Term Paper

Comparing predictions of PCA, LASSO, and classic OLS in an experimental randomized setting

Norman Lothar Metzinger
August 21, 2022

Computational Statistics
University of Bonn

Student number: 3501090
Summer Term 2022

submitted to:
Prof. Dr. Lena Janys

Inhalt

Table of Figures

Introduction

Randomized controlled trials (RCTs) are said to be the "gold standard" of impact evaluation (Jones 2021), especially in the field of development economics. Their advantage lies in overcoming the selection bias and the sparse need for econometric methods. Overcoming selection bias improves the treatment effect. Using econometric models sparsely leads to fewer assumptions, which improves the interpretability of results. Unfortunately, problems in the implementation of RCTs lead to additional ex-post econometric work. Examining 6 high-quality RCTs, Banerjee, Karlan, and Zinman (2015) point out that external validity, modest R-squareds, and especially low statistical power remains a challenge.

A machine learning approach that could tackle the low statistical power problem is the Least Absolute Shrinkage and Selection Operator (LASSO). The LASSO is already used in the RCT context. For example, Caria et al. (2020) use LASSO to improve the precision and power of their estimation in an RCT in Ethiopia to further validate their results. Research from an RCT in South Africa (Carranza et al. 2021) reports using LASSO and Dhar et al. (2022) use LASSO to select their control variables to check for robustness of their Ordinary Least Squares (OLS) estimation in an Indian context. LASSO seems to be a useful tool in the recent literature of RCTs and there might be a need for models that perform variable selection.

Principal component analysis (PCA) is a machine learning technique that performs dimension reduction and is especially useful when there are many variables but few observations. So far, this technique is mostly used to generate indices in RCTs (see Romero et al. 2020, Brune et al. 2021) and there is scarce evidence of how PCA could be used for predictions in RCTs. In theory, PCA has the potential to be a tool in RCTs, especially in a low-dimensional environment.

In this paper, I want to contribute with machine learning techniques to the challenges of modest R-squareds and low statistical power in RCTs. The two techniques I want to use are PCA and LASSO. To have a comparison of how well these perform, I compare their predictions to OLS in training and test sets. OLS is the common approach used in RCTs and therefore an important property to compare to. To be able to compare the different properties of these techniques, I set up a simulation study that uses a Data Generating Process (DGP) that is found but abstracted from the RCT of Oster and Thornton (2011) in Nepal.

The study of Oster and Thornton (2011) has multiple interesting properties why I selected it for this paper: I) Low amount of observations[1], which can lead to overfitting of OLS but LASSO and PCA could perform better, II) No significant treatment effects, III) When performing predictions on the test set, where the number of covariates gets relatively close to the observations, OLS tends to overfit.

My paper is structured as follows: First, a theoretical introduction to the Methodology of LASSO and PCA. Second, the code, results, and comments on the Simulation Study that I conducted. Lastly, I conclude my results.

1. Methodology

LASSO and PCA follow the idea that certain variables do not have the same importance as others when predicting. LASSO performs shrinkage to estimate certain values towards or even zero. As some variables can be zero, LASSO selects variables, which allows for easier interpretation of models. In contrast, PCA recombines the variables into smaller becoming components but keeps all of them. Both methods should reduce variance and thus lead to an increase in statistical power. However, they come with the trade-off of higher bias.

[1] In the cross-sectional data of baseline and endline I use.

1.1 LASSO

To get a better idea how LASSO works I view the following formulation:

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|=\hat{\beta}_\lambda^L \tag{1}$$

Where n is the amount of observations and p the amount of variables. Note that the first part of formula 1 is the estimation of the least squares estimator and the second part is the ℓ_1 norm penalty, which combined estimates the LASSO Coefficient. One can see that LASSO is equal to Least squares when the tuning parameter $\lambda = 0$. But if $\lambda > 0$, it is crucial to know which λ is chosen as it defines the performance of the LASSO model. When $\lambda \to \infty$ all coefficients will be shrunken down to zero. In my application I use Cross-Validation to choose the optimal λ, it is also possible to use other approaches like *Akaike* or *Bayesian Information Criterion*. I discuss the selection of optimal λ further in Section 1.3.

To best understand how the shrinkage works, I view equation (2). The goal is to find the best fit for β given there is a constraint s. The bigger s, the less are the ℓ_1 penalty coefficient values shrunken down and the closer is LASSO to OLS. But as constraint s is set, one can see that some or all β_j can be shrunken down to 0 to fulfil this condition.

$$minimize_\beta \sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 s.t.\sum_{j=1}^{p}|\beta_j|\le s \tag{2}$$

Finally, it is important to understand how LASSO reduces variance compared to OLS as it is one key mechanism why I use it in my Simulation Study. Intuitively, as p is reduced, the amount of explaining variables is reduced, and thus, the overall variance decreases. Consider a case with *signal* (predicting variables) and *noise* variables, where the latter does not contribute to predicting the model. When the LASSO correctly specifies the signal variables, it outperforms OLS which has to attribute any variable some predictability. Thus, when p is close to n and a lot of p variables are noisy, LASSO should perform well with a low variance but a bias when the p shrunken variables have predictability.

1.2 PCA

The idea of PCA is not to shrink down variables but to transform them. Meaning that the X_i $(i = 1,\dots,p)$ variables are transformed to Z_i $(i = 1,\dots,M)$ and the estimators from β_i to θ_i with the difference that finally are less PC variables given, with $M < p$. The last expression is the reason why PCA is also called a dimension reduction technique. To transform these variables and estimates, PCA produces multiple components (Z_i) by projecting the data points with an orthogonal shift on the principal component line (*Eigenvector*) and then minimizes the distance to the origin. This is repeated by adding perpendicular lines until the number of components is equal to p or n (depending on which is smaller).

$$Z_m = \sum_{j=1}^{p}\varphi_{jm}X_j \tag{3}$$

φ_{jm} are the principal component loadings which are chosen in a way that the variance of each component is maximized. One can see observing formula (3) and (4) that PCA works quite similar as OLS.

$$\beta_j = \sum_{m=1}^{M}\theta_{jm}\varphi_{jm} \tag{4}$$

When performing Principal Component Regression (PCR) one does simply use the Z_i variables to perform a linear regression. An important difference to OLS is that PCR has lower variance with $M < p$ but higher bias.

Thus, leading to three important practical notes for PCA:

i) As we minimize distances between points the data must not be categorical as their distances to each other have no meaning. Also, scaling is advised.
ii) PCR performs well when the first components account for the most variation[2] and just a few PC are needed to construct a model. Otherwise when $M \rightarrow p$, PCR tends to perform as OLS
iii) The optimal number of components M can be selected by Cross-Validation, which is what I do in my Simulation Study.

1.3 Addition: Cross-Validation (CV)

There are two main things the CV has to solve. First, given the data, how well does a certain statistical model perform regarding test Root Mean Squared Error (RMSE). Secondly, given the statistical model for which value of the parameters of this model is the test RMSE the smallest. Both are important for my Simulation Study.

In this paper, I use k-fold Cross-Validation. What the CV does is separate the data in equal folds, where one is used for validation and the rest as training. For example, I use LASSO in a 10-fold CV. Then, LASSO would train on the k-1 folds training set and then predict the one validation set. Afterward, the RMSE is calculated for this case to be able to evaluate how good LASSO predicts the validation set. This approach is repeated k times. The CV is then given by:

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i \tag{5}$$

Instead of a model like LASSO, one could also use resample parameters like M of PCA or λ of LASSO.

2. Simulation Study

For the Simulation Study, I use the paper of Oster and Thornton (2011). With their cross-sectional data of the endline survey from *clean_outcomes_data_reshape_use.dta* I recreate the data. In this part, I explain my DGP and run the Simulation 50 times to compare RMSE and R-squared of LASSO, OLS, and PCA. I also provide some insights into the optimal selection of CV folds and the selection of tuning parameter λ.

2.1 Data Generating Process

The DGP consists of 13 variables that are reported in the cross-sectional endline survey of Oster and Thornton (2011). Note that I assume a normal distribution for the continuous variables and a binomial distribution for the dummy variables. The DGP is formally as follows:

$$\begin{aligned} Y = {} & \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} \\ & + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} T + \epsilon \end{aligned} \tag{6}$$

[2] When the minimization problem of the principal component produces small values then it accounts for *small* variations in the data.

Table 1: Summary Variables

Y	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Share of days school Missed (0/1)	Education Mother	Education Father	Total Income	School Score	School Grade	Having wage work (0/1)	Having Mensa (0/1)
	X_8	X_9	X_{10}	X_{11}	X_{12}	T	
	Have ever used pads (0/1)	Use rags (0/1)	Use pads (0/1)	Use rags and pads (0/1)	Father Hindu ethnicity (0/1)	Treated with menstruation cups (0/1)	

After setting up the DGP I encountered multiple issues:

1. The authors did not collect the possible important variable Age on endline.
2. The variable Having Mensa has a mean of 0.97 in the real Data, as I use the mean of the real data as the probability for the binomial distribution it leads in this small sample case to a vector of 1 and thus to collinearity. I remove it in the simulation, but an increase in observations should solve this issue.
3. In the original data the variables X_1, X_2, X_3 are just given in natural numbers (e.g. 0,1, 2, 3, …). Approaches to tackle these problems led to big distortions of the mean especially more positive values.
4. Especially $X_1 but\ also\ X_2$, have a truncated distribution with the most mass on 0 to 1 (max =14). I am aware that assuming a normal distribution is not ideal. As my test sample is small with n = 40 a non-normal distribution would automatically lead to a higher bias of the least squares.

2.2 Simulation

```
#Remark on running the code. I encountered that running the code in the r-m
arkdown environment does produce errors sometimes and sometimes not without
any changes made. Running normally on r does not have any issues. Multiple
runs in r-markdown and deleting the workspace also does it.

rm(list = ls())
setwd("E:/Uni Bonn/Semester 2/Computational Statistics/Project")
library(dplyr)

library(psych)

library(caret)

library(haven)

clean_outcomes_data_reshape_use <- read_dta("clean_outcomes_data_reshape_us
e.dta")

#Data Cleaning------------------
df_prework <- subset(clean_outcomes_data_reshape_use, survey != "_bl") #rem
oved the baseline survey
df_prework$a1_age <- NULL #at the endline survey they do not have any age d
ata collected
df_prework$survey <- NULL #unnecessary indicator variable for baseline and
endline survey
df_prework$c1_mens_yn <- NULL #removed or collinearity problem later
```

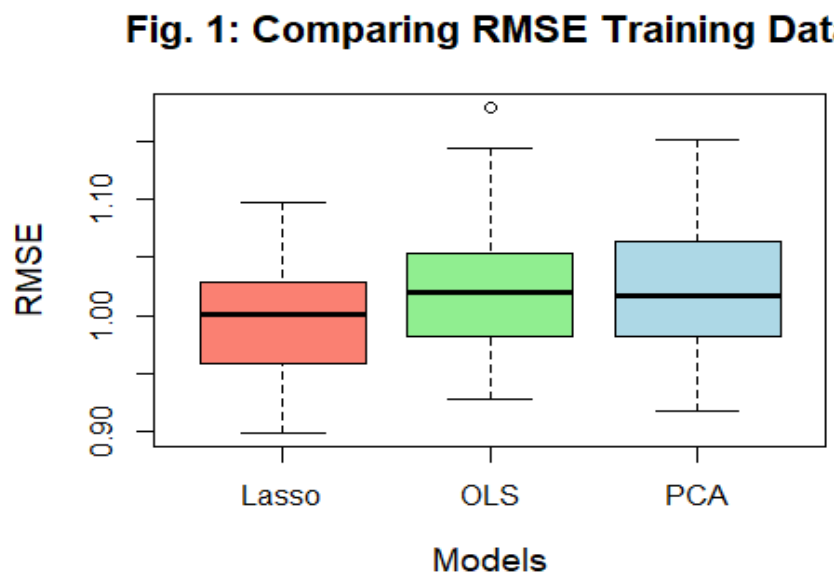
```r
#Data Cleaning-----------------
a <- lm(df_prework$c30_mens_missedschool_yn ~ ., data = df_prework)
#summary(a)
true_beta <- round(unname(a$coefficients),4)[-1] #intercept deleted, not tr
ue beta but rather a beta that generates new Y for every simulation run

true_y  <- df_prework$c30_mens_missedschool_yn #for the appendix
df_prework$c30_mens_missedschool_yn <- NULL #remove y
pred <- describe(df_prework, fast =TRUE, ranges =FALSE ) #table put in pape
r?
mu <- pred[,3]
sd <- pred[,4]
n <- 200
#Simulation containers----------
rep <- 50
cv_it <- 10 #cross validation iterations
#training insights
RMSE_container_lasso <- c()
RMSE_container_ols <- c()
RMSE_container_pcr <- c()
#test insights
RMSE_con_lasso_test <- c()
RMSE_con_ols_test<- c()
RMSE_con_pcr_test<- c()
R2_con_lasso_test<- c()
R2_con_ols_test<- c()
R2_con_pcr_test<- c()


#Simulation Loop----------

for (i in 1:rep){
  set.seed(i+50) #to be reproducible
  #DGP-----------
  mother_edu <- rnorm(n,mu[1],sd[1])
  father_edu <- rnorm(n,mu[2], sd[2])
  total_inc  <- rnorm(n,mu[3], sd[3])
  school_sc <- rnorm(n,mu[4], sd[4])
  school_grade <- rnorm(n,mu[5],sd[5])
  wage_work <-  rbinom(n,1,mu[6]) #prob = mean
  #mens_having <- rbinom(n,1,mu[7]) #prob = mean #remove see collinearity
  mens_evr_pads <- rbinom(n,1,mu[7]) #prob = mean
  mens_use_rags <- rbinom(n,1,mu[8]) #prob = mean
  mens_use_pads <- rbinom(n,1,mu[9]) #prob = mean
  mens_use_pads_rags <- rbinom(n,1,mu[10]) #prob = mean
  father_hindu <- rbinom(n,1,mu[11]) #prob = mean
  treatment <- rbinom(n,1,mu[12]) #prob = mean
  X <- cbind(mother_edu, father_edu, total_inc, school_sc, school_grade, wa
ge_work,
             mens_evr_pads, mens_use_rags, mens_use_pads, mens_use_pads_rag
s, father_hindu, treatment) #mens_having, is missing, put after wage work
  eps <- rnorm(n, 0, 1) #possible self given error term
  Y <-  X %*% true_beta + eps #way to reproduce Y that is predicted every i
teration
```

```
  df <- cbind(Y,X)
  #training and test data --------
  df <- as.data.frame(df)
  colnames(df)[1] <- "Y"
  partition <- createDataPartition(df$Y, p=.8, list =FALSE, times = 1) # te
st and training data
  training_data <- df[partition,]
  test_data <- df[-partition,]
  training_data <- as.data.frame(training_data)
  crossValid <- trainControl(method = "cv", number = cv_it, savePredictions
= "all")


  #Lassso--------------
  #lambdagrid
  lambda_grid <- 10^seq(5, -5, length = 500)
  lasso_mod <- train(Y ~ .,
                     data = training_data,
                     preProcess = c("center","scale"),
                     method = "glmnet",
                     tuneGrid = expand.grid(alpha=1,lambda = lambda_grid),
                     trControl=crossValid) # the fct. train provides an env
ironment for different tuning parameters and resampling methods and allows
to run different kinds of models. E.g. trainControl= takes the defined cros
s-validation from before which "folds" the training data and calculates the
optimal MSE for the training and validation sets. With preprocess, the data
is adjusted so that not the absolute size defines the predictions. tuneGrid
is the main part defining LASSO. Alpha = 1 sets the LASSO procedure. Optima
l lambda is chosen from the vector with the help of CV. The fct. train is u
sed similarly for OLS and PCR.


  prediction_lasso <- predict(lasso_mod, newdata = test_data) #predictions
of y
  modeltest_lasso <- postResample(prediction_lasso, test_data$Y)[-3]
  #OLS-------------
  ols_mod <- train(Y ~ .,
                   data = training_data,
                   preProcess = c("center","scale"), #not necessessary for
the model
                   method = "lm",
                   trControl = crossValid)
  #prediction_ols
  prediction_ols <- predict(ols_mod, newdata = test_data)
  modeltest_ols <- postResample(prediction_ols, test_data$Y)[-3]
  #pcr-----------------
  pcr_mod <- train(Y ~ .,
                   data = training_data,
                   preProcess = c("center","scale","pca"),
                   method = "lm", #also linear!
                   trControl = crossValid)
  prediction_pcr <- predict(pcr_mod, newdata = test_data)
  modeltest_pcr <- postResample(prediction_pcr, test_data$Y)[-3]
  models <- list(lasso_mod,ols_mod,pcr_mod)
  trainperform <- resamples(models) #shows all RMSEs of each model
```

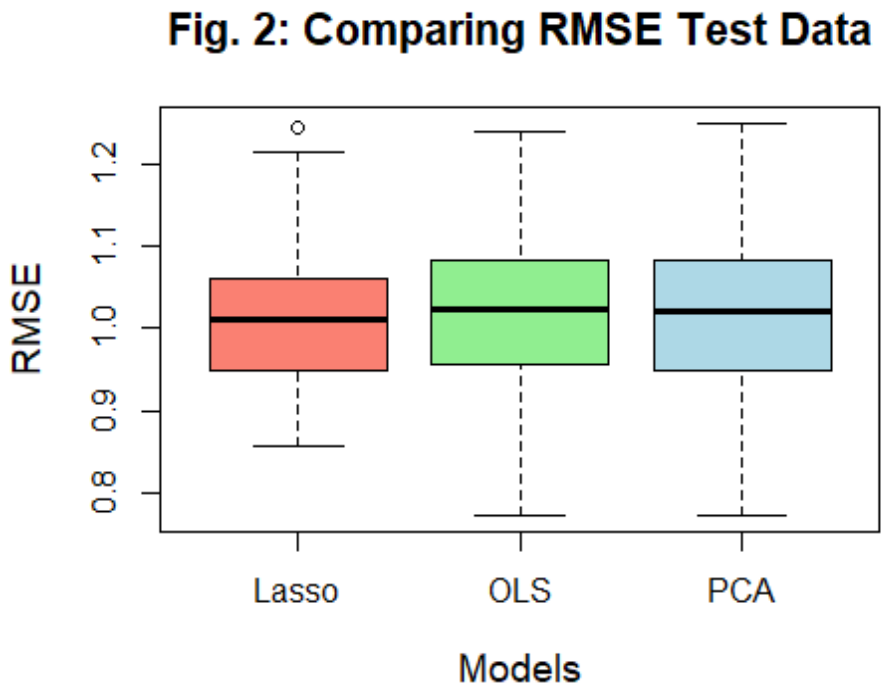
```
  #training insights
  #Mean MSE across variables
  RMSE_container_lasso[i] <- mean(trainperform$values$`Model1~RMSE`)
  RMSE_container_ols[i] <- mean(trainperform$values$`Model2~RMSE`)
  RMSE_container_pcr[i] <- mean(trainperform$values$`Model3~RMSE`)
  #test insights
  RMSE_con_lasso_test[i] <- mean(modeltest_lasso[1])
  RMSE_con_ols_test[i] <- mean(modeltest_ols[1])
  RMSE_con_pcr_test[i] <- mean(modeltest_pcr[1])
  R2_con_lasso_test[i] <- mean(modeltest_lasso[2])
  R2_con_ols_test[i] <- mean(modeltest_ols[2])
  R2_con_pcr_test[i] <- mean(modeltest_pcr[2])
} #end of simulation
```

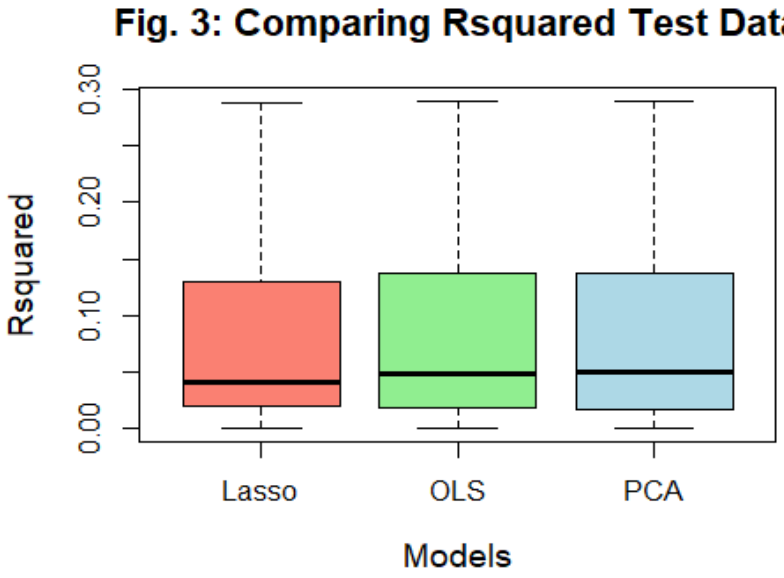
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trai
nInfo, :
## There were missing values in resampled performance measures.
```

```
# This Warning comes up every iteration of the loop running train(). The mo
del calculates the optimal lambda depending on the value of the RMSE, after
a certain value of lambda the RMSE becomes constant. As it suggests that th
e predictions are now constant for a part of the sample the R^2 cannot be c
alculated for these parts as there is no variance and thus is NA. This shou
ld not alter the results.

#PLOTS

RMSE_all_train <- cbind(RMSE_container_lasso, RMSE_container_ols, RMSE_cont
ainer_pcr)
#Note = 10 fold CV; Training RMSE
boxplot(RMSE_all_train, ylab = "RMSE", xlab = "Models", main ="Fig. 1: Comp
aring RMSE Training Data",
        cex.axis = 1, cex.lab = 1.2, cex.sub = 0.8, cex.main =1.4,
        col = (c("salmon","lightgreen","lightblue")), names = (c("Lasso","O
LS","PCA")))
```



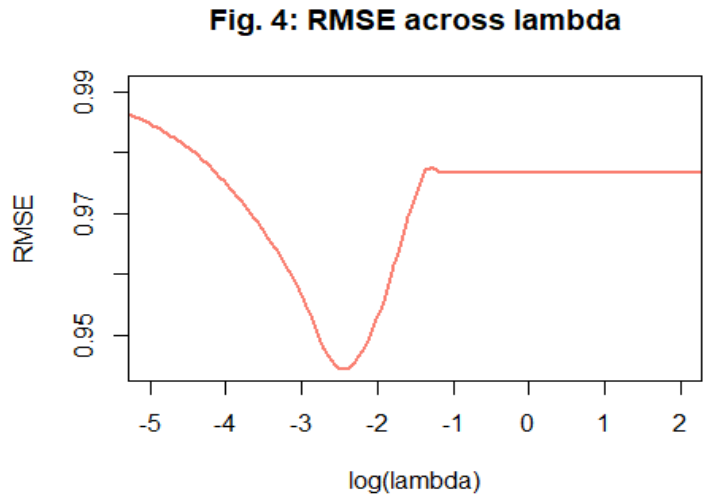
```
#Note = 10fold CV; Test RMSE
RMSE_all_test <- cbind(RMSE_con_lasso_test, RMSE_con_ols_test, RMSE_con_pcr
_test)
boxplot(RMSE_all_test, ylab = "RMSE", xlab = "Models", main ="Fig. 2: Compa
ring RMSE Test Data",
        cex.axis = 1, cex.lab = 1.2, cex.sub = 0.8, cex.main =1.4,
        col = (c("salmon","lightgreen","lightblue")), names = (c("Lasso","O
LS","PCA")))

#Note = 10fold CV; Test Rsquared
R2_all_test <- cbind(R2_con_lasso_test, R2_con_ols_test, R2_con_pcr_test)
boxplot(R2_all_test, ylab = "Rsquared", xlab = "Models", main ="Fig. 3: Com
paring Rsquared Test Data",
        cex.axis = 1, cex.lab = 1.2, cex.sub = 0.8, cex.main =1.4,
        col = (c("salmon","lightgreen","lightblue")), names = (c("Lasso","O
LS","PCA")))
```





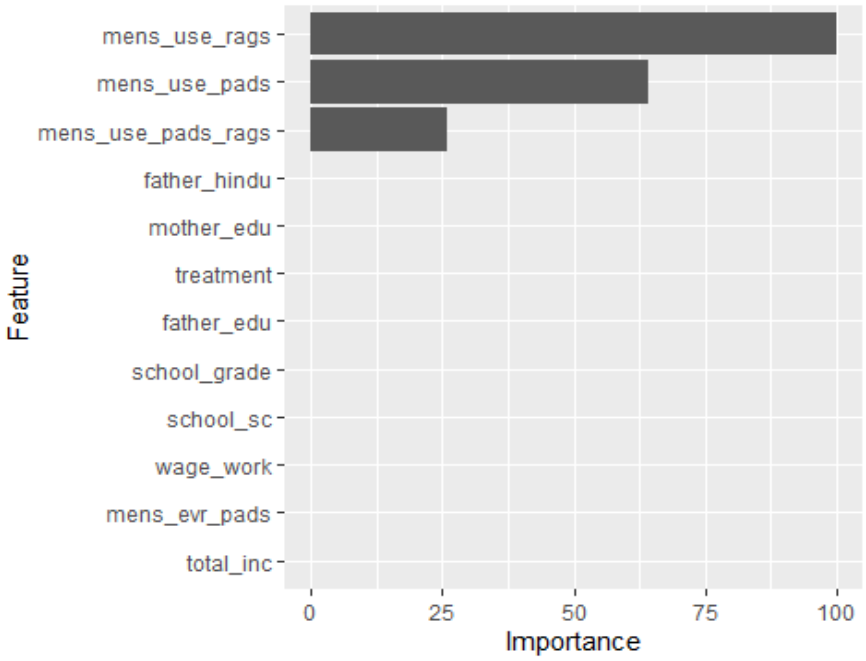

```
#Comments to Plots (1/4): While the training RMSE of LASSO is quite low, it
becomes closer to the test RMSE of the other models across simulations. One
can see the test R-squared of LASSO is smaller than the other models, which
makes sense as there are fewer variations kept with the variable selection.

#(2/4) PCA and OLS look very similar because PCA is not able to perform too
much dimension reduction and cannot put the most variation into the first c
omponents (see Fig. 6). As PCA is linear and in this case, M converges to p
, PCA and OLS become very close.

#(3/4) I use a lot of categorical variables in my data. PCA does not know h
ow to handle it when minimizing between data points and PC, because calcula
ting the distances of 0-1 does not have any explaining value. This also exp
lains why PCA tends to perform like OLS.

#(4/4) Converting the output with r-markdown does change the look of the pl
ots a bit and let the results look more similar. Running the code in R does
not change the results but makes the differences between the plots more obv
ious.

#Choose optimal lambda

set.seed(50)
crossValid <- trainControl(method = "cv", number = 10, savePredictions = "a
ll")
lambda_grid <- 10^seq(5, -5, length = 500)
lasso_mod <- train(Y ~ .,
                   data = training_data,
                   preProcess = c("center","scale"),
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha=1,lambda = lambda_grid),
                   trControl=crossValid)

plot(log(lasso_mod$results$lambda),
     lasso_mod$results$RMSE,
     xlab = "log(lambda)",
```

```
     ylab = "RMSE",
     xlim= c(-5,2),
     type = "l",
     col = "salmon",
     lwd = 2,
     main = "Fig. 4: RMSE across lambda")
```



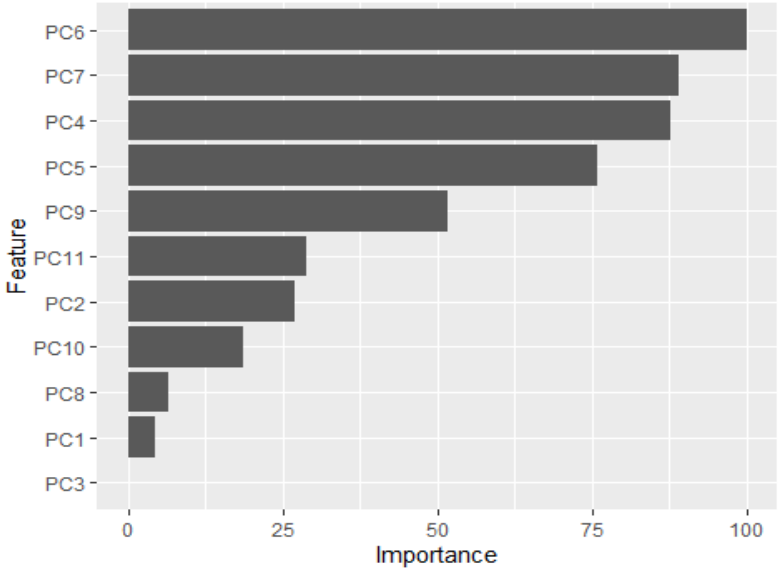

```
ggplot(varImp(lasso_mod)) #importance of variables in the variable selectio
n. LASSO selects with the described procedure in 1.1. One can see that all
variables except three are shrunken down to zero. This is the case for this
seed. Depending on the seed and the underlying data, the feature selection
varies.
```

Fig. 5: Example Variable Selection of LASSO

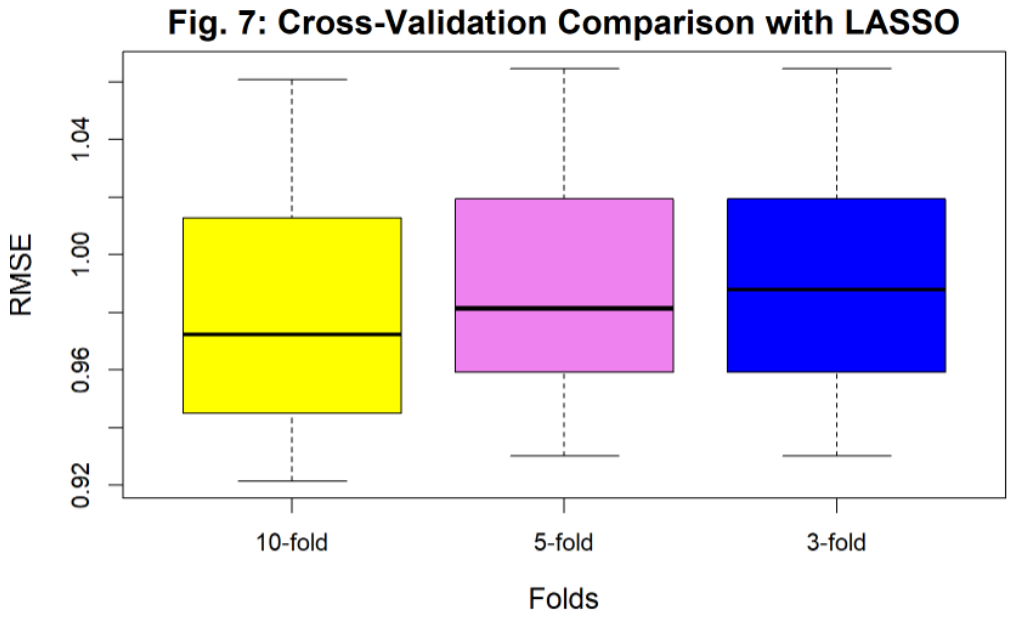


```
ggplot(varImp(pcr_mod))
```

Fig. 6: Example Principal Components Importance




```
rep = 10
cv_it <- 10
#run Simulation Study
cv10 <- RMSE_container_lasso
cv_it <- 5
#run Simulation Study
cv5 <- RMSE_container_lasso
cv_it <- 3
#run Simulation Study
cv3 <- RMSE_container_lasso
CV_comp<- cbind(cv10,cv5,cv3)
boxplot(CV_comp, ylab = "RMSE", xlab = "Folds", main ="Fig. 7: Cross-Valida
tion Comparison with LASSO",
        cex.axis = 1, cex.lab = 1.2, cex.sub = 0.8, cex.main =1.4,
        col = (c("yellow","violet","blue")), names = (c("10-fold","5-fold",
"3-fold")))

#How to choose the optimal folds for CV? An extensive procedure would be to
produce a grid of folds and then run it to estimate the optimal RMSE but th
ere are two main issues. First, it is computationally hardly feasible, runn
ing 50 times a CV and comparing it across models because it takes a lot of
time even with reasonable computing power. Second, there is a bias-variance
trade-off when increasing folds, where lower folds are more biased but have
generally lower variance. In Fig. 7 I give some suggestive insights why I c
hose 10-fold CV and not a 3-fold or even n-fold (Leave-One-Out Cross-Valida
tion) CV on the example of LASSO. One can see that 10-fold performs the bes
t regarding RMSE and, taking theory into account, should be also the lowest
biased one of the three. A caveat is that this just shows one model and a c
ertain seed but in application 10-fold CV seems the way to go.
```



3. Conclusion

In this simulation, I compare three different models by fitting them on a smaller test sample to compare their performance in terms of RMSE and R-squared. The comparison gives insight into whether the use of models like LASSO and PCA can contribute to an improvement of power and higher R-squareds in RCTs. In the simulation, I can show an improvement in terms of statistical power when using LASSO as the variance decreases. This comes at the cost of higher bias and lower R-squareds. PCA does not perform differently from OLS in this simulation setup. The use of this method is strongly dependent on the underlying data as discussed. Using k-fold CV as a resampling method does work well for choosing parameters and evaluating models, especially the 10-fold case seems to work well.

Finally, some remarks for improvements of the study. As the Y-variable is categorial, a classification method like *logistic regression* or *linear discriminant analysis* might be more advisable than OLS. This paper provides an environment of small n and even smaller p, where n becomes relatively close to p in the test set. It would be interesting to look into RCTs that have a situation of $p > n$ or $p \gg n$. The discussed theory predicts that OLS should perform worse and PCA should perform better than in this simulation. The last remark is about the time dimension in data of RCTs. Panel data is a big subject of RCTs as researchers often want to observe time-varying changes in subjects. Performing LASSO and PCA in this environment would be a useful addition but econometrically more challenging than in this cross-sectional case.

4. Appendix: An empirical Application

In this Appendix, I give an example of an empirical application. With the trained models of seed(100), I predict the original empirical data. The obvious issue of this naïve approach is, that the trained model and the empirical one have the same n, nonetheless predicting the original data shows that RMSE and especially R-squareds are much lower than on the average simulated data, indicating small variance but bad explaining ability. That can have multiple reasons like: I) The distribution of the simulation study data does not reflect the original data, II) More Training observations are needed, and III) Selecting the optimal model regarding RMSE and R-squareds. All these points can improve the models predicting ability. To get a better sense of how differently the models perform I applied a one-sided t-test between each models training predictions and the results are that none of the models perform significantly differently from the other ones. These results of course can change with different

seeds and improved model fit but they give doubts whether LASSO or PCA do perform better than OLS in this setting.

```
#run set.seed(100) for the simulation to get the predictions results
df_prework<- as.data.frame(df_prework)
new_row <- data.frame(a4_mother_edu = NA, d12_husband_edu = NA, i11_total_i
nc = NA, score62 = NA, a3_grade = NA, b1_work_yn = NA, c6_mens_evrusepads =
NA,
                      c7_mens_userags = NA, c7_mens_usepads =NA, c7_mens_us
epadsandrags =NA,father_hindu= NA, treatment = NA )
try <- rbind(df_prework,new_row)
try <- rbind(try,new_row)
try[is.na(try)] <- 0
test <- true_y
test <- append(test,0)
test <- append(test,0)
test[is.na(test)] <- 0

#LASSO
app_prediction_lasso <- predict(lasso_mod, newdata = try, na.action = na.pa
ss, se = "TRUE") #predictions of y
app_modeltest_lasso <- postResample(app_prediction_lasso, test)[-3]

#OLS--
#prediction_ols
app_prediction_ols <- predict(ols_mod, newdata = try)
app_modeltest_ols <- postResample(app_prediction_ols, test)[-3]

#pcr--
app_prediction_pcr <- predict(pcr_mod, newdata = try)
app_modeltest_pcr <- postResample(app_prediction_pcr, test)[-3]

RMSE_app<- cbind(app_modeltest_lasso[1],app_modeltest_ols[1],app_modeltest_
pcr[1])
Rsquared_app<- cbind(app_modeltest_lasso[2],app_modeltest_ols[2],app_modelt
est_pcr[2])

lasso_results <- c("LASSO",app_modeltest_lasso[1],app_modeltest_lasso[2])
pls_results <- c("OLS",app_modeltest_ols[1],app_modeltest_ols[2])
app_modeltest_pcr <- c("PCR",app_modeltest_pcr[1],app_modeltest_pcr[2])
print(cbind(lasso_results,pls_results,app_modeltest_pcr))

##          lasso_results         pls_results           app_modeltest_pcr
##          "LASSO"               "OLS"                 "PCR"
## RMSE     "0.55732213182351"    "0.661486227016294"   "0.648824662276396"
## Rsquared "0.00158687776586603" "0.00443057430625223" "0.0060376175744185
3"

compare_models(lasso_mod,ols_mod, metric = "RMSE")

##
##  One Sample t-test
##
## data:  x
## t = -0.87915, df = 9, p-value = 0.4022
```

```
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.1452765  0.0639602
## sample estimates:
##   mean of x
## -0.04065813

compare_models(lasso_mod,ols_mod, metric = "Rsquared")

##
##  One Sample t-test
##
## data:  x
## t = 0.89457, df = 9, p-value = 0.3943
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.06516252  0.15041060
## sample estimates:
##  mean of x
## 0.04262404

compare_models(lasso_mod,pcr_mod, metric = "RMSE")

##
##  One Sample t-test
##
## data:  x
## t = -0.74997, df = 9, p-value = 0.4724
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.13404052  0.06729265
## sample estimates:
##   mean of x
## -0.03337394

compare_models(lasso_mod,pcr_mod, metric = "Rsquared")

##
##  One Sample t-test
##
## data:  x
## t = 1.5054, df = 9, p-value = 0.1665
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.0331950  0.1652726
## sample estimates:
## mean of x
## 0.0660388

compare_models(pcr_mod,ols_mod, metric = "RMSE")

##
##  One Sample t-test
##
## data:  x
## t = -0.2336, df = 9, p-value = 0.8205
```

```
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.07782485  0.06325647
## sample estimates:
##   mean of x
## -0.00728419

compare_models(pcr_mod,ols_mod, metric = "Rsquared")

##
##  One Sample t-test
##
## data:  x
## t = -0.53399, df = 9, p-value = 0.6063
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.1226077  0.0757782
## sample estimates:
##   mean of x
## -0.02341476
```

5. References

1. Banerjee, A., Karlan, D., & Zinman, J. (2015). Six randomized evaluations of microcredit: Introduction and further steps. *American Economic Journal: Applied Economics, 7(1), 1-21*. DOI: 10.1257/app.20140287 https://www.aeaweb.org/articles?id=10.1257/app.20140287
2. Brune, L., Eric C., and J. Kerwin. 2021. "Pay Me Later: Savings Constraints and the Demand for Deferred Payments." *American Economic Review, 111 (7): 2179-2212*. DOI: 10.1257/aer.20191657 https://www.aeaweb.org/articles?id=10.1257%2Faer.20191657&fbclid=IwAR11AzQ-22RLNIy724s1kGo5HUoZ6V9yvszwaZJQmefuzvNxCCB6yKSz0iY
3. Caria, S. and Franklin, S. and Witte, M., Searching with Friends (2020). *IZA Discussion Paper No. 13857*, Available at SSRN: https://ssrn.com/abstract=3730455 or http://dx.doi.org/10.2139/ssrn.3730455
4. Carranza, E., Garlick, R., Orkin, K., & Rankin, N. (2021). Job Search and Hiring with Limited Information about Workseekers' Skills. *American Economic Review*. https://www.aeaweb.org/articles?id=10.1257/aer.20200961
5. Dhar, D., Tarun J., and S. Jayachandran. 2022. "Reshaping Adolescents' Gender Attitudes: Evidence from a School-Based Experiment in India." *American Economic Review, 112 (3): 899-927*. DOI: 10.1257/aer.20201112 https://www.aeaweb.org/articles?id=10.1257/aer.20201112
6. Jones, M. R. (2021), Randomized Control Trials [Online], [Accessed: 16.08.2022], *The World Bank*. Available from: https://dimewiki.worldbank.org/Randomized_Control_Trials
7. Oster, E., and R. Thornton. 2011. "Menstruation, Sanitary Products, and School Attendance: Evidence from a Randomized Evaluation." *American Economic Journal: Applied Economics, 3 (1): 91-100*. DOI: 10.1257/app.3.1.91 https://www.aeaweb.org/articles?id=10.1257/app.3.1.91
8. Romero, Mauricio, Justin Sandefur, and Wayne Aaron Sandholtz. 2020. "Outsourcing Education: Experimental Evidence from Liberia." *American Economic Review, 110 (2): 364-400*. DOI: 10.1257/aer.20181478 https://www.aeaweb.org/articles?id=10.1257/aer.20181478