Intermezzo

- **for next HW assignment:** variance, covariance, correlation
- **for next HW assignment:** Principal Component Analysis (PCA)
- regression table

Intermezzo

- **for next HW assignment:** variance, covariance, correlation
- **for next HW assignment:** Principal Component Analysis (PCA)
- regression table

Intermezzo: variance, covariance, correlation

recap:

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

$$\sigma^2 = var(x) = \int (x-\mu)^2\, p(x)\, dx$$

$$var(x) = \int (x-\mu)^2\, p(x)\, dx = E([x-\mu]^2)$$

variance can be interpreted as **mean of $[x-\mu]^2$**

$$= E(x^2 - 2x\mu + \mu^2)$$

$$= \int [x^2 - 2x\mu + \mu^2]\, p(x)\, dx$$

$$= \int x^2\, p(x)\, dx - 2\mu \int x\, p(x)\, dx + \mu^2 \int p(x)\, dx$$

$$= E(x^2) - 2\mu\, E(x) + \mu^2\, E(1)$$

$$\int p(x)\, dx = 1$$

$$= E(x^2) - 2\mu\, E(x) + \mu^2$$

$$\mu = E(x)$$

$$\boldsymbol{\sigma^2 = E(x^2) - E(x)^2}$$

Intermezzo:

recap:

a, b = const

$$\boldsymbol{\sigma^2 = E(x^2) - E(x)^2}$$

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

$$\sigma^2 = var(x) = \int (x-\mu)^2\, p(x)\, dx$$

$$var([a\, x_1 + b\, x_2]) = E([a\, x_1 + b\, x_2]^2) - E(a\, x_1 + b\, x_2)^2$$

$$= E(a^2\, x_1^2 + 2ab\, x_1 x_2 + b^2\, x_2^2) - E(a\, x_1 + b\, x_2)^2$$

$$= a^2 E(x_1^2) + 2ab\, E(x_1 x_2) + b^2\, E(x_2^2) - E(a\, x_1 + b\, x_2)^2$$

$$= a^2 E(x_1^2) + 2ab\, E(x_1 x_2) + b^2\, E(x_2^2) - [aE(x_1) + b\, E(x_2)]^2$$

$$= a^2 E(x_1^2) - a^2 E(x_1)^2 + b^2\, E(x_2^2) - b^2 E(x_2)^2 + 2ab\, E(x_1 x_2) - 2abE(x_1)E(x_2)$$

$$\boldsymbol{a^2\, var(x_1)} \qquad \boldsymbol{b^2\, var(x_2)} \qquad \boldsymbol{2ab\, cov(x_1, x_2)}$$

$$= \boldsymbol{a^2\, var(x_1) + b^2\, var(x_2) + 2ab\, cov(x_1, x_2)}$$

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

covariance

Intermezzo: variance, covariance, correlation

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

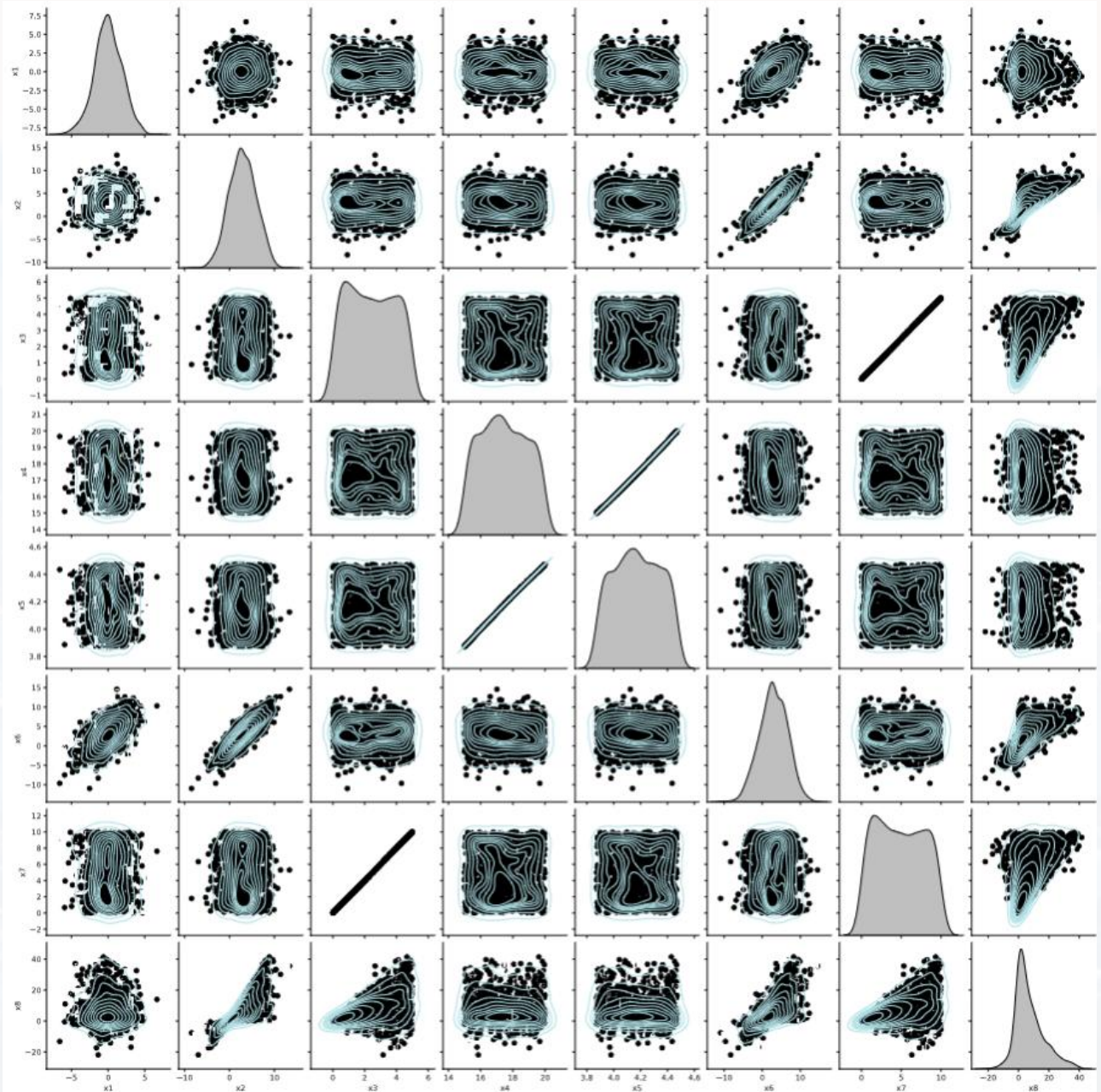
$$\mu = E(x) = \int x\, p(x)\, dx$$

$$\sigma^2 = var(x) = \int (x-\mu)^2\, p(x)\, dx$$

```
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')
```

Intermezzo:

variance, covariance, correlation



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

$$\sigma^2 = var(x) = \int (x-\mu)^2\, p(x)\, dx$$

based on the shape of the data cloud

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

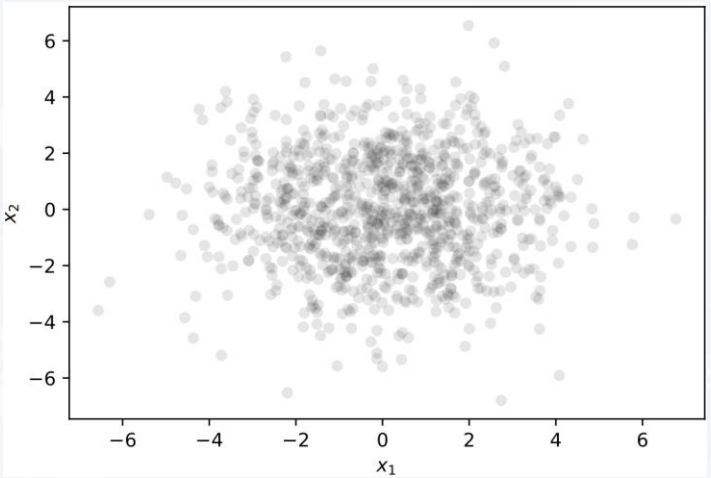
→ how to quantify?

Intermezzo: variance, covariance, correlation

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

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

$$\sigma^2 = var(x) = \int (x - \mu)^2\, p(x)\, dx$$



a) x_1 and x_2 are independent

$$E(x_1 x_2) - E(x_1)E(x_2)$$

$$= \iint x_1 x_2\, p(x_1)\, p(x_2)\, dx_1 dx_2 \; - \int x_1\, p(x_1)\, dx_1 \int x_2\, p(x_2)\, dx_2$$

x_1 and x_2 are independent:

x_1 is not a function of x_2 and vise verse
x_1 cannot be predicted by x_2 and vise verse

$$= \int x_1\, p(x_1)\, dx_1 \int x_2\, p(x_2)\, dx_2 - \int x_1\, p(x_1)\, dx_1 \int x_2\, p(x_2)\, dx_2 = 0$$

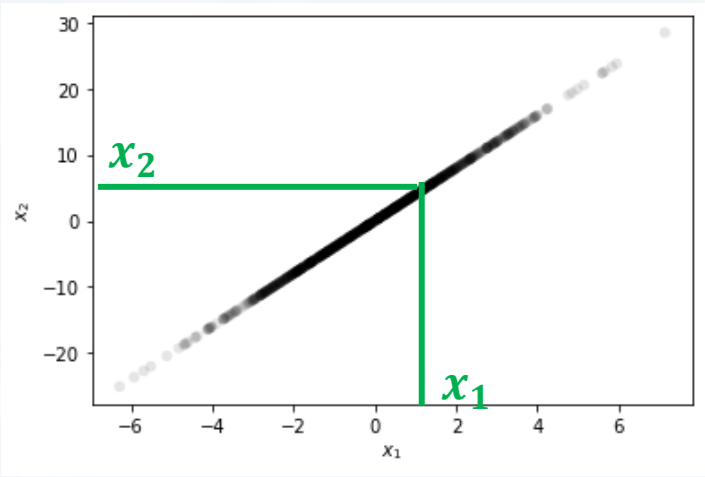
covariance equals **zero**
if samples are **independent**!

Intermezzo: variance, covariance, correlation

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

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

$$\sigma^2 = var(x) = \int (x - \mu)^2\, p(x)\, dx$$



b) x_1 and x_2 are **not** independent

$$E(x_1 x_2) - E(x_1)E(x_2)$$

$$= \iint x_1 x_2\, p(x_1)\, p(x_2)\, dx_1 dx_2 \; - \int x_1\, p(x_1)\, dx_1 \int x_2\, p(x_2)\, dx_2$$

x_1 and x_2 are **not** independent:

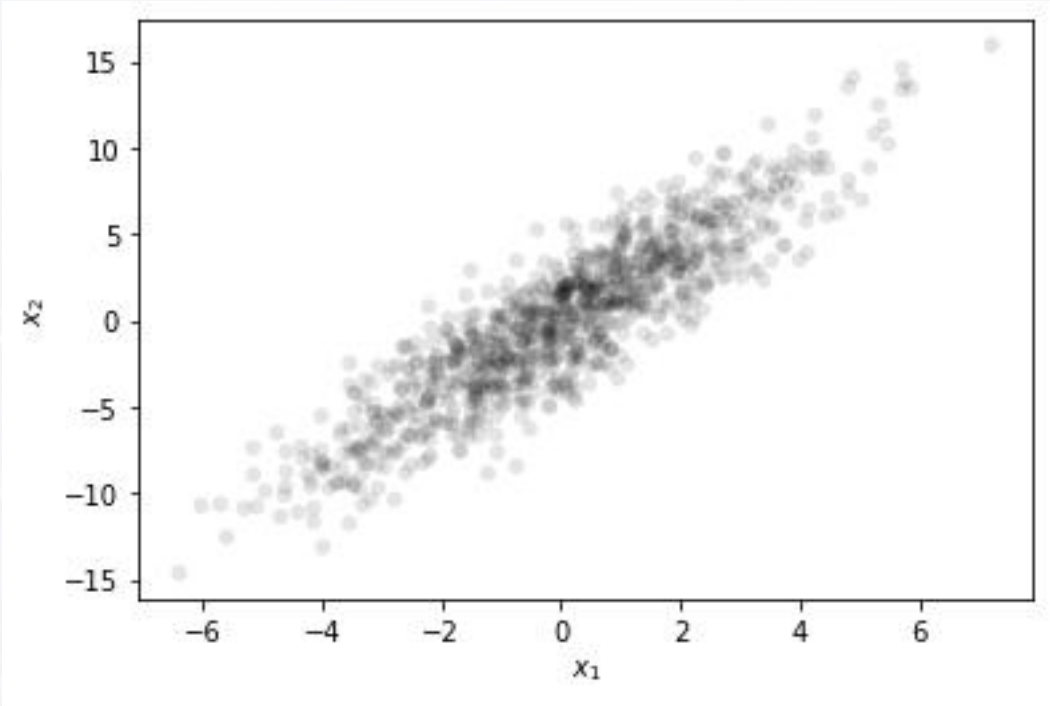
x_1 ***is*** a function of x_2 and vise verse
x_1 ***can*** be predicted by x_2 to certain degree and vise verse

$$= \iint x_1\, p(x_1)\;\; x_2(x_1)\, p(x_2(x_1))\, dx_1 dx_2(x_1) - \int x_1\, p(x_1)\, dx_1 \int x_2\, p(x_2)\, dx_2$$

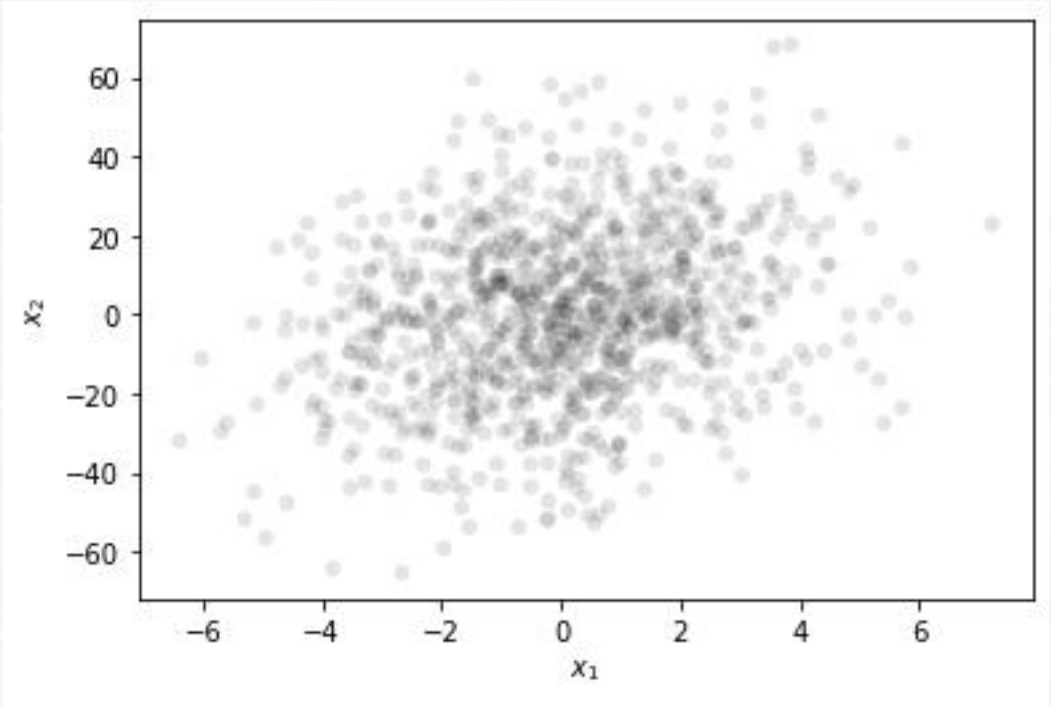
covariance **does not** equal **zero**!

$$cov(x_1, x_2) = cov(x_2, x_1) = E(x_1 x_2) - E(x_1)E(x_2)$$ **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,))
```

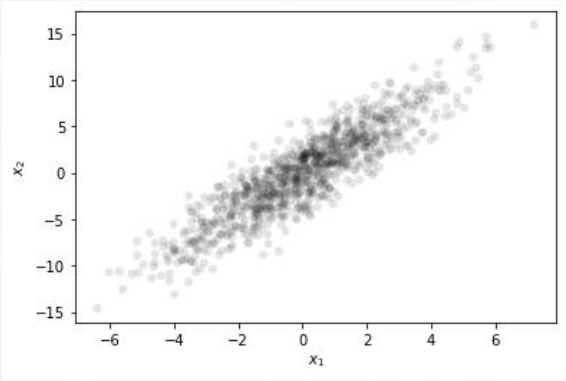


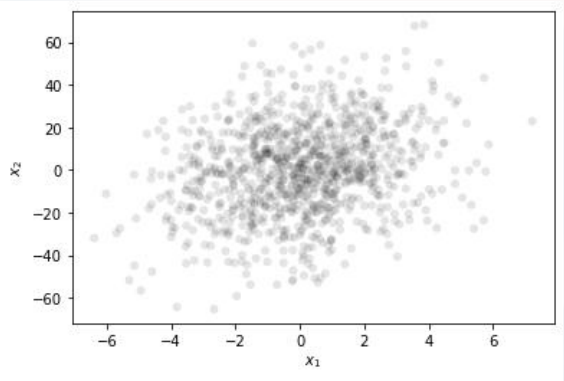
Same dependency, but different variance!

Intermezzo: variance, covariance, correlation

$$cov(x_1, x_2) = cov(x_2, x_1) = E(x_1 x_2) - E(x_1)E(x_2)$$

covariance





Same dependency, but different variance!

Need to scale for the variance!

Pearson's correlation coefficient

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

$\rho(x_1, x_2)$:

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

Intermezzo: variance, covariance, correlation

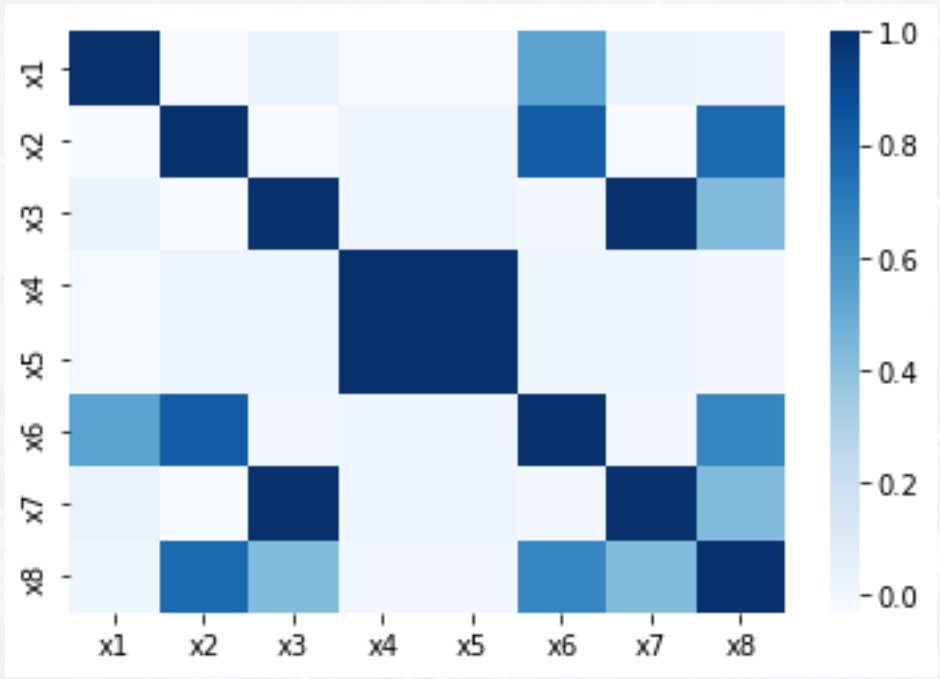
$$cov(x_1, x_2) = cov(x_2, x_1) = E(x_1 x_2) - E(x_1)E(x_2)$$

covariance

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

Pearson's correlation coefficient

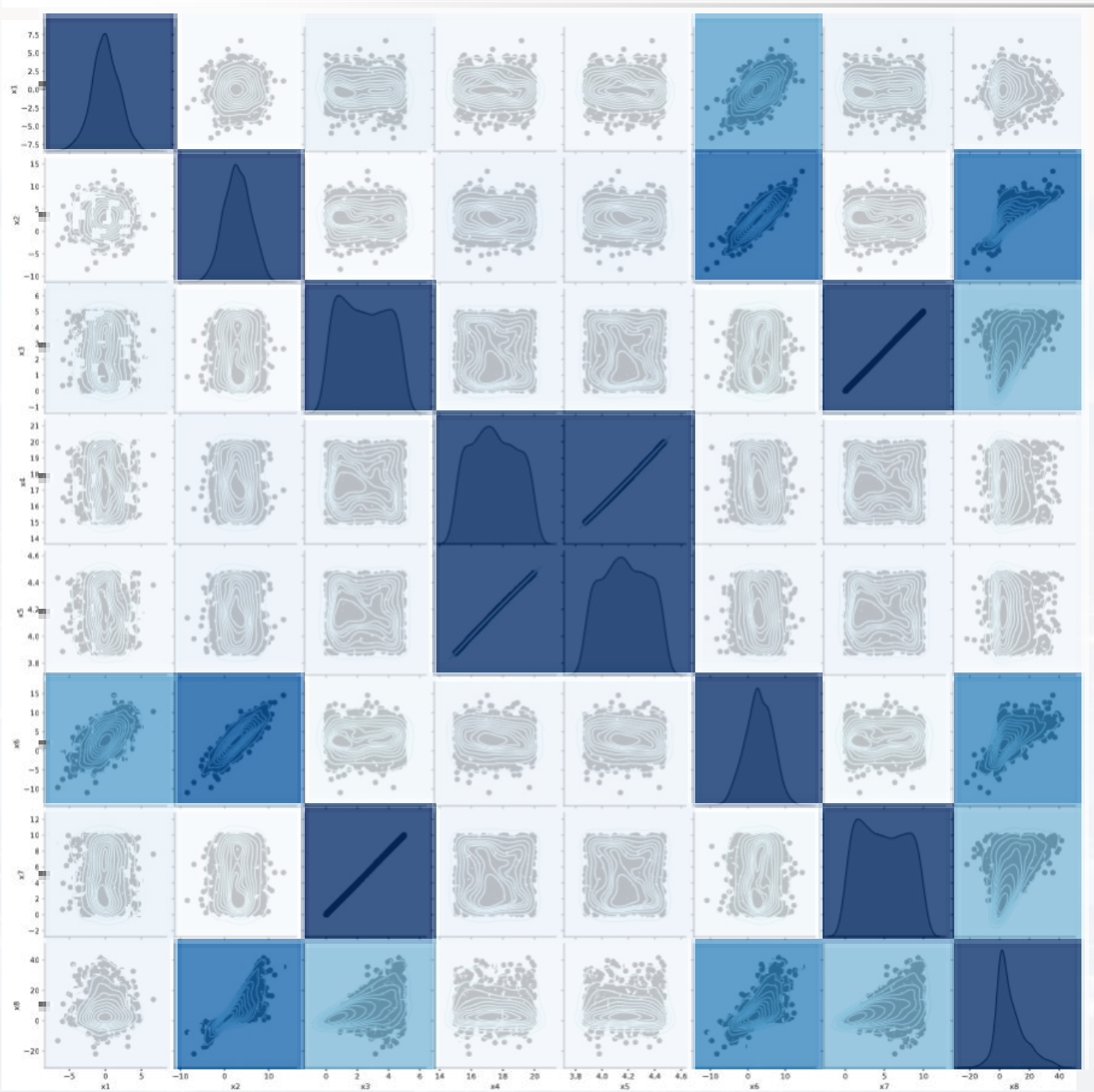
```
sns.heatmap(data.corr(), cmap = "Blues")
```



$\rho(x_1, x_2)$:

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

Intermezzo: variance, covariance, correlation



$\rho(x_1, x_2)$:

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

Intermezzo: variance, covariance, correlation

Important quantities you should know:

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

median *m* $\int_a^m p(x)\, dx = \frac{1}{2}$

variance $\sigma^2 = var(x) = \int (x-\mu)^2\, p(x)\, dx$

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

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

covariance $cov(x_1, x_2) = E(x_1 x_2) - E(x_1)E(x_2)$

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

note:

$\int (x-\mu)^n\, p(x)\, dx$

called n-th *moment* of a pdf

Intermezzo

- **for next HW assignment:** variance, covariance, correlation

- **for next HW assignment:** Principal Component Analysis (PCA)

- regression table

Intermezzo: Principal Component Analysis (PCA)

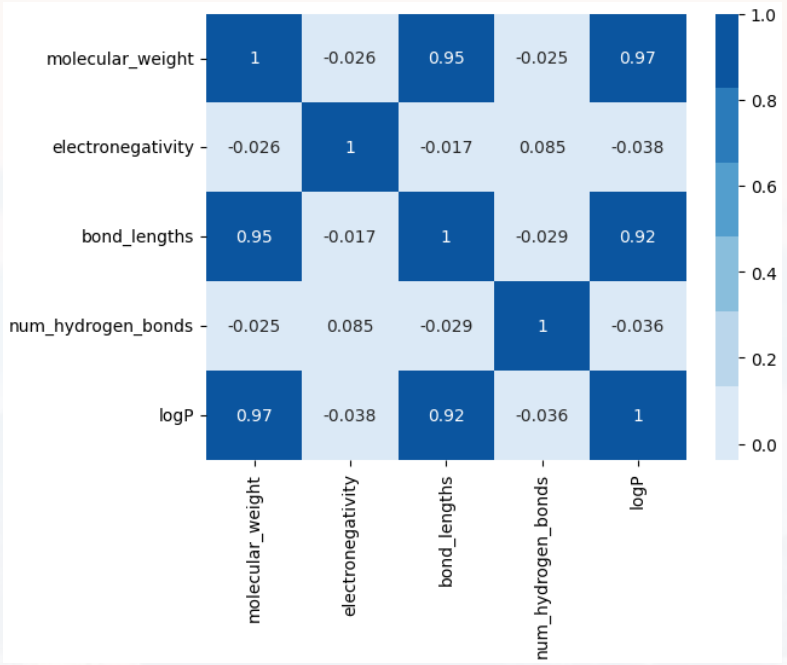
correlation means:

- features are **not mutually independent**

- we can predict feature ***a*** from feature ***b*** to some extend

- we don't need all features

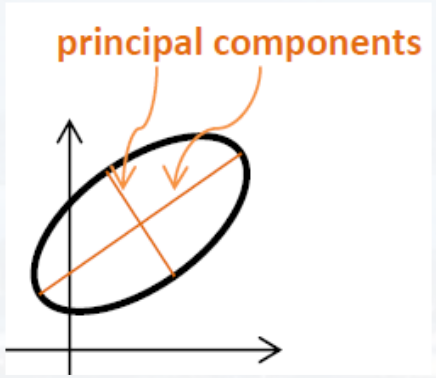
→ **reducing number of features** (dimensions) without losing information



$$\Sigma = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1N}^2 \\ \dots & \sigma_{ii}^2 & \dots & \sigma_{iN}^2 \\ \sigma_{N1}^2 & \sigma_{Ni}^2 & \dots & \sigma_{NN}^2 \end{pmatrix}$$

covariance matrix

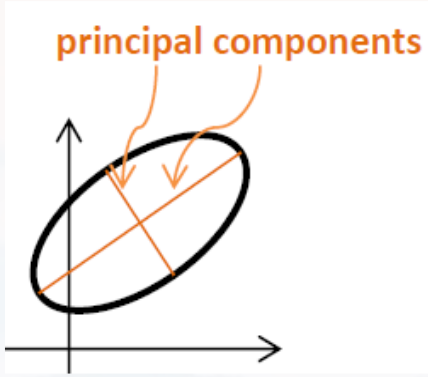
can be interpreted as a quadratic form



Sivia, "Data Analysis" section 3.2

$$\Sigma = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1N}^2 \\ \dots & \sigma_{ii}^2 & \dots & \sigma_{iN}^2 \\ \sigma_{N1}^2 & \sigma_{Ni}^2 & \dots & \sigma_{NN}^2 \end{pmatrix}$$

covariance matrix



Sivia, "Data Analysis" section 3.2

can be interpreted as a quadratic form

idea: diagonalize Σ

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

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

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

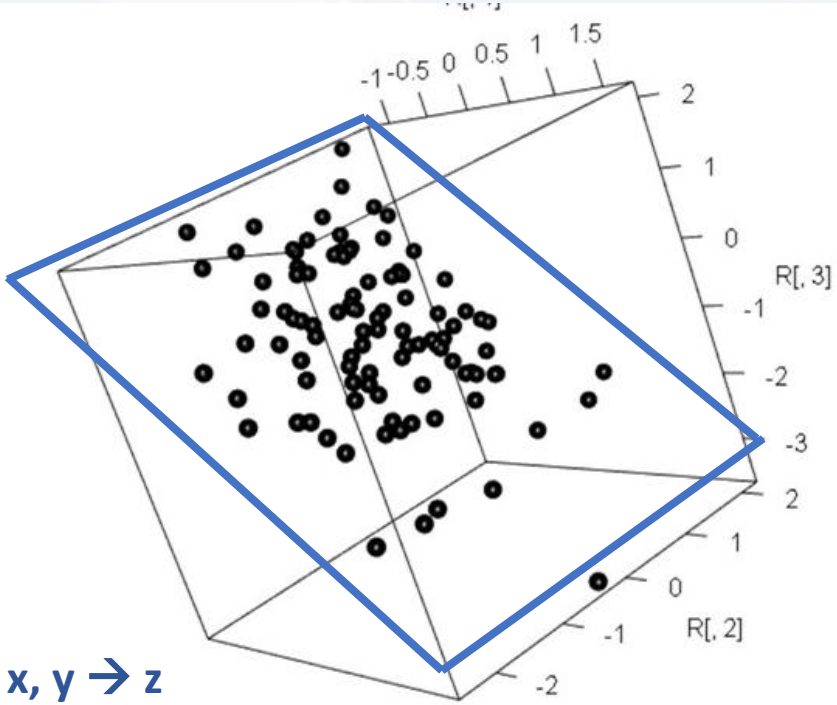
Intermezzo:

idea: diagonalize Σ

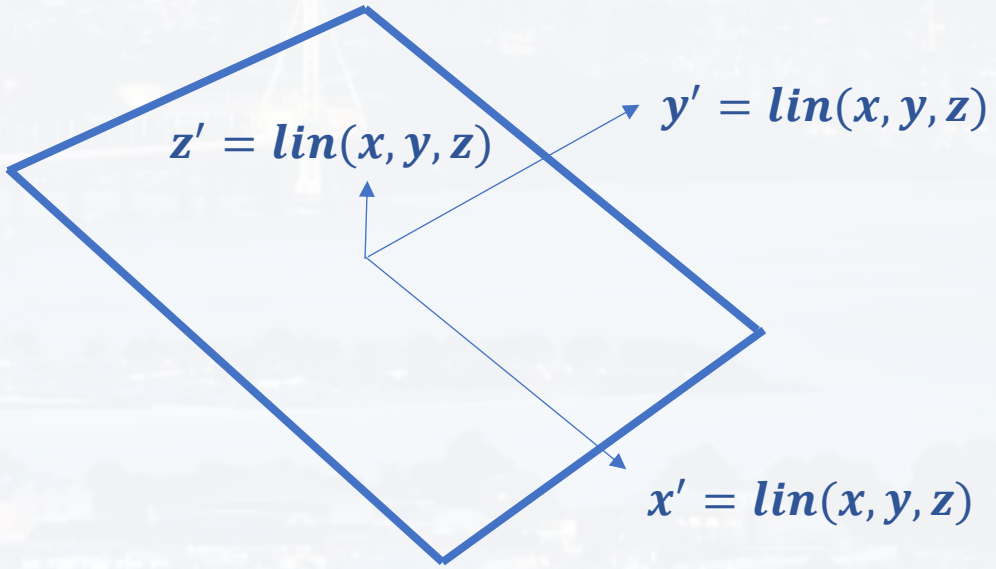
$$\begin{pmatrix} \lambda_1 & 0 \dots & 0 \\ 0 & \lambda_i & 0 \\ 0 & 0 & \lambda_N \end{pmatrix}$$

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

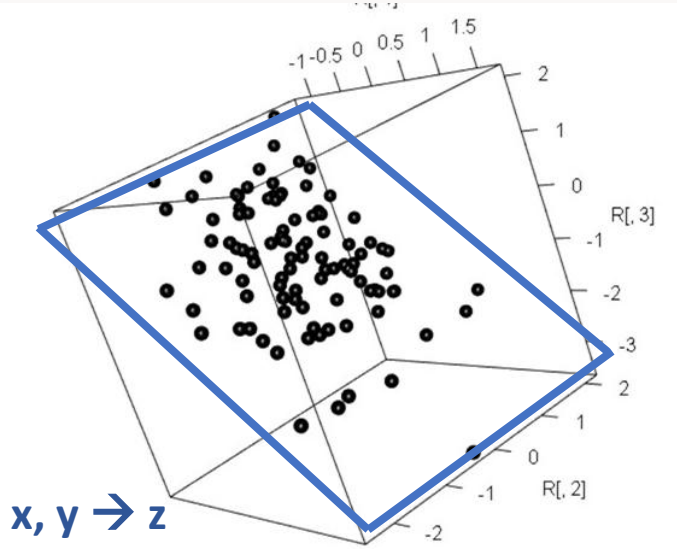
principal components of the new **covariance matrix** are **parallel** to the **new coordinate axes (= eigenvectors)**



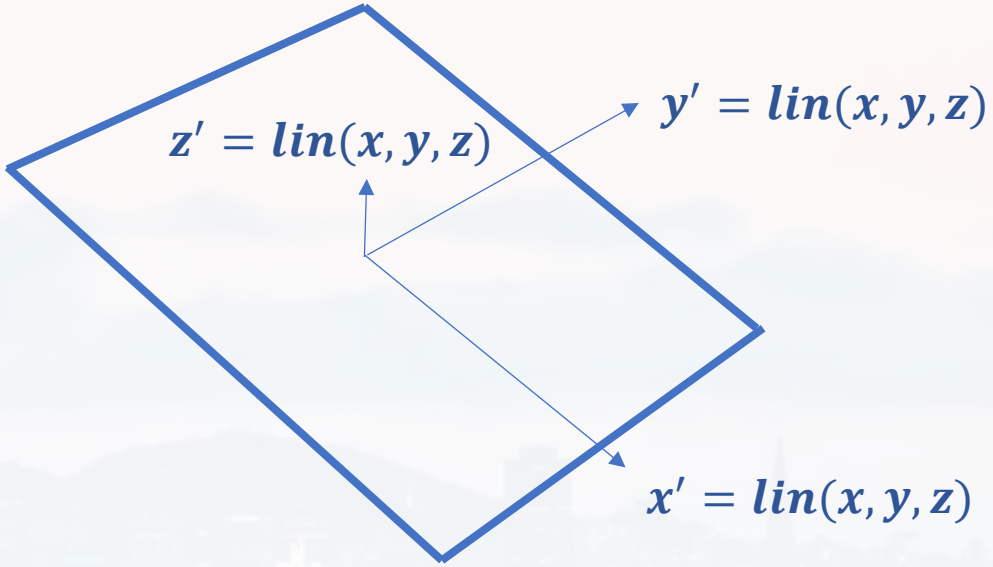
x, y → z correlate



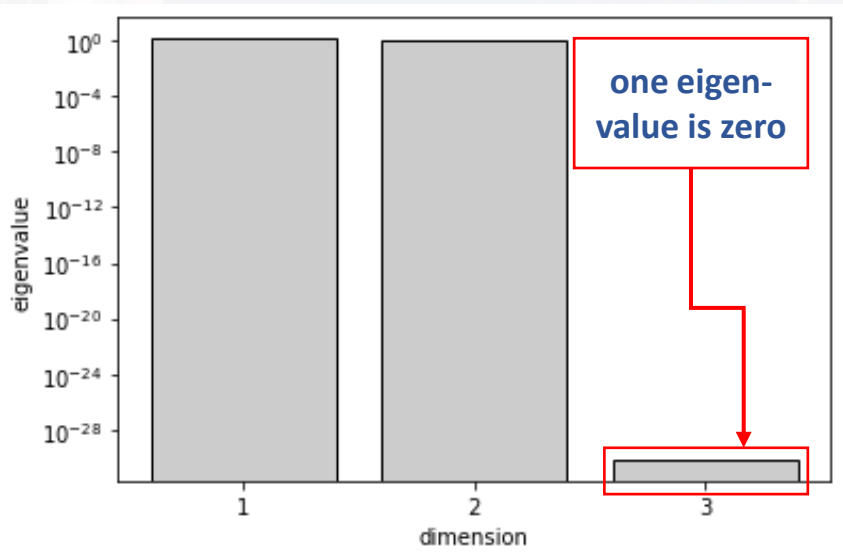
Intermezzo: Principal Component Analysis (PCA)



x, y → z
correlate




```
from sklearn.decomposition import PCA

out = PCA(n_components = 3).fit(XYZ)

eigenVec = out.components_
eigenVal = out.explained_variance_
eigenXYZ = out.transform(XYZ)
```



Intermezzo

- **for next HW assignment:** variance, covariance, correlation
- **for next HW assignment:** Principal Component Analysis (PCA)
- regression table

Intermezzo: Regression Table

see also discussion this week

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

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

Intermezzo:

Regression Table

```
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!

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

p-value for constant model

p-values for factors

2σ conf range of factors

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

Intermezzo:

Regression Table

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)
```

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

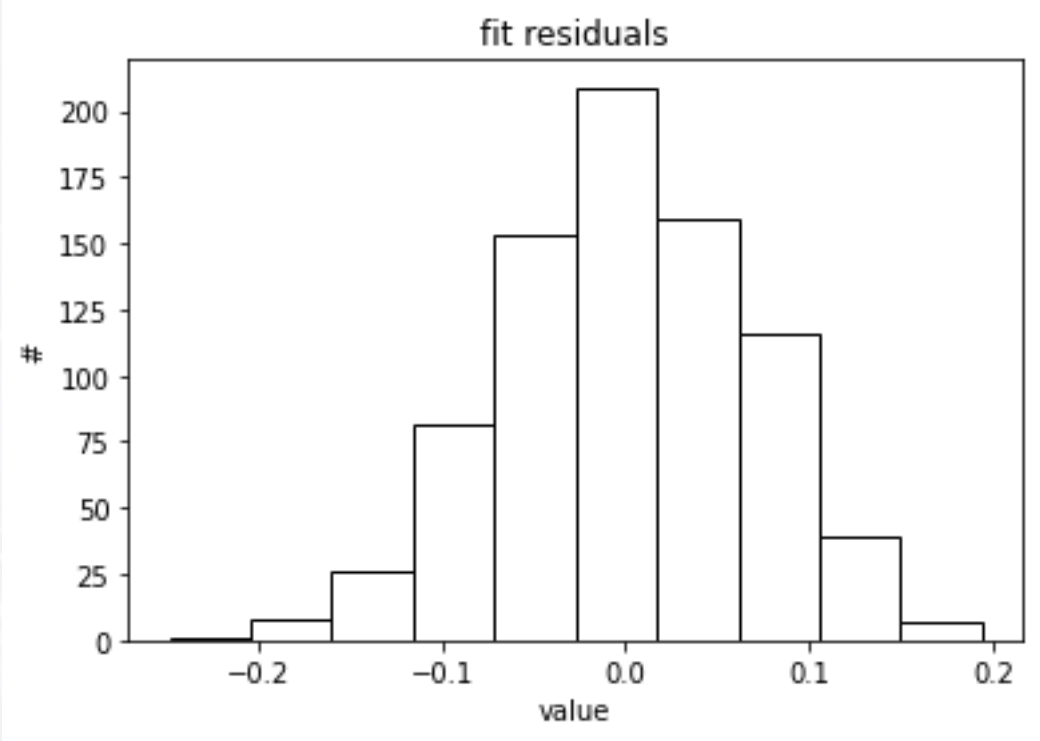
	df	sum_sq	mean_sq	F	PR(>F)
molecular_weight	1.0	13.346285	13.346285	2847.525516	8.024085e-265
electronegativity	1.0	0.640388	0.640388	136.631363	3.085962e-29
bond_lengths	1.0	0.000684	0.000684	0.145954	7.025342e-01
num_hydrogen_bonds	1.0	0.000703	0.000703	0.150055	6.985866e-01
logP	1.0	0.013917	0.013917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

Intermezzo:


```
residuals = my_model.resid

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



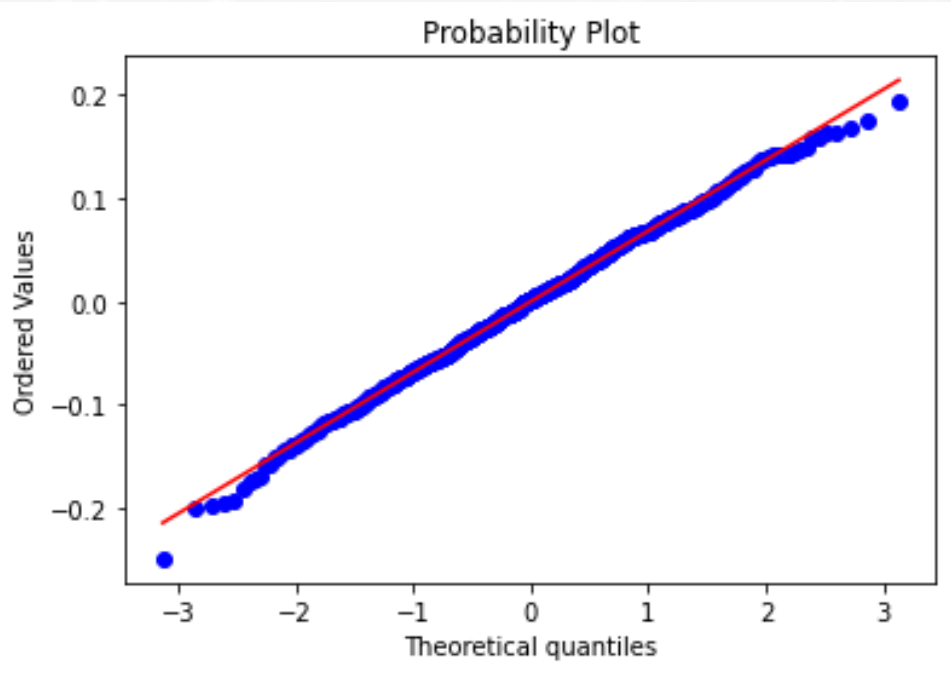
residuals approx. normally distributed around μ = 0

Intermezzo:


```
residuals = my_model.resid

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

residuals approx. normally distributed around μ = 0

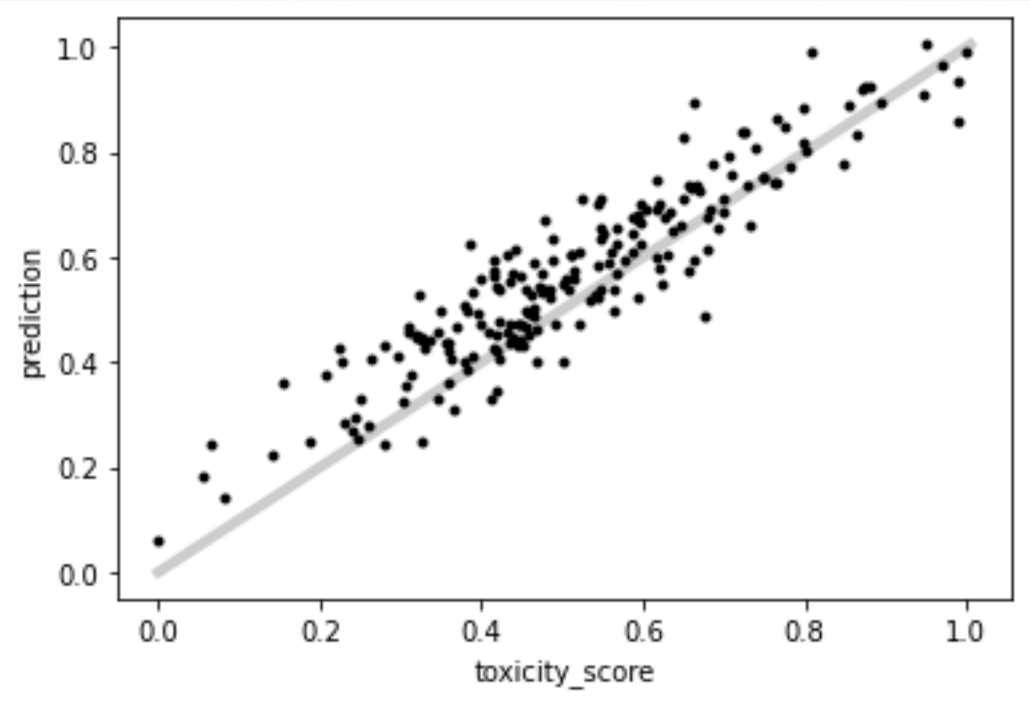
```
stats.probplot(residuals, dist = "norm", plot = pylab)
pylab.show()
```



Intermezzo:

Regression Table

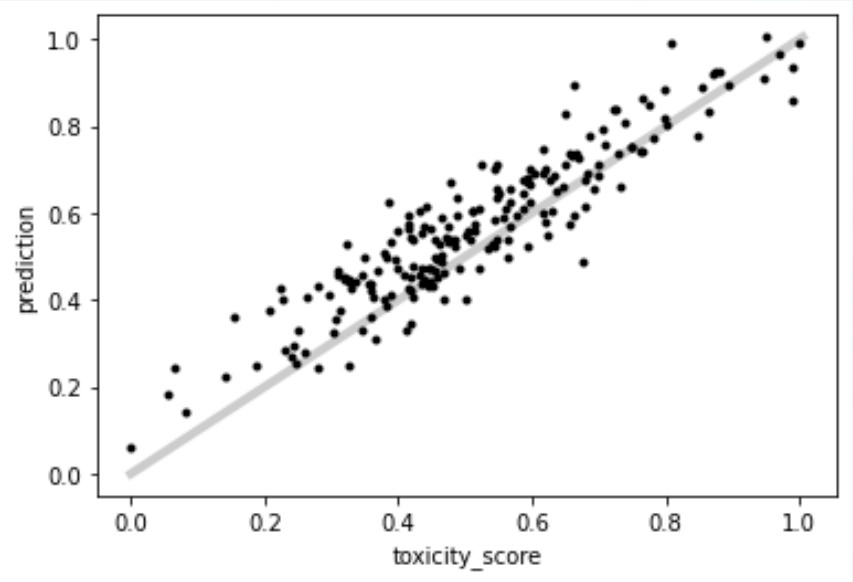
```
Ypred  = my_model.predict(TestS)

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()
```



Intermezzo:

Regression Table

```
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()
```




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

5%

Intermezzo

- **for next HW assignment:** variance, covariance, correlation
- **for next HW assignment:** Principal Component Analysis (PCA)
- regression table