Exercises

 Data Management - Summary statistics - Graphics

Probability distributions - Inference introduction

GLM and Mixed Models model fitting – Model selection

RHistory

S – S-Plus - R

glm: generalized linear models

90-ies: glim (Glim for Ecologists, Crawley 1993)

90-ies: SAS versus S-Plus: Superb manuals available for SAS

Specialized software increasingly often embedded into R

ASReml - Bioconductor.org – phyloconductor

Business:
http://www.revolutionanalytics.com/

Installing R

- Available for all common operating systems

- Use the R console for pasting or typing commands,
  or install an R gui (R commander) or an IDE (RStudio)

- Linux: Use command line or Emacs and ESS http://ess.r-project.org/

- Easy to use in conjunction with a good text editor

Tinn-R
Textpad

- http://how-to.linuxcareer.com/running-gnu-r-on-linux-operating-system

Installing additional Packages - Libraries

- Some come pre-installed with the base installation

- Use the menu

- With the appropriate commands

help(install.packages)

- If you don't have administrator rights, use different methods, e.g.

http://lamages.blogspot.fr/2012/04/installing-r-packages-without-admin.html

Getting help

```
library(MASS)
```

on a function of which you know the name:

```
help(stepAIC)
?stepAIC

help(install.packages)
```

when you only know a related word, you can search the help files:

```
help.search("Bayesian")
??Bayesian
```

read http://www.jstatsoft.org/ for overviews of new libraries (package) and their capabilities

Reading in data
Before you start: go to a working directory specific to your project using the setwd() command

http://www.r-bloggers.com/read-excel-files-from-r/

A straightforward way to do it: Save your excel spreadsheet as a tab-separated text file. Then run this type of command:

```
worms<-read.table("worms.txt",header=T,row.names=1)
```

When your file has no row names: `worms2<-read.table("worms.txt",header=T)`

Another straightforward way to do it: Save your excel spreadsheet as a comma-separated file. Then run this type of command:

```
worms<-read.csv("worms.csv")
```

Checking whether read.table() has worked correctly

```
summary(worms)
dim(worms)
[1] 20  7

head(worms)
```

	Area	Slope	Vegetation	Soil.pH	Damp	Worm.density	Area2
Nashs.Field	3.6	11	Grassland	4.1	FALSE	4	0.036
Silwood.Bottom	5.1	2	Arable	5.2	FALSE	7	0.051
Nursery.Field	2.8	3	Grassland	4.3	FALSE	2	0.028
Rush.Meadow	2.4	5	Meadow	4.9	TRUE	5	0.024
Gunness.Thicket	3.8	0	Scrub	4.2	FALSE	6	0.038
Oak.Mead	3.1	2	Grassland	3.9	FALSE	2	0.031

```
tail(worms)
```

R will assign properties to the variables (numeric - factor - character) which you can modify yourself

Tables

```
table(worms$Vegetation,worms$Damp)
```

	FALSE	TRUE
Arable	3	0
Grassland	8	1
Meadow	0	3
Orchard	1	0
Scrub	2	2

To get an overview of the data, to look at associations

$

Access variables:

```
> worms$Area
 [1] 3.6 5.1 2.8 2.4 3.8 3.1 3.5 2.1 1.9 1.5 2.9 3.3 3.7 1.8 4.1 3.9 2.2 4.4 2.9
[20] 0.8
```

Create new variables:

```
worms$Area2<-worms$Area/100
```

Accessing variables:

```
names(worms)
[1] "Area"        "Slope"       "Vegetation"  "Soil.pH"     "Damp"
[6] "Worm.density" "Area2"

Area2
Error: Object "Area2" not found

worms$Area2
 [1] 0.036 0.051 0.028 0.024 0.038 0.031 0.035 0.021 0.019 0.015 0.029 0.033
[13] 0.037 0.018 0.041 0.039 0.022 0.044 0.029 0.008
```

Data frames are like boxes with variables, attach() makes the box transparent:

```
attach(worms)

Area2
 [1] 0.036 0.051 0.028 0.024 0.038 0.031 0.035 0.021 0.019 0.015 0.029 0.033
[13] 0.037 0.018 0.041 0.039 0.022 0.044 0.029 0.008
```

Correcting values, accessing values

c() stands for column vector, it is considered to be a function with a variable number of arguments

```
x<-c(3,4,1)
x
[1] 3 4 1
```

[] can be used to access elements of a vector or array

```
x[2]<-6
x
[1] 3 6 1
```

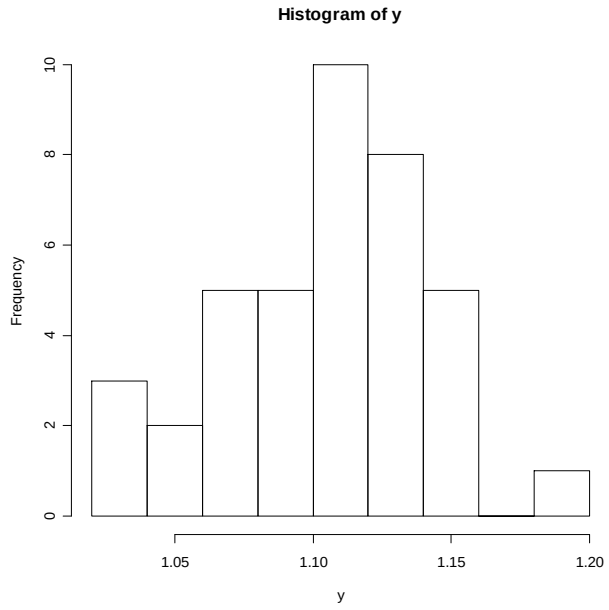
```
worms[,1:3]
worms[c(1,3,5:8),]
worms$Area[2]
```

Writing functions

When a procedure is reused on different variables

```
yvals<-read.table("yvalues.txt",header=T)
attach(yvals)

hist(y)
```



Write your own functions.
General structure:

name <- function(*arguments*){*commands*}

To calculate the median

```
mymedian<-function(x) {
+ length<-length(x)
+ odd.even<-length%%2
+ sorted<-sort(x)
+ if (odd.even == 0)  (sorted[length/2]+sorted[1+ length/2])/2
+ else  sorted[ceiling(length/2)]
+ }

mymedian(y)
[1] 1.108847

median(y)
[1] 1.108847
```

When to use ( ) or [ ] or { }

- ( ) in equations, for function arguments

- [ ] when accessing elements of objects by pointing at their index
- [[ ]] when accessing elements of **lists** by pointing at their index

- { } when enclosing multiple instructions in functions

A brief overview of what R can do

- S is a programming language designed to be used for statistical analysis
- You can use it for other purposes but it is often not very fast
- New methods are continually added as libraries/packages
- There is limited validation of the code - don't hesitate to check it yourself

Hypothesis tests

Most commonly used tests are already programmed

You have to check the formatting of your input data to be sure the test is executed correctly

In many cases the code behind a "familiar" test is not visible.

Hypothesis tests

```
t.test.data<-read.table('t.test.data.txt',header=T)
attach(t.test.data)
names(t.test.data)
[1] "gardenA" "gardenB"

t.test(gardenA,gardenB)

        Welch Two Sample t-test

data:  gardenA and gardenB
t = -3.873, df = 18, p-value = 0.001115
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.0849115 -0.9150885
sample estimates:
mean of x mean of y
        3         5
```

```
wilcox.test(gardenA,gardenB)

        Wilcoxon rank sum test with continuity correction

data:  gardenA and gardenB
W = 11, p-value = 0.002988
alternative hypothesis: true mu is not equal to 0

Warning message:
cannot compute exact p-value with ties in: wilcox.test.default(gardenA, gardenB)
```

See the structure of the returned object

```
str(wilcox.test(gardenA,gardenB))
 $ statistic  : Named num 11
  ..- attr(*, "names")= chr "W"
 $ parameter  : NULL
 $ p.value    : num 0.00299
 $ null.value : Named num 0
  ..- attr(*, "names")= chr "location shift"
 $ alternative: chr "two.sided"
 $ method     : chr "Wilcoxon rank sum test with continuity correction"
 $ data.name  : chr "gardenA and gardenB"
```

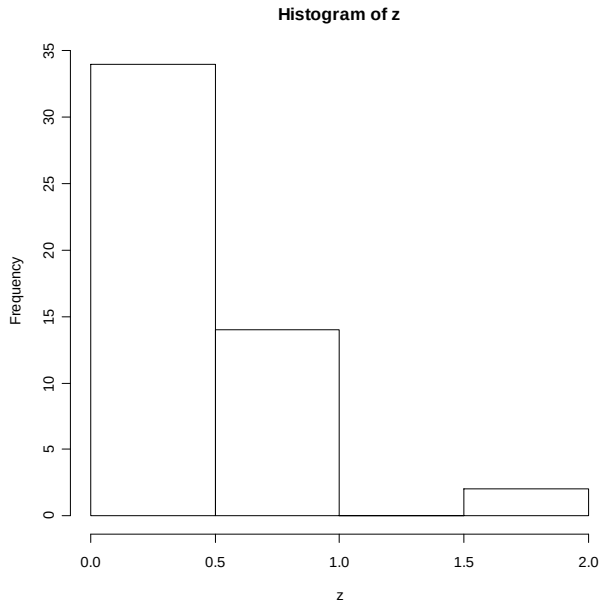
help.search("test")

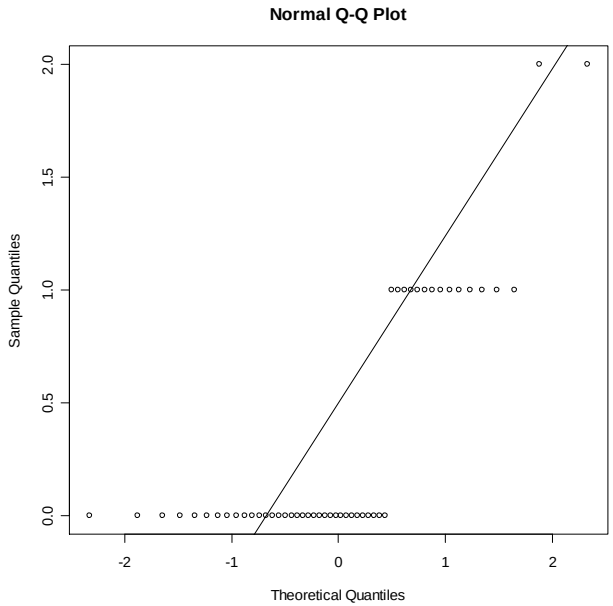
```
ansari.test(stats)      Ansari-Bradley Test
bartlett.test(stats)    Bartlett Test of Homogeneity of Variances
binom.test(stats)       Exact Binomial Test
Box.test(stats)         Box-Pierce and Ljung-Box Tests
chisq.test(stats)       Pearson's Chi-squared Test for Count Data
cor.test(stats)         Test for Association/Correlation Between
                        Paired Samples
fisher.test(stats)      Fisher's Exact Test for Count Data
fligner.test(stats)     Fligner-Killeen Test of Homogeneity of
                        Variances
friedman.test(stats)    Friedman Rank Sum Test
kruskal.test(stats)     Kruskal-Wallis Rank Sum Test
ks.test(stats)          Kolmogorov-Smirnov Tests
mantelhaen.test(stats)
                        Cochran-Mantel-Haenszel Chi-Squared Test for
                        Count Data
mauchley.test(stats)    Mauchley's test of sphericity
mcnemar.test(stats)     McNemar's Chi-squared Test for Count Data
```

When more libraries are loaded, their tests will be visible also.

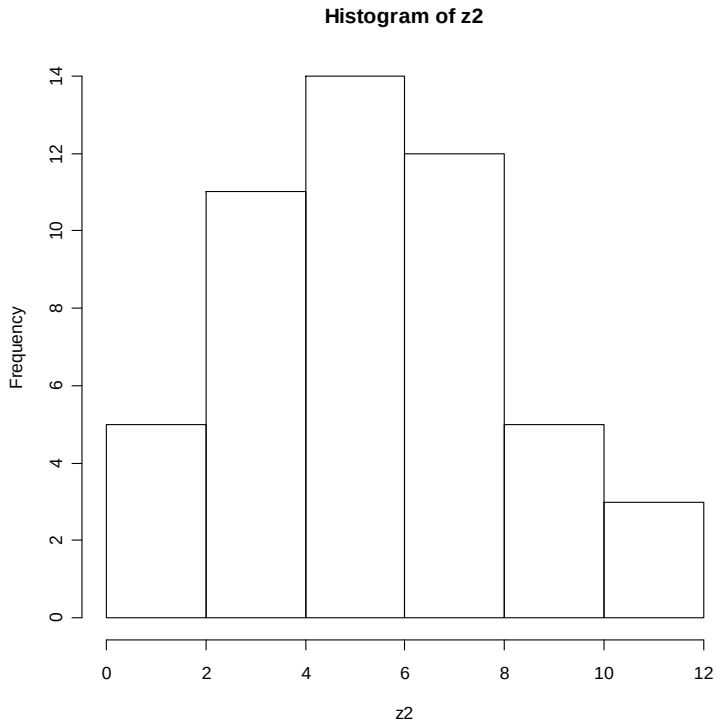
Simulating random numbers

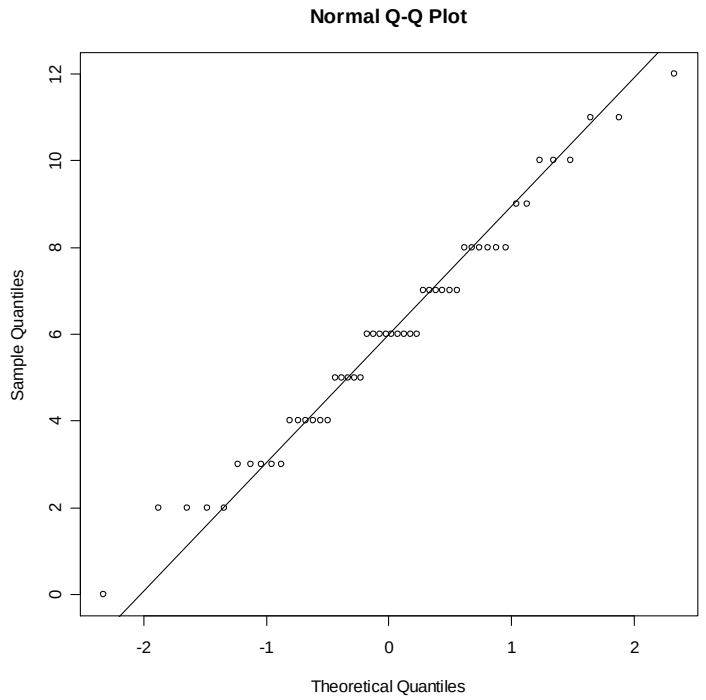
z1<-rpois(50,0.5) # 50 random numbers from a poisson distribution with parameter .5
hist(z1)
qqnorm(z1) # normal probability plot - a qq plot
qqline(z1)





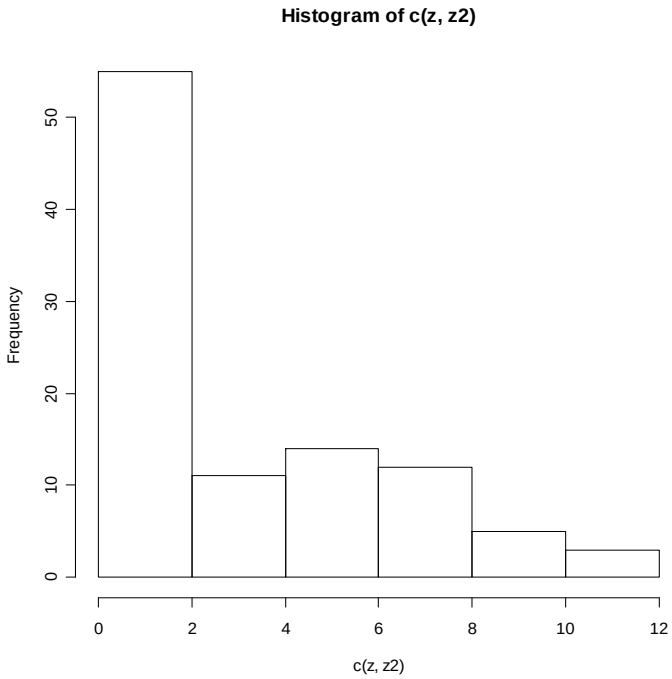
```
z2<-rpois(50,6.2)
hist(z2)
qqnorm(z2)
qqline(z2)
```

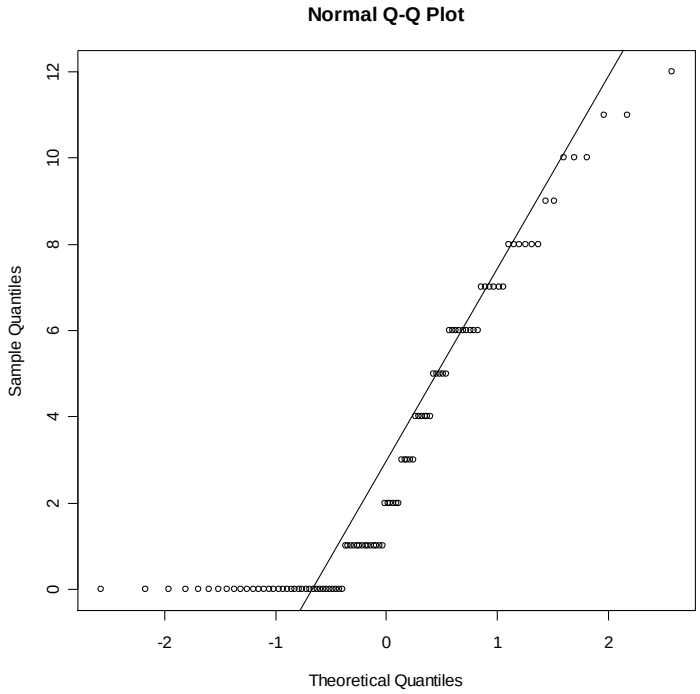




The two groups combined

hist(c(z1,z2))
qqnorm(c(z1,z2))
qqline(c(z1,z2))





What procedure/test to choose to decide if the means of *z1* and *z2* are equal?

If we wouldn't know to which probability distribution each observation belongs, can this compound distribution be unmixed?

Wilcoxon rank sum test or a Chisq test in a glm?

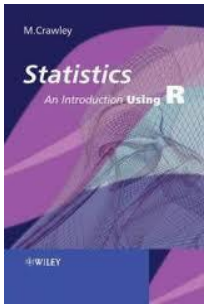
Their power is very similar, however

- glm allows you to calculate confidence intervals on the estimated means
- you can predict future observations using glm
- you can investigate to some extent whether the data are Poisson or not

In more complicated situations: only **glm** might allow hypothesis testing

R has functions to simulate data according a fitted model.
E.g. simulate().
This is very useful, because often H_0 assumes that the data are from a given model.

Further Exercises



Crawley 2005

http://www.imperial.ac.uk/bio/research/crawley/statistics