Week 3: ‘Tidying’ data

Alex Lishinski

August 31, 2021

Welcome!

Welcome to *week 3*!

Great job last week.

Now we feel like we have enough of an introduction to some of the basics of working with R that we can move to the more substantial material. However, don't be discouraged if you don't feel rock solid about any of the basics we've covered so far. We will continue to reinforce these as we move forward.

Record the meeting

Discussion

- How did you find the homework? Too easy, too hard, confusing?
- From the chapter introducing educational data science, which of the educational data science roles do you see as the most similar to the work you have done, are doing, or intend to do, and why? If you are coming from outside of education, discuss any similarities (or the lack thereof)!
- What are you looking forward to (general life question)?

(10 minutes)

Recap: Foundational R skills

A general framework for you to use as a foundation and as a set of concepts to help you work through the class.

The four core concepts we will use to build our framework are:

1. Projects
2. Packages
3. Data
4. Functions

You will use each of these in most of your analyses with R.

Recap: Foundational R skills

These foundational skills are what we will build on as we move to data wrangling and tidying

We're not done building your understanding of working with packages, data, and functions

Recap: Packages quick review

1. What are packages?
    - Code bundles that add functionality to R
    - Examples: ggplot2, dplyr, rtweet, quanteda, lme4
2. Where do we get packages?
    - CRAN or GitHub
3. How do we install packages?
    - `install.packages("pkg-name")`
4. How do we know what packages to use?
    - Searching
    - People and news related to R (more later - there are *tons*)
    - CRAN task views
5. How do we use packages?
    - `library(pkgname)`

Overview of today's class

Wrangling and Tidying

1. Reading in Data
2. Tidying data
3. Our Tidy data tools

1. Reading in Data

So far, we have used ***built-in data***. There is a lot of built-in data!

Loading different types of data, assigning to a name

Comma-separated values (`.csv`)

```
library(readr)
data <- readr::read_csv(here( "content", "data", "answer_export.csv"))
```

1. Reading in Data

.xlsx

```
library(readxl)

data <- read_excel((here("content", "data", "schedule.xlsx")))
```

1. Reading in Data

`.sav`

```
library(haven)
data <- read_sav((here("data", "file-name.sav")))
```

1. Reading in Data

Google Sheets

```
library(googlesheets4)
```

Web

```
data <- read_csv("https://github.com/data-edu/dataedu/raw/master/data-raw/wt01_online-science-motivat
```

1. Reading in Data

Pitfalls

Can't find data file

- Check file location
- Check working directory (here base directory)
- Make sure those line up

1. Reading in Data

Other considerations to keep in mind

- Missing column name headers, csv files in particular (`header = T`)

```
data <- readr::read_csv(here("content", "data", "answer_export.csv"), col_names = F, skip =1)
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   X1 = col_double(),
##   X2 = col_character(),
##   X3 = col_character(),
##   X4 = col_double()
## )
```

```
colnames(data)
```

```
## [1] "X1" "X2" "X3" "X4"
```

```
colnames(data) <- c("id", "content", "session_id", "question_id")

data
```

```
## # A tibble: 2,396 × 4
##       id content    session_id                          question_id
##    <dbl> <chr>      <chr>                                     <dbl>
##  1    10 Ready      SM275024c5512ea0f626560c2a7c974a8b            1
##  2    11 I'm ready. SMe14d7954247d88c43cf6bd5e70f17cc2            1
##  3    12 3          SM4c0c4addefec91a5ad5d92230606d8ec            1
##  4    13 4          SMc559b03477909c853ba0bd7cbe745651            1
##  5    14 3          SM3465833f00a6119440163db13a729c4b            1
##  6    15 4          SMb2ce42bc10c0a75172640c7b728f80ad            1
##  7    16 3          SM11cb1f8d5bcf86ca43d8e8802820d80f            2
##  8    17 4          SM391e2a4f2315eceee4ba95fb04e12ca6            1
##  9    18 4          SMe0a1f06bff4a847ebf00f87fb2b6d38d            2
```

1. Reading in Data

Other considerations to keep in mind

- renaming column headers with `dplyr::rename()`

```
data <- readr::read_csv(here("content", "data", "answer_export.csv"))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   id = col_double(),
##   content = col_character(),
##   session_id = col_character(),
##   question_id = col_double()
## )
```

```
colnames(data)
```

```
## [1] "id"          "content"     "session_id"  "question_id"
```

```
data <- data %>%
  rename(ID_num = id,
         Response = content,
         Session_num = session_id,
         Question_num = question_id)

colnames(data)
```

```
## [1] "ID_num"       "Response"     "Session_num"  "Question_num"
```

2. Tidying Data

Tidying data AKA wrangling data, cleaning data, etc.

- Data cleaning is 80% of the job
- Don't underestimate it, know what to expect

2. Tidying Data

```
“Happy families are all alike;
every unhappy family is unhappy in its own way.”
–– Leo Tolstoy

“Tidy datasets are all alike,
but every messy dataset is messy in its own way.”
–– Hadley Wickham
```

What does 'tidy' data mean

1. Every Column is a variable
2. Every Row is an observation
3. Every cell is a single value

(Don't worry too much about the details of what these mean quite yet)

Primary importance: the tools that we will use are built around tidy data.

2. Tidying data

Examining data is the first step

- `View()`
- `glimpse()`
- `colnames()` or `names()`
- `table()` or `tabyl()`

2. Tidying data

Deeper aspects of cleaning/tidying data

- Columns should be the right type (e.g. integer, double, factor, character, additional character types like date/time)
- Values in column should only be valid values for whatever that variable is supposed to be (sometimes your data sources will pre-validate this, sometimes not)
- Missing data indicators for missing values (NA type)

2. Tidying data

Data is stored in data frames or tibbles

Variables are vectors, and there are a few types of these in R based on what type of information the entries in the vector are

Character: Text strings, in double quotes Numeric: numbers, with decimals Logical: TRUE or FALSE values Integer: Whole number values Factor: categorical variables with a fixed number of levels

2. Tidying data

Vector types need to be correct for what you were expecting or functions won't work correctly

Most of the time this will work as expected but it's important to be mindful when cleaning data

```
glimpse(storms)
```

```
## Rows: 10,010
## Columns: 13
## $ name        <chr> "Amy", "Amy", "Amy", "Amy", "Amy", "Amy", "Amy", "Amy", "A…
## $ year        <dbl> 1975, 1975, 1975, 1975, 1975, 1975, 1975, 1975, 1975, 1975…
## $ month       <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7…
## $ day         <int> 27, 27, 27, 27, 28, 28, 28, 28, 29, 29, 29, 29, 30, 30, 30…
## $ hour        <dbl> 0, 6, 12, 18, 0, 6, 12, 18, 0, 6, 12, 18, 0, 6, 12, 18, 0,…
## $ lat         <dbl> 27.5, 28.5, 29.5, 30.5, 31.5, 32.4, 33.3, 34.0, 34.4, 34.0…
## $ long        <dbl> -79.0, -79.0, -79.0, -79.0, -78.8, -78.7, -78.0, -77.0, -7…
## $ status      <chr> "tropical depression", "tropical depression", "tropical de…
## $ category    <ord> -1, -1, -1, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
## $ wind        <int> 25, 25, 25, 25, 25, 25, 25, 30, 35, 40, 45, 50, 50, 55, 60…
## $ pressure    <int> 1013, 1013, 1013, 1013, 1012, 1012, 1011, 1006, 1004, 1002…
## $ ts_diameter <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ hu_diameter <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
```

3. Our Tidy Data Tools

What are our tools for tidy data?

- Tidyverse
- More specifically tidyr and dplyr are the wrangling tools
- tidyr narrower focus on reshaping data
- dplyr does more, stuff like filtering arranging selecting joining
- dplyr also has powerful tools for grouping data and grouped operations
- stringr: not talked about as much but useful operations for dealing with character strings

3. Our Tidy Data Tools

How dplyr functions work:

- First argument is always data frame (important for %>%)
- They don't modify the existing data frame, need to save the results

```
data <- data %>%
  rename(ID_num = id,
         Response = content,
         Session_num = session_id,
         Question_num = question_id)
```

3. Our Tidy Data Tools

How dplyr functions work:

Very amenable to chained operations with `%>%`

```
data <- readr::read_csv(here("content", "data", "answer_export.csv"))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   id = col_double(),
##   content = col_character(),
##   session_id = col_character(),
##   question_id = col_double()
## )
```

```
data_subset <- data %>%
  rename(ID_num = id,
         Response = content,
         Session_num = session_id,
         Question_num = question_id) %>%
  select(Response, Question_num)
```

3. Our Tidy Data Tools

How dplyr functions work:

Grouped operations

```
data <- readr::read_csv(here("content", "data", "answer_export.csv"))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   id = col_double(),
##   content = col_character(),
##   session_id = col_character(),
##   question_id = col_double()
## )
```

```
data_grp <- data_subset %>%
  group_by(Question_num)
```

3. Our Tidy Data Tools

Dplyr `mutate()` function

Can use to add columns or change columns

```
data <- data %>% dplyr::mutate(new_col = 1:nrow(data))
```

3. Our Tidy Data Tools

`stringr` package has a number of tools used to clean up text data

```
library(stringr)

data <- data %>% dplyr::mutate(stringr::str_trim(col1))
```

Wrapping up

On Slack:

- What is one thing you took away from today?
- What is something you want to learn more about?