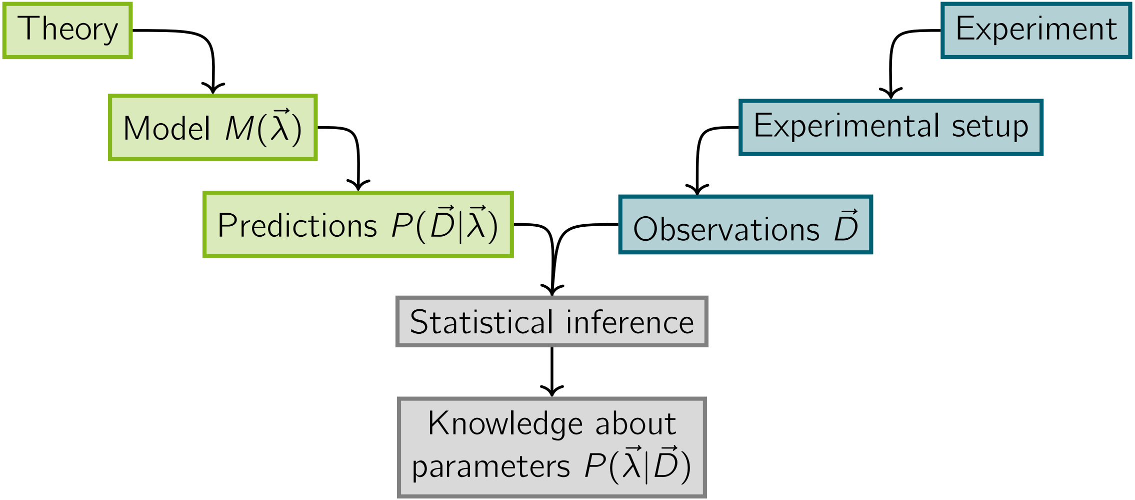
The Bayesian Analysis Toolkit and applications

Cornelius Grunwald[1], Kevin Kröninger[1],
Romain Madar[2], Stéphane Monteil[2], Lars Röhrig[1,2]

June 28, 2022

[1]Department of Physics – TU Dortmund University
[2]Laboratoire de Physique de Clermont – Université Clermont-Auvergne

What it's all about and who am I?



What is BAT.jl

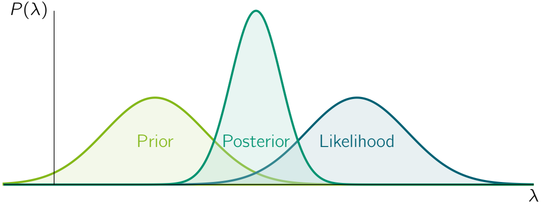
- Toolkit for performing Bayesian inference in a user-friendly way
- Provides a collection of algorithms and methods without relying on specific modeling language
- Focusing on sampling custom posterior distributions
- Further functionalities:
  - → Integration & marginalisation
  - → Optimisation & parameter estimation
  - → Limit setting, model comparison, goodness-of-fit tests

Bayes' Theorem

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

D: data

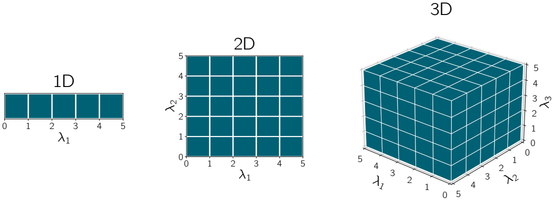
λ: parameters

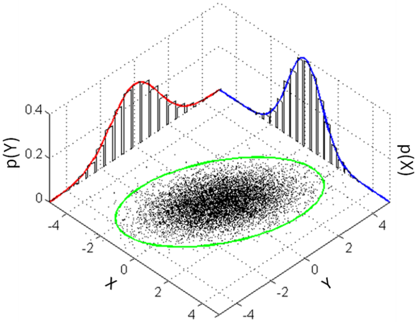


The need for numerical methods

The curse of dimensionality:

- In high-dimensional parameter spaces: unfeasible to evaluate the posterior at all points of the sampling-space

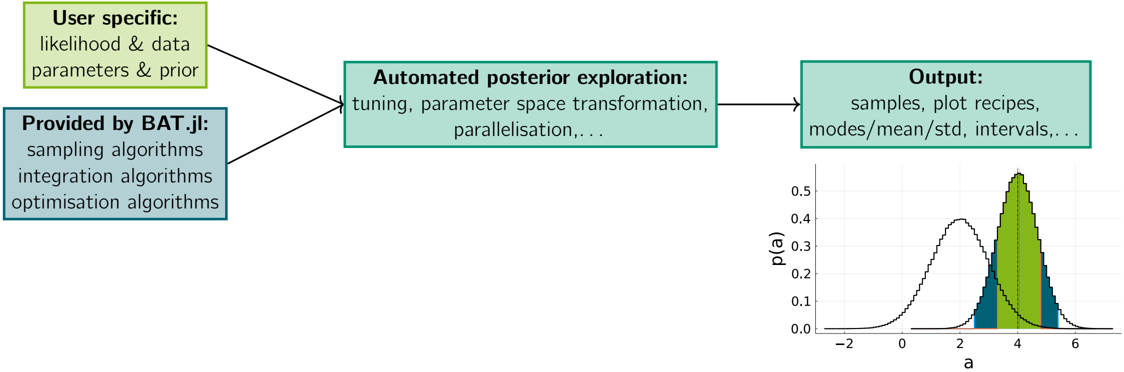




- Efficient algorithms for high-dimensional
  - → . . . sampling
  - → . . . optimatisation
  - → . . . integration

BAT.jl – The Bayesian Analysis Toolkit

- Originally developed in `C++`, but not maintained anymore
- `BAT.jl`: Rewrite in `Julia` programming language, first release in 2019
- Modern software package with toolkit-like character for easy expansion



User specific:
likelihood & data
parameters & prior

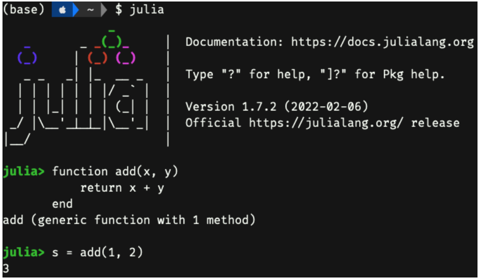
Provided by BAT.jl:
sampling algorithms
integration algorithms
optimisation algorithms

Automated posterior exploration:
tuning, parameter space transformation, parallelisation,. . .

Output:
samples, plot recipes, modes/mean/std, intervals,. . .

The Julia language

- Language designed for high-performance and efficient numercial computing
- First launch 2012 after 3 years of development at MIT
- `v1.0` in 2018, current `v1.7.3`
- Solved the two-language-problem:
    - As comfortable as `python`
    - As fast as `C++`
- Key features: dynamic type system, multiple dispatch, parallel & distributed computing, package manager, easy to call Fortran, `C/C++`, `python`,...
- Growing and very scientific community

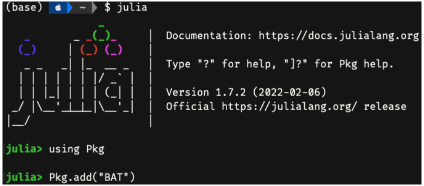


```
(base)  ~ $ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.7.2 (2022-02-06)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> function add(x, y)
           return x + y
       end
add (generic function with 1 method)

julia> s = add(1, 2)
3
```

Learn `julia` here.

Features of BAT.jl

- Use of custom posterior distributions (from user-specific likelihoods & priors)
- Collection of sampling algorithms:
  - MCMC: Metropolis-Hastings, Hamiltonian-MC
  - Importance Samplers
  - Nested Sampling
- Automated initialisation, tuning & convergence testing for MC chains
- Automated parameter space transformations
- Design idea: offer reasonable default settings for easy-to-use, but also allow for fine-grained control



```
(base) ~ $ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.7.2 (2022-02-06)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> using Pkg

julia> Pkg.add("BAT")
```

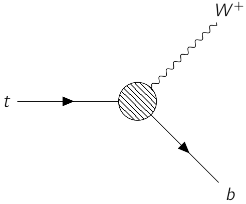
The package is available at GitHub here.

Let's explore together!

You're warmly invited to try yourself with this binder!

For what BAT.jl is used (among others): EFTfitter.jl

- Tool for combining multiple measurements & performing Bayesian inference on the underlying parameters
- Optimised for EFT interpretations of measurements with interface to `BAT.jl`
- SMEFT: higher-dimensional operators extending the SM-Lagrangian, can affect e.g. t-quark decay



Effective coupling at $t \to Wb$.

SMEFT-likelihood

$$\ln\left(\mathcal{L}\left(\vec{x}\middle|\vec{y}(\vec{\lambda})\right)\right) = \overbrace{\sum_{i,j=1}^{n}}^{\#\text{ measurements}} \underbrace{\left[\vec{x}-\vec{y}(\vec{\lambda})\right]_i}_{\text{distance between measured \& predicted values}} \overbrace{\mathcal{M}_{ij}^{-1}}^{\text{covariance matrix}} \left[\vec{x}-\vec{y}(\vec{\lambda})\right]_j$$

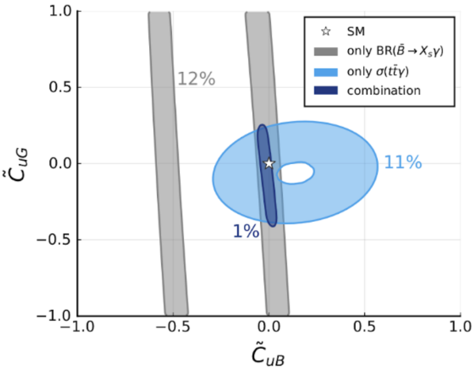
x: measured σ

λ: Wilson coefficients

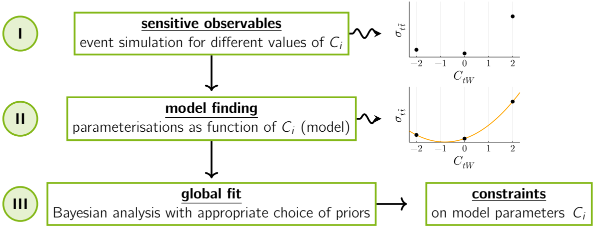
y: pred. $\sigma(\lambda)$

For what BAT.jl is used (among others): EFTfitter.jl

- Simulate predicted cross-sections with the Monte-Carlo generator `MadGraph`
- Parameterise the **simulations** according to $y(\lambda)$
- Compare with **measurements** from HEP collaborations, e.g. ATLAS, BELLE, ...
- Choosing flat priors on the **model parameters** to constrain, e.g. C_{uG}, C_{uB} and C_{uW}



Fit of t- and b-physics observables [1].



sensitive observables
event simulation for different values of C_i

model finding
parameterisations as function of C_i (model)

global fit
Bayesian analysis with appropriate choice of priors

constraints
on model parameters C_i

Conclusions and how to get started...

- BAT software highly efficient in `julia` programming language and provides a variety of algorithms for sampling, optimsation and integration
- Known weaknesses of existing sampling algorithms + modularity = room for contributions

- Package as GitHub repository available
- Full (API) documentation, as well as tutorials
- Implementation of new sampling algorithms in modular-like fashion, see e.g. different importance sampler

Conclusions and how to get started...

- BAT software highly efficient in `julia` programming language and provides a variety of algorithms for sampling, optimsation and integration
- Known weaknesses of existing sampling algorithms + modularity = room for contributions

- Package as GitHub repository available
- Full (API) documentation, as well as tutorials
- Implementation of new sampling algorithms in modular-like fashion, see e.g. different importance sampler

Thanks a lot for your attention!