Your country data and more on using R for SAE

Nairobi Workshop: Day 1

Ann-Kristin Kreutzmann

Josh Merfeld

August 26, 2024

Session 1: Country data in R

By you!

Welcome

- Welcome to the first day of the workshop!
- We are going to start with the task you were given as your pre-workshop assignment.

A quick reminder

- What did we ask from you?

1. Describe your country data set, e.g., number of observations and variables, content, sampling design, etc.
2. Describe the variable of interest, e.g., distribution, missing values, outstanding observations, etc.
3. Describe variables that you think are related to the variable of interest.
4. Demonstrate basic visualization of the selected indicators.

Session 2: Hands-on session

Data preparation for SAE

My slides

- Before we get into it, I have put all of my material on the web

- You can find my slides (along with copy-pasteable code) AND the data I'm using on my GitHub repository:
  - https://github.com/JoshMerfeld/nairobiworkshops
  - Scroll down to the bottom and you'll find all of the links you'll need.
  - I'd suggest you have these open during the workshop to make things easier.

Let’s get started!

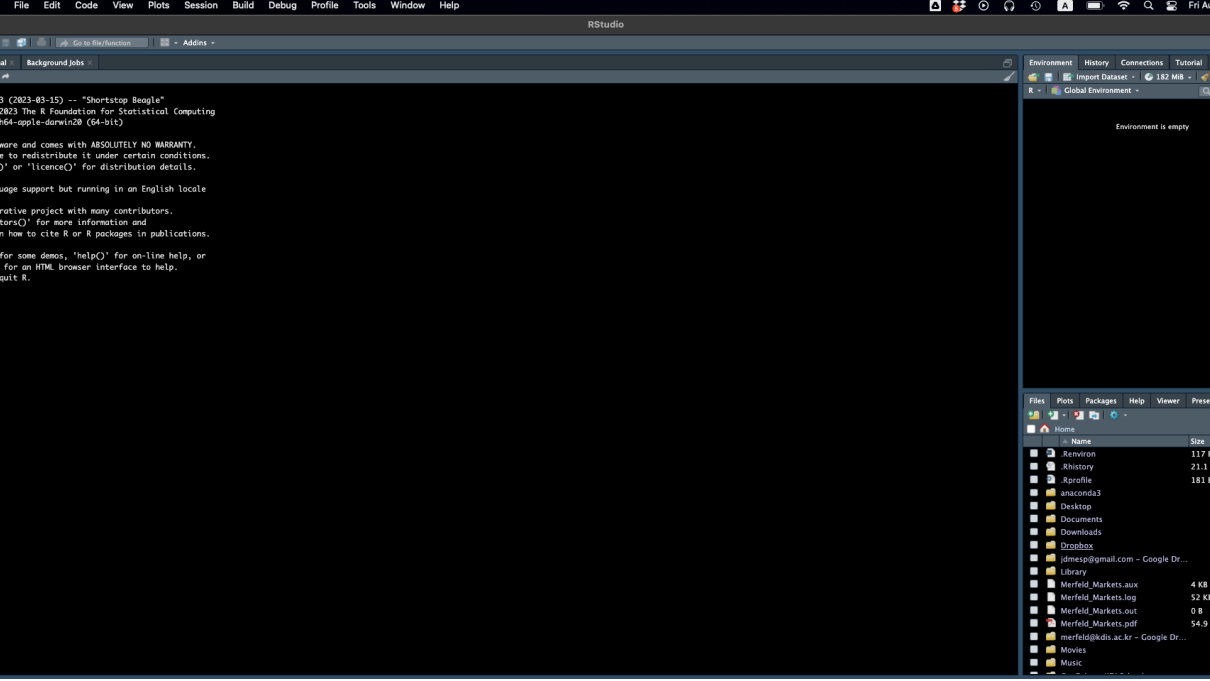
- Let’s get started with the hands-on session.
- As a first step, I’d like to hear from all of you:
  - How much experience do you have using R?
  - This is all experience, not just with SAE

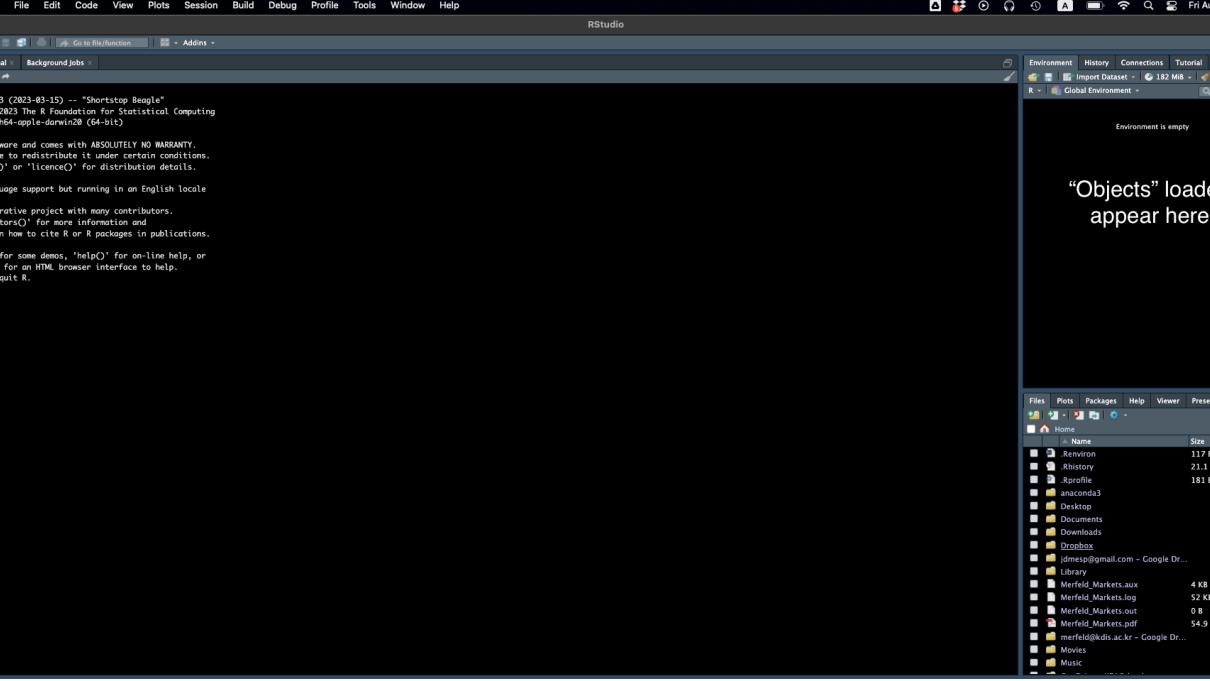
Some things to note

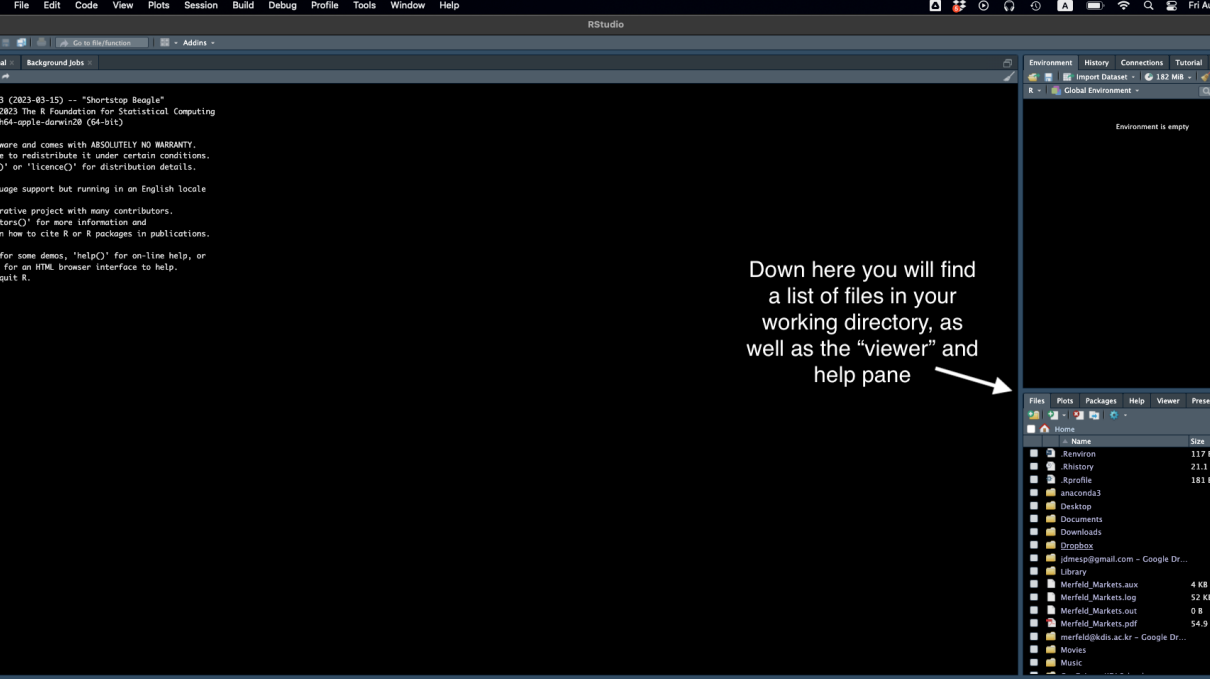
- We will be using **RStudio** throughout the workshops
  - There are other options you are welcome to use (VS Code is the most common alternative)
- Two general “data cleaning” pipelines:
  - The **tidyverse**
  - The **data.table** package (which builds on base R)
- We will be using the tidyverse

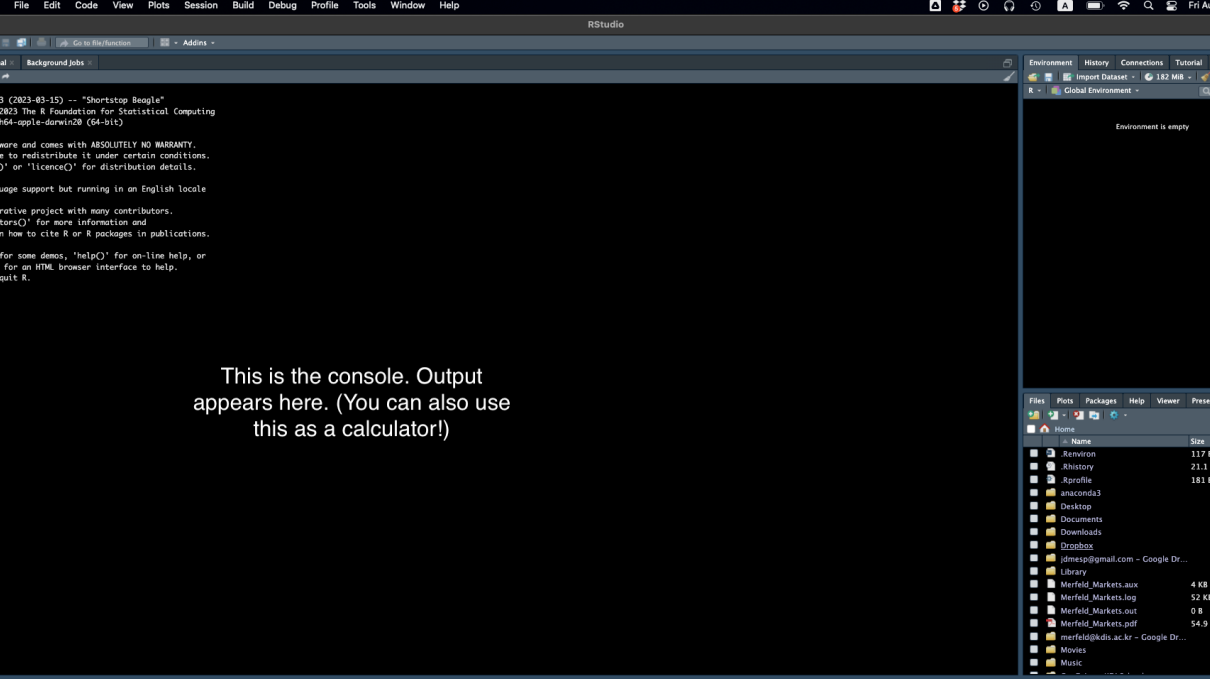
Getting started with RStudio

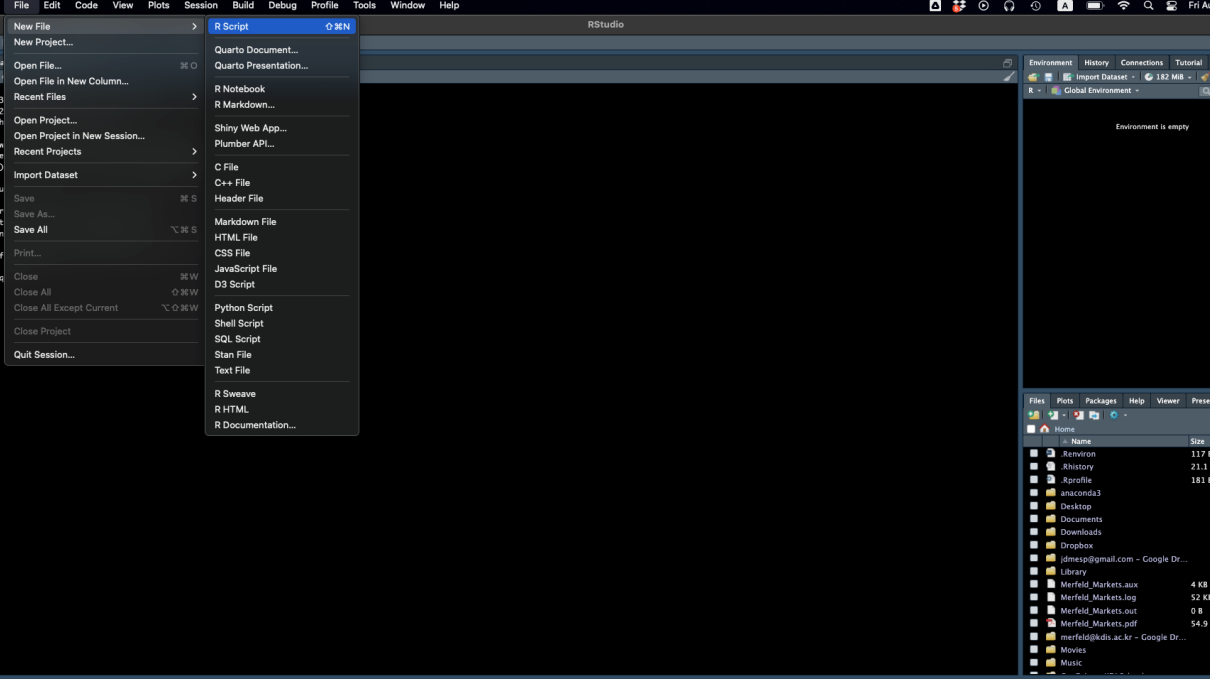
- Let's start by looking at the layout of RStudio.
- For those of you with ample R experience, nothing here will be new!

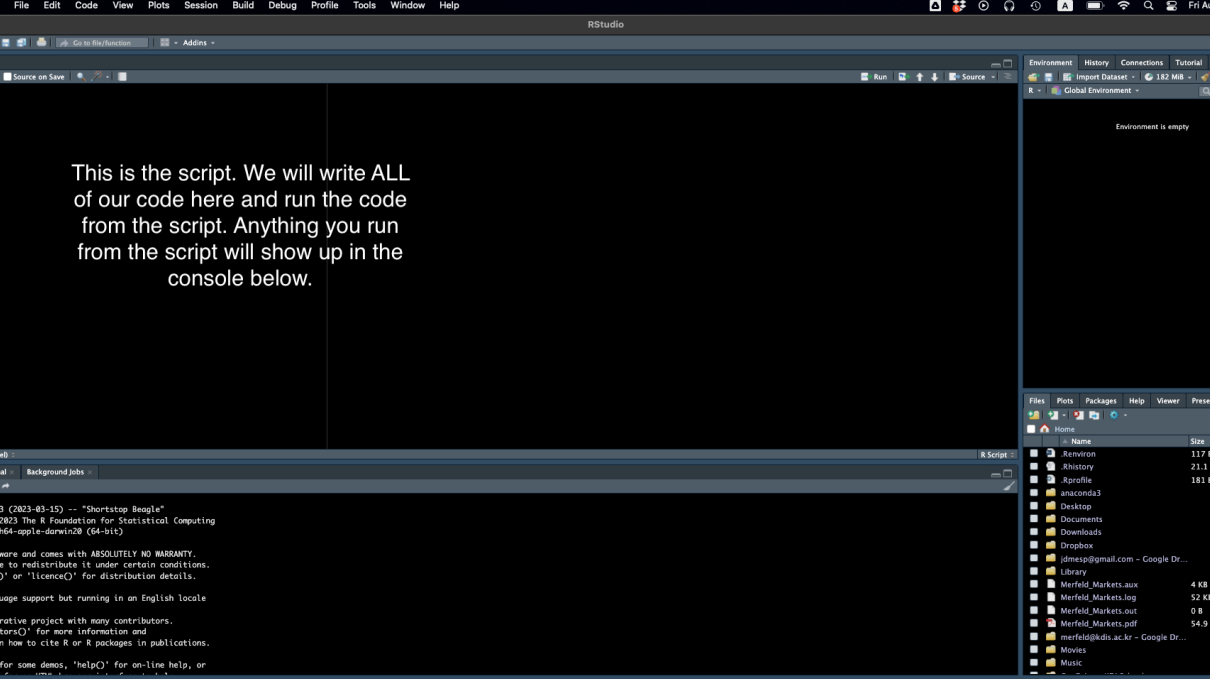

File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Go to file/function
Addins
Background Jobs
3 (2023-03-15) -- "Shortstop Beagle"
2023 The R Foundation for Statistical Computing
n64-apple-darwin20 (64-bit)
ware and comes with ABSOLUTELY NO WARRANTY.
e to redistribute it under certain conditions.
)' or 'licence()' for distribution details.
uage support but running in an English locale
rative project with many contributors.
tors()' for more information and
n how to cite R or R packages in publications.
for some demos, 'help()' for on-line help, or
for an HTML browser interface to help.
quit R.
Environment
History
Connections
Tutorial
Import Dataset
182 MiB
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Home
Name
Size
.Renviron
.Rhistory
.Rprofile
anaconda3
Desktop
Documents
Downloads
Dropbox
Library
Merfeld_Markets.aux
4 KB
Merfeld_Markets.log
Merfeld_Markets.out
0 B
Merfeld_Markets.pdf
Movies
Music

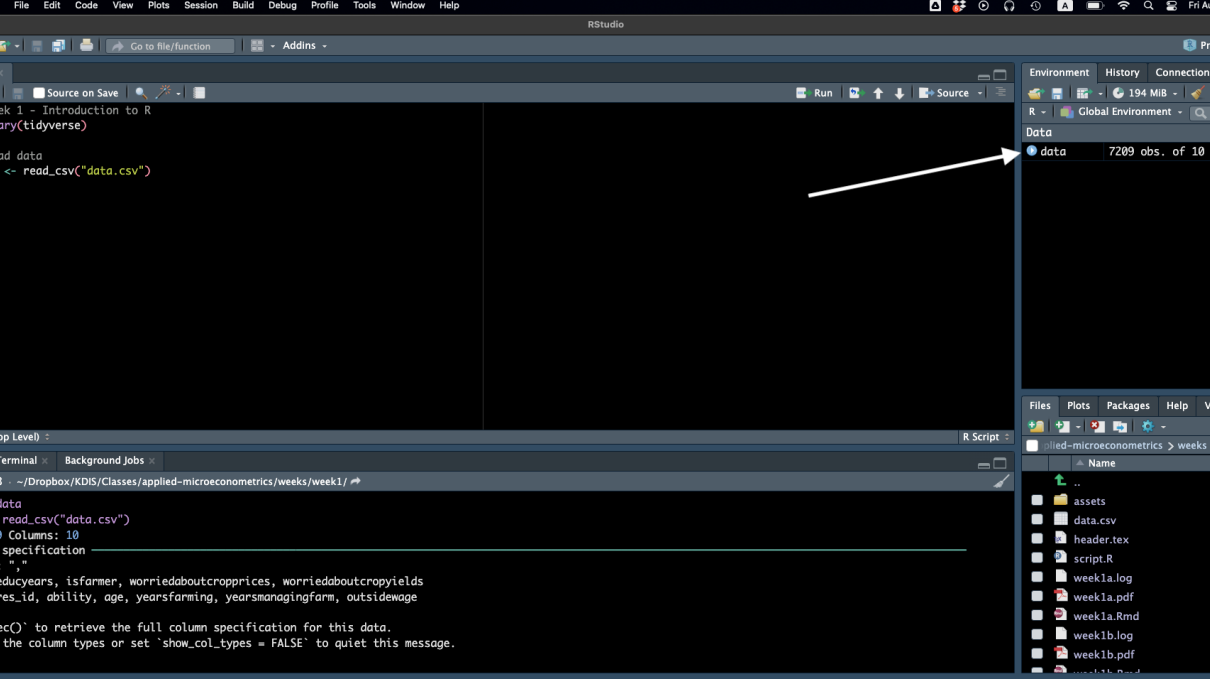

File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Addins
Background Jobs
3 (2023-03-15) -- "Shortstop Beagle"
2023 The R Foundation for Statistical Computing
n64-apple-darwin20 (64-bit)
ware and comes with ABSOLUTELY NO WARRANTY.
e to redistribute it under certain conditions.
)' or 'licence()' for distribution details.
uage support but running in an English locale
rative project with many contributors.
tors()' for more information and
n how to cite R or R packages in publications.
for some demos, 'help()' for on-line help, or
for an HTML browser interface to help.
quit R.
Environment
History
Connections
Tutorial
Import Dataset
182 MiB
Global Environment
Environment is empty
"Objects" loade
appear here
Files
Plots
Packages
Help
Viewer
Home
Name
Size
.Renviron
.Rhistory
.Rprofile
anaconda3
Desktop
Documents
Downloads
Dropbox
Library
Merfeld_Markets.aux
4 KB
Merfeld_Markets.log
Merfeld_Markets.out
0 B
Merfeld_Markets.pdf
Movies
Music

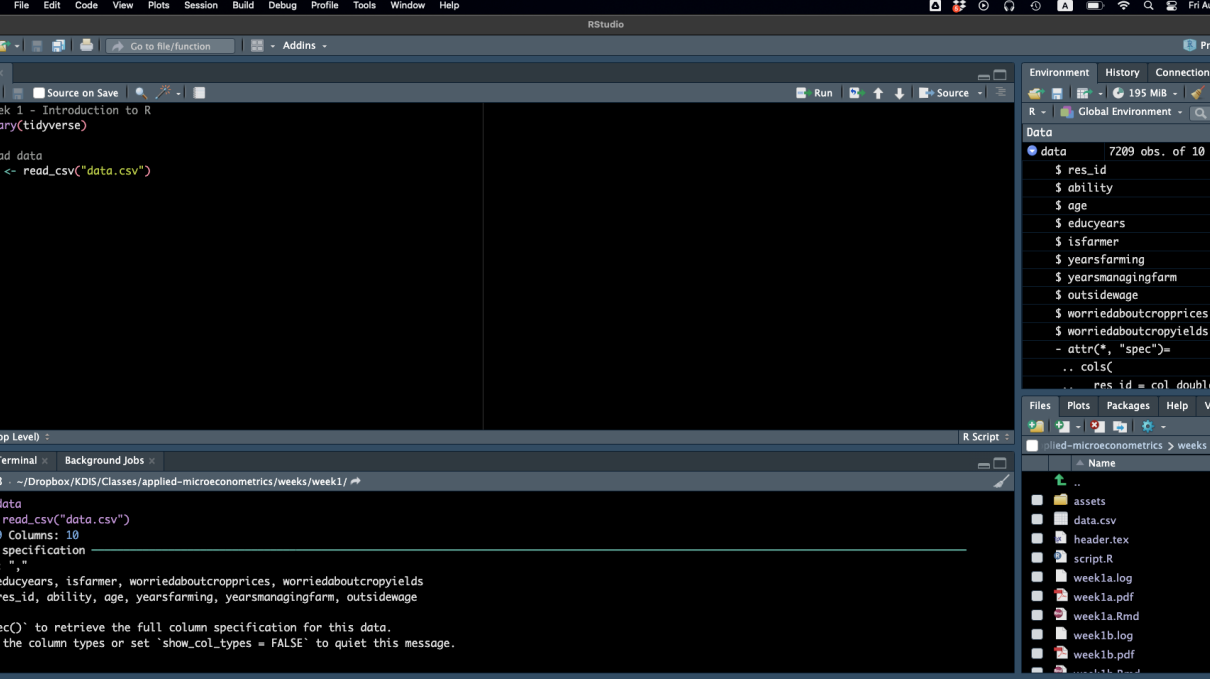

File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Addins
Background Jobs
3 (2023-03-15) -- "Shortstop Beagle"
2023 The R Foundation for Statistical Computing
n64-apple-darwin20 (64-bit)
ware and comes with ABSOLUTELY NO WARRANTY.
e to redistribute it under certain conditions.
)' or 'licence()' for distribution details.
uage support but running in an English locale
rative project with many contributors.
tors()' for more information and
n how to cite R or R packages in publications.
for some demos, 'help()' for on-line help, or
for an HTML browser interface to help.
quit R.
Down here you will find a list of files in your working directory, as well as the “viewer” and help pane
Environment
History
Connections
Tutorial
Import Dataset
182 MiB
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Home
Name
Size
.Renviron
.Rhistory
.Rprofile
anaconda3
Desktop
Documents
Downloads
Dropbox
Library
Merfeld_Markets.aux
4 KB
Merfeld_Markets.log
Merfeld_Markets.out
0 B
Merfeld_Markets.pdf
Movies
Music


File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Go to file/function
Addins
Background Jobs
3 (2023-03-15) -- "Shortstop Beagle"
2023 The R Foundation for Statistical Computing
h64-apple-darwin20 (64-bit)
ware and comes with ABSOLUTELY NO WARRANTY.
e to redistribute it under certain conditions.
)' or 'licence()' for distribution details.
uage support but running in an English locale
rative project with many contributors.
tors()' for more information and
n how to cite R or R packages in publications.
for some demos, 'help()' for on-line help, or
for an HTML browser interface to help.
quit R.
This is the console. Output appears here. (You can also use this as a calculator!)
Environment
History
Connections
Tutorial
Import Dataset
182 MiB
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Home
Name
Size
.Renviron
.Rhistory
.Rprofile
anaconda3
Desktop
Documents
Downloads
Dropbox
Library
Merfeld_Markets.aux
4 KB
Merfeld_Markets.log
Merfeld_Markets.out
0 B
Merfeld_Markets.pdf
Movies
Music


File Edit Code View Plots Session Build Debug Profile Tools Window Help
Fri Au
RStudio
New File
New Project...
Open File... ⌘O
Open File in New Column...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save ⌘S
Save As...
Save All ⌥⌘S
Print...
Close ⌘W
Close All ⇧⌘W
Close All Except Current ⌥⇧⌘W
Close Project
Quit Session...
R Script ⇧⌘N
Quarto Document...
Quarto Presentation...
R Notebook
R Markdown...
Shiny Web App...
Plumber API...
C File
C++ File
Header File
Markdown File
HTML File
CSS File
JavaScript File
D3 Script
Python Script
Shell Script
SQL Script
Stan File
Text File
R Sweave
R HTML
R Documentation...
Environment History Connections Tutorial
Import Dataset 182 MiB
R Global Environment
Environment is empty
Files Plots Packages Help Viewer Prese
Home
Name Size
.Renviron 117 B
.Rhistory 21.1
.Rprofile 181 B
anaconda3
Desktop
Documents
Downloads
Dropbox
jdmasp@gmail.com - Google Dr...
Library
Merfeld_Markets.aux 4 KB
Merfeld_Markets.log 52 KB
Merfeld_Markets.out 0 B
Merfeld_Markets.pdf 54.9
merfeld@kdis.ac.kr - Google Dr...
Movies
Music


File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Go to file/function
Addins
Source on Save
Run
Source
This is the script. We will write ALL of our code here and run the code from the script. Anything you run from the script will show up in the console below.
Environment History Connections Tutorial
Import Dataset
182 MiB
R
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Home
Name
Size
.Renviron
.Rhistory
.Rprofile
anaconda3
Desktop
Documents
Downloads
Dropbox
Library
Merfeld_Markets.aux
4 KB
Merfeld_Markets.log
Merfeld_Markets.out
Merfeld_Markets.pdf
Movies
Music
R Script
Background Jobs
(2023-03-11) -- "Shortstop Beagle"
2023 The R Foundation for Statistical Computing
64-apple-darwin20 (64-bit)
are and comes with ABSOLUTELY NO WARRANTY.
to redistribute it under certain conditions.
)' or 'licence()' for distribution details.
uage support but running in an English locale
rative project with many contributors.
tors()' for more information and
how to cite R or R packages in publications.
for some demos, 'help()' for on-line help, or


File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Go to file/function
Addins
Source on Save
Run
Source
ek 1 - Introduction to R
ary(tidyverse)
ad data
<- read_csv("data.csv")
Environment
History
Connection
194 MiB
R
Global Environment
Data
data
7209 obs. of 10
op Level)
R Script
Terminal
Background Jobs
~/Dropbox/KDIS/Classes/applied-microeconometrics/weeks/week1/
data
read_csv("data.csv")
Columns: 10
specification
","
educyears, isfarmer, worriedaboutcropprices, worriedaboutcropyields
res_id, ability, age, yearsfarming, yearsmanagingfarm, outsidewage
ec()` to retrieve the full column specification for this data.
the column types or set `show_col_types = FALSE` to quiet this message.
Files
Plots
Packages
Help
plied-microeconometrics > weeks
Name
..
assets
data.csv
header.tex
script.R
week1a.log
week1a.pdf
week1a.Rmd
week1b.log
week1b.pdf


File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Go to file/function
Addins
Source on Save
Run
Source
ek 1 - Introduction to R
ary(tidyverse)
ad data
<- read_csv("data.csv")
R Script
Terminal
Background Jobs
~/Dropbox/KDIS/Classes/applied-microeconometrics/weeks/week1/
data
read_csv("data.csv")
Columns: 10
specification
","
educyears, isfarmer, worriedaboutcropprices, worriedaboutcropyields
res_id, ability, age, yearsfarming, yearsmanagingfarm, outsidewage
ec()` to retrieve the full column specification for this data.
the column types or set `show_col_types = FALSE` to quiet this message.
Environment
History
Connection
195 MiB
R
Global Environment
Data
data
7209 obs. of 10
$ res_id
$ ability
$ age
$ educyears
$ isfarmer
$ yearsfarming
$ yearsmanagingfarm
$ outsidewage
$ worriedaboutcropprices
$ worriedaboutcropyields
- attr(*, "spec")=
.. cols(
res_id = col_doubl
Files
Plots
Packages
Help
plied-microeconometrics > weeks
Name
..
assets
data.csv
header.tex
script.R
week1a.log
week1a.pdf
week1a.Rmd
week1b.log
week1b.pdf

Why don’t you all give it a try

- Create a script in RStudio
- Save that script in a specific place (folder) on your computer
  - Make sure to keep track of where you save it!
  - I create a folder for each specific project I work on
  - e.g. you could create “Nairobi Workshops” and save the script as “day1.R”

First things first: the working directory

- The working directory is the folder that R is currently working in
  - This is where R will look for files
  - This is where R will save files
  - This is where R will create files
- You can always write out an entire file path, but this is tedious
  - More importantly, it makes your code less reproducible since the path is specific to YOUR computer

First things first: the working directory

- One nice thing about R is that the working directory will automatically be where you open the script from
  - Let's try this. Save your script to a folder on your computer, then open the script from that folder.
  - Let's see if it worked!

▼ Code

```
1 getwd() # this command will show you your current working directory
```

```
[1] "/Users/Josh/Dropbox/Papers/UN-SAE/workshops/africa/nairobiworkshops"
```

First things first: the working directory

- You can also set the working directory in RStudio
  - Session > Set Working Directory > Choose Directory (or Source File Location)
  - Give it a try and let's see if it worked!

▼ Code

```
1 getwd() # this command will show you your current working directory
```

```
[1] "/Users/Josh/Dropbox/Papers/UN-SAE/workshops/africa/nairobiworkshops"
```

Always use the same working directory!

- Make sure to always set the working directory to the same location when working in the same script!

- This will avoid problems later
  - It also makes your code more reproducible (e.g. if a colleague wants to run it, you just send the entire folder and it works with no changes)

R packages

- R is a language that is built on packages
  - Packages are collections of functions that do specific things
  - R comes with a set of “base” packages that are installed automatically
- We are going to use one package consistently, called the “tidyverse”
  - This consists of a set of packages that are designed to work together, with data cleaning in mind

R packages

The one exception to always using a script? I install packages in the CONSOLE. You can install packages like this:

▼ Code

```
1 install.packages("tidyverse") # this will install the tidyverse package. Note the quotes!
```

- You only need to install a package once on your computer.

R packages

The first thing you'll do in your script is load packages. You do it like this:

▼ Code

```
'''
This script is part of the Nairobi Workshop on SAE.
Date: 26 August 2024 (written earlier!)
Author: Josh Merfeld
'''
# Load packages (libraries)
library(tidyverse)
```

- Note that the first part is a comment I've added to the script.
  - I make a lot of comments!

Loading data

- Let's start by loading some data.
  - This is survey data from Tanzania (NPS 5, 2020)
  - It is a list of all individuals in all households in the survey
- Let's load the csv version. Tidyverse has a command for this: `read_csv()`

▼ Code

```
# Load packages (libraries)
library(tidyverse)
# load data
df <- read_csv("dayldata/tanzanialsms.csv")
```

Looking at the data

- `glimpse` is an easy way to look at the data:

▼ Code

```
# Load packages (libraries)
library(tidyverse)
# load data
df <- read_csv("dayldata/tanzanialsms.csv")
glimpse(df)
```

```
Rows: 23,592
Columns: 46
$ interview__key <chr> "39-26-37-98", "39-26-37-98", "39-26-37-98", "39-26-37-98", "04-06-65-04", "97-90-78-65", "97-90-78-65", "
$ y5_hhid        <chr> "1000-001-01", "1000-001-01", "1000-001-01", "1000-001-01", "1000-001-02", "1000-001-03", "1000-001-03", "
$ y4_hhid        <chr> "1000-001", "1000-001", "1000-001", "1000-001", "1000-001", "1000-001", "1000-001", "1000-001", "1000-001"
$ indidy5        <dbl> 1, 3, 5, 7, 1, 1, 2, 4, 6, 7, 8, 1, 2, 4, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 1, 2, 3, 1, 2, 3, 4, 5, 6, 1, 2
$ hh_b01         <chr> "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CON
$ hh_b02         <chr> "male", "female", "male", "male", "male", "male", "female", "female", "male", "female", "female", "male",
$ hh_b03_1       <chr> "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CON
$ hh_b03_2       <chr> "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CON
$ hh_b04         <dbl> 74, 44, 35, 9, 47, 36, 42, 21, 1, 1, 6, 32, 32, 2, 48, 45, 23, 21, 17, 14, 8, 2, 79, 71, 34, 63, 41, 22, 4
$ hh_b05         <chr> "head", "SON/DAUGHTER", "SON/DAUGHTER", "grandchild", "head", "head", "spouse", "SON/DAUGHTER", "SON/DAUGH
$ hh_b06         <chr> "1", "3", "5", "NOT PREVIOUSLY PRESENT", "2", "4", "NOT PREVIOUSLY PRESENT", "NOT PREVIOUSLY PRESENT", "NO
$ hh_b07         <chr> "yes", "yes", "no", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "no", "yes", "yes", "yes", "yes", "ye
$ hh_b08         <dbl> 31, 31, 0, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 0, 31, 31, 31, 31, 31, 31, 31, 31, 31,
$ hh_b09_1       <chr> "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "
$ hh_b0a         <chr> "yes", "yes", "no", "no", "yes", "yes", "yes", "yes", "yes", "no", "yes", "no", "yes", "no", "yes", "yes",
```

```
 18 $ hh_b0b         <dbl> NA, NA, 3, 3, NA, NA, NA, NA, NA, 1, NA, 2, NA, 2, NA, NA, 1, NA, NA, NA, NA, NA, NA, NA, NA, NA, 3, NA, N
 19 $ hh_b10         <dbl> 0, 0, 4, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, NA, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

Data is not always in .csv files

- Sometimes files come in other formats
- I have uploaded a Stata file, as well (tanzanialsms.dta)
  - To read Stata files (common with survey data), we need a different package, called `haven`
  - After we install it, you can load it using `read_dta()`
- Give it a try. Install the package and then load the data.

▸ Code

But let's use the .csv file for now

▼ Code

```
# Load packages (libraries)
library(tidyverse)
# load data
df <- read_csv("dayldata/tanzanialsms.csv")
glimpse(df)
```

```
Rows: 23,592
Columns: 46
$ interview__key <chr> "39-26-37-98", "39-26-37-98", "39-26-37-98", "39-26-37-98", "04-06-65-04", "97-90-78-65", "97-90-78-65", "
$ y5_hhid        <chr> "1000-001-01", "1000-001-01", "1000-001-01", "1000-001-01", "1000-001-02", "1000-001-03", "1000-001-03", "
$ y4_hhid        <chr> "1000-001", "1000-001", "1000-001", "1000-001", "1000-001", "1000-001", "1000-001", "1000-001", "1000-001"
$ indidy5        <dbl> 1, 3, 5, 7, 1, 1, 2, 4, 6, 7, 8, 1, 2, 4, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 1, 2, 3, 1, 2, 3, 4, 5, 6, 1, 2
$ hh_b01         <chr> "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CON
$ hh_b02         <chr> "male", "female", "male", "male", "male", "male", "female", "female", "male", "female", "female", "male",
$ hh_b03_1       <chr> "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CON
$ hh_b03_2       <chr> "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CONFIDENTIAL**", "**CON
$ hh_b04         <dbl> 74, 44, 35, 9, 47, 36, 42, 21, 1, 1, 6, 32, 32, 2, 48, 45, 23, 21, 17, 14, 8, 2, 79, 71, 34, 63, 41, 22, 4
$ hh_b05         <chr> "head", "SON/DAUGHTER", "SON/DAUGHTER", "grandchild", "head", "head", "spouse", "SON/DAUGHTER", "SON/DAUGH
$ hh_b06         <chr> "1", "3", "5", "NOT PREVIOUSLY PRESENT", "2", "4", "NOT PREVIOUSLY PRESENT", "NOT PREVIOUSLY PRESENT", "NO
$ hh_b07         <chr> "yes", "yes", "no", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "no", "yes", "yes", "yes", "yes", "ye
$ hh_b08         <dbl> 31, 31, 0, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 0, 31, 31, 31, 31, 31, 31, 31, 31,
$ hh_b09_1       <chr> "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "yes", "
$ hh_b0a         <chr> "yes", "yes", "no", "no", "yes", "yes", "yes", "yes", "yes", "no", "yes", "no", "yes", "no", "yes", "yes",
$ hh_b0b         <dbl> NA, NA, 3, 3, NA, NA, NA, NA, NA, 1, NA, 2, NA, 2, NA, NA, 1, NA, NA, NA, NA, NA, NA, NA, NA, NA, 3, NA, N
$ hh_b10         <dbl> 0, 0, 4, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, NA, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

Objects in memory

- The data frame is a matrix
  - Each row is an observation and each column is a variables
  - You can see it in the “environment” pane of RStudio
- We can also see the names of the columns like this:

▼ Code

```
1 colnames(df)
```

```
1 [1] "interview__key" "y5_hhid"        "y4_hhid"        "indidy5"        "hh_b01"         "hh_b02"         "hh_b03_1"       "hh_b
```

Calling variables in R

- Some of you might be used to Stata

- One big difference between the two is that Stata generally only has one data frame in memory at a time
  - This means that you can call a variable *without referencing the data frame*

- In R, if you want to look at a variable, you have to tell R which data frame it is in
  - This is done with the `$` operator
  - For example, if I want to look at the variable “age” in the data frame “data”, I would write `data$age`
  - In our dataset, the age variable is `hh_b04`

Summary stats for `hh_b04` (age)

- There are two common ways to quickly look at a variable:
  - `summary()`: gives you mean/median, and a few more
  - `table()`: gives you a frequency table

▼ Code

```
summary(df$hh_b04)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    8.00   18.00   22.78   33.00   95.00
```

Summary stats for `hh_b04` (age)

- There are two common ways to quickly look at a variable:
  - `summary()`: gives you mean/median, and a few more
  - `table()`: gives you a frequency table

▼ Code

```
1 table(df$hh_b04)
```

```
1
2   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
3 695 750 721 716 723 597 738 717 623 647 627 619 574 582 571 559 592 546 518 438 519 500 403 397 384 425 379 352 358 323 277 299 2
```

Summary stats for everything

- If you don't specify a variable, it will give you information for the ENTIRE dataframe!

▼ Code

```
summary(df)
```

```
 interview__key       y5_hhid            y4_hhid             indidy5          hh_b01             hh_b02             hh_b03_1        
 Length:23592       Length:23592       Length:23592       Min.   : 1.000   Length:23592       Length:23592       Length:23592       
 Class :character   Class :character   Class :character   1st Qu.: 2.000   Class :character   Class :character   Class :character   
 Mode  :character   Mode  :character   Mode  :character   Median : 4.000   Mode  :character   Mode  :character   Mode  :character   
                                                          Mean   : 4.493                                                            
                                                          3rd Qu.: 6.000                                                            
                                                          Max.   :42.000                                                            
                                                                                                                                    
```

Variable types

- The most common are numeric, character, and logical
  - Numeric variables are numbers
    - For example, the variable `hh_b04` is a numeric variable
    - Missing value: `NA`
  - Character variables are text
    - For example, the variable `y5_hhid` is a character variable
    - Missing value: `""`
  - Logical variables are TRUE/FALSE
    - We don't have any examples here (at least not yet)
    - Missing value: `NA`

Pipes

- One of the most useful things in R is the pipe operator (`|>` or `%>%`)
  - This is part of the tidyverse package
  - It allows you to chain commands together
  - It makes your code much easier to read
  - It makes your code much easier to write

- Let's use it to clean some variables

Pipes example

- I'm going to create some new variables. What do you think this does?

▼ Code

```
df <- df |>
  rename(gender = hh_b02, age = hh_b04) |>
  mutate(boy = gender=="male" & age<15,
    girl = gender=="female" & age<15,
    prime_aged_male = gender=="male" & age>=15 & age<=64,
    prime_aged_female = gender=="female" & age>=15 & age<=64,
    elderly_male = gender=="male" & age>64,
    elderly_female = gender=="female" & age>64)
```

Pipes example

- I'm going to create some new variables. What do you think this does?

▼ Code

```
1 df <- df |>
2   rename(gender = hh_b02, age = hh_b04) |>
3   mutate(boy = gender=="male" & age<15,
4     girl = gender=="female" & age<15,
5     prime_aged_male = gender=="male" & age>=15 & age<=64,
6     prime_aged_female = gender=="female" & age>=15 & age<=64,
7     elderly_male = gender=="male" & age>64,
8     elderly_female = gender=="female" & age>64)
```

① Renames the variables `hh_b02` and `hh_b04` to `gender` and `age`, respectively (line 2)
② Creates new variables based on the conditions given (lines 3-8)

Pipes example

- Let's look at the resulting variables:

▼ Code

```
summary(df[,c("boy", "girl", "prime_aged_male", "prime_aged_female", "elderly_male", "elderly_female")])
```

```
     boy             girl          prime_aged_male prime_aged_female elderly_male    elderly_female
 Mode :logical   Mode :logical   Mode :logical   Mode :logical      Mode :logical   Mode :logical
 FALSE:18687     FALSE:18597     FALSE:17396     FALSE:16988        FALSE:23211     FALSE:23081
 TRUE :4905      TRUE :4995      TRUE :6196      TRUE :6604         TRUE :381       TRUE :511
```

- What did I do here?
  - I summarized `df` based on the names of COLUMNS (variables)
    - I did this by creating a vector of column names (as characters)
  - The resulting variables are logical, or True/False
    - Why?

Let’s do some more cleaning

- What are some other common data cleaning tasks?
- Some common things:
  - Renaming variables
  - Creating new variables based on existing variables
    - Doing this based on groups
  - Replace values (e.g. replacing missings)
    - i.e. recoding variables

Let’s do some more cleaning

- What are some other common data cleaning tasks?
- Some common things:
  - **Renaming variables**
  - **Creating new variables based on existing variables**
    - Doing this based on groups
  - Replace values (e.g. replacing missings)
    - i.e. recoding variables

What do we want to do?

- To do more, we need to look at the survey questionnaire!
  - The data I have uploaded is from Section B
  - You can find the survey questionnaire in the data folder for day 1
- Here's the data:

```
# A tibble: 23,592 × 46
   interview__key y5_hhid     y4_hhid  indidy5 hh_b01           hh_b02 hh_b03_1         hh_b03_2         hh_b04 hh_b05       hh_b
   <chr>          <chr>       <chr>       <dbl> <chr>            <chr>  <chr>            <chr>             <dbl> <chr>        <chr
 1 39-26-37-98    1000-001-01 1000-001        1 **CONFIDENTIAL** male   **CONFIDENTIAL** **CONFIDENTIAL**     74 head         1
 2 39-26-37-98    1000-001-01 1000-001        3 **CONFIDENTIAL** female **CONFIDENTIAL** **CONFIDENTIAL**     44 SON/DAUGHTER 3
 3 39-26-37-98    1000-001-01 1000-001        5 **CONFIDENTIAL** male   **CONFIDENTIAL** **CONFIDENTIAL**     35 SON/DAUGHTER 5
 4 39-26-37-98    1000-001-01 1000-001        7 **CONFIDENTIAL** male   **CONFIDENTIAL** **CONFIDENTIAL**      9 grandchild   NOT
 5 04-06-65-04    1000-001-02 1000-001        1 **CONFIDENTIAL** male   **CONFIDENTIAL** **CONFIDENTIAL**     47 head         2
 6 97-90-78-65    1000-001-03 1000-001        1 **CONFIDENTIAL** male   **CONFIDENTIAL** **CONFIDENTIAL**     36 head         4
 7 97-90-78-65    1000-001-03 1000-001        2 **CONFIDENTIAL** female **CONFIDENTIAL** **CONFIDENTIAL**     42 spouse       NOT
 8 97-90-78-65    1000-001-03 1000-001        4 **CONFIDENTIAL** female **CONFIDENTIAL** **CONFIDENTIAL**     21 SON/DAUGHTER NOT
 9 97-90-78-65    1000-001-03 1000-001        6 **CONFIDENTIAL** male   **CONFIDENTIAL** **CONFIDENTIAL**      1 SON/DAUGHTER NOT
10 97-90-78-65    1000-001-03 1000-001        7 **CONFIDENTIAL** female **CONFIDENTIAL** **CONFIDENTIAL**      1 SON/DAUGHTER NOT
# ℹ 23,582 more rows
```

SECTION B: HOUSEHOLD MEMBER ROSTER

IN ORDER TO MAKE A COMPREHENSIVE LIST OF HOUSEHOLD MEMBERS, USE THE FOLLOWING PROBE QUESTIONS:

FIRST, ASK NAMES OF ALL THE MEMBERS OF YOUR IMMEDIATE (NUCLEAR) FAMILY WHO NORMALLY LIVE AND EAT THEIR MEALS TOGETHER HERE.
WRITE DOWN NAMES, SEX, AND RELATIONSHIP TO HOUSEHOLD HEAD
FILL IN QUESTIONS 1 TO 6
THEN, ASK NAMES OF ANY OTHER PERSONS RELATED TO YOU OR OTHER HOUSEHOLD MEMBERS WHO NORMALLY LIVE AND EAT THEIR MEALS TOGETHER HERE.
FILL IN QUESTIONS 1 TO 6
ALSO ASK OTHER PERSONS NOT HERE NOW WHO NORMALLY LIVE AND EAT THEIR MEALS HERE? FOR EXAMPLE, HOUSEHOLD MEMBERS STUDYING ELSEWHERE OR TRAVELING.
FILL IN QUESTIONS 1 TO 6.
THEN, ASK NAMES OF ANY OTHER PERSONS NOT RELATED TO YOU OR OTHER HOUSEHOLD MEMBERS, BUT WHO NORMALLY LIVE AND EAT THEIR MEALS TOGETHER HERE, SUCH AS LIVE-IN SERVANTS.
FILL IN QUESTIONS 1 TO 6
IF MORE THAN 12 INDIVIDUALS, USE SECOND QUESTIONNAIRE. MAKE SURE TO MARK BOX ON FIRST PAGE OF BOTH QUESTIONNAIRES.

Q.9 EXCEPTIONS
INFANTS LESS THAN 3 MONTHS
NEW HOUSEHOLD MEMBERS
BOARDING SCHOOL STUDENTS

INDIVIDUAL ID	1. NAME LIST HOUSEHOLD HEAD ON LINE 1. MAKE A COMPLETE LIST OF ALL INDIVIDUALS WHO NORMALLY LIVE AND EAT THEIR MEALS TOGETHER IN THIS HOUSEHOLD, STARTING WITH THE HEAD OF HOUSEHOLD. (CONFIRM THAT HOUSEHOLD HEAD HERE IS SAME AS HOUSEHOLD HEAD LISTED	2. Sex M..1 F..2	3. In what month and year was [NAME] born? **PUT "99" IF DON'T KNOW** YEAR	MONTH	4. How old is [NAME]? IF RESPONDENT DOESN'T KNOW, USE YEAR OF BIRTH TO CALCULATE AGE. CHECK THAT AGE IN QUESTION 4 AND YEAR OF BIRTH IN QUESTION 3 ARE CONSISTENT. YEARS	5. What is [NAME]'s relationship to the head of household? HEAD.....1 SPOUSE.....2 SON/DAUGHTER...3 STEP SON / DAUGHTER.....4 SISTER/BROTHER.5 GRANDCHILD.....6 FATHER/MOTHER..7 OTHER RELATIVE (SPECIFY).....8 LIVE-IN SERVANT.....9 OTHER NON-RELATIVES (SPECIFY)...10	6. **IF THIS MEMBER WAS PRESENT AT LAST SURVEY, ENTER Y4 ROSTER ID NUMBER FROM TRACKING FORM** **ELSE, ENTER 99** NPS Y3 ROSTER ID	7. Did [NAME] eat meals in this household in the last 7 days? YES..1 NO...2	8. For how many days in the last month was [NAME] present? DAYS	9. For the last 12 months has [NAME] stayed in this household for 3 months or more? **CROSS OUT ID CODE IN THE FLAP AND DO NOT ADMINISTER OTHER SECTIONS FOR INDIVIDUALS WITH CODE** YES...1 NO....2	INDIVIDUAL ID
1											1
2											2
3											3
4											4
5											5
6											6
7											7
8											8
9											9
10											10
11											11
12											12

Let’s do some cleaning

- Two things:
  - Calculate the education of the head’s mother
  - Collapse the data to the HOUSEHOLD level, with the following variables:
    - Number of boys/girls
    - Number of prime-aged males/females
    - Number of elderly males/females
    - The head’s gender

- How do we do this?

Let’s do some cleaning

- Let’s start by loading the data again
  - You might want to try this in a new script so you have a self-contained example of data cleaning
  - Go ahead and load libraries and read the .csv (not .dta) data:

▸ Code

Back to our task

- Calculate the education of the head's mother
- Collapse the data to the HOUSEHOLD level, with the following variables:
  - Number of boys/girls
  - Number of prime-aged males/females
  - Number of elderly males/females
  - The head's gender

- What are the variables we need? (look at the questionnaire)
  - `y5_hhid` (household identifier)
  - `hh_b02` (gender)
  - `hh_b04` (age)
  - `hh_b05` (relationship to head)

How are we going to do this?

- Create the new variables
  - Boys, girls, prime-aged males, prime-aged females, elderly males, elderly females
  - The head's gender

- Collapse the data to the household level
  - We will want to sum the count variables
  - We will want to take the highest value for the head's gender (since there is only one)

First, rename some variables

▼ Code

```
dfhh <- df |>
  rename(gender = hh_b02, age = hh_b04, relationship = hh_b05) |>
  select(y5_hhid, gender, age, relationship)
head(dfhh)
unique(dfhh$relationship)
```

① Renames the variables `hh_b02` and `hh_b04` to `gender` and `age`, respectively, as well as relationship to the head

② Just keep the variables we want

③ Shows us the first few lines

④ Unique values for `relationship`

```
# A tibble: 6 × 4
  y5_hhid     gender   age relationship
  <chr>       <chr>  <dbl> <chr>
1 1000-001-01 male      74 head
2 1000-001-01 female    44 SON/DAUGHTER
3 1000-001-01 male      35 SON/DAUGHTER
4 1000-001-01 male       9 grandchild
5 1000-001-02 male      47 head
6 1000-001-03 male      36 head
 [1] "head"                        "SON/DAUGHTER"                "grandchild"                  "spouse"
"OTHER RELATIVE (SPECIFY)"       "OTHER NON- RELATIVES (SPECIFY)" "FATHER/MOTHER"               "SISTER/BROTHER"              "STEP SON
/ DAUGHTER"             "LIVE-IN SERVANT"
```

Now, create new variables (with new `dfhh` object)

▼ Code

```
1 dfhh <- dfhh |>
2   mutate(boy = (gender=="male" & age<15),                                    ①
3     girl = (gender=="female" & age<15),
4     prime_aged_male = (gender=="male" & age>=15 & age<=64),
5     prime_aged_female = (gender=="female" & age>=15 & age<=64),
6     elderly_male = (gender=="male" & age>64),
7     elderly_female = (gender=="female" & age>64),
8     head_male = ifelse(relationship=="head", gender=="male", NA)) |>        ②
9   select(y5_hhid, boy, girl, prime_aged_male, prime_aged_female, elderly_male, elderly_female, head_male)   ③
```

① Creates new variables based on the conditions given (lines 3-8)
② Creates head's mother's education variable based on the condition given (line 9)
③ Just keep the variables we want

Finally, aggregate to household

▼ Code

```
dfhh <- dfhh |>
  group_by(y5_hhid) |>
  summarize(boy = sum(boy, na.rm = TRUE),
    girl = sum(girl, na.rm = TRUE),
    prime_aged_male = sum(prime_aged_male, na.rm = TRUE),
    prime_aged_female = sum(prime_aged_female, na.rm = TRUE),
    elderly_male = sum(elderly_male, na.rm = TRUE),
    elderly_female = sum(elderly_female, na.rm = TRUE),
    head_male = max(head_male, na.rm = TRUE))
summary(dfhh)
```

```
   y5_hhid              boy              girl        prime_aged_male prime_aged_female  elderly_male     elderly_female     head_male
 Length:4709        Min.   : 0.000   Min.   : 0.000   Min.   :0.000   Min.   :0.000     Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
 Class :character   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:1.000   1st Qu.:1.000     1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
 Mode  :character   Median : 1.000   Median : 1.000   Median :1.000   Median :1.000     Median :0.00000   Median :0.0000   Median :1.0000
                    Mean   : 1.042   Mean   : 1.061   Mean   :1.316   Mean   :1.402     Mean   :0.08091   Mean   :0.1085   Mean   :0.7297
                    3rd Qu.: 2.000   3rd Qu.: 2.000   3rd Qu.:2.000   3rd Qu.:2.000     3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:1.0000
                    Max.   :10.000   Max.   :11.000   Max.   :8.000   Max.   :8.000     Max.   :2.00000   Max.   :2.0000   Max.   :1.0000
```

Why is this important for SAE?

- You will use this type of aggregation *all* the time in SAE.
- We often model at an aggregate level
  - Could be the household
  - Could be the district
  - Could be the county
  - Etc.
- So bookmark this code!

Now it’s your turn!

- Select some variables from the dataset
- Create the following household-level dataset:
  - Total household size
  - Proportion of CHILDREN (<15) who are male
  - Proportion of ADULTS (15+) who are female

Visualization with `ggplot2`

What is `ggplot2`?

- `ggplot2` is included in the `tidyverse` package
  - It is a package for creating beautiful graphics
  - Let's look at some of the basics

- Let's start over with our .csv file

▸ Code

Basic syntax for `ggplot2`?

- The basic syntax is as follows:

```
ggplot() + geom_point(data = df, aes(x = xvar, y = yvar))
```

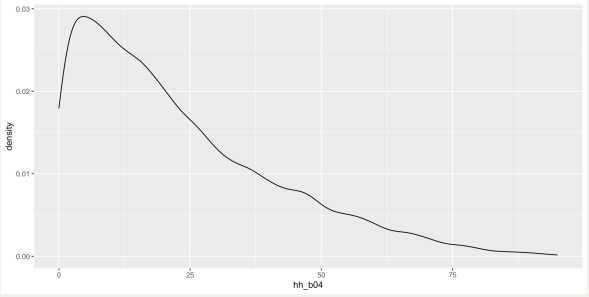
- Note the use of `+` and not `|>`
- `geom_point` can be replaced with all sorts of “geoms”!
  - `geom_line`
  - `geom_bar`
  - `geom_density` (this will have no)

Let's look at an example

- Let's start with one that is simple to understand
  - How about the distribution of age in the dataset?

▼ Code

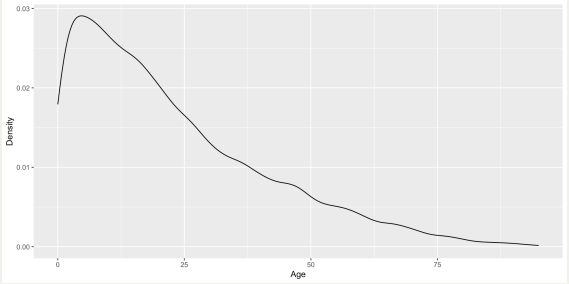
```
ggplot() +
  geom_density(data = df, aes(x = hh_b04))
```



Cleaning it up a bit

▼ Code

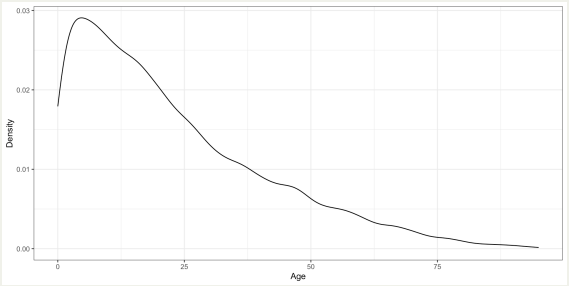
```
ggplot() +
  geom_density(data = df, aes(x = hh_b04)) +
  labs(x = "Age", y = "Density")
```



Cleaning it up even more (my favorite “theme”)

▼ Code

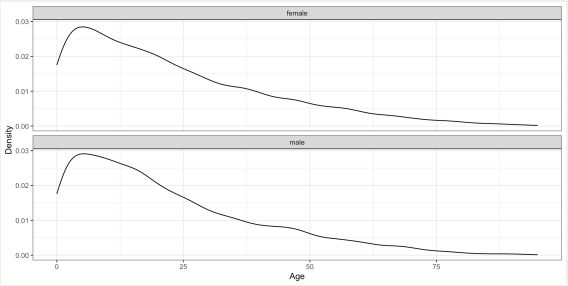
```
ggplot() +
  geom_density(data = df, aes(x = hh_b04)) +
  labs(x = "Age", y = "Density") +
  theme_bw()
```



What if we want to add by gender?

▼ Code

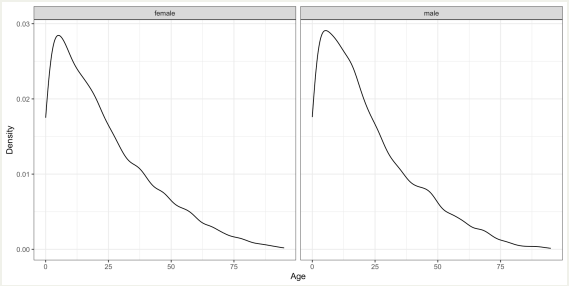
```
ggplot() +
  geom_density(data = df, aes(x = hh_b04)) +
  labs(x = "Age", y = "Density") +
  theme_bw() +
  facet_wrap(~hh_b02, ncol = 1)
```



What if we want to add by gender?

▼ Code

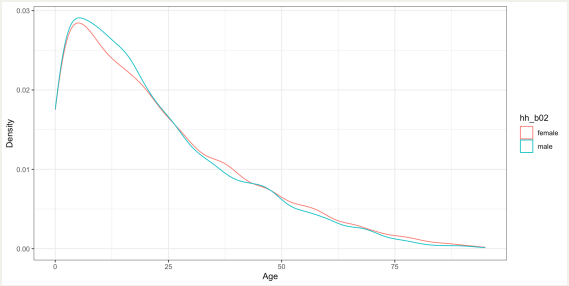
```
ggplot() +
  geom_density(data = df, aes(x = hh_b04)) +
  labs(x = "Age", y = "Density") +
  theme_bw() +
  facet_wrap(~hh_b02, ncol = 2)
```



Putting them on one plot is simple, too!

▼ Code

```
ggplot() +
  geom_density(data = df,
    aes(x = hh_b04, color = hh_b02)) +
  labs(x = "Age", y = "Density") +
  theme_bw()
```

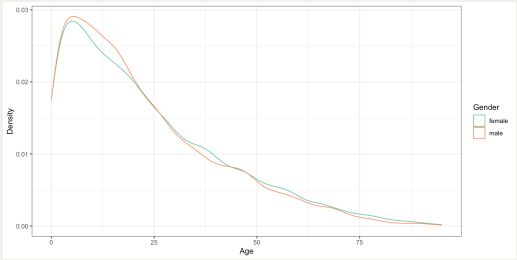


Cleaning up the legend

- Let's use the `scale_color_brewer` function to clean up the legend (title and colors)

▼ Code

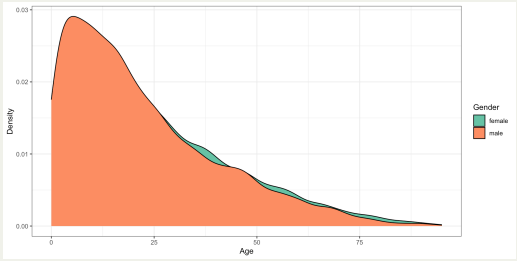
```
ggplot() +
  geom_density(data = df,
    aes(x = hh_b04, color = hh_b02)) +
  scale_color_brewer("Gender", palette = "Set2") +
  labs(x = "Age", y = "Density") +
  theme_bw()
```



Fill instead of color?

▼ Code

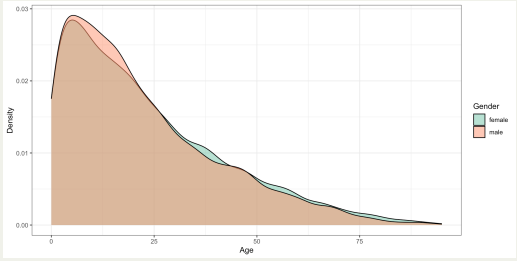
```
ggplot() +
  geom_density(data = df,
    aes(x = hh_b04, fill = hh_b02)) +
  scale_fill_brewer("Gender", palette = "Set2") +
  labs(x = "Age", y = "Density") +
  theme_bw()
```



- Note how hard it is to see both!

▼ Code

```
ggplot() +
  geom_density(data = df,
    aes(x = hh_b04, fill = hh_b02),
    alpha = 0.5) +                    ①
  scale_fill_brewer("Gender", palette = "Set2") +
  labs(x = "Age", y = "Density") +
  theme_bw()
```

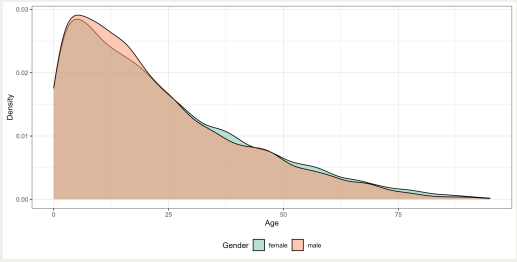


① Change opacity with `alpha`

Finally, changing the location of the legend!

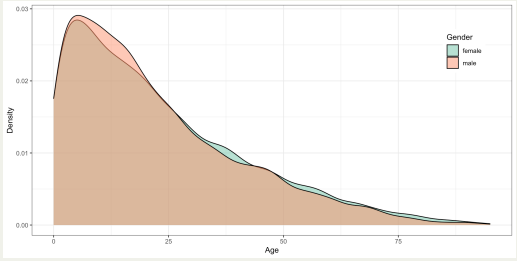
▼ Code

```
ggplot() +
  geom_density(data = df,
    aes(x = hh_b04, fill = hh_b02),
    alpha = 0.5) +
  scale_fill_brewer("Gender", palette = "Set2") +
  labs(x = "Age", y = "Density") +
  theme_bw() +
  theme(legend.position = "bottom")
```



▼ Code

```
ggplot() +
  geom_density(data = df,
    aes(x = hh_b04, fill = hh_b02),
    alpha = 0.5) +
  scale_fill_brewer("Gender", palette = "Set2") +
  labs(x = "Age", y = "Density") +
  theme_bw() +
  theme(legend.position = c(0.9, 0.8))
```

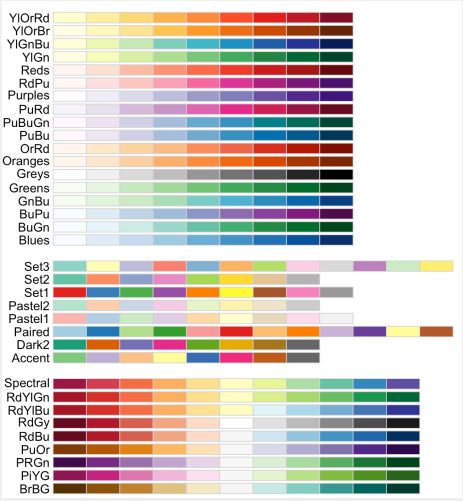


Just FYI, the R Color Brewer palettes

- Here you can see all the color palettes
- Some of these are for continuous variables
- The middle section is mostly for categorical variables (like gender)
- Choose color by changing the PALETTE NAME:

▼ Code

```
ggplot() +
  geom_density(data = df,
    aes(x = hh_b04, fill = hh_b02),
    alpha = 0.5) +
  scale_fill_brewer("Gender", palette = "PALETTE NAME") +
  labs(x = "Age", y = "Density") +
  theme_bw() +
  theme(legend.position = "bottom")
```



Let’s look at another example

- In this example, we’ll also practice aggregating to higher levels (again!)
  - Let’s look at the proportion of people in agriculture and in education BY AGE GROUP
  - We are going to use the following variables:
    - `hh_b04` (age)
    - `hh_b11` (main occupation)

▼ Code

```
1 table(df$hh_b11)
```

```
1
2              AGRICULTURE / LIVESTOCK                          disabled    EMPLOYED (NOT AG): WITH EMPLOYEES EMPLOYED (NOT AG):
3                                 5320                               238                                  360
```

- The relevant two values are “AGRICULTURE / LIVESTOCK” and “student”

First, we need to create age groups

- Let's create age groups using the `cut` function
  - This will create a *factor* variable

▼ Code

```
1 df <- df |>
2   mutate(age_groups = cut(hh_b04, breaks = seq(from = 0, to = max(df$hh_b04), by = 5)))
3 head(df$age_groups)
```

① Creates a new variable `age_groups` based on the conditions given (line 3)

```
1 [1] (70,75] (40,45] (30,35] (5,10]  (45,50] (35,40]
2 Levels: (0,5] (5,10] (10,15] (15,20] (20,25] (25,30] (30,35] (35,40] (40,45] (45,50] (50,55] (55,60] (60,65] (65,70] (70,75] (75,
```

- The `seq` function creates a sequence of numbers from 0 to the maximum age in the dataset, in increments of 5

Now, let's create the occupation variables

▼ Code

```
df <- df |>
  mutate(ag = ifelse(hh_b11 == "AGRICULTURE / LIVESTOCK", 1, 0),
    ed = ifelse(hh_b11 == "student", 1, 0))
```

- Now aggregate to `age_groups`

▼ Code

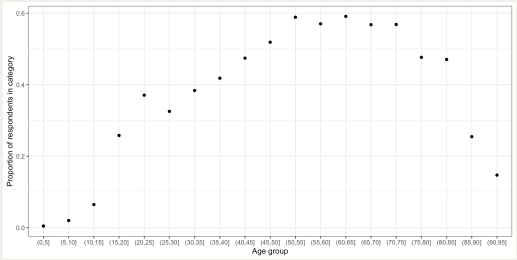
```
dfgroups <- df |>
  filter(!is.na(age_groups)) |>                    ①
  group_by(age_groups) |>                          ②
  summarize(ag = mean(ag, na.rm = TRUE),           ③
    ed = mean(ed, na.rm = TRUE)) |>
  ungroup()                                        ④
```

① Removes missing values
② Groups by `age_groups`
③ Calculates the mean of `ag` and `ed` (removing missing values)
④ Ungroups the data frame

Now we can plot them

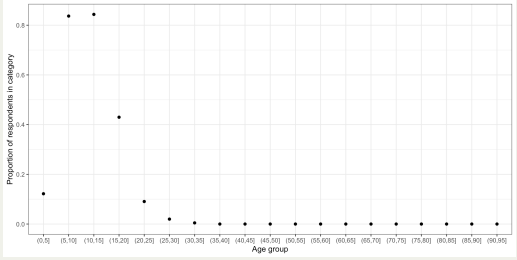
▼ Code

```
ggplot() +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ag)) +
  labs(x = "Age group",
    y = "Proportion of respondents in category") +
  theme_bw()
```



▼ Code

```
ggplot() +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ed)) +
  labs(x = "Age group",
    y = "Proportion of respondents in category") +
  theme_bw()
```

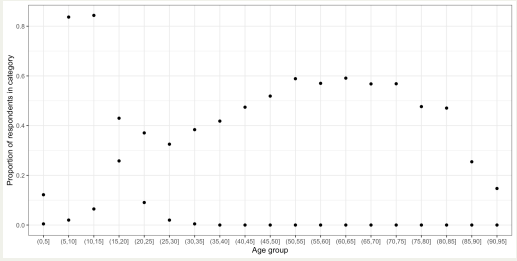


But what if we want to plot them on the same graph?

- This is a little more complicated!
- Look what happens here:

▼ Code

```
ggplot() +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ag)) +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ed)) +
  labs(x = "Age group",
    y = "Proportion of respondents in category") +
  theme_bw()
```



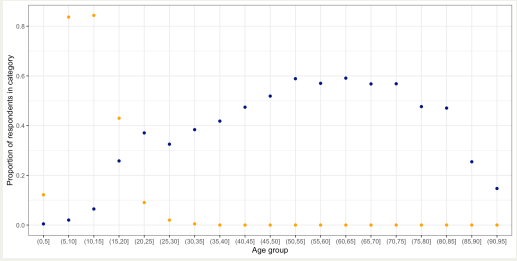
- Do you see the problem?

But what if we want to plot them on the same graph?

- We can specify individual colors.
- Look what happens here:

▼ Code

```
ggplot() +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ag), color = "navy") +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ed), color = "orange") +
  labs(x = "Age group",
    y = "Proportion of respondents in category") +
  theme_bw()
```



- Still a problem! What is it?

Here's the code. Let's discuss!

▼ Code

```
library(RColorBrewer)
colors <- brewer.pal(3, "Set2")
colors
```

```
[1] "#66C2A5" "#FC8D62" "#8DA0CB"
```

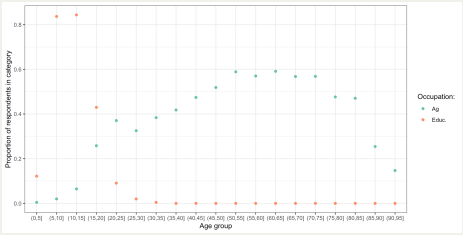
▼ Code

```
ggplot() +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ag, color = "Ag")) +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ed, color = "Educ.")) +
  labs(x = "Age group",
    y = "Proportion of respondents in category") +
  scale_color_manual("Occupation:",
    values = c("Ag" = colors[1], "Educ." = colors[2])) +
  theme_bw()
```

How does it look?

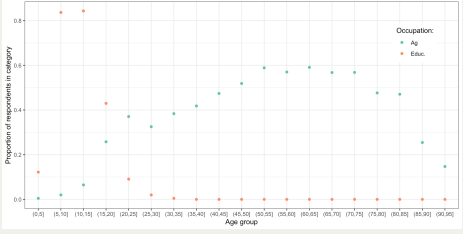
▼ Code

```
ggplot() +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ag, color = "Ag")) +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ed, color = "Educ.")) +
  labs(x = "Age group",
    y = "Proportion of respondents in category") +
  scale_color_manual("Occupation:",
    values = c("Ag" = colors[1], "Educ." = colors[2])) +
  theme_bw()
```



▼ Code

```
ggplot() +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ag, color = "Ag")) +
  geom_point(data = dfgroups,
    aes(x = age_groups, y = ed, color = "Educ.")) +
  labs(x = "Age group",
    y = "Proportion of respondents in category") +
  scale_color_manual("Occupation:",
    values = c("Ag" = colors[1], "Educ." = colors[2])) +
  theme_bw() +
  theme(legend.position = c(0.9, 0.8))
```



What about doing line graphs?

- We have to make a change! We cannot do a line graph with a factor variable!

▼ Code

```
ggplot() +
  geom_line(data = dfgroups, aes(x = as.numeric(age_groups), y = ag, color = "Ag")) +
  geom_line(data = dfgroups, aes(x = as.numeric(age_groups), y = ed, color = "Educ.")) +
  scale_x_continuous(breaks = as.numeric(unique(dfgroups$age_groups)), labels = unique(dfgroups$age_groups)) +
  labs(x = "Age group", y = "Proportion of respondents in category") +
  scale_color_manual("Occupation:", values = c("Ag" = colors[1], "Educ." = colors[2])) +
  theme_bw() +
  theme(legend.position = c(0.9, 0.8))
```

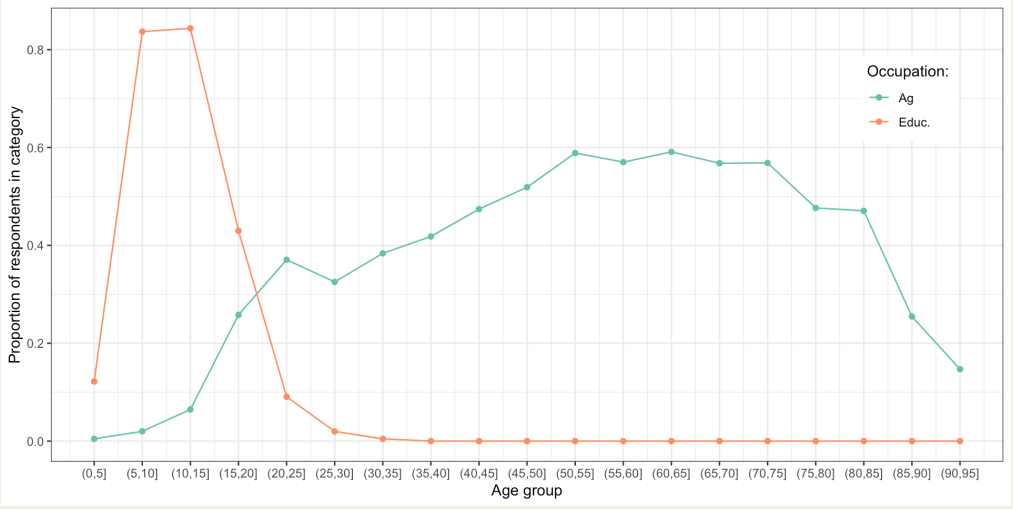
① Turn it into a numeric variable.
② Change the x-axis to the numeric values and label it with the original values.

Finally, adding them together!

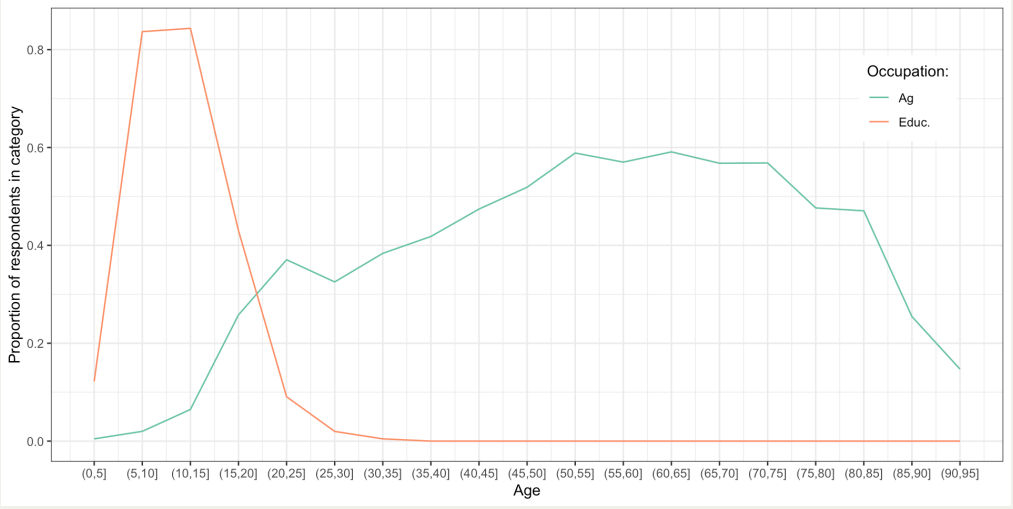
▼ Code

```
ggplot() +
  geom_point(data = dfgroups, aes(x = as.numeric(age_groups), y = ag, color = "Ag")) +
  geom_line(data = dfgroups, aes(x = as.numeric(age_groups), y = ag, color = "Ag")) +
  geom_point(data = dfgroups, aes(x = as.numeric(age_groups), y = ed, color = "Educ.")) +
  geom_line(data = dfgroups, aes(x = as.numeric(age_groups), y = ed, color = "Educ.")) +
  scale_x_continuous(breaks = as.numeric(unique(dfgroups$age_groups)), labels = unique(dfgroups$age_groups)) +
  labs(x = "Age group", y = "Proportion of respondents in category") +
  scale_color_manual("Occupation:", values = c("Ag" = colors[1], "Educ." = colors[2])) +
  theme_bw() +
  theme(legend.position = c(0.9, 0.8))
```

How it looks


Proportion of respondents in category
0.0
0.2
0.4
0.6
0.8
Occupation:
Ag
Educ.
(0,5]
(5,10]
(10,15]
(15,20]
(20,25]
(25,30]
(30,35]
(35,40]
(40,45]
(45,50]
(50,55]
(55,60]
(60,65]
(65,70]
(70,75]
(75,80]
(80,85]
(85,90]
(90,95]
Age group

What about doing line graphs?



Now it’s your turn!

- Select some variables from the dataset
- You need to create the following:
  - A graph with only one variable (e.g. density)
  - A graph with two variables (e.g. point or line)