1. Introduction to R and Data Manipulation with Dplyr

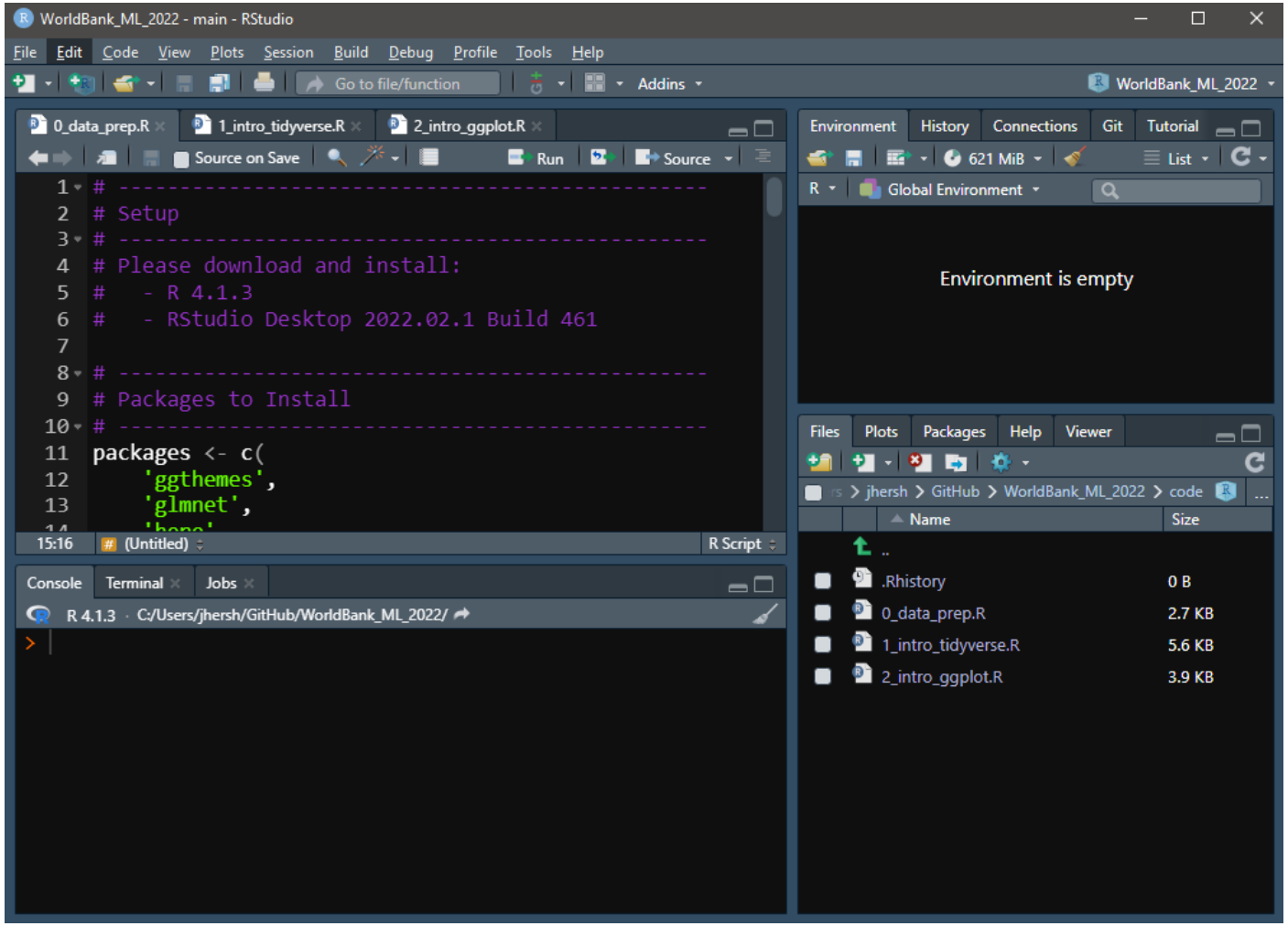
Jonathan Hersh, PhD (Chapman Argyros School of Business)

4/8/22

Section 1: Outline

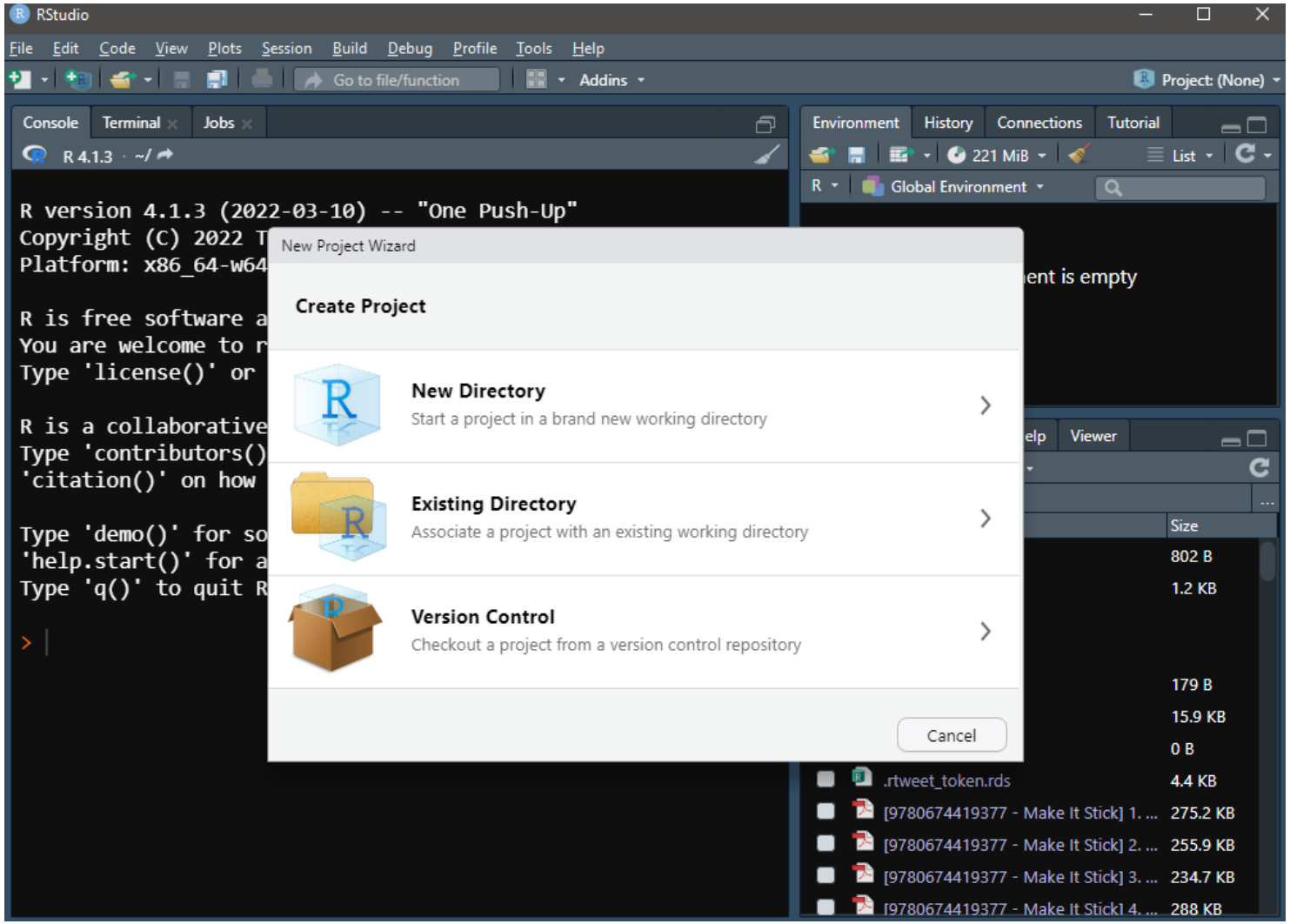
1. **R Projects**
2. **Data Analysis**
   - Loading data
   - Glimpse to view
   - Pipe operator
   - slice() to select rows
   - arrange() to order data frame
   - select() to choose variables
   - rename() to rename variables
   - filter() to select rows matching characteristics
   - mutate() to create new variables
   - group_by() and summarize() to create group

Rstudio Projects



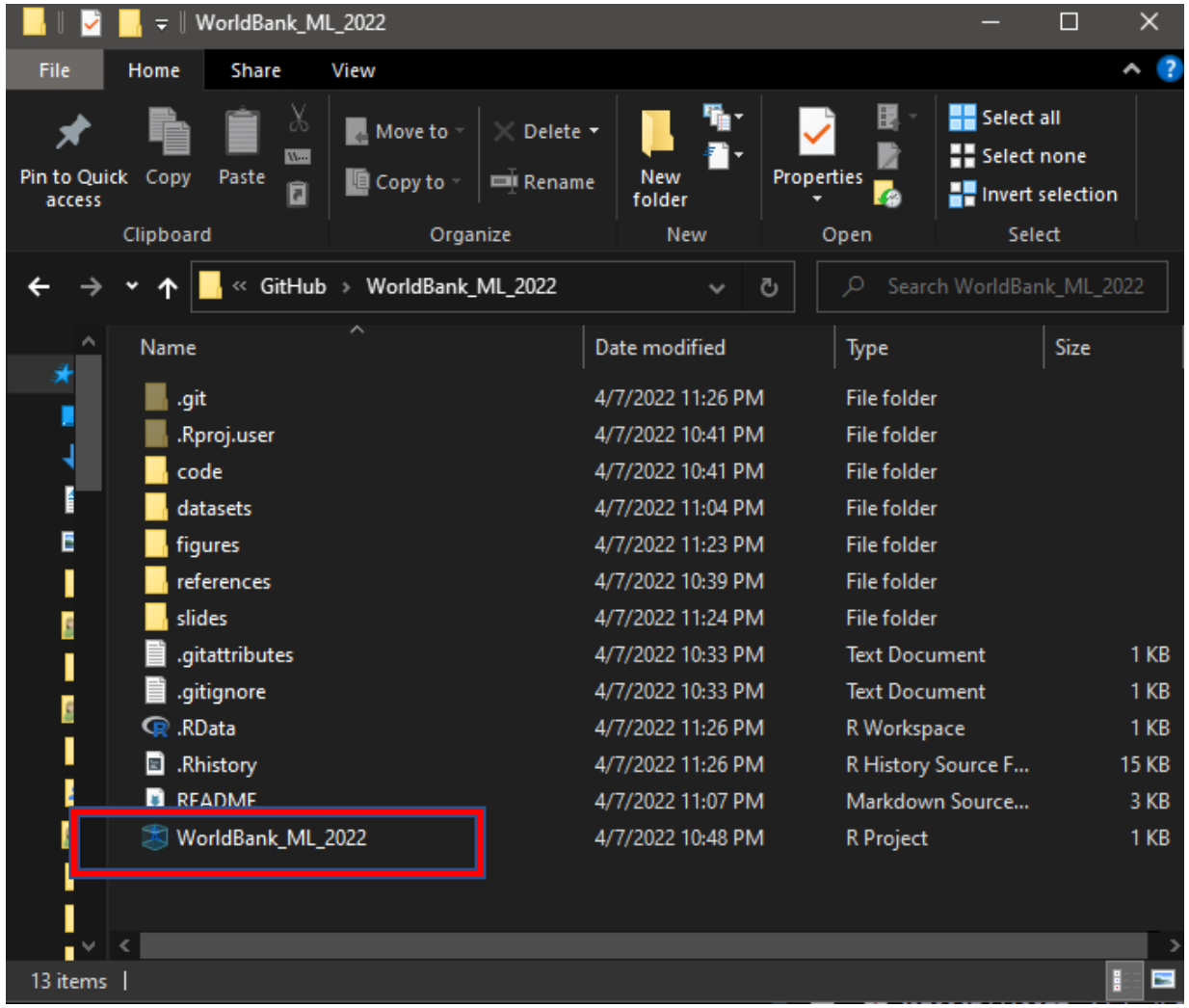
- Rstudio Projects are environments for your code
- They make your life easier, especially across teams and platforms
- Always use Rstudio Projects!

Rstudio Projects



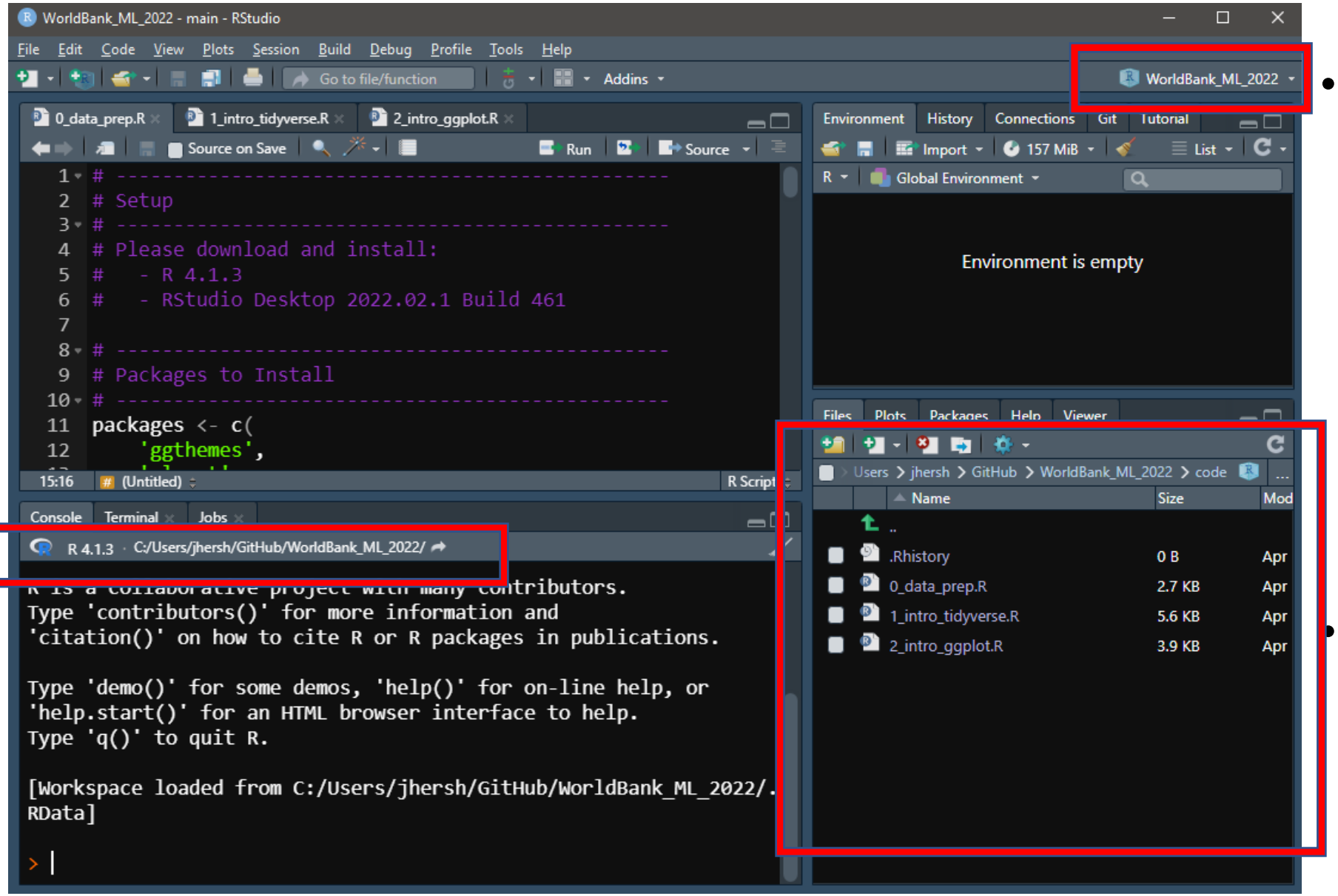
- I recommend you create a new project in your github repo folder
- Select file -> new project

Rstudio Projects



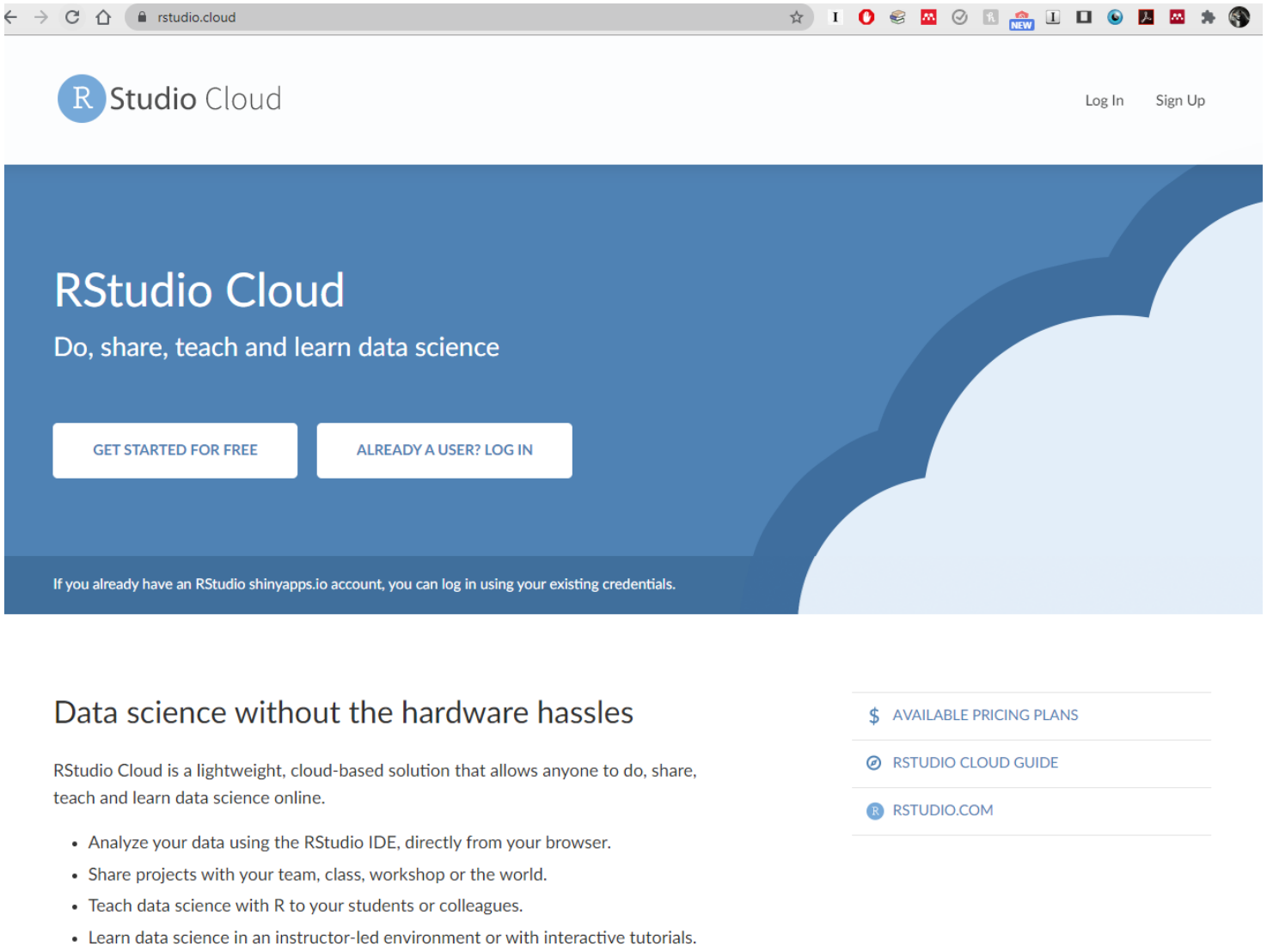
- This creates an .Rproj file you should click and open whenever you want to run code for this class

Rstudio Projects



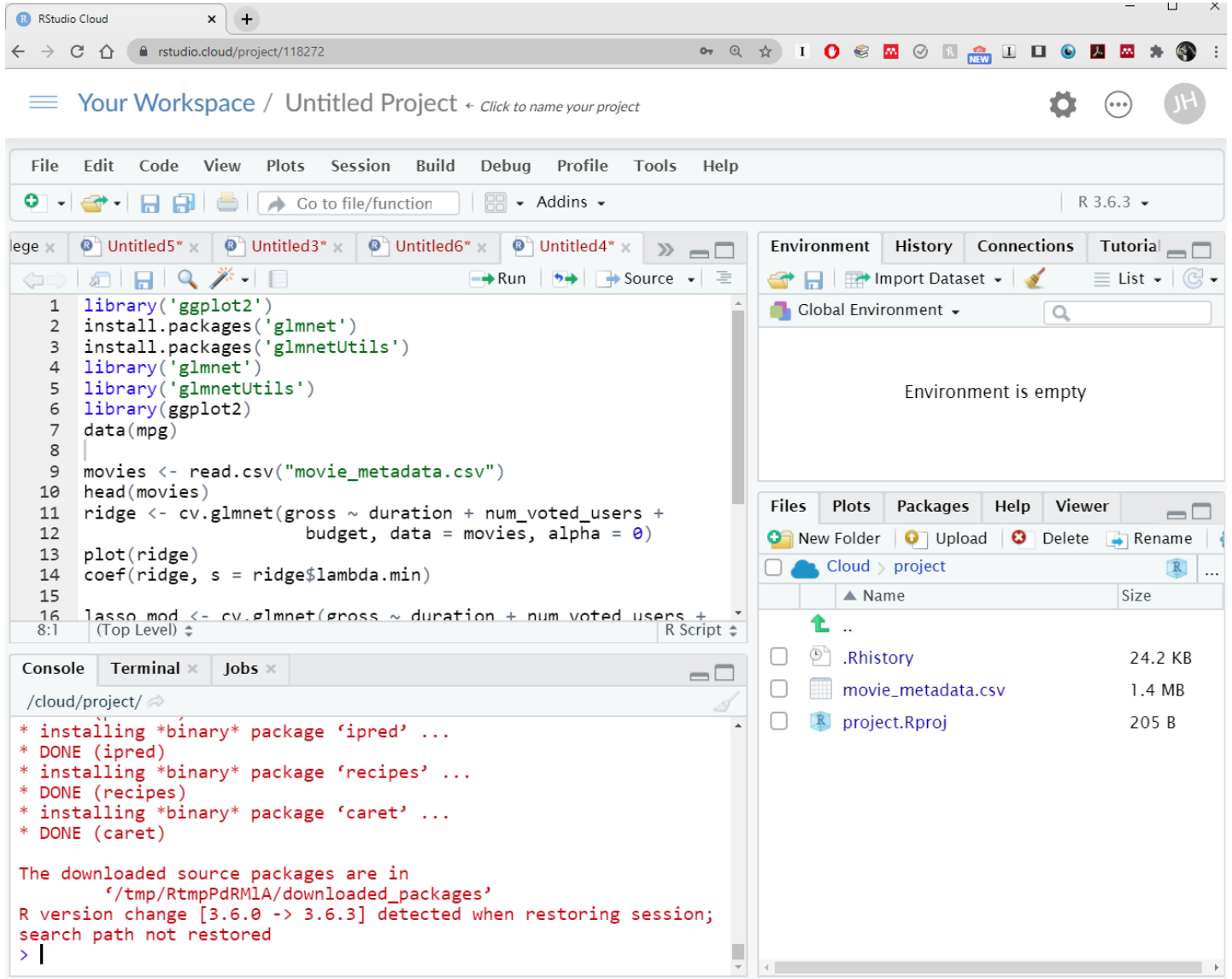
- You should see your project name in the top right and the "root directory" of the project in the bottom right
- All links to files, datasets are relative to this root folder

R Studio Cloud



- Go to rstudio.cloud if your version of R is ever not working

R Studio Cloud



- R Studio Cloud is a full featured version of R in your browser!

glimpse() to summarize the data

```
# ------------------------------------------------
# GLIMPSE to summarize data
# ------------------------------------------------
# let's summarize the IDB poverty data using the glimpse function

glimpse(CR_dat)
```

```
> glimpse(CR_dat)
Rows: 5,940
Columns: 19
$ household_ID <chr> "21eb7fcc1", "0e5d7a658", "2c7317ea8", "2b~
$ poor_stat    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ num_rooms    <int> 3, 4, 8, 5, 2, 3, 4, 2, 4, 3, 1, 5, 4, 5, ~
$ bathroom     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, ~
$ refrig       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
$ no_elect     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ no_toilet    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ~
$ comp         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ dep_rate     <dbl> 0.0000000, 1.0000000, 1.0000000, 1.0000000~
$ tv           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ~
$ mobile       <int> 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
$ num_hh       <int> 1, 1, 1, 4, 4, 2, 2, 4, 2, 2, 3, 1, 3, 2, ~
$ urban        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
$ mean_educ    <dbl> 10.000000, 12.000000, 11.000000, 11.000000~
$ num_children <int> 0, 0, 0, 2, 2, 1, 0, 2, 0, 0, 0, 0, 0, 0, ~
$ num_adults   <int> 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 3, 1, 3, 2, ~
$ num_elderly  <int> 0, 1, 1, 0, 0, 0, 1, 1, 0, 2, 0, 0, 1, 0, ~
$ disabled     <int> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ mar_stat     <fct> divorced, divorced, widowed, other, other,~
```

pipe operator %>%

```
# ------------------------------------------------
# Pipe Operator!
# ------------------------------------------------
# The pipe operator "%>%" is super useful!
# It allows us to execute a series of functions on an object in stages
# The general recipe is Data_Frame %>% function1() %>% function2() etc
# Functions are applied right to left

CR_dat %>% glimpse()
glimpse(CR_dat)

# cmd shift
CR_dat %>% glimpse()
glimpse(CR_dat)
```

slice() to select rows

```
# ------------------------------------------------
# Slice function: to select ROWS
# ------------------------------------------------
# SLICE: slice to view only the first 10 rows
CR_dat %>% slice(1:10)

# SLICE to view only rows 300 to 310
CR_dat %>% slice(300:310)
```

arrange() to order dataset

```
# ------------------------------------------------
# Arrange function: to ORDER dataset
# ------------------------------------------------

# arrange the dataframe in descending order by mean_educ
CR_dat %>%
    arrange(desc(mean_educ)) %>%
    head()

# arrange the dataframe in ascending order by mean_educ
CR_dat %>%
    arrange(mean_educ) %>%
    head()
```

select() to select columns in a dataset

```r
# ------------------------------------------------
# SELECT columns of the dataset using the 'select' function
# ------------------------------------------------
# select then pass to table function
CR_dat %>% select(poor_stat) %>% table()

# select only columns starting with particular characters
CR_dat %>%
    select(starts_with("num")) %>%
    head()

# remove variables using - operator
CR_dat %>%
    select(-num_rooms) %>%
    head()
```

rename() to rename variables in a dataset

```
# ------------------------------------------------
# RENAME variables using the RENAME function
# ------------------------------------------------
# note we must pass the DF back to the original data
CR_dat <- CR_dat %>%
    rename(HH_ID = household_ID)

CR_dat %>% names()

# change it back!
CR_dat <- CR_dat %>%
    rename(household_ID = HH_ID)
```

filter() to remove rows you don’t want

```
# ------------------------------------------------
# FILTER and ONLY allow certain rows using the FILTER function
# ------------------------------------------------
# only select households with poverty status
# and see # of rows
CR_dat %>%
    filter(poor_stat ==  1) %>%
    head()

CR_dat %>%
    filter(mar_stat == "divorced") %>%
    count()

CR_dat %>%
    filter(comp == 1 & num_hh > 3) %>%
    count()
```

mutate() to create new variables

```r
# ------------------------------------------------
# MUTATE to Transform variables in your dataset
# ------------------------------------------------
# adding new variables using mutate()
CR_dat <- CR_dat %>%
    mutate(mean_educ_sq = mean_educ * mean_educ,
           mean_educ_log = log(mean_educ + 1))

# see average education and educ squared
CR_dat %>% select(matches("educ")) %>% colMeans()

# Same thing, but using the package purrr to "map"
# the function mean to all the columns of the data frame
CR_dat %>% select(matches("educ")) %>% map_df(mean)
```

group_by() and summarize() to create group variables

```
# ------------------------------------------------
# Create summary statistics by GROUP using group_by()
# ------------------------------------------------
CR_dat <- CR_dat %>%
        # group by urban rural status
    group_by(urban)

glimpse(CR_dat)


# calculate average and sd of poverty by group
CR_urb <- CR_dat %>%
    # calculate average poor status
    summarize(pov_avg = mean(poor_stat),
              pov_sd = sd(poor_stat)) %>%
    print()
```

Data Wrangling
with dplyr and tidyr
Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

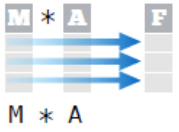
In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



Syntax - Helpful conventions for wrangling

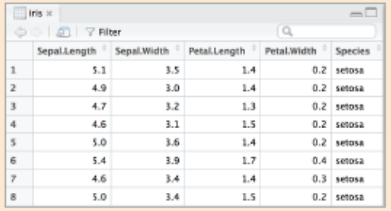
dplyr::**tbl_df(iris)**
Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**
Information dense summary of tbl data.

utils::**View(iris)**
View data set in spreadsheet-like display (note capital V).



	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

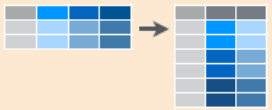
dplyr::**%>%**
Passes object on left hand side as first argument (or . argument) of function on righthand side.

```
x %>% f(y)  is the same as  f(x, y)
y %>% f(x, ., z)  is the same as  f(x, y, z )
```

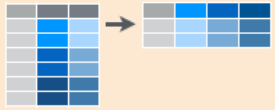
"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Reshaping Data - Change the layout of a data set



tidyr::**gather(cases, "year", "n", 2:4)**
Gather columns into rows.



tidyr::**spread(pollution, size, amount)**
Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**
Separate one column into several.

tidyr::**unite(data, col, ..., sep)**
Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**
Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**
Order rows by values of a column (low to high).

dplyr::**arrange(mtcars, desc(mpg))**
Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**
Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**
Extract rows that meet logical criteria.

dplyr::**distinct(iris)**
Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**
Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**
Randomly select n rows.

dplyr::**slice(iris, 10:15)**
Select rows by position.

dplyr::**top_n(storms, 2, date)**
Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**
Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains("."))
Select columns whose name contains a character string.

select(iris, ends_with("Length"))
Select columns whose name ends with a character string.

select(iris, everything())
Select every column.

select(iris, matches(".t."))
Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))
Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))
Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

https://www.rstudio.com/resources/cheatsheets/