# Performing Principal Components Regression (PCR) in R

R blog By Michy Alice July 21, 2016 Tags: data mining, PCR, Principal Components, Regression Models, statistical models
6 Comments

This article was originally posted on Quantide blog - see here.

Principal components regression (**PCR**) is a regression technique based on principal component analysis (**PCA**).

The basic idea behind **PCR** is to calculate the **principal components** and then use some of these components as predictors in a linear regression model fitted using the typical least squares procedure.

As you can easily notice, the core idea of PCR is very closely related to the one underlying PCA and the "trick" is very similar. In some cases a small number of principal components are enough to explain the vast majority of the variability in the data. For instance, say you have a dataset of 50 variables that you would like to use to predict a single variable. By using PCR you might found out that 4 or 5 principal components are enough to explain 90% of the variance of your data. In this case, you might be better off running PCR on with these 5 components instead of running a linear model on all the 50 variables. This is a rough example but I hope it helped to get the point through.

A core assumption of **PCR** is that the directions in which the predictors show the most variation are the exact directions associated with the response variable. On one hand, this assumption is not guaranteed to hold 100% of the times, however, even though the assumption is not completely true it can be a good approximation and yield interesting results.

Some of the most notable advantages of performing PCR are the following:

- Dimensionality reduction
- Avoidance of multicollinearity between predictors
- Overfitting mitigation

Let's briefly walk through each one of them:

# Dimensionality reduction

By using PCR you can easily perform dimensionality reduction on a high dimensional dataset and then fit a linear regression model to a smaller set of variables, while at the same time keep most of the variability of the original predictors. Since the use of only some of the principal components reduces the number of variables in the model, this can help in reducing the model complexity, which is always a plus. In case you need a lot of principal components to explain most of the variability in your data, say roughly as many principal components as the number of variables in your dataset, then PCR might not perform that well in that scenario, it might even be worse than plain vanilla linear regression.

PCR tends to perform well when the first principal components are enough to explain most of the variation in the predictors.

# Avoiding multicollinearity

A significant benefit of PCR is that by using the principal components, if there is some degree of multicollinearity between the variables in your dataset, this procedure should be able to avoid this problem since performing PCA on the raw data produces linear combinations of the predictors that are uncorrelated.

# Overfitting mitigation

If all the assumptions underlying PCR hold, then fitting a least squares model to the principal components will lead to better results than fitting a least squares model to the original data since most of the variation and information related to the dependent variable is condensend in the principal components and by estimating less coefficients you can reduce the risk of overfitting.

# Potential drawbacks and warnings

As always with potential benefits come potential risks and drawbacks.

For instance, a typical mistake is to consider PCR a feature selection method. PCR is not a feature selection method because each of the calculated principal components is a linear combination of the original variables. Using principal components instead of the actual features can make it harder to explain what is affecting what.

Another major drawback of PCR is that the directions that best represent each predictor are obtained in an unsupervised way. The dependent variable is not used to identify each principal component direction. This essentially means that it is not certain that the directions found will be the optimal directions to use

when making predictions on the dependent variable.

# Performing PCR on a test dataset

There are a bunch of packages that perform PCR however, in my opinion, the **pls package** offers the easiest option. It is very user friendly and furthermore it can perform data standardization too. Let's make a test.

Before performing PCR, it is preferable to standardize your data. This step is not necessary but strongly suggested since PCA is not scale invariant. You might ask why is it important that each predictor is on the same scale as the others. The scaling will prevent the algorithm to be skewed towards predictors that are dominant in absolute scale but perhaps not so relevant as others. In other words, variables with higher variance will influence more the calculation of the principal components and overall have a larger effect on the final results of the algorithm. Personally I would prefer to standardize the data most of the times.

Another thing to assess before running PCR is missing data: you should remove all the observations containing missing data, or approximate the missing observations with some technique before running the PCR function.

For this toy example, I am using the evergreen iris dataset.

```
1
2 require(pls)
3 set.seed (1000)
4
5 pcr_model <- pcr(Sepal.Length~., data = iris, scale = TRUE, validation = "CV")
```

By setting the parameter scale equal to TRUE the data is standardized before running the pcr algorithm on it. You can also perform validation by setting the argument validation. In this case I chose to perform 10 fold cross-validation and therefore set the validation argument to "CV", however there other validation methods available just type ?pcr in the R command window to gather some more information on the parameters of the pcr function.

In oder to print out the results, simply use the summary function as below

```
1
2 summary(pcr_model)
```

```
 1
 2 ## Data:    X dimension: 150 5
 3 ##  Y dimension: 150 1
 4 ## Fit method: svdpc
 5 ## Number of components considered: 5
 6 ##
 7 ## VALIDATION: RMSEP
 8 ## Cross-validated using 10 random segments.
 9 ##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
10 ## CV           0.8308   0.5141   0.5098   0.3947   0.3309   0.3164
```

```
## adjCV       0.8308   0.5136   0.5092   0.3941   0.3303   0.3156
##
## TRAINING: % variance explained
##                1 comps  2 comps  3 comps  4 comps  5 comps
## X                56.20    88.62    99.07    99.73   100.00
## Sepal.Length     62.71    63.58    78.44    84.95    86.73
```

As you can see, two main results are printed, namely the **validation error** and the **cumulative percentage of variance explained** using n components.

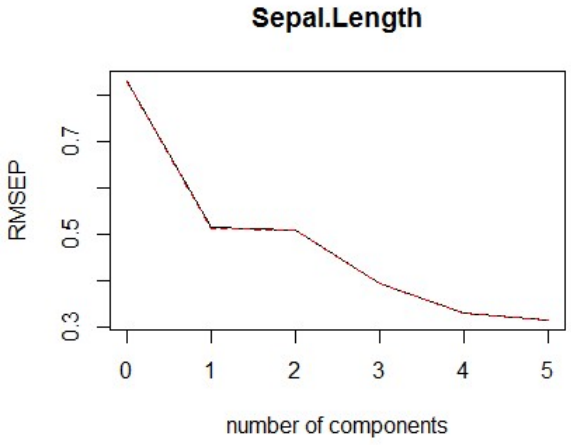
The **cross validation** results are computed for each number of components used so that you can easily check the score with a particular number of components without trying each combination on your own.

The pls package provides also a set of methods to plot the results of PCR. For example you can plot the results of cross validation using the **validationplot** function.

By default, the pcr function computes the root mean squared error and the **validationplot** function plots this statistic, however you can choose to plot the usual mean squared error or the R2 by setting the **val.type** argument equal to “MSEP” or “R2” respectively

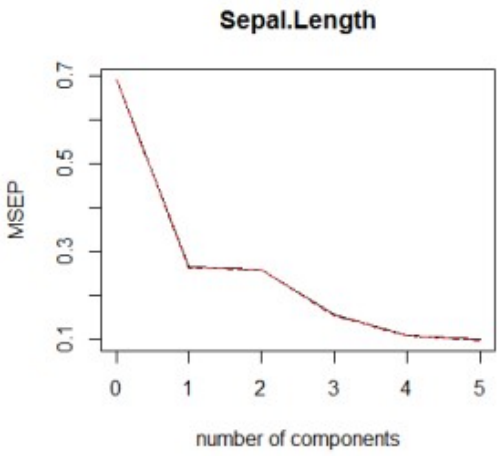
```

# Plot the root mean squared error
validationplot(pcr_model)
```

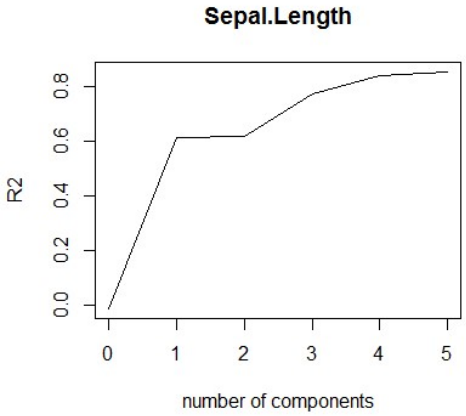



```

# Plot the cross validation MSE
validationplot(pcr_model, val.type="MSEP")
```



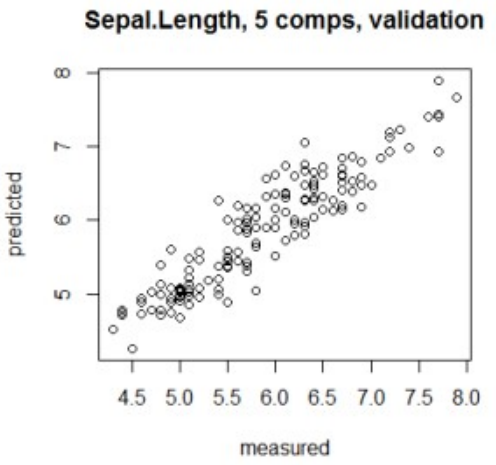

```
1
2 # Plot the R2
3 validationplot(pcr_model, val.type = "R2")
```




What you would like to see is a low **cross validation error** with a lower number of components than the number of variables in your dataset. If this is not the case or if the smalles cross validation error occurs with a number of components close to the number of variables in the original data, then no dimensionality reduction occurs. In the example above, it looks like 3 components are enough to explain more than 90% of the **variability** in the data although the CV score is a little higher than with 4 or 5 components. Finally, note that 6 components explain all the variability as expected.

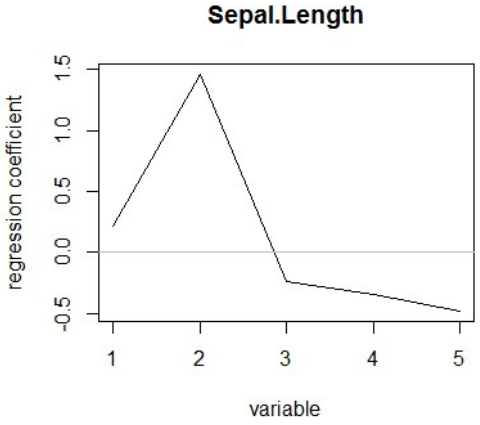
You can plot the predicted vs measured values using the predplot function as below

```
1
2 predplot(pcr_model)
```




while the regression coefficients can be plotted using the coefplot function

```
coefplot(pcr_model)
```




Now you can try to use PCR on a traning-test set and evaluate its performance using, for example, using only 3 components.

```
# Train-test split
train <- iris[1:120,]
y_test <- iris[120:150, 1]
test <- iris[120:150, 2:5]

pcr_model <- pcr(Sepal.Length~., data = train,scale =TRUE, validation = "CV")

pcr_pred <- predict(pcr_model, test, ncomp = 3)
mean((pcr_pred - y_test)^2)
```

```
## [1] 0.213731
```

With the iris dataset there is probably no need to use PCR, in fact, it may even be worse using it.

However, I hope this toy example was useful to introduce this model.

Thank you for reading this article, please feel free to leave a comment if you have any questions or suggestions and share the post with others if you find it useful.

## Related Post

**Cross-Validation for Predictive Analytics Using R**

Introduction Since ancient times, humankind has always avidly sought a way to predict the future. One of the most widely known examples of this kin...

**How to build a color palette from any image with R...**

Some weeks ago, I was working on a dataviz to show results coming from an analysis I had performed, and I found myself looking at that default ggplot2...

**My first date with R - Free live class in Milano**

R Users!! Are you ready for an awesome news?! Microsoft, in collaboration with Quantide, is offering a one-day live course in Milano....

Share:

**Michy Alice**

Electrical Engineering student at Polytechnic University of Milan and graduate from University of Genoa, I enjoy writing R code and expanding my knowledge of statistics and its applications in my free time.

## 6 Comments

**thecowboy**
July 22, 2016 at 7:18 am

You should be able to use the PLS package to accomplish the same goals using the XX' matrix rather than XY via predictive latent space regression (PLSR).

**Log in to Reply**

**Michy Alice**
July 24, 2016 at 8:03 pm

Hi thanks for commenting and for the suggestion! 🙂

**Log in to Reply**

**freerecall**
September 30, 2016 at 6:05 pm

Hi. Thanks so much for the post.

I'm just learning about PCA, and I have two, specific questions that may elaborate on the content posted here. (Apologies if the answers are really obvious - I'm a novice here.)

Firstly, where is the output for the regression coefficients corresponding to each principal component? Is that what the coefficient plot is showing?

Secondly, is it possible to do sequential regression with principal components? I imagine something like adding in sets of principal components with each iteration of the model.

Thanks for your help!

**Log in to Reply**

**Michy Alice**
October 16, 2016 at 3:58 pm

Hi! Thank you for reading it!

Yes the regression coefficients with 5 components are displayed using the coefplot function. You can get all the regression coefficients by using pcr_model$coefficients, however, if you need just those with 5 components, coefplot(pcr_model) will do.

If you would like more flexibility, I'd suggest using the R built-in PCA function princomp. This function runs a simple PCA on the data and returns a list with the results. In particular, that list contains a matrix where each column is one of the eigenvectors. From there you can pretty much decide what to do next in your analysis.

**Log in to Reply**

**r-fan**
February 15, 2017 at 1:35 am

Is there an intercept that is returned using pcr? Suppose I have two independent variables and I am looking for f(x1, x2) = ax1 + bx2 +c, does pcr return a c. I understand that I can get the values of a and b using coef(pcr_model).

Thank you,
R-fan.

**Log in to Reply**

**Michy Alice**
February 21, 2017 at 11:37 pm

Hi R-fan, yes exactly, you should get the regression coefficients just by using coef(pcr_model). The intercept is not included by default for some reason, if you need it, use coef(pcr_model, intercept=TRUE).

**Log in to Reply**

## Leave a Reply

You must be logged in to post a comment.

Search! type and hit enter

## Sponsored by


## Upcoming Events

| May 2016 | | | | | | |
|---|---|---|---|---|---|---|
| S | M | T | W | T | F | S |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | 31 | | | | |

Powered by Eventbrite

## Our free web books

**Rabbit - Introduction to R**

**Ramarro - R for Developers**

## We are part of

R-bloggers

## Join the community

**New user? → Join now**

**Close friend? → Login**

## R courses

**Data Manipulation with R**

**Statistical Models with R**

**R for Beginners**

**Data Mining with R**

**R for Developer**

## Our meetings

1st MilanoR Meeting

## Top Authors


MilanoR (67 Posts)


Quantide srl (16 Posts)


Nicola Sturaro Sommacal (13 Posts)


Enrico Tonini (7 Posts)


Michele Usuelli (6 Posts)


Andrea Spanò (5 Posts)


Andrea Pedretti (4 Posts)


Michy Alice (4 Posts)


Marco Ghislanzoni (3 Posts)


Mec Analytics (3 Posts)

## Tags

Announcements Big Data bioR book call for presentations courses data import data management data manipulation data mining data visualization dplyr GenABEL

ggmap ggplot ggplot2 graphics Hadoop MapReduce maps MilanoR MilanoR Facebook page MilanoR Meeting MilanoR meetings models My first... Next Courses Past Courses Past Meetings plot programming qplot quantide r-lab R blog Revolution Analytics r for beginners rmr2 RStudio sales dashboard shiny statistical learning tidyr tutorial useR2015

## R software

R-project

Task Views

## R documentation

R-project: R manuals

Quantide: R Web Books

## R community

The R Journal

R Bloggers

Crantastic

Sponsored by Quantide | Powered by WordPress