Berkeley MSSE
MOLECULAR SCIENCE & SOFTWARE ENGINEERING

Lecture 02:

Bayesian Methods

Markus Hohle

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units

Course Map

Lecture 1: Course Overview and Introduction to Machine Learning

classic ML tools & algorithms

Lecture 2: Bayesian Methods in Machine Learning

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

ANNs/AI/Deep Learning

Lecture 7: Introduction to Artificial Neural Networks - The Perceptron

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)

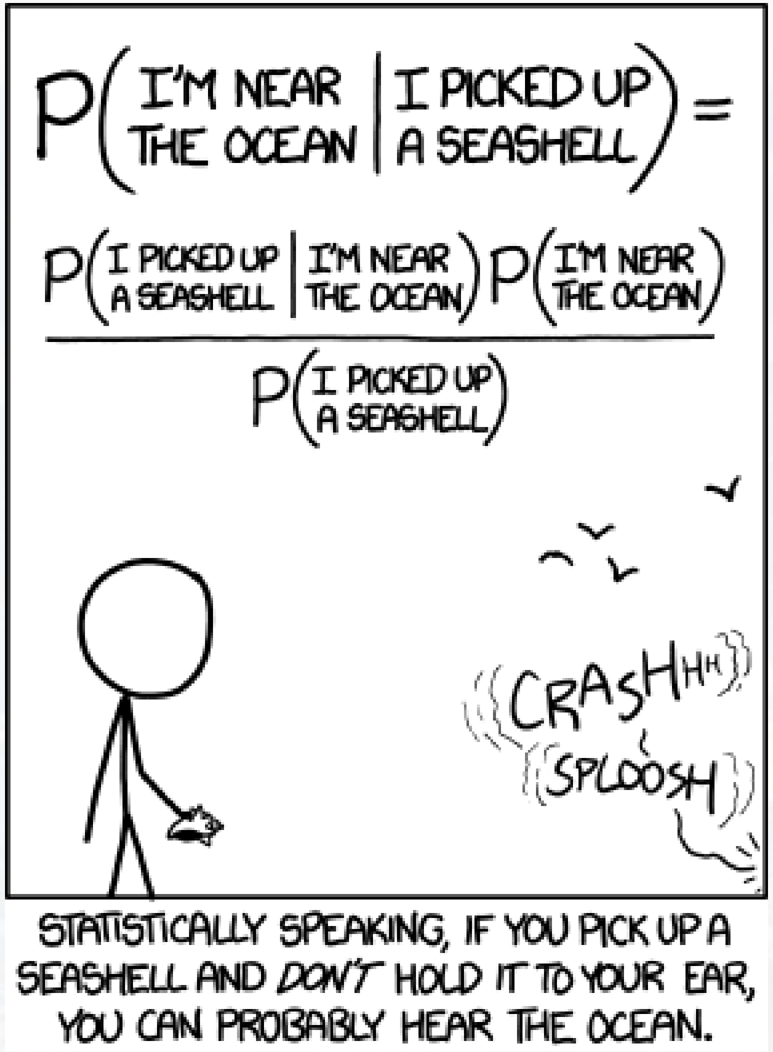
Lecture 12: Combining LSTMs and CNNs

Lecture 13: Running Models on GPUs and Parallel Processing

Lecture 14: Project Presentations

Lecture 15: Transformer

Lecture 16: GNN

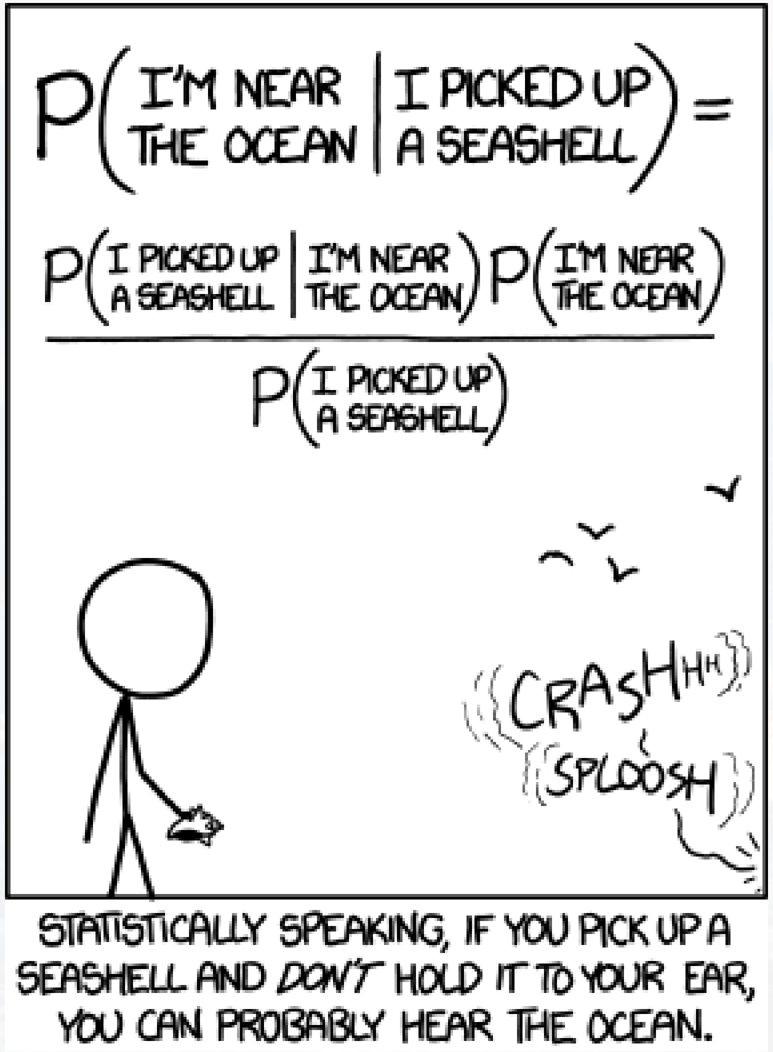


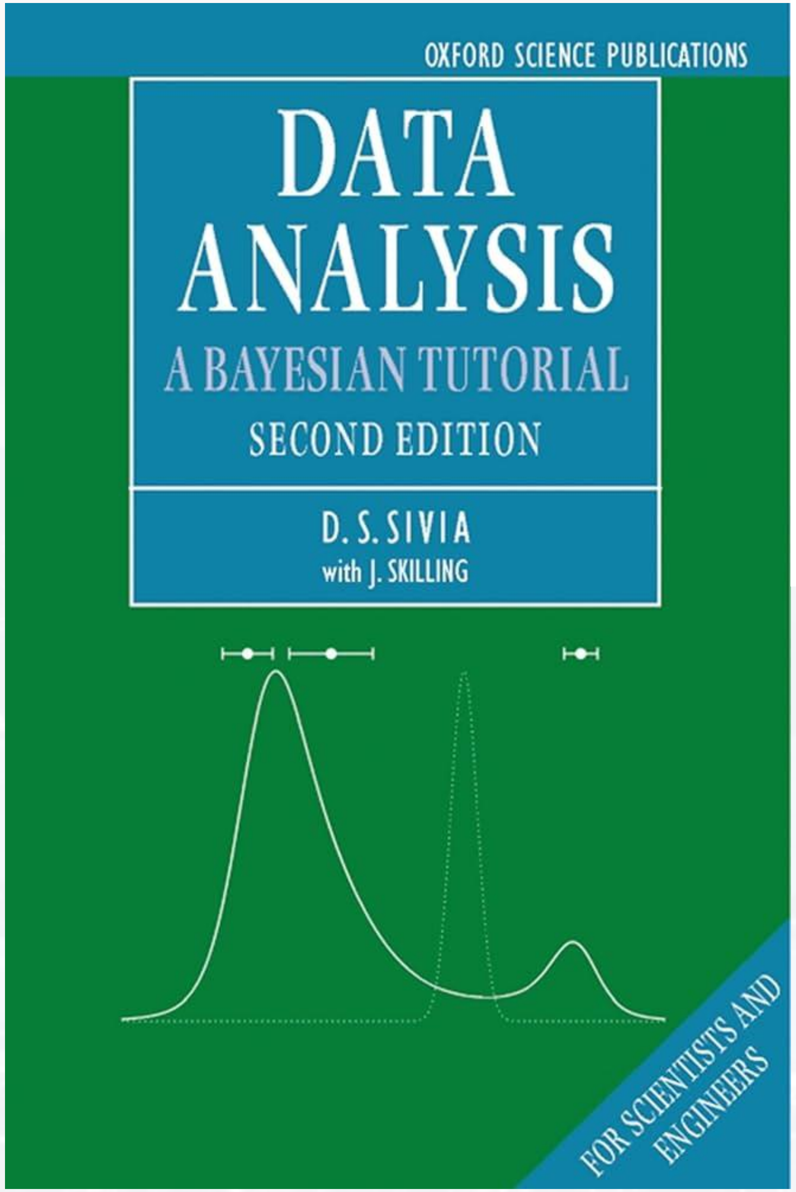
Outline

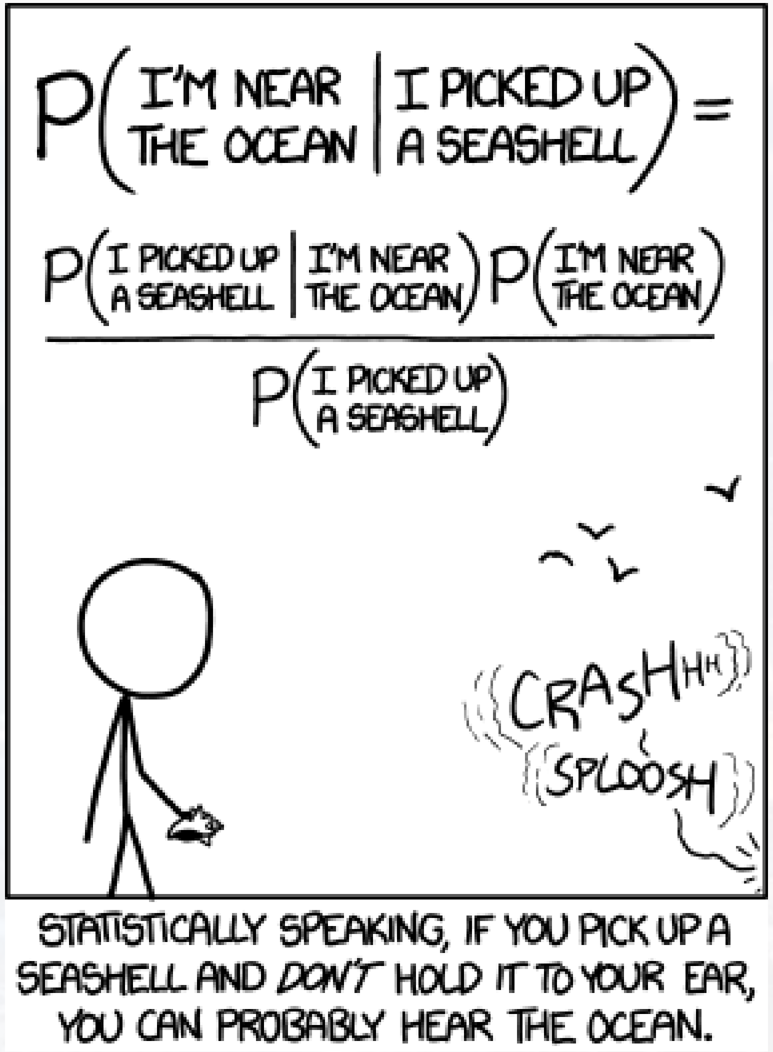
- The Idea and Bayes Theorem
- Naïve Bayes
- Parameter Estimation
- Model Selection

FYI

- Bayesian Networks (Graphs)
- Variational Bayes


P(I'M NEAR THE OCEAN | I PICKED UP A SEASHELL) =
P(I PICKED UP A SEASHELL | I'M NEAR THE OCEAN) P(I'M NEAR THE OCEAN)
P(I PICKED UP A SEASHELL)
CRASHHH
SPLOOSH
STATISTICALLY SPEAKING, IF YOU PICK UP A SEASHELL AND DON'T HOLD IT TO YOUR EAR, YOU CAN PROBABLY HEAR THE OCEAN.


OXFORD SCIENCE PUBLICATIONS
DATA ANALYSIS
A BAYESIAN TUTORIAL
SECOND EDITION
D. S. SIVIA
with J. SKILLING
FOR SCIENTISTS AND ENGINEERS



<u>Outline</u>

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

<u>FYI</u>

- Bayesian Networks (Graphs)
- Variational Bayes

Bayesian Methods:

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 constrains** (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

Bayesian Methods:

The Idea and Bayes Theorem

Thomas Bayes
(1701 - 1761)

$P(A \cap B)$ 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"

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

$$\longrightarrow \boxed{P(A|B)P(B) = P(B|A)P(A)}$$

Bayes Theorem

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

Bayesian Methods:

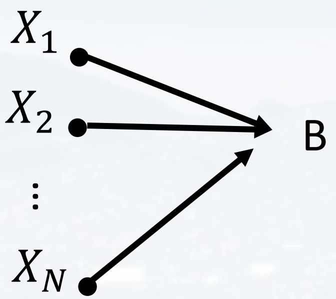
The Idea and Bayes Theorem

$$\boxed{P(A|B)P(B) = P(B|A)\mathrm{P(A)}}$$

Bayes Theorem

Thomas Bayes
(1701 - 1761)

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



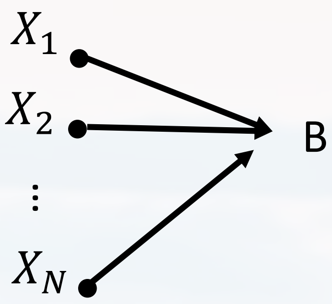
$$\left.\begin{aligned} P(B) &= \sum_{n=1}^{N} P(B|X_n)P(X_n) \\ P(B) &= \int P(B|X)P(X)\,dX \end{aligned}\right\}$$ **marginalization**

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

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

Bayesian Methods:

The Idea and Bayes Theorem



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

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

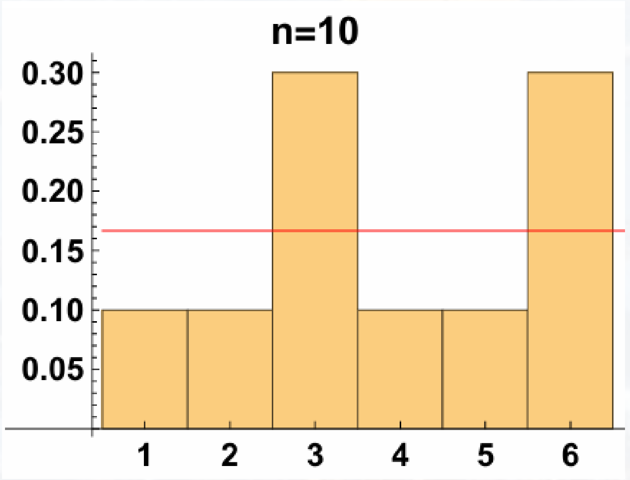
marginalization

Thomas Bayes (1701 - 1761)

model: M
data: D

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

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



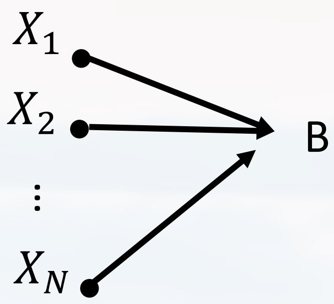
σ = 2, μ = 3.5

σ = 2, μ = 5.0

σ = 1.5, μ = 3.5

σ = 7.0, μ = 1.0 ....and so on

Bayesian Methods:



$$\underbrace{P(B) = \sum_{n=1}^{N} P(B|X_n)P(X_n) \qquad P(B) = \int P(B|X)P(X)\, dX}_{\text{marginalization}}$$

Thomas Bayes
(1701 - 1761)

example:

model: M
data: D

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

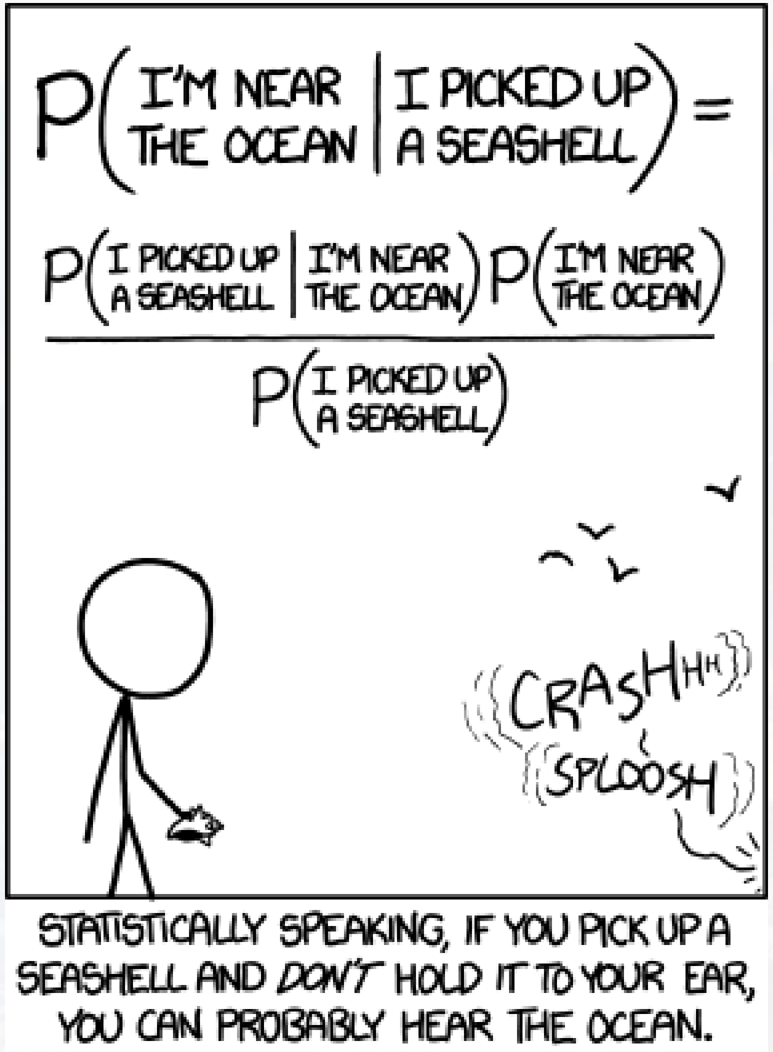
for a normal distribution $\mathcal{N}(\mu, \sigma)$

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

for a Poisson distribution $p(\lambda)$

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

and so on…



Outline

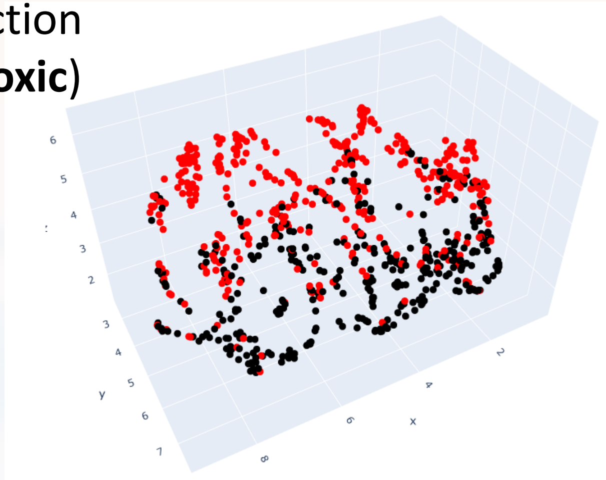
- The Idea and Bayes Theorem
- **Naïve Bayes**
- Parameter Estimation
- Model Selection

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)

Bayesian Methods:

$\vec{x}$: 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)

$P(C_k|\vec{x})$: 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

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

Bayes Theorem

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

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

Naïve Bayes:

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

Bayesian Methods:

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

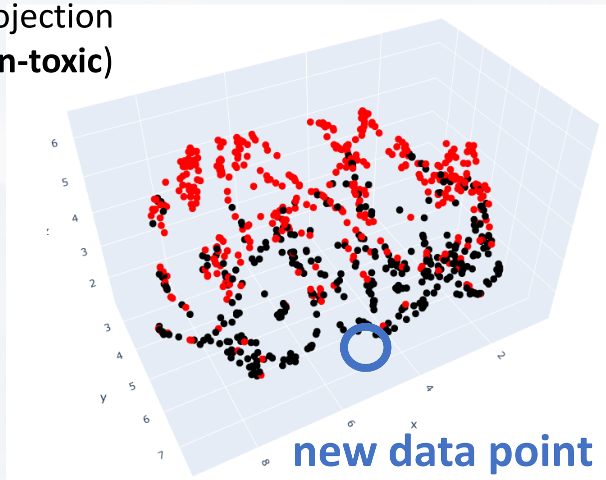
$$\sim P(C_k)\prod_{i=1}^{I} P(x_i|C_k)$$

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

$P(C_k|\vec{x})$: **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**)



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

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

from the training data → supervised learning

different models for $P(x_i|C_k)$

- multinomial
- Gaussian
- …

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)
```

Bayesian Methods:

Python:

```
from sklearn.naive_bayes import *
from sklearn.preprocessing import MinMaxScaler
```

importing methods for naïve bayes

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
```

Bayesian Methods:


```
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

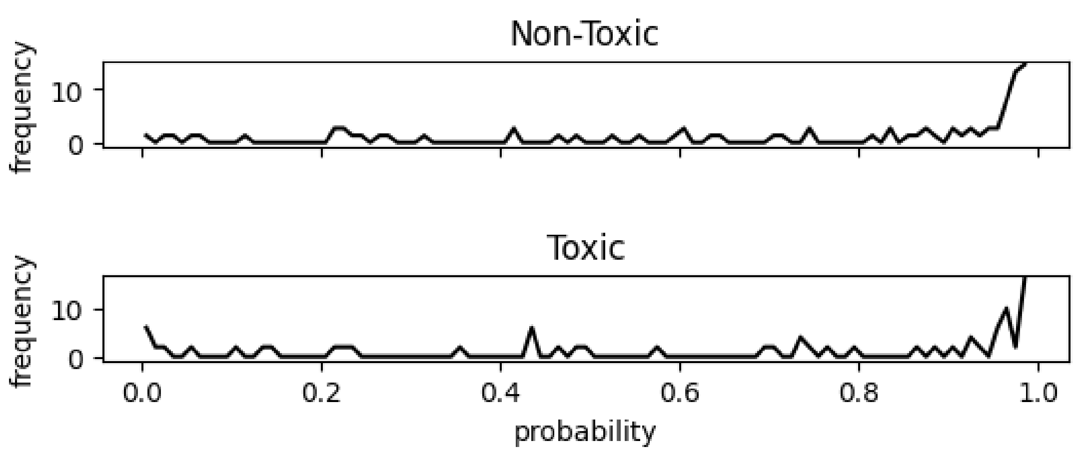
```
Ypred   = Fit.predict(XTestS)
Probs   = Fit.predict_proba(XTestS)
```

predicting the class C_k and calculating the probabilities

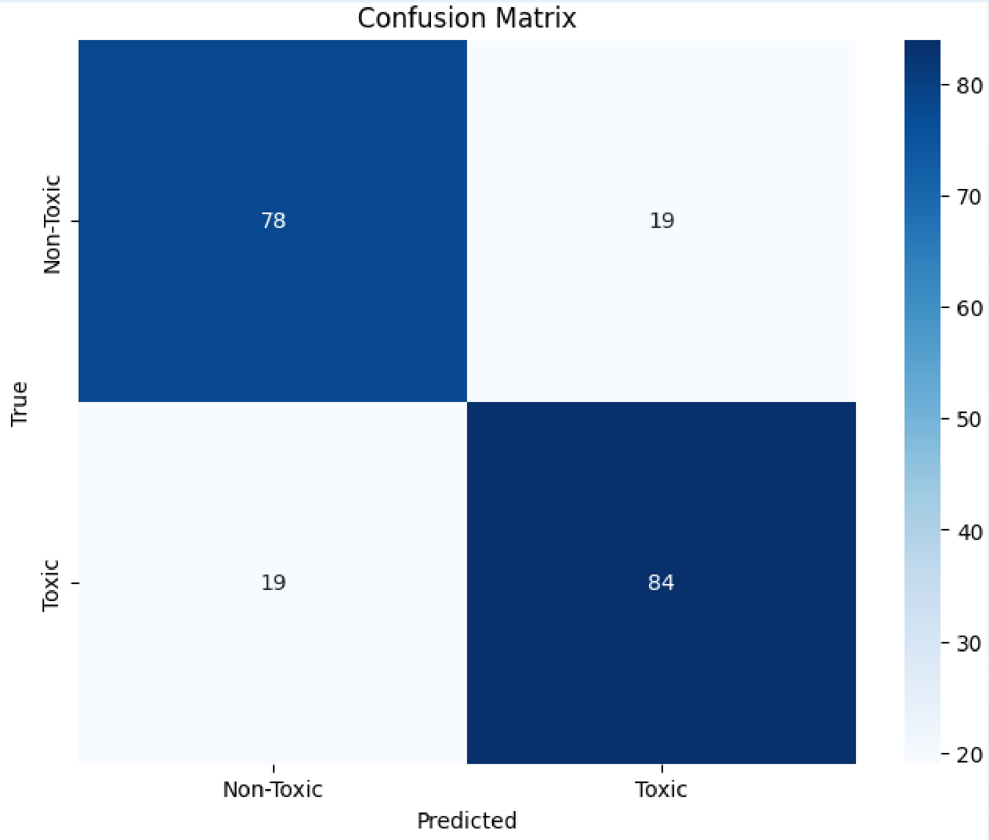
Bayesian Methods:

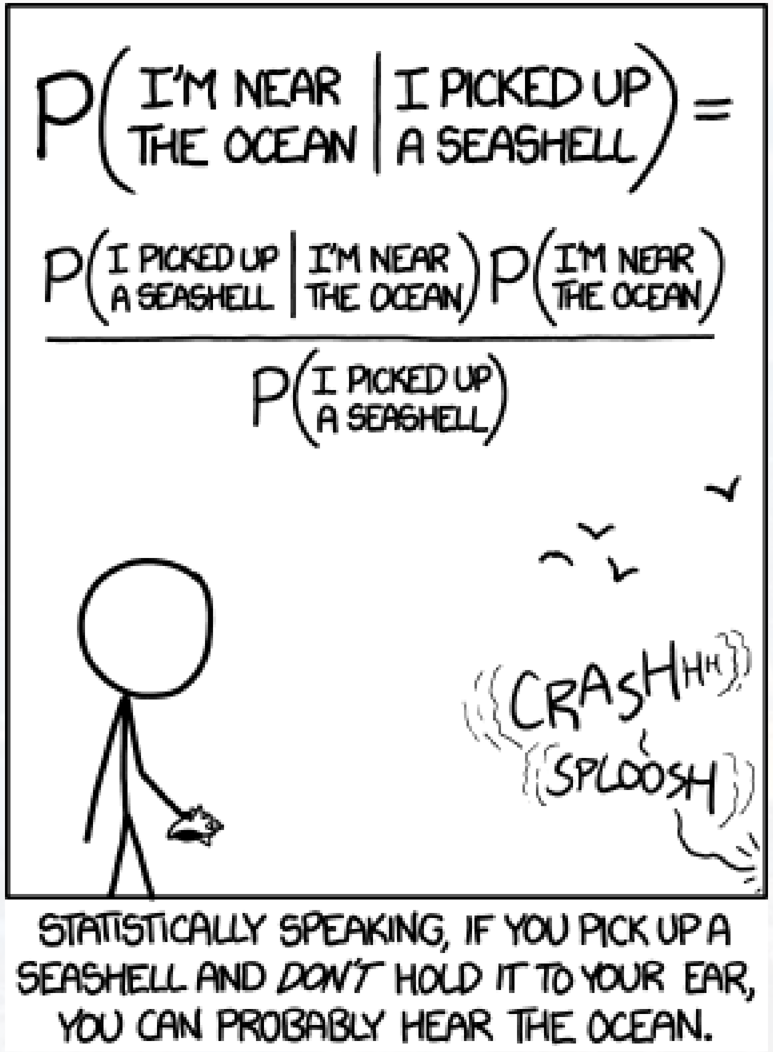

```
XTestS  = scaler.transform(XTest)

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

evaluation



see
`Walk_Through_NaiveBayes.ipynb`





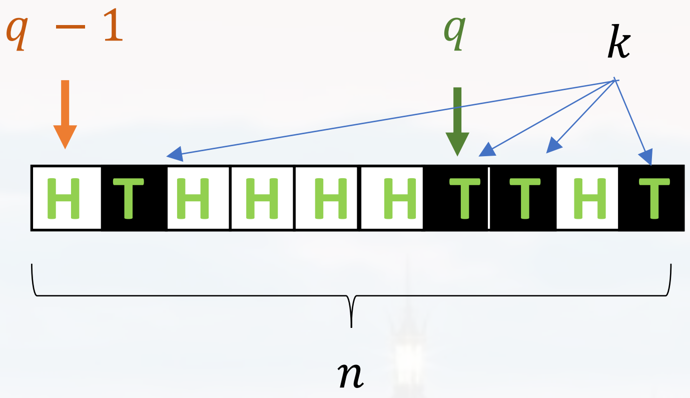
Outline

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

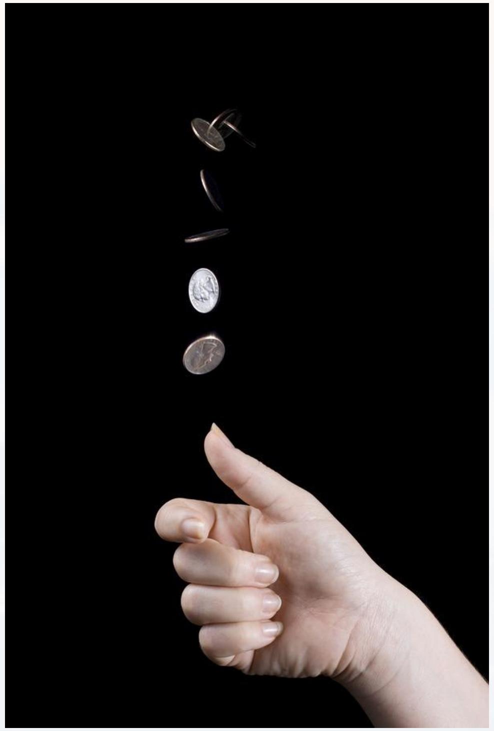
FYI

- Bayesian Networks (Graphs)
- Variational Bayes

Bayesian Methods:



$\boldsymbol{q} = ?$

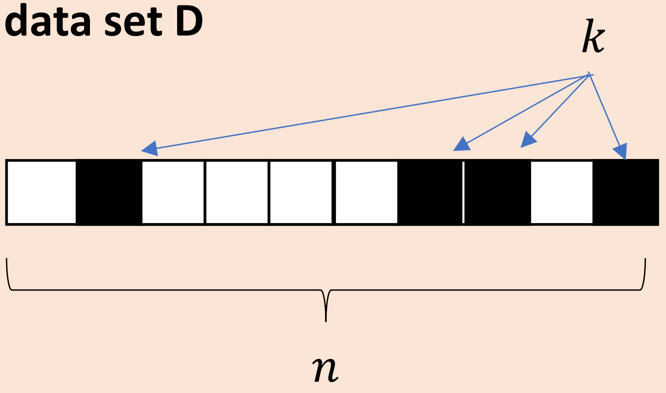


fair coin? q = 0.5 ???

mutation q = ??

Bayesian Methods:

data set D



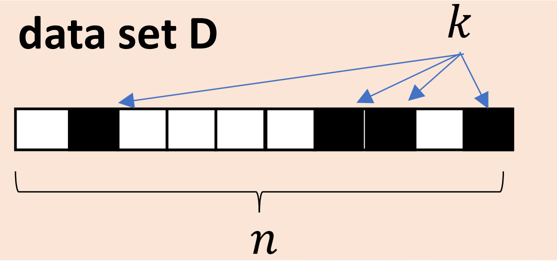
$\boldsymbol{q} = ?$

goal:

- **P(q|D)**
- the larger ***D***, the more certain ***q***

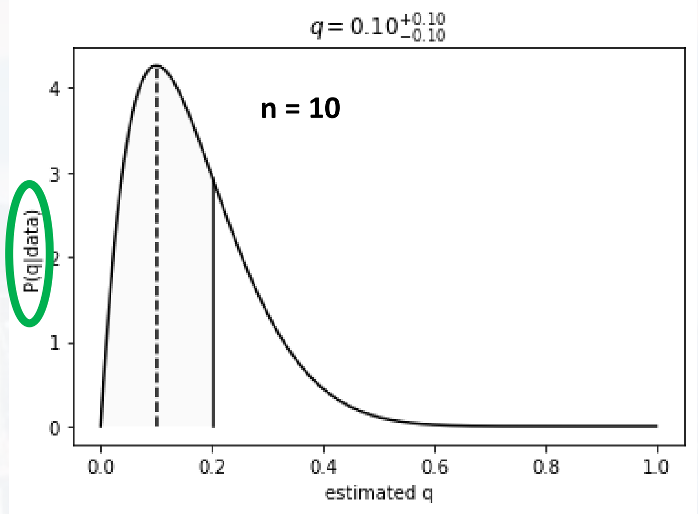
  → learning

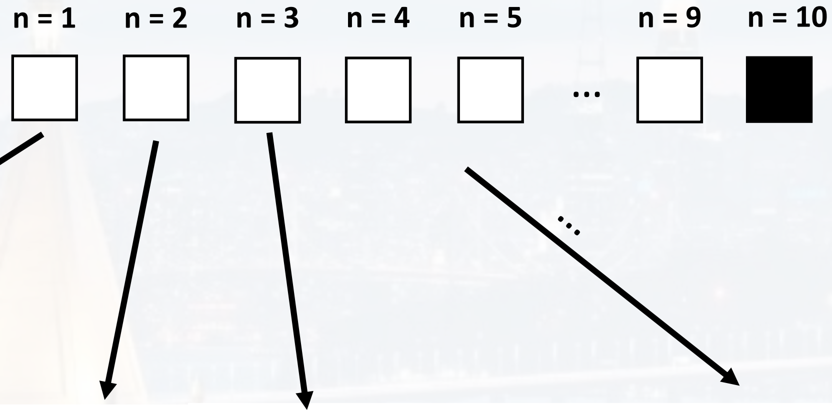
Bayesian Methods:

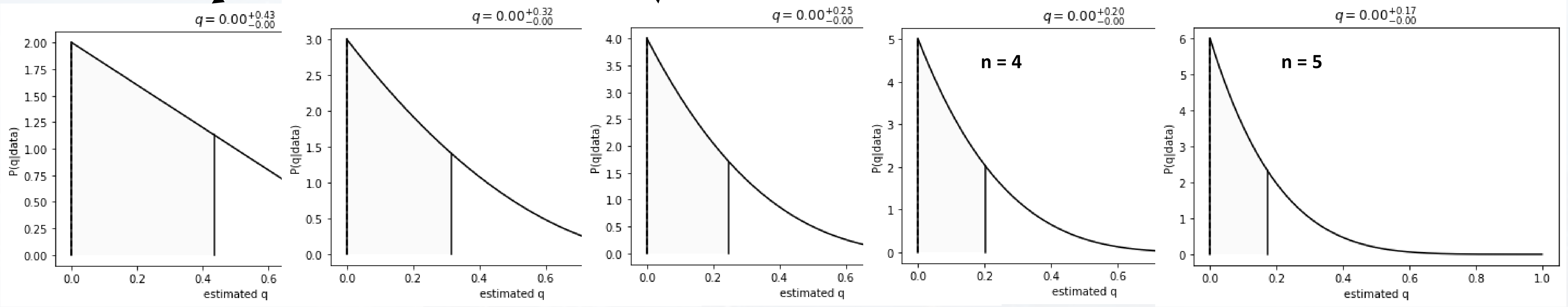


$q = ?$

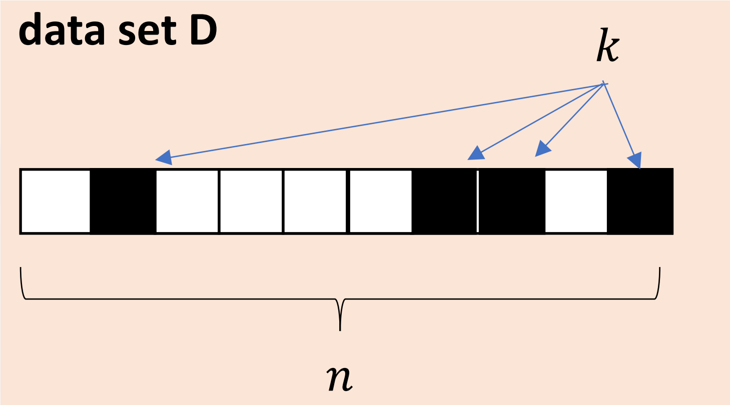
goal:
- **P(q|D)**
- the larger ***D***, the more certain ***q***
  → learning







Bayesian Methods:



$\boldsymbol{q} = ?$

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)**

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

prior

evidence **(const wrt q)**

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

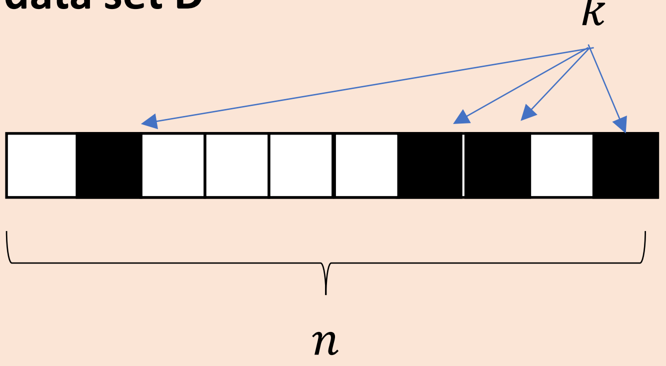
$$\sim q^k (1-q)^{n-k}\ P(q)$$

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

Bayesian Methods:

Parameter Estimation

data set D



$\boldsymbol{q} = ?$

goal:
- **P(q|D)**
- the larger ***D***, the more certain ***q***
  → learning

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

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

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

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

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

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

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

Bayesian Methods:

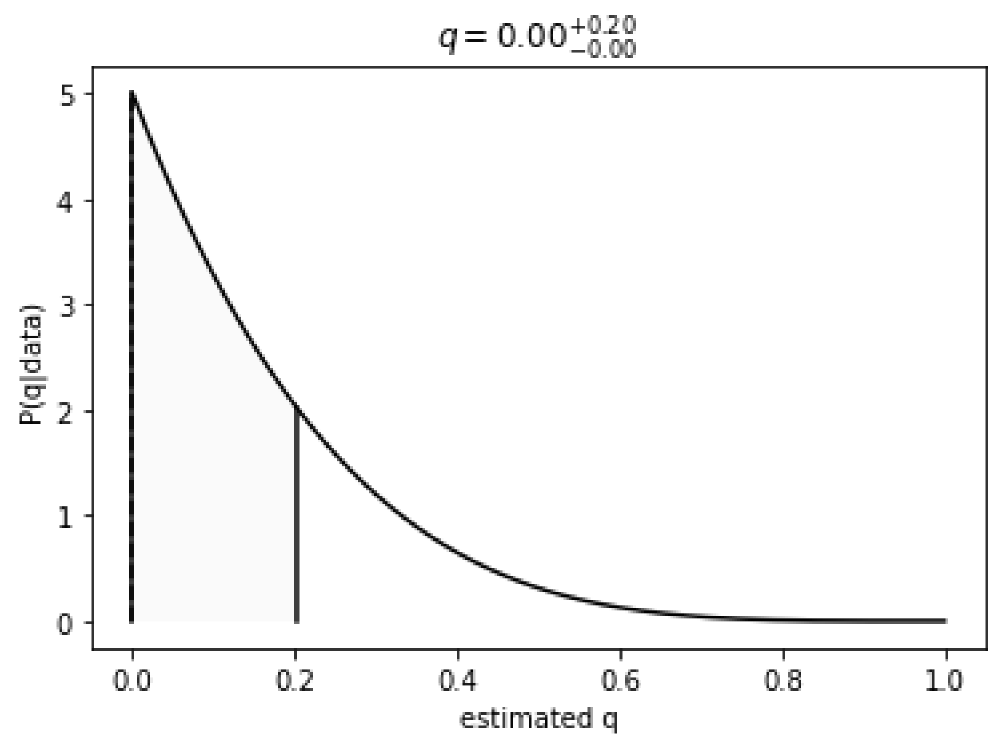
check out `bayesian_bino.py`

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

```
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)
```

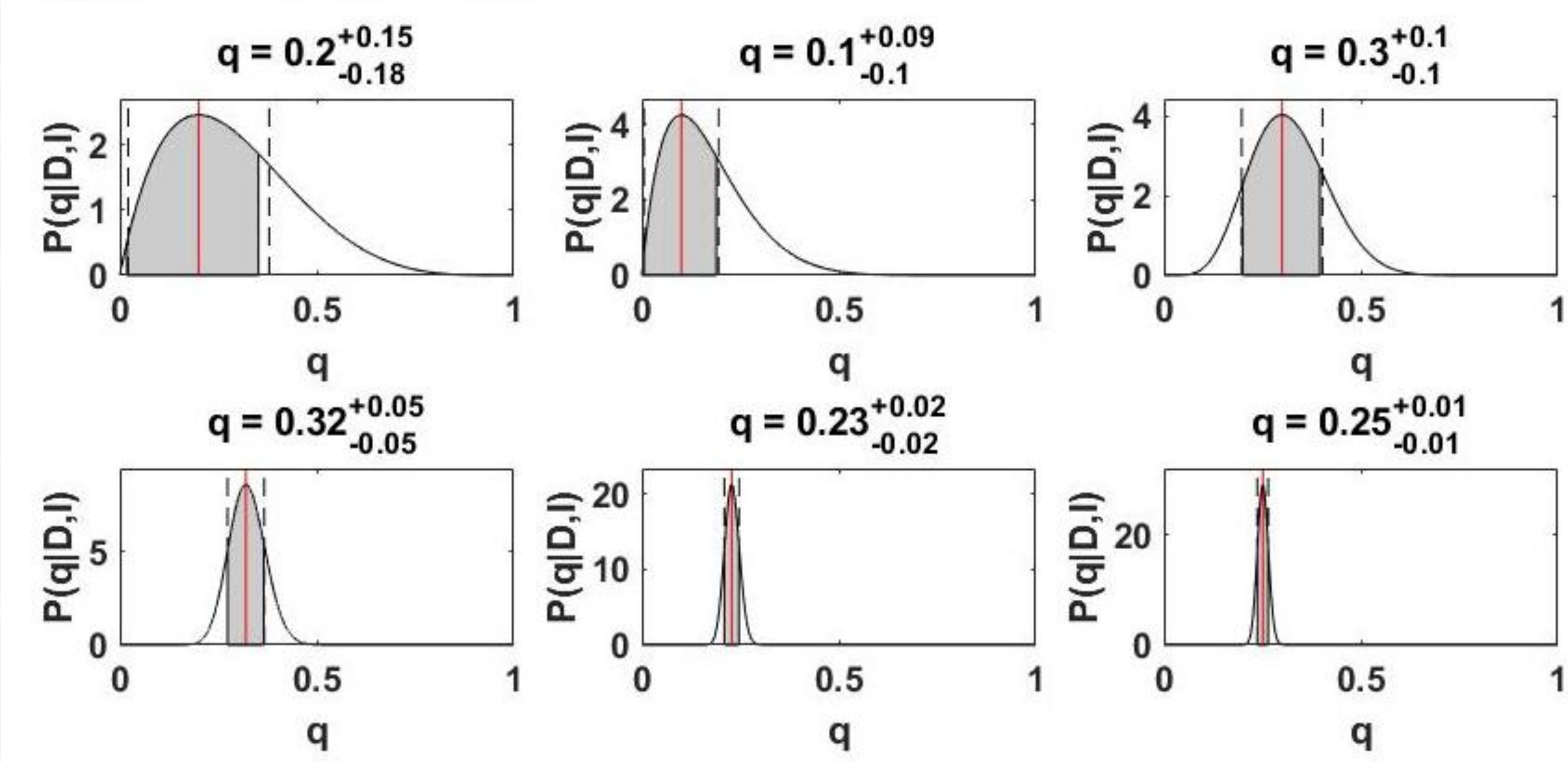


Bayesian Methods:

Parameter Estimation

check out `bayesian_bino.py`

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



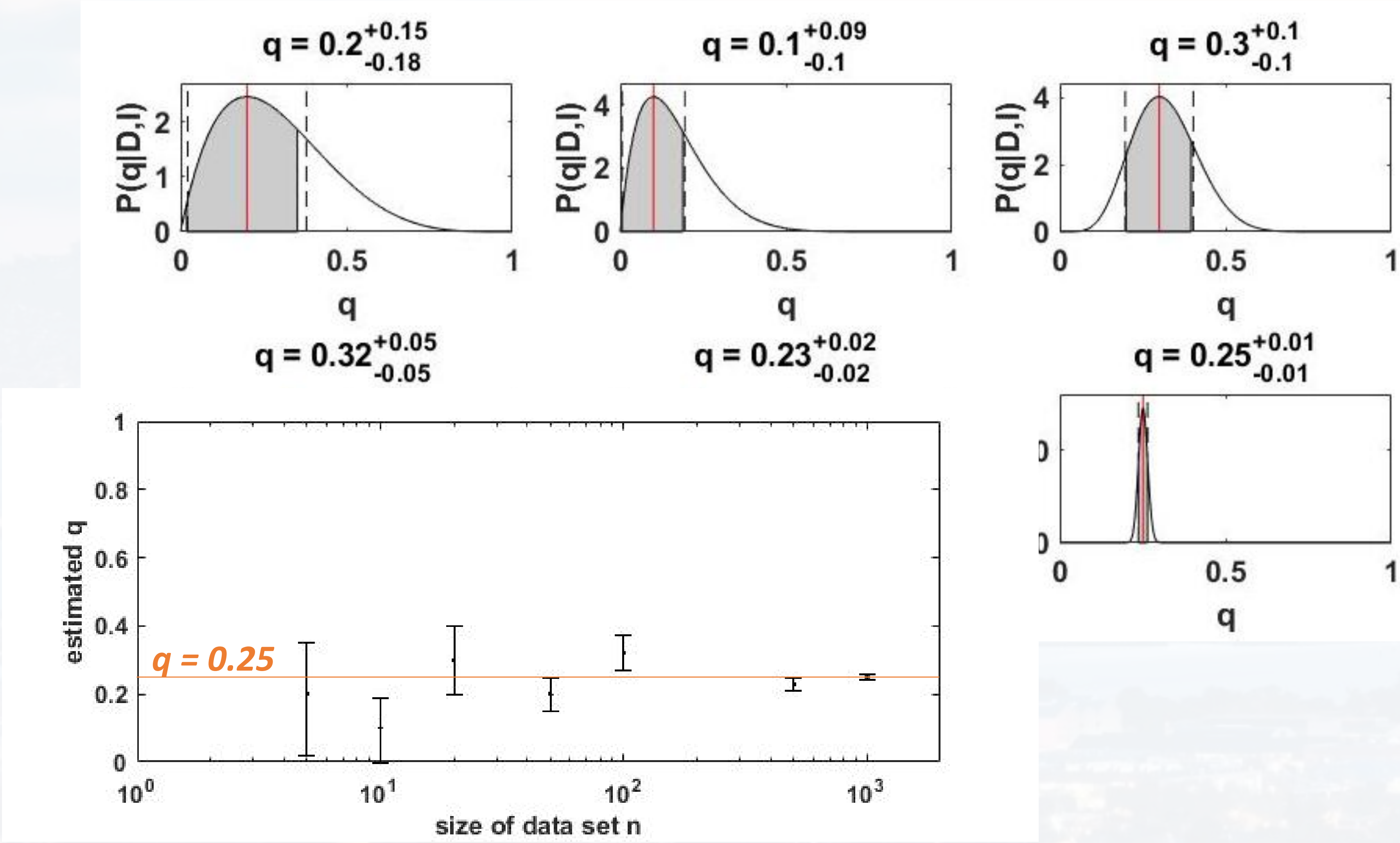
n	estimated q
5	$0.2^{+0.15}_{-0.18}$
10	$0.1^{+0.09}_{-0.1}$
20	$0.3^{+0.1}_{-0.1}$
50	$0.2^{+0.05}_{-0.06}$
100	$0.32^{+0.05}_{-0.05}$
500	$0.23^{+0.02}_{-0.02}$
1,000	$0.25^{+0.01}_{-0.01}$
infinity	0.25

Bayesian Methods:

Parameter Estimation

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

Bayesian Methods:

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

goal:
- **P(q|D)**
- the larger ***D***, the more certain ***q***
  → learning

likelihood function

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

prior

evidence **(const wrt q)**

<u>What is the average number of WhatsUp messages I get every day?</u>

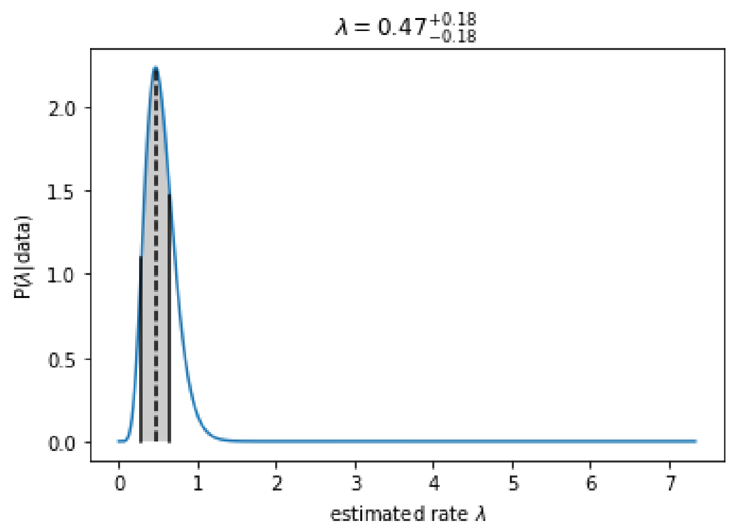
Mon:	5
Tue:	7
Wed:	1
Thu:	3
Fri:	9
Sat:	2
Sun:	5

event
- has no duration
- is rare

→ Poissonian

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

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



Bayesian Methods:

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

goal: - **P(q|D)**
- the larger ***D***, the more certain ***q***
→ learning

<u>What is the average number of WhatsUp messages I get every day?</u>

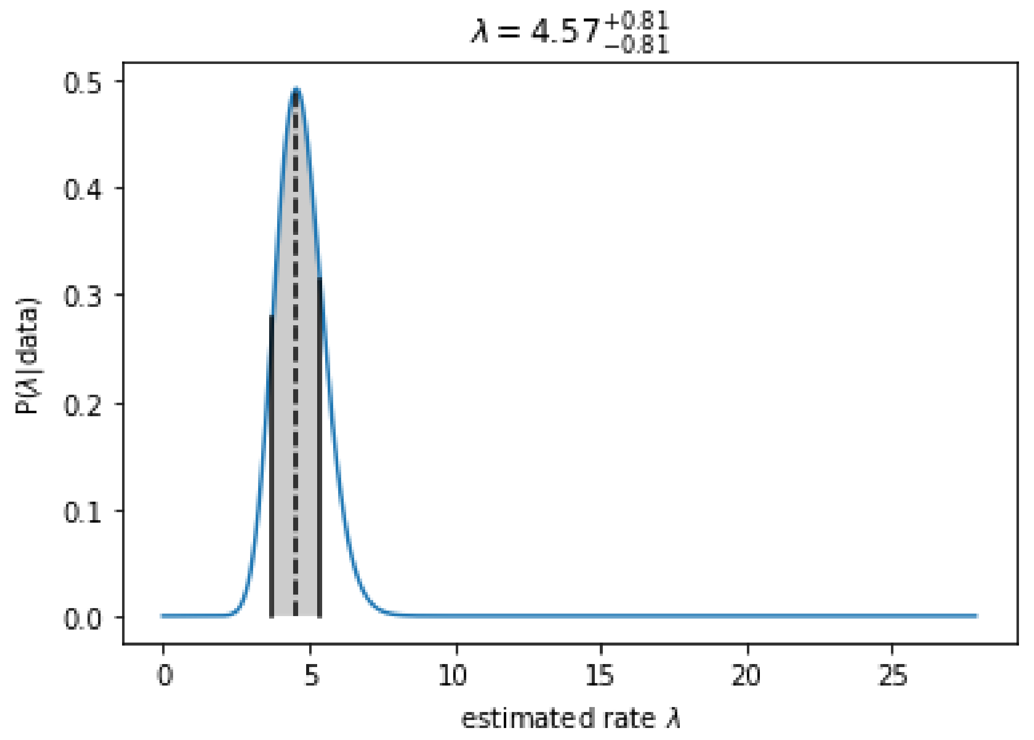
Mon:	5
Tue:	7
Wed:	1
Thu:	3
Fri:	9
Sat:	2
Sun:	5

event
- has no duration
- is rare

→ Poissonian

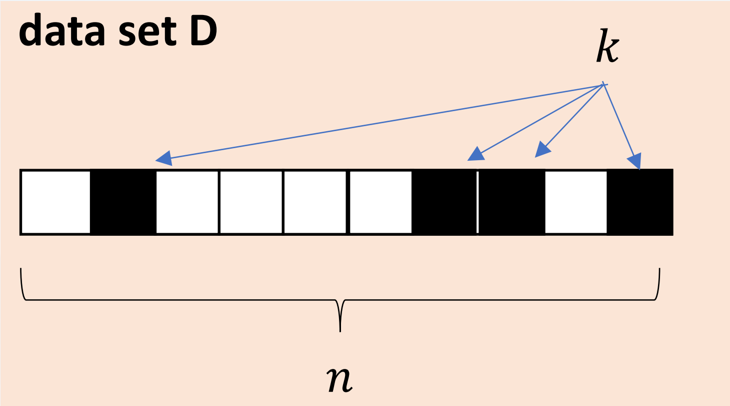
$$P(k|\lambda) = \frac{\lambda^k}{k!}\, e^{-\lambda}$$

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



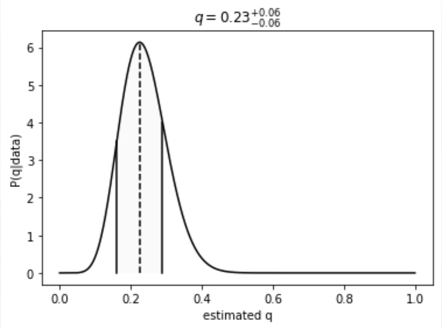
Bayesian Methods: Parameter Estimation

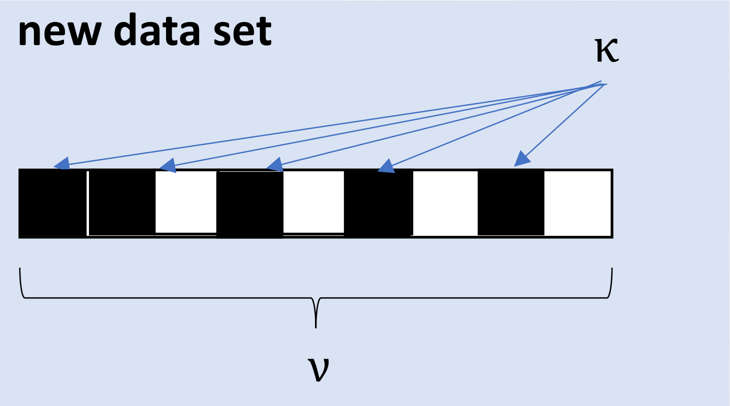
What if there is new data?



$\cancel{q = ?}$

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





if there **is** prior information ***I*** about *q*:

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

What if there is new data?

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

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

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

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

often: $\kappa = \alpha - 1$

$\beta = \nu - \kappa - 1$

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

Beta function

Bayesian Methods: Parameter Estimation

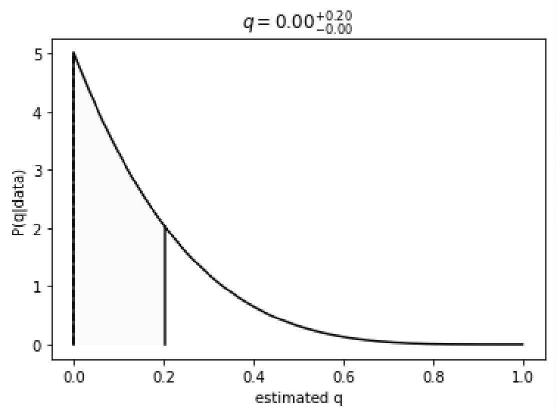
Berkeley UNIVERSITY OF CALIFORNIA

What if there is new data?

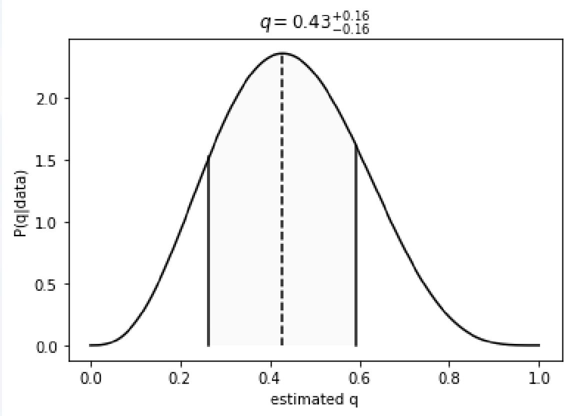
$$P(q|new\ data\ set, I) = \frac{q^{\kappa}(1-q)^{\nu-\kappa}\ q^{k}(1-q)^{n-k}}{\int_0^1 q^{\kappa}(1-q)^{\nu-\kappa}\ q^{k}(1-q)^{n-k}\ dq}$$

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

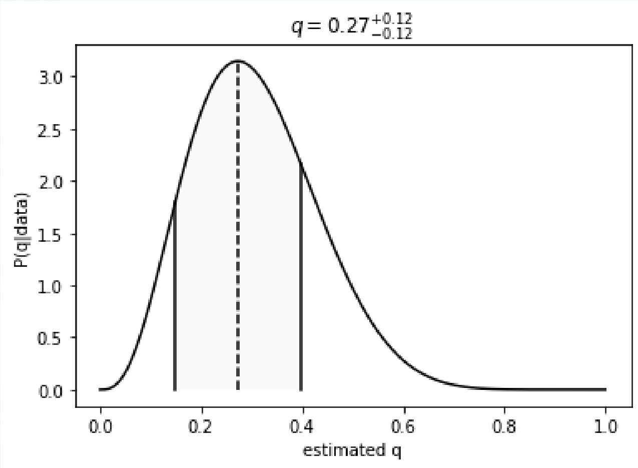
```
n2 = 7
k2 = np.random.binomial(n2, q = 0.2)
[_, _, _] = bayesian_bino(n2, k2)
```



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



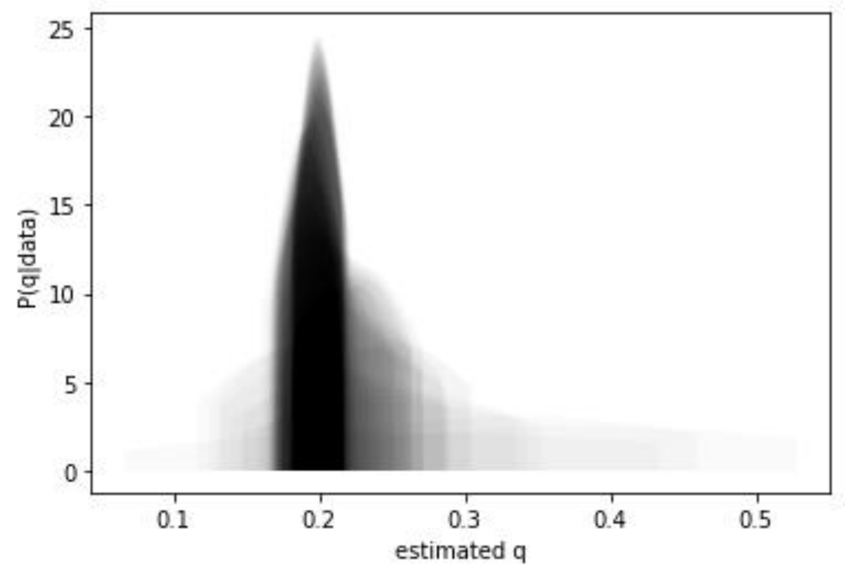

```
[_ , _, _] = bayesian_bino(n2, k2, Prior = Prior)
```



What if there is new data?

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

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. **V**ariational **A**uto **E**ncoders

2D images → 3D objects

credit: StableAI

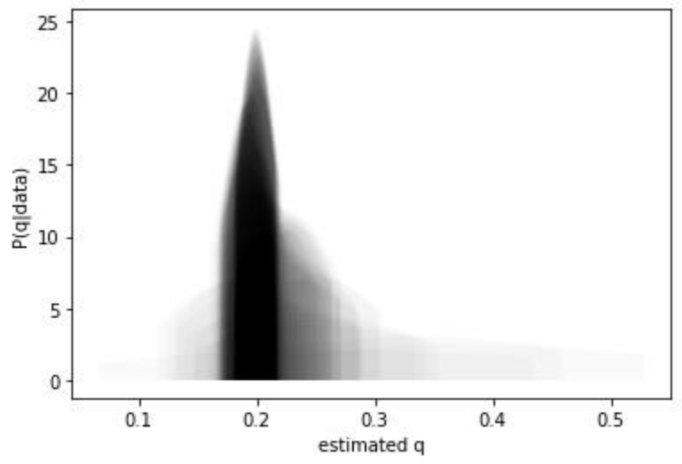
Bayesian Methods:

Parameter Estimation

What if there is new data?

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

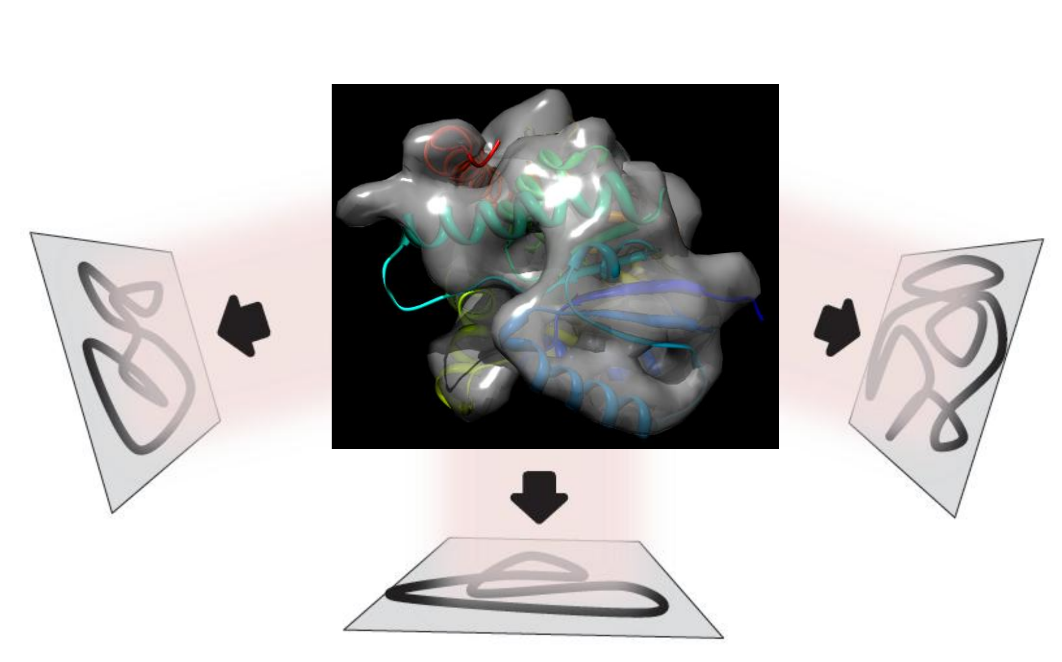
The **posterior** from the **previous** experiment is the **prior** of the **next** experiment



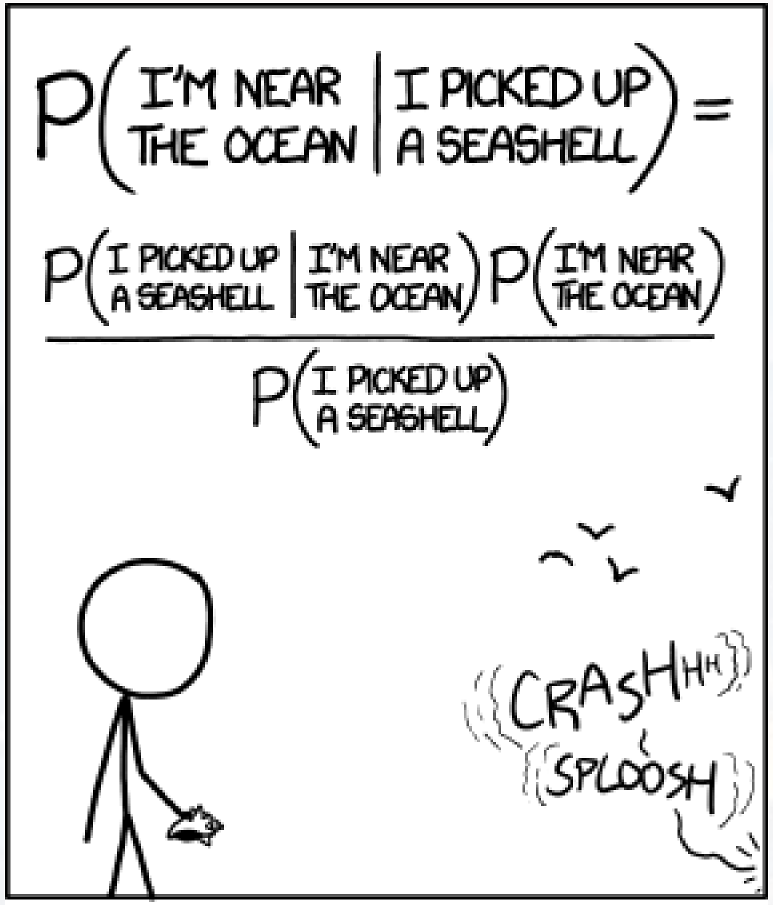
→ we become more certain about the model parameters
→ learning!

→ see e.g. **V**ariational **A**uto **E**ncoders   2D images → 3D objects

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



(image courtesy: Thomas Becker, GC LMU Munich)



STATISTICALLY SPEAKING, IF YOU PICK UP A SEASHELL AND *DON'T* HOLD IT TO YOUR EAR, YOU CAN PROBABLY HEAR THE OCEAN.

<u>Outline</u>

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

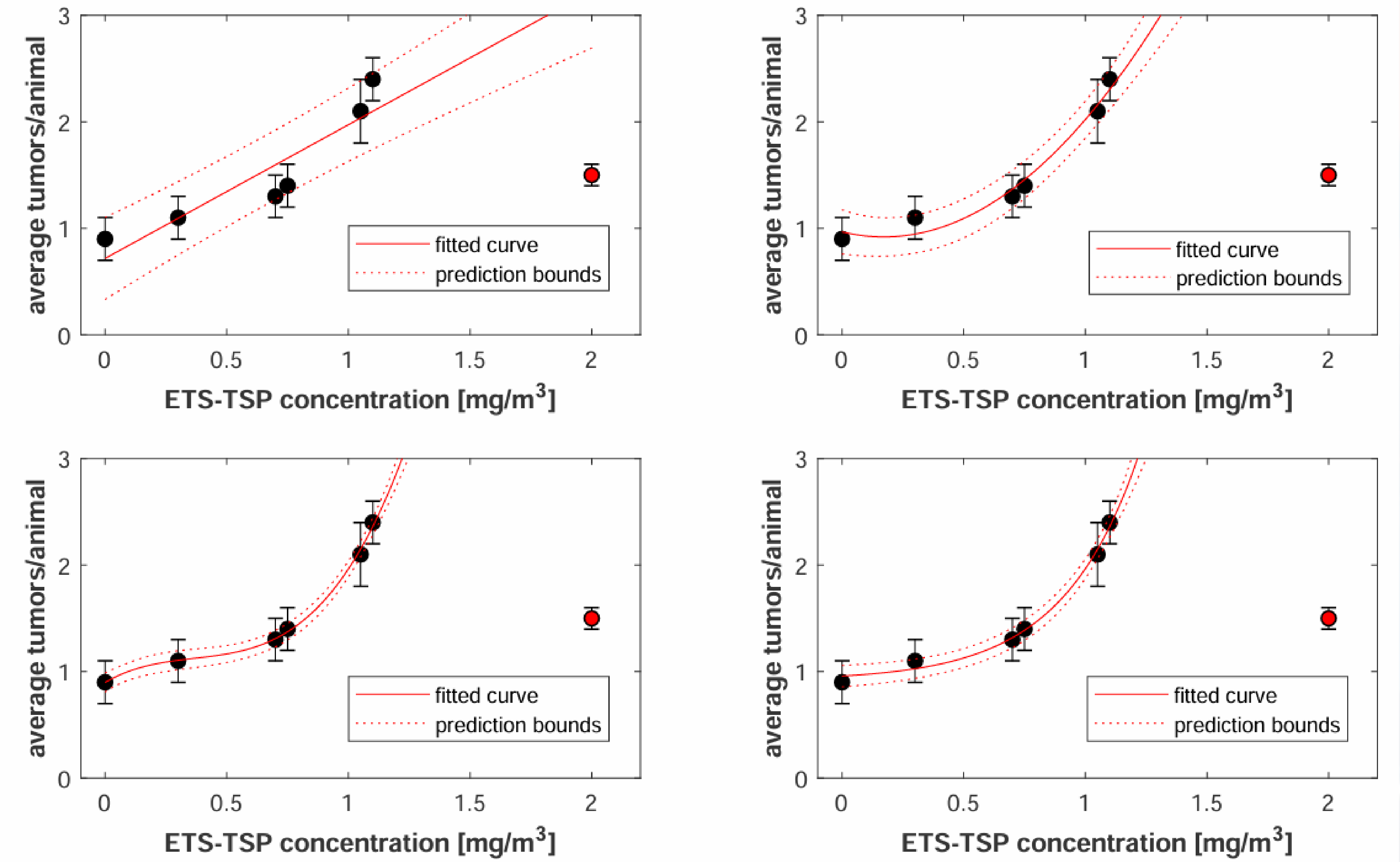
<u>FYI</u>

- Bayesian Networks (Graphs)
- Variational Bayes

Bayesian Methods:

Model Selection

often, we have many competing models → assigning probabilities if a model is correct



DOI: 10.1093/carcin/23.3.511
Source: PubMed

Bayesian Methods:

often, we have many competing models → assigning probabilities if a model is correct

D : data
$\mathrm{M_A}$: model A
$\mathrm{M_B}$: model B

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

Bayes' theorem

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

marginalization:

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

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

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

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

Bayesian Methods:

Model Selection

goal:

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

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

marginalization:

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

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

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

likelihood function
→ the actual model

prior of $\boldsymbol{\alpha_{ij}}$ BEVORE(!) measurement
Maximum Entropy without prior knowledge:

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

Bayesian Methods: Model Selection

goal:

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

D : data
$\mathbf{M_A}$: model A
$\boldsymbol{M}_{\mathbf{B}}$: model B
$\{\alpha\}_i$: all parameter of model $\boldsymbol{M_i}$

marginalization:

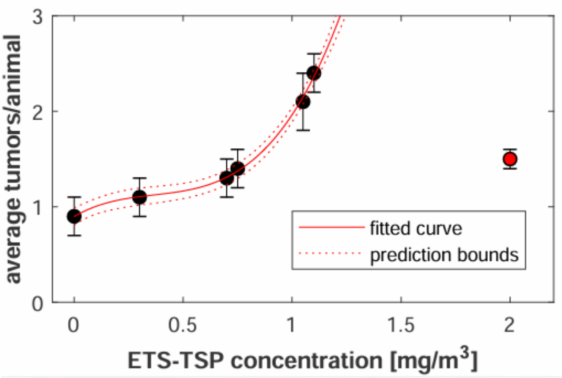
$$\boldsymbol{P(D|M_i) = \int P(D|\{\alpha\}_i\ M_i)\ P(\{\alpha\}_i|\ M_i)\ d\Omega_{\{\alpha\}_i}}$$

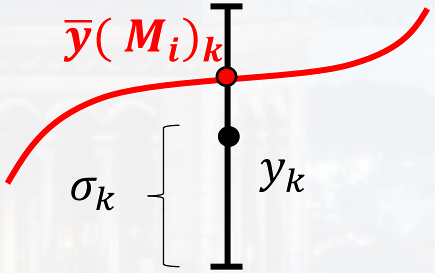
$$= \int P(D|\{\alpha\}_i, M_i) \prod_j P(\alpha_{ij}|\ M_i)\ d\alpha_{ij}$$

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

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

likelihood function
→ the actual model





$\boldsymbol{y_k}$: measured value
$\boldsymbol{\sigma_k}$: error
$\bar{\mathbf{y}}(\ \mathbf{M_i})_{\mathbf{k}}$: model value (after fit)

Bayesian Methods:

goal:

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

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

marginalization:

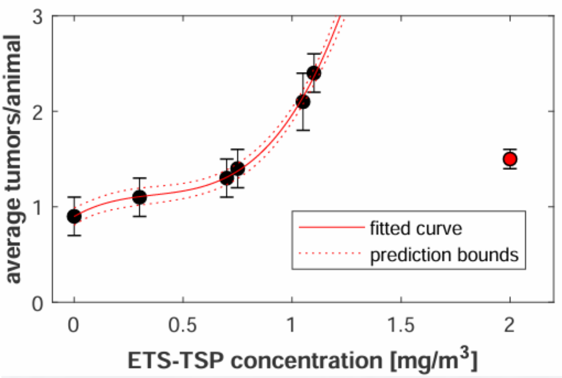
$$\boldsymbol{P(D|M_i) = \int P(D|\{\alpha\}_i\, M_i)\, P(\{\alpha\}_i|\, M_i)\ d\Omega_{\{\alpha\}_i}}$$

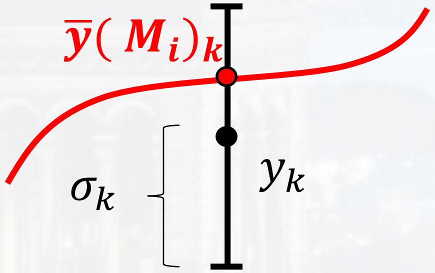
$$= \int P(D|\{\alpha\}_i, M_i) \prod_j P(\alpha_{ij}|\, M_i)\ d\alpha_{ij}$$

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

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

likelihood function
→ the actual model





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

for $\sigma_k \ll |y_k|$

D	: data
$\mathbf{M_A}$	: model A
$\boldsymbol{M}_{\mathbf{B}}$	: model B
$\{\alpha\}_i$	: all parameter of model $\boldsymbol{M_i}$
$\boldsymbol{y_k}$	: measured value
$\boldsymbol{\sigma_k}$	: error
$\bar{\mathbf{y}}(\mathbf{M_i})_{\mathbf{k}}$	: model value (after fit)

marginalization:

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

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

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

likelihood function

→ the actual model

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

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

Bayesian Methods:

Model Selection

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)

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

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

fit quality: integral over χ^2

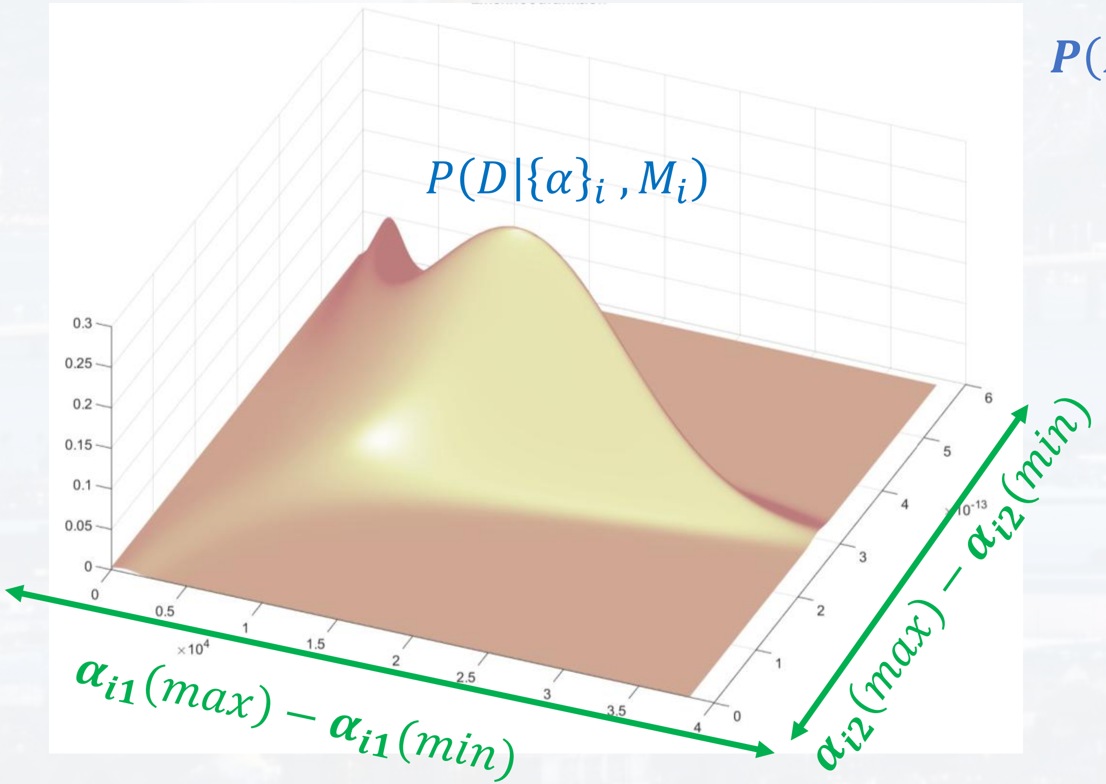
prior probability of each model: maximum entropy → 1:1

Occam's Razor: simple models are preferred

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

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

D	**: data**
$\mathbf{M_A}$	**: model A**
$\boldsymbol{M_B}$	**: model B**
$\{\alpha\}_i$	**: all parameter of model $\boldsymbol{M_i}$**
$\boldsymbol{y_k}$	**: measured value**
$\boldsymbol{\sigma_k}$	**: error**
$\bar{\mathbf{y}}(\mathbf{M_i})_\mathbf{k}$	**: model value (after fit)**



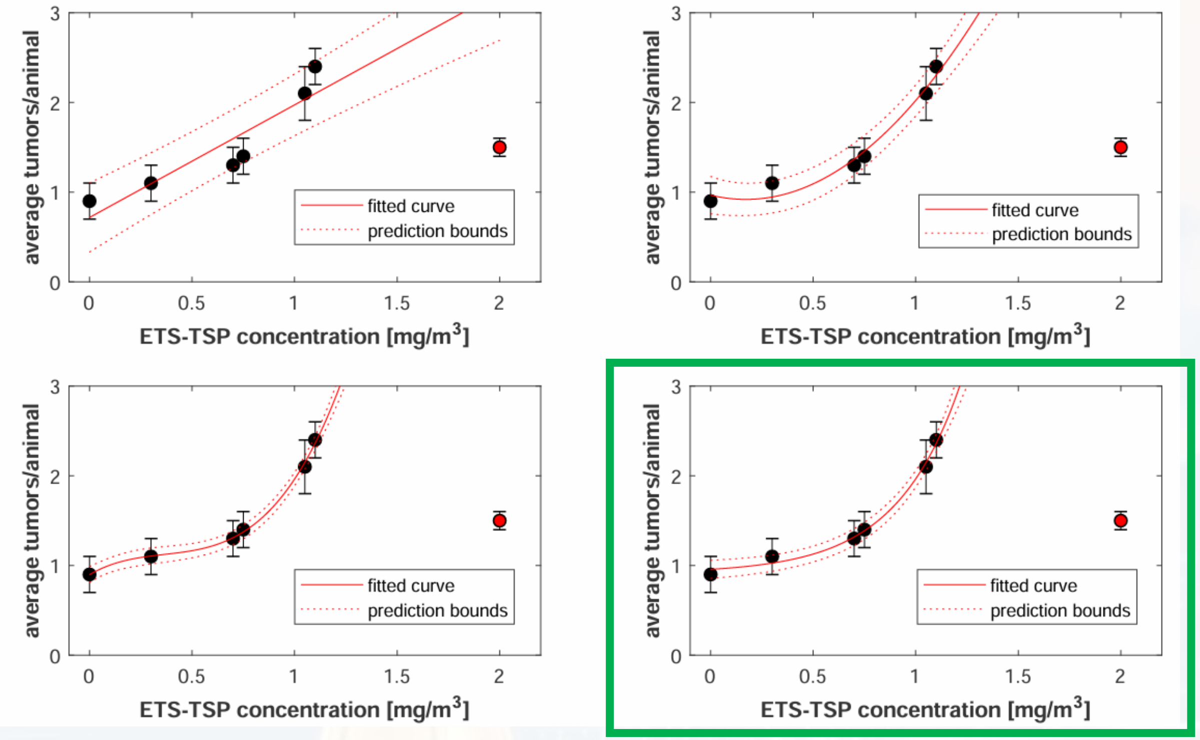
$$\boldsymbol{P(D|M_i) = \int P(D|\{\alpha\}_i\, M_i)\, P(\{\alpha\}_i|\, M_i)\;\, d\Omega_{\{\alpha\}_i}}$$

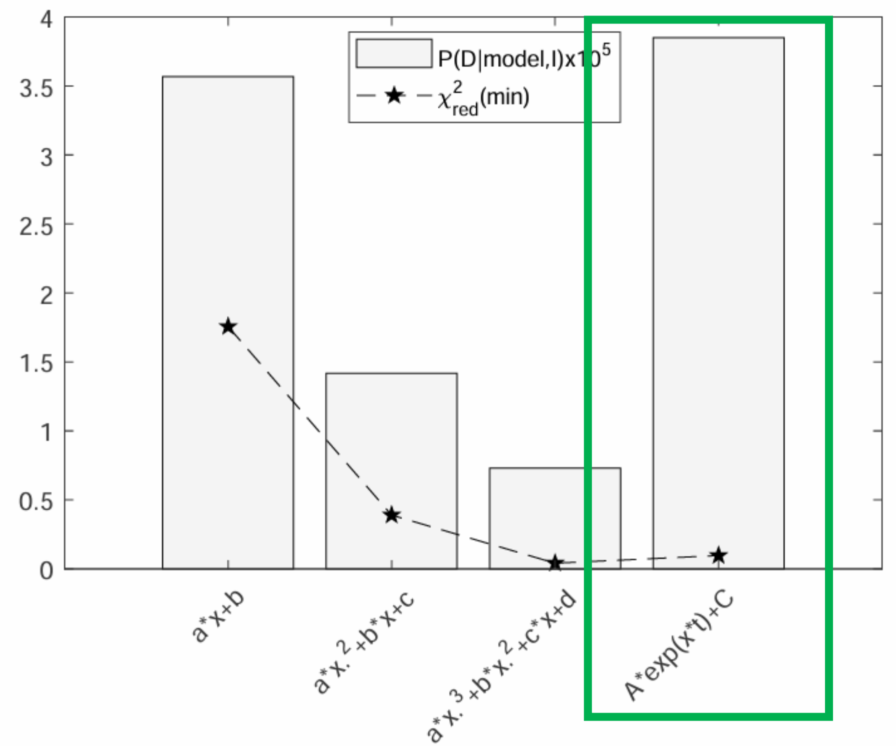
$$= \int P(D|\{\alpha\}_i, M_i) \prod_j P(\alpha_{ij}|\, M_i)\;\, d\alpha_{ij}$$

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

Bayesian Methods:

Model Selection





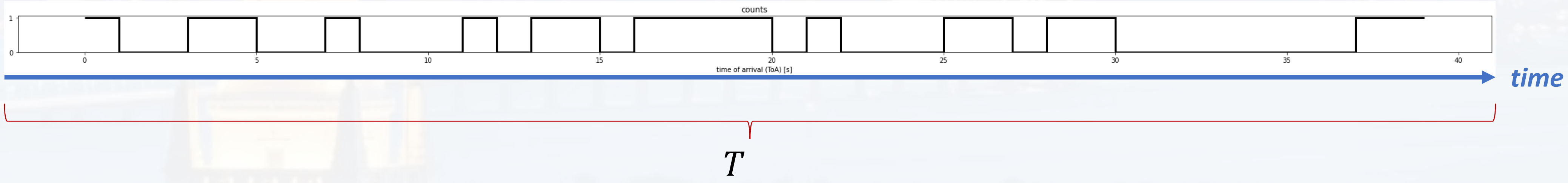
Bayesian Methods:

The key part is the likelihood function!

$$\rho = \frac{P(D|M_A)\, P(M_A)}{P(D|M_B)\, P(M_B)} \qquad \boldsymbol{P(D|M_i) = \int P(D|\{\alpha\}_i\, M_i)\, P(\{\alpha\}_i|\, M_i)\ d\Omega_{\{\alpha\}_i}}$$

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

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



M_A: constant model (no signal, just noise) $\rightarrow r(t) = const$

M_B: signal of unknown phase, amplitude & frequency $\rightarrow r(t) = f(t)$

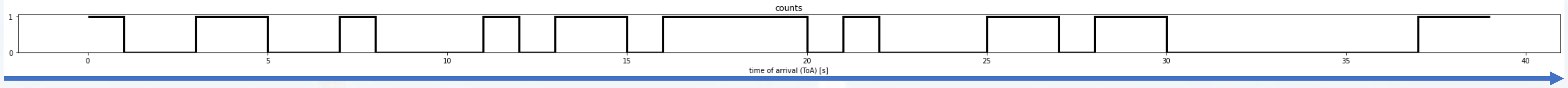
Bayesian Methods:

Model Selection

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

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

Poisson distribution



time

T

actual data (ToA)

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

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

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

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

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

Bayesian Methods:

Model Selection

$\boldsymbol{r(t)}$:	rate
$\boldsymbol{\Delta t}$:	time resolution
$\boldsymbol{n}$:	number of events
$\boldsymbol{T}$:	obs. time span
$\boldsymbol{D}$:	data set

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

Poisson distribution

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

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

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

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

Bayesian Methods:

Model Selection

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

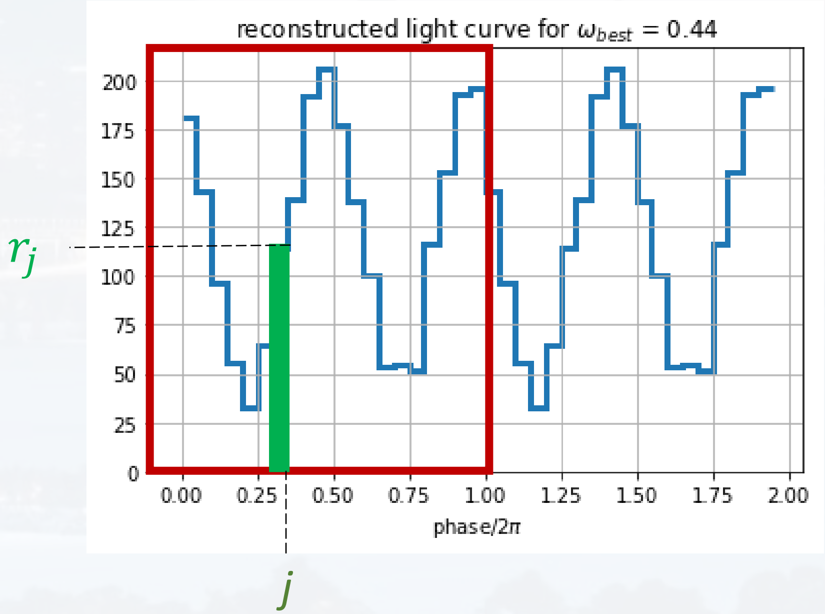
$$\boxed{P(D|\, r(t), t) = (\Delta t)^N \cdot \prod_{i=1}^{N} r(t_i) \cdot \exp\left(-\int_0^T r(t)\, dt\right)}$$

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

Bayesian Methods:

Model Selection

$r(t)$:	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

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

constant model: r_j = const $\forall\, j$

→ $r_j = A$

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

actual signal

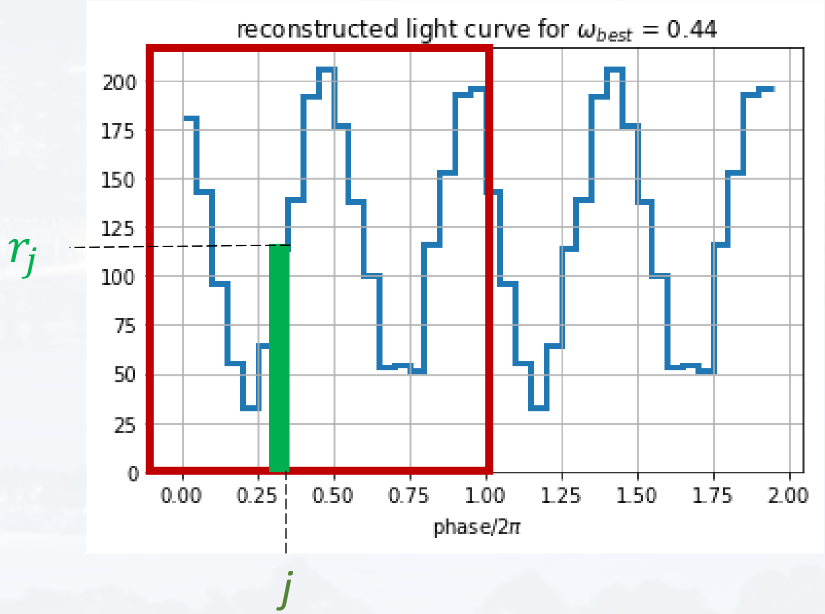
- amplitude
- phase
- frequency
- offset

detection →

- t_i

analysis →

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



Bayesian Methods:

Model Selection

$r(t)$:	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

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:

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

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

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

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

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

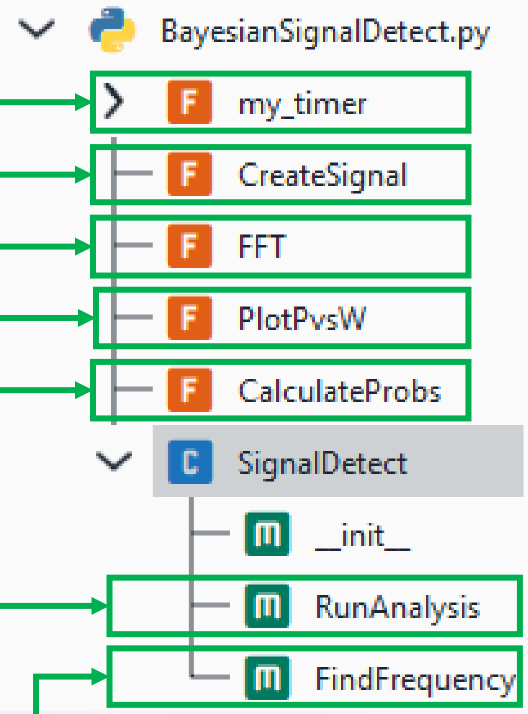
$$P(f|M_i) = (m-1)!\ \delta\left(1 - \sum_{j=1}^{m} f_j\right)$$

Bayesian Methods: **Model Selection**

you find the python package `BayesianSignalDetect.py` *here*

```python
from BayesianSignalDetect import *
```

- a *decorator* that measures runtime of a function → my_timer
- *function* creates a test dataset for illustration → CreateSignal
- *FFT* for comparison → FFT
- plot routine for *periodogram* → PlotPvsW
- calculates $P(D|\omega, \varphi, A, m)$ → CalculateProbs
- calls *CalculateProbs* and calculates $P(D|M_i)$ → RunAnalysis
- calls main part of the code – runs *RunAnalysis* on multiple cpus in parallel → FindFrequency



Bayesian Methods:

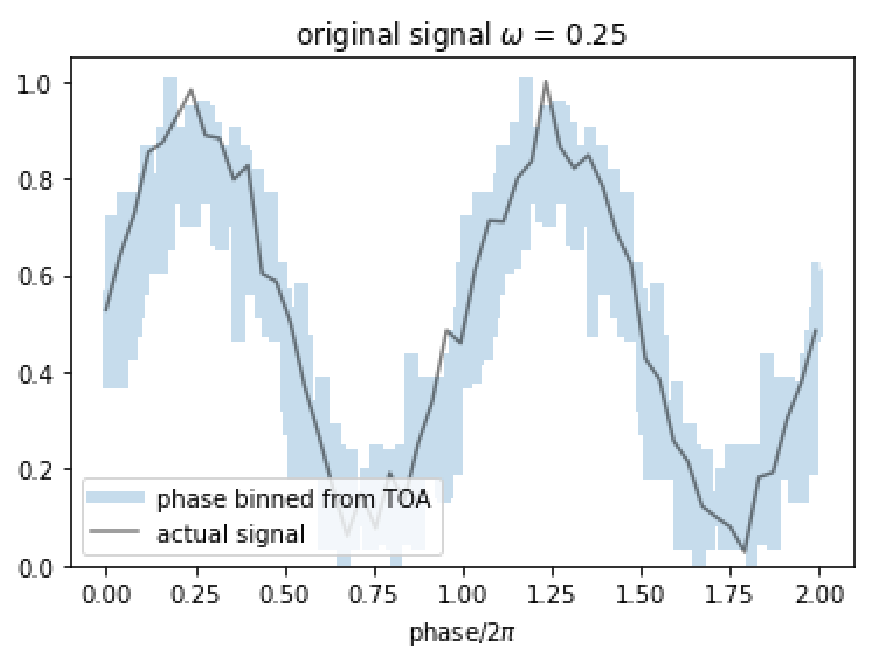
Model Selection

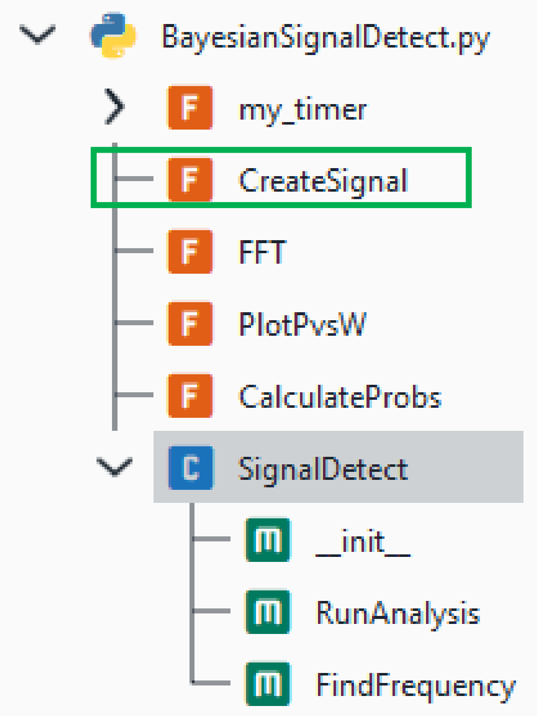
you find the python package `BayesianSignalDetect.py` *here*

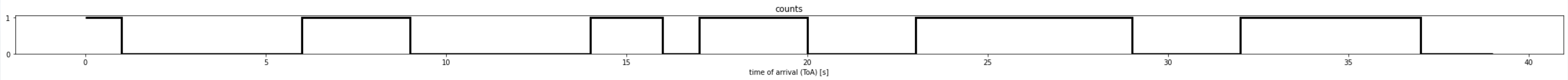
```
from BayesianSignalDetect import *
```

N + Q

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



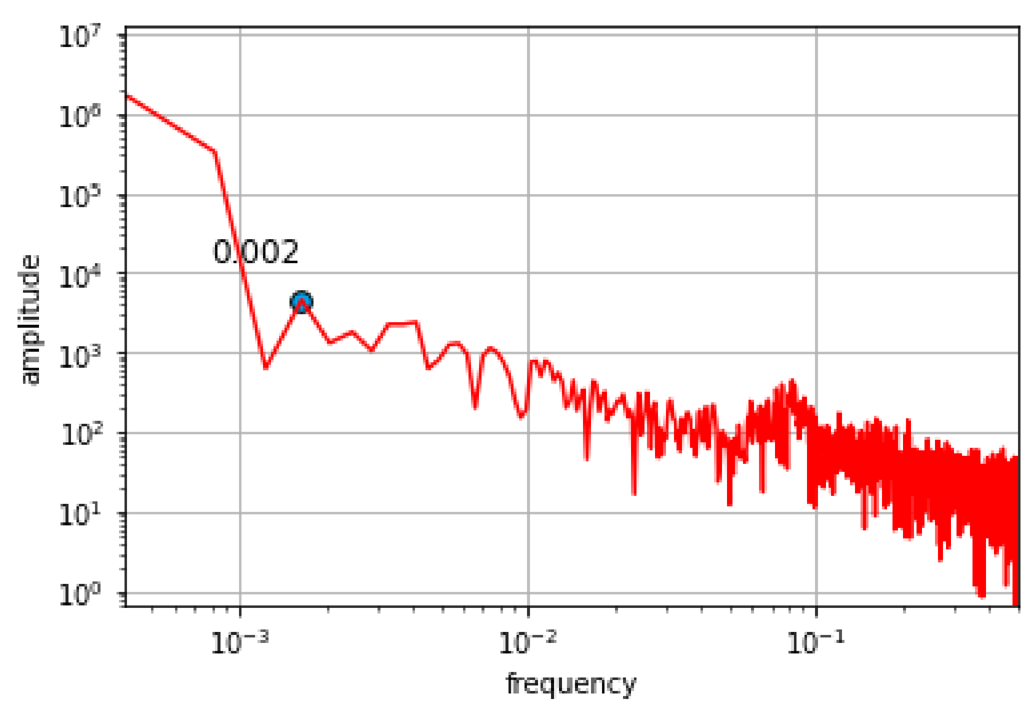


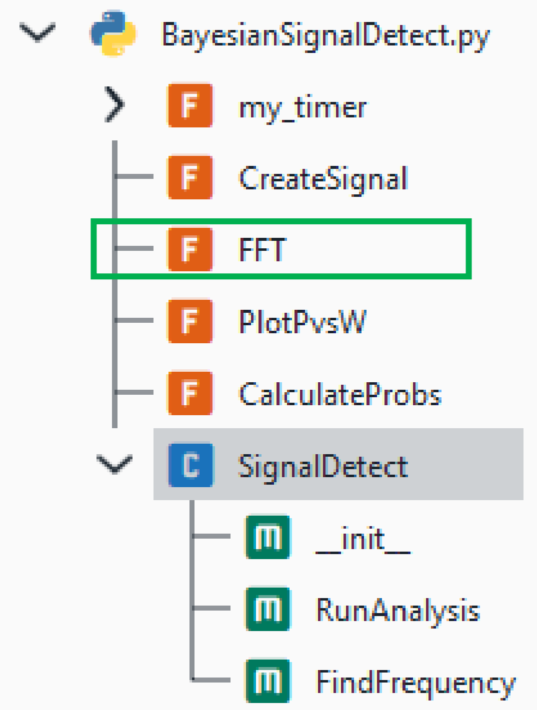


Bayesian Methods:

you find the python package `BayesianSignalDetect.py` *here*

```
from BayesianSignalDetect import *

T = CreateSignal(5000, 0.25, 0.1)

FFT(T)
```



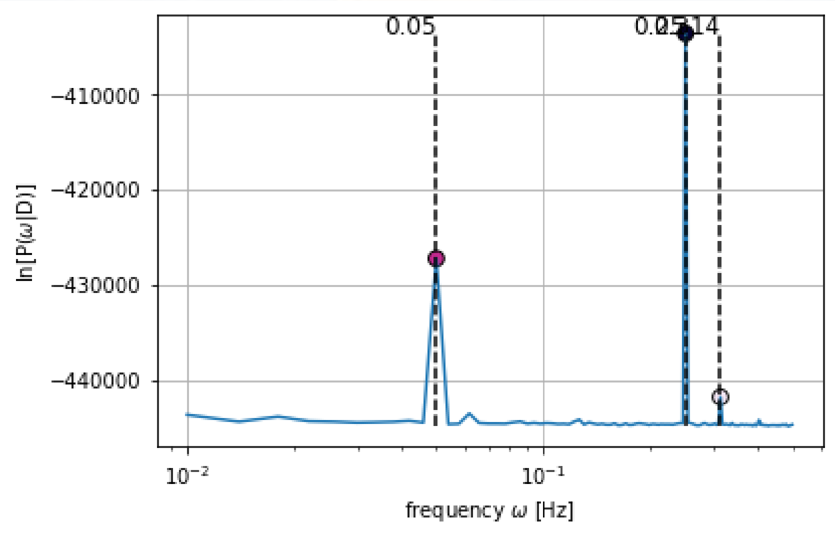


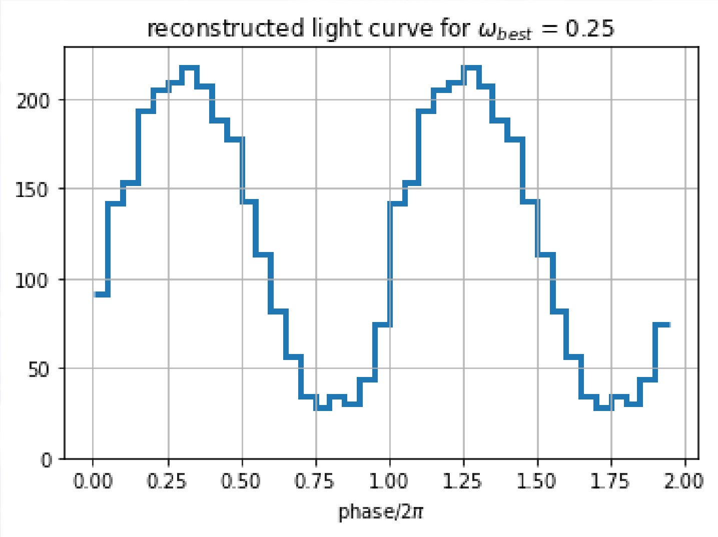
Bayesian Methods:

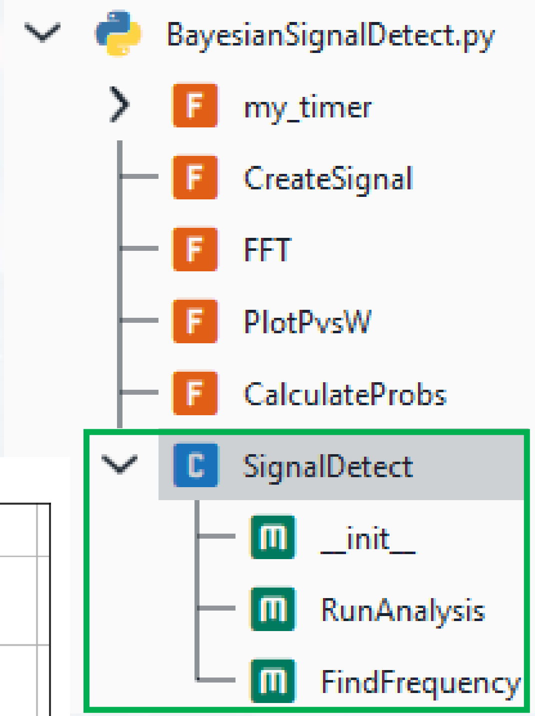
Model Selection

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()
```



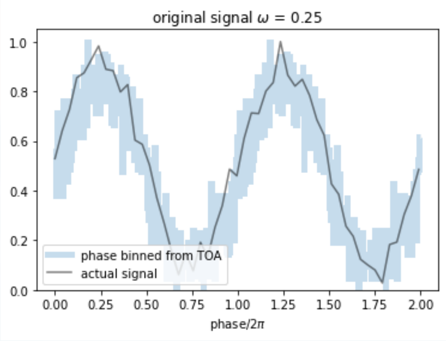


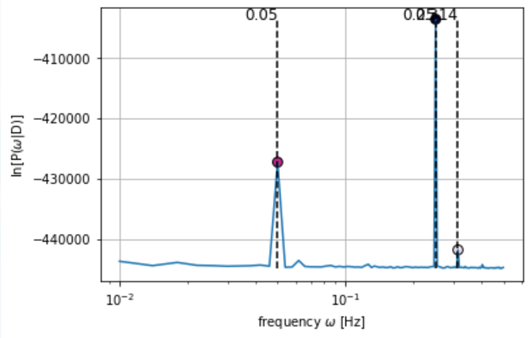


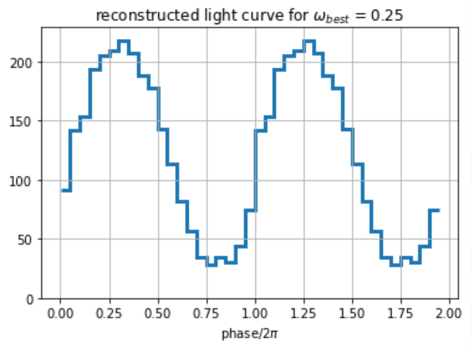
Bayesian Methods:

Model Selection

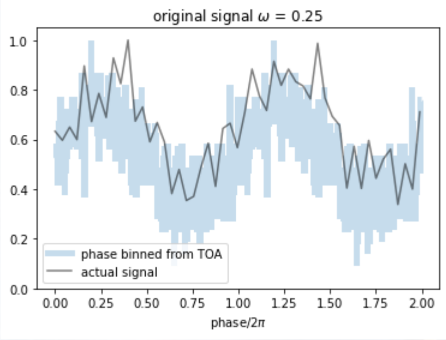
`T = CreateSignal(5000, 0.25, 0.1)`

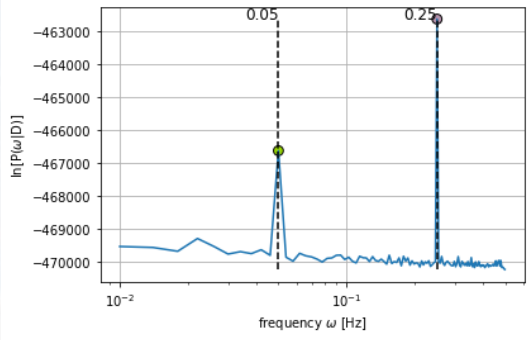


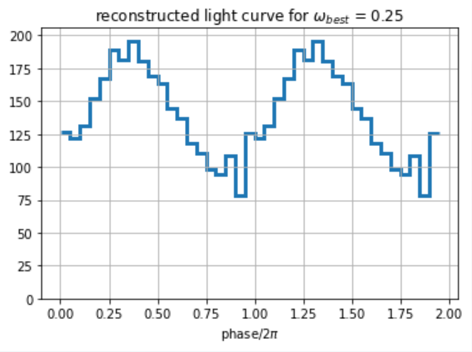




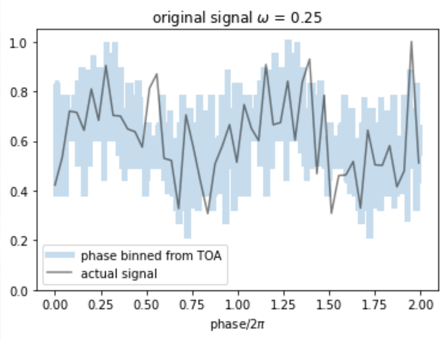
`T = CreateSignal(5000, 0.25, 0.5)`

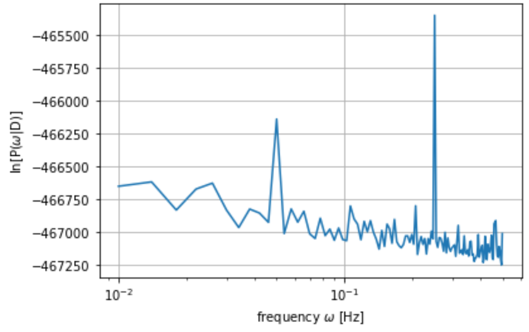


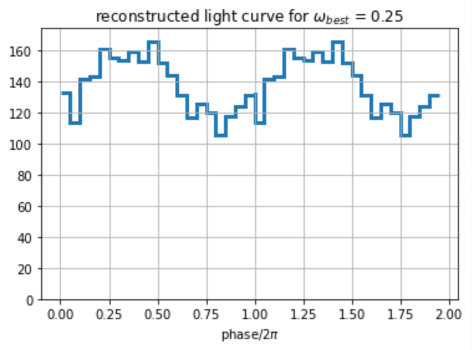




`T = CreateSignal(5000, 0.25, 1)`







Thank you for your attention