Big Data Analytics:

A guide to economists making the transition to Big Data

Ulrich Matter (University of St.Gallen)

Workshop on empirical research with large datasets | Porto 19-20 DEC. 2022

Background/Introduction

Background: big data, where to begin?

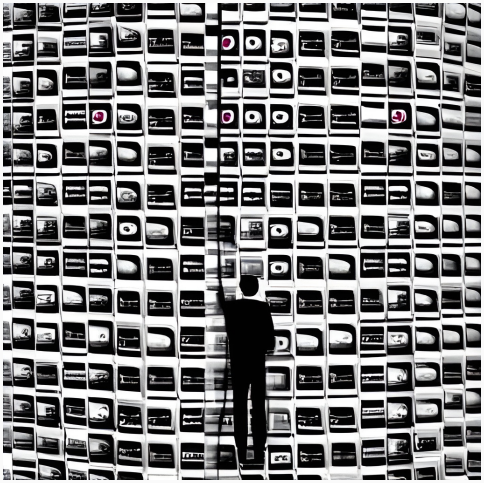


Figure 1: ‘A person analyzing a large amount of data’ by DALL-E 2

The data analysts'/economists' struggle (some examples)

Example 1: Overheard at a seminar.

Q:

Why do you use a sample of ten million? Uhm, I mean... apart from the fact that you can?

A:

The treatment is expected to have rather small effects, we need the power!

Example 2: water cooler chat during PhD times

Colleague:

> *We were running this fixed effects model and it took ages to compute*

Me:

> *Is it a memory issue?*

Colleague:

> *Ugh.. I don't know. We have just let it run for 3.5 days...*

Example 3: exchange with co-author

Co-author:

> *The bin-plot takes ages to compute and is huge in the compiled paper.*

Meanwhile on the engineering/software side of big data:

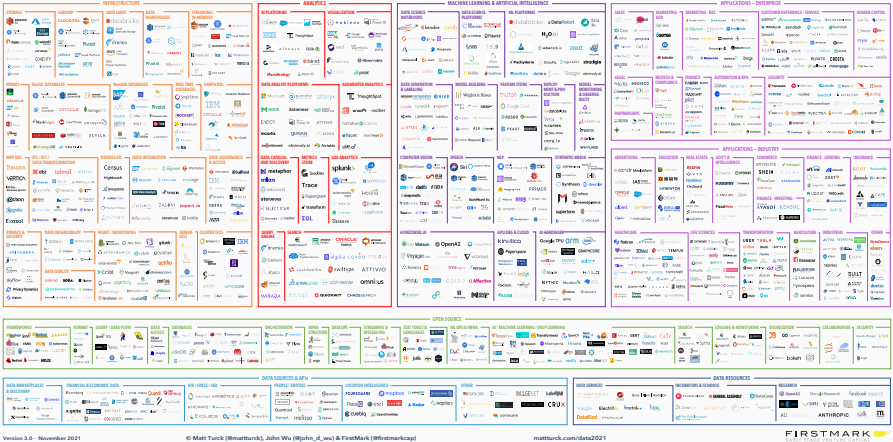


Figure 2: The Machine Learning, AI, and Data (MAD) Landscape, 2021. By Matt Turck (@mattturck), John Wu (@john_d_wu) & FirstMark (@firstmarkcap)

Personal perspective on Big Data Analytics

- *Focus during PhD/Postdoc-time* in the context of political economics and media economics.
- Conceptualization and teaching of *Big Data Analytics course* at the University of St.Gallen (graduate level).
- Book project *Big Data Analytics: A guide to data science practitioners making the transition to Big Data* (CRC Press, Data Science Series): umatter.github.io/BigData

This Talk

Agenda

1. Background/Introduction
2. Approaches to Big Data
3. Platform: Software/Hardware
4. Application: Some Examples
5. Wrap-up

Approaches to Big Data

Approaches: overview

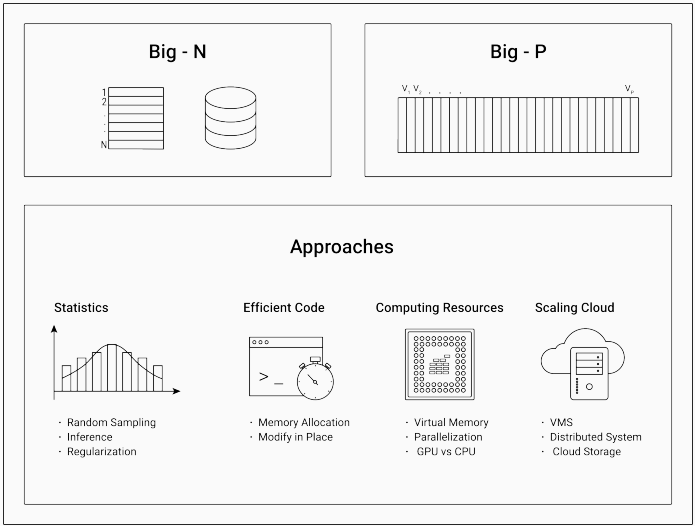


Figure 3: Domains of and approaches to Big Data Analytics.

The guide

The mantra: focus on transferable skills/knowledge (big data is a moving target).

1. Consider a statistics/econometrics solution.
2. Focus on few versatile high-level tools: here, R + SQL.
3. Recognize bottle-necks regarding the computational resources.
4. Use the cloud, but only if really necessary.

Platform

Software and hardware layers

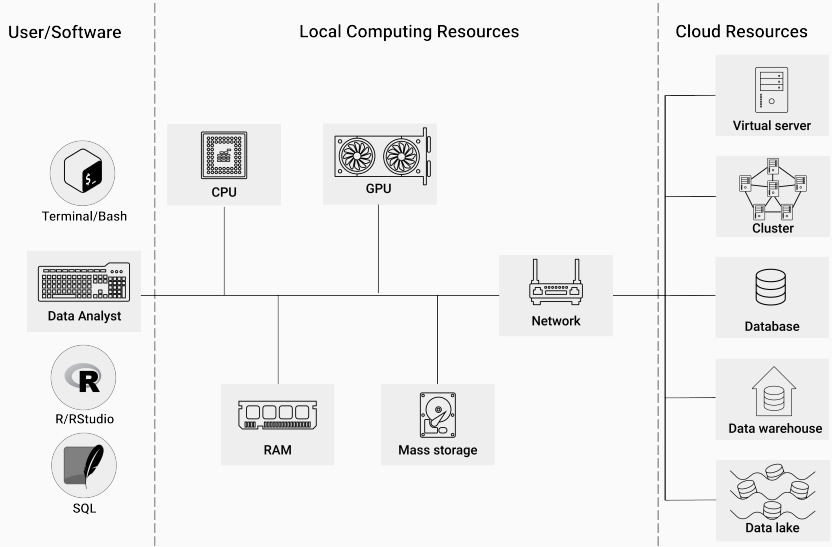


Figure 4: Software and hardware layers in Big Data Analytics.

Find bottlenecks in your R code

Tools to find bottlenecks in your big data analytics code: memory.

```
# packages
library(pryr) # memory profiling

# Example: how does a line of code affect memory
# initiate a vector with 1000 (pseudo)-random numbers
mem_change(thousand_numbers <- runif(1000))
```

```
15.6 kB
```

```
# initiate a vector with 1M (pseudo)-random numbers
mem_change(a_million_numbers <- runif(1000^2))
```

```
8 MB
```

Find bottlenecks in your R code

Tools to find bottlenecks in your big data analytics code: computing time.

```
# packages
library(bench) # computing time profiling

# Example: compare speed of alternative implementations
mark(
     # apply-approach to compute square roots
     sqrts1 <- sapply(thousand_numbers, sqrt),
     # exploit vectorization to compute square roots
     sqrts2 <- sqrt(thousand_numbers)
)[,c(1,4)]
```

Find bottlenecks in your R code

```
# A tibble: 2 x 3
  expression                                `itr/sec` mem_alloc
  <bch:expr>                                    <dbl> <bch:byt>
1 sqrts1 <- sapply(thousand_numbers, sqrt)      3333.   31.67KB
2 sqrts2 <- sqrt(thousand_numbers)            261692.    7.86KB
```

(Aside: use R's native vectorization whenever possible.)

Bottlenecks in applied econometrics

- Data preparation: raw data is *larger than RAM*.
- Regression analysis: large model matrix, $(\mathbf{X}^\top\mathbf{X})^{-1}$ – *a lot to compute (+ large objects)*
- Bootstrapping: *CPU at the limit?*
- Visualization: large scatter-plots: *too large vector images*.

Why does R/RStudio crash or slow down?

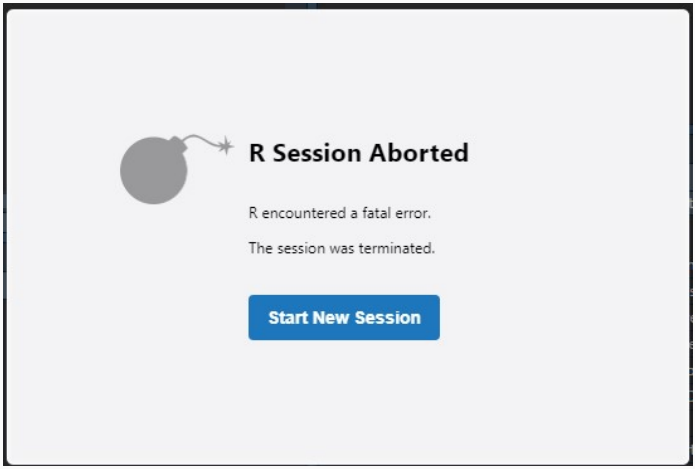


Figure 5: RStudio crash pop-up.

Why does R/RStudio crash or slow down?

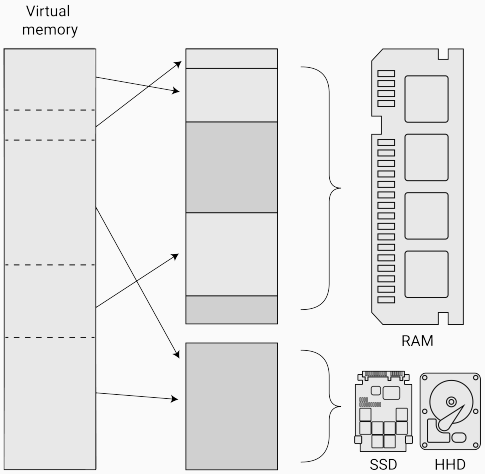


Figure 6: Illustration of virtual memory.

Hardware: use (virtual) memory the smart way

Out-of-memory approaches:

- Datasets are written in a easily-readable/writable structure to a dedicated part of the HD.
- The R session is connected to the dataset, but only contains the dataset's metadata (and connection information) in RAM.
- Robust to very large datasets, but processing is slower (reading from/writing to HD is slower than RAM).

Solutions for R: `ff`, `bigmemory`, Apache Spark (via `sparklyr` or `SparkR`).

Smart use of HD, and RAM – SQL + R

Exploit database ideas for simple analytics, learn SQL!

- Robust solutions to filter and prepare larger-than-memory datasets (e.g., `SQLite`).
- Specialized big data software solutions provide an SQL(-like) interface (Apache Spark, Apache Druid, AWS Athena, Google BigQuery, etc.)
- (No need to learn relational algebra, etc.)

Aside: row-based vs. column-based DBs

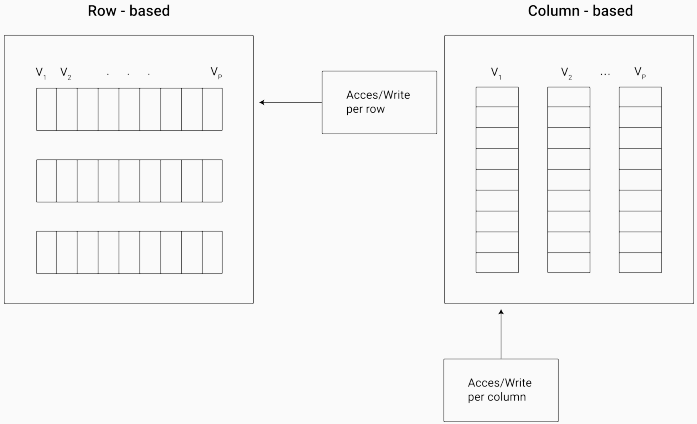


Figure 7: Illustration of row-based vs. column-based databases.

Column-based solutions: **Apache Druid, Google BigQuery, Amazon Redshift**.

Smart use of HD, and RAM – lazy evaluation

- Import metadata (enough to know what is in the dataset).
- Write analytics/data preparation scripts as if the dataset would be loaded into R.
- The data is only loaded into RAM *after evaluation* of all the analytics/data prep commands.

Figure 8: Apache Arrow logo. © 2016-2022 The Apache Software Foundation

Application

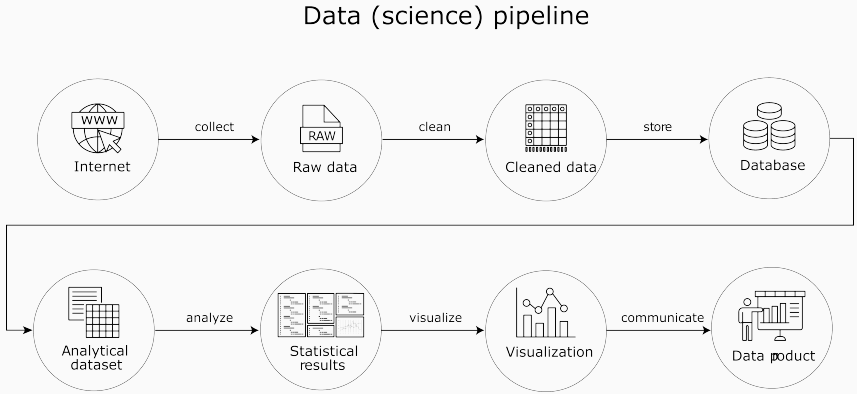


Figure 9: Data pipeline illustration.

Example 1: loading/filtering data – SQL + R

```r
# load packages
library(RSQLite)
# download example file
URL <-
"https://files.consumerfinance.gov/f/documents/NFWBS_PUF_2016_data.csv"
PATH <- paste0(getwd(), "/", basename(URL))
download.file(URL, PATH)

# initiate a database
con <- dbConnect(SQLite(), "mydb.sqlite")
# load data into the db
RSQLite::dbWriteTable(con, "fwb", PATH, overwrite=TRUE)
```

Example 1: loading/filtering data – SQL + R

```
# select/filter for analysis
query <-
"
SELECT
PUF_ID AS id,
FWBscore AS financial_wellbeing,
SWB_1 AS subjective_wellbeing
FROM fwb
"
adf <- dbGetQuery(con, query)
```

Example 1: loading/filtering data – SQL + R

```r
# inspect result
head(adf, 2)
```

```
     id financial_wellbeing subjective_wellbeing
1 10350                  55                    5
2  7740                  51                    6
```

Example 2: data aggregation with `arrow`

```
# data
# download the example data (3.4GB!) from here:
# https://bda-examples.s3.eu-central-1.amazonaws.com/tlc_trips.csv

# load packages
library(arrow)
library(dplyr)
library(pryr) # for profiling

# read the csv file
mem_change( taxi <-  read_csv_arrow("../data/tlc_trips.csv",
                      as_data_frame = FALSE))
```

```
5.72 MB
```

Example 2: Data aggregation with `arrow`

```
taxi
```

```
Table
27472535 rows x 18 columns
$vendor_name <string>
$Trip_Pickup_DateTime <timestamp[s]>
$Trip_Dropoff_DateTime <timestamp[s]>
$Passenger_Count <int64>
$Trip_Distance <double>
$Start_Lon <double>
$Start_Lat <double>
$Rate_Code <null>
$store_and_forward <int64>
$End_Lon <double>
$End_Lat <double>
```

Example 2: Data Aggregation with `arrow`

```
# clean the categorical variable, aggregate by group
taxi <-
   taxi %>%
   mutate(Payment_Type = tolower(Payment_Type))
```

Example 2: data aggregation with `arrow`

```
time_prof <- bench::mark(
taxi_summary <- taxi %>%
   mutate(percent_tip = (Tip_Amt/Total_Amt)*100 ) %>%
   group_by(Payment_Type) %>%
   summarize(avg_percent_tip = mean(percent_tip)) %>%
   collect()
)
```

Example 2: data aggregation with `arrow`

```
time_prof$median
```

```
[1] 385ms
```

```
time_prof$`itr/sec`
```

```
[1] 2.595632
```

The summary stats (by group) of 27,472,535 rows were computed in well below one second (!).

Example 2: data aggregation with `arrow`

```
taxi_summary
```

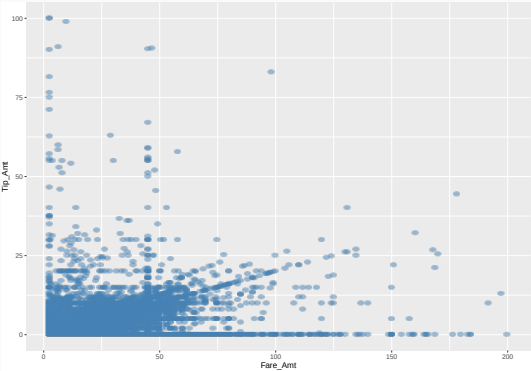
```
# A tibble: 4 x 2
  Payment_Type avg_percent_tip
  <chr>                  <dbl>
1 cash                 0.00599
2 credit              15.8
3 no charge            0.0631
4 dispute              0.0963
```

Example 3: visualization via rasterization

```
# load packages
library(ggplot2)
library(data.table) # fast reading of csv
library(scattermore) # for rasterization
# load data
taxi_dt <- fread("../data/tlc_trips.csv",
                 nrows = 1000000)
```

Example 3: visualization via rasterization

```
# plot
ggplot(taxi_dt, aes(y=Tip_Amt, x= Fare_Amt)) +
          geom_scattermore(pointsize = 3, color="steelblue", alpha=0.5)
```



Example 4: regression analysis with spark

```
# install the R-package
install.packages("sparklyr")
# local Spark installation
sparklyr::spark_install()
```

(this can take a while)

Example 4: regression analysis with spark

```
# load packages
library(sparklyr)
# connect with default configuration
sc <- spark_connect(master="local")
# load data
taxi_spark  <- copy_to(sc, taxi_dt)
```

Example 4: regression analysis with spark

```
# reg model specification
model <- Tip_Amt ~ Fare_Amt + Passenger_Count
# fit the model
mem_change(fit1_spark <- ml_linear_regression(taxi_spark, model))
# compute summary stats
summary(fit1_spark)
```

Example 4: regression analysis with spark

```
Deviance Residuals (approximate):
     Min       1Q   Median       3Q      Max
-9.53120 -0.43811 -0.25108 -0.09631 90.72312

Coefficients:
    (Intercept)        Fare_Amt Passenger_Count
   -0.135939484     0.064490020    -0.006364645

R-Squared: 0.127
Root Mean Squared Error: 1.218
```

Wrapping up

Takeaways (a suggestion)

1. Don’t dismiss statistics solutions: why not just take a random sample?
2. Know your R (memory allocation!) and SQL (way more than RDBMS).
3. Recognize the bottlenecks regarding computing resources: RAM? CPU? HD?
4. Exploit local options before turning to the cloud.
5. Invest in transferable big data skills/knowledge.
6. Consider Apache Spark (`sparklyr`), Apache Arrow (`arrow`), and Apache Druid.

Promising developments

What to keep an eye on? (some suggestions)

- `dplyr`-interfaces
- SQL-like interfaces to data warehouses, data lakes, semi-structured data
- Cloud: serverless data analytics solutions

Resources/follow up

- Taxi trips data used in examples:
  https://bda-examples.s3.eu-central-1.amazonaws.com/tlc_trips.csv
- Quarto-file of this slide deck (incl. all code examples):
  github.com/umatter/bigdata_workshop_porto
- Detailed explanations, corresponding tutorials: umatter.github.io/BigData

Thanks! Questions?