

-
- 1. Opus Datasets and Creating New Models**
 - 2. Interactive Model Exploration**
 - 3. Nested Logit Model in UrbanSim**

Hana Ševčíková

Puget Sound Regional Council
University of Washington, Seattle

hanas@uw.edu

Opus Dataset

Conceptually a $n \times m$ table with n records and m characteristics.

Example:

parcel_id	land_value	x	y
1	500	1	1
2	350	1	2
3	400	1	3
4	430	1	4
5	510	1	5
6	550	2	1
7	480	2	2
8	480	2	3
9	500	2	4
10	530	2	5
11	570	3	1
12	500	3	2
13	560	3	3
14	600	3	4
15	620	3	5
16	580	4	1
17	600	4	2
18	690	4	3
19	750	4	4
20	800	4	5

Dataset Implementation

Dataset

```
attribute_boxes = {'parcel_id': AttributeBox([1,2,3,...]),
                  'land_value': AttributeBox([500, 350,...]),
                  'x': AttributeBox([1,1,1,...]),
                  'y': AttributeBox([1,2,3,...])}
in_storage = Storage(...) [flt, tab, csv, sql, dbf, esri, dict]
out_storage = Storage(...)
dataset_name = 'parcel'
id_name = ['parcel_id']
```

AttributeBox

```
_data = numpy array
_version
_type
```

Demo, see tutorial

Opus Model

Requirements:

- Child of the class Model

Opus Model

Requirements:

- Child of the class Model

Include in a model system:

- Contains a method `run`
- Optionally contains a method `prepare_for_run`

Opus Model

Requirements:

- Child of the class `Model`

Include in a model system:

- Contains a method `run`
- Optionally contains a method `prepare_for_run`

For estimation:

- Contains a method `estimate`
- Optionally contains a method `prepare_for_estimate`

Hello World! Model

```
from opus_core.models.model import Model
class MyModel(Model):
    def run(self):
        print "\n\nHello World, Im an Opus model!"
```

Hello World! Model

```
from opus_core.models.model import Model
from opus_core.logger import logger
class MyModel(Model):
    model_name = "My Opus Model"
    def run(self):
        logger.log_status("Hello World, Im an Opus model!")
```

Hello World! Model

```
from opus_core.models.model import Model
from opus_core.logger import logger
class MyModel(Model):
    model_name = "My Opus Model"
    def run(self):
        logger.log_status("Hello World, Im an Opus model!")
```

Save into 'mymodel.py'. Then

```
>>> from mymodel import MyModel
>>> MyModel().run()
```

Hello World! Model

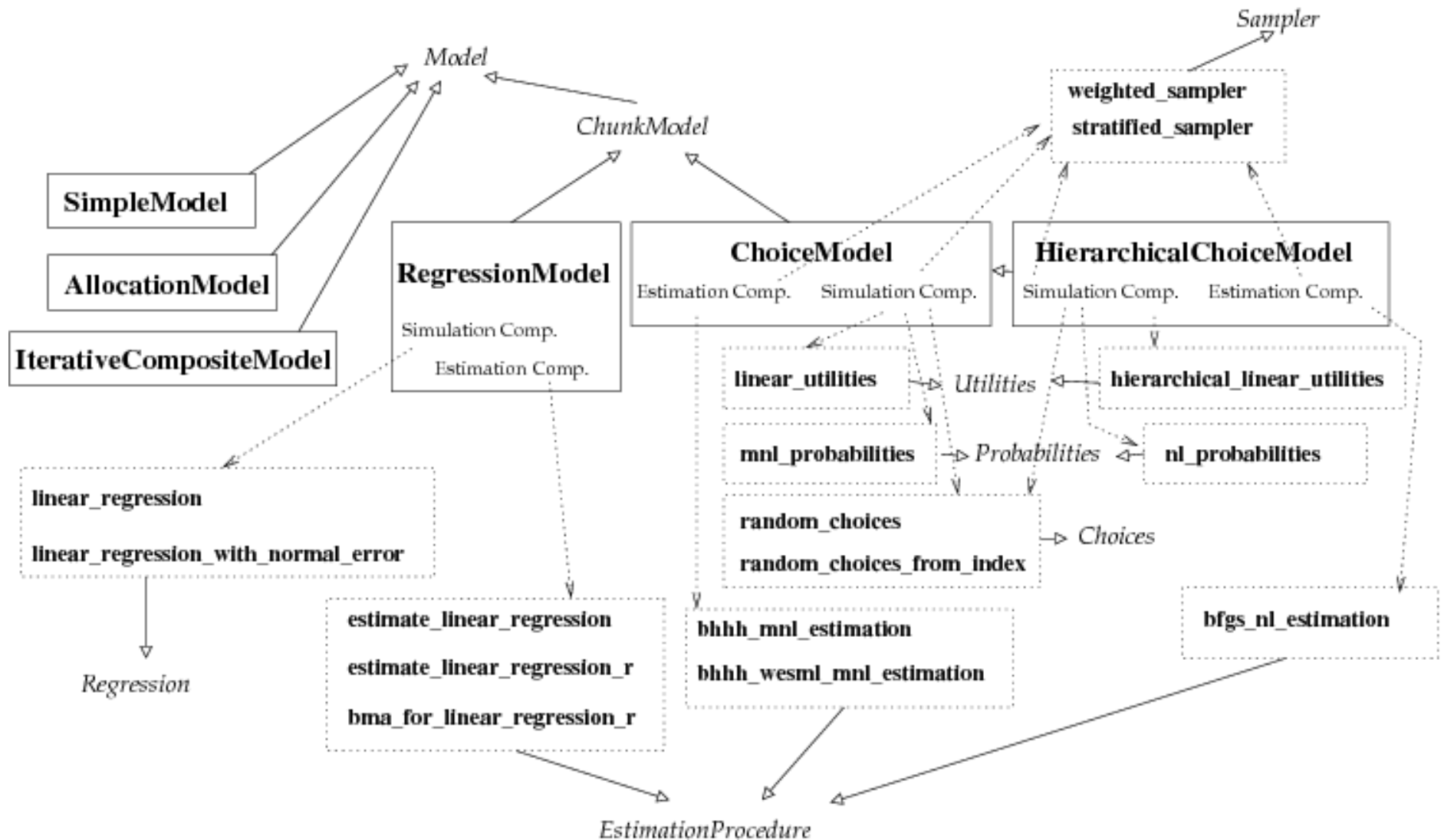
```
from opus_core.models.model import Model
from opus_core.logger import logger
class MyModel(Model):
    model_name = "My Opus Model"
    def run(self):
        logger.log_status("Hello World, Im an Opus model!")
```

Save into 'mymodel.py'. Then

```
>>> from mymodel import MyModel
>>> MyModel().run()
```

```
Running My Opus Model (from __main__):  started on July 5 10:07
    Hello World, Im an Opus model!
Running My Opus Model (from __main__):  completed.....0.0 sec
```

Opus Core Models and Components



Options for Writing Opus Models

1. Adapt configuration of an existing model.

Options for Writing Opus Models

1. Adapt configuration of an existing model.
2. Implement a child class of an existing model and overwrite a part of it, e.g. `prepare_for_run`.

Options for Writing Opus Models

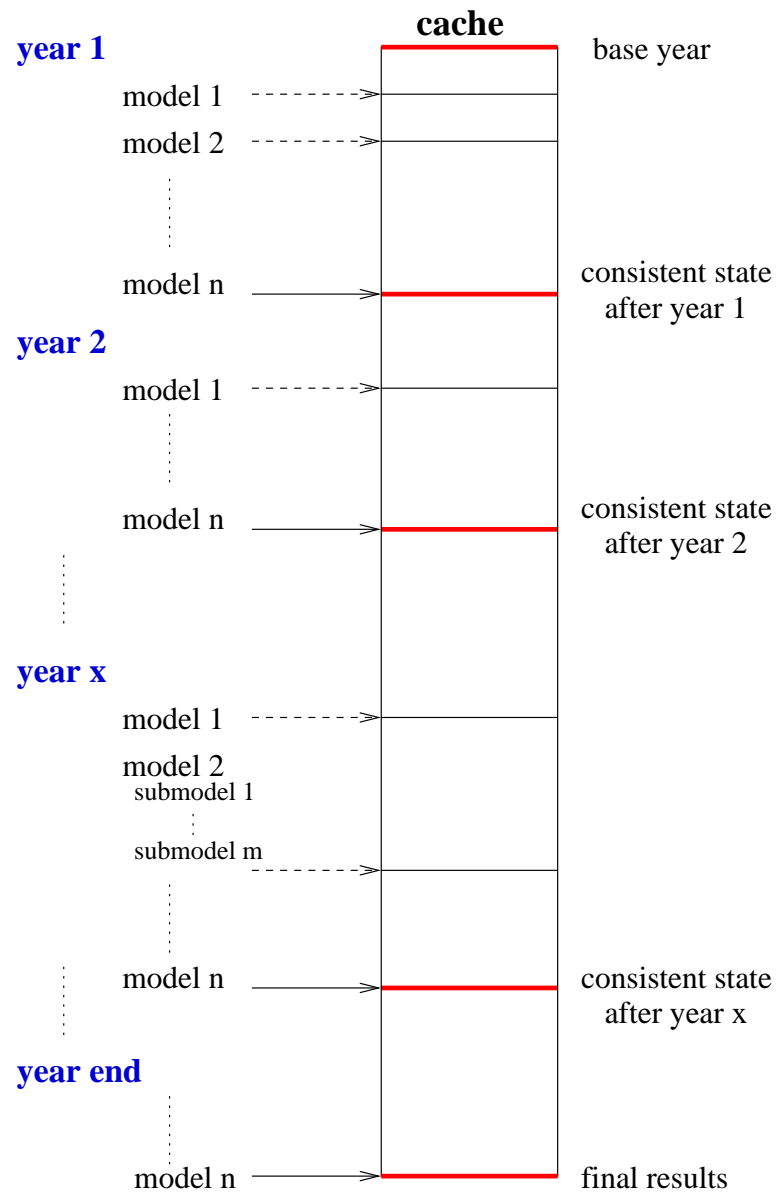
1. Adapt configuration of an existing model.
2. Implement a child class of an existing model and overwrite a part of it, e.g. `prepare_for_run`.
3. For regression and choice models: Implement your own model component, e.g. for computing utilities.

Options for Writing Opus Models

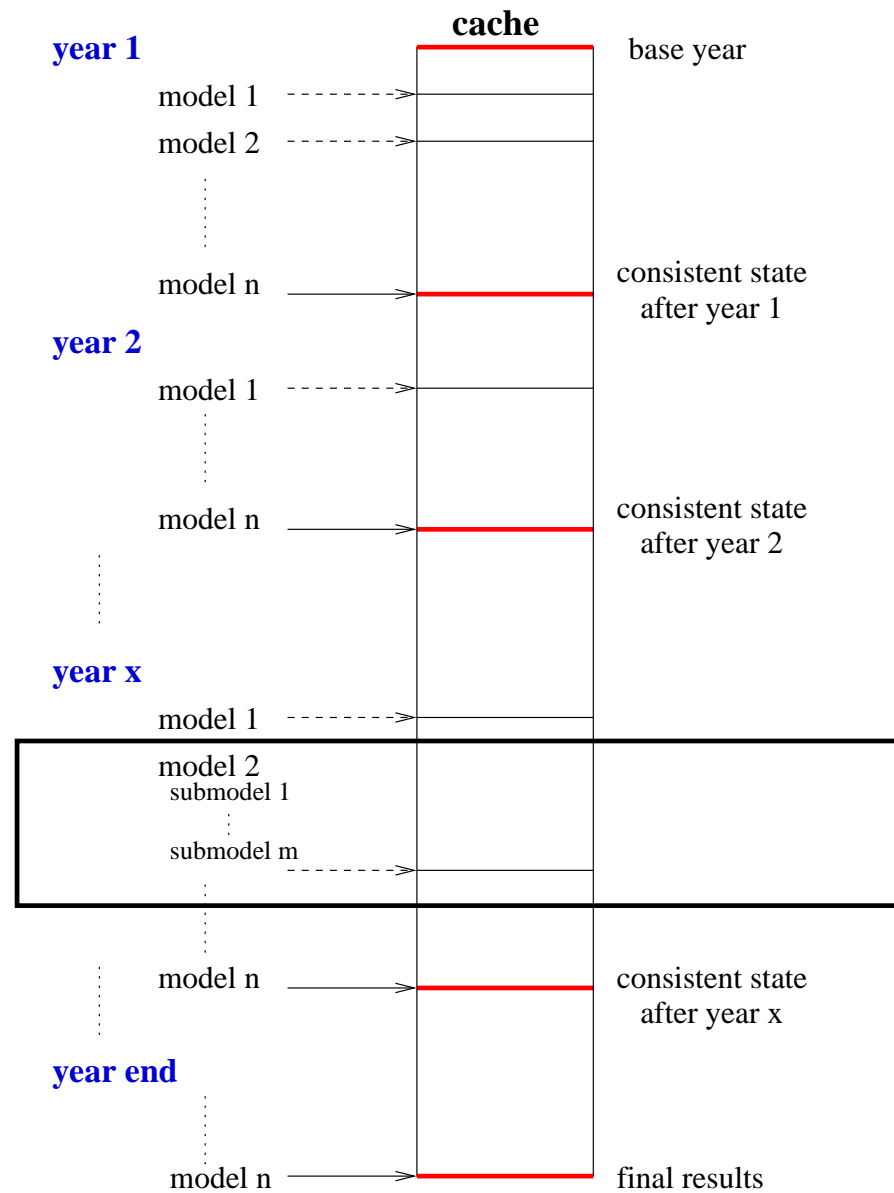
1. Adapt configuration of an existing model.
2. Implement a child class of an existing model and overwrite a part of it, e.g. `prepare_for_run`.
3. For regression and choice models: Implement your own model component, e.g. for computing utilities.
4. Implement your own model.

2. Interactive Model Exploration

Simulation Run and Caching



Simulation Run and Caching



Input:

- cache location
- year (of a consistent state of cache)
- model
- model dependencies (models to run prior the given model)
- xml configuration
- scenario

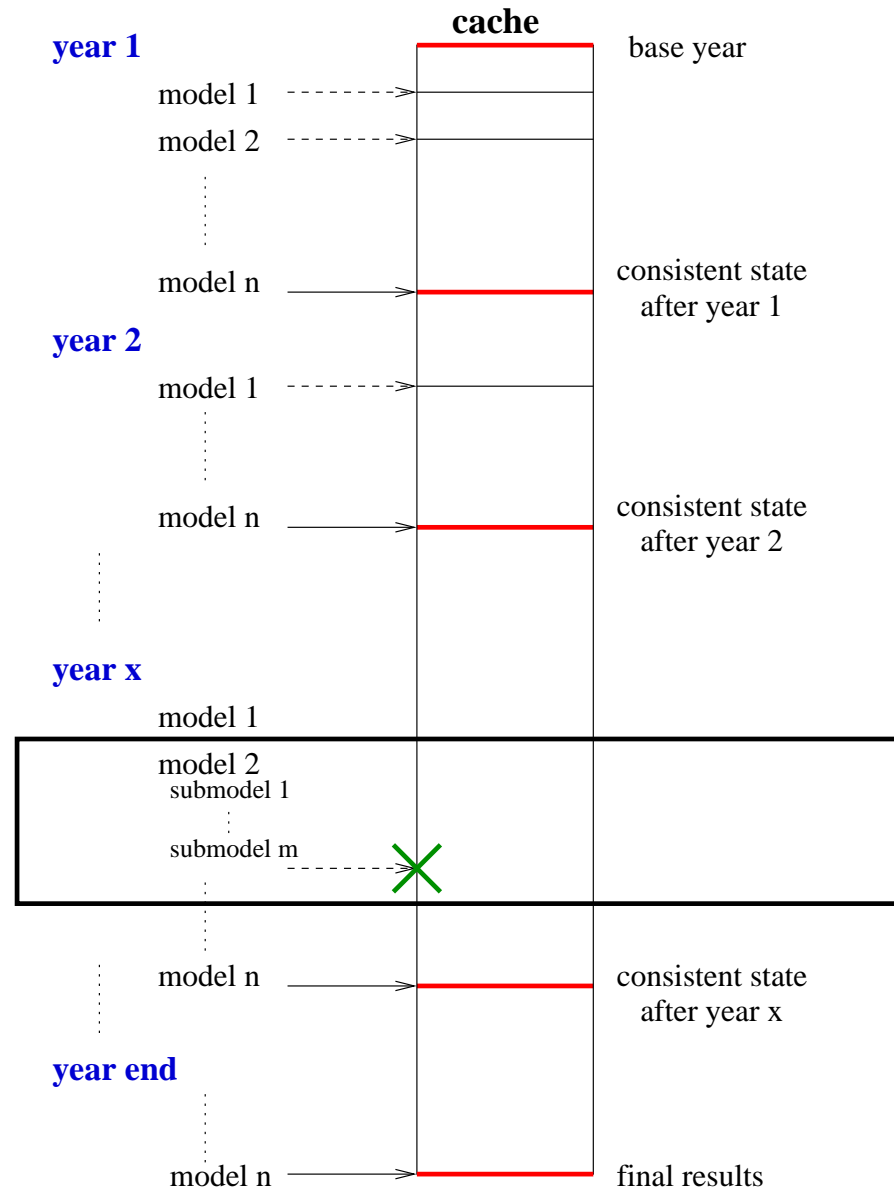
Model Explorer

Input:

- cache location
- year (of a consistent state of cache)
- model
- model dependencies (models to run prior the given model)
- xml configuration
- scenario

Run [ModelExplorer](#) \Rightarrow stops in python shell after simulating the given model

Model Explorer



Model Explorer

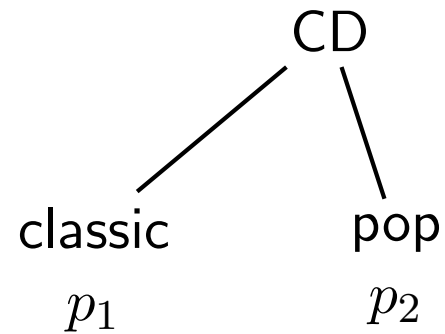
Features:

- analyze data entering the simulation procedure:
 - summary
 - plot histograms
 - plot variable correlation
 - access data directly
- output all dependent variables of the model (leaves of a depend. tree)
- access any dataset and its attributes
- get differences in a given attribute between before and after running a model
- access any model object

3. Nested Logit Model in UrbanSim

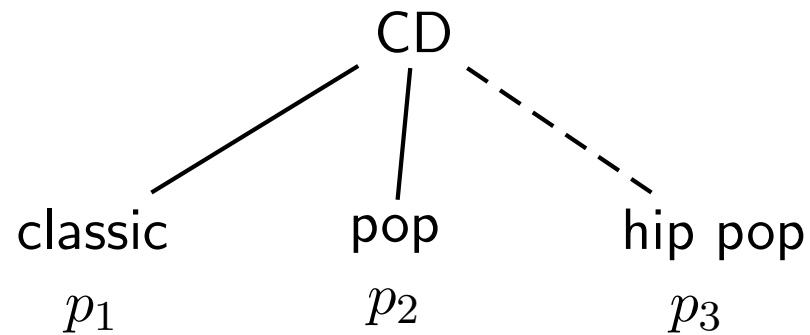
Motivation

- Multinomial logit model - independence of irrelevant alternatives (IIA)
- Biased estimates if IIA is violated
- Example:



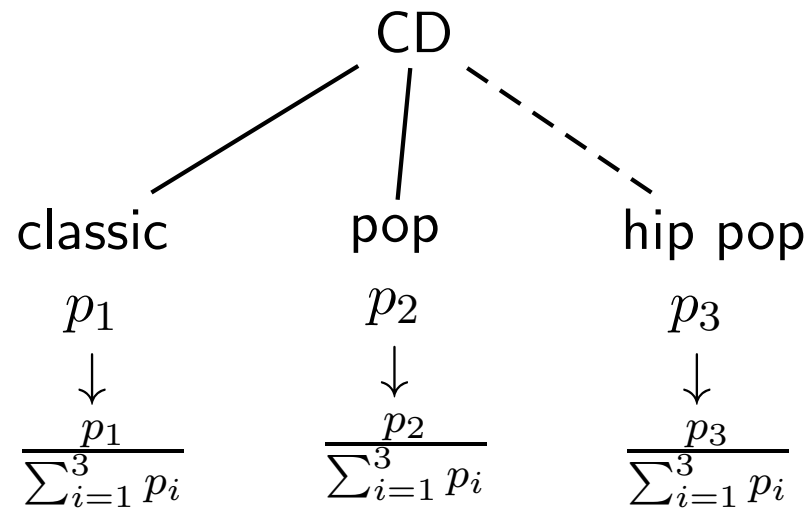
Motivation

- Multinomial logit model - independence of irrelevant alternatives (IIA)
- Biased estimates if IIA is violated
- Example:



Motivation

- Multinomial logit model - independence of irrelevant alternatives (IIA)
- Biased estimates if IIA is violated
- Example:



Nested Logit Model

- Allows interdependence between pairs of alternatives.
- IIA holds within nests, but not between nests.
- UrbanSim implements McFadden's (1978) nested logit:

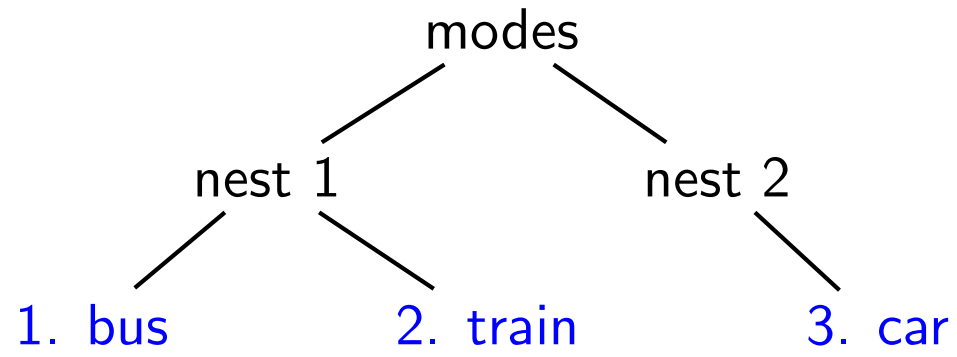
$$P_i = P_{i/m} \times P_m \quad \text{for alternative } i \text{ and nest } m$$

$$P_{i/m} = \frac{e^{V_i/\mu_m}}{\sum_{j \in N_m} e^{V_j/\mu_m}}$$

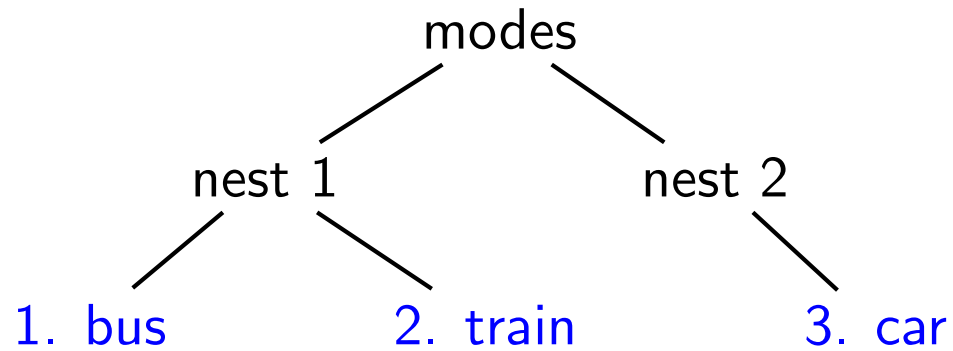
$$P_m = \frac{e^{\mu_m I_m}}{\sum_{n=1}^M e^{\mu_n I_n}}, \quad I_m = \ln \sum_{j \in N_m} e^{V_j/\mu_m}$$

μ_m is the logsum or inclusive value coefficient.

Example



Example



mode_id	mtype	cost_1	cost_2	cost_3	time_1	time_2	time_3
88	1	284.31	230.0	196.46	37.08	30.0	25.63
89	2	284.31	230.0	196.46	37.08	30.0	25.63
91	1	288.37	296.85	424.07	34.0	35.0	50.0
98	3	288.37	296.85	424.07	34.0	35.0	50.0
100	2	288.37	296.85	424.07	34.0	35.0	50.0
101	3	289.54	216.99	278.46	38.13	28.57	36.67
102	1	289.54	216.99	278.46	38.13	28.57	36.67
...					...		

Example - Notes on configuration

- Use expressions of type:

```
time = mode_x_choice.agent_times_choice(time)
```

- Use modules:

```
utilities='opus_core.upc.hierarchical_linear_utilities' (init)
```

```
probabilities='opus_core.upc.nl_probabilities' (init)
```

```
procedure='opus_core.bfgs_nl_estimation' (estimate)
```

- The likelihood function is not global concave, therefore starting values can be given.
 - The μ coefficient should be included for estimation using the keyword `--logsum_x`, with x being nest id.
 - `--logsum_x` can be set to a constant (not estimated).
 - If a nest has only one alternative, `--logsum_x` should not be included or should be set to 1.
-

Example - Specification

nest

$$1 \quad V_1 = \beta_1 \cdot \text{cost}_1 + \beta_2 \cdot \text{time}_1$$

$$1 \quad V_2 = \alpha_2 + \beta_1 \cdot \text{cost}_2 + \beta_2 \cdot \text{time}_2$$

$$2 \quad V_3 = \alpha_3 + \beta_1 \cdot \text{cost}_3 + \beta_2 \cdot \text{time}_3$$

$\mu_2 = 1$, estimate $\mu_1, \alpha_1, \alpha_2, \beta_1, \beta_2$