technische universität
dortmund

Combining (correlated) measurements with EFTfitter.jl

Cornelius Grunwald

TU Dortmund

September 2, 2021

What is EFTfitter ?

- tool for combining & interpreting of measurements in a user-friendly way
- for measurements of the same or of different observables
- uses a Bayesian approach for inference on model parameters & uncertainty propagation
- relies on BAT.jl & provides access to the full posterior distributions of the model parameters
- emphasis on correct statistical treatment of uncertainties & correlations
- allows the implementation of user-defined models & the formulation of physical constraints on observables and model parameters
- well suited for EFT interpretations (but not restricted to this field of applications)

Bayesian Analysis Toolkit

EFTfitter paper: https://link.springer.com/article/10.1140/epjc/s10052-016-4280-9

Combination of measurements & goals of EFTfitter.jl

➤ most accurate combination of measurements is achieved through the combination of likelihoods

But:

- likelihood-level combination usually needs information that is not publicly available but only inside the experimental collaborations
- can be technically complicated due to a lot of data, different frameworks for likelihood calculation, etc.

➡ need good approximative approaches for combining measurements in a simpler way

EFTfitter.jl:

- allows straightforward combination from minimal set of information (measured values, uncertainties, correlations)
- uses only one assumption: *measurements are Gaussian*, on that basis: thorough statistical treatment

Statistical principles of EFTfitter

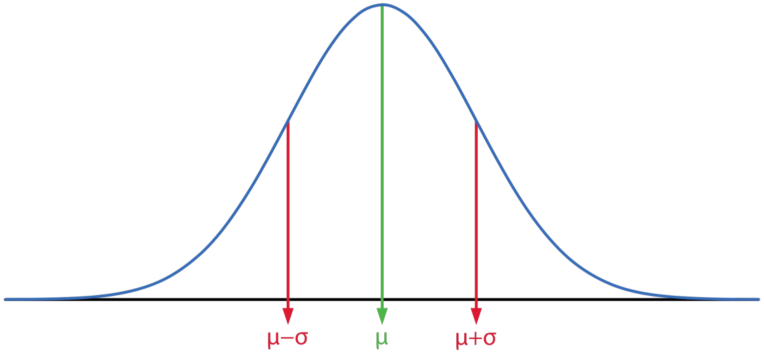
- EFTfitter uses Bayesian inference for updating knowledge about model parameters:

$$\underbrace{P(\lambda|D)}_{\text{posterior}} \propto \underbrace{P(D|\lambda)}_{\text{likelihood}} \cdot \underbrace{P(\lambda)}_{\text{prior}}$$

λ: parameters D: data

- only assumption within EFTfitter:

  Measurements are Gaussian



- multivariate normal distribution:

$$\mathcal{N}(\mu, \mathcal{M}) \propto \exp\left[-(x-\mu)^{\top}\mathcal{M}^{-1}(x-\mu)\right]$$

EFTfitter likelihood

measurements

number of measurements

$$\ln L(\vec{x}|\vec{y}(\vec{\lambda})) = \sum_{i=1}^{n}\sum_{j=1}^{n}[\vec{x} - U\vec{y}(\vec{\lambda})]_i \mathcal{M}_{ij}^{-1}[\vec{x} - U\vec{y}(\vec{\lambda})]_j$$

observables

model parameters λ (to be fitted)

matrix that maps measurements to the corresponding predictions:

U = 1 if x is a measurement of y
U = 0 otherwise

Example:

x: measured cross sections

λ: Wilson coefficients

y: predicted cross sections as a function of the Wilson coefficients

EFTfitter likelihood

number of measurements

measurements

$$\ln L(\vec{x}|\vec{y}(\vec{\lambda})) = \sum_{i=1}^{n} \sum_{j=1}^{n} [\vec{x} - U\vec{y}(\vec{\lambda})]_i \boxed{\mathcal{M}_{ij}^{-1}} [\vec{x} - U\vec{y}(\vec{\lambda})]_j$$

observables

model parameters λ (to be fitted)

matrix that maps measurements to the corresponding predictions:

U = 1 if x is a measurement of y
U = 0 otherwise

$$\boxed{\text{covariance matrix:} \quad \mathcal{M}_{ij}} = \mathrm{cov}[x_i, x_j] = \sum_{k=1}^{M} \mathrm{cov}^{(k)}[x_i, x_j]$$

sum over all *M* categories of uncertainties (e.g. stat., syst., ...)

EFTfitter likelihood

number of measurements

measurements

$$\ln L(\vec{x}|\vec{y}(\vec{\lambda})) = \sum_{i=1}^{n}\sum_{j=1}^{n}[\vec{x} - U\vec{y}(\vec{\lambda})]_i \mathcal{M}_{ij}^{-1}[\vec{x} - U\vec{y}(\vec{\lambda})]_j$$

observables

model
parameters λ
(to be fitted)

matrix that maps measurements to the corresponding predictions:

U = 1 if x is a measurement of y
U = 0 otherwise

combining multiple measurements of the same observable:

- y(λ) = λ, i.e. the fit parameter is directly observable
- when using flat priors ⇨ same results as the best linear unbiased estimator (BLUE)

(http://cds.cern.ch/record/183996/files/OUNP-88-05.pdf?subformat=pdfa&version=1)

Which EFT model is included in EFTfitter ?

➤ None

- there is no specific EFT model implemented in EFTfitter
- models enter through the user-defined observable functions $y(\lambda)$, which specify how an observable depends on the model parameters
- predictions for the observables as a function of the Wilson coefficients need to be determined before using EFTfitter
- this allows full flexibility for custom models, coming from various approaches
  (e.g. MC computations + approximations, or theory calculations)
- it allows to account for model-specific efficiency and acceptance corrections for the measurements or the presence of physical constraints on observables

EFTfitter.jl - API Example

1) define parameters + priors

```
parameters = BAT.NamedTupleDist(
    C1 = -3..3, # short for: Uniform(-3, 3)
    C2 = Normal(0, 0.5) # Normal distribution
)
```

2) implement functions for calculating observables as a function of the parameters

```
function xsec1(params)
    c = [20.1, 5.6, 325.5, ....]
    return c[1]*params.C1 + c[2]*params.C1*params.C2
          + c[3]*params.C2 + ...
end
...
```

3) specify measurements (values + uncertainties)

```
measurements = (
    Meas1 = Measurement(xsec1, 21.6,
            uncertainties = (stat=0.8, syst=1.8, another_unc=2.3), active=true),

    Meas2 = Measurement(Observable(xsec2, min=0), 1.9,
            uncertainties = (stat=0.6, syst=0.9, another_unc=1.1), active=true),

    MeasDist = MeasurementDistribution(diff_xsec, [1.9, 2.93, 4.4],
               uncertainties = (stat = [0.7, 1.1, 1.2], syst= [0.7, 0.8, 1.3],
               another_unc = [1.0, 1.2, 1.9]), active=[true, false, true]),
)
```

4) specify correlations for each type of uncertainty

```
correlations = (
    stat = NoCorrelation(active=true),

    syst = Correlation([1.0 0.5;
                        0.5 1.0 ], active=false),

    another_unc = Correlation(another_corr_matrix, active=true)
)
```

```
another_corr_matrix = to_correlation_matrix(measurements,
    (:Meas1, :Meas2, 0.4),
    (:Meas1, :MeasDist_bin1, 0.1),
    (:MeasDist, :MeasDist, [1 0.2; 0.2 1.0]),
    (:MeasDist_bin2, :MeasDist_bin3, 0.3),
)
```

EFTfitter.jl - API Example

5) create a model from the inputs:

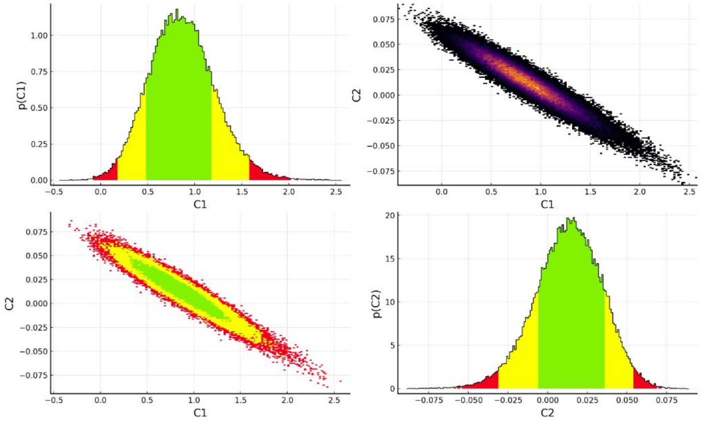
```julia
model = EFTfitterModel(parameters, measurements, correlations)
# use EFTfitter posterior:
posterior = PosteriorDensity(model)
```

6) sample the posterior using BAT.jl

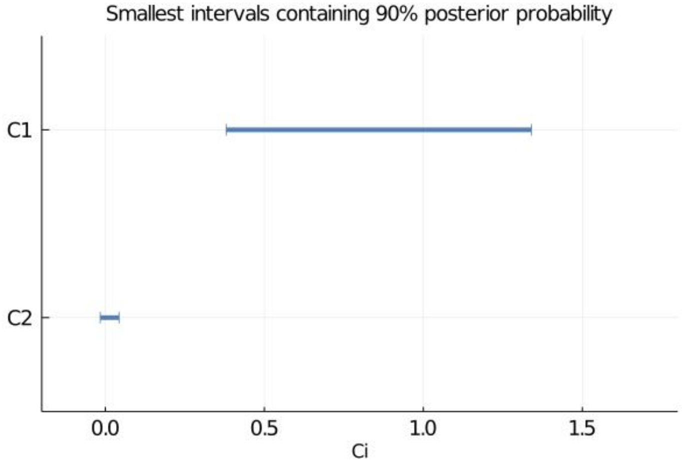
```julia
algorithm = MCMCSampling(mcalg = MetropolisHastings(), nsteps = 10^6, nchains = 4)
samples = bat_sample(posterior, algorithm).result
```

7) plot samples, print estimates, etc...

```julia
plot(samples)
```




```julia
plot(samples, 0.9)
```



Back up

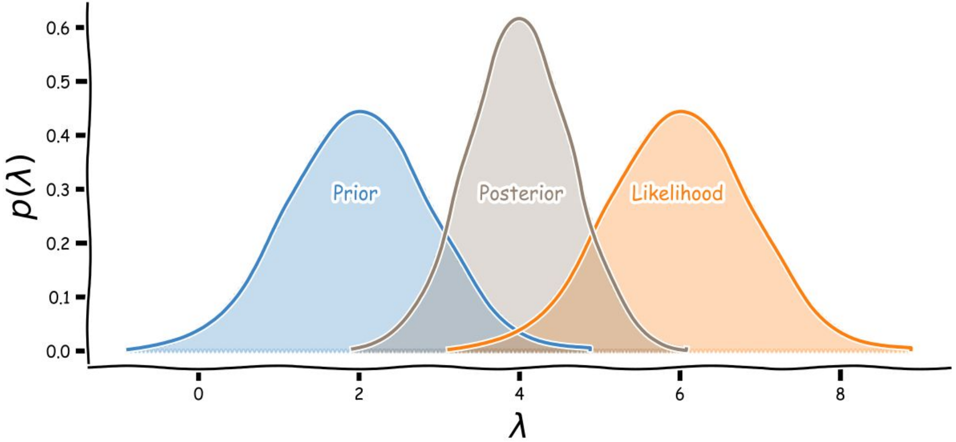
What is BAT.jl ?

- toolkit for performing Bayesian inference in a user-friendly way
- collection of algorithms & functions for solving user-specified problems, without relying on a specific modelling language / domain specific language
- focusing on sampling custom posterior distributions (particularly via MCMC methods)
- further functionalities for Bayesian analyses: integration & marginalization, optimization & parameter estimation, limit setting, model comparison, goodness-of-fit tests

Bayes' Theorem: (simple on paper, but numerics are hard)

$$P(\lambda|D) = \frac{P(D|\lambda)P(\lambda)}{\int P(D|\lambda)P(\lambda)\ \mathrm{d}\lambda}$$

D - data λ - parameters



The Bayesian Analysis Toolkit

- first released in 2008
- based on C++ & ROOT
- BAT paper: Comput. Phys. Commun. 180 (2009) 2197
- latest release v1.0 in 2018

  https://github.com/bat/bat/releases/tag/v1.0.0

- rewrite in Julia, independent of ROOT
- first released in 2019
- BAT.jl paper on arxiv: 2008.03132
- v2.0 of BAT.jl released in December

  https://github.com/bat/BAT.jl

goals of rewrite:

open BAT for users beyond the realm of particle physics (reduce domain-specific dependencies),

implement modern sampling approaches and algorithms, simplify parallelization & distribution

- BAT.jl already used in scientific analyses: Legend, EFT fits, COVID-19 lethality

Technical details of EFTfitter

- Bayesian inference requires dedicated algorithms for computation of posterior distributions (e.g. MCMC sampling & integration algorithms)
- EFTfitter uses ***BAT - The Bayesian Analysis Toolkit*** for sampling the posterior distributions
- EFTfitter originally in C++ (depending on C++ Version of BAT & ROOT)
- new version of BAT in Julia programming language:

https://github.com/bat/BAT.jl

- therefore: new version of EFTfitter in Julia to exploit all advantages of the new BAT.jl (e.g. new sampling & integration algorithms, parallelization, ...)
- contains all previous features + some improvements, e.g. simpler input of measurements, distributions & large correlation matrices, ranking of measurements / uncertainty categories
- super easy to install, setup & use

Bayes' Theorem:

(simple on paper, but numerics are hard)

$$P(\lambda|D) = \frac{P(D|\lambda)P(\lambda)}{\int P(D|\lambda)P(\lambda)\, \mathrm{d}\lambda}$$

λ - parameters   D - data

Installation of Julia, BAT.jl & EFTfitter.jl:

```
#download & unzip Julia
$ wget https://julialang-s3.julialang.org/bin/linux/x64/1.5/julia-1.5.3-linux-x86_64.tar.gz
$ tar -xvzf julia-1.5.3-linux-x86_64.tar.gz
$ cd julia-1.5.3/bin

# run Julia
$ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.5.3 (2020-11-09)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

# install EFTfitter.jl & BAT.jl (only on first use)
julia> using Pkg
julia> pkg"add https://github.com/tudo-physik-e4/EFTfitter.jl BAT"

# use EFTfitter.jl & BAT.jl
julia> using EFTfitter, BAT
```

Summary

- EFTfitter allows a straightforward combination & EFT interpretation of measurements from minimal information
- emphasis is placed on the correct treatment of correlated uncertainties
- high flexibility for including different EFT models, formulation of physical constraints on observables and parameters
- easy installation, human-readable input formats, simple usage of multiple features

➡ EFTfitter well suited for global EFT fits (without likelihood-level combination)

check it out at: https://github.com/tudo-physik-e4/EFTfitter.jl

or try it right now on binder: launch binder

EFTfitter paper: https://link.springer.com/article/10.1140/epjc/s10052-016-4280-9

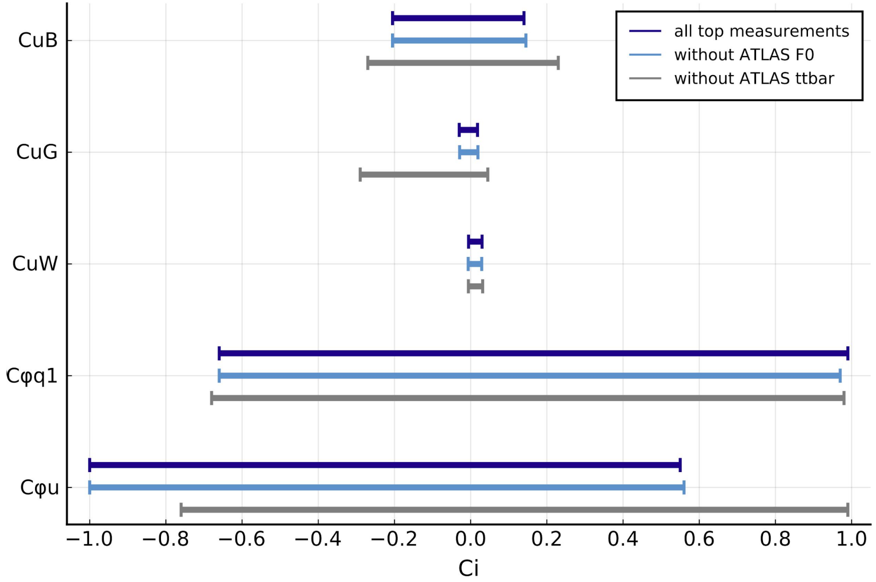
C++ EFTfitter: https://github.com/tudo-physik-e4/EFTfitterRelease

Influence of the individual measurements / uncertainty types

- BLUE assigns weights to the individual measurements included in the combination
- we try to estimate the influence of a measurement by considering its impact on the size of the posterior distribution
- idea: remove one measurement from the combination, see how the size of the smallest intervals/HDR changes
- expectation: when removing information, the result is more uncertain and the interval size should increase

Example:

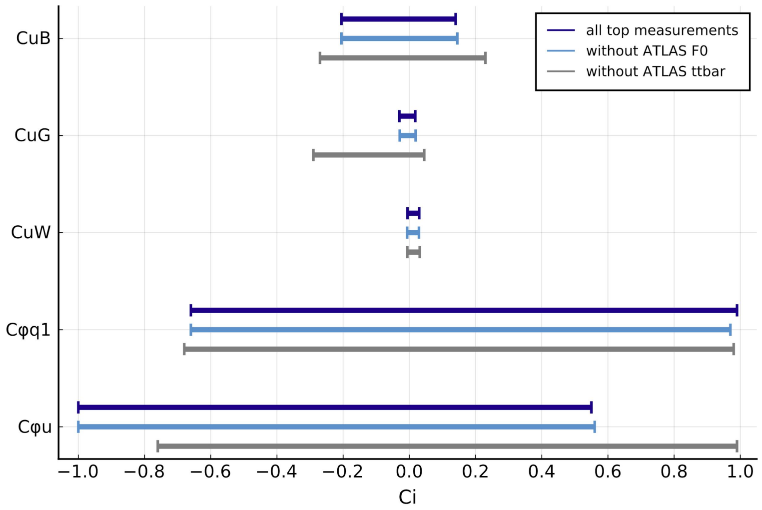
top-quark production & decay measurements from the top-quark & B physics combination study [2012.10456]

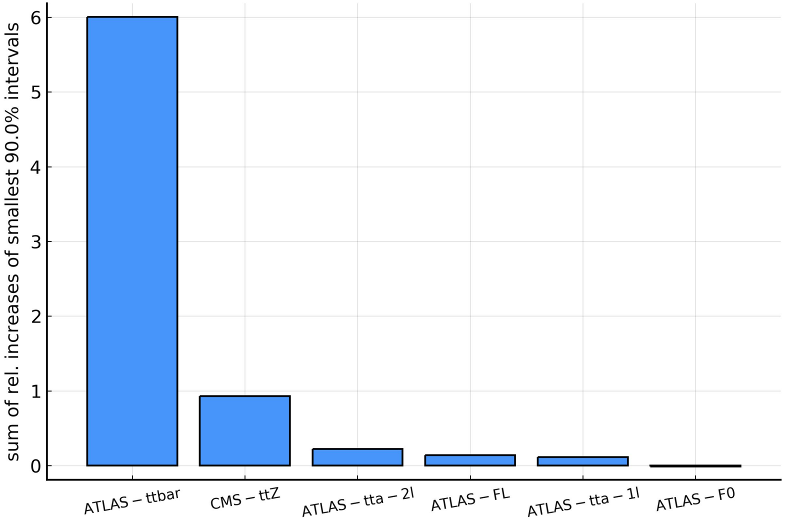


smallest intervals containing 90% posterior probability

Ranking example

```
measurement_ranking = EFTfitter.rank_measurements(model, criterion = SumOfSmallestIntervals(p=0.9))
```





- use relative increase of smallest p% intervals/areas as ranking criterion (default: sum over all rel. increases of 1d smallest intervals)
- for ranking of uncertainty categories: use relative decrease
- for contradicting measurements: posterior interval could also shrink when excluding one measurement (e.g. “outlier”)