TH_Exam_RG

Raphael Gondo

12 de junho de 2018

The data and problem assigned

We have the data of monthly earnings, and several socio-demographical variables, as IQ, average weekly work hours, knowledge of the world, and others. The data is in the file: wage2-wooldridge.txt, from Blackburn and Neumark (1992).

We want to explain log(wages) with the some subset of covariates, and these combined, to understand their effect in these 935 individuals database.

Our job in this analysis is to, assuming Gaussianity of the residuals, fit the data with the following models: OLS, Ridge, LASSO, Bayesian ridge, Bayesian LASSO, and the Bayesian horseshoe. First, we do in-sample, that is using all data available, and fitting the models to the data, and compare them by adjusted R^2.

Second, we check the power of the models out-of-sample by using half of the observations, chosen randomly, to fit the models, and the other half to test the forecast power of the models, by RMSE.

The Models

The first model to test fit and forecasting power is the OLS. This model, in a frequentist approach, meaning that it has a true value of the parameters,is the best linear unbiased estimator (BLUE). The parameters minimize the quadratic errors of the estimates, and data.

The Ridge model also minimizes betas, but it penalizes the minimization problem with the sum of the squares of the coefficients. The LASSO has the same intuition, but it penalizes the coefficients with the absolute values. Notice that this can render more zeros to some coefficients, as it has kinks in some dimensions.

Actually, pushing coefficients to zero is important for this model we will discuss, as it has 58 potential independent variables.

While the LASSO has it explicit, other models do it with the combination of the priors and the likelihoods. The priors used for the other models are concentrated into zero, to push zero, as said, but they are sparse enough so that data can be accomodated in both tails of their distributions without the priors pushing them to zero.

Data import and packages

Let us first import the required to work data, and give names for its columns y and x (as x is usually used for the state). Also, let us call the required packages for this study.

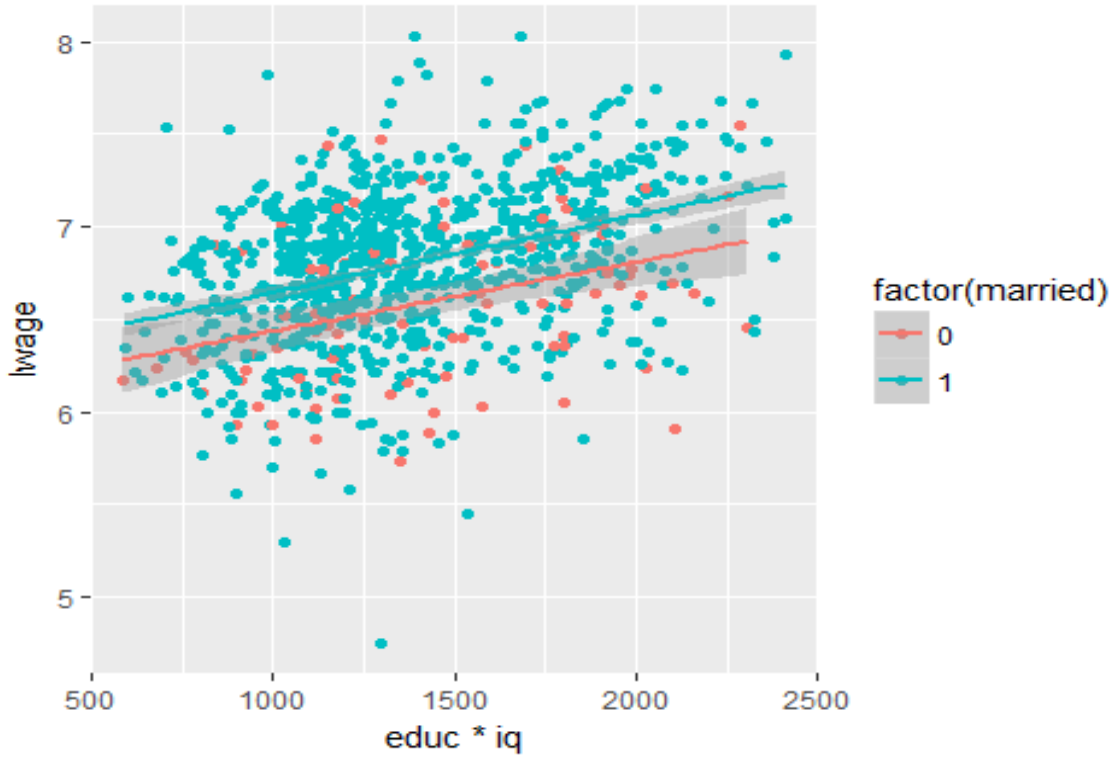
```
rm(list = ls())

library("ggplot2")
library("glmnet")

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16

library("bayeslm")
library("leaps")
library("grid")
library("gridExtra")

set.seed(1234)

mydata = read.table("http://hedibert.org/wp-content/uploads/2014/02/wage2
-wooldridge.txt",header=FALSE)
colnames(mydata) = c("wage","hours","iq","kww","educ","exper","tenure","a
ge","married",
                     "black","south","urban","sibs","brthord","meduc","fe
duc","lwage")
y = mydata[,ncol(mydata)]

X = model.matrix(lwage ~ kww + hours*exper*tenure*age + sibs +
                   (iq+educ+married + black + south + urban)^3-1,data=myd
ata)
n = nrow(X)
p = ncol(X)
```

Data visualization

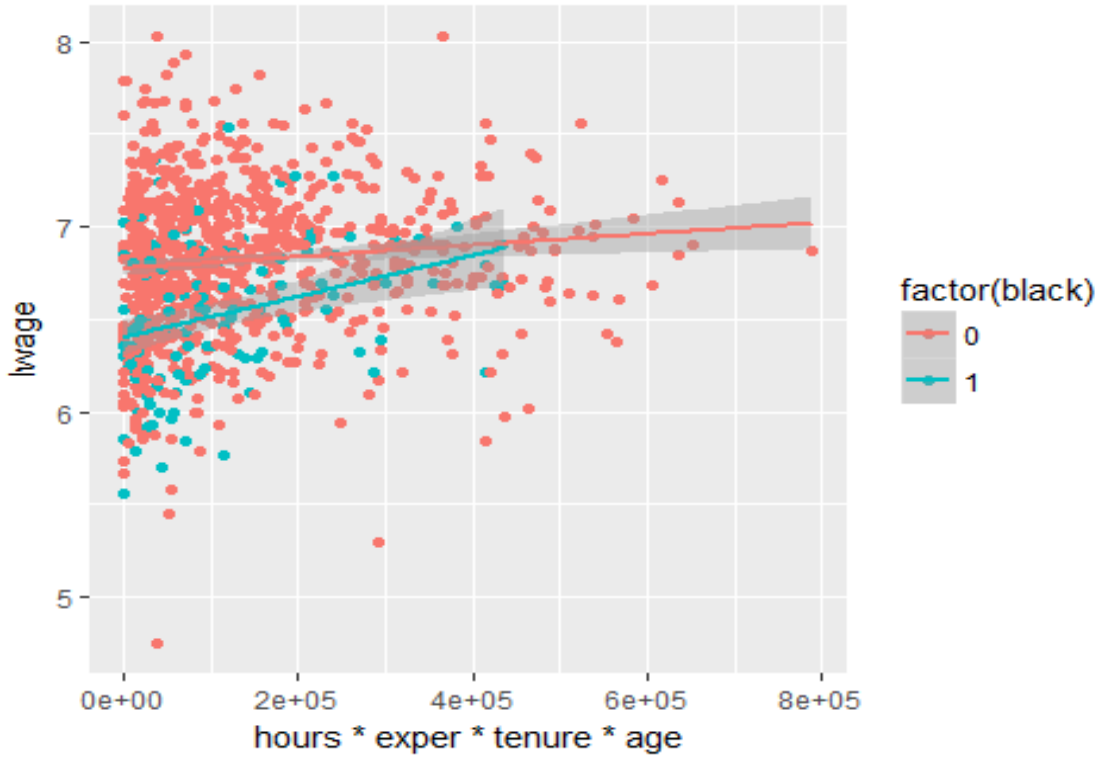
Let us first see what the data looks like and have some correlational intuitions of the relations we want to study.

```
ggplot(mydata, aes(x = hours*exper*tenure*age, y = lwage, col = factor(ma
rried))) +
  geom_point() +
  geom_smooth(method="lm")
```

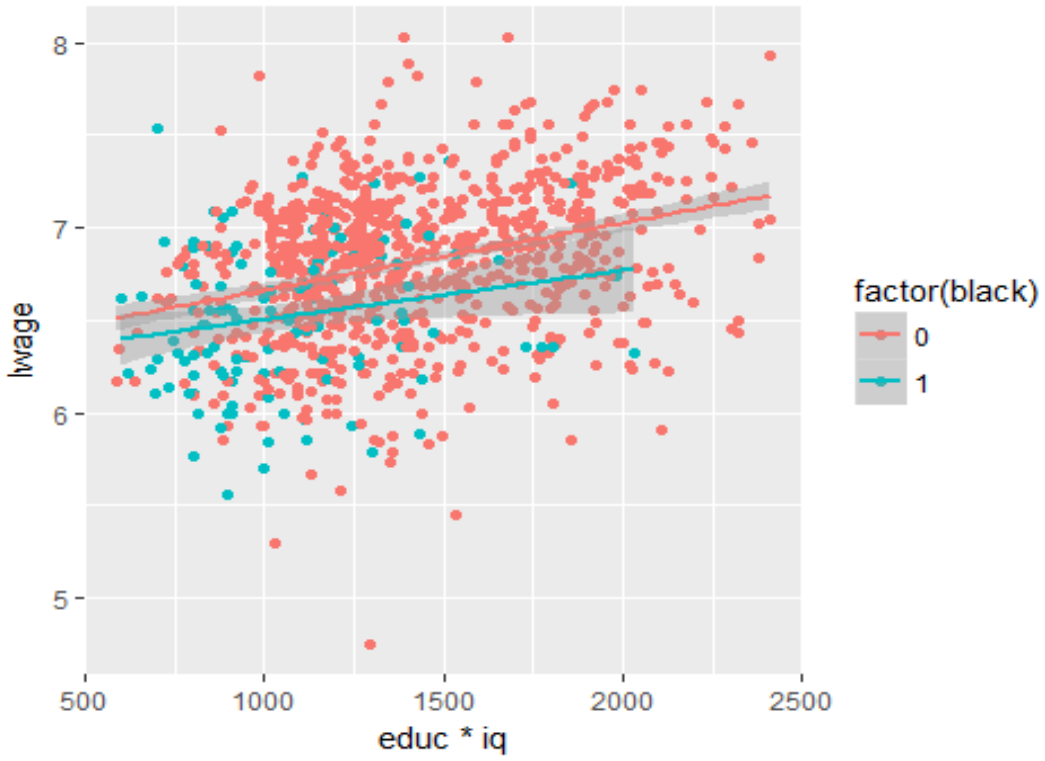

```
ggplot(mydata, aes(x = educ*iq, y = lwage, col = factor(married))) +
  geom_point() +
  geom_smooth(method="lm")
```



```
ggplot(mydata, aes(x = hours*exper*tenure*age, y = lwage, col = factor(bl
ack))) +
  geom_point() +
  geom_smooth(method="lm")
```




```
ggplot(mydata, aes(x = educ*iq, y = lwage, col = factor(black))) +
  geom_point() +
  geom_smooth(method="lm")
```



Keep in mind that they are all correlations, none claims causality. Given that, they tell some correlational stories (among several possible stories with other variables in the x axis).

The first one is about marriage. It seems that it pays off to be married, that is, there is a wage gap between the married and the unmarried individuals, favorably to the married ones; but, as hours worked, experience, tenure, and age increase (experiecens variables), this fades out. Maybe being married is a signal for commitment for whatever means. However, when one analyses the evolution of education and IQ (hard skill ones), the marriage gap does not close. Maybe, again, the signal is just socioemotional skills. But that is just suppositions.

The other story to be told is that black people seems to have a negative gap on wages too. It closes with the experience variables, but not with the hard skills. Maybe the employers have the same signalling for married and unmarried, showing some discrimination. But that is just too correlational to assert something.

Now, done with the fluffy discussions, let us dig into data.

FIRST PART

Let us estimate all our models, and compare the fit of them to data using adjusted R^2 statistic.

```
###### First Part ######

mean_y = mean(y)

SQtot_y = sum(sapply(y,function(x) (x - mean_y)^2))

##### ls
lm <- lm(y ~ X)
summary(lm)

##
## Call:
## lm(formula = y ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.94534 -0.21420  0.00206  0.21665  1.32529
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.074e+01  4.470e+00   2.402   0.0165 *
## Xkww                      4.172e-03  2.083e-03   2.002   0.0455 *
## Xhours                   -1.106e-01  8.660e-02  -1.277   0.2020
## Xexper                   -7.901e-02  3.612e-01  -0.219   0.8269
## Xtenure                  -8.056e-01  6.232e-01  -1.293   0.1965
## Xage                     -1.207e-01  1.176e-01  -1.026   0.3050
## Xsibs                    -1.780e-03  5.734e-03  -0.310   0.7563
## Xiq                       5.650e-04  2.097e-02   0.027   0.9785
## Xeduc                    -1.273e-01  1.636e-01  -0.778   0.4368
## Xmarried                  1.428e+00  1.914e+00   0.746   0.4559
## Xblack                    1.462e-01  2.253e+00   0.065   0.9483
## Xsouth                   -2.498e+00  1.466e+00  -1.704   0.0887 .
## Xurban                   -1.126e+00  1.596e+00  -0.706   0.4806
## Xhours:exper              2.681e-03  8.011e-03   0.335   0.7380
## Xhours:tenure             1.895e-02  1.403e-02   1.351   0.1772
## Xexper:tenure             5.112e-02  5.625e-02   0.909   0.3637
## Xhours:age                3.162e-03  2.542e-03   1.244   0.2138
## Xexper:age                2.464e-03  1.029e-02   0.239   0.8109
## Xtenure:age               2.418e-02  1.794e-02   1.348   0.1780
## Xiq:educ                  9.555e-04  1.457e-03   0.656   0.5121
## Xiq:married              -1.813e-02  1.832e-02  -0.990   0.3226
## Xiq:black                -1.362e-02  2.188e-02  -0.622   0.5339
## Xiq:south                 1.728e-02  1.383e-02   1.249   0.2118
## Xiq:urban                 6.585e-03  1.515e-02   0.435   0.6640
## Xeduc:married            -5.167e-02  1.439e-01  -0.359   0.7195
## Xeduc:black               2.133e-01  1.946e-01   1.096   0.2734
## Xeduc:south               1.942e-01  1.133e-01   1.714   0.0869 .
## Xeduc:urban               1.663e-01  1.185e-01   1.403   0.1610
## Xmarried:black           -1.287e+00  9.143e-01  -1.407   0.1597
## Xmarried:south            4.468e-01  7.121e-01   0.627   0.5305
```

```
## Xmarried:urban          -2.117e-01  7.653e-01  -0.277   0.7822
## Xblack:south             3.285e-01  8.076e-01   0.407   0.6843
## Xblack:urban            -6.207e-01  1.062e+00  -0.585   0.5589
## Xsouth:urban            -1.254e-01  5.477e-01  -0.229   0.8189
## Xhours:exper:tenure     -1.148e-03  1.283e-03  -0.895   0.3711
## Xhours:exper:age        -7.351e-05  2.265e-04  -0.325   0.7456
## Xhours:tenure:age       -5.597e-04  4.031e-04  -1.388   0.1654
## Xexper:tenure:age       -1.455e-03  1.591e-03  -0.915   0.3607
## Xiq:educ:married         9.067e-04  1.258e-03   0.721   0.4712
## Xiq:educ:black          -5.657e-04  1.620e-03  -0.349   0.7270
## Xiq:educ:south          -1.527e-03  9.434e-04  -1.619   0.1059
## Xiq:educ:urban          -1.137e-03  9.872e-04  -1.152   0.2495
## Xiq:married:black        1.897e-02  9.184e-03   2.065   0.0392 *
## Xiq:married:south       -1.663e-03  6.540e-03  -0.254   0.7993
## Xiq:married:urban        4.395e-03  7.423e-03   0.592   0.5540
## Xiq:black:south          8.595e-03  6.250e-03   1.375   0.1694
## Xiq:black:urban          9.672e-04  8.623e-03   0.112   0.9107
## Xiq:south:urban          1.294e-03  5.784e-03   0.224   0.8230
## Xeduc:married:black     -8.558e-02  7.575e-02  -1.130   0.2589
## Xeduc:married:south     -1.348e-03  4.924e-02  -0.027   0.9782
## Xeduc:married:urban     -1.595e-02  5.324e-02  -0.300   0.7646
## Xeduc:black:south       -1.047e-01  5.125e-02  -2.042   0.0414 *
## Xeduc:black:urban       -5.849e-02  7.565e-02  -0.773   0.4397
## Xeduc:south:urban        1.366e-02  3.418e-02   0.400   0.6895
## Xmarried:black:south    -1.543e-01  2.547e-01  -0.606   0.5448
## Xmarried:black:urban     9.541e-01  4.739e-01   2.013   0.0444 *
## Xmarried:south:urban    -2.950e-01  2.156e-01  -1.369   0.1714
## Xblack:south:urban       1.725e-01  4.860e-01   0.355   0.7227
## Xhours:exper:tenure:age  3.252e-05  3.621e-05   0.898   0.3694
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3576 on 876 degrees of freedom
## Multiple R-squared:  0.3238, Adjusted R-squared:  0.2791
## F-statistic: 7.233 on 58 and 876 DF,  p-value: < 2.2e-16

#computing adj_sqr
SQres_lm = sum(sapply(lm$residuals,function(x) x^2))
sqr_lm = 1-(SQres_lm/SQtot_y)
adjsqr_lm = 1-((1-sqr_lm)*(n-1)/(n-p-1))

##### best ls
# regsubsets(X,y,nbest=1,nvmax=58, force.in = NULL, force.out = NULL, int
ercept = TRUE,
#            method = c("exhaustive","backward","forward","seqrep"),reall
y.big = TRUE)

#parameters for Bayesian
burnin = 2000
M = 10000
```

```
thin = 1
Nsamples = burnin+M
block_vec = rep(1,ncol(X))

##### Ridge, alpha = 0
ridge <- glmnet(X, y, family = "gaussian", alpha = 0, nlambda = 100, lamb
da=NULL, lambda.min.ratio = 0.01,
                standardize = TRUE, intercept=TRUE)
adjsqr_ridge = 1-((1-max(ridge$dev.ratio))*(n-1)/(n-p-1))

## testing if the min lambda has the min standard errors
# max(ridge$dev.ratio)
# ridge$dev.ratio[100]

#LASSO, alpha = 1
lasso <- glmnet(X, y, family = "gaussian", alpha = 1, nlambda = 100, lamb
da=NULL, lambda.min.ratio = 0.01,
                standardize = TRUE, intercept=TRUE)
adjsqr_lasso = 1-((1-max(lasso$dev.ratio))*(n-1)/(n-p-1))

#Bayesian Ridge
bridge <- bayeslm(y,X,prior="ridge",block_vec=block_vec,N=Nsamples,icept=
TRUE,
        standardize=FALSE,singular=TRUE,
        verb=TRUE,thinning=thin,scale_sigma_prior=FALSE,cc=rep(1,p))

## ridge prior
## fixed running time 0.016757
## 1000
## 2000
## 3000
## 4000
## 5000
## 6000
## 7000
## 8000
## 9000
## 10000
## 11000
## sampling time 1.86172

# sqr_bridgep <- bridge$fitted.value

#computing adj_sqr for bridge
SQres_bridge = sum(sapply(bridge$residuals,function(x) x^2))
sqr_bridge = 1-SQres_bridge/SQtot_y
adjsqr_bridge = 1-((1-sqr_bridge)*(n-1)/(n-p-1))

#Bayesian LASSO
blasso <- bayeslm(y,X,prior="laplace",block_vec=block_vec,N=Nsamples,icep
```

```
t=TRUE,
                      standardize=FALSE,singular=TRUE,
                      verb=TRUE,thinning=thin,scale_sigma_prior=FALSE,cc=r
ep(1,p))
```

```
## laplace prior
## fixed running time 0.0156263
## 1000
## 2000
## 3000
## 4000
## 5000
## 6000
## 7000
## 8000
## 9000
## 10000
## 11000
## sampling time 1.53885
```

```
#computing adj_sqr for blasso
SQres_blasso = sum(sapply(blasso$residuals,function(x) x^2))
sqr_blasso = 1-SQres_blasso/SQtot_y
adjsqr_blasso = 1-((1-sqr_blasso)*(n-1)/(n-p-1))

#Bayesian Horseshoe
horseshoe <- bayeslm(y,X,prior="horseshoe",block_vec=block_vec,N=Nsamples
,icept=TRUE,
                     standardize=FALSE,singular=TRUE,
                     verb=TRUE,thinning=thin,scale_sigma_prior=FALSE,cc=rep(
1,p))
```

```
## horseshoe prior
## fixed running time 0.0156218
## 1000
## 2000
## 3000
## 4000
## 5000
## 6000
## 7000
## 8000
## 9000
## 10000
## 11000
## sampling time 2.51381
```

```
# horseshoep <- horseshoe$fitted.value

#computing adj_sqr for horseshoe
SQres_horseshoe = sum(sapply(horseshoe$residuals,function(x) x^2))
```

```
sqr_horseshoe = 1-SQres_horseshoe/SQtot_y
adjsqr_horseshoe = 1-((1-sqr_horseshoe)*(n-1)/(n-p-1))

# table of adj R sq
adjrsqr <- matrix(c(round(adjsqr_lm,3), round(adjsqr_ridge,3), round(adjs
qr_lasso,3), round(adjsqr_bridge,3), round(adjsqr_blasso,3), round(adjsqr
_horseshoe,3)),nrow=1,ncol=6)
colnames(adjrsqr) = c("lm", "ridge", "lasso", "bridge", "blasso", "horses
hoe")

grid.newpage()
grid.table(adjrsqr)
```

lm	ridge	lasso	bridge	blasso	horseshoe
0.279	0.185	0.249	0.249	0.255	0.239

The result in the table above is pretty much expected, as the OLS is, by contruction, the formula to minimize the squared errors. So it performed the best in-sample fit, against the linear models. But this has nothing to do with forecasting power. In the limit one can build a model that passes by each observation, so no error would be measured, but this would lead to overfitting, and zero predictive power.

The other models deal with fitting, but penalizes the model that fits too much. For example, the LASSO model penalizes high coefficients, so it can still be adaptable for out-of-sample observations.

In an extreme example, say that there are several high wages observations, that are actually outliers. An OLS model would try to reach those observations, but a LASSO model would also try to do this, but penalized for high coefficients.

SECOND PART

Now, the idea is to assess forecasting power of each model and compare them by the RMSE, reported in the Table below.

```
###### Second Part ######

# make predictions
#ridgep <- predict(ridge, X, se.fit = TRUE, dispersion = TRUE, na.action
= na.pass)

## Creating the fitting and the training subsamples

fit_size = round(n/2)
tsize = n - fit_size

fit_ind <- sample(n,replace=FALSE,size=fit_size)
fit <- mydata[fit_ind, ]
test <- mydata[-fit_ind, ]

y.fit = mydata[fit_ind,ncol(fit)]
X.fit  = model.matrix(lwage ~ kww + hours*exper*tenure*age + sibs + (iq+e
duc+married + black + south + urban)^3-1,data=fit)

y.test = mydata[-fit_ind,ncol(test)]
X.test = data.frame(model.matrix(lwage ~ kww + hours*exper*tenure*age + s
ibs + (iq+educ+married + black + south + urban)^3-1,data=test))

data <- as.data.frame(cbind(y, X))

interc <- rep(1,467)
newX <- cbind(interc,X[-fit_ind,])

# Testing fit of the models
lm_fit <- lm(y.fit ~ kww + hours*exper*tenure*age + sibs +
               (iq+educ+married + black + south + urban)^3-1,data=mydata[
fit_ind, ])

predlm <- predict.lm(lm_fit, X.test, se.fit = TRUE)

## Warning in predict.lm(lm_fit, X.test, se.fit = TRUE): prediction from
a
## rank-deficient fit may be misleading

rmselmi <- sqrt((y.test-predlm$fit)^2/tsize)
rmselm = sum(rmselmi)


ridge_fit <- glmnet(X.fit, y.fit, family = "gaussian", alpha = 0, nlambda
```

```
= 100, lambda=NULL, lambda.min.ratio = 0.01,
                standardize = TRUE, intercept=TRUE)
lambda_min<-ridge_fit$lambda[100]
newX <- model.matrix(~.-y.test,data=X.test)
newX = newX[,-1]
predridge <- predict.glmnet(ridge_fit, newX, se.fit = TRUE, s = lambda_mi
n)
rmseridgei <- sqrt((y.test-predridge)^2/tsize)
rmseridge = sum(rmseridgei)


lasso_fit <- glmnet(X.fit, y.fit, family = "gaussian", alpha = 1, nlambda
= 100, lambda=NULL, lambda.min.ratio = 0.01,
                standardize = TRUE, intercept=TRUE)
lambda_min<-lasso_fit$lambda[100]
predlasso <- predict.glmnet(lasso_fit, newX, se.fit = TRUE, s = lambda_mi
n)
rmselassoi <- sqrt((y.test-predlasso)^2/tsize)
rmselasso = sum(rmselassoi)


bridge_fit <- bayeslm(y.fit,X.fit,prior="ridge",block_vec=block_vec,N=Nsa
mples,icept=TRUE,
                  standardize=FALSE,singular=TRUE,
                  verb=TRUE,thinning=thin,scale_sigma_prior=FALSE,cc=rep(
1,p))
```

```
## ridge prior
## fixed running time 0.016789
## 1000
## 2000
## 3000
## 4000
## 5000
## 6000
## 7000
## 8000
## 9000
## 10000
## 11000
## sampling time 1.8679
```

```
lambda_min<-bridge_fit$lambda[100]

newX <- model.matrix(~.-y.test,data=X.test)
bridge.coefs <- as.matrix(apply(bridge_fit$beta,2,mean))
out.sample.pred <- newX%*%bridge.coefs
rmsebridgei <- sqrt((y.test-out.sample.pred)^2/tsize)
rmsebridge <- sum(rmsebridgei)
```

```
blasso_fit <- bayeslm(y.fit,X.fit,prior="laplace",block_vec=block_vec,N=N
samples,icept=TRUE,
                          standardize=FALSE,singular=TRUE,
                          verb=TRUE,thinning=thin,scale_sigma_prior=FALSE,
cc=rep(1,p))

## laplace prior
## fixed running time 0
## 1000
## 2000
## 3000
## 4000
## 5000
## 6000
## 7000
## 8000
## 9000
## 10000
## 11000
## sampling time 1.78474

blasso.coefs <- as.matrix(apply(blasso_fit$beta,2,mean))
out.sample.pred <- newX%*%blasso.coefs
rmseblassoi <- sqrt((y.test-out.sample.pred)^2/tsize)
rmseblasso <- sum(rmseblassoi)



horseshoe_fit <- bayeslm(y.fit,X.fit,prior="horseshoe",block_vec=block_ve
c,N=Nsamples,icept=TRUE,
                       standardize=FALSE,singular=TRUE,
                       verb=TRUE,thinning=thin,scale_sigma_prior=FALSE,cc=r
ep(1,p))

## horseshoe prior
## fixed running time 0
## 1000
## 2000
## 3000
## 4000
## 5000
## 6000
## 7000
## 8000
## 9000
## 10000
## 11000
## sampling time 3.07967

horseshoe.coefs <- as.matrix(apply(horseshoe_fit$beta,2,mean))
out.sample.pred <- newX%*%horseshoe.coefs
rmsehorseshoei <- sqrt((y.test-out.sample.pred)^2/tsize)
```

```r
rmsehorseshoe <- sum(rmsehorseshoei)


# Illustrating results
#rmselmi rmseridgei rmsebridgei rmsehorseshoei rmselassoi

cumrmsei <- data.frame(cbind(cumsum(rmselmi),cumsum(rmseridgei),cumsum(rm
selassoi),cumsum(rmsebridgei),cumsum(rmseblassoi),cumsum(rmsehorseshoei),
as.numeric(rownames(test))))
colnames(cumrmsei) = c("lm", "ridge", "lasso", "bridge", "blasso", "horse
shoe", "nobs")

analysisdata <- cbind(y,X,lm$fitted.values, bridge$fitted.value)
colnames(analysisdata)[60] = "lmfit"
colnames(analysisdata)[61] = "bridgefit"

# table of RMSE
RMSE <- matrix(c(round(rmselm,3), round(rmseridge,3), round(rmselasso,3),
round(rmsebridge,3), round(rmseblasso,3), round(rmsehorseshoe,3)), nrow=1
,ncol=6)
colnames(RMSE) = c("lm", "ridge", "lasso", "bridge", "blasso", "horseshoe
")

grid.newpage()
grid.table(RMSE)
```

lm	ridge	lasso	bridge	blasso	horseshoe
6.448	6.036	5.878	6.183	6.109	6.089

Now, exactly because of the kinks discussion in the beginning of this short analysis, and the previous discussion about overfitting, the results above are also expected. OLS here underperforms all models, and the LASSO is the best one for forecasting purposes with this dataset.

Forecast performance

For just another way to illustrate the forecast power of the models, I made a graph with the accumulated root mean squared error through the number of observations between the six models. The last table showed the accumulated RMSE until the last observation.

```
ggplot(cumrmsei, aes(nobs)) +
  geom_area(aes(y = lm, fill = "lm")) +
  geom_area(aes(y = bridge, fill = "bridge")) +
  geom_area(aes(y = blasso, fill = "blasso")) +
  geom_area(aes(y = horseshoe, fill = "horseshoe")) +
  geom_area(aes(y = ridge, fill = "ridge")) +
  geom_area(aes(y = lasso, fill = "lasso"))
```