Principal components analysis in clinical studies

Zhongheng Zhang[1], **Adela Castelló**[2,3]

[1]Department of Emergency Medicine, Sir Run-Run Shaw Hospital, Zhejiang University School of Medicine, Hangzhou 310016, China; [2]Cancer Epidemiology Unit, National Center for Epidemiology, Carlos III Institute of Health, Madrid 28029, Spain; [3]Consortium for Biomedical Research in Epidemiology & Public Health (CIBERESP), Carlos III Institute of Health, Madrid 28029, Spain

Correspondence to: Zhongheng Zhang. No. 3, East Qingchun Road, Hangzhou 310016, China. Email: zh_zhang1984@zju.edu.cn.

Abstract: In multivariate analysis, independent variables are usually correlated to each other which can introduce multicollinearity in the regression models. One approach to solve this problem is to apply principal components analysis (PCA) over these variables. This method uses orthogonal transformation to represent sets of potentially correlated variables with principal components (PC) that are linearly uncorrelated. PCs are ordered so that the first PC has the largest possible variance and only some components are selected to represent the correlated variables. As a result, the dimension of the variable space is reduced. This tutorial illustrates how to perform PCA in R environment, the example is a simulated dataset in which two PCs are responsible for the majority of the variance in the data. Furthermore, the visualization of PCA is highlighted.

Keywords: Principal component analysis; multicollinearity; regression; R

Submitted Apr 26, 2017. Accepted for publication Jun 28, 2017.
doi: 10.21037/atm.2017.07.12
View this article at: http://dx.doi.org/10.21037/atm.2017.07.12

Introduction and motivating example

Clinical studies utilizing electronic healthcare records (EHR) usually present a large number of variables. These variables frequently correlate with each other, which will introduce multicollinearity in the regression models (1). Although the problem of collinearity will not compromise the predictive ability of a regression model, it can interfere in determining the precise effect of each predictor. Additionally, the standard errors of the estimations affected by multicollinearity tend to be large, making the inference over such estimations less precise (wider confidence intervals and bigger P values).

The problem of multicollinearity in clinical studies is ubiquitous, and there are many statistical methods being developed to handle it (2). One of the most used methods is the principal component analysis (PCA). This statistical approach reduces a set of intercorrelated variables into a few dimensions that gather a big amount of the variability of the original variables (3). These dimensions are called components and have the properties of collecting highly correlated variables within each component and being uncorrelated with each other (4). An example of the application of this method can be found in Witteveen *et al.*'s article (5). The authors performed an observational study aiming to investigate the value of early systemic inflammation in predicting ICU-acquired weakness (5). Systemic inflammation can be represented by a variety of inflammatory cytokines such as interleukin (IL)-6, IL-8, IL-10, IL-13, tumor necrosis factor (TNF)-α and interferon gamma (IFNγ). These cytokines are correlated with each other, and incorporation of them into a regression model will result in significant collinearity. One type of cytokine is regarded as one dimension, and there are dozens of dimensions in the original dataset. In the study, the authors employed PCA to reduce the dimension. They found that the variance of these ten cytokines can be accounted for by three principal components (PC). As a result, the model was remarkably simplified. The aim of this tutorial is to provide readers with a step-by-step guidance on the performance of PCA, highlighting the interpretation of the output from R codes. Also, the R syntaxes will be explained in as many details as possible, helping readers adapt the syntaxes to their own work.

Dataset simulation

In this article, a dataset is simulated to illustrate the performance of PCA using R. One advantage of simulation is that the underlying structure of the dataset can be controlled. In the example, we set two PCs (y1 and y2) accounting for the variance of the five independent variables X=[x_1, x_2, x_3, x_4, x_5]. While x1 to x5 represent the observed values, y1 and y2 are PCs and, therefore, they do not represent actual data that measures a concrete characteristic of the population under study. Dataset is simulated so that, predictors with subscript of odd number x1, x3 and x5 are responsible for the variation in y2, and predictors with subscript of even number are responsible for the variation in y1.

```
> simData <- function(n) {
 y1 <- rnorm(n)
 y2 <- rnorm(n)
 z=y1+y2+rnorm(n)
 pr=1/(1+exp(-z))
 df <- data.frame(y=as.factor(rbinom(n,1,pr)))
 ySum <- list(y1,y2)
 for(i in 1:5) {
 vari <- ySum[[1+(i%%2)]] + rnorm(n)
 df[[paste("x",i,sep='')]] <- ncol(df)*vari+ncol(df)
 }
 df
 }
> set.seed(666)
> df<-simData(1000)
```

The database was simulated by using a function that was named as *simData()* with a single parameter *n* that represents the total number of observations we will include in the dataset (1,000 in our case). The linear predictor of the logistic regression model is defined as the sum of y1, y2 and a random variance term. Then the linear predictor is converted to the probability. Given the assumption that the outcome is binary and follows binomial distribution, the rbinom() function is employed to simulate the outcome y. The binary outcome variable y is converted to a factor with as.factor() function. The two PCs y1 and y2 are aggregated in a list to facilitate its use in for() loop. The symbol "%%" is a modulus operator that distinguishes even and odd numbers. To make the mean and variance of each x to be different, the variance of each x is scaled and centered by the number of columns at each loop. Notice that the number of columns is increased by one for each loop. Finally, a dataset with 1,000 observations is made up with set.seed() function to make sure that the results are reproducible.

We can have a look at the mean and variance with the following codes.

```
> lapply(df[,-1],function(x) {
        df.sum<-data.frame(mean=mean(x),
        sd=sd(x))
 })
$x1
         mean             sd
1        1.0419681        1.496501
$x2
         mean             sd
1        1.876087         2.844405
$x3
         sd               mean
1        3.133103         4.176136
$x4
         mean             sd
1        4.147199         5.601982
$x5
         mean             sd
1        5.249925         7.115041
```

The lapply() function applies user defined function to the corresponding element of df[,-1]. Here the function is applied to each column of the data frame df[,-1]. The lapply() function returns a list of the same length as the number of df[,-1]. The means are increased approximately by at a step of 1 from x1 to x5. The standard deviations of x1 to x5 are also increasing. The purpose of this step it to mimic the real world setting where variables are not centered at the mean of 0 and scaled to the unit variance.

Principal components analysis (PCA)

The most popular function to perform PCA is the prcomp() function shipped with the base R installation.

The first decisions that should be made are:

(I) Which variables will be included in the PCA: in this case the 5 correlated independent variables included in the PCA are x1 to x5.

(II) Rotation of the variance-covariance matrix: rotation of the variance-covariance matrix usually facilitates the

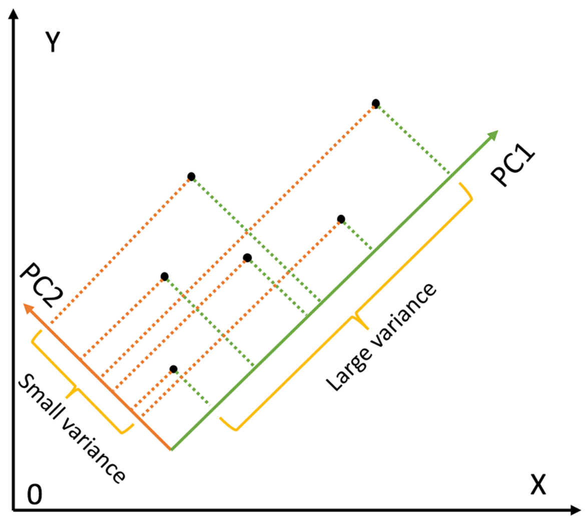

Figure 1 Schematic illustration of how the principal components analysis works.

interpretation of the components. However, this is not always the case. It is advisable to try the PCA with and without rotation and select the most easily interpretable output. In this case we are using the rotated results.

```
> prcom<-prcomp(~.-y,df,scale.=T,center=T)
> prcom
Standard deviations:
[1]  1.4300133   1.2221462     0.7316328    0.6887995    0.6720784
Rotation:
     PC1         PC2            PC3          PC4          PC5
x1   0.58079283  -0.004483858   0.34042830   0.2263993    -0.7039258
x2   0.05066609  -0.705158055   0.54549778   0.2330974    0.3850751
x3   0.56929755  0.085674284    -0.46255853  0.5305290    0.4160995
x4   0.02697392  -0.703686343   -0.60316141  -0.1863703   -0.3249005
x5   0.57903829  0.014746535    0.09368381   -0.7604039   0.2783999
```

The first argument of prcomp() function is a formula without outcome variable. Only numeric variables are allowed. The second argument specifies the data frame that contains the variables in the formula. The "scale.=T" indicates that the variables are scaled to have unit variance, and "center=T" is to shift the mean to zero. By default, the rotated variables will be returned by setting "retx=T". The standard deviations of the five principal components are shown at the beginning of the R output. It is noted that the first two components have the largest standard deviations. The loadings that characterize the role of each variable (x1–x5) in each component (PC1–PC5) are conveyed afterwards. As can be observed, PCA analysis reports as many PCs as the number of variables included in the analyses. Loadings have two properties: (I) their sum of squares within each component are the component's variance (eigenvalue); and (II) they are coefficients in linear combination predicting a variable by the (standardized) components (see calculation of PC1 and PC2 below).

The rule is to select principal components with the largest variance. Consider a dataset in x-y coordinate system, if we want to tease out variation, PCA finds a new coordinate system in which every point has a new (x,y) value. The axes PC1 and PC2 make up a new coordinate system, and they are combinations of the x-y (*Figure 1*). It is obvious that the points projected to PC1 have larger variance than that projected to PC2. As a result, PC1 is a better representation of these data.

The eigenvalue measures the explanatory "strength" of a particular PC. The variance of each PC can be visualized with screeplot() function. Usually a few PC explain a high amount of the variability and some of them need to be selected. The screeplot is used to make this decision as explained below.

```
> screeplot(prcom, npcs = 5,type = "lines")
```

The first argument of screeplot() is an object containing a *sdev* component, which is returned by the prcomp() function. The "npcs=5" specifies the number of PCs to be plotted. The type of plot is specified with type = "lines" argument. The plot shows a deep drop for PC1 and PC2 that stabilizes from PC3 onwards, indicating that the first two PCs collect most of the total variance (*Figure 2*). The importance of each PC can be viewed in the summary() output.

```
> summary(prcom)
Importance of components:
                        PC1     PC2     PC3     PC4      PC5
Standard deviation      1.430   1.2221  0.7316  0.68880  0.67208
Proportion of Variance  0.409   0.2987  0.1071  0.09489  0.09034
Cumulative Proportion   0.409   0.7077  0.8148  0.90966  1.00000
```

The standard deviation of each component is shown in the first row of the output table. The second row shows the proportion of variance explained by each component. It

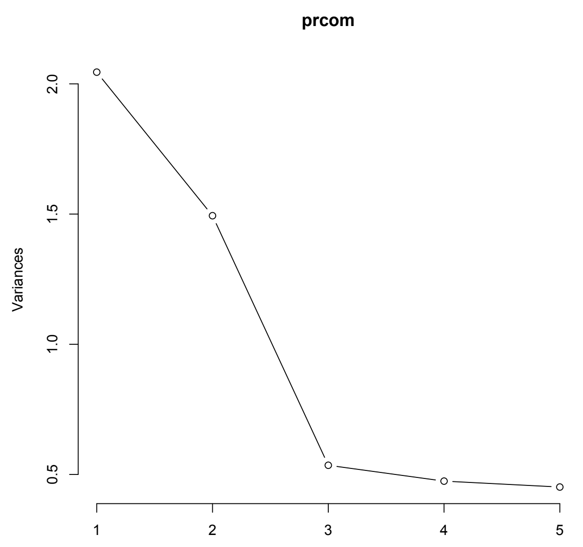

Figure 2 Screeplot representing the variances of all principal components.

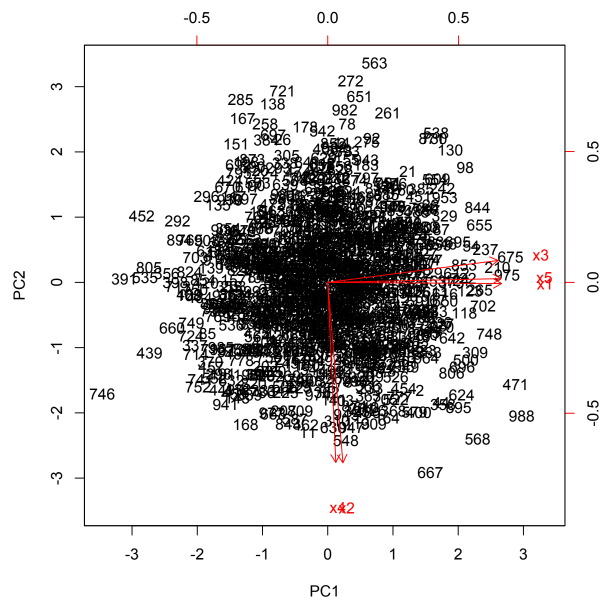

Figure 3 Graphical display of multivariate data with biplot.

appears that the first two PCs account for 70% of the total variance. The last row shows the cumulative proportion of variance.

To decide which variable is represented by each PC, a cut point for the values of the loadings is selected that might vary depending on the type of study. For biological markers, this cut point usually is around |0.5| (5), for nutritional data usually is around |0.3| (6,7) and for other type of studies it might be different depending on the natural correlation between the independent variables included in the PCA. In the current example, if the cut point is set at 0.5, it can be clearly seen that PC1 represents the variability of x1, x3 and x5 and PC2, represents the variability of x2 and x4. In this case PC1 and PC2 will be kept in representation of x1–x5 and the scores that measure the degree of compliance of each of the 1,000 observations of the sample with both of them will be calculated as follows:

$$PC1 = 0.58 \times x1 + 0.05 \times x2 + 0.57 \times x3 + 0.03 \times x4 + 0.58 \times x5$$

The observations that score high in PC1, show high values of x1, x3 and x5 (positive loadings over the cut point of 0.5).

$$PC2 = -0.00 \times x1 - 0.71 \times x2 + 0.09 \times x3 - 0.70 \times x4 + 0.01 \times x5$$

The observations that score high in PC2, have low negative values of x2 and x4 (negative values below the cut point |0.5|).

This two PCs are linearly uncorrelated (-9.18067×10^{-17}).

Results of PCA for real data might be more challenging. Usually the number of predictors included in the PCA is bigger and, in consequence, the number of PCA selected might also be bigger and difficult to assess. The general rule is to select the principal components with the largest variance with the help of the screeplot and keep only those that, explaining enough variance, make epidemiological and/or clinical sense.

Advanced visualization tools

A biplot is a graphical display of multivariate data and can be used in PCA (8). The biplot() shipped with prcomp() function is a good example to display multivariate data in a 2-D plane.

```
> biplot(prcom)
```

The biplot() takes the object prcom returned by prcomp() function. The number of each observation is displayed in the figure, together with the original five axes (*Figure 3*). The figure is a projection of high dimension space onto a 2-D plane. Note that the xs with odd subscript point to the right, whereas the xs with even subscript point downward.

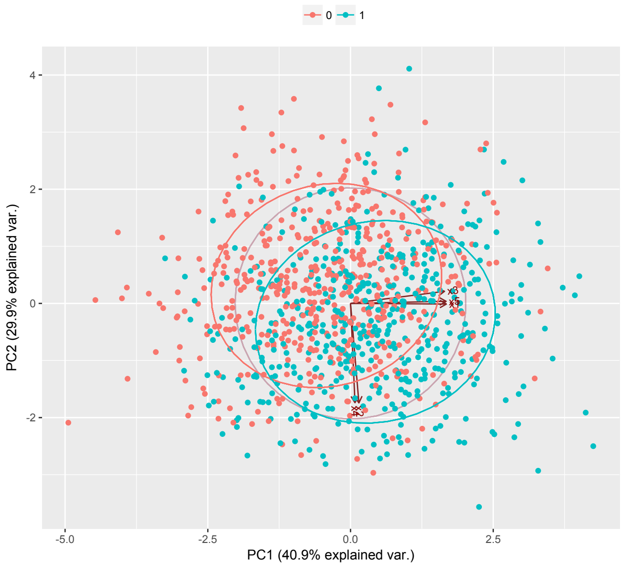

Figure 4 Graphical display of multivariate data with biplot.

The biplot can be modified to show points with different outcome status. The package for the purpose is available from the Github.

```
> library(devtools)
> install_github("ggbiplot", "vqv")
> library(ggbiplot)
> g <- ggbiplot(prcom, obs.scale = 1, var.scale = 1,
 ellipse = TRUE, groups=df$y,
 circle = TRUE)
> g <- g + scale_color_discrete(name = '')
> g <- g + theme(legend.direction = 'horizontal',
 legend.position = 'top')
> print(g)
```

Points with outcome 0 are displayed in red and those with outcome 1 are in green. The axis labels show that PC1 explains 40.9% of the total variance, and PC2 explains 29.9% (*Figure 4*).

Variable loadings can be visualized. The data must be preprocessed before calling plotting functions.

```
> prcom5<-prcom$rotation
> df.prcom = as.data.frame(prcom5)
> df.prcom$varName = rownames(df.prcom)
> library(tidyr)
> df.long.prcom = gather(df.prcom, "PC", "loading",
 starts_with("PC"))
```

As above-mentioned the rotation is a matrix of variable loadings which is extracted and assigned to an object called prcom5. Then the prcom5 object is converted to a data frame. We add an additional variable called varName to store the row names of the df.prcom. Finally, we called the gather() function from tidyr package and reshape the data frame into a "long" format.

```
> library(ggplot2)
> ggplot(df.long.prcom,
 aes_string(x="varName", y="loading", ymax="loading"))+
 geom_point() +
 geom_linerange(aes(ymin=0))+
 facet_wrap(~PC,nrow=1) +
 coord_flip() +
 ggtitle("variable loadings for principal components")
```

The plot is drawn with the ggplot system, in which the elements of a figure can be added layer-by-layer. There are five panels in the figure, each representing one PC. The horizontal axis is the loading values, and the vertical axis is the variable names. It appears that PC1 is primarily contributed by x1, x3 and x5, whereas PC2 is mainly accounted for by x2 and x4 (*Figure 5*).

Regression analysis after PCA

After dimension reduction, the next step is usually to perform regression analysis to explore the association of PCs with outcome variable y. Instead of including the five correlated independent variables (x1–x5) in the model, the two uncorrelated PCs are included, solving the multicollinearity problem.

```
> df.projected <- as.data.frame(predict(prcom,df[,-1]),
 stringsAsFactors = FALSE)
> df.projected$y<-df$y
> ncomp = 2
> regvar = paste(paste("PC", 1:ncomp, sep=''),
 collapse='+')
> fmla = paste("y ~", regvar)
```

The projected values in each PC must be obtained from

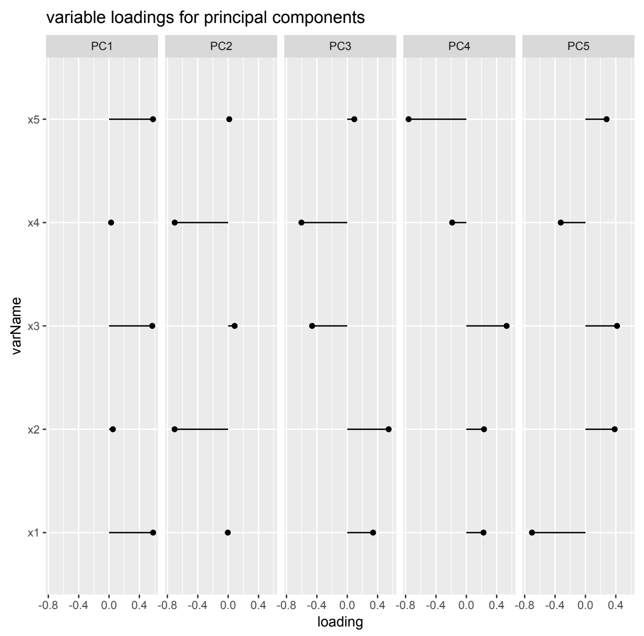

Figure 5 Component loadings that characterize the strength and sing of the association of each independent variable (x1–x5) with each principal component (PC1–PC5).

PCA. They can be obtained using predict() function, or the x component of the prcom object (prcom$x). Both of these functions calculate the PC1 to PC5 scores as specified in the formulas above. The df[,-1] contains variables with which the predict() function predicts values in each PC for each subject. Then the matrix is converted to a data frame by as.data.frame() function. The returned object df.projected has 1,000 rows and 5 columns. Each one of the columns contain a score that measures the level of compliance of the 1,000 observations with each of the 5 components. The outcome y is then attached to the df.projected data frame. The number of components are set to 2. The function paste() is used to connect names of PCs. The returned string will be used as a formula in building the regression model.

```
> mod.com <- glm(fmla,family=binomial,data=df.projected)
> exp(coef(mod.com))
(Intercept)        PC1            PC2
0.9709691          1.6487058      0.5998146
> exp(confint(mod.com))
                2.5%         97.5%
(Intercept)     0.8480303    1.1116326
PC1             1.4878175    1.8339611
PC2             0.5305955    0.6753215
```

Table 1 Regression analysis showing the association between the selected principal components and outcome

PC	Odds ratio (95% CI)	Main loadings of PC
PC1	1.65 (1.49–1.83)	x1, x3, x5
PC2	0.60 (0.53–0.68)	x2, x4

PC, principal component; CI, confidence interval.

The glm() function is used to fit a generalized linear model. By setting the family argument to "binomial", the glm model is a logistic regression model. The glm() function first takes a formula "y ~ PC1+PC2", in which only two PCs are included in the model. The results show that both PC1 and PC2 are significantly associated with outcome y. The exponentiation of regression coefficient gives the odds ratio, which is clinically interpretable. The results can be presented as that in *Table 1* and interpreted as follows: A high compliance with PC1 (high values of x1, x3 and x5) increases a 65% the odds of having the outcome and a high compliance with PC2 (low values of x2 and x4) decreases a 40% the odds of having the outcome.

Acknowledgements

Funding: The study was funded by Zhejiang Engineering Research Center of Intelligent Medicine (2016E10011) from the First Affiliated Hospital of Wenzhou Medical University.

Footnote

Conflicts of Interest: The authors have no conflicts of interest to declare.

References

1. Schisterman EF, Perkins NJ, Mumford SL, et al. Collinearity and Causal Diagrams: A Lesson on the Importance of Model Specification. Epidemiology 2017;28:47-53.
2. Vasquez MM, Hu C, Roe DJ, et al. Least absolute shrinkage and selection operator type methods for the

identification of serum biomarkers of overweight and obesity: simulation and application. BMC Med Res Methodol 2016;16:154.
3. Burt C. Factor analysis and canonical correlations. Br J Psychol 1948;1:95-106.
4. Rencher AC. editor. Principal Component Analysis. 2nd ed. New York: John Wiley & Sons, Inc, 2002.
5. Witteveen E, Wieske L, van der Poll T, et al. Increased Early Systemic Inflammation in ICU-Acquired Weakness; A Prospective Observational Cohort Study. Crit Care Med 2017;45:972-9.
6. Castelló A, Buijsse B, Martín M, et al. Evaluating the Applicability of Data-Driven Dietary Patterns to Independent Samples with a Focus on Measurement Tools for Pattern Similarity. J Acad Nutr Diet 2016;116:1914-6.
7. Castelló A, Lope V, Vioque J, et al. Reproducibility of data-driven dietary patterns in two groups of adult Spanish women from different studies. Br J Nutr 2016;116:734-42.
8. Gabriel KR, Odoroff CL. Biplots in biomedical research. Stat Med 1990;9:469-85.

Cite this article as: Zhang Z, Castelló A. Principal components analysis in clinical studies. Ann Transl Med 2017;5(17):351. doi: 10.21037/atm.2017.07.12