BANCO DE PORTUGAL
EUROSISTEMA

ACADEMIA
BANCO DE PORTUGAL

“Machine Learning” and “Big Data”: basic concepts and applications

Nelson Areal • Universidade do Minho
18 December 2018

Training session 1

Introductions

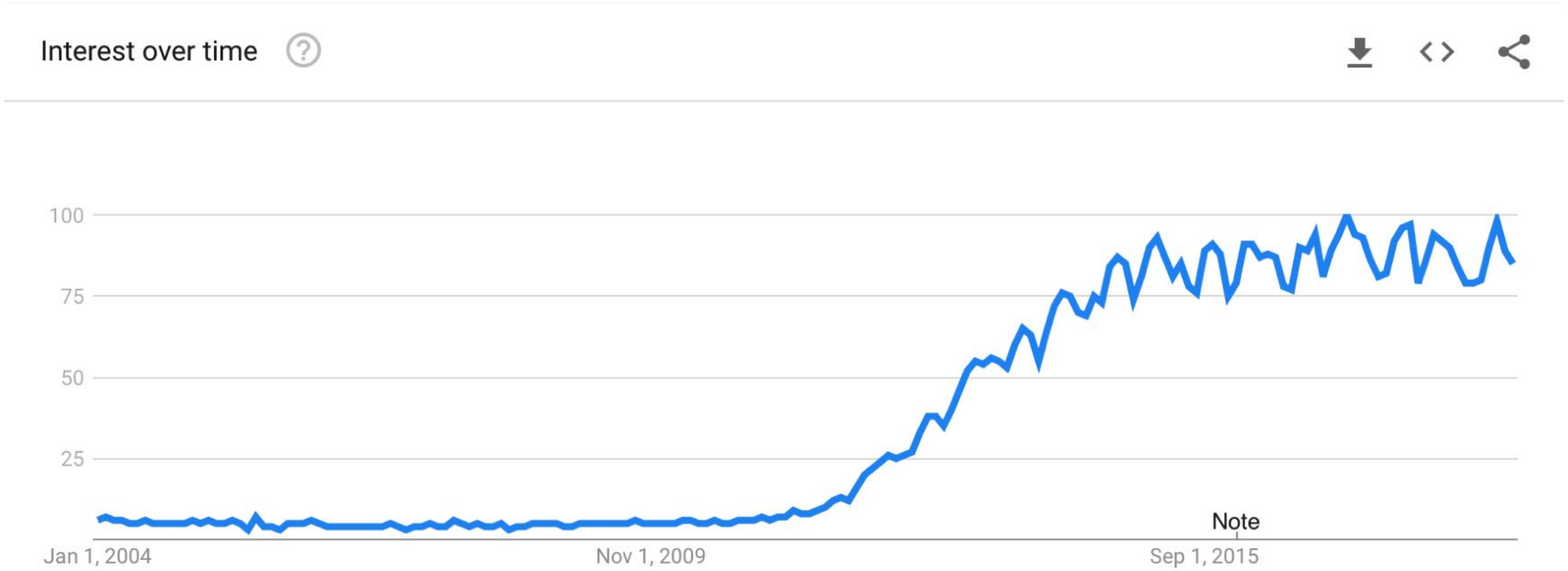
Nelson Areal

Management Department
School of Economics and Management
University of Minho

Twitter: @nareal
URL: http://nelsonareal.net

Big data

Big data



https://trends.google.com/trends/explore?date=all&q=Big%20data

Big data

?

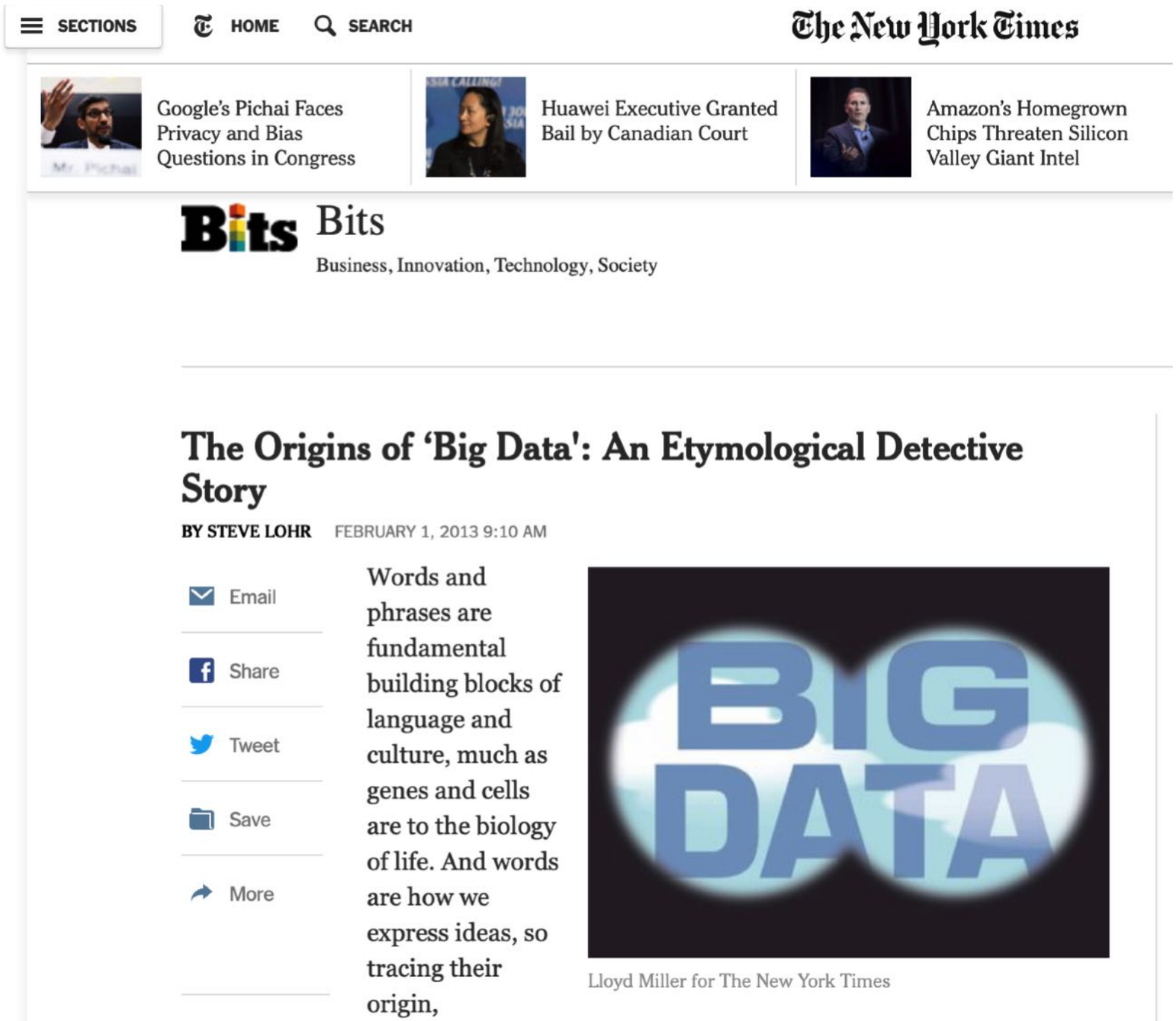
a trip down to memory lane…

Big data

Francis X. Diebold

Diebold, Francis X. (2003) “**Big data dynamic factor models for macroeconomic measurement and forecasting**" In Advances in Economics and Econometrics: Theory and Applications, Eighth World Congress of the Econometric Society,”(edited by M. Dewatripont, LP Hansen and S. Turnovsky), pp. 115-122.

Big data



https://bits.blogs.nytimes.com/2013/02/01/the-origins-of-big-data-an-etymological-detective-story/

Big data

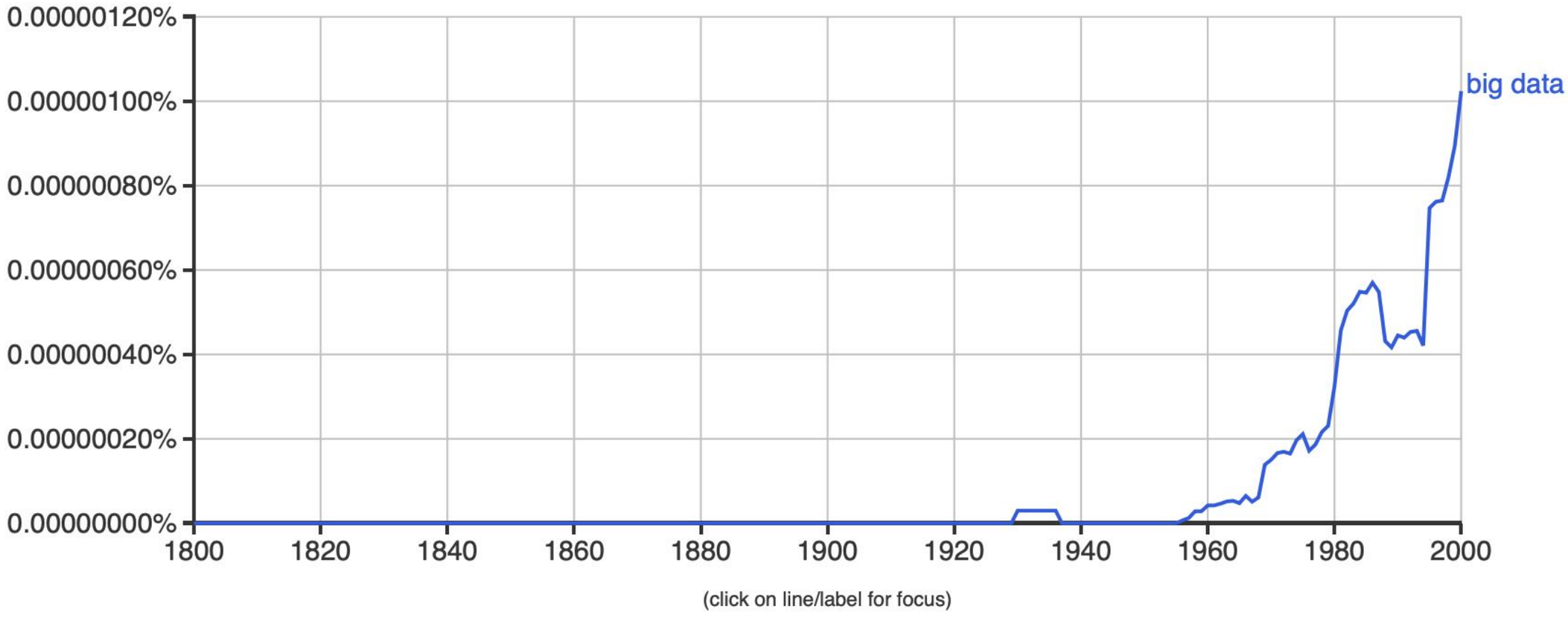
On the Origin(s) and Development of
the Term "Big Data"*

Francis X. Diebold
University of Pennsylvania
fdiebold@sas.upenn.edu

First Draft, August 2012
This Draft, September 21, 2012

Abstract: I investigate the origins of the now-ubiquitous term "Big Data," in industry and academics, in computer science and statistics/econometrics. Credit for coining the term must be shared. In particular, John Mashey and others at Silicon Graphics produced highly-relevant (unpublished, non-academic) work in the mid-1990s. The first significant academic references (independent of each other and of Silicon Graphics) appear to be Weiss and Indurkhya (1998) in computer science and Diebold (2000) in statistics/econometrics. Douglas Laney of Gartner also produced insightful work (again unpublished and non-academic) slightly later. Big Data the term is now firmly entrenched, Big Data the phenomenon continues unabated, and Big Data the discipline is emerging.

Big data



https://books.google.com/ngrams/

Big data

RESOURCE AND ENVIRONMENTAL DATA ANALYSIS

By DANIEL F. MERRIAM
Chairman, Department of Geology, Syracuse University

INTRODUCTION

All the predictions on the depletion of our national resources are pessimistic. We are exhausting our raw materials, and now it is only a question of when. It is of top importance, therefore, that optimum use be made of these resources as they are exploited, and it will be necessary to achieve a balance between use, conservation, and environmental considerations. As difficult as this may be, however, the real problem will be in finding new reserves to replace the depleted ones. Geologists will play a key role in the exploration for these new mineral resources.

of commodities."

The new philosophy will reflect team inquiry. The geologist now will integrate his thinking with computer scientists, statisticians, engineers, hydrologists, geochemists, and geophysicists in outlining areas of interest and in monitoring their development. Methods will reflect the latest available techniques and perhaps simulated real-world models based on millions or even billions of items of data. New ideas wil be put forth by the younger better disciplined geologists who will have been trained to interpret and judge the interdisciplinary multi-based data that have been treated in a complex

https://books.google.pt/books/content?id=64grjG0MLRUC&pg=PA41&img=1&zoom=3&hl=en&sig=ACfU3U0lfQ8jiFg0uvAuByxcbiyUNWuRpQ&ci=55%2C114%2C409%2C152&edge=0

Big data

*"**In the future big data storage and retrieval will be put into use and small specialized systems will proliferate**. Standards and quality control will gradually be installed at all levels and geologists will eventually record all their data on standard forms utilized worldwide The US Geological Survey as already pointed out can continue to be an effective leader in this area of electronic data processing where so much expertise is available from years of experience" (Merriam, 1974)*

Big data

My definition

Regular size

Biggish data

Big data

Anything that fits in memory

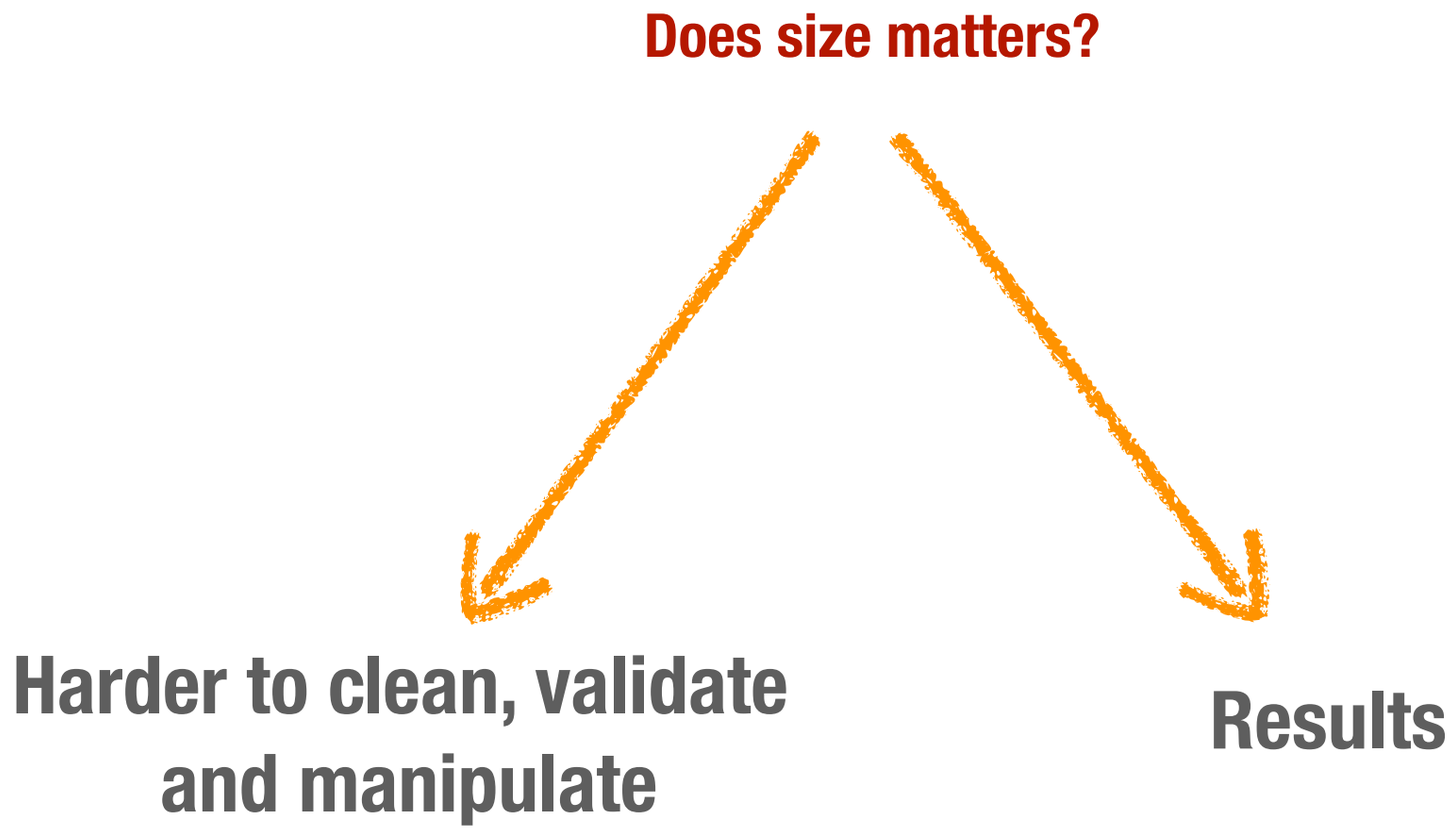
requires distributed storage

the deeper your pocket is the smaller the dataset will look

Big data

Does size matters?

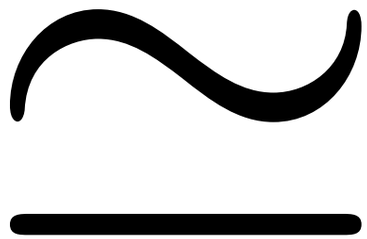
Big data


Does size matters?
Harder to clean, validate and manipulate
Results

Preliminares

Tools

Tools



“I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.” Maslow (1966)

Tools

Top Gear - Jeremy Clarkson's sophisticated way of fixing his BMW Estate

https://www.youtube.com/watch?v=6xsVX6029rU

Tools

Reliability

Computational Statistics & Data Analysis
Volume 40, Issue 4, 28 October 2002, Pages 713-721

On the accuracy of statistical procedures in Microsoft Excel 2000 and Excel XP

B D. McCullough [a], Berry Wilson [b]

Show more

https://doi.org/10.1016/S0167-9473(02)00095-6

Abstract

The problems that rendered Excel 97 unfit for use as a statistical package have not been fixed in either Excel 2000 or Excel 2002 (also called “Excel XP”). Microsoft attempted to fix errors in the standard normal random number generator and the inverse normal function, and in the former case actually made the problem worse.

McCullough, B. D., & Wilson, B. (2002). On the accuracy of statistical procedures in Microsoft Excel 2000 and Excel XP. Computational Statistics and Data Analysis, 40, 713-721.

Tools

Reliability

“In the small world where computer science overlaps with statistics, it was well known that Microsoft Excel was riddled with statistical errors. It was so well known that no one bothered to write about it.”

McCullough, B. (2006). The unreliability of Excel’s statistical procedures. *Foresight: International Journal of Applied Forecasting*, 3, 44-45.

Journal of Statistical Software

January 2013, Volume 52, Issue 7. *http://www.jstatsoft.org/*

Spreadsheets in the Cloud – Not Ready Yet

Bruce D. McCullough
Drexel University

A. Talha Yalta
TOBB University of Economics and Technology

Abstract

Cloud computing is a relatively new technology that facilitates collaborative creation and modification of documents over the internet in real time. Here we provide an introductory assessment of the available statistical functions in three leading cloud spreadsheets namely **Google Spreadsheet**, Microsoft **Excel Web App**, and **Zoho Sheet**. Our results show that the developers of cloud-based spreadsheets are not performing basic quality control, resulting in statistical computations that are misleading and erroneous. Moreover, the developers do not provide sufficient information regarding the software and the hardware, which can change at any time without notice. Indeed, rerunning the tests after several months we obtained different and sometimes worsened results.

Keywords: cloud computing, spreadsheet, accuracy, **Google Docs**, **Excel**, **Zoho**, Wilkinson tests.

McCullough, B. D & Yalta, A. T. (2010). Spreadsheets in the Cloud – Not Ready Yet. Journal of Statistical Software, 52(7), 1-14.

Tools

Reliability

Journal of Statistical Software

April 2010, Volume 34, Issue 4. *http://www.jstatsoft.org/*

On the Numerical Accuracy of Spreadsheets

Marcelo G. Almiron
Universidade Federal
de Alagoas

Bruno Lopes
Universidade Federal
de Alagoas

Alyson L. C. Oliveira
Universidade Federal
de Alagoas

Antonio C. Medeiros
Universidade Federal
de Alagoas

Alejandro C. Frery
Universidade Federal
de Alagoas

Abstract

This paper discusses the numerical precision of five spreadsheets (**Calc**, **Excel**, **Gnumeric**, **NeoOffice** and **Oleo**) running on two hardware platforms (i386 and amd64) and on three operating systems (Windows Vista, Ubuntu Intrepid and Mac OS Leopard). The methodology consists of checking the number of correct significant digits returned by each spreadsheet when computing the sample mean, standard deviation, first-order autocorrelation, F statistic in ANOVA tests, linear and nonlinear regression and distribution functions. A discussion about the algorithms for pseudorandom number generation provided by these platforms is also conducted. We conclude that there is no safe choice among the spreadsheets here assessed: they all fail in nonlinear regression and they are not suited for Monte Carlo experiments.

"This paper discusses the numerical precision of five spreadsheets (Calc, Excel, Gnumeric, NeoOffice and Oleo) running on two hardware platforms (i386 and amd64) and on three operating systems (Windows Vista, Ubuntu Intrepid and Mac OS Leopard).

(...)

We conclude that there is no safe choice among the spreadsheets here assessed: they all fail in nonlinear regression and they are not suited for Monte Carlo experiments."

Almiron, M. G., Lopes, B., Oliveira, A. L. C., Medeiros, A. C., & Frery, A. C. (2010). On the numerical accuracy of spreadsheets. Journal of Statistical Software, 34(4), 1-29.

Tools

Reproducibility

https://www.ft.com/content/01fc06b8-fb6e-3e36-acb0-a1f8b47a7271

Tools

Reproducibility

Does High Public Debt Consistently
Stifle Economic Growth?
A Critique of Reinhart and Rogoff

Thomas Herndon, Michael Ash and Robert Pollin

April 2013

WORKINGPAPER SERIES
Number 322

POLITICAL ECONOMY RESEARCH INSTITUTE

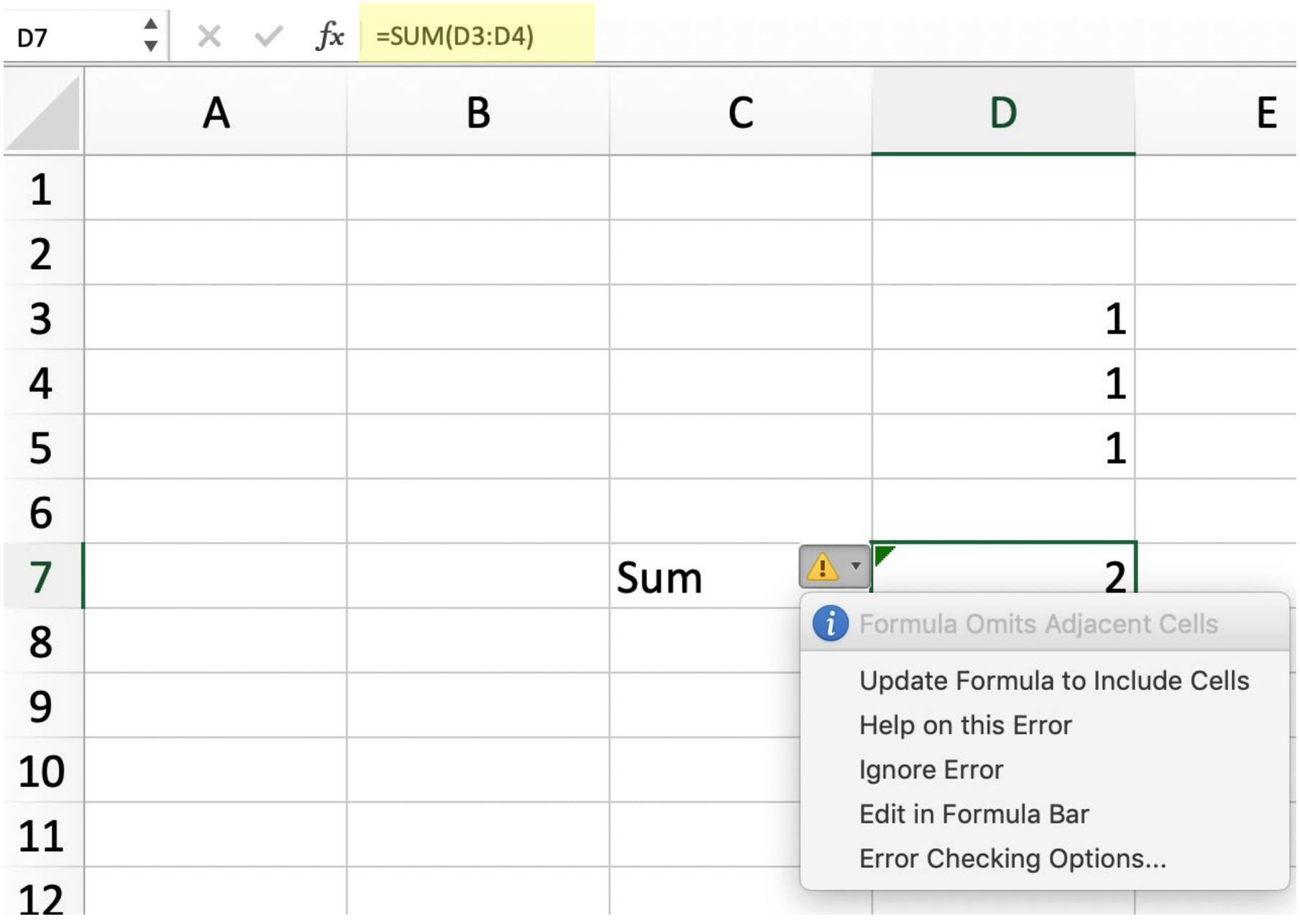
Gordon Hall
418 North Pleasant Street
Amherst, MA 01002

Phone: 413.545.6355
Fax: 413.577.0261
peri@econs.umass.edu
www.peri.umass.edu

"We replicate Reinhart and Rogoff (2010a and 2010b) and find that ***coding errors, selective exclusion of available data****, and unconventional weighting of summary statistics* ***lead to serious errors*** *that inaccurately represent the relationship between public debt and GDP growth among 20 advanced economies in the post-war period."*

Tools

Reproducibility



Tools

Data formats

BMC Part of Springer Nature | Explore Journals | Get Published | About BMC

BMC Bioinformatics

Home | About | Articles | Submission Guidelines

Correspondence | Open Access

Mistaken Identifiers: Gene name errors can be introduced inadvertently when using Excel in bioinformatics

Barry R Zeeberg †, Joseph Riss †, David W Kane, Kimberly J Bussey, Edward Uchio, W Marston Linehan, J Carl Barrett and John N Weinstein

†Contributed equally

BMC Bioinformatics 2004 **5**:80

https://doi.org/10.1186/1471-2105-5-80

Received: 05 March 2004 | **Accepted:** 23 June 2004 | **Published:** 23 June 2004

Abstract

Background

When processing microarray data sets, we recently noticed that some gene names were being changed inadvertently to non-gene names.

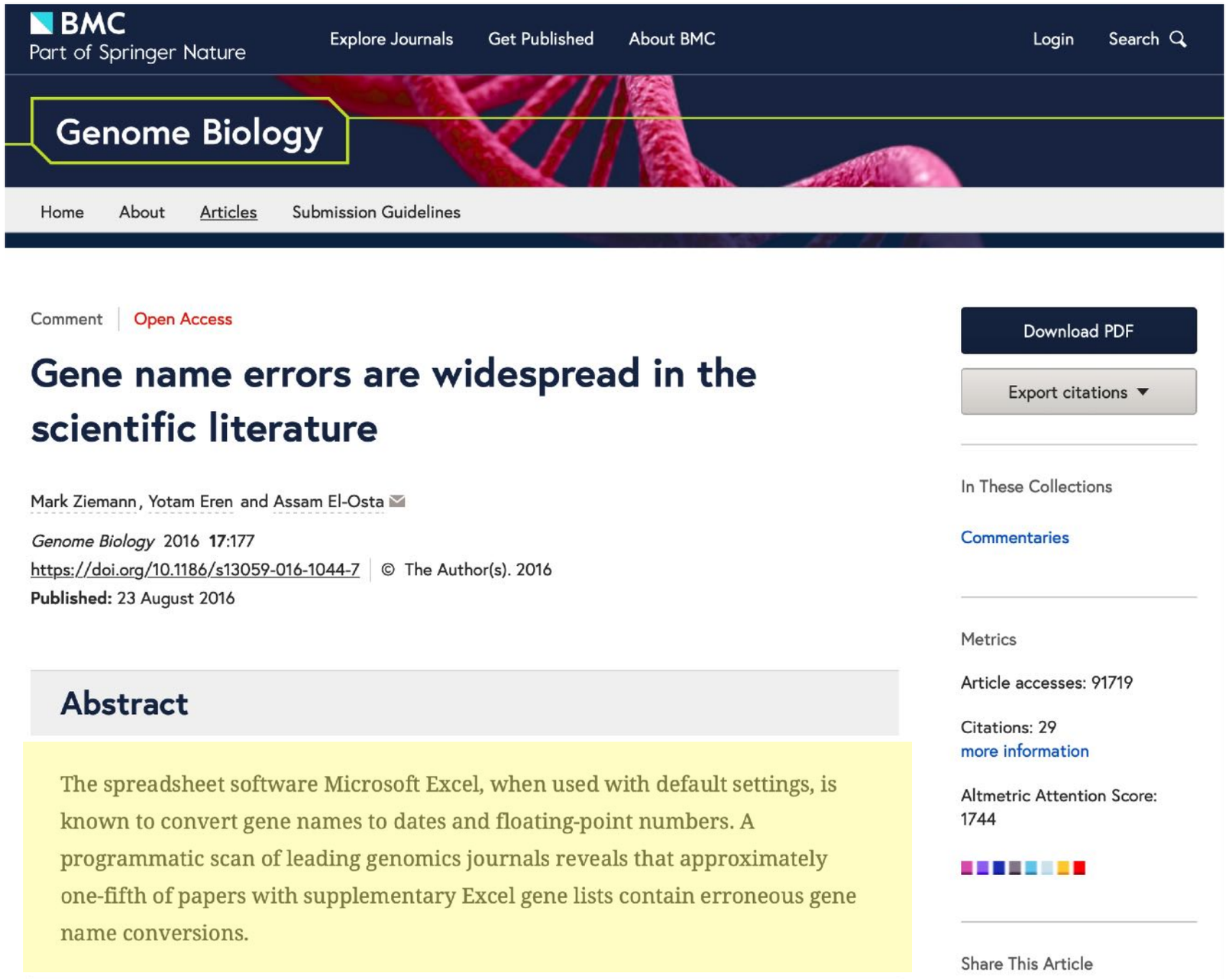
Results

A little detective work traced the problem to default date format conversions and floating-point format conversions in the very useful Excel program package. The date conversions affect at least 30 gene names; the floating-point conversions affect at least 2,000 if Riken identifiers are included. These conversions are irreversible; the original gene names cannot be recovered.

https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-5-80

Tools

Data formats



https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-1044-7

Tools

Bad practices

≃ $6b USD

"The London Whale"

2012

Tools

Bad practices

(…) Fourth, the model was approved despite observed operational problems. ***The Model Review Group noted that the VaR computation was being done on spreadsheets using a manual process and it was therefore "error prone" and "not easily scalable."*** *(…)*

From: Report of JPMorgan Chase & Co. Management Task Force Regarding 2012 CIO Losses

Tools

Bad practices

(…) Sixth, CIO's implementation of the model was flawed. CIO relied on the model creator, who reported to the front office, to operate the model. ***Data were uploaded manually without sufficient quality control****.* ***Spreadsheet-based calculations were conducted with insufficient controls and frequent formula and code changes were made****. Inadequate information technology resources were devoted to the process. Contrary to the action plan contained in the model approval, the process was never automated. (…)*

From: Report of JPMorgan Chase & Co. Management Task Force Regarding 2012 CIO Losses

Tools

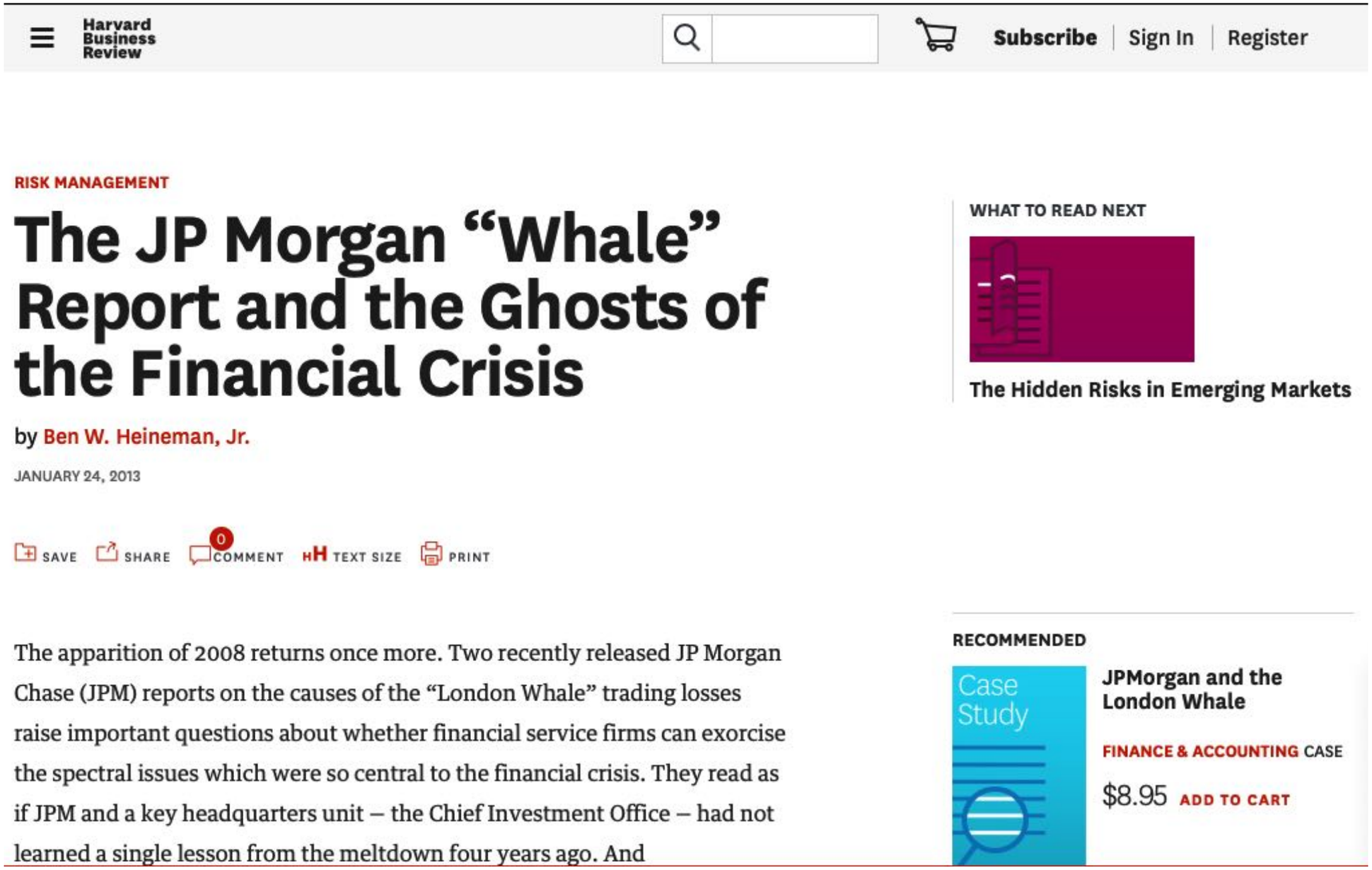
Bad practices

(…) Specifically, after subtracting the old rate from the new rate, the ***spreadsheet divided by their sum instead of their average****,* ***as the modeler had intended****. This error likely had the effect of muting volatility by a factor of two and of lowering the VaR, although it is unclear by exactly what amount, particularly given that it is unclear whether this error was present in the VaR calculation for every instrument, and that it would have been offset to some extent by correlation changes.* ***It also remains unclear when this error was introduced in the calculation.***

From: Report of JPMorgan Chase & Co. Management Task Force Regarding 2012 CIO Losses

Tools

Bad practices



https://hbr.org/2013/01/the-jp-morgan-whale-report-and

Tools

Notebooks

Literate programming

Interactivity

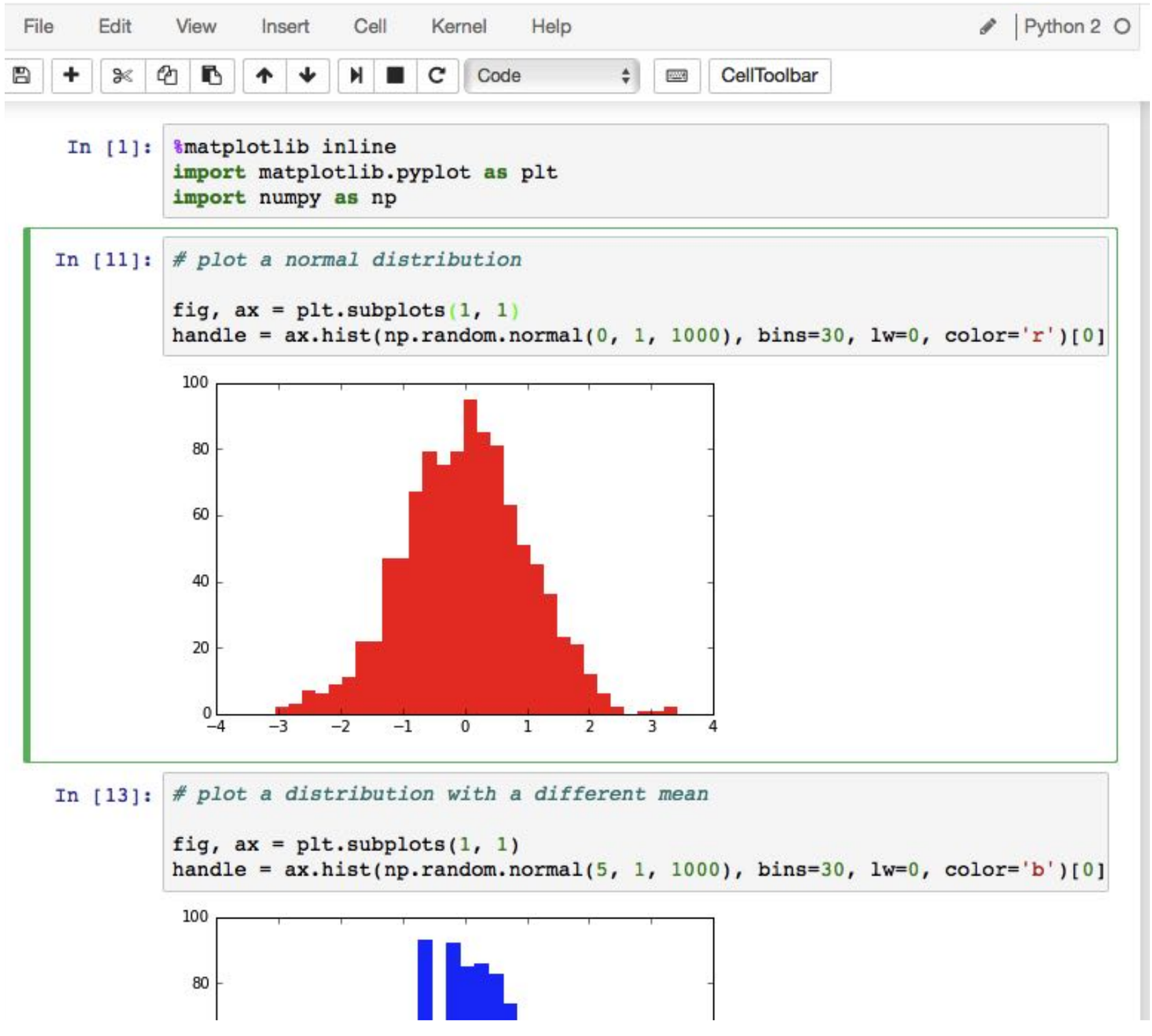
Different outputs from the same source document

Source control

Automatic report creation

Tools

Notebooks

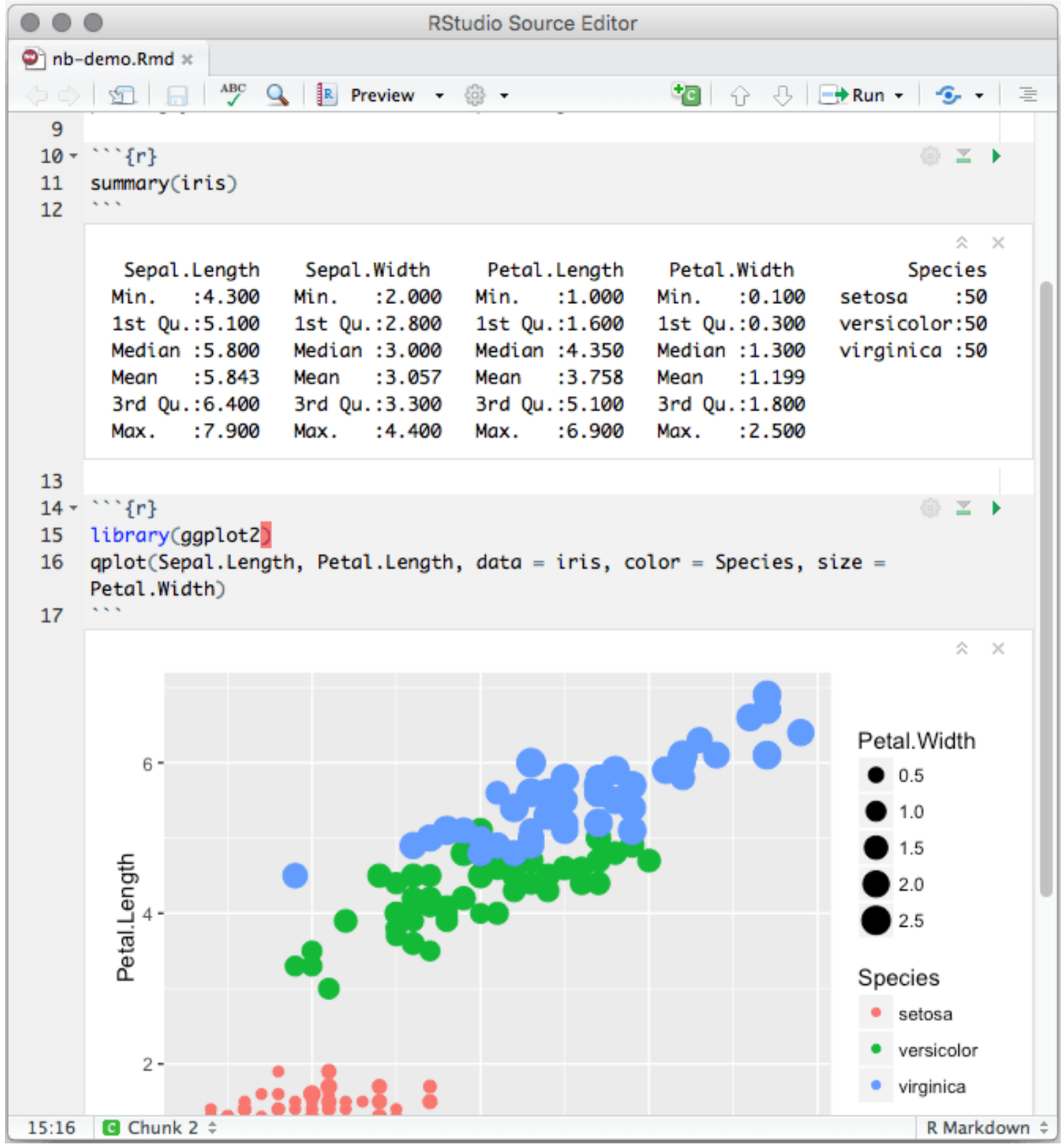


https://jupyter.org

Tools

Notebooks

R Notebooks



https://bookdown.org/yihui/rmarkdown/notebook.html

Tools

Revision control systems

https://git-scm.com

Tools

Revision control systems

GitLab

Tools

Tools

Only change files programmatically

Use checksum hashes to guarantee
the original files remains unaltered

All the examples are in R

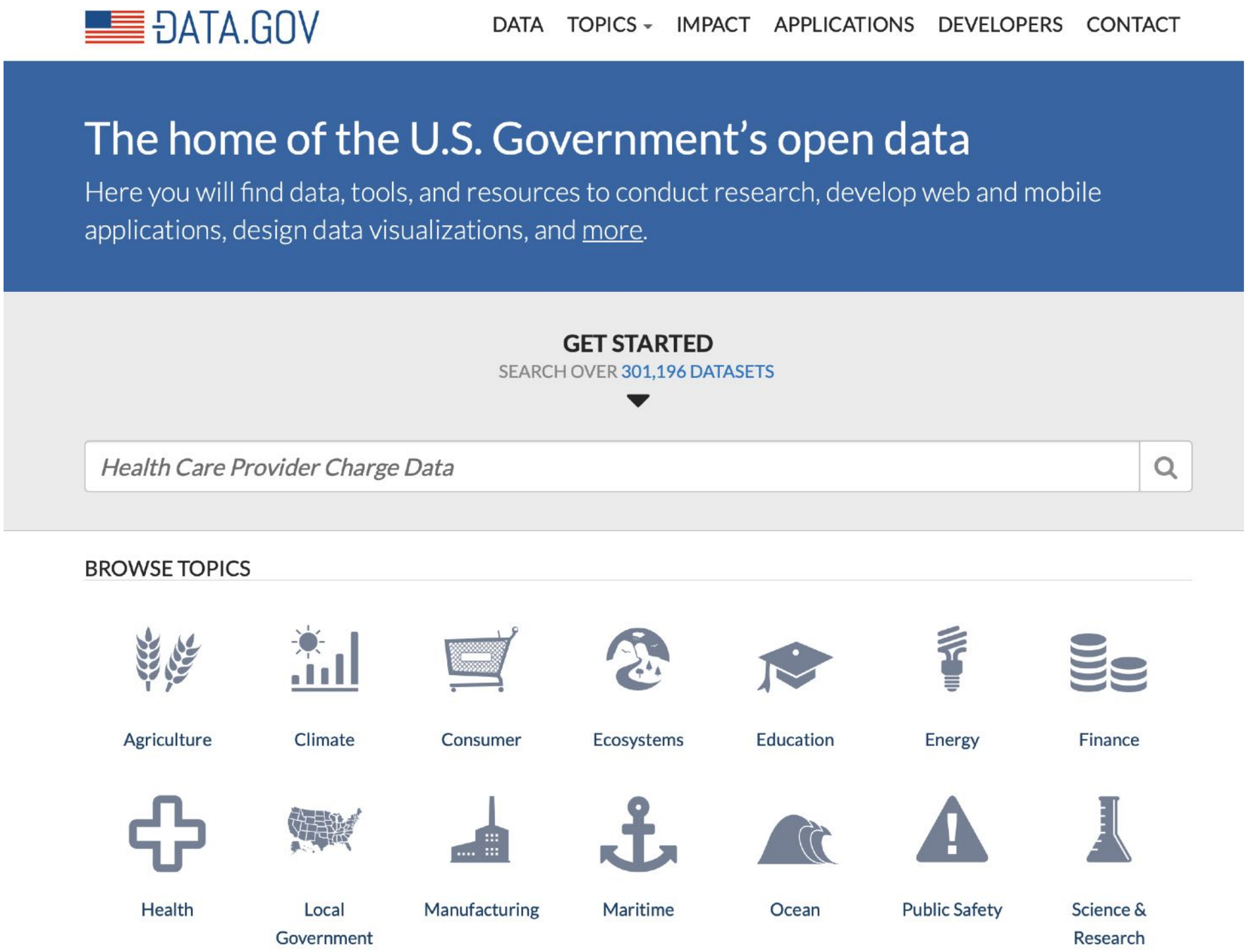
Data collection

Data collection

Data sources

Data collection

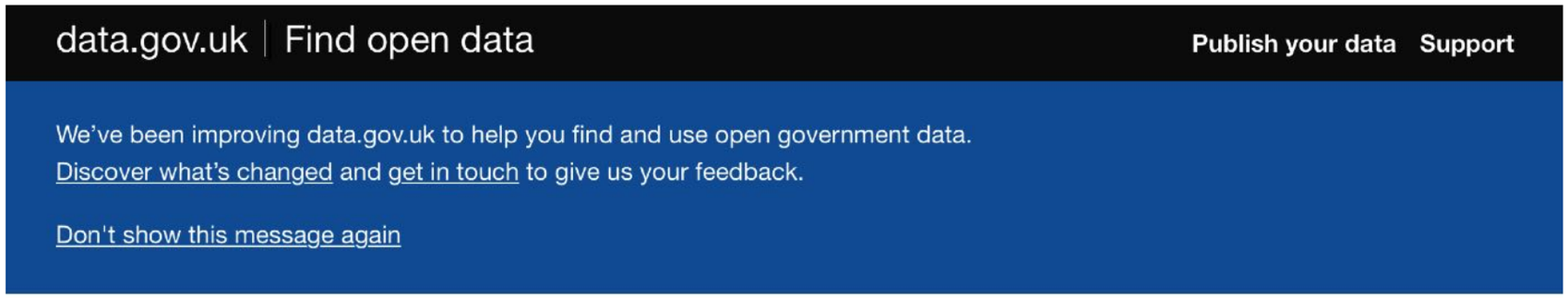
Data sources

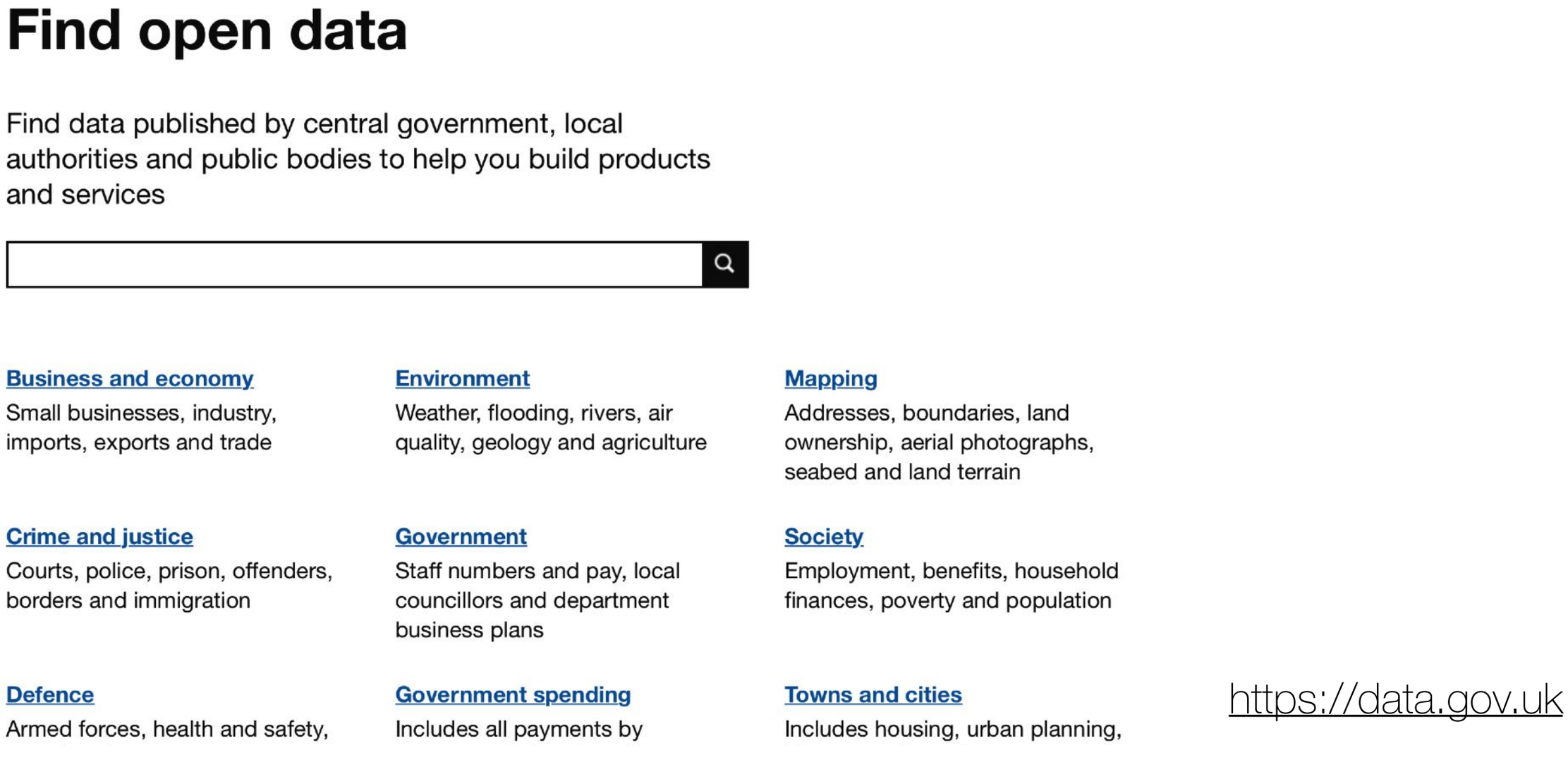


https://www.data.gov

Data collection

Data sources

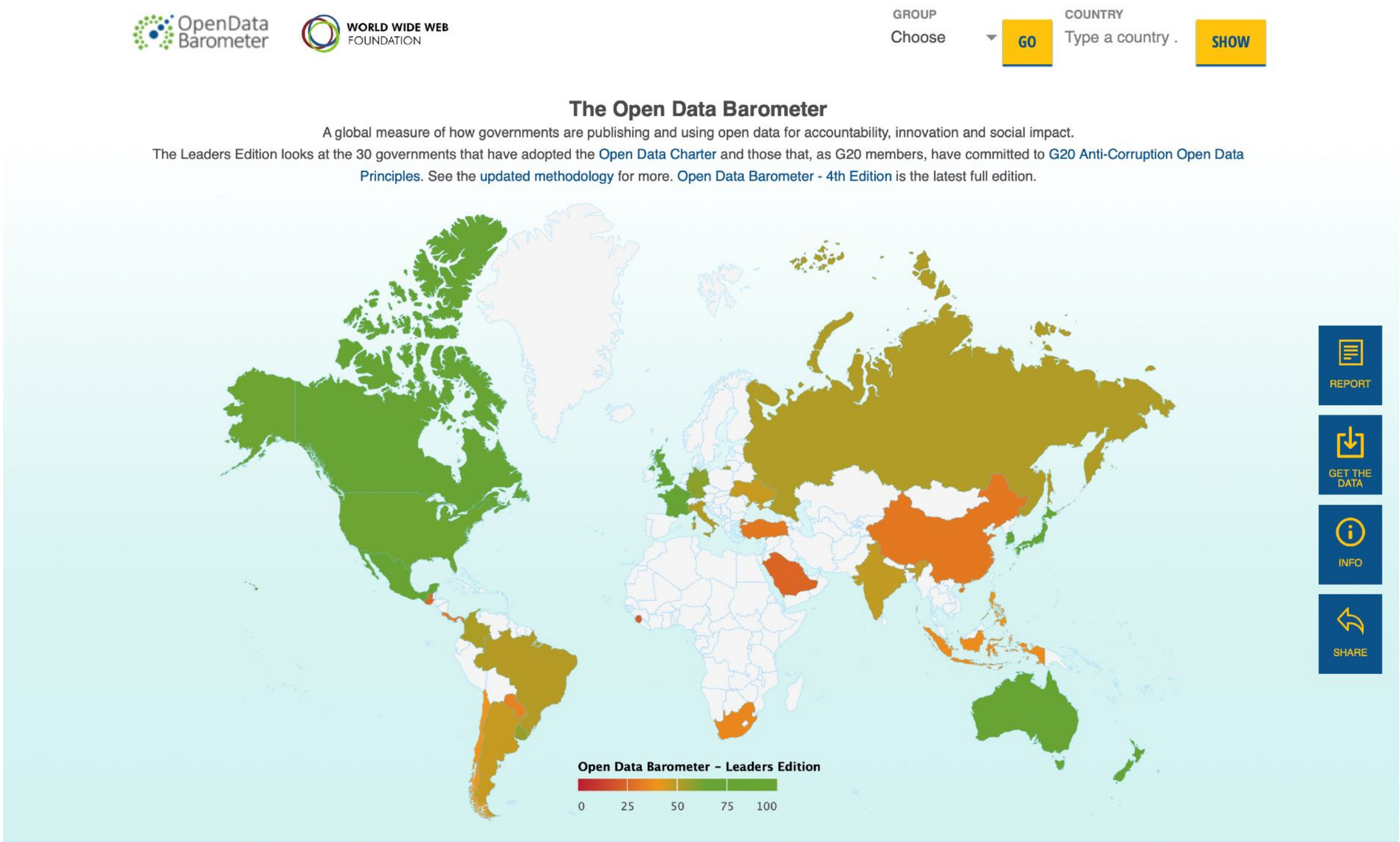




https://data.gov.uk

Data collection

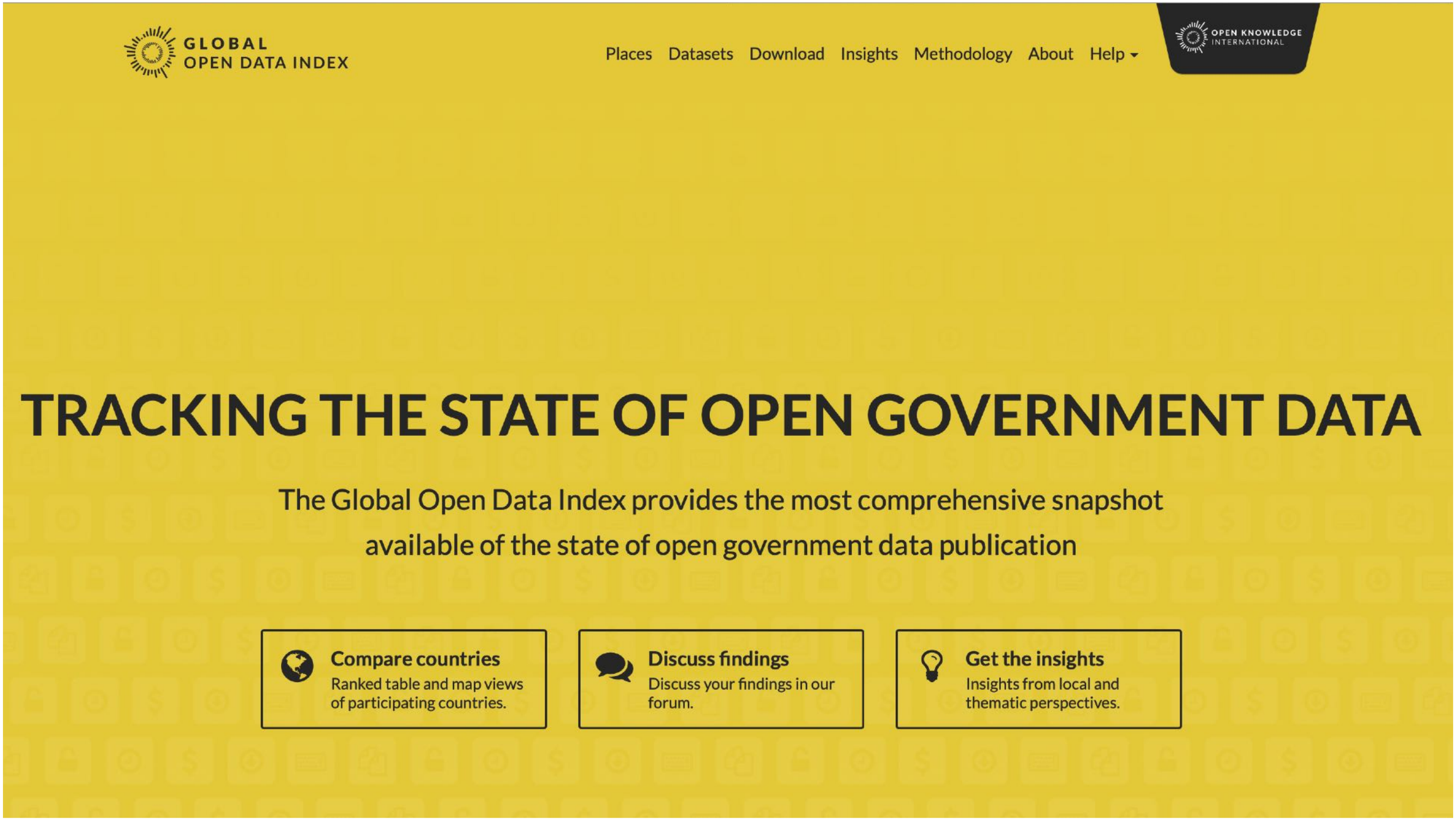
Data sources



https://opendatabarometer.org/

Data collection

Data sources



https://index.okfn.org

Data collection

Data sources

Registry of Open Data on AWS

About

This registry exists to help people discover and share datasets that are available via AWS resources. Learn more about sharing data on AWS.

See all usage examples for datasets listed in this registry.

Search datasets (currently 88 matching datasets)

Search datasets

Add to this registry

If you want to add a dataset or example of how to use a dataset to this registry, please follow the instructions on the Registry of Open Data on AWS GitHub repository.

Unless specifically stated in the applicable dataset documentation, datasets available through the Registry of Open Data on AWS are not provided and maintained by AWS. Datasets are provided and maintained by a variety of third parties under a variety of licenses. Please check dataset licenses and related documentation to determine if a dataset may be used for your application.

Sentinel-2

earth observation | satellite imagery | gis | natural resource | sustainability | disaster response

The Sentinel-2 mission is a land monitoring constellation of two satellites that provide high resolution optical imagery and provide continuity for the current SPOT and Landsat missions. The mission provides a global coverage of the Earth's land surface every 5 days, making the data of great use in on-going studies. L1C data are available from June 2015 globally. L2A data are available from April 2017 over wider Europe region, planned to be expanded globally in July 2018.

Details →

Usage examples

- EOS Land Viewer by Earth Observing System
- Sentinel Playground by Sinergise
- EO Browser by Sinergise
- FME Landsat-8/Sentinel-2 File Selector by Safe Software
- Using Vector tiles and AWS Lambda, we can build a really simple API to get Landsat and Sentinel images by Remote Pixel

See 16 usage examples →

Landsat 8

earth observation | satellite imagery | gis | natural resource | sustainability | disaster response

An ongoing collection of satellite imagery of all land on Earth produced by the Landsat 8 satellite.

Details →

Usage examples

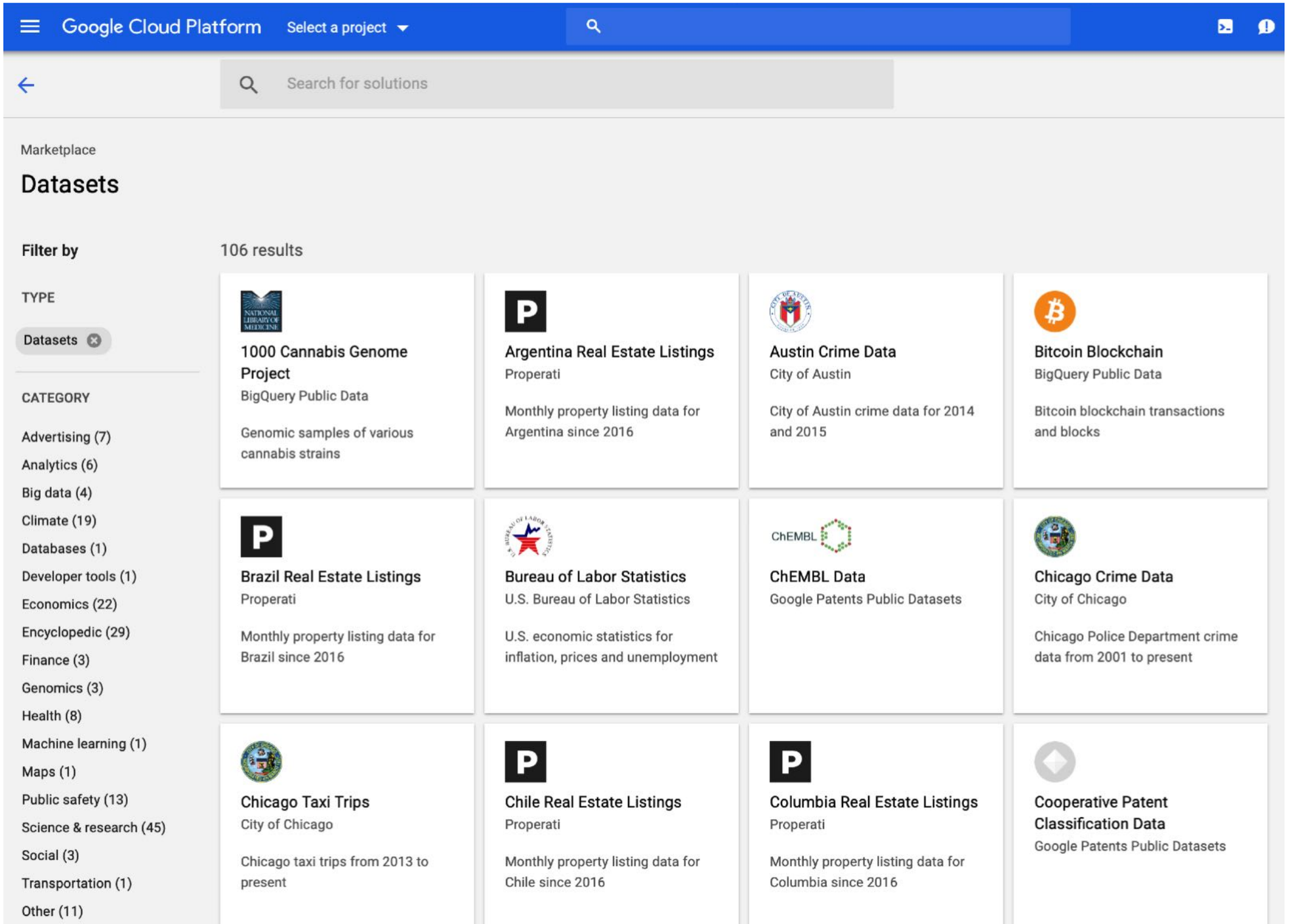
- Exploring the Chile wildfires with Landsat and Sentinel-2 imagery by Timothy Whitehead
- Development Seed Geolambda by Matthew Hanson
- Integrate imagery from the full Landsat archive into your own apps, maps, and analysis with Landsat image services by Esri
- FME Landsat-8/Sentinel-2 File Selector by Safe Software
- Sentinel Playground for Landsat by Sinergise

See 13 usage examples →

https://registry.opendata.aws

Data collection

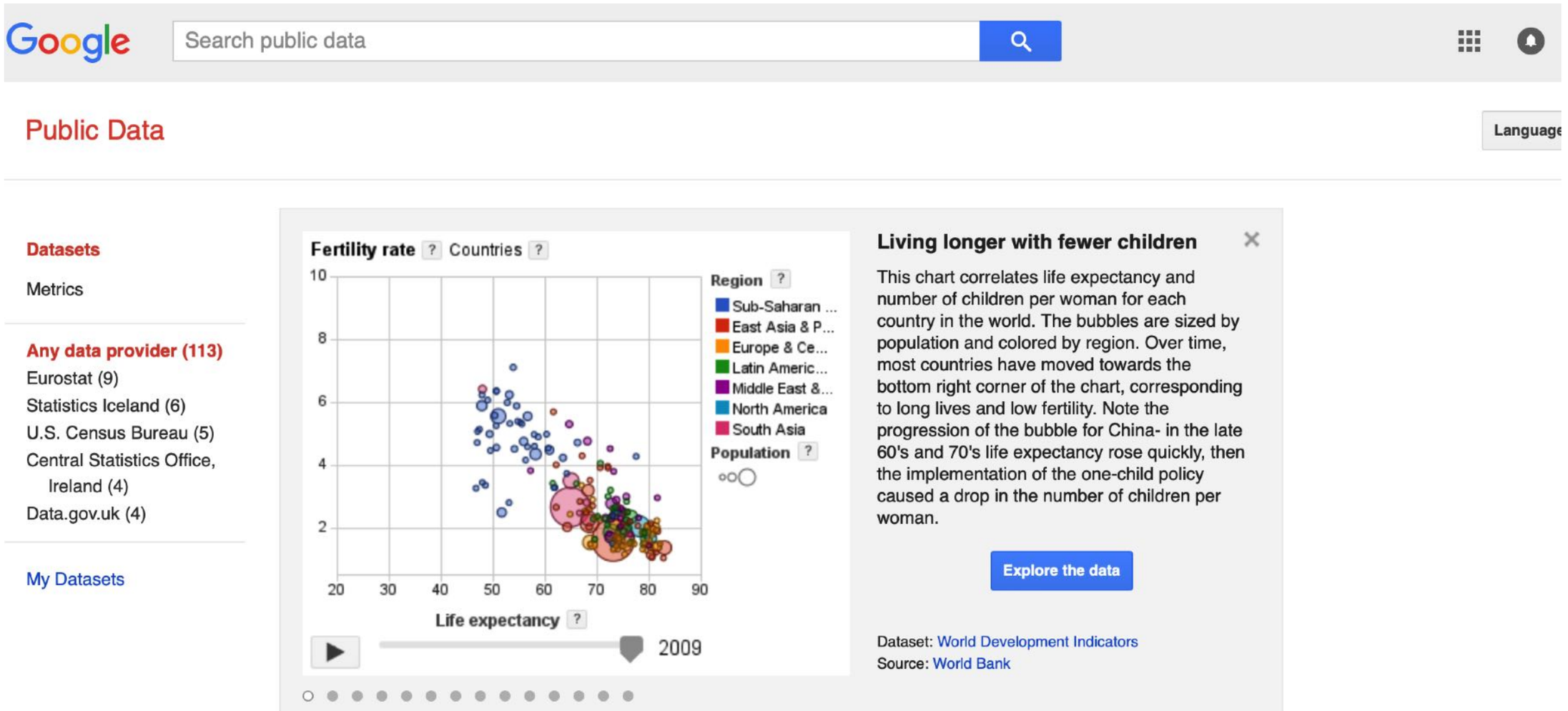
Data sources



https://console.cloud.google.com/marketplace/browse?filter=solution-type:dataset

Data collection

Data sources



https://www.google.com/publicdata/directory

Data collection

Data sources

Awesome Public Datasets

awesome

NOTICE: This repo is automatically generated by apd-core. Please **DO NOT** modify this file directly. We have provided a new way to contribute to Awesome Public Datasets. The original PR entrance directly on repo is closed forever.

- I am well.
- Please fix me.

This list of a topic-centric public data sources in high quality. They are collected and tidied from blogs, answers, and user responses. Most of the data sets listed below are free, however, some are not. Other amazingly awesome lists can be found in sindresorhus's awesome list.

Table of Contents

https://github.com/awesomedata/awesome-public-datasets

Data collection

Data sources

https://dados.gov.pt/pt/

Data collection

Consuming text files

Consuming text files

Text is one of the most reliable and common format to share files

Consuming text files

They can be:

- **csv** - comma separated file
- **tsk** - tab separated file
- **fwf** - fixed width file
- or use another delimiter

Consuming text files

Potential problems

Different file encoding your program expects

Quoted/unquoted strings

Comments in the records

Different number of columns per row

Different line break characters

Invisible characters

Consuming text files

Potential problems

```
read_delim(file, delim, quote = "\"", escape_backslash = FALSE,
  escape_double = TRUE, col_names = TRUE, col_types = NULL,
  locale = default_locale(), na = c("", "NA"), quoted_na = TRUE,
comment = "", trim_ws = FALSE, skip = 0, n_max = Inf,
guess_max = min(1000, n_max), progress = show_progress(),
skip_empty_rows = TRUE)
```

Consuming text files

Potential problems

Different line break characters

```
dos2unix
```

```
tr -d "\015" < file_name > new_file_name
```

Consuming text files

Tools

Command line tool to work with csv files

`csvkit`

https://csvkit.readthedocs.io/en/0.9.0/index.html#

Consuming text files

Tools

`tail`

`head`

`wc`

`tr`

`sed`

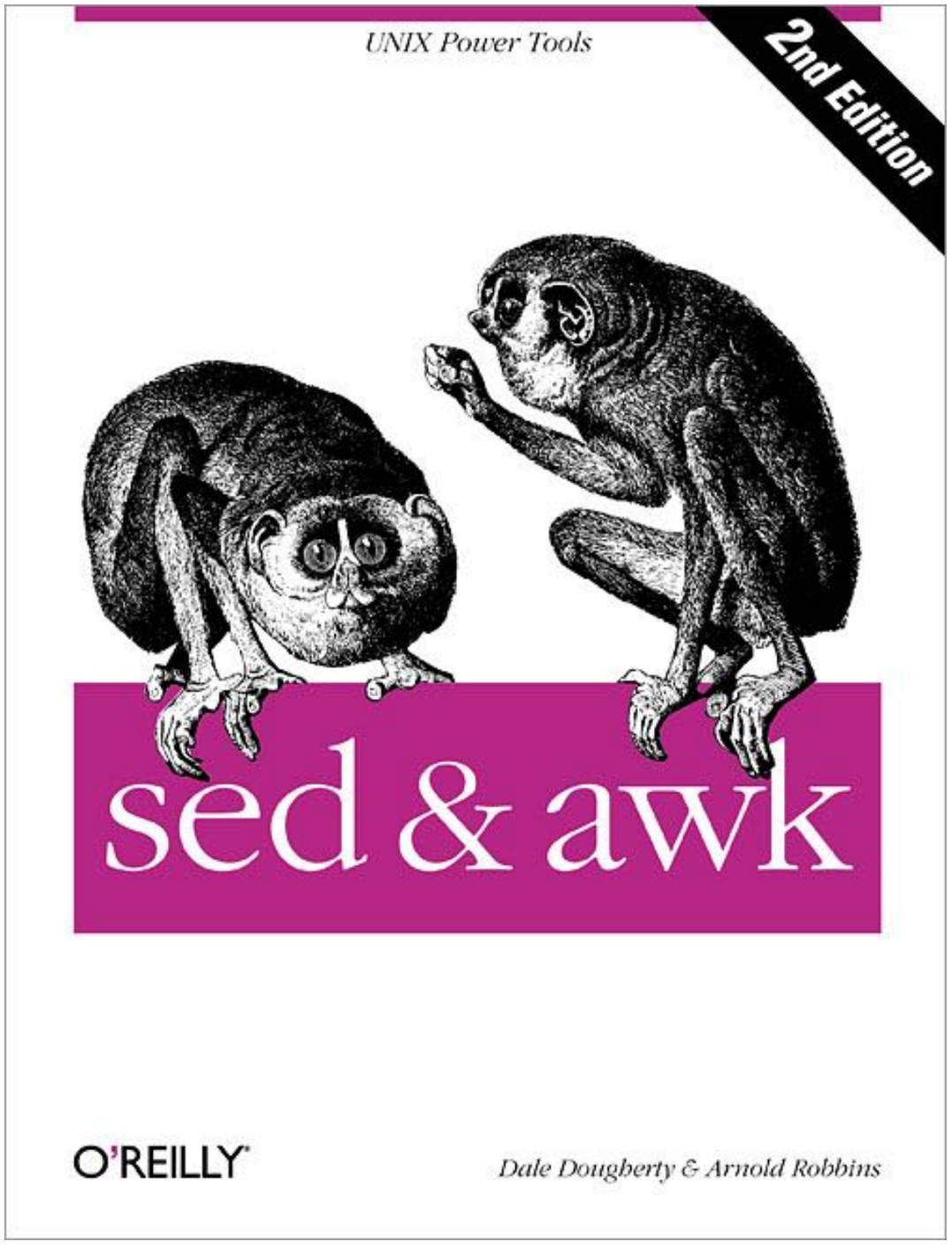
`cut`

`grep`

`awk`

Consuming text files

Tools

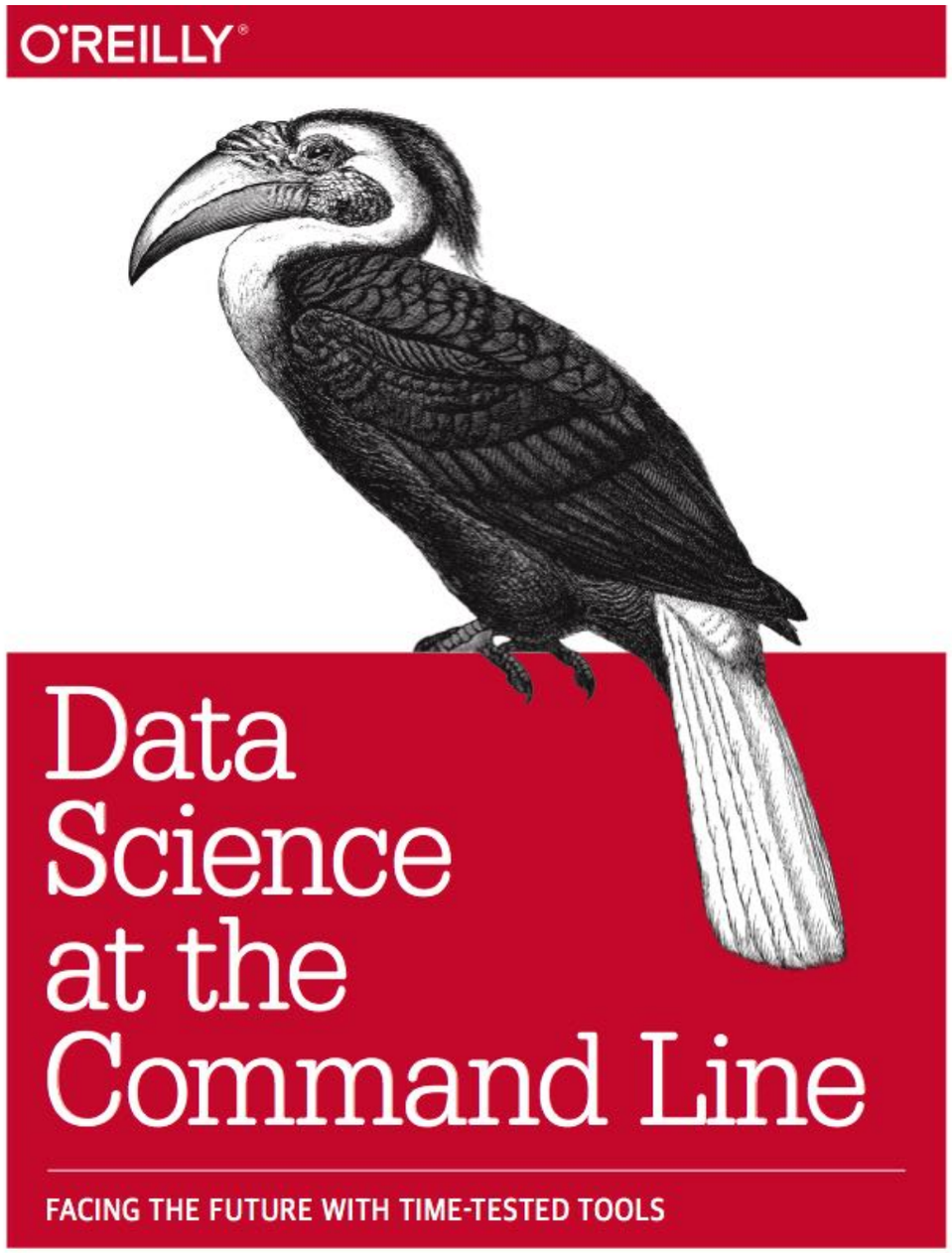


https://www.oreilly.com/library/view/sed-awk/1565922255/

Consuming text files

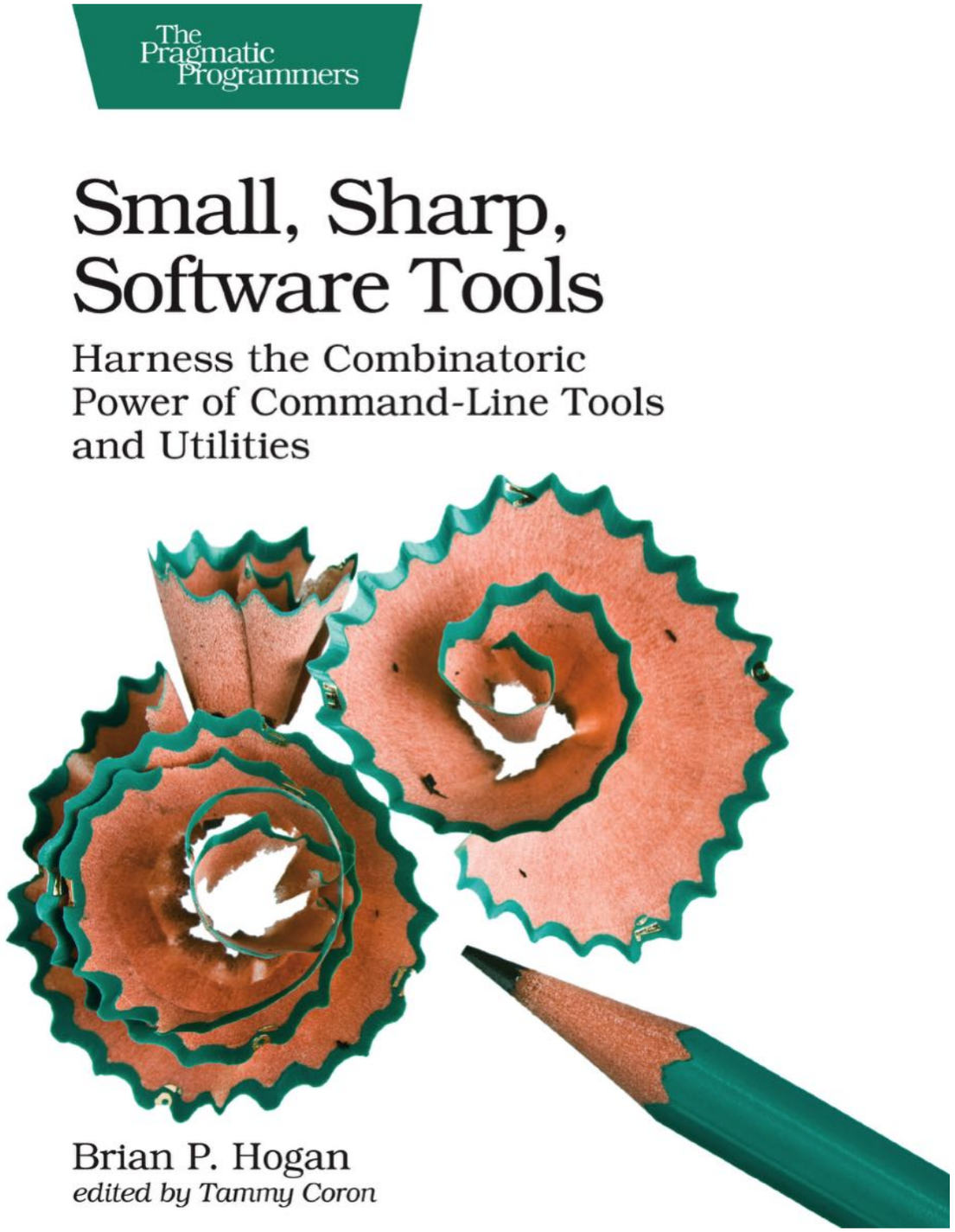
Tools

https://www.datascienceatthecommandline.com



Consuming text files

Tools



https://pragprog.com/book/bhcldev/small-sharp-software-tools

Consuming text files

Example

https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html

Consuming text files

Example

Part D Prescriber Detailed Data for 2013

Original compressed file: **532 MB**

Uncompressed file: **2.8 GB**

Consuming text files

Example

The results depend on:

- the hardware
- operating system
- system load
- and also the data itself

Consuming text files

Example

```
wc -l PartD_Prescriber_PUF_NPI_Drug_13.txt
```

```
23645874 PartD_Prescriber_PUF_NPI_Drug_13.txt
```

23 645 873 records

2.8 GB

Consuming text files

Example

head PartD_Prescriber_PUF_NPI_Drug_13.txt

```
† ~/Documents/work/uminho/aulas/2018-2019/workshop-bp/data/PartD_Prescriber_PUF_NPI_DRUG_13 hea
d PartD_Prescriber_PUF_NPI_Drug_13.txt
npi     nppes_provider_last_org_name    nppes_provider_first_name       nppes_provider_city     nppe
s_provider_state        specialty_description   description_flag        drug_name       generic_name
bene_count      total_claim_count       total_30_day_fill_count total_day_supply        total_drug_c
ost     bene_count_ge65 bene_count_ge65_suppress_flag   total_claim_count_ge65  ge65_suppress_flag t
otal_30_day_fill_count_ge65     total_day_supply_ge65   total_drug_cost_ge65
1003000126      ENKESHAFI       ARDALAN CUMBERLAND      MD      Internal Medicine       S       ISOS
ORBIDE MONONITRATE ER   ISOSORBIDE MONONITRATE          11      11      307     171.59          *  *
1003000126      ENKESHAFI       ARDALAN CUMBERLAND      MD      Internal Medicine       S       LEVO
FLOXACIN        LEVOFLOXACIN    26      26      26      165     227.1   15              15          1
5       106     159.72
1003000126      ENKESHAFI       ARDALAN CUMBERLAND      MD      Internal Medicine       S       LISI
NOPRIL  LISINOPRIL      17      19      19      570     100.37          #               #
1003000126      ENKESHAFI       ARDALAN CUMBERLAND      MD      Internal Medicine       S       METO
PROLOL TARTRATE METOPROLOL TARTRATE     28      30      31      916     154.65          #           #
1003000126      ENKESHAFI       ARDALAN CUMBERLAND      MD      Internal Medicine       S       PRED
NISONE  PREDNISONE      14      14      14      133     44.72           *               *
1003000126      ENKESHAFI       ARDALAN CUMBERLAND      MD      Internal Medicine       S       SIMV
ASTATIN SIMVASTATIN     16      17      17      487     102.06          #               #
1003000126      ENKESHAFI       ARDALAN CUMBERLAND      MD      Internal Medicine       S       WARF
ARIN SODIUM     WARFARIN SODIUM 11      11      13.6    324     165.45          *               *
1003000142      KHALIL  RASHID  TOLEDO  OH      Anesthesiology  S       ACETAMINOPHEN-CODEINE   ACET
AMINOPHEN WITH CODEINE  11      22      22      563     242.98          #               #
1003000142      KHALIL  RASHID  TOLEDO  OH      Anesthesiology  S       BACLOFEN        BACLOFEN    1
5       15      450     139.81          *               *
† ~/Documents/work/uminho/aulas/2018-2019/workshop-bp/data/PartD_Prescriber_PUF_NPI_DRUG_13
```

Consuming text files

Example

tail PartD_Prescriber_PUF_NPI_Drug_13.txt

```
† ~/Documents/work/uminho/aulas/2018-2019/workshop-bp/data/PartD_Prescriber_PUF_NPI_DRUG_13 tai
l PartD_Prescriber_PUF_NPI_Drug_13.txt
1992999825      DESCHENES       GEOFFREY        SEATTLE WA      Otolaryngology  S       AMOX TR-POTA
SSIUM CLAVULANATE       AMOXICILLIN/POTASSIUM CLAV      14      16      16      195     378.51      #
#
1992999825      DESCHENES       GEOFFREY        SEATTLE WA      Otolaryngology  S       FLUTICASONE
PROPIONATE      FLUTICASONE PROPIONATE  38      74      84      2491    2388.37         #       62 7
2       2131    2023.8
1992999825      DESCHENES       GEOFFREY        SEATTLE WA      Otolaryngology  S       IPRATROPIUM
BROMIDE IPRATROPIUM BROMIDE             27      30.5    755     619.17          *       27          3
0.5     755     619.17
1992999825      DESCHENES       GEOFFREY        SEATTLE WA      Otolaryngology  S       OMEPRAZOLE O
MEPRAZOLE               34      49      1442    433.48          *       19              33      9903
32.82
1992999825      DESCHENES       GEOFFREY        SEATTLE WA      Otolaryngology  S       OXYCODONE-AC
ETAMINOPHEN     OXYCODONE HCL/ACETAMINOPHEN     13      13      13      69      84.1            *  *
1992999833      SHAW    L. NOAH NEW YORK        NY      Psychoanalyst   T       SELEGILINE HCL  SELE
GILINE HCL              11      11      330     563.03          *       11              11      3305
63.03
1992999833      SHAW    L. NOAH NEW YORK        NY      Psychoanalyst   T       STALEVO 100     CARB
IDOPA/LEVODOPA/ENTACAPONE               11      11      330     6929.04         *       11          1
1       330     6929.04
1992999874      JOFFE   GABRIELLA       MECHANICSVILLE  VA      Internal Medicine       S       LEVO
FLOXACIN        LEVOFLOXACIN    17      17      17      102     181.09          #               #
1992999874      JOFFE   GABRIELLA       MECHANICSVILLE  VA      Internal Medicine       S       PANT
OPRAZOLE SODIUM PANTOPRAZOLE SODIUM             12      12      360     120.78          *       12 1
2       360     120.78
1992999874      JOFFE   GABRIELLA       MECHANICSVILLE  VA      Internal Medicine       S       PRED
NISONE  PREDNISONE      16      16      18      247     93.47           #               #
† ~/Documents/work/uminho/aulas/2018-2019/workshop-bp/data/PartD_Prescriber_PUF_NPI_DRUG_13
```

Consuming text files

Example

Reading text files

Consuming text files

Example

Using the readr package:

```
prescriber_data_readr <- read_tsv(file = "../data/
PartD_Prescriber_PUF_NPI_DRUG_13/
PartD_Prescriber_PUF_NPI_DRUG_13.txt")
```

```
   user  system elapsed
 60.197   6.876  82.337
```

Consuming text files

Example

Using the data.table package:

```
prescriber_data_fread <- fread(file = "../data/
PartD_Prescriber_PUF_NPI_DRUG_13/
PartD_Prescriber_PUF_NPI_DRUG_13.txt")
```

```
   user  system elapsed
 41.217   9.128  25.446
```

Consuming text files

Example

Saving data in a binary format

Writing binary files

Example

Using the default R compressed file format:

```
saveRDS(prescriber_data_readr,
        file="../data/prescriber_c.rds")
```

```
   user  system elapsed
215.866   5.148 223.887
```

527MB

Writing binary files

Example

Using the default R uncompressed file format:

```
saveRDS(prescriber_data_readr,
        file="../data/prescriber_u.rds",
        compress = FALSE)
```

```
 user  system elapsed
65.163  12.063  80.531
```

5.3GB

Writing binary files

fst package

“The fst package for R provides a fast, easy and flexible way to serialize data frames.

With access speeds of multiple GB/s, fst is specifically designed to unlock the potential of high speed solid state disks that can be found in most modern computers.

Data frames stored in the fst format have full random access, both in column and rows.”

http://www.fstpackage.org

Writing binary files

fst package

Using fst file format with the default compression:

```
write.fst(prescriber_data_readr,
          path="../data/prescriber_c.fst")
```

```
 user  system elapsed
14.177   4.898  13.372
```

2.3 GB

Writing binary files

fst package

Using the uncompressed fst file format:

```
write.fst(prescriber_data_readr,
          path="../data/prescriber_u.fst",
          compress = 0)
```

```
 user  system elapsed
26.165  16.720  27.069
```

4.5 GB

Writing binary files

Feather package

“Feather provides binary columnar serialization for data frames.

It is designed to make reading and writing data frames efficient, and to make sharing data across data analysis languages easy.”

No data compression!

https://github.com/wesm/feather

Writing binary files

Feather package

```
write_feather(prescriber_data_readr,
              path="../data/prescriber.feather")
```

```
user  system elapsed
7.571   3.756  12.975
```

4.4 GB

Reading and writing binary files

Example

Reading data in a binary format

Reading binary files

Example

Using the compressed R file format:

```
aux <- readRDS(file =
                 "../data/prescriber_c.rds")
```

```
 user  system elapsed
79.030   1.936  82.347
```

527MB

Reading binary files

Example

Using the uncompressed R file format:

```
aux <- readRDS(file =
                "../data/prescriber_u.rds")
```

```
 user  system elapsed
79.859   5.630  90.014
```

5.3 GB

Reading binary files

Example

Using the compressed fst file format:

```
aux <- read_fst(path =
                "../data/prescriber_c.fst")
```

```
   user  system elapsed
 16.754   3.888  16.950
```

2.3 GB

Reading binary files

Example

Using the uncompressed fst file format:

```
aux <- read_fst(path =
                “../data/prescriber_u.fst")
```

```
 user  system elapsed
11.033   4.555  17.682
```

4.5 GB

Reading binary files

Example

Using the feather file format:

```
aux <- read_feather(path =
                    "../data/prescriber.feather")
```

```
   user  system elapsed
 14.996   9.072  47.238
```

4.4 GB

Reading and writing binary files

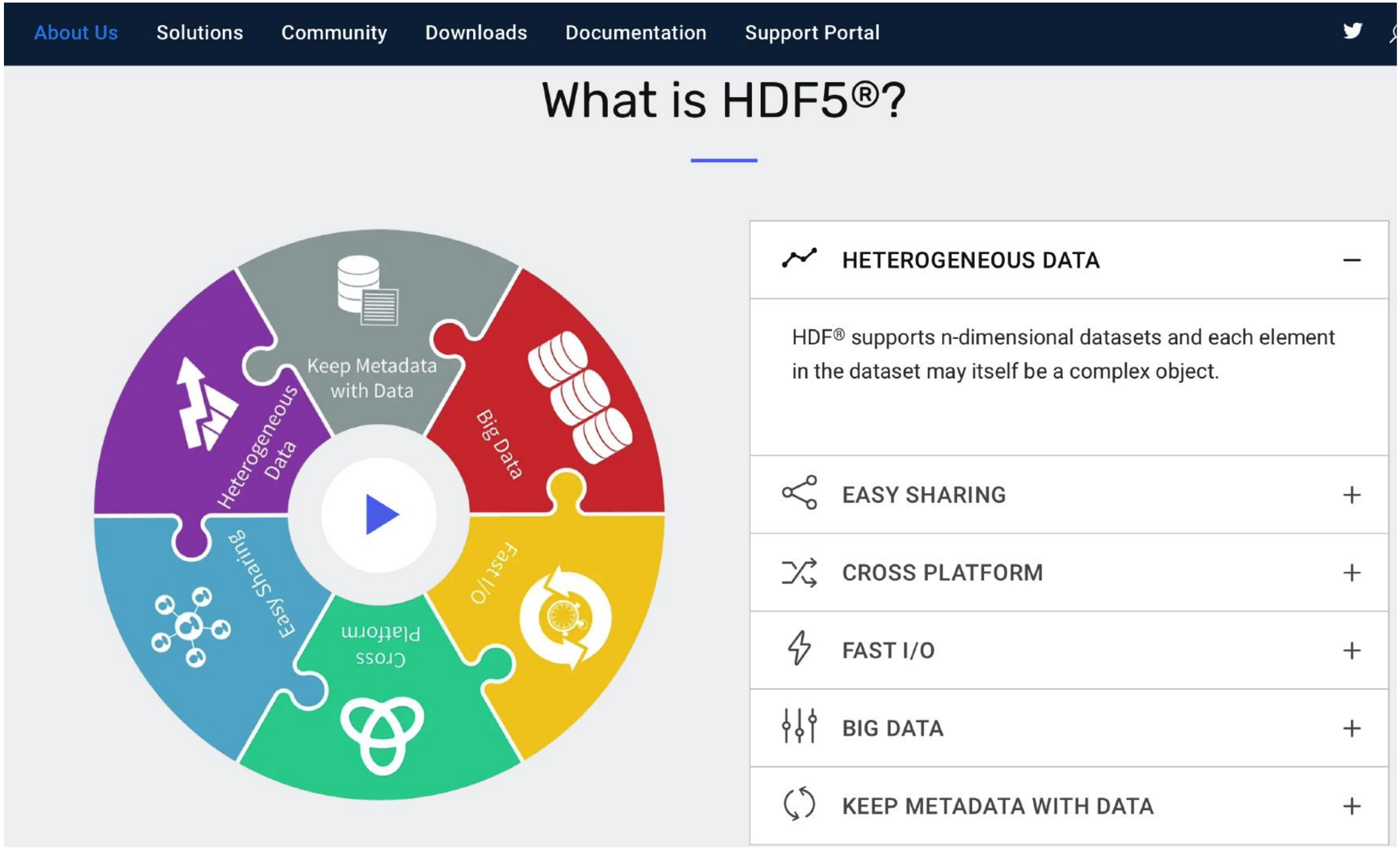
Example

Summary:

	Write	Read	File size
R format compressed	224s	82s	527 MB
R format uncompressed	81s	90s	5.3 GB
fst compressed	13s	17s	2.3 GB
fst uncompressed	27s	18s	4.5 GB
feather	13s	47s	4.4 GB

Consuming text files

Additional tools



https://www.hdfgroup.org/solutions/hdf5/

Data collection

Web data sources

Data collection

Web data sources

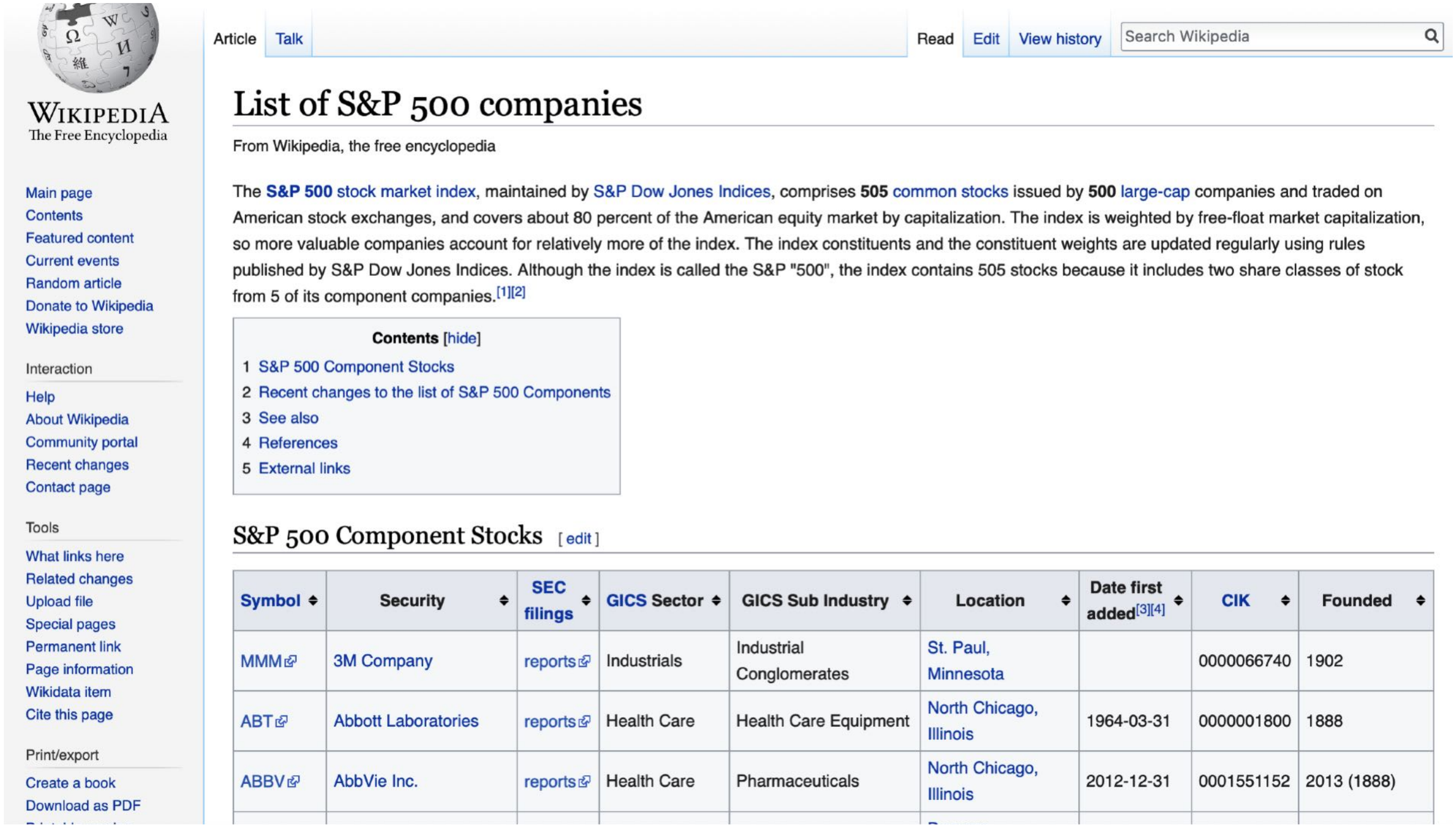
Web scraping

Web scraping

Friendly URLs with data in tables

Web scraping

Friendly URLs



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF

Article | Talk | Read | Edit | View history | Search Wikipedia

List of S&P 500 companies

From Wikipedia, the free encyclopedia

The **S&P 500** stock market index, maintained by S&P Dow Jones Indices, comprises **505** common stocks issued by **500** large-cap companies and traded on American stock exchanges, and covers about 80 percent of the American equity market by capitalization. The index is weighted by free-float market capitalization, so more valuable companies account for relatively more of the index. The index constituents and the constituent weights are updated regularly using rules published by S&P Dow Jones Indices. Although the index is called the S&P "500", the index contains 505 stocks because it includes two share classes of stock from 5 of its component companies.[1][2]

Contents [hide]

1 S&P 500 Component Stocks
2 Recent changes to the list of S&P 500 Components
3 See also
4 References
5 External links

S&P 500 Component Stocks [edit]

Symbol	Security	SEC filings	GICS Sector	GICS Sub Industry	Location	Date first added[3][4]	CIK	Founded
MMM	3M Company	reports	Industrials	Industrial Conglomerates	St. Paul, Minnesota		0000066740	1902
ABT	Abbott Laboratories	reports	Health Care	Health Care Equipment	North Chicago, Illinois	1964-03-31	0000001800	1888
ABBV	AbbVie Inc.	reports	Health Care	Pharmaceuticals	North Chicago, Illinois	2012-12-31	0001551152	2013 (1888)

https://en.wikipedia.org/wiki/List_of_S%26P_500_companies

Web scraping

Friendly URLs

```
library(rvest)

page <- read_html("https://en.wikipedia.org/wiki/
List_of_S%26P_500_companies")
```

```
constituents <- page %>% html_nodes("table") %>%
  .[[1]] %>%
  html_table()
```

Web scraping

Friendly URLs

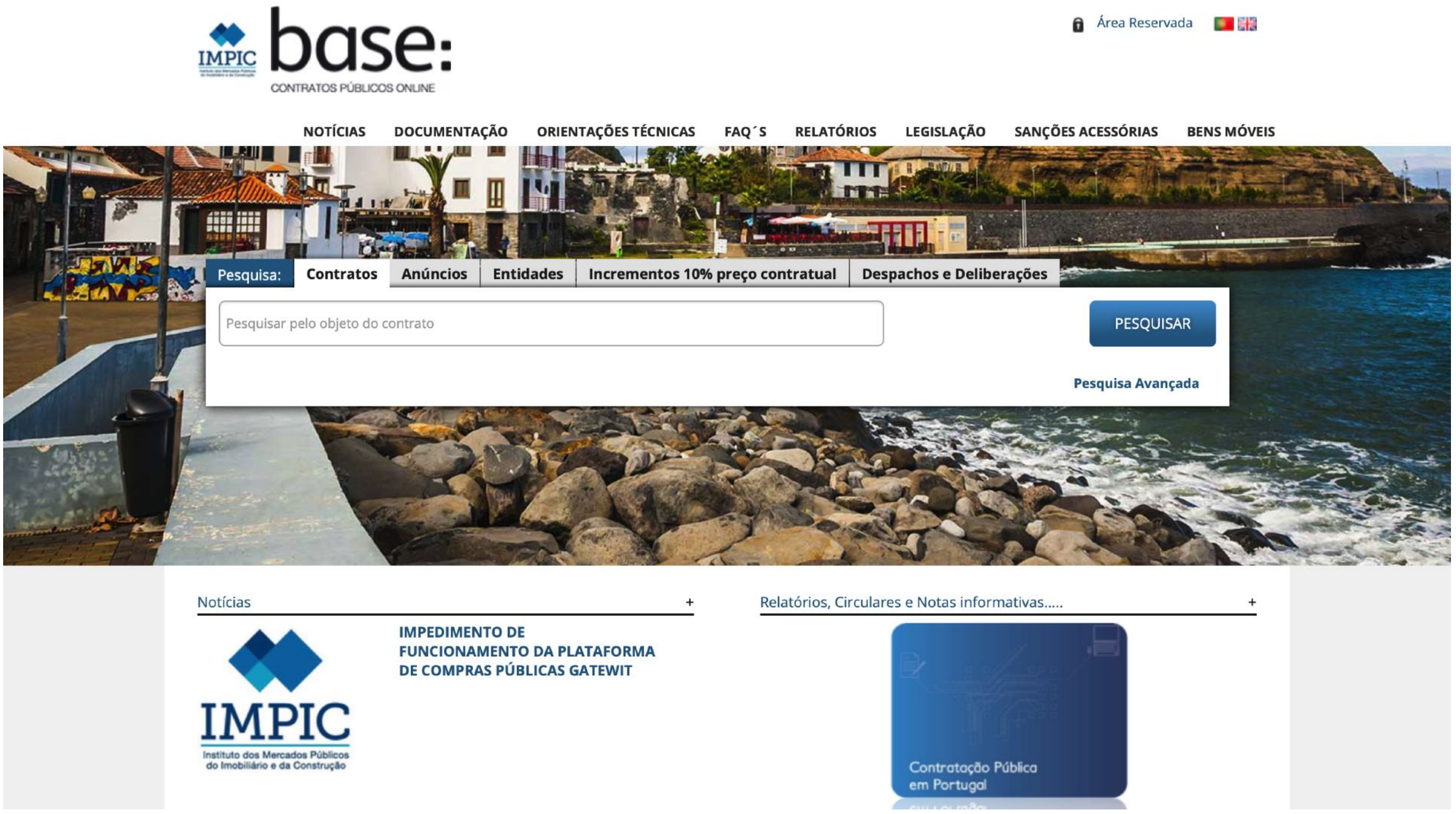
```
library(rvest)

page <- read_html("https://en.wikipedia.org/wiki/
List_of_S%26P_500_companies")
```

```
constituents <- page %>% html_nodes("table") %>%
  .[[1]] %>%
  html_table()
```

Web scraping

Friendly URLs



http://www.base.gov.pt/Base/pt/Homepage

Web scraping

Friendly URLs

Resultados

Foram encontrados **958252** resultados para a sua pesquisa.

Pesquisou por:

Exportar Resultados (Excel)

Objeto do Contrato	Preço contratual	Publicação	Adjudicante	Adjudicatário	
Construção de passeios e muros e tratamento de águas pluviais...	23.970,77 €	17-12-2018	Freguesia de Meirinhas	Construções da Cancelinha, Lda	+
Prestação de Serviços Médicos no Serviço de Neuropediatria - C.P....	36.639,00 €	17-12-2018	Hospital Garcia de Orta, E. P. E.	JOÃO NUNO FERREIRA DE CARVALHO	+
Aquisição de Equipamento para infraestrutura Wireless em Escolas e Jardins...	74.901,15 €	17-12-2018	Município de Lisboa	InstalPlus - Sistemas de Comunicação Informática, Lda	+
Aquisição aparelhos Ar Condicionado	6.200,00 €	17-12-2018	Guarda Nacional Republicana	Rolfrio - Sociedade Comercial de Equipamentos Hoteleiros, Lda	+
Requalificação do Largo Antunes Lima	388.279,16 €	17-12-2018	Município de Vila Verde	Bruno Barbosa Araújo Unipessoal Lda.	+
Aquisição de serviços de assessoria especializada no domínio das TIC...	25.179,00 €	17-12-2018	Município de Vila Verde	Ambiglobal - Prestação de Serviços de Segurança, Higiene e Saúde no Trabalho Lda.	+
AQUISIÇÃO DE EQUIPAMENTO PARA O TEATRO MUNICIPAL DE MATOSINHOS CONSTANTINO...	7.480,50 €	17-12-2018	Município de Matosinhos	JOSE PEDRO FERREIRA DOS SANTOS	+
O objeto deste contrato consiste na elaboração de estudos para...	43.000,00 €	17-12-2018	Município de Vila Verde	Whiteenergy.com - Soluções Informáticas Lda.	+
Aquisição de serviços de trabalhos de limpeza de faixas de...	14.630,00 €	17-12-2018	Município de Torres Vedras	João Carlos Beirante Santos Martinho	+

Web scraping

Friendly URLs

```
http://www.base.gov.pt/Base/pt/ResultadosPesquisa?
type=contratos&query=texto%3D%26tipo%3D0%26tipocontrato%
3D0%26cpv%3D%26numeroanuncio%3D%26aqinfo%3D%26adjudicant
e%3D%26adjudicataria%3D%26desdeprecocontrato_false%3D%26
desdeprecocontrato%3D%26ateprecocontrato_false%3D%26atep
recocontrato%3D%26desdedatacontrato%3D%26atedatacontrato
%3D%26desdedatapublicacao%3D%26atedatapublicacao%3D%26de
sdeprazoexecucao%3D%26ateprazoexecucao%3D%26desdedatafec
ho%3D%26atedatafecho%3D%26desdeprecoefectivo_false%3D%26
desdeprecoefectivo%3D%26ateprecoefectivo_false%3D%26atep
recoefectivo%3D%26pais%3D0%26distrito%3D0%26concelho%3D0
```

Web scraping

Selenium

SeleniumHQ Browser Automation

edit this page search selenium: Go

Projects | Download | Documentation | Support | About

What is Selenium?

Selenium automates browsers. That's it! What you do with that power is entirely up to you. Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) be automated as well.

Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser. It is also the core technology in countless other browser automation tools, APIs and frameworks.

Which part of Selenium is appropriate for me?

Selenium WebDriver

If you want to

- create robust, browser-based regression automation suites and tests
- scale and distribute scripts across many environments

Selenium IDE

If you want to

- create quick bug reproduction scripts
- create scripts to aid in automation-aided

Selenium is a suite of tools to automate web browsers across many platforms.

Selenium...

- runs in many browsers and operating systems
- can be controlled by many programming languages and testing frameworks.

Download Selenium

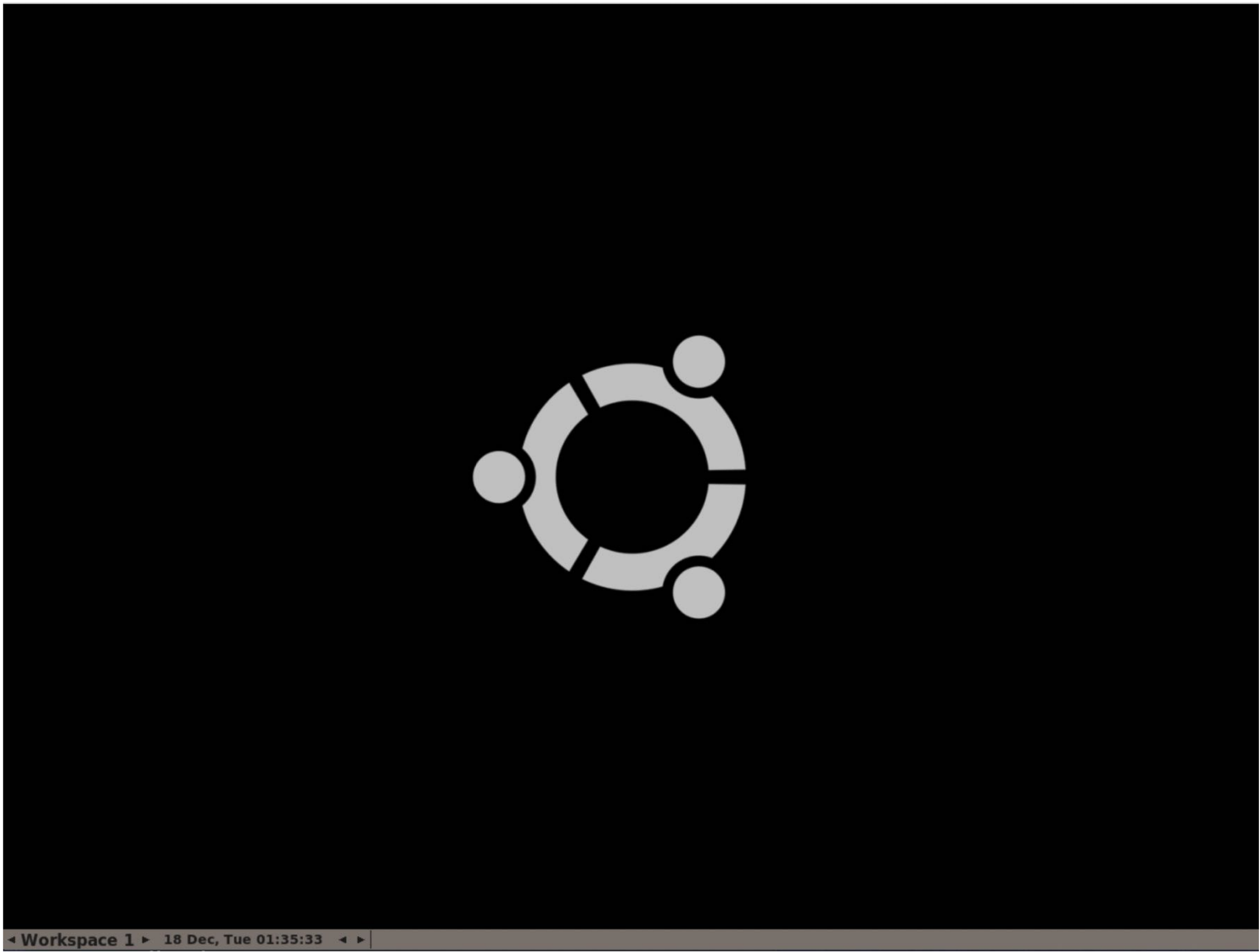
https://www.seleniumhq.org

Web scraping

Selenium

http://www.seg-social.pt/lista-de-devedores-na-seguranca-social

Web scraping



Selenium

Data collection

Web data sources

Using APIs

Using APIs

Application **P**rogramming **I**nterface (**API**)

an API is the part of a server that
handles requests and responses

Using APIs

Application **P**rogramming **I**nterface (**API**)

easy to use

allow automatic update of data sets

Using APIs

An example

Load the libraries:

```
library(jsonlite)
library(httr)
library(purrr)
library(dplyr)
library(tidyr)
```

Using APIs

An example

Load the first page:

```
url <- paste0("api.worldbank.org/v2/indicators?
format=json&page=", 1)
res <- GET(url)
parsed <- jsonlite::fromJSON(content(res, "text"))
```

Using APIs

An example


```
[
  {
    "page": 1,
    "pages": 313,
    "per_page": "50",
    "total": 15650
  },
  [
    {
      "id": "2.0.cov.Math.pl_3.prv",
      "name": "Coverage: Mathematics Proficiency Level 3, Private schools",
      "unit": "",
      "source": {
        "id": "37",
        "value": "LAC Equity Lab"
      },
      "sourceNote": "The coverage rate is the childhood access rate of a given opportunity
used in calculating the Human Opportunities Index (HOI). The coverage rate does not take
into account inequality of access between different circumstance groups.",
      "sourceOrganization": "LAC Equity Lab tabulations using PISA Data.",
      "topics": [
        {
          "id": "11",
          "value": "Poverty "
        }
      ]
    },
```

Using APIs

An example

A function to load the page, get the data in text format and convert the json into a data frame:

```
get_data <- function(page_n){
  url <- paste0("api.worldbank.org/v2/indicators?
format=json&page=",
                page_n)
  res <- GET(url)
  parsed <- jsonlite::fromJSON(content(res, "text"),
                               flatten = TRUE)
  Sys.sleep(1)
  return(as_data_frame(parsed[[2]]))
}
```

Using APIs

An example

Download the data for all pages:

```
get_data <- safely(get_data)
pages <- data_frame(page_n = 1:parsed[[1]]$pages)
indicators <- pages %>%
  mutate(data = map(page_n, get_data))
```

Using APIs

An example

Check for errors:

```
indicators %>%
  mutate(error = map(data, "error")) %>%
  mutate(error = map_int(error, ~ length(.x))) %>%
  dplyr::filter(error > 0)
```

Using APIs

An example

Get a data frame with all the indicators

```
indicators_clean <- indicators %>%
  mutate(result = map(data, "result")) %>%
  select(result) %>%
  unnest()
```

Using APIs

An example

RStudio

Go to file/function | Addins | Project: (None)

api_wb.Rmd* | indicators_clean

Filter

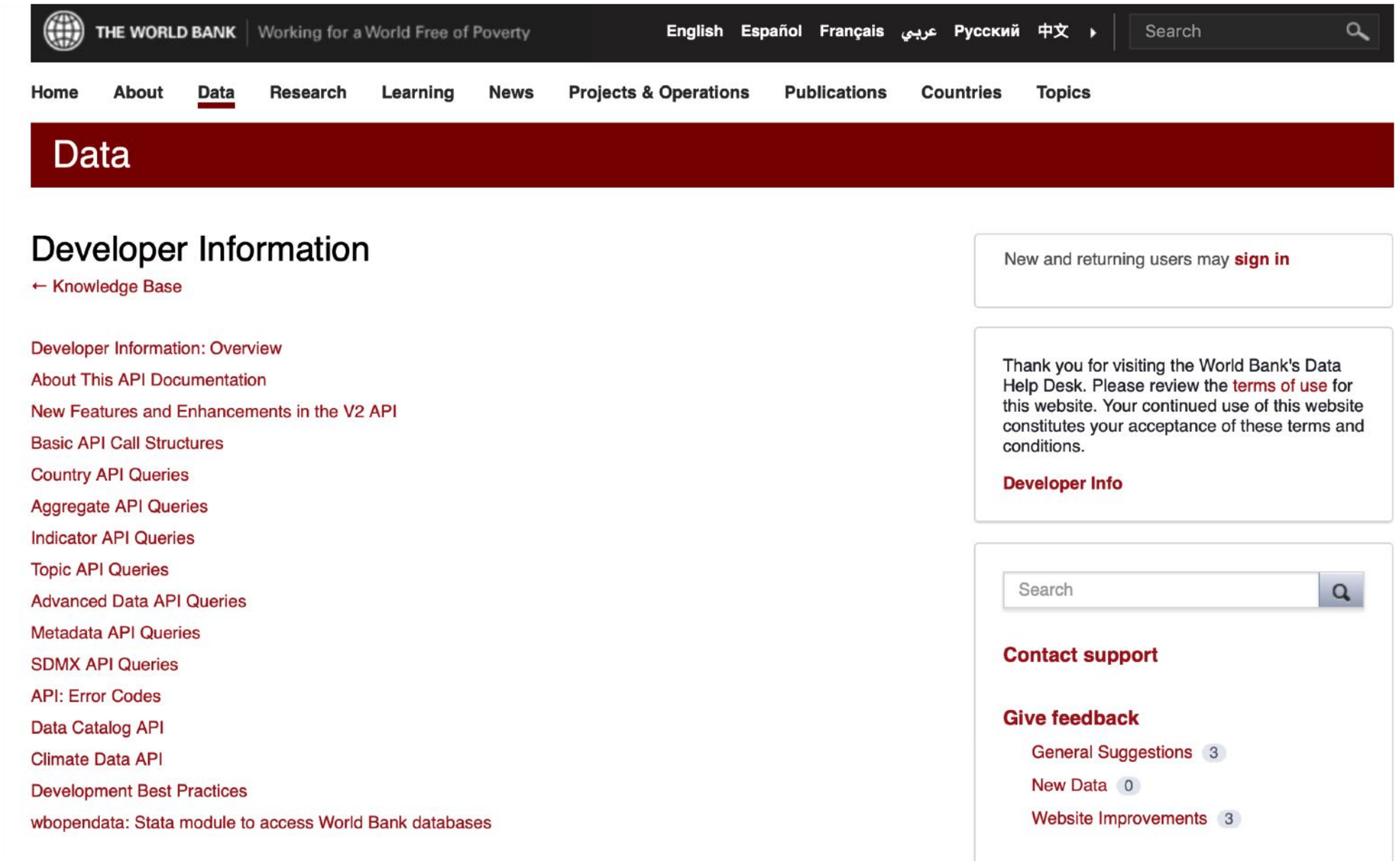
	id	name	unit	sourceNote	sourceOrganization	topics
1	2.0.cov.Math.pl_3.prv	Coverage: Mathematics Proficiency Level 3, Private sc...		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations using PISA Data.	*2 variables*
2	2.0.cov.Math.pl_3.all	Coverage: Mathematics Proficiency Level 3		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations using PISA Data.	*2 variables*
3	2.0.cov.Math.pl_2.pub	Coverage: Mathematics Proficiency Level 2, Public sch...		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations using PISA Data.	*2 variables*
4	2.0.cov.Math.pl_2.prv	Coverage: Mathematics Proficiency Level 2, Private sc...		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations using PISA Data.	*2 variables*
5	2.0.cov.Math.pl_2.all	Coverage: Mathematics Proficiency Level 2		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations using PISA Data.	*2 variables*
6	2.0.cov.Int	Coverage: Internet		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
7	2.0.cov.FPS	Coverage: Finished Primary School		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
8	2.0.cov.Ele	Coverage: Electricity		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
9	2.0.cov.Cel	Coverage: Mobile Phone		The coverage rate is the childhood access rate of a gi...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
10	1.3_ACCESS.ELECTRICITY.URBAN	Access to electricity (% of urban population)		Access to electricity is the percentage of total populat...	World Bank Global Electrification Database 2014	*list()*
11	1.2_ACCESS.ELECTRICITY.RURAL	Access to electricity (% of rural population)		Access to electricity is the percentage of rural populat...	World Bank Global Electrification Database 2013	*list()*
12	1.2.PSev.Poor4uds	Poverty Severity ($4 a day)-Urban		The poverty severity index combines information on ...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
13	1.2.PSev.2.5usd	Poverty Severity ($2.50 a day)-Urban		The poverty severity index combines information on ...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
14	1.2.PSev.1.90usd	Poverty Severity ($1.90 a day)-Urban		The poverty severity index combines information on ...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
15	1.2.PGap.Poor4uds	Poverty Gap ($4 a day)-Urban		The poverty gap captures the mean aggregate income...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
16	1.2.PGap.2.5usd	Poverty Gap ($2.50 a day)-Urban		The poverty gap captures the mean aggregate income...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
17	1.2.PGap.1.90usd	Poverty Gap ($1.90 a day)-Urban		The poverty gap captures the mean aggregate income...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
18	1.2.HCount.Vul4to10	Vulnerable ($4-10 a day) Headcount-Urban		The poverty headcount index measures the proportio...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
19	1.2.HCount.Poor4uds	Poverty Headcount ($4 a day)-Urban		The poverty headcount index measures the proportio...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
20	1.2.HCount.Ofcl	Official Moderate Poverty Rate-Urban		The poverty headcount index measures the proportio...	LAC Equity Lab tabulations of data from National Stati...	*2 variables*
21	1.2.HCount.Mid10to50	Middle Class ($10-50 a day) Headcount-Urban		The poverty headcount index measures the proportio...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
22	1.2.HCount.2.5usd	Poverty Headcount ($2.50 a day)-Urban		The poverty headcount index measures the proportio...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
23	1.2.HCount.1.90usd	Poverty Headcount ($1.90 a day)-Urban		The poverty headcount index measures the proportio...	LAC Equity Lab tabulations of SEDLAC (CEDLAS and th...	*2 variables*
24	1.1_YOUTH.LITERACY.RATE	Literacy rate, youth total (% of people ages 15-24)		The number of persons aged 15 to 24 years who can ...	Source of information: UNESCO Institute for Statistics ...	*list()*
25	1.1_TOTAL.FINAL.ENERGY.CONSUM	Total final energy consumption (TFEC)		Total final energy consumption (TFEC): This indicator ...	World Bank and International Energy Agency (IEA Stati...	*list()*

Showing 1 to 27 of 15,650 entries, 8 total columns

Environment | History | Connections

Using APIs

An example



https://datahelpdesk.worldbank.org/knowledgebase/topics/125589-developer-information

Data collection

Data cleaning and merging datasets

Data cleaning

Regular expressions

Regular expressions are sequences that define patterns of characters that can be used for search and replace

Data cleaning

Regular expressions

remove extra spaces

remove stop words

remove prepositions in names

remove accents

remove punctuation

Usage examples

Data cleaning

Regular expressions

A function to trim whitespaces:

```
trim <- function (x){
  gsub("^\\s+|\\s+$", "", x) %>%
    gsub("[ ]+", " ", .)
}

trim("    Alice     Maravilha   ")
```

```
[1] "Alice Maravilha"
```

Data cleaning

Regular expressions

A function to extract the first and last name:

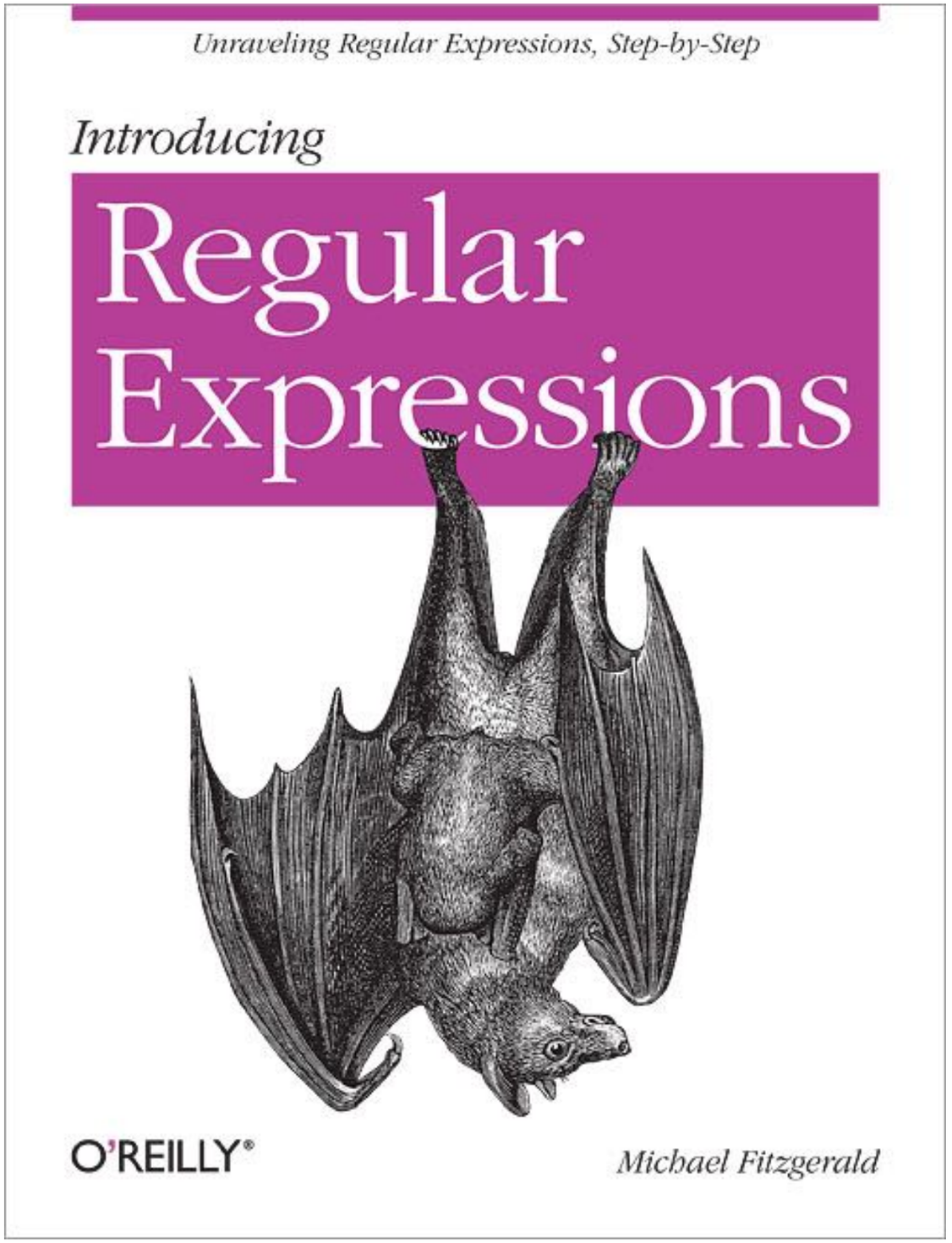
```
first_last <- function (x){
  gsub("^([A-z]+) .* ([A-z]+)$",
       "\\1 \\2", x)
}

first_last("Joaquim Adalberto do Porto e de Lisboa")
```

```
[1] "Joaquim Lisboa"
```

Data cleaning

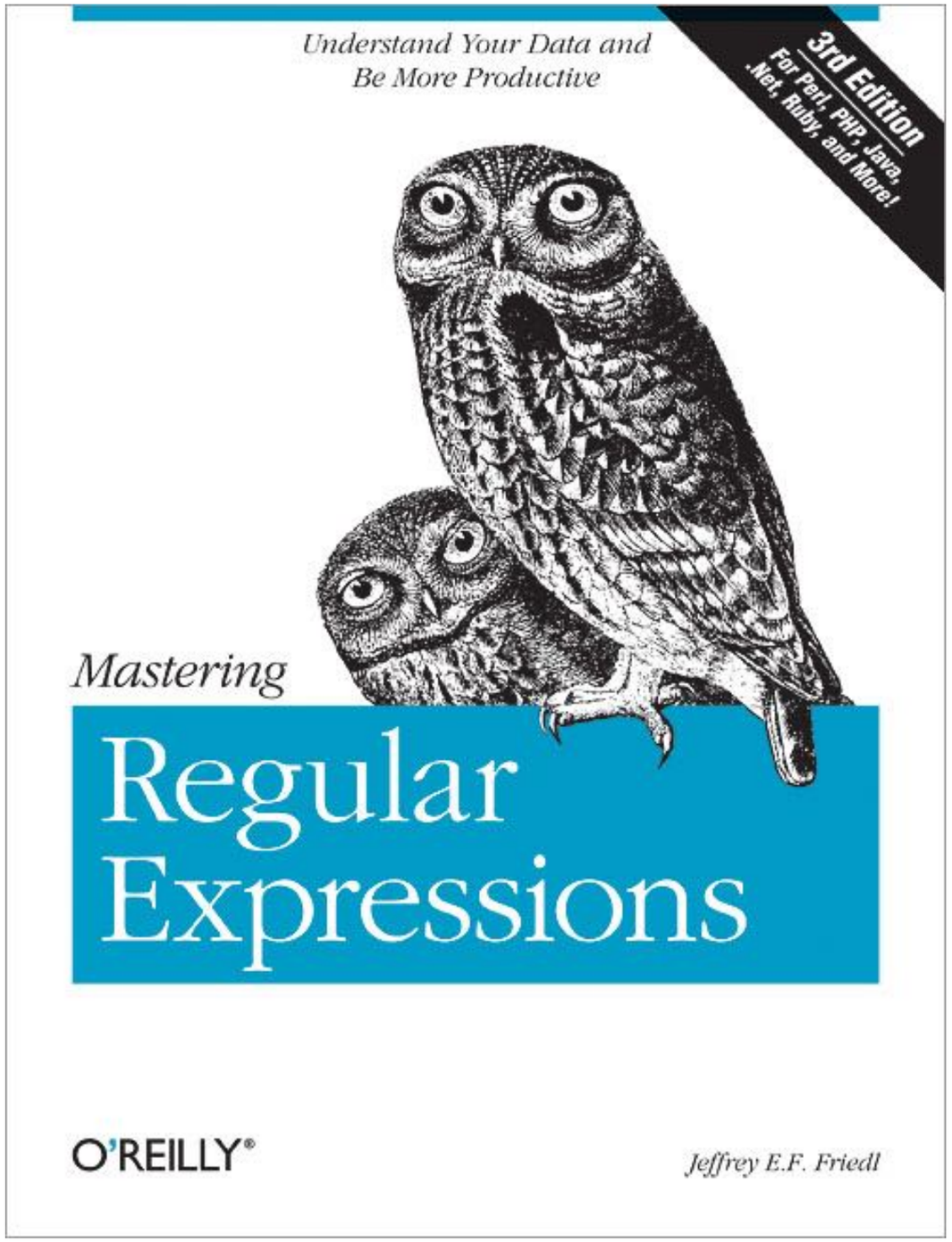
Regular expressions



http://shop.oreilly.com/product/0636920012337.do

Data cleaning

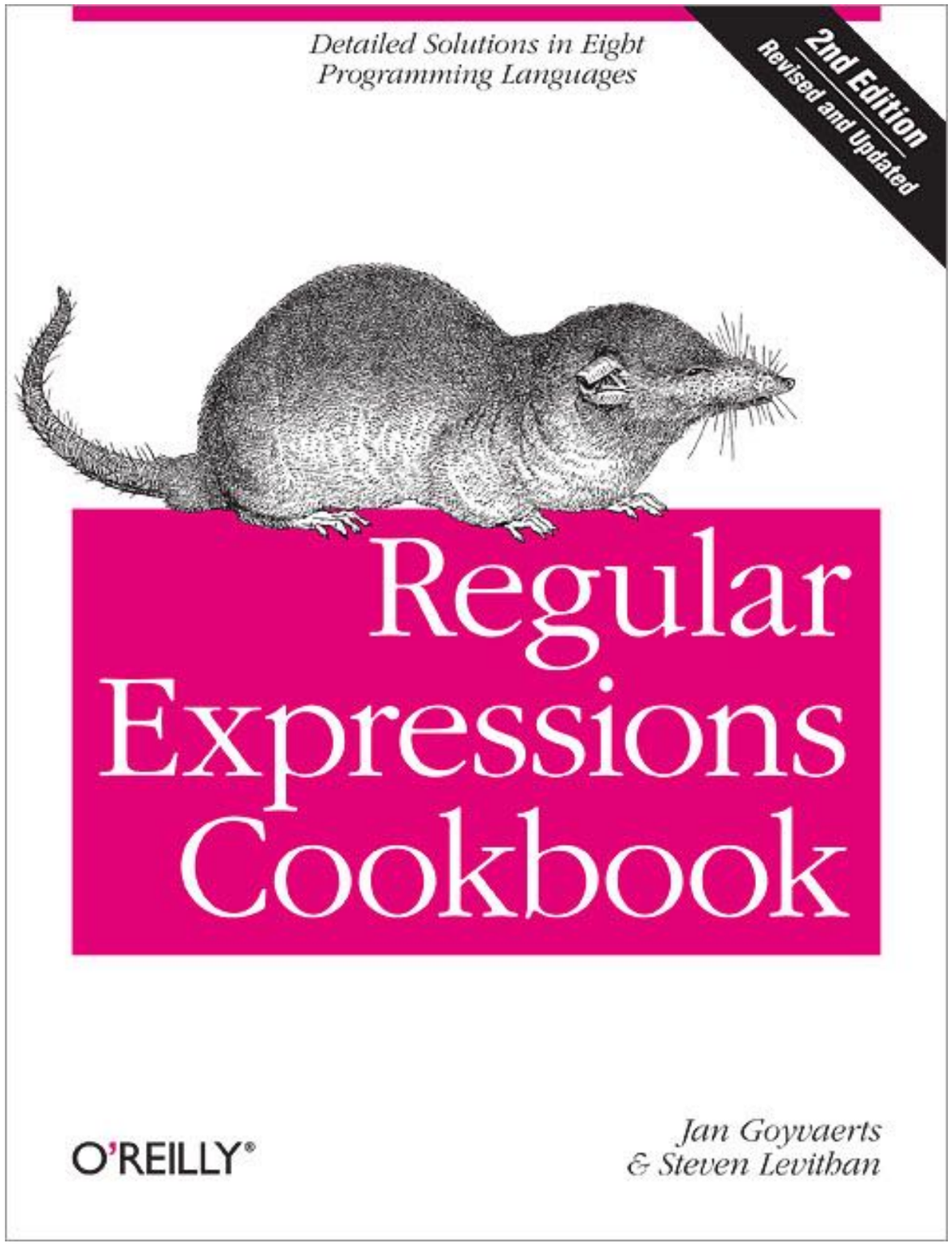
Regular expressions



http://shop.oreilly.com/product/9780596528126.do

Data cleaning

Regular expressions



http://shop.oreilly.com/product/0636920023630.do

Record disambiguation

Openrefine

“A free, open source, powerful tool for working with messy data”

http://openrefine.org

Record disambiguation

Openrefine

demonstration

Record disambiguation

Other tools

RecordLinkage: Record Linkage in R

https://cran.r-project.org/web/packages/RecordLinkage/index.html

fastLink: Fast Probabilistic Record Linkage with Missing Data

https://cran.r-project.org/web/packages/fastLink/index.html

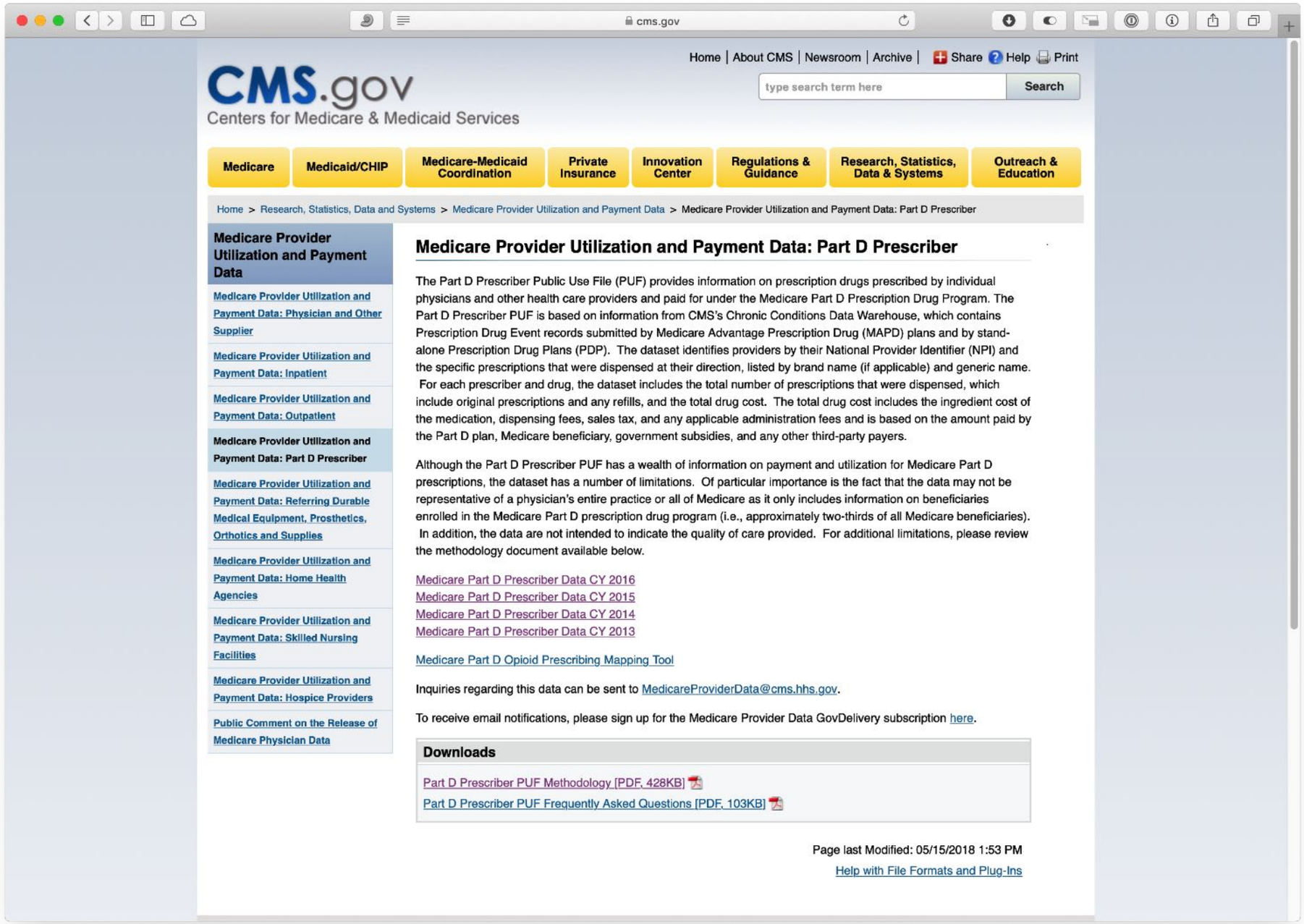
Working with large data sets

Working with large data sets

Biggish data

Biggish data

Example



2013 to 2016

https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html

Biggish data

Tools

All examples use **dplyr** and **dbplyr**

allows using the database as an in-memory data frame

Biggish data

No changes of default configuration!

better results are possible on the same hardware

Working with large data sets

Biggish data

Relational databases

sqlite

“SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.”

https://www.sqlite.org

sqlite

Advantages

self-contained file

no need to install additional software

supports databases up to 140 TB

files can be read in different languages

sqlite

Disadvantages

no user management

not adequate for concurrent writes to the database

not fast enough for interactive analysis

not optimized for data processing

sqlite

Example

```
# install.packages(c("dbplyr", "RSQLite"))
library(dplyr)
library(dbplyr)
library(readr)
library(DBI)
library(purrr)
```

sqlite

Example

Create the connection to the database:

```
dbcon <- dbConnect(RSQLite::SQLite(),
                   "../data/prescriptions.sqlite")
```

sqlite

Example

```
files <- data_frame(year = 13:16)

read_and_load_csv <- function(year, dbcon){
  prescriber_data <- read_tsv(
    file = paste0(
            "../data/PartD_Prescriber_PUF_NPI_DRUG_", year, "/
PartD_Prescriber_PUF_NPI_DRUG_", year, ".txt"))

  prescriber_data <- prescriber_data %>%
    mutate(year = year)

  dbWriteTable(dbcon,
               "prescriptions",
               prescriber_data,
               append = TRUE)

}

walk(files$year, read_and_load_csv,
     dbcon)
```

653.18s about 10 min

sqlite

Example

there are fasters ways to load the data…

sqlite

Example

```
dbListTables(dbcon)
```

```
## [1] "prescriptions"
```

```
dbListFields(dbcon, "prescriptions")
```

```
 [1] "npi"
 [2] "nppes_provider_last_org_name"
 [3] "nppes_provider_first_name"
 [4] "nppes_provider_city"
 [5] "nppes_provider_state"
 [6] "specialty_description"
 [7] "description_flag"
 [8] "drug_name"
 [9] "generic_name"
[10] "bene_count"
[11] "total_claim_count"
[12] "total_30_day_fill_count"
[13] "total_day_supply"
[14] "total_drug_cost"
[15] "bene_count_ge65"
[16] "bene_count_ge65_suppress_flag"
[17] "total_claim_count_ge65"
[18] "ge65_suppress_flag"
[19] "total_30_day_fill_count_ge65"
[20] "total_day_supply_ge65"
[21] "total_drug_cost_ge65"
[22] "year"
```

sqlite

Example

Create a reference to the table:

```
prescriptions <- tbl(dbcon,
                     "prescriptions")
```

it can now be used as a data frame!

sqlite

Example

Count the number of rows in the table:

```
prescriptions %>% tally()
```

```
97 255 685
```

takes 43s

sqlite

Example

Count the number of distinct prescribers per state and year:

```
qry <- prescriptions %>%
    group_by(nppes_provider_state, year) %>%
    summarise(n = n_distinct(npi)) %>%
    arrange(desc(n))
```

```
## # A tibble: 244 x 3
## # Groups: nppes_provider_state [61]
## nppes_provider_state year n
## <chr> <int> <int>
## 1 CA 16 91469
## 2 CA 15 89246
## 3 CA 14 86586
## 4 CA 13 83802
## 5 NY 16 67635
## 6 NY 15 65499
...
```

takes 3 min

sqlite

Example

Creating indices can help:

```
dbExecute(dbcon,
'CREATE INDEX nppes_provider_state_year ON
    prescriptions (nppes_provider_state, year);')
```

…but it will increase the database size!

takes 149s

sqlite

Example

```
qry <- prescriptions %>%
 group_by(nppes_provider_state, year) %>%
summarise(n = n_distinct(npi)) %>%
arrange(desc(n)) %>%
collect()
```

takes about 7 min!

sqlite

not fast enough for run-time analysis!

sqlite

can still be useful

work with a subset of the data in memory
and run the code in batch mode for the complete dataset

PostgreSQL

“PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.”

https://www.postgresql.org

PostgreSQL

Advantages

feature rich

active development

can be replicated across a cluster of computers
and use load balancing

good geographical information system (GIS)
support

many queries run in parallel

PostgreSQL

Disadvantages

Not optimized for data processing

Requires installing a server

although it can run R functions natively using the PL/R
(R Procedural Language for PostgreSQL)

https://github.com/postgres-plr/plr

PostgreSQL

Example

```
dbcon <- dbConnect(RPostgreSQL::PostgreSQL(),
                   host = "localhost",
                   dbname = "prescriptions",
                   user = "cms",
                   password = "secret")
```

PostgreSQL

Example

Loading the data into the database

took about **29 min**

the loading procedure is exactly the same
as for the sqlite database

PostgreSQL

Example

Count the number of rows in the table:

```
prescriptions %>% tally()
```

```
97 255 685
```

takes 2.6 min

PostgreSQL

Example

Count the number of distinct prescribers per state and year:

```
qry <- prescriptions %>%
    group_by(nppes_provider_state, year) %>%
    summarise(n = n_distinct(npi)) %>%
    arrange(desc(n))
```

```
## # A tibble: 244 x 3
## # Groups: nppes_provider_state [61]
## nppes_provider_state year n
## <chr> <int> <int>
## 1 CA 16 91469
## 2 CA 15 89246
## 3 CA 14 86586
## 4 CA 13 83802
## 5 NY 16 67635
## 6 NY 15 65499
...
```

takes 7.8 min

PostgreSQL

Example

Creating indices can help:

```
dbExecute(dbcon,
'CREATE INDEX nppes_provider_state_year ON
    prescriptions (nppes_provider_state, year);')
```

…but it will increase the database size!

takes 331.5s

PostgreSQL

Example

```
qry <- prescriptions %>%
 group_by(nppes_provider_state, year) %>%
summarise(n = n_distinct(npi)) %>%
arrange(desc(n)) %>%
collect()
```

takes about 7.8 min

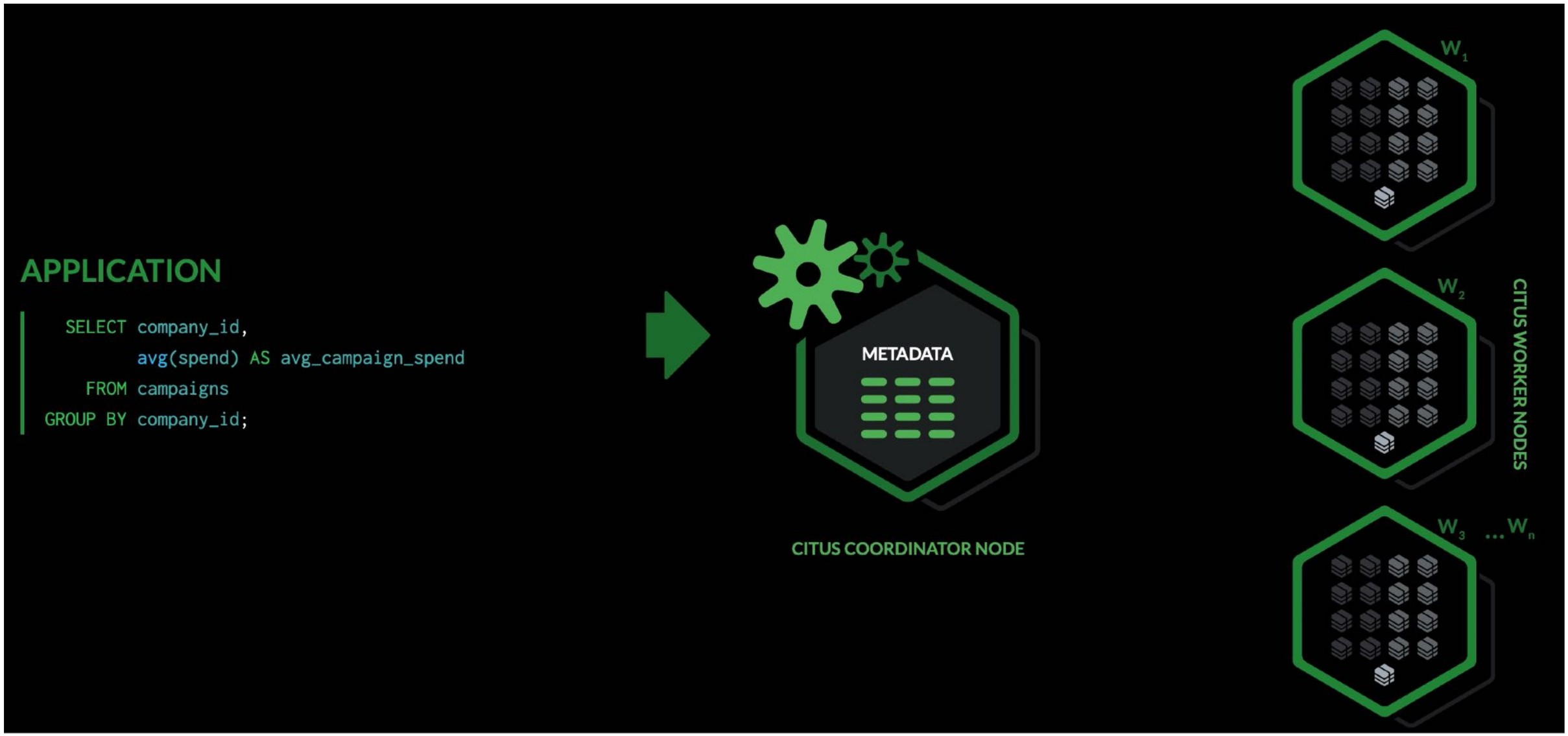
PostgreSQL

not fast enough for run-time analysis!

PostgreSQL

Parallelize queries across nodes in a cluster

Citus plugin



https://www.citusdata.com

Working with large data sets

Biggish data

Column store databases

Column store databases

These databases store data tables **by column** rather than by row and, this usually allows **faster queries**

Column store databases

“MonetDB is designed for multi-core parallel execution on desktops to reduce response time for complex query processing.”

It is SQL compliant

https://www.monetdb.org

MonetDB

Example

The MonetDBLite is compatible with dbplyr

all queries remain the same

no need to install a database server

the database creates the indices automatically

MonetDB

Example

```
dbcon <- dbConnect(MonetDBLite::MonetDBLite(),
                   "../data/monet")
```

MonetDB

Example

Loading the data into the database

took about **12 min**

the loading procedure is exactly the same
as for the sqlite and PostgreSQL databases

MonetDB

Example

Count the number of rows in the table:

```
prescriptions %>% tally()
```

```
97 255 685
```

takes 0.15s

MonetDB

Example

Count the number of distinct prescribers per state and year:

```
qry <- prescriptions %>%
    group_by(nppes_provider_state, year) %>%
    summarise(n = n_distinct(npi)) %>%
    arrange(desc(n))
```

```
## # A tibble: 244 x 3
## # Groups: nppes_provider_state [61]
## nppes_provider_state year n
## <chr> <int> <int>
## 1 CA 16 91469
## 2 CA 15 89246
## 3 CA 14 86586
## 4 CA 13 83802
## 5 NY 16 67635
## 6 NY 15 65499
...
```

takes 5.37s

MonetDB

Example

Find the number of individual doctors per speciality in 2016, ordering the results in descending order:

```
qry <- prescriptions %>%
    dplyr::filter(year == 16,
        !is.na(nppes_provider_first_name)) %>%
  group_by(specialty_description) %>%
  summarise(n = n_distinct(npi)) %>%
  arrange(desc(n)) %>%
  collect()
```

takes 15.8s

MonetDB

Example

	specialty_description	n
1	Internal Medicine	108859
2	Nurse Practitioner	108035
3	Family Practice	99198
4	Dentist	98750
5	Physician Assistant	69747
6	Emergency Medicine	36287
7	Psychiatry	23075
8	Obstetrics & Gynecology	21914
9	Student in an Organized Health Care Education/Trainin	21906
10	Optometry	21258
11	Cardiovascular Disease (Cardiology)	19743
12	Orthopedic Surgery	19325
13	Ophthalmology	18395
14	General Surgery	16920

MonetDB

fast enough to deal with 97m records
out-of-memory on a laptop

Column store databases

Alternatives

druid

Technology Use Cases Powered By Docs Community Download

Apache Druid (incubating) is a high performance analytics data store for event-driven data.

Download GitHub

Overview

Analyze event streams
Druid provides fast analytical queries, at high concurrency, on both real-time and historical data. Druid is often used to power interactive UIs.

Utilize reimagined architecture
Druid is a new class of data store that combines ideas from OLAP/analytic databases, timeseries databases, and search systems to enable new use cases with streaming and batch data.

Build next generation data stacks
Druid integrates natively with message buses (Kafka, AWS Kinesis, etc) and data lakes (HDFS, AWS S3, etc). Druid works especially well as a query layer for streaming data architectures.

Unlock new workflows
Druid is built for rapid, ad-hoc analytics on both real-time and historical data. Explain trends, explore data, and quickly iterate on queries to answer questions.

Deploy anywhere
Druid can be deployed in any *NIX environment on commodity hardware, both in the cloud and on premise. Druid is cloud-native: scaling up and down is as simple as adding and removing processes.

Latest releases

Druid 0.12.3 Released
Sep 18 2018

Druid 0.12.2 Released
Aug 9 2018

Druid 0.12.1 Released
Jun 8 2018

Druid 0.12.0 Released
Mar 8 2018

Upcoming Events

DEC 6 Big Data and Cloud Meetup
Demystifying AWS Analytics and Building Streaming Analytics using Kafka & Druid

Join a Druid Meetup!

Featured Content

How Druid enables analytics at Airbnb

http://druid.io

Working with large data sets

Biggish data

Document databases

Document databases

“MongoDB is a document database (…) stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use”

https://www.mongodb.com/what-is-mongodb

Document databases

JSON - Javascript Object Notation

https://en.wikipedia.org/wiki/JSON


```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Document databases

Use cases

Storing data from APIs with no clear schema definition

Document databases

It is also possible to store JSON documents
in a PostgreSQL database

Working with large data sets

Bigger data

Spark

“Apache Spark™ is a unified analytics engine for large-scale data processing.”

https://spark.apache.org

Spark

Apache Spark SQL

Download | Libraries | Documentation | Examples | Community | Developers

Spark SQL is Apache Spark's module for working with structured data.

Integrated

Seamlessly mix SQL queries with Spark programs.

Spark SQL lets you query structured data inside Spark programs, using either SQL or a familiar DataFrame API. Usable in Java, Scala, Python and R.

```
results = spark.sql(
  "SELECT * FROM people")
names = results.map(lambda p: p.name)
```

Apply functions to results of SQL queries.

Uniform Data Access

Connect to any data source the same way.

DataFrames and SQL provide a common way to access a variety of data sources, including Hive, Avro, Parquet, ORC, JSON, and JDBC. You can even join data across these sources.

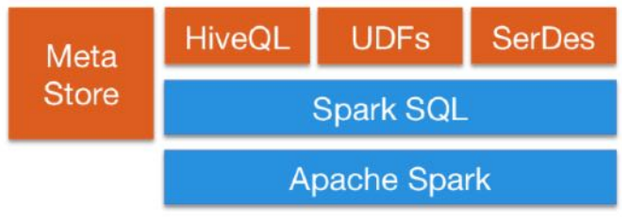
```
spark.read.json("s3n://...")
  .registerTempTable("json")
results = spark.sql(
  """SELECT *
     FROM people
     JOIN json ...""")
```

Query and join different data sources.

Hive Integration

Run SQL or HiveQL queries on existing warehouses.

Spark SQL supports the HiveQL syntax as well as Hive SerDes and UDFs, allowing you to access existing Hive warehouses.



Spark SQL can use existing Hive metastores, SerDes, and UDFs.

Spark

Can run on a **local machine**,
on a local cluster or
on a **cluster in the cloud**

Spark

R has an interface to **Spark** that allows using **dplyr**

not much changes!

Spark

Example

Load the required packages:

```
library(sparklyr)
library(dplyr)
library(readr)
library(purrr)
```

you'll need to install JDK
and spark locally

Spark

Example

Tune the Spark configuration and open the connection to Spark:

```
Sys.setenv(JAVA_HOME = "/Library/Java/
JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/")
config <- spark_config()
config$`sparklyr.shell.driver-java-options` <-
paste0("-Djava.io.tmpdir=", "../data/spark/")
config$`sparklyr.shell.driver-memory` <- "4G"
config$`sparklyr.shell.executor-memory` <- "4G"
config$`spark.yarn.executor.memoryOverhead` <- "512"

dbcon <- spark_connect(master = "local",
                       config = config)
```

Spark

Example

Function to read the csv files and copy them to Spark

```
files <- data_frame(year = 13:16)

read_and_load_csv <- function(year, dbcon){

  aux <- spark_read_csv(
      dbcon,
      name = "aux",
      path = paste0(
        "../data/PartD_Prescriber_PUF_NPI_DRUG_",
        year, "/PartD_Prescriber_PUF_NPI_DRUG_",
        year, ".txt"),
      delimiter = "\t",
      overwrite = TRUE,
      memory = FALSE)

  if ("prescriber" %in% src_tbls(dbcon)){
    prescriber_data <- tbl(dbcon, "prescriber")
    aux %>%
      mutate(year = year) %>%
      sdf_bind_rows(prescriber_data) %>%
      sdf_register("prescriber")
  } else {
    aux %>%
      mutate(year = year) %>%
      sdf_register("prescriber")
  }
}
```

Spark

Example

Read each csv file and copy them to Spark:

```
walk(files$year, read_and_load_csv,
dbcon)
```

```
   user  system elapsed
  4.926   0.390 160.748
```

Spark

Example

Count the number of distinct prescribers per state and year:

```
qry <- prescriptions %>%
group_by(nppes_provider_state, year) %>%
summarise(n = n_distinct(npi)) %>%
arrange(desc(n))
```

```
   user  system elapsed
  1.464   0.265 115.086
```

Spark

Example

Find the number of individual doctors per speciality in 2016, ordering the results in descending order:

```
qry <- prescriptions %>%
    dplyr::filter(year == 16,
        !is.na(nppes_provider_first_name)) %>%
  group_by(specialty_description) %>%
  summarise(n = n_distinct(npi)) %>%
  arrange(desc(n)) %>%
  collect()
```

```
   user  system elapsed
  0.732   0.118  56.614
```

Spark

Example

Deploy a virtual machine in the cloud to run Spark

Spark

Example

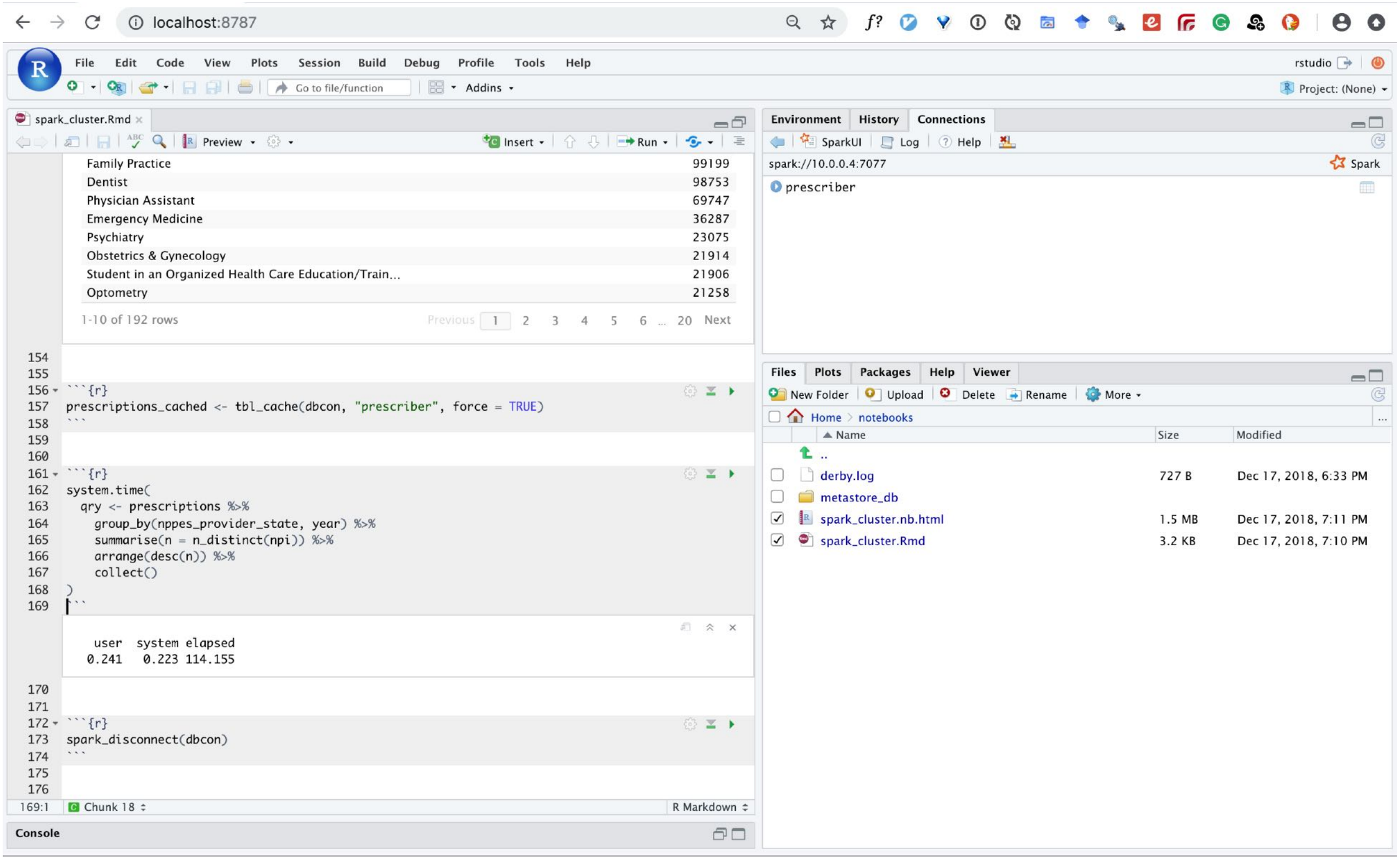
Azure Distributed Data Engineering Toolkit

"(…) is a python CLI application for provisioning on-demand Spark on Docker clusters in Azure. It's a cheap and easy way to get up and running with a Spark cluster, and a great tool for Spark users who want to experiment and start testing at scale."

https://github.com/Azure/aztk

Spark

Example



After provisioning the virtual machine, everything is the same

Spark

Example

1 F8 VM

The F-series virtual machines support 2-GiB RAM and 16 GB of local solid state drive (SSD) per CPU core, and are optimized for compute intensive workloads. The F-series is based on the 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) processor, which can achieve clock speeds as high as 3.2 GHz with the Intel Turbo Boost Technology 2.0. These virtual machines are suitable for scenarios like batch processing, web servers, analytics, and gaming.

For persistent storage, use the variant Fs virtual machines and purchase Premium Storage separately. The pricing and billing meters for Fs sizes are the same as F-series.

ADD TO ESTIMATE	INSTANCE	CORE	RAM	TEMPORARY STORAGE	PAY AS YOU GO	ONE YEAR RESERVED (% SAVINGS)	THREE YEAR RESERVED (% SAVINGS)
⊕	F1	1	2.00 GiB	16 GiB	€0.049/hour	€0.035/hour (~28%)	€0.024/hour (~51%)
⊕	F2	2	4.00 GiB	32 GiB	€0.097/hour	€0.07/hour (~28%)	€0.048/hour (~51%)
⊕	F4	4	8.00 GiB	64 GiB	€0.192/hour	€0.141/hour (~27%)	€0.094/hour (~51%)
⊕	F8	8	16.00 GiB	128 GiB	€0.383/hour	€0.281/hour (~27%)	€0.188/hour (~51%)
⊕	F16	16	32.00 GiB	256 GiB	€0.767/hour	€0.561/hour (~27%)	€0.376/hour (~51%)

https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/

Spark

Example

Count the number of distinct prescribers per state and year:

```
qry <- prescriptions %>%
group_by(nppes_provider_state, year) %>%
summarise(n = n_distinct(npi)) %>%
arrange(desc(n))
```

```
   user  system elapsed
  0.112   0.060  39.163
```

Spark

Example

Find the number of individual doctors per speciality in 2016, ordering the results in descending order:

```
qry <- prescriptions %>%
    dplyr::filter(year == 16,
        !is.na(nppes_provider_first_name)) %>%
  group_by(specialty_description) %>%
  summarise(n = n_distinct(npi)) %>%
  arrange(desc(n)) %>%
  collect()
```

```
   user  system elapsed
  0.065   0.042  18.491
```

Spark

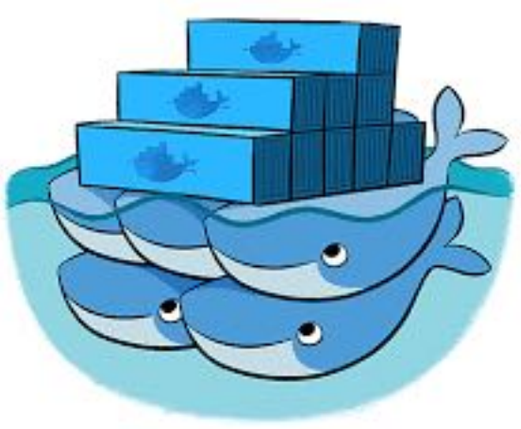
Example

it is equally easy to provision a cluster with Hadoop
these is where the speed gains occur

Spark

Example

kubernetes



docker swarm

Distributed computing

Distributed computing

Parallel computation

Parallel computation

Parallel computation

Example

Estimate a Normal-GARCH(1,1) model

for each of the S&P-500 constituents and

plot the conditional volatility,

over the period from

January, 1 2000 to December, 13 2018

Parallel computation

Example

Embarrassingly parallel

Parallel computation

Example

Get the tickers:

```
tickers <- readRDS(file =
                     "../data/sp_500_constituents.rds")
```

```
tickers <- tickers %>%
  select(Symbol, Security) %>%
  setNames(c("ticker", "co_name")) %>%
  mutate(ticker = ticker %>%
           gsub("\\.", "-", .))
```

Parallel computation

Example

Function to download the price data from Yahoo Finance:

```
get_data <- function(ticker){

  aux <- getSymbols(Symbols = ticker,
                    from = "2000-01-01",
                    to = "2018-12-13",
                    auto.assign = FALSE)
  aux <- Ad(aux)/lag(Ad(aux)) - 1
  aux <- na.omit(aux)
  return(aux)

}
```

Parallel computation

Example

Download the price data from Yahoo Finance for each stock:

```
returns <- tickers %>%
  mutate(ret = map(ticker, get_data))
```

Parallel computation

Example

Function to estimate the Normal-GARCH(1,1) model:

```
estimate_garch_model <- function(ret){

  spec <-  ugarchspec(variance.model =
                        list(model = "sGARCH",
                             garchOrder = c(1,1)),
                      mean.model =
                        list(armaOrder = c(0,0),
                             include.mean = TRUE),
                      distribution.model = "std")
  fit <-  ugarchfit(data = ret,
                    spec = spec)
}
```

Parallel computation

Example

Estimate the Normal-GARCH(1,1) model for each stock:

```
results <- foreach(i = 1:NROW(returns),
                   .packages = "rugarch")
%dopar%
  estimate_garch_model(returns$ret[[i]])
```

```
   user  system elapsed
332.186  11.141 355.618
```

Parallel computation

Example

Estimate the Normal-GARCH(1,1) model for each stock, but now in parallel (using the package doParallel):

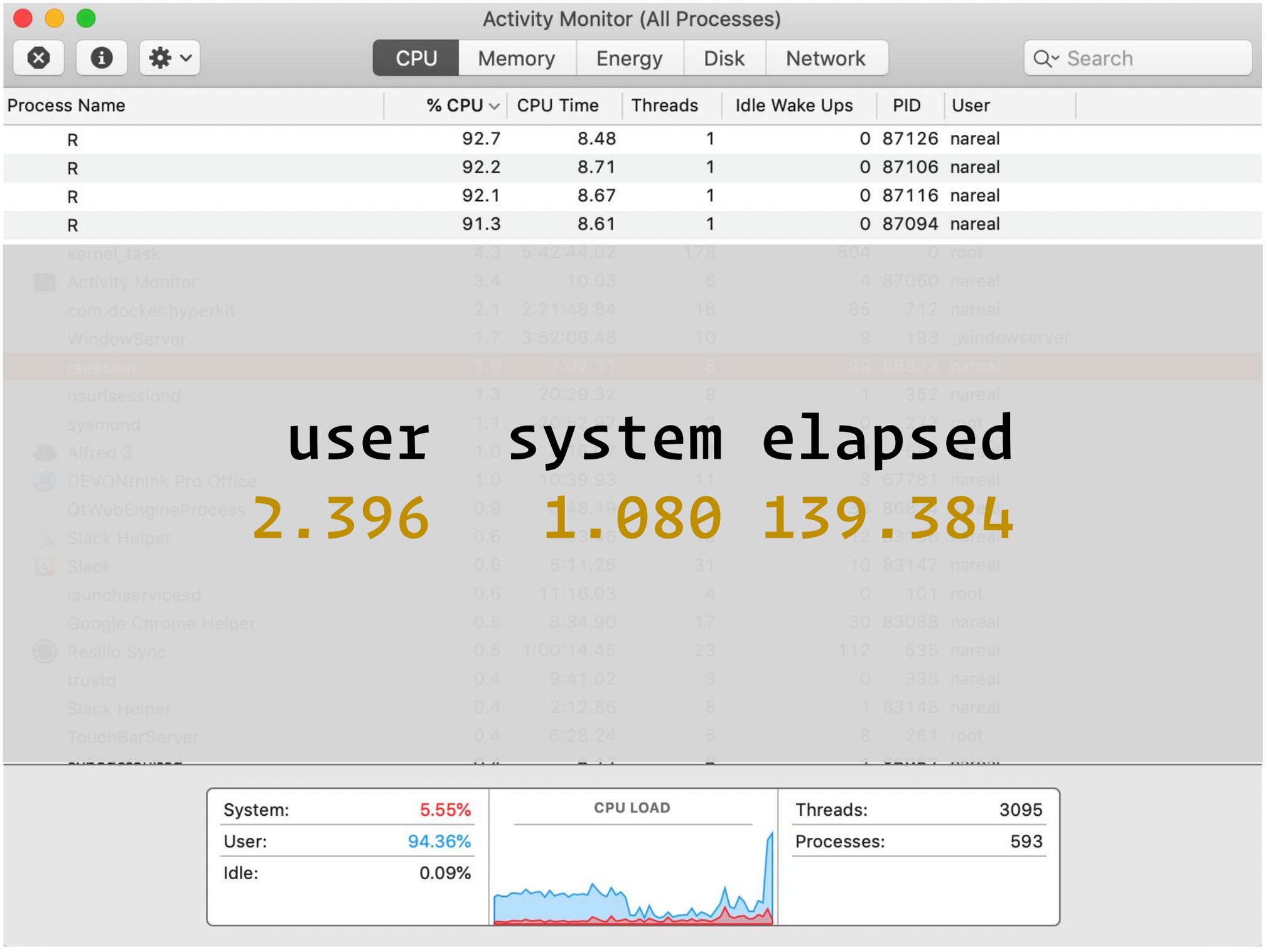
```
cl <- makeCluster(4)
registerDoParallel(cl)

results <- foreach(i = 1:NROW(returns),
                   .packages = "rugarch") %dopar%
    estimate_garch_model(returns$ret[[i]])

stopCluster(cl)
```

Parallel computation

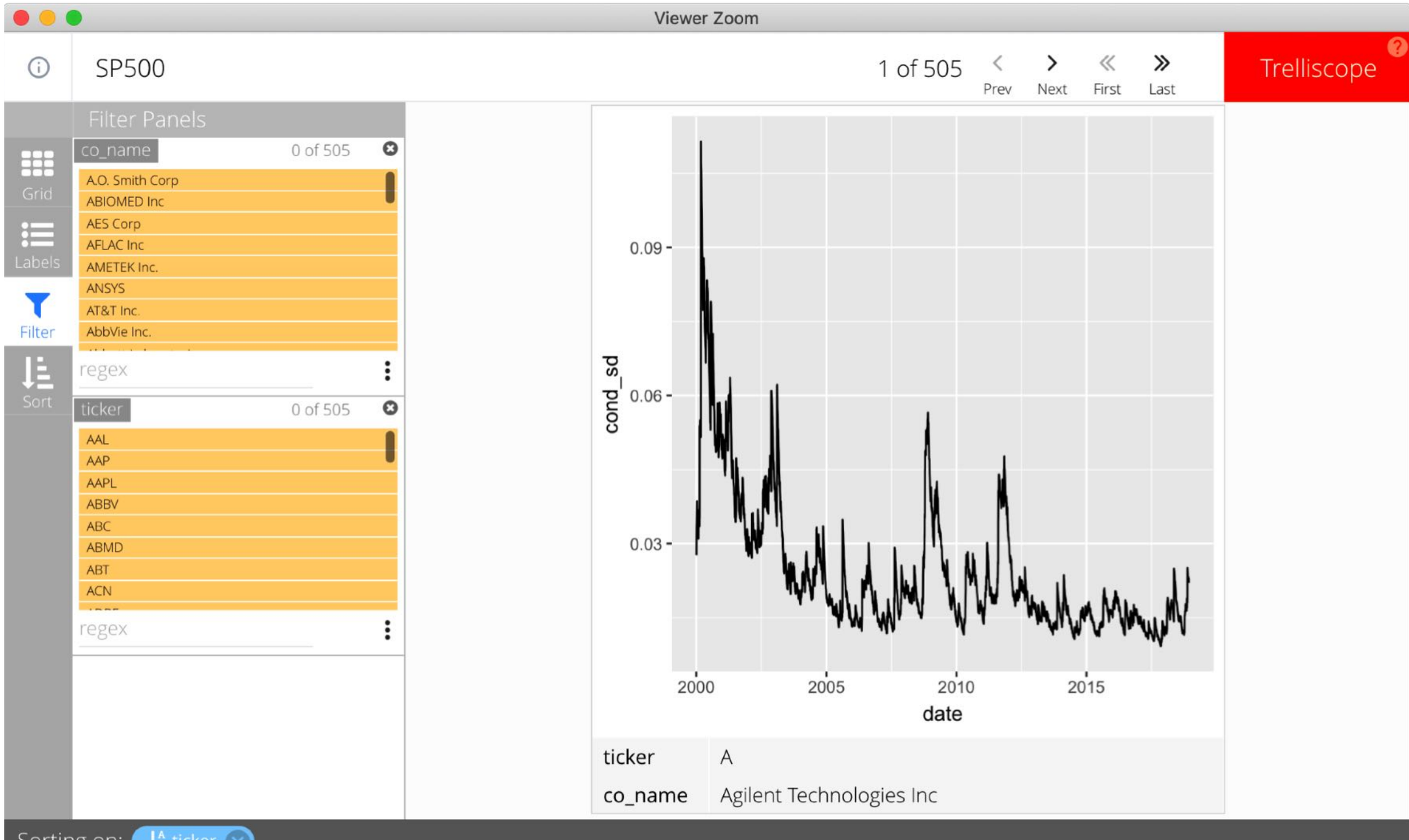
Example



Parallel computation

Example

Use trelliscopejs to display the results:



Distributed computing

Distributed computing

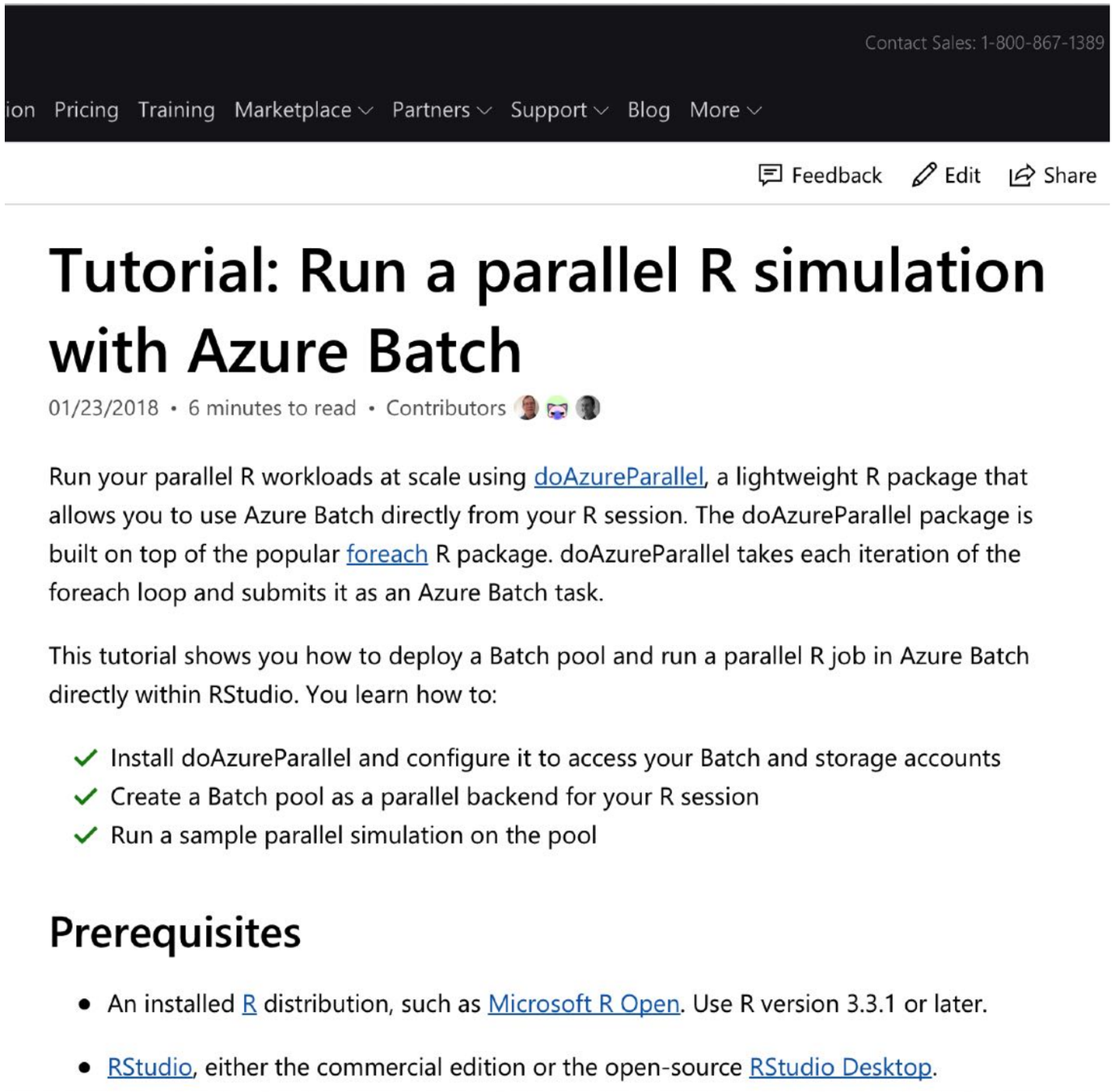
Distributed computation

To spread the computation over a cluster of computers is now trivial

just change the **doParallel** backend
for instance using snow (for a local cluster),
or **doAzureParallel** for a cloud cluster

Distributed computation

Example



Contact Sales: 1-800-867-1389

ion Pricing Training Marketplace Partners Support Blog More

Feedback Edit Share

Tutorial: Run a parallel R simulation with Azure Batch

01/23/2018 • 6 minutes to read • Contributors

Run your parallel R workloads at scale using doAzureParallel, a lightweight R package that allows you to use Azure Batch directly from your R session. The doAzureParallel package is built on top of the popular foreach R package. doAzureParallel takes each iteration of the foreach loop and submits it as an Azure Batch task.

This tutorial shows you how to deploy a Batch pool and run a parallel R job in Azure Batch directly within RStudio. You learn how to:

- ✓ Install doAzureParallel and configure it to access your Batch and storage accounts
- ✓ Create a Batch pool as a parallel backend for your R session
- ✓ Run a sample parallel simulation on the pool

Prerequisites

- An installed R distribution, such as Microsoft R Open. Use R version 3.3.1 or later.
- RStudio, either the commercial edition or the open-source RStudio Desktop.

https://docs.microsoft.com/en-us/azure/batch/tutorial-r-doazureparallel

Distributed computing

GPU

GPU

specialized processors, **not** suitable for general purpose computing

allows massive parallel execution of code

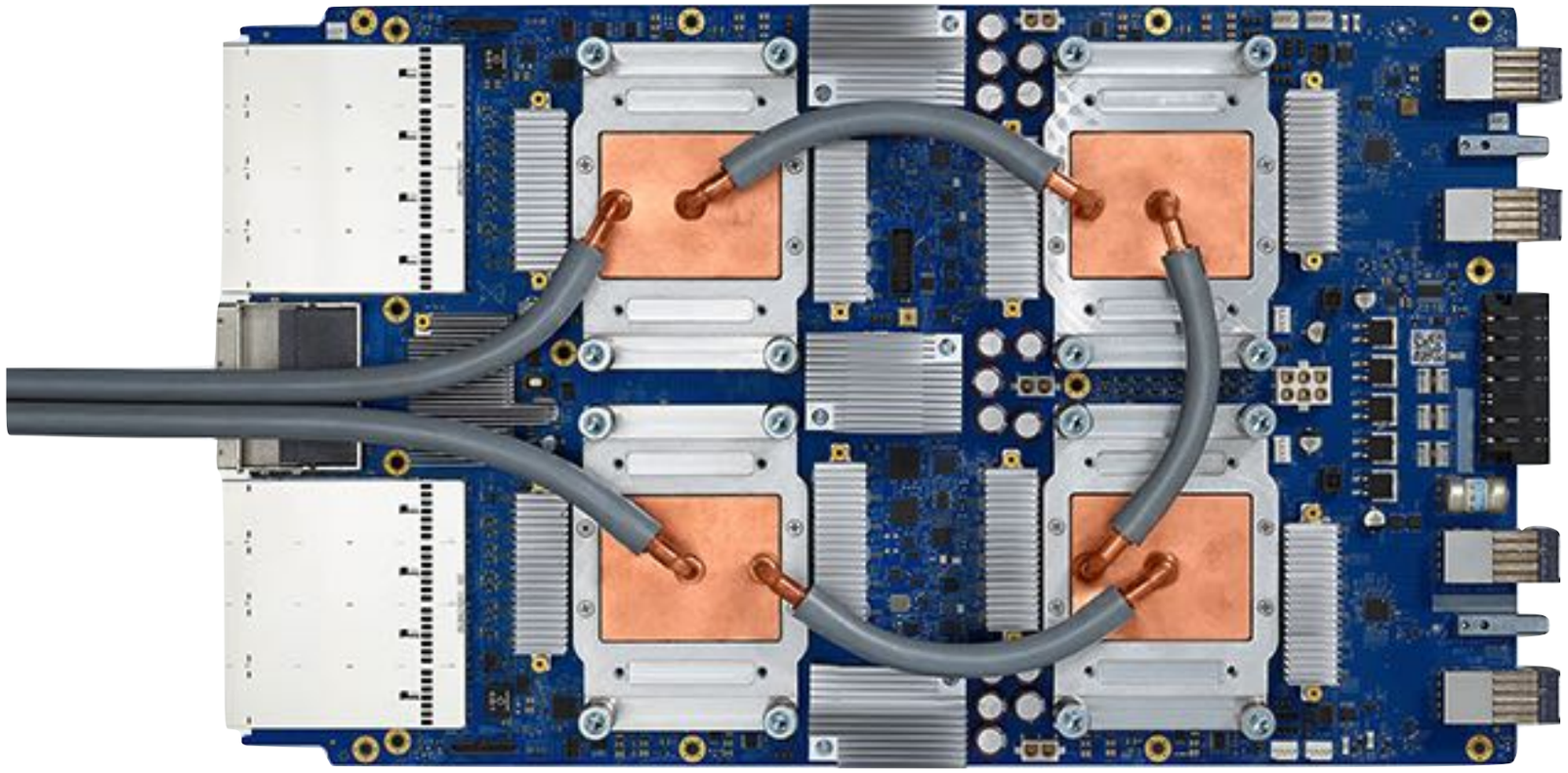
needs specific software to be used

there are basic linear algebra libraries that use it

often used in deep learning with great benefit

GPU

Google Tensor Processing Unit



Cloud TPU v3 Beta
420 teraflops
128 GB HBM

optimized for neural networks

GPU

MLlib library has support for GPUs

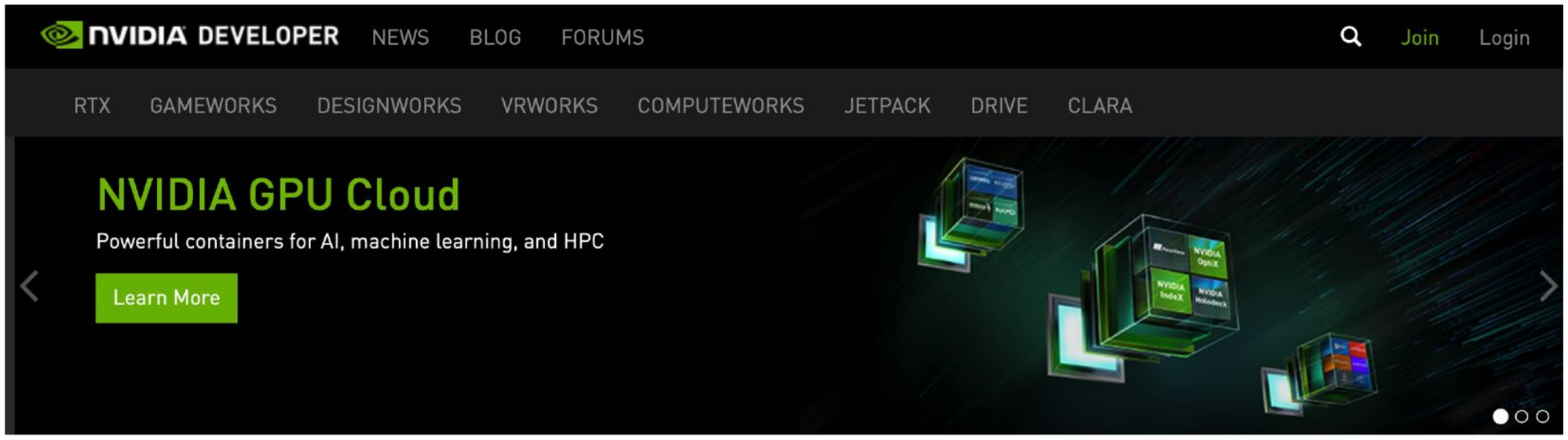
GPU

Google Cloud

cloudera

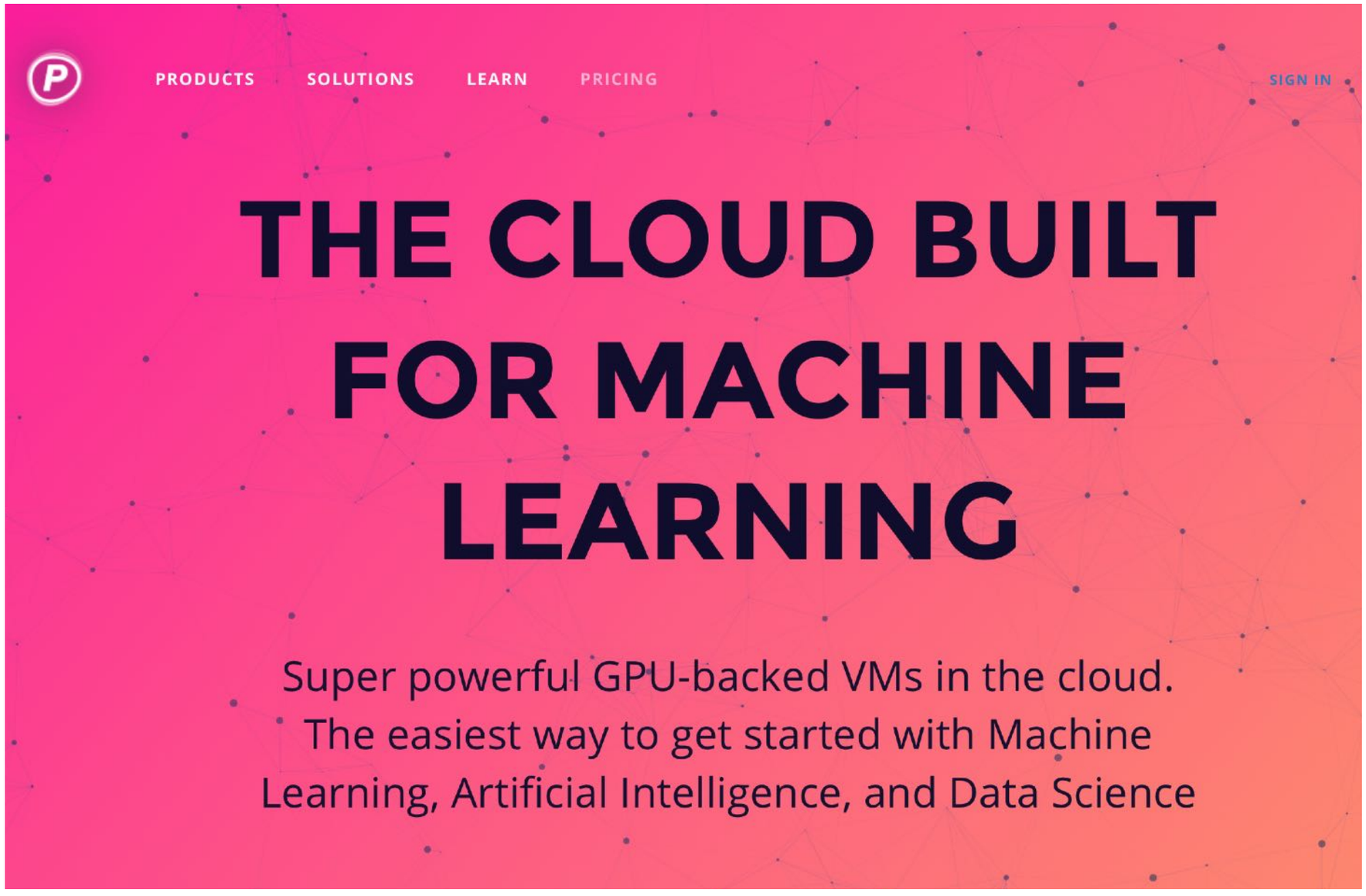
aws

GPU



https://developer.nvidia.com/nvidia-developer-zone

GPU



https://www.paperspace.com