

Σύντομη εισαγωγή στο R (www.r-project.org)

- Το R (τρέχουσα έκδοση 2.10.1) είναι ένα *υπολογιστικό πακέτο* που προσφέρει δυνατότητες διαχείρισης και *στατιστικής ανάλυσης δεδομένων* καθώς και δυνατότητες *κατασκευής γραφημάτων*.
- Βασίζεται στην *γλώσσα προγραμματισμού S* (που χρησιμοποιεί και το στατιστικό πακέτο S plus) και πρόκειται για *λογισμικό ανοικτού κώδικα* (open source) που διατίθεται ελεύθερα.
- Όποιος επιθυμεί μπορεί να συνεισφέρει και ήδη υπάρχουν εκατοντάδες πακέτα (επεκτάσεις) που προσφέρουν νέες δυνατότητες στο υπάρχον σύστημα.
- Για μια πιο αναλυτική εισαγωγή παραπέμπουμε στο αρχείο R-intro.pdf (An introduction to R, W.N. Venables, D.M. Smith and the R development core team) που παρέχεται στα help files του R.

Σύνταξη και εκτέλεση στοιχειωδών εντολών

- Οι εντολές προς το R δίνονται μέσω της **R-Console**: μετά το σύμβολο ">" γράφουμε την εντολή (ή το σύνολο των εντολών) που θέλουμε να εκτελεστούν και πατώντας ENTER λαμβάνουμε το αποτέλεσμα στην επόμενη γραμμή.
- Εναλλακτικά, μπορούμε να ανοίξουμε ένα *παράθυρο script* (**R-editor** από το menu: file/open script) όπου γράφουμε όσες εντολές επιθυμούμε και μετά μαρκάρουμε αυτές που θέλουμε να εκτελεστούν και πατάμε control+R. (αν δεν μαρκάρουμε τίποτε, με control+R θα εκτελεστούν μόνο οι εντολές της γραμμής που βρίσκεται ο *cursor*).
- Η σύνταξη εντολών στο R γίνεται χρησιμοποιώντας αλφαριθμητικές εκφράσεις (Αγγλικό αλφάβητο). Μπορούν επίσης να χρησιμοποιηθούν συγκεκριμένα σύμβολα, π.χ. ".", ",", "_".
- Αν δοθεί εντολή κατασκευής γραφήματος, το αποτέλεσμα εμφανίζεται σε ξεχωριστό παράθυρο γραφικών (**R- Graphics**).

- Οι στοιχειώδεις εντολές μπορεί να είναι *δύο ειδών*:

(α) *expressions*, όπου το αποτέλεσμα τυπώνεται στην οθόνη αλλά δεν καταχωρείται πουθενά, (π.χ. σε κάποια μεταβλητή), π.χ.

```
> 2 + 3  
5
```

(β) *assignments*, όπου το αποτέλεσμα δεν τυπώνεται στην οθόνη αλλά καταχωρείται σε μια μεταβλητή.

```
> x <- 2 + 3
```

(το "<-" είναι ισοδύναμο με "=").

Εάν θέλουμε να δούμε το αποτέλεσμα που έχει καταχωρηθεί στην μεταβλητή **x** μπορούμε στη συνέχεια να γράψουμε

```
> x  
ή  
> print(x)
```

- Το R μπορεί να εκτελέσει ένα σύνολο εντολών, διαδοχικά, τη μία μετά την άλλη. Οι εντολές μπορεί να χωρίζονται με ";" (ή με NEWLINE στον script editor).

```
> x<-1+2; print(x)
3
> y<-x^2/2; print(y)
4.5
```

- Αν μια εντολή δεν είναι πλήρης, τότε στην επόμενη γραμμή το R περιμένει (εμφανίζοντας ένα "+") την συνέχεια (αυτό γίνεται στην R-console)

```
> print(x
+ )
3
```

- Μπορούμε να ομαδοποιήσουμε εντολές γράφοντάς τις μεταξύ άγκιστρων

```
> {x<-2+3;
+ print(x);
+ x^2}
5
25
```

- Όσα ακολουθούν τον χαρακτήρα "#" σε μία γραμμή θεωρούνται σχόλια και δεν λαμβάνονται υπόψη

```
> x<-2+4 # to x einai to athroisma tou 2 syn 4  
> # typwnoyme to apotelesma  
> print(x)  
6
```

- Μπορούμε να επανεμφανίσουμε μία εντολή που έχουμε γράψει παραπάνω χρησιμοποιώντας τα βέλη (arrows) στο πληκτρολόγιο (στο R console).

- Αν θέλουμε να εκτελέσουμε μία ομάδα εντολών που βρίσκονται σε κάποιο αρχείο (που υπάρχει στο *working directory*) με όνομα π.χ. `progr.r` τότε εκτελούμε

```
source("progr.r").
```

- Εάν θέλουμε τα αποτελέσματα των εντολών μας να αποθηκεύονται σε ένα αρχείο, π.χ. στο αρχείο `output.txt`, τότε εκτελούμε

```
sink("output.txt").
```

Επανερχόμαστε στην αρχική κατάσταση (παρουσίαση αποτελεσμάτων στην κονσόλα) εκτελώντας `sink()`.

- Για να δούμε ποια αντικείμενα (π.χ. μεταβλητές) έχουν οριστεί στο session που εργαζόμαστε εκτελούμε

```
> objects()  
"x" "y"
```

Για να διαγράψουμε κάποια από αυτά εκτελούμε

```
> rm(x, y)
```

- Στο τέλος ενός session μπορούμε να αποθηκεύσουμε τις μεταβλητές που έχουμε δημιουργήσει καθώς και τις εντολές που έχουμε χρησιμοποιήσει. Αυτό γίνεται από:

File/save workspace, save history.

Μπορούμε να τα ξαναφορτώσουμε από:

File/load workspace, load history

Απλές πράξεις με αριθμούς και διανύσματα

- Η απλούστερη δομή δεδομένων που μπορούμε να χρησιμοποιήσουμε είναι το διάνυσμα. Αν θέλουμε π.χ. να ορίσουμε $x = (1, 2, 3, 5.2)$ εκτελούμε

```
> x<-c(1,2,4,5.2)
```

- το `c` προέρχεται από το concatenate (*συνδέω αλυσιδωτά*). Με το `c` μπορούμε να συνδέσουμε και διανύσματα σε μεγαλύτερα διανύσματα

```
> x<-c(1,2,4,5.2)
```

```
> z<-c(x,x^2,12,13)
```

```
> print(z)
```

```
1.00  2.00  4.00  5.20  1.00  4.00 16.00 27.04 12.00 13.00
```

- Πράξεις με διανύσματα γίνονται κατά συντεταγμένες, δηλαδή:

```
> x<-c(1,2,4,5.2)
```

```
> y<-x*2+1
```

```
> print(y)
```

```
3.0  5.0  9.0 11.4
```

- Πολύ χρήσιμο είναι το διάνυσμα της μορφής

$a:b$ που ισούται με $(a, a+1, \dots, b)$ αν $b > a$

ή $(a, a-1, \dots, b)$ αν $b < a$.

- Γενικότερα μπορούμε να χρησιμοποιήσουμε την εντολή **seq(a,b,step)**. Επίσης η εντολή **rep** χρησιμοποιείται για την επανάληψη στοιχείων ενός διανύσματος.

```
> i<-1:10
> print(i)
 1  2  3  4  5  6  7  8  9 10

> seq(1,3,0.2)
1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0

> x<-1:3
> rep(x,times=4)
 1 2 3 1 2 3 1 2 3 1 2 3

> rep(x,each=5)
 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
```

- Μπορούμε να χρησιμοποιήσουμε τις γνωστές πράξεις $+$, $-$, $*$, $/$, $^$ και συναρτήσεις $\log$, $\exp$, $\sin$, $\cos$, $\tan$, $\sqrt{}$, ...

```
> x<-1:10; x^2
  1    4    9   16   25   36   49   64   81  100
> x<-(0:6-3)/2
> print(x)
-1.5 -1.0 -0.5  0.0  0.5  1.0  1.5
> print(exp(-x^2/2)/(2*pi)^0.5)
0.1295176 0.2419707 0.3520653 0.3989423 0.3520653 0.2419707 0.1295176
```

- Συναρτήσεις επί ενός διανύσματος που επιστρέφουν μία τιμή είναι: $\min$, $\max$, sum , prod , length , mean , var

```
> x<-1:10; sum(x)
55
> prod(x)
3628800
> sum((x-mean(x))^2)/(length(x)-1)
9.166667
> var(x)
9.166667
```

- Η εντολή `sort` διατάσσει τα στοιχεία μιας λίστας κατά αύξουσα σειρά.

Λογικοί τελεστές

- Για τον έλεγχο της αλήθειας αριθμητικών ή αλφαριθμητικών εκφράσεων μπορούμε να χρησιμοποιήσουμε τις σχέσεις $<$, $>$, $<=$, $>=$, $==$, $!=$. Το αποτέλεσμα τέτοιων ελέγχων είναι μια boolean μεταβλητή η οποία μπορεί να λαμβάνει τιμές **T** (**TRUE**), **F** (**FALSE**), **NA** (*Not Available*).

```
> x <- 2 > 1
> x
TRUE
```

- Για την δημιουργία εκφράσεων προτασιακού λογισμού μπορούμε να χρησιμοποιήσουμε τους λογικούς τελεστές **&** (**AND**), **|** (**OR**), **!** (**NOT**)

```
> y <- 1:10
> x <- !((y > 1) & (y <= 4)) | (y == 8)
> x
TRUE FALSE FALSE FALSE TRUE TRUE TRUE FALSE TRUE TRUE
```

- Η τιμή **NA** μπορεί να χρησιμοποιηθεί ως missing value

```
> x <- c(1, 2, NA, 4)
```

- Μία άλλη missing value που χρησιμοποιεί το R είναι η **NaN** (*Not a Number*)

```
> 0/0
NaN
```

Μεταβλητές με αλφαριθμητικές τιμές (strings)

- Εκτός από αριθμητικές ή λογικές, οι τιμές μιας μεταβλητής (ή ενός διανύσματος) μπορεί επίσης να είναι και αλφαριθμητικές (string)

```
> x<-c("male","female")
> rep(x,c(2,3))
"male"    "male"    "female" "female" "female"
```

Επιλογή ή τροποποίηση στοιχείων ενός διανύσματος (λίστας)

- Για να επιλέξουμε το i στοιχείο μιας λίστας $\mathbf{x}$ γράφουμε $\mathbf{x}[i]$.

```
> x<-1:10/10
> x[3]+x[4]
0.7

> print(x[c(1,3)])
0.1 0.3

> y<-x[1:5];print(y)
0.1 0.2 0.3 0.4 0.5

> x<-1:10/10
> x[3]<-0;print(x)
0.1 0.2 0.0 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

- Για να διαγράψουμε το i στοιχείο μιας λίστας $\mathbf{x}$ γράφουμε $\mathbf{x}[-i]$

```
>  $\mathbf{x}[-7]$   
0.1 0.2 0.0 0.4 0.5 0.6 0.8 0.9 1.0
```

- Για να κρατήσουμε μόνο τα στοιχεία μιας λίστας $\mathbf{x}$ που ικανοποιούν μια συνθήκη γράφουμε $\mathbf{x}[\text{συνθήκη}]$

```
>  $\mathbf{x}[\mathbf{x}>0.5]$   
0.6 0.7 0.8 0.9 1.0
```

- Όπως θα δούμε στη συνέχεια, το R μπορεί να χειριστεί και πολυπλοκότερες δομές δεδομένων όπως π.χ. πίνακες ή γενικότερα arrays, λίστες (διανύσματα με τιμές μεταβλητές διαφόρων τύπων), data.frames, functions, factors, κλπ

Ο τύπος μεταβλητής factor

- Χρησιμοποιείται για να ομαδοποιήσει τιμές διανυσμάτων

```
> x<-c(1.1, 2.3, 4.6, 3.2, 5.5, 2.0)
> gender<-factor(c("male","male","male","female","female","female"))
> levels(gender)
"female" "male"
```

```
> tapply(x,gender,mean)
female      male
3.566667 2.666667
```

```
> tapply(x,gender,var)
female      male
3.163333 3.163333
```

Ο τύπος μεταβλητής Array

- Ένα array είναι ένα k -διάστατο σύνολο από στοιχεία (π.χ. αριθμούς). Τα διανύσματα είναι μονοδιάστατα arrays ενώ οι πίνακες είναι διδιάστατα arrays. Ένα array δημιουργείται από ένα διάνυσμα με την εντολή **dim**.

```
> z<-1:8
> dim(z)<-c(2,2,2)
> z
, , 1
      [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2
      [,1] [,2]
[1,]    5    7
[2,]    6    8
```

- Τα στοιχεία ενός array μπορούν να ανακληθούν, και να τροποποιηθούν όπως και τα στοιχεία των διανυσμάτων

```
> z[1,1,1]; z[1,1,2]; z[1,2,1]; z[1,2,2]; z[2,1,1]; z[2,1,1]; z[2,2,1];
z[2,2,2];
```

```
1
5
3
7
2
2
4
8
```

```
> z[1,,]
      [,1] [,2]
[1,]    1    5
[2,]    3    7
```

```
> z[, ,1]
      [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
> z[,2,1]
3 4
```

```
> dim(z)
2 2 2
```

- Πράξεις μεταξύ arrays μπορούν να γίνουν όπως και μεταξύ διανυσμάτων, αρκεί τα arrays να έχουν τις ίδιες διαστάσεις.

```
> w<-z[, ,1]
> w^2+1
```

	[,1]	[,2]
[1,]	2	10
[2,]	5	17

- Το εξωτερικό γινόμενο διανυσμάτων γίνεται με την εντολή **outer**

```
> a<-c(1,2,3)
> b<-c(4,5,6)
> outer(a,b,"*")
```

	[,1]	[,2]	[,3]
[1,]	4	5	6
[2,]	8	10	12
[3,]	12	15	18

Στην θέση του "*" μπορεί να μπει οποιαδήποτε πράξη ή γενικότερα function δύο μεταβλητών

- Για τον χειρισμό διδιάστατων arrays (δηλαδή πινάκων) υπάρχουν ειδικές εντολές. Οι πράξεις $*$, $+$, $/$, $-$, $^$ μεταξύ πινάκων γίνονται όπως και στα arrays, ανά στοιχείο. Ο πολλαπλασιασμός πινάκων γίνεται με $\%*\%$

```
> A<-c(1,2,3,4); dim(A)<-c(2,2);print(A)
```

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

```
> B<-c(1,1,2,2); dim(B)<-c(2,2);print(B)
```

```
      [,1] [,2]
[1,]    1    2
[2,]    1    2
```

```
> 2*A+B^2
```

```
      [,1] [,2]
[1,]    3   10
[2,]    5   12
```

```
> A%*%B
```

```
      [,1] [,2]
[1,]    4    8
[2,]    6   12
```

- Η εντολή **t(A)** (transpose) επιστρέφει τον ανάστροφο του πίνακα **A**,
- Η **diag(x)** επιστρέφει έναν διαγώνιο πίνακα που στην κύρια διαγώνιο έχει τα στοιχεία του διανύσματος **x**. Αν το **x** δεν είναι διάνυσμα αλλά αριθμός, τότε η **diag(x)** επιστρέφει τον μοναδιαίο πίνακα **x×x**.
- Αντιστροφή πίνακα (**solve**):

```
> A<-c(1,2,3,4) ; dim(A)<-c(2,2) ;print(A)
```

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

```
> B<-solve(A)
```

```
> B
```

```
      [,1] [,2]
[1,]    -2  1.5
[2,]     1 -0.5
```

```
> A%*%B
```

```
      [,1] [,2]
[1,]     1     0
[2,]     0     1
```

- Η εντολή `solve(A,b)` επιστρέφει το διάνυσμα $A^{-1}b$, δηλαδή τη λύση του συστήματος $b = Ax$. Οι ιδιοτιμές και τα ιδιοδιανύσματα ενός πίνακα λαμβάνονται:

```
> A<-c(1,2,3,4); dim(A)<-c(2,2);print(A)
```

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

```
> eval<-eigen(A)$val;print(eval)
5.3722813 -0.3722813
```

```
> evec<-eigen(A)$vec;print(evec)
      [,1] [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
```

```
> evec%*%diag(eval)%*%solve(evec)
      [,1] [,2]
[1,]     1     3
[2,]     2     4
```

το οποίο επαληθεύει το θεώρημα φασματικής ανάλυσης ενός πίνακα.

Ο τύπος μεταβλητής List

- List θεωρείται μία (διατεταγμένη) συλλογή μεταβλητών διαφόρων τύπων.

Για παράδειγμα αν θέλουμε να φτιάξουμε μία μεταβλητή τύπου List που θα αποτελείται από 3 διανύσματα (1,3,3,4,5), ("a","b","c"), (10,20) τότε γράφουμε

```
> a<-list(c(1,3,3,4,5),c("a","b","c"),c(10,20))  
> print(a)
```

```
[[1]]  
 1  3  3  4  5  
  
[[2]]  
"a" "b" "c"  
  
[[3]]  
10 20
```

- Στο παραπάνω παράδειγμα που δεν έχουμε δώσει ονόματα στα τρία στοιχεία της λίστας, μπορούμε να τα καλέσουμε χρησιμοποιώντας δείκτες του τύπου `a[[i]]`

```
> a[[1]]  
 1  3  3  4  5
```

- Μπορούμε όμως να δώσουμε και ονόματα στα στοιχεία μιας λίστας. π.χ.

```
> a<-list(x1=c(1,2,3),x2=c(4,12,13),fylo=c("female","male","male"))
> print(a)
```

```
$x1
```

```
1 2 3
```

```
$x2
```

```
4 12 13
```

```
$fylo
```

```
"female" "male" "male"
```

- Μπορούμε τώρα να καλέσουμε το i στοιχείο της λίστας (με όνομα *name*) χρησιμοποιώντας δείκτες τύπου `a[[i]]` όπως παραπάνω ή γράφοντας `a$name` (ή και `a[["name"]]`),

```
> a[[2]]
```

```
4 12 13
```

```
> a$x2
```

```
4 12 13
```

```
> a$fylo[2]
```

```
"male"
```

Ο τύπος μεταβλητής Data Frame

- Πρόκειται για μια μεταβλητή List με συγκεκριμένα χαρακτηριστικά. Ένα από αυτά είναι ότι τα στοιχεία της λίστας πρέπει να έχουν το ίδιο μήκος.
- Συνήθως χρησιμοποιείται για να εκφράσει έναν πίνακα δεδομένων που λαμβάνουμε από την πραγματοποίηση ενός "πειράματος" στο οποίο καταγράφονται k μεταβλητές με n τιμές η κάθε μία

π.χ. μπορεί να καταγράψουμε τις τιμές των μεταβλητών gender, age, weight σε $n = 10$ άτομα,

```
> data1<-data.frame(gender=c("f","f","m","m"), age=c(24,23,25,20),  
weight=c(75,81,68,62))
```

```
> data1
```

	gender	age	weight
1	f	24	75
2	f	23	81
3	m	25	68
4	m	20	62

- Αν θέλουμε να χρησιμοποιήσουμε κάποια από τις μεταβλητές, όμοια με τον τύπο List, γράφουμε

```
> data1$age  
24 23 25 20
```

ή

```
> data1[[2]]  
24 23 25 20
```

- Αν θέλουμε να δημιουργήσουμε αντίγραφα των μεταβλητών που υπάρχουν μέσα στο data.frame και να δουλέψουμε με αυτά εκτελούμε

```
> attach(data1)
```

```
The following object(s) are masked from data1 ( position 3 ) :
```

```
age gender weight
```

- Με την **attach** δημιουργούνται νέες μεταβλητές με ονόματα ίδια με αυτά των μεταβλητών μέσα στο data.frame δηλ. στο παράδειγμα: age, gender, weight. (Αρκεί τα ονόματα των μεταβλητών του data.frame να μην έχουν δοθεί προηγουμένως σε άλλες μεταβλητές)

```
> age  
24 23 25 20
```

- Η εντολή **search()** δείχνει ποια data.frames (καθώς και λίστες, πακέτα) έχουν γίνει attached.

```
> search()  
[1] ".GlobalEnv"          "data1"              "package:stats"      "package:graphics"  
[5] "package:grDevices"   "package:utils"      "package:datasets"   "package:methods"  
[9] "Autoloads"           "package:base"
```

- Η εντολή **ls(αριθμός attached στοιχείου)** δείχνει ποιες μεταβλητές περιέχει το attached στοιχείο

```
> ls(2)  
"age"      "gender"   "weight"
```

Μπορούμε να σβήσουμε ένα attached στοιχείο με την εντολή **detach(όνομα στοιχείου)**

```
detach(data1)
```

Οι αρχικές μεταβλητές στο data.frame δεν επηρεάζονται αν μεταβάλλουμε τις νέες attached μεταβλητές, π.χ.

```
> age<-c(1,2,3,4)  
> age  
1 2 3 4  
  
> data1$age  
24 23 25 20
```

Εισαγωγή δεδομένων από αρχείο

- Το R μπορεί να διαβάσει δεδομένα από ένα αρχείο txt (που βρίσκεται στο *working directory* το οποίο αλλάζει από το *menu: file/change dir*), αρκεί τα δεδομένα μέσα στο αρχείο να είναι σε συγκεκριμένη μορφή.

Για παράδειγμα, το data2.txt θα πρέπει να έχει την μορφή (3 μεταβλητές, 4 cases):

age	weight	gender
20	72	male
22	65	female
18	71	female
24	78	male

και μπορεί να φορτωθεί ως data.frame:

```
> data2<-read.table("data2.txt",header=TRUE)
> data2
  age weight gender
1  20     72   male
2  22     65 female
3  18     71 female
4  24     78   male

> data2$age
20 22 18 24
```

Γραφήματα στο R

- Η απλούστερη εντολή κατασκευής γραφήματος είναι η plot:

```
x<-c(1,3,4,8,6)  
plot(x)
```

κατασκευάζει ένα γράφημα στο παράθυρο R Graphics (αφού το καθαρίσει) με πέντε σημεία, τα (1,1), (2,3), (3,4), (4,8), (5,6). Η plot συντάσσεται και με δύο διανύσματα:

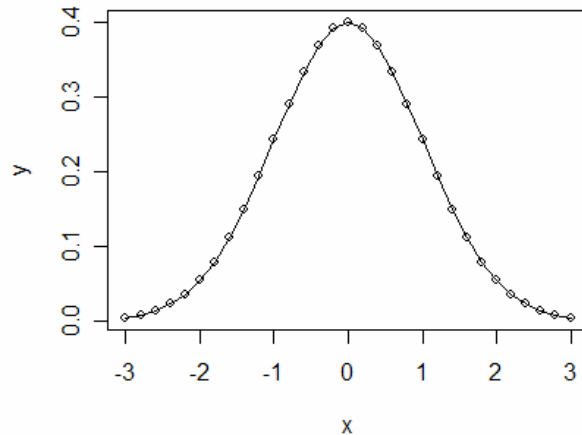
```
x<-c(1,4,8);y<-c(1,4,16)  
plot(x,y)
```

η παραπάνω κατασκευάζει ένα γράφημα με τρία σημεία, τα (1,1), (4,4), (8,16).

- Μέσα στην plot μπορούν να τεθούν διάφορα options (π.χ. `plot(x,y,options)`):

add=TRUE	<i>Forces the function to act as a low-level graphics function, superimposing the plot on the current plot (some functions only).</i>
axes=FALSE	<i>Suppresses generation of axes</i>
log="x" , log="y" , log="xy"	<i>Causes the x, y or both axes to be logarithmic</i>
type="p"	<i>Plot individual points (the default)</i>
type="l"	<i>Plot lines</i>
type="b"	<i>Plot points connected by lines (both)</i>
type="o"	<i>Plot points overlaid by lines</i>
type="h"	<i>Plot vertical lines from points to the zero axis (high-density)</i>
type="s" type="S"	<i>Step-function plots. In the first form, the top of the vertical defines the point; in the second, the bottom.</i>
type="n"	<i>No plotting at all. However axes are still drawn (by default) and the coordinate system is set up according to the data</i>
xlab=string, ylab=string	<i>Axis labels for the x and y axes</i>
main=string	<i>Figure title, placed at the top of the plot in a large font.</i>
sub=string	<i>Sub-title, placed just below the x-axis in a smaller font</i>

```
> x<-seq(-3,3,0.2)
> y<-1/(2*pi)^0.5*exp(-x^2/2)
> plot(x,y,type="o")
```



- Σημεία στο παράθυρο R-Graphics μπορούν να προστεθούν με την εντολή **points**

points(x,y) (προσθέτει τα σημεία $(x[1], y[1]), \dots, (x[n], y[n])$)

lines(x,y) (προσθέτει τα σημεία όπως και η **points** και τα ενώνει)

text(x, y, labels) (προσθέτει τα **labels[i]** στα σημεία που καθορίσουν οι $(x[i], y[i])$)

(οι παραπάνω τρεις εντολές δεν καθαρίζουν το παράθυρο R-Graphics, αλλά επιδρούν πάνω στο υπάρχον γράφημα)

Στατιστικές συναρτήσεις στο R

- Στο R (default πακέτο) είναι διαθέσιμες οι κατανομές

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

- Για να πάρουμε την σ.π.π. (density) ή την σ.κ. (CDF) στο **x** γράφουμε **dyyy (x, παράμετροι)** ή **pyyy (x, παράμετροι)** αντίστοιχα όπου **yyy** είναι το όνομα της κατανομής,

π.χ. οι εντολές

```
> dnorm(0, 1, 2)
0.1760327
```

```
> pnorm(0, 0, 1)
0.5
```

δίνουν την $f(0)$ και την $F(0)$, όπου f και F είναι η σ.π.π. και σ.κ. της κανονικής κατανομής με $\mu = 1, \sigma = 2$ και $\mu = 0, \sigma = 1$ αντίστοιχα.

- Για την αντίστροφη σ.κ. (quantile) στο **p** γράφουμε **qyyy (p, παράμετροι)** όπου **yyy** είναι το όνομα της κατανομής,

π.χ. η εντολή

```
> qnorm(0.975, 0, 1)
1.959964
```

δίνει την $F^{-1}(0.975)$, όπου F είναι η σ.κ. της κανονικής κατανομής με $\mu = 0, \sigma = 1$

- Για να πάρουμε n τυχαίους αριθμούς γράφουμε **`rnyyy(n, παράμετροι)`** όπου **`yyyy`** είναι το όνομα της κατανομής, π.χ. η εντολή

```
> rnorm(5, 0, 1)
-0.5966811  0.6406156  1.5487987 -0.2232449  0.7164562
```

δίνει 5 τυχαίους αριθμούς από την κανονική κατανομή με $\mu = 0$, $\sigma = 1$

- Με την εντολή **`summary`** μπορούμε να πάρουμε τις τιμές διάφορων στατιστικών συναρτήσεων ενός δείγματος:

```
> x<-c(1,2,4)
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.000  1.500   2.000   2.333  3.000   4.000
```

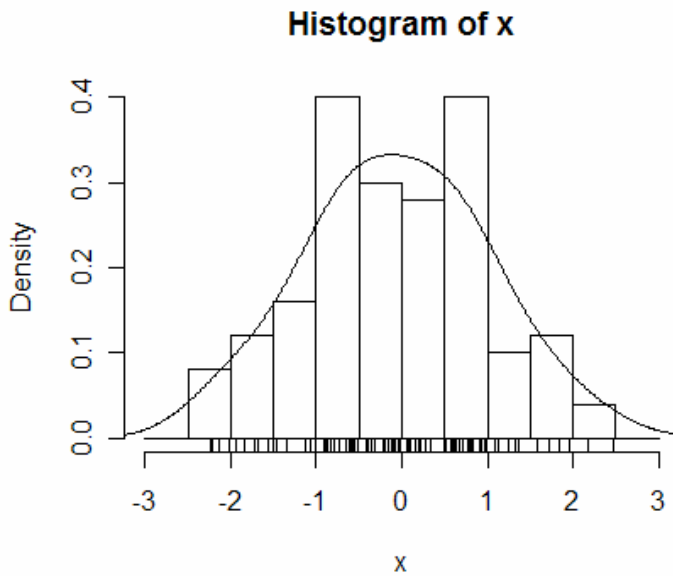
- Επίσης μπορούμε να φτιάξουμε ένα *ιστόγραμμα δεδομένων* με την **`hist`**:

```
> x<-rnorm(100,0,1)
> hist(x)
```

- Η εντολή **`hist(x, prob=TRUE)`** κατασκευάζει και πάλι το ιστόγραμμα, αυτή τη φορά έτσι ώστε το εμβαδό κάτω από αυτό να είναι 1 (δηλ. οι μπάρες εκφράζουν σχετικές συχνότητες)
- Επίσης η εντολή **`hist(x, seq(from, to, step))`** κατασκευάζει το ιστόγραμμα με κλάσεις αυτές που καθορίζονται από την **`seq`**

- Η εντολή `lines(density(x,bw=0.6))` προσθέτει στο ιστόγραμμα μια εκτίμηση της σ.π.π. της κατανομής (density estimation) από την οποία προέρχονται τα δεδομένα (το `bw` εκφράζει το πόσο «λεία» θα ληφθεί η εκτίμηση της σ.π.π.).
- Επίσης η εντολή `rug` προσθέτει στον οριζόντιο άξονα μικρές κάθετες γραμμές στα σημεία που εκφράζουν τις τιμές των δεδομένων.

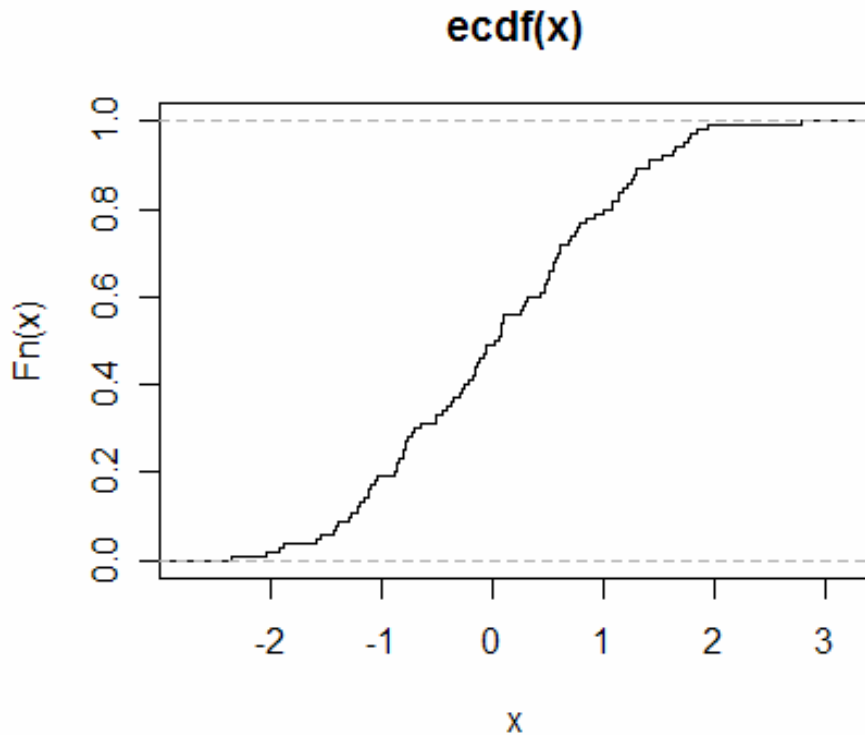
```
> x<-rnorm(100,0,1)
> hist(x,seq(-3,3,0.5),prob=T)
> lines(density(x,bw=0.5))
> rug(x)
```



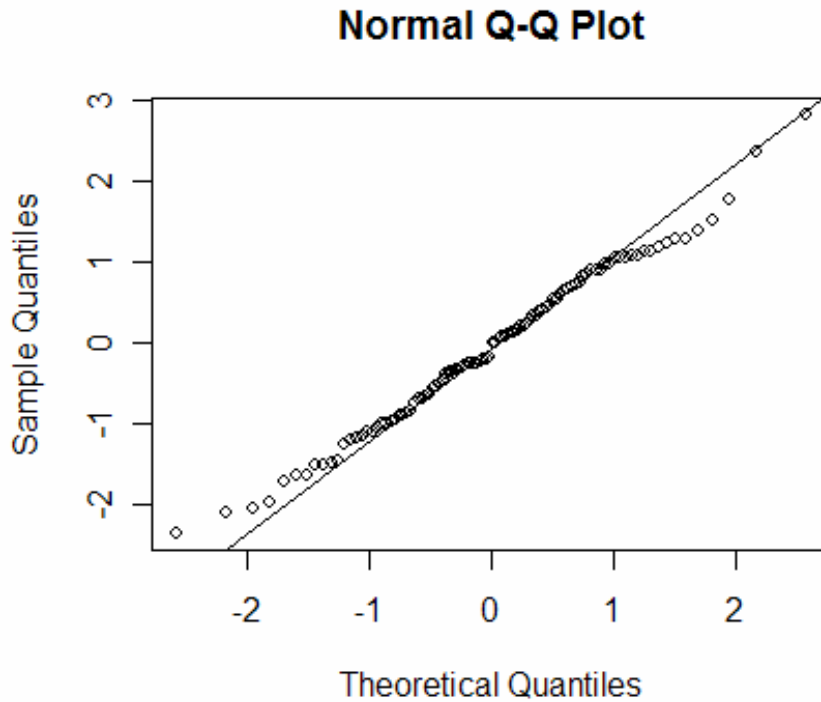
- Κατασκευή εμπειρικής συνάρτησης κατανομής:

```
> x<-rnorm(100,0,1)
```

```
> plot(ecdf(x), do.points=FALSE, verticals=TRUE)
```



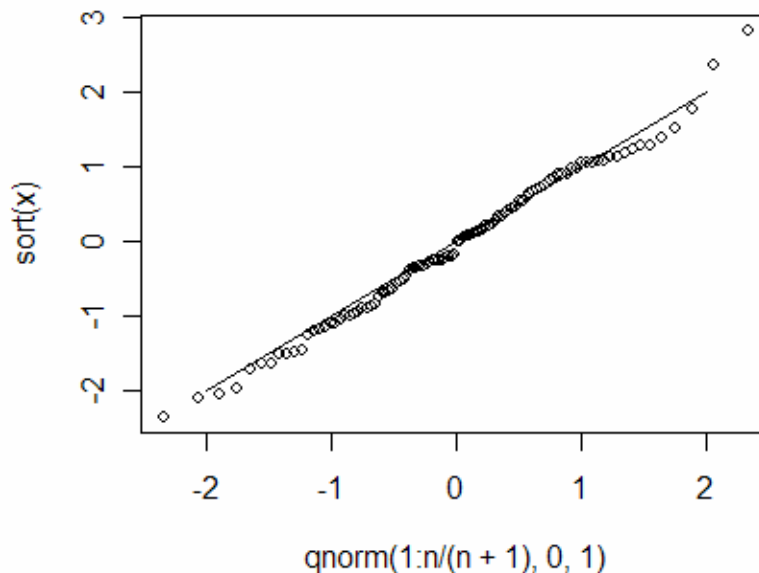
- Κατασκευή Q-Q Plot για την κανονική
> qqnorm(x) ; qqline(x)



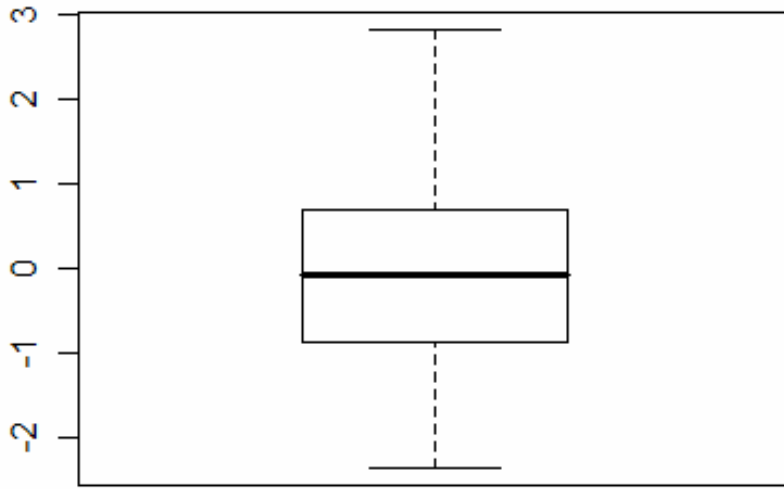
- Προφανώς μπορούμε να κατασκευάσουμε Q-Q plot χρησιμοποιώντας μόνο την εντολή `plot` όπως φαίνεται παρακάτω (στη θέση της `qnorm` μπορούμε να βάλουμε οποιαδήποτε κατανομή).

Στο ακόλουθο Q-Q plot θεωρούμε $\mu = 0, \sigma = 1$ (δεν εκτιμώνται από το δείγμα).

```
> n<-length(x)
> plot(qnorm(1:n/(n+1),0,1),sort(x))
> lines(c(-2,2),c(-2,2))
```



- Ένα θηκόγραμμα μπορεί να κατασκευαστεί με την εντολή **boxplot**
> boxplot(x)



Έλεγχοι Στατιστικών Υποθέσεων

- Το R μπορεί να εκτελέσει με μία εντολή και στατιστικούς ελέγχους. Για παράδειγμα θα παρουσιάσουμε πως μπορεί να γίνει το *Kolmogorov-Smirnov test* και το *t-test* (για περισσότερα στατιστικά τεστ ανατρέξτε στο manual)

- Π.χ. για το *Kolmogorov – Smirnov test* (έλεγχος καλής προσαρμογής των δεδομένων στην κανονική) εκτελούμε:

```
> ks.test(x, "pnorm", mean = mean(x), sd = sqrt(var(x)))
```

```
One-sample Kolmogorov-Smirnov test
```

```
data: x
```

```
D = 0.041, p-value = 0.996
```

```
alternative hypothesis: two-sided
```

- *T*-τεστ για δύο ανεξάρτητους πληθυσμούς:

```
> x<-rnorm(100,0,1)
> y<-rnorm(100,0.5,1)
> t.test(x,y,var.equal=TRUE)
```

Two Sample t-test

data: x and y

t = -4.2234, df = 198, p-value = 3.667e-05

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-0.7935844 -0.2883849

sample estimates: mean of x mean of y

0.0848393 0.6258239

- Επίσης έχει ενδιαφέρον να δούμε πως μπορεί να εφαρμοστεί ένα μοντέλο παλινδρόμησης (στο παράδειγμα κατασκευάζουμε πρώτα τεχνητά από προσομοίωση τα δεδομένα)

```
x<-runif(100,0,1);  
e<-rnorm(100,0,0.5);  
y<-2+3*x+e  
data1<-data.frame(y=y,x=x)  
model1<-lm(y ~ x, data = data1)
```

και στη συνέχεια μπορούμε να χρησιμοποιήσουμε τις εντολές:

```
anova(model1)  
coef(model1)  
formula(model1)  
plot(model1)  
print(model1)  
summary(model1)
```

Εντολές επανάληψης (for, while), δεσμευμένης εκτέλεσης εντολών (If)

Η γλώσσα του R παρέχει, όπως κάθε γλώσσα προγραμματισμού, εντολές επανάληψης και εντολές δεσμευμένης εκτέλεσης εντολών που λειτουργούν με τον κλασικό τρόπο:

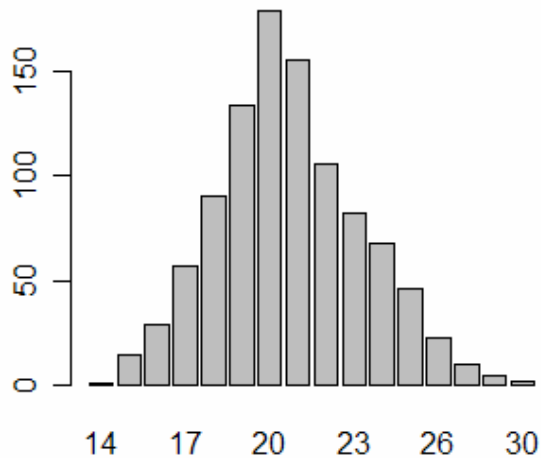
```
> x<-rnorm(10,0,1);s<-0;
> for(i in 1:10){s<-s+x[i]; print(c(i,x[i],s))}
1. -1.244098 -1.244098
2. -1.493813 -2.737910
3. 0.2906177 -2.4472924
4. 0.07361697 -2.37367542
5. 0.3541828 -2.0194926
6. 1.3165771 -0.7029155
7. 2.342201 1.639286
8. 0.1277773 1.7670629
9. 0.06044127 1.82750413
10. -1.0448327 0.7826714

> x<-rnorm(100000,0,1);s<--10;
> for(i in 1:100000){if(x[i]>s){s<-x[i];print(c(i,s))}}
```

```
1.      0.07391807
8.      0.7183461
10.     1.939915
56.     3.317352
371.    3.484235
4650.   3.535515
5161.   3.606771
9976.   3.77993
34006.  3.785858
35897.  4.033259
62950.  4.21976
```

- Για conditional loops χρησιμοποιούμε την εντολή **while(condition) expression**

```
> x<-1:1000
> for(i in 1:1000){
+   s<-0;n<-0;
+   while(s<10){s<-s+runif(1,0,1);n<-n+1};
+   x[i]<-n
+ }
> barplot(table(x))
```



- Επίσης μπορούμε να κατασκευάσουμε τις δικές μας συναρτήσεις

```
> records<-function(x) {  
+   s<--Inf;a<-c();  
+   for(i in 1:length(x))  
+     {if(x[i]>s){s<-x[i];a<-c(a,s)}};a  
+ }
```

και μετά να τις καλέσουμε

```
> records(rnorm(1000,0,1))  
0.7163191 0.9811541 1.5738380 1.7048937 3.2157284
```

Φόρτωση νέων πακέτων (Packages) με εντολές

- Μόλις εκκινήσει το R φορτώνει εντολές και συναρτήσεις από τα default πακέτα (η εντολή `search()` δείχνει ποια πακέτα έχουν φορτωθεί).
- Υπάρχει η δυνατότητα να φορτώσουμε και άλλες εντολές και συναρτήσεις που περιέχονται σε διάφορα εξειδικευμένα πακέτα. Ήδη υπάρχουν δεκάδες πρόσθετα πακέτα που έρχονται μαζί με το R (βρίσκονται στον φάκελο `../R/library`).
- Για παράδειγμα, μπορούμε να φορτώσουμε το πακέτο MASS (`menu/packages/load package/ MASS`) το οποίο έχει επιπλέον εντολές (μπορούμε να δούμε λεπτομέρειες για το πακέτο αυτό αν πάμε στο φάκελο `..program_files/R/library/MASS`).

Μία από τις επιπλέον εντολές π.χ. είναι η `fitdistr` η οποία εκτιμά παραμέτρους κατανομών από δεδομένα (με τη μέθοδο μέγιστης πιθανοφάνειας).

```
> x<-rnorm(100,3,5)
> fitdistr(x,"normal")

      mean      sd
3.7013648 4.8047403
(0.4804740) (0.3397464)
```

```

> x <- rgamma(100, shape = 5, rate = 0.1)
> fitdistr(x, "gamma")

      shape      rate
5.10825557 0.10903833
(0.69972396) (0.01569367)

> x3 <- rweibull(100, shape = 4, scale = 100)
> fitdistr(x3, "weibull")

      shape      scale
3.9267237 99.9342614
( 0.3018964) ( 2.6888457)

> x2 <- rt(250, df = 9)
> fitdistr(x2, "t", df = 9)

      m      s
0.02210409 0.99828578
(0.06934510) (0.05096042)

```

- Στο διαδίκτυο υπάρχουν εκατοντάδες πακέτα που καλύπτουν σχεδόν όλες τις περιοχές της εφαρμοσμένης στατιστικής (π.χ. `extRemes` για στατιστική ανάλυση ακραίων παρατηρήσεων).
- Τα πακέτα αυτά μπορούν να προστεθούν στον φάκελο `library` του R (ώστε στη συνέχεια να φορτωθούν και χρησιμοποιηθούν) από το `menu/install package/`. Επιλέγουμε `mirror`, π.χ. `USA(CA 1)` και στη συνέχεια επιλέγουμε από την λίστα που εμφανίζεται το πακέτο που θέλουμε να εγκαταστήσουμε (π.χ. `extRemes`).
- Αφού εγκατασταθεί το πακέτο μπορούμε να δούμε λεπτομέρειες για αυτό στα `help files` που το συνοδεύουν (π.χ. για το `extRemes` στον φάκελο `..R/library/extRemes`).
- Αν θέλουμε να το χρησιμοποιήσουμε το φορτώνουμε (`menu/packages/load package/MASS`).

Στατιστική ανάλυση ακραίων παρατηρήσεων - Το πακέτο ExtRemes

- Για την στατιστική επεξεργασία δεδομένων από ακραίες παρατηρήσεις θα χρειαστούμε εντολές που περιέχονται στο εξωτερικό πακέτο ExtRemes.
- Αρχικά κάνουμε install το πακέτο αυτό με την διαδικασία που περιγράφεται παραπάνω και το φορτώνουμε – load package (για να φορτωθεί πλήρως μπορεί να ζητηθεί να εγκατασταθούν και άλλα πακέτα, θα πρέπει να κάνουμε install και αυτά με την ίδια διαδικασία).
- Το πακέτο αυτό προσφέρει αρκετές νέες εντολές. Για τις ανάγκες των διαλέξεων με αντικείμενο "*Θεωρία Ακραίων Τιμών*" θα εξετάσουμε στη συνέχεια πολύ περιληπτικά κάποιες από αυτές (πληρέστερη περιγραφή των εντολών του πακέτου μπορεί να βρεθεί στα *help files* που βρίσκονται στον φάκελο library/extRemes)

- Έστω ένα σύνολο δεδομένων

$$x_1, x_2, \dots, x_m$$

και επιθυμούμε, σύμφωνα με τα δύο πρώτα κεφάλαια των σημειώσεων της "Θεωρίας Ακραίων Τιμών", να μελετήσουμε *την μορφή της δεξιάς ουράς* των δεδομένων.

- Αν τα δεδομένα βρίσκονται σε ένα αρχείο π.χ. `data_lr.txt` (που βρίσκεται στο *working directory*) με τη μορφή (περιέχει μία μόνο μεταβλητή)

```
logret
1.12
0.82
...
-1.3
```

μπορούν να φορτωθούν (ως `data.frame`) για επεξεργασία στο R εκτελώντας

```
> data0<-read.table("data_lr.txt ", header=TRUE)
> x<-data0[[1]] #thetoume x ta dedomena tis prwtis kai monadikhs metablhtis
```

και συνεχίζουμε την επεξεργασία χρησιμοποιώντας την `x`.

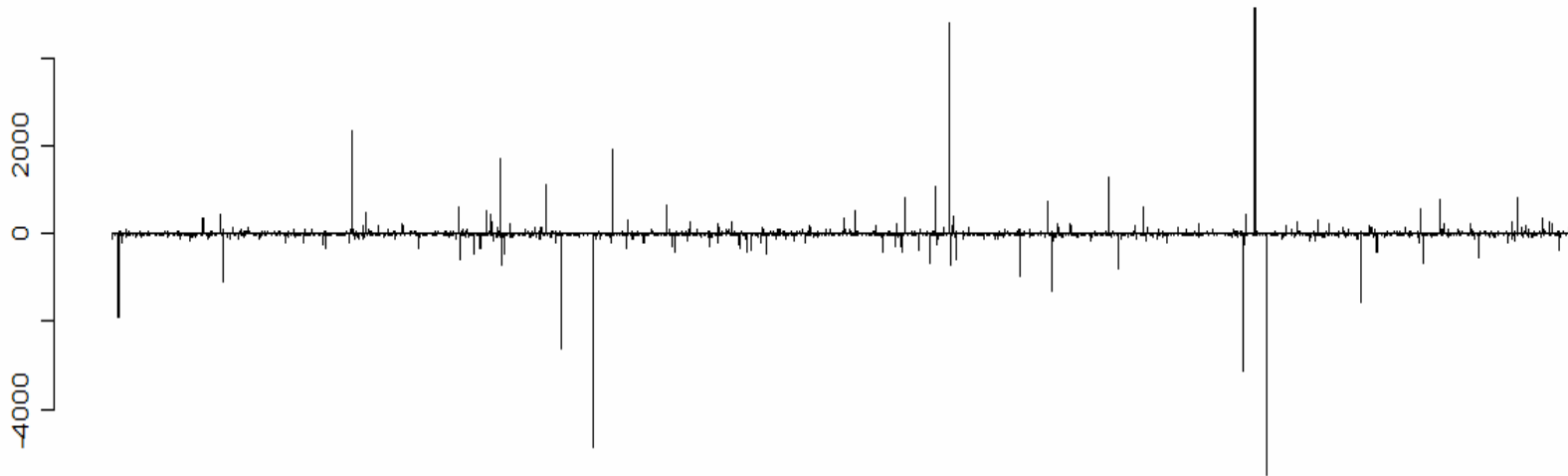
- Στη συνέχεια θα θεωρήσουμε ως δεδομένα *τυχαίους αριθμούς που θα παράγουμε από γνωστή κατανομή με δεδομένες παραμέτρους*.
- Ακολούθως, θα προβούμε στην στατιστική ανάλυση αυτών των δεδομένων (*προσποιούμενοι ότι δεν γνωρίζουμε από πού προέρχονται*) για να διερευνήσουμε την αποτελεσματικότητα των μεθόδων που θα χρησιμοποιήσουμε.
- Θα παράγουμε 10000 αριθμούς από την κατανομή Cauchy με παραμέτρους $l = 0$ (location) και $s = 3$ (scale). Η Cauchy έχει σ.π.π.

$$f(x) = \frac{1}{\pi s} \left(1 + \left(\frac{x-l}{s} \right)^2 \right)^{-1}, x \in \mathbb{R}$$

και, όπως γνωρίζουμε, έχει βαριά δεξιά ουρά με παράμετρο σχήματος $\xi = 1$.

- Ας αγνοήσουμε λοιπόν ότι τα δεδομένα προέρχονται από αυτή την κατανομή και ας περάσουμε στην ανάλυσή τους με τις μεθόδους που έχουμε παρουσιάσει.

```
n<-10000;set.seed(10)
data<-rcauchy(n, location = 0, scale =3);
barplot(data);
```



- Παρατηρούμε ότι μερικές τιμές είναι πάρα πολύ μεγάλες θετικές και πολύ μεγάλες αρνητικές σε σχέση με τις υπόλοιπες (στη συνέχεια θα μας απασχολήσουν οι πολύ μεγάλες θετικές τιμές).

- Λαμβάνουμε τις τιμές κάποιων περιγραφικών στατιστικών συναρτήσεων

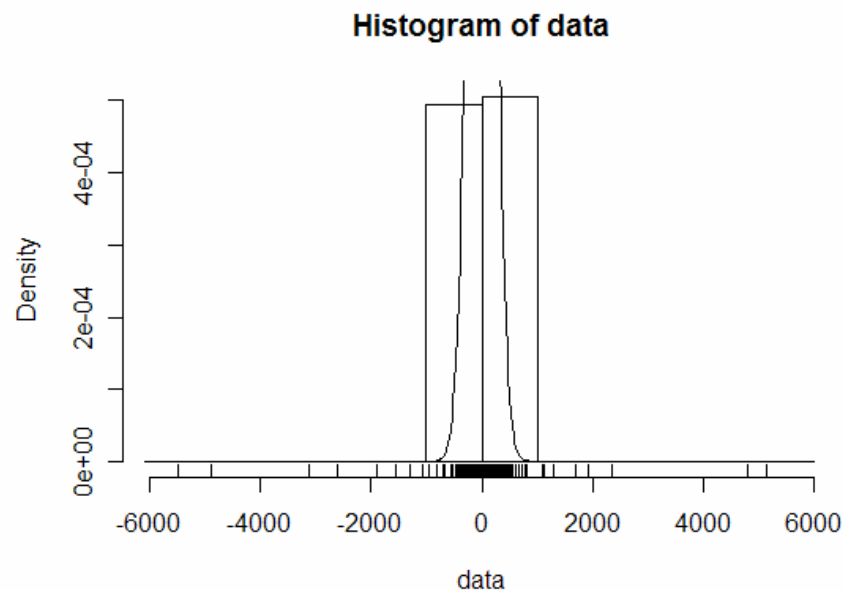
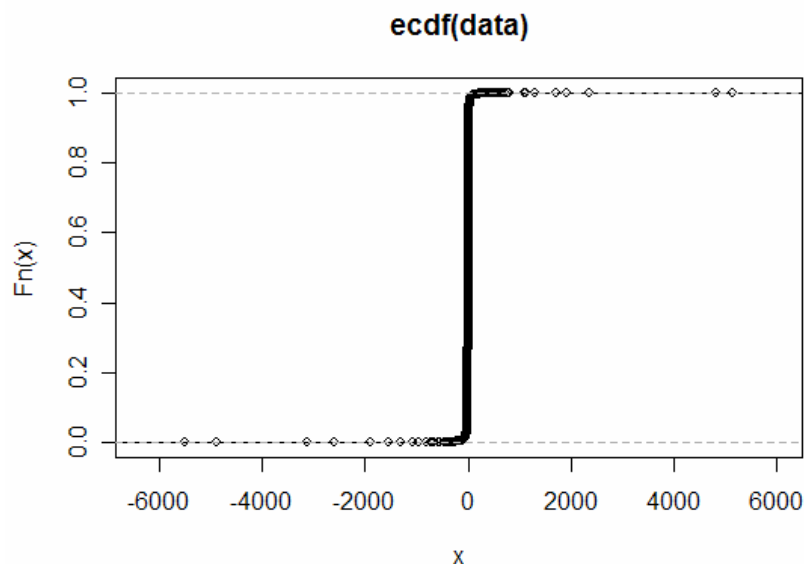
```
summary(data)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-5.491e+03	-2.801e+00	4.679e-02	-1.031e-01	3.119e+00	5.135e+03

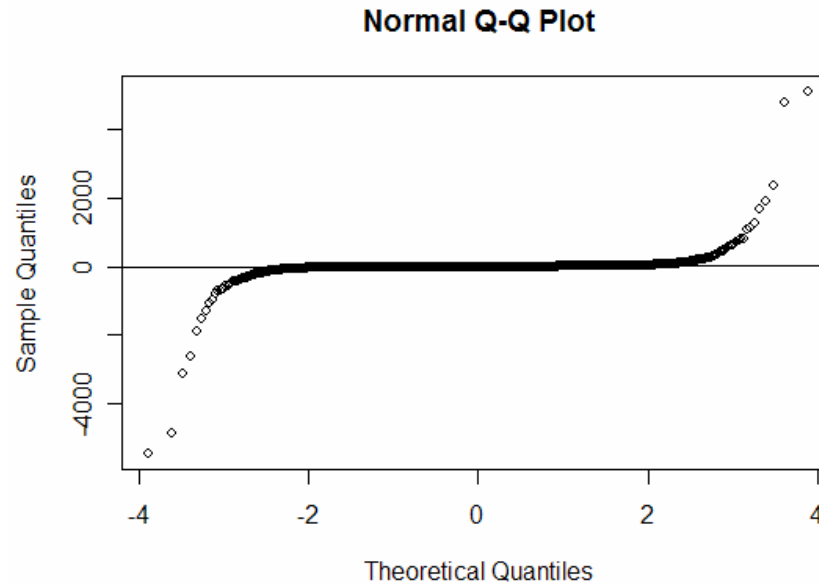
- Επίσης μπορούμε να δούμε το γράφημα της εμπειρικής συνάρτησης κατανομής και το ιστόγραμμα

```
plot(ecdf(data)) ;
```

```
hist(data, prob=TRUE) ; lines(density(data, bw=200)) ; rug(data)
```



- Επίσης ένα Q-Q plot ως προς την κανονική κατανομή θα είναι
`qqnorm(data) ; qqline(data)`

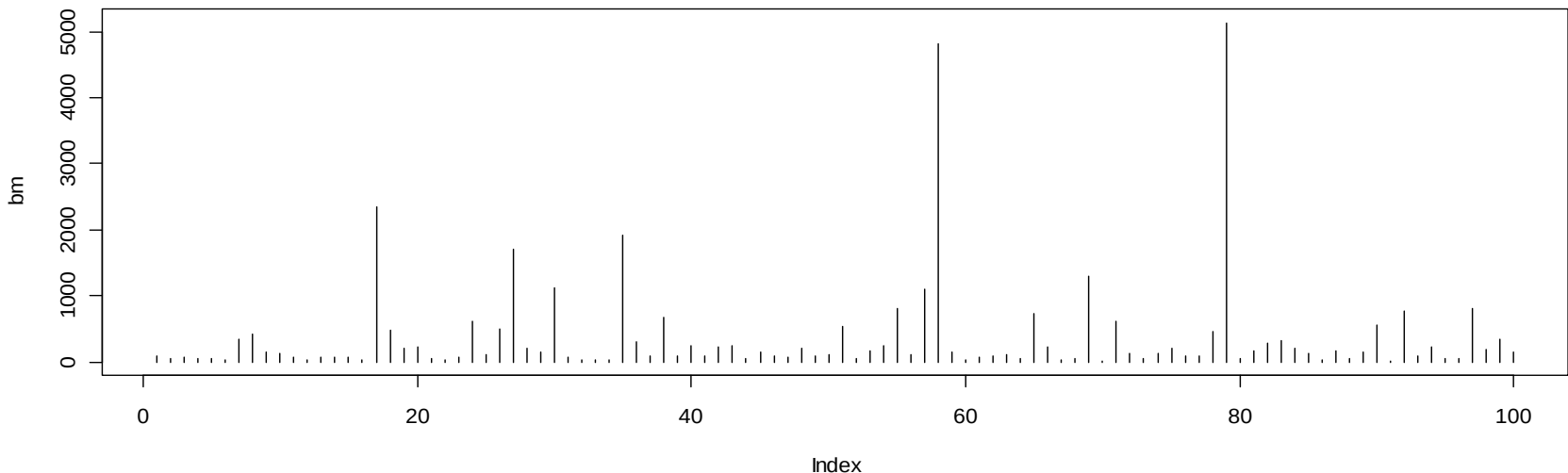


- Και στα τρία παραπάνω διαγνωστικά γραφήματα παρατηρούμε ότι σχεδόν το σύνολο των παρατηρήσεων βρίσκεται κοντά και γύρω από το 0, ενώ υπάρχουν και μερικές παρατηρήσεις που απέχουν (αριστερά ή δεξιά) πάρα πολύ από το 0 (για αυτό και τα παραπάνω γραφήματα έχουν μια ασυνήθιστη, ιδιάζουσα μορφή).

Εφαρμογή της μεθόδου Block Maxima

- Θα πρέπει να χωρίσουμε το δείγμα των 10000 παρατηρήσεων σε blocks και να πάρουμε την μέγιστη παρατήρηση από κάθε block.
- Ας χωρίσουμε το δείγμα σε $k=100$ blocks από $m = n/k = 100$ παρατηρήσεις το καθένα. Η μέγιστη παρατήρηση από κάθε block μπορεί π.χ. να ληφθεί ως εξής

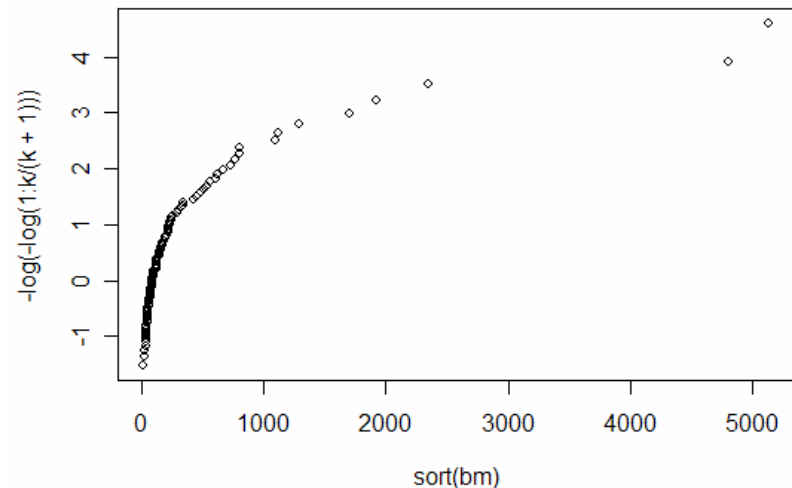
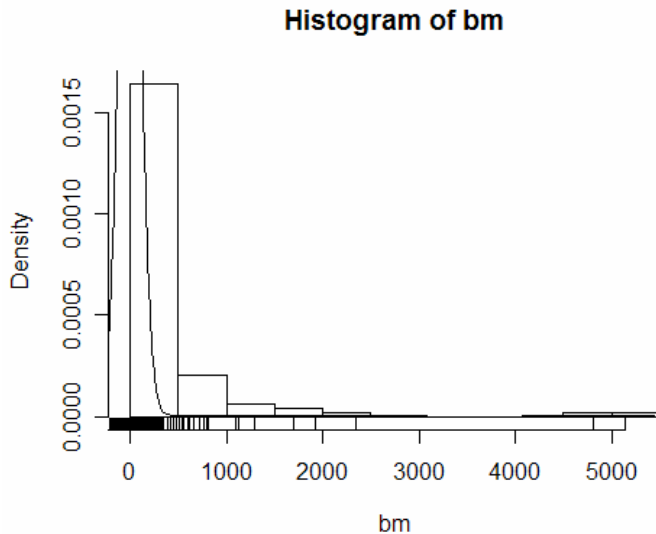
```
k=100; m=100; bm<-rep(0,k)
for(i in 1:k){bm[i]<-max(data[(i-1)*m+1:(i*m)])}
plot(bm,type="h")
```



- Τα block maxima θα ακολουθούν προσεγγιστικά την κατανομή **GEV** με κάποιες άγνωστες παραμέτρους μ , σ , ξ .
- Σκοπός μας είναι
 - (1) η *εκτίμηση των παραμέτρων* αυτών από τα block maxima και στη συνέχεια
 - (2) η *εκτίμηση της στάθμης απόδοσης* z_p για διάφορες τιμές του p .
 - (3) Ο έλεγχος *καλής προσαρμογής* των block maxima στην GEV (ώστε όλες οι προηγούμενες εκτιμήσεις να θεωρούνται αξιόπιστες).

- Αρχικά κατασκευάζουμε το ιστόγραμμα και ένα Gumbel Q-Q plot των block maxima (δηλ. γράφημα των σημείων $(Y_{(i)}, -\ln(-\ln \frac{i}{k+1}))$, $i = 1, 2, \dots, k$):

```
hist(bm, prob=TRUE);lines(density(data, bw=100));rug(data)
k<-length(bm); plot(sort(bm), -log(-log(1:k/(k+1)))) #Gumbel qqplot
```



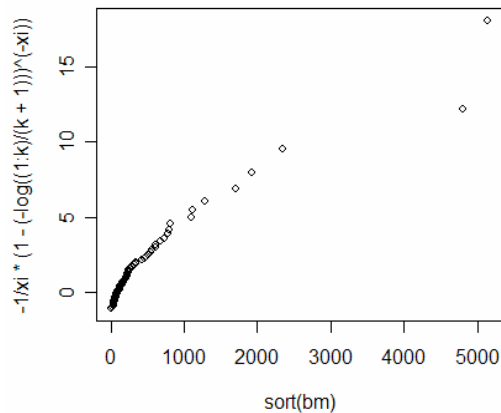
- Από το ιστόγραμμα διαπιστώνουμε ότι τα block maxima μάλλον έχουν βαριά δεξιά ουρά.
- Το γράφημα των σημείων στο Gumbel Q-Q plot φαίνεται να είναι κοίλο και άρα το ξ πρέπει να είναι θετικό.

- Όπως έχουμε περιγράψει, μπορούμε φτιάξουμε το Q-Q plot της GEV με $\mu=0$, $\sigma=1$,

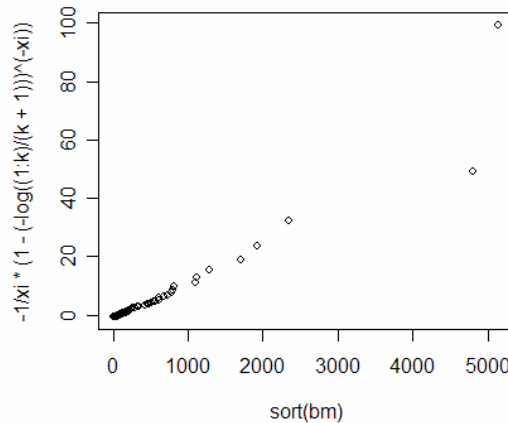
$$(Y_{(i)}, -\frac{1}{\hat{\xi}} (1 - (-\ln \frac{i}{k+1})^{-\hat{\xi}})), i = 1, 2, \dots, k$$

δοκιμάζοντας διάφορες τιμές του ξ έως ότου τα σημεία περίπου ευθυγραμμιστούν:

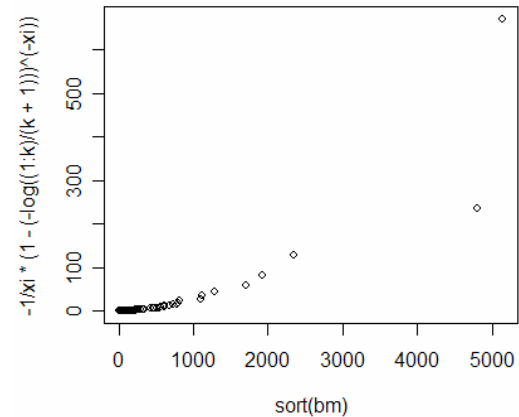
```
xi<-1; k<-length(bm);  
plot(sort(bm), -1/xi*(1-(-log((1:k)/(k+1)))^(-xi))) #gev qqplot
```



$\xi=0.5$,



$\xi=1$,



$\xi=1.5$

- Από τα παραπάνω γραφήματα βλέπουμε ότι το ξ μάλλον θα είναι κοντά στο 1.

- Επίσης, όπως έχουμε αναφέρει, τα σημεία θα βρίσκονται πάνω στην ευθεία $y = (x - \mu)/\sigma$. Από το γεγονός αυτό μπορούν να εκτιμηθούν (χοντρικά) και τα μ, σ .
- Για πιο ασφαλείς όμως εκτιμήσεις θα χρησιμοποιήσουμε την **μέθοδο μέγιστης πιθανοφάνειας**. Η εκτίμηση των μ, σ και ξ από τα block maxima μέσω της μεθόδου μέγιστης πιθανοφάνειας γίνεται με την εντολή **gev.fit** (πακέτο extRemes):

```
a<-gev.fit(bm)
```

```
$conv (The convergence code, a zero indicates successful convergence)  
0
```

```
$nllh (The negative logarithm of the likelihood evaluated at the maximum likelihood estimates)  
657.355
```

```
$mle  
88.1670475 86.9464949 0.9256367
```

```
$se  
9.7428471 12.5447776 0.1181692
```

Επομένως $\hat{\mu} = 88.1670475$, $\hat{\sigma} = 86.9464949$ $\hat{\xi} = 0.9256367$ εκτιμήσεις των τυπικών αποκλίσεων τους 9.7428471 12.5447776 0.1181692 αντίστοιχα με δ.ε.:

$$(\hat{\mu} \pm \sqrt{s_{1,1}} z_{a/2}), (\hat{\sigma} \pm \sqrt{s_{2,2}} z_{a/2}), (\hat{\xi} \pm \sqrt{s_{3,3}} z_{a/2}).$$

- Εκτελώντας `print(a)` μπορούμε να πάρουμε περισσότερα στοιχεία για τις ε.μ.π.

Για παράδειγμα η εντολή

`a$cov`

	[,1]	[,2]	[,3]
[1,]	94.9230691	103.1574992	-0.10248966
[2,]	103.1574992	157.3714445	0.51549154
[3,]	-0.1024897	0.5154915	0.01396397

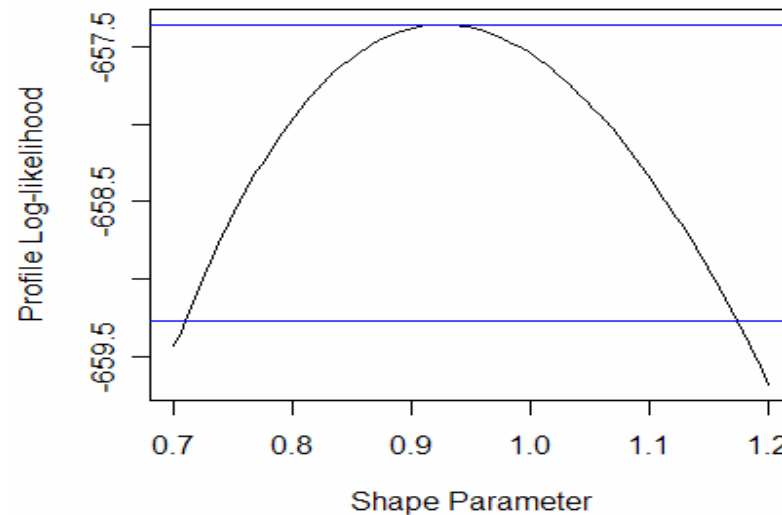
επιστρέφει τον εκτιμημένο πίνακα διασποράς – συνδιακύμανσης των τριών ε.μ.π.

- Ένα δ.ε. συντ. $1-a = 95\%$ για το ξ μέσω της profile Deviance Function είναι το

$$\{\xi : D_p(\xi) \leq \chi_1^2(a)\}, \quad \text{όπου} \quad D_p(\xi) = 2(l(\hat{\mu}, \hat{\sigma}, \hat{\xi}) - \max_{\mu, \sigma} l(\mu, \sigma, \xi)) \sim \chi_1^2$$

Με την ακόλουθη εντολή κατασκευάζεται το γράφημα της συνάρτησης $\max_{\mu, \sigma} l(\mu, \sigma, \xi)$ (*Profile log-likelihood*) για $\xi \in (0.7, 1.2)$ μαζί με την γραμμή $l(\hat{\mu}, \hat{\sigma}, \hat{\xi}) - \chi_1^2(a)/2$:

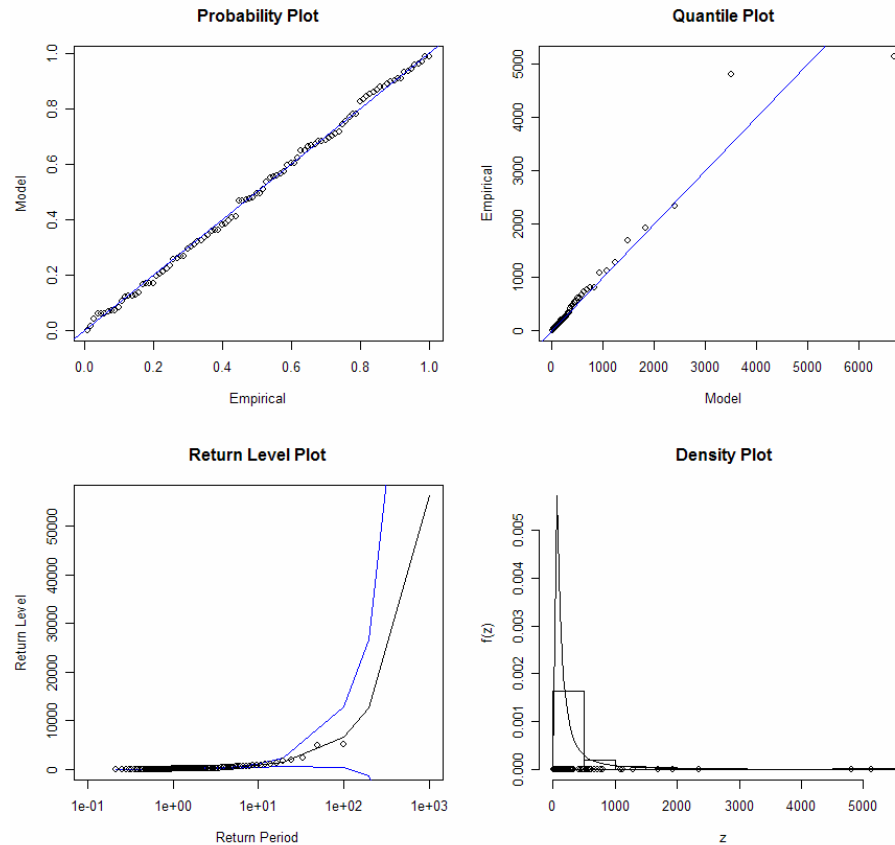
```
gev.profxi(a, 0.7, 1.2)
```



- Φαίνεται από το σχήμα ότι ένα δ.ε. 95% για το ξ είναι από 0.72 έως 1.18 περίπου και επομένως πρόκειται σίγουρα για δεδομένα από αρκετά βαριά δεξιά ουρά.

- Κατασκευάζουμε και τα *διαγνωστικά γραφήματα* για να εξετάσουμε την καλή προσαρμογή των block maxima στην κατανομή GEV. Ακολουθεί το γράφημα της στάθμης απόδοσης (εκτίμηση του $\mathbf{z}_p$ για διάφορες τιμές του $1/p$, μαζί με δ.ε.) και το ιστόγραμμα των block maxima μαζί με την σ.π.π. της GEV

`gev.diag(a)`



- Από τις εκτιμήσεις των μ , σ , ξ μπορούμε να εκτιμήσουμε και τη *στάθμη απόδοσης για $1/p$ χρονικές περιόδους* (1 block = μία χρονική περίοδο): (π.χ. $p = 0.1$)

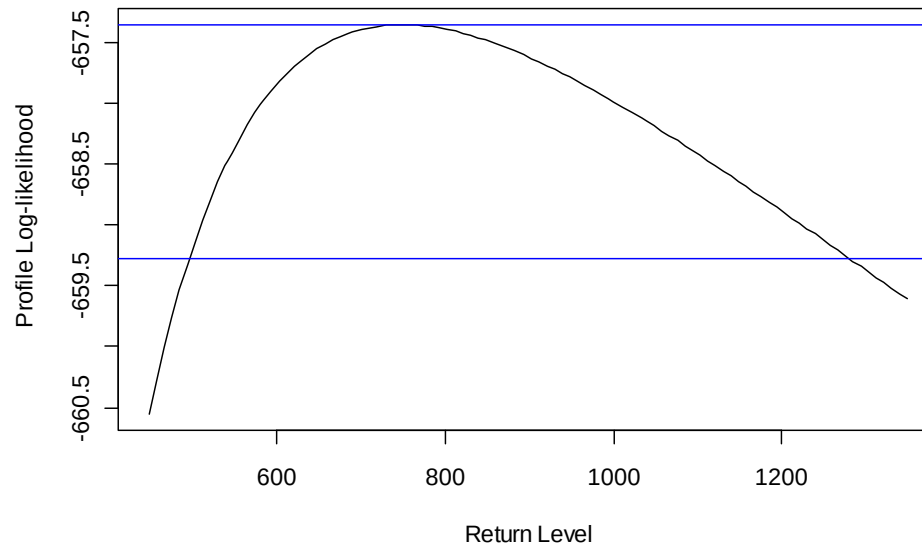
$$\hat{z}_p = \hat{\mu} - \frac{\hat{\sigma}}{\hat{\xi}} (1 - e^{\hat{\xi} w_p}), \quad w_p = -\ln(-\ln(1-p)).$$

```
mu<-a$mle[1];sigma<-a$mle[2];xi<-a$mle[3]
p=0.1;w=-log(-log(1-p));
zp<-mu-sigma/xi*(1-exp(xi*w));
print(zp)

748.3837
```

- Επίσης μπορούμε να εκτιμήσουμε την *στάθμη απόδοσης* και από την *profile log-likelihood*: (για $1/p = 10$):

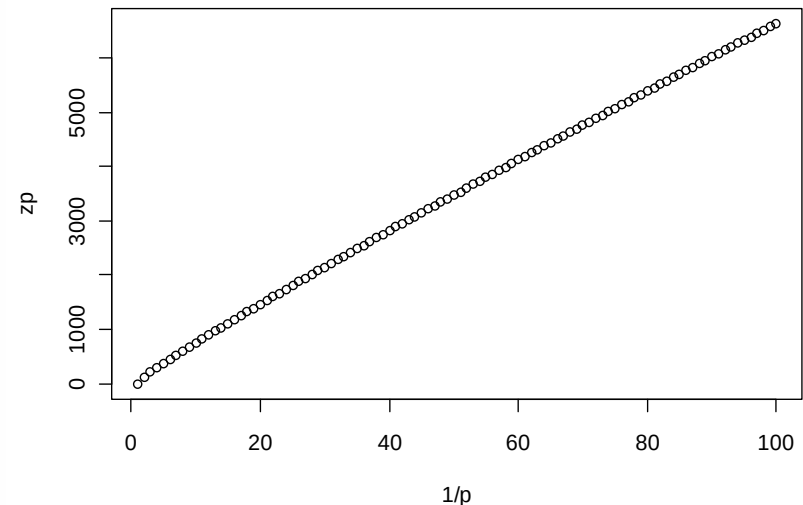
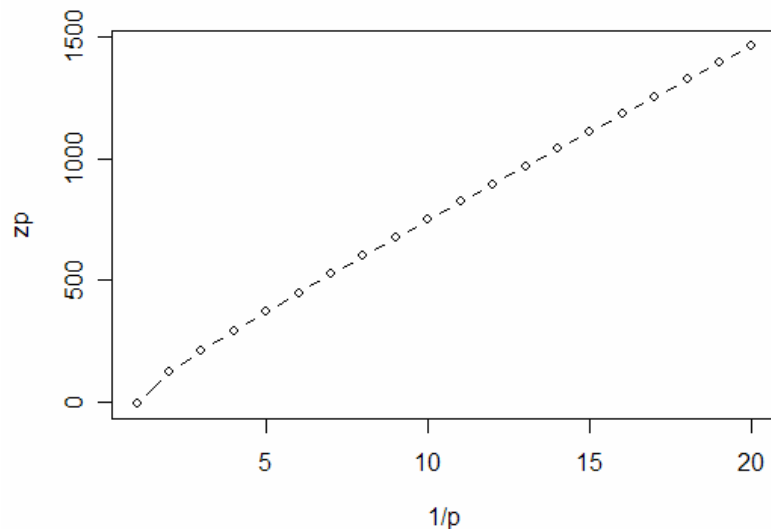
`gev.prof(a, 1/p, zp-300, zp+600)`



- Η εκτίμηση του z_p για $1/p = 10$ είναι περίπου 748 (με δ.ε. 95% από 500 έως 1300 περίπου):
 - Εκτιμούμε ότι ένα block maximum υπερβαίνει την τιμή 748 με πιθανότητα $p = 1/10$
 - ή ισοδύναμα ένα block maximum υπερβαίνει την τιμή 748 κατά μέσο όρο κάθε 10 χρονικές περιόδους – blocks

- Επίσης μπορούμε να κατασκευάσουμε και το γράφημα της στάθμης απόδοσης (εκτίμηση του z_p για διάφορες τιμές του $1/p$)

```
p<-1/1:20;w=-log(-log(1-p));
zp<-mu-sigma/xi*(1-exp(xi*w));
plot(1/p,zp,type="b")
```



Από το δεύτερο γράφημα προκύπτει π.χ. ότι η μεταβλητή που μελετάμε λαμβάνει τιμές μεγαλύτερες της στάθμης 5000 κατά μέσο όρο κάθε 75 χρονικές περιόδους περίπου.

Η μέθοδος POT (Peaks Over Threshold)

- Θα βασιστούμε στις υπερβάσεις των 10000 αρχικών δεδομένων πάνω από ένα υψηλό κατώφλι u .
- Οι υπερβάσεις αυτές θα ακολουθούν προσεγγιστικά μια κατανομή GPD με παραμέτρους $\tilde{\sigma}$, ξ .
- Σκοπός μας είναι η
 - (1) *εκτίμηση των παραμέτρων $\tilde{\sigma}$, ξ*
 - (2) *η εκτίμηση της στάθμης απόδοσης x_m για διάφορες τιμές του m .*
 - (3) *η εξέταση της καλής προσαρμογής των υπερβάσεων στην GPD (ώστε οι εκτιμήσεις που κάνουμε να θεωρούνται αξιόπιστες).*

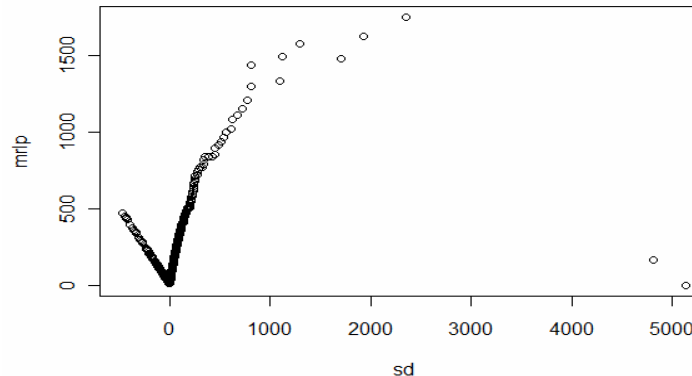
- Θα πρέπει πρώτα να επιλέξουμε το u . Κατασκευάζουμε το γράφημα της συνάρτησης

$$\hat{e}(u) = \frac{1}{k(u)} \sum_{i: X_i > u} (X_i - u) \quad (\text{mean residual life plot})$$

ως προς u , όπου $k(u)$ είναι το πλήθος των X_i που υπερβαίνουν το u .

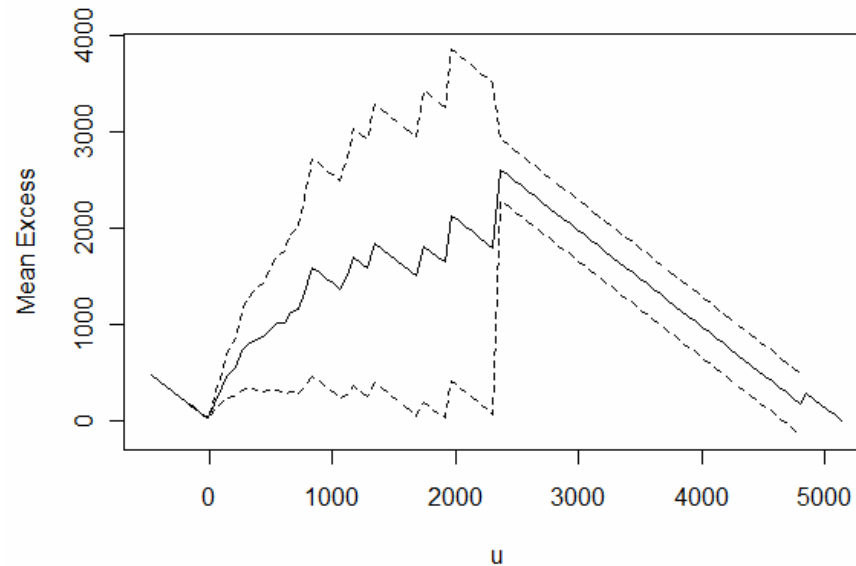
- Αρχικά κατασκευάζουμε το γράφημα αυτό για $u = X_{(n-k+1)}, \dots, X_{(n)}$ (όπου k : οι k αυτές τιμές να είναι > -500 , για να εξαιρέσουμε το αριστερό άκρο των παρατηρήσεων που δεν μας ενδιαφέρει),

```
sd<-sort(data[data>-500]); k<-length(sd); mrlp<-rep(0,k)  
for(i in 1:k){mrlp[i]<-sum(sd[i:k]-sd[i])/(k-i+1)}; plot(sd,mrlp)
```



- Επιλέγουμε ως κατώφλι το u πάνω από το οποίο το γράφημα φαίνεται να είναι περίπου γραμμικό (δεν λαμβάνουμε υπόψη τις τελευταίες δεξιά παρατηρήσεις γιατί βασίζονται σε λίγες ακραίες παρατηρήσεις και επομένως παρουσιάζουν μεγάλη διασπορά).

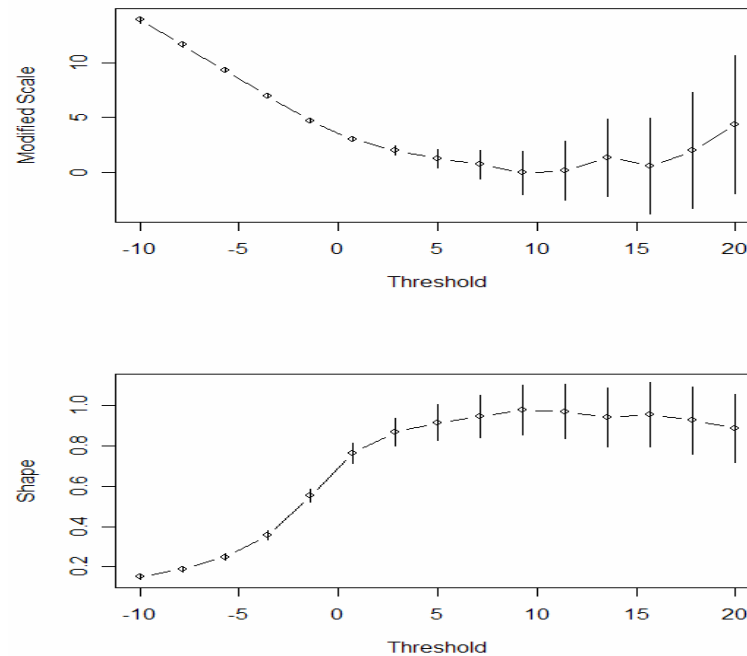
- Ένα ισοδύναμο γράφημα μπορεί να γίνει και από το πακέτο `extRemes` με την εντολή `mrl.plot(data[(data>-500)])`



- Από τα σχήματα αυτά φαίνεται ότι πρέπει να πάρουμε ένα $u > 0$ «κοντά» στο 0.

- Για να προσδιορίσουμε καλύτερα το u θα χρησιμοποιήσουμε και μια 2η μέθοδο.
- θα εκτιμήσουμε τις παραμέτρους ξ , $\tilde{\sigma} = \sigma + \xi(u - \mu)$ της GPD για διάφορες τιμές του u και κατασκευάζουμε τα αντίστοιχα γραφήματα.
- Από τα γραφήματα μπορούμε να επιλέξουμε το μικρότερο u πέρα από το οποίο το ξ σταθεροποιείται και το $\tilde{\sigma} = \sigma + \xi(u - \mu)$ μεταβάλλεται γραμμικά ως προς το u .

`gpd.fitrange(data, -10, 20, nint=15)`



- Από τα παραπάνω γραφήματα επιλέγουμε $u = 5$. Οι εκτιμήσεις (ε.μ.π.) των παραμέτρων $\tilde{\sigma}, \tilde{\xi}$ θα είναι

```

u<-5;
b<-gpd.fit(data,u,npj<-100) (npj = n per year/block)

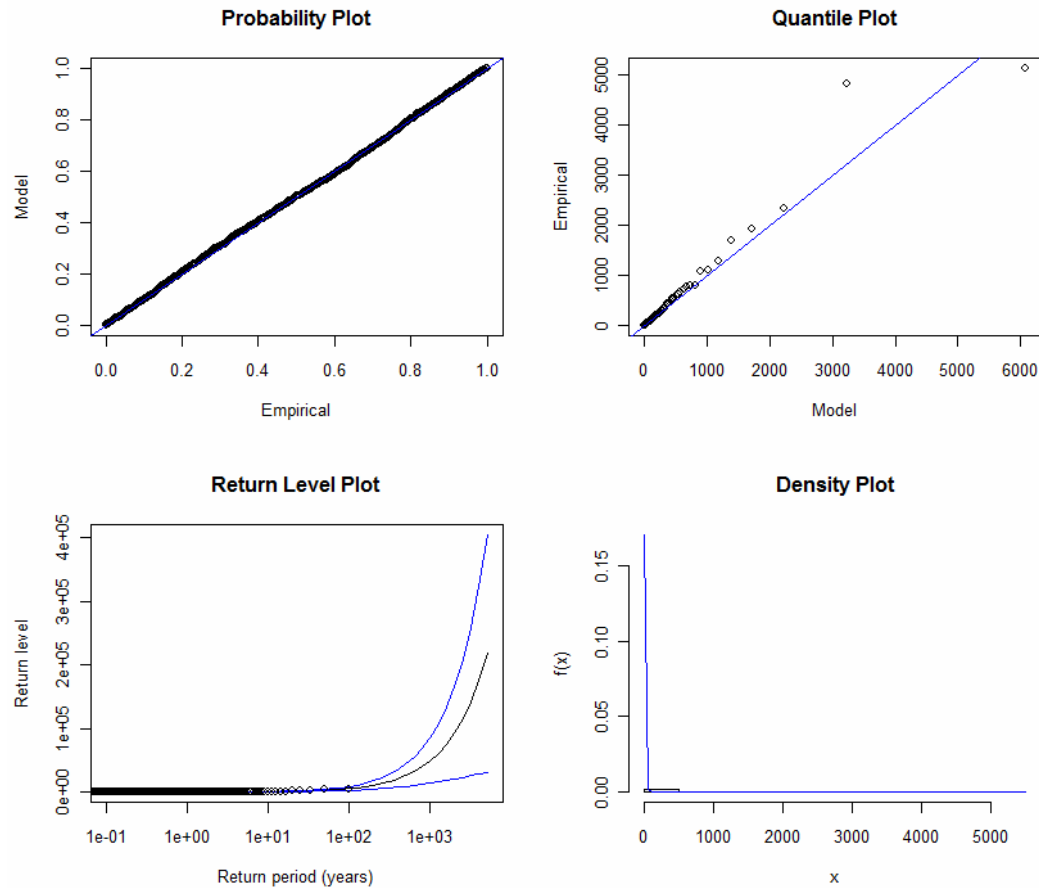
$threshold
5
$nexc
1799
$convc
0
$nlh
6626.758
$mle
5.8649878 0.9144182
$rate (The proportion of data points that lie above the threshold)
0.1799
$se
0.26731370 0.04456144

```

- Πάνω από αυτό το κατώφλι βρίσκονται $k(u) = 1799$ από τις 10000 παρατηρήσεις, ενώ οι εκτιμήσεις των παραμέτρων είναι 5.8649878 και 0.9144182.
- Μπορούμε να κατασκευάσουμε και πάλι τα αντίστοιχα (προσεγγιστικά) δ.ε. συντελεστού $1 - \alpha$, $(\hat{\sigma} \pm \sqrt{s_{2,2}} z_{\alpha/2}), (\hat{\xi} \pm \sqrt{s_{3,3}} z_{\alpha/2})$.

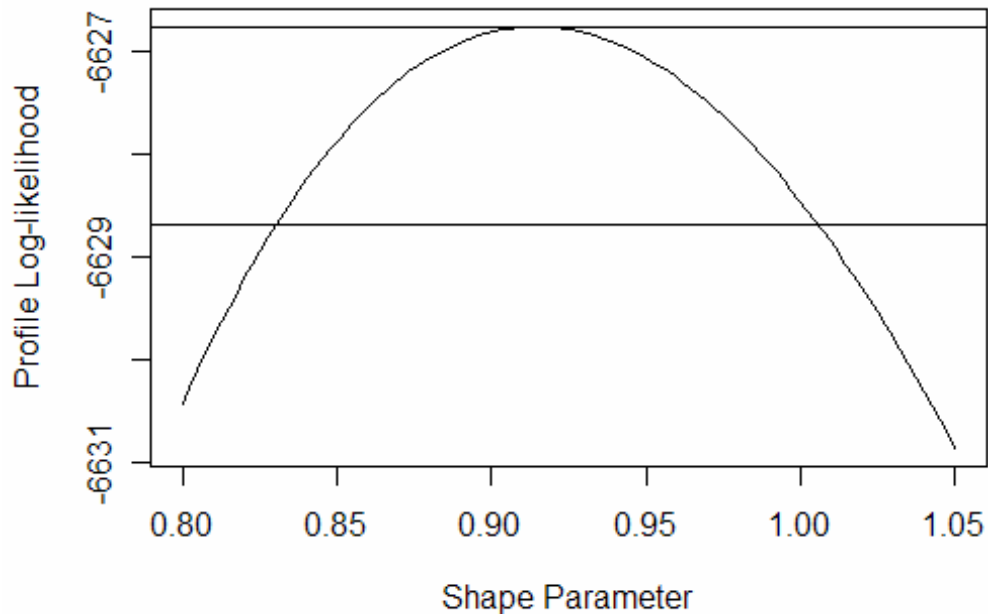
- Κατασκευάζουμε τα *διαγνωστικά γραφήματα* για να εξετάσουμε την καλή προσαρμογή των POT στην κατανομή GPD. Ακολουθεί το γράφημα της στάθμης απόδοσης και το ιστόγραμμα των POT μαζί με την σ.π.π. της GPD

gpd.diag(b)



- Επίσης, ένα δ.ε. συντελεστού $1 - \alpha = 95\%$ για το ξ μέσω της profile Deviance Function δίνεται από το ακόλουθο γράφημα

`gpd.profxi(b, 0.8, 1.05)`



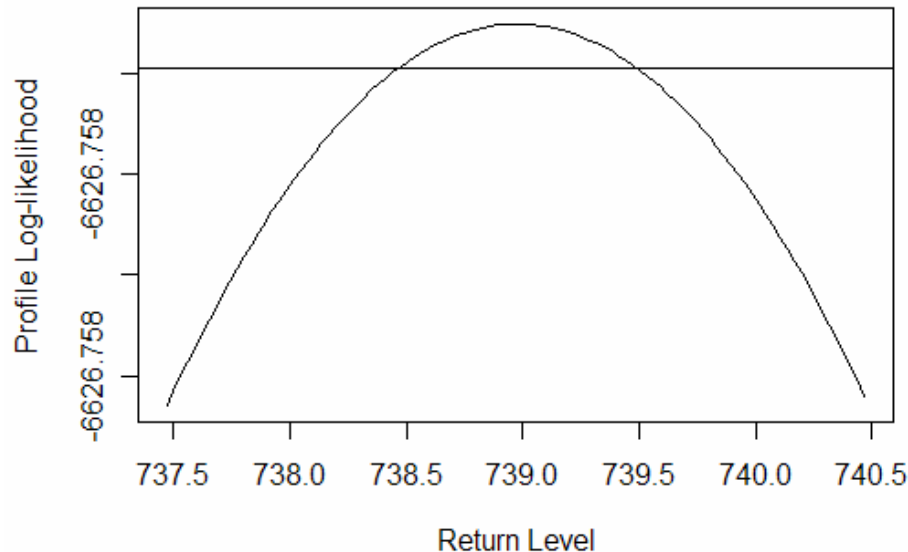
- Τέλος μπορούμε να εκτιμήσουμε και τη στάθμη απόδοσης για m παρατηρήσεις:

$$\hat{x}_m \approx u + \frac{\hat{\sigma}}{\hat{\xi}} \left(\left(\frac{mk(u)}{n} \right)^{\hat{\xi}} - 1 \right)$$

```
sigma2<-b$mle[1];xi2<-b$mle[2];
k<-b$nex
m<-100*10;
xm<-u+sigma2/xi2*((m*k/n)^xi2-1)
xm
738.4703
```

- Επίσης μπορούμε να εκτιμήσουμε την στάθμη απόδοσης για m παρατηρήσεις από την profile log-likelihood.
- Στην περίπτωση της μεθόδου block maxima είχαμε κάνει το ίδιο για $1/p = 10$ περιόδους (blocks). Ζητάμε και εδώ για λόγους σύγκρισης τη στάθμη απόδοσης για 10 χρονικές περιόδους,

```
gpd.prof(b, 10, xm-1, xm+2, npx<-100)
```



- Επίσης μπορούμε να κατασκευάσουμε μόνοι μας και το γράφημα της στάθμης απόδοσης (εκτίμηση του x_m για διάφορες τιμές του m) και να το συγκρίνουμε με το αντίστοιχο γράφημα που προέκυψε από την μέθοδο block maxima

```
m<-100*1:50
xm<-u+sigma2/xi2*((m*k/n)^xi2-1)
plot(m/100,xm,type="o")
```

```
p<-1/1:50;w=-log(-log(1-p));
zp<-mu-sigma/xi*(1-exp(xi*w));
points(1/p,zp,type="o")
```

