Learning goals of the week:

- learn how to generate simple toy MC samples
- sliding mean

Week 7

Monte Carlo basics

Monte Carlo method

Monte Carlo methods are numerical methods based on the random sampling of input distributions to solve problems such as systems simulations or numerical integration.

Typically they are used when a direct analytical solution is not achievable

The basic idea can be summarized as:

- find the domain space of all relevant variables of the system you want to simulate / integrate: this is typically a multi-dimensional space and the variables are coupled
- generate a set of data, sampling the distribution of variables in the domain space by throwing random numbers
- Compute the output of the MC based on the sampled distributions

Examples of applications:

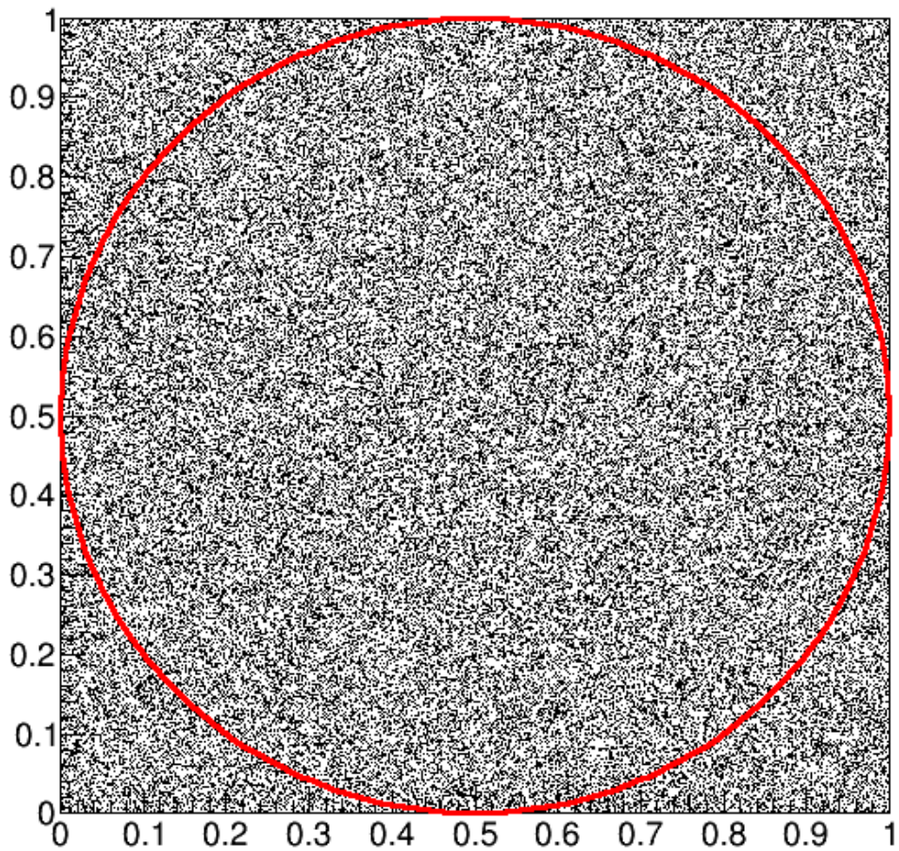
- particle scattering
- Numerical Integrations
- Risk analysis
- Finance, stock market, transaction probabilities…
- Investment banking
- …

History: compute π

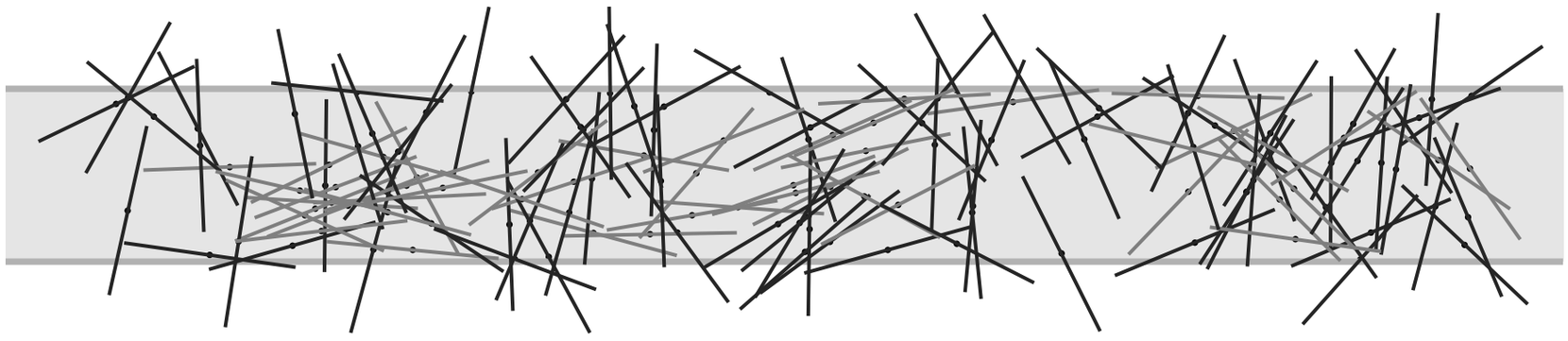
Wait for a rainy day and count the number of raindrops within a circle and the square inscribing it

$$\frac{\pi r^2}{4r^2} = \frac{in}{all}$$

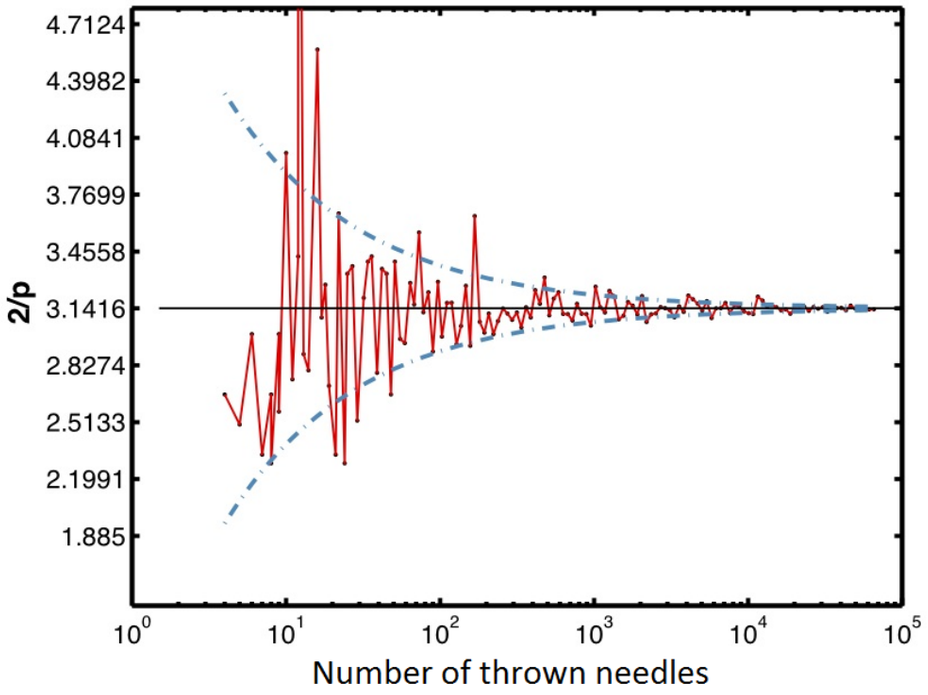
$$\pi = 4 \cdot \frac{in}{all}$$



History: compute π



(1777, Duke of Buffon) Compute π counting needles thrown on a strip of paper of width equal to the length of the needles (l).



k/n = 2/π : the ratio between the number of needles (k) crossing the border of the strip (the slightly darker ones in Fig. 4.0.1) and the total number of thrown needles (n) is 2/π

[proof: see e.g. wiki]

Random numbers generator

True random generators are based on physical processes. In the previous two examples: raindrop distribution, needles distribution
(see e.g. http://www.random.org offers true random numbers generated from atmospheric noise)

Examples of physical processes:

- noise level across a resistor
- time between the arrival of cosmic rays
- number of radioactive decays in a fixed time interval.

Early physical generator: use the stopping azimuthal position of a cylinder which had been put in rotation by a motor (activated by an operator) and turned off by a cosmic ray recorded by a detector

—> Too slow, not affordable to generate large sets of random numbers

No need for true randomness: pseudorandom (i.e. generated by a computer following an algorithm) is good enough.

(Pseudo-)Random numbers generator

Linear congruential generator —> generate uniformly distributed numbers in (0, 1]

$$n_{i+1} = (a \cdot n_i + c) \mod m$$

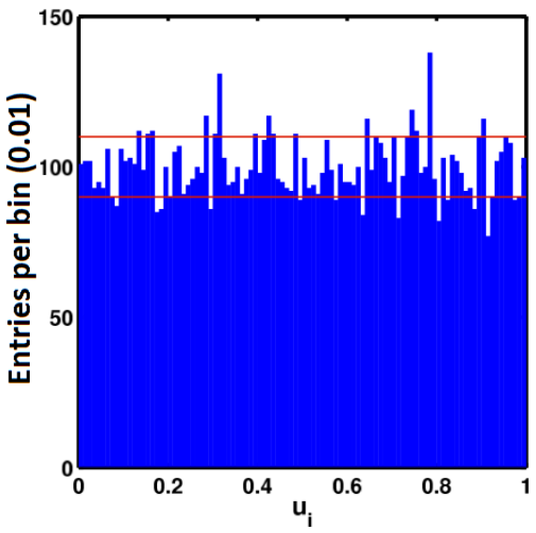
```
def lcg(modulus, a, c, seed):
  while True:
    seed = (a * seed + c) % modulus
    yield seed
```

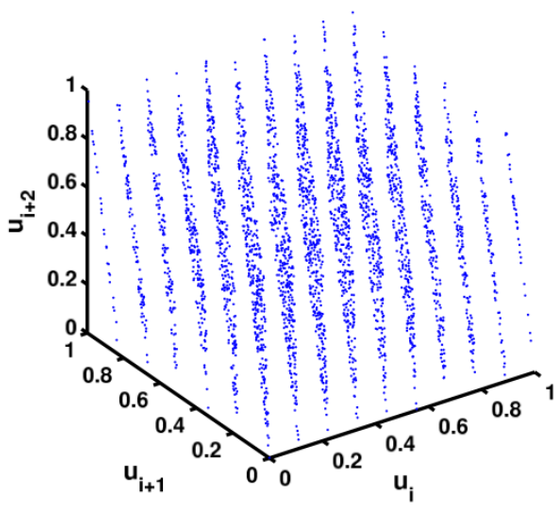
where a and c are integer numbers. The initial value n_1 is called the seed: all (pseudo-)random number generators can be started from an arbitrary initial state called “seed “. It will always produce the same sequence when initialized with that state

Properties:

periodicity: the maximum, over all starting states, of the length of the repetition-free prefix of the sequence.

correlations: If n successive random numbers are plotted as coordinates in an n-dimensional space, then these points lie on hyperplanes. A good generator has many hyperplanes uniformly distributed.





(Pseudo-)Random numbers generator

Example: Mersenne Twister —> python
period of $2^{19937}-1$ iterations ($\approx 4.3\times10^{6001}$)
equi-distributed in (up to) 623 dimensions (for 32-bit values)

```
>>> import numpy as np
```

Sequence of random numbers:

```
>>> np.random.seed(12345)

>>> 5 + np.random.sample(10) * 5
array([ 7.14292096,  6.84837089,  6.38203972,  8.80365208,  9.06627847,
        5.69871186,  6.37734538,  9.60618347,  9.34319843,  8.63550653])
```

Hit / Miss method

aka acceptance/rejection methods

Generate random numbers according to a probability distribution.
General method, but not very efficient

Assume $a < x < b$: determine an upper limit c with $c \geq \max(f(x))$

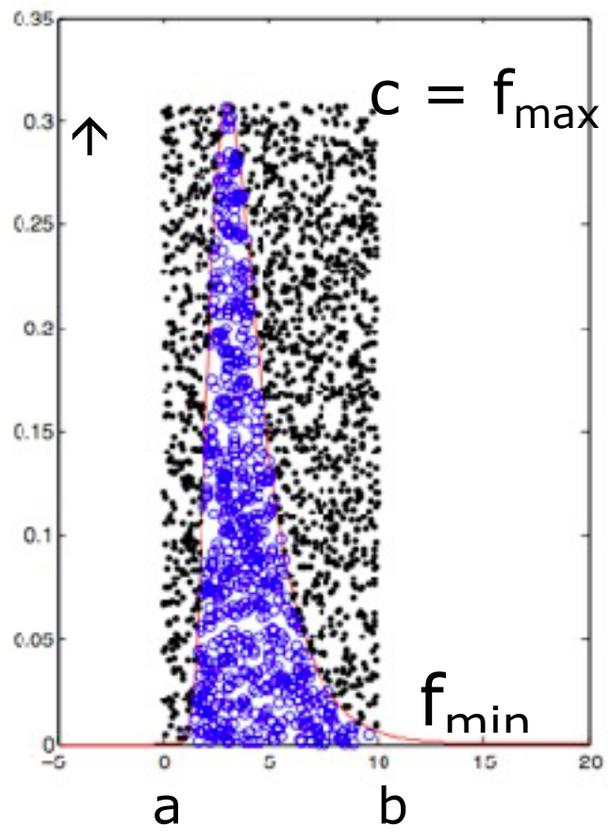
The hit/miss algorithm is:

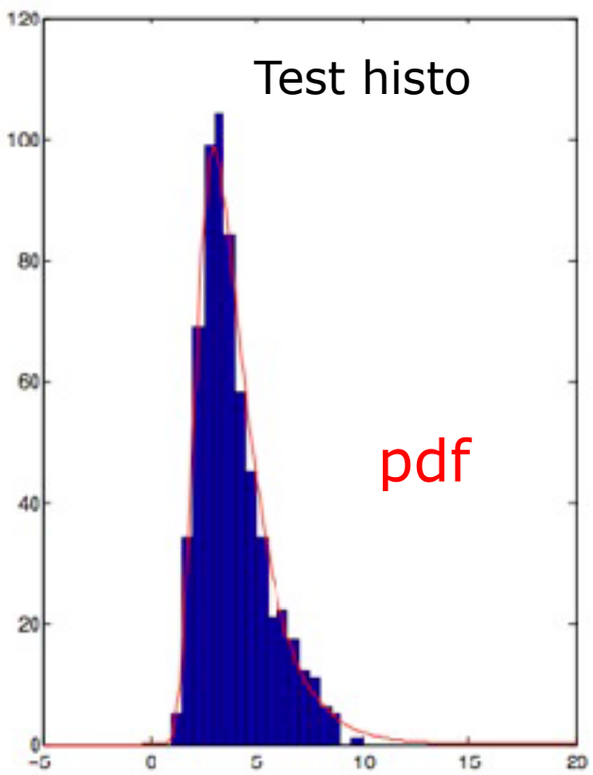
1. Choose x_i uniformly from the interval [a,b]: $x_i = a + u_i \cdot (b - a)$, with $u_i \in U(0, 1)$.

2. Choose another random number
   $u_j \in U(0,1)$.

3. If $f(x_i) > u_j \cdot c$, then "miss": go to step 1, otherwise "hit" x_i as random number.

The efficiency of this method is given by the ratio between the integral of f(x) over [a,b] and the total area $c \cdot (b - a)$ of the space of all generated pairs (u_i, u_j).

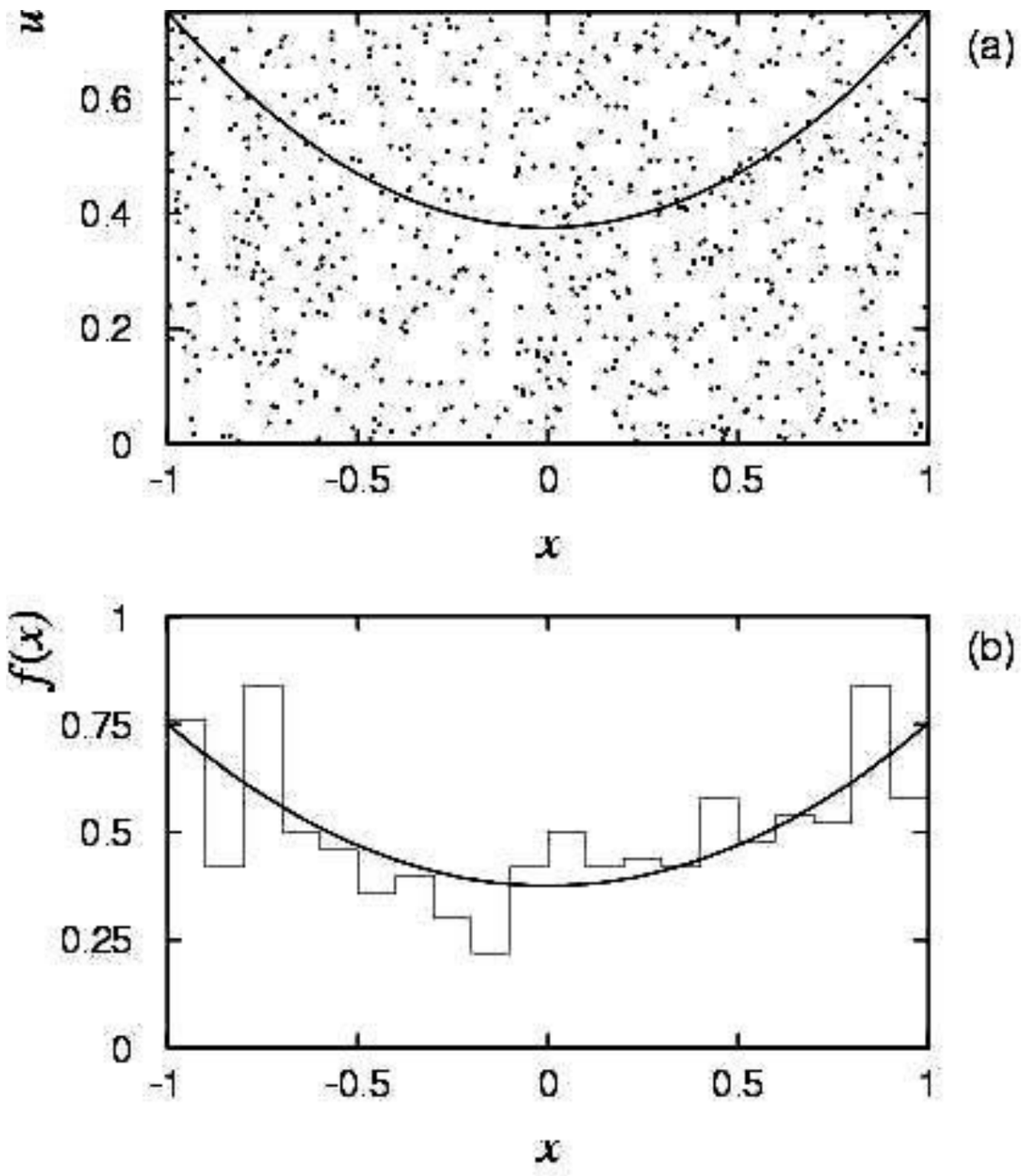




Hit / Miss method

$$f(x) = \frac{3}{8}(1 + x^2)$$

$$(-1 \leq x \leq 1)$$



If dot below curve, put

x value in the histogram.

cdf: Cumulative distribution function

The cdf is defined as $$F(x) = \int_{-\infty}^{x} f(t)dt$$

and it represents the probability to observe a value of the random variable less or equal to x

Properties:

- F(x) is a non decreasing, differentiable function of x
- $F(-\infty) = 0$ ($F(x_{min})$)
- $F(\infty) = 1$ ($F(x_{max})$)
- F(x) is dimensionless (as all probabilities)

From the cdf we can obtain back the pdf by differentiation $$f(x) = \frac{dF(x)}{dx}$$

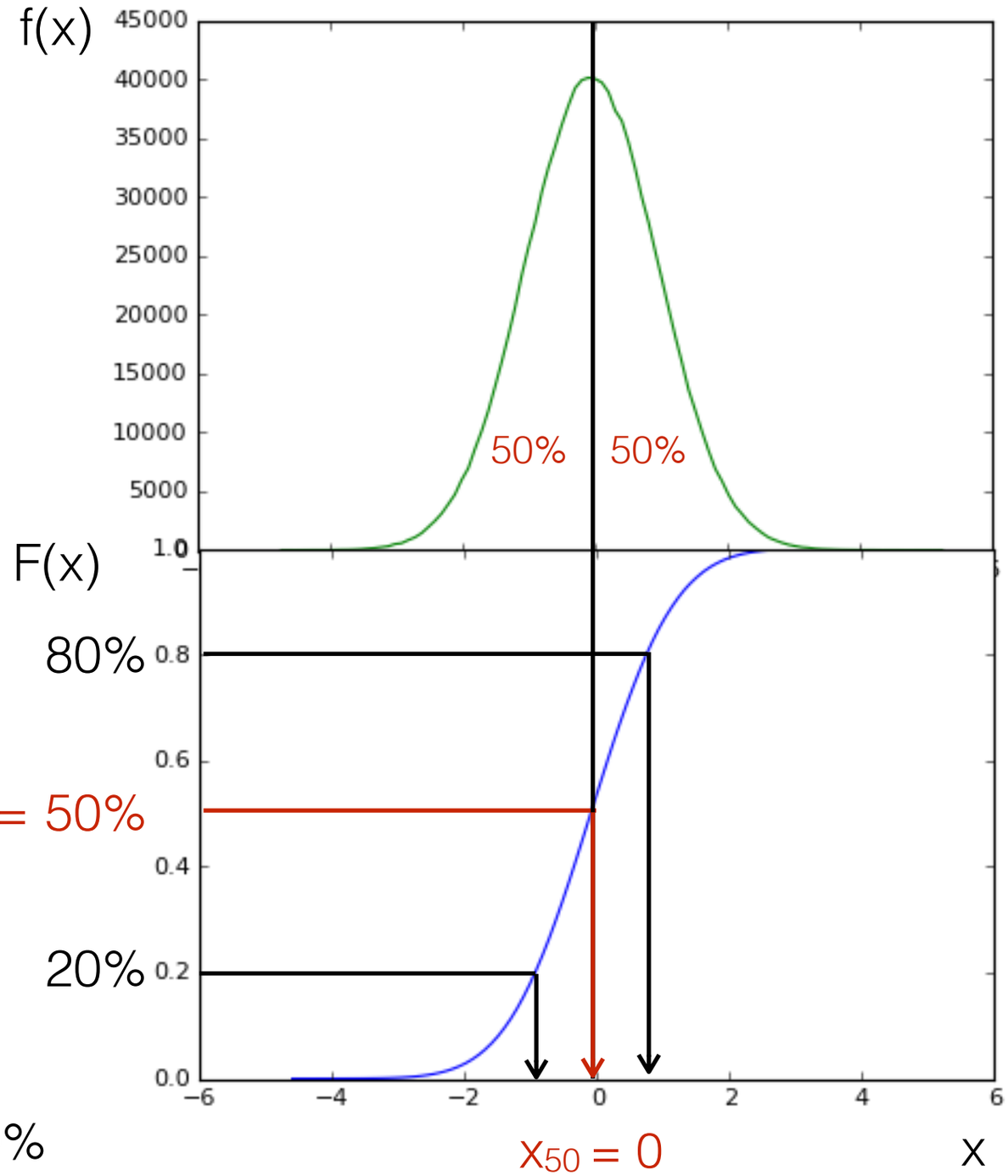
The probability to observe a random variable between two values x1 and x2 is

$$p(x_1 \leq x \leq x_2) = \int_{x_1}^{x_2} f(x')dx' = F(x_2) - F(x_1)$$

Cumulative and quantiles

x_q = quantile of order q (or q-point),
with $0 \leq q \leq 1$, of a distribution is
the value of x such that $F(x_q) = q$

The quantile is just the inverse of the cdf
$x_q = F^{-1}(q)$



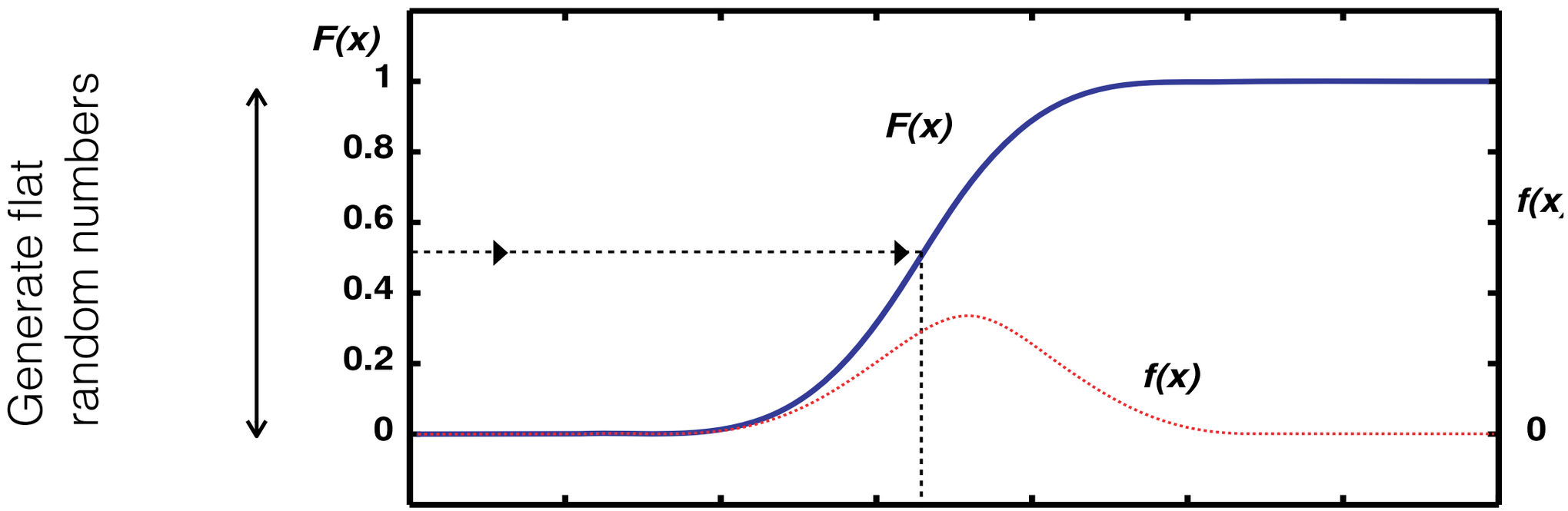
Quantiles with a name:
quartiles = 0%, 25%, 50%, 75%, 100%
percentiles = 0%,1%,2%,…, 98%, 99%, 100%

notebook https://gitlab.ethz.ch/mdonega/STAMET_FS18/blob/master/notebooks/cumulative.ipynb

Inverse cumulative method

The basic idea is to transform a uniform distribution of random numbers to any other (continuous) distribution by using its cdf.

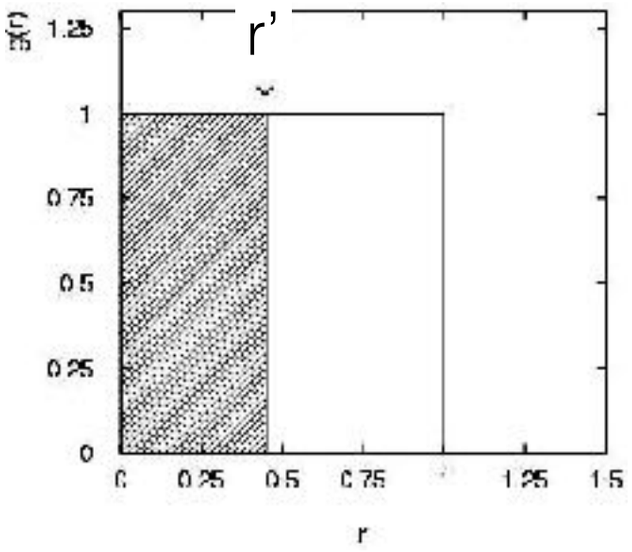
$$f(x)\,dx = U(0,1)\,du \qquad \int_{-\infty}^{x} f(t)\,dt = F(x) = u \qquad x = F^{-1}(u)$$

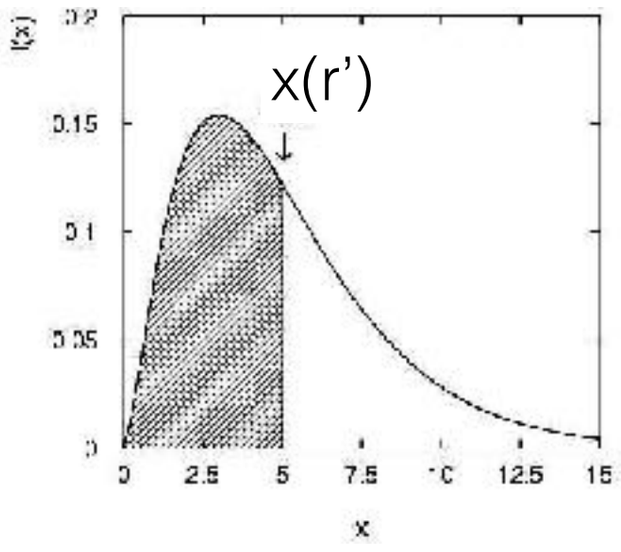


Inverse cumulative method

The principle is to match the quantile of one distribution to the corresponding quantile of the target.
e.g. the quantile corresponding to the first 10% is matched to the quantile of the first 10% of the target distribution.

Given n points $\{r_1, r_2, \ldots, r_n\}$ uniformly distributed in [0, 1], find $\{x_1, x_2, \ldots, x_n\}$ that follow f (x) by finding a suitable transformation x(r).





$$P(r \leq r') = P(x \leq x(r'))$$

$$\int_{-\infty}^{r'} g(r)\, dr = r' = \int_{-\infty}^{x(r')} f(x')\, dx' = F(x(r'))$$

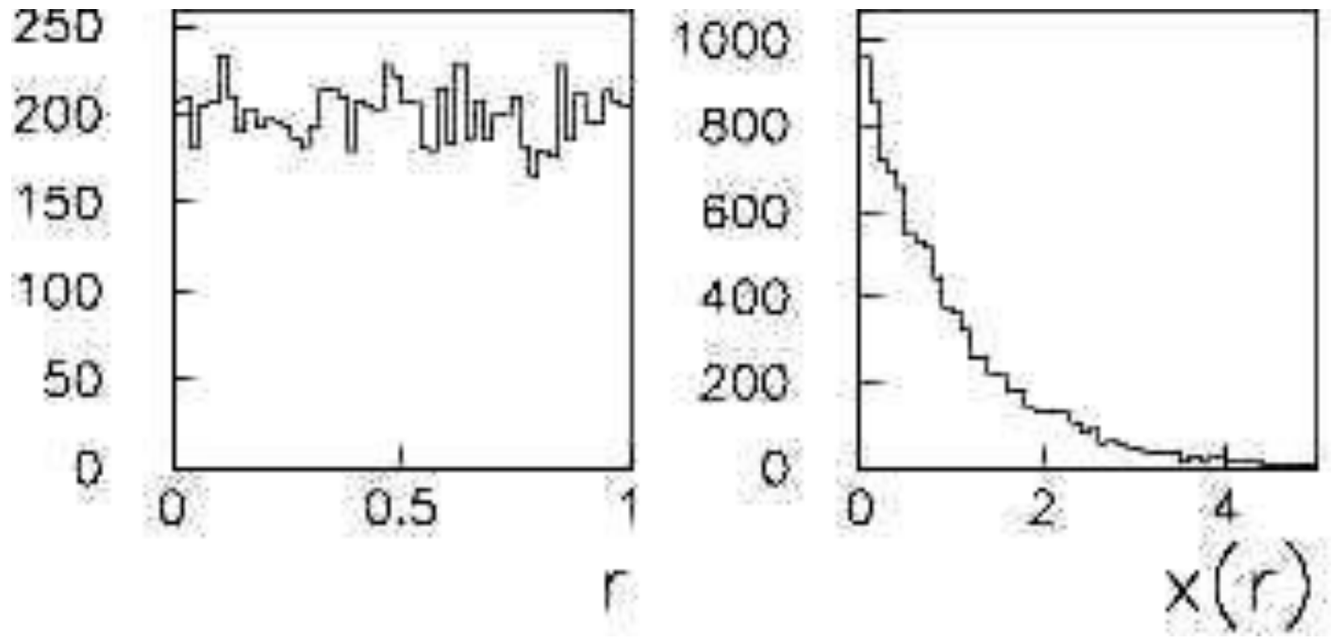
Solve r' = F(x(r')) inverting the cumulative x(r') = $F^{-1}(r')$

Example: exponential pdf

$$f(x;\xi) = \frac{1}{\xi} e^{-x/\xi} \quad (x \geq 0)$$

The cumulative is $$\int_0^x \frac{1}{\xi} e^{-x'/\xi}\, dx' = r$$

Invert it: $$x(r) = -\xi \ln(1-r)$$



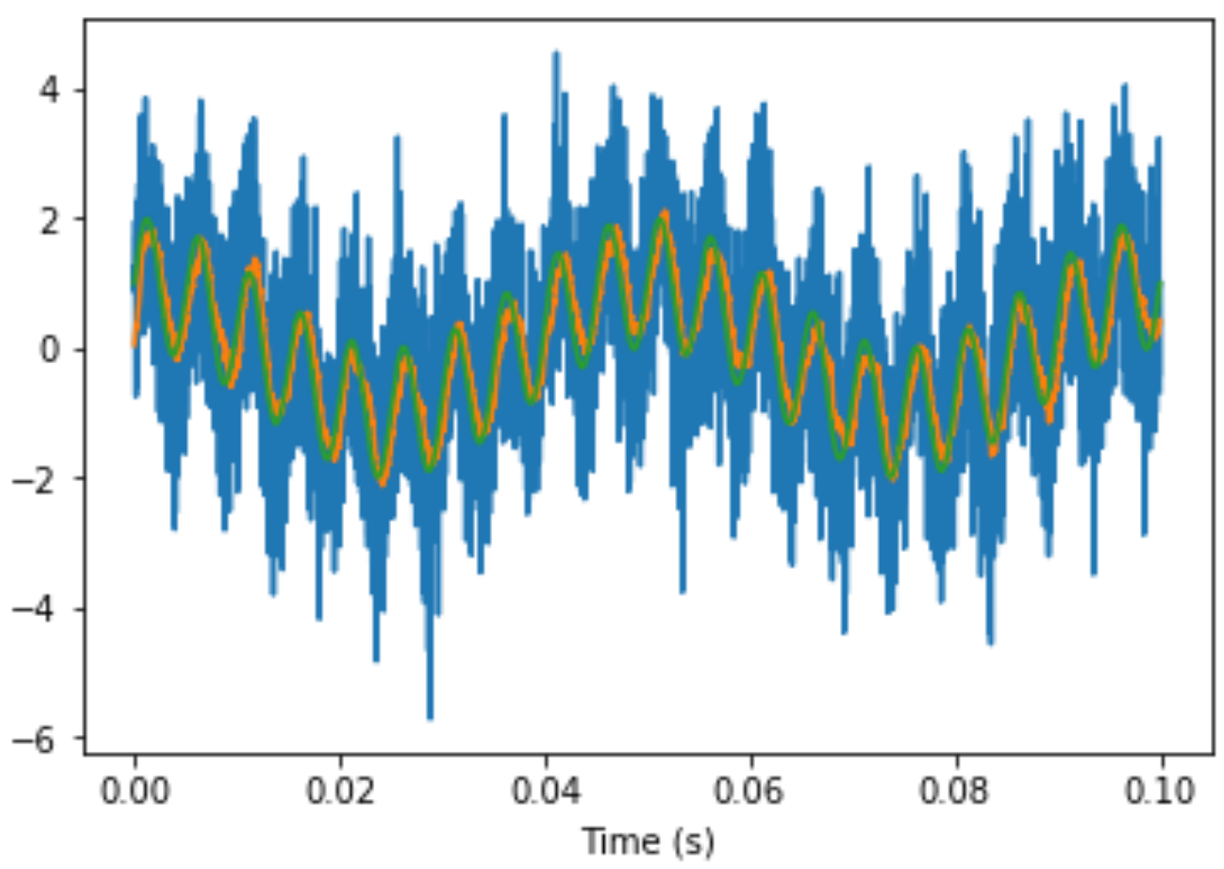
Trick: sliding mean

Any measurement you do is affected by noise.

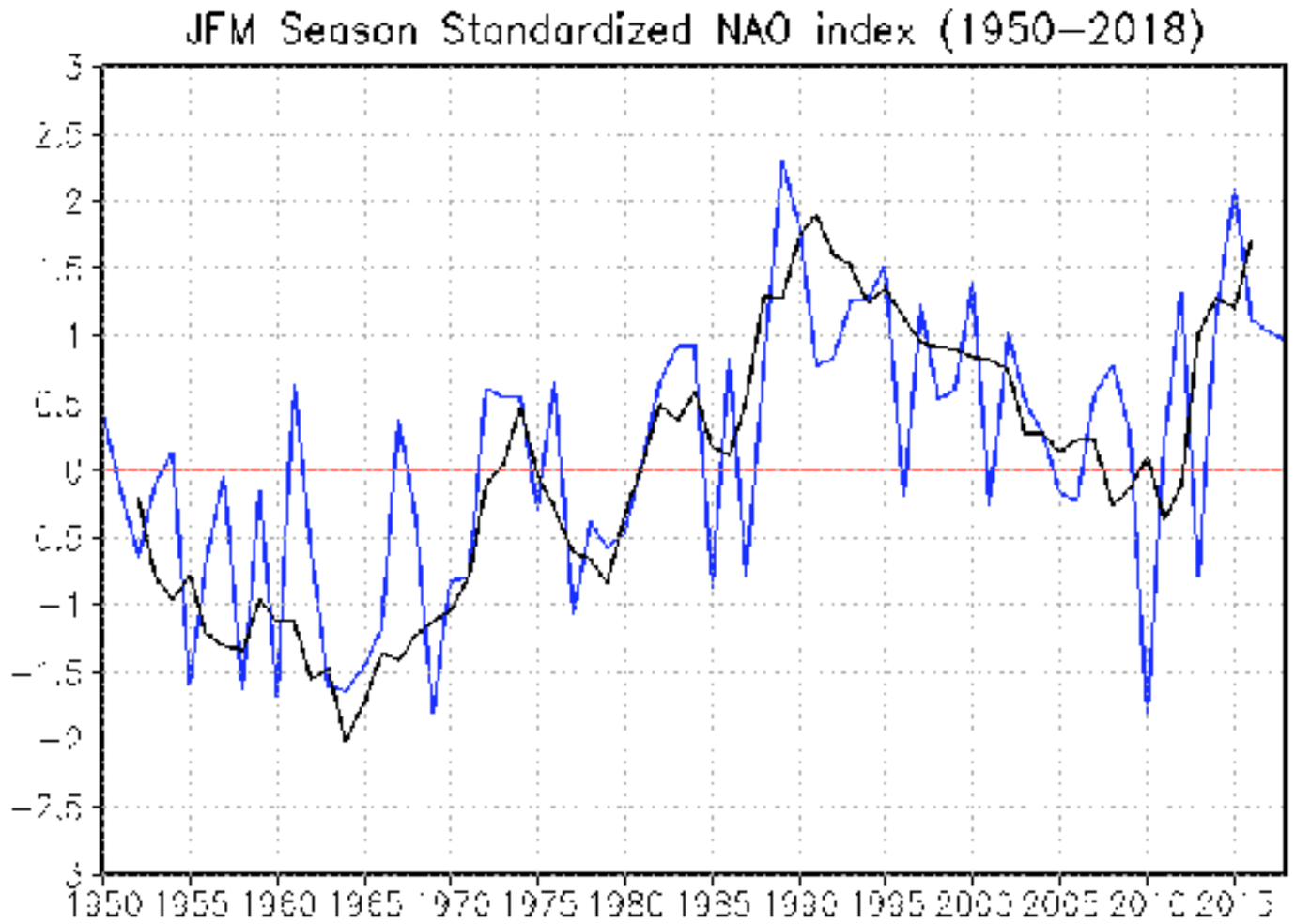
To reduce the effect of noise we learnt to take several measurements (sampling) and compute mean/variance them (week 1 - gaussian hp) or fit them and extract the parameters of the underlying pdf (week 5-6).

When plotting data, the physical signal you want to present might be obscured by large noise (random) fluctuations overlapped to it.

A simple trick to show the data is to average the data in a “sliding window” on the “x-axis” and plot it overlapped to the original data.



Trick: sliding mean



http://www.cpc.ncep.noaa.gov/products/precip/CWlink/pna/JFM_season_nao_index.shtml

The North Atlantic Oscillation (NAO) index is based on the surface sea-level pressure difference between the Subtropical (Azores) High and the Subpolar Low. https://www.ncdc.noaa.gov/teleconnections/nao/

Backup

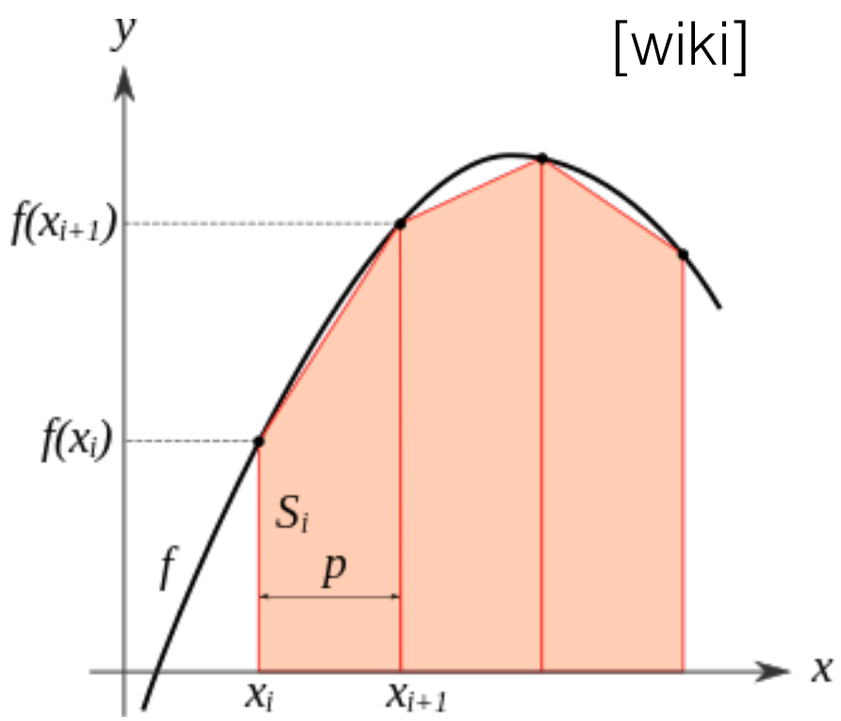
Monte Carlo integration

Reminder: simple numerical (definite) integral computation of $I = \int_a^b f(x)dx$

Split the range [a,b] in n intervals and approximate the function in the interval by its value in the centre of the bin, or interpolating between the two computing the area of the trapezoid

$$\frac{b-a}{n}\sum_{i=1}^{n} f(x_i) \qquad \text{with} \quad x_i = a + (i-0.5)(b-a)$$

deterministic approach



[wiki]

You can evaluate the function at the central value of the bin, or at the min and max and approximate the function with a linear interpolation, etc ...

The uncertainty goes as $n^{-2/d}$, where d is the number of dimensions of the integral.

Monte Carlo integration

The most naive implementation of the Monte Carlo technique follows the same principle but instead of using a regularly spaced grid points, we use randomly distributed numbers

$$I = \int_\Omega f(\vec{x}) d\vec{x}$$ where the Ω is in R^m and has volume V

The deterministic approach samples the function on a regular grid $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$

$$I \sim V \cdot \frac{1}{N} \sum_{i=1}^{N} f(\vec{x}_i) = V < f >$$ for large N

The uncertainty on I can be evaluated from the variance $V(I) = \frac{V^2}{N^2} \sum_{i=1}^{N} V(f) = V^2 \frac{V(f)}{N}$

obtaining $\delta I \sim V \frac{\sqrt{V(f)}}{\sqrt{N}} = V \frac{\sigma_N}{\sqrt{N}}$

The uncertainty goes as 1/sqrt{N} and it does not depend on the number of dimensions

Reducing the variance

There are several method to reduce the variance. Among the most used are:

Stratification: divide the range of integration in two regions and generate half of the Monte Carlo points in each of the regions reduces the variance. The reason is that in this way we allow a more uniform sampling of the distribution.

Importance sampling: reweight input distribution to emphasize “high-impact” regions of sampling space transform the sampling space.
Draw from an alternative distribution whose support/impact is concentrated in the high-impact sampling region:

$$\int_F sf(s)ds = \int_G s\frac{f(s)}{g(s)}g(s)ds$$

sampling s from f(s) distribution is equivalent to sampling s*w(s) from g(s) distribution, with importance sampling weight w(s) = f(s)/g(s) . (note: f and g should have same support)

VEGAS: samples points from the probability distribution such that the points are more densely distributed in the regions that make the largest contribution to the integral.

Toy simulations

"A good physicist knows how to build toy models." (Anonymous)

Toys are used to get a first understanding of a process picking only its essential characteristics (here is where the physicist has to be good) and avoiding all complications of real experimental life.

Toys are also an ubiquitous tool used in statistics (e.g. for hypothesis testing)

MLE uncertainty from toys

A different way to compute the uncertainty of an estimator:

- generate N toy-MCs of the same size n-evts of the data sample (using realistic resolutions etc.) —> N sets of $[x_k]$
- use as a true (input) value for parameter a in the MCs, the â obtained from data
- determine in every pseudo-experiment i the estimator $\hat{a}_i$ from maximum L
- determine the sample variance of â from the N pseudo-experiments

Bibliography

MC:
Lyons ch 6