LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Introduction to Machine Learning

PCA

Bernd Bischl, Christoph Molnar, Daniel Schalk, Fabian Scheipl

Department of Statistics – LMU Munich

Introduction

SUGGESTED LITERATURE

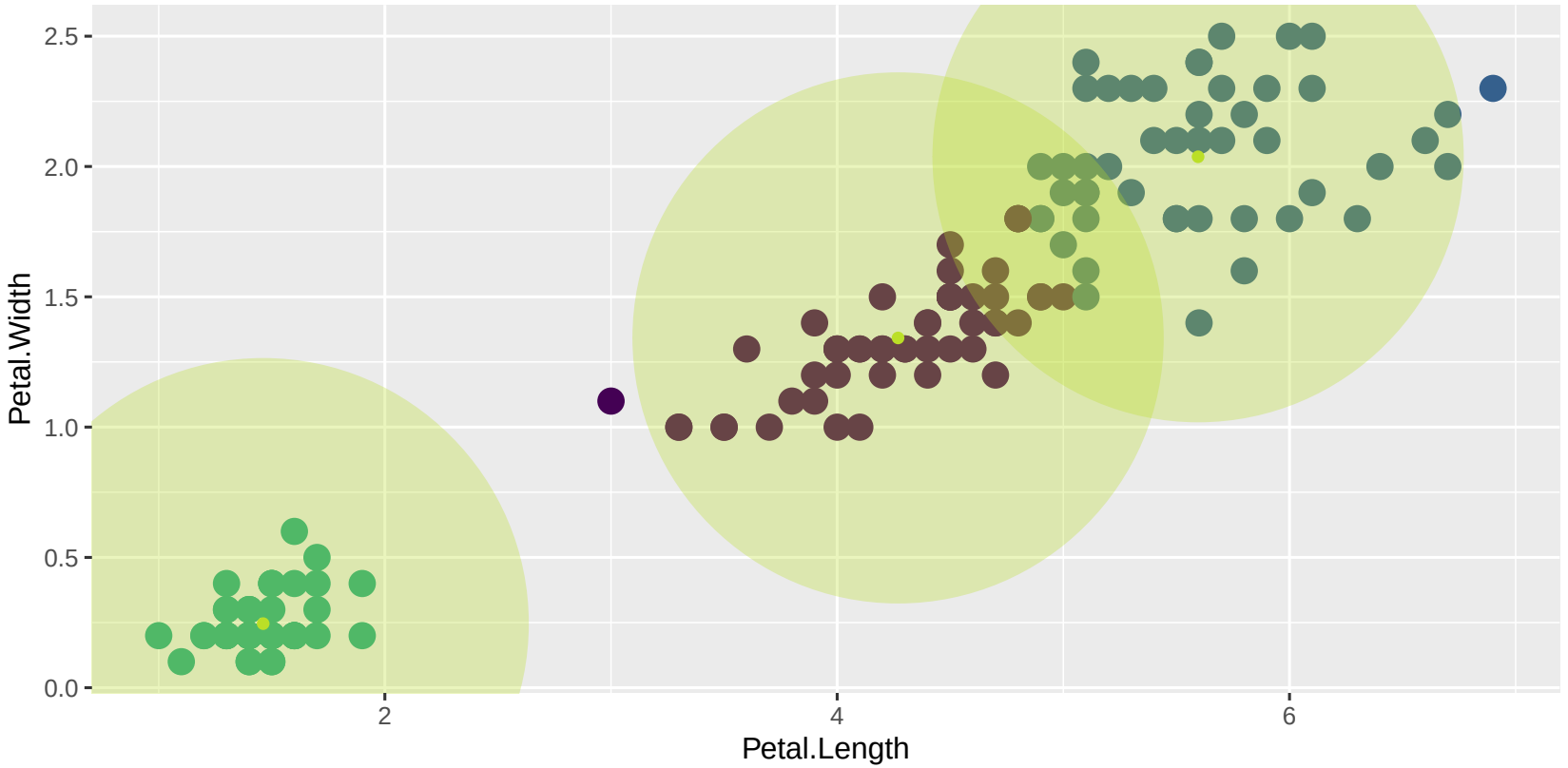
- Hastie, T., Tibshirani, R., Friedman, J. (2009): The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013): An Introduction to Statistical Learning with Applications in R. Springer.
- Aggarwal, C. C., & Reddy, C. K. (Eds.). (2013). Data Clustering: Algorithms and Applications. CRC press.

UNSUPERVISED LEARNING

- Supervised machine learning deals with *labeled* data, i.e., we have input data x and the outcome y of past events.
- Here, the aim is to learn relationships between x and y.
- Unsupervised machine learning deals with data that is *unlabeled*, i.e., there is no real output y.
- Here, the aim is to search for patterns within the inputs x.

CLUSTERING TASK

Goal: Group data into similar clusters (or estimate fuzzy membership probabilities)



CLUSTERING: CUSTOMER SEGMENTATION

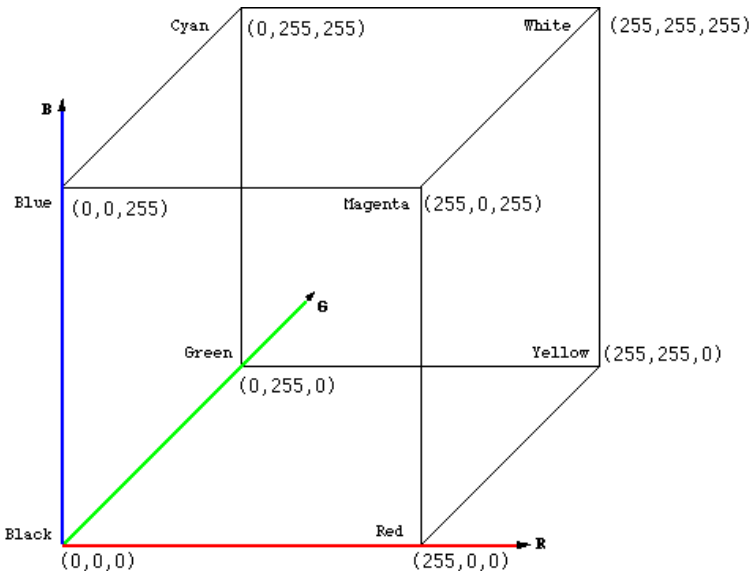
- In marketing, customer segmentation is an important task to understand customer needs and to meet with customer expectations.
- Customer data is partitioned in terms of similiarities and the characteristics of each group are summarized.
- Marketing strategies are designed and prioritized according to the group size.

Example Use Cases:

- Personalized ads (e.g., recommend articles).
- Music/Movie recommendation systems.

CLUSTERING: IMAGE COMPRESSION

- An image consists of pixels arranged in rows and columns.
- Each pixel contains **RGB** color information, i.e., a mix of the intensity of 3 **primary colors**: **R**ed, **G**reen and **B**lue.
- Each primary color takes intensity values between 0 and 255.



Source: By Ferlixwangg CC BY-SA 4.0, from Wikimedia Commons.

CLUSTERING: IMAGE COMPRESSION

An image can be compressed by reducing its color information, i.e., by replacing similar colors of each pixel with, say, k distinct colors.

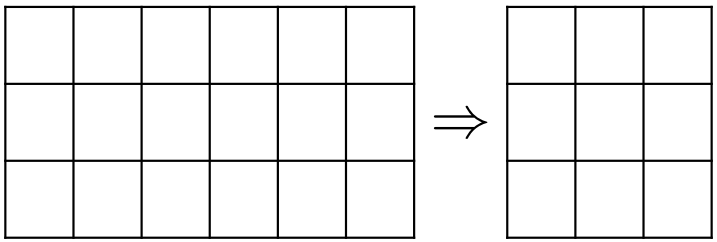
Example:

Original Image

Image using 16 Colors

DIMENSIONALITY REDUCTION TASK

Goal: Describe data with fewer features (reduce number of columns).
$\Rightarrow$ there will always be an information loss.



Unsupervised Methods:

- Principle Component Analysis (PCA).
- Factor Analysis (FA).
- Feature filter methods.

Supervised Methods:

- Linear Discriminant Analysis (LDA).
- Feature filter methods.

Principal Component Analysis

NORMALIZING DATA

A variable X can be normalized by substracting its values with the mean $\bar{X}$ and dividing by the standard deviation s_X, e.g. $\tilde{X} = \frac{X - \bar{X}}{s_X}$.

Example:
Consider the following body heights measured in different units:

	Person A	Person B	Person C	mean	sd
body height (cm)	180.00	172.00	175.00	175.67	4.04
body height (m)	1.80	1.72	1.75	1.76	0.04
body height (feet)	5.91	5.64	5.74	5.76	0.13

After normalizing, we always obtain the normalized body height (no matter which unit was used):

	Person A	Person B	Person C	mean	sd
normalized body height	1.07	-0.91	-0.16	0.00	1.00

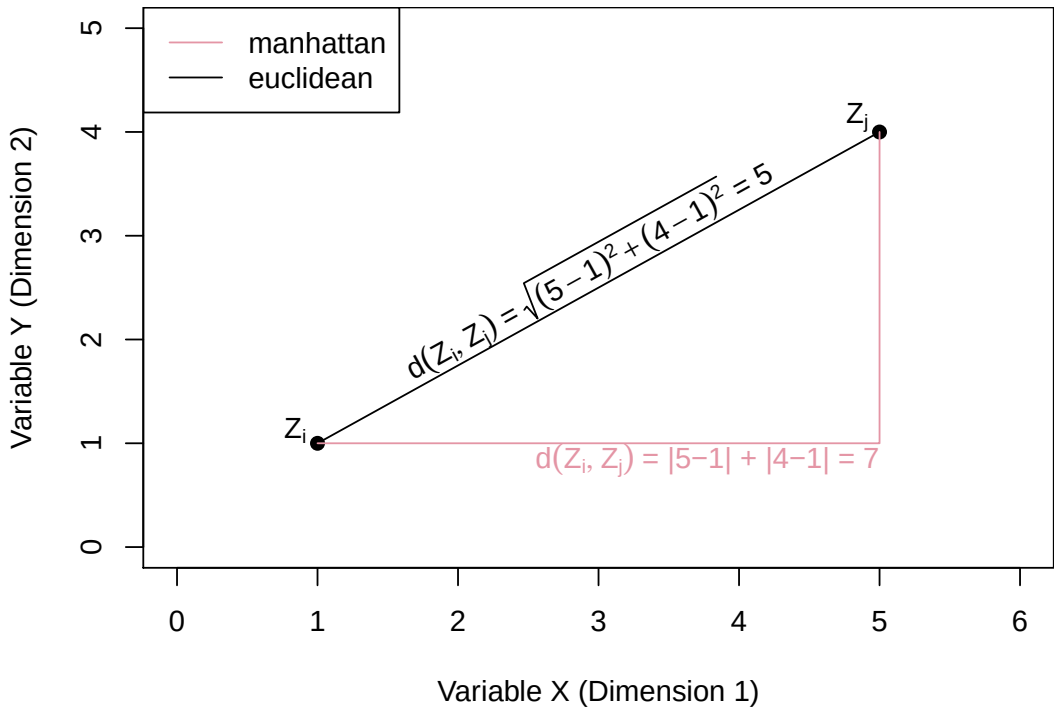
NORMALIZING DATA

Normalizing all variables in a data set, can have several advantages:

- It puts all variables into *comparable* units, i.e., we make sure that all normalized variables have mean 0 and standard deviation of 1.
- It can avoid numerical instabilites in several algorithms, e.g. if a variable has very low / high values.
- It helps in computing meaningful *distances* between observations.

NORMALIZING DATA:DISTANCES

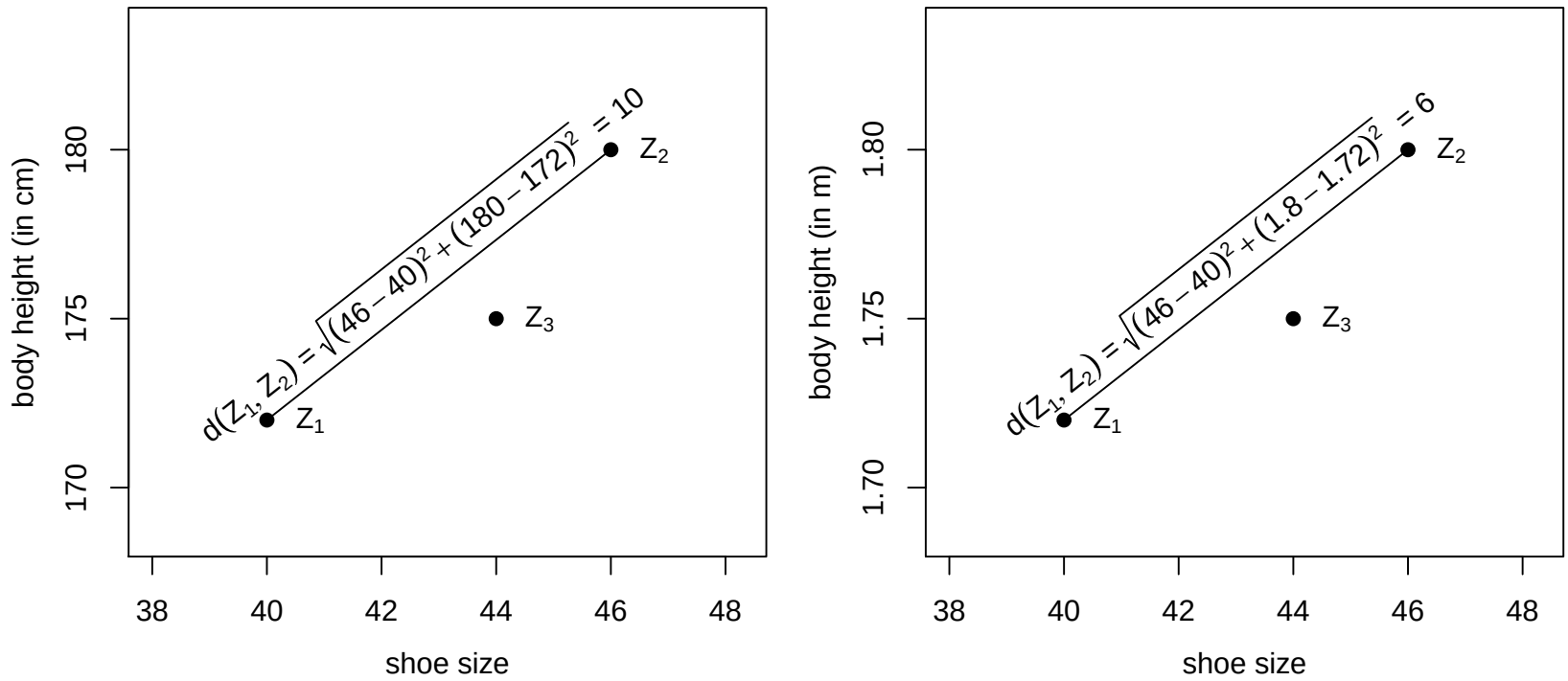
There are many ways to define the distance between two points, e.g., $Z_i = (X_i, Y_i)$ and $Z_j = (X_j, Y_j)$:



- manhattan: sum up the absolute distances in each dimension.
- euclidean: remember Pythagoras theorem from school?

NORMALIZING DATA:DISTANCES

It is often a good idea to *normalize* the data before computing distances, especially when the scale of variables is different, e.g. the euclidean distance between the point Z_1 and Z_2:



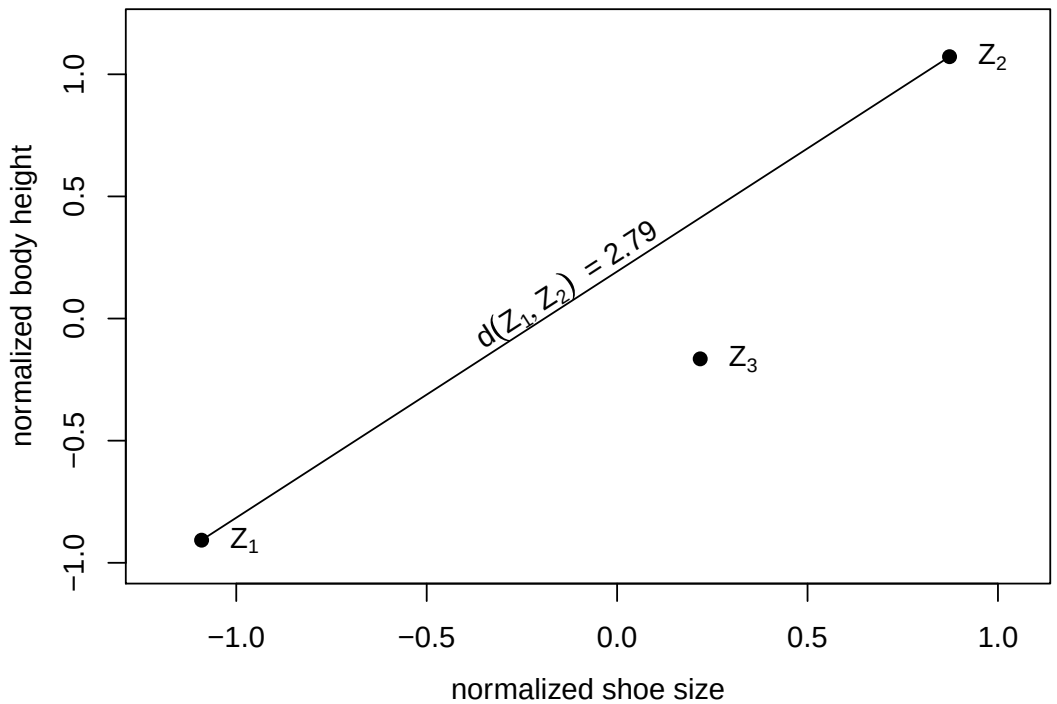
On the right plot, the distance is dominated by `shoe size`.

NORMALIZING DATA: DISTANCES

The normalized variable $\tilde{X}_{\texttt{shoe.size}}$ is computed by

$$\tilde{X}_{\texttt{shoe.size}} = \frac{X_{\texttt{shoe.size}} - \bar{X}_{\texttt{shoe.size}}}{s_{X_{\texttt{shoe.size}}}}.$$

Distances based on normalized data are better comparable and **robust** in terms of linear transformations (e.g., conversion of physical units).



NORMALIZING: COVARIANCE VS. CORRELATION

The **variance** of a normalized variable is always 1, its mean is always 0.

The **covariance** of two normalized variables $\tilde{X} = \frac{X - \bar{X}}{s_X}$ and $\tilde{Y} = \frac{Y - \bar{Y}}{s_Y}$ is the same as the **correlation** of the non-normalized variables X and Y.

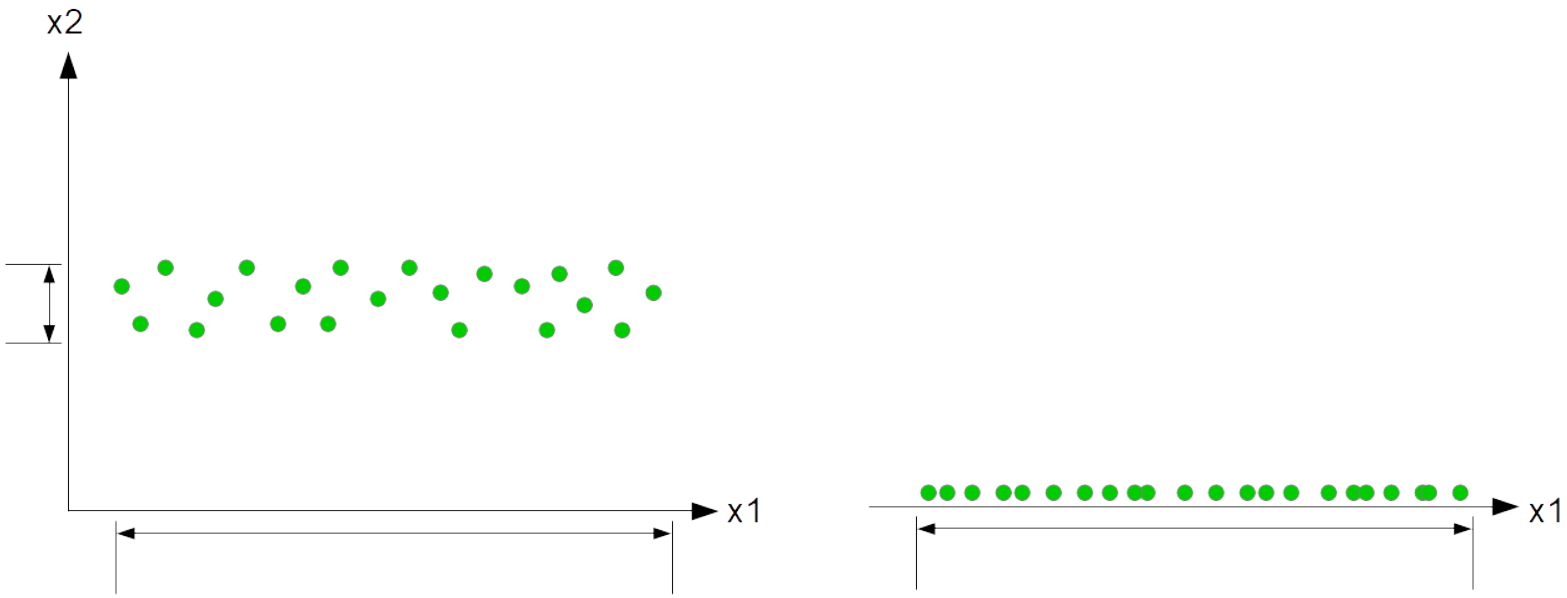
One can proof this with the help of

$$s_{\tilde{X}\tilde{Y}} = \frac{1}{n-1} \sum_{i=1}^{n} (\tilde{x}_i - \bar{\tilde{x}})(\tilde{y}_i - \bar{\tilde{y}}) = \ldots = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(x_i - \bar{x})}{s_X} \frac{(y_i - \bar{y})}{s_Y} = r_{XY}.$$

PCA INTUITION

Motivational example I:

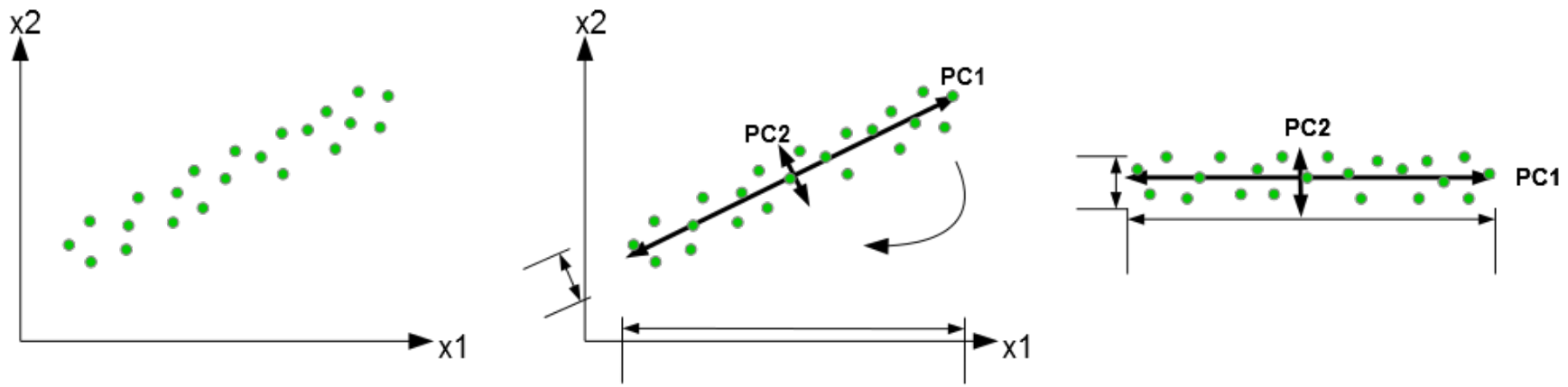
- Variable x_1 explains most of the variation.
- Variable x_2 has a lower variance than x_1.
- If we disregard x_2 and project the points into the 1-dimensional space of x_1, we do not lose much information w.r.t. variability.



PCA INTUITION

Motivational example II:

- x_1 and x_2 are correlated and have similar variances.
- Find a new orthogonal axes (e.g. PC1 and PC2), where PC1 explains most of the variation.
- Rotate the points and consider PC1 and PC2 as new coordinate system (situation as in the previous example).
- We can now project points onto PC1 and disregard PC2 (hopefully without losing much information).

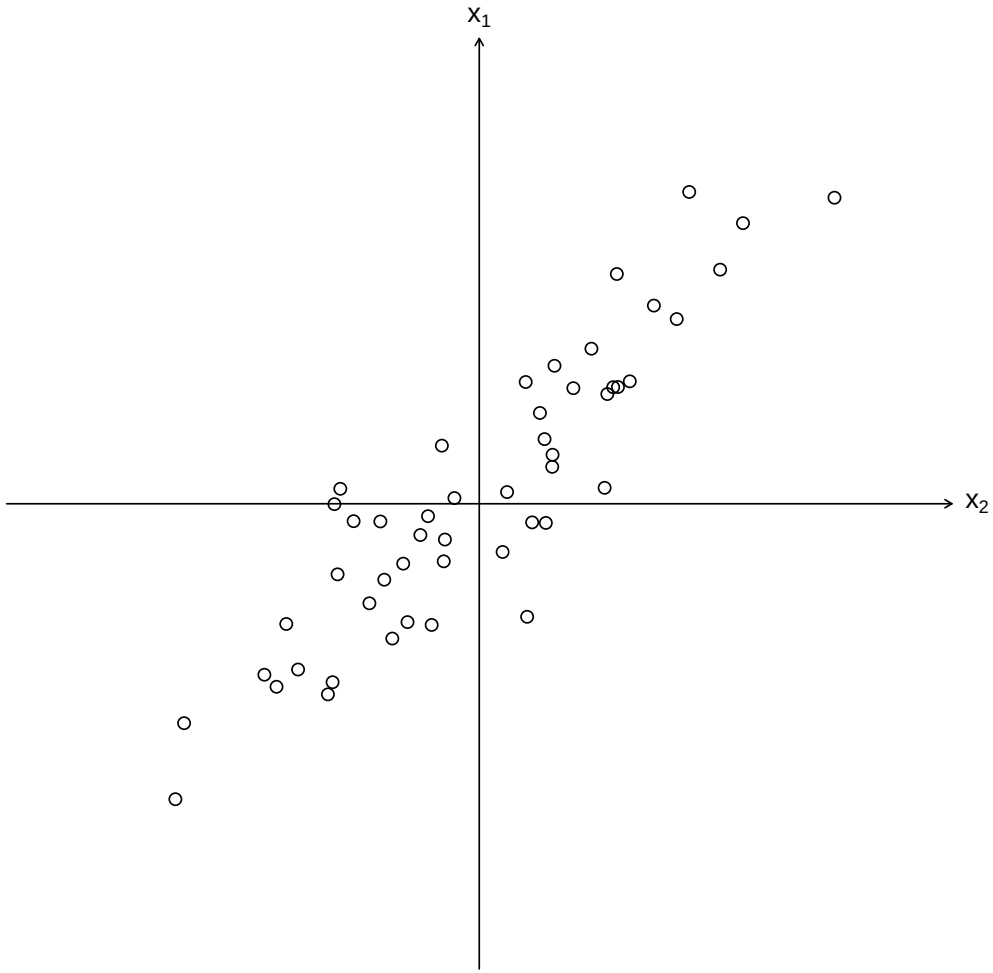


PCA INTUITION

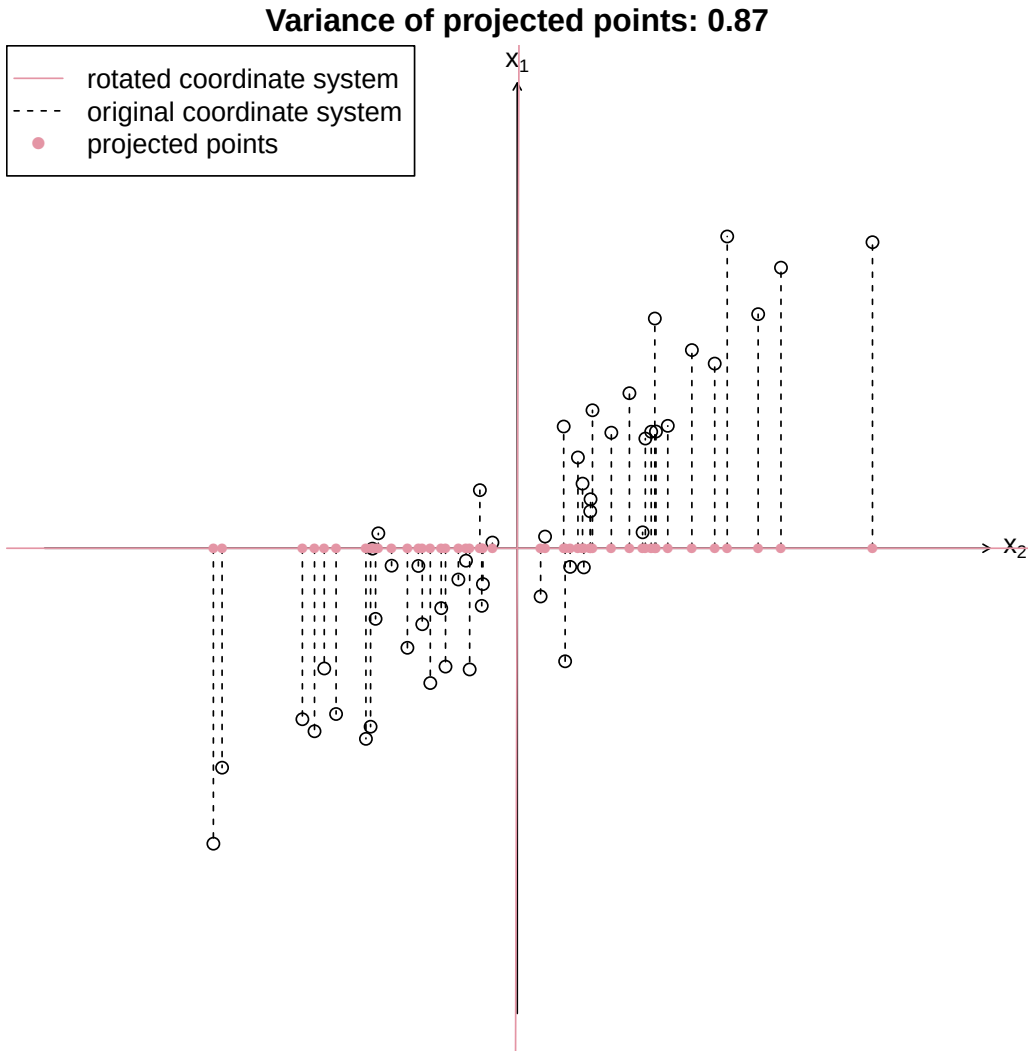
General procedure:

1. Rotate the original p-dimensional coordinate system until the first PC that explains most of the variation is found.
2. Fix the first PC and proceed with rotating the remaining $p - 1$ coordinates until the second PC (which is orthogonal to the first PC) is found that explains most of the *remaining* variation, etc.
3. We can reduce the dimensions by projecting the points onto the first, say $k < p$, PC.

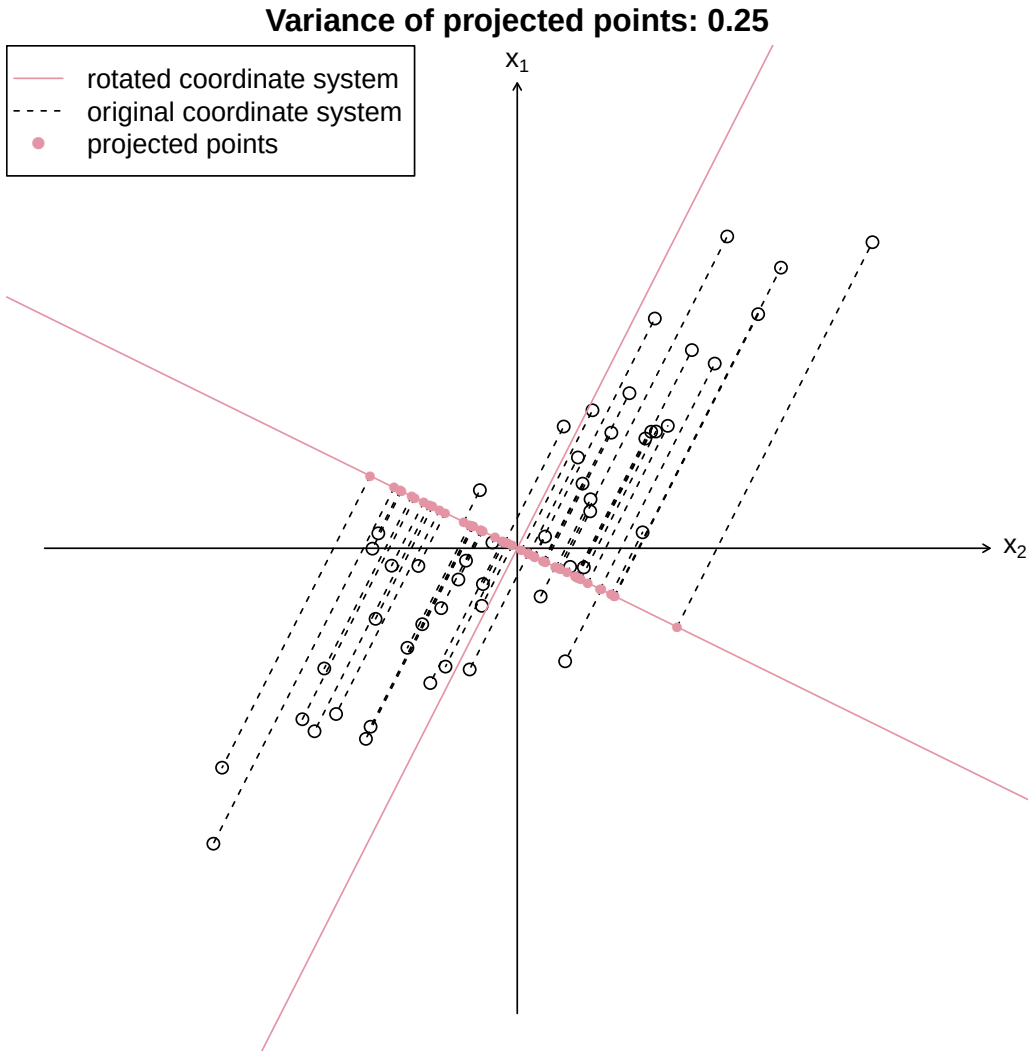
PCA INTUITION: FIND FIRST PC



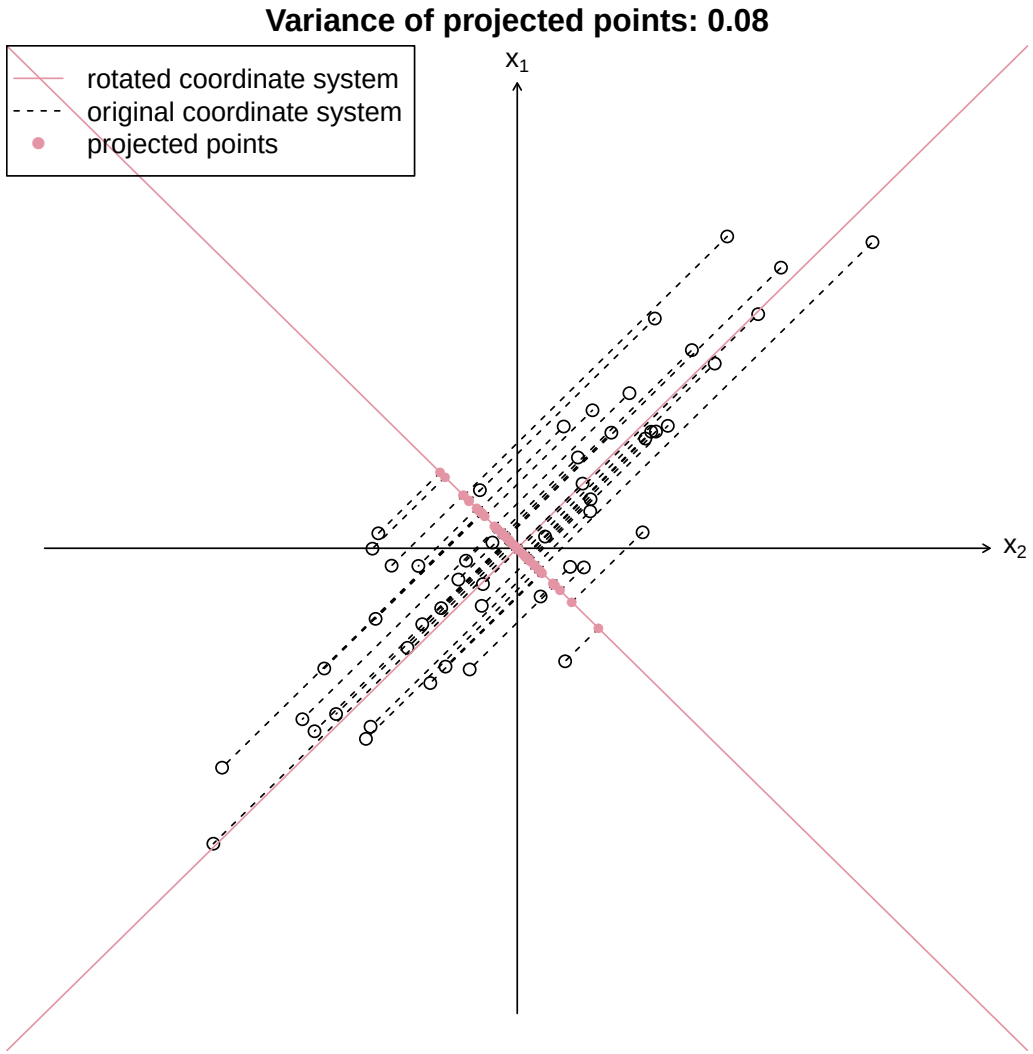
PCA INTUITION: FIND FIRST PC



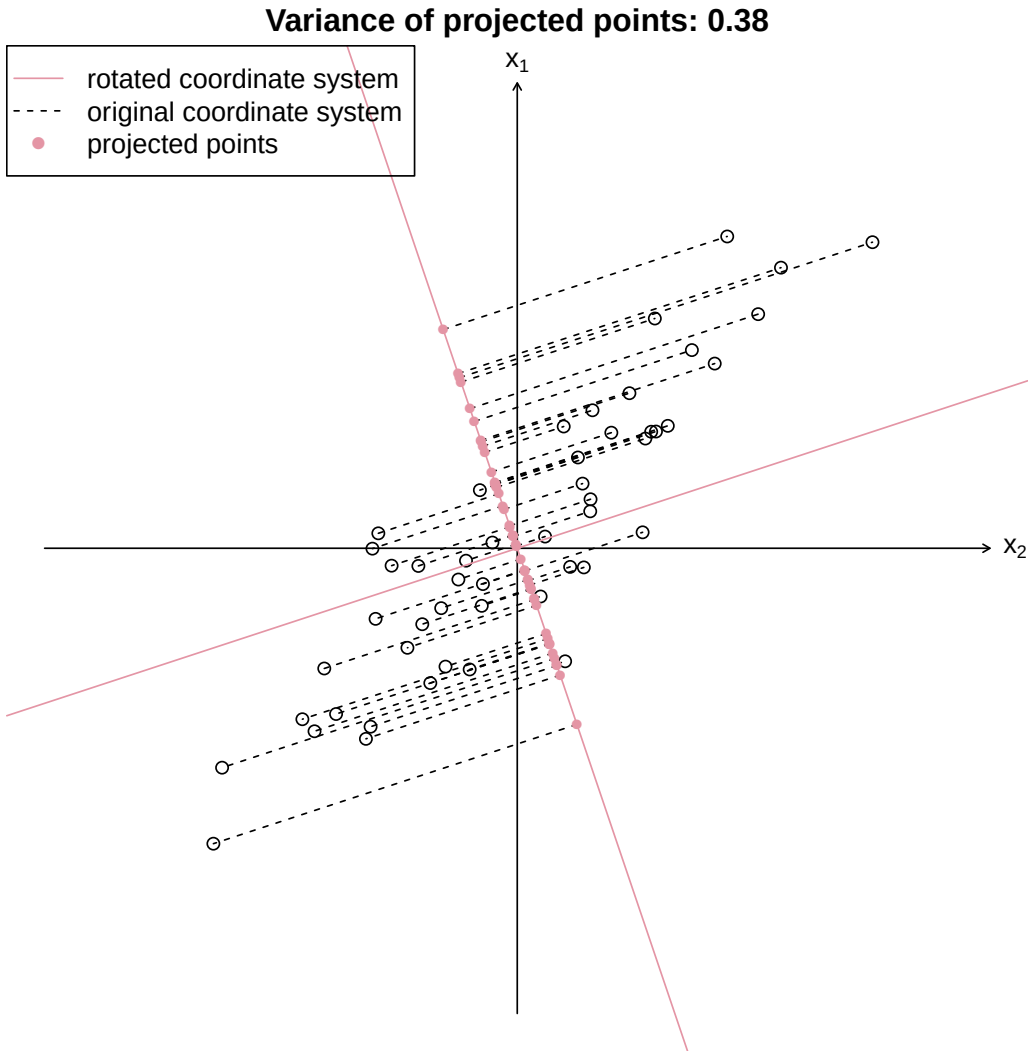
PCA INTUITION: FIND FIRST PC



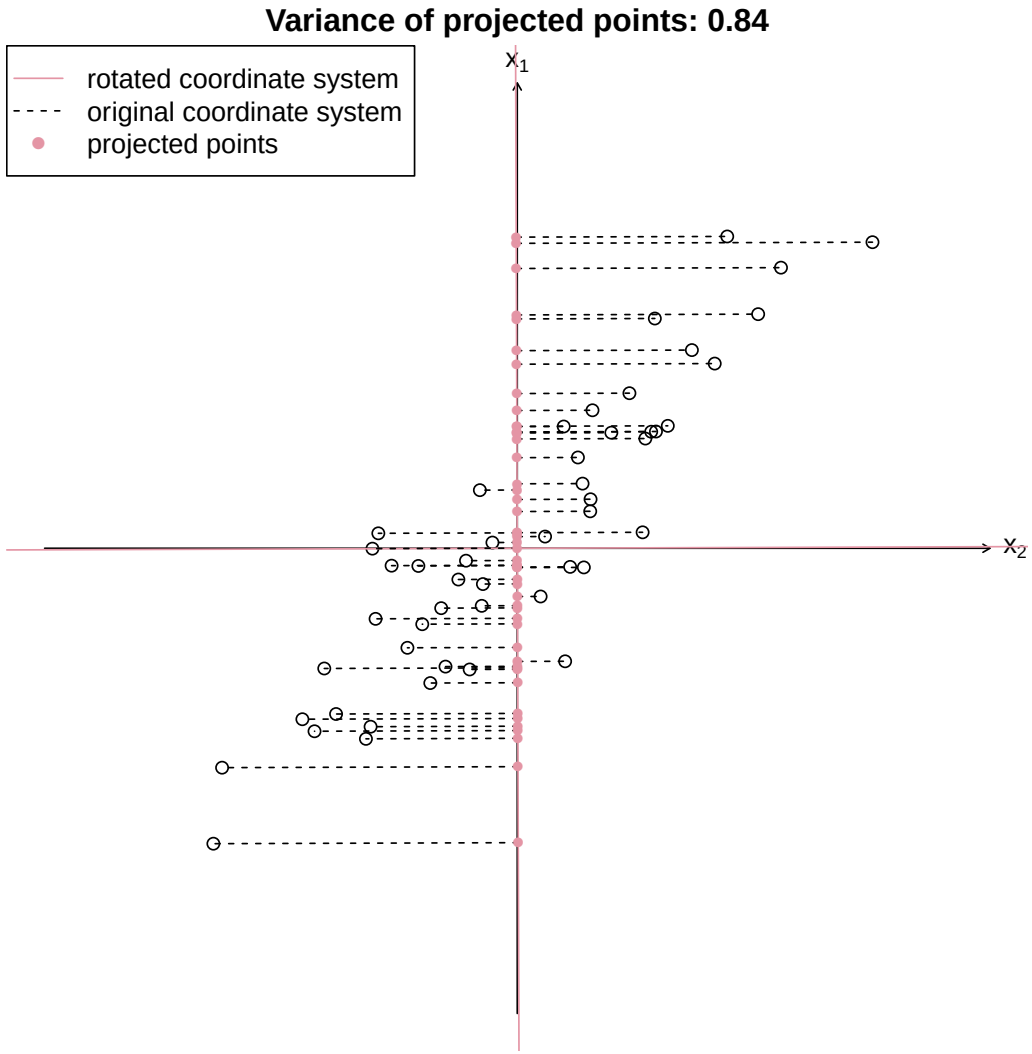
PCA INTUITION: FIND FIRST PC



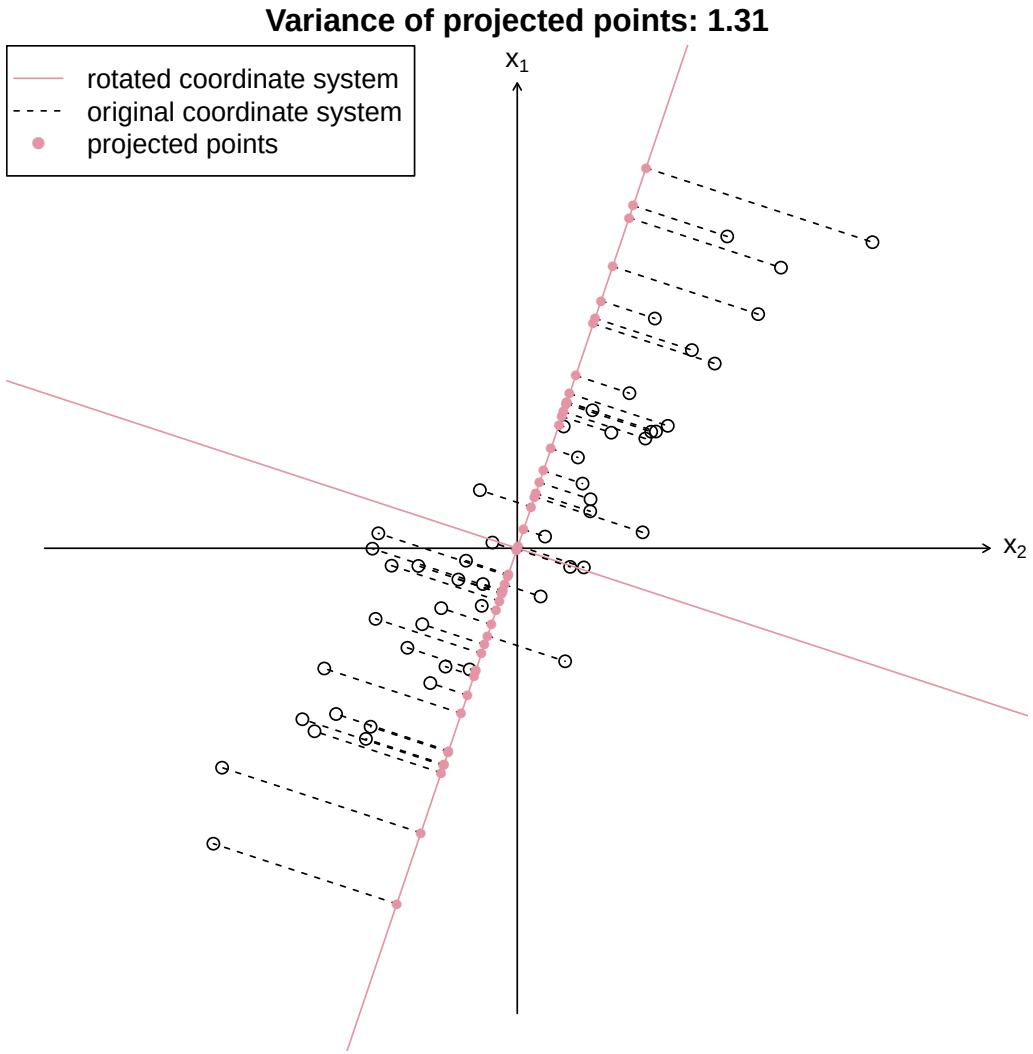
PCA INTUITION: FIND FIRST PC



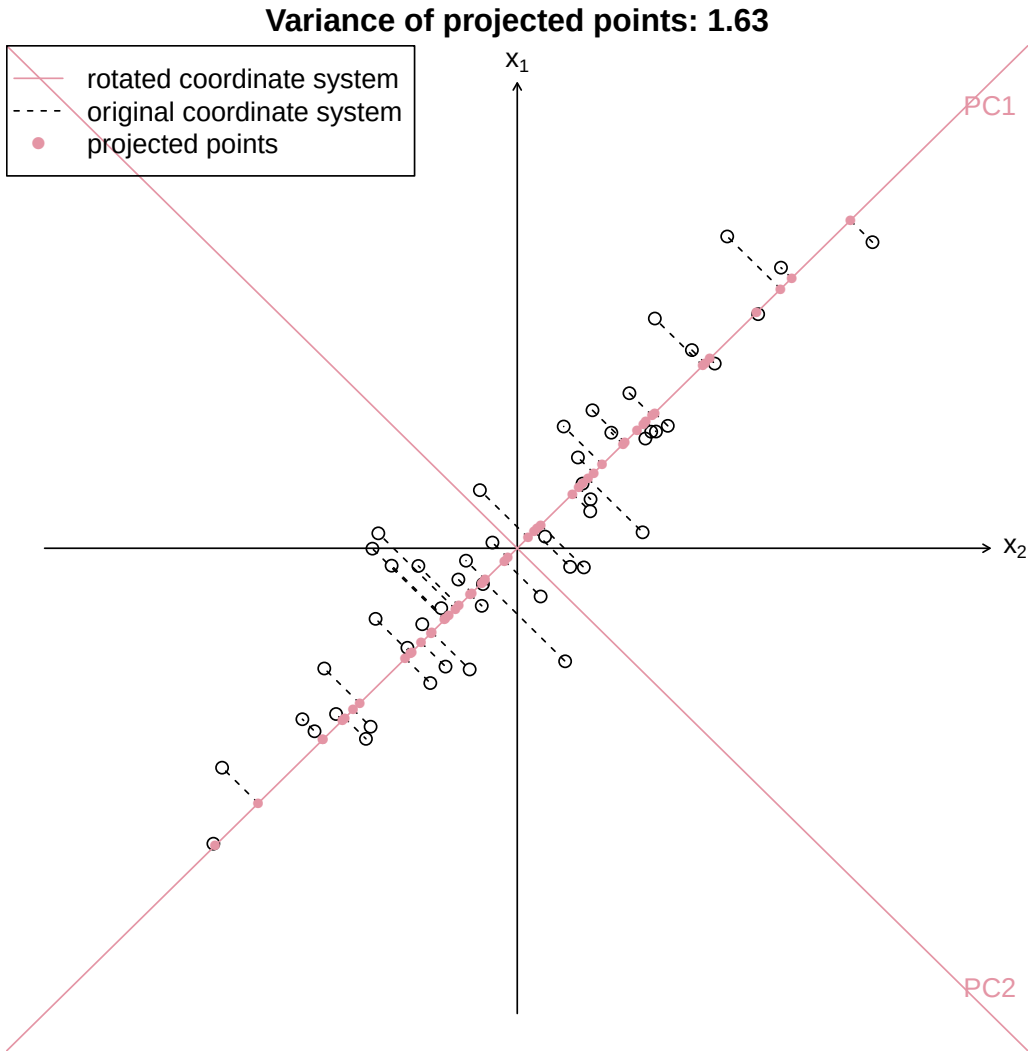
PCA INTUITION: FIND FIRST PC



PCA INTUITION: FIND FIRST PC

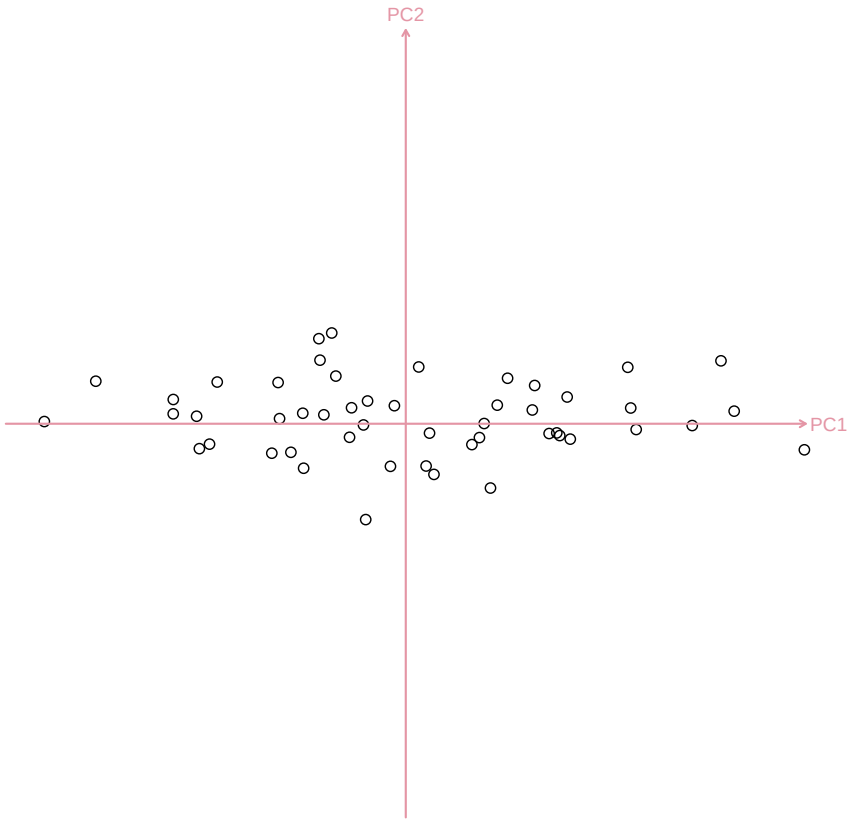


PCA INTUITION: FIND FIRST PC



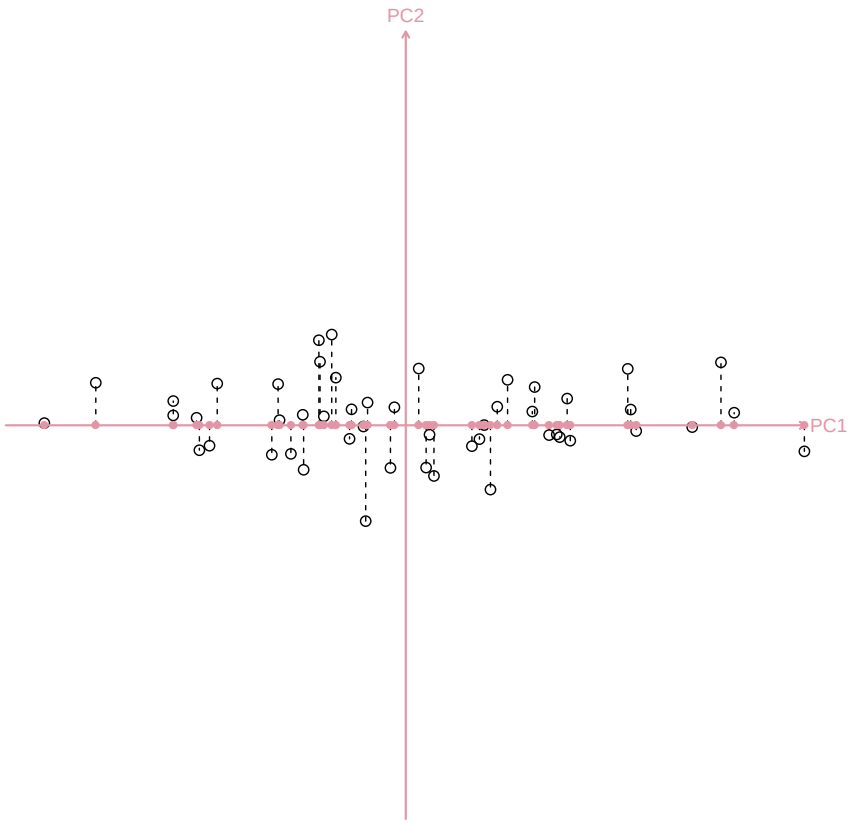
PCA INTUITION: REDUCE DIMENSIONALITY

Rotate the points and use PC1 and PC2 as new coordinate system.
Here, the PC1 axis explains most of the variance:



PCA INTUITION: REDUCE DIMENSIONALITY

Dimensionality can be reduced by projecting the points onto the PC1 (and by disregarding PC2). The hope is that we won't lose much information this way.



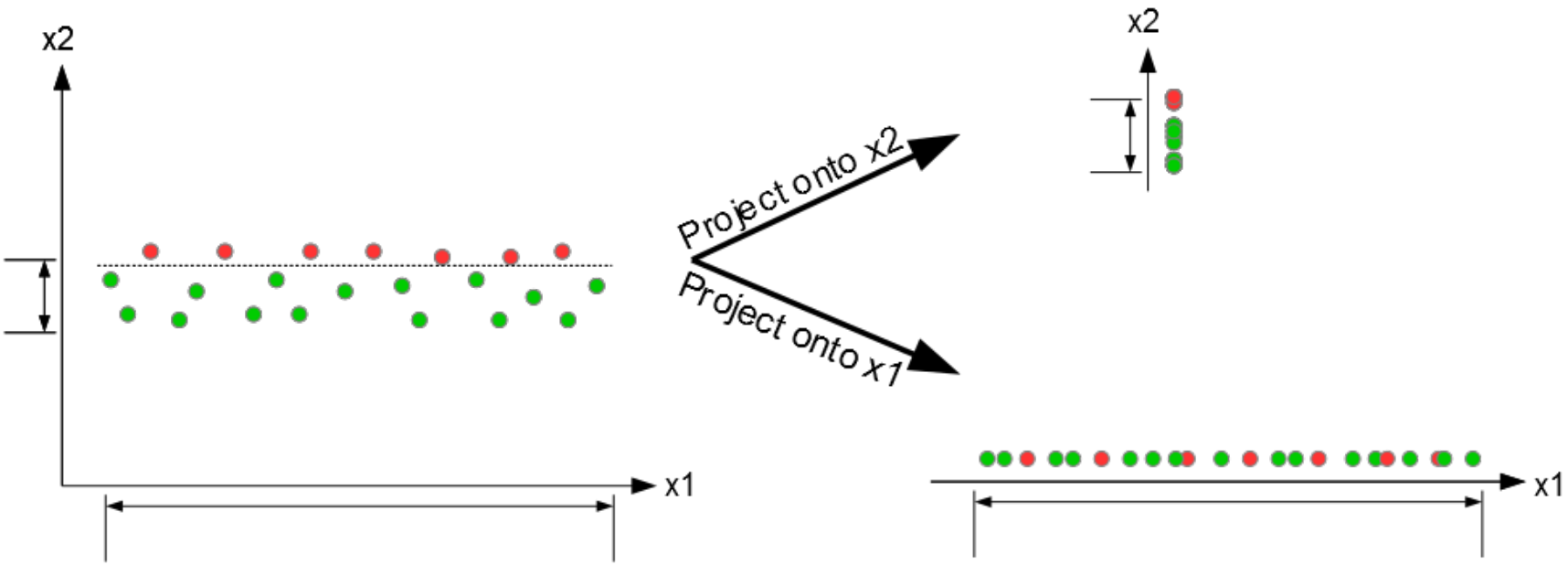
PCA INTUITION: SUMMARY

Idea: Transform an original set of correlated metric variables to a new set of uncorrelated (orthogonal) metric variables, called principal components (PC), that explain the variability in the data.

- The objective is to investigate if only a few PC account for most of the variability in the original data.
- If the objective is fulfilled, we can use fewer PCs to reduce the dimensionality.
- The PCs remove collinearity of the input variables as they are orthogonal to each other.

PCA INTUITION: FINAL REMARKS

- PCA is used for dimensionality reduction by disregaring dimensions with lower variability.
- There is always an information loss, especially for other criteria.
- E.g., dimensionality reduciton can worsen the classification accuracy when the task is to classify two groups:



DERIVING THE FIRST PC MATHEMATICALLY

Aim: Find a new set of variables (PC scores) $\mathbf{pc}_1, \ldots, \mathbf{pc}_p$ based on the original data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$ so that

- each PC score $\mathbf{pc}_1, \ldots, \mathbf{pc}_p$ is a linear combination of the original metric variables with coefficient weights (so-called **loading vectors**) $\mathbf{a}_1, \ldots, \mathbf{a}_p$, i.e.

$$\mathbf{pc}_j = a_{j1}\mathbf{x}_1 + a_{j2}\mathbf{x}_2 + \ldots + a_{jp}\mathbf{x}_p = \mathbf{X}\mathbf{a}_j.$$

- the set is mutually uncorrelated: $Cov(\mathbf{pc}_j, \mathbf{pc}_k) = 0, \ \forall j \neq k$.
- the variances of the PC scores decrease:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_p, \quad \text{where } \lambda_k := Var(\mathbf{pc}_k).$$

DERIVING THE FIRST PC MATHEMATICALLY

We look for the loading vector $\mathbf{a}_1 = (a_{11}, a_{21}, \ldots, a_{p1})^\top$ that maximizes the variance of $\mathbf{pc}_1$:

$$\max_{\mathbf{a}_1} \; Var(\mathbf{pc}_1) = Var(\mathbf{X}\mathbf{a}_1) = \mathbf{a}_1^\top \Sigma \mathbf{a}_1$$

subject to the normalization constraint $\mathbf{a}_1^\top \mathbf{a}_1 = \sum_{k=1}^{p} a_{k1}^2 = 1$.

The constraint is required for identifiability reasons, otherwise we could maximize the variance by just increasing the values in $\mathbf{a}_1$.

Repeat this maximization step for the other PCs and additionally use the orthogonality constraint, i.e. for the second PC:

$$\mathbf{a}_2^\top \mathbf{a}_1 = 0.$$

EXAMPLE: THE OLYMPIC HEPTATHLON DATA

The `heptathlon` data set in the R package "HSAUR3" contains the competition results of 25 athletes in 7 disciplines for the Olympics held in Seoul in 1988.

```
## install.packages("HSAUR3")
data(heptathlon, package = "HSAUR3")
```

Aim: Rank the athletes according to their overall performance in all 7 disciplines.

Idea: Use PCA to reduce the dimensionality (i.e., reduce the results of the 7 disciplines to one dimension) and compare the scores of the first PC with the official scores.

EXAMPLE: THE OLYMPIC HEPTATHLON DATA

Variables of the `heptathlon` data:

- `hurdles`: results 100m hurdles (in seconds).
- `highjump`: results high jump (in m).
- `shot`: results shot putt (in m).
- `run200m`: results 200m race (in seconds).
- `longjump`: results long jump (in m).
- `javelin`: results javelin (in m).
- `run800m`: results 800m race (in seconds).
- `score`: total score of the official scoring system.

EXAMPLE: THE OLYMPIC HEPTATHLON DATA

The variables `hurdles`, `run200m` and `run800m` are time measurements, i.e. low values are better. For all other variables high values are better.

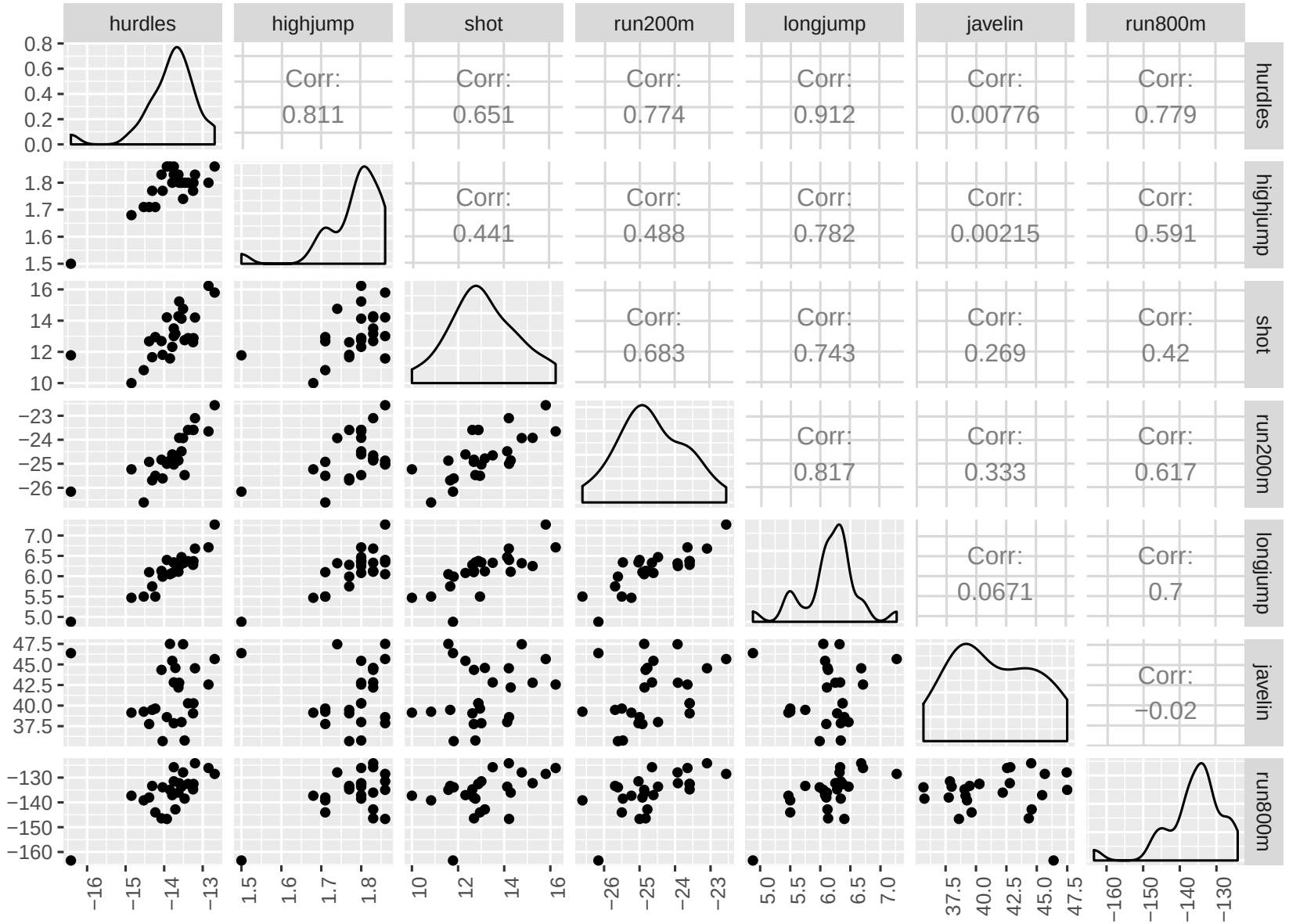
Results of the best and worst participant:

	hurdles	highjump	shot	run200m	longjump	javelin	run800m	score
Joyner-Kersee (USA)	12.69	1.86	15.80	22.56	7.27	45.66	128.5	7291
Launa (PNG)	16.42	1.50	11.78	26.16	4.88	46.38	163.4	4566

We use negative time measurements so that higher values are better and therefore all variables have the same direction:

	hurdles	highjump	shot	run200m	longjump	javelin	run800m	score
Joyner-Kersee (USA)	-12.69	1.86	15.80	-22.56	7.27	45.66	-128.5	7291
Launa (PNG)	-16.42	1.50	11.78	-26.16	4.88	46.38	-163.4	4566

SCATTER PLOT MATRIX



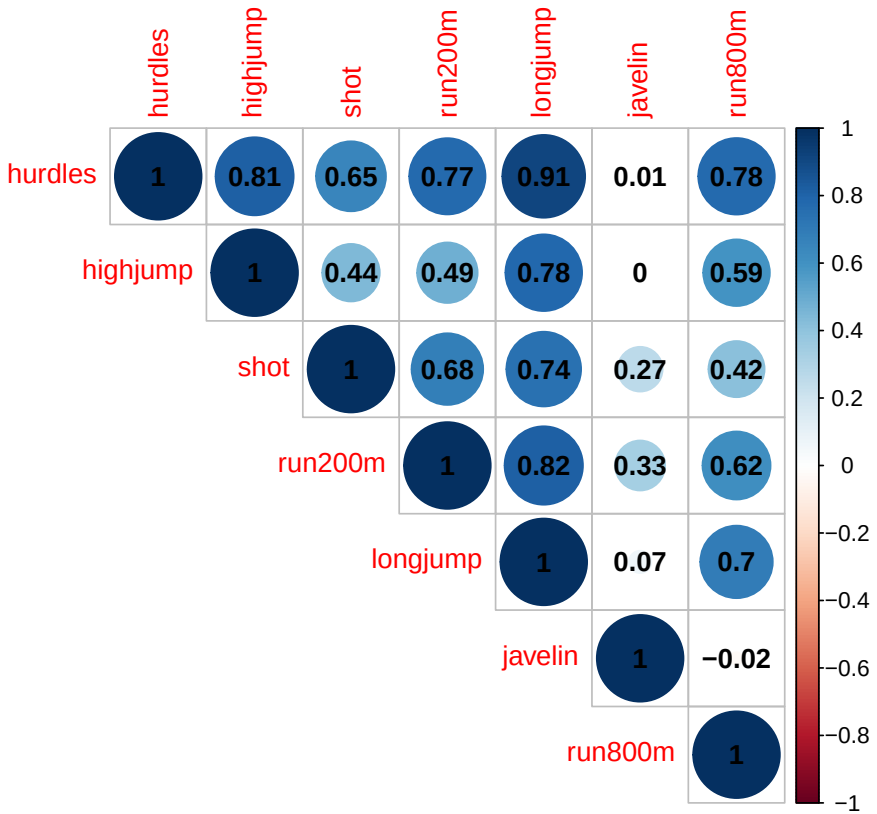
CORRELATION MATRIX

We can compute all pairwise correlations of the variables (without the `score` column):

```
##           hurdles highjump   shot run200m longjump   javelin  run800m
## hurdles  1.000000 0.811403 0.6513  0.7737  0.91213  0.007763  0.77926
## highjump 0.811403 1.000000 0.4408  0.4877  0.78244  0.002153  0.59116
## shot     0.651335 0.440786 1.0000  0.6827  0.74307  0.268989  0.41962
## run200m  0.773721 0.487664 0.6827  1.0000  0.81721  0.333043  0.61681
## longjump 0.912134 0.782442 0.7431  0.8172  1.00000  0.067108  0.69951
## javelin  0.007763 0.002153 0.2690  0.3330  0.06711  1.000000 -0.02005
## run800m  0.779257 0.591163 0.4196  0.6168  0.69951 -0.020049  1.00000
```

CORRELOGRAM

```
library(corrplot)
corrplot(cor.mat, type = "upper", addCoef.col = "black")
```



PERFORM PCA USING `PRINCOMP()`

Remember: The first PC is the linear combination

$$\mathbf{pc}_1 = a_{11}\mathbf{x}_1 + a_{12}\mathbf{x}_2 + \cdots + a_{1p}\mathbf{x}_p$$

and has the largest sample variance among all other PCs.

- In R, the `princomp()` function carries out a PCA via an eigendecomposition of the sample covariance matrix Σ.
- If variables are on very different scales, PCA should be carried out on the correlation matrix (which is equivalent to the correlation matrix if normalized variables are used).

PERFORM PCA USING `PRINCOMP()`

As the variables of the heptathlon data are on different scales, we perform the PCA based on the correlation matrix.

```
hept.pca = princomp(heptathlon[, -8], cor = TRUE)
```

Alternatively, we could also perform the PCA based on the covariance matrix but on the normalized heptathlon data.

The resulting object of `princomp()` contains at least:

- The loadings $\mathbf{a}_1, \ldots, \mathbf{a}_p$,
- The PC scores $\mathbf{pc}_1, \ldots, \mathbf{pc}_p$ and
- The variance $\lambda_1, \ldots, \lambda_p$ (or standard deviation) of the PC scores.

LOADINGS

The loadings $\mathbf{a}_1, \ldots, \mathbf{a}_p$ are contained in

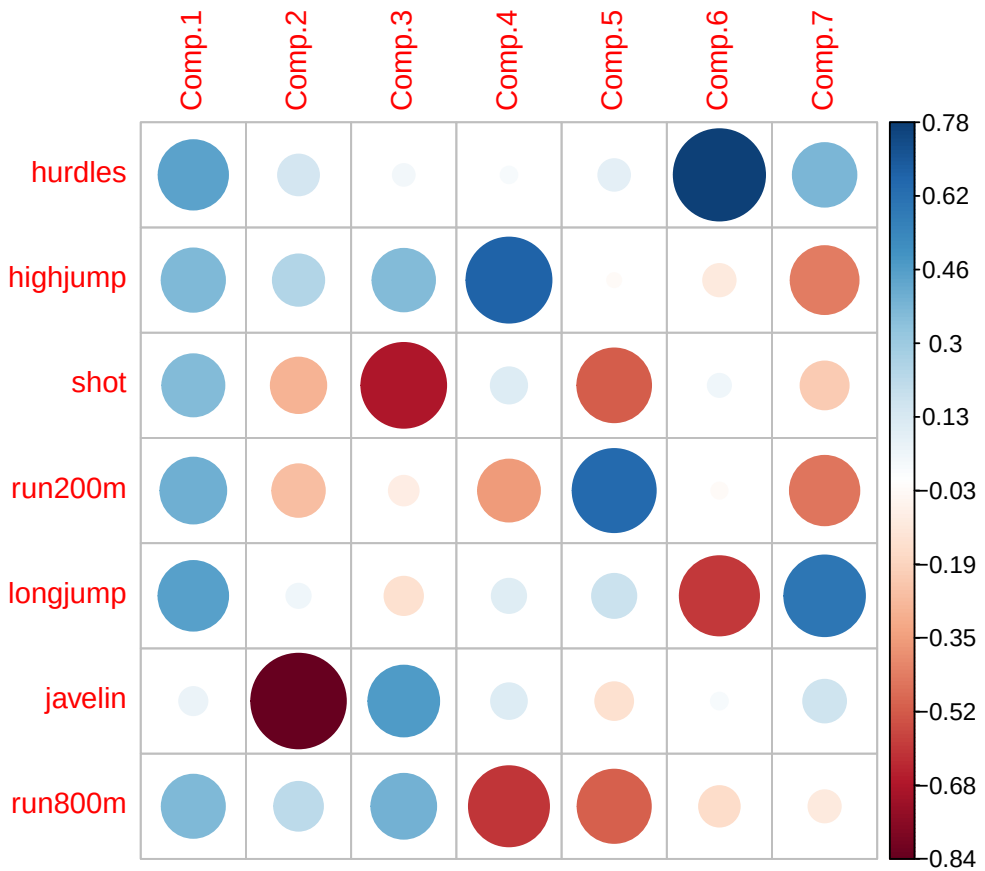
```
hept.pca$loadings
```

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7
hurdles	0.45	0.16	0.05	0.03	0.09	0.78	0.38
highjump	0.38	0.25	0.37	0.68	-0.02	-0.10	-0.43
shot	0.36	-0.29	-0.68	0.12	-0.51	0.05	-0.22
run200m	0.41	-0.26	-0.08	-0.36	0.65	-0.02	-0.45
longjump	0.46	0.06	-0.14	0.11	0.18	-0.59	0.61
javelin	0.08	-0.84	0.47	0.12	-0.14	0.03	0.17
run800m	0.37	0.22	0.40	-0.60	-0.50	-0.16	-0.10

LOADINGS

Visualize the coefficient weights (loadings) of the linear combinations of the PC scores:

```
corrplot(hept.pca$loadings, is.corr = FALSE)
```



LOADINGS

Check the normalization constraint of the first PC:

```
(a1 = hept.pca$loadings[, 1]) # loadings of the first PC

##  hurdles highjump     shot  run200m longjump  javelin  run800m
##  0.45287  0.37720  0.36307  0.40790  0.45623  0.07541  0.37496

a1 %*% a1 # check constraint: is sum of squares equal to 1egin{vbframe}?

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

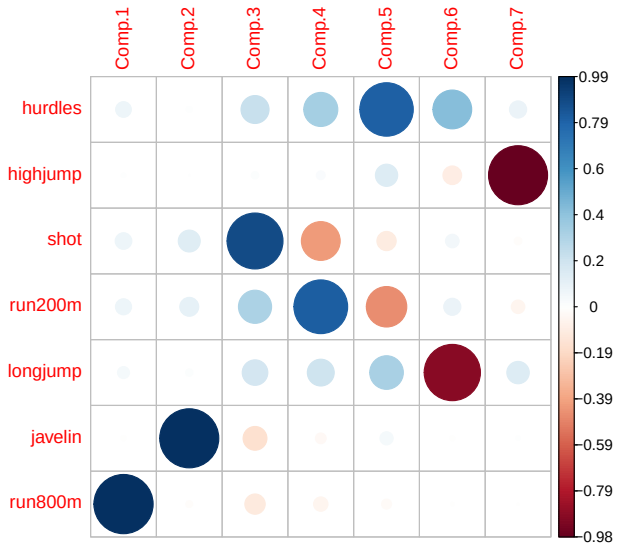
Check the orthogonality constraint of the first two PCs:

```
(a2 = hept.pca$loadings[, 2]) # loadings of the second PC

##  hurdles highjump     shot  run200m longjump  javelin  run800m
##  0.15792  0.24807 -0.28941 -0.26039  0.05587 -0.84169  0.22449

a1 %*% a2 # check if 1st and 2nd PC are orthogonal

##              [,1]
## [1,] 5.551e-17
```

LOADINGS

If we perform PCA on the covariance matrix (without normalizing the data), each component mainly loads on a single variable:

```
hept.pca.cov = princomp(heptathlon[, -8], cor = FALSE)
corrplot(hept.pca.cov$loadings, is.corr = FALSE)
```



Reason: Variables have very different scales (e.g., time measurement of 200m and 800m run).

PROPORTION OF EXPLAINED VARIANCE

- The total variance of the p PC scores is equal the total variance of the original variables, i.e.,

$$\sum_{j=1}^{p} \lambda_j = s_1^2 + s_2^2 + \cdots + s_p^2,$$

  where λ_j is the variance of the jth PC and s_j^2 is the sample variance of variable $\mathbf{x}_j$.

- The proportion of explained variance of the j-th PC is

$$\frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j}.$$

- The first k PCs account for a proportion

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j}.$$

PROPORTION OF EXPLAINED VARIANCE

In R, the proportion of explained variance can be obtained by

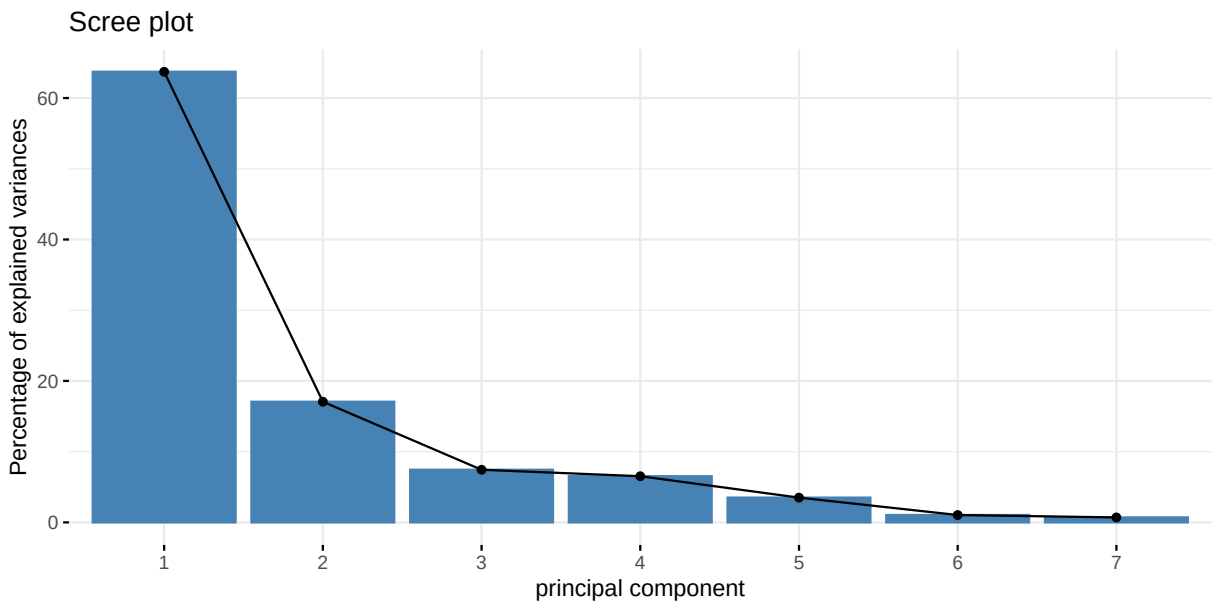
```
summary(hept.pca)

## Importance of components:
##                          Comp.1 Comp.2  Comp.3  Comp.4
## Standard deviation     2.1119 1.0928 0.72181 0.67614
## Proportion of Variance 0.6372 0.1706 0.07443 0.06531
## Cumulative Proportion  0.6372 0.8078 0.88223 0.94754
##                           Comp.5  Comp.6 Comp.7
## Standard deviation     0.49524 0.27010 0.2214
## Proportion of Variance 0.03504 0.01042 0.0070
## Cumulative Proportion  0.98258 0.99300 1.0000
```

Question: How do we choose the number of PCs?

CHOOSING THE NUMBER OF PCS

Two simple rules of thumb for choosing the number of PCs:

1. Retain the first k components, which explain a large proportion of the total variation, e.g., 70-80%.
2. Use a scree plot: Plot the component variances vs. the component number and look for an *elbow*. For components after the *elbow*, the variance decreases more slowly.



PC SCORES VS. OFFICIAL SCORES

The first PC explains $63,72\%$ of the variation, the loadings of the first PC are:

```
hept.pca$loadings[,1]

##  hurdles highjump     shot  run200m longjump  javelin  run800m
##  0.45287  0.37720  0.36307  0.40790  0.45623  0.07541  0.37496
```

Dimensionality reduction:

- Project all 8 features onto the first PC.
- Compare the scores of the first PC with the official scores used to rank the athletes.

PC SCORES VS. OFFICIAL SCORES

The scores of the first PC $\mathbf{pc}_1$ have a similar ranking as the scores of the official scoring system, i.e., we can reduce the dimension to the first PC without losing much information: