GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

BioQuant
MODEL base of LIFE

Parameter Estimation with COPASI

Frank T. Bergmann

Ursula Kummer

Introduction

- When starting a modeling project usually many parameters of the model are not known
- As we saw earlier, we can see the effect of parameter changes in the model using:
  - Sliders
  - Sensitivities / Metabolic control analysis
  - Parameter scans
- But how do we find parameter values for a given observed effect?

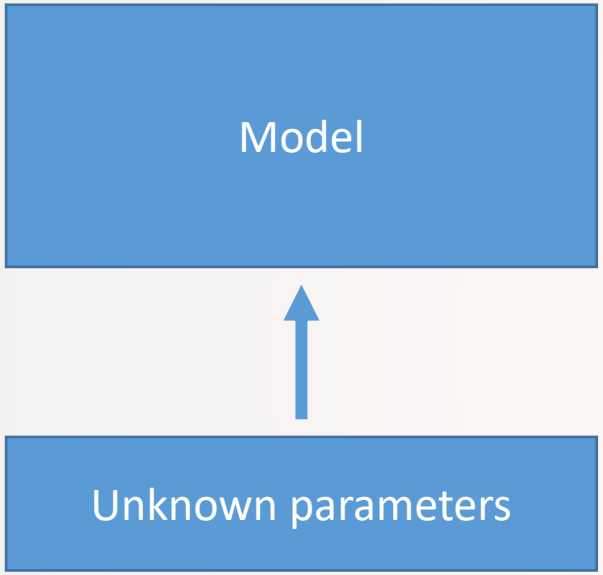
Introduction

- simple approach: try to design an experiment for measuring the specific parameter
  - typically, *in vitro* experiment
  - e.g., for rate constants: put different amounts of substrate in a test tube and measure how fast the reaction proceeds
- Problems:
  - often not possible, different from *in vivo* conditions

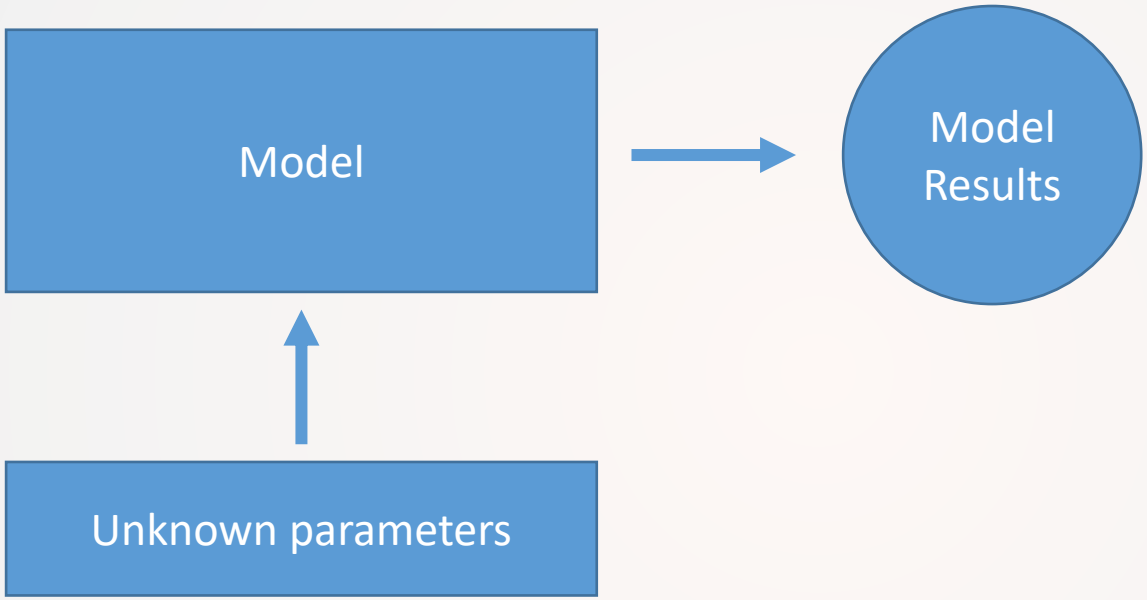
Introduction

- Systems biology approach: adapt a complete model to experimental data
  - indirect method: use the model to find out about a parameter by measuring something else
  - can also be used to answer more complex questions, such as model identification
  - more difficult

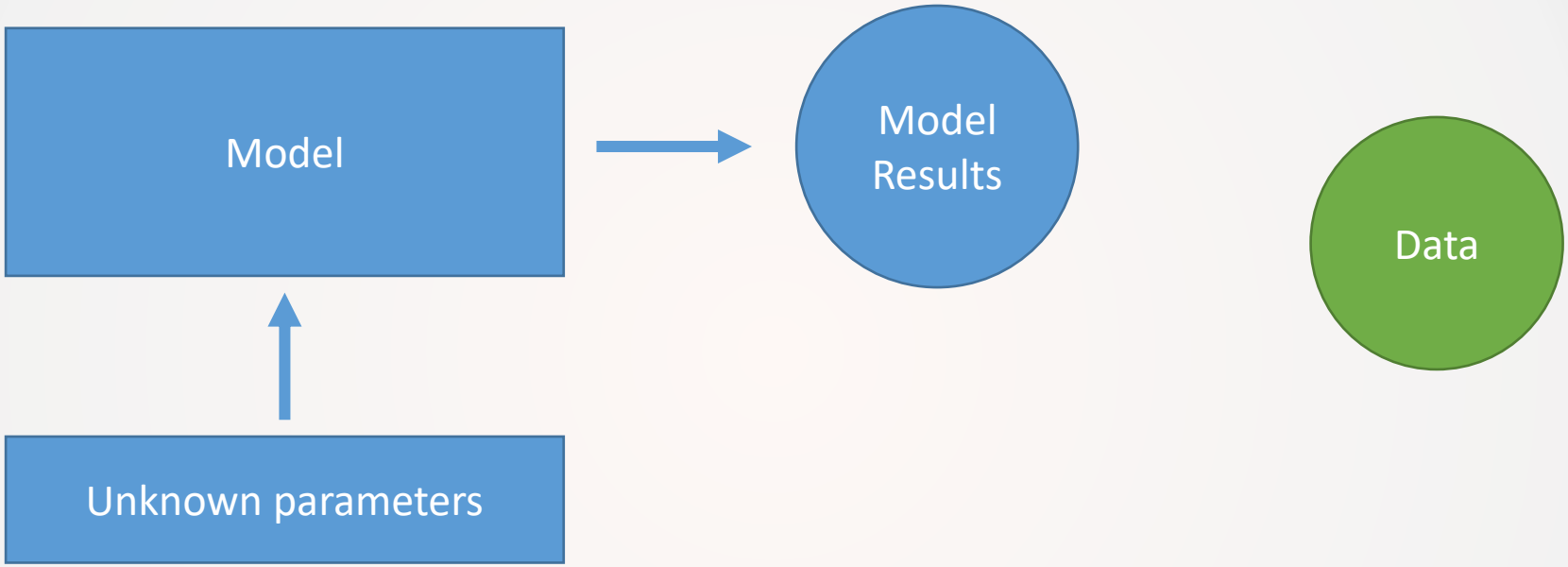
Basic idea of parameter fitting



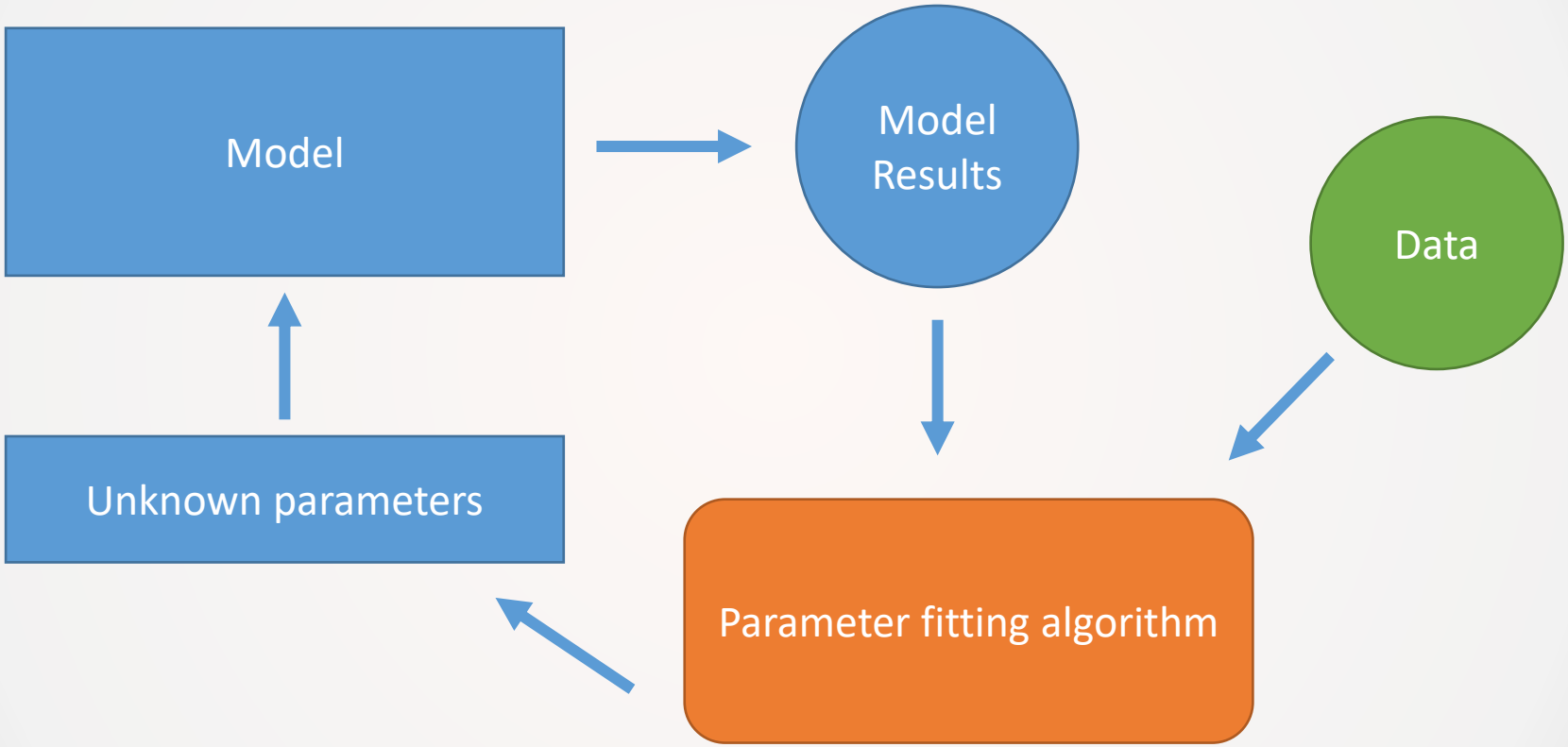
Basic idea of parameter fitting



Basic idea of parameter fitting



Basic idea of parameter fitting



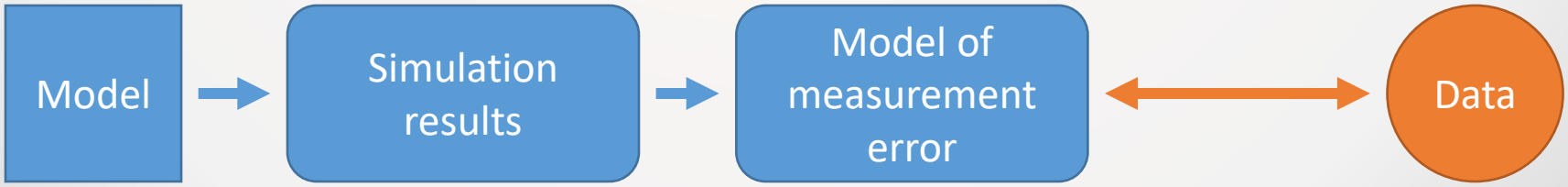
Change the parameters of a model so that it “best” fits the data

Criteria for parameter fitting

- What is the „best“ fit for a given set of data?
  - This is a mathematically difficult question
  - Fortunately, heuristically there are simple solutions.
- Required information:
  - Model
  - Knowledge about measurement process

Maximum Likelihood

- In principle, if we know the model and the measurement process, we can calculate the probability that the measurements would be the result of a simulation of the model.
- If this probability is high, the model is good.



Maximum Likelihood

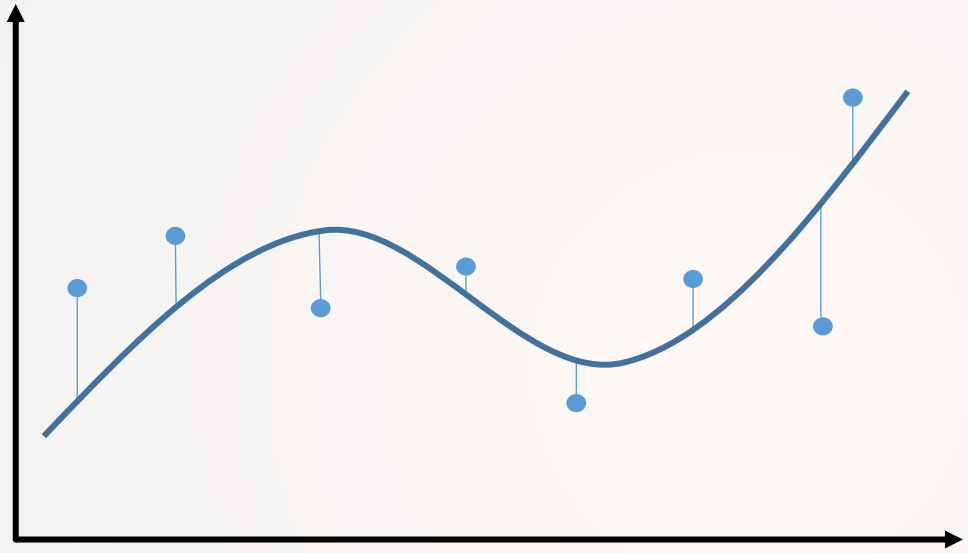
- To do parameter fitting, we need an algorithm that changes the unknown parameters so that the likelihood becomes maximal.
- This is mathematically very nice, but the probability is difficult to calculate in realistic cases.

Least squares method

If we make some assumptions about the measurement errors, we can find a rather simple criterion:

- **Assumption**: Error follows a normal distribution, measurement error is uncorrelated
- Leads to **Criterion**: Likelihood is maximal when the difference (as defined on the next slide) between measurements and simulation results is minimal
  - This is easy to calculate but not the most general case

Least squares distance measure

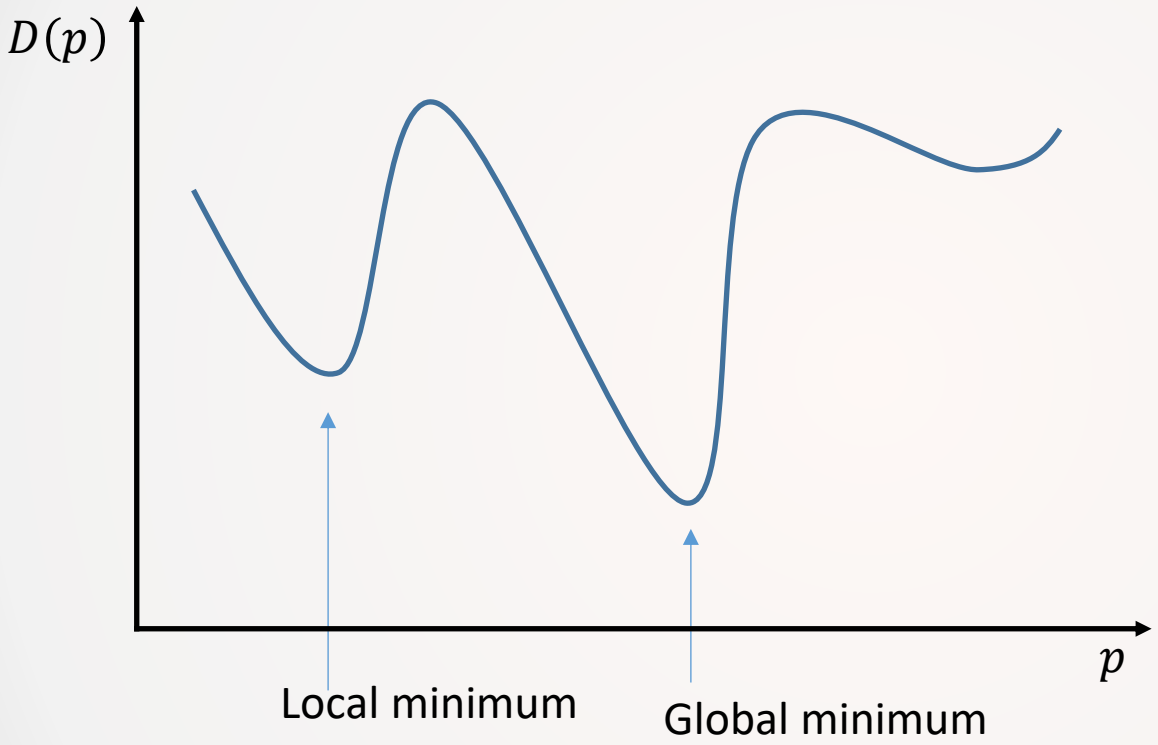


$$D(p) = \sum_{i=1}^{N} \left(x_i - y_i(p)\right)^2$$

x_i Measured values for time t_i

$y_i(p)$ Simulated values for time t_i using parameter p

The target function



- The function is usually high dimensional
- This is the function that needs to be minimized
- usually has many local minima

Parameter space

For a complete specification of the parameter fitting problem we still need to specify the unknown parameters:

- List of M parameters with allowed range of values
- These parameters span an M-dimensional space, the parameter space.
- One specific set of parameters corresponds to a point in parameter space

Optimization problem

- We now need a way to find the set of parameter values (a point p in parameter space) for which the distance $D(p)$ is minimal (the best fit).
- A systematic scan of the parameter space is not possible when the dimensionality is large (many unknown parameters)
- Example: 10 parameters with 10 values each: 10^{10} evaluations. Even if we can do 100 simulations/s, it would take 3 years.

Optimization algorithms

- In general, it is very difficult to find the parameter values for which $D(p)$ is minimal
- It can be shown that there is no optimal optimization algorithm for all cases (this means there is no way to decide which one is best for a given problem)
- This means you should always try several algorithms for difficult parameter estimation problems

Optimization Algorithms

- Based on derivatives
  - Steepest descent
  - Newton
  - Levenberg-Marquardt
- Using geometry
  - Nelder-Mead (simplex)
  - Hooke-Jeeves (pattern)
- Based on genetics
  - Genetic algorithm
  - Evolutionary programing
  - Evolution strategy
- Other stochastic
  - random search
  - particle swarm
  - simulated annealing

Specification of a parameter estimation problem

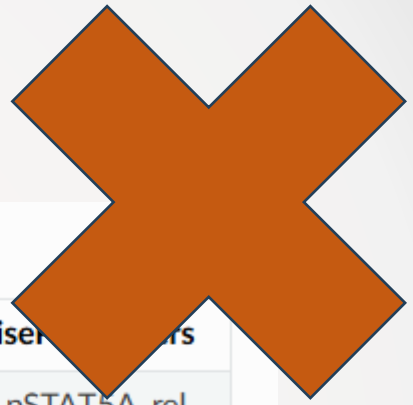
- What kind of information is needed for the computer to do a parameter estimation?
  - the model
  - the experimental data
  - the mapping between experimental data and model simulation results
  - the ranges of possible values for the unknown parameters
  - the optimization algorithm

Preparing the data

Measurement table `measurement_data.tsv` .

observableId	simulationConditionId	measurement	time	noiseParameters
pSTAT5A_rel	epo_stimulation	7.9	0	sd_pSTAT5A_rel
...	...	...	...	...
pSTAT5A_rel	epo_stimulation	15.4	240	sd_pSTAT5A_rel
pSTAT5B_rel	epo_stimulation	4.6	0	sd_pSTAT5B_rel
...	...	...	...	...
pSTAT5B_rel	epo_stimulation	10.96	240	sd_pSTAT5B_rel
rSTAT5A_rel	epo_stimulation	14.7	0	sd_rSTAT5A_rel
...	...	...	...	...
rSTAT5A_rel	epo_stimulation	32.2	240	sd_rSTAT5A_rel

Preparing the data



Measurement table `measurement_data.tsv` .

observableId	simulationConditionId	measurement	time	noise… …rs
pSTAT5A_rel	epo_stimulation	7.9	0	sd_pSTAT5A_rel
…	…	…	…	…
pSTAT5A_rel	epo_stimulation	15.4	240	sd_pSTAT5A_rel
pSTAT5B_rel	epo_stimulation	4.6	0	sd_pSTAT5B_rel
…	…	…	…	…
pSTAT5B_rel	epo_stimulation	10.96	240	sd_pSTAT5B_rel
rSTAT5A_rel	epo_stimulation	14.7	0	sd_rSTAT5A_rel
…	…	…	…	…
rSTAT5A_rel	epo_stimulation	32.2	240	sd_rSTAT5A_rel

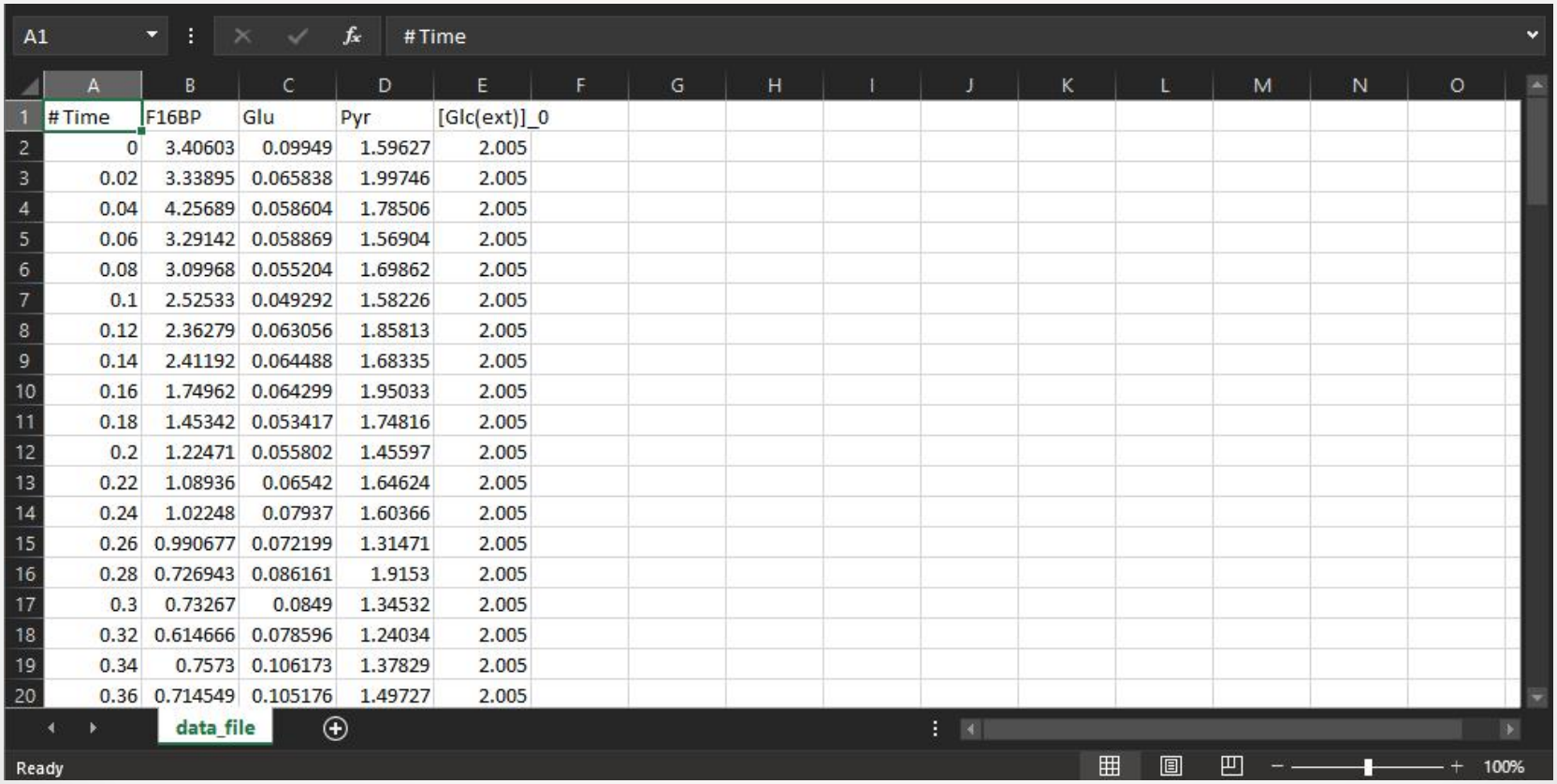
Preparing the data

	Protein			Reactant		Enter data via transpose (Rightclick		
	Initial concentration	Unit	Name	Initial concentration	Unit			
						0	10	
	0.2	g / l	Mevalonate	50	mmole / l	38.29	14.07	
						0	10	
	0.2	g / l	Mevalonate	50	mmole / l	41.23	14.99	
						0	10	
	0.2	g / l	Mevalonate	50	mmole / l	31.98	14.82	

Preparing the data

	Protein			Reactant		Enter data via transpose (Rightclick		
	Initial concentration	Unit	Name	Initial concentration	Unit			
						0	10	
	0.2	g / l	Mevalonate	50	mmole / l	38.29	14.07	
						0	10	
	0.2	g / l	Mevalonate	50	mmole / l	41.23	14.99	
						0	10	
	0.2	g / l	Mevalonate	50	mmole / l	31.98	14.82	

Preparing the data



A1 | # Time

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	# Time	F16BP	Glu	Pyr	[Glc(ext)]_0										
2	0	3.40603	0.09949	1.59627	2.005										
3	0.02	3.33895	0.065838	1.99746	2.005										
4	0.04	4.25689	0.058604	1.78506	2.005										
5	0.06	3.29142	0.058869	1.56904	2.005										
6	0.08	3.09968	0.055204	1.69862	2.005										
7	0.1	2.52533	0.049292	1.58226	2.005										
8	0.12	2.36279	0.063056	1.85813	2.005										
9	0.14	2.41192	0.064488	1.68335	2.005										
10	0.16	1.74962	0.064299	1.95033	2.005										
11	0.18	1.45342	0.053417	1.74816	2.005										
12	0.2	1.22471	0.055802	1.45597	2.005										
13	0.22	1.08936	0.06542	1.64624	2.005										
14	0.24	1.02248	0.07937	1.60366	2.005										
15	0.26	0.990677	0.072199	1.31471	2.005										
16	0.28	0.726943	0.086161	1.9153	2.005										
17	0.3	0.73267	0.0849	1.34532	2.005										
18	0.32	0.614666	0.078596	1.24034	2.005										
19	0.34	0.7573	0.106173	1.37829	2.005										
20	0.36	0.714549	0.105176	1.49727	2.005										

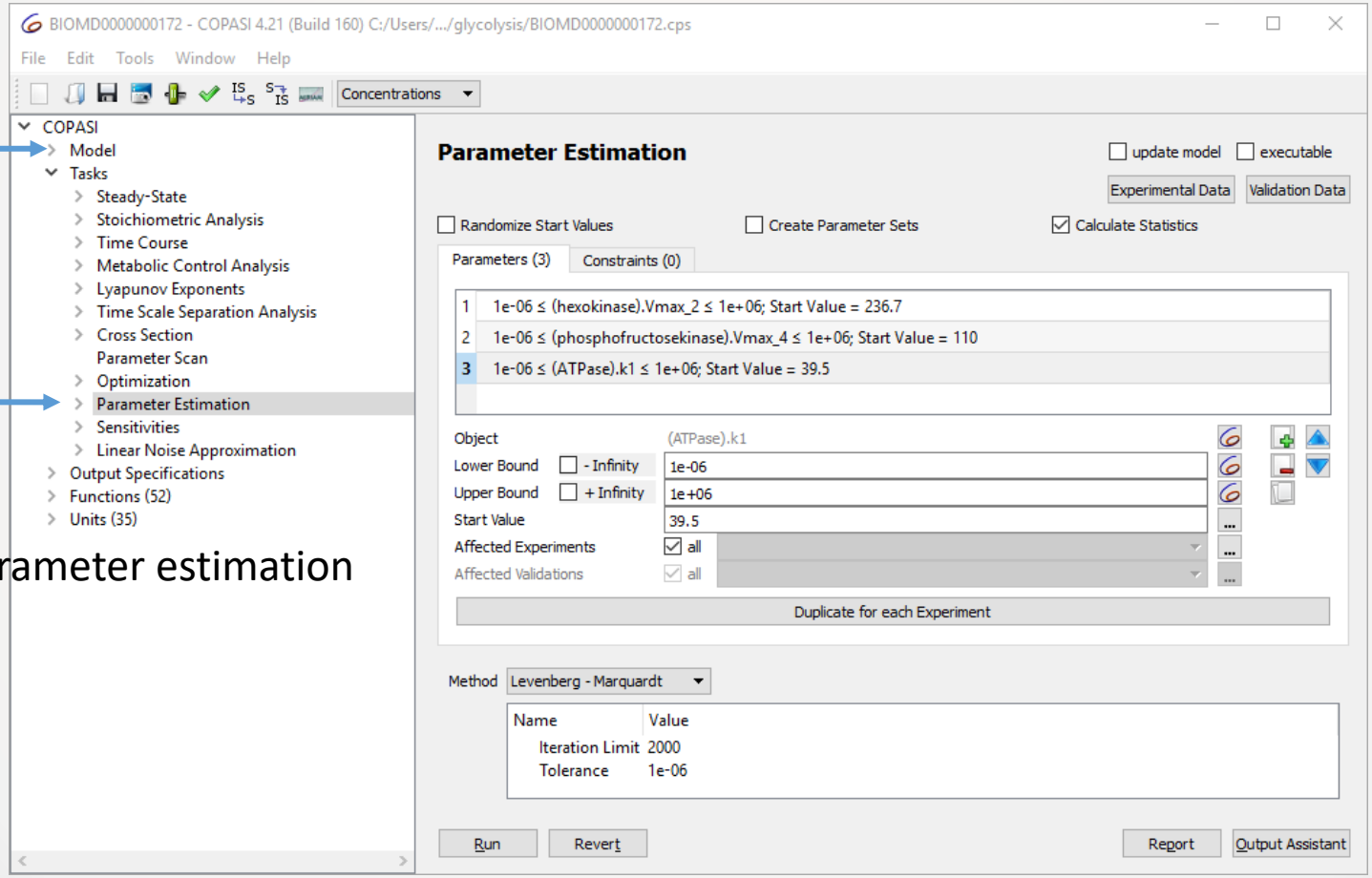
data_file

Ready 100%

Setting up…

Model

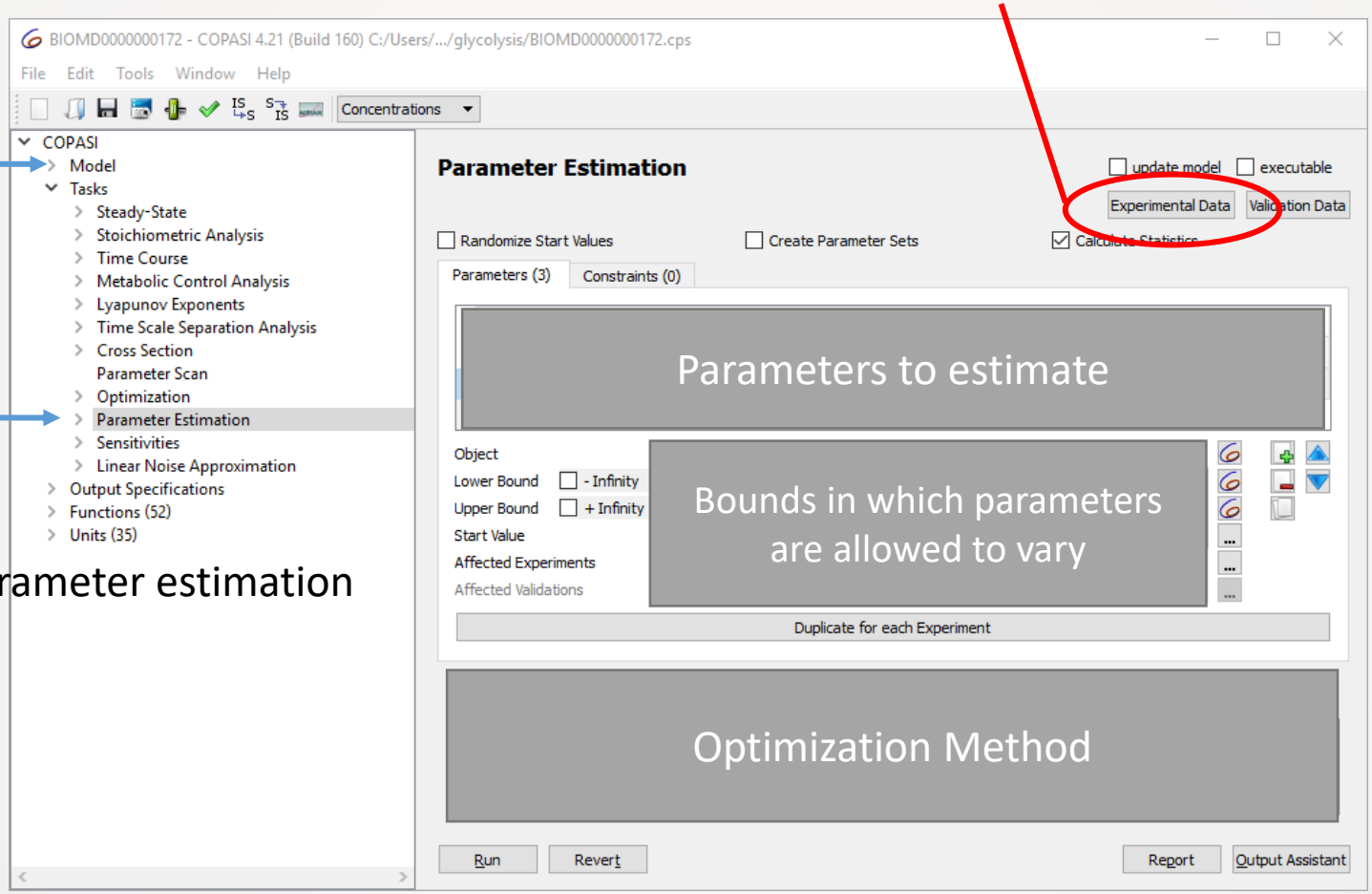
Select parameter estimation



Setting up…

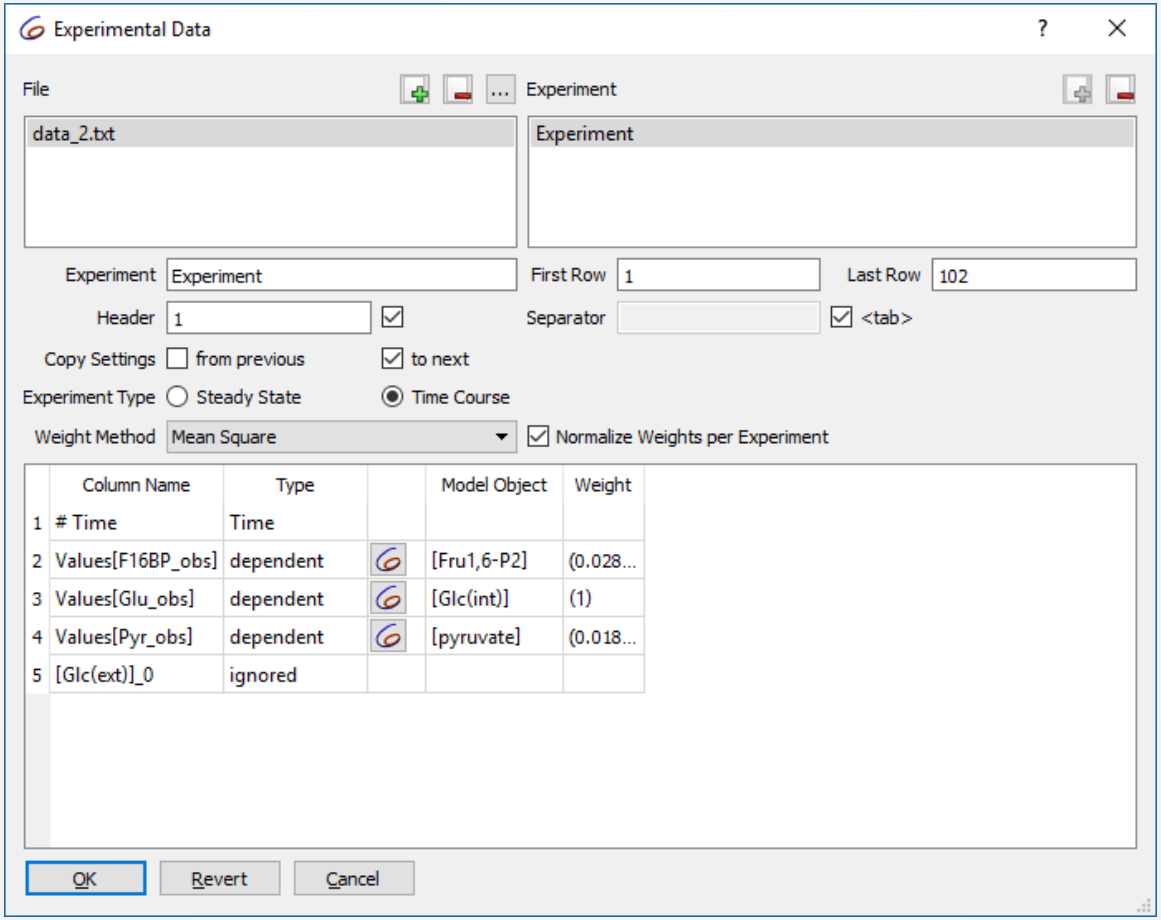
Choose experimental data

Model



Select parameter estimation

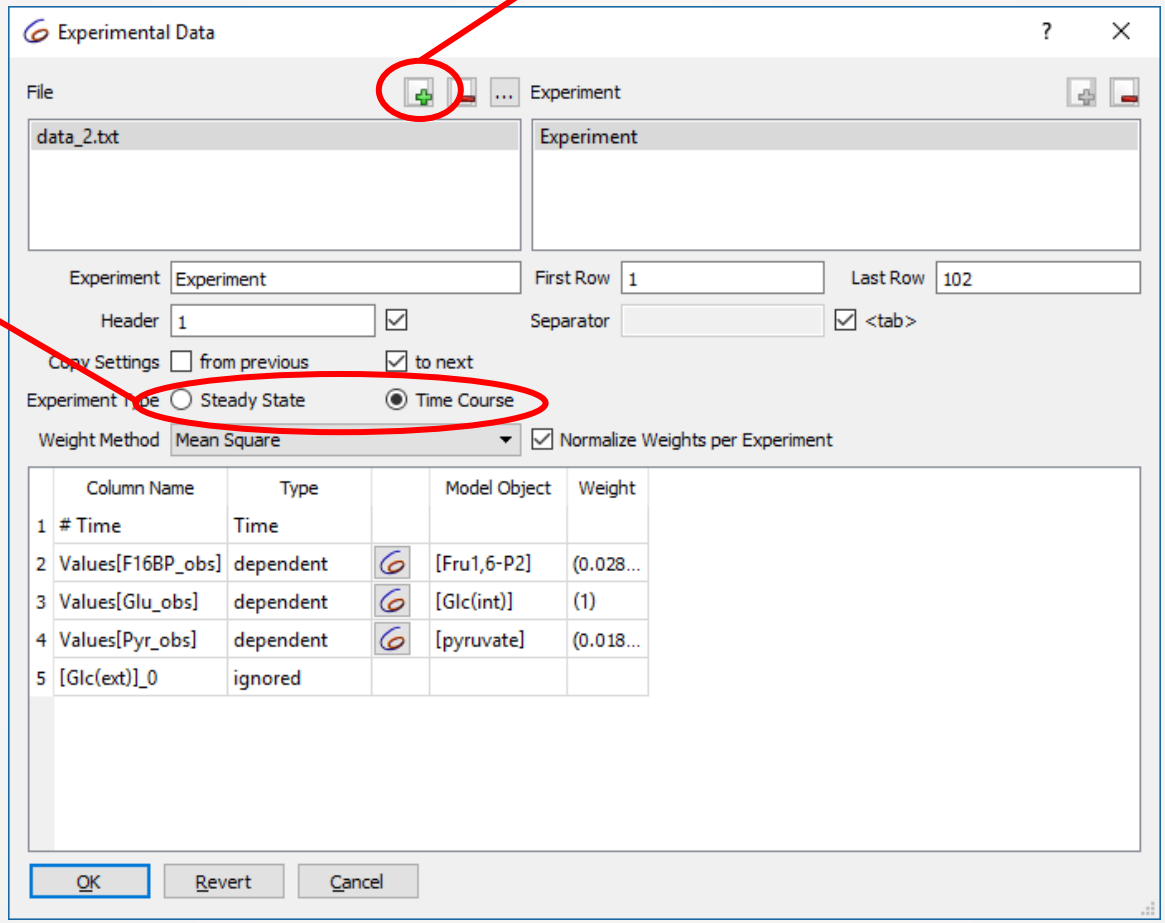
Experimental Data



Map experimental data

1. Load experiment file

Specify experiment type

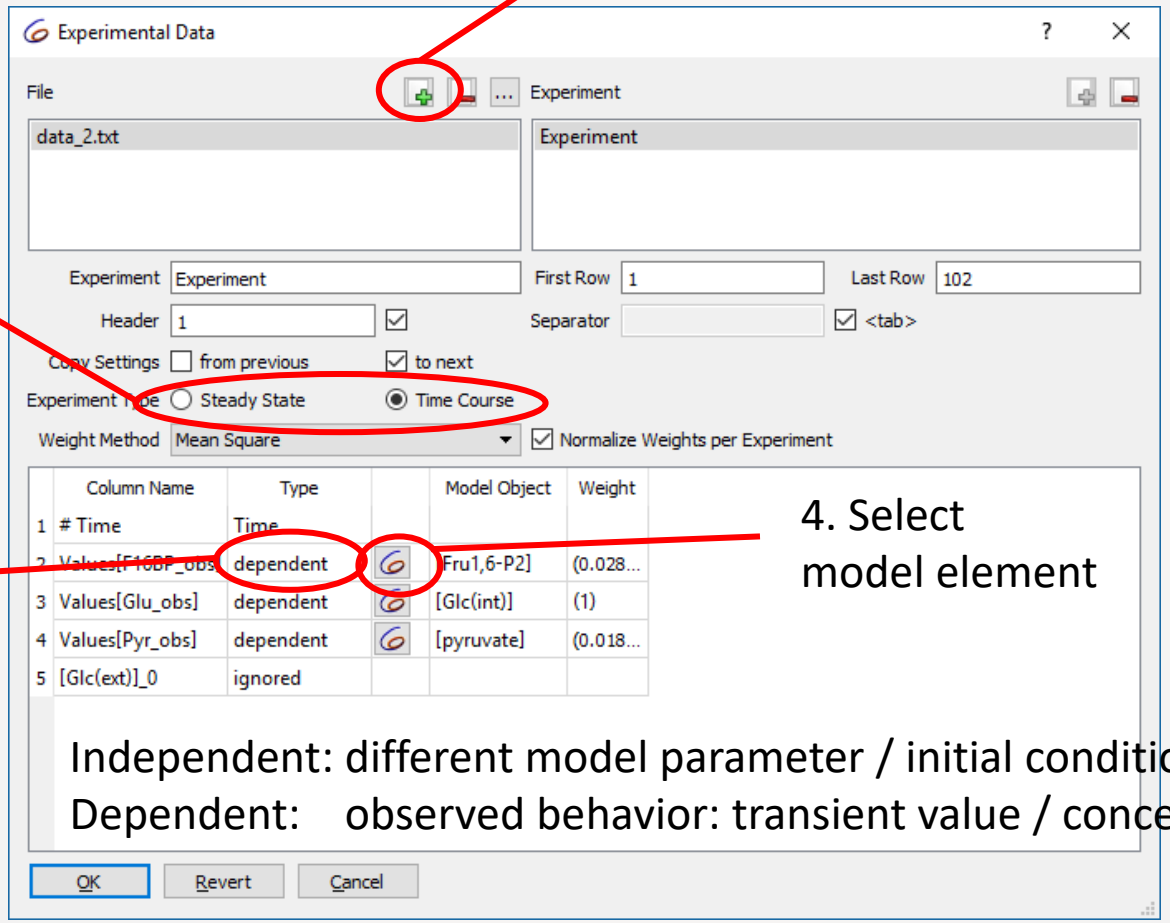


Map experimental data

1. Load experiment file

2. Specify experiment type

3. Specify type of mapping



4. Select model element

Independent: different model parameter / initial condition

Dependent: observed behavior: transient value / concentration

Important

- The result of a parameter fitting always needs to be inspected afterwards!
- Having a good result for a fit does **not** mean that the parameter value is the „true“ one. This depends on the assumptions about the errors and the correctness of the model.
- For the stochastic algorithms, the result is not reproducible!

Demo Parameter Sets

Save different parameterizations, see results from different experiments.

Example 1

Set up Parameter Estimation for the model you created earlier.

Example 2

- Import BioModel #10, set up parameter estimation with the provided data file.

Using several experiments for parameter estimation

- The more data available, the better.
  - if you have data from several experiments, it should be used for parameter estimation simultaneously
- COPASI can deal with an arbitrary number of experiments, also of different kinds (combined time course/steady state, different variables, different time points, etc.)

several experiments...

- Adding data from several experiments is straight forward in COPASI. Several data files can be specified and each can contain several experiments
- Important information: What is the same for all experiments and what is different between experiments? For the things that are different, are they known or unknown?

several experiments...

- Simplest case: Repeated experiments.
  - nothing special needs to be done in COPASI
- Several experiments under different conditions. The conditions are known.
  - Example: Different stimulations in several experiments.
  - In COPASI: The stimulation needs to be a parameter in the model. In the experimental data specification this value is selected as an independent parameter. Independent data is known data that is provided in the data file. Dependent data is data that is used for fitting.

several experiments...

- Experiments where some conditions are different, but not known
  - Example: *In vivo* experiments, even if the experiment is repeated with the same preparation, the initial conditions (inside the organism) are typically different.
  - In COPASI: The user can specify that some parameters are fitted for all experiments, and some are fitted for a specific subset of experiments.

Example 3

- Use BioModel #172 with the provided data.
- Add a second data file

Example 4

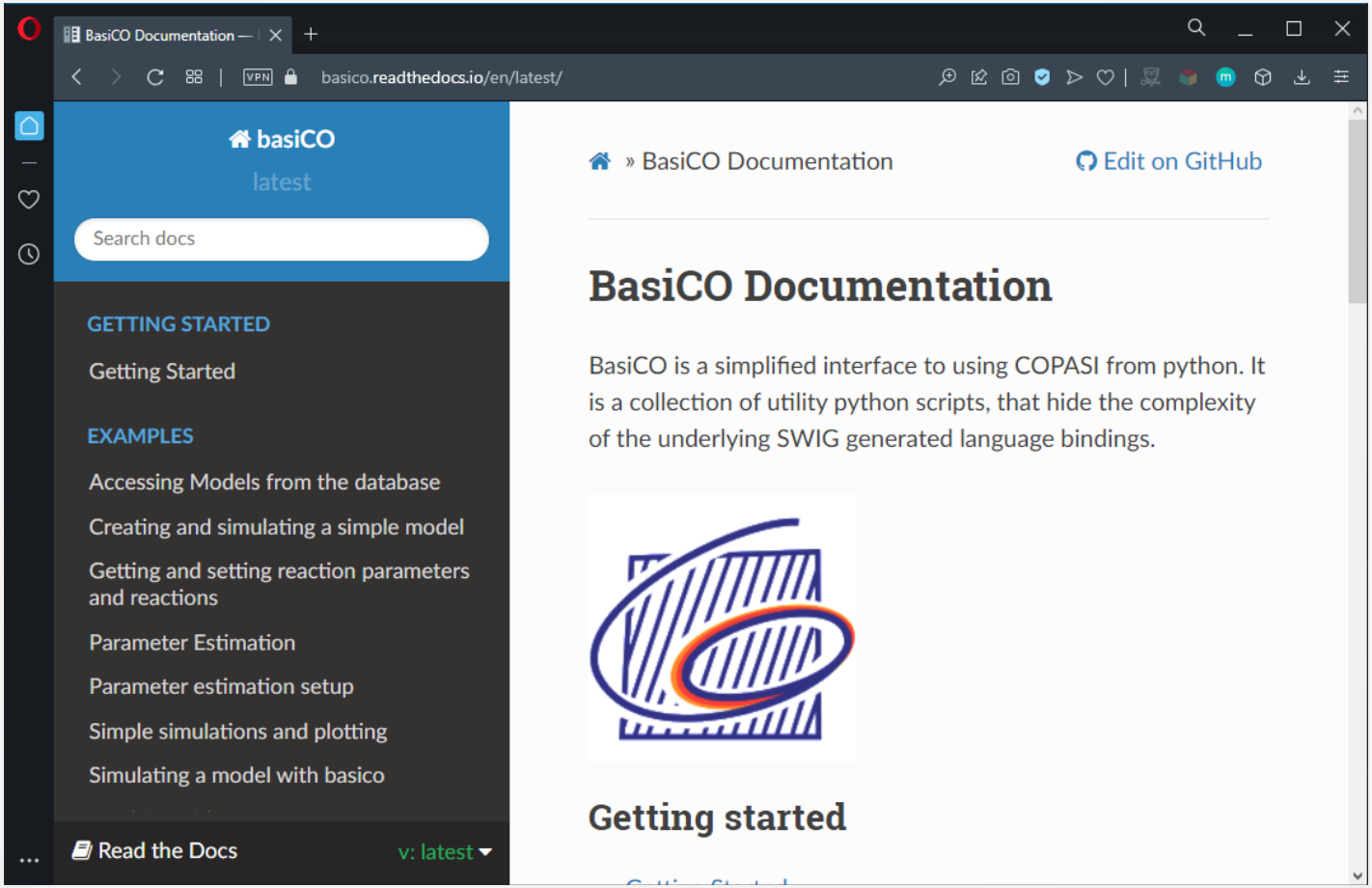
- An example demonstrating a file with many local minimas. Thus, local methods might not always yield a good result.

Scripting / Summary

Using COPASI from other languages

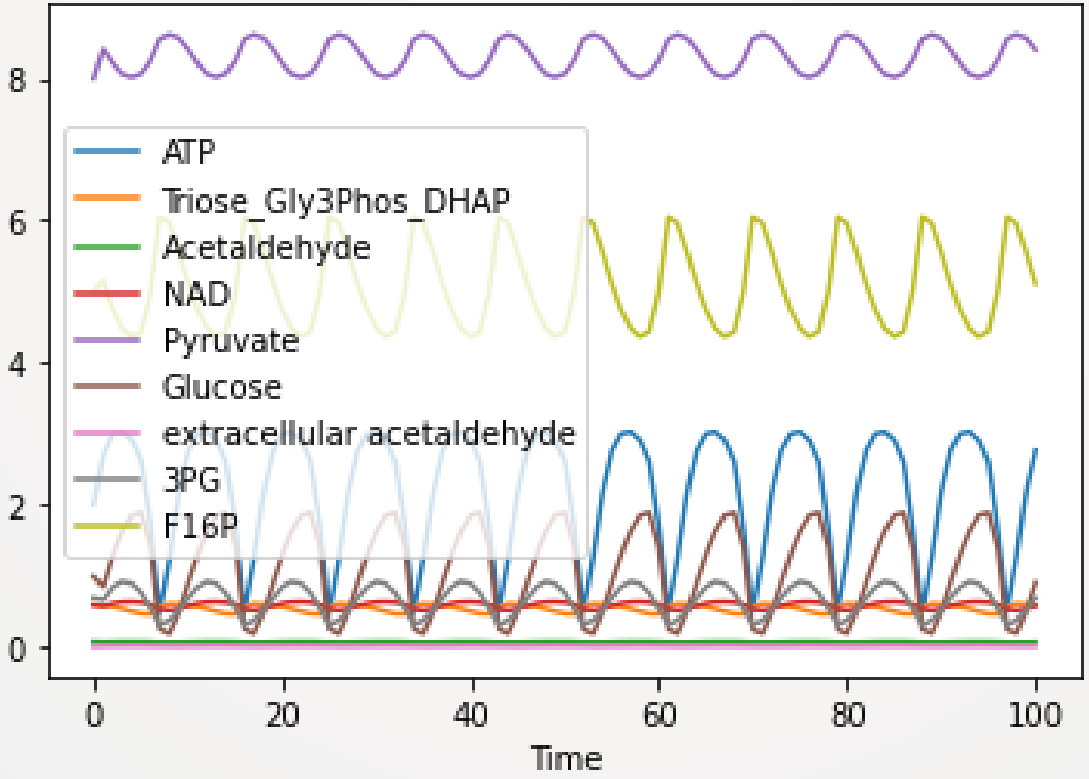
- COPASI is written in C++, other languages (Java, .NET, Python, Ruby, R) are supported by generating language bindings via SWIG.
- On top of those automatically generated wrappers, packages are available for Python and R that abstract away from the inherent complexities:
  - R: CoRC: https://jpahle.github.io/CoRC/
  - Python: https://basico.readthedocs.io

BasiCO


BasiCO Documentation
basico.readthedocs.io/en/latest/
basiCO
latest
Search docs
GETTING STARTED
Getting Started
EXAMPLES
Accessing Models from the database
Creating and simulating a simple model
Getting and setting reaction parameters and reactions
Parameter Estimation
Parameter estimation setup
Simple simulations and plotting
Simulating a model with basico
Read the Docs
v: latest
» BasiCO Documentation
Edit on GitHub
BasiCO Documentation
BasiCO is a simplified interface to using COPASI from python. It is a collection of utility python scripts, that hide the complexity of the underlying SWIG generated language bindings.
Getting started

Simplifies to

```
load_biomodel(206)
run_time_course(duration = 100).plot()
```



Aim for this tutorial

Set you up so you can:

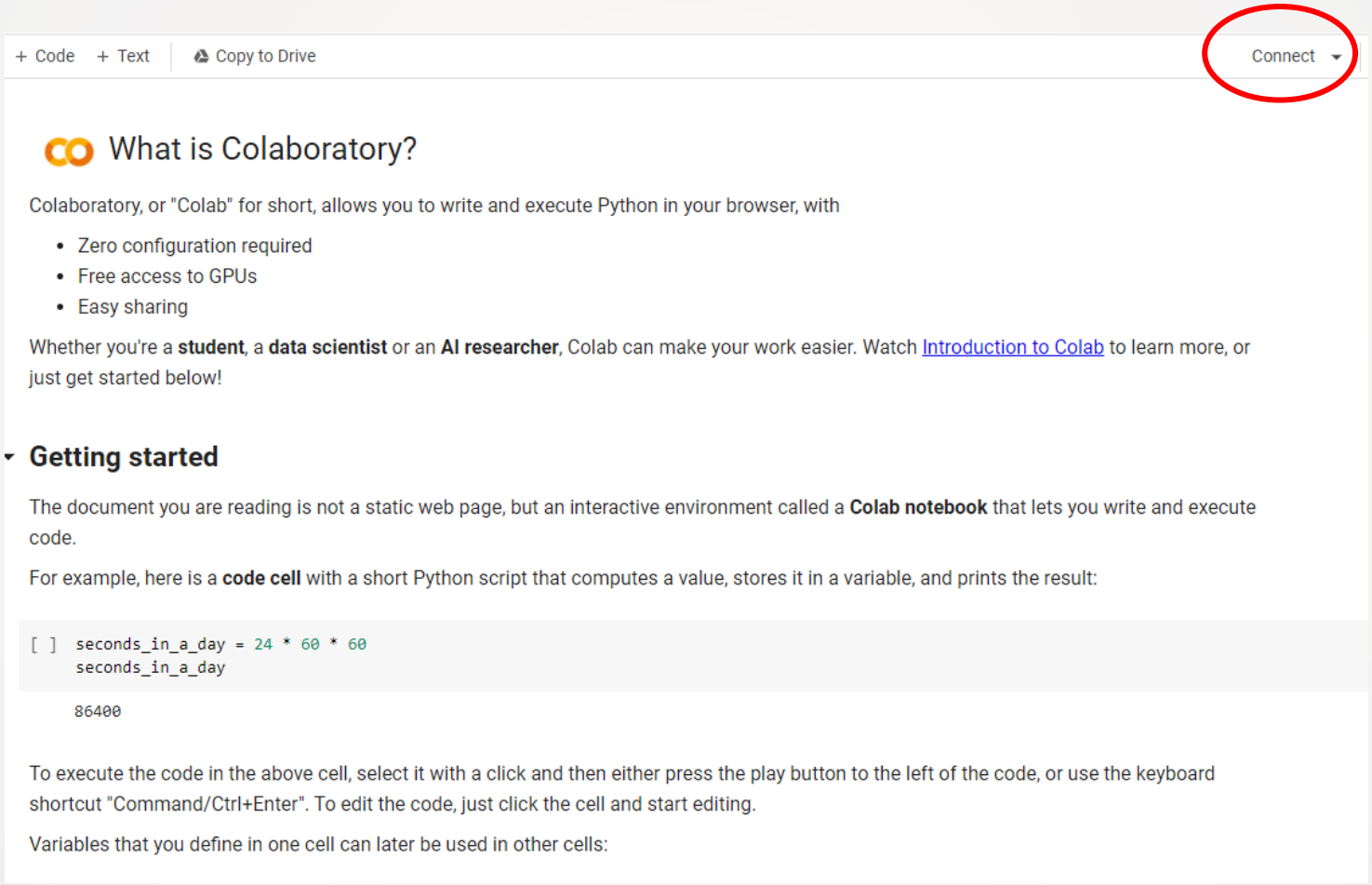
- Create / load models into basico
- Modify them
- Run simulations on them
- (potentially do parameter estimation)
- Know where to go for more information

Installation

```
pip install copasi-basico
```

- Will install on windows / linux / osx.
- Will install on
  - https://mybinder.org/
  - https://colab.research.google.com/

Colab



https://colab.research.google.com

Creating Models

basico.model_io.new_model(***kwargs*)

Creates a new model and sets it as current.

Parameters: **kwargs** – optional arguments

- *name* (str): the name for the new model
- *quantity_unit* (str): the unit to use for species
- *time_unit* (str): the time unit to use
- *volume_unit* (str): the unit to use for 3D compartments
- *area_unit* (str): the unit to use for 2D compartments
- *length_unit* (str): the unit to use for 1D compartments
- *notes*: sets notes for the model (either plain text, or valid xhtml)

Returns: the new model

Return type: COPASI.CDataModel

Importing Models

basico.model_io.load_example(*selector*)

Loads the example matching the selector.

Parameters: **selector** (*str*) – the filter expression to use for the examples see `get_examples()`

Returns: the loaded model, or None, if none matched

Return type: COPASI.CDataModel or None

basico.model_io.load_model(*location*)

Loads the model and sets it as current

Parameters: **location** (*str*) – either a filename, url or raw string of a COPASI / SBML model

Returns: the loaded model

Return type: COPASI.CDataModel

Opening in COPASI

basico.model_io.open_copasi(***kwargs*)

Saves the model as COPASI file and opens it in COPASI.

The file will be written to a temporary file, and then it will be executed, so that the application registered to open it will start.

Parameters: **kwargs** – optional arguments:

- *model*: to specify the data model to be used (if not specified the one from `get_current_model()` will be taken)
- *filename* (str): the file name to write to, if not given a temp file will be created that will be deleted at the end of the python session.

Returns: None

Saving the Model

basico.model_io.save_model(*filename*, ***kwargs*)

Saves the model to the given filename.

Parameters:
- **filename** (*str*) – the file to be written
- **kwargs** – optional arguments:

- *model*: to specify the data model to be used (if not specified the one from `get_current_model()` will be taken)

- *type* (str): *copasi* to write COPASI files, *sbml* to write SBML files (defaults to *copasi*)
- *overwrite* (bool): whether the file should be overwritten if present (defaults to True)
- *level* (int): SBML level to export
- *version* (int): SBML version to export
- *export_copasi_miriam* (bool): whether to export copasi miriam annotations
- *export_incomplete* (bool): whether to export incomplete SBML model

Returns: None

Running a simulation

Examples

To run a time course for the duration of 10 time units use

```
>>> run_time_course(10)
```

To run a time course for the duration of 10 time units, in 50 simulation steps use

```
>>> run_time_course(10, 50)
```

To run a time course from 0, for the duration of 10 time units, in 50 simulation steps use:

```
>>> run_time_course(0, 10, 50)
```

Running Parameter Estimation

```
[13]: run_parameter_estimation(method='Levenberg - Marquardt', update_model=True)
```

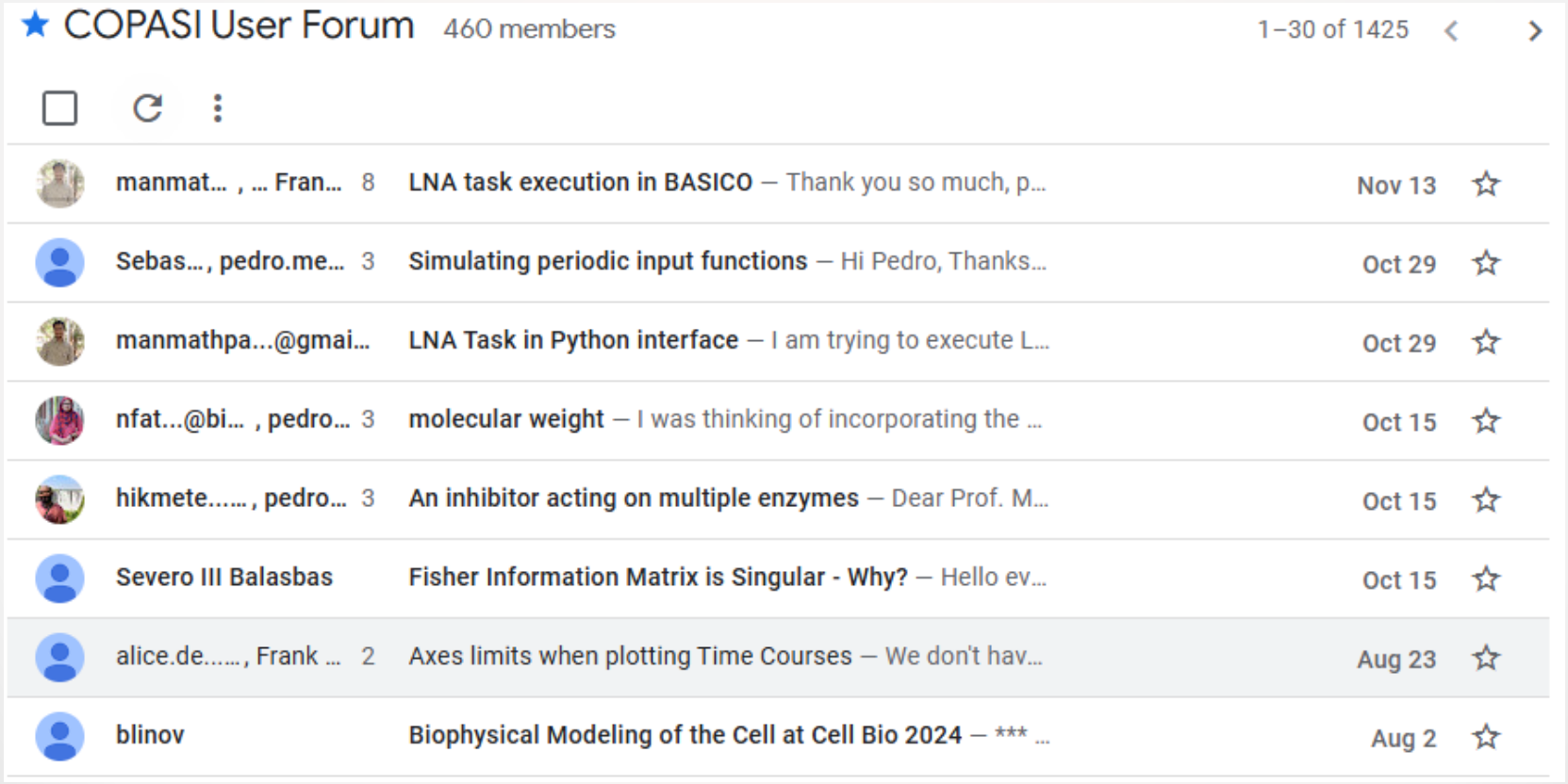
[13]:

name	lower	upper	sol	affected
(R1).k2	1e-6	1e6	0.000002	[]
(R2).k1	1e-6	1e6	44.729444	[]
Values[offset]	-0.2	0.4	0.043013	[Experiment_1]
Values[offset]	-0.2	0.4	0.054212	[Experiment_3]
Values[offset]	-0.2	0.4	-0.050942	[Experiment]
Values[offset]	-0.2	0.4	0.040968	[Experiment_4]
Values[offset]	-0.2	0.4	0.047976	[Experiment_2]

https://basico.readthedocs.io/en/latest/notebooks/Parameter_Estimation_Example.html

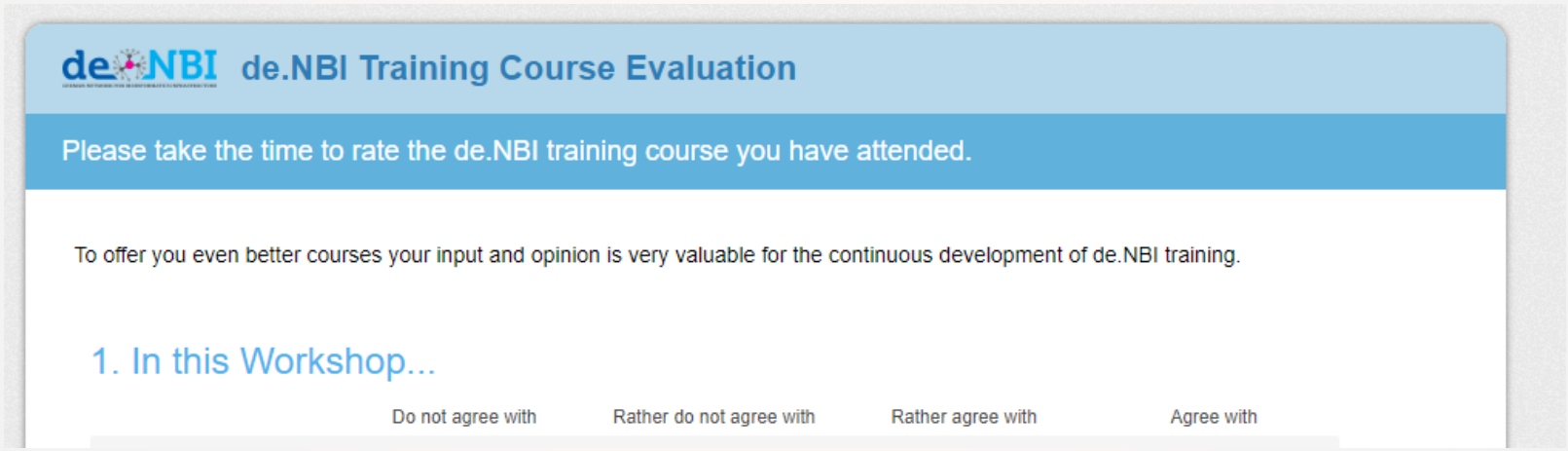
Feedback / Suggestions

User Forum



https://groups.google.com/g/copasi-user-forum

de.NBI Survey



https://tinyurl.com/survey-denbi-sysbio-2024

Thank you !