Day 1, Part 2: RStudio

Brennan & Matt

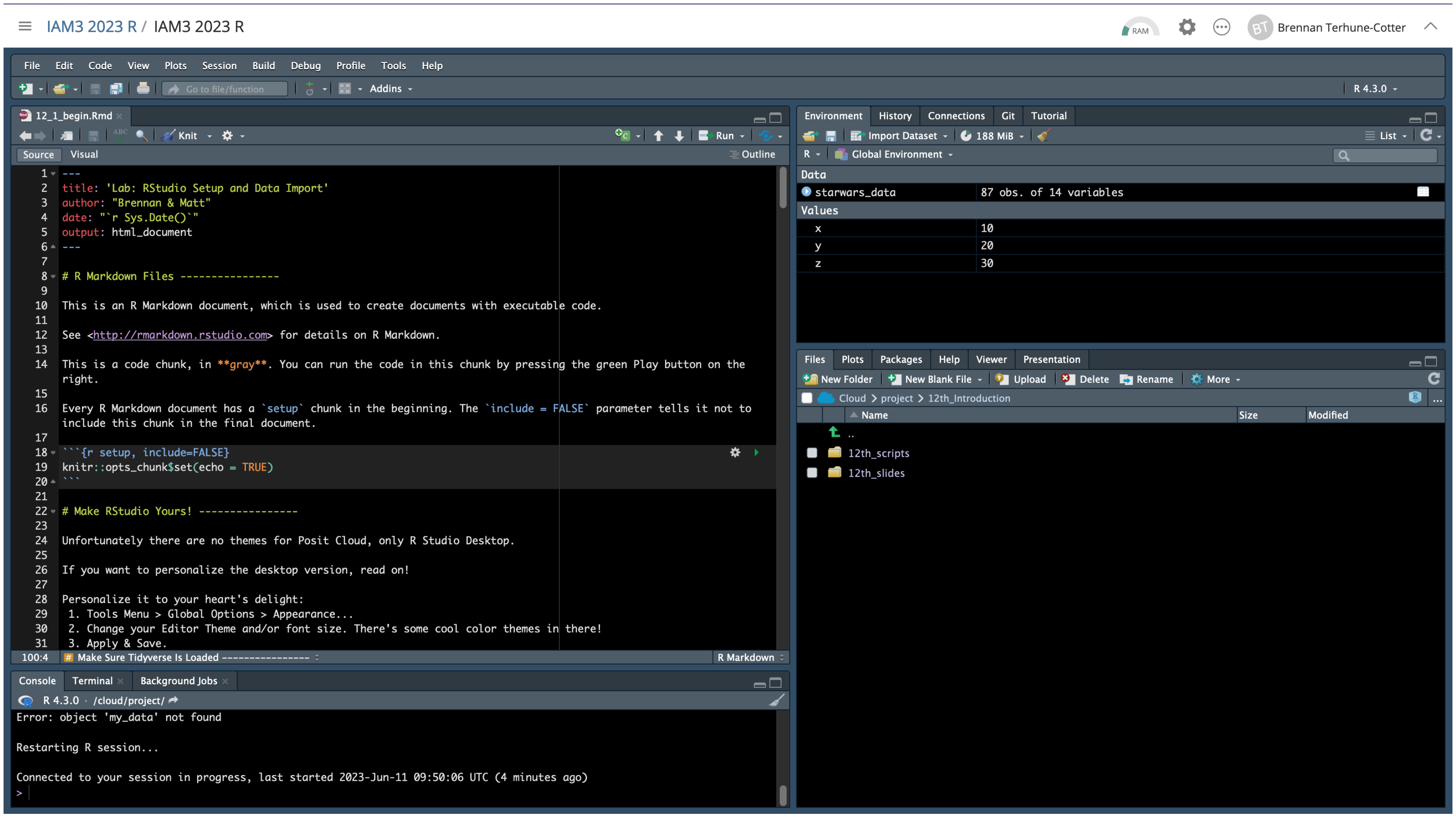
R Studio

- The most popular open-source IDE, developed by Posit
  - IDE = **i**ntegrated **d**evelopment **e**nvironment
  - Basically a fancy word for software that makes your life easier when coding ☺
- Can be downloaded as desktop software
  - Recommended if you want to keep using R after the workshop!
- Can use on the cloud
  - Guided labs will be via RStudio Cloud

RStudio Desktop vs. Cloud

- Posit has a desktop version of RStudio and a cloud version.
- For guided labs, we will be using the cloud to reduce time spent on troubleshooting while in class.
- We recommend familiarizing yourself with the desktop version if you want to keep using R after the workshop.
- We are available to help troubleshoot with the desktop version ☺

Posit Cloud


IAM3 2023 R / IAM3 2023 R
Brennan Terhune-Cotter
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R 4.3.0
12_1_begin.Rmd
Knit
Run
Source Visual
Outline
Environment History Connections Git Tutorial
Import Dataset
188 MiB
List
Global Environment
Files Plots Packages Help Viewer Presentation
New Folder New Blank File Upload Delete Rename More
Cloud > project > 12th_Introduction
Name Size Modified
Console Terminal Background Jobs
R Markdown

~~~
---
title: 'Lab: RStudio Setup and Data Import'
author: "Brennan & Matt"
date: "`r Sys.Date()`"
output: html_document
---

# R Markdown Files ----------------

This is an R Markdown document, which is used to create documents with executable code.

See <http://rmarkdown.rstudio.com> for details on R Markdown.

This is a code chunk, in **gray**. You can run the code in this chunk by pressing the green Play button on the
right.

Every R Markdown document has a `setup` chunk in the beginning. The `include = FALSE` parameter tells it not to
include this chunk in the final document.

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

# Make RStudio Yours! ----------------

Unfortunately there are no themes for Posit Cloud, only R Studio Desktop.

If you want to personalize the desktop version, read on!

Personalize it to your heart's delight:
 1. Tools Menu > Global Options > Appearance...
 2. Change your Editor Theme and/or font size. There's some cool color themes in there!
 3. Apply & Save.
~~~

100:4 # Make Sure Tidyverse Is Loaded ----------------

Data

starwars_data	87 obs. of 14 variables

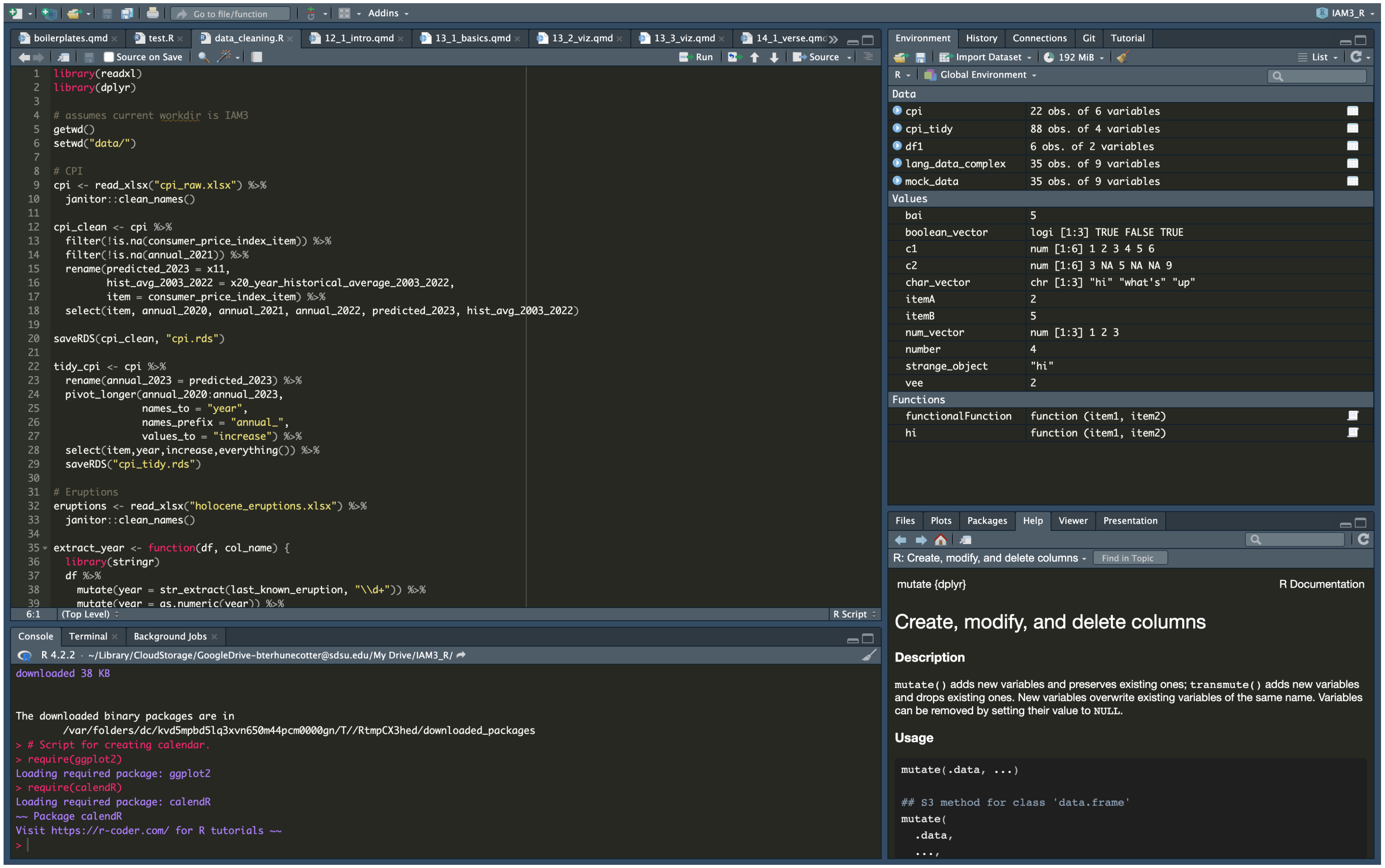
Values

x	10
y	20
z	30

- ..
- 12th_scripts
- 12th_slides

```
R 4.3.0 · /cloud/project/
Error: object 'my_data' not found

Restarting R session...

Connected to your session in progress, last started 2023-Jun-11 09:50:06 UTC (4 minutes ago)
>
```

Posit RStudio (Desktop)



```r
library(readxl)
library(dplyr)

# assumes current workdir is IAM3
getwd()
setwd("data/")

# CPI
cpi <- read_xlsx("cpi_raw.xlsx") %>%
  janitor::clean_names()

cpi_clean <- cpi %>%
  filter(!is.na(consumer_price_index_item)) %>%
  filter(!is.na(annual_2021)) %>%
  rename(predicted_2023 = x11,
         hist_avg_2003_2022 = x20_year_historical_average_2003_2022,
         item = consumer_price_index_item) %>%
  select(item, annual_2020, annual_2021, annual_2022, predicted_2023, hist_avg_2003_2022)

saveRDS(cpi_clean, "cpi.rds")

tidy_cpi <- cpi %>%
  rename(annual_2023 = predicted_2023) %>%
  pivot_longer(annual_2020:annual_2023,
               names_to = "year",
               names_prefix = "annual_",
               values_to = "increase") %>%
  select(item,year,increase,everything()) %>%
  saveRDS("cpi_tidy.rds")

# Eruptions
eruptions <- read_xlsx("holocene_eruptions.xlsx") %>%
  janitor::clean_names()

extract_year <- function(df, col_name) {
  library(stringr)
  df %>%
    mutate(year = str_extract(last_known_eruption, "\\d+")) %>%
    mutate(year = as.numeric(year)) %>%
```

```
R 4.2.2 · ~/Library/CloudStorage/GoogleDrive-bterhunecotter@sdsu.edu/My Drive/IAM3_R/
downloaded 38 KB


The downloaded binary packages are in
        /var/folders/dc/kvd5mpbd5lq3xvn650m44pcm0000gn/T//RtmpCX3hed/downloaded_packages
> # Script for creating calendar.
> require(ggplot2)
Loading required package: ggplot2
> require(calendR)
Loading required package: calendR
~~ Package calendR
Visit https://r-coder.com/ for R tutorials ~~
> 
```

```
Data
cpi                  22 obs. of 6 variables
cpi_tidy             88 obs. of 4 variables
df1                  6 obs. of 2 variables
lang_data_complex    35 obs. of 9 variables
mock_data            35 obs. of 9 variables
Values
bai                  5
boolean_vector       logi [1:3] TRUE FALSE TRUE
c1                   num [1:6] 1 2 3 4 5 6
c2                   num [1:6] 3 NA 5 NA NA 9
char_vector          chr [1:3] "hi" "what's" "up"
itemA                2
itemB                5
num_vector           num [1:3] 1 2 3
number               4
strange_object       "hi"
vee                  2
Functions
functionalFunction   function (item1, item2)
hi                   function (item1, item2)
```

mutate {dplyr} R Documentation

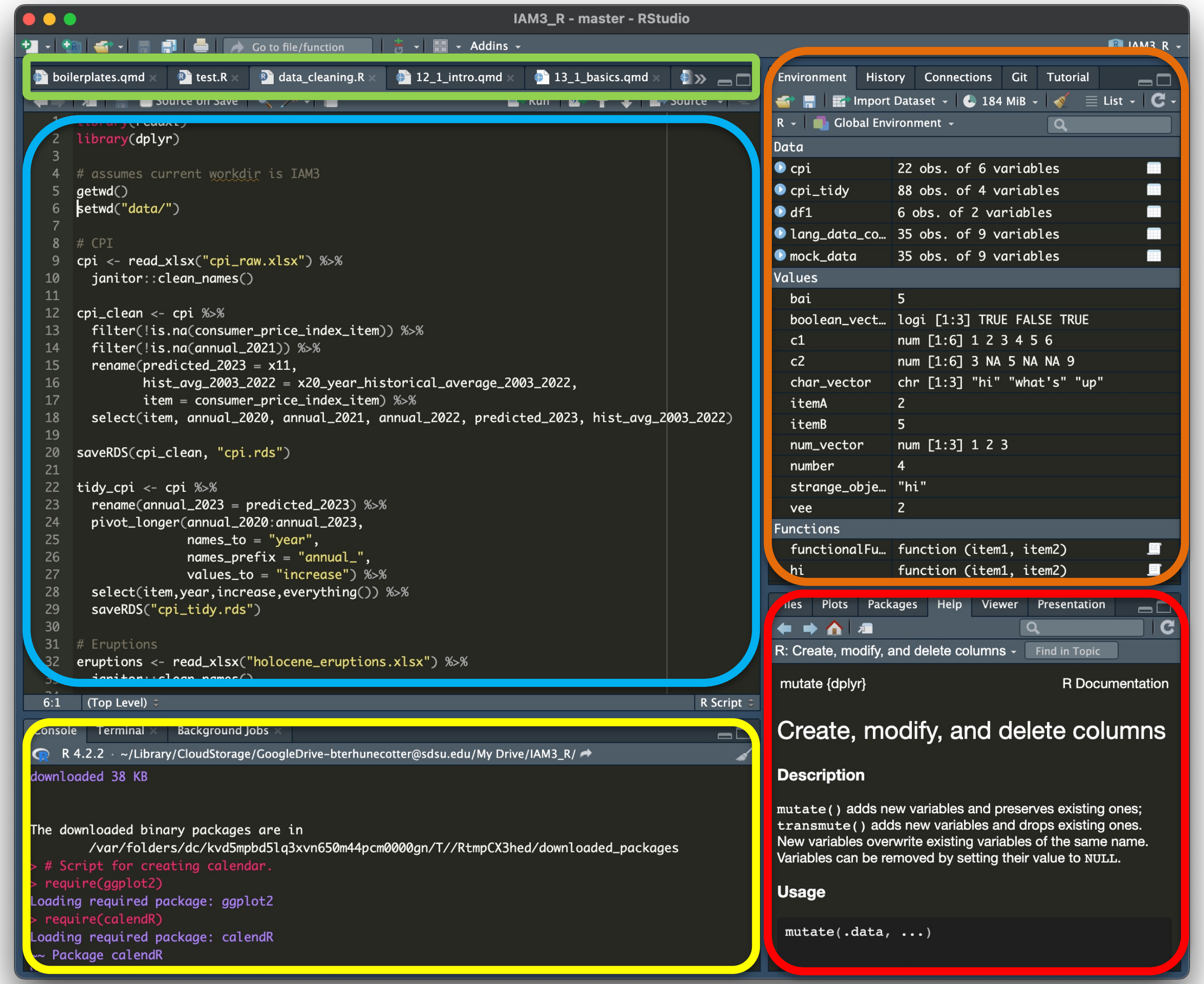
Create, modify, and delete columns

Description

`mutate()` adds new variables and preserves existing ones; `transmute()` adds new variables and drops existing ones. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to `NULL`.

Usage

```r
mutate(.data, ...)

## S3 method for class 'data.frame'
mutate(
  .data,
  ...,
```



```r
library(readxl)
library(dplyr)

# assumes current workdir is IAM3
getwd()
setwd("data/")

# CPI
cpi <- read_xlsx("cpi_raw.xlsx") %>%
  janitor::clean_names()

cpi_clean <- cpi %>%
  filter(!is.na(consumer_price_index_item)) %>%
  filter(!is.na(annual_2021)) %>%
  rename(predicted_2023 = x11,
         hist_avg_2003_2022 = x20_year_historical_average_2003_2022,
         item = consumer_price_index_item) %>%
  select(item, annual_2020, annual_2021, annual_2022, predicted_2023, hist_avg_2003_2022)

saveRDS(cpi_clean, "cpi.rds")

tidy_cpi <- cpi %>%
  rename(annual_2023 = predicted_2023) %>%
  pivot_longer(annual_2020:annual_2023,
               names_to = "year",
               names_prefix = "annual_",
               values_to = "increase") %>%
  select(item,year,increase,everything()) %>%
  saveRDS("cpi_tidy.rds")

# Eruptions
eruptions <- read_xlsx("holocene_eruptions.xlsx") %>%
  janitor::clean_names()
```

```
Data
cpi               22 obs. of 6 variables
cpi_tidy          88 obs. of 4 variables
df1               6 obs. of 2 variables
lang_data_co…     35 obs. of 9 variables
mock_data         35 obs. of 9 variables
Values
bai               5
boolean_vect…     logi [1:3] TRUE FALSE TRUE
c1                num [1:6] 1 2 3 4 5 6
c2                num [1:6] 3 NA 5 NA NA 9
char_vector       chr [1:3] "hi" "what's" "up"
itemA             2
itemB             5
num_vector        num [1:3] 1 2 3
number            4
strange_obje…     "hi"
vee               2
Functions
functionalFu…     function (item1, item2)
hi                function (item1, item2)
```

```
R 4.2.2 · ~/Library/CloudStorage/GoogleDrive-bterhunecotter@sdsu.edu/My Drive/IAM3_R/
downloaded 38 KB


The downloaded binary packages are in
        /var/folders/dc/kvd5mpbd5lq3xvn650m44pcm0000gn/T//RtmpCX3hed/downloaded_packages
> # Script for creating calendar.
> require(ggplot2)
Loading required package: ggplot2
> require(calendR)
Loading required package: calendR
~ Package calendR
```

R: Create, modify, and delete columns

mutate {dplyr} R Documentation

Create, modify, and delete columns

Description

`mutate()` adds new variables and preserves existing ones; `transmute()` adds new variables and drops existing ones. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to `NULL`.

Usage

```r
mutate(.data, ...)
```

Your open files (scripts, data frames, documents, etc.)

The source: where you'll do 99% of your coding

The console: displays errors and output; write one-time commands

Displays all objects in your environment

Displays help documentation for functions, or plots, or other things

Source

- *Source* is where you add code to your script(s).
- It works like any document: you write code, Save As a file, and save as often as possible ☺
- You also will want to run code you write here pretty often

Console

- *Console* shows you the *output* of all code you run, including source code
- When your code doesn't work, error messages show up in the console.
  - Caution! Error messages can be frustrating and do not tell you the *real* problem with the code. They just tell you what the computer noticed.
- You can also type and run "temporary" code in the console.

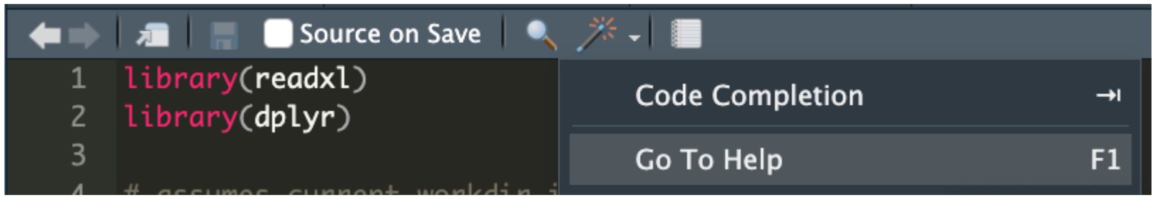
Environment

- **Environment** shows all active objects and custom functions in your global environment
  - This is <u>critical</u> to understanding what your code has access to!
- **History** shows your command history *(I've never used this)*

(I've never used Packages, Viewer, or Presentations)

Files/Plots/Help

- **Files** displays the files in your working directory (current folder)
- **Plots** displays plots when you generate them
- **Help** displays documentation for functions
  - *Very useful!!*
  - Press **F1** or **magic wand -> Go To Help** when your text cursor (|) is on a function name:

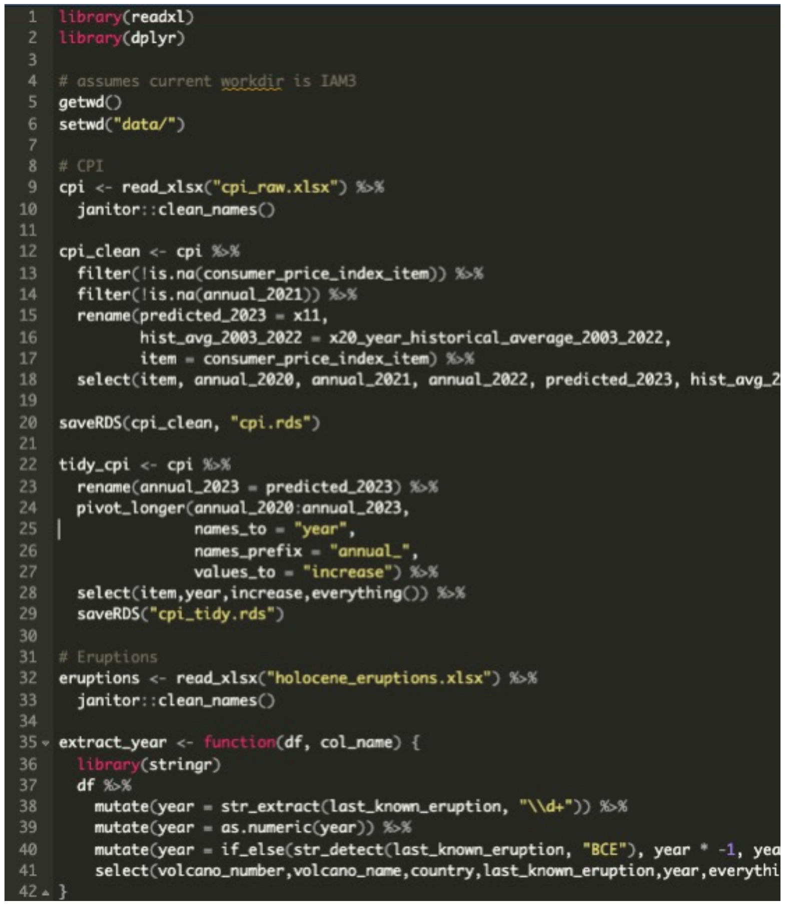


```
library(readxl)
library(dplyr)

# assumes current workdir i
```

  - **OR** search for a function in the search box

(I've never used Packages, Viewer, or Presentations)

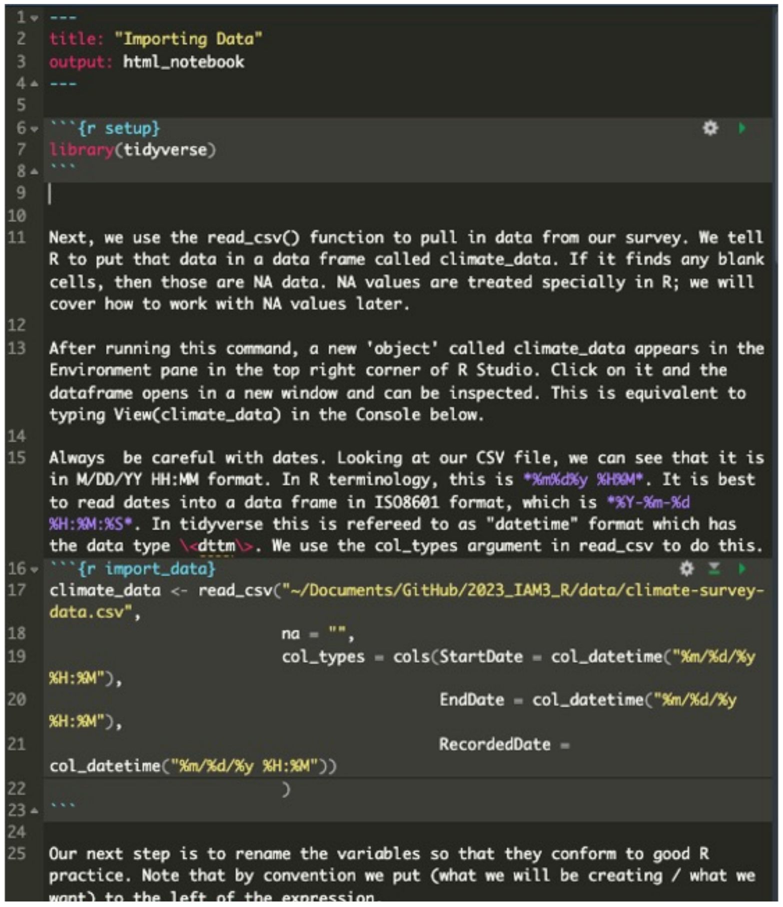
R Scripts



```r
library(readxl)
library(dplyr)

# assumes current workdir is IAM3
getwd()
setwd("data/")

# CPI
cpi <- read_xlsx("cpi_raw.xlsx") %>%
  janitor::clean_names()

cpi_clean <- cpi %>%
  filter(!is.na(consumer_price_index_item)) %>%
  filter(!is.na(annual_2021)) %>%
  rename(predicted_2023 = x11,
         hist_avg_2003_2022 = x20_year_historical_average_2003_2022,
         item = consumer_price_index_item) %>%
  select(item, annual_2020, annual_2021, annual_2022, predicted_2023, hist_avg_2

saveRDS(cpi_clean, "cpi.rds")

tidy_cpi <- cpi %>%
  rename(annual_2023 = predicted_2023) %>%
  pivot_longer(annual_2020:annual_2023,
               names_to = "year",
               names_prefix = "annual_",
               values_to = "increase") %>%
  select(item,year,increase,everything()) %>%
  saveRDS("cpi_tidy.rds")

# Eruptions
eruptions <- read_xlsx("holocene_eruptions.xlsx") %>%
  janitor::clean_names()

extract_year <- function(df, col_name) {
  library(stringr)
  df %>%
    mutate(year = str_extract(last_known_eruption, "\\d+")) %>%
    mutate(year = as.numeric(year)) %>%
    mutate(year = if_else(str_detect(last_known_eruption, "BCE"), year * -1, yea
    select(volcano_number,volcano_name,country,last_known_eruption,year,everythi
}
```

- *Scripts* have *code* which can be executed
- R scripts end with an '.R' extension
- Everything in an R script will be executed unless it is *commented out* with **#** at the beginning of the line

R Markdown



~~~
---
title: "Importing Data"
output: html_notebook
---

```{r setup}
library(tidyverse)
~~~
|

Next, we use the read_csv() function to pull in data from our survey. We tell R to put that data in a data frame called climate_data. If it finds any blank cells, then those are NA data. NA values are treated specially in R; we will cover how to work with NA values later.

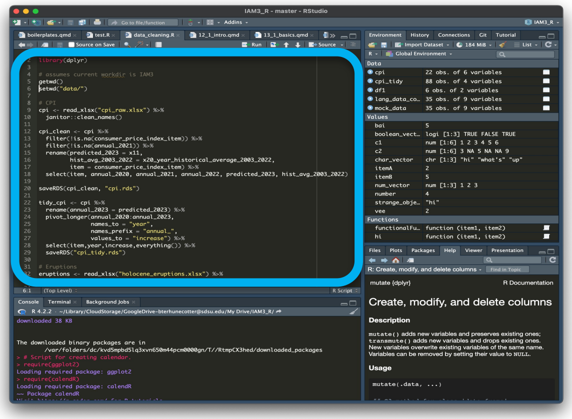
After running this command, a new 'object' called climate_data appears in the Environment pane in the top right corner of R Studio. Click on it and the dataframe opens in a new window and can be inspected. This is equivalent to typing View(climate_data) in the Console below.

Always be careful with dates. Looking at our CSV file, we can see that it is in M/DD/YY HH:MM format. In R terminology, this is *%m%d%y %H%M*. It is best to read dates into a data frame in ISO8601 format, which is *%Y-%m-%d %H:%M:%S*. In tidyverse this is refereed to as "datetime" format which has the data type \<dttm\>. We use the col_types argument in read_csv to do this.
```{r import_data}
climate_data <- read_csv("~/Documents/GitHub/2023_IAM3_R/data/climate-survey-data.csv",
                         na = "",
                         col_types = cols(StartDate = col_datetime("%m/%d/%y %H:%M"),
                                          EndDate = col_datetime("%m/%d/%y %H:%M"),
                                          RecordedDate = col_datetime("%m/%d/%y %H:%M"))
                         )
```

Our next step is to rename the variables so that they conform to good R practice. Note that by convention we put (what we will be creating / what we want) to the left of the expression.
```

- RStudio also provides R Markdown/Notebook documents (".Rmd")
- Rmd files are *text files* that contain executable *code chunks*
- Better when you want to type a lot of non-code text
  - No need to comment out everything!
```

# Running Source Code in R Scripts

- You run source code in R scripts by:
  - Pressing CMD-ENTER (or CTRL-ENTER for PC)
    - Runs the line of code your text cursor is on
  - Highlighting code and pressing CMD-ENTER
    - Runs everything you highlighted
- If you want to run the entire script:
  - CMD-A then CMD-ENTER
  - If you accidentally press ENTER and erase everything, CMD-Z!



```r
library(dplyr)

# assumes current workdir is IAM3
getwd()
setwd("data/")

# CPI
cpi <- read_xlsx("cpi_raw.xlsx") %>%
  janitor::clean_names()

cpi_clean <- cpi %>%
  filter(!is.na(consumer_price_index_item)) %>%
  filter(!is.na(annual_2021)) %>%
  rename(predicted_2023 = x11,
         hist_avg_2003_2022 = x20_year_historical_average_2003_2022,
         item = consumer_price_index_item) %>%
  select(item, annual_2020, annual_2021, annual_2022, predicted_2023, hist_avg_2003_2022)

saveRDS(cpi_clean, "cpi.rds")

tidy_cpi <- cpi %>%
  rename(annual_2023 = predicted_2023) %>%
  pivot_longer(annual_2020:annual_2023,
               names_to = "year",
               names_prefix = "annual_",
               values_to = "increase") %>%
  select(item,year,increase,everything()) %>%
  saveRDS("cpi_tidy.rds")

# Eruptions
eruptions <- read_xlsx("holocene_eruptions.xlsx") %>%
```

```
R 4.2.2 · ~/Library/CloudStorage/GoogleDrive-bterhunecotter@sdsu.edu/My Drive/IAM3_R/
downloaded 38 KB


The downloaded binary packages are in
        /var/folders/dc/kvd5mpbd5lq3xvn650m44pcm0000gn/T//RtmpCX3hed/downloaded_packages
> # Script for creating calendar.
> require(ggplot2)
Loading required package: ggplot2
> require(calendR)
Loading required package: calendR
~~ Package calendR
```

Data

cpi	22 obs. of 6 variables
cpi_tidy	88 obs. of 4 variables
df1	6 obs. of 2 variables
lang_data_co…	35 obs. of 9 variables
mock_data	35 obs. of 9 variables

Values

bai	5
boolean_vect…	logi [1:3] TRUE FALSE TRUE
c1	num [1:6] 1 2 3 4 5 6
c2	num [1:6] 3 NA 5 NA NA 9
char_vector	chr [1:3] "hi" "what's" "up"
itemA	2
itemB	5
num_vector	num [1:3] 1 2 3
number	4
strange_obje…	"hi"
vee	2

Functions

functionalFu…	function (item1, item2)
hi	function (item1, item2)

R: Create, modify, and delete columns

mutate {dplyr} R Documentation

Create, modify, and delete columns

Description

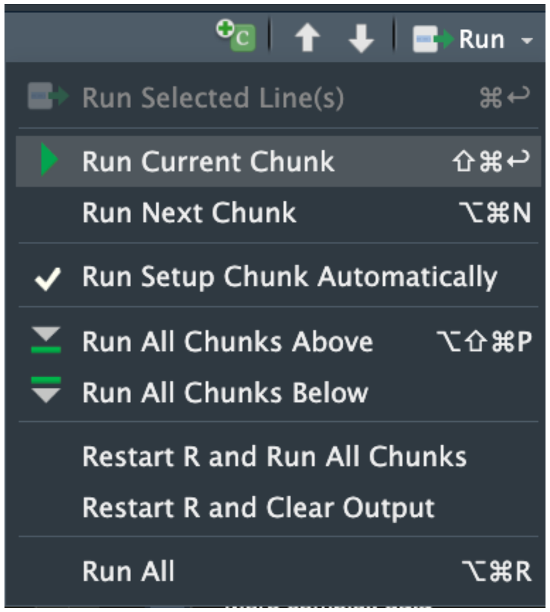
`mutate()` adds new variables and preserves existing ones; `transmute()` adds new variables and drops existing ones. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to `NULL`.

Usage

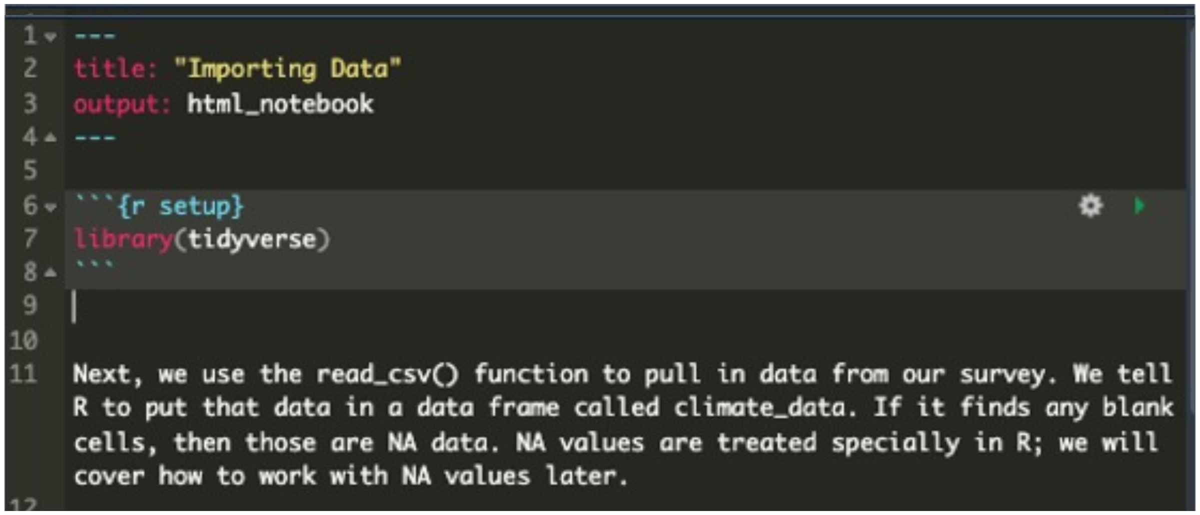
```r
mutate(.data, ...)
```

Running Source Code in R Markdown

- .Rmd documents have code in *chunks*.
- Run code in a chunk by pressing ▶
- You can also run chunks in different ways ->



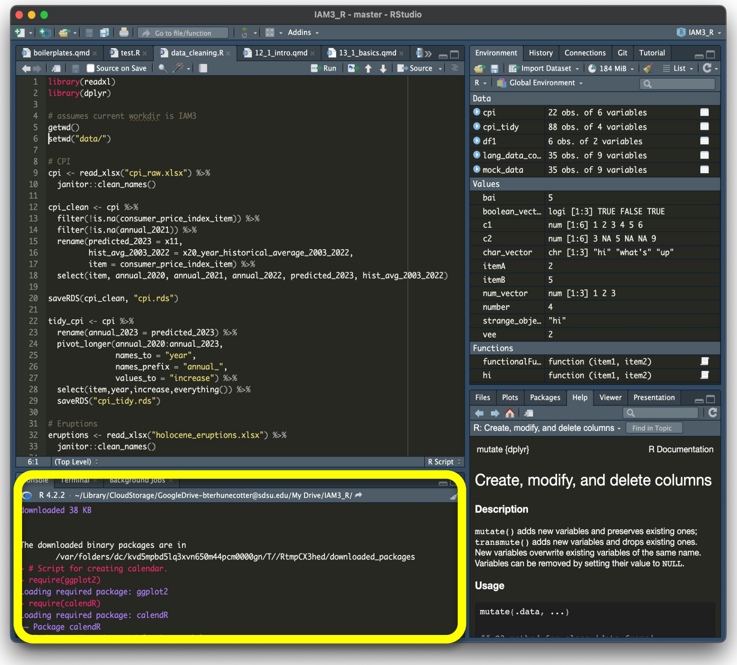
Run
Run Selected Line(s) ⌘↩
Run Current Chunk ⇧⌘↩
Run Next Chunk ⌥⌘N
✓ Run Setup Chunk Automatically
Run All Chunks Above ⌥⇧⌘P
Run All Chunks Below
Restart R and Run All Chunks
Restart R and Clear Output
Run All ⌥⌘R



~~~
---
title: "Importing Data"
output: html_notebook
---

```{r setup}
library(tidyverse)
```


Next, we use the read_csv() function to pull in data from our survey. We tell
R to put that data in a data frame called climate_data. If it finds any blank
cells, then those are NA data. NA values are treated specially in R; we will
cover how to work with NA values later.
~~~

Running Console Code

- Coding in the console works like in any console: you type out your command and press Enter ☺
  - You type in the line starting with **>**
- Meant to be for one-time code only
  - Viewing data
  - Doing calculations
  - Checking things before adding to source



```r
library(readxl)
library(dplyr)

# assumes current workdir is IAM3
getwd()
setwd("data/")

# CPI
cpi <- read_xlsx("cpi_raw.xlsx") %>%
  janitor::clean_names()

cpi_clean <- cpi %>%
  filter(!is.na(consumer_price_index_item)) %>%
  filter(!is.na(annual_2021)) %>%
  rename(predicted_2023 = x11,
         hist_avg_2003_2022 = x20_year_historical_average_2003_2022,
         item = consumer_price_index_item) %>%
  select(item, annual_2020, annual_2021, annual_2022, predicted_2023, hist_avg_2003_2022)

saveRDS(cpi_clean, "cpi.rds")

tidy_cpi <- cpi %>%
  rename(annual_2023 = predicted_2023) %>%
  pivot_longer(annual_2020:annual_2023,
               names_to = "year",
               names_prefix = "annual_",
               values_to = "increase") %>%
  select(item,year,increase,everything()) %>%
  saveRDS("cpi_tidy.rds")

# Eruptions
eruptions <- read_xlsx("holocene_eruptions.xlsx") %>%
  janitor::clean_names()
```

Data	
cpi	22 obs. of 6 variables
cpi_tidy	88 obs. of 4 variables
df1	6 obs. of 2 variables
lang_data_co…	35 obs. of 9 variables
mock_data	35 obs. of 9 variables
Values	
bai	5
boolean_vect…	logi [1:3] TRUE FALSE TRUE
c1	num [1:6] 1 2 3 4 5 6
c2	num [1:6] 3 NA 5 NA NA 9
char_vector	chr [1:3] "hi" "what's" "up"
itemA	2
itemB	5
num_vector	num [1:3] 1 2 3
number	4
strange_obje…	"hi"
vee	2
Functions	
functionalFu…	function (item1, item2)
hi	function (item1, item2)

```
R 4.2.2 · ~/Library/CloudStorage/GoogleDrive-bterhunecotter@sdsu.edu/My Drive/IAM3_R/
downloaded 38 KB


The downloaded binary packages are in
        /var/folders/dc/kvd5mpbd5lq3xvn650m44pcm0000gn/T//RtmpCX3hed/downloaded_packages
> # Script for creating calendar.
> require(ggplot2)
Loading required package: ggplot2
> require(calendR)
Loading required package: calendR
~~ Package calendR
```

mutate {dplyr} R Documentation

Create, modify, and delete columns

Description

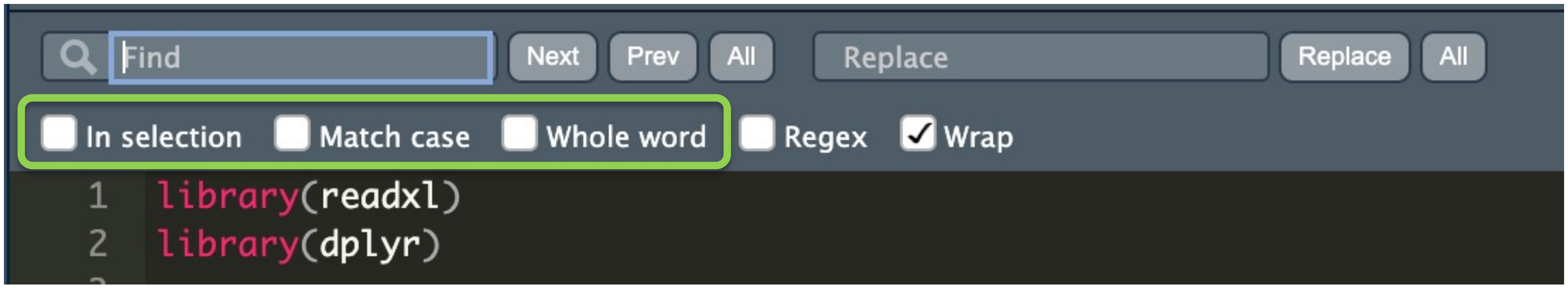
`mutate()` adds new variables and preserves existing ones; `transmute()` adds new variables and drops existing ones. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to `NULL`.

Usage

```
mutate(.data, ...)
```

RStudio Tips

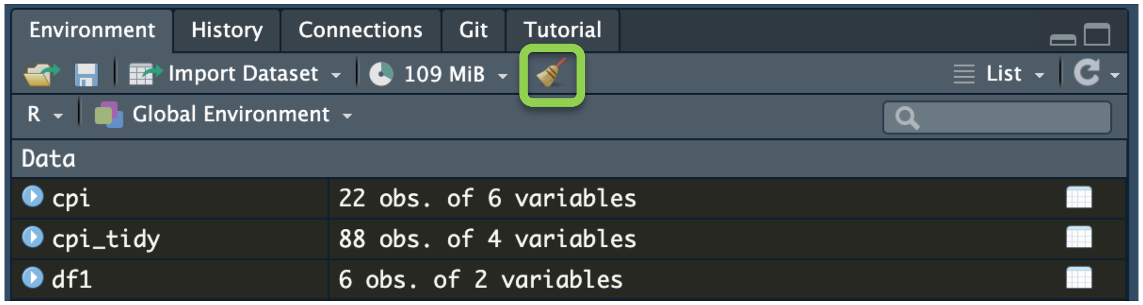
- Their **find and replace** function is excellent
  - CMD-F on Mac | CTRL-F on PC
- Find specific text (names, etc.) and replace one or all
  - CMD-Z | CTRL-Z will undo replacements



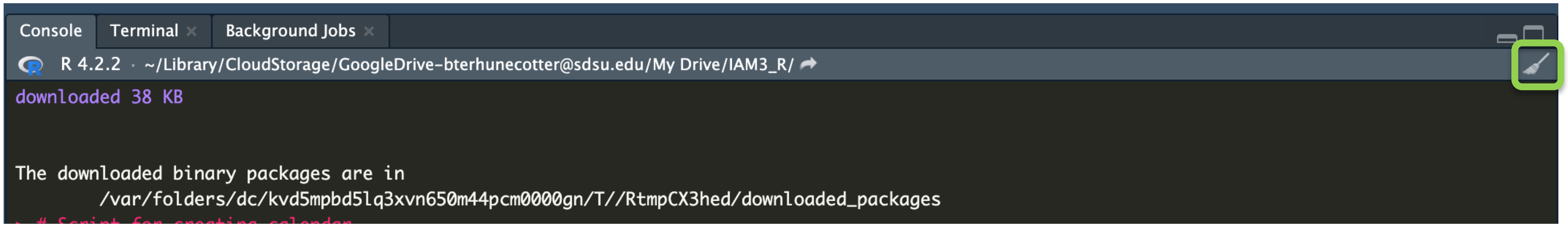
```
library(readxl)
library(dplyr)
```

RStudio Tips

- The **broom icon** will wipe clean your environment or your console history.
- **For your environment:** this is important to make sure your script doesn't call "leftover" objects which should not exist at that point in the script.
- **For your console history:** it will just keep you sane ☺



```
cpi        22 obs. of 6 variables
cpi_tidy   88 obs. of 4 variables
df1        6 obs. of 2 variables
```


Console
Terminal
Background Jobs
R 4.2.2 · ~/Library/CloudStorage/GoogleDrive-bterhunecotter@sdsu.edu/My Drive/IAM3_R/

```
downloaded 38 KB


The downloaded binary packages are in
	/var/folders/dc/kvd5mpbd5lq3xvn650m44pcm0000gn/T//RtmpCX3hed/downloaded_packages
```

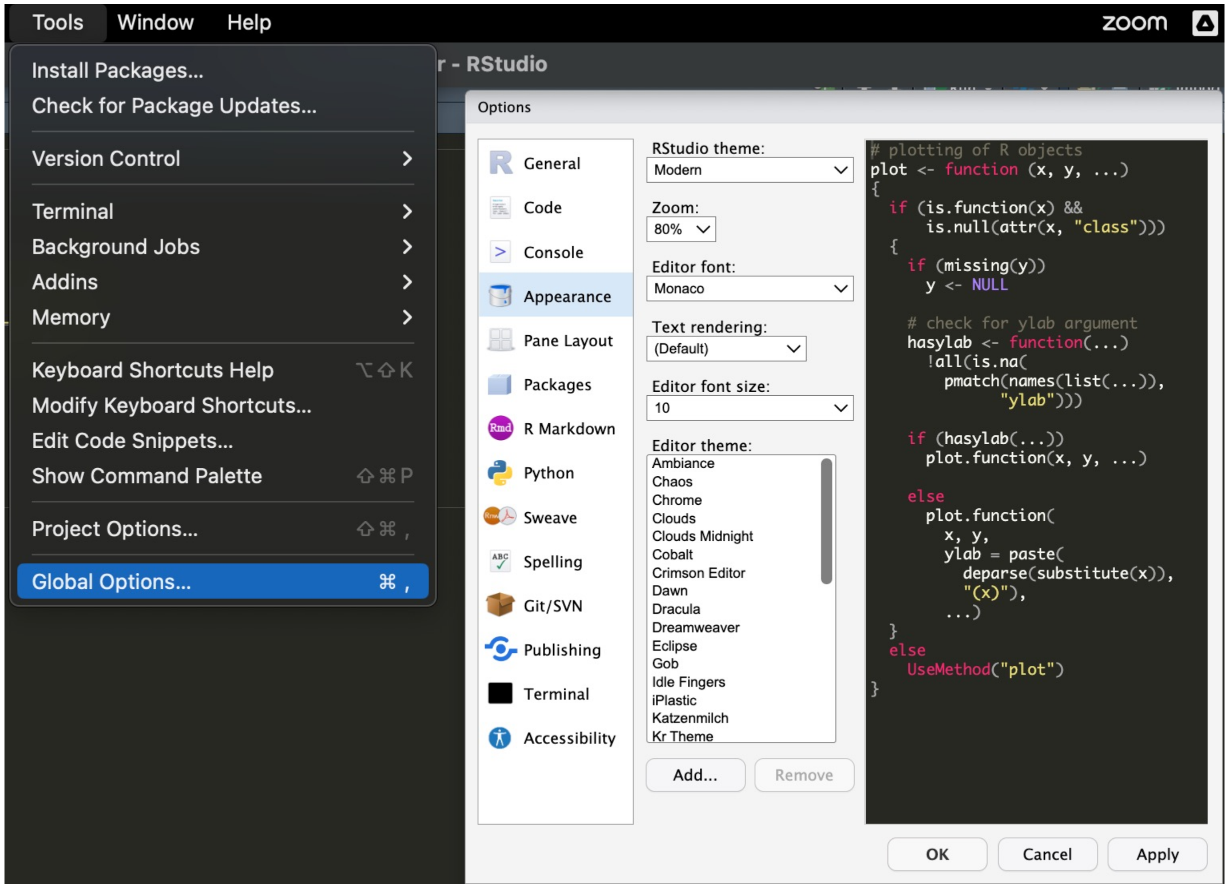
RStudio Tips

- To ensure you don't end up with a broken script, do this semi-frequently:
  1. Broom up everything in the Environment (and Console if you want, but don't need)
  2. Re-run your entire script:
     1. Press CMD-A (highlights everything)
     2. Press CMD-ENTER
  3. Fix the error that pops up and do steps 1-2 again
  4. You're done when everything runs correctly ☺

- This can be *super* annoying, but is critical to catching errors early
- The more often you do it, the less annoying it is because you know where the errors are

Picking an RStudio & Posit Cloud color theme!

- You won’t be a good coder if your screen doesn’t look cool!
- Click on `Tools > Global Options > Appearance` and pick your favorite theme.
- There are light and dark themes
- I like Monokai and Tomorrow Night; choose whatever you think is prettiest! ☺



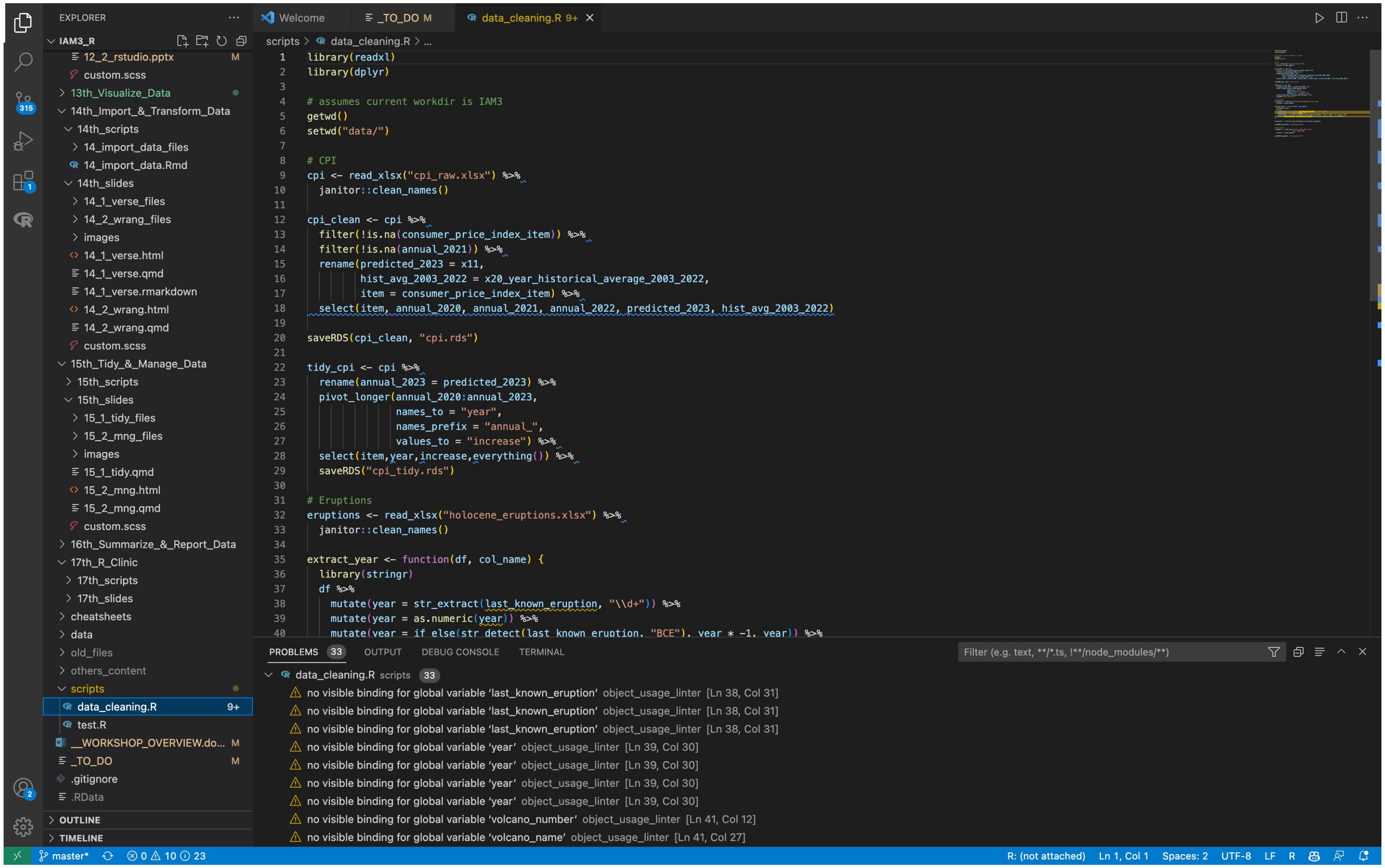
RStudio theme: Modern
Zoom: 80%
Editor font: Monaco
Text rendering: (Default)
Editor font size: 10
Editor theme:
Ambiance
Chaos
Chrome
Clouds
Clouds Midnight
Cobalt
Crimson Editor
Dawn
Dracula
Dreamweaver
Eclipse
Gob
Idle Fingers
iPlastic
Katzenmilch
Kr Theme

```
# 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")
}
```

Side Note…

- RStudio is not the only IDE for R
- Another good one is Visual Studio Code (VSC)
  - VSC is a great IDE for multiple programming languages
  - It is not specialized for R, but it is very polished and has cool features for programming.
    - GitHub Copilot (AI to help autocomplete code)
  - I don’t recommend VSC for learning R but if you want to use other languages or explore more features, check it out!

Visual Studio Code



```r
library(readxl)
library(dplyr)

# assumes current workdir is IAM3
getwd()
setwd("data/")

# CPI
cpi <- read_xlsx("cpi_raw.xlsx") %>%
  janitor::clean_names()

cpi_clean <- cpi %>%
  filter(!is.na(consumer_price_index_item)) %>%
  filter(!is.na(annual_2021)) %>%
  rename(predicted_2023 = x11,
         hist_avg_2003_2022 = x20_year_historical_average_2003_2022,
         item = consumer_price_index_item) %>%
  select(item, annual_2020, annual_2021, annual_2022, predicted_2023, hist_avg_2003_2022)

saveRDS(cpi_clean, "cpi.rds")

tidy_cpi <- cpi %>%
  rename(annual_2023 = predicted_2023) %>%
  pivot_longer(annual_2020:annual_2023,
               names_to = "year",
               names_prefix = "annual_",
               values_to = "increase") %>%
  select(item,year,increase,everything()) %>%
  saveRDS("cpi_tidy.rds")

# Eruptions
eruptions <- read_xlsx("holocene_eruptions.xlsx") %>%
  janitor::clean_names()

extract_year <- function(df, col_name) {
  library(stringr)
  df %>%
    mutate(year = str_extract(last_known_eruption, "\\d+")) %>%
    mutate(year = as.numeric(year)) %>%
    mutate(year = if_else(str_detect(last_known_eruption, "BCE"), year * -1, year)) %>%
```

```
data_cleaning.R scripts 33
⚠ no visible binding for global variable 'last_known_eruption' object_usage_linter [Ln 38, Col 31]
⚠ no visible binding for global variable 'last_known_eruption' object_usage_linter [Ln 38, Col 31]
⚠ no visible binding for global variable 'last_known_eruption' object_usage_linter [Ln 38, Col 31]
⚠ no visible binding for global variable 'year' object_usage_linter [Ln 39, Col 30]
⚠ no visible binding for global variable 'year' object_usage_linter [Ln 39, Col 30]
⚠ no visible binding for global variable 'year' object_usage_linter [Ln 39, Col 30]
⚠ no visible binding for global variable 'year' object_usage_linter [Ln 39, Col 30]
⚠ no visible binding for global variable 'volcano_number' object_usage_linter [Ln 41, Col 12]
⚠ no visible binding for global variable 'volcano_name' object_usage_linter [Ln 41, Col 27]
```

Next up…

- We will tell you how to import files into RStudio
- Then you will try importing your (or our) data into RStudio!