APIs and web scraping

Extracting online content with R

Andrew Heiss

September 12, 2022

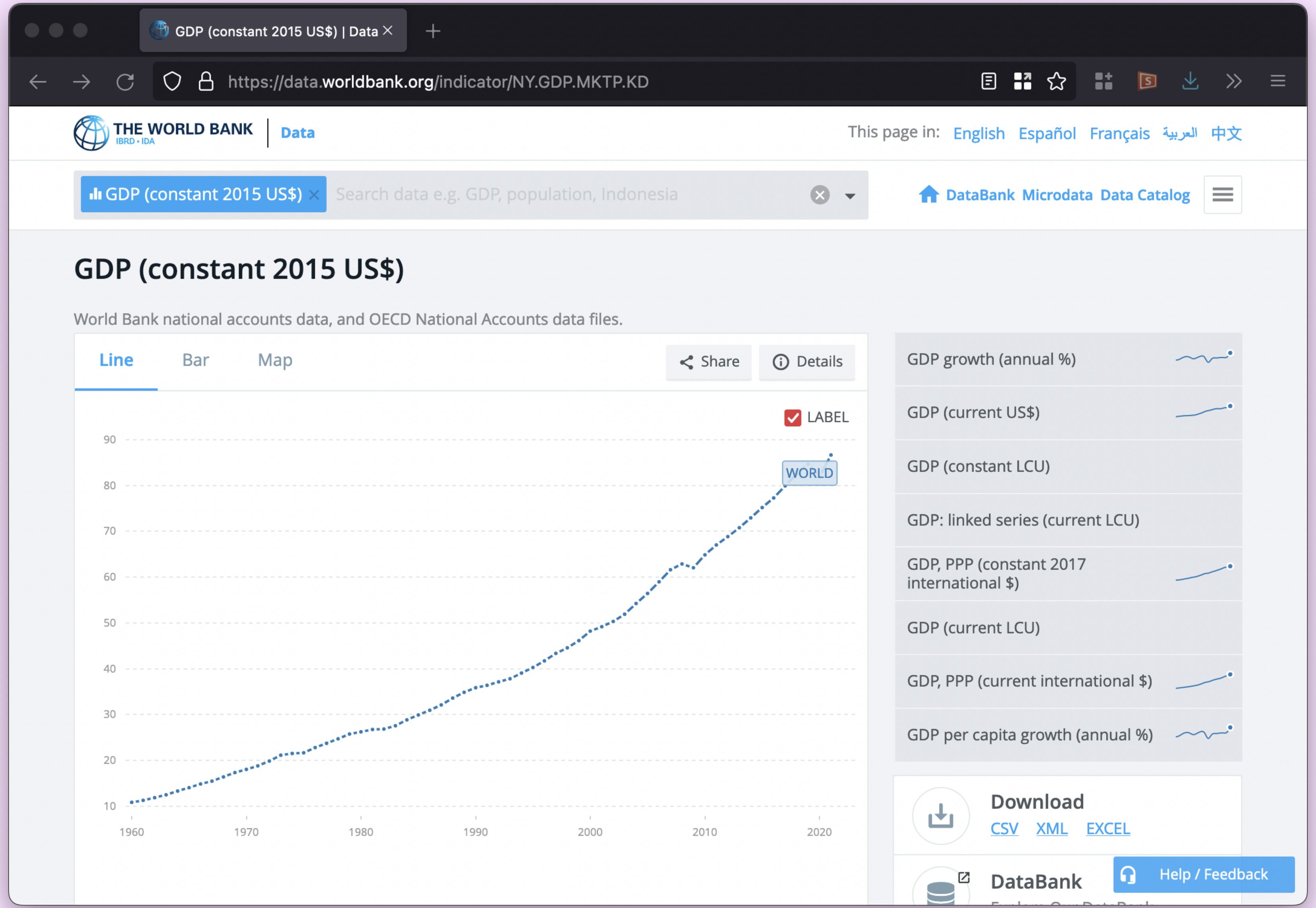
Plan for today

- **Working with raw data**
- **Pre-built API packages**
- **Accessing APIs yourself**
- **Scraping websites**

Working with raw data

Finding data online

- Data is everywhere online!
- Often provided as CSV or Excel files
- Read the file into R and do stuff with it

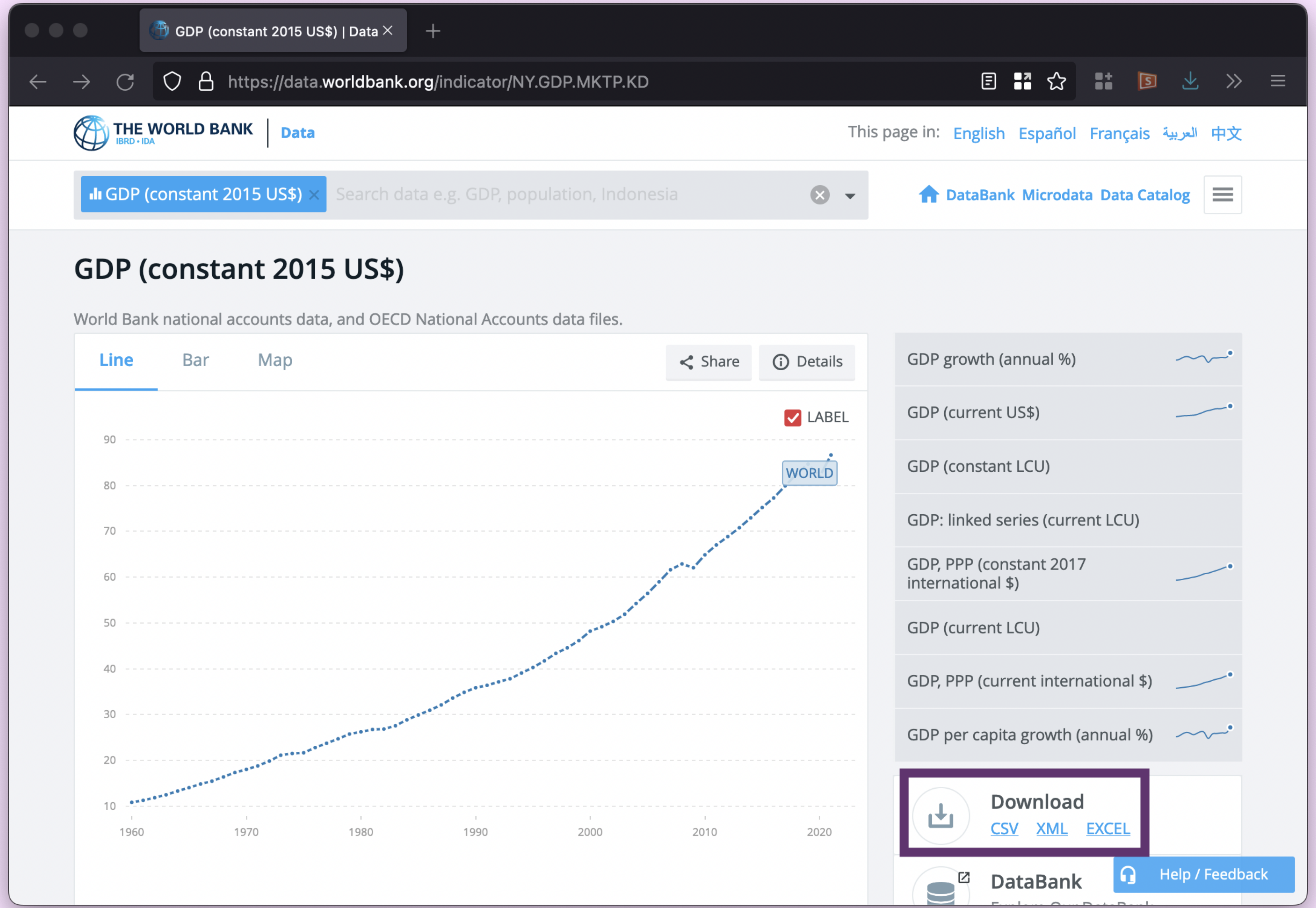


GDP (constant 2015 US$) | Data
https://data.worldbank.org/indicator/NY.GDP.MKTP.KD
THE WORLD BANK
IBRD • IDA
Data
This page in: English Español Français العربية 中文
GDP (constant 2015 US$)
Search data e.g. GDP, population, Indonesia
DataBank Microdata Data Catalog

GDP (constant 2015 US$)

World Bank national accounts data, and OECD National Accounts data files.

Line
Bar
Map
Share
Details
☑ LABEL
90
80
70
60
50
40
30
20
10
1960
1970
1980
1990
2000
2010
2020
WORLD
GDP growth (annual %)
GDP (current US$)
GDP (constant LCU)
GDP: linked series (current LCU)
GDP, PPP (constant 2017 international $)
GDP (current LCU)
GDP, PPP (current international $)
GDP per capita growth (annual %)
Download
CSV
XML
EXCEL
DataBank
Help / Feedback

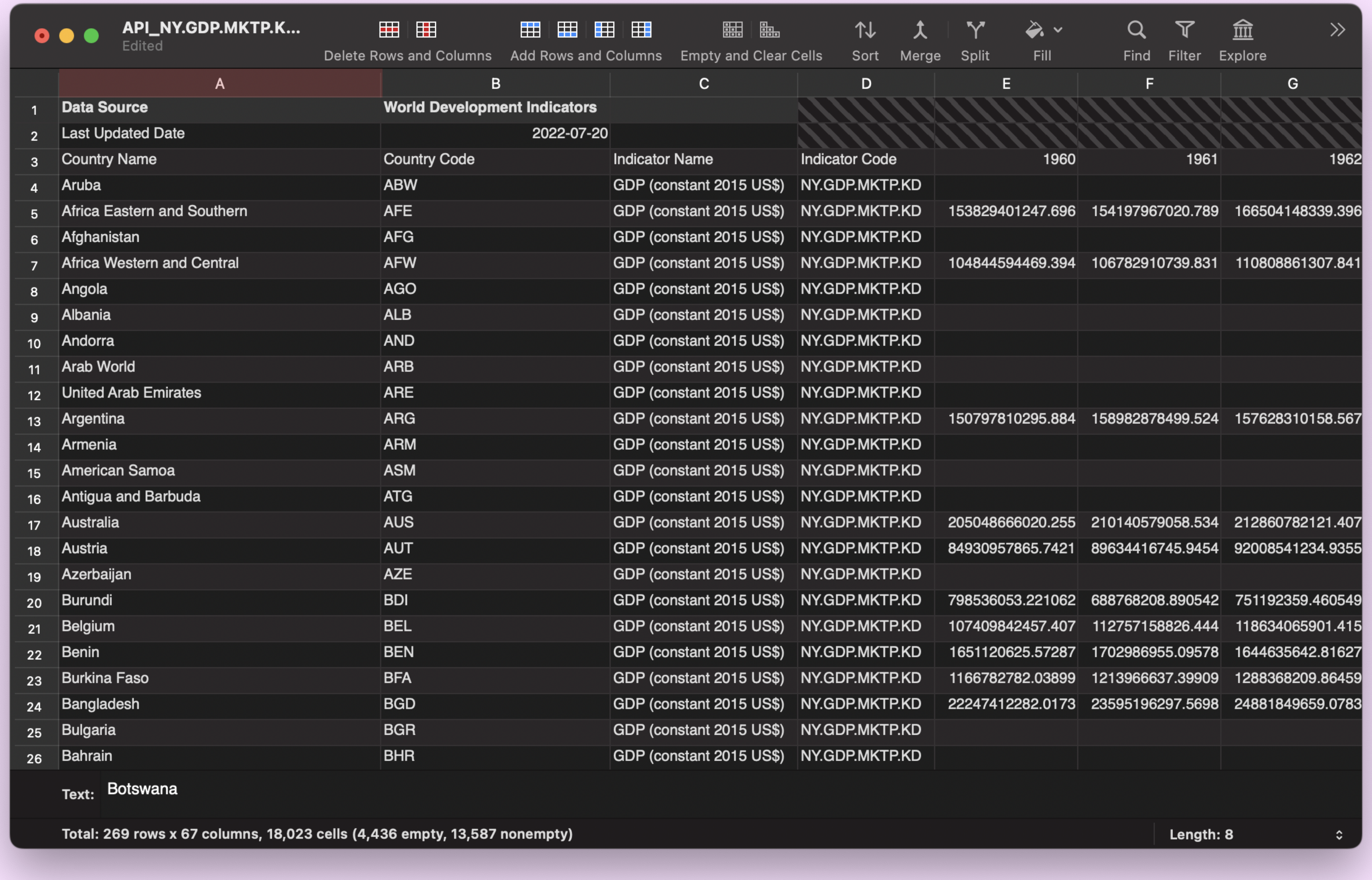


GDP (constant 2015 US$) | Data
https://data.worldbank.org/indicator/NY.GDP.MKTP.KD
THE WORLD BANK
IBRD • IDA
Data
This page in: English Español Français العربية 中文
GDP (constant 2015 US$)
Search data e.g. GDP, population, Indonesia
DataBank Microdata Data Catalog

GDP (constant 2015 US$)

World Bank national accounts data, and OECD National Accounts data files.

Line
Bar
Map
Share
Details
☑ LABEL
90
80
70
60
50
40
30
20
10
1960
1970
1980
1990
2000
2010
2020
WORLD
GDP growth (annual %)
GDP (current US$)
GDP (constant LCU)
GDP: linked series (current LCU)
GDP, PPP (constant 2017 international $)
GDP (current LCU)
GDP, PPP (current international $)
GDP per capita growth (annual %)
Download
CSV XML EXCEL
DataBank
Help / Feedback



	A	B	C	D	E	F	G
1	**Data Source**	**World Development Indicators**					
2	Last Updated Date	2022-07-20					
3	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962
4	Aruba	ABW	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
5	Africa Eastern and Southern	AFE	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	153829401247.696	154197967020.789	166504148339.396
6	Afghanistan	AFG	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
7	Africa Western and Central	AFW	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	104844594469.394	106782910739.831	110808861307.841
8	Angola	AGO	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
9	Albania	ALB	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
10	Andorra	AND	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
11	Arab World	ARB	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
12	United Arab Emirates	ARE	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
13	Argentina	ARG	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	150797810295.884	158982878499.524	157628310158.567
14	Armenia	ARM	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
15	American Samoa	ASM	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
16	Antigua and Barbuda	ATG	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
17	Australia	AUS	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	205048666020.255	210140579058.534	212860782121.407
18	Austria	AUT	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	84930957865.7421	89634416745.9454	92008541234.9355
19	Azerbaijan	AZE	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
20	Burundi	BDI	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	798536053.221062	688768208.890542	751192359.460549
21	Belgium	BEL	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	107409842457.407	112757158826.444	118634065901.415
22	Benin	BEN	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	1651120625.57287	1702986955.09578	1644635642.81627
23	Burkina Faso	BFA	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	1166782782.03899	1213966637.39909	1288368209.86459
24	Bangladesh	BGD	GDP (constant 2015 US$)	NY.GDP.MKTP.KD	22247412282.0173	23595196297.5698	24881849659.0783
25	Bulgaria	BGR	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			
26	Bahrain	BHR	GDP (constant 2015 US$)	NY.GDP.MKTP.KD			

Text: Botswana

Total: 269 rows x 67 columns, 18,023 cells (4,436 empty, 13,587 nonempty)

Length: 8

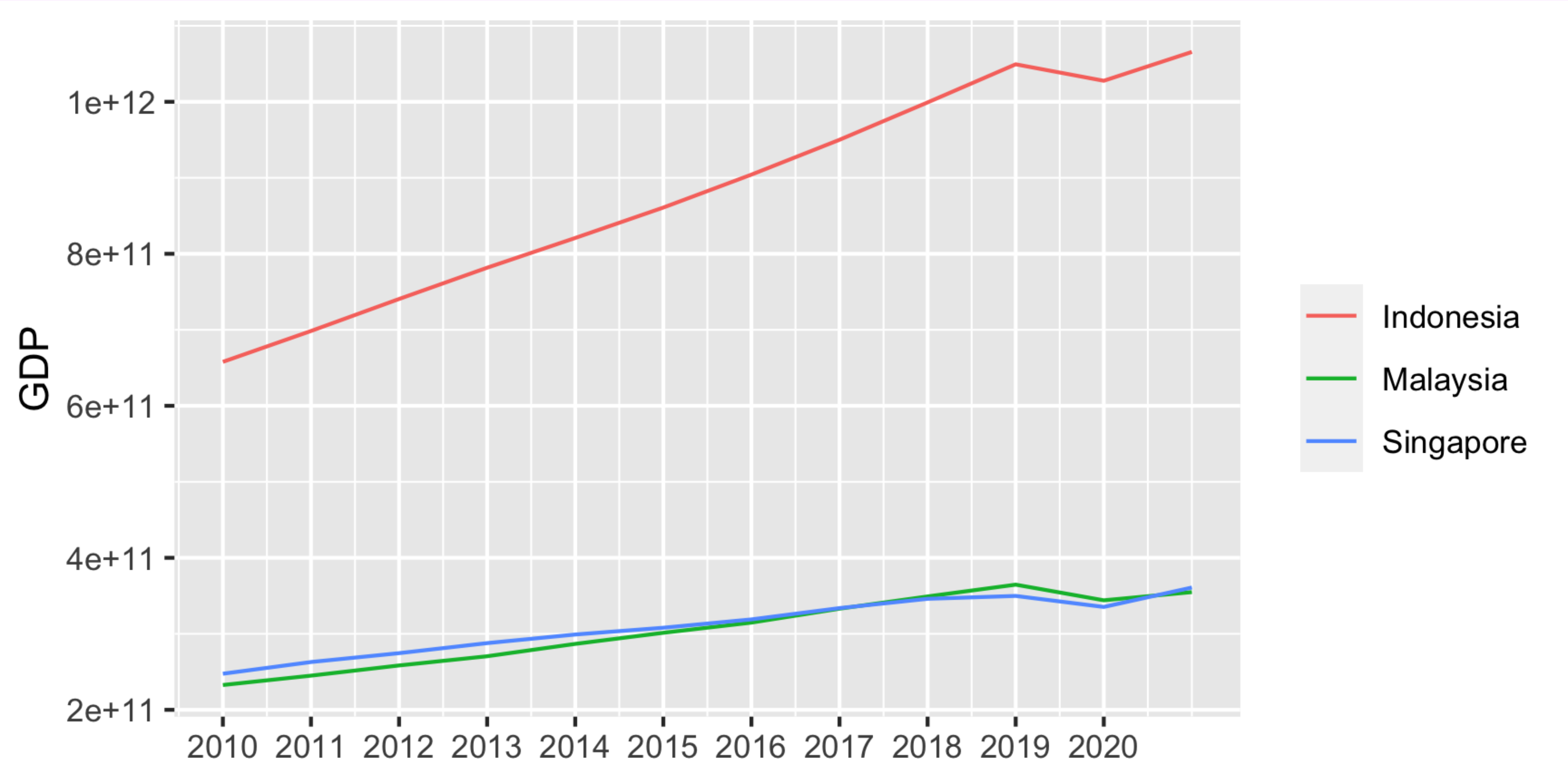
```r
library(tidyverse)

wdi_raw <- read_csv("API_NY.GDP.MKTP.KD_DS2_en_csv_v2_4487681.csv",
                    skip = 3)
wdi_raw
```

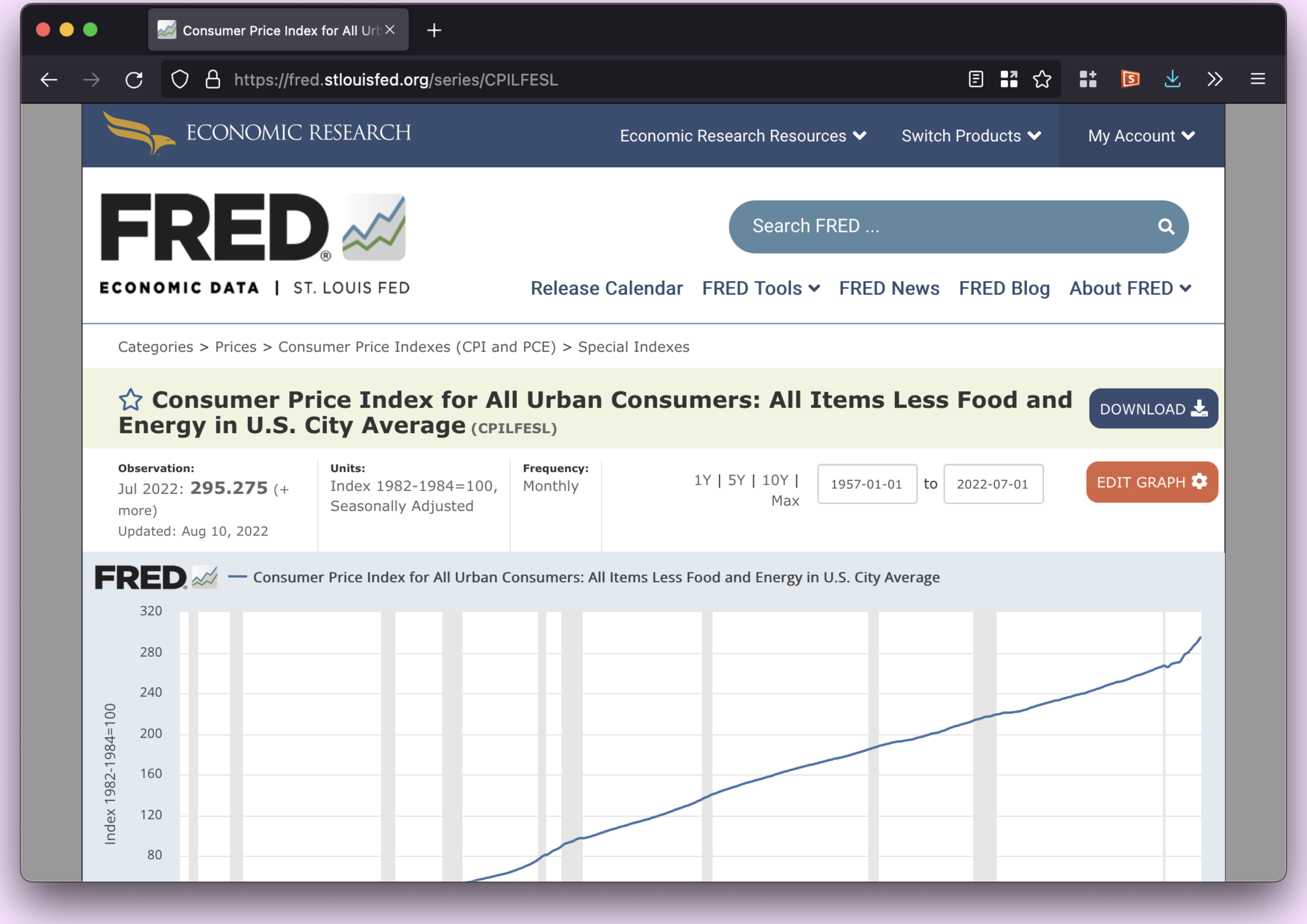
```
## # A tibble: 6 × 67
##   Country Nam…¹ Count…² Indic…³ Indic…⁴  `1960`  `1961`  `1962`  `1963`
##   <chr>         <chr>   <chr>   <chr>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 Aruba         ABW     GDP (c… NY.GDP… NA      NA      NA      NA     
## 2 Africa Easte… AFE     GDP (c… NY.GDP…  1.54e11 1.54e11 1.67e11 1.75e11
## 3 Afghanistan   AFG     GDP (c… NY.GDP… NA      NA      NA      NA     
## 4 Africa Weste… AFW     GDP (c… NY.GDP…  1.05e11 1.07e11 1.11e11 1.19e11
## 5 Angola        AGO     GDP (c… NY.GDP… NA      NA      NA      NA     
## 6 Albania       ALB     GDP (c… NY.GDP… NA      NA      NA      NA     
## # … with 59 more variables: `1964` <dbl>, `1965` <dbl>, `1966` <dbl>,
## #   `1967` <dbl>, `1968` <dbl>, `1969` <dbl>, `1970` <dbl>, `1971` <dbl>,
## #   `1972` <dbl>, `1973` <dbl>, `1974` <dbl>, `1975` <dbl>, `1976` <dbl>,
## #   `1977` <dbl>, `1978` <dbl>, `1979` <dbl>, `1980` <dbl>, `1981` <dbl>,
## #   `1982` <dbl>, `1983` <dbl>, `1984` <dbl>, `1985` <dbl>, `1986` <dbl>,
## #   `1987` <dbl>, `1988` <dbl>, `1989` <dbl>, `1990` <dbl>, `1991` <dbl>,
## #   `1992` <dbl>, `1993` <dbl>, `1994` <dbl>, `1995` <dbl>,
```

```r
wdi_clean <- wdi_raw %>%
  select(-`Indicator Name`, -`Indicator Code`) %>%
  pivot_longer(-c(`Country Name`, `Country Code`)) %>%
  mutate(year = as.numeric(name)) %>%
  filter(year >= 2010, `Country Code` %in% c("MYS", "IDN", "SGP"))

head(wdi_clean)
```

```
# A tibble: 6 × 5
  `Country Name` `Country Code` name          value  year
  <chr>          <chr>          <chr>         <dbl> <dbl>
1 Indonesia      IDN            2010  657835435591.  2010
2 Indonesia      IDN            2011  698422462409.  2011
3 Indonesia      IDN            2012  740537690665.  2012
4 Indonesia      IDN            2013  781691322851.  2013
5 Indonesia      IDN            2014  820828015499.  2014
6 Indonesia      IDN            2015  860854235065.  2015
```

```r
ggplot(wdi_clean, aes(x = year, y = value, color = `Country Name`)) +
  geom_line() +
  scale_x_continuous(breaks = 2010:2020) +
  labs(x = NULL, y = "GDP", color = NULL)
```





Categories > Prices > Consumer Price Indexes (CPI and PCE) > Special Indexes

☆ Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average (CPILFESL)

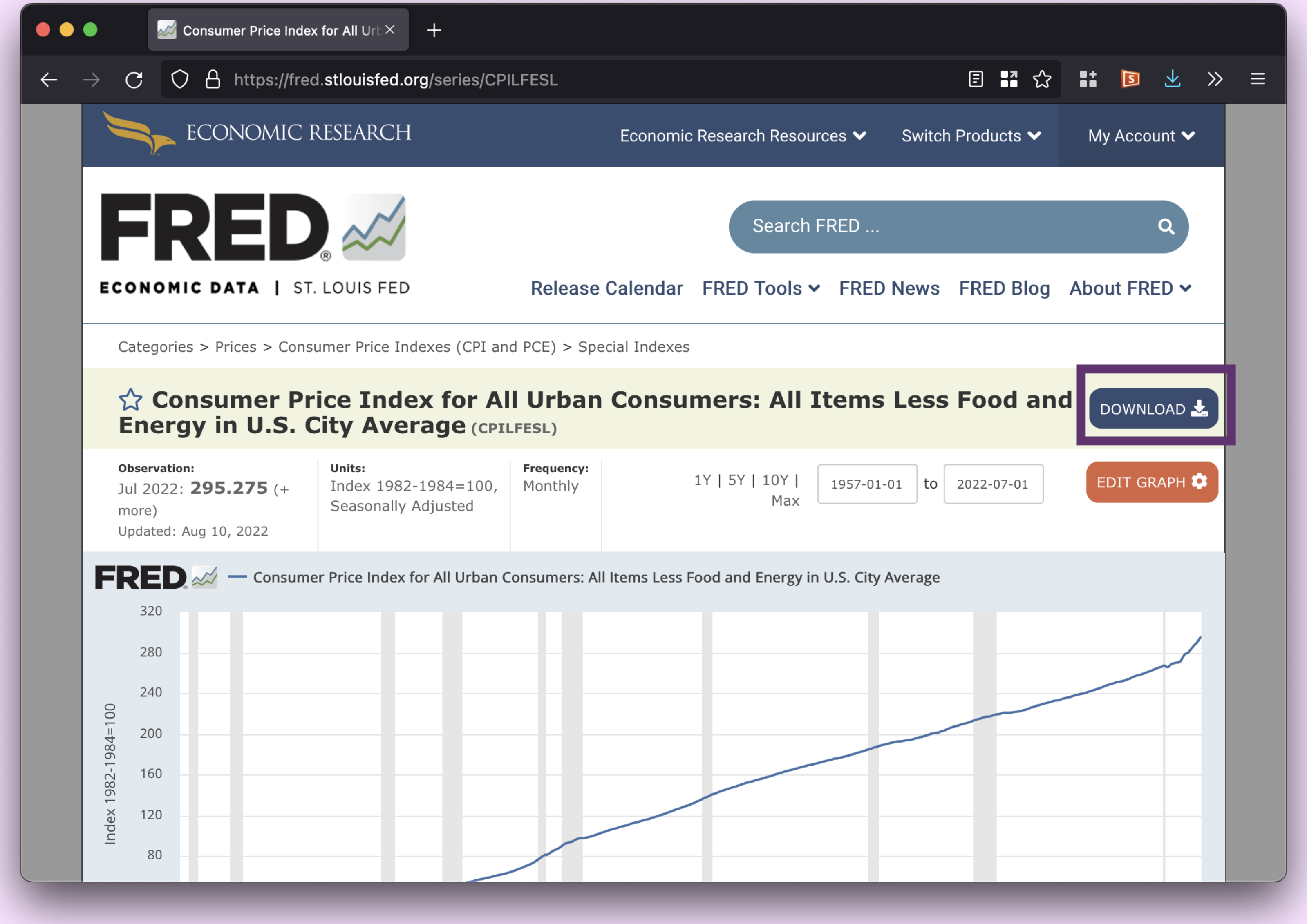
Observation:
Jul 2022: **295.275** (+ more)
Updated: Aug 10, 2022

Units:
Index 1982-1984=100, Seasonally Adjusted

Frequency:
Monthly

1957-01-01 to 2022-07-01

— Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average



Consumer Price Index for All Urb

https://fred.stlouisfed.org/series/CPILFESL

Categories > Prices > Consumer Price Indexes (CPI and PCE) > Special Indexes

☆ Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average (CPILFESL)

Observation: Jul 2022: **295.275** (+ more)
Updated: Aug 10, 2022

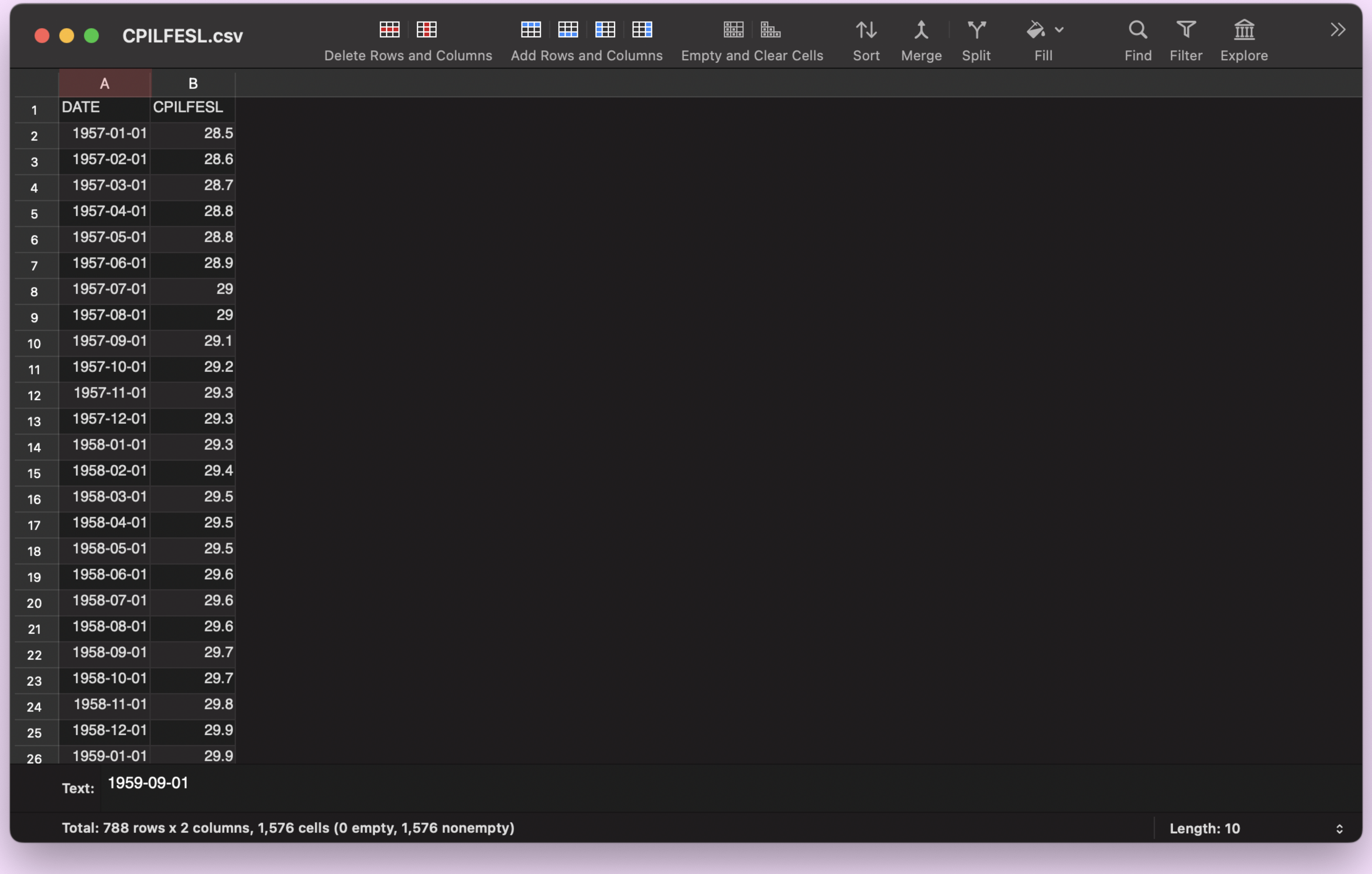
Units: Index 1982-1984=100, Seasonally Adjusted

Frequency: Monthly

1Y | 5Y | 10Y | Max

1957-01-01 to 2022-07-01

EDIT GRAPH
FRED
Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average
320
280
240
200
160
120
80
Index 1982-1984=100



	A	B
1	DATE	CPILFESL
2	1957-01-01	28.5
3	1957-02-01	28.6
4	1957-03-01	28.7
5	1957-04-01	28.8
6	1957-05-01	28.8
7	1957-06-01	28.9
8	1957-07-01	29
9	1957-08-01	29
10	1957-09-01	29.1
11	1957-10-01	29.2
12	1957-11-01	29.3
13	1957-12-01	29.3
14	1958-01-01	29.3
15	1958-02-01	29.4
16	1958-03-01	29.5
17	1958-04-01	29.5
18	1958-05-01	29.5
19	1958-06-01	29.6
20	1958-07-01	29.6
21	1958-08-01	29.6
22	1958-09-01	29.7
23	1958-10-01	29.7
24	1958-11-01	29.8
25	1958-12-01	29.9
26	1959-01-01	29.9

Text: 1959-09-01

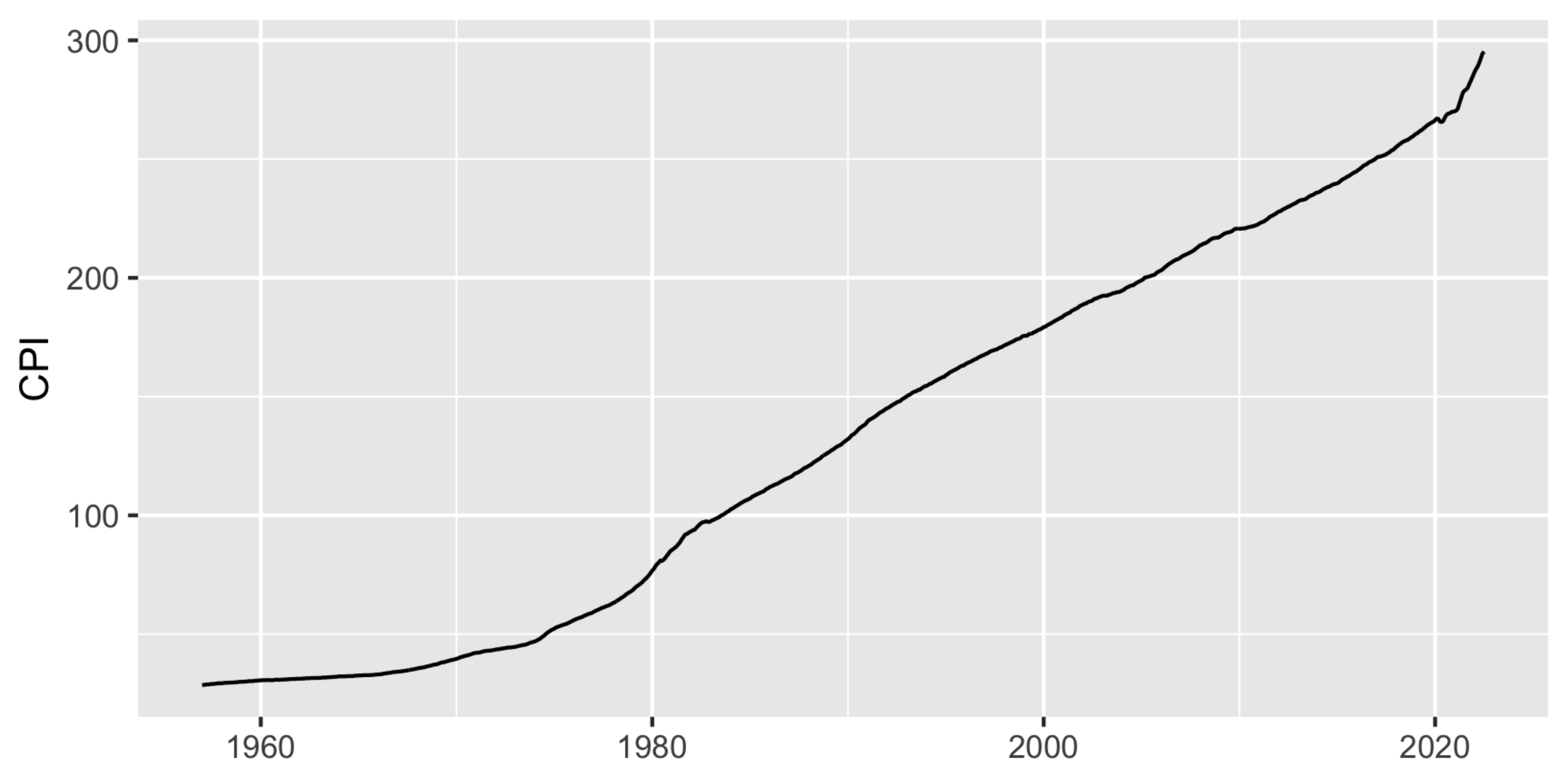
Total: 788 rows x 2 columns, 1,576 cells (0 empty, 1,576 nonempty)

Length: 10

```r
library(tidyverse)

fred_raw <- read_csv("CPILFESL.csv")
fred_raw
```

```
## # A tibble: 6 × 2
##   DATE       CPILFESL
##   <date>        <dbl>
## 1 1957-01-01     28.5
## 2 1957-02-01     28.6
## 3 1957-03-01     28.7
## 4 1957-04-01     28.8
## 5 1957-05-01     28.8
## 6 1957-06-01     28.9
```

```r
ggplot(fred_raw, aes(x = DATE, y = CPILFESL)) +
  geom_line() +
  labs(x = NULL, y = "CPI")
```

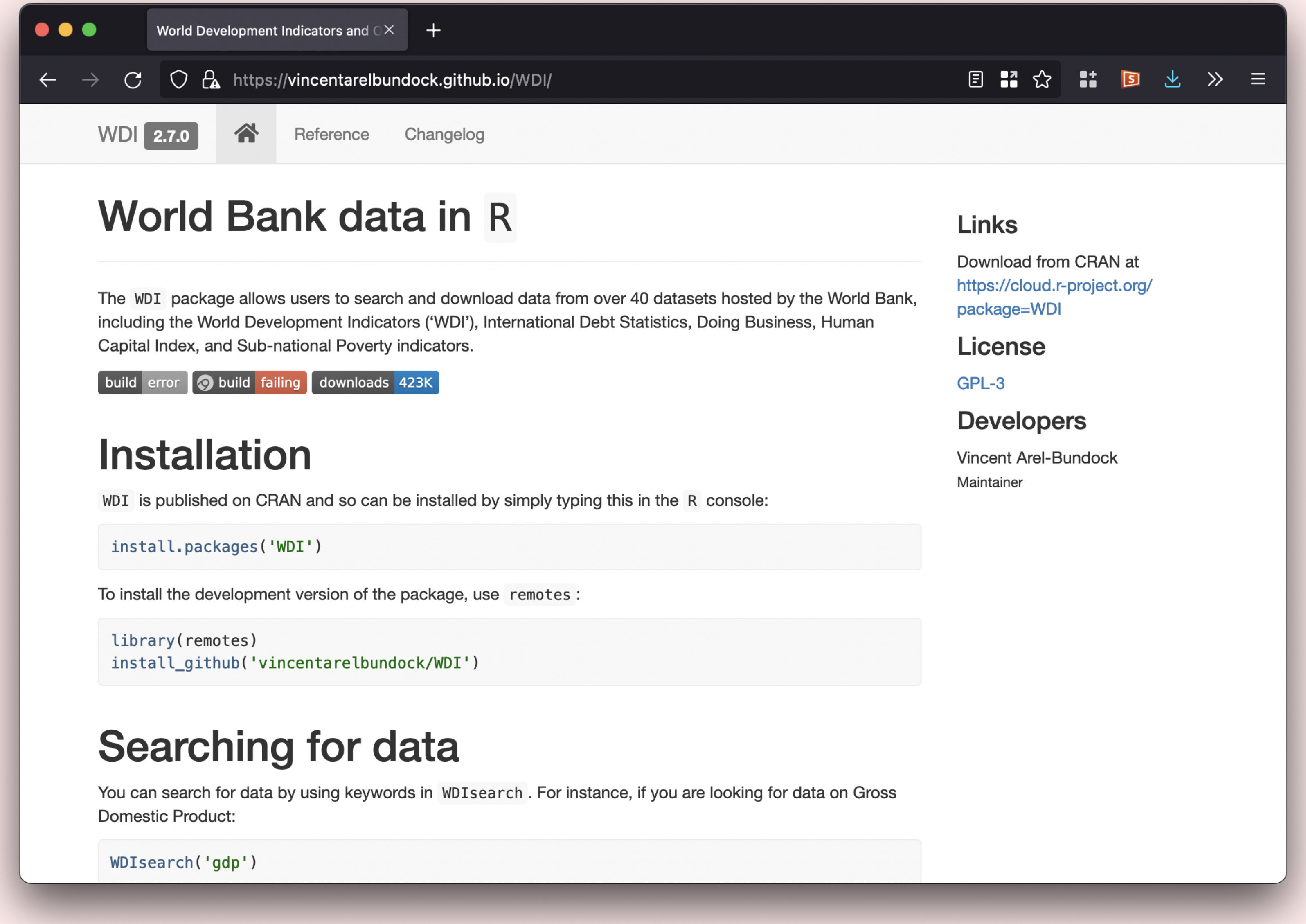


Your turn!

Pre-built API packages

Avoid extra work!

- Try to avoid downloading raw data files whenever possible!
- Many data-focused websites provide more direct access to data through an **application programming interface**, or **API**
- Big list of public data APIs


World Development Indicators and C
https://vincentarelbundock.github.io/WDI/
WDI 2.7.0
Reference
Changelog

World Bank data in R

The WDI package allows users to search and download data from over 40 datasets hosted by the World Bank, including the World Development Indicators ('WDI'), International Debt Statistics, Doing Business, Human Capital Index, and Sub-national Poverty indicators.

build error
build failing
downloads 423K

Installation

WDI is published on CRAN and so can be installed by simply typing this in the R console:

```
install.packages('WDI')
```

To install the development version of the package, use remotes :

```
library(remotes)
install_github('vincentarelbundock/WDI')
```

Searching for data

You can search for data by using keywords in WDIsearch . For instance, if you are looking for data on Gross Domestic Product:

```
WDIsearch('gdp')
```

Links

Download from CRAN at https://cloud.r-project.org/package=WDI

License

GPL-3

Developers

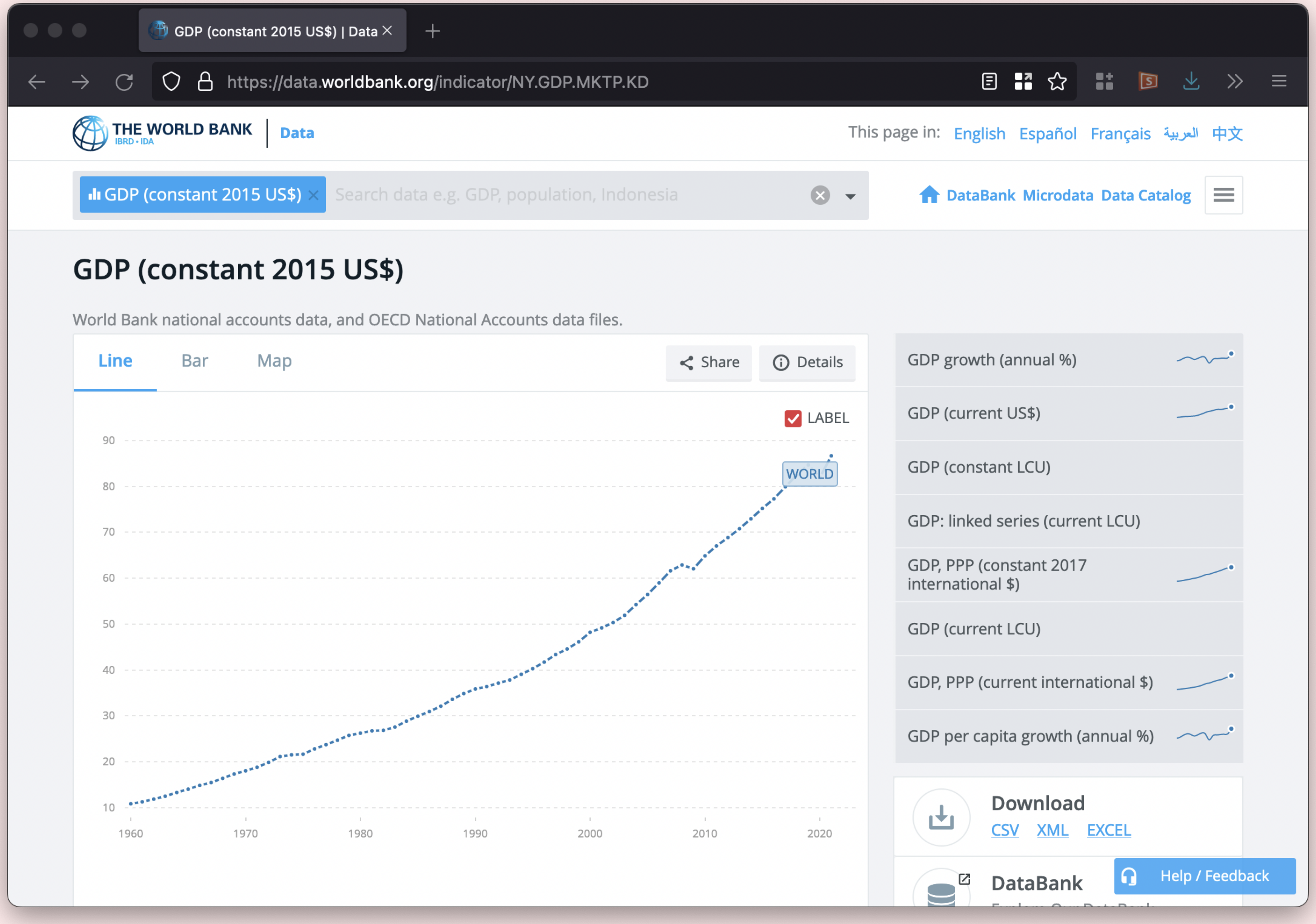
Vincent Arel-Bundock
Maintainer

```r
library(WDI)

data <- WDI(country = ___,
            indicator = ___,
            extra = ___,
            start = ___,
            end = ___)
```

```r
library(WDI)

data <- WDI(country = "MY",
            indicator = ___,
            extra = ___,
            start = ___,
            end = ___)
```

```r
library(WDI)

data <- WDI(country = c("MY", "ID", "SG"),
            indicator = ___,
            extra = ___,
            start = ___,
            end = ___)
```

```r
library(WDI)

data <- WDI(country = c("all"),
            indicator = ___,
            extra = ___,
            start = ___,
            end = ___)
```



GDP (constant 2015 US$) | Data

https://data.worldbank.org/indicator/NY.GDP.MKTP.KD

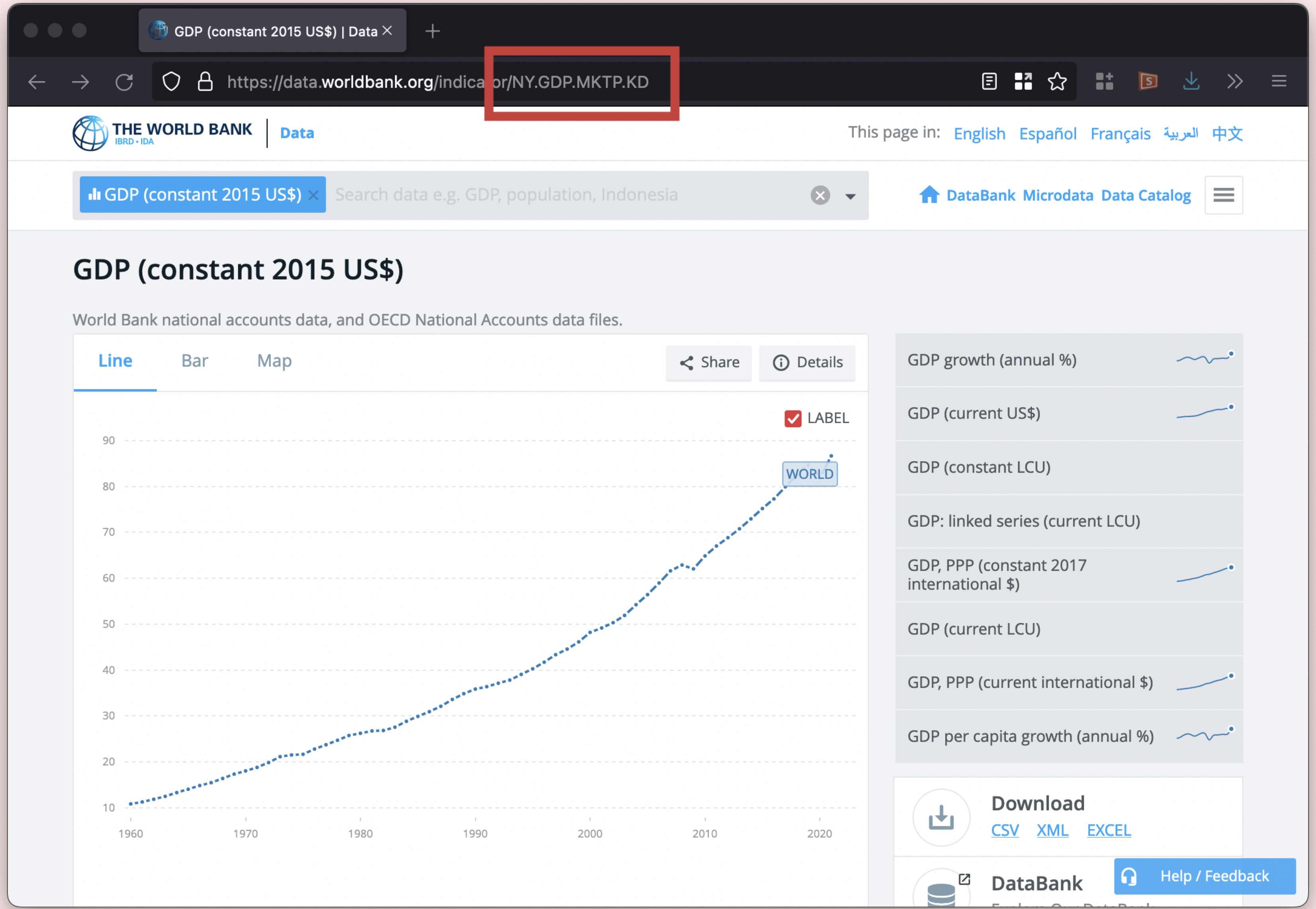
GDP (constant 2015 US$)

World Bank national accounts data, and OECD National Accounts data files.

- GDP growth (annual %)
- GDP (current US$)
- GDP (constant LCU)
- GDP: linked series (current LCU)
- GDP, PPP (constant 2017 international $)
- GDP (current LCU)
- GDP, PPP (current international $)
- GDP per capita growth (annual %)

Download
CSV XML EXCEL

DataBank


GDP (constant 2015 US$) | Data
https://data.worldbank.org/indicator/NY.GDP.MKTP.KD
THE WORLD BANK
IBRD • IDA
Data
This page in: English Español Français العربية 中文
GDP (constant 2015 US$)
Search data e.g. GDP, population, Indonesia
DataBank Microdata Data Catalog
Line
Bar
Map
Share
Details
☑ LABEL
90
80
70
60
50
40
30
20
10
1960
1970
1980
1990
2000
2010
2020
WORLD
Help / Feedback

GDP (constant 2015 US$)

World Bank national accounts data, and OECD National Accounts data files.

- GDP growth (annual %)
- GDP (current US$)
- GDP (constant LCU)
- GDP: linked series (current LCU)
- GDP, PPP (constant 2017 international $)
- GDP (current LCU)
- GDP, PPP (current international $)
- GDP per capita growth (annual %)

Download

CSV XML EXCEL

DataBank

```r
library(WDI)

data <- WDI(country = c("MY", "ID", "SG"),
            indicator = "NY.GDP.MKTP.KD",
            extra = ___,
            start = ___,
            end = ___)
```

```r
library(WDI)

data <- WDI(country = c("MY", "ID", "SG"),
            indicator = c("NY.GDP.MKTP.KD",      # GDP, 2015 USD
                          "NY.GDP.MKTP.KD.ZG"),  # GDP growth, annual %
            extra = ___,
            start = ___,
            end = ___)
```

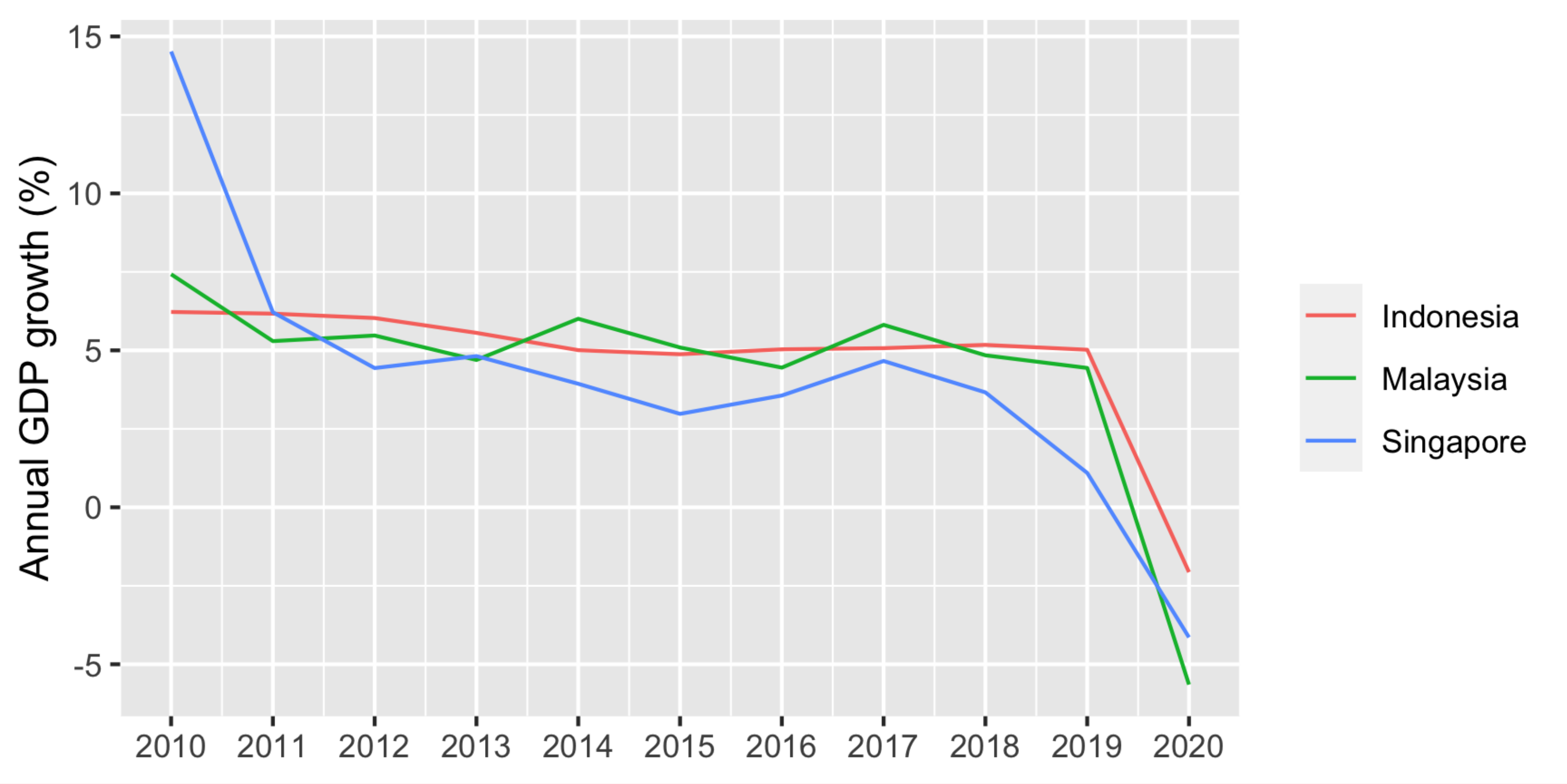
```r
library(WDI)

data <- WDI(country = c("MY", "ID", "SG"),
            indicator = c("NY.GDP.MKTP.KD",      # GDP, 2015 USD
                          "NY.GDP.MKTP.KD.ZG"),  # GDP growth, annual %
            extra = TRUE,  # Population, region, and other helpful columns
            start = ___,
            end = ___)
```

```r
library(WDI)

data <- WDI(country = c("MY", "ID", "SG"),
            indicator = c("NY.GDP.MKTP.KD",       # GDP, 2015 USD
                          "NY.GDP.MKTP.KD.ZG"),  # GDP growth, annual %
            extra = TRUE,  # Population, region, and other helpful columns
            start = 2010,
            end = 2020)
```

```r
library(WDI)

data <- WDI(country = c("MY", "ID", "SG"),
            indicator = c("NY.GDP.MKTP.KD",        # GDP, 2015 USD
                          "NY.GDP.MKTP.KD.ZG"),  # GDP growth, annual %
            extra = TRUE,  # Population, region, and other helpful columns
            start = 2010,
            end = 2020)
head(data)
```

```
## # A tibble: 6 × 12
##   iso2c country   year NY.GD…¹ NY.GD…² iso3c region capital longi…³ latit…⁴
##   <chr> <chr>    <int>   <dbl>   <dbl> <chr> <chr>  <chr>   <chr>   <chr>
## 1 ID    Indones…  2010 6.58e11    6.22 IDN   East … Jakarta 106.83  -6.197…
## 2 ID    Indones…  2011 6.98e11    6.17 IDN   East … Jakarta 106.83  -6.197…
## 3 ID    Indones…  2012 7.41e11    6.03 IDN   East … Jakarta 106.83  -6.197…
## 4 ID    Indones…  2013 7.82e11    5.56 IDN   East … Jakarta 106.83  -6.197…
## 5 ID    Indones…  2014 8.21e11    5.01 IDN   East … Jakarta 106.83  -6.197…
## 6 ID    Indones…  2015 8.61e11    4.88 IDN   East … Jakarta 106.83  -6.197…
## # … with 2 more variables: income <chr>, lending <chr>, and abbreviated
## #   variable names ¹NY.GDP.MKTP.KD, ²NY.GDP.MKTP.KD.ZG, ³longitude,
## #   ⁴latitude
```

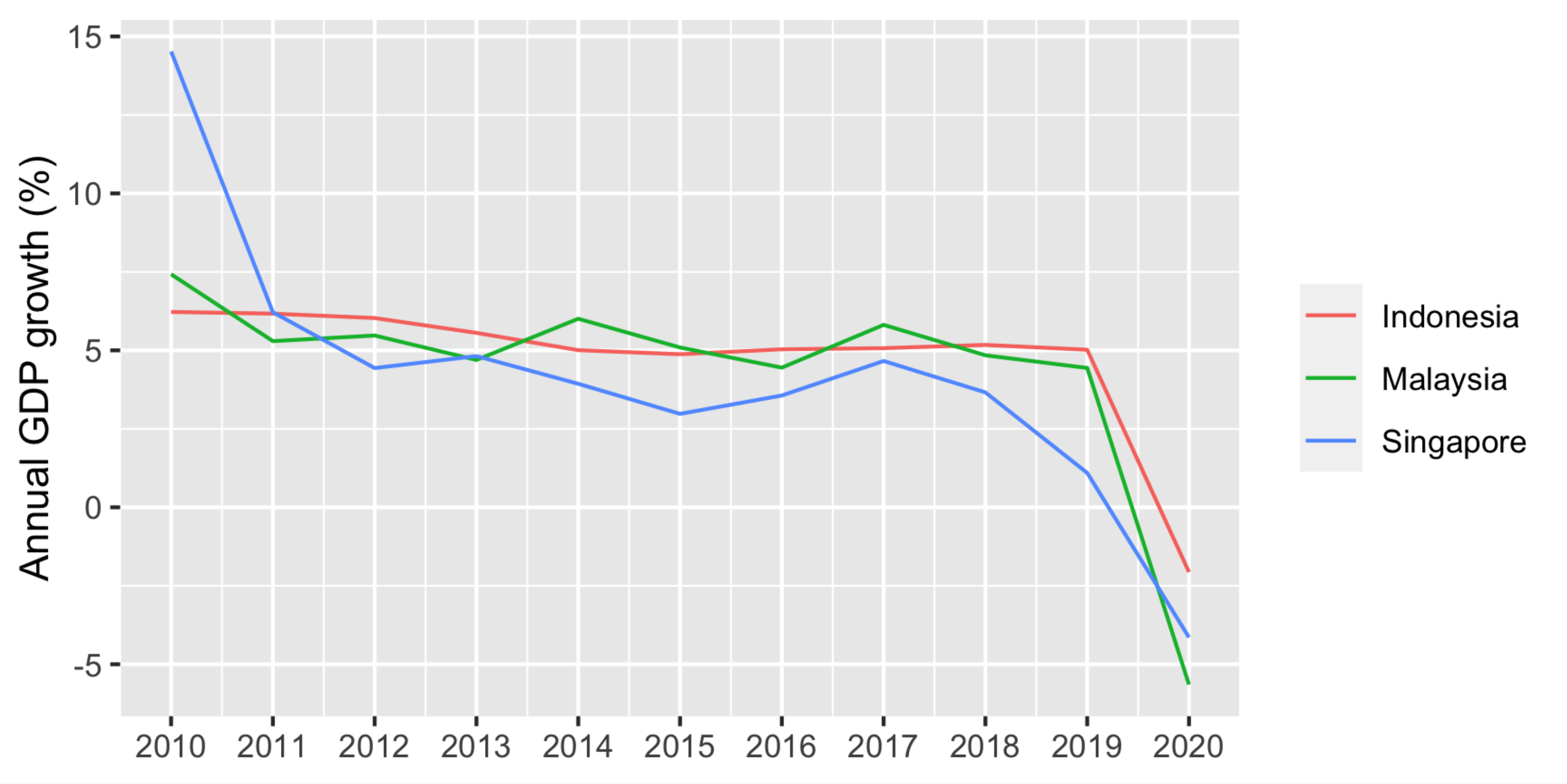
```r
library(ggplot2)

ggplot(data, aes(x = year, y = NY.GDP.MKTP.KD.ZG, color = country)) +
  geom_line() +
  scale_x_continuous(breaks = 2010:2020) +
  labs(x = NULL, y = "Annual GDP growth (%)", color = NULL)
```



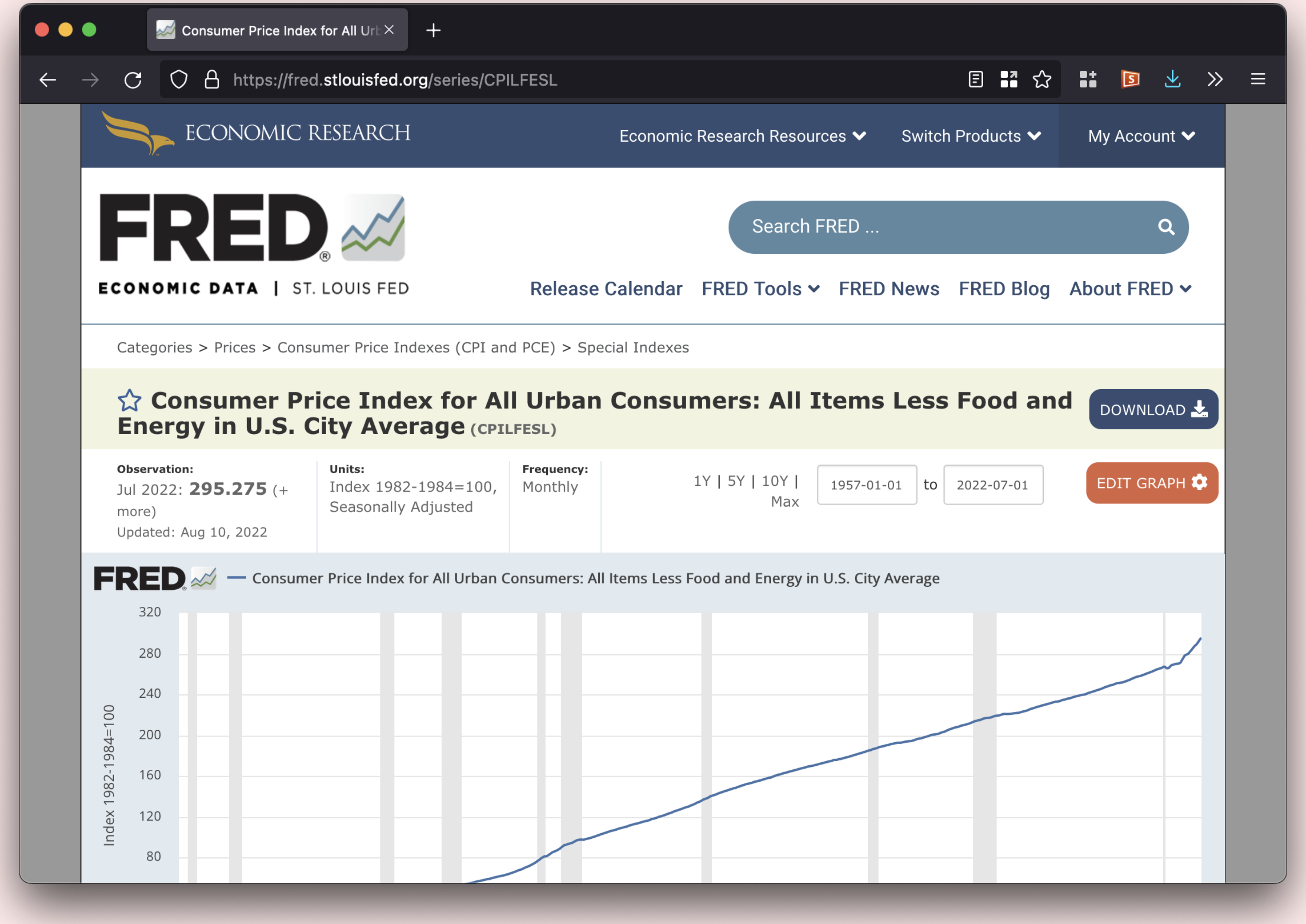
```r
library(WDI)

data <- WDI(country = c("MY", "ID", "SG"),
            indicator = c(gdp = "NY.GDP.MKTP.KD",      # GDP, 2015 USD
                          gdp_growth = "NY.GDP.MKTP.KD.ZG"),  # GDP growth,
            extra = TRUE,  # Population, region, and other helpful columns
            start = 2010,
            end = 2020)
head(data)
```

```
## # A tibble: 6 × 6
##   iso2c country    year           gdp gdp_growth others
##   <chr> <chr>     <int>         <dbl>      <dbl> <chr>
## 1 ID    Indonesia  2010 657835435591.       6.22 ...
## 2 ID    Indonesia  2011 698422462409.       6.17 ...
## 3 ID    Indonesia  2012 740537690665.       6.03 ...
## 4 ID    Indonesia  2013 781691322851.       5.56 ...
## 5 ID    Indonesia  2014 820828015499.       5.01 ...
## 6 ID    Indonesia  2015 860854235065.       4.88 ...
```

```r
library(ggplot2)

ggplot(data, aes(x = year, y = gdp_growth, color = country)) +
  geom_line() +
  scale_x_continuous(breaks = 2010:2020) +
  labs(x = NULL, y = "Annual GDP growth (%)", color = NULL)
```



Your turn!



Categories > Prices > Consumer Price Indexes (CPI and PCE) > Special Indexes

Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average (CPILFESL)

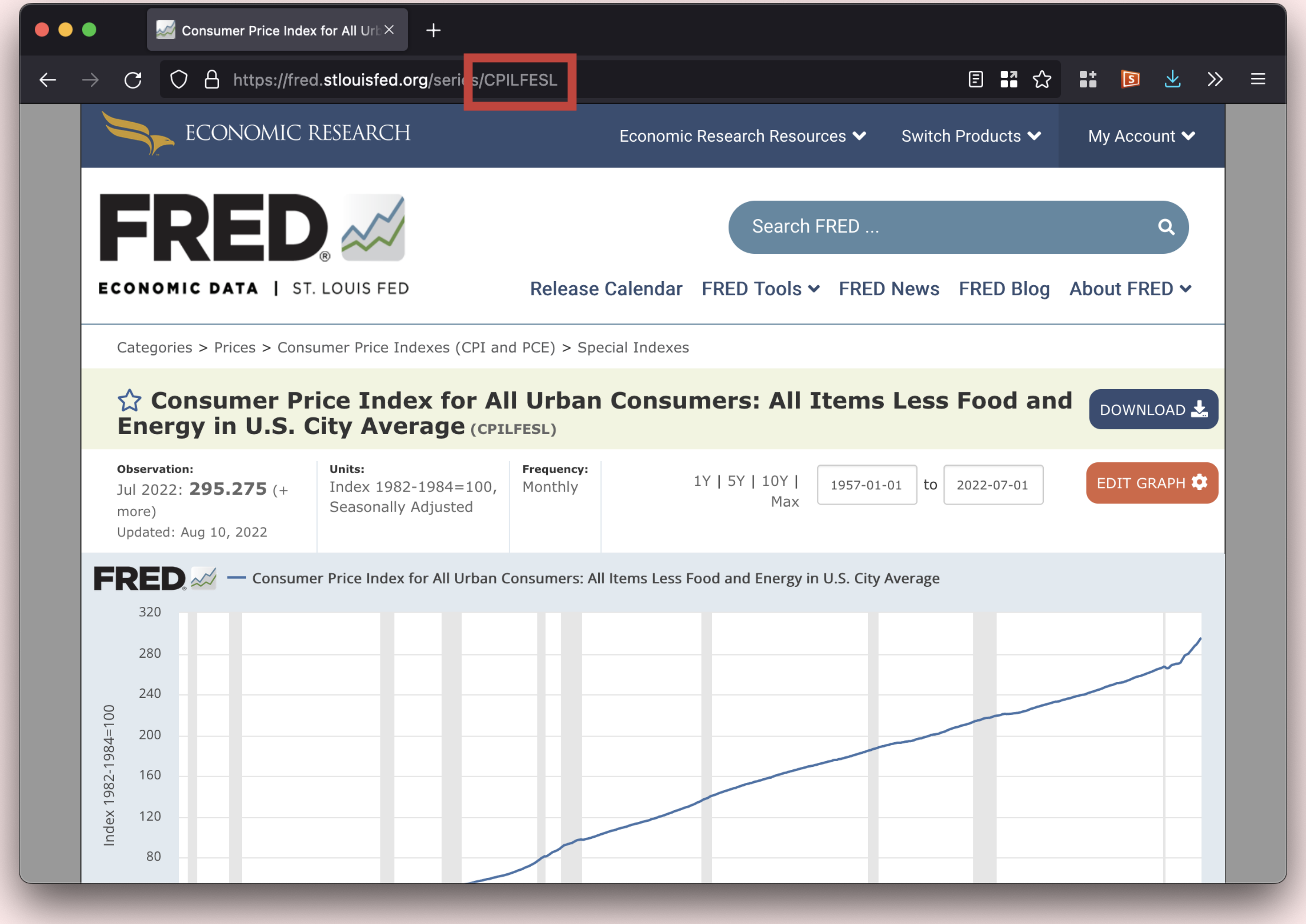
Observation:
Jul 2022: **295.275** (+ more)
Updated: Aug 10, 2022

Units:
Index 1982-1984=100, Seasonally Adjusted

Frequency:
Monthly

1957-01-01 to 2022-07-01

— Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average



Categories > Prices > Consumer Price Indexes (CPI and PCE) > Special Indexes

☆ Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average (CPILFESL)

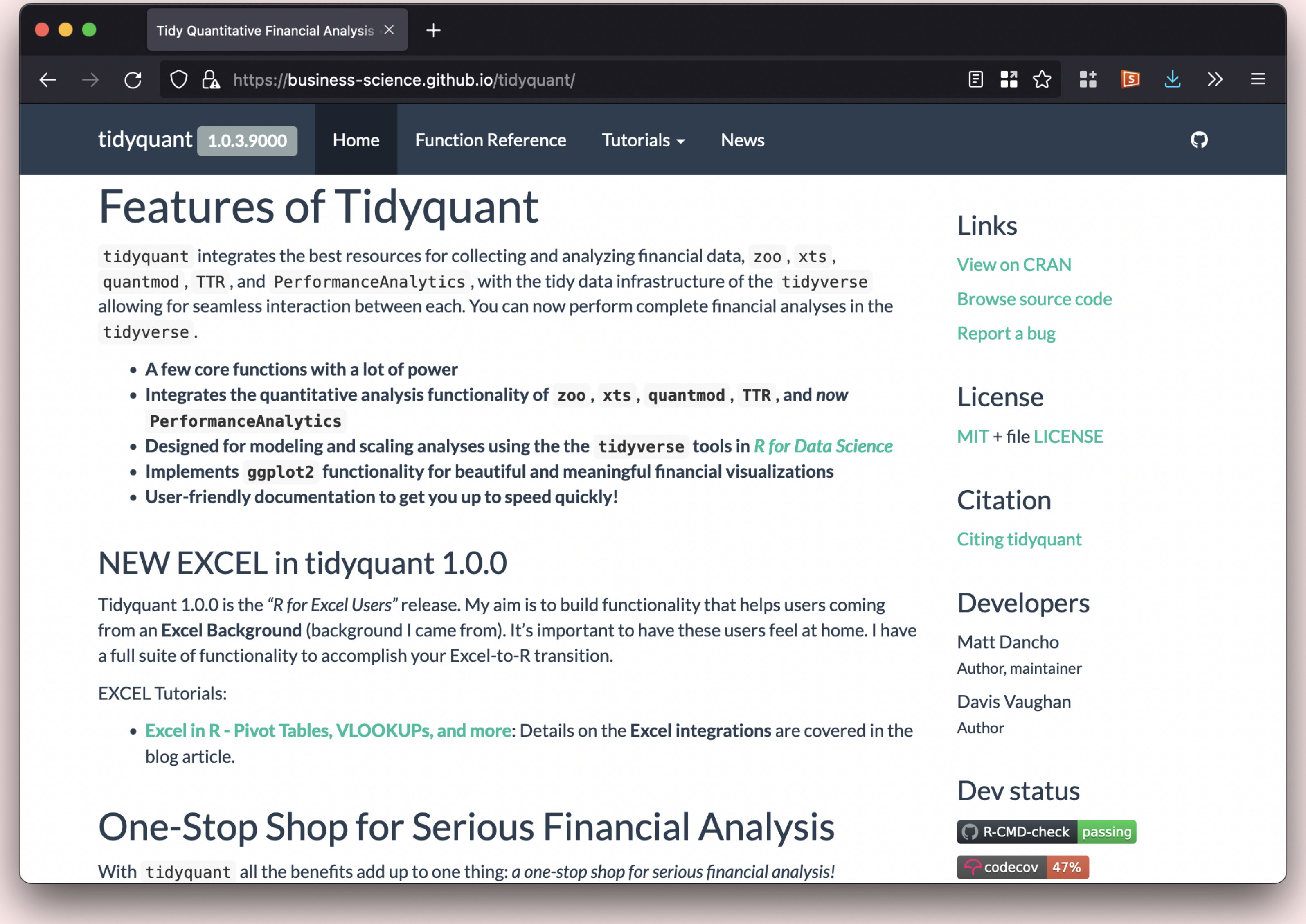
Observation:
Jul 2022: **295.275** (+ more)
Updated: Aug 10, 2022

Units:
Index 1982-1984=100,
Seasonally Adjusted

Frequency:
Monthly

1957-01-01 to 2022-07-01

FRED — Consumer Price Index for All Urban Consumers: All Items Less Food and Energy in U.S. City Average


Tidy Quantitative Financial Analysis
https://business-science.github.io/tidyquant/
tidyquant 1.0.3.9000
Home
Function Reference
Tutorials
News

Features of Tidyquant

`tidyquant` integrates the best resources for collecting and analyzing financial data, `zoo`, `xts`, `quantmod`, `TTR`, and `PerformanceAnalytics`, with the tidy data infrastructure of the `tidyverse` allowing for seamless interaction between each. You can now perform complete financial analyses in the `tidyverse`.

- **A few core functions with a lot of power**
- **Integrates the quantitative analysis functionality of `zoo`, `xts`, `quantmod`, `TTR`, and *now* `PerformanceAnalytics`**
- **Designed for modeling and scaling analyses using the the `tidyverse` tools in *R for Data Science***
- **Implements `ggplot2` functionality for beautiful and meaningful financial visualizations**
- **User-friendly documentation to get you up to speed quickly!**

NEW EXCEL in tidyquant 1.0.0

Tidyquant 1.0.0 is the *"R for Excel Users"* release. My aim is to build functionality that helps users coming from an **Excel Background** (background I came from). It's important to have these users feel at home. I have a full suite of functionality to accomplish your Excel-to-R transition.

EXCEL Tutorials:

- **Excel in R - Pivot Tables, VLOOKUPs, and more**: Details on the **Excel integrations** are covered in the blog article.

One-Stop Shop for Serious Financial Analysis

With `tidyquant` all the benefits add up to one thing: *a one-stop shop for serious financial analysis!*

Links

View on CRAN

Browse source code

Report a bug

License

MIT + file LICENSE

Citation

Citing tidyquant

Developers

Matt Dancho
Author, maintainer

Davis Vaughan
Author

Dev status

R-CMD-check passing

codecov 47%

```r
library(tidyquant)

data <- tq_get(x = ___,
               get = ___,
               from = ___,
               to = ___)
```

```r
library(tidyquant)

data <- tq_get(x = c("CPILFESL",  # CPI
                     "RSXFSN",  # Advance retail sales
                     "USREC"),  # US recessions
               get = ___,
               from = ___,
               to = ___)
```

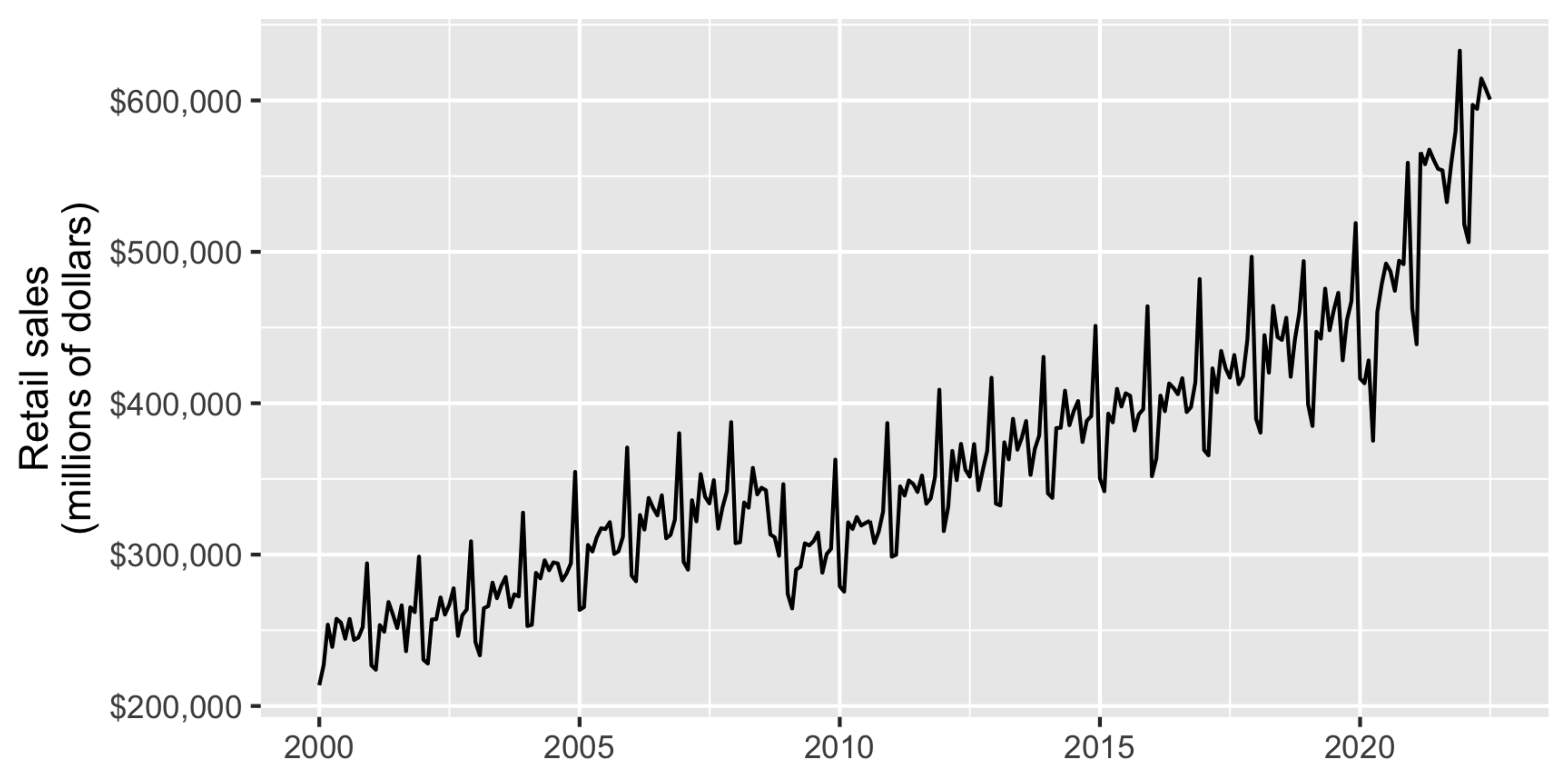
```r
library(tidyquant)

data <- tq_get(x = c("CPILFESL",  # CPI
                     "RSXFSN",  # Advance retail sales
                     "USREC"),  # US recessions
               get = "economic.data",  # Use FRED
               from = ___,
               to = ___)
```

```r
library(tidyquant)

data <- tq_get(x = c("CPILFESL",  # CPI
                     "RSXFSN",  # Advance retail sales
                     "USREC"),  # US recessions
               get = "economic.data",  # Use FRED
               from = "2000-01-01",
               to = "2022-09-01")
```

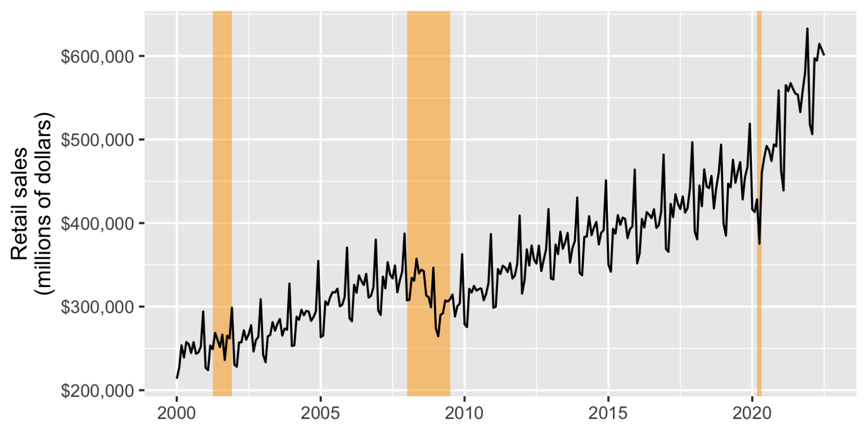
```r
library(tidyquant)

data <- tq_get(x = c("CPILFESL",  # CPI
                     "RSXFSN",  # Advance retail sales
                     "USREC"),  # US recessions
               get = "economic.data",  # Use FRED
               from = "2000-01-01",
               to = "2022-09-01")
head(data)
```

```
## # A tibble: 6 × 3
##   symbol   date       price
##   <chr>    <date>     <dbl>
## 1 CPILFESL 2000-01-01  179.
## 2 CPILFESL 2000-02-01  179.
## 3 CPILFESL 2000-03-01  180
## 4 CPILFESL 2000-04-01  180.
## 5 CPILFESL 2000-05-01  181.
## 6 CPILFESL 2000-06-01  181.
```

```r
library(tidyverse)
retail <- data %>% filter(symbol == "RSXFSN")

ggplot(retail, aes(x = date, y = price)) +
  geom_line() +
  scale_y_continuous(labels = scales::label_dollar()) +
  labs(x = NULL, y = "Retail sales\n(millions of dollars)")
```



```r
recessions_start_end <- data %>%
  filter(symbol == "USREC") %>%
  mutate(recession_change = price - lag(price)) %>%
  filter(recession_change != 0)

recessions <- tibble(start = filter(recessions_start_end, recession_change
                     end = filter(recessions_start_end, recession_change ==

ggplot(retail, aes(x = date, y = price)) +
  geom_rect(data = recessions,
            aes(xmin = start, xmax = end, ymin = -Inf, ymax = Inf),
            inherit.aes = FALSE, fill = "orange", alpha = 0.5) +
  geom_line() +
  scale_y_continuous(labels = scales::label_dollar()) +
  labs(x = NULL, y = "Retail sales\n(millions of dollars)")
```



Your turn!

Accessing APIs yourself

What if there's an API but no pre-built package?

- You can still use the API!
- You need to write the code to access the API using the {httr} R package

Sending data to websites

GET

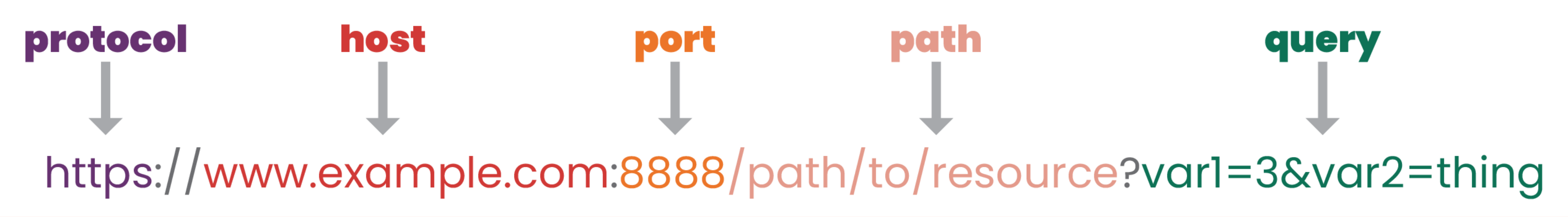
Data sent to server via URLs with parameters

Parameters all visible in the URL

POST

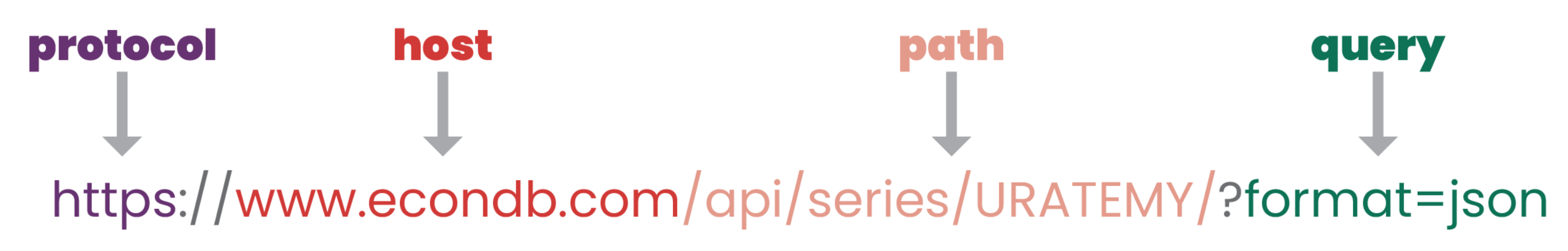
Data sent to server via invisible request

Forms with usernames and passwords



protocol host port path query

https://www.example.com:8888/path/to/resource?var1=3&var2=thing



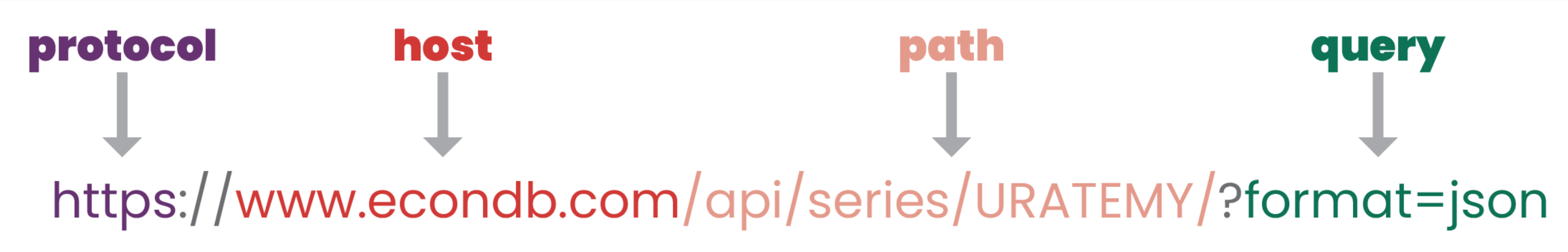
protocol

host

path

query

https://www.econdb.com/api/series/URATEMY/?format=json


protocol

host

path

query

https://www.econdb.com/api/series/URATEMY/?format=json

```r
library(httr)
api_url <- modify_url("https://www.econdb.com/",
                      path = "api/series/URATEMY",
                      query = list(format = "json"))
api_url
## [1] "https://www.econdb.com/api/series/URATEMY?format=json"
```

Why not just build the URL on your own?

It’s easy enough to just write this:

```
https://www.econdb.com/api/series/URATEMY?format=json
```

Special characters!

```
modify_url("https://www.example.com/",
           path = "blah",
           query = list(var1 = 142.5, var2 = "Thing & 10%"))
## [1] "https://www.example.com/blah?var1=142.5&var2=Thing%20%26%2010%25"
```

Getting data from websites

Status code

- 200: OK
- 400: Bad Request
- 403: Forbidden
- 404: Not Found

Full list

Headers

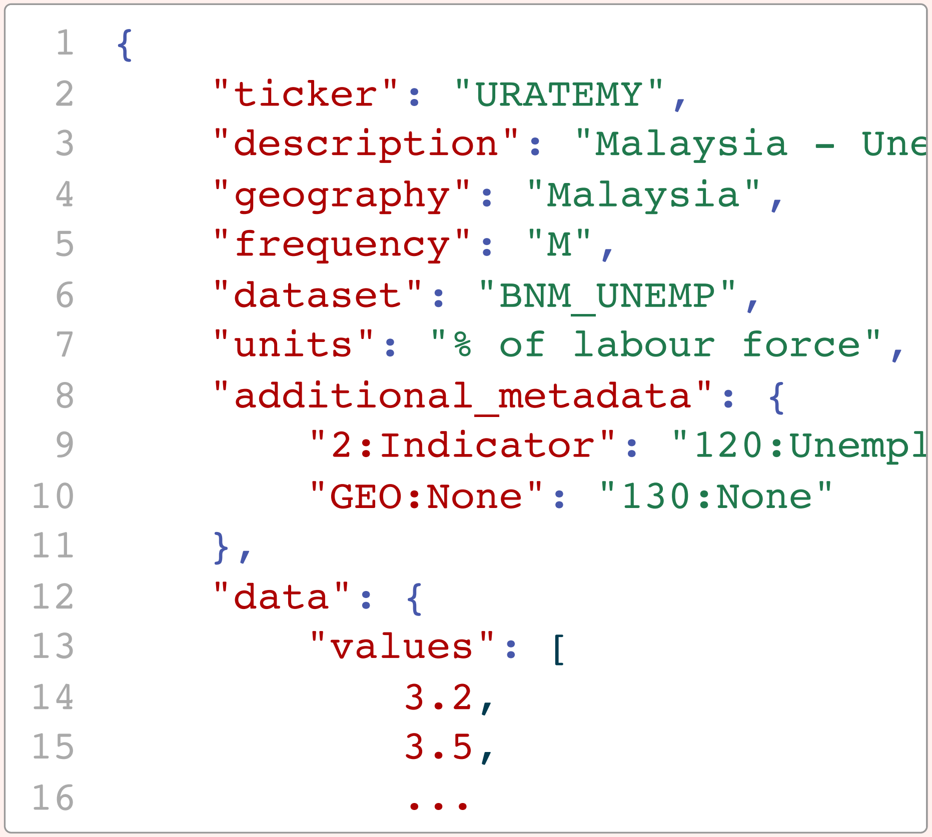
```
Date: Sun, 11 Sep 2022 16:45:20 GM
Content-Type: application/json
Content-Encoding: gzip
...
```

Full list

Content types

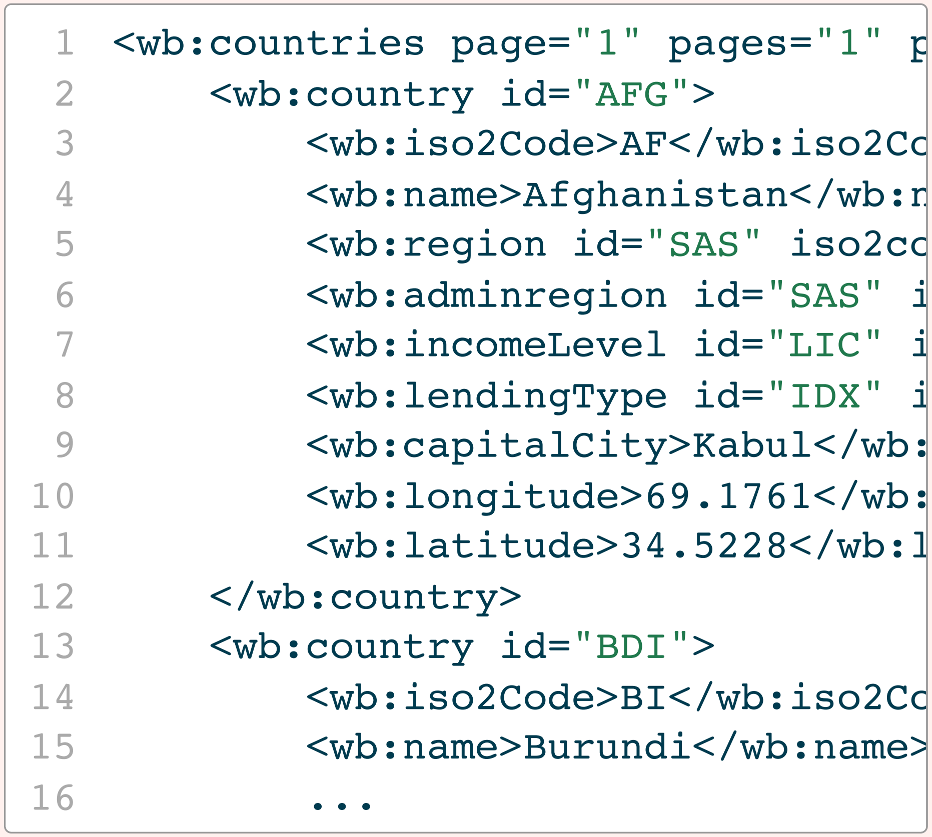
Data can be returned as text, JSON, XML, files, etc.

JSON

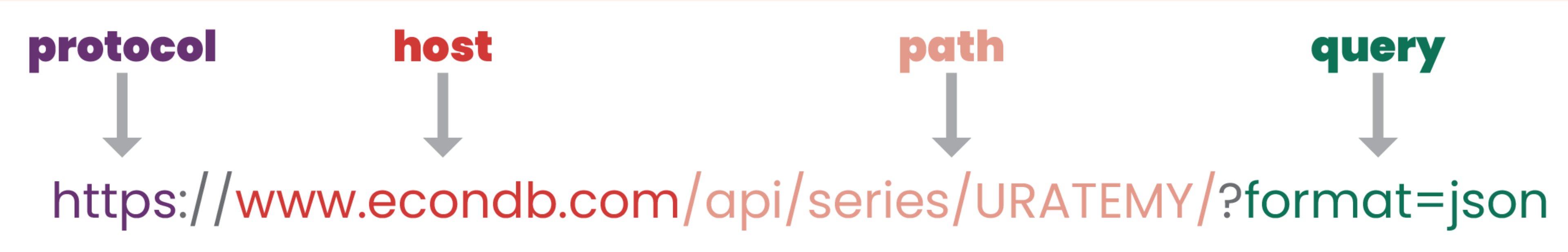


```
{
    "ticker": "URATEMY",
    "description": "Malaysia - Une
    "geography": "Malaysia",
    "frequency": "M",
    "dataset": "BNM_UNEMP",
    "units": "% of labour force",
    "additional_metadata": {
        "2:Indicator": "120:Unempl
        "GEO:None": "130:None"
    },
    "data": {
        "values": [
            3.2,
            3.5,
            ...
```

XML



```
<wb:countries page="1" pages="1" p
    <wb:country id="AFG">
        <wb:iso2Code>AF</wb:iso2Co
        <wb:name>Afghanistan</wb:n
        <wb:region id="SAS" iso2co
        <wb:adminregion id="SAS" i
        <wb:incomeLevel id="LIC" i
        <wb:lendingType id="IDX" i
        <wb:capitalCity>Kabul</wb:
        <wb:longitude>69.1761</wb:
        <wb:latitude>34.5228</wb:l
    </wb:country>
    <wb:country id="BDI">
        <wb:iso2Code>BI</wb:iso2Co
        <wb:name>Burundi</wb:name>
        ...
```



protocol

host

path

query

https://www.econdb.com/api/series/URATEMY/?format=json

```r
# Build the URL query
api_url <- modify_url("https://www.econdb.com/",
                      path = "api/series/URATEMY",
                      query = list(format = "json"))

# Submit the query
r <- GET(api_url)
r
```

```
## Response [https://www.econdb.com/api/series/URATEMY/?format=json]
##   Date: 2022-09-11 15:56
##   Status: 200
##   Content-Type: application/json
##   Size: 3.7 kB
```

```
headers(r)
## $date
## [1] "Sun, 11 Sep 2022 15:56:19 GMT"
##
## $`content-type`
## [1] "application/json"
##
## $vary
## [1] "Accept-Encoding"
##
## $vary
## [1] "Accept, Origin"
##
## $allow
## [1] "GET, HEAD, OPTIONS"
##
## $`x-frame-options`
## [1] "DENY"
##
```

```r
content(r)
## $ticker
## [1] "URATEMY"
##
## $description
## [1] "Malaysia - Unemployment"
##
## $geography
## [1] "Malaysia"
##
## $frequency
## [1] "M"
##
## $dataset
## [1] "BNM_UNEMP"
##
## $units
## [1] "% of labour force"
##
```

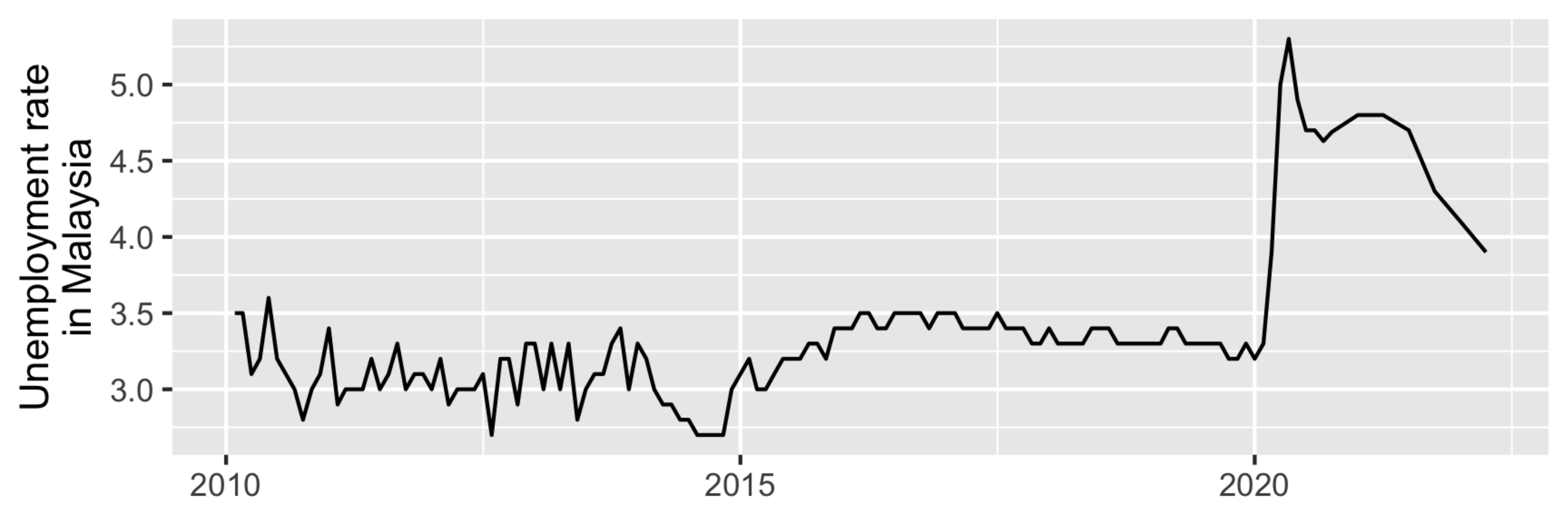
```r
content(r, "text")
## [1] "{\"ticker\":\"URATEMY\",\"description\":\"Malaysia - Unemployment\"
```

```json
{
    "ticker": "URATEMY",
    "description": "Malaysia - Unemployment",
    "geography": "Malaysia",
    "frequency": "M",
    "dataset": "BNM_UNEMP",
    "units": "% of labour force",
    "additional_metadata": {
        "2:Indicator": "120:Unemployment ",
        "GEO:None": "130:None"
    },
    "data": {
        "values": [
            3.2,
            3.5,
            3.5,
            3.5,
            3.1,
```

```r
library(jsonlite)
api_data <- data.frame(fromJSON(content(r, "text"))$data)
```

```
## # A tibble: 137 × 3
##    values dates      status
##     <dbl> <chr>      <chr>
##  1    3.2 2001-10-01 Final
##  2    3.5 2010-01-01 Final
##  3    3.5 2010-02-01 Final
##  4    3.5 2010-03-01 Final
##  5    3.1 2010-04-01 Final
##  6    3.2 2010-05-01 Final
##  7    3.6 2010-06-01 Final
##  8    3.2 2010-07-01 Final
##  9    3.1 2010-08-01 Final
## 10    3   2010-09-01 Final
## # … with 127 more rows
```

```r
library(ggplot2)
library(lubridate)

api_data_clean <- api_data %>%
  mutate(dates = ymd(dates)) %>%
  filter(dates > ymd("2010-01-01"))

ggplot(api_data_clean, aes(x = dates, y = values)) +
  geom_line() +
  labs(x = NULL, y = "Unemployment rate\nin Malaysia")
```



Your turn!

Every API is different

- Each API will accept different arguments, use different URLs, return different variables and formats
- **Read the documentation!**

API authentication

Services will often limit your access

- Rate limiting (*x* API calls/hour)
- Subscription limiting (must have an account)

“Logging in” to an API

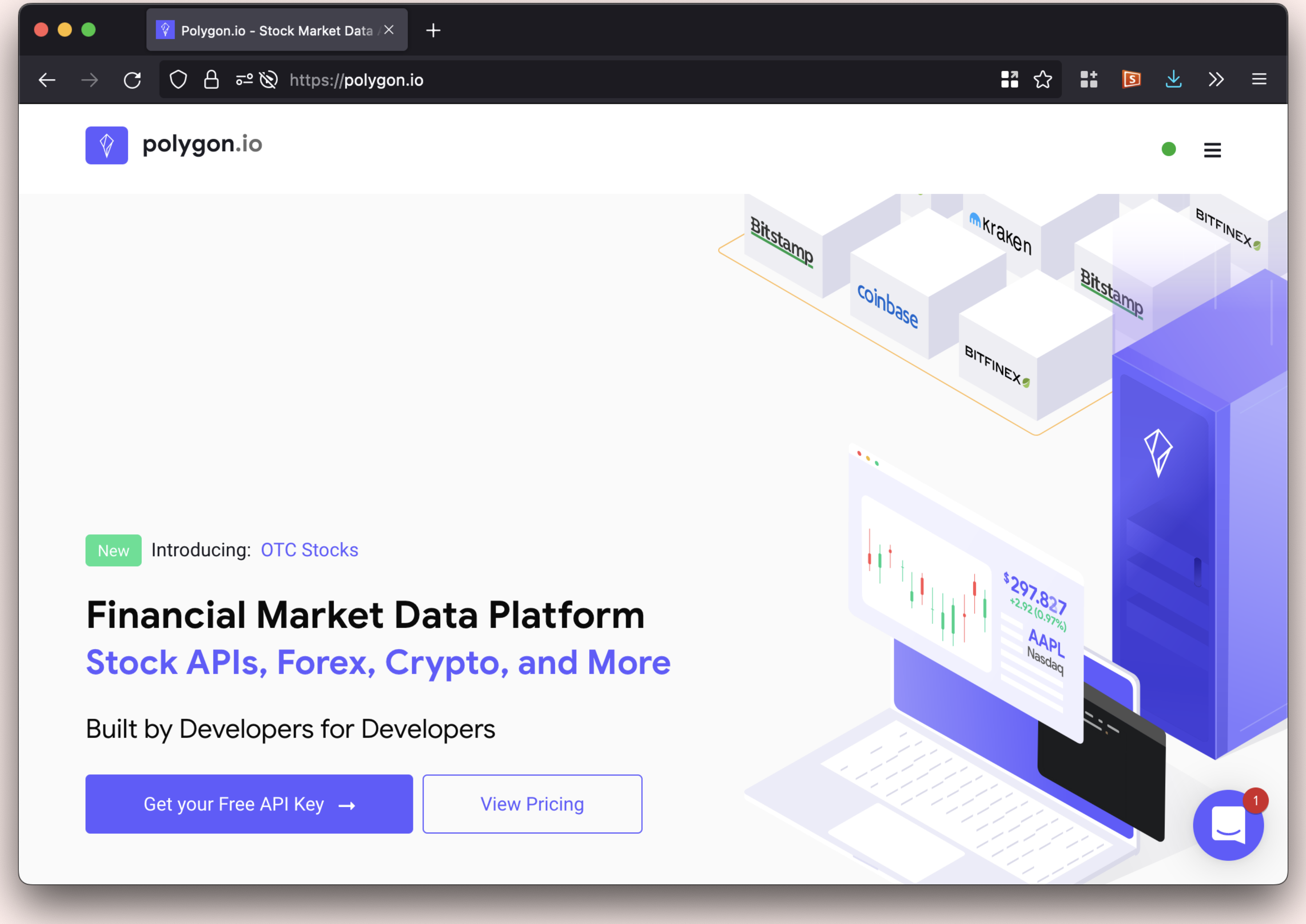
API key

A special parameter that you must include in the query

oAuth authentication

A special file called a “token” that contains the login information for the service

How to create an oAuth token with R



New Introducing: OTC Stocks

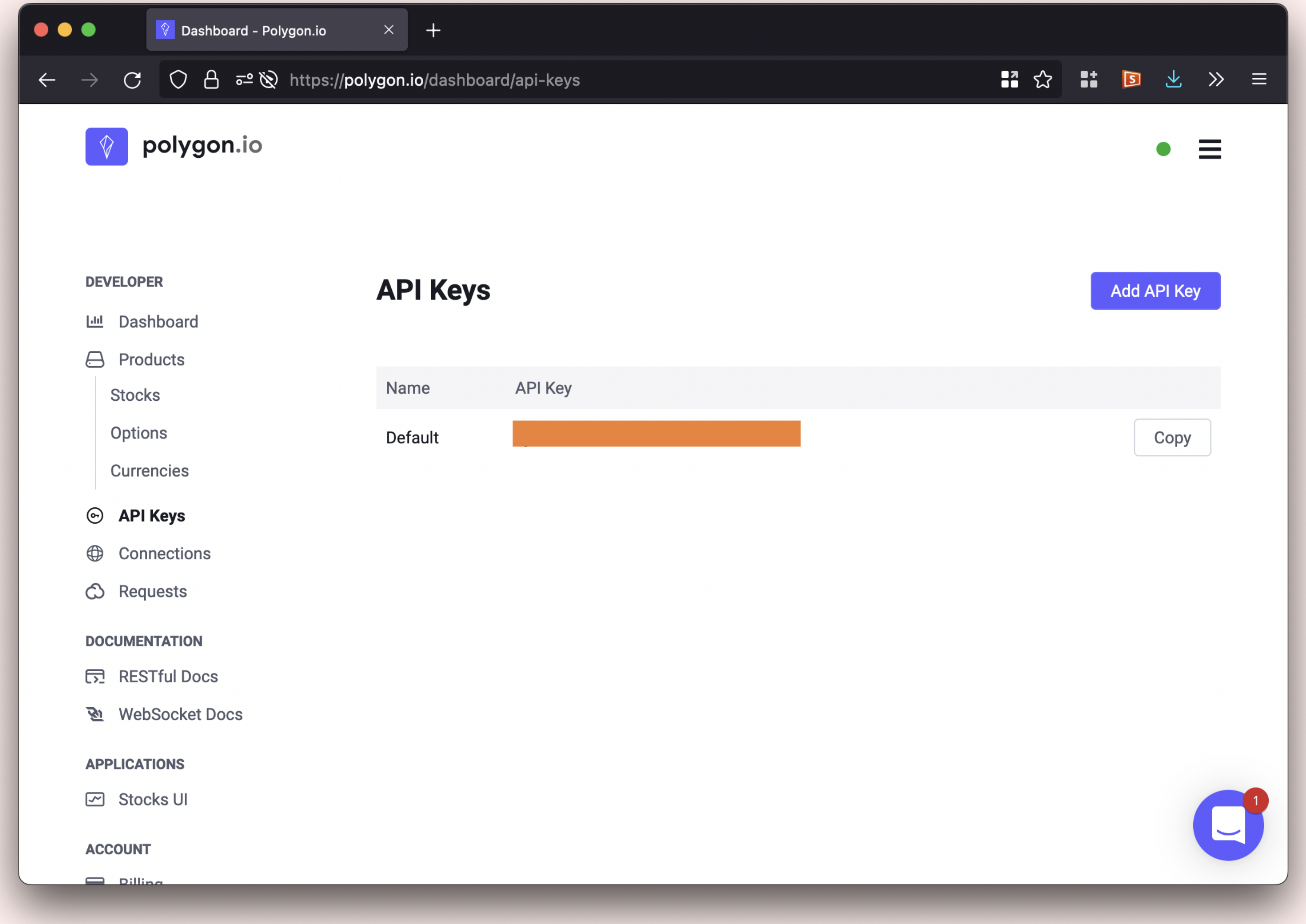
Financial Market Data Platform

Stock APIs, Forex, Crypto, and More

Built by Developers for Developers

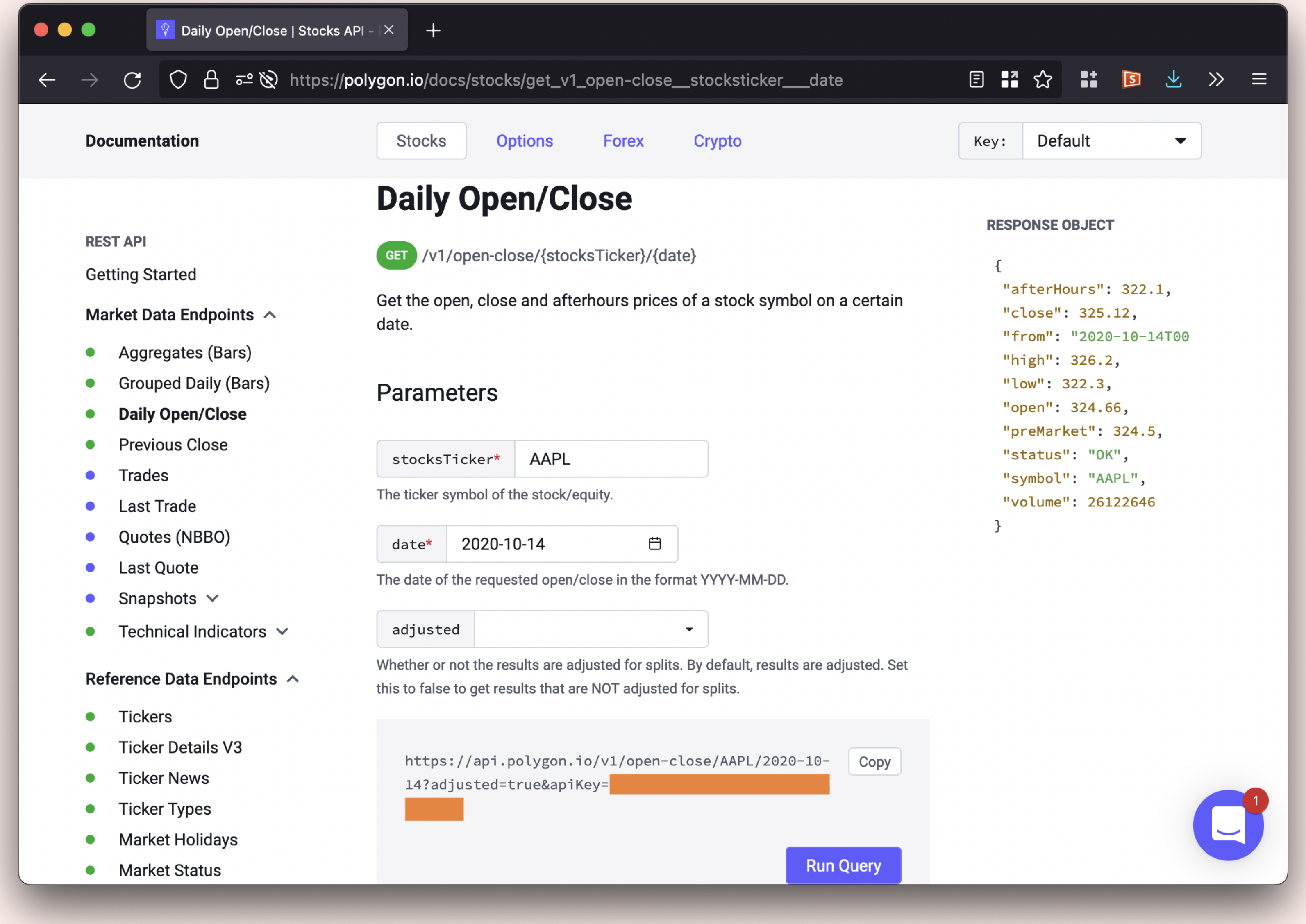
Get your Free API Key →

View Pricing



API Keys

Name	API Key
Default	[redacted]



Daily Open/Close | Stocks API -
https://polygon.io/docs/stocks/get_v1_open-close__stocksticker___date
Documentation
Stocks
Options
Forex
Crypto
Key:
Default

Daily Open/Close

GET /v1/open-close/{stocksTicker}/{date}

Get the open, close and afterhours prices of a stock symbol on a certain date.

Parameters

stocksTicker* AAPL

The ticker symbol of the stock/equity.

date* 2020-10-14

The date of the requested open/close in the format YYYY-MM-DD.

adjusted

Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.

```
https://api.polygon.io/v1/open-close/AAPL/2020-10-14?adjusted=true&apiKey=[redacted]
```

RESPONSE OBJECT

```
{
 "afterHours": 322.1,
 "close": 325.12,
 "from": "2020-10-14T00
 "high": 326.2,
 "low": 322.3,
 "open": 324.66,
 "preMarket": 324.5,
 "status": "OK",
 "symbol": "AAPL",
 "volume": 26122646
}
```

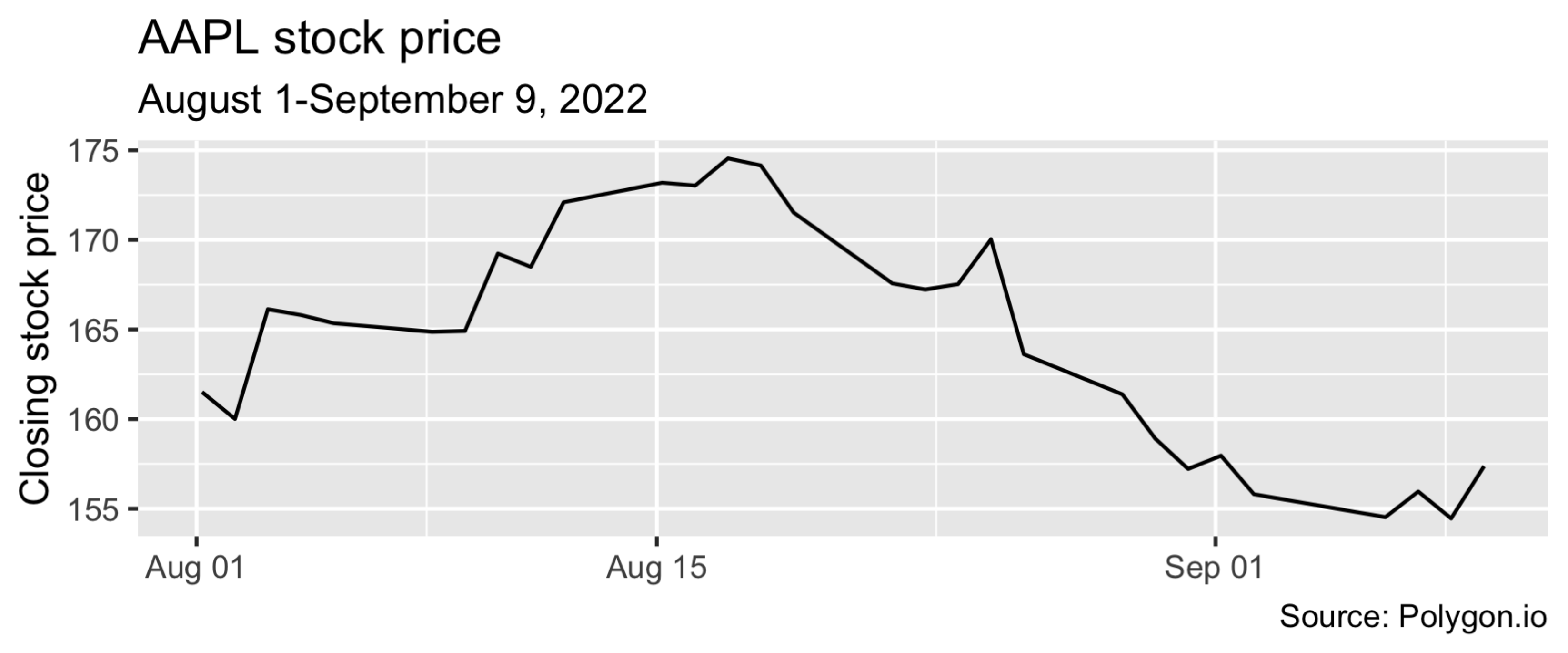
```r
library(httr)
polygon_url <- modify_url(
  "https://api.polygon.io",
  path = "v2/aggs/ticker/AAPL/range/1/day/2022-08-01/2022-09-09",
  query = list(apiKey = "SUPER_SECRET_THING")
)
```

```r
r <- GET(polygon_url)
r
## Response [https://api.polygon.io/v2/aggs/ticker/AAPL/range/1/day/
## 2022-08-01/2022-09-09?apiKey=SUPER_SECRET_THING]
##   Date: 2022-09-11 16:45
##   Status: 200
##   Content-Type: application/json
##   Size: 3.25 kB
```

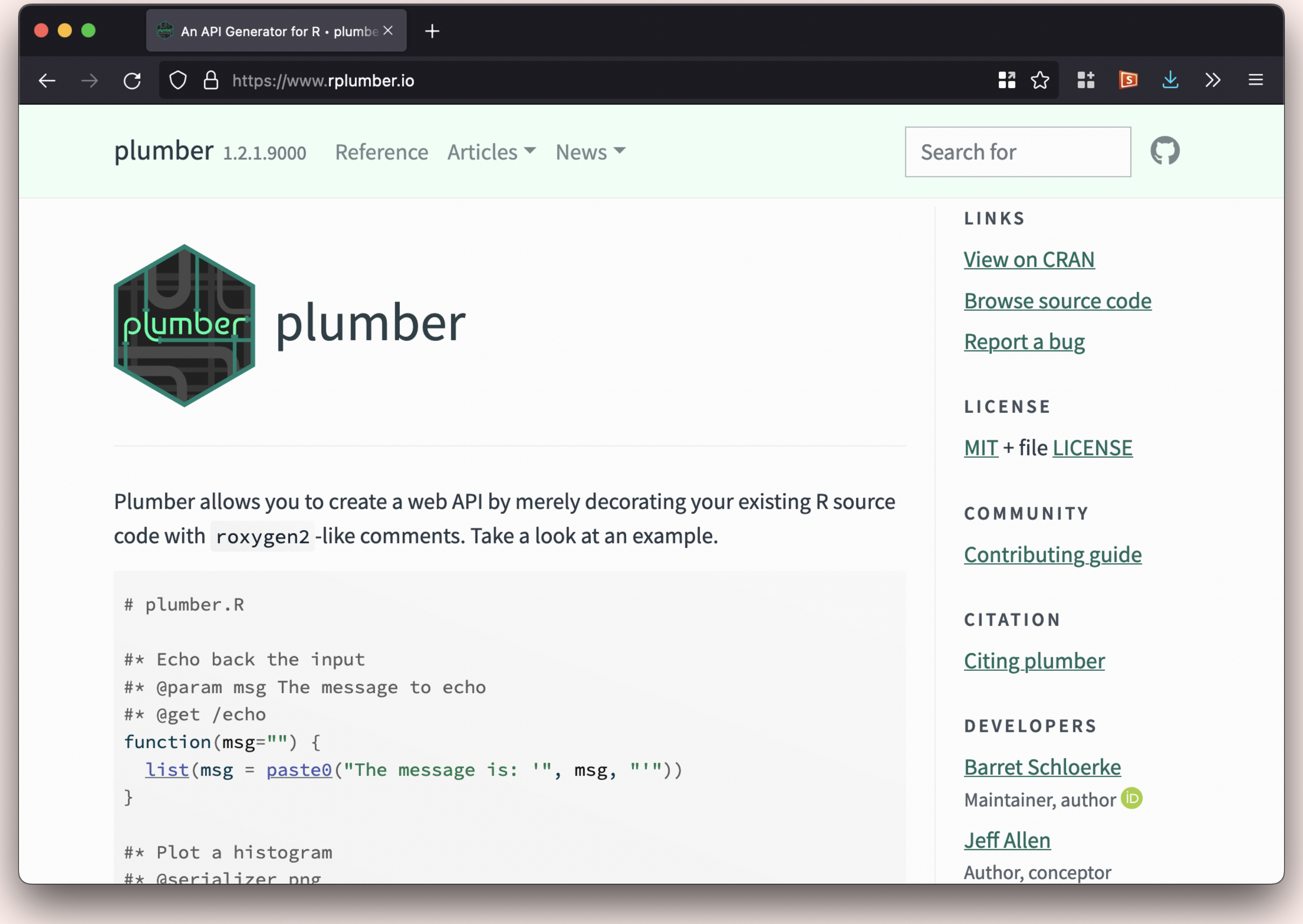
```r
content(r, "text")
## [1] "{\"ticker\":\"AAPL\",\"queryCount\":29,\"resultsCount\":29,\"adjust
```

```json
{
    "ticker": "AAPL",
    "queryCount": 29,
    "resultsCount": 29,
    "adjusted": true,
    "results": [
        {
            "v": 6.7778379e+07,
            "vw": 162.1045,
            "o": 161.01,
            "c": 161.51,
            "h": 163.59,
            "l": 160.89,
            "t": 1659326400000,
            "n": 594290
        },
        {
            "v": 5.9907025e+07,
            "vw": 160.6021
```

```r
library(ggplot2)
library(lubridate)

aapl <- data.frame(fromJSON(content(r, "text"))$results) %>%
  mutate(ts = as_datetime(t / 1000))

ggplot(aapl, aes(x = ts, y = c)) +
  geom_line() +
  labs(x = NULL, y = "Closing stock price", title = "AAPL stock price",
       subtitle = "August 1-September 9, 2022", caption = "Source: Polygon.
```



Make your own API



plumber

Plumber allows you to create a web API by merely decorating your existing R source code with `roxygen2`-like comments. Take a look at an example.

```
# plumber.R

#* Echo back the input
#* @param msg The message to echo
#* @get /echo
function(msg="") {
  list(msg = paste0("The message is: '", msg, "'"))
}

#* Plot a histogram
#* @serializer png
```

LINKS

View on CRAN

Browse source code

Report a bug

LICENSE

MIT + file LICENSE

COMMUNITY

Contributing guide

CITATION

Citing plumber

DEVELOPERS

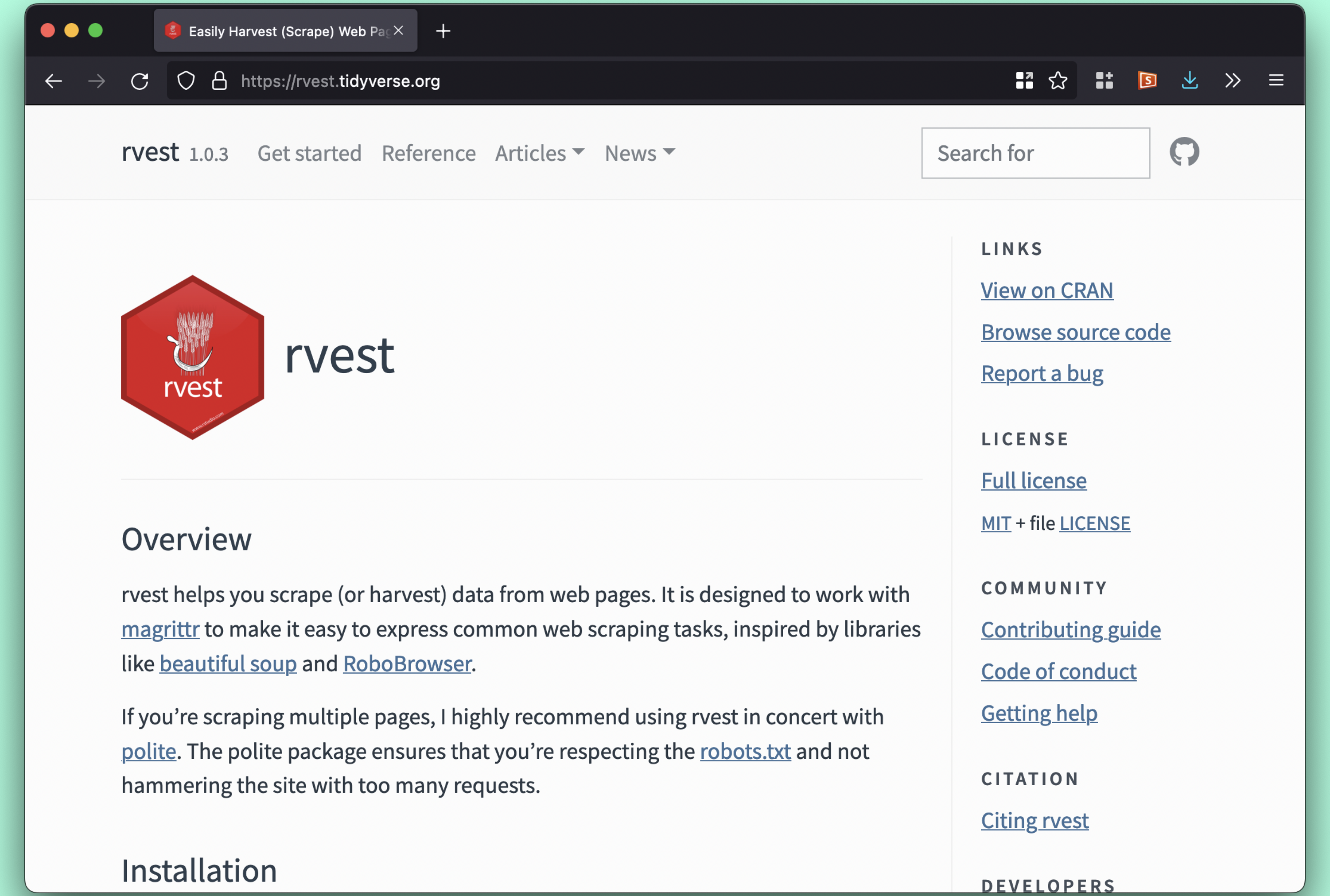
Barret Schloerke
Maintainer, author

Jeff Allen
Author, conceptor

Scraping websites

What if there's no API? :(

- Copy and paste
- Scrape the website
  (fancy copying and pasting)



rvest

Overview

rvest helps you scrape (or harvest) data from web pages. It is designed to work with magrittr to make it easy to express common web scraping tasks, inspired by libraries like beautiful soup and RoboBrowser.

If you're scraping multiple pages, I highly recommend using rvest in concert with polite. The polite package ensures that you're respecting the robots.txt and not hammering the site with too many requests.

Installation

LINKS

View on CRAN

Browse source code

Report a bug

LICENSE

Full license

MIT + file LICENSE

COMMUNITY

Contributing guide

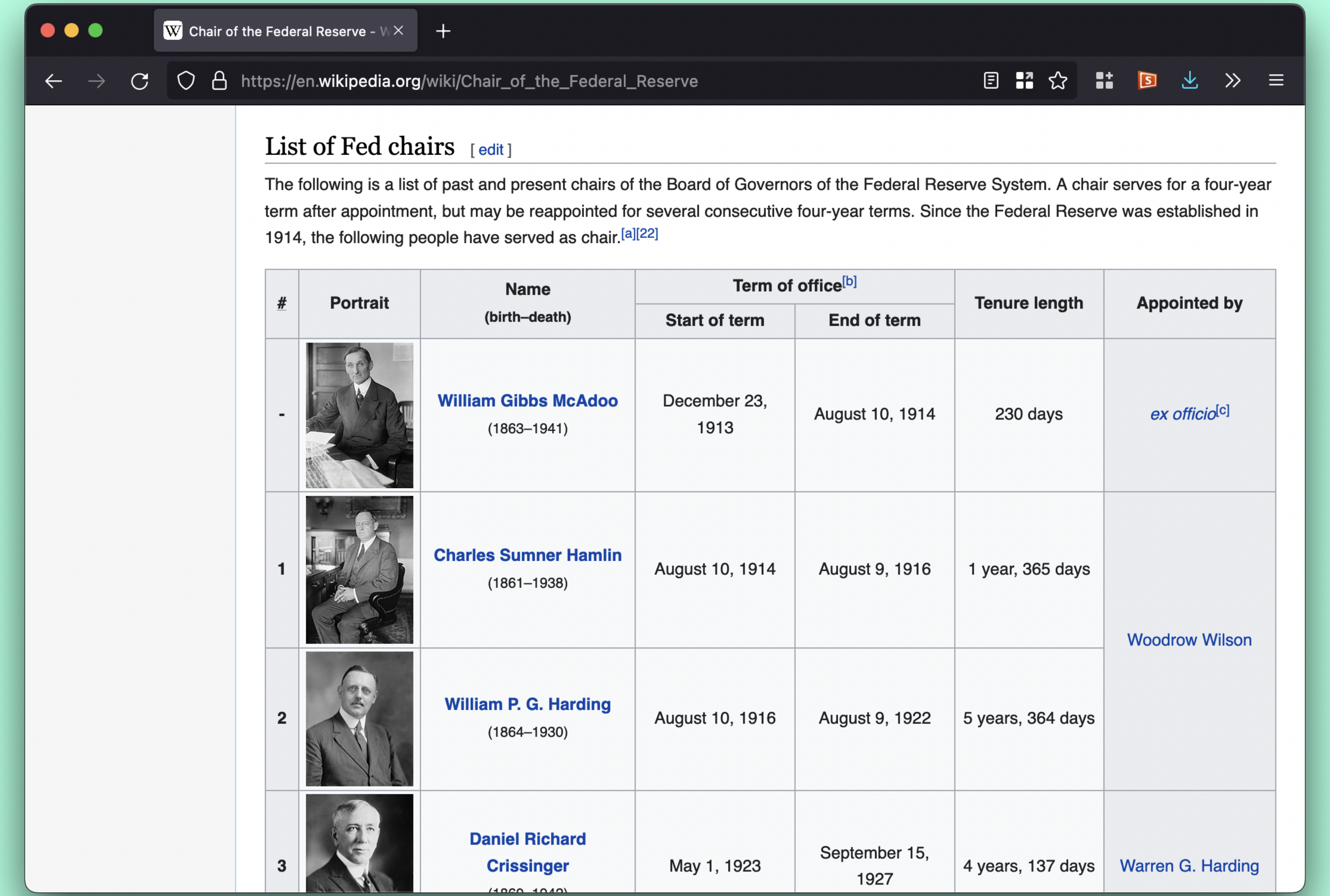
Code of conduct

Getting help

CITATION

Citing rvest

DEVELOPERS


Chair of the Federal Reserve - W
https://en.wikipedia.org/wiki/Chair_of_the_Federal_Reserve

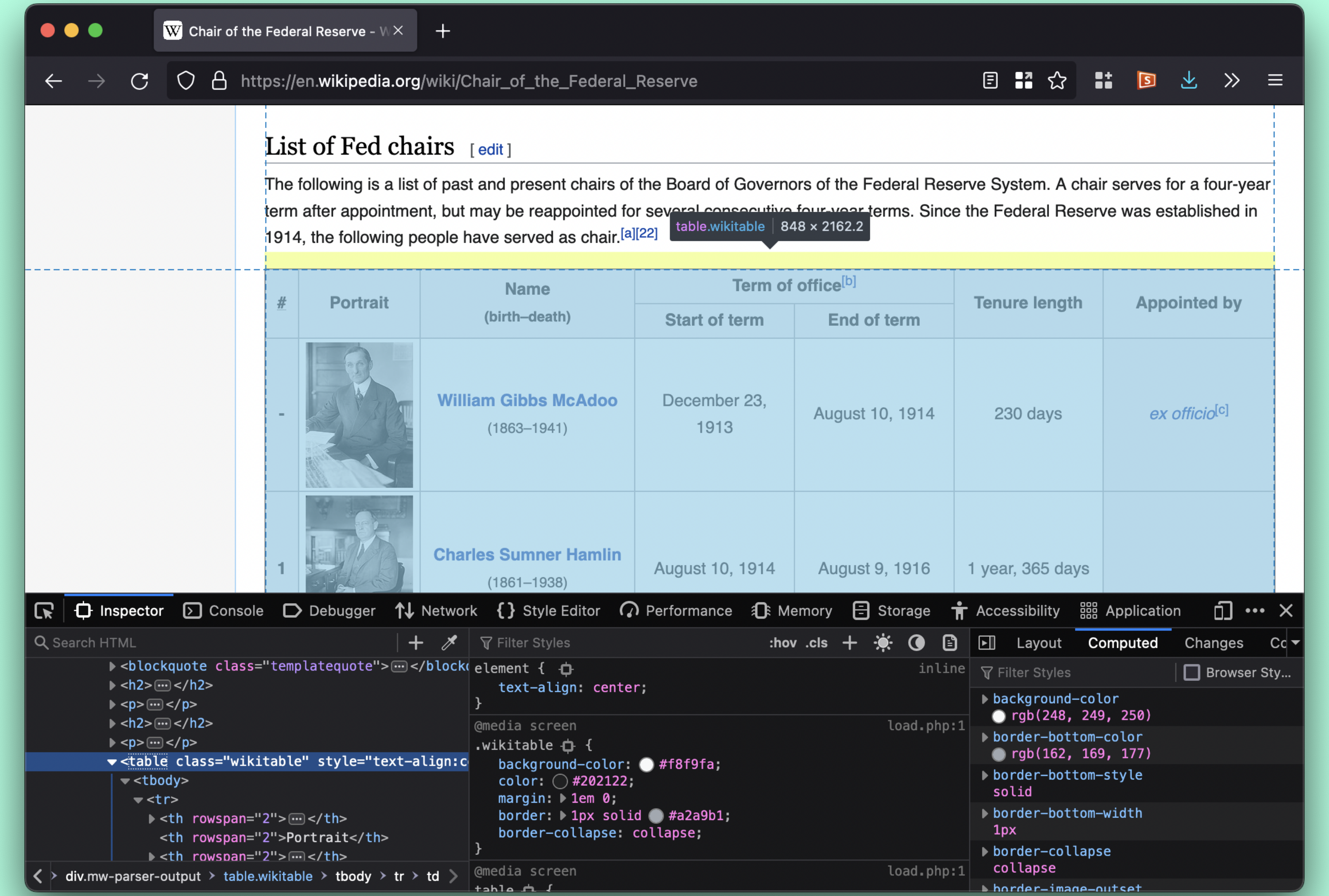
List of Fed chairs [edit]

The following is a list of past and present chairs of the Board of Governors of the Federal Reserve System. A chair serves for a four-year term after appointment, but may be reappointed for several consecutive four-year terms. Since the Federal Reserve was established in 1914, the following people have served as chair.[a][22]

#	Portrait	Name (birth–death)	Term of office[b]		Tenure length	Appointed by
			Start of term	End of term		
-		**William Gibbs McAdoo** (1863–1941)	December 23, 1913	August 10, 1914	230 days	*ex officio*[c]
1		**Charles Sumner Hamlin** (1861–1938)	August 10, 1914	August 9, 1916	1 year, 365 days	Woodrow Wilson
2		**William P. G. Harding** (1864–1930)	August 10, 1916	August 9, 1922	5 years, 364 days	
3		**Daniel Richard Crissinger** (1860–1942)	May 1, 1923	September 15, 1927	4 years, 137 days	Warren G. Harding

```r
library(rvest)

wiki_url <- "https://en.wikipedia.org/wiki/Chair_of_the_Federal_Reserve"

# Even better to use the Internet Archive since web pages change over time
wiki_url <- "https://web.archive.org/web/20220908211042/https://en.wikipedi

wiki_raw <- read_html(wiki_url)
wiki_raw
```

```
## {html_document}
## <html class="client-nojs" lang="en" dir="ltr">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=
...
## [2] <body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-sub
...
```



List of Fed chairs [edit]

The following is a list of past and present chairs of the Board of Governors of the Federal Reserve System. A chair serves for a four-year term after appointment, but may be reappointed for several consecutive four-year terms. Since the Federal Reserve was established in 1914, the following people have served as chair.[a][22]

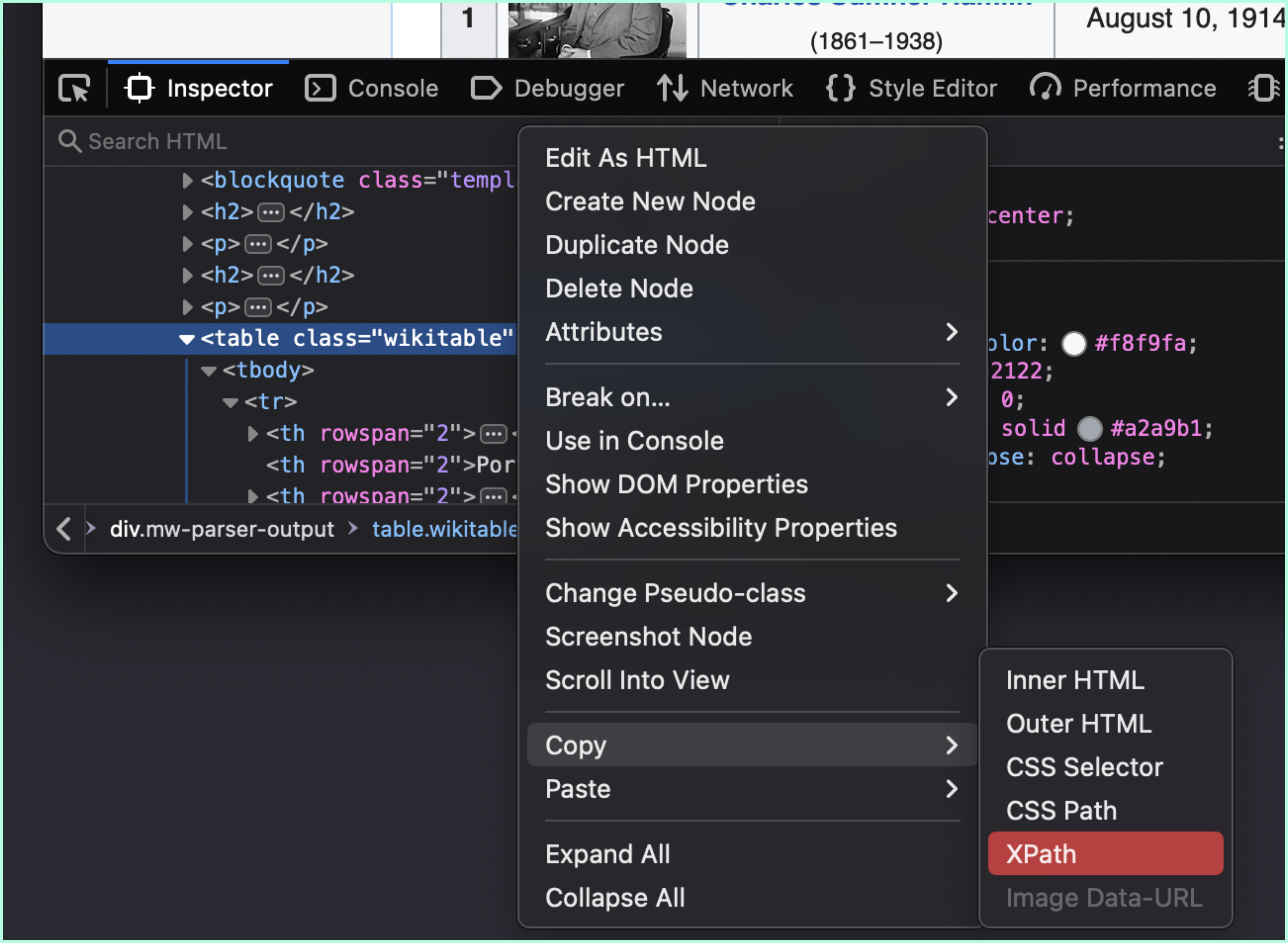
#	Portrait	Name (birth–death)	Term of office[b]		Tenure length	Appointed by
			Start of term	End of term		
-		William Gibbs McAdoo (1863–1941)	December 23, 1913	August 10, 1914	230 days	*ex officio*[c]
1		Charles Sumner Hamlin (1861–1938)	August 10, 1914	August 9, 1916	1 year, 365 days	

```
<blockquote class="templatequote">⋯</blockq
<h2>⋯</h2>
<p>⋯</p>
<h2>⋯</h2>
<p>⋯</p>
<table class="wikitable" style="text-align:c
  <tbody>
    <tr>
      <th rowspan="2">⋯</th>
      <th rowspan="2">Portrait</th>
      <th rowspan="2">⋯</th>
```

div.mw-parser-output › table.wikitable › tbody › tr › td

```
element {                                  inline
   text-align: center;
}
@media screen                          load.php:1
.wikitable {
   background-color: #f8f9fa;
   color: #202122;
   margin: 1em 0;
   border: 1px solid #a2a9b1;
   border-collapse: collapse;
}
@media screen                          load.php:1
table {
```

```
background-color
  rgb(248, 249, 250)
border-bottom-color
  rgb(162, 169, 177)
border-bottom-style
  solid
border-bottom-width
  1px
border-collapse
  collapse
border-image-outset
```



```
<blockquote class="templ
<h2>⋯</h2>
<p>⋯</p>
<h2>⋯</h2>
<p>⋯</p>
<table class="wikitable"
  <tbody>
    <tr>
      <th rowspan="2">⋯
      <th rowspan="2">Por
      <th rowspan="2">⋯
```

```
center;
olor: #f8f9fa;
2122;
 0;
 solid #a2a9b1;
ose: collapse;
```

div.mw-parser-output > table.wikitable

```r
wiki_raw %>%
  html_nodes(xpath = "/html/body/div[3]/div[3]/div[5]/div[1]/table[2]")
```

```
{xml_nodeset (1)}
[1] <table class="wikitable" style="text-align:center"><tbody>\n<tr>\n<t ...
```

```r
wiki_raw %>%
  html_nodes(xpath = "/html/body/div[3]/div[3]/div[5]/div[1]/table[2]") %>%
  html_table()
```

```
[[1]]
# A tibble: 18 × 7
   `#`   Portrait   `Name(birth–death)`       Term …¹ Term …² Tenur…³ Appoi…⁴
   <chr> <chr>      <chr>                     <chr>   <chr>   <chr>   <chr>
 1 #     "Portrait" Name(birth–death)         Start … End of… Tenure… Appoin…
 2 -     ""         William Gibbs McAdoo(1… Decemb… August… 230 da… ex off…
 3 1     ""         Charles Sumner Hamlin(… August… August… 1 year… Woodro…
 4 2     ""         William P. G. Harding(… August… August… 5 year… Woodro…
 5 3     ""         Daniel Richard Crissin… May 1,… Septem… 4 year… Warren…
 6 4     ""         Roy A. Young(1882–1960) Octobe… August… 2 year… Calvin…
 7 5     ""         Eugene Meyer(1875–1959) Septem… May 10… 2 year… Herber…
 8 6     ""         Eugene Robert Black(18… May 19… August… 1 year… Frankl…
 9 7     ""         Marriner S. Eccles[d](… Novemb… Januar… 13 yea… Frankl…
10 8     ""         Thomas B. McCabe(1893–… April … March … 2 year… Harry …
11 9     ""         William McChesney Mart… April … Januar… 18 yea… Harry …
12 10    ""         Arthur F. Burns[e](190… Februa… Januar… 7 year… Richar…
```

```r
wiki_raw %>%
  html_nodes(xpath = "/html/body/div[3]/div[3]/div[5]/div[1]/table[2]") %>%
  html_table() %>%
  bind_rows()
```

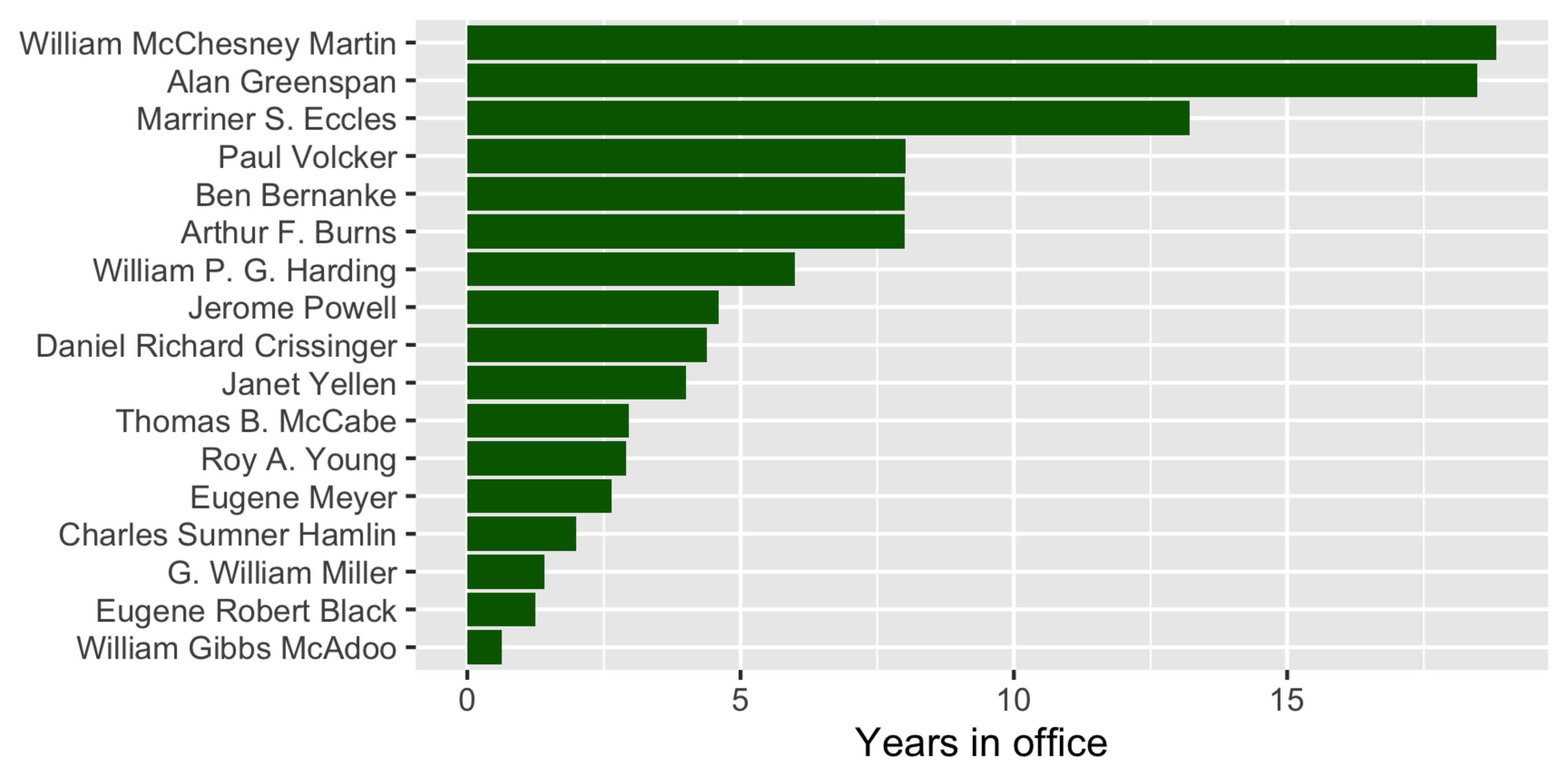
```
# A tibble: 18 × 7
   `#`   Portrait   `Name(birth–death)`     Term …¹ Term …² Tenur…³ Appoi…⁴
   <chr> <chr>      <chr>                   <chr>   <chr>   <chr>   <chr>
 1 #     "Portrait" Name(birth–death)       Start … End of… Tenure… Appoin…
 2 -     ""         William Gibbs McAdoo(1… Decemb… August… 230 da… ex off…
 3 1     ""         Charles Sumner Hamlin(… August… August… 1 year… Woodro…
 4 2     ""         William P. G. Harding(… August… August… 5 year… Woodro…
 5 3     ""         Daniel Richard Crissin… May 1,… Septem… 4 year… Warren…
 6 4     ""         Roy A. Young(1882–1960) Octobe… August… 2 year… Calvin…
 7 5     ""         Eugene Meyer(1875–1959) Septem… May 10… 2 year… Herber…
 8 6     ""         Eugene Robert Black(18… May 19… August… 1 year… Frankl…
 9 7     ""         Marriner S. Eccles[d](… Novemb… Januar… 13 yea… Frankl…
10 8     ""         Thomas B. McCabe(1893–… April … March … 2 year… Harry …
11 9     ""         William McChesney Mart… April … Januar… 18 yea… Harry …
12 10    ""         Arthur F. Burns[e](190… Februa… Januar… 7 year… Richar…
13 11    ""         G. William Miller(1925… March … August… 1 year… Jimmy…
```

```r
wiki_clean <- wiki_raw %>%
  html_nodes(xpath = "/html/body/div[3]/div[3]/div[5]/div[1]/table[2]") %>%
  html_table() %>%
  bind_rows() %>%
  # Remove first row
  slice(-1) %>%
  # Extract name
  separate(`Name(birth–death)`, into = c("Name", "birth-death"), sep = "\\(
  mutate(Name = str_remove(Name, "\\[.\\]")) %>%
  # Calculate duration in office
  mutate(tenure_length = as.period(`Tenure length`)) %>%
  mutate(seconds = as.numeric(tenure_length)) %>%
  mutate(years = seconds / 60 / 60 / 24 / 365.25) %>%
  # Put name in order of duration
  arrange(tenure_length) %>%
  mutate(Name = fct_inorder(Name))
```

```r
wiki_clean %>%
  select(Name, `Tenure length`, years)
```

```
# A tibble: 17 × 3
   Name                        `Tenure length`     years
   <fct>                       <chr>               <dbl>
 1 William Gibbs McAdoo        230 days            0.630
 2 Eugene Robert Black         1 year, 88 days     1.24
 3 G. William Miller           1 year, 151 days    1.41
 4 Charles Sumner Hamlin       1 year, 365 days    2.00
 5 Eugene Meyer                2 years, 236 days   2.65
 6 Roy A. Young                2 years, 331 days   2.91
 7 Thomas B. McCabe            2 years, 350 days   2.96
 8 Janet Yellen                4 years, 0 days     4
 9 Daniel Richard Crissinger 4 years, 137 days   4.38
10 Jerome Powell               4 years, 218 days   4.60
11 William P. G. Harding       5 years, 364 days   6.00
12 Arthur F. Burns             7 years, 364 days   8.00
```

```r
ggplot(wiki_clean, aes(x = years, y = Name)) +
  geom_col(fill = "darkgreen") +
  labs(x = "Years in office", y = NULL)
```



More complex scraping

- What if there are multiple tables, or entries, or sections, or web pages?
    - Star Wars example
- Loops!
    - Do it politely

Your turn!

Summary

Real world data is a mess

- Every dataset is unique
- Every API is unique
- Every website is unique

General principles

- Try to use APIs to access data directly from data sources
  - Ideally use a pre-built R package
  - If not, use {httr}
  - Consider making an API package (best practices)
- If there's no API, scrape (politely) with {rvest}