Programming in R

Anja Bråthen Kristoffersen

Biomedical Research Group

UNIVERSITY OF OSLO

Function

```
panel.hist <- function(x, ...){
  usrx <- par("usr")
  on.exit(par(usrx))
  par(usr = c(usrx[1:2], 0, 1.5) )  # indicates the position in the plot
  hi <- hist(x, plot = FALSE)      # calculate histogram without plotting it
  Breaks <- hi$breaks              # define breaks used in h
  nB <- length(Breaks)             # count the number of breaks
  y <- hi$counts                   # find the counts in each interval
  y <- y/max(y)                    # scale y for plotting
  rect(Breaks[-nB], 0, Breaks[-1], y)  #plots rectangulars in existing plot
}
pairs(dat[,7:11], col = dat[,5], cex = 0.5, pch = 24, diag.panel = panel.hist,
upper.panel=NULL)
```

R Console

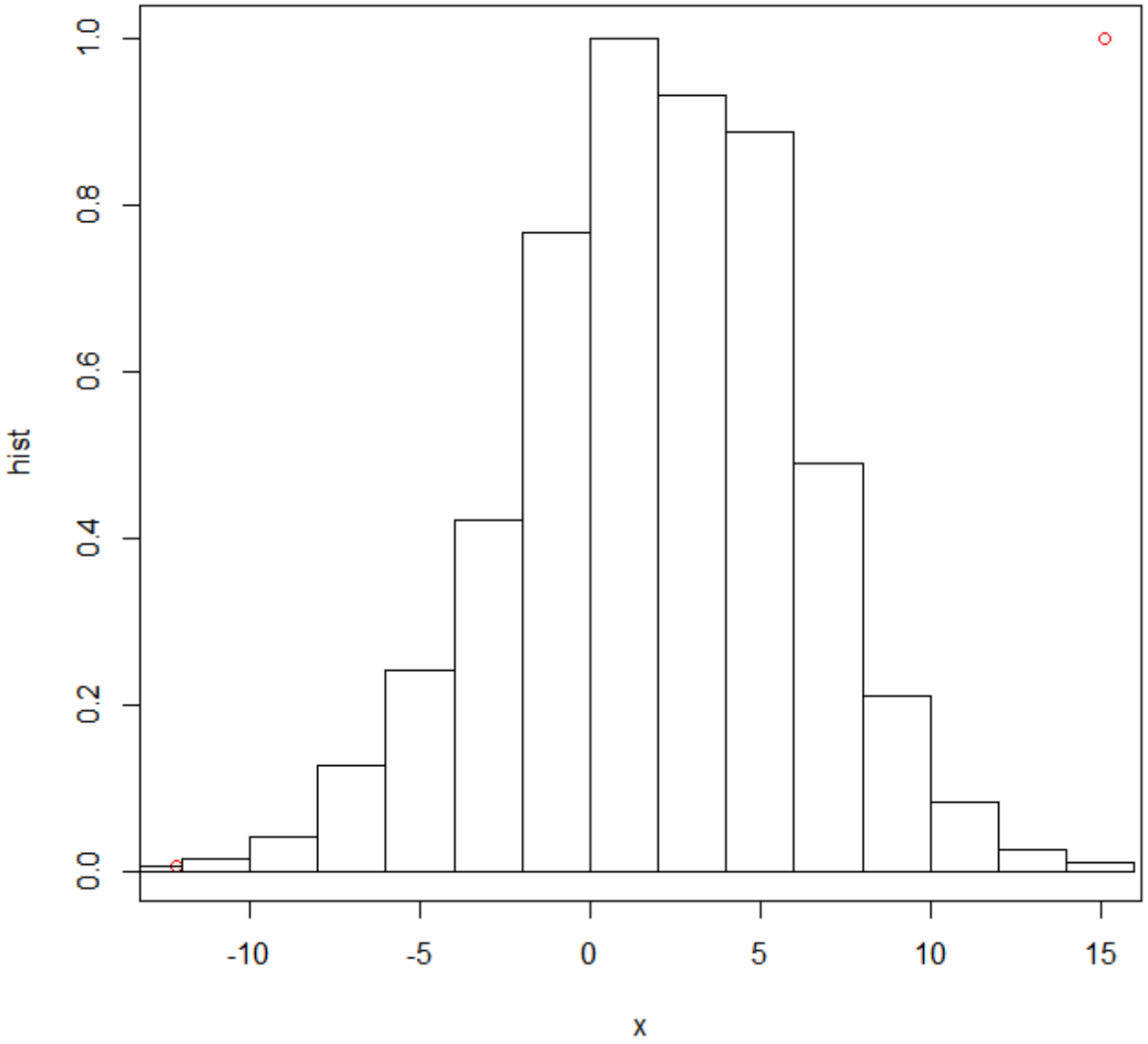
```
> x <- rnorm(1000,2,4)
> hi <- hist(x, plot = FALSE)      # beregner histogram men plotter ikke
> hi
$breaks
 [1] -14 -12 -10  -8  -6  -4  -2   0   2   4   6   8  10  12  14  16

$counts
 [1]   1   3   8  24  46  80 146 190 177 169  93  40  16   5   2

$intensities
 [1] 0.0005 0.0015 0.0040 0.0120 0.0230 0.0400 0.0730 0.0950 0.0885 0.0845
[11] 0.0465 0.0200 0.0080 0.0025 0.0010

$density
 [1] 0.0005 0.0015 0.0040 0.0120 0.0230 0.0400 0.0730 0.0950 0.0885 0.0845
[11] 0.0465 0.0200 0.0080 0.0025 0.0010

$mids
 [1] -13 -11  -9  -7  -5  -3  -1   1   3   5   7   9  11  13  15

$xname
[1] "x"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
>
```

RGui (32-bit)

File Edit View Misc Packages Windows Help

R Console

```
> Breaks <- hi$breaks                 # definerer breaks brukt i h
> Breaks
 [1] -14 -12 -10  -8  -6  -4  -2   0   2   4   6   8  10  12  14  16
> nB <- length(Breaks)                # finner antall breaks
> nB
[1] 16
> y <- hi$counts
> y                           # finner antall innen hvert intervall
 [1]   1   3   8  24  46  80 146 190 177 169  93  40  16   5   2
> y <- y/max(y)       # skalerer y
> y
 [1] 0.005263158 0.015789474 0.042105263 0.126315789 0.242105263 0.421052632
 [7] 0.768421053 1.000000000 0.931578947 0.889473684 0.489473684 0.210526316
[13] 0.084210526 0.026315789 0.010526316
> plot(c(min(x), max(x)), c(min(y), max(y)), col = "red", xlab = "x", ylab = "h$
> rect(Breaks[-nB], 0, Breaks[-1], y)  #plotter rektangler i eksisterende plot
>
```

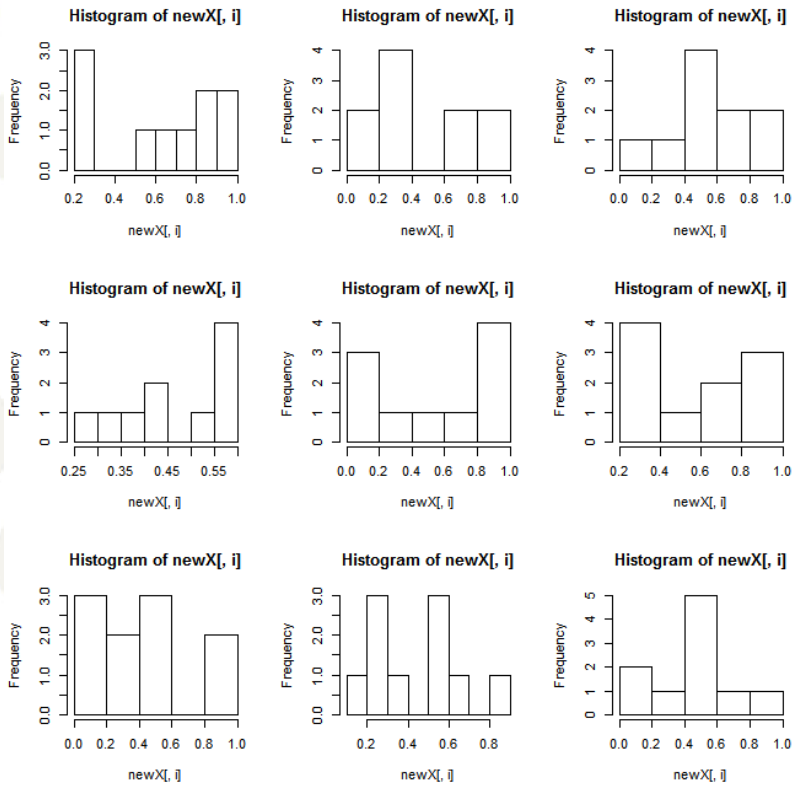


apply()

- Use apply when you want the same thing done on every row or column of a matrix.
- apply(d, 1, functionA) will use functionA on every row of dataset d.
- apply(d, 2, functionA) will use functionA on every column of dataset d.

Example: apply()

```
d <- matrix(runif(90), ncol = 10, nrow = 9)
head(d)
par(mfrow = c(3,3))
apply(d, 1, hist)
```



for – loop

- for (*name* in *expr_1*) *expr_2*
  - name on variable: i
  - expr_1 should be 1:antall
  - expr_2 should be

    sum(dbinom(1:6, 8, p[i])), but you have to save it somewhere:

    beta8[i] <- sum(dbinom(1:6, 8, p[i])) then you have to create beta8 before the loop.

for – loop

```
p <- seq(0.6, 1, 0.01)
antall <- length(p)
beta8 <- rep(NA, antall) #allocate space for beta8
for(i in 1:antall){
  beta8[i] <- sum(dbinom(1:6, 8, p[i]))
}
power8 <- 1 - beta8
plot(p, power8, type = "l")
```

R Console

```
> p <- seq(0.6, 1, 0.01)
> p
 [1] 0.60 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.70 0.71 0.72 0.73 0.74
[16] 0.75 0.76 0.77 0.78 0.79 0.80 0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89
[31] 0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99 1.00
> antall <- length(p)
> antall
[1] 41
> beta8 <- rep(NA, antall) #allocate space for beta8
> beta8
 [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
[26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
> for(i in 1:antall){
+   beta8[i] <- sum(dbinom(1:6, 8, p[i]))
+ }
> beta8
 [1] 0.892968960 0.882240491 0.870674126 0.858239347 0.844906652 0.830647950
 [7] 0.815436989 0.799249805 0.782065211 0.763865313 0.744636060 0.724367835
[13] 0.703056075 0.680701940 0.657313014 0.632904053 0.607497778 0.581125712
[19] 0.553829063 0.525659666 0.496680960 0.466969037 0.436613735 0.405719786
[25] 0.374408036 0.342816713 0.311102764 0.279443260 0.248036860 0.217105351
[31] 0.186895260 0.157679534 0.129759300 0.103465702 0.079161816 0.057244650
[37] 0.038147228 0.022340758 0.010336892 0.002690078 0.000000000
> power8 <- 1 - beta8
> plot(p, power8, type = "l")
>
```

if statment

- if (*expr_1*) *expr_2* else *expr_3*
- *expr_1* is a statement that is either true or false
- expr_2 is preformed if expr_1 is true
- expr_3 is optional, but preformed if expr_1 is false and else statement is included

Example: if()

```
x <- runif(1,0,1)
y <- rnorm(1,0,1)
if(x > y){
  print(paste("x =", round(x,3), "is greater
than y =", round(y,3), sep = " "))
} else {
  print(paste("x =", round(x,3), "is less than
y =", round(y,3), sep = " "))
}
```

```
[1] "x = 0.783 is greater than y 0.536"
```

When finished

- You could save your workspace in R, you could then save the workspace in different project folders. I do not do that!
- You could just save your script and run it once again if you want to work further on it.
- You could write your partial results (matrix) that you want to work further on to a .txt file, use write.table(objectname, ”path”, sep = ”\t”)
  - objectname is the name of your object in R
  - ”path” is your path to where you will save the object including the file name of the object.
  - sep = ”\t” ensure that your txt file is tabulate separated and makes it easier to open in excel.