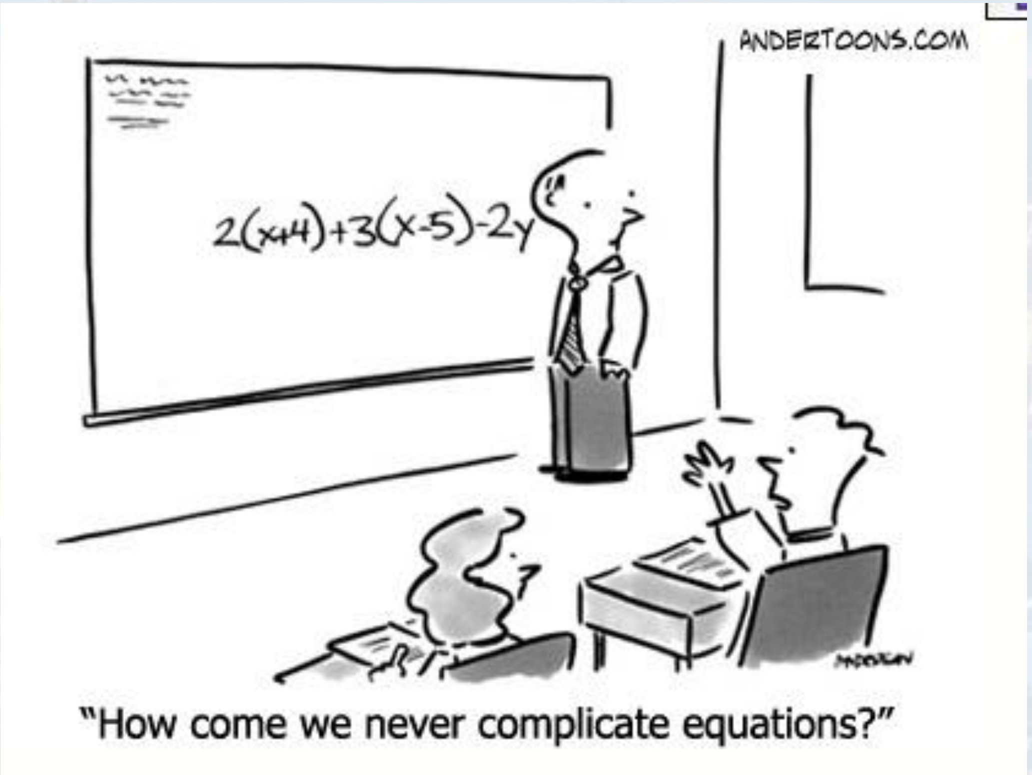
M. Hohle:

Physics 77: Introduction to Computational Techniques in Physics



Introduction to Computational Techniques in Physics: syllabus

Week	Date	Topic
1	June 12th	Programming Environment & UIs for Python, Programming Fundamentals
2	*June 19th*	Basic Types in Python
3	June 26th	Parsing, Data Processing and File I/O, Visualization
4	July 3rd	Functions, Map & Lambda
5	July 10th	Random Numbers & Probability Distributions, Interpreting Measurements
6	July 17th	Numerical Integration and Differentiation
7	July 24th	Root finding, Interpolation
8	**July 31st**	**Systems of Linear Equations,** Ordinary Differential Equations (ODEs)
9	Aug 7th	Stability of ODEs, Examples
10	Aug 14th	Final Project Presentations

finding the intersection of **two lines**:

$$y_1 = a_1 x_1 + c_1$$

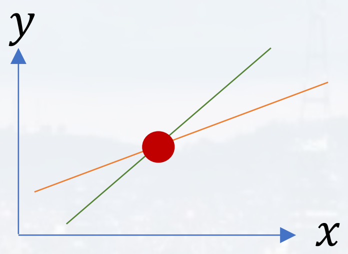
$$y_2 = a_2 x_2 + c_2$$

$$x_1 = x_2$$
$$y_1 = y_2$$

$$a_2 x + c_2 = a_1 x + c_1$$

$$x = \frac{c_2 - c_1}{a_1 - a_2}$$

$$y = a_1 \frac{c_2 - c_1}{a_1 - a_2} + c_1$$



finding the intersection of **three planes**:

$$y_1 = a_{11} x_{11} + a_{12} x_{12} + c_1$$

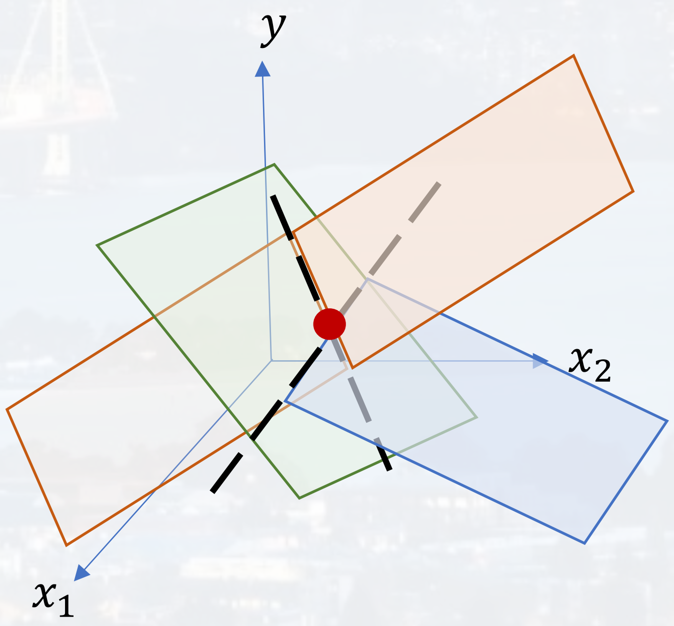
$$y_2 = a_{21} x_{21} + a_{22} x_{22} + c_2$$

$$y_2 = a_{31} x_{31} + a_{32} x_{32} + c_2$$

$$x_{11} = x_{21} = x_{31} = x_1$$

$$x_{12} = x_{22} = x_{32} = x_2$$

$$y_1 = y_2 = y_3 = y$$

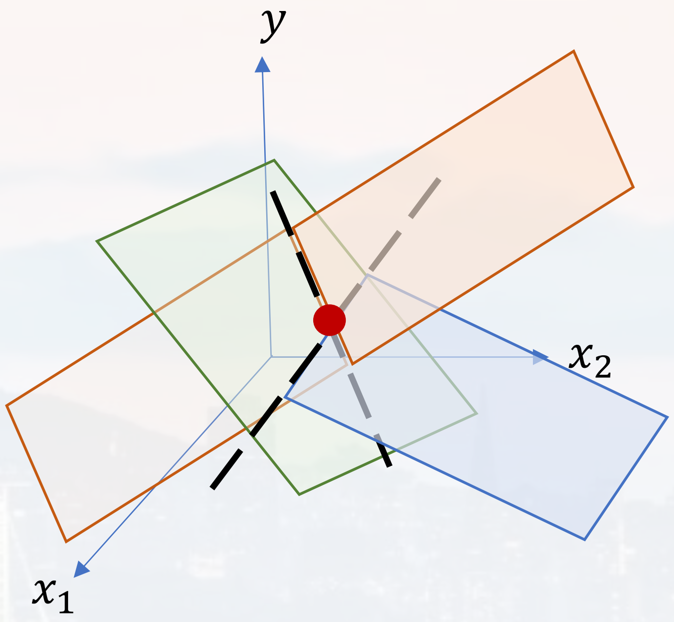


more general:

$$x_{11} = x_{21} = x_{31} = x_1$$

$$x_{12} = x_{22} = x_{32} = x_2$$

$$y_1 = y_2 = y_3 = y$$

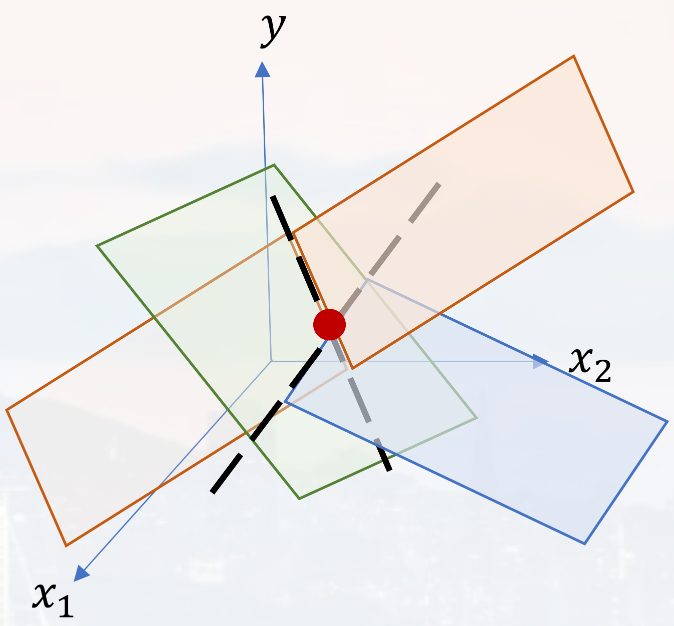


$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ \ldots + a_{1n}x_n = c_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ \ldots + a_{2n}x_n = c_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ \ldots + a_{3n}x_n = c_3$$
$$\ldots$$
$$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ \ldots + a_{mn}x_n = c_m$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix}$$

more general:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix}}_{\vec{c}}$$

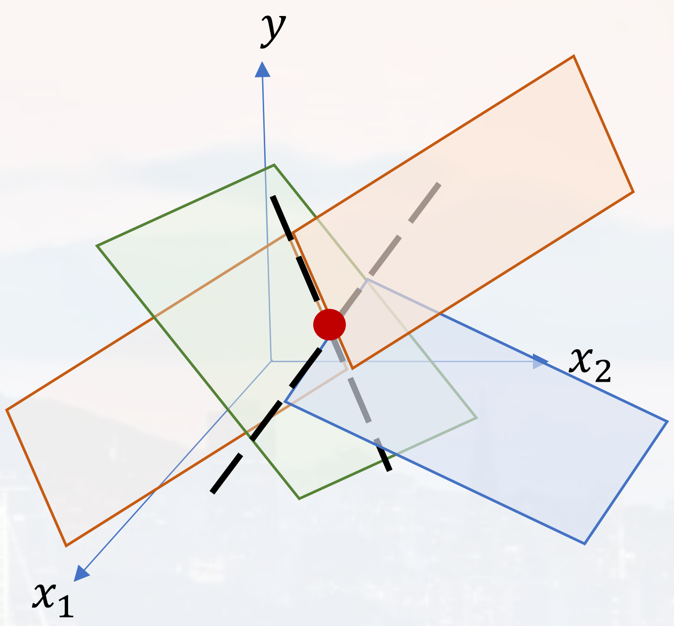


$$\boxed{A\vec{x} = \vec{c}}$$

more general:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix}}_{\vec{c}}$$

$$\boxed{A\vec{x} = \vec{c}}$$



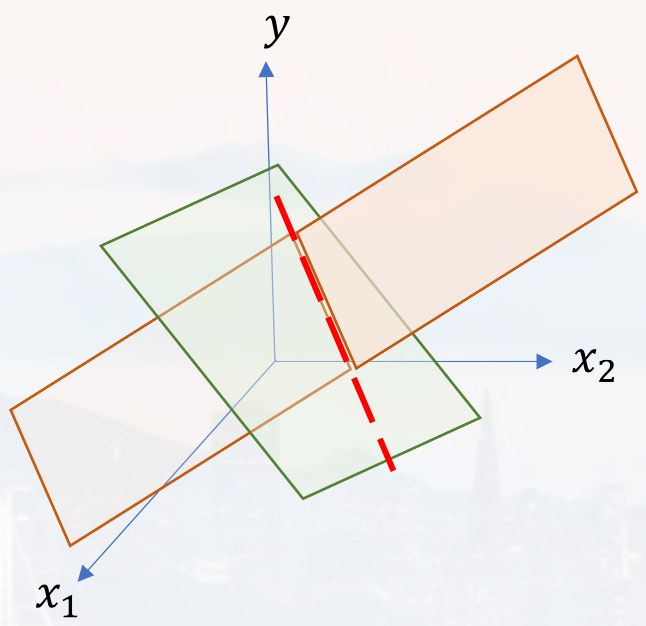
general set of solutions

for n = m → solution is unique: a point

more general:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix}}_{\vec{c}}$$

$$\boxed{A\vec{x} = \vec{c}}$$



general set of solutions

for n = m → solution is unique: a point

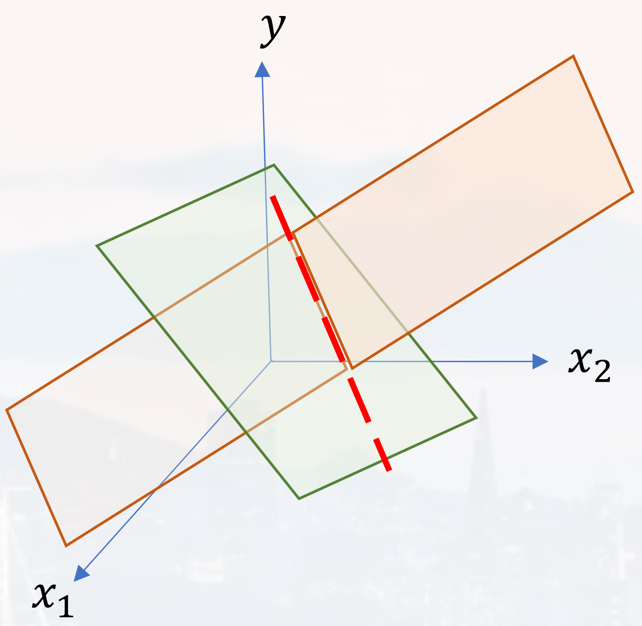
for n > m (more variables than equations)
→ solution is not unique: line, hyperplane

for n < m (more equations than variables)
→ no solution

<u>more general:</u>

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix}}_{\vec{c}}$$

$$\boxed{A\vec{x} = \vec{c}}$$



<u>**general** set of solutions</u>

for n = m → solution is unique: a point

for n > m (more variables than equations)
→ solution is not unique: line, hyperplane

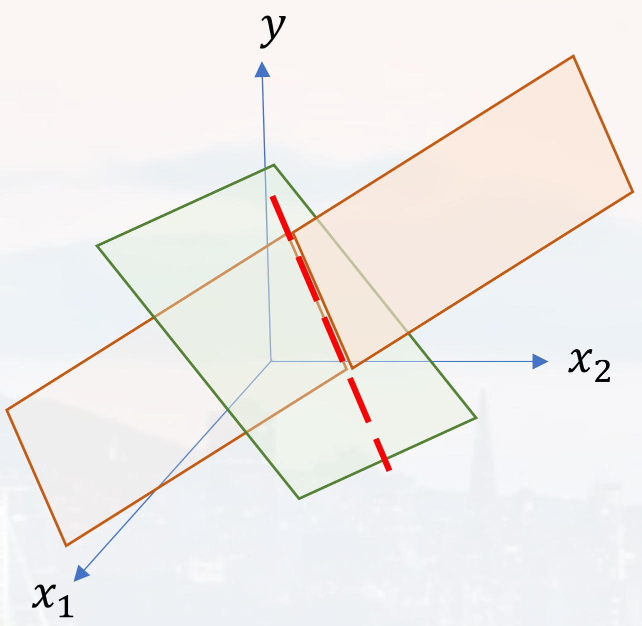
for n < m (more equations than variables)
→ no solution

exceptions!

more general:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix}}_{\vec{c}}$$

$$\boxed{A\vec{x} = \vec{c}}$$



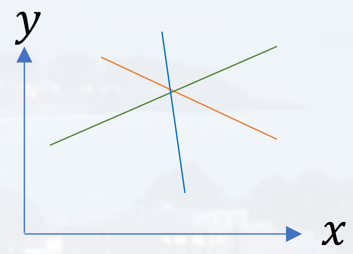
general set of solutions

for n = m → solution is unique: a point

for n > m (more variables than equations)
→ solution is not unique: line, hyperplane

for n < m (more equations than variables)
→ no solution

exceptions!



more general:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}}_{A} \underset{\vec{x}}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}} = \underset{\vec{c}}{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_m \end{bmatrix}}$$

$$\boxed{A\vec{x} = \vec{c}}$$

$\vec{x}$ = ?

for n = m

$$A^{-1}A\vec{x} = A^{-1}\vec{c}$$

$$\vec{x} = A^{-1}\vec{c}$$

$A = [a_{ij}]$

inverse:	$A^{-1}A = \mathrm{I}$
identity:	$I\ M = M$
transpose:	$[a_{ij}]^T = [a_{ji}]$
symmetry:	$[a_{ij}] = [a_{ji}]$
conjugate transpose:	A^{+}
unitary:	$A^{-1} = A^{+}$
idempotency:	$AA = A \rightarrow\ A^n = A$
normal:	$A^{+}A = AA^{+}$

solving for x:

$$\boxed{A\vec{x} = \vec{c}}$$

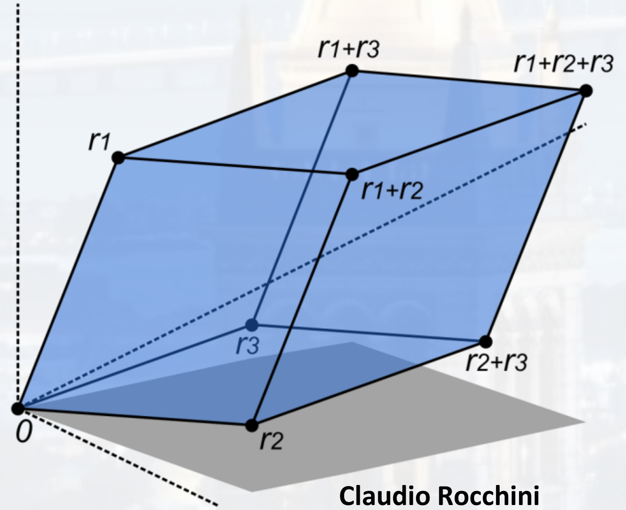
→ need to calculate A^{-1}
→ need to calculate a quantity called

determinant of A, *det(A)*

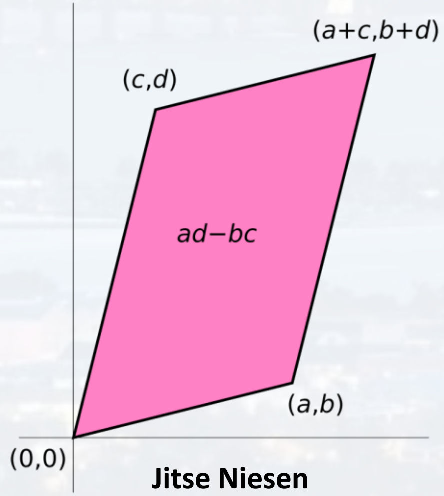
$$A^{-1} \sim \frac{1}{\det(A)}$$

- if *det(A) = 0* → no solution
- *|det(A)|*: volume spanned by the vectors in A

inverse:	$A^{-1}A = \mathrm{I}$
identity:	$I\ M = M$
transpose:	$[a_{ij}]^T = [a_{ji}]$
symmetry:	$[a_{ij}] = [a_{ji}]$
conjugate transpose:	A^{+}
unitary:	$A^{-1} = A^{+}$
idempotency:	$AA = A \rightarrow\ A^n = A$
normal:	$A^{+}A = AA^{+}$



Claudio Rocchini



Jitse Niesen

$$\boxed{A\vec{x} = \vec{c}}$$

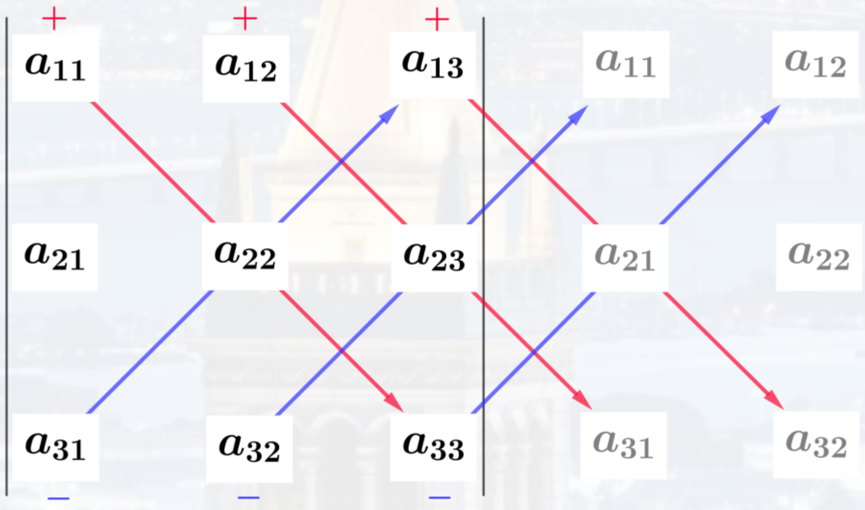
solving for x:

→ need to calculate A^{-1}
→ need to calculate a quantity called

determinant of A, *det(A)*

$$A^{-1} \sim \frac{1}{\det(A)}$$

inverse:	$A^{-1}A = \mathrm{I}$
identity:	$I\,M = M$
transpose:	$[a_{ij}]^T = [a_{ji}]$
symmetry:	$[a_{ij}] = [a_{ji}]$
conjugate transpose:	A^+
unitary:	$A^{-1} = A^+$
idempotency:	$AA = A \rightarrow\ A^n = A$
normal:	$A^+A = AA^+$



Kmhkmh

$$\det(A) = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{31}a_{22}a_{13} - a_{32}a_{23}a_{11} - a_{33}a_{21}a_{12}$$

solving for x:

$$\boxed{A\vec{x} = \vec{c}}$$

inverse:	$A^{-1}A = \mathrm{I}$
identity:	$I\,M = M$
transpose:	$[a_{ij}]^T = [a_{ji}]$
symmetry:	$[a_{ij}] = [a_{ji}]$
conjugate transpose:	A^+
unitary:	$A^{-1} = A^+$
idempotency:	$AA = A \rightarrow A^n = A$
normal:	$A^+A = AA^+$

→ need to calculate A^{-1}
→ need to calculate a quantity called

determinant of A, *det(A)*

$$A^{-1} \sim \frac{1}{\det(A)}$$

N x N matrix:

$$\det(A) = \sum_{i_1, i_2, \dots, i_n} \varepsilon_{i_1 \dots i_n} a_{1,i_1} \dots a_{n\,i_n}$$

where

$$\varepsilon_{i_1 \dots i_n} = \boxed{\prod_{1 \le \mu < \vartheta \le n} \operatorname{sgn}(i_\vartheta - i_\mu)}$$

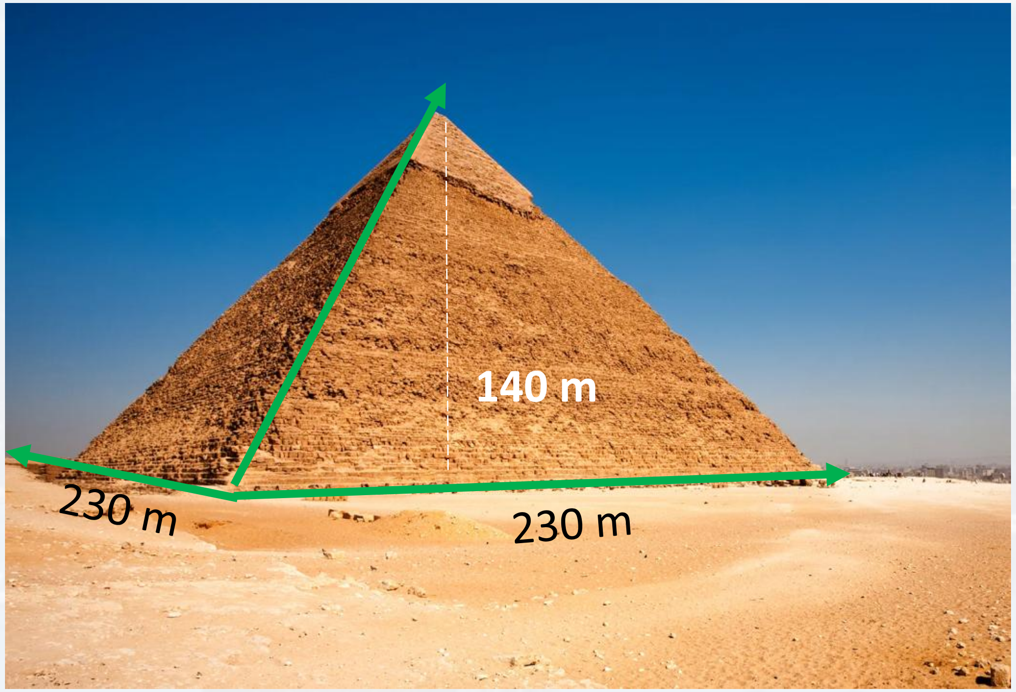
(Levi-Civita symbol)

changing indices
does not change |det(A)|

determinant of A, *det(A)*

$$\boxed{A\vec{x} = \vec{c}}$$

inverse:	$A^{-1}A = \mathrm{I}$
identity:	$I\ M = M$
transpose:	$[a_{ij}]^T = [a_{ji}]$
symmetry:	$[a_{ij}] = [a_{ji}]$
conjugate transpose:	A^{+}
unitary:	$A^{-1} = A^{+}$
idempotency:	$AA = A \rightarrow\ A^n = A$
normal:	$A^{+}A = AA^{+}$



$$\varepsilon_{i_1 \dots i_n} = \prod_{1 \le \mu < \vartheta \le n} \operatorname{sgn}(i_\vartheta - i_\mu)$$

$$V = \left| \det\begin{pmatrix} 230 & 0 & 115 \\ 0 & 230 & 115 \\ 0 & 0 & 140 \end{pmatrix} \right| \frac{1}{3} = \frac{230 * 230 * 140 + 0 + 0\ -\ 0\ - 0 - 0}{3} = 2{,}468{,}666\ m^3$$

determinant of A, *det(A)*

$$\varepsilon_{i_1 \dots i_n} = \prod_{1 \le \mu < \vartheta \le n} \operatorname{sgn}(i_\vartheta - i_\mu)$$

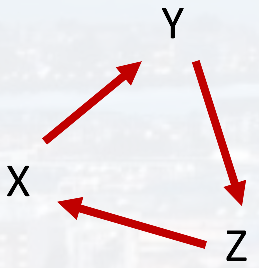
$$V = \left| \det \begin{pmatrix} 230 & 0 & 115 \\ 0 & 230 & 115 \\ 0 & 0 & 140 \end{pmatrix} \right| \frac{1}{3} = \frac{230 * 230 * 140 + 0 + 0 - \ 0 - 0 - 0}{3} = 2{,}468{,}666\ m^3$$

volume does not depend on **where** I put my **coord origin**…

$$V = \left| \det \begin{pmatrix} 0 & 230 & 115 \\ 230 & 0 & 115 \\ 0 & 0 & 140 \end{pmatrix} \right| \frac{1}{3} = \left| \frac{0 + 0 + 0 - \ 140 * 230 * 230 - 0 - 0}{3} \right| = 2{,}468{,}666\ m^3$$

…or how I **turn** the object!

$$V = \left| \det \begin{pmatrix} 115 & 230 & 0 \\ 115 & 0 & 230 \\ 140 & 0 & 0 \end{pmatrix} \right| \frac{1}{3} = \left| \frac{0 + 230 * 230 * 140 + 0 - \ 0 - 0 - 0}{3} \right| = 2{,}468{,}666\ m^3$$



$$V = \left| \det \begin{pmatrix} 140 & 0 & 0 \\ 115 & 230 & 0 \\ 115 & 0 & 230 \end{pmatrix} \right| \frac{1}{3} = \left| \frac{140 * 230 * 230 + 0 + 0 - \ 0 - 0 - 0}{3} \right| = 2{,}468{,}666\ m^3$$

linear regression

Goal 1: finding a model that tells us how we can **predict y_k from all the x_i**

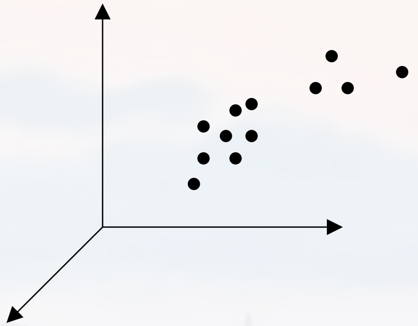
	x_1	x_2	x_3	x_4	x_5	y_k
Index	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP	toxicity_score
0	341.704	2.65585	3.09407	2	9.11147	80.9281
1	335.951	3.22262	2.89039	7	8.92848	83.4911
2	235.203	2.44115	2.48203	1	6.49731	61.8406
3	246.505	2.76656	2.71547	7	7.45089	57.0538
4	437.939	3.4801	3.59569	3	10.9156	131.326
	336.453	2.81474	3.11	9	8.55696	?

kth row

Goal 2: once we have a model: **predict y_k from new data set**

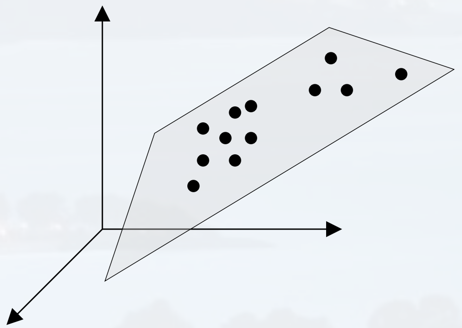
linear regression

idea: data point y_k in N dimensional space



$\rightarrow y_k = f(x_1, \ldots x_n, \ldots x_N) + \epsilon$ for each data point k

ansatz: $\boxed{y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon}$ *linear* combination



y: response
x: regressors **(assumed to be independent)**
β: factors **(how a regressor contributes to the response)**
β_0: intercept
ε: error **(stochasticity of the data, assumed to be normally dist.)**

Finding β_n is the model!

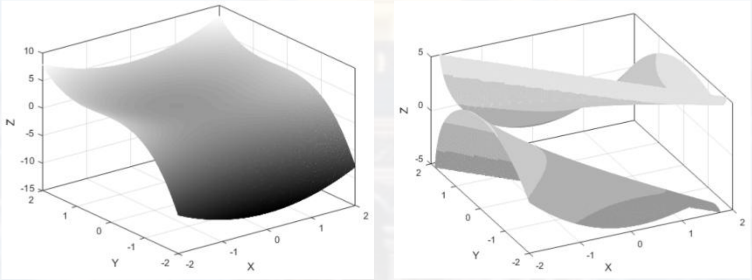
y:	**response**
x:	**regressors**
β:	**factors**
$\boldsymbol{\beta_0}$**:**	**intercept**
ε:	**error**

general: linear refers to the **factors**

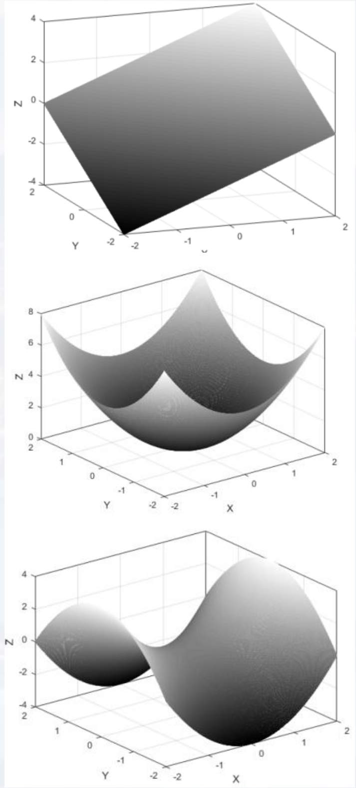
$$y_k = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$ 2*D* plane in 3*D* space

$$y_k = \beta_0 + \beta_1 {x_1}^2 + \beta_2 {x_2}^2$$ 2*D* parabolic

$$y_k = \beta_0 + \beta_1 {x_1}^2 - \beta_2 {x_2}^2$$ 2*D* hyperbolic



...and many more...

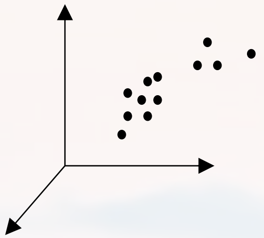


all linear

$$\boldsymbol{y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon}$$

for *K* data points in *N* dimensional space

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$



y:	**response**
x:	**regressors**
β:	**factors**
$\boldsymbol{\beta_0}$**:**	**intercept**
ε:	**error**

$$\underbrace{\begin{pmatrix} y_1 \\ \dots \\ y_k \\ \dots \\ \dots \\ y_K \end{pmatrix}}_{\text{Y}} = \underbrace{\begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} & \dots & x_{1N} \\ \dots & \dots & & & \dots & & \\ 1 & x_{k1} & & & x_{kn} & & \\ 1 & \dots & & & \dots & & \\ 1 & x_{K1} & x_{K2} & \dots & x_{Kn} & \dots & x_{KN} \end{pmatrix}}_{\text{X}} \underbrace{\begin{pmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_n \\ \dots \\ \beta_N \end{pmatrix}}_{\beta} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \dots \\ \varepsilon_k \\ \dots \\ \varepsilon_K \end{pmatrix}}_{\varepsilon}$$

$$\boxed{Y = X\beta + \varepsilon}$$

<u>fitting:</u> finding the best β in terms of minimizing the errors

$$(Y - X\beta)^T (Y - X\beta) = \sum_k {\varepsilon_k}^2$$

the model

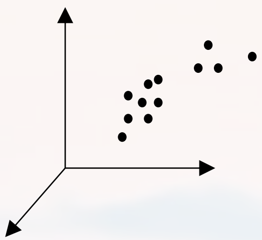
$$\frac{\partial}{\partial \beta} \sum_k {\varepsilon_k}^2 = 0 \longrightarrow \beta_{best} = \hat{\beta} = (X^T X)^{-1} X^T Y \longrightarrow \boldsymbol{\hat{Y} = X\hat{\beta}} = X(X^T X)^{-1} X^T Y$$

X and Y are all observables

for *K* data points in *N* dimensional space

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$

$$\boxed{Y = X\beta + \varepsilon}$$



y: response
x: regressors
β: factors
β_0: intercept
ε: error

check out `Walk_Through_LinRegression.ipynb`

predicting **toxicity of molecules** based on their features

Index	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP	toxicity_score
0	341.704	2.65585	3.09407	2	9.11147	80.9281
1	335.951	3.22262	2.89039	7	8.92848	83.4911
2	235.203	2.44115	2.48203	1	6.49731	61.8406
3	246.505	2.76656	2.71547	7	7.45089	57.0538
4	437.939	3.4801	3.59569	3	10.9156	131.326

see Walk_Through_LinRegression.ipynb

```python
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pylab
import scipy.stats as stats
import statsmodels.api as sm

from statsmodels.formula.api import ols
from sklearn.preprocessing import MinMaxScaler
```

- pandas: reading .xlsx .csv .txt …
- matplotlib.pyplot: standard plots
- seaborn: fancy plots: here a pair-plot
- pylab, scipy.stats: Q-Q plot
- statsmodels.api, statsmodels.formula.api: the actual super tool for superb data analysis
- sklearn.preprocessing: scaling and normalizing

1) loading data
2) plotting data
3) scaling data
4) fitting model
5) evaluating model

```
Train  = pd.read_csv("molecular_train_gbc.csv")

Test   = pd.read_csv("molecular_test_gbc.csv")
```

	x_1	x_2	x_3	x_4	x_5	y_k
Index	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP	toxicity_score
0	341.704	2.65585	3.09407	2	9.11147	80.9281
1	335.951	3.22262	2.89039	7	8.92848	83.4911
2	235.203	2.44115	2.48203	1	6.49731	61.8406
3	246.505	2.76656	2.71547	7	7.45089	57.0538
4	437.939	3.4801	3.59569	3	10.9156	131.326

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$

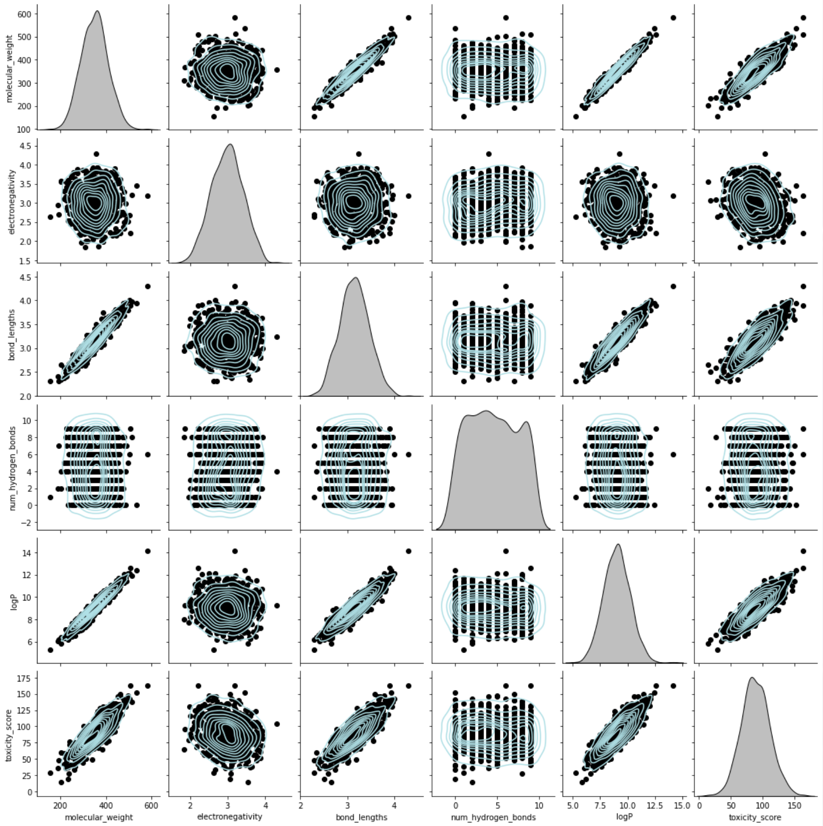
y: toxicity_score

x_n : molecular_weight, electronegativity, bond_lengths, num_hydrogen_bonds, logP

```
Train  = pd.read_csv("molecular_train_gbc.csv")
Test   = pd.read_csv("molecular_test_gbc.csv")


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

1) loading data
2) plotting data
3) scaling data
4) fitting model
5) evaluating model



```
Train  = pd.read_csv("molecular_train_gbc.csv")
Test   = pd.read_csv("molecular_test_gbc.csv")


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

Scaling the data because unit system is arbitrary!

Large numerical values dominate the optimization! → rescaling!

molecular_weight	electronegativity	bond_lengths
341.704	2.65585	3.09407

```
scaler  = MinMaxScaler(feature_range = (0, 1))
TrainS  = scaler.fit_transform(Train)
TestS   = scaler.transform(Test)
```

the scaler returns an `np.array`
→ convert back to data `frame`

```
TrainS = pd.DataFrame(TrainS, columns = Train.columns)
TestS  = pd.DataFrame(TestS, columns = Train.columns)
```

```
TrainS = pd.DataFrame(TrainS, columns = Train.columns)
TestS  = pd.DataFrame(TestS,  columns = Train.columns)
```

```
equation = 'toxicity_score ~ ' + '+'.join(Train.columns[:-1])
print(equation)
```

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$

```
toxicity_score ~      molecular_weight + electronegativity +
                      bond_lengths + num_hydrogen_bonds + logP
```

```
my_model = ols(equation, data = TrainS).fit()
my_model.summary()
```

OLS (ordinary least squares)

1) loading data
2) plotting data
3) scaling data
4) fitting model
5) evaluating model

```
my_model.summary()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          toxicity_score   R-squared:                       0.790
Model:                             OLS   Adj. R-squared:                  0.789
Method:                  Least Squares   F-statistic:                     597.5
Date:                 Fri, 13 Sep 2024   Prob (F-statistic):          3.34e-266
Time:                         20:57:10   Log-Likelihood:                 1013.0
No. Observations:                  800   AIC:                            -2014.
Df Residuals:                      794   BIC:                            -1986.
Df Model:                            5
Covariance Type:             nonrobust
=======================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
Intercept               0.1494      0.012     12.533      0.000       0.126       0.173
molecular_weight        0.7961      0.089      8.982      0.000       0.622       0.970
electronegativity      -0.1682      0.015    -11.591      0.000      -0.197      -0.140
bond_lengths            0.0204      0.049      0.417      0.677      -0.076       0.116
num_hydrogen_bonds      0.0035      0.008      0.458      0.647      -0.011       0.018
logP                    0.1246      0.072      1.723      0.085      -0.017       0.267
==============================================================================
Omnibus:                         2.249   Durbin-Watson:                   1.984
Prob(Omnibus):                   0.325   Jarque-Bera (JB):                2.240
Skew:                           -0.129   Prob(JB):                        0.326
Kurtosis:                        2.980   Cond. No.                         65.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

not the fit quality!

p-value for constant model

p-values for factors (should be < 0.01)

2σ conf range of factors

number of data points is much larger than the number of regressors → degree of freedom approx. no of obs

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$

more accurate: determining **the p-values for the factors using ANOVA** for the corresponding residuals

```
table    = sm.stats.anova_lm(my_model, typ = 1)
print(table)
```

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$

	df	sum_sq	mean_sq	F	PR(>F)
molecular_weight	1.0	13.346285	13.346285	2847.525516	8.024085e-265
electronegativity	1.0	0.640388	0.640388	136.631363	3.085962e-29
bond_lengths	1.0	0.000684	0.000684	0.145954	7.025342e-01
num_hydrogen_bonds	1.0	0.000703	0.000703	0.150055	6.985866e-01
logP	1.0	0.013917	0.013917	2.969353	8.524510e-02
Residual	794.0	3.721459	0.004687	NaN	NaN

vs from t-test

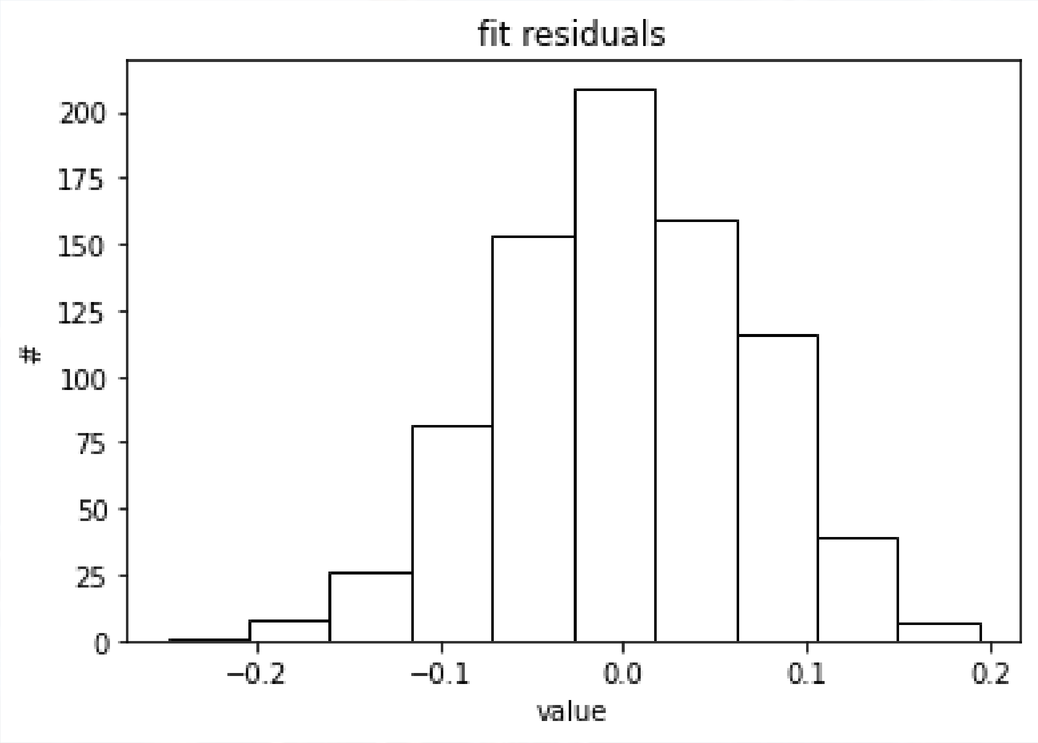
0.0000
0.0000
0.6766
0.6473
0.0852

```
residuals = my_model.resid

plt.hist(residuals, color = 'w', edgecolor = 'black')
plt.title('fit residuals')
plt.ylabel('#')
plt.xlabel('value')
plt.show()
```

1) loading data
2) plotting data
3) scaling data
4) fitting model
5) evaluating model

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$



residuals approx. normally distributed around μ = 0

1) loading data
2) plotting data
3) scaling data
4) fitting model
5) evaluating model

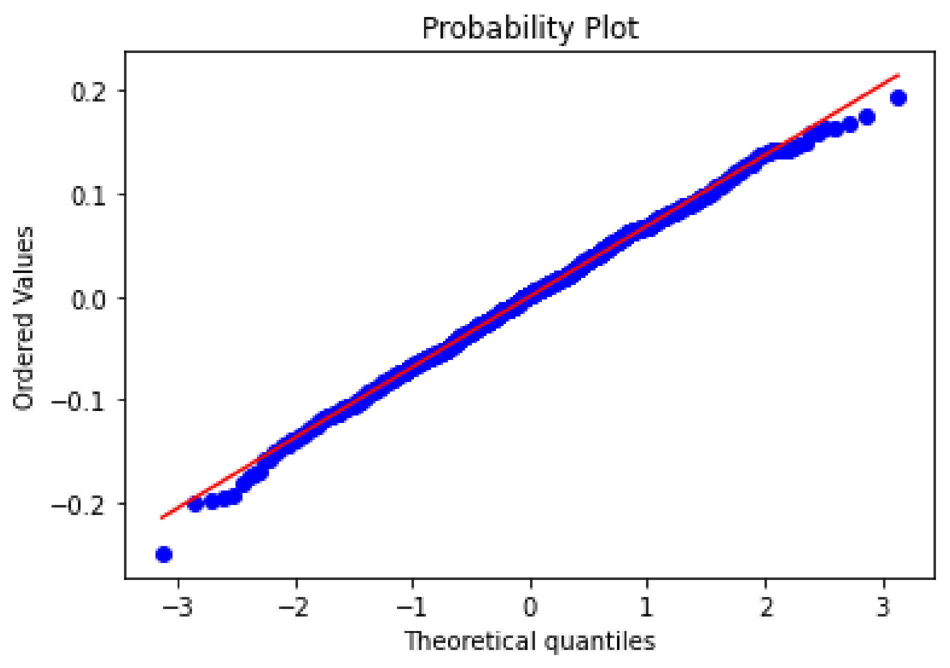
$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$

```
residuals = my_model.resid

plt.hist(residuals, color = 'w', edgecolor = 'black')
plt.title('fit residuals')
plt.ylabel('#')
plt.xlabel('value')
plt.show()

stats.probplot(residuals, dist = "norm", plot = pylab)
pylab.show()
```

residuals approx. normally distributed around μ = 0



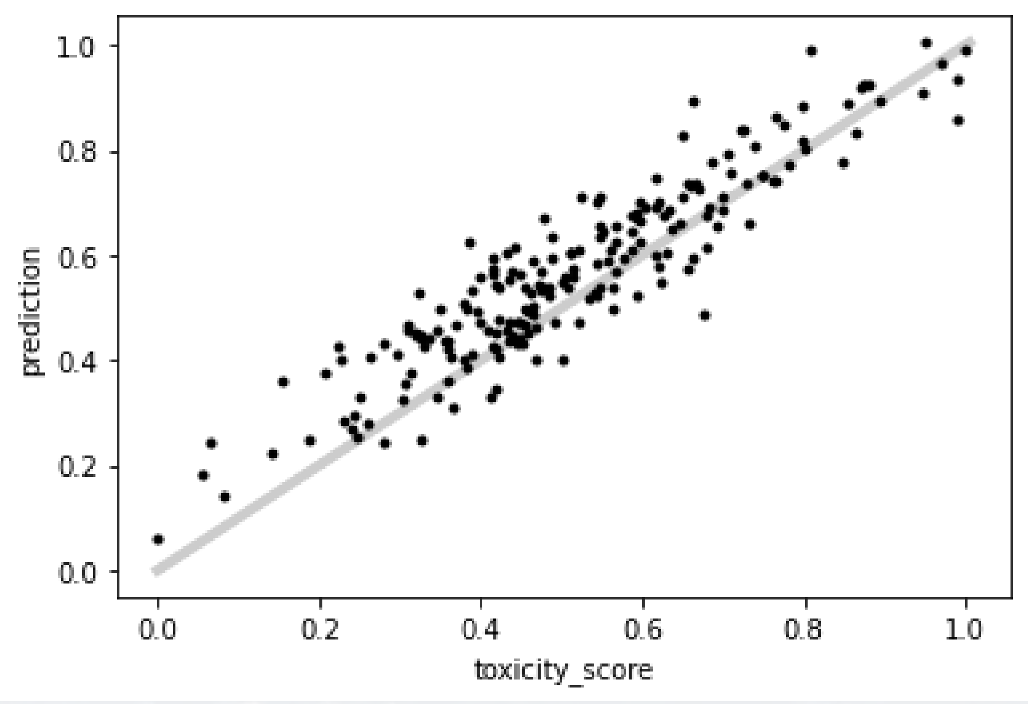
```
Ypred  = my_model.predict(TestS)

higher = np.max([Ypred, TestS.toxicity_score])
lower  = np.min([Ypred, TestS.toxicity_score])

plt.plot([lower, higher], [lower, higher], c = [0, 0, 0, 0.2],\
         linewidth = 4)
plt.scatter(TestS.toxicity_score, Ypred, marker = '.', c = 'k')
plt.ylabel('prediction')
plt.xlabel('toxicity score')
plt.show()
```

1) loading data
2) plotting data
3) scaling data
4) fitting model
5) evaluating model

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$



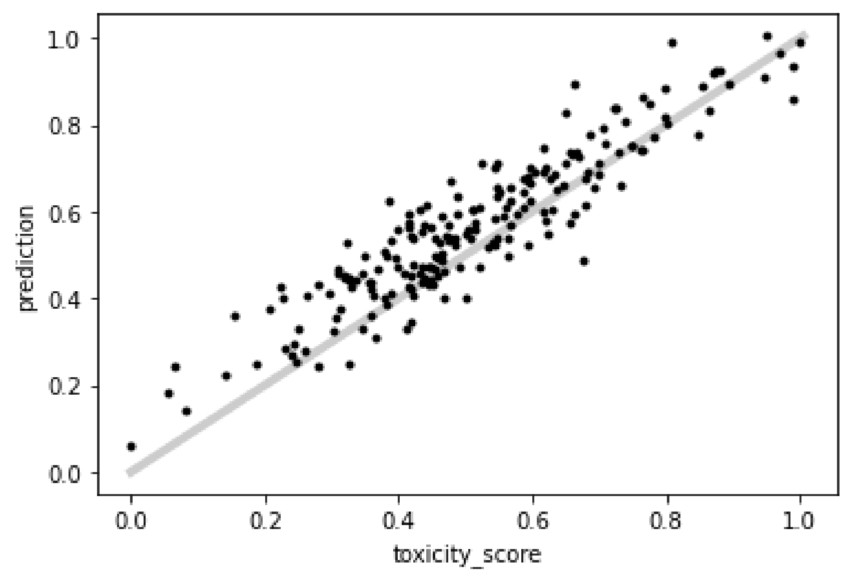
```
Ypred  = my_model.predict(TestS)

higher = np.max([Ypred, TestS.toxicity_score])
lower  = np.min([Ypred, TestS.toxicity_score])

plt.plot([lower, higher], [lower, higher], c = [0, 0, 0, 0.2], linewidth = 4)
plt.scatter(TestS.toxicity_score, Ypred, marker = '.', c = 'k')
plt.ylabel('prediction')
plt.xlabel('toxicity score')
plt.show()
```

1) loading data
2) plotting data
3) scaling data
4) fitting model
5) evaluating model

$$y_k = \beta_0 + \sum_{n=1}^{N} \beta_n x_n + \epsilon$$

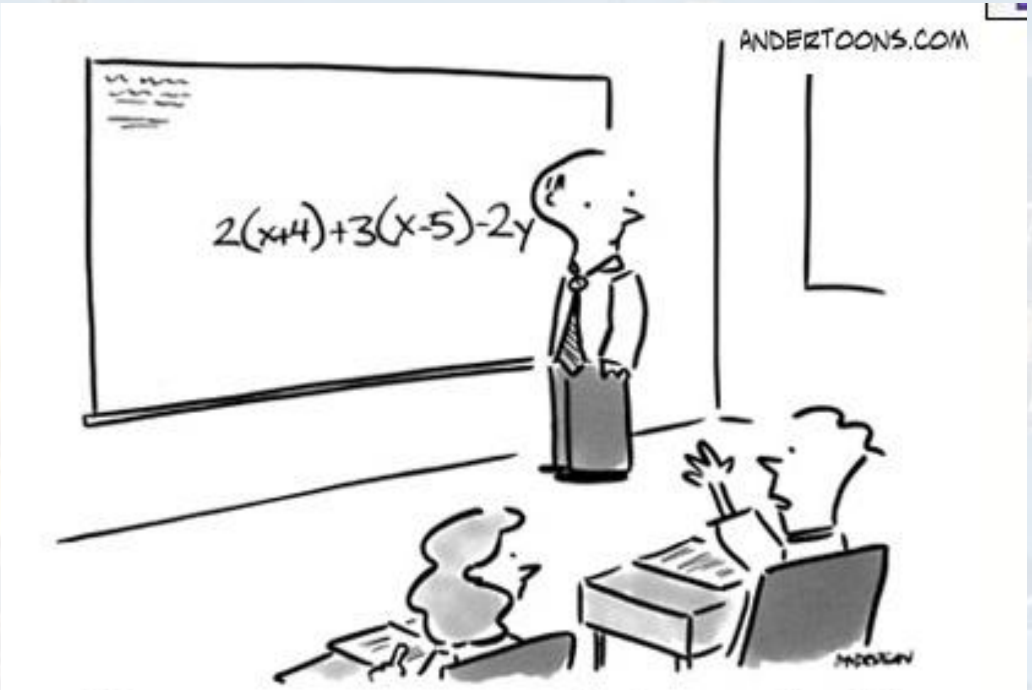



```
mean_dev = np.sum( abs(TestS.toxicity_score - Ypred) )/len(Ypred)
print(mean_dev)
```

5%

M. Hohle:

Thank you for your attention!



"How come we never complicate equations?"