Introduction to R Session 2

Department of Sociology | University of Oxford

Casey Breen

2023-10-06

Intro to R – Housekeeping

- Course materials available from:
  - www.github.com/caseybreen/intro_r
- My email: casey.breen@sociology.ac.uk.ox

Recap of session 1

- What’s the difference between `R` and `RStudio`?
- What’s a `vector`? What’s a `data.frame`?
- What does the `$` operator do? What does `data$column_b` do?
- What does `%in%` operator do?
- What does the `!` operator do?

Session 2

- Reading in data
- Data manipulation (`dplyr`)
- Data visualization (`ggplot2`)
- Best practices: coding style, commenting, and documentation
- Resources for self-study

Reading in data: paths

- **Absolute Path**: Specifies the full URL or address to locate a file or directory. Starts with the root directory.
  - Windows: `C:\Users\username\folder\file.csv`
  - macOS/Linux: `/home/username/folder/file.csv`
- **Relative Path**: Specifies how to find the file or directory based on the current working directory.
  - `folder/file.csv`

Getting working directory

- The working directory is the folder where your R session or script looks for files to read, or where it saves files you write
- Commands like `read.csv("file.csv")` or `write.csv(data, "file.csv")` will read from or write to this directory by default
- You can check the current working directory with `getwd()` and set it with `setwd("/path/to/folder")` in R

```r
getwd()
```

```
[1] "/Users/caseybreen/workspace/teaching/intro_r/slides"
```

Reading in .csv files

- To read in .csv files use `read_csv()`
  - This will read in the .csv file into memory as a `data frame`

```r
library(tidyverse)
df <- read_csv("dataset.csv")
```

- Write out .csv file using `write_csv()`:

```r
write_csv(data, "dataset_v2.csv")
```

In-class exercise 1

1. Load and install the `tidyverse` packages using the commands `install.packages()` and `library()`
2. Use the `read_csv()` function to read in the dataset and assign it to the object `censoc`
3. Use the `head` command to look at the first 5 rows
4. How many columns are in the dataset?
5. How many rows are in the dataset?
6. List the column names. What are a few research questions that could be addressed using this dataset?

Exercise 1 solutions

1. Load and install the `tidyverse` packages using the commands `install.packages()` and `library()`

```r
install.packages(tidyverse)
library(tidyverse)
```

3. Use the `read_csv()` function to read in the dataset and assign it to the object `censoc`

```r
censoc <- read_csv("censoc_numident_demo_codebook_v2.1.pdf")
```

3. Use the `head()` command to look at the first 5 rows

```r
head(censoc)
```

Exercise 1 solutions (cont.)

4. How many columns are in the dataset?

```
ncol(censoc)
```

```
[1] 39
```

5. How many rows are in the dataset?

```
nrow(censoc)
```

```
[1] 85865
```

6. List the column names.

```
names(censoc)
```

```
 [1] "histid"                      "byear"
 [3] "bmonth"                      "dyear"
 [5] "dmonth"                      "death_age"
 [7] "race_first"                  "race_first_cyear"
 [9] "race_last"                   "bpl_string"
[11] "zip_residence"               "socstate"
[13] "socstate_string"             "age_first_application"
[15] "link_abe_exact_conservative" "weight"
[17] "weight_conservative"         "perwt"
[19] "age"                         "sex"
[21] "bpl"                         "mbpl"
[23] "fbpl"                        "educd"
[25] "empstatd"                    "hispan"
[27] "incnonwg"                    "incwage"
[29] "marst"                       "nativity"
```

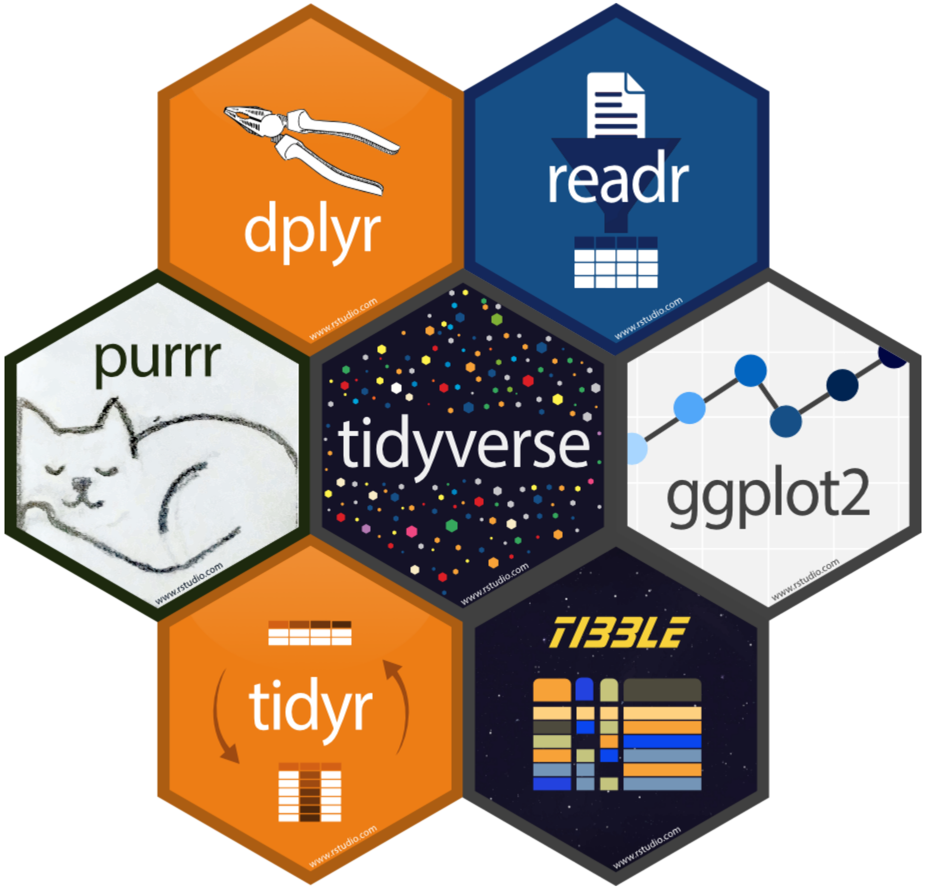
```
[31] "occ"                         "occscore"
[33] "ownershp"                    "pernum"
[35] "race"                        "rent"
[37] "serial"                      "statefip"
[39] "urban"
```

Break

- 10 minutes

Tidyverse

- Packages: Collection of R packages designed for data science.
- Data manipulation: Simplifies data cleaning and transformation with `dplyr`.
- Data Visualization: Enables advanced plotting with `ggplot2`.



Data Manipulation using `dplyr`

`filter`: Select rows based on conditions.

```r
filtered_df <- filter(df, age > 21)
```

`select`: choose specific columns

```r
filtered_df <- select(df)
```

`mutate`: Add or modify columns

```r
df <- mutate(df, age_next_year = age + 1)
```

`summarize` or `summarise` : aggregate or summarize data based on some criteria

```r
filtered_df <- summarize(df, mean(age))
```

`group_by`: Group data by variables. Often used with **`summarise()`**.

```r
filtered_df <- df %>%
  group_by(gender) %>%
  summarize(mean(age))
```

The Pipe Operator %>% (or |>) in R

- Takes the output of one function and passes it as the first argument to another function
- Simply put: “And then do…”
- What’s the below code doing?

```r
filtered_df <- df %>%
  group_by(gender) %>%
  summarize(mean(age))
```

Live coding demo - data manipulation

- Filter data
- Selecting data
- Calculating summary statistics
- Calculating summary statistics by group
- Creating new variable

In-class exercise 2

1. Filter the `censoc` data frame to include only women (sex == 2). Use the `filter` command.
2. Filter the dataset to only include people born between 1905 and 1920 using the `byear` variable.
3. Select the columns `histid`, `death_age`, `sex`, and `ownershp`
4. Calculate the average age of death for women (hint: refer to question 1)

Exercise 2 solutions

1. Filter the `censoc` data frame to include only women (sex == 2). Use the `filter` command.

```r
censoc %>%
  filter(sex == 2)
```

2. Filter the dataset to only include people born between 1905 and 1920 using the `byear` variable. Do this two different ways.

```r
## method 1
censoc %>%
  filter(byear >= 1905 & byear <=1920)

## method 2
censoc %>%
  filter(byear >= 1904 & byear <=1920)
```

Exercise 2 solutions (cont.)

3. Select the columns `histid`, `death_age`, `sex`, and `ownershp`

```r
censoc_select <- censoc %>%
  select(histid, death_age, sex, ownershp)

head(censoc_select)
```

```
# A tibble: 6 × 4
  histid                               death_age   sex ownershp
  <chr>                                    <dbl> <dbl>    <dbl>
1 235C4FA2-B407-4E61-A31D-DBF299C1C120        85     1        1
2 0DE161A7-34A7-47EA-B053-EA8549172CCC        77     1        1
3 EFF79CEC-DA83-482A-AB9A-FFCAC3C9A6A5        77     1        1
4 B51D01FA-54A4-4E5E-8BCF-B6D9521A2983        73     2        2
5 D545AEB1-C5C3-4E32-BB22-4BF58CF50311        73     1        2
6 A71A537B-C440-4E85-A276-334B05B723A7        82     2        1
```

4. Calculate the average age of death for women (hint: refer to question 1)

```r
censoc %>%
  filter(sex == 2) %>%
  summarize(mean_death_age_women = mean(death_age))
```

```
# A tibble: 1 × 1
  mean_death_age_women
                 <dbl>
1                 78.2
```

Data visualization using ggplot

- `ggplot2` provides a powerful and flexible system for creating a variety of data visualizations
- `aes`: Defines what data to show
- `geoms`: Chooses the type of plot
  - `geom_point()`: Scatter plot
  - `geom_line()`: Line plot
  - `geom_bar()`: Bar chart
  - `geom_histogram()`: Histogram

Data visualization using ggplot

```r
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

```r
ggplot(data = censoc) +
  geom_histogram(aes(x = death_age)) +
  facet_wrap(~sex) + ## look for both men and women
  theme_bw() ## make prettier
```

Break

- 10 minutes

Understanding NA Values in R

- `NA` represents missing or undefined data.
    - Can vary by data type (e.g., **`NA_character_`** and **`NA_integer_`**)
- **`NA`** values can affect summary statistics and data visualization.
- What happens when you run the code below?

```r
vec <- c(1, 2, 3, NA)
mean(vec)
```

Recoding values in R

- Sometime you want to recode a variable to take different values (e.g., recoding exact income to binary high/low income variable)
- The `case_when()` function in R is part of the `dplyr` package and is used for creating new variables based on multiple conditions:

```r
new_var <- case_when(
  condition1 ~ value1,
  condition2 ~ value2,
  TRUE ~ value_otherwise
)
```

In-class exercise 3

1. Make a histogram of the variable `death_age`. When are most people dying?
2. Make a histogram of the variable `byear`. When are most people born?
3. Recode the variable `sex` from numeric (1, 2) to take values "men" and "women"
4. Calculate the mean of of death for both men and women using `group_by()` and `summarize()`. Do men or women live longer?
5. Make a histogram of the variable `death_age` for both men and women. Use the `filter()` command.
6. Now try adding the following line to the histogram you made in question 1: `+ facet_wrap(~sex)`

In-class exercises 3

1. Make a histogram of the variable `death_age`. When are most people dying?

```r
ggplot(data = censoc) +
  geom_histogram(aes(x = death_age))
```

2. Make a histogram of the variable `byear`. When are most people born?

```r
ggplot(data = censoc) +
  geom_histogram(aes(x = death_age))
```

3. Calculate the mean of of death for both men and women using `group_by()` and `summarize()`. Do men or women live longer?

```r
ggplot(data = censoc) +
  geom_histogram(aes(x = death_age))
```

In-class exercises 3 (cont.)

4. Make a histogram of the variable `death_age` for both men and women. Use the `filter()` command.

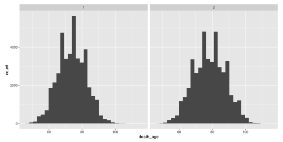
```r
## filter
censoc_men <- censoc %>% filter(sex == 1)
censoc_women <- censoc %>% filter(sex == 2)

## histogram for men
ggplot(data = censoc_men) +
  geom_histogram(aes(x = death_age))

## histogram for women
ggplot(data = censoc_women) +
  geom_histogram(aes(x = death_age))
```

5. Now try adding the following line to the histogram you made in question 1: `+ facet_wrap(~sex)`

```r
ggplot(data = censoc) +
  geom_histogram(aes(x = death_age)) +
  facet_wrap(~sex)
```



Best practices (opinionated)

- **Style**: use descriptive names and “snake_case”
- **Documentation**: Start commenting your code early, it’s a good habit for the future.
- **Learn `tidyverse`**: It offers a more coherent syntax and is widely used in data science.
- **Eventually**: R-packages, github integration, etc.

When you’re stuck

- Google
  - Lots of packages have documentation available online
  - Stack overflow – excellent resource
- Use help syntax (e.g., `?dplyr`)
- GPT4 (decent, but be careful!)

Resources for learning more

Questions?

In-class exercise 4

Do homeowners in the United States live longer than renters in the United States?

1. Google “IPUMS ownershp variable” and look at what each numerical value means.
2. Recode `ownershp` to create a character variable `homeowner` that takes value “homeowner” or “renter”. Filter out cases where we don’t know whether someone was a homeowner or not.
3. Make a histogram on the age of death for “homeowner” and “renter” groups using `ggplot`
4. Calculate the average age of death for “homeowner” and “renter” groups. Which group lives longer, on average? Does this analysis tell us anything about homeownership and longevity?

Exercise 4 solution

2. Recode `ownershp` to create a character variable `homeowner` that takes value “homeowner” or “renter”. Filter out cases where we don’t know whether someone was a homeowner or not.

```r
censoc <- censoc %>%
  filter(ownershp != 0) %>%
  mutate(homeowner = case_when(
    ownershp == 1 ~ "homeowner",
    ownershp == 2 ~ "renter"
  ))
```

3. Make a histogram on the age of death for “homeowner” and “renter” groups using `ggplot`

```r
censoc %>%
  ggplot(aes(x = death_age)) +
  geom_histogram() +
  facet_wrap(~homeowner)
```

Exercise 4 solution (cont.)

4. Calculate the average age of death for “homeowner” and “renter” groups. Which group lives longer, on average? Does this analysis tell us anything about homeownership and longevity?

```
censoc %>%
  group_by(homeowner) %>%
  summarize(mean(death_age))
```

```
# A tibble: 2 × 2
  homeowner `mean(death_age)`
  <chr>                 <dbl>
1 homeowner              76.5
2 renter                 75.8
```

Thank you

- Course materials available from:
  - www.github.com/caseybreen/intro_r
- Recommendation: try to finish exercises
- Questions: casey.breen@sociology.ox.ac.uk