

Implementation of Monte Carlo fitter and Selection of $B \rightarrow K^* \mu \mu$ decays in LHCb Run2 data.

Oliver Dahme

University of Zurich

o.dahme@cern.ch

June 19, 2018

Supervisors: Prof. Dr. Nicola Serra, Prof. Dr. Marcin Chrzaszcz

- 1 LHCb Detector
- 2 The $B_0 \rightarrow K^* \mu^+ \mu^-$ decay
- 3 Selection of $B_0 \rightarrow K^* \mu^+ \mu^-$ decays
- 4 Reweighting
- 5 RooMCMarkovChain

The LHCb Detector

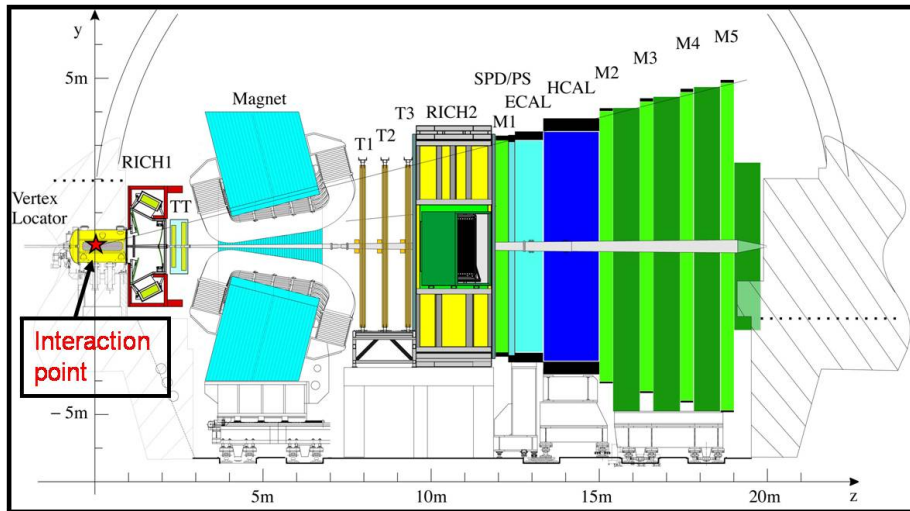


Figure: Detector overview

The $B_0 \rightarrow K^* \mu^+ \mu^-$ decay

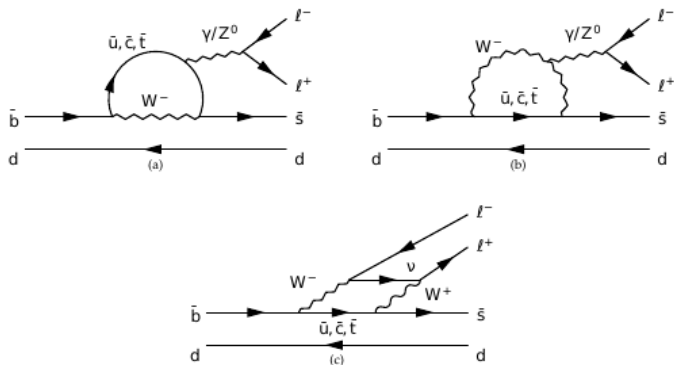


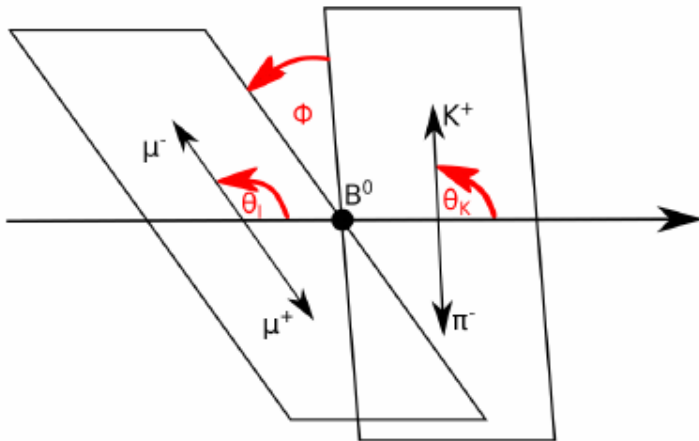
Figure: Feynman diagrams for decay $B(\bar{b}, d) \rightarrow K^*(\bar{s}, d) l^+ l^-$ at lowest order

Recent experimental results of the LHCb collaboration [JHEP08 (2017) 055] suggest a violation of the lepton universality:

$$R_{K^*0} = \frac{\mathcal{B}(B^0 \rightarrow K^{*0} \mu^+ \mu^-)}{\mathcal{B}(B^0 \rightarrow K^{*0} J/\psi(\rightarrow \mu^+ \mu^-))} \bigg/ \frac{\mathcal{B}(B^0 \rightarrow K^{*0} e^+ e^-)}{\mathcal{B}(B^0 \rightarrow K^{*0} J/\psi(\rightarrow e^+ e^-))} ,$$
$$R_{K^*0} = \begin{cases} 0.66_{-0.07}^{+0.11}(\text{stat}) \pm 0.03(\text{syst}) & \text{for } 0.045 < q^2 < 1.1 \text{ GeV}^2/c^4, \\ 0.69_{-0.07}^{+0.11}(\text{stat}) \pm 0.05(\text{syst}) & \text{for } 1.1 < q^2 < 6.0 \text{ GeV}^2/c^4. \end{cases} \quad (1)$$

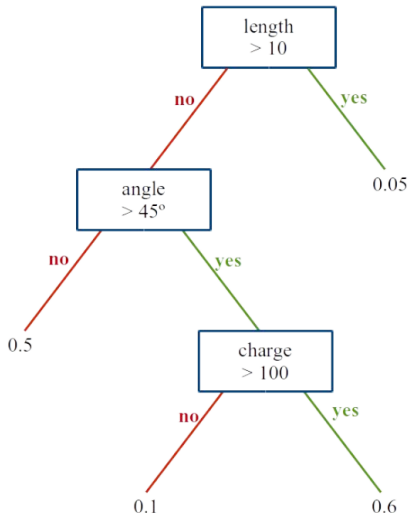
Variables defining the decay

The kinematics and directions of the final state particles from the decay are defined by the three angles θ_K , θ_L and ϕ and the invariant di-muon mass square q^2 .



Decision tree

Decision trees are structured in Nodes, which represent the input features and in Leaves, which can be seen as terminal nodes.



Selection of $B_0 \rightarrow K^* \mu^+ \mu^-$ decays

To select the decay from the raw Run2 data, boost decision trees and the K-folding technique were used. To find the best boost decision tree algorithm, the following were tested:

- Ada Boost
- uGB + knnAda (k-nearest neighbor AdaBoost)
- uBoost
- uGB + FI (flatness loss)
- xgb
- sk_bdtg
- sk_bdt

Selection of $B_0 \rightarrow K^* \mu^+ \mu^-$ decays

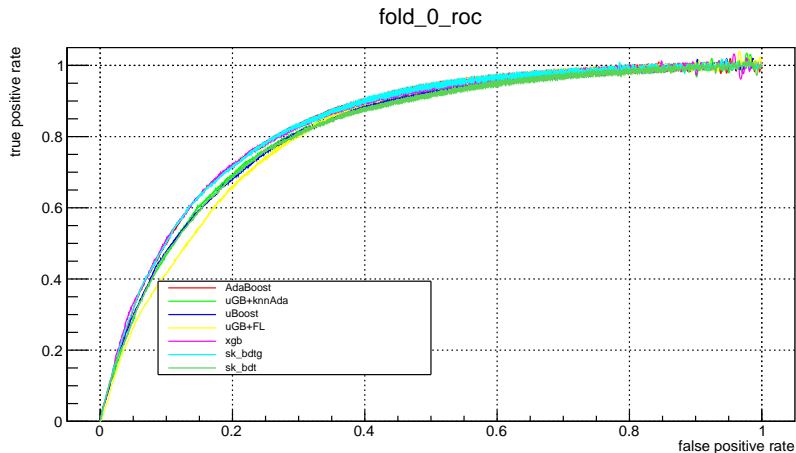


Figure: ROC (receiver operating charactertic) curve

Selection of $B_0 \rightarrow K^* \mu^+ \mu^-$ decays

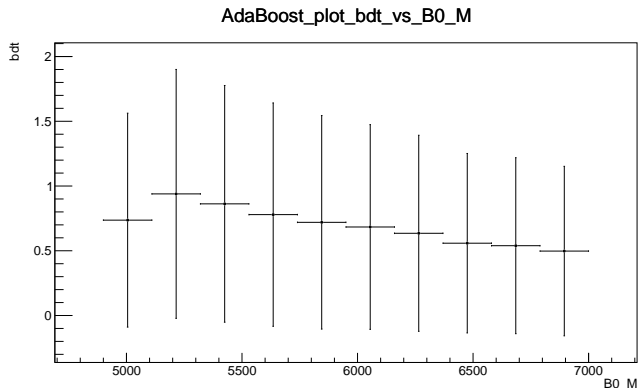


Figure: Correlation plot between B mass in Mev and the response of the AdaBoost algorithm.

Selection of $B_0 \rightarrow K^* \mu^+ \mu^-$ decays

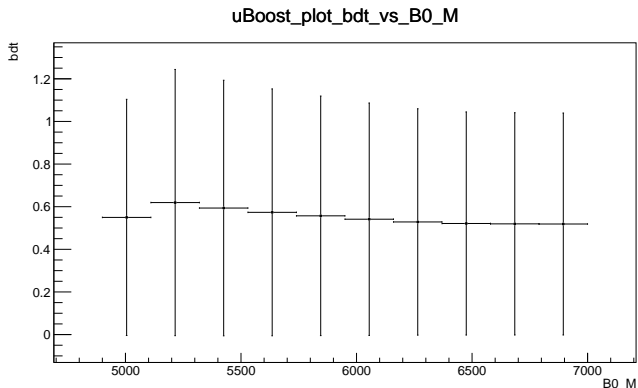


Figure: Correlation plot between B mass in Mev and the response of the uBoost algorithm.

- procedure to match Monte Carlo simulation to data
- get detector efficiencies
- done by applying weights to the MC events
- chosen by comparing MC and data

compare parameters:

- number of Tracks
- the quality of the $K\pi\mu\mu$ vertex
- transversal momentum of the B meson

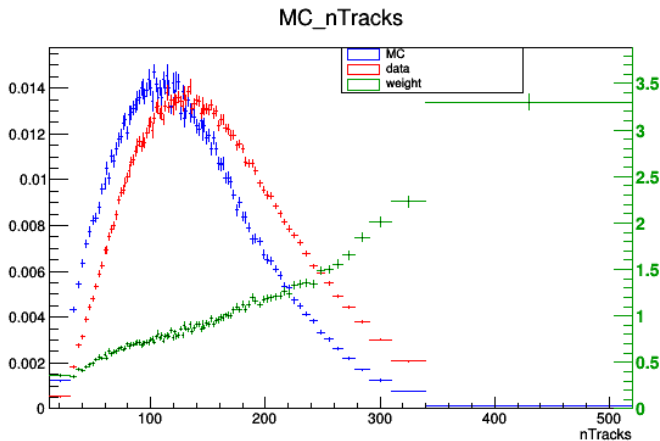


Figure: Reweighting result for the number of tracks

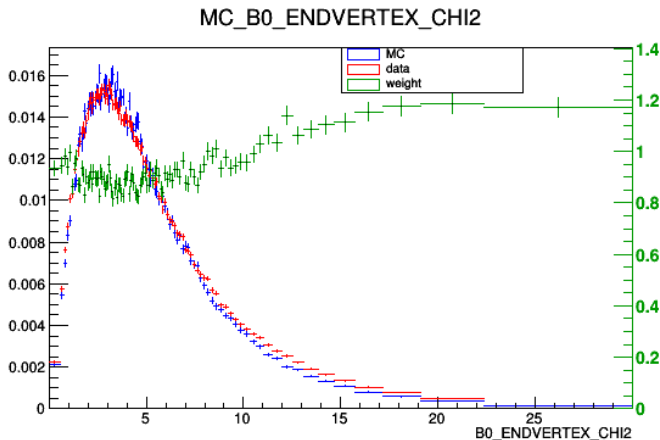


Figure: Reweighting result for the quality of the $K\pi\mu\mu$ vertex (B0_ENDVERTEX_CHI2)

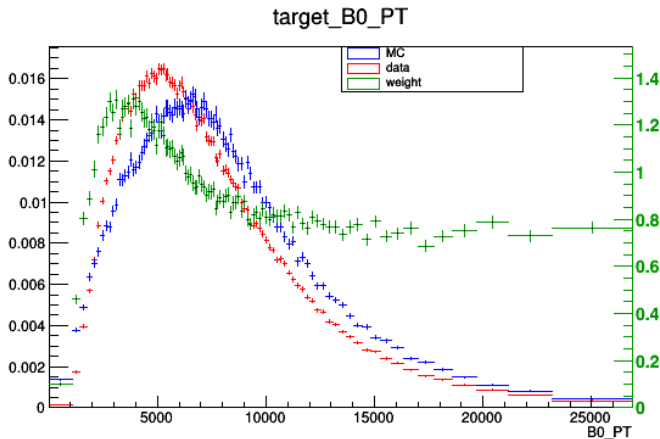


Figure: Reweighting result for the transversal momentum of the B meson ($B0_PT$) in MeV.

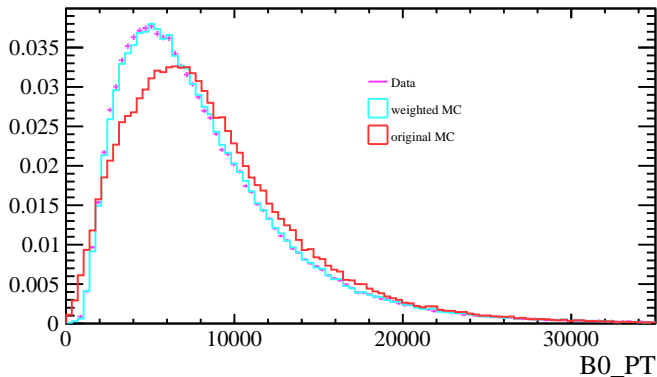


Figure: Result after applying the weights for the transversal momentum of the B meson (B_0_PT) in MeV.

Monte Carlo Markov Chain

A Markov Chain is a random process which undergoes several states. From each state there is a probability distribution to change into another state or to stay.

Most important is the assumption that every next step just depends on the current state.

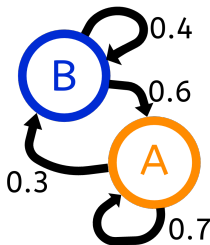


Figure: Example of states and probabilities of states in a Markov Chain

- Current state x is proposed to move to y
- Calculate Hastings ratio:

$$r(x, y) = \frac{h(y) \cdot q(y, x)}{h(x) \cdot q(x, y)} \quad (2)$$

- Where $q(x, \cdot)$ is the conditional probability density and h is the unnormalized density of the specified distribution
- Accept the proposed move to y with the probability:

$$a(x, y) = \min(1, r(x, y)). \quad (3)$$

The algorithm

The robust adaptive Metropolis process is defined recursively through

- (R1) compute $Y_n := X_{n-1} + S_{n-1}U_n$, where $U_n \sim q$ is an independent random vector,
- (R2) with probability $\alpha_n := \min\{1, \pi(Y_n)/\pi(X_{n-1})\}$ the proposal is accepted, and $X_n := Y_n$; otherwise the proposal is rejected and $X_n := X_{n-1}$, and
- (R3) compute the lower-diagonal matrix S_n with positive diagonal elements satisfying the equation

$$(1) \quad S_n S_n^T = S_{n-1} \left(I + \eta_n (\alpha_n - \alpha_*) \frac{U_n U_n^T}{\|U_n\|^2} \right) S_{n-1}^T$$

where $I \in \mathbb{R}^{d \times d}$ stands for the identity matrix.

Figure: [arXiv: 1011.4381v2](#)

Here Y_n is the next state and X_{n-1} is the current state.

The S -matrix determines the direction and step size of the next step.

Example

fitting the following pdf:

$$\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + f \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} \quad (4)$$

Fit of a double gauss

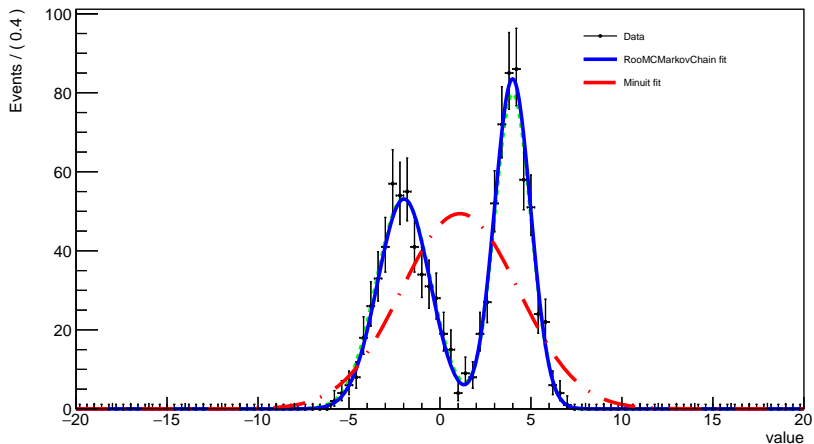


Figure: Fit of a double gauss

- The constructor behaves like the RooMinuit constructor
`RooMCMC(RooAbsReal *negativeloglikelihood)`
- then mcmc performs the walk and error calculation
- `RooMCMkovchain.mcmc(int npoints, int cutoff, string errorstrategy)`
- there are two errorstrategies: "gaus" for symmetric errors and "interval" for asymmetric ones
- The terminal output is similar to the one of Minuit. It first prints the parameters with errors and then the correlation coefficients.

Features

- To look at the profile of the nll one can use:
- TGraph *profile = **getProfile**(string name, bool cutoff)
- with cutoff bool one can include or exclude the cutoff points

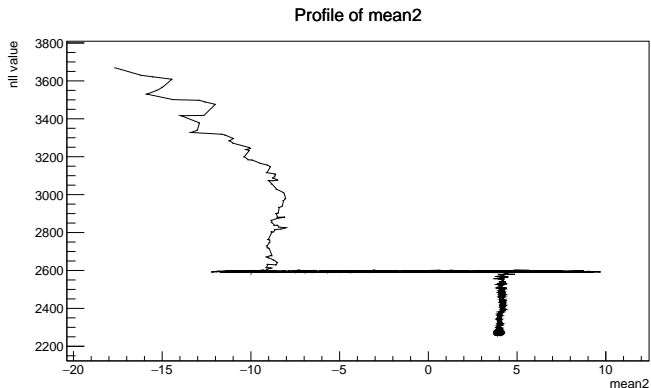


Figure: Profile from the mean of the second gauss

Features

- It is very important to look at the walk distribution, to check if the cutoff is well placed:
- `TMultiGraph *walkdis = getWalkDis(string name, bool cutoff)`

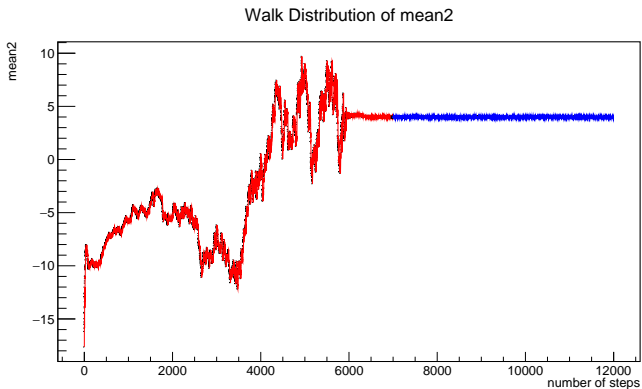


Figure: Walk distribution from the mean of the second gaus

Features

- It is also possible to get a histogram of the walk distribution, to check if the errors are symmetric or asymmetric.
- TH1F *walkdishis = **getWalkDisHis**(string name, int xbins, bool cutoff)

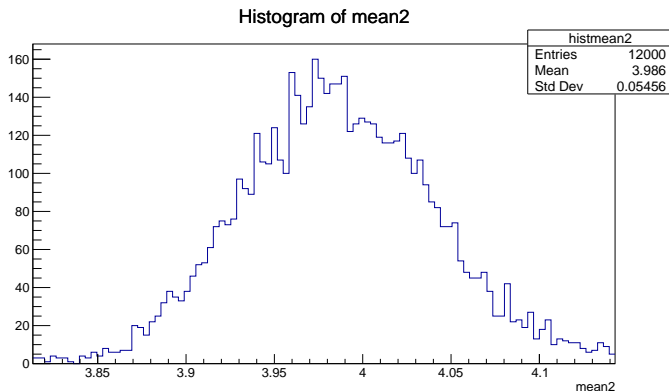


Figure: Histogram from the walk distribution of the mean of the second gauss

Features

- To check for correlations between parameters one can create a cornerplot between them.
- TH2D *corner = **getCornerPlot**(string name1, string name2, int nbinsx, int nbinsy, bool cutoff)

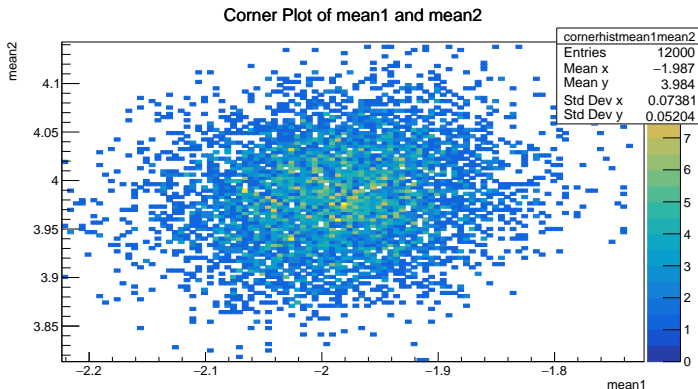


Figure: Cornerplot between the mean values of the two gauss.

- Now to put everything together:
- **saveCornerPlotAs(string picname)**
- It saves a picture with a histogram of each parameter plus a correlation plot with each parameter pair.
- One can see directly if there is any correlation and if errors are symmetric or asymmetric

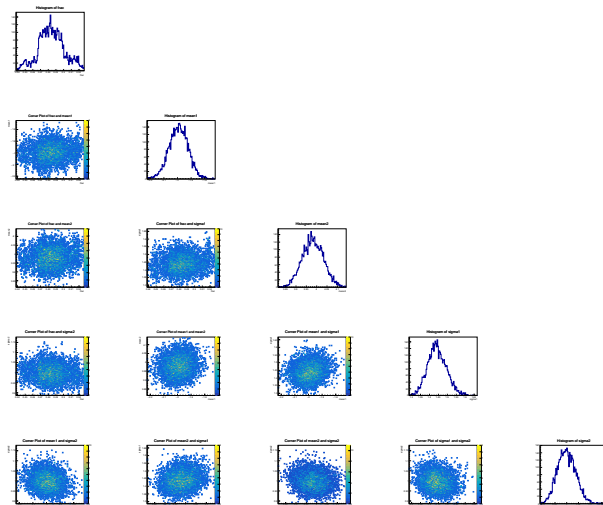


Figure: Corner Plot of the double gauss fit

Pull Request: <https://github.com/root-project/root/pull/1422>

E-mail: o.dahme@cern.ch

The RooMCMakovChain fitting routine is currently beeing added to the ROOT Data analysis framework.

The best algorithm uBoost from the boost decision tree test will be used to find future $B \rightarrow K^* \mu \mu$ decays.

The rewighting done in this thesis will be used for the future analysis of the $B \rightarrow K^* \mu \mu$ in the current LHCb data.