Markus Müller, Holger Metzler, Verónika Ceballos Núñez, Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou, Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang, Alison Bennett, Chenyu Bian, Lifen 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?

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

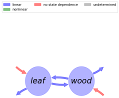
Why would **you** want a model data base?

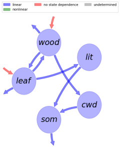
- **Find** (rather than reinvent) the **right model** for a specific **task**?
- **Implement** a **new** model but start from a similar one?
- **Use common infrastructure** to compute diagnostics that are difficult to implement.

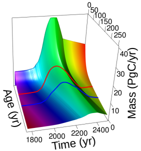
Why would **you** want a model data base?

- **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 . . .

$$\frac{d}{dt}\begin{bmatrix} leaf \\ wood \end{bmatrix} = \begin{bmatrix} I_{leaf}(t) \\ I_{wood} \end{bmatrix} + \begin{bmatrix} -k_{leaf2wood} - k_{leafo}(t) & k_{wood2leaf} \\ k_{leaf2wood} & -k_{wood2leaf} - k_{woodo} \end{bmatrix} \begin{bmatrix} leaf \\ wood \end{bmatrix}$$





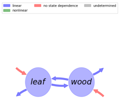


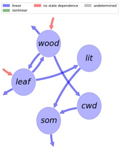
Markus Müller, Holger Metzler, Verónika Ceballos Núñez, Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou, Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang, Alison Bennett, Chenyu Bian, Lifen Jiang, Song Wang, Chengcheng Gang, Carlos Sierra, Yiqi Luo

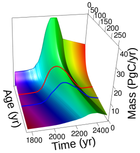
Why would **you** want a model data base?

- **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 . . .

$$\frac{d}{dt}\begin{bmatrix} leaf \\ wood \end{bmatrix} = \begin{bmatrix} I_{leaf}(t) \\ I_{wood} \end{bmatrix} + \begin{bmatrix} -k_{leaf2wood} - k_{leafo}(t) & k_{wood2leaf} \\ k_{leaf2wood} & -k_{wood2leaf} - k_{woodo} \end{bmatrix} \begin{bmatrix} leaf \\ wood \end{bmatrix}$$







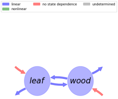
- Find and **reduce** sources of **uncertainty** in carbon predictions

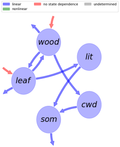
Markus Müller, Holger Metzler, Verónika Ceballos Núñez, Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou, Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang, Alison Bennett, Chenyu Bian, Lifen Jiang, Song Wang, Chengcheng Gang, Carlos Sierra, Yiqi Luo

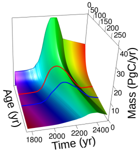
Why would **you** want a model data base?

- **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 . . .

$$\frac{d}{dt}\begin{bmatrix} leaf \\ wood \end{bmatrix} = \begin{bmatrix} I_{leaf}(t) \\ I_{wood} \end{bmatrix} + \begin{bmatrix} -k_{leaf2wood} - k_{leafo}(t) & k_{wood2leaf} \\ k_{leaf2wood} & -k_{wood2leaf} - k_{woodo} \end{bmatrix} \begin{bmatrix} leaf \\ wood \end{bmatrix}$$







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

Markus Müller, Holger Metzler, Verónika Ceballos Núñez, Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou, Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang, Alison Bennett, Chenyu Bian, Lifen 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

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

How can you build one?

We need:

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

    Arora2005GCB-1 ,CARDAMOM ,. . . ,

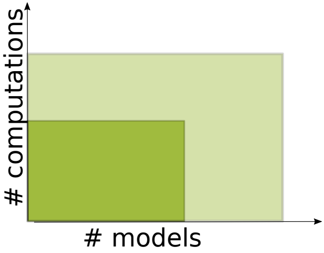
    Zelenev2000MicrobialEcology
  - of many computable **properties** and diagnostics to **compare** them by

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

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

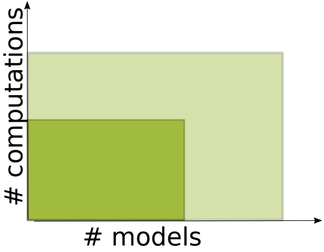


Markus Müller, Holger Metzler, Verónika Ceballos Núñez, Kostiantyn Viatkin, Thomas Lotze, Jon Wells, Yu Zhou, Cuijuan Liao, Aneesh Chandel, Feng Tao, Yuanyuan Huang, Alison Bennett, Chenyu Bian, Lifen Jiang, Song Wang, Chengcheng Gang, Carlos Sierra, Yiqi Luo

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



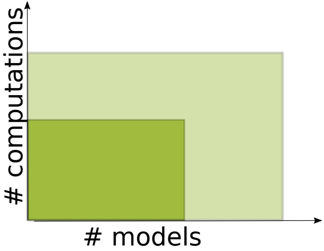
- ‘Copy and paste’?
  → bilinear increase of code
  → unmaintainable, untestable, errorprone, hard to change …

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

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 **r**epeat **y**ourself

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

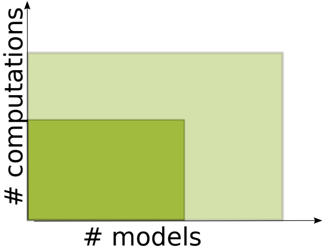
How can you build one?

We need:

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

    Arora2005GCB-1 ,CARDAMOM ,. . . ,

    Zelenev2000MicrobialEcology
  - of many computable **properties** and diagnostics to **compare** them by



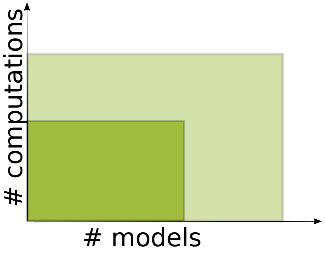
- 'Copy and paste'?
  → bilinear increase of code
  → unmaintainable, untestable, errorprone, hard to change ...

- better use the **dry** principle
  Don't **r**epeat **y**ourself

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

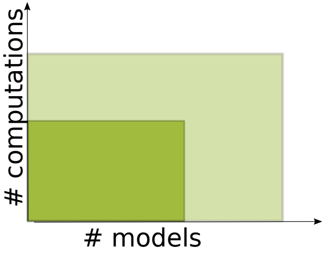


- ‘Copy and paste’?
  → bilinear increase of code
  → unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
  Don’t **r**epeat **y**ourself

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

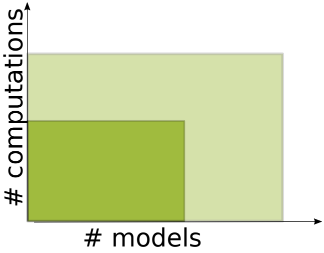


- ‘Copy and paste’?
  → bilinear increase of code
  → unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
  Don’t **r**epeat **y**ourself

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



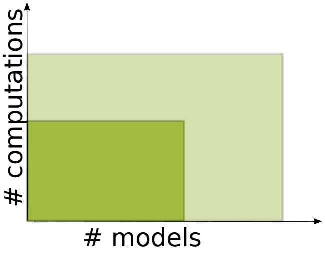
- ‘Copy and paste’?
  → bilinear increase of code
  → unmaintainable, untestable, errorprone, hard to change ...
- better use the **dry** principle
  Don’t **r**epeat **y**ourself

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

How can you build one?

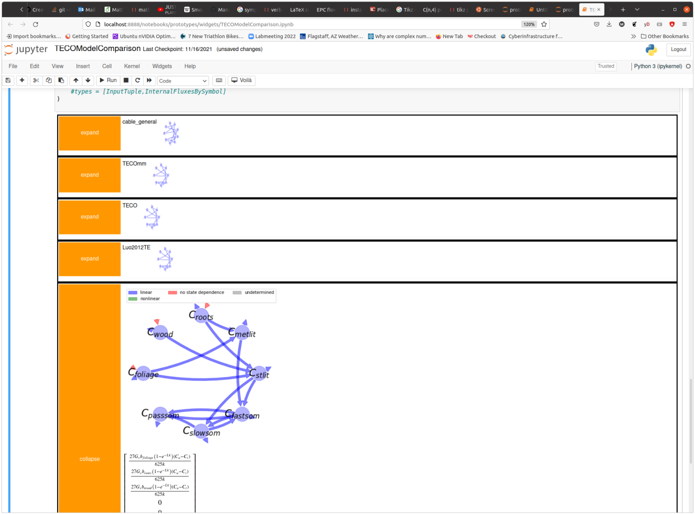
We need:

- **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
  - **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 **r**epeat **y**ourself

Example widget for query result

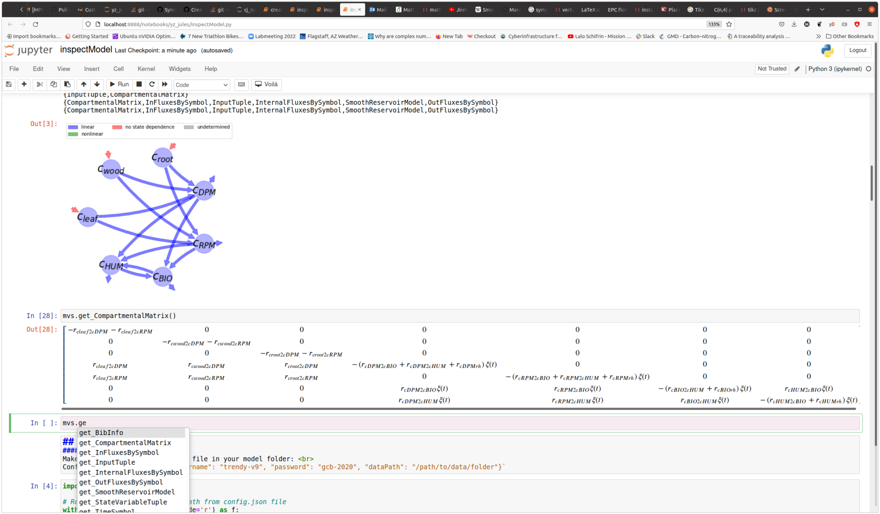


```
    #types = [InputTuple,InternalFluxesBySymbol]
)
```

$$\begin{bmatrix} \frac{27G_s b_{foliage}\left(1-e^{-Lk}\right)\left(C_a-C_i\right)}{625k} \\ \frac{27G_s b_{roots}\left(1-e^{-Lk}\right)\left(C_a-C_i\right)}{625k} \\ \frac{27G_s b_{wood}\left(1-e^{-Lk}\right)\left(C_a-C_i\right)}{625k} \\ 0 \\ 0 \end{bmatrix}$$

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

Analysis with symbolic tools (sympy) . . .



```
{InputTuple,CompartmentalMatrix}
{CompartmentalMatrix,InFluxesBySymbol,InputTuple,InternalFluxesBySymbol,SmoothReservoirModel,OutFluxesBySymbol}
{CompartmentalMatrix,InFluxesBySymbol,InputTuple,InternalFluxesBySymbol,SmoothReservoirModel,OutFluxesBySymbol}
```

Out[3]:

In [28]: `mvs.get_CompartmentalMatrix()`

Out[28]:

$$\left[\begin{array}{ccccccc}
-r_{cleaf2cDPM}-r_{cleaf2cRPM} & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -r_{cwood2cDPM}-r_{cwood2cRPM} & 0 & 0 & 0 & 0 & 0\\
0 & 0 & -r_{croot2cDPM}-r_{croot2cRPM} & 0 & 0 & 0 & 0\\
r_{cleaf2cDPM} & r_{cwood2cDPM} & r_{croot2cDPM} & -\left(r_{cDPM2cBIO}+r_{cDPM2cHUM}+r_{cDPMrh}\right)\xi(t) & 0 & 0 & 0\\
r_{cleaf2cRPM} & r_{cwood2cRPM} & r_{croot2cRPM} & 0 & -\left(r_{cRPM2cBIO}+r_{cRPM2cHUM}+r_{cRPMrh}\right)\xi(t) & 0 & 0\\
0 & 0 & 0 & r_{cDPM2cBIO}\xi(t) & r_{cRPM2cBIO}\xi(t) & -\left(r_{cBIO2cHUM}+r_{cBIOrh}\right)\xi(t) & r_{cHUM2cBIO}\xi(t)\\
0 & 0 & 0 & r_{cDPM2cHUM}\xi(t) & r_{cRPM2cHUM}\xi(t) & r_{cBIO2cHUM}\xi(t) & -\left(r_{cHUM2cBIO}+r_{cHUMrh}\right)\xi(t)
\end{array}\right.$$

In [ ]: `mvs.ge`

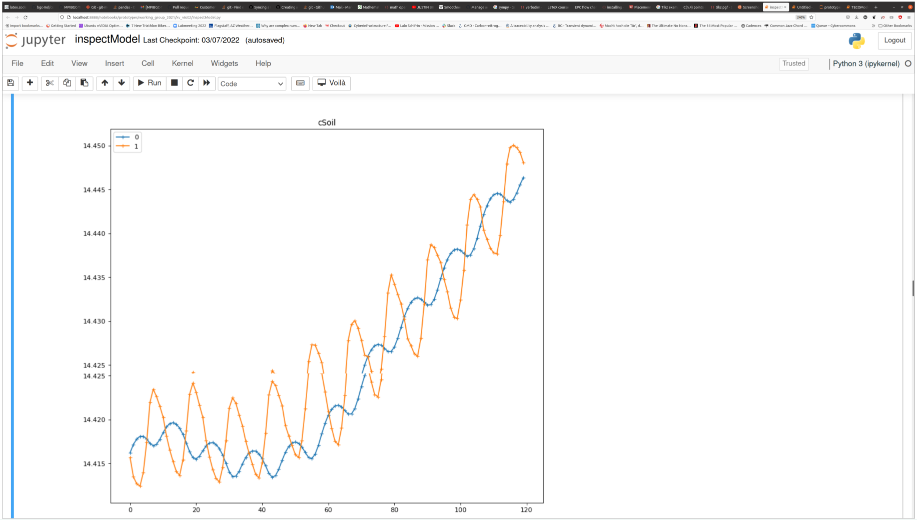
```
get_BibInfo
get_CompartmentalMatrix
get_InFluxesBySymbol
get_InputTuple
get_InternalFluxesBySymbol
get_OutFluxesBySymbol
get_SmoothReservoirModel
get_StateVariableTuple
get_TimeSymbol
```

```
##
####
Make                        file in your model folder: <br>
Conf                       rname": "trendy-v9", "password": "gcb-2020", "dataPath": "/path/to/data/folder"}`
```

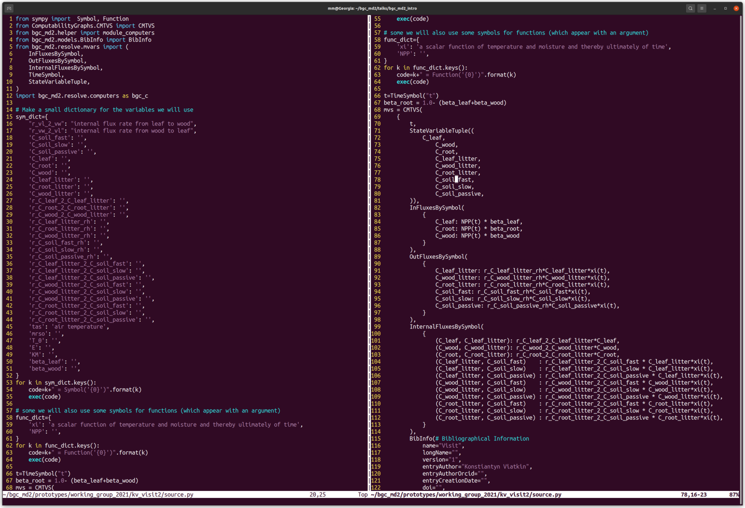
In [4]:

```
impo

# Re                        ath from config.json file
with                        de='r') as f:
```

. . . or numerically



Database records are python modules



```
from sympy import  Symbol, Function
from ComputabilityGraphs.CMTVS import CMTVS
from bgc_md2.helper import module_computers
from bgc_md2.models.BibInfo import BibInfo
from bgc_md2.resolve.mvars import (
    InFluxesBySymbol,
    OutFluxesBySymbol,
    InternalFluxesBySymbol,
    TimeSymbol,
    StateVariableTuple,
)
import bgc_md2.resolve.computers as bgc_c

# Make a small dictionary for the variables we will use
sym_dict={
    "r_vl_2_vw": "internal flux rate from leaf to wood",
    "r_vw_2_vl": "internal flux rate from wood to leaf",
    'C_soil_fast': '',
    'C_soil_slow': '',
    'C_soil_passive': '',
    'C_leaf': '',
    'C_root': '',
    'C_wood': '',
    'C_leaf_litter': '',
    'C_root_litter': '',
    'C_wood_litter': '',
    'r_C_leaf_2_C_leaf_litter': '',
    'r_C_root_2_C_root_litter': '',
    'r_C_wood_2_C_wood_litter': '',
    'r_C_leaf_litter_rh': '',
    'r_C_root_litter_rh': '',
    'r_C_wood_litter_rh': '',
    'r_C_soil_fast_rh': '',
    'r_C_soil_slow_rh': '',
    'r_C_soil_passive_rh': '',
    'r_C_leaf_litter_2_C_soil_fast': '',
    'r_C_leaf_litter_2_C_soil_slow': '',
    'r_C_leaf_litter_2_C_soil_passive': '',
    'r_C_wood_litter_2_C_soil_fast': '',
    'r_C_wood_litter_2_C_soil_slow': '',
    'r_C_wood_litter_2_C_soil_passive': '',
    'r_C_root_litter_2_C_soil_fast': '',
    'r_C_root_litter_2_C_soil_slow': '',
    'r_C_root_litter_2_C_soil_passive': '',
    'tas': 'air temperature',
    'mrso': '',
    'T_0': '',
    'E': '',
    'KM': '',
    'beta_leaf': '',
    'beta_wood': '',
}
for k in sym_dict.keys():
    code=k+" = Symbol('{0}')".format(k)
    exec(code)

# some we will also use some symbols for functions (which appear with an argument)
func_dict={
    'xi': 'a scalar function of temperature and moisture and thereby ultimately of time',
    'NPP': '',
}
for k in func_dict.keys():
    code=k+" = Function('{0}')".format(k)
    exec(code)

t=TimeSymbol("t")
beta_root = 1.0- (beta_leaf+beta_wood)
mvs = CMTVS(
```

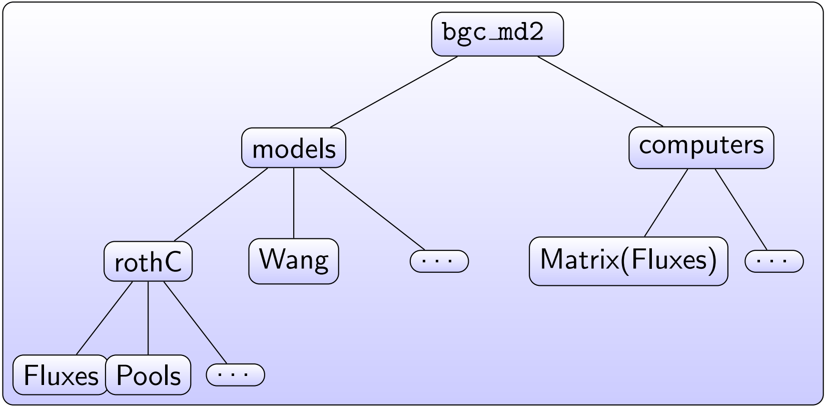
```
~/bgc_md2/prototypes/working_group_2021/kv_visit2/source.py        20,25         Top
```

```
    exec(code)

# some we will also use some symbols for functions (which appear with an argument)
func_dict={
    'xi': 'a scalar function of temperature and moisture and thereby ultimately of time',
    'NPP': '',
}
for k in func_dict.keys():
    code=k+" = Function('{0}')".format(k)
    exec(code)

t=TimeSymbol("t")
beta_root = 1.0- (beta_leaf+beta_wood)
mvs = CMTVS(
```

```
    {
        t,
        StateVariableTuple((
            C_leaf,
                C_wood,
                C_root,
                C_leaf_litter,
                C_wood_litter,
                C_root_litter,
                C_soil_fast,
                C_soil_slow,
                C_soil_passive,
        )),
        InFluxesBySymbol(
            {
                C_leaf: NPP(t) * beta_leaf,
                C_root: NPP(t) * beta_root,
                C_wood: NPP(t) * beta_wood
            }
        ),
        OutFluxesBySymbol(
            {
                C_leaf_litter: r_C_leaf_litter_rh*C_leaf_litter*xi(t),
                C_wood_litter: r_C_wood_litter_rh*C_wood_litter*xi(t),
                C_root_litter: r_C_root_litter_rh*C_root_litter*xi(t),
                C_soil_fast: r_C_soil_fast_rh*C_soil_fast*xi(t),
                C_soil_slow: r_C_soil_slow_rh*C_soil_slow*xi(t),
                C_soil_passive: r_C_soil_passive_rh*C_soil_passive*xi(t),
            }
        ),
        InternalFluxesBySymbol(
            {
                (C_leaf, C_leaf_litter): r_C_leaf_2_C_leaf_litter*C_leaf,
                (C_wood, C_wood_litter): r_C_wood_2_C_wood_litter*C_wood,
                (C_root, C_root_litter): r_C_root_2_C_root_litter*C_root,
                (C_leaf_litter, C_soil_fast)    : r_C_leaf_litter_2_C_soil_fast * C_leaf_litter*xi(t),
                (C_leaf_litter, C_soil_slow)    : r_C_leaf_litter_2_C_soil_slow * C_leaf_litter*xi(t),
                (C_leaf_litter, C_soil_passive) : r_C_leaf_litter_2_C_soil_passive * C_leaf_litter*xi(t),
                (C_wood_litter, C_soil_fast)    : r_C_wood_litter_2_C_soil_fast * C_wood_litter*xi(t),
                (C_wood_litter, C_soil_slow)    : r_C_wood_litter_2_C_soil_slow * C_wood_litter*xi(t),
                (C_wood_litter, C_soil_passive) : r_C_wood_litter_2_C_soil_passive * C_wood_litter*xi(t),
                (C_root_litter, C_soil_fast)    : r_C_root_litter_2_C_soil_fast * C_root_litter*xi(t),
                (C_root_litter, C_soil_slow)    : r_C_root_litter_2_C_soil_slow * C_root_litter*xi(t),
                (C_root_litter, C_soil_passive) : r_C_root_litter_2_C_soil_passive * C_root_litter*xi(t),
            }
        ),
        BibInfo(# Bibliographical Information
            name="Visit",
            longName="",
            version="1",
            entryAuthor="Konstiantyn Viatkin",
            entryAuthorOrcid="",
            entryCreationDate="",
            doi="",
```

```
~/bgc_md2/prototypes/working_group_2021/kv_visit2/source.py        78,16-23        87%
```

Internal Structure of bgc_md2

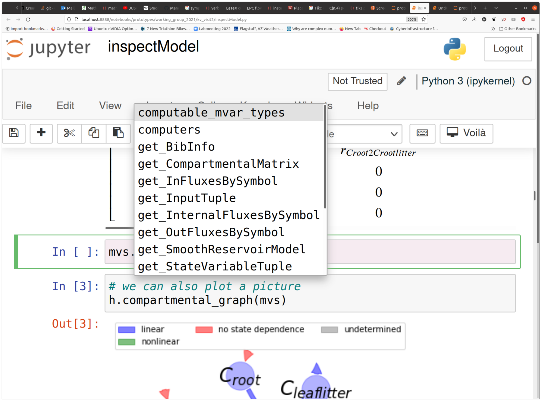


The bgc_md library provides I:

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.

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

Userinterface using computability graphs



```
computable_mvar_types
computers
get_BibInfo
get_CompartmentalMatrix
get_InFluxesBySymbol
get_InputTuple
get_InternalFluxesBySymbol
get_OutFluxesBySymbol
get_SmoothReservoirModel
get_StateVariableTuple
```

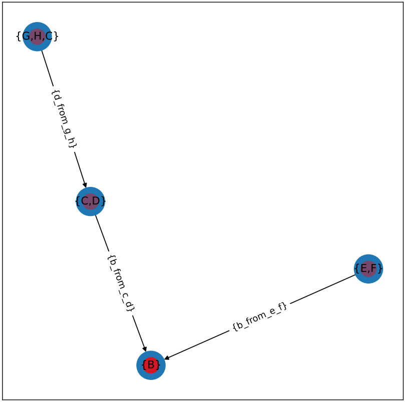
```
In [ ]: mvs.
```

```
In [3]: # we can also plot a picture
        h.compartmental_graph(mvs)
Out[3]:
```

Suggested methods automatically created by `ComputabilityGraphs`

Finding what's missing in the model description

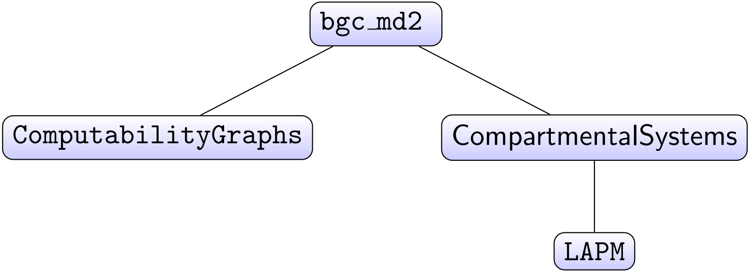
given a set of functions: a(i), b(c,d), b(e,f), c(b), d(b), d(g,h), e(b), f(b) and the target variable B e.g. `CompartmentalMatrix`, The algorithm computes all possible combinations and paths from which B can be computed.



The `bgc_md` library provides II:

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



bgc_md2

ComputabilityGraphs

CompartmentalSystems

LAPM

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

Example computation via CompartmentalSystems

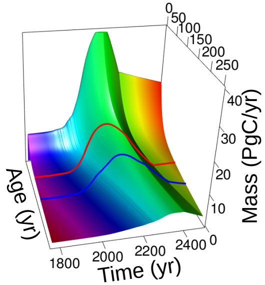


Figure: age distribution of a pool as function of time

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`
- `https://mybinder.org/v2/gh/MPIBGC-TEE/bgc_md2/binder` binder for testing some tutorials (jupyter notebooks) for the creation of new models or a model comparison No installation necessary.