STAT 451 - Visualizing Data - Autumn 2025

Ariane Ducellier

10/02/2025

Tutorial Tidyverse part 2

Today, we are going to continue reviewing useful R functions for reading and exploring data sets. We will focus on dealing with missing data and getting data from the web.

We will need the following R libraries:

```
library(httr)
library(jsonlite)
library(mice)
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(rvest)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter()         masks mice::filter(), stats::filter()
## x purrr::flatten()        masks jsonlite::flatten()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x dplyr::lag()            masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

1. Dealing with missing data

We will read a data set and replace the ? by NA.

```
header <- c("age", "workclass", "fnlwgt", "education",
  "education_num", "marital_status", "occupation",
  "relationship", "race", "sex", "capital_gain",
```

```
  "capital_loss", "hours_per_week", "native_country", "target")
df <- read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data",
  col_names=header, trim_ws=TRUE)
```

```
## Rows: 32561 Columns: 15
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (9): workclass, education, marital_status, occupation, relationship, rac...
## dbl (6): age, fnlwgt, education_num, capital_gain, capital_loss, hours_per_week
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df <-df %>%
  mutate(workclass = na_if(workclass, "?"),
         occupation = na_if(occupation, "?"),
         native_country = na_if(native_country, "?"))
```

1.1 Filling values with previous value

A first method is to fill the missing data with the previous value in the table. This is a good method when dealing with time series data for example.

```
df_fill1 <- df %>%
  fill(workclass, occupation, native_country, .direction="down")
```

1.2 Filling values with most frequent value

This method is useful for categorical variables.

```
m_freq_workcls <- names(table(df$workclass))[which.max(table(df$workclass))]
m_freq_occup <- names(table(df$occupation))[which.max(table(df$occupation))]
df_fill2 <- df %>%
  replace_na(list(workclass = m_freq_workcls,
                  occupation = m_freq_occup))
```

1.3 Dropping rows with missing values

You can drop all the rows that have at least one missing value,

```
df_no_na <- df %>% na.omit()
```

or drop only the rows that have missing values for specific columns.

```
df_native <- df %>%
  drop_na(native_country)
```

1.4 Imputing with mice

Let us read the dataset.

```
data("txhousing")
txhousing$date <- date_decimal(txhousing$date, tz="GMT")
txhousing$city <- as.factor(txhousing$city)
```

We drop the rows that have 5 missing values because it will be difficult to impute with so many missing columns.

```
idx <- which(rowSums(is.na(txhousing)) == 5)
txhousing <- txhousing[-idx,]
```

For the sales, volume and median columns, we impute the missing data with the median value of the variable.

```
txhousing$sales[is.na(txhousing$sales)] <- median(txhousing$sales, na.rm=TRUE)
txhousing$volume[is.na(txhousing$volume)] <- median(txhousing$volume, na.rm=TRUE)
txhousing$median[is.na(txhousing$median)] <- median(txhousing$median, na.rm=TRUE)
```

We use the mice (Multivariate Imputation by Chained Equations) package to impute the missing values for the listings and inventory columns.

```
impute <- mice(data.frame(txhousing[,7:8]), seed=123)
```

```
##
##  iter imp variable
##   1   1  listings  inventory
##   1   2  listings  inventory
##   1   3  listings  inventory
##   1   4  listings  inventory
##   1   5  listings  inventory
##   2   1  listings  inventory
##   2   2  listings  inventory
##   2   3  listings  inventory
##   2   4  listings  inventory
##   2   5  listings  inventory
##   3   1  listings  inventory
##   3   2  listings  inventory
##   3   3  listings  inventory
##   3   4  listings  inventory
##   3   5  listings  inventory
##   4   1  listings  inventory
##   4   2  listings  inventory
##   4   3  listings  inventory
##   4   4  listings  inventory
##   4   5  listings  inventory
##   5   1  listings  inventory
##   5   2  listings  inventory
##   5   3  listings  inventory
##   5   4  listings  inventory
##   5   5  listings  inventory
```

```
impute_data <- complete(impute, 1)
txhousing_clean <- txhousing %>%
  mutate(listings = impute_data[,1],
         inventory = impute_data[,2])
```

1.5 Implicit missing values

In this example, the price for the 1st quarter of 2021 is missing, but you won't see it by just looking for the rows with NA in the data set.

```
stocks <- tibble(
  year  = c(2020, 2020, 2020, 2020, 2021, 2021, 2021),
  qtr   = c(   1,    2,    3,    4,    2,    3,    4),
  price = c(1.88, 0.59, 0.35,   NA, 0.92, 0.17, 2.66)
```

```
)
```

It becomes obvious that the row is missing when you pivot the data set to a wider table, but then you no longer have tidy data.

```
stocks %>%
  pivot_wider(
    names_from = qtr,
    values_from = price
  )
```

```
## # A tibble: 2 x 5
##    year   `1`   `2`   `3`   `4`
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  2020  1.88  0.59  0.35 NA
## 2  2021 NA     0.92  0.17  2.66
```

The complete function will fill your tidy data set with the missing rows and use NAs for the missing values.

```
stocks %>% complete(year, qtr)
```

```
## # A tibble: 8 x 3
##    year   qtr price
##   <dbl> <dbl> <dbl>
## 1  2020     1  1.88
## 2  2020     2  0.59
## 3  2020     3  0.35
## 4  2020     4 NA
## 5  2021     1 NA
## 6  2021     2  0.92
## 7  2021     3  0.17
## 8  2021     4  2.66
```

2. Getting data from the web

This is an example on how to copy a data table from a web page.

- Go to the Wiki page.
- Right-click and select Inspect.
- Find the piece of code that highlights the table.
- Right-click and select Copy > XPath.

You first start by reading the entire content of the web page.

```
page <- "https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)"
gdp <- rvest::read_html(page)
```

Then you can start by getting the text of the first paragraph.

```
p1 <- gdp %>%
  html_elements("p") %>%
  html_text()
p1[3]
```

```
## [1] "Gross domestic product (GDP) is the market value of all final goods and services from a nation i
```

If you have found the XPath corresponding to the table that you are interested in, you can read the table.

```
gdp_df <- gdp %>%
  html_elements(xpath = '/html/body/div[2]/div/div[3]/main/div[3]/div[3]/div[1]/table[2]') %>%
  html_table() %>%
  .[[1]]
```

3. Getting data from an API

To get data from an API, you will need the base URL and the end point. It should be provided in the API documentation.

The base URL is: https://api.fiscaldata.treasury.gov/services/api/fiscal_service

The end point is: /v1/accounting/mts/mts_table_1

Gathering both gives you data in the JSON format.

```
url <- "https://api.fiscaldata.treasury.gov/services/api/fiscal_service/v1/accounting/mts/mts_table_1"
treasury_api <- GET(url)
```

You can then transform the JSON format into a data frame.

```
result <- content(treasury_api, "text", encoding="UTF-8")
df_json <- fromJSON(result, flatten=TRUE)
df <- as.data.frame(df_json$data)
```

4. Miscellaneous functions

This is a list of miscellaneous useful functions that we have not covered yet. The first one is used to apply the same function to all the columns in the data set.

```
mtcars %>%
  select(hp, wt) %>%  map(mean)
```

```
## $hp
## [1] 146.6875
##
## $wt
## [1] 3.21725
```

This function is used to combine data sets by rows,

```
A <- mtcars[1:3, ]
B <- mtcars[4:6, ]
AB <- A %>% bind_rows(B)
```

and this one to combine data sets by columns.

```
A <- mtcars[1:5, 1:3]
B <- mtcars[1:5, 4:6]
 AB <- A %>% bind_cols(B)
```

This is another way of creating a new column with a condition on another column. It allows handling multiple cases of logical tests.

```
mtcars %>%
  mutate(transmission_type =
           case_when(
             am == 0 ~ "automatic",
             am == 1 ~ "manual"))
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
##                     transmission_type
## Mazda RX4                      manual
## Mazda RX4 Wag                  manual
## Datsun 710                     manual
## Hornet 4 Drive              automatic
## Hornet Sportabout           automatic
## Valiant                     automatic
## Duster 360                  automatic
## Merc 240D                   automatic
## Merc 230                    automatic
## Merc 280                    automatic
## Merc 280C                   automatic
## Merc 450SE                  automatic
## Merc 450SL                  automatic
## Merc 450SLC                 automatic
## Cadillac Fleetwood          automatic
## Lincoln Continental         automatic
## Chrysler Imperial           automatic
## Fiat 128                       manual
## Honda Civic                    manual
## Toyota Corolla                 manual
```

```
## Toyota Corona              automatic
## Dodge Challenger           automatic
## AMC Javelin                automatic
## Camaro Z28                 automatic
## Pontiac Firebird           automatic
## Fiat X1-9                     manual
## Porsche 914-2                 manual
## Lotus Europa                  manual
## Ford Pantera L                manual
## Ferrari Dino                  manual
## Maserati Bora                 manual
## Volvo 142E                    manual
```