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 (plus how it presents very big/small numbers)

- Data types
- Data objects in R
- How does R presents very big/small numbers

Textbook

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 `|>`.

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**.

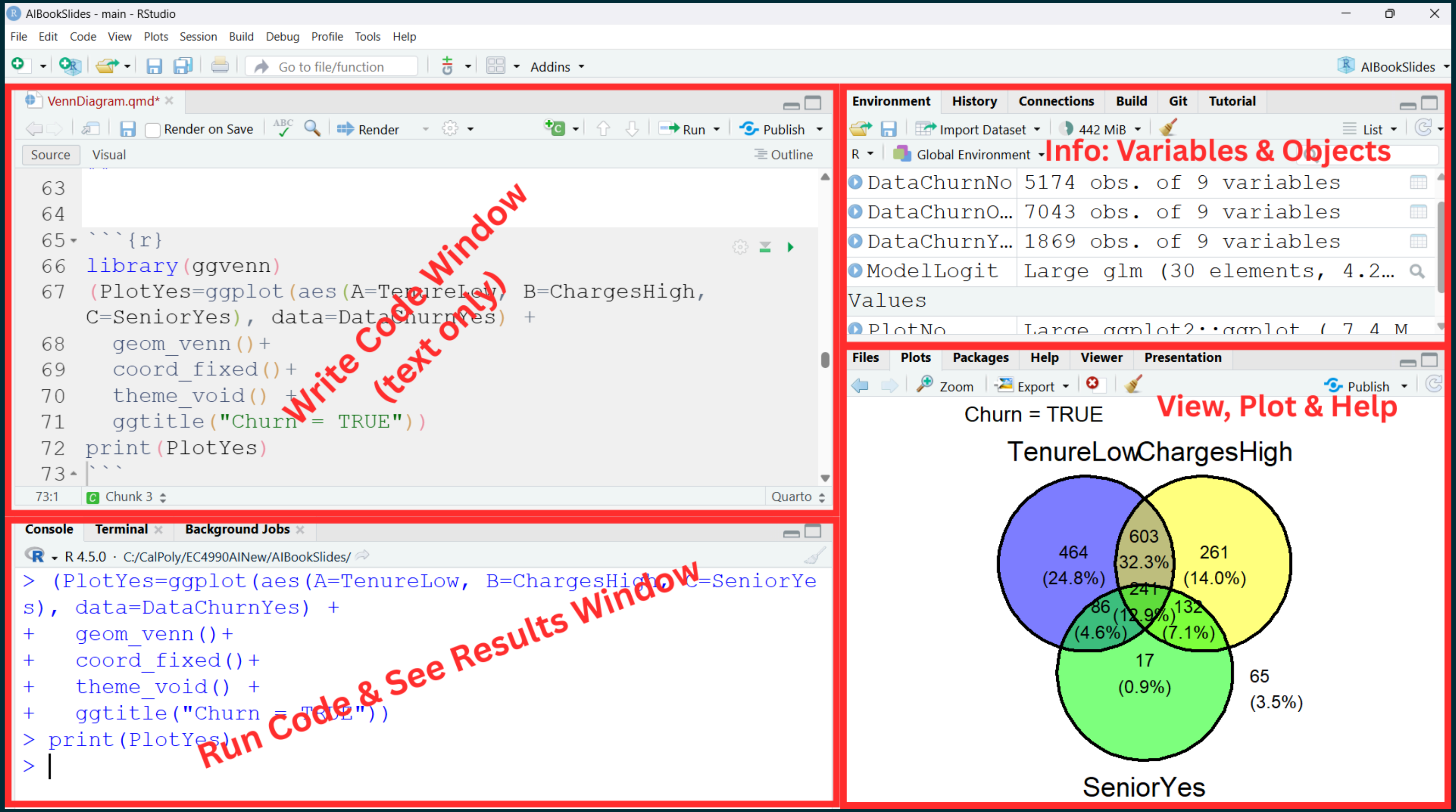
You can download R here: Download R

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)



~~~
```{r}
library(ggvenn)
(PlotYes=ggplot(aes(A=TenureLow, B=ChargesHigh,
C=SeniorYes), data=DataChurnYes) +
  geom_venn()+
  coord_fixed()+
  theme_void() +
  ggtitle("Churn = TRUE"))
print(PlotYes)
```
~~~

```
> (PlotYes=ggplot(aes(A=TenureLow, B=ChargesHigh, C=SeniorYe
s), data=DataChurnYes) +
+   geom_venn()+
+   coord_fixed()+
+   theme_void() +
+   ggtitle("Churn = TRUE"))
> print(PlotYes)
> 
```

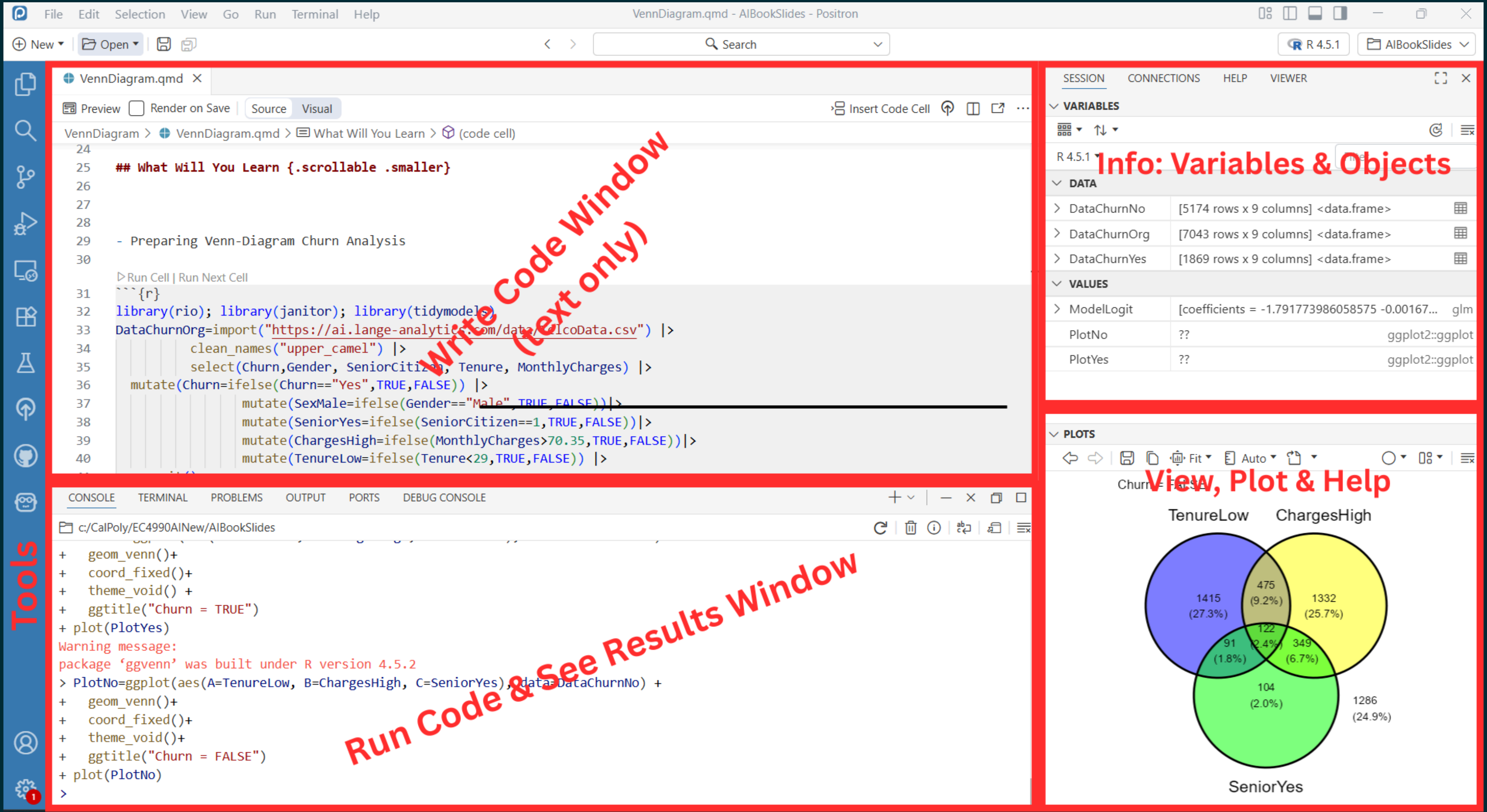
```
DataChurnNo     5174 obs. of 9 variables
DataChurnO...   7043 obs. of 9 variables
DataChurnY...   1869 obs. of 9 variables
ModelLogit      Large glm (30 elements, 4.2...
Values
PlotNo          Large ggplot2::ggplot ( 7 4 M
```

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)



VennDiagram.qmd - AIBookSlides - Positron
File Edit Selection View Go Run Terminal Help
New Open Search R 4.5.1 AIBookSlides
VennDiagram.qmd
Preview Render on Save Source Visual Insert Code Cell
VennDiagram > VennDiagram.qmd > What Will You Learn > (code cell)
Write Code Window (text only)
Tools
CONSOLE TERMINAL PROBLEMS OUTPUT PORTS DEBUG CONSOLE
c:/CalPoly/EC4990AINew/AIBookSlides
Run Code & See Results Window
SESSION CONNECTIONS HELP VIEWER
VARIABLES
Info: Variables & Objects
PLOTS
View, Plot & Help
Churn = FALSE
TenureLow
ChargesHigh
SeniorYes

~~~
## What Will You Learn {.scrollable .smaller}



- Preparing Venn-Diagram Churn Analysis

▷Run Cell | Run Next Cell
```{r}
library(rio); library(janitor); library(tidymodels)
DataChurnOrg=import("https://ai.lange-analytics.com/data/TelcoData.csv") |>
          clean_names("upper_camel") |>
          select(Churn,Gender, SeniorCitizen, Tenure, MonthlyCharges) |>
  mutate(Churn=ifelse(Churn=="Yes",TRUE,FALSE)) |>
                 mutate(SexMale=ifelse(Gender=="Male",TRUE,FALSE))|>
                 mutate(SeniorYes=ifelse(SeniorCitizen==1,TRUE,FALSE))|>
                 mutate(ChargesHigh=ifelse(MonthlyCharges>70.35,TRUE,FALSE))|>
                 mutate(TenureLow=ifelse(Tenure<29,TRUE,FALSE)) |>
~~~

```
+   geom_venn()+
+   coord_fixed()+
+   theme_void() +
+   ggtitle("Churn = TRUE")
+ plot(PlotYes)
Warning message:
package ‘ggvenn’ was built under R version 4.5.2
> PlotNo=ggplot(aes(A=TenureLow, B=ChargesHigh, C=SeniorYes), data=DataChurnNo) +
+   geom_venn()+
+   coord_fixed()+
+   theme_void()+
+   ggtitle("Churn = FALSE")
+ plot(PlotNo)
>
```

R 4.5.1

DATA	
DataChurnNo	[5174 rows x 9 columns] <data.frame>
DataChurnOrg	[7043 rows x 9 columns] <data.frame>
DataChurnYes	[1869 rows x 9 columns] <data.frame>
VALUES	
ModelLogit	[coefficients = -1.791773986058575 -0.00167... glm
PlotNo	?? ggplot2::ggplot
PlotYes	?? ggplot2::ggplot

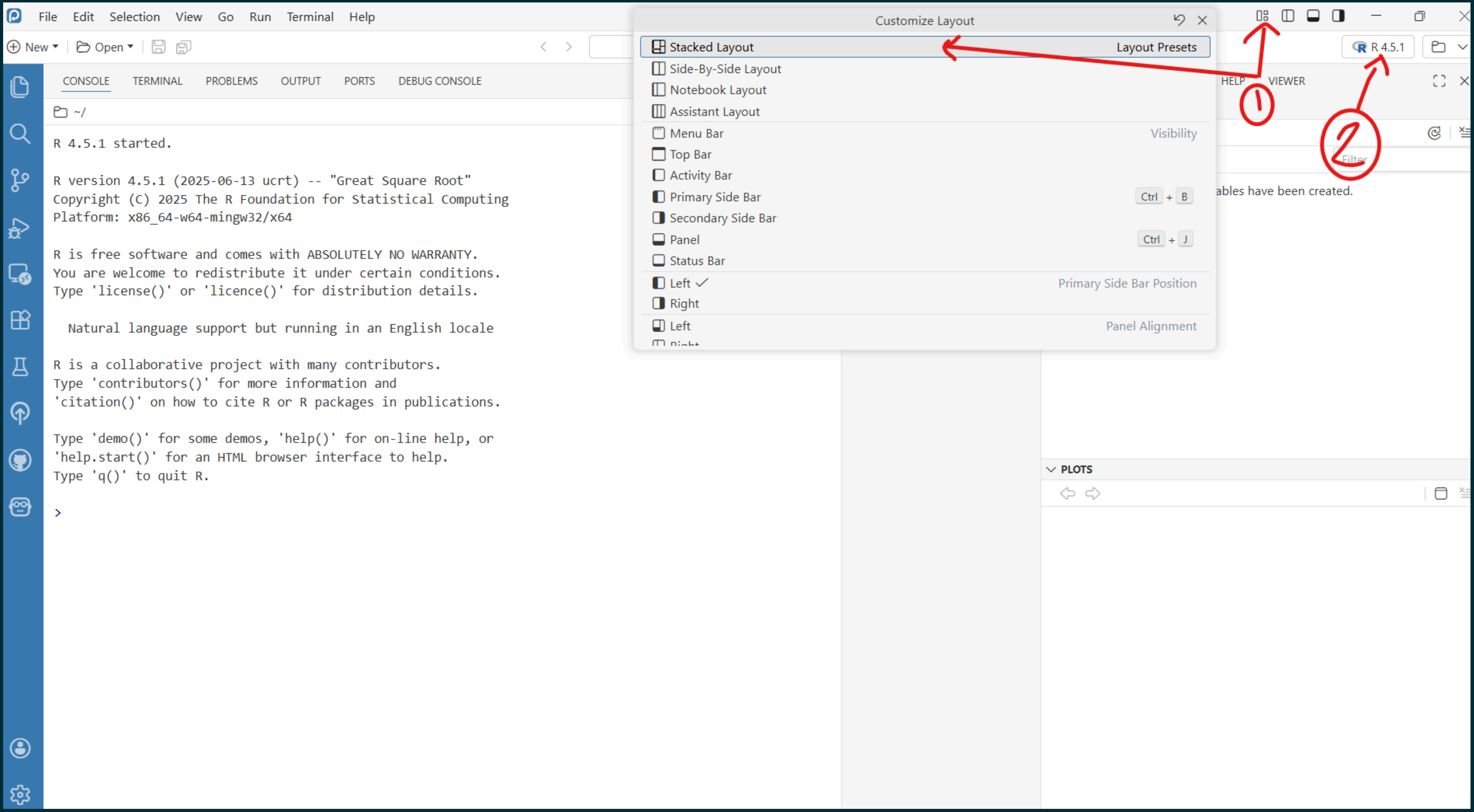
Venn diagram values: TenureLow only 1415 (27.3%); ChargesHigh only 1332 (25.7%); TenureLow & ChargesHigh 475 (9.2%); all three 122 (2.4%); TenureLow & SeniorYes 91 (1.8%); ChargesHigh & SeniorYes 349 (6.7%); SeniorYes only 104 (2.0%); outside 1286 (24.9%).

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



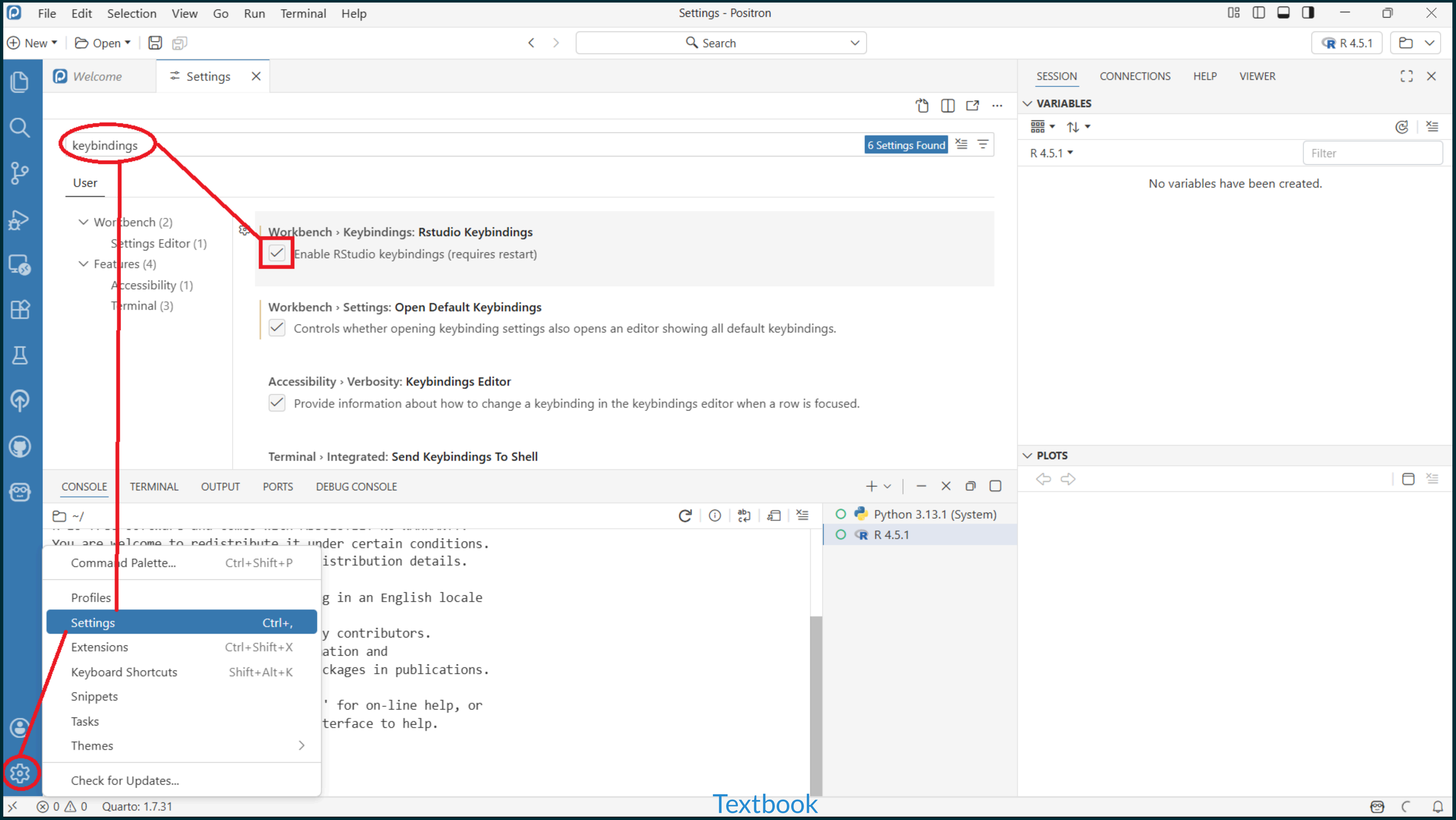
```
R 4.5.1 started.

R version 4.5.1 (2025-06-13 ucrt) -- "Great Square Root"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

ables have been created.

Textbook

RSTUDIO KEYBINDINGS (NEEDS TO DONE ONLY ONCE)

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



Settings - Positron

keybindings — 6 Settings Found

Workbench › Keybindings: **Rstudio Keybindings**
✓ Enable RStudio keybindings (requires restart)

Workbench › Settings: **Open Default Keybindings**
✓ Controls whether opening keybinding settings also opens an editor showing all default keybindings.

Accessibility › Verbosity: **Keybindings Editor**
✓ Provide information about how to change a keybinding in the keybindings editor when a row is focused.

Terminal › Integrated: **Send Keybindings To Shell**

No variables have been created.

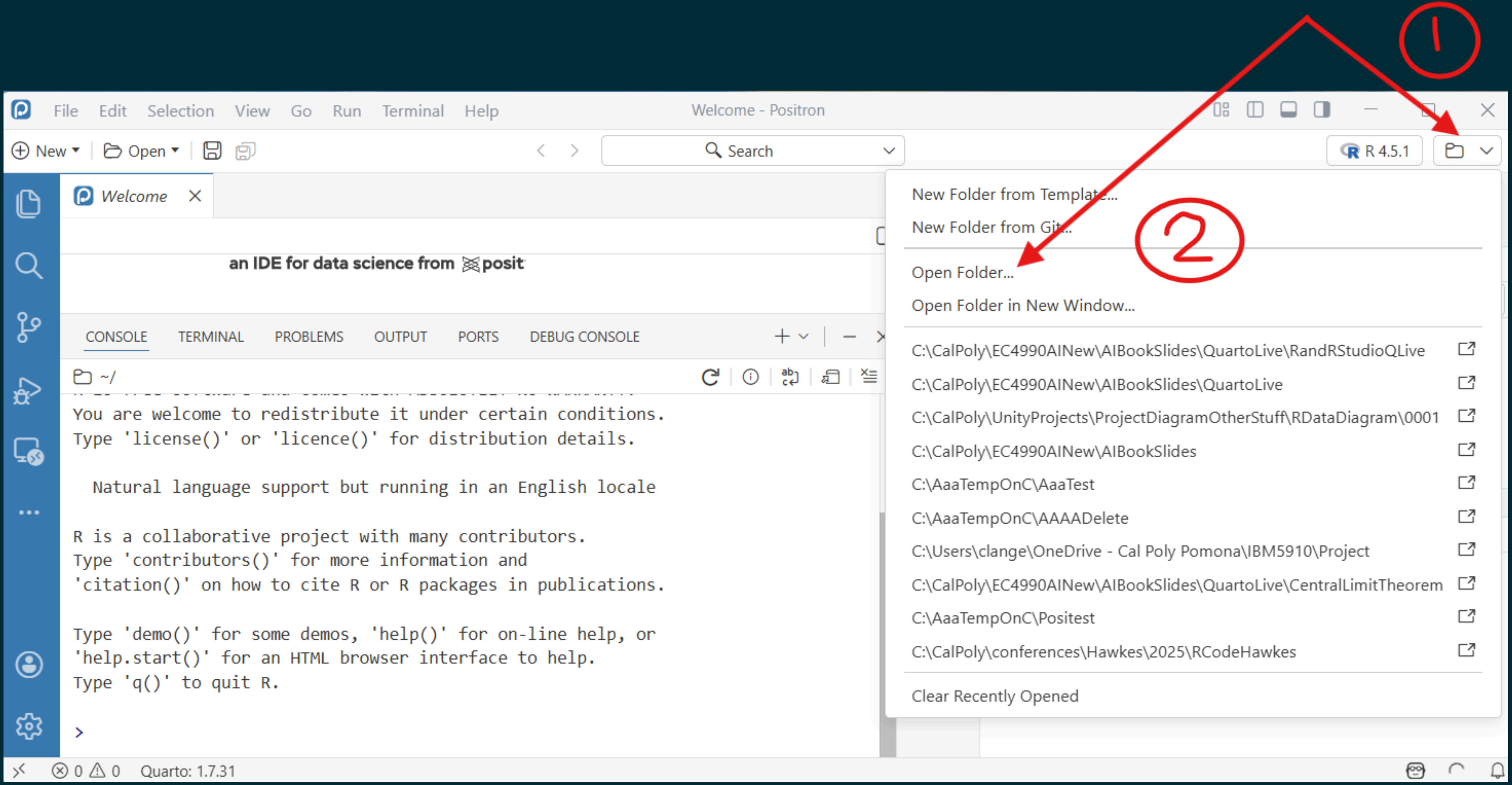
Textbook

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


File Edit Selection View Go Run Terminal Help
Welcome - Positron
New
Open
Search
R 4.5.1
Welcome
an IDE for data science from posit
CONSOLE TERMINAL PROBLEMS OUTPUT PORTS DEBUG CONSOLE
~/
New Folder from Template...
New Folder from Git...
Open Folder...
Open Folder in New Window...
Clear Recently Opened
1
2
Quarto: 1.7.31

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

C:\CalPoly\EC4990AINew\AIBookSlides\QuartoLive\RandRStudioQLive
C:\CalPoly\EC4990AINew\AIBookSlides\QuartoLive
C:\CalPoly\UnityProjects\ProjectDiagramOtherStuff\RDataDiagram\0001
C:\CalPoly\EC4990AINew\AIBookSlides
C:\AaaTempOnC\AaaTest
C:\AaaTempOnC\AAAADelete
C:\Users\clange\OneDrive - Cal Poly Pomona\IBM5910\Project
C:\CalPoly\EC4990AINew\AIBookSlides\QuartoLive\CentralLimitTheorem
C:\AaaTempOnC\Positest
C:\CalPoly\conferences\Hawkes\2025\RCodeHawkes

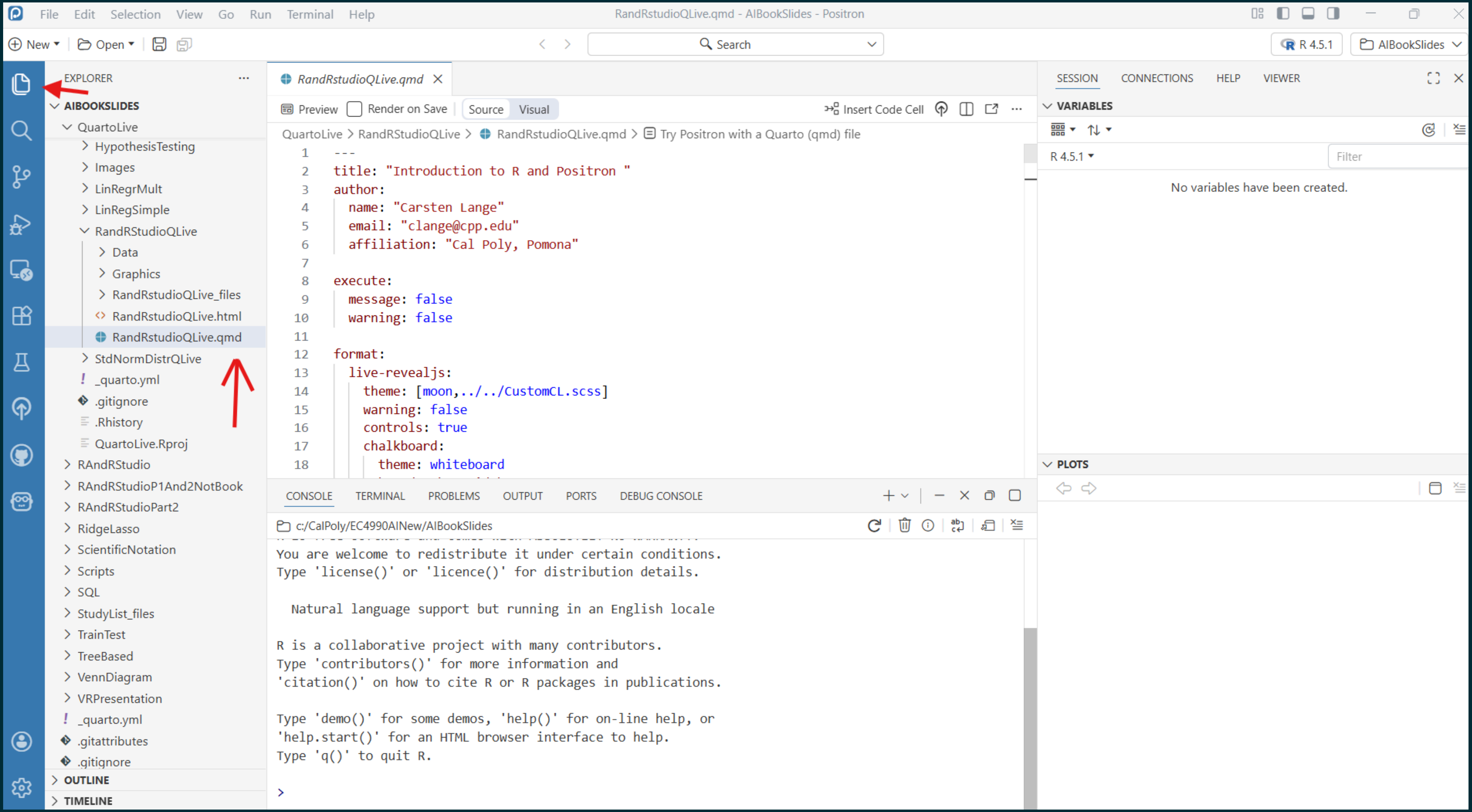
Textbook

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



```
---
title: "Introduction to R and Positron "
author:
  name: "Carsten Lange"
  email: "clange@cpp.edu"
  affiliation: "Cal Poly, Pomona"

execute:
  message: false
  warning: false

format:
  live-revealjs:
    theme: [moon,../../CustomCL.scss]
    warning: false
    controls: true
    chalkboard:
      theme: whiteboard
```

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Textbook

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.

Textbook

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!

Textbook

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 Code   Start Over   Run Code

```r
print(...)
a=3
b=4
c= ...
cat(...)
```

Textbook

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.

Textbook

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.

R Code Start Over Run Code

```r
library(tidyverse)
library(rio)
DataHousing =
  import("Data/HousingData.csv") |>
  select(Price=price, Sqft=sqft_living, Bedrooms=bedrooms,Waterfront=waterfront)
print(DataHousing)
```

Textbook

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

Textbook

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:

R Code Start Over Run Code

```r
A=@@@
print(@@@)
```

Textbook

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.

R Code Start Over Run Code

```r
@@@ = @@@
@@@ = @@@
@@@ = @@@ * @@@
cat("The area of the lot is", @@@, "square feet.")
```

Textbook

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.

R Code Start Over Run Code

```r
Var1=@@@
Var2=@@@
Var3=@@@
Var4=@@@

cat(@@@, @@@, "is", @@@, "years old and lives in ZIP code", @@@)
```

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:

R Code Start Over Run Code

```r
A=1.5
B="beers are too much to drive?"
C=TRUE
cat(A, B, "Answer:", C)
```

Textbook

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:

R Code Start Over Run Code

```r
VecDay=c(1,2,3)
VecTemp=c(70, 68, 55)
VecWindSpeed=c("low","low","high")
VecIsSunny=c(TRUE,TRUE,FALSE)
print("Four vectors successfully defined!")
```

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()`):

R Code Start Over Run Code

```r
MeanTemp=mean(VecTemp)
cat("The average forecasted temperature is", MeanTemp)
```

R Code Start Over Run Code

Textbook

```r
ForecDays=length(VecTemp)
cat("The forecast is for", ForecDays, "days.")
```

Textbook

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:

R Code Start Over Run Code

```r
DataWeather=data.frame(Day=c(1,2,3), Temp=c(70,68,55),
                       WindSpeed=c("low","low","high"),
                       IsSunny=c(TRUE,TRUE,FALSE))
kable(DataWeather)
```

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.

R Code Start Over Run Code

```r
library(rio)
DataTitanic=import("Data/Titanic.csv")
head(DataTitanic)
```

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

R Code Start Over Run Code

```r
str(DataTitanic)
```

Textbook

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:

R Code   Start Over   Run Code

```r
VecFareInPounds=DataTitanic$FareInPounds
AvgFare=mean(VecFareInPounds)
cat("The average fare of Titanic passengers was:", AvgFare, "British Pounds")
```

Textbook

EXTRACTING THE VECTORS AND PERFORMING CALCULATIONS (LOGICAL VECTORS)

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

R Code Start Over Run Code

```r
DataTitanic$Survived=as.logical(DataTitanic$Survived)
str(DataTitanic)
```

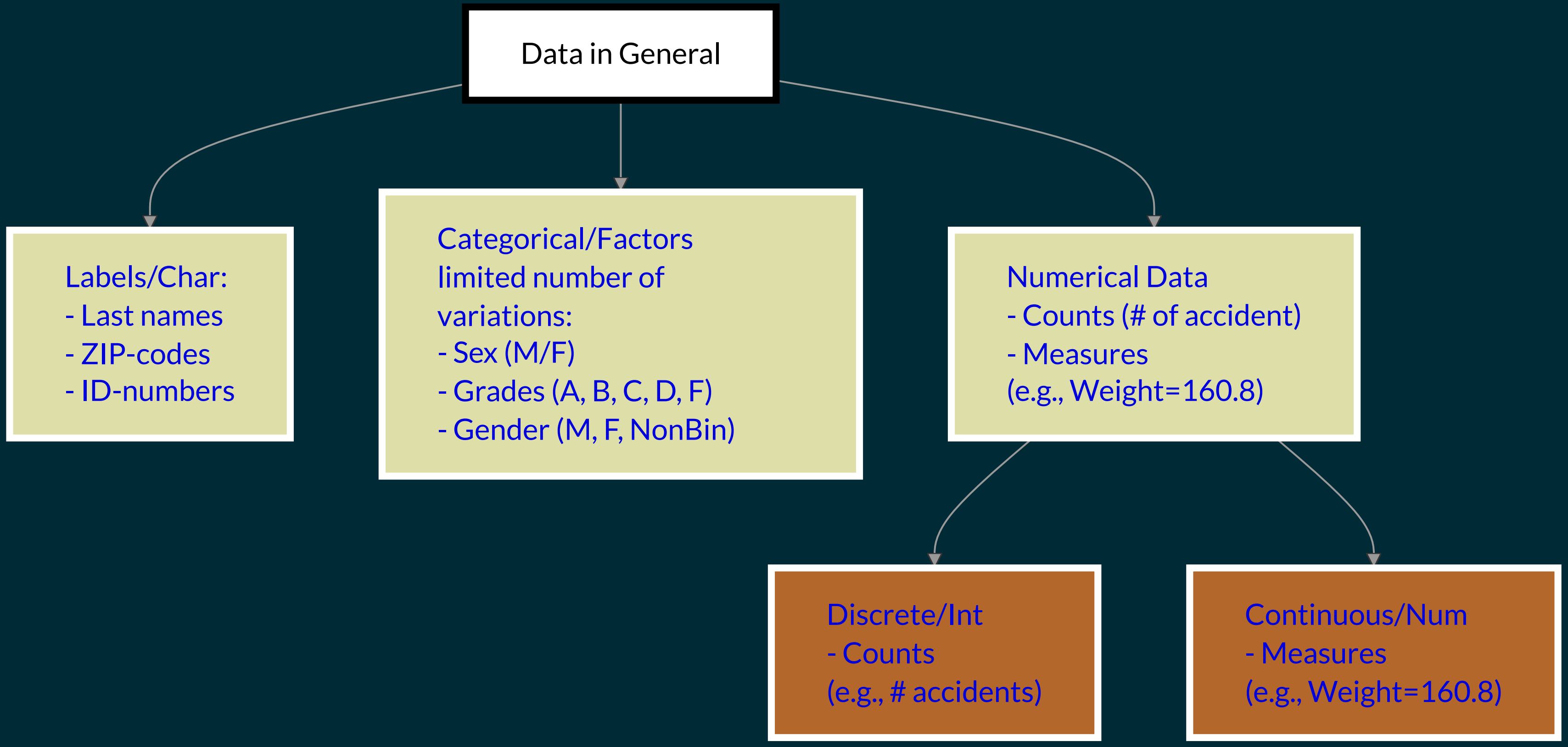
We can use the logical vector *Survived* (remember, `TRUE=1`, `FALSE=0`) to calculate the survival rate:

R Code Start Over Run Code

```r
SurvRate=mean(DataTitanic$Survived)
cat("The average survival rate of Titanic passengers was:", SurvRate)
```

Textbook

SUMMARY DATA TYPES



Textbook

SUMMARY DATA TYPES AND OBJECTS

Data Type and Object Structure

Textbook

HOW ARE VERY BIG NUMBERS PRESENTED

The GDP for 2021 in the US was $ 22,996,086,000,000 (rounded to millions)

$$\begin{aligned} GDP &= 2.2996086 \cdot 10000000000000 \\ &\iff \\ GDP &= 2.2996086 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \\ &\iff \\ GDP &= 2.2996086 \cdot 10^{13} \end{aligned}$$

Let us see what R does:

R Code Start Over Run Code

```r
GDPUS=22996086000000
print(GDPUS)
```

Textbook

HOW ARE VERY SMALL NUMBERS PRESENTED

Main Why?

The probability of getting struck by lightning in the US is about 0.000000000365 on any randomly chosen day.

$$
\begin{aligned}
ProbLight &= \frac{3.65}{10000000000} \\
&\iff \\
ProbLight &= \frac{3.65}{10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10} \\
&\iff \\
ProbLight &= \frac{3.65}{10^{10}} \\
&\iff \\
ProbLight &= 3.65 \cdot 10^{-10}
\end{aligned}
$$

Let us see what R does:

R Code Start Over Run Code

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.

R Code Start Over Run Code

```r
VecTest=c(1,2,3)

sum(x=VecTest)
mean(VecTest)
```

- 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:

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

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

```
VecTest=        c        (1        ,2        ,3        )
```

R Code Start Over Run Code

```r
Result=mean(x=VecTest, trim=0, na.rm=FALSE)
cat("The mean of the values in vector VecTest is:", Result)
```

R Code Start Over Run Code

```r
Result=mean(VecTest, 0, FALSE)
cat("The mean of the values in vector VecTest is:", Result)
```

R Code Start Over Run Code

```r
Result=mean(VecTest)
cat("The mean of the values in vector VecTest is:", Result)
```

All three examples are equivalent

Textbook

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

Use `?mean` or `help(mean)` in the *Positron* 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)

Textbook

TITANIC DATASET

R Code Start Over Run Code

```r
library(rio)
DataTitanic=import("Data/Titanic.csv")
head(DataTitanic)
```

Textbook

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:

R Code Start Over Run Code

```r
library(tidyverse)
DataTitanicSelVar=select(DataTitanic,Survived,PasClass=Pclass,Sex,Age,FareInPounds)
head(DataTitanicSelVar)
```

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):

R Code | Start Over | Run Code

```r
DataTitanicSelVarFem=filter(DataTitanicSelVar, Sex=="female")
head(DataTitanicSelVarFem)
```

Textbook

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:*

R Code   Start Over   Run Code

```r
DataTitanicSelVarFemDolFare=mutate(DataTitanicSelVarFem,
                                   FareIn2023Dollars=108.5*FareInPounds,
                                   FareInPounds=NULL)
head(DataTitanicSelVarFemDolFare)
```

Textbook

SUMMARY

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

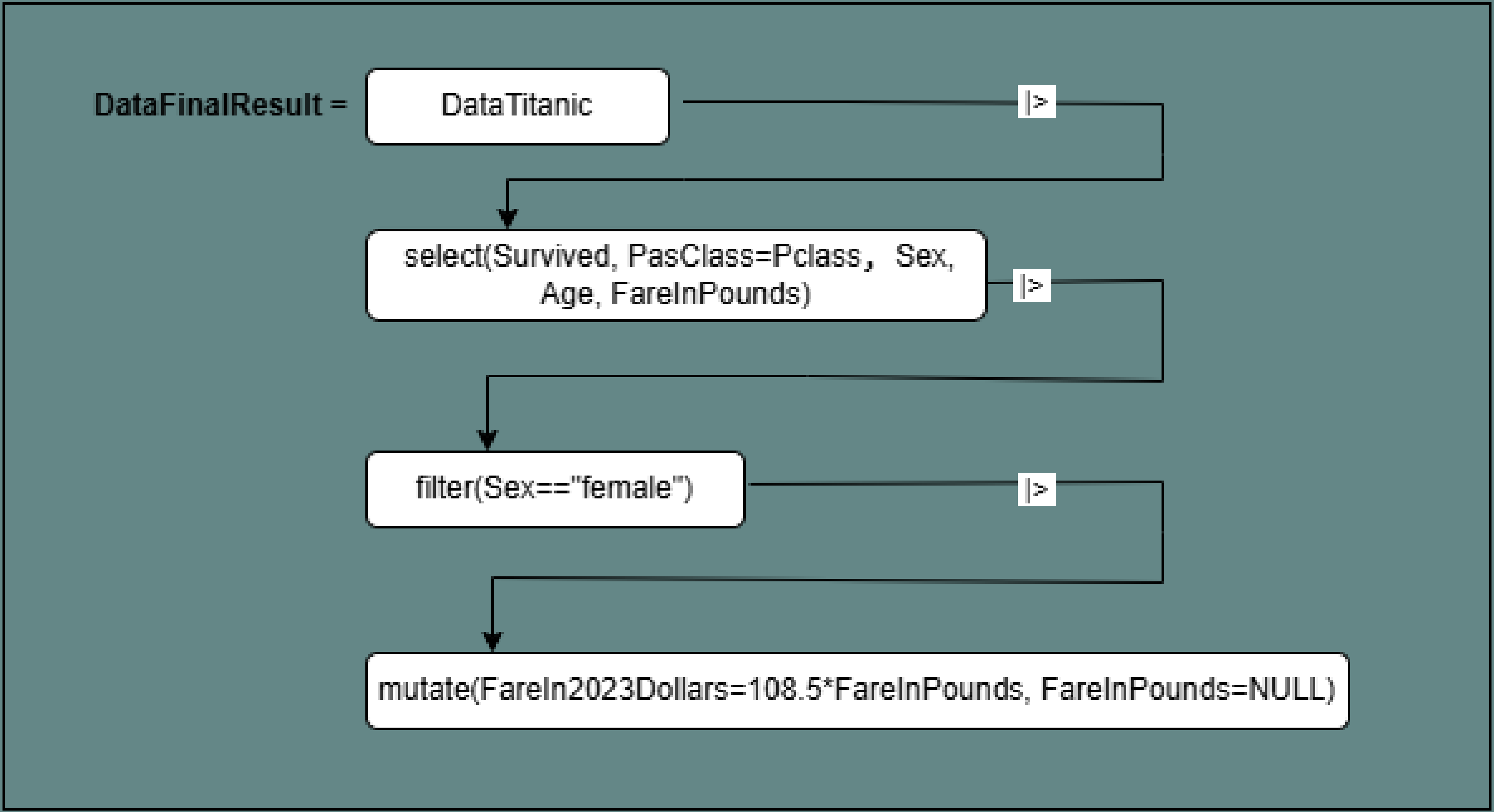
How did we get there:

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


DataFinalResult =
DataTitanic
|>
select(Survived, PasClass=Pclass， Sex, Age, FareInPounds)
|>
filter(Sex=="female")
|>
mutate(FareIn2023Dollars=108.5*FareInPounds, FareInPounds=NULL)

Piping Schema

Textbook

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 `%>%`.

R Code | Start Over | Run Code

```r
library(tidyverse)
DataTitanicFinal=DataTitanic |>
                 select(Survived, PasClass=Pclass , Sex, Age, FareInPounds) |>
                 filter(Sex=="female") |>
                 mutate(FareIn2023Dollars=108.5*FareInPounds, FareInPounds=NULL)
head(DataTitanicFinal)
```

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 Code   Start Over   Run Code

```python
import pandas as pd
DataTitanicPy=pd.read_csv("Data/Titanic.csv")
DataTitanicPy=DataTitanicPy[["Sex", "FareInPounds", "Survived"]] \
             .rename(columns={"Survived": "Surv"}) \
             .assign(FareInDollars = DataTitanicPy["FareInPounds"] * 108.5) \
             .query("Sex=='female'")
MeanFareWomen=DataTitanicPy["FareInDollars"].mean()
print(MeanFareWomen)
```

Textbook

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
```

Python Code Start Over Run Code

```python
import polars as pl
DataTitanicPy=pl.read_csv("Data/Titanic.csv", infer_schema_length=None) \
             .select(pl.col("Survived").alias("Surv"), pl.col("Sex"), pl.col("FareInPounds"),
                     (pl.col("FareInPounds")*108.5).alias("FareInDollars")) \
             .filter(pl.col("Sex") == "female")
MeanFareWomen=DataTitanicPy["FareInDollars"].mean()
print(MeanFareWomen)
```

Note, `polars` is currently not supported in WASM

Textbook

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`:

R Code Start Over Run Code

```r
DataMale = DataTitanic |>
           select(@@@) |>
           filter(@@@)
head(DataMale)
```

Textbook

THE FEMALE DATA FRAME

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

R Code Start Over Run Code

```r
DataFemale = DataTitanic |>
             select(@@@) |>
             filter(@@@)
head(DataFemale)
```

Textbook

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)`.

R Code Start Over Run Code

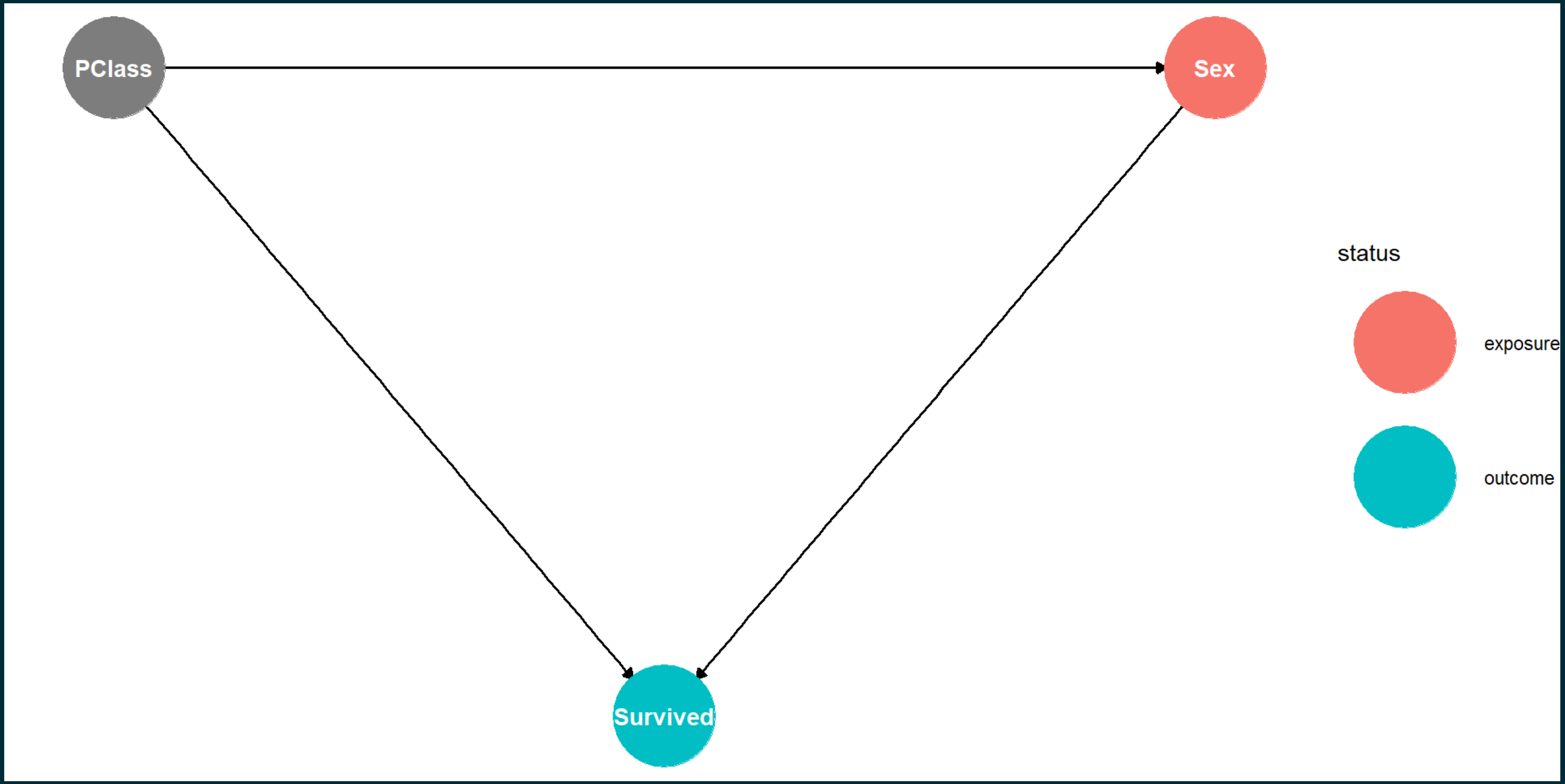
```r
PropFemSurv = @@@
PropMaleSurv = @@@
cat(@@@, @@@)
cat(@@@, @@@)
```

Textbook

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`)

R Code Start Over Run Code

```r
DataFemaleClass1=DataTitanic |>
                 select(@@@, @@@, PasClass=Pclass) |>
                 filter(@@@) |>
                 filter(@@@)
DataMaleClass1=DataTitanic |>
               select(@@@, @@@, PasClass=Pclass) |>
               filter(@@@) |>
               filter(@@@)
cat(" Female Survival Proportion in Class 1:", mean(DataFemaleClass1$Survived), "\n",
    "Male Survival Proportion in Class 1:", mean(DataMaleClass1$Survived))
```

Textbook

SURVIVAL RESEARCH FOR PASSENGER CLASS 2

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

R Code Start Over Run Code

```r
DataFemaleClass2=DataTitanic |>
                 select(@@@, @@@, PasClass=Pclass) |>
                 filter(@@@) |>
                 filter(@@@)
DataMaleClass2=DataTitanic |>
               select(@@@, @@@, PasClass=Pclass) |>
               filter(@@@) |>
               filter(@@@)
cat(" Female Survival Proportion in Class 2:", mean(DataFemaleClass2$Survived), "\n",
    "Male Survival Proportion in Class 2:", mean(DataMaleClass2$Survived))
```

Textbook

SURVIVAL RESEARCH FOR PASSENGER CLASS 3

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

R Code Start Over Run Code

```r
DataFemaleClass3=DataTitanic |>
                 select(@@@, @@@, PasClass=Pclass) |>
                 filter(@@@) |>
                 filter(@@@)
DataMaleClass3=DataTitanic |>
               select(@@@, @@@, PasClass=Pclass) |>
               filter(@@@) |>
               filter(@@@)
cat(" Female Survival Proportion in Class 3:", mean(DataFemaleClass3$Survived), "\n",
    "Male Survival Proportion in Class 3:", mean(DataMaleClass3$Survived))
```

Textbook