

Lecture 02:

Bayesian Methods



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)

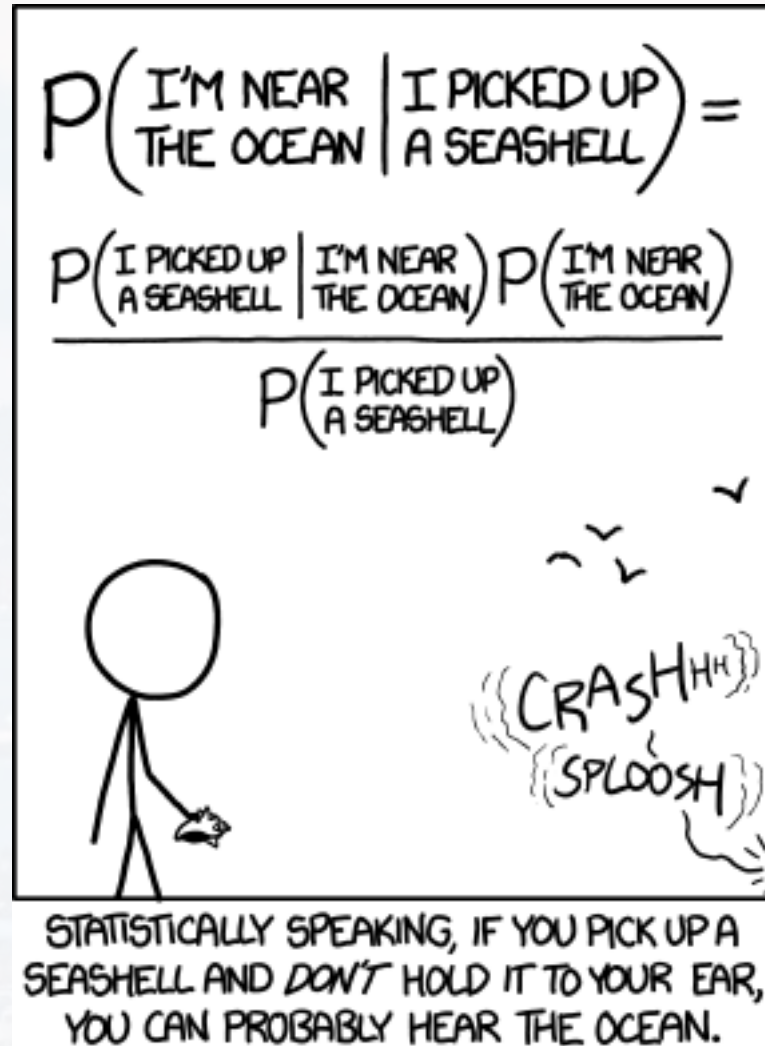
Lecture 12: Combining LSTMs and CNNs

Lecture 13: Running Models on GPUs and Parallel Processing

Lecture 14: Project Presentations

Lecture 15: Transformer

Lecture 16: GNN

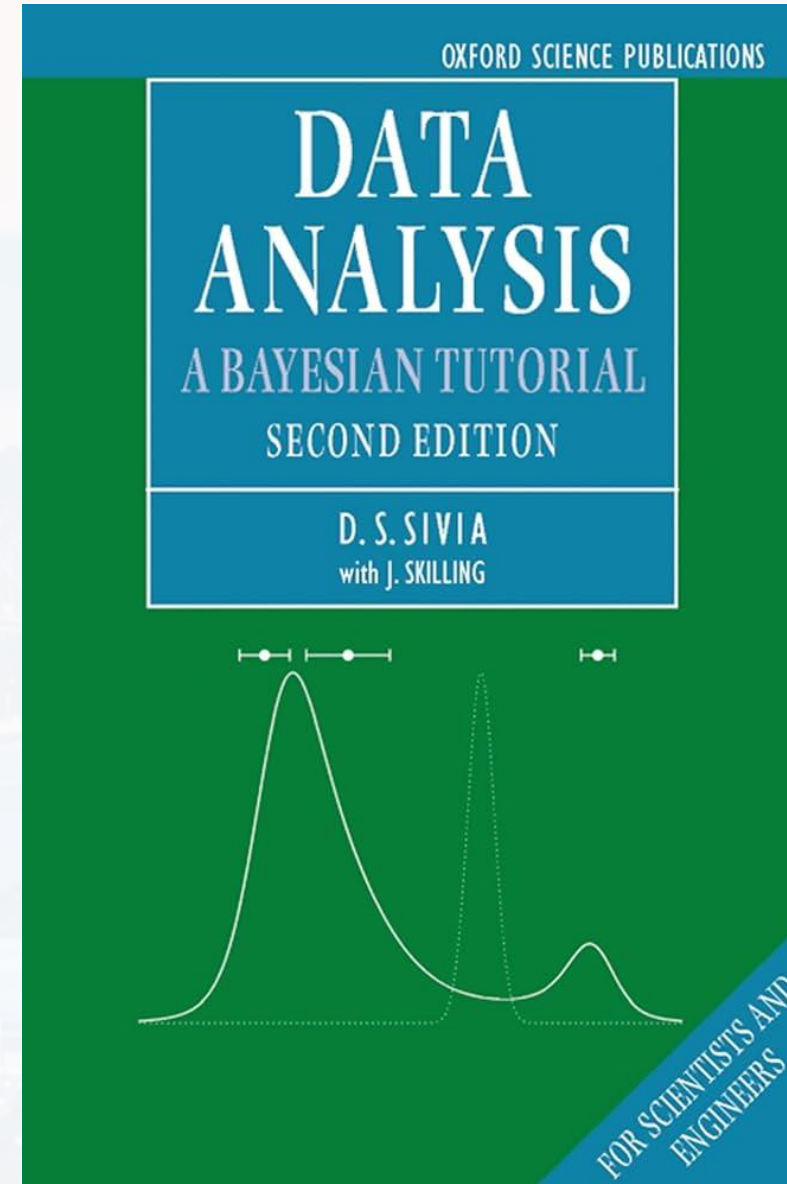
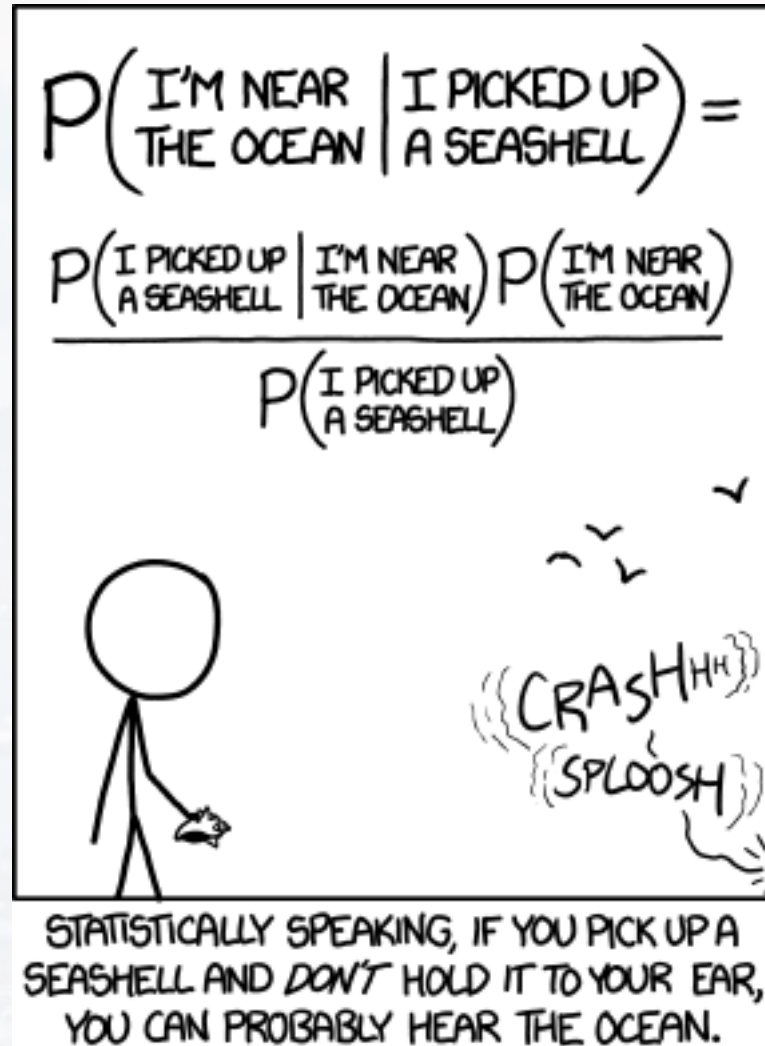


Outline

- The Idea and Bayes Theorem
- Naïve Bayes
- Parameter Estimation
- Model Selection

FYI

- Bayesian Networks (Graphs)
- Variational Bayes





Outline

- The Idea and Bayes Theorem

- Naïve Bayes

- Parameter Estimation

- Model Selection

FYI

- Bayesian Networks (Graphs)

- Variational Bayes



Why Bayesian Statistics?

frequentist: assuming sample is infinite (even tough there are corrections for small n)

vs:

Bayesian:

- taking the **exact amount** of information into account that's available
- model **"learns"** by adding more data (BPE)
- is based on **information theory & links to quantum mechanics**

→ **maximum entropy, given constraints** (prior knowledge)

→ variational calculus

- o EM algorithm (GMM, HMM etc)

- o **V**ariational **A**uto **E**ncoder

→ non-parametric (e. g. in contrast to MLE)

....and more



$P(A \cap B)$ probability **P** that the events **A** and **B** occur

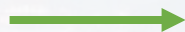
so far: A and B were independent $P(A \cap B) = P(A)P(B) = P(B)P(A)$

now: **conditional probabilities** | “given” or “under the condition”



Thomas Bayes
(1701 - 1761)

$$\begin{aligned} P(A \cap B) &= P(A|B)P(B) \\ &= P(B|A)P(A) \end{aligned}$$



$$P(A|B)P(B) = P(B|A)P(A)$$

Bayes Theorem

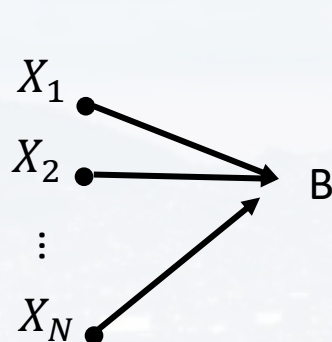
$$\text{posterior } \mathbf{P(A|B)} = \frac{P(B|A)\mathbf{P(A)}}{P(B)} \text{ prior}$$



$$P(A|B)P(B) = P(B|A)P(A)$$

Bayes Theorem

posterior $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ prior



$$P(B) = \sum_{n=1}^N P(B|X_n)P(X_n)$$

$$P(B) = \int P(B|X)P(X) dX$$

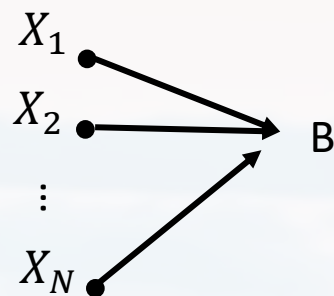
marginalization



Thomas Bayes
(1701 - 1761)

Probability $P(B)$ that I am going to be too late for a meeting:

$$P(B) = P(B|I \text{ forgot that I have a meeting}) P(I \text{ forgot that I have a meeting}) + \\ P(B|I \text{ got sick}) P(I \text{ got sick}) + \\ P(B|BART \text{ was too late}) P(BART \text{ was too late}) + \dots$$



$$P(B) = \sum_{n=1}^N P(B|X_n)P(X_n)$$

$$P(B) = \int P(B|X)P(X) dX$$

marginalization



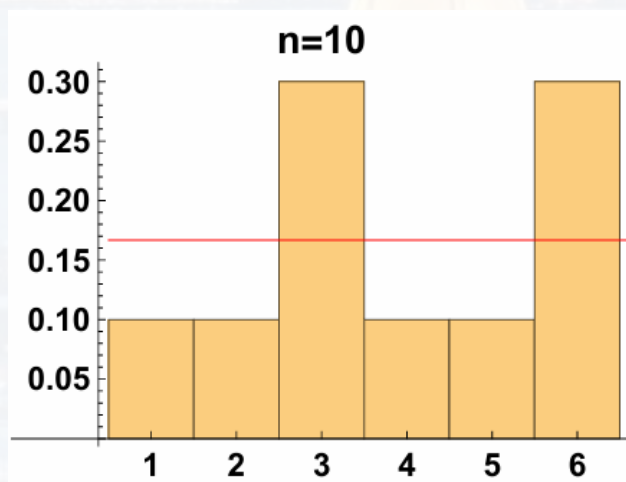
Thomas Bayes
(1701 - 1761)

model: M

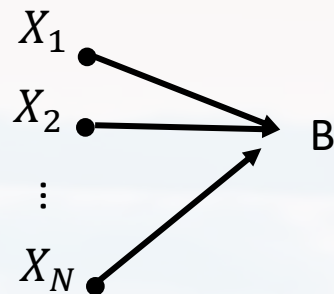
data: D

for a normal distribution M: $\mathcal{N}(\mu, \sigma)$

$$P(D|M) = \int P(D|\mu, \sigma, M) P(\mu, \sigma|M) d\Omega_{\mu, \sigma}$$



$\sigma = 2, \mu = 3.5$
 $\sigma = 2, \mu = 5.0$
 $\sigma = 1.5, \mu = 3.5$
 $\sigma = 7.0, \mu = 1.0$ and so on



$$P(B) = \sum_{n=1}^N P(B|X_n)P(X_n)$$

$$P(B) = \int P(B|X)P(X) dX$$

} marginalization



Thomas Bayes
(1701 - 1761)

example:

model: M
data: D

$$P(D|M) = \int P(D|\text{all model param}, M) P(\text{all model param}|M) d \text{ all model param}$$

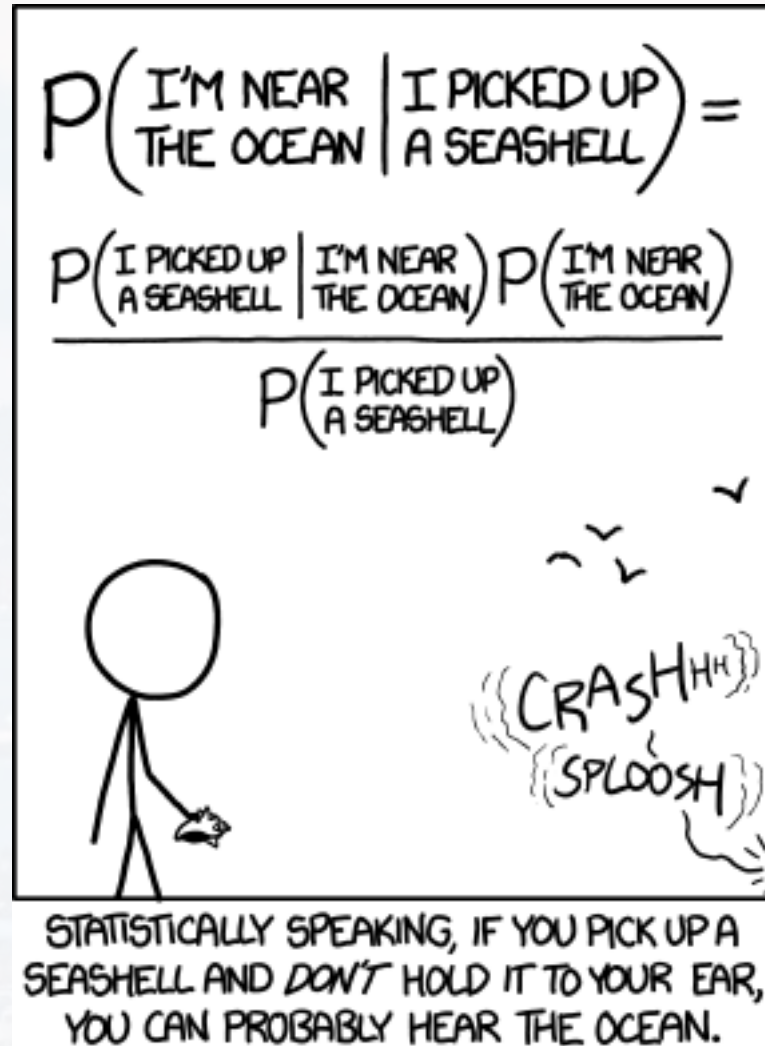
for a normal distribution $\mathcal{N}(\mu, \sigma)$

$$P(D|\mathcal{N}) = \int P(D|\mathcal{N}(\mu, \sigma)) P(\mu, \sigma|\mathcal{N}(\mu, \sigma)) d \Omega_{\mu, \sigma}$$

for a Poisson distribution $p(\lambda)$

$$P(D|p) = \int P(D|p(\lambda)) P(\lambda|p(\lambda)) d \lambda$$

and so on...



Outline

- The Idea and Bayes Theorem
- **Naïve Bayes**
- Parameter Estimation
- Model Selection

FYI

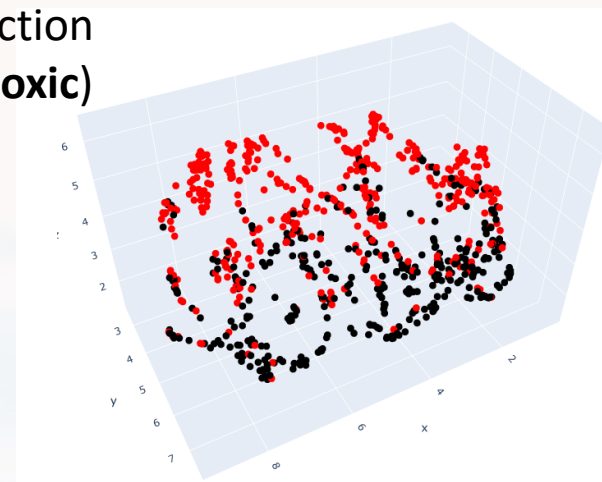
- Bayesian Networks (Graphs)
- Variational Bayes



$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes Theorem

UMAP projection
(**toxic**, **non-toxic**)



\vec{x} : vector with all model parameters (or features)

Index	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP	label
0	413.228	2.94416	3.41991	1	10.4335	Toxic
1	447.945	3.55371	3.66831	7	10.3475	Toxic
2	309.199	3.19761	2.84841	0	7.88825	Non-Toxic
3	382.554	3.8653	3.46237	8	9.59041	Toxic
4	310.904	3.18141	2.87774	6	7.85477	Non-Toxic
5	353.857	3.12105	3.32724	6	8.58887	Non-Toxic

K different classes
(here K = 2)



\vec{x} : vector with all model parameters (or features)

Index	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP	label
0	413.228	2.94416	3.41991	1	10.4335	Toxic
1	447.945	3.55371	3.66831	7	10.3475	Toxic
2	309.199	3.19761	2.84841	0	7.88825	Non-Toxic
3	382.554	3.8653	3.46237	8	9.59041	Toxic
4	310.904	3.18141	2.87774	6	7.85477	Non-Toxic
5	353.857	3.12105	3.32724	6	8.58887	Non-Toxic

$P(C_k|\vec{x})$: probability that datapoint belongs to class C_k , given \vec{x}
 $P(\vec{x}|C_k)$: probability that datapoint has features \vec{x} , given class C_k

K different classes
(here $K = 2$)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \text{Bayes Theorem}$$

$$P(C_k|\vec{x}) = \frac{P(\vec{x}|C_k)P(C_k)}{P(\vec{x})} = \frac{P(C_k)}{P(\vec{x})} \prod_{i=1}^I P(x_i|C_k) \sim P(C_k) \prod_{i=1}^I P(x_i|C_k) \quad \sum_{k=1}^K P(C_k|\vec{x}) = 1$$

Naïve Bayes:

- all features are **mutually independent**
- i. e.: no **correlation** between features
- features can be factorized



$$P(C_k|\vec{x}) = \frac{P(\vec{x}|C_k)P(C_k)}{P(\vec{x})} = \frac{P(C_k)}{P(\vec{x})} \prod_{i=1}^I P(x_i|C_k)$$

$$\sim P(C_k) \prod_{i=1}^I P(x_i|C_k)$$

$$\sum_{k=1}^K P(C_k|\vec{x}) = 1$$

new data point: finding the class C_k , that maximizes $P(C_k|\vec{x})$

$$k_{new} = \underset{k}{\operatorname{argmax}} \left\{ P(C_k) \prod_{i=1}^I P(x_i|C_k) \right\}$$

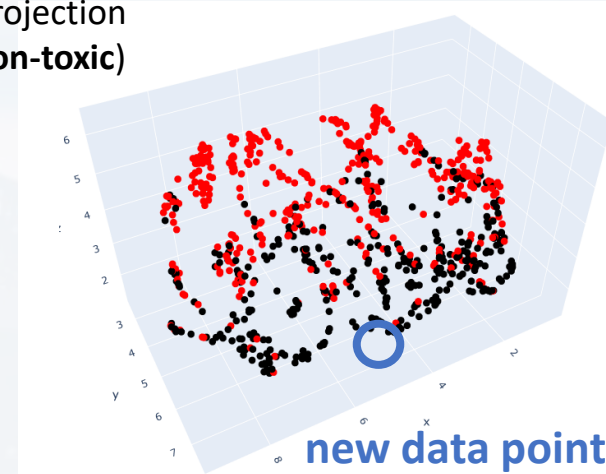
from the training data → supervised learning

different models for $P(x_i|C_k)$

- multinomial
- Gaussian
- ...

$P(C_k|\vec{x})$: probability that datapoint belongs to class C_k , given \vec{x}
 $P(\vec{x}|C_k)$: probability that datapoint has features \vec{x} , given class C_k

UMAP projection
(toxic, non-toxic)





from the training data → supervised learning

1) creating the model:

```
my_model = library.method(argument1 = 'arg1', ... )
```

2) training the model

```
out = my_model.fit(xtrain, ytrain)
```

3) evaluation

```
ypred = out.predict(xeval)  
accur = (ypred == yeval).sum()/len(yeval)
```

4) prediction (actual application)

```
ypred = out.predict(xnew)
```



Python:

```
from sklearn.naive_bayes import *
```

importing methods for
naïve bayes

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

scaling/normalizing data

```
Train = pd.read_csv('molecular_train_gbc_cat.csv')
```

```
Test = pd.read_csv('molecular_test_gbc_cat.csv')
```

```
XTrain = Train.drop('Label', axis = 1).values
```

```
YTrain = Train['Label']
```

```
XTest = Test.drop('Label', axis = 1).values
```

```
YTest = Test['Label']
```

```
print(YTrain[:10])
```

```
0      Toxic
1      Toxic
2  Non-Toxic
3  Non-Toxic
4  Non-Toxic
5      Toxic
6  Non-Toxic
7  Non-Toxic
8      Toxic
9  Non-Toxic
Name: label, dtype: object
```



```
scaler = MinMaxScaler(feature_range = (0, 1))  
XTrainS = scaler.fit_transform(XTrain)
```

scaling the data to
mean = 0 and std = 1

```
gnb = GaussianNB()  
Fit = gnb.fit(XTrainS, YTrain)
```

the actual fit

applying the model to the test data set

```
XTestS = scaler.transform(XTest)
```

scaling the test set
without fitting

```
Ypred = Fit.predict(XTestS)  
Probs = Fit.predict_proba(XTestS)
```

predicting the class
 C_k and calculating
the probabilities

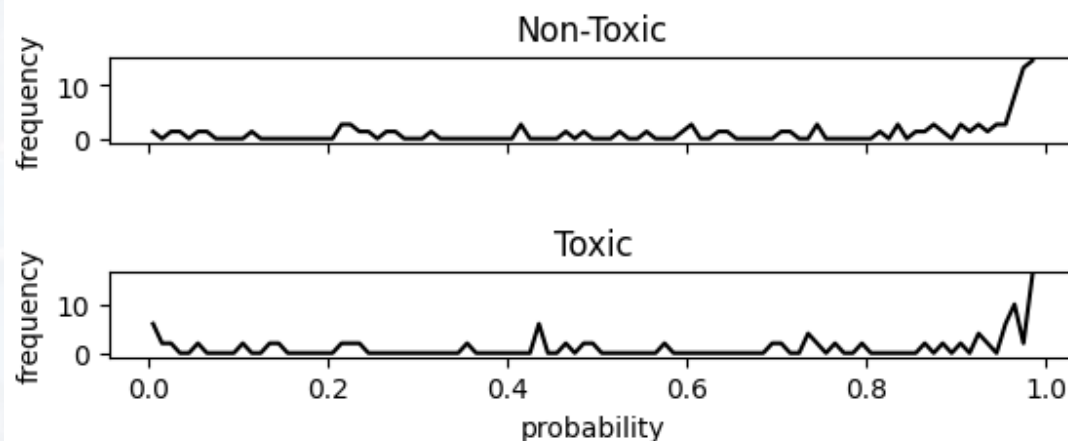


```
XTestS = scaler.transform(XTest)
```

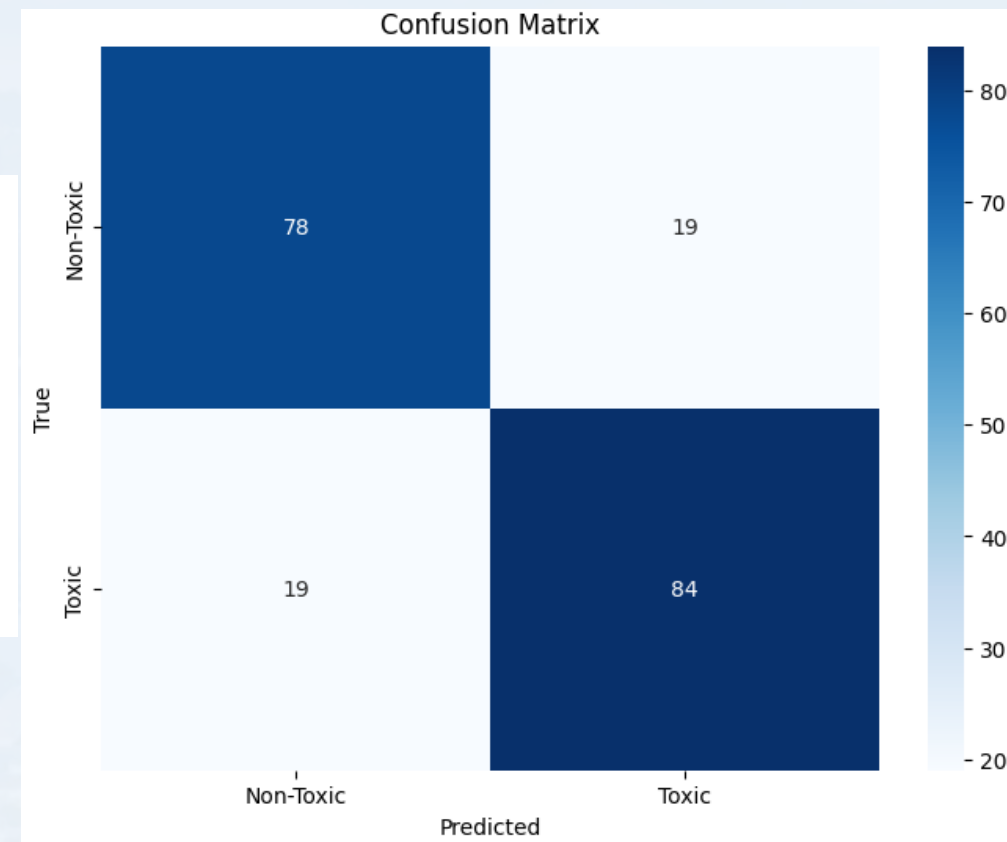
```
Ypred = Fit.predict(XTestS)
```

```
Probs = Fit.predict_proba(XTestS)
```

evaluation



see
`Walk_Through_NaiveBayes.ipynb`



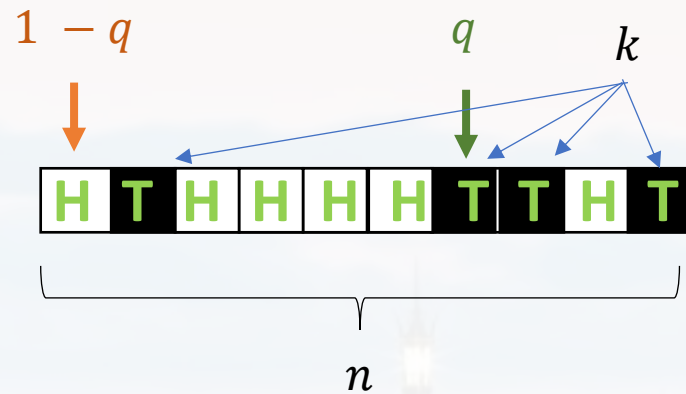


Outline

- The Idea and Bayes Theorem
- Naïve Bayes
- **Parameter Estimation**
- Model Selection

FYI

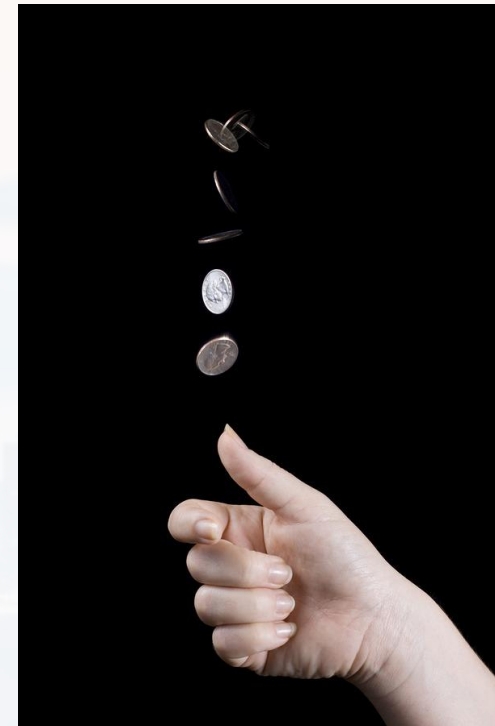
- Bayesian Networks (Graphs)
- Variational Bayes

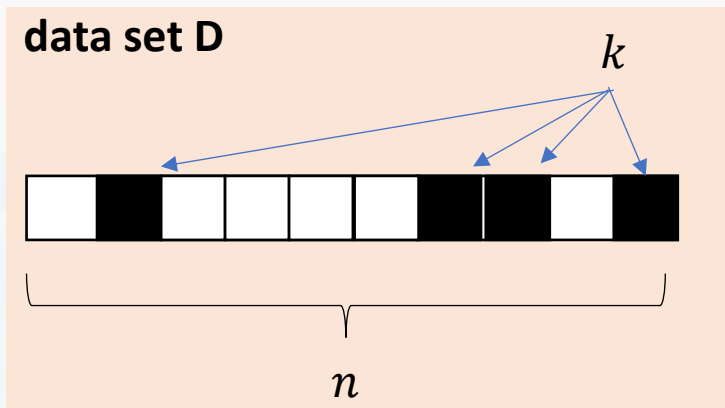


$$q = ?$$

fair coin? $q = 0.5$???

mutation $q = ??$



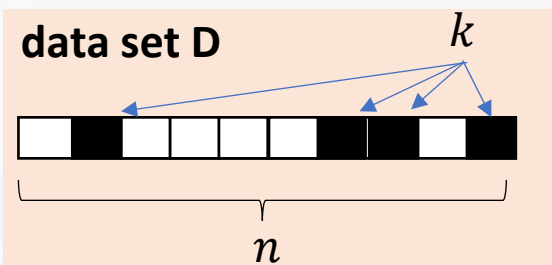


$q = ?$

goal:

- $P(q|D)$

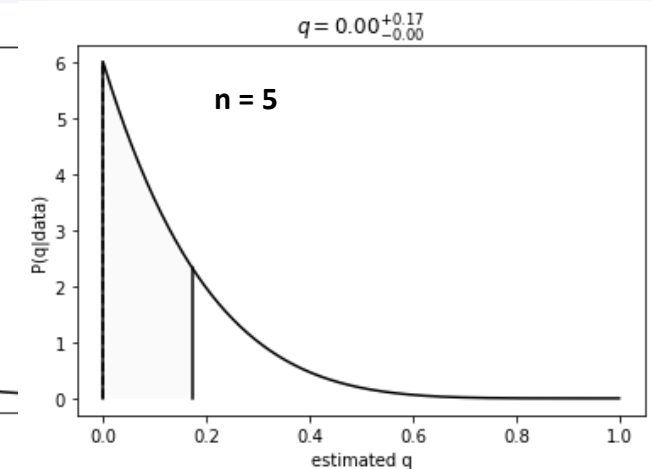
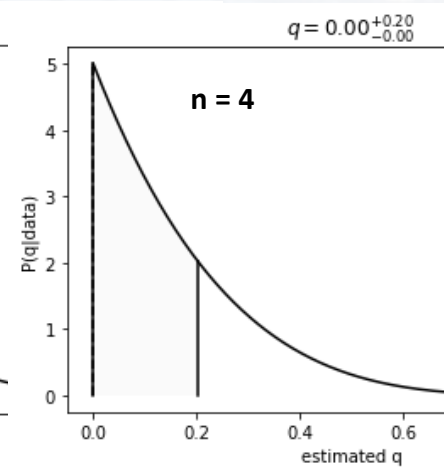
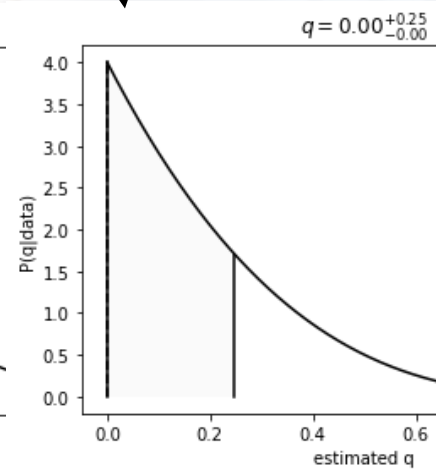
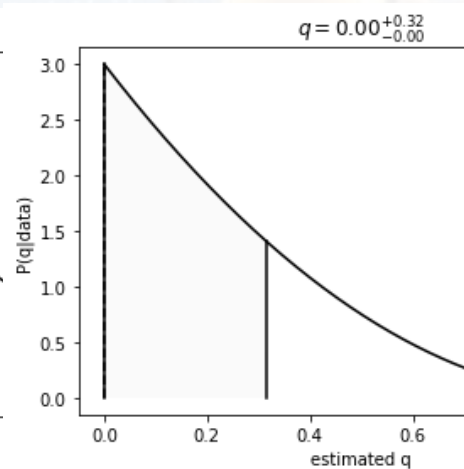
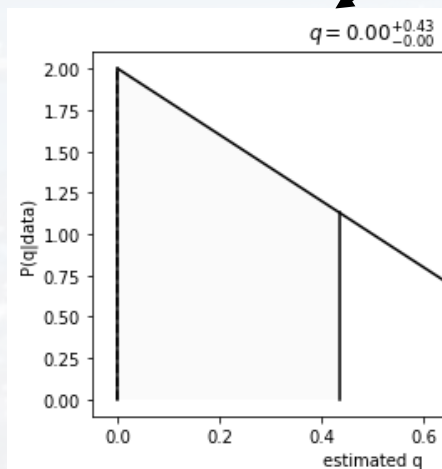
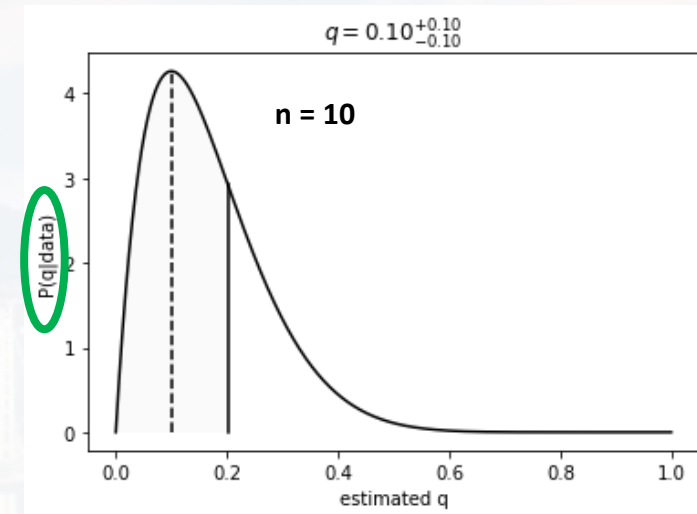
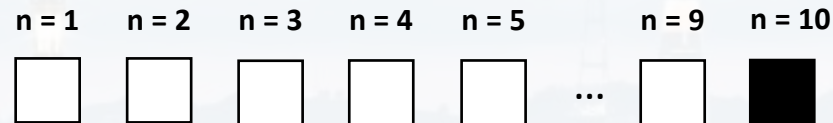
- the larger D , the more certain q
→ learning

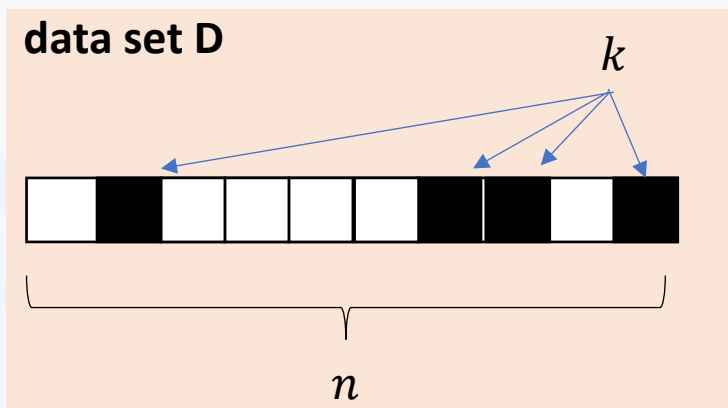


$q = ?$

goal:

- $P(q|D)$
- the larger D , the more certain q
→ learning





$q = ?$

goal:

- $P(q|D)$
- the larger D , the more certain q
→ learning

$$P(k|n, q) = \binom{n}{k} q^k (1 - q)^{n-k}$$

Bayes' theorem:

likelihood function (here: binomial)

$$P(q|\text{data set}) = \frac{P(\text{data set}|q)P(q)}{P(\text{data set})}$$

prior

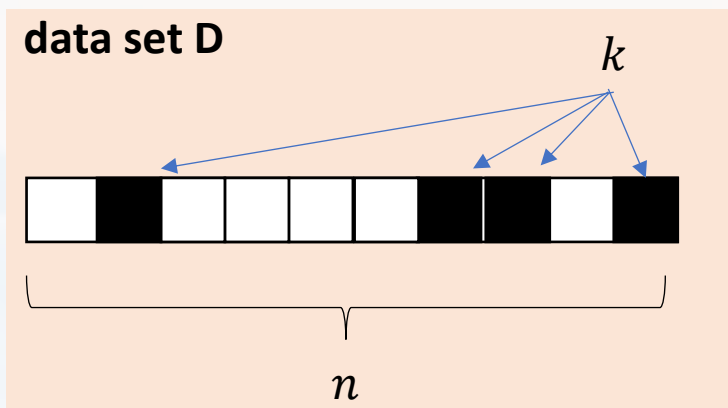
evidence (const wrt q)

$$= \frac{\binom{n}{k} q^k (1-q)^{n-k}}{P(D)} P(q)$$

$$\sim q^k (1 - q)^{n-k} P(q)$$

$P(D)$ and $\binom{n}{k}$ are no functions of q





$q = ?$

goal:

- $P(q|D)$
- the larger D , the more certain q
→ learning

$$P(k|n, q) = \binom{n}{k} q^k (1 - q)^{n-k}$$

$$P(q|data\ set) = \frac{P(data\ set|q)P(q)}{P(data\ set)}$$

$$= \frac{\binom{n}{k} q^k (1-q)^{n-k}}{P(D)} P(q)$$

$$\sim q^k (1 - q)^{n-k} P(q)$$

$$\sim q^k (1 - q)^{n-k}$$

max. entropy: $P(q) = \text{const}$
if no prior information about q

$$P(q|data\ set) = \frac{q^k (1 - q)^{n-k}}{\int_0^1 q^k (1 - q)^{n-k} dq}$$



check out `bayesian_bino.py`

```
n1 = 4
```

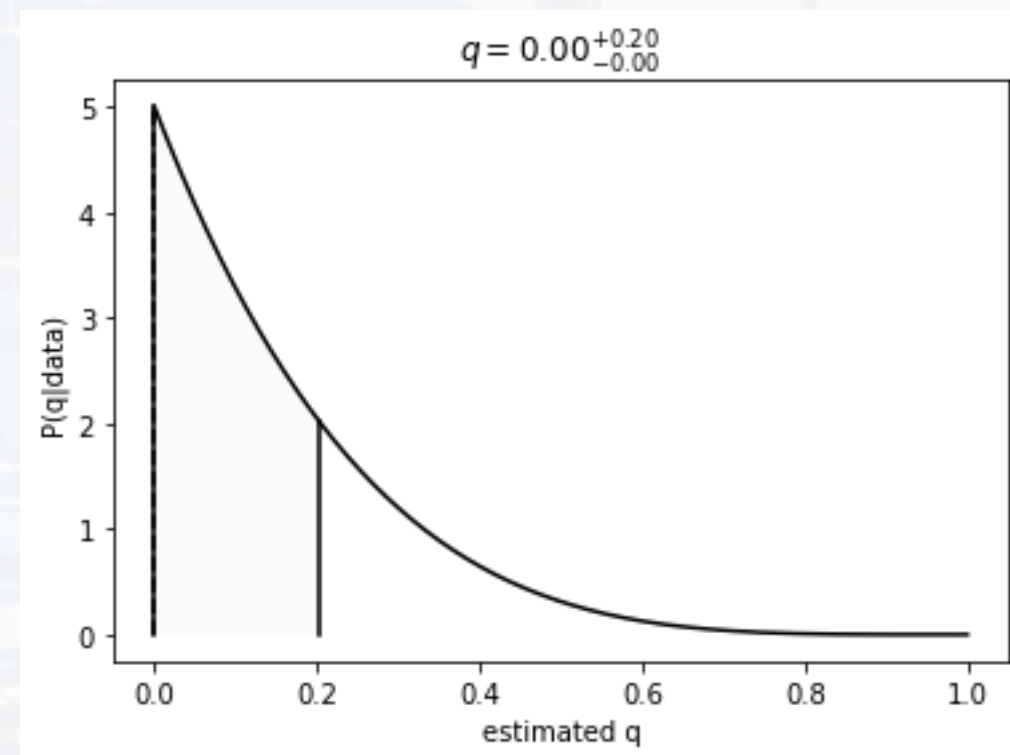
```
k1 = np.random.binomial(n1, 0.25)
```

creating artificial data set

note: in reality q is unknown!

```
[q1, b, _] = bayesian_bino(n1, k1)
```

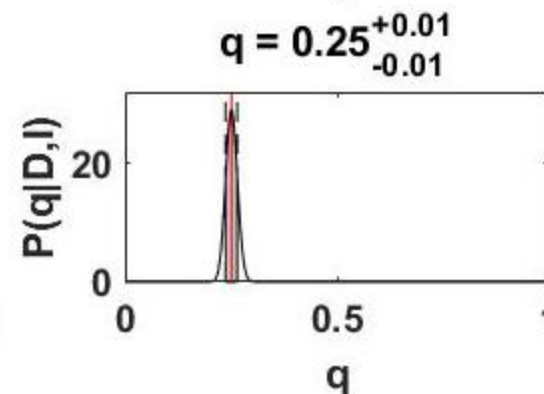
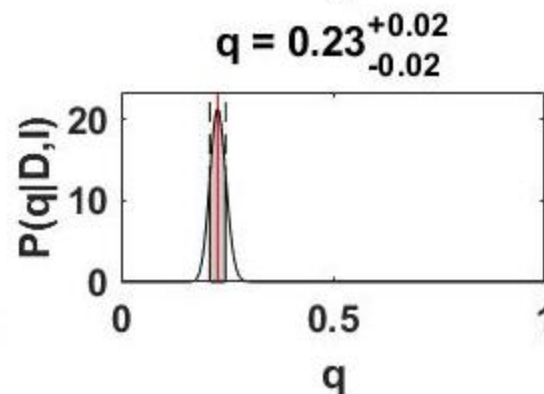
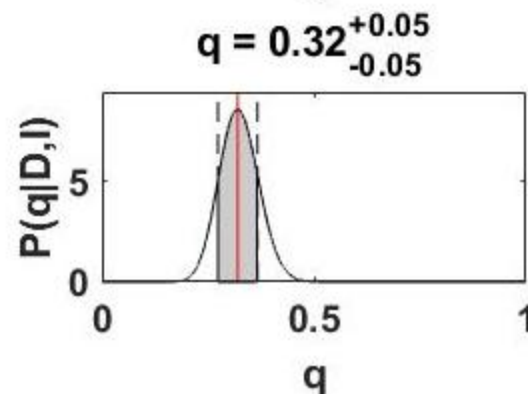
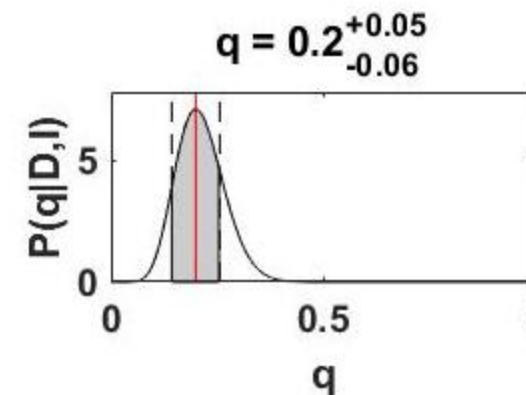
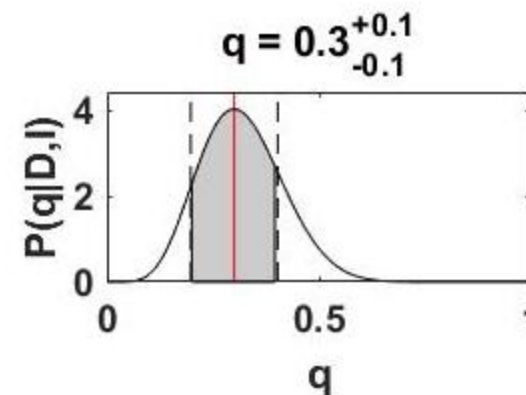
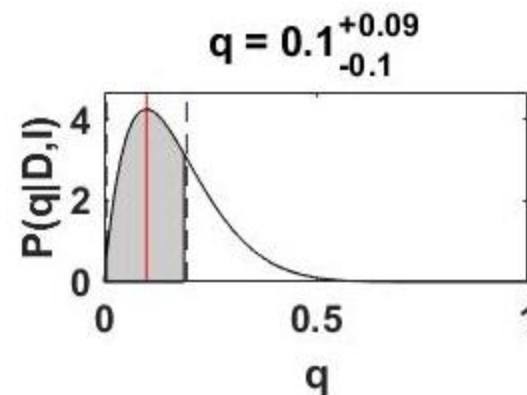
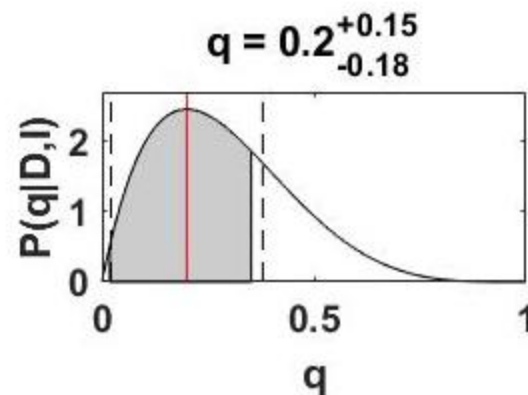
$$P(q|data\ set) = \frac{q^k(1-q)^{n-k}}{\int_0^1 q^k(1-q)^{n-k} dq}$$





check out `bayesian_bino.py`

$$P(q|data\ set) = \frac{q^k(1-q)^{n-k}}{\int_0^1 q^k(1-q)^{n-k} dq}$$

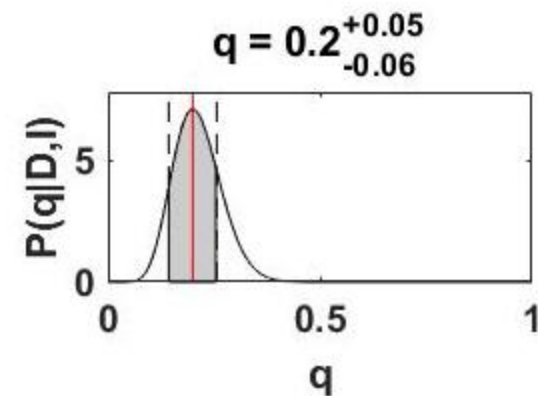
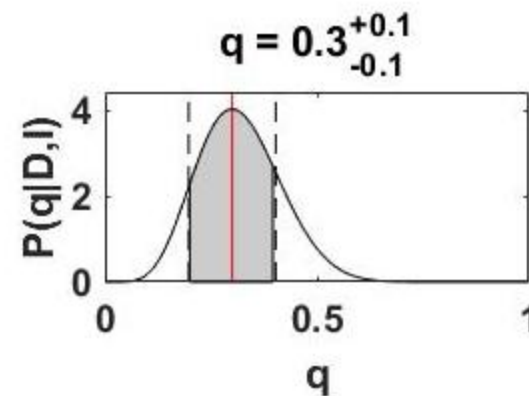
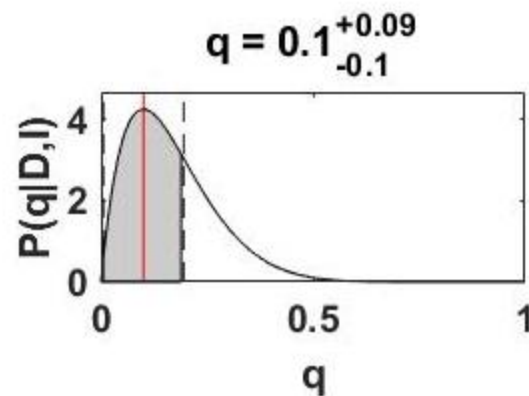
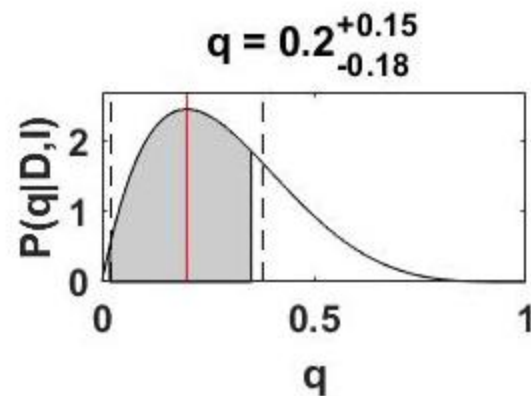


n	estimated q
5	$0.2^{+0.15}_{-0.18}$
10	$0.1^{+0.09}_{-0.1}$
20	$0.3^{+0.1}_{-0.1}$
50	$0.2^{+0.05}_{-0.06}$
100	$0.32^{+0.05}_{-0.05}$
500	$0.23^{+0.02}_{-0.02}$
1,000	$0.25^{+0.01}_{-0.01}$
infinity	0.25



check out `bayesian_bino.py`

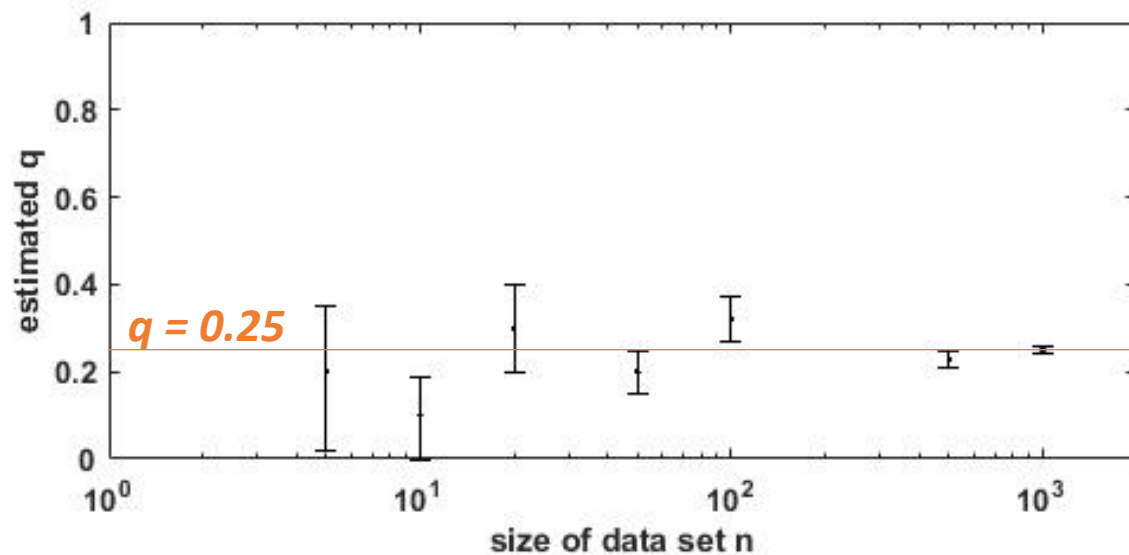
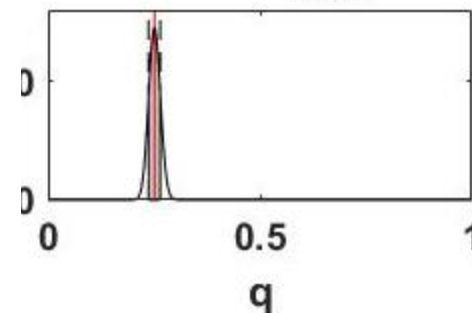
$$P(q|data\ set) = \frac{q^k(1-q)^{n-k}}{\int_0^1 q^k(1-q)^{n-k} dq}$$



$q = 0.32^{+0.05}_{-0.05}$

$q = 0.23^{+0.02}_{-0.02}$

$q = 0.25^{+0.01}_{-0.01}$



n	estimated q
5	$0.2^{+0.15}_{-0.18}$
10	$0.1^{+0.09}_{-0.1}$
20	$0.3^{+0.1}_{-0.1}$
50	$0.2^{+0.05}_{-0.06}$
100	$0.32^{+0.05}_{-0.05}$
500	$0.23^{+0.02}_{-0.02}$
1,000	$0.25^{+0.01}_{-0.01}$
infinity	0.25



Of course, Bayesian Parameter Estimation works with **any other pdf**

goal:

- $P(q|D)$
- the larger D , the more certain q
→ learning

likelihood function

$$P(q|data\ set) = \frac{P(data\ set|q)P(q)}{P(data\ set)}$$

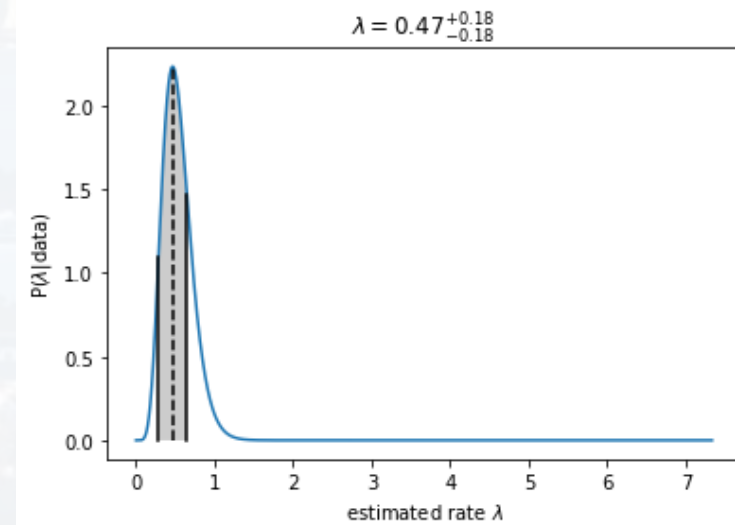
prior
evidence (const wrt q)

What is the average number of WhatsUp messages I get every day?

Mon:	5	event	- has no duration
Tue:	7		- is rare
Wed:	1		
Thu:	3		→ Poissonian
Fri:	9		
Sat:	2		
Sun:	5		

```
data = np.random.poisson(lam = 0.4, 15)
poissfit(data)
```

$$P(k|\lambda) = \frac{\lambda^k}{k!} e^{-\lambda}$$





Of course, Bayesian Parameter Estimation works with **any other pdf**

goal:

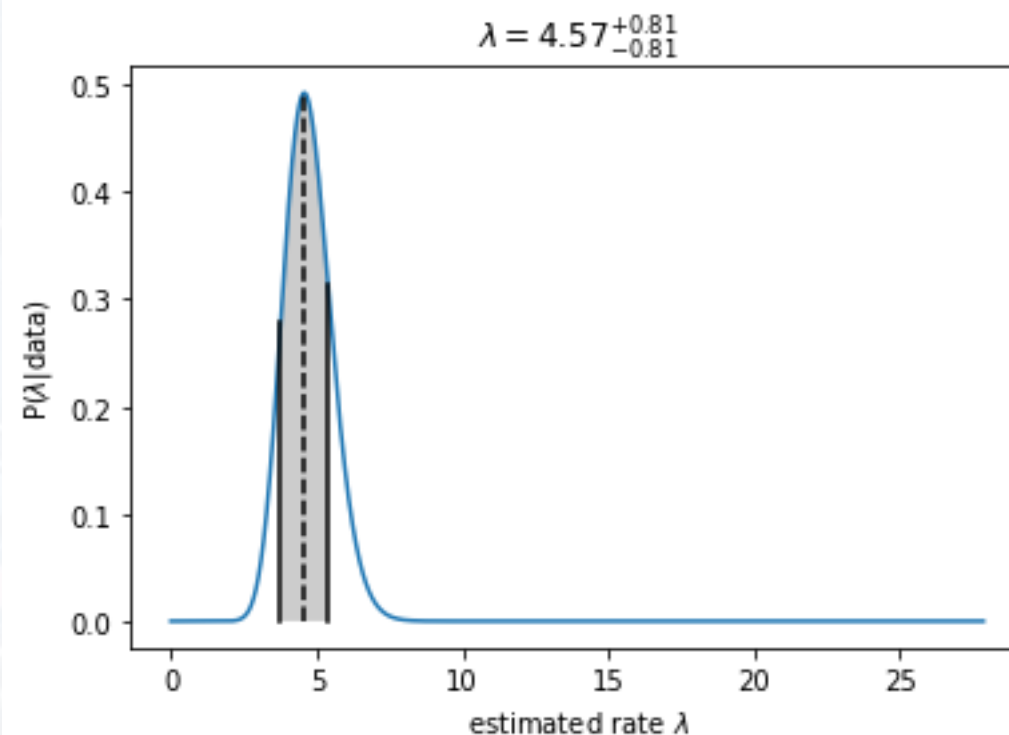
- $P(\mathbf{q}|\mathbf{D})$
- the larger \mathbf{D} , the more certain \mathbf{q}
→ learning

What is the average number of WhatsUp messages I get every day?

Mon:	5	event	- has no duration
Tue:	7		- is rare
Wed:	1		
Thu:	3		→ Poissonian
Fri:	9		
Sat:	2		
Sun:	5		

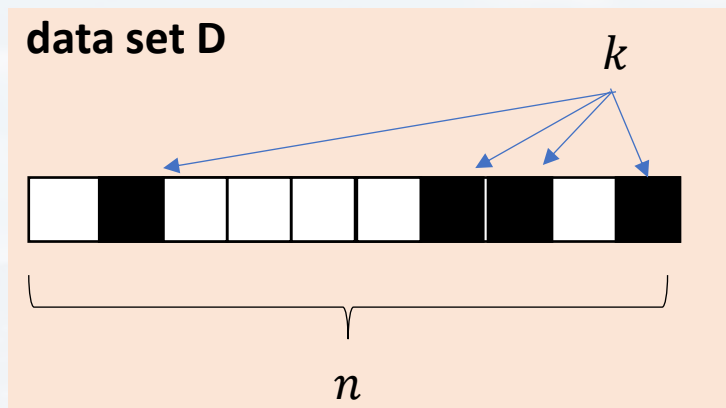
$$P(k|\lambda) = \frac{\lambda^k}{k!} e^{-\lambda}$$

```
poissfit([5, 7, 1, 3, 9, 2, 5])
```



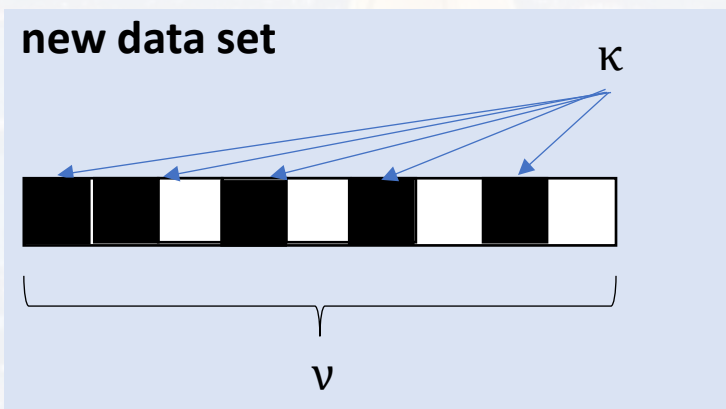
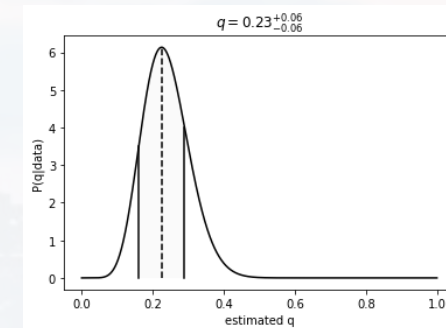


What if there is new data?



~~$q = ?$~~

$$P(q|\text{data set}) = \frac{q^k(1-q)^{n-k}}{\int_0^1 q^k(1-q)^{n-k} dq}$$



if there is prior information I about q :

$$P(q|\text{new data set}, I) = \frac{P(\text{new data set}|q, I) P(q, I)}{P(\text{new data set})}$$



What if there is new data?

$$P(q|\text{new data set}, I) = \frac{P(\text{new data set}|q, I) \mathbf{P(q, I)}}{P(\text{new data set})}$$

$$P(q|\text{data set}) = \frac{q^k(1-q)^{n-k}}{\int_0^1 q^k(1-q)^{n-k} dq}$$

$$= \frac{q^\kappa(1-q)^{\nu-\kappa} q^k(1-q)^{n-k}}{\int_0^1 q^\kappa(1-q)^{\nu-\kappa} q^k(1-q)^{n-k} dq}$$

$$= \frac{q^{k+\kappa}(1-q)^{\nu-\kappa+n-k}}{\int_0^1 q^{k+\kappa}(1-q)^{\nu-\kappa+n-k} dq}$$

$$= \frac{q^{k+\alpha-1}(1-q)^{n-k+\beta-1}}{\int_0^1 q^{k+\alpha-1}(1-q)^{n-k+\beta-1} dq}$$

often: $\kappa = \alpha - 1$
 $\beta = \nu - \kappa - 1$

Beta function

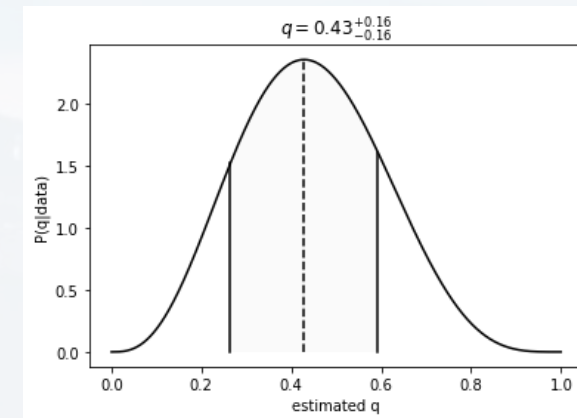
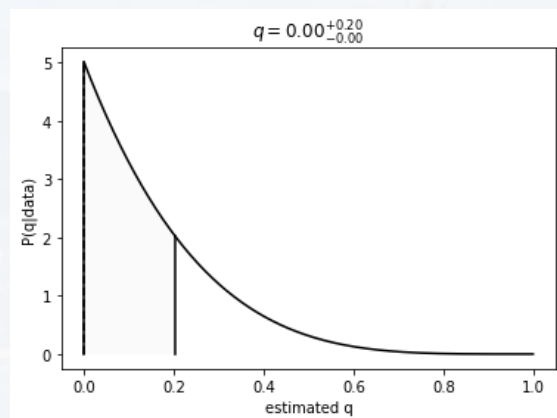


What if there is new data?

$$P(q|\text{new data set}, I) = \frac{q^\kappa(1-q)^{\nu-\kappa}}{\int_0^1 q^\kappa(1-q)^{\nu-\kappa} dq} \frac{q^k(1-q)^{n-k}}{q^\kappa(1-q)^{n-k}} dq$$

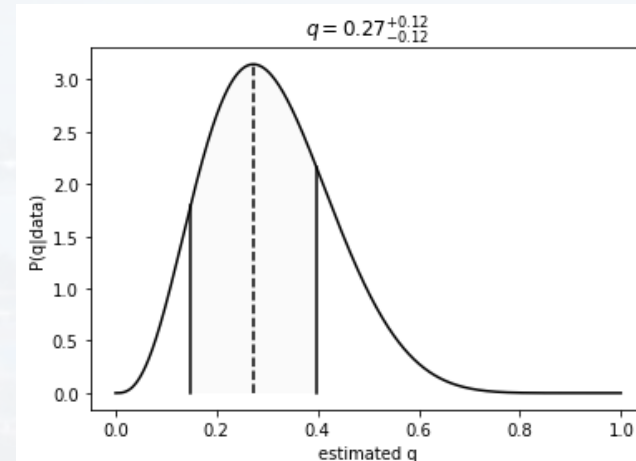
```
n1 = 4  
k1 = np.random.binomial(n1, q = 0.2)  
[_, _, Prior] = bayesian_bino(n1, k1)
```

```
n2 = 7  
k2 = np.random.binomial(n2, q = 0.2)  
[_, _, _] = bayesian_bino(n2, k2)
```



$$P(q, I) = \frac{q^k(1-q)^{n-k}}{\int_0^1 q^k(1-q)^{n-k} dq}$$

```
[_, _, _] = bayesian_bino(n2, k2, Prior = Prior)
```

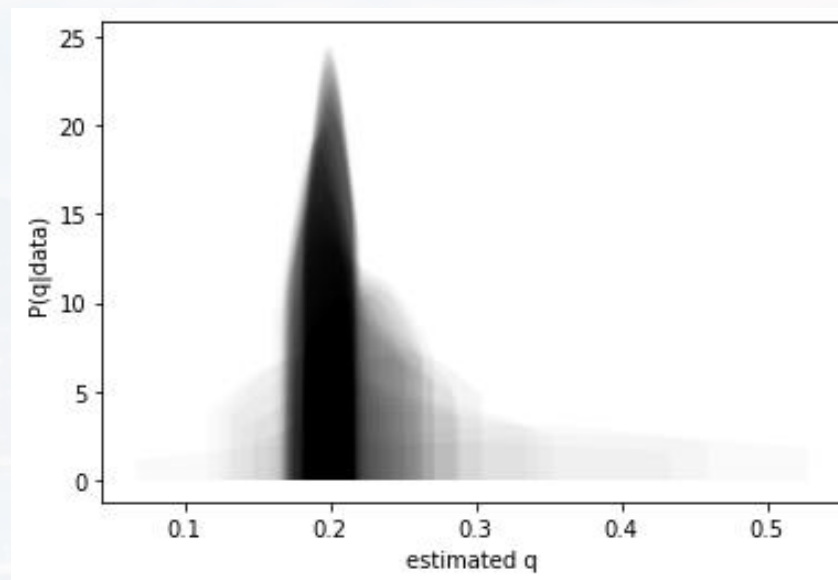




What if there is new data?

$$P(q|\text{new data set}, I) = \frac{q^\kappa(1-q)^{\nu-\kappa}}{\int_0^1 q^\kappa(1-q)^{\nu-\kappa}} \frac{q^k(1-q)^{n-k}}{q^k(1-q)^{n-k}} dq$$

The **posterior** from the **previous** experiment is the **prior** of the **next** experiment



→ we become more certain about the model parameters
→ learning!

→ see e.g. **Variational Auto Encoders**

2D images → 3D objects



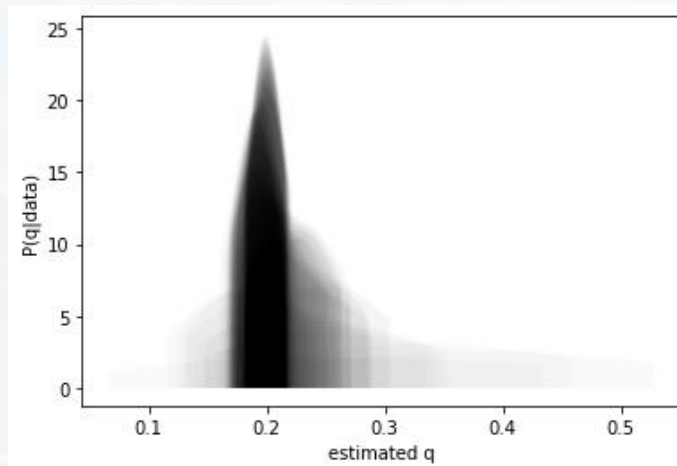
credit: StableAI



What if there is new data?

$$P(q|\text{new data set}, I) = \frac{q^\kappa(1-q)^{\nu-\kappa}}{\int_0^1 q^\kappa(1-q)^{\nu-\kappa}} \frac{q^k(1-q)^{n-k}}{q^k(1-q)^{n-k}} dq$$

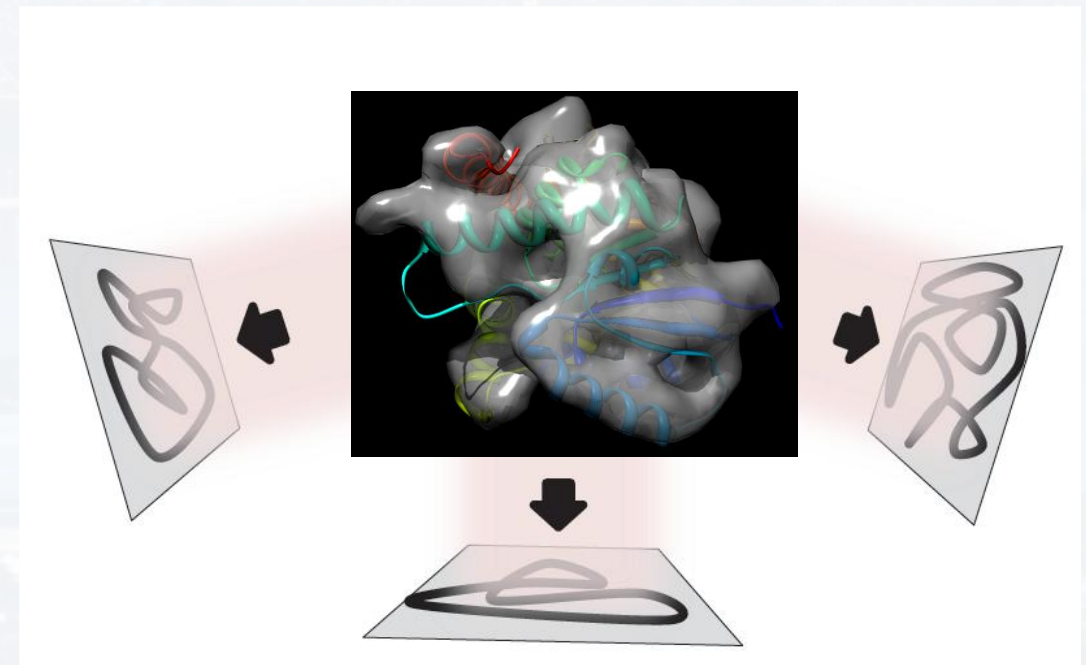
The **posterior** from the **previous** experiment is the **prior** of the **next** experiment



→ we become more certain about the model parameters
→ learning!

→ see e.g. **Variational Auto Encoders** 2D images → 3D objects

Cryo – EM: 3D structure from 2D images/projections



(image courtesy: Thomas Becker, GC LMU Munich)



Outline

- The Idea and Bayes Theorem
- Naïve Bayes
- Parameter Estimation
- Model Selection

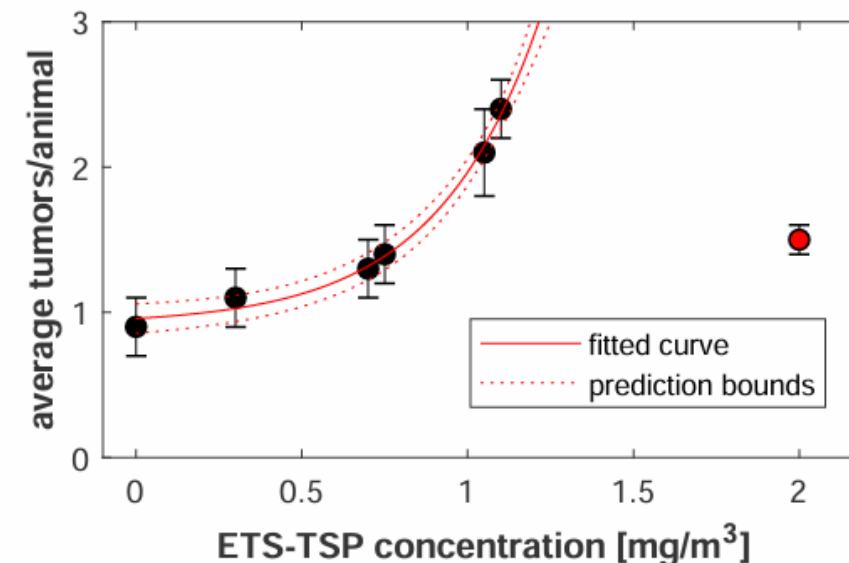
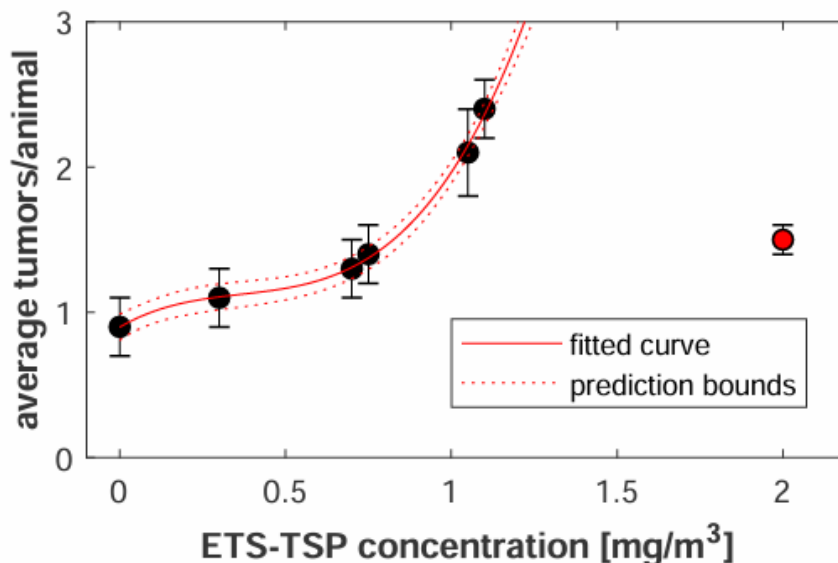
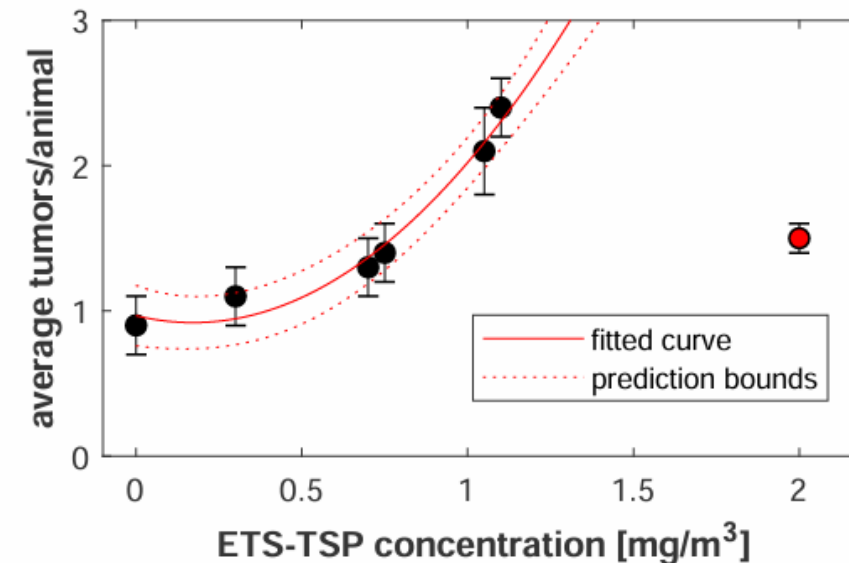
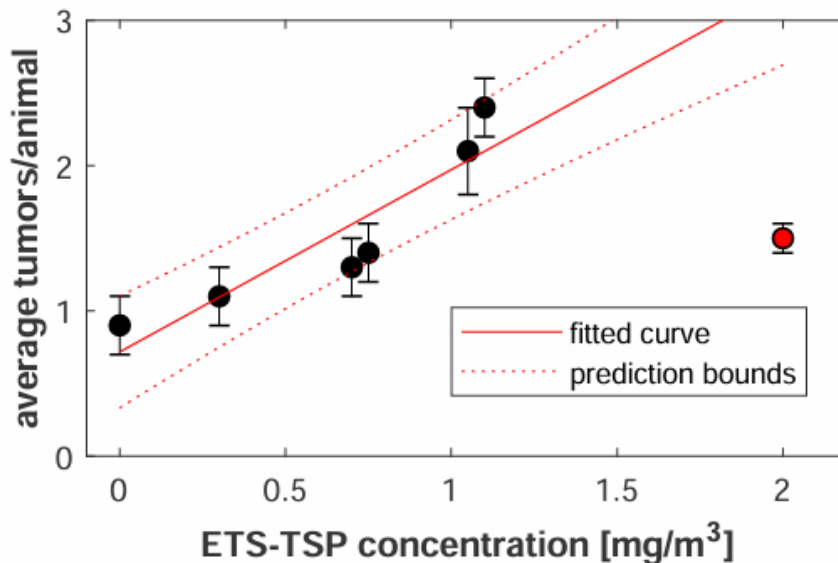
FYI

- Bayesian Networks (Graphs)
- Variational Bayes



often, we have many competing models

→ assigning probabilities if a model is correct





often, we have many competing models

→ assigning probabilities if a model is correct

goal: $\rho = \frac{P(M_A|D)}{P(M_B|D)}$ **Bayes' theorem**

D : data
M_A : model A
M_B : model B

$$= \frac{P(D|M_A) P(M_A)}{P(D)} \cdot \frac{P(D)}{P(D|M_B) P(M_B)}$$

marginalization:

$\{\alpha\}_i$: all parameter of model M_i

$$\begin{aligned} P(D|M_i) &= \int P(D|\{\alpha\}_i, M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i} \\ &= \int P(D|\{\alpha\}_i, M_i) \prod_j P(\alpha_{ij} | M_i) d\alpha_{ij} \end{aligned}$$

assuming all α_{ij} are
mutually independent
(Naïve Bayes)



goal:

$$\rho = \frac{P(M_A|D)}{P(M_B|D)} = \frac{P(D|M_A) P(M_A)}{P(D) P(M_B)}$$

D	: data
M_A	: model A
M_B	: model B
$\{\alpha\}_i$: all parameter of model M_i

marginalization:

$$P(D|M_i) = \int P(D|\{\alpha\}_i M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i}$$

$$= \int \underbrace{P(D|\{\alpha\}_i, M_i)}_{\text{likelihood function}} \prod_j \underbrace{P(\alpha_{ij} | M_i)}_{\text{prior of } \alpha_{ij} \text{ BEFORE(!) measurement}}$$

assuming all α_{ij} are
mutually independent
(Naïve Bayes)

likelihood function
→ the actual model

prior of α_{ij} BEFORE(!) measurement
Maximum Entropy without prior knowledge:

$$\frac{1}{\alpha_{ij}(max) - \alpha_{ij}(end)}$$



goal:

$$\rho = \frac{P(M_A|D)}{P(M_B|D)} = \frac{P(D|M_A) P(M_A)}{P(D)} \cdot \frac{P(D)}{P(D|M_B) P(M_B)}$$

D	: data
M_A	: model A
M_B	: model B
$\{\alpha\}_i$: all parameter of model M_i

marginalization:

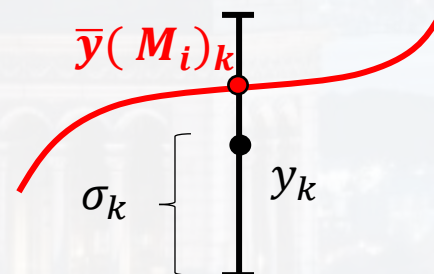
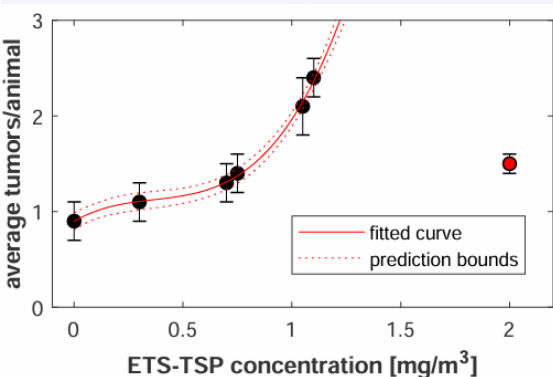
$$P(D|M_i) = \int P(D|\{\alpha\}_i M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i}$$

$$= \int P(D|\{\alpha\}_i, M_i) \prod_j P(\alpha_{ij} | M_i) d\alpha_{ij}$$

$$= \prod_j \frac{1}{\alpha_{ij}(\max) - \alpha_{ij}(\min)} \cdot \int P(D|\{\alpha\}_i, M_i) d\alpha_{ij}$$

assuming all α_{ij} are
mutually independent
(Naïve Bayes)

likelihood function
→ the actual model



y_k	: measured value
σ_k	: error
$\bar{y}(M_i)_k$: model value (after fit)



goal:

$$\rho = \frac{P(M_A|D)}{P(M_B|D)} = \frac{P(D|M_A) P(M_A)}{P(D) P(M_B)} \cdot \frac{P(D)}{P(D)}$$

D	: data
M_A	: model A
M_B	: model B
$\{\alpha\}_i$: all parameter of model M_i
y_k	: measured value
σ_k	: error
$\bar{y}(M_i)_k$: model value (after fit)

marginalization:

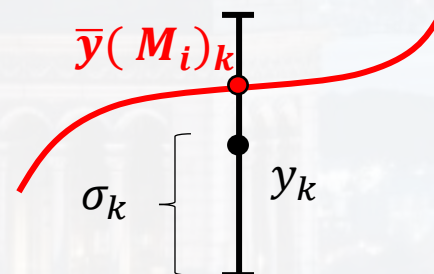
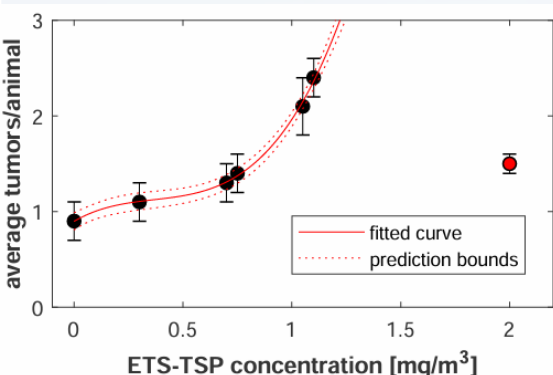
$$P(D|M_i) = \int P(D|\{\alpha\}_i M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i}$$

$$= \int P(D|\{\alpha\}_i, M_i) \prod_j P(\alpha_{ij} | M_i) d\alpha_{ij}$$

$$= \prod_j \frac{1}{\alpha_{ij}(\max) - \alpha_{ij}(\min)} \cdot \int P(D|\{\alpha\}_i, M_i) d\alpha_{ij}$$

assuming all α_{ij} are
mutually independent
(Naïve Bayes)

likelihood function
→ the actual model



$$P(y_k | \alpha_{ij}, M_i) \approx \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{1}{2} \frac{(\bar{y}(M_i)_k - y_k)^2}{\sigma_k^2}} \quad \text{for } \sigma_k \ll |y_k|$$



marginalization:

$$P(D|M_i) = \int P(D|\{\alpha\}_i M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i}$$

$$= \int P(D|\{\alpha\}_i, M_i) \prod_j P(\alpha_{ij} | M_i) d\alpha_{ij}$$

$$= \prod_j \frac{1}{\alpha_{ij}(\max) - \alpha_{ij}(\min)} \cdot \int P(D|\{\alpha\}_i, M_i) d\alpha_{ij}$$

$$P(D|\{\alpha\}_i M_i) = \prod_k P(y_k | \alpha_{ij}, M_i) = \prod_k \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{1}{2} \frac{(\bar{y}(M_i)_k - y_k)^2}{\sigma_k^2}}$$

$$= \left(\prod_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \right) \cdot e^{-\frac{1}{2} \sum_k \frac{(\bar{y}(M_i)_k - y_k)^2}{\sigma_k^2}} = \left(\prod_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \right) \cdot e^{-\frac{1}{2} \chi_i^2}$$

D	: data
M_A	: model A
M_B	: model B
{α}_i	: all parameter of model M_i
y_k	: measured value
σ_k	: error
ȳ(M_i)_k	: model value (after fit)

likelihood function
→ the actual model



$$\rho = \frac{P(D|M_A) P(M_A)}{P(D|M_B) P(M_B)}$$

D	: data
M_A	: model A
M_B	: model B
$\{\alpha\}_i$: all parameter of model M_i
y_k	: measured value
σ_k	: error
$\bar{y}(M_i)_k$: model value (after fit)

$$= \frac{P(M_A)}{P(M_B)} \cdot \frac{\int e^{-\frac{1}{2}\chi_A^2} d\Omega_{\{\alpha\}_A}}{\int e^{-\frac{1}{2}\chi_B^2} d\Omega_{\{\alpha\}_B}} \cdot \frac{\prod_j \alpha_{jB}(max) - \alpha_{jB}(min)}{\prod_j \alpha_{jA}(max) - \alpha_{jA}(min)}$$

prior probability of each
model: maximum entropy \rightarrow 1:1

fit quality: integral over χ^2

Occam's Razor: simple models are
preferred

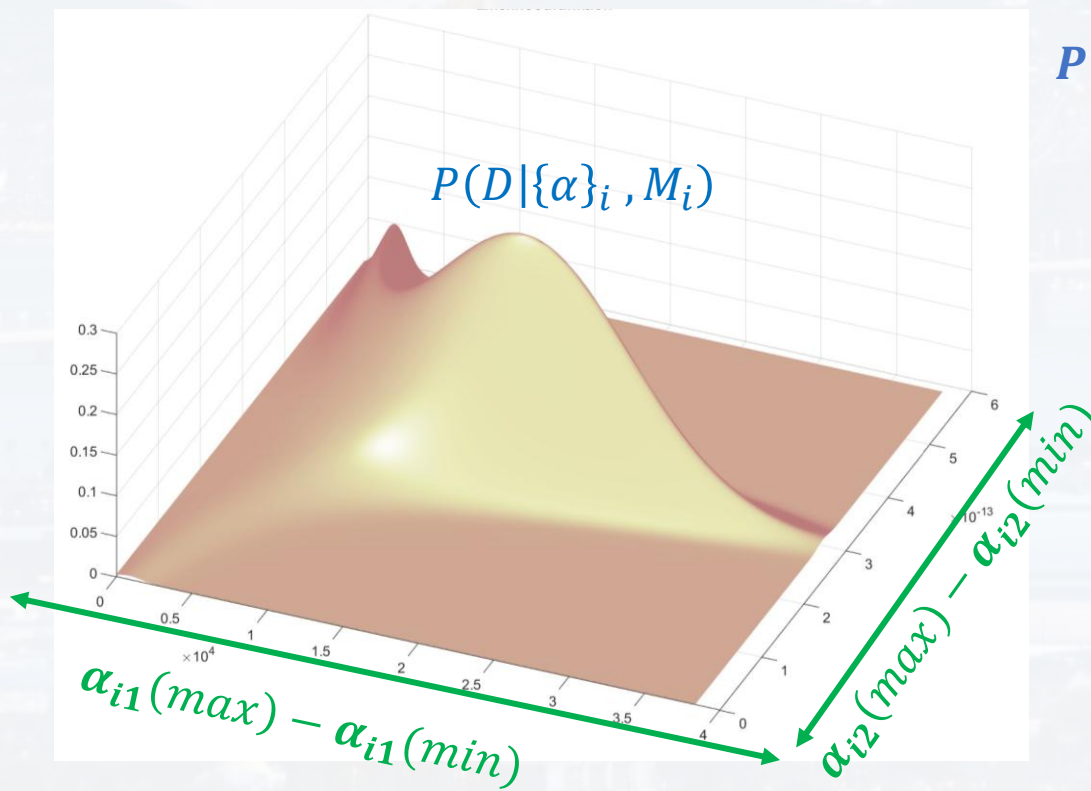


$$\rho = \frac{P(D|M_A) P(M_A)}{P(D|M_B) P(M_B)}$$

$$= \frac{P(M_A)}{P(M_B)} \cdot$$

$$\frac{\int e^{-\frac{1}{2}\chi_A^2} d\Omega_{\{\alpha\}_A}}{\int e^{-\frac{1}{2}\chi_B^2} d\Omega_{\{\alpha\}_B}} \cdot \frac{\prod_j \alpha_{jB}(\max) - \alpha_{jB}(\min)}{\prod_j \alpha_{jA}(\max) - \alpha_{jA}(\min)}$$

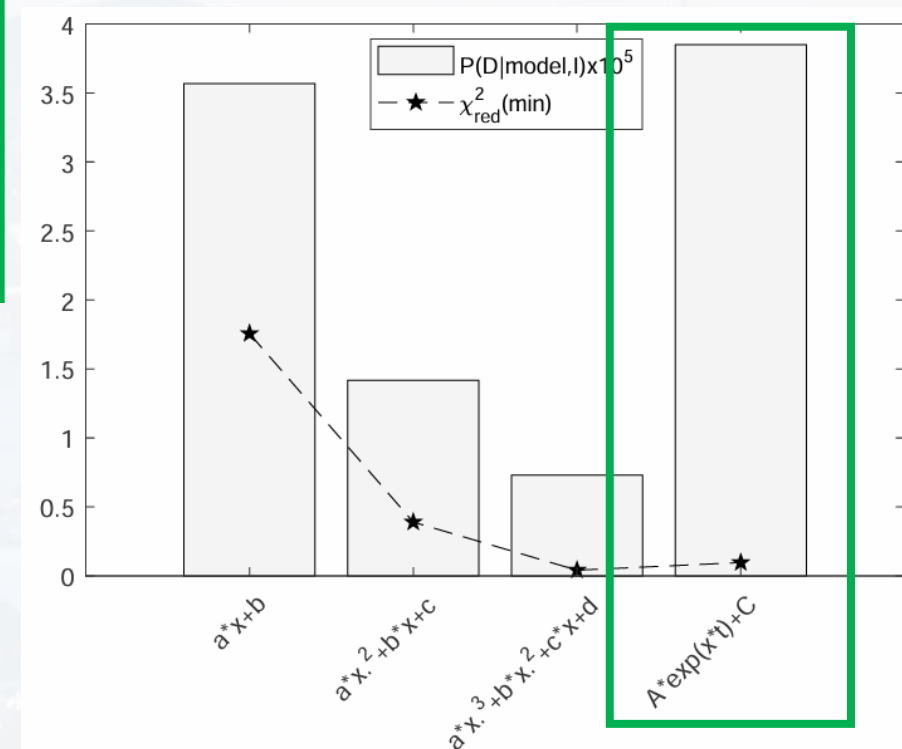
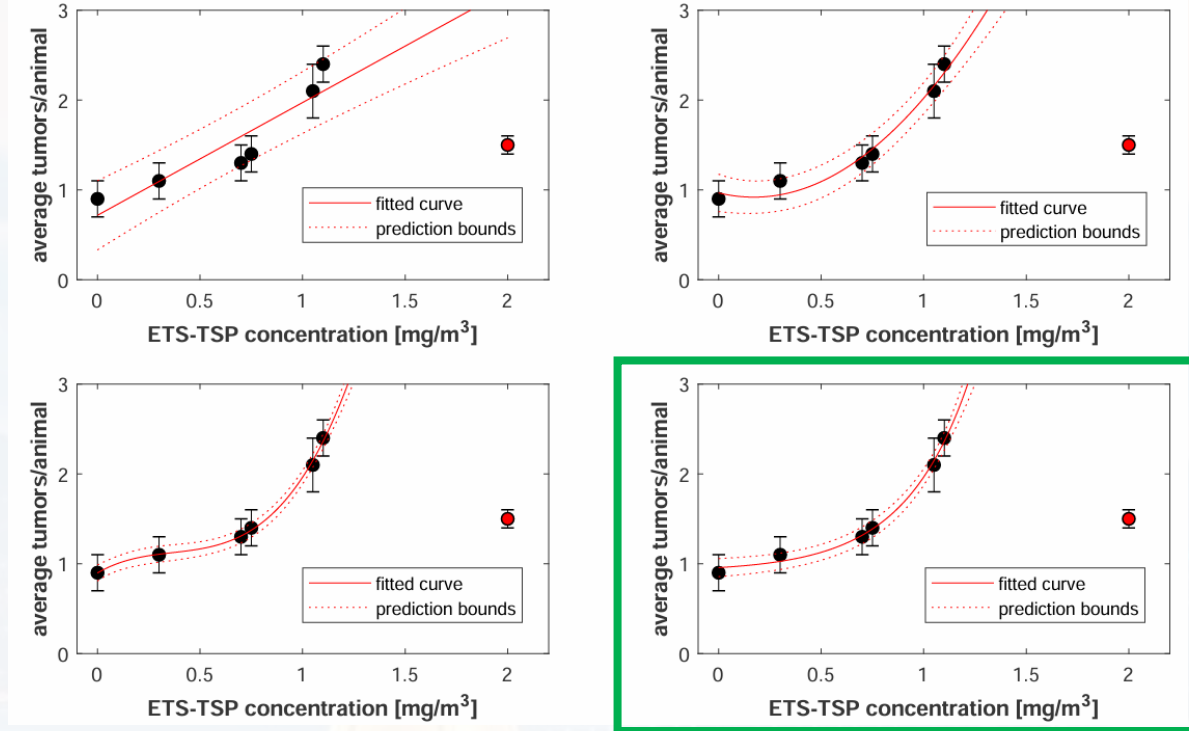
D	: data
M_A	: model A
M_B	: model B
{α}_i	: all parameter of model M_i
y_k	: measured value
σ_k	: error
ȳ(M_i)_k	: model value (after fit)



$$P(D|M_i) = \int P(D|\{\alpha\}_i, M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i}$$

$$= \int P(D|\{\alpha\}_i, M_i) \prod_j P(\alpha_{ij} | M_i) d\alpha_{ij}$$

$$= \prod_j \frac{1}{\alpha_{ij}(\max) - \alpha_{ij}(\min)} \cdot \int P(D|\{\alpha\}_i, M_i) d\alpha_{ij}$$





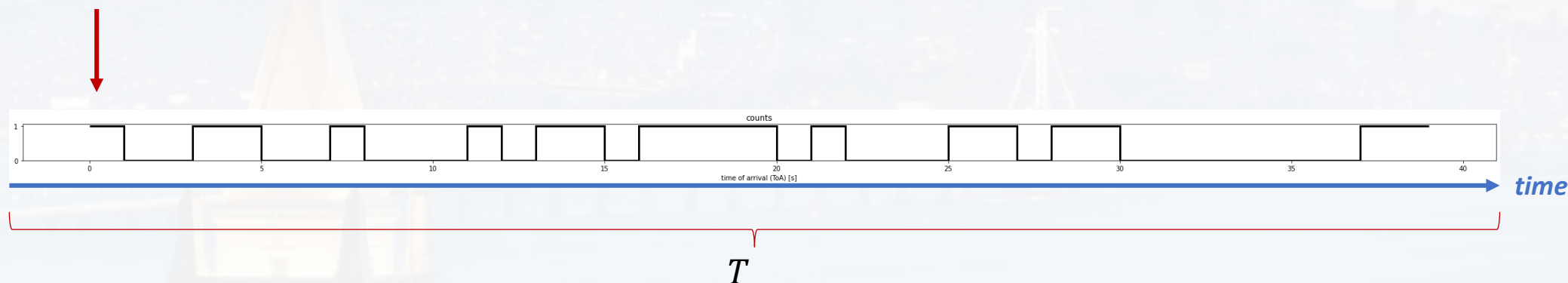
The key part is the likelihood function!

$$\rho = \frac{P(D|M_A) P(M_A)}{P(D|M_B) P(M_B)}$$

$$P(D|M_i) = \int P(D|\{\alpha\}_i M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i}$$

$$P(n|r(t)) = \frac{(r(t) \cdot \Delta t)^n}{n!} e^{-r(t) \cdot \Delta t}$$

now: **Poisson** distribution, see also max. ent. distributions



M_A : constant model (no signal, just noise)

$\rightarrow r(t) = \text{const}$

M_B : signal of unknown phase, amplitude & frequency

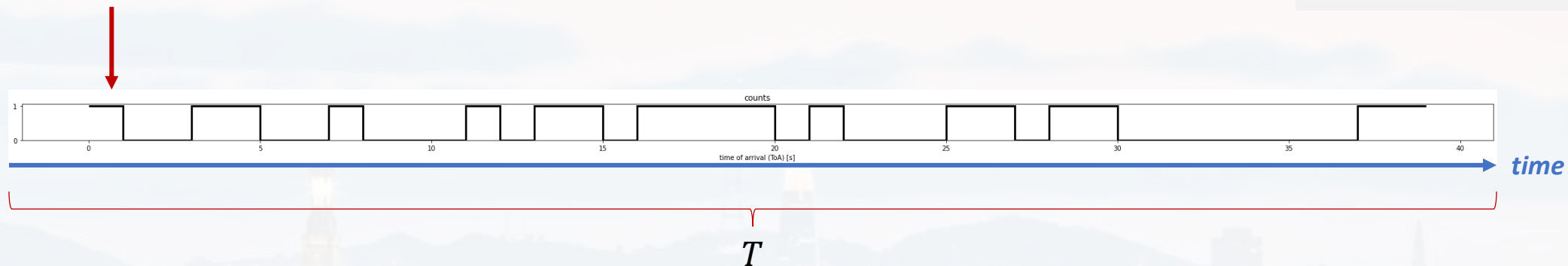
$\rightarrow r(t) = f(t)$



$$P(n|r(t)) = \frac{(r(t) \cdot \Delta t)^n}{n!} e^{-r(t) \cdot \Delta t}$$

Poisson distribution

$r(t)$:	rate
Δt :	time resolution
n :	number of events
T :	obs. time span
D :	data set



actual data (ToA)

N intervals with $n = 1$
 Q intervals with $n = 0$

$$(N + Q)\Delta t = T$$

$$P(D| r(t), t) = \prod_{i=1}^N r(t_i) \cdot \Delta t e^{-r(t_i) \cdot \Delta t} \cdot \prod_{i=1}^Q e^{-r(t_i) \cdot \Delta t}$$

$$P(D|M_i) = \int P(D|\{\alpha\}_i M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i}$$

$$P(D| r(t), t) = (\Delta t)^N \cdot \prod_{i=1}^N r(t_i) \cdot \exp \left[- \sum_{i=1}^{Q+N} r(t_i) \cdot \Delta t \right]$$



$$P(n|r(t)) = \frac{(r(t) \cdot \Delta t)^n}{n!} e^{-r(t) \cdot \Delta t}$$

Poisson distribution

$r(t)$:	rate
Δt :	time resolution
n :	number of events
T :	obs. time span
D :	data set

N intervals with $n = 1$
 Q intervals with $n = 0$

$$(N + Q)\Delta t = T$$

$$P(D| r(t), t) = (\Delta t)^N \cdot \prod_{i=1}^N r(t_i) \cdot \exp \left[- \underbrace{\sum_{i=1}^{Q+N} r(t_i) \cdot \Delta t}_{= \int_0^T r(t) dt} \right]$$

$$P(D| r(t), t) = (\Delta t)^N \cdot \prod_{i=1}^N r(t_i) \cdot \exp \left(- \int_0^T r(t) dt \right)$$



$$P(D | r(t), t) = (\Delta t)^N \cdot \prod_{i=1}^N r(t_i) \cdot \exp\left(-\int_0^T r(t) dt\right)$$

$r(t)$:	rate
Δt :	time resolution
n :	number of events
T :	obs. time span
D :	data set

m phase bins:

r_j

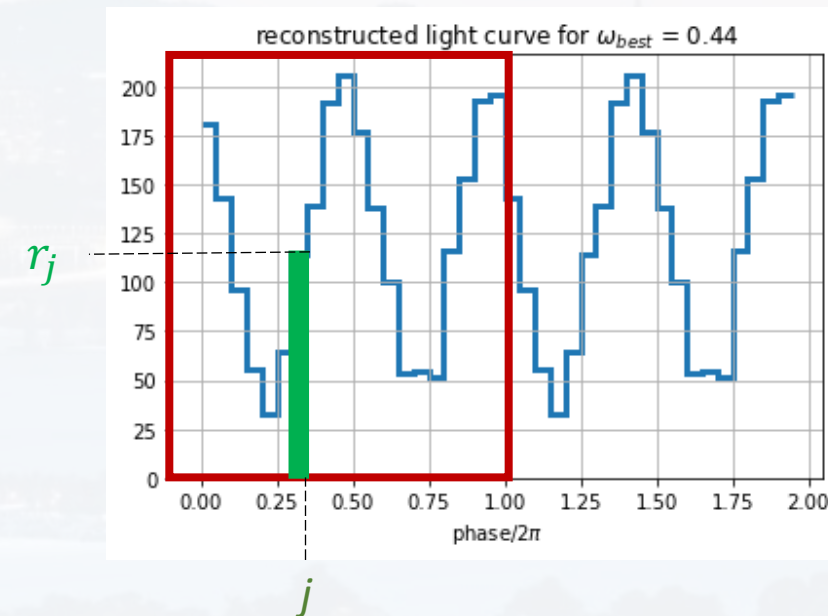
rate in each phase bin j

$$A = \frac{1}{m} \sum_{j=1}^m r_j$$

average rate

$$f_j = \frac{r_j}{\sum_{j=1}^m r_j} = \frac{r_j}{A m}$$

fraction of total rate in
each phase bin j



Each light curve of any shape is being fully described by f_j



$$P(D | r(t), t) = (\Delta t)^N \cdot \prod_{i=1}^N r(t_i) \cdot \exp\left(-\int_0^T r(t) dt\right)$$

constant model: $r_j = \text{const } \forall j$

$\rightarrow r_j = A$

$$P(D | r(t), t) = (\Delta t)^N \cdot A^N \cdot e^{-AT}$$

$r(t)$:	rate
Δt :	time resolution
n :	number of events
T :	obs. time span
D :	data set
N :	number of intervals with $n=1$
r_j	rate in each phase bin j
A :	average rate
m :	number of phase bins
f_j :	fraction of total rate in j

actual signal

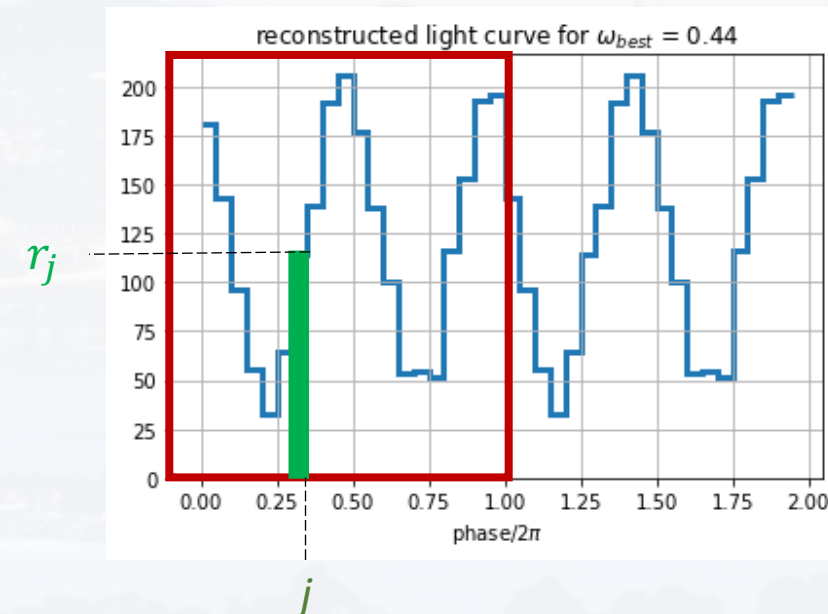
- amplitude
- phase
- frequency
- offset

detection

- t_i

analysis

- phase
- frequency
- $f_j(A, m)$





M_A (*constant*): $P(D | r(t), t) = (\Delta t)^N \cdot A^N \cdot e^{AT}$

M_B (*signal*): $P(D | r(t), t) = (\Delta t)^N \cdot \prod_{i=1}^N r(t_i) \cdot \exp\left(-\int_0^T r(t) dt\right)$

$$P(D | M_i) = \int P(D | \{\alpha\}_i | M_i) P(\{\alpha\}_i | M_i) d\Omega_{\{\alpha\}_i}$$

$$P(\omega, \varphi, A, f | M_i) = P(\omega | M_i) P(\varphi | M_i) P(A | M_i) P(f | M_i)$$

max entropy:

$$P(\omega | M_i) = \frac{1}{\omega \ln(\omega_{max}/\omega_{min})}$$

$$\omega_{max} = \frac{2\pi N}{T}$$

$$\omega_{min} = \frac{2\pi}{T}$$

in practice: $\omega_{min} = 10 \frac{2\pi}{T}$

$$P(\varphi | M_i) = \frac{1}{2\pi}$$

$$P(A | M_i) = \frac{1}{A_{max}}$$

$$P(f | M_i) = (m-1)! \delta\left(1 - \sum_{j=1}^m f_j\right)$$

$r(t)$:	rate
Δt :	time resolution
n :	number of events
T :	obs. time span
D :	data set
N :	number of intervals with $n=1$
r_j :	rate in each phase bin j
A :	average rate
m :	number of phase bins
f_j :	fraction of total rate in j
ω :	frequency
φ :	phase



you find the python package BayesianSignalDetect.py [here](#)

```
from BayesianSignalDetect import *
```

a *decorator* that measures runtime of a function

function creates a test dataset for illustration

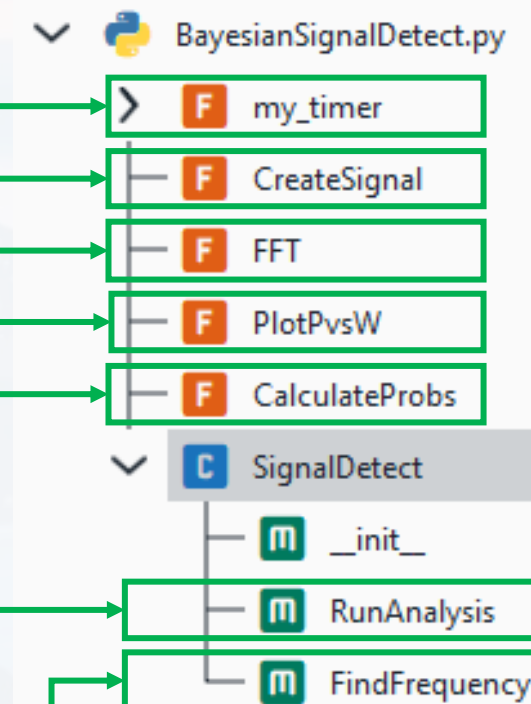
FFT for comparison

plot routine for *periodogram*

calculates $P(D|\omega, \varphi, A, m)$

calls *CalculateProbs* and calculates $P(D|M_i)$

calls main part of the code – runs *RunAnalysis* on multiple cpus in parallel



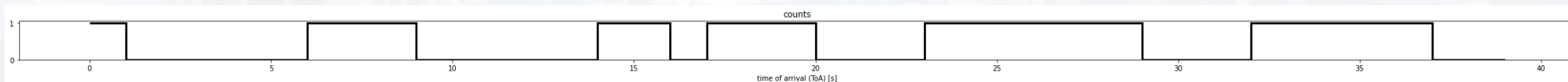
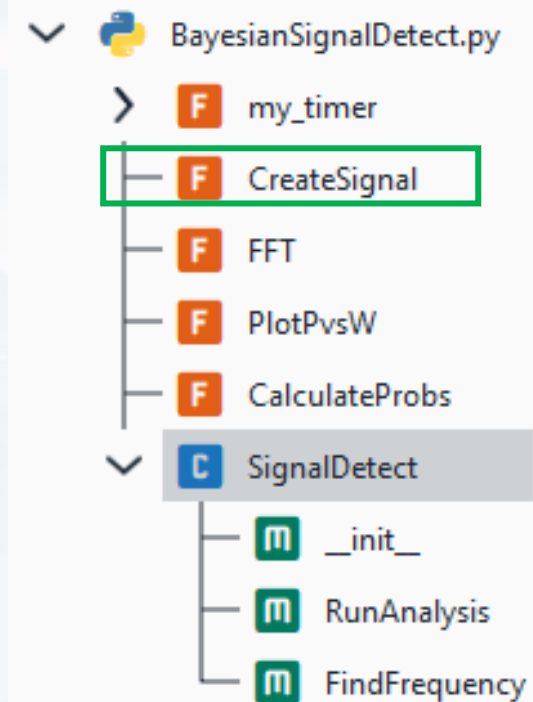
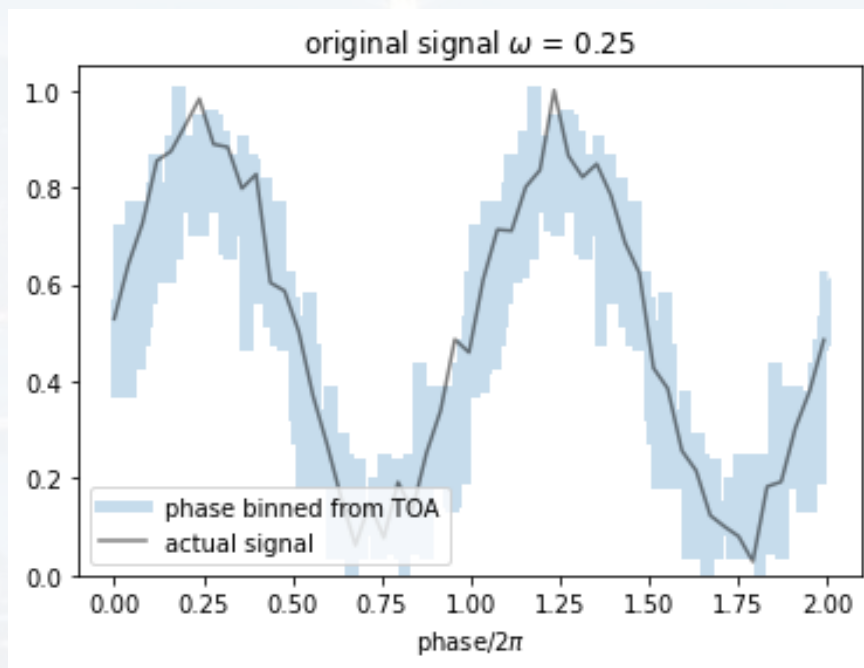


you find the python package BayesianSignalDetect.py [here](#)

```
from BayesianSignalDetect import *
```

```
T = CreateSignal(5000, 0.25, 0.1)
```

$N + Q$



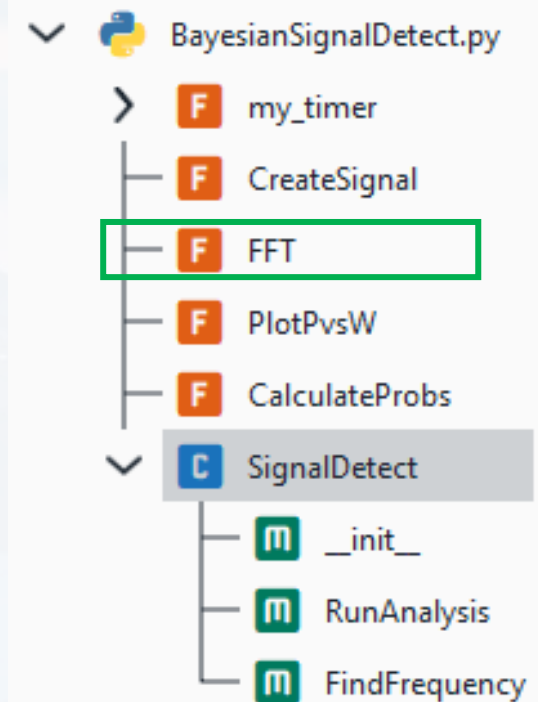
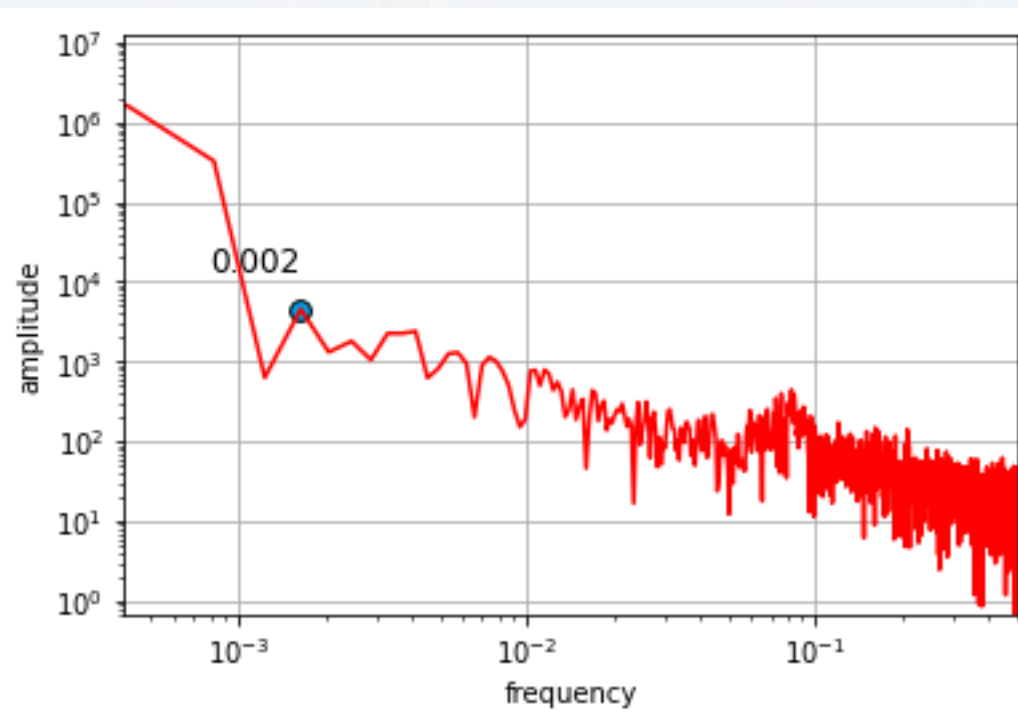


you find the python package BayesianSignalDetect.py [here](#)

```
from BayesianSignalDetect import *
```

```
T = CreateSignal(5000, 0.25, 0.1)
```

```
FFT(T)
```



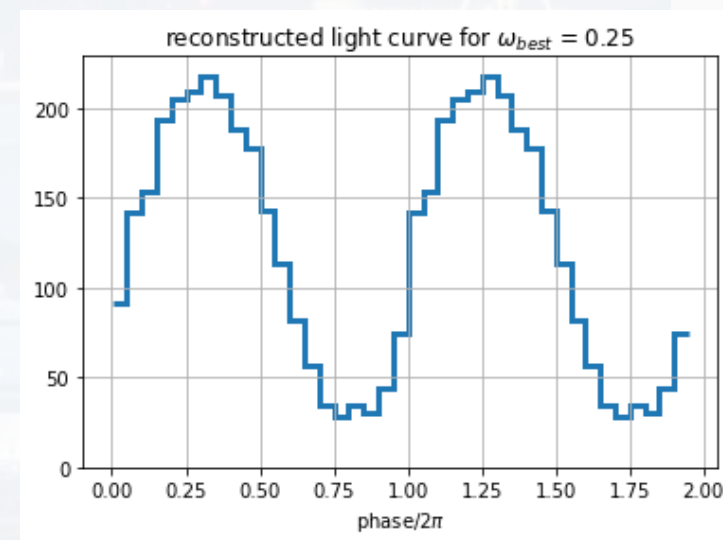
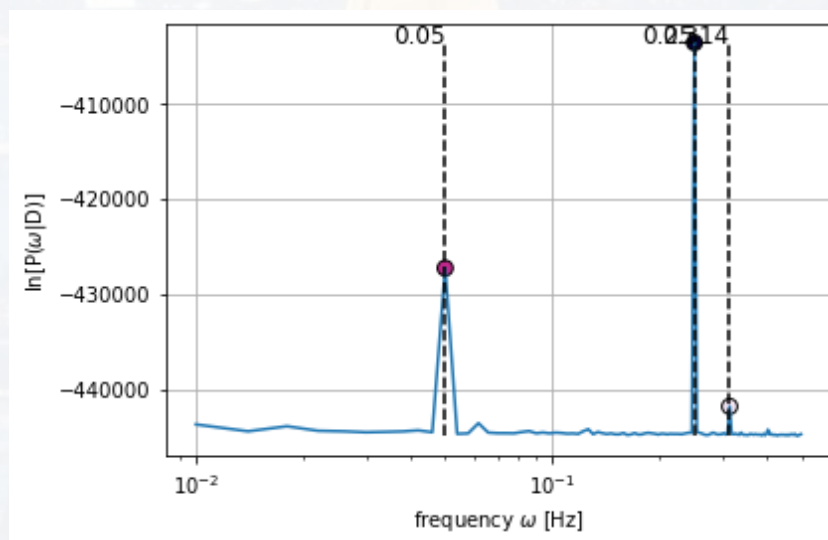
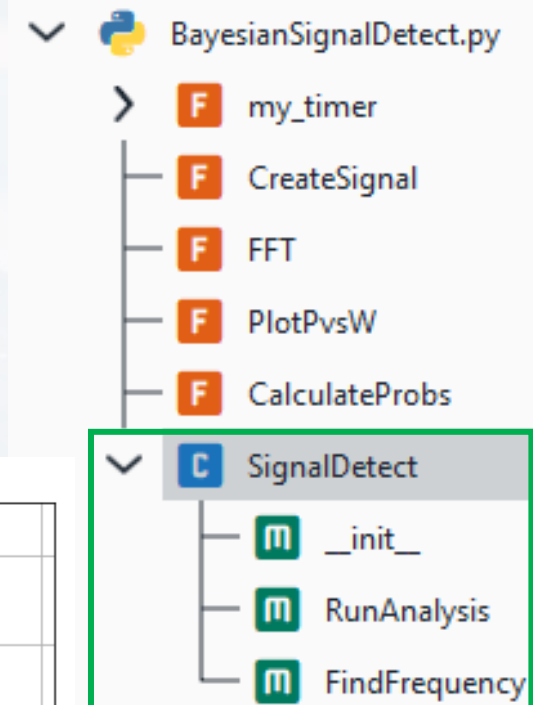


you find the python package BayesianSignalDetect.py [here](#)

```
from BayesianSignalDetect import *
```

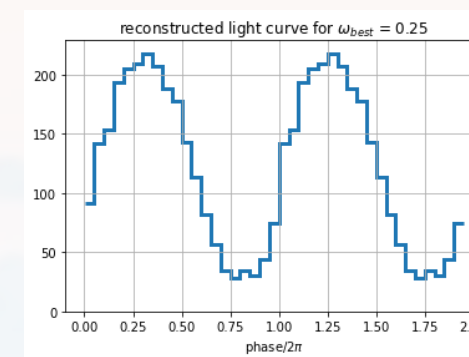
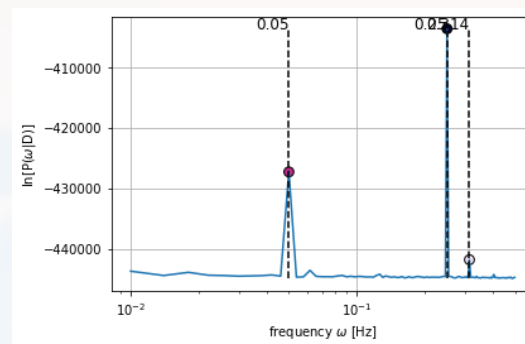
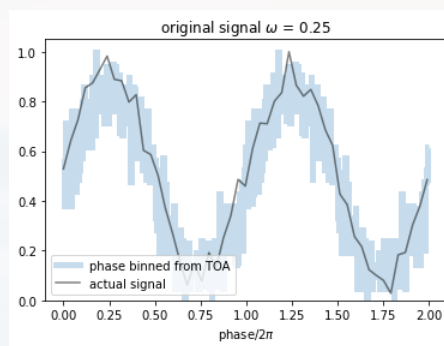
```
T = CreateSignal(5000, 0.25, 0.1)
```

```
S = SignalDetect(T, w_end = 0.5, w_start = 0.01)  
[Omega, P] = S.FindFrequency()
```

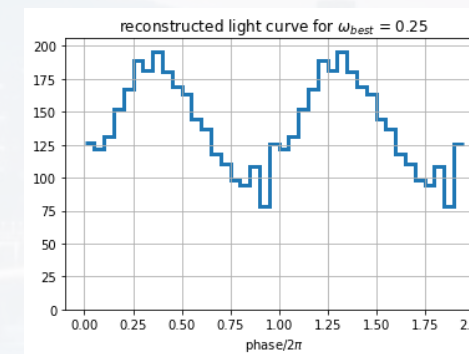
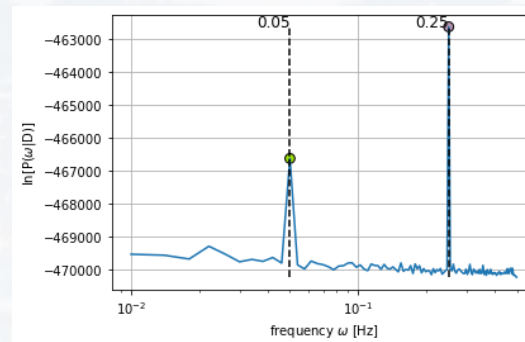
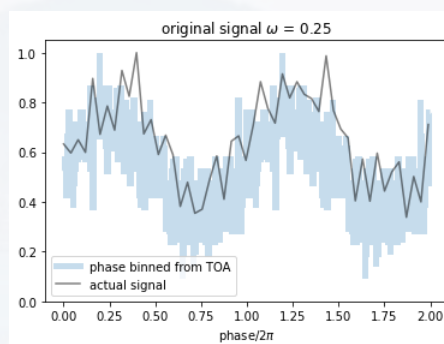




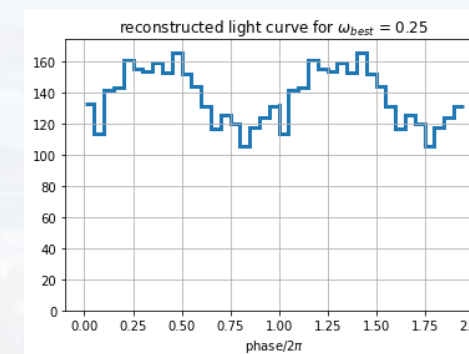
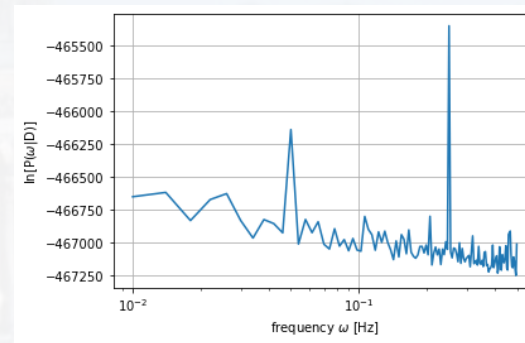
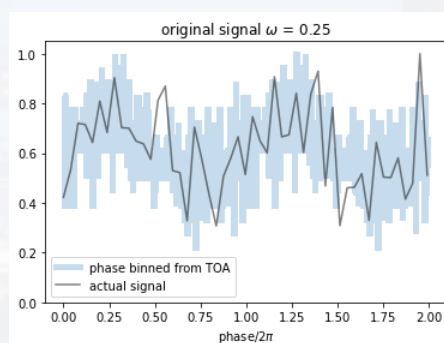
$T = \text{CreateSignal}(5000, 0.25, 0.1)$



$T = \text{CreateSignal}(5000, 0.25, 0.5)$



$T = \text{CreateSignal}(5000, 0.25, 1)$



Thank you for your attention