Introduction to R, Session 1

MRC Clinical Sciences Centre

http://mrccsc.github.io/r_course/introToR_Session1.html

Introduction to R, Session 1

author: MRC Clinical Sciences Centre date:http://mrccsc.github.io/r_course/introToR_Session1.html width: 1440 height: 1100 autosize: true font-import: font-family: 'Slabo 27px', serif; css:style.css

Overview

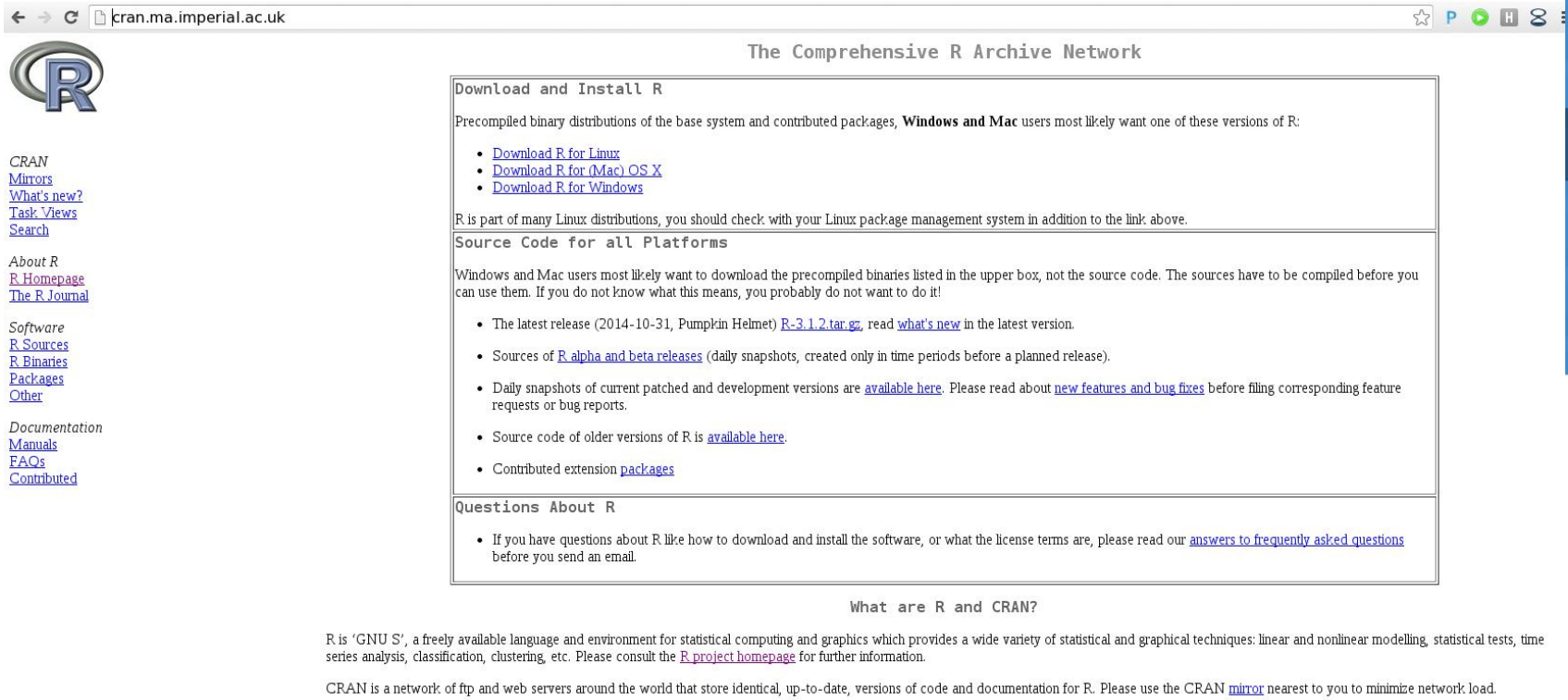
- Background to R
- Data types in R
- Reading and writing data in R
- Plotting in R
- Statistics in R

Background to R

type:section id: background

What is R?

R is a scripting language and environment for **statistical**



A quick tour of RStudio

left: 30% Four main panels - Scripting panel - R interface - Environment and history - Files, directories and help

Let's load RStudio and take a look *

rbind() functions to bind to a matrix as rows.

```
newerMatrix <- rbind(newMatrix,z)
newerMatrix
```

```
   x  y
   1 11
   2 12
   3 13
   4 14
   5 15
   6 16
   7 17
   8 18
   9 19
  10 20
z 21 22
```

Matrices (4/12) - Joining incompatable vectors and matrices

When creating a matrix using **cbind()** or **matrix()** from

For **rbind()** function, the longer vector is clipped.

```
recycledMatrix3 <- rbind(recycledMatrix2,c(1:5))
recycledMatrix3
```

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    1
[4,]    1    2
```

Matrices (5/12) - Column and row names

As with vectors, matrices can be named. For matrices the naming is done by columns and rows using **colnames()** and **rownames()** functions.

```
namedMatrix <- matrix(1:10,ncol=5,nrow=2)
colnames(namedMatrix) <- paste("Column",1:5,sep="_")
rownames(namedMatrix) <- paste("Row",1:2,sep="_")
```

```
narrowMatrix[narrowMatrix[,1] < 5,]
```

```
      Column_1 Column_2
Row_1        1        6
Row_2        2        7
Row_3        3        8
Row_4        4        9
```

Matrices (10/12) - Arithmetic operations.

As with vectors, matrices can have arithmetic operations applied to cells,rows, columns or the whole matrix

```
narrowMatrix
```

```
      Column_1 Column_2
Row_1        1        6
Row_2        2        7
Row_3        3        8
```

We can also use order to arrange multiple columns in a data frame by providing multiple vectors to order() function. Ordering will be performed in order of arguments.

```
dfExample[order(dfExample$Type,
                dfExample$Survival,
                decreasing=T),]
```

```
      Name   Type Survival_Time
3 patientX   male             2
1 patientX   male             1
2 patient2 female            30
4 patient4 female            20
```

Data frames (11/12) - Merging data frames

A common operation is to join two data frames by a column of common values.

```
dfExample <- data.frame(Name=patientName,
```

```
dfExample2 <- data.frame(Name=patientName[1:3],
                         height=c(6.1,5.1,5.5))

dfExample2
```

```
      Name height
1 patient1    6.1
2 patient2    5.1
3 patient3    5.5
```

Data frames (12/12) - Merging data frames with merge()

To do this we can use the **merge()** function with the data frames as the first two arguments. We can then specify the columns to merge by with the **by** argument. To keep only data pertaining to values common to both data frames the **all** argument is set to TRUE.

```
mergedDF <- merge(dfExample,dfExample2,by=1,all=F)
mergedDF
```

Bar Charts

Let's start with a simple bar chart graphing the treatment vector:
Plot treatment

```
barplot(treatment)
```

##

Let's now read the data from the example.txt data file, add labels, blue borders around the bars, and density lines:

Read values from tab-delimited example.txt

```
data <- read.table("data/example.txt", header=T, sep="\t")
```

Plot treatment with specified labels for axes. Use blue borders and diagonal lines in bars.

```
barplot(data$treatment, main="Treatment", xlab="Days",ylab=
```

Histograms

Let's start with a simple histogram plotting the distribution of the treatment vector:

Create a histogram for treatment

```
hist(treatment)
```

##

Concatenate the three vectors

```
all <- c(data$control, data$treatment)
```

Create a histogram for data in light blue with the y axis ranging from 0-10

```
hist(all, col="lightblue", ylim=c(0,10))
```

Now change the breaks so none of the values are grouped together and flip the y-axis labels horizontally.

Compute the largest value used in the data

```
max_num <- max(all)
```

Create a histogram for data with fire colors, set breaks so each number is in its own group, make x axis range from 0-max_num, disable right-closing of cell intervals, set heading, and make y-axis labels horizontal. ##

```
hist(all, col=heat.colors(max_num), breaks=max_num, xlim=c(
main="Histogram", las=1)
```

breaks: a single number giving the number of cells for the histogram, An open interval does not include its endpoints, and is indicated with parentheses.

For example (0,1) means greater than 0 and less than 1.

Combining Plots

R makes it easy to combine multiple plots into one overall graph, using either the par( ) or layout( ) function.
With the par( ) function, you can include the option mfrow=c(nrows, ncols) to create a matrix of nrows x ncols plots that are filled in by row. mfcol=c(nrows, ncols) fills in the matrix by columns.

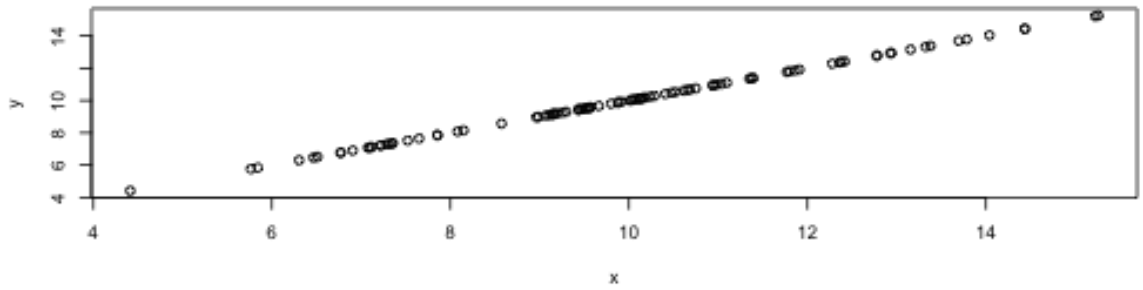
Define a layout with 2 rows and 2 columns

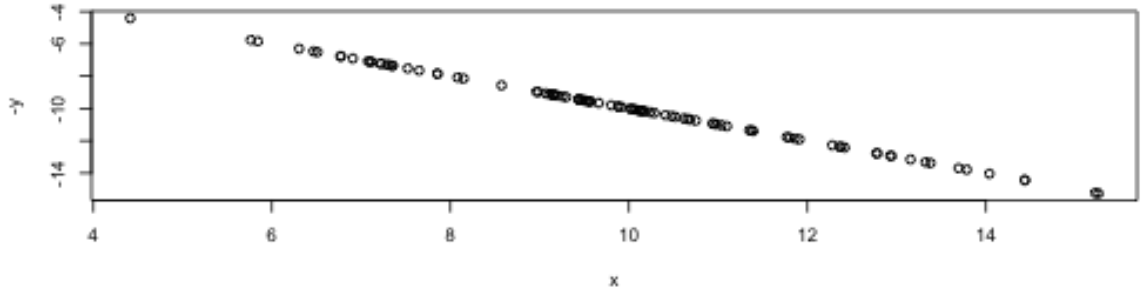
```
par(mfrow=c(2,2))
```

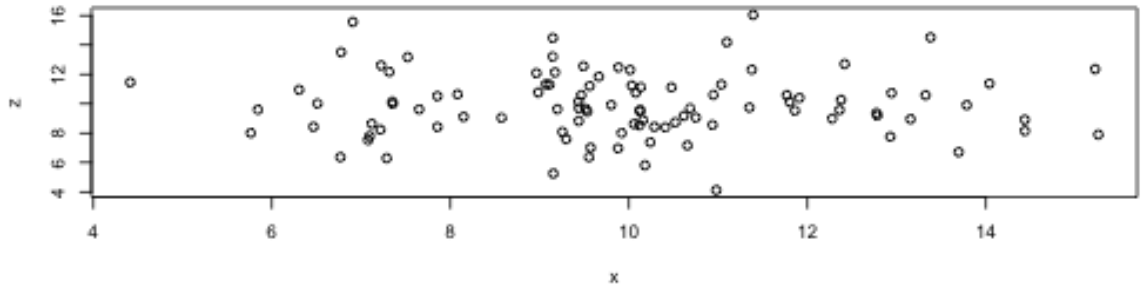
Here, we will use different dataset with two columns each for treated and untreated samples.

```
data1 <- read.table("data/gene_data.txt", header=T, sep="\t
head(data1)
```

```
      ensembl_gene_id Untreated1 Untreated2  Treated1   Trea
```







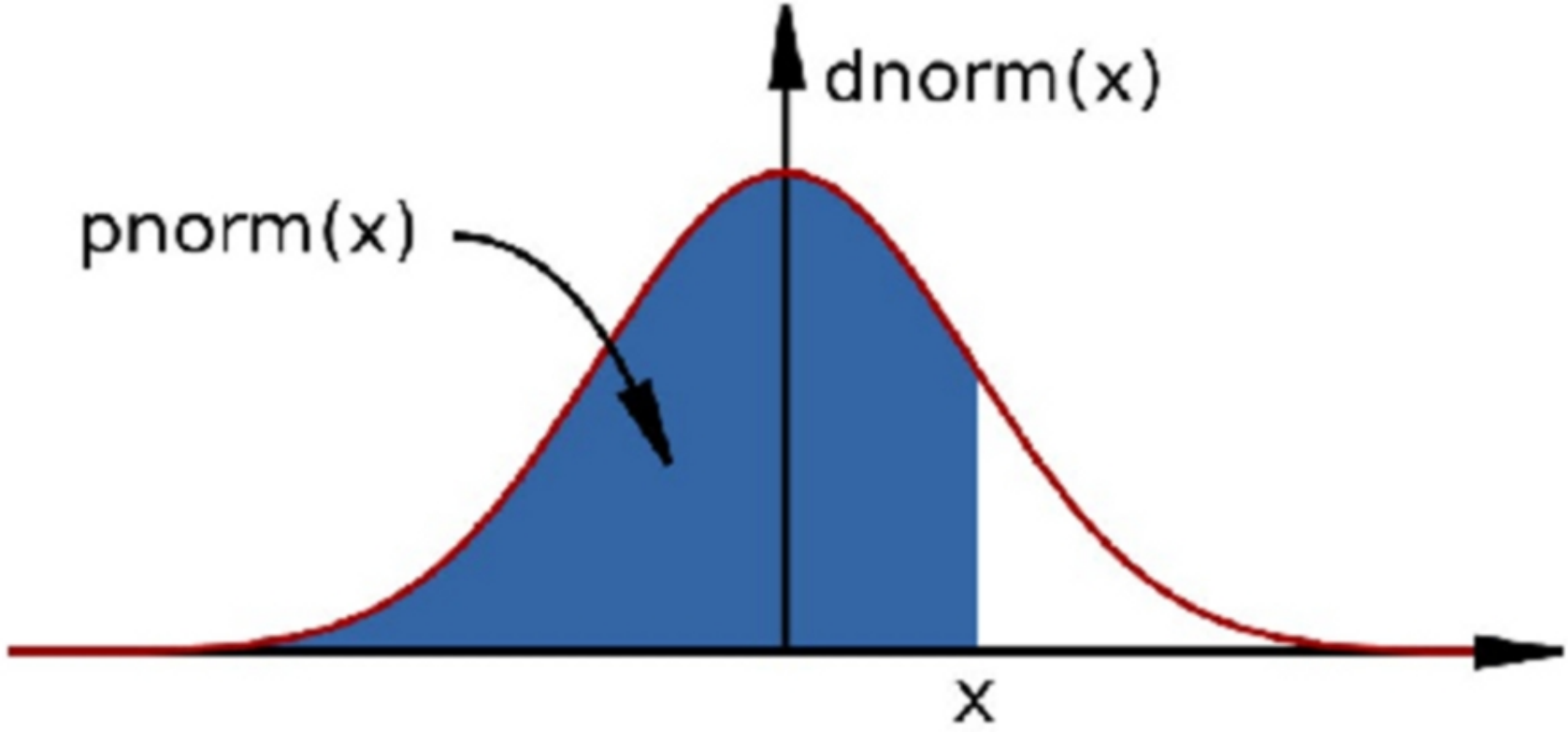
Statistics (8/26) - Correlation over a matrix

left: 70% Often we wish to apply correlation analysis to all columns or rows in a matrix in a pair-wise manner. To do this in R, we can

Statistics (10/26) - Distributions

R comes with functions for extracting information from most common distibutions types. An example of standard R functions for dealing with distibution can be seen here using the normal distributions.

- pnorm - cumulative distribution for x
- qnorm - inverse of pnorm (from probability gives x)
- dnorm - distribution density
- rnorm - random number from normal distribution



Statistics (11/26) - Many distributions available.

Similar functions are available for other distibution types including: - pbinom (binomial) - pnbinom (negative binomial), - phyper (hypergeometric) - pt (T distribution)

Statistics (12/26) - Distribution examples

We can use rnorm to generate random values following a normal

The **residuals** are the difference between the predicted and actual values. To retrieve the residuals we can access the slot or use the resid() function.

```
> summary(resid(lmResult))
```

```
    Min.  1st Qu.   Median     Mean 3rd Qu.     Max.
-5.0150 -2.3690 -0.2079  0.0000  2.6070  5.0540
```

```
> summary(lmResult$residual)
```

```
    Min.  1st Qu.   Median     Mean 3rd Qu.     Max.
-5.0150 -2.3690 -0.2079  0.0000  2.6070  5.0540
```

Ideally you would want your residuals to be normally distributed around 0.

Statistics (25/26) - R-squared