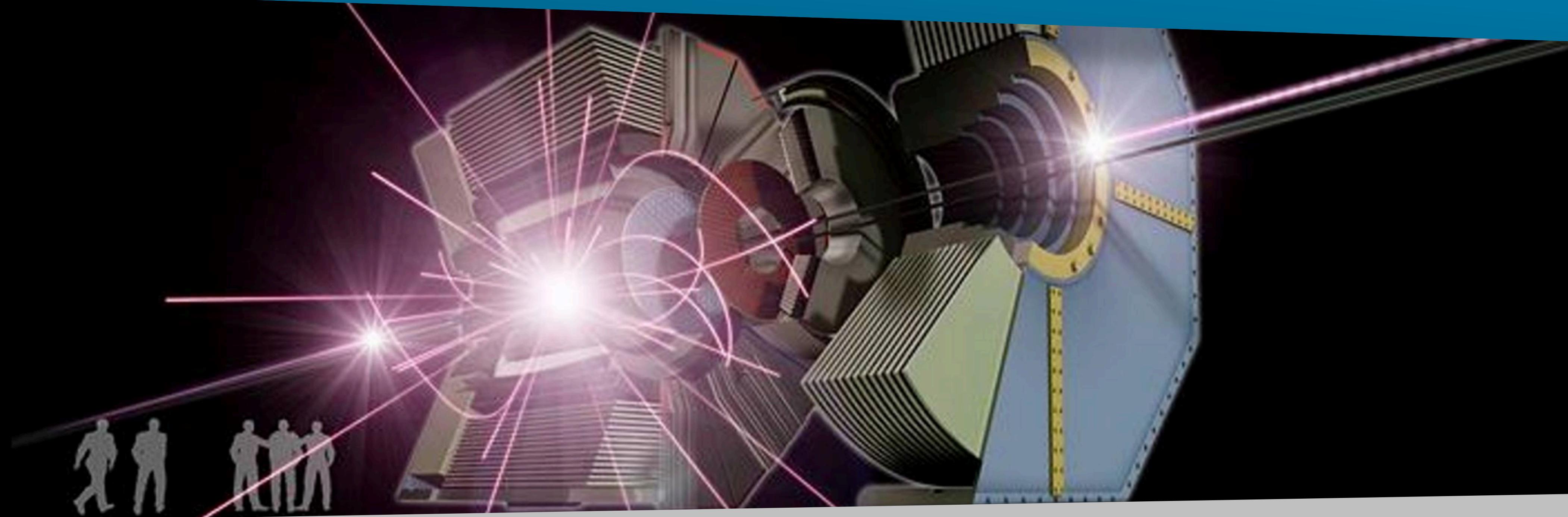


# A hands-on introduction of PyRooUnfold

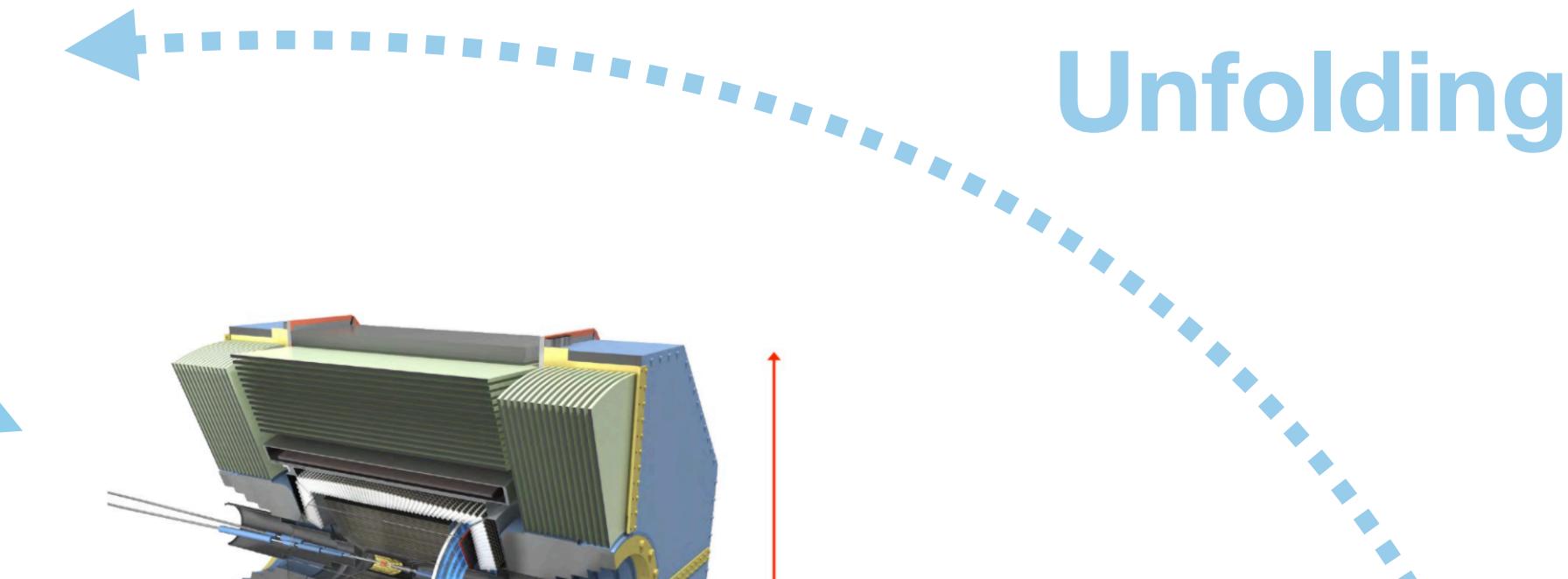


Version: v2.0.0

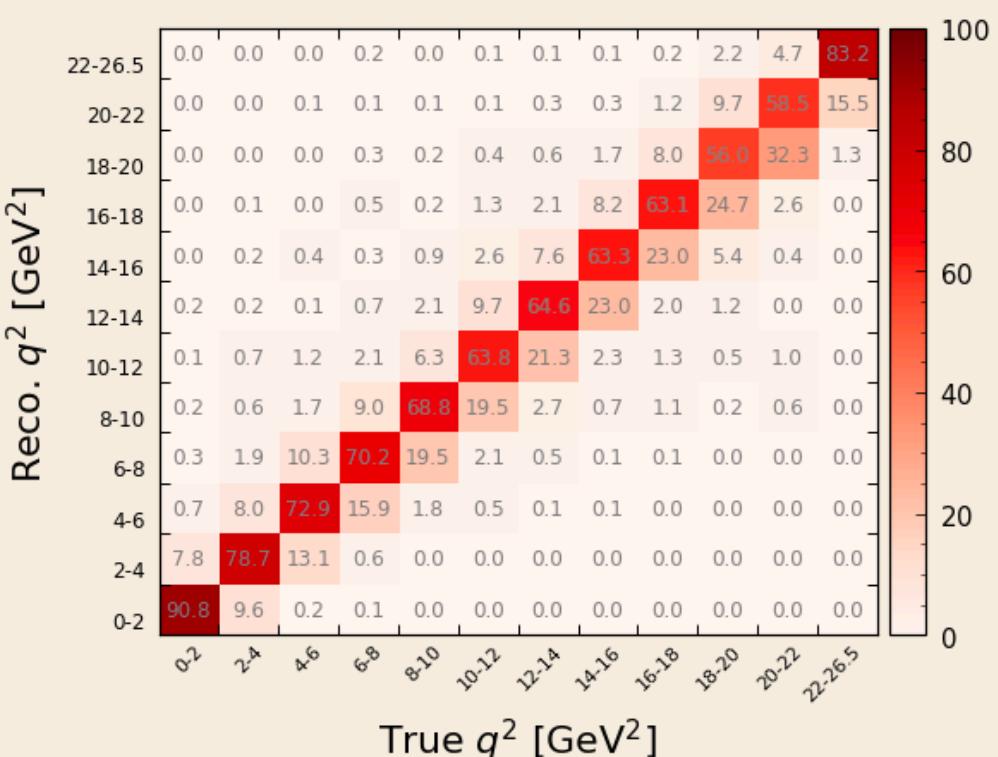
Lu Cao, 8 August 2024

# Recap of Unfolding

- Measured spectra are usually distorted and transformed by various detector effects (resolution, acceptance, etc.)
- Unfolding procedure is necessary/crucial for “shape” measurement



- Detector effects can be extracted with MC and wrap into migration matrix (also called response matrix)



$M(i, j)$  indicates the probability (%) to observe an event in bin  $i$  if it had a generator-level value in bin  $j$

- Unfolded spectra should recover MC truth in executable bias tests (not only Asimov); and avoid oscillating result while keeping minimal bias
- NO universally “excellent” method! Performance highly depends on data shape, resolution, statistics, etc.
- Need validation for each specific case**

# Unfolding Methods

| Method                       | Description  | Parameters  | Comments  | Function in PyRooUnfold |
|------------------------------|--|---|---|-------------------------|
| <b>Bin-by-bin correction</b> | unregularised, correct bin content with MC bin-by-bin factors  | -   | assumes no inter-bin migration; could have biases from the MC model   | do_BinByBin             |
| <b>Matrix inversion</b>      | unregularised matrix inversion with singular value removal   | -   | give large bin-bin correlations and magnify statistical fluctuations; not accurate for small matrices   | do_Invert               |
| <b>TUnfold</b>               | matrix inversion with 0-, 1-, or 2-order polynomial, an adjustable regularisation term of neighbouring bins  | reg. strength $\tau$                                  | optimal tau can be chosen by scanning L-curve; better when nbin_mea > bin_true. By default, tau is determined internally by plotting log10(chi2 squared) against log10(tau) | do_TUnfold( $\tau$ )    |
| <b>Bayes</b>                 | use Bayes' theorem to invert response matrix, regularisation by stopping iterations before reaching "true" inverse   | iteration n<br>(default 4)                            | n needs to be tuned with statistics, bins   | do_Bayes( n )           |
| <b>IDS</b>                   | iterative, dynamically stabilised method, uses stat. significance of the data-MC differences in each bin for regularisation  | iteration n   | allows to treat the effects of new structures in data and the large fluctuations from background subtraction  | do_Ids( n )             |
| <b>SVD</b>                   | singular value decomposition, regularisation with a smooth cut-off on small singular value contribution  | k = 1..nbins  | k too small: dominated by MC truth<br>k too large: dominated by stat. fluctuations<br>k needs to be tuned with bins, sample size  | do_Svd( k )             |
| <b>GP</b>                    | Gaussian Process (GP) method, uses RooUnfoldInvertT to get an initial solution and uses a marginal likelihood minimization to find the optimal regularization. [arXiv: 1811.01242] | kernel choice<br>(only radial "1" is implemented now) | kernel function needs to describe truth well, and global regularisation strength may be varied locally along the spectrum by using a non-stationary kernel                  | do_GP                   |

# PyRooUnfold

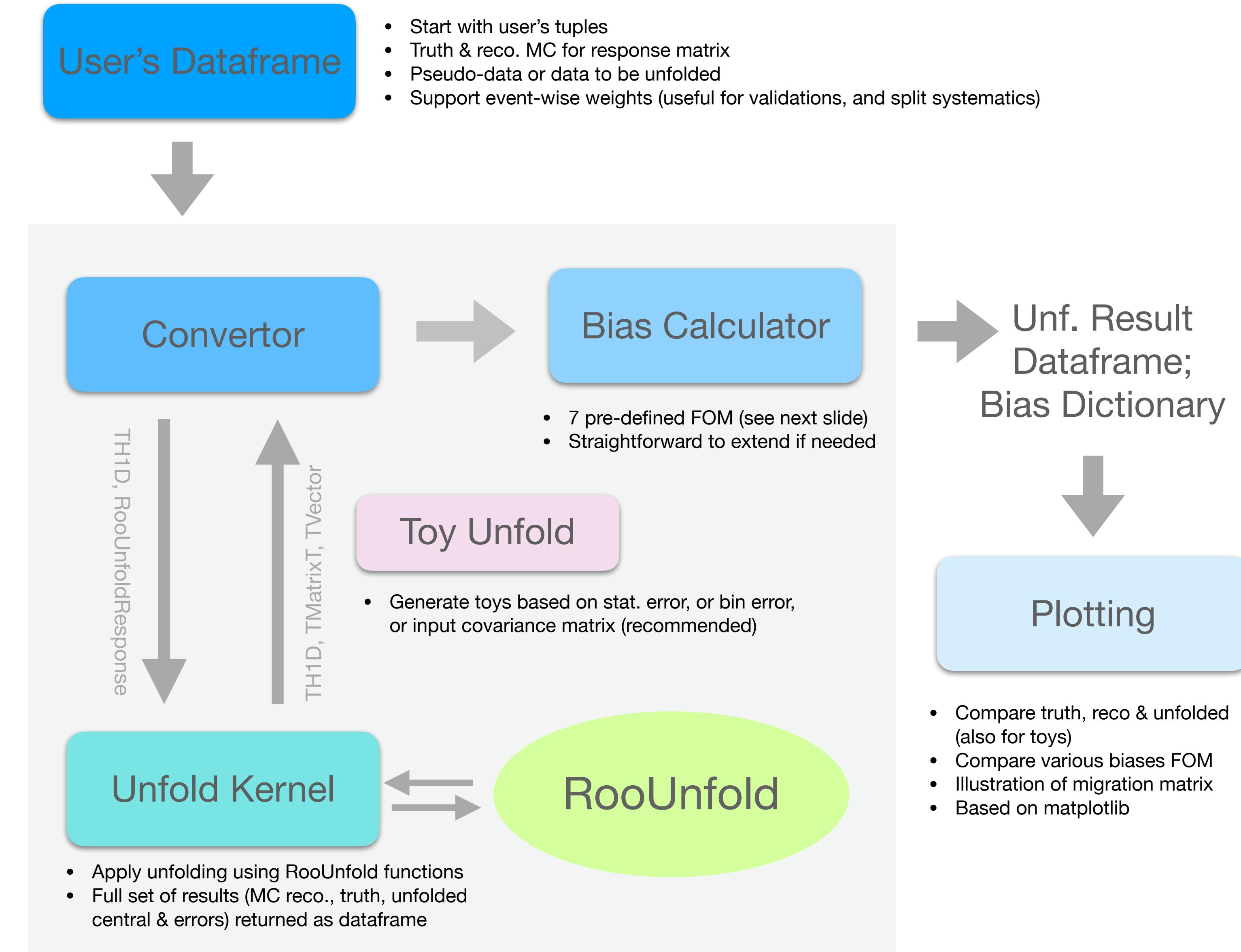
Not only **A Python wrapper of RooUnfold**

Included in the current version:

- Interface of all above methods and major features of RooUnfold
- Toys generation with input error/cov.
- Bias calculator
- Commonly needed plotting functions

**Versions/Tags:**

- v2.0.0: compatible with RooUnfold 3.0.0(2)
- v1.0.0: compatible with RooUnfold 2.0.1



# Bias Study F.O.M.

| Figure of Merit   | Description  | Nickname |
|---|--|----------|
| $\sum_i b_i$  | Sum of absolute biases in each bin, $b_i = N_i^{\text{unfolded}} - N_i^{\text{true}}$  | a        |
| $\sum_i b_i / N_i^{\text{true}}$  | Sum of normalized absolute biases ( $N_i^{\text{true}} \neq 0$ )   | b        |
| $\sum_i b_i$  | Sum of biases, which can be zero when significant biases are present in individual bins but cancel in the sum (balanced oscillation)   | c        |
| $\sqrt{\sum_{i,j} Cov_{i,j}}$   | The square root of the sum of all elements of the post-unfold covariance matrix.   | d        |
| $\sum_i \frac{b_i}{\sqrt{\sum_{i,j} Cov_{i,j}}}$  | Ratio of the sum of absolute biases and the error taking into account bin-to-bin correlations.   | e        |
| $\sqrt{\left(\sum_i b_i\right)^2 + \sum_{i,j} Cov_{i,j}}$                               | The total biases absorbing the total error with bin-to-bin correlations, which is used to check the general oscillation.   | f        |
| $\sum_i \frac{b_i}{\sqrt{Cov_{ii}}}$  | The summed ratio of the absolute bias and unfolding error in each bin.   | g        |
| $\Phi\left(\frac{b_i}{\sigma_i} + 1\right) - \Phi\left(\frac{b_i}{\sigma_i} - 1\right)$ | Coverage probability of intervals $[N_i^{\text{unfolded}} - \sigma_i, N_i^{\text{unfolded}} + \sigma_i]$ , assuming $N_i^{\text{unfolded}}$ is Gaussian distributed and $\Phi$ is the normal cumulative distribution function. | h        |

Every type has a good reason but also has limitations. Checking a combination of several is needed.

```
from pyroounfold.unfold import unfold

example_q2 = unfold(
    df_train = df_mc,
    weight_train = df_mc['weight_train'],           ← df for defining migration matrix
                                                    Asimov unfolding, if train = test
    df_test = df_mc,
    weight_test = df_mc['weight_test'],             ← df of unfolding target, can be re-set by example_q2.set_hist_measure(bin_centre, bin_error)
    name_var_true = 'true_q2',
    name_var_reco = 'reco_q2',                      ← column names of variables
    show_var = r'$q^2$', # for plotting
    bins = q2_bins                                     ← binning of variables
#optional input:
    reco_bin_error ← reco. bin-wise uncertainties need to propagated in unfolding
    reco_cov ← reco. covariance matrix, alternative of bin-wise uncertainties
    kcovtoy ← Error propagation method (RooUnfold):
              False - default, errors from the covariance matrix given by the unfolding (propagate reco_bin_error & reco_cov)
              True - based on the variation of internal toys tests
    mc_stat_err
)
) see next slide
```

# MC Stat. Error from Migration Matrix

```
mc_stat_err <- Interfaced argument from RooUnfolding::SystematicsTreatment dosys
```

Include systematic errors from response matrix due to limited MC statistics?

- 0 [kNoSystematics]: leave out the effect of MC statistical uncertainties on the migration matrix.
- 1 [kGammas]: include the effect by running internal toys
- 2 [kAlphas]: to exclude measurement errors.
- 3 [kAll IncludeSystematics()default]: all included
- 4 [kNoStatistics]

# How-to

# Single-run unfolding

| Method                | Description  | Parameters   | Comments  | Function in PyRooUnfold |
|-----------------------|--|--|---|-------------------------|
| Bin-by-bin correction | unregularised, correct bin content with MC bin-by-bin factors  | -  | assumes no inter-bin migration; could have biases from the MC model   | do_BinByBin             |
| Matrix inversion      | unregularised matrix inversion with singular value removal   | -  | give large bin-bin correlations and magnify statistical fluctuations; not accurate for small matrices   | do_Invert               |
| TUnfold               | matrix inversion with 0-, 1-, or 2-order polynomial, an adjustable regularisation term of neighbouring bins  | reg. strength $\tau$                               | optimal tau can be chosen by scanning L-curve; better when nbin_mea > bin_true. By default, tau is determined internally by plotting log10(chi2 squared) against log10(tau) | do_TUnfold( $\tau$ )    |
| Bayes                 | use Bayes' theorem to invert response matrix, regularisation by stopping iterations before reaching "true" inverse   | iteration n (default 4)                            | n needs to be tuned with statistics, bins   | do_Bayes( n )           |
| IDS                   | iterative, dynamically stabilised method, uses stat. significance of the data-MC differences in each bin for regularisation  | iteration n  | allows to treat the effects of new structures in data and the large fluctuations from background subtraction  | do_ids( n )             |
| SVD                   | singular value decomposition, regularisation with a smooth cut-off on small singular value contribution  | k = 1..nbins                                       | k too small: dominated by MC truth<br>k too large: dominated by stat. fluctuations<br>k needs to be tuned with bins, sample size  | do_Svd( k )             |
| GP                    | Gaussian Process (GP) method, uses RooUnfoldInvertT to get an initial solution and uses a marginal likelihood minimization to find the optimal regularization. [arXiv: 1811.01242] | kernel choice (only radial "1" is implemented now) | kernel function needs to describe truth well, and global regularisation strength may be varied locally along the spectrum by using a non-stationary kernel                  | do_GP                   |

Regularisation parameter can be indicated as input:

```
example_q2.do_Svd(5)
```

```
example_q2.do_Bayes(30)
```



**example\_q2.do\_Invert()** ← apply unfolding using matrix-inversion method  
**example\_q2.result\_df** ← result of unfolding  
**example\_q2.result\_cov** ← post-unfolding covariance matrix

| bin_index | truth_central | measured_central | unfolded_central | unfolded_error | coverage_perbin |
|-----------|---------------|------------------|------------------|----------------|-----------------|
| 0         | 0             | 0.578678         | 0.578678         | 0.598678       | 0.119736        |
| 1         | 1             | 0.296794         | 0.296794         | 0.290794       | 0.058159        |
| 2         | 2             | 0.652948         | 0.652948         | 0.658948       | 0.131790        |
| 3         | 3             | 0.606196         | 0.606196         | 0.616196       | 0.123239        |
| 4         | 4             | 0.643521         | 0.643521         | 0.623521       | 0.124704        |
| 5         | 5             | 0.624760         | 0.624760         | 0.604760       | 0.120952        |
| 6         | 6             | 0.682689         | 0.682689         | 0.782689       | 0.156538        |
| 7         | 7             | 0.682324         | 0.682324         | 0.682324       | 0.136465        |
| 8         | 8             | 0.681740         | 0.681740         | 0.681740       | 0.136348        |
| 9         | 9             | 0.635936         | 0.635936         | 0.635936       | 0.127187        |

# How-to

# Single-run unfolding

| Method                | Description  | Parameters   | Comments   | Function in PyRooUnfold |
|-----------------------|--|--|--|-------------------------|
| Bin-by-bin correction | unregularised, correct bin content with MC bin-by-bin factors  | -  | assumes no inter-bin migration; could have biases from the MC model  | do_BinByBin             |
| Matrix inversion      | unregularised matrix inversion with singular value removal   | -  | give large bin-bin correlations and magnify statistical fluctuations; not accurate for small matrices  | do_Invert               |
| TUnfold               | matrix inversion with 0-, 1-, or 2-order polynomial, an adjustable regularisation term of neighbouring bins  | reg. strength $\tau$                               | optimal tau can be chosen by scanning L-curve; better when nbins_mea > bin_true. By default, tau is determined internally by plotting log10(chi2 squared) against log10(tau) | do_TUnfold( $\tau$ )    |
| Bayes                 | use Bayes' theorem to invert response matrix, regularisation by stopping iterations before reaching "true" inverse   | iteration n (default 4)                            | n needs to be tuned with statistics, bins  | do_Bayes(n)             |
| IDS                   | iterative, dynamically stabilised method, uses stat. significance of the data-MC differences in each bin for regularisation  | iteration n  | allows to treat the effects of new structures in data and the large fluctuations from background subtraction   | do_ids(n)               |
| SVD                   | singular value decomposition, regularisation with a smooth cut-off on small singular value contribution  | k = 1..nbins                                       | k too small: dominated by MC truth<br>k too large: dominated by stat. fluctuations<br>k needs to be tuned with bins, sample size   | do_Svd(k)               |
| GP                    | Gaussian Process (GP) method, uses RooUnfoldInvertT to get an initial solution and uses a marginal likelihood minimization to find the optimal regularization. [arXiv: 1811.01242] | kernel choice (only radial "1" is implemented now) | kernel function needs to describe truth well, and global regularisation strength may be varied locally along the spectrum by using a non-stationary kernel                   | do_GP                   |

Regularisation parameter can be indicated as input:

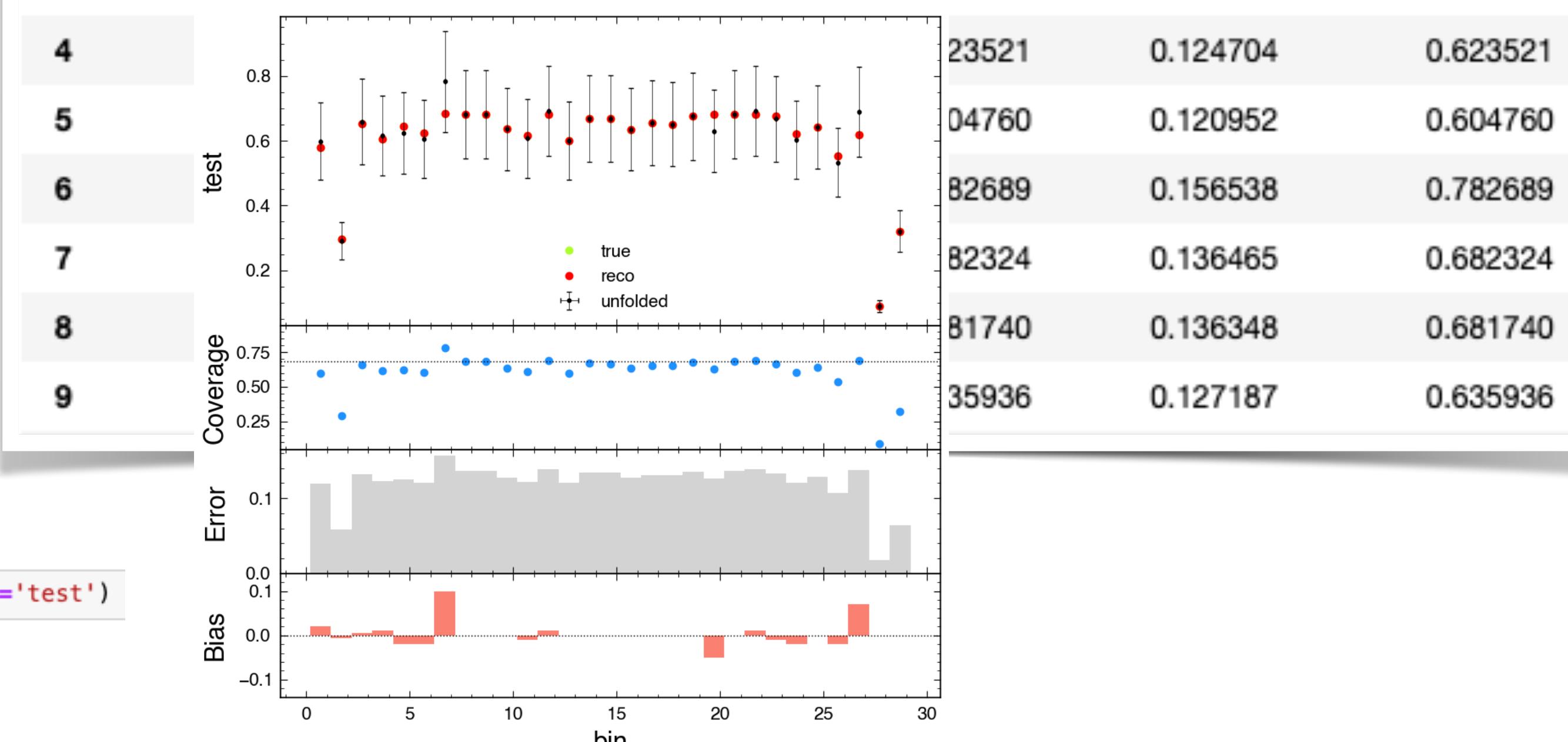
```
example_q2.do_Svd(5)
```

```
example_q2.do_Bayes(30)
```

```
fig = plot_compare_single_run_check(example_q2.result_df, bins, x_title='bin', y_title='test')
```

example\_q2.do\_Invert() ← apply unfolding using matrix-inversion method  
example\_q2.result\_df ← result of unfolding  
example\_q2.result\_cov ← post-unfolding covariance matrix

| bin_index | truth_central | measured_central | unfolded_central | unfolded_error | coverage_perbin |
|-----------|---------------|------------------|------------------|----------------|-----------------|
| 0         | 0.578678      | 0.578678         | 0.598678         | 0.119736       | 0.598678        |
| 1         | 0.296794      | 0.296794         | 0.290794         | 0.058159       | 0.290794        |
| 2         | 0.652948      | 0.652948         | 0.658948         | 0.131790       | 0.658948        |
| 3         | 0.606196      | 0.606196         | 0.616196         | 0.123239       | 0.616196        |
| 4         |               |                  | 23521            | 0.124704       | 0.623521        |
| 5         |               |                  | 04760            | 0.120952       | 0.604760        |
| 6         |               |                  | 82689            | 0.156538       | 0.782689        |
| 7         |               |                  | 82324            | 0.136465       | 0.682324        |
| 8         |               |                  | 81740            | 0.136348       | 0.681740        |
| 9         |               |                  | 35936            | 0.127187       | 0.635936        |



```

from pyroounfold.toy_unfold import toy_unfold

example_toy = toy_unfold(
    df_train = df_train,
    weight_train = df_train['weight'],
    df_test = df_test,
    weight_test = df_test['weight'],      ← same as defined in single-run unfolding
    name_var_true = 'truth',
    name_var_reco = 'measured',
    show_var = 'variable',
    bins = my_bins,
    toy_size = 1000 ← number of toys
)

## optional input
kcovtoy
mc_stat_err

reco_cov ← if reco_cov is given, toys will be generated based on it with multivariate Gaussian

reco_bin_error ← if no reco_cov, toys generation is based on reco_bin_error using Gaussian;
)           if none of above two are given, Poisson distribution based on stat. error of bin count will be used.
               Generated toys are stored as dataframe example_toy.toys_df.

example_toy.do_toyUnfold( method='Invert',      ← apply unfolding with chosen method/parameters
                         get_fom=True)
example_toy.do_toyUnfold( method='Svd', para=5,
                         get_fom=True) ← flag to calculate and save bias F.O.M.

example_toy.result_cen_mean ← median of central values from unfolded toys
example_toy.result_cen_err ← standard deviation of errors from unfolded toys
example_toy.result_coverage_perbin ← averaged coverages from unfolded toys for each bin

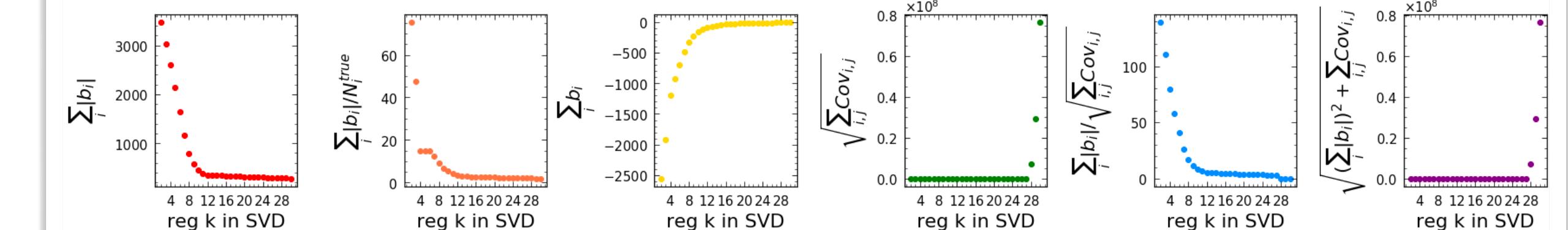
```

## For toys:

```
example_toy.dict_fom ← result of bias FOM if get_fom=True; returned as dictionary {'fom_a', 'fom_b', ... nicknames listed previously }
```

## For toys & various regulation parameters:

```
svd_cen, svd_err, svd_fom = example_toy.do_toyUnfold_scan( method='Svd', para_arr=np.arange(2,30), get_fom=True)
```



## For single-run unfolding:

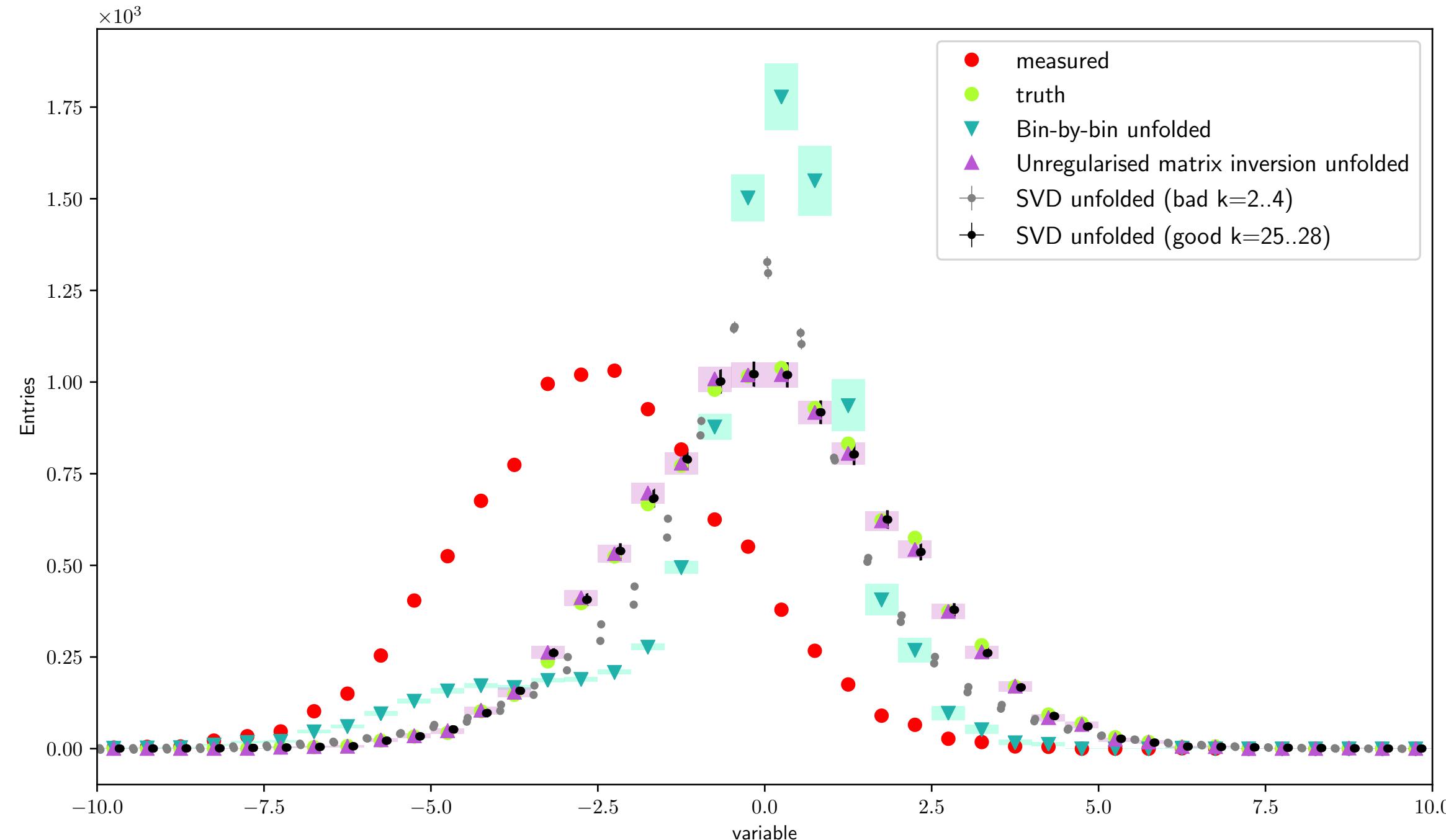
```
example_q2.do_Svd(5)
```

```
example_q2.check_bias() ← calculate and save bias F.O.M.
```

```
example_q2.bias_a ← results of bias F.O.M, bias_a, ..., bias_h
example_q2.bias_b
```

| Figure of Merit   | Description  | Nickname |
|---|--|----------|
| $\sum_i  b_i $  | Sum of absolute biases in each bin, $b_i = N_i^{\text{unfolded}} - N_i^{\text{true}}$  | a        |
| $\sum_i  b_i /N_i^{\text{true}}$  | Sum of normalized absolute biases ( $N_i^{\text{true}} \neq 0$ )   | b        |
| $\sum_i b_i$  | Sum of biases, which can be zero when significant biases are present in individual bins but cancel in the sum (balanced oscillation)   | c        |
| $\sqrt{\sum_{i,j} Cov_{ij}}$  | The square root of the sum of all elements of the post-unfold covariance matrix.   | d        |
| $\sum_i \frac{ b_i }{\sqrt{\sum_{i,j} Cov_{ij}}}$                                       | Ratio of the sum of absolute biases and the error taking into account bin-to-bin correlations.   | e        |
| $\sqrt{\left(\sum_i  b_i \right)^2 + \sum_{i,j} Cov_{ij}}$                              | The total biases absorbing the total error with bin-to-bin correlations, which is used to check the general oscillation.   | f        |
| $\sum_i \frac{ b_i }{\sqrt{Cov_{ii}}}$  | The summed ratio of the absolute bias and unfolding error in each bin.   | g        |
| $\Phi\left(\frac{b_i}{\sigma_i} + 1\right) - \Phi\left(\frac{b_i}{\sigma_i} - 1\right)$ | Coverage probability of intervals $[N_i^{\text{unfolded}} - \sigma_i, N_i^{\text{unfolded}} + \sigma_i]$ , assuming $N_i^{\text{unfolded}}$ is Gaussian distributed and $\Phi$ is the normal cumulative distribution function. | h        |

# Validation Method And Parameters

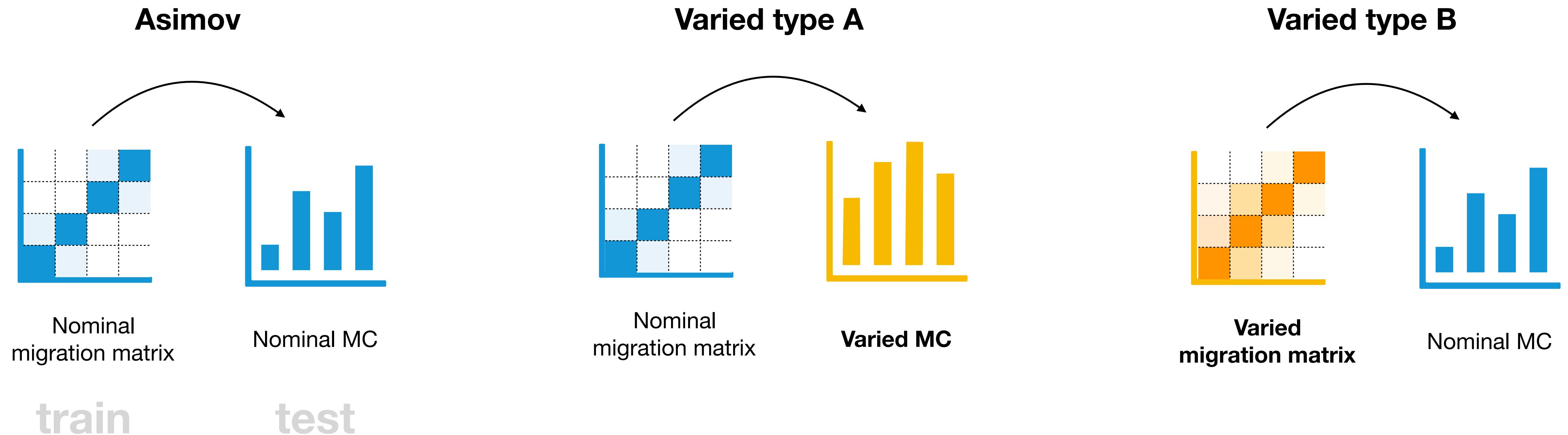


- In this test, matrix-inv and SVD (good  $k$ ) recover the truth centres well.
- Regulation parameter  $k$  in SVD:
  - + dependence on MC shape
  - + unfolding uncertainty

2 ← **small  $k$**  → **#bins** **large  $k$**

**Take-home message:**  
**choose method / tune para. with  
cautions for each specific case**

# Validation Scenario



- Asimov test is mandatory validation but might be not sufficient
- Varied MC / migration matrix provide test scenario for potential mis-modelled shape and resolution in MC
- Such variations can be used to evaluate systematic uncertainty

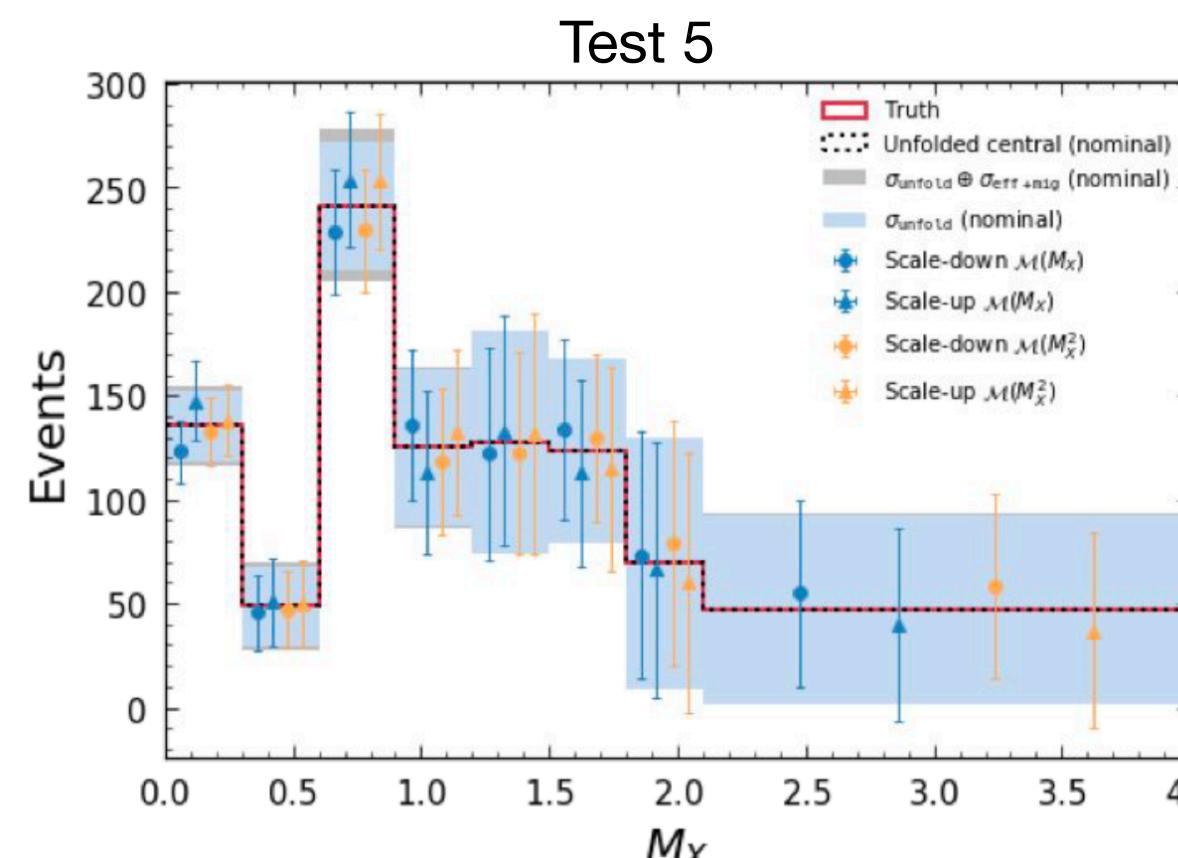
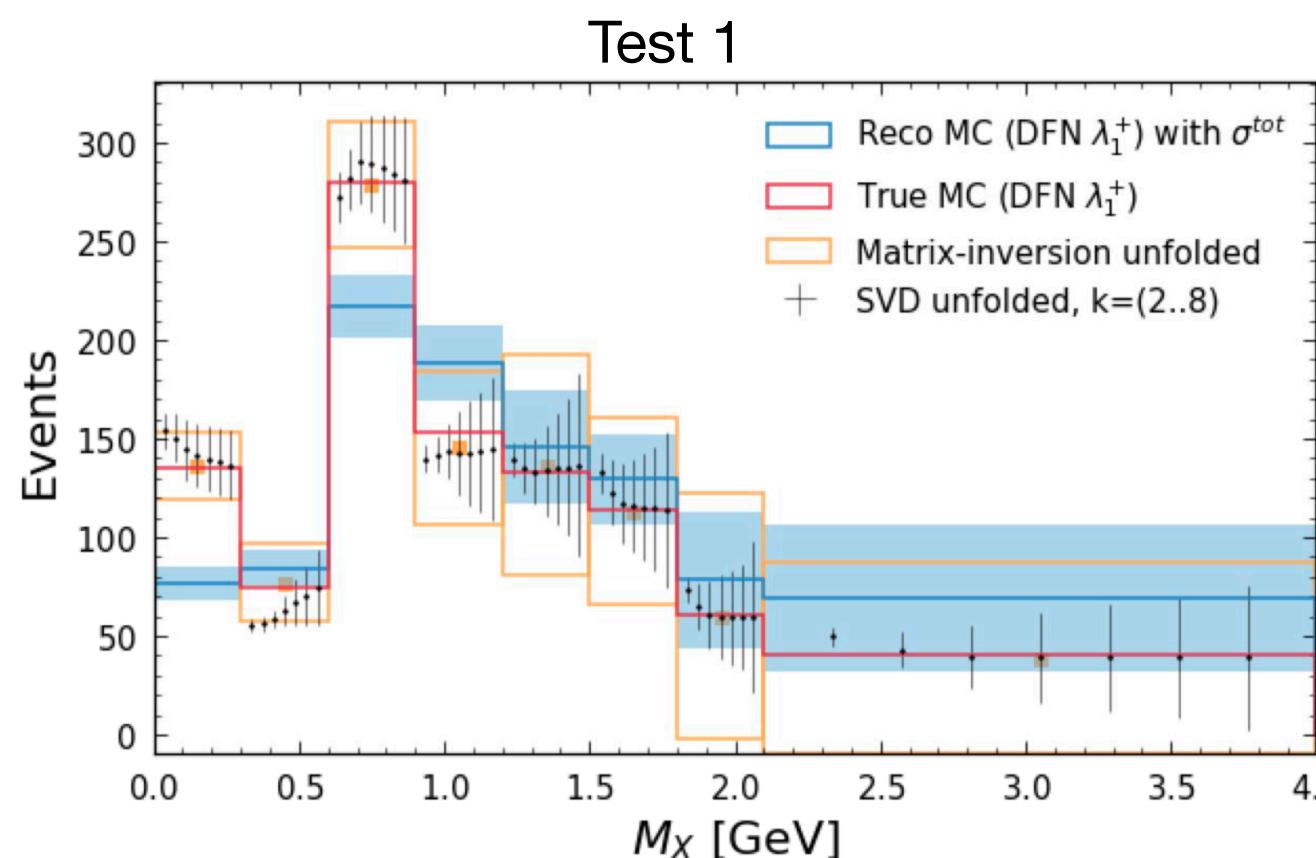
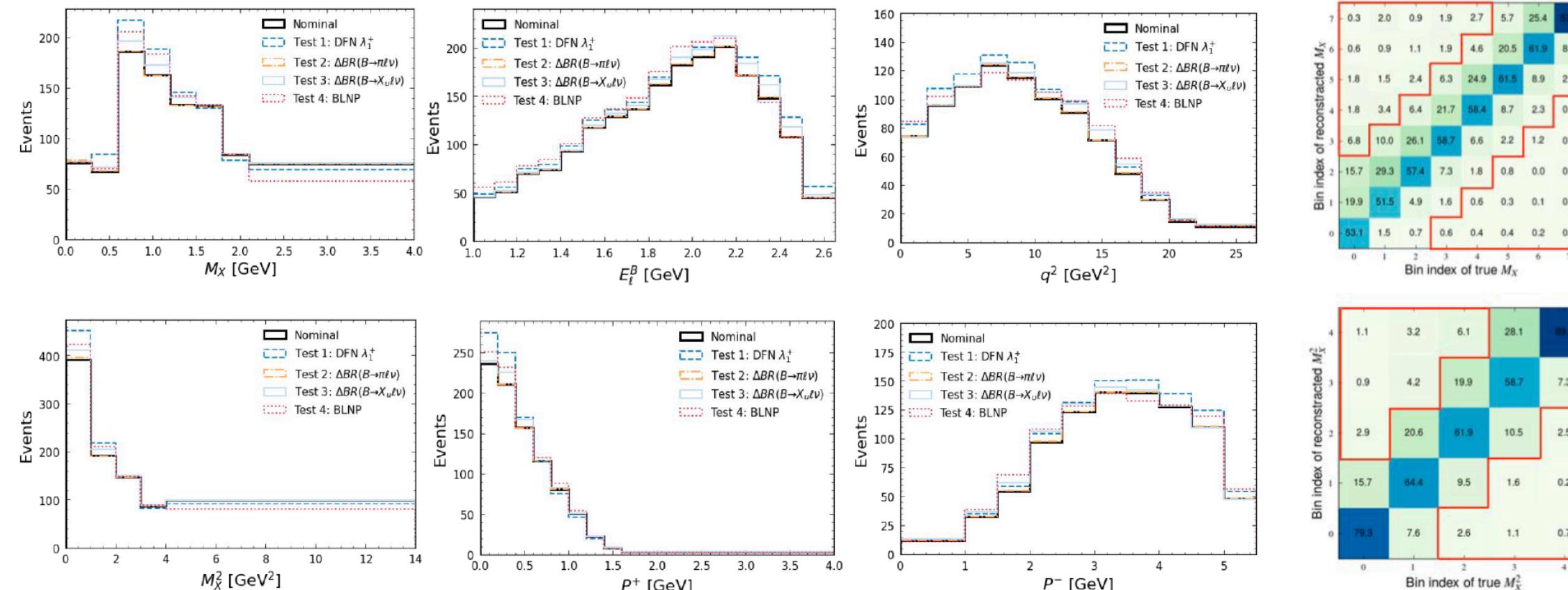
# Benchmark Tests of Unfolding Differential $B \rightarrow X_u \ell \nu$ Spectra

## Type A:

- Use nominal migration matrix to **unfold shifted MC**:
  - **Test 1:** shifted on one eigen-direction of inclusive modelling para. **DFN  $\lambda_1^+$** 
    - Enlarge +50% unc. in used DFN model for testing at an extreme point (max. 60 MeV for  $m_b$  and 0.6 for  $a$ )
  - **Test 2:** shifted  $1\sigma$  on excl. component  **$\Delta BR(B \rightarrow \pi l \nu)$**
  - **Test 3:** shifted  $1\sigma$  on incl.  **$\Delta BR(B \rightarrow X_u l \nu)$**
  - **Test 4:** replace DFN to **BLNP**

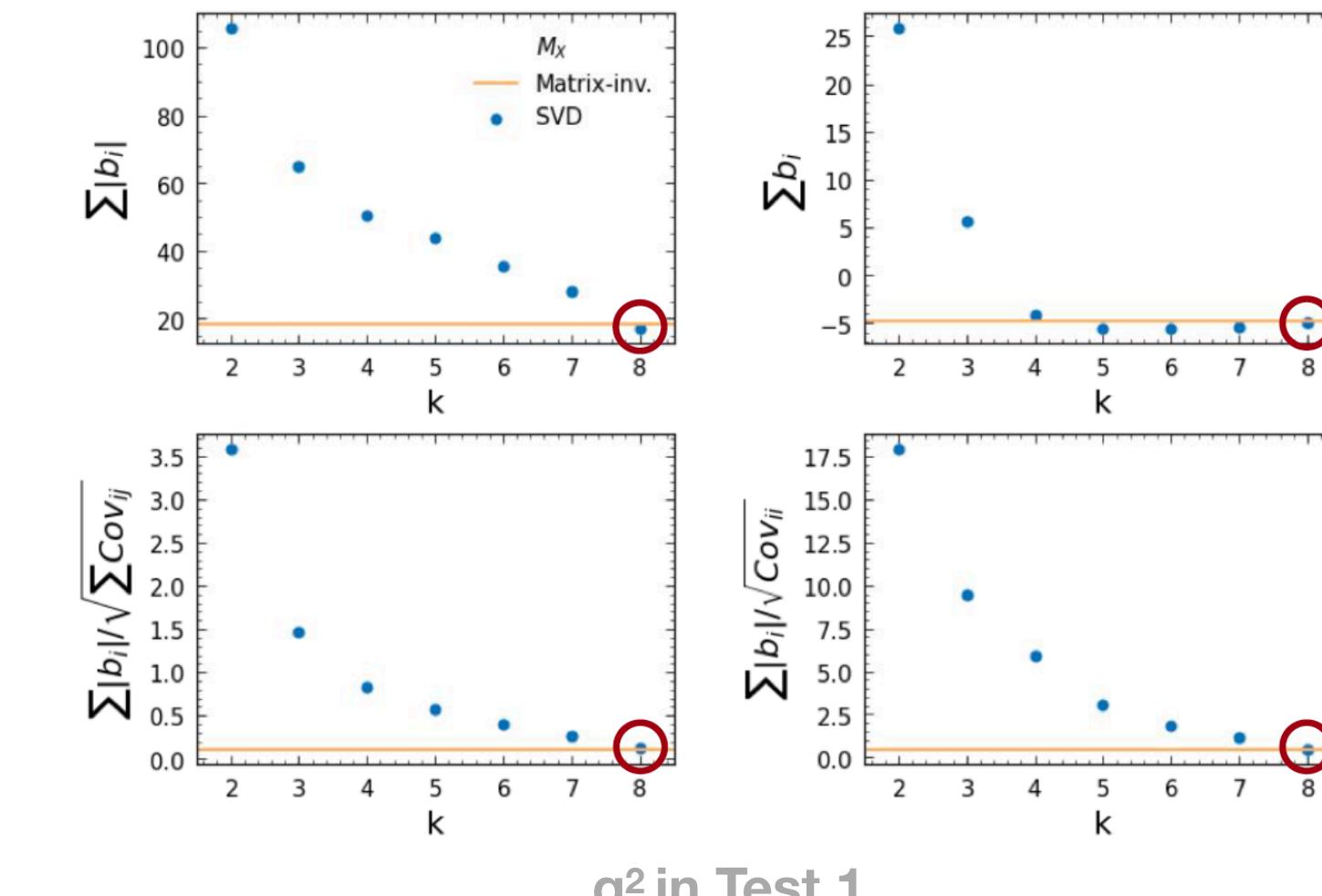
## Type B:

- **Test 5:** varied resolution effects in  $M_X$  and  $M_X^2$  migration matrix by  $\pm 100\%$  in tail region

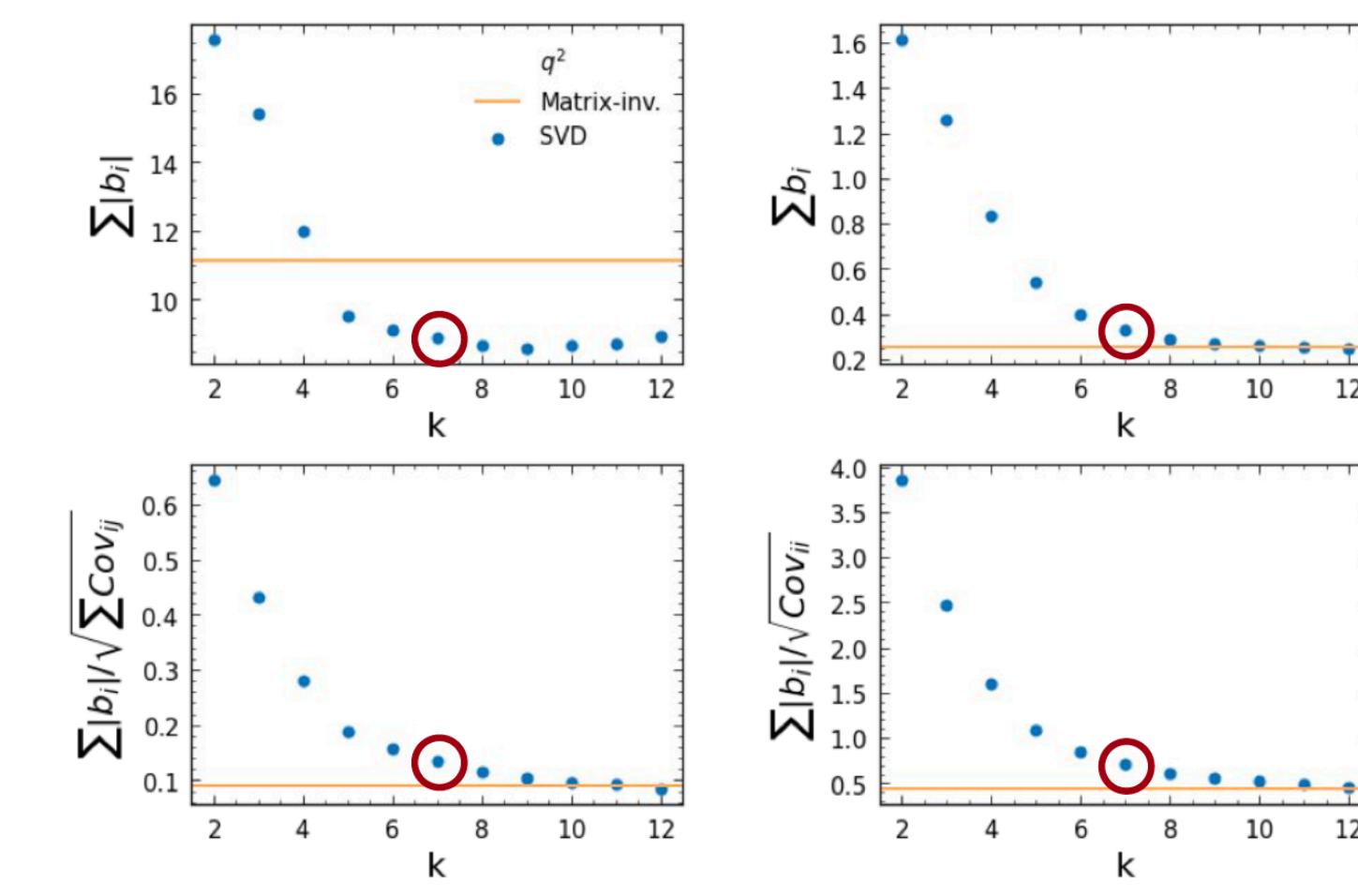


- Bias FOM evaluated for tuning  $k$  in SVD method for each kin. variable
- Matrix-inv result shown as comparison
- Good  $k$  should be satisfied in all tests

$M_X$  in Test 1

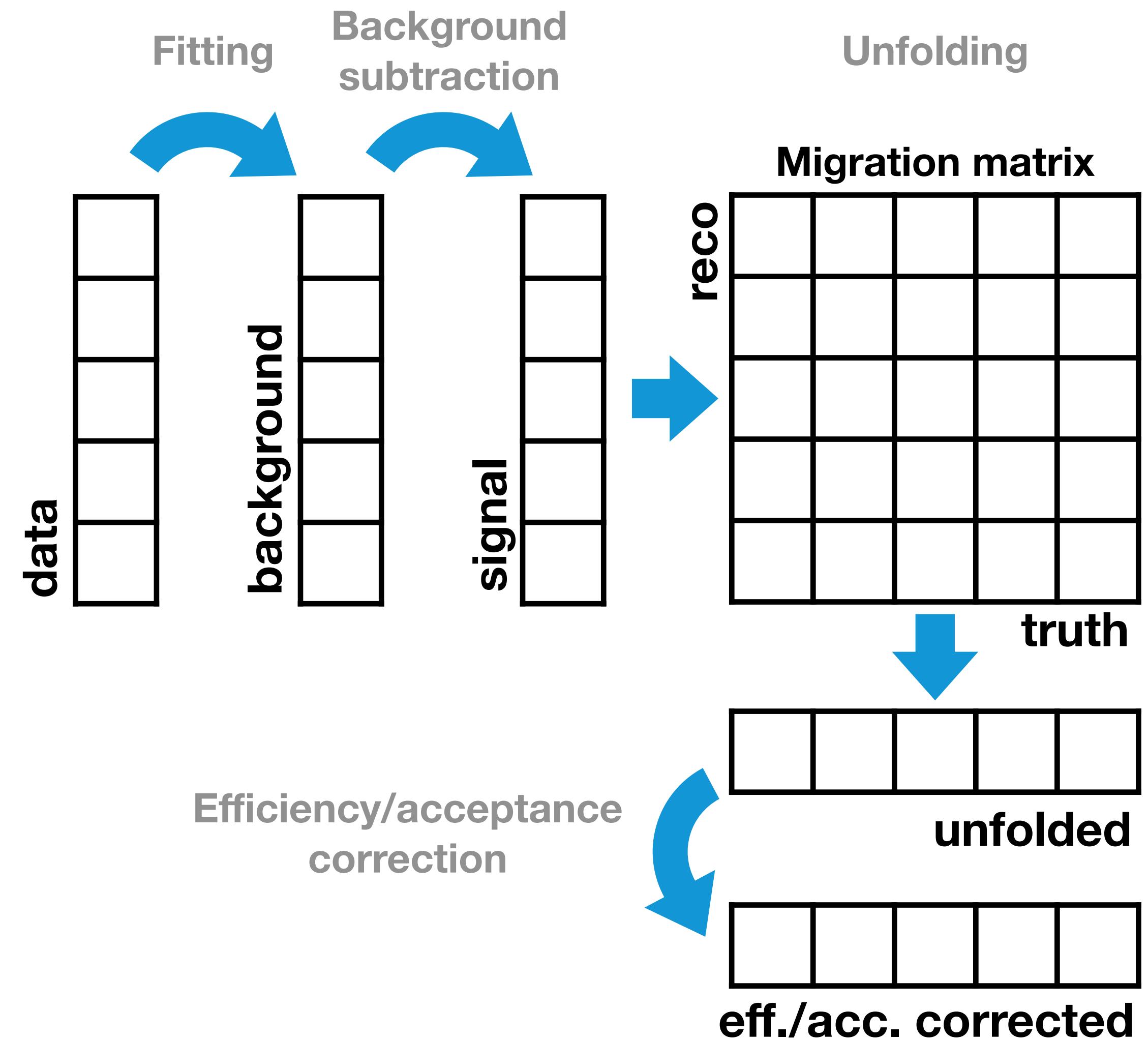


$q^2$  in Test 1



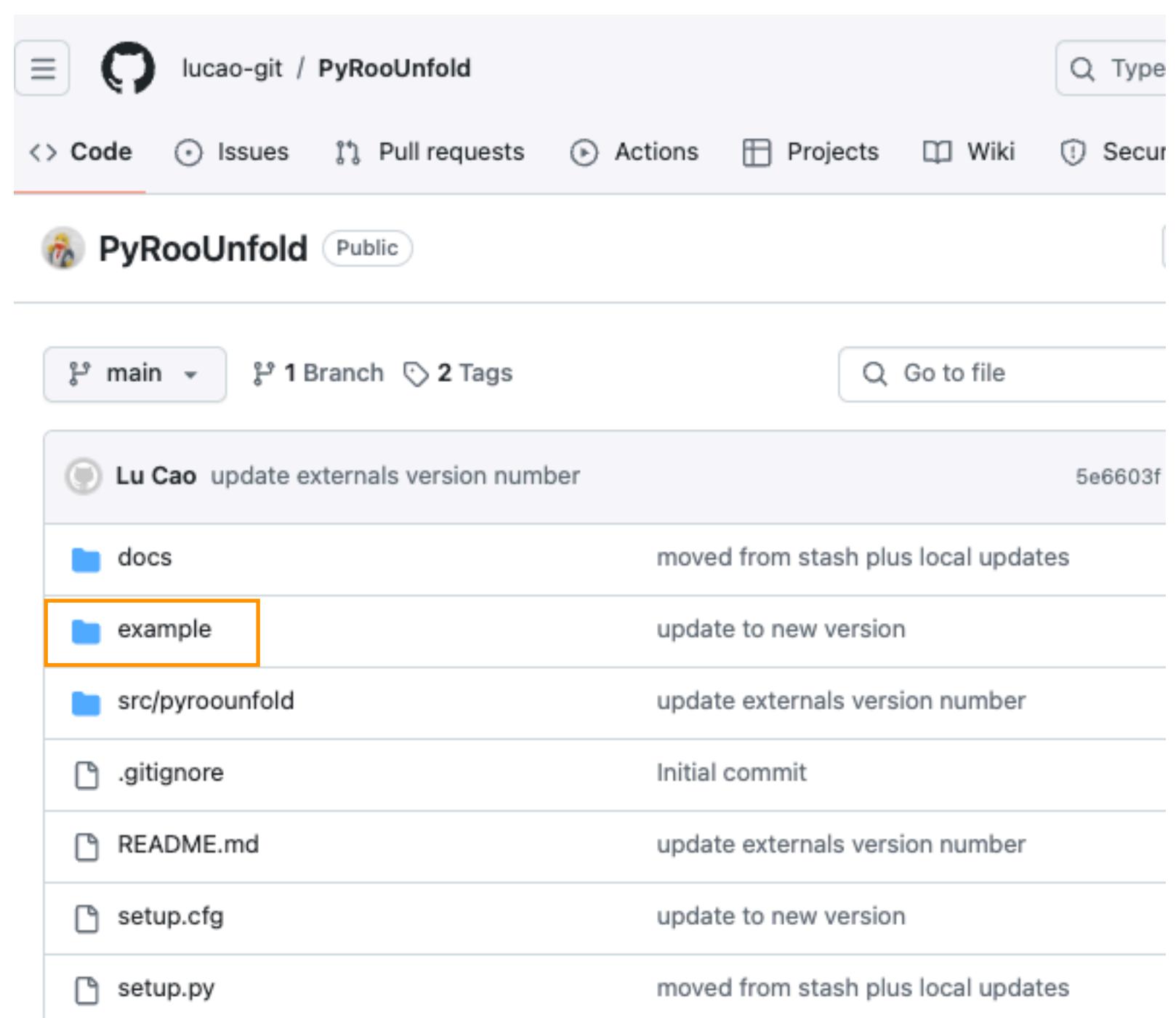
# About Error Propagation

- The final uncertainty of unfolded result is a combination of
  - ▶ Propagated pre-unfolding uncertainty (resulted from background subtraction)
  - ▶ Variations of unfolded results considering modelling impacts of migration matrix
  - ▶ Bias due to choice of method and regularisation parameters
- Experimental systematic sources need to be varied simultaneously for unfolding migration matrix and efficiency correction, preferably also in the procedure of background subtraction (e.g. RooFitUnfold)
- Except bin-by-bin unfolding, most methods handle bin correlation according to regularisation or iteration parameter. Post-unfold correlation (covariance) should be checked



# Repository

- Source code provided on <https://github.com/lucao-git/PyRooUnfold>
- Demo Jupyter notebook can be found in [./example](#)



## Install

- Please make sure RooUnfold has been installed before using PyRooUnfold.

If not, please first follow the steps to install RooUnfold

<https://gitlab.cern.ch/RooUnfold/RooUnfold/-/blob/master/README.md>

or, use a pre-installed RooUnfold (e.g. from CVMFS).

- After installing RooUnfold, please set the environment variable for your RooUnfold library as

```
export ROOUNFOLD_PATH="/path/for/your/libRooUnfold.so"
```

e.g. in my case

```
export ROOUNFOLD_PATH="/Users/caolu/Workspace/RooUnfold/libRooUnfold.so"
```

or using CVMFS version provided as externals of Belle-II software

```
export ROOUNFOLD_PATH="/cvmfs/belle.cern.ch/el9/externals/v02-02-01/Linux_x86_64/common/lib/li
```

- To install PyRooUnfold, you can first check out the needed version, e.g.

```
git checkout -b v2.0.0
```

and then install the package via

```
pip install .
```

If install via setup.py, you need to include the package into your python library

```
export PYTHONPATH=$PYTHONPATH:<path-of-PyroUnfold>
```

## User Guide

- A hands-on introduction and a simple demo Jupyter notebook can be found in ./example folder.