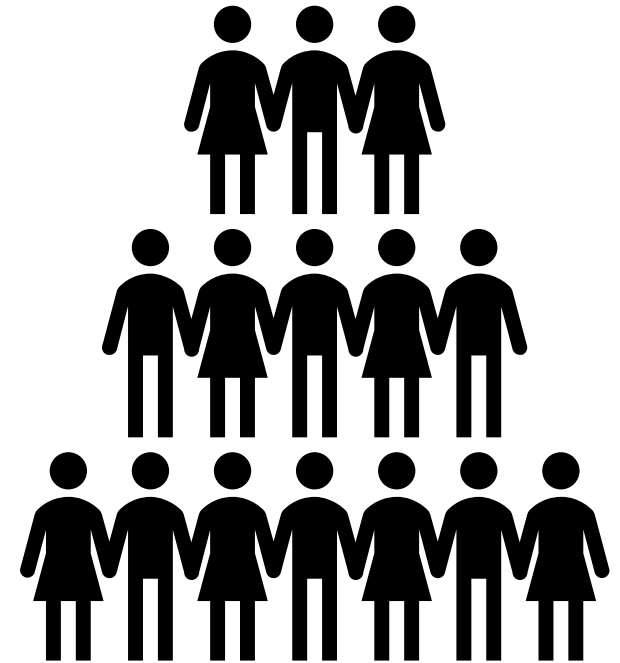


CogLab: Visualize Data

WEEK 7 / R101

logistics: project milestone #4

- **full experiment code** is due Oct 21 (Mon)
- incorporate feedback from design draft (milestone #3)
- include a demographics survey
 - after experiment, before thank you
 - think about how to record the data



cycling through multiple conditions

- where do the multiple conditions come in – which plugin trial is going to change based on the condition?

```
var priming_proc = {
  timeline: [fixation, image, prime, target, priming_feedback],
  timeline_variables: test_stimuli,
  randomize_order: true,
  repetitions: 2
};

var practice_proc = {
  timeline: [fixation, image, prime, target, priming_feedback],
  timeline_variables: practice_stimuli,
  randomize_order: true
};
```

cycling through multiple conditions

- before priming_proc, create a new CONDITION variable that randomly selects between 0 and 1
- print out to console verify

```
var CONDITION = jsPsych.randomization.sampleWithoutReplacement([...Array(2).keys()], 1)[0];  
console.log("CONDITION= " + CONDITION);  
  
var priming_proc = {
```

```
CONDITION = 0
```

cycling through multiple conditions

- create a new variable `experiment_stimuli` that will bring different lists in, depending on the value of `CONDITION`

```
var CONDITION = jsPsych.randomization.sampleWithoutReplacement([...Array(2).keys()], 1)[0];  
console.log("CONDITION= " + CONDITION);  
var experiment_stimuli = get_stimuli(CONDITION)
```

cycling through multiple conditions

- now we need to define the `get_stimuli` function in our code
- try this in your project code and make changes where needed!

```
var CONDITION = jsPsych.randomization.sampleWithoutReplacement([...Array(2).keys()], 1)[0];  
  
console.log("CONDITION= " + CONDITION);  
  
var experiment_stimuli = get_stimuli(CONDITION)
```

```
var thank_you = {  
  type: jsPsychHtmlKeyboardResponse,  
  stimulus: 'Thank you! You can press any key to end the experiment.',  
  data: {  
    typeoftrial: 'thank_you'  
  },  
}  
  
function get_stimuli(CONDITION){  
  if(CONDITION == 0){return test_stimuli_1}  
  else{return test_stimuli_2}  
}
```

today's agenda

- R 101
- programming basics
- visualizing data

intuitions about data

- review Savic et al.'s **results** section
- what is the **key research question**?
- what **kinds of data** will answer this research question?
- **which trials** do we need to analyze?

preliminary analyses

- how do we calculate performance on attention check questions?

Preliminary Analyses: Attention to Sentences and Pseudoword Forms

To assess whether participants attended to the Training sentences and learned the pseudoword forms, we analyzed participants' responses on the attention check questions and the free association task.

Performance on attention check questions was high ($M = .94$, $SD = .08$), which confirmed that participants read the sentences. Performance of two participants was below .75 accuracy, so their data were excluded from the further analyses.

priming

- which trials were **analyzed?**
- which trials were **excluded?**

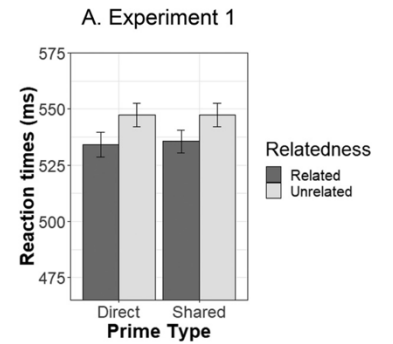
semantic priming task. Specifically, we tested whether participants more rapidly identified a familiar noun (Target: *apple, horse*) when it was preceded by a novel pseudoword (Prime) in the Related (Direct and Shared) versus the Unrelated (Direct and Shared) condition. Following the logic of extensive semantic priming research (e.g., [McRae & Boisvert, 1998](#)), if participants linked pseudowords with familiar words based on direct and shared co-occurrence, pseudowords should prime the familiar words from the same triad. Specifically, novel pseudowords should allow participants to respond more quickly to Targets from the same triad (Related condition) than to Targets from the opposite triad (Unrelated condition). Prior to analyzing reaction times, we removed data from both incorrect trials, and trials with extremely short (< 200 ms) and extremely long response latencies ($> 1,500$ ms). This resulted in a removal of 5.6% of all trials. Summary statistics are reported in [Table 2](#).

priming: model

- what were the independent variables?
- what was the dependent variable?
- what kind of statistical test was employed?

We analyzed reaction times by fitting them to linear mixed effects models with fixed effects of Prime Type (levels: Direct and Shared), Relatedness (levels: Related and Unrelated), and their interaction. The random-effects structure was based on the log likelihood ratio test (Wagenmakers & Farrell, 2004). Specifically, following Wagenmakers and Farrell (2004), we compared models with the same fixed-effects structure but varying complexity in their random-effects structure, and settled on the simplest among the candidate models that provided the best fit to the data. The best fitting random effects structure, as indicated by log-likelihood ratio test, included only a random intercept for participant and random intercept for stimuli (i.e., Triad).⁴ This model revealed no significant effect of Prime Type, neither as a main effect nor in interaction with Relatedness ($F_s < 1.0$, $p_s > .10$). Critically, the model revealed a significant effect of Relatedness, $F(1, 2443.4) = 5.85$, $p = .016$, with participants responding faster in Related than in Unrelated conditions (Figure 4A). In other words, participants responded faster to familiar words (Targets) when they were preceded by novel pseudowords with which they directly co-occurred or shared co-occurrence in training (Related Prime), than when they were preceded by novel pseudowords that directly co-occurred or shared co-occurrence with a different familiar word (Unrelated Prime).

analysis preview



phase	measure	type	exclusion criteria
attention	accuracy	descriptive	< 0.75
priming	$RT_{related}$ vs. $RT_{unrelated}$ for direct and shared pairs	inferential (mixed effects model / ANOVA)	$RT < 200$ ms and $RT > 1500$ ms correct responses related/unrelated and direct/shared trials

R 101

- R is a statistical programming language
- **Rstudio** is a graphical user interface for R that makes R a little less scary!
- we will use R within Rstudio to analyze data



installation check-in

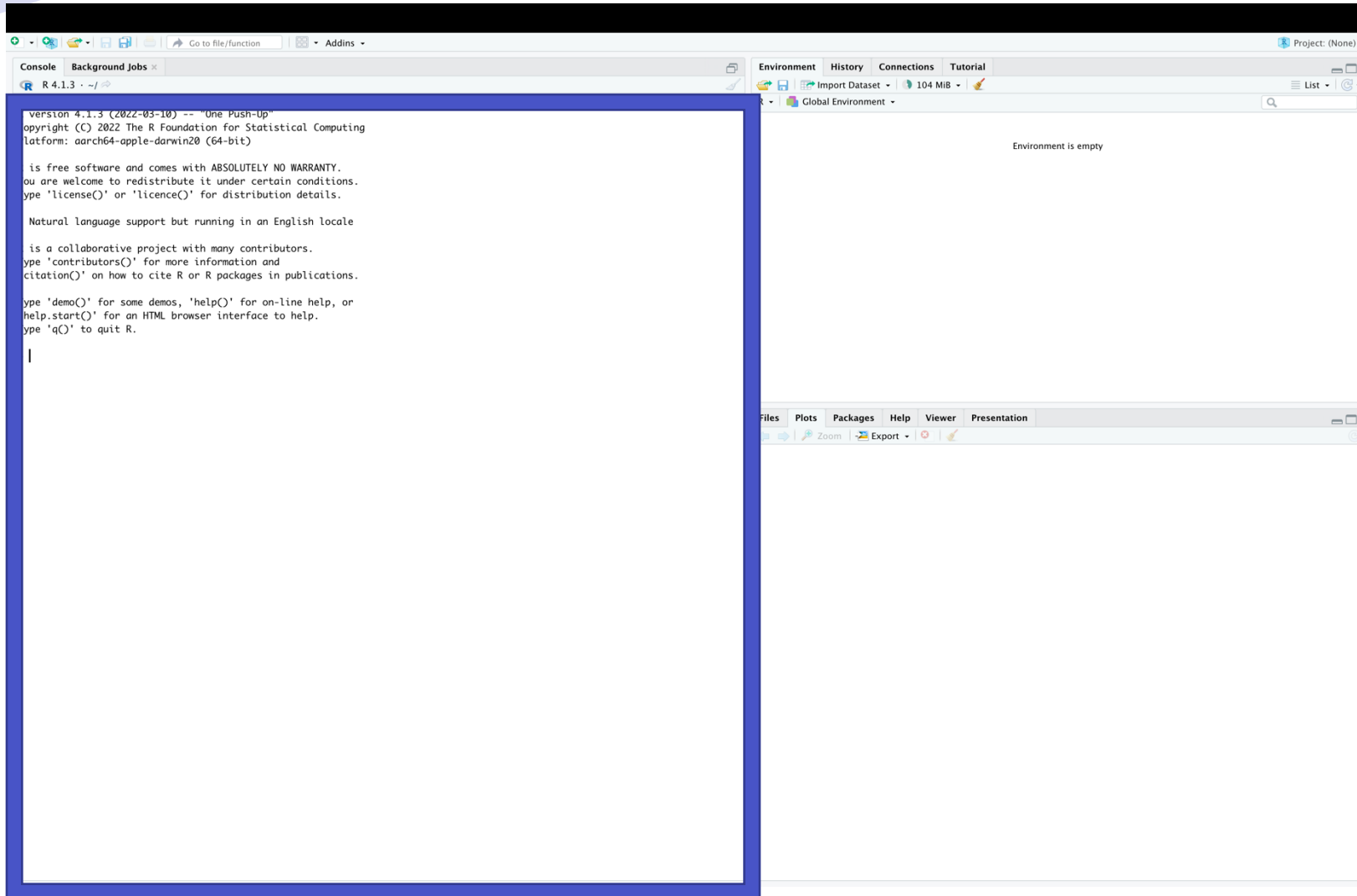
Rstudio: basics

The screenshot displays the RStudio interface with the following components:

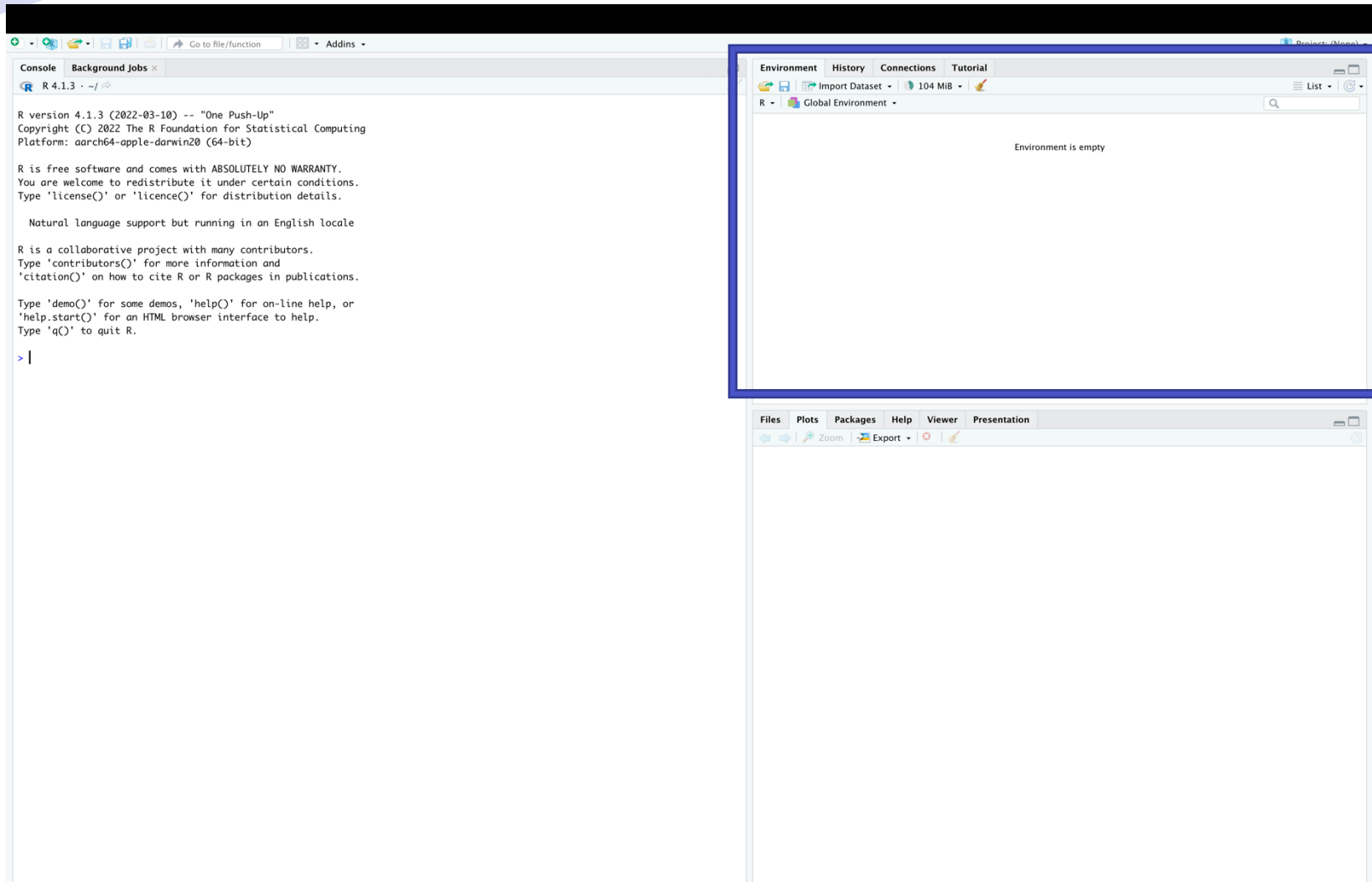
- Console:** Shows the R version 4.1.3 (2022-03-10) -- "One Push-Up" and platform information (aarch64-apple-darwin20 (64-bit)). It includes a welcome message and instructions on how to use R, such as typing 'license()' for distribution details and 'demo()' for on-line help. The prompt is currently at '> |'.
- Environment:** Shows the 'Global Environment' and indicates that the environment is empty.
- Files, Plots, Packages, Help, Viewer, Presentation:** These panes are currently empty.

Rstudio: basics

This is called
the “console”.
This is where
any code you
write is run by
R

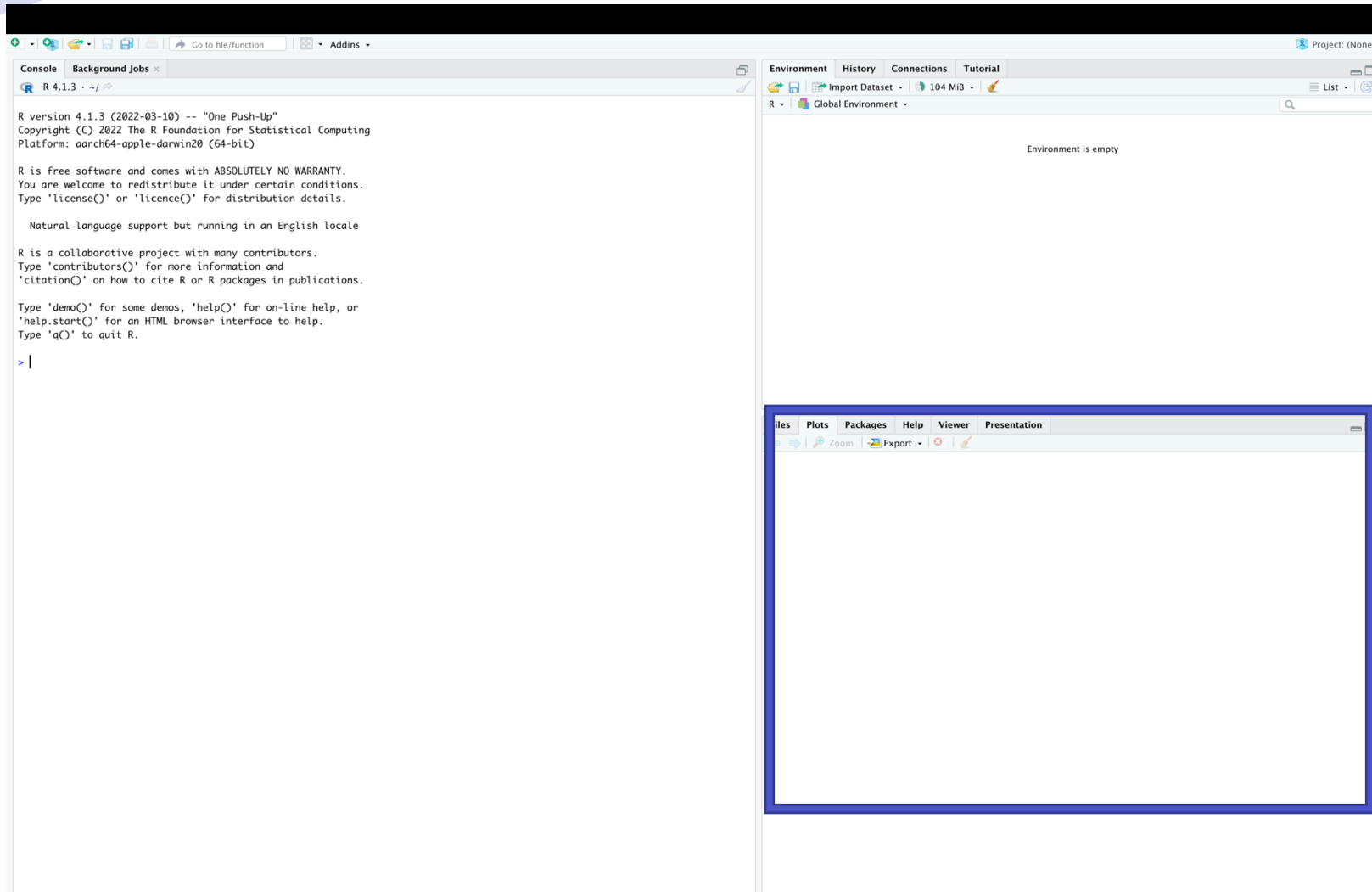


Rstudio: basics



This is called the “environment”, this is where any data you import will show up. This is also where any variables you define or create will show up

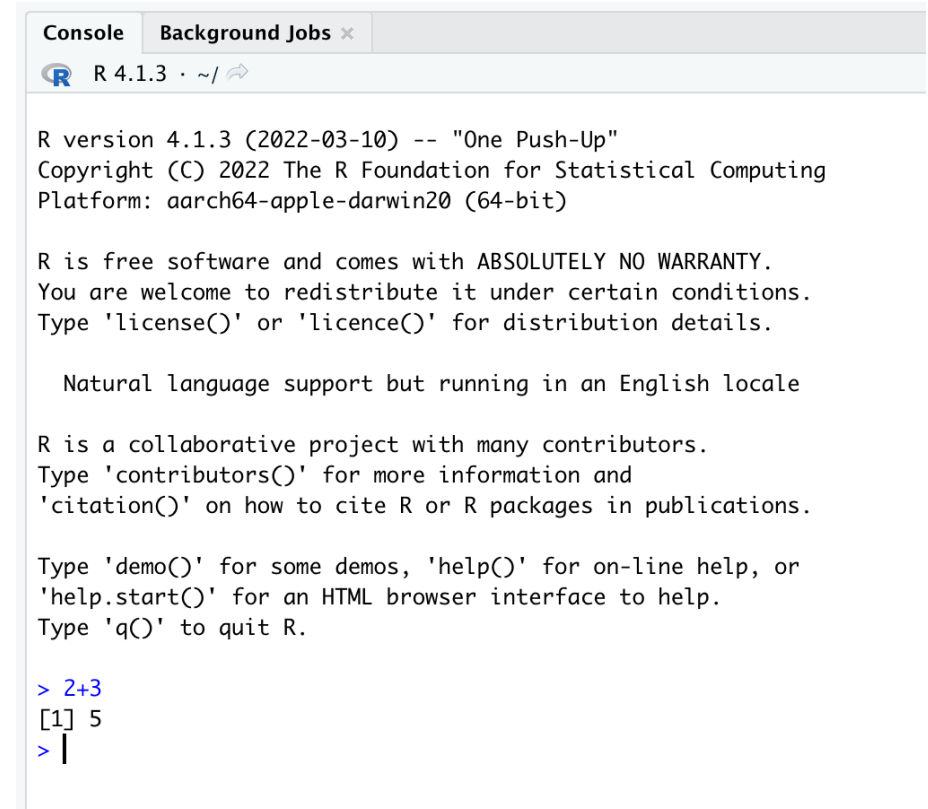
Rstudio: basics



This is called the files and plots pane. Currently it is on “plots”, so any plots you make in the console in R will show up here as a preview

testing the R console

- you can type any code in the console and it will try to understand and run it
- try typing `2 + 3` where you see the blinking cursor and press `return`
- voila! R just added those up for you!



```
Console Background Jobs x
R 4.1.3 · ~/ ↗

R version 4.1.3 (2022-03-10) -- "One Push-Up"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

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.

> 2+3
[1] 5
> |
```

programming basics

- functions
- arguments
- objects
- vectors
- types
- lists
- packages

functions

- functions do things in R
- compute
 - the square root of 64
 - log of 1 with base 10
 - log of 1 with default base
 - sum of 1, 2, and 3
- how do we get more info about a function (e.g., sum)?

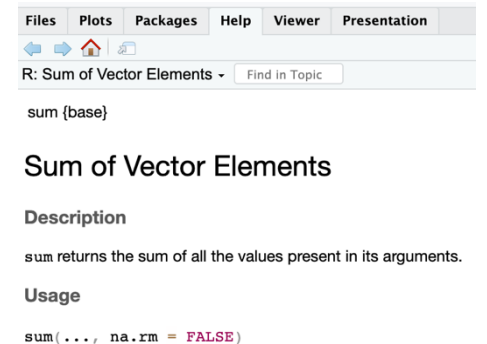
```
> sqrt(64)
[1] 8
```

```
> log(1, base = 10)
[1] 0
```

```
> log(1)
[1] 0
```

```
> sum(1,2,3)
[1] 6
```

```
- -
> ?sum
|
```



The screenshot shows the R help window for the `sum` function. The title bar includes 'Files', 'Plots', 'Packages', 'Help', 'Viewer', and 'Presentation'. The main content area displays the following information:

- sum (base)
- Sum of Vector Elements**
- Description**
sum returns the sum of all the values present in its arguments.
- Usage**
sum(..., na.rm = FALSE)

arguments

- arguments are inputs to functions
 - args (function-name)
- find out the arguments for:
 - factorial
 - round
- round 15.789 to 1 digit
- compute 10 random values from a normal distribution with mean 15 and sd 5

```
> args(factorial)
function (x)
NULL
```

```
> args(round)
function (x, digits = 0)
NULL
```

```
> round(15.789, 1)
[1] 15.8
```

```
> rnorm(10, mean = 15, sd = 5)
[1] 13.556844  9.609776 15.397840 14.853597 23.752432
[6] 20.804199  7.595791  6.920951 18.733704  9.639373
```

objects

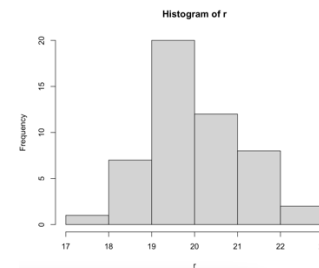
- objects store information
- store:
 - “I love pancakes” in an object called `mystring`
 - sum of 1,2, and 3 in `x`
 - 50 random values from a normal distribution with mean 20 and sd 1 in an object called `r`
- plot the histogram of `r`
- what is the mean of `r`?
- what is the standard deviation of `r`?

```
> mystring = "I love pancakes"
> |
```

```
> x = sum(1,2,3)
> x
[1] 6
> |
```

```
> r = rnorm(50, mean = 20, sd = 1)
```

```
> hist(r)
```



Environment	History	Connections	Tutorial
📁 📄 📊 Import Dataset	🌐 106 MiB	🧹	
R	Global Environment		
Values			
mystring	"i love pancakes"		

Environment	History	Connections	T
📁 📄 📊 Import	🌐 110 MiB	🧹	
R	Global Environment		🔍
Values			
mystring	"I love pancakes"		
x	6		

Environment	History	Connections	Tutorial
📁 📄 📊 Import Dataset	🌐 151 MiB	🧹	
R	Global Environment		
Values			
mystring	"I love pancakes"		
r	num [1:50] 19.9 19.9		
x	6		

vectors

- **vectors** are 1-d arrays of values
 - $x = c(1,2,3)$
- create a vector of integers from 51 to 55
- R counts from 1 (unlike JavaScript!)
- retrieve the **fourth value** from this vector

```
> vec = c(51,52,53,54,55)
> vec
[1] 51 52 53 54 55
```

```
> vec = c(51:55)
> vec
[1] 51 52 53 54 55
```

```
> vec[4]
[1] 54
```


data types

- R has 6 basic data types: integer, double, character (string), logical, complex, raw
- vectors can only contain values of the same data type
- try storing a character string and a number together in a vector
- R will automatically convert them to the same datatype (character is default)

```
> v = c("ABC", 34)
> v
[1] "ABC" "34"
```

```
> v = c(23.45, 34)
> v
[1] 23.45 34.00
```

lists

- lists store multiple data types
- create a list called `myclass` of `names` and `ages`
- retrieve the `ages` from `myclass`
- compute the mean of the `ages`

```
> myclass = list(names = c("Dyana", "Gia", "Stephen"), ages = c(87, 43, 12))
> myclass
$names
[1] "Dyana" "Gia" "Stephen"

$ages
[1] 87 43 12

> myclass$ages
[1] 87 43 12

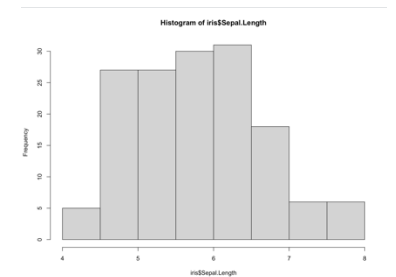
> mean(myclass$ages)
[1] 47.33333
> x = myclass$ages
> mean(x)
[1] 47.33333
```

dataframes

- **tabular data** is stored as a dataframe in R
- dataframes are simply a **list of vectors**
- `View(iris)`
- create a histogram of sepal lengths from iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

```
> View(iris)
> hist(iris$Sepal.Length)
```

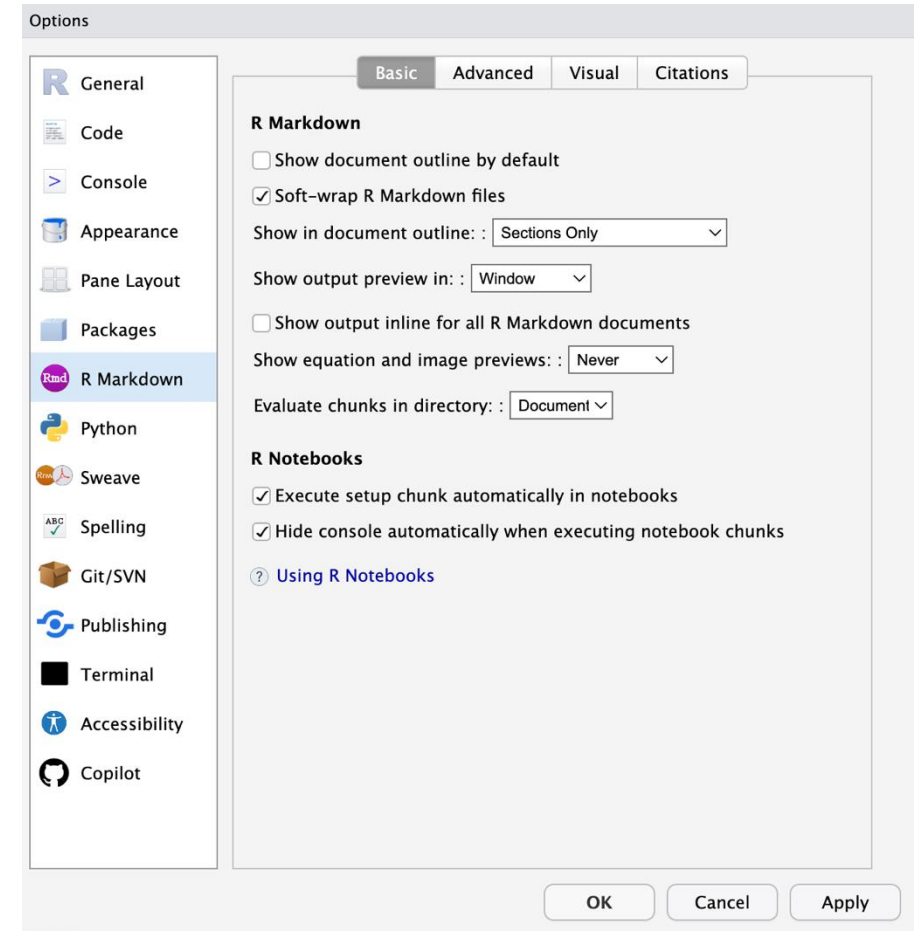
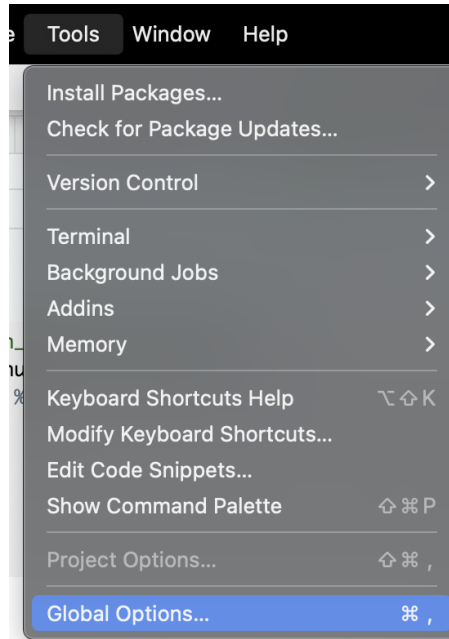


storing code

- anything you type in the console will go away when you close R
- **R projects and notebooks** are a good way to keep track of your code and maintain a reproducible workflow

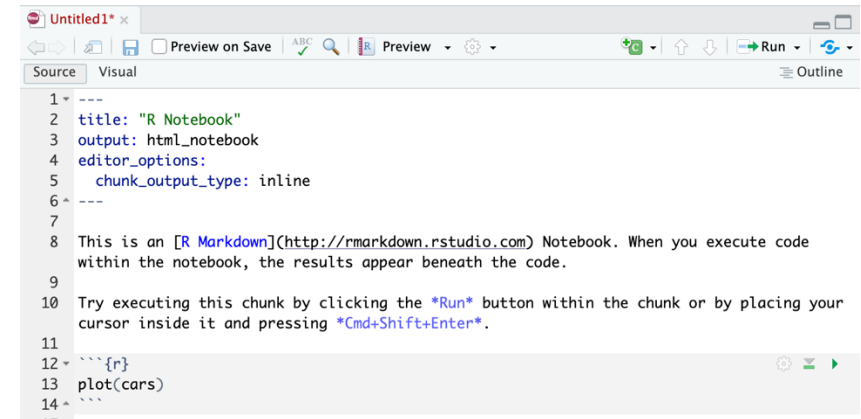
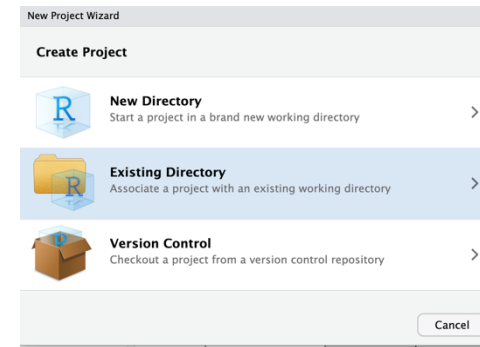
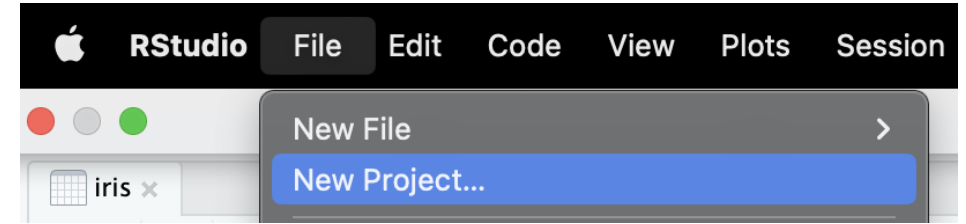
setting global options

- go to tools > global options
- uncheck “show output inline”
- apply

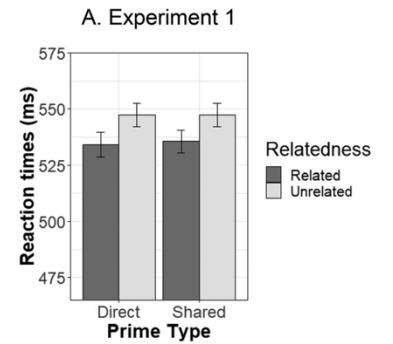


creating a project in R

- File > New Project > existing directory
 - select the directory of first-jspych-experiment
- File > New file > R Markdown
 - opens a notebook in **Markdown** format
- you can add “chunks” of code
 - between ```{r}` and ````
- save (Command +S) this file as “first-R-notebook” with the .Rmd extension
- we will write all our analysis code in this file!



analysis preview

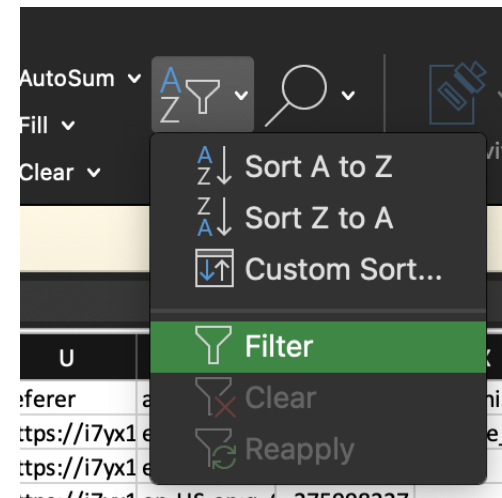


phase	measure	type	exclusion criteria
attention	accuracy	descriptive	< 0.75
priming	$RT_{related}$ vs. $RT_{unrelated}$ for direct and shared pairs	inferential (mixed effects model / ANOVA)	$RT < 200$ ms and $RT > 1500$ ms correct responses related/unrelated and direct/shared trials

exploring our data

- download and save class_data.csv in first-jspsych-experiment folder
- open class_data in Excel
- apply a filter to the data

success	timeout	failed_image	failed_audio	failed_video	trial_type	trial_index	time_elapse
					instructions	0	22022
					html-keyboa	1	27396
					html-keyboa	2	34360
					html-keyboa	3	39447
					html-keyboa	4	43870
					html-keyboa	5	47892
					html-keyboa	6	50499
					html-keyboa	7	52899
					survey-text	8	60315
					html-keyboa	9	63420



filtering rows in excel

- locate the column that identifies each type of trial in our experiment
- filter to only attention trials
- locate the column that contains the accuracy

	Z	
	typeoftria	st
2021	instructions	
5371	sentence	S
6963	sentence	lr
5086	sentence	T

AE	AF	AG	
el2	novel3	correct	cue
p	NOT_FOUND		0
ziland	NOT_FOUND		1
_FOUNE	NOT_FOUND		0
c	NOT_FOUND		1
_FOUNE	NOT_FOUND		1
_FOUNE	NOT_FOUND		0
_FOUNE	NOT_FOUND		0
p	NOT_FOUND		1
_FOUNE	NOT_FOUND		1

typeoftrial

Sort

Ascending Descending

By color: None

Filter

By color: None

Equals attention

And Or

Choose One

Search

- (Select All)
- association
- association_instructions
- attention
- feedback
- fixation
- image

Auto Apply

Apply Filter Clear Filter

exploring individual responses

- find the **response** column
- click on the tiny arrow that shows all the unique responses participants provided
- should minor spelling mistakes be forgiven?

	Z	AA	AB
1	typeoftria	stimulus	response
3	attention		apple
3	attention		foobly
1	attention		horse
5	attention		dodish
2	attention		dodish
3	attention		geck
3	attention		zimziland
5	attention		foobly

The screenshot shows a data table with columns AB, AC, AD, AE, and AF. The 'response' column is selected, and a filter overlay is displayed. The overlay has a 'Sort' section with 'Ascending' and 'Descending' options, and a 'Filter' section with a 'By color' dropdown and a search box. The search box contains the text 'Search'. Below the search box, there is a list of unique responses with checkboxes: (Select All), →horse, →mipp, Any, apple, didsh, and dobish. The 'Auto Apply' checkbox is checked. At the bottom of the overlay, there are 'Apply Filter' and 'Clear Filter' buttons.

next class

- **before** class
 - *prep*: work on project milestone #5
- **during** class
 - reading in and plotting data