Scraping Wikipedia

Data Analysis for the Social Sciences with R

2025-06-04

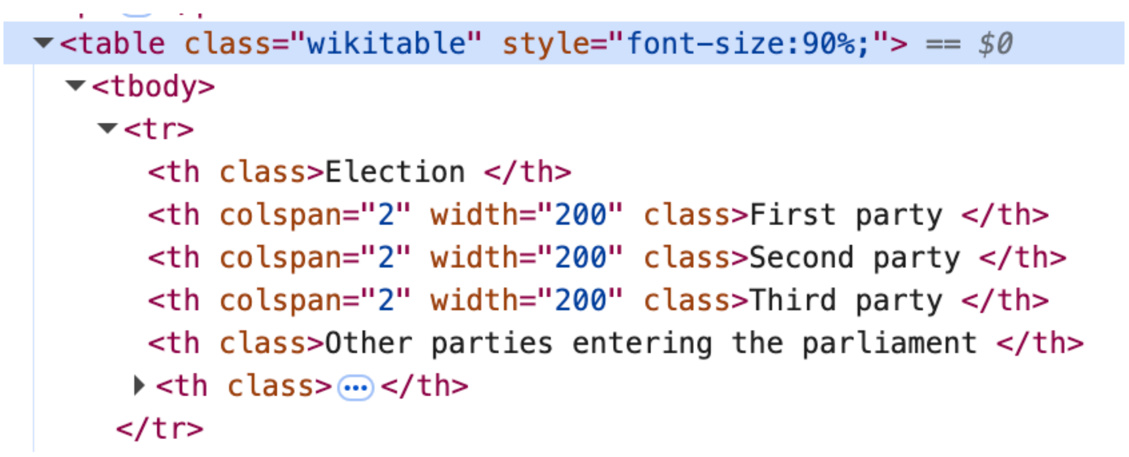
Table of contents

1 Scraping Wikipedia

We want to produce a plot with the vote share of the strongest party and the runner-up in Turkish elections since 1950. This information is available as a table on Wikipedia. We will first use the `rvest` package to read the Wikipedia page and extract the relevant information. Then we use `tidyverse` tools to wrangle the data and `ggplot2` to create a plot. We report everything in R markdown.

1.1 Inspecting the page

We first inspect the source code of the Wikipedia page to identify which structural elements we are looking for. This suggests that table elements on Wikipedia are contained in the tag `<table class='wikitable'></table>`.



```
▾<table class="wikitable" style="font-size:90%;"> == $0
  ▾<tbody>
    ▾<tr>
       <th class>Election </th>
       <th colspan="2" width="200" class>First party </th>
       <th colspan="2" width="200" class>Second party </th>
       <th colspan="2" width="200" class>Third party </th>
       <th class>Other parties entering the parliament </th>
      ▸<th class>⋯</th>
     </tr>
```

Inspecting HTML

How exactly you can inspect HTML code depends on the browser you use. I find Google Chrome to be most flexible.

1.2 Extracting tables

```
url <- "https://en.wikipedia.org/wiki/Parliamentary_elections_in_Turkey"

page <- read_html(url)

tables <- page %>%
  html_elements(".wikitable")
```

There are 3 `.wikitable` elements on the page. We are interested in the third. We can use the `html_table()` function to extract the table contents and directly convert them into a `tibble`.

```
table3 <- tables[3] %>%
  html_table()
head(table3, 5)
```

```
[[1]]
# A tibble: 21 x 9
   Election `First party` `First party`             `Second party` `Second party`
   <chr>    <lgl>         <chr>                     <lgl>          <chr>
 1 1946     NA            Republican People's Par~ NA             Democrat Part~
 2 1950     NA            Democrat Party(Celal Ba~ NA              Republican Pe~
 3 1954     NA            Democrat Party(Adnan Me~ NA              Republican Pe~
 4 1957     NA            Democrat Party(Adnan Me~ NA              Republican Pe~
 5 1961     NA            Republican People's Par~ NA              Justice Party~
 6 1965     NA            Justice Party(Süleyman ~ NA              Republican Pe~
 7 1969     NA            Justice Party(Süleyman ~ NA              Republican Pe~
 8 1973     NA            Republican People's Par~ NA              Justice Party~
 9 1977     NA            Republican People's Par~ NA              Justice Party~
10 1983     NA            Motherland Party(Turgut~ NA              Populist Part~
# i 11 more rows
# i 4 more variables: `Third party` <lgl>, `Third party` <lgl>,
#   `Other parties entering the parliament` <chr>, `Cabinets formed` <chr>
```

2 Data wrangling

The function extracted the data in the table, but in a rather unhelpful format. We need to reformat the data so that we can create a plot. First, there are useless columns (I suspect these create the colors in the original). I transform the `tibble` into a `data.frame`, remove the empty columns, rename all the columns, make year numerical, and filter out the first year.

```r
table3 <- table3 %>%
  as.data.frame() %>%
  select(-First.party,
         -Second.party,
         -Third.party,
         -Third.party.1,
         -Other.parties.entering.the.parliament,
         -Cabinets.formed) %>%
  rename(year=Election,
         first=First.party.1,
         second=Second.party.1) %>%
  mutate(year=as.numeric(year)) %>%
  filter(year>1946)
```

The resulting data frame is not **tidy**. The individual columns contain various pieces of information which should be separate. This is as good a time as any to start working with **regular expressions**, or regex for short.

2.1 Regex

Regular Expressions

Regular expressions (regex) are powerful patterns used to match and manipulate text based on specific rules. They are widely used in programming, data cleaning, and search operations. At their core, regex patterns are made up of **literal characters** (like `a`, `b`, `1`) and **metacharacters** that define complex search criteria.

Some of the most important elements include

- `.` (matches any character),
- `*` (matches zero or more of the preceding element),
- `+` (one or more),
- `?` (zero or one),
- `[]` (character classes),
- `[^]` (negative character classes),
- `\\s` (any whitespace),
- `^` (start of string),
- `$` (end of string),
- `()` (grouping),
- `?=pattern` (positive lookahead) and `?!pattern` (negative lookahead), meaning the match must (not) be directly before the pattern
- `?<=pattern` (positive lookbehind) and `?<!pattern` (negative lookbehind), meaning the match must (not) be directly after the pattern

You can also use `|` for logical OR and `\` to escape special characters. Together, these elements let you build precise and flexible search patterns for a wide range of text processing tasks.

Our cell entries look like this:

```
table3$first[1]
```

```
[1] "Democrat Party(Celal Bayar)52,67%  415 MPs"
```

Note the following patterns:

1. The party name is everything before the opening bracket `(`;
2. The leader name is enclosed in brackets;
3. The vote percentage is a number followed by `%`;
4. The number of MPs is a number followed by a white space and `MP`

We can create regular expressions based on these patterns. First, we will need the `stringr` package which allows us to manipulate strings in R. This package includes a function called `str_extract()` which takes two arguments, first a character string and then a regular expression defining the portion of the string to be extracted.

For the first pattern, we want everything from the beginning of the string up to, but excluding, the opening bracket. In regex, we can say `^[^(]+`. This breaks down as follows: `^` anchors everything to the beginning of the string; `[^]` is a negative character class, meaning: everything but; this class contains the opening bracket `(`; the negative charater class is followed by a `+` to indicate that we are looking for one or more characters, as long as they are not opening brackets.

Let's see if this works:

```r
str_extract(table3$first[1], "^[^(]+")
```

```
[1] "Democrat Party"
```

It does.

Now let's define a regex pattern to extract everything between brackets. The pattern is `(?<=\\().+(?=\\))`. Let's break this down as well: First, we start with a positive lookbehind: `(?<=\\()`, asserting that our match must be preceded by an opening bracket. The bracket itself needs to be escaped (`\\(`) since brackets are special characters in regex. Second, we select one or more of any character `.+`. Finally, we assert that the match must be followed by a closing bracket with a positive lookahead, `(?=\\))`, again escaping the bracket.

Does that work?

```r
str_extract(table3$first[1], "(?<=\\().+(?=\\))")
```

```
[1] "Celal Bayar"
```

Yep.

The vote percentage is one or two digits followed by a comma as a decimal separator (`,`), followed by at least one digit, followed by a percent sign `%`. In regex, we can say `\\d+,\\d+(?=%)`: one or more digits, `\\d+`, comma `,`, one or more digits, `\\d+`, plus a positive lookahead checking for a percent sign immediately after.

Did we get it?

```r
str_extract(table3$first[1], "\\d+,\\d+(?=%)")
```

```
[1] "52,67"
```

Looks like it.

Finally, we want the number of MPs. This is one or more digits, followed by a whitespace and "MP", or: `\\d+(?=\\s*MP)`.

```
str_extract(table3$first[1], "\\d+(?=\\s*MP)")
```

```
[1] "415"
```

Got it.

Now, let's put it all together and create new variables with this information:

```
table3 <- table3 %>%
  mutate(first_party=str_extract(first, "^[^(]+"),
         first_leader=str_extract(first, "(?<=\\().+(?=\\))"),
         first_percent=str_extract(first, "\\d+,\\d+(?=%)"),
         first_mps=str_extract(first, "\\d+(?=\\s*MP)"),
         second_party=str_extract(second, "^[^(]+"),
         second_leader=str_extract(second, "(?<=\\().+(?=\\))"),
         second_percent=str_extract(second, "\\d+,\\d+(?=%)"),
         second_mps=str_extract(second, "\\d+(?=\\s*MP)")
         ) %>%
  select(-first, -second)
```

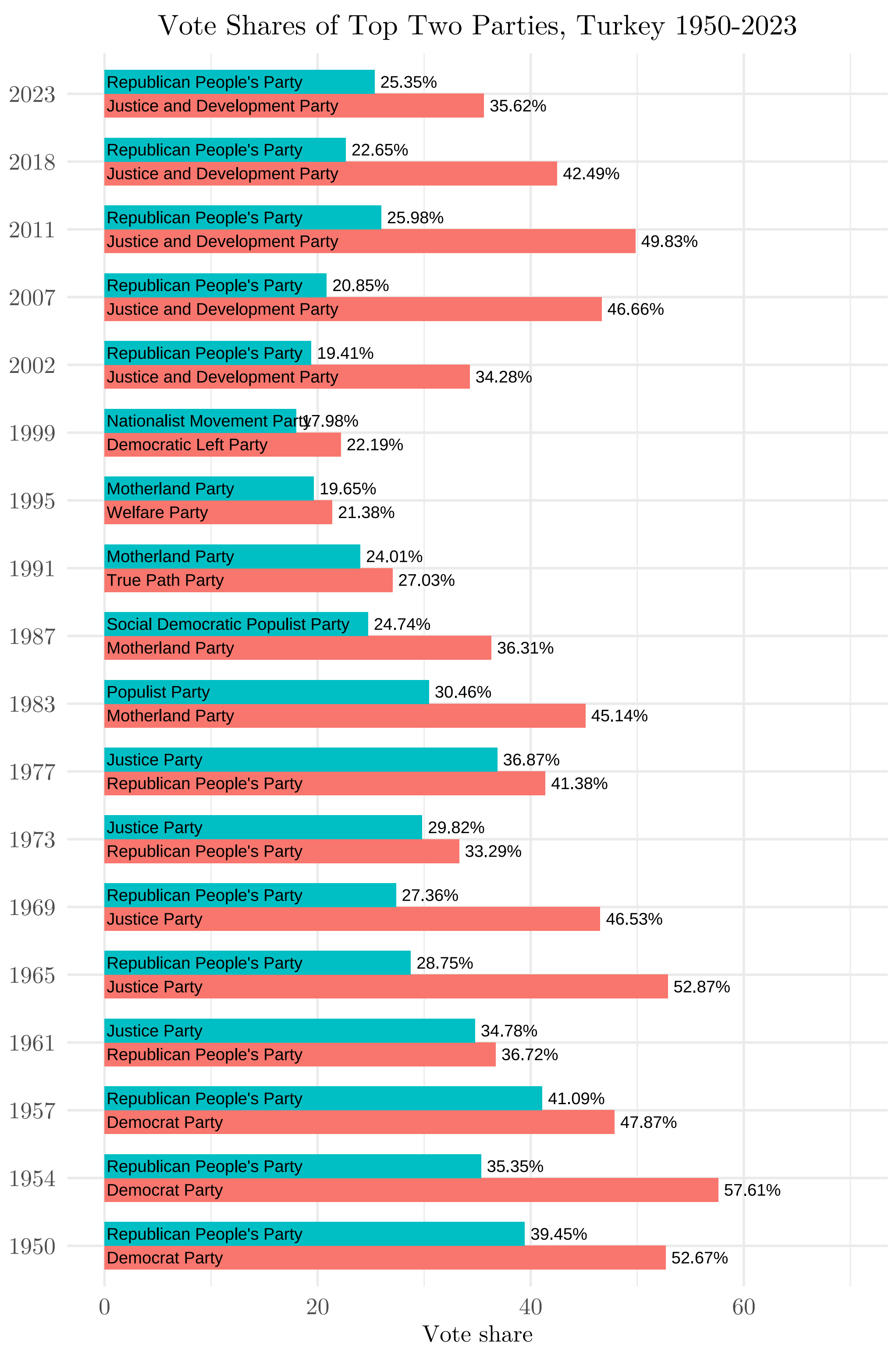
The data is still not **tidy**, we reshape

```
table_long <- table3 %>%
  pivot_longer(
    cols = c(first_party,
             second_party,
             first_leader,
             second_leader,
             first_percent,
             second_percent,
             first_mps,
             second_mps),
    names_to = c("position", ".value"),
    names_pattern = "(first|second)_(.+)"
  ) %>%
  mutate(
    mps = as.integer(mps),
    percent = as.numeric(gsub(",", ".", percent))
  ) %>%
  select(year, party, position, leader, percent, mps)
```

3 Plot

Now we can create the plot

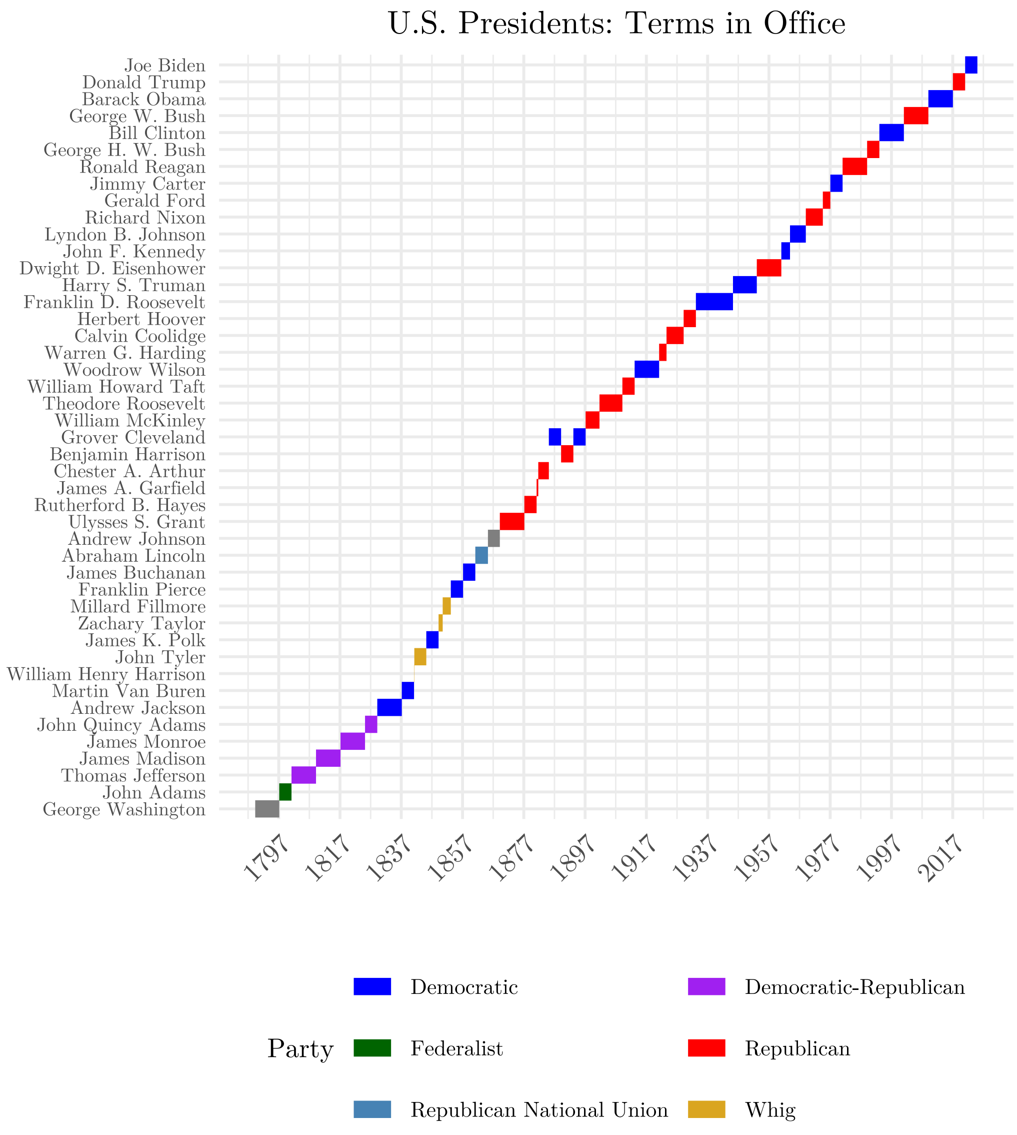
```
ggplot(table_long, aes(x = factor(year), y = percent, fill = position)) +
  geom_col(position = "dodge", width = 0.7) +
  geom_text(aes(label = party, y=0.2),
            position = position_dodge(width = 0.7),
            size = 2.5, hjust = 0) +
  geom_text(aes(label = paste0(percent,"%")),
            position = position_dodge(width = 0.7),
            size = 2.5, hjust=-0.1) +
  labs(title = "Vote Shares of Top Two Parties, Turkey 1950-2023",
       y = "Vote share",
       x= "") +
  custom_theme +
  ylim(0,70) +
  theme(
    legend.position = "none"
  ) +
  coord_flip()
```



Vote Shares of Top Two Parties, Turkey 1950-2023
2023
Republican People's Party 25.35%
Justice and Development Party 35.62%
2018
Republican People's Party 22.65%
Justice and Development Party 42.49%
2011
Republican People's Party 25.98%
Justice and Development Party 49.83%
2007
Republican People's Party 20.85%
Justice and Development Party 46.66%
2002
Republican People's Party 19.41%
Justice and Development Party 34.28%
1999
Nationalist Movement Party 17.98%
Democratic Left Party 22.19%
1995
Motherland Party 19.65%
Welfare Party 21.38%
1991
Motherland Party 24.01%
True Path Party 27.03%
1987
Social Democratic Populist Party 24.74%
Motherland Party 36.31%
1983
Populist Party 30.46%
Motherland Party 45.14%
1977
Justice Party 36.87%
Republican People's Party 41.38%
1973
Justice Party 29.82%
Republican People's Party 33.29%
1969
Republican People's Party 27.36%
Justice Party 46.53%
1965
Republican People's Party 28.75%
Justice Party 52.87%
1961
Justice Party 34.78%
Republican People's Party 36.72%
1957
Republican People's Party 41.09%
Democrat Party 47.87%
1954
Republican People's Party 35.35%
Democrat Party 57.61%
1950
Republican People's Party 39.45%
Democrat Party 52.67%
0
20
40
60
Vote share

4 Exercises

1. Get a list of all U.S. Presidents since George Washington from Wikipedia and plot their duration in office



Tip: The plot type I used is `geom_segment()`.

2. The `popoulation.csv` file on GitHub contains the population of countries around the world from 1995 to 2013. The format is a bit inconvenient, as the country, population, and year are in the same cell as in: `Afghanistan: 17586073 (1995)`. Load the data into

R using `read_csv()` and write `regex` functions to create separate columns for `country`, `population`, and `year`.