Lab 1: An introduction to R

Peter Diggle & Emanuele Giorgi

Lancaster Medical School, Lancaster University, Lancaster, UK

Model-based geostatistics: geospatial statistical methods for public health applications, 5-9 October 2015

Overview

Prerequisite: having R installed on your own computer.

Recommended reading: Venables, W. N., Smith, D. M., (2015) *An introduction to R*.

All the material is available at

`http://www.lancaster.ac.uk/staff/diggle/Malawi2015/`

- 09:00-10:30. Objects in R, reading data from files, probability distributions.
- 10:30-11:00. Break.
- 11:00-12:30. Conditional executions, loops and writing of your own function.

Getting help in R

```
> sqrt(4)
[1] 2
> ?sqrt
> apropos("sq")
[1] "chisq.test" "dchisq"     "pchisq"     "qchisq"     "rchisq"     "sqrt"
[7] "sQuote"
```

Help page

MathFun {base} R Documentation

Miscellaneous Mathematical Functions

Description

`abs(x)` computes the absolute value of `x`, `sqrt(x)` computes the (principal) square root of `x`, $\sqrt{x}$.

The naming follows the standard for computer languages such as C or Fortran.

Usage

```
abs(x)
sqrt(x)
```

Arguments

`x` a numeric or `complex` vector or array.

Details

These are internal generic primitive functions: methods can be defined for them individually or via the `Math` group generic. For complex arguments (and the default method), `z`, `abs(z) == Mod(z)` and `sqrt(z) == z^0.5`.

`abs(x)` returns an `integer` vector when `x` is `integer` or `logical`.

S4 methods

Both are S4 generic and members of the `Math` group generic.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`Arithmetic` for simple, `log` for logarithmic, `sin` for trigonometric, and `Special` for special mathematical functions.

'plotmath' for the use of `sqrt` in plot annotation.

Examples

```
require(stats) # for spline
require(graphics)
xx <- -9:9
plot(xx, sqrt(abs(xx)),  col = "red")
lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")
```

Numbers and vectors

```
> x <- c(1,0.3,4,5,-12,-4.12)
> x
[1]   1.00   0.30   4.00   5.00 -12.00  -4.12
>
> y <- c(x,0,x)
> y
 [1]   1.00   0.30   4.00   5.00 -12.00  -4.12   0.00   1.00   0.30   4.00   5.00 -12.00
[13]  -4.12
>
> v <- 2*x+abs(x)
> v[3]
[1] 36
> 2*4^2+abs(4)
[1] 36
>
> sum(x)
[1] -5.82
> x[1]+x[2]+x[3]+x[4]+x[5]+x[6]
[1] -5.82
>
> mean(x)
[1] -0.97
> (x[1]+x[2]+x[3]+x[4]+x[5]+x[6])/length(x)
[1] -0.97
> sum(x)/length(x)
[1] -0.97
```

Main number and vector arithmetic functions

Function	Command in R
Addition	`+`
Subtraction	`-`
Division	`/`
Multiplication	`*`
Power	`^`
Minimum	`min`
Maximum	`max`
Length of a vector	`length`
Sum of the elements of a vector	`sum`
Mean of the elements of a vector	`mean`
Product of the elements of a vector	`prod`

Exercise 1

Harmonic and geometric mean

Compute the harmonic and geometric mean of the integers from 1001 to 2000. Recall that

$$\text{Harmonic mean} = \frac{1000}{\sum_{i=1001}^{2000} \frac{1}{i}}$$

$$\text{Geometric mean} = \left(\prod_{i=1001}^{2000} i \right)^{1/1000}.$$

Exercise 1

Harmonic and geometric mean

Compute the harmonic and geometric mean of the integers from 1001 to 2000. Recall that

$$\text{Harmonic mean} = \frac{1000}{\sum_{i=1001}^{2000} \frac{1}{i}}$$

$$\text{Geometric mean} = \left(\prod_{i=1001}^{2000} i \right)^{1/1000}.$$

Solution: Harmonic mean ≈ 1443.215. Geometric mean ≈ 1472.028.

Other vector manipulations (1)

- Generating regular sequences.

```
> x1 <- c(-4, -2, -1, 0, 1, 2, 3, 4)
>
> x2 <- seq(from=-4,to=4,by=1)
> x2
[1] -4 -3 -2 -1  0  1  2  3  4
> x3 <- -4:4
> x3
[1] -4 -3 -2 -1  0  1  2  3  4
```

Other vector manipulations (1)

- Generating regular sequences.

```
> x1 <- c(-4, -2, -1, 0, 1, 2, 3, 4)
>
> x2 <- seq(from=-4,to=4,by=1)
> x2
[1] -4 -3 -2 -1  0  1  2  3  4
> x3 <- -4:4
> x3
[1] -4 -3 -2 -1  0  1  2  3  4
```

- Repeating values.

```
> x <- rep(1,5)
> x
[1] 1 1 1 1 1
> y <- rep(1:3, 3)
> y
[1] 1 2 3 1 2 3 1 2 3
> z <- rep(1:3, each=3)
> z
[1] 1 1 1 2 2 2 3 3 3
```

Other vector manipulations (1)

- Generating regular sequences.

```
> x1 <- c(-4, -2, -1, 0, 1, 2, 3, 4)
>
> x2 <- seq(from=-4,to=4,by=1)
> x2
[1] -4 -3 -2 -1  0  1  2  3  4
> x3 <- -4:4
> x3
[1] -4 -3 -2 -1  0  1  2  3  4
```

- Repeating values.

```
> x <- rep(1,5)
> x
[1] 1 1 1 1 1
> y <- rep(1:3, 3)
> y
[1] 1 2 3 1 2 3 1 2 3
> z <- rep(1:3, each=3)
> z
[1] 1 1 1 2 2 2 3 3 3
```

- Logical vectors.

```
> x <- seq(-3,3,by=0.5)
> x
 [1] -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
> y1 <- x > 0
> y1
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
> y2 <- x >= 0
> y2
 [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Other vector manipulations (2)

- Character vectors.

```
> x <- c("Bye", "Arrivederci", "Ndapita", "Caraysiiyo")
> x
[1] "Bye"          "Arrivederci" "Ndapita"     "Caraysiiyo"
> y <- paste(x,"Mr. Alinafe")
> y
[1] "Bye Mr. Alinafe"          "Arrivederci Mr. Alinafe" "Ndapita Mr. Alinafe"
[4] "Caraysiiyo Mr. Alinafe"
```

Other vector manipulations (2)

- Character vectors.

```
> x <- c("Bye", "Arrivederci", "Ndapita", "Caraysiiyo")
> x
[1] "Bye"          "Arrivederci" "Ndapita"     "Caraysiiyo"
> y <- paste(x,"Mr. Alinafe")
> y
[1] "Bye Mr. Alinafe"          "Arrivederci Mr. Alinafe" "Ndapita Mr. Alinafe"
[4] "Caraysiiyo Mr. Alinafe"
```

- Missing values.

```
> x <- c(0:10,NA,12)
> x
 [1]  0  1  2  3  4  5  6  7  8  9 10 NA 12
> ind <- is.na(x)
> ind
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
> !ind
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
> y <- x[!ind]
> y
 [1]  0  1  2  3  4  5  6  7  8  9 10 12
```

Other vector manipulations (2)

- Character vectors.

```
> x <- c("Bye", "Arrivederci", "Ndapita", "Caraysiiyo")
> x
[1] "Bye"          "Arrivederci" "Ndapita"     "Caraysiiyo"
> y <- paste(x,"Mr. Alinafe")
> y
[1] "Bye Mr. Alinafe"          "Arrivederci Mr. Alinafe" "Ndapita Mr. Alinafe"
[4] "Caraysiiyo Mr. Alinafe"
```

- Missing values.

```
> x <- c(0:10,NA,12)
> x
 [1]  0  1  2  3  4  5  6  7  8  9 10 NA 12
> ind <- is.na(x)
> ind
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
> !ind
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
> y <- x[!ind]
> y
 [1]  0  1  2  3  4  5  6  7  8  9 10 12
```

- Selecting and modifying a subset of the data

```
> x <- seq(-0.5,3,by=0.5)
> x
[1] -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
> y <- x[x > 0.51]
> y
[1] 1.0 1.5 2.0 2.5 3.0
```

Other objects (1)

- Matrices.

```
> M <- matrix(NA,nrow=5,ncol=3)
> M[2,3] <- 1
> M[c(1,3,4,5),1] <- -2
> M
     [,1] [,2] [,3]
[1,]   -2   NA   NA
[2,]   NA   NA    1
[3,]   -2   NA   NA
[4,]   -2   NA   NA
[5,]   -2   NA   NA
> M[1:3,c(1,3)]
     [,1] [,2]
[1,]   -2   NA
[2,]   NA    1
[3,]   -2   NA
```

Other objects (1)

- Matrices.

```
> M <- matrix(NA,nrow=5,ncol=3)
> M[2,3] <- 1
> M[c(1,3,4,5),1] <- -2
> M
     [,1] [,2] [,3]
[1,]   -2   NA   NA
[2,]   NA   NA    1
[3,]   -2   NA   NA
[4,]   -2   NA   NA
[5,]   -2   NA   NA
> M[1:3,c(1,3)]
     [,1] [,2]
[1,]   -2   NA
[2,]   NA    1
[3,]   -2   NA
```

- Unordered and ordered factors.

```
> state <- c(rep("England",3), "Italy", rep("Malawi",10), rep("Somalia",5))
> state <- factor(state)
> levels(state)
[1] "England" "Italy"   "Malawi"  "Somalia"
> table(state)
state
England   Italy  Malawi Somalia
      3       1      10       5
>
> income <- c(rep("(1000-3000]",5),rep("(3000-5500]",3),rep("(5500-10000]",2))
> income <- factor(income, ordered=TRUE)
> income
 [1] (1000-3000]  (1000-3000]  (1000-3000]  (1000-3000]  (1000-3000]  (3000-5500]
 [7] (3000-5500]  (3000-5500]  (5500-10000] (5500-10000]
Levels: (1000-3000] < (3000-5500] < (5500-10000]
```

Other objects (2)

- Lists.

```
> Ls <- list(name="Emanuele", age=27, Male=TRUE, weight=NA)
> str(Ls)
List of 4
 $ name  : chr "Emanuele"
 $ age   : num 27
 $ Male  : logi TRUE
 $ weight: logi NA
> Ls$age
[1] 27
> Ls[[2]]
[1] 27
```

Other objects (2)

- Lists.

```
> Ls <- list(name="Emanuele", age=27, Male=TRUE, weight=NA)
> str(Ls)
List of 4
 $ name  : chr "Emanuele"
 $ age   : num 27
 $ Male  : logi TRUE
 $ weight: logi NA
> Ls$age
[1] 27
> Ls[[2]]
[1] 27
```

- Data-frames.

```
> data <- data.frame(ID=c("A", "B", "C", "D"), age=c(20,21,24,NA),
+                    gender=c("M","M","F","F"),
+                    employed=c(1,1,1,0),
+                    state=c("Germany", "France", "England", "Italy"))
> str(data)
'data.frame': 4 obs. of  5 variables:
 $ ID      : Factor w/ 4 levels "A","B","C","D": 1 2 3 4
 $ age     : num  20 21 24 NA
 $ gender  : Factor w/ 2 levels "F","M": 2 2 1 1
 $ employed: num  1 1 1 0
 $ state   : Factor w/ 4 levels "England","France",..: 3 2 1 4
 > data[data$ID=="B",]
  ID age gender employed  state
2  B  21      M        1 France
```

Reading and saving data files

- The `read.table()` and `read.csv()` functions.

```
> args(read.table)
function (file, header = FALSE, sep = "", quote = "\"'", dec = ".",
...
NULL
> args(read.csv)
function (file, header = TRUE, sep = ",", quote = "\"", dec = ".",
    fill = TRUE, comment.char = "", ...)
NULL
>
> data.Liberia <- read.csv("http://www.lancaster.ac.uk/staff/
                           diggle/Malawi2015/LiberiaRemoData.csv")
> data.Liberia <- read.csv(file.choose())
> str(data.Liberia)
'data.frame': 90 obs. of  4 variables:
 $ lat  : num  6.29 6.57 6.57 6.73 6.02 ...
 $ long : num  -10.5 -10.5 -10 -10.3 -10 ...
 $ ntest: int  50 46 43 50 48 50 43 50 47 43 ...
 $ npos : int  14 10 11 10 9 13 9 11 11 0 ...
>
> write.table(data.Liberia,file="Liberia.txt",row.names=FALSE
```

Exercise 2

A mini spatial data-set

- Create a data-frame called `spatial_data` with the following structure

```
> spatial_data
   ID Longitude Latitude         Village gender age
1   1    -14.33    33.83     Muli-bwanji      M  20
2   2    -14.33    33.83     Muli-bwanji      F  NA
3   3    -14.33    33.83     Muli-bwanji      M  22
4   4    -14.33    33.83     Muli-bwanji      F  23
5   5    -14.25    33.95 Mwadzuka-bwanji      M  24
6   6    -14.25    33.95 Mwadzuka-bwanji      F  25
7   7    -14.25    33.95 Mwadzuka-bwanji      M  NA
8   8    -14.25    33.95 Mwadzuka-bwanji      F  NA
9   9    -14.22    34.02 Mwaswera-bwanji      M  28
10 10    -14.22    34.02 Mwaswera-bwanji      F  29
11 11    -14.22    34.02 Mwaswera-bwanji   <NA>  30
12 12    -14.22    34.02 Mwaswera-bwanji   <NA>  31
```

- Remove the missing data from the data frame.
- How many individuals are male? How many from each village?
- Save the data-frame as ``spatial_data.csv'' and then read it using `read.csv()`.

Probability distributions

Distribution	R name	additional arguments
beta	`beta`	`shape1, shape2, ncp`
binomial	`binom`	`size, prob`
Cauchy	`cauchy`	`location, scale`
chi-squared	`chisq`	`df, ncp`
exponential	`exp`	`rate`
F	`f`	`df1, df2, ncp`
gamma	`gamma`	`shape, scale`
geometric	`geom`	`prob`
hypergeometric	`hyper`	`m, n, k`
log-normal	`lnorm`	`meanlog, sdlog`
logistic	`logis`	`location, scale`
negative binomial	`nbinom`	`size, prob`
normal	`norm`	`mean, sd`
Poisson	`pois`	`lambda`
signed rank	`signrank`	`n`
Student's t	`t`	`df, ncp`
uniform	`unif`	`min, max`
Weibull	`weibull`	`shape, scale`
Wilcoxon	`wilcox`	`m, n`

Example: Gaussian distribution (1)

- Density function:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}.$$

- $E(X) = \mu$; $\mathsf{var}(X) = \sigma^2$.

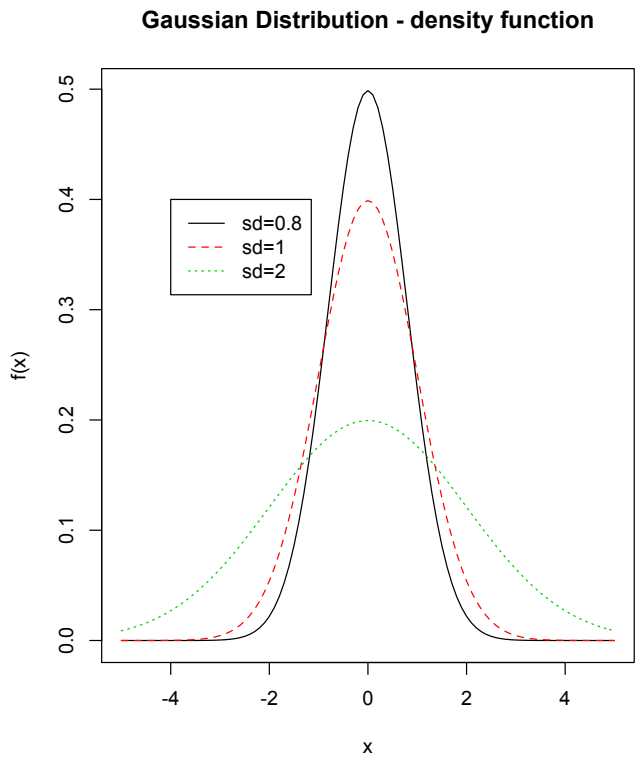
Example: Gaussian distribution (1)

- Density function:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}.$$

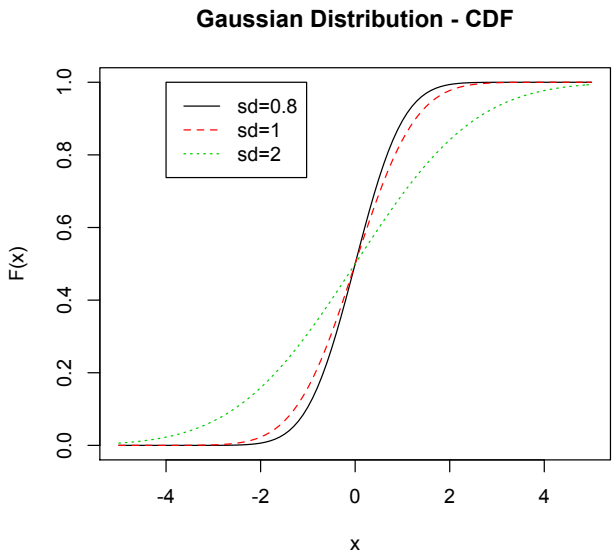
- $E(X) = \mu$; $\text{var}(X) = \sigma^2$.

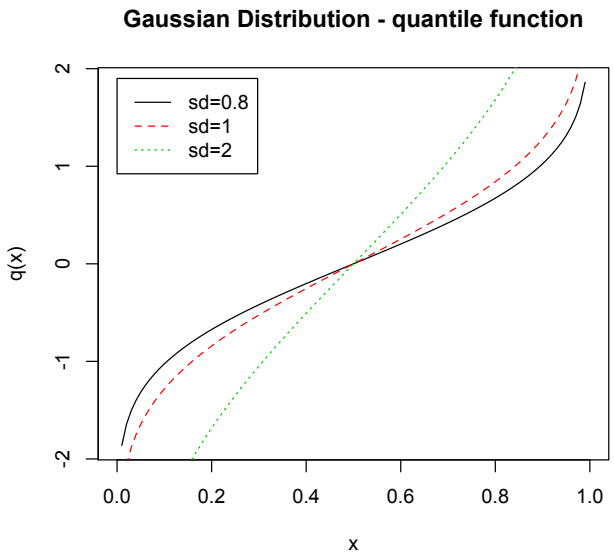
```
> curve(dnorm(x,mean=0,sd=0.8),xlim=c(-5,5),col=1,lty=1,
+       ylab="f(x)",xlab="x",main="Gaussian Distribution")
> curve(dnorm(x,mean=0,sd=1),xlim=c(-5,5),col=2,lty=2,add=TRUE)
> curve(dnorm(x,mean=0,sd=2),xlim=c(-5,5),col=3,lty=3,add=TRUE)
>
> legend(-4,0.4,c("sd=0.8", "sd=1", "sd=2"),col=1:3,lty=1:3)
```



Example: Gaussian distribution (2)

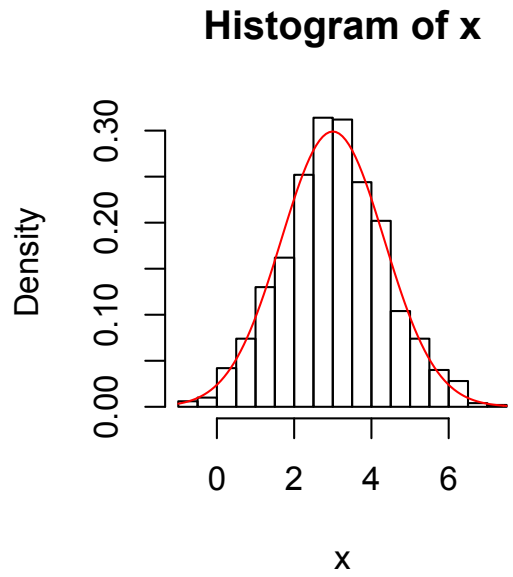
```
par(mfrow=c(1,2))

curve(pnorm(x,mean=0,sd=0.8),xlim=c(-5,5),col=1,lty=1,
      ylab="F(x)",xlab="x",main="Gaussian Distribution - CDF")
curve(pnorm(x,mean=0,sd=1),xlim=c(-5,5),col=2,lty=2,add=TRUE)
curve(pnorm(x,mean=0,sd=2),xlim=c(-5,5),col=3,lty=3,add=TRUE)

legend(-4,1,c("sd=0.8", "sd=1", "sd=2"),col=1:3,lty=1:3)

curve(qnorm(x,mean=0,sd=0.8),xlim=c(0,1),col=1,lty=1,
      ylab="q(x)",xlab="x",main="Gaussian Distribution - quantile function")
curve(qnorm(x,mean=0,sd=1),xlim=c(0,1),col=2,lty=2,add=TRUE)
curve(qnorm(x,mean=0,sd=2),xlim=c(0,1),col=3,lty=3,add=TRUE)

legend(-4,1,c("sd=0.8", "sd=1", "sd=2"),col=1:3,lty=1:3)
```

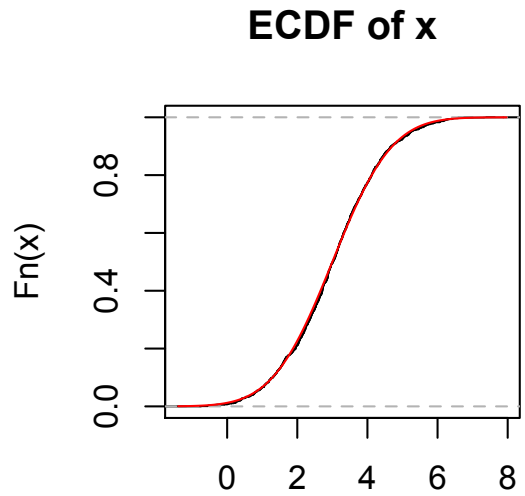




Example: Gaussian distribution (3)

```
> mu <- 3
> sigma2 <- 1.78
> n <- 1000
>
> set.seed(123)
> x <- rnorm(n,mean=mu,sd=sqrt(sigma2))
>
> par(mfrow=c(1,2))
> hist(x,prob=TRUE,nclass=20)
> curve(dnorm(x,mean=mu,sd=sqrt(sigma2)),add=TRUE,col=2)
>
> plot(ecdf(x),main="ECDF")
> curve(pnorm(x,mean=mu,sd=sqrt(sigma2)),add=TRUE,col=2)
> mu <- 3
> sigma2 <- 1.78
> n <- 1000
>
> set.seed(123)
> x <- rnorm(n,mean=mu,sd=sqrt(sigma2))
>
> par(mfrow=c(1,2))
> hist(x,prob=TRUE,nclass=20)
> curve(dnorm(x,mean=mu,sd=sqrt(sigma2)),add=TRUE,col=2)
>
> plot(ecdf(x),main="ECDF of x")
> curve(pnorm(x,mean=mu,sd=sqrt(sigma2)),add=TRUE,col=2)
```





Exercise 3

Gamma distribution

Density function

$$f(x) = \frac{\beta^{-\alpha}}{\Gamma(\alpha)} x^{\alpha-1} \exp\{-x/\beta\}, x > 0, \alpha > 0, \beta > 0.$$

$E(X) = \alpha\beta$; $\text{var}(X) = \alpha\beta^2$.

- Plot the density, CDF and quantile function for $\alpha = 0.5, 1, 2$ and $\beta = 2$.
- Simulate three samples of size $100, 500$ and 1000 from a Gamma distribution with $\alpha = 1.5$ and $\beta = 1$. For each of these plot the histogram and ECDF, together with their theoretical counterpart.

Loops

- Conditional executions.

```
 > x <- 4
> if(x==4) {
+     x <- 5
+ }
> x
[1] 5
```

Loops

- Conditional executions.

```
 > x <- 4
> if(x==4) {
+    x <- 5
+ }
> x
[1] 5
```

- `for` loops.

```
> n.iter <- 20
> fib <- rep(NA,n.iter+2)
> fib[1] <- 0
> fib[2] <- 1
>
> for(i in 3:(n.iter+2)) {
+  fib[i] <- fib[i-1]+fib[i-2]
+ }
> fib
 [1]     0     1     1     2     3     5     8    13    21    34    55    89   144   233
[15]   377   610   987  1597  2584  4181  6765 10946
```

Loops

- Conditional executions.

```
 > x <- 4
> if(x==4) {
+    x <- 5
+ }
> x
[1] 5
```

- `for` loops.

```
> n.iter <- 20
> fib <- rep(NA,n.iter+2)
> fib[1] <- 0
> fib[2] <- 1
>
> for(i in 3:(n.iter+2)) {
+  fib[i] <- fib[i-1]+fib[i-2]
+ }
> fib
 [1]     0     1     1     2     3     5     8    13    21    34    55    89   144   233
[15]   377   610   987  1597  2584  4181  6765 10946
```

- `while` loops.

```
> n.iter <- 20
> counter <- 1
> fib <- rep(NA,n.iter+2)
> fib[1] <- 0
> fib[2] <- 1
>
> while(counter <= n.iter) {
+  fib[counter+2] <- fib[counter+1]+fib[counter]
+  counter <- counter + 1
+ }
```

Exercise 4

The exponential correlation matrix

Read the data frame ``LiberiaRemoData.csv''. Using a `for` loop, create a correlation matrix, say R, with (i, j)-th entry given by

$$[R]_{ij} = \exp\left\{-\frac{\|x - x'\|}{2}\right\},$$

where $\|x - x'\|$ is the distance between locations x and x'.

Hint: what is the dimension of R? Note also that R is symmetric, hence $[R]_{ij} = [R]_{ji}$.

Write your own function

```
> sum.vectors <- function(x,y) {
+    n.x <- length(x)
+    n.y <- length(y)
+    if(n.x!=n.y) stop("'x' and 'y' must be of the same length")
+
+    z <- rep(NA,n.x)
+    for(i in 1:n.x) {
+     z[i] <- x[i] + y[i]
+    }
+    return(z)
+ }
>
> sum.vectors(1:4,2:5)
[1] 3 5 7 9
> 1:4+2:5
[1] 3 5 7 9
>
> sum.vectors(1:3,2:5)
Errore in sum.vectors(1:3, 2:5) : 'x' and 'y' must be of the same length
> 1:3+2:5
[1] 3 5 7 6
Warning message:
In 1:3 + 2:5 :
  longer object length is not a multiple of shorter object length
```

Exercise 5

Summary of a numeric vector

Create an R function which returns main summaries of a numeric vector with the following structure.

- **Input.** The function must have two arguments. The first called `x` is a numeric vector, the second is a logical argument called `display.boxplot` (see `?boxplot`).
- **Output.** The function must return a list with five components corresponding to the mean, standard deviation, median, first quartile and third quartile of `x`. Additionally, if `display.boxplot=TRUE`, it should also display the box plot of `x`.

Extra task. If there are missing values, the function should remove those, compute the relevant quantities and return a warning message such as `''Missing values were found and removed''` (see `?warning`).