07112023_Statistical_Learning

Mattia G.

2023-11-07

R Markdown

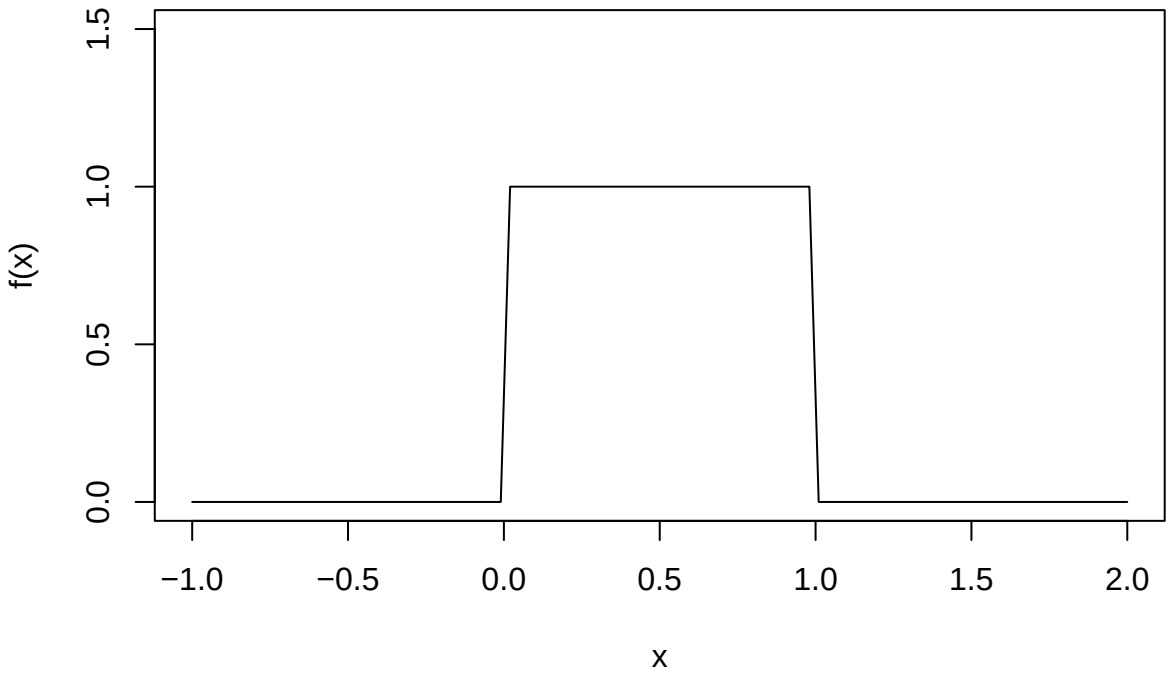
for a better experience I suggest to input the following code chunks into RStudio.

R Markdown

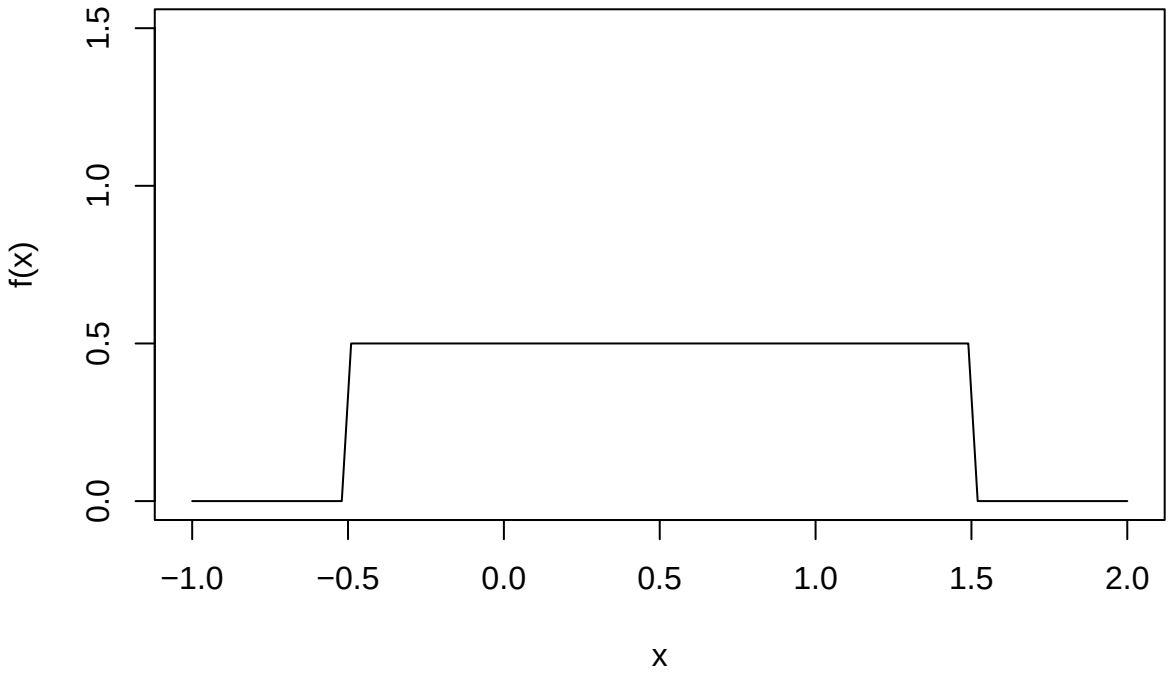
```
#######################################
#    PROBABILITY DISTRIBUTIONS
# -- continuous Random Variables --
#######################################

# uniform distribution
########################


# density function
dunif(0.2, 0, 1)
```

```
## [1] 1
```

```
# plot the density function
curve(dunif, -1, 2, ylab="f(x)", ylim=c(0,1.5))
```



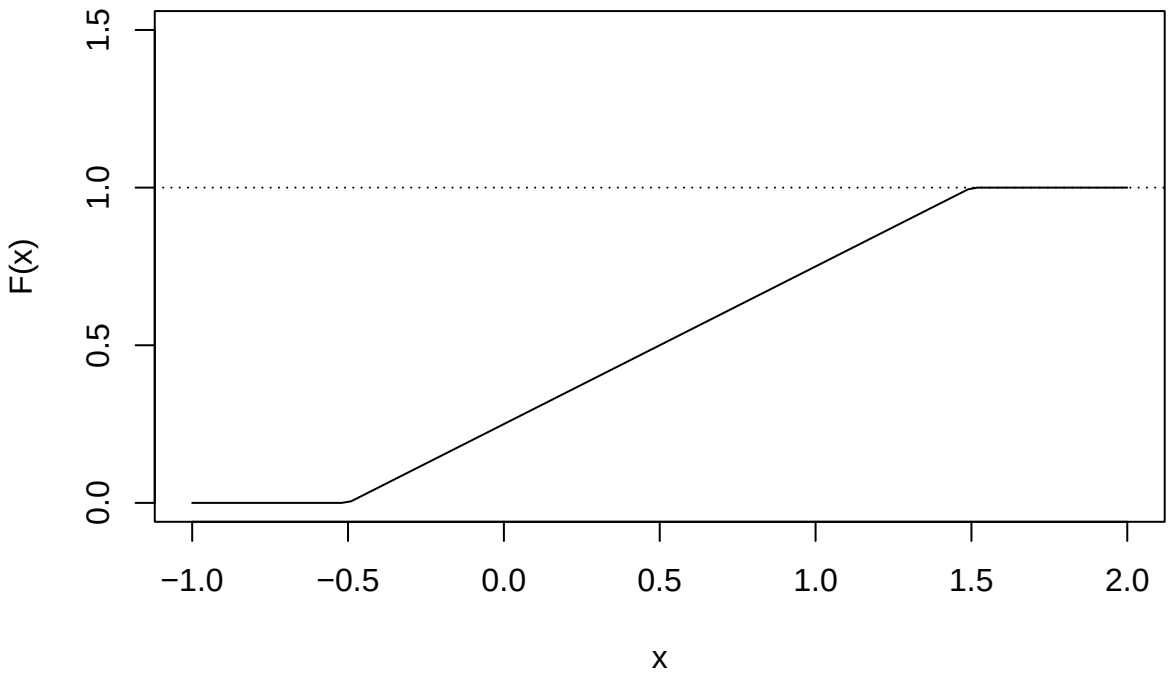
```r
curve(dunif(x, min=-0.5, max=1.5), -1, 2, ylab="f(x)", ylim=c(0,1.5))
```




```r
# cumulative distribution function
punif(0.2, 0, 1)
```

```
## [1] 0.2
```

```r
# plot the cumulative distribution function
curve(punif(x, min=-0.5, max=1.5),-1, 2, ylab="F(x)", ylim=c(0,1.5))
abline(h=1, lty=3)
```



```r
# uniformly distributed random values
runif(3, -0.5, 1.5)
```

```
## [1] 1.0372747 0.2916805 0.9738783
```

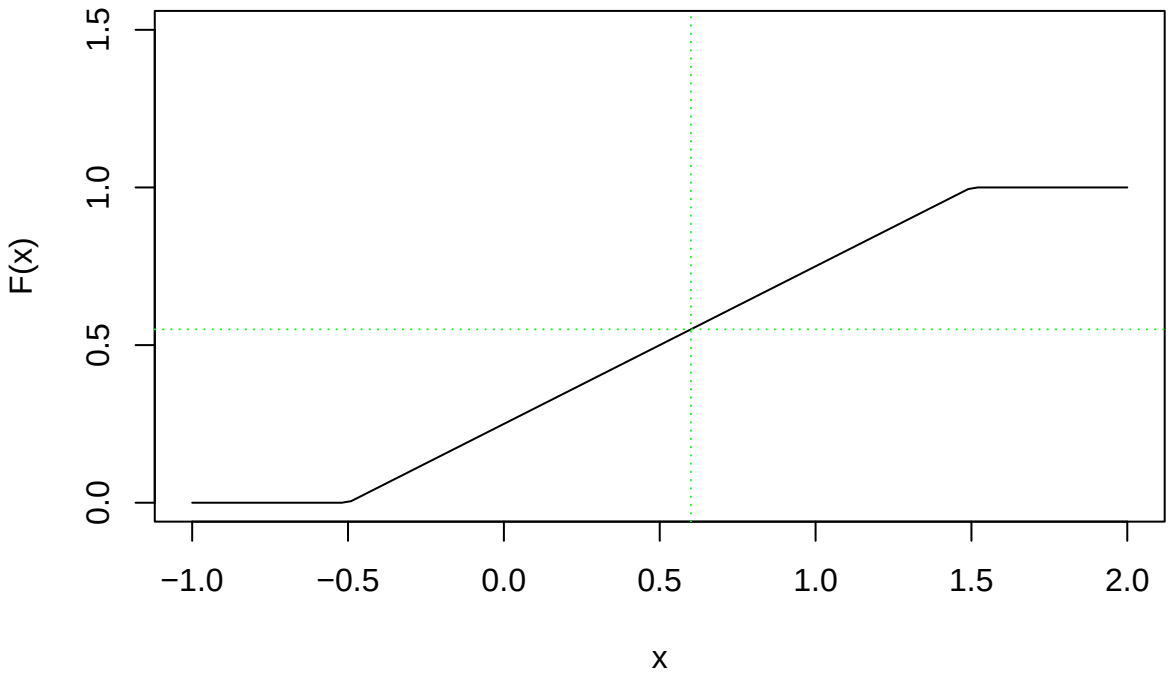
```r
# quantiles of the uniform distribution
qunif(.3, -0.5, 1.5)
```

```
## [1] 0.1
```

```r
# how to see quantiles on the cumulative distribution function

pr <- 0.55
qunif(pr, -0.5, 1.5)
```

```
## [1] 0.6
```

```r
curve(punif(x, min=-0.5, max=1.5), -1, 2, ylab="F(x)", ylim=c(0,1.5))
abline(h=pr, v=qunif(pr, -0.5, 1.5), lty=3, col="green")
```




```r
# exponential distribution
############################

f <- function(x, lambda=1) lambda*exp(-lambda*x)

f(0.5)
```

```
## [1] 0.6065307
```

```r
f(0.5, lambda=1/3)
```

```
## [1] 0.2821606
```

```r
# or equivalently

dexp(0.5)
```

```
## [1] 0.6065307
```

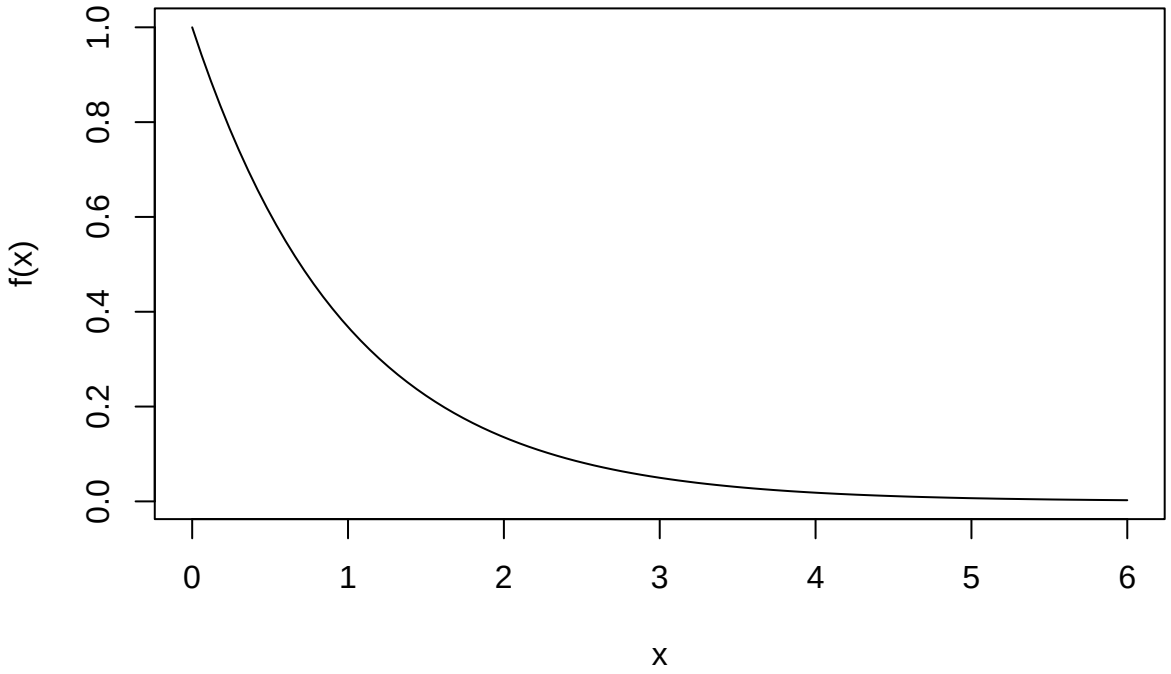
```
dexp(0.5, rate=1/3)
```

```
## [1] 0.2821606
```

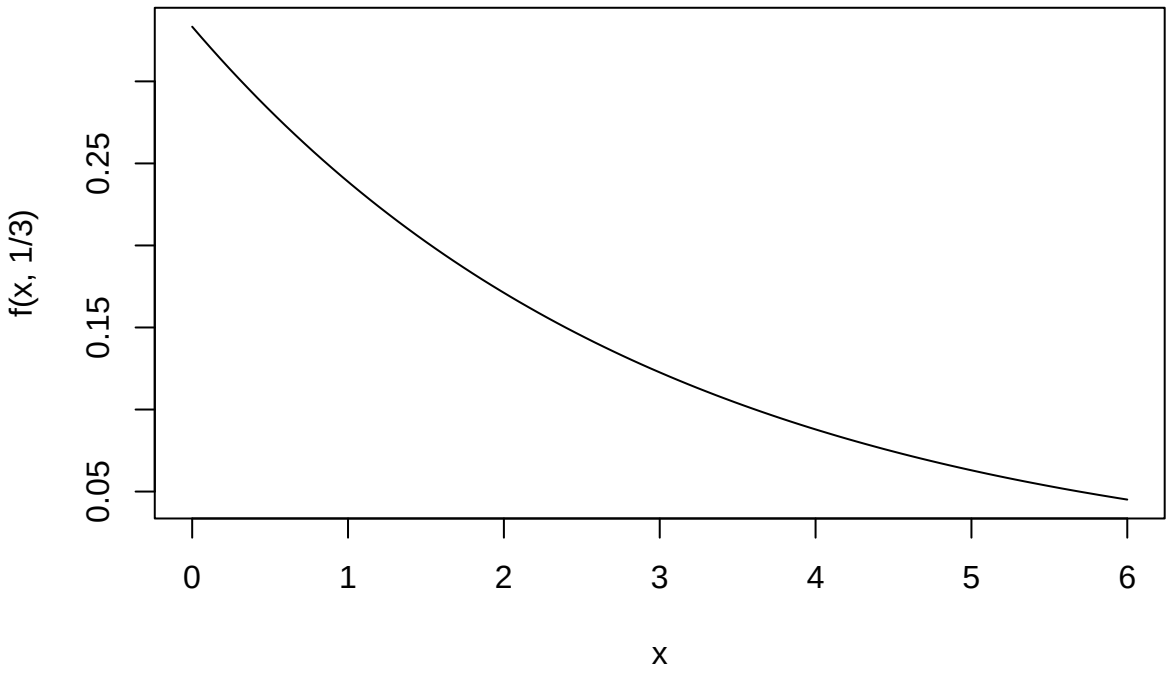
```
# numerical integration
# check the area under the curve is equal to 1

integrate(f, 0, Inf)
```

```
## 1 with absolute error < 5.7e-05
```

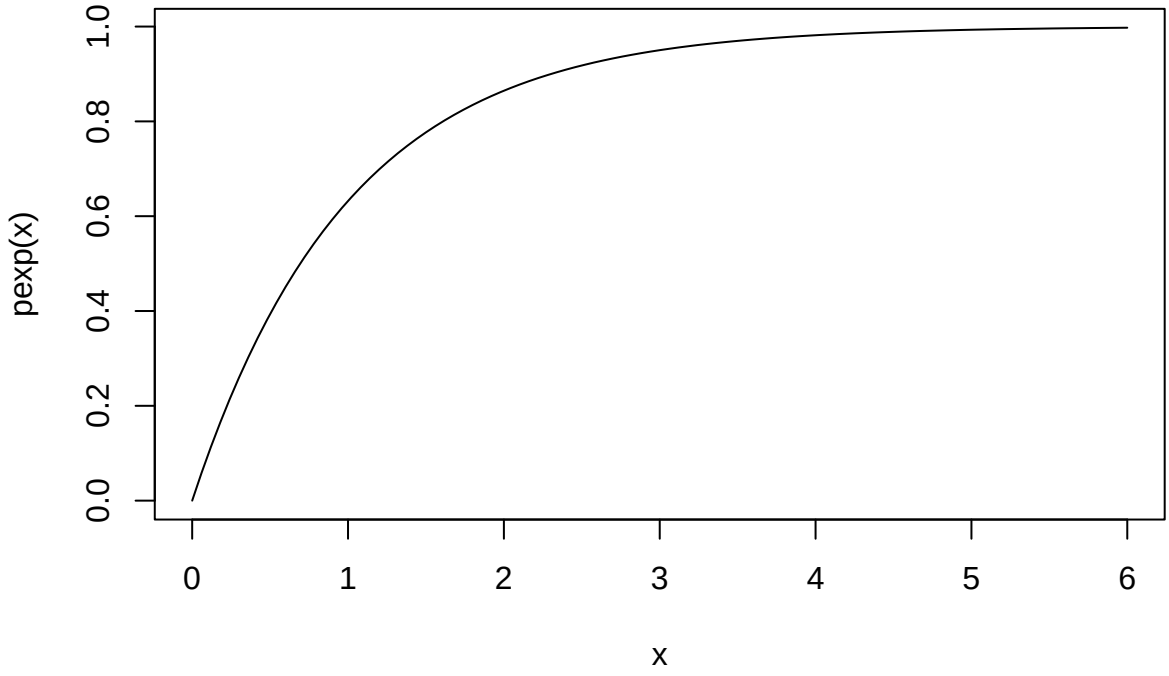
```
# plot the pdf

curve(f, from=0, to=6)
```



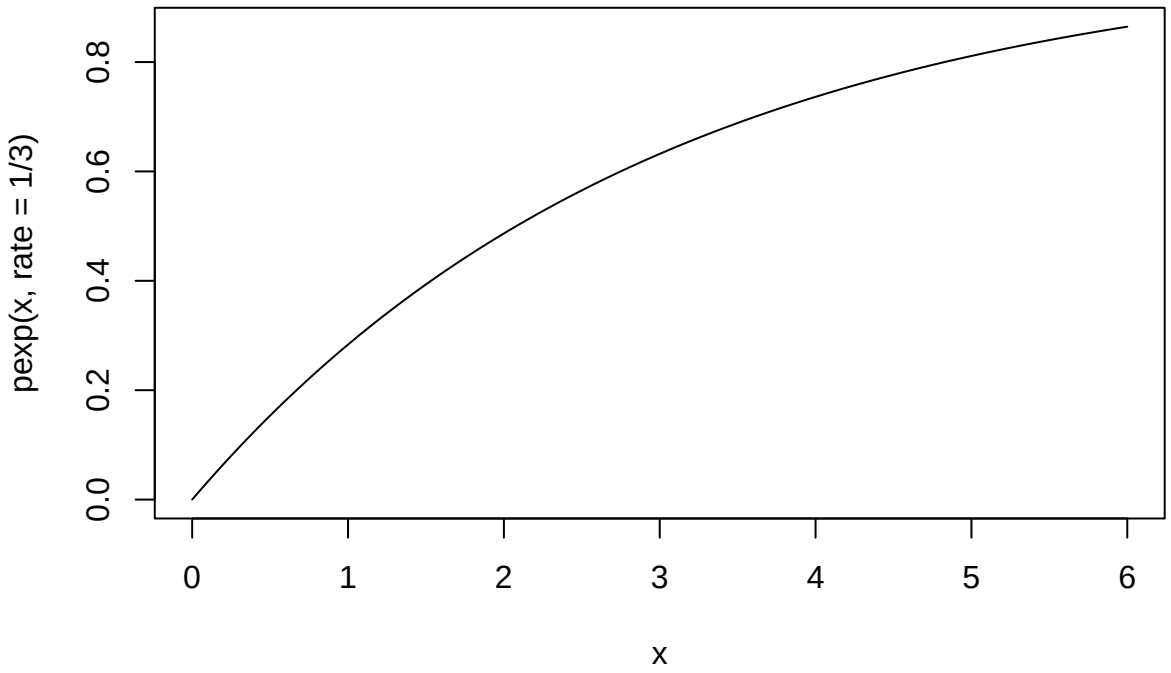

```
curve(f(x, 1/3), 0, 6)
```




```
# plot the cdf

curve(pexp, 0, 6)
```



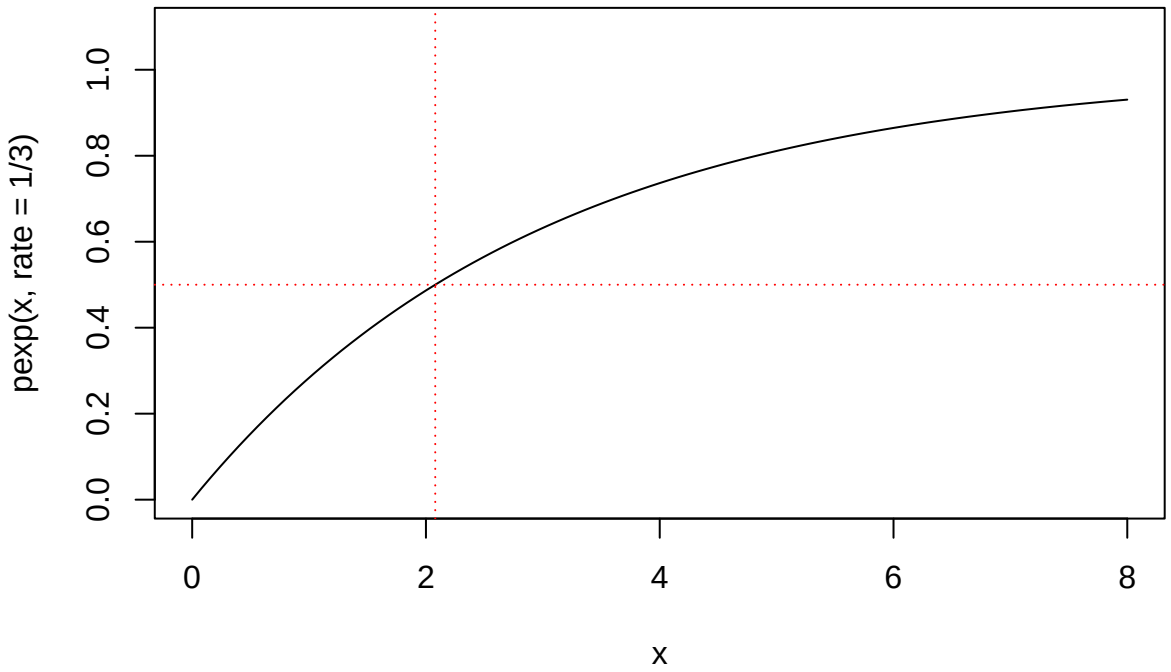

```
curve(pexp(x, rate=1/3), 0, 6)
```




```
# quantiles of the exponential distribution


pr <- 0.5
qexp(pr, rate=1/3)
```

```
## [1] 2.079442
```

```
curve(pexp(x, rate=1/3), 0, 8, ylim=c(0, 1.1))
abline(h=pr, v=qexp(pr, rate=1/3), col="red",lty=3)
```



```r
# E(X)

f <- function(y) y*exp(-y)
integrate(f, 0, Inf)
```

```
## 1 with absolute error < 6.4e-06
```

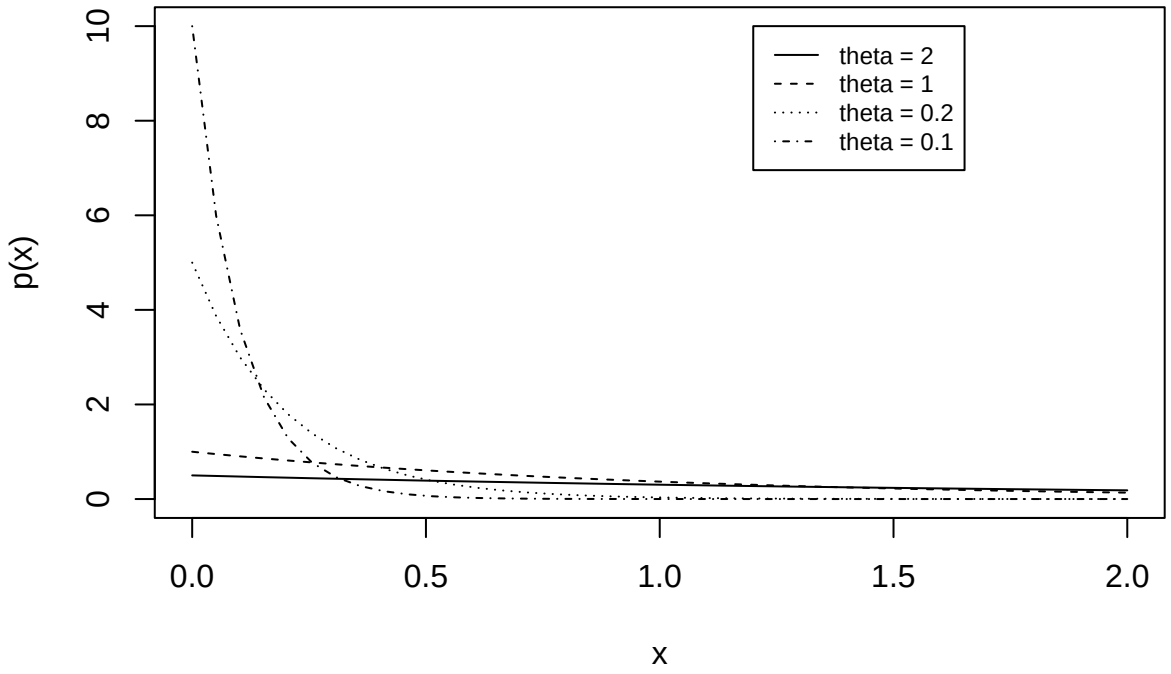
```r
# E(Y^2)

f <- function(y) y^2*exp(-y)
integrate(f, 0, Inf)
```

```
## 2 with absolute error < 7.1e-05
```

```r
# variance

f <- function(y) (y-1)^2*exp(-y)
integrate(f, 0, Inf)
```

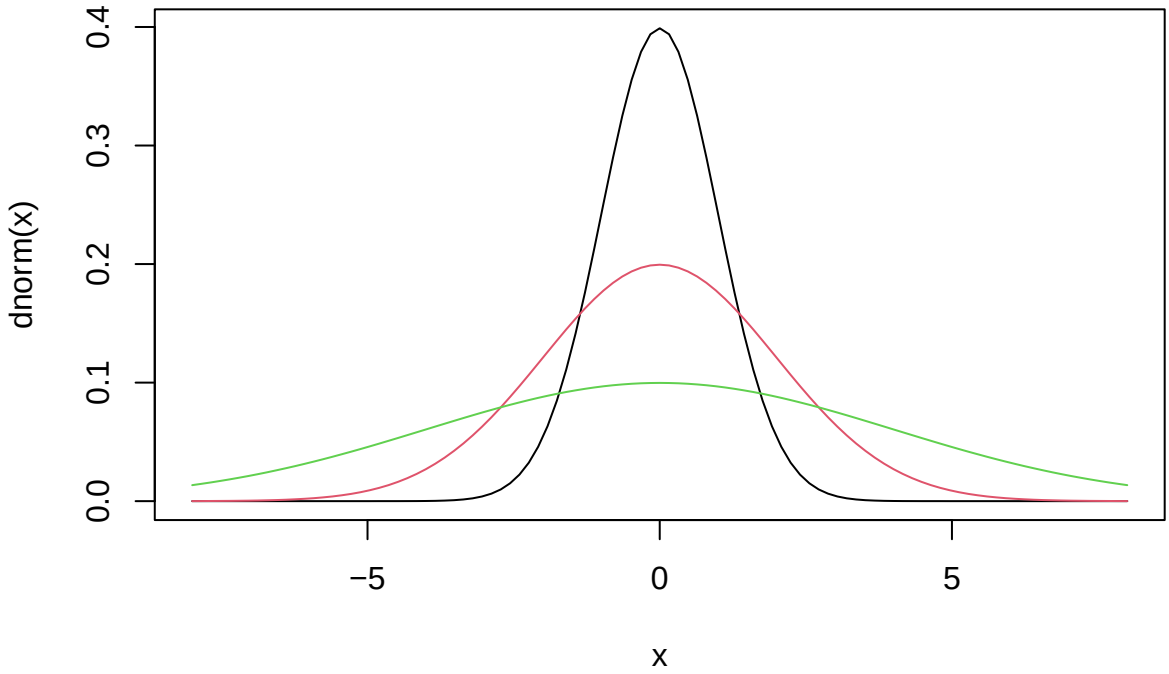
```
## 1 with absolute error < 5.8e-05
```

```r
# comparison of exponential densities

x <- seq(0, 2, length=40)
theta <- c(2, 1, .2, .1) # mean of distribution
y <- matrix(NA, 40, 4)
for (i in 1:4) {
  y[,i] <- dexp(x, 1/theta[i]) # parameter is the rate
}
matplot(x, y, type="l", xlab="x", ylab="p(x)", lty=1:4, col=1)
legend(1.2, 10, paste("theta =", theta), lty=1:4, cex=.75)
```

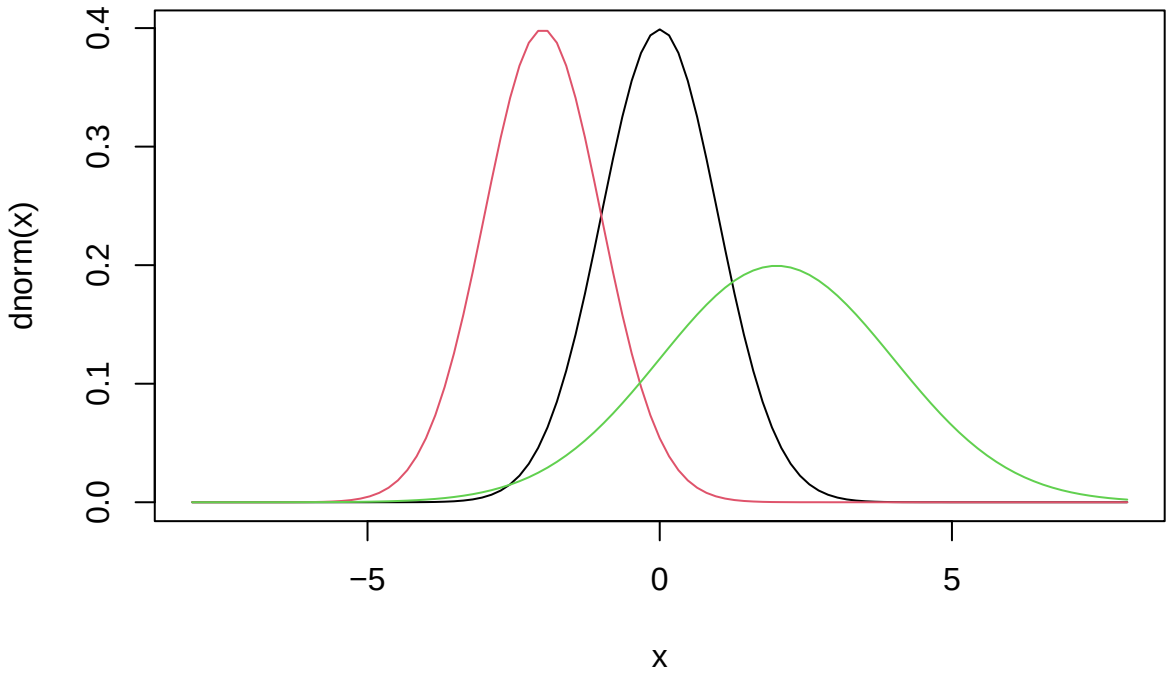



```r
###################################
# normal distribution
###################################
```

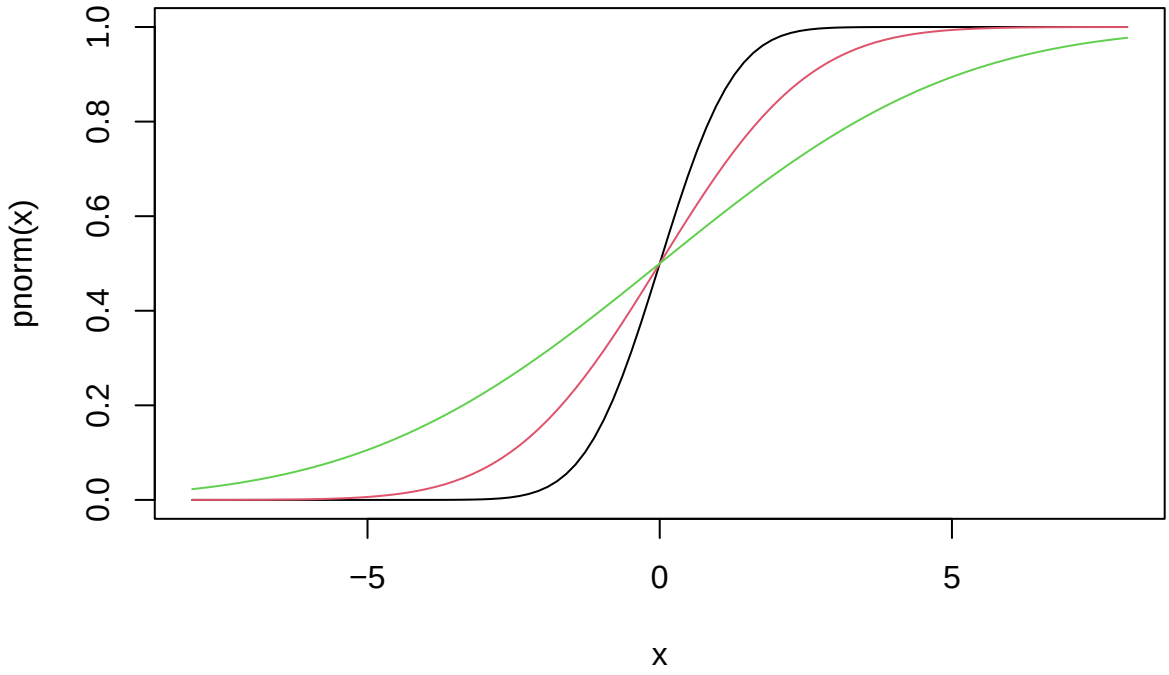
```r
# distribution for different values of mu and sigma

# some pdf's

curve(dnorm,-8,8)
curve(dnorm(x, mean=0, sd=2),add=TRUE, col=2)
curve(dnorm(x, mean=0, sd=4),add=TRUE, col=3)
```




```r
curve(dnorm, -8, 8)
curve(dnorm(x, -2, 1), add=TRUE, col=2)
curve(dnorm(x,  2, 2), add=TRUE, col=3)
```




```
# and corresponding cdf's

curve(pnorm,-8,8)
curve(pnorm(x,0,2),add=TRUE, col=2)
curve(pnorm(x,0,4),add=TRUE, col=3)
```




```
# quantiles and pdf

par(mfrow=c(1,2))
q.15 <- qnorm(0.15)
curve(dnorm,-3,3)
```

```
abline(v=q.15, lty=2, col="blue")
pnorm(q.15)
```

```
## [1] 0.15
```

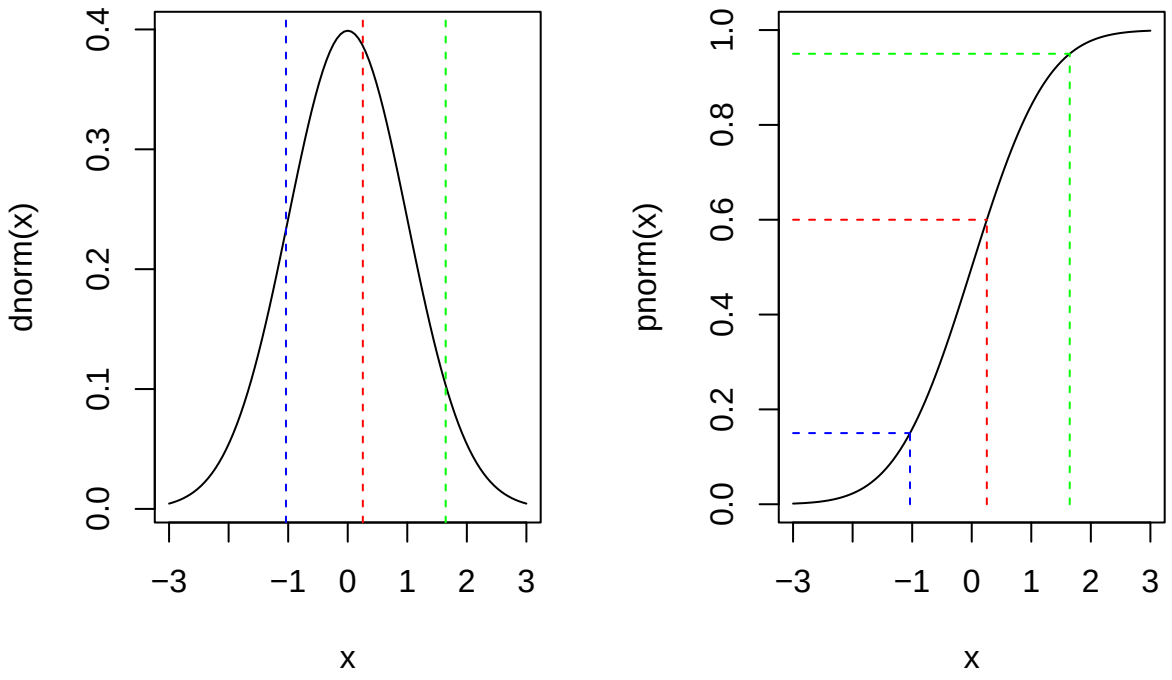
```
q.60 <- qnorm(0.60)
abline(v=q.60, lty=2, col="red")
pnorm(q.60)
```

```
## [1] 0.6
```

```
q.95 <- qnorm(0.95)
abline(v=q.95, lty=2, col="green")
pnorm(q.95)
```

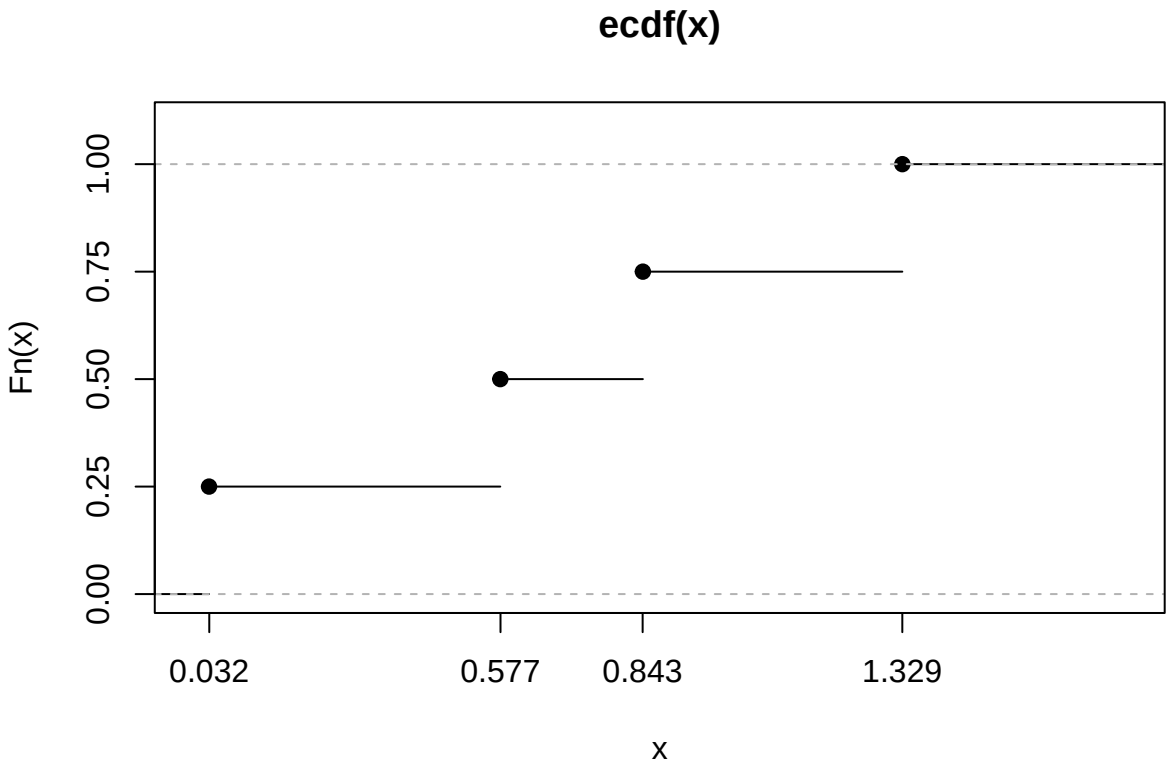
```
## [1] 0.95
```

```
# quantiles and cdf

curve(pnorm,-3,3)
lines(x=c(-3, q.15), y=c(0.15, 0.15), lty=2, col="blue")
lines(x=c(q.15, q.15), y=c(0, 0.15), lty=2, col="blue")

lines(x=c(-3, q.60), y=c(0.60, 0.60), lty=2, col="red")
lines(x=c(q.60, q.60), y=c(0, 0.60), lty=2, col="red")

lines(x=c(-3, q.95), y=c(0.95, 0.95), lty=2, col="green")
lines(x=c(q.95, q.95), y=c(0, 0.95), lty=2, col="green")
```

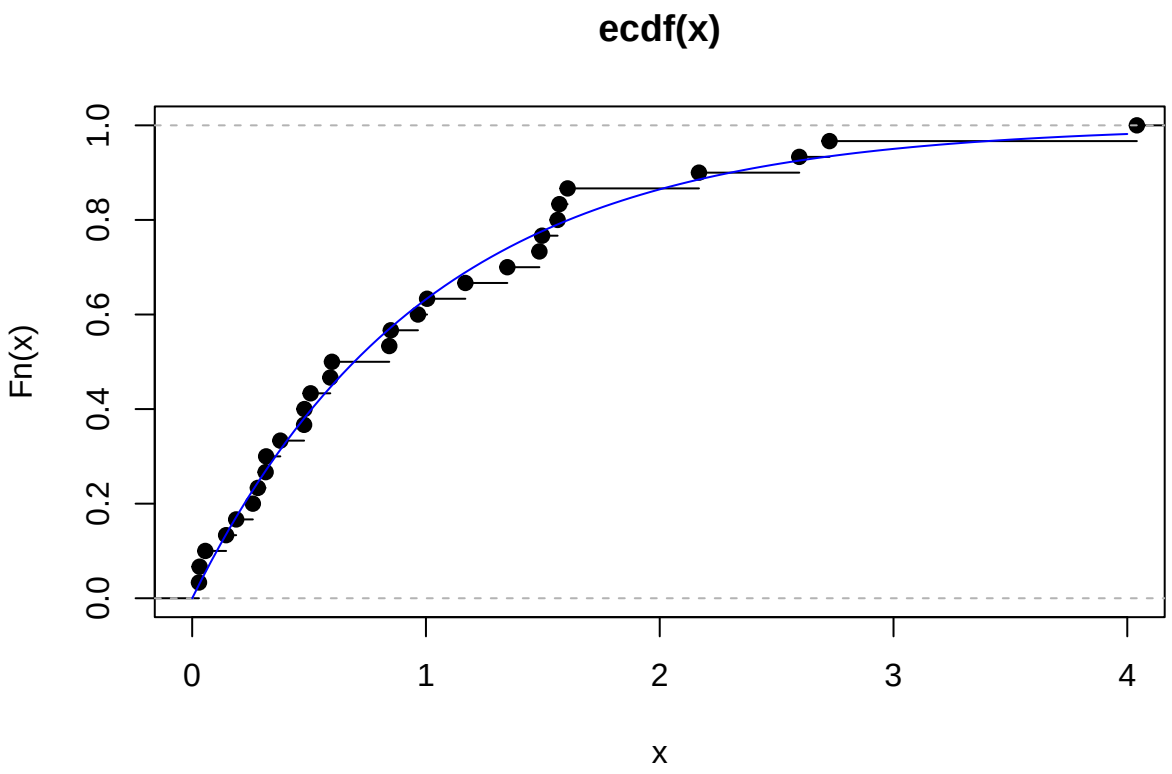



```
par(mfrow=c(1,1))

# empirical pdf  (function ecdf())

set.seed(123)
```

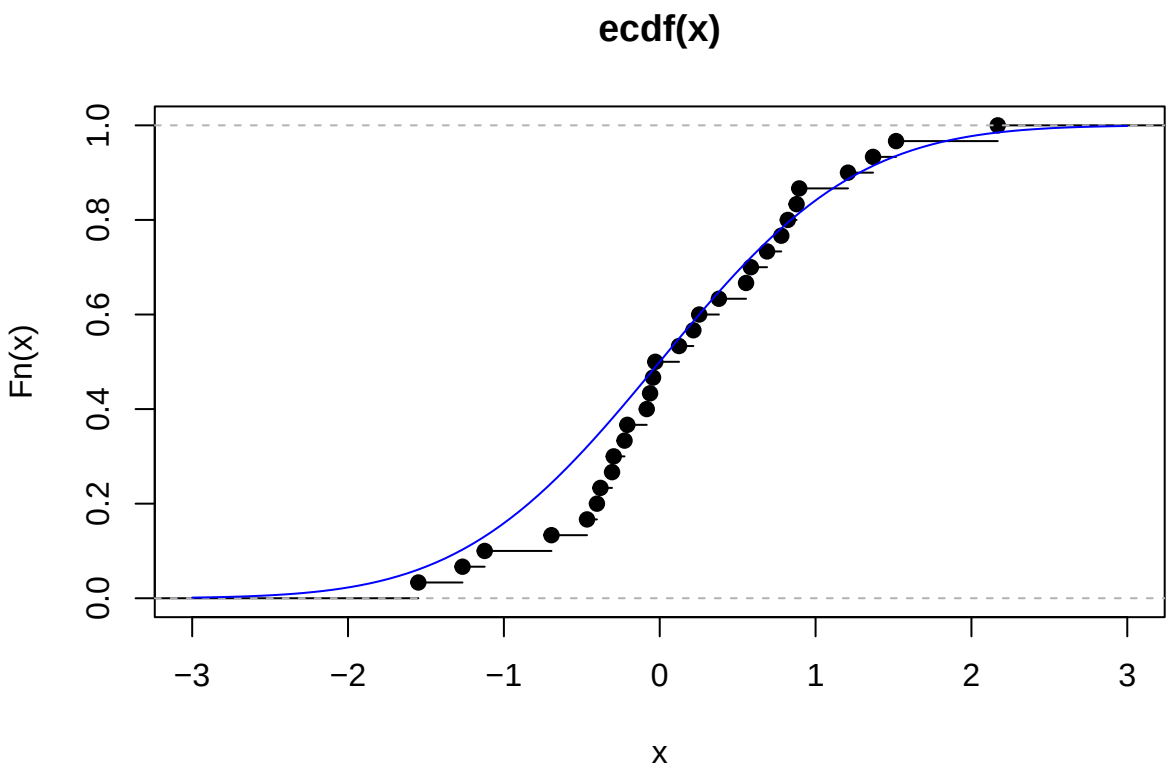
```r
x <- rexp(4)
plot(ecdf(x), xlim=c(0,1.75), ylim=c(0, 1.1), axes=FALSE)
box()
axis(2, at=c(0, 0.25, 0.5, 0.75, 1))
axis(1, at=round(x, 3))
```



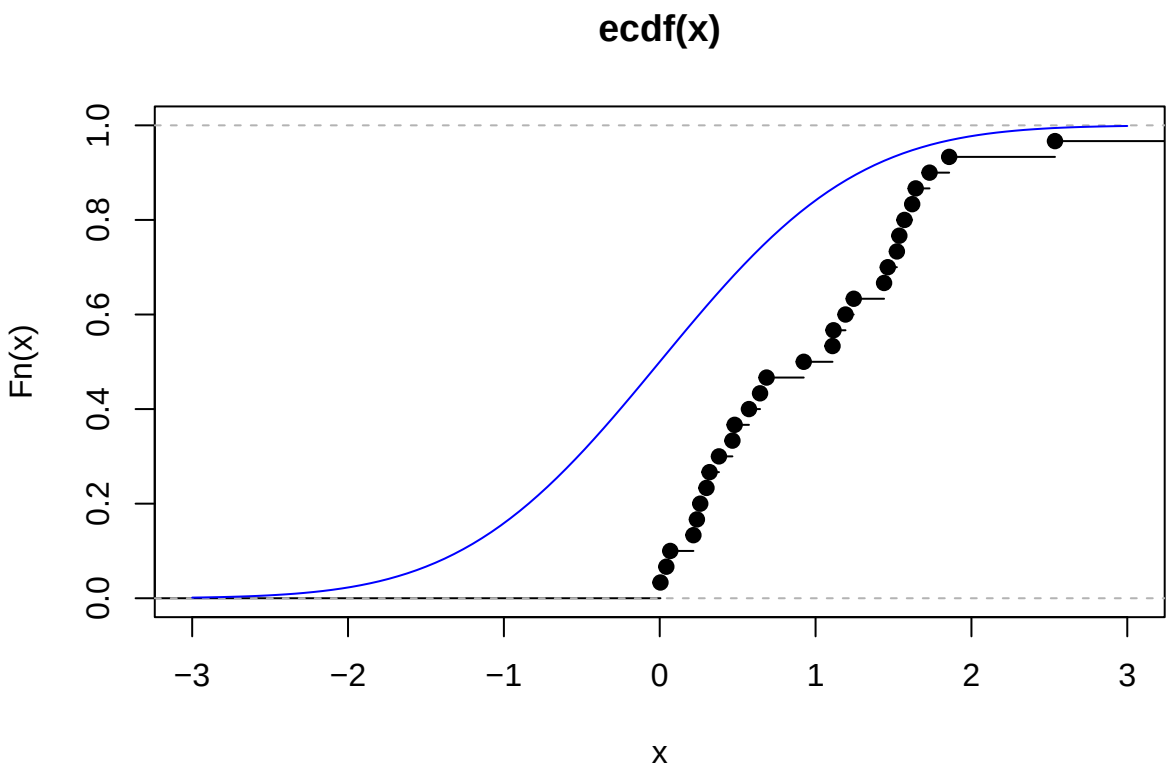

```r
# exponential distribution
# comparison between epdf and "exact" pdf

x <- rexp(30)
plot(ecdf(x), xlim=c(0,4))
curve(pexp, add=TRUE, col="blue")
```



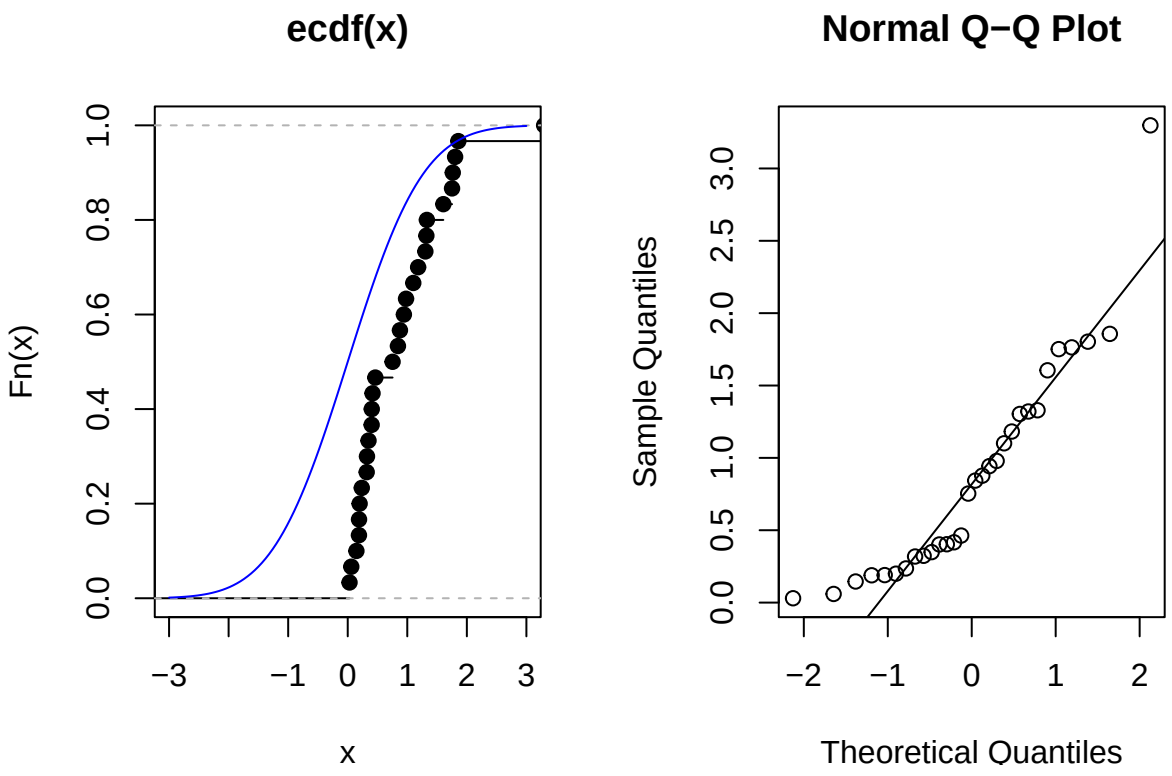

```
# normal distribution
# comparison between epdf and "exact" pdf

x <- rnorm(30)
plot(ecdf(x), xlim=c(-3,3))
curve(pnorm, add=TRUE, col="blue")
```



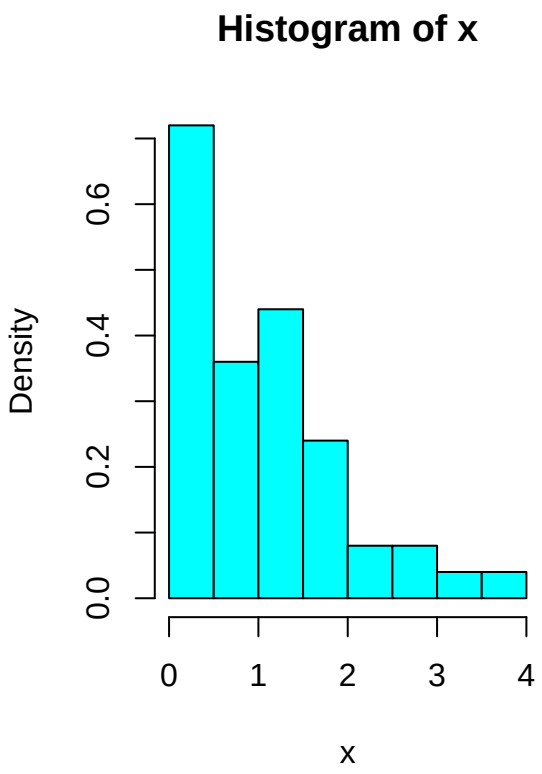

```
# comparison between ecdf from exponential
# and pdf from standard normal

x <- rexp(30)
plot(ecdf(x), xlim=c(-3,3))
curve(pnorm, add=TRUE, col="blue")
```

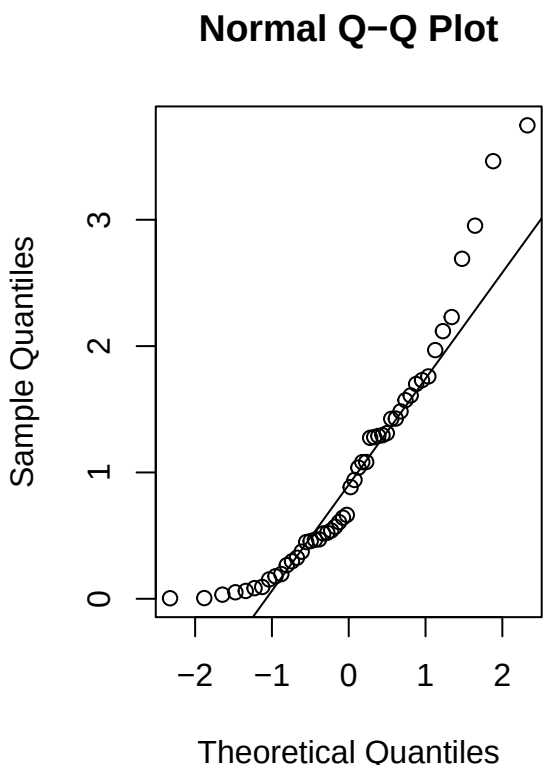



```
##############################
# Normal QQPLOT
##############################

# empirical cdf

par(mfrow=c(1,2))

x <- rexp(30)
plot(ecdf(x), xlim=c(-3,3))
curve(pnorm, add=TRUE, col="blue")


qqnorm(x)
#abline(0, 1)
qqline(x)
```

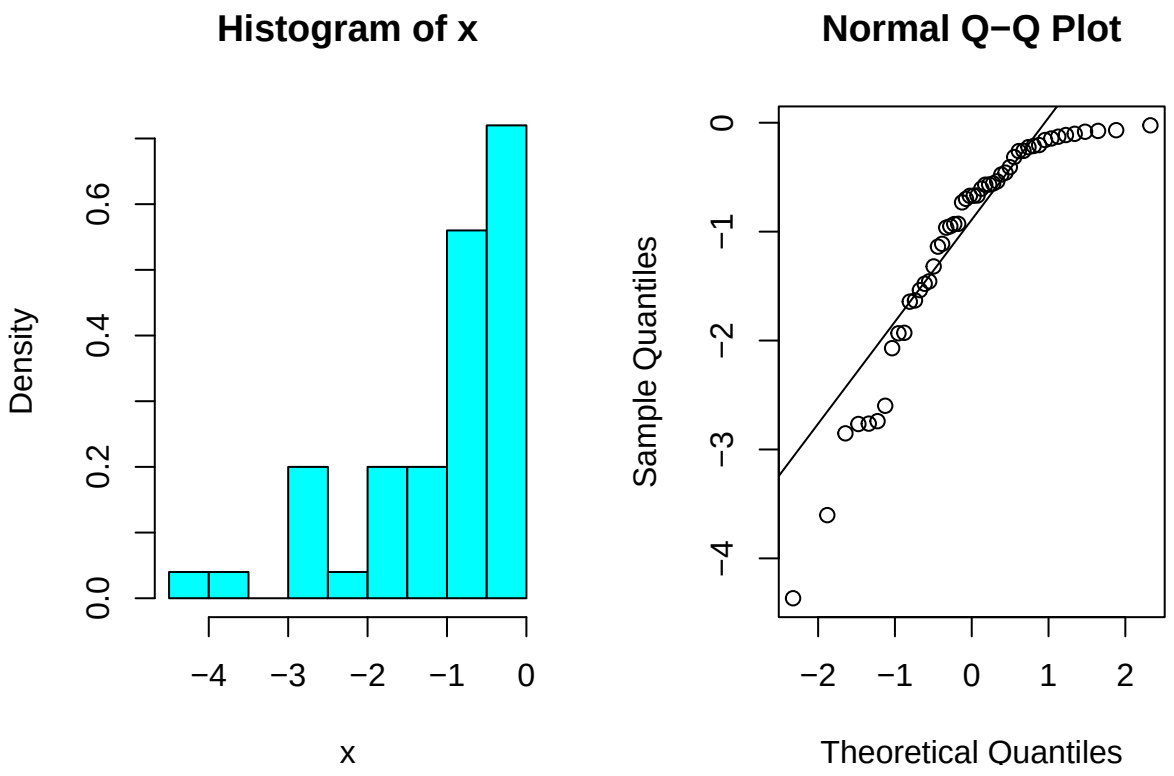



```
par(mfrow=c(1,1))

# comparison with different cases

# right skewed

x <- rexp(50)

par(mfrow=c(1,2))
hist(x, freq=FALSE, col="cyan")
qqnorm(x)
qqline(x)
```

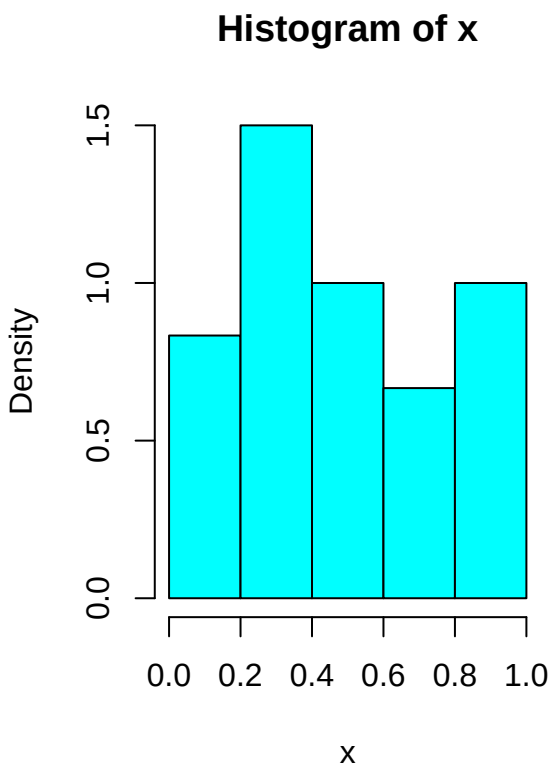


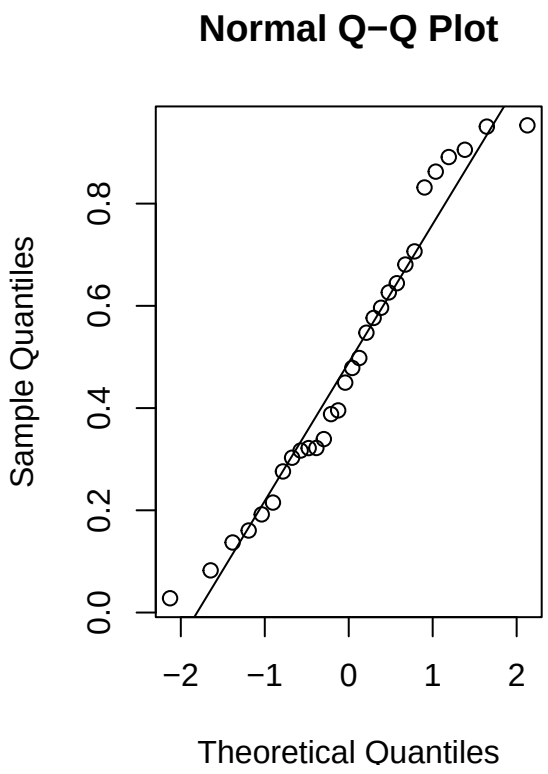



```r
par(mfrow=c(1,1))

# left skewed

x <- -rexp(50)

par(mfrow=c(1,2))
hist(x, freq=FALSE, col="cyan")
qqnorm(x)
qqline(x)
```

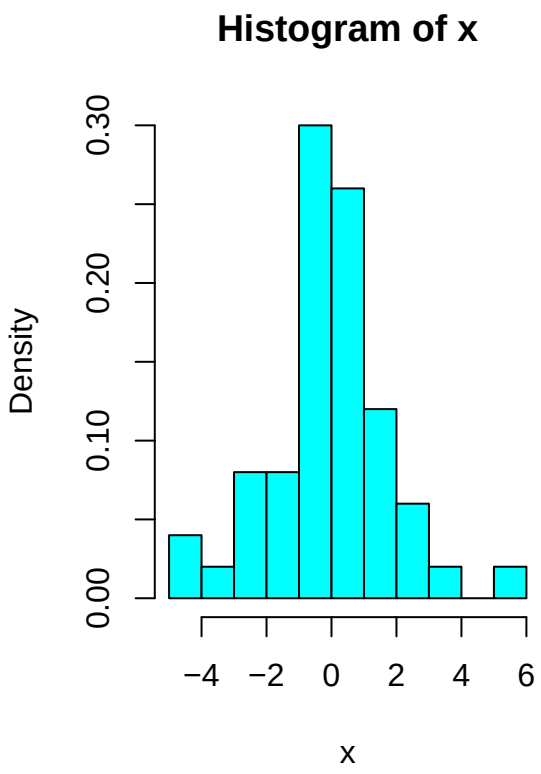


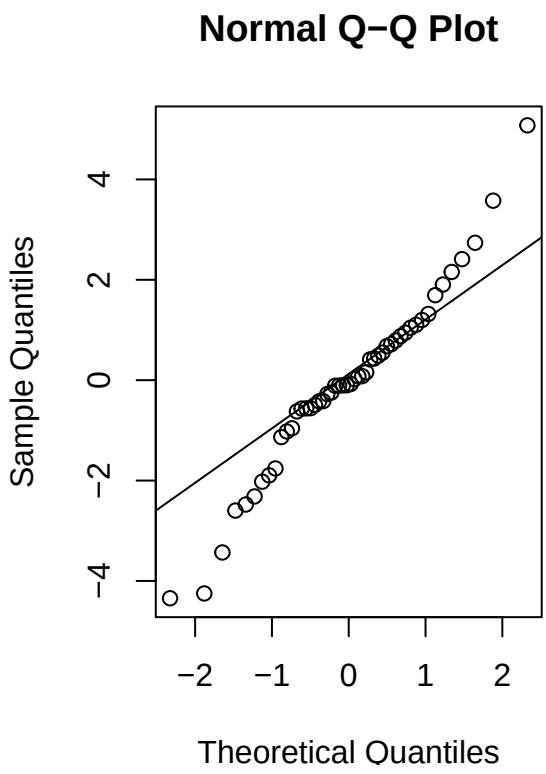

```
par(mfrow=c(1,1))


# light tails

x <- runif(30)


par(mfrow=c(1,2))
hist(x, freq=FALSE, col="cyan")
qqnorm(x)
qqline(x)
```



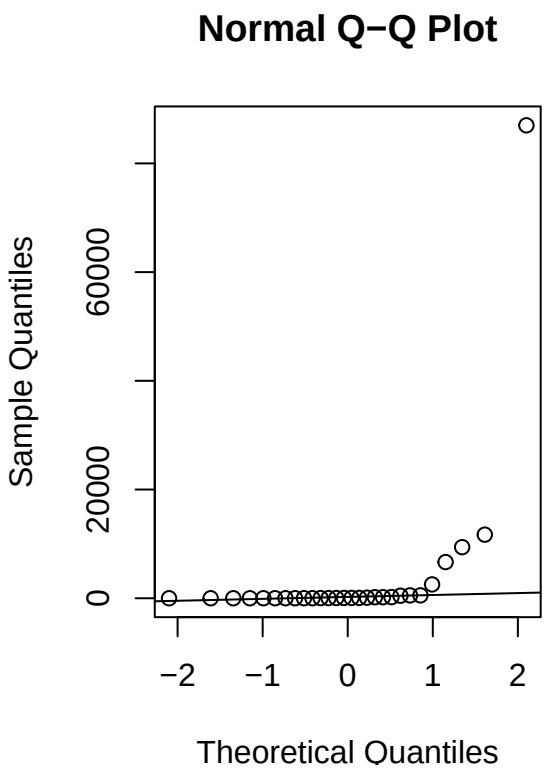


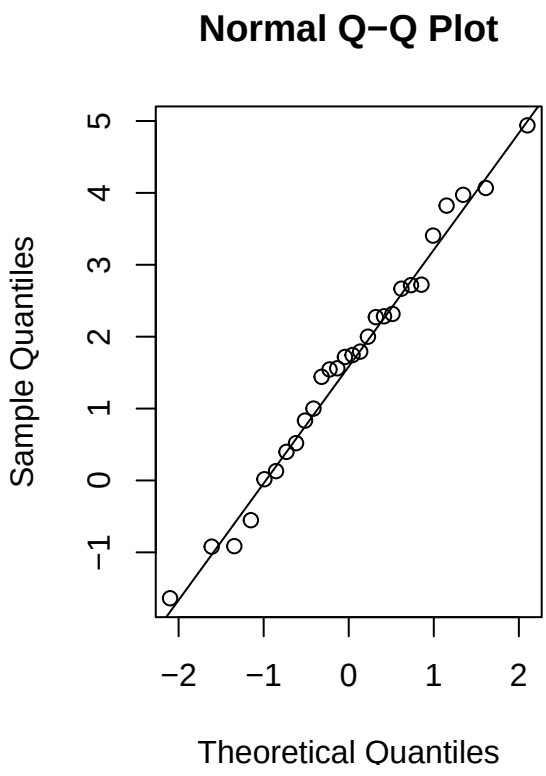

```r
par(mfrow=c(1,1))

# heavy tails

x <- rt(50, 2)

par(mfrow=c(1,2))
hist(x, freq=FALSE, col="cyan")
qqnorm(x)
qqline(x)
```



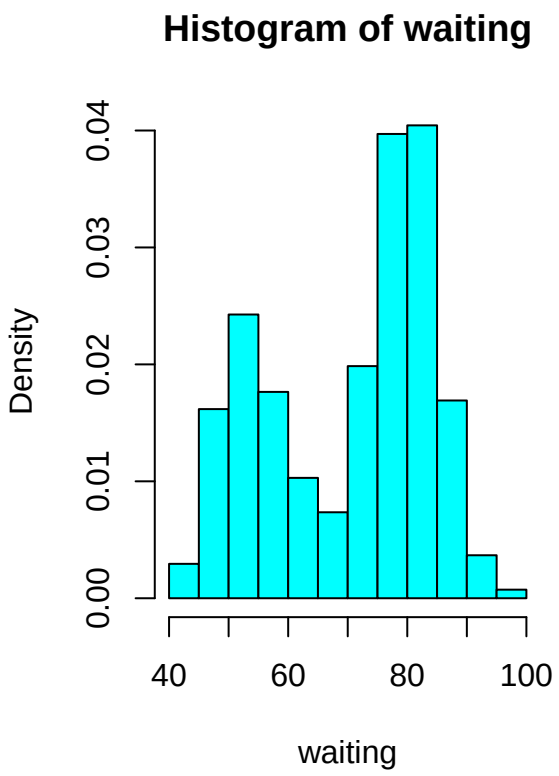


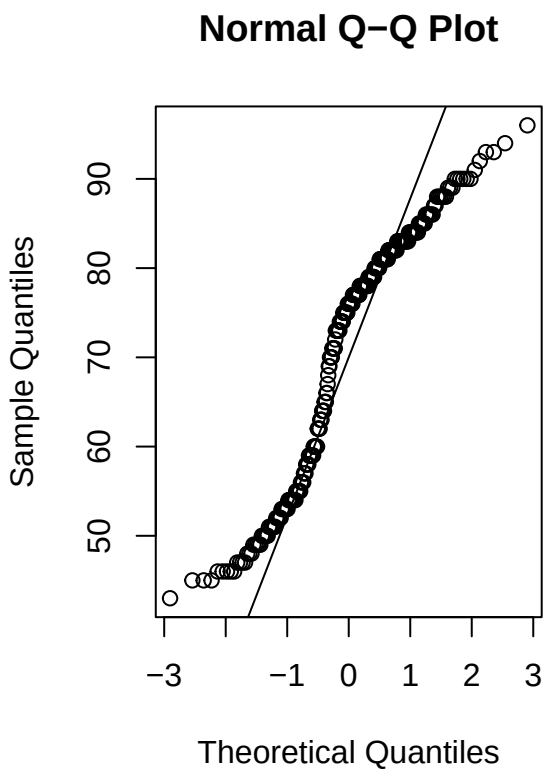

```
par(mfrow=c(1,1))

# data animals and z.scores

library(MASS)
data("Animals")
attach(Animals)

l.body <- log10(body)


par(mfrow=c(1, 2))
qqnorm(body)
qqline(body)
qqnorm(l.body)
qqline(l.body)
```



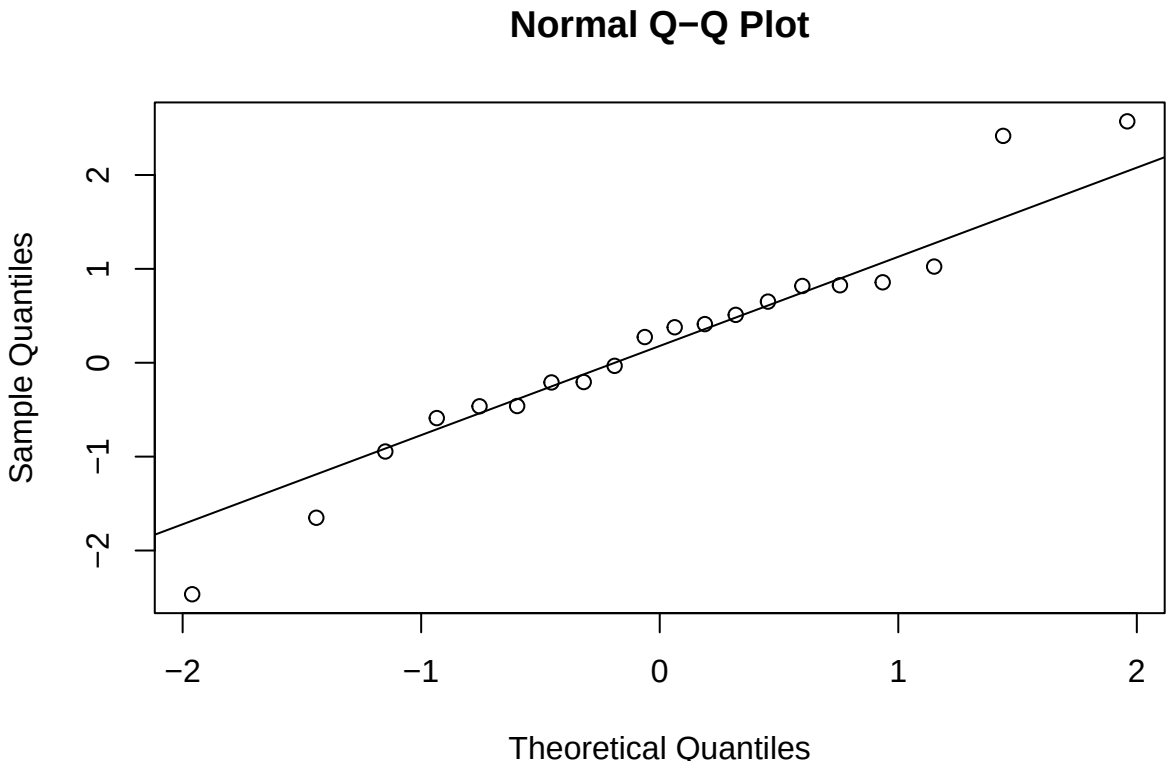



```
par(mfrow=c(1,1))

# faithful gayser data and bimodality

data(faithful)
attach(faithful)

par(mfrow=c(1,2))
hist(waiting, freq=FALSE, col="cyan")
qqnorm(waiting)
qqline(waiting)
```





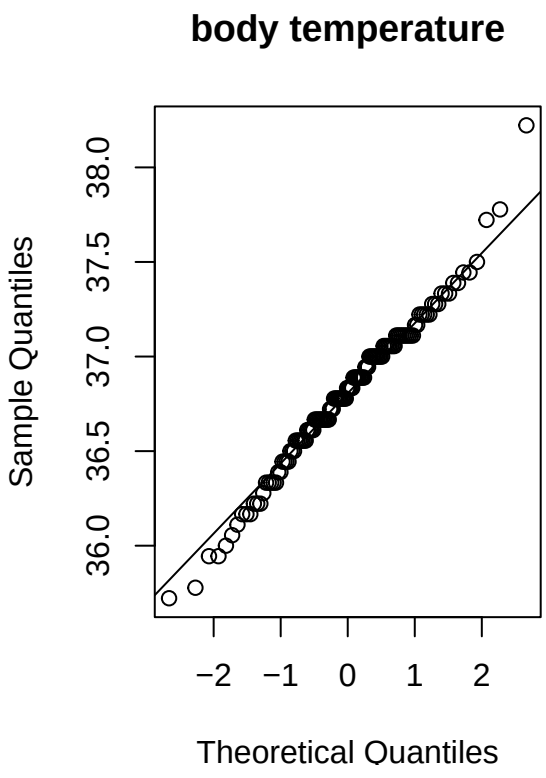

```
par(mfrow=c(1,1))

################################
# Loading data into R
################################

setwd("/Users/mattiagugole/Downloads/normtemp-data")

temp.data <- read.table("normtemp.txt", head=TRUE)
attach(temp.data)
temp.C <- (temperature-32)*5/9




# body temperatures and comparison with normal distribution

x <- rnorm(20)
qqnorm(x)
qqline(x)
```

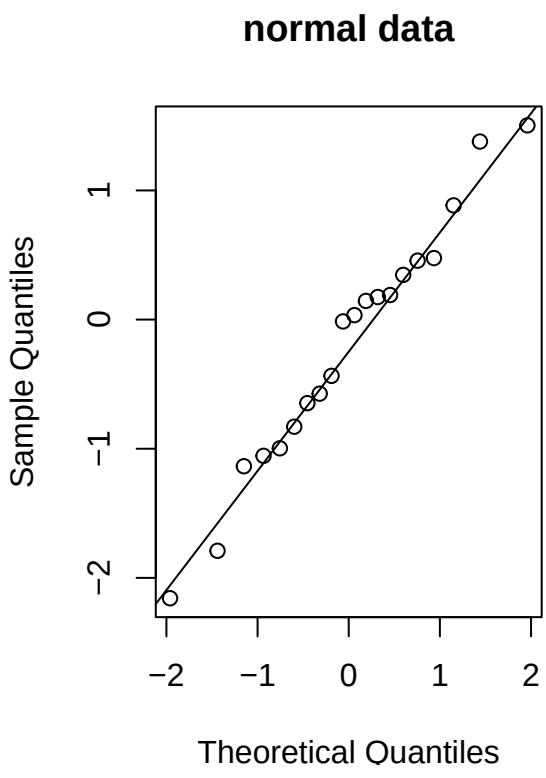



```
normtemp <- read.table("normtemp.txt", head=TRUE)
attach(normtemp)
```

```
## The following objects are masked from temp.data:
##
##     gender, hr, temperature
```

```
temp.C <- (temperature-32)*5/9


par(mfrow=c(1, 2))
qqnorm(temp.C, main="body temperature")
qqline(temp.C)

x <- rnorm(20)
qqnorm(x, main="normal data")
qqline(x)
```






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

# missing values
#
# default na.strings = "NA"
mydata <- read.table("normtemp-with-NA.txt", head=TRUE)

mydata$temperature[1:10]
```

```
##  [1] "96.3" "*"    "96.9" "97.0" "97.1" "97.1" "97.1" "97.2" "97.3" "97.4"
```

```
is.vector(mydata$temperature)
```

```
## [1] TRUE
```

```
is.character(mydata$temperature)
```

```
## [1] TRUE
```

```
is.numeric(mydata$temperature)
```

```
## [1] FALSE
```

```
# set na.strings="*"
mydata <- read.table("normtemp-with-NA.txt", head=TRUE, na.strings = "*")

mydata$temperature[1:10]
```

```
##  [1] 96.3   NA 96.9 97.0 97.1 97.1 97.1 97.2 97.3 97.4
```

```
is.vector(mydata$temperature)
```

```
## [1] TRUE
```

```
is.character(mydata$temperature)
```

```
## [1] FALSE
```

```
is.numeric(mydata$temperature)
```

```
## [1] TRUE
```

```
# factors
#
mydata <- read.table("normtemp-with-ordinal-var.txt", head=TRUE, comment.char = "#")

is.vector(mydata$age)
```

```
## [1] TRUE
```

```
is.character(mydata$age)
```

```
## [1] TRUE
```

```
is.factor(mydata$age)
```

```
## [1] FALSE
```

```
age <- mydata$age
is.vector(mydata$age)
```

```
## [1] TRUE
```

```
is.vector(age)
```

```
## [1] TRUE
```

```
is.factor(age)
```

```
## [1] FALSE
```

```
# convert into a factor (categorical variable)
age.f <- factor(age)
is.vector(age.f)
```

```
## [1] FALSE
```

```
is.factor(age.f)
```

```
## [1] TRUE
```

```
# different behaviour of the print() function and of summary()
age[1:10]
```

```
##  [1] "<30"      "<30"      "[50, 70)" "<30"      ">=70"     ">=70"
##  [7] "[50, 70)" "[50, 70)" "<30"      "<30"
```

```
age.f[1:10]
```

```
##  [1] <30      <30      [50, 70) <30      >=70     >=70     [50, 70) [50, 70)
##  [9] <30      <30
## Levels: [30, 50) [50, 70) <30 >=70
```

```
summary(age)
```

```
##    Length     Class      Mode
##       130 character character
```

```
summary(age.f)
```

```
## [30, 50) [50, 70)      <30     >=70
##       18       39       56       17
```

```
is.ordered(age.f)
```

```
## [1] FALSE
```

```
# convert into an ordered factor

age.fo <- factor(age, ordered=TRUE, levels=  c("<30", "[30, 50)", "[50, 70)", ">=70" ))

# change directly the data.frame
mydata$age <- factor(mydata$age, ordered=TRUE, levels=  c("<30", "[30, 50)", "[50, 70)", ">=70" ))



is.ordered(age.fo)
```

```
## [1] TRUE
```

```
summary(age.f)
```

```
## [30, 50) [50, 70)      <30     >=70
##       18       39       56       17
```

```
summary(age.fo)
```

```
##      <30 [30, 50) [50, 70)     >=70
##       56       18       39       17
```