# Matrix methods – Principal component analysis

## MATH 2740 – Mathematics of Data Science – Lecture 11

**Julien Arino**
julien.arino@umanitoba.ca

**Department of Mathematics @ University of Manitoba**

**Fall 202X**

The University of Manitoba campuses are located on original lands of Anishinaabeg, Ininew, Anisininew, Dakota and Dene peoples, and on the National Homeland of the Red River Métis. We respect the Treaties that were made on these territories, we acknowledge the harms and mistakes of the past, and we dedicate ourselves to move forward in partnership with Indigenous communities in a spirit of Reconciliation and collaboration.

# Outline

Principal component analysis (PCA)

# Principal component analysis (PCA)

# Dimensionality reduction

One of the reasons the SVD is used is for dimensionality reduction. However, SVD has many many other uses

Now we look at another dimensionality reduction technique, PCA

PCA is often used as a blackbox technique, here we take a look at the math behind it

# What is PCA?

Linear algebraic technique

Helps reduce a complex dataset to a lower dimensional one

Non-parametric method: does not assume anything about data distribution (distribution from the statistical point of view)

# Principal component analysis (PCA)

- **A crash course on probability**
- A running example: fingerprints
- Change of basis
- Back to PCA
- A 2D example to begin: hockey players
- Back to fingerprints

# Brief “review” of some probability concepts

Proper definition of *probability* requires to use *measure theory*.. will not get into details here

A **random variable** $X$ is a *measurable* function $X : \Omega \to E$, where $\Omega$ is a set of outcomes (*sample space*) and $E$ is a measurable space

$\mathbb{P}(X \in S \subseteq E) = \mathbb{P}(\omega \in \Omega | X(\omega) \in S)$

**Distribution function** of a r.v., $F(x) = \mathbb{P}(X \leq x)$, describes the distribution of a r.v.

R.v. can be discrete or continuous or .. other things.

## Definition 1 (Variance)

Let $X$ be a random variable. The **variance** of $X$ is given by

$$\text{Var } X = E\left[(X - E(X))^2\right]$$

where $E$ is the expected value

## Definition 2 (Covariance)

Let $X$, $Y$ be jointly distributed random variables. The **covariance** of $X$ and $Y$ is given by

$$\text{cov}(X, Y) = E\left[(X - E(X))\,(Y - E(Y))\right]$$

Note that $\text{cov}(X, X) = E\left[(X - E(X))^2\right] = \text{Var } X$

# In practice: “true law” versus “observation”

In statistics: we reason on the *true law* of distributions, but we usually have only access to a sample

We then use **estimators** to .. estimate the value of a parameter, e.g., the mean, variance and covariance

## Definition 3 (Unbiased estimators of the mean and variance)

Let $x_1, \ldots, x_n$ be data points (the *sample*) and

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

be the **mean** of the data. An unbiased estimator of the variance of the sample is

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

## Definition 4 (Unbiased estimator of the covariance)

Let $(x_1, y_1), \ldots, (x_n, y_n)$ be data points,

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ and } \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

be the means of the data. An estimator of the covariance of the sample is

$$\operatorname{cov}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

# What does covariance do?

Variance explains how data disperses around the mean, in a 1-D case

Covariance measures the relationship between two dimensions. E.g., height and weight

More than the exact value, the sign is important:

- $\text{cov}(X, Y) > 0$: both dimensions change in the same "direction"; e.g., larger height usually means higher weight
- $\text{cov}(X, Y) < 0$: both dimensions change in reverse directions; e.g., time spent on social media and performance in this class
- $\text{cov}(X, Y) = 0$: the dimensions are independent from one another; e.g., sex/gender and "intelligence"

# The covariance matrix

Typically, we consider more than 2 variables..

## Definition 5

Suppose $p$ random variables $X_1, \ldots, X_p$. Then the covariance matrix is the symmetric matrix

$$\begin{pmatrix} \operatorname{cov}(X_1, X_1) & \operatorname{cov}(X_1, X_2) & \cdots & \operatorname{cov}(X_1, X_p) \\ \operatorname{cov}(X_2, X_1) & \operatorname{cov}(X_2, X_2) & \cdots & \operatorname{cov}(X_2, X_p) \\ \vdots & \vdots & & \vdots \\ \operatorname{cov}(X_p, X_1) & \operatorname{cov}(X_p, X_2) & \cdots & \operatorname{cov}(X_p, X_p) \end{pmatrix}$$

i.e., using the properties of covariance,

$$\begin{pmatrix} \operatorname{Var} X_1 & \operatorname{cov}(X_1, X_2) & \cdots & \operatorname{cov}(X_1, X_p) \\ \operatorname{cov}(X_1, X_2) & \operatorname{Var} X_2 & \cdots & \operatorname{cov}(X_2, X_p) \\ \vdots & \vdots & & \vdots \\ \operatorname{cov}(X_1, X_p) & \operatorname{cov}(X_2, X_p) & \cdots & \operatorname{Var} X_p \end{pmatrix}$$

# Principal component analysis (PCA)

- A crash course on probability
- **A running example: fingerprints**
- Change of basis
- Back to PCA
- A 2D example to begin: hockey players
- Back to fingerprints

# Example of a PCA problem

We collect a bunch of information about a bunch of people.. for instance this data from Loughborough University

> *This dataset contains the height, weight and 4 fingerprint measurements (length, width, area and circumference), collected from 200 participants.*

What best describes a participant?

# The variables

Each participant is associated to 11 variables

- "Participant Number"
- "Gender"
- "Age"
- "Dominant Hand"
- "Height (cm) (average of 3 measurements)"
- "Weight (kg) (average of 3 measurements)"
- "Fingertip Temperature (°C)"
- "Fingerprint Height (mm)"
- "Fingerprint Width (mm)"
- "Fingerprint Area (mm2)"
- "Fingerprint Circumference (mm)"

# Nature of variables

Variables have different natures

- "Participant Number": $\in \mathbb{N}$ (not interesting)
- "Gender": categorical
- "Age": $\in \mathbb{N}$
- "Dominant Hand": categorical
- "Height (cm) (average of 3 measurements)": $\in \mathbb{R}$
- "Weight (kg) (average of 3 measurements)": $\in \mathbb{R}$
- "Fingertip Temperature (°C)": $\in \mathbb{R}$
- "Fingerprint Height (mm)": $\in \mathbb{R}$
- "Fingerprint Width (mm)": $\in \mathbb{R}$
- "Fingerprint Area (mm2)": $\in \mathbb{R}$
- "Fingerprint Circumference (mm)": $\in \mathbb{R}$

# Setting things up

Each participant is a row in the matrix (an *observation*)

Each variable is a column

So we have an $200 \times 10$ matrix (we discard the “Participant number” column)

We want to find what carries the most information

For this, we are going to project the information in a new basis in which the first “dimension” will carry most variance, the second dimension will carry a little less, etc.

In order to do so, we need to learn how to change bases

# Principal component analysis (PCA)

- A crash course on probability
- A running example: fingerprints
- **Change of basis**
- Back to PCA
- A 2D example to begin: hockey players
- Back to fingerprints

In the following slide,

$$[\boldsymbol{x}]_{\mathcal{B}}$$

denotes the coordinates of $\boldsymbol{x}$ in the basis $\mathcal{B}$

The aim of a change of basis is to express vectors in another coordinate system (another basis)

We do so by finding a matrix allowing to move from one basis to another

# Change of basis

## Definition 6 (Change of basis matrix)

$\mathcal{B} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ and $\mathcal{C} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ bases of vector space $V$
The **change of basis matrix** $P_{\mathcal{C}\leftarrow\mathcal{B}} \in \mathcal{M}_n$,

$$P_{\mathcal{C}\leftarrow\mathcal{B}} = [[\boldsymbol{u}_1]_{\mathcal{C}} \cdots [\boldsymbol{u}_n]_{\mathcal{C}}]$$

has columns the coordinate vectors $[\boldsymbol{u}_1]_{\mathcal{C}}, \ldots, [\boldsymbol{u}_n]_{\mathcal{C}}$ of vectors in $\mathcal{B}$ with respect to $\mathcal{C}$

## Theorem 7

*$\mathcal{B} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ and $\mathcal{C} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ bases of vector space $V$ and $P_{\mathcal{C}\leftarrow\mathcal{B}}$ a change of basis matrix from $\mathcal{B}$ to $\mathcal{C}$*

1. $\forall \boldsymbol{x} \in V$, $P_{\mathcal{C}\leftarrow\mathcal{B}}[\boldsymbol{x}]_{\mathcal{B}} = [\boldsymbol{x}]_{\mathcal{C}}$
2. $P_{\mathcal{C}\leftarrow\mathcal{B}}$ *s.t.* $\forall \boldsymbol{x} \in V$, $P_{\mathcal{C}\leftarrow\mathcal{B}}[\boldsymbol{x}]_{\mathcal{B}} = [\boldsymbol{x}]_{\mathcal{C}}$ *is* ***unique***
3. $P_{\mathcal{C}\leftarrow\mathcal{B}}$ *invertible and* $P^{-1}_{\mathcal{C}\leftarrow\mathcal{B}} = P_{\mathcal{B}\leftarrow\mathcal{C}}$

# Row-reduction method for changing bases

**Theorem 8**

*$\mathcal{B} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ and $\mathcal{C} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ bases of vector space $V$. Let $\mathcal{E}$ be any basis for $V$,*

$$B = [[\boldsymbol{u}_1]_{\mathcal{E}}, \ldots, [\boldsymbol{u}_n]_{\mathcal{E}}] \ \textit{and}\ C = [[\boldsymbol{v}_1]_{\mathcal{E}}, \ldots, [\boldsymbol{v}_n]_{\mathcal{E}}]$$

*and let $[C|B]$ be the augmented matrix constructed using $C$ and $B$. Then*

$$RREF\,([C|B]) = [\mathbb{I}|P_{\mathcal{C}\leftarrow\mathcal{B}}]$$

If working in $\mathbb{R}^n$, this is quite useful with $\mathcal{E}$ the standard basis of $\mathbb{R}^n$ (it does not matter if $\mathcal{B} = \mathcal{E}$)

So the question now becomes

*How do we find what new basis to look at our data in?*

(Changing the basis does not change the data, just the view you have of it)

(Think of what happens when you do a headstand.. your up becomes down, your right and left switch, but the world does not change, just your view of it)

(Changes of bases are *fundamental* operations in Science)

# Principal component analysis (PCA)

- A crash course on probability
- A running example: fingerprints
- Change of basis
- **Back to PCA**
- A 2D example to begin: hockey players
- Back to fingerprints

# Setting things up

I will use notation (mostly) as in Joliffe's *Principal Component Analysis* (PDF of older version available for free from UofM Libraries)

$\boldsymbol{x} = (x_1, \ldots, x_p)$ vector of $p$ random variables

We seek a linear function $\boldsymbol{\alpha}_1^T \boldsymbol{x}$ with maximum variance, where $\boldsymbol{\alpha}_1 = (\alpha_{11}, \ldots, \alpha_{1p})$, i.e.,

$$\boldsymbol{\alpha}_1^T \boldsymbol{x} = \sum_{j=1}^{p} \alpha_{1j} x_j$$

Then we seek a linear function $\boldsymbol{\alpha}_2^T \boldsymbol{x}$ with maximum variance, uncorrelated to $\boldsymbol{\alpha}_1^T \boldsymbol{x}$

And we continue...

At *k*th stage, we find a linear function $\boldsymbol{\alpha}_k^T \boldsymbol{x}$ with maximum variance, uncorrelated to $\boldsymbol{\alpha}_1^T \boldsymbol{x}, \ldots, \boldsymbol{\alpha}_{k-1}^T \boldsymbol{x}$

$\boldsymbol{\alpha}_i^T \boldsymbol{x}$ is the *i*th **principal component** (PC)

# Case of known covariance matrix

Suppose we know $\Sigma$, covariance matrix of $\boldsymbol{x}$ (i.e., typically: we know $\boldsymbol{x}$)

Then the $k$th PC is

$$z_k = \boldsymbol{\alpha}_k^T \boldsymbol{x}$$

where $\boldsymbol{\alpha}_k$ is an eigenvector of $\Sigma$ corresponding to the $k$th largest eigenvalue $\lambda_k$

If, additionally, $\|\boldsymbol{\alpha}_k\| = \boldsymbol{\alpha}_k^T \boldsymbol{\alpha} = 1$, then $\lambda_k = \text{Var } z_k$

# Why is that?

Let us start with

$$\boldsymbol{\alpha}_1^T \boldsymbol{x}$$

We want maximum variance, where $\boldsymbol{\alpha}_1 = (\alpha_{11}, \ldots, \alpha_{1p})$, i.e.,

$$\boldsymbol{\alpha}_1^T \boldsymbol{x} = \sum_{j=1}^{p} \alpha_{1j} x_j$$

with the constraint that $\|\boldsymbol{\alpha}_1\| = 1$

We have

$$\text{Var } \boldsymbol{\alpha}_1^T \boldsymbol{x} = \boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1$$

# Objective

We want to maximise Var $\boldsymbol{\alpha}_1^T \boldsymbol{x}$, i.e.,

$$\boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1$$

under the constraint that $\|\boldsymbol{\alpha}_1\| = 1$

$\Longrightarrow$ use **Lagrange multipliers**

# Maximisation using Lagrange multipliers

(A.k.a. super-brief intro to multivariable calculus)

We want the max of $f(x_1, \ldots, x_n)$ under the constraint $g(x_1, \ldots, x_n) = k$

1. Solve

$$\nabla f(x_1, \ldots, x_n) = \lambda \nabla g(x_1, \ldots, x_n)$$
$$g(x_1, \ldots, x_n) = k$$

   where $\nabla = (\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n})$ is the **gradient operator**
2. Plug all solutions into $f(x_1, \ldots, x_n)$ and find maximum values (provided values exist and $\nabla g \neq \mathbf{0}$ there)

$\lambda$ is the **Lagrange multiplier**

# The gradient

(Continuing our super-brief intro to multivariable calculus)

$f : \mathbb{R}^n \to \mathbb{R}$ function of several variables, $\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right)$ the gradient operator

Then

$$\nabla f = \left(\frac{\partial}{\partial x_1} f, \ldots, \frac{\partial}{\partial x_n} f\right)$$

So $\nabla f$ is a *vector-valued* function, $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$; also written as

$$\nabla f = f_{x_1}(x_1, \ldots, x_n)\boldsymbol{e}_1 + \cdots f_{x_n}(x_1, \ldots, x_n)\boldsymbol{e}_n$$

where $f_{x_i}$ is the partial derivative of $f$ with respect to $x_i$ and $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$ is the standard basis of $\mathbb{R}^n$

# Bear with me..

(You may experience a brief period of discomfort)

$\boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1$ and $\|\boldsymbol{\alpha}_1\|^2 = \boldsymbol{\alpha}_1^T \boldsymbol{\alpha_1}$ are functions of $\boldsymbol{\alpha}_1 = (\alpha_{11}, \ldots, \alpha_{1p})$

In the notation of the previous slide, we want the max of

$$f(\alpha_{11}, \ldots, \alpha_{1p}) := \boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1$$

under the constraint that

$$g(\alpha_{11}, \ldots, \alpha_{1p}) := \boldsymbol{\alpha}_1^T \boldsymbol{\alpha_1} = 1$$

and with gradient operator

$$\nabla = \left( \frac{\partial}{\partial \alpha_{11}}, \ldots, \frac{\partial}{\partial \alpha_{1p}} \right)$$

# Effect of $\nabla$ on $g$

$g$ is easiest to see:

$$\begin{aligned}
\nabla g(\alpha_{11}, \ldots, \alpha_{1p}) &= \left( \frac{\partial}{\partial \alpha_{11}}, \ldots, \frac{\partial}{\partial \alpha_{1p}} \right) (\alpha_{11}, \ldots, \alpha_{1p}) \begin{pmatrix} \alpha_{11} \\ \vdots \\ \alpha_{1p} \end{pmatrix} \\
&= \left( \frac{\partial}{\partial \alpha_{11}}, \ldots, \frac{\partial}{\partial \alpha_{1p}} \right) \left( \alpha_{11}^2 + \cdots + \alpha_{1p}^2 \right) \\
&= (2\alpha_{11}, \ldots, 2\alpha_{1p}) \\
&= 2\boldsymbol{\alpha}_1
\end{aligned}$$

(And that's a general result: $\nabla \|\boldsymbol{x}\|_2^2 = 2\boldsymbol{x}$ with $\|\cdot\|_2$ the Euclidean norm)

# Effect of $\nabla$ on $f$

Expand (write $\Sigma = [s_{ij}]$ and do not exploit symmetry)

$$\boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1 = (\alpha_{11}, \ldots, \alpha_{1p}) \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1p} \\ s_{21} & s_{22} & \cdots & s_{2p} \\ \vdots & \vdots & & \vdots \\ s_{p1} & s_{p2} & & s_{pp} \end{pmatrix} \begin{pmatrix} \alpha_{11} \\ \alpha_{12} \\ \vdots \\ \alpha_{1p} \end{pmatrix}$$

$$= (\alpha_{11}, \ldots, \alpha_{1p}) \begin{pmatrix} s_{11}\alpha_{11} + s_{12}\alpha_{12} + \cdots + s_{1p}\alpha_{1p} \\ s_{21}\alpha_{11} + s_{22}\alpha_{12} + \cdots + s_{2p}\alpha_{1p} \\ \vdots \\ s_{p1}\alpha_{11} + s_{p2}\alpha_{12} + \cdots + s_{pp}\alpha_{1p} \end{pmatrix}$$

$$= (s_{11}\alpha_{11} + s_{12}\alpha_{12} + \cdots + s_{1p}\alpha_{1p})\alpha_{11}$$
$$+ (s_{21}\alpha_{11} + s_{22}\alpha_{12} + \cdots + s_{2p}\alpha_{1p})\alpha_{12}$$
$$\vdots$$
$$+ (s_{p1}\alpha_{11} + s_{p2}\alpha_{12} + \cdots + s_{pp}\alpha_{1p})\alpha_{1p}$$

We have

$$\begin{aligned}\boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1 = &(s_{11}\alpha_{11} + s_{12}\alpha_{12} + \cdots + s_{1p}\alpha_{1p})\alpha_{11} \\ &+ (s_{21}\alpha_{11} + s_{22}\alpha_{12} + \cdots + s_{2p}\alpha_{1p})\alpha_{12} \\ &\vdots \\ &+ (s_{p1}\alpha_{11} + s_{p2}\alpha_{12} + \cdots + s_{pp}\alpha_{1p})\alpha_{1p}\end{aligned}$$

$$\begin{aligned}\implies \frac{\partial}{\partial \alpha_{11}} \boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1 &= (s_{11}\alpha_{11} + s_{12}\alpha_{12} + \cdots + s_{1p}\alpha_{1p}) + s_{11}\alpha_{11} \\ &\quad + s_{21}\alpha_{12} + \cdots + s_{p1}\alpha_{1p} \\ &= s_{11}\alpha_{11} + s_{12}\alpha_{12} + \cdots + s_{1p}\alpha_{1p} \\ &\quad + s_{11}\alpha_{11} + s_{21}\alpha_{12} + \cdots + s_{p1}\alpha_{1p} \\ &= 2(s_{11}\alpha_{11} + s_{12}\alpha_{12} + \cdots + s_{1p}\alpha_{1p})\end{aligned}$$

(last equality stems from symmetry of $\Sigma$)

In general, for $i = 1, \ldots, p$,

$$\begin{aligned}\frac{\partial}{\partial \alpha_{1i}} \boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1 &= s_{i1}\alpha_{11} + s_{i2}\alpha_{12} + \cdots + s_{ip}\alpha_{1p} \\ &\quad + s_{i1}\alpha_{11} + s_{2i}\alpha_{12} + \cdots + s_{pi}\alpha_{1p} \\ &= 2(s_{i1}\alpha_{11} + s_{i2}\alpha_{12} + \cdots + s_{ip}\alpha_{1p})\end{aligned}$$

(because of symmetry of $\Sigma$)

As a consequence,

$$\nabla \boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1 = 2\Sigma \boldsymbol{\alpha}_1$$

So solving

$$\nabla f(x_1, \ldots, x_n) = \lambda \nabla g(x_1, \ldots, x_n)$$

means solving

$$2\Sigma\boldsymbol{\alpha}_1 = \lambda 2\boldsymbol{\alpha}_1$$

i.e.,

$$\Sigma\boldsymbol{\alpha}_1 = \lambda\boldsymbol{\alpha}_1$$

$\implies$ $(\lambda, \boldsymbol{\alpha}_1)$ eigenpair of $\Sigma$, with $\boldsymbol{\alpha}_1$ having unit length

# Picking the right eigenvalue

$(\lambda, \boldsymbol{\alpha}_1)$ eigenpair of $\Sigma$, with $\boldsymbol{\alpha}_1$ having unit length

But which $\lambda$ to choose?

Recall that we want Var $\boldsymbol{\alpha}_1^T \boldsymbol{x} = \boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1$ maximal

We have

$$\text{Var } \boldsymbol{\alpha}_1^T \boldsymbol{x} = \boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_1^T (\Sigma \boldsymbol{\alpha}_1) = \boldsymbol{\alpha}_1^T (\lambda \boldsymbol{\alpha}_1) = \lambda (\boldsymbol{\alpha}_1^T \boldsymbol{\alpha}_1) = \lambda$$

$\implies$ we pick $\lambda = \lambda_1$, the largest eigenvalue (covariance matrix symmetric so eigenvalues real)

# What we have this far..

The first principal component is $\boldsymbol{\alpha}_1^T \boldsymbol{x}$ and has variance $\lambda_1$, where $\lambda_1$ the largest eigenvalue of $\Sigma$ and $\boldsymbol{\alpha}_1$ an associated eigenvector with $\|\boldsymbol{\alpha}_1\| = 1$

We want the second principal component to be *uncorrelated* with $\boldsymbol{\alpha}_1^T \boldsymbol{x}$ and to have maximum variance Var $\boldsymbol{\alpha}_2^T \boldsymbol{x} = \boldsymbol{\alpha}_2^T \Sigma \boldsymbol{\alpha}_2$, under the constraint that $\|\boldsymbol{\alpha}_2\| = 1$

$\boldsymbol{\alpha}_2^T \boldsymbol{x}$ uncorrelated to $\boldsymbol{\alpha}_1^T \boldsymbol{x}$ if $\text{cov}(\boldsymbol{\alpha}_1^T \boldsymbol{x}, \boldsymbol{\alpha}_2^T \boldsymbol{x}) = 0$

We have

$$\begin{aligned}
\text{cov}(\boldsymbol{\alpha}_1^T \boldsymbol{x}, \boldsymbol{\alpha}_2^T \boldsymbol{x}) &= \boldsymbol{\alpha}_1^T \Sigma \boldsymbol{\alpha}_2 \\
&= \boldsymbol{\alpha}_2^T \Sigma^T \boldsymbol{\alpha}_1 \\
&= \boldsymbol{\alpha}_2^T \Sigma \boldsymbol{\alpha}_1 \quad [\Sigma \text{ symmetric}] \\
&= \boldsymbol{\alpha}_2^T (\lambda_1 \boldsymbol{\alpha}_1) \\
&= \lambda \boldsymbol{\alpha}_2^T \boldsymbol{\alpha}_1
\end{aligned}$$

So $\boldsymbol{\alpha}_2^T \boldsymbol{x}$ uncorrelated to $\boldsymbol{\alpha}_1^T \boldsymbol{x}$ if $\boldsymbol{\alpha}_1 \perp \boldsymbol{\alpha}_2$

This is beginning to sound a lot like Gram-Schmidt, no?

# In short

Take whatever covariance matrix is available to you (known $\Sigma$ or sample $S_X$) – assume sample from now on for simplicity

For $i = 1, \ldots, p$, the $i$th principal component is

$$z_i = \boldsymbol{v}_i^T \boldsymbol{x}$$

where $\boldsymbol{v}_i$ eigenvector of $S_X$ associated to the $i$th largest eigenvalue $\lambda_i$

If $\boldsymbol{v}_i$ is normalised, then $\lambda_i = \text{Var } z_k$

# Covariance matrix

$\Sigma$ the covariance matrix of the random variable, $S_X$ the sample covariance matrix

$X \in \mathcal{M}_{mp}$ the data, then the (sample) covariance matrix $S_X$ takes the form

$$S_X = \frac{1}{n-1} X^T X$$

where the data is centred!

Sometimes you will see $S_X = 1/(n-1)XX^T$. This is for matrices with observations in columns and variables in rows. Just remember that you want the covariance matrix to have size the number of variables, not observations, this will give you the order in which to take the product

# Principal component analysis (PCA)

- A crash course on probability
- A running example: fingerprints
- Change of basis
- Back to PCA
- **A 2D example to begin: hockey players**
- Back to fingerprints

# A 2D example

See a dataset on this page for a dataset of height and weight of some hockey players

```
data = read.csv("https://figshare.com/ndownloader/files/5303173")

## Error in file(file, "rt"):  cannot open the connection to
'https://figshare.com/ndownloader/files/5303173'

head(data, n=3)

##
## 1 function (..., list = character(), package = NULL, lib.loc = NULL,
## 2     verbose = getOption("verbose"), envir = .GlobalEnv, overwrite = TRU
## 3 {

dim(data)

## NULL
```

In case you are wondering, this is a database of ice hockey players at IIHF world championships, 2001-2016, assembled by the dataset's author

See some comments here

As usual, it is a good idea to plot this to get a sense of the lay of the land

```
## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'plot':  object of type 'closure'
is not subsettable
```


FIGS/L11-plot-hockey-1-1.pdf

The author of the study is interested in the evolution of weights, so it is likely that the same person will be in the dataset several times

Let us check this: first check will be `FALSE` if the number of unique names does not match the number of rows in the dataset

```
length(unique(data$name)) == dim(data)[1]

## Error in data$name:  object of type 'closure' is not subsettable

length(unique(data$name))

## Error in data$name:  object of type 'closure' is not subsettable
```

Not interested in the evolution of weights, so simplify: if more than one record for someone, take average of recorded weights and heights
To be extra careful, could check as well that there are no major variations on player height (homonymies?)

```
data_simplified = data.frame(name = unique(data$name))

## Error in data$name:  object of type 'closure' is not subsettable

w = c()
h = c()
for (n in data_simplified$name) {
    tmp = data[which(data$name == n),]
    h = c(h, mean(tmp$height))
    w = c(w, mean(tmp$weight))
}

## Error:  object 'data_simplified' not found

data_simplified$weight = w
```

```
data = data_simplified

## Error:  object 'data_simplified' not found

head(data_simplified, n = 6)

## Error in h(simpleError(msg, call)):  error in evaluating the
argument 'x' in selecting a method for function 'head':  object
'data_simplified' not found
```

```
## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'plot':  object of type 'closure'
is not subsettable
```


FIGS/L11-plot-hockey-2-1.pdf

# Centre the data

```
mean(data$weight)

## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'mean':  object of type 'closure'
is not subsettable

mean(data$height)

## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'mean':  object of type 'closure'
is not subsettable

data$weight.c = data$weight-mean(data$weight)

## Error in data$weight:  object of type 'closure' is not subsettable

data$height.c = data$height-mean(data$height)

## Error in data$height:  object of type 'closure' is not subsettable
```


FIGS/L11-plot-hockey-centred-1.pdf

# Covariance

The function `cov` returns the covariance of two samples

Note that the functions deals equally well with data that is not centred as with data that is centred

```
cov(data$height, data$weight)

## Error in data$weight:  object of type 'closure' is not subsettable

cov(data$height.c, data$weight.c)

## Error in data$weight.c:  object of type 'closure' is not
subsettable
```

# Covariance matrix

As we could see from plotting the data, there is a positive linear relationship between the two variables

Let us compute the sample covariance matrix

```
X = as.matrix(data[,c("height.c", "weight.c")])

## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'as.matrix':  object of type
'closure' is not subsettable

S = 1/(dim(X)[1]-1)*t(X) %*% X

## Error:  object 'X' not found

S

## Error:  object 'S' not found
```

# Covariance matrix

The off-diagonal entries do match the computed covariance. Let us check that the variances are indeed a match too.

```
var(X[,1])

## Error:  object 'X' not found

var(X[,2])

## Error:  object 'X' not found
```

Hey, that works. Is math not cool? ;)

# Principal components

Now compute the principal components. We need eigenvalues and eigenvectors

```
ev = eigen(S)

## Error:  object 'S' not found

ev

## Error:  object 'ev' not found
```

(`eigen` returns eigenvalues sorted in decreasing order and normalised eigenvectors)

# First principal component

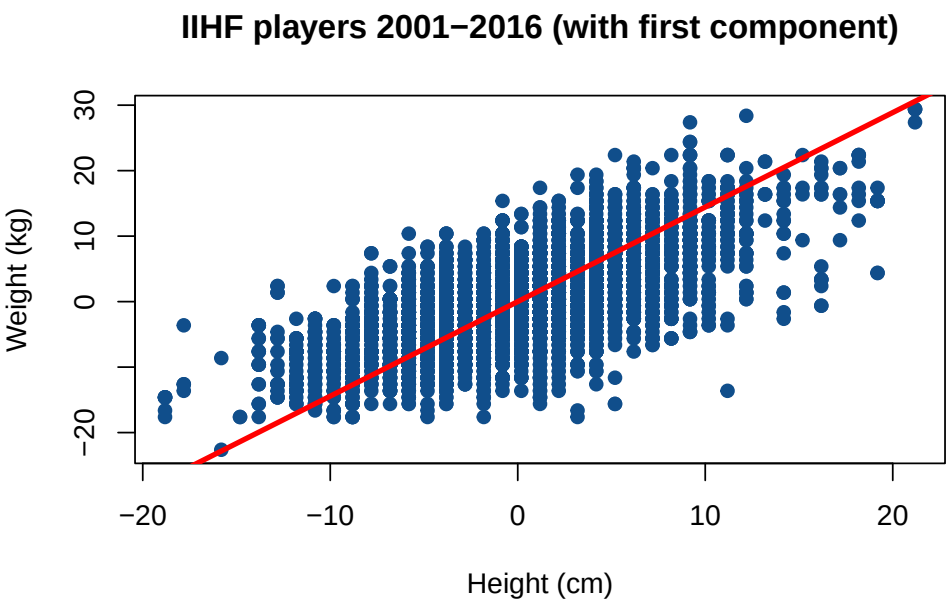
Let us plot this first eigenvector (well, the line carrying this first eigenvector)
To use the function `abline`, we need to give the coefficients of the line in the form of (intercept,slope). Intercept is easy, as the line goes through the origin (by construction and because we have centred the data). The slope is also quite simple..

```
plot(data$height.c, data$weight.c,
    pch = 19, col = "dodgerblue4",
    main = "IIHF players 2001-2016 (with first component)",
    xlab = "Height (cm)", ylab = "Weight (kg)")
```

```
## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'plot':  object of type 'closure'
is not subsettable
```

```
abline(a = 0, b = ev$vectors[2,1]/ev$vectors[1,1],
       col = "red", lwd = 3)
```

```
## Error: object 'ev' not found
```


IIHF players 2001−2016 (with first component)
Weight (kg)
−20
0
10
20
30
−20
−10
0
10
20
Height (cm)

# Rotating the data

Let us rotate the data so that the red line becomes the $x$-axis

To do that, we use a rotation matrix

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

To find the angle $\theta$, recall that $\tan\theta$ is equal to opposite length over adjacent length, i.e.,

$$\tan\theta = \frac{\texttt{ev\$vectors}[2,1]}{\texttt{ev\$vectors}[1,1]}$$

So we just use the $\arctan$ of this

Note that angles are in radians

# Rotating the data

```
theta = atan(ev$vectors[2,1]/ev$vectors[1,1])

## Error:  object 'ev' not found

theta

## Error:  object 'theta' not found

R_theta = matrix(c(cos(theta), -sin(theta),
                   sin(theta), cos(theta)),
                 nr = 2, byrow = TRUE)

## Error:  object 'theta' not found

R_theta

## Error:  object 'R_theta' not found
```

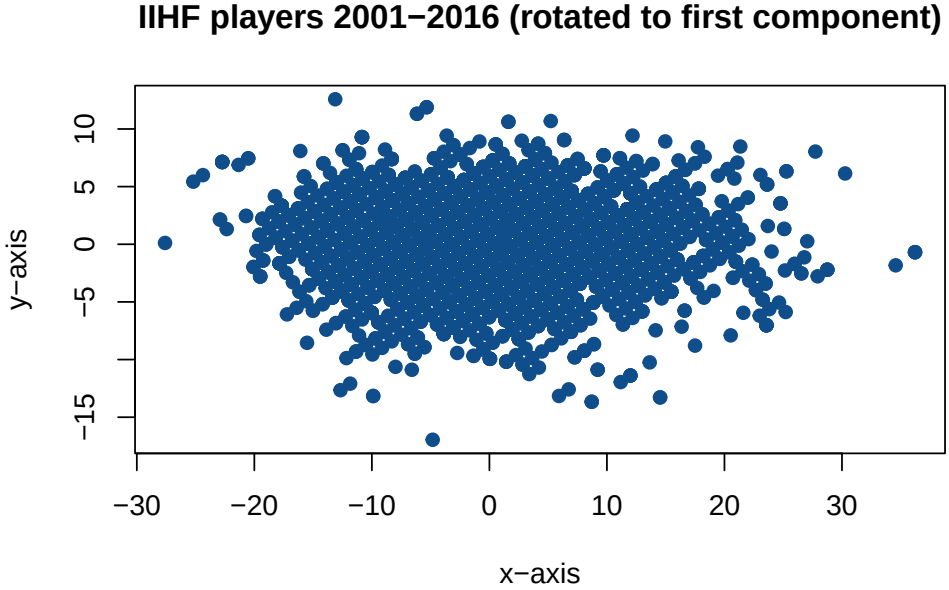
# Rotating the data

And now we rotate the points

(In this case, we think of the points as vectors, of course)

```
tmp_in = matrix(c(data$weight.c, data$height.c),
                nc = 2)

## Error in data$weight.c:  object of type 'closure' is not
subsettable

tmp_out = c()
for (i in 1:dim(tmp_in)[1]) {
    tmp_out = rbind(tmp_out,
                    t(R_theta %*% tmp_in[i,]))
}

## Error:  object 'tmp_in' not found

data$weight.c_r = tmp_out[,1]
```

```
## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'plot':  object of type 'closure'
is not subsettable
```


IIHF players 2001−2016 (rotated to first component)
y−axis
10
5
0
−5
−15
−30
−20
−10
0
10
20
30
x−axis

# Principal components

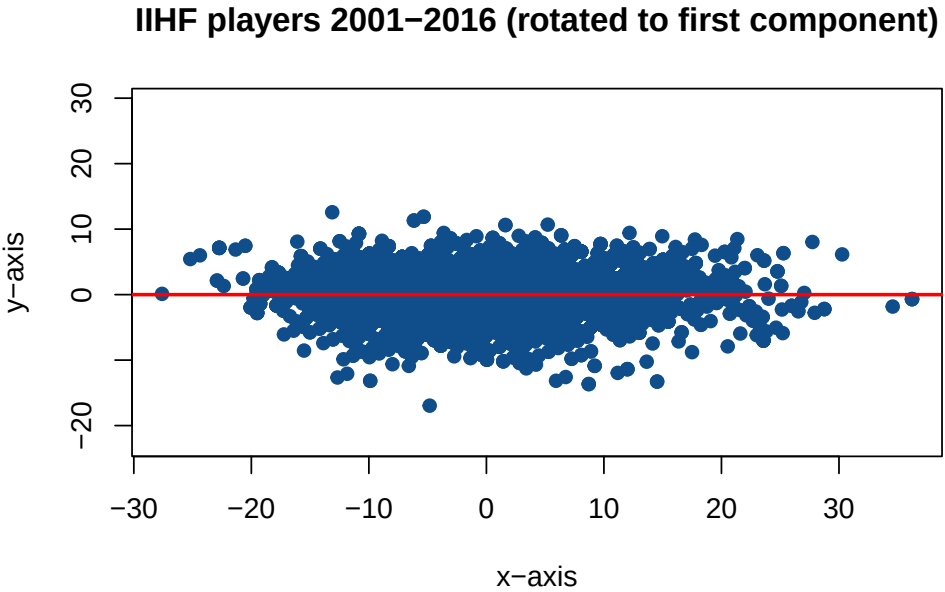
Note that the axes have changed quite a lot, hence the very different aspect
Let us plot with the same range as for the non-rotated data for the y-axis

```
plot(data$height.c_r, data$weight.c_r,
    pch = 19, col = "dodgerblue4",
    xlab = "x-axis", ylab = "y-axis",
    main = "IIHF players 2001-2016 (rotated to first component)",
    ylim = range(data$weight.c))
```

```
## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'plot':  object of type 'closure'
is not subsettable
```

```
abline(h = 0, col = "red", lwd = 2)
```

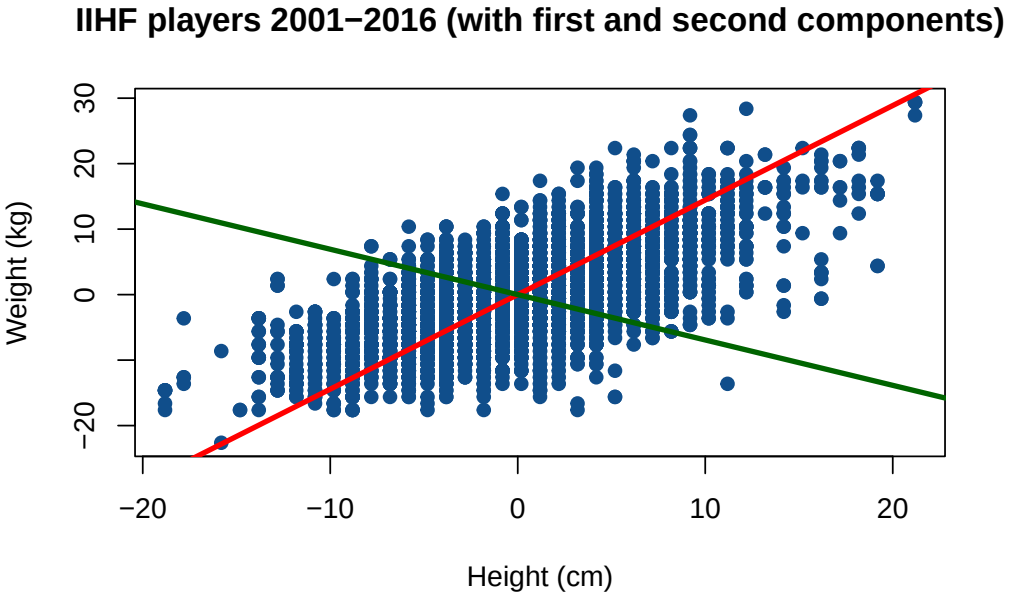
```
## Error in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...):
plot.new has not been called yet
```


IIHF players 2001−2016 (rotated to first component)
y−axis
x−axis
30
20
10
0
−20
−30
−20
−10
0
10
20
30

# First and second principal components

Plot the first and second eigenvectors

```
plot(data$height.c, data$weight.c,
    pch = 19, col = "dodgerblue4",
    main = "IIHF players 2001-2016 (with first and second components)",
    xlab = "Height (cm)", ylab = "Weight (kg)")

## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'plot':  object of type 'closure'
is not subsettable

abline(a = 0, b = ev$vectors[2,1]/ev$vectors[1,1],
       col = "red", lwd = 3)

## Error:  object 'ev' not found

abline(a = 0, b = ev$vectors[2,2]/ev$vectors[1,2],
       col = "darkgreen", lwd = 3)
```


IIHF players 2001−2016 (with first and second components)
Weight (kg)
30
20
10
0
−20
−20
−10
0
10
20
Height (cm)

# Proper change of basis

Let us change the basis so that, in the new basis, the first component is the $x$-axis and the second component is the $y$-axis

We want to use Theorem 8

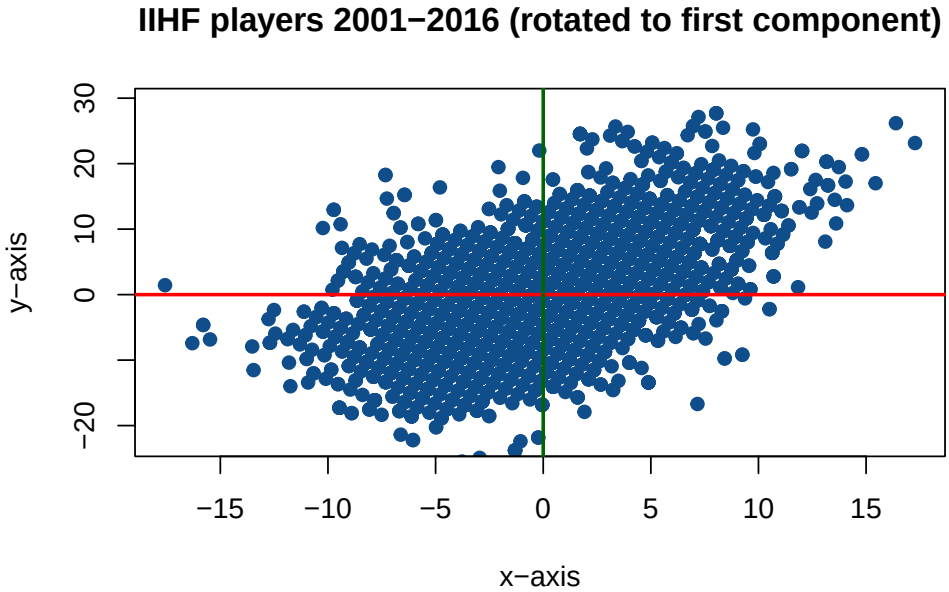
We need the coordinates of the new basis in the canonical basis of $\mathbb{R}^2$

Since both axes go through the origin, we can just use $y = ax$, with $a$ the slope of the lines and, say, $x = 1$, i.e., $(x, y) = (1, a)$

We then normalise the resulting vectors

# Proper change of basis

```
red_line = c(1, ev$vectors[2,1]/ev$vectors[1,1])

## Error:  object 'ev' not found

red_line = red_line/sqrt(sum(red_line^2))

## Error:  object 'red_line' not found

green_line = c(1, ev$vectors[2,2]/ev$vectors[1,2])

## Error:  object 'ev' not found

green_line = green_line/sqrt(sum(green_line^2))

## Error:  object 'green_line' not found

augmented_M = cbind(red_line,green_line, diag(2))

## Error:  object 'red_line' not found
```


IIHF players 2001−2016 (rotated to first component)
y−axis
30
20
10
0
−20
−15
−10
−5
0
5
10
15
x−axis

# PCA using built-in functions

Now do things “properly”

```
GS = pracma::gramSchmidt(A = ev$vectors, tol = 1e-10)

## Error:  object 'ev' not found

GS

## Error:  object 'GS' not found
```

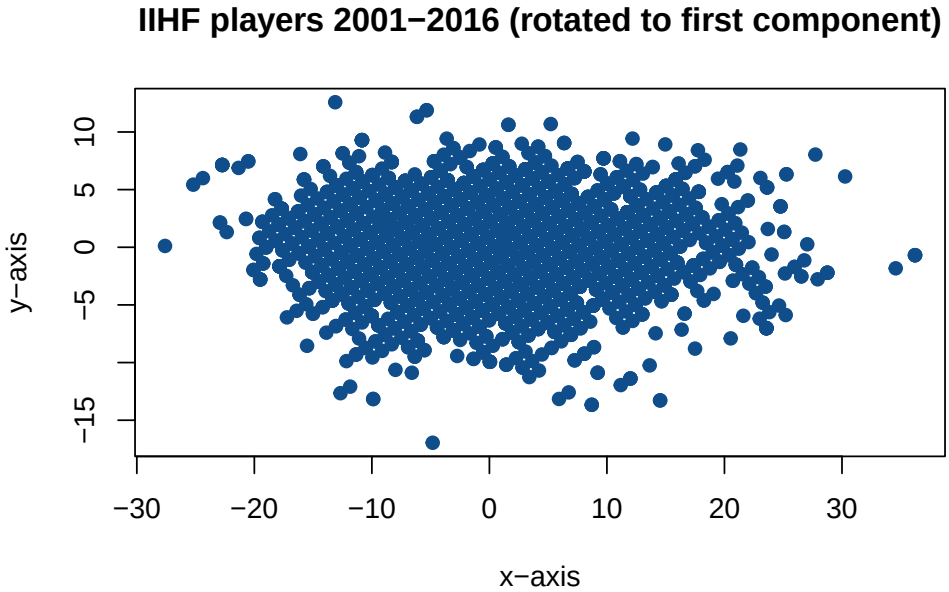
# PCA using built-in functions

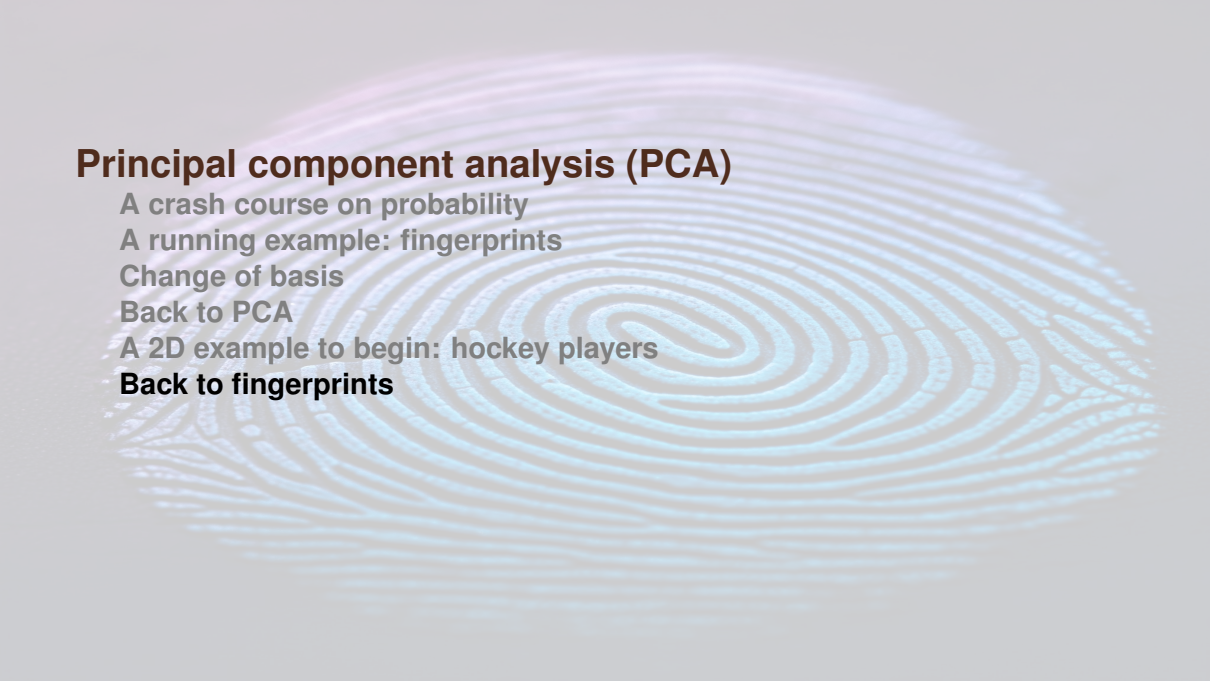
Now recall we saw a theorem that told us how to construct a new basis..

```
A=matrix(c(GS$Q,1,0,0,1), nr = 2)

## Error:  object 'GS' not found

A

## Error:  object 'A' not found

pracma::rref(A)

## Error:  object 'A' not found
```

# PCA using built-in functions

```
P = pracma::rref(A)[,c(3,4)]

## Error:  object 'A' not found
## Error:  object 'P' not found

X.new = X %*% t(P)

## Error:  object 'X' not found
## Error in h(simpleError(msg, call)):  error in evaluating the
argument 'x' in selecting a method for function 'plot':  object
'X.new' not found
```


IIHF players 2001−2016 (rotated to first component)
y−axis
10
5
0
−5
−15
−30
−20
−10
0
10
20
30
x−axis



# Principal component analysis (PCA)

- A crash course on probability
- A running example: fingerprints
- Change of basis
- Back to PCA
- A 2D example to begin: hockey players
- **Back to fingerprints**

We get the data from here

This time, we first download the data, then open the file

The file is an excel table, so we need to use a library for doing that

# Loading the excel fingerprint data

```
download.file(url = "https://repository.lboro.ac.uk/ndownloader/files/140157
              destfile = "../CODE/fingerprint_data.xlsx")

## Error in download.file(url =
"https://repository.lboro.ac.uk/ndownloader/files/14015774", :  cannot
open URL 'https://repository.lboro.ac.uk/ndownloader/files/14015774'

data = openxlsx::read.xlsx("../CODE/fingerprint_data.xlsx")

## Error in read.xlsx.default("../CODE/fingerprint_data.xlsx"):  File
does not exist.

head(data, n=3)

##
## 1 function (..., list = character(), package = NULL, lib.loc = NULL,
## 2     verbose = getOption("verbose"), envir = .GlobalEnv, overwrite = TRU
## 3 {
```

# Some wrangling

Let us rework the names of columns a bit, for convenience. Let us also get rid of a few columns we are not using

```
data = data[,2:dim(data)[2]]

## Error in 2:dim(data)[2]:  argument of length 0

colnames(data) = c("gender", "age", "handedness", "height", "weight",
                   "fing_temp", "fing_height", "fing_width",
                   "fing_area", "fing_circ")

## Error in `colnames<-`(`*tmp*`, value = c("gender", "age",
"handedness", :  attempt to set 'colnames' on an object with less than
two dimensions

head(data, n=3)

##
## 1 function (, list = character(), package = NULL, lib.loc = NULL,
```

# Some wrangling – Centering

Plotting all these variables is complicated, so we forgo this for the time being

Let us centre the data. That there are some `NA` values, so we remove them using the function `complete.cases`, which identifies rows where at least one of the variables is `NA`

(We could also use `na.rm = TRUE` when taking the average to remove these values.)

We make new columns with the prefix `.c`, just to still have the initial data handy if need be.

# Some wrangling – Centering

```
data = data[complete.cases(data),]

## Error in complete.cases(data):  invalid 'type' (closure) of
argument

to_centre = c("age", "height",
              "weight", "fing_temp",
              "fing_height", "fing_width",
              "fing_area", "fing_circ")
for (c in to_centre) {
    new_c = sprintf("%s.c", c)
    data[[new_c]] = data[[c]] - mean(data[[c]], na.rm = TRUE)
}

## Error in data[[c]]:  object of type 'closure' is not subsettable

head(data)
```

# Covariance matrix

```
X = as.matrix(data[, to_centre])

## Error in (function (cond) :  error in evaluating the argument 'x'
in selecting a method for function 'as.matrix':  object of type
'closure' is not subsettable

S = 1/(dim(X)[1]-1)*t(X) %*% X

## Error:  object 'X' not found

S

## Error:  object 'S' not found
```

# Eigenvalues

```
ev = eigen(S)

## Error:  object 'S' not found

ev$values

## Error:  object 'ev' not found
```

Let us add the singular values to `ev`

```
ev$sing_values = sqrt(ev$values)

## Error:  object 'ev' not found
```

# Use built-in functions

```
GS = pracma::gramSchmidt(A = ev$vectors)

## Error:  object 'ev' not found

GS$Q

## Error:  object 'GS' not found

# Just to check that Q is indeed with normalised columns
colSums(GS$Q[,1:dim(GS$Q)[2]]^2)

## Error in h(simpleError(msg, call)):  error in evaluating the
argument 'x' in selecting a method for function 'colSums':  object
'GS' not found

GS$Q[,1] %*% GS$Q[,2]

## Error:  object 'GS' not found
```

# Some wrangling

Now recall we saw a theorem that told us how to construct a new basis..

```
# Make an identity matrix
Id = diag(dim(GS$Q)[1])

## Error in h(simpleError(msg, call)): error in evaluating the
argument 'x' in selecting a method for function 'diag': object 'GS'
not found

# Make the augmented matrix
A = cbind(GS$Q, Id)

## Error: object 'GS' not found

# Compute the RREF and extract the relevant matrix
P = pracma::rref(A)[,(dim(GS$Q)[2]+1):dim(A)[2]]

## Error: object 'A' not found
```

# Use built-in functions

Use the built in function `prcomp` or `PCA` from the `FactoMineR` package

```
# data.pca = prcomp(X, center = TRUE, scale = TRUE)
data.pca = PCA(X, scale.unit = TRUE, graph = FALSE)

## Error:  object 'X' not found

summary(data.pca)

## Error in h(simpleError(msg, call)):  error in evaluating the
argument 'object' in selecting a method for function 'summary':
object 'data.pca' not found
```

# Percentage of variance

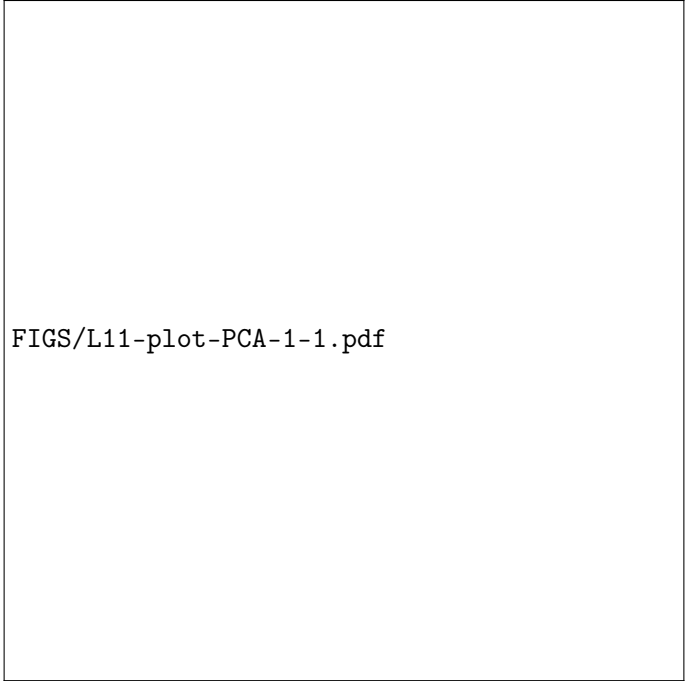
The “proportion of variance” (or “percentage of variance”) information is actually the proportion (and then cumulative proportion) represented by the singular value associated to each principal component

We check this (approximately) by comparing with the singular values we computed

```
ev$sing_values/(sum(ev$sing_values))

## Error:  object 'ev' not found

cumsum(ev$sing_values)/(sum(ev$sing_values))

## Error:  object 'ev' not found
```

# Plot results

```
plot.PCA(data.pca, axes = c(1,2), choix = "ind", habillage = 4)

## Error:  object 'data.pca' not found
```


FIGS/L11-plot-PCA-1-1.pdf