The RAVE Vertex Reconstruction Toolkit: status and new applications

Wolfgang Waltenberger, **Fabian Moser**
and **Winfried Mitaroff**
with contributions by **Bernhard Pflugfelder** and **Herbert V. Riedel**

Austrian Academy of Sciences, Institute of High Energy Physics
Nikolsdorfer Gasse 18, A-1050 Vienna, Austria, Europe

Outline:

- The RAVE toolkit and VERTIGO framework in short;
- Improved track reconstruction by RAVE's smoother;
- Progress on the implementation of kinematics fitting;
- Summary: present status and outlook to the future.

Reminder of the main goals

1. Creation of an extensible, detector-independent toolkit (RAVE) for vertex reconstruction, to be embedded into various environments:

- *RAVE = "Reconstruction (of vertices) in abstract versatile environments"*;
- The toolkit includes both vertex finding (a pattern recognition task a.k.a. track bundling) and vertex fitting (estimation of the vertex parameters and errors);
- Synergy used: starting point was the old CMS software (ORCA) which has recently been refactored and ported to the new CMS framework (CMSSW);
- Principal algorithmic assets are robustified reconstruction methods with estimators based on adaptive filters, thus downweighting the influence of outlier tracks;
- RAVE is foreseen to stay source-code compatible with CMSSW, but thanks to its generic API may easily be embedded into other software environments.

2. Creation of a simple stand-alone framework (VERTIGO) for fast implementation, debugging and analyzing of the core algorithms:

- *VERTIGO = "Vertex reconstruction tools and interfaces to generic objets"*;
- Framework tools available: Visialization, Histogramming, a Vertex Gun for generating artificial test events, an LCIO input event generator, and a Data Harvester & Seeder for flexible I/O.
- VERTIGO is able to emulate various detector setups by a flexible "skin" concept.

Example of using the RAVE toolkit

Calling RAVE from a C++ environment:

Easiest use case:

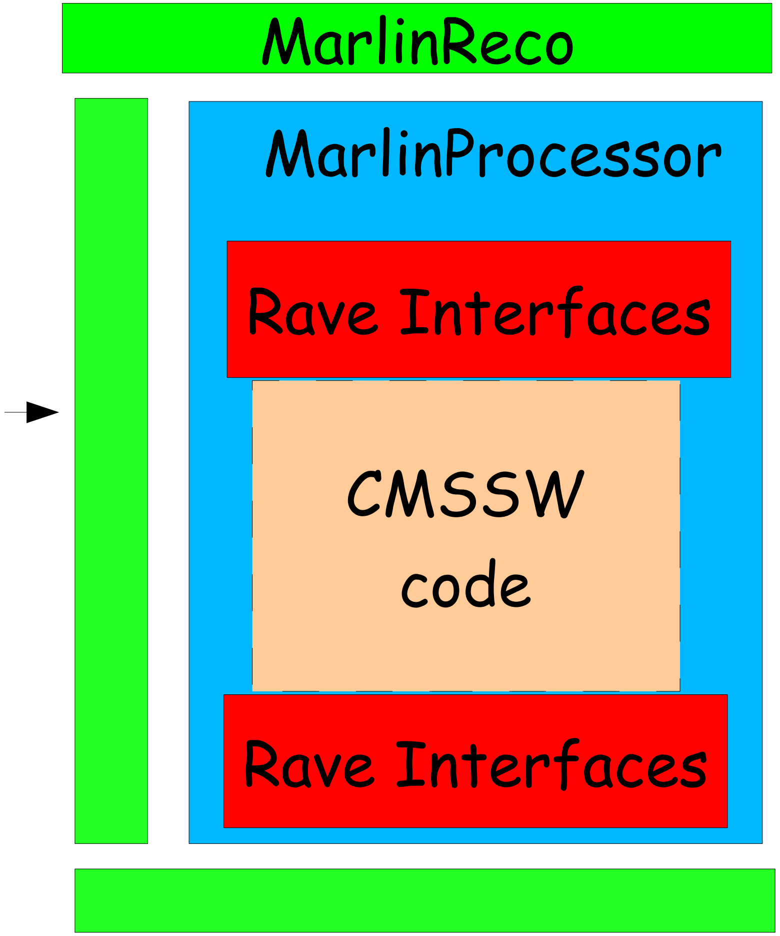
User must implement interfaces

```
rave::Factory factory (MyMagneticField(), MyPropagator() );

std::vector < rave::Vertex > vertices = factory.create( tracks,
"method" );
```

Factory creates vertices from tracks, using method “method”

Note: the factory class name has recently been renamed rave::VertexFactory.



MarlinReco

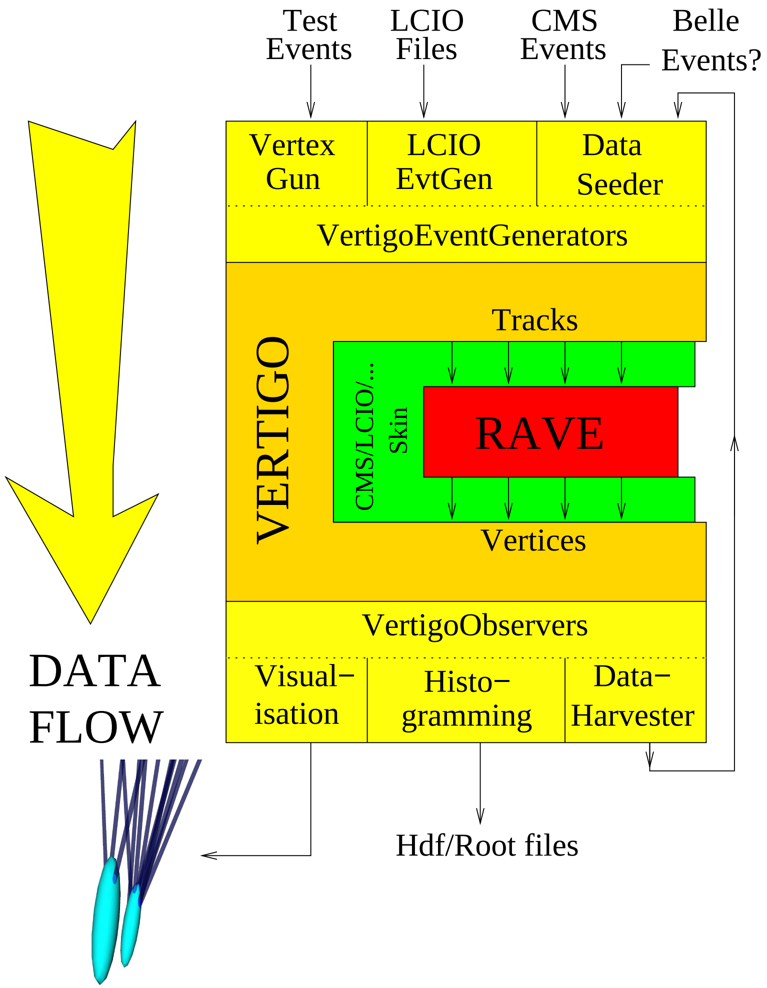
MarlinProcessor

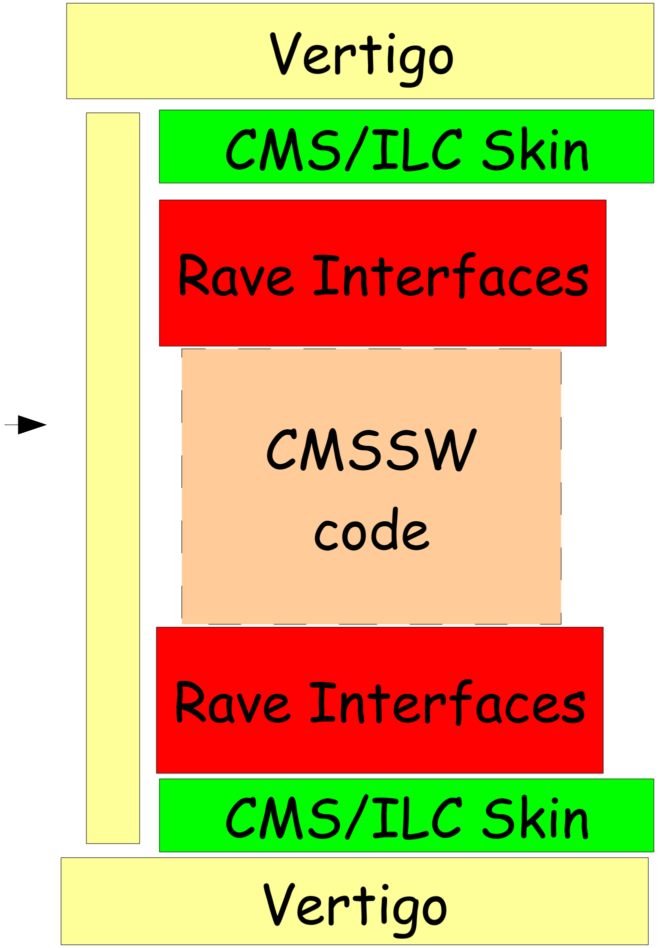
Rave Interfaces

CMSSW code

Rave Interfaces

Functionality of the VERTIGO framework





What is a full geometric vertex fit ?

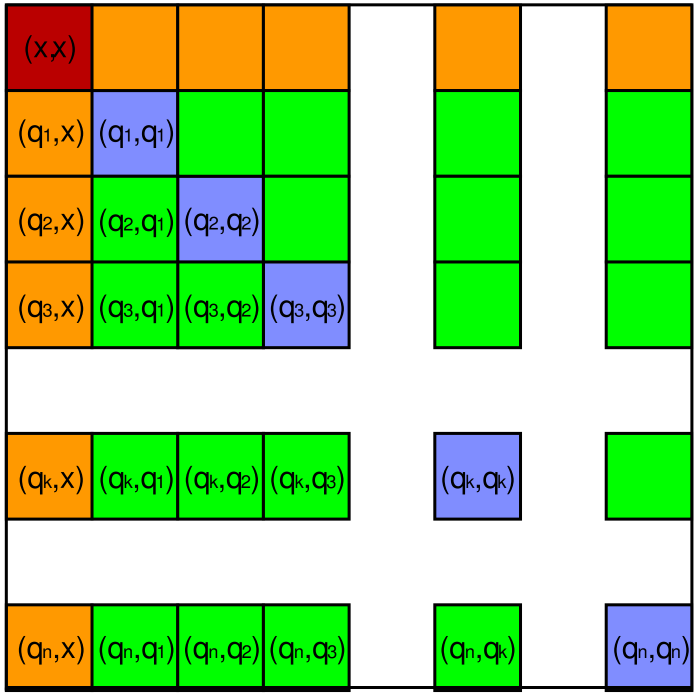
Input to a geometric vertex fit:

- Fitted tracks: $\mathbf{p}_i$, $\mathrm{cov}(\mathbf{p}_i, \mathbf{p}_i)$, $i = 1 \ldots n$
  - $\mathbf{p}_i$ = 5-vectors of track parameters,
  - $\mathrm{cov}(\mathbf{p}_i, \mathbf{p}_i)$ = symmetric 5×5 matrices.
- Beam int. profile (optional): $\mathbf{v}$, $\mathrm{cov}(\mathbf{v}, \mathbf{v})$
  - $\mathbf{v}$ = 3-vector of centre of the beam i.p.,
  - $\mathrm{cov}(\mathbf{v}, \mathbf{v})$ = symm. or diag. 3×3 matrix.

Results of a geometric vertex fit:

- Vertex position: $\mathbf{x}$, $\mathrm{cov}(\mathbf{x}, \mathbf{x})$
  - $\mathbf{x}$ = 3-vector of space coordinates,
  - $\mathrm{cov}(\mathbf{x}, \mathbf{x})$ = symmetric 3×3 matrix.
- Tracks at vertex: $\mathbf{q}_i$, $\mathrm{cov}(\mathbf{q}_i, \mathbf{q}_i)$, $i = 1 \ldots n$
  - $\mathbf{q}_i$ = 3-vectors of re-fitted track parameters,
  - $\mathrm{cov}(\mathbf{q}_i, \mathbf{q}_i)$ = symmetric 3×3 matrices

  (obtained by the **Kalman "smoother"**).

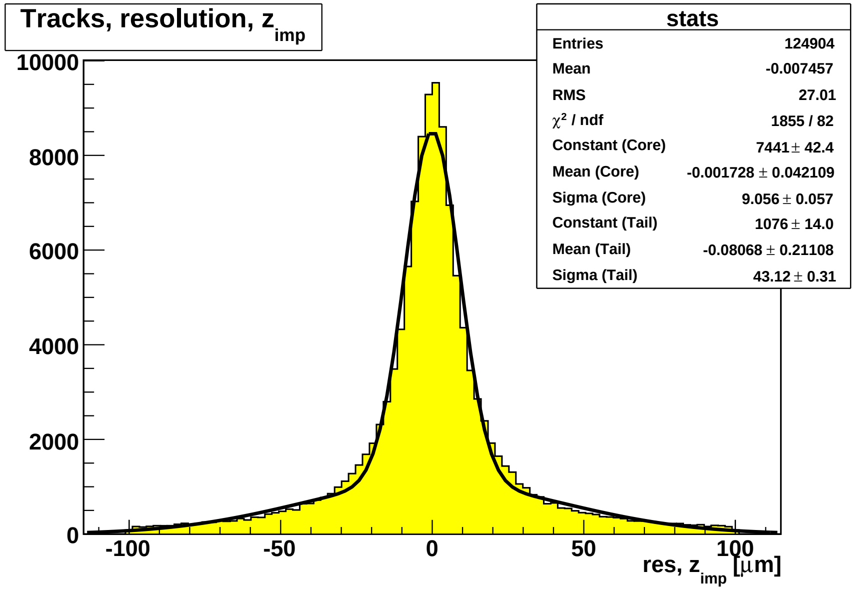


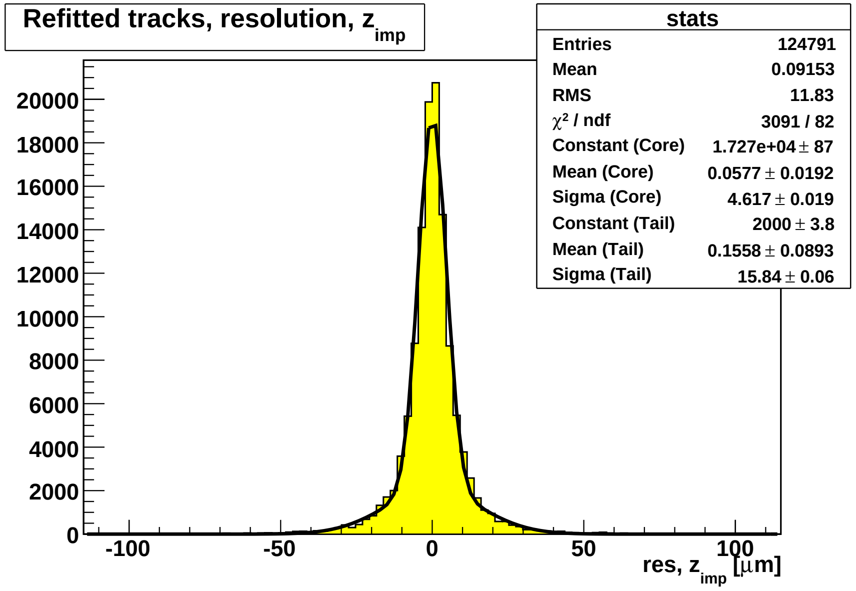
(x,x)	🟧	🟧	🟧	🟧	🟧
(q1,x)	(q1,q1)	🟩	🟩	🟩	🟩
(q2,x)	(q2,q1)	(q2,q2)	🟩	🟩	🟩
(q3,x)	(q3,q1)	(q3,q2)	(q3,q3)	🟩	🟩
(qk,x)	(qk,q1)	(qk,q2)	(qk,q3)	(qk,qk)	🟩
(qn,x)	(qn,q1)	(qn,q2)	(qn,q3)	(qn,qk)	(qn,qn)

- But this is not the full information from a vertex fitter; there are more covariances to deal with:
  - $\mathrm{cov}(\mathbf{q}_i, \mathbf{x})$, $\quad i = 1 \ldots n$ — n asymmetric 3×3 matrices,
  - $\mathrm{cov}(\mathbf{q}_i, \mathbf{q}_j)$, $i, j = 1 \ldots n$ with $i \neq j$ — $n \cdot (n-1)$ asymmetric 3×3 matrices.

In practice, the $\mathrm{cov}(\mathbf{q}_i, \mathbf{q}_j)$, $i \neq j$ are only needed in case of a subsequent vertex re-fit with kinematic constraints. The n asymmetric $\mathrm{cov}(\mathbf{q}_i, \mathbf{x})$, however, should not be forgotten for persistency (e.g. `LCIO`).

Tracks re-fitted by the smoother: z_{imp}





- Track parameters in "perigee representation" (δ_T, Δz, φ, . . .) w.r.t. an arbitrarily defined pivot point;
- histogram of $z_{imp} \equiv \Delta z^{(fitted)} - \Delta z^{(true)}$ of reconstructed tracks (left), and re-fitted tracks (right);
- no use was made of the beam interaction profile.

Kinematics fitting by RAVE

Algorithm used:

- The algorithm is based on a least squares fit with Lagrangian multipliers for the kinematic constraints;
- Flexibility is achieved by separating the kinematics fitting proper from the "decay chain tree" steering;
- For details, see *K. Prokofiev*, Proc. CHEP '04, Interlaken (CH): CERN-2005-002 and `http://indico.cern.ch/getFile.py/access?contribId=75&sessionId=4&resId=1&materialId=paper&confId=0`

Implementation:

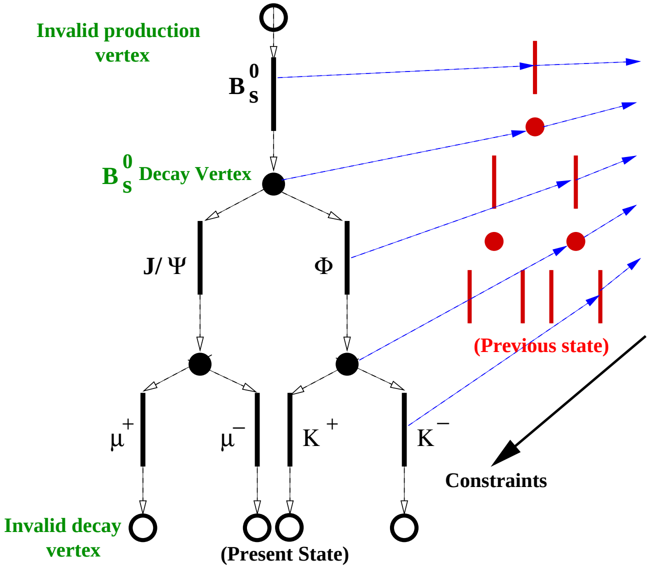
- Work has started on 1 October: the core code has been extracted from the CMS framework CMSSW;
- The wrapper code for interfacing with the RAVE toolkit is finished, and all has successfully compiled;
- Calling it from a C++ environment proceeds via the new factory class rave::KinematicTreeFactory (both building the tree, and performing the fit);
- Requires assignment of particle masses to tracks, and definition of the kinematic constraints.

Tests in limbo:

- Functional tests of the new code will be performed using the VERTIGO stand-alone framework;
- Before starting, new VERTIGO tools must be implemented in order to support the kinematics:
  - a vertex gun generating kinematically correct events;
  - observers able to handle the kinematics information.

Decay tree and kinematics fit

KinematicTree



The kinematic fit: LMS minimization

- χ^2 minimization with the set of additional constraints $H(y_{ref})=0$, linearized (first order Taylor expansion) around some given point $y_{\exp}$

$$\frac{\partial H(y_{\exp})}{\partial y}(y-y_{\exp})+H(y_{\exp})=D\,\delta y+d=0$$

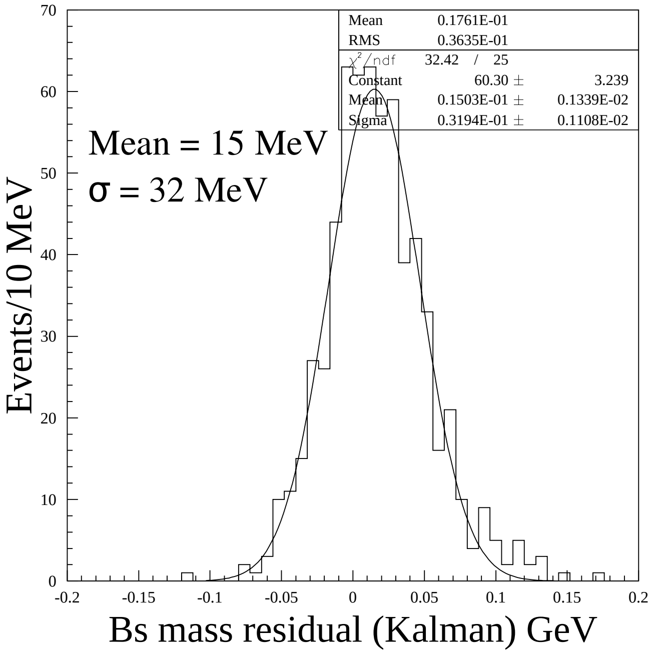
  - D: matrix of derivatives, one line per constraint equation *(n_equations x n_parameters)*
  - d: vector of values of constraints
- Function to minimize with respect to (y_{ref},λ):

$$\chi^2=(y^{ref}-y)V_y^{-1}(y^{ref}-y)^T+2\lambda^T(D\,\delta y+d)\rightarrow min$$

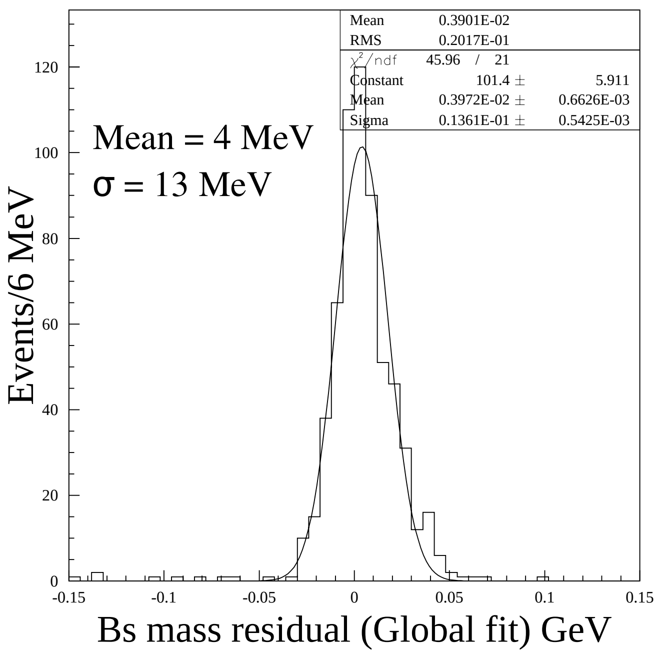
Decay tree example for $B_s^0 \rightarrow J/\psi\Phi \rightarrow \mu^+\mu^-K^+K^-$.

Example: $B_s^0 \to J/\psi\Phi$ (simulation CMS)

Residual of the $\mu^+\mu^- K^+ K^-$ 4-track invariant mass with and without constraints:



Without constraints $M_{B_s}^{rec} - M_{B_s}^{PDG}$



With constraints $M_{B_s}^{rec} - M_{B_s}^{PDG}$

CHEP'04, Interlaken, Switzerland, 30 September 2004

K.Prokofiev, Th.Speer Universität Zürich

Summary: status and outlook

Present status:

- Code written in C++; RAVE depends on CLHEP and Boost, VERTIGO's visualization also on Coin3d;
- Tested with Linux (Debian and SLC4), Mac OSX and CygWin/DLL, on Intel and PPC hardware;
- RAVE may be called directly from C++, and with a wrapper (SWIG) also from Java and Python;
- Adaptor classes ("glue code") exist for embedding RAVE into the reconstruction frameworks Marlin (C++) and org.lcsim (Java); tested with LDC and SiD simulation data;
- Simple VERTIGO skins exist for the LDC and SiD detectors; tested with LCIO formatted input data.

Work in progress:

- Interfacing RAVE with the ZvTop/ZvRes "topological vertex search" algorithm is almost finished;
- Augmenting RAVE with "heavy flavour tagging" algorithms is in progress (see talk at LCWS '07);
- Augmenting RAVE with a powerful kinematics fitting algorithm has started (see the previous slides).

Availability:

- A beta release is available; our institute is committed to maintenance, documentation and distribution;
- `http://projects.hepforge.org/rave/` (HepForge repository for RAVE source code, docus, etc);
- `http://stop.itp.tuwien.ac.at/websvn/` (repository for Marlin & org.lcsim glue code, and VERTIGO).

The Vertexing Circle’s credo

“The method of Least Squares is seen to be our best course when we have thrown overboard a certain portion of our data -- a sort of sacrifice which has often to be made by those who sail the stormy seas of Probability.”

F. Y. Edgeworth, 1887

“Let us not throw away data all too hastily. Instead, let us weight and reweight the data, consider and reconsider alternative models. Only if we must, at the latest possible stage, shall we distinguish between 'in' and 'out', between signal and noise.”

The CMS vertexing circle, 2004