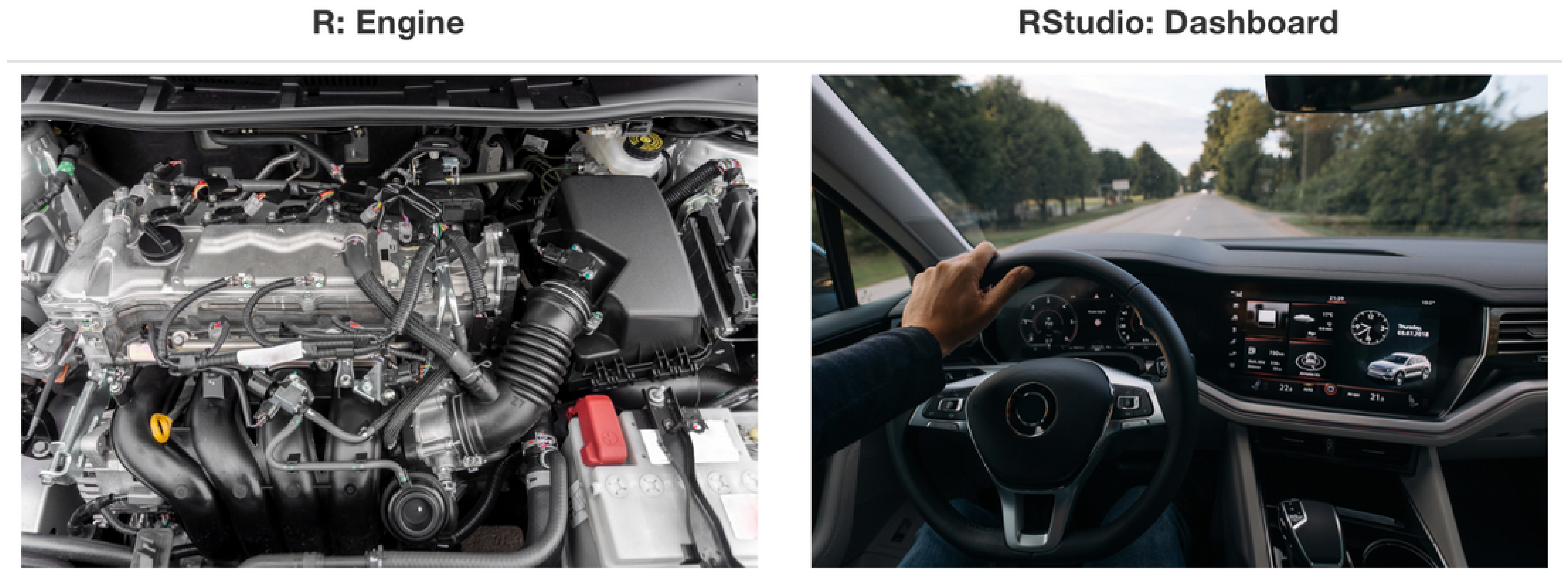
Intro to R

Amanda and Pilar

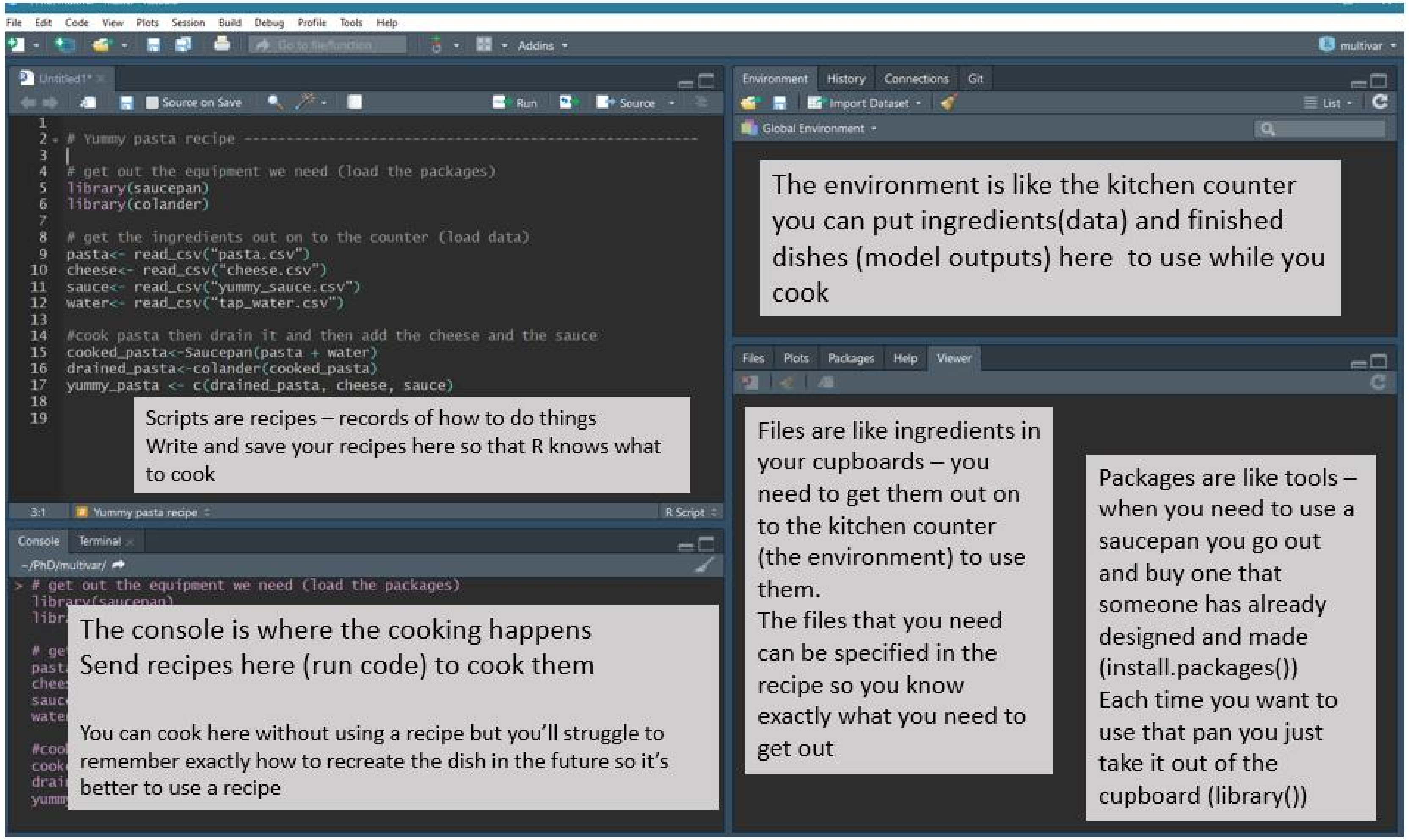
08/31/2020

R and R studio



Source: Modern Drive

R Studio Interface



```
# Yummy pasta recipe -----------------------------------------------------

# get out the equipment we need (load the packages)
library(saucepan)
library(colander)

# get the ingredients out on to the counter (load data)
pasta<- read_csv("pasta.csv")
cheese<- read_csv("cheese.csv")
sauce<- read_csv("yummy_sauce.csv")
water<- read_csv("tap_water.csv")

#cook pasta then drain it and then add the cheese and the sauce
cooked_pasta<-Saucepan(pasta + water)
drained_pasta<-colander(cooked_pasta)
yummy_pasta <- c(drained_pasta, cheese, sauce)
```

Scripts are recipes – records of how to do things
Write and save your recipes here so that R knows what to cook

```
> # get out the equipment we need (load the packages)
  library(saucepan)
  libr

  # ge
  past
  chee
  sauc
  wate

  #coo
  cook
  drai
  yumm
```

The console is where the cooking happens
Send recipes here (run code) to cook them

You can cook here without using a recipe but you'll struggle to remember exactly how to recreate the dish in the future so it's better to use a recipe

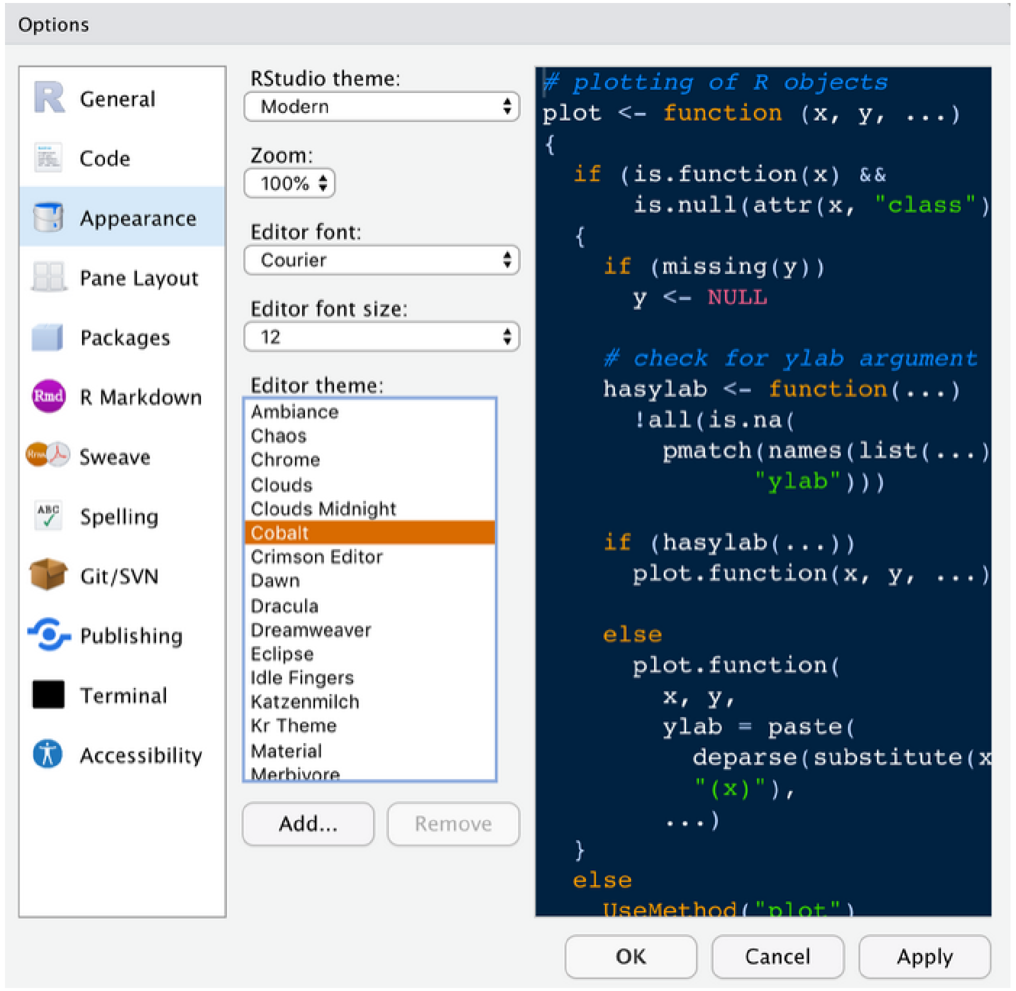
The environment is like the kitchen counter you can put ingredients(data) and finished dishes (model outputs) here to use while you cook

Files are like ingredients in your cupboards – you need to get them out on to the kitchen counter (the environment) to use them.
The files that you need can be specified in the recipe so you know exactly what you need to get out

Packages are like tools – when you need to use a saucepan you go out and buy one that someone has already designed and made (install.packages())
Each time you want to use that pan you just take it out of the cupboard (library())

Source: R Ladies

Customizing R Studio



RStudio theme: Modern

Zoom: 100%

Editor font: Courier

Editor font size: 12

Editor theme:
- Ambiance
- Chaos
- Chrome
- Clouds
- Clouds Midnight
- [✓ Cobalt]
- Crimson Editor
- Dawn
- Dracula
- Dreamweaver
- Eclipse
- Idle Fingers
- Katzenmilch
- Kr Theme
- Material
- Merbivore

```
# plotting of R objects
plot <- function (x, y, ...)
{
  if (is.function(x) &&
      is.null(attr(x, "class")
  {
    if (missing(y))
      y <- NULL

    # check for ylab argument
    hasylab <- function(...)
      !all(is.na(
        pmatch(names(list(...)
               "ylab")))

    if (hasylab(...))
      plot.function(x, y, ...)

    else
      plot.function(
        x, y,
        ylab = paste(
          deparse(substitute(x
          "(x)"),
        ...)
  }
  else
    UseMethod("plot")
```

R as a calculator

```r
2+2
```

```
## [1] 4
```

```r
2/3
```

```
## [1] 0.6666667
```

```r
log(10)
```

```
## [1] 2.302585
```

```r
abs(-1)
```

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

```r
sqrt(9)
```

```
## [1] 3
```

Logical statements

Common logical operators:

- == (is equal)
- != (not equal)
- < (greater than), > (less than)
- & (and), | (or)

Logical statements

```r
1 < 2
```

```
## [1] TRUE
```

```r
1 == 2
```

```
## [1] FALSE
```

```r
2 != 2
```

```
## [1] FALSE
```

```r
1 > 0 & 2 > 0
```

```
## [1] TRUE
```

```r
1 < 0 | 2 <= 3
```

```
## [1] TRUE
```

PRACTICE

1) Calculate square root of 109090

2) What number is larger: The log of 2000 or the square root of 51? (Try to do this in one line only)

3) What is the maximum number between: the square root of 200, seven times 2, and log of 3000 (Try to do this in one line only)

Objects

- R is based on objects: variables, functions, dataframes, etc.

- Objects can be of different types (or "class"). The types of operations you can perform will depend on the class.

- Most common class of objects: numeric, character, logical, matrix, data.frame, list, function.

Objects

We usually want to store objects so we can work with them later. We do this by attributing a name to that object.

```r
year <- 2020
```

What type of object is "year"?

```r
class(year)
```

```
## [1] "numeric"
```

```r
prof_name <- "Jean"
prof_name
```

```
## [1] "Jean"
```

```r
class(prof_name)
```

```
## [1] "character"
```

Vectors (combining objects)

A vector is a combination of more than one object (of the same class). We can create vectors with c() which stands for "combine".

```
names <- c("Jean", "Amanda", "Pilar")
class(names)
```

```
## [1] "character"
```

```
grad_year <- c(2016, 2021, 2023)
class(grad_year)
```

```
## [1] "numeric"
```

```
area <- c("Methods", "American Politics", "Comparative Politics")
```

PRACTICE

1) Create an object with your first name and a second object with your last name.

2) Create a vector that contains your first and last name. (Try the function "paste" too)

Functions to describe numeric vectors

summary()

mean()

median()

sd()

var()

Dataframes

- Data frames are the core data structure in R. A data frame is a list of named vectors with the same length.
    - Data frames are *heterogenous*: the vectors in a data frames can each be of a different data type.
    - Columns are typically variables and rows are observations.
    - You can make make data frames with `data.frame()`, or by combining vectors with `cbind()` or `rbind()`.

Dataframes (combining vectors)

```r
dataset <- cbind(names, grad_year, area)
dataset
```

```
##      names    grad_year area
## [1,] "Jean"   "2016"    "Methods"
## [2,] "Amanda" "2021"    "American Politics"
## [3,] "Pilar"  "2023"    "Comparative Politics"
```

```r
dataset <- data.frame(names = c("Jean","Amanda","Pilar"),
                 grad_year = c(2016, 2021, 2023),
                 area = c("Methods", "American Politics", "Comparative Politics"))
dataset
```

```
##    names grad_year                 area
## 1   Jean      2016              Methods
## 2 Amanda      2021    American Politics
## 3  Pilar      2023 Comparative Politics
```

Dataframes

Data frames can be indexed by using variable/column names: `df$var` or `df["var"]`.

```
dataset$names
```

```
## [1] "Jean"   "Amanda" "Pilar"
```

```
dataset$grad_year[dataset$names == "Jean"]
```

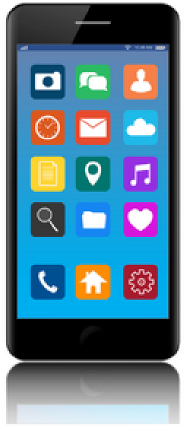
```
## [1] 2016
```

```
dataset$new_grad_year <- dataset$grad_year - 2

dataset$new_grad_year
```

```
## [1] 2014 2019 2021
```

Packages

R: A new phone	R Packages: Apps you can download
	

On phone	On R
Download app	install.packages("")
Open app	library()

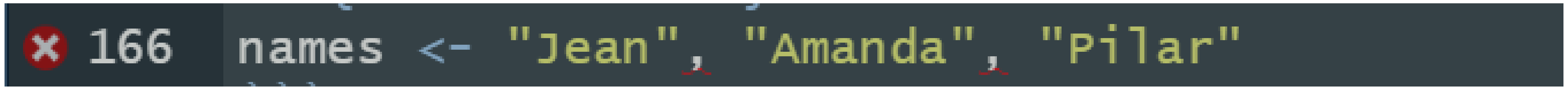
Basics of R: Errors and Warnings

Error: If you get an error, the command will not be executed. This can be due to many things (including silly spelling mistakes, missing parentheses, etc.)

```
names <- "Jean", "Amanda", "Pilar"
```

```
## Error: <text>:1:16: unexpected ','
## 1: names <- "Jean",
##                    ^
```

In some occasions, R will warn you about this even before executing the code.



```
166  names <- "Jean", "Amanda", "Pilar"
```

Basics of R: Errors and Warnings

If you get a warning, the command will still be executed, but with some tweaking.

```
x <- as.numeric(c("1", "2", "X"))
```

```
## Warning: NAs introduced by coercion
```

Make sure that "tweaking" still gets you the result you want.

```
x
```

```
## [1]  1  2 NA
```

Where to find help

1) In R:

- type ?mean
- In "Help" window on lower-right pane

2) Google

3) StackExchange, StackOverflow

4) Package documentation, Package vignettes

PRACTICE

1) Install and load the package "dplyr"

2) Find the help file for the command "mutate"