ECON7333 Assignment 2

```
options(scipen=999)
set.seed(1234)
LR2 <- read.table(file="./LR2.csv", header = TRUE, sep = ",")
attach(LR2)
```

Assignment 2

Exercise 1

$$\Pr(Y=1|X=x) = \Phi(\beta_0 + \beta_1 x)$$
$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp^{-\frac{1}{2}t^2} \, dt$$

$$\Phi(z) = \mathrm{P}(Z \le z), Z \sim \mathcal{N}(0,1)$$

Thus $\Phi(\beta_0 + \beta_1 x) = \mathrm{P}(Z \le z)$

Write an `R` function that computes the maximum likelihood estimate along with bootstrapped errors.

```
# objective function
probit_mle_b <- function(x,y,...) {

  opts <- list(...)

  # probit link
  #
  probit <- function(b,x,y) {
    n <- length(y)
    ll <- 0
    for(i in 1:n) {
      z <- b[1]+b[2]*x[i]
      z <- pnorm(z, mean=0, sd=1, log.p = FALSE)
      ll <- ll + log(z)*(y[i]==1) + log(1-z)*(y[i]==0)
    }

    return(-ll)
  }

  # mle
  #
  obj = optim(c(0,0), probit, x=x, y=y)

  coef1 <- obj$par[1]
  coef2 <- obj$par[2]
```

```r
  return_list <- list(
    model = obj,
    fitted = pnorm(coef1+coef2*x,0,1),
    coefficients = c(coef1,coef2)
  )

  ## Bootstrap
  ##

  B <- 100 # boots

  b_boot = matrix(rep(0,2*B),B,2)
  n <- length(y) #n=1000
  for (i in 1:B) {
    # indices for the i-th bootstrap subsample
    ind_ = sample(n,n,replace=TRUE)
    # input vector in the subsample
    xb = x[ind_]
    # output vector in the subsample
    yb = y[ind_]

    # compute the maximum likelihood estimates
    obj = optim(c(0,0), probit, x=xb, y=yb)

    b_boot[i,1] = obj$par[1]
    b_boot[i,2] = obj$par[2]
  }

  return_list$standard_errors <- c(sd(b_boot[,1]),sd(b_boot[,2]))
  return_list$boots <- b_boot


  # If optional vector of new obs is supplied
  # then predict vector of response
  #
  if(!is.null(opts)) {
    opts <- unlist(opts)
    return_list$response = ifelse(pnorm(coef1+coef2*opts,0,1)>1/2,1,0)
  }

  return(
    return_list
  )
}
```

Apply the probit model to the `LR2` data set.

```r
# Apply probit_mle_bto LR2, supply optional vector x
#
est <- probit_mle_b(LR2$x,LR2$y,x)
est$coefficients
```

```
## [1] -4.242606 -3.086030
```

```r
#-4.242606 -3.086030

# Calc test error
1-sum(diag(prop.table(table(est$response,y))))
```

```
## [1] 0.043
```

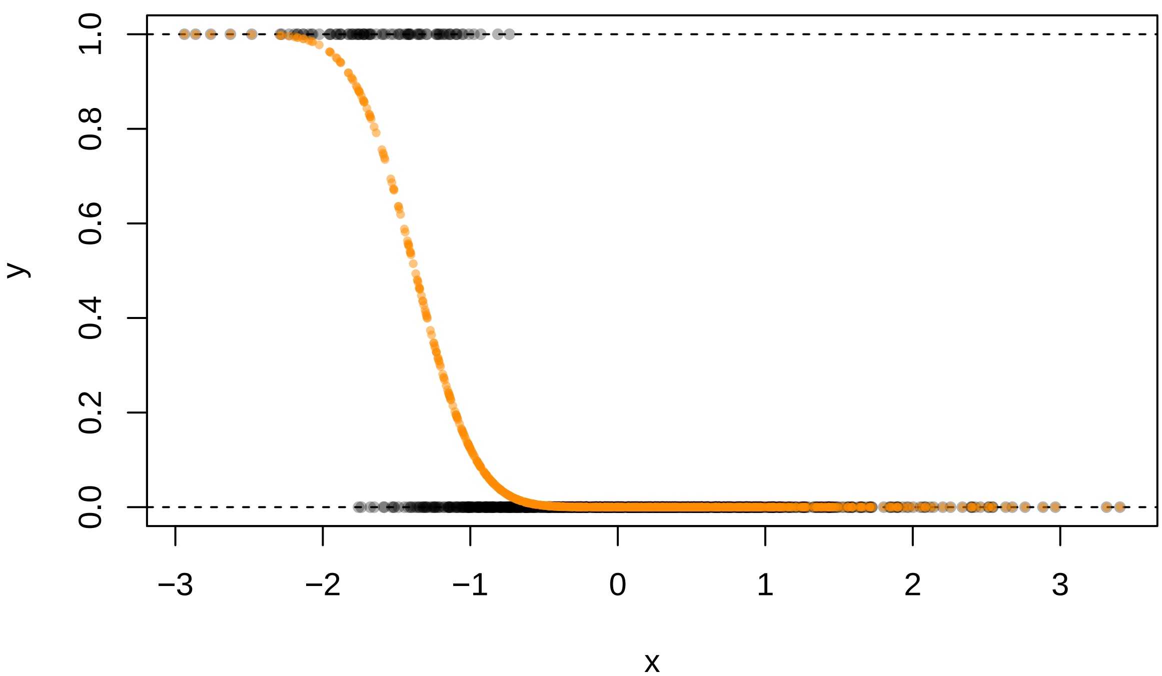
```r
# Setup train/test held out, then estimate
train <- LR2[1:800,]
test <- LR2[801:1000,]
est.2 <- probit_mle_b(train$x,train$y,test$x)
est.2$coefficients
```

```
## [1] -4.557228 -3.213275
```

```r
#-4.557228 -3.213275

# Calc test error
1-sum(diag(prop.table(table(est.2$response,test$y))))
```

```
## [1] 0.06
```

```r
glm.est <- suppressWarnings(glm(y~x,family=binomial(link = "probit")))
plot(x,y, pch=20, col=scales::alpha("black",alpha = 0.3))
abline(h=1, lty=2)
abline(h=0, lty=2)
y1 <- sort(est$fitted,TRUE)
points(sort(x),y1,pch=20,cex=0.7,col=scales::alpha("darkorange",0.5))
```



Exercise 2

Consider the model. Let X and U be two independent uniformly distributed random variables and let Y be given by the equation

$$Y = I\left(U \leq \frac{1}{1+\exp(-\beta_0 - \beta_1 X - \beta_2 X^3 - \beta_3 \log(X))}\right)$$

where

1. Assuming one observes a sample of size n of the variables X and Y, comment the underlying model for that dataset while explaining its structure in plain English.

The indicator function constructs a test that compares a standard logistic regression function and sample uniform random variable. The logistic function takes an explanatory variable x drawn from the uniform random variable X, and computes a response. The response is compared to an independent sample of a random variable from the uniform distribution, U. Where the logistic response is greater than or equal to the uniform random variable, the indicator variable classifies

2. Construct an R function that generates a sample of size n of variables X and Y.

```
link <- function(b,n) {
  X <- runif(n)
  U <- runif(n)

  logit_X <- function(b,x) {
    (1 + exp(-b[1]-b[2]*x-b[3]*x^3-b[4]*log(x)))^-1
  }

  Y <- ifelse(U<=logit_X(b,X),1,0)
  return(data.frame(X=X,U=U,Y=Y))
}
```

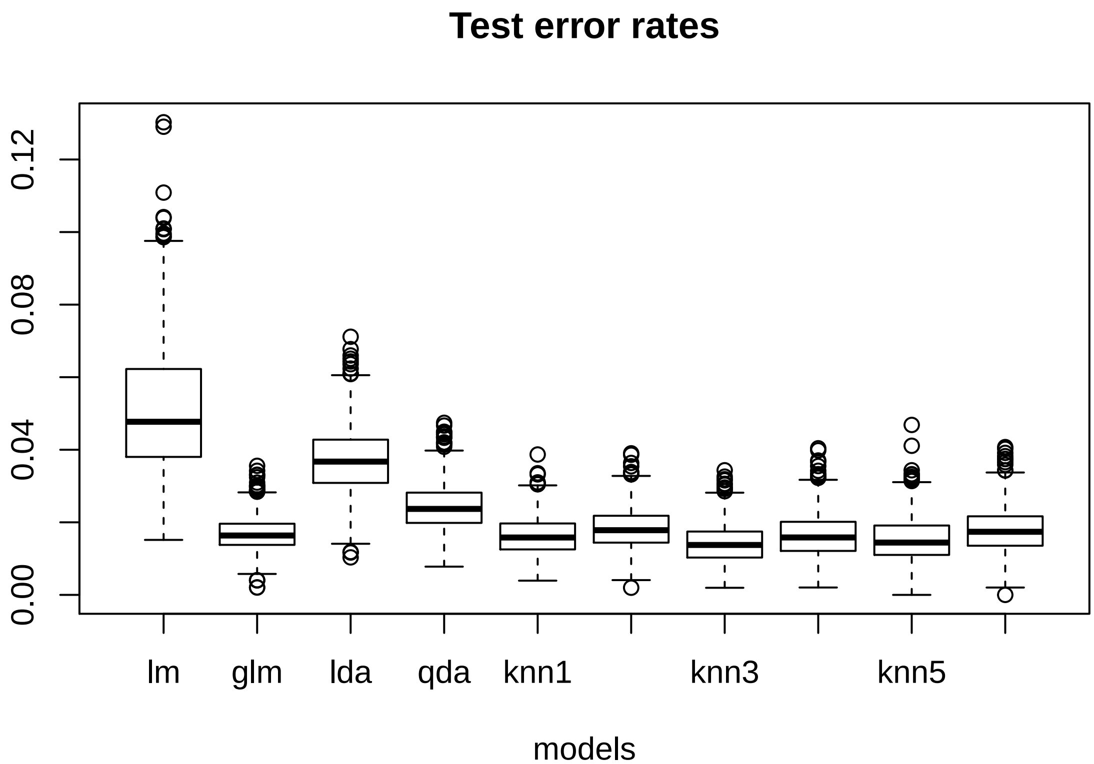
3. Constructe a box and whisker plot of test error rates.

```
# Helper function to draw samples
r.sample <- function(n,p) {
  train <- runif(n)<=p

  train
}

# Predict response convenience wrapper
r.pred <- function(model,thr=0.5,newdata=NULL) {
  if(is.null(newdata)) print("No data")
  prob <- suppressWarnings(predict(model,newdata,type="response"))

  if (class(model)%in%c("lda","qda")) {
    pred <- prob$class

    return(
      pred
    )

  } else {
    pred <- rep(0,length(prob))
    pred[prob>thr]=1

    return(
      #vector of predictions
      pred
```

```
    )
  }
}
```

```
ter <- NULL
b <- c(-4,2,5,4) # beta constants
B <- 1000 # bootstraps
n <- 1000 # sample size
p <- 1/2 # hold out half
# draw sample
#
r <- link(b,n)
for(i in 1:B) {
  train <- r.sample(n,p)
  test <- !train

  # linear probability model
  #
  r.lm <- lm(Y~.,r,subset=train)
  r.lm.pred <- r.pred(r.lm,newdata=r[test,])
  ter_ <- data.frame(lm=1-mean(r.lm.pred==r$Y[test]))

  # logistic regression
  #
  r.glm <- glm(Y~.,r,family = binomial,subset=train)
  r.glm.pred <- r.pred(r.glm,newdata=r[test,])
  ter_$glm <- 1-mean(r.glm.pred==r$Y[test])

  # linear discriminant analysis
  #
  r.lda <- MASS::lda(Y~.,r,subset=train)
  r.lda.pred <- r.pred(r.lda,newdata=r[test,])
  ter_$lda <- 1-mean(r.lda.pred==r$Y[test])

  # quadratic discriminant analysis
  #
  r.qda <- MASS::qda(Y~.,r,subset=train)
  r.qda.pred <- r.pred(r.qda,newdata=r[test,])
  ter_$qda <- 1-mean(r.qda.pred==r$Y[test])

  # K-nearest neighbours
  #
  r.knn1 <- class::knn(
    use.all = TRUE,
    train=r[train,1:2],
    test=r[test,1:2],
    cl=r$Y[train],
    k=1
  )
  ter_$knn1 <- 1-mean(r.knn1==r$Y[test])

  r.knn2 <- class::knn(
    use.all = TRUE,
    train=r[train,1:2],
```

```
    test=r[test,1:2],
    cl=r$Y[train],
    k=2
  )
  ter_$knn2 <- 1-mean(r.knn2==r$Y[test])

  r.knn3 <- class::knn(
    use.all = TRUE,
    train=r[train,1:2],
    test=r[test,1:2],
    cl=r$Y[train],
    k=3
  )
  ter_$knn3 <- 1-mean(r.knn3==r$Y[test])

  r.knn4 <- class::knn(
    use.all = TRUE,
    train=r[train,1:2],
    test=r[test,1:2],
    cl=r$Y[train],
    k=4
  )
  ter_$knn4 <- 1-mean(r.knn4==r$Y[test])

  r.knn5 <- class::knn(
    use.all = TRUE,
    train=r[train,1:2],
    test=r[test,1:2],
    cl=r$Y[train],
    k=5
  )
  ter_$knn5 <- 1-mean(r.knn5==r$Y[test])

  r.knn6 <- class::knn(
    use.all = TRUE,
    train=r[train,1:2],
    test=r[test,1:2],
    cl=r$Y[train],
    k=6
  )
  ter_$knn6 <- 1-mean(r.knn6==r$Y[test])

  ter <- rbind(ter,ter_)
}

boxplot(ter, xlab="models", main="Test error rates")
```



Variance and distribution of test error rates observed from the bootstrap.