Principal Components Analysis & Singular Value Decomposition

Nik Bear Brown

In this lesson we apply principal component analysis (PCA) that uses eigenvalues and eigenvectors in an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. We also perform Eigen decomposition of various data matrices into a canonical form, whereby the matrices is represented in terms of their eigenvalues and eigenvectors which allow us to rank and use the most important features in a data set which may make many calculations much quicker and easier.

# Additional packages needed

To run the code in M03\_Lesson\_02.Rmd you may need additional packages.

* If necessary install the following packages.
* install.packages("ggplot2");
* install.packages("reshape2");
* install.packages("psych");

require(ggplot2)

## Loading required package: ggplot2

require(reshape2)

## Loading required package: reshape2

require(psych)

## Loading required package: psych  
##   
## Attaching package: 'psych'  
##   
## The following object is masked from 'package:ggplot2':  
##   
## %+%

# Data

We will be the diamonds dataset as well as creating some random data.

# Load data  
data(diamonds)  
head(diamonds)

## carat cut color clarity depth table price x y z  
## 1 0.23 Ideal E SI2 61.5 55 326 3.95 3.98 2.43  
## 2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31  
## 3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.31  
## 4 0.29 Premium I VS2 62.4 58 334 4.20 4.23 2.63  
## 5 0.31 Good J SI2 63.3 58 335 4.34 4.35 2.75  
## 6 0.24 Very Good J VVS2 62.8 57 336 3.94 3.96 2.48

set.seed(555)  
trails<-333  
m<-33.0  
s<-3.0  
r\_normal <-data.frame(A=rnorm(n=trails,mean=m,sd=sqrt(s)),  
 B=rnorm(n=trails,mean=m,sd=sqrt(s^2)),  
 C=rnorm(n=trails,mean=m,sd=sqrt(s^3)),  
 D=rnorm(n=trails,mean=m,sd=sqrt(s^4)))   
head(r\_normal)

## A B C D  
## 1 32.42867 33.85643 37.31134 31.35921  
## 2 33.87234 34.45005 31.08412 42.42666  
## 3 33.64843 39.80698 36.59153 40.99866  
## 4 36.27119 38.09784 29.63552 39.45627  
## 5 29.91712 38.17761 29.78021 34.92817  
## 6 34.53396 39.51926 35.55332 18.03430

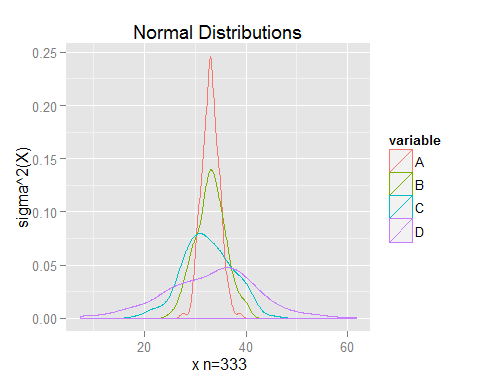
require(reshape2)  
rnd <- melt(data=r\_normal)

## No id variables; using all as measure variables

head(rnd)

## variable value  
## 1 A 32.42867  
## 2 A 33.87234  
## 3 A 33.64843  
## 4 A 36.27119  
## 5 A 29.91712  
## 6 A 34.53396

ggplot(rnd, aes(x=value)) + geom\_density(aes(group=variable,color=variable)) + labs(title="Normal Distributions", y="sigma^2(X)", x=" x n=333")



# Principal Components Analysis in R

### Properties of Principal Components

* The principal components are eigenvectors
* Maximizes variance in order of the components
* They are orthogonal (and form a basis)
* If use all of the principal components we can get lossless reconstruction
* N diminsions to M diminsions (each diminsion has an eigenvalue the order (highest eigenvalues have highest variance) i.e. can "throw away" the lowest eigenvalues. if eigenvalue has 0 value can throw it away without cost.
* We can readjust to origin 0
* There are very fast algorithms to compute PCA

head(diamonds)

## carat cut color clarity depth table price x y z  
## 1 0.23 Ideal E SI2 61.5 55 326 3.95 3.98 2.43  
## 2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31  
## 3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.31  
## 4 0.29 Premium I VS2 62.4 58 334 4.20 4.23 2.63  
## 5 0.31 Good J SI2 63.3 58 335 4.34 4.35 2.75  
## 6 0.24 Very Good J VVS2 62.8 57 336 3.94 3.96 2.48

keep <- c("carat", "depth", "table", "price", "x", "y", "z")  
diamonds<- diamonds[keep]  
head(diamonds)

## carat depth table price x y z  
## 1 0.23 61.5 55 326 3.95 3.98 2.43  
## 2 0.21 59.8 61 326 3.89 3.84 2.31  
## 3 0.23 56.9 65 327 4.05 4.07 2.31  
## 4 0.29 62.4 58 334 4.20 4.23 2.63  
## 5 0.31 63.3 58 335 4.34 4.35 2.75  
## 6 0.24 62.8 57 336 3.94 3.96 2.48

## princomp()

*princomp()* performs a principal components analysis on the given numeric data matrix and returns the results as an object of class princomp.

### Arguments

* formula - a formula with no response variable, referring only to numeric variables.
* data - an optional data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
* subset - an optional vector used to select rows (observations) of the data matrix x.
* na.action - a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit.
* x - a numeric matrix or data frame which provides the data for the principal components analysis.
* cor - a logical value indicating whether the calculation should use the correlation matrix or the covariance matrix. (The correlation matrix can only be used if there are no constant variables.)
* scores - a logical value indicating whether the score on each principal component should be calculated.
* covmat - a covariance matrix, or a covariance list as returned by cov.wt (and cov.mve or cov.mcd from package MASS). If supplied, this is used rather than the covariance matrix of x.

### Value

*princomp()* returns a list with class "princomp" containing the following components:

* sdev - the standard deviations of the principal components.
* loadings - the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). This is of class "loadings": see loadings for its print method.
* center - the means that were subtracted.
* scale - the scalings applied to each variable.
* n.obs - the number of observations.
* scores - if scores = TRUE, the scores of the supplied data on the principal components. These are non-null only if x was supplied, and if covmat was also supplied if it was a covariance list. For the formula method, napredict() is applied to handle the treatment of values omitted by the na.action.
* call - the matched call.

diamonds.fit.A <- princomp(diamonds, cor = TRUE)  
diamonds.fit.A

## Call:  
## princomp(x = diamonds, cor = TRUE)  
##   
## Standard deviations:  
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7   
## 2.1826394 1.1339612 0.8311506 0.4168373 0.2007666 0.1815120 0.1113495   
##   
## 7 variables and 53940 observations.

We can also use a formula.

diamonds.fit.A <- princomp(formula = ~., data = diamonds, cor = TRUE, na.action=na.exclude)  
diamonds.fit.A

## Call:  
## princomp(formula = ~., data = diamonds, na.action = na.exclude,   
## cor = TRUE)  
##   
## Standard deviations:  
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7   
## 2.1826394 1.1339612 0.8311506 0.4168373 0.2007666 0.1815120 0.1113495   
##   
## 7 variables and 53940 observations.

## prcomp()

Performs a principal components analysis on the given data matrix and returns the results as an object of class prcomp.

### Arguments

* formula  
  a formula with no response variable, referring only to numeric variables.
* data - an optional data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
* subset - an optional vector used to select rows (observations) of the data matrix x.
* na.action - a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit.
* retx - a logical value indicating whether the rotated variables should be returned.
* center - a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of x can be supplied. The value is passed to scale.
* scale - a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is FALSE for consistency with S, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of x can be supplied. The value is passed to scale.
* tol - a value indicating the magnitude below which components should be omitted. (Components are omitted if their standard deviations are less than or equal to tol times the standard deviation of the first component.) With the default null setting, no components are omitted. Other settings for tol could be tol = 0 or tol = sqrt(.Machine$double.eps), which would omit essentially constant components.

### Value

*prcomp()* returns a list with class "prcomp" containing the following components:

* sdev - the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix).
* rotation - the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). The function princomp returns this in the element loadings.
* x - if retx is true the value of the rotated data (the centred (and scaled if requested) data multiplied by the rotation matrix) is returned. Hence, cov(x) is the diagonal matrix diag(sdev^2). For the formula method, napredict() is applied to handle the treatment of values omitted by the na.action.
* center, scale - the centering and scaling used, or FALSE.

diamonds.fit.B <- prcomp(diamonds, retx=TRUE, center=TRUE, scale.=TRUE)  
diamonds.fit.B

## Standard deviations:  
## [1] 2.1826394 1.1339612 0.8311506 0.4168373 0.2007666 0.1815120 0.1113495  
##   
## Rotation:  
## PC1 PC2 PC3 PC4 PC5  
## carat 0.4524454941 -0.034696011 0.005494814 -0.06835945 0.13399948  
## depth -0.0009161301 -0.730679714 -0.672829294 -0.04724800 -0.08873829  
## table 0.0995160875 0.675067376 -0.728069469 -0.05954060 -0.01037614  
## price 0.4255192667 -0.035257945 0.105449477 -0.84977817 -0.05377206  
## x 0.4532125054 0.003512550 0.039508824 0.24299509 0.08898016  
## y 0.4472649035 0.002157912 0.054188788 0.32846061 -0.77405793  
## z 0.4459536619 -0.089035176 -0.039603439 0.31700727 0.60339656  
## PC6 PC7  
## carat 0.76815114 0.425880295  
## depth 0.01445027 -0.055600264  
## table -0.02526831 -0.002049255  
## price -0.27330947 -0.082814286  
## x 0.19846061 -0.828658219  
## y -0.21526655 0.208857094  
## z -0.49867040 0.279957944

## principal() library(psych)

### Arguments

* r - a correlation matrix. If a raw data matrix is used, the correlations will be found using pairwise deletions for missing values.
* nfactors - Number of components to extract
* residuals - FALSE, do not show residuals, TRUE, report residuals
* rotate - "none", "varimax", "quatimax", "promax", "oblimin", "simplimax", and "cluster" are possible rotations/transformations of the solution.
* n.obs - Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics.
* covar - If false, find the correlation matrix from the raw data or convert to a correlation matrix if given a square matrix as input.
* scores - If TRUE, find component scores
* missing - if scores are TRUE, and missing=TRUE, then impute missing values using either the median or the mean
* impute - "median" or "mean" values are used to replace missing values
* oblique.scores - If TRUE (default), then the component scores are based upon the structure matrix. If FALSE, upon the pattern matrix.
* method - Which way of finding component scores should be used. The default is "regression"

### Value

* values - Eigen Values of all components – useful for a scree plot
* rotation - which rotation was requested?
* n.obs - number of observations specified or found
* communality - Communality estimates for each item. These are merely the sum of squared factor loadings for that item.
* loadings - A standard loading matrix of class “loadings"
* fit - Fit of the model to the correlation matrix
* fit.off - how well are the off diagonal elements reproduced?
* residual - Residual matrix – if requested
* dof - Degrees of Freedom for this model. This is the number of observed correlations minus the number of independent parameters (number of items \* number of factors - nf*(nf-1)/2. That is, dof = niI*  (ni-1)/2 - ni \* nf + nf\*(nf-1)/2.
* objective - value of the function that is minimized by maximum likelihood procedures. This is reported for comparison purposes and as a way to estimate chi square goodness of fit. The objective function is log(trace ((FF'+U2)^{-1} R) - log(|(FF'+U2)^-1 R|) - n.items. Because components do not minimize the off diagonal, this fit will be not as good as for factor analysis.
* STATISTIC - If the number of observations is specified or found, this is a chi square based upon the objective function, f. Using the formula from factanal: chi^2 = (n.obs - 1 - (2 \* p + 5)/6 - (2 \* factors)/3)) \* f
* PVAL - If n.obs > 0, then what is the probability of observing a chisquare this large or larger?
* phi - If oblique rotations (using oblimin from the GPArotation package) are requested, what is the interfactor correlation.
* scores - If scores=TRUE, then estimates of the factor scores are reported
* weights - The beta weights to find the principal components from the data
* R2 - The multiple R square between the factors and factor score estimates, if they were to be found. (From Grice, 2001) For components, these are of course 1.0.
* valid - The correlations of the component score estimates with the components, if they were to be found and unit weights were used. (So called course coding).

library(psych)  
diamonds.fit.C <- principal(diamonds, nfactors=5, rotate="varimax")  
diamonds.fit.C

## Principal Components Analysis  
## Call: principal(r = diamonds, nfactors = 5, rotate = "varimax")  
## Standardized loadings (pattern matrix) based upon correlation matrix  
## PC1 PC3 PC2 PC4 PC5 h2 u2 com  
## carat 0.98 0.09 0.03 0.13 -0.03 0.98 2.2e-02 1.1  
## depth 0.02 -0.15 0.99 0.00 0.00 1.00 4.5e-05 1.0  
## table 0.10 0.98 -0.15 0.01 0.00 1.00 2.1e-05 1.1  
## price 0.89 0.03 -0.02 0.45 0.00 1.00 2.5e-03 1.5  
## x 0.99 0.09 -0.03 0.00 -0.01 0.99 9.8e-03 1.0  
## y 0.98 0.08 -0.03 -0.03 0.17 1.00 2.1e-03 1.1  
## z 0.98 0.06 0.09 -0.04 -0.11 0.99 9.2e-03 1.1  
##   
## PC1 PC3 PC2 PC4 PC5  
## SS loadings 4.66 1.01 1.01 0.22 0.04  
## Proportion Var 0.67 0.14 0.14 0.03 0.01  
## Cumulative Var 0.67 0.81 0.96 0.99 0.99  
## Proportion Explained 0.67 0.15 0.15 0.03 0.01  
## Cumulative Proportion 0.67 0.82 0.96 0.99 1.00  
##   
## Mean item complexity = 1.1  
## Test of the hypothesis that 5 components are sufficient.  
##   
## The root mean square of the residuals (RMSR) is 0   
## with the empirical chi square 31.16 with prob < NA   
##   
## Fit based upon off diagonal values = 1

## PCA By-Hand in R

We can also calculate the PCA by hand in R

# PCA by hand in R:  
cm <- cor(diamonds)  
# calculate a correlation matrix  
eig <- eigen(cm)  
# eigen find s the eigenvalues and eigenvectors of correlation matrix  
names(eig)

## [1] "values" "vectors"

# The eigenvalues are stored in eig$values  
# The eigenvectors (loadings) are stored in eig$vectors  
sv.ByHand <- sqrt(eig$values)  
# Calculating the singular values from eigenvalues  
loadings.ByHand <- eig$vectors  
rownames(loadings.ByHand) <- colnames(diamonds)  
# saving as loadings, and setting rownames  
# X <- apply(diamonds, MARGIN=2, FUN=standardize)  
# Transforming the data to zero mean and unit variance  
# scores.ByHand <- X %\*% loadings.ByHand  
# calculating scores from eigenanalysis

# How Many Components Should We Extract?

How many components should ee extract from a fit the diamonds.fit.A, diamonds.fit.B, diamonds.fit.C

print(diamonds.fit.A)

## Call:  
## princomp(formula = ~., data = diamonds, na.action = na.exclude,   
## cor = TRUE)  
##   
## Standard deviations:  
## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7   
## 2.1826394 1.1339612 0.8311506 0.4168373 0.2007666 0.1815120 0.1113495   
##   
## 7 variables and 53940 observations.

summary(diamonds.fit.A)

## Importance of components:  
## Comp.1 Comp.2 Comp.3 Comp.4  
## Standard deviation 2.1826394 1.1339612 0.83115057 0.4168373  
## Proportion of Variance 0.6805593 0.1836954 0.09868732 0.0248219  
## Cumulative Proportion 0.6805593 0.8642547 0.96294202 0.9877639  
## Comp.5 Comp.6 Comp.7  
## Standard deviation 0.200766577 0.181511957 0.111349506  
## Proportion of Variance 0.005758174 0.004706656 0.001771245  
## Cumulative Proportion 0.993522100 0.998228755 1.000000000

names(diamonds.fit.A)

## [1] "sdev" "loadings" "center" "scale" "n.obs" "scores"   
## [7] "call"

print(diamonds.fit.B)

## Standard deviations:  
## [1] 2.1826394 1.1339612 0.8311506 0.4168373 0.2007666 0.1815120 0.1113495  
##   
## Rotation:  
## PC1 PC2 PC3 PC4 PC5  
## carat 0.4524454941 -0.034696011 0.005494814 -0.06835945 0.13399948  
## depth -0.0009161301 -0.730679714 -0.672829294 -0.04724800 -0.08873829  
## table 0.0995160875 0.675067376 -0.728069469 -0.05954060 -0.01037614  
## price 0.4255192667 -0.035257945 0.105449477 -0.84977817 -0.05377206  
## x 0.4532125054 0.003512550 0.039508824 0.24299509 0.08898016  
## y 0.4472649035 0.002157912 0.054188788 0.32846061 -0.77405793  
## z 0.4459536619 -0.089035176 -0.039603439 0.31700727 0.60339656  
## PC6 PC7  
## carat 0.76815114 0.425880295  
## depth 0.01445027 -0.055600264  
## table -0.02526831 -0.002049255  
## price -0.27330947 -0.082814286  
## x 0.19846061 -0.828658219  
## y -0.21526655 0.208857094  
## z -0.49867040 0.279957944

summary(diamonds.fit.B)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6  
## Standard deviation 2.1826 1.1340 0.83115 0.41684 0.20077 0.18151  
## Proportion of Variance 0.6806 0.1837 0.09869 0.02482 0.00576 0.00471  
## Cumulative Proportion 0.6806 0.8642 0.96294 0.98776 0.99352 0.99823  
## PC7  
## Standard deviation 0.11135  
## Proportion of Variance 0.00177  
## Cumulative Proportion 1.00000

names(diamonds.fit.B)

## [1] "sdev" "rotation" "center" "scale" "x"

print(diamonds.fit.C)

## Principal Components Analysis  
## Call: principal(r = diamonds, nfactors = 5, rotate = "varimax")  
## Standardized loadings (pattern matrix) based upon correlation matrix  
## PC1 PC3 PC2 PC4 PC5 h2 u2 com  
## carat 0.98 0.09 0.03 0.13 -0.03 0.98 2.2e-02 1.1  
## depth 0.02 -0.15 0.99 0.00 0.00 1.00 4.5e-05 1.0  
## table 0.10 0.98 -0.15 0.01 0.00 1.00 2.1e-05 1.1  
## price 0.89 0.03 -0.02 0.45 0.00 1.00 2.5e-03 1.5  
## x 0.99 0.09 -0.03 0.00 -0.01 0.99 9.8e-03 1.0  
## y 0.98 0.08 -0.03 -0.03 0.17 1.00 2.1e-03 1.1  
## z 0.98 0.06 0.09 -0.04 -0.11 0.99 9.2e-03 1.1  
##   
## PC1 PC3 PC2 PC4 PC5  
## SS loadings 4.66 1.01 1.01 0.22 0.04  
## Proportion Var 0.67 0.14 0.14 0.03 0.01  
## Cumulative Var 0.67 0.81 0.96 0.99 0.99  
## Proportion Explained 0.67 0.15 0.15 0.03 0.01  
## Cumulative Proportion 0.67 0.82 0.96 0.99 1.00  
##   
## Mean item complexity = 1.1  
## Test of the hypothesis that 5 components are sufficient.  
##   
## The root mean square of the residuals (RMSR) is 0   
## with the empirical chi square 31.16 with prob < NA   
##   
## Fit based upon off diagonal values = 1

summary(diamonds.fit.C)

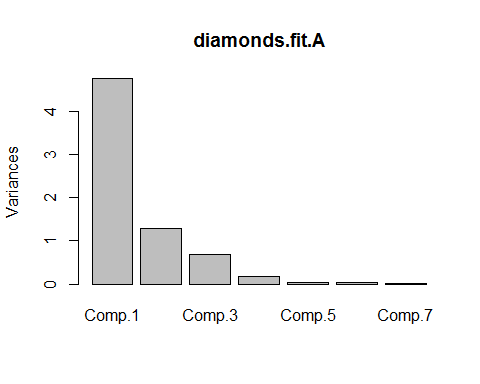
##   
## Factor analysis with Call: principal(r = diamonds, nfactors = 5, rotate = "varimax")  
##   
## Test of the hypothesis that 5 factors are sufficient.  
## The degrees of freedom for the model is -4 and the objective function was 0.61   
## The number of observations was 53940 with Chi Square = 32932.93 with prob < NA   
##   
## The root mean square of the residuals (RMSA) is 0

names(diamonds.fit.C)

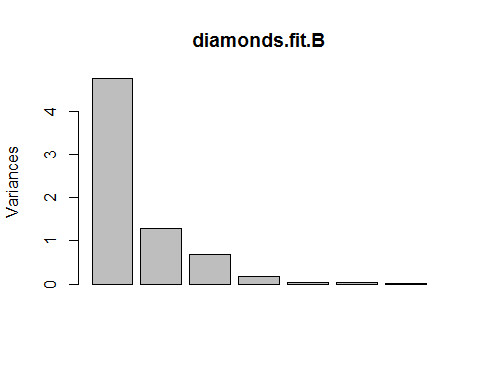
## [1] "values" "rotation" "n.obs" "communality"   
## [5] "loadings" "fit" "fit.off" "fn"   
## [9] "Call" "uniquenesses" "complexity" "chi"   
## [13] "EPVAL" "R2" "objective" "residual"   
## [17] "rms" "factors" "dof" "null.dof"   
## [21] "null.model" "criteria" "STATISTIC" "PVAL"   
## [25] "weights" "r.scores" "Structure" "scores"

The default plot method for classes "princomp" and "prcomp" is a screeplot.

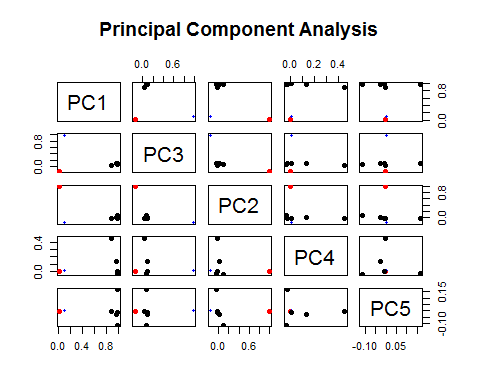
plot(diamonds.fit.A)



plot(diamonds.fit.B)

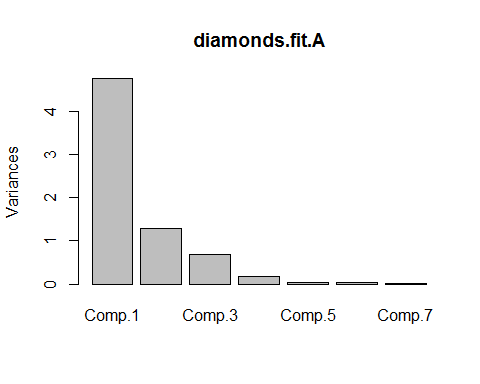


plot(diamonds.fit.C)

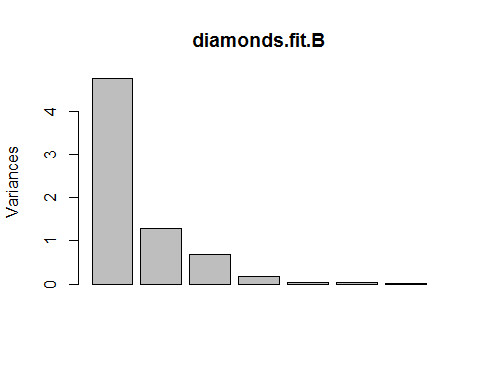


Screeplots plots variances against the number of the principal component.

screeplot(diamonds.fit.A)



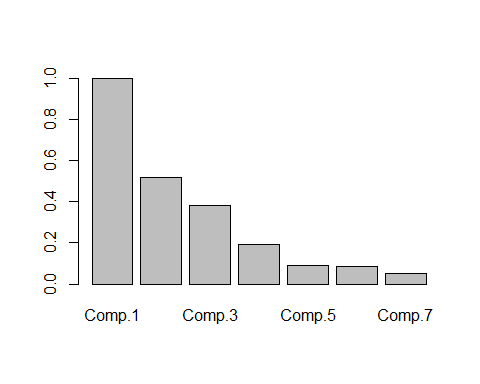
screeplot(diamonds.fit.B)



# screeplot(diamonds.fit.C)

If the first principal component is large we can scale the principal component's by the first principal component.

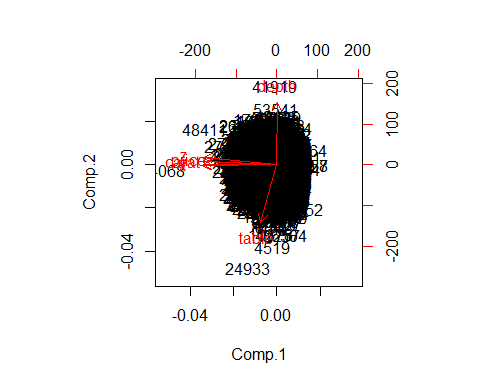
barplot(diamonds.fit.A$sdev/diamonds.fit.A$sdev[1])



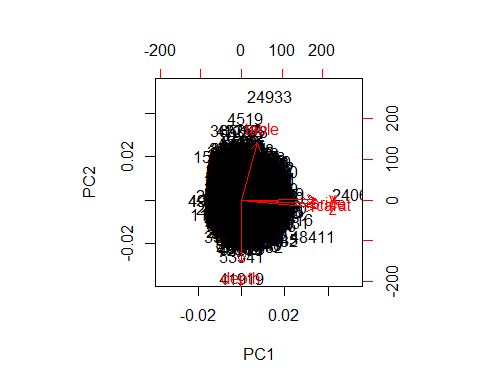
### Biplots

Evidence of clustering in the data can be visualized by looking scores of the first principal component against the scores of the second, scores of the second principal component against the scores of the third, etc. These are called biplots of the principal component's.

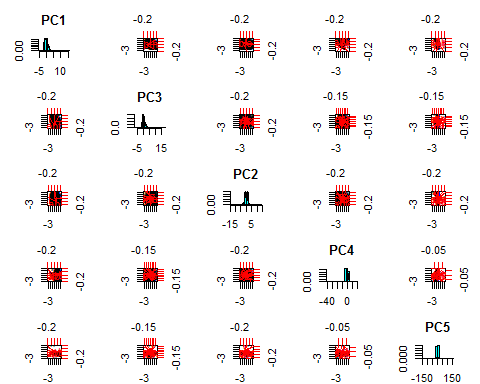
biplot(diamonds.fit.A)



biplot(diamonds.fit.B)



biplot(diamonds.fit.C)



# Singular Value Decomposition in R

[singular value decomposition (SVD)](https://en.wikipedia.org/wiki/Singular_value_decomposition) is a factorization of a real or complex matrix. A a matrix decomposition or matrix factorization is a factorization of a matrix into a product of matrices.

If is a m × n matrix whose entries come from , then the SVD a factorization, called a singular value decomposition of , of the form gives the factors in columns of .

Note that,  
 is a m × n diagonal matrix with non-negative real numbers on the diagonal, and  
 is an m × m, and is an n × n, unitary matrix over .

The diagonal entries, , of are known as the singular values of . The singular values are usually listed in descending order.

Note that [Eigendecomposition](https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix) is a form of matrix factorization.

, where D is a diagonal matrix formed from the eigenvalues of A, and the columns of V are the corresponding eigenvectors of A.

*svd()* computes the singular-value decomposition of a rectangular matrix.

### Arguments

* x - a numeric or complex matrix whose SVD decomposition is to be computed. Logical matrices are coerced to numeric.
* nu - the number of left singular vectors to be computed. This must between 0 and n = nrow(x).
* nv - the number of right singular vectors to be computed. This must be between 0 and p = ncol(x).

### Value

The returned value is a list with components

* d - a vector containing the singular values of x, of length min(n, p).
* u - a matrix whose columns contain the left singular vectors of x, present if nu > 0. Dimension c(n, nu).
* v - a matrix whose columns contain the right singular vectors of x, present if nv > 0. Dimension c(p, nv).

diamonds.svd <- svd(diamonds)  
names(diamonds.svd)

## [1] "d" "u" "v"

diamonds.svd$d

## [1] 1.301135e+06 1.397768e+04 4.946712e+02 1.905483e+02 4.633127e+01  
## [6] 3.049019e+01 2.029887e+01

A <- diag(diamonds.svd$d)  
A

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 1301135 0.00 0.0000 0.0000 0.00000 0.00000 0.00000  
## [2,] 0 13977.68 0.0000 0.0000 0.00000 0.00000 0.00000  
## [3,] 0 0.00 494.6712 0.0000 0.00000 0.00000 0.00000  
## [4,] 0 0.00 0.0000 190.5483 0.00000 0.00000 0.00000  
## [5,] 0 0.00 0.0000 0.0000 46.33127 0.00000 0.00000  
## [6,] 0 0.00 0.0000 0.0000 0.00000 30.49019 0.00000  
## [7,] 0 0.00 0.0000 0.0000 0.00000 0.00000 20.29887

u<-as.matrix(diamonds.svd$u[,1:2])  
v<-as.matrix(diamonds.svd$v[,1:2])  
d<-as.matrix(diag(diamonds.svd$d)[1:2, 1:2])  
s2<-u%\*%d%\*%t(v)  
head(s2)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0.3966085 60.42653 56.00373 325.9995 4.733347 4.739562 2.927893  
## [2,] 0.4088509 62.47590 57.90233 325.9996 4.891129 4.897564 3.025510  
## [3,] 0.4116267 62.92226 58.31592 326.9999 4.925743 4.932226 3.046923  
## [4,] 0.4093728 62.41687 57.84820 333.9996 4.888574 4.894996 3.023916  
## [5,] 0.4124356 62.91128 58.30630 334.9996 4.926887 4.933361 3.047618  
## [6,] 0.4077572 62.10983 57.56388 335.9994 4.865431 4.871819 3.009595

diamonds.svd$d

## [1] 1.301135e+06 1.397768e+04 4.946712e+02 1.905483e+02 4.633127e+01  
## [6] 3.049019e+01 2.029887e+01

# Assingment

* Download the compressed data from the U.S. [Bureau of Labor Statistics](https://en.wikipedia.org/wiki/Bureau_of_Labor_Statistics) [<http://www.bls.gov/>](http://www.bls.gov/) @ <http://www.bls.gov/cew/data/files/2014/csv/2014_annual_singlefile.zip>, and extract the .csv file.
* Run Principal Components Analysis on the BLS data and answer the following questions. (You can use any PCA function you wish, i.e. princomp(), prcomp(), principal() or by hand.)
* Questions:

1. What proportion of the total variation in the data is explained by each of the principal components?
2. Plot a screeplot.
3. Based on the variation explained for each of these components, which, if any, components would you use?
4. Is there evidence of clustering in the data by creating biplots of the each of the components plotted against one another?
5. Do any of the biplots reveal any interesting structure?
6. How many pcs are required to explain 75% of the variance in the data?

Write up your report as an .Rmd file.

# Resources

{Computing and visualizing PCA in R] (<http://www.r-bloggers.com/computing-and-visualizing-pca-in-r/>)