Econometrics, Exercise 2

Demirol, Engelen & Kuschnig

29 April 2018

Part I, Question 1

i)

First we read in the data from http://www.wiley.com/legacy/wileychi/koopbayesian/supp/HPRICE.TXT and store the create some variables and matrices from it.

```
require(readr)
data <- read_table2("exercise_2_data.txt")

n = nrow(data)
y = as.matrix(data[, 1])
x = as.matrix(data.frame(rep(1, n), data[, 2:5]))
k = 5
```

We continue our setup as ported from Koop's MATLAB scripts (with slight changes to accomodate the code to R).

```
v0 = 5 # i.e. about 1% of the data informations weight
b0 = c(0, 10, 5000, 10000, 10000)
s02 = 1 / 4.0e-8 # i.e. 1 / sigma^2
capv0 = diag(c(2.4, 6.0e-7, .15, .6, .6), k) # according to var = vs^2 / (v-2) * V
capv0inv = solve(capv0)
```

Then we source our ported script (see appendix) to do the posterior analysis and store the values needed for the table to be.

```
source("exercise_2_post.R")
koop_table = data.frame(rep(0, k))
koop_table$informative_prior_mean = round(b1, 2)
koop_table$informative_prior_sd = round(bsd, 2)
```

We repeat this with a non-informative prior obtained by setting v0 to 0 and updating the prior variance. Once again we store the values for the table and promptly show that we have successfully managed to generate the results from Table 3.1.

```
v0 = 0
capv0inv = 0 * diag(k)
source("exercise_2_post.R")
koop_table$non_informative_prior_mean = round(b1, 2)
koop_table$non_informative_prior_sd = round(bsd, 2)
```

Table 1: Prior and Posterior Means and Standard Deviations

informative_prior_mean	informative_prior_sd	non_informative_prior_mean	non_informative_prior_sd
-4035.05	3530.16	-4009.55	3593.16
5.43	0.37	5.43	0.37
2886.81	1184.93	2824.61	1211.45
16965.24	1708.02	17105.17	1729.65
7641.23	997.02	7634.90	1005.19

ii)

As is, the results in Table 3.1 are not overly sensitive to prior assumptions. That is because we have set v0 to 5 while having a total of 546 observations - leading to a relatively non-informative prior. "Loosely speaking, we are saying our prior information about h should have about 1% of the weight as the data information" (Koop 2003).

iii)

The expression of the marginal likelihood is as follows: $p(y_j|M_j) = c_j(\frac{|\overline{V}_j|}{|\underline{V}_j}})^{1/2}(\overline{v}_j\overline{s}_j^2)^{-\overline{v}_j/2}$.

We evaluate the impact of differing prior variances ($\underline{V}_\beta$ instead of $\underline{V}^{-1}$ as discussed by Koop (2003)) on the given grid and store the marginal likelihood in a vector.

```
mrg_lkl = vector("numeric")
v0 = 5

alpha = c(0.0001, 0.1, 0.5, 1, 2, 10)
for(j in 1:length(alpha)) {
  capv0 = diag(k) * alpha[j]
  capv0inv = solve(capv0)

  source("exercise_2_post.R")
  mrg_lkl[j] = lmarglik
}
```

Alpha	Likelihood
1e-04	-6158.3
1e-01	-6154.1
5e-01	-6157.1
1e+00	-6158.7
2e+00	-6160.4
1e+01	-6164.4

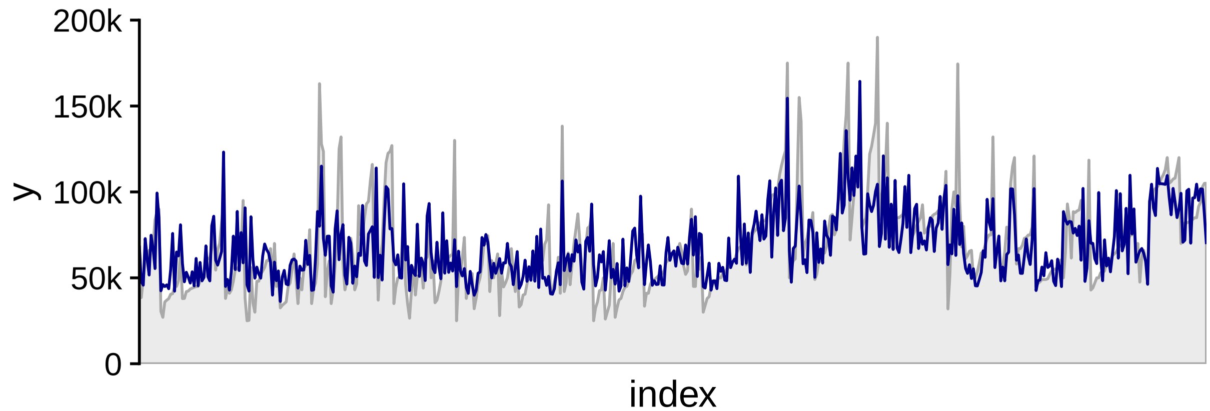
We obtain the highest likelihood for an alpha-value of 0.1 with a likelihood of -6154.11.

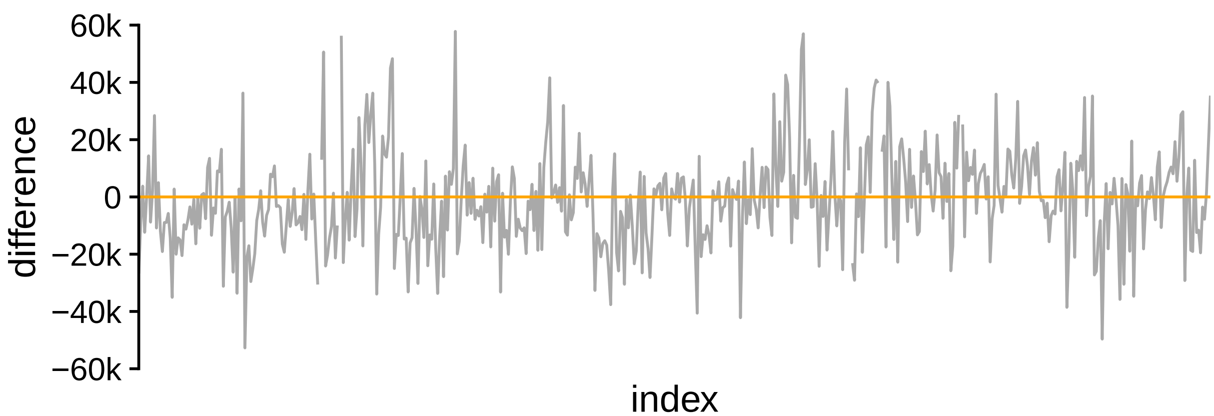
We go about our simulation scheme using (slightly modified) code from our lectures.

```
ntot <- 10000
beta.store <- matrix(NA, ntot, k)
sigma2.store <- matrix(NA, ntot, 1)
sigma2.draw <- s12
y.store <- matrix(NA, ntot, nrow(x))

for (i in 1:ntot) {
  sigma2.draw <- 1 / rgamma(1, v1 / 2, v1s12 / 2)

  beta.draw <- b1 + t(chol(sigma2.draw * capv1)) %*% rnorm(k)

  y.store[i, ] <- x %*% beta.draw + rnorm(nrow(x), 0, sqrt(sigma2.draw))

  beta.store[i, ] <- beta.draw
  sigma2.store[i, ] <- sigma2.draw
}
```

```r
beta.mean <- apply(beta.store, 2, mean)
beta.sd <- apply(beta.store, 2, sd)

y.mean <- apply(y.store, 2, mean)
```

Finally, we plot our results. Our simulations are visible in blue, while the true values are portrayed by the gray areas.





Appendix

The contents of exercise_2_post.R, as ported from Koop (2003).

```r
# OLS quantities
bols = solve(t(x) %*% x) %*% t(x) %*% y
s2 = t(y - x %*% bols) %*% (y - x %*% bols) / (n - k); s2 = s2[1]
bolscov = s2 * solve(t(x) %*% x)
bolssd = vector("numeric", k)
for(i in 1:k) {
  bolssd[i] = sqrt(bolscov[i, i])
}
v = n - k

# Posterior hyperparameters for Normal-Gamma
xsquare = t(x) %*% x
v1 = v0 + n
```

```r
capv1inv = capv0inv + xsquare
capv1 = solve(capv1inv)
b1 = capv1 %*% (capv0inv %*% b0 + xsquare %*% bols)
if(det(capv0inv) > 0) {
  v1s12 = v0 %*% s02 + v %*% s2 + t(bols - b0) %*% solve(capv0 + solve(xsquare)) %*% (bols - b0)
} else {
  v1s12 = v0 %*% s02 + v %*% s2
}
v1s12 = v1s12[1]
s12 = v1s12 / v1

bcov = capv1 * v1s12 / (v1 - 2)
bsd = vector("numeric", k)
for(i in 1:k) {
  bsd[i] = sqrt(bcov[i, i])
}

# # Posterior probability of each element of beta being positive + 95% & 99% HPDIs for each
# probpos = vector("numeric", k)
# bhpdi95 = matrix("numeric", k, 2)
# bhpdi99 = matrix("numeric", k, 2)
#
# invcdf95 = qt(.975, df = v1)
# invcdf99 = qt(.995, df = v1)
#
# for(i in 1:k) {
#   tnorm = -b1[i] / sqrt(s12 * capv1[i, i])
#   probpos[i] = 1 - pt(tnorm, v1)
#   bhpdi95[i, 1] = b1[i] - invcdf95 * sqrt(s12 * capv1[i, i])
#   bhpdi95[i, 2] = b1[i] + invcdf95 * sqrt(s12 * capv1[i, i])
#   bhpdi99[i, 1] = b1[i] - invcdf99 * sqrt(s12 * capv1[i, i])
#   bhpdi99[i, 2] = b1[i] + invcdf99 * sqrt(s12 * capv1[i, i])
# }

# Posterior mean and variance of error precision
hmean = 1 / s12
hvar = 2 / v1s12
hsd = sqrt(hvar)

# Predictive inference
if(k == 5) {
  xstar = t(c(1, 5000, 2, 2, 1))
  ystarm = xstar %*% b1; ystarm = ystarm[1]
  ystarcapv = (1 + xstar %*% capv1 %*% t(xstar)) * s12; ystarcapv = ystarcapv[1]
  ystarv = ystarcapv * v1 / (v1 - 2)
  ystarsd = sqrt(ystarv)
}

# Log of marginal likelihood if the prior is informative
if(det(capv0inv) != 0) {
  intcon = lgamma(.5 * v1) + .5 * v0 * log(v0 * s02) - lgamma(.5 * v0) - .5 * n * log(pi)
  lmarglik = intcon + .5 * log(det(capv1) / det(capv0)) - .5 * v1 * log(v1s12)
}
```