

Lecture 03:

Dimension Reduction and PCA



Markus Hohle

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units



Lecture 1: Course Overview and Introduction to Machine Learning

Lecture 2: Bayesian Methods in Machine Learning

classic ML tools & algorithms

Lecture 3: Dimensionality Reduction: Principal Component Analysis

Lecture 4: Linear and Non-linear Regression and Classification

Lecture 5: Unsupervised Learning: Clustering and Gaussian Mixture Models

Lecture 6: Adaptive Learning and Gradient Descent Optimization Algorithms

Lecture 7: Introduction to Artificial Neural Networks - The Perceptron

ANNs/AI/Deep Learning

Lecture 8: Introduction to Artificial Neural Networks - Building Multiple Dense Layers

Lecture 9: Convolutional Neural Networks (CNNs) - Part I

Lecture 10: CNNs - Part II

Lecture 11: Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs)

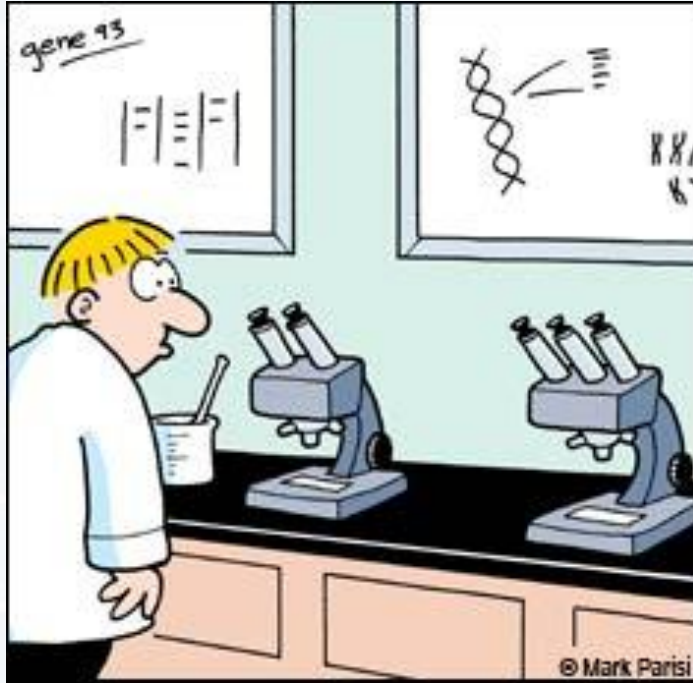
Lecture 12: Combining LSTMs and CNNs

Lecture 13: Running Models on GPUs and Parallel Processing

Lecture 14: Project Presentations

Lecture 15: Transformer

Lecture 16: GNN

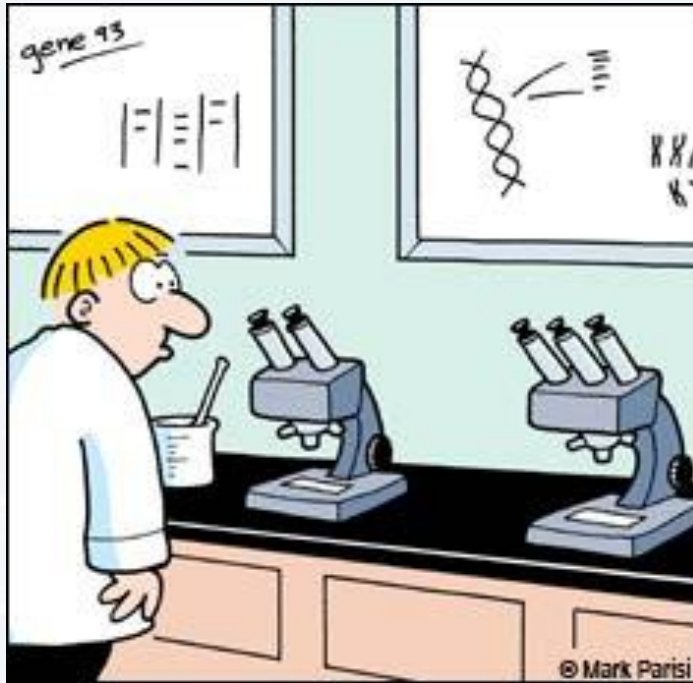


Outline

Variance and Covariance

Correlation

Principal Component Analysis (PCA)



Outline

Variance and Covariance

Correlation

Principal Component Analysis (PCA)



short refresher

the mean μ

(barycenter)

the variance σ^2

(natural scatter)

discrete (= countable)

$$\mu = E(x) = \sum_i x_i p(x_i)$$

$$\sigma^2 = \text{var}(x) = \sum_i (x_i - \mu)^2 p(x_i)$$

continuous

$$\mu = E(x) = \int x p(x) dx$$

$$\sigma^2 = \text{var}(x) = \int (x - \mu)^2 p(x) dx$$



short refresher

Important quantities you should know:

mean

$$\mu = E(x) = \int x p(x) dx$$

variance

$$\sigma^2 = \text{var}(x) = \int (x - \mu)^2 p(x) dx$$

$$\sigma^2 = E(x^2) - E(x)^2$$

$$\sigma_{tot}^2 = \sigma_1^2 + \sigma_2^2 + 2 \text{cov}(x_1, x_2)$$

covariance

$$\text{cov}(x_1, x_2) = E(x_1 x_2) - E(x_1)E(x_2)$$

**correlation
coefficient**

$$\rho(x_1, x_2) = \frac{\text{cov}(x_1, x_2)}{\sqrt{\sigma_1^2 \sigma_2^2}}$$



$$\text{cov}(x_1, x_2) = E(x_1 x_2) - E(x_1)E(x_2)$$

plotting two sets of random number: x_1 and x_2

```
x1 = np.random.normal(0, 2, (1000,))
```

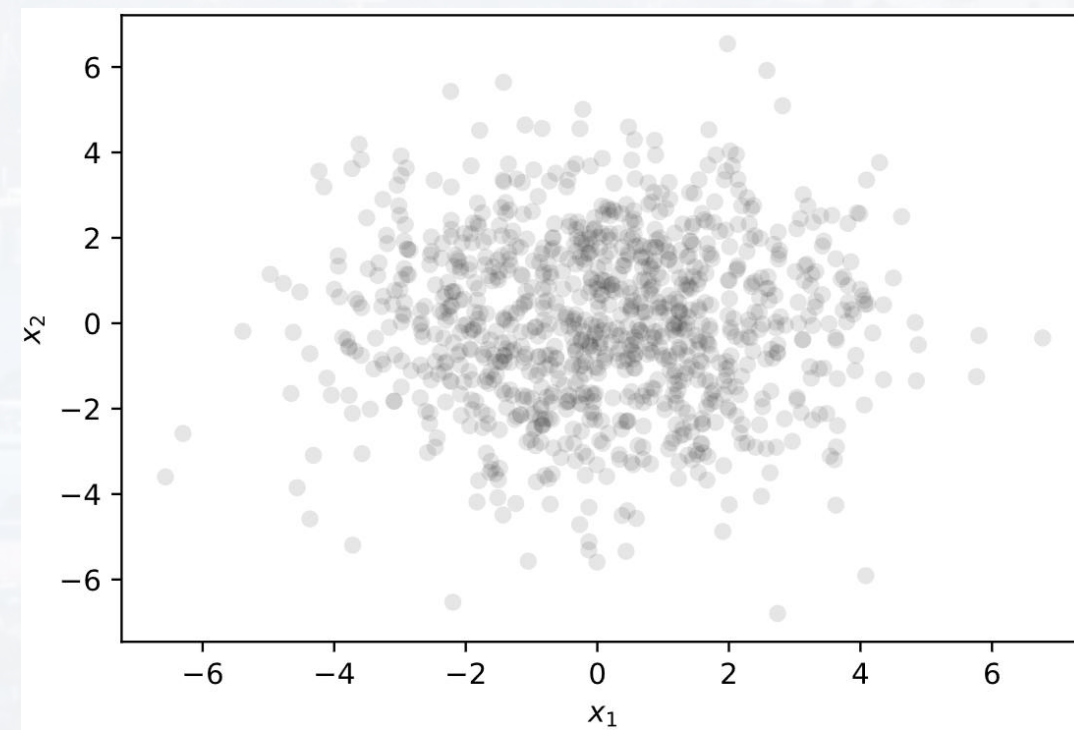
```
x2 = np.random.normal(0, 2, (1000,))
```

```
plt.scatter(x1, x2, color = 'k', alpha = 0.1, edgecolor = 'none')
```

```
plt.xlabel('$x_1$')
```

```
plt.ylabel('$x_2$')
```

x_1 and x_2 are unrelated and mutually **independent**
→ featureless data cloud





$$\text{cov}(x_1, x_2) = E(x_1 x_2) - E(x_1)E(x_2)$$

plotting two sets of random number: x_1 and x_2

```
x1 = np.random.normal(0, 2, (1000,))
```

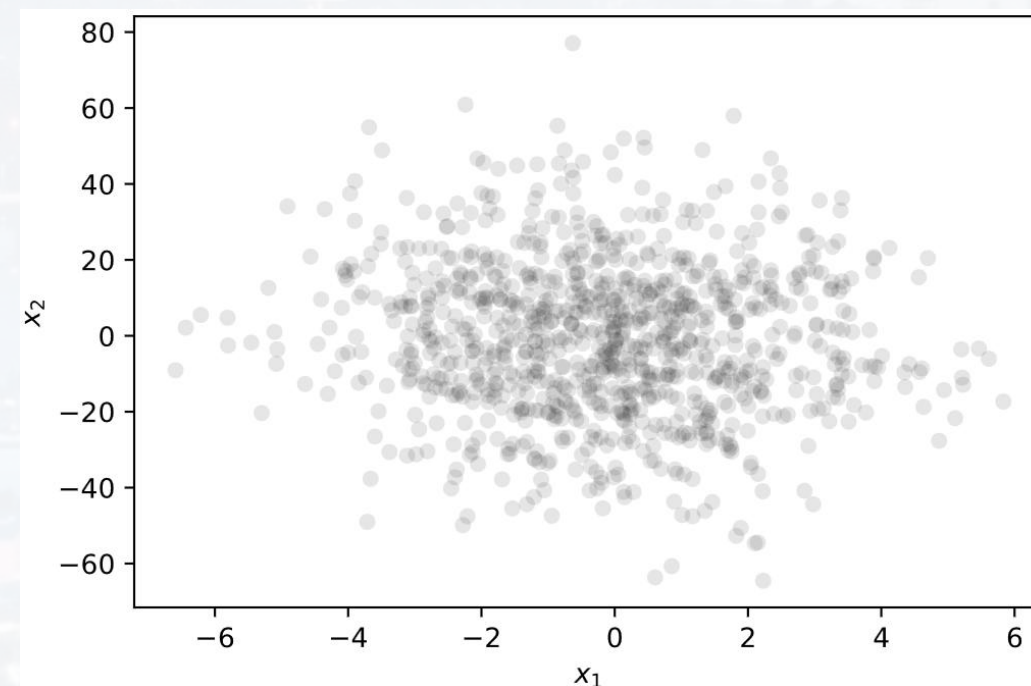
```
x2 = np.random.normal(0, 20, (1000,))
```

```
plt.scatter(x1, x2, color = 'k', alpha = 0.1, edgecolor = 'none')
```

```
plt.xlabel('$x_1$')
```

```
plt.ylabel('$x_2$')
```

x_1 and x_2 are unrelated and mutually **independent**
→ featureless data cloud





$$\text{cov}(x_1, x_2) = E(x_1 x_2) - E(x_1)E(x_2)$$

plotting two sets of random number: x_1 and x_2

```
x1 = np.random.normal(0, 2, (1000,))
```

```
x2 = x1**2 + x1
```

```
#x2 = 4*x1
```

```
plt.scatter(x1, x2, color = 'k', alpha = 0.1, edgecolor = 'none')
```

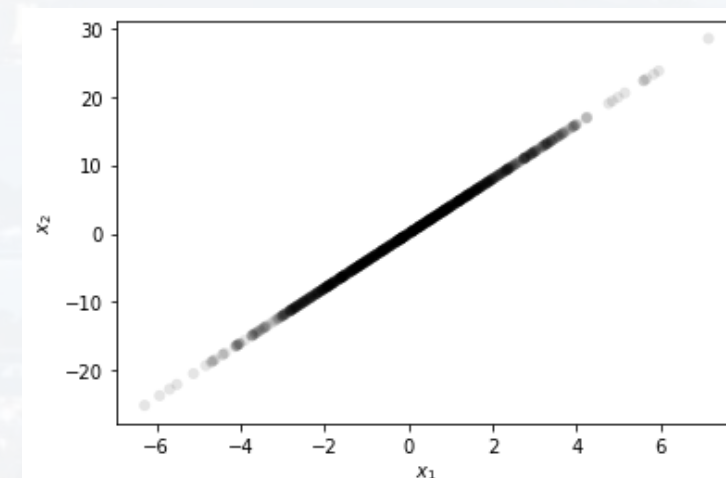
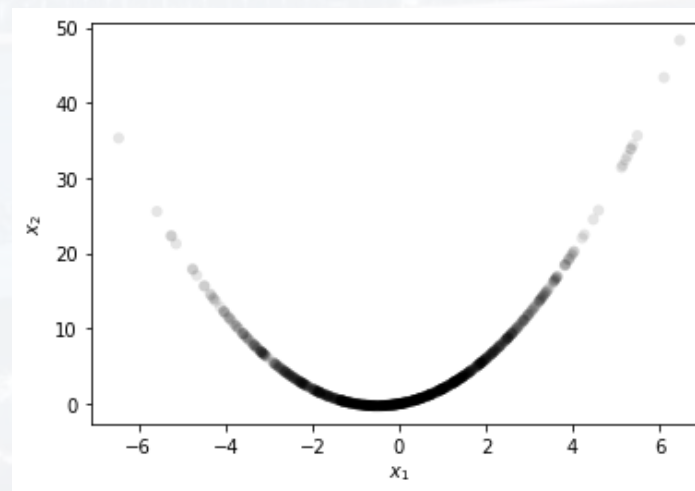
```
plt.xlabel('$x_1$')
```

```
plt.ylabel('$x_2$')
```

based on the shape of the
data cloud

→ prediction how x_1 and x_2
are related, i. e.

how they **correlate**





$$\text{cov}(x_1, x_2) = E(x_1 x_2) - E(x_1)E(x_2)$$

```
x1 = np.random.normal(0, 2, (1000,))
x2 = np.random.normal(3, 3, (1000,))

x3 = np.random.uniform(0, 5, (1000,))
x4 = 5*np.random.uniform(3, 4, (1000,))

x5 = np.sqrt(x4)
x6 = x1 + x2
x7 = 2*x3
x8 = x3*x2

All = np.vstack((x1, x2, x3, x4, x5, x6, x7, x8))
data = pd.DataFrame(All.transpose(),
                    columns = ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8'])

out = sns.pairplot(data, kind = "kde", \
                  plot_kws = {'color':[176/255, 224/255, 230/255]}, \
                  diag_kws = {'color':'black'})
out.map_offdiag(plt.scatter, color = 'black')
```



Dimension Reduction and PCA:

Variance and Covariance

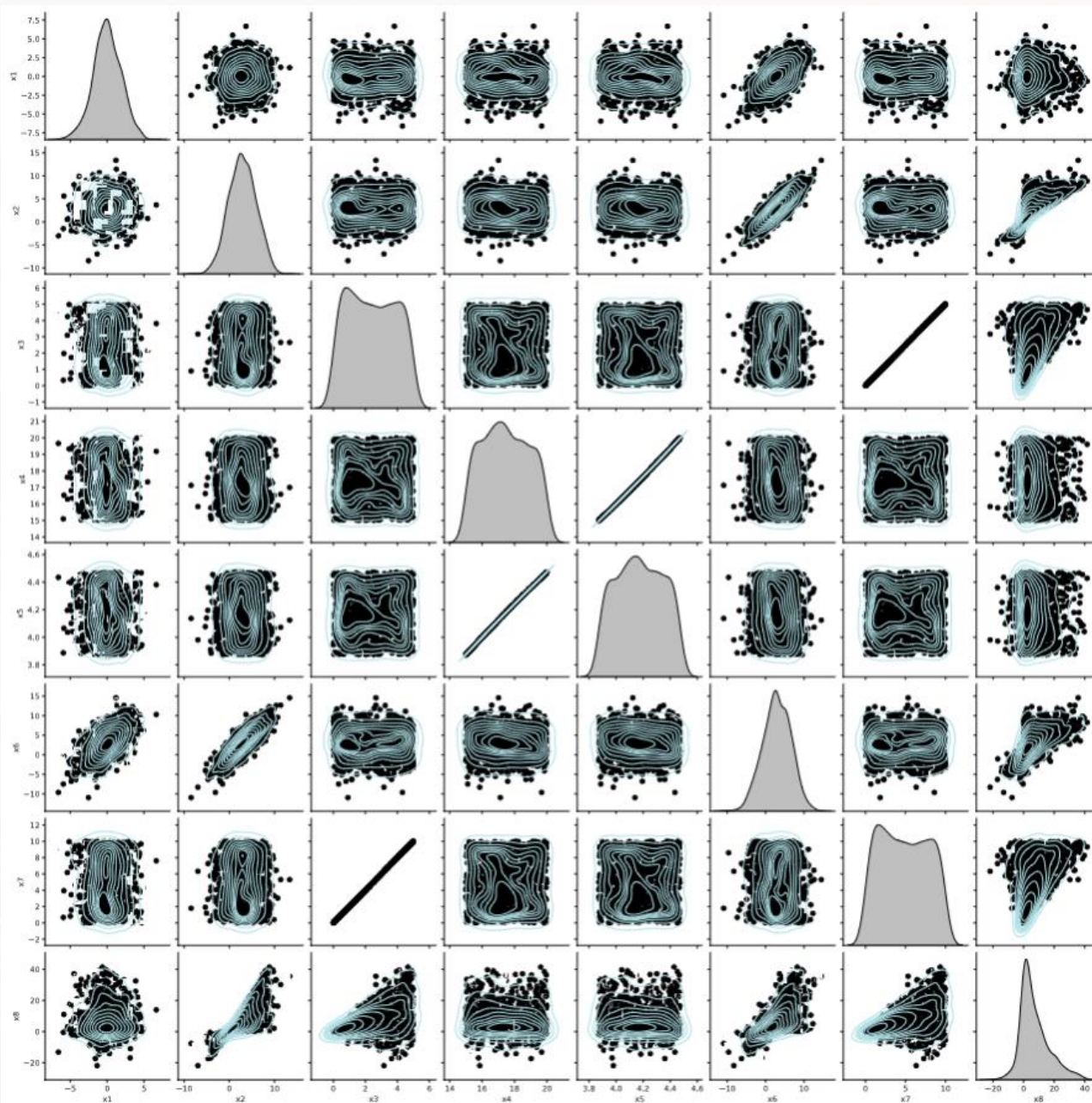
$$\text{cov}(x_1, x_2) = E(x_1 x_2) - E(x_1)E(x_2)$$

based on the shape of the data cloud

→ prediction how x_1 and x_2 are related, i. e. how they **correlate**

→ one can show:
 $\text{cov}(x_1, x_2) = 0$, x_1, x_2 are mutually independent

→ how to quantify exactly?





covariance matrix

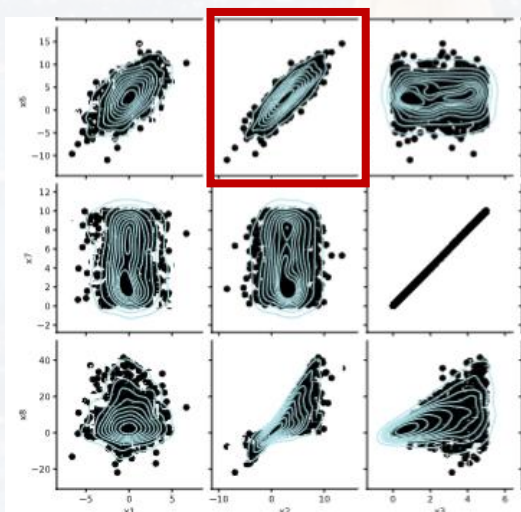
calculating $cov(x_i, x_j)$ for each combination x_i, x_j

$$\Sigma = \begin{pmatrix} cov(x_1, x_1) & \dots cov(x_1, x_j) \dots & cov(x_1, x_I) \\ \vdots & \vdots & \vdots \\ cov(x_i, x_1) & \dots cov(x_i, x_i) \dots & cov(x_i, x_I) \\ \vdots & \vdots & \vdots \\ cov(x_I, x_1) & \dots cov(x_I, x_j) \dots & cov(x_I, x_I) \end{pmatrix}$$

$$cov(x_i, x_j) = E(x_i x_j) - E(x_i)E(x_j)$$

for $i = j$

$$\begin{aligned} cov(x_i, x_i) &= E(x_i x_i) - E(x_i)E(x_i) \\ &= E(x_i^2) - E(x_i)^2 \\ &= \sigma^2(x_i) \end{aligned}$$





covariance matrix

$$\Sigma = \begin{pmatrix} \text{cov}(\mathbf{x}_1, \mathbf{x}_1) & \dots \text{cov}(\mathbf{x}_1, \mathbf{x}_j) \dots & \text{cov}(\mathbf{x}_1, \mathbf{x}_I) \\ \vdots & \vdots & \vdots \\ \text{cov}(\mathbf{x}_i, \mathbf{x}_1) & \dots \text{cov}(\mathbf{x}_i, \mathbf{x}_i) \dots & \text{cov}(\mathbf{x}_i, \mathbf{x}_I) \\ \vdots & \vdots & \vdots \\ \text{cov}(\mathbf{x}_I, \mathbf{x}_1) & \dots \text{cov}(\mathbf{x}_I, \mathbf{x}_j) \dots & \text{cov}(\mathbf{x}_I, \mathbf{x}_I) \end{pmatrix}$$

diagonal = variance of the variable

$$\text{cov}(\mathbf{x}_i, \mathbf{x}_j) = E(\mathbf{x}_i \mathbf{x}_j) - E(\mathbf{x}_i)E(\mathbf{x}_j)$$

for $i = j$

$$\begin{aligned} \text{cov}(\mathbf{x}_i, \mathbf{x}_i) &= E(\mathbf{x}_i \mathbf{x}_i) - E(\mathbf{x}_i)E(\mathbf{x}_i) \\ &= E(\mathbf{x}_i^2) - E(\mathbf{x}_i)^2 \\ &= \sigma^2(\mathbf{x}_i) \end{aligned}$$



covariance matrix

```
data = pd.read_csv('Cystfibr.txt',  
                  delimiter = '\t')
```

$$\Sigma = \begin{pmatrix} \text{cov}(x_1, x_1) & \dots \text{cov}(x_1, x_j) \dots & \text{cov}(x_1, x_I) \\ \vdots & \ddots & \vdots \\ \text{cov}(x_i, x_1) & \dots \text{cov}(x_i, x_i) \dots & \text{cov}(x_i, x_I) \\ \vdots & \ddots & \vdots \\ \text{cov}(x_I, x_1) & \dots \text{cov}(x_I, x_j) \dots & \text{cov}(x_I, x_I) \end{pmatrix}$$

diagonal = variance of the variable

age	sex	height	weight	bmp	fev1	rv	frc	tlc	pemax
7	0	109	13.1	68	32	258	183	137	95
7	1	112	12.9	65	19	449	245	134	85
8	0	124	14.1	64	22	441	268	147	100

run PlotCystFibr.py

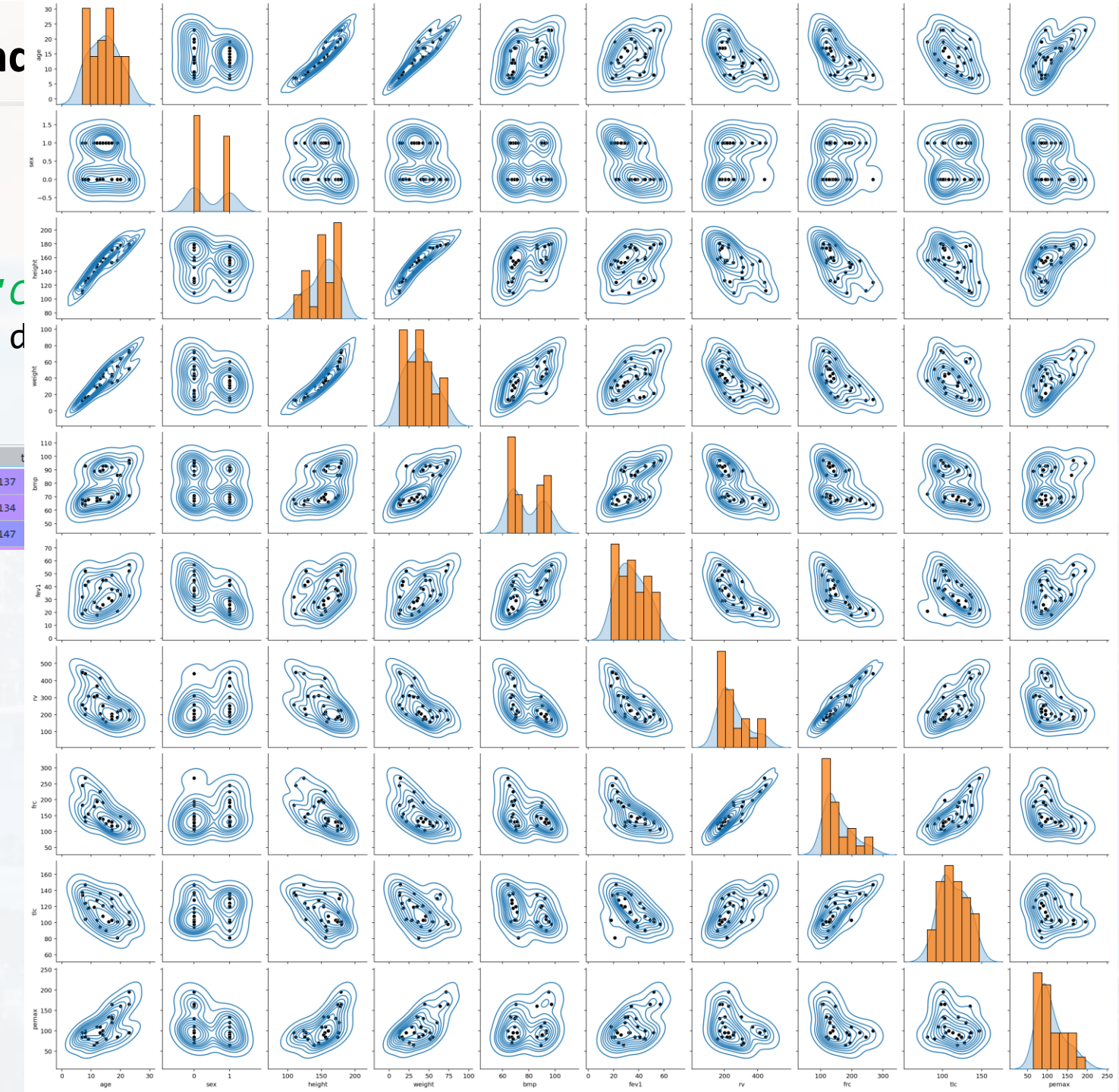


covariance matrix

```
data = pd.read_csv('cystfibr.csv')
```

age	sex	height	weight	bmp	fev1	rv	frc	t
7	0	109	13.1	68	32	258	183	137
7	1	112	12.9	65	19	449	245	134
8	0	124	14.1	64	22	441	268	147

run PlotCystFibr.py





covariance matrix

$$\Sigma = \begin{pmatrix} \text{cov}(x_1, x_1) & \dots \text{cov}(x_1, x_j) \dots & \text{cov}(x_1, x_I) \\ \vdots & \ddots & \vdots \\ \text{cov}(x_i, x_1) & \dots \text{cov}(x_i, x_i) \dots & \text{cov}(x_i, x_I) \\ \vdots & \ddots & \vdots \\ \text{cov}(x_I, x_1) & \dots \text{cov}(x_I, x_j) \dots & \text{cov}(x_I, x_I) \end{pmatrix}$$

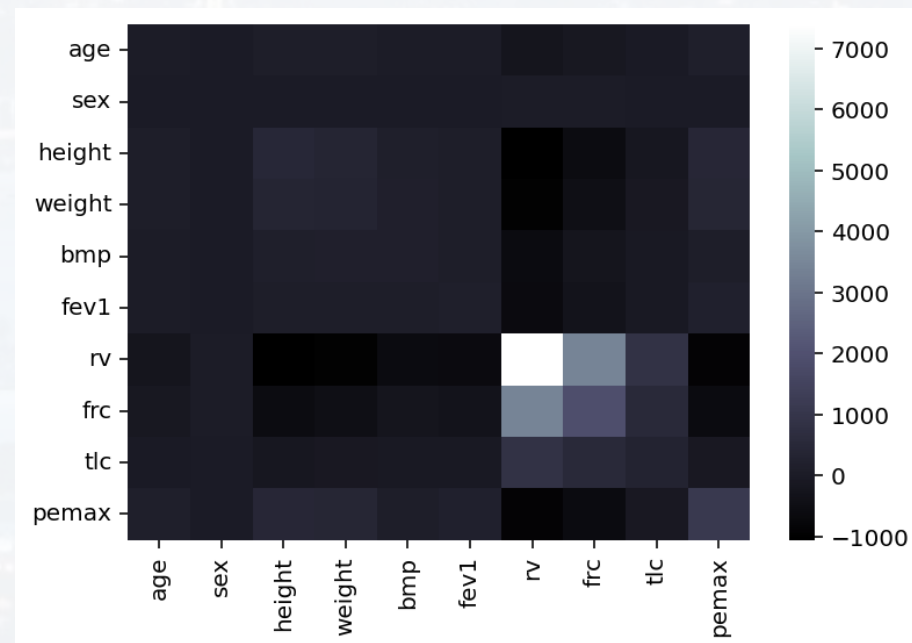
diagonal = variance of the variable

```
data = pd.read_csv('Cystfibr.txt', delimiter = '\t')
sns.heatmap(data.cov(), cmap = 'bone')
```

note: covariance is **not** scale invariant!

$$\text{cov}(x_i, x_j) = E(x_i x_j) - E(x_i)E(x_j)$$

$$\text{cov}(x_i, x_j) = \iint x_i x_j p(x_i) p(x_j) dx_i dx_j - \int x_i p(x_i) dx_i \int x_j p(x_j) dx_j$$





```
data = pd.read_csv('Cystfibr.txt', delimiter = '\t')  
sns.heatmap(data.cov(), cmap = 'bone')
```

note: covariance is **not** scale invariant!

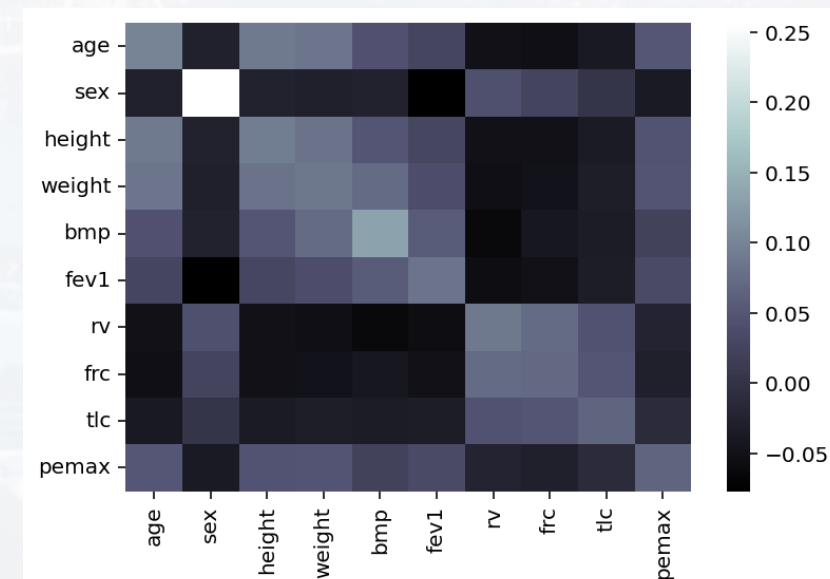
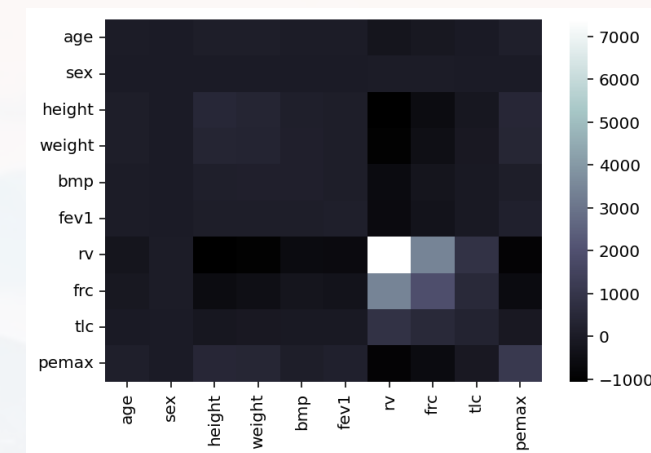
```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
```

```
data_scaled = scaler.fit_transform(data)
```

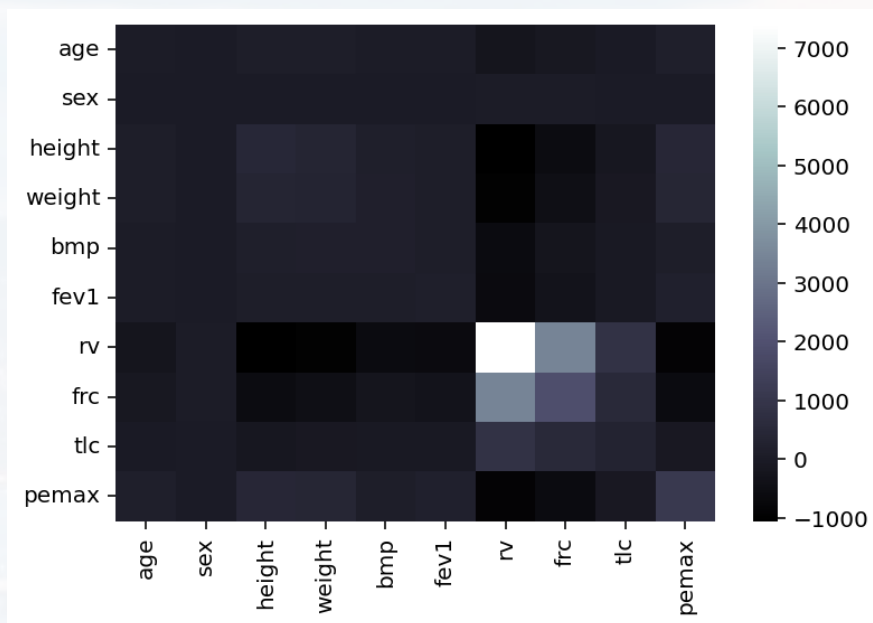
```
data_scaled = pd.DataFrame(data_scaled, columns = data.columns)
```

```
sns.heatmap(data_scaled.cov(), cmap = 'bone')
```

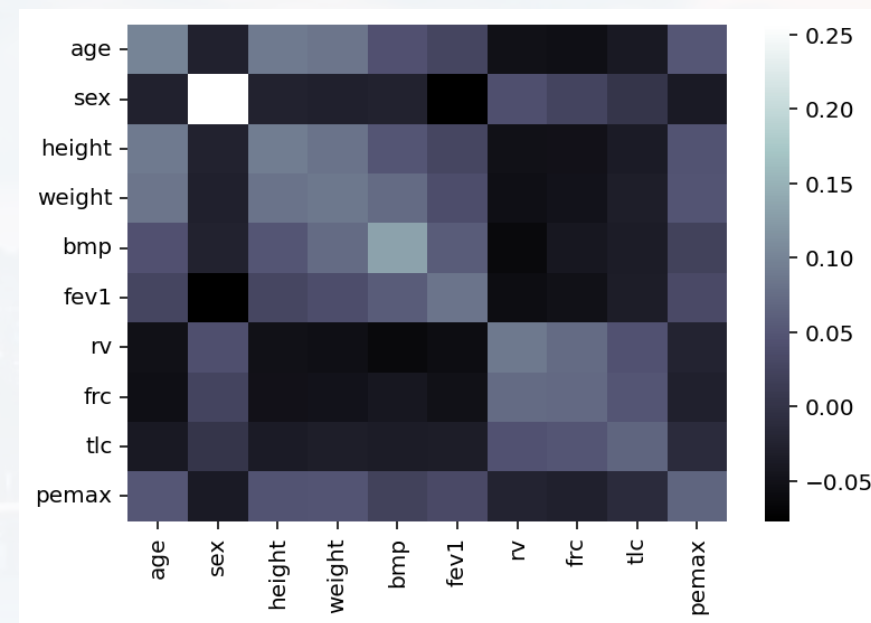




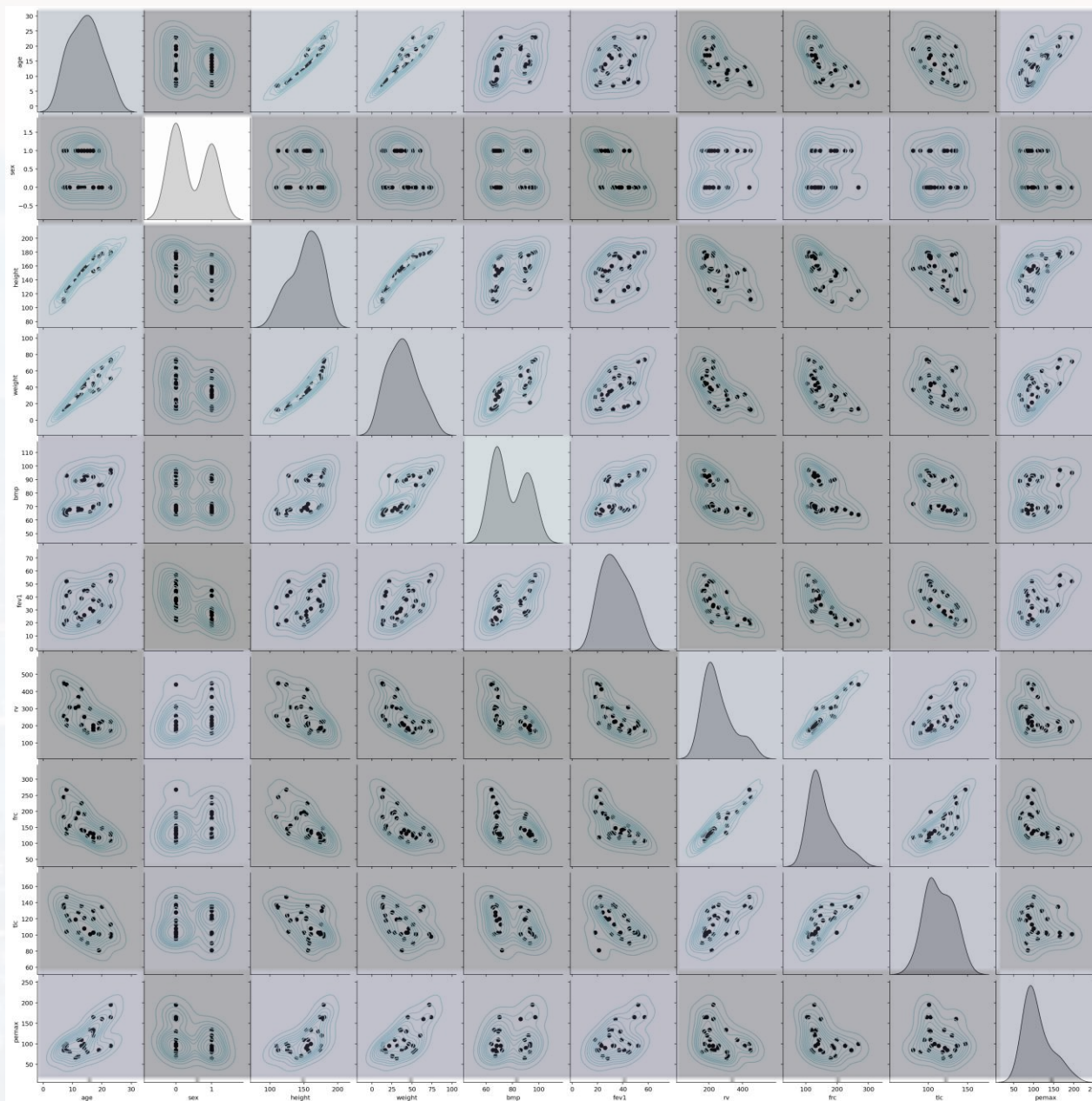
note: covariance is **not** scale invariant!



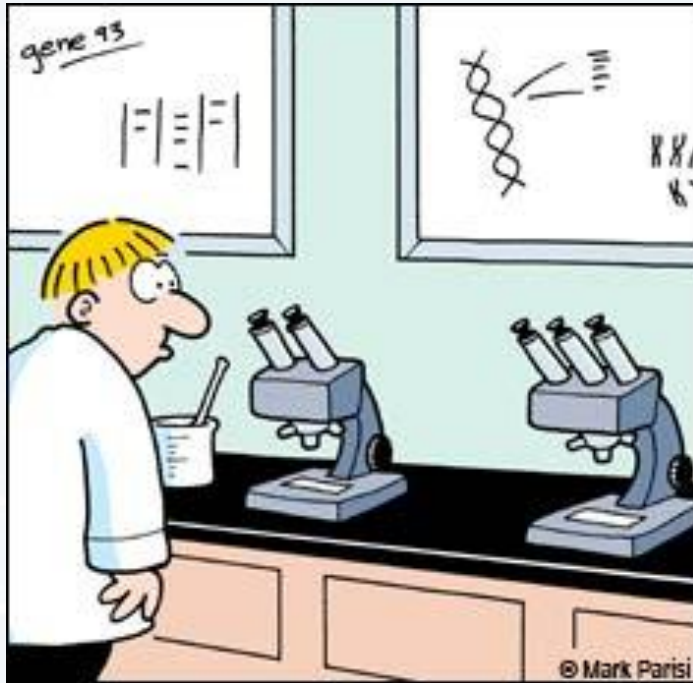
```
sns.heatmap(data.cov(),  
             cmap = 'bone')
```



```
sns.heatmap(data_scaled.cov(),  
             cmap = 'bone')
```



covariance gives us
some idea on how
the different features
relate to each other



Outline

Variance and Covariance

Correlation

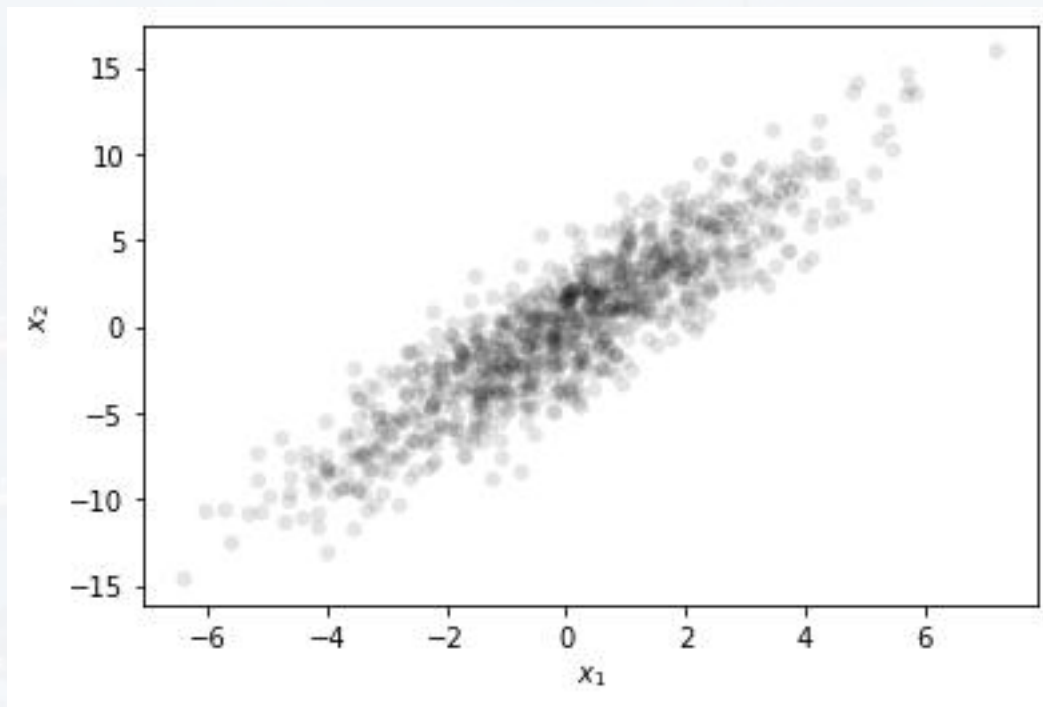
Principal Component Analysis (PCA)



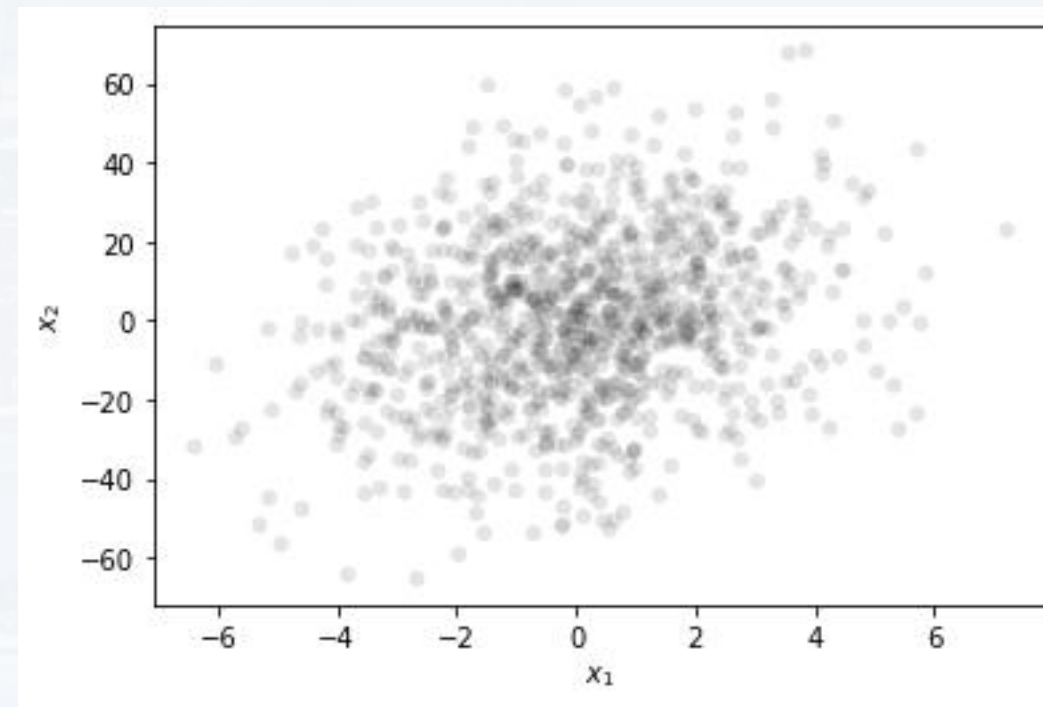
$$\text{cov}(x_1, x_2) = \text{cov}(x_2, x_1) = E(x_1 x_2) - E(x_1)E(x_2)$$

covariance

```
x1 = np.random.normal(0, 2, (1000,))  
x2 = 2*x1 + np.random.normal(0, 2, (1000,))
```



```
x1 = np.random.normal(0, 2, (1000,))  
x2 = 2*x1 + np.random.normal(0, 20, (1000,))
```

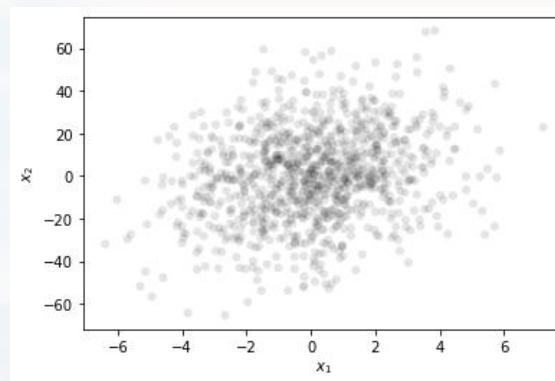
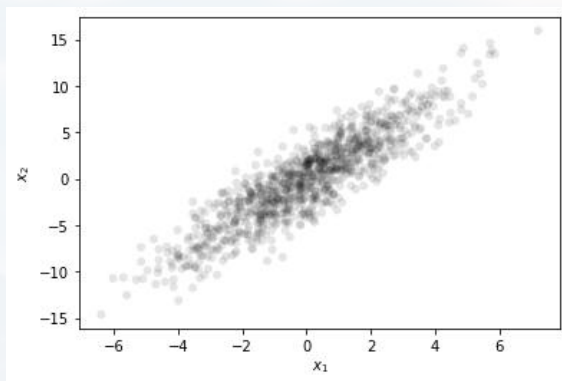


Same dependency, but different variance!



$$\text{cov}(x_1, x_2) = \text{cov}(x_2, x_1) = E(x_1 x_2) - E(x_1)E(x_2)$$

covariance



Same dependency, but different variance!

Need to scale for the variance!

Pearson's correlation
coefficient

$$\rho(x_1, x_2) = \frac{\text{cov}(x_1, x_2)}{\sqrt{\sigma_1^2 \sigma_2^2}}$$

$\rho(x_1, x_2)$:

- ranges from -1 to +1
- zero: no correlation
(completely independent)
- -1: max anti correlation
- +1: max correlation



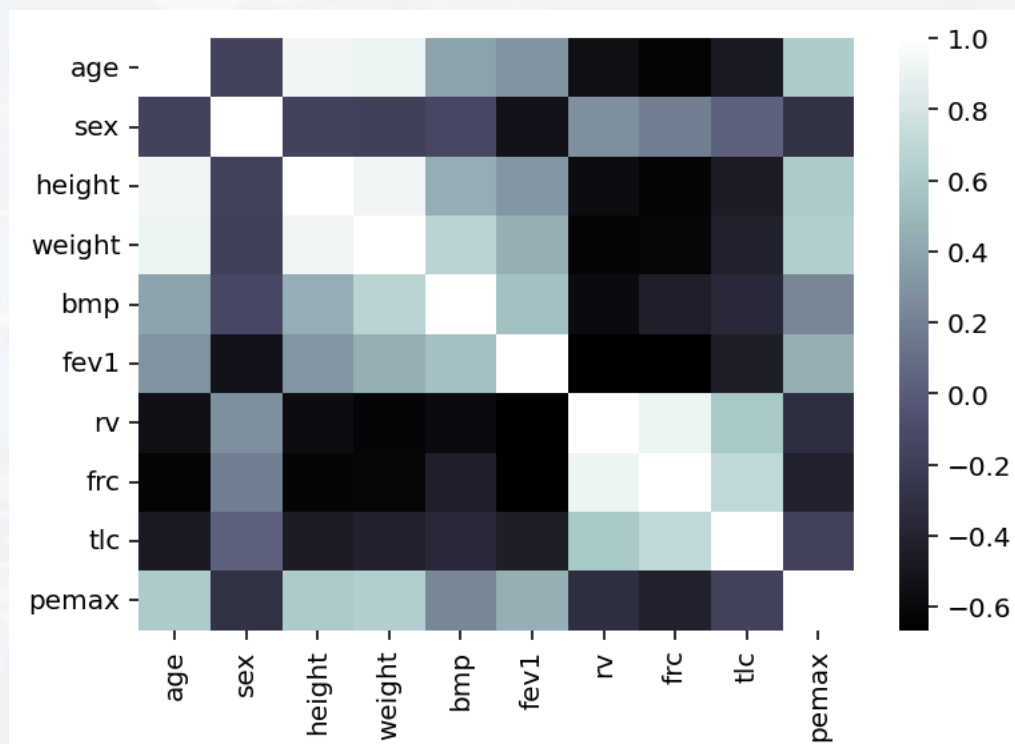
$$\text{cov}(x_1, x_2) = \text{cov}(x_2, x_1) = E(x_1 x_2) - E(x_1)E(x_2)$$

covariance

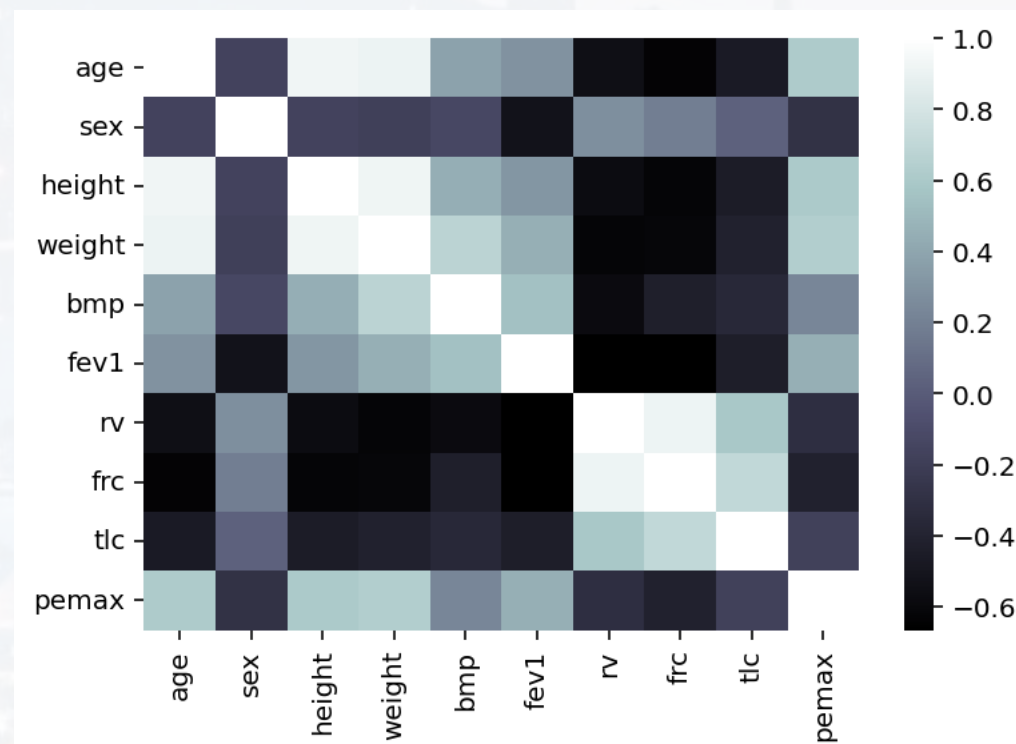
$$\rho(x_1, x_2) = \frac{\text{cov}(x_1, x_2)}{\sqrt{\sigma_1^2 \sigma_2^2}}$$

Pearson's correlation
coefficient

```
sns.heatmap(data.corr(), cmap = "bone")
```



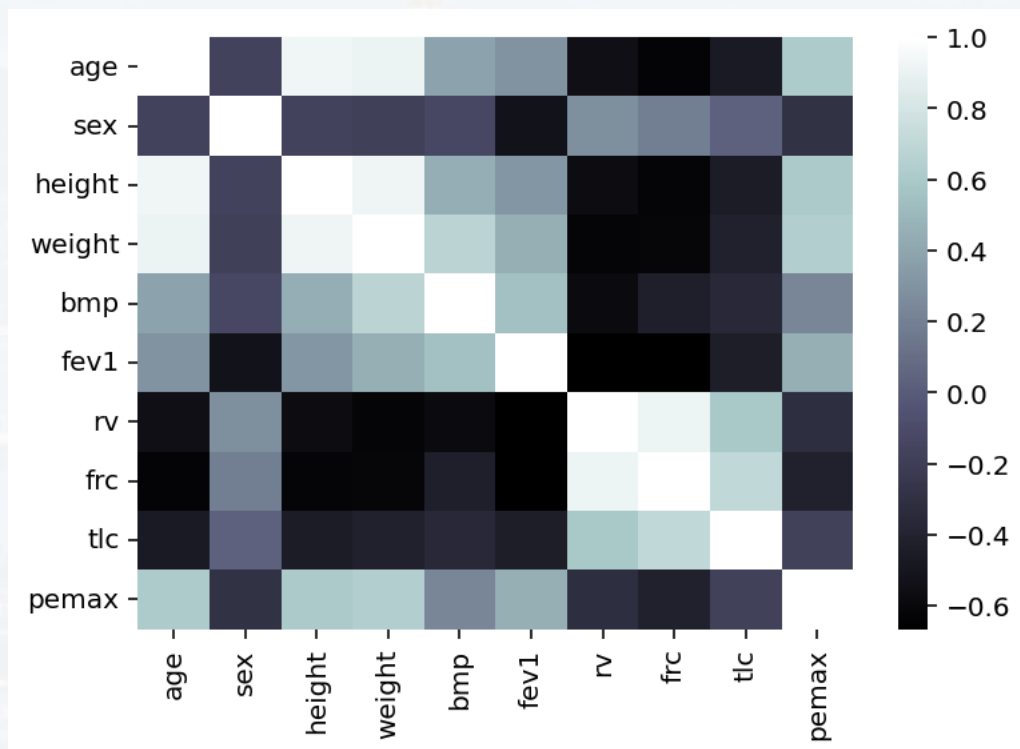
```
sns.heatmap(data_scaled.corr(),  
             cmap = "bone")
```



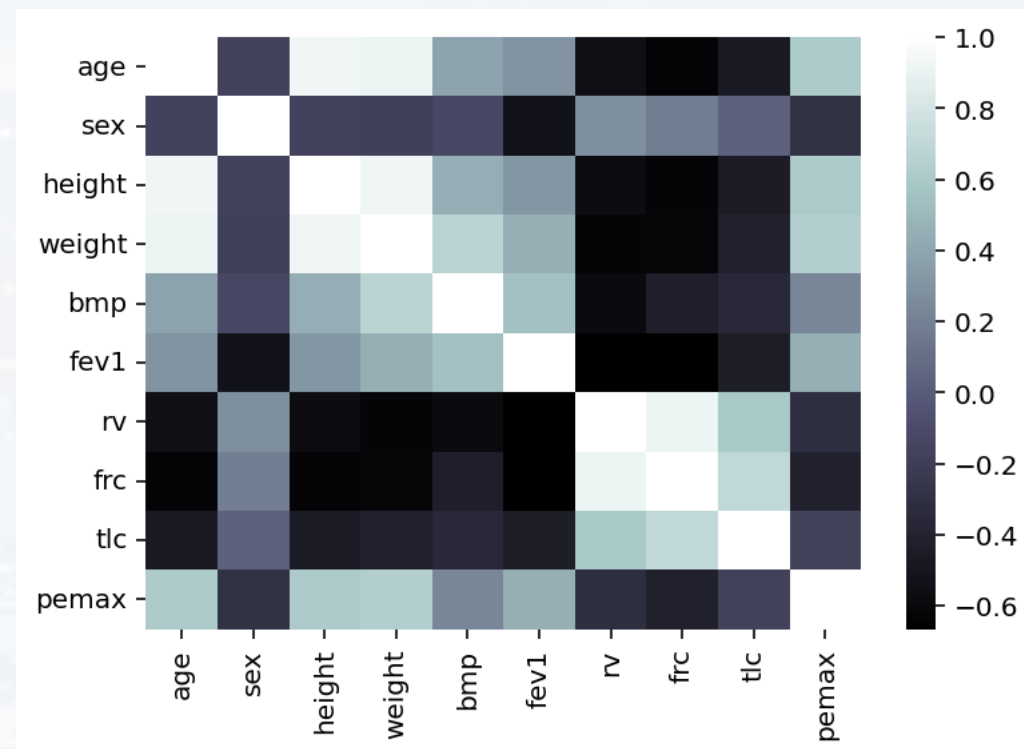


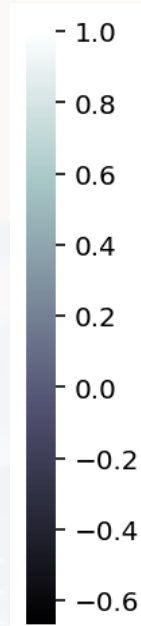
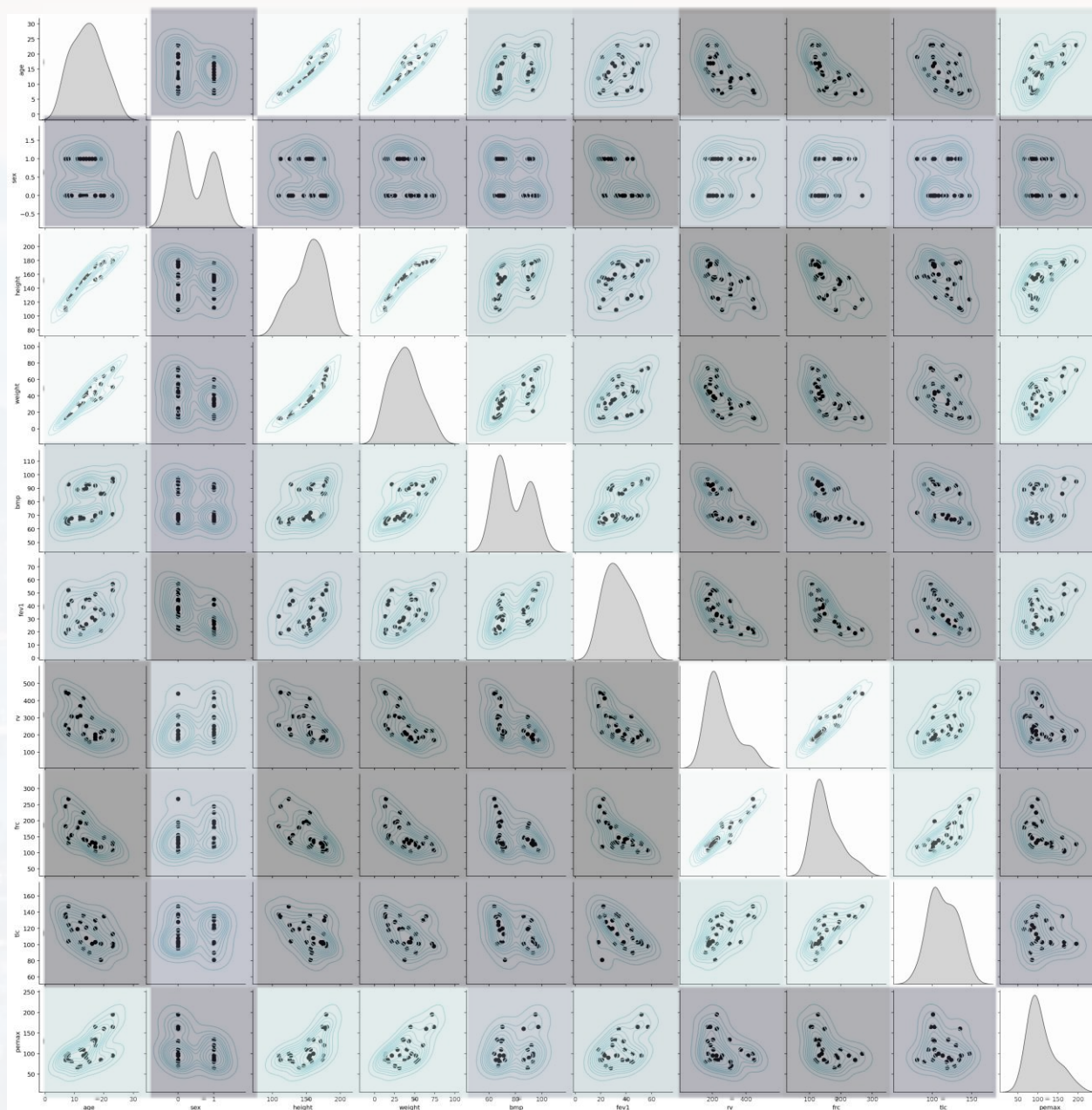
correlation **is** scale invariant!

```
sns.heatmap(data corr(), cmap = "bone")
```



```
sns.heatmap(data_scaled corr(),  
cmap = "bone")
```

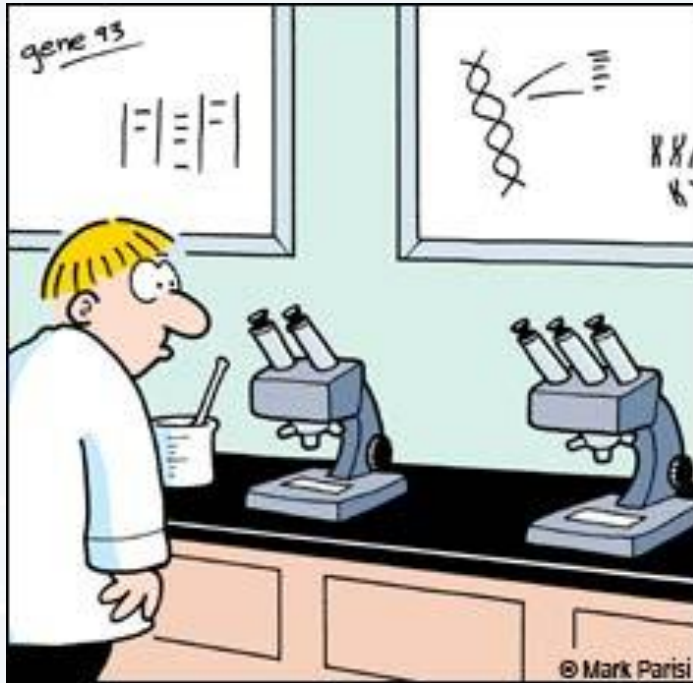




$$\rho(x_1, x_2) = \frac{\text{cov}(x_1, x_2)}{\sqrt{\sigma_1^2 \sigma_2^2}}$$

$\rho(x_1, x_2)$:

- ranges from -1 to +1
- zero: no correlation
(completely independent)
- -1: max anti correlation
- +1: max correlation
- **scale invariant**



Outline

Variance and Covariance

Correlation

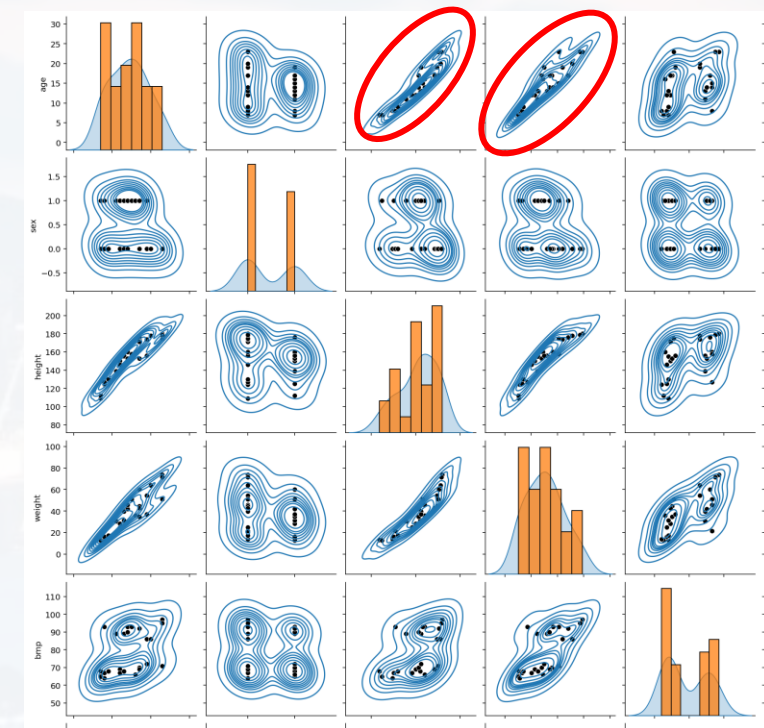
Principal Component Analysis (PCA)



correlation means:

- features are **not mutually independent**
 - we can predict feature x_i from feature x_j to some extent
 - we don't need all features
- **reducing number of features** (dimensions) without losing information

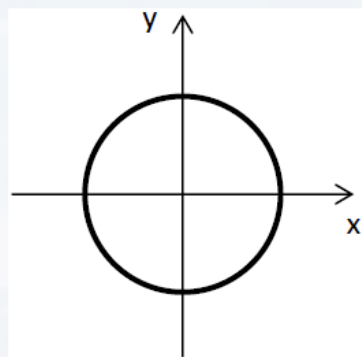
age	sex	height	weight	bmp	fev1	rv	frc	tlc	pemax
7	0	109	13.1	68	32	258	183	137	95
7	1	112	12.9	65	19	449	245	134	85
8	0	124	14.1	64	22	441	268	147	100





about quadratic forms

circle



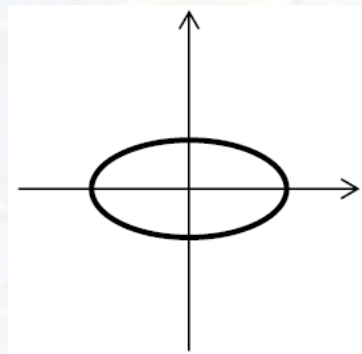
“distance to a reference point is constant”

$$x^2 + y^2 = \text{const} = r^2$$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \text{const}$$

$$a = b \rightarrow x^2 + y^2 = r^2$$

ellipse



“stretching the coordinate system”

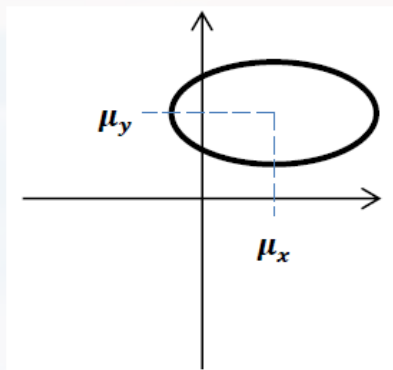
$$a \neq b$$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \text{const}$$



about quadratic forms

ellipse



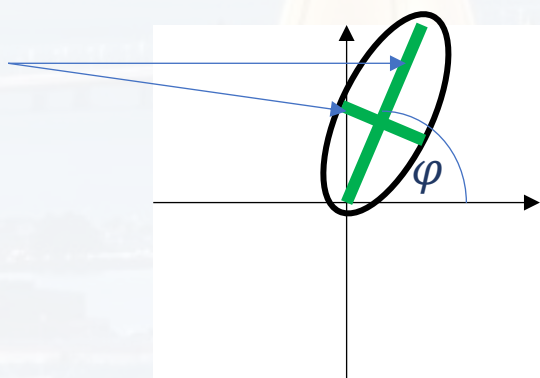
“moving the center of the ellipse”

$$\frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} = \text{const}$$

$$x_0 = 0, x_0 \rightarrow \mu_x$$

$$y_0 = 0, y_0 \rightarrow \mu_y$$

principal
axes



“turning the ellipse by an angle φ ”

$$\varphi = \frac{1}{2} \text{atan} \left(\frac{c}{\frac{1}{a^2} - \frac{1}{b^2}} \right)$$

$$\frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} + c(x - \mu_x)(y - \mu_y) = \text{const}$$

turning the form



about quadratic forms

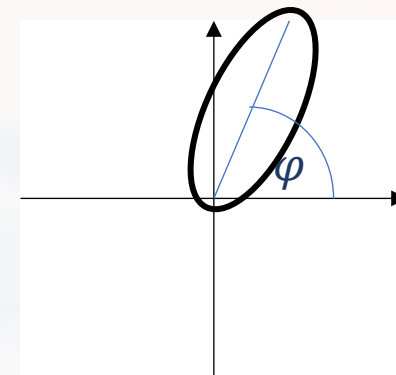
$$\frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} + c(x - \mu_x)(y - \mu_y) = \text{const}$$

matrix form:

$$\text{const} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & c/2 \\ c/2 & 1/b^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

often:

$$\text{const} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} A & C/2 \\ C/2 & B \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$



more general:

$$\text{const} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \\ 1 \end{pmatrix}^T \begin{pmatrix} A & C/2 & D/2 \\ C/2 & B & E/2 \\ D/2 & E/2 & F \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \\ 1 \end{pmatrix}$$

depending on A, B, C, D, E, F

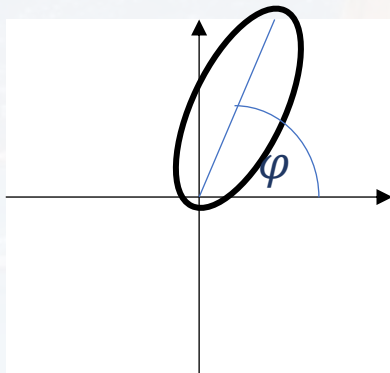
- circle
- ellipse
- parabola
- hyperbola



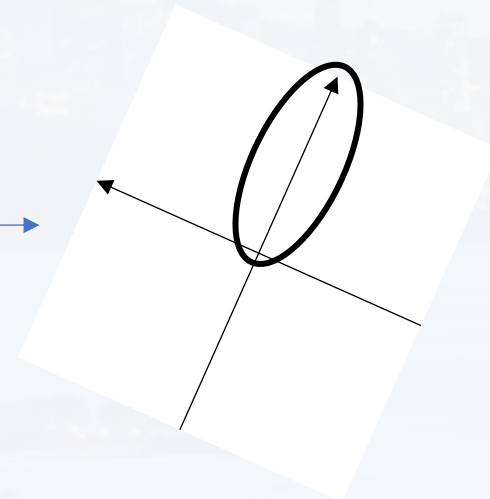
about quadratic forms

$$\frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} + \boxed{c(x - \mu_x)(y - \mu_y)} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & \boxed{c/2} \\ \boxed{c/2} & 1/b^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix} = \text{const}$$

turning the form



turning the coordinate system



$$\frac{(x_{\text{new}} - \mu_{x(\text{new})})^2}{a_{\text{new}}^2} + \frac{(y_{\text{new}} - \mu_{y(\text{new})})^2}{b_{\text{new}}^2} = \begin{pmatrix} x_{\text{new}} - \mu_{x(\text{new})} \\ y_{\text{new}} - \mu_{y(\text{new})} \end{pmatrix}^T \begin{pmatrix} 1/a_{\text{new}}^2 & 0 \\ 0 & 1/b_{\text{new}}^2 \end{pmatrix} \begin{pmatrix} x_{\text{new}} - \mu_{x(\text{new})} \\ y_{\text{new}} - \mu_{y(\text{new})} \end{pmatrix} = \text{const}$$



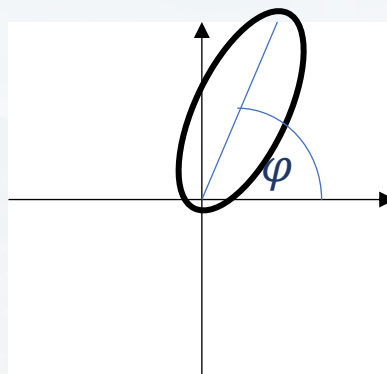
about quadratic forms

non – diagonal elements:

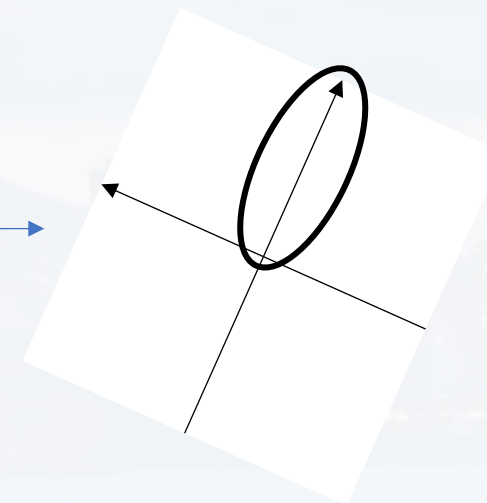
- turn/shear the object

diagonal elements:

- stretches (or flips, if negative) the object



turning the coordinate system



not turned/sheared

→ principal axes of the object are **parallel to the coordinate** axes

new coord. axis are called: **eigenvectors** \vec{v} (“eigen”, German for “proper”)

→ they span the proper coordinate system!

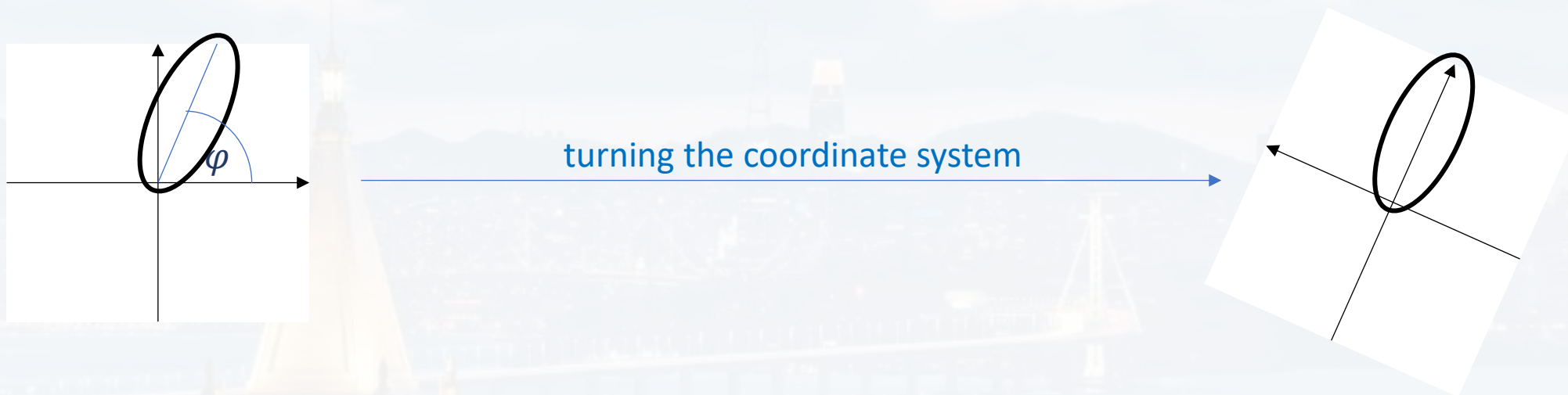
$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_i & & 0 \\ 0 & 0 & & \lambda_N \end{pmatrix}$$

in the proper coordinate system: matrix is diagonal (entries are called **“eigenvalues”** λ)



about quadratic forms

- non – diagonal elements: - turn/shear the object
diagonal elements: - stretches (or flips, if negative) the object



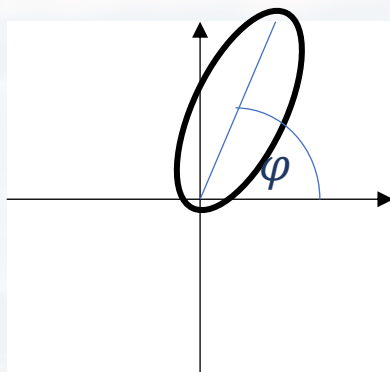
In a coordinate system in which the principal axes are **parallel to the coordinate** axes

- the matrix that defines the form is **diagonal**
- the **entries** of the now diagonal matrix are called **eigenvalues λ**
- the **axes** of this coordinates system are called **eigenvectors \vec{v}**
- eigen means **“proper”**, i. e. it is the **“most suitable”** coordinate system

$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_i & & 0 \\ 0 & 0 & & \lambda_N \end{pmatrix}$$

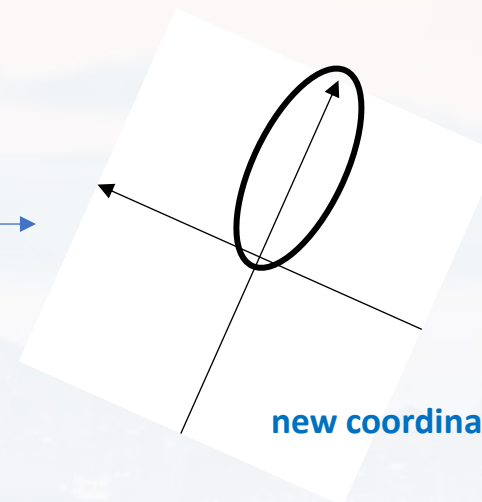


about quadratic forms



old coordinates

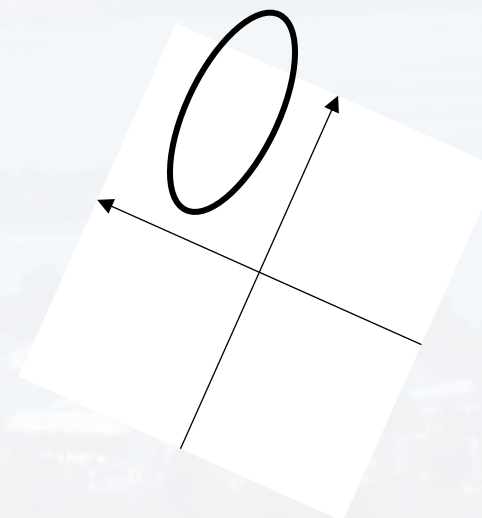
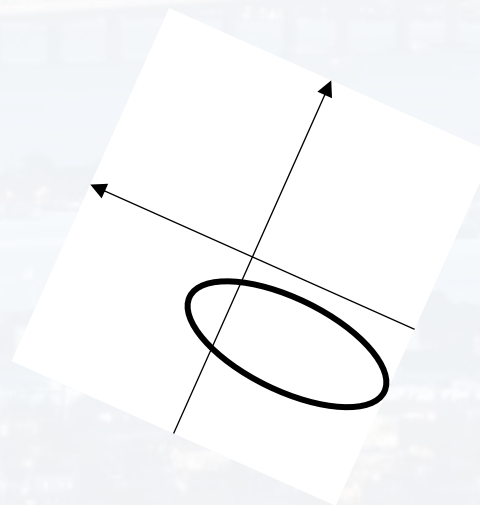
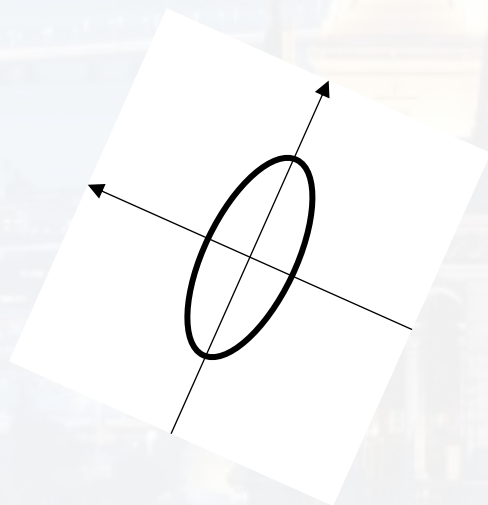
turning the coordinate system



new coordinates

not turned/sheared

→ principal axes of the object are **parallel to the coordinate** axis

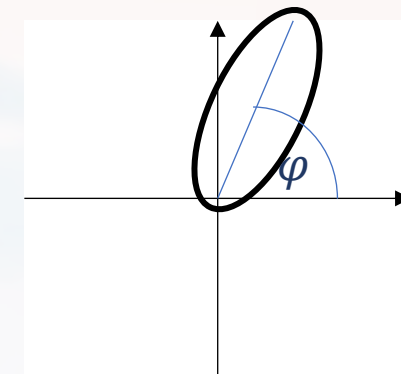




...so, what PCA actually is:

1) turned ellipse

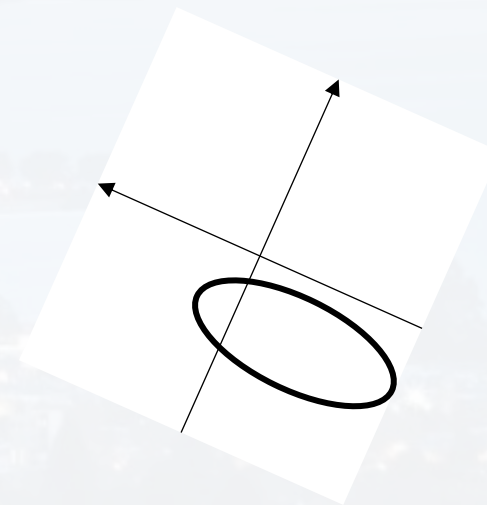
$$\begin{pmatrix} x \\ y \end{pmatrix}^T \underbrace{\begin{pmatrix} A & C/2 \\ C/2 & B \end{pmatrix}}_M \begin{pmatrix} x \\ y \end{pmatrix} = A x^2 + B y^2 + C xy$$



2) turning the coordinate system, such that principal axes of the are **parallel to the coordinate**

$$\begin{pmatrix} x_{new} \\ y_{new} \end{pmatrix}^T \underbrace{\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}}_{M_{new}} \begin{pmatrix} x_{new} \\ y_{new} \end{pmatrix} = \lambda_1 x_{new}^2 + \lambda_2 y_{new}^2$$

eigenvalues λ



But what does it have to do with covariance and correlation?



$$\sigma_{tot}^2 = \boxed{\sigma_x^2} + \boxed{\sigma_y^2} + \boxed{2 \operatorname{cov}(x, y)}$$

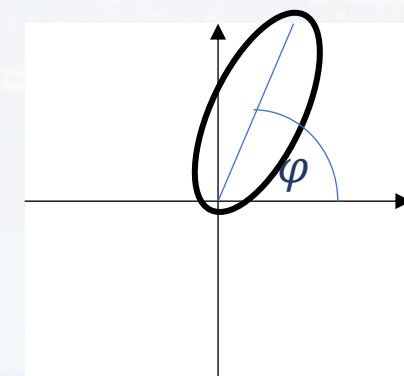
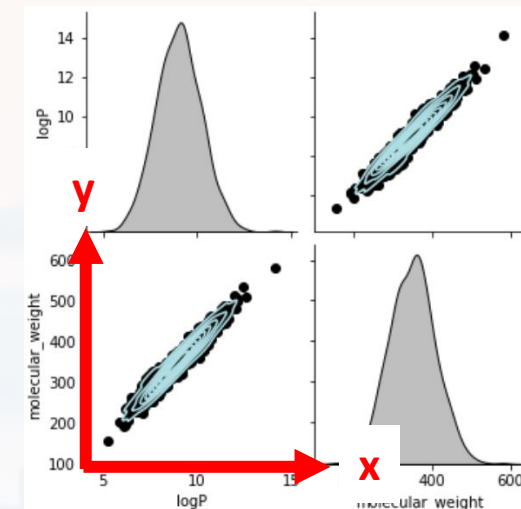
$$= \boxed{\sum_i^N (x_i - \mu_x)^2} + \boxed{\sum_j^M (y_j - \mu_y)^2} + \boxed{2 \sum_j^M \sum_i^N (x_i - \mu_x)(y_j - \mu_y)}$$

$$const = \boxed{\frac{(x - \mu_x)^2}{a^2}} + \boxed{\frac{(y - \mu_y)^2}{b^2}} + \boxed{2 c (x - \mu_x)(y - \mu_y)}$$

It is the same structure!

$$const = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & c \\ c & 1/b^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

$$C = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} \sigma_x^2 & \operatorname{cov}(y, x) \\ \operatorname{cov}(x, y) & \sigma_y^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix} \quad \operatorname{cov}(y, x) = \operatorname{cov}(x, y)$$





$$\sigma_{tot}^2 = \sigma_x^2 + \sigma_y^2 + 2 \operatorname{cov}(x, y)$$

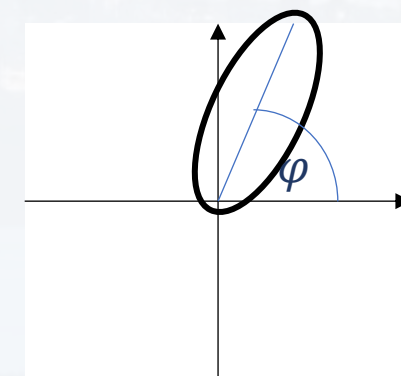
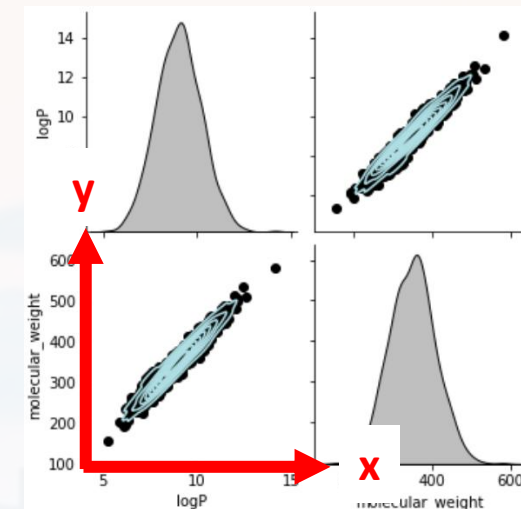
$$= \sum_i^N (x_i - \mu_x)^2 + \sum_j^M (y_j - \mu_y)^2 + 2 \sum_j^M \sum_i^N (x_i - \mu_x)(y_j - \mu_y)$$

$$const = \frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} + 2c(x - \mu_x)(y - \mu_y)$$

It is the same structure!

$$const = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & c \\ c & 1/b^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

$$C = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} \sigma_x^2 & \operatorname{cov}(y, x) \\ \operatorname{cov}(x, y) & \sigma_y^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix} \quad \operatorname{cov}(y, x) = \operatorname{cov}(x, y)$$





$$C = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \underbrace{\begin{pmatrix} \sigma_x^2 & \text{cov}(y, x) \\ \text{cov}(x, y) & \sigma_y^2 \end{pmatrix}}_{\text{covariance matrix } \Sigma} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix} \quad \text{cov}(y, x) = \text{cov}(x, y)$$

- geometrically, the **covariance matrix** can be interpreted as quadratic form
- the covariances are the **non-diagonal** elements of the **covariance matrix**
- aim: finding a coordinate transformation, where the **covariance matrix** is diagonal

$$\begin{pmatrix} \lambda_1 & \ddots & 0 & \dots & 0 \\ 0 & & \lambda_i & \ddots & 0 \\ 0 & & 0 & & \lambda_I \end{pmatrix}$$

the diagonal
entries are called
eigenvalues (= variances in
new coordinate system)

- all variables **are independent**
- principal components of the **covariance matrix**
are **parallel** to the **new coordinate axes** (= **eigenvectors**)



How to interpret the eigenvalues?

$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_i & & 0 \\ 0 & 0 & & \lambda_I \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} \sigma_{11}^2 & \dots \text{cov}(x_1, x_j) \dots & \text{cov}(x_1, x_I) \\ \dots & \dots & \dots \\ \text{cov}(x_i, x_1) & \dots \sigma_{ii}^2 \dots & \text{cov}(x_i, x_I) \\ \dots & \dots & \dots \\ \text{cov}(x_I, x_1) & \dots \text{cov}(x_I, x_j) \dots & \sigma_{II}^2 \end{pmatrix}$$

PCA



$$\Sigma_{new} = \begin{pmatrix} \sigma_{11}^2(new) & \dots 0 \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots \sigma_{ii}^2(new) \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots 0 \dots & \sigma_{II}^2(new) \end{pmatrix}$$



How to interpret the eigenvalues?

$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_i & & 0 \\ 0 & 0 & & \lambda_I \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} \sigma_{11}^2 & \dots \text{cov}(x_1, x_j) \dots & \text{cov}(x_1, x_I) \\ \dots & \dots & \dots \\ \text{cov}(x_i, x_1) & \dots \sigma_{ii}^2 \dots & \text{cov}(x_i, x_I) \\ \dots & \dots & \dots \\ \text{cov}(x_I, x_1) & \dots \text{cov}(x_I, x_j) \dots & \sigma_{II}^2 \end{pmatrix}$$

PCA



$$\Sigma_{new} = \begin{pmatrix} \sigma_{11}^2(new) & \dots 0 \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots \sigma_{ii}^2(new) \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots 0 \dots & \sigma_{II}^2(new) \end{pmatrix}$$

the **eigenvalues** of the **covariance matrix** are the **variances** in the **new coordinate** system in which the **principal components** of the **covariance matrix** are **parallel** to the new coordinate axis (= **eigenvectors**)

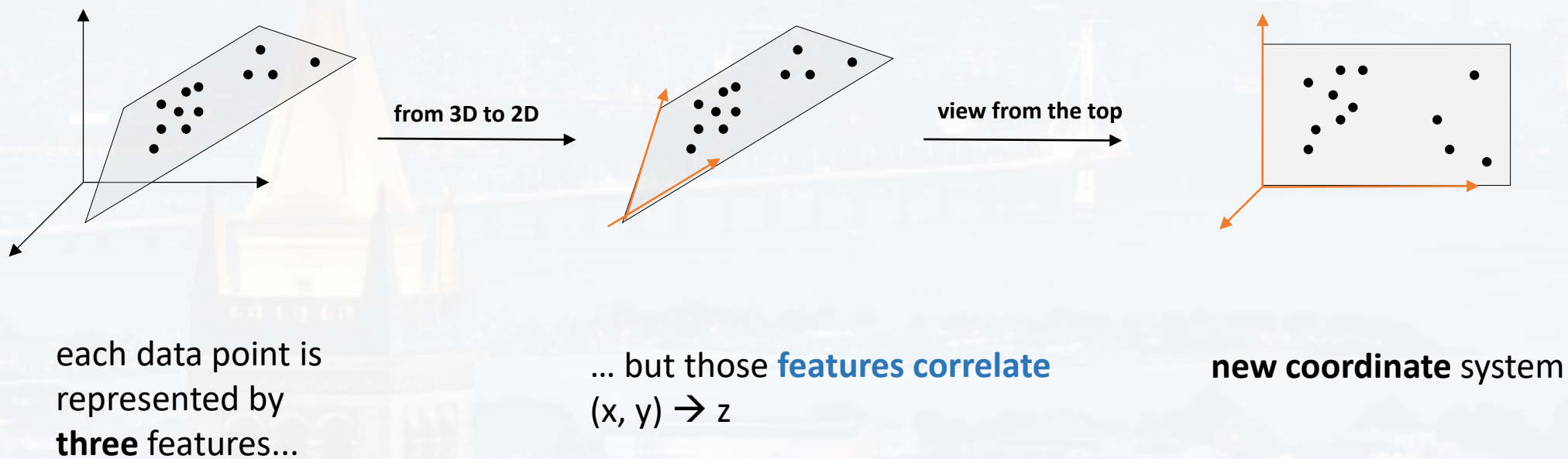


How to interpret the eigenvalues?

$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_i & & 0 \\ 0 & 0 & & \lambda_I \end{pmatrix}$$

the **eigenvalues** of the **covariance matrix** are the **variances** in the **new coordinate** system

What if eigenvalues are close to zero?



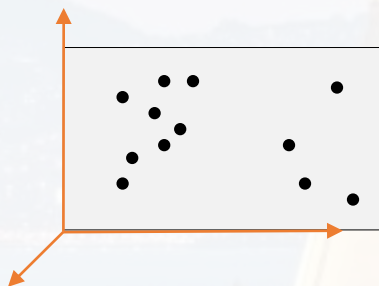


How to interpret the eigenvalues?

$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_i & & 0 \\ 0 & 0 & & \lambda_I \end{pmatrix}$$

the **eigenvalues** of the **covariance matrix** are the **variances** in the **new coordinate** system

What if eigenvalues are close to zero?



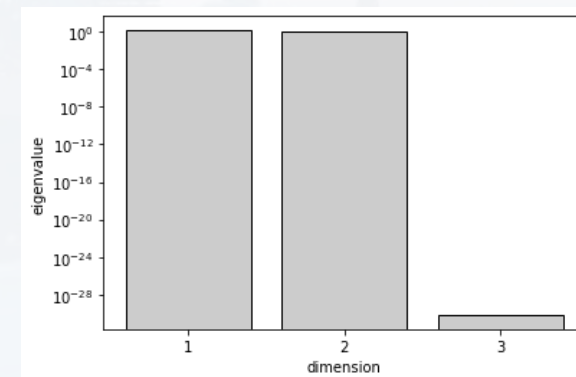
new coordinate system

some variance in x_{new}, y_{new}

almost no variance in z_{new}

→ because data exists on 2D, not 3D

→ dimensionality reduction!!



We can reduce the complexity of the data set without losing information

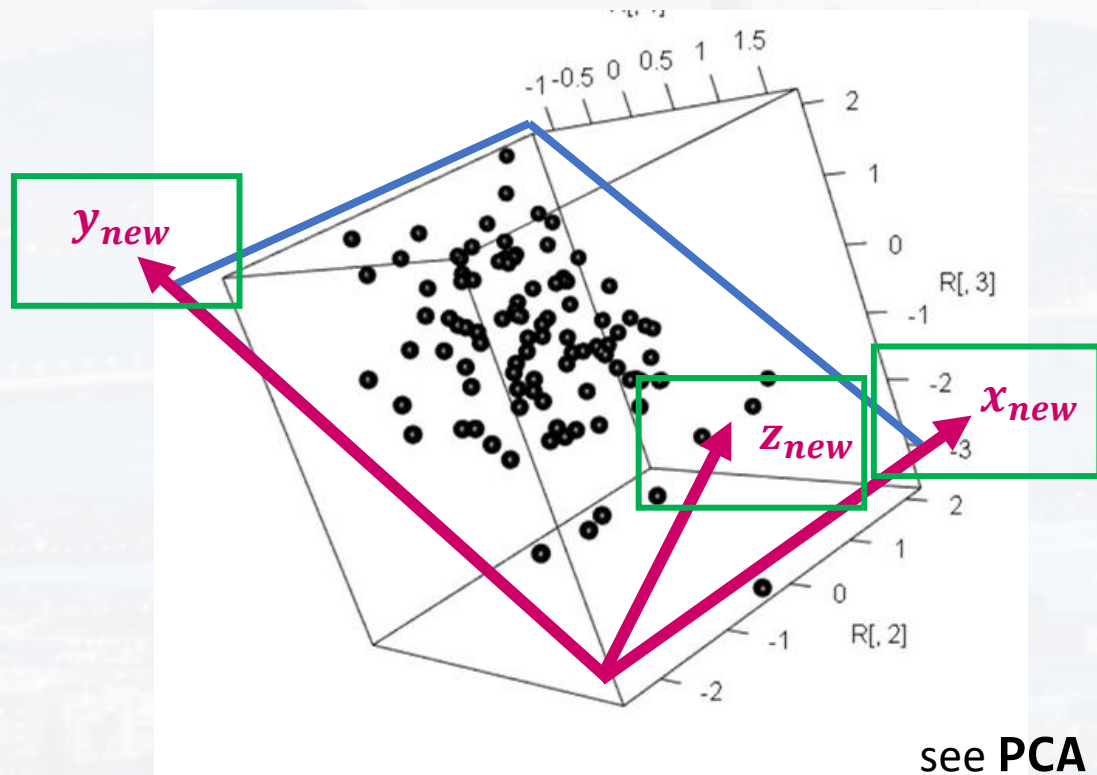


How to interpret the eigenvalues?

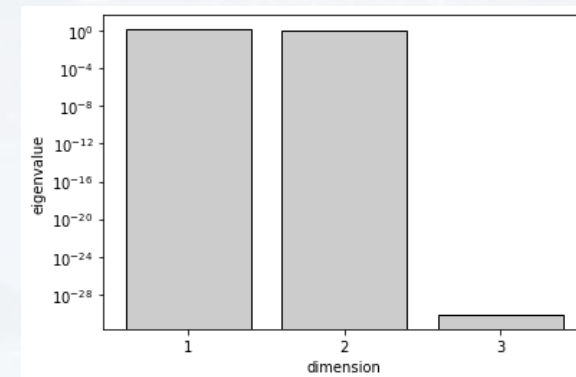
$$\begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_i & & 0 \\ 0 & 0 & & \lambda_I \end{pmatrix}$$

the **eigenvalues** of the **covariance matrix** are the **variances** in the **new coordinate** system

What if eigenvalues are close to zero?



each eigenvector is a **linear combination** of the old coordinate axes



see `PCA_simple.ipynb`

We can reduce the complexity of the data set without losing information



Summary:

- PCA transforms the data into a proper (= **eigen**) coordinate system
- The coordinate axis of this coordinate system are called **eigenvectors**
- Covariance terms in the covariance matrix Σ_{new} disappear
- Therefore, Σ_{new} is a **diagonal matrix**
- The diagonal elements are called **eigenvalues**...
- ...which are the **variances of the data in the proper (=eigen) coordinate system**
- If some eigenvalues are smaller than others, it means there is less variance in that direction
- We only select those directions **with large eigenvalues** (90%, 95% of total variance)
- and therefore **reduce dimensionality**



Note:

- PCA **is not scale invariant** because it acts on variances/covariances which are **not scale invariant**
- **scale/normalize data first!**
- run PCA before you continue with any ML method
- PCA is a linear operation (= linear algebra). It does not work if data lives on a curved space (manifold)
 - flat eigenvalue spectrum could be an evidence that PCA is not applicable
- Alternatives are SVMs, LDA, and UMAP for visualization

It is all still a bit abstract, so let us explore **PCA_Molecules.ipynb**

Thank you very much for your attention!

