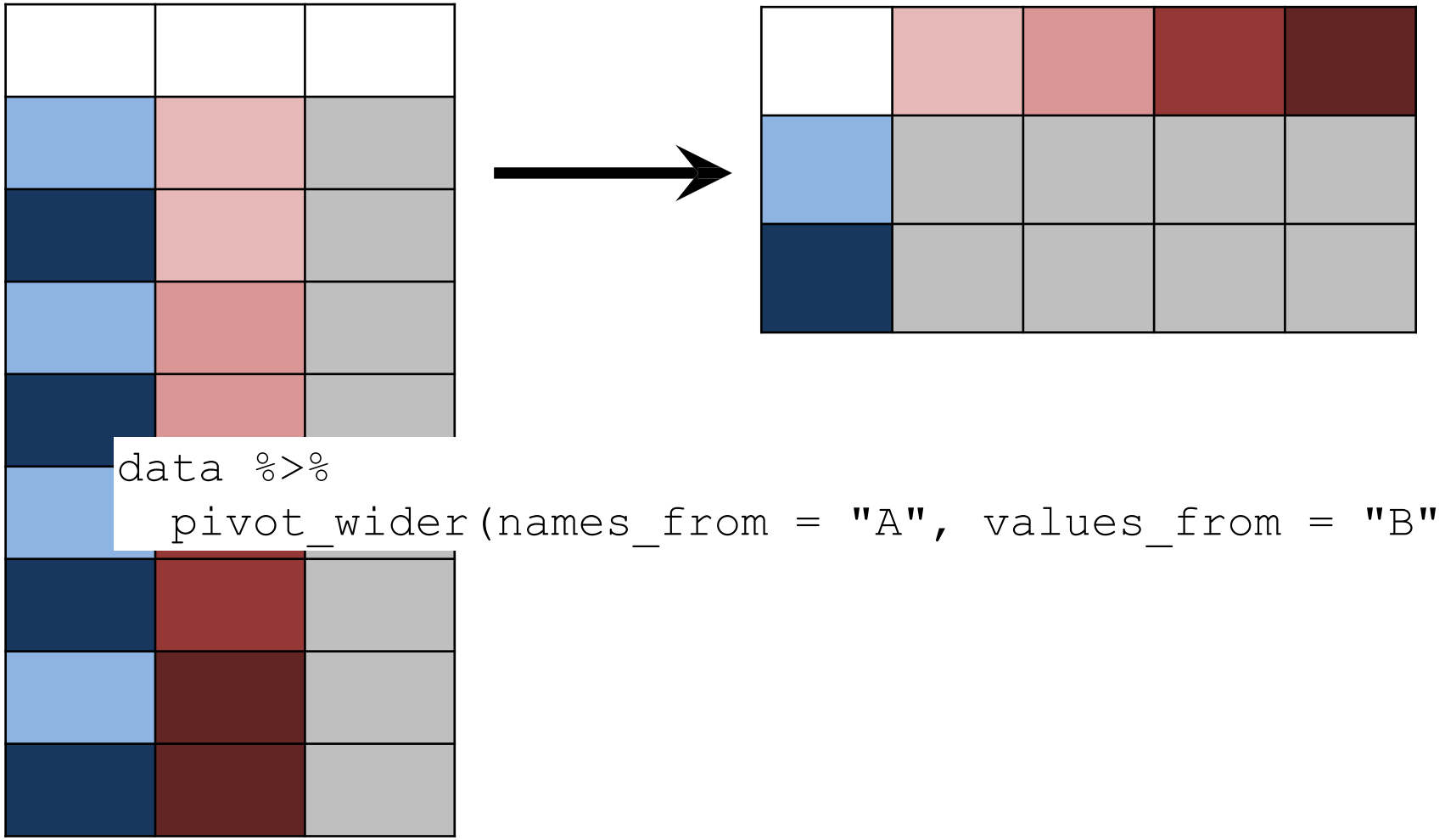
Example: Tidying dataset using pivot_longer()

```
> sitka_wide %>%
  pivot_longer(
    t152:t258, names_to = "time", values_to = "size"
  )
# A tibble: 395 x 4
    tree treat time   size
   <int> <fct> <chr> <dbl>
 1     1 ozone t152   4.51
 2     1 ozone t174   4.98
 3     1 ozone t201   5.41
 4     1 ozone t227   5.9
 5     1 ozone t258   6.15
 6     2 ozone t152   4.24
 7     2 ozone t174   4.2
 8     2 ozone t201   4.68
 9     2 ozone t227   4.92
10     2 ozone t258   4.96
# … with 385 more rows
```

pivot_wider()

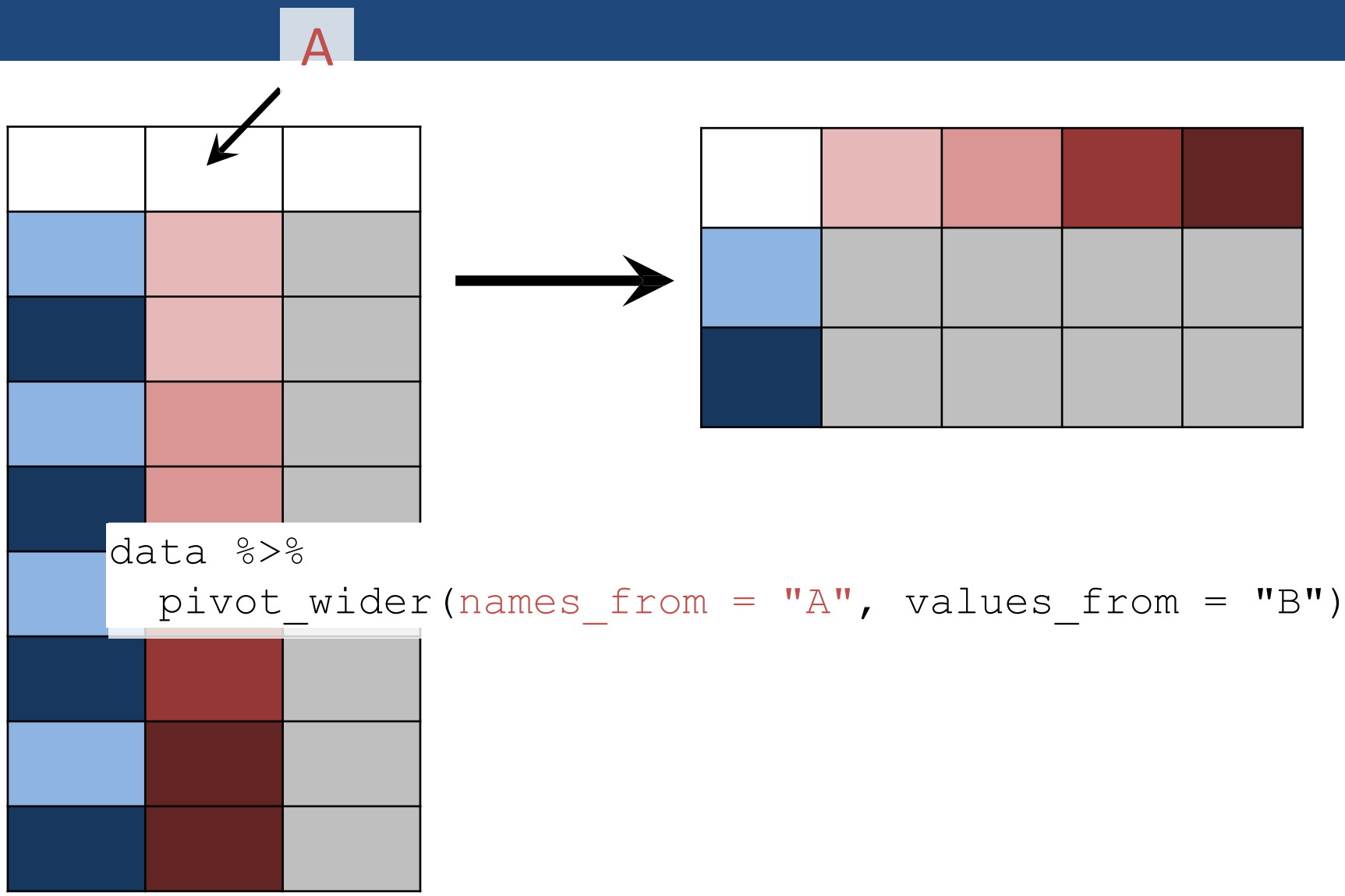


pivot_wider()



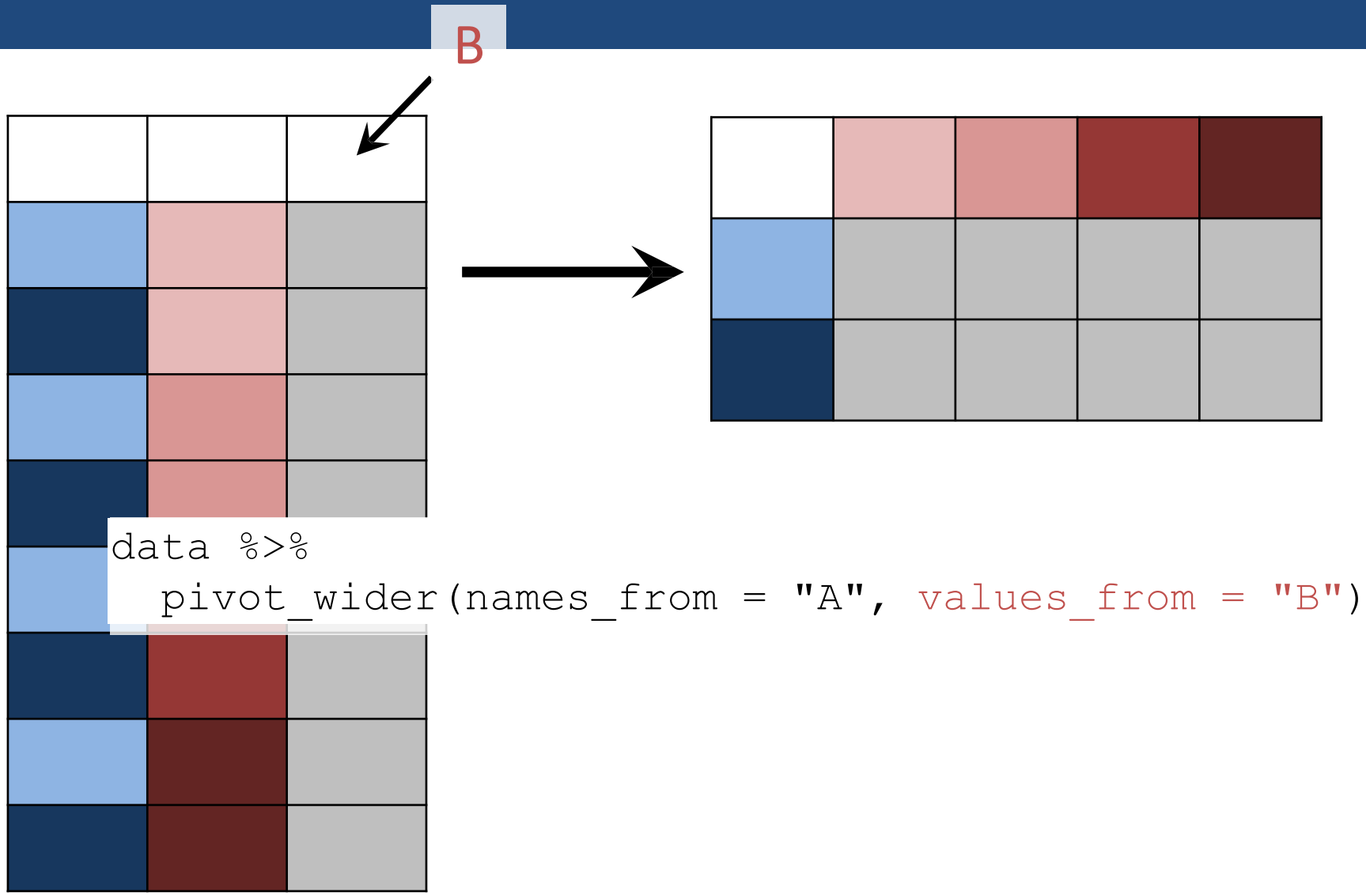
```
data %>%
  pivot_wider(names_from = "A", values_from = "B")
```

pivot_wider()




```
data %>%
  pivot_wider(names_from = "A", values_from = "B")
```

pivot_wider()




```
data %>%
  pivot_wider(names_from = "A", values_from = "B")
```

Example: Let's turn the sitka data into a wide table

```
> head(sitka)
   size Time tree treat
1 4.51   152    1 ozone
2 4.98   174    1 ozone
3 5.41   201    1 ozone
4 5.90   227    1 ozone
5 6.15   258    1 ozone
6 4.24   152    2 ozone
```

```
sitka %>%
  pivot_wider(names_from="Time", values_from="size")
```

Example: Let's turn the Sitka data into a wide table

```
> sitka %>%
  pivot_wider(names_from="Time", values_from="size")

# A tibble: 79 x 7
    tree treat `152` `174` `201` `227` `258`
   <int> <fct> <dbl> <dbl> <dbl> <dbl> <dbl>
 1     1 ozone  4.51  4.98  5.41  5.9   6.15
 2     2 ozone  4.24  4.2   4.68  4.92  4.96
 3     3 ozone  3.98  4.36  4.79  4.99  5.03
 4     4 ozone  4.36  4.77  5.1   5.3   5.36
 5     5 ozone  4.34  4.95  5.42  5.97  6.28
 6     6 ozone  4.59  5.08  5.36  5.76  6
 7     7 ozone  4.41  4.56  4.95  5.23  5.33
 8     8 ozone  4.24  4.64  4.95  5.38  5.48
 9     9 ozone  4.82  5.17  5.76  6.12  6.24
10    10 ozone  3.84  4.17  4.67  4.67  4.8
# … with 69 more rows
```

Working with tidy data in R: tidyverse

Fundamental actions on data tables:

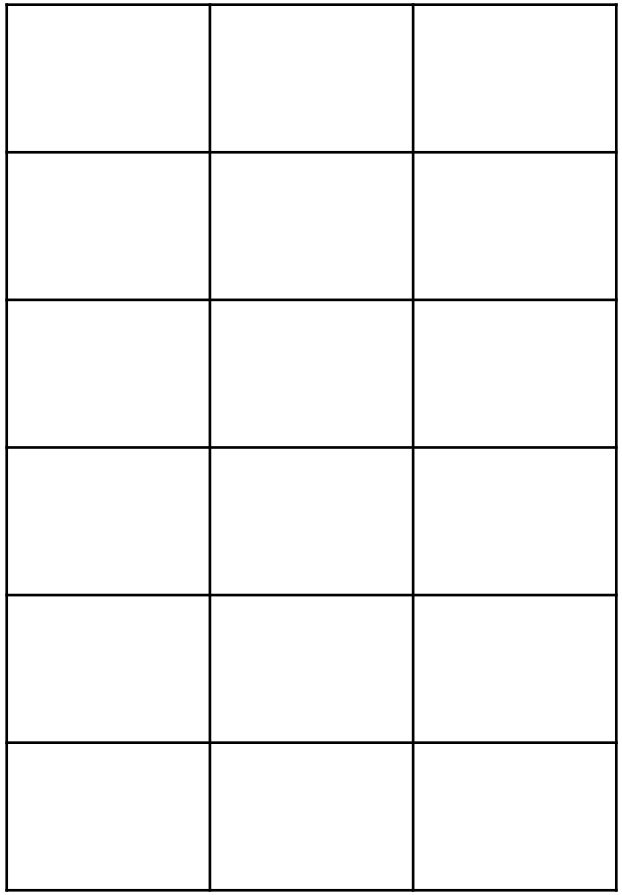
- make new columns — `mutate()`
- combine tables, adding columns — `left_join()`
- combine tables, adding rows — `bind_rows()`
- choose rows — `filter()`
- choose columns — `select()`
- arrange rows — `arrange()`
- calculate summary statistics — `summarize()`
- work on groups of data — `group_by()`

Working with tidy data in R: tidyverse

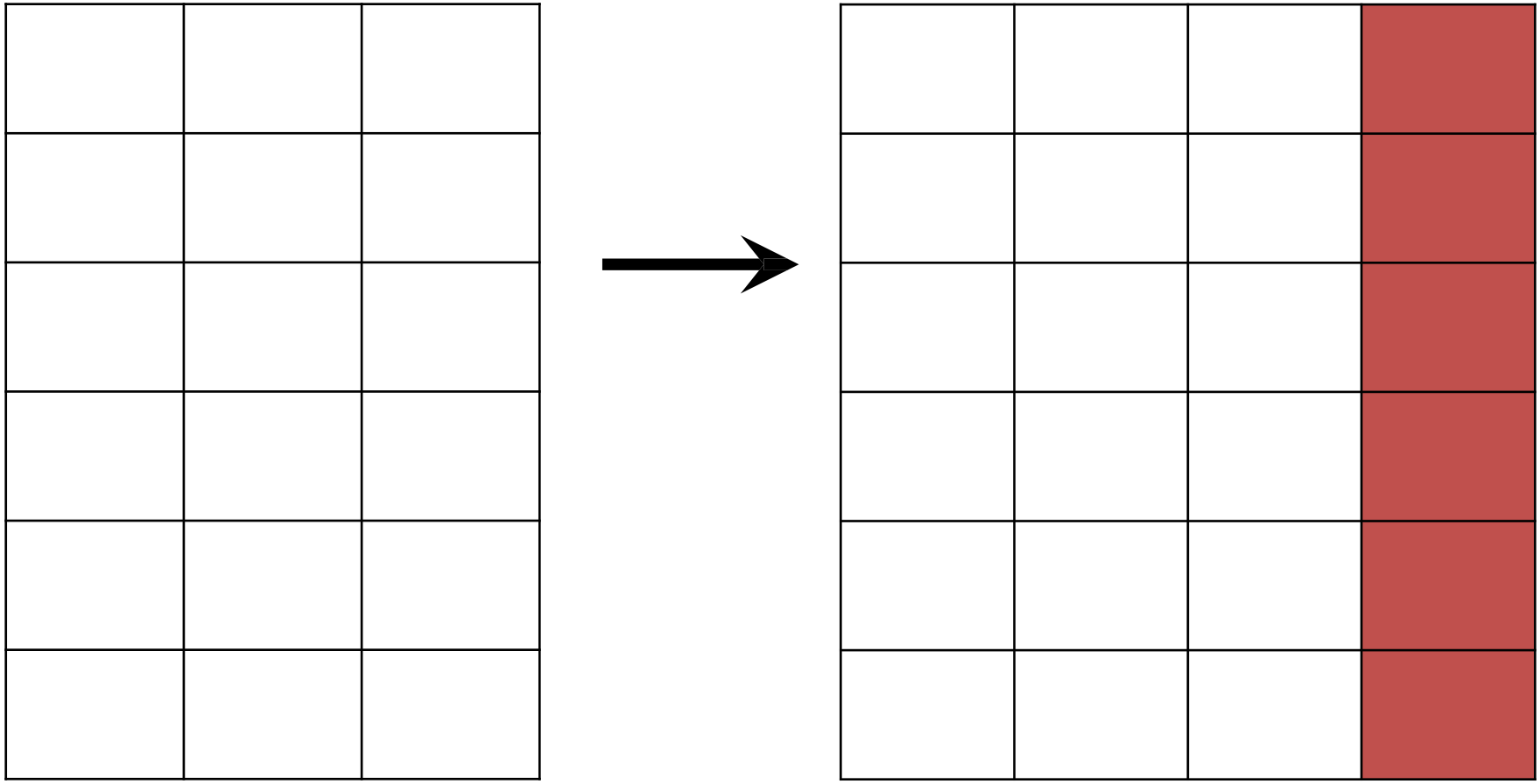
Fundamental actions on data tables:

- make new columns — `mutate()`
- combine tables, adding columns — `left_join()`
- combine tables, adding rows — `bind_rows()`
- choose rows — `filter()`
- choose columns — `select()`
- arrange rows — `arrange()`
- calculate summary statistics — `summarize()`
- work on groups of data — `group_by()`

`mutate()`: make new columns



`mutate()`: make new columns



Make new column with ratio of Sepal.Length to Sepal.Width

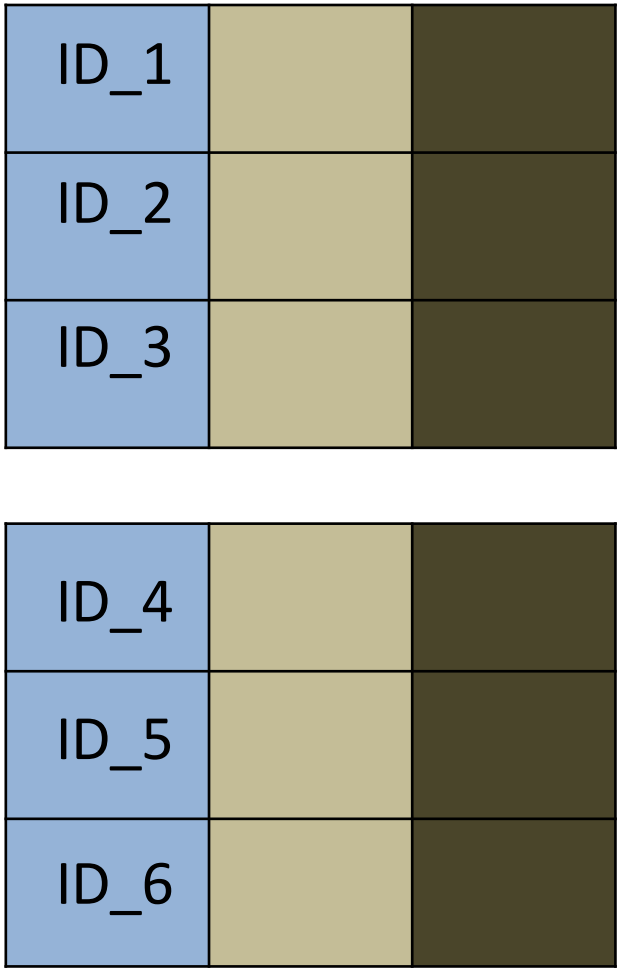
```
> mutate(iris, sepal_length_to_width = Sepal.Length/Sepal.Width)
```

Make new column with ratio of Sepal.Length to Sepal.Width

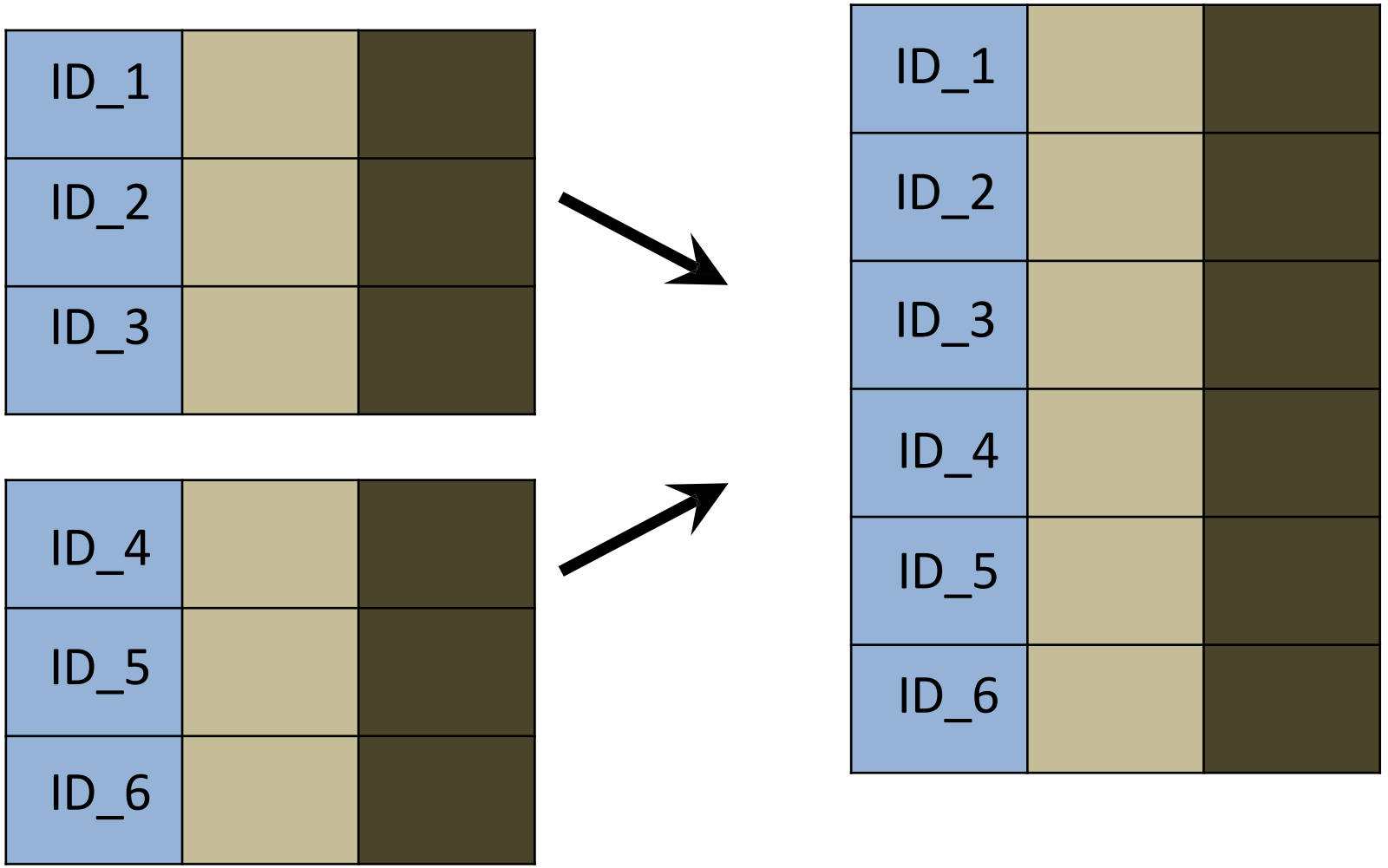
```
> mutate(iris, sepal_length_to_width = Sepal.Length/Sepal.Width)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	sepal_length_to_width
1	5.1	3.5	1.4	0.2	setosa	1.457143
2	4.9	3.0	1.4	0.2	setosa	1.633333
3	4.7	3.2	1.3	0.2	setosa	1.468750
4	4.6	3.1	1.5	0.2	setosa	1.483871
5	5.0	3.6	1.4	0.2	setosa	1.388889
6	5.4	3.9	1.7	0.4	setosa	1.384615
7	4.6	3.4	1.4	0.3	setosa	1.352941
8	5.0	3.4	1.5	0.2	setosa	1.470588
9	4.4	2.9	1.4	0.2	setosa	1.517241
10	4.9	3.1	1.5	0.1	setosa	1.580645
11	5.4	3.7	1.5	0.2	setosa	1.459459
12	4.8	3.4	1.6	0.2	setosa	1.411765
13	4.8	3.0	1.4	0.1	setosa	1.600000
14	4.3	3.0	1.1	0.1	setosa	1.433333
15	5.8	4.0	1.2	0.2	setosa	1.450000
16	5.7	4.4	1.5	0.4	setosa	1.295455
17	5.4	3.9	1.3	0.4	setosa	1.384615
18	5.1	3.5	1.4	0.3	setosa	1.457143
19	5.7	3.8	1.7	0.3	setosa	1.500000
20	5.1	3.8	1.5	0.3	setosa	1.342105

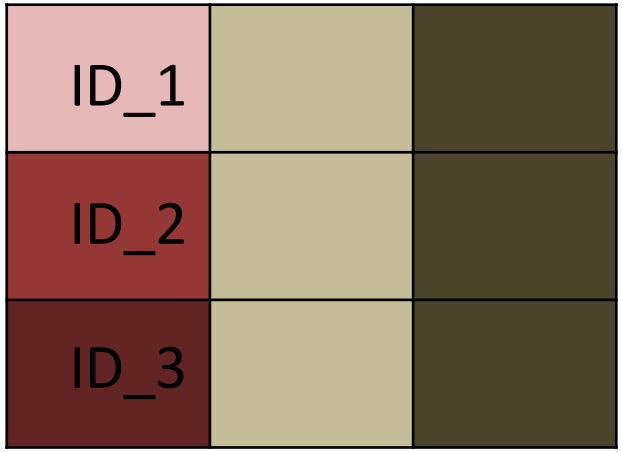
rbind() or bind_rows()

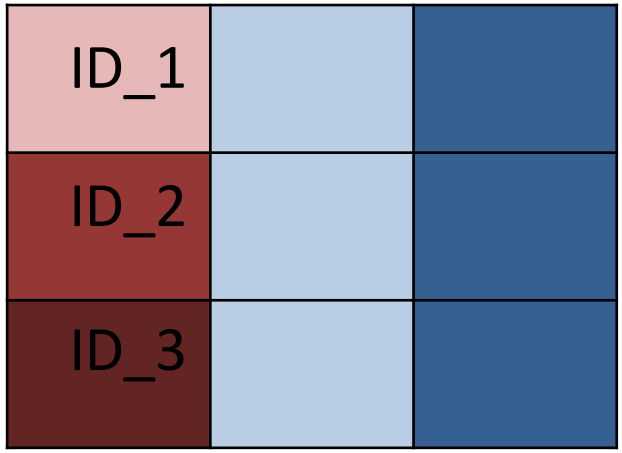


`rbind()` or `bind_rows()`: Stack tables

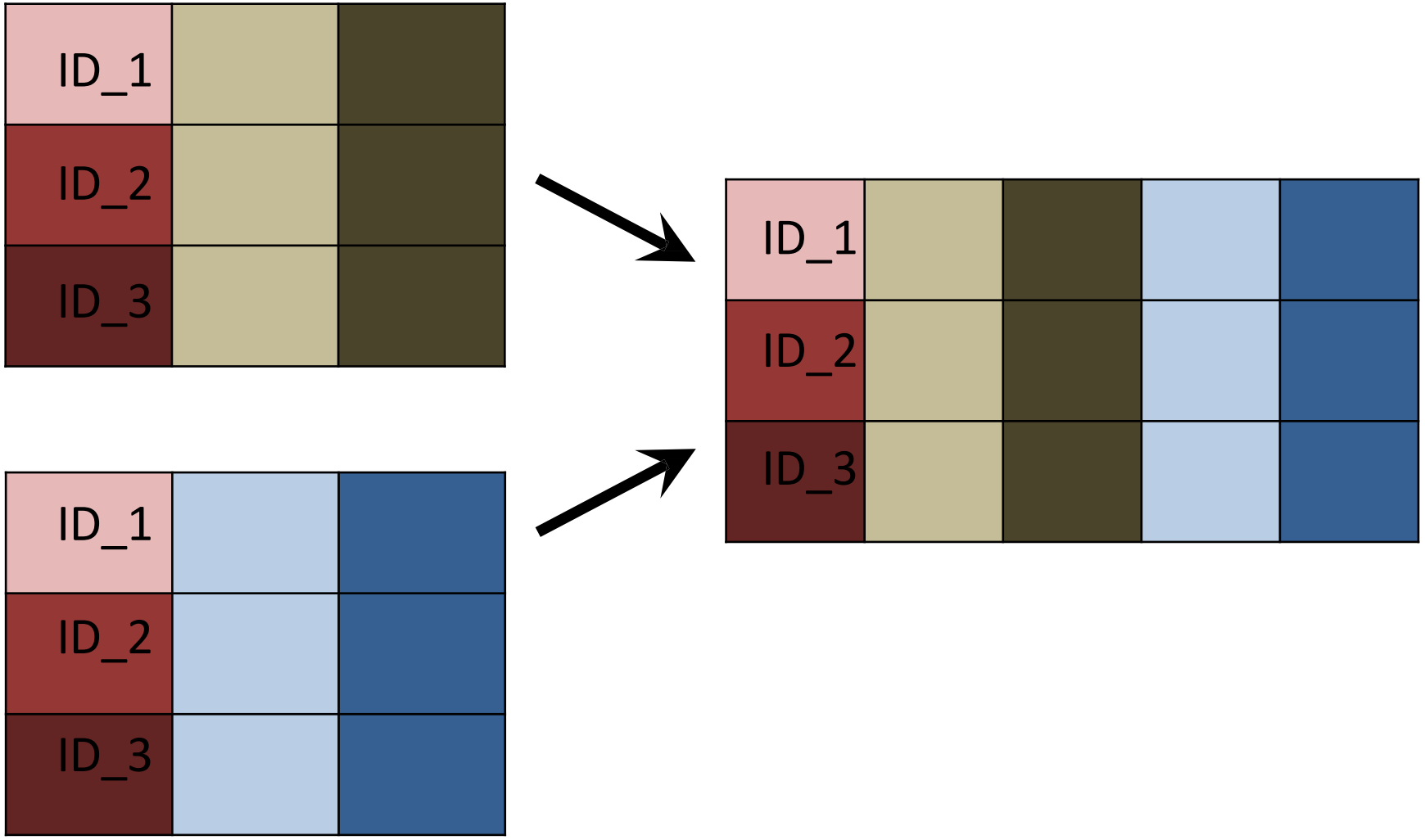


`left_join()`: combine two tables

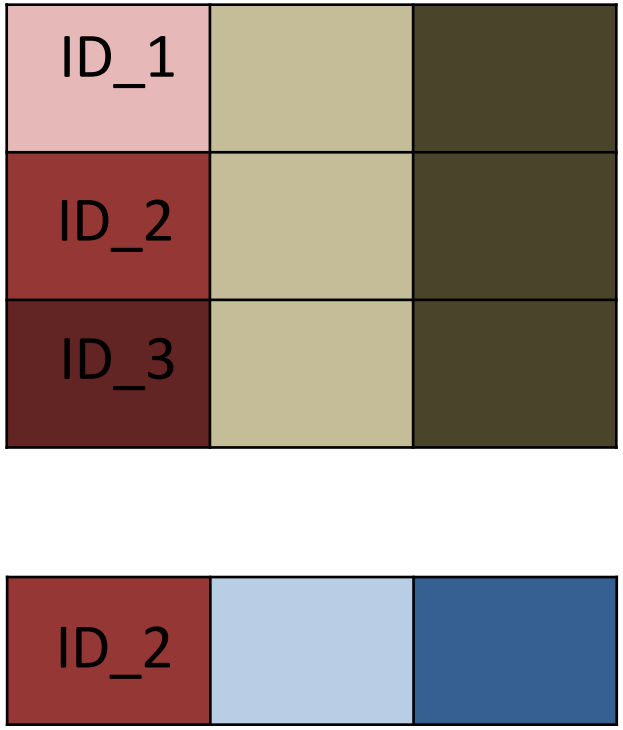




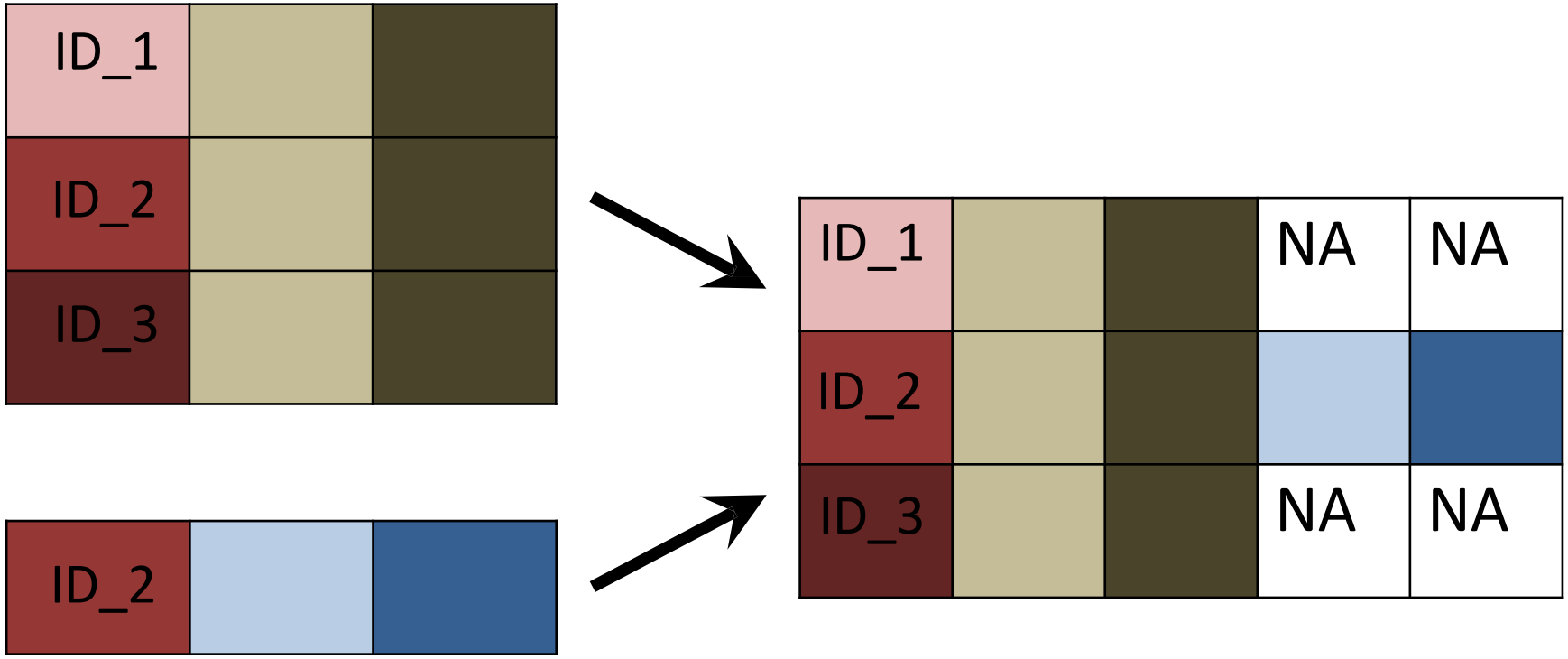
`left_join()`: combine two tables



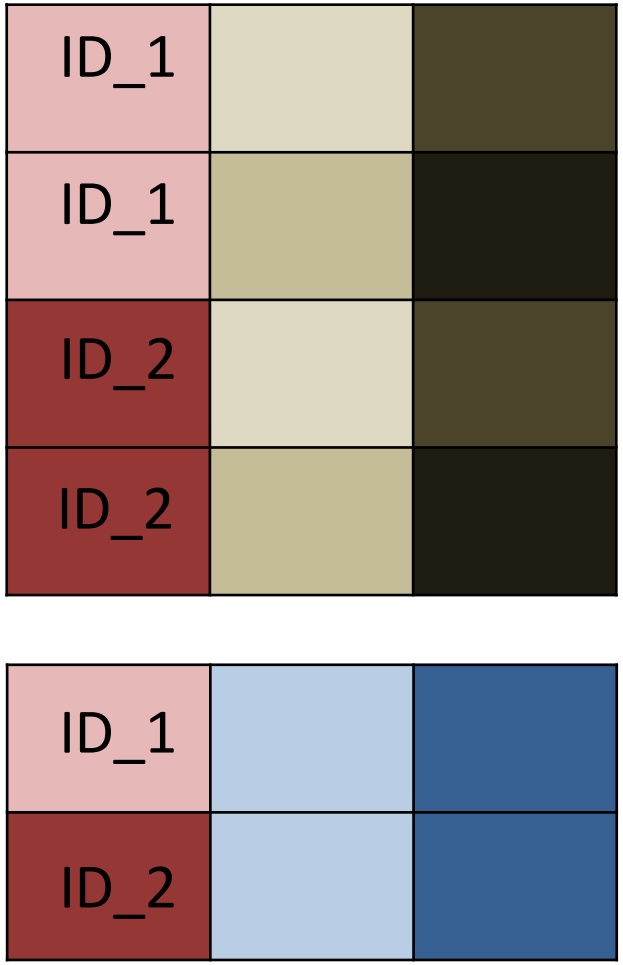
`left_join()`: missing values in 2nd table are set to NA



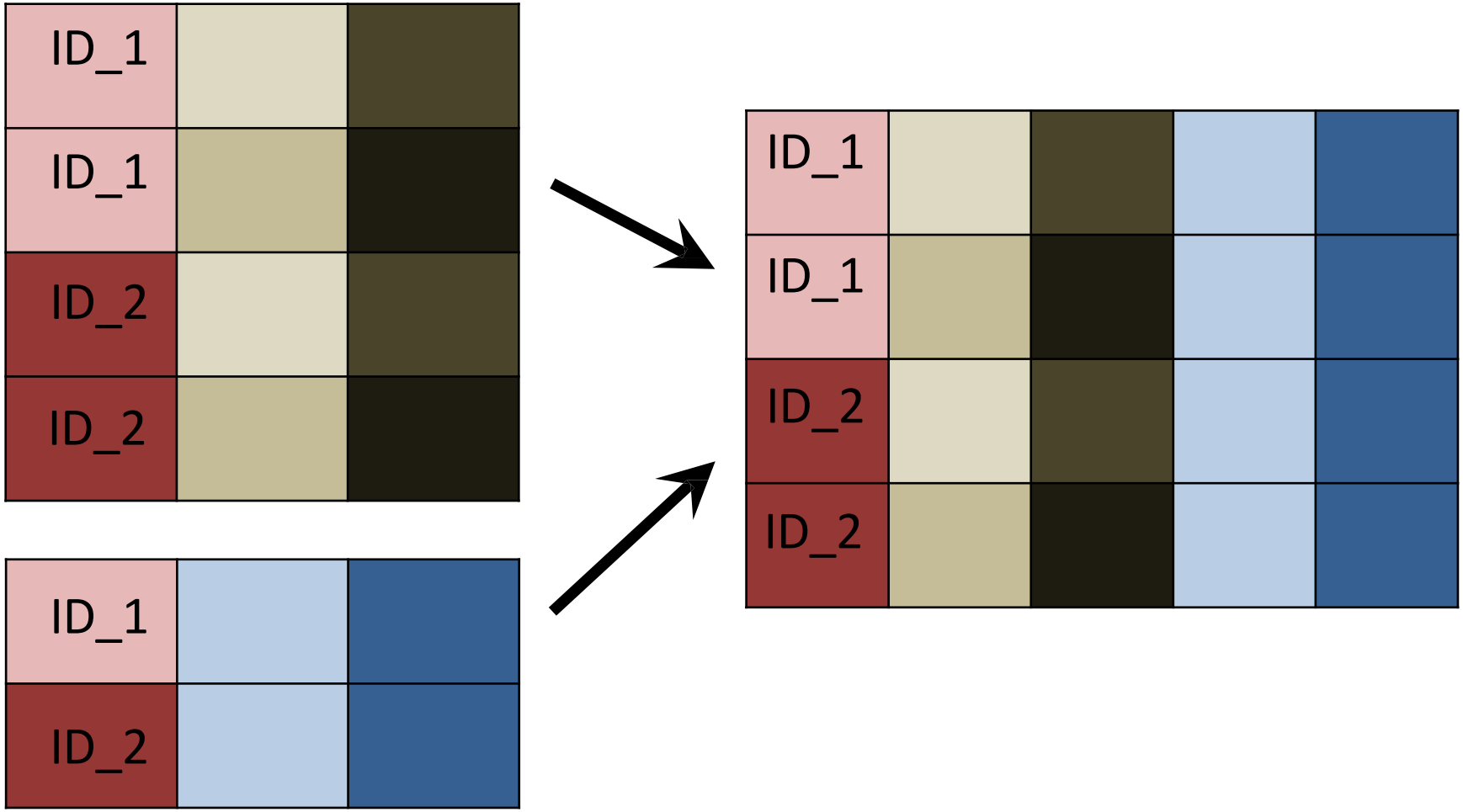
`left_join()`: missing values in 2nd table are set to NA



`left_join()`: values from 2nd table are duplicated where necessary



`left_join()`: values from 2nd table are duplicated where necessary



Example: Joining tables

Let’s extract two tables from msleep:

Example: Joining tables

Let's extract two tables from msleep:

```
> order_table <- select(msleep, name, order)
> order_table
                                  name           order
1                              Cheetah       Carnivora
2                           Owl monkey        Primates
3                        Mountain beaver        Rodentia
4      Greater short-tailed shrew    Soricomorpha
5                                  Cow    Artiodactyla
6                    Three-toed sloth          Pilosa
7                   Northern fur seal       Carnivora
8                        Vesper mouse        Rodentia
9                                  Dog       Carnivora
10                          Roe deer    Artiodactyla
```

Example: Joining tables

Let's extract two tables from msleep:

```
> awake_table <- select(msleep, name, awake)
> awake_table
                                  name awake
1                              Cheetah 11.90
2                           Owl monkey  7.00
3                       Mountain beaver  9.60
4      Greater short-tailed shrew  9.10
5                                  Cow 20.00
6                    Three-toed sloth  9.60
7                  Northern fur seal 15.30
8                        Vesper mouse 17.00
9                                  Dog 13.90
10                            Roe deer 21.00
```

Example: Joining tables

And put them back together:

```
> left_join(order_table, awake_table)
```

Example: Joining tables

And put them back together:

```
> left_join(order_table, awake_table)
Joining by: "name"
                                  name          order awake
1                              Cheetah      Carnivora 11.90
2                           Owl monkey       Primates  7.00
3                       Mountain beaver       Rodentia  9.60
4          Greater short-tailed shrew  Soricomorpha  9.10
5                                  Cow  Artiodactyla 20.00
6                     Three-toed sloth         Pilosa  9.60
7                    Northern fur seal      Carnivora 15.30
8                         Vesper mouse       Rodentia 17.00
9                                  Dog      Carnivora 13.90
10                            Roe deer  Artiodactyla 21.00
```

Several different join functions are available

- `left_join()`
- `right_join()`
- `inner_join()`
- `semi_join()`
- `full_join()`
- `anti_join()`

Demonstration Time!

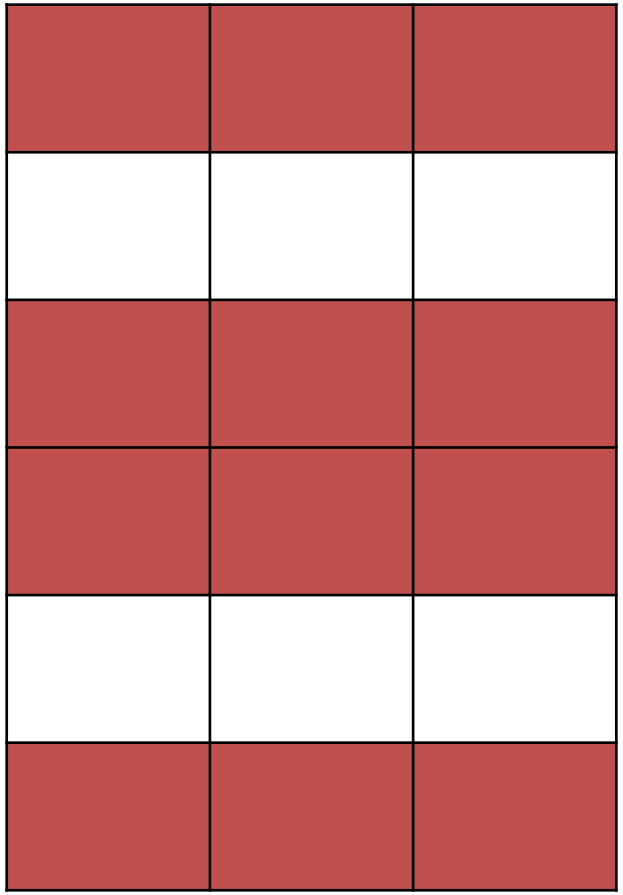
Refer to workbook Section 5

Working with tidy data in R: tidyverse

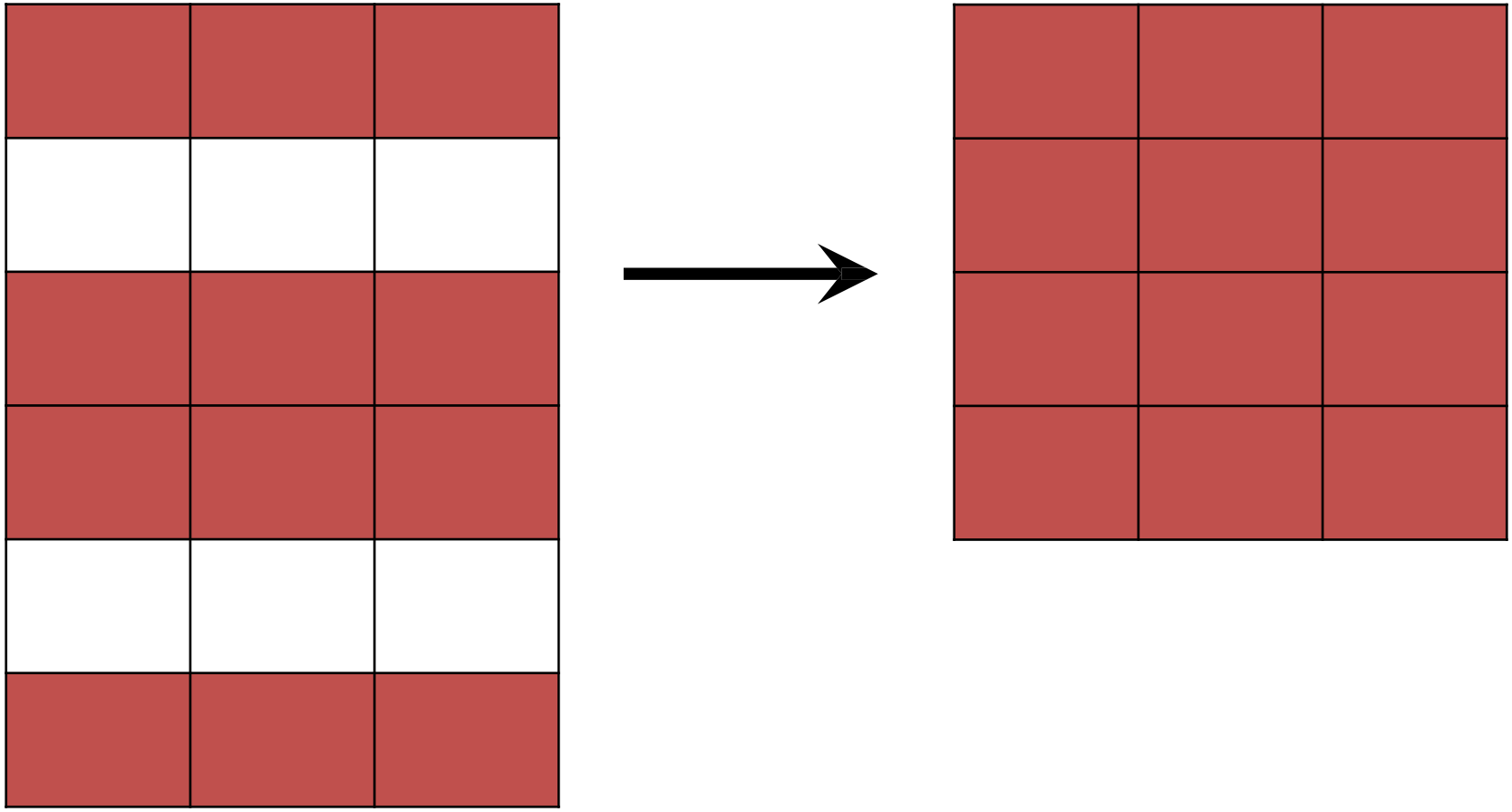
Fundamental actions on data tables:

- make new columns — `mutate()`
- combine tables, adding columns — `left_join()`
- combine tables, adding rows — `bind_rows()`
- choose rows — `filter()`
- choose columns — `select()`
- arrange rows — `arrange()`
- calculate summary statistics — `summarize()`
- work on groups of data — `group_by()`

`filter()`: pick rows



`filter()`: pick rows



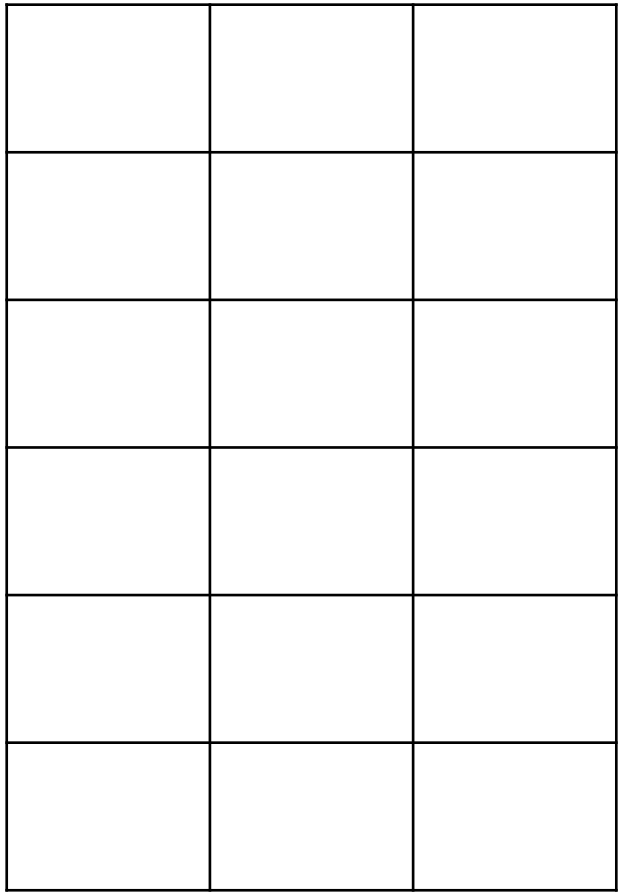
Choose rows with Sepal.Width > 4

```
> filter(iris, Sepal.Width > 4)
```

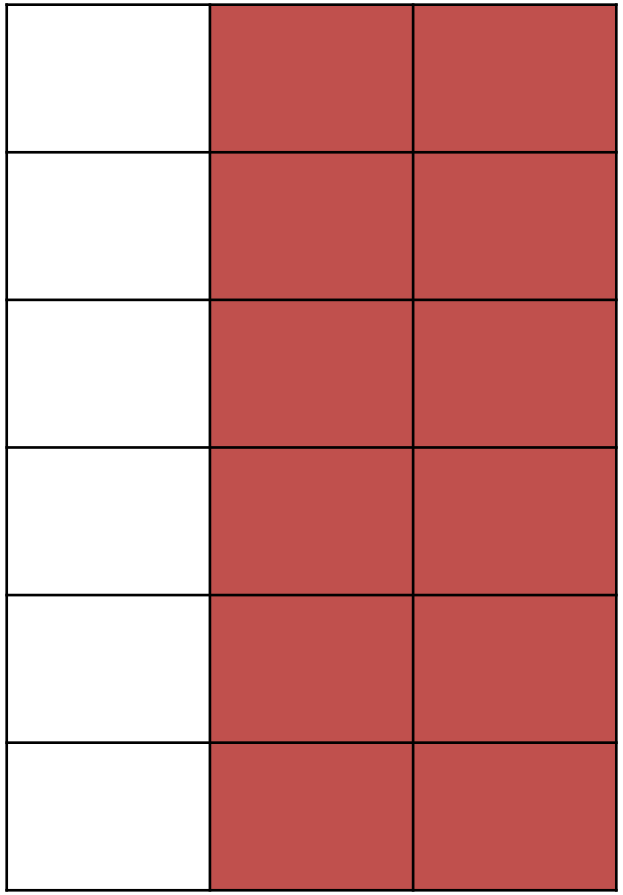
Choose rows with Sepal.Width > 4

```
> filter(iris, Sepal.Width > 4)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.7         4.4          1.5         0.4  setosa
2          5.2         4.1          1.5         0.1  setosa
3          5.5         4.2          1.4         0.2  setosa
```

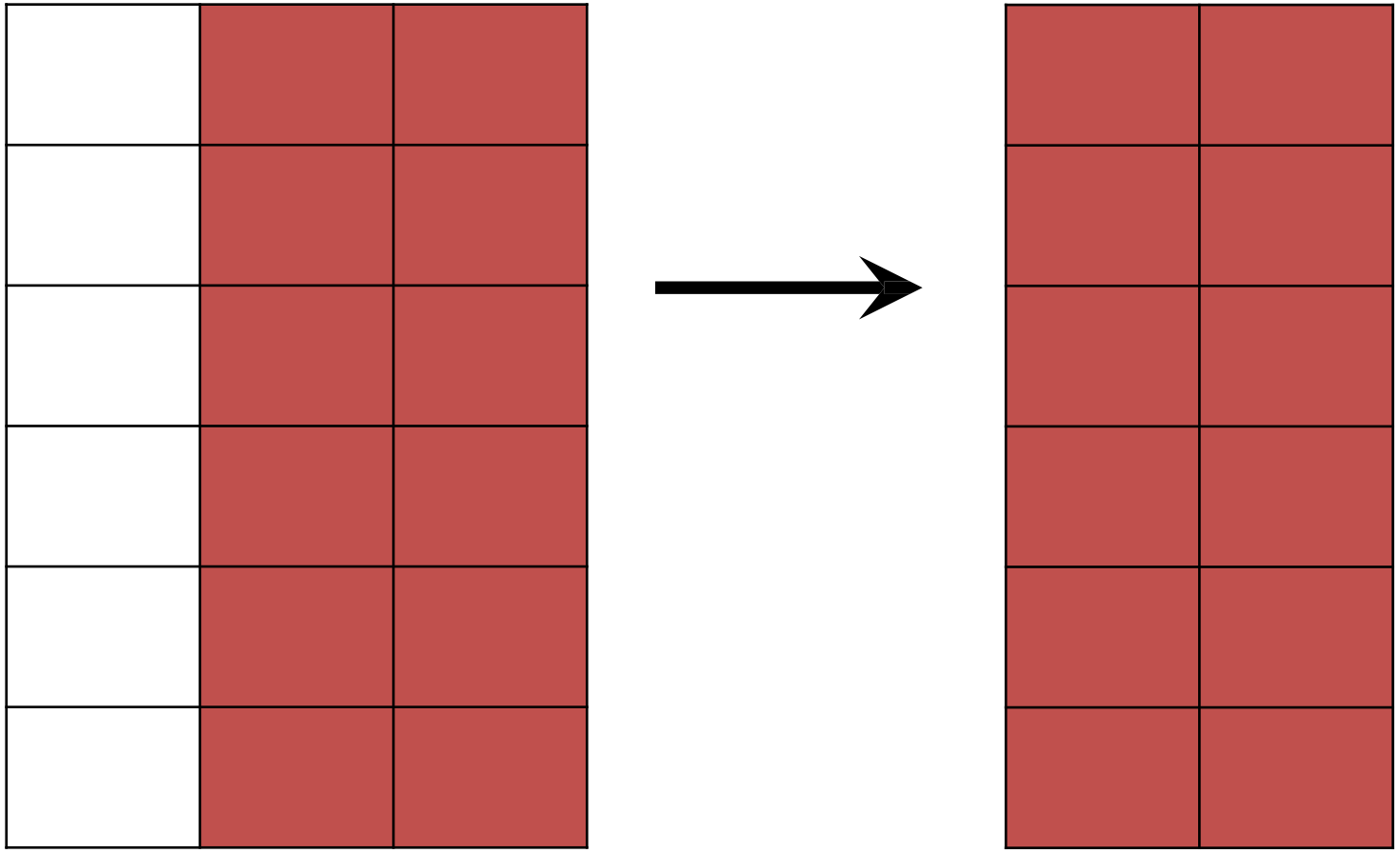
`select()`: pick columns



`select()`: pick columns



`select()`: pick columns



Choose the two columns Species and Sepal.Width

```
> select(iris, Species, Sepal.Width)
```

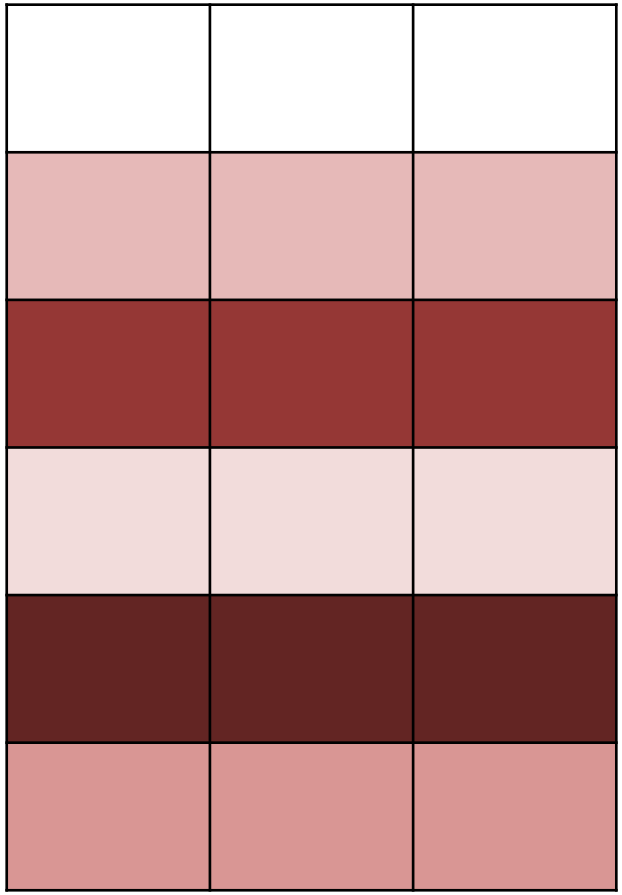
Choose the two columns Species and Sepal.Width

```
> select(iris, Species, Sepal.Width)
       Species Sepal.Width
1       setosa         3.5
2       setosa         3.0
3       setosa         3.2
4       setosa         3.1
5       setosa         3.6
6       setosa         3.9
7       setosa         3.4
8       setosa         3.4
9       setosa         2.9
10      setosa         3.1
11      setosa         3.7
12      setosa         3.4
13      setosa         3.0
14      setosa         3.0
```

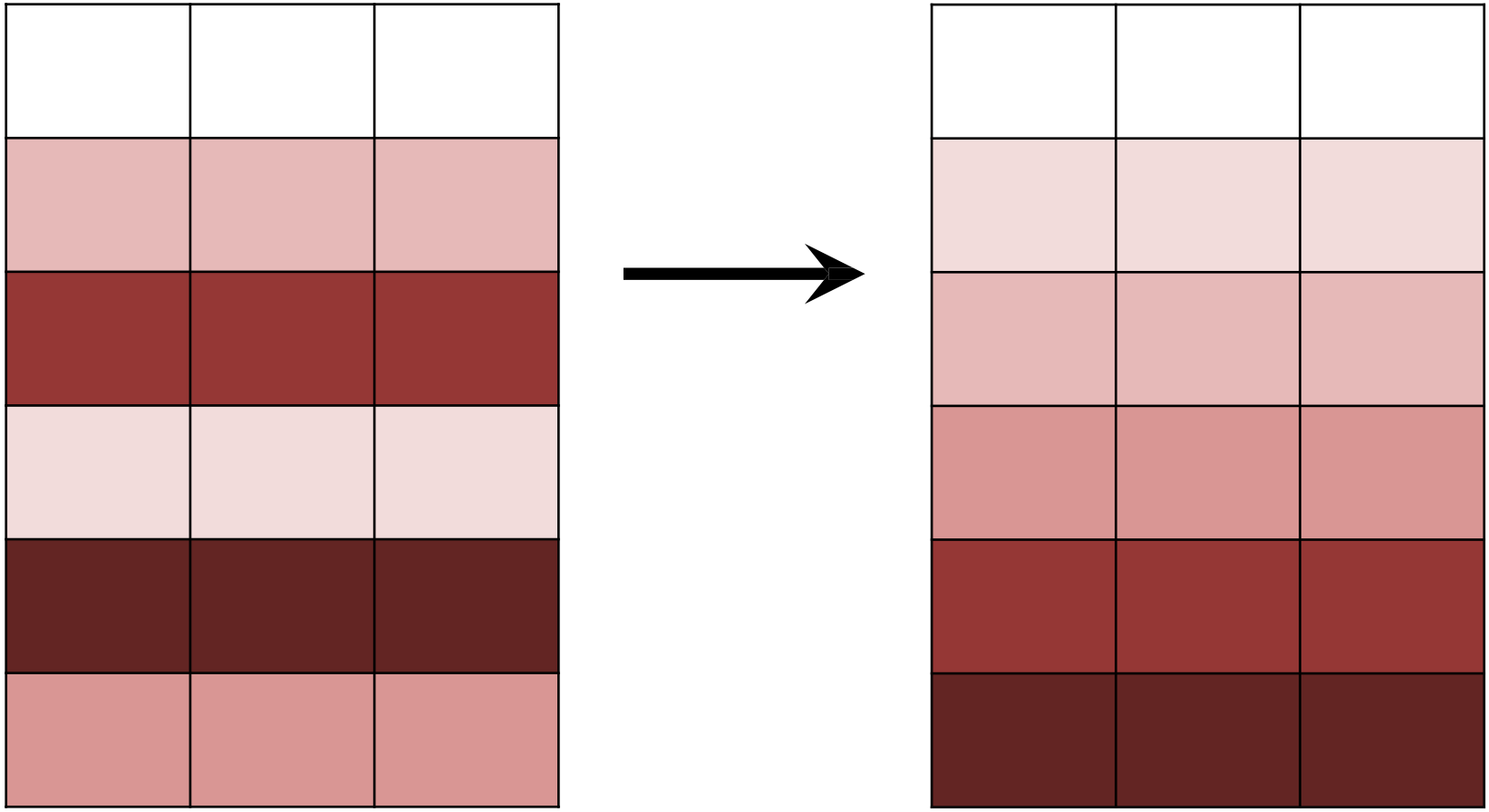
Demonstration Time!

Refer to Section #6

`arrange()`: change row order



`arrange()`: change row order



Sort by increasing order of Sepal.Width

```
> arrange(iris, Sepal.Width)
```

Sort by increasing order of Sepal.Width

```
> arrange(iris, Sepal.Width)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1             5.          2.0          3.5         1.0 versicolor
2             6.          2.2          4.0         1.0 versicolor
3             6.2         2.2          4.5         1.5 versicolor
4             6.0         2.2          5.0         1.5  virginica
5             4.5         2.3          1.3         0.3     setosa
6             5.5         2.3          4.0         1.3 versicolor
7             6.3         2.3          4.4         1.3 versicolor
8             5.0         2.3          3.3         1.0 versicolor
9             4.9         2.4          3.3         1.0 versicolor
10            5.5         2.4          3.8         1.1 versicolor
11            5.          2.4          3.7         1.0 versicolor
12            6.          2.5          3.9         1.1 versicolor
```

Sort by decreasing order of Sepal.Length

```
> arrange(iris, desc(Sepal.Length))
```

Sort by decreasing order of Sepal.Length

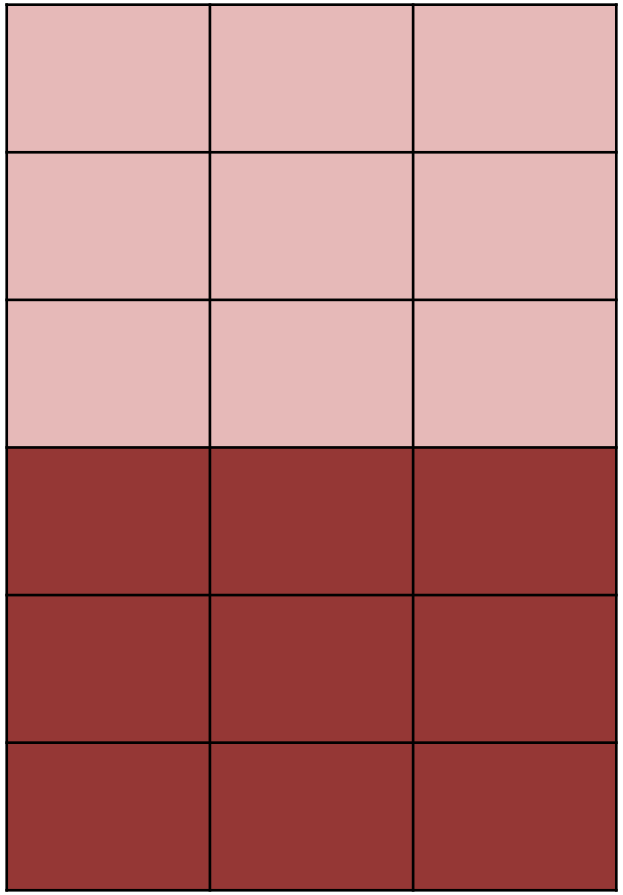
```
> arrange(iris, desc(Sepal.Length))
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1           7.9         3.8          6.4         2.0  virginica
2           7.7         3.8          6.7         2.2  virginica
3           7.7         2.6          6.9         2.3  virginica
4           7.7         2.8          6.7         2.0  virginica
5           7.7         3.0          6.1         2.3  virginica
6           7.6         3.0          6.6         2.1  virginica
7           7.4         2.8          6.1         1.9  virginica
8           7.3         2.9          6.3         1.8  virginica
9           7.2         3.6          6.1         2.5  virginica
10          7.2         3.2          6.0         1.8  virginica
11          7.2         3.0          5.8         1.6  virginica
12          7.1         3.0          5.9         2.1  virginica
```

Working with tidy data in R: tidyverse

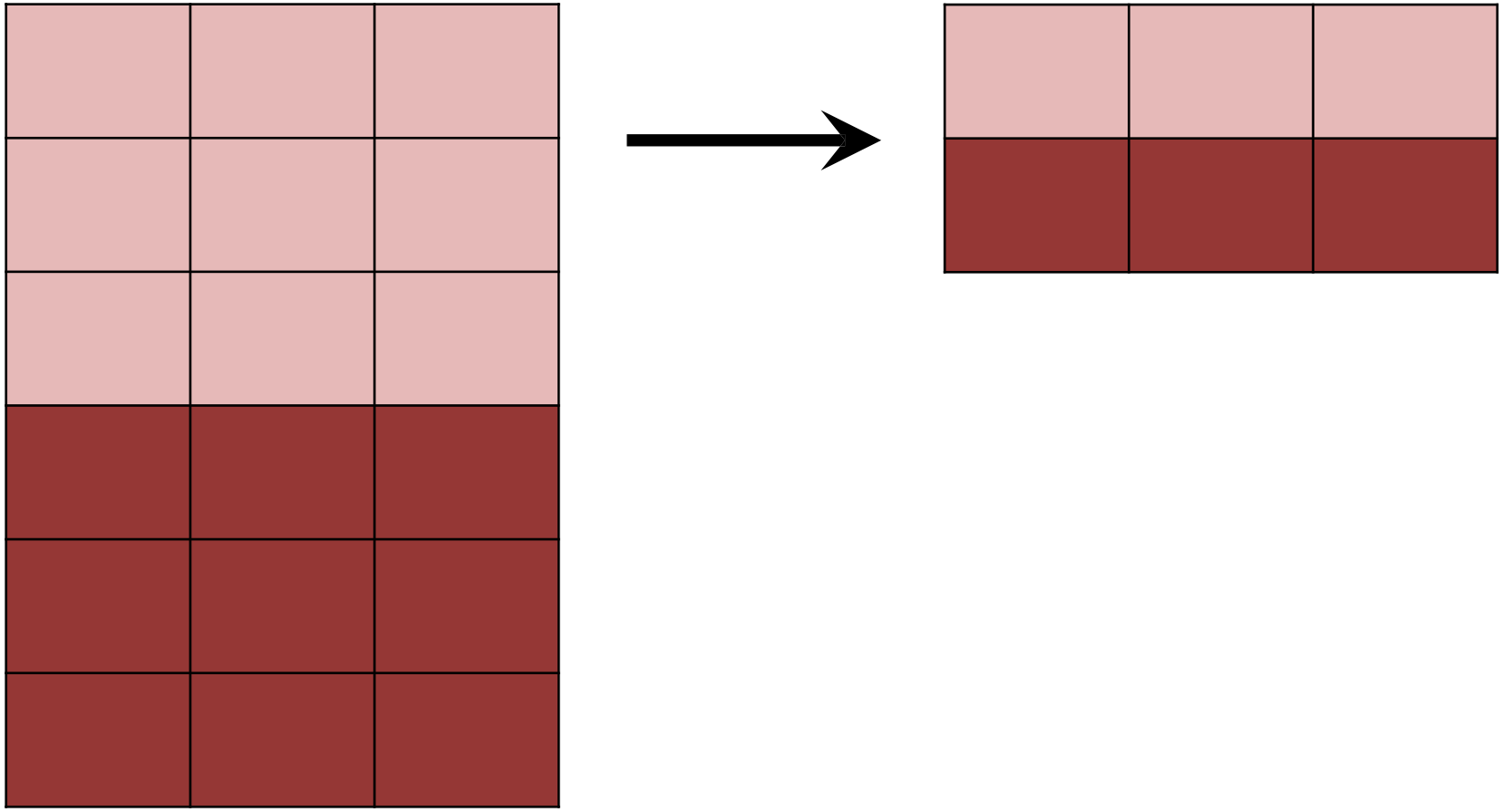
Fundamental actions on data tables:

- make new columns — `mutate()`
- combine tables, adding columns — `left_join()`
- combine tables, adding rows — `bind_rows()`
- choose rows — `filter()`
- choose columns — `select()`
- arrange rows — `arrange()`
- calculate summary statistics — `summarize()`
- work on groups of data — `group_by()`

`summarize()`: collapse multiple rows



`summarize()`: collapse multiple rows



Calculate mean and standard deviation of Sepal.Length

```
> summarize(iris, mean_sepal_length = mean(Sepal.Length),
                  sd_sepal_length   = sd(Sepal.Length))
```

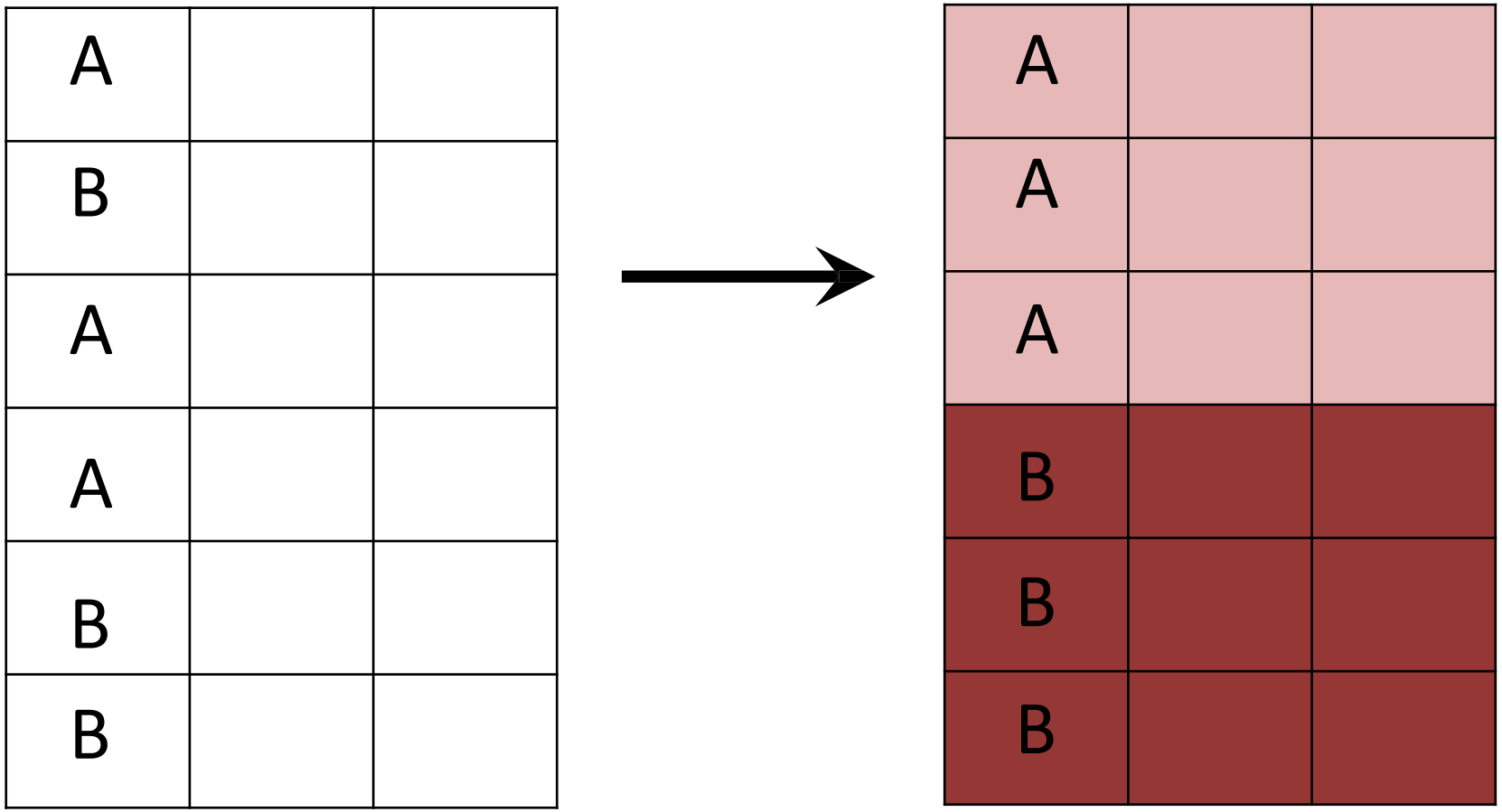
Calculate mean and standard deviation of Sepal.Length

```
> summarize(iris, mean_sepal_length = mean(Sepal.Length),
                  sd_sepal_length   = sd(Sepal.Length))
  mean_sepal_length sd_sepal_length
1          5.843333       0.8280661
```

`group_by()`: set up groupings

A		
B		
A		
A		
B		
B		

`group_by()`: set up groupings



Calculate mean and standard deviation of Sepal.Length, grouped by Species

```
> summarize(group_by(iris, Species),
                  mean_sepal_length = mean(Sepal.Length),
                  sd_sepal_length   = sd(Sepal.Length))
```

Calculate mean and standard deviation of Sepal.Length, grouped by Species

```
> summarize(group_by(iris, Species),
                  mean_sepal_length = mean(Sepal.Length),
                  sd_sepal_length   = sd(Sepal.Length))
Source: local data frame [3 x 3]

     Species mean_sepal_length sd_sepal_length
1     setosa              5.006       0.3524897
2 versicolor              5.936       0.5161711
3  virginica              6.588       0.6358796
```

Pipe example 1: count how many herbivores of different orders there are in `msleep`

	name	genus	vore	order	conse…[1]	sleep…[2]	sleep…[3]	sleep…[4]	awake	brainwt	bodywt
	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	Cheetah	Acinonyx	carni	Carniv…	lc	12.1	NA	NA	11.9	NA	50
2	Owl monkey	Aotus	omni	Primat…	NA	17	1.8	NA	7	0.0155	0.48
3	Mountain beaver	Aplodontia	herbi	Rodent…	nt	14.4	2.4	NA	9.6	NA	1.35
4	Greater short-tailed shrew	Blarina	omni	Soricо…	lc	14.9	2.3	0.133	9.1	0.00029	0.019
5	Cow	Bos	herbi	Artiod…	domest…	4	0.7	0.667	20	0.423	600
6	Three-toed sloth	Bradypus	herbi	Pilosa	NA	14.4	2.2	0.767	9.6	NA	3.85

… with abbreviated variable names [1]conservation, [2]sleep_total, [3]sleep_rem, [4]sleep_cycle

Pipe example 1: count how many herbivores of different orders there are in `msleep`

```
msleep %>%
  filter(vore == "herbi")
```

Pipe example 1: count how many herbivores of different orders there are in `msleep`

```
msleep %>%
  filter(vore == "herbi") %>%
  group_by(order)
```

Pipe example 1: count how many herbivores of different orders there are in `msleep`

```
msleep %>%
  filter(vore == "herbi") %>%
  group_by(order) %>%
  summarize(count = n())
```

Pipe example 1: count how many herbivores of different orders there are in `msleep`

```
msleep %>%
  filter(vore == "herbi") %>%
  group_by(order) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

Pipe example 1: count how many herbivores of different orders there are in `msleep`

```
msleep %>%
  filter(vore == "herbi") %>%
  group_by(order) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
           order count
1       Rodentia    16
2   Artiodactyla     5
3 Perissodactyla     3
4     Hyracoidea     2
5    Proboscidea     2
6  Diprotodontia     1
7     Lagomorpha     1
8         Pilosa     1
9       Primates     1
```

Pipe example 2: What is the median awake time of different orders in `msleep`?

Pipe example 2: What is the median awake time of different orders in `msleep`?

```
msleep %>%
  group_by(order)
```

Pipe example 2: What is the median awake time of different orders in `msleep`?

```
msleep %>%
  group_by(order) %>%
  summarize(med_awake = median(awake))
```

Pipe example 2: What is the median awake time of different orders in `msleep`?

```
msleep %>%
  group_by(order) %>%
  summarize(med_awake = median(awake)) %>%
  arrange(med_awake)
```

Pipe example 2: What is the median awake time of different orders in `msleep`?

```
msleep %>%
  group_by(order) %>%
  summarize(med_awake = median(awake)) %>%
  arrange(med_awake)
```

```
             order med_awake
1       Chiroptera      4.20
2  Didelphimorphia      5.30
3        Cingulata      6.25
4     Afrosoricida      8.40
5           Pilosa      9.60
6         Rodentia     11.10
7    Diprotodontia     11.60
8     Soricomorpha     13.70
9        Carnivora     13.75
10  Erinaceomorpha     13.80
```

Demonstration Time!

Refer to Section #7