TDDE07 - Lab 2

Ludwig Thaung (ludth852), Elon Brange (elobr959

5/4/2019

Task 1

(a)

```
TempLinkoping <- read_csv("TempLinkoping.csv")
```

```
## Parsed with column specification:
## cols(
##   time = col_double(),
##   temp = col_double()
## )
```

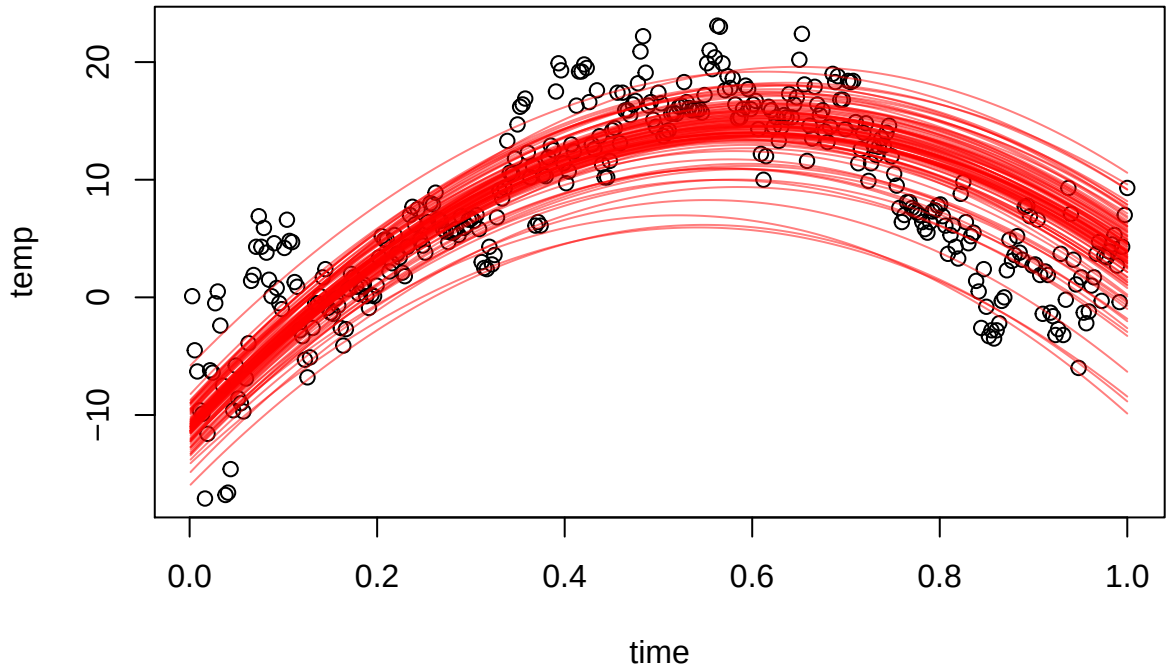
```
x = TempLinkoping['time']
y = TempLinkoping['temp']
x['time2'] = x^2
x['1'] = x['time']/x['time']
ph = x['time']
#Just to have the betas in right order
x['time'] = x['1']
x['1'] = x['time2']
x['time2'] = ph

matrix_x = data.matrix(x)
matrix_y = data.matrix(y)

mu0 = c(-11,85,-70)
omega0 = matrix(c(0.03, 0, 0, 0, 0.01, 0, 0, 0, 0.03), 3, 3)
v0 = 3
sigmasq0 = 0.03

e = rnorm(1, mean = 0, sd = sigmasq0)


betahat = inv((t(matrix_x)%*%matrix_x))%*%(t(matrix_x)%*%matrix_y)
randomSigma2 <- rinvchisq(n = 10, df = v0, scale = sigmasq0)
randomBetas <- c()
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
plot(TempLinkoping, col="black")
for(singleSigma in randomSigma2) {
  randomBeta <- rmvt(n = 10,mu = t(mu0), S = singleSigma*inv(omega0))
```

```
  for(k in 1:10) {
    ys = c()
    xs = cbind(matrix(1, 1000, 1), matrix(1:1000, 1000, 1)/1000, (matrix(1:1000, 1000, 1)/1000)^2)
    ys = xs%*%randomBeta[k, ]
    #for (i in 1:1000) {
    #  y = randomBeta[k, 1] + randomBeta[k, 2]*i/1000 + randomBeta[k, 3]*(i/1000)^2
    #  ys = c(ys, y)
    #  xs = c(xs, i/1000)
    #}
    lines(matrix(1:1000, 1000, 1)/1000, ys, col=alpha("red", 0.5))
  }
}
```

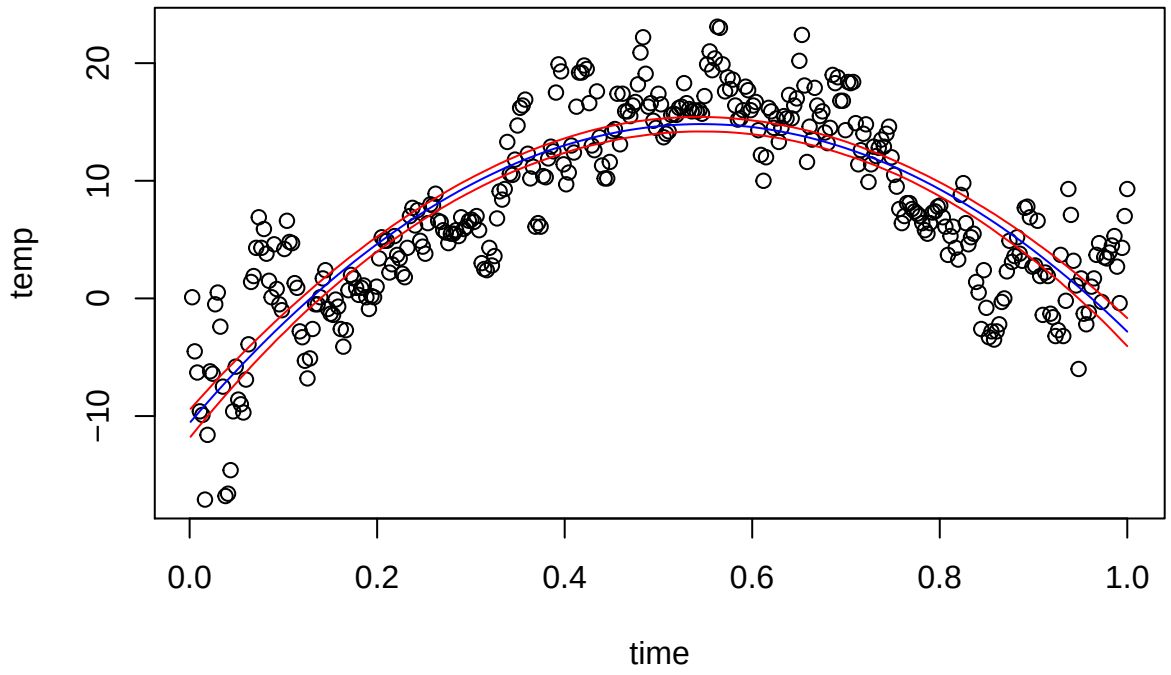


The collection of the curves look reasonable as they seem to follow the generalized curve of the datapoints.

(b)

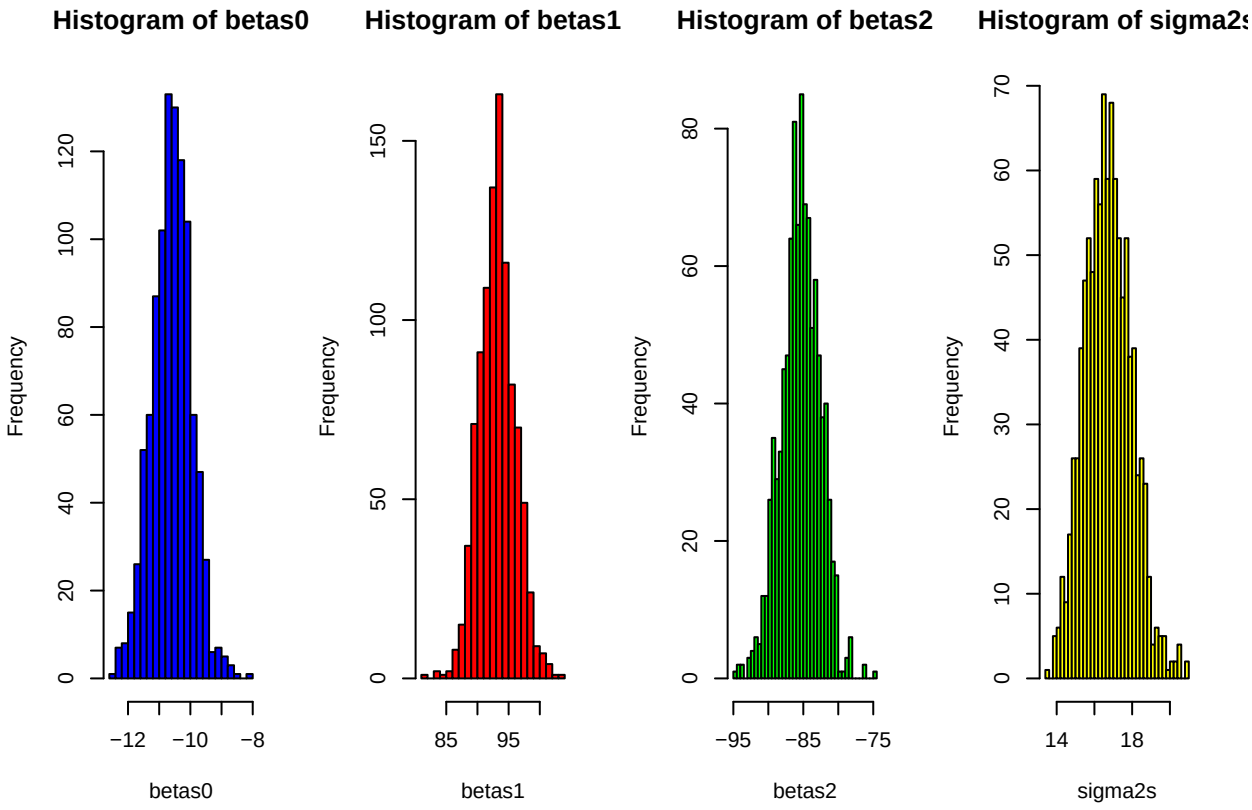
```
muN = inv(t(matrix_x)%*%matrix_x + omega0)%*%(t(matrix_x)%*%matrix_x%*%betahat + omega0%*%mu0)
omegaN = t(matrix_x)%*%matrix_x + omega0
vN = v0 + 366
vNsigmaN2 = v0*sigmasq0 + (t(matrix_y)%*%matrix_y + t(mu0)%*%omega0%*%mu0 - t(muN)%*%omegaN%*%muN)

randomSigma2 <- rinvchisq(n = 1000, df = vN, scale = (vNsigmaN2/vN))
randomBetas <- c()
plot(TempLinkoping, col="black")
sigmas = data.frame(randomSigma2)
y_df = data.frame(matrix(1, 1000, 1))
betas0 = c()
betas1 = c()
betas2 = c()
sigma2s = c()
```

```
ibeta = 0
for(singleSigma in randomSigma2) {
  randomBeta <- rmvt(n = 1,mu = t(muN), S = singleSigma*inv(omegaN))
  ibeta = ibeta + 1
  ys = c()
  for (i in 1:1000) {
    y = randomBeta[1, 1] + randomBeta[1, 2]*i/1000 + randomBeta[1, 3]*(i/1000)^2
    ys = c(ys, y)
  }
  betas0 = c(betas0, randomBeta[1, 1])
  betas1 = c(betas1, randomBeta[1, 2])
  betas2 = c(betas2, randomBeta[1, 3])
  sigma2s = c(sigma2s, singleSigma)
  y_df[paste0("trial", ibeta)] <- data.frame(ys)
}
xs = c()
for(i in 1:1000) {
  xs = c(xs, i/1000)
}
y_df = subset(y_df, select = -c(1) )

mediany = matrix(1, 1000, 1)
lowery = matrix(1, 1000, 1)
uppery = matrix(1, 1000, 1)
for (row in 1:nrow(y_df)) {
  mediany[row] = median(as.numeric(as.vector(y_df[row, ])))
  lowery[row] = quantile(x = as.numeric(as.vector(y_df[row, ])), probs = 0.025)
  uppery[row] = quantile(x = as.numeric(as.vector(y_df[row, ])), probs = 0.975)
}
plot(TempLinkoping, col="black")
lines(xs, mediany, col="blue")
lines(xs, lowery, col="red")
lines(xs, uppery, col="red")
```



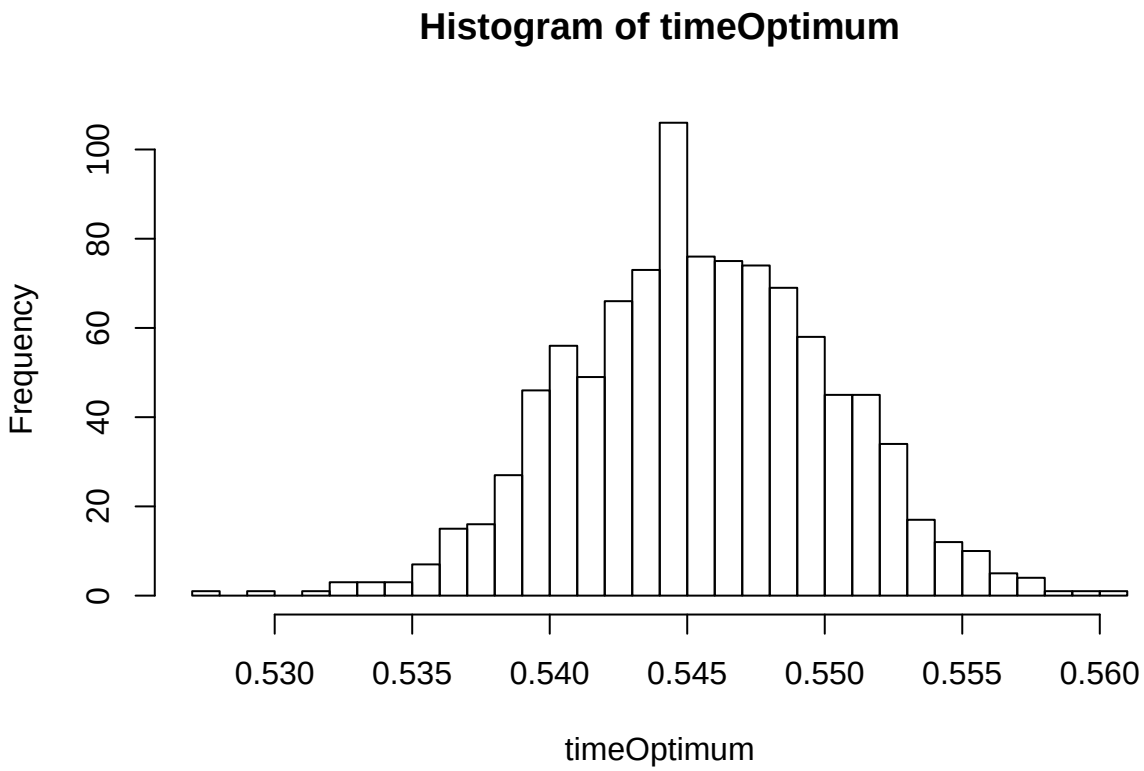
The interval band does not contain all the datapoints since it is a 95% approximation of the regression model and not the data points.

```
attach(mtcars)
par(mfrow=c(1,4))
hist(betas0, nclass=30, col='blue')
hist(betas1, nclass=30, col='red')
hist(betas2, nclass=30, col='green')
hist(sigma2s, nclass=30, col='yellow')
```


Histogram of betas0
Frequency
0
20
40
60
80
100
120
−12
−10
−8
betas0
Histogram of betas1
Frequency
0
50
100
150
85
95
betas1
Histogram of betas2
Frequency
0
20
40
60
80
−95
−85
−75
betas2
Histogram of sigma2s
Frequency
0
10
20
30
40
50
60
70
14
18
sigma2s

(c)

```
timeOptimum = -betas1/(2*betas2)
hist(timeOptimum, nclass=30)
```



(d)

If there is a higher order but we are more certain that higher order parameters are not needed we can set the omega-values high as it creates a stronger prior and the mu-prior values at 0. Which would mitigate the problem of an unneccesary high order polynom.

Task 2

(a)

```
womenWork<-read.table("WomenWork.dat",header=TRUE)
glmModel <- glm(Work ~ 0 + ., data = womenWork, family = binomial)
print(glmModel)
```

```
##
## Call:  glm(formula = Work ~ 0 + ., family = binomial, data = womenWork)
##
## Coefficients:
##    Constant   HusbandInc    EducYears     ExpYears    ExpYears2
##     0.64430     -0.01977      0.17988      0.16751     -0.14436
##         Age  NSmallChild    NBigChild
##    -0.08234     -1.36250     -0.02543
##
## Degrees of Freedom: 200 Total (i.e. Null);  192 Residual
## Null Deviance:       277.3
## Residual Deviance: 222.7     AIC: 238.7
```

(b)

```
chooseCov <- c(1:8) # Here we choose which covariates to include in the model
tau <- 10; # Prior scaling factor such that Prior Covariance = (tau^2)*I

# install.packages("mvtnorm") # Loading a package that contains the multivariate normal pdf
library("mvtnorm") # This command reads the mvtnorm package into R's memory. NOW we can use dmvnorm fun

# Loading data from file
Data<-read.table("WomenWork.dat",header=TRUE)  # Spam data from? Hastie et al.
y <- as.vector(womenWork[,1]); # Data from the read.table function is a data frame. Let's convert y and
X <- as.matrix(womenWork[,2:9]);
covNames <- names(womenWork)[2:length(names(womenWork))];
X <- X[,chooseCov]; # Here we pick out the chosen covariates.
covNames <- covNames[chooseCov];
nPara <- dim(X)[2];

# Setting up the prior
mu <- as.vector(rep(0,nPara)) # Prior mean vector
Sigma <- tau^2*diag(nPara);

LogPostLogistic <- function(betaVect,y,X,mu,Sigma){
  nPara <- length(betaVect);
  linPred <- X%*%betaVect;

  logLik <- sum( linPred*y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer away from he
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), Sigma, log=TRUE);
  return(logLik + logPrior)
}
```

```
initVal <- as.vector(rep(0,dim(X)[2]));
logPost = LogPostLogistic;
OptimResults<-optim(initVal,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),control=list(fnscale=-1),hess
```

```
# Printing the results to the screen
names(OptimResults$par) <- covNames # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(-solve(OptimResults$hessian))) # Computing approximate standard deviations.
approxPostStd <- sqrt(diag(-solve(OptimResults$hessian)))
names(approxPostStd) <- covNames # Naming the coefficient by covariates

betatilde = OptimResults$par
#Betatilde:
print(betatilde)
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  0.62672884 -0.01979113  0.18021897  0.16756670 -0.14459669 -0.08206561
## NSmallChild   NBigChild
## -1.35913317 -0.02468351
```

```
# Jacobiany beta:
print(approxPostStd)
```

```
##    Constant  HusbandInc   EducYears    ExpYears   ExpYears2         Age
##  1.50533138  0.01589983  0.07885556  0.06596754  0.23575129  0.02680412
## NSmallChild   NBigChild
##  0.38892439  0.14132327
```

```
#Intervall NSmallChild
upperb = betatilde["NSmallChild"] + 1.96*approxPostStd["NSmallChild"]
lowerb = betatilde["NSmallChild"] - 1.96*approxPostStd["NSmallChild"]
print(upperb)
```

```
## NSmallChild
##  -0.5968414
```

```
print(lowerb)
```

```
## NSmallChild
##   -2.121425
```

(c)

```
covarMatrix = -inv(OptimResults$hessian)

ladyInput = c(1, 10, 8, 10, (10/10)^2, 40, 1, 1)
workOrNots = c()
workOrNotsBinary = c()
betas = rmvt(n = 1000,mu = matrix(betatilde), S = covarMatrix)
for(row in 1:nrow(betas)) {
  working = exp(ladyInput %*% betas[row, ])/(1 + exp(ladyInput %*% betas[row, ]))
  workingBinary = round(working)
  workOrNots = c(workOrNots, working)
  workOrNotsBinary = c(workOrNotsBinary, workingBinary)
}
attach(mtcars)
```

```
## The following objects are masked from mtcars (pos = 3):
##
##     am, carb, cyl, disp, drat, gear, hp, mpg, qsec, vs, wt
```

```
par(mfrow=c(1,2))
hist(workOrNots, n=30)
hist(workOrNotsBinary, n=30)
```