iCMS 2015
INTERNATIONAL CONFERENCE ON COMPUTING, MATHEMATICS AND STATISTICS
Bridging Research Endeavour in Computer and Mathematical Sciences
For more information, please visit http://www.icms2015.org
Organized by : FACULTY of COMPUTER & MATHEMATICAL SCIENCES UiTM KEDAH
Jointly organized by : RESEARCH & INDUSTRIAL LINKAGES
4th - 5th November 2015
Langkawi Island, MALAYSIA
UNIVERSITI TEKNOLOGI MARA CAWANGAN KEDAH

# PRE-CONFERENCE WORKSHOP "INTRODUCTION TO R AND DATA VISUALIZATION"

Ciprian Alexandru

R-omania Team | www.r-project.ro

# Presentation



- The R platform provides a powerful and comprehensive platform for visualizing data, understanding and evaluating statistical models, and effectively communicating research results to both technical and nontechnical audiences. This 2 days workshop will provide practical review of R's major graphing capabilities; including base functionality and new capabilities provided by the lattice and ggplot2 packages.

# Date & Location



- Date: 2 - 3 November 2015
- Time: 9 am - 5 pm
- Venue: Melur 1, Langkawi Lagoon, Langkawi Island, MALAYSIA

# Who should attend?



- R is widely used within the academia especially in the fields of computational biology, applied science, quantitative finance and business intelligence. R is capable of solving challenging problems and among the strengths of R are its powerful built-in tools for inferential statistics, its compact modeling syntax, and its data visualization capabilities. In addition, R's open source nature and its extensibility via add-on "packages" has allowed it to keep up with the leading edge in academic research. This workshop on R and Data Visualization is suitable and relevant for:
- Lecturers, Researchers, Engineers, Students, Industry Professionals and Scientists of any discipline who wish to explore R. Prior experience with R is not required. Interested to join??? Please register here.

# Speakers



*Antoniade-Ciprian Alexandru* is an Associate Professor at the Ecological University of Bucharest and the dean of the Faculty of Economics. He is also attached with the National Institute of Statistics, Bucharest as an expert trainer in data analyst using R environment. Dr Alexandru is one of the six members of the R-omania team, a team that promotes R projects for statistical computing by providing a free and open source software environment for data analysis and graphics. The team acts as a user community for development of R projects among individuals, institutions, commercial entities and non-profit organizations. Dr Alexandru participated in various research projects, workshops, and, national and international conferences. His research works were published in various international databases. Currently, he is working on a project that implements the use of R as a tool for analyzing the evolution of indices on the stock market.

*Nicoleta Caragea* is an Associate Professor at the Faculty of Economics, Ecological University of Bucharest and a senior expert at the National Institute of Statistics. Her teaching activity is focused mainly in the field of statistics, through courses and seminars and master degree programs (statistics, economic statistics, social statistics, economic and financial analysis). Dr Caragea participated as a national expert in various projects, workshops and conferences organized by EUROSTAT, OECD, WHO, World Bank and UNICEF-UIS. She is one of the other six members of the R-omania team, a team that promotes R projects for statistical computing by providing a free and open source software environment for data analysis and graphics. She also acted as a consultant in projects in Europe. Her latest work was as a technical assistance to a consultancy work in Turkey.

# Course Outline 1/2



- Introduction to R Statistical Software
  - The beginning of R
  - R - Introducing the R Console
  - R - Installation, Packages, CRAN, Components
  - Graphical User Interfaces: R Console, R Studio, R Commander, R resources and online community
- Databases
  - Data manipulation
  - Queries
  - Using SQL within R
  - Data aggregation
  - Matching

# Course Outline 2/2



- Data Visualization & Graphics Environments
  - Base graphics (Scatterplot, Box-and-whiskers plot, Histogram)
  - Lattice
  - ggplot2
  - Interactive graphics in R
  - Reproducibility
- Regression Analysis with R
  - Linear regression models
  - OLS-ordinary least squares method for estimating the unknown parameters in a linear regression model
  - Interpreting the regression coefficients
  - Extensions to generalized linear models. Logistic regression
  - Parameter estimates – maximum likelihood method
  - Definition of the odds and odds ratio
  - Evaluating goodness of fit

# Databases



- Data manipulation
- Queries
- Using SQL within R
- Data aggregation
- Matching

# Operators & functions



- standard arithmetic operators: +, -, *, and /
- mathematical functions: sqrt, exp, and log
- relational operators <=, <, ==, >, >= and !=
- logical operators: | for OR and & for AND
- assignment operators: <- or = and ->

```
Variable x gets value 2:
x <- 2

Value 2 goes to variable x:
2 -> x
```

```
> x <- pi/sqrt(2)
> x
[1] 2.221441
> pi/sqrt(2) -> y
> y
[1] 2.221441
> x == y
[1] TRUE
```

# Operator syntax



- $ component extraction
- [ [[ indexing
- : sequence operator

```
> x <- c(1:10)
> x[(x < 5) | (x > 8)]
[1] 1 2 3 4 9 10
```

```
> 1:5
[1] 1 2 3 4 5
> (a <- data.frame(name = c("John", "Mary"), income = c(1800, 2500)))
  name income
1 John   1800
2 Mary   2500
> a$name
[1] John Mary
Levels: John Mary
> a[1]
  name
1 John
2 Mary
> a[2]
  income
1   1800
2   2500
> a[[1]]
[1] John Mary
Levels: John Mary
```

# Functions



**Rounding**

Various sorts of rounding (rounding up, rounding down, rounding to the nearest integer)

```
# rounding down, the 'greatest integer less than' function is floor
floor(5.7)
[1] 5
# rounding up, the 'next integer' function is ceiling
ceiling(5.7)
[1] 6
# rounding to the nearest integer by adding 0.5 to the number then using floor
rounded<-function(x) floor(x+0.5)
rounded(5.7)
[1] 6
rounded(5.4)
[1] 5
```

# +Inf, -Inf, NaN



- R is properly infinite numerical values
- NaN – Not a Number
- Complex number:

```
> sqrt(as.complex(-2))
[1] 0+1.414214i
> sqrt(-2+0i)
[1] 0+1.414214i
```

```
> (a <- 2/0)
[1] Inf
> class(a)
[1] "numeric"
> exp(a)
[1] Inf
> exp(-a)
[1] 0
> a - a
[1] NaN
> sqrt(a)
[1] Inf
```

# R objects



- Five "atomic" classes of objects:
  - character
  - numeric (real numbers)
  - integer
  - complex
  - logical (True/False)

```
> (x <- "a")   # character
[1] "a"
> class(x)
[1] "character"
> (x <- 1)     # numeric
[1] 1
> class(x)
[1] "numeric"
> (x <- 1:5)   # integer
[1] 1 2 3 4 5
> class(x)
[1] "integer"
> (x <- 2+3i) # complex
[1] 2+3i
> class(x)
[1] "complex"
> (x <- TRUE) # logical
[1] TRUE
> class(x)
[1] "logical"
```

```
> a <- 1
> class(a)
[1] "numeric"
> typeof(a)
[1] "double"
> b <- 1:2
> class(b)
[1] "integer"
> typeof(b)
[1] "integer"
> is.numeric(a)
[1] TRUE
> is.numeric(b)
[1] TRUE
```

```
> a <- 1
> b <- as.integer(1)
> a == b
[1] TRUE
> identical(a, b)
[1] FALSE
```

*near equality*

```
> (a <- 0.2 + 0.2 + 0.2)
[1] 0.6
> (b <- 0.6)
[1] 0.6
> a == b
[1] FALSE
> all.equal(a, b)
[1] TRUE
> identical(a, b)
[1] FALSE
```

# R – data structures



- factors
- atomic vector
- matrix
- array
- data frame
- list
- table


| | Homogeneous | Heterogeneous |
|---|---|---|
| 1d | Atomic vector | List |
| 2d | Matrix | Data frame |
| nd | Array | |

# R - factor object



- Factors - categorical data (unordered or ordered)

```
> y <- c("yes", "no", "yes", "yes", "yes", "no")
> x <- c("yes", "no", "yes", "yes", "yes", "no")
> y <- as.factor(x)
> x
[1] "yes" "no"  "yes" "yes" "yes" "no"
> y
[1] yes no  yes yes yes no
Levels: no yes
> str(x)
 chr [1:6] "yes" "no" "yes" "yes" "yes" "no"
> str(y)
 Factor w/ 2 levels "no","yes": 2 1 2 2 2 1
```

*alphabetical order*

```
> x <- factor(c("yes", "no", "yes", "yes", "no"), levels = c("yes", "no"))
> x
[1] yes no  yes yes no
Levels: yes no
```

```
> table(y)
y
 no yes
  2   4
> levels(y)
[1] "no"  "yes"
```

# R – data structures – atomic vector



- vector("character", length = 5)

- character()
- logical()
- numeric()
- integer()
- complex()

- as.character()
- as.logical()
- as.numeric()
- as.integer()
- as.complex()

```
> (a <- vector("character", length = 5))
[1] "" "" "" "" ""

> character(3)
[1] "" "" ""
> logical(3)
[1] FALSE FALSE FALSE
> numeric(3)
[1] 0 0 0
> integer(3)
[1] 0 0 0
> complex(3)
[1] 0+0i 0+0i 0+0i

> (a <- numeric(3))
[1] 0 0 0
> as.logical(a)
[1] FALSE FALSE FALSE
> as.character(a)
[1] "0" "0" "0"
```

# R – data structures – matrix



- 2D vector, homogeneous data type
- matrix(nrow = 5, ncol = 2)
- cbind()
- rbind()

```
> (m <- matrix(1:10, nrow = 5, ncol =2))
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> dim(m)
[1] 5 2
> m[2,]
[1] 2 7
> m[,1]
[1] 1 2 3 4 5
> m[3,2]
[1] 8
```

```
> age <- c(25, 43, 27, 36)
> height <- c(175, 180, 168, 183)
> (employees <- cbind(age, height))
     age height
[1,]  25    175
[2,]  43    180
[3,]  27    168
[4,]  36    183
> row.names(employees) <- c("John", "Mary", "Edy", "Tony")
> employees
     age height
John  25    175
Mary  43    180
Edy   27    168
Tony  36    183
> (empl <- rbind(age, height))
       [,1] [,2] [,3] [,4]
age      25   43   27   36
height  175  180  168  183
```

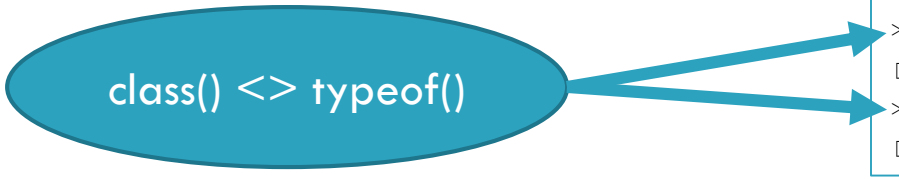
# R – data structures – array



- *n*D vector, homogeneous data type

- array(data, dim = length(data), dimnames = NULL)
- as.array(x, )
- is.array(x)

```
> (3d_vector <- array(1:24, c(3, 4, 2)))
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
> class(3d_vector)
[1] "array"
> typeof(3d_vector)
[1] "integer"
```



# R – data structures – data frame



- specific for **data analysis**/statisticians
- fundamental data structure by most of **R**'s modeling software
- 2D vector (matrix), Heterogeneous data type
- list of vectors of equal length
- data.frame(…, row.names = NULL, check.rows = FALSE, check.names = TRUE, stringsAsFactors = default.stringsAsFactors())
- is.data.frame(x)
- dim(x)
- ncol(x)
- nrow(x)
- x[*row*, *col*] or x[*observation*, *variable*]

```
> age <- c(25, 43, 27, 36)
> height <- c(175, 180, 168, 183)
> eye_color <- c("amber", "brown", "blue", "brown")
> (employees <- data.frame(age, height, eye_color)
+ )
  age height eye_color
1  25    175     amber
2  43    180     brown
3  27    168      blue
4  36    183     brown
> class(employees)
[1] "data.frame"
> typeof(employees)
[1] "list"
> class(employees$age)
[1] "numeric"
> class(employees$eye_color)
[1] "factor"
> employees[3, 1]
[1] 27
```

# R – data structures – list



- 2D vector (matrix), Heterogeneous data type
- list of vectors of unequal length
- an ordered collection of objects (components)

- list(…)
- as.list()
- is.list(x)

```
> age <- c(25, 43, 27, 36)
> my_matrix <- matrix(1:12, 3, 4)
> (my_list <- list(age = age, m <- my_matrix))
$age
[1] 25 43 27 36

[[2]]
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> my_list[[1]]
[1] 25 43 27 36
> my_list[[2]]
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> my_list[[2]][2, 2]
[1] 5
```

# R – data structures – table



- contingency table of the counts at each combination of factor levels

- table(…)
- as.table()
- is.table(x)

```
> a <- rep(c(NA, 1/0:3), 10)
> table(a)
a
0.333333333333333               0.5                 1               Inf
               10                10                10                10
> a
 [1]        NA       Inf 1.0000000 0.5000000 0.3333333        NA       Inf
 [8] 1.0000000 0.5000000 0.3333333        NA       Inf 1.0000000 0.5000000
[15] 0.3333333        NA       Inf 1.0000000 0.5000000 0.3333333        NA
[22]       Inf 1.0000000 0.5000000 0.3333333        NA       Inf 1.0000000
[29] 0.5000000 0.3333333        NA       Inf 1.0000000 0.5000000 0.3333333
[36]        NA       Inf 1.0000000 0.5000000 0.3333333        NA       Inf
[43] 1.0000000 0.5000000 0.3333333        NA       Inf 1.0000000 0.5000000
[50] 0.3333333
> table(a, exclude = NULL)
a
0.333333333333333               0.5                 1               Inf
               10                10                10                10
             <NA>
               10
```

# R – object attributes



- names
- dimnames
- dim
- class
- comment
- row.names
- attributes (contain metadata)

```
> a <- "John Dow"
> str(a)
 chr "John Dow"
> class(a)
[1] "character"
> length(a)
[1] 1
> nchar(a)
[1] 8

> v <- 1:5
> str(v)
 int [1:5] 1 2 3 4 5
> class(v)
[1] "integer"
> length(v)
[1] 5
```

```
> v <- 1:5
> str(v)
 int [1:5] 1 2 3 4 5
> class(v)
[1] "integer"
> length(v)
[1] 5
> v
[1] 1 2 3 4 5
> names(v) <- paste("Col", 1:5, sep="_")
> v
Col_1 Col_2 Col_3 Col_4 Col_5
    1     2     3     4     5
```

# Viewing data series available in packages



- data() – list all available packages in R environment
- data(package = "nlme") – list all available packages from *nlme*
- data(Earthquake, package = "nlme") – load into memory the *Eartquake* dataset

```
> data(package = "nlme")
> data(Earthquake, package = "nlme")
> head(Earthquake,3)
    Quake Richter distance soil accel
132    20       5      7.5    1 0.264
133    20       5      8.8    1 0.263
134    20       5      8.9    1 0.230
```

# Data selection and manipulation I



- which.max(x), which.min(x) - returns the index of the greatest/smallest element of x
- rev(x) - reverses the elements of x
- sort(x) - sorts the elements of x in increasing order; to sort in decreasing order: rev(sort(x))
- cut(x,breaks) - divides x into intervals (factors); breaks is the number of cut intervals or a vector of cut points
- match(x, y) returns a vector of the same length as x with the elements of x that are in y (NA otherwise)
- which(x == a) returns a vector of the indices of x if the comparison operation is true (TRUE), in this example the values of i for which x[i] == a (the argument of this function must be a variable of mode logical)
- choose(n, k) computes the combinations of k events among n repetitions = n!/[(n − k)!k!]
- na.omit(x) suppresses the observations with missing data (NA)
- na.fail(x) returns an error message if x contains at least one NA complete.cases(x) returns only observations (rows) with no NA
- unique(x) if x is a vector or a data frame, returns a similar object but with the duplicates suppressed

# Data selection and manipulation II



- table(x) returns a table with the numbers of the different values of x (typically for integers or factors)
- split(x, f) divides vector x into the groups based on f
- subset(x, ...) returns a selection of x with respect to
- criteria (..., typically comparisons: x$V1 < 10); if x is a data frame, the option select gives variables to be kept (or dropped, using a minus)
- na.fail(x) returns an error message if x contains at least one NA
- complete.cases(x) returns only observations (rows) with no NA
- unique(x) if x is a vector or a data frame, returns a similar object but with the duplicates suppressed
- table(x) returns a table with the numbers of the different values of x (typically for integers or factors)
- split(x, f) divides vector x into the groups based on f
- subset(x, ...) returns a selection of x with respect to
- criteria (..., typically comparisons: x$V1 < 10); if x is a data frame, the option select gives variables to be kept (or dropped, using a minus)

# Data reshaping



- merge(a,b) merge two data frames by common col or row names
- stack(x, ...) transform data available as separate cols in a data frame or list into a single col
- unstack(x, ...) inverse of stack()
- rbind(...) , cbind(...) combines supplied matrices, data frames, etc. by rows or cols
- melt(data, id.vars, measure.vars) changes an object into a suitable form for easy casting, (reshape2 package)
- cast(data, formula, fun) applies fun to melted data using formula (reshape2 package)
- recast(data, formula) melts and casts in a single step (reshape2 package)
- reshape(x, direction...) reshapes data frame between 'wide' (repeated measurements in
- separate cols) and 'long' (repeated measurements in separate rows) format based on direction
- aggregate(x,by,fun) input df; output df; applies fun to subsets of x, as grouped based on index. Can use formula notation

# Online resources



- http://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf
- http://cran.r-project.org/doc/contrib/YanchangZhao-refcard-data-mining.pdf

# R – Terminology (R vs. RDBMS*)



| R | SQL/RDBMS |
|---|---|
| data frame | table (relation) |
| observation | row |
| variable | column (attribute) |
| various ( [], subset(), order(), sort() ) | SELECT statements |

*) RDBMS - Relational Database Management Systems

# R – data manipulation



- selection
- sorting
- concatenation / merging data frames
- finding and removing duplicate records
- levels identification
- renaming levels
- changing the order of levels of a factor
- adding and removing variables (columns) from a data frame
- reordering the variables in a data frame
- renaming of variables

image source: http://www.swansea.ac.uk/medicine/courses/msc-health-informatics/r-courses/

# R – selection 1



- selecting observations / rows satisfy a certain condition

```
> data("airquality", package = "datasets")
> ls()
[1] "airquality"
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
> dim(airquality)
[1] 153   6
> str(airquality)
'data.frame':   153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
> levels(as.factor(airquality$Month))
[1] "5" "6" "7" "8" "9"
> mySel <- subset(airquality, Month == 6)
> dim(mySel)
[1] 30  6
```

# R – selection 2



- selecting variables / cols satisfy a certain condition

```
> mySel <- subset(airquality, Month == 5 | Month == 6,
+ select = c(Wind:Month))
> dim(mySel)
[1] 61  3
> head(mySel, 3)
  Wind Temp Month
1  7.4   67     5
2  8.0   72     5
3 12.6   74     5
```

```
> mySel <- subset(airquality, select = c(Wind, Temp))
> dim(mySel)
[1] 153   2
> head(mySel, 3)
  Wind Temp
1  7.4   67
2  8.0   72
3 12.6   74
> mySel1 <- subset(airquality, select = c(3:6))
> dim(mySel1)
[1] 153   4
> head(mySel1, 3)
  Wind Temp Month Day
1  7.4   67     5   1
2  8.0   72     5   2
3 12.6   74     5   3
> mySel2 <- subset(airquality, select = c(Wind:Day))
> all.equal(mySel1, mySel2)
[1] TRUE
```

# R – selection []



- selecting with [] operator, specific to data.frame:
- observations:
  - dataframename[condition, ]
- variables:
  - dataframename[ , condition]

```
> mySel1 <- airquality[airquality$Month == 6, ]
> dim(mySel1)
[1] 30  6
> head(mySel1, 3)
   Ozone Solar.R Wind Temp Month Day
32    NA     286  8.6   78     6   1
33    NA     287  9.7   74     6   2
34    NA     242 16.1   67     6   3
> mySel2 <- airquality[, c("Wind", "Temp")]
> dim(mySel2)
[1] 153   2
> head(mySel2, 3)
  Wind Temp
1  7.4   67
2  8.0   72
3 12.6   74
> mySel <- airquality[, c(Wind:Temp)]
Error in `[.data.frame`(airquality, , c(Wind:Temp)) :
  object 'Wind' not found
```

# R – sorting 1



- sort() function returns an array with elements properly sequenced data options by arguments, ascending or descending

- order() returns a vector with the same length as an argument passed to the function, but with the positions that values should handle vector elements source, in ascending or descending order

```
> data("airquality", package="datasets")
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
> head(airquality$Temp)
[1] 67 72 74 62 56 66
> sort(head(airquality$Temp))
[1] 56 62 66 67 72 74
> order(head(airquality$Temp))
[1] 5 4 6 1 2 3
> airquality <- airquality[order(airquality$Temp), ]
> head(airquality, 3)
   Ozone Solar.R Wind Temp Month Day
5     NA      NA 14.3   56     5   5
18     6      78 18.4   57     5  18
25    NA      66 16.6   57     5  25
> airquality <- airquality[order(airquality$Temp, decreasing = TRUE), ]
> head(airquality, 3)
    Ozone Solar.R Wind Temp Month Day
120    76     203  9.7   97     8  28
122    84     237  6.3   96     8  30
121   118     225  2.3   94     8  29
```

# R – sorting 2



- na.last = TRUE

- na.last = NA

```
> airquality <- airquality[order(airquality$Ozone,
na.last = TRUE), ]
> head(airquality, 3)
   Ozone Solar.R Wind Temp Month Day
21     1       8  9.7   59     5  21
23     4      25  9.7   61     5  23
18     6      78 18.4   57     5  18
> dim(airquality)
[1] 153   6
> airquality <- airquality[order(airquality$Ozone,
na.last = NA), ]
> head(airquality, 3)
   Ozone Solar.R Wind Temp Month Day
21     1       8  9.7   59     5  21
23     4      25  9.7   61     5  23
18     6      78 18.4   57     5  18
> dim(airquality)
[1] 116   6
```

# R – concatenation/merging data frames



- merge()

```
> (df1 <- data.frame(ID = 1:3, Name = c("John", "Mary", "Tony")))
  ID Name
1  1 John
2  2 Mary
3  3 Tony


> (df2 <- data.frame(ID = 1:3, Salariu = c(1400, 1800, 1500)))(df2 <-
data.frame(ID = 1:3, Salariu = c(1400, 1800, 1500)))>
> (df2 <- data.frame(ID = 3:1, Income = c(1400, 1800, 1500)))
  ID Income
1  3   1400
2  2   1800
3  1   1500
> (df3 <- merge(df1, df2, by = "ID"))
  ID Name Income
1  1 John   1500
2  2 Mary   1800
3  3 Tony   1400
```

# R – merging 2



- !
- merge dataframe with the same number of observations
- if the two sets of data have different numbers of observations - multiply the values in a data set for the other set of data is complete

```
> (df1 <- data.frame(ID = c(1:3,2), Nume = c("John", "Peter", "Mary", "Tony")))
  ID  Nume
1  1  John
2  2 Peter
3  3  Mary
4  2  Tony
> (df2 <- data.frame(ID = c(1:3,3), Income = c(1400, 1800, 1500, 2000)))
  ID Income
1  1   1400
2  2   1800
3  3   1500
4  3   2000
> (df3 <- merge(df1, df2, by = "ID"))
  ID  Nume Income
1  1  John   1400
2  2 Peter   1800
3  2  Tony   1800
4  3  Mary   1500
5  3  Mary   2000
```

# R – finding and removing duplicate records



- duplicated() - identify all duplicated elements of a vector, and return a logical vector, the length of the verified value for all elements that are TRUE and FALSE otherwise

- unique() - returns a vector / dataframe that contains only unique values

```
> set.seed(50)
> x1 <- sample(1:100, replace = TRUE)
> duplicate0 <- x1[duplicated(x1)]
> length(duplicate0)
[1] 39
> no_duplicate0 <- x1[!duplicated(x1)]
> length(no_duplicate0)
[1] 61
> unique0 <- unique(x1)
> length(unique0)
[1] 61
> all.equal(no_duplicate0, unique0)
[1] TRUE
```

# R – levels identification



- levels()
- table()

```
> data("airquality", package = "datasets")
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
> class(airquality$Month)
[1] "integer"
> airquality$Month <- as.factor(airquality$Month)
> levels(airquality$Month)
[1] "5" "6" "7" "8" "9"
> table(airquality$Month)

 5  6  7  8  9
31 30 31 31 30
```

# R – renaming levels



```
> levels(airquality$Month) <- c("May", "Jun", "Jul",
"Aug", "Sep")
> levels(airquality$Month)
[1] "May" "Jun" "Jul" "Aug" "Sep"
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67   May   1
2    36     118  8.0   72   May   2
3    12     149 12.6   74   May   3
> tail(airquality, 3)
    Ozone Solar.R Wind Temp Month Day
151    14     191 14.3   75   Sep  28
152    18     131  8.0   76   Sep  29
153    20     223 11.5   68   Sep  30
> table(airquality$Month)

May Jun Jul Aug Sep
 31  30  31  31  30
```

# R – reorder the levels of a factor



- relevel()

Important for data analysis function

```
> levels(airquality$Month) <- c("May", "Jun", "Jul", "Aug",
"Sep")
> airquality$Month <- relevel(airquality$Month, ref = "Jun")
> levels(airquality$Month)
[1] "Jun" "May" "Jul" "Aug" "Sep"
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67   May   1
2    36     118  8.0   72   May   2
3    12     149 12.6   74   May   3
```

# R – adding and removing variables



- just add a variable
- transform()

```
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
> class(airquality$Temp)
[1] "integer"
> airquality$TempC <- round((airquality$Temp - 32) / 1.8, 2)
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day TempC
1    41     190  7.4   67     5   1 19.44
2    36     118  8.0   72     5   2 22.22
3    12     149 12.6   74     5   3 23.33
> airquality <- transform(airquality, TempCt = round((Temp - 32) / 1.8, 2))
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day TempC TempCt
1    41     190  7.4   67     5   1 19.44  19.44
2    36     118  8.0   72     5   2 22.22  22.22
3    12     149 12.6   74     5   3 23.33  23.33
> airquality$TempCt <- NULL
> head(airquality, 3)
  Ozone Solar.R Wind Temp Month Day TempC
1    41     190  7.4   67     5   1 19.44
2    36     118  8.0   72     5   2 22.22
3    12     149 12.6   74     5   3 23.33
```

# R – reordering the variables in a data frame



- by col number
- by variable name

```
> data("airquality", package = "datasets")
> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
> airquality <- airquality[c(2, 3, 4, 6, 1, 5)]
> names(airquality)
[1] "Solar.R" "Wind"    "Temp"    "Day"     "Ozone"
"Month"


> data("airquality", package = "datasets")
> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
> airquality <- airquality[c("Solar.R", "Wind", "Temp",
"Day", "Month", "Ozone")]
> names(airquality)
[1] "Solar.R" "Wind"    "Temp"    "Day"     "Month"
"Ozone"
```

# R – renaming of variables - clasic



- names()
- colnames()

```
> data("airquality", package = "datasets")
> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
> names(airquality) <- c("Ozon", "Solar.R", "Angin", "Temp",
"Bulan", "Hari")
> names(airquality)
[1] "Ozon"    "Solar.R" "Angin"   "Temp"    "Bulan"   "Hari"
> names(airquality)[3] <- "Wind"
> head(airquality, 3)
  Ozon Solar.R Wind Temp Bulan Hari
1   41     190  7.4   67     5    1
2   36     118  8.0   72     5    2
3   12     149 12.6   74     5    3
> colnames(airquality)[3] <- "Angin"
> colnames(airquality)
[1] "Ozon"    "Solar.R" "Angin"   "Temp"    "Bulan"   "Hari"
> names(airquality)
[1] "Ozon"    "Solar.R" "Angin"   "Temp"    "Bulan"   "Hari"
```

# R – renaming of variables - package



- *data.table* package
- setnames()

or

- *plyr* package
- rename()

```
> install.packages("data.table")
> library(data.table)
> setnames(airquality, "Angin", "Wind")
> names(airquality)
[1] "Ozon"    "Solar.R" "Wind"    "Temp"    "Bulan"   "Hari"


> install.packages("plyr")
> library(plyr)
> airquality <- rename(airquality, c('Bulan' = 'Month',
'Hari' = 'Day'))
> names(airquality)
[1] "Ozon"    "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

# R – How we work with databases?



- RDBMS*) → text file type → R exploration → R analysis
- Importing data in DataFrame → R exploration → R analysis

**DB → DataFrame**

*) RDBMS - Relational Database Management Systems

# Packages…Packages… R ←→ RDBMS*)



- We need an interface (DBI interface) between R and relational DBMS:
  - RJDBC package for JDBC;
  - RMySQL package for MySQL;
  - RODBC package for ODBC;
  - ROracle package for Oracle;
  - RpgSQL package for PostgreSQL;
  - RSQLite package for SQLite.

*) RDBMS - Relational Database Management Systems

# Connection via ODBC



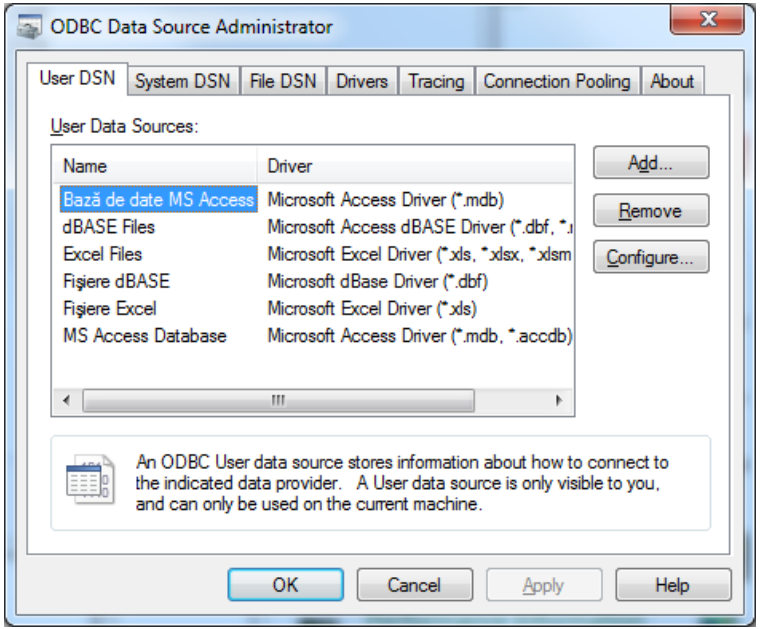
- install.packages("RODBC")

- Windows – ODBC is included in system
- Linux – the ODBC driver must be installed

# Windows – ODBC Data Source Administrator



- Control Panel -> Administration Tools menu



# RODBC - connection and metadata functions



| Function | Description | Input | Output |
|---|---|---|---|
| odbcDataSources () | Provides a list of available DSNs. | None required. | Character vector of DSNs. |
| odbcConnect (dsn, uid, pwd, ...) | Establishes a connection to a database server. | dsn="DSN_name", uid="USERID", pwd="password" - other various optional parameters | Channel object that represents an active connection to a database. |
| odbcDriverConnect (connection ="", ...) | Establishes a connection to a database server. | connection string. The values for DSN, user id, and password must be provided in a single string. | Channel object that represents an active connection to a database. |
| odbcGetInfo ( channel ) | Provides detailed information about the active database connection. | channel - object representing an active connection to a database server | Named character vector describing details about the connection including the ODBC driver type and level of conformance to the API standards. |

# RODBC - database and table metadata functions



| Function | Description | Input | Output |
|---|---|---|---|
| sqlTypeInfo (channel, ...) | Provides information about the supported data types of the ODBC database. | channel | Data frame of the supported data types and their characteristics. |
| sqlTables (channel, ...) | Provides a description of the table-like objects defined within a database. | channel recommended optional parameters : "schema=", "tableType=" | Data frame containing details about the tables, views, or other table-like objects in the database. |
| sqlColumns (channel, sqtable, ...) | Provides a description of the columns defined within a table. | channel, table name | Data frame containing details about the column names and other attributes for a table. |

# RODBC – R code



```
library(RODBC)
dsn.name <- "dsn_name"
user.name <- "guest"

con1 <- odbcConnect(dsn=dsn.name,uid=user.name,pwd)

table.list <- sqlTables(con1,tableType="TABLE", schema="DB2INST1")
cat("There are", nrow(table.list), "tables in the DB2INST1 schema.\n")

table.name <- "DB2INST1.US_FUEL_ECONOMY_AUGUST_2013"
col.list <- sqlColumns(con1,table.name)
cat("There are", nrow(col.list), "columns defined in", table.name,"\n")

# Display one row from the table
cars <- sqlFetch(con1, table.name)
print (cars[1,1:4], row.names=FALSE)

# Close connections
odbcCloseAll()
cat("Database connections are closed.\n")

---- OUTPUT from Script

There are 27 tables in the DB2INST1 schema.
There are 18 columns defined in DB2INST1.US_FUEL_ECONOMY_AUGUST_2013
 MODEL_YEAR MFR_NAME DIVISION          CARLINE
       2013      BMW      BMW 135i Convertible
Database connections are closed.
```

# RODBC - direct connection method



```
driver.name <- "{IBM DB2 ODBC DRIVER}"
db.name <- "SAMPLEDB"
host.name <- "bluforcloud.imdemocloud.com"
port <-"50001"
user.name <-"granthut"

# Use a full connection string to connect to a SAMPLE database
con.text <- paste("DRIVER=",driver.name,
                  ";Database=",db.name,
                  ";Hostname=",host.name,
                  ";Port=",port,
                  ";PROTOCOL=TCPIP",
                  ";UID=", user.name,
                  ";PWD=",pwd,sep="")

con1 <- odbcDriverConnect(con.text)
```

# RODBC - querying and deleting data



| Function | Description | Input | Output |
|---|---|---|---|
| sqlQuery ( channel, query, ... ) | Executes the SQL query on the database server and provides the results. | channel, query<br>recommended options: errors=FALSE (helps to capture any errors) | Data frame of the result set. The data will be mapped to compatible R data types. |
| sqlDrop ( channel, sqtable, ... ) | Removes the table contents and definition from the database. | channel, table | Note that this function will attempt to execute a DROP TABLE statement. |
| sqlClear ( channel, sqtable, ... ) | Removes all of the rows in a table from the database. | channel, table | Note that this function will attempt to execute a TRUNCATE TABLE statement. |

# RODBC - diagnosing errors



```
res <- sqlQuery(con1,"CREATE TABLE TESTDATA (c1 INTEGR)", errors=FALSE)
if (res == -1){
  cat ("An error has occurred.\n")
  msg <- odbcGetErrMsg(con1)
  print (msg)
} else {
  cat ("Table was created successfully.\n")
}

---- OUTPUT from Script

An error has occurred.
[1] "42704 -204 [IBM][CLI Driver][DB2/LINUXX8664]
SQL0204N  \"INTEGR\" is an undefined name.  SQLSTATE=42704\r\n"
[2] "[RODBC] ERROR: Could not SQLExecDirect 'CREATE TABLE TESTDATA (c1 INTEGR)'"
```

# RODBC - saving data



```
tab.name <- "CLASSMARKS"
NAMES <- c("Bob","Mary","Fred")
MARKS <- c(78,88,91)

# Create a data frame of test scores and names
CLASSMARKS <- data.frame (NAMES,MARKS,stringsAsFactors=FALSE)

# Create a new table and populate it with the data frame CLASSMARKS
sqlSave(con1, CLASSMARKS, rownames=FALSE,safer=FALSE)

NEWCLASS <- sqlFetch(con1,tab.name)
cat( "Mean mark for the class is", mean(NEWCLASS[,"MARKS"]),"\n")

---- OUPUT from Script

Mean mark for the class is 85.66667
```

# RODBC - stored procedures



```
# Call the stored procedure to find the median mark based on a subject (input variable)
  subject <-"MATH"
  median <- sqlQuery(con1,"CALL GETMEDIAN ( subject )")
  print (median)
```

Source: http://www.ibm.com/developerworks/data/library/techarticle/dm-1402db2andr/index.html?ca=drs

# Databases



- Data manipulation
- Queries
- Using SQL within R
- Data aggregation
- Matching

# Packages…



- sqldf
- PASWR
- ggplot2

```
# Install the package
install.packages('sqldf')

# Load the package
library(sqldf)
```

# Opening an existing dataset



```
# Load the package
library(sqldf)

# Use the titanic data set
data(titanic3, package="PASWR")
colnames(titanic3)
head(titanic3)
```

# sqldf Package



```
sqldf('select age, count(*) from
titanic3 where age is not null
group by age')
```

# sqldf Package - histogram



```
library(ggplot2)


DF <- sqldf('select age from titanic3
  where age != "NA"')


qplot(DF$age,data=DF,
  geom="histogram")
```

# sqldf Package – group by



```
DF <- sqldf('select count(*) total
  from titanic3 where age=29 group by
  survived')
DF2 <- t(DF)


colnames(DF2) <- c('Died','Survived')
```

Source: http://www.r-bloggers.com/make-r-speak-sql-with-sqldf/

# Databases



- Data manipulation
- Queries
- Using SQL within R
- Data aggregation
- Matching

# A small example



- sqldf
- PASWR
- ggplot2

```
> install.packages("sqldf")
> require(sqldf)
> myCO2 <- CO2
> head(CO2, 3)
 Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
> attributes(myCO2) <- attributes(CO2)[c("names", "row.names",
    "class")]
> class(myCO2)
> class(myCO2) <- "data.frame"
```

# Column & Row names



```
colnames(myCO2)
```

The result is a vector of character strings.

Columns = fields

# Subsetting columns



SQL

```
s01 <- sqldf("select Type, conc from myCO2")
```

R

```
r01 <- myCO2[, c("Type", "conc")]
```

Testing s01 vs. r01

```
all.equal(s01, r01)
```

# All columns



SQL

```
s02 <- sqldf("select * from myCO2")
```

R

```
r02 <- myCO2[ , ]
```

# Only one column



SQL

```
s03 <- sqldf("select Type from myCO2")
```

R

```
r03 <- myCO2[ , "Type"]
```

# Verify



```
all.equal(s03, r03)

class(r03)
# possible error
# mean function works with vector, not data.frame
mean(myCO2[, "uptake"])
mean(myCO2$uptake)
```

# Case sensitivity



SQL is **not** case-sensitive

```
s04 <- sqldf("select type, coNC from myCO2")
```

R **is** case-sensitive

```
r04 <- myCO2[, c("type", "coNC")]
```

# R extensions



```
myCO2[, c(1, 3, 5)]

# the order of the columns in an R is important
myCO2[, c(5, 2)]
# is diferent than
myCO2[, c(2, 5)]
```

# Other column selections



Column selection with logical values

```
myCO2[, c(TRUE, FALSE, FALSE, TRUE, FALSE)]
```

or

```
myCO2[, colnames(myCO2) > "d"]
```

# Subsetting rows - conditions



SQL

```
s05 <- sqldf("select * from myCO2 where uptake <
  20")
```

R

```
r05 <- myCO2[ myCO2[, "uptake"] < 20, ]
```

# Subsetting rows - with



R

```
r05w <- with(myCO2, myCO2[uptake < 20, ]) # same as
  r05
```

# Logical operators



SQL

```
s06 <- sqldf("select * from myCO2 where uptake < 20
  and Type='Quebec'")
```

R

```
r06 <- with(myCO2, myCO2[uptake < 20 & Type ==
  'Quebec', ])
```

# First few



SQL

```
s07 <- sqldf("select * from myCO2 limit 6")
```

R

```
r07 <- head(myCO2)
```

# Row names versus numbers



row names are character:

```
r06
```

select the first 3 rows:

```
r06[1:3,]
```

different from:

```
r06[c("1", "2", "3"), ]
```

doesn't work either:

```
r06[c(1, 8, 15), ]
```

# NULL



let's play:

```
r08 <- r06
r08[2:4, 1] <- NA
r08[5, 4] <- NA
```

how it looks?

```
r08
```

# Not NULL



SQL

```
s09 <- sqldf("select * from r08 where plant is not
  null")
```

R

```
r09 <- with(r08, r08[!is.na(Plant), ])
```

# Is NULL



SQL

```
s10 <- sqldf("select * from r08 where plant is
  null")
```

R

```
r10 <- with(r08, r08[is.na(Plant), ])
```

# no missing values



R

```
na.omit(r08)
```

# Quotes



In SQL single quotes are used to delimit character strings.

A single quote inside a string is given with two single quotes in a row.

In R be free to use single or double quotes.

```
c("he's", 'he\'s', "she has \"it\"")
```

The backslash is used to escape a quote character that is the same as the delimiting quote.

Source: http://www.burns-stat.com/translating-r-sql-basics/

# Databases



- Data manipulation
- Queries
- Using SQL within R
- Data aggregation
- Matching

# transpose



- t()

```
> data("airquality", package = "datasets")
> (airq <- airquality[1:5, 1:3])
  Ozon Solar.R Wind
1   41     190  7.4
2   36     118  8.0
3   12     149 12.6
4   18     313 11.5
5   NA      NA 14.3
> t(airq)
            1   2     3     4    5
Ozon     41.0  36  12.0  18.0   NA
Solar.R 190.0 118 149.0 313.0   NA
Wind      7.4   8  12.6  11.5 14.3
```

# aggregate



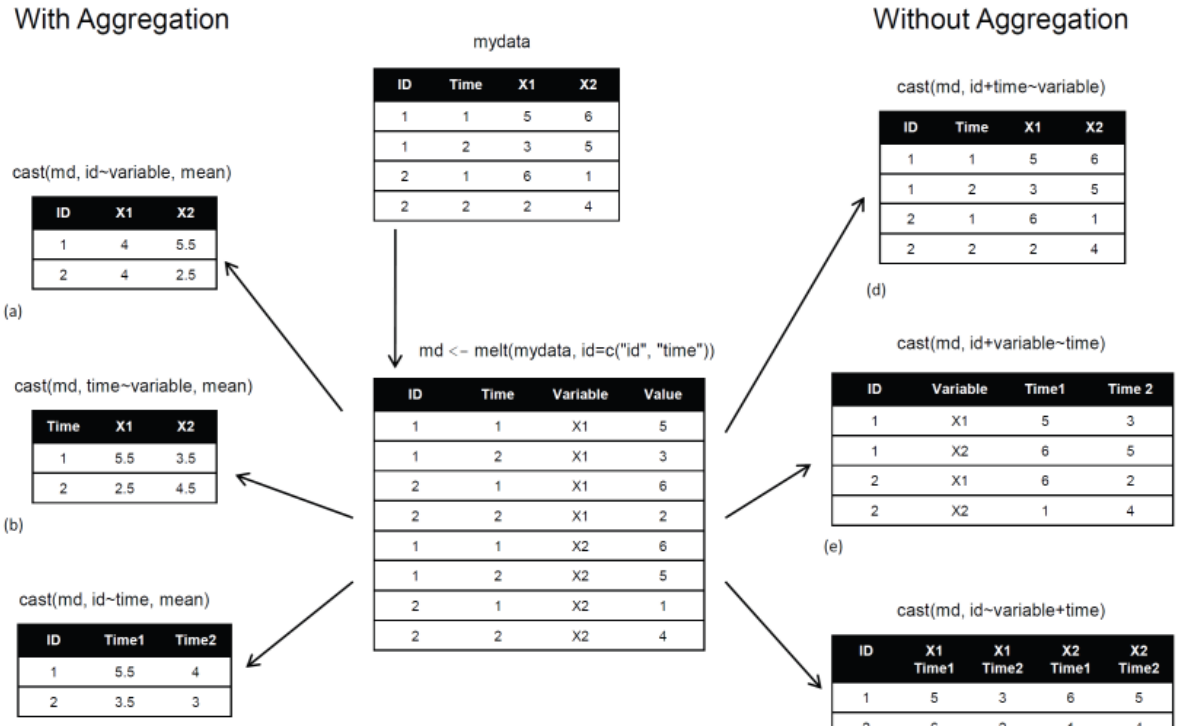
- aggregate()

```
> head((myaggdata <- aggregate(airquality$Solar.R,
    by=list(airquality$Month), FUN=mean)), 3)

  Group.1        x
1       5       NA
2       6 190.1667
3       7 216.4839
> head((myaggdata <- aggregate(airquality$Solar.R,
    by=list(airquality$Month), FUN=mean,
    na.rm=TRUE)), 3)

  Group.1        x
1       5 181.2963
2       6 190.1667
3       7 216.4839
```

```
> data("airquality", package = "datasets")
> head(airquality, 3)
  Ozon Solar.R Wind Temp Month Day
1   41     190  7.4   67     5   1
2   36     118  8.0   72     5   2
3   12     149 12.6   74     5   3
> myaggdata <- aggregate(airquality, by=list(MonthG = airquality$Month), FUN=mean,
    na.rm=TRUE)
> myaggdata
  MonthG     Ozon  Solar.R      Wind     Temp Month  Day
1      5 23.61538 181.2963 11.622581 65.54839     5 16.0
2      6 29.44444 190.1667 10.266667 79.10000     6 15.5
3      7 59.11538 216.4839  8.941935 83.90323     7 16.0
4      8 59.96154 171.8571  8.793548 83.96774     8 16.0
5      9 31.44828 167.4333 10.180000 76.90000     9 15.5


> myaggdata <- aggregate(airquality, by=list(airquality$Month), FUN=mean,
    na.rm=TRUE)
or
> attach(airquality)
> myaggdata <- aggregate(airquality, by=list(Month), FUN=mean, na.rm=TRUE)
```

# with vs. without aggregation






Source: http://www.r-statistics.com/2012/01/aggregation-and-restructuring-data-from-r-in-action/

# {reshape} package



- melt() - melting
- cast()

```
> cast_mycars <- cast(melt_mycars, model~variable)
> cast_mycars
              model  mpg cyl disp  hp
1         Datsun 710 22.8   4  108  93
2     Hornet 4 Drive 21.4   6  258 110
3 Hornet Sportabout 18.7   8  360 175
4          Mazda RX4 21.0   6  160 110
5      Mazda RX4 Wag 21.0   6  160 110
```

```
> (mycars <- mtcars[1:5, 1:4])
                   mpg cyl disp  hp
Mazda RX4         21.0   6  160 110
Mazda RX4 Wag     21.0   6  160 110
Datsun 710        22.8   4  108  93
Hornet 4 Drive    21.4   6  258 110
Hornet Sportabout 18.7   8  360 175
> mycars$model <- row.names(mycars)
> row.names(mycars) <- NULL
> melt_mycars <- melt(mycars, id=(c("model")))
> melt_mycars
               model variable value
1          Mazda RX4      mpg  21.0
2      Mazda RX4 Wag      mpg  21.0
3         Datsun 710      mpg  22.8
4     Hornet 4 Drive      mpg  21.4
5  Hornet Sportabout      mpg  18.7
6          Mazda RX4      cyl   6.0
7      Mazda RX4 Wag      cyl   6.0
8         Datsun 710      cyl   4.0
9     Hornet 4 Drive      cyl   6.0
10 Hornet Sportabout      cyl   8.0
11         Mazda RX4     disp 160.0
12     Mazda RX4 Wag     disp 160.0
13        Datsun 710     disp 108.0
...
```

# {reshape} example



- melt()
- cast()

- but there is much more:
  http://had.co.nz/reshape/introduction.pdf

```
> (mydata <- data.frame(id=c(1,1,2,2), time=c(1,2,1,2),
    x1=c(5,3,6,2), x2=c(6,5,1,4)))
  id time x1 x2
1  1    1  5  6
2  1    2  3  5
3  2    1  6  1
4  2    2  2  4
> (melt_data <- melt(mydata, id=c("id", "time")))
  id time variable value
1  1    1       x1     5
2  1    2       x1     3
3  2    1       x1     6
4  2    2       x1     2
5  1    1       x2     6
6  1    2       x2     5
7  2    1       x2     1
8  2    2       x2     4
> (idmeans <- cast(melt_data, id~variable, mean))
  id x1  x2
1  1  4 5.5
2  2  4 2.5
> (timemeans <- cast(melt_data, time~variable, mean))
  time  x1  x2
1    1 5.5 3.5
2    2 2.5 4.5
```

# Databases



- Data manipulation
- Queries
- Using SQL within R
- Data aggregation
- Matching

# match()

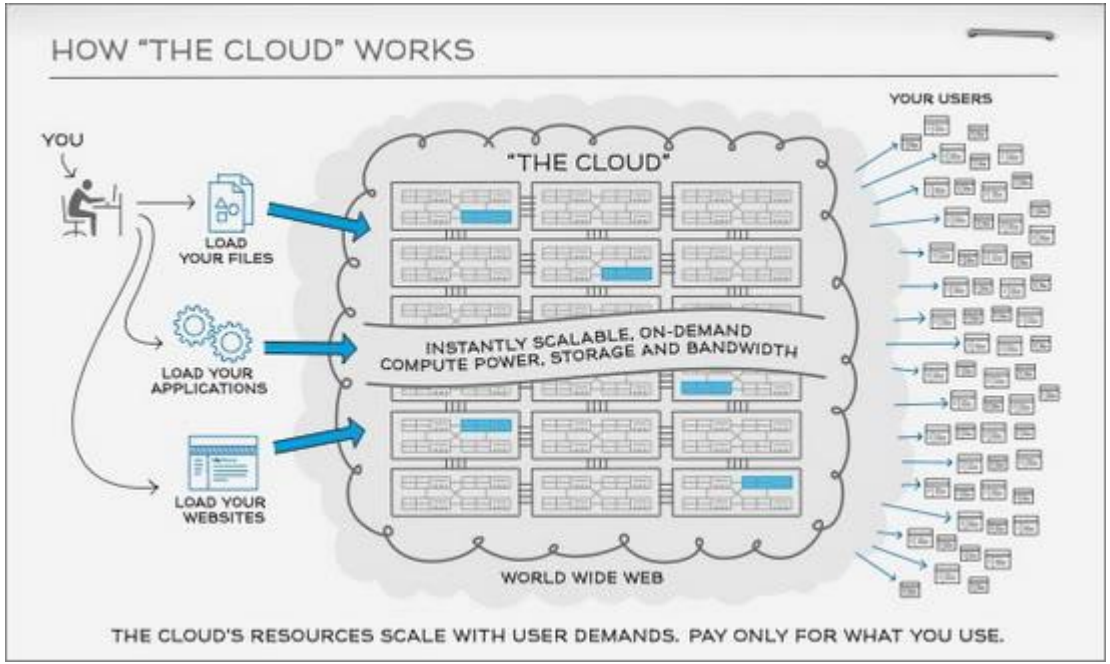


```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
> mycars <- mtcars
> mycars$model <- row.names(mycars)
> row.names(mycars) <- NULL
> head(mycars)
   mpg cyl disp  hp drat    wt  qsec vs am gear carb              model
1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4          Mazda RX4
2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4      Mazda RX4 Wag
3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1         Datsun 710
4 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1     Hornet 4 Drive
5 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 Hornet Sportabout
6 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1           Valiant
> a <- mycars[ , c(1:5,12)]
> b <- mycars[ , c(6:11,12)]
```

```
> head(a, 3)
  mpg cyl disp  hp drat             model
1  21   6  160 110 3.90         Mazda RX4
2  21   6  160 110 3.90     Mazda RX4 Wag
3  23   4  108  93 3.85        Datsun 710
> head(b, 3)
     wt  qsec vs am gear carb             model
1 2.620 16.46  0  1    4    4         Mazda RX4
2 2.875 17.02  0  1    4    4     Mazda RX4 Wag
3 2.320 18.61  1  1    4    1        Datsun 710
> b <- b[order(b$model) , ]
> head(b, 3)
      wt  qsec vs am gear carb              model
23 3.435 17.30  0  0    3    2        AMC Javelin
15 5.250 17.98  0  0    3    4 Cadillac Fleetwood
24 3.840 15.41  0  0    3    4         Camaro Z28
> head((merge(a, b, by="model")), 3)
                model  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1        AMC Javelin 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
2 Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
3         Camaro Z28 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
```

# Thank you!






Ciprian Alexandru
alexcipro@yahoo.com