Data Analysis in R

REU Data Carpentry Workshop

June 27th-28th

Elizabeth McDaniel, University of Wisconsin-Madison

After this lesson, you will be able to:

- Use RStudio for writing R scripts
- Implement the basic building blocks of an R script
- Load your data into R
- Manipulate data within R

Getting Started

- Understand the difference between an Rscript and RStudio
- Identify the different panels of an Rstudio window
- Use Rstudio to find help on R functions
- Describe how to troubleshoot problems with the wider R community in different settings

What is R?

- “R” refers to both the programming language itself, and the software platform to interpret scripts written in it

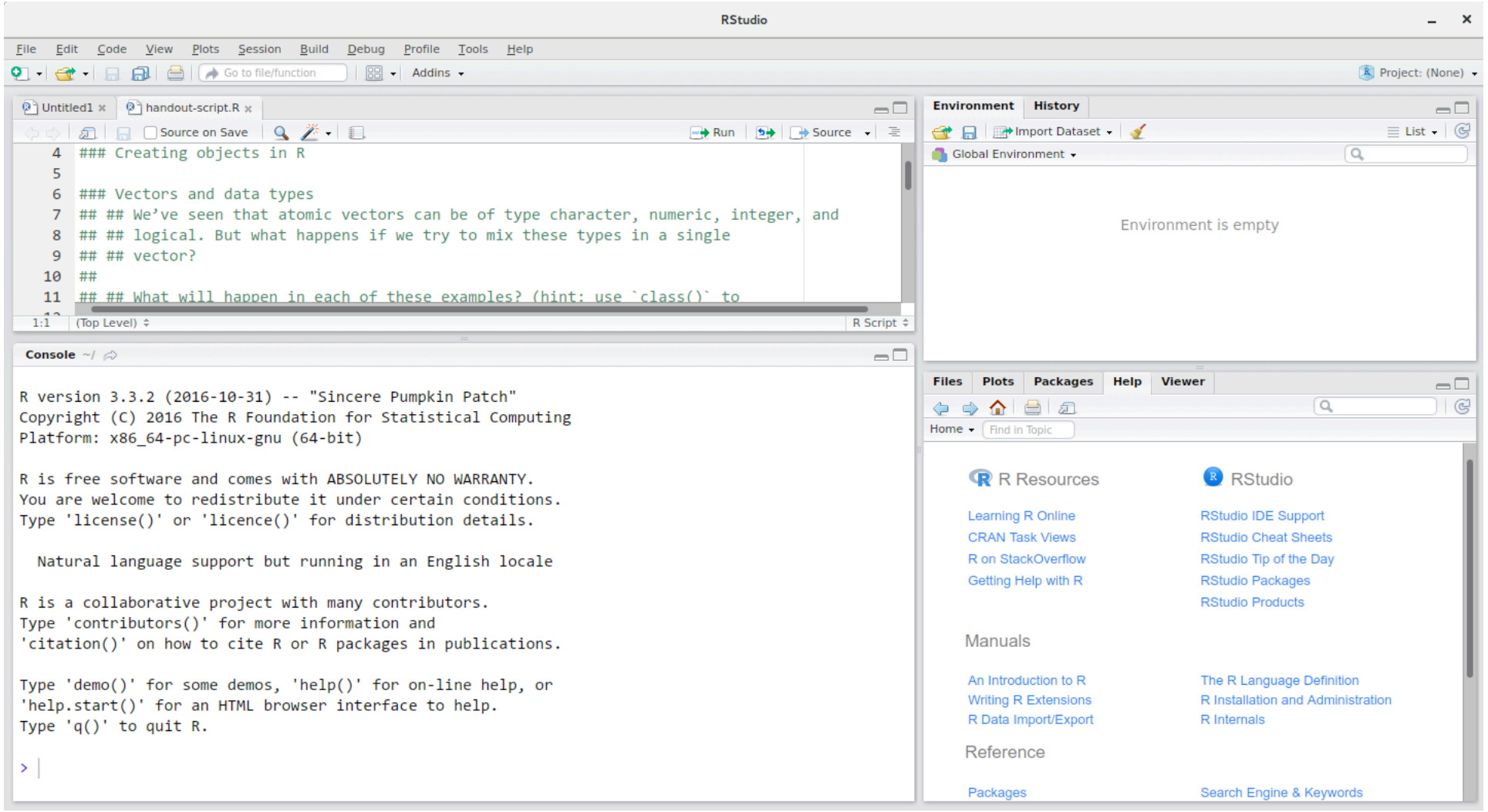
Why learn R?

- Less pointing and clicking involved
- Writing scripts makes analyses more reproducible than clicking a series of buttons
- Interdisciplinary (field-specific extensions or packages) for genomics, statistical simulations, etc.
- Takes data of all shapes and sizes
- Produce high-quality graphics without $$$
- Large, welcoming, open source community always developing new packages

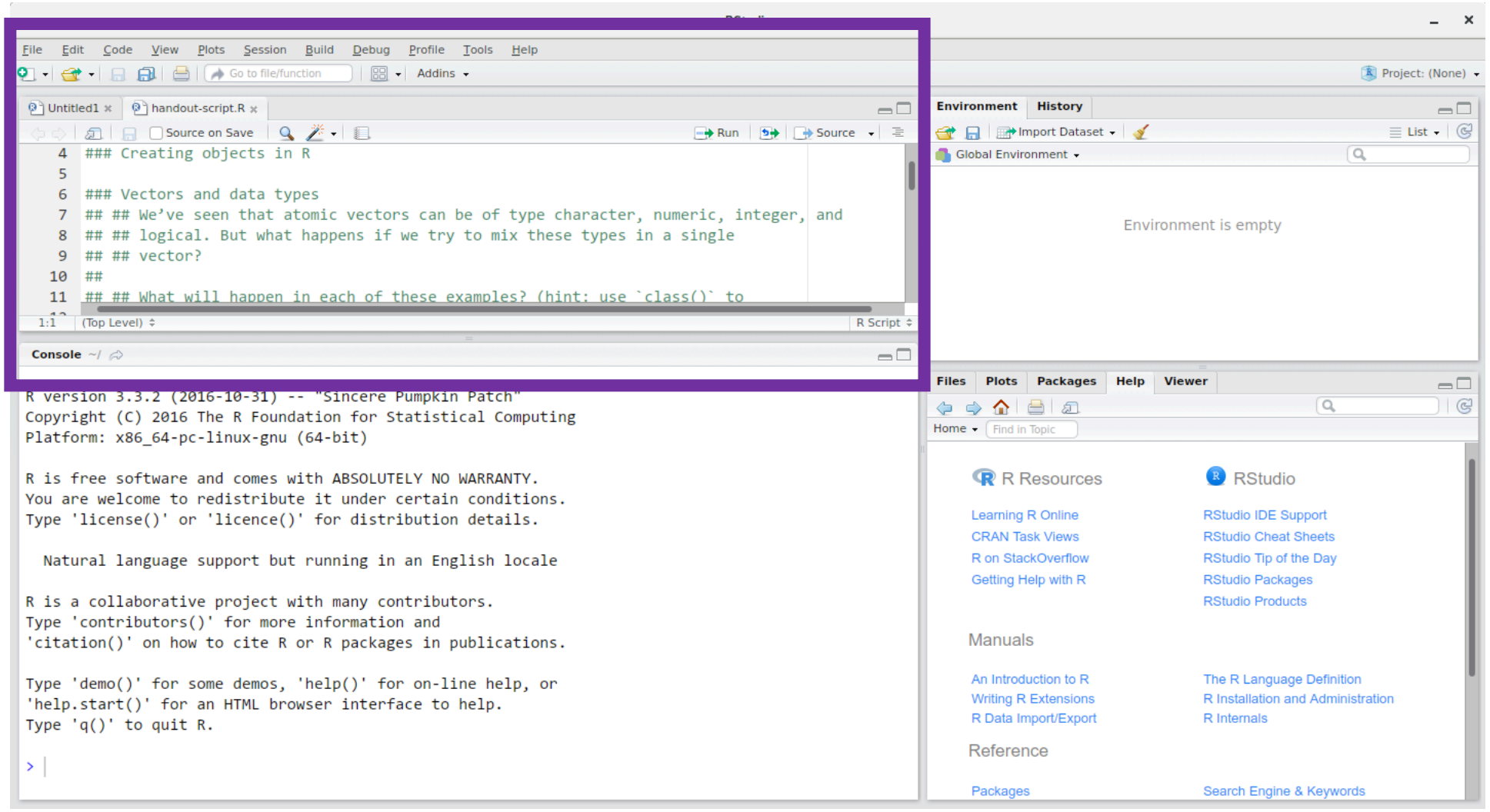
What is RStudio?

- Rstudio is an **I**ntegrated **D**evelopment **E**nvironment (IDE) for working with R
- Can write code, navigate files, view variables you created, and inspect plots within RStudio
- Lots of other IDEs that can suit a wide variety of programming languages

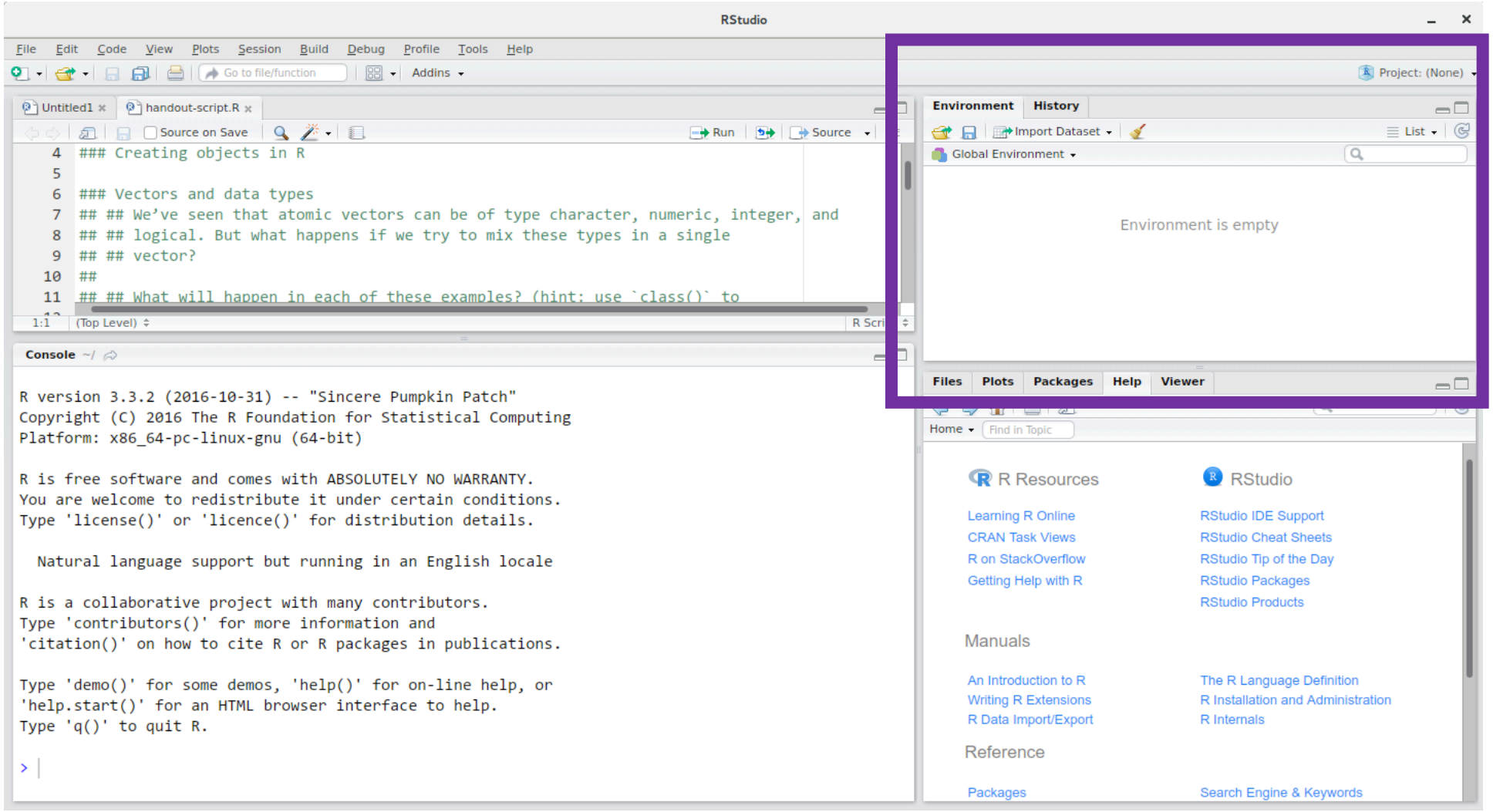
The RStudio Interface



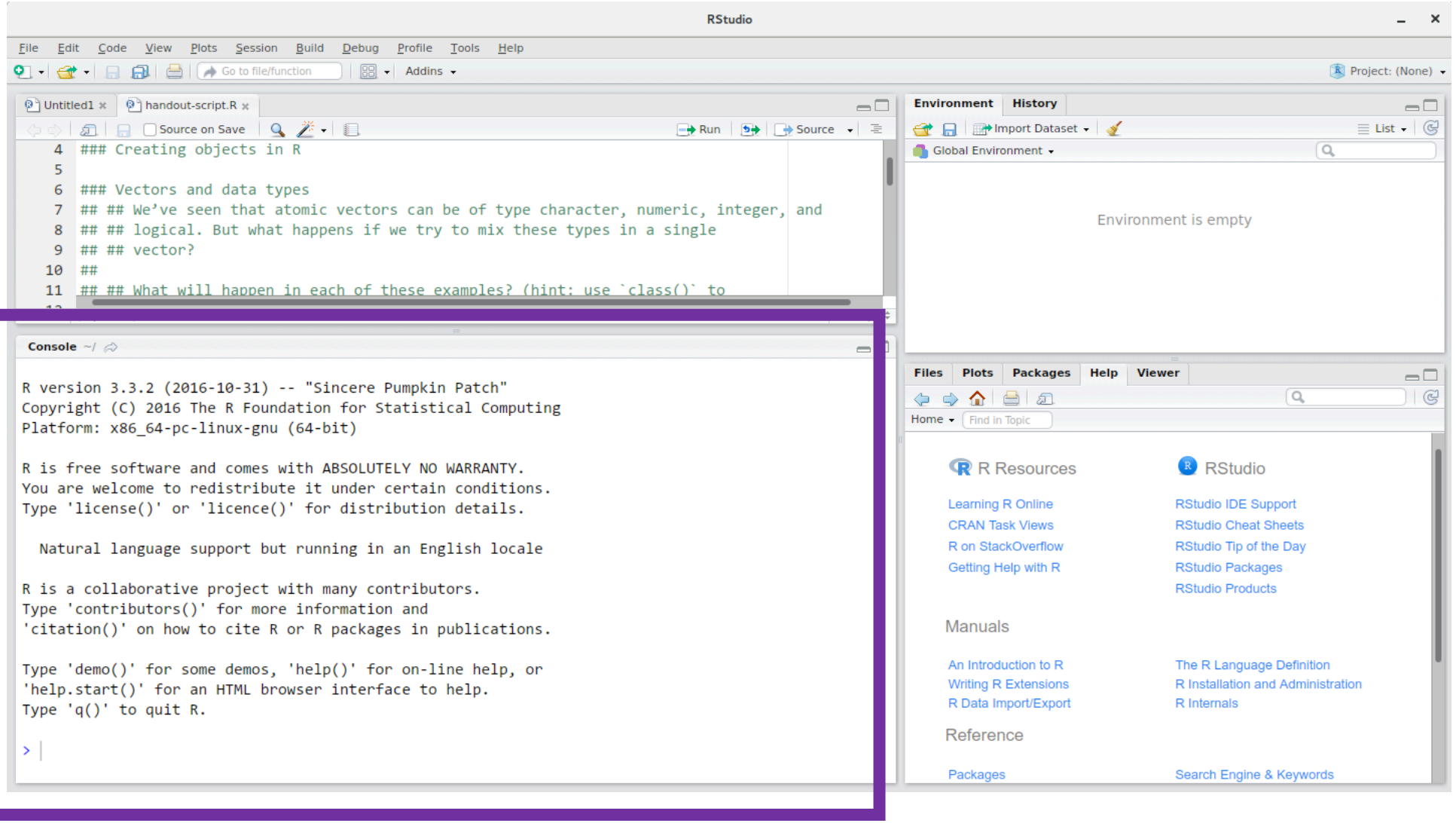
The RStudio Interface



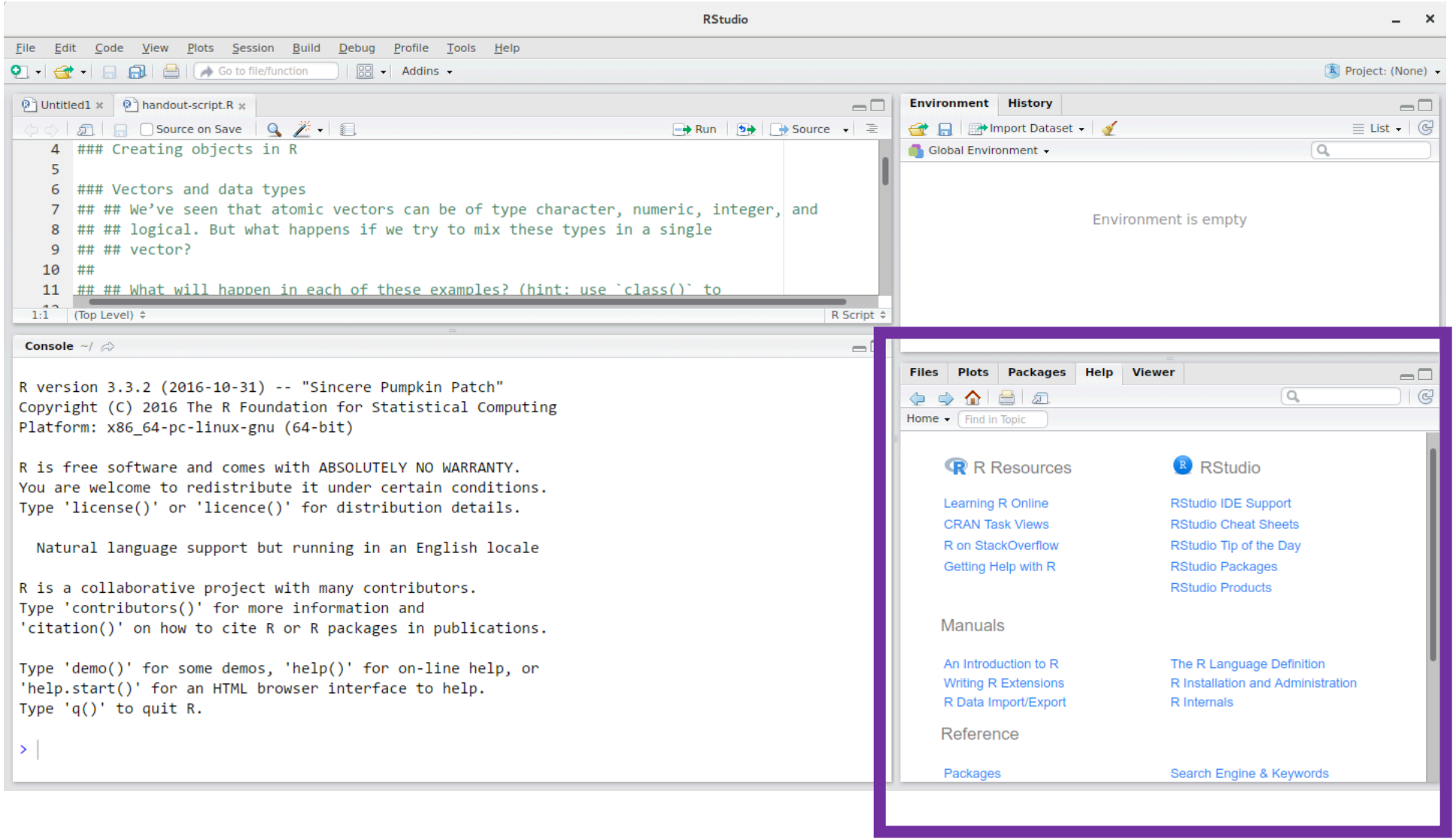
The RStudio Interface



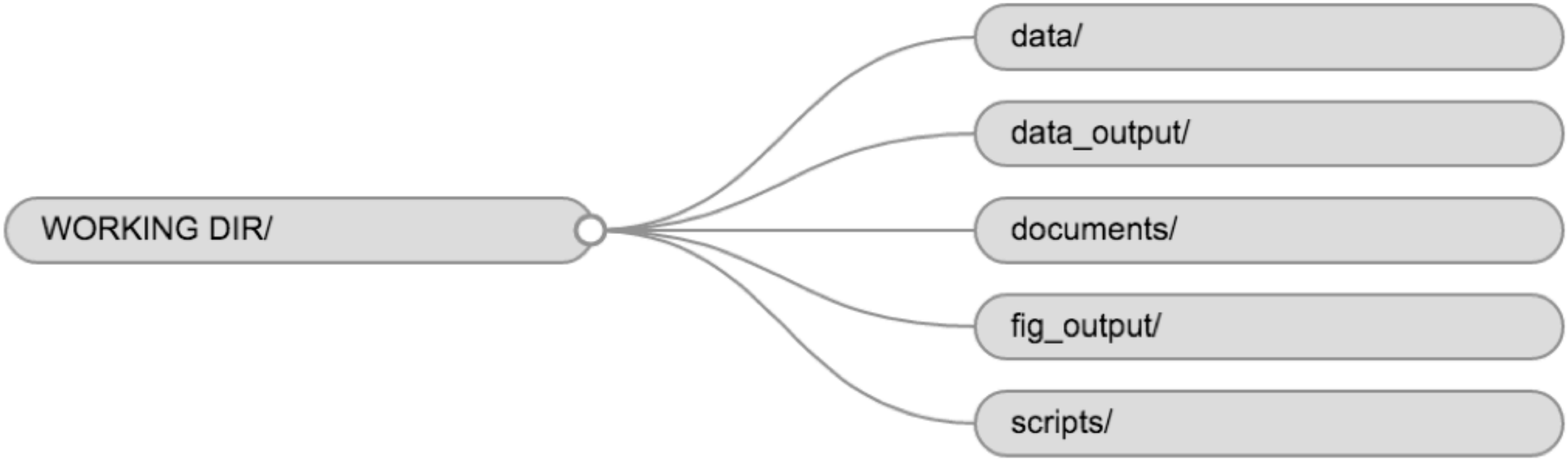
The RStudio Interface



The RStudio Interface



Your Working Directory



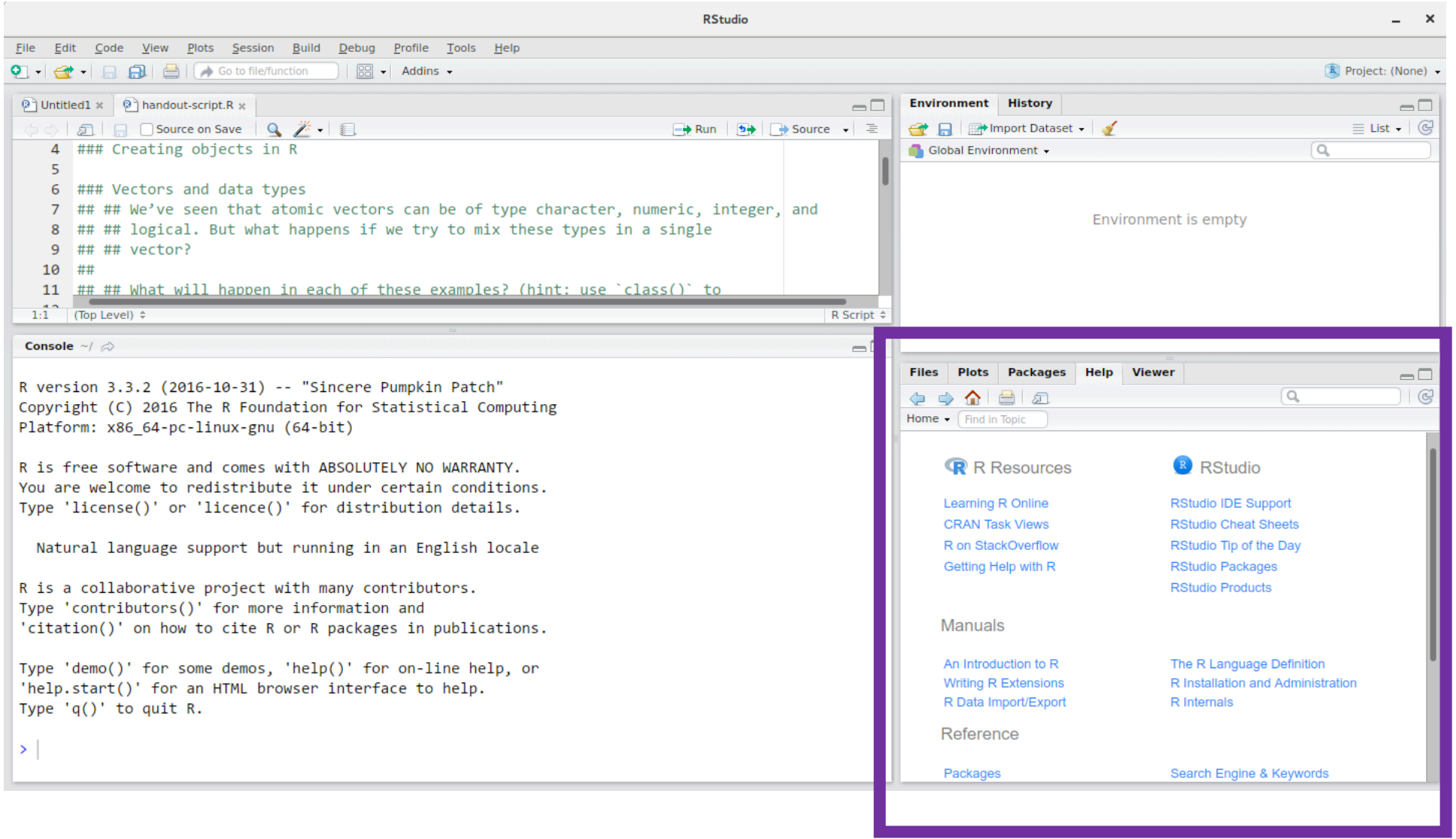
Working in RStudio

1. Start RStudio
2. File > New Project > New Directory
3. Enter a name for the new folder, which will be your **working directory**
4. Click Create Project

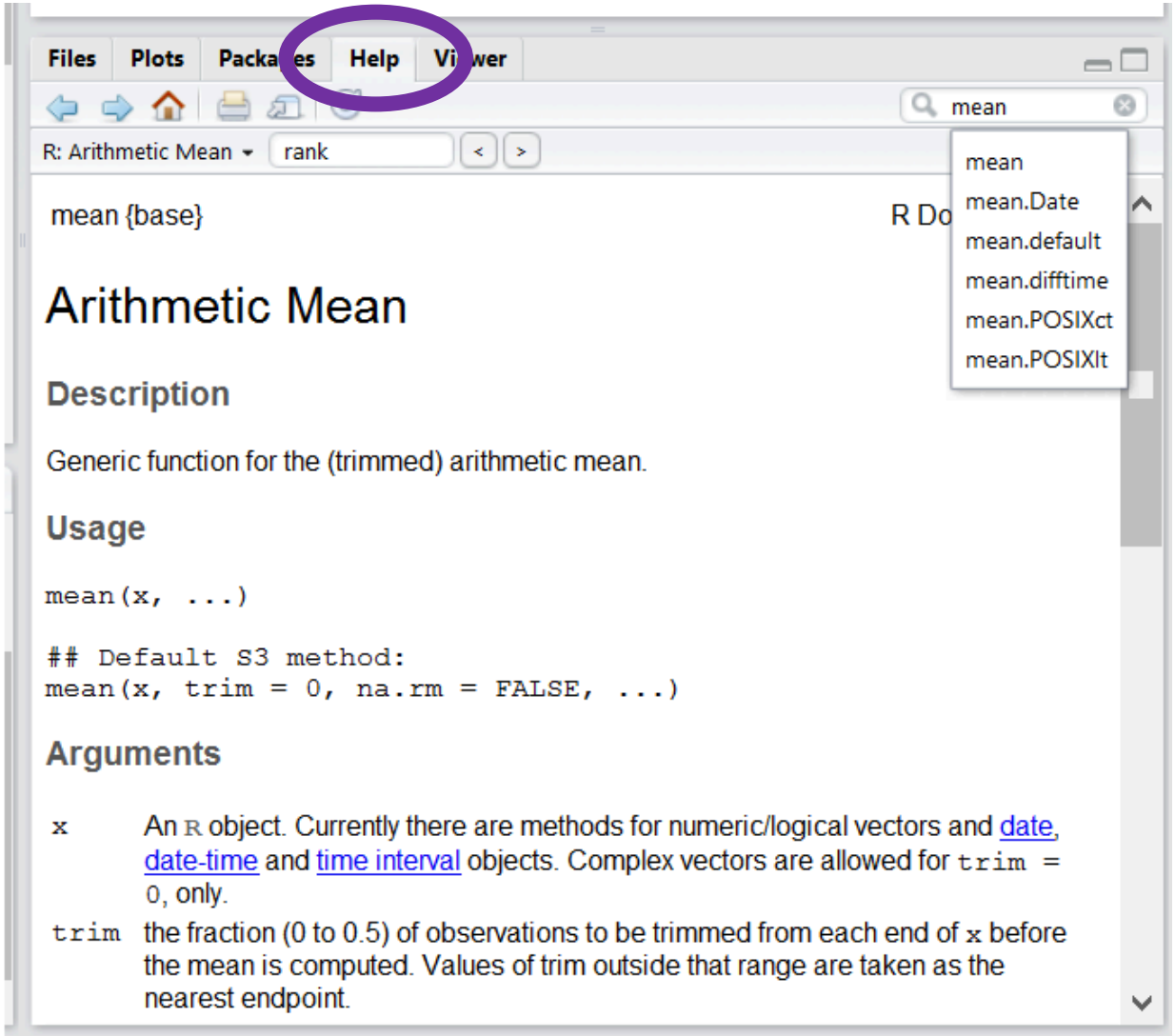
How do I find help?

Built-In RStudio help interface

The RStudio Interface

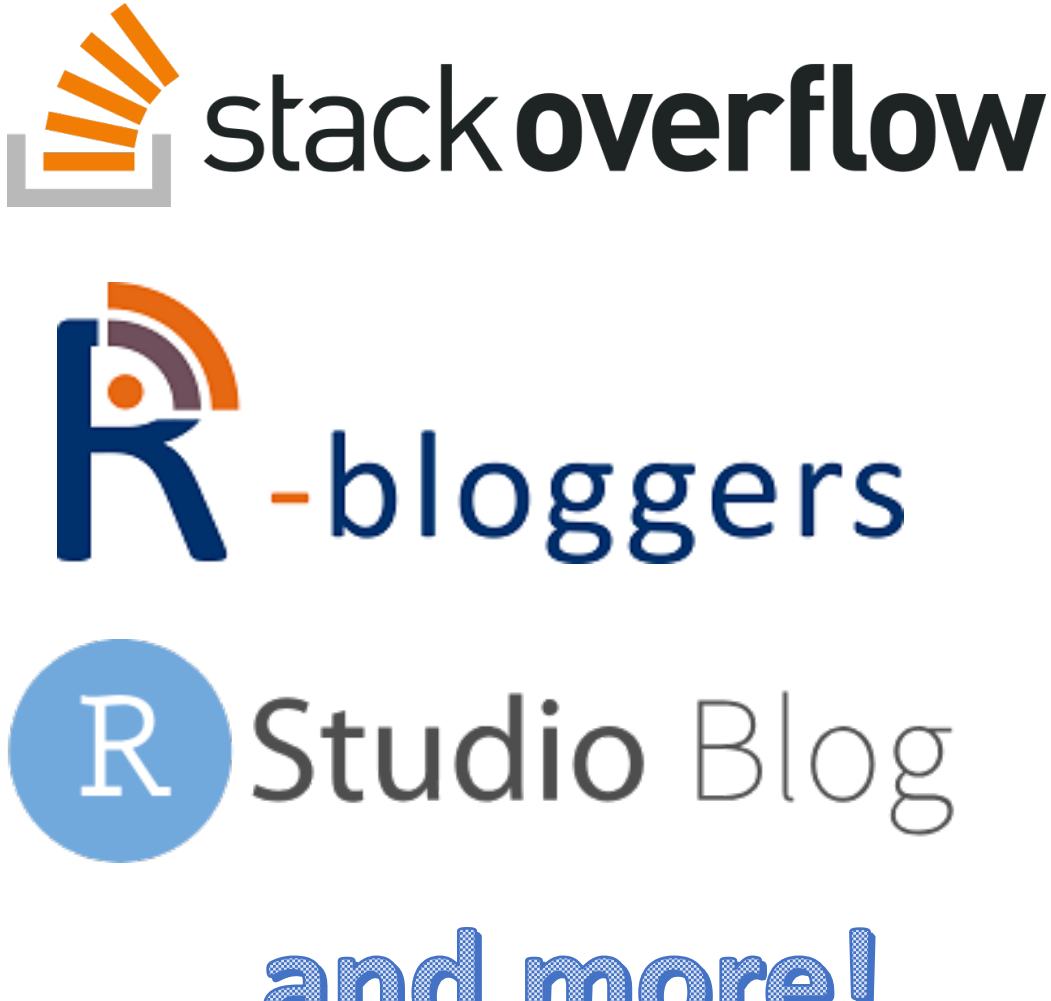


The RStudio Interface



How do I find help?

Google is a great place to start!



What information do I include when I ask for help?

- Make your code human-readable
- Include the output of your session info, which includes information about your platform, versions of R and packages you are using

✓ Getting Started

- ✓Understand the difference between an Rscript and RStudio
- ✓Identify the different panels of an Rstudio window
- ✓Use Rstudio to find help on R functions
- ✓Describe how to troubleshoot problems with the wider R community

Introduction to R

- Terminology:
  - object
  - assign
  - call
  - function
  - arguments
  - options

Introduction to R

- Assign values and names to objects in R
- Use comments to give future you/collaborators helpful information
- Solve arithmetic operations in R
- Call functions and use arguments to change default parameters
- Inspect vectors and manipulate their content
- Subset and extract values from vectors
- Analyze vectors with missing data

Objects

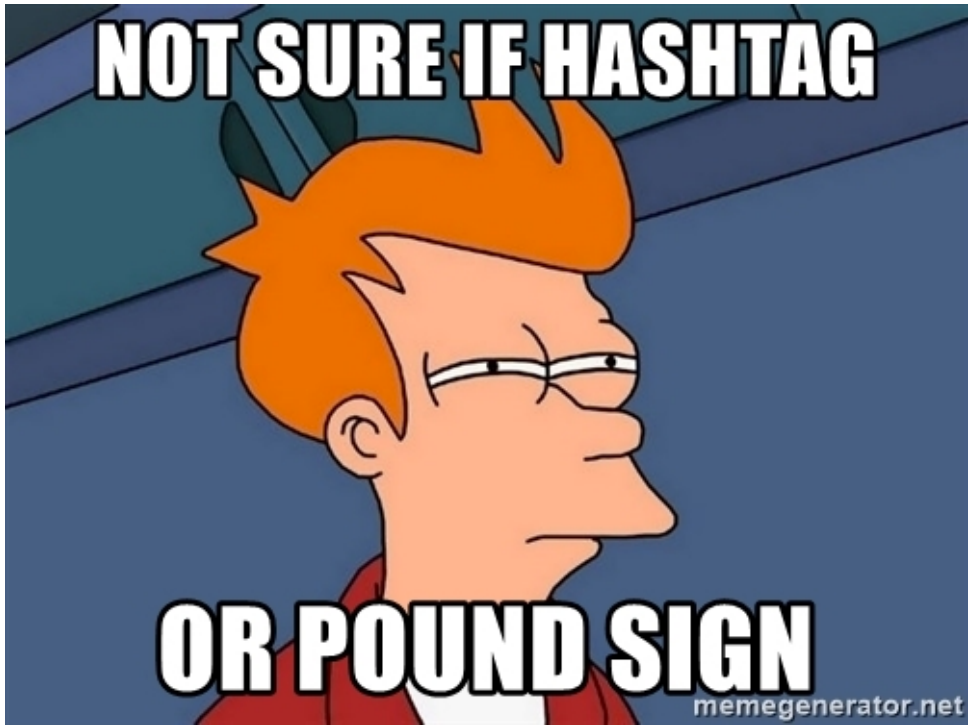
- Assign a value:
  - Give a name to the object
  - Use the <- assignment operator
- Can be any name (with no spaces!), but cannot start with a number. Names are case sensitive. Cannot use a fundamental function (if, else, for) as an object name

Comments

“Your closest collaborator is you six months ago,
but you don’t reply to emails.
- Mark Holder”
- Karl Broman

Comments

- Leave yourself useful notes about why you wrote lines of code a certain way (you won't remember…trust me)
- The comment character in R (and most programming languages) is the # symbol



Challenge #1

Challenge

What are the values after each statement in the following?

```
mass <- 47.5            # mass?
age  <- 122             # age?
mass <- mass * 2.0      # mass?
age  <- age - 20        # age?
mass_index <- mass/age  # mass_index?
```

Vector

- Most common data type in R
- Composed of a series of values, either numbers or characters
- Assign a series of values to a vector using the c() function, which **c**ombines the arguments to form a vector

Subsetting Vectors

- Extract one or several specific values from a vector
- Provide an indices with brackets

Functions

- Pre-packaged or “canned” scripts with complex set of commands and arguments

Conditional Subsetting

- TRUE or FALSE logical factors
- Combining multiple tests
- & = both conditions are satisfied
- | = one OR the other condition is satisfied
- %in% test if any elements of a search vector are found

Missing Data

- R was designed to analyze datasets, which may sometimes have missing data
- Missing data in vectors is represented as NA

Challenge #2

Challenge

1. Using this vector of heights in inches, create a new vector, `heights_no_na`, with the NAs removed.

```
heights <- c(63, 69, 60, 65, NA, 68, 61, 70, 61, 59, 64, 69, 63, 63, NA, 72, 65
```

2. Use the function `median()` to calculate the median of the `heights` vector.
3. Use R to figure out how many people in the set are taller than 67 inches.

Introduction to R

- Terminology:
  - object
  - assign
  - call
  - function
  - arguments
  - options

✓ Introduction to R

- ✓Assign values and names to objects in R
- ✓Use comments to give future you/collaborators helpful information
- ✓Solve arithmetic operations in R
- ✓Call functions and use arguments to change default parameters
- ✓Inspect vectors and manipulate their content
- ✓Subset and extract values from vectors
- ✓Analyze vectors with missing data

Break

1 red sticky for something to improve
1 green sticky for something you liked during the session

Loading Data into R

- Terminology:
  - Data frame
  - Factor
  - String

Loading Data into R

- Load external data from a .csv file into a data frame
- Describe what a data frame is
- Summarize the contents of a data frame
- Use indexing to subset specific parts of a data frame
- Describe a factor
- Reorder and rename factors

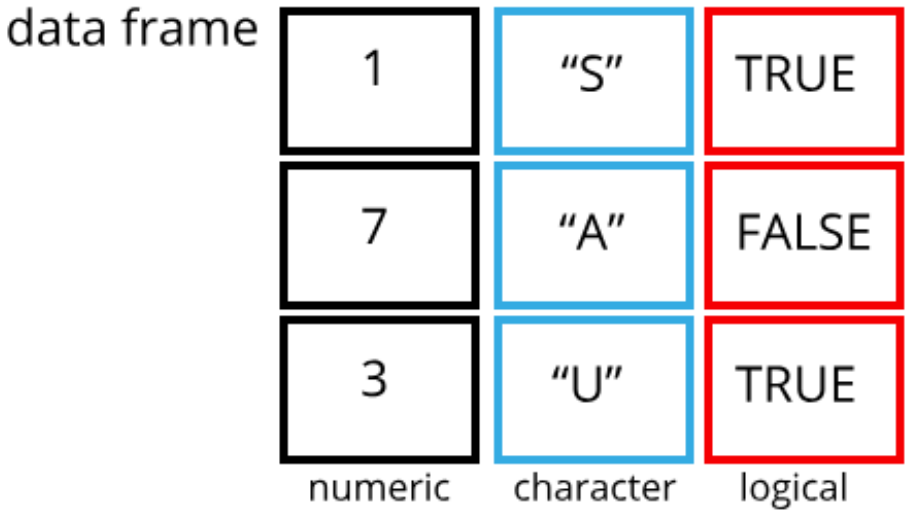
Survey Data

- Studying species repartition and weight of animals caught in plots in our study area

Column	Description
record_id	Unique id for the observation
month	month of observation
day	day of observation
year	year of observation
plot_id	ID of a particular plot
species_id	2-letter code
sex	sex of animal (“M”, “F”)
hindfoot_length	length of the hindfoot in mm
weight	weight of the animal in grams
genus	genus of animal
species	species of animal
taxon	e.g. Rodent, Reptile, Bird, Rabbit
plot_type	type of plot

Data Frames

- Data structures for tabular data, further used for statistics and plotting
- Representation of data in the format of a table, columns are vectors with the same length, and contains the same type of data



Challenge #3

Challenge

Based on the output of `str(surveys)`, can you answer the following questions?

- What is the class of the object `surveys`?
- How many rows and how many columns are in this object?
- How many species have been recorded during these surveys?

Indexing and Subsetting Data Frames

- Survey data frame contains rows and columns = 2 dimensions
- Extract information from the data frame, need to specify “coordinates”
- Row numbers come first, then the column number

Challenge #4

Challenge

1. Create a `data.frame` (`surveys_200`) containing only the data in row 200 of the `surveys` dataset.
2. Notice how `nrow()` gave you the number of rows in a `data.frame`?
    - Use that number to pull out just that last row in the data frame.
    - Compare that with what you see as the last row using `tail()` to make sure it's meeting expectations.
    - Pull out that last row using `nrow()` instead of the row number.
    - Create a new data frame (`surveys_last`) from that last row.
3. Use `nrow()` to extract the row that is in the middle of the data frame. Store the content of this row in an object named `surveys_middle`.
4. Combine `nrow()` with the `-` notation above to reproduce the behavior of `head(surveys)`, keeping just the first through 6th rows of the surveys dataset.

Factors

- Factors represent categorical data
- Several of the columns of our surveys data frame contain integers
- However, columns such as genus, species, sex, plot_type… are categorical data
- Factors are stored as integers with labels, and can either be ordered or unordered
- Contain pre-defined set of values = levels

Loading Data into R

- Terminology:
  - Data frame
  - Factor
  - String

✓ Loading Data into R

- ✓Load external data from a .csv file into a data frame
- ✓Describe what a data frame is
- ✓Summarize the contents of a data frame
- ✓Use indexing to subset specific parts of a data frame
- ✓Describe a factor
- ✓Reorder and rename factors

Manipulating and Analyzing Data

- Terminology:
  - pipe operator
  - split-apply-combine concept

Manipulating and Analyzing Data

- Describe the purpose of the dplyr and tidyr packages
- Select columns and filter rows
- Use the pipe %>% operator
- Understand and use the split-apply-combine concept for data analysis
- Apply summary statistics and combine results
- Describe the concept of wide and long table formats, and reshape data to and from these formats
- Export data to a .csv file

Packages in R

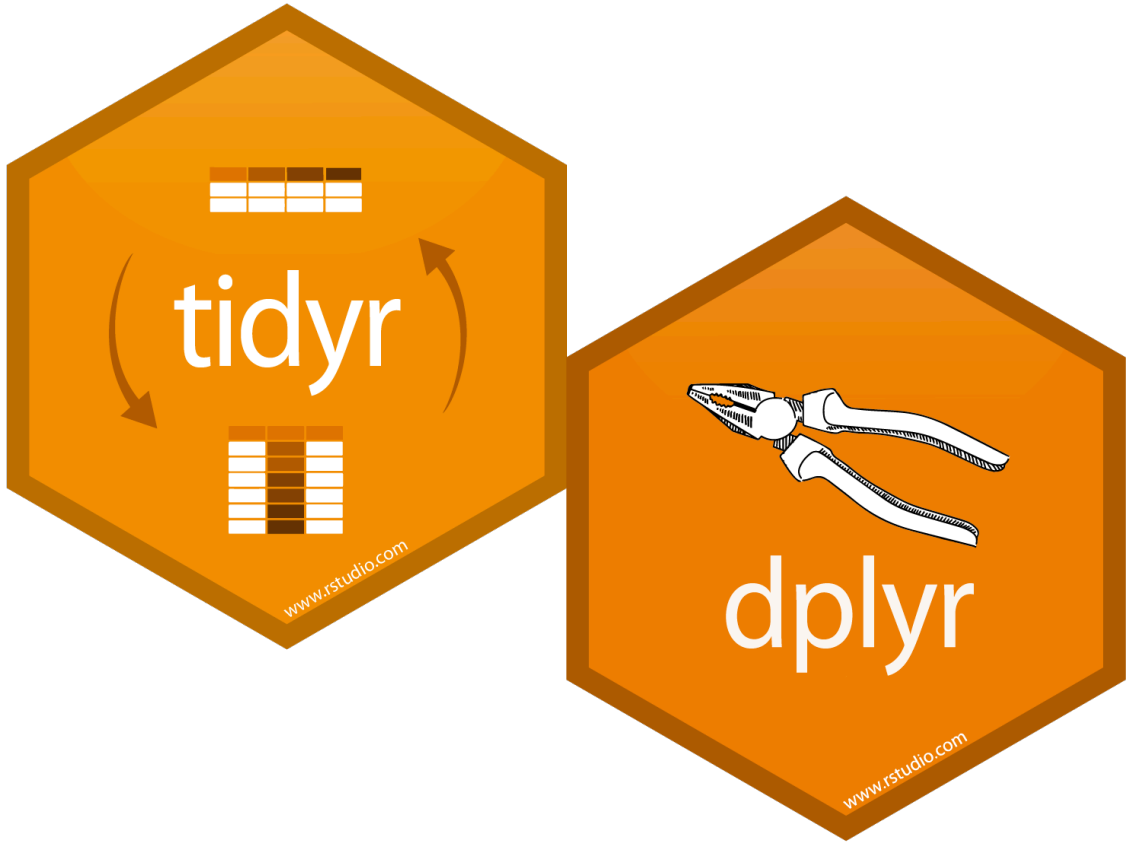
- Packages are extensions of R with additional functions that do not come in base R

tidyverse

- The tidyverse is an umbrella of a suite of packages: tidyr, dplyr, ggplot2, and others.
- Addresses 3 common issues when doing data analysis with functions in base R:
  1. Results from a base R function depend on type of data
  2. R expression used in a non standard way, can be confusing for new learners
  3. Hidden arguments, contains default operations that new learners don't have to be aware of

dplyr and tidyr Packages

- dplyr is a package for manipulating tabular data easily
- tidyr enables conversion between different data formats for plotting and analysis purposes



Tibbles

- When a dataset is read into R using read_csv from the tidyverse package readr, the class of the data is a tbl_df or a “tibble”
- This data structure is very similar to a data frame, with the differences are:
  1. Displays data type of each column under its name, but only prints the first few rows of data and as many columns as will fit on the screen
  2. Columns of class character are never converted into factors

Select Columns and Filter Rows

- Selecting columns of a data frame
- Filter rows by specific criteria

Multiple Operations

- What if you want to select columns and filter rows at the same time?
  - You could do intermediate steps
  - Nested functions

Pipes

- Pass information from one program to another, useful for doing many operations on the same dataset
- Pipes in R are represented by %>%, which the shortcut is Cmd+shift+M or Ctrl+Shift+M

Challenge #5

Challenge

Using pipes, subset the `surveys` data to include animals collected before 1995 and retain only the columns `year`, `sex`, and `weight`.

Mutate

- Create new columns based on values in existing columns

Challenge #6

Challenge

Create a new data frame from the `surveys` data that meets the following criteria: contains only the `species_id` column and a new column called `hindfoot_half` containing values that are half the `hindfoot_length` values. In this `hindfoot_half` column, there are no `NA`s and all values are less than 30.

Hint: think about how the commands should be ordered to produce this data frame!

Split-apply-combine Approach

- Many data analysis tasks can be tackled with the split-apply-combine approach
- Split the data into groups
- Apply some analysis to each group
- Combine the results
- Exemplified through the dplyr group_by() function

Summarize

- Group_by is often used with summarize to collapse each group into a single-row summary of that group
- Takes as arguments as the categorical variables that you want to calculate summary statistics

Challenge #7

Challenge

1. How many animals were caught in each `plot_type` surveyed?
2. Use `group_by()` and `summarize()` to find the mean, min, and max hindfoot length for each species (using `species_id`). Also add the number of observations (hint: see `?n`).
3. What was the heaviest animal measured in each year? Return the columns `year`, `genus`, `species_id`, and `weight`.

Reshaping data

- From the Excel lesson we learned that data should be structured so that:
    1. Each variable has its own column
    2. Each observation has its own row
    3. Each value has its own cell
    4. Each type of observational unit forms a table

Gather

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

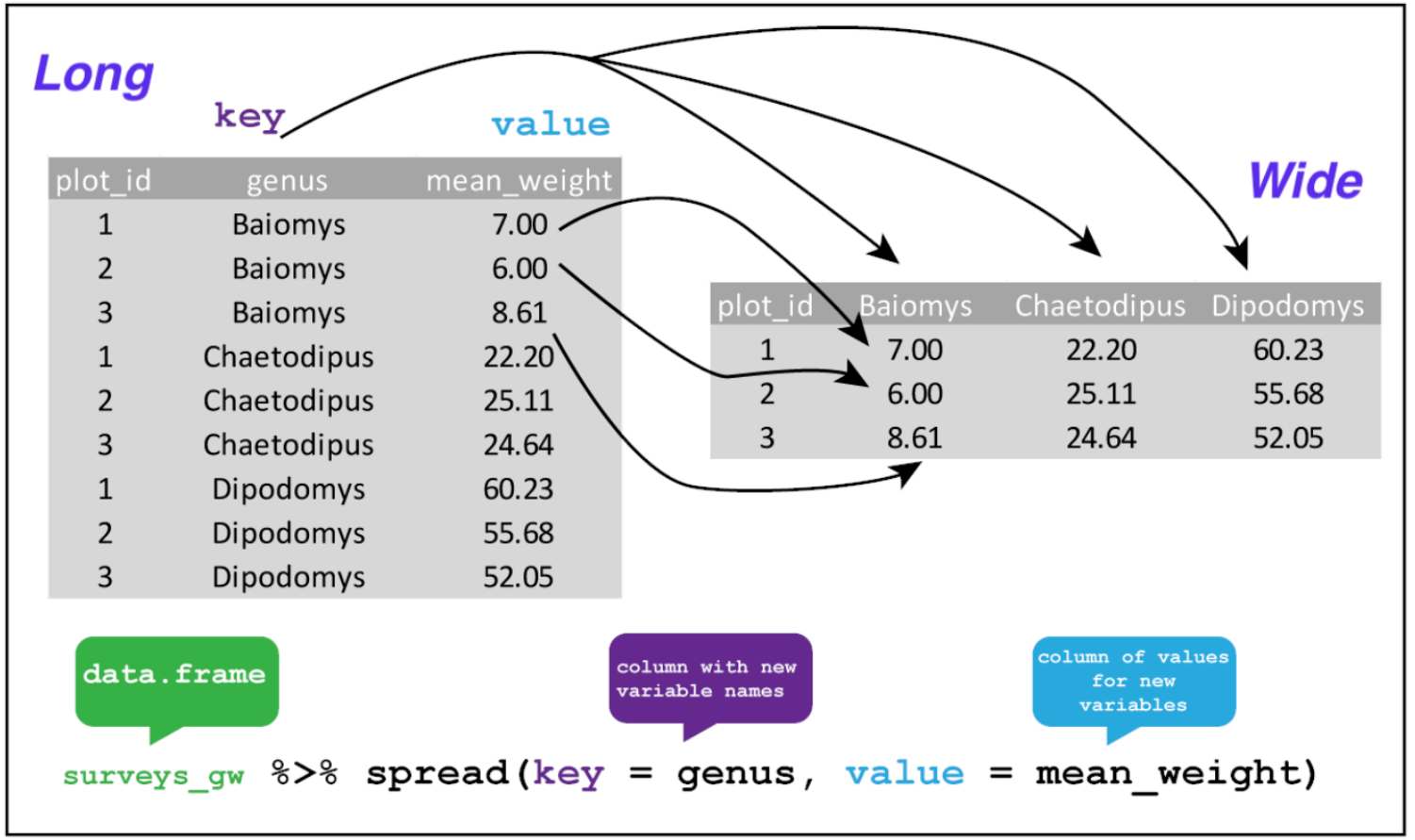
country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

Spread

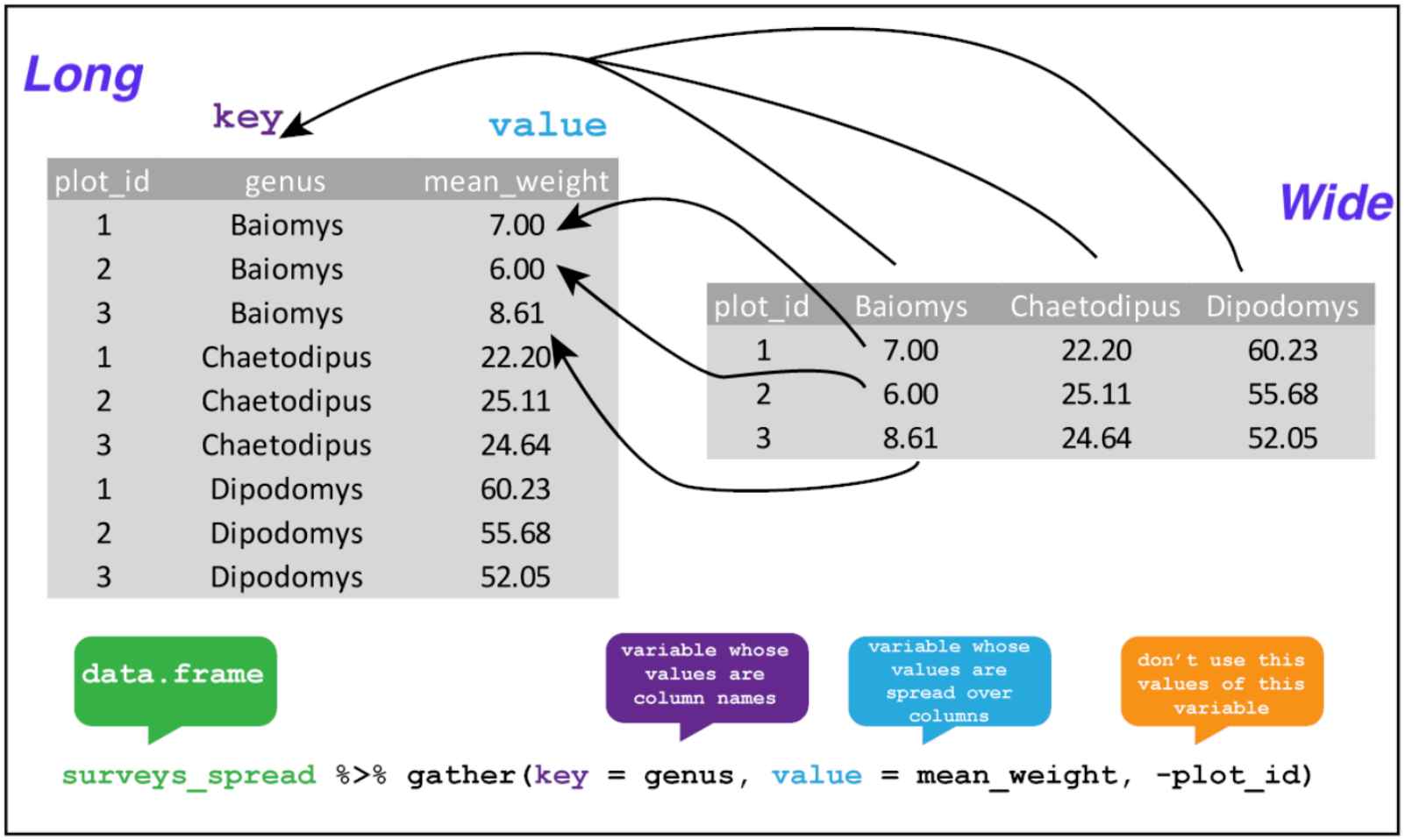
country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Spreading our data



Gathering our data



Challenge #8

Challenge

1. Spread the `surveys` data frame with `year` as columns, `plot_id` as rows, and the number of genera per plot as the values. You will need to summarize before reshaping, and use the function `n_distinct()` to get the number of unique genera within a particular chunk of data. It's a powerful
2. Now take that data frame and `gather()` it again, so each row is a unique `plot_id` by `year` combination.
3. The `surveys` data set has two measurement columns: `hindfoot_length` and `weight`. This makes it difficult to do things like look at the relationship between mean values of each measurement per year in different plot types. Let's walk through a common solution for this type of problem. First, use `gather()` to create a dataset where we have a key column called `measurement` and a `value` column that takes on the value of either `hindfoot_length` or `weight`. *Hint*: You'll need to specify which columns are being gathered.
4. With this new data set, calculate the average of each `measurement` in each `year` for each different `plot_type`. Then `spread()` them into a data set with a column for `hindfoot_length` and `weight`. *Hint*: You only need to specify the key and value columns for `spread()`.

Exporting data

- Generate CSV files from dataframes you created in R

Manipulating and Analyzing Data

- ✓Describe the purpose of the dplyr and tidyr packages
- ✓Select columns and filter rows
- ✓Use the pipe %>% operator
- ✓Understand and use the split-apply-combine concept for data analysis
- ✓Apply summary statistics and combine results
- ✓Describe the concept of wide and long table formats, and reshape data to and from these formats
- ✓Export data to a .csv file