PCA, Singular Values and Eigenvalues

Rei Sanchez-Arias, Ph.D.

PCA, SVD and the eigendecomposition of the covariance matrix

Pre-requisites

Checklist

☑ Load the `tidyverse` package

```
library(tidyverse)
```

To perform Principal Component Analysis (PCA) we use the function `prcomp()` from the `stats` package (you don't need to install any package to use it, since it comes with your R installation)

☑ For data visualization you will be using the `factoextra` package

```
library(factoextra)
```

☑ For data preprocessing we use the `caret` package

```
library(caret)
```

Example: 1974 Motor Trend US Magazine

`mtcars` dataset

We use data from Motor Trend Car Road Tests, with auto design and performance for 32 automobiles (1973 - 1974 models). This dataset has $n = 32$ observations (rows), and we focus here on $m = 6$ numerical attributes (columns)

```
mycars <- mtcars %>%
  select(-c("cyl", "vs", "am",
            "gear" , "carb"))
```

Let us save the data in a matrix $\mathbf{X}$:

```
X <- as.matrix(mycars)
```

	mpg	disp	hp	drat	wt	qsec
Mazda RX4	21	160	110	3.9	2.62	16.46
Mazda RX4 Wag	21	160	110	3.9	2.875	17.02
Datsun 710	22.8	108	93	3.85	2.32	18.61
Hornet 4 Drive	21.4	258	110	3.08	3.215	19.44
Hornet Sportabout	18.7	360	175	3.15	3.44	17.02
Valiant	18.1	225	105	2.76	3.46	20.22
Duster 360	14.3	360	245	3.21	3.57	15.84
Merc 240D	24.4	146.7	62	3.69	3.19	20

Previous 1 2 3 4 Next

PCA calculation

PCA using `prcomp()`

How does `prcomp()` calculates the principal components? It uses a singular value decomposition of the (centered and possibly scaled) data matrix, *not* by using the eigendecomposition of the covariance matrix.

```r
# perform PCA with standardized data
cars_pca <- prcomp(mycars, scale. = T)
summary(cars_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4    PC5     PC6
## Standard deviation     2.0463 1.0715 0.57737 0.39289 0.3533 0.22799
## Proportion of Variance 0.6979 0.1913 0.05556 0.02573 0.0208 0.00866
## Cumulative Proportion  0.6979 0.8892 0.94481 0.97054 0.9913 1.00000
```

Proportion of Variance

How to recreate the summary above? First, identify the variances:

```
variances <- cars_pca$sdev^2
```

Second, compute the proportion of variance explained by each:

```
variances / sum(variances)
```

```
## [1] 0.697899413 0.191352020 0.055559444 0.025726757 0.020799335 0.008663031
```

Third, get the cumulative proportion of variance explained:

```
cumsum(variances) / sum(variances)
```

```
## [1] 0.6978994 0.8892514 0.9448109 0.9705376 0.9913370 1.0000000
```

Covariance Matrix

Covariance Matrix Calculation

To calculate the covariance matrix, we can use the `cov()` function. We will use a **standardized version** of the data matrix. To pre-process the data we could use the `caret` package

```
# define pre-processing scheme
my_prep <- preProcess(mycars, method = c("center", "scale"))
# standardized data
X_c <- predict(my_prep, mycars)
# save as a matrix
X_c <- as.matrix(X_c)
```

Covariance Matrix Calculation (2)

```r
M <- cov(X_c)
M
```

```
##              mpg       disp         hp   0.(
## mpg    1.0000000 -0.8475514 -0.7761684   0.(
## disp  -0.8475514  1.0000000  0.7909486  -0.
## hp    -0.7761684  0.7909486  1.0000000  -0.
## drat   0.6811719 -0.7102139 -0.4487591   1.(
## wt    -0.8676594  0.8879799  0.6587479  -0.
## qsec   0.4186840 -0.4336979 -0.7082234   0.(
```

```r
n <- nrow(mycars)
1/(n-1) * (t(X_c) %*% (X_c) )
```

```
##              mpg       disp         hp
## mpg    1.0000000 -0.8475514 -0.7761684   0.(
## disp  -0.8475514  1.0000000  0.7909486  -0.
## hp    -0.7761684  0.7909486  1.0000000  -0.
## drat   0.6811719 -0.7102139 -0.4487591   1.(
## wt    -0.8676594  0.8879799  0.6587479  -0.
## qsec   0.4186840 -0.4336979 -0.7082234   0.(
```

You can also calculate the covariance matrix by computing: $\frac{1}{n-1}\mathbf{X_c}^T\mathbf{X_c}$, where $\mathbf{X}_c$ is the *standardized* version of your matrix. n is the number of rows of $\mathbf{X}$, i.e. the number of observations, and the denominator $n-1$ is used which gives an unbiased estimator of the (co)variance for i.i.d. observations

Singular Value Decomposition

Singular Value Decomposition (SVD)

The SVD factorization allows to represent a matrix $\mathbf{X}$ as $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The matrix $\mathbf{\Sigma}$ contains the **singular values** in its diagonal, and both $\mathbf{U}^T\mathbf{U}$ and $\mathbf{V}^T\mathbf{V}$ are the identity matrix (of the appropriate size). U and V are matrices whose columns form an orthonormal set.

```
# perform SVD of matrix
cars_svd <- svd(X_c)
```

```
str(cars_svd)
```

```
## List of 3
##  $ d: num [1:6] 11.39 5.97 3.21 2.19 1.97 ...
##  $ u: num [1:32, 1:6] -0.07395 -0.07087 -0.1479 -0.00846 0.11336 ...
##  $ v: num [1:6, 1:6] -0.459 0.466 0.426 -0.367 0.439 ...
```

σ_i values, $\mathbf{U}$ and $\mathbf{V}$

Singular values are given by:

```
# singular values
cars_svd$d
```

```
## [1] 11.393388  5.965859  3.214663  2.187505  1.966895  1.269379
```

```
U <- cars_svd$u
dim(U)
```

```
## [1] 32  6
```

```
# inner product between two columns of U
t(U[ , 2]) %*% U[ , 4]
```

```
##              [,1]
## [1,] 3.404395e-16
```

```
V <- cars_svd$v
dim(V)
```

```
## [1] 6 6
```

```
# inner product between two columns of V
t(V[ , 3]) %*% V[ , 1]
```

```
##              [,1]
## [1,] 1.387779e-17
```

Eigenvalue Decomposition

$\mathbf{M}\mathbf{w} = \lambda\mathbf{w}$

In this section we compute the eigendecomposition of the covariance matrix $\mathbf{M}$ (notice this is a square matrix). We can get the eigendecomposition $\mathbf{M}$ using `eigen()`:

```r
# compute eigendecomposition
cars_eigen <- eigen(M)
```

```r
# matrix with eigenvectors
W <- cars_eigen$vectors
```

With the eigendecomposition, $\mathbf{M}\mathbf{w}_i = \lambda_i \mathbf{w}_i$ where $i = 1, \ldots, m$ and the (eigenvectors) $\mathbf{w}_i$ are the columns of $\mathbf{W}$. If $\mathbf{\Lambda} = \mathrm{diag}(\lambda_i)$, then $\mathbf{M}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}$

The loadings (rotations) matrix is $\mathbf{W}_{m \times m}$ coming from the eigendecompositon of $\mathbf{M}$, and the scores matrix is $\mathbf{T}$. They are related by $\mathbf{T}_{n \times m} = \mathbf{X}_c \mathbf{W}_{m \times m}$

Connection with the SVD

It turns out that every column of $\mathbf{W}$ is (up to a chance in sign) identical to every column of the matrix $\mathbf{V}$ in the SVD of the *standardized* version of the data $\mathbf{X}_c$

```
W
```

```
##                [,1]        [,2]        [,3]
## [1,]  0.4586835 -0.05867609 -0.19479235 0
## [2,] -0.4660354  0.06065296  0.09688406 0
## [3,] -0.4258534 -0.36147576  0.14613554 0
## [4,]  0.3670963 -0.43652537  0.80049152 0
## [5,] -0.4386179  0.29953457  0.41776208 0
## [6,]  0.2528320  0.76284877  0.34059066 0
```

```
V
```

```
##                [,1]        [,2]        [,3]
## [1,] -0.4586835  0.05867609 -0.19479235 0
## [2,]  0.4660354 -0.06065296  0.09688406 0
## [3,]  0.4258534  0.36147576  0.14613554 0
## [4,] -0.3670963  0.43652537  0.80049152 0
## [5,]  0.4386179 -0.29953457  0.41776208 0
## [6,] -0.2528320 -0.76284877  0.34059066 0
```

- From the eigendecomposition: $\mathbf{T}_{n\times m} = \mathbf{X}_c\mathbf{W}_{m\times m}$
- From the SVD factorization: $\mathbf{X}_c = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$

Since $\mathbf{W} = \mathbf{V}$

NOTICE: $\mathbf{T}_{n\times m} = \mathbf{X}_c\mathbf{W} = \mathbf{X}_c\mathbf{V} = \left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\right)\mathbf{V} = \mathbf{U}\boldsymbol{\Sigma}\left(\mathbf{V}^T\mathbf{V}\right) = \mathbf{U}\boldsymbol{\Sigma},$

$$
\begin{aligned}
\mathbf{W}\boldsymbol{\Lambda} = \mathbf{M}\mathbf{W} &= \frac{1}{n-1}\mathbf{X}^T\mathbf{X}\mathbf{V} = \frac{1}{n-1}\mathbf{X}^T\left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\right)\mathbf{V} \\
&= \frac{1}{n-1}\mathbf{X}^T\mathbf{U}\boldsymbol{\Sigma} = \frac{1}{n-1}\left(\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^T\right)\mathbf{U}\boldsymbol{\Sigma} = \frac{1}{n-1}\mathbf{V}\boldsymbol{\Sigma}^2
\end{aligned}
$$

We get a relationship between the **eigenvalues** of the covariance matrix and the **singular values** of the data matrix: $\lambda_i = \frac{\sigma_i^2}{n-1}$

Eigenvalues and Singular Values

```
cars_eigen$values
```

```
## [1] 4.18739648 1.14811212 0.33335666 0.15436054 0.12479601 0.05197818
```

```
(cars_svd$d^2)/(n-1)
```

```
## [1] 4.18739648 1.14811212 0.33335666 0.15436054 0.12479601 0.05197818
```

Notice that the values above, match the values found earlier, and stored as `variances <- cars_pca$sdev^2`:

```
variances
```

```
## [1] 4.18739648 1.14811212 0.33335666 0.15436054 0.12479601 0.05197818
```

Proportion of Variance Explained

Proportion of Variance Explained

Recall, the proportion of variance explained shown in the summary

```
summary(cars_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4    PC5     PC6
## Standard deviation     2.0463 1.0715 0.57737 0.39289 0.3533 0.22799
## Proportion of Variance 0.6979 0.1913 0.05556 0.02573 0.0208 0.00866
## Cumulative Proportion  0.6979 0.8892 0.94481 0.97054 0.9913 1.00000
```

$$\text{Proportion of variance captured by ith PC} = \frac{\lambda_i}{\sum_i \lambda_i} = \frac{\sigma_i^2}{\sum_i \sigma_i^2}$$

where λ_i is the ith **eigenvalue** from the eigenvalue decomposition of the covariance matrix, and σ_i is the ith **singular value** from the SVD factorization of the data matrix.

Principal Components

Principal Components - Where are they?

The columns of $\mathbf{W}$ (and therefore the columns of $\mathbf{V}$) represent the principal components (up to a scalar). We confirm this by checking the 1st column in each case:

```
W[ , 1]       # first column of W
```

```
## [1]  0.4586835 -0.4660354 -0.4258534  0.3670963 -0.4386179  0.2528320
```

```
V[ , 1]       # first column of V
```

```
## [1] -0.4586835  0.4660354  0.4258534 -0.3670963  0.4386179 -0.2528320
```

```
# first column from prcomp() rotation component
cars_pca$rotation[ , 1]
```

```
##        mpg       disp         hp       drat         wt       qsec
## -0.4586835  0.4660354  0.4258534 -0.3670963  0.4386179 -0.2528320
```

Projections

Projections (Scores)

`prcomp()` returns the principal components (PCs) stored in the element `.$rotation`, with elements of each column representing the **loadings** for each PC. Similarly, the **scores** are returned in the element `.$x`, and the matrix of scores provides the projection of each data point in the different PCs.

	PC1	PC2	PC3	PC4	PC5	PC6
Mazda RX4	-0.84258	0.87347	-0.22828	-0.37427	0.51523	-0.05294
Mazda RX4 Wag	-0.8075	0.55634	-0.01267	-0.33369	0.443	-0.15771
Datsun 710	-1.68504	-0.04001	-0.15649	-0.40572	-0.0334	0.10756
Hornet 4 Drive	-0.09644	-1.29438	-0.57023	0.25208	-0.04326	0.18173

Previous 1 2 3 4 5 ... 8 Next

Projections (using eigendecomposition)

The scores matrix $\mathbf{T}$ which satisfies $\mathbf{T} = \mathbf{X}\mathbf{W}$, with $\mathbf{W}$ the matrix of eigenvectors of the covariance matrix, also satisfies:

$\mathbf{T} = \mathbf{X}\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V} = \mathbf{U}\mathbf{\Sigma}$ `= U %*% diag(cars_svd\$d)`

```
U %*% diag(cars_svd$d)
```

V1	V2	V3	V4	V5	V6
-0.84258	0.87347	-0.22828	-0.37427	0.51523	-0.05294
-0.8075	0.55634	-0.01267	-0.33369	0.443	-0.15771
-1.68504	-0.04001	-0.15649	-0.40572	-0.0334	0.10756
-0.09644	-1.29438	-0.57023	0.25208	-0.04326	0.18173

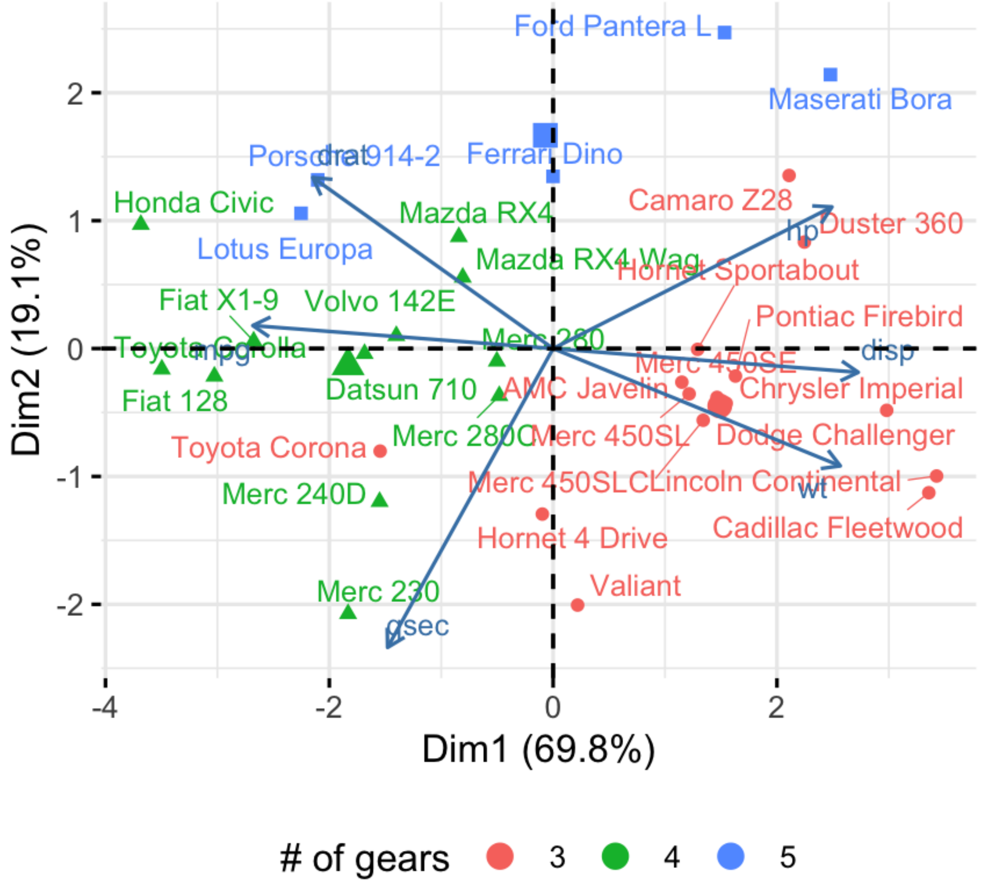
Previous 1 2 3 4 5 ... 8 Next

Visualization

Biplot

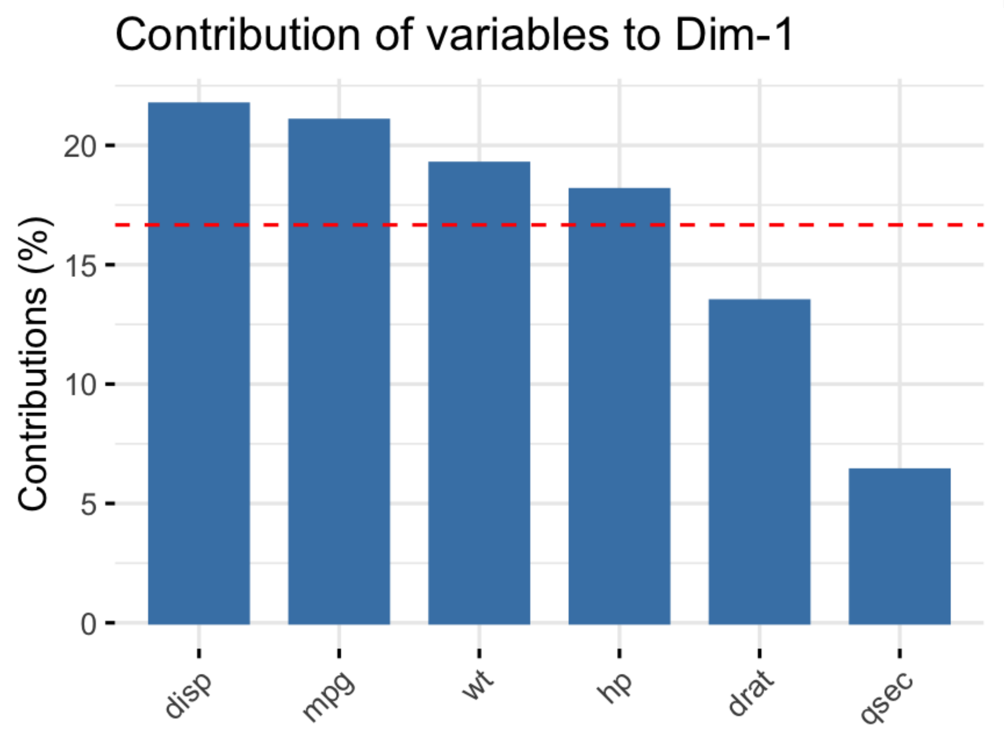
Biplots can be generated using `factoextra::fviz_pca()`.

```
fviz_pca(cars_pca,
         geom.ind = c("point", "text"),
         repel = TRUE,
         label = c("var", "ind"),
         habillage = factor(mtcars$gear)
)
```



Contributions to principal components

```r
fviz_contrib(cars_pca, choice = "var",
             axes = 1)
```




```r
fviz_contrib(cars_pca, choice = "var",
             axes = 2)
```