Matrix methods – Principal component analysis (2)

MATH 2740 – Mathematics of Data Science – Lecture 11

Julien Arino

julien.arino@umanitoba.ca

Department of Mathematics @ University of Manitoba

Fall 202X

Outline

Back to PCA

Hockey players

Back to PCA

Hockey players

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

Then the kth PC is

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

where $\boldsymbol{\alpha}_k$ is an eigenvector of Σ corresponding to the kth largest eigenvalue λ_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 $\alpha_1^T \boldsymbol{x}$, i.e.,

$$\alpha_1^T \Sigma \alpha_1$$

under the constraint that $\|\alpha_1\| = 1$

$\Longrightarrow$ 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

$$\begin{aligned} \nabla f(x_1, \ldots, x_n) &= \lambda \nabla g(x_1, \ldots, x_n) \\ g(x_1, \ldots, x_n) &= k \end{aligned}$$

   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)

λ 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 ∇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 ∇ on g

g is easiest to see:

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

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

Effect of ∇ on f

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

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

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 Σ)

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

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

(because of symmetry of Σ)

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 Σ, with $\boldsymbol{\alpha}_1$ having unit length

Picking the right eigenvalue

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

But which λ 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 $\alpha_1^T \boldsymbol{x}$ and has variance λ_1, where λ_1 the largest eigenvalue of Σ and α_1 an associated eigenvector with $\|\alpha_1\| = 1$

We want the second principal component to be *uncorrelated* with $\alpha_1^T \boldsymbol{x}$ and to have maximum variance Var $\alpha_2^T \boldsymbol{x} = \alpha_2^T \Sigma \alpha_2$, under the constraint that $\|\alpha_2\| = 1$

$\alpha_2^T \boldsymbol{x}$ uncorrelated to $\alpha_1^T \boldsymbol{x}$ if $\text{cov}(\alpha_1^T \boldsymbol{x}, \alpha_2^T \boldsymbol{x}) = 0$

We have

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

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 Σ or sample S_X) – assume sample from now on for simplicity

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

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

where $\boldsymbol{v}_i$ eigenvector of S_X associated to the ith largest eigenvalue λ_i

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

Covariance matrix

Σ 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

Back to PCA

Hockey players

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)

## [1] 25.98646

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

## [1] 25.98646
```

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

##          height.c weight.c
## height.c 28.98592 25.98646
## weight.c 25.98646 48.49557
```

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])

## [1] 28.98592

var(X[,2])

## [1] 48.49557
```

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

Principal components

Now compute the principal components. We need eigenvalues and eigenvectors

```
ev = eigen(S)
ev

## eigen() decomposition
## $values
## [1] 66.49777 10.98372
##
## $vectors
##            [,1]       [,2]
## [1,] 0.5694576 -0.8220207
## [2,] 0.8220207  0.5694576
```

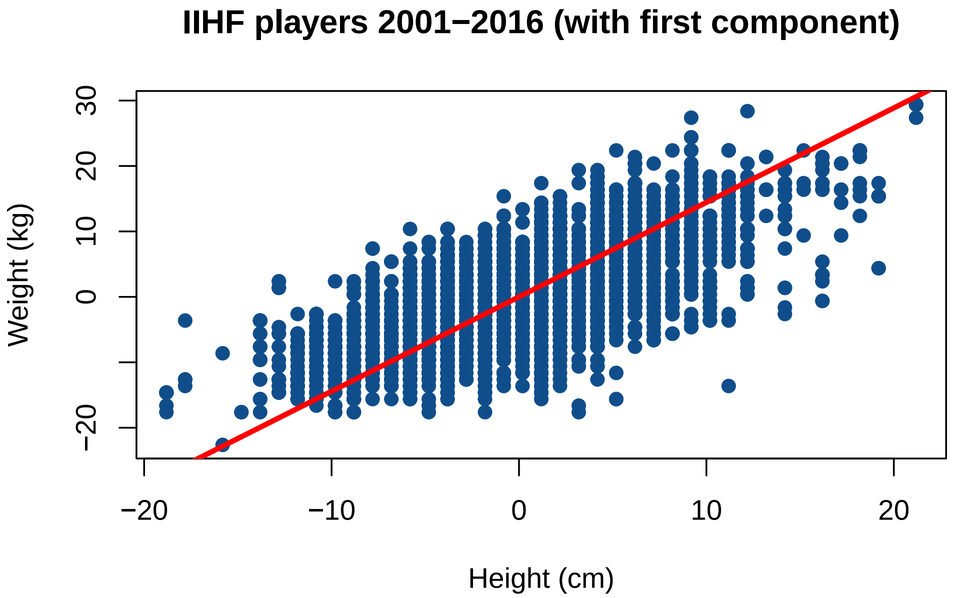
(`eigen` returns eigenvalues sorted in decreasing order and normalised eigenvectors)

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..

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


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

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 θ, 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

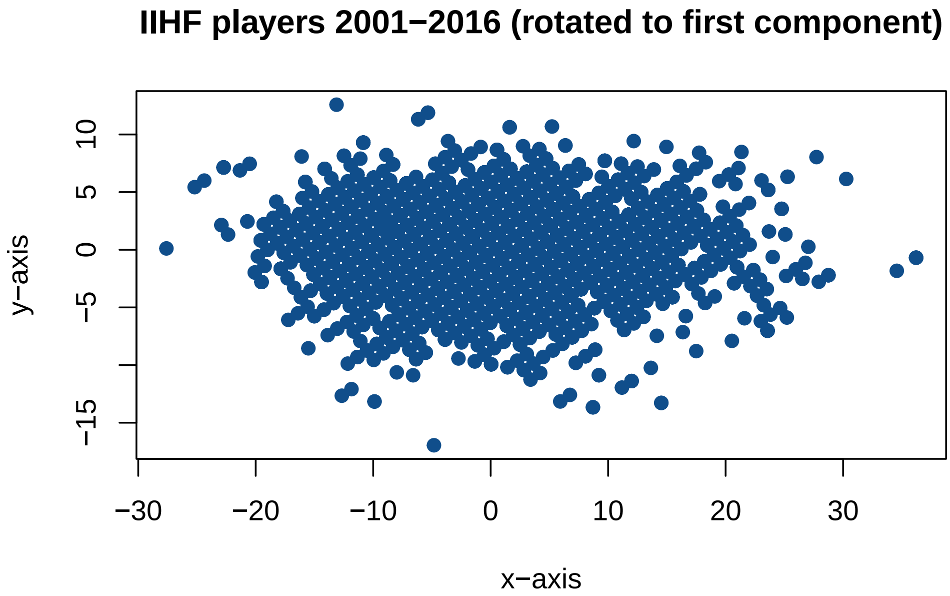
```
theta = atan(ev$vectors[2,1]/ev$vectors[1,1])
theta

## [1] 0.9649505

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

##            [,1]       [,2]
## [1,] 0.5694576 -0.8220207
## [2,] 0.8220207  0.5694576
```

Rotating the data

And now we rotate the points

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

```r
tmp_in = matrix(c(data$weight.c, data$height.c),
                nc = 2)
tmp_out = c()
for (i in 1:dim(tmp_in)[1]) {
    tmp_out = rbind(tmp_out,
                    t(R_theta %*% tmp_in[i,]))
}
data$weight.c_r = tmp_out[,1]
data$height.c_r = tmp_out[,2]
```


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

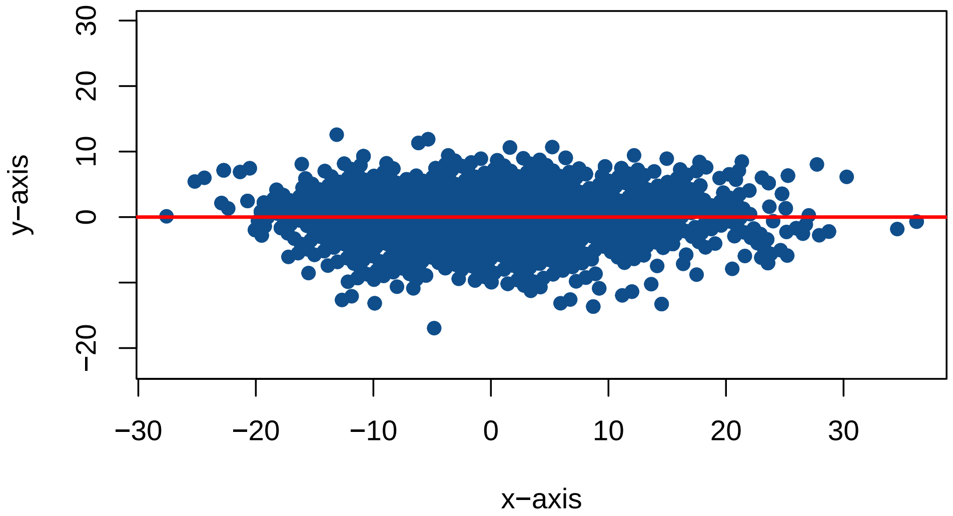
Principal components

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

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

IIHF players 2001–2016 (rotated to first component)

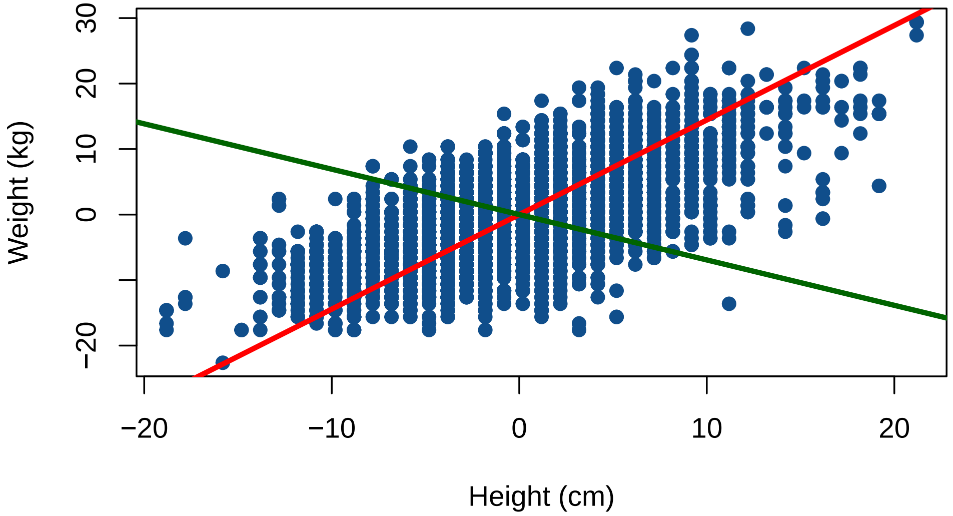


First and second principal components

Plot the first and second eigenvectors

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



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 **??**

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

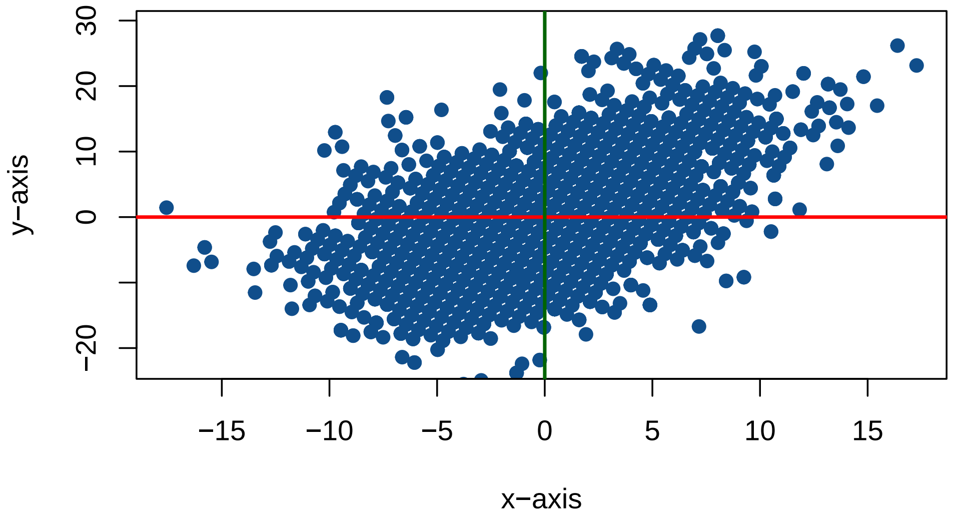
Since both axes go through the origin, we can just use $y = ax$, with a the slope of the lines and, say, $x = 1$, i.e., $(x, y) = (1, a)$

We then normalise the resulting vectors

Proper change of basis

```r
red_line = c(1, ev$vectors[2,1]/ev$vectors[1,1])
red_line = red_line/sqrt(sum(red_line^2))
green_line = c(1, ev$vectors[2,2]/ev$vectors[1,2])
green_line = green_line/sqrt(sum(green_line^2))
augmented_M = cbind(red_line,green_line, diag(2))
P = rref(augmented_M)[,3:4]

tmp_in = matrix(c(data$weight.c, data$height.c), nc = 2)
tmp_out = c()
for (i in 1:dim(tmp_in)[1]) {
    tmp_out = rbind(tmp_out, t(P %*% tmp_in[i,]))
}
data$weight.c_r2 = tmp_out[,1]
data$height.c_r2 = tmp_out[,2]
```

IIHF players 2001–2016 (rotated to first component)



PCA using built-in functions

Now do things “properly”

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

## $Q
##            [,1]       [,2]
## [1,] 0.5694576 -0.8220207
## [2,] 0.8220207  0.5694576
##
## $R
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

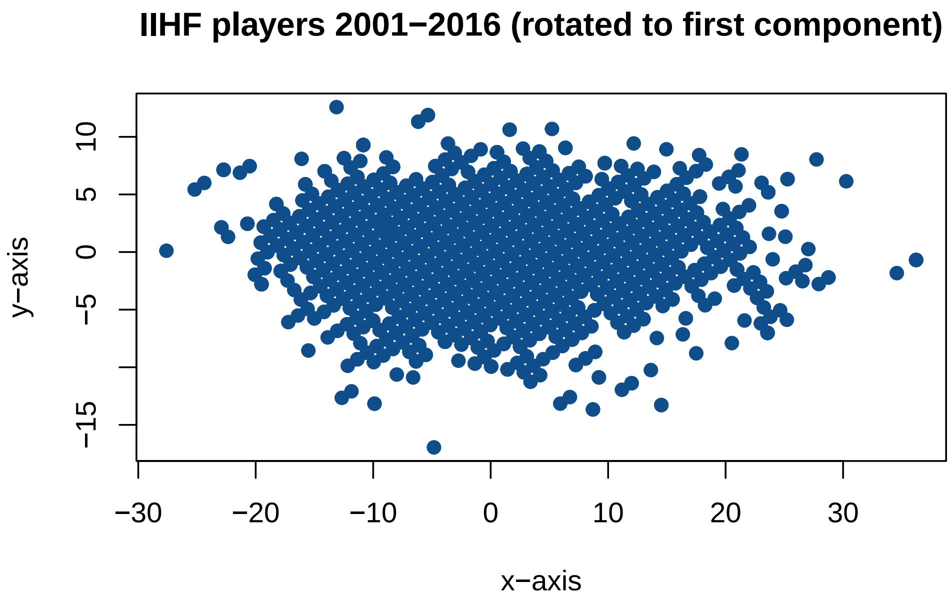
PCA using built-in functions

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)
A

##           [,1]       [,2] [,3] [,4]
## [1,] 0.5694576 -0.8220207    1    0
## [2,] 0.8220207  0.5694576    0    1

pracma::rref(A)

##      [,1] [,2]       [,3]      [,4]
## [1,]    1    0  0.5694576 0.8220207
## [2,]    0    1 -0.8220207 0.5694576
```

PCA using built-in functions

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

##             [,1]      [,2]
## [1,]  0.5694576 0.8220207
## [2,] -0.8220207 0.5694576

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


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