

# 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?

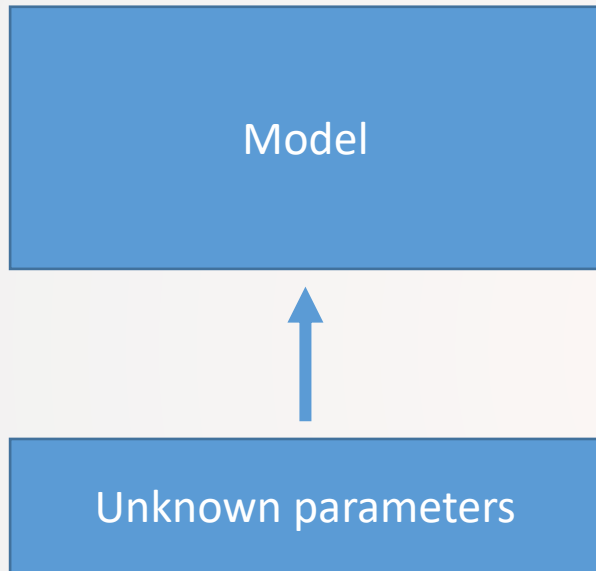
# Introduction

- 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

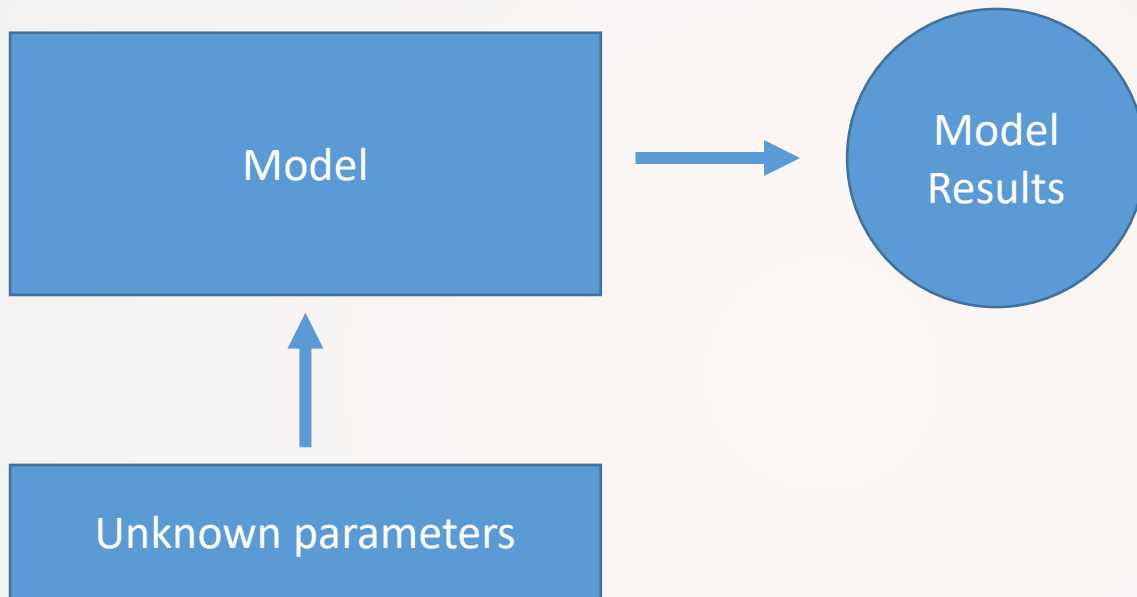
# Introduction

- 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

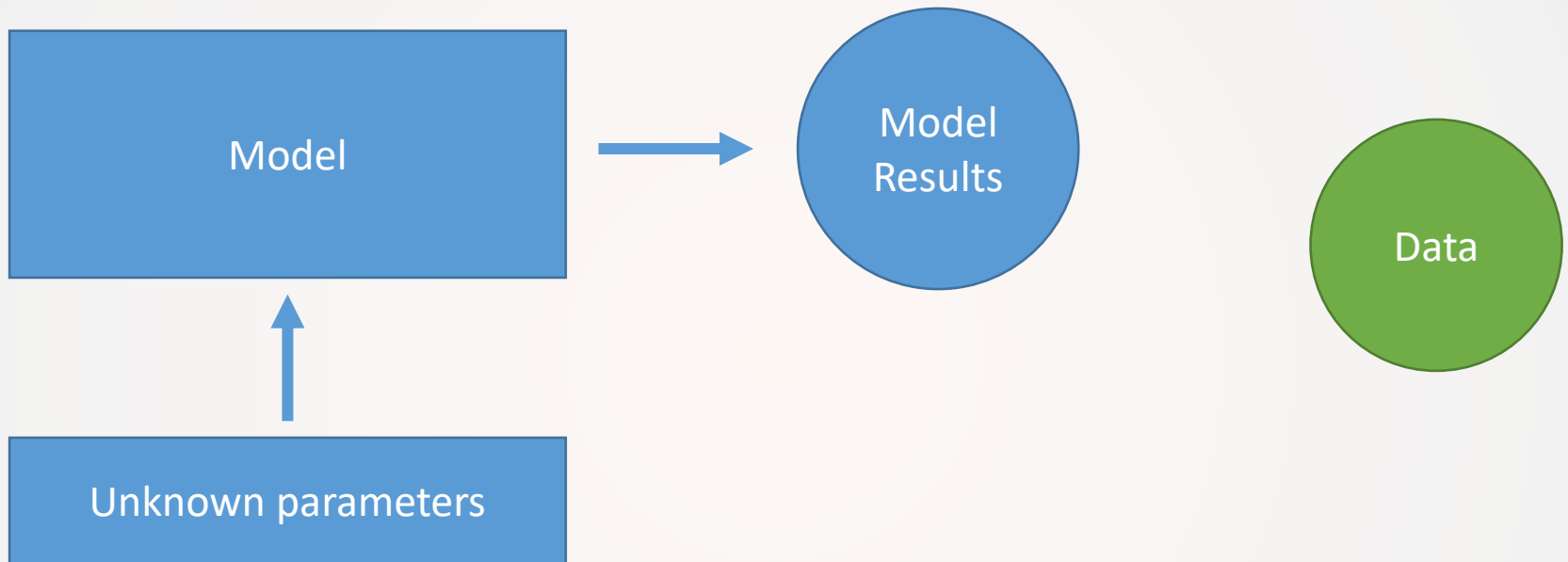
# Basic idea of parameter fitting



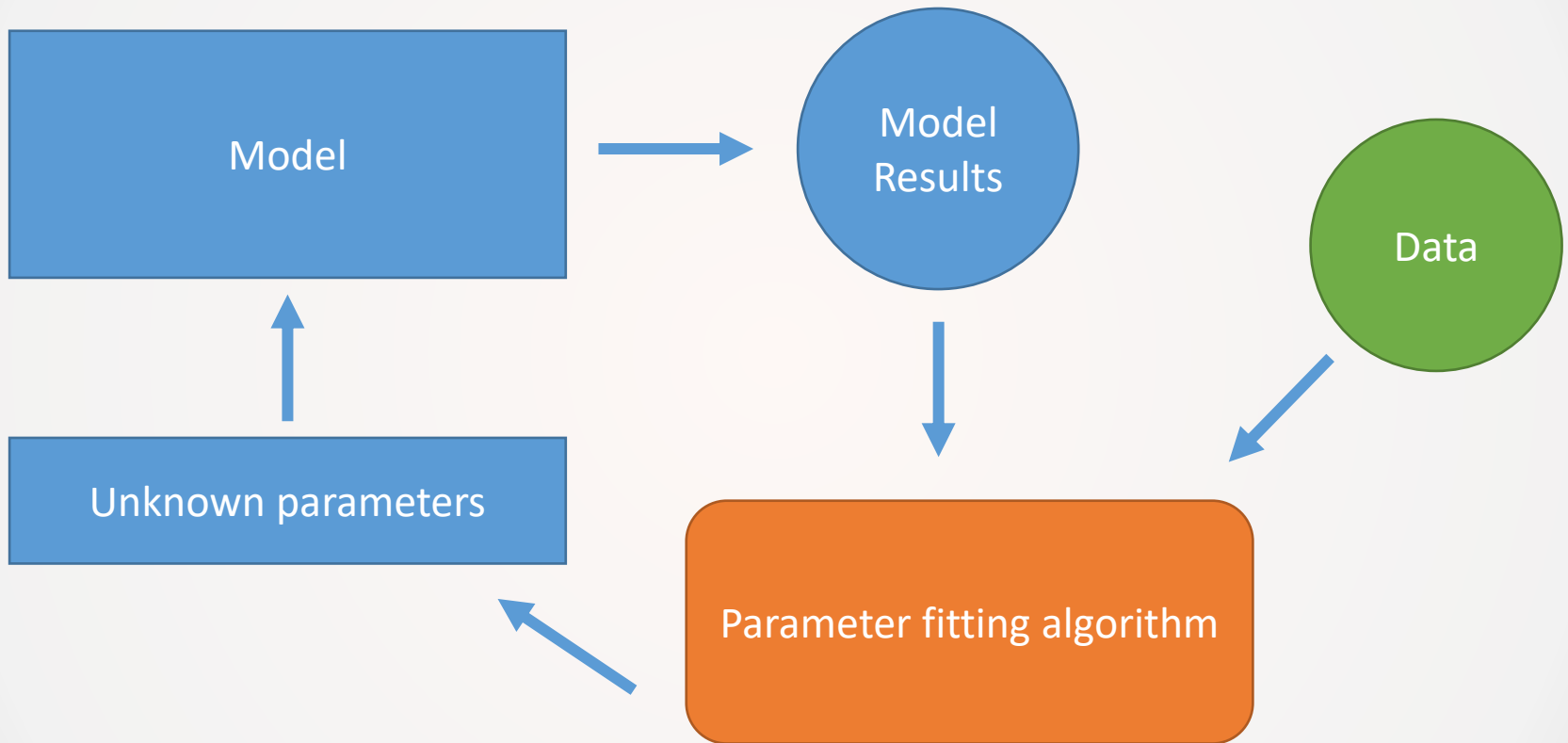
# Basic idea of parameter fitting



# Basic idea of parameter fitting



# Basic idea of parameter fitting



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

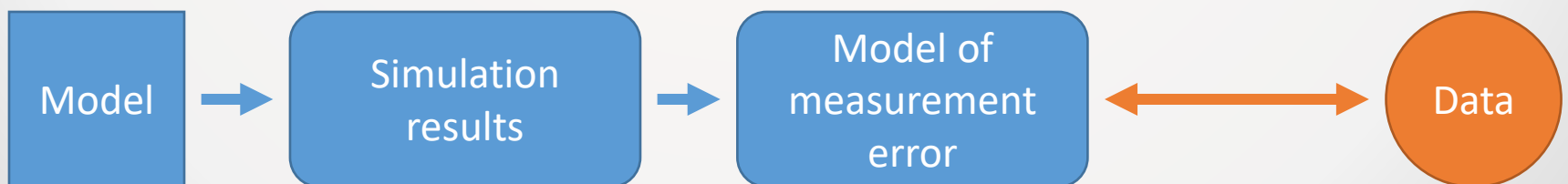


# Criteria for parameter fitting

- 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

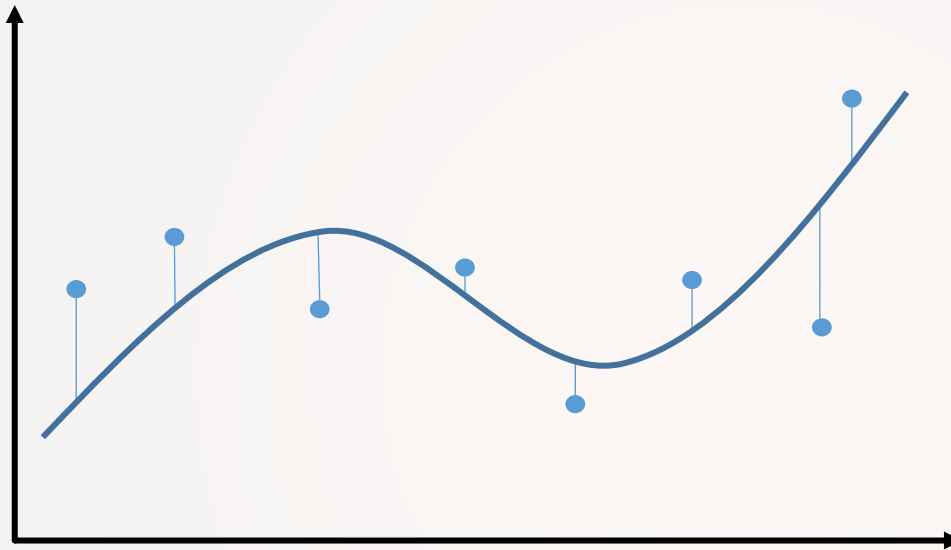
- To do parameter fitting, we need an algorithm that changes the unknown parameters so that the likelihood becomes maximal.
- This is mathematically very nice, but the probability is difficult to calculate in realistic cases.

# 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

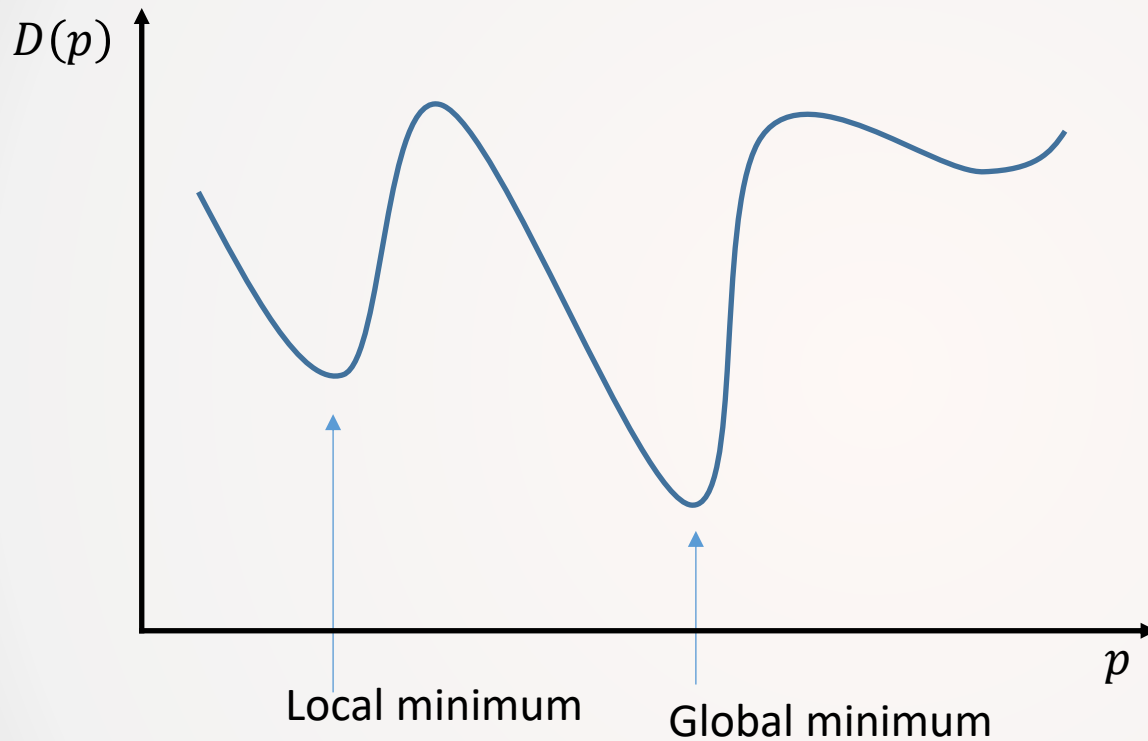


$$D(p) = \sum_{i=1}^N (x_i - y_i(p))^2$$

$x_i$  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 programming
  - 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`.

<b>observableId</b>	<b>simulationConditionId</b>	<b>measurement</b>	<b>time</b>	<b>noiseParameters</b>
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`.

<b>observableId</b>	<b>simulationConditionId</b>	<b>measurement</b>	<b>time</b>	<b>noiseParameters</b>
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

<i>Protein</i>			<i>Reactant</i>			<i>Enter data via transpose (Rightclick)</i>		
<i>Initial concentration</i>	<i>Unit</i>	<i>Name</i>	<i>Initial concentration</i>	<i>Unit</i>				
					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



<i>Protein</i>			<i>Reactant</i>			<i>Enter data via transpose (Rightclick)</i>		
<i>Initial concentration</i>	<i>Unit</i>	<i>Name</i>	<i>Initial concentration</i>	<i>Unit</i>				
					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		





# Setting up...

Model

Select parameter estimation

The screenshot shows the COPASI software interface. The left sidebar displays a tree view of the model structure. The 'Parameter Estimation' option is selected and highlighted. The main window displays the 'Parameter Estimation' configuration panel. It includes options for 'update model', 'executable', 'Experimental Data', and 'Validation Data'. There are checkboxes for 'Randomize Start Values', 'Create Parameter Sets', and 'Calculate Statistics'. A list of parameters is shown, with three parameters selected: 1.  $1e-06 \leq (\text{hexokinase}).V_{\text{max}_2} \leq 1e+06$ ; Start Value = 236.7; 2.  $1e-06 \leq (\text{phosphofructokinase}).V_{\text{max}_4} \leq 1e+06$ ; Start Value = 110; 3.  $1e-06 \leq (\text{ATPase}).k_1 \leq 1e+06$ ; Start Value = 39.5. The third parameter is currently selected. Below the list, the configuration for the selected parameter (Object: (ATPase).k1) is shown, including Lower Bound (1e-06), Upper Bound (1e+06), Start Value (39.5), Affected Experiments (all), and Affected Validations (all). The Method is set to Levenberg - Marquardt. A table shows the current values: Iteration Limit 2000 and Tolerance 1e-06. Buttons for 'Run', 'Revert', 'Report', and 'Output Assistant' are visible at the bottom.

BIOMD000000172 - COPASI 4.21 (Build 160) C:/Users/.../glycolysis/BIOMD000000172.cps

File Edit Tools Window Help

Concentrations

COPASI

- > Model
- > Tasks
  - > Steady-State
  - > Stoichiometric Analysis
  - > Time Course
  - > Metabolic Control Analysis
  - > Lyapunov Exponents
  - > Time Scale Separation Analysis
  - > Cross Section
    - Parameter Scan
  - > Optimization
  - > Parameter Estimation
  - > Sensitivities
  - > Linear Noise Approximation
- > Output Specifications
- > Functions (52)
- > Units (35)

### Parameter Estimation

update model  executable

Experimental Data Validation Data

Randomize Start Values  Create Parameter Sets  Calculate Statistics

Parameters (3) Constraints (0)

1	$1e-06 \leq (\text{hexokinase}).V_{\text{max}_2} \leq 1e+06$ ; Start Value = 236.7
2	$1e-06 \leq (\text{phosphofructokinase}).V_{\text{max}_4} \leq 1e+06$ ; Start Value = 110
3	$1e-06 \leq (\text{ATPase}).k_1 \leq 1e+06$ ; Start Value = 39.5

Object (ATPase).k1

Lower Bound  - Infinity 1e-06

Upper Bound  + Infinity 1e+06

Start Value 39.5

Affected Experiments  all

Affected Validations  all

Duplicate for each Experiment

Method Levenberg - Marquardt

Name	Value
Iteration Limit	2000
Tolerance	1e-06

Run Revert Report Output Assistant

# Setting up...

Choose experimental data

Model

Select parameter estimation

The screenshot shows the COPASI software interface. The left sidebar displays a tree view of tasks, with 'Parameter Estimation' selected. The main window is titled 'Parameter Estimation' and contains several sections: 'Parameters (3)' and 'Constraints (0)' tabs, a large grey box labeled 'Parameters to estimate', a section for 'Object' with 'Lower Bound' and 'Upper Bound' options, a large grey box labeled 'Bounds in which parameters are allowed to vary', and a large grey box labeled 'Optimization Method'. At the bottom, there are 'Run', 'Revert', 'Report', and 'Output Assistant' buttons. A red circle highlights the 'Experimental Data' and 'Validation Data' buttons in the top right corner, with a red arrow pointing to the text 'Choose experimental data' above the window.

# Experimental Data

Experimental Data

File Experiment

data\_2.txt Experiment




Experiment Experiment First Row 1 Last Row 102

Header 1  Separator   <tab>

Copy Settings  from previous  to next

Experiment Type  Steady State  Time Course

Weight Method Mean Square  Normalize Weights per Experiment

	Column Name	Type		Model Object	Weight
1	# Time	Time			
2	Values[F16BP_obs]	dependent		[Fru1,6-P2]	(0.028...
3	Values[Glu_obs]	dependent		[Glc(int)]	(1)
4	Values[Pyr_obs]	dependent		[pyruvate]	(0.018...
5	[Glc(ext)]_0	ignored			

OK Revert Cancel

# Map experimental data

1. Load experiment file

Specify experiment type

Experimental Data

File

data\_2.txt

Experiment

Experiment Experiment

First Row 1 Last Row 102

Header 1  Separator   <tab>

Copy Settings  from previous  to next

Experiment Type  Steady State  Time Course

Weight Method Mean Square  Normalize Weights per Experiment

	Column Name	Type		Model Object	Weight
1	# Time	Time			
2	Values[F16BP_obs]	dependent		[Fru1,6-P2]	(0.028...
3	Values[Glu_obs]	dependent		[Glc(int)]	(1)
4	Values[Pyr_obs]	dependent		[pyruvate]	(0.018...
5	[Glc(ext)]_0	ignored			

OK Revert Cancel

# Map experimental data

1. Load experiment file

2. Specify experiment type

3. Specify type of mapping

4. Select model element

Experimental Data

File

data\_2.txt

Experiment

Experiment Experiment

First Row 1 Last Row 102

Header 1  Separator   <tab>

Copy Settings  from previous  to next

Experiment Type  Steady State  Time Course

Weight Method Mean Square  Normalize Weights per Experiment

Column Name	Type		Model Object	Weight
1 # Time	Time			
2 Values[Fru1,6-P2_obs]	dependent		Fru1,6-P2]	(0.028...
3 Values[Glu_obs]	dependent		[Glc(int)]	(1)
4 Values[Pyr_obs]	dependent		[pyruvate]	(0.018...
5 [Glc(ext)]_0	ignored			

Independent: different model parameter / initial condition  
Dependent: observed behavior: transient value / concentration

OK Revert Cancel

# 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.

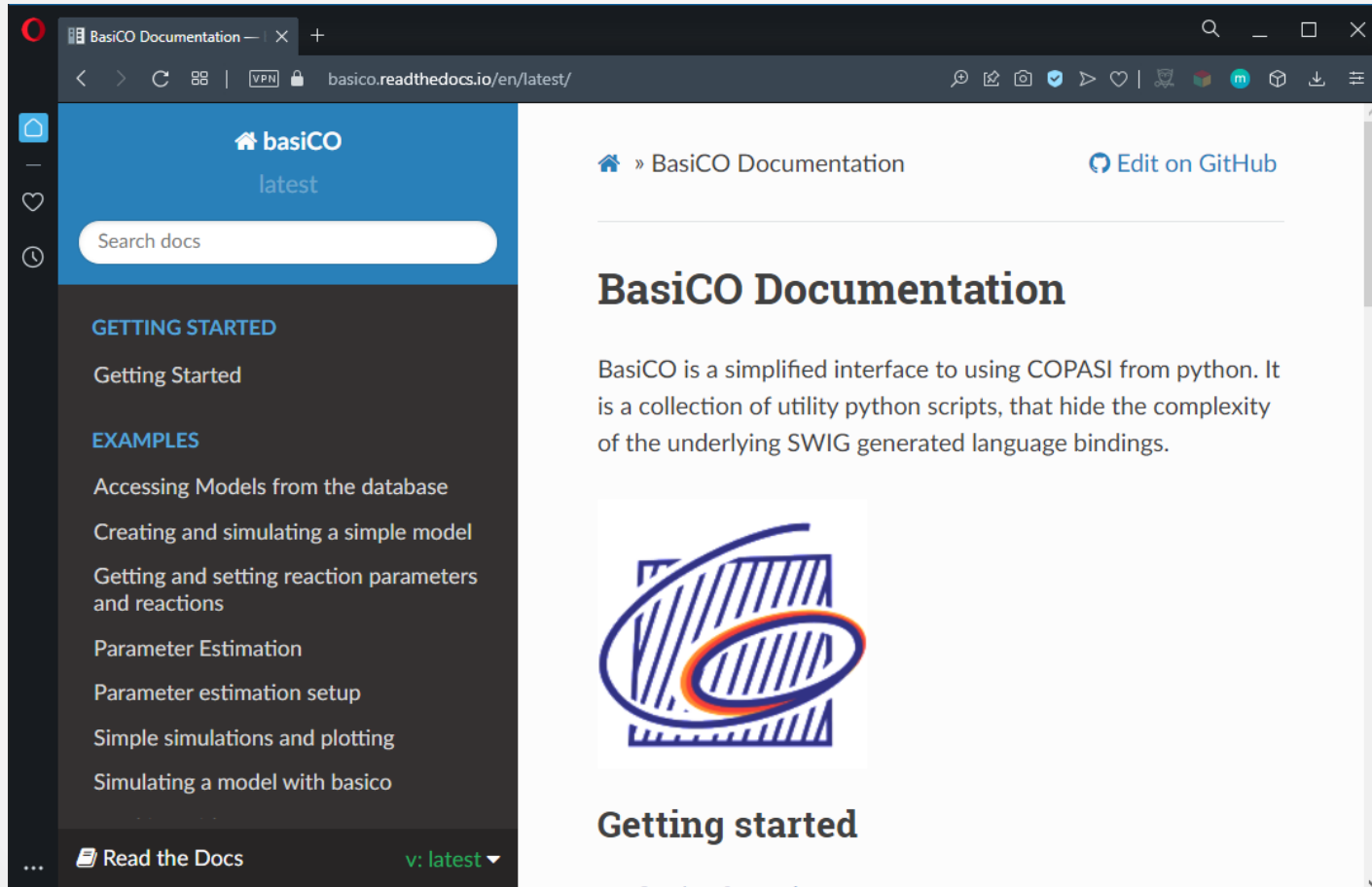
Scripting / Summary



# 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



The screenshot shows a web browser displaying the BasiCO documentation page. The browser's address bar shows the URL `basico.readthedocs.io/en/latest/`. The page features a dark blue header with the BasiCO logo and the word "latest". A search bar is located below the header. The left sidebar contains a navigation menu with sections for "GETTING STARTED" and "EXAMPLES". The main content area displays the title "BasiCO Documentation" and a description of the tool. Below the text is a logo consisting of a blue square with diagonal lines and a red and blue oval. The page also includes a "Getting started" section and an "Edit on GitHub" link.

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.

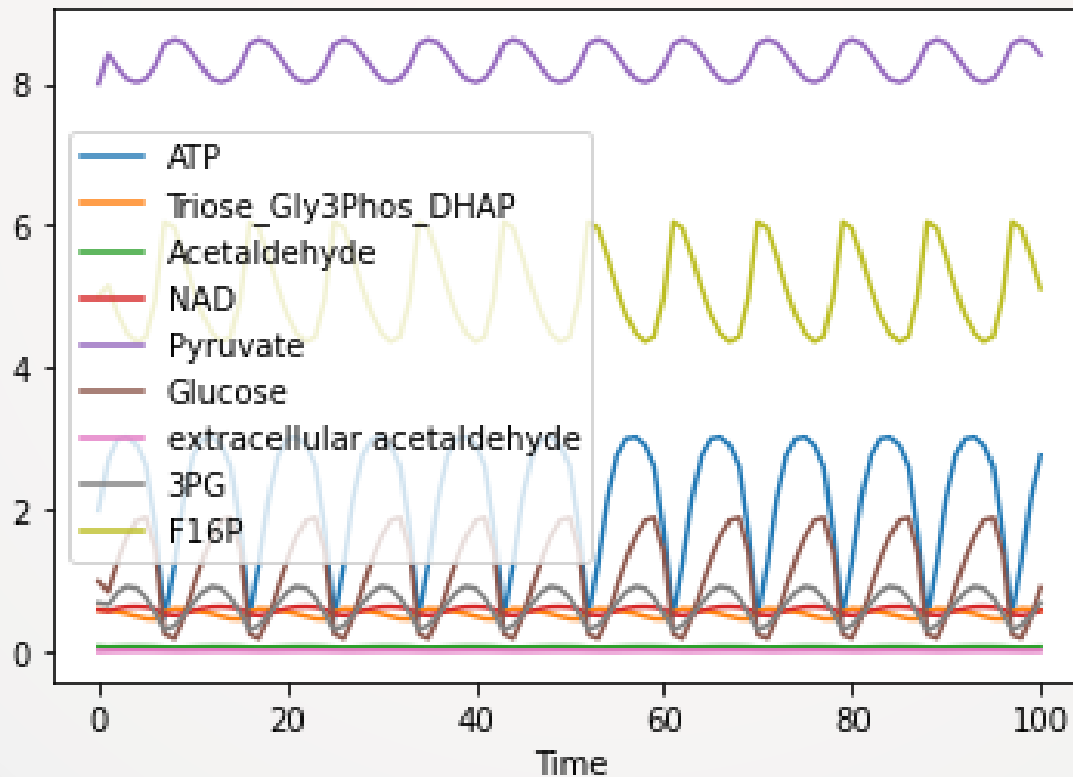


### Getting started

Simplifies to

```
load_biomodel(206)
```

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



# Aim for this tutorial

Set you up so you can:

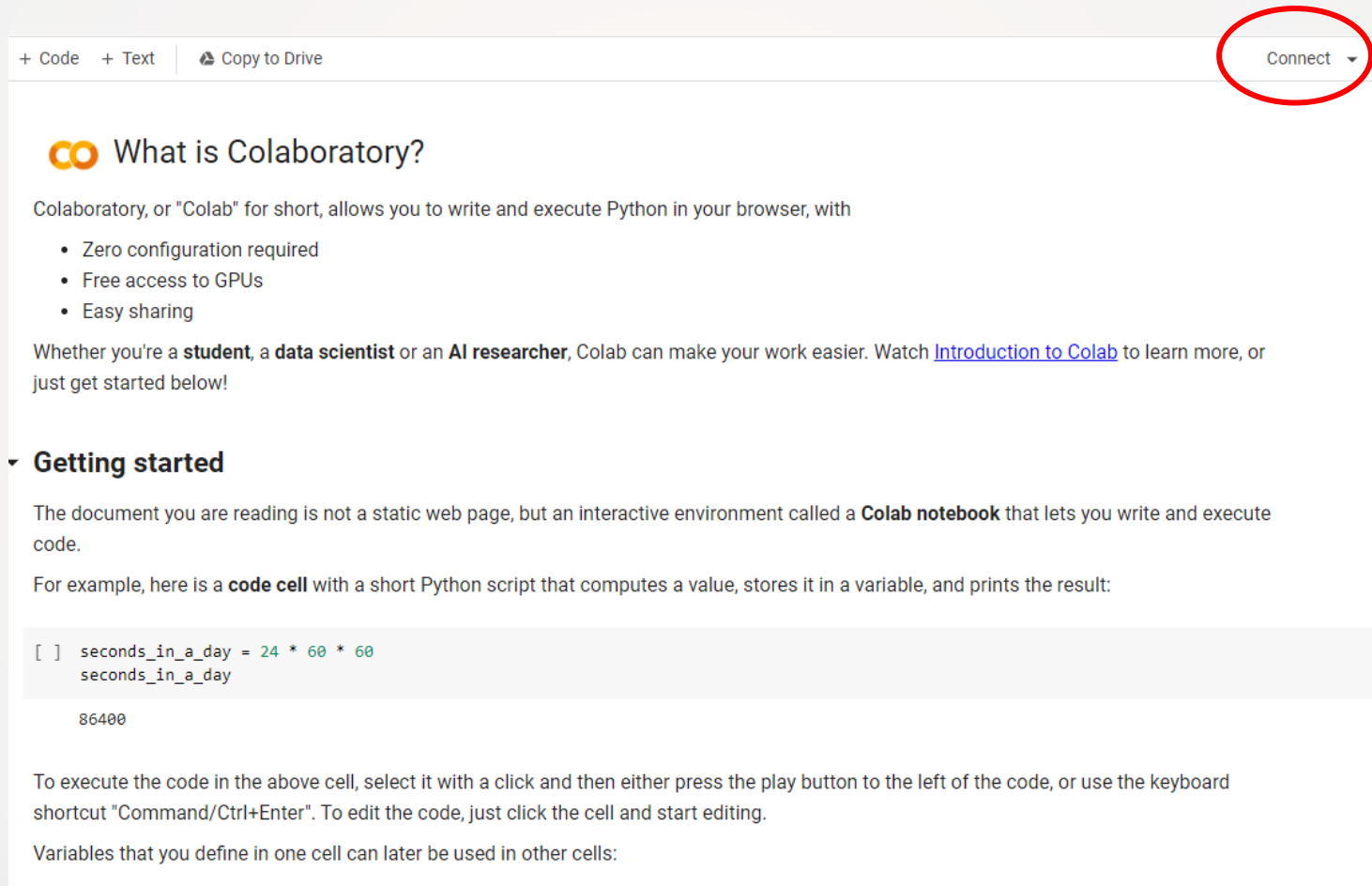
- Create / load models into basico
- Modify them
- Run simulations on them
- (potentially do parameter estimation)
- Know where to go for more information

# Installation

```
pip install copasi-basico
```

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

# Colab



The screenshot shows the top navigation bar of the Google Colaboratory interface. On the left, there are buttons for '+ Code' and '+ Text', and a 'Copy to Drive' icon. On the right, a 'Connect' button with a dropdown arrow is circled in red. Below the navigation bar, the main content area displays the Colab logo and the heading 'What is Colaboratory?'. The text explains that Colab allows writing and executing Python in a browser, listing benefits like zero configuration, free GPU access, and easy sharing. It also provides a link to an 'Introduction to Colab' video. A section titled 'Getting started' explains that the document is an interactive 'Colab notebook' and provides an example of a code cell. The code cell contains a Python script that calculates the number of seconds in a day (24 \* 60 \* 60) and prints the result, which is 86400. The text below the code cell explains how to execute and edit code cells.

+ Code + Text Copy to Drive **Connect**

## What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

### Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day

86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

# 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]:
```









	lower	upper	sol	affected
name				
(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]

Feedback / Suggestions

# User Forum

★ COPASI User Forum 460 members 1–30 of 1425 < >


□ ↻ ⋮

	manmat... , ... Fran... 8	LNA task execution in BASICO – Thank you so much, p...	Nov 13	☆
	Sebas..., pedro.me... 3	Simulating periodic input functions – Hi Pedro, Thanks...	Oct 29	☆
	manmathpa...@gmai... 3	LNA Task in Python interface – I am trying to execute L...	Oct 29	☆
	nfat...@bi... , pedro... 3	molecular weight – I was thinking of incorporating the ...	Oct 15	☆
	hikmete....., pedro... 3	An inhibitor acting on multiple enzymes – Dear Prof. M...	Oct 15	☆
	Severo III Balasbas	Fisher Information Matrix is Singular - Why? – Hello ev...	Oct 15	☆
	alice.de....., Frank ... 2	Axes limits when plotting Time Courses – We don't hav...	Aug 23	☆
	blinov	Biophysical Modeling of the Cell at Cell Bio 2024 – *** ...	Aug 2	☆

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

# de.NBI Survey



 de.NBI Training Course Evaluation

Please take the time to rate the de.NBI training course you have attended.

To offer you even better courses your input and opinion is very valuable for the continuous development of de.NBI training.

## 1. In this Workshop...

Do not agree with

Rather do not agree with

Rather agree with

Agree with

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

Thank you !