

Reproducible Carbon Cycle Models

Biogeochemical Model Database `bgc_md2`



Markus Müller, Holger Metzler, Verónica Ceballos Núñez, Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou, Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang, Alison Bennett, Chenyu Bian, Lifeng Jiang, Song Wang, Chengcheng Gang, Carlos Sierra, Yiqi Luo

Why would **you** want a model data base?



Why would **you** want a model data base?

- **Find** (rather than reinvent) the **right model** for a specific **task**?



Why would **you** want a model data base?

- **Find** (rather than reinvent) the **right model** for a specific **task**?
- **Implement** a **new** model but start from a similar one?



Why would **you** want a model data base?

- **Find** (rather than reinvent) the **right model** for a specific **task**?
- **Implement** a **new** model but start from a similar one?
- **Use common infrastructure** to compute diagnostics that are difficult to implement.

Why would **you** want a model data base?

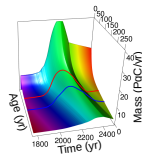
- **Find** (rather than reinvent) the **right model** for a specific **task**?
- **Implement** a **new** model but start from a similar one?
- **Use common infrastructure** to compute diagnostics that are difficult to implement. e.g. carbon flux diagrams, transit times, age distributions . . .

$$\frac{d}{dt} \begin{bmatrix} \text{leaf} \\ \text{wood} \end{bmatrix} = \begin{bmatrix} l_{\text{leaf}}(t) \\ l_{\text{wood}} \end{bmatrix} + \begin{bmatrix} -k_{\text{leaf} \rightarrow \text{wood}} - k_{\text{leaf},0}(t) & k_{\text{wood} \rightarrow \text{leaf}} \\ k_{\text{leaf} \rightarrow \text{wood}} & -k_{\text{wood} \rightarrow \text{leaf}} - k_{\text{wood},0} \end{bmatrix} \begin{bmatrix} \text{leaf} \\ \text{wood} \end{bmatrix}$$

leaf wood



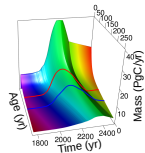
leaf wood lit som cwd



Why would **you** want a model data base?

- **Find** (rather than reinvent) the **right model** for a specific **task**?
- **Implement** a **new** model but start from a similar one?
- **Use common infrastructure** to compute diagnostics that are difficult to implement. e.g. carbon flux diagrams, transit times, age distributions . . .

$$\frac{d}{dt} \begin{bmatrix} \text{leaf} \\ \text{wood} \end{bmatrix} = \begin{bmatrix} I_{\text{leaf}}(t) \\ 0 \end{bmatrix} + \begin{bmatrix} -k_{\text{leaf} \rightarrow \text{wood}} - k_{\text{leaf} \rightarrow \text{lit}} & k_{\text{wood} \rightarrow \text{leaf}} \\ k_{\text{leaf} \rightarrow \text{wood}} & -k_{\text{wood} \rightarrow \text{leaf}} - k_{\text{wood} \rightarrow \text{cwd}} \end{bmatrix} \begin{bmatrix} \text{leaf} \\ \text{wood} \end{bmatrix}$$

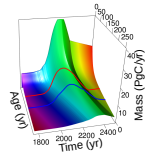


- Find and **reduce** sources of **uncertainty** in carbon predictions

Why would **you** want a model data base?

- **Find** (rather than reinvent) the **right model** for a specific **task**?
- **Implement** a **new** model but start from a similar one?
- **Use common infrastructure** to compute diagnostics that are difficult to implement. e.g. carbon flux diagrams, transit times, age distributions . . .

$$\frac{d}{dt} \begin{bmatrix} \text{leaf} \\ \text{wood} \end{bmatrix} = \begin{bmatrix} I_{\text{leaf}}(t) \\ I_{\text{wood}} \end{bmatrix} + \begin{bmatrix} -k_{\text{leaf} \rightarrow \text{wood}} - k_{\text{leaf},0}(t) & k_{\text{wood} \rightarrow \text{leaf}} \\ k_{\text{leaf} \rightarrow \text{wood}} & -k_{\text{wood} \rightarrow \text{leaf}} - k_{\text{wood},0} \end{bmatrix} \begin{bmatrix} \text{leaf} \\ \text{wood} \end{bmatrix}$$



- Find and **reduce** sources of **uncertainty** in carbon predictions
- ... ???

Biogeochemical Model Database bgc_md2



Markus Müller, Holger Metzler, Verónica Ceballos Núñez,
Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou,
Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang,
Alison Bennett, Chenyu Bian, Lifeng Jiang, Song Wang,
Chengcheng Gang, Carlos Sierra, Yiqi Luo

How can you build one?

We need:

- **collections**



How can you build one?

We need:

- **collections**

- ▶ of **many models**: (from A to Z):

Arora2005GCB-1 ,CARDAMOM ,. . . ,

Zelenev2000MicrobialEcology

How can you build one?

We need:

■ collections

- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM ,. . . ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by

We need:

- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM , . . . ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by

We need:

- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM , . . . ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by

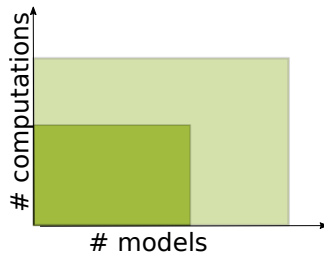
- 'Copy and paste'?
 - bilinear increase of code
 - unmaintainable, untestable, errorprone, hard to change ...

How can you build one?

We need:

- **collections**

- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM ,... ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by



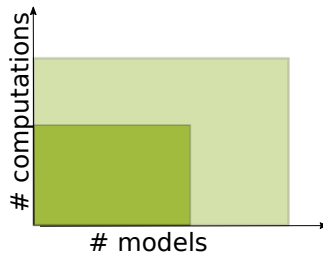
- 'Copy and paste'?
 - bilinear increase of code
 - unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
Don't repeat yourself

How can you build one?

We need:

- **collections**

- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM ,... ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by



- 'Copy and paste'?
 - bilinear increase of code
 - unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
Don't repeat yourself

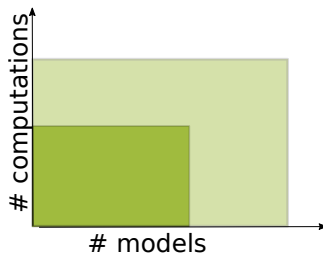
How can you build one?

We need:

- **collections**

- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM ,... ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by

- Ways of **organizing** both collections



- 'Copy and paste'?
 - bilinear increase of code
 - unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
Don't repeat yourself

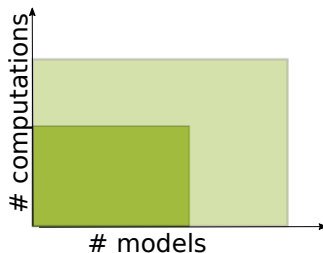
How can you build one?

We need:

■ collections

- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM ,... ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by

- Ways of **organizing** both collections
 - ▶ building blocks for models



- 'Copy and paste'?
 - bilinear increase of code
 - unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
Don't repeat yourself

How can you build one?

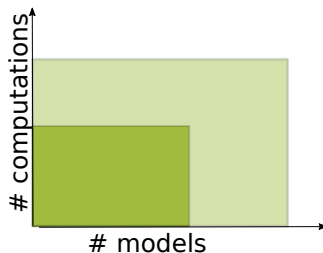
We need:

■ collections

- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM ,... ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by

■ Ways of **organizing** both collections

- ▶ building blocks for models
- ▶ functions of building blocks



- 'Copy and paste'?
 - bilinear increase of code
 - unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
Don't repeat yourself

We need:

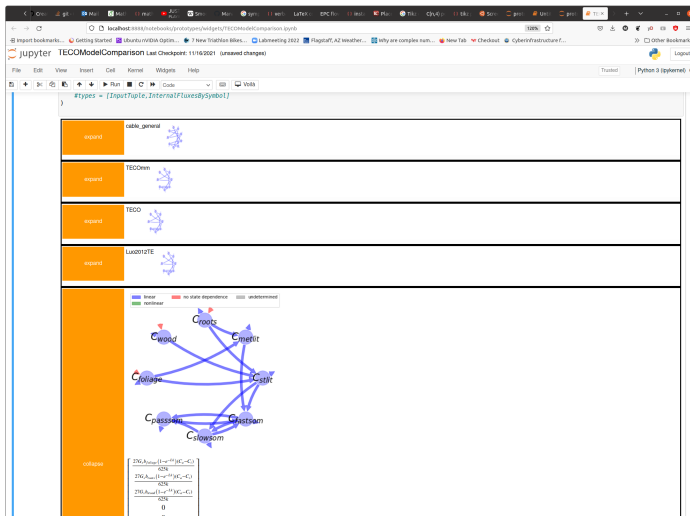
- ▶ of **many models**: (from A to Z):
Arora2005GCB-1 ,CARDAMOM , . . . ,
Zelenev2000MicrobialEcology
- ▶ of **many** computable **properties** and diagnostics to **compare** them by

- ▶ building blocks for models
- ▶ functions of building blocks
- ▶ **computational graph** for **different / evolving** building blocks

- 'Copy and paste'?
 - bilinear increase of code
 - unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
Don't repeat yourself



Example widget for query result



Analysis with symbolic tools (sympy) ...

The screenshot displays a Jupyter Notebook environment with the following content:

- Code Cell:** Imports `inspectModel` and defines a function `inputtuple, compartmentalMatrix, compartmentalMatrix, infFluxesBySymbol, inputtuple, internalFluxesBySymbol, smoothReservoirModel, outFluxesBySymbol, compartmentalMatrix, infFluxesBySymbol, inputtuple, internalFluxesBySymbol, smoothReservoirModel, outFluxesBySymbol`.
- Output [3]:** A network diagram showing carbon pools (C_{wood} , C_{root} , C_{leaf} , C_{DPM} , C_{RPM} , C_{HUM} , C_{BIO}) and their interactions with arrows indicating fluxes.
- Code Cell:** Executes `mvs.get_CompartmentalMatrix()`.
- Output [28]:** A large matrix representing the compartmental matrix, showing various fluxes and their dependencies on state variables like $\xi(t)$.
- Code Cell:** Executes `mvs.get_BibInfo()` and displays a list of attributes including `get_CompartmentalMatrix`, `get_InfFluxesBySymbol`, `get_InputTuple`, `get_InternalFluxesBySymbol`, `get_OutFluxesBySymbol`, `get_SmoothReservoirModel`, and `get_StateVariableTuple`.
- Code Cell:** Executes `mvs.get_StateVariableTuple()` and displays a list of state variables including `get_BibInfo`, `get_CompartmentalMatrix`, `get_InfFluxesBySymbol`, `get_InputTuple`, `get_InternalFluxesBySymbol`, `get_OutFluxesBySymbol`, `get_SmoothReservoirModel`, and `get_StateVariableTuple`.

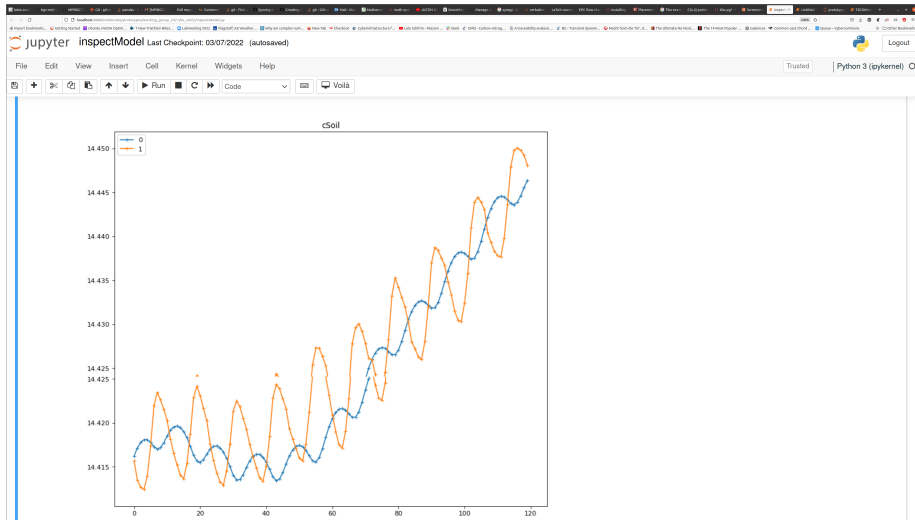
Reproducible Carbon Cycle Models

Biogeochemical Model Database `bgc_md2`



Markus Müller, Holger Metzler, Verónica Ceballos Núñez, Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou, Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang, Alison Bennett, Chenyu Bian, Lifan Jiang, Song Wang, Chengcheng Gang, Carlos Sierra, Yiqi Luo

...or numerically



Reproducible Carbon Cycle Models

Biogeochemical Model Database `bgc_md2`

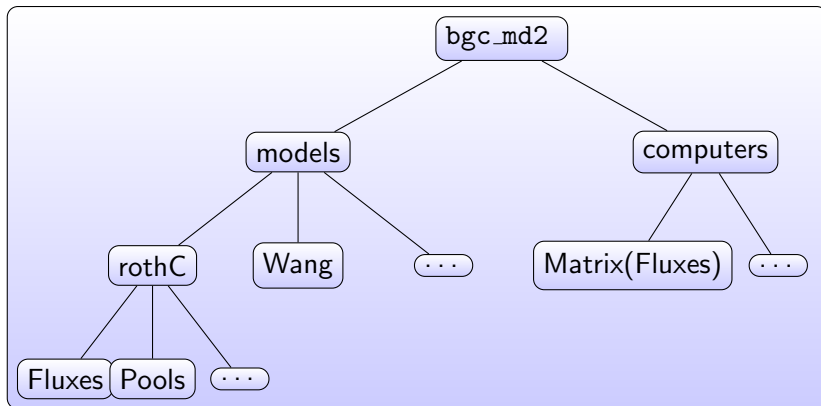


Markus Müller, Holger Metzler, Verónica Ceballos Núñez, Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou, Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang, Alison Bennett, Chenyu Bian, Lifeng Jiang, Song Wang, Chengcheng Gang, Carlos Sierra, Yiqi Luo

Database records are python modules

```
1 from sympy import Symbol, Function
2 from CerebralBiologyModels.OHMS import OHMS
3 from bgc_md2.helper import module_computers
4 from bgc_md2.models.BBInfo import BBInfo
5 from bgc_md2.resolver.mwars import (
6     InfluxesBySymbol,
7     OutFluxesBySymbol,
8     InternalFluxesBySymbol,
9     TimesSymbol,
10     StateVariableTable,
11 )
12 import bgc_md2.resolver.computers as bgc_c
13
14 # Make a small dictionary for the variables we will use
15 sym_dict = {
16     'r_leaf_2_w': 'Internal flux rate from leaf to wood',
17     'r_w_2_l': 'Internal flux rate from wood to leaf',
18     'C_soil_fast': '',
19     'C_soil_slow': '',
20     'C_soil_passive': '',
21     'C_leaf': '',
22     'C_root': '',
23     'C_wood': '',
24     'C_leaf_litter': '',
25     'C_root_litter': '',
26     'C_wood_litter': '',
27     'r_C_leaf_2_C_leaf_litter': '',
28     'r_C_root_2_C_root_litter': '',
29     'r_C_wood_2_C_wood_litter': '',
30     'r_C_leaf_litter_rh': '',
31     'r_C_root_litter_rh': '',
32     'r_C_wood_litter_rh': '',
33     'r_C_soil_fast_rh': '',
34     'r_C_soil_slow_rh': '',
35     'r_C_soil_passive_rh': '',
36     'r_C_leaf_litter_2_C_soil_fast': '',
37     'r_C_leaf_litter_2_C_soil_slow': '',
38     'r_C_leaf_litter_2_C_soil_passive': '',
39     'r_C_wood_litter_2_C_soil_fast': '',
40     'r_C_wood_litter_2_C_soil_slow': '',
41     'r_C_wood_litter_2_C_soil_passive': '',
42     'r_C_root_litter_2_C_soil_fast': '',
43     'r_C_root_litter_2_C_soil_slow': '',
44     'r_C_root_litter_2_C_soil_passive': '',
45     'tair': 'Air temperature',
46     'tsoil': 'Soil temperature',
47     'tveg': 'Vegetation temperature',
48     'tveg': 'Vegetation temperature',
49     'tveg': 'Vegetation temperature',
50     'beta_leaf': '',
51     'beta_wood': '',
52 }
53
54 # For k in func_dict.keys():
55     codekv = sym_dict['(k)'].format(k)
56     exec(code)
57
58 # some we will also use some symbols for functions (which appear with an argument)
59 func_dict = {
60     'tair': 'A scalar function of temperature and moisture and thereby ultimately of time',
61     'tsoil': 'Soil temperature',
62 }
63
64 # For k in func_dict.keys():
65     codekv = Function('f')(k).format(k)
66     exec(code)
67
68 # to timesymbol('t')
69 beta_root = 1 - (beta_leaf + beta_wood)
70 mw = OHMS()
71
72 # StateVariableTable
73 C_leaf = Symbol('C_leaf')
74 C_wood = Symbol('C_wood')
75 C_soil_fast = Symbol('C_soil_fast')
76 C_soil_slow = Symbol('C_soil_slow')
77 C_soil_passive = Symbol('C_soil_passive')
78 C_leaf_litter = Symbol('C_leaf_litter')
79 C_root_litter = Symbol('C_root_litter')
80 C_wood_litter = Symbol('C_wood_litter')
81
82 InfluxesBySymbol(
83     {
84         C_leaf: NP(t) * beta_leaf,
85         C_root: NP(t) * beta_root,
86         C_wood: NP(t) * beta_wood
87     }
88 )
89 OutFluxesBySymbol(
90     {
91         C_leaf_litter: r_C_leaf_litter_rh * C_leaf_litter * tair,
92         C_wood_litter: r_C_wood_litter_rh * C_wood_litter * tair,
93         C_root_litter: r_C_root_litter_rh * C_root_litter * tair,
94         C_soil_fast: r_C_soil_fast_rh * C_soil_fast * tsoil,
95         C_soil_slow: r_C_soil_slow_rh * C_soil_slow * tsoil,
96         C_soil_passive: r_C_soil_passive_rh * C_soil_passive * tsoil,
97     }
98 )
99 InternalFluxesBySymbol(
100     {
101         (C_leaf, C_leaf_litter): r_C_leaf_2_C_leaf_litter * C_leaf,
102         (C_wood, C_wood_litter): r_C_wood_2_C_wood_litter * C_wood,
103         (C_root, C_root_litter): r_C_root_2_C_root_litter * C_root,
104         (C_leaf_litter, C_soil_fast): r_C_leaf_litter_2_C_soil_fast * C_leaf_litter * tair,
105         (C_leaf_litter, C_soil_slow): r_C_leaf_litter_2_C_soil_slow * C_leaf_litter * tair,
106         (C_leaf_litter, C_soil_passive): r_C_leaf_litter_2_C_soil_passive * C_leaf_litter * tair,
107         (C_wood_litter, C_soil_fast): r_C_wood_litter_2_C_soil_fast * C_wood_litter * tair,
108         (C_wood_litter, C_soil_slow): r_C_wood_litter_2_C_soil_slow * C_wood_litter * tair,
109         (C_wood_litter, C_soil_passive): r_C_wood_litter_2_C_soil_passive * C_wood_litter * tair,
110         (C_root_litter, C_soil_fast): r_C_root_litter_2_C_soil_fast * C_root_litter * tair,
111         (C_root_litter, C_soil_slow): r_C_root_litter_2_C_soil_slow * C_root_litter * tair,
112         (C_root_litter, C_soil_passive): r_C_root_litter_2_C_soil_passive * C_root_litter * tair,
113     }
114 )
115 BBInfo = BBInfo(logical_information=
116     {
117         'longname': '
118         'version': '
119         'entryAuthor': 'Kostiantyn Viatkin',
120         'entryAuthorId': '
121         'entryCreatedDate': '
122         'doi': '
123     }
124 )
```

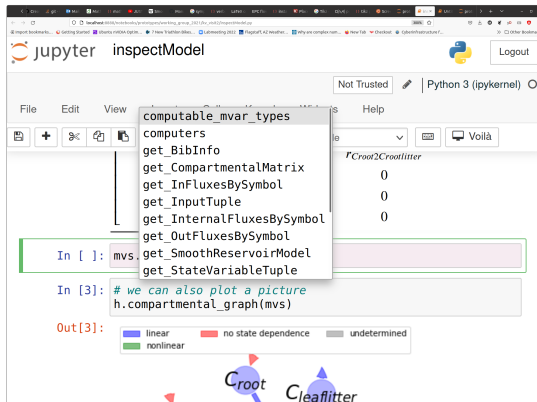
Internal Structure of `bgc_md2`



The `bgc_md` library provides I:

- 1 Datatypes defining **building blocks** of models e.g.
`CompartmentalMatrix`, `InternalFluxesBySymbol`, ...
- 2 Functions operating on those properties (forming the edges of the graph where the Datatypes are nodes)
- 3 A user interface based on graph algorithms to
 - 1 compute the set of computable properties (e.g. the comparable criteria for a set of models, database queries)
 - 2 actually compute the desired properties by recursively connecting several function applications.
 - 3 show what is missing to compute a desired property.

Userinterface using computability graphs

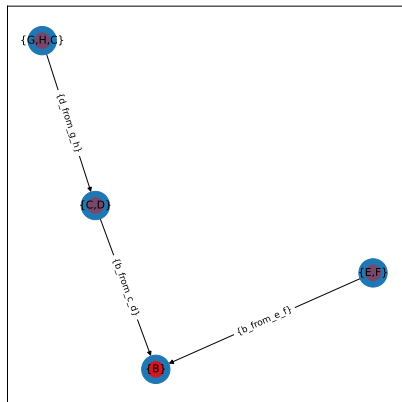


Suggested methods automatically created by ComputabilityGraphs



Finding what's missing in the model description

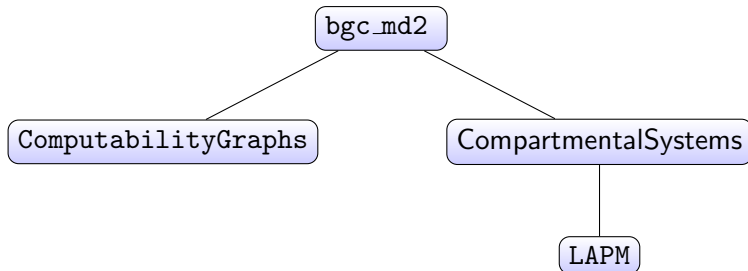
given a set of
functions:
 $a(i)$, $b(c,d)$, $b(e,f)$,
 $c(b)$, $d(b)$, $d(g,h)$,
 $e(b)$, $f(b)$ and the
target variable **B** e.g.
CompartmentalMatrix,
The algorithm
computes all possible
combinations and
paths from which **B**
can be computed.



The `bgc_md` library provides 11:

- 1 30+ vegetation, soil or ecosystem models for carbon and nitrogen cycling as reusable python modules using the building blocks in a flexible way.
- 2 An interface to *many algorithms* in CompartmentalSystems to compute diagnostic variables for *many models* in bgc_md2.

Relation to other Python Packages





Example computation via CompartmentalSystems

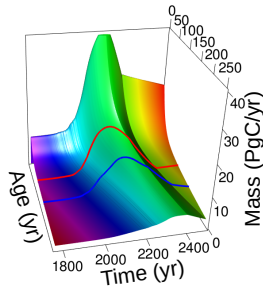


Figure: age distribution of a pool as function of time



Metzler, H., Müller, M., and Sierra, C. (2018).

Transit-time and age distributions for nonlinear time-dependent compartmental systems.

Proceedings of the National Academy of Sciences, 115:201705296.

Links

- The README of the package on github (with installation instructions): https://github.com/MPIBGC-TEE/bgc_md2
- To **explore** some rudimentary tutorials **without installation** use https://mybinder.org/v2/gh/MPIBGC-TEE/bgc_md2/binder
 - ▶ Click on the link!
 - ▶ After jupyter lab has started go to `/binder_notebooks/illustrativeExamples/`
 - ▶ right click on `createModel.py`
 - ▶ choose Open With
 - ▶ choose Jupyter Notebook

This will open an example notebook exploring some of the concepts. More applied examples are coming.