SAT R DAY 2016 BUDAPEST

Door to door fraud detection

András Tajti
Andego

2016-09-03

What we promised

The client will provide new car insurance claim data every day

- *Information about the accident*
- *Information about participants*
- *Information about contract*

We will compute scores for every claim according to

- *claim features based on accident data*
- *participants' previous claim data*
- *a network based on customer and company database*

What the client wants

To upload data automatically every night

Get the fresh results as Excel workbook in e-mail next morning

See all cases in another program

What is needed to do with R

Get data from FTP server (`RCurl`)

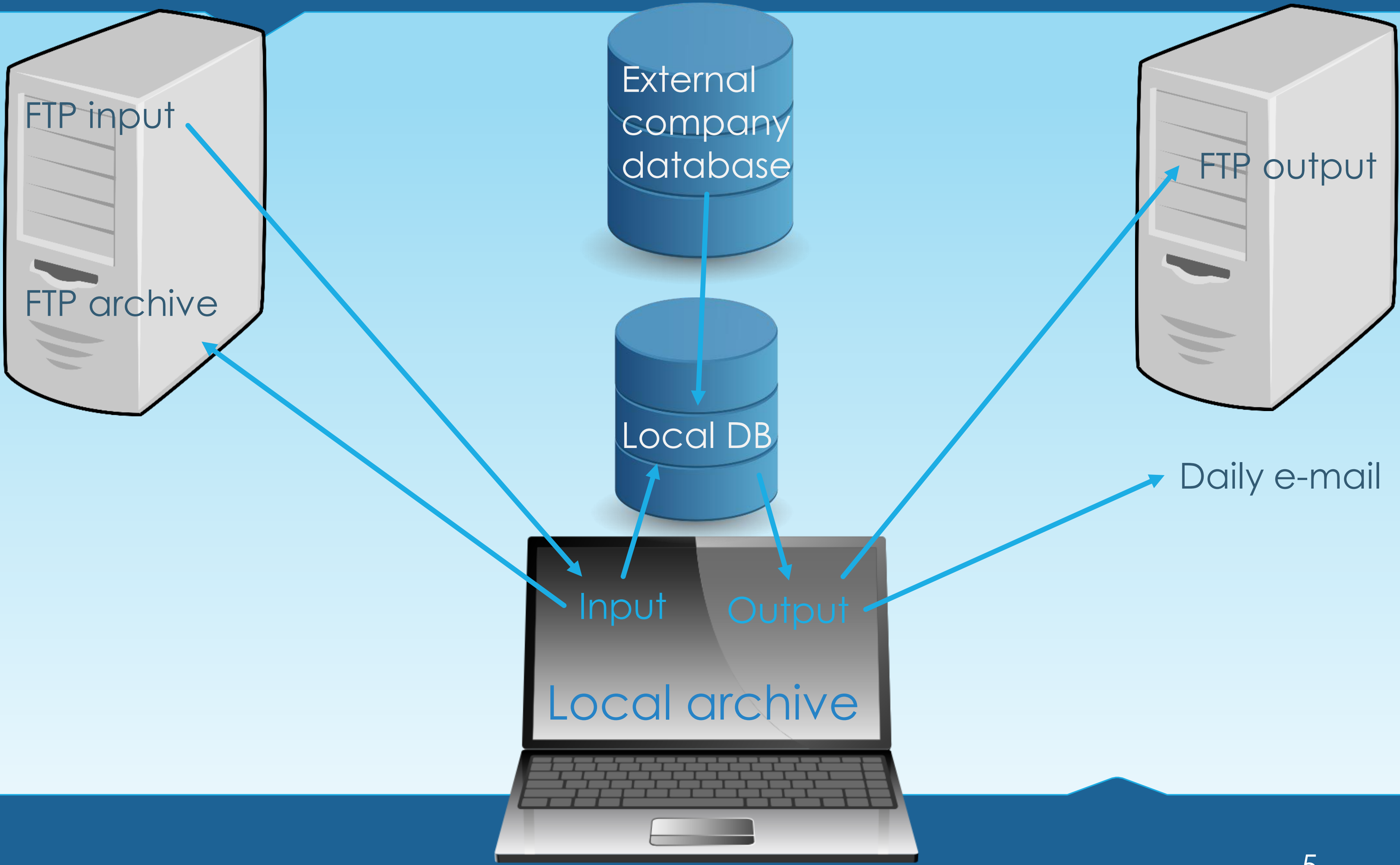
Identify customers (`data.table`, `igraph`)

Connect them with external DB available through SSH (`RMySQL`)

Send fresh results as an Excel attachment in e-mail (`xlsx`, `mailR`)

Upload to another FTP server for the reporting system (`RCurl`)

Data flow

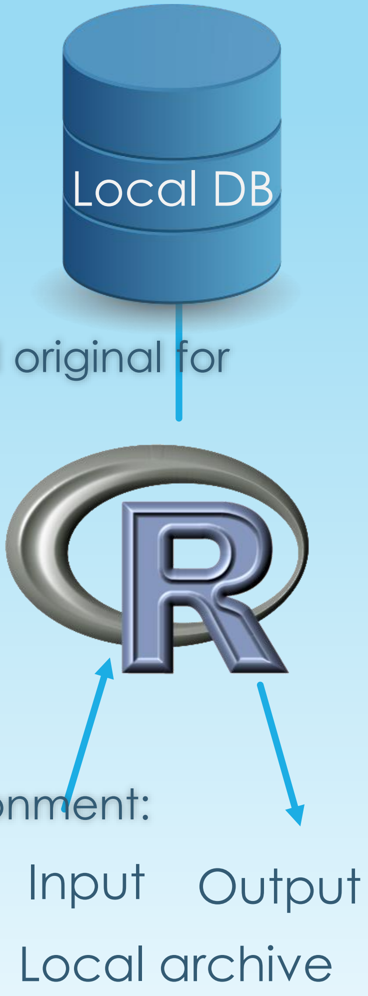

FTP input
FTP archive
External company database
Local DB
FTP output
Daily e-mail
Input
Output
Local archive

Connect to MySQL DMBS

Local and remote

Local database

- Almost copy-paste case with **`RMySQL`**
- Difference between Windows and Ubuntu:
  - `dbListTables()` returns lowercase names for Windows, and original for Ubuntu
  - Encoding must be set explicitly on Ubuntu
  - File paths must use / not \\
- Chatch and solution
  - Sometimes database connection is lost
  - A function to create the connection object in global environment:
    - `Dbcon <- function(){DB <<- dbConnect()}`



External database

- SSH:
  - With external channel it works (`ssh` in terminal), but it must be always open and cannot controlled from R,
  - From R, with `system()` it presented the remote computer's terminal
  - With `system(ssh, wait=FALSE)` it also failed
- Once the connection established
  - set encoding with `dbGetQuery(DB, "SET NAMES utf8")`
  - use `RMySQL` as with the local database

FTP & data

RCURL + FTP commands

- Task: Check if there is new a-priori upload folder, if not, download files, then move them to a folder named to `Sys.Date()`
- Conclusions:
  - Windows: ftp commands did not worked through `system()`
  - `list.files()` did not worked as the connection has authentication
  - Use `RCurl::curlPerform()` and its arguments properly:
    - `curlPerform(`
      - `url= ” ftp://server.address”`
      - `quote= ” ftp command”`
      - `userpwd= ” user:pwd”)`

Data cleaning

  - People write scripts to generate tables ⟹
- Automatically generated tables' quality can (and should) be questioned:
  - We got files with headers in the middle
  - Files with 0 rows
- Proper cleaning can use much memory
  - If the cleaning script is sourced, `gc()` will not give back all memory
  - Use system('Rscript "path/to/script" argument')
    - Before that, copy 'path/to/Rscript.exe' to the PATH system variable on Windows
    - This also makes a script more robust but debugging more pain, as when the script called through `system()` fails, R will move on to the next statement

Customer matching

Identifying participants

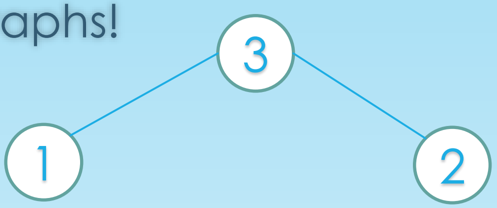
- Basics:
  - Two person are the same if the name, and two other from birth date, address and mother’s name are matching
  - Two firms are the same if their name and address are matching
  - Not just insurance data, but company information must be used, too
- Obstacles:
  - Person matching needs three merges, after handling NAs
  - Matching should be fuzzy, merging does not support that
- Solutions:
  - Merging is faster and more memory-efficient with `data.table`
  - Size of data to merge can be decreased with pre-selecting possible reference
  - Fuzzy matching — we didn’t solved that yet ☹

Connect customers

- After identifying pairs, how to detect indirect connections?

id	Name	Birth	Mother's name	Address
1	Jimmy Paron	1976.02.04	Julie Smith	
2	Jimmy Paron		Julie Smith	Mountain View
3	Jimmy Paron	1976.02.04	Julie Smith	Mountain View

- Graphs!



- Obstacle:
  - Memory usage: graph from all records filled up all memory and swap (~11GiB)
- Solution:
  - Multiple switching tables:
    - row number ↔ unique row number (10 million ↔ 5.7 million)
    - Unique row ↔ exactrly matching records (5.7 million vs 370 000)
      - data.table[, grp := .grp]
    - exactly matching records ↔ new id (370 000 ↔ 360 000)

Create Excel file

With conditional formatting!

xlsx can do magic!

(though you'll have to beg() for it)

- Create multi-sheet conditionally formatted xlsx files in just a few easy steps!
- 1.: create the workbook
  - `createWorkbook()`
  - `addDataFrame()`
- 2.: Check every cell for the value and set color:
  - `sheet <- getSheets(wb)[sheetNumber]`
  - `row <- getRows(sheet, rowIndex) #for each row`
  - `cols <- ncol(get(sheet$getSheetName()))`
  - `cell <- getCells(row, colIndex) #for each coloumn`
  - `value <- getCellValue(cell[[1]])`
  - `lapply(cell, setCellStyle, cellStyle())`
- Save it asap!
  - `saveWorkbook()`

Lessons learned

- Use the same operating system, to avoid character encoding and other issues
- Never trust incoming data
- For every larger step use a different R session for robustness and memory efficiency
- Use data.table for speed and memory efficiency
- With R you can do almost everything

Andego Tanácsadó Kft.

1092 Budapest,
Ráday utca 42. 3/308-309.

06-1/210-0446

www.andego.hu

András Tajti

atajti.hu
atajti@andego.hu