

STT3030 - Cours #11

Arthur Charpentier

Automne 2024

Apprentissage non-supervisé

Deux problèmes standards en apprentissage non-supervisé:

- ▶ La mise-en-grappe/le regroupement : Méthodes et techniques afin de créer des groupes de variables *similaires*
- ▶ La réduction de dimension: Méthodes et techniques pour projeter les données $\mathbf{X} \in \mathcal{X}$ sur $\mathbf{Z} \in \mathcal{Z}$ où la dimension de \mathcal{Z} , un *score* ou un *code*, est **beaucoup** plus petite que celle de \mathcal{X} .

Réduction de dimension

- ▶ Beaucoup de modèles supervisés sont moins performants et lents à optimiser lorsque p est trop grand. (exemple: modèles linéaires si $p > n$)
- ▶ Il peut parfois être difficile de visualiser des données $\mathbf{X} \in \mathcal{X}$ lorsqu'elles sont de très grande dimension.
- ▶ Dans la réduction de dimension on vise à apprendre une fonction $f: \mathcal{X} \subset \mathbb{R}^p \rightarrow \mathcal{Z} \subset \mathbb{R}^d$ où $d \ll p$.
- ▶ On veut souvent conserver le maximum d'information pour distinguer les observations malgré la réduction de dimension.
- ▶ On doit aussi penser à la forme de f , il est pratique d'apprendre $g: \mathcal{Z} \rightarrow \mathcal{X}$ pour reconstruire \mathbf{x} étant donné \mathbf{z} .

Pas de mesure de succès

- ▶ Un grand défi de l'apprentissage non-supervisé est le manque d'une bonne mesure de succès.
- ▶ Comme il n'y a pas de réponse, nous ne pouvons pas utiliser l'erreur de prédiction/classification.
- ▶ Il n'y a pas de vrais groupes ni de vraie représentation de petite dimension non plus.
- ▶ Un problème est donc qu'il est difficile d'établir si nous avons raisonnablement accompli notre tâche d'apprentissage non-supervisé.
- ▶ Le succès du modèle est plus subjectif.
- ▶ L'évaluation du succès d'une technique d'apprentissage non-supervisé est un problème encore d'actualité.

Analyse en composante principale

- ▶ L'Analyse en Composante Principale (ACP ou PCA en anglais) est la technique de réduction de dimension la plus utilisée.
- ▶ C'est une compression assez simple, mais très efficace, avec de belles propriétés théoriques.
- ▶ Utilisé à ce jour dans tout problème, il serait surprenant qu'en tant que scientifique des données vous ne vous en serviez pas au moins une fois.
- ▶ Je PCA tout le temps!

Analyse en composante principale: intuition

Peut-on résumer l'information en moins de variables ?

	100m	Long.jump	Shot.put	High.jump	400m	110m.hurdle	Discus	Pole.vault	Javeline	1500m	Rank
Sebrle	10.85	7.84	16.36	2.12	48.36	14.05	48.72	5.0	70.52	280.01	1
clay	10.44	7.96	15.23	2.06	49.19	14.13	50.11	4.9	69.71	282.00	2
Karpov	10.50	7.81	15.93	2.09	46.81	13.97	51.65	4.6	55.54	278.11	3
Macey	10.89	7.47	15.73	2.15	48.97	14.56	48.34	4.4	58.46	265.42	4
Warners	10.62	7.74	14.48	1.97	47.97	14.01	43.73	4.9	55.39	278.05	5
Zsivoczky	10.91	7.14	15.31	2.12	49.40	14.95	45.62	4.7	63.45	269.54	6
Hernu	10.97	7.19	14.65	2.03	48.73	14.25	44.72	4.8	57.76	264.35	7
Nool	10.80	7.53	14.26	1.88	48.81	14.80	42.05	5.4	61.33	276.33	8
Bernard	10.69	7.48	14.80	2.12	49.13	14.17	44.75	4.4	55.27	276.31	9
Schwarzl	10.98	7.49	14.01	1.94	49.76	14.25	42.43	5.1	56.32	273.56	10

Figure 1: Extrait de Bishop and Nasrabadi (2006).

Analyse en composante principale: intuition

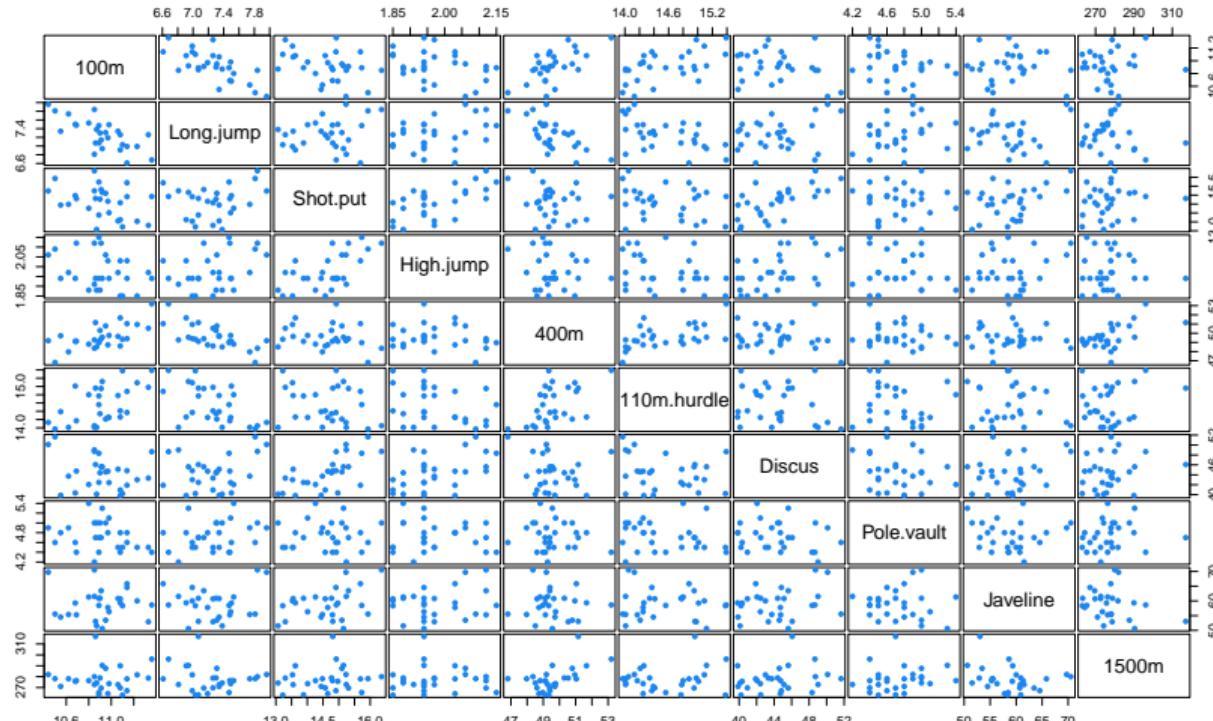
Si les variables sont corrélées peut-être qu'on a pas besoin de tout

	100m	Long.jump	Shot.put	High.jump	400m	110m.hurdle	Discus	Pole.vault	Javeline	1500m
100m	1.00000000	-0.75373835	-0.2170047	-0.07983862	0.52031353	0.50062239	-0.5418075	0.1364291	-0.09463474	-0.7269805
Long.jump	-0.75373835	1.00000000	0.3452939	-0.02023852	-0.45956236	-0.66200761	0.5305147	0.1753492	0.35120755	0.8508241
Shot.put	-0.21700468	0.34529394	1.0000000	0.79793846	-0.43996876	-0.18343091	0.8625095	-0.4373048	0.47118467	0.1299440
High.jump	-0.07983862	-0.02023852	0.7979385	1.00000000	-0.09570229	-0.08990021	0.6870671	-0.8262690	0.21381129	-0.1647813
400m	0.52031353	-0.45956236	-0.4399688	-0.09570229	1.00000000	0.50703040	-0.4966634	0.1346460	0.20735404	-0.3026672
110m.hurdle	0.50062239	-0.66200761	-0.1834309	-0.08990021	0.50703040	1.00000000	-0.3777307	0.1413458	0.10667411	-0.4961073
Discus	-0.54180755	0.53051475	0.8625095	0.68706712	-0.49666342	-0.37773065	1.0000000	-0.4620998	0.37637738	0.2553976
Pole.vault	0.13642909	0.17534921	-0.4373048	-0.82626899	0.13464601	0.14134579	-0.4620998	1.0000000	0.32529312	0.3372737
Javeline	-0.09463474	0.35120755	0.4711847	0.21381129	0.20735404	0.10667411	0.3763774	0.3252931	1.00000000	0.3670799
1500m	-0.72698051	0.85082406	0.1299440	-0.16478130	-0.30266722	-0.49610729	0.2553976	0.3372737	0.36707992	1.0000000

Figure 2: Extrait de Bishop and Nasrabadi (2006).

Analyse en composante principale: intuition

On peut prendre des combinaisons linéaires qui résume l'effet conjointe de plusieurs variables. Ici, longjump et 100m sont très corrélés, on pourrait les combiner!



Linear Independence

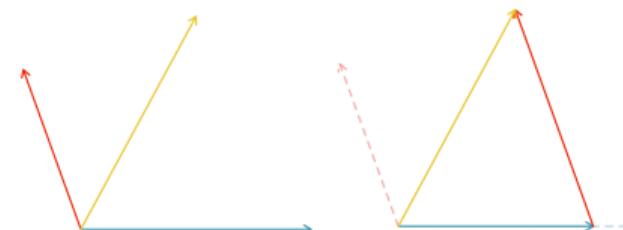
A collection of k vectors $\{\vec{x}_1, \dots, \vec{x}_k\}$ (in \mathbb{R}^n) are **linearly dependent** if there are $\alpha_1, \dots, \alpha_k$ such that

$$\alpha_1 \vec{x}_1 + \alpha_2 \vec{x}_2 + \cdots + \alpha_k \vec{x}_k = \vec{0}$$

They are **linearly independent** if they are not linearly dependent, i.e.

$$\alpha_1 \vec{x}_1 + \alpha_2 \vec{x}_2 + \cdots + \alpha_k \vec{x}_k = \vec{0} \implies \alpha_1 = \cdots = \alpha_k = 0$$

Proposition A collection of linearly independent vectors in \mathbb{R}^n can have, at most, n elements.



$(n+1)$ vectors in \mathbb{R}^n are linearly dependent

Orthonormalization (Gram-Schmidt)

If vectors $\{\vec{x}_1, \dots, \vec{x}_k\}$ are linearly independent, the Gram-Schmidt algorithm produces an **orthonormal** collection of vectors $\{\vec{u}_1, \dots, \vec{u}_k\}$ with the following properties

- ▶ every \vec{x}_j is a linear combination of $\{\vec{u}_1, \dots, \vec{u}_j\}$
- ▶ every \vec{u}_j is a linear combination of $\{\vec{x}_1, \dots, \vec{x}_j\}$

Algorithm 1: Gram-Schmidt

- 1 initialization : $\{\vec{x}_1, \dots, \vec{x}_n\};$
 - 2 **for** $t=1,2,\dots,k$ **do**
 - 3 $\vec{u}_t = \vec{x}_t - [\langle \vec{u}_1, \vec{x}_t \rangle \vec{u}_1 + \dots + \langle \vec{u}_{t-1}, \vec{x}_t \rangle \vec{u}_{t-1}]; \vec{u}_t = \vec{u}_t / \|\vec{u}_t\|$
-



Matrices

Soient $m, n \geq 1$. Une matrice \mathbf{A} de taille (m, n) à coefficients réels est un tableau de nombres réels ayant m lignes et n colonnes. On note également par $(\mathbf{A})_{ij}$ ou plus simplement A_{ij} l'élément sur la ligne i et sur la colonne j de \mathbf{A} .

Example:

$$\mathbf{A} = \begin{pmatrix} 1.5 & 2 & 3.1 & 8 \\ -1 & 4 & 5 & 6.5 \end{pmatrix}$$

\mathbf{A} est de taille (2×4) et par exemple $A_{13} = 3.1$.

Une matrice ne contenant qu'une colonne est appelée un vecteur et une matrice ne contenant qu'une ligne est un vecteur ligne.

Par exemple $\mathbf{x} = \begin{pmatrix} 1.5 \\ -1 \end{pmatrix}$ et $\mathbf{y} = (1.5 \ 2 \ 3.1 \ 8)$ sont respectivement de taille $(2, 1)$ et $(1, 4)$.

Products

If \mathbf{A} and \mathbf{B} are (respectively) $k \times m$ and $m \times n$ matrices,

$$\mathbf{C}_{ij} = \mathbf{A}_{i\cdot}^\top \mathbf{B}_{\cdot j} = A_{i1}B_{1j} + \cdots + A_{im}B_{mj} = \sum_{k=1}^m A_{ik}B_{kj},$$

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ \textcolor{teal}{A}_{21} & \textcolor{teal}{A}_{22} & \cdots & \textcolor{teal}{A}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & \textcolor{teal}{B}_{12} & \cdots & B_{1p} \\ B_{21} & \textcolor{teal}{B}_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & \textcolor{teal}{B}_{m2} & \cdots & B_{mp} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1p} \\ C_{21} & \textcolor{teal}{C}_{22} & \cdots & C_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{np} \end{pmatrix}$$

```
1 > A = matrix(1:6,2,3)
2 > B = matrix(1:12,3,4)
3 > A %*% B
4     [,1] [,2] [,3] [,4]
5 [1,]    22    49    76   103
6 [2,]    28    64   100   136
```

Le produit matriciel n'est pas commutatif pour deux matrices quelconque de même taille: $\mathbf{AB} \neq \mathbf{BA}$

Products

For vectors $\mathbf{a} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^n$, one can define the **dot product**

$$\mathbf{a} \cdot \mathbf{b} = \langle \vec{\mathbf{a}}, \vec{\mathbf{b}} \rangle = \mathbf{a}^\top \mathbf{b} = \sum_{i=1}^n a_i b_i \in \mathbb{R}$$

For matrices $\mathbf{A} \in \mathbb{R}^{m,n}$ and $\mathbf{B} \in \mathbb{R}^{m,p}$, one can define the **cross product**

$$\mathbf{A} \cdot \mathbf{B} = \mathbf{A}^\top \mathbf{B} \in \mathbb{R}^{n \times p}$$

```
1 > crossprod(A, B)
```

For matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^{m \times n}$, one can define the **Hadamard product** - or **element-wise product** - as

$$\mathbf{A} \odot \mathbf{B} = \mathbf{C} \in \mathbb{R}^{m \times n}, \quad C_{ij} = A_{ij} B_{ij}$$

```
1 > A*B
```

Products

For matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^{n \times p}$, one can define the **matrix product** as

$$\mathbf{AB} = \mathbf{C} \in \mathbb{R}^{m \times p}, \quad C_{ij} = \sum_{k=1}^n A_{ik} B_{kj} = \mathbf{A}_{\cdot i}^\top \cdot \mathbf{B}_{\cdot j} = \langle \vec{\mathbf{A}}_{\cdot i}, \vec{\mathbf{B}}_{\cdot j} \rangle$$

```
1 > A %*% B
```

For matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^{p \times q}$, one can define the **Kronecker product** as

$$\mathbf{A} \otimes \mathbf{B} = \mathbf{C} \in \mathbb{R}^{mp \times nq} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B} & \cdots & \mathbf{A}_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{m1}\mathbf{B} & \cdots & \mathbf{A}_{mn}\mathbf{B} \end{bmatrix}$$

```
1 > kronecker(A, B)
```

Rotation

One can use matrices to transform vectors, e.g. $\vec{y} = \mathbf{A}\vec{x}$, with $\vec{x}, \vec{y} \in \mathbb{R}^n$, and \mathbf{A} is some $n \times n$ matrix.

Example:

$$\mathbf{A}\vec{x} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos \theta - y\sin \theta \\ x\sin \theta + y\cos \theta \end{bmatrix}.$$

If $\mathbf{A} = R_0(\theta)$, $\mathbf{A}^\top = R_0(-\theta) = \mathbf{A}^{-1}$

Example: $\mathbf{A} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} = R_0(\theta)$ and $\mathbf{B} = \begin{bmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{bmatrix} = R_0(\phi)$ then

$$\mathbf{AB} = \begin{bmatrix} \cos \theta \cos \phi - \sin \theta \sin \phi & -\cos \theta \sin \phi - \sin \theta \cos \phi \\ \sin \theta \cos \phi + \cos \theta \sin \phi & -\sin \theta \sin \phi + \cos \theta \cos \phi \end{bmatrix} \text{ i.e.}$$

$$\mathbf{AB} = \begin{bmatrix} \cos(\theta + \phi) & -\sin(\theta + \phi) \\ \sin(\theta + \phi) & \cos(\theta + \phi) \end{bmatrix} = R_0(\theta + \phi)$$

Rotation & Orthogonal Matrices

In higher dimension n , a $n \times n$ matrix \mathbf{A} is **orthogonal** if its columns and rows are orthogonal unit vectors, i.e.

$$\mathbf{A}^\top \mathbf{A} = \mathbf{A} \mathbf{A}^\top = \mathbb{I}$$

or equivalently, $\mathbf{A}^{-1} = \mathbf{A}^\top$.

In dimension 2,

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} = \text{rotation, and } \begin{bmatrix} \cos \theta & \sin \theta \\ \sin \theta & -\cos \theta \end{bmatrix} = \text{reflection}$$

Linear vs. affine

Note that

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

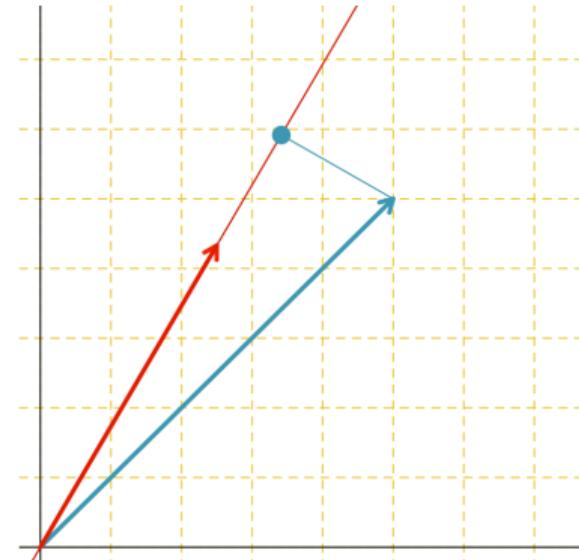
is linear but not affine (there is no constant here). Trick

$$\begin{bmatrix} 1 & 0 & 0 \\ a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ a + bx + cy \\ d + ex + fy \end{bmatrix}$$

Projection

Consider the projection (in \mathbb{R}^2) on $\{\vec{x}_1\}$. Let $\mathbf{X} = [x_1]$,
 $\mathbf{P} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$ is the (orthogonal) projection on $\{\vec{x}_1\}$

```
1 > theta = pi/3
2 > u=c(cos(theta),sin(theta))
3 > X = matrix(u,2,1)
4 > P = X %*% solve(t(X)%*%X) %*% t(X)
5 > P
6
7      [,1]      [,2]
8 [1,] 0.2500000 0.4330127
9 [2,] 0.4330127 0.7500000
10 > P %*% c(1,1)
11
12      [,1]
13 [1,] 0.6830127
14 [2,] 1.1830127
```

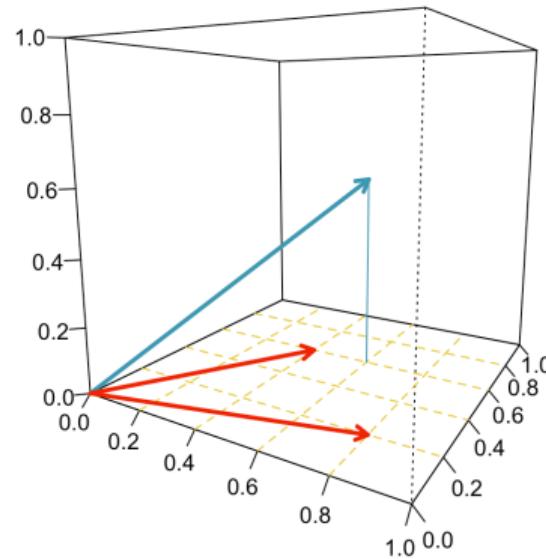


Note: projection on $\{\vec{x}_1\}$ is the projection on the straight line that goes through $\mathbf{0}$, with direction \vec{x}_1 .

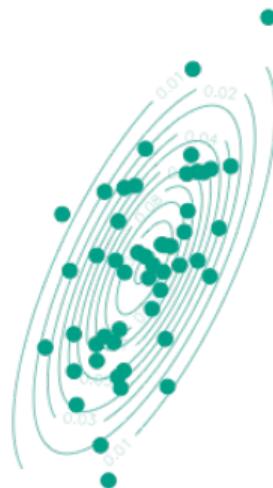
Projection

Consider the projection (in \mathbb{R}^3) on $\{\vec{x}_1, \vec{x}_2\}$. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2]$.
 $\mathbf{P} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ is the (orthogonal) projection on $\{\vec{x}_1, \vec{x}_2\}$

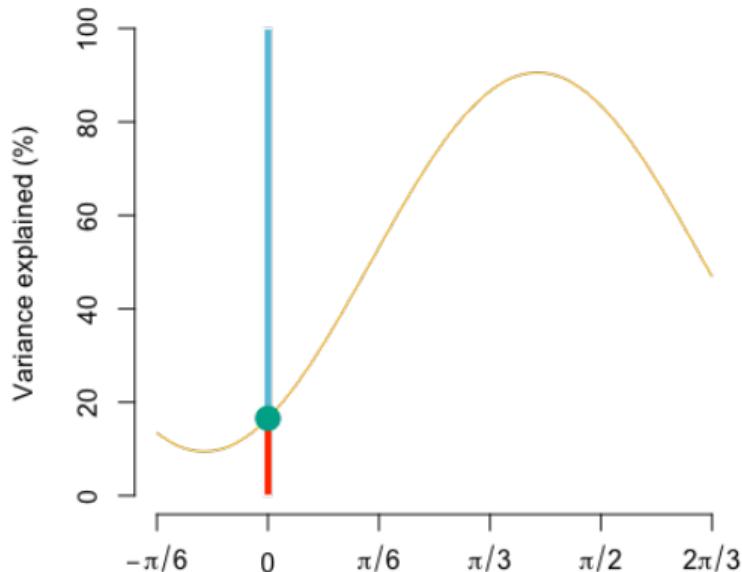
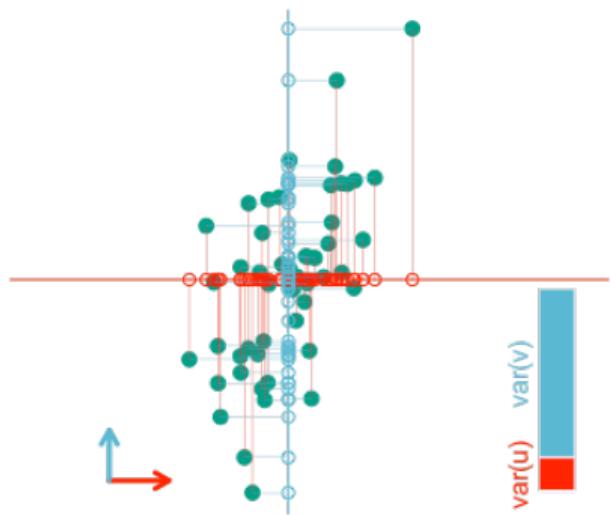
```
1 > X=cbind(c(.8,.2,0),c(.4,.6,0))
2 > P=X %*% solve(t(X)%*%X) %*% t(X)
3 > P
4      [,1] [,2] [,3]
5 [1,]    1    0    0
6 [2,]    0    1    0
7 [3,]    0    0    0
8 > P %*% c(.6,.6,0.6)
9      [,1]
10 [1,]   0.6
11 [2,]   0.6
12 [3,]   0.0
```



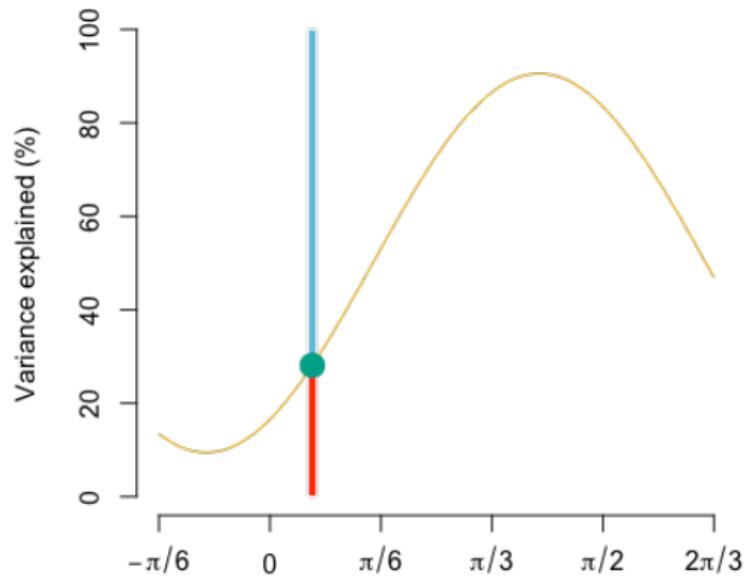
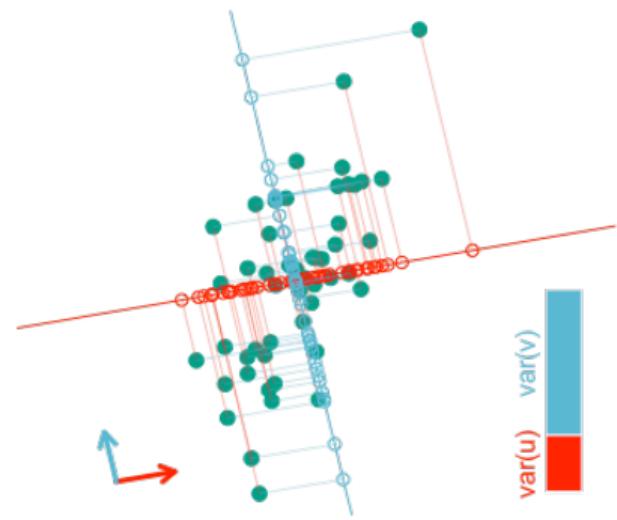
Projection et inertie (ou décomposition de la variance)



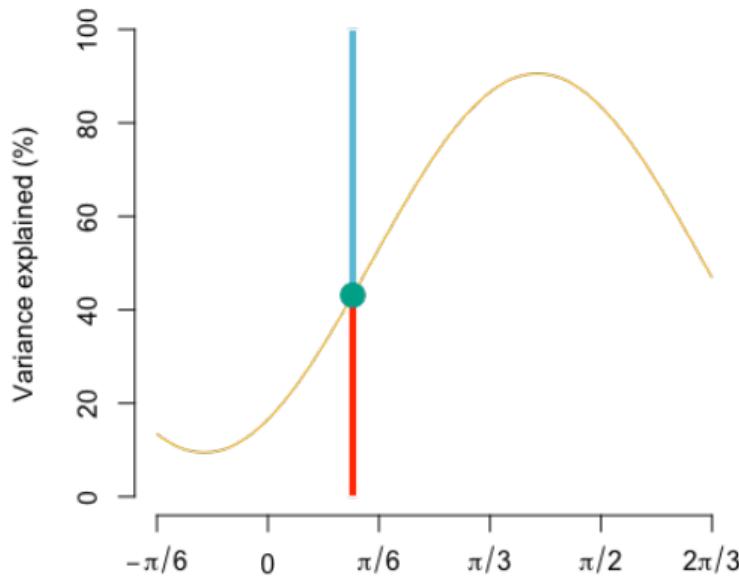
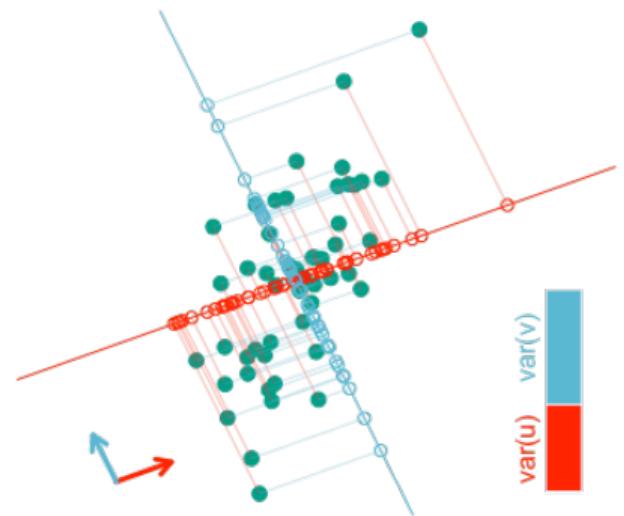
Projection et inertie (ou décomposition de la variance)



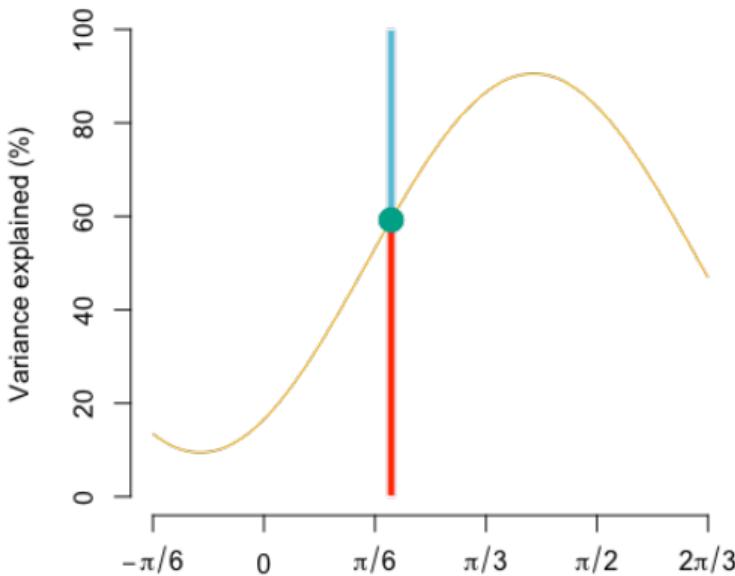
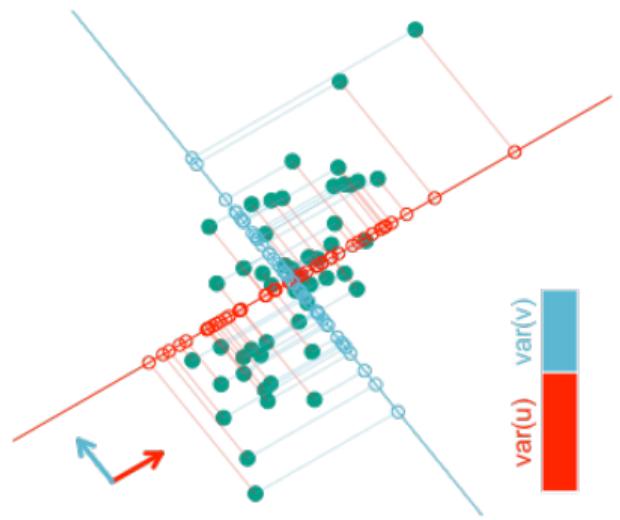
Projection et inertie (ou décomposition de la variance)



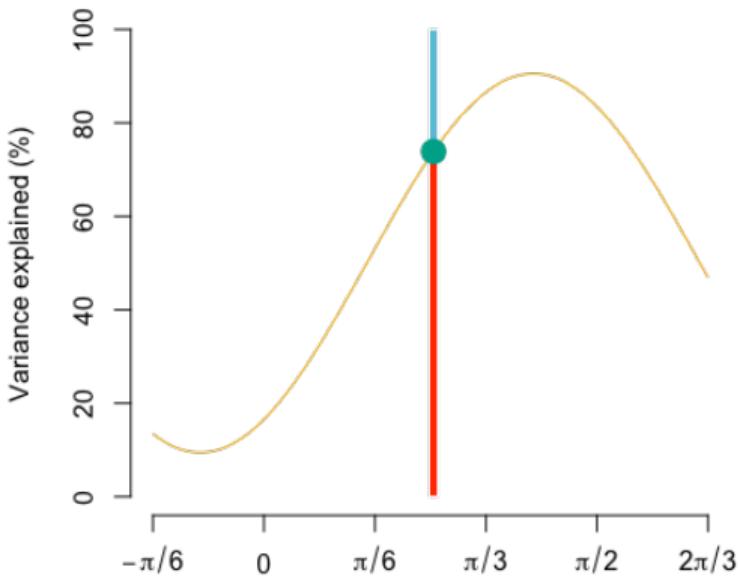
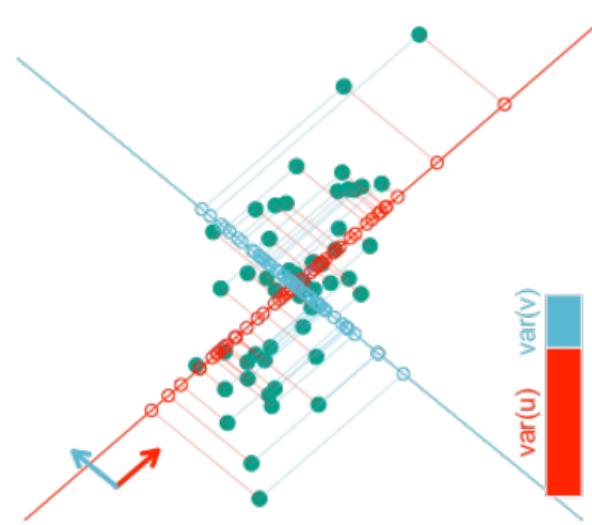
Projection et inertie (ou décomposition de la variance)



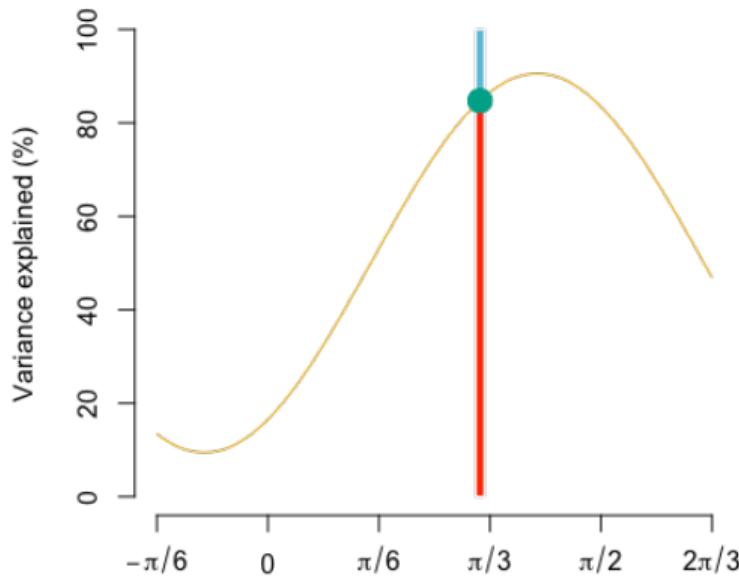
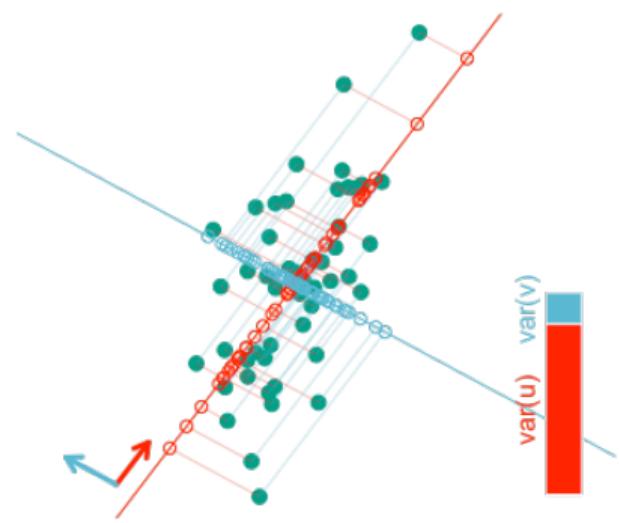
Projection et inertie (ou décomposition de la variance)



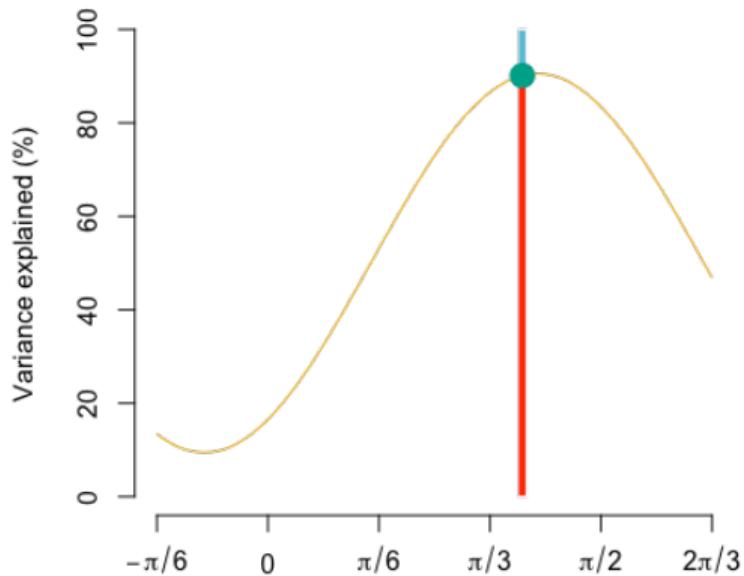
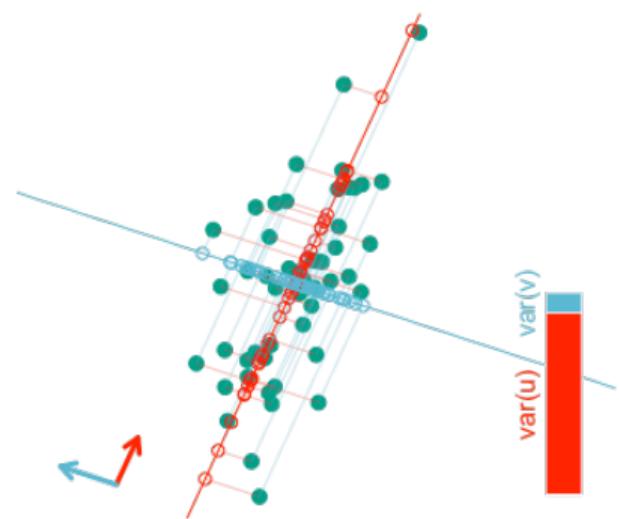
Projection et inertie (ou décomposition de la variance)



Projection et inertie (ou décomposition de la variance)



Projection et inertie (ou décomposition de la variance)



Rank

The **rank** of a matrix $r \times c$ is defined as

- ▶ the maximum number of linearly independent column vectors in the matrix
 - ▶ the maximum number of linearly independent row vectors in the matrix
- (the two are equivalent).

Note: $\text{rank}(\mathbf{M}) \leq \min\{r, c\}$.

The rank of matrix \mathbf{M} is the dimension of the vector space generated (spanned) by its columns. It equals to the number of non-zero singular values in SVD (or eigenvalues for squared matrices).

Example: $\text{rank}(\mathbf{M}^\top \mathbf{M}) = \text{rank}(\mathbf{M}\mathbf{M}^\top) = \text{rank}(\mathbf{M}^\top) = \text{rank}(\mathbf{M})$

Positive Matrices

- ▶ A real symmetric $n \times n$ \mathbf{M} is positive semidefinite (denoted $\mathbf{M} \geq 0$) if $\mathbf{z}^\top \mathbf{M} \mathbf{z} \geq 0$ for all $\mathbf{z} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$
- ▶ \mathbf{M} is positive definite (denoted $\mathbf{M} > 0$) if $\mathbf{z}^\top \mathbf{M} \mathbf{z} > 0$ for all $\mathbf{z} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$

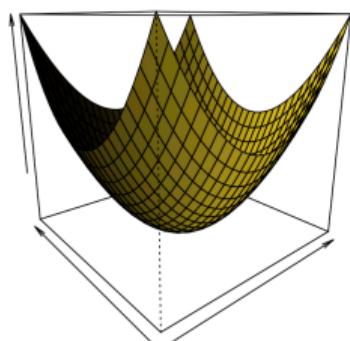
Note: \mathbf{M} is positive definite if all its eigenvalues λ_i are > 0 .

$$\mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

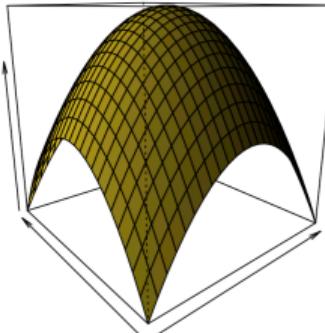
$$\mathbf{M} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

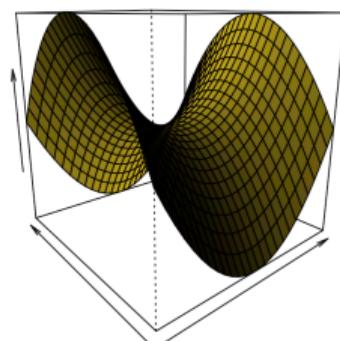
$$\mathbf{M} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$



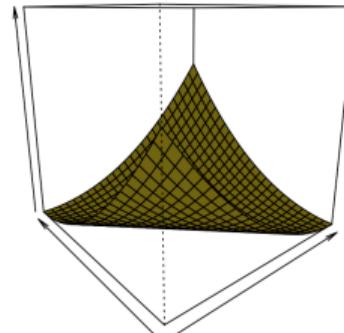
positive
definite



negative
definite



indefinite



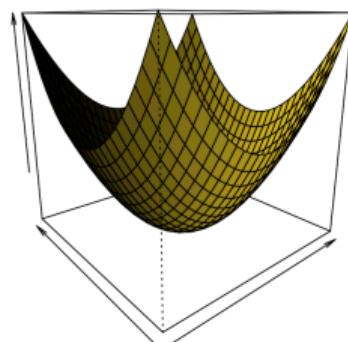
positive
semi-definite

Quadratic Forms

On peut tracer la surface $S(\mathbf{z}) = \mathbf{z}^\top \mathbf{M} \mathbf{z}$, i.e.

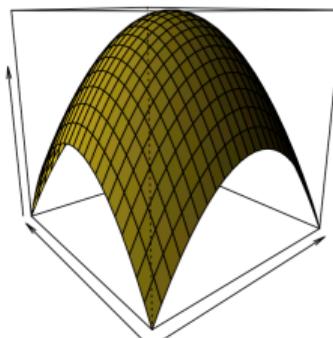
$$S : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x & y \end{pmatrix} \mathbf{M} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



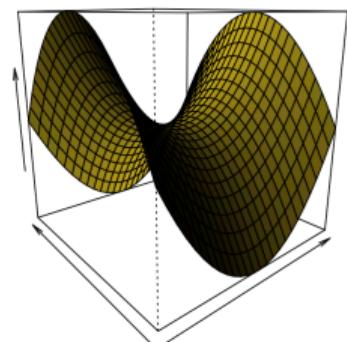
positive
definite

$$\mathbf{M} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$



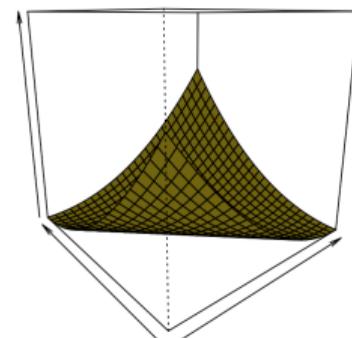
negative
definite

$$\mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$



indefinite

$$\mathbf{M} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$



positive
semi-definite

Eigenvalues & Eigenvectors for Squared Matrices

Let \mathbf{M} denote some real $n \times n$ matrix. λ is an eigenvalue, associated to eigenvector \vec{u} if one of the following holds

- ▶ $\mathbf{M}\vec{u} = \lambda \vec{u}$
- ▶ $(\mathbf{M} - \lambda \mathbb{I})$ cannot be inverted, or $\det(\mathbf{M} - \lambda \mathbb{I}) = 0$

Example Let \mathbf{M} be a real symmetric $n \times n$ matrix. Then \mathbf{M} has n real eigenvalues (not necessarily distinct).

Furthermore, there is a set of n corresponding eigenvectors $\{\vec{u}_1, \vec{u}_2, \dots, \vec{u}_n\}$, that constitute an orthonormal basis of \mathbb{R}^n , that is $\langle \vec{u}_i, \vec{u}_j \rangle = \delta_{i,j}$.

Example Let \mathbf{M} be a real symmetric $n \times n$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$, then

$$\lambda_1 = \max_{\mathbf{z}: \|\mathbf{z}\|=1} \{\mathbf{z}^\top \mathbf{M} \mathbf{z}\} \text{ and } \lambda_n = \min_{\mathbf{z}: \|\mathbf{z}\|=1} \{\mathbf{z}^\top \mathbf{M} \mathbf{z}\}$$

and the optimum is obtained when $\mathbf{z} \propto \vec{u}_1$ and $\mathbf{z} \propto \vec{u}_n$

Example

```
1 > M=c(1,2,3,4)
2 > dim(M)=c(2,2)
3 > eigen(M)
4 $values
5 [1] 5.3722813 -0.3722813
6
7 $vectors
8          [,1]      [,2]
9 [1,] -0.5657675 -0.9093767
10 [2,] -0.8245648  0.4159736
11 > L=eigen(M)$values
12 > P=eigen(M)$vector
13 > M %*% P[,1]
14          [,1]
15 [1,] -3.039462
16 [2,] -4.429794
17 > M %*% P[,2]
18          [,1]
19 [1,]  0.338544
20 [2,] -0.154859
```

```
1 > L[1] * P[,1]
2 [1] -3.039462 -4.429794
3 > L[2] * P[,2]
4 [1]  0.338544 -0.154859
5 > t(P) %*% P
6          [,1] [,2]
7 [1,]  1.0  -0.4
8 [2,] -0.4   1.0
```

hence

$$M\vec{u}_1 = \lambda_1 \vec{u}_1$$

$$M\vec{u}_2 = \lambda_2 \vec{u}_2$$

with $\|\vec{u}_1\| = \|\vec{u}_2\| = 1$, $\vec{u}_1 \neq \vec{u}_2$

Example

```
1 > u=c(cos(pi/6),sin(pi/6))
2 > X = matrix(u,2,1)
3 > M = X %*% solve(t(X)%*%X)
   %*% t(X)
4 > eigen(M)
5 $values
6 [1] 1 0
7 $vectors
8      [,1]      [,2]
9 [1,] -0.8660254  0.5000000
10 [2,] -0.5000000 -0.8660254
11 > u
12 [1] 0.8660254  0.5000000
```

```
1 > M=c(1,3,3,1)
2 > dim(M)=c(2,2)
3 > eigen(M)
4 $values
5 [1] 4 -2
6
7 $vectors
8      [,1]      [,2]
9 [1,] 0.7071068 -0.7071068
10 [2,] 0.7071068  0.7071068
11 > P = eigen(M)$vectors
12 > t(P) %*% P
13      [,1] [,2]
14 [1,]    1   0
15 [2,]    0   1
```

Spectral Decomposition

As a special case, for every $n \times n$ real symmetric matrix, the eigenvalues are real and the eigenvectors can be chosen such that they are orthogonal to each other. Thus a real symmetric matrix \mathbf{M} can be decomposed as

$$\mathbf{M} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}$$

where \mathbf{P} is an orthonormal matrix whose columns are the eigenvectors of \mathbf{M} , and \mathbf{D} is a diagonal matrix whose entries are the eigenvalues of \mathbf{M} .

Let $\mathbf{P} = [\vec{\mathbf{u}}_1, \vec{\mathbf{u}}_2, \dots, \vec{\mathbf{u}}_n]$, then $\langle \vec{\mathbf{u}}_i, \vec{\mathbf{u}}_j \rangle = \delta_{i,j}$.

$$\mathbf{M} = \sum_{i=1}^n \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$$

Example

```
1 > M=c(1,2,3,4)
2 > dim(M)=c(2,2)
3 > eigen(M)
4 $values
5 [1] 5.3722813 -0.3722813
6
7 $vectors
8      [,1]      [,2]
9 [1,] -0.5657675 -0.9093767
10 [2,] -0.8245648  0.4159736
11
12 > P = eigen(M)$vectors
13 > D = diag(eigen(M)$values)
14 > P %*% D %*% solve(P)
15      [,1] [,2]
16 [1,]    1    3
17 [2,]    2    4
18
19 > M=c(1,3,3,1)
20 > dim(M)=c(2,2)
21 > eigen(M)
22 $values
23 [1] 4 -2
24
25 $vectors
26      [,1]      [,2]
27 [1,] 0.7071068 -0.7071068
28 [2,] 0.7071068  0.7071068
29
30 > P = eigen(M)$vectors
31 > D = diag(eigen(M)$values)
32 > P %*% D %*% solve(P)
33      [,1] [,2]
34 [1,]    1    3
35 [2,]    3    1
```

Approximation

On peut utiliser tout ce qu'on a vu auparavant pour faire de l'approximation de matrice, par une matric de rang plus faible...

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 7 & 5 \\ 5 & 3 & 7 \end{bmatrix} = \underbrace{\begin{bmatrix} 0.19 & -0.32 & 0.74 \\ 0.75 & -0.59 & 0.16 \\ 0.64 & 0.74 & -0.65 \end{bmatrix}}_P \underbrace{\begin{bmatrix} 12.04 & 0 & 0 \\ 0 & 2.41 & 0 \\ 0 & 0 & 0.55 \end{bmatrix}}_D \mathbf{P}^{-1}$$
$$\underbrace{\begin{bmatrix} 0.19 & -0.32 & 0.74 \\ 0.75 & -0.59 & 0.16 \\ 0.64 & 0.74 & -0.65 \end{bmatrix}}_P \underbrace{\begin{bmatrix} 12.04 & 0 & 0 \\ 0 & 2.41 & 0 \\ 0 & 0 & \textcolor{red}{0.00} \end{bmatrix}}_{D'} \mathbf{P}^{-1} = \begin{bmatrix} 0.31 & 2.26 & 0.91 \\ 2.85 & 7.05 & 4.98 \\ 5.60 & 2.78 & 7.08 \end{bmatrix} = \mathbf{M}'$$

Analyse en composante principale

- ▶ On veut projeter la variable $\mathbf{x} \in \mathbb{R}^p$ vers un nouvel espace de plus petite dimension disons $d << p$.
- ▶ Une manière simple mathématique est donc de faire une projection linéaire (une combinaison linéaire).
- ▶ Pour $d = 1$, on prend $\mathbf{u}_{p \times 1}$, le vecteur de projection, et soit $\mathbf{x}_{p \times 1}$ alors $\mathbf{u}^\top \mathbf{x}$ est une combinaison linéaire des p dimensions de \mathbf{x} , la projetant ainsi sur un espace d'une dimension.

Analyse en composante principale: intuition

Le concept de l'analyse en composante principale est donc de former les combinaisons linéaires optimales pour résumer l'effet d'un ensemble de variables.

Soit $z_1 = \sum_{j=1}^p u_{1,j}x_j = \mathbf{u}^\top \mathbf{x}$, un résumé de dimension un, comment déterminer les coefficients u ?

Analyse en composante principale

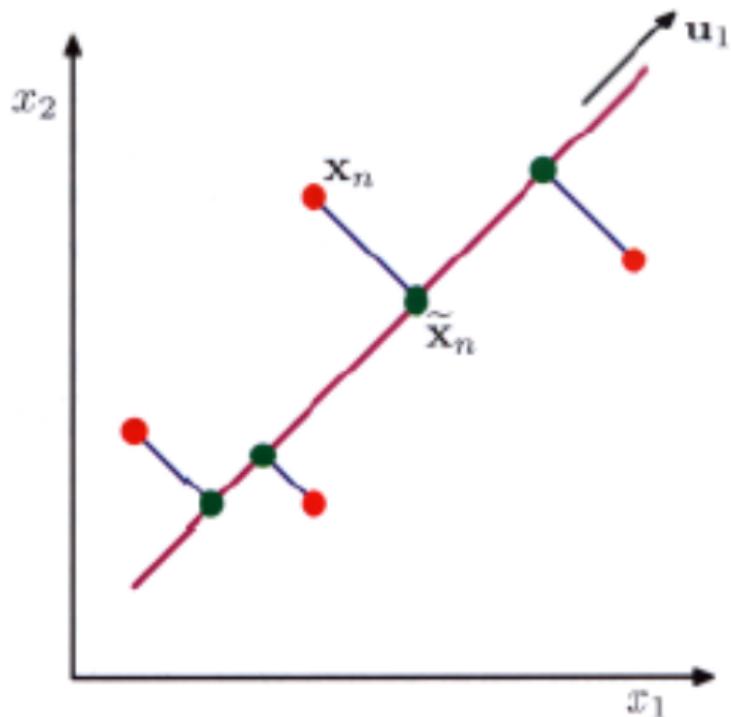


Figure 4: Extrait de Bishop and Nasrabadi (2006).

Analyse en composante principale

- ▶ Peu importe les valeurs de $\mathbf{u} = [u_1, u_2, \dots, u_p]$ nous aurons réduits la taille des données en le projetant sur un espace d'une dimension. (super!)
- ▶ Il serait important de déterminer de bonnes valeurs pour u_j .
- ▶ On veut réduire la dimension, mais on veut quand même pouvoir différencier les n observations de notre jeu de données. *Si on projette tout vers 0, bien on n'a plus aucune information!* (pas super!)

Analyse en composante principale: projection à variance maximale

- ▶ On veut préserver la diversité dans les données; on veut donc déterminer la projection qui maximise la variance dans l'espace réduit.
- ▶ On veut conserver le maximum de la diversité (variance) des observations afin de pouvoir les différencier; la variabilité c'est de l'information.

Analyse en composante principale: projection à variance maximale

- Soit $\mathbf{x}_{n \times p}$ un jeu de données de n observations de dimension p .
- Soit $\bar{\mathbf{x}}_{1 \times p} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$, la moyenne.
- Soit \mathbf{u} la projection et $\mathbf{u}^\top \bar{\mathbf{x}}$ la moyenne de la projection, la variance de la projection est donc :

$$\frac{1}{n} \sum_{i=1}^n (\mathbf{u}^\top \mathbf{x}_i - \mathbf{u}^\top \bar{\mathbf{x}})^2 = \mathbf{u}^\top \mathbf{u} \quad (1)$$

où $S = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})^\top (\mathbf{x}_i - \bar{\mathbf{x}})$ la matrice de covariance empirique. (Démonstration exercice)

Analyse en composante principale: projection à variance maximale

- ▶ On voit que si on veut maximiser la variance $\mathbf{u}^T \mathbf{u}$ de la projection $\mathbf{u}^T \mathbf{x}$, on pourrait prendre \mathbf{u} arbitrairement grand.
- ▶ Ce n'est pas exactement ce qu'on cherche, on cherche plutôt une orientation; on veut savoir quelle proportion de quelle dimension de \mathbf{x} on veut.
- ▶ Cela veut dire, que nous devons tout d'abord centré-réduire les données. Nous ne sommes pas intéressé dans les différences d'échelle, mais bien dans les corrélations.

Analyse en composante principale: projection à variance maximale

- ▶ En d'autres mots, on veut un \mathbf{u} de taille normalisée avec $\mathbf{u}^\top \mathbf{u} = 1$
- ▶ On veut une combinaison qui prend des *proportions* de chacun des prédicteurs x .
- ▶ Cela transforme donc notre problème d'optimisation de \mathbf{u}^u en un problème d'optimisation sous contrainte. (*Comme pour Lasso. En gros l'optimisation c'est important et on devrait en faire plus.*)

Analyse en composante principale: projection à variance maximale

Pour forcer une condition, une contrainte dans un problème d'optimisation, on utilise le Lagrangien.

Pour moi ce que ça veut dire c'est si on veut que $\mathbf{u}^\top \mathbf{u} = 1$, il faut ajouter $\lambda(\mathbf{u}^\top \mathbf{u} - 1)$ à la fonction objective que l'on cherche à optimiser, ici on veut maximiser:

$$\mathbf{u}^\top + \lambda(\mathbf{u}^\top \mathbf{u} - 1) \quad (2)$$

Analyse en composante principale: projection à variance maximale

Il y existe une solution analytique à ce problème, on dérive par rapport à \mathbf{u} :

$$2(S\mathbf{u}) - 2\lambda\mathbf{u} \quad (3)$$

et on met égale à 0 pour trouver:

$$2(S\mathbf{u}) - 2\lambda\mathbf{u} = 0 \quad (4)$$

$$S\mathbf{u} = \lambda\mathbf{u} \quad (5)$$

Vecteurs propres et valeurs propres

On dit que \mathbf{u} est un vecteur propre pour une matrice M si et seulement s'il existe une valeur constante $\lambda \in \mathbb{R}$ tel que $M\mathbf{u} = \lambda\mathbf{u}$.

Donc, faire le produit matriciel entre M et un vecteur propre \mathbf{u} revient à simplement multiplier le vecteur propre \mathbf{u} par une constante, λ appelé valeur propre.

- ▶ Les acquis de la classe à ce sujet sont hétérogènes et ce n'est pas mon expertise, mais merci à l'algèbre linéaire.
- ▶ Les vecteurs et valeurs propres sont des thèmes essentiels en algèbre linéaire, ceux-ci peuvent être utiles dans des tonnes d'application, l'ACP en est une.

Vecteurs propres et valeurs propres

Si nous avons $S\mathbf{u} = \lambda\mathbf{u}$, cela veut dire que la projection que nous cherchons \mathbf{u} est un vecteur de propre de S , la matrice de variance-covariance empirique.

Comment trouver les vecteurs propres d'une matrice ? Encore une fois, merci à l'algèbre linéaire.

Vecteurs propres et valeurs propres

Si une matrice M est symétrique définie positive et carré, disons $p \times p$, alors la décomposition suivante peut-être calculée aisément (diagonalisation de matrice):

$$M = V L V^\top$$

où V est une matrice une matrice orthonormal tel que sa j ème colonne, \mathbf{u}_j , est un vecteur propre et où L est une matrice diagonale $L = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_p)$ tel que λ_j est la valeur propre associée à \mathbf{u}_j .

Vecteurs propres et valeurs propres

- ▶ Mais est-ce que S est symétrique définie positive ?
- ▶ Oui, les variances/covariances sont positives et la symétrie est évidente puisque $\rho_{j,k} = \rho_{k,j}$.
- ▶ Donc merci à l'algèbre linéaire, nous pourrons déterminer facile les vecteurs et les valeurs propres de S .
- ▶ Ces diapositives font un peu honte à toute la puissance des vecteurs et valeurs propres.

Analyse en composante principale: projection à variance maximale

- ▶ Le point maximal de notre problème d'optimisation est $S\mathbf{u} = \lambda\mathbf{u}$.
- ▶ Donc, afin d'obtenir une projection, \mathbf{u} , normalisée, il nous faut utiliser un vecteur propre, mais lequel ?
- ▶ En choisissant \mathbf{u} tel que $S\mathbf{u} = \lambda\mathbf{u}$, alors la variance d'une tel projection est $\mathbf{u}^T\mathbf{u} = \mathbf{u}^T\lambda\mathbf{u} = \lambda\mathbf{u}^T\mathbf{u} = \lambda$.
- ▶ Afin de maximiser la variance, il nous faut prendre le vecteur propre \mathbf{u} associé à la plus grande valeur propre λ ; \mathbf{u}_1 et λ_1 .
- ▶ La variance associé à la première composante principale est donc la plus grande valeur propre.

Analyse en composante principale: projection à variance maximale

- ▶ Le vecteur propre \mathbf{u}_1 associé à la plus grande valeur propre λ_1 représente l'orientation dans \mathbb{R} avec la plus grande variance dans les données $\mathbf{X}_{n \times p}$.
- ▶ On appelle \mathbf{u}_1 la première composante principale, λ_1 est la variance de la projection $\mathbf{u}_1^\top \mathbf{x}$.
- ▶ En utilisant donc la colonne de V de la décomposition $S = VLV^\top$ associé au plus grand λ , on en peut projeter nos données de dimension p vers un espace de dimension 1: $\mathbf{X}_{n \times p} \mathbf{u}_{p \times 1}$.
- ▶ En réduisant à une dimension, on réduit peut-être trop la dimension des données et on se prive peut-être de beaucoup d'information.

Analyse en composante principale: nombre de composantes

- ▶ Supposons que l'on veut maintenant un deuxième vecteur de projection \mathbf{u}_2 , on veut qu'il maximise la variance restante, que le vecteur soit de norme 1 et soit orthogonal à \mathbf{u}_1 .
- ▶ La diagonalisation de la matrice retourne une matrice V tel que les colonnes sont orthonormales!
- ▶ La réponse sera de prendre encore un vecteur propre, mais cette fois-ci, celui associé à la deuxième plus grande valeur propre λ_2 .
- ▶ Le résultat, $U_{p \times 2} = [\mathbf{u}_1, \mathbf{u}_2]$ une matrice de projection vers un espace de deux dimensions.

Analyse en composante principale: nombre de composantes

- ▶ Pour chaque composante \mathbf{u}_j on capture une variance λ_j : $\mathbf{u}_j^T \mathbf{u}_j = \lambda_j$.
- ▶ Conséquemment, on pourrait démontrer que la variance totale parmi les données est de $\sum_{j=1}^p \lambda_j$.
- ▶ Donc si nous conservons les d vecteurs propres associé au d plus grandes valeurs propres, nous conservons $\frac{\sum_{j=1}^d \lambda_j}{\sum_{j=1}^p \lambda_j}$ comme proportion de variances des données.
- ▶ Il est commun de choisir la taille de la projection, la dimension $d << p$ nécessaire afin de conserver α proportion de la variance, e.g. 90%.

Analyse en composante principale: nombre de composantes

- ▶ Si on conserve toutes les p composantes ($U = V$), on conserve la dimension des données, on n'en fait en sorte qu'une rotation:

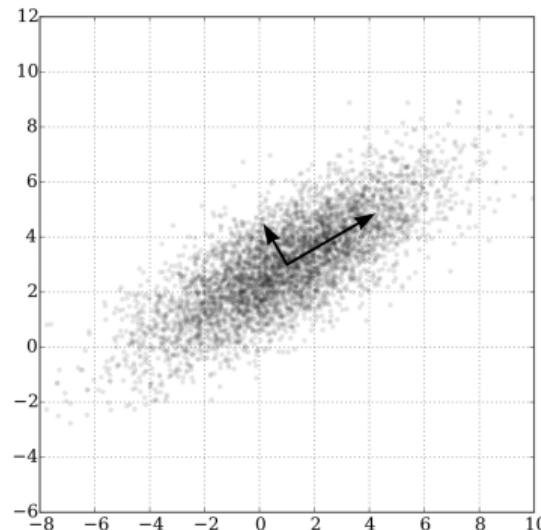


Figure 5: Trouvé sur Wiki.

Analyse en composante principale: procédure

- ▶ Préparer les données: **Centrer-réduire les données**
- ▶ Calculer la matrice de variance-covariance S .
- ▶ Déterminer d ou une proportion de variances expliquées désirées menant à d .
- ▶ Calculer la diagonalisation de la matrice $S = VLV^\top$.
- ▶ Conservé les d colonnes de V associé aux d plus grandes valeurs propres λ pour obtenir nos d composantes principales, formant collectivement la matrice de projection $U_{p \times d}$

Analyse en composante principale

- ▶ Nous venons d'apprendre que les vecteurs propres forment la projection qui maximise la variabilité.
- ▶ Il y existe d'autres critères pour une bonne projection.

Encodage et décodage

En théorie de l'information, on parle d'une fonction de compression une fonction $f: \mathcal{X} \rightarrow \mathcal{Z}$ si $\|\mathcal{Z}\| < \|\mathcal{X}\|$ puisque l'on compresse des données $\mathbf{x} \in \mathcal{X}$ vers $\mathbf{z} \in \mathcal{Z}$ de plus petite dimension.

On définit $g: \mathcal{Z} \rightarrow \mathcal{X}$ une fonction de décompression ou de reconstruction. On prend des données compressées \mathbf{z} et on reconstruit des données de pleine dimension \mathbf{x} .

Il est commun d'appeler la compression \mathbf{z} , le code. Dans ce contexte, dit que f est une fonction d'encodage et g une fonction de décodage.

Encodage et décodage

- ▶ Un critère d'évaluation de nos fonctions d'encodage et de décodage est l'erreur de reconstruction.
- ▶ Soit $\tilde{\mathbf{x}} = g(f(\mathbf{x}))$ la reconstruction de \mathbf{x} après avoir été encodé $f(\mathbf{x})$ puis décodé $g(f(\mathbf{x}))$, on désire avoir $\tilde{\mathbf{x}}$ le plus près de \mathbf{x} possible.
- ▶ Un bon système d'encodage-décodage (appelé auto-encodeur), permet la compression des données (la projection vers un espace de plus petite dimension) sans trop perdre d'information, donc avec $\tilde{\mathbf{x}} \approx \mathbf{x}$.
- ▶ Un exemple concret sont les images, les formats .jpeg ou .png sont des encodages qui permettent d'enregistrer des images en prenant moins d'espace.

Encodage et décodage: minimisation de l'erreur de reconstruction

- Soit $\tilde{\mathbf{x}} = g(f(\mathbf{x}))$ la reconstruction de \mathbf{x} , on veut apprendre une fonction f et g telle que l'on minimise:

$$\text{ReconError} = \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2 \quad (6)$$

$$= \frac{1}{n} \sum_{i=1}^n \sqrt{\sum_{j=1}^p (x_{i,j} - \tilde{x}_{i,j})^2} \quad (7)$$

On peut essayer de minimiser la fonction objective en fonction de f et g .

Analyse en composante principale: minimisation de l'erreur de reconstruction

Supposons ici que f et g sont de simples combinaisons linéaires.

- ▶ Alors f est une projection linéaire, disons $U_{p \times d}$. Conséquemment on aura $f: \mathbb{R}^p \rightarrow \mathbb{R}^d$
- ▶ et g peut aussi être représenté aussi par un produit matriciel avec $U_{d \times p}^{-1}$.
- ▶ Nous aurons que $\mathbf{z} = \mathbf{x}U$ et que $\tilde{\mathbf{x}} = (\mathbf{x}U)U^{-1}$.
- ▶ Il nous faut donc déterminer les éléments des matrices U , U^{-1} qui minimise l'erreur de reconstruction; autrement dit, les colonnes formant chacune des dimensions de notre projection.

Base et réorientation

Soit $U_{p \times p}$ qui forme une base orthonormal $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p]$, c'est-à-dire que l'on peut ré-écrire chaque observation \mathbf{x}_i comme un étant une combinaison linéaire $\mathbf{x}_i = \sum_{j=1}^p z_j \mathbf{u}_j$ où z_j est un score en quelque sorte (un réel, un coefficient) et \mathbf{u}_j un vecteur de taille $(1 \times p)$.

La collection de scores $z_i = \mathbf{x}_i^\top U$ forme une nouvelle représentation de nos données. Avec cette représentation, on peut récupérer les données originales sans aucune erreur de reconstruction: $\mathbf{x}_i = \sum_{j=1}^p z_{i,j} \mathbf{u}_j = \sum_{j=1}^p (\mathbf{x}_i^\top \mathbf{u}_j) \mathbf{u}_j$.

Essentiellement, la représentation ici $z_i = \mathbf{x}_i^\top U$ ne correspond donc qu'à un *genre* de rotation vers un nouveau système d'axe ($U = V$) (puisque on suppose que la base est orthogonale).

Analyse en composante principale: minimisation de l'erreur de reconstruction

- ▶ Le problème que nous posons est d'apprendre une base \mathbf{z} de dimension $d \ll p$ afin de faire de la réduction de dimension.
- ▶ Donc le problème revient à mettre au point des vecteurs \mathbf{u} orthonormals qui font en sorte que cette projection est d'erreur minimale.
- ▶ Soit la reconstruction suivant $\tilde{x}_i = \sum_{j=1}^d z_{i,j} \mathbf{u}_j + \sum_{j=d+1}^p b_j \mathbf{u}_j$ où on l'on apprend une projection \mathbf{z} de taille d et où on prend une combinaison linéaire fixe (b_j ne dépend pas de i) des bases restantes,
- ▶ alors comment former cette base $\mathbf{z}_i = \mathbf{x}_i^\top (U)$? Quels sont les vecteurs $U = [\mathbf{u}_1, \dots, \mathbf{u}_d]$ nous permettant de former la meilleure projection ?

Analyse en composante principale: minimisation de l'erreur de reconstruction

$$\text{ReconError} = \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2 \quad (8)$$

$$= \frac{1}{n} \sum_{i=1}^n \sqrt{\sum_{j=1}^p (x_{i,j} - \tilde{x}_{i,j})^2} \quad (9)$$

(10)

Analyse en composante principale: minimisation de l'erreur de reconstruction

En prenant la dérivé et en le mettant égal à 0 (pas évident d'ailleurs) on récupère:

- ▶ $z_{i,j} = \mathbf{x}_i^\top \mathbf{u}_j$, comme avant avec la base complète
- ▶ mais nous avons maintenant $b_j = \bar{\mathbf{x}}^\top \mathbf{u}_j$.

En gros quand on ne peut pas utiliser la i ème observation pour calculer le j ème score, on utilisera la moyenne $\bar{\mathbf{x}}$ pour le calcule du score et donc:

Analyse en composante principale: minimisation de l'erreur de reconstruction.

$$\text{ReconError} = \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2 \quad (11)$$

$$= \frac{1}{n} \sum_{i=1}^n \sqrt{\sum_{j=1}^d (x_{i,j} - z_{i,j}\mathbf{u}_j)^2 + \sum_{j=d+1}^p (x_{i,j} - b_j\mathbf{u}_j)^2} \quad (12)$$

$$= \frac{1}{n} \sum_{i=1}^n \sqrt{\sum_{j=1}^d (0)^2 + \sum_{j=d+1}^p (x_{i,j} - \bar{\mathbf{x}}\mathbf{u}_j)^2} \quad (13)$$

$$= \frac{1}{n} \sum_{i=1}^n \sqrt{\sum_{j=d+1}^p (\mathbf{x}_i^\top \mathbf{u}_j - \bar{\mathbf{x}}\mathbf{u}_j)^2} \quad (14)$$

(15)

Analyse en composante principale: minimisation de l'erreur de reconstruction

Finalement, minimiser

$$\text{ReconError} = \frac{1}{n} \sum_{i=1}^n \sqrt{\sum_{j=d+1}^p (\mathbf{x}_i^\top \mathbf{u}_j - \bar{\mathbf{x}}^\top \mathbf{u}_j)^2} \quad (16)$$

(17)

est équivalent à minimiser

$$\text{ReconError} = \frac{1}{n} \sum_{i=1}^n \sum_{j=d+1}^p (\mathbf{x}_i^\top \mathbf{u}_j - \bar{\mathbf{x}}^\top \mathbf{u}_j)^2 = \sum_{j=d+1}^p \mathbf{u}_j^\top \mathbf{u}_j \quad (18)$$

(19)

Analyse en composante principale: minimisation de l'erreur de reconstruction

- ▶ Pour minimiser la variance de l'erreur, des $p - d$ composantes restantes, il faut maximiser la variance des d première.
- ▶ On retrouve ainsi la même solution qu'avant, c'est-à-dire que la projection \mathbf{z}_i qui minimise l'erreur de reconstruction est celle formée en utilisant les d vecteurs propres associés aux d plus grandes valeurs propres.
- ▶ Les vecteurs propres forment la projection optimale au sens de la minimisation de l'erreur de reconstruction.
- ▶ Deux problèmes d'optimisation différents mènent à la même solution!

Analyse en composante principale: procédure

- ▶ Préparer les données: **Centrer-réduire les données**
- ▶ Calculer la matrice de variance-covariance S .
- ▶ Déterminer d ou une proportion de variances expliquées désirées menant à d .
- ▶ Calculer la diagonalisation de la matrice $S = VLV^\top$.
- ▶ Conservé les d colonnes de V associé aux d plus grandes valeurs propres λ pour obtenir nos d composantes principales, formant collectivement la matrice de projection $U_{p \times d}$
- ▶ Projeter les données $\mathbf{X}_{n \times p} U_{p \times d}$.

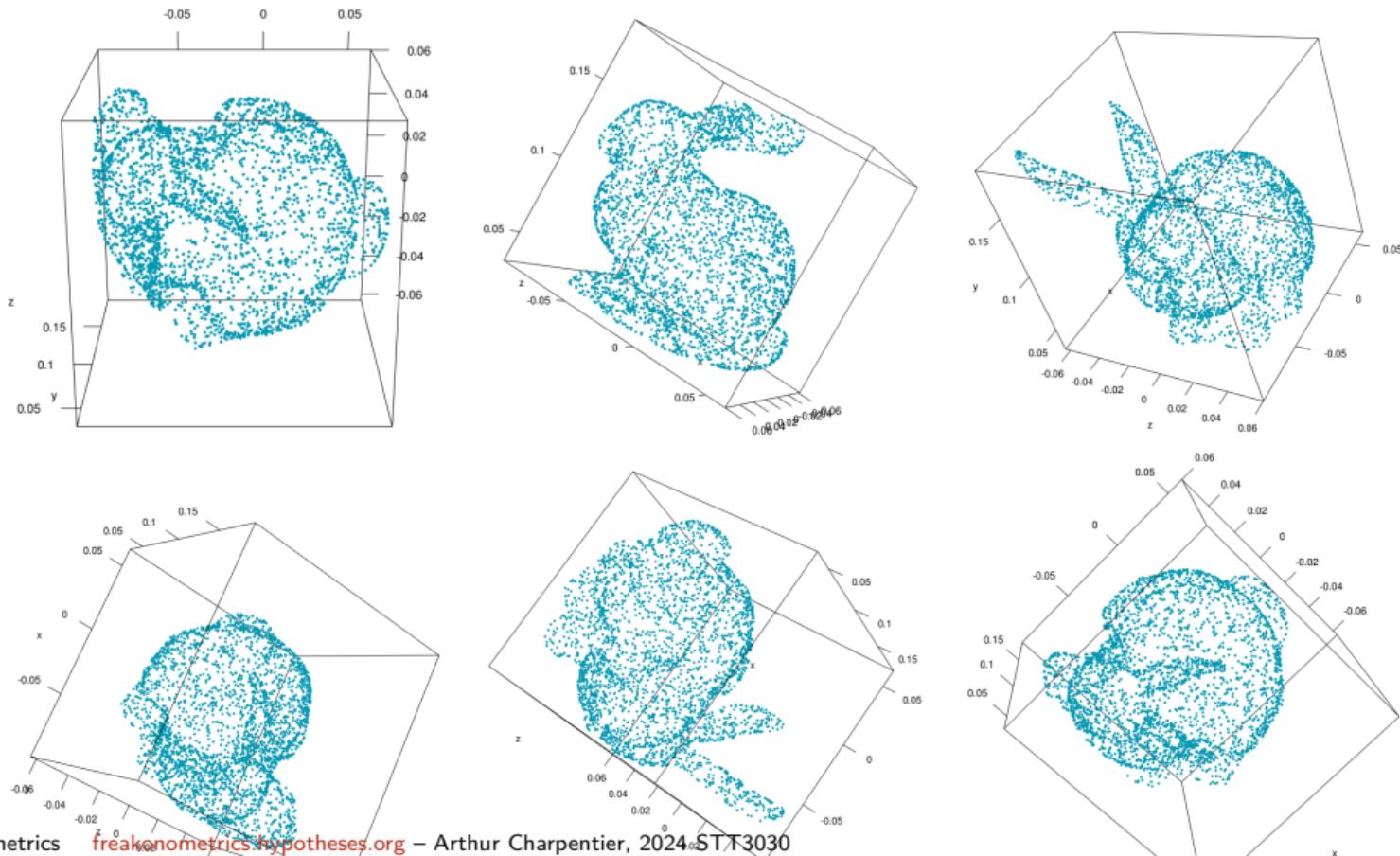
Noter que l'on peut récupérer les données dans leur dimension originale en multipliant la projection par U^t : $\tilde{\mathbf{X}}_{n \times p} = \mathbf{X}_{n \times p} U_{p \times d} U_{d \times p}^t$ et appelons ça la reconstruction de \mathbf{X}

Analyse en composante principale: application

Réduire la dimension des données avant de:

- ▶ visualiser les données
- ▶ les analyser de manière supervisée
- ▶ ou former des grappes.

ACP, projeter optimalement n points de \mathbb{R}^k dans \mathbb{R}^2



PCA with R & References

- ▶ Lebart, Piron & Morineau (2006) *Statistique exploratoire multidimensionnelle*, Dunod
- ▶ Husson, Lê & Pagès (2016) *Analyse de données avec R*, chapitre 1, PUR
- ▶ Husson, Lê & Pagès (2017) *Exploratory Multivariate Analysis by Example Using R*, chapter 1, CRC
- ▶ Jolliffe (2002) *Principal Component Analysis*, Springer

```
1 > stats::prcomp()  
2 > stats::princomp()  
3 > FactoMineR::PCA()  
4 > ade4::dudi.pca()  
5 > ExPosition::epPCA()
```

 package R  fonction R

```
1 > library(factoextra)
```

Dataset (Countries)

```
1 > url = "http://user.math.uzh.ch/furrer/download/sta121/europejobs.csv"  
2 > jobs = read.csv(url, header=TRUE)
```

cf [europejobs.csv](#)

	Agr	Min	Man	PS	Con	SI	Fin	SPS	TC
Belgium	3.3	0.9	27.6	0.9	8.2	19.1	6.2	26.6	7.2
Denmark	9.2	0.1	21.8	0.6	8.3	14.6	6.5	32.2	7.1
France	10.8	0.8	27.5	0.9	8.9	16.8	6.0	22.6	5.7
W. Germany	6.7	1.3	35.8	0.9	7.3	14.4	5.0	22.3	6.1
...

Percentages of employment in: agriculture (Agr), mining (Min), manufacturing (Man), power supply industries (PS), construction (Con), service industries (SI), finance (Fin), social and personal services (SPS), and transport and communications (TC),

Dataset (Countries)

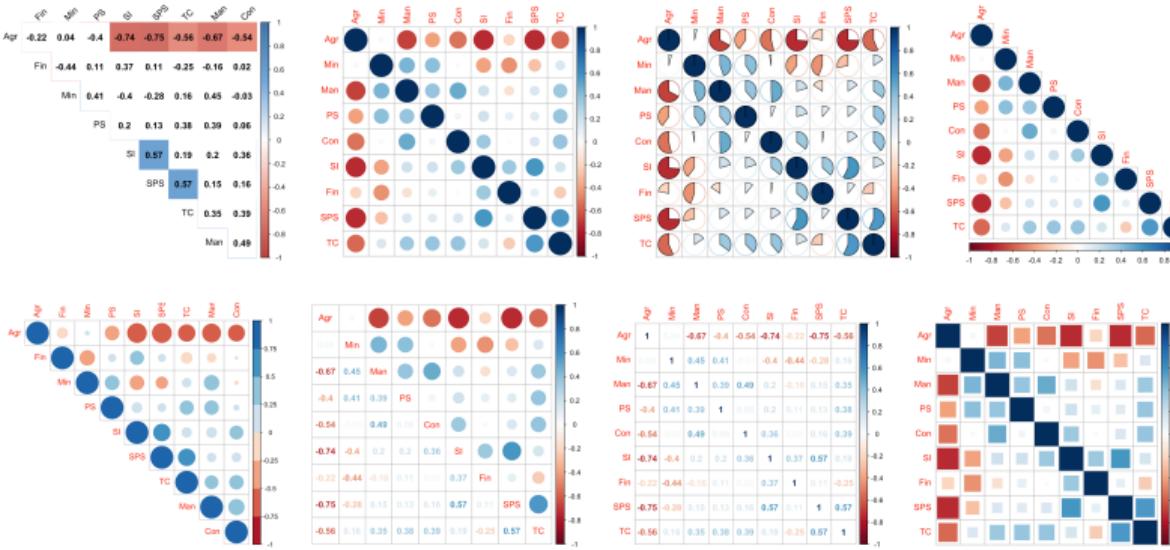
Here is the correlation matrix of \mathbf{X} ,

	Agr	Min	Man	PS	Con	SI	Fin	SPS	TC
Agr	1.00	0.04	-0.67	-0.40	-0.54	-0.74	-0.22	-0.75	-0.56
Min	0.04	1.00	0.44	0.40	-0.03	-0.40	-0.44	-0.28	0.16
Man	-0.67	0.44	1.00	0.38	0.49	0.20	-0.16	0.15	0.35
PS	-0.40	0.40	0.38	1.00	0.06	0.20	0.11	0.13	0.38
Con	-0.54	-0.03	0.49	0.06	1.00	0.36	0.02	0.16	0.39
SI	-0.74	-0.40	0.20	0.20	0.36	1.00	0.37	0.57	0.19
Fin	-0.22	-0.44	-0.16	0.11	0.02	0.37	1.00	0.11	-0.25
SPS	-0.75	-0.28	0.15	0.13	0.16	0.57	0.11	1.00	0.57
TC	-0.56	0.16	0.35	0.38	0.39	0.19	-0.25	0.57	1.00

```
1 > r = cor(jobs)
2 > library(corrplot)
```

Dataset (Countries)

```
1 > corrplot(r, method = "square")
2 > corrplot(r, method = "number")
3 > corrplot(r, type="lower")
```

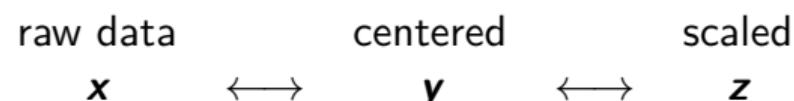


Dataset (Countries)

Consider the following sub-dataset, \mathbf{X} , with $n = 6$ observations, $k = 3$ variables,

	Agr	Min	Man
Belgium	3.3	0.9	27.6
Denmark	9.2	0.1	21.8
France	10.8	0.8	27.5
W. Germany	6.7	1.3	35.8
Ireland	23.2	1.0	20.7
Italy	15.9	0.6	27.6

- ▶ $\mathbf{x}_i = \mathbf{x}_{i\cdot} \in \mathbb{R}^k$ is row i (observation)
 - ▶ $\mathbf{x}^j = \mathbf{x}_{\cdot j} \in \mathbb{R}^n$ is column j (variable)



Dataset Transformation

Let $\bar{x}^j = \frac{1}{n} \sum_{i=1}^n x_{ij}$ and $s_j^2 = \frac{1}{n-1} \sum_{i=1}^n (x_{ij} - \bar{x}^j)^2$

Define $y_{ij} = x_{ij} - \bar{x}^j$ (centered) and $z_{ij} = \frac{x_{ij} - \bar{x}^j}{s_j}$

Given a distance d , $d(y_i, y_{i'}) = d(x_i, x_{i'}) \neq d(z_i, z_{i'})$ (scaled)

The **inertia of \mathbf{X}** , associated with d is

$$\mathcal{I}_d(\mathbf{X}) = \sum_{i=1}^n d^2(x_i, \bar{x}), \quad \bar{x} = (\bar{x}^1, \dots, \bar{x}^k)$$

If $d = d_\Delta$ for some diagonal matrix Δ , so that $d_\Delta(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\top \Delta \mathbf{b}$

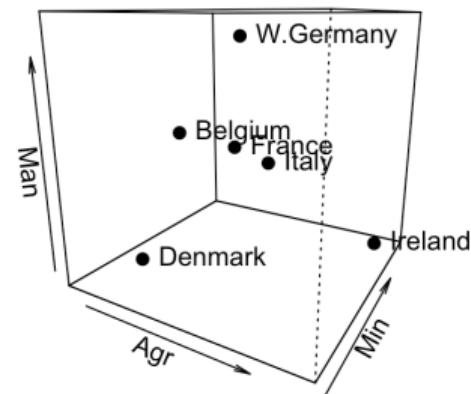
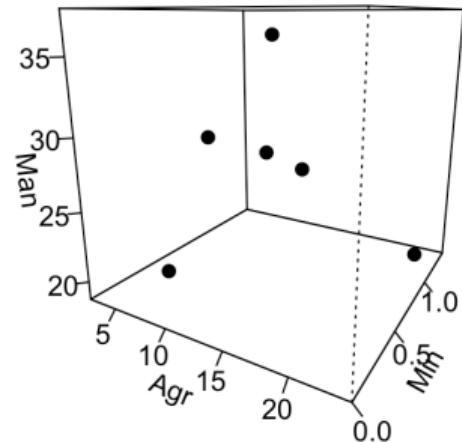
$$\mathcal{I}_\Delta(\mathbf{X}) = \sum_{i=1}^n \Delta_{j,j} \text{Var}(x^j)$$

Note that $\mathcal{I}_{\mathbb{I}}(\mathbf{Z}) = k$.

Dataset (Countries)

Raw data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_6\}$ in \mathbb{R}^3
scatterplot of individual observations
(in the space of variables)

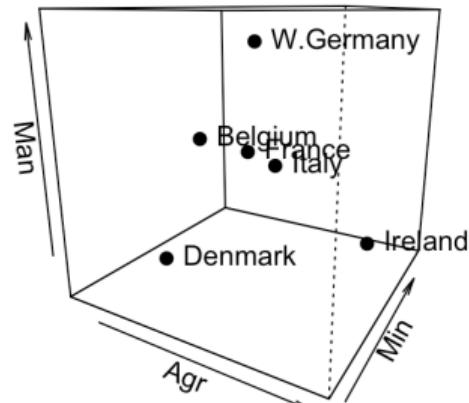
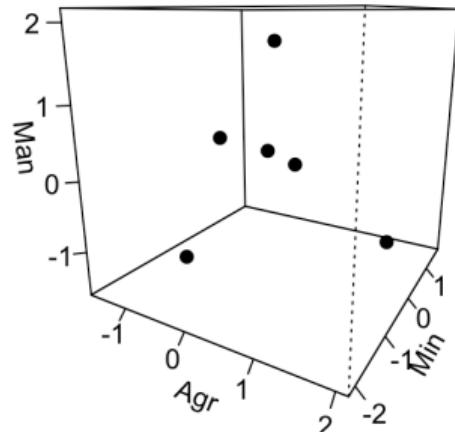
	Agr	Min	Man
Belgium	3.3	0.9	27.6
Denmark	9.2	0.1	21.8
France	10.8	0.8	27.5
W. Germany	6.7	1.3	35.8
Ireland	23.2	1.0	20.7
Italy	15.9	0.6	27.6



Dataset (Countries)

Scaled data $\mathbf{Z} = \{z_1, \dots, z_6\}$ in \mathbb{R}^3
scatterplot of individual observations
(in the space of variables)

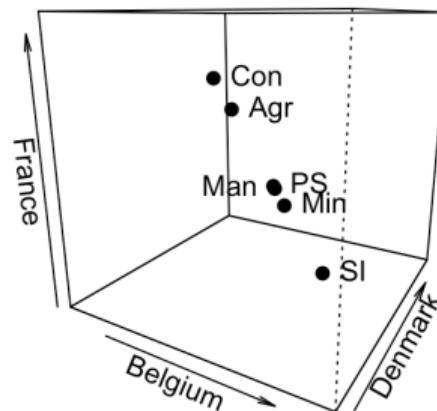
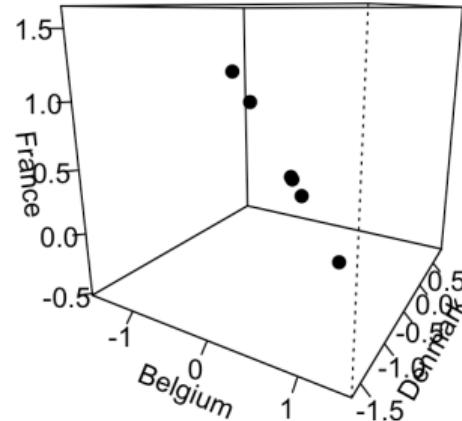
	Agr	Min	Man
Belgium	-1.16	0.29	0.14
Denmark	-0.33	-1.68	-0.93
France	-0.10	0.04	0.12
W.Germany	-0.68	1.27	1.67
Ireland	1.64	0.53	-1.14
Italy	0.62	-0.45	0.14



Dataset (Countries)

Scaled data $\mathbf{Z} = \{z^1, \dots, z^6\}$ in \mathbb{R}^3
scatterplot of scaled variables
(in the space of individuals)

	Belgium	Denmark	France
Agr	-1.13	0.36	0.77
Min	0.69	-1.15	0.46
Man	0.59	-1.15	0.56
PS	0.58	-1.15	0.58
Con	-0.70	-0.44	1.14
SI	1.01	-0.99	-0.01

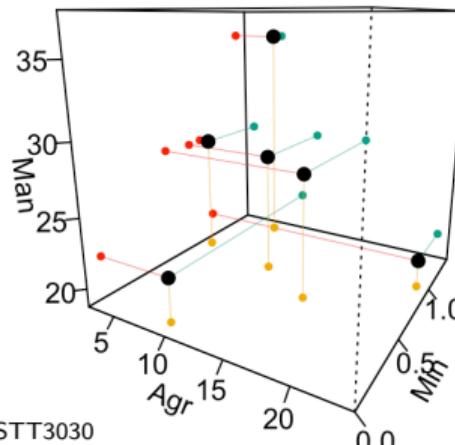
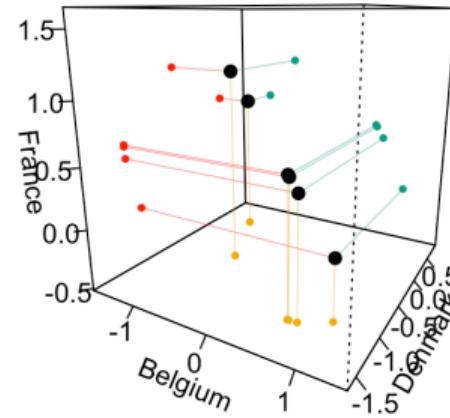


Dataset (Countries)

Scaled data $\mathbf{Z} = \{z^1, \dots, z^6\}$ in \mathbb{R}^3
scatterplot of scaled variables
(in the space of individuals)

vs.

Scaled data $\mathbf{Z} = \{z_1, \dots, z_6\}$ in \mathbb{R}^3
scatterplot of individual observations
(in the space of variables)



Which Distance ?

On peut définir, à partir d'une matrice définie positive M

- ▶ un produit scalaire : $\langle \mathbf{a}, \mathbf{b} \rangle_M = \mathbf{a}^\top M \mathbf{b}$;
- ▶ une norme: $\|\mathbf{a}\|_M^2 = \langle \mathbf{a}, \mathbf{a} \rangle_M$
- ▶ une distance: $d_M(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_M$.

d_I is the standard Euclidean distance

Let $\Sigma = \text{Var}[\mathbf{X}]$, then $d_{\Sigma^{-1}}$ is a distance.

Let $\Sigma_\Delta = \text{diag}(\text{Var}[\mathbf{X}])$, then $d_{\Sigma_\Delta^{-1}}(\mathbf{x}_i, \mathbf{x}_j) = d_I(\mathbf{z}_i, \mathbf{z}_j)$, called **scaled** distance, or **Mahalonobis** distance.

Spectral Decomposition

Soit \mathbf{M} une matrice $k \times k$. Les k couples de **valeurs propres** (λ_i) et de **vecteurs propres** (\mathbf{a}_i) forment la décomposition spectrale d'une matrice donnée.

Les valeurs propres correspondent aux k solutions possibles de l'équation :

$$\det(\mathbf{M} - \lambda \mathbb{I}_k) = 0$$

avec $k = \text{rang}(\mathbf{M})$ le nombre maximal de lignes/colonnes indépendantes dans la matrice considérée. Le vecteur propre associé à la valeur propre λ_i est donné par

$$\mathbf{M}\mathbf{a}_i = \lambda_i \mathbf{a}_i.$$

Notons que

$$\sum_{j=1}^k \lambda_j = \text{tr}(\mathbf{M}) \text{ et } \prod_{j=1}^k \lambda_j = \det(\mathbf{M}).$$

Spectral Decomposition

Example:

Spectral decomposition of

$$M = \begin{bmatrix} 7 & 4 \\ 3 & 2 \end{bmatrix}.$$

is

$$\lambda_1 = 8.772 \rightarrow \mathbf{a}_1 = \begin{bmatrix} 0.914 \\ 0.405 \end{bmatrix}$$

$$\lambda_2 = 0.228 \rightarrow \mathbf{a}_2 = \begin{bmatrix} -0.509 \\ 0.8610 \end{bmatrix}.$$

```
1 > M=c(7,3,4,2)
2 > dim(M)=c(2,2)
3 > M
4      [,1] [,2]
5 [1,]    7    4
6 [2,]    3    2
7 > eigen(M)
8 $values
9 [1] 8.7720019 0.2279981
10
11 $vectors
12      [,1]      [,2]
13 [1,] 0.9143009 -0.5085750
14 [2,] 0.4050357  0.8610177
```

Spectral Decomposition

Example:

$$M = \begin{bmatrix} 7 & 4 & 6 \\ 3 & 2 & 1 \end{bmatrix}.$$

then $M^T M$ and MM^T have the "same" eigenvalues

```
1 > M=c(7,3,4,2,6,1)
2 > dim(M)=c(2,3)
3 > M
4      [,1] [,2] [,3]
5 [1,]    7     4     6
6 [2,]    3     2     1
```

```
1 > eigen(M%*%t(M))
2 $values
3 [1] 113.332338   1.667662
4
5 $vectors
6           [,1]      [,2]
7 [1,] -0.94316  0.33232
8 [2,] -0.33232 -0.94316
9 > eigen(t(M)%*%M)
10 $values
11 [1] 113.33  1.66 -5e-15
12
13 $vectors
14           [,1]      [,2]      [,3]
15 [1,] -0.714  -0.390   0.582
16 [2,] -0.417  -0.431  -0.800
17 [3,] -0.563   0.814  -0.145
```

Spectral Decomposition

Example:

Si $M = U\Delta V^\top$ (SVD)

- ▶ $U = [\vec{a}_1, \dots, \vec{a}_k]$
- ▶ $V = [\vec{b}_1, \dots, \vec{b}_n]$
- ▶ $\delta_i = \sqrt{\lambda_i}$

```
1 > svd(M)$d
2 [1] 10.64577 1.29138
3 > svd(M)$d^2
4 [1] 113.3323 1.66766
```

```
1 > svd(M)
2 $d
3 [1] 10.64577 1.29138
4
5 $u
6 [1,] [,1] [,2]
7 [1,] -0.94316 -0.33232
8 [2,] -0.33232 0.94316
9
10 $v
11 [1,] [,1] [,2]
12 [1,] -0.71381 0.38966
13 [2,] -0.41681 0.43133
14 [3,] -0.56278 -0.81369
```

Spectral Decomposition

If \vec{a} est un vecteur propre de $\mathbf{M}^\top \mathbf{M}$, pour la valeur propre λ ,

$$\mathbf{M}^\top \mathbf{M} \vec{a} = \lambda \vec{a}$$

$$\mathbf{M} \mathbf{M}^\top \underbrace{\mathbf{M} \vec{a}}_{\vec{b}} = \lambda \underbrace{\mathbf{M} \vec{a}}_{\vec{b}}$$

$\vec{b} = \mathbf{M} \vec{a}$ est un vecteur propre de $\mathbf{M} \mathbf{M}^\top$, pour la valeur propre λ .

If \vec{b} est un vecteur propre de $\mathbf{M} \mathbf{M}^\top$, pour la valeur propre λ ,

$$\mathbf{M} \mathbf{M}^\top \vec{b} = \lambda \vec{b}$$

$$\mathbf{M}^\top \mathbf{M} \underbrace{\mathbf{M} \mathbf{M}^\top \vec{b}}_{\vec{\alpha}} = \lambda \underbrace{\mathbf{M}^\top \vec{b}}_{\vec{\alpha}}$$

$\vec{\alpha} = \mathbf{M}^\top \vec{b}$ est un vecteur propre de $\mathbf{M}^\top \mathbf{M}$ pour la valeur propre λ .

Analyse en Composantes Principales

- ▶ Une analyse en composantes principales peut se faire à partir des données centrées (\mathbf{Y}) ou des données centrée-réduites (\mathbf{Z}).
- ▶ Une analyse en composantes principales se fait en étudiant deux nuages: le nuage des n observations dans \mathbb{R}^k avec la métrique $\mathbf{M} = \mathbb{I}_k$ et le nuage des k variables dans \mathbb{R}^n avec la métrique $\mathbf{N} = (1/n)\mathbb{I}_n$.
- ▶ Dans la très grande majorité des cas, on réalise une *analyse en composantes principales normée*, c'est-à-dire à partir de la matrice des corrélations \mathbf{R} construites à partir des données centrées-réduites \mathbf{Z} .

Inertie

- ▶ L'inertie d'un nuage d'observations est une mesure de la dispersion de ce même nuage. Elle est définie par

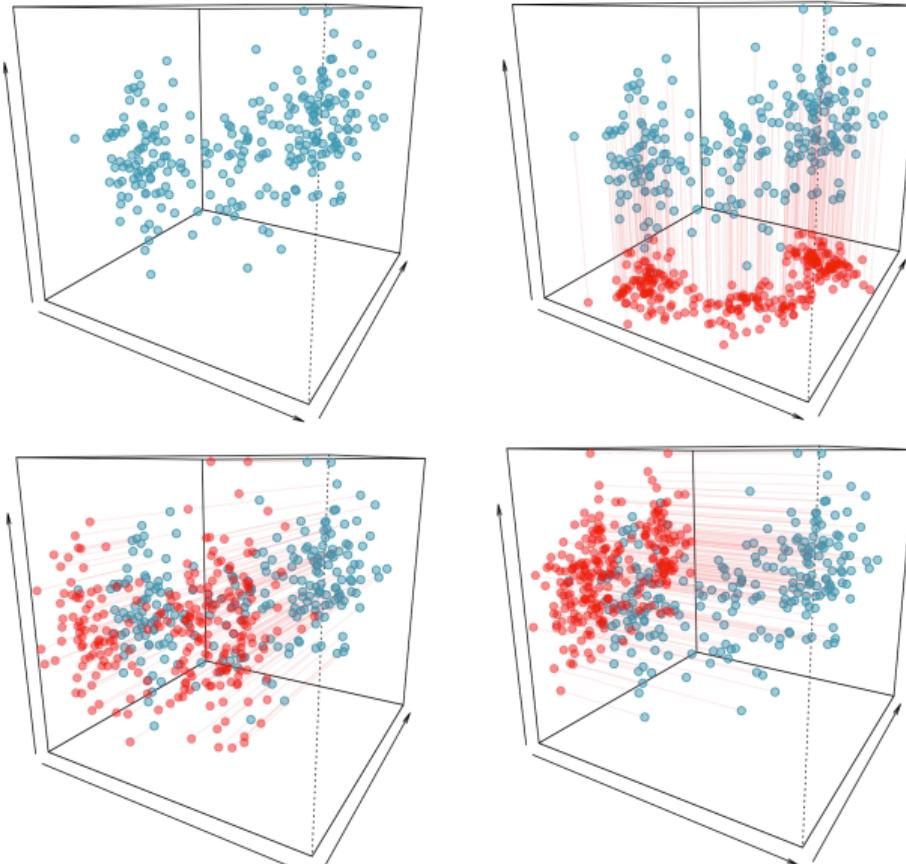
$$\mathcal{I}(\mathbf{X}) = \sum_{i=1}^n w_i d_{\mathbf{M}}^2(\mathbf{x}_i^T, \mathbf{x}), \quad \text{avec } \mathbf{x} = \begin{bmatrix} \bar{x}^1 \\ \vdots \\ \bar{x}^k \end{bmatrix}.$$

Proposition Si on choisit une métrique diagonale $\mathbf{M} = \text{diag}(m_1, \dots, m_k)$ et $w_i = 1/n$, $\forall n$, on obtient

$$\mathcal{I}(\mathbf{X}) = \sum_{i=1}^k m_j \text{Var}(\mathbf{x}^j).$$

Si on utilise $\mathbf{M} = \mathbb{I}_k$, alors $\mathcal{I}(\mathbf{Y}) = \sum_{i=1}^k \text{Var}(\mathbf{x}^j)$ et $\mathcal{I}(\mathbf{Z}) = k$.

Projection d'observations (sur un plan) ?



Projection d'une observation (sur une droite)

- ▶ Dans un premier temps, on cherche un axe défini par un vecteur $\mathbf{v}_1 \in \mathbb{R}^k$ tel que

$$\mathbf{v}_1 = \underset{\mathbf{v} \in \mathbb{R}^k; \|\mathbf{v}\|=1}{\operatorname{argmax}} \{\operatorname{Var}(\mathbf{Z}\mathbf{v})\}.$$

- ▶ Ce problème d'optimisation sous contraintes peut se réécrire sous la forme

$$\mathbf{v}_1 = \underset{\mathbf{v} \in \mathbb{R}^k; \|\mathbf{v}\|=1}{\operatorname{argmax}} \left\{ \mathbf{v}^\top \mathbf{R} \mathbf{v} \right\},$$

où $\mathbf{R} = (1/n)\mathbf{Z}^\top \mathbf{Z}$ est la **matrice de corrélation**.

- ▶ On peut démontrer que la solution à ce problème d'optimisation est le vecteur propre \mathbf{a}_1 obtenu par une décomposition spectrale de la matrice \mathbf{R} .

ACP et optimisation

On veut résoudre

$$\max_{\mathbf{v}} \left\{ \sum_{i=1}^n \langle \mathbf{z}_i, \mathbf{v} \rangle \right\} = \max_{\mathbf{v}} \left\{ \mathbf{v}^\top \mathbf{Z}^\top \mathbf{Z} \mathbf{v} \right\} \text{ subject to } \mathbf{v}^\top \mathbf{v} = 1$$

Le Lagrangien associé est

$$\mathcal{L}(\mathbf{v}, \lambda) = \mathbf{v}^\top \mathbf{Z}^\top \mathbf{Z} \mathbf{v} - \lambda(1 - \mathbf{v}^\top \mathbf{v})$$

$$\nabla_{\mathbf{v}, \lambda} \mathcal{L}(\mathbf{v}, \lambda) = 0 \Leftrightarrow \begin{cases} \mathbf{Z}^\top \mathbf{Z} \mathbf{v} = \lambda \mathbf{v} \\ \mathbf{v}^\top \mathbf{v} = 1 \end{cases}$$

De la première équation, \mathbf{v} est un vecteur propre de $\mathbf{Z}^\top \mathbf{Z}$.

$\mathbf{v} = \mathbf{a}_1$ est le vecteur propre associé à la première valeur propre.

Projection d'une observation (sur une droite)

- ▶ Dans un second temps, on cherche un axe défini par un vecteur $\mathbf{v}_2 \in \mathbb{R}^k$ tel que

$$\mathbf{v}_2 = \operatorname{argmax}_{\mathbf{v} \in \mathbb{R}^k; \|\mathbf{v}\|=1; \mathbf{v} \text{ et } \mathbf{v}_1 \text{ non corr.}} \{\operatorname{Var}(\mathbf{Z}\mathbf{v})\}.$$

- ▶ On peut démontrer que la solution à ce problème d'optimisation est le vecteur propre \mathbf{a}_2 obtenu par une décomposition spectrale de la matrice \mathbf{R} .

Pour $1 \leq q \leq r = \operatorname{rang}(\mathbf{Z})$, on construit les composantes principales $\mathbf{p}_1, \dots, \mathbf{p}_q$ qui formeront une droite ($q = 1$), un plan ($q = 2$), etc. sur lequel on projettera les observations.

Projection d'une observation (sur une droite)

- ▶ Les composantes principales ($\mathbf{p}_\alpha, \alpha = 1, \dots, q$) sont q nouvelles variables telles que
 - ▶ les corrélations entre les composantes principales sont nulles
 - ▶ la variance est $\text{Var}(\mathbf{p}_\alpha) = \lambda_\alpha$.
- ▶ Les q premières composantes principales représentent une inertie totale de

$$\mathcal{I}(\mathbf{p}_1, \dots, \mathbf{p}_q) = \lambda_1 + \dots + \lambda_q.$$

Projection d'une variable (sur une droite)

- ▶ La projection (**N**-orthogonale) d'une variable $\mathbf{z}^j \in \mathbb{R}^n$ sur un axe G_β décrit par un vecteur \mathbf{u}_β de norme 1, c'est-à-dire tel que

$$\mathbf{u}_\beta^\top \mathbf{N} \mathbf{u}_\beta = 1,$$

a pour coordonnée

$$t_{j\beta} = \langle \mathbf{z}^j, \mathbf{u}_\beta \rangle = (\mathbf{z}^j)^\top \mathbf{N} \mathbf{u}_\beta.$$

- ▶ Pour l'ensemble de l'échantillon, on notera

$$\mathbf{t}_\beta = \begin{bmatrix} t_{1\beta} \\ \vdots \\ t_{k\beta} \end{bmatrix} = \mathbf{Z}^\top \mathbf{N} \mathbf{u}_\beta.$$

Projection d'une variable

- ▶ Dans un premier temps, on cherche un axe défini par un vecteur $\mathbf{u}_1 \in \mathbb{R}^n$ tel que

$$\mathbf{u}_1 = \operatorname{argmax}_{\mathbf{u} \in \mathbb{R}^n; \|\mathbf{u}\|=1} \left\{ \|\mathbf{Z}^\top \mathbf{N} \mathbf{u}\|^2 \right\}.$$

- ▶ En posant $\mathbf{N} = (1/n)\mathbb{I}_n$, on peut démontrer que la solution à ce problème d'optimisation est le vecteur propre \mathbf{b}_1 obtenu par une décomposition spectrale de la matrice $(1/n)\mathbf{Z}^\top \mathbf{Z}$.

Sélection des vecteurs \mathbf{u}

- ▶ Dans un second temps, on cherche un axe défini par un vecteur $\mathbf{u}_2 \in \mathbb{R}^n$ tel que

$$\mathbf{u}_2 = \operatorname{argmax}_{\mathbf{u} \in \mathbb{R}^n; \|\mathbf{u}\|=1; \mathbf{u} \text{ et } \mathbf{u}_1 \text{ non corr.}} \left\{ \|\mathbf{Z}^\top \mathbf{N} \mathbf{u}\|^2 \right\}.$$

- ▶ La solution à ce problème d'optimisation est le vecteur propre \mathbf{b}_2 obtenu par une décomposition spectrale de la matrice $(1/n)\mathbf{Z}^\top \mathbf{Z}$.

Pour $1 \leq q^* \leq r^* = \operatorname{rang}(\mathbf{Z}^\top \mathbf{Z})$, on construit les composantes principales $\mathbf{t}_1, \dots, \mathbf{t}_{q^*}$ qui formeront une droite ($q^* = 1$), un plan ($q^* = 2$), etc. sur lequel on projettera les variables.

Qualité de la projection

- ▶ La qualité de la représentation est mesurée par le pourcentage de l'inertie initiale des données que les q premières composantes principales permettent d'expliquer:

$$\mathcal{I}(\mathbf{Z}) = \lambda_1 + \dots + \lambda_r = p$$

$$\mathcal{I}(\mathbf{p}_1, \dots, \mathbf{p}_q) = \lambda_1 + \dots + \lambda_q \leq \mathcal{I}(\mathbf{Z}).$$

- ▶ La i ème composante principale permet d'expliquer une proportion de

$$\frac{\lambda_i}{\lambda_1 + \dots + \lambda_r}$$

de l'information initiale.

- ▶ Le passage de r à q dimension (avec $q < r$) se fait en préservant une proportion de

$$\frac{\lambda_1 + \dots + \lambda_q}{\lambda_1 + \dots + \lambda_r}$$

de l'information initiale.

Qualité de la représentation des observations

- ▶ Pour deux observations “bien projetées”, la distance en projection est similaire à la distance dans le nuage initiale \mathbb{R}^k .
- ▶ La qualité de la projection de l’observation i sur l’axe D_α est mesurée par le carré du cosinus de l’angle $\theta_{i\alpha}$ formé entre le vecteur \mathbf{z}_i et l’axe D_α :

$$\cos^2(\theta_{i\alpha}) = \frac{\mathbf{p}_{i\alpha}^2}{\|\mathbf{z}_i\|^2}.$$

- ▶ Si l’espace final est un plan formé des axes D_1 et D_2 , la qualité de la projection de l’observation i sur ce plan est donnée par

$$\cos^2(\theta_{i(1,2)}) = \frac{\mathbf{p}_{i1}^2 + \mathbf{p}_{i2}^2}{\|\mathbf{z}_i\|^2}.$$

- ▶ Plus cette valeur est près de 1, meilleure est la projection.

Qualité de la représentation des observations

- ▶ Les observations qui contribuent de manière élevée à la construction des axes sont sources d'instabilité. La contribution d'une observation à un axe est mesurée par la proportion de l'inertie de l'axe expliquée par cette observation.
- ▶ Pour l'axe D_α , l'inertie totale est $\lambda_\alpha = \sum_{i=1}^n \mathbf{p}_{i\alpha}^2$.
- ▶ La contribution de l'observation i sur l'axe D_α est mesurée par

$$C(i, \alpha) = \frac{\mathbf{p}_{i\alpha}^2}{\lambda_\alpha}.$$

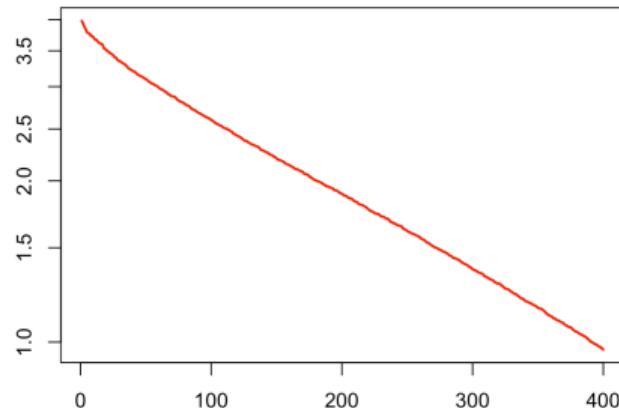
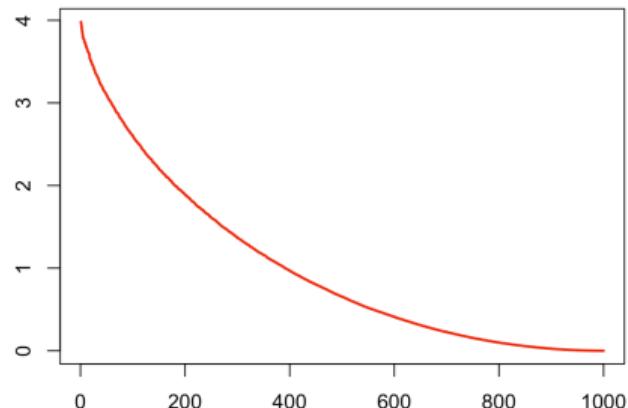
- ▶ Si l'espace final est un plan formé des axes D_1 et D_2 , la contribution de l'observation i sur ce plan est donnée par

$$C(i, (1, 2)) = \frac{\mathbf{p}_{i1}^2 + \mathbf{p}_{i2}^2}{\lambda_1 + \lambda_2}.$$

Troncation

- **Exponential decay** variance λ_j of Gaussian noise decays in an exponential manner

Example valeurs propres de $\hat{\Sigma}$, matrice de variance de $\{\mathbf{X}_1, \dots, \mathbf{X}_n\}$ où les $\mathbf{X}_i \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$ (ici $d = 1000$)



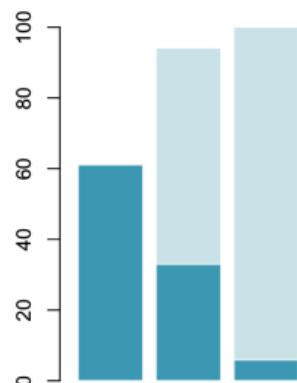
Trouver k tel que λ_j , pour $j > k$, décroît exponentiellement vite...

- **North's rule of thumb** truncate the expansion when the sampling error becomes larger than the distance between two eigenvalues, i.e., when $\lambda_j \lambda_{j+1} < \sqrt{2/n} \lambda_j$

Countries (subset, 6×3)

```
1 > X = jobs[1:6,1:3]
2 > Z=X
3 > for(j in 1:3) Z[,j]=(X[,j]
   ]-mean(X[,j]))/sd(X[,j])
4 > (R=cor(Z))
5      Agr    Min    Man
6 Agr  1.00 -0.01 -0.56
7 Min -0.01  1.00  0.61
8 Man -0.56  0.61  1.00
9 > eigen(R)
10 eigen() decomposition
11 $values
12 [1] 1.83 0.99 0.18
13
14 $vectors
15      [,1] [,2]  [,3]
16 [1,]  0.48  0.74 -0.47
17 [2,] -0.52  0.67  0.52
18 [3,] -0.70  0.00 -0.71
```

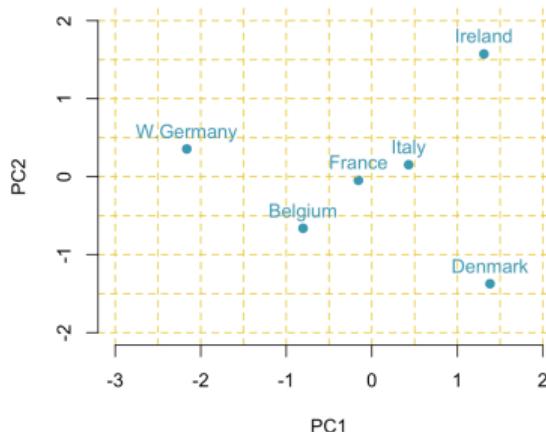
```
1 > L=eigen(R)$values
2 > barplot(cumsum(L)/sum(L))
3 > barplot(L/sum(L),add=T)
4 > L/sum(L)*100
5 [1] 61.08983 32.99279
      5.91738
```



Countries (subset, 6×3)

```
1 > as.matrix(Z) %*% eigen(R)$vectors  
2  
3 Belgium      [,1]   [,2]   [,3]  
4 Denmark      1.38  -1.37  -0.06  
5 France       -0.16  -0.05  -0.02  
6 W.Germany   -2.16   0.36  -0.20  
7 Ireland      1.31   1.57   0.31  
8 Italy        0.43   0.15  -0.63  
9 > phi=eigen(R)$vectors[,1:2]  
10 > row.names(phi) = names(Z)  
11 > colnames(phi) = c("PC1",  
12     "PC2")  
13 > PC1 = as.matrix(Z) %*% phi[,1]  
14 > PC2 = as.matrix(Z) %*% phi[,2]
```

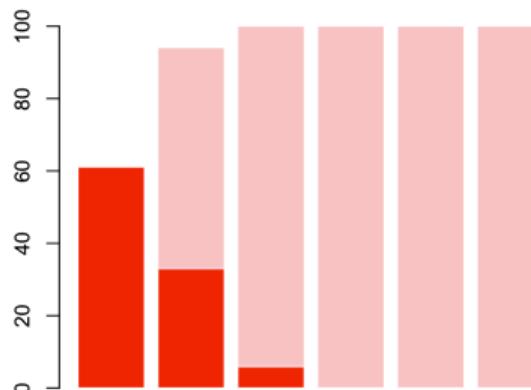
```
1 > plot(PC1,PC2)  
2 > text(PC1,PC2,rownames(Z),  
3       pos=3)
```



Countries (subset, 6×3)

```
1 > M=(as.matrix(Z)) %*% t(as.  
     matrix(Z))  
2 > eigen(M)  
3 eigen() decomposition  
4 $values  
5 [1] 9.16 4.95 0.89  
6 [4] 0.00 0.00 0.00  
7  
8 $vectors  
9      [,1]   [,2]   [,3]   [,4]  
10 [1,]  0.27 -0.30  0.63 -0.6  
11 [2,] -0.46 -0.62 -0.06  0.0  
12 [3,]  0.05 -0.02 -0.02  0.0  
13 [4,]  0.71  0.16 -0.21  0.0  
14 [5,] -0.43  0.71  0.32 -0.1  
15 [6,] -0.14  0.07 -0.67 -0.7
```

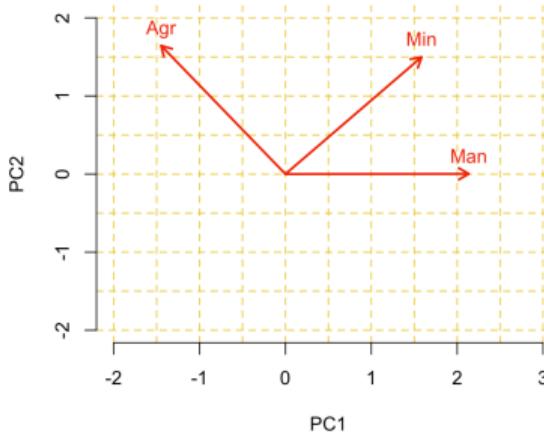
```
1 > L = eigen(M)$values  
2 > barplot(cumsum(L)/sum(L))  
3 > barplot(L/sum(L), add=T)  
4 > L/sum(L)*100  
5 [1] 61.089 32.992  5.917  
6 [4]  0.000  0.000  0.000
```



Countries (subset, 6×3)

```
1 > t(as.matrix(Z))%*%eigen(M)
  $vectors
  [,1] [,2] [,3] [,4]
2 Agr -1.45 1.64 -0.45 0
3 Min  1.59 1.50  0.49 0
4 Man  2.13 0.00 -0.67 0
5
6 > phi = eigen(M)$vectors
  [,1:2]
7 > row.names(phi) = rownames(Z)
8 > colnames(phi) = c("PC1",
  "PC2")
9 > PC1 = t(as.matrix(Z)) %*%
  phi[,1]
10 > PC2 = t(as.matrix(Z)) %*%
  phi[,2]
```

```
1 > plot(PC1,PC2)
2 > text(PC1,PC2,colnames(Z),
  pos=3)
3 > arrows(0,0,PC1,PC2)
```



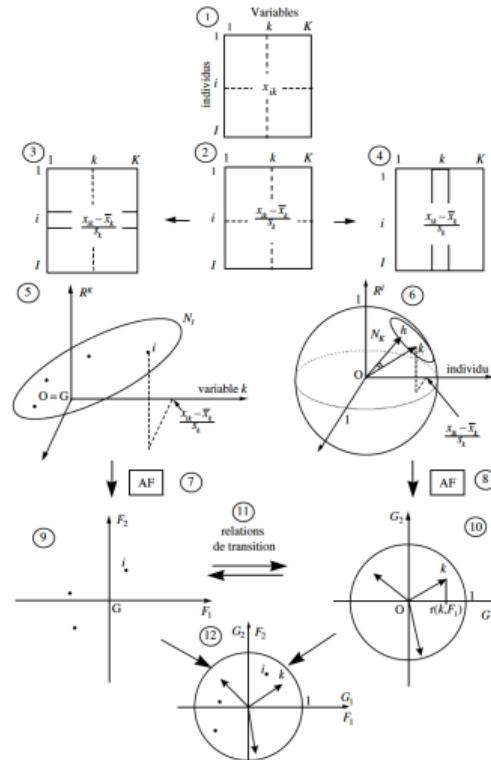
PCA, in a nutshell

Double lecture

- ▶ analyse des individus dans l'espace des variables
- ▶ analyse des variables dans l'espace des individus

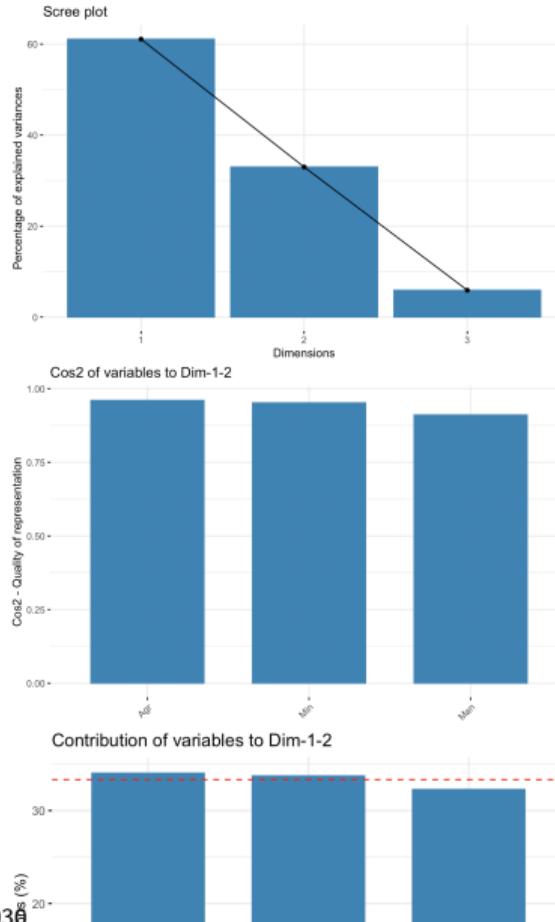
via [Escofier and Pagès \(2023\)](#)

Pour la suite, des applications...



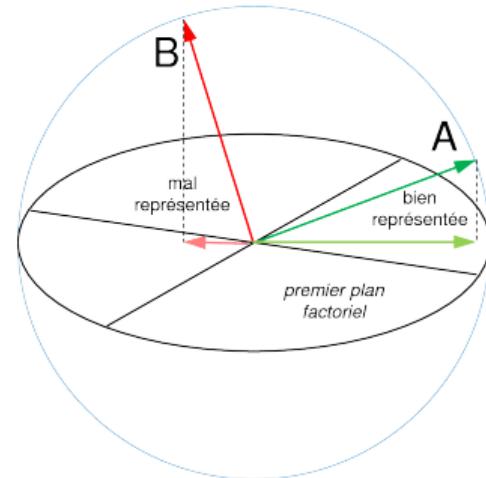
Countries (subset, 6×3)

```
1 > M = jobs[1:6,1:3]
2 > M
3
4 Belgium      Agr  Min  Man
5 Denmark      9.2  0.1 21.8
6 France       10.8 0.8 27.5
7 W.Germany    6.7  1.3 35.8
8 Ireland      23.2 1.0 20.7
9 Italy         15.9 0.6 27.6
10 > names(M) = rownames(jobs)[1:3]
11 > library(FactoMineR)
12 > res.pca = PCA(M, graph = FALSE)
13 > library("factoextra")
14 > eig.val = get_eigenvalue(res.pca)
15 > eig.val
16
17   eigenvalue  var.pct  cum.pct
18 Dim.1     1.8326    61.089    61.089
19 Dim.2     0.9897    32.992    94.082
```



Cercle des corrélations

- ▶ Si deux variables sont bien projetées, alors leur angle en projection est similaire à celui dans l'espace initial (\mathbb{R}^n).
- ▶ Corrélation entre deux variables = cosinus de l'angle entre les variables centrées-réduites.
 - ▶ un angle droit (90 degrés) correspond à une corrélation nulle;
 - ▶ un angle nul correspond à une corrélation de 1; et
 - ▶ un angle plat (180 degrés) correspond à un corrélation de -1.
- ▶ La corrélation entre la variable i et la composante principale j est $\sqrt{\lambda_j}a_{ij}$, où a_{ij} est le i ème élément du vecteur propre correspondant à la j ème plus grande valeur propre.



Countries (subset, 6×3)

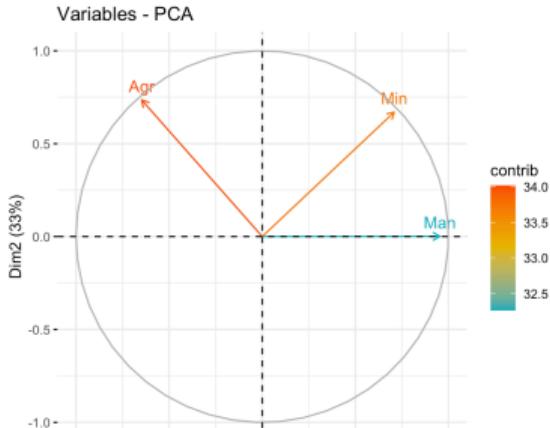
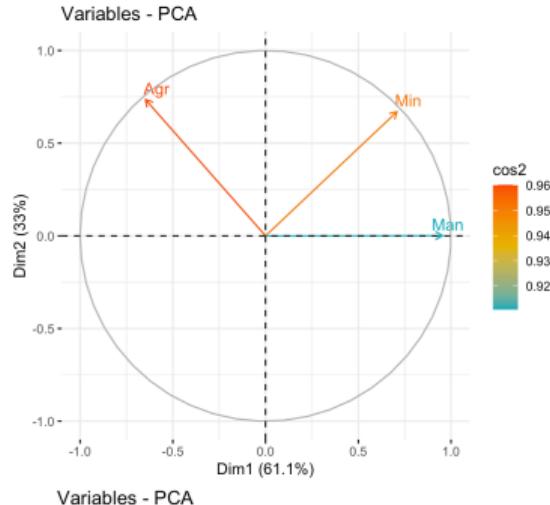
```
1 > fviz_pca_var(res.pca)
```

avec une couleur fonction des \cos^2

```
1 > fviz_pca_var(res.pca,  
2   col.var = "cos2", repel = TRUE)
```

ou de la contribution des variables

```
1 > fviz_pca_var(res.pca,  
2   col.var = "contrib")
```



Qualité de la projection des variables

- ▶ La qualité de la projection de la variable j sur l'axe G_β est mesurée par le carré du cosinus de l'angle $\theta_{j\beta}$ formé entre le vecteur \mathbf{z}^j et l'axe G_β :

$$\cos^2(\theta_{j\beta}) = \frac{\mathbf{t}_{j\beta}^2}{\|\mathbf{z}^j\|^2} = \mathbf{t}_{j\beta}^2.$$

- ▶ Si l'espace final est un plan formé des axes G_1 et G_2 , la qualité de la projection de la variable j sur ce plan est donnée par

$$\cos^2(\theta_{j(1,2)}) = \mathbf{t}_{j1}^2 + \mathbf{t}_{j2}^2.$$

- ▶ On obtient ainsi une flèche de longueur $\sqrt{\cos^2(\theta_{j(1,2)})}$ sur le cercle des corrélations pour chacune des variables. Plus la flèche est près de la circonference du cercle, meilleure est la représentation de la variable.

```
1 > fviz_cos2(res.pca, choice = "var", axes = 1:2)
```

Contribution des variables

- ▶ Les contributions des variables aux axes permettent de donner une interprétation à ceux-ci. Pour déterminer la contribution d'une variable à un axe, on évalue la proportion de l'inertie totale de l'axe expliquée par la variable.
- ▶ Pour l'axe G_β , l'inertie totale est $\lambda_\beta = \sum_{j=1}^p \mathbf{t}_{j\beta}^2$.
- ▶ La contribution de la variable j sur l'axe G_β est mesurée par

$$C(j, \beta) = \frac{\mathbf{t}_{j\beta}^2}{\lambda_\beta}.$$

- ▶ Si l'espace final est un plan formé des axes G_1 et G_2 , la contribution de la variable j sur ce plan est donnée par

$$C(j, (1, 2)) = \frac{\mathbf{a}_{j1}^2 + \mathbf{a}_{j2}^2}{\lambda_1 + \lambda_2}.$$

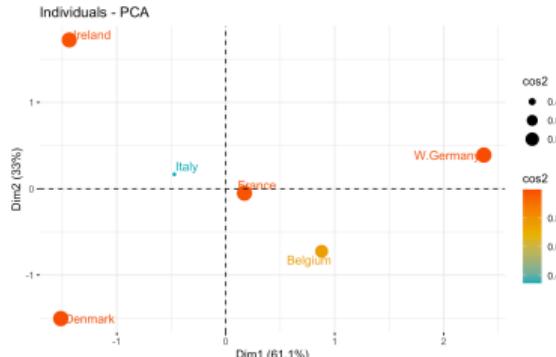
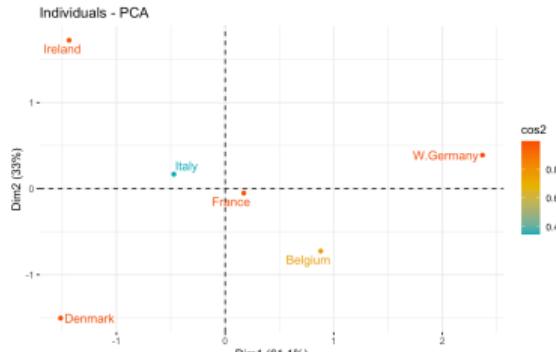
```
1 > fviz_contrib(res.pca, choice = "var", axes = 1:2)
```

Countries (subset, 6×3)

```
1 > fviz_pca_ind(res.pca)
```

```
1 > fviz_pca_ind(res.pca,  
2   col.ind = "cos2", repel = TRUE)
```

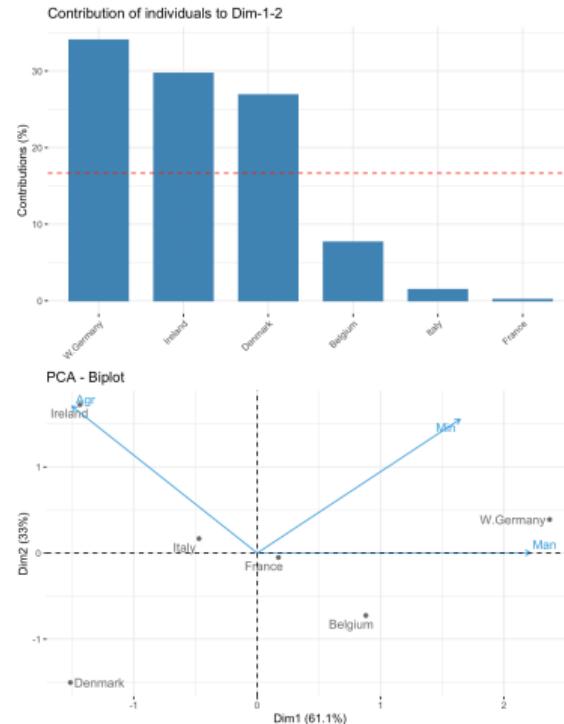
```
1 > fviz_pca_ind(res.pca,  
2   col.ind = "cos2",  
3   pointsize = "cos2",  
4   pointshape = 21)
```



Countries (subset, 6×3)

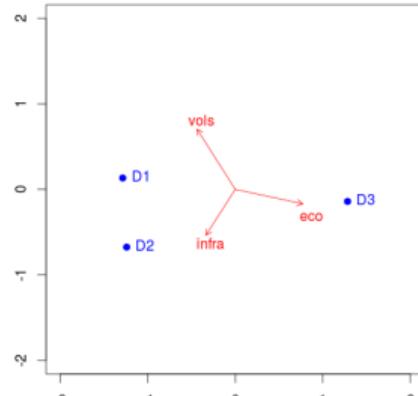
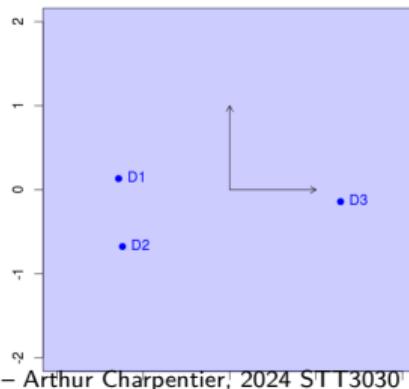
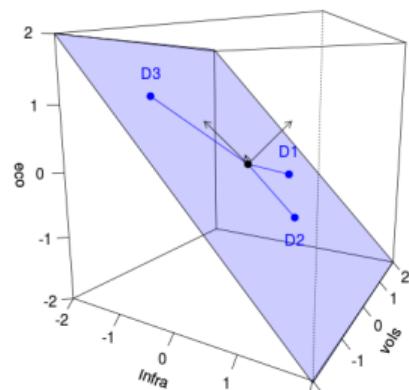
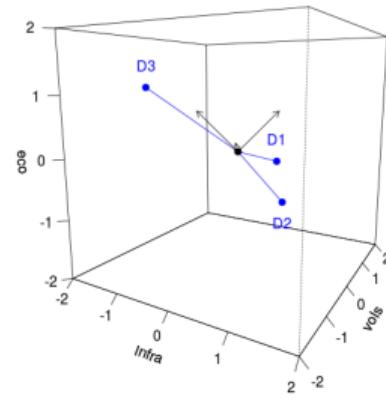
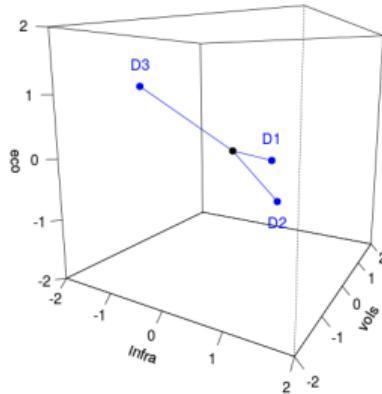
```
1 > fviz_contrib(res.pca, choice = "ind",
  axes = 1:2)
```

```
1 > fviz_pca_biplot(res.pca, repel = TRUE
  )
```



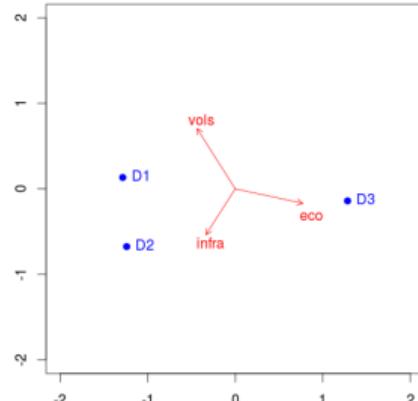
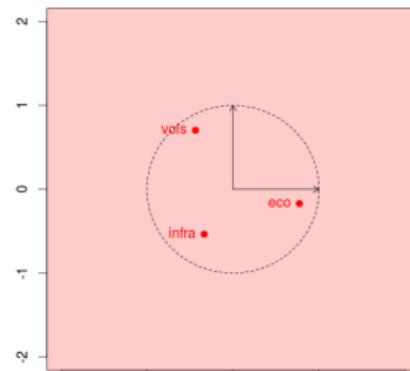
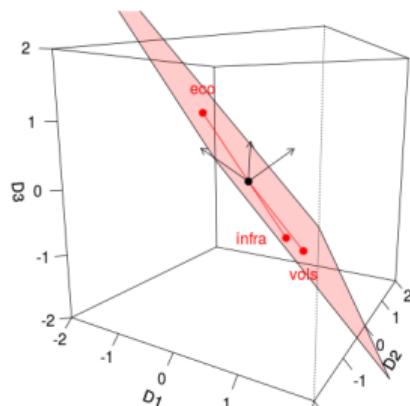
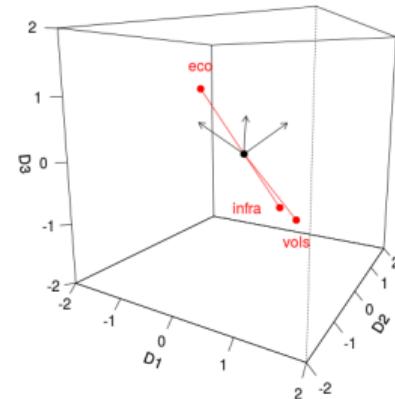
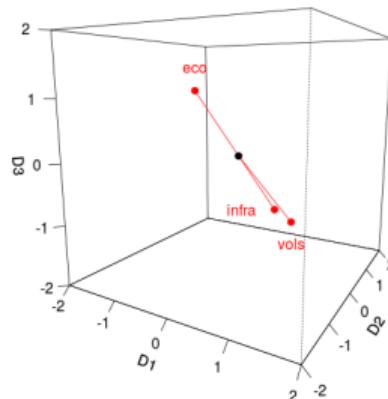
Dualité et ACP

- ▶ analyse des individus dans l'espace des variables
- ▶ analyse des variables dans l'espace des individus



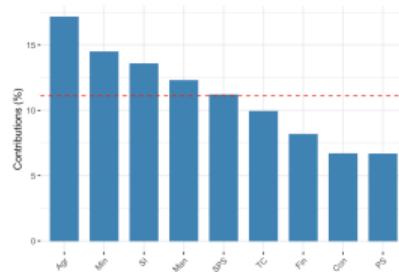
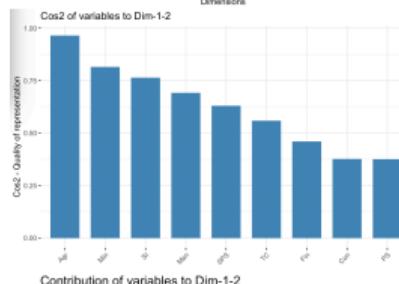
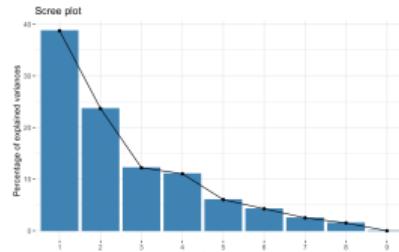
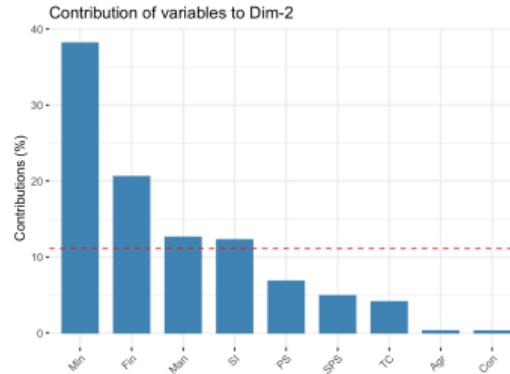
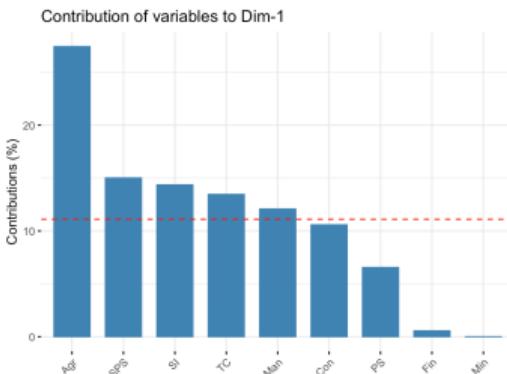
Dualité et ACP

- ▶ analyse des individus dans l'espace des variables
- ▶ analyse des variables dans l'espace des individus



Countries (entiere dataset, 26×9)

```
1 > res.pca = PCA(jobs, graph = FALSE)
2 > fviz_eig(res.pca)
3 > fviz_cos2(res.pca, choice = "var", axes = 1:2)
4 > fviz_contrib(res.pca, choice = "var", axes = 1)
```

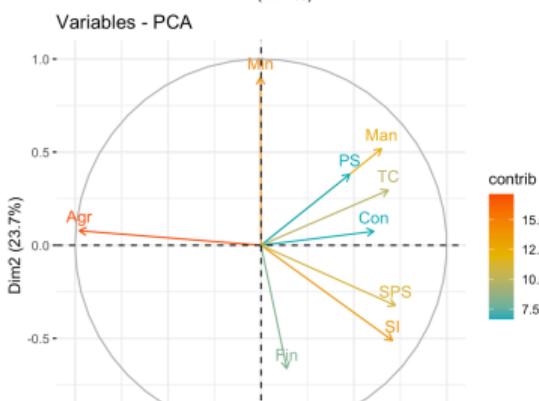
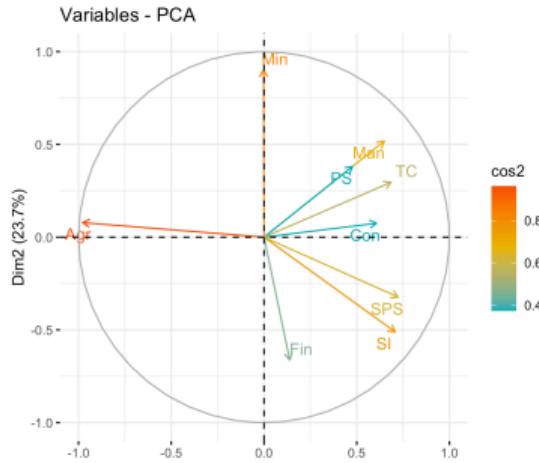
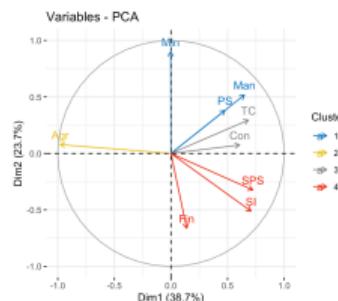
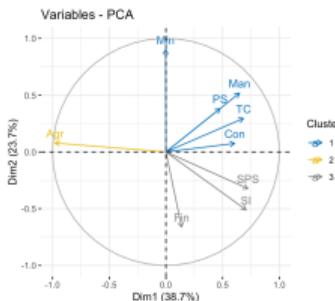


```
1 > cumsum(eigen(cor(jobs))$values)/sum(eigen(cor(
      jobs))$values)*100
2 [1] 38.74 62.41 74.62 85.67 91.71
```

Countries (entiere dataset, 26×9)

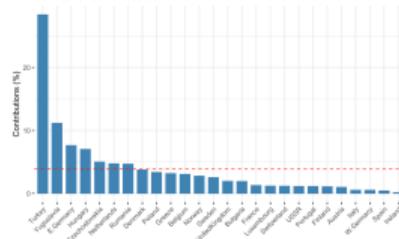
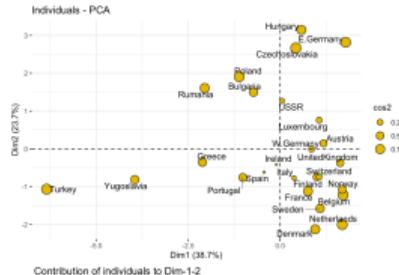
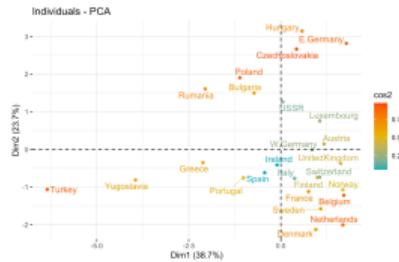
```
1 > fviz_pca_var(res.pca, col.var = "cos2"
+     , repel = TRUE)
2 > fviz_pca_var(res.pca, col.var = "
+     contrib")
```

```
1 > var <- get_pca_var(res.pca)
2 > res.km <- kmeans(var$coord, centers =
+     3, nstart = 25)
3 > grp <- as.factor(res.km$cluster)
4 > fviz_pca_var(res.pca, col.var = grp)
```



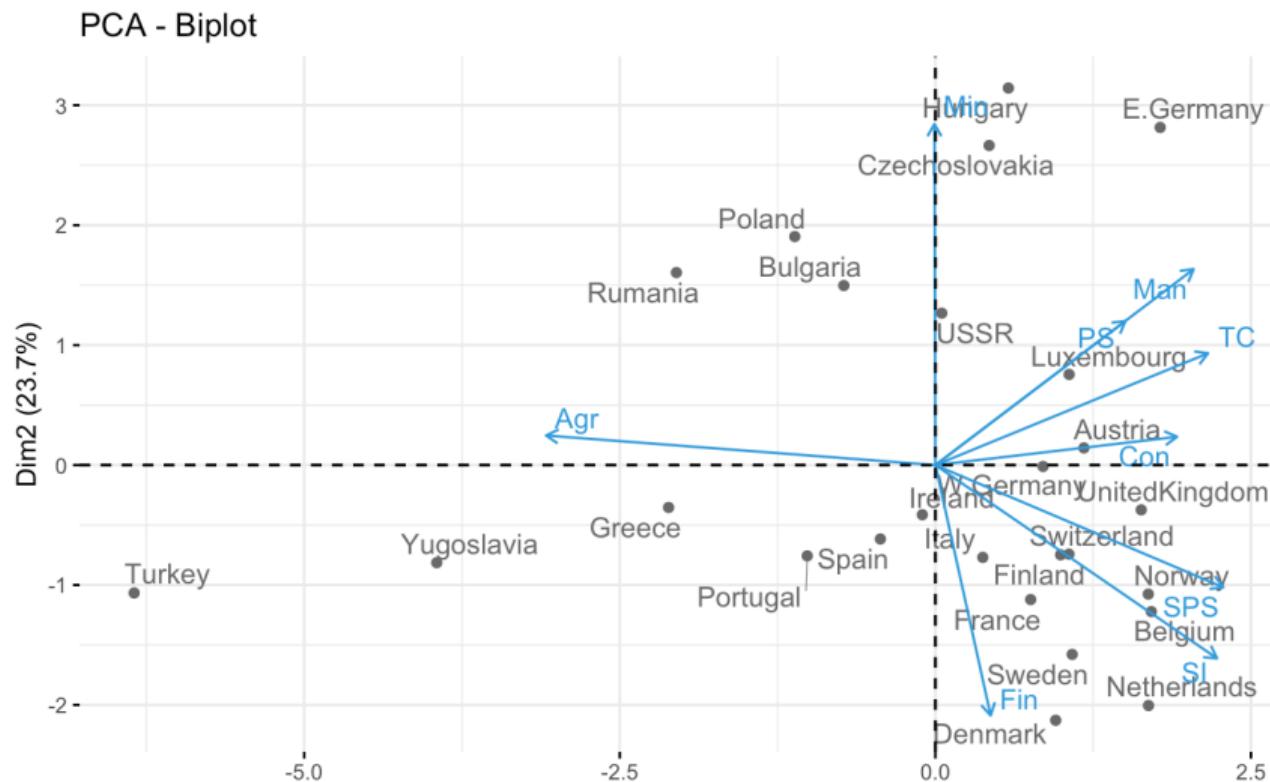
Countries (entiere dataset, 26×9)

```
1 > fviz_pca_ind(res.pca, pointsize = "cos2",
+                 pointshape = 21, repel = TRUE)
2 > fviz_contrib(res.pca, choice = "ind", axes =
+                 1:2)
```



Countries (entiere dataset, 26×9)

```
1 > fviz_pca_biplot(res.pca, repel = TRUE)
```



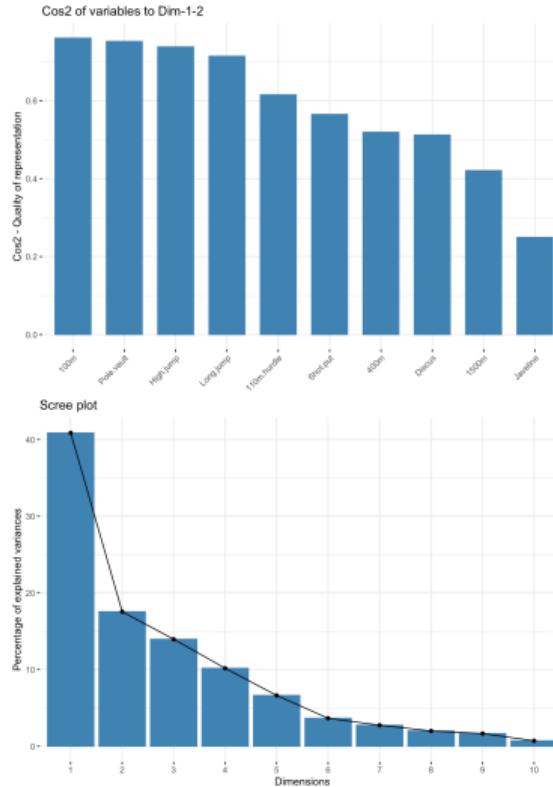
Dataset (Decathlon)

	100m +	Long jump —	Shot put —	High jump —	400m +
SEBRLE	11.04	7.58	14.83	2.07	49.81
CLAY	10.76	7.40	14.26	1.86	49.37
KARPOV	11.02	7.30	14.77	2.04	48.37
BERNARD	11.02	7.23	14.25	1.92	48.93
YURKOV	11.34	7.09	15.19	2.10	50.42
...
	110m hurdle +	Discus —	Pole vault —	Javeline —	1500m +
SEBRLE	14.69	43.75	5.02	63.19	291.7
CLAY	14.05	50.72	4.92	60.15	301.5
KARPOV	14.09	48.95	4.92	50.31	300.2
BERNARD	14.99	40.87	5.32	62.77	280.1
YURKOV	15.31	46.26	4.72	63.44	276.4
...

```
1 > library("FactoMineR")
2 > decathlon2 = decathlon[1:23, 1:10]
```

Decathlon

```
1 > res.pca <- PCA(decathlon2, graph = FALSE)
2 > res.pca = PCA(M, graph = FALSE)
3 > library("factoextra")
4 > fviz_eig(res.pca)
```

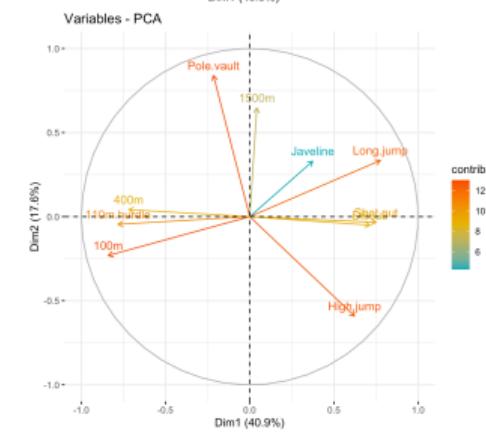
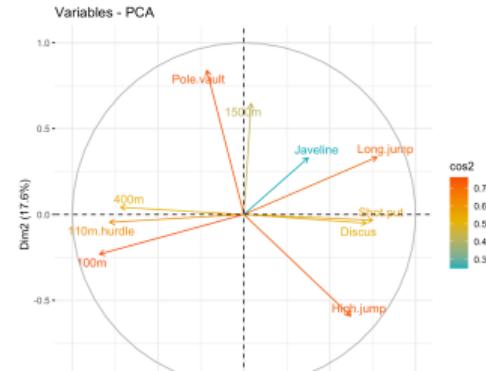
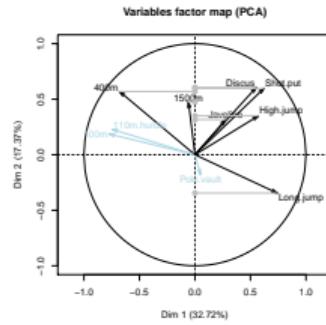
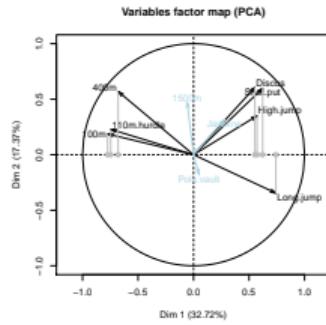
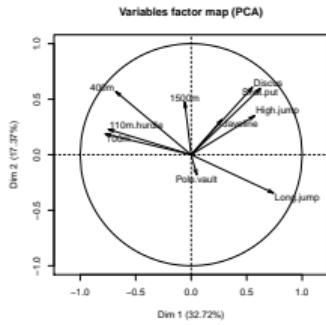


Decathlon

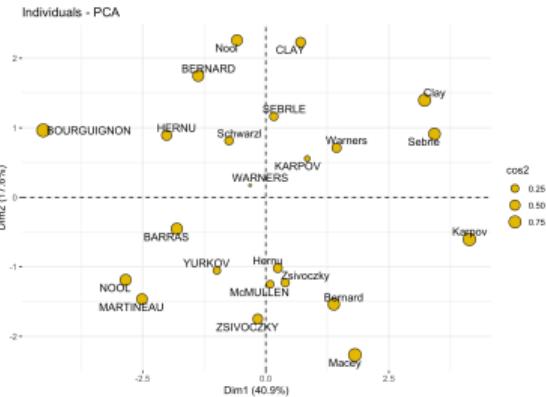
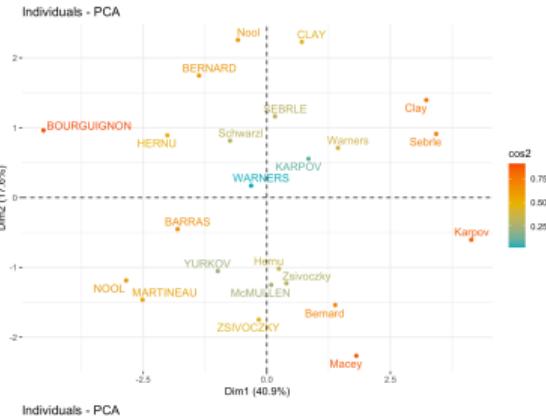
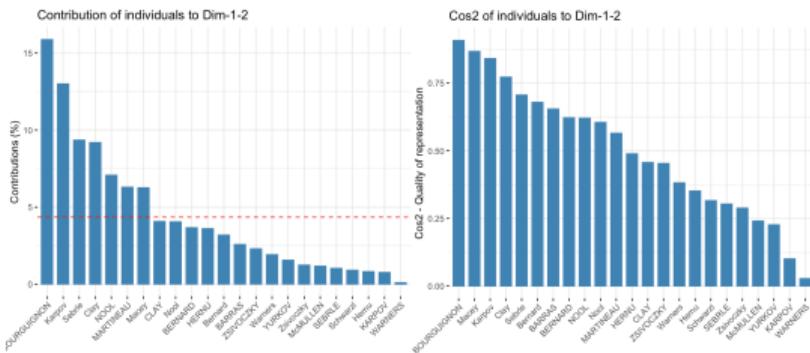
```
1 > facto_summarize(res.pca, "var", axes = 1:2)[,-1]
2
3   Dim.1   Dim.2 coord cos2 contrib
4 100m    -0.841 -0.231 0.760 0.760 13.011
5 Long.jump  0.776  0.335 0.714 0.714 12.217
6 Shot.put   0.751 -0.033 0.565 0.565  9.662
7 High.jump  0.623 -0.592 0.738 0.738 12.625
8 400m     -0.719  0.042 0.519 0.519  8.878
9 110m.hurdle -0.783 -0.045 0.615 0.615 10.522
10 Discus    0.713 -0.050 0.512 0.512  8.756
11 Pole.vault -0.216  0.840 0.752 0.752 12.867
12 Javeline   0.374  0.330 0.249 0.249  4.264
13 1500m     0.042  0.647 0.421 0.421  7.198
```

Decathlon

```
1 > fviz_pca_var(res.pca, col.var = "cos2", repel = TRUE)
2 > fviz_pca_var(res.pca, col.var = "contrib")
```

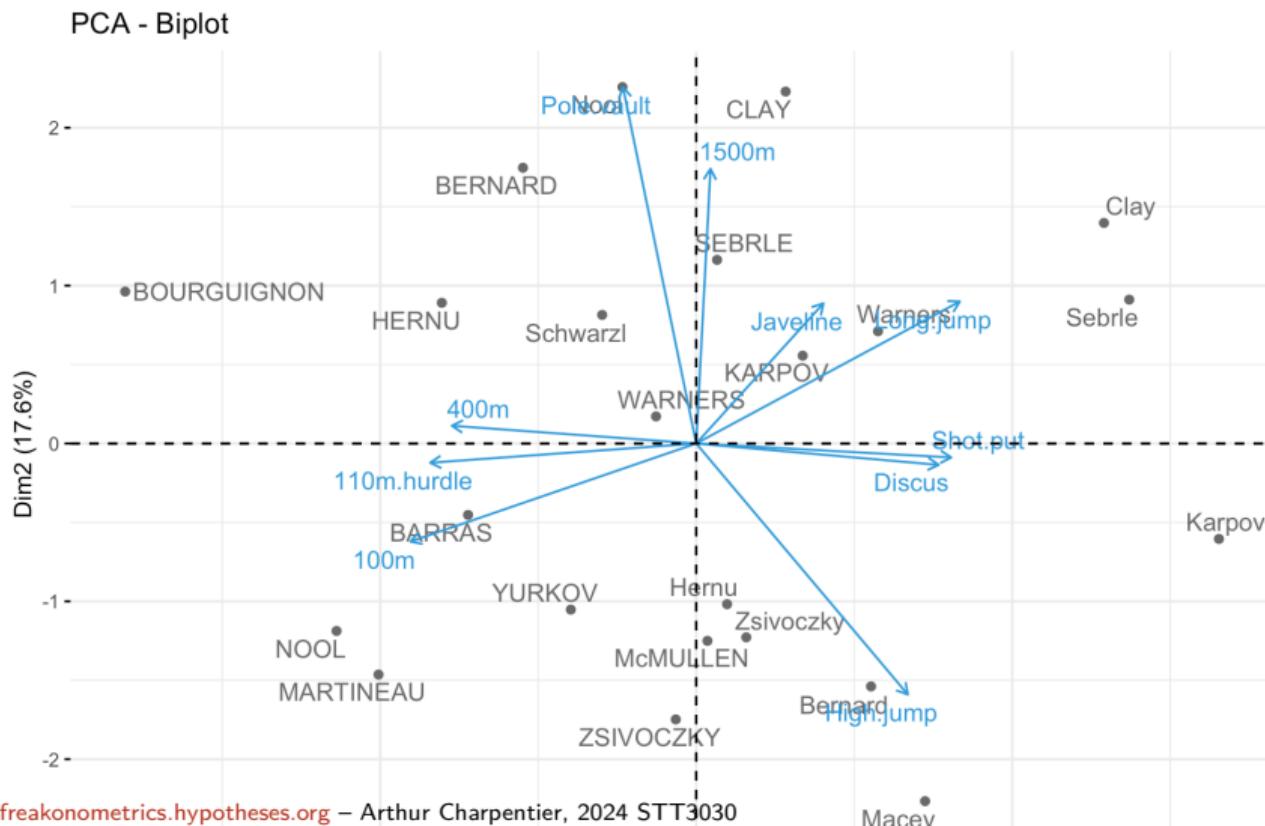


Decathlon



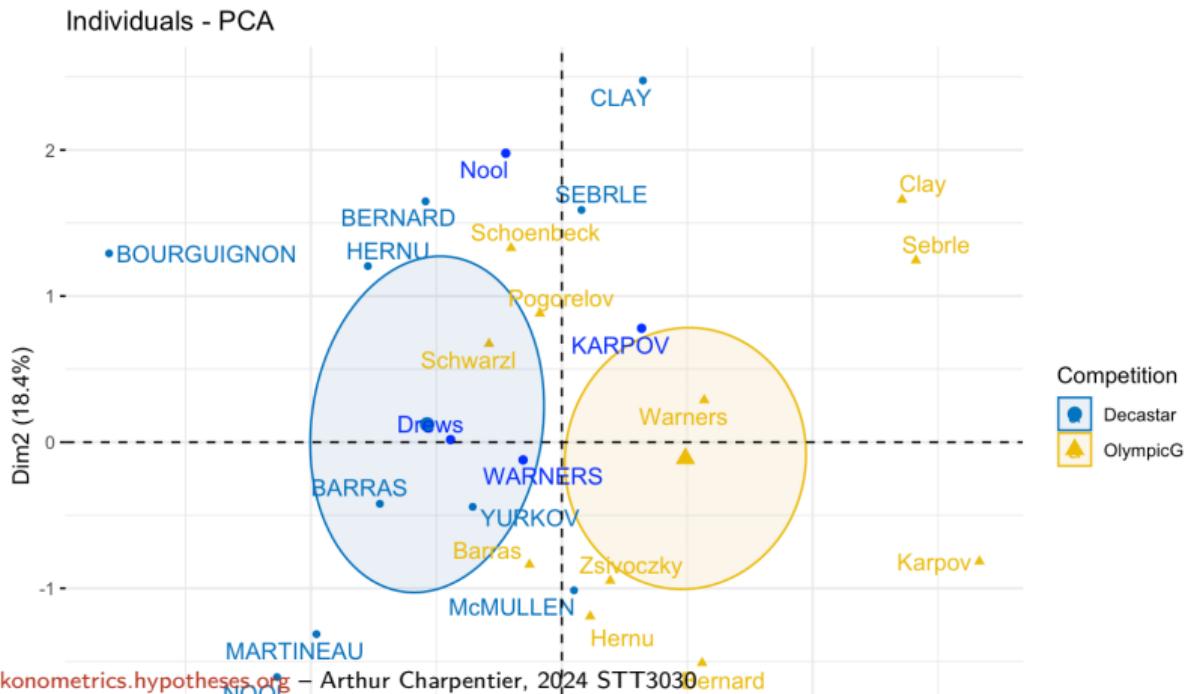
Decathlon

```
1 > fviz_pca_biplot(res.pca, repel = TRUE)
```

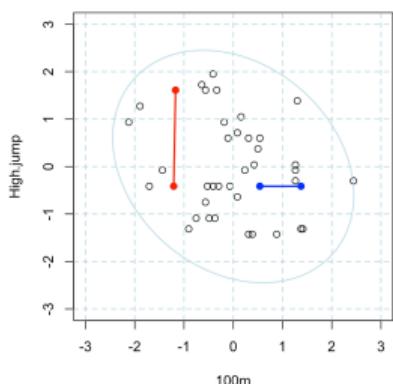
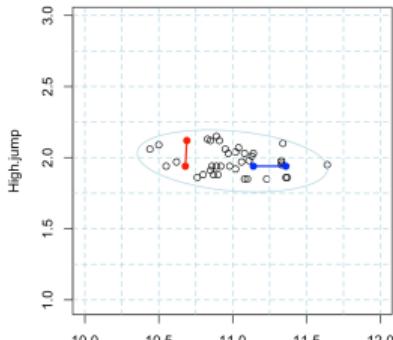
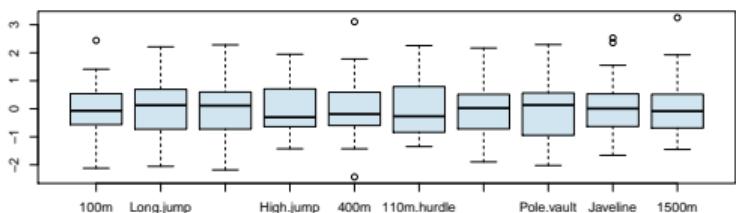
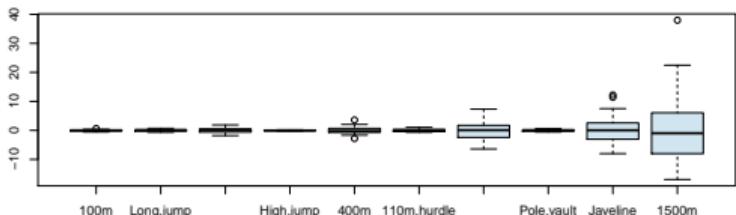
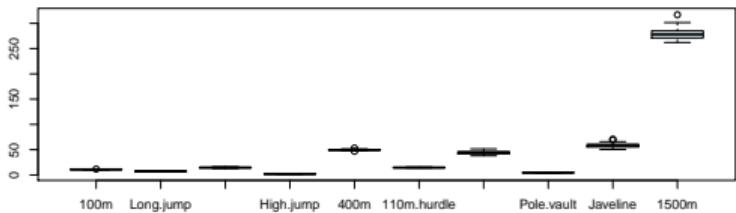


Decathlon

```
1 > res.pca = PCA(decathlon2, ind.sup = 24:27, quanti.sup = 11:12, quali.  
    sup = 13, graph=FALSE)  
2 > fviz_pca_ind(res.pca, habillage = 13, addEllipses =TRUE, ellipse.type =  
    "confidence")
```

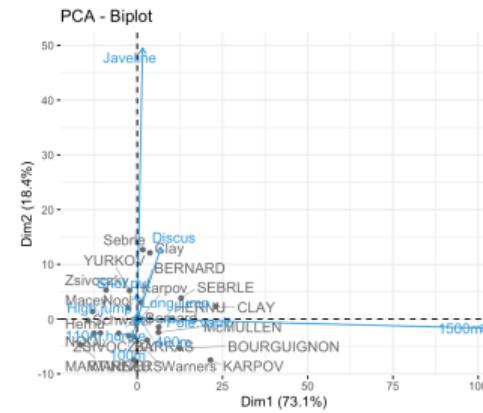
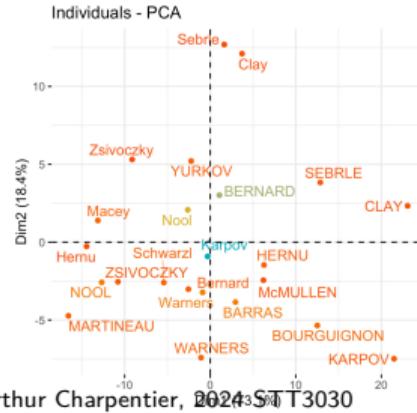
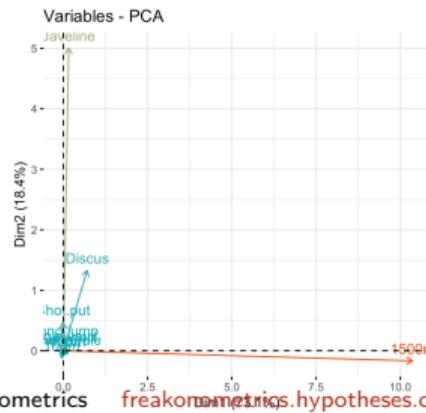
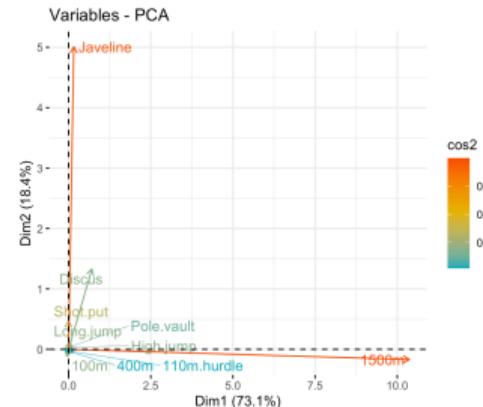
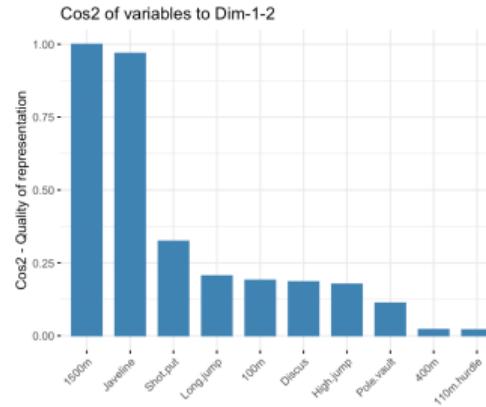
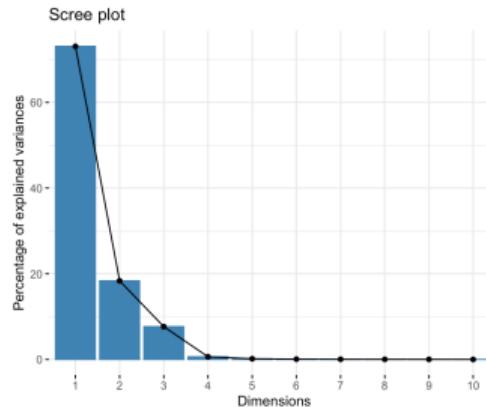


Decathlon, Euclidean vs. Mahalanobis



Decathlon, Euclidean vs. Mahalanobis

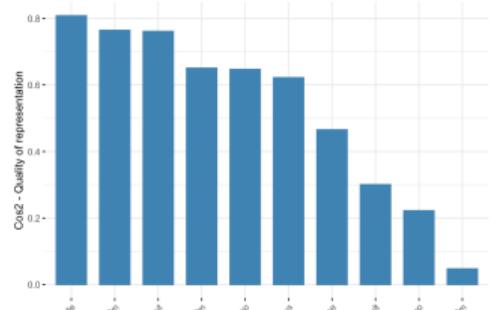
```
1 > res.pca = PCA(decathlon, graph = FALSE, scale.unit = FALSE)
```



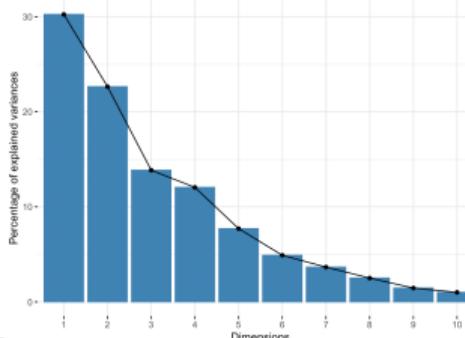
Decathlon (ranks)

```
1 > decathlon3 = decathlon2
2 > for(i in c(1,5,6,10)) decathlon3[,i]=rank(decathlon3[,i],ties="random")
3 > for(i in c(2,3,4,7,8,9)) decathlon3[,i]=rev(rank(decathlon3[,i],ties="
   random"))
4 > decathlon3
5          100m Long.jump Shot.put High.jump 400m
6 SEBRLE      15        15       5       6     18
7 CLAY         5        14      16      20     14
8 KARPOV      13        16       9       4      4
9 BERNARD     14        6      14      13     9
10 > res.pca = PCA(decathlon2.active, graph = FALSE)
```

Cos2 of variables to Dim-1-2

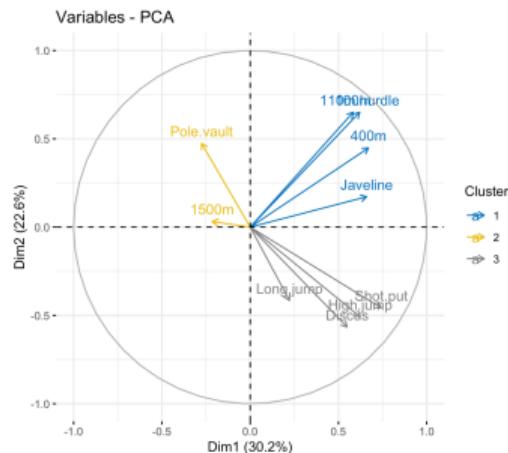
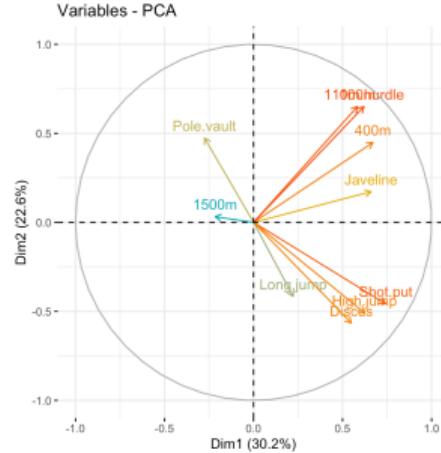
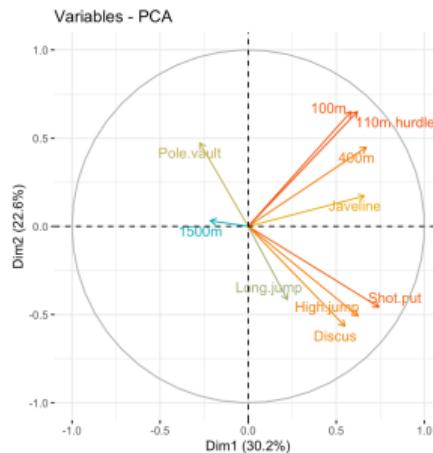


Scree plot



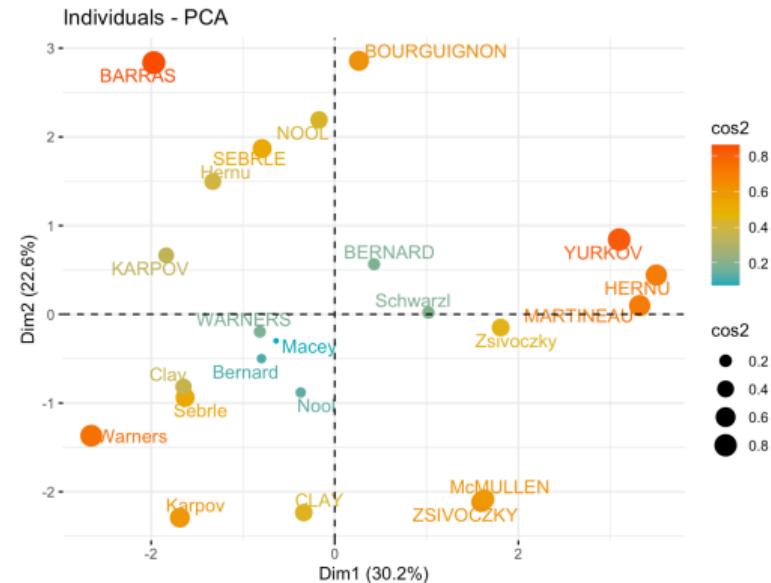
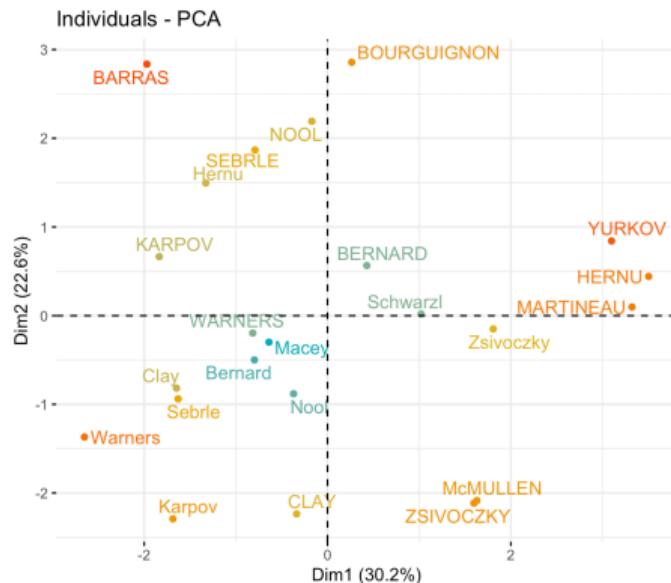
Decathlon (ranks)

```
1 > decathlon2 = decathlon[1:23, 1:10]
2 > for(i in c(1,5,6,10)) decathlon2[,i]=rank(decathlon2.active[,i],ties="
   random")
3 > for(i in c(2,3,4,7,8,9)) decathlon2[,i]=rev(rank(decathlon2.active[,i],
   ties="random"))
```



Decathlon (ranks)

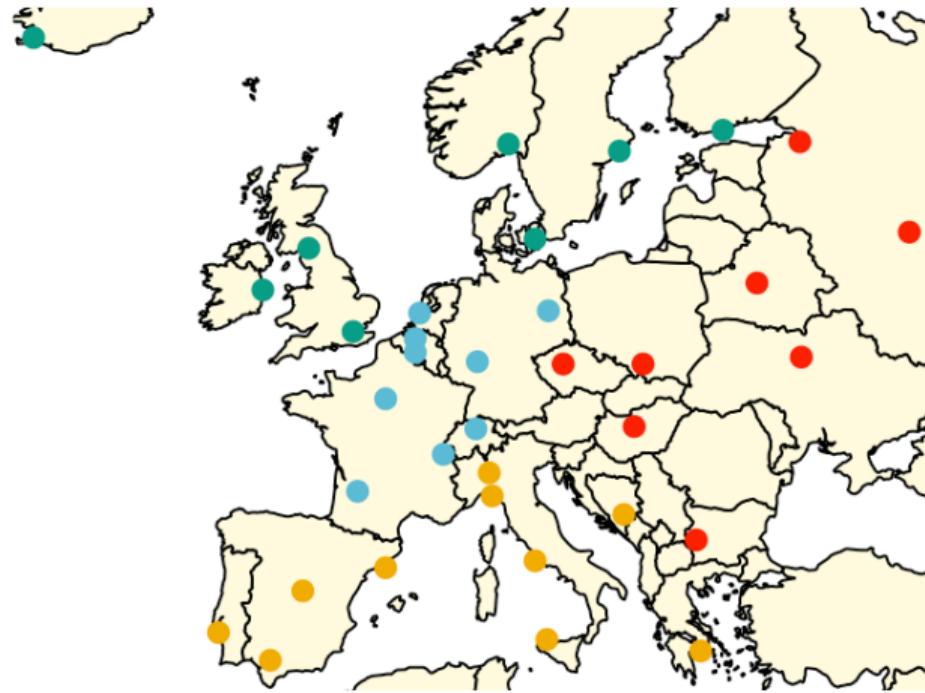
```
1 > decathlon2 = decathlon[1:23, 1:10]
2 > for(i in c(1,5,6,10)) decathlon2[,i]=rank(decathlon2.active[,i],ties="random")
3 > for(i in c(2,3,4,7,8,9)) decathlon2[,i]=rev(rank(decathlon2.active[,i],ties="random"))
```



Dataset (European Temperature)

```
1 > temperature = read.table("http://factominer.free.fr/bookV2/temperature.csv", header=TRUE, sep=";", dec=".")
```

35 cities in Europe (rows)
(average) temperature
per month (column)



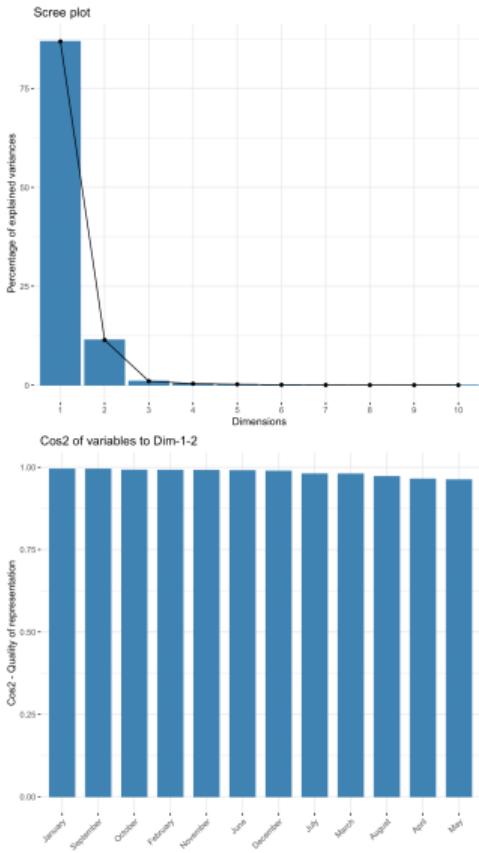
Dataset (European Temperature)

See [european_temperature.csv](#)

	Jan	Feb	Mar	Apr	May	Jun
Amsterdam	2.9	2.5	5.7	8.2	12.5	14.8
Athens	9.1	9.7	11.7	15.4	20.1	24.5
Berlin	-0.2	0.1	4.4	8.2	13.8	16.0
Brussels	3.3	3.3	6.7	8.9	12.8	15.6
Budapest	-1.1	0.8	5.5	11.6	17.0	20.2
...
	Jul	Aug	Sept	Oct	Nov	Dec
Amsterdam	17.1	17.1	14.5	11.4	7.0	4.4
Athens	27.4	27.2	23.8	19.2	14.6	11.0
Berlin	18.3	18.0	14.4	10.0	4.2	1.2
Brussels	17.8	17.8	15.0	11.1	6.7	4.4
Budapest	22.0	21.3	16.9	11.3	5.1	0.7
...

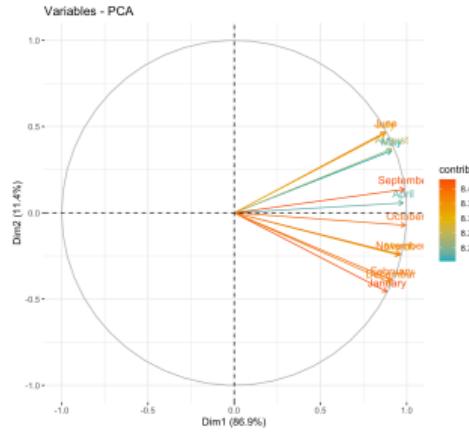
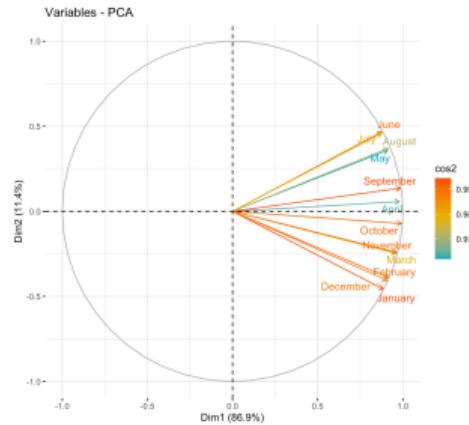
Dataset (European Temperature)

```
1 > library("FactoMineR")
2 > temperature2 = temperature[,1:12]
3 > res.pca = PCA(temperature2, graph = FALSE)
4 > library("factoextra")
5 > fviz_eig(res.pca)
6 > fviz_cos2(res.pca, choice = "var", axes = 1:2)
```



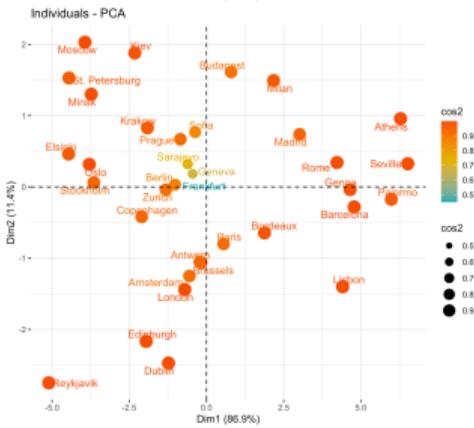
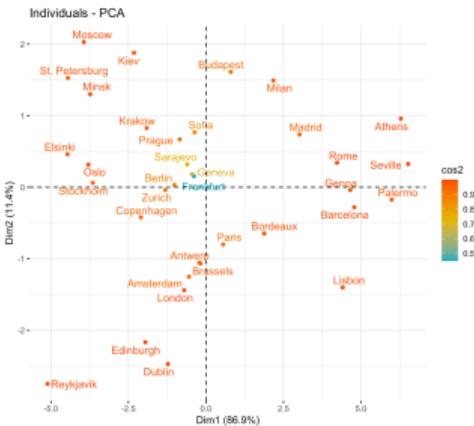
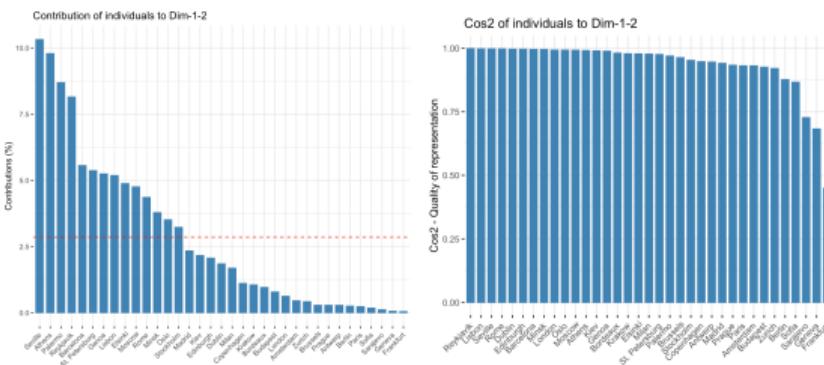
Dataset (European Temperature)

```
1 > fviz_pca_var(res.pca, col.var = "cos2"
+     , repel = TRUE)
2 > fviz_pca_var(res.pca, col.var = "
+     contrib")
```



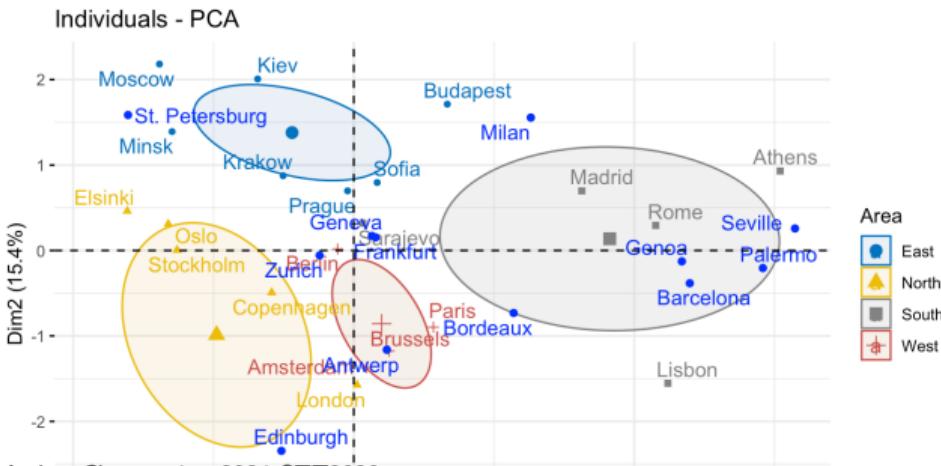
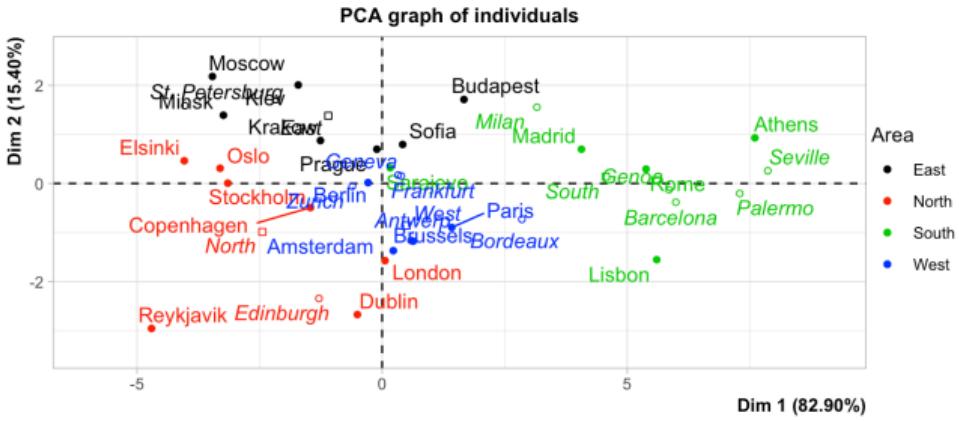
Dataset (European Temperature)

```
1 > fviz_pca_ind(res.pca, col.ind = "cos2"
2   , repel = TRUE)
3 > fviz_pca_ind(res.pca, col.ind = "cos2"
4   , pointsize = "cos2",repel = TRUE)
5 > fviz_contrib(res.pca, choice = "ind",
6   axes = 1:2)
7 > fviz_cos2(res.pca, choice = "ind",
8   axes = 1:2)
```



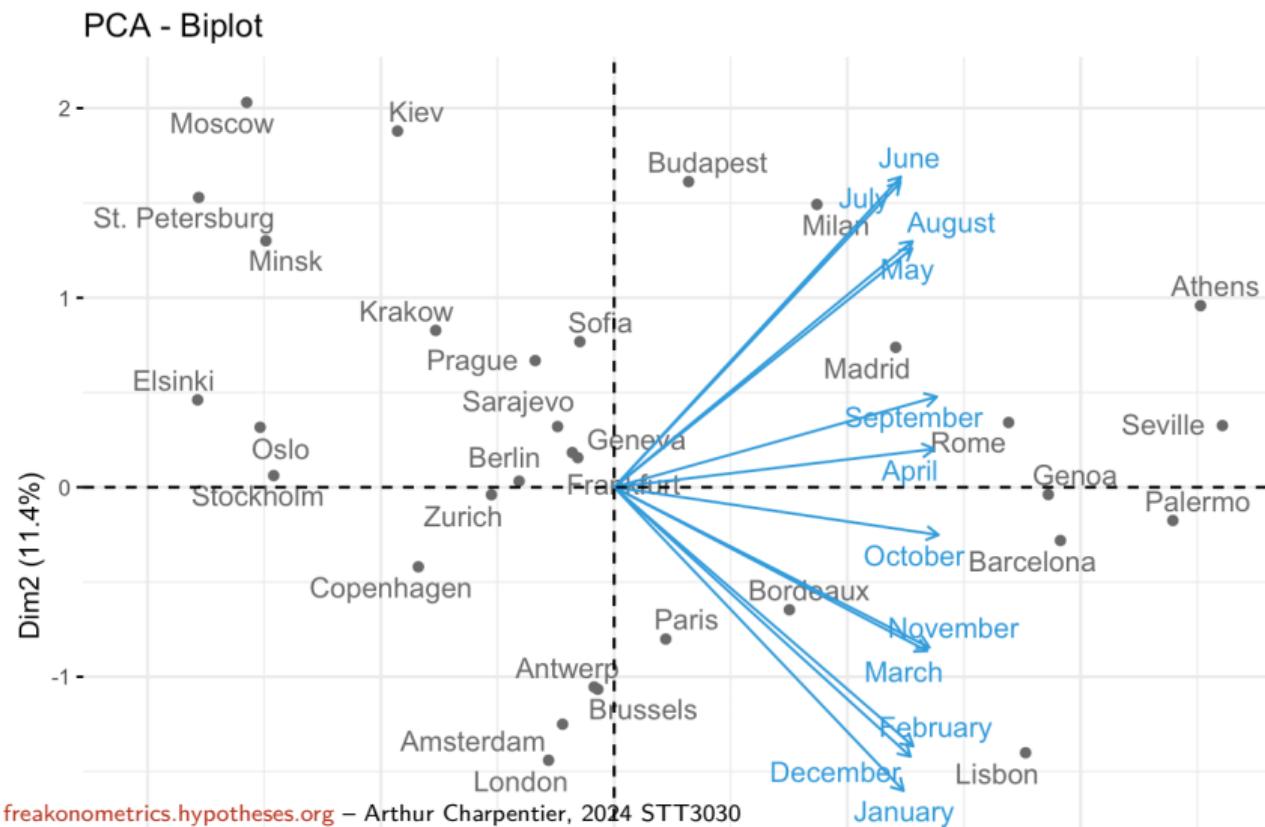
Dataset (European Temperature)

```
1 > res<-PCA(  
2   temperature ,  
3   ind.sup=24:35 ,  
4   quanti.sup=13:16 ,  
5   quali.sup=17)  
6 > plot.PCA(res ,  
7   choix="ind" ,  
8   habillage=17)
```



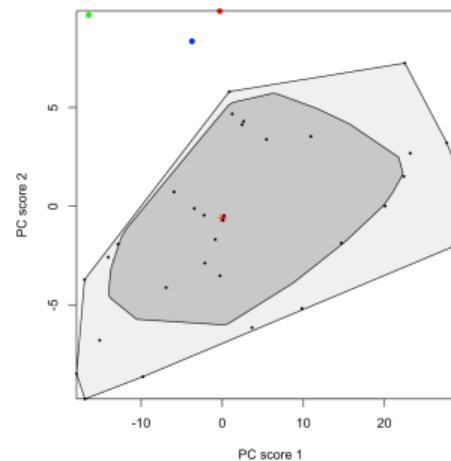
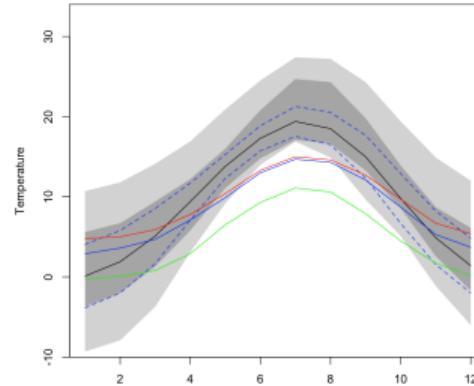
Dataset (European Temperature)

```
1 > fviz_pca_biplot(res.pca, repel = TRUE)
```



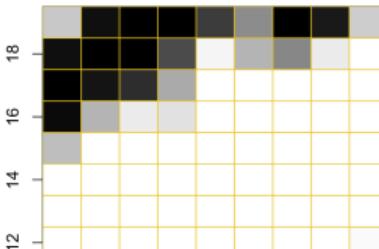
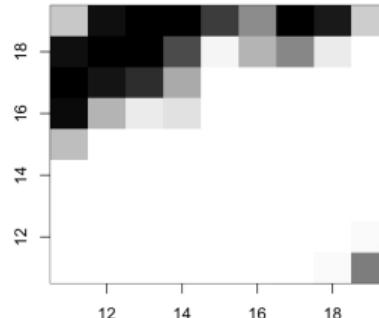
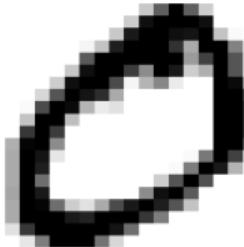
Dataset (European Temperature)

```
1 > library(rainbow)
2 > TPS=fts(x = 1:12, y = t(temperature
   [,1:12]), xname = "Month", yname =
   "Temperature")
3 > fboxplot(data = TPS, plot.type =
   "functional", type = "bag",legendpos
   = "topright",ylim=c(-10,34),axis=
   FALSE)
4 > fboxplot(data = TPS, plot.type =
   "bivariate", type = "bag",legendpos
   = "topright",axis=FALSE)
```



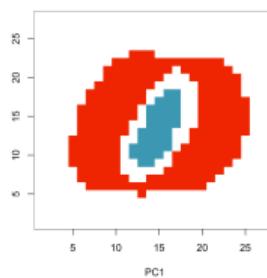
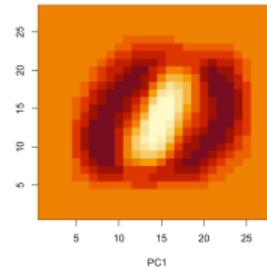
PCA & pictures

```
1 > library(keras)
2 > mnist = dataset_mnist()
3 > idx3 = which(mnist$train$y %in% c(1,0))
4 > V = mnist$train$x[idx3[1:2500], ,]
5 > Y = mnist$train$y[idx3][1:2500]
6 > MV = NULL
7 > for(i in 1:2500) MV= cbind(MV,as.vector(V[
     i,,]))
8 > MV = t(MV)
```



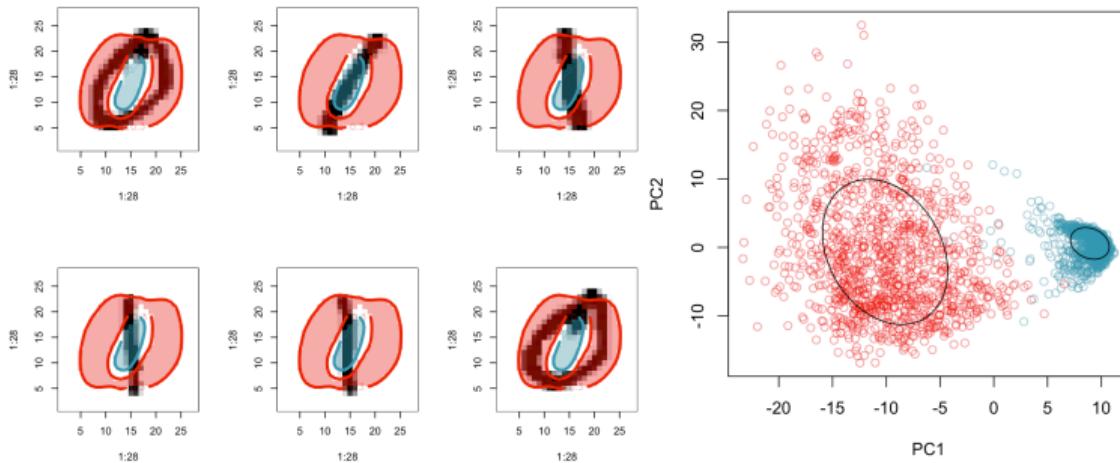
PCA & pictures: $\{0, 1\}$

```
1 > idx3 = which(mnist$train$y %in% c(1,0))  
  
1 > v = apply(MV, 2, var)  
2 > MV = MV[,which(v>0)]  
3 > res.pca = PCA(MV)  
4 > V1 = rep(0,length(v))  
5 > V1[v>0]=res.pca$var$coord[,1]  
6 > MV1=matrix(V1,28,28)  
7 > image(1:28,1:28,t(MV1)[,28:1])
```



PCA & pictures: $\{0, 1\}$

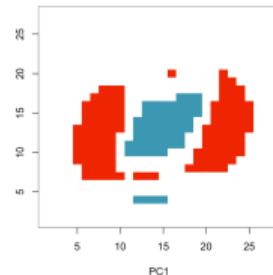
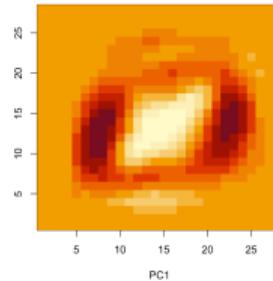
```
1 > idx3 = which(mnist$train$y %in% c(1,0))
```



PCA & pictures: {0, 8}

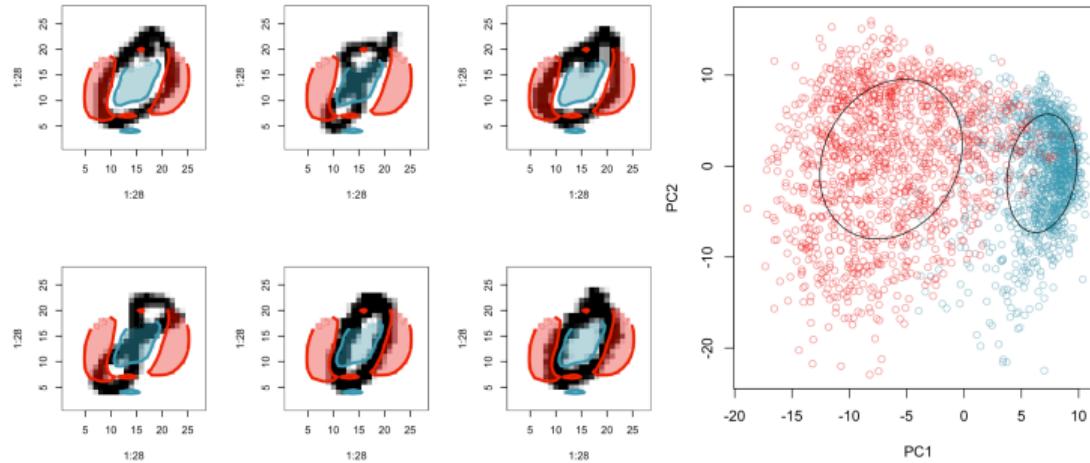
```
1 > idx3 = which(mnist$train$y %in% c(8,0))
```

```
1 > v = apply(MV, 2, var)
2 > MV = MV[, which(v>0)]
3 > res.pca = PCA(MV)
4 > V1 = rep(0, length(v))
5 > V1[v>0]=res.pca$var$coord[,1]
6 > MV1=matrix(V1, 28, 28)
7 > image(1:28, 1:28, t(MV1) [,28:1])
```



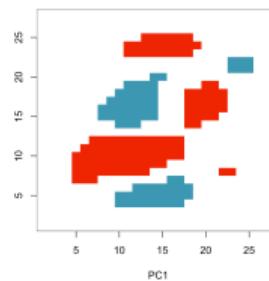
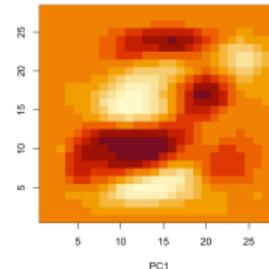
PCA & pictures: {0, 8}

```
1 > idx3 = which(mnist$train$y %in% c(8,0))
```



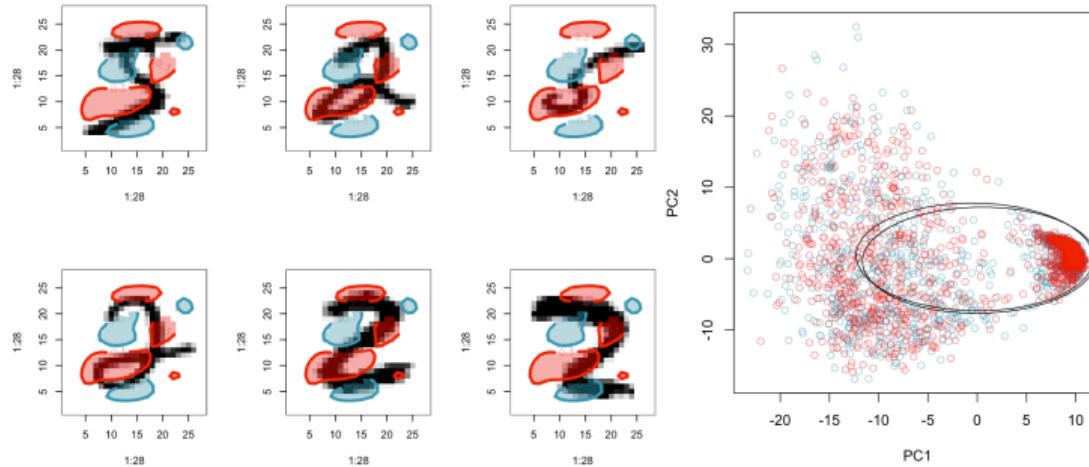
PCA & pictures: {2, 5}

```
1 > idx3 = which(mnist$train$y %in% c(2,5))  
  
1 > v = apply(MV, 2, var)  
2 > MV = MV[,which(v>0)]  
3 > res.pca = PCA(MV)  
4 > V1 = rep(0,length(v))  
5 > V1[v>0]=res.pca$var$coord[,1]  
6 > MV1=matrix(V1,28,28)  
7 > image(1:28,1:28,t(MV1)[,28:1])
```

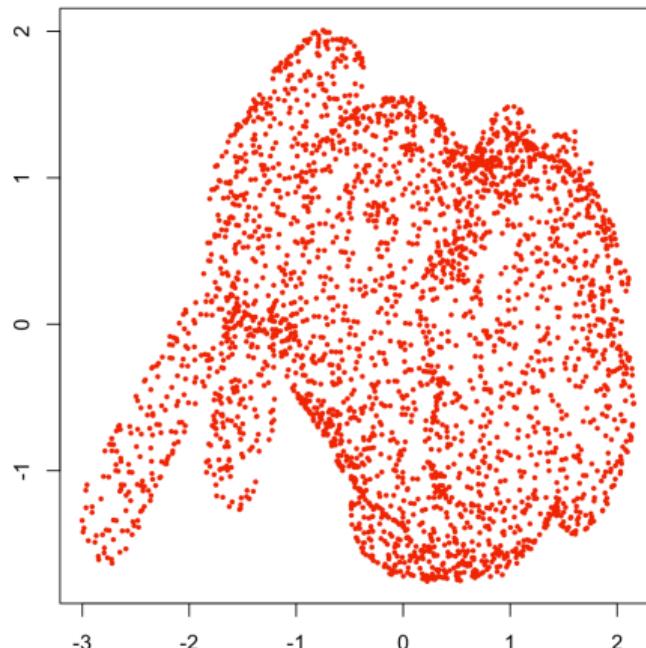
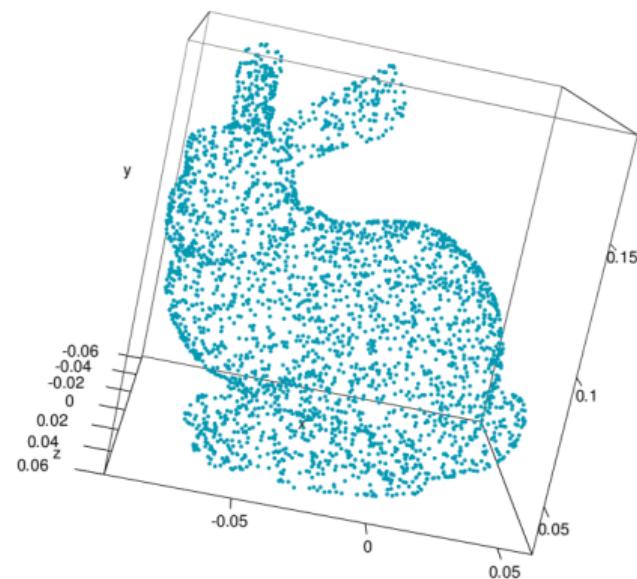


PCA & pictures: {2, 5}

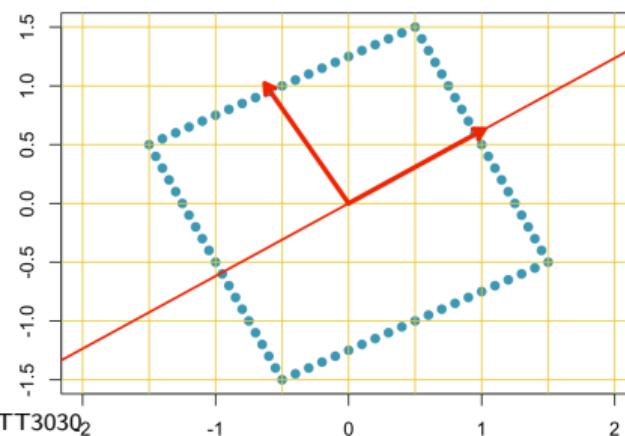
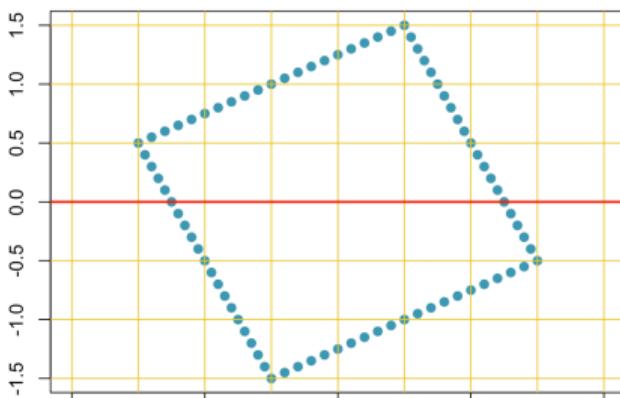
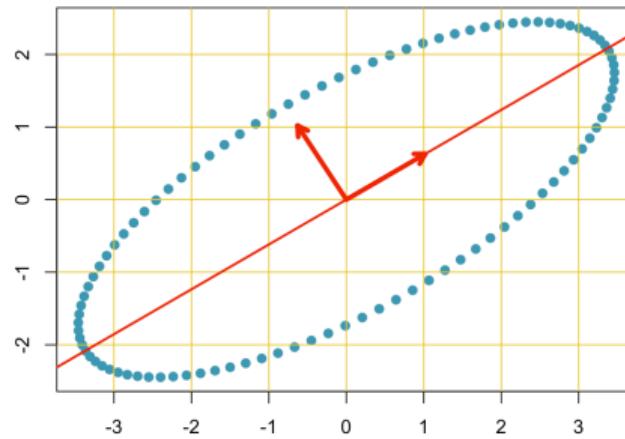
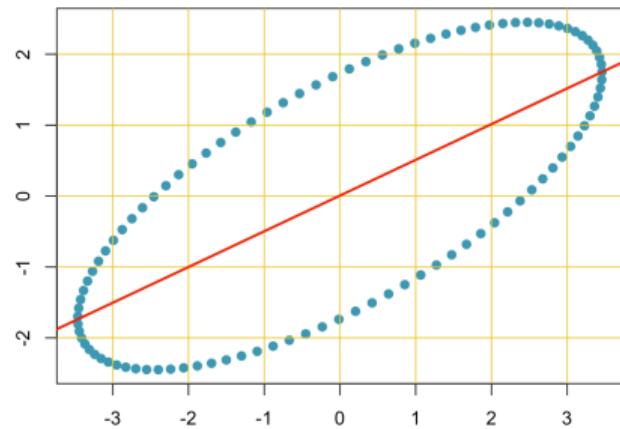
```
1 > idx3 = which(mnist$train$y %in% c(2,5))
```



PCA & projections

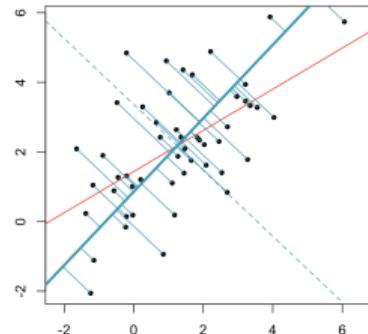
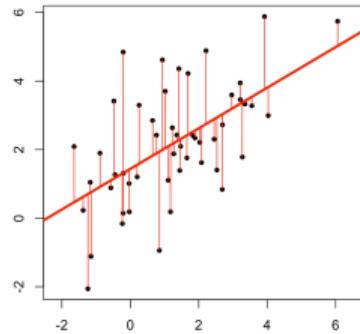
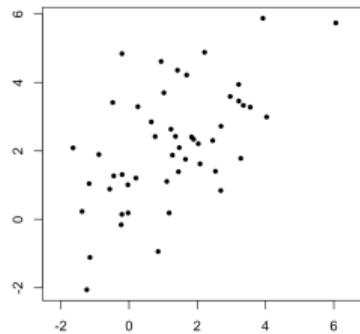


PCA vs. regression (OLS)



PCA & regression

Why not consider a PCA on the dataset $[Y, \mathbf{X}]$?



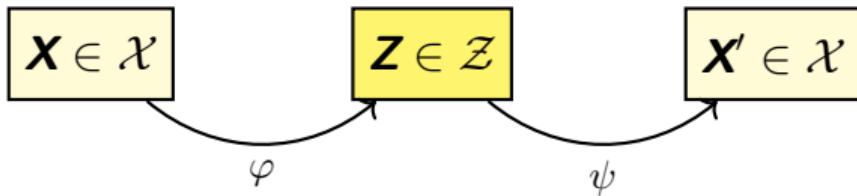
With **PCA**, the first component $\mathbf{z}_1 = \omega_1^\top \mathbf{x}$ is

$$\omega_1 = \underset{\|\omega\|=1}{\operatorname{argmax}} \left\{ \text{Var}[\omega^\top \mathbf{X}] \right\} = \underset{\|\omega\|=1}{\operatorname{argmax}} \left\{ \langle \omega^\top \mathbf{x}, \omega^\top \mathbf{x} \rangle \right\}$$

With **PLS**, the first component $\mathbf{z}_1 = \omega_1^\top \mathbf{x}$ is

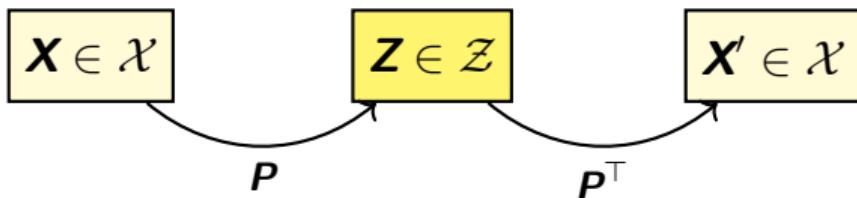
$$\omega_1 = \underset{\|\omega\|=1}{\operatorname{argmax}} \left\{ \text{Cov}[Y, \omega^\top \mathbf{X}] \right\} = \underset{\|\omega\|=1}{\operatorname{argmax}} \left\{ \langle y, \omega^\top \mathbf{x} \rangle \right\}$$

PCA & neural nets: Autoencoders



The error function of a **nonlinear autoencoder** is

$$\|\mathbf{X} - \mathbf{X}'\|^2 = \|\mathbf{X} - \psi \circ \varphi(\mathbf{X})\|^2 = \sum_{i=1}^n (\psi \circ \varphi(\mathbf{x}_i) - \mathbf{x}_i)^\top (\psi \circ \varphi(\mathbf{x}_i) - \mathbf{x}_i)$$



The error function of a **linear autoencoder** is

$$\|\mathbf{X} - \mathbf{X}'\|^2 = \|\mathbf{X} - P^\top P \mathbf{X}\|^2 = \sum_{i=1}^n (P^\top P \mathbf{x}_i - \mathbf{x}_i)^\top (P^\top P \mathbf{x}_i - \mathbf{x}_i)$$

Autoencoders

The error function of a linear autoencoder is

$$\sum_{i=1}^n (\mathbf{P}^\top \mathbf{P} \mathbf{x}_i - \mathbf{x}_i)^\top (\mathbf{P}^\top \mathbf{P} \mathbf{x}_i - \mathbf{x}_i)$$

i.e.

$$\sum_{i=1}^n \text{trace} \left[(\mathbf{P}^\top \mathbf{P} - \mathbb{I}) \mathbf{x}_i \mathbf{x}_i^\top (\mathbf{P}^\top \mathbf{P} - \mathbb{I}) \right]$$

$$\text{trace} \left[(\mathbf{P}^\top \mathbf{P} - \mathbb{I}) \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top (\mathbf{P}^\top \mathbf{P} - \mathbb{I}) \right]$$

the middle term is a covariance matrix, thus it is $\mathbf{V}\Delta\mathbf{V}^\top$, so we recognize

$$\|(\mathbf{P}^\top \mathbf{P} - \mathbb{I}) \mathbf{V}\Delta\mathbf{V}^{1/2}\|_F^2$$

where $\|\cdot\|_F$ denotes the Frobenius / entrywise ℓ_2 norm of a matrix,

$$\|\mathbf{M}\|_F^2 = \sum_{i,j} M_{i,j}^2 = \text{trace}(\mathbf{M}\mathbf{M}^\top) \text{ where } \mathbf{M} = [M_{i,j}]$$

Alternative Approach

Let $\|\cdot\|_F$ denote the Frobenius / entrywise ℓ_2 norm of a matrix,

$$\|\mathbf{M}\|_F^2 = \sum_{i,j} M_{i,j}^2 \text{ where } \mathbf{M} = [M_{i,j}]$$

Consider the following problem,

$$\min_{\mathbf{Y}} \{\|\mathbf{X} - \mathbf{Y}\|_F^2\} \text{ subject to } \text{rank}(\mathbf{Y}) = k \quad (\leq \text{rank}(\mathbf{X}))$$

If $\mathbf{X} = \mathbf{U}\Delta\mathbf{V}^\top$ then $\mathbf{Y} = \mathbf{U}_k\Delta_k\mathbf{V}_k^\top$ where we keep the first k columns of \mathbf{U} , \mathbf{V} and Δ .

One can rewrite

$$\min_{P \in \Pi} \{\|\mathbf{X} - P\mathbf{X}\|_F^2\} \text{ subject to } \text{rank}(P) = k$$

where Π is the set of projection matrices.

If $\mathbf{S} = \mathbf{X}^\top\mathbf{X}$, we can write (equivalently)

$$\max_{P \in \Pi} \{\text{trace}(\mathbf{SP})\} \text{ subject to } \text{rank}(P) = k$$

Alternative Approach

or

$$\max_{P \in \mathcal{P}} \{\text{trace}(SP)\}$$

where

$$\mathcal{P} = \{P \in \Pi : \text{eigenvalues}(P) \in \{0, 1\} \text{ and } \text{trace}(P) = k\}$$

Then $P^* = V_k V_k^\top$

Note that \mathcal{P} is note convex. But one can consider

$$\mathcal{P}_* = \{P \in \Pi : \text{eigenvalues}(P) \in [0, 1] \text{ and } \text{trace}(P) = k\}$$

which is the convex Hull of \mathcal{P} . And one can prove that the convex problem

$$\max_{P \in \mathcal{P}_*} \{\text{trace}(SP)\}$$

Same solution as the non-convex one (proof in [On a Theorem of Weyl Concerning Eigenvalues of Linear Transformations](#)).

Pour le prochain cours:

- ▶ Lecture: 12 ([Bishop and Nasrabadi \(2006\)](#)), 12.2 ([James \(2013\)](#))
- ▶ Préparation: Examen final, révision
- ▶ Lab: 12.5.1
- ▶ Exercices: 12.6.10

Références

- Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.
- Escofier, B. and Pagès, J. (2023). *Analyses factorielles simples et multiples*. Dunod.
- James, G. (2013). *An introduction to statistical learning*. Springer.