

# Big data analysis with RHadoop

assoc. prof. Janez Povh

University of Ljubljana, Faculty of mechanical engineering

Funded by PRACE-6IP (GA: 823767)



Ljubljana, 16. 9. 2020

# Goals of this session

- 1 R, Hadoop, RHadoop
- 2 RStudio
- 3 Basic data management with R
- 4 Basic (parallel) data management with RHadoop



# What is R

- 1 Software for Statistical Data Analysis
- 2 Based on S
- 3 Programming Environment
- 4 Interpreted Language
- 5 Data Storage, Analysis, Graphing
- 6 Free and Open Source Software



# How to obtain R

- 1 R current version 4.0.2 (released on 2020-06-22).
- 2 <http://cran.r-project.org>
- 3 Binary source codes
- 4 Windows executables



## Pros:

- 1 Free and Open Source
- 2 Strong User Community
- 3 Highly extensible, flexible
- 4 Implementation of high end statistical methods
- 5 Flexible graphics and intelligent defaults

## Cons

- 1 Steep learning curve
- 2 Slow for large datasets



- 1 R Supports virtually any type of data
- 2 Numbers, characters, logicals (TRUE/ FALSE)
- 3 Arrays of virtually unlimited sizes
- 4 Simplest: Vectors and Matrices
- 5 Lists: Can Contain mixed type variables
- 6 Data Frame: Rectangular Data Set



## Linear

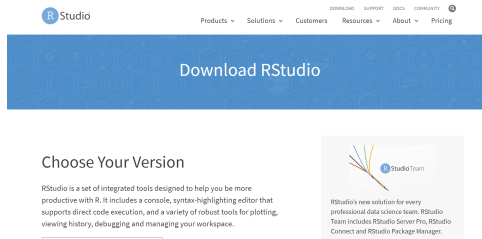
- 1 vectors (all same type)
- 2 lists (mixed types)

## Rectangular

- 1 data frame
- 2 matrix



I recommend RStudio.



The screenshot shows the RStudio website's download page. At the top, there is a navigation menu with links for 'Products', 'Solutions', 'Customers', 'Resources', 'About', and 'Pricing'. A large blue banner in the center contains the text 'Download RStudio'. Below this, the section 'Choose Your Version' is visible, with a sub-header 'RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.' To the right of this text is a small image of the RStudio logo and a brief description of the RStudio Team's solution.

Slika: <https://rstudio.com/products/rstudio/download/>



Ni vamo | viz.hpc.fs.uni-lj.si/rstudio/auth-sign-in

Sign in to RStudio

Username:  
campus04

Password:  
\*\*\*\*\*

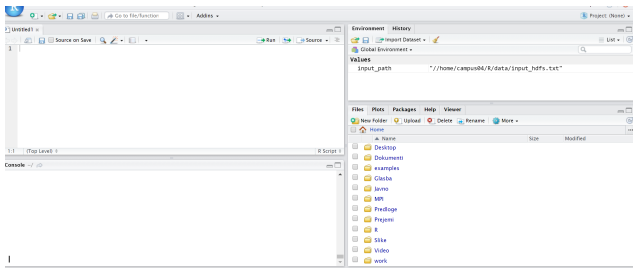
Stay signed in

Sign In

Slika: <http://viz.hpc.fs.uni-lj.si/rstudio/auth-sign-in>



# RStudio on HPCFS



# The first R script file

- 1 Open new script file **CTRL+SHIFT+N**
- 2 **Save** the script file.

## Create directory for R scripts

```
myDir=paste("/home", Sys.getenv("USER"), 'myRscripts', sep="/")
unlink(myDir, recursive = TRUE)
dir.create(myDir)
setwd(myDir)
getwd()
dir()
```



# Creating the first scrip file

## Create and save simple data file

```
N=1000;
#create data frame with three variables: group, ints, reals
Data=data.frame(group=character(N),ints=numeric(N),reals=numeric(N))
Data$group=sample(c("a","b","c"), 1000, replace=TRUE);
Data$ints=rbinom(N,10,0.5);
Data$reals=rnorm(N);
head(Data)

#write data frame as txt table
write.table(Data, file='Data_Ex_1.txt', append = FALSE, dec = ".",col.names = TRUE)

ls()
rm(list = ls())
```



# Load and analyse the data

## Load data

```
#load data from local file
myDir=paste("/home", Sys.getenv("USER"),'myRscripts', sep="/")
setwd(myDir)

Data_read<- read.table(file='Data_Ex_1.txt',header = TRUE)
# first few rows
head(Data_read)

#10 th row
Data_read[10,]

# column group
Data_read$group
Data_read[,1]
```



# Load and analyse the data

## Load data

```
# compute means and counts by groups
group| count_ints mean_ints
a    | 341 | 4.964809
b    | 335 | 4.943284
c    | 324 | 5.104938

# primitive solution
Group_lev=levels(Data_read$group)

Tab_summary=data.frame(group=character(3),count_ints=integer(3),mean_ints=numeric(3))
Tab_summary$group<-Group_lev
for (i in c(1:3)){
  sub_data = subset(Data_read,group==Group_lev[i])
  Tab_summary$count_ints[i]<-nrow(sub_data)
  Tab_summary$mean_ints[i]<-mean(sub_data$ints)
}
```

# Analyse the data with dplyr, magrittr

- 1 Library dplyr: "select", "filter", "arrange", "mutate" and "summarize".
- 2 Library magrittr: "%>%"

## dplyr

```
library(dplyr)
Tab_summary1<-group_by(Data_read,group) %>%
summarise(count_ints=n(),mean_ints=mean(ints))

#select two columns
Data_read_group_ints<-Data_read %>% select(group,ints)

#add new variable reals/ints
# Data_read<-mutate(Data_read,ratio=reals/ints)
Data_read<-Data_read %>% mutate(ratio=reals/ints)

#arrange
#sort accordind to increasing group
Data_read<-Data_read %>% arrange(desc(group))
Data_read<-Data_read %>% arrange(group)
```

# The goals of the second part

- 1 Demonstrating **basic** data management operations with RHadoop;
- 2 By few examples **showing** basic data analysis with RHadoop;

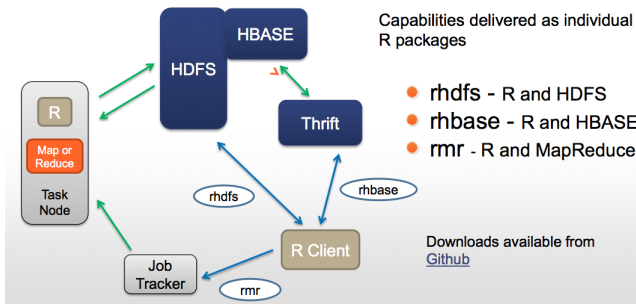


# Motivation

- 1 Do data analysis (statistics), do not bother with low level settings
- 2 Stay within R (and RStudio)



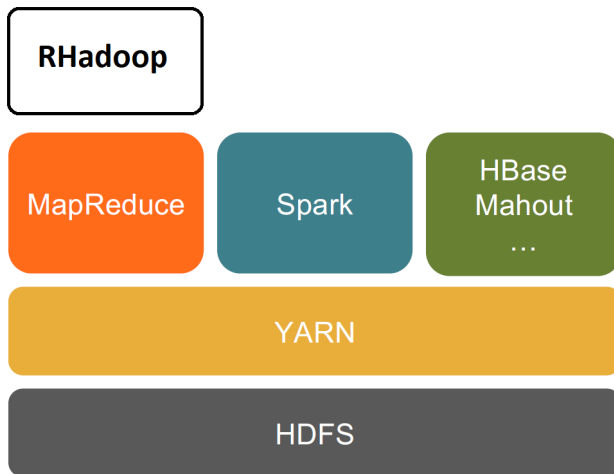
# Overall picture



Slika: <https://www.r-bloggers.com/slides-and-replay-from-r-and-hadoop-webinar/>



# Overall picture



# First little example

content...



# Setting up RHadoop using terminal window

```
export LD_LIBRARY_PATH=/opt/apps/software/Java/1.7.0_80/lib:${LD_LIBRARY_PATH}
export PATH=/opt/apps/software/Java/1.7.0_80:${PATH}
export JAVA_HOME=/opt/apps/software/Java/1.7.0_80
export PATH=/opt/apps/software/Hadoop/2.6.0-cdh5.8.0-native/bin:${PATH}
export PATH=/opt/apps/software/Hadoop/2.6.0-cdh5.8.0-native/sbin:${PATH}
export LD_LIBRARY_PATH=/opt/apps/software/Hadoop/2.6.0-cdh5.8.0-native/lib:${LD_LIBRARY_PATH}
export HADOOP_HOME=/opt/apps/software/Hadoop/2.6.0-cdh5.8.0-native/share/hadoop/mapreduce
```



5 R packages provided by **RevolutionAnalytics**<sup>12</sup>:

- **rhdfs** - basic connectivity to the Hadoop Distributed File System (browse, read, write, and modify files stored in HDFS)
- **rhbase** - basic connectivity to the HBASE distributed database, using the Thrift server.
- **plyrmr** - enables the R user to perform common data manipulation operations, as found in plyr and reshape2
- **rmr2** - allows R developer to perform statistical analysis in R via Hadoop MapReduce functionality on a Hadoop cluster.
- **ravro** - adds the ability to read, write and manipulate avro files from local and HDFS file system.

---

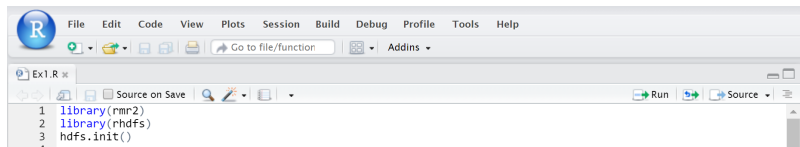
<sup>1</sup><https://github.com/RevolutionAnalytics>

<sup>2</sup><https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads>



## Setting up the **Rhadoop** - cnt.

- 1 Establish the connectivity to the Hadoop Distributed File System by loading the library `rhdfs`. `library(rhdfs)`
- 2 Load libraries to work with Hadoop MapReduce `library(rmr2)`
- 3 Initialize HDFS `hdfs.init()`.
- 4 All together:



The screenshot shows the RStudio interface with a script editor containing the following R code:

```
1 library(rmr2)
2 library(rhdfs)
3 hdfs.init()
4
```



# Basic data operations with RHadoop.

List files in the root directory of DFS `hdfs.ls("/")`

```
> hdfs.ls("/")
permission owner      group      size      modtime      file
1 -rw-r--r-- hadoop     hadoop 184814018 2018-08-09 18:11 /BigData_reg_class
2 -rw-r--r-- hadoop     hadoop 33602002 2018-08-09 18:11 /CEnetBig
3 -rw-r--r-- hadoop     hadoop 1366 2018-08-09 18:11 /README.txt
4 drwxr-xr-x hadoop     hadoop 0 2018-08-09 18:11 /public
5 drwxr-xr-x hadoop supergroup 0 2019-03-08 11:15 /system
6 drwxr-xr-x hadoop     hadoop 0 2018-08-09 18:11 /test
7 drwxrwxrwx hadoop     hadoop 0 2019-09-17 23:25 /tmp
8 drwxr-xr-x hadoop     hadoop 0 2019-09-13 10:34 /user
```



# Basic data operations with RHadoop.

List files in the home directory of each user

```
hdfs.ls("/user/campus04")
```

```
hdfs.ls("/user/campus04")
permission  owner group      size      modtime      file
1 -rw-r--r-- campus04 hadoop    12466 2020-09-16 06:47 /user/campus04/OurSmallData
2 -rw-r--r-- campus04 hadoop 18836041094 2020-09-11 09:16 /user/campus04/safecast.csv
3 -rw-r--r-- campus04 hadoop 336031560 2020-09-15 15:30 /user/campus04/wiki321MB
4 drwxr-xr-x campus04 hadoop      0 2020-09-15 15:30 /user/campus04/wordcount_out
```



# Moving data around - FileZilla

The screenshot shows the FileZilla client interface. At the top, the title bar reads "sftp://campus04@forge.fs.uni-lj.si - FileZilla". Below the title bar is a menu bar with "File", "Edit", "View", "Transfer", "Server", "Bookmarks", and "Help". A toolbar contains various icons for file operations. Below the toolbar, the connection fields are: Host: sftp://forge.fs.uni-..., Username: smpus04, Password: masked with dots, and Port: empty. A "Quickconnect" button is to the right. The status area below shows a series of messages: "Connecting to forge.fs.uni-lj.si...", "Connected to forge.fs.uni-lj.si", "Retrieving directory listing...", "Listing directory /home/campus04", "Directory listing of '/home/campus04\*' successful", "Retrieving directory listing of '/home/campus04/R'...", "Listing directory /home/campus04/R", and "Directory listing of '/home/campus04/R\*' successful".

Local site: C:\Users\jpovh\Documents\RESEARCH\Matlab\jpcode\cases\

- cases
  - BiqBin
  - bw
  - Cedric Josz
  - cut
  - fab
  - FMF\_izPogOptim

Remote site: /home/campus04/R

- Glasba
- Javno
- MPI
- Predloge
- Prejemi
- R
- Sluke



## Copy from other account

```
cp /home/campus04/R/data/iris.csv /home/campusxx/R/data/iris.csv
```

## Copy from internet

```
curl -o /home/campus04/R/data/iris.csv  
https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/  
639388c2cbc2120a14dcf466e85730eb8be498bb/iris.csv
```



# Moving data around with Linux or RHadoop

## Copy from internet address or local folder to hdfs within RHadoop

```
curl https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/639388c2cbc2120a14dcf466e85730eb8be498bb/iris.csv  
hadoop fs -appendToFile - /user/campus04/iris.csv
```

```
system('curl_https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/639388c2cbc2120a14dcf466e85730eb8be498bb/iris.csv')  
hadoop_fs_-appendToFile_-_/user/campus04/iris.csv')
```

```
system('curl_file:///home/campus04/R/data/iris.csv|_hadoop_fs_-appendToFile_-_/user/campus04/iris.csv')
```

```
system('hadoop_fs_-appendToFile_/_home/campus04/R/data/iris.csv/_user/campus04/iris.csv')
```



# Create and store data in HDFS

Use small data created at the beginning and stored as

```
file_name = paste("/home", Sys.getenv("USER"), 'myRscripts', 'Data_Ex-1.txt', sep="/")
Data_read<-read.table(file=file_name,header = TRUE)

myDFS_File=paste("/user", Sys.getenv("USER"), "OurSmallData", sep="/")
hdfs.rm(myDFS_File)
OurSmallData=to.dfs(Data_read, myDFS_File,format="native")
SmallData1_DFS=from.dfs(OurSmallData)
system("hdfs_fsck_/user/campus04/OurSmallData")
```



CEnetBig contains data about customers of company X: for each customer we have one row containing

- ID of the customer;
- the values of their bills for period January 2016-December 2016;
- type of product that they have;

```
id 2016_1 2016_2 2016_3 2016_4 2016_5 2016_6 2016_7 2016_8 2016_9 2016_10 2016_11 2016_12 type
[1,] 100373 137.66 141.57 128.83 133.00 97.39 116.62 123.97 156.83 90.50 98.62 118.61 152.34 4
[2,] 100194 98.32 119.40 120.30 105.67 90.26 80.13 80.62 108.63 104.30 123.31 101.93 140.85 2
[3,] 100565 127.60 133.79 90.15 62.33 87.96 92.20 72.04 113.69 65.95 82.69 85.72 121.81 2
```



# HDFS statistics for CEnetBig

- 1 From RStudio `system("hdfs fsck /CEnetBig")`
- 2 From command line: `hadoop fsck /CEnetBig`

```
> system("hdfs_fsck_/CEnetBig")
Connecting to namenode via http://viz.hpc:50070
FSCK started by campus04 (auth:SIMPLE) from /10.0.2.99 for path /CEnetBig at Wed Sep 16 07:35:36 CEST 2020
.Status: HEALTHY
Total size:      33602002 B
Total dirs:      0
Total files:     1
Total symlinks:  0
Total blocks (validated):      1 (avg. block size 33602002 B)
Minimally replicated blocks:  1 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks:      0 (0.0 %)
Mis-replicated blocks:        0 (0.0 %)
Default replication factor:    3
Average block replication:     5.0
Corrupt blocks:                0
Missing replicas:              0 (0.0 %)
Number of data-nodes:          16
Number of racks:               8
FSCK ended at Wed Sep 16 07:35:36 CEST 2020 in 1 milliseconds

The filesystem under path '/CEnetBig' is HEALTHY
```



# HDFS statistics for CEnetBig

## 1 From RStudio `system("hdfs fsck /user/jpovh/safecast.csv")`

```
Connecting to namenode via http://viz.hpc:50070
FSCK started by campus04 (auth:SIMPLE) from /10.0.2.99 for path /user/campus04/safecast.csv at Wed Sep 16 07
.Status: HEALTHY
Total size:      18836041094 B
Total dirs:      0
Total files:     1
Total symlinks:  0
Total blocks (validated):      141 (avg. block size 133588943 B)
Minimally replicated blocks:  141 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks:  0 (0.0 %)
Mis-replicated blocks:  0 (0.0 %)
Default replication factor:    3
Average block replication:     3.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 16
Number of racks: 8
FSCK ended at Wed Sep 16 07:39:21 CEST 2020 in 10 milliseconds
```

The filesystem under path '/user/campus04/safecast.csv' is HEALTHY



- 1 Load data into active memory:

```
CEnetBig<-from.dfs("/CEnetBig")
```

- 2 CEnetBig is a key-value pair with void key.

```
> CEnetBig$key
NULL
> CEnetBig$val[1:3,]
id 2016_1 2016_2 2016_3 2016_4 2016_5 2016_6 2016_7 2016_8 2016_9 2016_10 2016_11 2016_12 type
[1,] 100373 137.66 141.57 128.83 133.00 97.39 116.62 123.97 156.83 90.50 98.62 118.61 152.34 4
[2,] 100194 98.32 119.40 120.30 105.67 90.26 80.13 80.62 108.63 104.30 123.31 101.93 140.85 2
[3,] 100565 127.60 133.79 90.15 62.33 87.96 92.20 72.04 113.69 65.95 82.69 85.72 121.81 2
```



# First Big Data challenge

**Goal:** In the column 2016\_1 find the maximum value.

**Use:**  $\max\{\cup_i A_i\} = \max_i\{\max A_i\}$ .

$$\begin{aligned} 9 &= \max\{1, 5, 4, 7, 9, 2, 3, 5\} \\ &= \underbrace{\max\{1, 5, 4, 7\}, \max\{9, 2, 3, 5\}}_{\max} \end{aligned}$$

Suppose  $XX$  is submatrix of `CEnetBig` of 1st 100 rows. We find the maximum of column 2016\_1 by

```
XX=CEnetBig$val[1:100,]  
M=max(XX[, "2016_1"])
```



# Finding maximum by Map-Reduce

- **MAP:**

```
mapper = function (., X) {  
  M=max(X[, "2016_1"]);  
  keyval(1,M)  
}
```

- **REDUCE:**

```
reducer = function(k, A) {  
  keyval(k, list(Reduce("max", A))) # take maximum of maxima  
}
```



# Finding maximum by Map-Reduce - cnt.

- **MAP-REDUCE:**

```
GlobalMaxMR = from.dfs(  
  mapreduce(  
    input = "/CEnetBig",  
    map = mapper,  
    reduce = reducer  
  )  
)
```

- **Final code:**

```
GlobMax =GlobalMaxMR$val
```

- **Result**

```
> GlobalMaxMR$val  
[[1]]  
[1] 243.25
```



# Finding maximum, number of map calls and block sizes

```
mapper2 = function (., X) {
  M=max(X[, "2016-1"]);
  keyval(1:3,list(1,M,dim(X)[1]))
}

reducer2 = function(k, A) {
  if(k==1){
    keyval(k, list(Reduce("+", A))) # take maximum of maxima
  } else if (k==2) {
    keyval(k, list(Reduce("max", A))) # take maximum of maxima
  } else {
    keyval(k, A)
  }
}

GlobalMaxNumMR = from.dfs(
  mapreduce(
    input = "/CEnetBig",
    map = mapper2,
    reduce = reducer2
  )
)
```



# Finding maximum, number of map calls and block sizes - cnt.

```
> GlobalMaxNumMR
$key
[1] 1 2 3 3 3 3 3 3 3 3 3 3 3 3 3

> GlobalMaxNumMR$val
[[1]]
[1] 12
[[2]]
[1] 243.25
[[3]]
[1] 26797
[[4]]
[1] 89284
[[5]]
[1] 89285
[[6]]
[1] 89285
[[7]]
[1] 89285
...
[[13]]
[1] 89285
[[14]]
[1] 80356

> sum(unlist(GlobalMaxNumMR$val[3:14]))
[1] 1000000
```



## Second Big Data challenge

**Goal:** Compute the mean value of the column 2016\_1 ..

**Note:**  $\bar{x} = \sum_i X_i/n$

Suppose  $XX$  is submatrix of `CEnetBig` of 1st 100 rows. We find mean value of column 2016\_1 by

```
XX=CEnetBig$val[1:100,]  
m=mean(XX[, "2016_1"])
```

- If  $s_i$  and  $n_i$  are sums and sizes of blocks of data, respectively, then the mean value of all data is

$$\bar{x} = \frac{\sum_i s_i}{\sum_i n_i}$$



# Finding mean value by Map-Reduce

- **MAP:**

```
mapper_mean = function (., X) {  
  n=nrow(X);  
  mi=sum(X[,2]);  
  keyval(1:2,list(n,mi));  
}
```

- **REDUCE:**

```
reducer_mean = function(k, A) {  
  keyval(k,list(Reduce('+', A)))  
}
```



# Finding mean value by Map-Reduce - cnt.

- **MAP-REDUCE:**

```
Block_means <- from.dfs(  
  mapreduce(  
    input = "/CEnetBig",  
    map = mapper_mean,  
    reduce = reducer_mean  
  )  
)
```

- **Final code:**

```
GlobalMean=Block_means$val[[2]]/Block_means$val[[1]]
```

- **Result**

```
> GlobalMean  
[1] 129.4716
```



# Third Big Data challenge

**Goal:** Compute the variance of  $\sigma^2$  of the `CEnetBig[3]`.

**Note:**  $\sigma^2 = \frac{\sum_k (X_{k,2} - \bar{x}_2)^2}{n} = \frac{\sum_k X_{k,2}^2}{n} - \bar{x}_2^2$ .



# Third Big Data challenge - cnt.

```
mapper_var = function (., X) {  
  n=nrow(X);  
  mi=sum(X[,2]);  
  si=sum(X[,2]^2);  
  keyval(1:3,list(n,mi,si));  
}
```

```
reducer_var = function(k, A) {  
  keyval(k,list(Reduce('+', A)))  
}
```

```
Block_var <- from.dfs(  
  mapreduce(  
    input = "/CEnetBig",  
    map = mapper_var,  
    reduce = reducer_var  
  )  
)
```

```
globalVar=Block_var$val[[3]]/Block_var$val[[1]]-(Block_var$val[[2]]/Block_var$val[[1]])^2  
> globalVar  
[1] 595.1341
```



# Fourth Big Data challenge

**Goal:** Compute the covariance matrix  $\Sigma$  of the `CEnetBig[,2:13]` .

**Note:**  $\Sigma_{ij} = \frac{\sum_k (X_{ik} - \bar{x}_i)(X_{jk} - \bar{x}_j)}{n} = \frac{1}{n} (\tilde{X}^T \tilde{X})_{ij}$ .

Suppose `XX` is submatrix of `CEnetBig` of 1st 100 rows and with columns '2016\_1', ..., '2016\_12'. We find covariance matrix of `XX`

```
XX=CEnetBig$val[1:100,2:13]
Sigma=cov(XX)
```

**Note:** Naive approach will visit the data several times.



# Third Big Data challenge - cnt.

```
> Sigma
2016_1  2016_2  2016_3  2016_4  2016_5  2016_6  2016_7  2016_8  2016_9  2016_10  2016_11  2016_12
2016_1  554.66627 197.7795 144.7789 131.1854 249.1535 124.1262 252.6528  53.31369 199.2839 120.2593 257.9729 158.0293
2016_2  197.77949 687.8934 302.7297 307.0862 266.9029 261.8073 280.3199 252.36691 274.6391 247.4709 310.5588 140.8925
2016_3  144.77895 302.7297 762.0102 284.1748 247.8277 175.4163 283.0150 217.00145 321.8898 244.9201 413.3578 173.4369
2016_4  131.18542 307.0862 284.1748 605.7750 169.2399 253.4410 292.7296 209.68617 283.8475 247.4226 422.2579 219.1580
2016_5  249.15355 266.9029 247.8277 169.2399 541.3642 171.9361 227.3288 194.71391 293.5147 218.3279 253.6789 219.2686
2016_6  124.12617 261.8073 175.4163 253.4410 171.9361 567.5522 232.6065 183.04757 219.4846 192.3792 272.8218 140.0293
2016_7  252.65276 280.3199 283.0150 292.7296 227.3288 232.6065 681.2422 261.19614 293.7390 211.6760 450.0655 208.6689
2016_8   53.31369 252.3669 217.0015 209.6862 194.7139 183.0476 261.1961 639.62214 260.6902 101.4208 189.6450 187.1999
2016_9  199.28392 274.6391 321.8898 283.8475 293.5147 219.4846 293.7390 260.69023 635.4909 186.6704 370.9400 294.8504
2016_10 120.25931 247.4709 244.9201 247.4226 218.3279 192.3792 211.6760 101.42076 186.6704 706.0847 296.6746 169.5679
2016_11 257.97290 310.5588 413.3578 422.2579 253.6789 272.8218 450.0655 189.64504 370.9400 296.6746 877.7393 243.8821
2016_12 158.02993 140.8925 173.4369 219.1580 219.2686 140.0295 208.6689 187.19898 294.8569 169.5678 243.8821 561.2400
```



- Some mathematics:

$$\Sigma_{ij} = \frac{\sum_k (X_{ik} - \bar{x}_i)(X_{jk} - \bar{x}_j)}{n} = \frac{\sum_k X_{ik}X_{jk}}{n} - \bar{x}_i\bar{x}_j.$$

$$\Sigma = \frac{1}{n}X^T X - \bar{x}\bar{x}^T$$

- Block structure: Suppose we decompose

$$X = \begin{bmatrix} X^1 \\ X^2 \\ \vdots \\ X^k \end{bmatrix}$$

where  $X^i$  is a block of  $X$  having  $n_i$  rows.

- The “tough” product rewrites as

$$X^T X = \sum_{i=1}^k (X^i)^T X^i.$$



- **Similarly:** if  $n_i, s_i$  are row-sizes and column sums of blocks  $X^i$

$$\bar{x} = \frac{\sum_i s_i}{\sum_i n_i}. \quad (1)$$

```
mapperSS = function (., X) {  
  ni=nrow(X);  
  si=colSums(X[,2:13]);  
  SSi=t(X[,2:13])%*%X[,2:13];  
  keyval(1:3,list(ni,si,SSi));  
}
```

- **REDUCE:**

```
reducerSS = function(k, A) {  
  keyval(k,list(Reduce('+', A)))  
}
```



## ● MAP-REDUCE:

```
CovMatrixRaw <- from.dfs(  
  mapreduce(  
    input = "/CEnetBig",  
    map = mapperSS,  
    reduce = reducerSS  
  )  
)
```

## ● Final code

```
meanVec <- CovMatrixRaw$val[[2]]/CovMatrixRaw$val[[1]]  
CovMat <- CovMatrixRaw$val[[3]]/CovMatrixRaw$val[[1]] -outer(meanVec,meanVec)
```



Visit our **MOOC**:

Categories Courses Programs Degrees

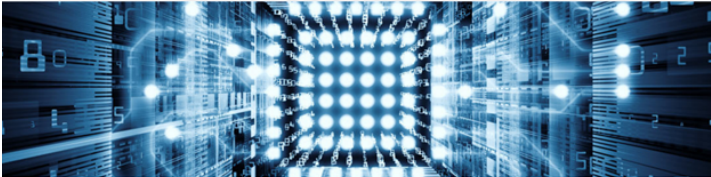
ONLINE COURSE

## Managing Big Data with R and Hadoop

Learn how to manage and analyse big data using the R programming language and Hadoop programming framework.

Join now – starts 23 Apr


Overview Topics Start dates Requirements Educators



Duration  
5 weeks

4 hours  
per week

Learn for free



# Challenge

Count the number of consumers with total consumption larger than 1500.



# Challenge

For each type find a list of consumers having this type of contract.

