# Processing and Analysis of Biological Data

A primer of multivariate analysis

Øystein H. Opedal

2 Dec 2022

## Introduction

Biological data are very often multivariate, in the sense that we are dealing with potentially large sets of correlated variables. Some times the patterns of covariation among variables is the focus of the investigation, and other times we may have taken several measurements that describe similar aspects of the biology in question. In any case, we very often have to deal with analyses of multiple variables. Note that when we refer to multivariate analyses, we mean analyses where there is more than one response variable. Thus, a multiple regression is normally considered a univariate analysis even though there are multiple correlated predictor variables.

## Variance matrices and eigendecomposition

Multivariate data can be summarized as *variance matrices*, which are symmetrical matrices with variances on the diagonal and covariances on the off-diagonals. We will work with the following example variance matrix **CM**.

```
cm = matrix(c(0.7, 0.2, -0.3,
              0.2, 1.2, 0.4,
              -0.3, 0.4, 0.6),
              nrow=3)
cm
```

```
##      [,1] [,2] [,3]
## [1,]  0.7  0.2 -0.3
## [2,]  0.2  1.2  0.4
## [3,] -0.3  0.4  0.6
```

EXERCISE: One way to confirm that a matrix is symmetrical is to show that it is identical to it´s transpose ($\mathbf{A} = \mathbf{A}^T$). Confirm that **CM** is symmetrical. To transpose a matrix in R we can use the `t` function.

```
##      [,1] [,2] [,3]
## [1,] TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE
```

Correlations are standardized covariances, i.e. they are the covariances between standardized variables (mean = 0, sd = 1). We can compute the correlation between two variables as

$Cor(x, y) = Cov(x, y)/\sqrt{Var(x)Var(y)}$.

EXERCISE: Translate the covariance matrix into a correlation matrix.

```
##            [,1]      [,2]       [,3]
## [1,]  1.0000000 0.2182179 -0.4629100
## [2,]  0.2182179 1.0000000  0.4714045
## [3,] -0.4629100 0.4714045  1.0000000
```

### The multivariate normal distribution

We can use a variance matrix to simulate data from the multivariate normal distribution $MVN(\bar{x}, \Sigma)$, where $\Sigma$ is a variance matrix.

```
library(MASS)
library(ellipse)
```

```
## Warning: package 'ellipse' was built under R version 4.2.2
```

```
##
## Attaching package: 'ellipse'
```

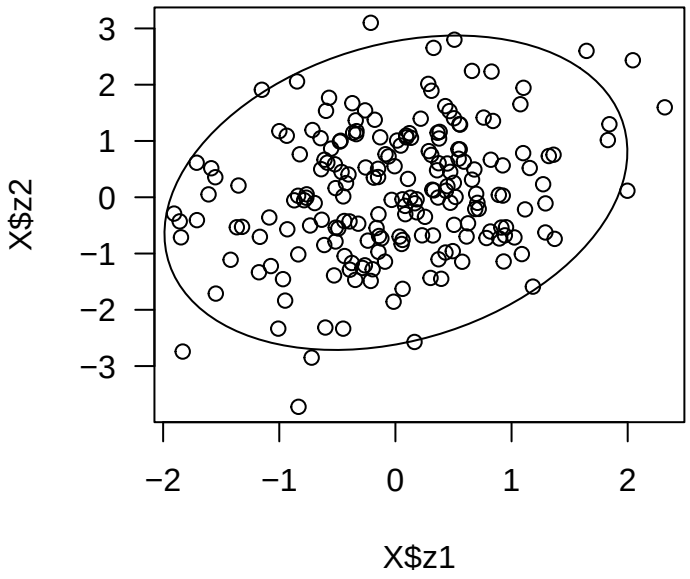
```
## The following object is masked from 'package:graphics':
##
##     pairs
```

```
X = data.frame(mvrnorm(200, mu=c(0,0,0), Sigma=cm))
colnames(X) = c("z1", "z2", "z3")
head(X)
```

```
##            z1          z2          z3
## 1  0.70631474 -0.09907062 -0.51868438
## 2 -0.47178431  1.00383052  0.76322321
## 3  0.37222333  0.60428078  0.23164847
## 4 -0.93094839 -0.56925207 -0.10574330
## 5  0.06113457 -0.75849360  0.06500311
## 6  0.25449990 -0.34791547  0.18138033
```

```
means = c(apply(X[,1:2], 2, mean))

plot(X$z1, X$z2, las=1)
lines(ellipse(cov(X[,1:2]), centre=means))
```




## Eigendecomposition

Variance matrices have several useful properties for multivariate analysis. A common operation is a so-called eigendecomposition (or spectral decomposition).

A vector $v$ is an eigenvector of the matrix $\mathbf{A}$ if it satisfies the condition

$\mathbf{Av} = \lambda\mathbf{v}$,

where $\lambda$ is an eigenvalue of $\mathbf{A}$. From this follows also the relation

$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$.

where $\mathbf{Q}$ is a matrix with the eigenvectors in columns, and $\Lambda$ is a square matrix with the eigenvalues on the diagonal.

Biologically, the eigen analysis allows us to 'rotate' the variation in the data (say, in the multivariate phenotype of an organism) so that the first 'trait' (leading eigenvector) represents the multivariate direction of most variation. Often, this can be interpreted roughly as the size of the organism. The subsequent eigenvectors represent other axes of variation, that could e.g. represent shape. The subsequent eigenvectors are *orthogonal*, so that e.g. the second eigenvector is perpendicular to the first.

In `R`, the function `eigen` performs the eigendecomposition and returns the eigenvectors and corresponding eigenvalues.

```
eigen(cm)
```

```
## eigen() decomposition
## $values
## [1] 1.4034727 0.9349665 0.1615607
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,] 0.07739007  0.8285983  0.5544688
## [2,] 0.90374270  0.1765485 -0.3899741
## [3,] 0.42102245 -0.5312773  0.7351766
```

The eigenvalues represent the amount of variance associated with each eigenvector (given in columns). We can thus compute the proportion of variance associated with each eigenvector as $\lambda_i / \sum \lambda$.

Before continuing, we need to recall the rules for matrix multiplication. There are several forms of matrix multiplication, but the 'normal' matrix multiplication requires that the number of columns in the first matrix equals the number of rows in the second matrix, and the resulting matrix will have the same number of rows as the first matrix, and the same number of columns as the second matrix. If we multiply a matrix of dimensions $m \times n$ with one of dimensions $n \times l$, we get a matrix of dimensions $m \times l$. The matrix multiplication operator in `R` is `%*%`.

EXERCISE: Compute the proportion of variance associated with each eigenvector of **CM**.

```
## [1] 0.56138909 0.37398661 0.06462429
```

EXERCISE: Confirm that the eigenvectors are of unit length (length = 1) and that the angle between them is 90 degrees.

Recall that the length of a vector is the square root of the sum of the vector elements, and the angle between two vectors $u_1$ and $u_2$ is $\frac{180}{\pi} cos^{-1}(u_1 u_2)$.

Length of the eigenvectors

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

Angle between first and second eigenvector

```
##      [,1]
## [1,]   90
```

EXERCISE: Reconstruct the matrix **CM** from the eigenvalues and eigenvectors.

```
##      [,1] [,2] [,3]
## [1,]  0.7  0.2 -0.3
## [2,]  0.2  1.2  0.4
## [3,] -0.3  0.4  0.6
```

```
##      [,1] [,2] [,3]
## [1,]  0.7  0.2 -0.3
## [2,]  0.2  1.2  0.4
## [3,] -0.3  0.4  0.6
```

## Principal Component Analysis

Eigenanalysis is a core component of principal component analysis. In it's simplest form, the principal components are the same as the eigenvectors. Let us derive some new traits along the eigenvectors of **CM**.

```
dim(as.matrix(X))
```

```
## [1] 200   3
```

```
dim(as.matrix(eigen(cm)$vectors[,1]))
```

```
## [1] 3 1
```

```
t1 = as.matrix(X) %*% eigen(cm)$vectors[,1]
t2 = as.matrix(X) %*% eigen(cm)$vectors[,2]
t3 = as.matrix(X) %*% eigen(cm)$vectors[,3]

c(var(X[,1]), var(X[,2]), var(X[,3]))
```

```
## [1] 0.6630232 1.3029074 0.7238522
```

```
c(var(t1), var(t2), var(t3))
```

```
## [1] 1.6177712 0.8913354 0.1806762
```

Notice that the variances of new traits decreases from the first to the third trait, which was not the case for the original traits. However, the total variance stays the same (because we have just reorganized the variation).

```
var(t1) + var(t2) + var(t3)
```

```
##          [,1]
## [1,] 2.689783
```

```
var(X[,1]) + var(X[,2]) + var(X[,3])
```

```
## [1] 2.689783
```

The eigenvectors are *orthogonal*, i.e. they are not correlated with each other.

A very similar operation is performed by several `R` packages for principal component analysis, e.g. `prcomp`. The principal components are not exactly the same as defining traits along the eigenvectors, but are subject to some further rotation. However, the principal components will be strongly correlated with the traits defined along the eigenvectors.

```
pca = princomp(X)
summary(pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3
## Standard deviation     1.2716861 0.9417859 0.4149549
## Proportion of Variance 0.6042541 0.3314088 0.0643371
## Cumulative Proportion  0.6042541 0.9356629 1.0000000
```

The proportion of variance explained by each principal component is computed as the variance of each principal component divided by the total, which is basically equal to the corresponding eigenvalue divided by the sum of the eigenvalues, i.e. $\lambda_i / \sum \lambda$.

```
pca$sdev^2/sum(pca$sdev^2)
```

```
##    Comp.1    Comp.2    Comp.3
## 0.6042541 0.3314088 0.0643371
```

```
eigen(cm)$values/sum(eigen(cm)$values)
```

```
## [1] 0.56138909 0.37398661 0.06462429
```

The small difference is again due to how the principal components are calculated, but biologically the interpretation is the same.

To understand how each principal component is constructed, we look at the loadings of each original variable onto the PCs.

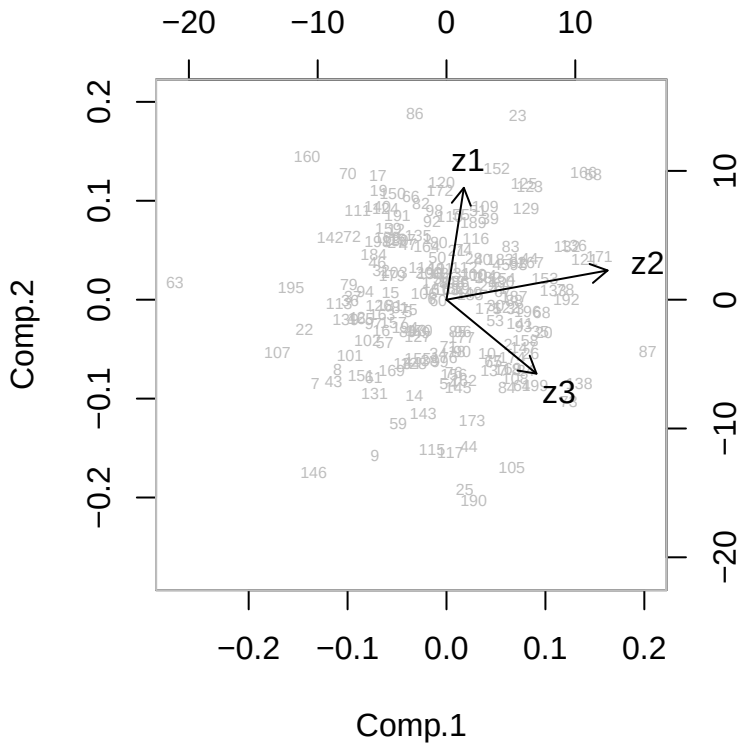
```
pca$loadings[1:3, 1:3]
```

```
##        Comp.1     Comp.2     Comp.3
## z1 0.09365193  0.8150658  0.5717491
## z2 0.86921580  0.2130937 -0.4461557
## z3 0.48548244 -0.5387567  0.6885114
```

Here, the first PC represents variation in `z2` and, to a lesser extent, `z3` (the two variables must thus be positively correlated). The second PC represents mostly `z1`, and also separates variation in `z2` from variation in `z3`.

A PCA can also be illustrated by a biplot.

```
biplot(pca, col=c("grey", "black"), cex=c(.5, 1))
```




## Data exercise: principal component analysis

Choose any dataset comprosing three or more continuous variables (for example the *blossoms* data or the *alpine plants* data). Perform a principal component analysis and produce a biplot. Interpret the results in terms of which biological entitites are represented by each principal component.

## Principal component regression

One application of principal component analysis is principal component regression, where we fit the regression model to the principal components instead of the original variables. This could be useful if we have multiple correlated variables and want to include them while avoiding multicollinearity issues.

Below we first define a response variable $y$ based on the data we simulated from the multivariate normal distribution, and then fit a standard linear regression model.

```
XX = as.data.frame(scale(X))
y = 0.5*XX$z1 -0.3*XX$z2 + 0.2*XX$z3 + rnorm(200, 0, 1)

m0 = lm(y~XX$z1+XX$z2+XX$z3)
summary(m0)
```

```
##
## Call:
## lm(formula = y ~ XX$z1 + XX$z2 + XX$z3)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.04424 -0.74933  0.04343  0.65130  2.65215
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.06918    0.07077  -0.978    0.329
## XX$z1        0.54770    0.09386   5.836 2.19e-08 ***
## XX$z2       -0.16639    0.10453  -1.592    0.113
## XX$z3        0.06816    0.10814   0.630    0.529
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.001 on 196 degrees of freedom
## Multiple R-squared:  0.1992, Adjusted R-squared:  0.1869
## F-statistic: 16.25 on 3 and 196 DF,  p-value: 1.793e-09
```

We then perform a principal component analysis, extract the three principal components, and fit a second linear models with the PCs as predictors.

```
pca = princomp(XX)
pc1 = pca$scores[,1]
pc2 = pca$scores[,2]
pc3 = pca$scores[,3]

m3 = lm(y~pc1+pc2+pc3)
```

```
summary(m3)
```

```
##
## Call:
## lm(formula = y ~ pc1 + pc2 + pc3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.04424 -0.74933  0.04343  0.65130  2.65215
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.06918    0.07077  -0.978  0.32949
## pc1          0.15401    0.05682   2.710  0.00732 **
## pc2          0.37505    0.06390   5.869 1.84e-08 ***
## pc3          0.40979    0.15530   2.639  0.00899 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.001 on 196 degrees of freedom
## Multiple R-squared:  0.1992, Adjusted R-squared:  0.1869
## F-statistic: 16.25 on 3 and 196 DF,  p-value: 1.793e-09
```

Notice that the $r^2$ of this model is the same as for the standard model fitted to the original variables. This is expected because we have just reorganized the original variables into new variables. In fact, we can calculate

the slopes of the standard model from the slopes of the principal-component regression by using the loadings of the principal components,

$\beta = \mathbf{Qb}$

where $\mathbf{Q}$ is a matrix of loadings for each PC (similar to eigenvectors), and $\mathbf{b}$ is a vector of regression coefficients.

```
Q = pca$loadings
b = as.matrix(summary(m3)$coefficients[-1,1])
dim(Q)
```

```
## [1] 3 3
```

```
dim(b)
```

```
## [1] 3 1
```

```
Q %*% b
```

```
##           [,1]
## z1  0.54770020
## z2 -0.16639480
## z3  0.06816457
```

These are the slopes of the standard model, so we have not necessarily learned very much new by fitting the principal component regression. But what if we fit the model to only the first principal component?

```
m1 = lm(y~pc1)
summary(m1)
```

```
##
## Call:
## lm(formula = y ~ pc1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.39164 -0.82388  0.01201  0.83536  2.72336
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.06918    0.07749  -0.893   0.3731
## pc1          0.15401    0.06222   2.475   0.0142 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.096 on 198 degrees of freedom
## Multiple R-squared:  0.03001,    Adjusted R-squared:  0.02511
## F-statistic: 6.126 on 1 and 198 DF,  p-value: 0.01416
```

The model explains less of the variance, but we have all the original variables represented.

## Data exercise: Principal component regression

Load the alpine plants dataset, and use principal component analysis to reduce the environmental variables into a reduced set of principal components (you can use a single PCA or several, for example one for temperature separately). Fit a principal-component regression and interpret the results.