> #########################
### SCRIPT FOR THE MULTIVARIATE NORMAL CALCULATIONS II
#########################
### generating samples from normal distributions
###########################################
### univariate, mean 1, variance 4, for example
### sd = standard deviation, 100 is the sample size
########################################
> y <- rnorm(100, mean = 1, sd = 2)
> summary(y)
  Min. 1st Qu.  Median   Mean 3rd Qu.   Max.
-4.05  -0.3946   0.4014   0.59  1.633   6.056
> ########################################
### summary function gives a standard numerical summaries of the data
########################################
### generating multivariate normal observations from Np(a,b)
### sample size 1000, mean vector a (of dimension p), covariance matrix b (p by p)
### store the result in a data matrix x. here p=2
########################################
> a <- c(1, 2)
> b <- matrix(c(1, 2, 2, 9), byrow = T, ncol = 2)
> x <- rmvnorm(1000, mean = a, cov = b)
> ########################
### next commands show you how to get means and variances of the columns of the
### data matrix, if you need those
########################################
> colMeans(x, na.rm = T)
[1] 1.039272 2.110964
> colVars(x, na.rm = T, unbiased = T, SumSquares = F)
[1] 1.037949 9.461873
> ####################################
### next command gives you covariance matrix
### for the data
########################################
> var(x)
     [,1]     [,2]
[1,] 1.037949 2.100056
[2,] 2.100056 9.461873
> ########################
### AS AN EXERCISE, execute the commands starting from
### sample generation for bivariate normal to the last
### command above and look at the estimated and true
### parameters.
##########################################
### Normal probability plot: for a data set versus standard normal quantiles
### I am using the data set y generated earlier

qqnorm(y)

### QQplot (or probability plot) of data with a distribution (assuming the distribution is in Splus, like chi-square)

> # generate sample, so have something to work with
> # sample size 100, degrees of freedom=25

ch <- rchisq(100, 25)

### to get qq-plot, need sample quantiles (order stats) and the corresponding chi-square distribution quantiles

> probs <- ppoints(ch)

### next get chi-square quantiles, you have to specify degrees of freedom for chi-square distribution.

> quant.chi <- qchisq(probs, df = 25)
### next plot. I will not get good fit because my the distribution of my data and
### chi-sq distribution for qq-plot are different (different degrees of freedom)
###
```r
> plot(quant.chi, sort(ch))
```

### To find the optimal lambda in the Box-Cox transformation
### plot the sample variance of the transformed data as in (4-37)
### versus lambda and choose a convenient value of lambda near the minimum.
### To plot, use the command box.cox(x), where x is a univariate
### data set (of positive values). Here is a box.cox function:(remember to compile it
### before using on a data set. This one plots on the
### interval from -1 to 1.
###
```r
> box.cox <- function(x)
{  
  a <- exp(mean(log(x)))
  y <- 1:100
  lam <- 1:100
  for(i in 1:200) {
    lam[i] <- -1.001 + 0.02 * i
    y[i] <- var((x^lam[i] - 1)/(lam[i] * a^(lam[i] - 1)))
  }
  plot(lam, y)
}
```

### Modified box.cox below will plot for
### lambda between -4 and 4
###
```r
> box.cox1 <- function(x)
{  
  a <- exp(mean(log(x)))
  y <- 1:800
  lam <- 1:800
  for(i in 1:800) {
    lam[i] <- -4.001 + 0.01 * i
    y[i] <- var((x^lam[i] - 1)/(lam[i] * a^(lam[i] - 1)))
  }
  plot(lam, y)
}
### examples of use of the `box.cox` functions

```r
> box.cox(rnorm(1000) + 10)
> box.cox(rexp(100))
```

---

```
> par(mfrow = c(1, 2))
> qqnorm(rexp(100))
> qqnorm(log(rexp(100)))
```

---

### I used lambda=0 (seems like it is convenient number close to minimum of the graph).
### Below are qqplots of the original exponential sample (left) and log(exp(sample)) (right)
### The right graph is more straight than the left one.
### By the way, the command `par(mfrow = c(1, 2))` makes the graphics space
### divided into 2 columns and 1 row. Then, I can gave two figures on one graphics window.

---

```
> box.cox(rnorm(100) + 1000)
```

---

### IF THE DATA IS NOT ALL POSITIVE, shift it by adding a number to all observations to
### make the new (shifted) data positive. However, do not add too much! Here
### is an example of warning - adding too much to the data
### to make it positive may cause loosing the right perspective

---

```r
> box.cox(rnorm(100) + 1000)
```
The data was normal, so the minimum should occur at \( \lambda = 1 \), but the function looks increasing all the time. That is because we do not have the right resolution, I shifted by too much.