Variable selection task

Marek Chadim

10-01-2024‘

Exercise 1: Within-time period prediction

Data Preprocessing

```
# Load the dataset for 1992-2002
d_92_02 <- read.csv("task5/growthdata92_02.csv")
d_92_02 <- d_92_02[, 3:ncol(d_92_02)] # Remove first two columns

# Split data into training (80%) and test (20%) sets
set.seed(42)
train_indices_92_02 <- sample(1:nrow(d_92_02), 0.8 * nrow(d_92_02), replace = FALSE)
train_data_92_02 <- d_92_02[train_indices_92_02, ]
test_data_92_02 <- d_92_02[-train_indices_92_02, ]
```

Naive Prediction

```
# Naive prediction: predict the mean growth for every country in the test set
mean_growth_92_02 <- mean(train_data_92_02$growth)
naive_pred_92_02 <- rep(mean_growth_92_02, nrow(test_data_92_02))

# Calculate RMSE for naive prediction
rmse_naive_92_02 <- sqrt(mean((naive_pred_92_02 - test_data_92_02$growth)^2))
```

Kitchen Sink Regression

```
# Kitchen sink regression: use all variables to predict growth
ks_model_92_02 <- lm(growth ~ ., data = train_data_92_02)
ks_pred_92_02 <- predict(ks_model_92_02, newdata = test_data_92_02)

# Calculate RMSE for kitchen sink regression
rmse_ks_92_02 <- sqrt(mean((ks_pred_92_02 - test_data_92_02$growth)^2))
```

Best Subset Selection

```
library(leaps)

# Perform best subset selection using 10-fold cross-validation
set.seed(42)
folds_92_02 <- sample(rep(1:10, length.out = nrow(train_data_92_02)))
```

```r
# Custom prediction function for regsubsets
predict.regsubsets <- function(object, newdata, id, ...) {
    form <- as.formula(object$call[[2]])
    mat <- model.matrix(form, newdata)
    coefi <- coef(object, id = id)
    xvars <- names(coefi)
    mat[, xvars] %*% coefi
}

# Cross-validation for best subset selection
cv_errors_92_02 <- matrix(NA, 10, 27)
for (j in 1:10) {
    best_fit_92_02 <- regsubsets(growth ~ ., data = train_data_92_02[folds_92_02 != j,], nvmax = 27)
    for (i in 1:27) {
        pred <- predict.regsubsets(best_fit_92_02, train_data_92_02[folds_92_02 == j,], id = i)
        cv_errors_92_02[j, i] <- mean((train_data_92_02$growth[folds_92_02 == j] - pred)^2)
    }
}

# Mean cross-validation errors and best model
mean_cv_errors_92_02 <- apply(cv_errors_92_02, 2, mean)
mean_cv_errors_92_02
```

```
##  [1] 0.0006109482 0.0008118162 0.0008268749 0.0006316388 0.0005470676
##  [6] 0.0005085600 0.0005368586 0.0005539064 0.0006311454 0.0006656833
## [11] 0.0007451374 0.0125001537 0.0023689864 0.0037544948 0.0009607569
## [16] 0.0008325033 0.0017335323 0.0012923233 0.0010922950 0.0008589996
## [21] 0.0012359840 0.0010427753 0.0008453640 0.0010161471 0.0008688895
## [26] 0.0008938316 0.0008846898
```

```r
best_subset_size_92_02 <- which.min(mean_cv_errors_92_02)

# Fit the best subset model and calculate test error
best_fit_final_92_02 <- regsubsets(growth ~ ., data = train_data_92_02, nvmax = best_subset_size_92_02)
best_pred_92_02 <- predict.regsubsets(best_fit_final_92_02, test_data_92_02, id = best_subset_size_92_02
rmse_best_subset_92_02 <- sqrt(mean((best_pred_92_02 - test_data_92_02$growth)^2))
```

Ridge Regression

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
# Prepare model matrix for ridge regression
x_train_92_02 <- model.matrix(growth ~ ., train_data_92_02)[, -1]
x_test_92_02 <- model.matrix(growth ~ ., test_data_92_02)[, -1]
y_train_92_02 <- train_data_92_02$growth
y_test_92_02 <- test_data_92_02$growth

# Perform ridge regression with cross-validation
ridge_mod_92_02 <- glmnet(x_train_92_02, y_train_92_02, alpha = 0)
cv_ridge_92_02 <- cv.glmnet(x_train_92_02, y_train_92_02, alpha = 0)
best_lambda_ridge_92_02 <- cv_ridge_92_02$lambda.min
```

```r
# Predict on test set and calculate RMSE
ridge_pred_92_02 <- predict(ridge_mod_92_02, s = best_lambda_ridge_92_02, newx = x_test_92_02)
rmse_ridge_92_02 <- sqrt(mean((ridge_pred_92_02 - y_test_92_02)^2))
```

Lasso Regression

```r
# Perform lasso regression with cross-validation
lasso_mod_92_02 <- glmnet(x_train_92_02, y_train_92_02, alpha = 1)
cv_lasso_92_02 <- cv.glmnet(x_train_92_02, y_train_92_02, alpha = 1)
best_lambda_lasso_92_02 <- cv_lasso_92_02$lambda.min

# Predict on test set and calculate RMSE
lasso_pred_92_02 <- predict(lasso_mod_92_02, s = best_lambda_lasso_92_02, newx = x_test_92_02)
rmse_lasso_92_02 <- sqrt(mean((lasso_pred_92_02 - y_test_92_02)^2))
```

Principal Component Regression

```r
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```r
# Perform PCR with 10-fold cross-validation
pcr_model_92_02 <- pcr(growth ~ ., data = train_data_92_02, scale = TRUE, validation = "CV")

# Use the best number of components and calculate test RMSE
ncomp_pcr_92_02 <- which.min(MSEP(pcr_model_92_02)$val[1, , ])
pcr_pred_92_02 <- predict(pcr_model_92_02, ncomp = ncomp_pcr_92_02, newdata = x_test_92_02)
rmse_pcr_92_02 <- sqrt(mean((pcr_pred_92_02 - y_test_92_02)^2))
```

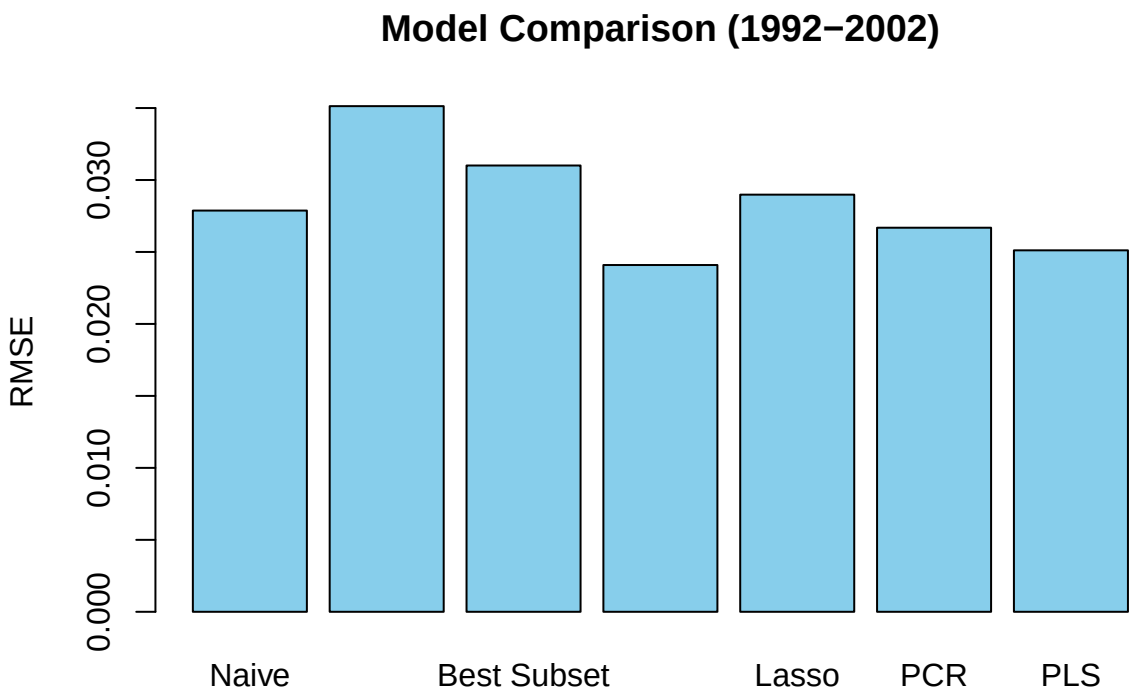
Partial Least Squares

```r
# Perform PLS with 10-fold cross-validation
pls_model_92_02 <- plsr(growth ~ ., data = train_data_92_02, scale = TRUE, validation = "CV")

# Use the best number of components and calculate test RMSE
ncomp_pls_92_02 <- which.min(MSEP(pls_model_92_02)$val[1, , ])
pls_pred_92_02 <- predict(pls_model_92_02, ncomp = ncomp_pls_92_02, newdata = x_test_92_02)
rmse_pls_92_02 <- sqrt(mean((pls_pred_92_02 - y_test_92_02)^2))
```

Model Comparison

```r
# Collect RMSE values for all models
models_92_02 <- c("Naive", "Kitchen Sink", "Best Subset", "Ridge", "Lasso", "PCR", "PLS")
rmse_values_92_02 <- c(rmse_naive_92_02, rmse_ks_92_02, rmse_best_subset_92_02, rmse_ridge_92_02, rmse_

# Create a dataframe for comparison
rmse_comparison_92_02 <- data.frame(Model = models_92_02, RMSE = rmse_values_92_02)
```

```r
# Plot RMSE comparison
barplot(rmse_comparison_92_02$RMSE, names.arg = rmse_comparison_92_02$Model,
        col = "skyblue", main = "Model Comparison (1992-2002)", ylab = "RMSE")
```




```r
rmse_comparison_92_02
```

```
##          Model       RMSE
## 1        Naive 0.02787608
## 2 Kitchen Sink 0.03513107
## 3  Best Subset 0.03100708
## 4        Ridge 0.02409284
## 5        Lasso 0.02898131
## 6          PCR 0.02668408
## 7          PLS 0.02511308
```

The best predictor out of was Ridge. Best Subset and Lasso did worse than the naive mean estimator. The Kitchen Sink did worst.

Exercise 2: Out-of-sample prediction

```r
# Load the 1992-2002 dataset
d_92_02 <- read.csv("task5/growthdata92_02.csv")
d_92_02 <- d_92_02[, 3:ncol(d_92_02)]

# Load the 2002-2011 dataset
```

```r
d_02_11 <- read.csv("task5/growthdata02_11.csv")
d_02_11 <- d_02_11[, 3:ncol(d_02_11)]

# Train using the full 1992-2002 dataset
x_92_02 <- model.matrix(growth ~ ., d_92_02)[, -1]
y_92_02 <- d_92_02$growth
x_02_11 <- model.matrix(growth ~ ., d_02_11)[, -1]
y_02_11 <- d_02_11$growth
```

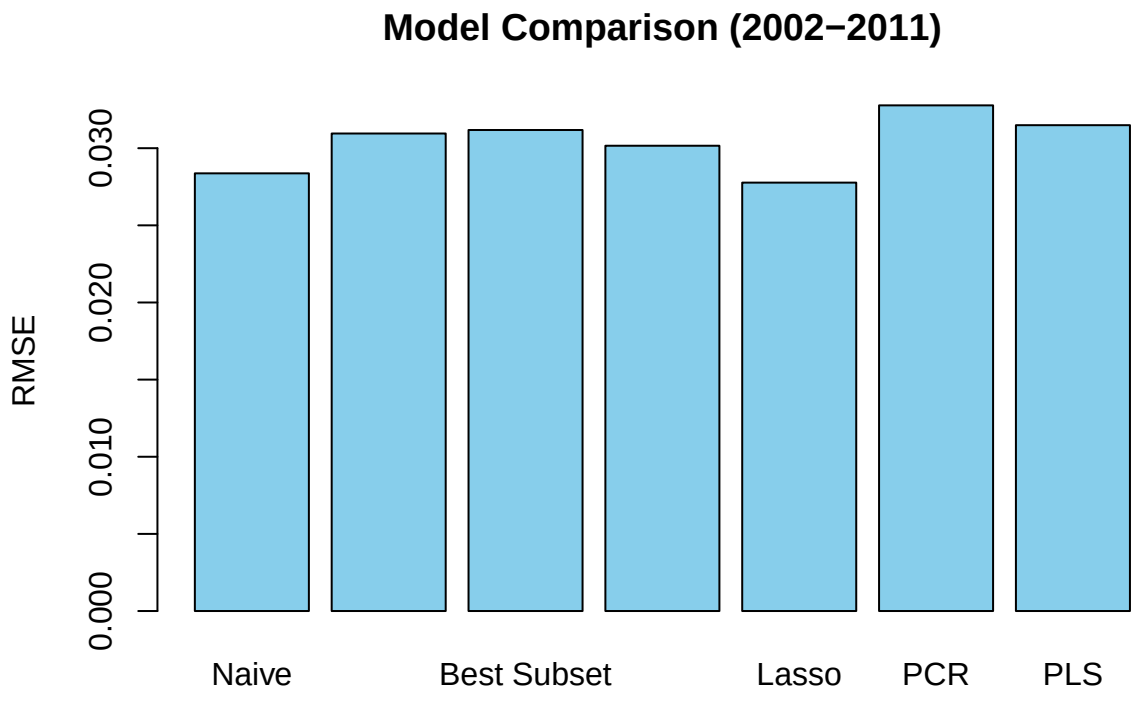
```r
# Naive prediction: predict the mean growth for the 2002-2011 dataset
mean_growth_92_02 <- mean(y_92_02)
naive_pred_02_11 <- rep(mean_growth_92_02, nrow(d_02_11))

# Calculate RMSE for naive prediction
rmse_naive_02_11 <- sqrt(mean((naive_pred_02_11 - y_02_11)^2))
rmse_naive_02_11
```

```
## [1] 0.02836891
```

```r
# Kitchen sink regression: use all variables to predict 2002-2011 growth
ks_model_92_02 <- lm(growth ~ ., data = d_92_02)
ks_pred_02_11 <- predict(ks_model_92_02, newdata = d_02_11)

# Calculate RMSE for kitchen sink regression
rmse_ks_02_11 <- sqrt(mean((ks_pred_02_11 - y_02_11)^2))

# Perform best subset selection on the full 1992-2002 dataset
best_fit_full_92_02 <- regsubsets(growth ~ ., data = d_92_02, nvmax = 27)

# Use the previously defined predict function
best_subset_size <- 7 # Assuming we use the previously selected size
best_subset_pred_02_11 <- predict.regsubsets(best_fit_full_92_02, d_02_11, id = best_subset_size)

# Calculate RMSE for best subset model
rmse_best_subset_02_11 <- sqrt(mean((best_subset_pred_02_11 - y_02_11)^2))

# Perform ridge regression with full 1992-2002 data
ridge_mod_full_92_02 <- glmnet(x_92_02, y_92_02, alpha = 0, lambda = 10^seq(10, -2, length = 100))
cv_ridge_full_92_02 <- cv.glmnet(x_92_02, y_92_02, alpha = 0)
best_lambda_ridge_full_92_02 <- cv_ridge_full_92_02$lambda.min

# Predict on the 2002-2011 dataset
ridge_pred_02_11 <- predict(ridge_mod_full_92_02, s = best_lambda_ridge_full_92_02, newx = x_02_11)
rmse_ridge_02_11 <- sqrt(mean((ridge_pred_02_11 - y_02_11)^2))

# Perform lasso regression with full 1992-2002 data
lasso_mod_full_92_02 <- glmnet(x_92_02, y_92_02, alpha = 1, lambda = 10^seq(10, -2, length = 100))
cv_lasso_full_92_02 <- cv.glmnet(x_92_02, y_92_02, alpha = 1)
best_lambda_lasso_full_92_02 <- cv_lasso_full_92_02$lambda.min

# Predict on the 2002-2011 dataset
lasso_pred_02_11 <- predict(lasso_mod_full_92_02, s = best_lambda_lasso_full_92_02, newx = x_02_11)
rmse_lasso_02_11 <- sqrt(mean((lasso_pred_02_11 - y_02_11)^2))
```

```r
# Perform PCR on full 1992-2002 dataset
pcr_model_full_92_02 <- pcr(growth ~ ., data = d_92_02, scale = TRUE, validation = "CV")
ncomp_pcr_full_92_02 <- which.min(MSEP(pcr_model_full_92_02)$val[1, , ])
pcr_pred_02_11 <- predict(pcr_model_full_92_02, ncomp = ncomp_pcr_full_92_02, newdata = x_02_11)

# Calculate RMSE for PCR
rmse_pcr_02_11 <- sqrt(mean((pcr_pred_02_11 - y_02_11)^2))

# Perform PLS on full 1992-2002 dataset
pls_model_full_92_02 <- plsr(growth ~ ., data = d_92_02, scale = TRUE, validation = "CV")
ncomp_pls_full_92_02 <- which.min(MSEP(pls_model_full_92_02)$val[1, , ])
pls_pred_02_11 <- predict(pls_model_full_92_02, ncomp = ncomp_pls_full_92_02, newdata = x_02_11)

# Calculate RMSE for PLS
rmse_pls_02_11 <- sqrt(mean((pls_pred_02_11 - y_02_11)^2))

# Collect RMSE values for all models
models_02_11 <- c("Naive", "Kitchen Sink", "Best Subset", "Ridge", "Lasso", "PCR", "PLS")
rmse_values_02_11 <- c(rmse_naive_02_11, rmse_ks_02_11, rmse_best_subset_02_11, rmse_ridge_02_11, rmse_

# Create a dataframe for comparison
rmse_comparison_02_11 <- data.frame(Model = models_02_11, RMSE = rmse_values_02_11)
```

```r
# Plot RMSE comparison
barplot(rmse_comparison_02_11$RMSE, names.arg = rmse_comparison_02_11$Model,
        col = "skyblue", main = "Model Comparison (2002-2011)", ylab = "RMSE")
```



```
rmse_comparison_02_11
```

```
##          Model       RMSE
## 1        Naive 0.02836891
## 2 Kitchen Sink 0.03095116
## 3  Best Subset 0.03117753
## 4        Ridge 0.03015930
## 5        Lasso 0.02776886
## 6          PCR 0.03277614
## 7          PLS 0.03149069
```

The fit is worse for all models other thanKitchen Sink as when evaluated within the same time-period.

Exercise 3: Testing for Changing Data Generating Process

```
# Load the 2002-2011 dataset
d_02_11 <- read.csv("task5/growthdata02_11.csv")

# Remove first two columns (assuming the same structure as the previous dataset)
d_02_11 <- d_02_11[, 3:ncol(d_02_11)]
sum(is.na(d_02_11))
```

```
## [1] 0
```

```
# Split data into training (80%) and test (20%) sets
set.seed(42)
train_indices_02_11 <- sample(1:nrow(d_02_11), 0.8 * nrow(d_02_11), replace = FALSE)
train_data_02_11 <- d_02_11[train_indices_02_11, ]
test_data_02_11 <- d_02_11[-train_indices_02_11, ]
```

```
# Naive prediction: predict the mean growth for every country in the test set (2002-2011 data)
mean_growth_02_11 <- mean(train_data_02_11$growth)
naive_pred_02_11 <- rep(mean_growth_02_11, nrow(test_data_02_11))

# Calculate RMSE for naive prediction
rmse_naive_02_11 <- sqrt(mean((naive_pred_02_11 - test_data_02_11$growth)^2))
rmse_naive_02_11
```

```
## [1] 0.0291664
```

```
# Kitchen sink regression: use all variables to predict growth in 2002-2011 data
ks_model_02_11 <- lm(growth ~ ., data = train_data_02_11)
ks_pred_02_11 <- predict(ks_model_02_11, newdata = test_data_02_11)

# Calculate RMSE for kitchen sink regression
rmse_ks_02_11 <- sqrt(mean((ks_pred_02_11 - test_data_02_11$growth)^2))
rmse_ks_02_11
```

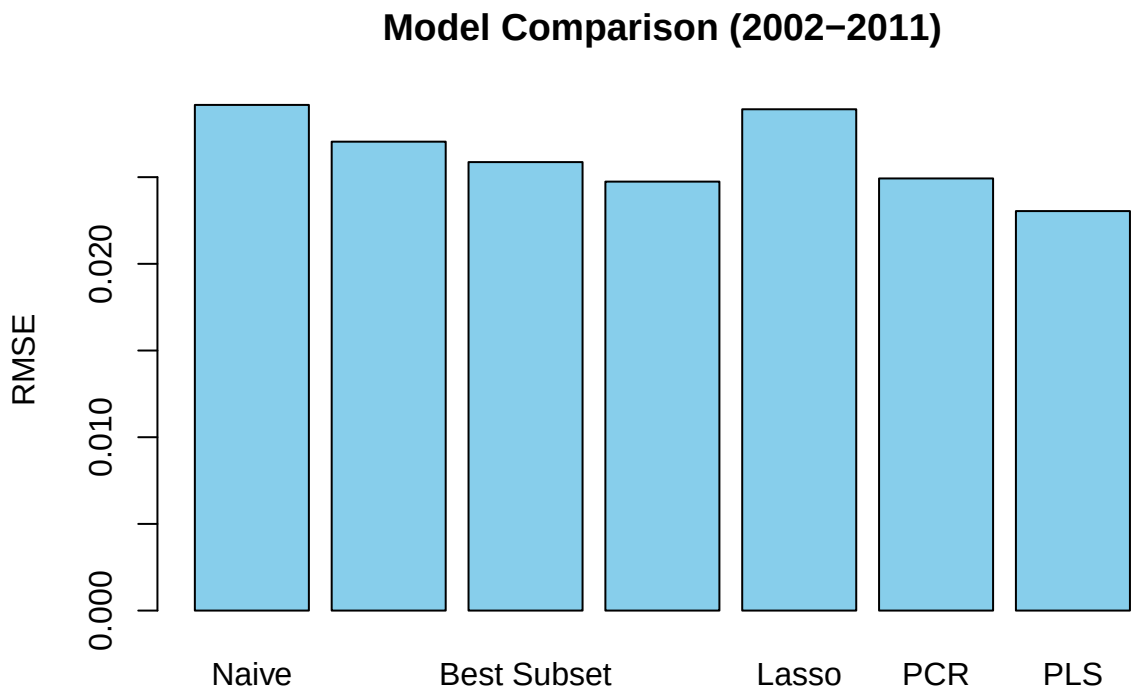
```
## [1] 0.02704583
```

```
# Cross-validation for best subset selection
folds_02_11 <- sample(rep(1:10, length.out = nrow(train_data_02_11)))
cv_errors_02_11 <- matrix(NA, 10, 27)
for (j in 1:10) {
    best_fit_02_11 <- regsubsets(growth ~ ., data = train_data_02_11[folds_02_11 != j,], nvmax = 27)
    for (i in 1:27) {
```

```
        pred <- predict.regsubsets(best_fit_02_11, train_data_02_11[folds_02_11 == j,], id = i)
        cv_errors_02_11[j, i] <- mean((train_data_02_11$growth[folds_02_11 == j] - pred)^2)
    }
}

# Mean cross-validation errors and best model
mean_cv_errors_02_11 <- apply(cv_errors_02_11, 2, mean)
best_subset_size_02_11 <- which.min(mean_cv_errors_02_11)

# Fit the best subset model and calculate test error
best_fit_final_02_11 <- regsubsets(growth ~ ., data = train_data_02_11, nvmax = best_subset_size_02_11)
best_pred_02_11 <- predict.regsubsets(best_fit_final_02_11, test_data_02_11, id = best_subset_size_02_1
rmse_best_subset_02_11 <- sqrt(mean((best_pred_02_11 - test_data_02_11$growth)^2))


# Prepare model matrix for ridge regression on 2002-2011 data
x_train_02_11 <- model.matrix(growth ~ ., train_data_02_11)[, -1]
x_test_02_11 <- model.matrix(growth ~ ., test_data_02_11)[, -1]
y_train_02_11 <- train_data_02_11$growth
y_test_02_11 <- test_data_02_11$growth
grid <- 10^seq(10, -2, length = 100)

# Perform ridge regression with cross-validation
ridge_mod_02_11 <- glmnet(x_train_02_11, y_train_02_11, alpha = 0, lambda = grid)
cv_ridge_02_11 <- cv.glmnet(x_train_02_11, y_train_02_11, alpha = 0)
best_lambda_ridge_02_11 <- cv_ridge_02_11$lambda.min

# Predict on the test set and calculate RMSE
ridge_pred_02_11 <- predict(ridge_mod_02_11, s = best_lambda_ridge_02_11, newx = x_test_02_11)
rmse_ridge_02_11 <- sqrt(mean((ridge_pred_02_11 - y_test_02_11)^2))

# Perform lasso regression with cross-validation on 2002-2011 data
lasso_mod_02_11 <- glmnet(x_train_02_11, y_train_02_11, alpha = 1, lambda = grid)
cv_lasso_02_11 <- cv.glmnet(x_train_02_11, y_train_02_11, alpha = 1)
best_lambda_lasso_02_11 <- cv_lasso_02_11$lambda.min

# Predict on the test set and calculate RMSE
lasso_pred_02_11 <- predict(lasso_mod_02_11, s = best_lambda_lasso_02_11, newx = x_test_02_11)
rmse_lasso_02_11 <- sqrt(mean((lasso_pred_02_11 - y_test_02_11)^2))

# Perform PCR with 10-fold cross-validation on 2002-2011 data
pcr_model_02_11 <- pcr(growth ~ ., data = train_data_02_11, scale = TRUE, validation = "CV")
ncomp_pcr_02_11 <- which.min(MSEP(pcr_model_02_11)$val[1, , ])
pcr_pred_02_11 <- predict(pcr_model_02_11, ncomp = ncomp_pcr_02_11, newdata = x_test_02_11)

# Calculate RMSE for PCR
rmse_pcr_02_11 <- sqrt(mean((pcr_pred_02_11 - y_test_02_11)^2))

# Perform PLS with 10-fold cross-validation on 2002-2011 data
pls_model_02_11 <- plsr(growth ~ ., data = train_data_02_11, scale = TRUE, validation = "CV")
ncomp_pls_02_11 <- which.min(MSEP(pls_model_02_11)$val[1, , ])
pls_pred_02_11 <- predict(pls_model_02_11, ncomp = ncomp_pls_02_11, newdata = x_test_02_11)
```

```r
# Calculate RMSE for PLS
rmse_pls_02_11 <- sqrt(mean((pls_pred_02_11 - y_test_02_11)^2))

# Collect RMSE values for all models
models_02_11 <- c("Naive", "Kitchen Sink", "Best Subset", "Ridge", "Lasso", "PCR", "PLS")
rmse_values_02_11 <- c(rmse_naive_02_11, rmse_ks_02_11, rmse_best_subset_02_11, rmse_ridge_02_11, rmse_
```

```r
# Create a dataframe for comparison
rmse_comparison_02_11 <- data.frame(Model = models_02_11, RMSE = rmse_values_02_11)

# Plot RMSE comparison
barplot(rmse_comparison_02_11$RMSE, names.arg = rmse_comparison_02_11$Model,
        col = "skyblue", main = "Model Comparison (2002-2011)", ylab = "RMSE")
```




```r
rmse_comparison_02_11
```

```
##          Model       RMSE
## 1        Naive 0.02916640
## 2 Kitchen Sink 0.02704583
## 3  Best Subset 0.02586858
## 4        Ridge 0.02474256
## 5        Lasso 0.02891534
## 6          PCR 0.02492482
## 7          PLS 0.02304222
```

Parameter Stability Comparison

The algorithm does not pick up the same variables.

```
# Extract coefficients from the best subset, ridge, and lasso models for comparison
coef(best_fit_full_92_02, best_subset_size_92_02)
```

```
##   (Intercept)          ln_y     inflation       fem_emp      inf_mort
##   0.020576543  -0.028487152  -0.006526219  -0.006646641  -0.013982603
##  presidential effectiveness
##  -0.005613240   0.017827982
```

```
coef(best_fit_final_02_11, best_subset_size_02_11)
```

```
##   (Intercept)           tfr parliamentary         voice
##   0.029184546  -0.013035558  -0.008849581  -0.012012983
```

```
# Compare lasso coefficients
lasso_coef_92_02 <- predict(lasso_mod_full_92_02, s = best_lambda_lasso_full_92_02, type = "coefficients
lasso_coef_02_11 <- predict(lasso_mod_02_11, s = best_lambda_lasso_02_11, type = "coefficients")[1:27, ]
lasso_coef_92_02[lasso_coef_92_02 != 0]
```

```
##   (Intercept)     inflation effectiveness           law
##  2.111670e-02 -4.433619e-03  9.782542e-05  2.321913e-05
```

```
lasso_coef_02_11[lasso_coef_02_11 != 0]
```

```
##  (Intercept)        voice
##  0.031759684 -0.000793918
```

If we do the analysis with the same variables the parameter estimates are not stable across the two sets.

```
lm(growth ~ ., data = d_92_02)$coefficients[1:10]
```

```
##   (Intercept)          ln_y            hc       gvmnt_c           gcf
##  0.0209040520 -0.0287351354  0.0006853459 -0.0011593425 -0.0005345856
##       ext_bal         trade     inflation       fem_emp       tot_emp
## -0.0030054766 -0.0001605588 -0.0071409326 -0.0059311448 -0.0006343544
```

```
lm(growth ~ ., data = d_02_11)$coefficients[1:10]
```

```
##   (Intercept)          ln_y            hc       gvmnt_c           gcf
##  0.0568728972 -0.0219677215  0.0020395952  0.0004345909  0.0100378485
##       ext_bal         trade     inflation       fem_emp       tot_emp
##  0.0081836077  0.0001398400  0.1011999101  0.0142927738 -0.0141780778
```

If anything, this teaches us that the results in Exercise 2 are more reliable as estimates of the out-of-sample error (and that the +1 SD lambda empirical rule might be beneficial, indeed, the most parsimonious model of a simple mean did best, which lasso chose in this case). This is related to the Lucas critique, which suggests that in predicting the effects of economic policy changes, we should not rely solely on historical data.