DATA2002: Lecture 32

Dimension reduction

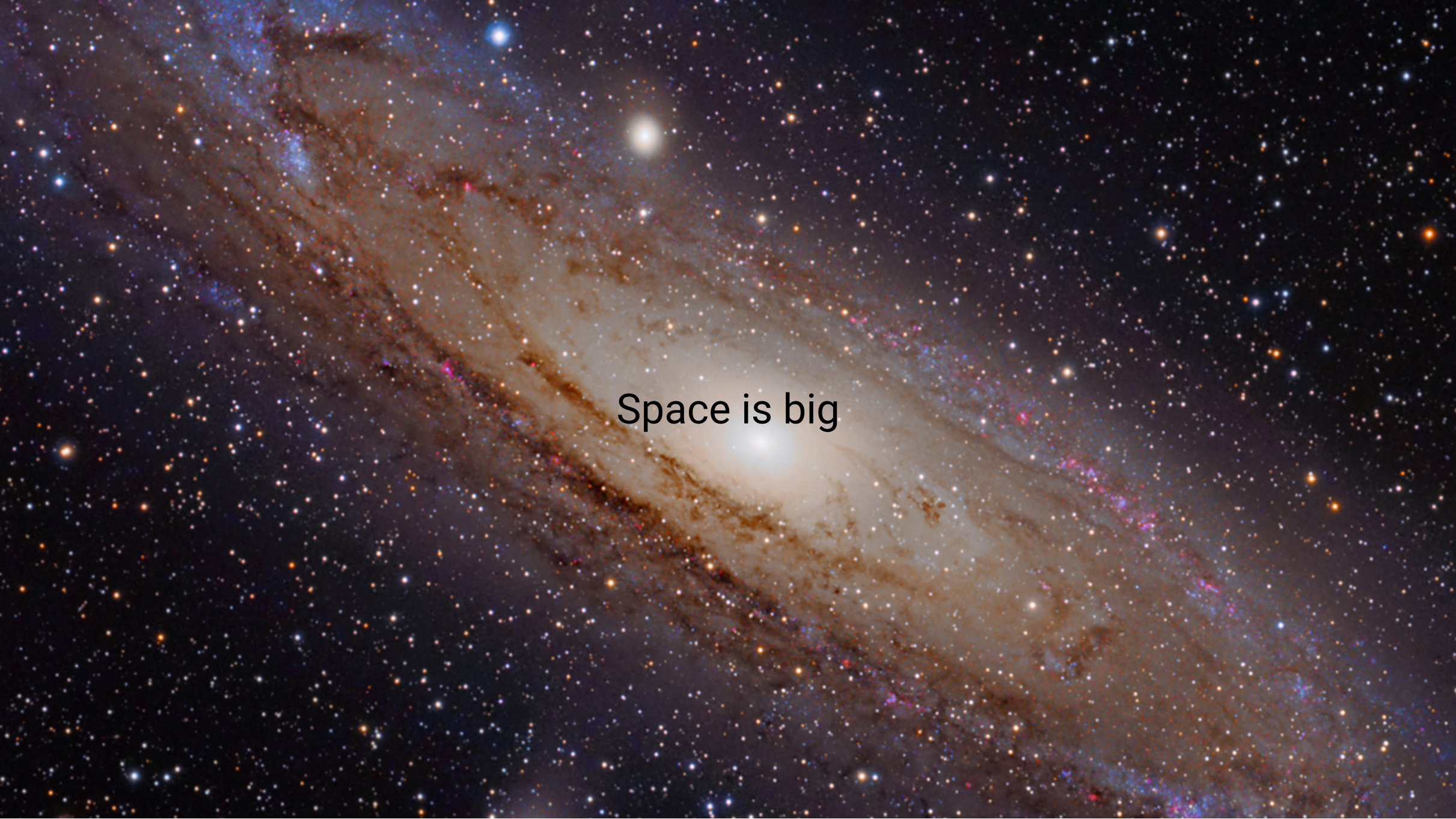
Garth Tarr

THE UNIVERSITY OF SYDNEY

Principal component analysis

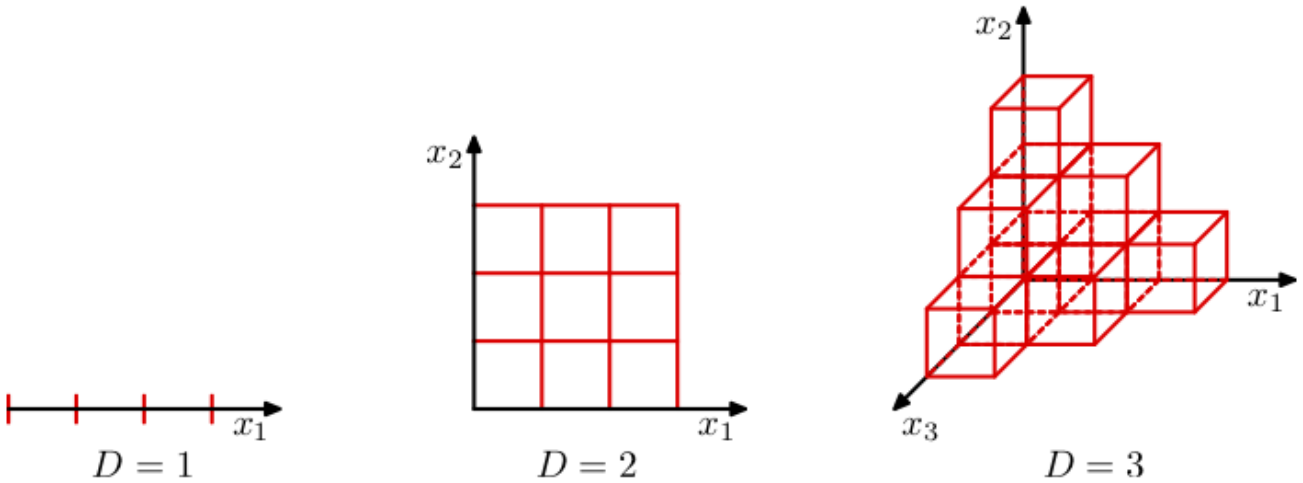
Eigenfaces

Dimension reduction + clustering

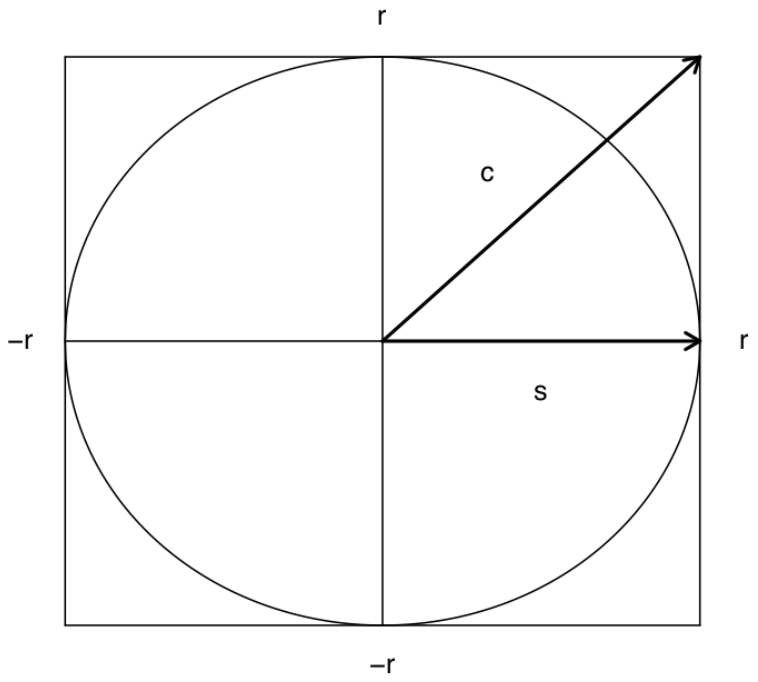

Space is big

Cubes and spheres

Space expands exponentially with dimension:



As dimension increases the volume of a sphere of same radius as cube side length becomes much smaller than the volume of the cube:



The curse of dimensionality 🎃👻

Sub-spaces

- Data will often be confined to a region of the space having lower intrinsic dimensionality.
- The data lives in a low-dimensional subspace.
- The goal is to reduce the dimensionality, to the subspace containing the data.

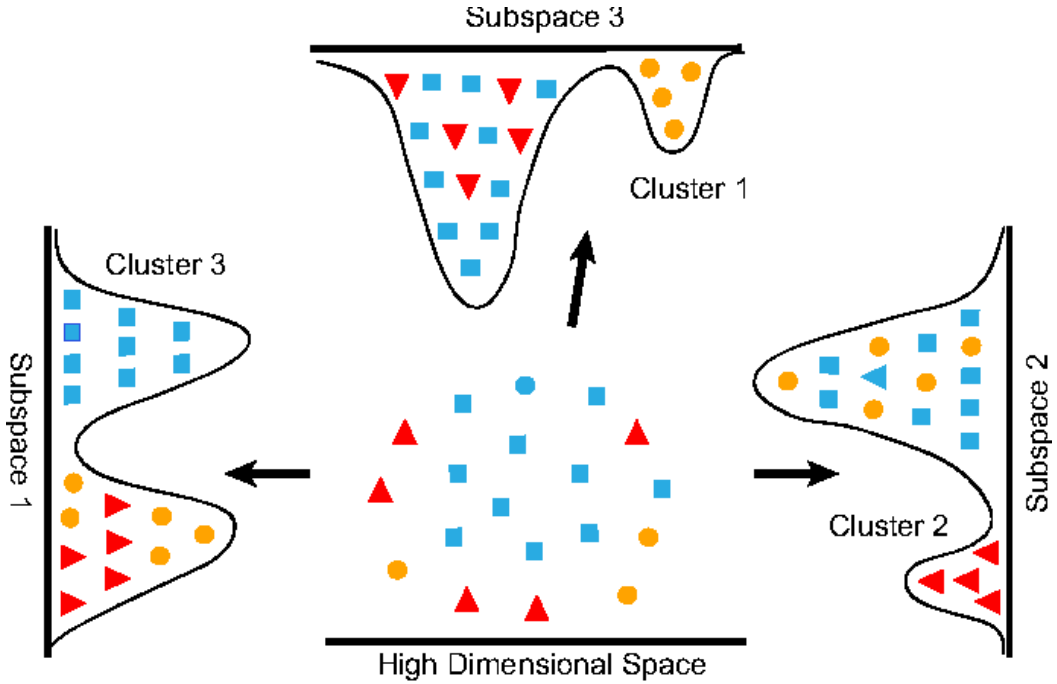


Illustration of clustering in subspaces. Separation of clusters in appropriate selection of dimension subspaces can be much easier than that in the original high dimensional space.

Source: Yuan, Ren, Wang, et al. (2013)

PCA

- Principal component analysis (PCA) produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated.
- It is an **unsupervised learning** method.

Why?

- We may have too many predictors for a **regression**. Instead, we can use the first few principal components. PCA regression.
- **Understanding relationships** between variables - similar to a correlation matrix.
- **Data visualisation**: we can plot a small number of variables more easily than a large number of variables.

PCA: Definition

First principal component

The first principal component of a set of variables $x_1, x_2, \ldots, x_p$ is the linear combination

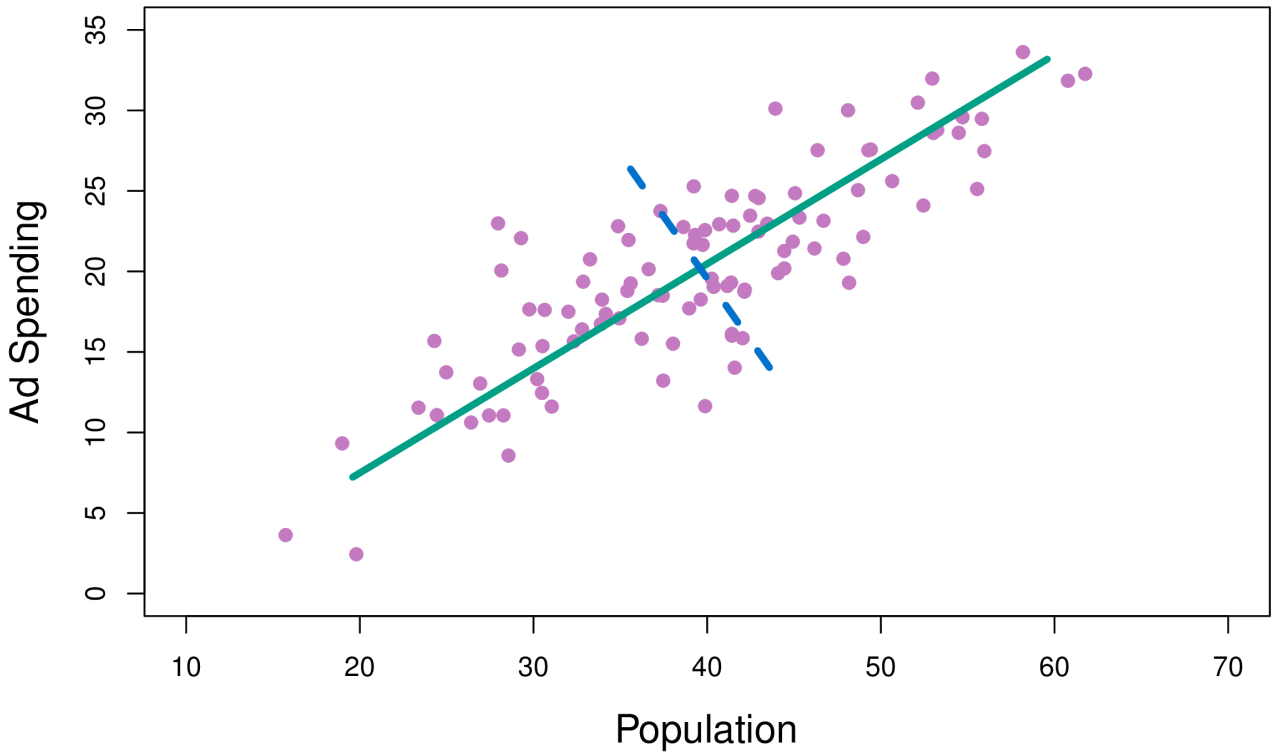
$$z_1 = \phi_{11} x_1 + \phi_{21} x_2 + \cdots + \phi_{p1} x_p$$

that has the largest variance such that

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

The elements $\phi_{11}, \ldots, \phi_{p1}$ are the loadings of the first principal component.

Example

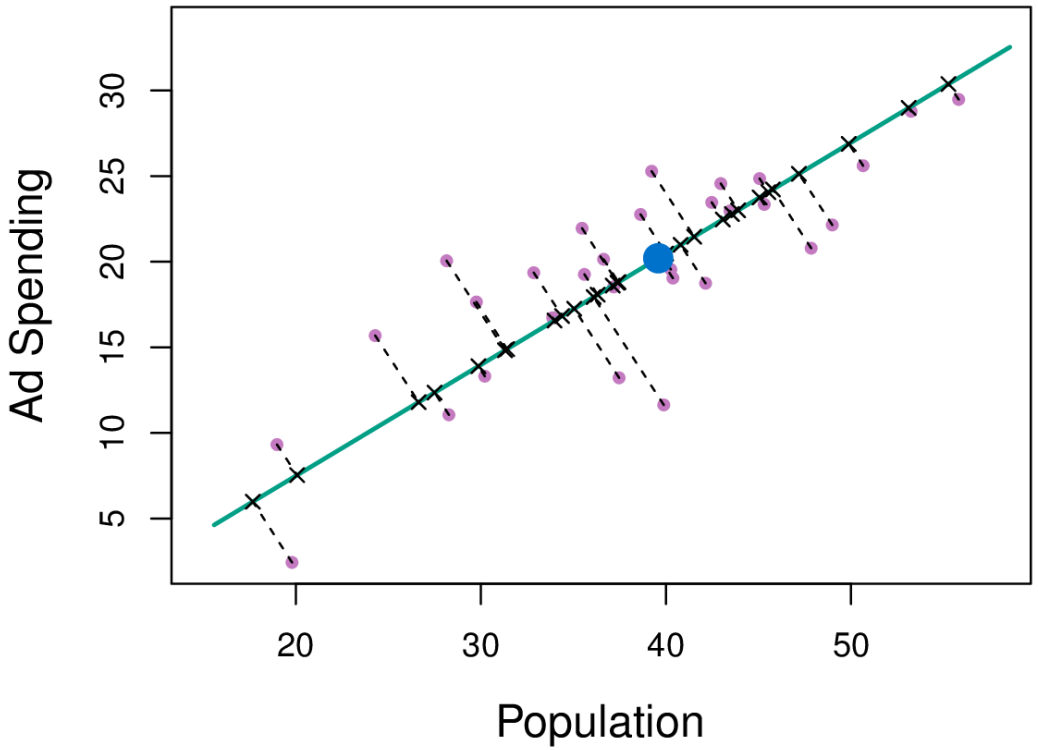


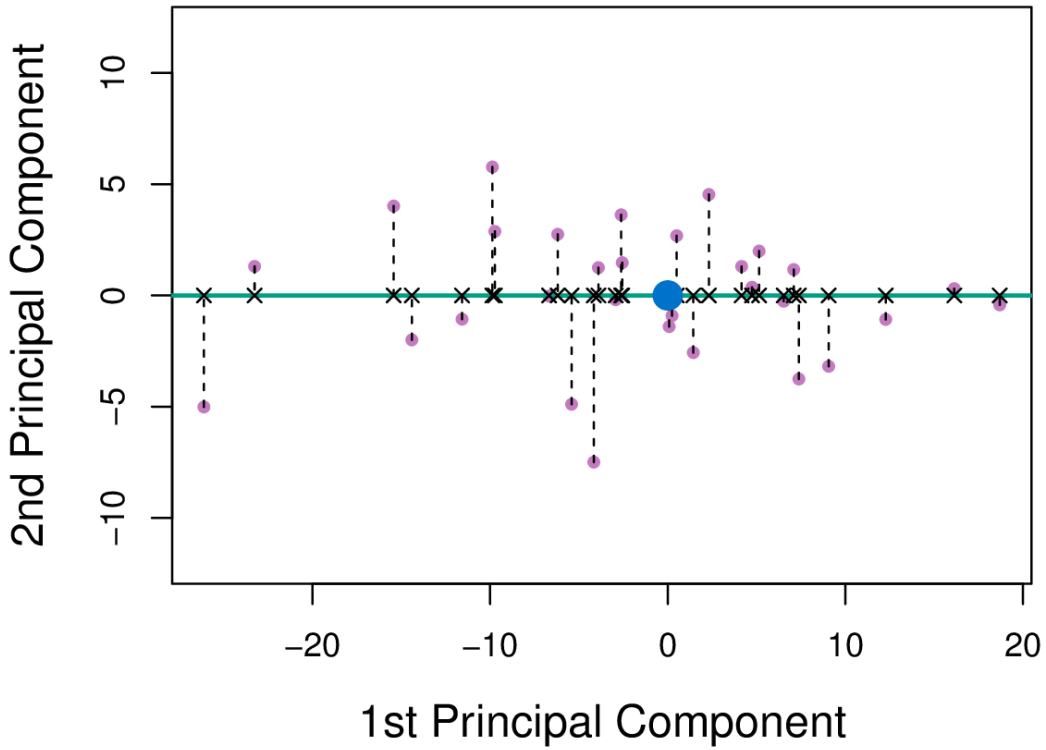
First PC; second PC

Source: James, Witten, Hastie, et al. (2017; Figure 6.14)

PCA: Geometry

- The loading vector $\boldsymbol{\phi}_1 = [\phi_{11}, \ldots, \phi_{p1}]'$ defines direction in feature space along which data vary most.
- If we **project** the n data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ onto this direction, the projected values are the principal component scores $\boldsymbol{z}_1 = [z_{11}, \ldots, z_{n1}]'$.
- The second principal component is the linear combination $z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \cdots + \phi_{p2}x_{ip}$ that has maximal variance among all linear combinations that are *uncorrelated* with $\boldsymbol{z}_1$.
- Equivalent to constraining $\boldsymbol{\phi}_2$ to be orthogonal (perpendicular) to $\boldsymbol{\phi}_1$. And so on.
- There are at most $\min(n-1, p)$ principal components.



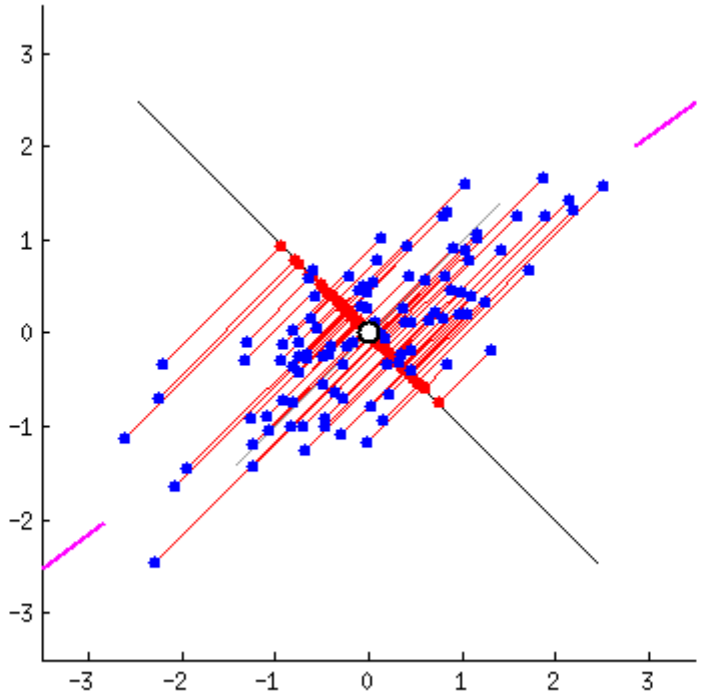


If you think of the first few PCs like a linear model fit, and the others as the error, it is like regression, except that errors are orthogonal to model.

Image source: James, Witten, Hastie, et al. (2017; Figure 6.15)

PCA explained

There's a super nice answer on CrossValidated that explains PCA in language suitable for your family.



PCA projects data onto a new **coordinate system** where the principal component vectors are **orthogonal** (independent) to each other. The first PC tries to account for as much variation as possible in the data.

Total variance

Total variance in data (assuming variables centered at 0):

$$\mathrm{TV} = \sum_{j=1}^{p} \mathrm{Var}(x_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

If variables are standardised, TV=number of variables!

Variance explained by mth principal component:

$$V_m = \mathrm{Var}(z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2$$

$$\mathrm{TV} = \sum_{m=1}^{M} V_m \text{ where } M = \min(n-1, p).$$

Choosing the number of PCs

- Proportion of variance explained:

$$\text{PVE}_m = \frac{V_m}{\text{TV}}$$

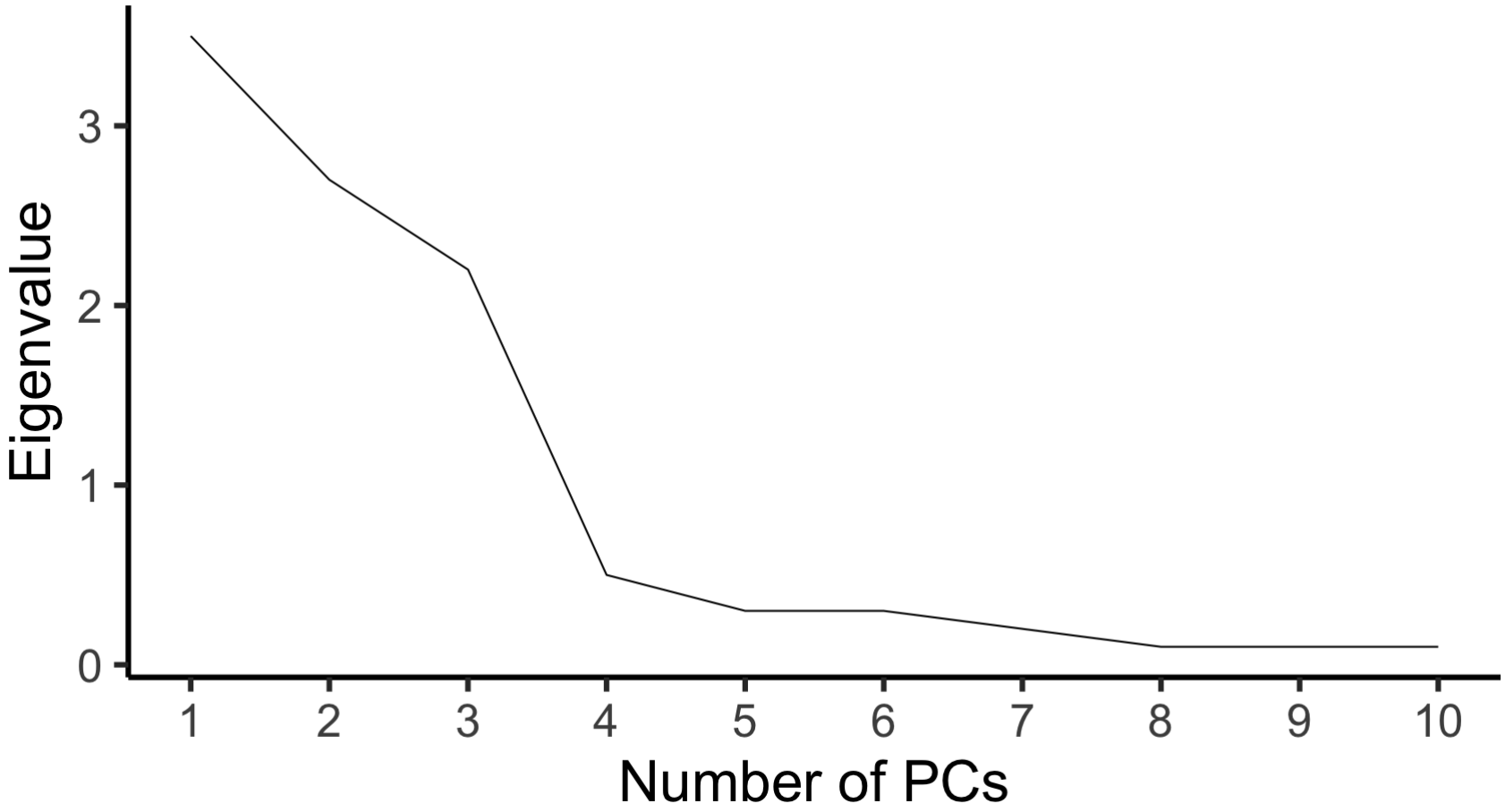
- Choosing the number of PCs that adequately summarises the variation in $\boldsymbol{X}$, is achieved by examining the cumulative proportion of variance explained.
- Cumulative proportion of variance explained:

$$\text{CPVE}_k = \sum_{m=1}^{k} \frac{V_m}{\text{TV}}$$

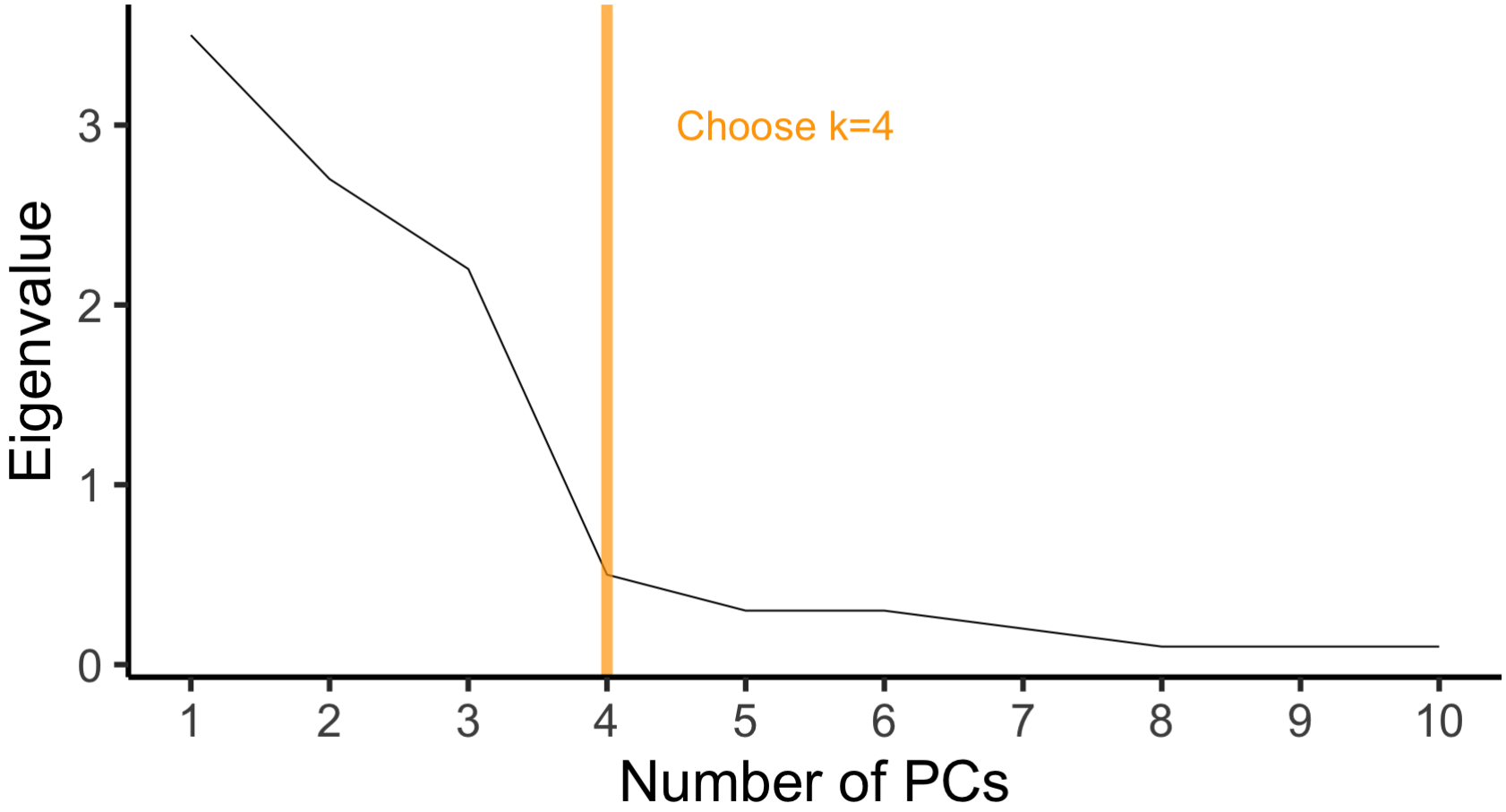
and also by a scree plot.

Image: Scree slope at the bottom of Yamnuska, Alberta, Canada. Source: Wikimedia Commons

Choosing the number of PCs: scree plot



Choosing the number of PCs: scree plot



National track records

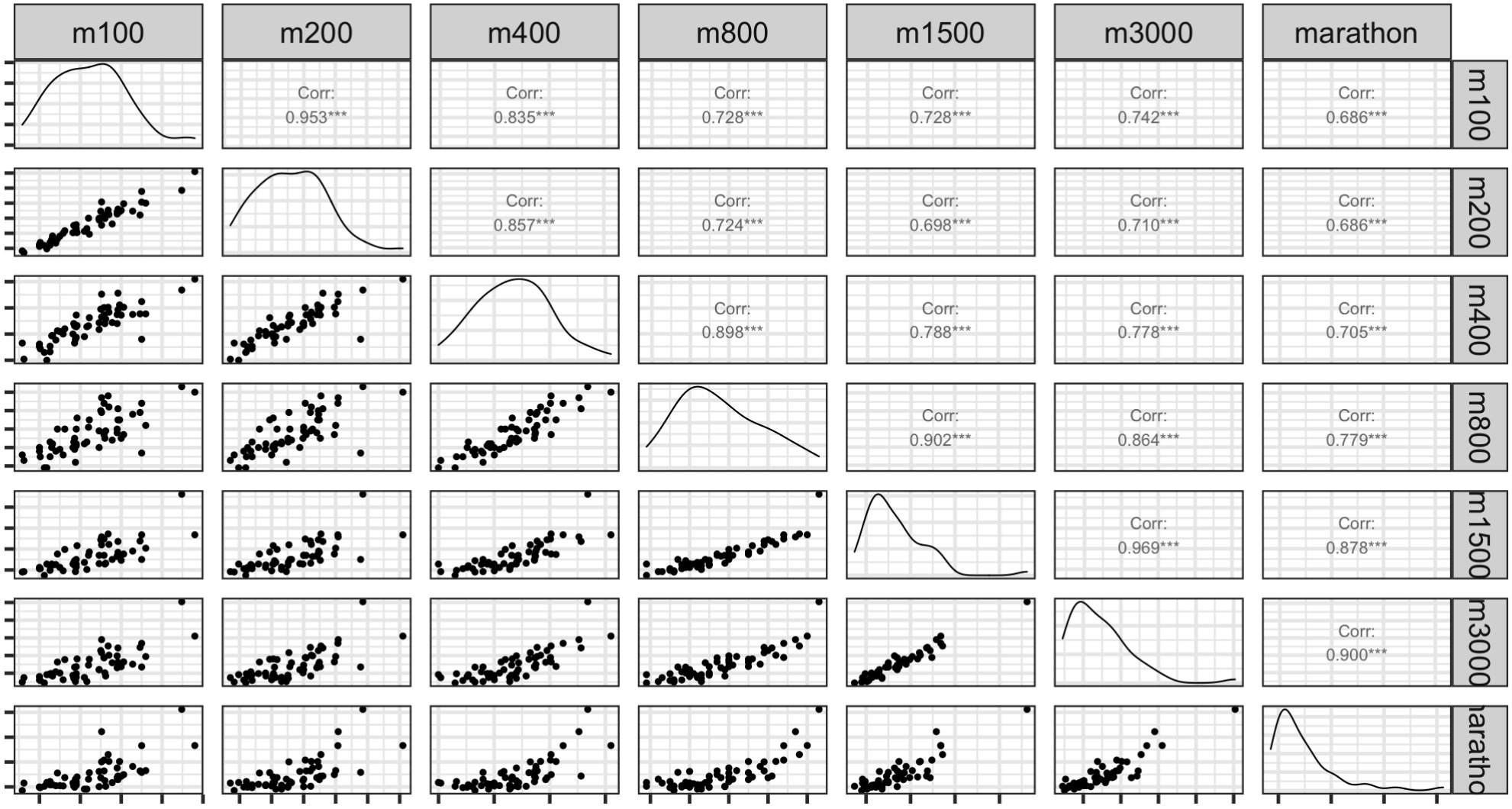
Example: the data on national track records for women (as at 1984).

```
track = readr::read_csv("data/womens_track.csv")
dplyr::glimpse(track)
```

```
## Rows: 55
## Columns: 8
## $ m100     <dbl> 11.61, 11.20, 11.43, 11.41, 11.46, 11.31,…
## $ m200     <dbl> 22.94, 22.35, 23.09, 23.04, 23.05, 23.17,…
## $ m400     <dbl> 54.50, 51.08, 50.62, 52.00, 53.30, 52.80,…
## $ m800     <dbl> 2.15, 1.98, 1.99, 2.00, 2.16, 2.10, 2.18,…
## $ m1500    <dbl> 4.43, 4.13, 4.22, 4.14, 4.58, 4.49, 4.45,…
## $ m3000    <dbl> 9.79, 9.08, 9.34, 8.88, 9.81, 9.77, 9.51,…
## $ marathon <dbl> 178.52, 152.37, 159.37, 157.85, 169.98, 1…
## $ country  <chr> "argentin", "australi", "austria", "belgi…
```

Source: Yuan, Ren, Wang, et al. (2013)

```
GGally::ggpairs(track[,1:7]) + theme_bw(base_size = 24) + theme(axis.text = element_blank())
```



Compute

```
track_pca = prcomp(track[,1:7], center = TRUE, scale = TRUE)
track_pca
```

```
## Standard deviations (1, .., p=7):
## [1] 2.41 0.81 0.55 0.35 0.23 0.20 0.15
##
## Rotation (n x k) = (7 x 7):
##           PC1   PC2    PC3    PC4    PC5     PC6    PC7
## m100     0.37  0.49 -0.286  0.319  0.231  0.6198  0.052
## m200     0.37  0.54 -0.230 -0.083  0.041 -0.7108 -0.109
## m400     0.38  0.25  0.515 -0.347 -0.572  0.1909  0.208
## m800     0.38 -0.16  0.585 -0.042  0.620 -0.0191 -0.315
## m1500    0.39 -0.36  0.013  0.430  0.030 -0.2312  0.693
## m3000    0.39 -0.35 -0.153  0.363 -0.463  0.0093 -0.598
## marathon 0.37 -0.37 -0.484 -0.672  0.131  0.1423  0.070
```

Assess

```
summary(track_pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.409 0.8085 0.5476 0.3542 0.23198 0.19761 0.14981
## Proportion of Variance 0.829 0.0934 0.0428 0.0179 0.00769 0.00558 0.00321
## Cumulative Proportion  0.829 0.9228 0.9656 0.9835 0.99122 0.99679 1.00000
```

Summary of the principal components:

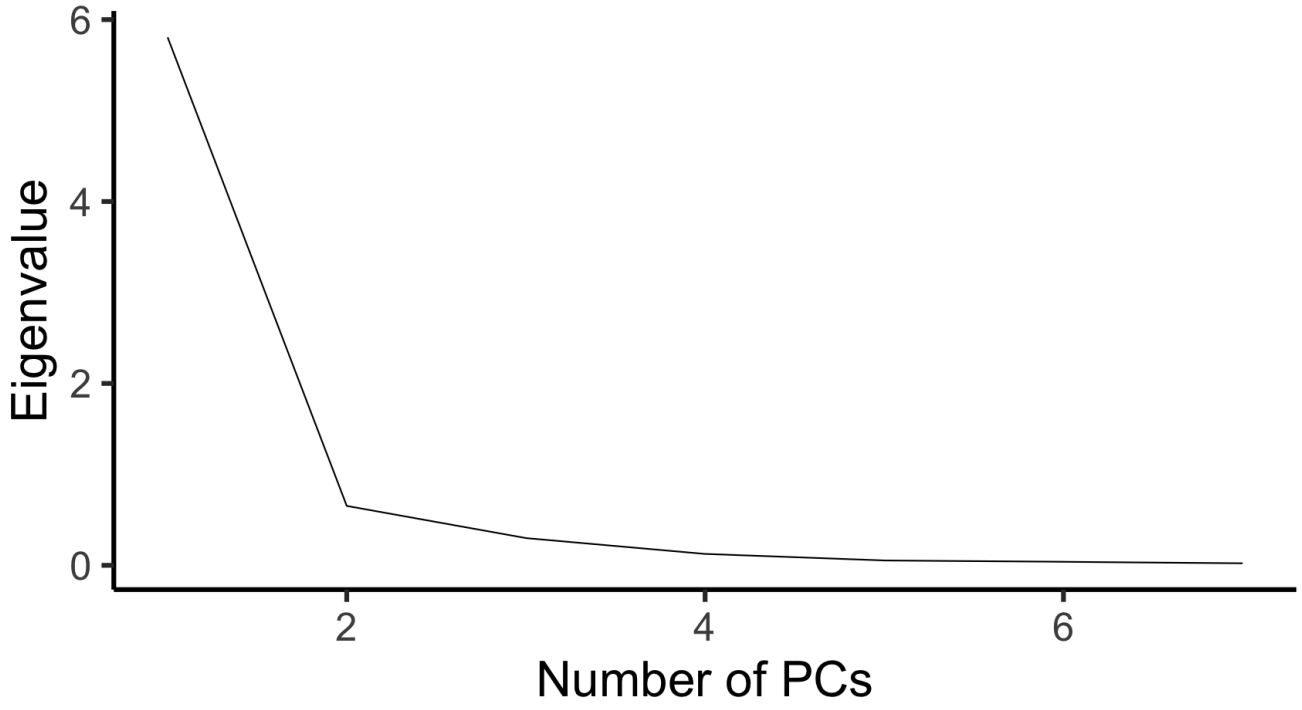
	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Variance	5.81	0.65	0.30	0.13	0.05	0.04	0.02
Proportion	0.83	0.09	0.04	0.02	0.01	0.01	0.00
Cum. prop	0.83	0.92	0.97	0.98	0.99	1.00	1.00

Increase in variance explained large until $k = 3$ PCs, and then tapers off. A choice of 3 PCs would explain 97% of the total variance.

Assess

Scree plot: Where is the elbow?

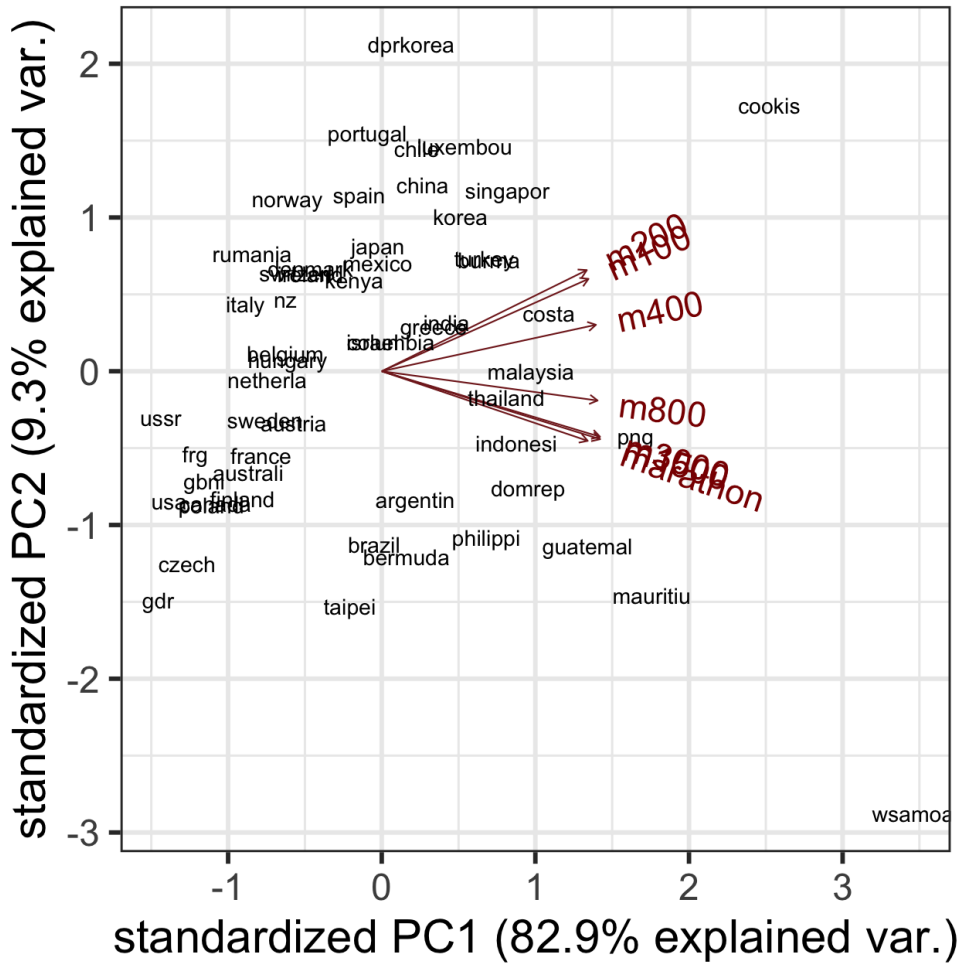
```
track_pca_var = tibble(n=1:length(track_pca$sdev), evl=track
ggplot(track_pca_var, aes(x=n, y=evl)) + geom_line() +
  xlab("Number of PCs") + ylab("Eigenvalue")
```



At $k = 2$, the scree plot suggests 2 PCs would be sufficient to explain the variability.

Visualise

Visualise model using a **biplot**: plot the principal component scores, and also the contribution of the original variables to the principal component.



Interpret

Explain and interpret: using the coefficients of the principal components.

- PC1 measures overall magnitude, the strength of the athletics program. High positive values indicate poor programs with generally slow times across events.
- PC2 measures the contrast in the program between short and long distance events. Some countries have relatively stronger long distance athletes, while others have relatively stronger short distance athletes.
- There are several outliers visible in this plot, `wsamoa`, `cookis`, `dpkorea`. PCA, because it is computed using the variance in the data, can be affected by outliers. It may be better to remove these countries, and re-run the PCA.

Related problems

You have multivariate variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ so $\boldsymbol{x}_1 = (x_{11}, \ldots, x_{1p})$

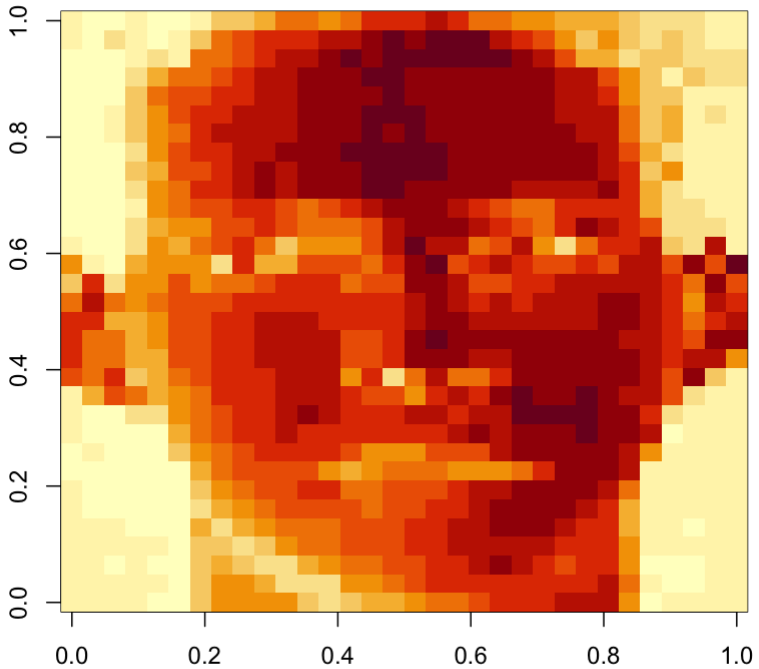
- Find a new set of multivariate variables that are uncorrelated and explain as much variance as possible.
- If you put all the variables together in one matrix, find the best matrix created with fewer variables (lower rank) that explains the original data.

The first goal is **statistical** and the second goal is **data compression** .

Eigenfaces

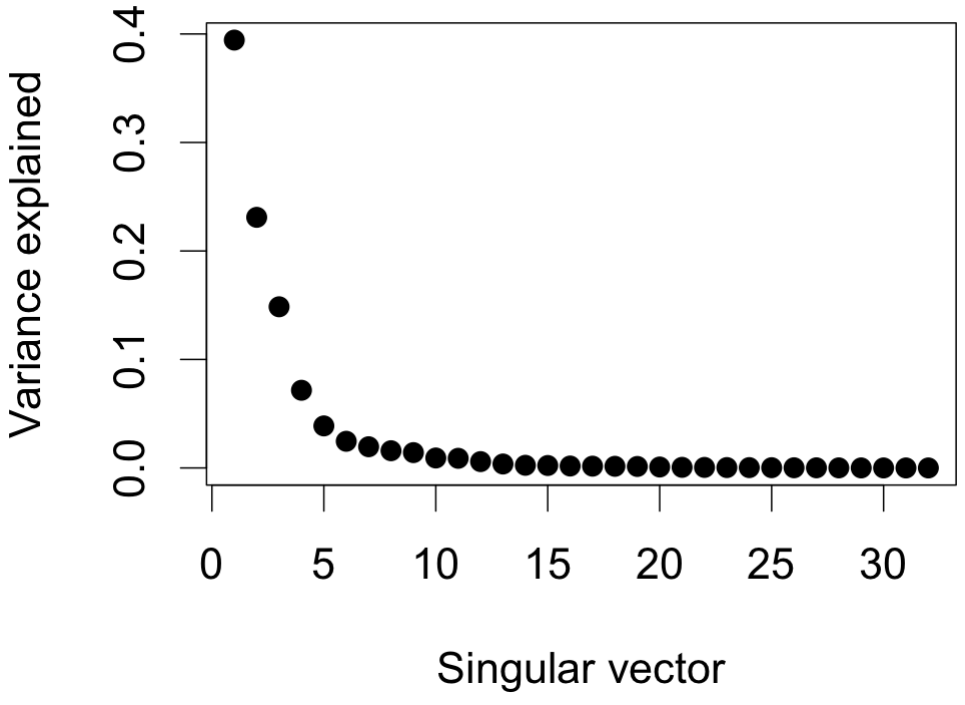
Face example

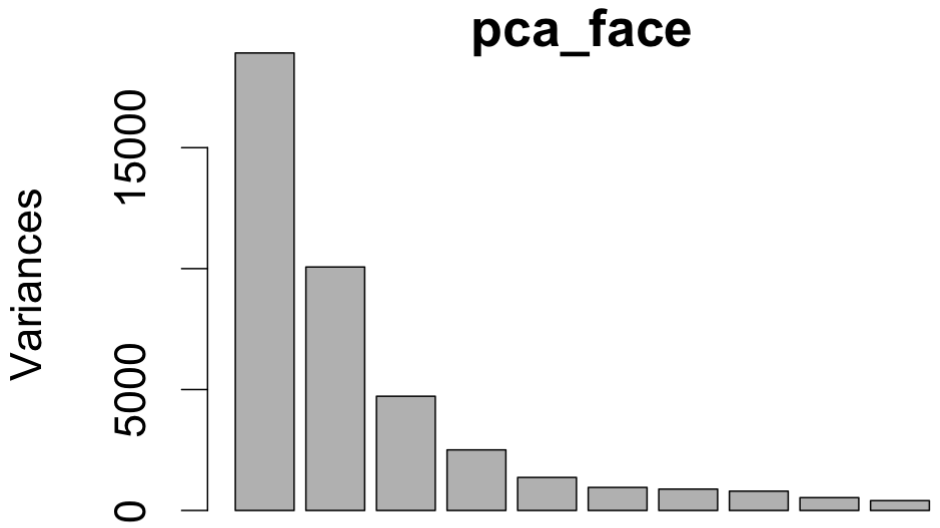
```
load(url("https://github.com/DATA2002/data/raw/master/face.rda"))
image(t(faceData)[,nrow(faceData):1])
```



Face example - variance explained

```
svd_face = svd(scale(faceData))
pca_face = prcomp(faceData)
par(mfrow=c(1,2), cex = 1.8, mar = c(4,4,1,1))
plot(svd_face$d^2/sum(svd_face$d^2),pch=19,xlab="Singular vector",ylab="Variance explained")
screeplot(pca_face)
```





Face example - create approximations

We can reconstruct the face using lower dimensional approximations.

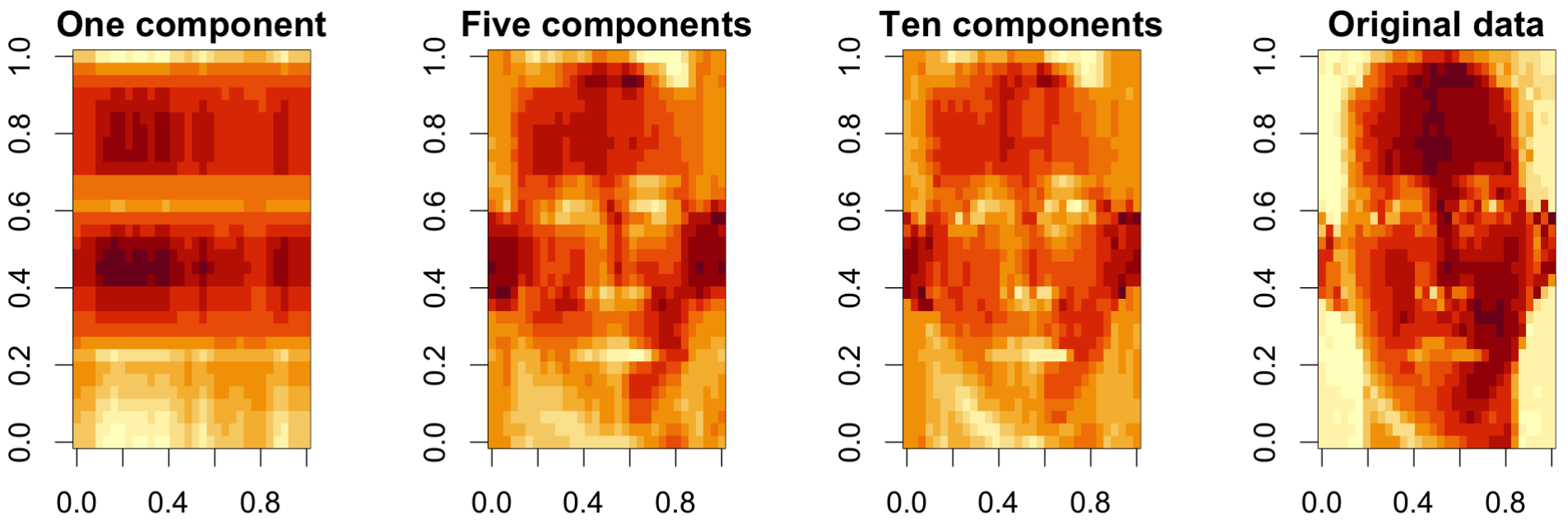
```
svd1 = svd(scale(faceData))
## Note that %*% is matrix multiplication

# Here svd1$d[1] is a constant
approx1 = svd1$u[,1] %*% t(svd1$v[,1]) * svd1$d[1]

# In these examples we need to make the diagonal matrix out of d
approx5 = svd1$u[,1:5] %*% diag(svd1$d[1:5]) %*% t(svd1$v[,1:5])
approx10 = svd1$u[,1:10] %*% diag(svd1$d[1:10]) %*% t(svd1$v[,1:10])
```

Face example - plot approximations

```
par(mfrow=c(1,4), mar = c(4,4,1.5,1), cex = 1.5)
image(t(approx1)[,nrow(approx1):1], main = "One component")
image(t(approx5)[,nrow(approx5):1], main = "Five components")
image(t(approx10)[,nrow(approx10):1], main = "Ten components")
image(t(faceData)[,nrow(faceData):1], main = "Original data")  ## Original data
```



Dimension reduction and clustering

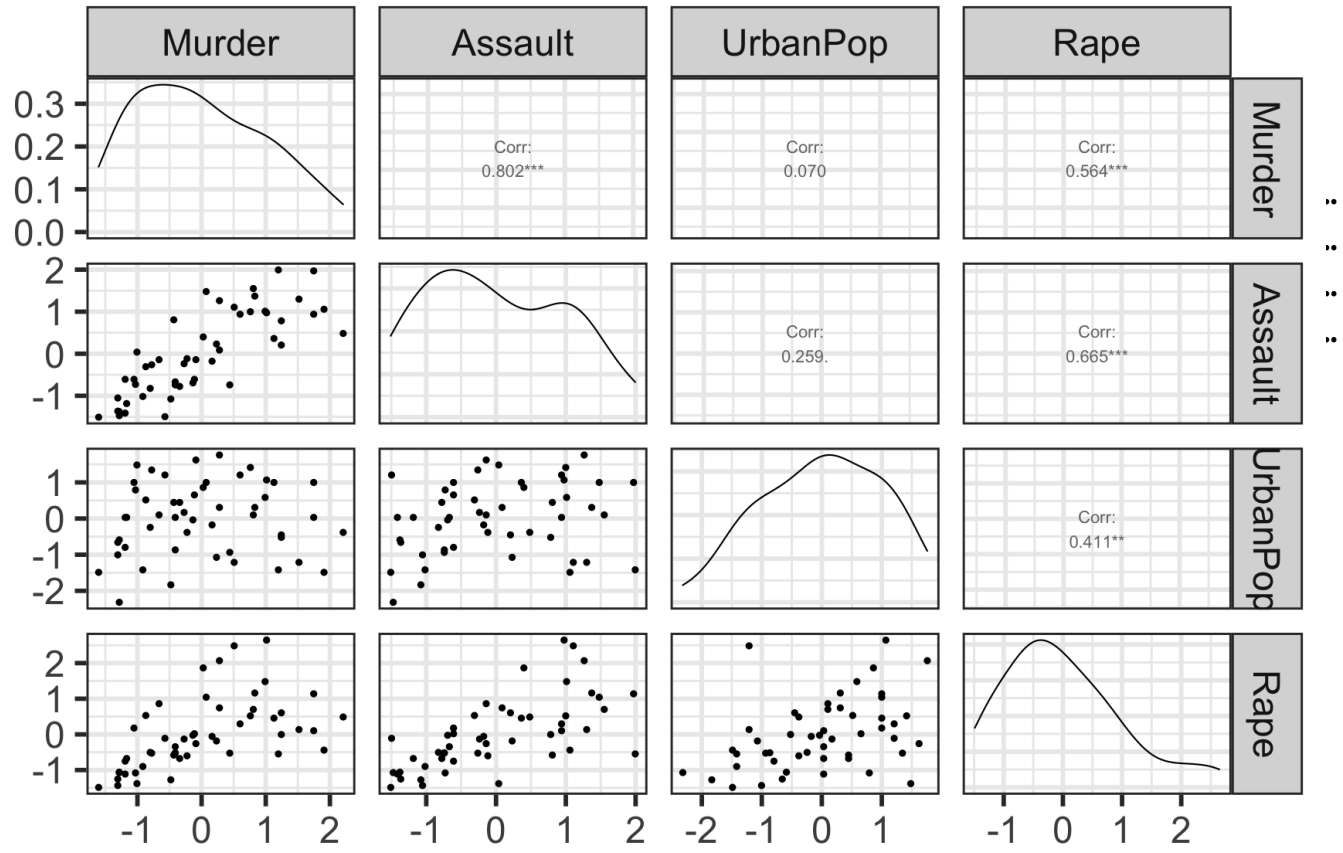
US arrests data

```
library(tidyverse)
data("USArrests")
glimpse(USArrests)
```

```
## Rows: 50
## Columns: 4
## $ Murder   <dbl> 13.2, 10.0, 8.1, 8.8,
## $ Assault  <int> 236, 263, 294, 190, 2
## $ UrbanPop <int> 58, 48, 80, 50, 91, 7
## $ Rape     <dbl> 21.2, 44.5, 31.0, 19.
```

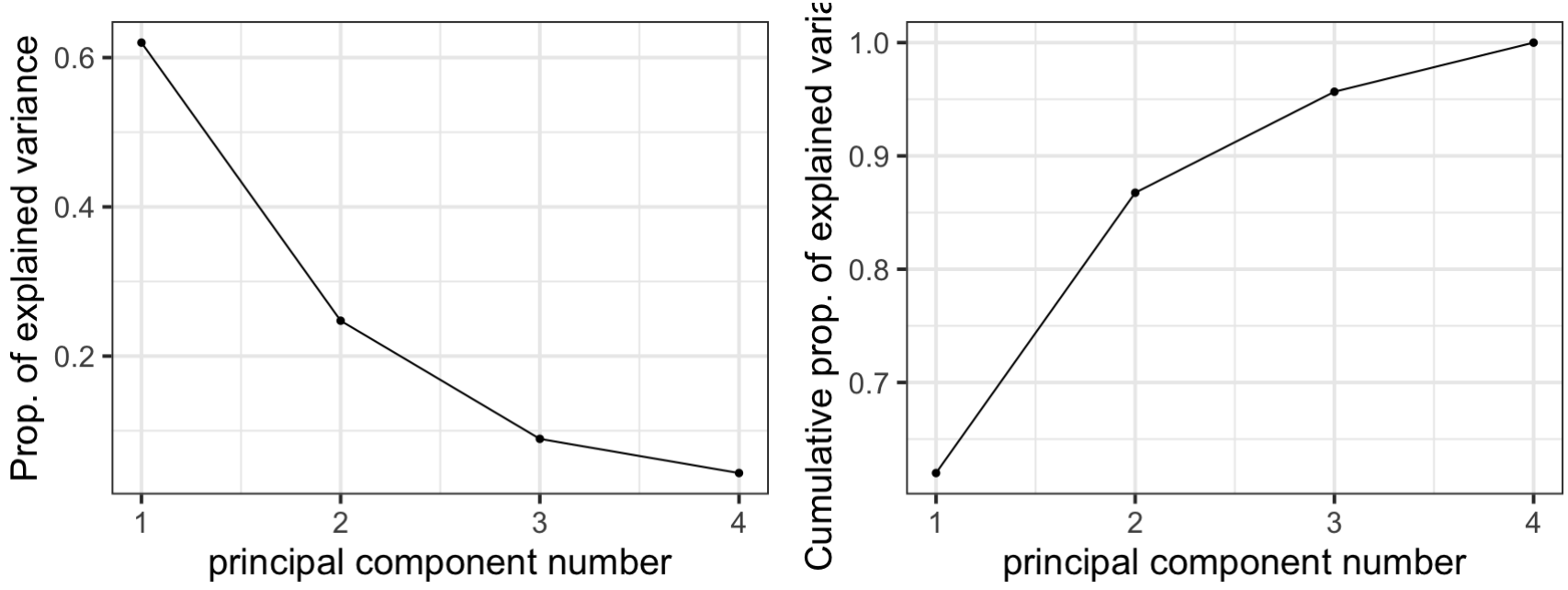
```
df = data.frame(scale(USArrests))
```

```
GGally::ggpairs(df) + theme_bw(base_size = 30)
```



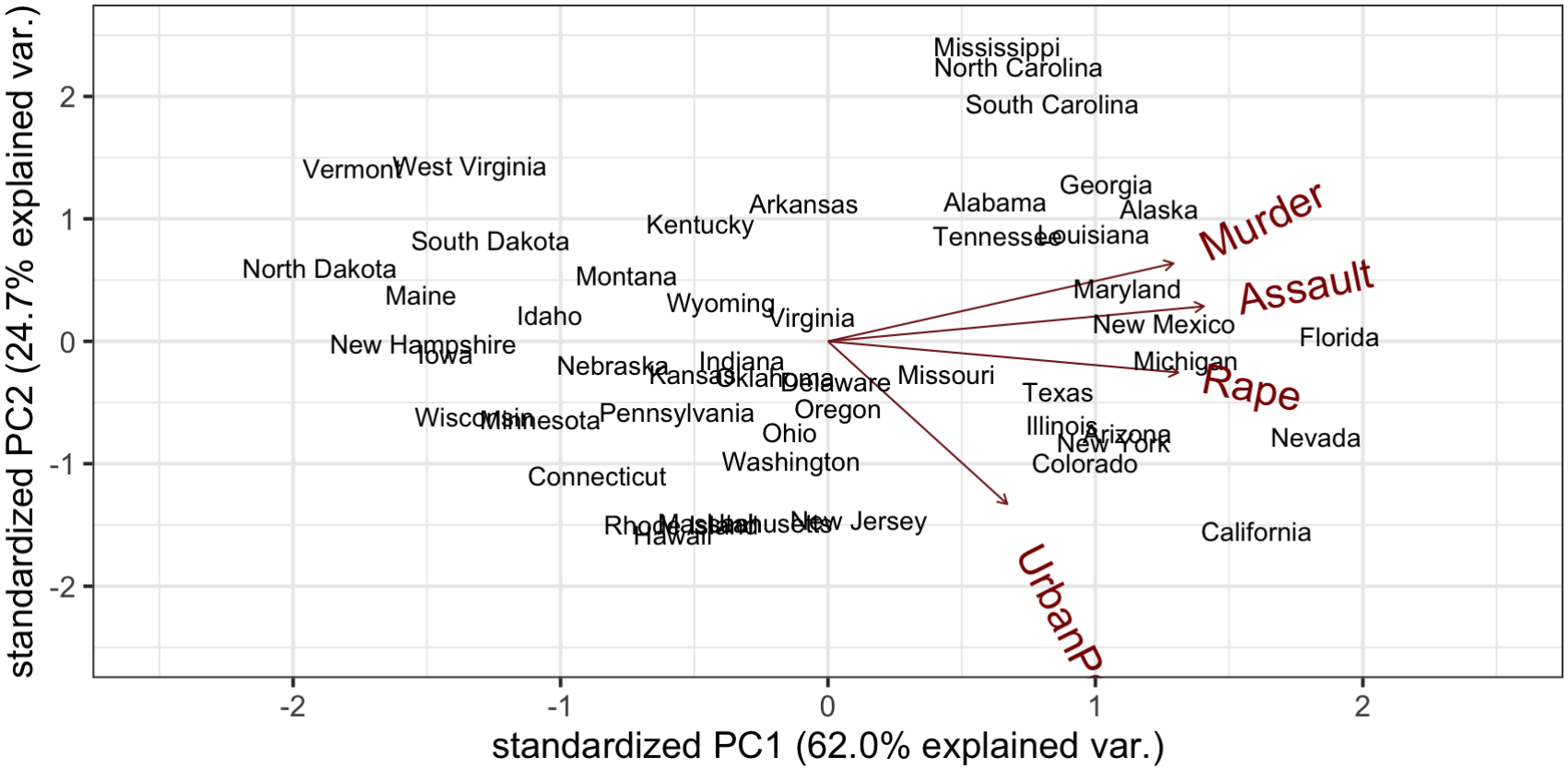
PCA for US arrests

```
pca_arrests = princomp(df)
# install.packages("remotes")
# remotes::install_github("vqv/ggbiplot")
library(ggbiplot)
p1 = ggscreeplot(pca_arrests) +
  theme_bw(base_size = 20) + labs(y = "Prop. of explained variance")
p2 = ggscreeplot(pca_arrests,type = "cev") + theme_bw(base_size = 20) + labs(y = " Cumulative prop. o
gridExtra::grid.arrange(p1, p2, ncol=2)
```



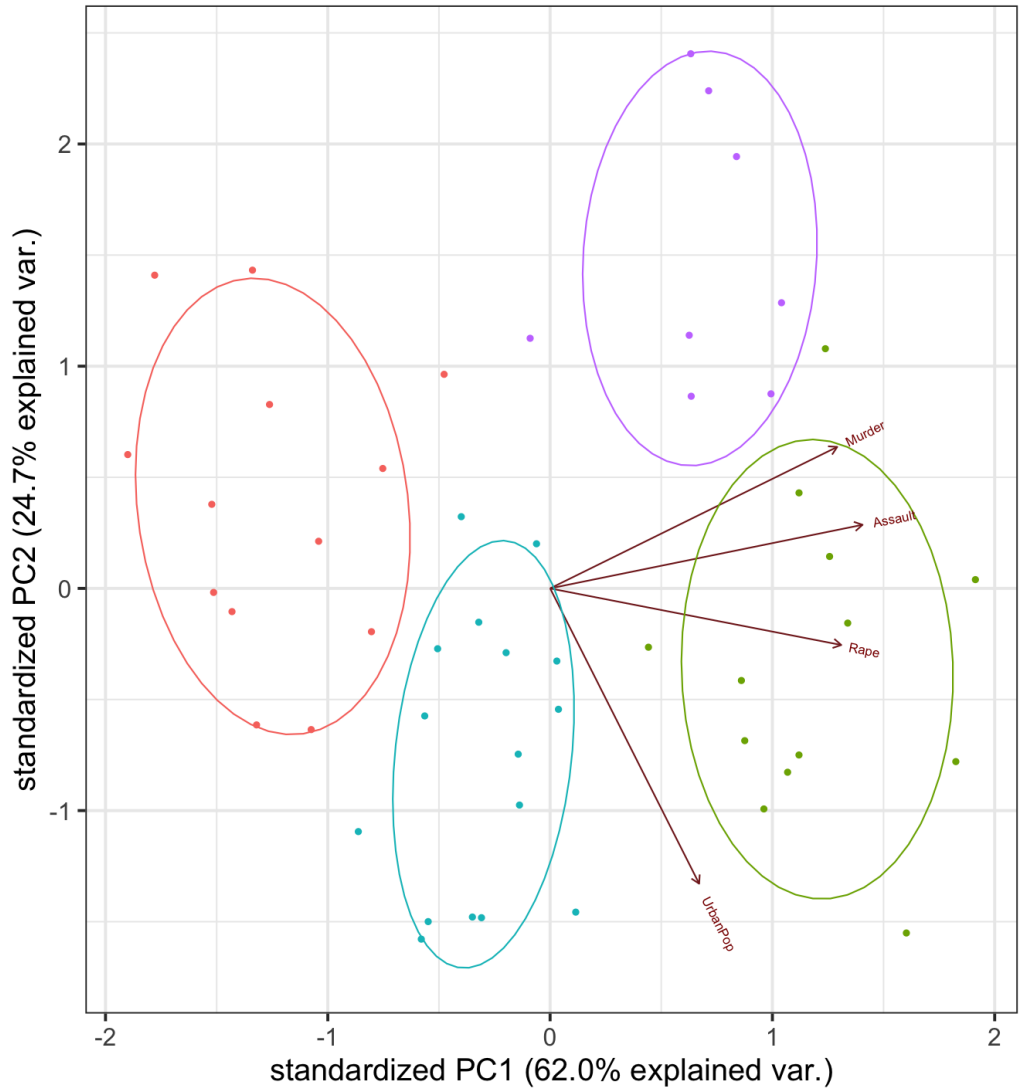
Biplot

```
ggbiplot(pca_arrests, labels =  rownames(USArrests), labels.size = 5, varname.size = 8) +
  theme_bw(base_size = 20) + coord_cartesian(ylim = c(-2.5,2.5), xlim = c(-2.5,2.5))
```

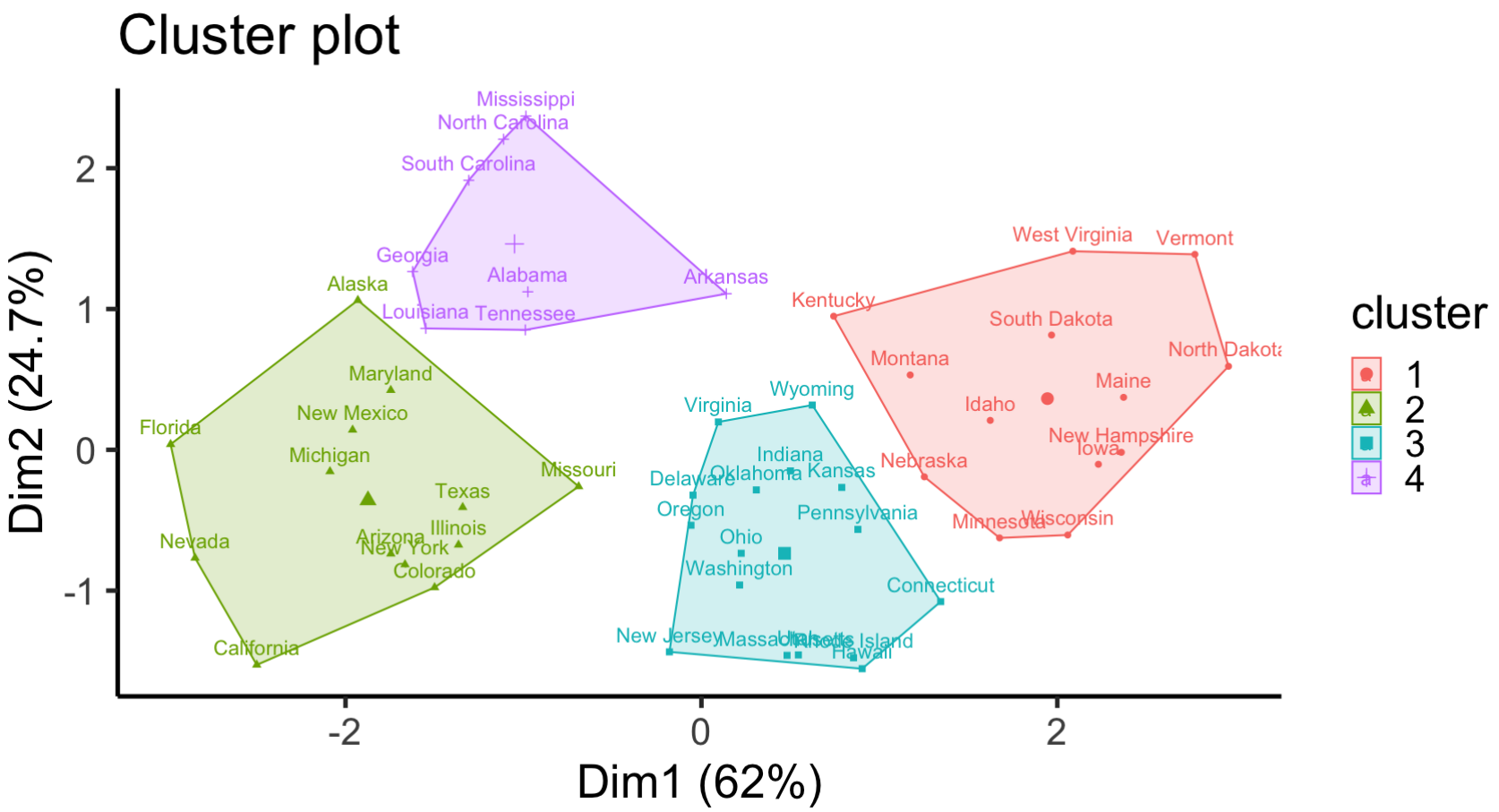


Clustering

```
set.seed(1)
cl_arrests = kmeans(
  df, centers = 4, nstart = 10
)
p = ggbiplot(
  pca_arrests,
  groups = factor(cl_arrests$cluster
  ellipse = TRUE) +
  theme_bw(base_size = 20) +
  theme(legend.position = "none")
p
```



```
library(factoextra)
fviz_cluster(cl_arrests, data = df) + theme_classic(base_size = 26)
```

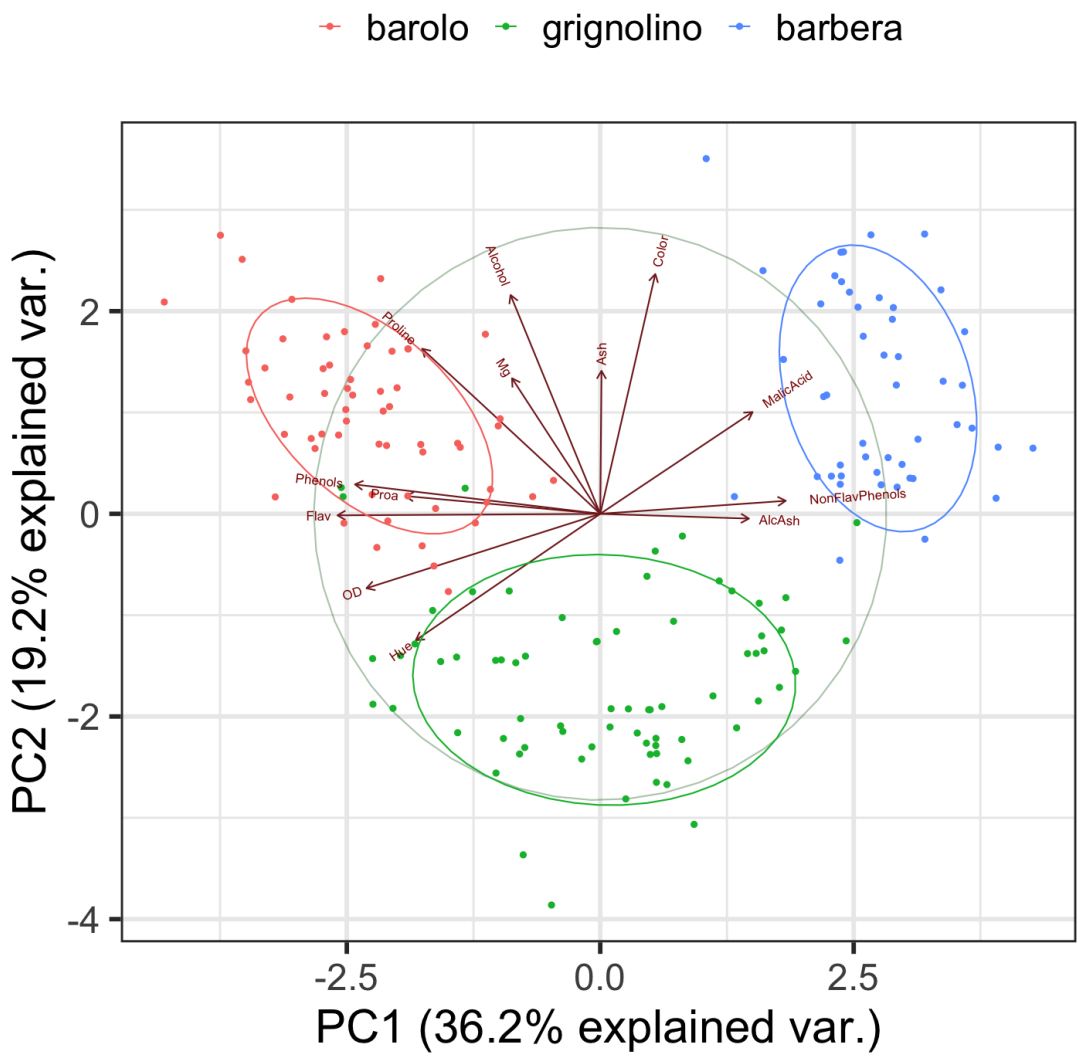


Wine

```
data(wine)
glimpse(wine)
```

```
## Rows: 178
## Columns: 13
## $ Alcohol        <dbl> 14, 13, 13, 14, 13, 14, 14, 14, 15, 14, 14, 14, 14, 15, 14, 14, 14, 14, 14,…
## $ MalicAcid      <dbl> 1.7, 1.8, 2.4, 1.9, 2.6, 1.8, 1.9, 2.1, 1.6, 1.4, 2.2, 1.5, 1.7, 1.7, 1.9, …
## $ Ash            <dbl> 2.4, 2.1, 2.7, 2.5, 2.9, 2.5, 2.5, 2.6, 2.2, 2.3, 2.3, 2.3, 2.4, 2.4, 2.4, …
## $ AlcAsh         <dbl> 16, 11, 19, 17, 21, 15, 15, 18, 14, 16, 18, 17, 16, 11, 12, 17, 20, 20, 16,…
## $ Mg             <int> 127, 100, 101, 113, 118, 112, 96, 121, 97, 98, 105, 95, 89, 91, 102, 112, 1…
## $ Phenols        <dbl> 2.8, 2.6, 2.8, 3.9, 2.8, 3.3, 2.5, 2.6, 2.8, 3.0, 3.0, 2.2, 2.6, 3.1, 3.3, …
## $ Flav           <dbl> 3.1, 2.8, 3.2, 3.5, 2.7, 3.4, 2.5, 2.5, 3.0, 3.1, 3.3, 2.4, 2.8, 3.7, 3.6, …
## $ NonFlavPhenols <dbl> 0.28, 0.26, 0.30, 0.24, 0.39, 0.34, 0.30, 0.31, 0.29, 0.22, 0.22, 0.26, 0.2…
## $ Proa           <dbl> 2.3, 1.3, 2.8, 2.2, 1.8, 2.0, 2.0, 1.2, 2.0, 1.9, 2.4, 1.6, 1.8, 2.8, 3.0, …
## $ Color          <dbl> 5.6, 4.4, 5.7, 7.8, 4.3, 6.8, 5.2, 5.0, 5.2, 7.2, 5.8, 5.0, 5.6, 5.4, 7.5, …
## $ Hue            <dbl> 1.04, 1.05, 1.03, 0.86, 1.04, 1.05, 1.02, 1.06, 1.08, 1.01, 1.25, 1.17, 1.1…
## $ OD             <dbl> 3.9, 3.4, 3.2, 3.5, 2.9, 2.9, 3.6, 3.6, 2.9, 3.5, 3.2, 2.8, 2.9, 2.7, 3.0, …
## $ Proline        <int> 1065, 1050, 1185, 1480, 735, 1450, 1290, 1295, 1045, 1045, 1510, 1280, 1320…
```

```
library(ggbiplot)
wine.pca = prcomp(wine,
                  scale. = TRUE)
p = ggbiplot(wine.pca,
             obs.scale = 1,
             var.scale = 1,
             groups = wine.class,
             ellipse = TRUE,
             circle = TRUE) +
  scale_color_discrete(name = '') +
  theme_bw(base_size = 30) +
  theme(legend.direction = 'horizont
        legend.position = 'top')
p
```



Alternatives

- Factor analysis
- t-distributed stochastic neighbor embedding

References

James, G., D. Witten, T. Hastie, and R. Tibshirani (2017). *An Introduction to Statistical Learning: With Applications in R*. New York: Springer. URL: https://www-bcf.usc.edu/~gareth/ISL/.

Yuan, X., D. Ren, Z. Wang, and C. Guo (2013). "Dimension Projection Matrix/Tree: Interactive Subspace Visual Exploration and Analysis of High Dimensional Data". In: *IEEE Transactions on Visualization and Computer Graphics* 19, pp. 2625-2633.