Lab4 Report

Ludvig Noring, Michael Sörsäter

May 17, 2017

Poisson regression

a)

Maximum likelihood estimation is performed with the generalized model using the family poisson. The result can be shown in the following table.

	Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
1	1.0724	-0.0205	-0.3945	0.4438	-0.0522	-0.2209	0.0707	-0.1207	-1.8941

Table 1: Coefficients for glm

The covarities that significantly influence the response are VerifyID, Sealed and MinBidShare.

b)

We implemented the log posterior method for the Poisson model. This produced similar results to what we got with glm. The result can be shown in the following table.

	Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
1	1.0698	-0.0205	-0.3930	0.4436	-0.0525	-0.2212	0.0707	-0.1202	-1.8920

Table 2: Coefficients for optim

c)

We implemented a general random walk-metropolis alhorithm and used it with our log-Poission-Posterior function. With 1000 draws the theta converges as shown in Figure 1.

	Const	PowerSeller	VerifyID	Sealed	Minblem	MajBlem	LargNeg	LogBook	MinBidShare
1	1.0671	-0.0154	-0.4038	0.4414	-0.0506	-0.2268	0.0700	-0.1172	-1.8949

Table 3: Coefficients for optim

d)

Using the MCMC from above the probability for k number of bids are presented in Figure 2. As shown in the Figure, the probability of 0 bids are 35.6 %.

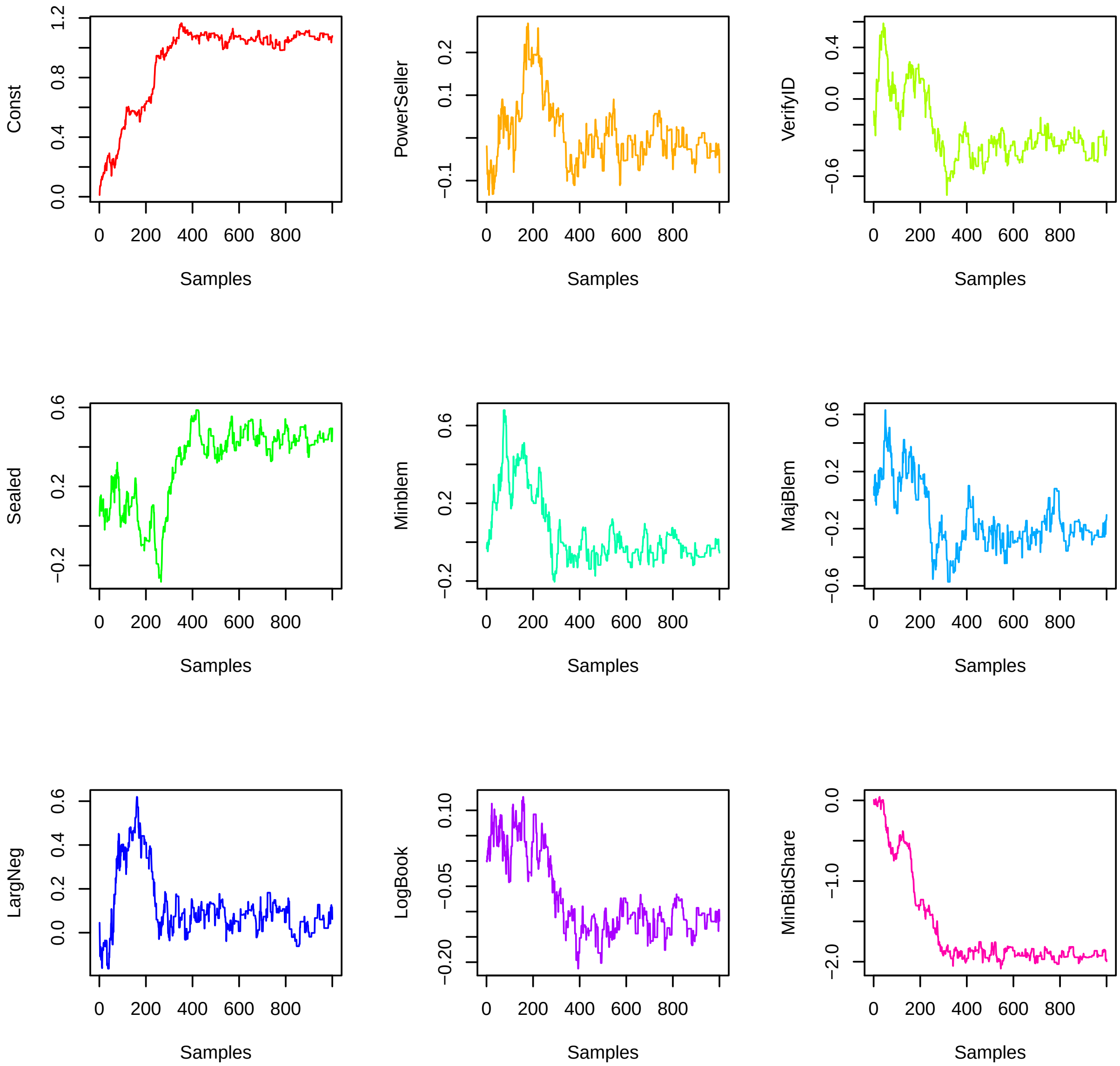

Figure 1: Convergence of θ

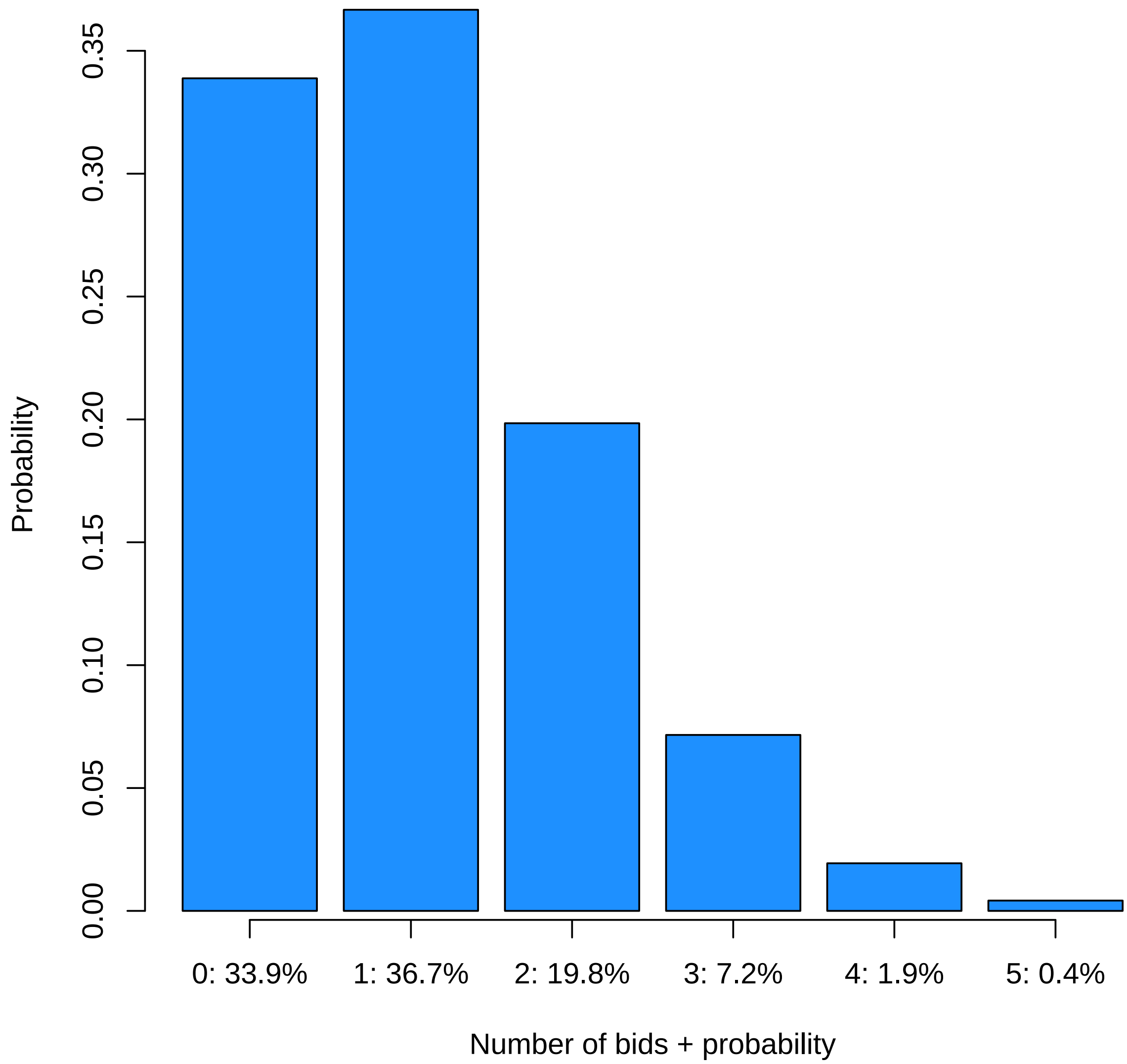

Figure 2: Bid Predictions

Appendix - Code for the assignment

```
require('MASS')
require('geoR')
require('mvtnorm')

ebay = read.table('eBayNumberOfBidderData.dat', header = TRUE)
X = as.matrix(ebay[,-1])
Y = ebay[,1]

nFeats = ncol(X)
nObs = nrow(X)

# a
glmModel = glm(nBids ~ 0 + ., data=ebay, family=poisson)
coefs = glmModel$coefficients
coefs

# b
sigma2_0 = 100 * solve(t(X) %*% X)
mu_0 = rep(0, nFeats)

beta_prior = mvrnorm(n=1, mu=mu_0, Sigma=sigma2_0)

# Calculate the posterior for the poisson model
logPostPoi = function(betas, y, X) {
  yPred = as.matrix(X) %*% betas
  lambda = exp(yPred)

  logLike = sum(log(exp(-lambda) * lambda^y / factorial(y)))
  logPrior = dmvnorm(betas, mu_0, sigma2_0, log=TRUE);

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

initValues = rep(0, nFeats)
optimResults = optim(initValues,
                     logPostPoi, y=Y, X=X,
                     method=c('BFGS'), control=list(fnscale=-1), hessian=TRUE)

postMode = optimResults$par
postCov = -solve(optimResults$hessian)
approxPostStd = sqrt(diag(postCov)/nFeats)
postMode

# c
metroDraw = function(logPostFunc, thetas, sigma, c_tilde, ...) {
  draws = 1000
  result_theta = matrix(0, draws, nFeats)
  mean_conv = matrix(0, draws, nFeats)
  thetas_c = thetas
  prob = c()
```

```r
  for(i in 1:draws){
    thetas_p = (mvrnorm(1, thetas_c, c_tilde*sigma))
    alpha = min(1, exp(logPostFunc(thetas_p, ...) - logPostFunc(thetas_c, ...)))
    prob = c(prob, alpha)
    # Update with probability alpha
    update = sample(c(TRUE, FALSE), 1, prob=c(alpha, 1-alpha))
    if(update)
      thetas_c = thetas_p

    result_theta[i,] = thetas_c

    if (i > 1)
      mean_conv[i,] = colMeans(result_theta[1:i,])
  }
  print(paste('Mean of alpha:', mean(prob)))
  return(result_theta)
 # return (mean_conv)
}

c_tilde = 0.6
thetas = metroDraw(logPostPoi, thetas=rep(0, length(beta_prior)), sigma=postCov,
                   c_tilde=c_tilde, y=Y, X=X)

pdf('plots/convergence.pdf')
  par(mfrow=c(3,3))
  for(i in 1:nFeats) {
    #plot(thetas[,i], type='l', col=rainbow(nFeats)[i], xlab='Samples', ylab=colnames(X)[i])
    traceplot(mcmc(thetas[,i]), col=rainbow(nFeats)[i], xlab='Samples', ylab=colnames(X)[i])
  }
dev.off()

# d
optTheta = thetas[nrow(thetas),]
femkrona_1972 = c(1, 1, 1, 1, 0, 0, 0, 1, 0.5)
names(femkrona_1972) = colnames(X)
lambda = exp(femkrona_1972%*%optTheta)

xGrid = 0:5
xAxis = paste(xGrid, ': ', round(100*dpois(xGrid, lambda), 1), '%', sep='')

pdf('plots/femkrona.pdf')
  barloc = barplot(dpois(xGrid, lambda), xaxt='n',
                   col='dodgerblue', xlab='Number of bids + probability', ylab='Probability')
  axis(side=1, at=barloc, labels = xAxis)
dev.off()
```