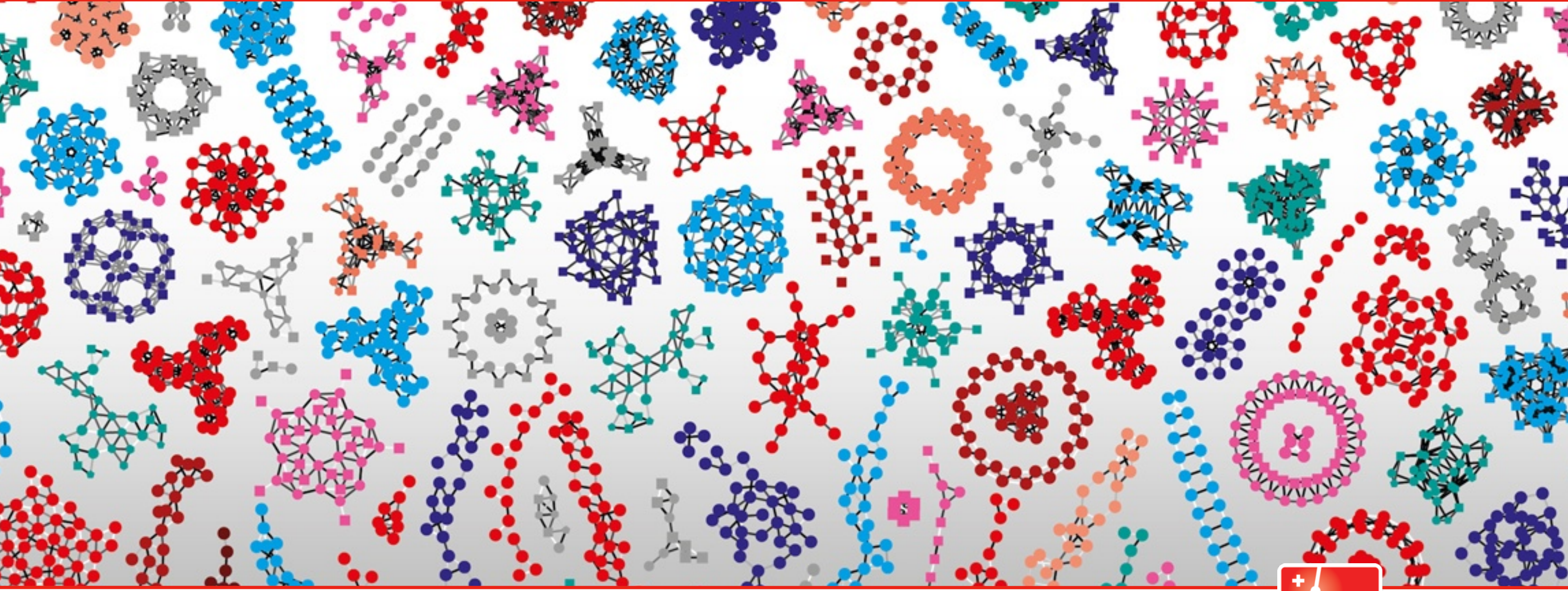


SIB
Swiss Institute of Bioinformatics

BCF

Statistical methods for big data in life sciences and health with R

Linda Dib, Frédéric Schütz

4th of June 2018

elixir SWITZERLAND

www.sib.swiss

Optimizing R available functions to handle big data set and use HPC

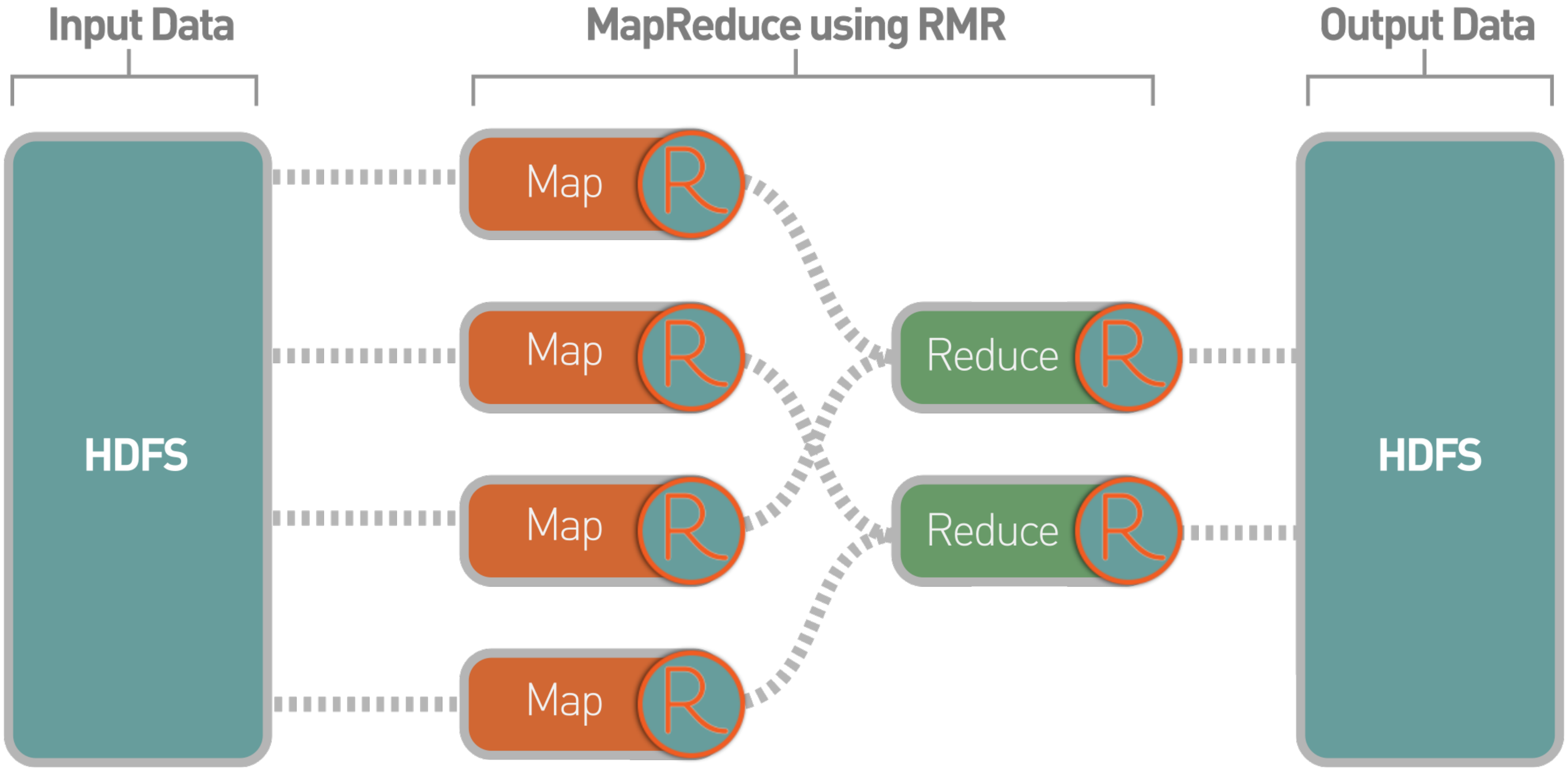
RevoScaleR Overview

- RevoScaleR package adds capabilities to R:
  - Data Import/Clean/Explore/Transform
  - Analytics – Descriptive and Predictive
  - Parallel and distributed computing
  - Visualization
- Scales from small local data to huge distributed data
- Scales from laptop to server to cluster to cloud
- Portable – the same code works on small and big data, and on laptop, server, cluster, Hadoop

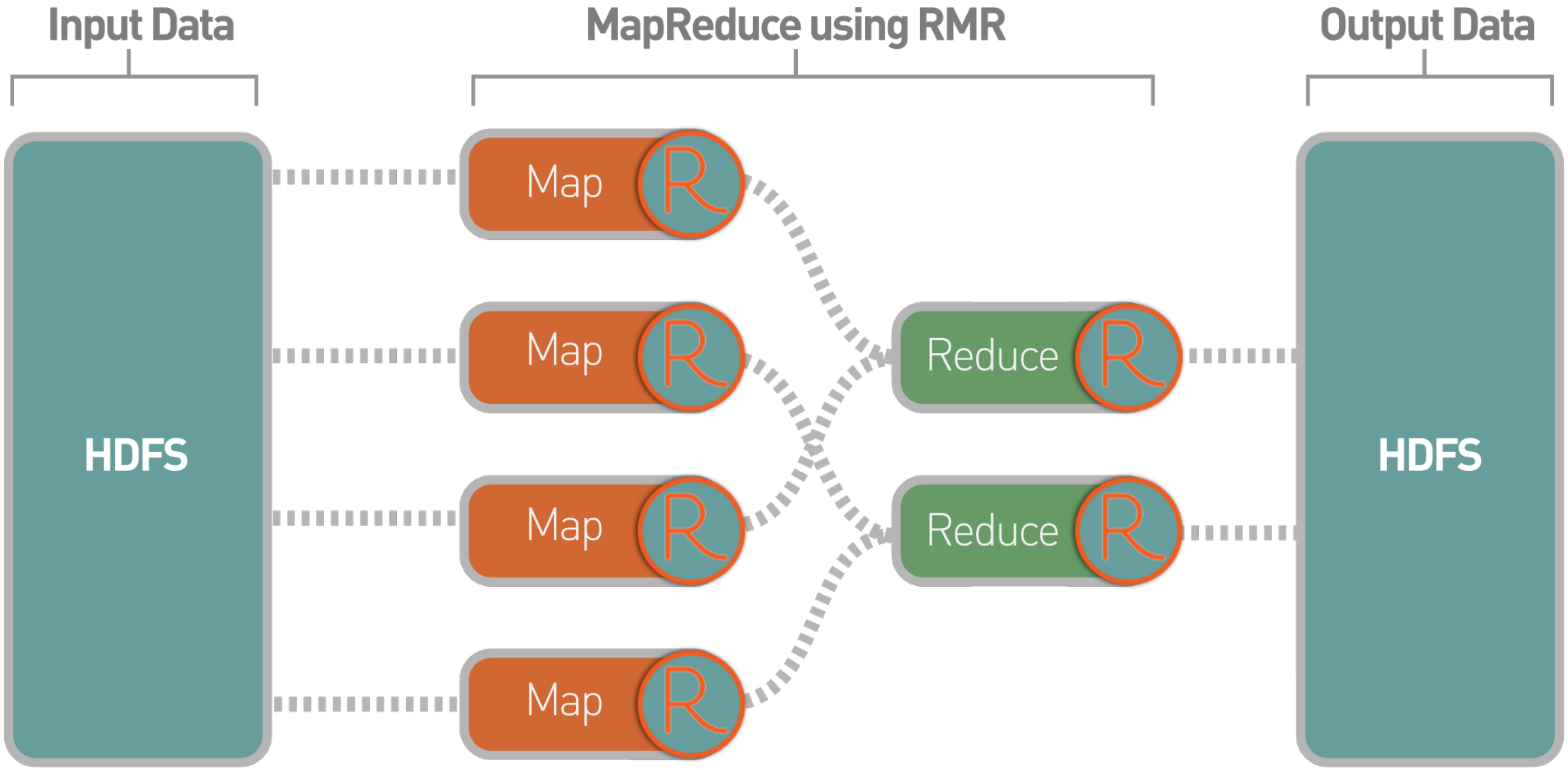
R and Hadoop

Revolution R combines R and Hadoop

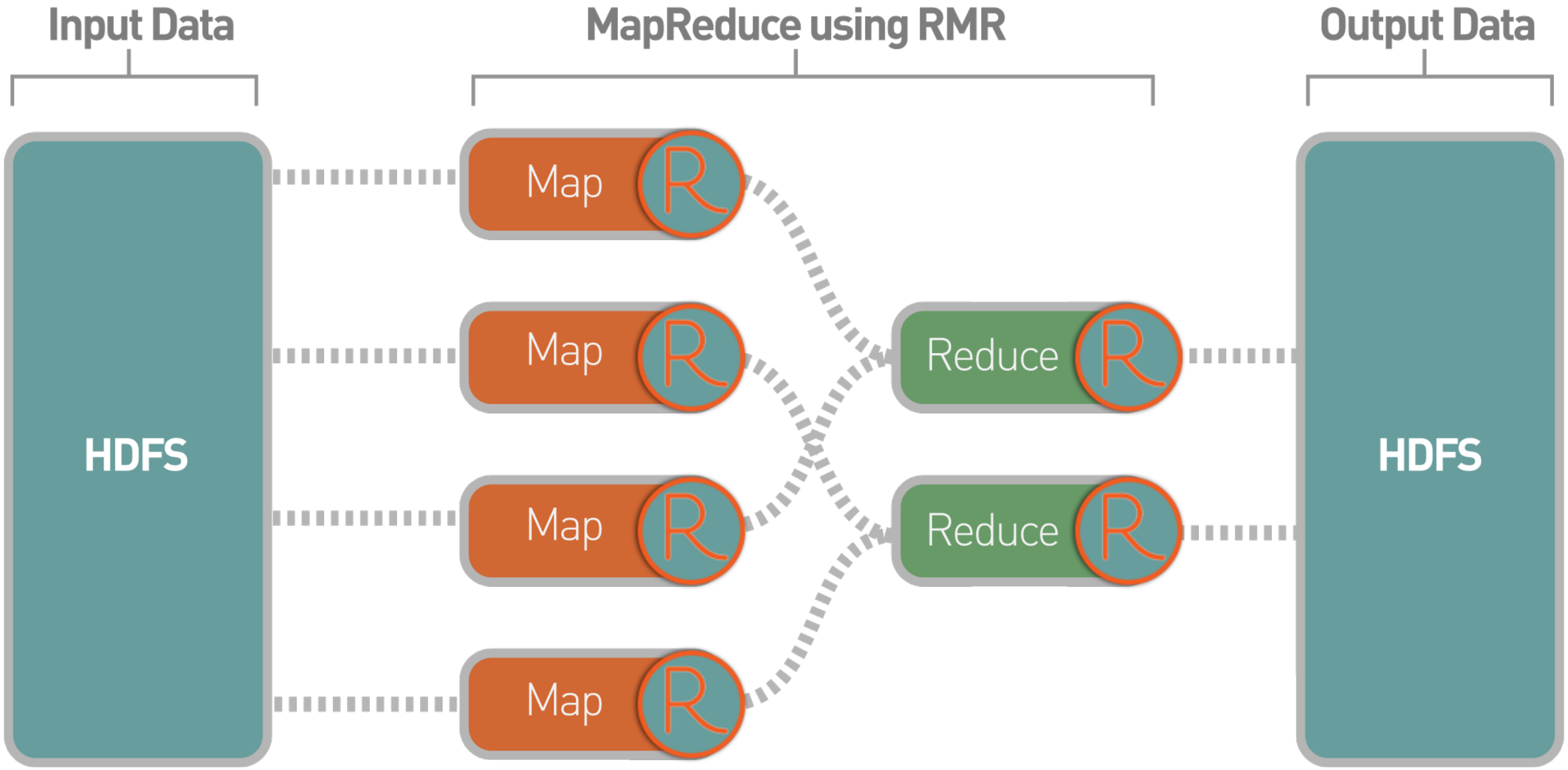
- Rmr2 : R and MAP-REDUCE
- Rhdfs : R and HDFS
- Rhbase : R and HBASE



- Use Hadoop for data storage and preparation
- No need to learn Java, Pig, Python or any other language



- Use RevolScaleR on connected server fore descriptive and predictive modeling



- Use Hadoop for model deployment

High Performance Analytics (HPA)

R Data Step

Descriptive Statistics

Statistical Tests

Sampling

Predictive Models

Data Visualization

Machine Learning

Simulation

Key HPA features

- Handles an arbitrarily large number of rows in a fixed amount of memory
- Scales linearly with the number of rows
- Scales linearly with the number of nodes
- Scales well with the number of cores per node
- Scales well with the number of parameters
- Extremely high performance

What makes it so fast?

- Lots of things!
- This kind of performance requires
  - Careful architecting that supports performance
  - Constant, intense focus on all of the details
  - A review of every line of code with an eye to performance (in addition to giving the correct answers, of course)
  - Extensive profiling and continuous benchmarking to detect problems and improve code

Specific speed-related factors

- Efficient computational algorithms
- Efficient memory management – minimize data copying and data conversion
- Heavy use of C++ templates; optimal code
- Efficient data file format; fast access by row and column
- Models are pre-analyzed to detect and remove duplicate computations and points of failure (singularities)
- Handle categorical variables efficiently

R & RevoScaleR

To be able to use RevoScaleR library:

- Install Microsoft R Open
- Install Microsoft R Server

To be able to use RevoScaleR library:

- Install Microsoft R Open

Microsoft R Open is the enhanced distribution of R (based on R 3.4.3) from Microsoft Corporation. It is a complete open source platform for statistical analysis and data science.

Updated the bundled packages
checkpoint, curl, doParallel, foreach, and iterators

To be able to use RevoScaleR library:

- Install Microsoft R Open
- Install Microsoft R Client or Server

R Client

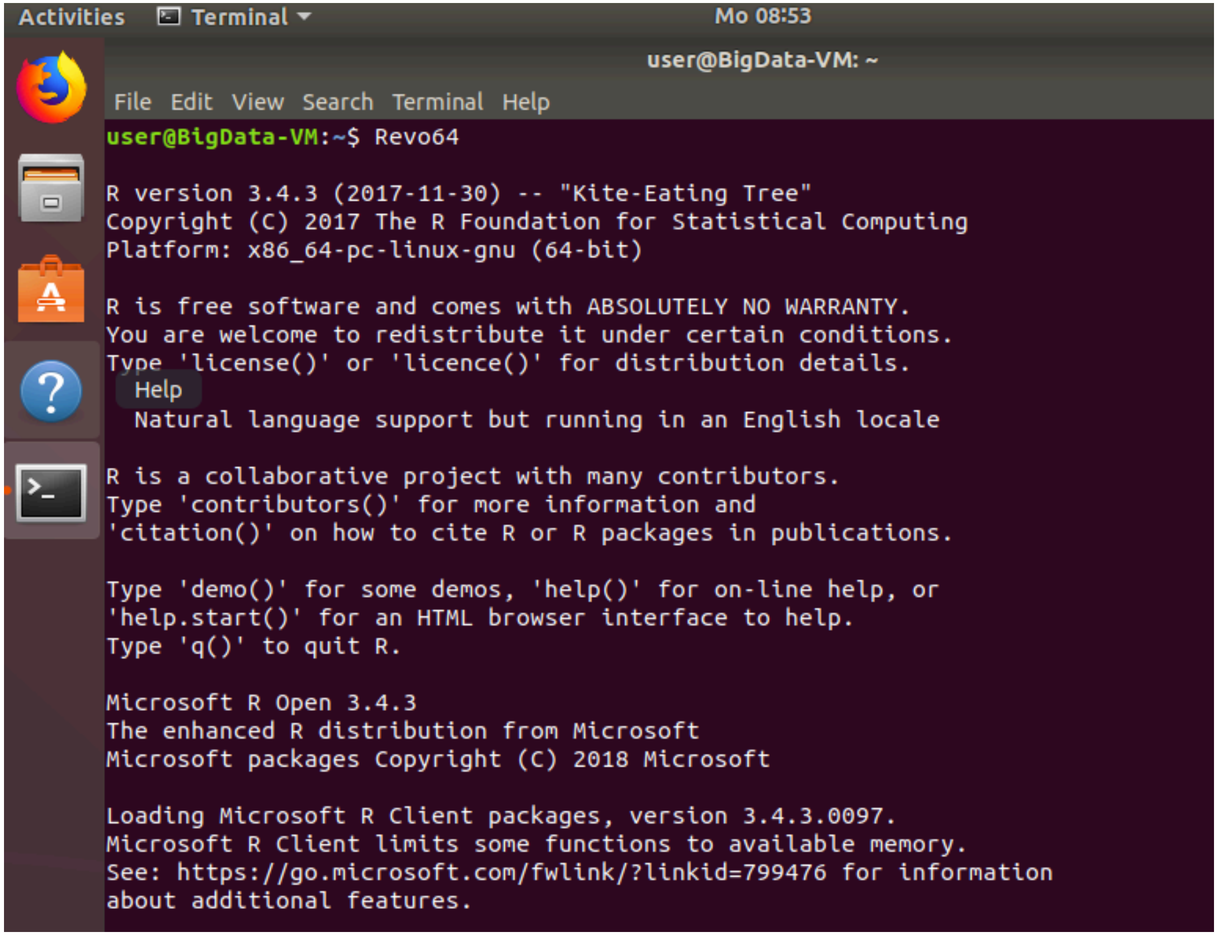
- for data scientists
- run locally

Machine Learning Server

- commercial software
- runs on a range of platforms
- greater scale
- with infrastructure for handling major workloads

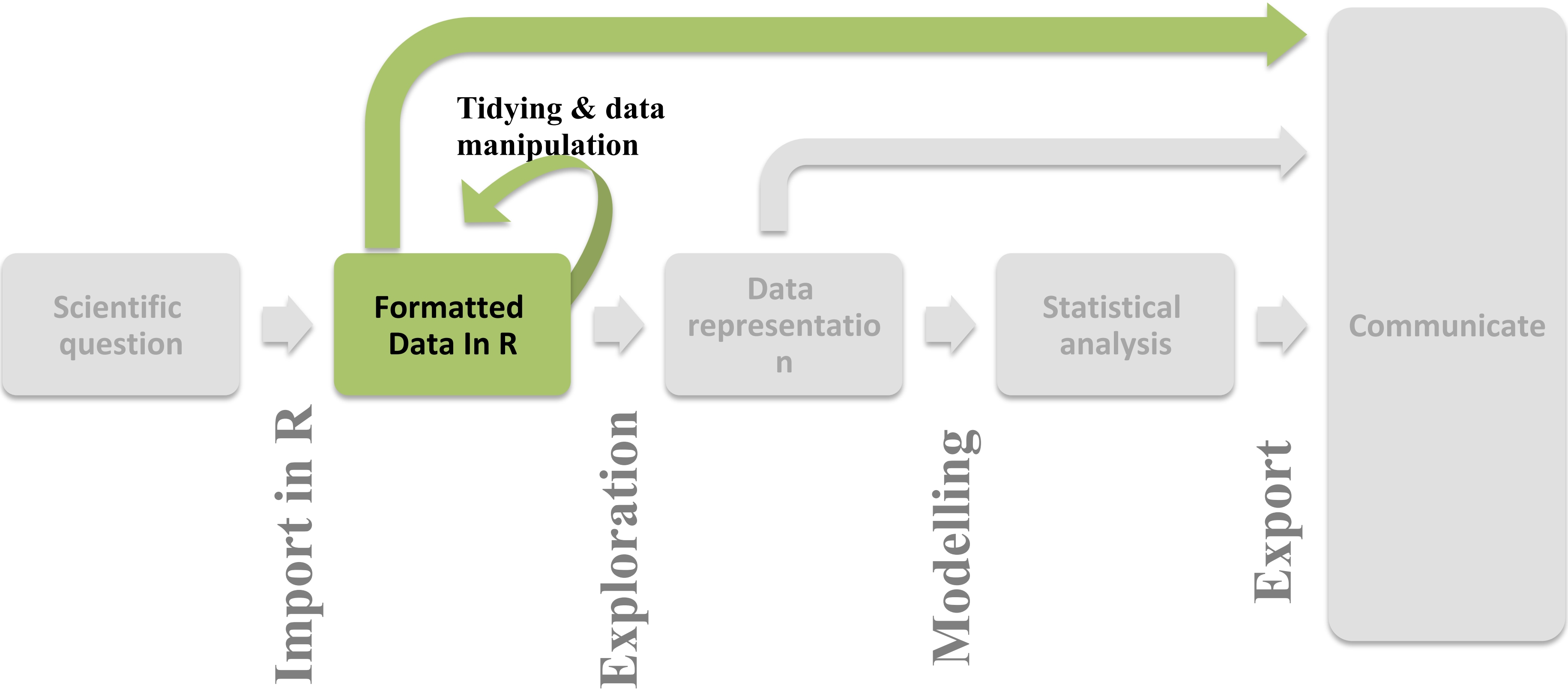
Virtual machine

In console type:
Revo64



```
user@BigData-VM:~$ Revo64

R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Microsoft R Open 3.4.3
The enhanced R distribution from Microsoft
Microsoft packages Copyright (C) 2018 Microsoft

Loading Microsoft R Client packages, version 3.4.3.0097.
Microsoft R Client limits some functions to available memory.
See: https://go.microsoft.com/fwlink/?linkid=799476 for information
about additional features.
```

R steps



Tidying & data manipulation

Scientific question

Formatted Data In R

Data representatio n

Statistical analysis

Communicate

Import in R

Exploration

Modelling

Export

Import data

rxImport function: allows you to import data from fixed or delimited text files, SAS files, SPSS files, or a SQL Server, Teradata, or ODBC connection.

```
>dataExample<-rxImport(inData = inDataFile)
```

Import data

rxImport function: allows you to import data from fixed or delimited text files, SAS files, SPSS files, or a SQL Server, Teradata, or ODBC connection.

```
>dataExample<-rxImport(inData = inDataFile)
```

Note:

There’s no need to have SAS or SPSS installed on your system to import those file types.

For database: you need a locally installed ODBC driver for your database to access data on a local or remote computer.

Import data

rxImport function: allows you to import data from fixed or delimited text files, SAS files, SPSS files, or a SQL Server, Teradata, or ODBC connection.

```
>dataExample<-rxImport(inData = inDataFile)
>nrow(dataExample)
>ncol(dataExample)
>names(dataExample)
>head(dataExample)
```

Import data

rxImport function: allows you to import data from fixed or delimited text files, SAS files, SPSS files, or a SQL Server, Teradata, or ODBC connection.

```
>dataExample<-rxImport(inData = inDataFile)
>rxGetInfo(dataExample, getVarInfo = TRUE, numRows=3)
```

Select and transform data

rxDataStep function: provides a framework for the majority of your data manipulation tasks.

```
>dataExampleNew<-rxDataStep(
    inData = dataExample,
    # Put in a placeholder for an output file
    outFile = "/media/sf_docVM/outFile2.csv", varsToDrop = c("year”))
```

Select and transform data

rxDataStep function: provides a framework for the majority of your data manipulation tasks.

```
>dataExampleNew<-rxDataStep(
    inData = dataExample,
    # Put in a placeholder for an output file
    outFile = "/media/sf_docVM/outFile2.csv", varsToDrop = c("year”),
    rowSelection = score < 850)
```

Select and transform data

rxDataStep function: provides a framework for the majority of your data manipulation tasks.

```
>dataExampleNew<-rxDataStep(
    inData = dataExample,
    outFile = outFile2, # Put in a placeholder for an output file
    varsToDrop = c("year"), # Specify any variables to keep or drop
    rowSelection = score < 850, # Specify rows to select
    # Specify a list of new variables to create
    transforms = list(
        expression = cut(ccExp, breaks = c(0, 6500, 13000),
            labels   = c("Low exp", "High exp")))
```

Select and transform data

rxDataStep function: provides a framework for the majority of your data manipulation tasks.

```
>dataExampleNew<-rxDataStep(
    inData = dataExample,
    outFile = outFile2, # Put in a placeholder for an output file
    varsToDrop = c("year"), # Specify any variables to keep or drop
    rowSelection = score < 850, # Specify rows to select
    # Specify a list of new variables to create
     transforms = list(
        expression = floor(ccExp/100)))
```

Select and transform data

rxDataStep function: provides a framework for the majority of your data manipulation tasks.

```
>dataExampleNew<-rxDataStep(
    inData = dataExample,
    outFile = outFile2, # Put in a placeholder for an output file
    varsToDrop = c("year"), # Specify any variables to keep or drop
    rowSelection = score < 850, # Specify rows to select
    # Create a new column ”newCol" in data.
      transforms = list(newCol= ID)
)
```

Merge

```
>rxMerge (inData1=xxx,
          inData2= yyy,
          outFile = outFile,
          type="inner",
          matchVars=c(gene))
```

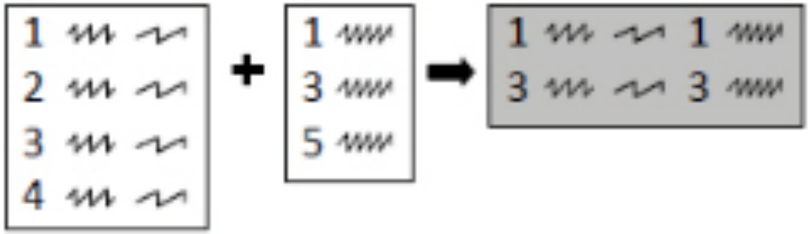
xxx		
geneID	sample1	sample2

yyy				
geneID	sample3	sample4	sample5	sample6

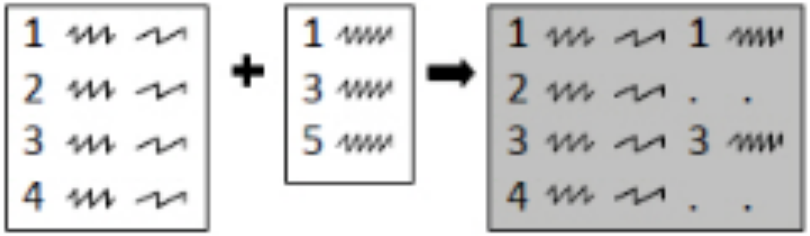
Merge

```
>rxMerge  (inData1=xxx,
           inData2= yyy,
           outFile = outFile,
           type=“inner”,
           matchVars=c(gene))
```

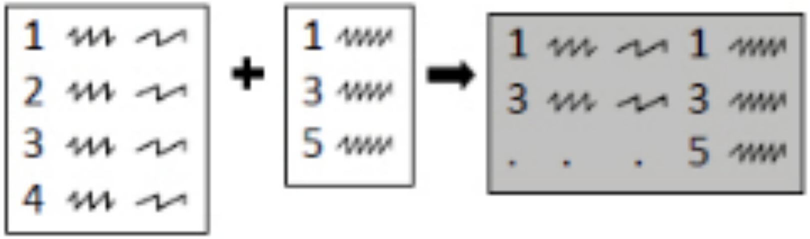
geneID	sample1	sample2	sample3	sample4	sample5	sample6



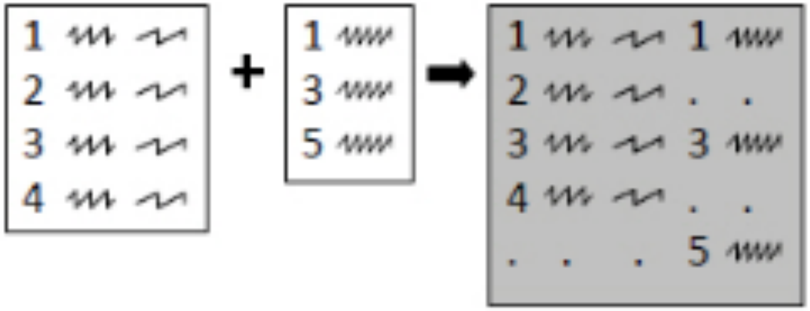
Inner joins An inner join, the default, keeps only rows that match.



Left joins In a left join, all rows from the table on the left are kept even if they do not have a match in the other table.



Right joins In a right join, all rows from the table on the right are kept even if they do not have a match in the other table.



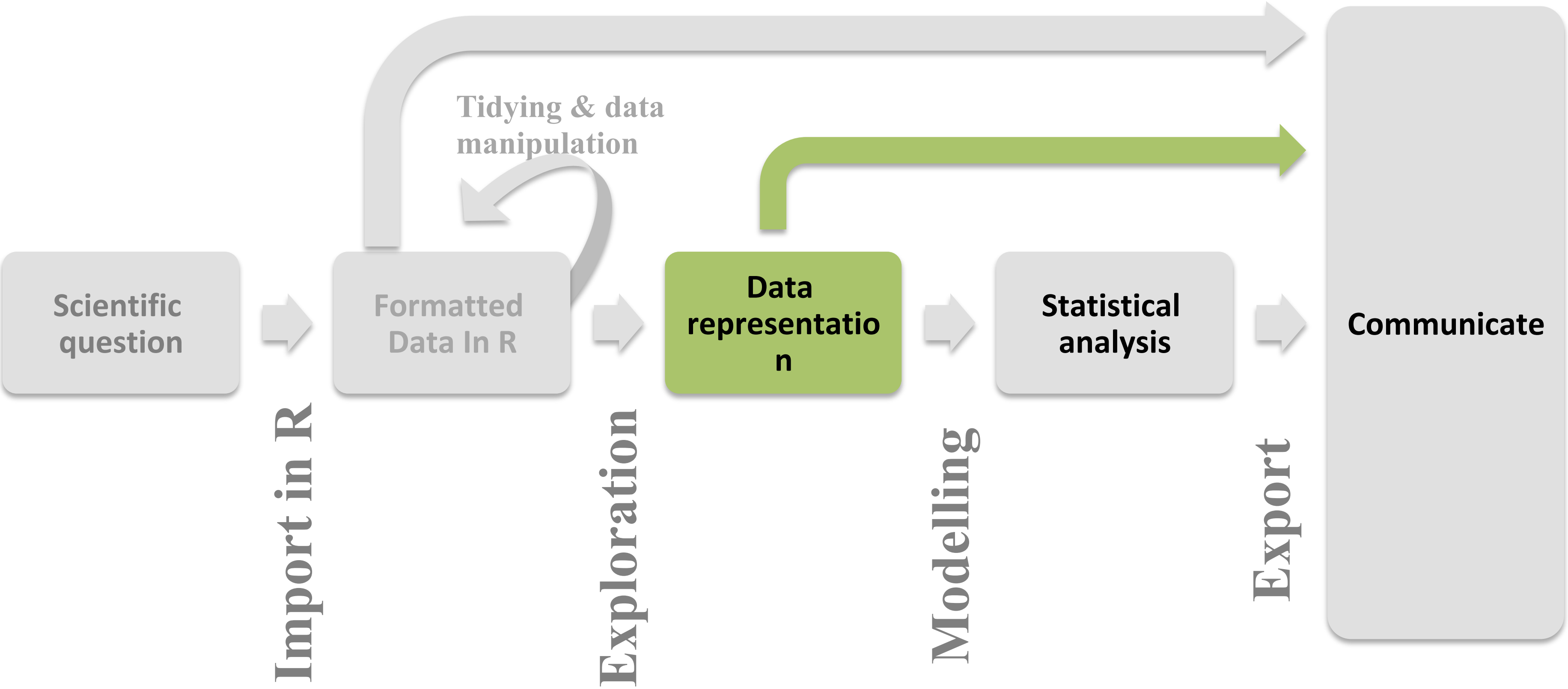
Full outer joins In a full outer join, all rows are kept from both tables.

factor

Call the rxFactors() function to convert variable to categorical.

```
>rxFactors(inData = destData_mrs,
           outFile = outFileFinal,
           sortLevels = TRUE,
           factorInfo = c(”var1”),
           overwrite = TRUE)
```

R steps



Scientific question

Formatted Data In R

Data representatio n

Statistical analysis

Communicate

plot

Histogram

```
>rxHistogram(~score, data = dataExample )
```

Exercise1: first steps

1.Use the virtual machine and import the mortality.csv file using RevoScaleR Package.

2. Transform the data by dropping the column named year

3. Transform the data by adding a new column with values high or low: Low when ccExp are between 0 and 6500 and high when when ccExp are between 6500 and 13000

4. Transform the data by keeping only score < 625

5. Plot the historgram of score for each of the newdatasets created in 1, 2, 3 and 4.

Exercise1: first steps-solution

```
>datacsv <- rxImport("/media/sf_docVM/firststeps/mortality.csv”)
>rxGetInfo(datacsv, getVarInfo = TRUE, numRows=3)
>dataExampleNew1<-rxDataStep(inData = datacsv,outFile = "/media/sf_docVM/firststeps/outFile1.csv", varsToDrop = c("year"))
>dataExampleNew2<-rxDataStep(inData = datacsv, outFile = "/media/sf_docVM/firststeps/outFile2.csv", varsToDrop = c("year")),  rowSelection = score < 850,  transforms = list(expression = cut(ccExp, breaks = c(0, 6500, 13000),labels  = c("Low exp", "High exp")),clinicLow  = score < 625))
>rxHistogram(~score, data = datacsv )
>datacsvN1 <- rxImport("/media/sf_docVM/firststeps/outFile1.csv")
>datacsvN2 <- rxImport("/media/sf_docVM/firststeps/outFile2.csv")
>rxHistogram(~score, data = datacsvN1 )
>rxHistogram(~score, data = datacsvN2 )
```

Exercise2:

1.Use the virtual machine and import Flight_Delays_Sample.csv and Weather_Sample.csv files using RevoScaleR Package.

2. Merge the data sets once you have renamed the columns of weather, using originalAirportID

3. Join flight records and weather data using the destination of the flight DestAirportID

4. Call the rxFactors() function to convert OriginAirportID and DestAirportID as categorical.

Thank you for your attention

HPC Capabilities in RevoScaleR

- Execute (essentially) any R function in parallel on nodes and cores
- Results from all runs are returned in a list to the user's laptop
- Extensive control over parameters
- Extensive control over nodes, cores, and times to run
- Ideal for simulations and for running R functions on small amounts of data

Key ways RevoScaleR enhances R

- High Performance Computing (HPC) functions: Parallel/Distributed computing
- High Performance Analytics (HPA) functions: Big Data + Parallel/Distributed computing
- XDF file format; rapidly store and extract data
- Use results from HPA and HPC functions in other R packages