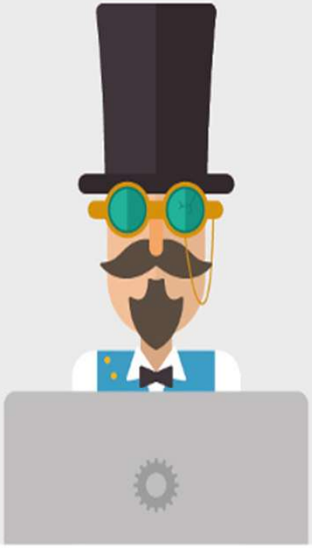
Analysing GitHub commits with R

Barbara Fusinska

@BasiaFusinska

barbarafusinska.com



geecon

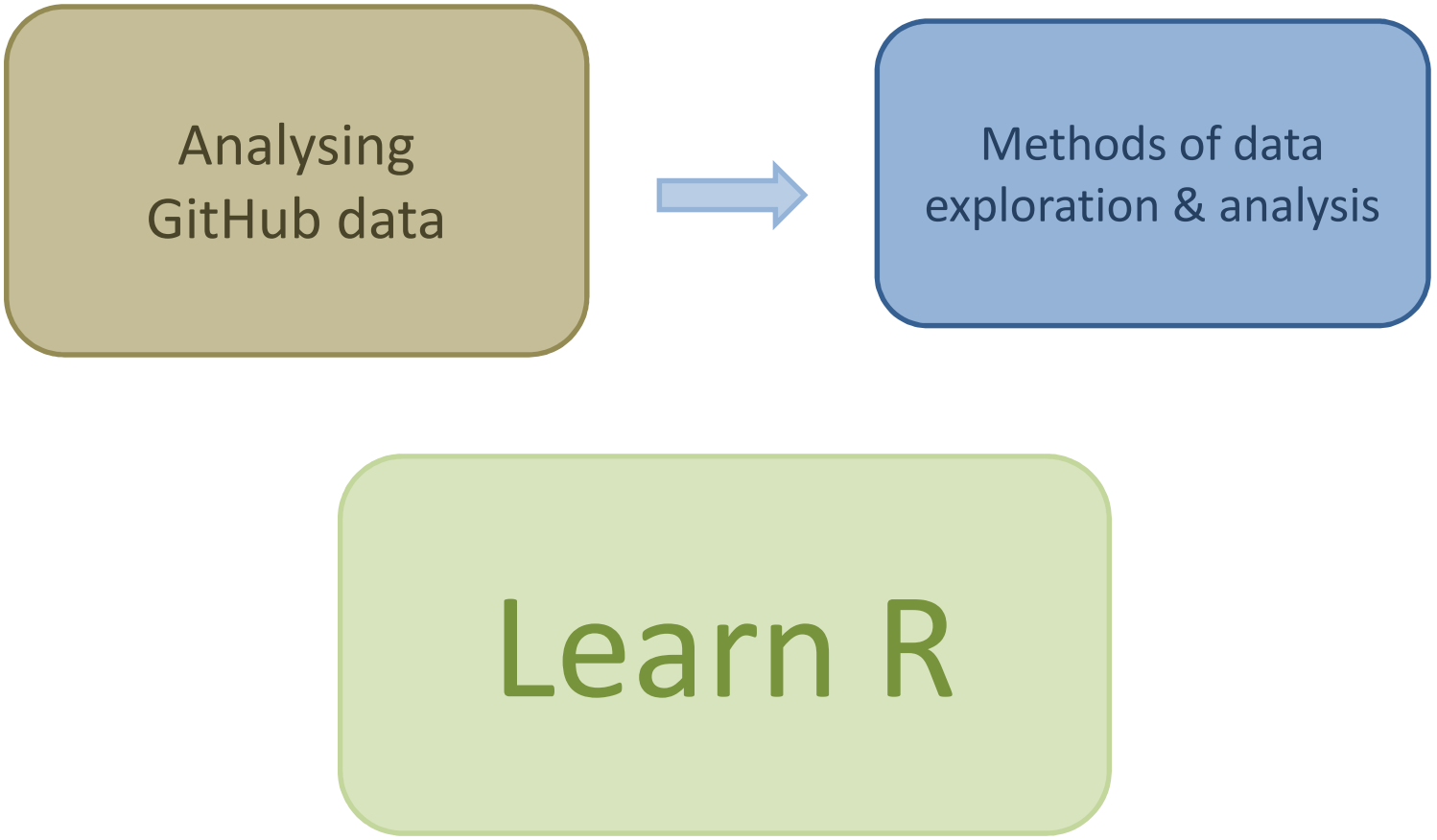
About me

Programmer

Manager

Sweet tooth

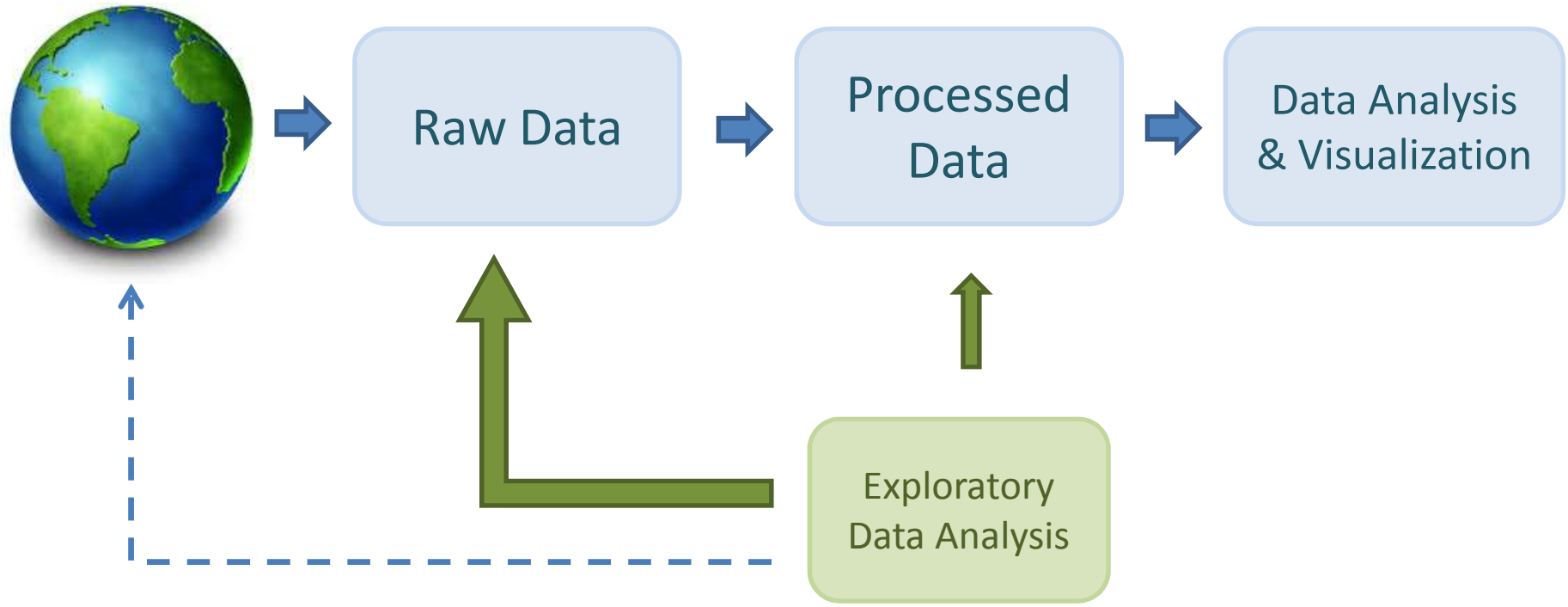
Goals


Analysing
GitHub data
Methods of data
exploration & analysis
Learn R

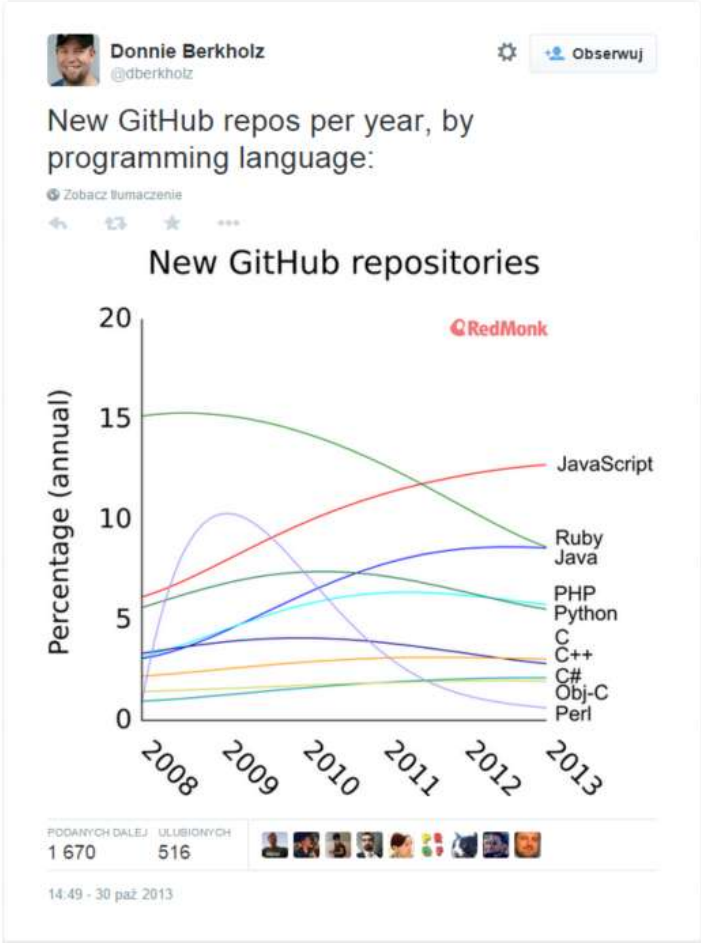
Agenda

- Data analysis
- R basics
- Data exploration & processing
- Big Data
- Outside GitHub:
  - Data visualisation
  - Libraries & tools

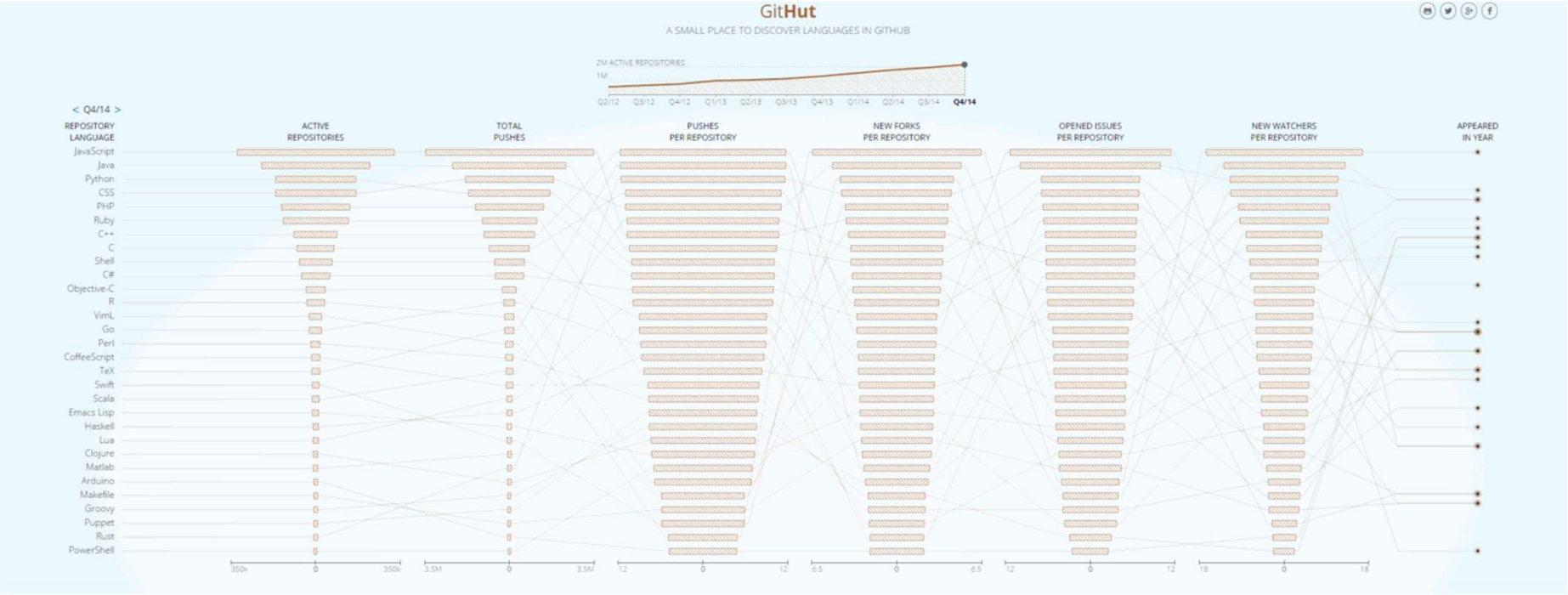
Data analysis process



What do we want to find out?

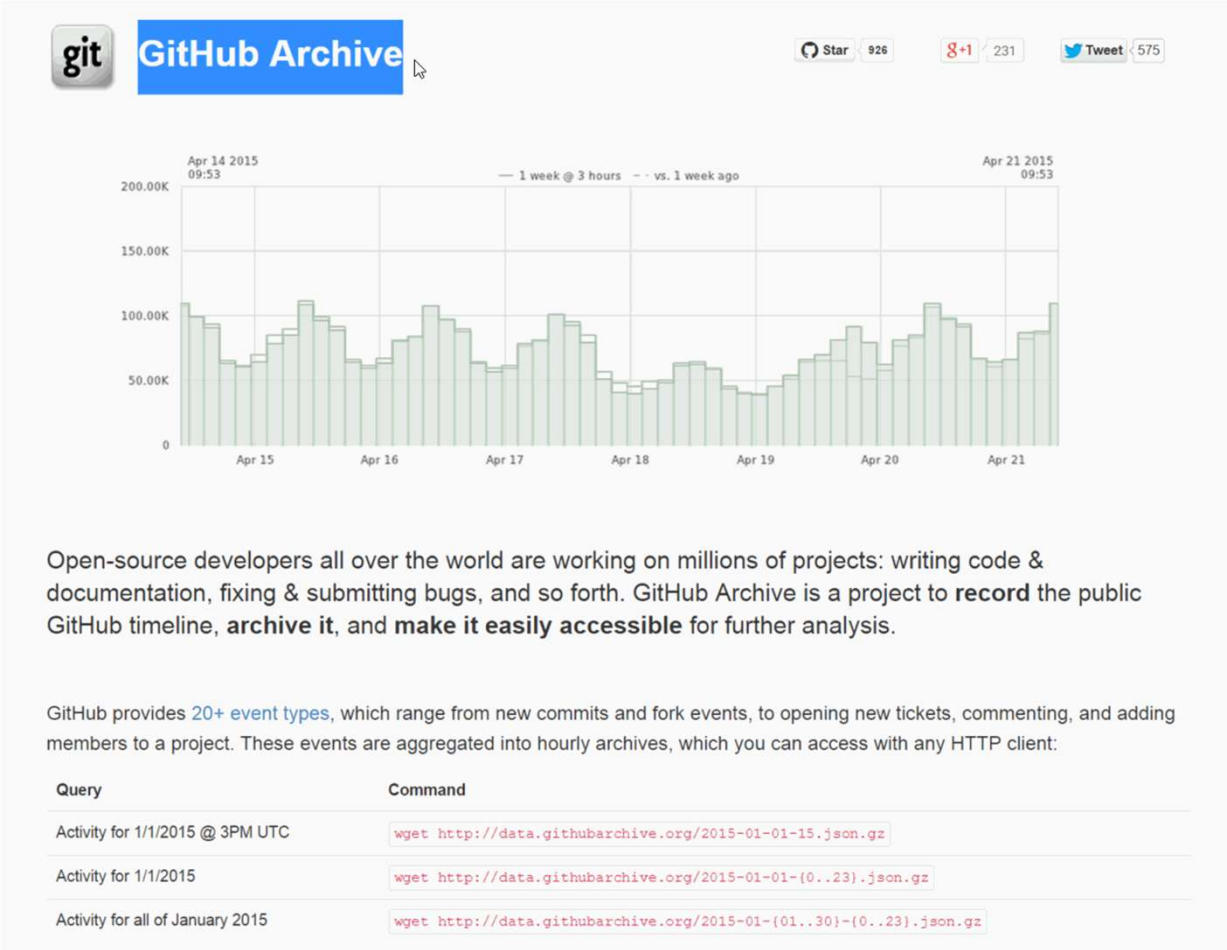


GitHut Visualisation



http://githut.info/

GitHub Archive



https://www.githubarchive.org/

Google BigQuery

Analyzing event data with BigQuery

The entire GitHub Archive is also available as a public dataset on Google BigQuery: the dataset is automatically updated every hour and enables you to run arbitrary SQL-like queries over the entire dataset in seconds - i.e. no need to download or process any data on your own. To get started:

1. If you don't already have a Google project...
   - Login into the Google Developer Console
   - Create a project and activate the BigQuery API
2. Open public dataset: https://bigquery.cloud.google.com/table/githubarchive:day.events_20150101
3. Execute your first query...

```
/* count of issues opened, closed, and reopened between 1/1/2015 and 2/1/2015 */
SELECT event as issue_status, COUNT(*) as cnt FROM (
  SELECT type, repo.name, actor.login,
    JSON_EXTRACT(payload, '$.action') as event,
  FROM (TABLE_DATE_RANGE([githubarchive:day.events_],
    TIMESTAMP('2015-01-01'),
    TIMESTAMP('2015-02-01')
  ))
  WHERE type = 'IssuesEvent'
)
GROUP by issue_status;
```

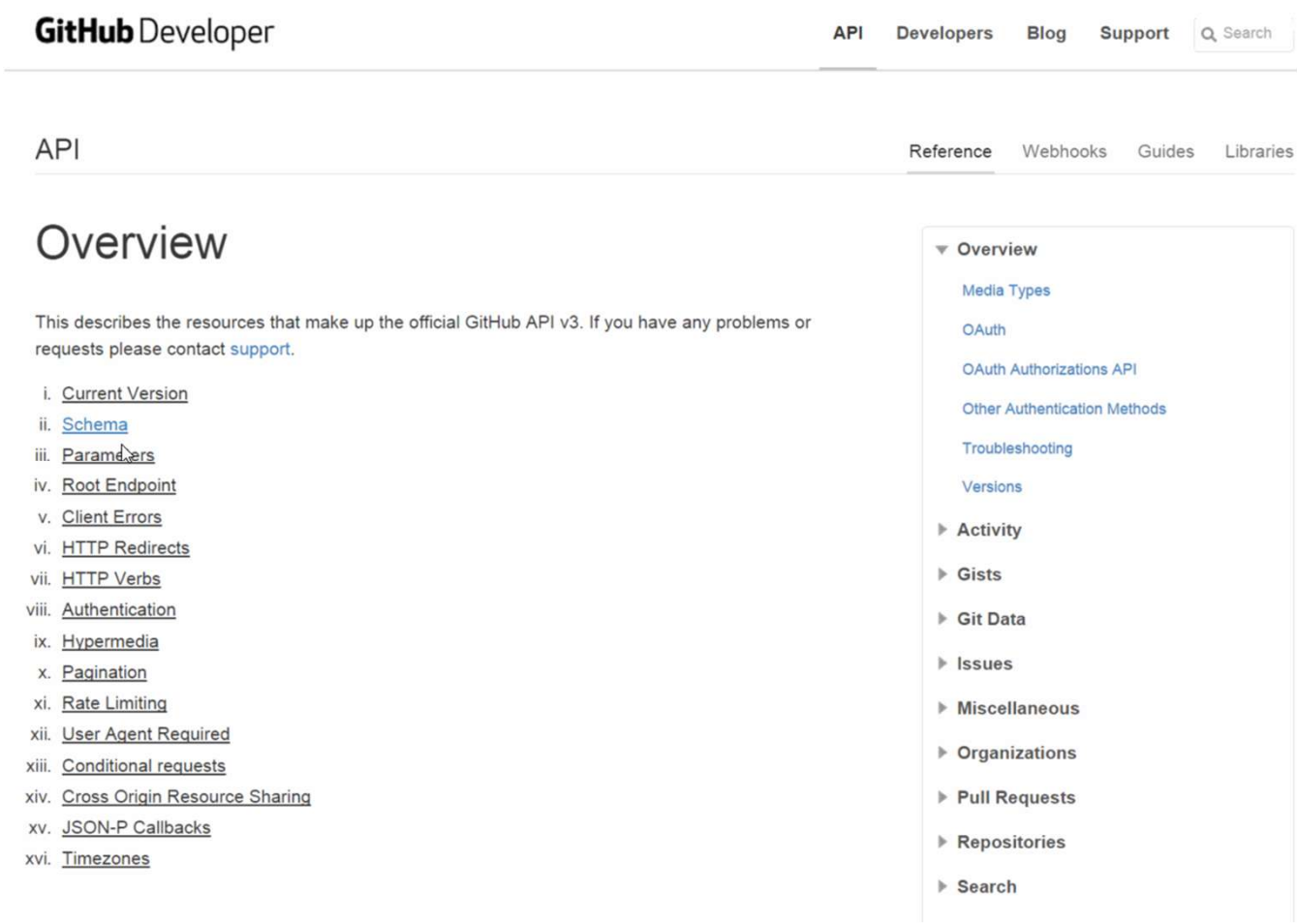
bq.sql hosted with ♥ by GitHub view raw

For convenience, note that there are multiple tables that you can use for your analysis:

1. **year dataset:** `2011`, `2012`, `2013`, and `2014` tables contain all activities for each respective year.
   - The schema is in a "flattened" format where each field is mapped into a distinct column.
   - The schema is the same between all years.
2. **month dataset:** contains activity for each respective month between 2011-today - e.g. `201501`.
   - The schema for 201101~201412 tables is the same as 2011-2014 year tables: flattened with each field in a distinct column.
   - The schema for 201501+ tables contains nested records plus a JSON encoded payload - see below.
3. **day dataset:** contains activity for each day starting on January 1, 2015 - e.g. `20150101`.
   - The schema contains distinct columns for common activity fields (see same response format), plus a payload string field

https://cloud.google.com/bigquery/what-is-bigquery

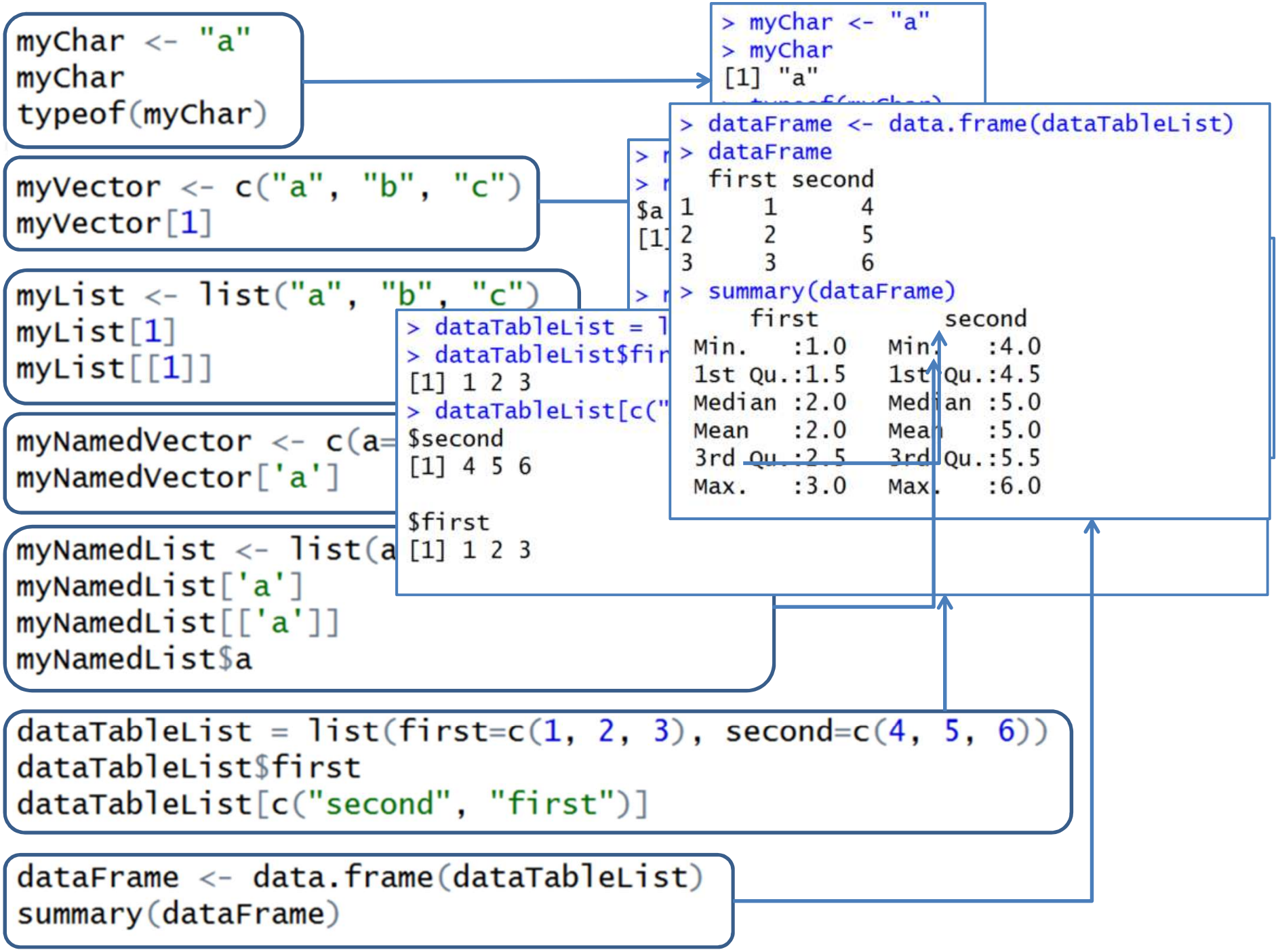
GitHub API



https://developer.github.com/v3/

DEMO: R basics

- RStudio
- Data types
- Data filtering


myChar <- "a"
myChar
typeof(myChar)
> myChar <- "a"
> myChar
[1] "a"
myVector <- c("a", "b", "c")
myVector[1]
myList <- list("a", "b", "c")
myList[1]
myList[[1]]
myNamedVector <- c(a=
myNamedVector['a']
myNamedList <- list(a
myNamedList['a']
myNamedList[['a']]
myNamedList$a
> dataTableList = l
> dataTableList$fir
[1] 1 2 3
> dataTableList[c("
$second
[1] 4 5 6
$first
[1] 1 2 3
> dataFrame <- data.frame(dataTableList)
> dataFrame
first second
1 1 4
2 2 5
3 3 6
> summary(dataFrame)
first second
Min. :1.0 Min. :4.0
1st Qu.:1.5 1st Qu.:4.5
Median :2.0 Median :5.0
Mean :2.0 Mean :5.0
3rd Qu.:2.5 3rd Qu.:5.5
Max. :3.0 Max. :6.0
dataTableList = list(first=c(1, 2, 3), second=c(4, 5, 6))
dataTableList$first
dataTableList[c("second", "first")]
dataFrame <- data.frame(dataTableList)
summary(dataFrame)

```r
listToFilter <- list("aa", "abc", "bc")
lResult = listToFilter[nchar(listToFilter) == 2]
lResult
lResult = Filter(function(element) {nchar(element) == 2}, listToFilter)
```

```r
listToFilter <- list(first=c(1, 2, 3), second=c(4, 5, 6), third=c(7, 8, 9))
dfToFilter <- data.frame(listToFilter)
dfToFilter
dtResult <- dfToFilter[dfToFilter$first %% 2 == 1,]
dtResult
```

```r
column <- dfToFilter[,2]
column
columns <- dfToFilter[,c(1,3)]
columns
columns <- dfToFilter[,c('first','second')]
columns
```

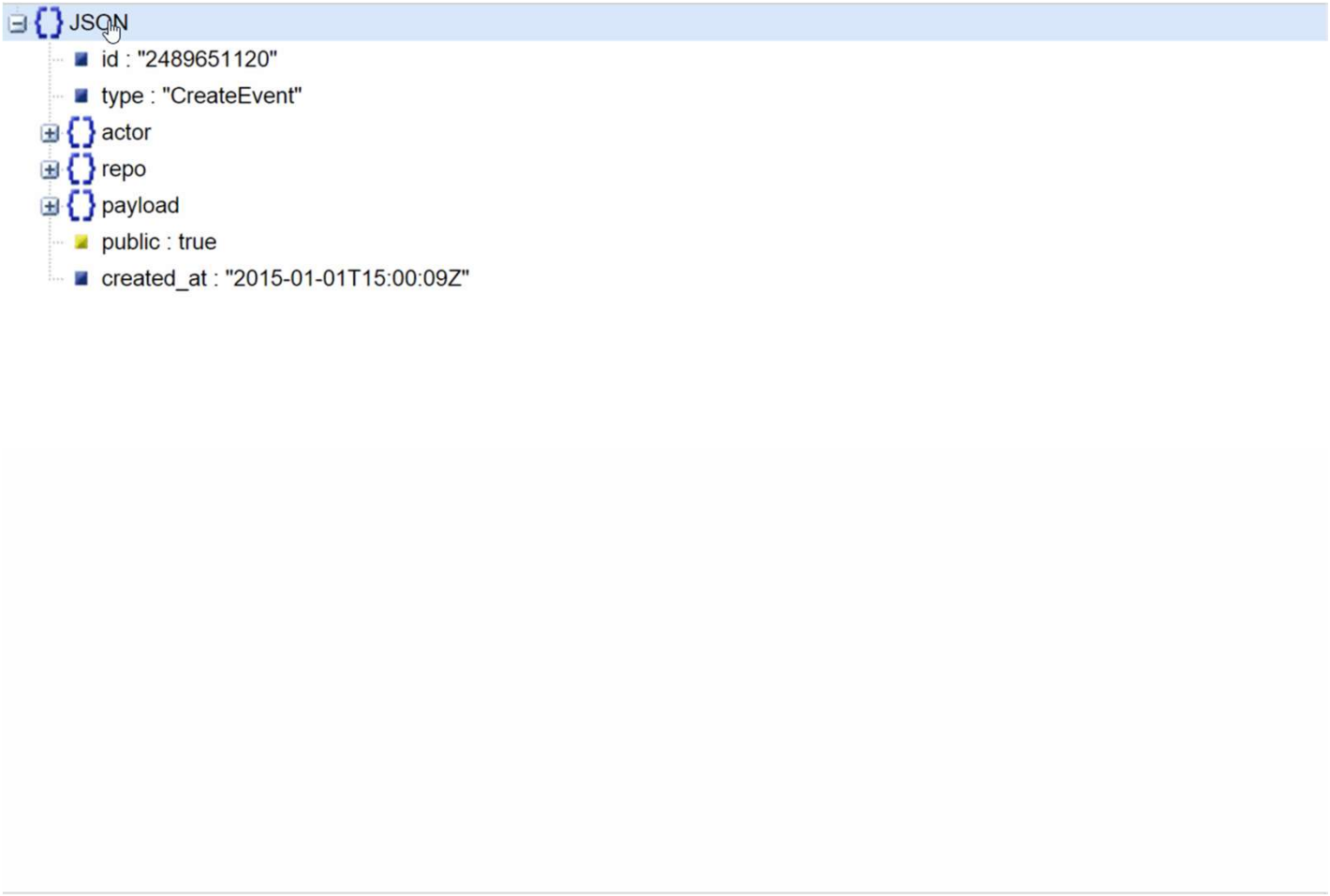
```
> dfToFilter <- data.frame(list
> dfToFilter
  first second third
1     1      4     7
2     2      5     8
3     3      6     9
> dtResult <- dfToFilter[dfToFi
> dtResult
  first second third
1     1      4     7
3     3      6     9
```

```
> column <- dfToFilter[,2]
> column
[1] 4 5 6
> columns <- dfToFilter[,c(1,3)]
> columns
  first third
1     1     7
2     2     8
3     3     9
> columns <- dfToFilter[,c('first','second')]
> columns
  first second
1     1      4
2     2      5
3     3      6
```

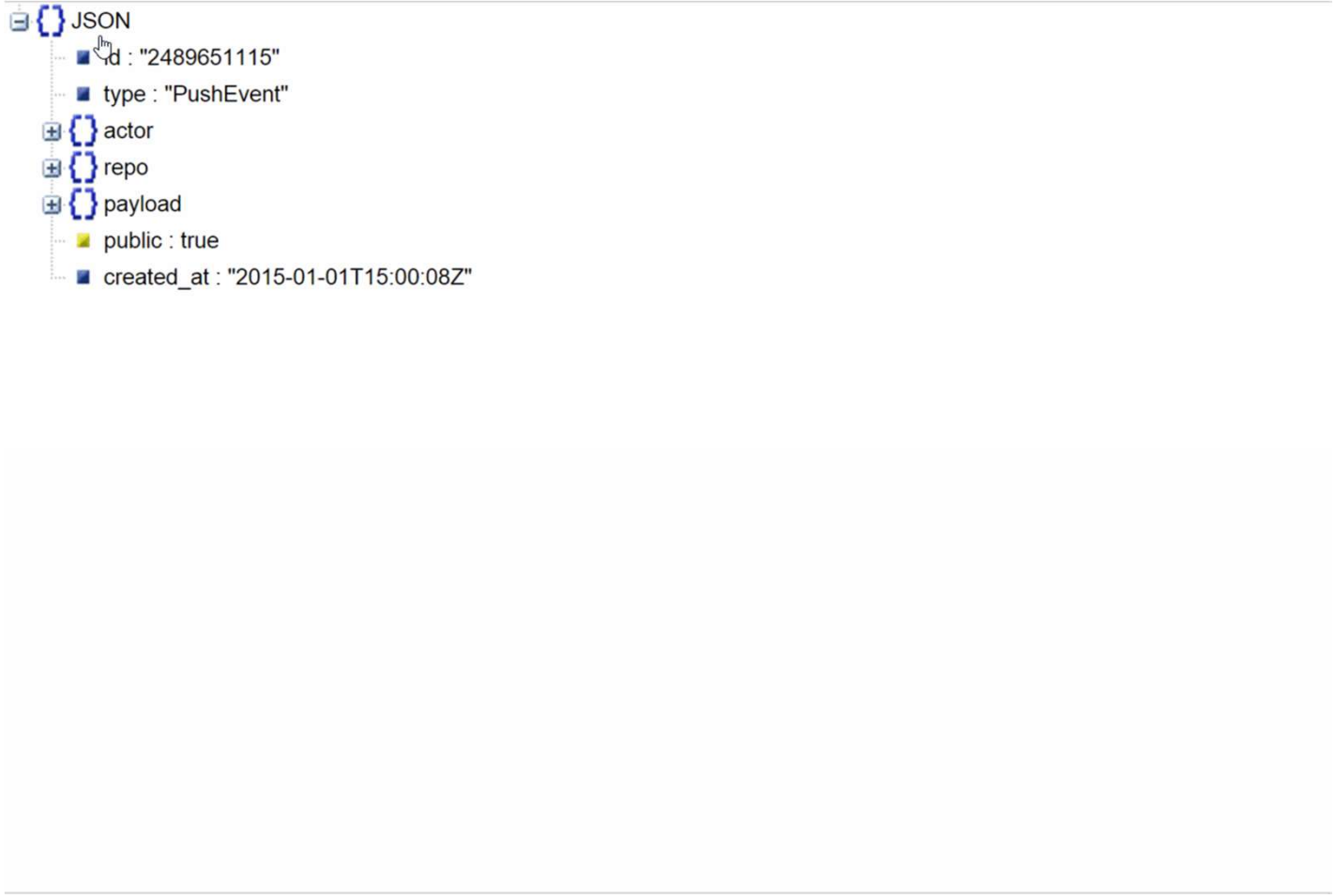
DEMO: Active repositiories

- What is an active repository?
- Where can we get data from?
- How to get useful information?
- How to handle missing data?

GitHub Archive CreateEvent

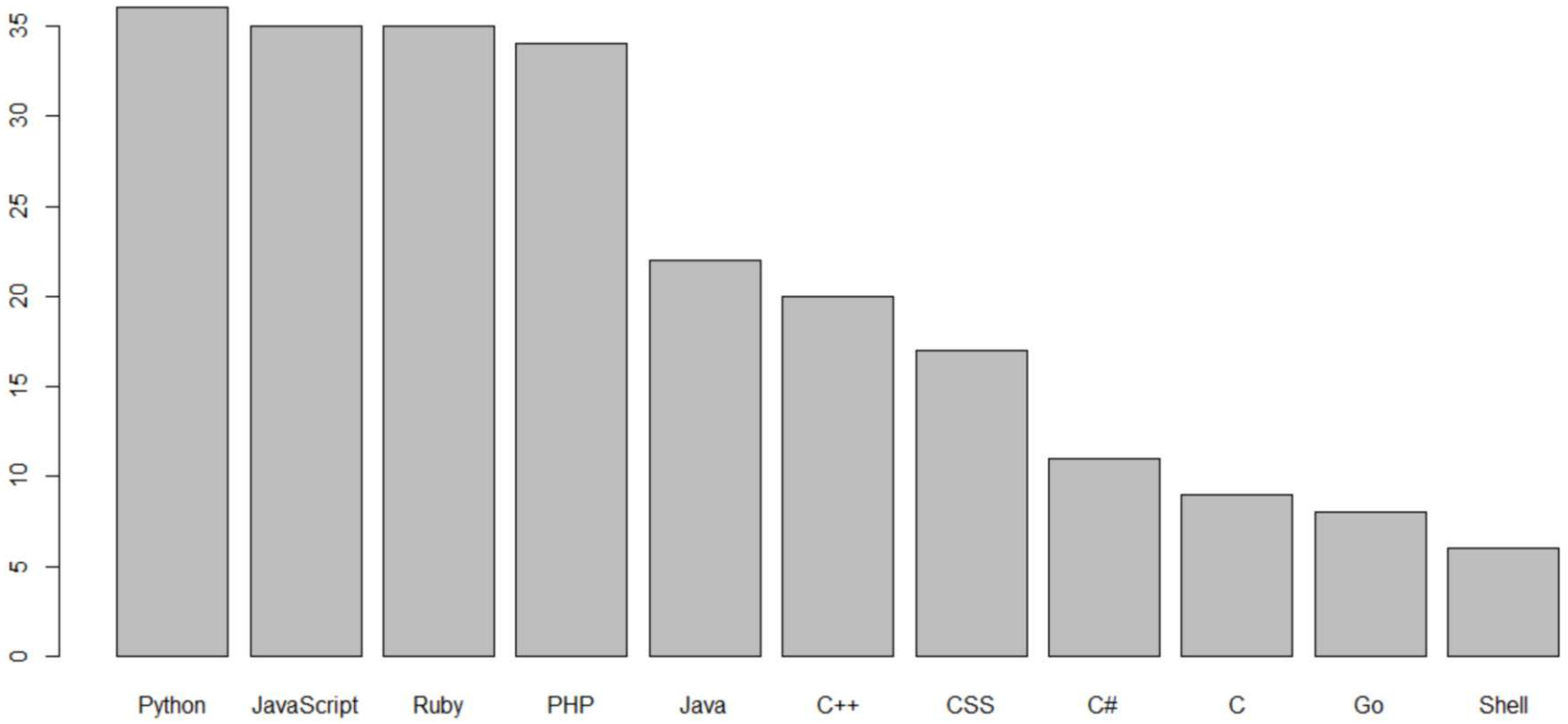


GitHub Archive PushEvent



GitHub Archive PullRequestEvent

Expected result



Read Pull Requests

```
library("rjson")

readEventsLists <- function(file, eventNames) {
  lines <- readLines(file)
  jsonEvents <- lapply(lines, fromJSON)
  specificEvents <- Filter(function(e){ e$type %in% eventNames}, jsonEvents)
  return(specificEvents)
}

fileName <- "Data/2015-01-01-15.json"

pullRequests <- readEvents(fileName, list("PullRequestEvent"))

repositoryData <- lapply(pullRequests, function(x) {
  c(id=x$payload$pull_request$base$repo$id,
    language=x$payload$pull_request$base$repo$language)
})

repositoryDataFrame <- data.frame(do.call(rbind, repositoryData))
head(repositoryDataFrame)

summary(repositoryDataFrame)
```

Summary Output

```
> head(repositoryDataFrame)
        id   language
1  3542607        C++
2 10391073     Python
3 28668460     Python
4 28608107       Ruby
5  5452699 JavaScript
6 19777872         C#
> summary(repositoryDataFrame)
        id              language
 28648149: 12   Ruby       : 66
 28688863:  8   PHP        : 55
 20413356:  5   Python     : 53
 28668553:  5   JavaScript: 47
 10160141:  4   C++        : 30
 206084  :  4   Java       : 30
 (Other) :436   (Other)    :193
>
```

Unique results

```
uniqueData <- unique(repositoryData)
repositoryDataFrame <- data.frame(do.call(rbind, uniqueData))
summary(repositoryDataFrame)
```

```
> uniqueData <- unique(repositoryData)
> repositoryDataFrame <- data.frame(do.call(rbind, uniqueData))
> summary(repositoryDataFrame)
        id              language
 25994257:  2   Python     : 36
 28528325:  2   JavaScript: 35
 10126031:  1   Ruby       : 35
 10160141:  1   PHP        : 34
 10344201:  1   Java       : 22
 10391073:  1   C++        : 20
 (Other) :297   (Other)    :123
```

Language information

```
languages <- table(repositoryDataFrame$language)
head(languages)
languagesNames <- names(languages)
languagesNames
```

```
> languages <- table(repositoryDataFrame$language)
> head(languages)

12745174 13381132 13723754 14098069  1652784 20825841
       1        1        1        1        1        1
> languagesNames <- names(languages)
> languagesNames
 [1] "12745174"     "13381132"     "13723754"     "14098069"     "1652784"      "20825841"
 [8] "2147969"      "22369870"     "22440251"     "25994257"     "26539815"     "27859829"
[15] "2794457"      "28528325"     "28532420"     "28663952"     "28668553"     "28686835"
[22] "28688390"     "28688581"     "28688617"     "28688727"     "28688734"     "28688837"
[29] "Bluespec"     "C"            "C#"           "C++"          "Clojure"      "CoffeeScript"
[36] "DM"           "Elixir"       "Emacs Lisp"   "F#"           "Go"           "Haskell"
[43] "Java"         "JavaScript"   "Julia"        "Lua"          "Makefile"     "Matlab"
[50] "Perl"         "PHP"          "Puppet"       "Python"       "Ruby"         "Rust"
[57] "Shell"        "Swift"        "TeX"          "VimL"
```

Missing information

```
lang <- languagesNames[2]
lang
repositoryDataFrame$id[repositoryDataFrame$language == lang]
```

```
> lang <- languagesNames[2]
> lang
[1] "13381132"
> repositoryDataFrame$id[repositoryDataFrame$language == lang]
[1] 13381132
303 Levels: 10126031 10160141 10344201 10391073 10435644 10484513 10658331 1099265 11075527 ... 9852918
> 
```

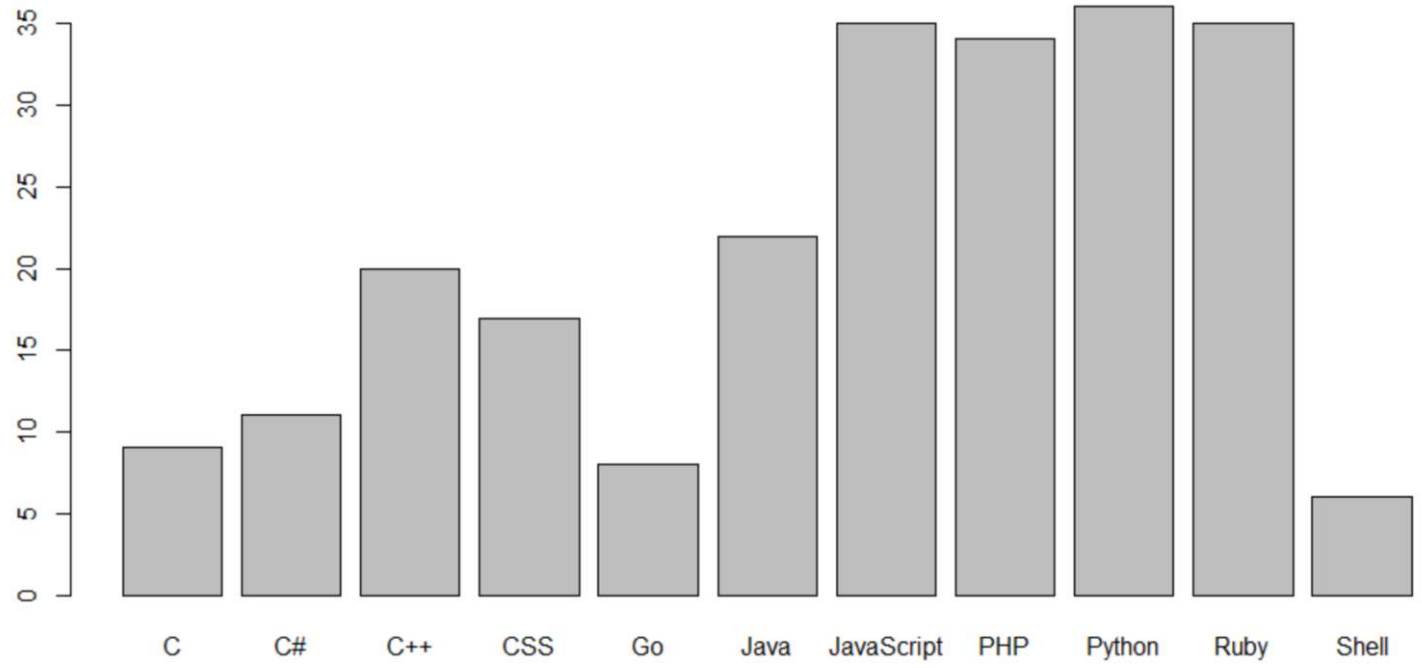
Filter missing

```
repositoryDataFrame <- repositoryDataFrame[
  as.character(repositoryDataFrame$language) != as.character(repositoryDataFrame$id),]

languages <- table(repositoryDataFrame$language)
head(languages)
```

```
> repositoryDataFrame <- repositoryDataFrame[
+ as.character(repositoryDataFrame$language) != as.character(repositoryDataFrame$id),]
> languages <- table(repositoryDataFrame$language)
> head(languages)

12745174 13381132 13723754 14098069  1652784 20825841
       0        0        0        0        0        0
```

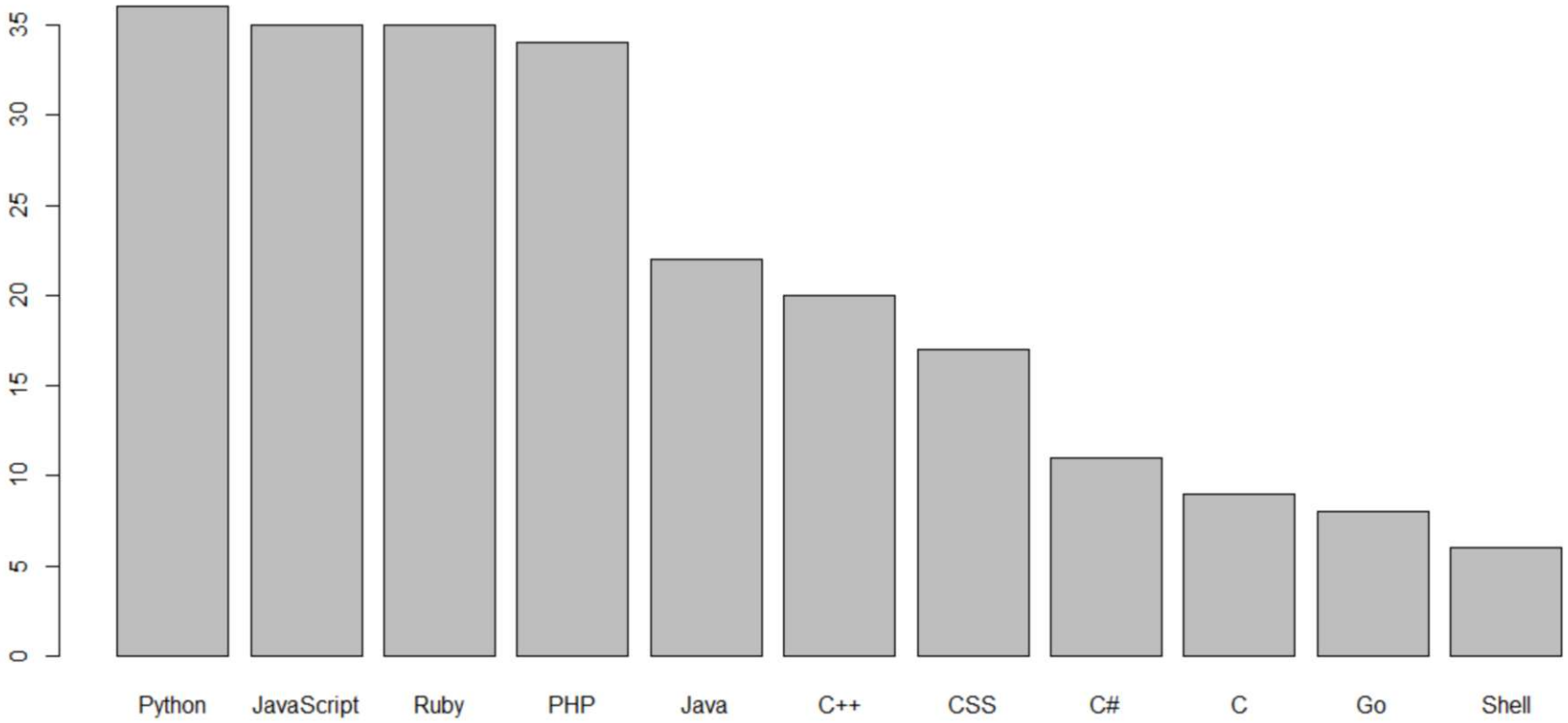
Plotting

```
languages <- languages[languages > 5]
barplot(languages)
```



Languages

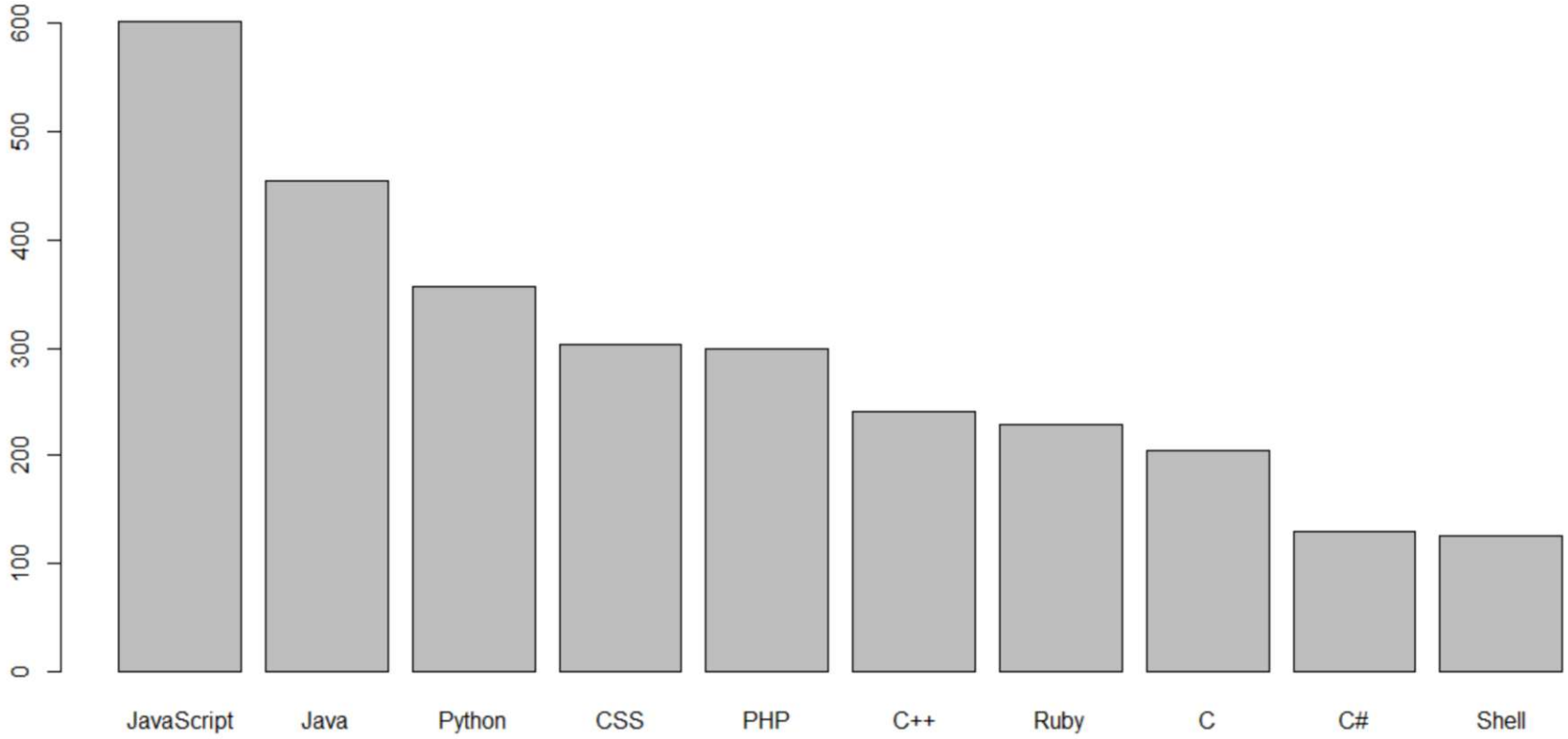
```
languages <- sort(languages, decreasing=TRUE)
languagesNames <- names(languages)
languagesNames

barplot(languages)
```



DEMO: Processing various data sources

- Active repositories – from Create, Push and PullRequest events
- Missing language information:
  - Google BigQuery
  - GitHub API

Expected result



Google BigQuery

New Query

```
1 SELECT repository_name, repository_url, repository_language
2 FROM [githubarchive:github.timeline]
3 WHERE (type = 'PushEvent'
4   or type = 'CreateEvent'
5   or type = 'PullRequestEvent')
6     and PARSE_UTC_USEC(created_at) >= PARSE_UTC_USEC('2015-01-01 15:00:00')
7     and PARSE_UTC_USEC(created_at) < PARSE_UTC_USEC('2015-01-01 16:00:00');
```

RUN QUERY | Save Query | Save View | Show Options

GitHub API from R

```
library("httr")

repositoryLanguageAPIInfo <- GET("https://api.github.com/repos/leethomason/tinyxml2/languages")
repositoryLanguageAPIInfo$content
languageContent <- content(repositoryLanguageAPIInfo)
languageContent
names(languageContent)
```

```
> repositoryLanguageAPIInfo <- GET("https://api.github.com/repos/leethomason/tinyxml2/languages")
> repositoryLanguageAPIInfo$content
 [1] 7b 0a 20 20 22 43 2b 2b 22 3a 20 31 37 35 39 30 35 2c 0a 20 20 22 50 79 74 68 6f 6e 22 3a 20 33
[33] 31 30 32 2c 0a 20 20 22 43 4d 61 6b 65 22 3a 20 32 39 33 35 2c 0a 20 20 22 4d 61 6b 65 66 69 6c
[65] 65 22 3a 20 31 31 35 0a 7d 0a
> languageContent <- content(repositoryLanguageAPIInfo)
> languageContent
$`C++`
[1] 175905

$Python
[1] 3102

$CMake
[1] 2935

$Makefile
[1] 115

> names(languageContent)
[1] "C++"      "Python"   "CMake"    "Makefile"
```

GitHub API Search

```
nextPage <- GET("https://api.github.com/search/repositories?q=created%3A2015-01-01&page=2")
nextPageContent = content(nextPage)

nextPageContent$items[[5]]$language
nextPageContent$items[[5]]$url
nextPageContent$items[[5]]$stargazers_count
```

```
> nextPage <- GET("https://api.github.com/search/repositories?q=created%3A2015-01-01&page=2")
> nextPageContent = content(nextPage)
> nextPageContent$items[[5]]$language
[1] "Python"
> nextPageContent$items[[5]]$url
[1] "https://api.github.com/repos/visualfabriq/bquery"
> nextPageContent$items[[5]]$stargazers_count
[1] 20
```

Combining data

```
urls$language <- apply(urls, 1, function(row){
  language = googleData[as.character(googleData$url) == as.character(row[2]), ]$repository_language

  as.character(language[1])
})

head(urls)
```

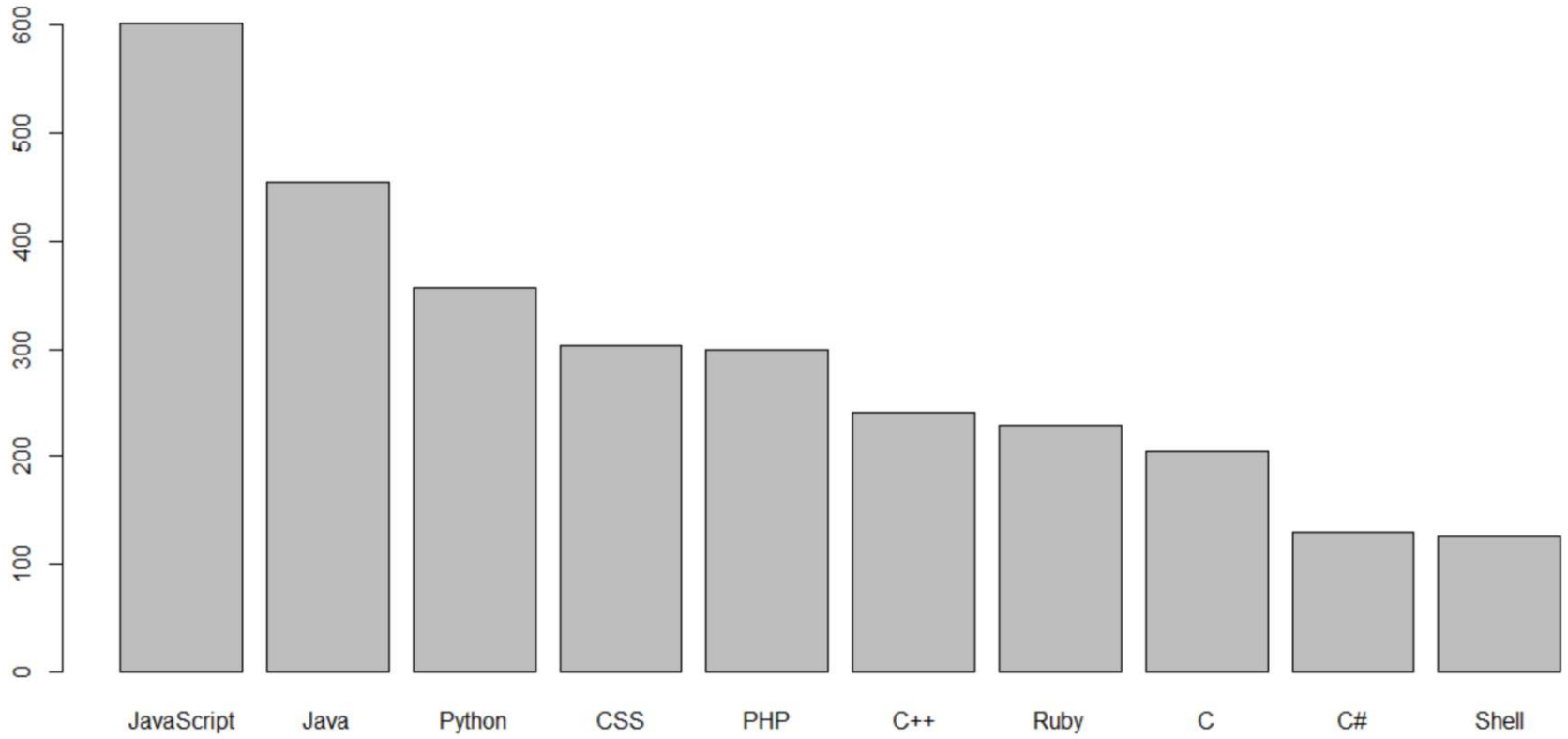
```
> urls$language <- apply(urls, 1, function(row){
+ language = googleData[as.character(googleData$url) == as.character(row[2]), ]$repository_language
+ as.character(language[1])
+ })
> head(urls)
        id                          url   language
1 28688495               petroav/6.828          C
2 28671719            rspt/rspt-theme        CSS
3 28270952 izuzero/xe-module-ajaxboard JavaScript
4 28593843           winterbe/streamjs JavaScript
5 27826205      hermanwahyudi/selenium     Python
6 28682546       jdilt/jdilt.github.io JavaScript
>
```

Sorted data

```
languages <- table(urls$language)
languages <- sort(languages, decreasing=TRUE)
barplot(languages)
```

Languages

```
languages <- table(urls[as.character(urls$language) != "null",]$language)
languages <- sort(languages[languages > 50], decreasing=TRUE)
barplot(languages)
```



Big Data in R

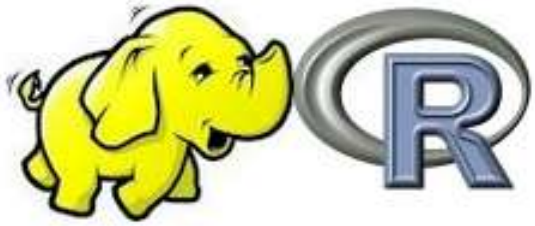
- What's _Big_Data_ anyway?
- R processes data in memory

- Bring down only the data you need
- Streaming the data from database

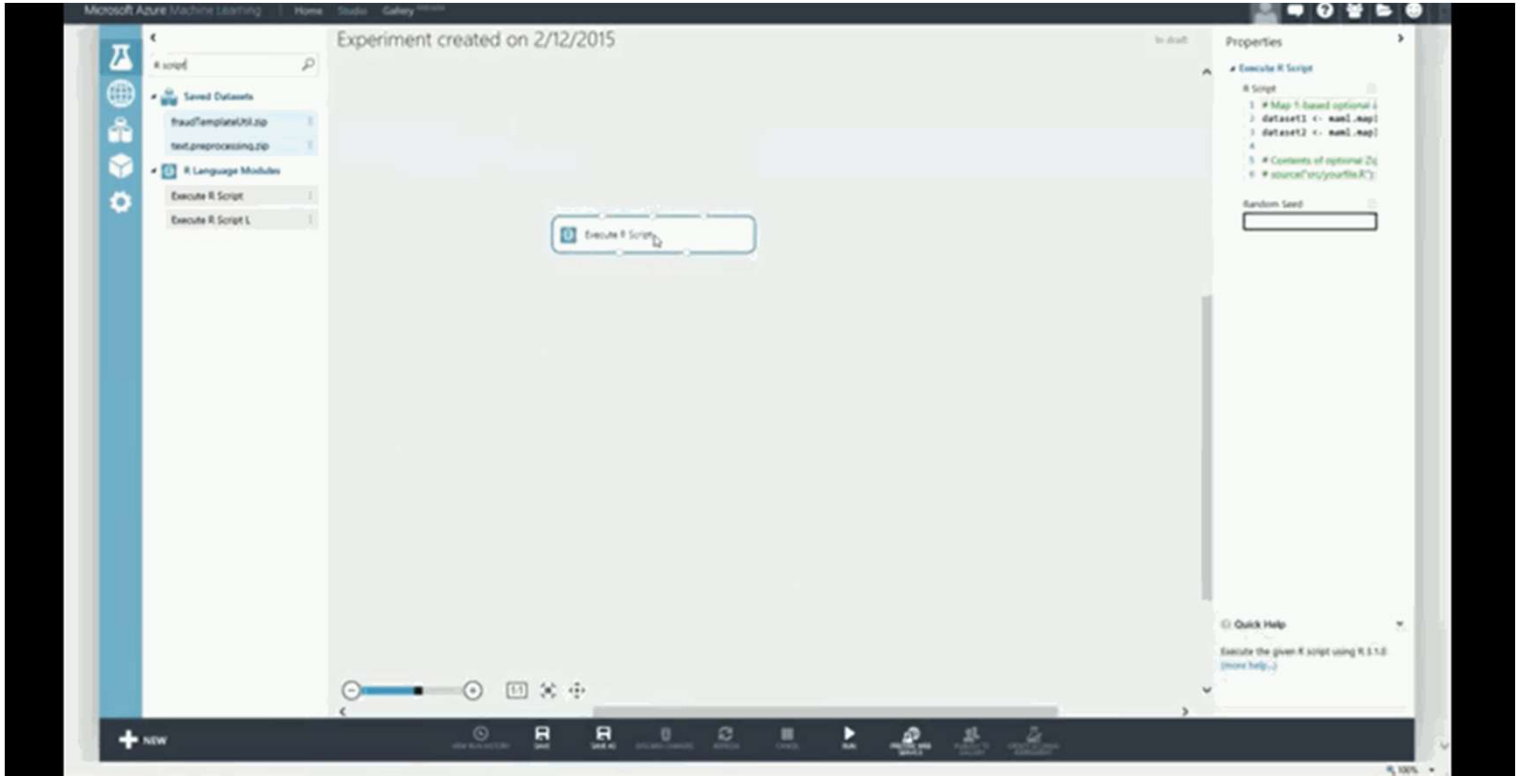
R, Hadoop, and How They Work Together

- **Hadoop Streaming** – utilities available as R scripts
- **ORCH** (**Oracle R Connector for Hadoop**) - provides access to a Hadoop cluster from R
- **RHIPE** (**R and Hadoop Integrated Programming Environment**) – techniques designed for analysing large sets of data
- **Rhadoop** - R packages that allow users to manage and analyse data with Hadoop

https://blog.udemy.com/r-hadoop/



Azure Machine Learning Studio

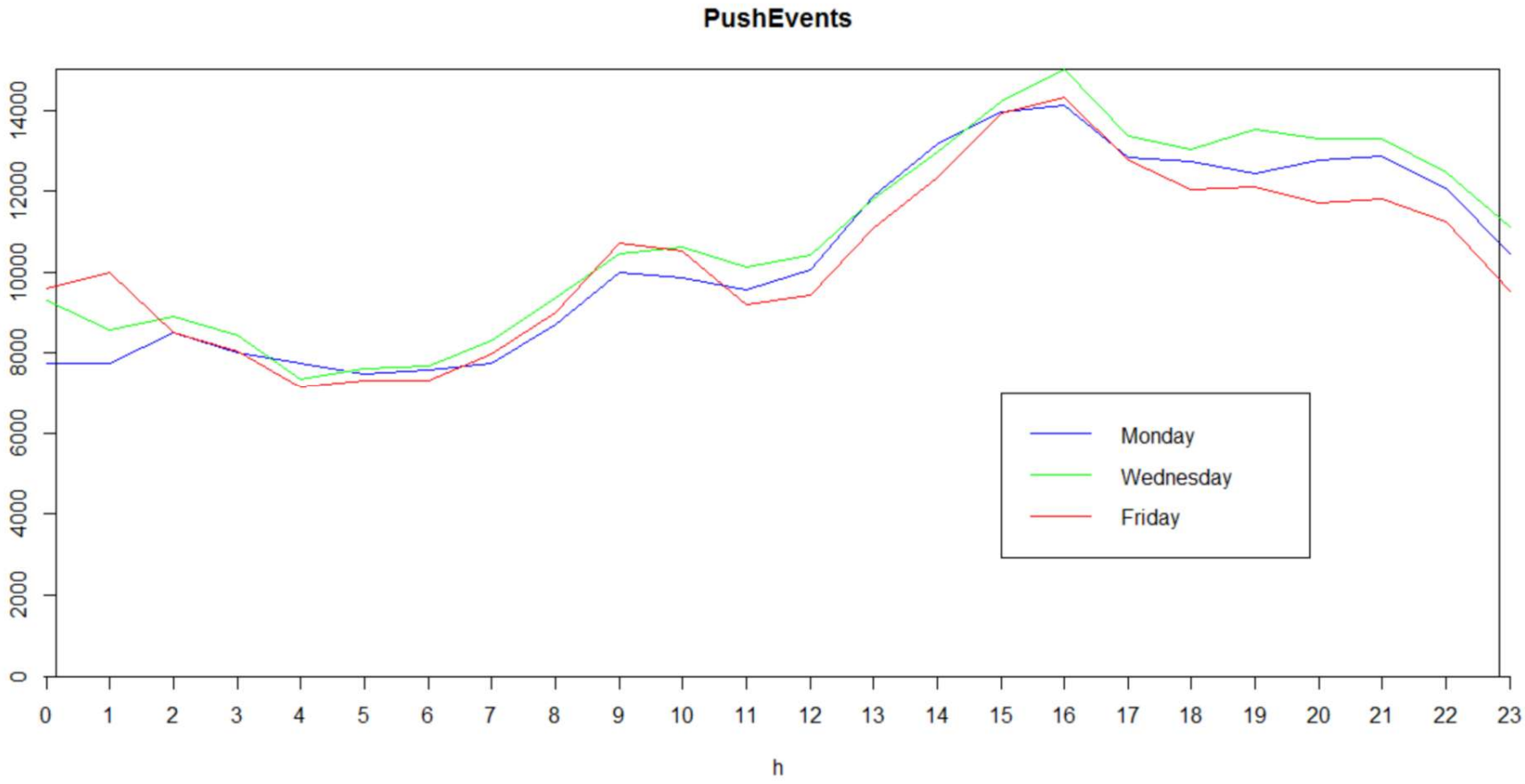


http://channel9.msdn.com/Blogs/Windows-Azure/R-in-Azure-ML-Studio

DEMO: Processing multiple files

- Reading files line by line
- Reading multiple files
- Saving files

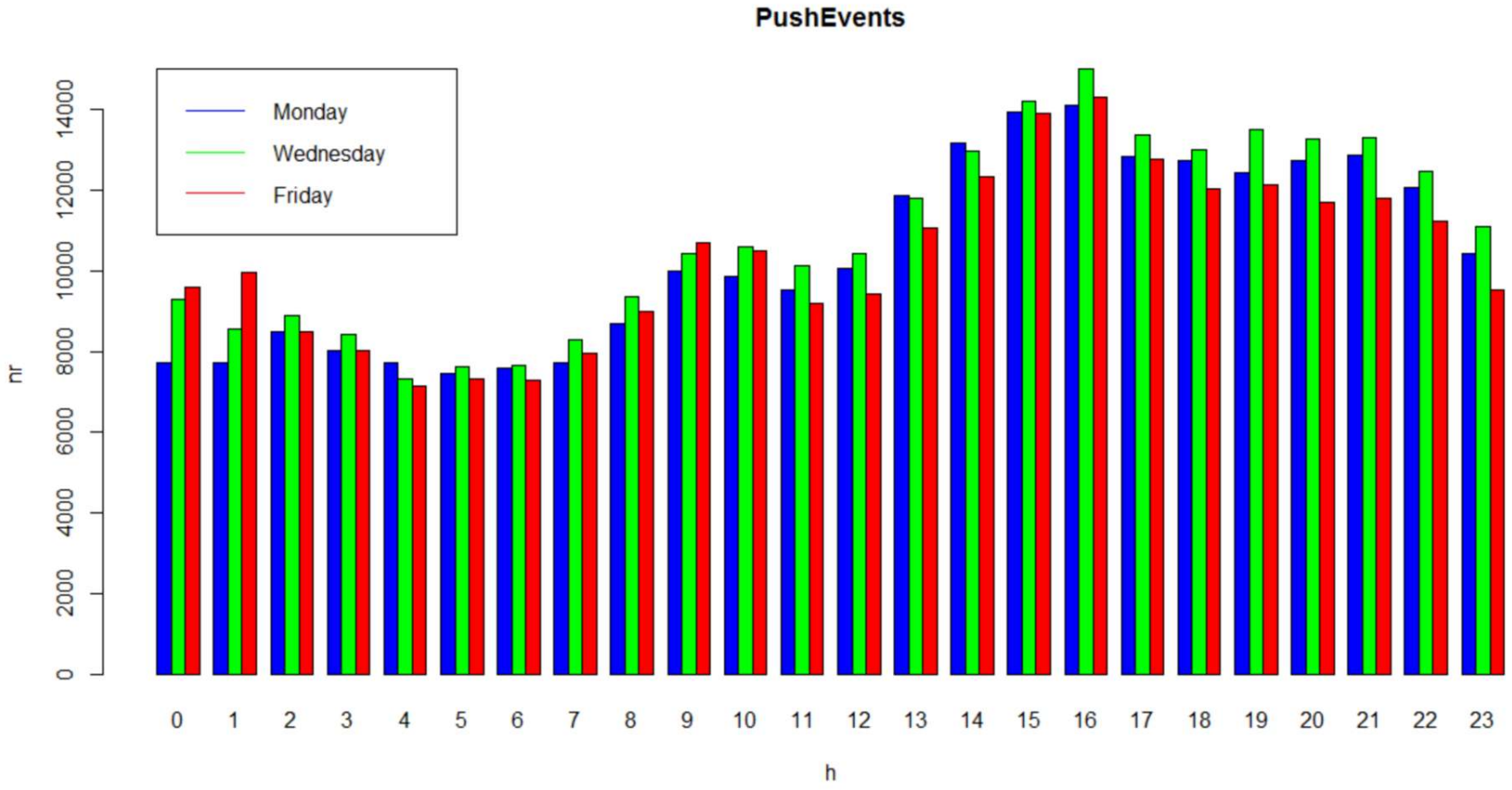
Analysing week



Read multiple files line by line

```
readEventsLineByLine <- function(file, eventNames) {
  eventCount <- 0
  while(length(line <- readLines(file, n = 1, warn = FALSE)) > 0) {
    json <- fromJSON(line)
    if (json$type %in% eventNames) {
      eventCount <- eventCount + 1
    }
  }
  return(eventCount)
}

readDayFolder = function(folderName) {
  pushes = integer()
  for(n in 0:23) {
    fileName = paste(folderName, n, ".json", sep="")
    stream <- file(fileName, open="r")
    pushesPerHour = readEventsLineByLine(stream, list("PushEvent"))
    close(stream)
    pushes = c(pushes, pushesPerHour)
  }

  return(pushes)
}

week = data.frame(0:23)
week$Monday = readDayFolder(mondayFolder)
week$Wednesday = readDayFolder(wednesdayFolder)
week$Friday = readDayFolder(fridayFolder)

write.csv(week, file="Data/weekData.csv")

weekData <- read.csv(file=weekDataFile, sep=",",head=TRUE)
```

Push Events during the week



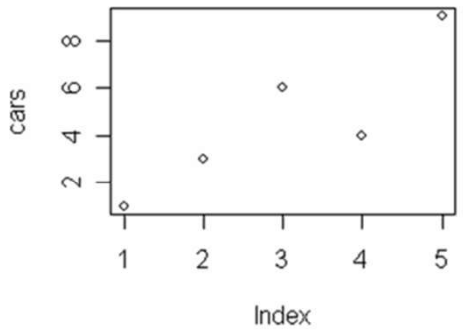
DEMO: Plotting in R

1. Line Charts
2. Bar Charts
3. Histograms
4. Pie Charts
5. Dotcharts
6. Misc

Line Charts

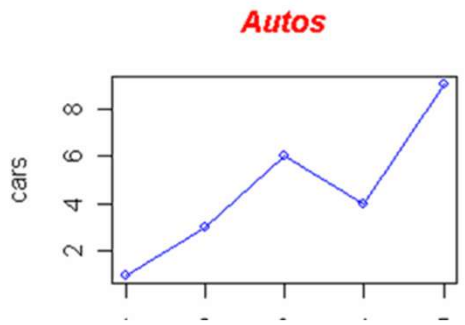
First we'll produce a very simple graph using the values in the car vector:

```
# Define the cars vector with 5 values
cars <- c(1, 3, 6, 4, 9)

# Graph the cars vector with all defaults
plot(cars)
```



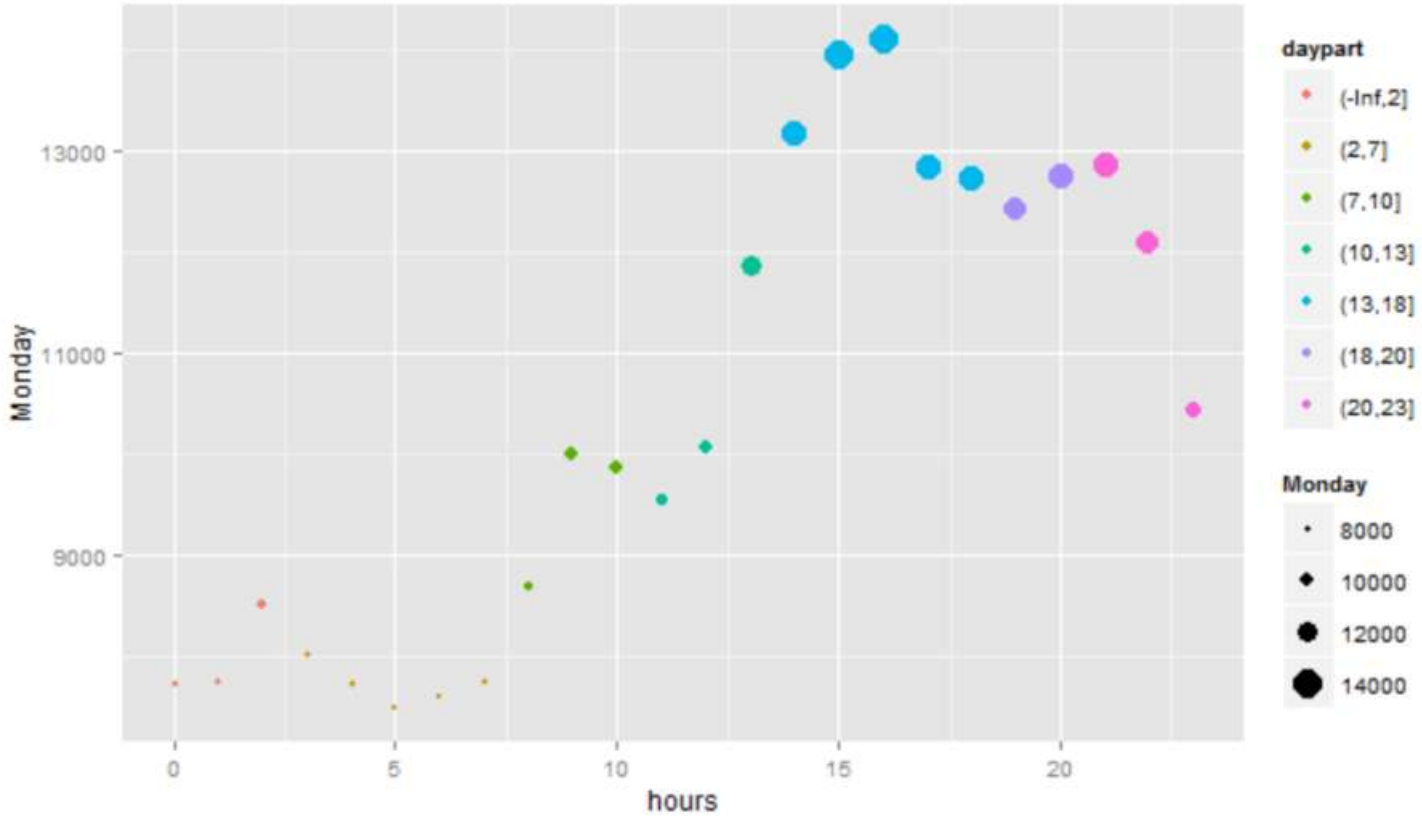
Let's add a title, a line to connect the points, and some color:

```
# Define the cars vector with 5 values
cars <- c(1, 3, 6, 4, 9)

# Graph cars using blue points overlayed by a line
plot(cars, type="o", col="blue")

# Create a title with a red, bold/italic font
title(main="Autos", col.main="red", font.main=4)
```

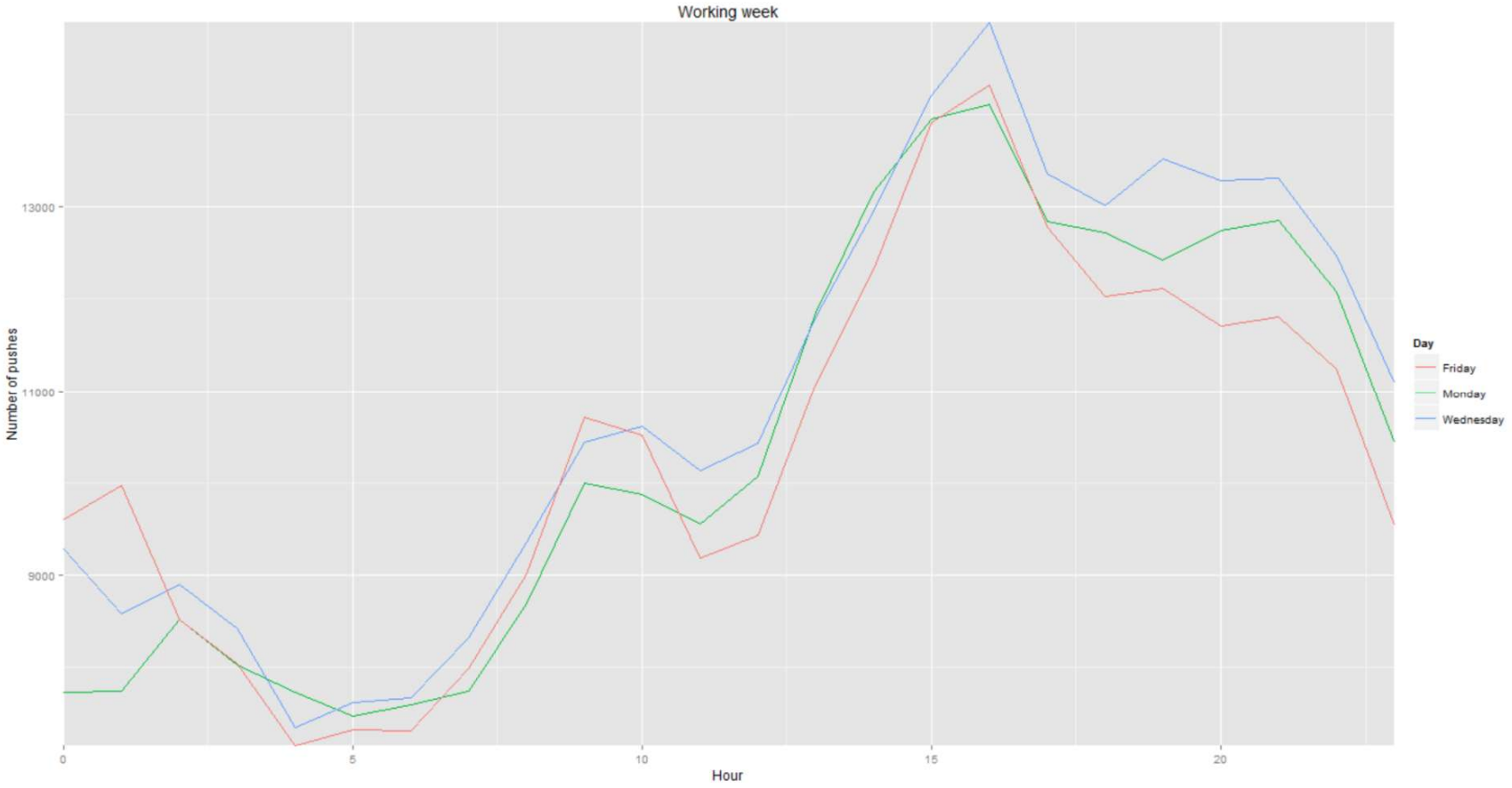


http://www.harding.edu/fmccown/r/

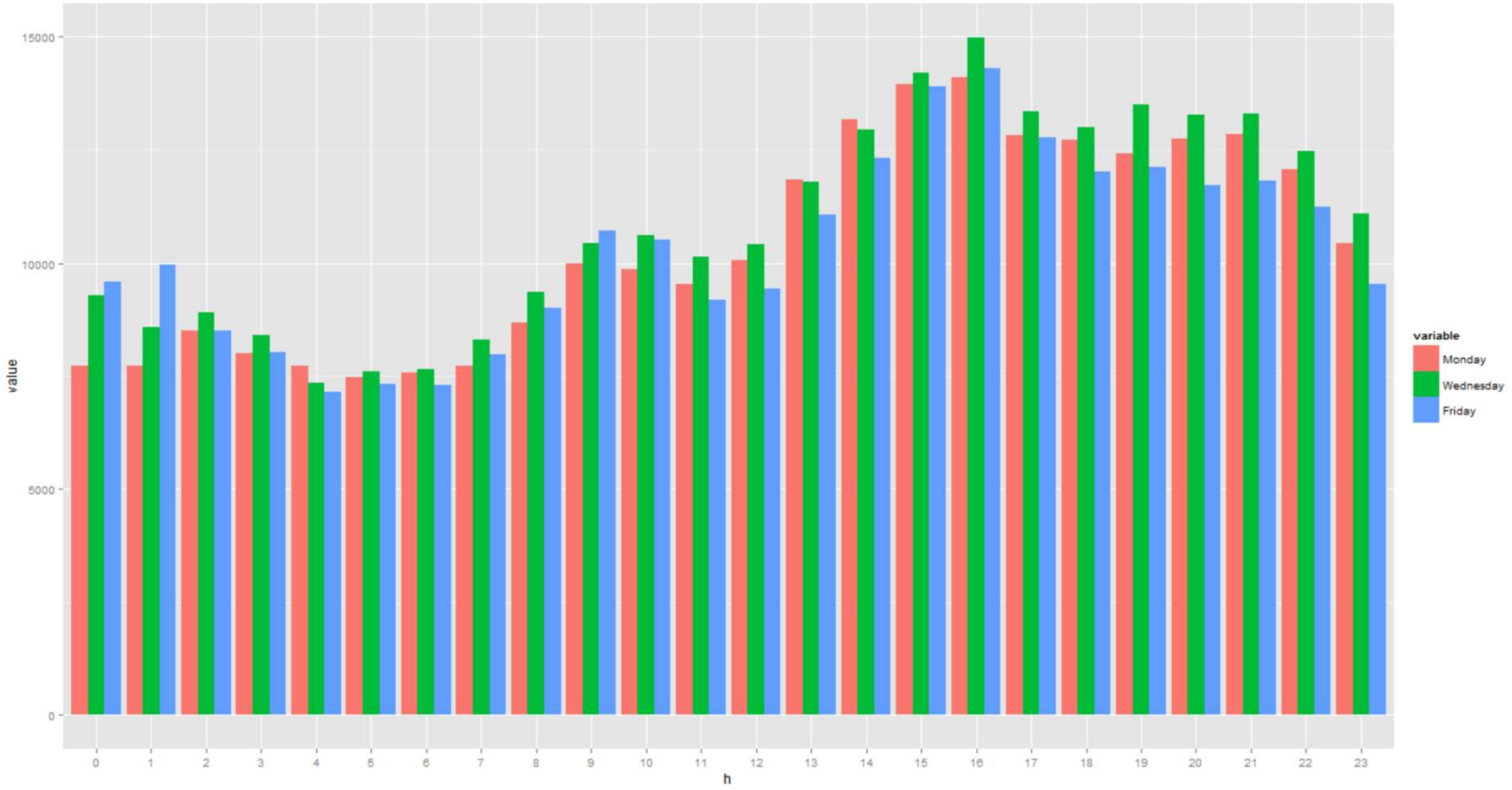
Qplot - dots


Monday
13000
11000
9000
0
5
10
15
20
hours
daypart
(-Inf,2]
(2,7]
(7,10]
(10,13]
(13,18]
(18,20]
(20,23]
Monday
8000
10000
12000
14000

qqplot - lines



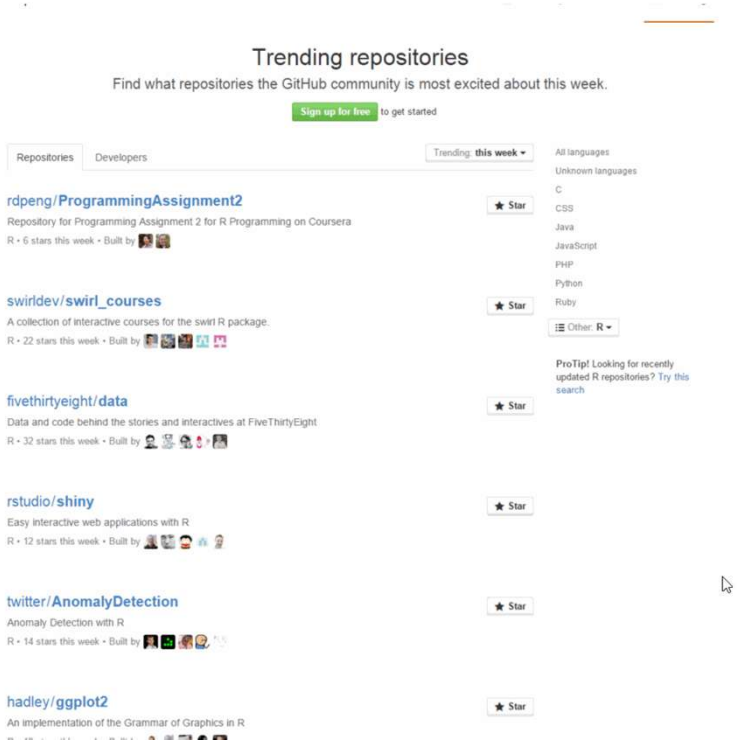
qqplot - bars


15000
10000
5000
0
value
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
h
variable
Monday
Wednesday
Friday

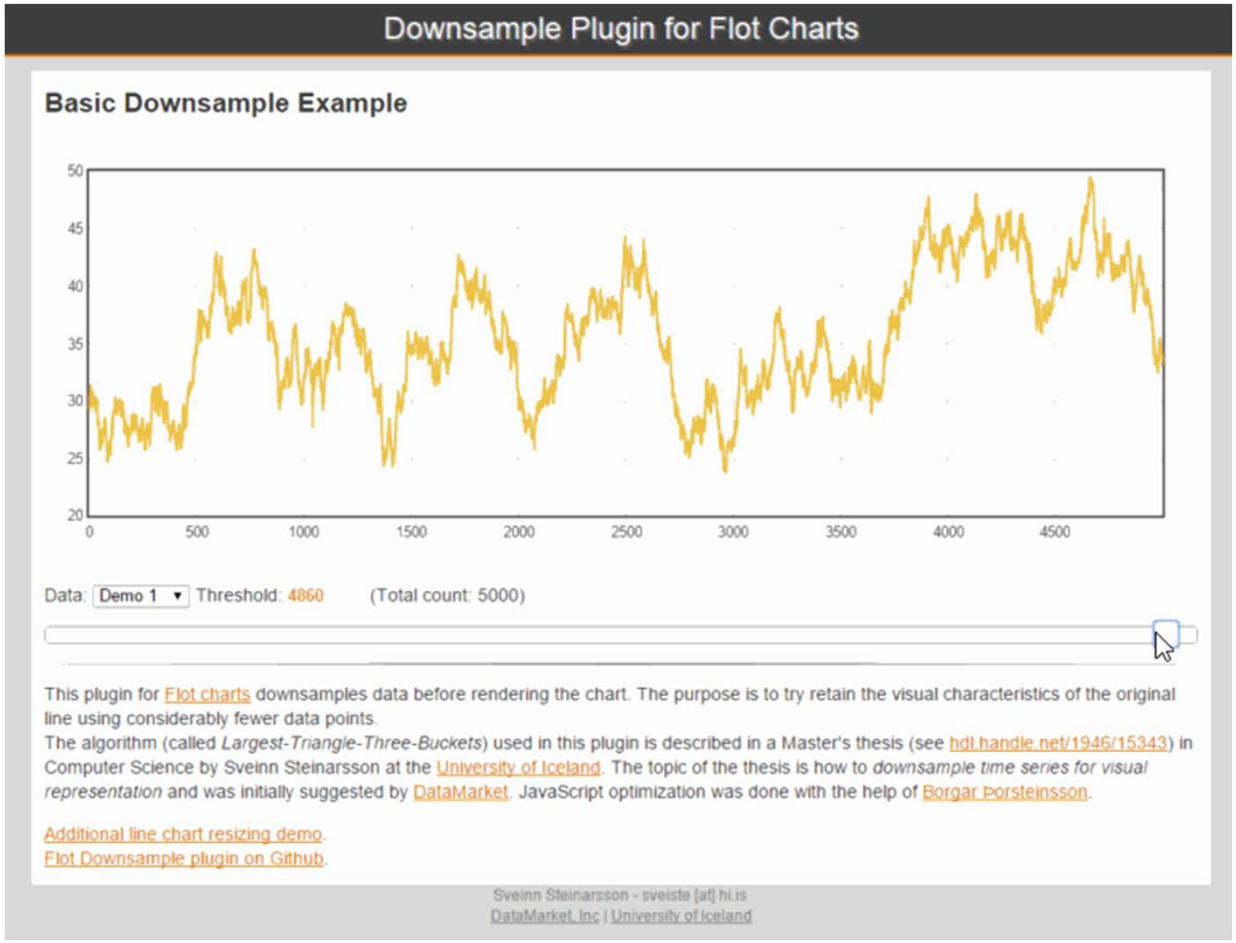
Libraries & Tools

- DataScience

- JSON processing
- Http requests
- Summarising
- Plotting

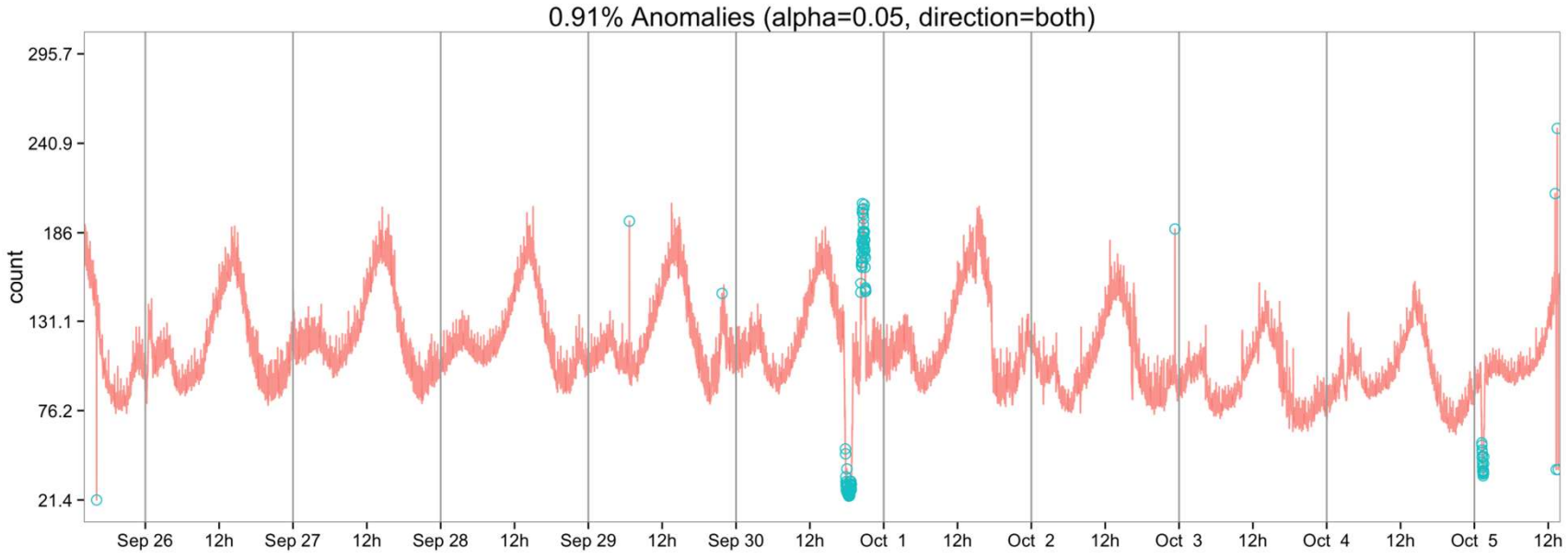


DEMO: Downsampling



https://github.com/javiljoen/LTTB

DEMO: Anomalies detection



https://github.com/twitter/AnomalyDetection

To summarize…

- GitHub Archive, GitHub API

- R language – RStudio, types, data exploration, I/O operations, etc.
- Big Data in R
- 3rd party & build-in libraries
- Visualization

Thank you

Barbara Fusinska

@BasiaFusinska

barbarafusinska.com

https://github.com/BasiaFusinska/RTalk