STAT_430_Project_ielopez2

Isaias Lopez

4/28/2020

JMJ+

Introduction

In the first section, there will be a comparison between two different Principle Component Analysis (PCA) functions using prcomp and princomp (the function learned in class), including visualisation comparisons using the factoextra package. After this comparison, we hope to see if one is more interpretable than the other. The next section will then compare PC regression results using various modeling methods. In the next section, we will then compare the results using factor analysis, comparing loadings and how implementing a varimax rotation affects both PCA and Factor Analysis loadings results.

The dataset that we will be using to help anaylze these methods will be the birthwt data from the MASS package, a data set concerning with the "Risk Factors Associated with Low Infant Birth Weight". All variables are numeric (int).

During the section on regression methods, we will be predicting birth weight in grams of infants, comparing results using principle components calculated from prcomp() and princomp() as the feature variables. We will also explore comparisons with using the train model function to implement PC regression and partial linear regression, a method that has relevance with PC regression.

Packages used in this project include MASS, caret, factoextra, corrplot, tidyverse, psych and GPArotation.

Methods

Differences between prcomp() and princomp for computing PCA

Singular Value Decomposition vs. Spectral Value Decomposition

According to the R documentation for prcomp(), the principle components are caculated using Singular Value Decomposition on the "(centered and possibly scaled) data matrix". It does not use eigen() on the covarience matrix. The R documentation suggests this is the best method in terms of numerical accuracy.

In comparison with princomp(), the principle components are calculated using Spectral Value Decomposition. According to the R documentation, calculations of principle components are done using the eigen() function on the covariance and correlation matrix. eigen() is a function that computes the "spectral deomposition of a matrix". This was done to be compaitible with results from S-PLUS.

Spectral and Singular Value decomposition are simliar, but the main difference is that singular decomposition "can be applied to any $m \times n$ matrix." Spectral value decomposition can be "only applied to diagonalizable matrices."

Here is a review of Singular Value Decomposition:

Let the singular decomposition of X be

$$X = U\Sigma W^T$$

Let Σ be an $n \times p$ rectangular diagonal matrix of σ_k, positive numbers. These are the singular values of X.

Let U be an $n \times n$ orthoganol matrix where vectors $u_j, j = 1, ..n$ is an orthonormal basis for the column vector space of X.

The columns of the $p \times p$ matrix W are orthogonal unit vectors with length p. These are the "right singular vectors of X".

Provided by wikipedia, the factorization and the eigen decomposition of $X^T X$ can be written as:

$$\begin{aligned} X^T X &= W\Sigma^T U^T U\Sigma W^T \\ &= W\Sigma^T \Sigma W^T \\ &= W\hat{\Sigma}^2 W^T \end{aligned}$$

The sample covariance for X is $S = X^T X/N$.

Where $\hat{\Sigma}$ is a square diagonal matrix that has the singular values of X.

Knowing this, comparison of the eigenvector factorization of $X^T X$ demonstrates the right singular vectors from W from those of X are the equal to eigenvectors from $X^T X$. The singular values of X are also equivalent to the square-root of eigenvalues from $X^T X$.

Knowing $X^T X$ to be a semi-positive definite matrix, eigenvalues are non-negative.

The eigenvectors associated with $X^T X$ are the "principle component directions" of X.

Say v_k are the eigen vectors that are associated with eigenvalues λ_k^2 from $X^T X$.

Let us look at an example of using only two principle components.

- v_1 is associated with largest eigenvalue λ_1^2 from $X^T X$.
- $z_1 = Xv_1$ would have the largest sample variance from among all the normalized linear combinations from columns X.
- z_1 would be "the first principle component of X", where $Var(z_1) = \lambda_1^2/N$.

The second principle component would have v_2 and would be associated with λ_1^2, the next largest eigenvalue from $X^T X$.

Principle Component k is orthoganal to Principle Component $k - 1$.

As dicussed earlier, prcomp() uses the Singular Value decomposition, which makes it a Q-mode analysis, where pattern similiarities are sought among the subjects.

With princomp(), using the spectral value decomposition allows for R mode analysis, which through using the correlations and covariances, patterns similiarities are sought among the variables.

Using factoextra and corrplot for PCA result data visualizations

The "factoextra" package is supplied with functions that are supposed to be easy-to-use for extraction and visualization of results from Multivariate Analysis. This include Principal Component Analysis (PCA) computed using different R packages. The data visualizations are provided through using ggplot2.

Functions used include:

- fviz_eig(): used to create a screeplot.
- fviz_pca_var(): used to graph variables.

- fviz_pca_ind(): used to graph individual points.
- get_pca_var(): used to extract results from variables.

The corrplot package is used for graphical display of a general matrix, correlation matrix and confidence interval plus algorithms for matrix reordering.

Function used include:

- corrplot() - used to visualize correlation matricies

Comparing Regression results using Principle Components computed from prcomp() and princomp()

Using the principle components as feature variables, supervised regression models will be used, starting with a parametric linear model using a k = 10 fold cross validation. The function used will be lm().

We will also explore regression methods using the train() function from the caret package. The train() function is used to tune models accordingly with optimum resampling statistics. In the case of this project, we are just implmenting a k = 10 cross validation and setting scale equal to true which scales vector X "by dividing each variable by its sample standard deviation." The special option with train() is that we are able to use all kinds of parametric and non-parametric models. In this case, we will create models using the "method" parameter equal to:

- pcr - principle component analysis
- pls - partial least squares

A partial least squares model will be explored because of its relation to principle component analysis, seeing if it outperforms pcr in prediction. Using these methods in train() invoke the pls package.

Using fa() from the psych package

To implement factor analysis, the funciton fa() from the "psych" will be used. This function does an Exploratory Factor analysis, using minimum residual as the default factoring method. Instead of principle components, factor analysis will find a set of latent variables that may explain the variance between variables.

After implementing fa() on the correlation matrix, computed using cor(), of the variables in the "Risk Factors Associated with Low Infant Birth Weight" dataset. We will use a screeplot to choose the number of factor variables to use. The loadings will be compared to the loadings computed from the PCA functions prcomp() and princomp().

Using varimax from the "stats" package

This project will conclude by rotating the loadings of both Factor Analysis and PCA results, implementing a varimax rotation to the latent factors. Unlike fa() which has a "rotate" parameter that can be set to implement a varimax rotation, prcomp() and princomp() do not have this option. That is why we will have to use the varimax() function to rotate the loadings using varimax(). The only issue is that we will have to do some extra steps because the varimax function was developed specifically for rotating "loading matrices in factor analysis."

Simply rotating PCA results, according to a StackExchange post, does not project orthogonaly in the rotated loadings direction, where the rotation happens in the latent space, not the original space. The writer argues that rather the more succinct procedure occuring is "PCA followed by a varimax rotation". That is why we first calculate the raw loadings and then use the varimax() function on the raw loadings obtained.

Analysis

Prcomp() screeplot

```r
library("MASS")
library("factoextra")
```

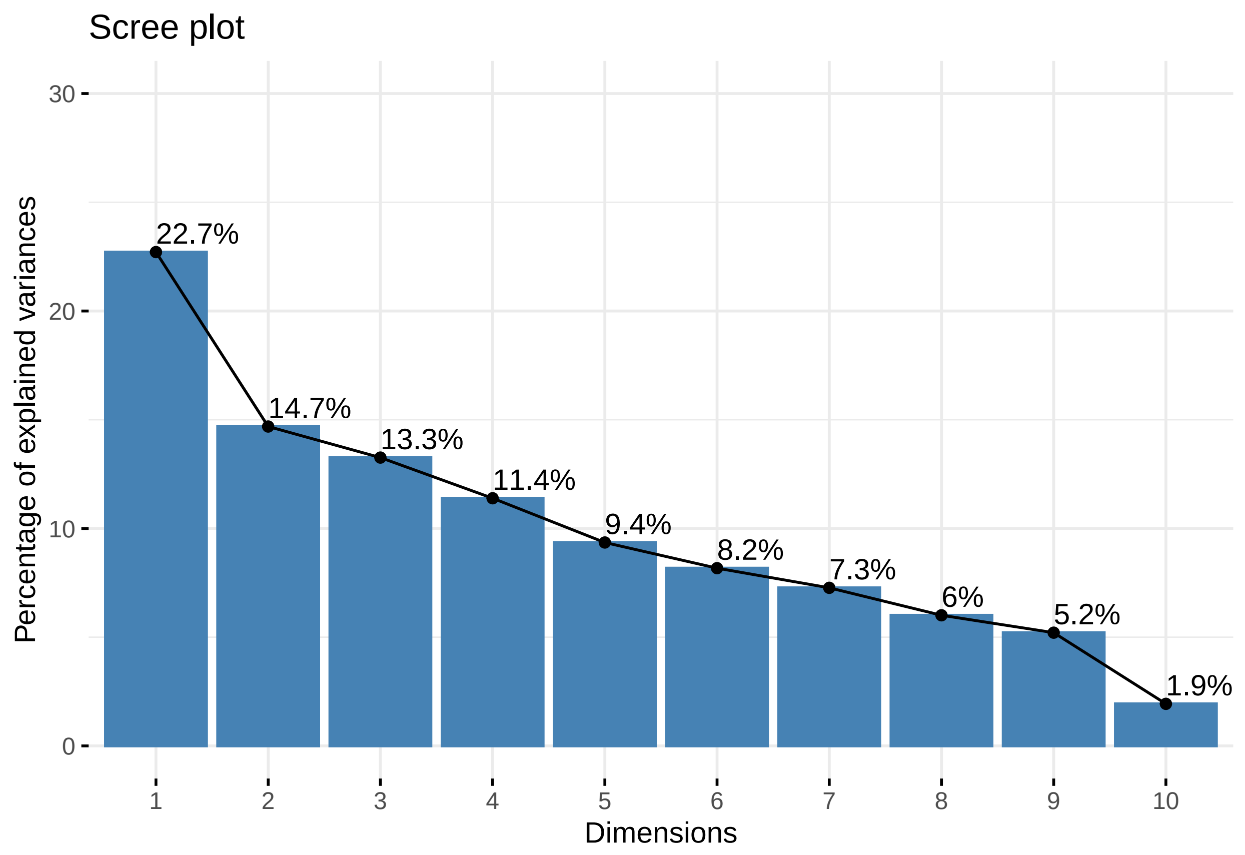
```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
?birthwt

cor_matrix = cor(birthwt)
eigen_info = eigen(cor_matrix)

pca_singleVD = prcomp(x = birthwt, scale = TRUE)
summary(pca_singleVD)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.5070 1.2120 1.1515 1.0672 0.96721 0.90414 0.85268
## Proportion of Variance 0.2271 0.1469 0.1326 0.1139 0.09355 0.08175 0.07271
## Cumulative Proportion  0.2271 0.3740 0.5066 0.6205 0.71403 0.79577 0.86848
##                            PC8     PC9    PC10
## Standard deviation     0.77512 0.72167 0.43995
## Proportion of Variance 0.06008 0.05208 0.01936
## Cumulative Proportion  0.92856 0.98064 1.00000
```

```r
fviz_eig(X = pca_singleVD, addlabels = TRUE, ylim = c(0, 30))
```



```
sum(pca_singleVD$sdev^2 > 1)
```

```
## [1] 4
```

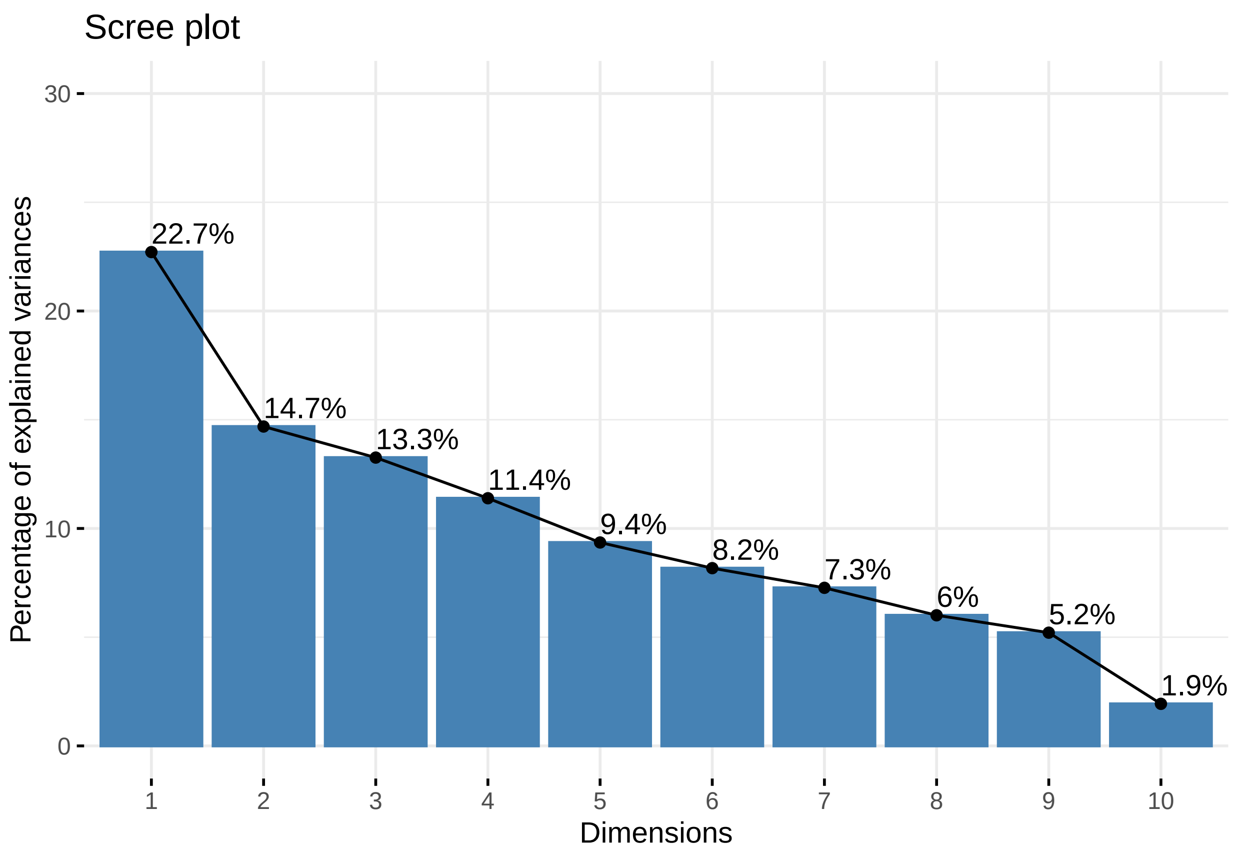
According to the screeplot, using the prcomp() function for computing principle components, we may be only interested in the first 4 principle components since they are greater than one.

Princomp() screeplot

```
pca_spectralVD = princomp(x = birthwt, cor = TRUE, scores = TRUE)
summary(pca_spectralVD)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3    Comp.4     Comp.5
## Standard deviation     1.5070089 1.2119519 1.1514767 1.0672327 0.96721190
## Proportion of Variance 0.2271076 0.1468827 0.1325899 0.1138986 0.09354989
## Cumulative Proportion  0.2271076 0.3739903 0.5065802 0.6204787 0.71402860
##                           Comp.6    Comp.7     Comp.8     Comp.9    Comp.10
## Standard deviation     0.9041360 0.8526840 0.77512124 0.72167426 0.43994929
## Proportion of Variance 0.0817462 0.0727070 0.06008129 0.05208137 0.01935554
## Cumulative Proportion  0.7957748 0.8684818 0.92856309 0.98064446 1.00000000
```

```
fviz_eig(X = pca_spectralVD, addlabels = TRUE, ylim = c(0, 30))
```



```
# plot(pca_spectralVD$sdev^2, ylab = "Eigenvalues using princomp()")
sum(pca_spectralVD$sdev^2 > 1)
```

```
## [1] 4
```

When using princomp(), it seems that the screeplot indicates that only the eigenvalues for the first four principle components are greater than one, therefore we also chose to use only 4 principle components.

Upon inspection of the percentage of the variances accounted for by each principle component (PC), the percentages for principle components calculated using princomp() are close to those calculated using prcomp().

```
score_difference = pca_spectralVD$scores - pca_singleVD$x
score_difference[1:6,]
```

```
##           Comp.1     Comp.2        Comp.3      Comp.4      Comp.5        Comp.6
## 85  6.960571e-04 -1.2961782  0.0001427414  0.07735904 -2.18518238 -0.0077405848
## 86 -3.060788e-03  0.2484341  0.0022890512  5.65186443  0.89884612  0.0005157149
## 87  1.323994e-04  1.2717096 -0.0019746616 -2.19584441  2.33321552  0.0004879043
## 88  1.787175e-03  2.0689348 -0.0041637988  0.14165381  1.42728451 -0.0054655081
## 89  2.915517e-03  0.6456123 -0.0043785563 -2.65205574 -0.03384481 -0.0052261836
## 91 -2.549736e-05 -3.4661584  0.0007014947 -0.09746993 -0.25565338  0.0007072018
##           Comp.7     Comp.8     Comp.9    Comp.10
## 85  0.0015311737  1.5190525  0.1597235 -1.3004388
## 86 -0.0002609420  0.5126734 -1.9514126 -1.3500177
## 87 -0.0004981447 -0.8986886 -0.5550984 -1.7136438
## 88 -0.0005884235 -2.5004550 -1.4038210 -0.8558428
## 89  0.0013138323 -1.6977717 -1.0438896 -0.9454668
## 91  0.0005333657  0.7561358 -0.6973536 -1.5244205
```

```
apply(abs(score_difference), 2, mean)
```

```
##      Comp.1      Comp.2      Comp.3      Comp.4      Comp.5      Comp.6
## 0.003392017 2.012008473 0.002004732 1.633552609 1.413310130 0.001756018
##      Comp.7      Comp.8      Comp.9     Comp.10
## 0.001740541 1.141832004 1.165248856 0.730777099
```

```
loadings_difference = pca_singleVD$rotation - pca_spectralVD$loadings
loadings_difference
```

```
##
## Loadings:
##       PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9    PC10
## low          -0.246        -0.213 -0.423               -0.322 -0.573 -1.317
## age          -0.730        -1.041  0.461                0.219  0.521 -0.133
## lwt          -0.545                0.432               -1.301
## race          1.155        -0.585  0.261               -0.455  1.223 -0.209
## smoke        -1.066         0.838 -0.372               -0.220  1.259 -0.190
## ptl          -0.492        -0.291  1.204               -0.337 -0.469  0.181
## ht           -0.271         0.398  0.659                1.179  0.179
## ui                         -0.374  0.577                0.469  0.256 -0.266
## ftv          -0.524        -1.163 -0.917                0.382
## bwt           0.216         0.296  0.398                0.174        -1.435
##
##                PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10
## SS loadings      0 4.0 0.0 4.0 4.0 0.0 0.0 4.0 4.0  4.0
## Proportion Var   0 0.4 0.0 0.4 0.4 0.0 0.0 0.4 0.4  0.4
## Cumulative Var   0 0.4 0.4 0.8 1.2 1.2 1.2 1.6 2.0  2.4
```

Taking the difference of the scores between results using prcomp() and princomp(), upon inspecting the first six rows, the scores for the first princple components are very similiar, where the difference is close to zero. This is the case for principle components three, six, and seven as well. The rest of the principle components however indicate a large magnitude difference in comparison,in particular the fourth principle component. This reflects in taking the mean absolute value of each principle component.

The loadings also have a similiar trend, where principle component one, three, six and seven have each correponding difference values close to zero. The rest show large magnitudes of difference.

This is puzzling because the expected result would be that the scores between prcomp() and princomp() would be expected to be the same or at least simliar since both functions are set to compute their corresponding decomposition methods on the correlation matrix.

However, the difference could be coming from how the variance is calculated. As stated from the help file for prcomp(), the variances in prcomp() are calculated with the $N-1$ as the divisor. In princomp(), it uses an N as a divisor which is shown in a snippet of the princomp() source code below:

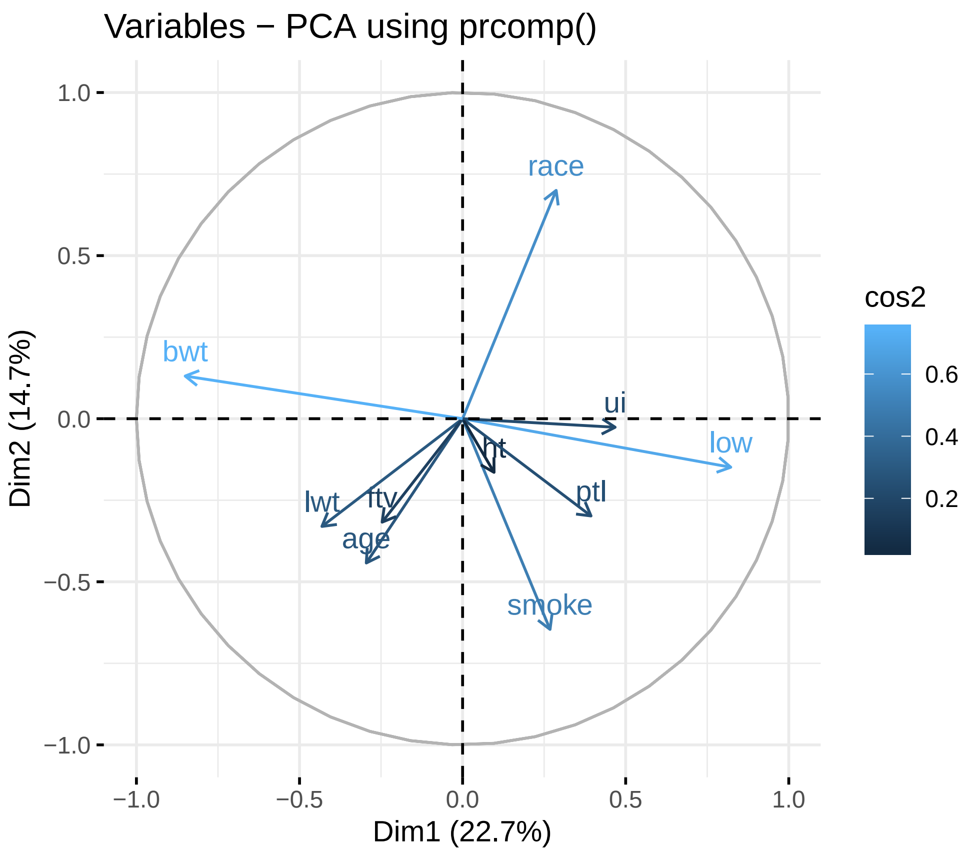
```
  else if (is.null(covmat)) {
    dn <- dim(z)
    if (dn[1L] < dn[2L])
      stop("'princomp' can only be used with more units than variables")
    covmat <- cov.wt(z)
    n.obs <- covmat$n.obs
    cv <- covmat$cov * (1 - 1/n.obs) # N as the divisor
    cen <- covmat$center
  }
```

This difference may be causing the slight difference in values when each functions calculates its corresponding correlation matrix, ultimatly affecting the score values.

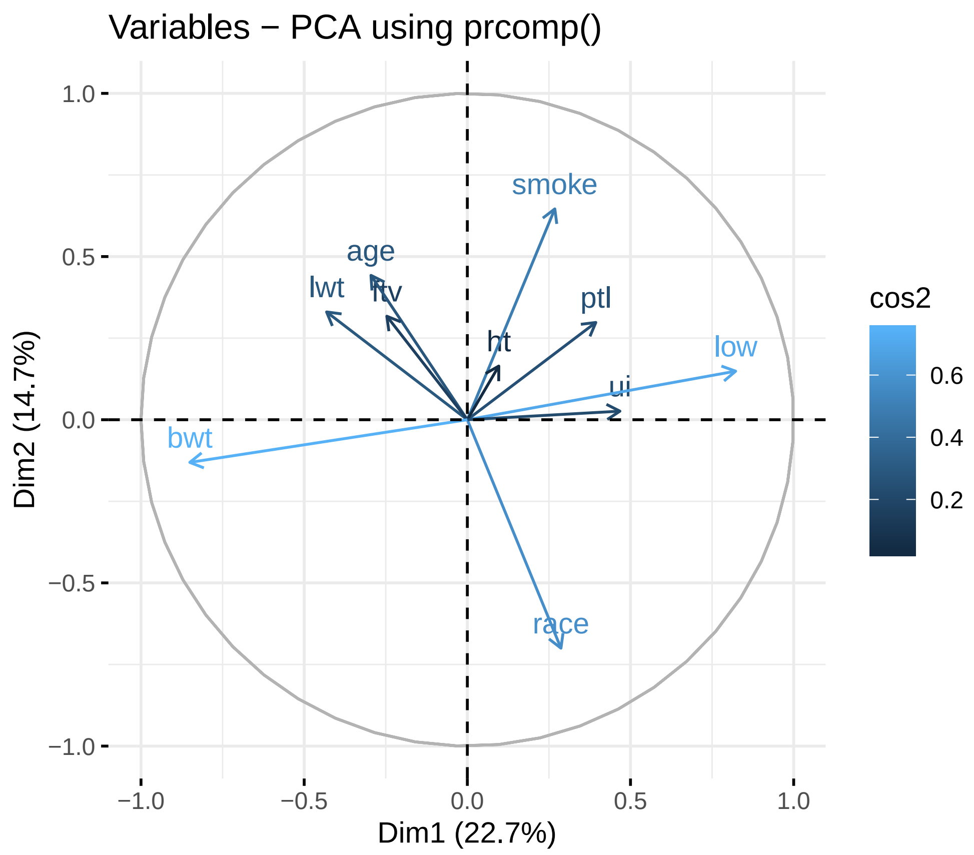
Visualizations of PCA and differences between using prcomp() and princomp()

Correlation Circle Visualizations

```
par(mfrow=c(2,2))

#PCA Correlation Circle Visualization using prcomp().
fviz_pca_var(X = pca_singleVD,
             col.var = "cos2", title = "Variables - PCA using prcomp()")
```



```
#PCA Correlation Circle Visualization using prcomp().
fviz_pca_var(X = pca_spectralVD,
             col.var = "cos2", title = "Variables - PCA using prcomp()")
```



The “variable correlation plots” display relationships between variables. The graphs diplayed represents the correlation of variables with Principle component 1 (Dim.1) and principle component 2 (Dim.2).

The variables that are positively correlated may be grouped together, while the variables negatively correlated are on the opposed quadrants.

Distance between the variables as well as the origin is the measure of the variables’ quality represented on the factor map, which the “quality of representation” is $\cos^2$. The higher the $\cos^2$, the more it is represented in the Principle Component, plus, as indicated on the factor map, the lighter shade of blue. Darker shades of blue indicate a lower measure of representation the specific variable has. The correlation coefficient between any two vectors that represent variables is represented by the cosine angle between them.

In the case of the current correlation circles, it seems for both prcomp() and princomp(), both the visuals indicate that for principle component one, variables race and smoke contribute the most. For principle component two, variables bwt and low contribute the most.

While the positions of these and all the vectors look different between the plots using prcomp() and princomp(), upon close inspection, they are essentially very similiar. The only difference is that the visualizations using princomp() is the reflection across the horizonatal axis of the graphs using prcomp().

As mentioned before on the section concerning scores, the reflection could be caused by the differences of how the variances are computed. Looking at the help documentation of prcomp(), it states “unlike princomp, variances are computed with the usual divisor N - 1.”

Extra methods for finding contributions of variables to specific Principle Components

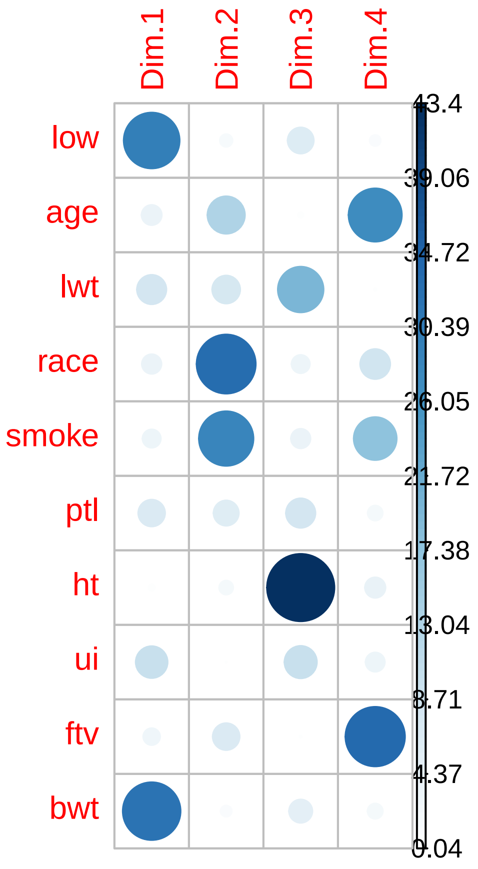
```
library("corrplot")
```

```
## corrplot 0.84 loaded
```

```
# Contribution tables for variables using prcomp():PC 1 and 2
prcomp_var = get_pca_var(pca_singleVD)
prcomp_var$contrib[, 1:4]
```

```
##            Dim.1       Dim.2       Dim.3       Dim.4
## low   29.7701285  1.50727697  6.47529056  1.13318034
## age    3.8407305 13.32456249  0.03598973 27.07627593
## lwt    8.1940228  7.42713444 19.89359396  0.09194011
## race   3.6242436 33.36414808  3.14586748  8.54527464
## smoke  3.1640679 28.41977602  3.59908683 17.56537490
## ptl    6.8275548  6.05185878  8.24302270  2.11509430
## ht     0.4089388  1.84238659 43.39557325  3.96534683
## ui     9.6316686  0.04691004  9.97871211  3.48934658
## ftv    2.6809580  6.85257285  0.08077856 33.83111143
## bwt   31.8576865  1.16337374  5.15208481  2.18705493
```

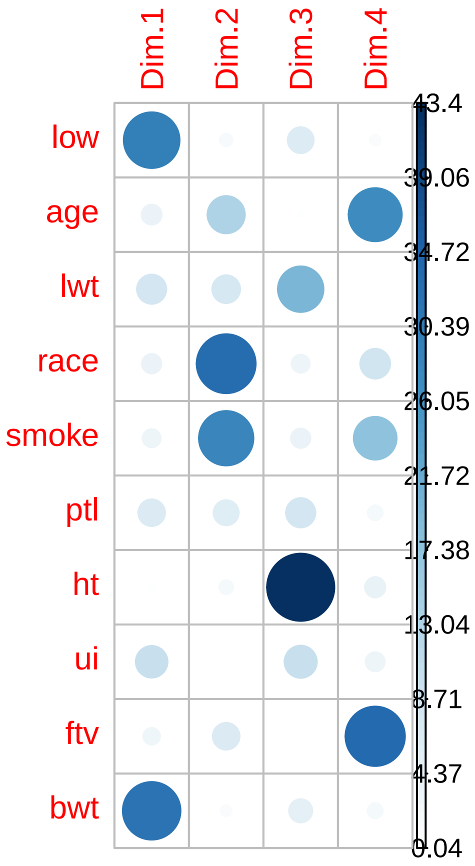
```
corrplot(prcomp_var$contrib[, 1:4], is.corr = FALSE)
```



```
# Contribution tables for variables using princomp():PC 1 and 2
princomp_var = get_pca_var(pca_spectralVD)
princomp_var$contrib[, 1:4]
```

```
##            Dim.1       Dim.2       Dim.3       Dim.4
## low   29.7701285  1.50727697  6.47529056  1.13318034
## age    3.8407305 13.32456249  0.03598973 27.07627593
## lwt    8.1940228  7.42713444 19.89359396  0.09194011
## race   3.6242436 33.36414808  3.14586748  8.54527464
## smoke   3.1640679 28.41977602  3.59908683 17.56537490
## ptl     6.8275548  6.05185878  8.24302270  2.11509430
## ht      0.4089388  1.84238659 43.39557325  3.96534683
## ui      9.6316686  0.04691004  9.97871211  3.48934658
## ftv     2.6809580  6.85257285  0.08077856 33.83111143
## bwt    31.8576865  1.16337374  5.15208481  2.18705493
```

```
corrplot(princomp_var$contrib[, 1:4], is.corr = FALSE)
```



Another method that clarifies which variables contribute the most to which principle component is using the contribution table which can be plotted to be a barplot or a correlation plot. The contribution is the measure of the variability of each variable in a particular principle component, which are expressed as percentages. The higher the value, the more it contributes.

As stated before, principle component 1 (Dim.1) has the most variability mainly coming from bwt, and low. Principle Component 2 (Dim.2) from race and smoke. Principle Component 3 (Dim.3) from ht, and lwt. The fourth, from ftv and age.

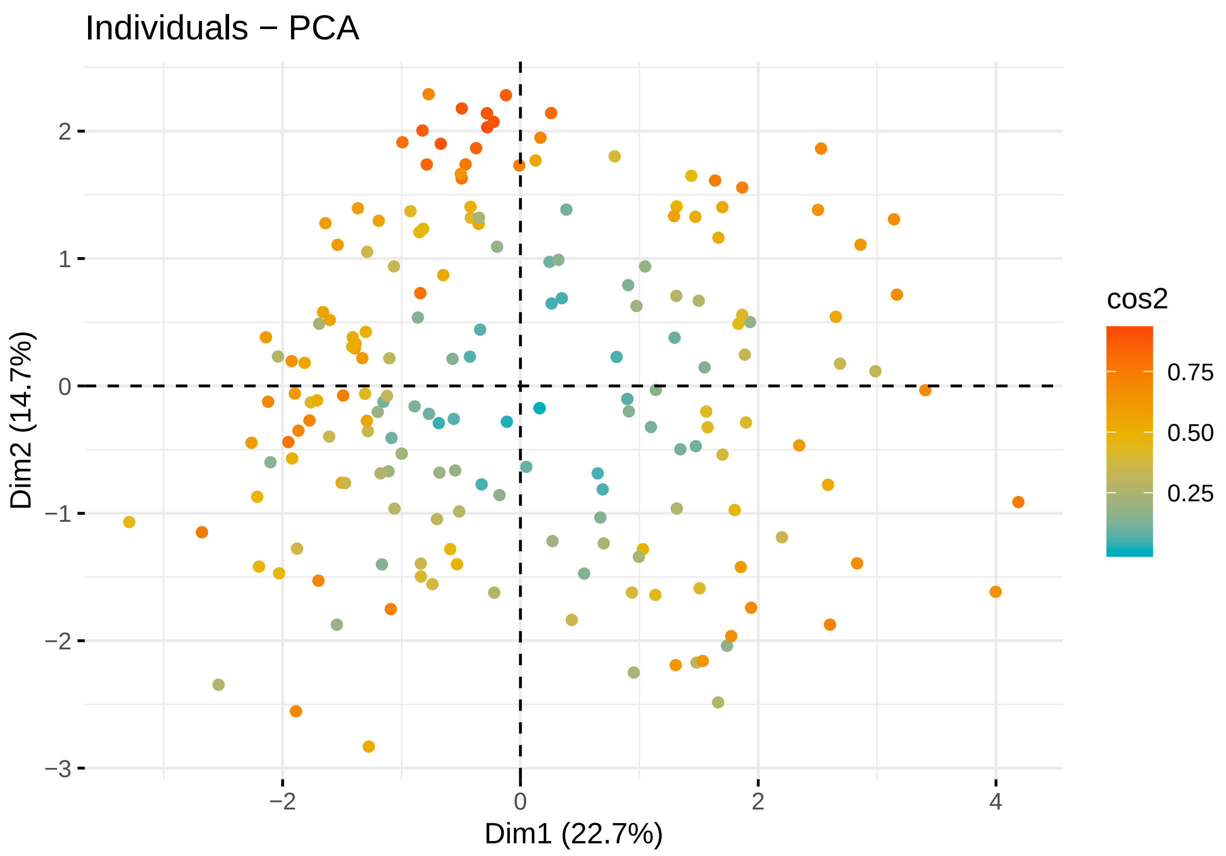
The correlation plot is another good visual to use to notice contribution levels of variables within principle components where each cell contains a circle as a visual measure of how much the variables contributes. The larger the percentage and contribution to the principle component, the bigger and darker the circle is.

Note: Dim.k is the kth principle component.

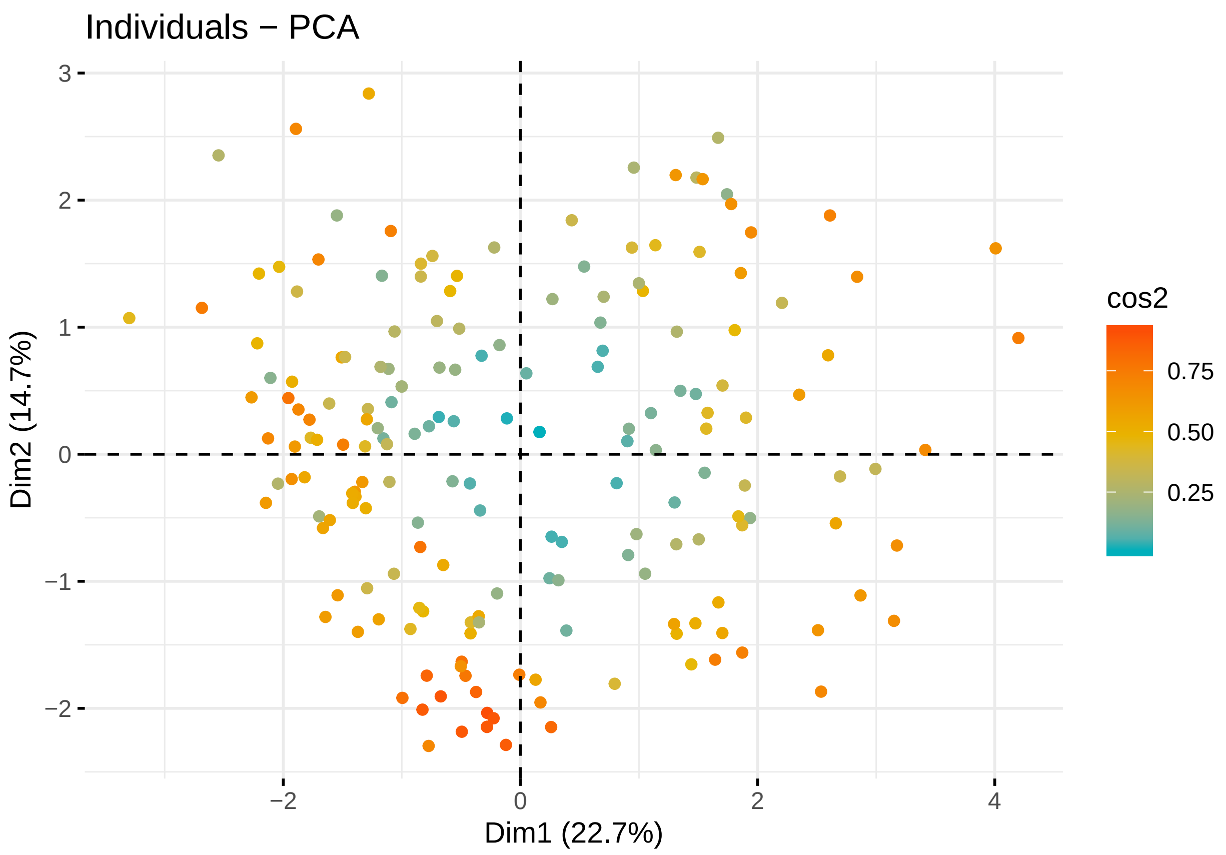
"Individuals" visulizations

```
par(mfrow=c(1,2))

#"Individials" visualization for PCA using prcomp()
```

```r
fviz_pca_ind(X = pca_singleVD,
             col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             geom = "point")
```



```r
#"Individials" visualization for PCA using princomp()
fviz_pca_ind(X = pca_spectralVD,
             col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             geom = "point")
```



The points on the graph represent each individual mother in the dataset. The points, similiar to the variables in the correlation circle visual, have a factor map that colors each point by its cos^2. The points that are grouped close to each other are similiar. This is another way you take a closer look at analysing the contibutions of points and group of points to particular principle components, in this case, to principle components one and two.

As pointed out before, while the graphs look different, the points on prcomp() is merely the reflection across the horizontal axis of the graph for princomp(). Again the reason could be the differences in how each function calculates its corresponding variances.

Note: The points are represented through their projection.

Comparing Principle Component Regression results using princomp() vs. prcomp() for PCA

For this section, we will be comparing Principle Component Regression (PCR) when using prcomp() and princomp(). In this analysis, since our analysis before concluded that four principle components decribes most of the variability in the data, we will regress using four principal components. We will be predicting the birth weight of children given these four PCs. The model used will be parametric linear models, using the lm() function. We will also explore using the pls package with its own functions to implement PCR. Our measure for how well our model predicts is the Root Mean Square Error.

```r
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(pls)
```

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

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

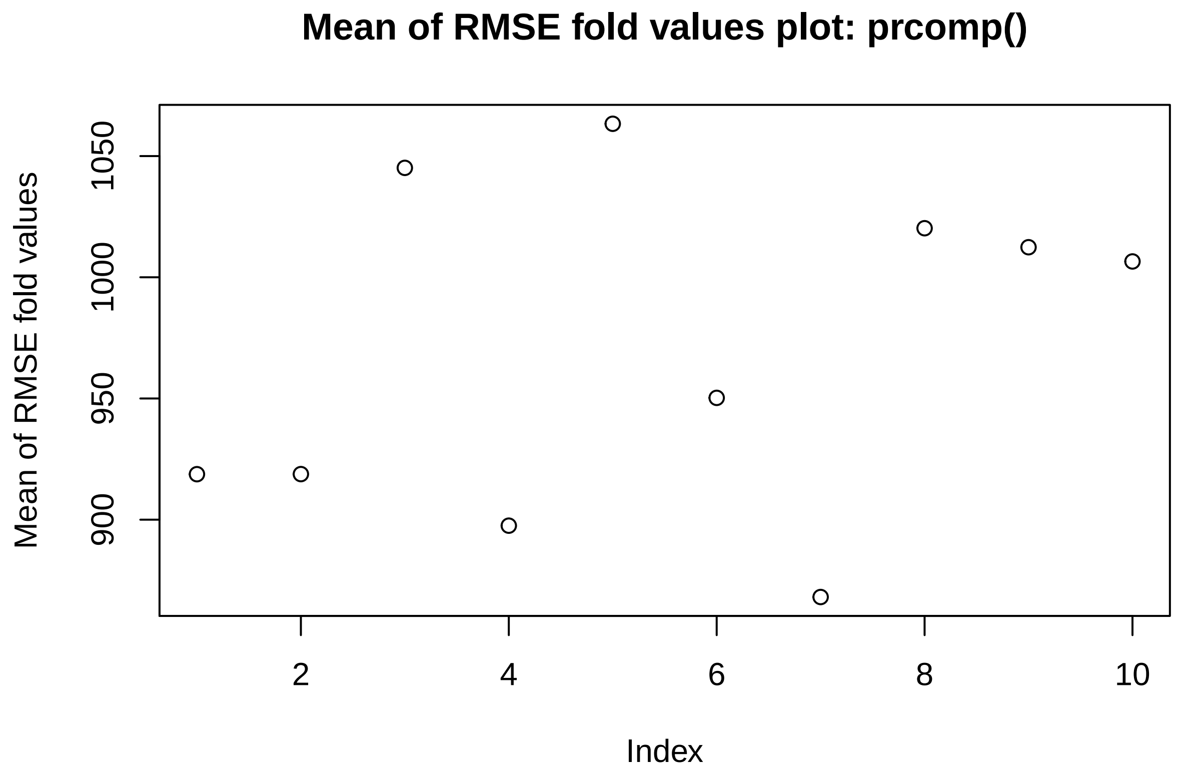
```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
#train splitting the data
bwt_data = birthwt
trn_idx = sample(nrow(bwt_data), size = .8 * nrow(bwt_data))
bwt_trn = bwt_data[trn_idx, ]
bwt_tst = bwt_data[-trn_idx, ]


calc_rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}
```

```
#cross validation of k = 10
set.seed(430)
idx_fold = caret::createFolds(bwt_data$bwt, k = 10)

calc_metric_folds_prcomp = function(fold_idx) {
  est = bwt_data[-fold_idx, ]
  val = bwt_data[fold_idx, ]
  pca_singleVD_est = prcomp(x = est, scale = TRUE)
  PC_1_est = pca_singleVD_est$x[, 1]
  PC_2_est = pca_singleVD_est$x[, 2]
  PC_3_est = pca_singleVD_est$x[, 3]
  PC_4_est = pca_singleVD_est$x[, 4]
  lin_mod = lm(bwt ~ PC_1_est + PC_2_est + PC_3_est + PC_4_est, data = est)
  pred_lin = predict(lin_mod, val)
  calc_rmse(actual = val$bwt, predicted = pred_lin)
}

map_metric_folds_prcomp = invisible(map_dbl(idx_fold, ~calc_metric_folds_prcomp(.x)))

mean(map_metric_folds_prcomp)
```

```
## [1] 970.1079
```

```
plot(map_metric_folds_prcomp, ylab = "Mean of RMSE fold values", main = "Mean of RMSE fold values plot:
```

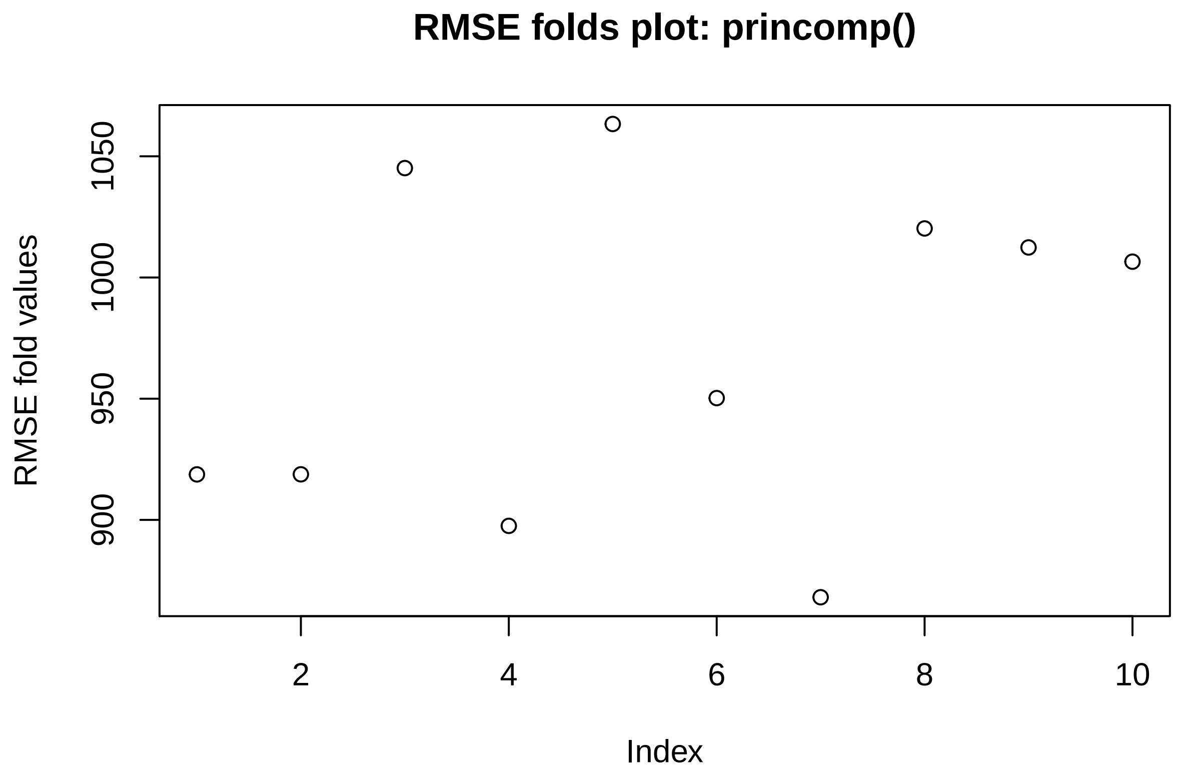


```
set.seed(430)
idx_fold = createFolds(bwt_data$bwt, k =  10)

calc_metric_folds_princomp = function(fold_idx) {
  est = bwt_data[-fold_idx, ]
  val = bwt_data[fold_idx, ]
  pca_spectralVD_est = princomp(est, cor = TRUE, scores = TRUE)
  PC_1_est = pca_spectralVD_est$scores[, 1]
  PC_2_est = pca_spectralVD_est$scores[, 2]
  PC_3_est = pca_spectralVD_est$scores[, 3]
  PC_4_est = pca_spectralVD_est$scores[, 4]
  lin_mod = lm(bwt ~ PC_1_est + PC_2_est + PC_3_est + PC_4_est, data = est)
  pred_lin = predict(lin_mod, val)
  calc_rmse(actual = val$bwt, predicted = pred_lin)
}

map_metric_folds_princomp = invisible(map_dbl(idx_fold, ~calc_metric_folds_princomp(.x)))
```

```
mean(map_metric_folds_princomp)
```

```
## [1] 970.1079
```

```
plot(map_metric_folds_princomp,  ylab = "RMSE fold values", main = "RMSE folds plot: princomp()")
```



When using either prcomp() or princomp() for principle component regression, each corresponding mean of the RMSE values are very close to each other, PCR with prcomp() having an overall mean of RMSE fold values equal to 970.1079485, and PCR with princomp() having an overall mean of RMSE fold values equal to 970.1079485. When ploting the RMSE values for each fold, both the plot for prcomp() and princomp() RMSE fold values look simliar.

Note: When calculating the RMSE, the warning *"'newdata' had 18 rows but variables found have 171 rowslonger object length is not a multiple of shorter object length"* is ocurring because we are doing a k = 10 fold cross validation. It only sees 10 rows when using each fold, expecting 189.

Using PCR Method from caret package

```
set.seed(430)
#using pcr function invokes pls package.
pcr_mod = train(bwt ~ ., data = bwt_trn, method = "pcr", scale = TRUE, trControl = trainControl("cv", nu
summary(pcr_mod)
```

```
## Data:    X dimension: 151 9
##  Y dimension: 151 1
## Fit method: svdpc
## Number of components considered: 3
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps
## X           20.63    36.54    50.67
## .outcome    33.51    33.59    39.75
```

```
pred_pcr_mod = predict(pcr_mod, bwt_tst)

calc_rmse(actual = bwt_tst$bwt, predicted = pred_pcr_mod)
```

```
## [1] 639.7769
```

```
?plsr
```

It seems that among principle component regression models, using the train() function with the pcr method has the best RMSE compared to using linear models for either using prcomp() or princomp().

This is esspecially the case because while the model has a smaller RMSE, it also is using less principle components.

Using Partial Least Squares Method from caret package

```
set.seed(430)
# using the pcr method invokes the pls package
pls_mod = train(bwt ~ ., data = bwt_trn, method = "pls", scale = TRUE,
                trControl = trainControl("cv", number = 10))
summary(pls_mod)
```

```
## Data:    X dimension: 151 9
##  Y dimension: 151 1
## Fit method: oscorespls
## Number of components considered: 3
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps
## X           18.79    30.21    41.08
## .outcome    59.12    67.17    67.87
```

```
pred_pcr_caret = predict(pls_mod, bwt_tst)

calc_rmse(actual = bwt_tst$bwt, predicted = pred_pcr_caret)
```

```
## [1] 476.2847
```

The partial least squears model is the best so far in minimizing RMSE. Partial least squares regresion is related to principal components regression, but the key difference is that PLS finds linear regression models by projecting observable variables and predicted variables onto a another new space. PCR finds hyperplanes of the maximum variance specifically between independent and response variables.

The partial least squares regression is a supervised regression model that finds principal components that are both associated with the predictors and to the response outcome. This is something that principle component regression had issues with, where principle components were not associated with the response variable.

The pls regression is the best because it is only using 3 principle components and the RMSE is the best when comparing to PCR using the train model plus when using linear models.

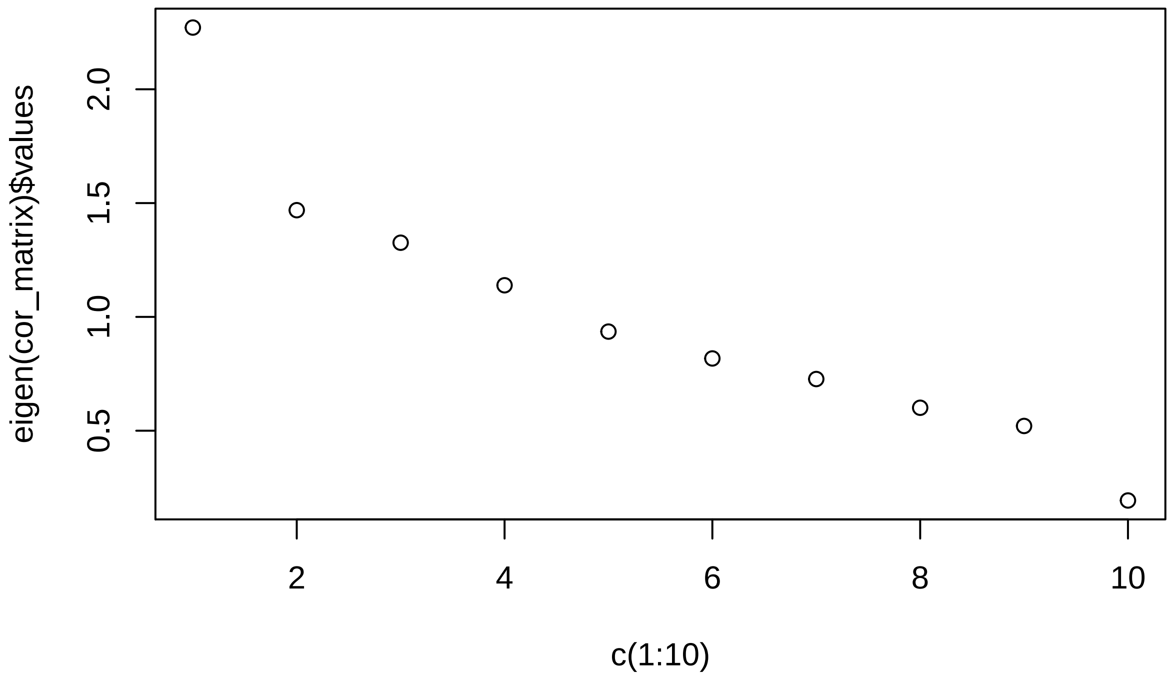
Comparing Principle Component Analysis with Factor Analysis

```
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
```

```
##     %+%, alpha
```

```
library(GPArotation)

#revisit screeplot
plot(c(1:10), eigen(cor_matrix)$values)
```



```
sum(eigen(cor_matrix)$values > 1)
```

```
## [1] 4
```

```
#We are using 4 factors with default settings.
factor_mod = fa(cor_matrix, nfactors = 4)
factor_mod$loadings
```

```
##
## Loadings:
##       MR1    MR2    MR3    MR4
## low    0.803
## age                         0.617
## lwt                  0.518  0.191
## race   0.191 -0.379        -0.246
## smoke         0.995
## ptl    0.207  0.135 -0.236  0.132
## ht     0.192         0.503 -0.123
## ui     0.242        -0.303
## ftv                         0.356
## bwt   -0.964
##
##                  MR1   MR2   MR3   MR4
## SS loadings    1.762 1.156 0.677 0.640
## Proportion Var 0.176 0.116 0.068 0.064
## Cumulative Var 0.176 0.292 0.360 0.424
```

```r
# PCA loadings usign prcomp().
pca_singleVD$rotation[, 1:4]
```

```
##               PC1         PC2         PC3         PC4
## low    0.54562009 -0.12277121  0.25446592 -0.10645094
## age   -0.19597782 -0.36502825  0.01897096 -0.52034869
## lwt   -0.28625204 -0.27252769  0.44602235 -0.03032163
## race   0.19037446  0.57761707  0.17736593 -0.29232302
## smoke  0.17787827 -0.53310202 -0.18971259  0.41911066
## ptl    0.26129590 -0.24600526 -0.28710665 -0.14543364
## ht     0.06394832 -0.13573454  0.65875317  0.19913179
## ui     0.31034930 -0.02165873 -0.31589100 -0.18679793
## ftv   -0.16373631 -0.26177419  0.02842157 -0.58164518
## bwt   -0.56442614  0.10785980 -0.22698204  0.14788695
```

```r
# PCA loading using princomp().
pca_spectralVD$loadings[, 1:4]
```

```
##            Comp.1      Comp.2      Comp.3      Comp.4
## low    0.54562009  0.12277121  0.25446592  0.10645094
## age   -0.19597782  0.36502825  0.01897096  0.52034869
## lwt   -0.28625204  0.27252769  0.44602235  0.03032163
## race   0.19037446 -0.57761707  0.17736593  0.29232302
## smoke  0.17787827  0.53310202 -0.18971259 -0.41911066
## ptl    0.26129590  0.24600526 -0.28710665  0.14543364
## ht     0.06394832  0.13573454  0.65875317 -0.19913179
## ui     0.31034930  0.02165873 -0.31589100  0.18679793
## ftv   -0.16373631  0.26177419  0.02842157  0.58164518
## bwt   -0.56442614 -0.10785980 -0.22698204 -0.14788695
```

Looking at the factor loadings, the first factor variable, it is clear that it seems to be representative mainly from low, and bwt variables. The second one mainly from the smoke variable. The third from the ht and lwt variable. The fourth from the age variable.

Comparing this to the results of the loadings from PCA using prcomp() and princomp() to calculate the Principle Components, thay show a similiar trend. The difference is that the interpretation seems to be stronger since the percentage of the loadings from factor analysis have a greater magnitude. For example, looking at the first principle component from both prcomp() and princomp(), the representation of variance of for the first principle component comes mostly from the bwt and low variables, where both have a value of about .50. In factor analysis, low and bwt have a loading above .80, which shows a much stronger indication that these represent the first factor variable. Although, the fourth factor variable shows a larger difference when compared to PCA. The variable ftv does not represent as strongly in the fourth manifest variable, whereas in the fourth principle component variable, fvt has a stronger representation.

Implementing Varimax rotation for PCA and FA

```r
# Factor analysis with Varimac rotation
factor_mod_varimax = fa(cor_matrix, nfactors = 4, rotate = "varimax")
factor_mod_varimax$loadings
```

```
##
## Loadings:
##       MR1    MR2    MR4    MR3
## low    0.802         -0.128
## age                   0.608
## lwt   -0.143          0.292  0.513
## race    0.158 -0.411 -0.248
## smoke   0.191  0.971 -0.110
## ptl     0.225  0.130        -0.246
## ht      0.180                0.486
## ui      0.254               -0.306
## ftv                   0.356
## bwt    -0.953
##
##                   MR1   MR2   MR4   MR3
## SS loadings    1.784 1.134 0.688 0.666
## Proportion Var 0.178 0.113 0.069 0.067
## Cumulative Var 0.178 0.292 0.361 0.427
```

```
#Varimax rotation for pca using prcomp().

pca_singleVD = prcomp(x = birthwt, scale = TRUE)
sinVD_loadings = pca_singleVD$rotation[, 1:10] %*% diag(pca_singleVD$sdev, 10, 10)
varimax(sinVD_loadings)$loadings[,1:4]
```

```
##              [,1]         [,2]         [,3]        [,4]
## low    0.94808005  0.039435630  0.068338202  0.02206862
## age   -0.05426216 -0.080249655 -0.008638314 -0.10472878
## lwt   -0.10216643 -0.075346172  0.128268504 -0.06698683
## race   0.10404195  0.969470004  0.009489302  0.04409753
## smoke  0.11073930 -0.182663458  0.003800105  0.01391015
## ptl    0.09596211  0.005059957 -0.006090282  0.02003642
## ht     0.09991696  0.009196074  0.984649018  0.03987252
## ui     0.12782269  0.017622883 -0.058942618  0.02488653
## ftv   -0.02453406 -0.041284214 -0.039100682 -0.99014997
## bwt   -0.89855137 -0.108506567 -0.073301340 -0.01606965
```

```
#Varimax rotation for pca using princomp().

pca_spectralVD = princomp(x = birthwt, cor = TRUE, scores = TRUE)
specVD_loadings = pca_spectralVD$loading[, 1:10] %*% diag(pca_spectralVD$sdev, 10, 10)
varimax(sinVD_loadings)$loadings[,1:4]
```

```
##              [,1]         [,2]         [,3]        [,4]
## low    0.94808005  0.039435630  0.068338202  0.02206862
## age   -0.05426216 -0.080249655 -0.008638314 -0.10472878
## lwt   -0.10216643 -0.075346172  0.128268504 -0.06698683
## race   0.10404195  0.969470004  0.009489302  0.04409753
## smoke  0.11073930 -0.182663458  0.003800105  0.01391015
## ptl    0.09596211  0.005059957 -0.006090282  0.02003642
## ht     0.09991696  0.009196074  0.984649018  0.03987252
## ui     0.12782269  0.017622883 -0.058942618  0.02488653
## ftv   -0.02453406 -0.041284214 -0.039100682 -0.99014997
## bwt   -0.89855137 -0.108506567 -0.073301340 -0.01606965
```

When implementing a varimax rotation for both FA and PCA, the loadings for FA does not change, while the loadings from PCA have a drastic change. The variables that show contribution for each specific principle component is the same, but the magnitude of these coeficients are much greater which is stronger evidence of their contribution to the principle component. For example, in the first component, the variable "low" jumped from having a 0.546 to a .948 and the variable "bwt" jumped from a loading value of -.564 to about a -.898.

For PCA followed by a varimax rotation, the second component using both prcomp() and princomp suggest "race" to be the variable representing most of the variance withtin that particular principle component (PC). For the third, "ht" dominates with ~.98. For the fourth, "ftv" dominates with ~.99.

These last three components with a varimax rotation have changed, where now there is mainly one variable in each dominating representation of variance. For example, PC two has loadings of "race" ~.57 and "smoke" ~.53. With varimax, "race" dominates with a ~.96, "smoke" dropping to a ~.11. This occurs for the fourth PC as well, "ftv" jumping from ~-.58 to a ~-.98 and "age" dropping from a ~-.52 to a ~-.10. The change is after using a varimax rotation

Note: Before rotating the factor analysis function fa() to have a varimax rotation, the default rotation is an oblimin rotation.

Summary

Using the factoextra package to plot visualizations to compare the results of PCA using prcomp() and princomp() has been a substantial help. It is esspecially helpful in interpretation of the principle components, making it easier for the viewer to recognize levels of contribution each variable gives, which could help in dimension reduction. Looking at the results for PCA using prcomp() and princomp() demonstrate that they output similiar results, although there are some underlying differences such as certain princple components having different score values as well as visualizations showing that one method is the reflection of the other across the horizontal axis. In the end, they essentially give similiar results, although according to R documentation, prcomp() uses preferred methods for numerical accuracy.

The similiarities of these functions are also present when implementing a principle component regression, where using a linear model, the RMSE is very close. We also found that using a partial least square model actually minimized the RMSE better than using pcr. This could suggest that having principle components associated with the response and feature variables is a helpful method in prediction.

Upon comparing our PCA results with factor analysis, it seems that more interpretation could possibly come from doing a factor analysis using the fa() function, which uses "minimum residual" as the default factoring method. However, implementing a varimax rotation to the loadings in PCA methods drastically changed the loadings, revealing stronger suggestions that certain variables represent the variance more in particular principle components.

REFERENCE SECTION

[R] Q and R mode in Principal Component Analysis. (n.d.). Retrieved from https://stat.ethz.ch/pipermail/r-help/2011-September/289101.html

Kassambara, A. (n.d.). Package 'factoextra.' Retrieved from https://cran.r-project.org/web/packages/factoextra/factoextra.pdf

Wei, T., & Simko, V. (n.d.). Package 'corrplot.' Retrieved from https://cran.r-project.org/web/packages/corrplot/corrplot.pdf

Principal component analysis. (2020, May 14). Retrieved from https://en.wikipedia.org/wiki/Principal_component_analysis

Lesson 6: Principal Components Analysis. (n.d.). Retrieved from https://online.stat.psu.edu/stat508/book/export/html/639

Singular value decomposition. (2020, May 6). Retrieved from https://en.wikipedia.org/wiki/Singular_value_decomposition#SVD_and_spectral_decomposition

Alam, Mike, & Konig. (2017, October 7). Principal Component Analysis in R: prcomp vs princomp. Retrieved from http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/118-principal-component-analysis-in-r-prcomp-vs-princomp/

modLmakurmodLmakur. (1966, February 1). PCA: why do I get so different results from princomp() and prcomp()? Retrieved from https://stackoverflow.com/questions/37349662/pca-why-do-i-get-so-different-results-from-princomp-and-prcomp

prcomp. (n.d.). Retrieved from https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prcomp

princomp. (n.d.). Retrieved from https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/princomp

Jolliffe, I. T., & Cadima, J. (2016, April 13). Principal component analysis: a review and recent developments. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4792409/

Dontas, G. D. G. (1960, December 1). Why do the R functions 'princomp' and 'prcomp' give different eigenvalues? Retrieved from https://stats.stackexchange.com/questions/9500/why-do-the-r-functions-princomp-and-prcomp-give-different-eigenvalues

fviz_pca: Quick Principal Component Analysis data visualization - R software and data mining. (n.d.). Retrieved from http://www.sthda.com/english/wiki/fviz-pca-quick-principal-component-analysis-data-visualization-r-software-and-data-mining

Principal component analysis : the basics you should read - R software and data mining. (n.d.). Retrieved from http://www.sthda.com/english/wiki/wiki.php?id_contents=7896

hans0l0hans0l0 1, DaveDave, ttnphnsttnphns, pengchypengchy, & Yanlong LiYanlong Li. What is the difference between R functions prcomp and princomp? Retrieved from https://stats.stackexchange.com/questions/20101/what-is-the-difference-between-r-functions-prcomp-and-princomp

Kassambara, Visitor, Kassambara, Sfd, & SFer. (2018, March 11). Principal Component and Partial Least Squares Regression Essentials. Retrieved from http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/152-principal-component-and-partial-least-squares-regression-essentials/

Factor analysis. (2020, April 22). Retrieved from https://en.wikipedia.org/wiki/Factor_analysis

R Tutorial Series: Exploratory Factor Analysis. (n.d.). Retrieved from http://rtutorialseries.blogspot.com/2011/10/r-tutorial-series-exploratory-factor.html

ScottScott, amoebaamoeba, F. TusellF. Tusell 6, & Alain DanetAlain Danet. (1963, January 1). How to compute varimax-rotated principal components in R? Retrieved from https://stats.stackexchange.com/questions/59213/how-to-compute-varimax-rotated-principal-components-in-r

Luštrik, L. R. (1960, April 1). Is PCA followed by a rotation (such as varimax) still PCA? Retrieved from https://stats.stackexchange.com/questions/612/is-pca-followed-by-a-rotation-such-as-varimax-still-pca

fa. (n.d.). Retrieved from https://www.rdocumentation.org/packages/psych/versions/1.9.12.31/topics/fa