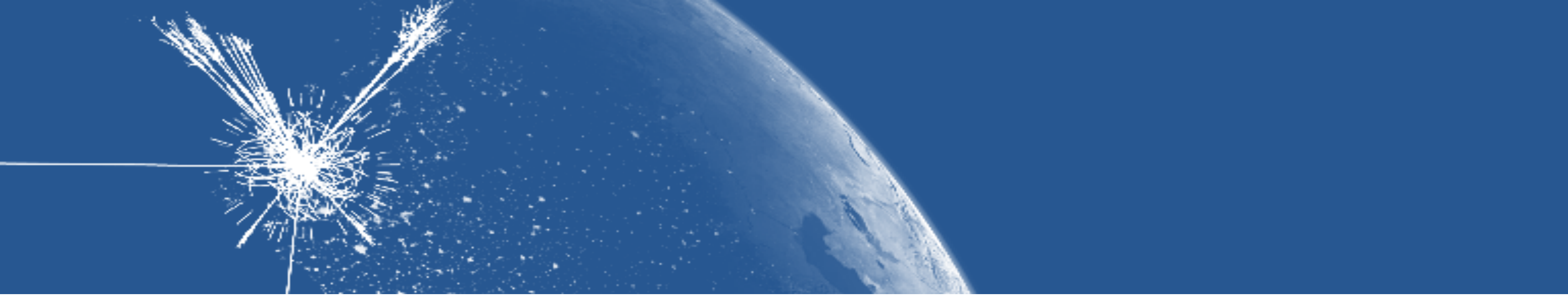
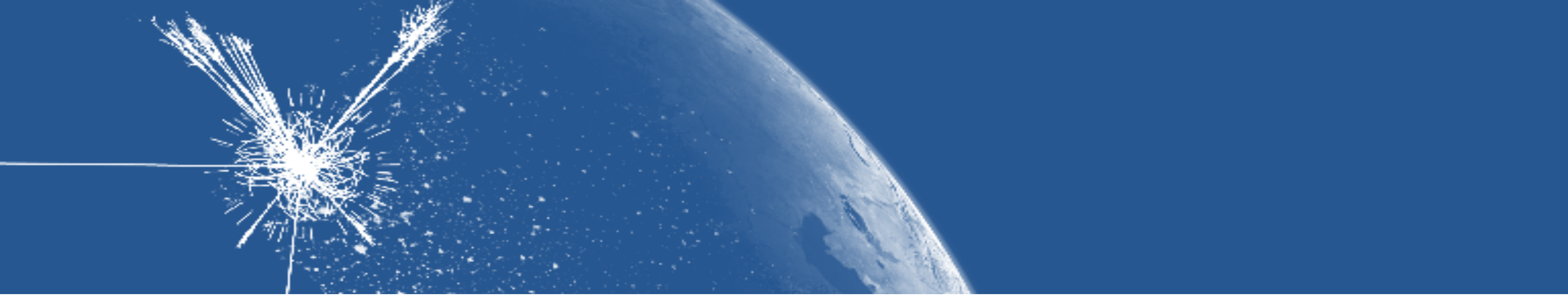


Introduction to probability and statistics (4)

Andreas Hoecker (CERN)

CERN Summer Student Lecture, 17–21 July 2017

If you have questions, please do not hesitate to contact me: **andreas.hoecker@cern.ch**



Outline (4 lectures)

1st lecture:

- Introduction
- Probability

2nd lecture:

- Probability axioms and hypothesis testing
- Parameter estimation
- Confidence levels (some catch up to do…)

3rd lecture:

- Maximum likelihood fits
- Monte Carlo methods
- Data unfolding

4th lecture:

- Multivariate techniques and machine learning

Monte Carlo techniques

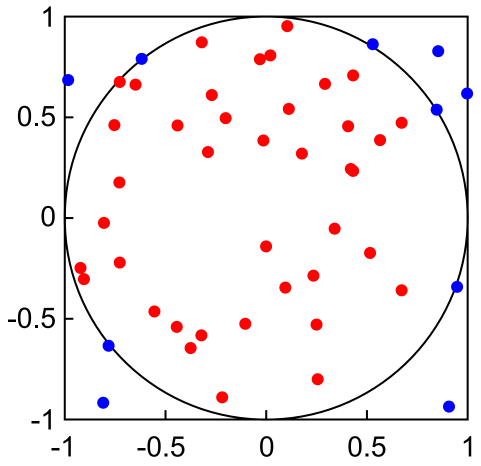
Why "Monte Carlo" techniques ?

Monte Carlo (MC) techniques are computational algorithms that rely on repeated random sampling to obtain numerical results

They are used when analytical solutions are too complex or not even known

Examples:

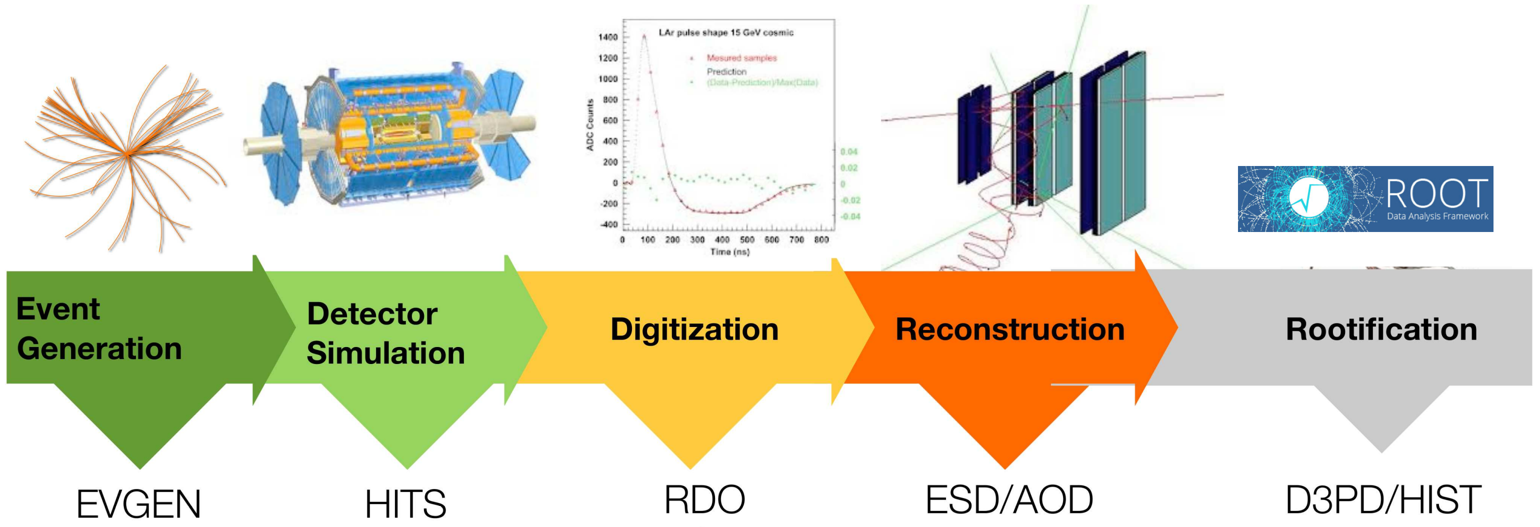
- Numerical integration of complex, multidimensional integrals (eg phase-space integration of matrix elements describing particle physics processes)
- Simulation of LHC particle collisions ("events") as measured by the particle detectors. This involves:
  - Matrix element generation of collision
  - Decay of produced particles and propagation of stable particles through detector material
  - Electronic response of active detector layers, and reconstructions of signals
  - Physics analysis

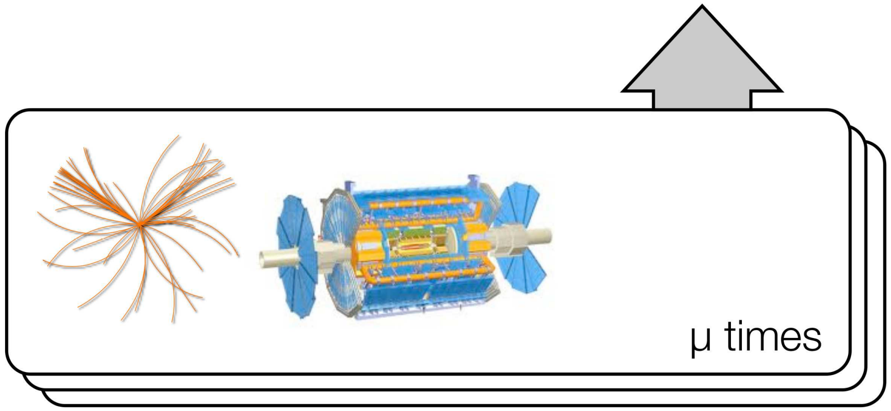


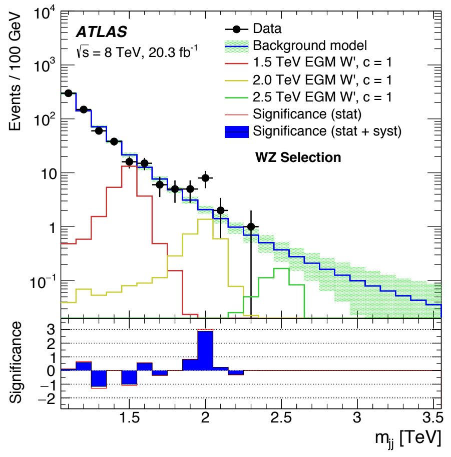
Numerical estimation of circle area by taking ratio of red to red+blue points times the square's area

Figure from: https://en.wikipedia.org/wiki/Monte_Carlo_integration

Typical Monte Carlo Production Chain







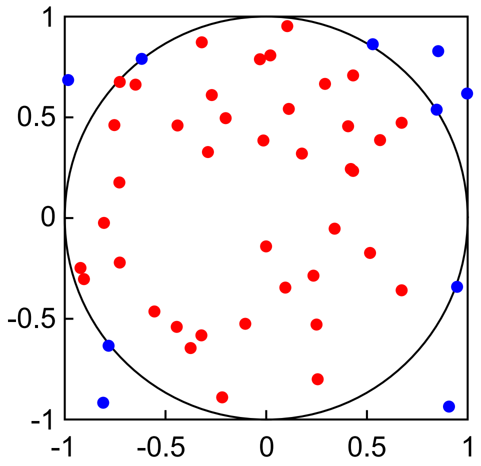
Why “Monte Carlo” techniques ?

Monte Carlo (MC) techniques are computational algorithms that rely on repeated random sampling to obtain numerical results

They are used when analytical solutions are too complex or not even known

Examples:

- Numerical integration of complex, multidimensional integrals (eg phase-space integration of matrix elements describing particle physics processes)
- Simulation of LHC particle collisions (“events”) as measured by the particle detectors. This involves:
  - Matrix element generation of collision
  - Decay of produced particles and propagation of stable particles through detector material
  - Electronic response of active detector layers, and reconstructions of signals
  - Physics analysis
- Simpler: error propagation and estimation of error on a measured quantity with unknown property (→ next slide)



Numerical estimation of circle area by taking ratio of red to red+blue points times the square’s area

Figure from: https://en.wikipedia.org/wiki/Monte_Carlo_integration

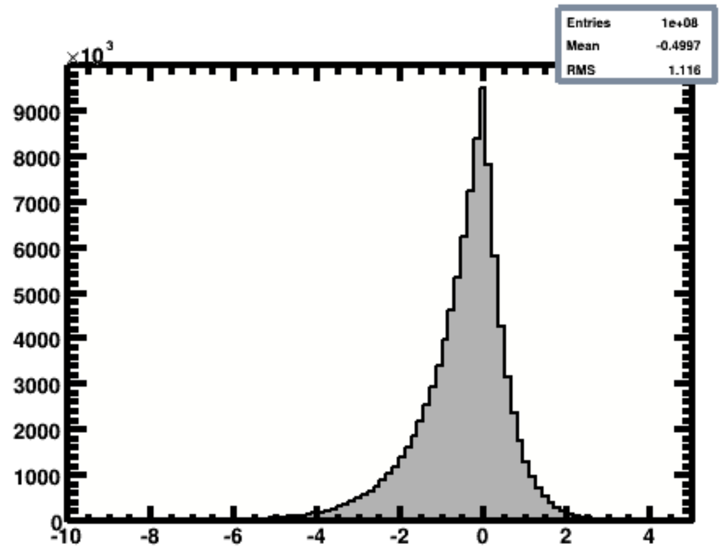
Bootstrap method

Consider the following problem: a quantity $\boldsymbol{x}$ was measured $\boldsymbol{N}$ times: $\boldsymbol{x_i}$ $(\boldsymbol{i = 1 \dots N})$

One wants to determined a derived quantity $\boldsymbol{y(x_1, \dots, x_N)}$, and needs an error for it.

→ Error propagation (remember: $\sigma_y = \left.\frac{dy(x)}{dx}\right|_{x=\bar{x}} \cdot \sigma_x$), but it requires to know the PDF of $\boldsymbol{x}$

Assume the distribution of the measured $\boldsymbol{x_i}$ looks like this:



- This ***is*** a PDF, and the best available information
- One can obtain a new set to “simulate” the measurements by applying *resampling with replacement*
- That is: one draws $\boldsymbol{M}$ events from the ensemble allowing to re-draw the same event multiple times
- One does this many times

→ **Bootstrapping**

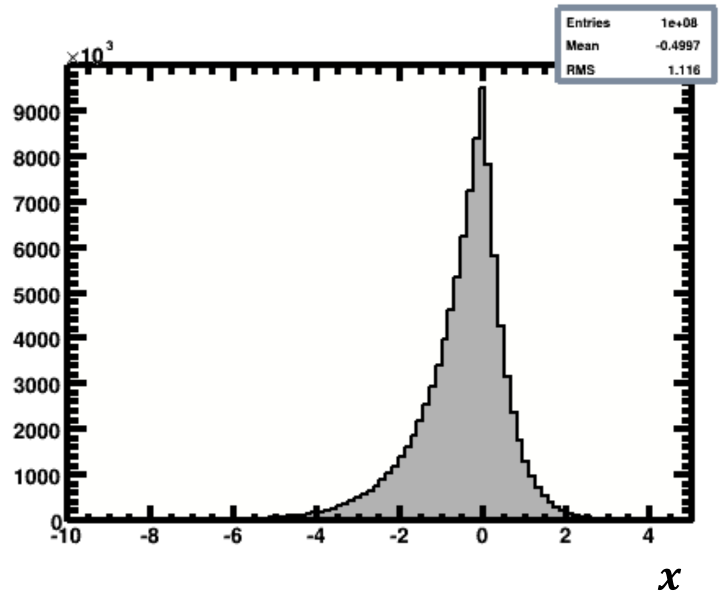
Bootstrap method

Consider the following problem: a quantity $\boldsymbol{x}$ was measured $\boldsymbol{N}$ times: $\boldsymbol{x_i}\ (\boldsymbol{i = 1 \dots N})$

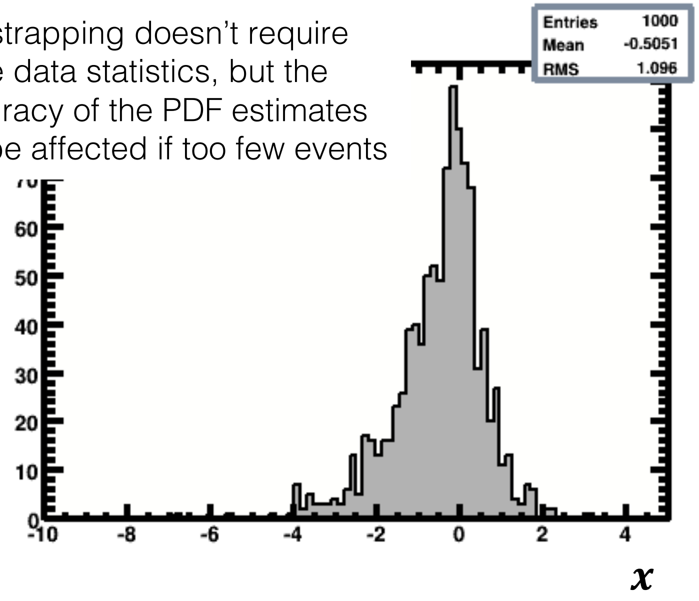
One wants to determined a derived quantity $\boldsymbol{y(x_1, \dots, x_N)}$, and needs an error for it.

→ Error propagation (remember: $\sigma_y = \left.\frac{dy(x)}{dx}\right|_{x=\bar{x}} \cdot \sigma_x$), but it requires to know the PDF of $\boldsymbol{x}$

Assume the distribution of the measured $\boldsymbol{x_i}$ looks like this:



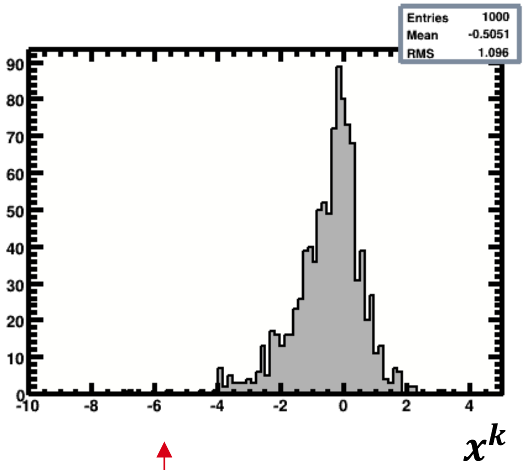
Boostrapping doesn't require large data statistics, but the accuracy of the PDF estimates will be affected if too few events

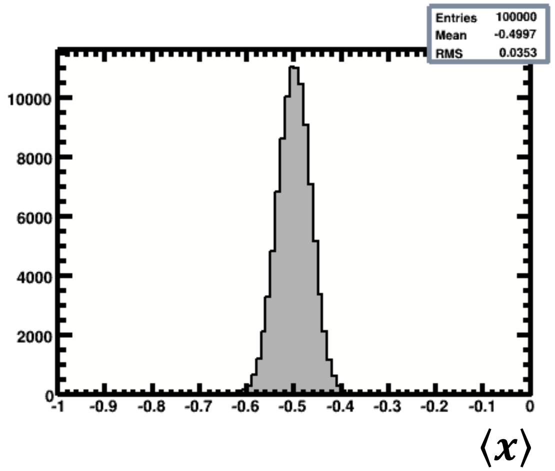


Bootstrap method — does it really work ?

Let's try with our toy example: simulate 100,000 experiments with 1,000 events each sampled from some analytic PDF (Nature's unknown truth) that we want to approximate

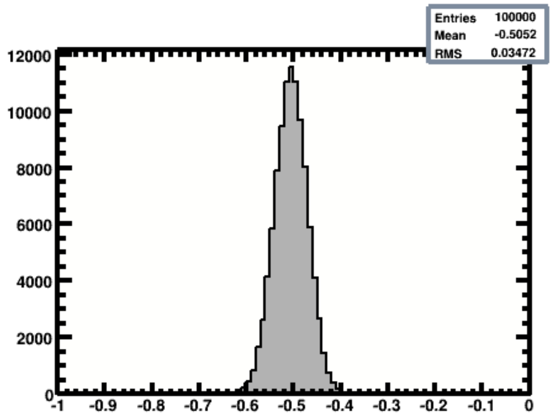
One example experiment $\boldsymbol{k}$





Distribution of mean values among all experiments

Now, use **this experiment** to sample 100000 bootstrap experiments

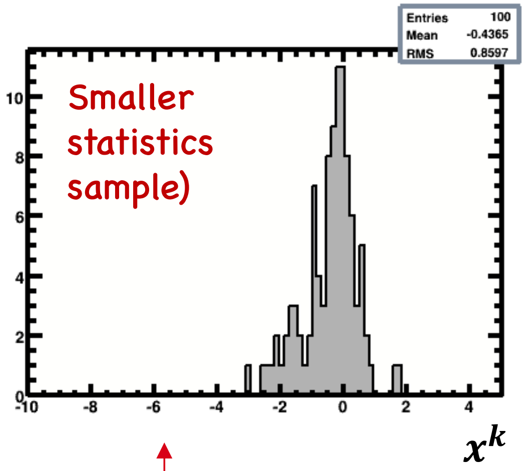


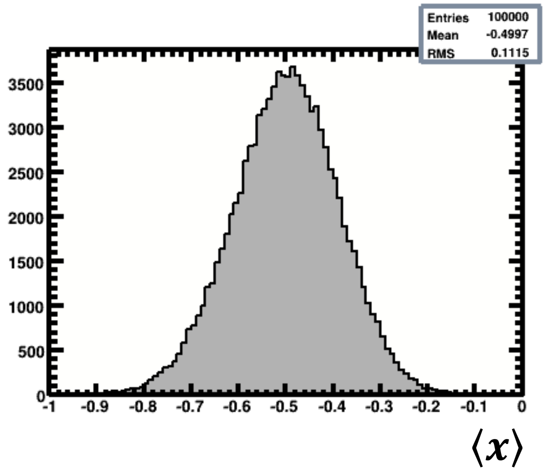
Satisfying result: RMS reproduced within 1.7%

Bootstrap method — does it really work ?

Let's try with our toy example: simulate 100,000 experiments with **100** events each sampled from some analytic PDF (Nature's unknown truth) that we want to approximate

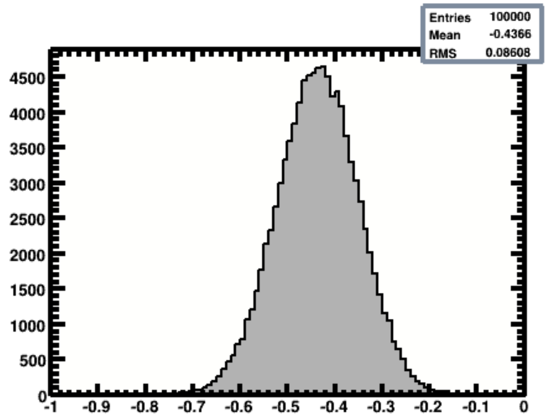
One example experiment $\boldsymbol{k}$





Distribution of mean values among all experiments

Now, use **this experiment** to sample 100000 bootstrap experiments



Mediocre result: RMS reproduced within 30%

See: http://projecteuclid.org/download/pdf_1/euclid.ss/1177013815 for more information on bootstrap method

Jackknife resampling method (also called: leave-one-out cross validation)

Old method (~1950), basically replaced by bootstrap. Nevertheless instructive to know

Let's again consider: a quantity $\boldsymbol{x}$ was measured $\boldsymbol{N}$ times: $\boldsymbol{x_i}\ (\boldsymbol{i} = \mathbf{1} \ldots \boldsymbol{N})$

One wants to determine a derived quantity $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x_1}, \ldots, \boldsymbol{x_N})$, and needs an error for it:

- Study how $\boldsymbol{y}$ changes when *leaving out one measurement* at the time

  let: $y_i = y_i(x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_N)$,

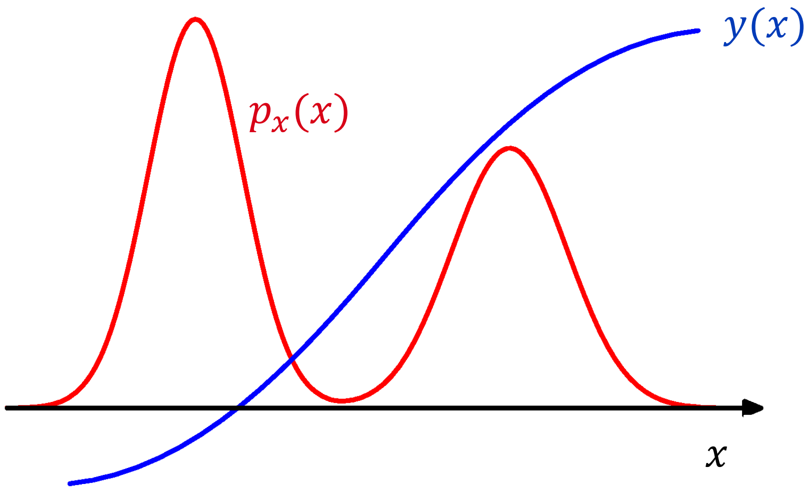
  and compute the bias-corrected jackknife estimator of y: $y_i^{\text{Jack}} = Ny - (N-1)y_i$

- Plot y_i^{Jack} for all $i = 1 \ldots N$ and treat them as if they were independent samples of the measured quantity.

- Compute mean or variance from y_i^{Jack} ensemble

Monte Carlo (MC) integration

Want to *numerically* compute an expectation value: $\boldsymbol{E[y] = \int y(x) p_x(x) dx}$



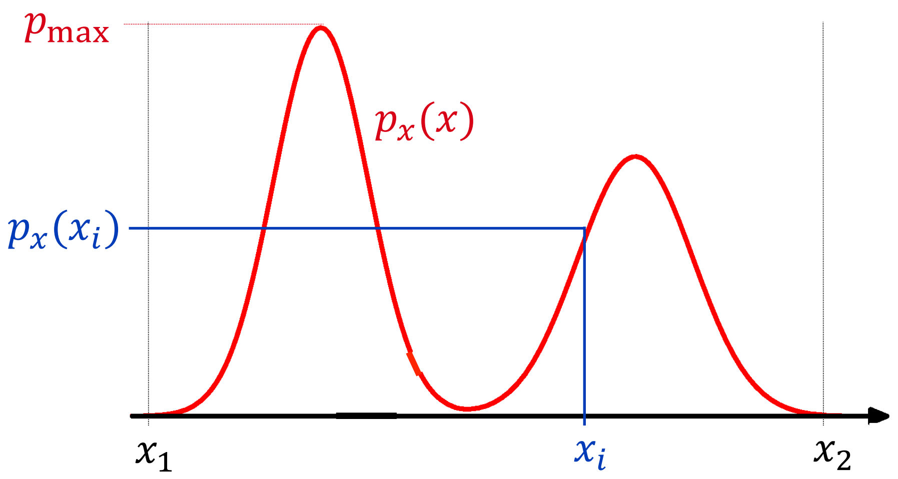
- Simplest solution: n-equidistant stepwise summation
- Works in 1, possibly *few* dimensions D
- Bad curse of dimensionality: exponential growth of n with D
- Random MC phase-space sampling converges faster for large D

MC integration to compute $E[y]$ requires MC sampling according to PDF $p_x(x)$

That given, one finds: $\boldsymbol{\int y(x) p_x(x) dx \approx \frac{1}{N_{\text{samples}}} \sum_{i=1}^{N_{\text{samples}}} y(x_i)}$

“Hit-or-miss” rejection sampling

Simplest way to generate random numbers (to “sample”) according to PDF $p_x(x)$



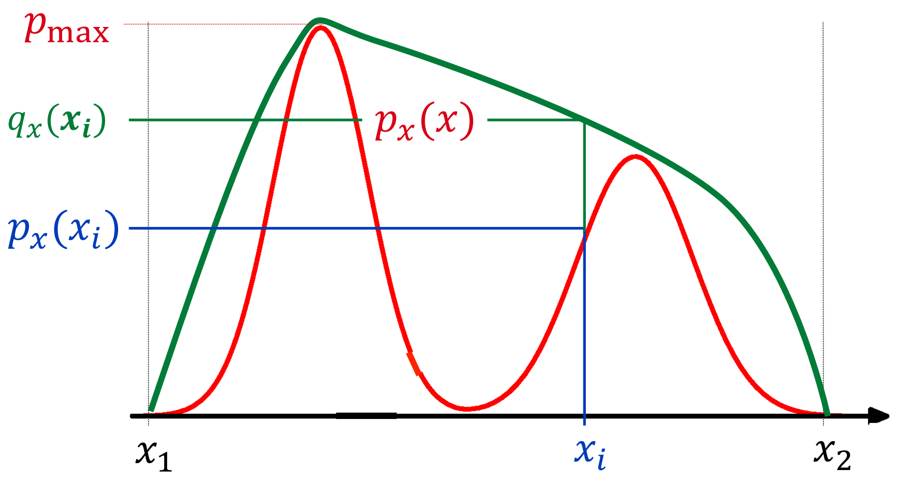
1. Generate uniform random number in interval $[x_1, x_2] \rightarrow \boldsymbol{x_i}$ and $\boldsymbol{p_x(x_i)}$
2. Generate another uniform random number in interval $[0, p_{\max}] \rightarrow \boldsymbol{p_i}$
3. If $\boldsymbol{p_i} < \boldsymbol{p_x(x_i)}$: **accept** $\boldsymbol{x_i}$; **else**: **reject**

$\boldsymbol{q_x(x)} = p_{\max}$, the (here uniform) PDF of generated $\boldsymbol{x}$ values defines *proposal distribution*

$\rightarrow$ One could be smarter to have a larger “accept” rate (efficiency)

Rejection sampling

One can choose a (known) proposal distribution $q_x(x)$ closer to $p_x(x)$



1. Generate random number according to $q_x(x)$ in interval $[x_1, x_2] \rightarrow \boldsymbol{x_i}$ and $\boldsymbol{p_x(x_i)}$
2. Generate another uniform random number in interval $[0, q_x(\boldsymbol{x_i})] \rightarrow \boldsymbol{p_i}$
3. If $\boldsymbol{p_i} < \boldsymbol{p_x(x_i)}$: **accept** $\boldsymbol{x_i}$; **else: reject**

Fraction of accepted events now larger than before (there are techniques to adapt automatically the proposal distribution during the generation)

$\rightarrow$ Can be even more clever if only integration needed, no random event generation

Markov chain Monte Carlo (MCMC) method

So far, the accuracy of the sampling depended on how closely $\boldsymbol{q_x(x)}$ follows $\boldsymbol{p_x(x)}$

This is a problem for sparsely known $\boldsymbol{p_x(x)}$ in case of complex multi-*D* structure. Every random point is chosen independently of every other one.

Markov chain: (eg, ”random walk”) Andrey Markov (1856–1922)

- Consecutive random steps depend on previous ones in random variable space
- Allows to favor stepping into regions where $\boldsymbol{p_x(x)}$ is large

Several Markov chain algorithms: *Metropolis*, *Gibbs*, … → next pages

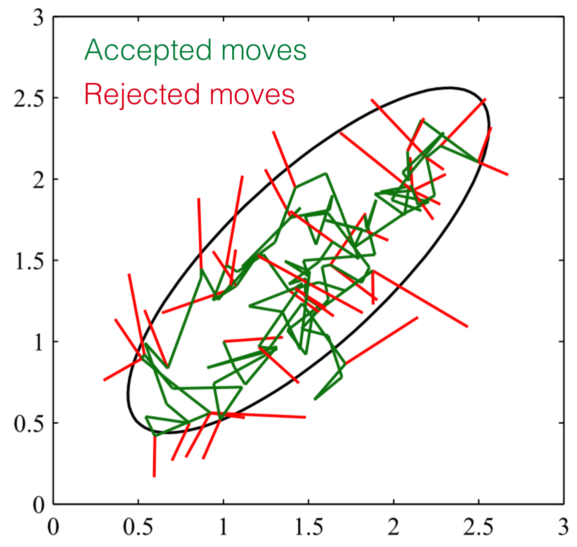
Metropolis sampling algorithm (1953)

Autocorrelation sampling of PDF $\boldsymbol{p_x(x)}$

1. Start anywhere in (multidimensional) $\boldsymbol{x}$ space and sample this point: $\boldsymbol{p_1 = p_x(x_1)}$
2. Provide proposal distribution $\boldsymbol{q_x(x_2|x_1)}$ to move from $\boldsymbol{x_1} \rightarrow \boldsymbol{x_2}$
   - $\boldsymbol{q_x(x_2|x_1)}$ could be Gaussian with appropriate metric in $\boldsymbol{x}$ space to cover full space
   - Accept $\boldsymbol{x_2}$ if: $\boldsymbol{p_2 > p_1}$ else: according to probability $\boldsymbol{p_2/p_1}$
   - If not accepted, set $\boldsymbol{x_2 = x_1}$ (no walk)
3. Iterate step 2. for $\boldsymbol{x_3}$ vs. $\boldsymbol{x_2}$, etc.

Sample points $\boldsymbol{x}$ will wander closer and closer to the peak of the PDF, still jumping enough from time to time to sample the whole space

(Algorithm requires sufficient iterations. Test by checking stability of derived result, or by comparing several sampling ensembles obtained with different start values)

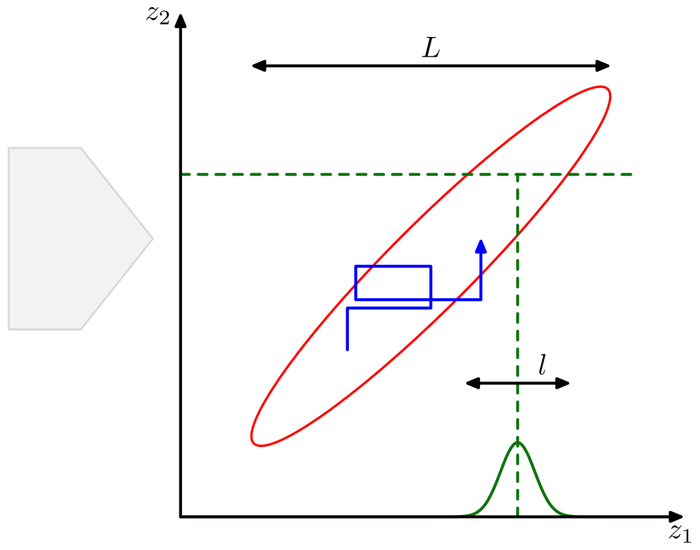


This subfigure from PRML, Bishop (2006)

Gibbs sampling algorithm (1984)

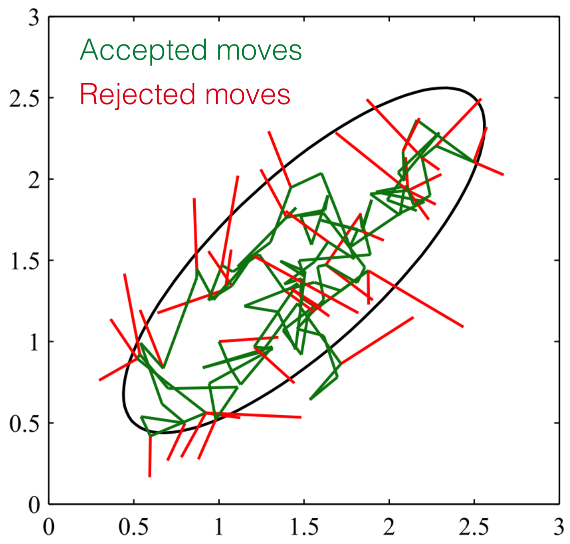
A method with no rejection:

1. Instead of moving along all dimensional components (and reject low-probability moves), the Gibbs sampler moves along 1 component according to the PDF conditioned on all other components.
2. Cycle through all components



Markov chain Monte Carlo usually converge fast and, if metric well chosen, cover the full space

However: care needs to be taken as the sample points $\boldsymbol{x}$ are correlated (with either sampling method, although there exist tricks to reduce the correlations): it depends on the application whether or not this is an issue



This subfigure from PRML, Bishop (2006)

Data unfolding

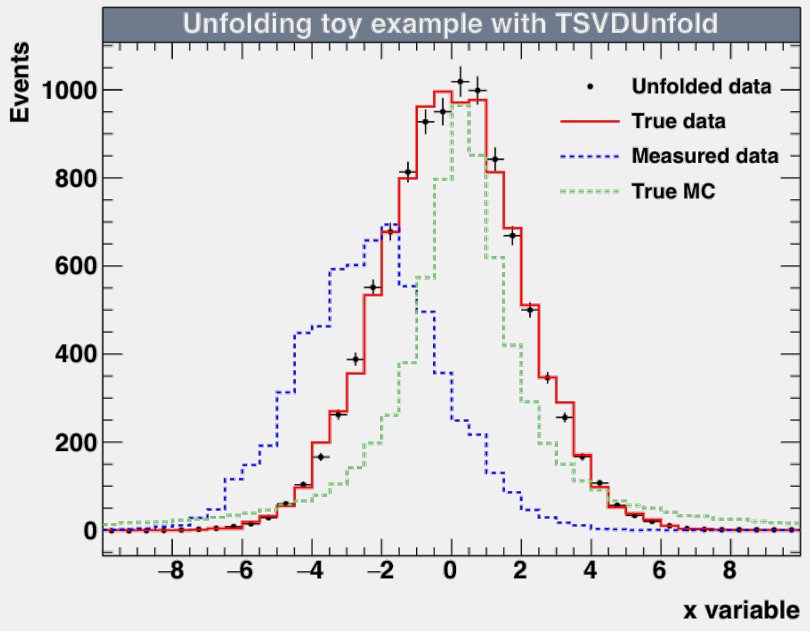


Figure from arXiv:1112.2226v1

Data unfolding — introduction

"Unfolding" means correcting measured data for any effects related to the measurement device. The unfolded data can be directly compared to theory or among experiments

Consider a measured histogram $\boldsymbol{y}^{\text{data}} = \{y_1^{\text{data}}, \dots, y_m^{\text{data}}\}$, a corresponding Monte Carlo histogram $\boldsymbol{y}^{\text{MC}}$ of the same process as the data that underwent full detector simulation, its *truth* distribution (ie, before detector simulation) $\boldsymbol{x}^{\text{MC}} = \{x_1^{\text{MC}}, \dots, x_n^{\text{MC}}\}$, and the $m \times n$ matrix $\boldsymbol{A}^{\text{MC}}$ obtained from MC that describes the "smearing" process due to the measurement:

$$\boldsymbol{A}^{\text{MC}} \cdot \boldsymbol{x}^{\text{MC}} = \boldsymbol{y}^{\text{MC}}$$

Note that in general $\boldsymbol{y}^{\text{data}} \neq \boldsymbol{y}^{\text{MC}}$ (the physics leading to $\boldsymbol{y}^{\text{data}}$ is what we want to measure), but we assume $\boldsymbol{A}^{\text{MC}} = \boldsymbol{A}^{\text{data}}$ (we know the detector response).

Hence, to obtain the truth information $\boldsymbol{x}^{\text{data}}$, one "just" needs to invert $\boldsymbol{A}$:

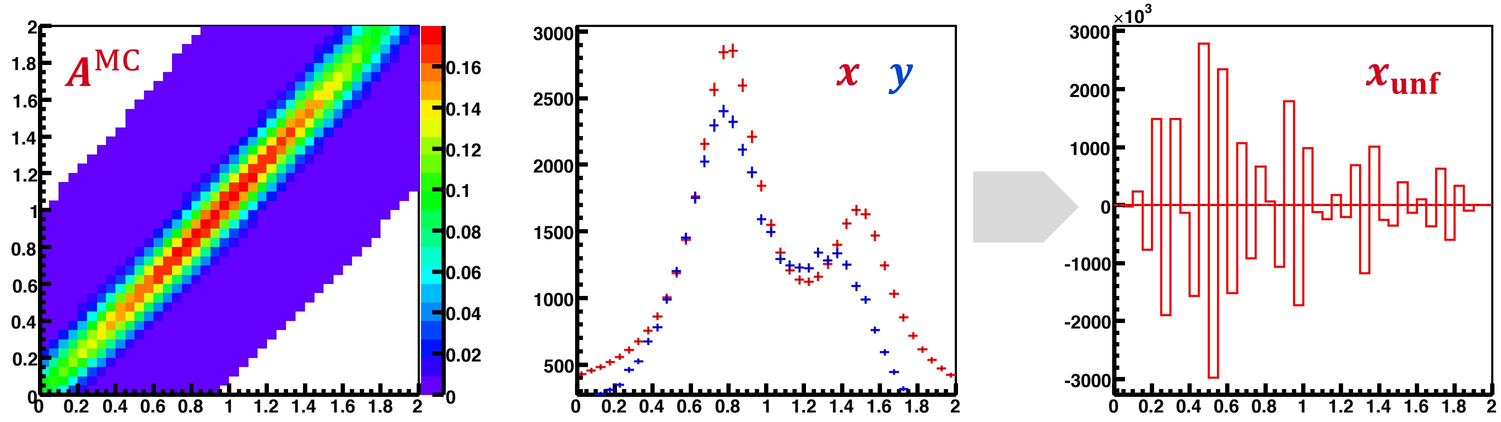
$$\boldsymbol{x}^{\text{data}} = \left(\boldsymbol{A}^{\text{MC}}\right)^{-1} \cdot \boldsymbol{y}^{\text{data}}$$

This is where the trouble begins …

Data unfolding — introduction

V. Kartvelishvili at https://indico.cern.ch/event/107747

The distribution $\boldsymbol{y}^{\mathrm{data}}$ and the matrix $\boldsymbol{A}^{\mathrm{MC}}$ have finite statistics. An attempt to solve the problem directly and "exactly" will end up looking like this:



Large bin-to-bin oscillations ("Gibbs phenomenon") in unfolded spectrum

The poor solution, bin-by-bin corrections, $x_i^{\mathrm{data}} = \frac{x_i^{\mathrm{MC}}}{y_i^{\mathrm{MC}}} \cdot y_i^{\mathrm{data}}$, only works if $\boldsymbol{A}^{\mathrm{MC}}$ is square and ~diagonal so that the ratio $x_i^{\mathrm{MC}}/y_i^{\mathrm{MC}}$ corrects for mainly efficiency effects, or if $y_i^{\mathrm{data}} \cong y_i^{\mathrm{MC}}$.

A better solution is to regularise the matrix inversion problem …

Data unfolding — regularisation

Regularisation damps the oscillations, by suppressing statistically insignificant bins in the data distribution and response matrix.

In simplified form, one can write the unfolding problem as a minimisation of

$$\chi^2(\boldsymbol{x}^{\mathrm{data}}) = \left(\boldsymbol{A}^{\mathrm{MC}} \cdot \boldsymbol{x}^{\mathrm{data}} - \boldsymbol{y}^{\mathrm{data}}\right)^T \left(\boldsymbol{A}^{\mathrm{MC}} \cdot \boldsymbol{x}^{\mathrm{data}} - \boldsymbol{y}^{\mathrm{data}}\right) + \tau \cdot \left(C\boldsymbol{x}^{\mathrm{data}}\right)^T \left(C\boldsymbol{x}^{\mathrm{data}}\right)$$

where C is a matrix and $C\boldsymbol{x}^{\mathrm{data}}$ is the sum of squares of the 2nd derivative of $\boldsymbol{x}^{\mathrm{data}}$

Minimising χ^2 wrt. the first term only corresponds to the (bad) exact inversion solution. The second term regularises the inversion by damping the oscillations.

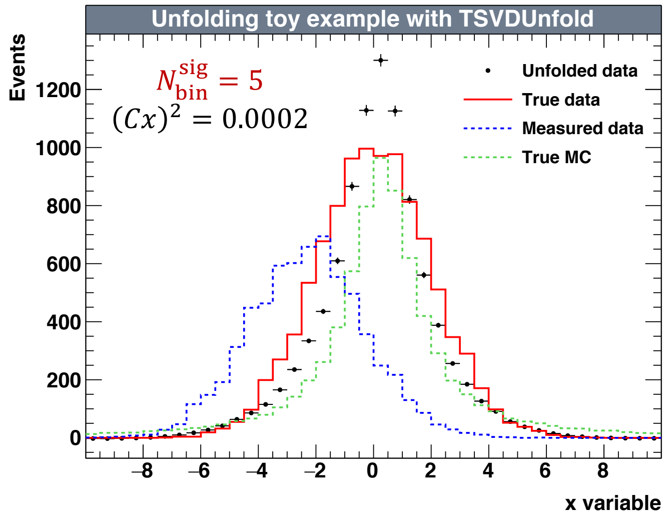
The parameter τ regulates the strength of the damping:

- If τ too small $\rightarrow$ oscillations
- If τ too large $\rightarrow$ information in $\boldsymbol{x}^{\mathrm{data}}$ is suppressed
  ($\boldsymbol{x}^{\mathrm{data}}$ becomes too “smooth” and will be biased towards $\boldsymbol{x}^{\mathrm{MC}}$)
- The right choice captures the significant information and discards the rest

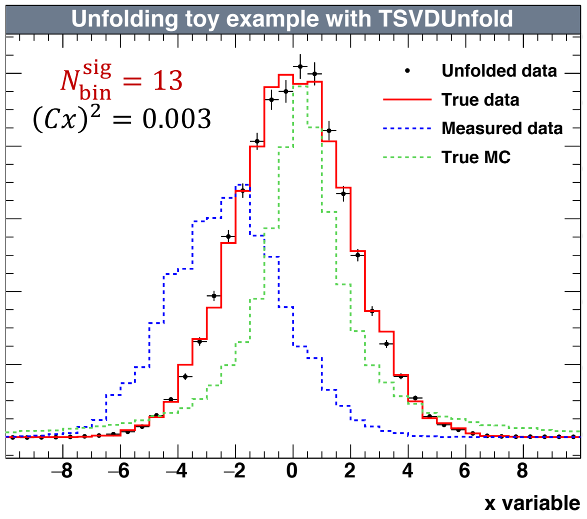
Data unfolding — example

TSVUnfold @ ROOT

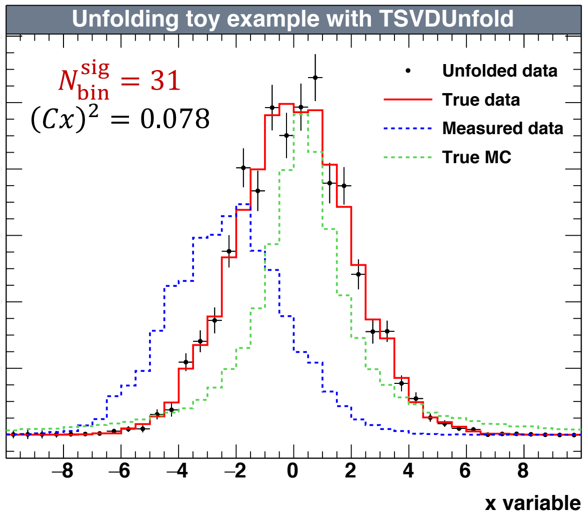
Over-regularised



Best regularisation choice



Under-regularised



The parameter τ regulates the strength of the damping:

- If τ too small $\rightarrow$ oscillations
- If τ too large $\rightarrow$ information in $\boldsymbol{x}^{\text{data}}$ is suppressed
  ($\boldsymbol{x}^{\text{data}}$ becomes too "smooth" and will be biased towards $\boldsymbol{x}^{\text{MC}}$)
- The right choice captures the significant information and discards the rest

Folding versus unfolding

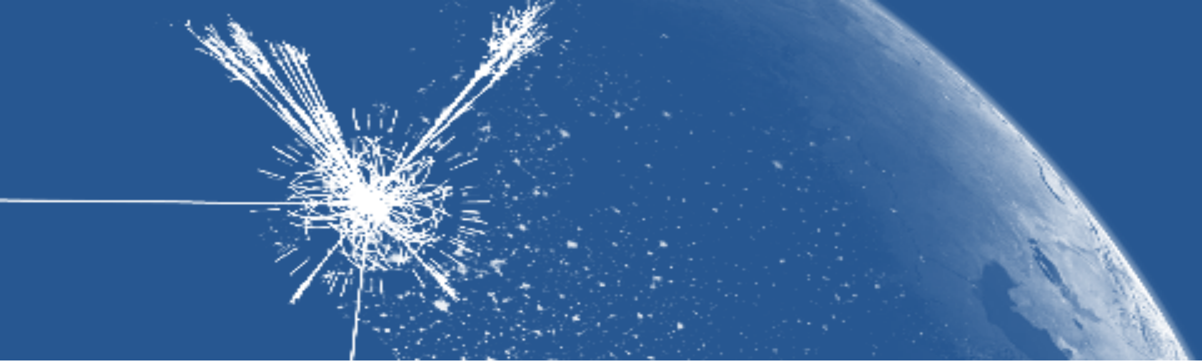
Unfolding is an ill-defined problem which necessarily leads to some obstruction of information in the data and transfer of statistical uncertainty to a systematic one after regularisation (this is similar to a non-parametric fit to data)

Technically simpler and mathematically well defined is the folding of a theoretical prediction $\boldsymbol{x}^{\text{theo}}(\boldsymbol{\theta})$, depending on a set of parameters $\boldsymbol{\theta}$, through the detector response and direct comparison with the measured data. It allows the statistical test:

$$\chi^2(\boldsymbol{x}^{\text{theo}}(\theta)) = \left(\boldsymbol{A}^{\text{MC}} \cdot \boldsymbol{x}^{\text{theo}}(\theta) - \boldsymbol{y}^{\text{data}}\right)^T \left(\boldsymbol{A}^{\text{MC}} \cdot \boldsymbol{x}^{\text{theo}}(\theta) - \boldsymbol{y}^{\text{data}}\right)$$

Folding requires that the experiments either perform the test, or publish $\boldsymbol{A}^{\text{MC}}$ and $\boldsymbol{y}^{\text{data}}$

Folding does not allow a model-independent combination or comparison among experiments. In most case, unfolding is the only viable solution for easy and long-term use of the experimental results.



Summary for today

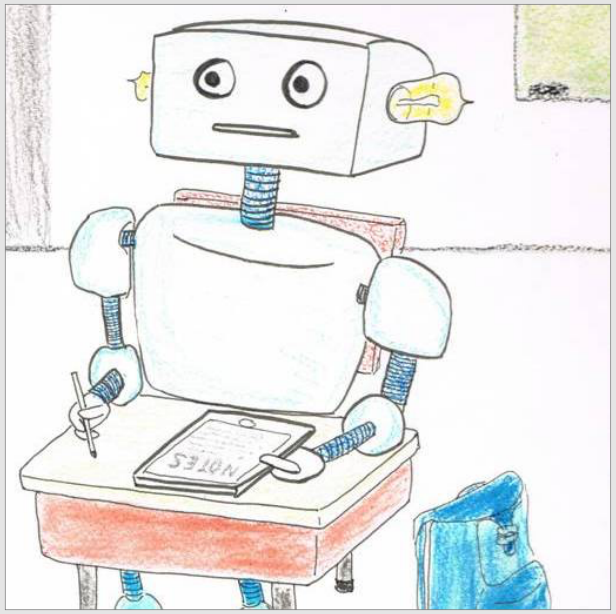
Bootstrapping methods allow to straightforwardly re-sample measured data for the purpose of error propagation

Brief introduction to Monte Carlo integration and the sampling of random data according to any arbitrary PDF

Markov-Chain Monte Carlo integration is a very effective method that “automatically” samples the important regions (where the PDF is large) more often than tails. Try yourself!

Unfolding is a delicate mathematical operation that requires careful regularisation. Folding can help in some cases.

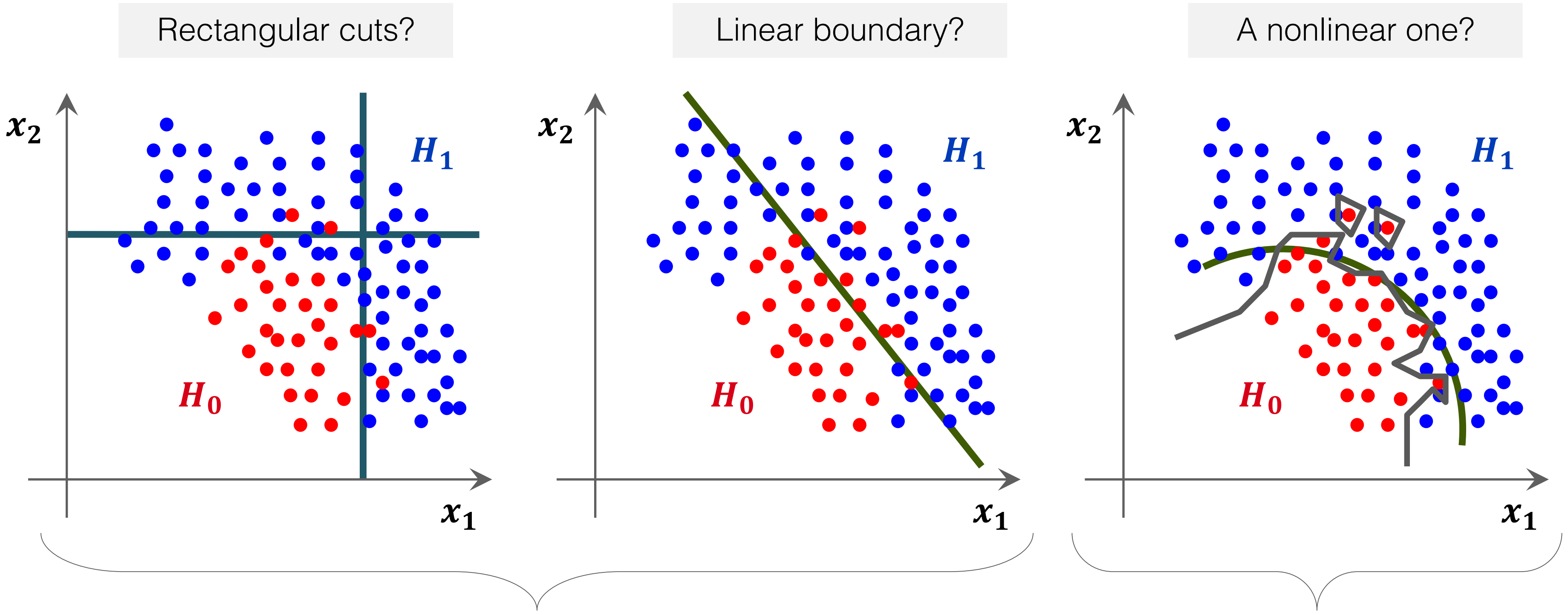
Multivariate techniques and machine learning



Event classification

Suppose data sample with two types of events: H_0, H_1

- We have found discriminating input variables $x_1, x_2, \ldots$
- What decision boundary should we use to select events of type H_1?



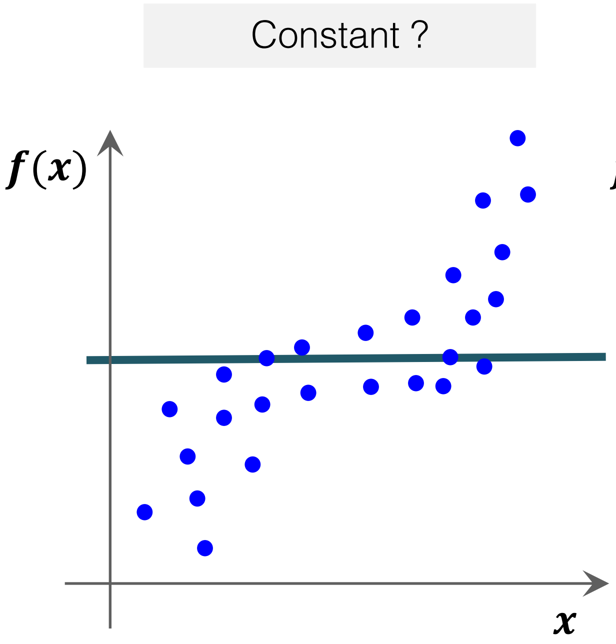
Low variance (stable), high bias methods

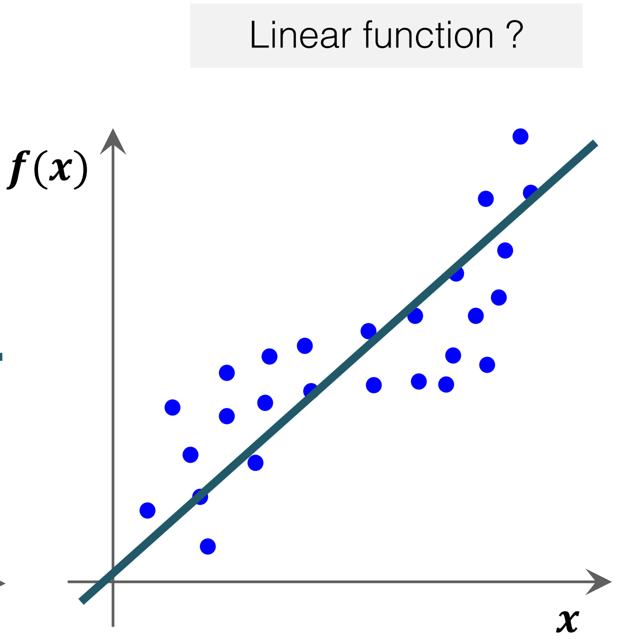
High variance, small bias methods

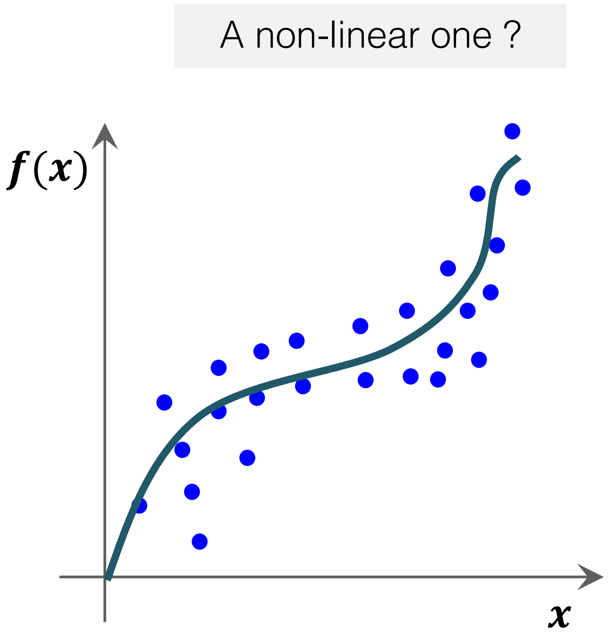
Parameter regression

How to estimate a *functional behaviour* from a set of measurements? HEP examples:

- Energy deposit in a the calorimeter, distance between overlapping photons, …
- Entry location of a particle in the calorimeter or on a silicon pad, …







Looks trivial? What if we have many input variables?

Note: the goal is *not to fit* given data, but to learn $\boldsymbol{f(x)}$ vs. $\boldsymbol{x}$ to *predict* ***target*** $\boldsymbol{f(x)}$ for new measurements $\boldsymbol{x}$

These are the simplest applications of statistical machine learning (ML). Most particle physics utilisations so far fall into this category

However, there is no limit of use cases for complex machine learning…

Image recognition

Table tennis (KUKA advertisement)

Autonomous driving (Google)

+ speech recognition, language understanding, syntax parsing, translation, face recognition, road hazard detection, …
and: **PHYSICS !**

…as long as the ML algorithms are smart and efficient enough, there is sufficient computing power, and a complete set of training data

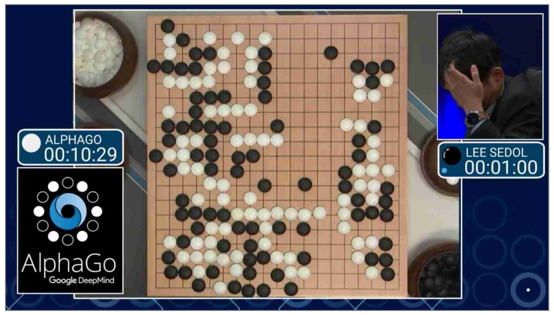
Also in particle physics we can apply ML to more complex problems. Among these: track reconstruction, calibration, tuning

Machine learning is giving computers the ability to learn without explicitly programming them (Arthur Samuel, 1959)

It is fundamentally different from applying a set of fixed rules (a “program”) to solve a problem

Chess computers in the 1980’s and 90’s

Alpha Go in 2016

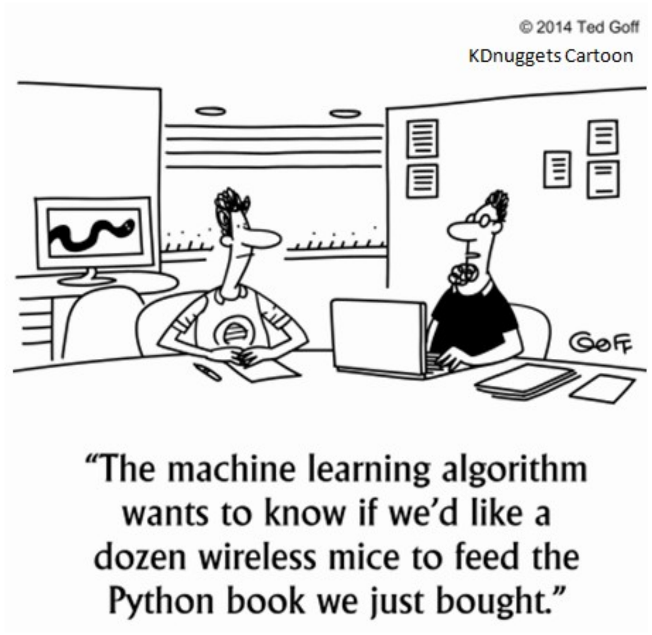


Rule-based programming

Machine learning

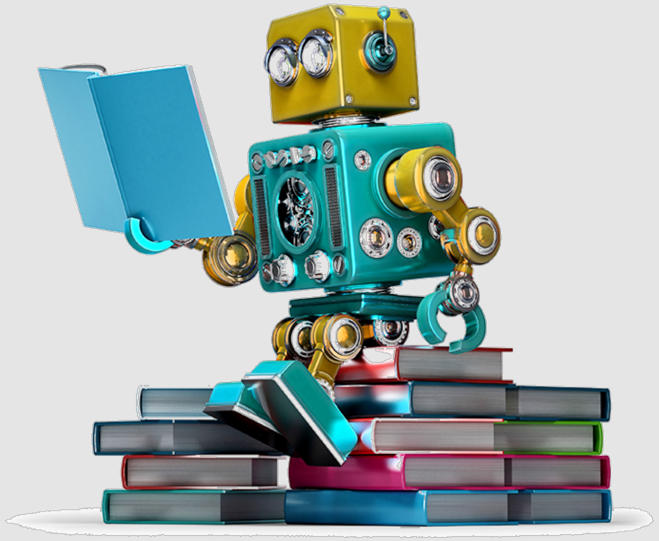
(Deep reinforcement learning)

Not so long ago, real-life artificial intelligence used to be like this:


© 2014 Ted Goff
KDnuggets Cartoon
GOFF
"The machine learning algorithm wants to know if we'd like a dozen wireless mice to feed the Python book we just bought."

Such things still happen, but the improvements have nevertheless been astounding

More tomorrow…



Picture from: https://redshift.autodesk.com/machine-learning/