Introduction to R

Stephane Rion, Big Data Partnership

02/08/2015

Overview

- A brief presentation
- Getting data in R and some useful packages
- Basic functions for data manipulation
- Dataframes
- Simple statistics
- More advanced modelling techniques
- Visualisation
- Useful links and external ressources

What is R ?

- Statistical programming language descendant of S
- Open source scripting language
- More than 6000 packages !
- Comes as a standalone program or can be run within an IDE (RStudio)

How to run R scipts

- Run in R console using source: source('script.R')
- From the command line using Rscript: Rscript script.R

Explore your environment

- Use *dir* to list the files in your current directory
- *getwd* / *setwd* to change your working folder
- *ls* to list object in curent R session
- *remove* or *rm* to delete a variable
- ? followed by a command name to get help

Getting data into R (1)

Let’s load a CSV file:

```
setwd(’odi_course_082015’)
visits <- read.table(’london_visits.csv’,row.names=1,sep=’
visits <- read.csv(’london_visits.csv’,row.names=1)
```

Getting data into R (2)

```r
install.packages('RODBC')
library(ODBC)
con <- odbcConnect("dbss")
sqlQuery(con, paste("SELECT * FROM School"))
```

But there is plenty of other types of data source accessible: ElasticSearch, MongoDB etc.

Some basic commands (1)

```r
head(visits,2)
```

```
##           Y2008 Y2009 Y2010 Y2011 Y2012 Y2013 Y2014p
## Argentina  20.1  55.4  52.4  61.1  93.7  93.5   92.5
## Australia 583.7 571.2 624.3 681.3 596.9 687.3  614.4
```

```r
tail(visits,2)
```

```
##         Y2008   Y2009   Y2010   Y2011   Y2012   Y2013   Y
## USA    1907.9  1839.1  1765.6  1842.8  1862.3  1877.9   1
## Total 14753.0 14211.3 14705.5 15289.5 15460.9 16810.8 17
```

Some basic commands (2)

```
summary(t(visits)[,1:5])
```

```
##    Argentina       Australia        Austria          Bahr
##  Min.   :20.10   Min.   :571.2   Min.   :141.7   Min.   
##  1st Qu.:53.90   1st Qu.:590.3   1st Qu.:144.6   1st Qu.
##  Median :61.10   Median :614.4   Median :153.0   Median 
##  Mean   :66.96   Mean   :622.7   Mean   :154.8   Mean   
##  3rd Qu.:93.00   3rd Qu.:652.8   3rd Qu.:165.4   3rd Qu.
##  Max.   :93.70   Max.   :687.3   Max.   :168.8   Max.   
##                                                  NA's   
##     Belgium     
##  Min.   :307.7  
##  1st Qu.:378.1  
##  Median :470.3  
##  Mean   :434.9  
##  3rd Qu.:489.5  
##  Max.   :530.7  
##
```

R main data types (1)

- Vector

```
v <- c(1,-1,10,8,-3.5)
print(v)
```

```
## [1]  1.0 -1.0 10.0  8.0 -3.5
```

- Character (string)

```
s <- 'hello world'
```

- Boolean

```
(1 == 2)
```

```
## [1] FALSE
```

R main data types (2)

- List

```
l <- list(1,2,3,4)
l[[1]]
```

- Matrix, Dataframes

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

```
## [1] 1 4 7
```

We can always convert a matrix to a dataframe:

```
df <- data.frame(m)
```

R main data types (3)

```
myData <- data.frame(time=c("2014-01-23 14:28:21","2014-01-
                              "2014-01-23 14:29:02","2014-0
                     speed=c(2.0,2.2,3.4,5.5))

myData
```

```
##                  time speed
## 1 2014-01-23 14:28:21   2.0
## 2 2014-01-23 14:28:55   2.2
## 3 2014-01-23 14:29:02   3.4
## 4 2014-01-23 14:31:18   5.5
```

Dataframe manipulation

- Most used object type in R to represent data
- Values in a dataframe can have several types
- Easy to manipulate using vectorized functions

Let's have a closer look

```
data(mtcars)
unique(row.names(mtcars))
```

```
##  [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsu
##  [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valia
##  [7] "Duster 360"          "Merc 240D"           "Merc 2
## [10] "Merc 280"            "Merc 280C"           "Merc 4
## [13] "Merc 450SL"          "Merc 450SLC"         "Cadil
## [16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 1
## [19] "Honda Civic"         "Toyota Corolla"      "Toyota
## [22] "Dodge Challenger"    "AMC Javelin"         "Camar
## [25] "Pontiac Firebird"    "Fiat X1-9"           "Porsc
## [28] "Lotus Europa"        "Ford Pantera L"      "Ferra
## [31] "Maserati Bora"       "Volvo 142E"
```

```r
mtcars[mtcars$mpg<15,]
```

```
##                      mpg cyl disp  hp drat    wt  qsec v
## Duster 360          14.3   8  360 245 3.21 3.570 15.84
## Cadillac Fleetwood  10.4   8  472 205 2.93 5.250 17.98
## Lincoln Continental 10.4   8  460 215 3.00 5.424 17.82
## Chrysler Imperial   14.7   8  440 230 3.23 5.345 17.42
## Camaro Z28          13.3   8  350 245 3.73 3.840 15.41
```

```r
mtcars[grep('Toy',row.names(mtcars)),]
```

```
##                 mpg cyl  disp hp drat    wt  qsec vs am
## Toyota Corolla 33.9   4  71.1 65 4.22 1.835 19.90  1  1
## Toyota Corona  21.5   4 120.1 97 3.70 2.465 20.01  1  0
```

```r
max(mtcars$cyl)
```

```
## [1] 8
```

Manipulating dataframes with plyr

```
library(plyr)
census <- read.csv('odi_course_082015/census.data',sep=',',
colnames(census) <- c('X,','age','work_class','fnlwgt','edu
ddply(census,"race",nrow)
```

```
##                 race    V1
## 1  Amer-Indian-Eskimo   311
## 2  Asian-Pac-Islander  1039
## 3               Black  3124
## 4               Other   271
## 5               White 27816
```

Now with custom function declaration

```r
library(plyr)
myfunction <- function(x){
df <- summary(x$income)/nrow(x)
return(df)
}
ddply(census,"education",myfunction)
```

```
##       education     <=50K       >50K
## 1          10th 0.9335477 0.06645230
## 2          11th 0.9489362 0.05106383
## 3          12th 0.9237875 0.07621247
## 4       1st-4th 0.9642857 0.03571429
## 5       5th-6th 0.9519520 0.04804805
## 6       7th-8th 0.9380805 0.06191950
## 7           9th 0.9474708 0.05252918
## 8    Assoc-acdm 0.7516401 0.24835989
## 9     Assoc-voc 0.7387844 0.26121563
## 10    Bachelors 0.5852474 0.41475257
```

Built-in statistical functions

- *mean*, *sd*, *quantile*, *cor* etc.

```
mean(c(1,2,-1,3))
```

```
## [1] 1.25
```

- Statistical tests: z-test, t-test, chi2

Ex of t-test:

```
x <- c(175, 168, 168, 190, 156, 181, 182, 175, 174, 179)
y <- c(185, 169, 173, 173, 188, 186, 175, 174, 179, 180)
t.test(x,y, var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  x and y
## t = -0.9474, df = 18, p-value = 0.356
```

Basic Probability Distributions (1)

- Normal, binomial, Chi-Squared and t distribution For every distribution there are four commands. The commands for each distribution are prepended with a letter to indicate the functionality:
- "d" returns the height of the probability density function
- "p" returns the cumulative density function
- "q" returns the inverse cumulative density function (quantiles)
- "r" returns randomly generated numbers

Example for the Normal distribution¶

```
dnorm(0)
```

```
## [1] 0.3989423
```

```
v <- c(0,1,2)
dnorm(v)
```

```
## [1] 0.39894228 0.24197072 0.05399097
```

Linear regression

```
df1 <- as.data.frame(state.x77)
colnames(df1)[4] = "Life.Exp"
colnames(df1)[6] = "HS.Grad"
df1[,9] = df1$Population * 1000 / df1$Area
colnames(df1)[9] = "Density"
cor(df1)
```

```
##              Population     Income   Illiteracy     Life.E
## Population   1.00000000  0.2082276  0.107622373 -0.068051
## Income       0.20822756  1.0000000 -0.437075186  0.340255
## Illiteracy   0.10762237 -0.4370752  1.000000000 -0.588477
## Life.Exp    -0.06805195  0.3402553 -0.588477926  1.000000
## Murder       0.34364275 -0.2300776  0.702975199 -0.780845
## HS.Grad     -0.09848975  0.6199323 -0.657188609  0.582216
## Frost       -0.33215245  0.2262822 -0.671946968  0.262068
## Area         0.02254384  0.3633154  0.077261132 -0.107331
## Density      0.24622789  0.3299683  0.009274348  0.091061
##                 HS.Grad       Frost          Area        Den
```

The full model

```
full_model = lm(Life.Exp ~ Population + Income + Illiteracy
summary(full_model)
```

```
##
## Call:
## lm(formula = Life.Exp ~ Population + Income + Illiteracy
##     HS.Grad + Frost + Area + Density, data = df1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47514 -0.45887 -0.06352  0.59362  1.21823
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.995e+01  1.843e+00  37.956  < 2e-16 ***
## Population    6.480e-05  3.001e-05   2.159   0.0367 *
## Income        2.701e-04  3.087e-04   0.875   0.3867
## Illiteracy    3.029e-01  4.024e-01   0.753   0.4559
```

Updated model

```
model2 = update(full_model, .~.-Area)
summary(model2)
```

```
##
## Call:
## lm(formula = Life.Exp ~ Population + Income + Illiteracy
##     HS.Grad + Frost + Density, data = df1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.50252 -0.40471 -0.06079  0.58682  1.43862
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.094e+01  1.378e+00  51.488  < 2e-16 ***
## Population    6.249e-05  2.976e-05   2.100   0.0418 *
## Income        1.485e-04  2.690e-04   0.552   0.5840
## Illiteracy    1.452e-01  3.512e-01   0.413   0.6814
```

Classification with logistic regression (1)

```
wine_data <- read.table(file = 'odi_course_082015/winequal
head(wine_data)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.su
## 1           7.4             0.70        0.00
## 2           7.8             0.88        0.00
## 3           7.8             0.76        0.04
## 4          11.2             0.28        0.56
## 5           7.4             0.70        0.00
## 6           7.4             0.66        0.00
##   free.sulfur.dioxide total.sulfur.dioxide density   pH
## 1                  11                   34  0.9978 3.51
## 2                  25                   67  0.9968 3.20
## 3                  15                   54  0.9970 3.26
## 4                  17                   60  0.9980 3.16
## 5                  11                   34  0.9978 3.51
## 6                  13                   40  0.9978 3.51
##   quality
```

Classification with logistic regression (2)

```
cmodel1 <- glm(quality ~ .,data = wine_data,family ="poisso
summary(cmodel1)
```

```
##
## Call:
## glm(formula = quality ~ ., family = "poisson", data = wi
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.23430  -0.15584  -0.02713   0.18619   0.81124
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>
## (Intercept)           3.6537989 13.6703607   0.267  0.78
## fixed.acidity         0.0036583  0.0166334   0.220  0.82
## volatile.acidity     -0.1977012  0.0803898  -2.459  0.01
## citric.acid          -0.0359233  0.0961410  -0.374  0.70
## residual.sugar        0.0026177  0.0097360   0.269  0.78
```

Advanced methods for classification

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
wine_data$quality <- factor(wine_data$quality)
inTraining <- createDataPartition(wine_data$quality, p = .7
inTraining[1:30]
```

```
##  [1]  2  3  4  5  6  8 12 13 14 15 16 17 18 19 20 21 22
## [24] 29 31 32 33 34 35 37
```

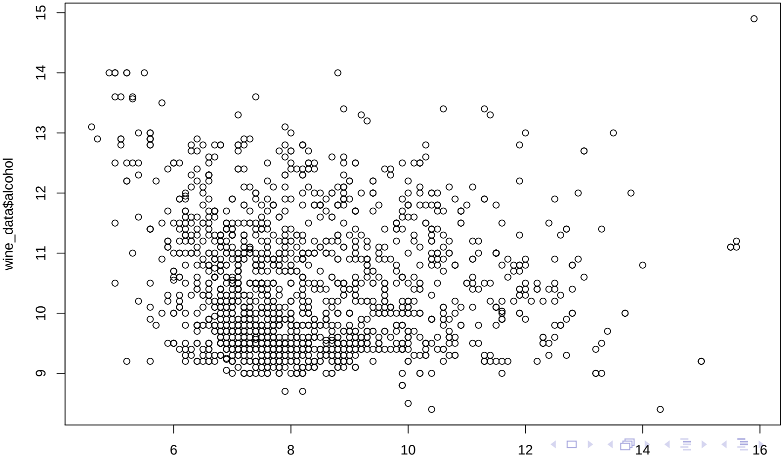
```
xtrain <- wine_data[ inTraining,]
xtest  <- wine_data[-inTraining,]
#cross validation
cv <- trainControl(method = "cv",number = 10)

tree_model <- train(quality ~ .,
```

Visualisation (1)

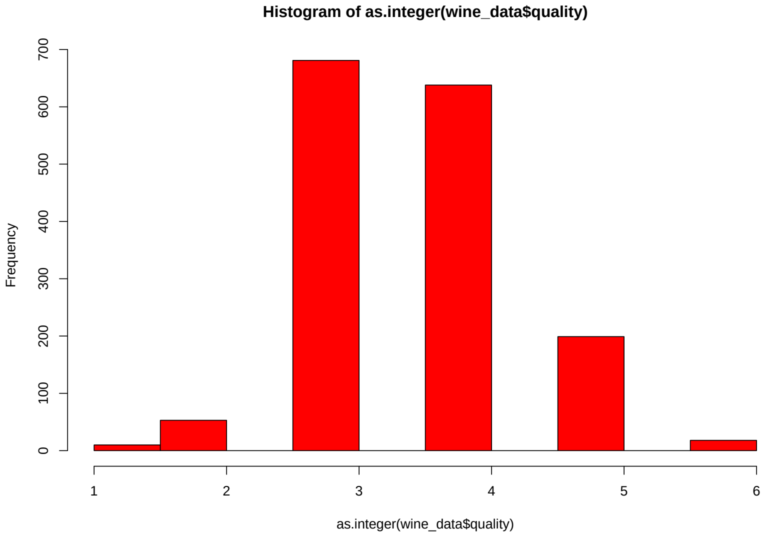
- Simple graphs can be achievd using plot and lines function

```
plot(wine_data$fixed.acidity,wine_data$alcohol)
```



- Histograms, density, quantile to quantile etc.

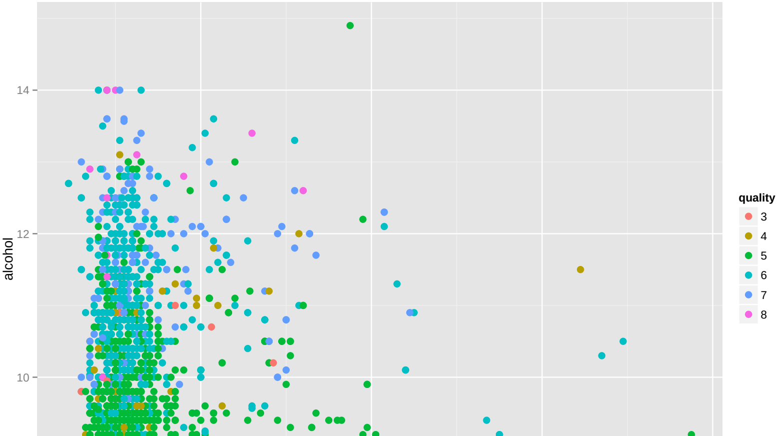
```
hist(as.integer(wine_data$quality),breaks = 12,col='red')
```



Visualisation (2)

- ggplot2 (`http://ggplot2.org`)

```
library(ggplot2)
p <- ggplot(data=wine_data)
p + geom_point(size=3) + aes(residual.sugar,alcohol, colour
```



And books

- An Introduction To Statistical Learning
- Practical Data Science with R
- R Graphics Cookbook
- The Art of R Programming