INTRODUCTION TO R AND POSITRON

Carsten Lange

clange@cpp.edu

Cal Poly, Pomona

Textbook

LEARNING OUTCOMES

Part 1: Setup

- How to install R and *Positron*
- What is the windows layout of *Positron*
- How to use a (project) folder in *Positron*
- How to extend R's functionality with R-packages and which packages to install

LEARNING OUTCOMES

Part 2: How R Stores the Data:

- Data types
- Data objects in R

LEARNING OUTCOMES

Part 3: The `tidyverse` Package:

- The Structure of R commands
- About the `tidyverse` package for data frames
  - `select()` and rename columns (variables)
  - `filter()` rows (observations)
  - `mutate()` (define columns (variables); overwrite old or create new)
  - `arrange()` sort observations in a data frame.
  - piping (connecting commands) with `|>`.

Textbook

PART 1: INSTALL AND SETUP R AND POSITRON

A typical setup to work with R consists of two components:

- the **R Console** which executes R code and
- an integrated development environment (**IDE**) such as **RStudio** or **Positron**.

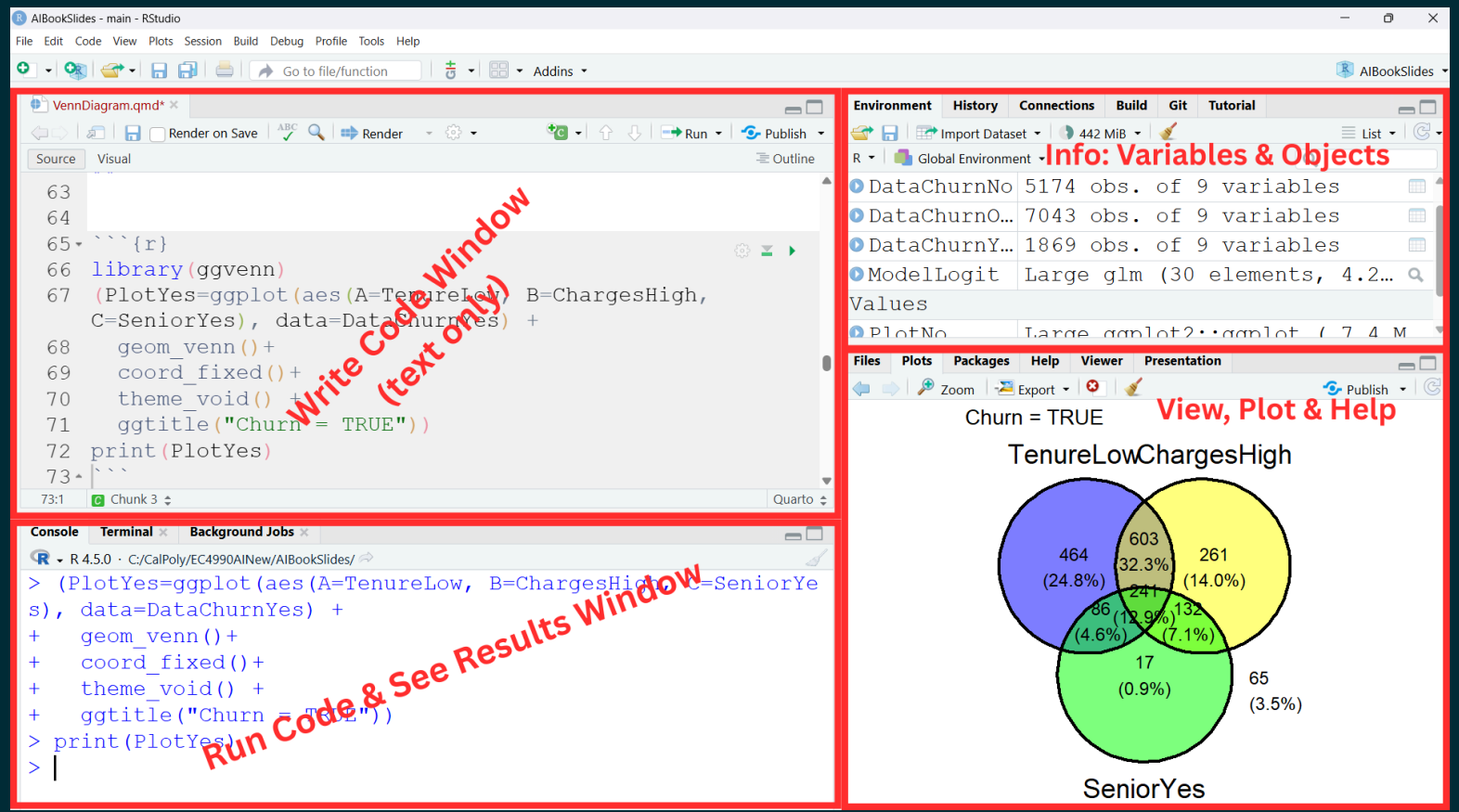
You can download R here: Download R

You can download *Positron* here: Download *Positron*

> **Note**
> - Install R before *Positron*
> - If an older R version exists uninstall it before installing the newer version

RSTUDIO – INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

(DOES NOT HAVE NEWEST FEATURES OF A MODERN IDE)



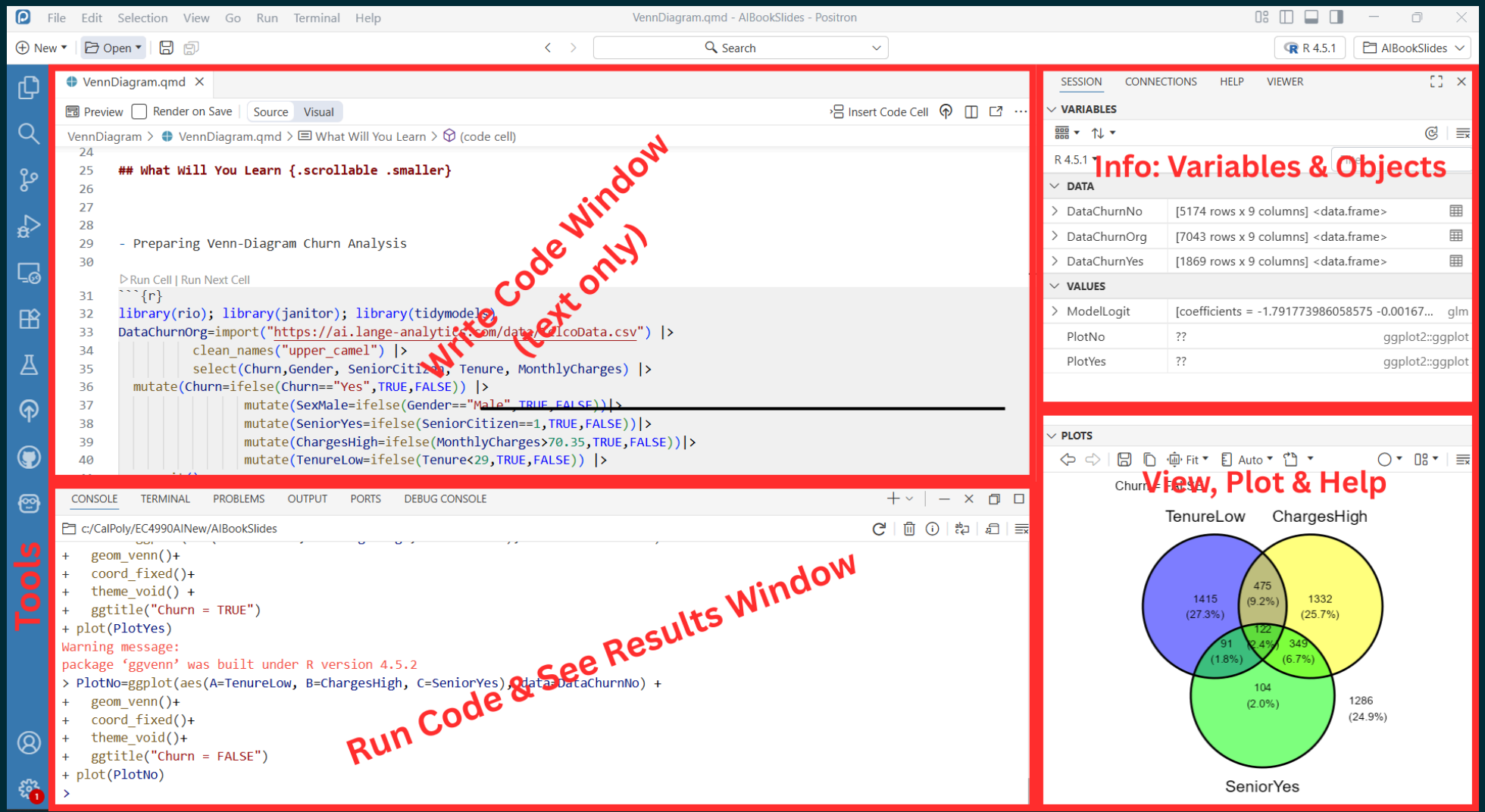
RStudio Window

Video for *First Steps* to setup *R* and *RStudio*: Click here

(However, **it is recommended** to work wit *Positron* rather than *RStudio*)

Textbook

POSITRON – INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

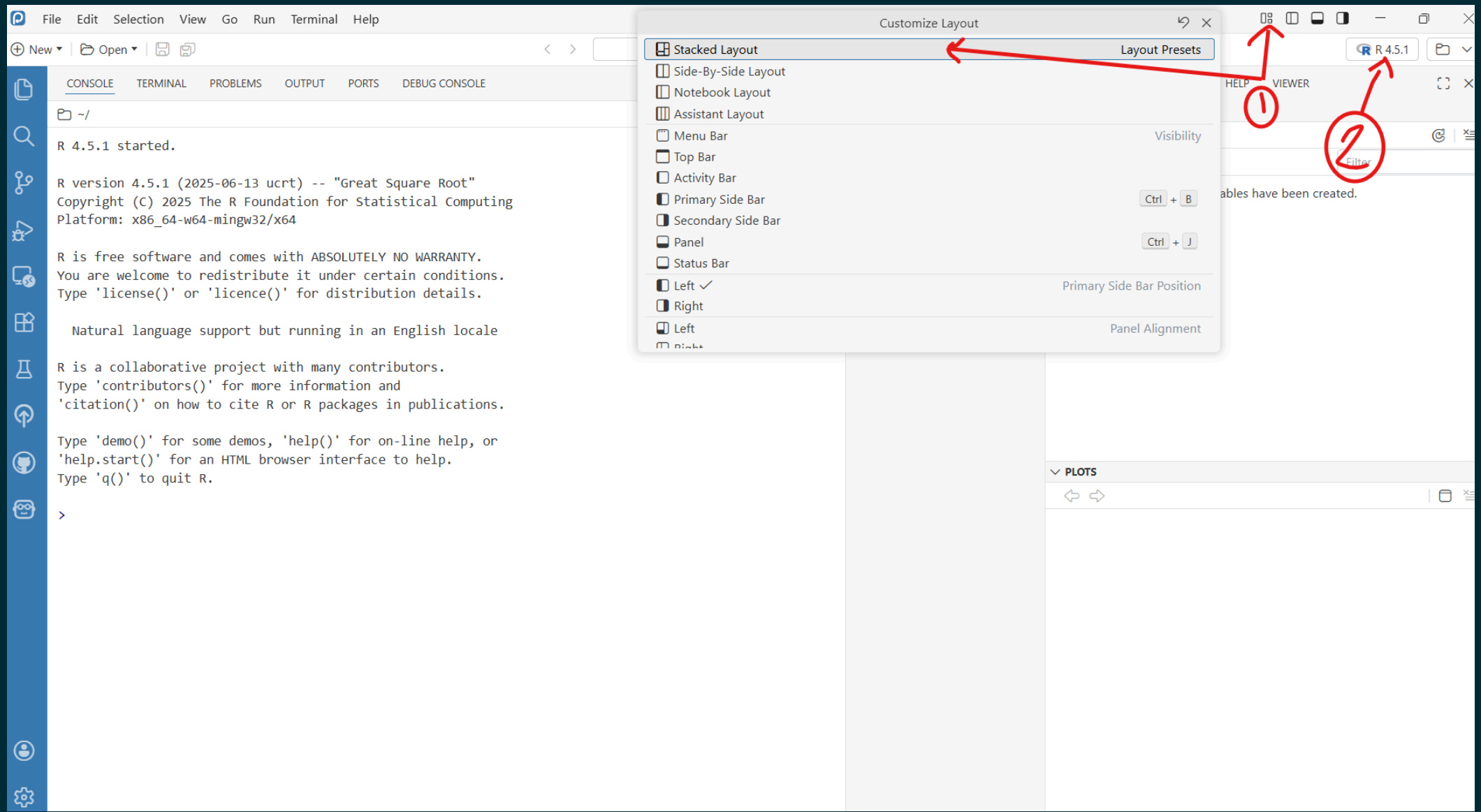


Positron Window

First steps to setup R and *Posit* can be found in this video: coming soon

Textbook

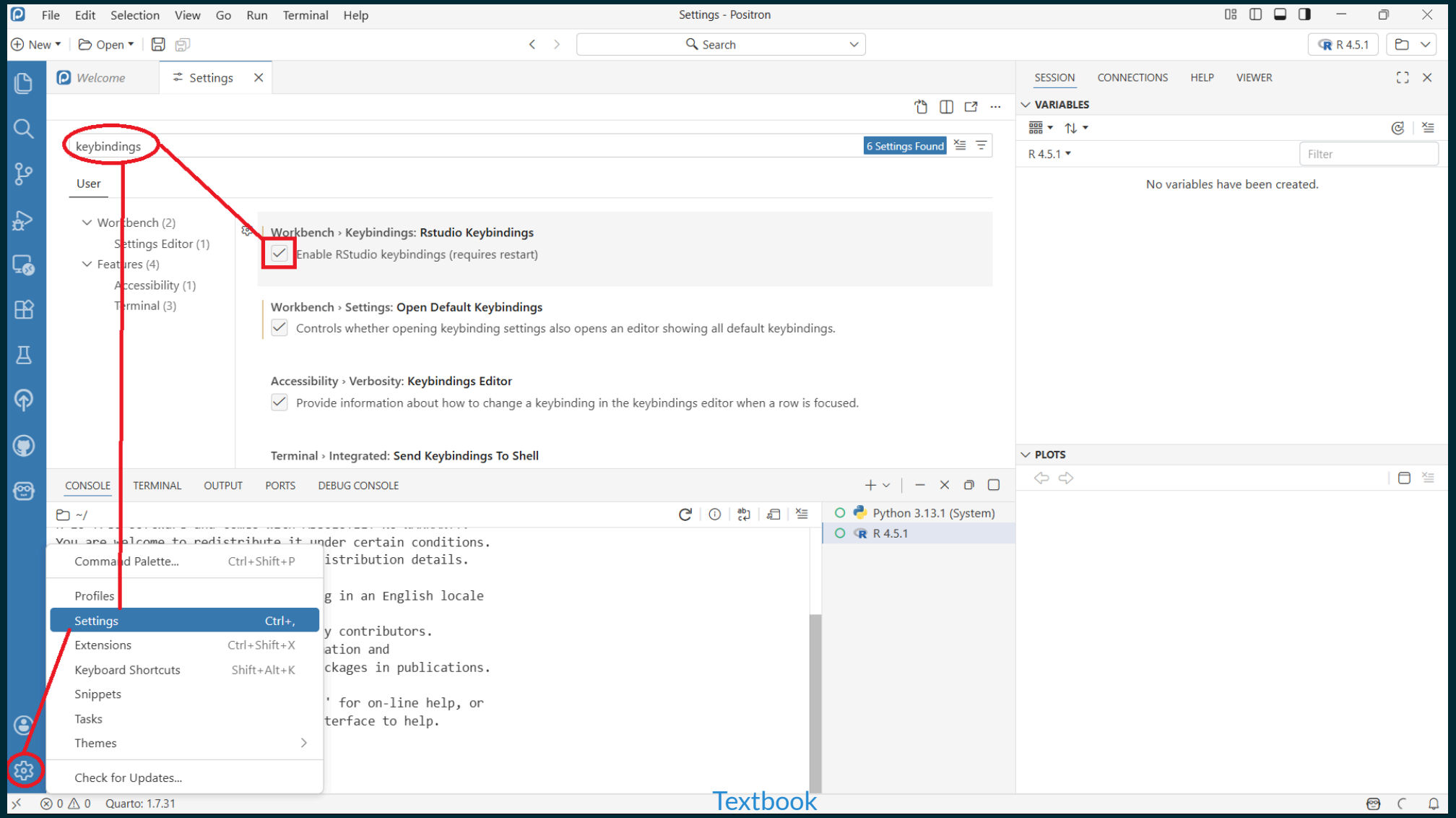
SET WINDOW LAYOUT AND CHOOSE R INTERPRETER



Textbook

RSTUDIO KEYBINDINGS (NEEDS TO DONE ONLY ONCE)

Click Gear Icon -> Choose: Settings -> Search for: Keybindings -> Toggle-On RStudio Keybindings



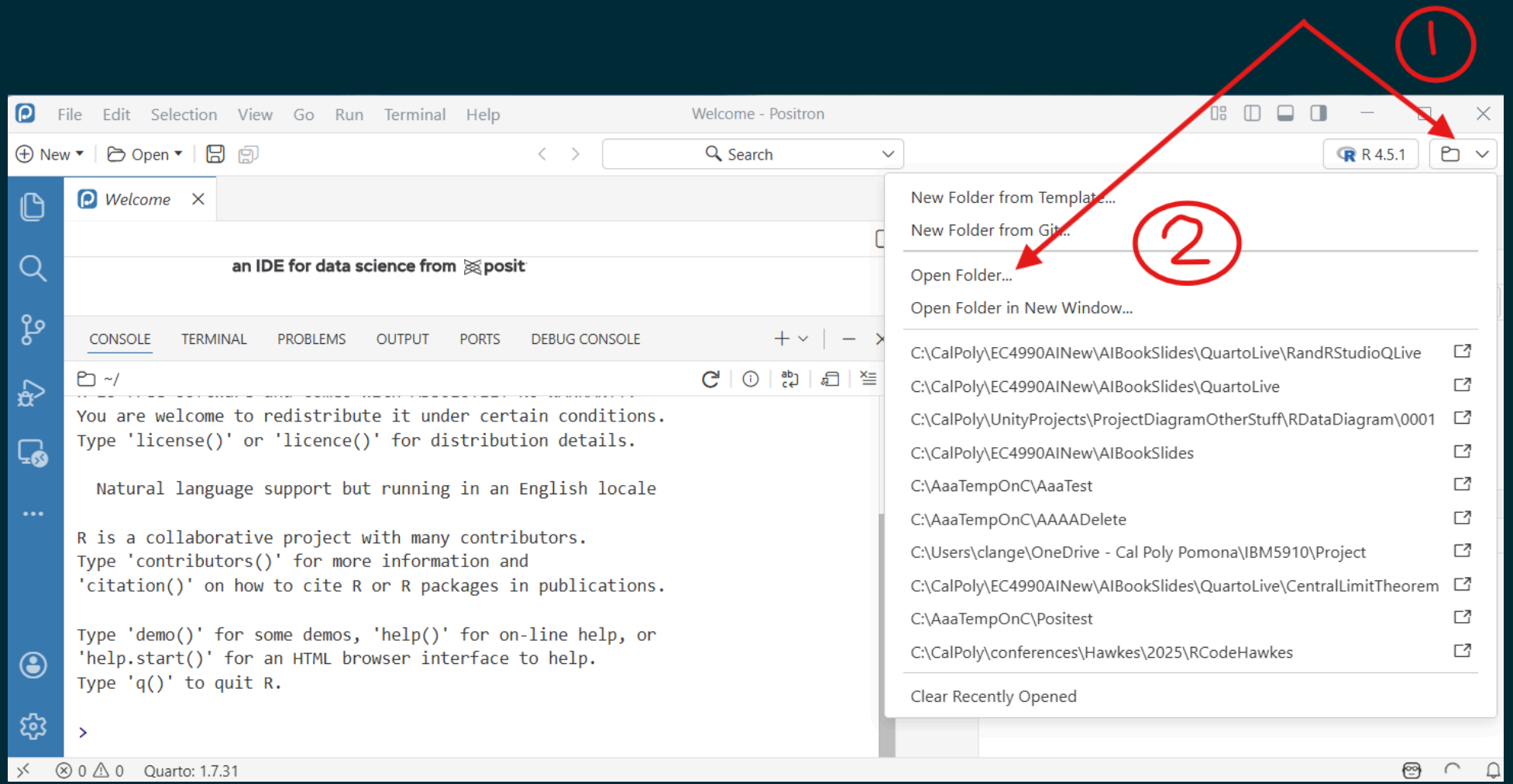
Textbook

Textbook

ALWAYS WORK WITH FOLDERS

Always open or create a folder first

The folder is the one where all your *R* (`.r`), Quarto (`.qmd`), and data (e.g., `.csv`) files are stored



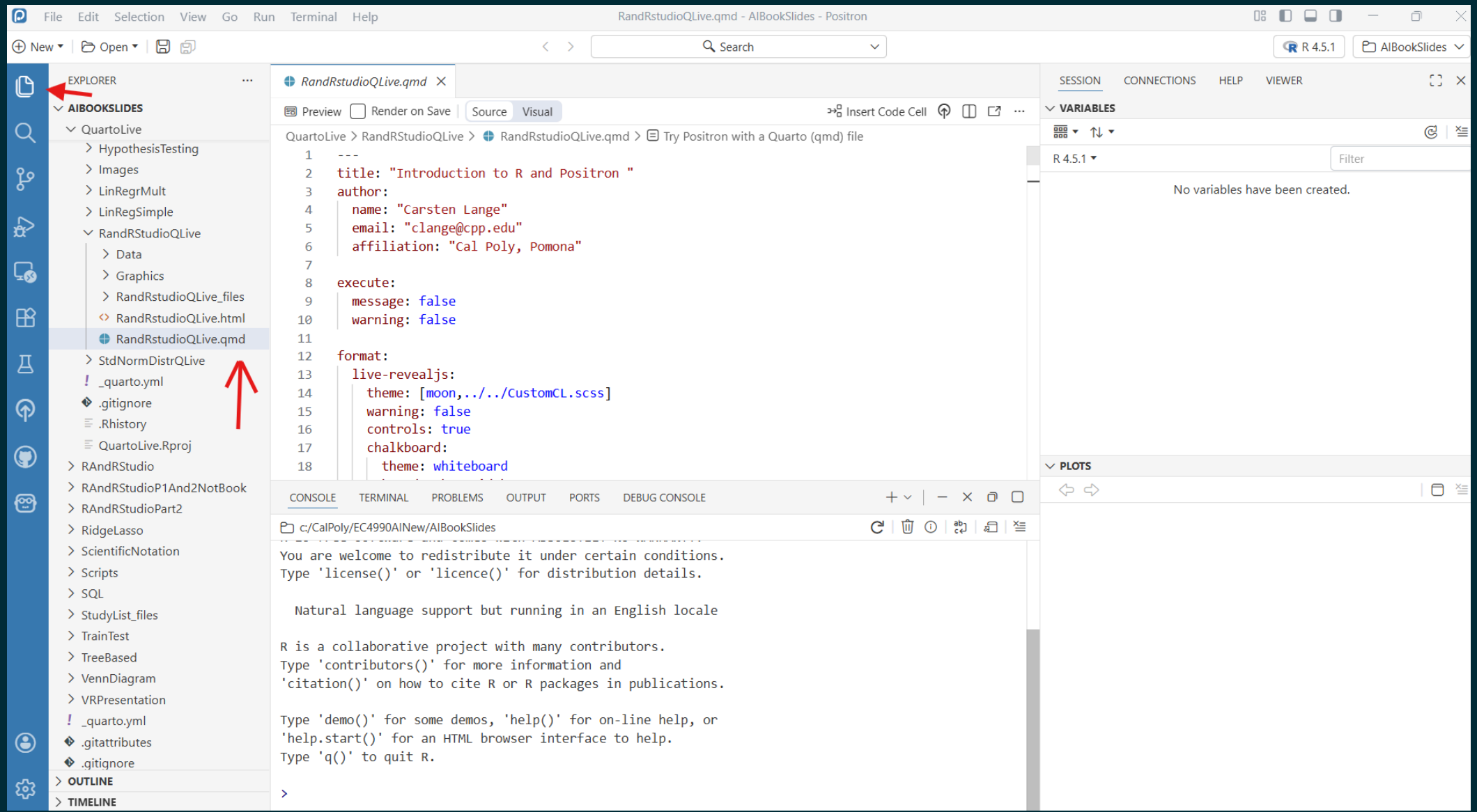
Textbook

USE COMMAND PALETTE TO CREATE A NEW `R` OR `Quarto` FILE

`CTRL SHIFT P` (Windows) or `⌘ SHIFT P` (Mac) to open **Command Pallette**:

1. Type either “New Quarto Document” or “New R Document” into search bar
2. Click and Create file
3. Save file right away

OPEN A FILE FROM AN EXISTING POSITRON FOLDER



POSITRON: FIRST STEPS WITH AN R FILE `(.r)`

- Remember, always open a folder first!

AN R FILE SUCH AS `MyFile.r` CONTAINS ONLY R-CODE

1. Print "Hello world" using the `print()` command.
2. Assign values 3 and 4 to the legs `a` and `b` of an right-angled triangle.

- calculate the hypotenuse c:

$$c^2 = a^2 + b^2 \iff c = \sqrt{a^2 + b^2}$$

- print the result using the `cat(command)`
- Assign the number 2 to the variable ("R" objects) `a` and run the `cat()` command again.

TRY POSITRON WITH A QUARTO FILE `(.qmd)`

- Remember, always open a folder first!

A QUARTO FILE SUCH AS `MyFile.qmd` CONTAINS TEXT AS WELL AS R-CODE

- Text is written in *MarkDown*
- Code is surrounded by:

> ```{r}
> MyCode goes here
> ```

TRY POSITRON WITH A QUARTO FILE (`.qmd`)

Instructions:

1. Print "Hello world" using the `print()` command.
2. Assign values 3 and 4 to the legs `a` and `b` of an right-angled triangle.

- calculate the hypotenuse c:

$$c^2 = a^2 + b^2 \iff c = \sqrt{a^2 + b^2}$$

- print the result using the `cat(command)`

Note, now we want everything nicely commented!

WASM: R RUNS IN A BROWSER INCLUDING LIBRARIES AND DATA

- Print "Hello world" using the `print()` command.
- Assign values 3 and 4 to the legs `a` and `b` of an right-angled triangle.
- Calculate the hypotenuse c

$$c^2 = a^2 + b^2 \iff c = \sqrt{a^2 + b^2}$$

- Print the result using the `cat(command)`
- Assign the number 2 to the variable ("R" object) `a` and run the `cat()` command again.

R PACKAGES

R Packages extend R's functionality. They have to be **installed** only once:

For example, to install the `tidyverse` package, type in the consol window:
`install.packages("tidyverse)`
Needs to be only done once!

After installation packages they need to be **loaded** in every new R script or Quarto file with: `library()`.

Packages frequently used in this course (**please install soon**):

- `tidyverse`: supports easy data processing .
- `rio`: allows loading various data resources with one `import()` command from the user's hard drive or the Internet.
- `janitor`: provides functionality to clean data and rename variable names to avoid spaces and special characters.

VIDEOS FOR THE `rio` AND THE `tidyverse` PACKAGE

Example: How to install the tidyverse package: Click here

Video about the rio package: Click here

USING THE `rio` AND THE `tidyverse` PACKAGE

Example: `rio` and `tidyverse` package (assuming they are installed already)

`import()` would not work if the `rio` package were not loaded.
`select()` would not work if the `tidyverse` package were not loaded.

PART 2: DATA TYPES & DATA OBJECTS

- **Data Types:** Which type of values can R store?
  - numerical `num` (such as: 0.1, 2.3, 3.14157)
  - numerical `int` (such as: 1, 2, 7)
  - character `chr` (such as: “Hello”, “Hi”, “World”)
  - categorical `factor` (such as: “Female”, “Male” Or: “small”, “medium”, “large”)
  - boolean `logic` (True, False)
- **Data Objects:** What are the **containers** R uses to store data?
  - single value as: `single entry`
  - list of entries as: `vector`
  - table as: `dataframe` or `tibble`
  - *advanced objects* can hold: plots, models, prediction results

Textbook

ANALOGY: DATA TYPES & DATA OBJECTS EXAMPLE FOR THREE ALCOHOLIC BEVERAGES

- **Data Types:** Which type of fluids can we store?
  - beer
  - wine
  - whiskey
- **Data Objects:** What are the **containers** to store our liquids?
  - bottles
  - cartons (incl. six packs)
  - cargo containers

DATA TYPES

Numerical Data Type (`num` and `int`): Numerical values (e.g., 1, 523, 7 or 3.45, 0.1, 8.0) are used for calculations. In contrast, ZIP-Codes are not numerical data type.

Character Data Type (`chr`): Storing sequence of characters, numbers, and/or symbols to form a word or even a sentence is called a `character` data type (e.g. first or last names, street addresses, or Zip-codes)

Categorical Data Type (`factor`): A `factor` is an R data type that stores *categorical* data in an effective way. `factor` data types are also required by many classification models in R.

Logic Data Type(`logic`): A data type that stores the logic states `TRUE` and `FALSE` is called a `logic` object (sometimes called Boolean)

PRINT

Print `Hello world!` by using variable A:

CALCULATE WITH VARIABLES AND OUTPUT WITH `cat()`

A rectangular lot has a width of 200 feet (`Width`) and a length (`Length`) of 300 feet. Calculate the area (`Area`) and create a full sentence output.

EXERCISE: `cat()` COMMAND AND SINGLE VALUE OBJECTS WITH DIFFERENT DATA TYPES

Assign your own first and last name, your ZIP code, and your your age, to three character variables (first name, last name, Zip code) and one numerical variable (age). Use `var1`, `var2`, `var3`, `var4`. Afterward, use `Cat()` to output a sentence like `Carsten Lange is 55 years old and lives in ZIP code 92656` using the variables you had created.

AGAIN: DATA TYPES & DATA OBJECTS

- **Data Types:** Which type of values can R store? ✓
  - Numerical
  - Character
  - Categorical / Factor

- **Data Objects:** What are the containers R uses to store data? ?

DATA OBJECTS

- **Single Value Object**
- **Vector Object**
- **Data Frame (Tibble) Object**
- **List Object** (not covered in this course)
- **Advanced Object** such as plots, models, recipes

SINGLE VALUE OBJECT

Objects just store a single value:

VECTOR-OBJECTS

A vector object stores a list of values (numerical, character, factor, or logic; mixing of data types is not allowed)

Example: Weather during the last three days in Stattown:

Vector objects can be used as arguments for an R command to calculate statistics such as the `mean()` or the number of entries in the *vector* (`length()`):

DATA FRAMES (TIBBLES)

A data frame is similar to an Excel table. **A data frame stores the values of R Vectors as variables entries in its columns** .

Note, that the `c()` command **c**ombines values to a vector.

Below we show how the **values from the four vectors** `VecDay`, `VecTemp`, `VecWindSpeed`, and `VevIsSunny` are stored in the *data frame* `DataWeather`.

The columns hold the values from the four vectors and the rows (with the exception of the first row), hold the observations for the various days. The first row contains the variable names:

DATA FRAME FROM TITANIC DATA

Most of the times, we do not build a *data frame* from its vectors (columns). Instead we load the *data frame* from a file (for example, a `csv` file).

Below we load the *Titanic* dataset. Note, only the first six observations are shown.

We can see the *structure* of the data frame by using the `str()` command. This includes the type of all variables/vectors:

EXTRACTING THE VECTORS AND PERFORMING CALCULATIONS (NUMERICAL VECTORS)

Since the columns of a data frame are made up of *vectors*, we can extract these vectors, and use the values for data analysis (remember: observations are in the rows, variables are in the columns).

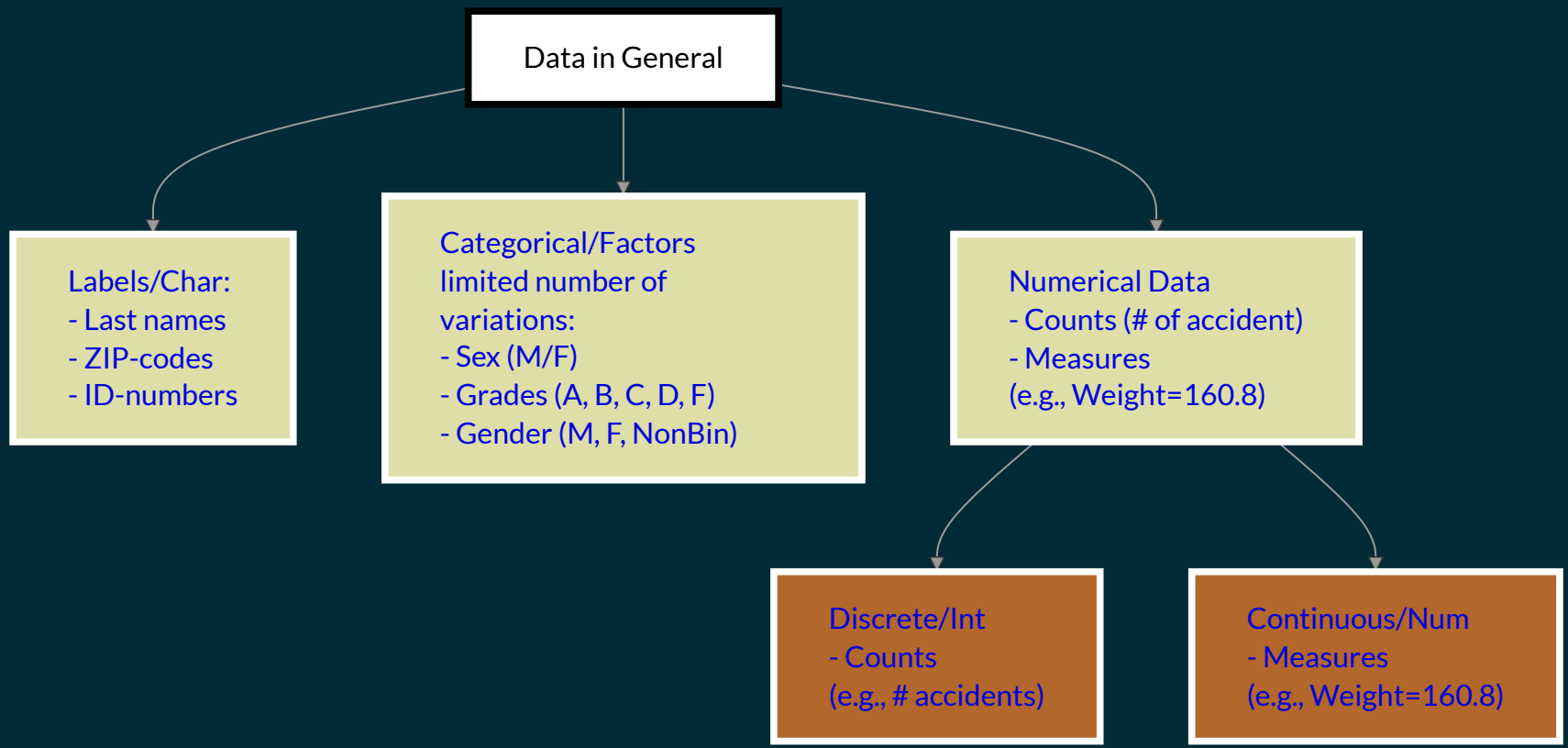
We can use the notation `DataFrameName$VectorName` to extract the vectors:

EXTRACTING THE VECTORS AND PERFORMING CALCULATIONS (LOGICAL VECTORS)

If we like, we can change a vector inside a data frame:

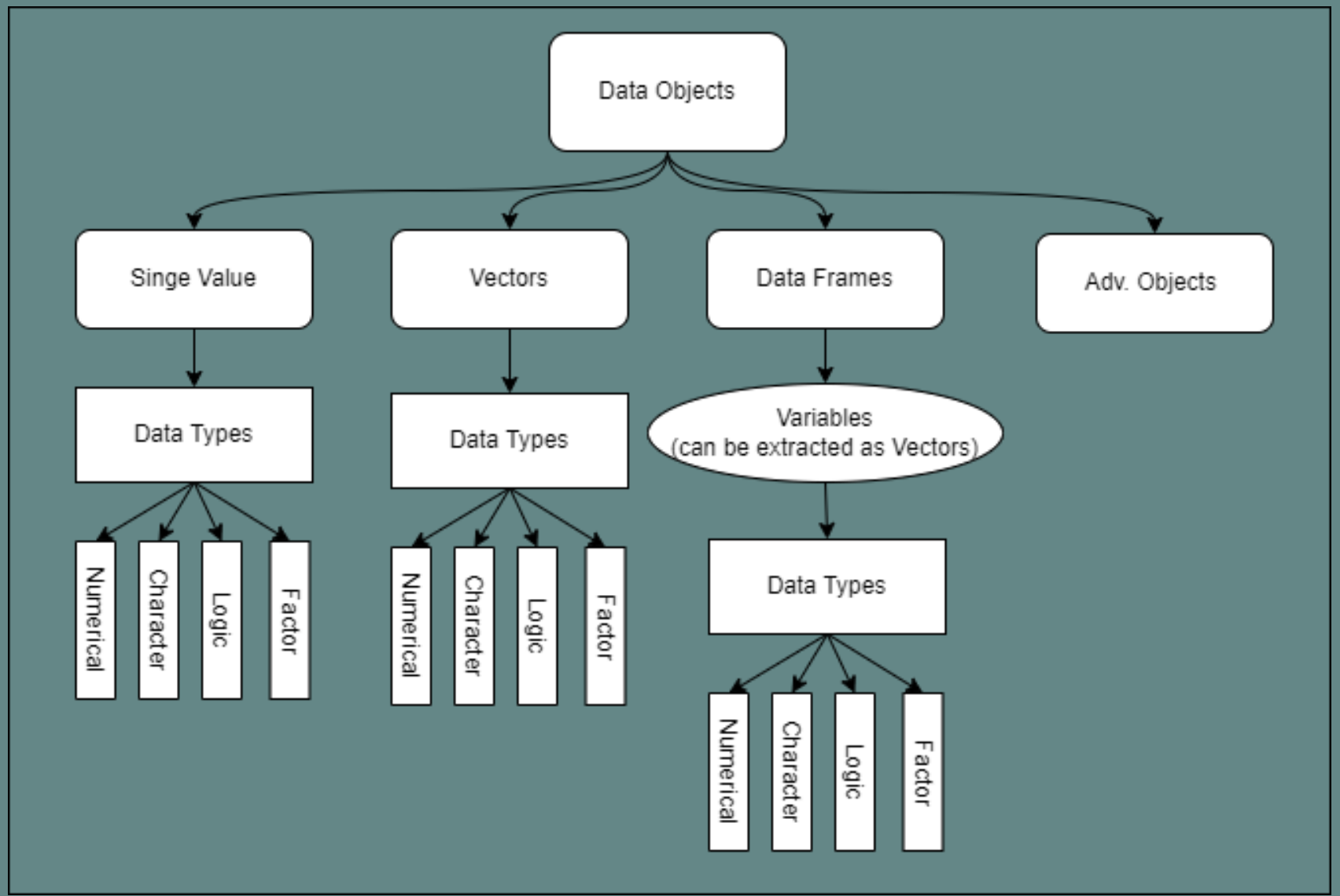
We can use the logical vector *Survived* (remember, `TRUE=1`, `FALSE=0`) to calculate the survival rate:

SUMMARY DATA TYPES



Textbook

SUMMARY DATA TYPES AND OBJECTS



Data Type and Object Structure

Textbook

PART 3: THE `tidyverse` AND PIPING

Textbook

BASICS OF R COMMANDS

R commands consists of the **command's name followed by a pair of parentheses**: `command()`

Inside the `()` we can define one or more **arguments** for the command.

- Arguments in a command usually have names such as `x=` or `data=`
- R does not require to use the argument's name, but **order matters**
- R commands have many arguments. Most have default values
- We can nest commands. However, nesting too deeply makes code difficult to read.»

Textbook

STRUCTURE OF R COMMANDS

Most R commands have the following structure:

$$\underbrace{DataNew}_{\text{R object storing the result}} = \underbrace{Command}_{\text{Name of the command}} \underbrace{(\overbrace{Data}^{\text{1. Argument: Data to process}}, \overbrace{Arg2, Arg3, \ldots, ArgN}^{\text{More Arguments}})}_{\text{Arguments inside () and separated by comma}}$$

Often the `data` argument is the first argument in a command. Usually named `data=` or `x=`.»

Textbook

USE A COMMAND WITH AND WITHOUT ARGUMENT NAMES

All three examples are equivalent

GETTING HELP ABOUT A COMMAND (E.G., `mean`)

Use `?mean` or `help(mean)` in the RStudio console to see the default values.

You can also *mark/highlight* and then press *F1*

Try it for the `mean()` command.

IMPORTANT COMMANDS FROM `tidyverse`/`dplyr` PACKAGE

- `dplyr` package is part of the `tidyverse` (meta) package

- `library(tidyverse)` (loads the `tidyverse` and its packages)
- `select()` selects columns (variables) from a data frame
- `filter()` filters rows (observations) for specific criteria
- `mutate()` calculates new or overwrites existing columns (variables) based on other columns (just like Excel)
- `arrange()` sorts a data frame according to one or more columns in ascending order (use argument `desc()` for descending order)

TITANIC DATASET

EXAMPLE: USING THE `tidyverse` FOR DATA ANALYSIS

Goal: Create a data frame with a few selected variables, that contains only female observations, and the fare in current U.S.-$.

THE `select()` COMMAND

- `select(DataMine, Var1, Var2)` selects columns (variables) `Var1` and `Var2` from a data frame `DataMine`. The first argument is the `data=` argument followed by the names of the selected variables.
- `select(DataMine, -Var1, -Var2)` selects all columns (variables) except `Var1` and `Var2` from a data frame `DataMine`.

Here is an example using the `DataTitanic` data frame with a few selected variables:

THE `filter()` COMMAND

The `filter()` command filters rows (observations) of a data frame for specific criteria. The first argument is the `data=` argument followed by the filter criteria.

E.g., *filter* for female passengers:
We use `DataTitanicSelVar` that we created in the previous slide at as a starting dataframe and save the result in `DataTitanicSelVarFem`.
Note, we have to use `==` instead of `=` for the criteria):

THE `mutate()` COMMAND

`mutate()` creates or overwrites columns (variables) based on other columns (just like Excel). The first argument is the `data=` argument followed by the instructions on how to create the new variable.

E.g., *mutate* calculates the `FareIn2023Dollars` by multiplying `FareInPounds` by 108.5. The command uses `DataTitanicSelVarFem` from the previous slide:*

SUMMARY

We now have a data frame with only women and columns $Survived$, $PasClass$, Sex, Age, and $FareIn2023Dollars$.

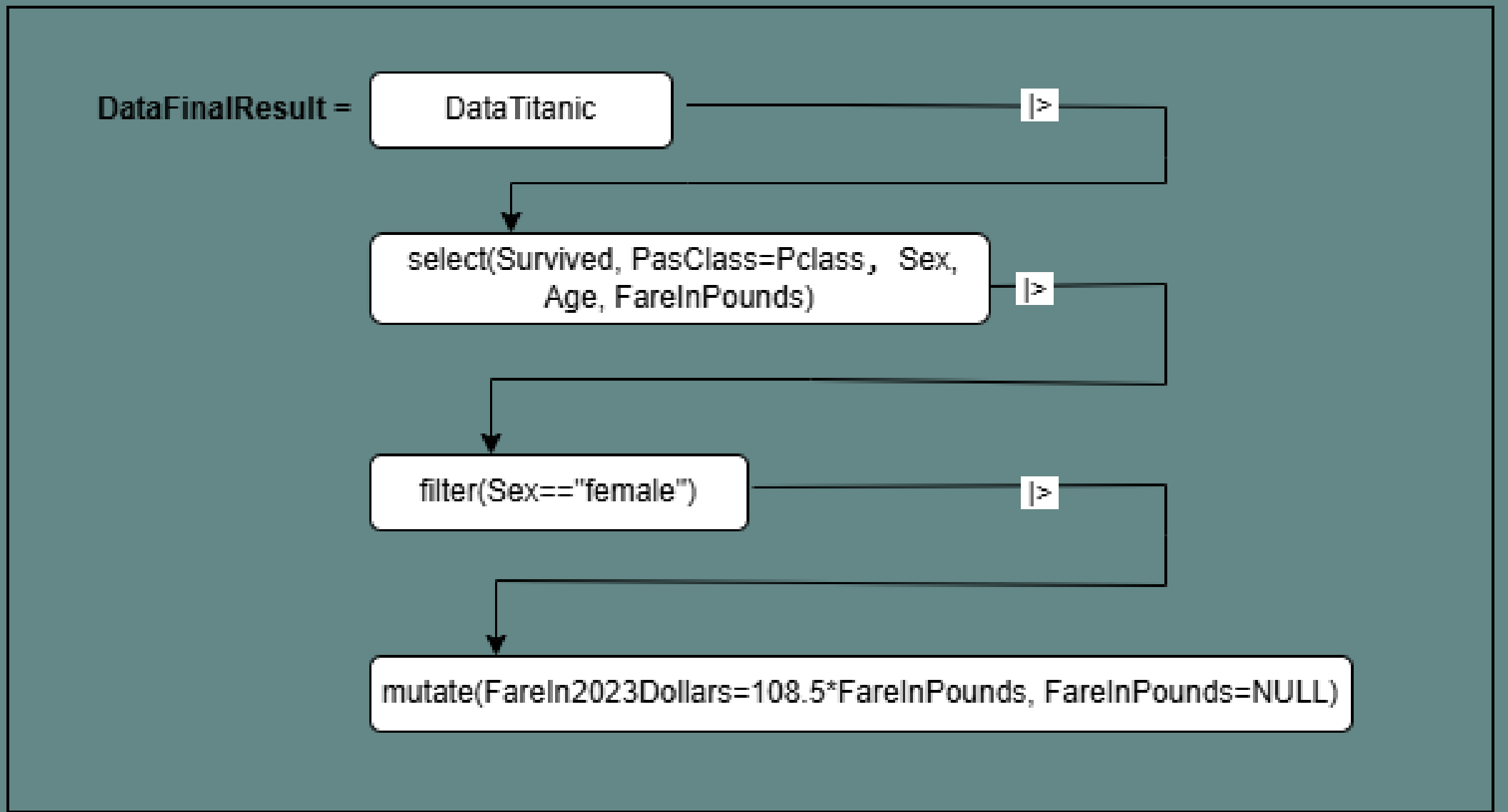
How did we get there:

1. We selected variables $Survived$, $PasClass$, Sex, Age, $FareInPounds$ and saved in `DataTitanicSelVar`
2. We filtered for females and saved in `DataTitanicSelVarFem`
3. We mutated to calculate a new variable $FareIn2023Dollars$ and saved finally in `DataTitanicSelVarFemDolFare`

Could this be done easier?

Note, overwriting data frames such as `DataTitanic` is usually a bad idea! Nesting the command is possible but very difficult to read.

PIPING SCHEMA



Piping Schema

ALTERNATIVE: PIPING

(WILL BE USED THROUGHOUT THE COURSE/BOOK)

Shortcut for `|>`: `CTRL SHIFT M` (Windows) or `⌘ SHIFT M` (Mac).

The pipe operator `|>` is for most practical purposes equivalent to `%>%`.

WHY R?

- Excel analytics is not reproducible
- SPSS focuses on surveys
- STATA and SAS are commercial products
  - not free
  - progress has to go through the corporate hierarchy and therefore is slower
  - limited support community

R AND/OR PYTHON

- Analysis is always reproducible with little effort
- free
- extensive support
- R or Python
  - R is easier to understand for users with limited coding experience
  - Python is faster in incorporating cutting-edge algorithms
  - transfer from R to Python or vice versa is easy
  - Quarto supports both R and Python even simultaneously in the same project

PYTHON VS. R – THE TASK

Let us compare code for the same tasks between *R* and *Python*:

- Download the Titanic dataset
- select the variables `Sex`, `FareInPounds`, `Survived` (renamed to: `Surv`)
- Calculate a new column `FareInDollars` by multiplying `FareInPounds` by 108.5
- Filter for `Sex` being *female*
- Calculate the mean of `FareInDollars`

PYTHON VS. R – THE RESULTS (USING PANDAS)

```r
library(tidyverse)
library(rio)
DataTitanicR = import("Data/Titanic.csv") |>
  select(Sex, FareInPounds, Surv = Survived) |>
  mutate(FareInDollars = FareInPounds * 108.5) |>
  filter(Sex == "female")
MeanFareWomen = mean(DataTitanicR$FareInDollars)
print(MeanFareWomen)
```

```
[1] 4826.06
```

PYTHON VS. R – THE RESULTS (USING POLARS)

```r
library(tidyverse)
library(rio)
DataTitanicR = import("Data/Titanic.csv") |>
  select(Sex, FareInPounds, Surv = Survived) |>
  mutate(FareInDollars = FareInPounds * 108.5) |>
  filter(Sex == "female")
MeanFareWomen = mean(DataTitanicR$FareInDollars)
print(MeanFareWomen)
```

```
[1] 4826.06
```

Note, `polars` is currently not supported in WASM

WAS CHIVALRY DEAD IN 1912?

To answer the question, we develop a male and a female data frame and compare the survival rates.

In each data frame we would need only the variables `Sex` and `Survived` but we add also `PasClass` for additional analysis.

EXERCISE: THE MALE DATA FRAME

We select `Sex`, `Survived`, and `PasClass=Pclass` and filter for `male`:

THE FEMALE DATA FRAME

We select `Sex`, `Survived`, and `PasClass=Pclass` and filter for `female`:

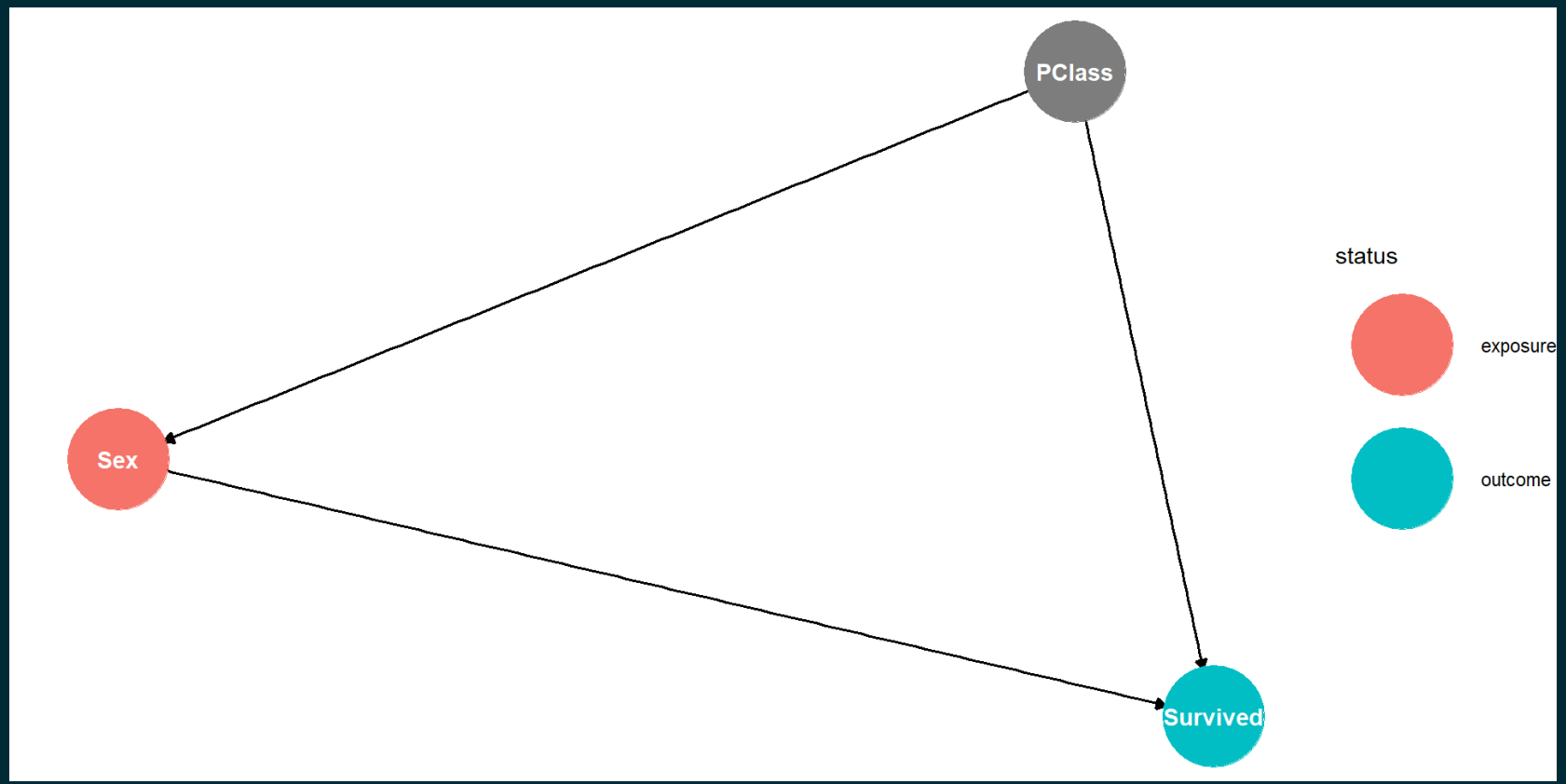
COMPARING THE SURVIVAL PROPORTION OF MALES TO FEMALES

Hint: You could either calculate the female proportion as `sum(DataFemale$Survived/nrow(DataFemale))` or `mean(DataFemale$Survived)`.

BE CRITICAL WITH YOUR OWN RESEARCH

`PasClass` IS A CONFOUNDER

The third class was deep in the hull of the Titanic with low survival chances and more men were traveling in that class. This makes `PasClass` a confounder. Therefore we have to analyze male and female survival by class: We have to filter for `Sex` and `PasClass`.



Textbook

SURVIVAL RESEARCH FOR PASSENGER CLASS 1

- Select `Survived`, `Sex`, `PasClass=Pclass`
- Filter for `PasClass` and `Sex` (`female` and `male`)

SURVIVAL RESEARCH FOR PASSENGER CLASS 2

- Select `Survived`, `Sex`, `PasClass=Pclass`
- Filter for `PasClass` and `Sex` (`female` and `male`)

SURVIVAL RESEARCH FOR PASSENGER CLASS 3

- Select `Survived`, `Sex`, `PasClass=Pclass`
- Filter for `PasClass` and `Sex` (`female` and `male`)