Computer Lab 2

phiho267 & zijfe244

9-4-2019

Contents

```
##
## Attaching package: 'LaplacesDemon'
```

```
## The following objects are masked from 'package:mvtnorm':
##
##     dmvt, rmvt
```

1. Linear and polynomial regression

The dataset *TempLinkoping.txt* contains daily temperatures (in Celcius degrees) at *Malmslätt*, Linköping over the course of the year 2016 (366 days since 2016 was a leap year). The response variable is temp and the covariate is

$$time = \frac{the\ number\ of\ days\ since\ beginning\ of\ year}{366}$$

The task is to perform a Bayesian analysis of a quadratic regression

$$temp = \beta_0 + \beta_1 \cdot\ time + \beta_2 \cdot\ time^2 + \varepsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$$

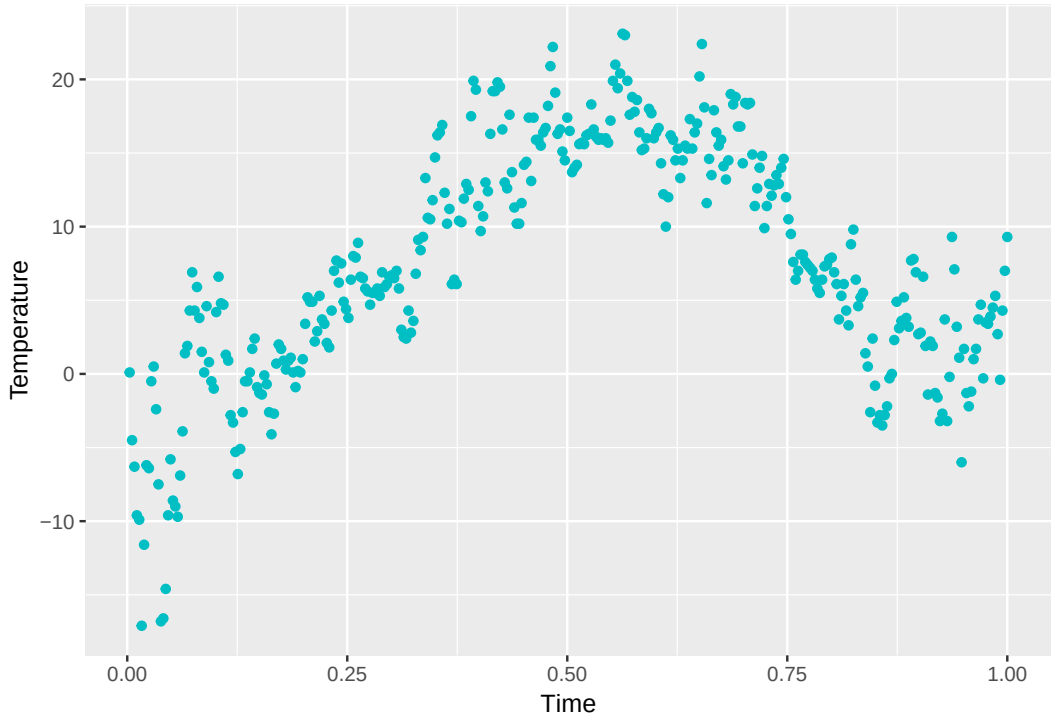
(a) Determining the prior distribution of the model parameters.

Use the conjugate prior for the linear regression model. Your task is to set the prior hyperparameters $\mu_0, \Omega_0, \upsilon_0$ and $\sigma_0^2 = 1$ to sensible values. Start with $\mu_0 = -10, 100, -100)^T$,$\Omega_0 = 0.01 \cdot I_3, , \upsilon_0 = 4, \sigma_0^2 = 1$. Check if this prior agrees with your prior opinions by simulating draws from the joint prior of all parameters and for every draw compute the regression curve. This gives a collection of regression curves, one for each draw from the prior. Do the collection of curves look reasonable? If not, change the prior hyperparameters until the collection of prior regression curves do agree with your prior beliefs about the regression curve. [Hint: the R package *mvtnorm* will be handy. And use your $Inv - \chi^2$ simulator from Lab 1.]

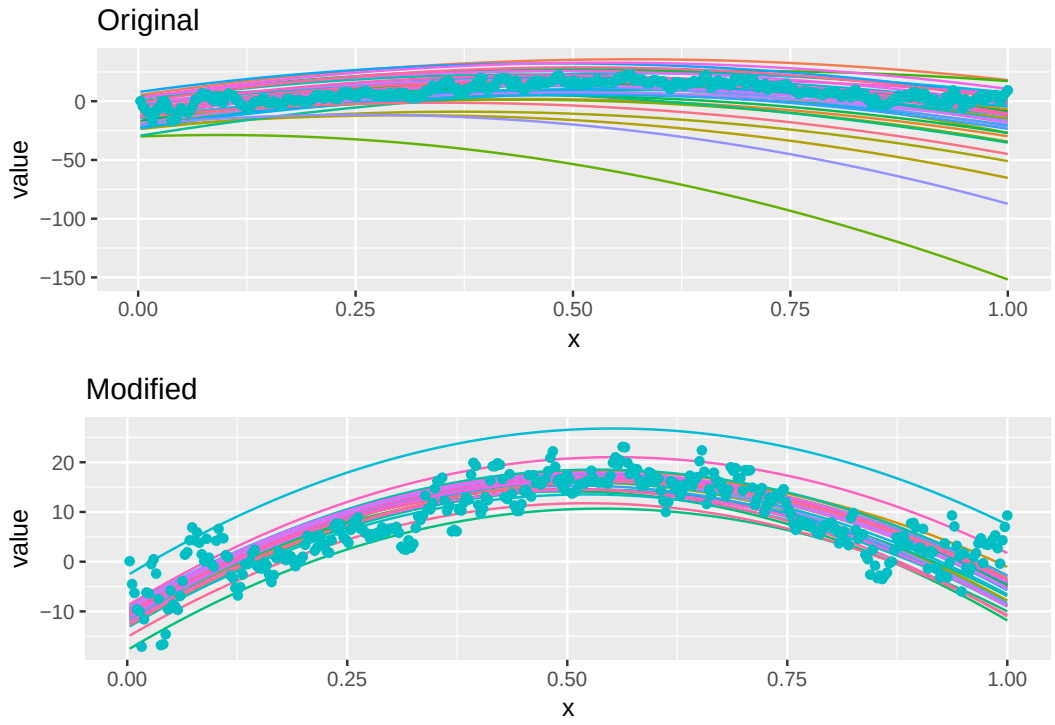
```
# linear model - without beta
lmTemp = lm( temp ~ time + I(time^2), data = tempdata)
summary(lmTemp)
```

```
##
## Call:
## lm(formula = temp ~ time + I(time^2), data = tempdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.0408  -2.6971  -0.1414   2.5157  12.2085
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.6754     0.6475  -16.49   <2e-16 ***
## time         93.5980     2.9822   31.39   <2e-16 ***
## I(time^2)   -85.8311     2.8801  -29.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.107 on 363 degrees of freedom
## Multiple R-squared:  0.7318, Adjusted R-squared:  0.7304
## F-statistic: 495.3 on 2 and 363 DF,  p-value: < 2.2e-16
```

```r
# plot the data
ggplot(tempdata, aes(x = time, y=temp)) +
  geom_point(color = "#00BFC4") +                    # default blue
  ylab("Temperature") + xlab("Time")
```




```r
# create simulation
set.seed(123456)

# joint conjugate prior
N <- 40  # nr of simulations

# data frame to save prior coef data
prior <- matrix(ncol = 3, nrow = N)
for (i in 1:N) {

  # lecture 5 slide 7 - Inv - chi^2(v_0,sigma^2_0)
  var  <- LaplacesDemon::rinvchisq(1 ,v_0, sigma2_0)

  # solve(A)    Inverse of A where A is a square matrix
  beta <- MASS::mvrnorm(1, mu_0, var*solve(omega_0))

  prior[i,1:3] <- beta
}
```

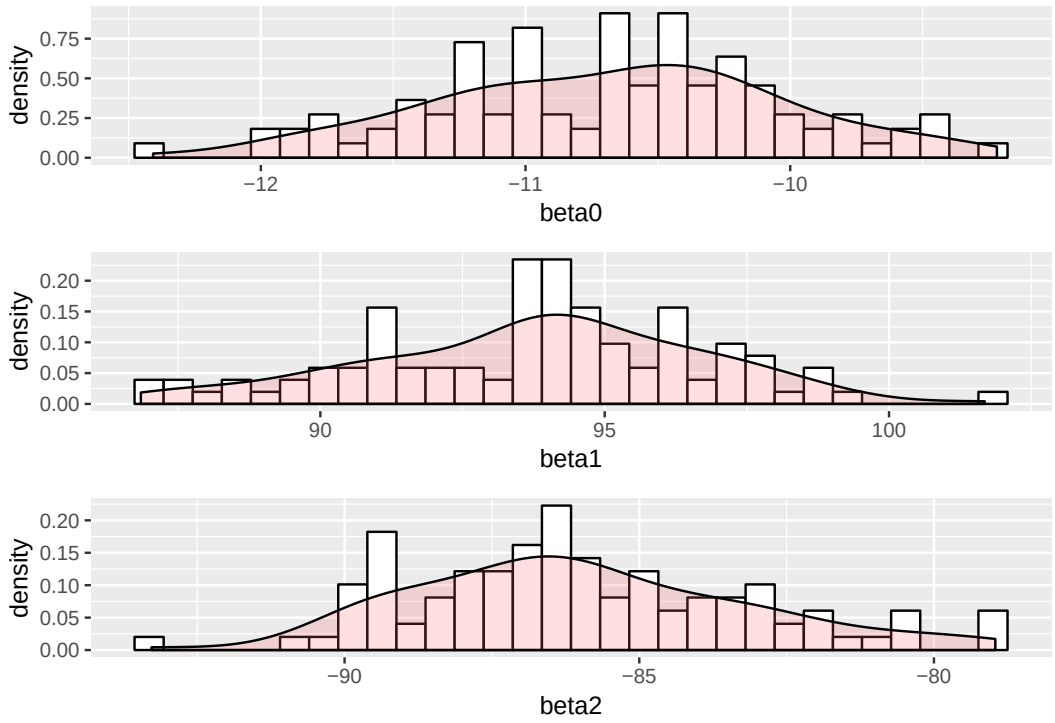


(b) Write a program that simulates from the joint posterior distribution

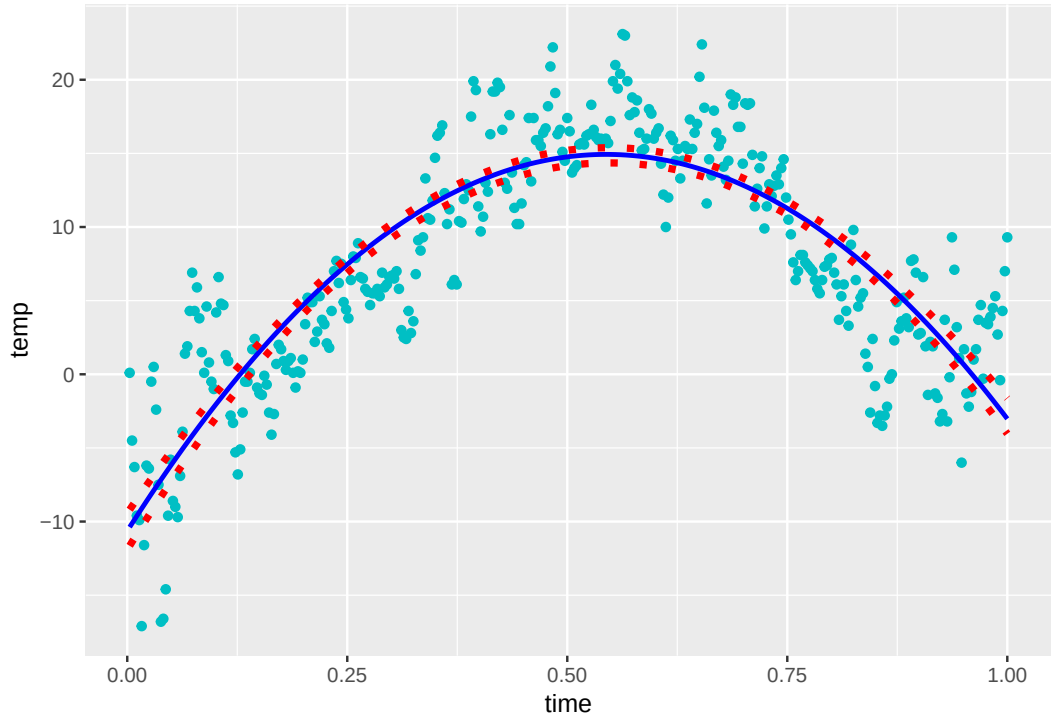
of $\beta_0, \beta_1, \beta_2$ and σ^2. Plot the marginal posteriors for each parameter as a histogram. Also produce another figure with a scatter plot of the temperature data and overlay a curve for the posterior median of the regression function $f(\ time) = \beta_0 + \beta_1 \cdot\ time, \beta_2 \cdot\ time^2$, computed for every value of *time*. Also overlay curves for the lower 2.5% and upper 97.5% posterior credible interval for $f(\ time)$. That is, compute the 95% equal tail posterior probability intervals for every value of time and then connect the lower and upper limits of the interval by curves. Does the interval bands contain most of the data points? Should they?

```
# posterior
k        <- 3 #  nr of regression coefficients
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
mu_n     <- solve(t(X) %*% X + omega_0) %*% (t(X) %*% X %*% beta_hat + omega_0 %*% mu_0)
omega_n    <- t(X) %*% X+omega_0
v_n      <- v_0 + n
sigma2_n <- (v_0 * sigma2_0 + (t(y) %*% y + t(mu_0) %*% omega_0 %*% mu_0 - t(mu_n) %*% omega_n %*% mu_n

# marginal posterior
###### Zijie idea:
# The arguments of marginal posterior of conjugate prior
# are guessed form marginal posterior of uniform prior (page 6)
######

data_hist <- as.data.frame(
  mvtnorm::rmvt(n = 100, delta = mu_n, df = n-k,
                sigma = as.numeric(sigma2_n) * solve(omega_n))
  )
cnames <- c("beta0", "beta1", "beta2")
colnames(data_hist) <- cnames
```

```
f <- function(cname){
  ggplot(data_hist, aes_string(x=cname)) +
    geom_histogram(aes(y = ..density..),
                   colour = "black",
                   fill   = "white",
                   bins   = 30) +
    geom_density(alpha = .2, fill = "#FF6666")
}
plot(arrangeGrob(grobs = lapply(cnames, f)))
```




```
# overlay a curve for the posterior median
beta_median = matrixStats::colMedians(as.matrix(data_hist))
pred.1b = beta_median %*% t(X)

# credible intervals
    # each column represents all possible values at a fixed time
preds <- as.matrix(data_hist) %*% t(X)
pred_interval <- data.frame(nrow = n, nrow = 2)

colnames(pred_interval) <- c("lower","upper")
for(i in 1:n){
  data_t <- preds[,i]
  pred_interval[i,] <- quantile(data_t, probs = c(0.025,0.975))
}
data.1b <- cbind(tempdata, t(pred.1b), pred_interval)
ggplot(data.1b) +
  geom_point(aes(x = time, y = temp), color = "#00BFC4") +
  geom_line(aes(x = time, y = t(pred.1b)),color = "Blue",size = 1) +
  geom_line(aes(x = time, y = lower), color = "Red", linetype = "dotted", size = 1.5) +
  geom_line(aes(x = time, y = upper), color = "Red", linetype = "dotted", size = 1.5)
```

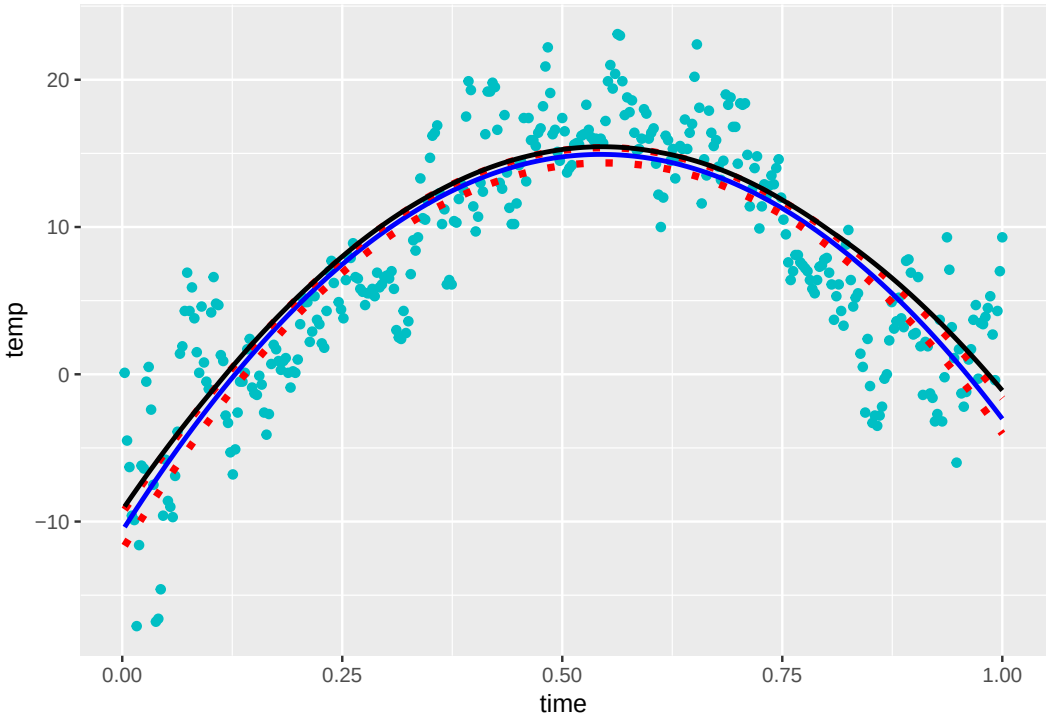


NO, the cridible intervals do not contain most of points. They shouldn't. Credible intervals describe the robustness of beta, not of model. There exists variance when we predict Y using our regression model, and the corresponding intervals describing the condience of model is predective interval.

(c) Locate the time with the highest expected temperature

It is of interest to locate the *time* with the highest expected temperature (that is, the time where $f($ *time*$)$ is maximal). Let??s call this value $\widetilde{x}$. Use the simulations in b) to simulate from the posterior distribution of $\widetilde{x}$. [Hint: the regression curve is a quadratic. You can find a simple formula for $\widetilde{x}$ given β_0, β_1 and β_2.]

```
pred_highest <- c()
for(i in 1:366){
  pred_highest[i] <- max(preds[,i])
}
data.1c <- cbind(tempdata, t(pred.1b), pred_interval, pred_highest)
ggplot(data.1c) +
  geom_point(aes(x = time, y = temp), color = "#00BFC4") +
  geom_line(aes(x = time, y = t(pred.1b)),color = "Blue",size = 1) +
  geom_line(aes(x = time, y = lower), color = "Red", linetype = "dotted", size = 1.5) +
  geom_line(aes(x = time, y = upper), color = "Red", linetype = "dotted", size = 1.5) +
  geom_line(aes(x = time,y = pred_highest), color = "black", linetype = "solid", size = 1)
```



(d) Polynomial model of order 7

Say now that you want to *estimate a polynomial model of order 7*, but you suspect that higher order terms may not be needed, and you worry about overfitting. Suggest a suitable prior that mitigates this potential problem. You do not need to compute the posterior, just write down your prior. [Hint: the task is to specify μ_0 and Ω_0 in a smart way.]

```r
# clean up environment
rm(list=ls())
```

2. Posterior approximation for classification with logistic regression

The dataset **WomenWork.dat** contains $n = 200$ observations (i.e. women) on the following nine variables:

(a) Consider the logistic regression

$$Pr(y = 1 \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{x}^T \beta)}{1 + \exp(\boldsymbol{x}^T \beta)}$$

where y is the binary variable with $y = 1$ if the woman works and $y = 0$ if she does not. $\boldsymbol{x}$ is a 8-dimensional vector containing the eight features (including a one for the constant term that models the intercept). Fit the logistic regression using maximum likelihood estimation by the command: **glmModel <- glm(Work ∼ 0 + ., data = WomenWork, family = binomial)**. Note how I added a zero in the model formula so that R doesn???t add an extra intercept (we already have an intercept term from the *Constant* feature). Note also that a dot (.) in the model formula means to add all other variables in the dataset as features. *family = binomial* tells R that we want to fit a logistic regression.

```r
# load the data & define y (response) and x 8 dimensional vector of the features
WomenWork = read.table("WomenWork.dat", header = TRUE)
y = WomenWork$Work
X = WomenWork[,2:9]
X = as.matrix(X)
n = nrow(WomenWork) # nr of observations
```

```r
# logistic regression
glmModel <- glm(Work ~ 0 + ., data = WomenWork, family = binomial)
summary(glmModel)
```

```
##
## Call:
## glm(formula = Work ~ 0 + ., family = binomial, data = WomenWork)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1662  -0.9299   0.4391   0.9494   2.0582
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## Constant     0.64430    1.52307   0.423 0.672274
## HusbandInc  -0.01977    0.01590  -1.243 0.213752
## EducYears    0.17988    0.07914   2.273 0.023024 *
## ExpYears     0.16751    0.06600   2.538 0.011144 *
## ExpYears2   -0.14436    0.23585  -0.612 0.540489
## Age         -0.08234    0.02699  -3.050 0.002285 **
## NSmallChild -1.36250    0.38996  -3.494 0.000476 ***
## NBigChild   -0.02543    0.14172  -0.179 0.857592
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 277.26  on 200  degrees of freedom
## Residual deviance: 222.73  on 192  degrees of freedom
## AIC: 238.73
##
## Number of Fisher Scoring iterations: 4
```

(b) Now the fun begins.

Our goal is to approximate the posterior distribution of the 8-dim parameter vector β with a multivariate normal distribution

$$\beta \mid \boldsymbol{y}, \boldsymbol{X} \sim \mathcal{N}\big(\widetilde{\beta}, J_{\boldsymbol{y}}^{-1}(\widetilde{\beta})\big)$$

where $\widetilde{\beta}$ is the posterior mode and $J(\widetilde{\beta}) = \frac{\partial^2 \ ln \ p(\beta|\boldsymbol{y})}{\partial\beta\partial\beta^T} \mid_{\beta=\tilde{\beta}}$ is the observed Hessian evaluated at the posterior mode. Note that $\frac{\partial^2 \ lnp(\beta|\boldsymbol{y})}{\partial\beta\partial\beta^T}$ is an 8×8 matrix with second derivatives on the diagonal and cross-derivatives $\frac{\partial^2 \ lnp(\beta|\boldsymbol{y})}{\partial\beta_i\partial\beta_j}$ on the offdiagonal. It is actually not hard to compute this derivative by hand, but don???t worry, we will let the computer do it numerically for you. Now, both $\tilde{\beta}$ and $J(\tilde{\beta})$ are computed by the optim function in R. See my code ** https://github.com/ mattiasvillani/BayesLearnCourse/raw/master/Code/MainOptimizeSpam. ** **zip** where I have coded everything up for the spam prediction example (it also does probit regression, but that is not needed here). I want you to implement you own version of this. You can use my code as a template, but I want you to write your own file so that you understand every line of your code. Don???t just copy my code. Use the prior $\beta \sim \mathcal{N}(0, \tau^2 I)$, with $\tau = 10$. Your report should include your code as well as numerical values for $\tilde{\beta}$ and $J_{\boldsymbol{y}}^{-1}(\tilde{\beta})$ for *Womanwork* data. Compute an approximate 95% credible interval for the variable NSmallChild. Would you say that this feature is an important determinant of the probability that a women works?

```r
# given values for the prior
mu = 0
tau = 10  # Prior scaling factor such that Prior Covariance = (tau^2)*I - need tau to define sigma late

#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

# rename x columns
covNames <- names(WomenWork)[2:length(names(WomenWork))]
nPara <- ncol(X) #dim(x)[2]

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


# Defining the functions that returns the log posterior (Logistic and Probit models). Note that the fir

# this function must be the one that we optimize on, i.e. the regression coefficients.

LogPostLogistic <- function(betaVect,y,X,mu,Sigma){

  nPara <- length(betaVect);
  linPred <- X%*%betaVect;

  # evaluating the log-likelihood
  logLik <- sum( linPred*y -log(1 + exp(linPred)));
```

```
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer away from he

  # evaluating the prior
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), Sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)
}

# Different starting values. Ideally, any random starting value gives you the same optimum (i.e. optimu
initVal <- as.vector(rep(0,dim(X)[2]))
# Or a random starting vector: as.vector(rnorm(dim(X)[2]))
# Set as OLS estimate: as.vector(solve(crossprod(X,X))%*%t(X)%*%y); # Initial values by OLS

OptimResults<-optim(initVal,LogPostLogistic,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),control=list(fnscale=

OptimResults<-optim(initVal,
                    LogPostLogistic,
                    gr=NULL,
                    y,X,mu,Sigma,
                    method=c("BFGS"),
                    control=list(fnscale=-1),
                    hessian=TRUE) # hessian=TRUE -> output includes hessian matrix


# Printing the results to the screen
postMode <- OptimResults$par
postCov <- -solve(OptimResults$hessian) # Posterior covariance matrix is -inv(Hessian)
names(postMode) <- covNames # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(postCov)) # Computing approximate standard deviations.
names(approxPostStd) <- covNames # Naming the coefficient by covariates
print('The posterior mode is:')
```

```
## [1] "The posterior mode is:"
```

```
print(postMode)
```

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

```
print('The approximate posterior standard deviation is:')
```

```
## [1] "The approximate posterior standard deviation is:"
```

```
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
```

(c) Simulates from the predictive distribution

Write a function that simulates from the predictive distribution of the response variable in a logistic regression. Use your normal approximation from 2(b). Use that function to simulate and plot the predictive distribution for the *Work* variable for a 40 year old woman, with two children (3 and 9 years old), 8 years of education, 10 years of experience. and a husband with an income of 10. [Hint: the R package *mvtnorm* will again be handy. And remember my discussion on how Bayesian prediction can be done by simulation.]

Appendix

```
knitr::opts_chunk$set(echo = TRUE, out.width = "400px")
# packages we use in the lab
library(mvtnorm)
library(LaplacesDemon)
library(ggplot2)
library(reshape2)
library(gridExtra)
# clean environment
rm(list=ls())
# load the data
tempdata <- read.table("TempLinkoping.txt",header = TRUE, sep = "\t")
y        <- tempdata$temp
X        <- cbind(1, tempdata$time, tempdata$time**2)
n        <- nrow(X)

# given values
mu_0     <- c(-10, 100, -100)
omega_0  <- 0.01 * diag(3)
v_0      <- 4
sigma2_0 <- 1
# linear model - without beta
lmTemp = lm( temp ~ time + I(time^2), data = tempdata)
summary(lmTemp)

# plot the data
ggplot(tempdata, aes(x = time, y=temp)) +
  geom_point(color = "#00BFC4") +                          # default blue
  ylab("Temperature") + xlab("Time")
# create simulation
set.seed(123456)

# joint conjugate prior
N <- 40  # nr of simulations

# data frame to save prior coef data
prior <- matrix(ncol = 3, nrow = N)
for (i in 1:N) {

  # lecture 5 slide 7 - Inv - chi^2(v_0,sigma^2_0)
  var  <- LaplacesDemon::rinvchisq(1 ,v_0, sigma2_0)

  # solve(A)    Inverse of A where A is a square matrix
  beta <- MASS::mvrnorm(1, mu_0, var*solve(omega_0))

  prior[i,1:3] <- beta
}

data.1a <- as.data.frame(cbind(tempdata$time, X%*%t(prior)))

cnames <- c("x")
for (i in 1:N) {
```

```r
  cnames[1+i] <- paste0("pred.",i)
}
colnames(data.1a) <- cnames
data.1a <- melt(data.1a, id.vars = "x")

plot1a.ori <- ggplot(data.1a)+
  geom_line(aes(x = x, y = value, color = variable)) +
  geom_point(data = tempdata, aes(x = time, y = temp), color = "#00BFC4") +
  ggtitle("Original") +
  theme(legend.position = "none")
# modify the parameters
mu_0     <- c(-11, 104, -98)
omega_0  <- 0.01 * diag(3)
v_0      <- 4
sigma2_0 <- 0.02

prior <- matrix(ncol = 3, nrow = N)
for (i in 1:N) {
  var  <- LaplacesDemon::rinvchisq(1 ,v_0, sigma2_0)
  beta <- MASS::mvrnorm(1, mu_0, var*solve(omega_0))
  prior[i,1:3] <- beta
}
data.1a <- as.data.frame(cbind(tempdata$time, X%*%t(prior)))

cnames <- c("x")
for (i in 1:N) {
  cnames[1+i] <- paste0("pred.",i)
}
colnames(data.1a) <- cnames
data.1a <- melt(data.1a, id.vars = "x")

plot1a.mdf <- ggplot(data.1a)+
  geom_line(aes(x = x, y = value, color = variable)) +
  geom_point(data = tempdata, aes(x = time, y = temp), color = "#00BFC4") +
  ggtitle("Modified") +
  theme(legend.position = "none")

plot(arrangeGrob(plot1a.ori, plot1a.mdf, nrow = 2))
# plot1a.ori
# plot1a.mdf
# posterior
k        <- 3 #  nr of regression coefficients
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
mu_n     <- solve(t(X) %*% X + omega_0) %*% (t(X) %*% X %*% beta_hat + omega_0 %*% mu_0)
omega_n    <- t(X) %*% X+omega_0
v_n      <- v_0 + n
sigma2_n <- (v_0 * sigma2_0 + (t(y) %*% y + t(mu_0) %*% omega_0 %*% mu_0 - t(mu_n) %*% omega_n %*% mu_n

# marginal posterior
###### Zijie idea:
# The arguments of marginal posterior of conjugate prior
# are guessed form marginal posterior of uniform prior (page 6)
######
```

```r
data_hist <- as.data.frame(
  mvtnorm::rmvt(n = 100, delta = mu_n, df = n-k,
                sigma = as.numeric(sigma2_n) * solve(omega_n))
  )
cnames <- c("beta0", "beta1", "beta2")
colnames(data_hist) <- cnames

f <- function(cname){
  ggplot(data_hist, aes_string(x=cname)) +
    geom_histogram(aes(y = ..density..),
                   colour = "black",
                   fill   = "white",
                   bins   = 30) +
    geom_density(alpha = .2, fill = "#FF6666")
}
plot(arrangeGrob(grobs = lapply(cnames, f)))
# overlay a curve for the posterior median
beta_median = matrixStats::colMedians(as.matrix(data_hist))
pred.1b = beta_median %*% t(X)

# credible intervals
    # each column represents all possible values at a fixed time
preds <- as.matrix(data_hist) %*% t(X)
pred_interval <- data.frame(nrow = n, nrow = 2)

colnames(pred_interval) <- c("lower","upper")
for(i in 1:n){
  data_t <- preds[,i]
  pred_interval[i,] <- quantile(data_t, probs = c(0.025,0.975))
}
data.1b <- cbind(tempdata, t(pred.1b), pred_interval)
ggplot(data.1b) +
  geom_point(aes(x = time, y = temp), color = "#00BFC4") +
  geom_line(aes(x = time, y = t(pred.1b)),color = "Blue",size = 1) +
  geom_line(aes(x = time, y = lower), color = "Red", linetype = "dotted", size = 1.5) +
  geom_line(aes(x = time, y = upper), color = "Red", linetype = "dotted", size = 1.5)
pred_highest <- c()
for(i in 1:366){
  pred_highest[i] <- max(preds[,i])
}
data.1c <- cbind(tempdata, t(pred.1b), pred_interval, pred_highest)
ggplot(data.1c) +
  geom_point(aes(x = time, y = temp), color = "#00BFC4") +
  geom_line(aes(x = time, y = t(pred.1b)),color = "Blue",size = 1) +
  geom_line(aes(x = time, y = lower), color = "Red", linetype = "dotted", size = 1.5) +
  geom_line(aes(x = time, y = upper), color = "Red", linetype = "dotted", size = 1.5) +
  geom_line(aes(x = time,y = pred_highest), color = "black", linetype = "solid", size = 1)
# clean up environment
rm(list=ls())
knitr::include_graphics("lab2_table_ex2.png")
# load the data & define y (response) and x 8 dimensional vector of the features
WomenWork = read.table("WomenWork.dat", header = TRUE)
y = WomenWork$Work
```

```r
X = WomenWork[,2:9]
X = as.matrix(X)
n = nrow(WomenWork) # nr of observations
# logistic regression
glmModel <- glm(Work ~ 0 + ., data = WomenWork, family = binomial)
summary(glmModel)

# given values for the prior
mu = 0
tau = 10  # Prior scaling factor such that Prior Covariance = (tau^2)*I - need tau to define sigma late

#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

# rename x columns
covNames <- names(WomenWork)[2:length(names(WomenWork))]
nPara <- ncol(X) #dim(x)[2]

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


# Defining the functions that returns the log posterior (Logistic and Probit models). Note that the fir

# this function must be the one that we optimize on, i.e. the regression coefficients.

LogPostLogistic <- function(betaVect,y,X,mu,Sigma){

  nPara <- length(betaVect);
  linPred <- X%*%betaVect;

  # evaluating the log-likelihood
  logLik <- sum( linPred*y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer away from he

  # evaluating the prior
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), Sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)
}

# Different starting values. Ideally, any random starting value gives you the same optimum (i.e. optimu
initVal <- as.vector(rep(0,dim(X)[2]))
# Or a random starting vector: as.vector(rnorm(dim(X)[2]))
# Set as OLS estimate: as.vector(solve(crossprod(X,X))%*%t(X)%*%y); # Initial values by OLS

OptimResults<-optim(initVal,LogPostLogistic,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),control=list(fnscale=


OptimResults<-optim(initVal,
                    LogPostLogistic,
```

```
                    gr=NULL,
                    y,X,mu,Sigma,
                    method=c("BFGS"),
                    control=list(fnscale=-1),
                    hessian=TRUE) # hessian=TRUE -> output includes hessian matrix


# Printing the results to the screen
postMode <- OptimResults$par
postCov <- -solve(OptimResults$hessian) # Posterior covariance matrix is -inv(Hessian)
names(postMode) <- covNames # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(postCov)) # Computing approximate standard deviations.
names(approxPostStd) <- covNames # Naming the coefficient by covariates
print('The posterior mode is:')
print(postMode)
print('The approximate posterior standard deviation is:')
print(approxPostStd)


##############################################################################################
# Author: Mattias Villani, Link??ping University.
#         E-mail: mattias.villani@liu.se
#         web: http://mattiasvillani.com
# Script to illustrate numerical maximization of the Logistic or Probit regression
##############################################################################################

############   BEGIN USER INPUTS   #################
Probit <- 0           # If Probit <-0, then logistic model is used.
chooseCov <- c(1:16)  # Here we choose which covariates to include in the model
tau <- 10000;         # Prior scaling factor such that Prior Covariance = (tau^2)*I
############     END USER INPUT    #################


#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("SpamReduced.dat",header=TRUE)  # Spam data from Hastie et al.
y <- as.vector(Data[,1]); # Data from the read.table function is a data frame. Let's convert y and X to
X <- as.matrix(Data[,2:17]);
covNames <- names(Data)[2:length(names(Data))];
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);

# Defining the functions that returns the log posterior (Logistic and Probit models). Note that the fir

# this function must be the one that we optimize on, i.e. the regression coefficients.
```

```r
LogPostLogistic <- function(betaVect,y,X,mu,Sigma){

  nPara <- length(betaVect);
  linPred <- X%*%betaVect;

  # evaluating the log-likelihood
  logLik <- sum( linPred*y -log(1 + exp(linPred)));
  if (abs(logLik) == Inf) logLik = -20000; # Likelihood is not finite, stear the optimizer away from he

  # evaluating the prior
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), Sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)
}

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

  # The following is a more numerically stable evaluation of the log-likelihood in my slides:
  # logLik <- sum(y*log(pnorm(linPred)) + (1-y)*log(1-pnorm(linPred)) )
  logLik <- sum(y*pnorm(linPred, log.p = TRUE) + (1-y)*pnorm(linPred, log.p = TRUE, lower.tail = FALSE))

  # evaluating the prior
  logPrior <- dmvnorm(betaVect, matrix(0,nPara,1), Sigma, log=TRUE);

  # add the log prior and log-likelihood together to get log posterior
  return(logLik + logPrior)

}

# Calling the optimization routine Optim. Note the auxilliary arguments that are passed to the function
# Note how I pass all other arguments of the function logPost (i.e. all arguments except betaVect which
# The argument control is a list of options to the optimizer. Here I am telling the optimizer to multip
# Optim finds a minimum, and I want to find a maximum. By reversing the sign of logPost I can use Optim

# Different starting values. Ideally, any random starting value gives you the same optimum (i.e. optimu
initVal <- as.vector(rep(0,dim(X)[2]));
# Or a random starting vector: as.vector(rnorm(dim(X)[2]))
# Set as OLS estimate: as.vector(solve(crossprod(X,X))%*%t(X)%*%y); # Initial values by OLS

if (Probit==1){
  logPost = LogPostProbit;
} else{
  logPost = LogPostLogistic;
}

OptimResults<-optim(initVal,logPost,gr=NULL,y,X,mu,Sigma,method=c("BFGS"),control=list(fnscale=-1),hessi

# Printing the results to the screen
postMode <- OptimResults$par
postCov <- -solve(OptimResults$hessian) # Posterior covariance matrix is -inv(Hessian)
```

```r
names(postMode) <- covNames # Naming the coefficient by covariates
approxPostStd <- sqrt(diag(postCov)) # Computing approximate standard deviations.
names(approxPostStd) <- covNames # Naming the coefficient by covariates
print('The posterior mode is:')
print(postMode)
print('The approximate posterior standard deviation is:')
print(approxPostStd)

# Plotting some of the marginal posteriors
par(mfrow = c(2,2))
for (k in 1:4){
  betaGrid <- seq(0, postMode[k] + 4*approxPostStd[k], length = 1000)
  plot(betaGrid, dnorm(x = betaGrid, mean = postMode[k], sd = approxPostStd[k]), type = "l", lwd = 2, ma
}

# Plot a bivariate distribution for two beta coefficients - they are almost independent in this example
par1 <- 3
par2 <- 6
beta1Values <- seq(postMode[par1] - 3*approxPostStd[par1], postMode[par1] + 3*approxPostStd[par1], leng
beta2Values <- seq(postMode[par2] - 3*approxPostStd[par2], postMode[par2] + 3*approxPostStd[par2], leng
dens <- matrix(NA,length(beta1Values),length(beta2Values))
for (i in 1:length(beta1Values)){
  for (j in 1:length(beta1Values)){
    dens[i,j] <- dmvnorm(c(beta1Values[i],beta2Values[j]), postMode[c(par1,par2)], postCov[c(par1,par2)
  }
}
contour(beta1Values, beta2Values, dens)
postCov[par1,par2]/(sqrt(postCov[par1,par1])*sqrt(postCov[par2,par2]))
```