

# Custom Orthogonal Weight functions (COWS)

An improved event weighting procedure for removing background from signal

[Matthew Kenzie](#)

*University of Warwick*

**Warwick HEP Seminar**

5<sup>th</sup> May 2022



Moooo!

- ▶ I will predominantly be discussing the findings of [arXiv:2112.04574] (Dembinski, MK, Langenbruch, Schmelling) [1]

## Custom Orthogonal Weight functions (COWs) for Event Classification

Hans Dembinski<sup>1</sup>  
*TU Dortmund, Germany*

Matthew Kenzie<sup>1</sup>  
*University of Warwick, United Kingdom*

Christoph Langenbruch<sup>2</sup>  
*RWTH Aachen, Germany*

Michael Schmelling<sup>3</sup>  
*Max Planck Institute for Nuclear Physics, Heidelberg, Germany*  
(Dated: January 27, 2022)

A common problem in data analysis is the separation of signal and background. We revisit and generalise the so-called *sWeights* method, which allows one to calculate an empirical estimate of the signal density of a control variable using a fit of a mixed signal and background model to a discriminating variable. We show that *sWeights* are a special case of a larger class of Custom Orthogonal Weight functions (COWs), which can be applied to a more general class of problems in which the discriminating and control variables are not necessarily independent and still achieve close to optimal performance. We also investigate the properties of parameters estimated from fits of statistical models to *sWeights* and provide closed formulas for the asymptotic covariance matrix of the fitted parameters. To illustrate our findings, we discuss several practical applications of these techniques.

- ▶ Born from discussions in the LHCb Statistics Group and the innovations of Schmelling [2]
- ▶ A fresh look at the *sPlot* paper - [NIM A 555 (2005) 356] (Pivk and Le Diberder) [3]
- ▶ The concept was first mentioned by Barlow - [J. Comp. Phys. 72 (1987) 202] [4]
- ▶ Important related developments on parameter estimates when fitting (s)weighted data by Langenbruch - [arXiv:1911.01303] [5]

# The bet

Perhaps the *most enjoyable part* of this entire enterprise



Matthew William Kenzie 3:58 PM

i bet you i can get the word "cow" into a paper 5 times



Patrick Haworth Owen 4:00 PM

whats the bet



Matthew William Kenzie 4:00 PM

dunno



Patrick Haworth Owen 4:00 PM

dinner at meyrinoise

fully inclusive

as many limonchellos as you like

# The bet

Perhaps the *most enjoyable part* of this entire enterprise



Matthew William Kenzie 3:58 PM

i bet you i can get the word "cow" into a paper 5 times



Patrick Haworth Owen 4:00 PM

whats the bet



Matthew William Kenzie 4:00 PM

dunno



Patrick Haworth Owen 4:00 PM

dinner at meyrinoise

fully inclusive

as many limonchellos as you like



Matthew William Kenzie 4:03 PM

is there are a bonus for getting it in the abstract?



Patrick Haworth Owen 4:04 PM


yes

then you get dinnner at lavation

if you can get it in the title cafe de paris



# The bet

 **Matthew William Kenzie** 4:05 PM  
do they stack?

 **Patrick Haworth Owen** 4:05 PM  
yes  
they stack

 **Matthew William Kenzie** 11:13 AM  
shall we shake on it?



 **Patrick Haworth Owen** 11:19 AM  


# The bet



Matthew William Kenzie 4:05 PM

do they stack?



Patrick Haworth Owen 4:05 PM

yes

they stack

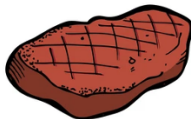
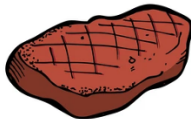
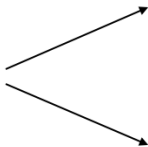
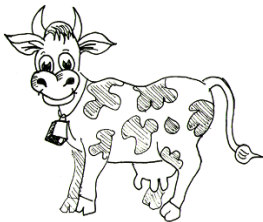


Matthew William Kenzie 11:13 AM

shall we shake on it?



Patrick Haworth Owen 11:19 AM



## Corresponding software tools

- ▶ The developments in the paper [1] have corresponding software implementations
- ▶ The python `sweights` package (`readthedocs`) has implementations of:
  - ▶ `sWeights`: in each of the “Variants” discussed below
  - ▶ `COWs`: in any of the scenarios discussed below
  - ▶ `Sandwich Estimator`: the full (and approximate) covariance correction when fitting (s)weighted samples
- ▶ Also provide a wrapper for p.d.f.s defined in `RooFit`
- ▶ The  $\mathcal{U}$ -statistic permutation (USP) test is implemented in the python `resample` package
- ▶ Implementation of the sandwich estimator in `iminuit` is a WIP

**sweights**

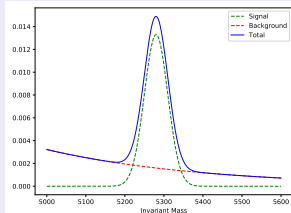
*i*minuit

resample

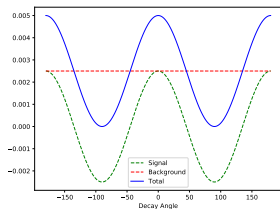
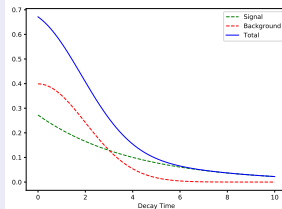
# Setting up the problem

- ▶ In particle physics we often want to extract some properties of an observed signal
- ▶ But we typically have a non-negligible background contribution, usually distinguished using invariant mass
- ▶ The properties we want to extract are in some other dimension
  - ▶ **Lifetime**: Decay time distribution
  - ▶ **Spin**: Angular distributions
  - ▶ **Amplitudes**: Dalitz distributions
- ▶ Often we don't know (or don't want to have to understand) the background distribution in these other dimensions

## “Discriminant variable”



## “Control variable(s)”





# Setting up the problem

## SO WHAT CHOICES DO WE HAVE?

- ▶ Fit the full nD distribution
  - ▶ Requires a suitable model description for each component in each dimension
- ▶ Sideband subtraction or “slicing”
  - ▶ Not statistically very powerful
  - ▶ Requires discriminant and control variables factorise
- ▶ sWeighting
  - ▶ Might think of this as “per-event” slicing
  - ▶ Requires discriminant and control variables factorise
- ▶ Custom Orthogonal Weight functions (COWs)
  - ▶ Might think of this as “per-event” slicing
  - ▶ Does not necessarily require factorisation

Part 1: sWeights

## sWeights as orthogonal functions

- ▶ Require signal and background components both factorise in the discriminant and control variables
- ▶ In other words our total p.d.f. has the form

$$f(m, t) = \underbrace{z g_s(m) h_s(t)}_{\text{Signal}} + \underbrace{(1 - z) g_b(m) h_b(t)}_{\text{Background}} \quad (1)$$

- ▶ We then want to find a weight function,  $w_s(m)$ , which when multiplied by  $f(m, t)$  projects out  $h_s(t)$

$$z h_s(t) = \int w_s(m) f(m, t) dm \quad (2)$$

$$= \int w_s(m) [z g_s(m) h_s(t) + (1 - z) g_b(m) h_b(t)] dm \quad (3)$$

$$= z h_s(t) \int w_s(m) g_s(m) dm + (1 - z) h_b(t) \int w_s(m) g_b(m) dm \quad (4)$$

- ▶ Therefore we require

$$\underbrace{\int w_s(m) g_s(m) dm = 1}_{w_s(m) \text{ is normal to } g_s(m)} \quad \text{and} \quad \underbrace{\int w_s(m) g_b(m) dm = 0}_{w_s(m) \text{ is orthogonal to } g_b(m)} \quad (5)$$

## Choosing the orthonormal functions

- ▶ There are infinitely many choices for  $w_s(m)$  but an optimal choice **minimises the variance** over the discriminating p.d.f.  $g(m)$ .
- ▶ This is a constrained optimisation problem solved with **Lagrange multipliers** (calculation is in the back up). The solution (for the signal component) is

$$w_s(m) = \frac{\alpha_s g_s(m) + \alpha_b g_b(m)}{g(m)} \quad (6)$$

Signal weight function

where the constants  $\alpha_s$  and  $\alpha_b$  are obtained by solving

$$\begin{pmatrix} W_{ss} & W_{sb} \\ W_{sb} & W_{bb} \end{pmatrix} \cdot \begin{pmatrix} \alpha_s \\ \alpha_b \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{where } W_{xy} = \int \frac{g_x(m)g_y(m)}{g(m)} dm. \quad (7)$$

- ▶ You can then follow this through for any component and generalise to

$$\underbrace{\begin{pmatrix} W_{ss} & W_{sb} \\ W_{sb} & W_{bb} \end{pmatrix}}_{W_{k\ell}} \cdot \underbrace{\begin{pmatrix} \alpha_s & \beta_s \\ \alpha_b & \beta_b \end{pmatrix}}_{A_{k\ell} = W_{k\ell}^{-1}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \quad (8)$$

## Application to a finite sample

- ▶ The above derivation assumed knowledge of the true p.d.f. to compute the  $W$ -matrix
- ▶ In practise these are unknown and would be replaced by a sample estimate (typically obtained from a fit)
- ▶ The plugin estimate for  $W$  is then simply

$$\hat{W}_{xy} = \int \frac{\hat{g}_x(m)\hat{g}_y(m)}{\hat{g}(m)} dm \quad (9)$$

sWeights "Variant A"

- ▶ This can also be replaced with a sum over observations (for a large sample) because

$$\int \phi(m) dm = \int g(m) \frac{\phi(m)}{g(m)} dm = \langle \frac{\phi(m)}{g(m)} \rangle \rightarrow \frac{1}{N} \sum_i \frac{\phi(m_i)}{g(m_i)} \quad (10)$$

expectation value

arithmetic mean

- ▶ So an alternative computation is

$$\hat{W}_{xy} = \frac{1}{N} \sum_i \frac{\hat{g}_x(m_i)\hat{g}_y(m_i)}{\hat{g}(m_i)^2} \quad (11)$$

sWeights "Variant B"

- ▶ This also has the nice property that the sum of weights is the number of events *i.e.*  
 $\sum_i \hat{w}_s(m_i) = N\hat{z} = \hat{N}_s$

## Application to a finite sample

- ▶ There is then an interesting connection between the result in Eq. (11) and an extended maximum likelihood (EML) fit
- ▶ Turns out that the  $W$ -matrix is closely related to the covariance matrix of an EML fit with **only the yields floating**

$$\begin{pmatrix} \hat{\alpha}_s & \hat{\beta}_s \\ \hat{\alpha}_b & \hat{\beta}_b \end{pmatrix} = \frac{1}{N^2} \begin{pmatrix} C_{ss} & C_{sb} \\ C_{sb} & C_{bb} \end{pmatrix} \quad (12)$$

sWeights "Variant C"

- ▶ Most of the above (at least the finite sample case) **was already shown** in the *sPlot* paper [3] although we take a different approach
- ▶ They find the link with the correlation matrix in Eq. (12) and name that as the "sWeight"
  - ▶ The implementation in [TSPLOT](#) uses the Minuit / HESSE covariance matrix directly (numerical inaccuracies)
  - ▶ The implementation in [RooStats::SPlot](#) directly computes Eq. (11)

## Application to a finite sample

- ▶ The original *sPlot* formalism [3] comes with **a few limitations**
  - ▶ All shape parameters must be fixed
  - ▶ All yields are freely floating and not expressed as fractions
  - ▶ Can only get weights for the unbinned sample you have fitted
  - ▶ The **RooStats** implementation enforces the above
  
- ▶ The advantages with what we have found is that you **simply need to provide a description of the p.d.f.s** and **you get back a weight function for each component**
  - ▶ Shapes and yields can be determined however you want (even with constraints)
  - ▶ Can perform fit to a different sample to the one you use to extract weights (e.g. wider fit range)
  - ▶ Can perform the fit to a binned sample (if there are many events) and still extract a weight per-event
  
- ▶ There are still **some caveats** which apply to both
  - ▶ The description for each component must factorise in the discriminant and control variables
  - ▶ Factorisation means independence (which is more than just not-linearly-correlated)
  - ▶ However what we will see shortly (COWs) can circumvent this as well

# Properties of different sWeight variants

## Variant A

- ▶ If the true p.d.f.s are known,  $\hat{g}_x(m) = g_x(m)$ , then the  $W_{xy}$  in Eq. (9) produce an unbiased estimate of  $w_s(m)$  with minimum variance
- ▶ Therefore, the sum of weights in bins of the control variable are uncorrelated and the variance in each bin is the sum of squared weights,  $\sum_i \hat{w}_s(m_i)^2$
- ▶ This greatly simplifies parameter estimates when fitting the weighted data

## Variant B

- ▶ The sum of weights reproduces the fitted yield,  $\sum_i \hat{w}_s(m_i) = N\hat{z} = \hat{N}_s$ , which is almost but not exactly true for Variant A
  - ▶ But even if the true p.d.f.s are known,  $\hat{g}_x(m) = g_x(m)$ , then the sums of weights in bins of the control variable are correlated (although in practise the effect is very small)
  - ▶ The correlation means that the variance in each bin is slightly smaller than the estimate from the sum of squared weights
- ▶ In practise the true shapes,  $g_x(m)$ , are rarely known and so in either case care must be taken when fitting the weighted data
- it is wrong to assume the weights are unbiased in finite samples



## How do we extract parameter estimates from the weighted data?

- ▶ This deserves, and requires, a seminar in its own right
- ▶ Fitting weighted data (with uncorrelated weights) in bins is straight forward
- ▶ But bins are **only uncorrelated** when using Variant A **and** if the true shapes,  $g_x(m)$ , are known
- ▶ In any other case you need a **correction to the covariance**
- ▶ This has been realised by **Langenbruch** who has provided a detailed description of how to deal with both the binned and unbinned fits to weighted data [5]  
→ the first calculation to account for correlations between the sWeight estimation and parameter estimation
- ▶ For an unbinned fit we want to maximise the weighted likelihood by solving

$$\sum_i w_i \frac{\partial \ln h_s(t_i; \phi)}{\partial \phi_k} \stackrel{!}{=} 0 \quad (13)$$

- ▶ This is **not a product of probabilities** so the inverse of the Hessian matrix **does not provide an estimate of the covariance**

## How do we extract parameter estimates from the weighted data?

- ▶ For the unbinned case construct the quasi-score function,

$$\mathbf{S}(\boldsymbol{\lambda}) = \mathbf{S}(N_s, N_b, \boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi})$$

$$\mathbf{S}(\boldsymbol{\lambda}) = \begin{pmatrix} \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial N_s \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial N_b \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial \theta_1 \\ \vdots \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial \theta_n \\ \psi_{ss}(N_s, N_b, \boldsymbol{\theta}, W_{ss}) \\ \psi_{sb}(N_s, N_b, \boldsymbol{\theta}, W_{sb}) \\ \psi_{bb}(N_s, N_b, \boldsymbol{\theta}, W_{bb}) \\ \xi_1(\boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi}) \\ \vdots \\ \xi_p(\boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi}) \end{pmatrix} \quad (14)$$

### Sandwich estimator

The “sandwich estimator”,

$$\mathbf{C}_\lambda = \mathbb{E} \left[ \frac{\partial \mathbf{S}}{\partial \boldsymbol{\lambda}^T} \right]^{-1} \mathbb{E} \left[ \mathbf{S} \mathbf{S}^T \right] \mathbb{E} \left[ \frac{\partial \mathbf{S}}{\partial \boldsymbol{\lambda}^T} \right]^{-T} \quad (15)$$

provides the full unbiased covariance

## How do we extract parameter estimates from the weighted data?

- For the unbinned case construct the quasi-score function,

$$\mathbf{S}(\boldsymbol{\lambda}) = \mathbf{S}(N_s, N_b, \boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi})$$

$$\mathbf{S}(\boldsymbol{\lambda}) = \begin{pmatrix} \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial N_s \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial N_b \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial \theta_1 \\ \vdots \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial \theta_n \\ \psi_{ss}(N_s, N_b, \boldsymbol{\theta}, W_{ss}) \\ \psi_{sb}(N_s, N_b, \boldsymbol{\theta}, W_{sb}) \\ \psi_{bb}(N_s, N_b, \boldsymbol{\theta}, W_{bb}) \\ \xi_1(\boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi}) \\ \vdots \\ \xi_p(\boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi}) \end{pmatrix} \quad (14)$$

### Sandwich estimator

The “sandwich estimator”,

$$\mathbf{C}_\lambda = \mathbb{E} \left[ \frac{\partial \mathbf{S}}{\partial \boldsymbol{\lambda}^T} \right]^{-1} \mathbb{E} \left[ \mathbf{S} \mathbf{S}^T \right] \mathbb{E} \left[ \frac{\partial \mathbf{S}}{\partial \boldsymbol{\lambda}^T} \right]^{-T} \quad (15)$$

provides the full unbiased covariance

### For Variant A with known shapes

Fewer terms in  $\mathbf{S}(\boldsymbol{\lambda})$

# How do we extract parameter estimates from the weighted data?

- For the unbinned case construct the quasi-score function,

$$\mathbf{S}(\boldsymbol{\lambda}) = \mathbf{S}(N_s, N_b, \boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi})$$

$$\mathbf{S}(\boldsymbol{\lambda}) = \begin{pmatrix} \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial N_s \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial N_b \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial \theta_1 \\ \vdots \\ \partial \ln \mathcal{L}(N_s, N_b, \boldsymbol{\theta}) / \partial \theta_n \\ \psi_{ss}(N_s, N_b, \boldsymbol{\theta}, W_{ss}) \\ \psi_{sb}(N_s, N_b, \boldsymbol{\theta}, W_{sb}) \\ \psi_{bb}(N_s, N_b, \boldsymbol{\theta}, W_{bb}) \\ \xi_1(\boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi}) \\ \vdots \\ \xi_p(\boldsymbol{\theta}, W_{ss}, W_{sb}, W_{bb}, \boldsymbol{\phi}) \end{pmatrix} \quad (14)$$

## Sandwich estimator

The “sandwich estimator”,

$$\mathbf{C}_\lambda = \mathbf{E} \left[ \frac{\partial \mathbf{S}}{\partial \boldsymbol{\lambda}^T} \right]^{-1} \mathbf{E} \left[ \mathbf{S} \mathbf{S}^T \right] \mathbf{E} \left[ \frac{\partial \mathbf{S}}{\partial \boldsymbol{\lambda}^T} \right]^{-T} \quad (15)$$

provides the full unbiased covariance

## For Variant A with known shapes

Fewer terms in  $\mathbf{S}(\boldsymbol{\lambda})$

## For Variant B with known shapes

Simplifies to

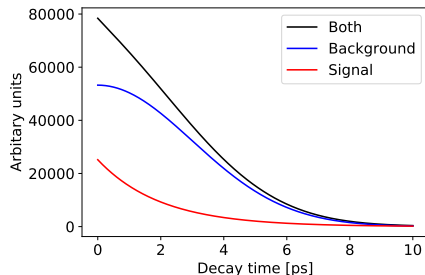
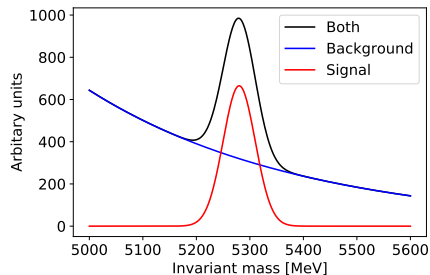
$$\widehat{\mathbf{C}}_\phi = \mathbf{H}^{-1} \mathbf{H}' \mathbf{H}^{-T} - \mathbf{H}^{-1} \mathbf{E} \mathbf{C}' \mathbf{E}^T \mathbf{H}^{-T} \quad (16)$$

### sWeights in practise

Some example applications on toy Monte Carlo

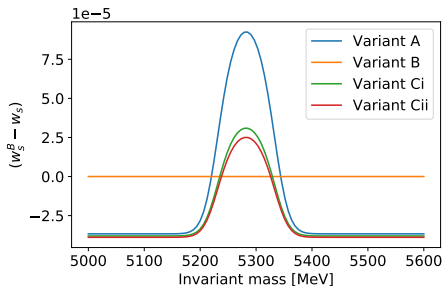
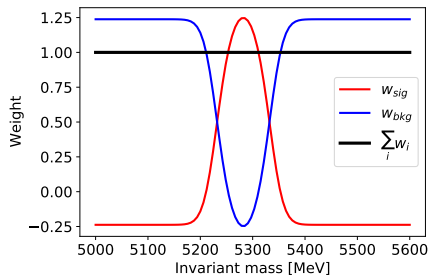
## An example application on toy MC

- ▶ A simple example with signal (background) distributed normally (exponentially) in a discriminant variable, invariant mass
- ▶ The control variable, decay time, is distributed exponentially (normally) for signal (background)



# An example application on toy MC

- ▶ Extract the weight functions using the description above
- ▶ All variants give similar performance - with some small differences



## An example application on toy MC

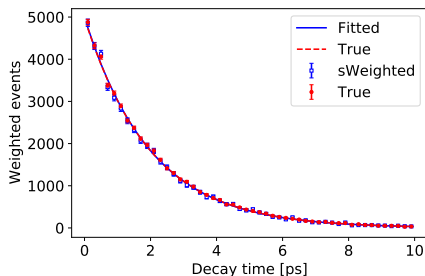
- ▶ Extract the weight functions using the description above
- ▶ All variants give similar performance - with some small differences

<b>Fit methods</b>	$N_s$	$\sigma(N_s)$	$N_b$	$\sigma(N_b)$
EML Fit (all pars.)	49591.22	351.23	200409.16	523.61
EML Fit (yields only)	49591.22	311.25	200409.16	497.69
<b>sWeight methods</b>	$\sum w_s$	$\sqrt{\sum w_s^2}$	$\sum w_b$	$\sqrt{\sum w_b^2}$
Variant A	49591.01	311.26	200408.99	497.70
Variant B	49591.22	311.25	200409.16	497.69
Variant C	49595.97	311.24	200408.98	497.67



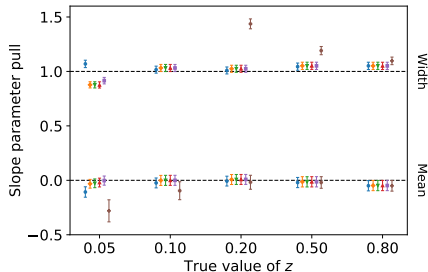
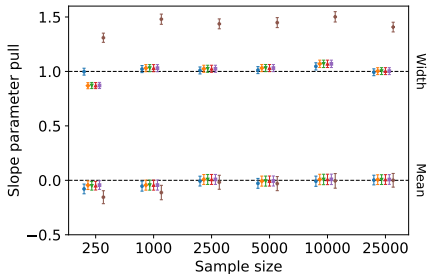
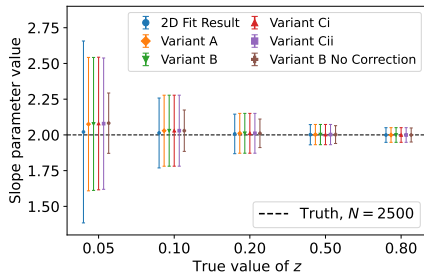
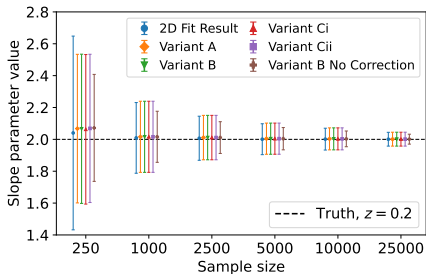
## An example application on toy MC

- ▶ Now apply the weights to the data
- ▶ And fit or extract parameter of interest in the control dimension
- ▶ Including the appropriate correction to the weighted data
- ▶ Minimal loss in precision compared to a full 2D fit (in this case)



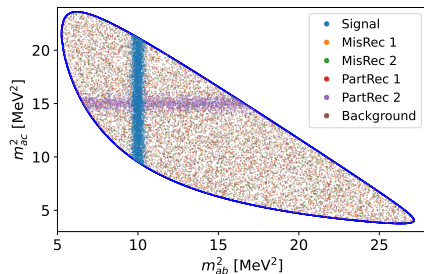
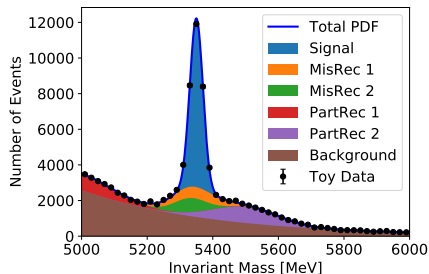
Method	Fit Result
2D Fit	$2.0025 \pm 0.0137$
Variant A	$2.0067 \pm 0.0138$
Variant B	$2.0067 \pm 0.0138$
Variant C	$2.0068 \pm 0.0138$

# The performance over an ensemble



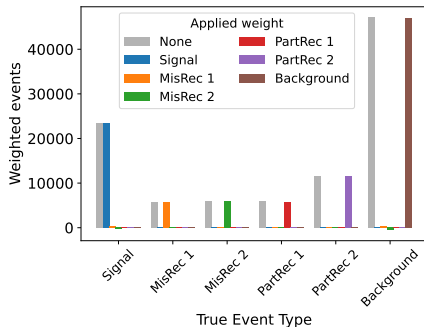
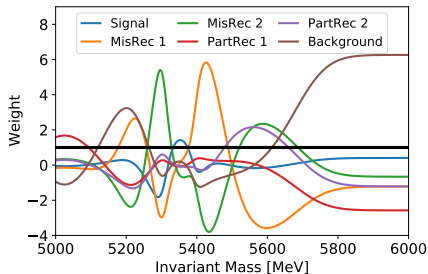
## Walk through a more complex example

- ▶ Now consider several factorising components which overlap in mass
- ▶ Common in heavy flavour physics when final state is mis-reconstructed
- ▶ Have six components, some of which peak under or near the signal
- ▶ Use a Dalitz space as the control variable



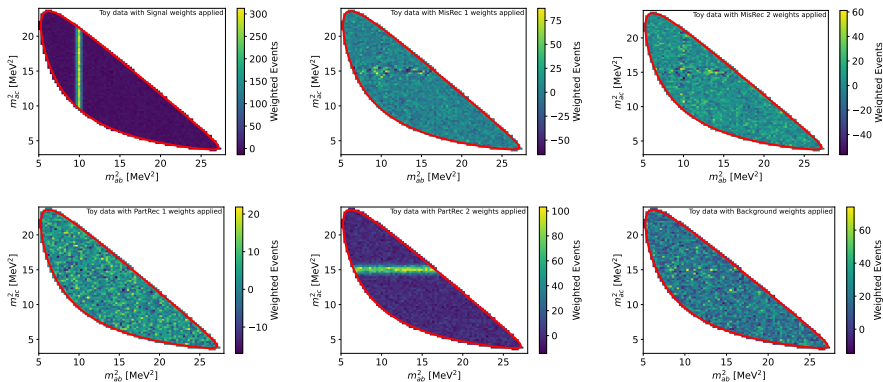
## Walk through a more complex example

- ▶ Extract the weights as above for each of the six components
- ▶ Inspect control distributions when each weight is applied
- ▶ Get a nice recovery of the true distributions



# Walk through a more complex example

- ▶ Get a nice recovery of the true distributions

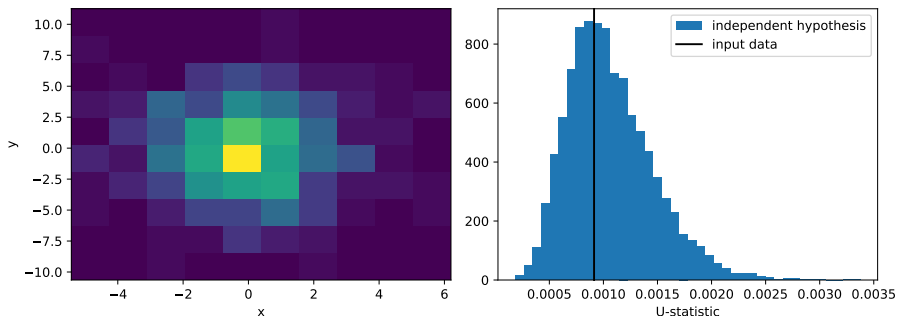


Part 2: COWs

# What if our p.d.f. doesn't factorise?

- ▶ Check for **independence** - need **both** signal and background to factorise for sWeights
- ▶ Checking for correlation is **not a sufficient test** of statistical independence
- ▶ We recommend the  **$U$ -statistic permutation (USP)** test [6] → **powerful and efficient**
- ▶ Implemented in the python **resample** package
- ▶ Input is a 2D histogram of the sample in the discriminant and control variables
- ▶ Output is a  **$p$ -value for consistency with the independent hypothesis**

x,y are independent: p-value=0.6014

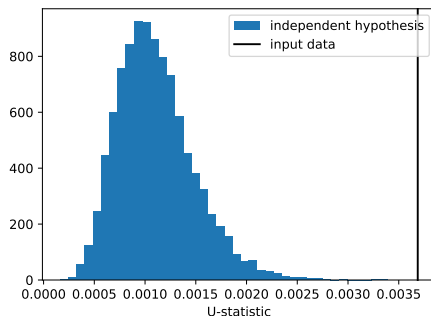
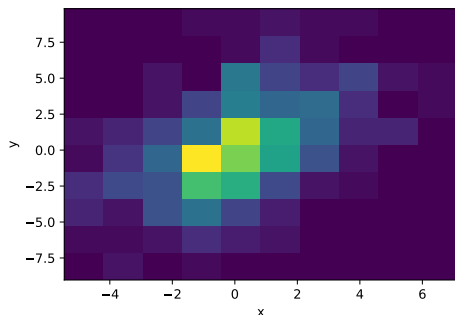


There are also ongoing developments in this area (Schmelling, Dembinski, Langenbruch, MK, Lyons, Berrett, Samworth, Junk)

# What if our p.d.f. doesn't factorise?

- ▶ Check for **independence** - need **both** signal and background to factorise for sWeights
- ▶ Checking for correlation is **not a sufficient test** of statistical independence
- ▶ We recommend the  **$U$ -statistic permutation (USP)** test [6] → **powerful and efficient**
- ▶ Implemented in the python **resample** package
- ▶ Input is a 2D histogram of the sample in the discriminant and control variables
- ▶ Output is a  **$p$ -value for consistency with the independent hypothesis**

x,y are correlated: p-value=0.0001



There are also ongoing developments in this area (Schmelling, Dembinski, Langenbruch, MK, Lyons, Berrett, Samworth, Junk)



- ▶ Let's generalise even more to include a non-factorising efficiency and a non-factorising true p.d.f. so that our observed data is distributed like

$$\rho(m, t) = N_e \epsilon(m, t) f(m, t) \quad \text{with} \quad N_e = 1 / \int \epsilon(m, t) f(m, t) dm dt \quad (17)$$

true distribution
ensure proper normalisation

- ▶ Kolmogorov-Arnold representation theorem [7, 8] states any function  $f(m, t)$  can be represented by a finite sum of factorising terms *i.e.*

$$f(m, t) = \sum_k^n z_k g_k(m) h_k(t) \quad \text{with} \quad \sum_k^n z_k = 1 \quad (18)$$

- ▶ So can associate the first  $j$  terms to our signal and the remaining  $n - j$  to our background(s)

$$f(m, t) = \sum_{k=0}^{j-1} z_k g_k(m) h_k(t) + \sum_{k=j}^n z_k g_k(m) h_k(t) \quad (19)$$

Signal
Background

- ▶ Clearly the more accurate you can be the fewer terms you need (and this is problem specific)

- ▶ Applying the same arguments as above (with orthonormal functions) any single component  $h_k(t)$  of  $f(m, t)$  can be isolated by a weight function

$$w_k(m) = \sum_{l=0}^n \frac{A_{kl} g_l(m)}{I(m)} \quad \text{with} \quad A_{kl}^{-1} = W_{kl} = \int \frac{g_k(m) g_l(m)}{I(m)} dm \quad (20)$$

- ▶ We call  $I(m)$  the **variance function** and it can be any non-zero function you like!
  - ▶ In the paper [1] we show that *any choice* of  $I(m)$  gives unbiased weights
  - ▶ Notice that  $A_{kl}$  is the  $\alpha, \beta$  matrix given above in Eq. (8)
- ▶ It follows that

$$\boxed{\sum_{i=0}^n A_{ki} W_{ij} = \delta_{kj}} \quad \text{and} \quad \boxed{\int \frac{w_k(m) g_l(m)}{I(m)} dm = \delta_{kl}} \quad (21)$$

Unitary condition Ortho-normality condition

- ▶ In the case of a non-uniform efficiency,  $\epsilon(m, t) \neq 1$ , then the weights to project out each component,  $h_k(t)$ , are  $w_k(m)/\epsilon(m, t)$ .
- ▶ For the total signal and background components then

$$\boxed{w_s = \sum_{k=0}^{j-1} \frac{w_k(m)}{\epsilon(m, t)}} \quad \text{and} \quad \boxed{w_b = \sum_{k=j}^n \frac{w_k(m)}{\epsilon(m, t)}} \quad (22)$$

Signal weight function Background weight function

- ▶ For any problem, the basis functions,  $g_k(m)$ , and variance function,  $I(m)$ , determine a set of Custom Orthonormal Weight functions (COWs)
- ▶  $I(m) = 1$  is a valid but suboptimal choice
- ▶ Can we look for some more optimal choices?

A. Choice that minimises the variances of the  $\hat{z}_k$ :

$$I_A(m) = q(m) = \int \frac{\rho(m, t)}{\epsilon^2(m, t)} dt \quad (23)$$

- ▶ This can be obtained from a  $1/\epsilon^2(m, t)$  weighted histogram of the data
- ▶ In the case of a single bin this means  $I(m) \rightarrow 1$
- ▶ A sufficiently fine-binned histogram will approach  $q(m)$

B. Choice where  $\hat{z}_k$  are the ML fit estimates:

$$I_B(m) = \sum_k \hat{z}_k g_k(m) \quad (24)$$

- ▶ Can be obtained from a fit to the data (weighted by  $1/\epsilon(m, t)$ ) or iteratively in a short time
- ▶ This is the *sWeights* solution when  $\epsilon(m, t) = 1$

- ▶ To extract a COW you never have to perform a fit!

- ▶ Comparing these two choices of variance function when there is a uniform efficiency

$$I_A(m) = \sum_{k=0}^n z_k g_k(m) \quad \text{and} \quad I_B(m) = \sum_{k=0}^n \hat{z}_k g_k(m) \quad (25)$$

Minimal variance of  $\hat{z}_k$   ML estimates of  $\hat{z}_k$

- ▶ The *sWeights* solution,  $I_B(m)$ , is the maximum-likelihood estimate of the theoretically optimal weight function,  $I_A(m)$  (asymptotically equivalent)
- ▶ A rather nice finding is that **mismodelling the signal density** does **not bias the weights** it will only increase the variance (proof in the paper [1])
- ▶ Practically, non-factorising background (or signal) components can be handled by a suitable sum of polynomial terms (recommend the Bernstein basis)

### When choosing your COWs you just need to pick

1. A signal density with large, preferably maximal, overlap with the true signal density
2. A background density modelled by a truncated sum of polynomials
3. A variance function obtained directly from the data

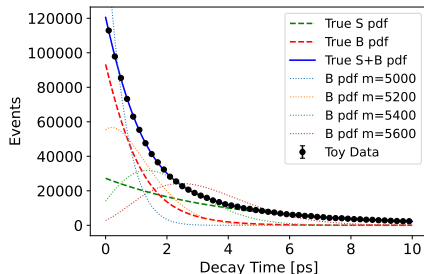
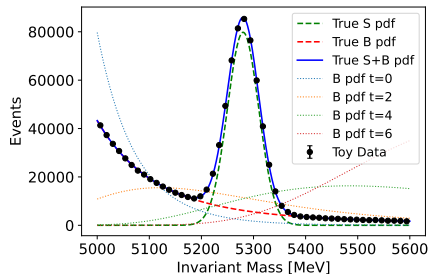
- ▶ **This works regardless of whether the true p.d.f. factorises**

## Milking the COWs

Some example applications on toy Monte Carlo

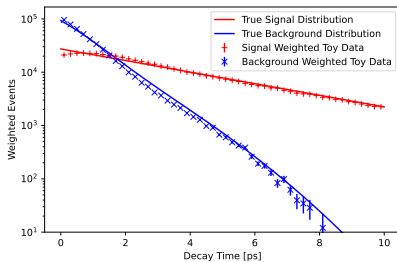
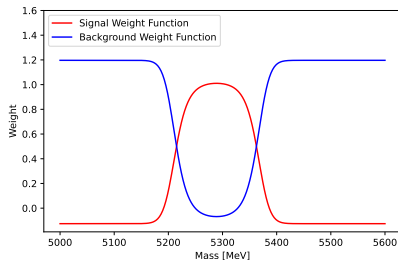
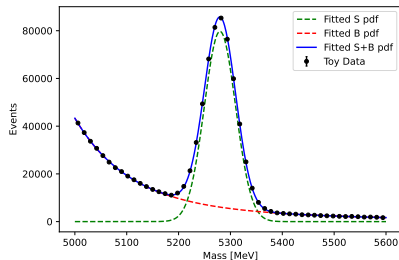
# Some toy examples with COWs

- ▶ Now we can consider a **non-factorising background** with a factorising signal
- ▶ This is a very extreme (and probably highly non-physical) example



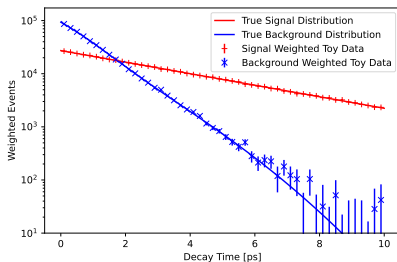
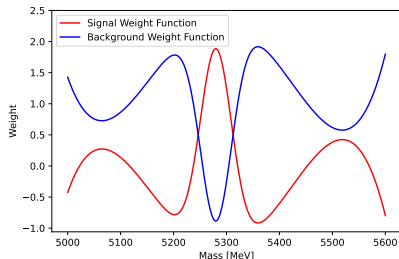
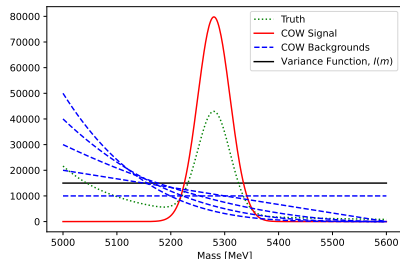
# Some toy examples with COWs

- First we try the classic *sWeights* → which fails



# Some toy examples with COWs

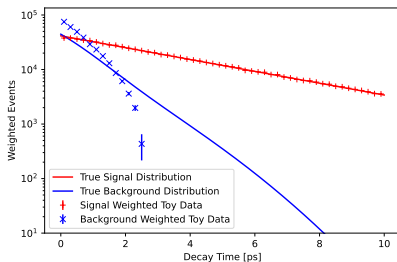
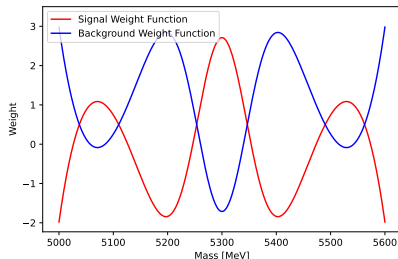
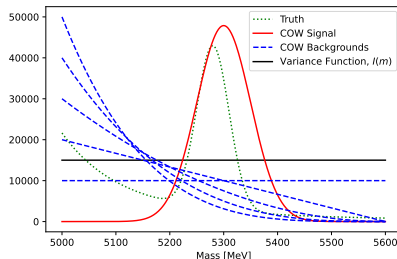
- Then COWs with up to 4th order polynomials for background, the true signal and  $I(m) = 1 \rightarrow$  which works





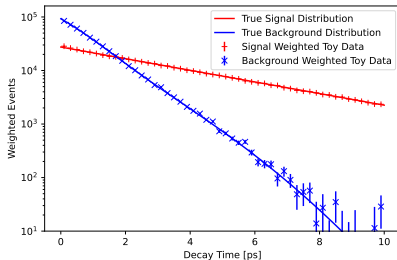
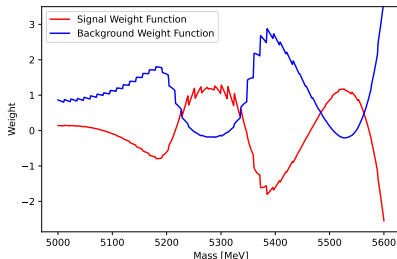
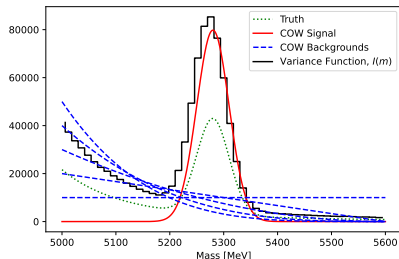
## Some toy examples with COWs

- And then COWs with up to 4th order polynomials for background, the **wrong** signal and  $I(m) = 1 \rightarrow$  which works **but only** for the signal weights



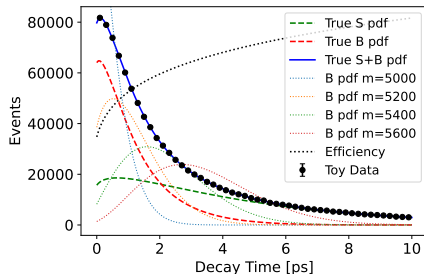
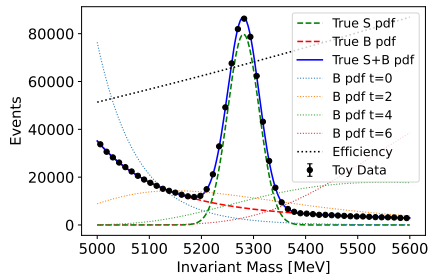
# Some toy examples with COWs

- And then COWs with up to 4th order polynomials for background, the true signal and  $I(m) = q(m) \rightarrow$  which works



## Some toy examples with COWs

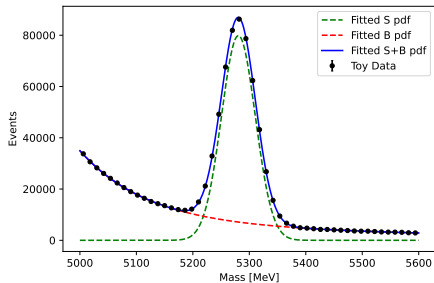
- ▶ Now we can also consider **on top of this** a non-factorising efficiency term
- ▶ More like the real use case in HEP but still highly non-physical



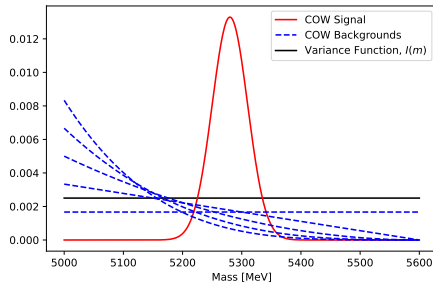
# Some toy examples with COWs

- ▶ Once again compare classic *sWeights* with a COW (4th order, same signal,  $I(m) = 1$ )

## sWeights



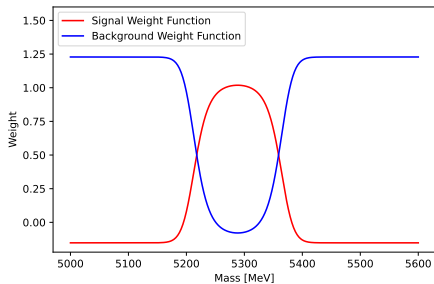
## COWs



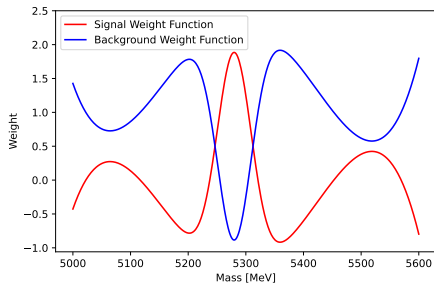
# Some toy examples with COWs

- ▶ Extract the weight functions

## sWeights



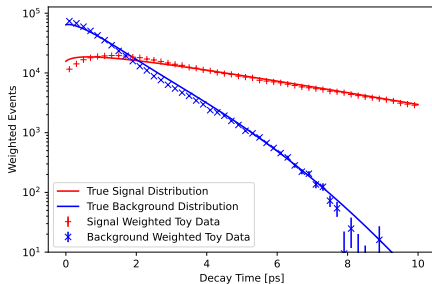
## COWs



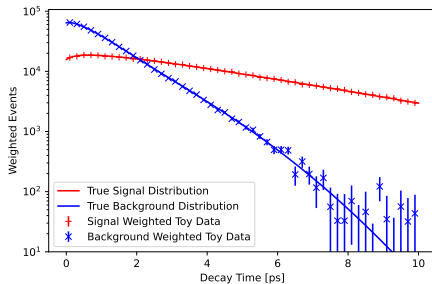
# Some toy examples with COWs

- Fit the weighted distribution in  $t$

## sWeights



## COWs

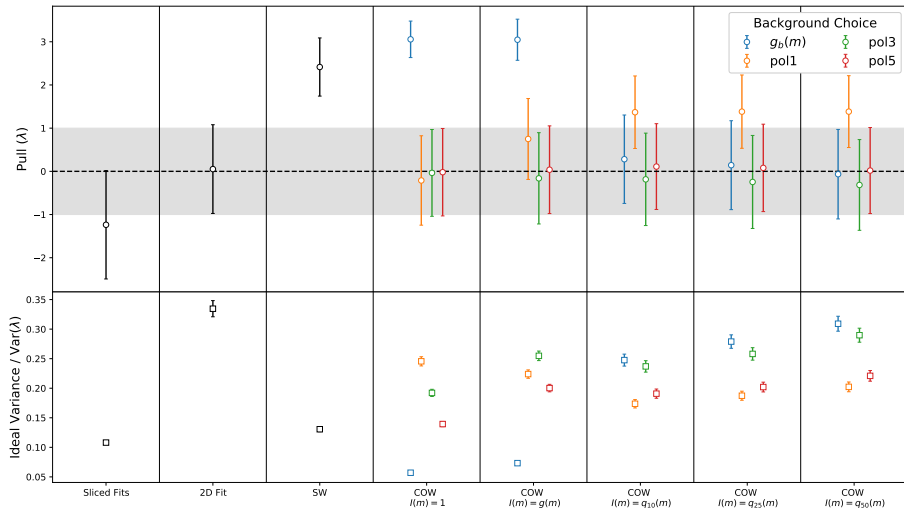


## So what did we learn?

- ▶ COWs do not require any specific fitting
- ▶ You just need to choose
  1. A **signal density** with large, preferably maximal, overlap with the true signal density
  2. A **background density** modelled by a truncated sum of polynomials
  3. A **variance function** obtained directly from the data
- ▶ Any non-factorising component can be dealt with an appropriate sum of polynomials
- ▶ For accurate signal weights, modelling the background is what is important (the signal choice is not), and vice-versa
- ▶ The variance function is arbitrary
- ▶ Better choices of the signal density, variance function and fewer polynomials in the background will provide a smaller variance of the weights

# But we should really check this on ensembles right?

- ▶ Assess performance of various scenarios with slope parameter pull:  $(\lambda_{\text{true}} - \lambda_{\text{fit}}) / \sigma_{\lambda_{\text{fit}}}$
- ▶ Compare sensitivity using the equivalent sample size:  $(\sum_i w_i)^2 / \sum_i w_i^2 / N$

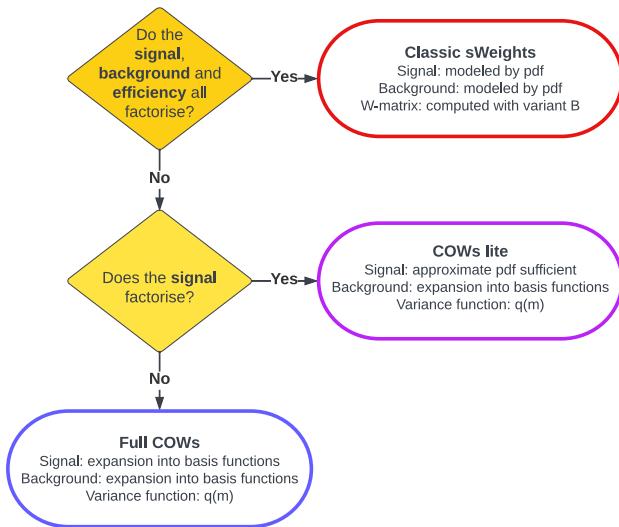




# Summary

Putting the COWs to sleep

# Recommendations



# Conclusions

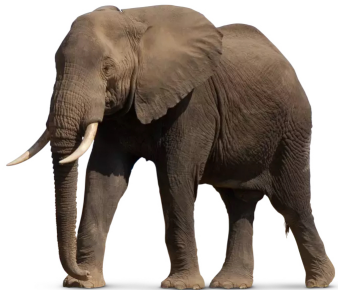
- ▶ I have presented the findings documented in “Custom Orthogonal Weights functions (COWs) for Event Classification” [[arXiv:2112.04574](https://arxiv.org/abs/2112.04574)]
- ▶ We take a fresh look at *sWeights* and derive new formula for their application
- ▶ We describe some of the limitations of the classic *sWeights* approach
- ▶ We show that *sWeights* are a specific case of a more general class of COWs
- ▶ Show that COWs can be used to derive (s)weights for non-factorising problems and problems with non-uniform efficiency
- ▶ Provide closed formulas for computing the covariance of fits to (s)weighted data



*"The ringing in your ears—I think I can help."*

# Conclusions

There is an old saying that with sufficient orders of polynomial you can fit an elephant



*Phrase stolen from [T. Gershon](#)*

With sufficient orders of polynomial you don't even need a fit to extract a COW



## References I

- [1] H. Dembinski, M. Kenzie, C. Langenbruch, and M. Schmelling.  
Custom Orthogonal Weight functions (COWs) for Event Classification .  
*arXiv:2112.04574*.
- [2] M. Schmelling.  
Using sWeights to disentangle signal and background .  
<https://indico.cern.ch/event/940874/contributions/3953530/>.
- [3] Muriel Pivk and Francois R. Le Diberder.  
SPlot: A Statistical tool to unfold data distributions .  
*Nucl. Instrum. Meth. A*, 555:356–369, 2005.
- [4] Roger J. Barlow.  
Event Classification Using Weighting Methods .  
*J. Comput. Phys.*, 72:202–219, 1987.
- [5] C. Langenbruch.  
Parameter uncertainties in weighted unbinned maximum likelihood fits .  
*arXiv:1911.01303*.

- [6] Thomas B. Berrett and Richard J. Samworth.

USP: an independence test that improves on Pearson's chi-squared and the  $G$ -test .

*Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2256):20210549, 2021.

- [7] A. N. Kolmogorov.

On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition.

*Dokl. Akad. Nauk SSSR*, 114:953–956, 1957.

- [8] V. I. Arnold.

On functions of three variables.

*Dokl. Akad. Nauk SSSR*, 114:679–681, 1957.

Back Up

## Optimal choice for $w_s(m)$

- ▶ There are infinitely many choices for  $w_s(m)$  but a sensible (optimal) choice is the one which has minimal variance over the discriminating p.d.f.  $g(m)$  where

$$g(m) = \int f(m, t) dt = z g_s(m) + (1 - z) g_b(m) \quad (26)$$

- ▶ Thus the expectation and variance are

$$\langle w_s(m) \rangle = \int w_s(m) g(m) dm = z \quad (27)$$

$$\langle w_s(m)^2 \rangle - \langle w_s(m) \rangle^2 = \int w_s(m)^2 g(m) dm - z^2 \quad (28)$$

- ▶ Minimise using Lagrange multipliers and known constraints to find extremum of

$$L = \int w_s(m)^2 g(m) dm - z^2 \quad (29)$$

$$- 2\alpha_s \left( \int [w_s(m) g_s(m) - 1] dm \right) - 2\alpha_b \left( \int w_s(m) g_b(m) dm \right) \quad (30)$$



## Optimal choice for $w_s(m)$ continued

- ▶ Using variational calculus:

$$\delta \int w_s(m) \phi(m) dm = \int \delta w_s(m) \phi(m) dm \quad (31)$$

$$\delta \int w_s(m)^2 \phi(m) dm = \int 2w_s(m) \delta w_s(m) \phi(m) dm \quad (32)$$

- ▶ Now want to find where the variation is minimal *i.e.* where

$$\delta L = 2 \int \delta w_s(m) [w_s(m)g(m) - \alpha_s g_s(m) - \alpha_b g_b(m)] dm = 0 \quad (33)$$

- ▶ This is only satisfied for any continuous  $\delta w_s(m)$  if the integrand in the brackets is zero and thus

$$w_s(m) = \frac{\alpha_s g_s(m) + \alpha_b g_b(m)}{g(m)} \quad (34)$$