

A Python Package for Parameter Estimation and Uncertainty Quantification

Jakob Vanhoefer, Yannik Schälte, Jan Hasenauer
+ pyPESTO developers

What is the focus of pyPESTO?

What is the focus of pyPESTO?

- Optimization and uncertainty quantification
- Simple & unified interface to various optimizers/samplers

What is the focus of pyPESTO?

- Optimization and uncertainty quantification
- Simple & unified interface to various optimizers/samplers
- Jupyter notebook & slides :  /  jvanhoefer
 - <https://github.com/jvanhoefer/COMBINE-2020-talk>

Objective Functions in pyPESTO

User Defined Objective Functions

```
# define objective function
def f(x: np.array):
    return x[0]**2 + x[1]**2

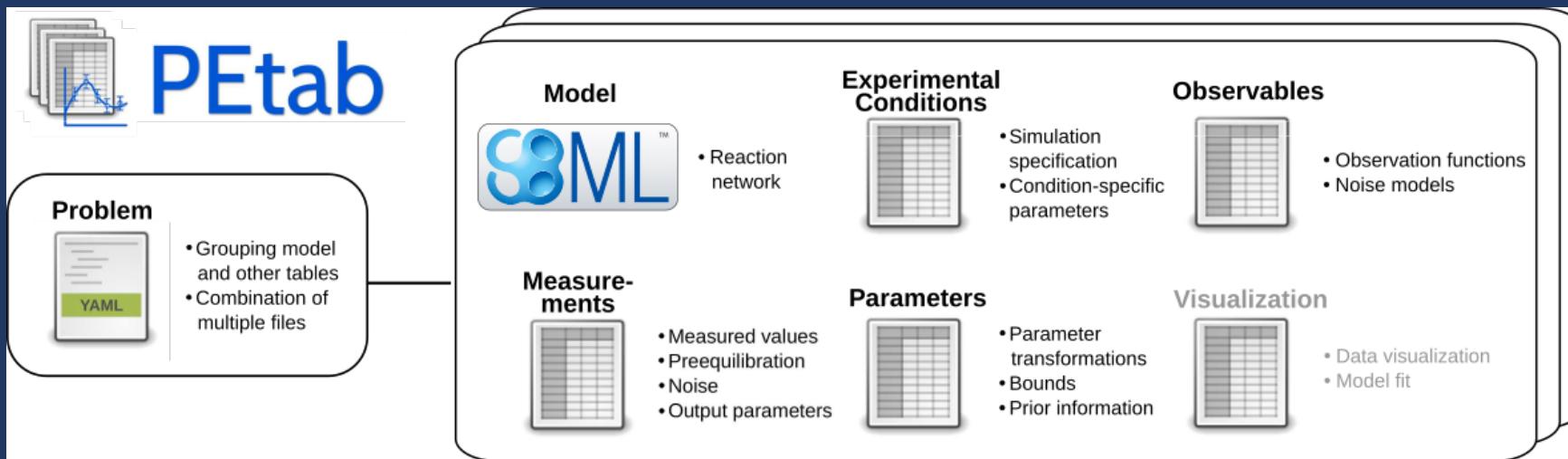
# define gradient
def grad(x: np.array):
    return 2*x

# define objective
custom_objective = pypesto.Objective(fun=f, grad=grad)
```

Objective = negative-log-likelihood/posterior

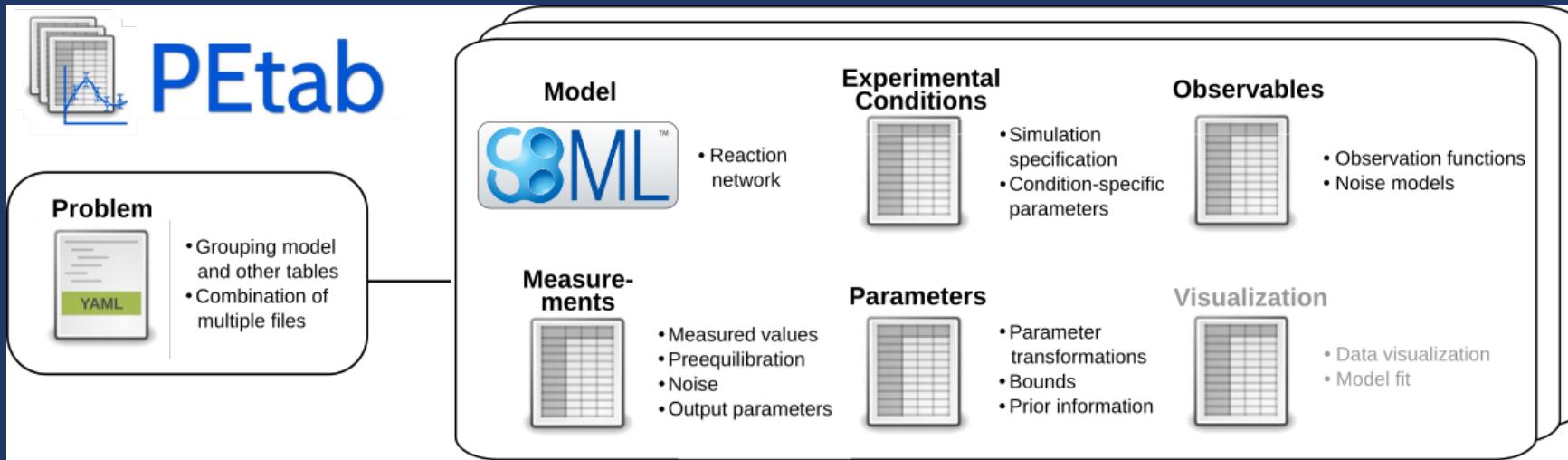
Objective Functions from PEtab

- data format for specifying parameter estimation problems



Objective Functions from PEtab

- data format for specifying parameter estimation problems



- PEtab-dev/PEtab
- Objective = negative log-likelihood/log-posterior
- Model simulation → AMICI

Talk by Daniel Weindl
Thu 21:45



Model Import and Optimization via PEtab

Import

```
# directory of the PEtab problem
petab_yaml = './Boehm_JProteomeRes2014/Boehm_JProteomeRes2014.yaml'

importer = pypesto.petab.PetabImporter.from_yaml(petab_yaml)
problem = importer.create_problem()
```

Model Import and Optimization via PEtab

Import

```
# directory of the PEtab problem
petab_yaml = './Boehm_JProteomeRes2014/Boehm_JProteomeRes2014.yaml'

importer = pypesto.petab.PetabImporter.from_yaml(petab_yaml)
problem = importer.create_problem()
```

Optimize

```
optimizer = optimize.ScipyOptimizer()

# do the optimization
result = optimize.minimize(problem=problem,
                            optimizer=optimizer,
                            n_starts=10)
```

Optimization in pyPESTO

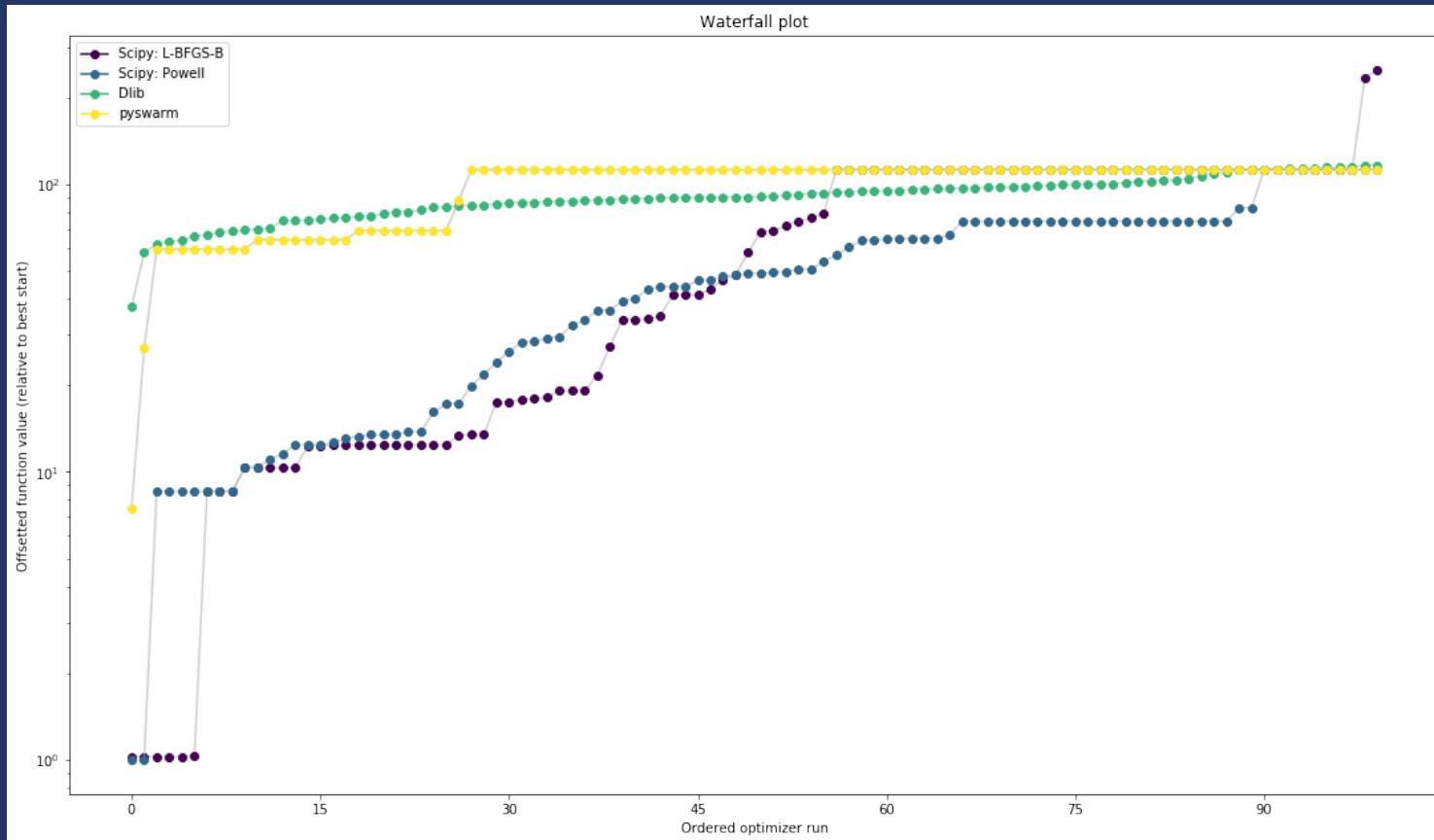
Which optimizers can I use in pyPESTO?

- pyPESTO provides a unified interface to
 - All SciPy-optimizers
 - IpOpt
 - Dlib
 - Particle swarm (pyswarm)
 - CMA-ES

Which optimizers can I use in pyPESTO?

- pyPESTO provides a unified interface to
 - All SciPy-optimizers
 - IpOpt
 - Dlib
 - Particle swarm (pyswarm)
 - CMA-ES
- Change optimizer = one line of code:
`optimizer = optimize.PyswarmOptimizer()`

Optimizer Performance



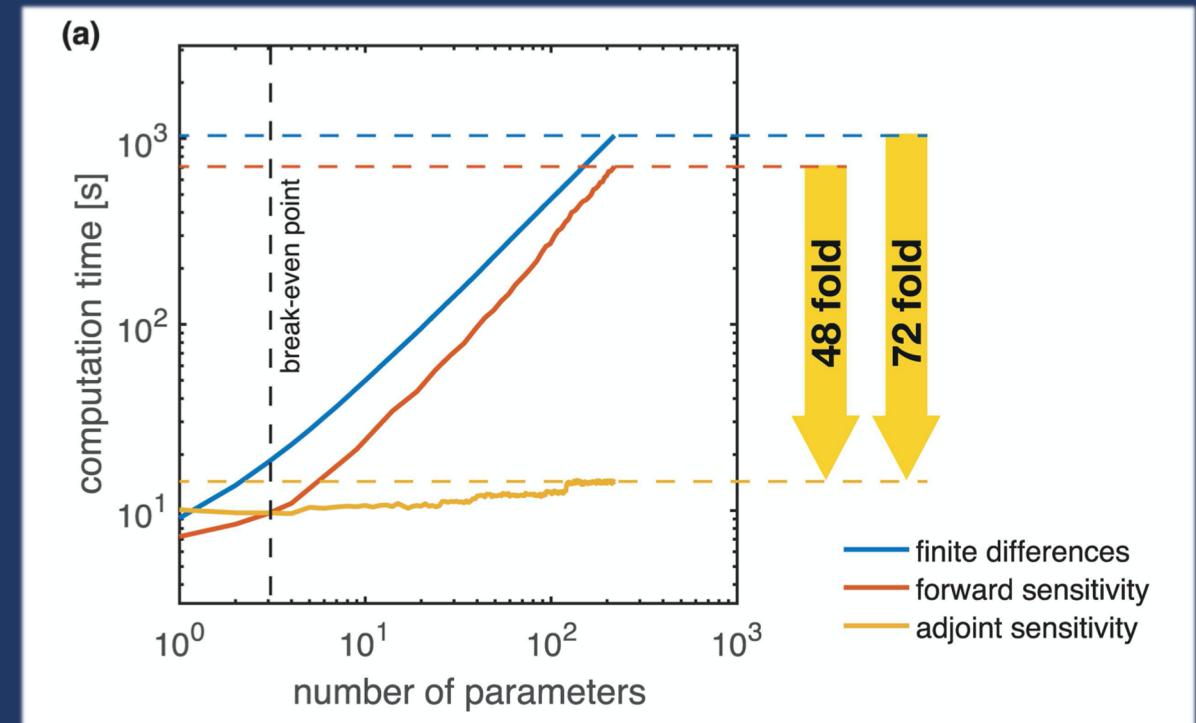
Average Run time per start:

Scipy: L-BFGS-B: 3.327784 s
Scipy: Powell: 18.523545 s
Dlib: 114.969435 s
pyswarm: 26.666935 s

```
pypesto.visualize.waterfall(optimizer_results, legends = optimizer_names)
```

Large Scale Models in pyPESTO

- Fast and reliable gradient computation is key!
⇒ **Adjoint sensitivities (AMICI)**
- Parallelization



Fröhlich et al. PlosCB 2017

Large Scale Models in pyPESTO: Code

```
# Set gradient computation method to adjoint
problem.objective.amici_solver.setSensitivityMethod(amici.SensitivityMethod.adjoint)

# Parallelize
engine = pypesto.engine.MultiProcessEngine()

# Optimize
result = optimize.minimize(problem=
                            optimizer=optimizer_scipy_lbfgsb,
                            engine=engine,
                            n_starts=100)
```

Uncertainty Quantification in pyPESTO

Profile Likelihoods

- Method to compute confidence intervals
- Maximum projection + likelihood ratio test

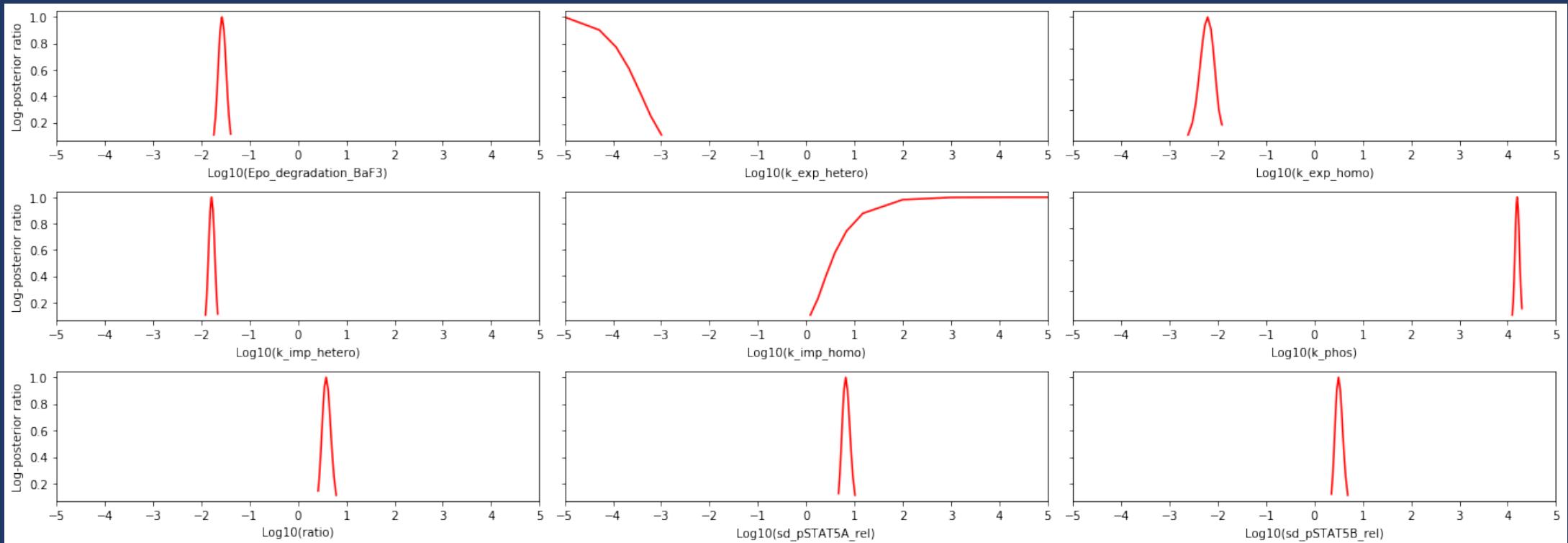
Profile Likelihoods

- Method to compute confidence intervals
- Maximum projection + likelihood ratio test

```
import pypesto.profile as profile

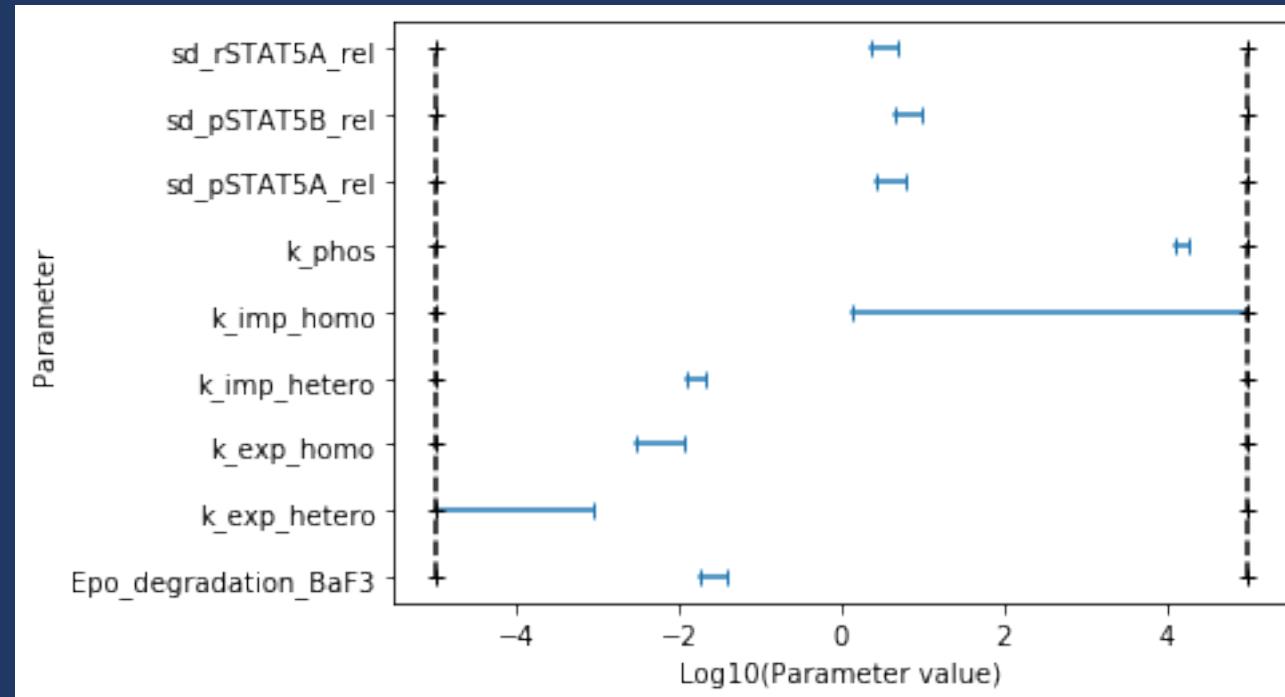
result = profile.parameter_profile(problem=problem,
                                    result=result,
                                    optimizer=optimizer_scipy_lbfgsb)
```

Profile Likelihoods: Projections



```
visualize.profiles(result, x_labels = x_labels, show_bounds=True)
```

Profile Likelihoods: 95% Confidence Intervals



```
pypesto.visualize.profile_cis(result, confidence_level=0.95)
```

(MCMC) Sampling

```
import pypesto.sample as sample

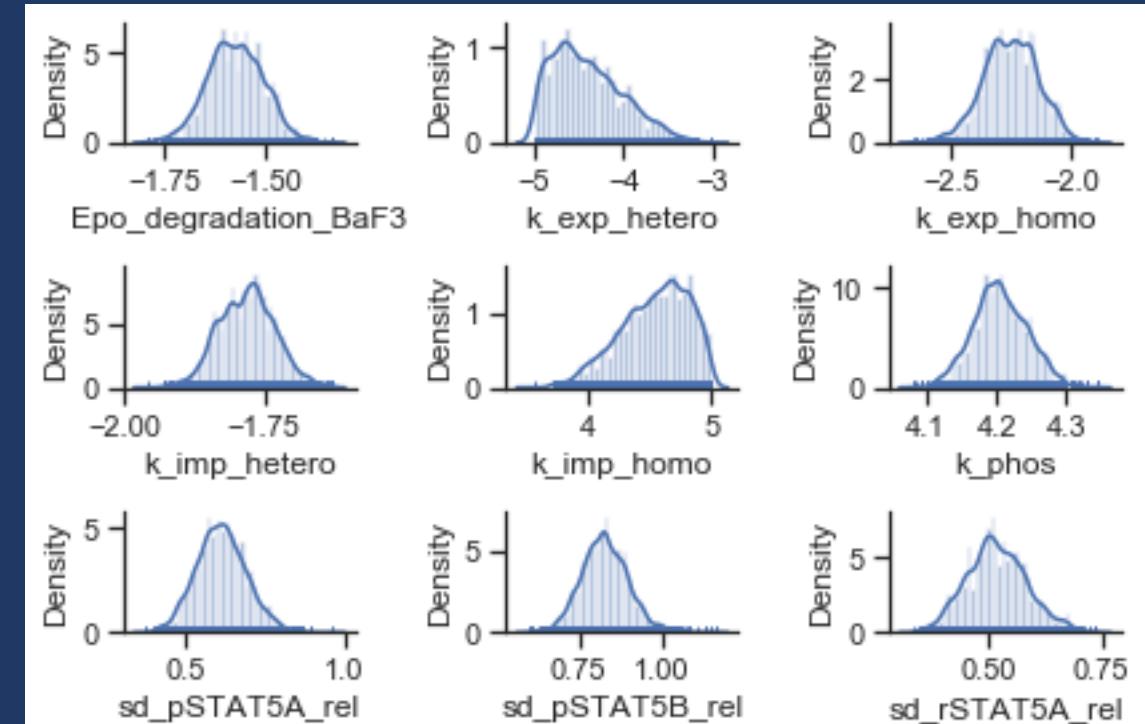
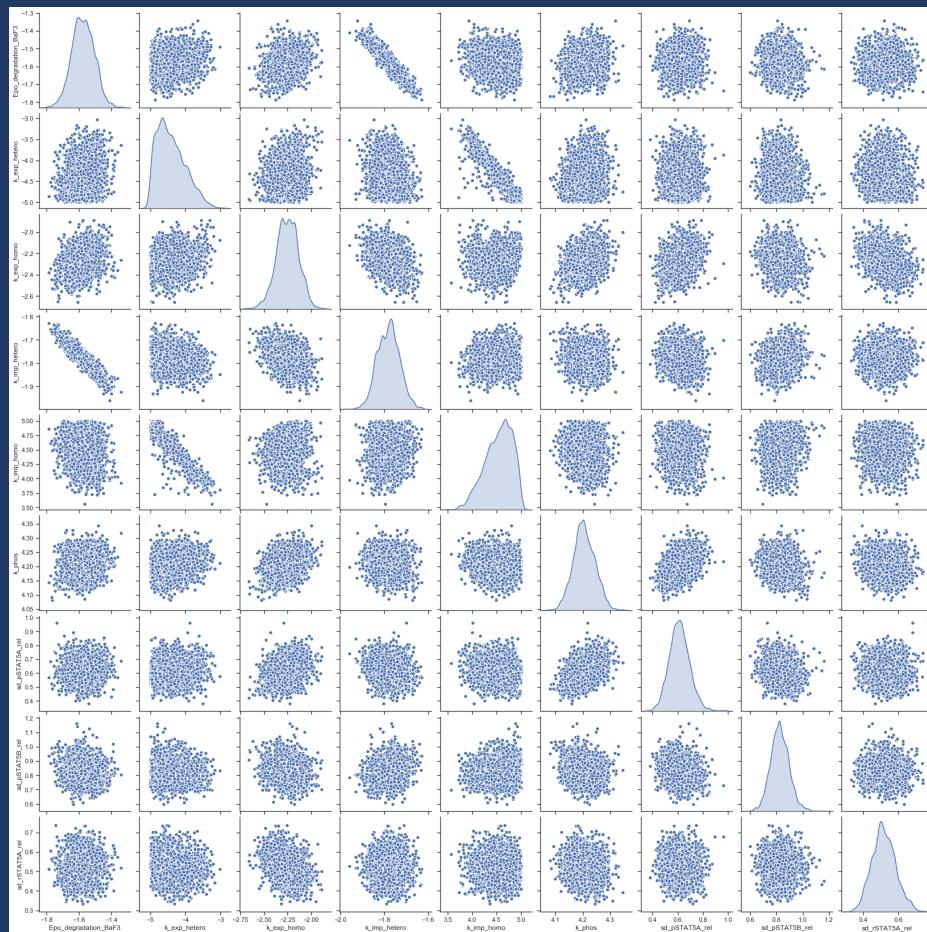
n_samples = 10000

sampler = sample.AdaptiveMetropolisSampler()

result = sample.sample(problem,
                      n_samples=n_samples,
                      sampler=sampler,
                      result=result)
```

100%|██████████| 10000/10000 [00:17<00:00, 561.75it/s]

Sampling: Visualization



Which samplers can I use in pyPESTO?

- pyPESTO implements/interfaces
 - Adaptive Metropolis
 - Adaptive parallel Tempering
 - Interfaces pymc3

Which samplers can I use in pyPESTO?

- pyPESTO implements/interfaces
 - Adaptive Metropolis
 - Adaptive parallel Tempering
 - Interfaces pymc3
- Sampling diagnostics
 - Geweke test
 - effective sample size
 - ...

How is pyPESTO structured?

- Modularity as Design Principle
- objective function
- key parts of analysis/algorithms are interchangeable
- code object oriented and modular

Development & Availability

- Open Source:  ICB-DCM/pyPESTO
- `pip install pypesto`
- Docu (RTD) + example notebooks
- Continuous integration + testing via travis

Development & Availability

- Open Source:  ICB-DCM/pyPESTO
- `pip install pypesto`
- Docu (RTD) + example notebooks
- Continuous integration + testing via travis
- 5 – 10 people using, extending and maintaining pyPESTO
- Contributions and feature requests welcome

Currently under Development

- More Optimizer/Sampler
- Model Selection
- Experimental Design
- Hierarchical optimization of scaling/noise parameters
- Ordinal data



Thank you!