CNApy Users Guide

This manuscript ([permalink](#)) was automatically generated from [cnapy-org/users-guide@28bd12f](#) on July 8, 2021.

Authors

- **Sven Thiele**
  (iD) [0000-0002-5812-6963](#) · ⬡ [sthiele](#)
  Analysis and Redesign of Biological Networks, Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg

- **Axel von Kamp**
  · ⬡ [axelvonkamp](#)
  Analysis and Redesign of Biological Networks, Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg

Introduction

This is the user manual of CNApy. CNApy is a graphical interface for the modelling and analysis of metabolic networks on the basis of constraint-based (stoichiometric) modeling approaches. It allows

- the import/export of SBML models
- creating COBRApy/CNA models
- linking a graphical representation to the model data
- convenient exploration and editing of the model
- model analysis with standard and advanced constraint-based methods
- and saving everything as a CNApy *.cna project

The methods provided for model analysis (some of them are part of CNApy but most are functions from CNA and COBRApy) include:

- Flux balance analysis
- Parsimonious flux balance analysis
- Flux variability analysis
- Minimal cut sets
- Elementary modes
- Phase plane analysis
- Yield optimization

CNApy is available at https://github.com/cnapy-org/CNApy under the [Apache-2.0 License](#).

We appreciate any comments or suggestions for improvements and we are greatly interested in your feedback which you can give at our [User Forum](#).

For feature requests and bug reports please use our [Issue tracker](#)

Thank you for using CNApy!

Installation

The easiest way to install CNApy is using conda. First install conda and then:

- Create a conda environment with all dependencies

```
conda create -n cnapy-1.0.0 -c conda-forge -c cnapy cnapy=1.0.0
```

- Activate the cnapy conda environment

```
conda activate cnapy-1.0.0
```

- Run CNApy

```
cnapy
```

On windows you may also use the graphical installer.

Configuration

When we start CNApy for the first time we are greeted with a dialog like this.

Figure 1: CNApy Configuration dialog.

With this dialog you can configure which Matlab/Octave installation CNApy should use.

The minimal cut set computation in CNApy uses functions that are provided by CellNetAnalyzer. To be able to use these functions you have to provide CNApy with either a path to a Matlab installation >R2019 or the path to an Octave executable >4, and of course with a path to a recent CellNetAnalyzer installation. You can do this by clicking these buttons (1). If you change the settings CNApy performs some basic tests to make sure everything is working. If all checks are successful green check marks are shown, if a check fails a red cross is shown.

If no check can be performed, a yellow question mark is shown. For example, the check of the CNA directory needs a working Matlab or Octave installation.

You can also configure which colors CNApy should use to highlight reactions on the map or in the reaction list.

User interface overview

This section gives an introduction to the CNApy UI and its components and functions.

When we have finished our first configuration we are greeted with a window like this.

Figure 2: CNApy with an empty project.

This is an empty project on the right hand side we see the empty lists of reactions and metabolites. On the left we see the embedded Jupyter Console where one can interact programmatically with the model and UI. We can create a new project by importing SBML models, editing the reactions and adding graphical maps. We can also load one of the projects included in our projects repository.

Let's go to Project in our menubar and open the `ECC2comp.cna` project.
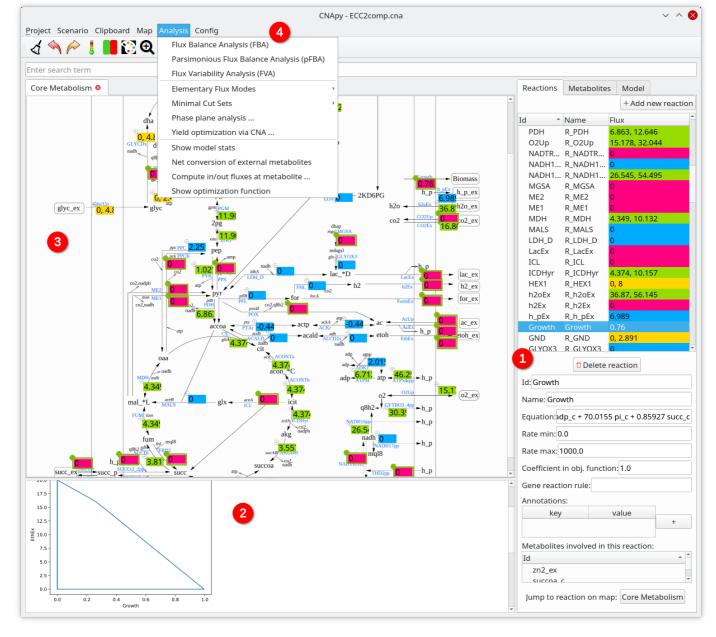
Figure 3: CNApy with the ECC2comp project.

In this picture we see CNApy with the open ECC2comp project. On the right hand side (1) we see the populated reactions and metabolites lists with the color coded current values and corresponding details for the selected reaction/metabolite. The reactions list contains buttons that lets us add/delete reactions to/from the model. The console (2) now shows the output of some computations, and above the console we have a map view (3) with a graphical representation of our network. On top (4) we have the menu bar which gives us access to the various functionalities of CNApy and a Toolbar for quick access to often used functions.

We can add new maps via the Map menu, and drag reactions from the reaction list onto the desired position on the map.

Create a new project

In this Section we explain how to create a new network project. We also have a video on that topic https://youtu.be/bsNXZBmtyWw.

CNApy starts with an empty Project. You can directly add new reactions via the reaction mask on the right. By pressing the *Add new reaction* button.

Then simply type the reaction formula into the Equation field. For example `A + 2 B -> C`.

The associated metabolites `A`, `B` and `C` are then automatically added to the model.

Changing metabolite identifiers in the metabolite form instantly rewrites the reaction equation of the associated reactions.

Creating big models by adding single reactions is very cumbersome. Therefore CNApy allows you to create a new project from an SBML file. You find the point *New project from SBML* in the *Project* menu.

After importing an SBML model the reaction list is populated with all the reactions from the model.

You can add a map to our project, by going to the *Map* menu and clicking *add new map*.

From the *Map* menu we can also *change the background image* of our map. Any SVG image can be used as a background.

You can add multiple maps that highlight different aspects of your model.

Reaction boxes can be put on the map by dragging over them from the reactions list.

You can adjust the box size with the keyboard shortcuts `Ctrl` + `+` and `Ctrl` + `-`. You can move the reaction box to the desired position by dragging them on their handles. If you want to move all reaction boxes together, press `Ctrl` while dragging.

Positioning a big number of reaction boxes can be tiresome. CNApy allows you to load and save reaction box positions. This facilitates reuse among similar projects.

The size of the background image can be adjusted with the shortcuts `Ctrl` + `Shift` + `+` and `Ctrl` + `Shift` + `-`.

To save a project click *Save Project as* in the *Project* menu. It is important to add the `.cna` extension to the file name. This way your files can be found by CNApy.

Scenarios

In CNApy you can define a scenario under which metabolic analysis like the FBA will be performed. In essence a scenario is a set of flux constraints for a set of reactions. These constraints can fix the flux of a reaction or constraint lower and upper bounds for the flux.

The scenario can be edited via the map, by entering values into the corresponding reaction boxes. Accepted values are either a single float like `1.5` or a pair of floats like `(-10, 1.2)`. A single float fixes the flux of the reaction to this value (`1.5`). While a pair sets the lower flux bound of the reaction to the first value (`-10`) and the upper flux bound to the second value (`1.2`).

Figure 4: Reaction box with scenario value.

Figure 4: Reaction box with scenario value.

Reactions boxes with scenario constraints are marked by the scenario color and a green frame. The frame turns yellow, if the scenario constraint contradicts the reaction constraints of the model.

To remove the constraints on a reaction simply delete the scenario value in the reaction box.

You can save and load scenarios as `*.scen` files. One scenario can be set as the default scenario of your project. This can be done via *set current scenario as default scenario* in the *Scenario* menu. The project must then be saved, and the next time you open the project the scenario is already set.

CNApy implements an edit history for the scenario with the tool buttons you can undo and redo your changes to the scenario. You can also import all the values that are currently in the reaction boxes into the scenario, and you can change the model's reaction bounds to the current scenario values.

Analysis functions

Flux balance analysis (FBA)

Optimizes the value of the current objective function and shows a flux distribution that realizes this optimum. The objective function is a linear function over the reaction rates in the network, its coefficients can be modified in the reaction dialogs. To display the current objective function you can choose *Show optimization function* from the *Analysis* menu. The current objective function in shown in the console. Note that the flux distribution calculated by FBA is usually not unique.

Parsimonious FBA (pFBA)

In parsimonious FBA first the objective function optimized and then with this optimum as additional constraint the sum of the fluxes in the network is minimized. This has the advantage that redundant internal cyclic fluxes are suppressed and that a lot of reactions will have zero flux which makes the solution easier to understand.

Flux variability analysis (FVA)

In FVA the maximal/minimal rate of each reaction is calculated. This gives the possible flux range for each reaction.

Elementary modes (EFM)/elementary flux vectors (EFV)

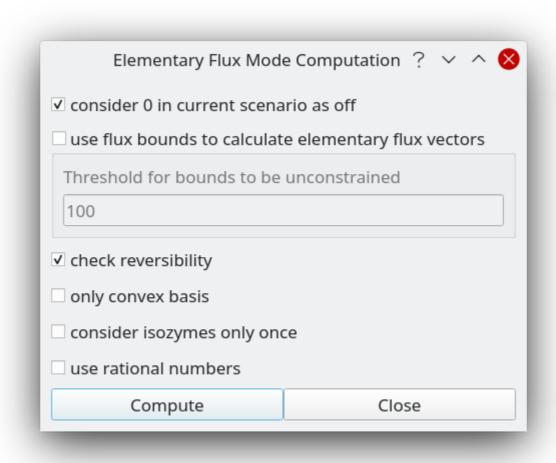EFM can be calculated via CNA or directly via efmtool while for EFV CNA is required.

Figure 5: Elementary Flux Mode Computation Dialog.

Options for CNA and efmtool

When reactions are marked with 0 flux in a scenario and the option "consider 0 in current scenario as off" is activated then only the subset of EFM/EFV is calculated in which these reactions do not participate

Additional options for CNA

- use flux bounds to calculate elementary flux vectors To calculate EFV instead of EFM, activate this option. Flux bounds defined in the reactions and the current scenario are then taken into account as inhomogeneous constraints. A flux bound is only included when its absolute value is greater than the value given as "Threshold for bounds to be unconstrained."

- check reversibility

Unchecking this option will lead to all reactions being considered reversible even if they have a rate minimum ≥ 0.

- only convex basis

Currently has no effect (need to be able to switch computation method first).

- consider isozymes only once

If a group of reactions has the same stoichiometry then only the modes with one representative of that group are calculated.

- use rational numbers

Use rational numbers of arbitrary length for EFM/EFV computation (efmtool part only, results will still appear as floating point numbers) To calculate EFV instead of EFM, activate this option. Flux bounds defined in the reactions and the current scenario are then taken into account as inhomogeneous constraints. A flux bound is only included when its absolute value is greater than the value given as "Threshold for bounds to be unconstrained."

Minimal cut sets

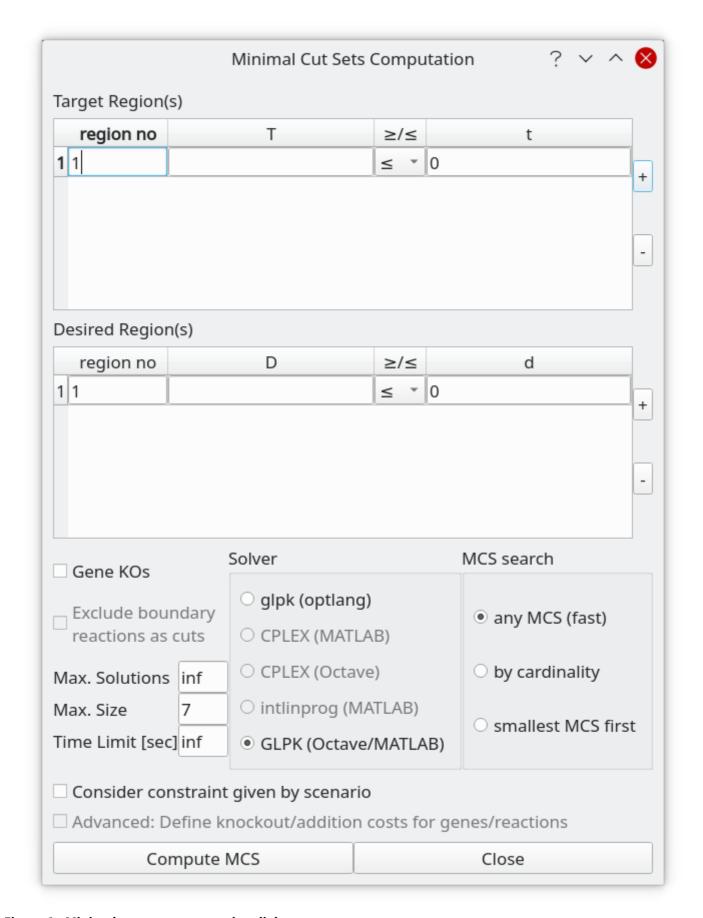Minimal cut sets (MCS) can be calculated in CNApy using the dual method.

Figure 6: Minimal cut sets computation dialog.

- Target and Desired region(s)

For the dual method one or more target regions must be defined which describe the network behaviour that is to be suppressed. In addition, one or more desired regions may be defined which describe the network behaviour that is to be preserved. Each target/desired region is defined by a set

of linear equality constraints over the reaction rates in the network. All target and desired regions must be initially feasible or an error will be raised. Also make sure that 0 is not included in any target or desired region (0 as target cannot be suppressed and 0 as desired is always fulfilled).

Note that all non-default reaction bounds (as defined by cobra.Configuration().bounds) are automatically added to all target and desired regions before MCS computation.

- Solver

MCS can either be calculated via CNA (using either Octave or Matlab) or via optlang using different MILP solvers. For CNA the solver must be chosen while optlang uses the solver for the current model (cf. Solver for current model). Not all variants and solvers allow for the same options so these will partly be disabled depending on the chosen solver.

- Max. solutions

Maximal number of MCS to calculate.

- Max. size

Only calculate MCS which consists of up to this number of cuts.

- Time limit

Do not run the search for longer than this limit. It is strongly recommended to use this parameter because MCS computation can take very long especially when searching for smallest MCS.

- MCS search

MCS can be enumerated iteratively, either the smallest first (least number of cuts) or any MCS (up to the number of cuts given by max. size). With the latter option new MCS are found more quickly but may not be the smallest.

With CPLEX or Gurobi as solver, MCS can also be enumerated by cardinality, i.e. all MCS of successively increasing size (up to max. size) are computed. In addition, with these solvers a continuous search mode is possible which is similar to the any MCS method but often faster.

- Gene KOs (CNA only)

If the model contains the gene-reaction association, search for gene knock-outs instead of reaction knock-outs.

- Exclude boundary reactions as cuts (optlang only)

Do not allow knock-out of boundary reactions (reactions that cross the system boundary, e.g. exchange reactions).

- Consider constraints given by scenario

Take reaction bounds set by the current scenario into account for all target and desired regions.

Yield optimization

Figure 7: Yield optimization dialog.

Maximizes the value of a yield function *c/d* where *c* and *d* are linear expressions over the fluxes in the network. For this to work correctly, there must be no flux vector in the network where the denominator becomes zero and the nominator is non-zero. The flux values of the current scenario are taken into during the optimization.

A typical application would be the maximization of a product yield: When the exchange fluxes for ethanol and glucose are *EX_etoh* and *EX_glc* then maximizing *EX_etoh/-EX_glc gives* the maximal ethanol yield on glucose.
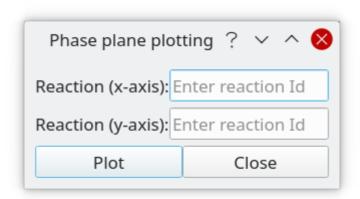
Phase plane plot

Figure 8: Phase plane plot dialog.

Creates a plot (in the console) that shows the projection of the solution space onto a two-dimensional plane of two selected reaction rates. The flux values of the current scenario are taken into account during the calculations.

Compute in/out fluxes at a metabolite

If a flux distribution has been calculated (e.g. by FBA) this function shows the magnitude of all fluxes going in/coming out of a specified metabolite. The reaction fluxes are displayed as stacked bar graphs

on the console. You can either call this function from the Analysis menu and specify a metabolite or right-click on a metabolite in the list and call the function from the context menu.
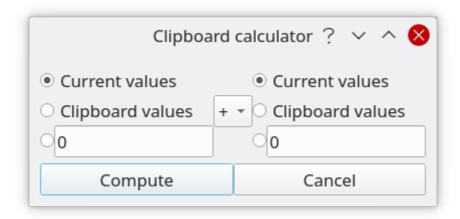
Clipboard calculator

Figure 9: Clipboard calculator.

The clipboard calculator allows you to perform arithmetic operations with the values stored in the clipboard, the current reaction rates or a fixed value that you can enter in this dialog. The result of the operation replaces the current flux values.

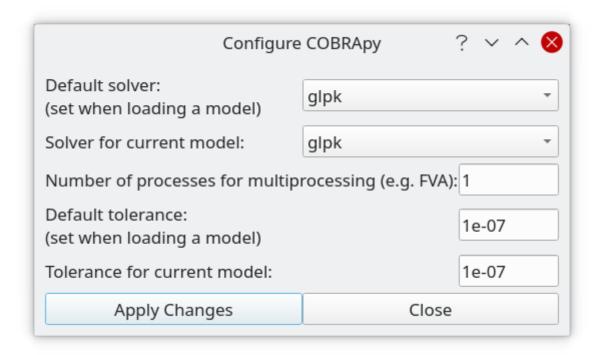## Configure COBRApy

Figure 10: COBRApy configuration dialog.

With this dialog selected global settings (`class cobra.Configuration()`) of the COBRApy toolbox and selected parameters of the current model can be modified. The global settings are saved, but the parameters of the current model are not and will revert to the global default values are (re-)loading a model.

- Default solver (global)

Select which solver to use as global default from the dropdown list. COBRApy supports a variety of solvers but only the ones which are installed for COBRApy on your system will be available on the list. Also, different solvers have different capabilities so you should set a default that has all capabilities that you require (e.g. glpk_exact cannot be used for MILPs).

- Solver for current model

Here you can change the solver for the current model.

- Number of processes for multiprocessing (global)

Number of processes run in parallel when Python multiprocessing is used, e.g. for FVA. Setting this to 1 disables multiprocessing.

> **ⓘ IMPORTANT**
> Should be set to 1 on Windows because Python multiprocessing performs very poorly on this OS.

- Default tolerance (global)

This value is used to distinguish zero from non-zero elements for various purposes. In particular, this value is propagated to the solver (see the COBRApy source code which solver parameters are affected). Most solvers restrict the range of this value, so it must be between 1E-9 and 0.1 For details where this value is used see the COBRApy documentation and source code.

- Tolerance for the current model

Here you can change the tolerance for the current model. You may enter an arbitrary value >= 0 here but if it is not compatible with the current solver then "Apply Changes" can lead to an error message.

Programming CNApy

Under Construction

- accessing the COBRApy Model
- accessing scenario and computed values
- accessing the CNApy UI

The rest of this document is a full list of formatting elements/features supported by Manubot. Compare the input (`.md` files in the `/content` directory) to the output you see below.

Basic formatting

Bold text

Semi-bold text

<div align="center">Centered text</div>

<div align="right">Right-aligned text</div>

Italic text

Combined *italics and **bold***

~~Strikethrough~~

1. Ordered list item
2. Ordered list item
    a. Sub-item
    b. Sub-item
        i. Sub-sub-item
3. Ordered list item
    a. Sub-item

- List item
- List item
- List item

subscript: H_2O is a liquid

superscript: 2^{10} is 1024.

[unicode superscripts]^{0123456789}

[unicode subscripts]$_{0123456789}$

A long paragraph of text. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Putting each sentence on its own line has numerous benefits with regard to [editing] and [version control].

Line break without starting a new paragraph by putting
two spaces at end of line.

Document organization

Document section headings:

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

A heading centered on its own printed page

Horizontal rule:

`Heading 1`'s are recommended to be reserved for the title of the manuscript.

`Heading 2`'s are recommended for broad sections such as *Abstract*, *Methods*, *Conclusion*, etc.

`Heading 3`'s and `Heading 4`'s are recommended for sub-sections.

Links

Bare URL link: https://manubot.org

Long link with lots of words and stuff and junk and bleep and blah and stuff and other stuff and more stuff yeah

Link with text

Link with hover text

Link by reference

Citations

Citation by DOI [1].

Citation by PubMed Central ID [2].

Citation by PubMed ID [3].

Citation by Wikidata ID [4].

Citation by ISBN [5].

Citation by URL [6].

Citation by alias [7].

Multiple citations can be put inside the same set of brackets [1,5,7]. Manubot plugins provide easier, more convenient visualization of and navigation between citations [2,3,7,8].

Citation tags (i.e. aliases) can be defined in their own paragraphs using Markdown's reference link syntax:

Referencing figures, tables, equations

Figure 11

Figure 12

Quotes and code

> Quoted text

> Quoted block of text
>
> Two roads diverged in a wood, and I—
> I took the one less traveled by,
> And that has made all the difference.

Code `in the middle` of normal text, aka `inline code`.

Code block with Python syntax highlighting:

```python
from manubot.cite.doi import expand_short_doi

def test_expand_short_doi():
    doi = expand_short_doi("10/c3bp")
    # a string too long to fit within page:
    assert doi == "10.25313/2524-2695-2018-3-vliyanie-enhansera-copia-i-
        insulyatora-gypsy-na-sintez-ernk-modifikatsii-hromatina-i-
        svyazyvanie-insulyatornyh-belkov-vtransfetsirovannyh-geneticheskih-
        konstruktsiyah"
```

Code block with no syntax highlighting:

```
Exporting HTML manuscript
Exporting DOCX manuscript
Exporting PDF manuscript
```

Figures

Figure 11: A square image at actual size and with a bottom caption. Loaded from the latest version of image on GitHub.

Figure 12: An image too wide to fit within page at full size. Loaded from a specific (hashed) version of the image on GitHub.

Figure 13: A tall image with a specified height. Loaded from a specific (hashed) version of the image on GitHub.

Figure 14: A vector `.svg` image loaded from GitHub. The parameter `sanitize=true` is necessary to properly load SVGs hosted via GitHub URLs. White background specified to serve as a backdrop for transparent sections of the image.

Tables

Table 1: A table with a top caption and specified relative column widths.

Bowling Scores	Jane	John	Alice	Bob
Game 1	150	187	210	105
Game 2	98	202	197	102
Game 3	123	180	238	134

Table 2: A table too wide to fit within page.

	Digits 1-33	Digits 34-66	Digits 67-99	Ref.
pi	3.14159265358979323 846264338327950	28841971693993751 0582097494459230	78164062862089986 2803482534211706	`piday.org`
e	2.71828182845904523 536028747135266	24977572470936999 5957496696762772	40766303535475945 7138217852516642	`nasa.gov`

Table 3: A table with merged cells using the `attributes` plugin.

Size	Colors	
	Text Color	**Background Color**
Size	**Text Color**	**Background Color**
big	blue	orange
small	black	white

Equations

A LaTeX equation:

$$\int_0^\infty e^{-x^2}\,dx = \frac{\sqrt{\pi}}{2} \tag{1}$$

An equation too long to fit within page:

$$x = a + b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t \\ + u + v + w + x + y + z + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 \tag{2}$$

Special

⚠ **WARNING** *The following features are only supported and intended for* `.html` *and* `.pdf` *exports. Journals are not likely to support them, and they may not display correctly when converted to other formats such as* `.docx`*.*

> **LINK STYLED AS A BUTTON**

Adding arbitrary HTML attributes to an element using Pandoc's attribute syntax:

> Manubot Manubot Manubot Manubot Manubot. Manubot Manubot Manubot Manubot. Manubot Manubot Manubot. Manubot Manubot. Manubot.

Adding arbitrary HTML attributes to an element with the Manubot `attributes` plugin (more flexible than Pandoc's method in terms of which elements you can add attributes to):

> Manubot Manubot Manubot Manubot Manubot. Manubot Manubot Manubot Manubot. Manubot Manubot Manubot. Manubot Manubot. Manubot.

Available background colors for text, images, code, banners, etc:

`white` `lightgrey` `grey` `darkgrey` `black` `lightred` `lightyellow` `lightgreen` `lightblue` `lightpurple` `red` `orange` `yellow` `green` `blue` `purple`

Using the [Font Awesome](#) icon set:

✔ ? ★ 🔔 ✖ ⋯

> 🏛 **Light Grey Banner**
> useful for *general information* - manubot.org

> ℹ️ **Blue Banner**
> useful for *important information* - manubot.org

> 🚫 **Light Red Banner**
> useful for *warnings* - manubot.org

References

1. **Sci-Hub provides access to nearly all scholarly literature**
   Daniel S Himmelstein, Ariel Rodriguez Romero, Jacob G Levernier, Thomas Anthony Munro, Stephen Reid McLaughlin, Bastian Greshake Tzovaras, Casey S Greene
   eLife (2018-03-01) https://doi.org/ckcj
   DOI: 10.7554/elife.32822 · PMID: 29424689 · PMCID: PMC5832410

2. **Reproducibility of computational workflows is automated using continuous analysis**
   Brett K Beaulieu-Jones, Casey S Greene
   Nature biotechnology (2017-04) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6103790/
   DOI: 10.1038/nbt.3780 · PMID: 28288103 · PMCID: PMC6103790

3. **Bitcoin for the biological literature.**
   Douglas Heaven
   Nature (2019-02) https://www.ncbi.nlm.nih.gov/pubmed/30718888
   DOI: 10.1038/d41586-019-00447-9 · PMID: 30718888

4. **Plan S: Accelerating the transition to full and immediate Open Access to scientific publications**
   cOAlition S
   (2018-09-04) https://www.wikidata.org/wiki/Q56458321

5. **Open access**
   Peter Suber
   MIT Press (2012)
   ISBN: 9780262517638

6. **Open collaborative writing with Manubot**
   Daniel S Himmelstein, Vincent Rubinetti, David R Slochower, Dongbo Hu, Venkat S Malladi, Casey S Greene, Anthony Gitter
   Manubot (2020-05-25) https://greenelab.github.io/meta-review/

7. **Opportunities and obstacles for deep learning in biology and medicine**
   Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, … Casey S Greene
   Journal of The Royal Society Interface (2018-04-04) https://doi.org/gddkhn
   DOI: 10.1098/rsif.2017.0387 · PMID: 29618526 · PMCID: PMC5938574

8. **Open collaborative writing with Manubot**
   Daniel S Himmelstein, Vincent Rubinetti, David R Slochower, Dongbo Hu, Venkat S Malladi, Casey S Greene, Anthony Gitter
   PLOS Computational Biology (2019-06-24) https://doi.org/c7np
   DOI: 10.1371/journal.pcbi.1007128 · PMID: 31233491 · PMCID: PMC6611653