WELCOME TO NDACAN MONTHLY OFFICE HOURS!

NATIONAL DATA ARCHIVE ON CHILD ABUSE & NEGLECT
DUKE UNIVERSITY & CORNELL UNIVERSITY

NDACAN

Children's Bureau
An Office of the Administration for Children & Families

- The session will begin at 11am EST
  - 11:00 - 11:30am – LeaRn with NDACAN (Introduction to R)
  - 11:30 - 12:00pm – Office hours breakout sessions
- Please submit LeaRn questions to the Q&A box
- This session is being recorded.
- See ZOOM Help Center for connection issues: https://support.zoom.us/hc/en-us
  - If issues persist and solutions cannot be found through Zoom, contact Andres Arroyo at aa17@cornell.edu.

LEARN WITH NDACAN

Created by SaRah SeRnakeR

WHY R?

- Built for statistical computing
- Compatible with all computing systems (Windows, Mac, Linux)
- Open-source, free
- State of the art graphics

MATERIALS FOR THIS COURSE

- Course Box folder (https://cornell.box.com/v/LeaRn-with-R-NDACAN-2024-2025) contains
  - Data (will be released as used in the lessons)
    - Census state-level data, 2015-2019
    - AFCARS state-aggregate data, 2015-2019
    - AFCARS (FAKE) individual-level data, 2016-2019
    - NYTD (FAKE) individual-level data, 2017 Cohort
  - Documentation/codebooks for the provided datasets
  - Slides used in each week's lesson
  - Exercises as that correspond to each week's lesson
  - An .R file that will have example, usable R code for each lesson – will be updated and appended with code from each lesson

MATERIALS FOR THIS COURSE

- Using R in Action as a guide and reference to go with slides
  - https://www.cs.uni.edu/~jacobson/4772/week11/R_in_Action.pdf
- Other useful resources
  - R for Data Science: https://r4ds.had.co.nz/
  - Intro to R for Social Scientists: https://jaspertjaden.github.io/course-intro2r/
  - Link to list of even more useful resources: https://guides.library.brandeis.edu/c.php?g=302090&p=2013481

WEEK 1: INTRODUCTION TO R

September 30, 2024

DATA USED IN THIS WEEK'S EXAMPLE CODE

- Census aggregate data from 2015-2019 (census_2015_2019.csv)
  - 8 columns: cy, stfips, state, st, sex, race6, hisp, pop
  - 6120 rows: population counts for each state from 2015-2019, over sex X race6 X hisp

- Publicly available from CDC Wonder: https://wonder.cdc.gov/single-race-population.html

PROGRAMMING IN R

- "R" is a *programming language*, specifically built for statistical computing and analyses
  - Open-source, fully free and downloadable through *The Comprehensive R Archive Network (CRAN)*

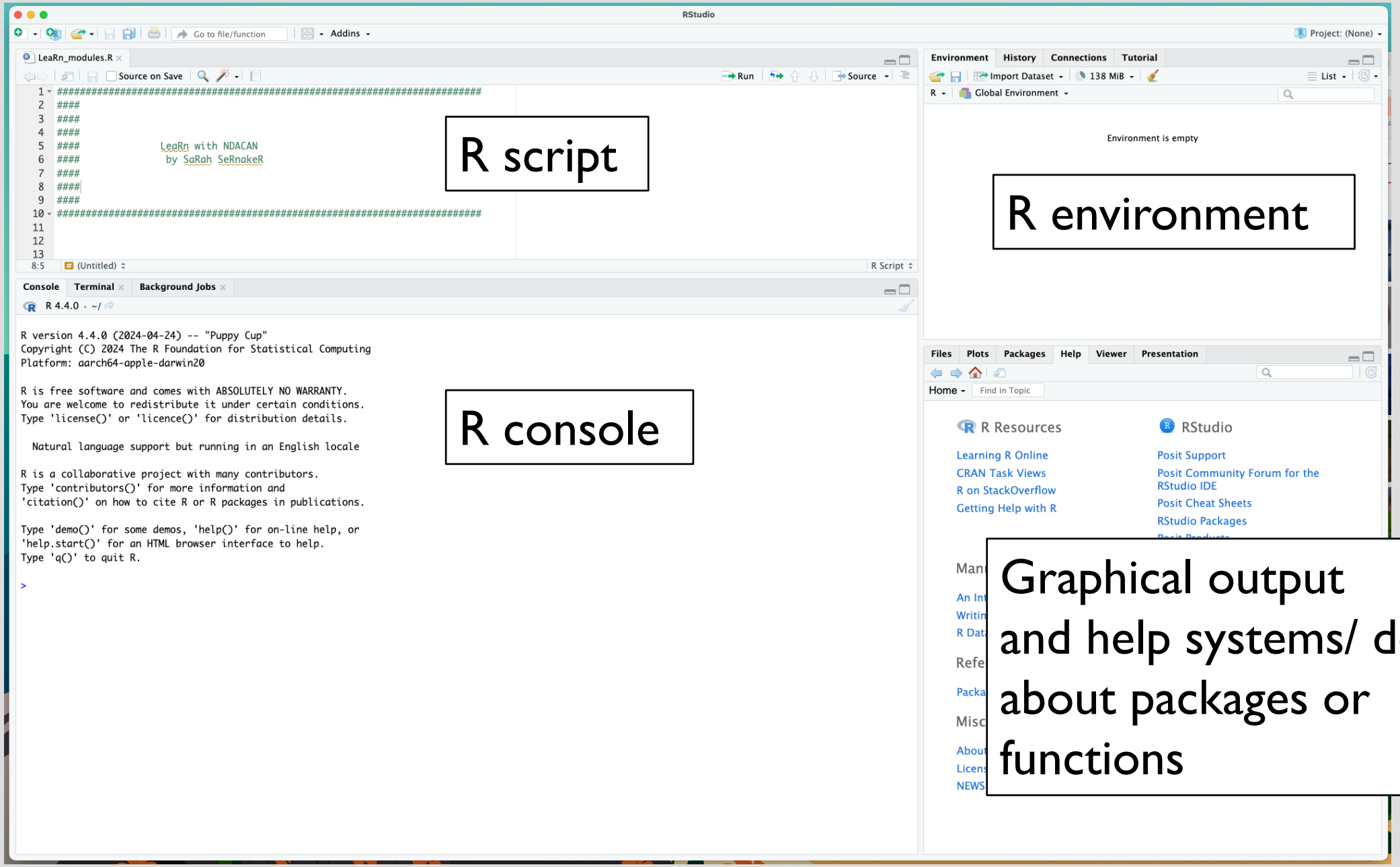
- RStudio is the *graphical user interface (GUI)* that makes writing R code and working with data much easier and more manageable

GETTING STARTED WITH R

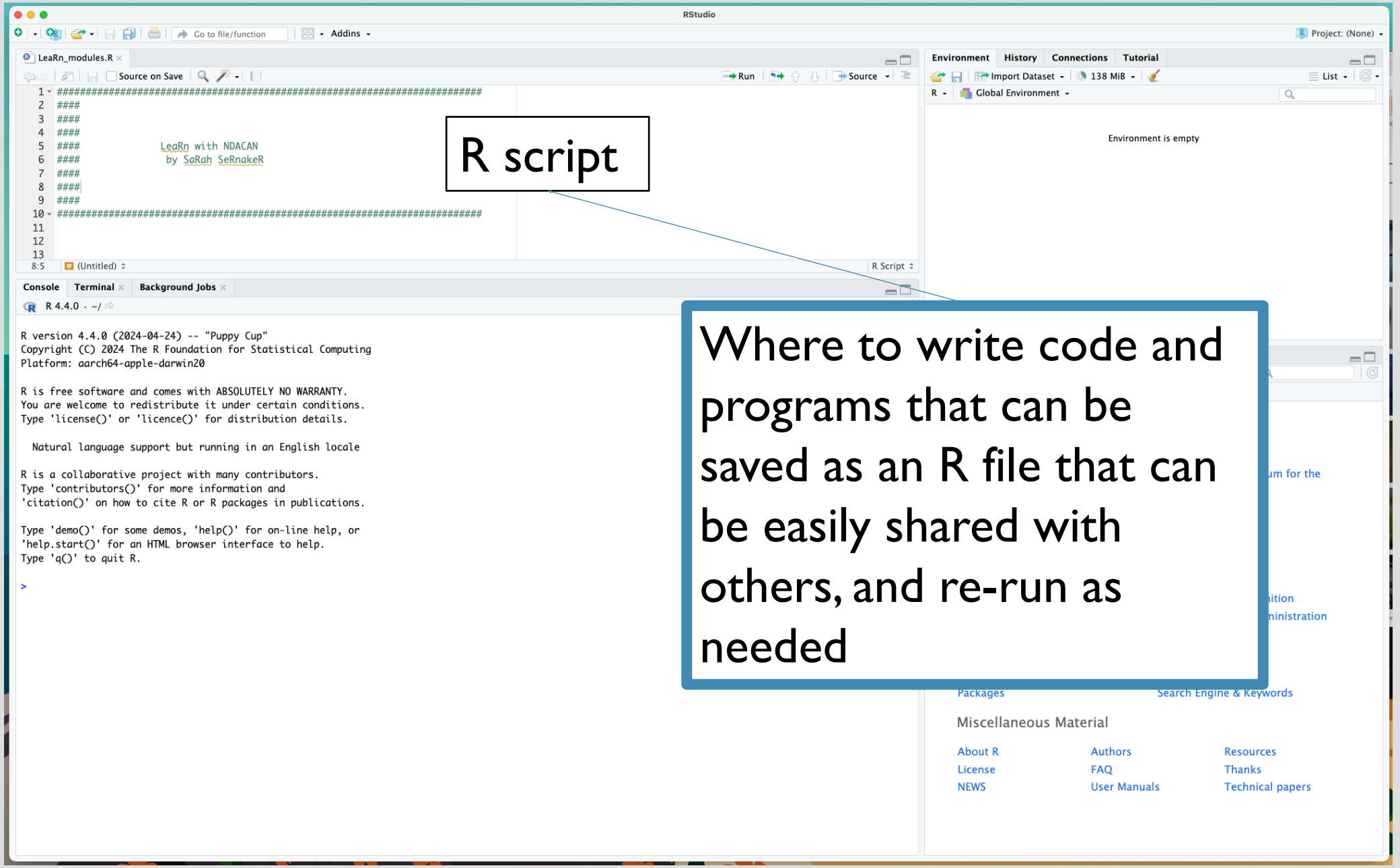
1. Download and install R programming language from CRAN:
   - https://cran.r-project.org/
2. Download and install RStudio from Posit:
   - https://posit.co/download/rstudio-desktop/
3. Open RStudio
4. Click the "File" button at the top, then "New File", then "R Script" to open a new R script to work in.
   - R scripts are where we write executable code and programs that we can save and re-run

R STUDIO INTERFACE

R STUDIO INTERFACE: OVERVIEW



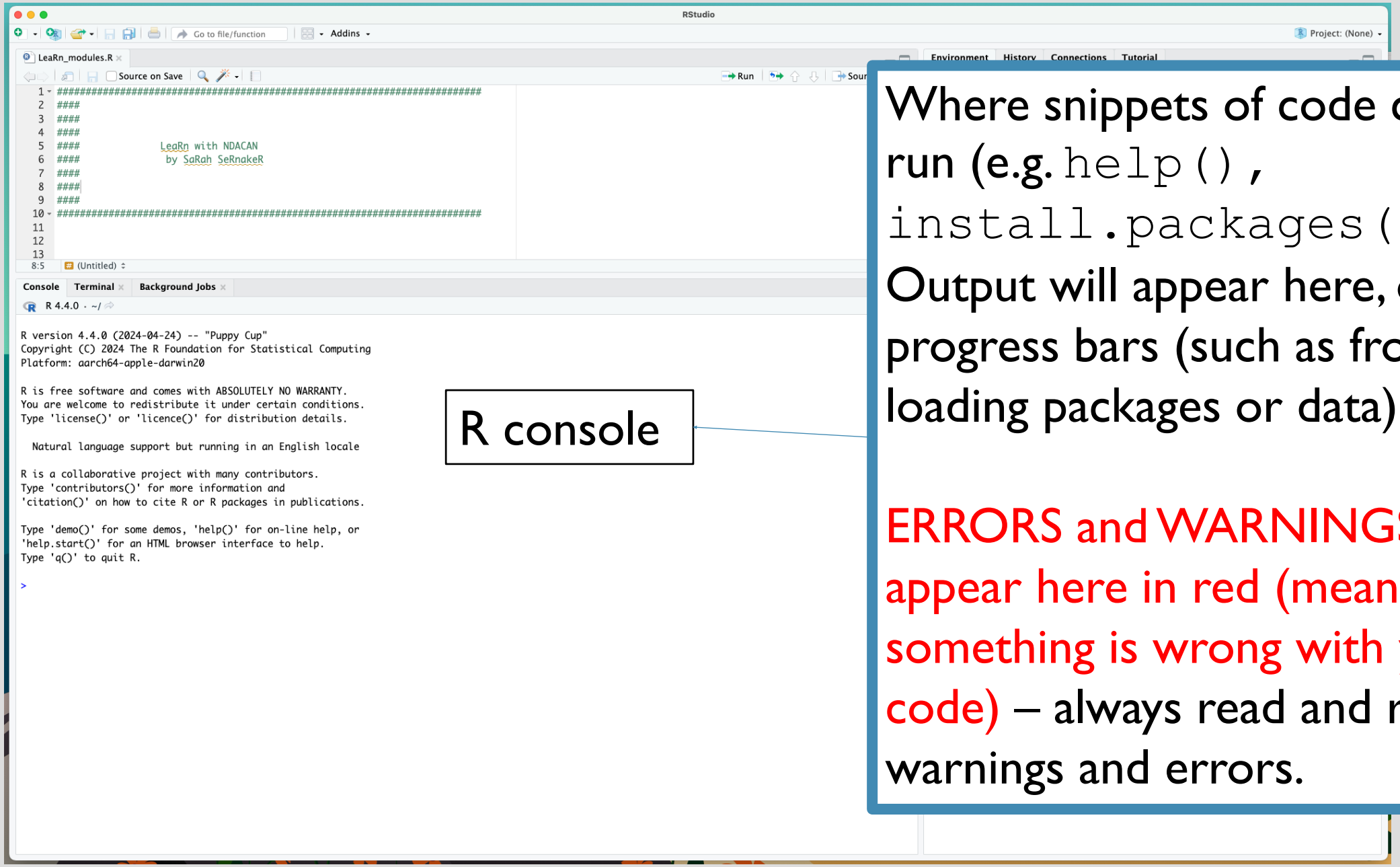
R STUDIO INTERFACE: R SCRIPT



R script

Where to write code and programs that can be saved as an R file that can be easily shared with others, and re-run as needed

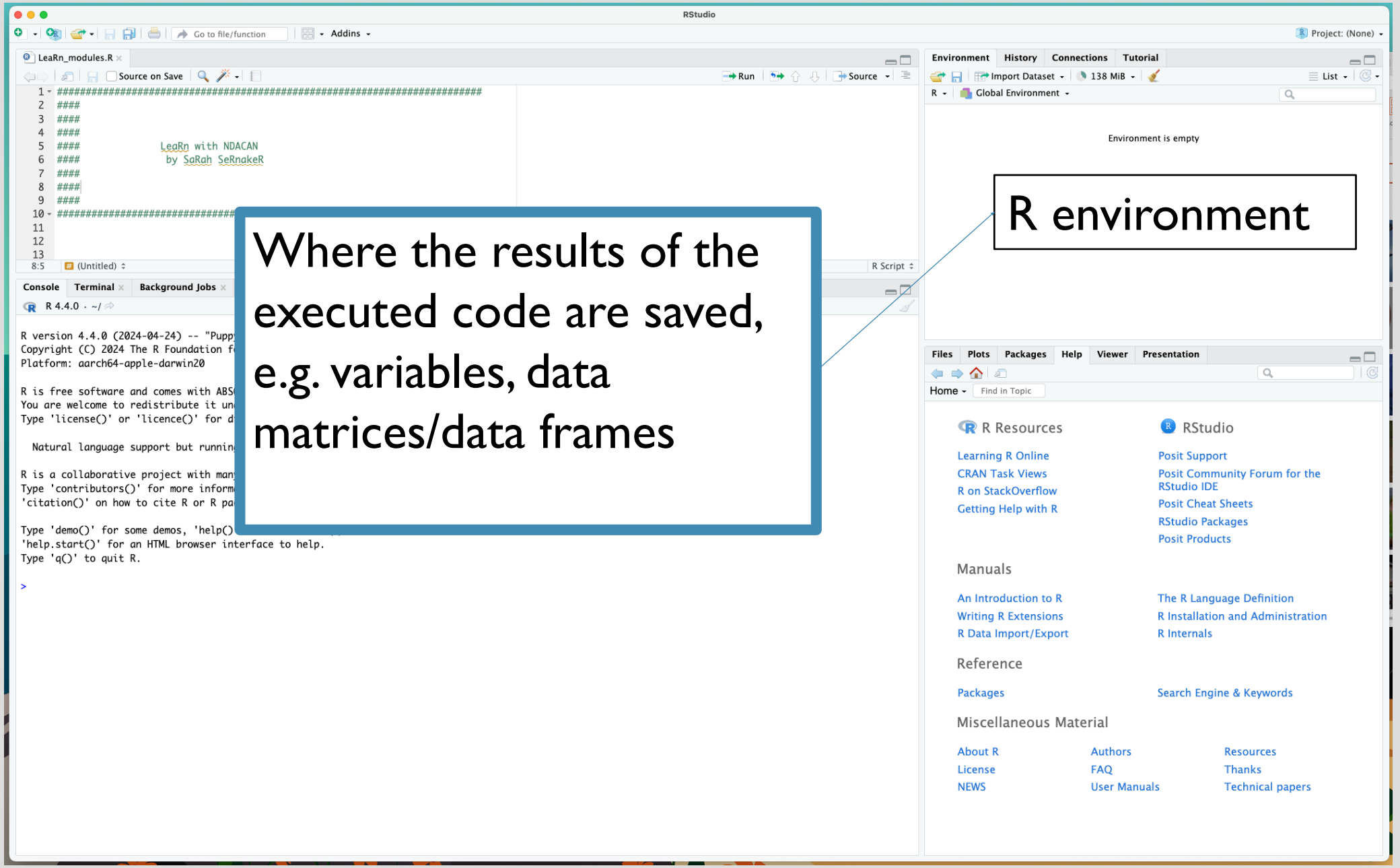
R STUDIO INTERFACE: R CONSOLE



Where snippets of code can be run (e.g. `help()`, `install.packages()`. Output will appear here, or progress bars (such as from loading packages or data).

ERRORS and WARNINGS will appear here in red (meaning something is wrong with your code) – always read and resolve warnings and errors.

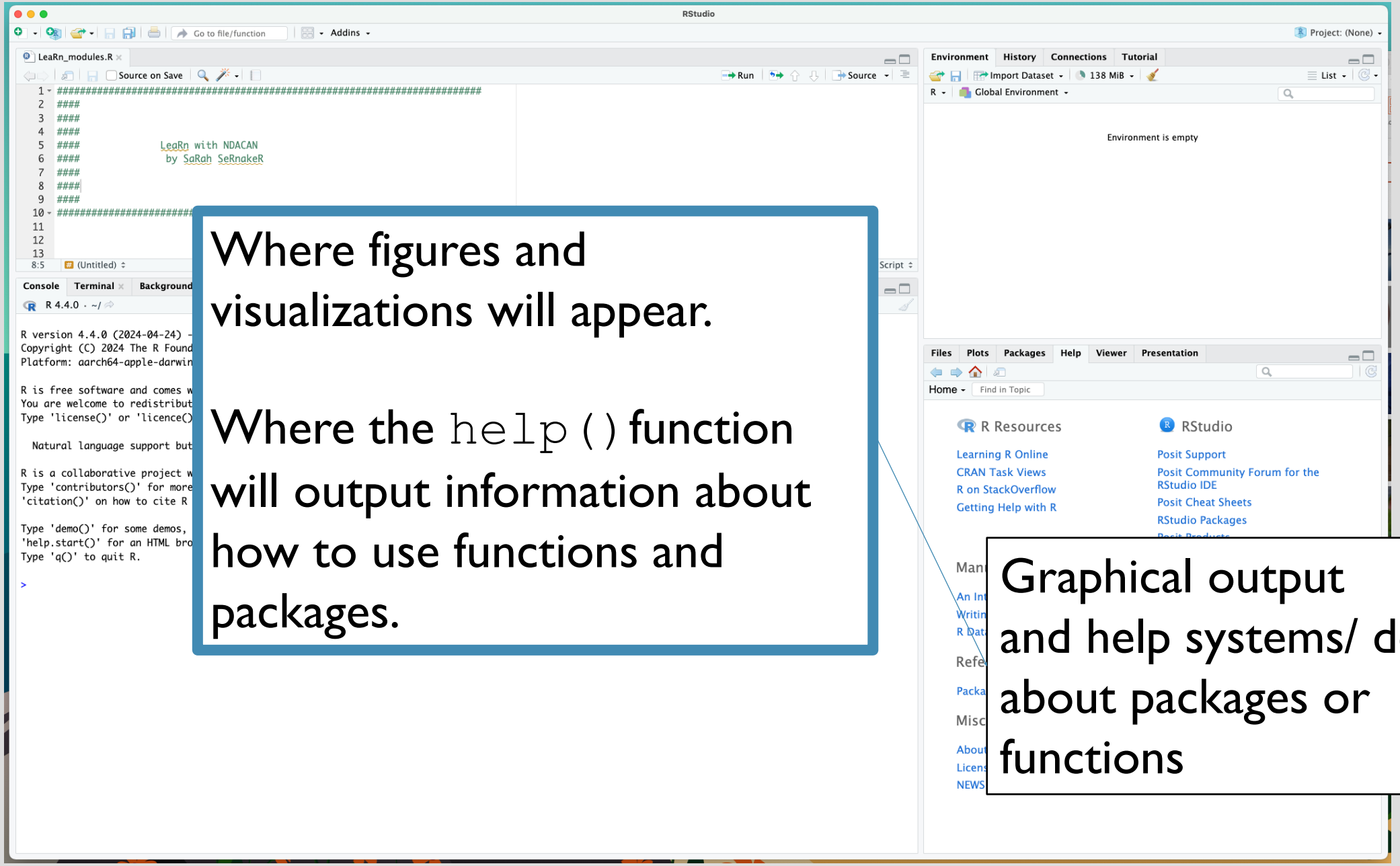
R STUDIO INTERFACE: R ENVIRONMENT



Where the results of the executed code are saved, e.g. variables, data matrices/data frames

R environment

R STUDIO INTERFACE: AREA FOR GRAPHICAL OUTPUT



Where figures and visualizations will appear.

Where the `help()` function will output information about how to use functions and packages.

Graphical output and help systems/ details about packages or functions

PROGRAMMING IN R

R FUNCTIONS AND PACKAGES

- There are a lot of built-in functions for basic statistical analyses – called "base R" functions

- Anything not already built-in to R must be installed from external packages from CRAN (or GitHub in some cases)
  - Tidyverse syntax and suite (`tidyverse`), advanced and niche methodologies (`survey`, `mice`), state of the art methods (`neuralnet`), advanced graphics (`ggplot2`)
  - `install.packages('PACKAGENAME')`

- Must load any needed (and already installed) packages at the start of your script/coding
  - `library(PACKAGENAME)  # note there are no quotes here`
  - Can also reference functions within library using double colons `LIBRARYNAME::FUNCTION_in_LIBRARYNAME()`

DOCUMENTATION AND HELP

- Package documentation
  - CRAN website
- Function documentation
  - Use the `help(FUNCTIONNAME)` function to access
  - Use `??SEARCHTERM` to browse functions in downloaded packages related to search term
- Any supplemental documentation relating to a package published elsewhere (just Google around)
  - For example, MICE has a great published article with lots more context and examples with it: https://www.jstatsoft.org/article/view/v045i03

PROGRAMMING CONCEPTS TO REFRESH

- Data types
  - String, characters, numeric, factor, ordered, logical (TRUE/FALSE)
  - Matrix, data frame, vector, lists
  - Missing/invalid values: NA, Null, Inf
- Variables
  - Assigning variables: e.g. x <- 3
  - Using and manipulating stored variables or objects
- Conditionals or loops
  - ‘if else’ statements
  - ‘for’ loops

PROGRAMMING CONCEPTS TO REFRESH

- Operators
  - <, <=, >, >=, ==, !=, !, a|b, a & b
- Coding style
  - Using spaces, indents, new lines in a way that makes code easier to read
- Comments
  - Thoroughly comment code using # with details about what code does and other relevant information – not just helpful for others but for future you!
- Seeking programming help
  - Google, Stack overflow
  - `help(FUNCTIONNAME)`
  - `??SEARCHTERM`

READING DATA INTO R

- Some external packages offer datasets included in the library (see https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html)

- Read from comma or tab separated files (.csv, .tab, .txt)
  - `read.table(file = "C:/pathname/DATATABLE.tab")`
  - `read.csv(file = "C:/pathname/DATATABLE.csv")`

- Read excel – need to use external package to read .xlsx files like `readxl`
  - `readxl::read_xlsx(file = "C:/pathname/DATATABLE.xlsx")`

- Read data from other programming language formats (Stata, SPSS, SAS) – need to use external package like `haven`
  - `haven::read_spss(file = "C:/pathname/DATATABLE.sav")`
  - `haven::read_stata(file = "C:/pathname/DATATABLE.dta")`
  - `haven::read_sas(file = "C:/pathname/DATATABLE.sas7bdat")`

CONSIDERATIONS WHEN WRITING CODE

- Conceptualize what you want to do first
  - Sketch out plan and pseudo code, especially for figures and tables
- Understand what you can get out of your data and any limitations you may face when using it in R and any R package limitations
  - Some very niche or highly complex combination of analyses may be lacking from existing R packages)
- There are many ways to accomplish the same task and approach writing a program, do what makes most sense to you with however intermediate steps
- Use informative but concise variable naming conventions and formats,
  - use _ in names, upper and lower cases

CONSIDERATIONS WHEN WRITING CODE

- Use common programming format standards and guidelines to make code consistent, readable, and maintainable
  - Comment, comment, comment code
  - Use indentations and line breaks for readable
  - Use informative but concise variable naming conventions and formats,
    - use _ in names (e.g. `var_yr2010`), upper and lower cases (e.g. `raceEthn`)
  - Try to avoid "hard-coding" values, may cause errors later
    - For example, rather than calculating the mean of a variable as 2.3 and setting `x = 2.3`. Instead, define `x = mean(VARIABLE)` so that if `VARIABLE` changes at all the mean will update in the code accordingly

CONSIDERATIONS WHEN WRITING CODE

- Code in R can be split across multiple lines – must be split in such a way that the code would continue on and not just end, lines should not start with operators.
  For example:

  Would just end at line 1 and throw error at line 2

  (line 1) X = 1 + 2 + 3

  (line 2) + 4

  Would evaluate full summation

  (line 1) X = 1 + 2 +

  (line 2) 3 + 4

ADDITIONAL RANDOM R TIPS

- R is case sensitive
- Use <- or = to assign or create variables in R
- Vectors are created using `c()`, must all be same data type, for example: `c("one", "two", "three","four")` or `c(1,2,3,4,5)` or `c(x,y)`
- Index variables within a data table using dollar signs `DATASET$VAR1` or brackets, `DATASET[, "VAR1"]`

ADDITIONAL RANDOM R TIPS

- Check data types, and know how to do type conversions – lots of errors or problems arise because of incompatible or incorrect data types, e.g. categorical variables in a model as numeric
- Characters can be referenced with single or double quotes – but if you have quotes within quotes, the outer quotes should differ from the inner quotes, ex. "County's population", or 'The "substantiated" cases'
- Many coding techniques can be combined into one line (e.g. simultaneously using logical statements, subsetting syntax, assigning new values)

MANIPULATING DATA IN R

- Joining data
  - `merge(DATA_A, DATA_B, by = "shared_variable")`
  - `cbind(DATA_A,DATA_B)`
  - `rbind(DATA_A,DATA_B)`
- Subsetting/filtering data
  - `subset(DATA, var1 == CONDITION & var2 < 100 )`
  - `sample(DATA)`
- Mutating or creating variables, for example
  - `DATA$var1_rate1k = DATA$var1 / 1000`
  - `DATA$sex = ifelse(DATA$sex == 1, "Male", "Female")`

STRING DATA IN R

- Taking substring
  - `substr()`
- String length
  - `nchar()`
- Replace string
  - `str_replace()`
- Make upper or lower case
  - `str_to_upper()`, `str_to_lower()`
- Sort
  - `sort()`
- Look for character or substring
  - `grep()`, `grepl()`
- Join strings
  - `paste()`
- Split
  - `strsplit()`

NEXT SESSION…

October 18th, 2024 at
11am ET

Topic: "Tidyverse" Functions