Intro to R for Data Analysis

Kristin Yeager, Manager, Statistical Consulting

November 18, 2021

In this workshop

- What is R, and why should I use it?
- What is RStudio, and why should I use it?
- Key terms
- Worked example of the R workflow
    - Setting up RStudio project
    - Importing data
    - Data cleaning/data management
    - Data analysis

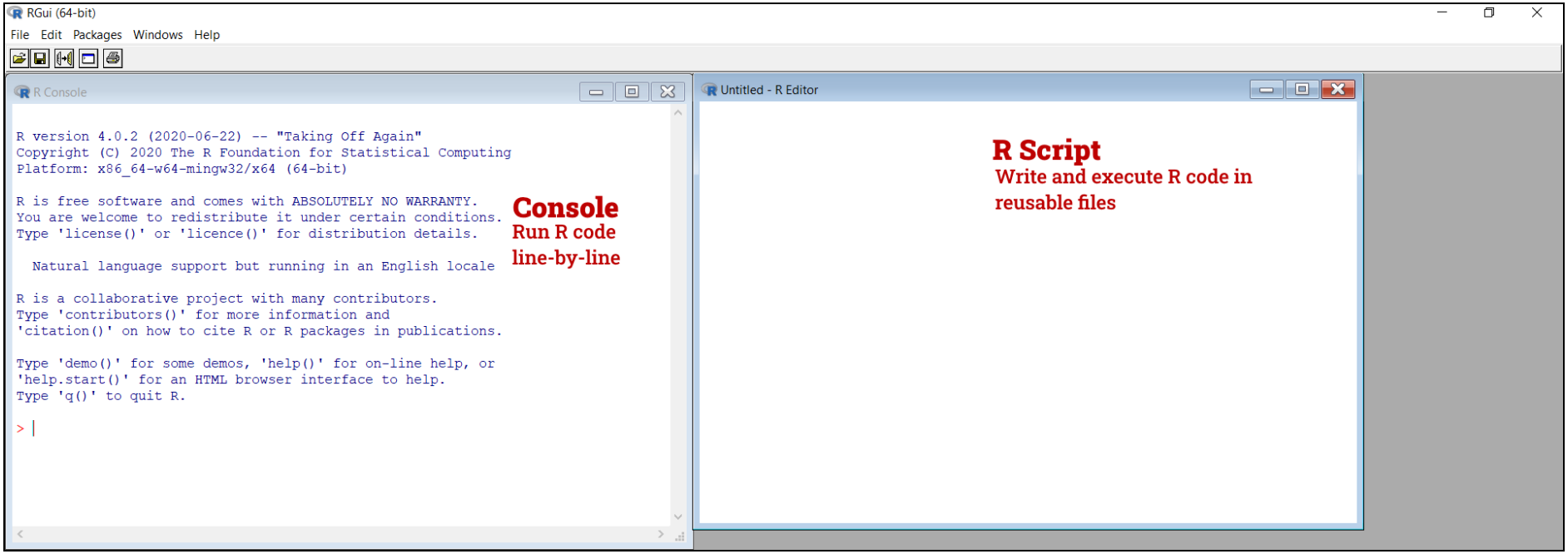
What is R?

And why you should use it

About R

- What it is: Free and open source statistical software and statistical programming language
- Cost: Free!
- Operating systems: Windows, Mac, Linux
- To install: cran.r-project.org

The R user interface



- The **console** is the "command line" interface part of R – type code into it, press enter, get results back
- The console is fine for "playing around", but most of the time, you'll put everything into an **R script** – a file containing executable R code

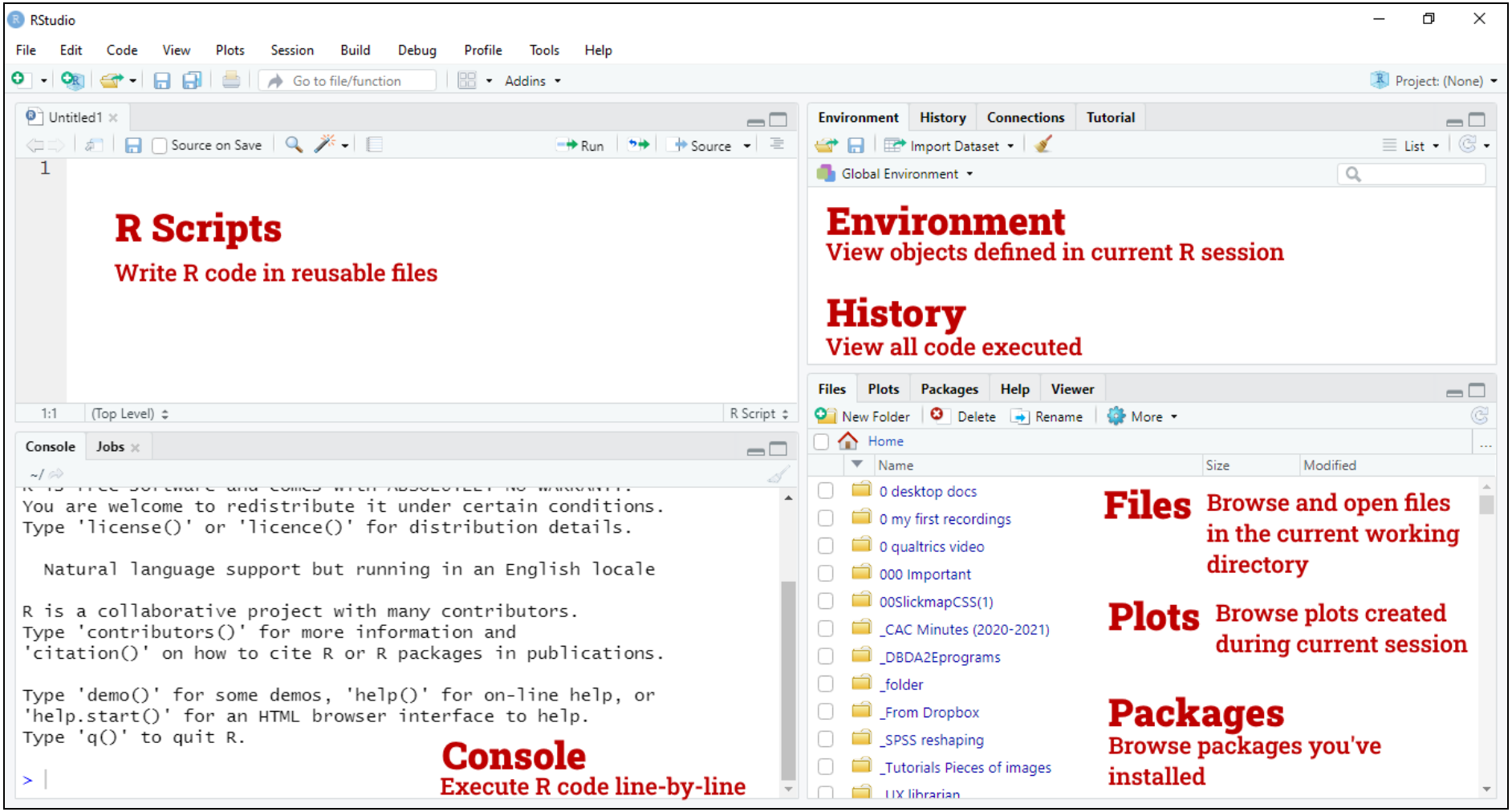
Why use R?

- Free and open-source
- Reproducibility and communication (tools for combining written report with the code used to generate tables, figures, analysis)
- Beautiful, publication-quality graphics (especially ggplot2)
- Frequently the first place to find new and cutting-edge statistical methods, often before they're available in big-name stat software programs
- Large and active user community
  - Better free tutorials
  - Active forums and social media community with users at all experience levels
  - Opportunities to promote yourself professionally

About RStudio

- Completely optional but can make writing R code much easier!
- What it is: Interactive development environment (IDE) for R
  - View objects in your R environment
  - Syntax highlighting and auto-completion features
  - Keyboard shortcuts for writing code
  - And much, much more
- Requires R to work – can't be used without first installing R
- Cost: Free
- Operating systems: Windows, Mac, Linux
- To install: rstudio.com/products/rstudio/download

The RStudio User Interface



R syntax

Necessary vocabulary before we can get to the “good stuff”

Anatomy of a dataset - Data structure

	A	B	C	D	E	F	G	H	I
1	ids	bday	Gender	Age	Athlete	Smoking	Height	Weight	Mile
2	20055	8/6/1995	0	24	1	1	64.5	127.5	494
3	20075		0		0	1	74	195.5	441
4	20087		0		0	1	68.5	153	
5	20088	1/8/2001	1	18	0		67.5	158.5	514
6	20135	12/26/1998	0	20	0	1	75	152	450
7	20161	12/2/2000	1	18	1	1	73.5	177.5	403
8	20188	8/6/2001	0	18	0	1	76	181	
9	20215	8/5/1997	0	22	0	2	76	155.5	563
10	20250	12/14/1994		24	0	1	66.5	119	653
11	20354	8/5/1999	1	20	0	1	62.5	147.5	684
12	20640	5/31/2000	1	19	0	1	63	161	798
13	20739	6/23/1999	1	20	0	1	72	190.5	612

- First row contains variable names (no spaces or special characters; numbers, periods, underscores OK; R is case-sensitive)
- Rows = observations (1 row = 1 respondent or unit)
- Columns = variables (1 column = 1 distinct variable)

Objects & Environment

- **Objects** are named “things” you’ve created in R – can be manipulated, modified, plotted, etc.
- Objects are created using the assignment operator: `<-`
- There are MANY types of objects in R – in this session, we’ll be focusing on **dataframes** and **vectors**
  - Dataframes are R’s representation of the data set structure from the previous page
  - Vectors are sets of numbers (or other data values) that operations can be applied to
    - Each column within a dataframe is technically a vector

Objects & Environment

Example: Create new objects called `new_object` and `new_vector` and see what happens when we add them:

```
new_object <- 5
print(new_object)
```

```
## [1] 5
```

```
new_vector <- c(3, 1, 1, 0)
print(new_vector)
```

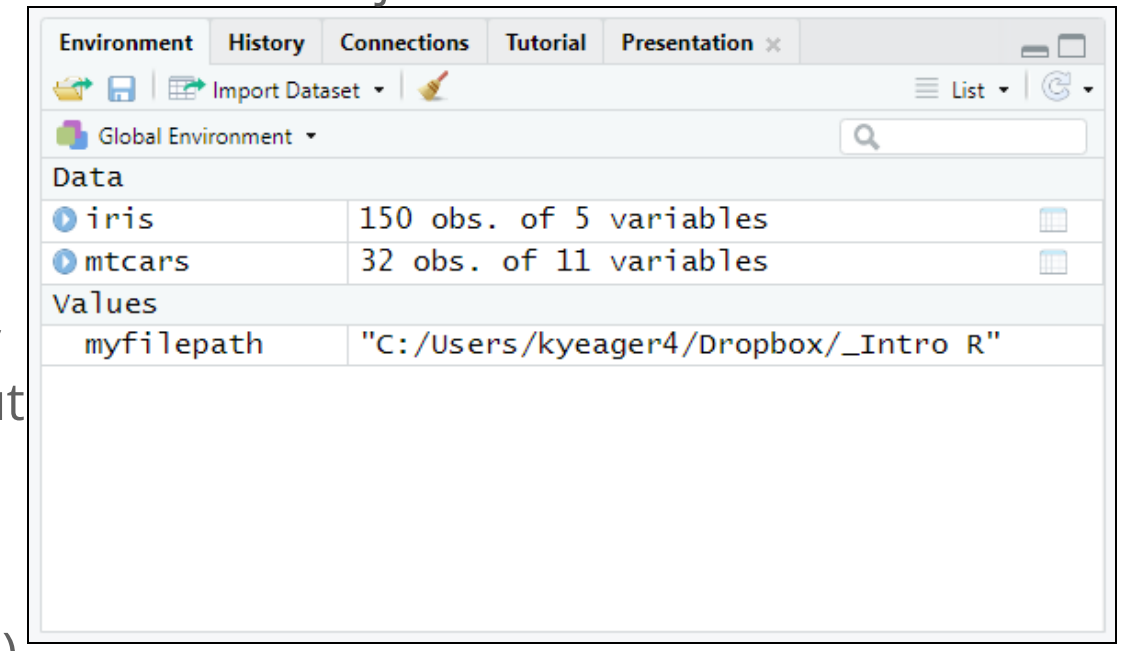
```
## [1] 3 1 1 0
```

```
new_vector + new_object
```

```
## [1] 8 6 6 5
```

Objects & Environment

- The **environment** is the collection of all R objects that have been defined during the current session
- If you're using RStudio, you can view all objects in the Environment panel
- The environment is normally cleared when you close R, but you can choose to save the environment so it still exists the next time you launch R (normally not recommended)



Functions and arguments

- R syntax is generally driven by **functions**, which are pre-programmed tasks that R can do
- Most (but not all) functions have **arguments** that can modify or control the behavior of the function

Example: Compute the mean of `new_vector` from a couple pages ago:

```
mean(x=new_vector, na.rm=TRUE)
```

- `mean()` is the name of the function
- `x` and `na.rm` are arguments of `mean()`
  - `x` is the vector of data to compute the mean of
  - `na.rm=TRUE` tells R to ignore any missing values in the input vector

Functions and arguments

Example: Compute the mean of `new_vector` from a couple pages ago:

```
#Reminder of the values of new_vector
print(new_vector)
```

```
## [1] 3 1 1 0
```

```
mean(x=new_vector, na.rm=TRUE)
```

```
## [1] 1.25
```

Functions and arguments

- You can open a function's documentation page using the function `help()`:

```
help("mean")
help("read.csv")
```

R packages

- R packages are user-created modules of code intended to fulfill a narrow task or purpose
  - e.g. Reading Excel files into R, fitting zero-inflated negative binomial regression models, interfacing with web APIs, …
- Packages are shared through *repositories*, of which CRAN is "main" one
  - See CRAN Task Views to explore R packages by topic
  - See also: Bioconductor
- An R user "installs" an R package from the online repository to their local package directory using the function `install.packages()`
- Once a package has been installed, it can be used in your R scripts – simply load the package using `library()`

Worked Example

Today's sample data

Tutorial sample data:

- Simulated survey dataset we use for our online tutorials
- n=435 "college students"
- Available in CSV, TXT, SAS, SPSS formats
- Download from our website: libguides.library.kent.edu/SPSS/ (direct download link for CSV, direct download link for SPSS format)

Research questions we'll consider today:

- Do students who study more get less sleep?
- Do underclassmen (freshmen and sophomores) get less sleep than upperclassmen (juniors and seniors)?

The R Data Analysis Workflow

1. Obtain data and set up project directory
2. Read/import data into R
   - Determine if special packages needed to read data
3. Data management
   - Check data quality
   - Compute new variables
   - Filter rows
   - Select columns
4. Data analysis
   - Descriptive statistics
   - Plots
   - Statistical tests and models

Working in R, part 0

Before you open the program

Set up your project

We'll start by setting up a project folder, where everything related to this project (data, R scripts, output, etc) will go. In addition to staying organized, this will make it easier to write our scripts (and our scripts will be less likely to break if we move files around!)

For a big project, consider using subdirectories inside your project folder to keep everything organized:

- project
    - code
    - data
    - plots
    - output
    - reports

RStudio Projects

- Reason to do this: Can declare a folder/directory on your computer as an "RStudio project"
- Allows you to read and write files using *relative paths* instead of *absolute paths*
    - Relative paths tell your computer what folder to start looking in for a given file

Example: Absolute file path on Windows:

```
"C:/Users/yourusername/yourprojectfolder/mydata.csv"
```

Example: Absolute file path on Mac:

```
"/Users/yourusername/yourprojectfolder/mydata.csv"
```

Example: Relative file path after creating an RStudio project in the "yourprojectfolder" folder:

```
"mydata.csv"
```

Working in R, part 1

Importing text-based data

Step 1: Get data into R

- ***Data frames*** are R's object type for traditional "data sets"
- Base R function to import text and CSV files as a data frame: `read.table` or `read.csv`

```
mydata <- read.csv(file="Sample_Dataset_2019.csv")
```

This code creates an object named `mydata` containing the data from this file as a data frame.

Step 1: Get data into R

- **What if the data isn't in CSV or plaintext format?**
- Oftentimes, it is necessary (or simply a LOT easier) to find an R package that can read other data formats. Here's my package recommendations for some of the more common formats I see (generally easy to get working on Mac and Windows):
  - Excel: `readxl`
  - SAS, SPSS, and Stata 13+: `haven`

Example: Reading SPSS-format dataset into R using package `haven` (relative file path)

```
library(haven)
mydata <- haven::read_spss(file="Sample_Dataset_2019.sav")
```

Step 2: Look at the data we imported into R

How do we know if our import was successful?

The following functions take a data frame as an argument, and return previews or summaries about that data frame:

- View the dataframe as a spreadsheet using `View(mydata)`
- Print variable types using function `str(mydata)` (str is short for **str**ucture)
- Print minimum, maximum, median, missing values for all variables in dataframe using `summary(mydata)`
- Print first 5 rows of dataframe using `head(mydata)`
- Print last 5 rows of dataframe using `tail(mydata)`
- Print names of variables in dataframe using `names(mydata)`

Tip

Never use the `attach()` function, even if you see it in an online tutorial and it looks like it would make things easier.

Step 3: Access variables within our dataset

- Some functions expect a dataframe; other functions expect a *vector* of observations – that is, a single, specific column from a dataframe
- Variables are simply a named vector inside our dataframe

When we need to access a column of a dataframe as a vector, we use the `$` operator to extract it:

```
mydata$Mile
```

This syntax can be combined with the assignment operator to add new variables to a dataframe, or edit existing variables in a dataframe:

```
mydata$Mile_minutes <- mydata$Mile/60
```

Step 4: Subsetting rows or columns of our dataset

Normally when we want to access specific **rows**, what we really want is to filter our data by some condition. We can do this using function `subset()`:

```
freshmen_only <- subset(mydata, subset = Rank==1)
```

The first argument to `subset` is the name of the dataframe. The second argument is a **logical condition** for which rows to keep (here, `Rank==1`, i.e. keep only freshman).

If we instead want to drop or keep specific columns of our dataset, we can also use the subset function, but use its `select` argument:

```
grade_data <- subset(mydata, select = English:Science)
fitness_data <- subset(mydata, select = c(Athlete, Smoking, Mile))
data_dropbday <- subset(mydata, select = -bday)
```

Working in R, part 2

Summary statistics

Summary statistics for continuous numeric variables

- `mean()`, `sd()`, `min()`, `max()`, `median()`, `sum()`
  - The `na.rm=TRUE` argument
- Base R graphics: `hist()`, `boxplot()`

Summary statistics for categorical variables

- `table()`
- `prop.table()`
- Base R graphics: `barplot(table(...))`

Working in R, part 3

Installing an R package for plotting

Installing package ggplot2

- The ggplot2 package creates beautiful, publication-quality graphics using the *grammar of graphics* syntax
- To install the package:

```
install.packages("ggplot2")
```

Loading package ggplot2

After you've installed the package, you can load the package whenever you want to use it:

```
library(ggplot2)
```

We typically put these package loading statements at the start of our scripts.

Working in R, part 4

Data analysis

Examples

- Correlation using `cor`
- Scatterplot using `ggplot2`
- Linear regression using `lm()`

Questions?

Appendix A

R version used for this workshop

R version used for this workshop

```
devtools::session_info()[[1]]
```

```
##  setting  value
##  version  R version 4.1.2 (2021-11-01)
##  os       Windows 10 x64 (build 19042)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United States.1252
##  ctype    English_United States.1252
##  tz       America/New_York
##  date     2021-11-18
##  pandoc   2.14.1 @ C:/PROGRA~1/Pandoc/ (via rmarkdown)
```

```
devtools::package_info(c("haven", "ggplot2"), dependencies=FALSE)
```

```
##  package * version date (UTC) lib source
##  ggplot2   3.3.5   2021-06-25 [1] CRAN (R 4.1.0)
##  haven     2.4.3   2021-08-04 [1] CRAN (R 4.1.1)
##
##  [1] C:/Users/kyeager4/Documents/R/win-library/4.1
##  [2] C:/Program Files/R/R-4.1.2/library
```