|  |  |  |
| --- | --- | --- |
|  | PTV VISUM example description | |
|  | | |
| AN ACTIVITY-BASED MODEL | |  |
|  | |  |
|  | |  |

# Brief description

The example provides a basis for building an activity-based model in PTV Visum. The model structure is explained, as well as the model use.

# Requirements

Visum modules: ABM

# Objective

Contact: ptvvision@ptv.de

Modeling activity-based models in Visum is different from modeling other types of demand models. Visum does not provide exactly predefined procedures with easy-to-use dialogs for the choice models. Instead, the procedure run is done using Python scripts, and the model parameters are kept in user-defined tables.

While comfort may suffer to some extent, this opens up the possibility of implementing one's own ideas in a very individual way. This is particularly important because, at present, there is still no uniformly recognized model structure for ABM.

The present example represents a complete ABM. It is not necessary to adjust the model structure. The example can be applied to your own situation without programming skills. In particular, the Python scripts do not need to be understood or adapted.

In this respect, this example is much more than an ordinary example: it not only explains how to use the functionalities, but it serves as a template for an own model. Furthermore, the model structure and all submodels are discussed in detail in this description.

The model is an extension of the model published with Visum 2020. It was the result of a cooperation between PTV AG and the Swiss Federal Railways (SBB).

This document covers the areas:

* How to calculate the example
* How to build your own model
* Description of the model structure and submodels
* Explanation of the program code

# Model preparation

An activity-based demand model models the mobility behavior for each person of the population individually. For this purpose, a synthetic population was created for the present example, whose characteristics are similar to those of the actual population. This population data as well as suitable structural data is included in the example and will be read in the procedure sequence. The model already contains all private and public transport network data. If you want to set up your own ABM, all data must first be generated and then imported into Visum. For information on this topic, see the section "How to build your own model".

Depending on the preferred level of spatial resolution, population and structural data can be modeled at zone level or any higher resolution. In this example, the data has a very high resolution and is represented by locations.

# Model overview

An ABM generates a complete schedule for each person in the population with all activities, their start and end times, associated trips, and their modes. The model calculation is divided into several parts.

First, all necessary skims are calculated, which are later used in choice models. These calculations are defined in the procedure sequence. They do not differ significantly from those found in conventional macroscopic demand models. In addition to the typical macroscopic parameters (i.e., matrices on zone level), this is also data that serves as the basis for shortest path searches later in the process. This includes skim matrices on the level of stop areas as well as private transport assignments including the storage of assignment results.

Then, several scripts are executed to simulate the mobility decisions of all persons individually:

* Long-term choices on workplace and/or training places/school places
* Generation of trip chains
  + Number of tours, sub-tours, and stopovers
  + Choice of activities, their start times, and durations
* Destination and mode choice

At the end, the trips are aggregated into demand matrices and assigned to the network.

The present example model can be calculated in two variants: the generation of the trip chains can be model-based or data-based. In the model-based variant, the formation of individual trip chains is simulated based on choice models. In the data-based variant, the trip chains are taken from a household survey. The two variants are commented on in more detail below.

After the simulation of each choice, the results are stored in the data model. For example, if the simulation of the number-of-work-tours-model for a certain person results in two work-tours, two work-tour objects are generated in Visum and linked to the respective person. By storing the intermediate results in this way, it is possible to calculate the choice models one after the other and to validate the intermediate results.

The present example is disaggregated in many ways:

* Residential and activity locations are represented by Visum’s activity "location" object with x and y coordinates.
* In a first step, the destination choice is based on conventional traffic zones. The second step is to distribute the trips among the activity locations within the zone, with the distribution proportional to the attraction potential.
* Mode choice is performed for the specific start and end locations. It is based on the shortest trips between the (active) nodes closest to the locations.
* The model is divided into eight time periods (analysis time intervals) from 'early morning' to 'night'. Skims are calculated per time range and are included in the destination and mode choice.
* All choices are simulated individually for each person. In doing so, characteristics of persons can be directly incorporated into the choice models. The choice models are typically defined for certain sets of persons. These person set definitions do not have to remain constant over the entire model calculation but can differ from submodel to submodel. For example, there is a choice model for the whole population on the choice of the tour start time. Mode choice, in contrast, distinguishes between students and adults in two different choice models.

# Model setup and calculation

The following describes how the calculation can be carried out and reproduced for the example model. It is also discussed how you can visualize the input and result data. The exact model specification and its coding can be found below.

1. Set all project directories to the example folder.
2. Open the version file Halle\_ABM.ver.

The model already contains the transport networks, but no demand data yet. Load them to the model first:

1. Open 1\_UDA.net to load the user-defined attributes.
2. Use 2\_Model\_specifications.net to load the model specifications, i.e. utility functions etc.
3. Open 3\_Base\_Demand\_Model.dmd to load the ABM demand model and its activities, and 4\_Base\_Demand\_Model\_Data.att to load all other necessary data.
4. Open 5\_Structural\_Data.dmd to load the structural data, i.e. locations and activity locations.

The next step is to read in the synthetic population using a script. For this, the script expects three data files:

* HHI.dmd with the original data of households and persons as well as their trip chains from a household survey
* SynPopHH.csv and SynPopPerson.csv with the synthetic households as well as persons generated from the household survey

If the data is created for your own model, you should keep the structure of the data files so that the same import script can be used. A description of the files can be found below in the section "How to build your own model”.

1. Start the two procedures in the procedure sequence from the Preparation group. You can ignore the warning that is possibly generated.

The model now contains all input data.

1. You can now look at the model specifications, population and structural data in lists and also in the network editor. For this, open the global layout file Data.lay.

In the **Persons** list, all persons of the model are listed. Their individual properties, such as age and employment category, are stored in user-defined attributes.

In the **Activity locations** list, all pairs of "Activity x Location" are listed. Activity locations are locations at which the associated activity can be carried out. The attraction potential concerning the activity, for example, the number of primary school places in the case of the activity "Education Primary Pupil", is saved in the attribute "AttractionPotential".

If you select one or more data records in the lists, these are highlighted in the network editor.

1. Switch to the user-defined table **Model Steps**.

The **Model Steps** table lists all model steps. It starts with the long-term choices (column "ChoiceModel" = "PrimLoc") and ends with the destination and mode choice. The "Specification" column contains the name of the table that contains the model parameters for each model step.

The name "PrimLocStu" is stored there for the first model step.

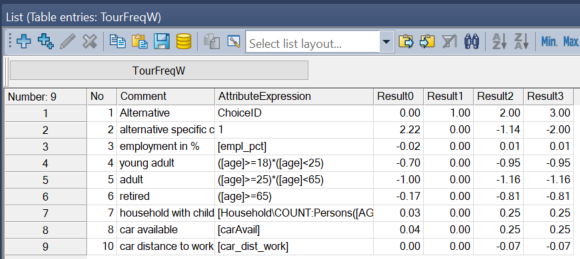
1. Switch to the PrimLocStu table.

In this table, you can find the definition of the utility uod for the destination choice of the school location d from the residency o. For secondary school students, it is defined by:

uod = 0.8 • Matrix(LogSumNoCarAv, AM)od

The Matrix(LogSumNoCarAv, AM) was already calculated during the skim calculation. It represents the utility (= negative impedance) of a change of location in the form of a LogSum across all transport modes. The impedances of the transport modes are calculated for the morning peak (AM).

1. Switch to the TourFreqW table.



This list contains the parameters of the choice model for the number of work tours. The "Attribute Expression” column lists the person attributes (xi) that contribute to the utility of each choice alternative. The other columns contain the corresponding parameters (βA,i), which belong to the respective alternative (A) and the person attribute xi. The alternative itself (i.e. the numbers 0, 1, 2, 3) is defined in the row with Attribute Expression = ChoiceID (see the first row in the list).

The utility uA of an alternative A is calculated as follows:

uA = Σi βA,i • xi (Σi here means sum over i)

More specifically, for example for the alternative "0 work tours”, this means

u0 = 2.22

- 0.02 • [empl\_pct]

- 0.70 • [age\_18\_u25]

- 1.00 • [age\_25\_u65]

- 0.17 • [age\_65plus]

+ 0.03 • [has\_child]

+ 0.04 • [carAvail]

The parameters βA,i show what influence the attributes have on the utility of the choice alternatives. Young age (18 - 25 years), for example, reduces the utility of the "0 work trips" alternative by 0.7. In contrast, the presence of children increases the utility of the alternative "2 work tours" by 0.25.

The parameters for the individual attributes are usually estimated based on a mobility survey. In this model, the parameters were set arbitrarily for demonstration purposes.

In this case, the attributes xi are always user-defined attributes of persons. However, it is also possible to use derived attributes. The notation for such derived attributes is demonstrated, for example, in row 7. In addition, the attribute expressions can also contain certain functions (see, for example, in rows 4 and 5). The syntax of the expressions is identical to the regular formulas in Visum.

The choice model described here is a so-called utility-based logit model, a typical model for activity-based models (referenced below as the "Standard choice model"). To the choice probabilities pA of alternative A, the following applies:

pA = exp (uA ) / Σi exp (uA,i ).

The example version already contains the tours, trips, and activity executions for all persons from the survey data read in the previous step. Just like in the usual 4-step models, only the destination and mode choices are missing. You can view the data in the corresponding lists.

The data was read in for the so-called data-based ABM variant. Here, the creation of the trip chains is not simulated by choice models, but copied directly from the results of a survey. In case the model-based ABM variant is to be calculated, activate the script procedure run\_init\_schedules.py in the ABM group and delete all result data.

In the following groups, skim matrices are calculated, which are required for the destination and mode choice models. For this purpose, the respective demand segments are calculated in advance. For the Walk and Bike demand segments, the demand does not play a role, as they are calculated according to the best path. However, all connectors are to be used according to their weight, which is why an assignment of a dummy demand is necessary.

1. Start all procedures of the initialize demand matrices group so that the demand matrices are initialized.
2. Start the procedures of the groups PT Skims and PrT Skims without Car.

For the calculation of the car skims, a static assignment is calculated for each analysis time period. For this purpose, an assignment is calculated in the **Car Skims** group for each time interval and, based on this, skims as well as LogSum matrices for the choice of long-term choices. The demand per time interval is initially always identical. Only when the entire demand loop is run several times and the passenger car demand is updated in the process, do the different demands of the different time intervals affect the assignment results. At the end of the group, the impedance and travel time are saved at all path objects for later shortest path searches.

1. Start the procedures of the Car Skims group.

Now all data and parameters are available, and the calculation of the ABM can be started.

1. Start the first procedure of the ABM group.

This procedure checks if the model specifications contain typos.

1. Start the second procedure of the ABM group.

The opened filter activates all nodes that are suitable for connecting the locations to the network. Locations are not explicitly connected to the network, but always via the nearest active node. Therefore, care must be taken to ensure that all nodes that are not suitable for connecting are disabled by the filter.

1. In case the model-based ABM variant is to be calculated, activate the third procedure run\_init\_schedules.py to delete all result data.

In the next procedure, the long-term choices are simulated, i.e. the destinations of mandatory activities such as work or school.

1. Start the next procedure of the ABM group.

If the activity chains are to be generated model-based, start also the next procedure (run\_tripchain\_choice.py).

1. In the last step of the ABM, you start the destination and mode choice (script run\_destination\_and\_mode\_choice.py).
2. Now the demand from the trips is saved in demand matrices so that they can be used in an assignment. To do this, start the next procedure (script run\_aggregate\_trip\_volumes.py), which aggregates the demand from the trips per demand segment and analysis time interval and writes the result to the corresponding demand matrices (these must have the name "assignment" in addition to the corresponding demand segment and the attributes From-Time as well as To-Time).

If the demand is to be calculated with a demand loop so that the car travel times update each time, activate and start the last procedure of the ABM group (Go to the procedure).

After the demand calculation is completed, the results can be analyzed (see the following section). In order to be able to examine the results in the context of assignments (e.g. flow bundle calculations), the demand will be assigned for the morning peak as an example. In addition, the tours are converted into path sequences.

1. Start the first two procedures from the Assignment group.

Subsequently, the demand segments can be calculated in assignments.

1. To do this, start the two assignments from the Assignment group.

## Results

In the following, the results of the data-based ABM variant are presented after a single run, i.e., without activating the “Go to” procedure.

Switch alternately to the Tours and Path sequences lists.

The Tours list shows all tours created during the model calculation. If you select any of them, they are displayed in the network editor.   
The Path sequences list shows the associated path sequences, for which the corresponding assignment paths are also displayed in the network editor.

Further results can be found in the lists:

* Trips
* Activity executions

Halle\_ABM\_Result.ver is a version file with a complete calculation of the ABM.

# How to build your own model

The present example, unlike other examples, serves not only to illustrate certain features but also directly as a basis for an own ABM. So, to build an ABM, it makes sense to start with the example and adjust it accordingly.

In the following, the elements to be adjusted are discussed in more detail.

## Network model, zone structure, demand model type

First, the private and public transportation network models, the zone structure, and the connectors must be exchanged. Transport systems, modes, and demand segments are also adjusted or exchanged. This process is not different from the construction of a classical macroscopic demand model.

Since the model is disaggregated in time, the analysis time intervals must be adjusted or extended to your requirements. All analysis time intervals must cover the entire day without overlapping. A calendar is not supported.

A demand model object of the ABM type must be added. The only attributes of this demand model type are the activities. Exactly one of them must be marked as the home activity (this must have the code "H").

The attribution of demand objects required for ABM is described later in the model specification in the Attributing the demand objects section.

## Synthetic population and structural data

The population is represented as a set of concrete persons, called a synthetic population. Similar to macroscopic models, households must be selected so that the marginal distributions are as accurate as possible. Visum does not provide any special functionality for this data preparation (the same applies to the macroscopic case, by the way).

However, there is a lot of software written exactly for this use case. We recommend using the PopulationSim[[1]](#footnote-2) software. It is an open platform, so it can be used freely within the scope of its license. PopulationSim is very well documented, easy to use, and ideally suited for this use case.

The basis of the synthetic population is, as is usually the case with macroscopic demand models, a household survey (which was not necessarily conducted in the model area) and various marginal distributions (number of students, number of employees, ...). In classical demand models, the number of persons per behaviorally homogeneous person groups is estimated for each zone, plus the associated production rates per activity pair or trip chain. Accordingly, concrete households for so-called micro zones are generated for an ABM. The households are then copies of the households from the household survey, including the associated persons.

In the procedure sequence of the example version, the reading of the synthetic population is implemented as a script. Three files are expected in this process:

* HHI.dmd
* SynPopHH.csv
* SynPopPerson.csv

The HHI.dmd file contains data from a household survey:

* The tables Locations and ActivityLocations each create a dummy object that must exist for technical reasons. Both tables are deleted again after being read in. You only have to make sure that the location number is not already used in the model.
* The Households table contains
  + a HH number (it does not play a role in the further course)
  + the household key (it is always identical (H,1) because the exact location is defined in the SynPopHH.csv file)
  + the original household ID, which connects to the corresponding ID in the SynPopHH.csv file
  + as well as any number of other household attributes (in this case with NOCars as the number of passenger cars).
* The Persons table contains
  + the person and household number (these do not play a role in the further course)
  + the original person ID, which connects to the corresponding ID in the SynPopPerson.csv file
  + as well as any number of other person attributes that play a role in choice models
* The Schedules table contains only the corresponding person number and the number of the schedule. Since there is always only one schedule per person, the number is constantly 1.
* The Tours table contains only the associated numbers of the schedule and the person, as well as the number of the tour itself.
* The Activity executions table with the following attributes:
  + The corresponding person and schedule number and the index of the activity execution. The indices correspond to the order of activity executions.
  + Activity execution code; each tour begins and ends with a home activity execution (code "H"). Consecutive tours in a schedule share a home activity execution.
  + The boolean attribute IsMajorActivity: 1 if the activity execution is the major activity execution of the tour (this is the central activity execution of a tour, which significantly influences the destination, mode, and start time choice of all trips and activity executions of the tour). Each tour must have such a major activity execution. A second major activity execution may only exist for sub-tours and only if it is of the same type (see below).
  + Duration of activity execution: each activity execution must have a duration.
  + Activity execution start time: the major activity execution must have a start time. Additional start times are allowed but will be overwritten during the ABM calculation.
  + The boolean attribute IsPartOfSubtour: 1 if the activity execution is part of a sub-tour. Sub-tours may only occur in primary tours (maximum one sub-tour). Primary tours are tours whose major activity is a mandatory activity (school, work).   
    If there are two major activities of identical type in a tour, both together with all activity executions in between form a sub-tour (IsPartOfSubtour = True for all associated activity executions). This ensures that the sub-tour ends at the same point where it has started. In real survey data, more than two work activities may occur in the same tour. In this case, only the first and last are marked as major activity execution (ISMAJORACTIVITY = True).
  + The location number, which is always constant 1 (the location is determined later in the destination choice model)
* The Trips table
  + The corresponding person, schedule and tour number, as well as index of the activity execution. The indices correspond to the order of the trips.
  + The indices of the previous and subsequent activity executions (a trip always lies between two activity executions)

The two SynPop files correspond to the output of a software that generates a synthetic population. SynPopHH.csv contains the location number, the household ID, and the original household ID. The latter is used to place a copy of the original household from the household survey at the respective location. SynPopPerson.csv has a similar structure, it contains the person and household ID, and the original person ID, which is used to create a copy of the original person from the household survey.

**Structural data**

Structural data is modeled in ABM as activity locations, that is, places where a particular activity can be performed. An activity location thus consists of a location and an activity. The potential, e.g. the number of jobs or the size of the sales floor, is stored in the attribute "Attraction potential".

The data does not necessarily have to be available in microscopic resolution. In particular, structural data can be modeled in a way analogous to macroscopic models: a location can stand for a building as well as for an area. However, the more precisely the structural data is located, the more accurate the results will be.

In the example procedure described above, the construction of a model is carried out with example data.

## Generation

The generation in ABM is essentially the same as the first step of a classical 4-step model. The result of the generation for each person are the associated trip chains, i.e. tours, trips, and activity executions, plus the start time of the major activity, but all still without concrete destinations, without modes, and without travel times. The generation in ABM can be data-based, analogous to classical models. However, model-based generation is more common among most ABM modelers. In the following, we address both approaches.

**Data-based generation**

The basis of the generation is the household survey, which was already used in the generation of the synthetic population. In classical demand models, such a household survey is used to derive the trip rates per activity pair or trip chain. Accordingly, for an ABM, individuals are fed their original trip chains. The impact model of the ABM then only determines the destination and mode choice, comparable to a macroscopic 4-step model.

In the procedure sequence of the example version, the feeding of the trip chains to the synthetic persons is implemented as a script. Example files are included with the example. If model-based generation is used, this generation data must be deleted again (see also in the example description).

When generating the DMD file containing the trip chains from the household survey, you should follow the sample data provided in terms of format (the format is described above in the description of the synthetic population).

The advantage of data-based generation is the high degree of consistency of trip chain elements, which is very difficult to achieve with model-based generation. In particular, the interaction of activity sequences and their durations as well as tours executed one after the other match automatically and are realistic by definition.

**Model-based generation**

The model-based generation simulates decisions for all elements one after another: first the number of tours, then the number of activities per tour, the type of each activity, and so on. The generation model is performed by the script run\_tripchain\_choice.py. For more details, see the model specification below.

The main difference to data-based generation lies in the calibration: while data-based generation by definition exactly matches the observed data, this must first be ensured by calibration in the case of model-based generation. Calibration is performed along the chain of submodels, where each submodel is calibrated separately. In each case, model results are compared with observations, and model parameters are adjusted so that the two differ only slightly.

Such a model calibration can sometimes be very lengthy. Each submodel has a lot of influencing factors, the effects of which often cannot be well estimated. While the submodels in the generation model act sequentially and essentially independently of each other, the reality is much more complex. When choosing a particular schedule, in reality, all partial choices, e.g. number of tours, length and duration of activity executions are interrelated and are not performed sequentially but together.

An advantage of model-based generation is its potential reactivity to changes. When traffic or structural changes occur, a generation model could respond appropriately. A data-based generation model cannot do this to this extent; it is essentially constant.

Another advantage is the usually higher variability of the trip chains. Data-based generation can only reproduce the trip chains that were collected in a survey. If the sample numbers are small, the variability of the surveyed mobility is correspondingly small. This can lead to parallel uniform movement patterns of a great number of persons and in extreme cases unrealistic results.

In model-based generation, it is important to note that the probability distributions of the start times of activity executions refer to major activities and not to all activity executions.

**Comparison of the two generation models**

The effects of structural changes on the composition of trip chains and their temporal location are likely to be small. For most scenarios, it is probably also not at all clear in what way and to what extent changes are to be expected. Suppose travel times were to increase by 10%: which changes in production would this imply and to what extent? Outbound trips could be started earlier, return trips later, activity executions could be shorter, activities could be skipped, tours could be combined. All responses seem plausible, but there is no data basis for any of them and thus no calibrated model.

On the other hand, a data-based model can also be used to model responses to structural changes in the way that macroscopic models have always done: the effects are determined in an external model and then implemented by shifting the shares of person groups. For example, if you expect a reduction in car availability, you reduce the share of people with car availability in a macroscopic model. In an ABM, this can be done in the same way: the generation of the synthetic population is based on marginal distributions, e.g. also the share of people with car availability. If you change this share based on an external assessment and generate a new synthetic population, it can serve as an appropriate future scenario[[2]](#footnote-3).

**Tours overlapping in time**

Model-based generation can create tours that would not be feasible in reality due to temporal overlaps: the first tour ends later than the second tour begins[[3]](#footnote-4). In addition, activity executions can also occur at very unlikely times. You could easily change this by deleting such tours. The submodels would then have to be calibrated accordingly concerning the remaining tours so that the correct marginal distributions are achieved again.

However, in most cases, this effort is not necessary: the goal of an ABM is not to make all tours feasible, but to improve responsiveness to structural or traffic changes. Thus, an adjustment of the generation model is only necessary if scenarios are to be considered in which the conditions for valid tours change. This would be the case, for example, if the effects of changed opening hours on the generation are to be investigated. Another case would be the reduction of mobility due to longer travel times. Since such questions are currently rarely answered using transportation models alone, this limitation is probably rather insignificant.

Thus, in the present model, we refrained from modeling tours without temporal overlap. Unfortunately, this makes the results appear less descriptive than perhaps expected. In particular, the start and end times of home activity executions are not consistent between two tours: they only ever match one tour.

## Behavioral data

Behavioral data consists essentially of model parameters (including time series, which represent the distribution of start times of major activities). For the case of mode and destination choice, these can be taken from macroscopic models. However, the calibration constants must be re-estimated in the process.

Calibration is performed analogously to macroscopic models: modal split and journey distance distribution are calibrated[[4]](#footnote-5). Since the microscopic result structure is usually not the focus of the analyses, calibration is performed at the level of zones.

If new influencing variables are introduced, the associated parameters must of course be re-estimated. Such estimation can often be done based on a mobility survey: the parameter is adjusted until the model results resemble the observed ones.

# Specification of the activity-based demand model

The implementation of the model distinguishes two areas of the model specification. The architecture of the individual choice models and the order in which they are simulated are defined by the concrete implementation in the Python code.

In contrast, the attributes and parameters used in the choice models can primarily be found in user-defined tables. The data storage in user-defined tables enables a simple overview of the model specifications and facilitates their extension or adjustment if this is necessary, e.g. in the course of model calibration.

The following model description is intended as a reference. It is not necessary to know all the details in advance to create a model.

## Attributing the demand objects

In addition to the user-defined tables, there are other attributes of demand objects that play a role in model specification:

* Modes
  + ID: The ID is used to identify the mode. This is important, for example, for the mode choice model. Modes with IDs that do not appear as alternatives in any choice model are irrelevant.
  + Interchangeable: Defines whether the mode is interchangeable within a tour or not (usually, e.g. public transport modes and PuT Walk are interchangeable, Car is not).
  + Rank: The interchangeable modes must have a unique rank. If a tour is performed with different interchangeable modes, the tour mode will be the one with the highest rank.
* Activities
  + ID: The ID is used to identify the activity (see, for example, the "AddData" column in the first rows of the Model Steps table). Activities with IDs that do not appear as alternatives in any choice model are irrelevant.
  + IsHome: Exactly one activity with code "H" must be a home activity.
* Locations
  + The locations attribute "Zone number" is not relevant. The allocation of a location to a zone is done via the coordinates of the location and the zone polygon. Thus, you must ensure that the zone polygons do not overlap and that all locations are within a zone polygon.
* Activity locations
  + Attraction potential: The potential of an activity location. It is used in the destination choice models.
* Households  
  In some choice models, the attribute NOCars (number of cars), i.e. the number of cars in the household, is used. If the model specification is changed, this attribute may no longer be needed.
* Persons  
  The following attributes are used in the choice models currently in use. If the model specification is changed, these attributes may no longer be needed.
  + age
  + car availability: true if the person has a car at his disposal
  + driving license (for cars)
  + employment percent: employment level; 100 for full-time employees, correspondingly less for part-time employees
  + is primary pupil, is secondary pupil, is apprentice, is student: 1 each if the characteristic applies to the person.
* Time series
  + Time series per activity are required for model-based generation. The weight of a time period corresponds to the probability that the associated activity (as the major activity of a tour) starts within the time period. The allocation of the time series to the corresponding activity is done in the corresponding sub-model of the start time choice (see also the user-defined table StartTime).

The following attributes are either taken directly from a mobility survey (data-based generation) or added automatically during model execution (model-based generation model).

* Activity executions
  + Index: A unique index (within a schedule)
  + Start time: Each execution of a major activity (central activity of a trip chain, see also below) requires a start time.   
    The start times of all other activity executions result from the start time of the major activity and the travel times of the trips.
  + Duration
  + Activity code
  + IsMajorActivity: Each tour usually has exactly one major activity execution. This is the central activity execution and essentially determines the destination and mode choice of all activity executions. If the major activity execution is primary (i.e., a mandatory activity), the associated destination choice always results in the corresponding long-term choice.  
    Primary tours can have (at most) one sub-tour, in which case there are two major activity executions of identical type. They form the start and end of the sub-tour respectively (in this case the attribute IsPartOfSubtour must also be set accordingly). A tour may not have more than two major activity executions.
  + IsPartOfSubtour: A sub-tour is an intermediate tour that starts at the workplace and returns there, e.g. W-L-W within the trip chain H-W-L-W-H. A classic example is lunch.
* Trips
  + Index: Defines the order of the trips within the tour.
  + FromActivityExecutionIndex, ToActivityExecutionIndex: Origin and destination of the trip.

## Model steps

A central role is played by the table Model Steps, which for each choice model (column "ChoiceModel") links demand strata with the corresponding model specification. The list contains all model steps that are carried out during a program run. The order used has no influence on the order of the submodels, which is given by the Python code. If the data-based variant is to be calculated, the rows with the numbers 201 - 701 have no meaning.

In each row, the column "ChoiceModel" defines to which submodel the respective row belongs.

The demand stratum is defined by the column "Filter": Only those data records are selected for the current simulation for which the specified condition is fulfilled. The attributes specified in the filter refer to different reference objects, depending on the respective choice model. For example, the reference object for long-term choices is "Person", while the reference object for the number of tour stops is "Tour". The table below in the model parameters section provides an overview of the reference objects for the respective choice models.

The “Specification” column contains the name of the table that contains the model parameters (see below).

In some cases, a further parameter for the model specification is necessary, which is then stored in the column "AddData". This is the case for the first rows. They indicate the ID of the activity for which the choice is made. The activity ID is defined in the activity list. The value 5 in the first row, for example, means that the location choice is carried out for activity E1 (Education primary Pupil).

## Model parameters

The other tables contain model attributes and associated parameters (see table below). The structures of the lists differ and are related to the implemented choice model. The exact meaning of the respective entries in the lists results from the program code.

**Tables for the model-based and data-based variants:**

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Choice | Reference object | Model |
| PrimLocStu | Choice of school/training place | Person | Destination choice model |
| PrimLocAdult | Choice of the workplace | Person | Destination choice model |
| DestMajorStu | Destination choice for the major activity for pupils and students | Activity execution at the destination of the trip | Destination choice model |
| DestMajorAdult | Destination choice for the major activity for adults | Activity execution at the destination of the trip | Destination choice model |
| ModeMajorStu | Mode choice for the trip to the major activity for pupils and students | Major activity execution | Standard choice model |
| ModeMajorAdult | Mode choice for the trip to the major activity for adults | Major activity execution | Standard choice model |
| DestMinorStu | Destination choice for minor activities for pupils and students | Activity execution at the destination of the trip | Destination choice model |
| DestMinorAdult | Destination choice for minor activities for adults | Activity execution at the destination of the trip | Destination choice model |
| ModeMinorStu | Mode choice for trips to minor activities for students | Activity execution at the origin or destination of the trip | Standard choice model |
| ModeMinorAdult | Mode choice for trips to minor activities for adults | Activity execution at the origin or destination of the trip | Standard choice model |

**Tables for the model-based variant:**

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Choice | Reference object | Model |
| TourFreqE | Number of educational tours | Person | Standard choice model |
| TourFreqW | Number of work tours | Person | Standard choice model |
| TourFreqO | Number of other tours | Person | Standard choice model |
| StopFreqInbW | Number of tour stops of a work tour between workplace and residence | Tour | Standard choice model |
| StopFreqInbE | Number of tour stops of an educational tour between school/training place and residence | Tour | Standard choice model |
| StopFreqInbO | Number of tour stops between major activity and residence | Tour | Standard choice model |
| StopFreqOutbW, StopFreqOutbE, StopFreqOutbO | Number of tour stops between residence and major activity | Tour | Standard choice model |
| SubFreqW | Number of sub-tours of a work tour | Tour | Standard choice model |
| SubFreqE | Number of sub-tours of an educational tour | Tour | Standard choice model |
| ActType | Activity type | Activity execution | Standard choice model |
| ActDur | Duration of an activity execution | Activity execution | Continuous choice model |
| StartTime | Choice of the start time of a tour | Activity execution of the major activity | Complex choice model (see Python script code) |

## Partial models

In the following, all partial models of the ABM, that is, the choice models implemented in the Python script, are briefly described. For better differentiation, we use the following definitions:

* Primary: an activity is referred to as 'primary' if it is a mandatory activity (in this model work or education). The mandatory activity is always carried out at the same location: it is the one chosen when the long-term choices are made. A tour is called "primary" if it contains a mandatory activity.
* Secondary: activities, tours, and locations are called "secondary" if they are not primary.
* Major: an activity is a major activity if it is the determining or central activity of a tour. The major destination and mode of a tour is determined based on the trip from the home location to the location of the major activity. The tour stops (of the minor activities) are located between these two locations.

In the case of a primary tour, the associated primary activity is always the major activity.

* Minor: an activity is referred to as a minor activity if it is not a major activity.

Each model step relates to a specific Visum object. For example, the choice of locations for primary activities refers to the person. The filter definitions (in the Model Steps table) and the utility functions (in the respective model specification table) each refer to this object, i.e. the attributes specified there are attributes of this object (in the example just mentioned they are therefore always person attributes).

The headers of the following sections each contain in parentheses the name of the corresponding choice model from the Model Steps table.

**Submodels for the model-based and data-based variant**

Long-term choices: Locations of primary activities (PrimLoc)

Each person and each primary activity selectable by that person will be assigned a location. These choices correspond to the long-term choices on workplace and school place. The difference to other choice decisions is that they are often not simulated again in scenario calculations since these decisions normally do not change due to minor changes in the traffic situation.

The choice is made based on accessibility and attraction potential for each demand stratum, and uses destination-bound binding across all demand strata. It thus corresponds to a typical destination choice model in macroscopic models.

The accessibility results from a LogSum matrix, which is composed of the travel times of all available transport modes (see also step 4 above in the section "Model calculation"). The ID of the activity for which the location is chosen is defined in the "AddData" column of the Model Steps table.

The choice of the long-term choice, like all destination choices in this model, takes place in two steps: first, a macroscopic choice is made based on zones, which is then discretized to locations in the second step. In both cases, the attraction potential comes from the attraction potentials of the activity locations, with the first step aggregating over all locations within the zone polygon.

The specification of the model can be found in the tables "PrimLocStu", "PrimLocWithCar", and "PrimLocWithoutCar". The reference object is the person. It consists of only one row per population segment. The utility of a destination is accordingly defined as

u = parameter \* matrix().

If necessary, further matrix and attribute expressions can be added at this point, which would add up to the total utility. The choice model is a logit model based on the utilities.

Destination choice for major activities (DestMajor)

The destination of the major activity (= major destination) is chosen for each tour. In the case of primary tours, the choice is trivial: it is the primary location defined at the beginning (see long-term choices). In the other cases, the destination choice is like the choice of the primary locations. However, no destination-bound binding is performed, as this has already been considered in the long-term choices.

The specification of the model can be found in the tables "DestMajorStu" and "DestMajorAdult". They have a similar structure as the choice model for the locations of primary activities (PrimLoc). However, the reference object is the activity execution, not the person.

* Attribute expression: scalars as well as person filters, if utility elements shall only apply to sub-populations
* MatrixExpr: If other parameters, such as calibration matrices, are to be used in addition to the LogSum.
* ModeChoiceTable: This entry refers to the LogSum that is calculated based on the specified mode choice table. Since the destination choice is based on zone matrices but the mode choice is based on shortest path searches, it is still necessary to specify which matrices correspond to the shortest path searches and replace them in the mode choice. This is done in the "DestinationImpedance" column in the mode choice specification table. If time-varying matrices are involved, this must be defined in the matrix definition by the addition "[FromTime] = CONTEXT[FromTime] & [ToTime] = CONTEXT[ToTime])".

To calculate the utility, all components are multiplied by each other for each row, i.e.

Attribute expression \* MatrixExpr \* LogSum(ModeChoiceTable).

The utility then results from the addition of all row results.

The model parameters of the destination and mode choice are intertwined since the mode choice utility functions enter the destination choice as a LogSum. Therefore the destination choice is also defined by the tables "ModeMajorStu" and "ModeMajorAdult".

Mode choice for major activities (ModeMajor)

The major mode of the tour is determined for each tour. It is chosen based on the travel times (outward and return) between the residence and the major destination (even if the tours "residence à major destination" or „major destination à residence“ do not exist at all because of possible stopovers). If the chosen mode is interchangeable, the major mode is not yet specified but defined as "interchangeable". The specific choice of the mode takes place in a later step.

The specification of the model can be found in the tables "ModeMajorStu" and "ModeMajorAdult". It is very similar in structure to the lists in the previous sections. The following parameters must be set:

* Attribute expression: scalars as well as person filters if utility elements shall only apply to sub-populations
* MatrixExpr: If the impedance from the shortest paths is to be corrected on matrix level.
* DestinationImpedance: This entry refers to the generation of the LogSum in the destination choice model, see there.
* ModeImpedance: The entries PrTShortestPathImpedance(), PuTShortestPathImpedance(), and DirectDistance() are possible here.
  + The argument of the PrTShortestPathImpedance() function is the name of the attribute containing the impedance. It must be defined for all objects that are part of the PrT impedance, i.e. for links, nodes, (main) turns, connectors, as well as traffic areas. In the example version, these attributes are created and assigned in the procedure sequence. If time-varying attributes are involved, the corresponding time is automatically considered correctly during mode choice.
  + The argument of the PuTShortestPathImpedance() function is the name of the corresponding demand stratum. The public transport shortest path search is based on stop area matrices of the impedance (code IPD) and journey time (code RIT and ADT). The demand segment and the time interval are also considered accordingly. The walk to the stop area is calculated in the network for the transport system stored in the global attribute WalkPrTSys (this is defined in the user-defined table Global attributes). The weight of the walk time relative to the impedance from the stop area matrix is defined in the global attribute WalkTimeImpedanceFactor, and the maximum walk time is defined in the attribute MaxWalkTimeInMinutesForPutShortestPathSearch.
  + DirectDistance() does not contain an argument. The conversion of the direct distance into walk time is done using the global attribute DirectDistanceSpeed from the Global Attributes table.
* Bike, Car, etc.: Parameters for the respective modes. They are chosen as in macroscopic models.

To calculate the utility, all parameters of a row are multiplied together and summed up over all rows.

Destination and mode choice for minor activities

The destination and mode choice for minor activities are carried out together. One by one, destinations and the associated modes of the trips to or from the destinations are determined. It starts with the first destination before the major activity and then proceeds with all activity executions one after the other until the home activity is reached. Then it continues with the first destination after the major activity and calculates again until the home activity is reached.

The filters defined in the Model Steps for destination and mode choice for minor activities must not overlap and must collectively cover all activity executions.

Destination choice for minor activities (DestMinor)

A location is chosen for each activity that does not yet have a location. The choice is like the destination choice for major activities but uses the Rubberbanding method.

In the Rubberbanding method, the tour stops are successively inserted between the home and major activities. The choice is based on the sum of the travel times between the new destination resp. next destination. So, for example, if a destination is inserted between the home and major activity, the choice of that intermediate destination is based on the sum of the travel times between the home activity and the new destination, and between the new destination and the major activity. Thus, when choosing intermediate stops, the trips to and from the intermediate stop are always taken into account (and tend to be chosen as minimally as possible).

The specification of the model is identical to the specification of the destination choice of the major activities (DestMajor).

Mode choice for minor activities (ModeMinor)

For each activity execution, the mode of the associated trip is selected. According to the process mentioned above, alternately the trip to or from the activity execution is processed.

The available modes are the major mode and for sub-tours also the interchangeable modes.

After the mode choice, all trips are assigned their travel times from the shortest path search.

**Submodels for the model-based variant**

Choice of the number of primary tours (TourFreqPrim)

For each person, an appropriate number of tours is chosen for each primary activity selectable by that person. The ID of the activity for which the number of tours is selected is defined in the "AddData" column of the Model Steps table.

When defining the population segments (i.e. the filter in the Model Steps table), make sure that they are compatible with the corresponding definitions in the choice of the Locations of primary activities (PrimLoc). If the number of primary tours is selected for a person, a location for the corresponding activity must already have been selected. If this is not the case, the Python program terminates with an error message.

The specification of the model is found in the "TourFreqE" and "TourFreqW” tables, the reference object is the person. Columns Result0, Result1, Result2, and Result3 contain the utility parameters for the respective alternatives. Note that the column name does not affect the model. The value of the alternative is defined in the first row (with Attritbute expression = ChoiceID).

The utility of an alternative is the sum of the products of the attribute expressions and their associated parameters. If required, further attribute expressions can be added at this point. If you want to add another alternative, you must create another user-defined attribute (in this case a UDA named "Result4" would be suitable).

The choice model is the standard choice model for discrete choices with a few alternatives mentioned above in step 7 in the section "Model calculation".

Choice of the number of secondary tours (TourFreqSec)

The number of secondary tours is selected for each person.

The specification of the model can be found in the "TourFreqO” table. It has the same structure as the lists in the previous section. The reference object is the person.

The choice model is identical to the standard choice model above.

Choice of the number of tour stops (TourStopFreq)

The number of additional inbound and outbound stops is chosen for each tour. Outbound stops correspond to activities on the way to the major activity.

The choice model again corresponds to the standard choice model.

The specification of the model is in the tables "StopFreqInbE", "StopFreqInbO", "StopFreqInbW", "StopFreqOutbE", "StopFreqOutbO", and "StopFreqOutbW". They have the same structure as the lists in the previous sections. The reference object is the tour, the choice model is the standard choice model again.

Choice of the number of sub-tours (SubtourFreq)

For each primary tour, it is decided whether there is a sub-tour and if so, how many tour stops it has. A sub-tour is a tour that starts and ends at the location where the primary activity is performed. It has one or two stops, after which the path leads back to the initial location. Sub-tours can only occur on primary tours.

The specification of the model can be found in the "SubFreqE" and "SubFreqW” tables. They have the same structure as the lists in the previous sections. The reference object is the tour, the choice model is the standard choice model again.

In the choice model, 0, 1, or 2 are available as alternatives. The alternative 0 stands for "no sub-tour", 1 stands for "sub-tour with one stop", 2 stands for "sub-tour with 2 stops".

Choice of the activity type (ActType)

The activity associated with each stop is selected. Since primary activities only appear as major activities of primary tours, only secondary activities can be selected at this point. In particular, the major activities (i.e. the central activities) of secondary tours (i.e. tours that do not have a primary activity) are also defined here.

The specification of the model can be found in the "ActType” table. It has the same structure as the lists in the previous sections. Note that the alternative value (i.e. the activity type) is not defined by the column header (i.e. the name of the user-defined attribute). The values of the alternatives to be selected are in the row with Attribute expression = ChoiceID. The IDs belonging to the activity types are listed there. The IDs are defined in the activity list.

The reference object is the activity execution, the choice model is the standard choice model again.

Choice of the duration of the activity executions (ActDur)

For each activity, the duration of its execution is chosen. The choice model differs from all other previous discrete choice models. Each demand stratum has a suitable distribution function with certain parameters based on which the duration of the activity executions is chosen.

The specification of the model is in the table "ActDur", the reference object is the activity execution. The lists are structured slightly differently than the previous lists. The attribute 'Attribute expression' acts like a filter here: the condition specified defines the demand stratum. The other parameters define the distribution function for the duration of the activity execution.

The models for activities from sub-tours and other activities differ. The duration for sub-tours is on average shorter.

Three different distribution functions are available for the activity durations: Weibull, Normal, and LogNormal.

Choice of the time-of-day (StartTime)

When selecting the time-of-day, only the start time of the major activity is selected for each tour. All other times result automatically from this since both the duration of the activity executions and the travel times are known. The start times of the activity executions are chosen in such a way that, aggregated over all activity executions, the time series belonging to the activity is matched as accurately as possible.

The time-of-day model implemented in the script example is rather simple and does not represent a utility-based choice, unlike most of the other sub-models used here. It reproduces as accurately as possible the fixed distribution of the start of an activity execution and is therefore not sensitive to changes in supply. The start times of different tours are not checked for overlap. It can therefore happen that tours of one person occur in parallel.

Such a model is quite sufficient for many analyses. However, for certain analyses it may be necessary to choose a different time-of-day model.

The specification of the model is in the "StartTime” table, the reference object is the activity execution of the major activity of a tour. The list is similar to the previous one. The attribute 'Attribute expression' again acts as a filter for defining the demand segments. The attribute ‘TimeSeriesNo’ contains the number of the standard time series. The time series can be found in the menu Demand > Demand data on the Standard time series tab.

# Tips and tricks

The following list is the result of our experience in various pilot projects. If the present Visum example is used as a starting point for the development of your own ABM, you should consider the following points:

* In general, you will use different person attributes than those in the example. This can be done without any problems as long as you make sure that the appropriate attributes exist and that the filters in the Model Steps table and the utility functions in the model parameters tables are adjusted. It has proven to be useful to simultaneously keep the corresponding list containing the utility attributes open when adjusting the utility functions or filters. If, for example, the utility function is adjusted to the number of primary tours (TourFreqPrim), the Persons list should be opened at the same time so that no typing errors occur when selecting the person attributes.
* If another term is to be added to a utility function, a row must be added to the corresponding user-defined table. The additional utility term can then be written in this row.
* If another alternative is to be added to a model specification, an additional user-defined attribute of the associated table must be created. For reasons of clarity, the name of the attribute should be based on the value of the new alternative. In the table row with the entry "ChoiceID" under Attribute expression, the expression of the new alternative must be specified.
* In the present example, for most choice models, there are at most two sets of model parameters: one for adults and one for students. If an additional parameter set is to be created for a choice model (e.g. for pensioners), a new user-defined table must be added for this purpose. This table must then be linked to the sub-model as well as the corresponding demand stratum in the Model Steps table.
* During the modeling process, it is useful to calculate the ABM step by step and to check the results of each step before the next step is calculated. You can do this very easily by commenting out several steps in the Python script. To do this, you can add three double quotation marks in the file abm.py as the start and end of the commented-out sections. So, for example, if only the TourFreq step is to be calibrated, you would adjust the tripchain\_choice function in the abm.py script as follows:

def tripchain\_choice(self):

...

logging.info('--- tour frequency choice ---')

self.tour\_frequency\_choice()

"""

logging.info('--- stop frequency choice ---')

self.stop\_frequency\_choice()

logging.info('--- subtour choice ---')

self.subtour\_choice()

...

"""

Please keep in mind that the first rows (up to and including self.initialize\_data\_and\_config()) must not be commented-out.

* During the development of the model, it may be necessary to debug the model again and again, i.e. to set breakpoints and thus analyze intermediate states of the script flow. This process can be greatly facilitated by using a development environment such as Visual Studio code. The installation is usually uncomplicated (see also the notes below).
* Since repeated model calculations are often necessary in the course of the modeling process, it is advisable to model only a small number of persons initially. This way the computing time during the development of the model can be kept low.

# Code overview

This section gives a brief overview of the script code. It serves as an entry point and for better orientation in the code. However, it is not a complete specification of the code. To get more detailed information about the implementation of the model, you should look at the code directly.

The script is written in Python and can be opened and edited with the usual editors.

## Overview

The ABM model is divided into several scripts that are normally started from the procedure sequence. For debugging purposes, it can also be started from a development environment.

The abm module contains the class ABM, which contains the actual execution of the ABM script. In the method ABM.run\_full\_abm(), all individual steps of the ABM procedure are called one after the other.

The individual steps are implemented in the src.core submodule. Shared functions of the choice model are located in src.choice\_engine, src.location\_choice\_engine, and src.mode\_choice\_engine. The module src.config allows reading the configuration from the tables of the version file. More useful functions can be found in src.utilities.

## Preparation

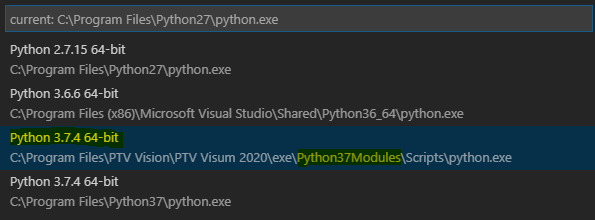
For debugging purposes, a Python development environment is recommended for execution. Among others, PyCharm Community Edition (JetBrains) and Visual Studio Code (Microsoft) are supported. It is also possible to call the script from a console.

All required Python libraries are already installed by the Visum setup. An execution with Python 3 is recommended.

The Visum setup installs a virtual environment (venv) for Python 3, which includes additional packages located in the Exe\Python37Modules subfolder of the Visum installation path. This virtual environment must be used when executing the script.

In Visual Studio Code, for example, the environment used can be set in the lower left corner of the window.





To execute the ABM calculation, you simply need to execute the file run\_abm.py. The method run\_abm defined there contains several calculation options, of which the appropriate one must be set active. If the program is not started from Visum, make sure that the virtual environment of Visum is used.

If the program is not started from Visum, Visum is started first and the version file (Halle\_ABM.ver) specified in the Python code is loaded. Otherwise, the current Visum instance is automatically used, and the script is executed with the currently open version file.

Please note that a calculation of the ABM within a development environment is not identical to a regular calculation through the procedure sequence. In the procedure sequence, various other Visum procedures are carried out between individual script calculations. For example, all procedure steps of the procedure sequence that precede the ABM group must first be carried out before the run\_abm.py script is started from a development environment.

It is also possible to run individual steps of the ABM script separately. To do this, procedures that are not to be executed are commented out in the run\_full\_abm method in the abm.py script. It should be noted that the method ABM.initialize\_data\_and\_config() must not be commented out.

## Logging

The logging module outputs debug output to the console. If the script is executed within Visum, the debug output is saved to the Messages.txt file.

## Use of numpy and dask

The numpy and dask modules are used for the efficient handling of large numerical data structures (arrays/vectors, matrices, and n-dimensional vectors). This is used in many places in the ABM script. Among other things, the entire choice engine works with numpy data structures.

## Reading the configuration tables

The configuration tables are read via the src.config module in the ABM.initialize\_data\_and\_config() method and are available as members (ABM.config) during script execution.

The configuration of a single procedure step is made available by self.config.load\_choice\_para('[name of step]') and contains a list of segments that are considered in this step (see table **Model steps**).

## Carrying out a simple discrete choice

In many ABM steps, a classical discrete choice is performed. This is carried out segment by segment, whereby the segments usually represent a partition (disjoint subdivision) of the object set. The object set is the set of all persons, all activity executions as well as all tours and all trips.

In discrete choice, there are often several aspects that influence the choice. These are displayed in several rows of a choice table. The individual aspects are multiplied by a fixed weight Beta and a variable weight resulting from the evaluation of a formula expression AttrExpr. This formula expression can take into account the properties of the individual object, such as the age of a person.

The discrete choice is executed by calling choice\_engine.run\_simple\_choice(object set, segment). The Segment object that was filled from the tables contains the following data:

* Filter: Filter expression according to which the set of objects is first filtered.
* Choices: The discrete set of choices, usually 0..n or 1..n
* AttrExpr: One formula expression per utility component of the choice. This expression is evaluated individually for each object and multiplied by the constant weight.
* Beta: One fixed weight per aspect of the choice.
* ResAttr: Result attribute to which the selected values are saved.

## Efficient data transfer via COM

When transferring large amounts of data from Visum to the Python script and vice versa, there are several things to consider.

* For the efficient reading and writing of the data the methods VPH.GetMulti() and VPH.SetMulti() or for several attributes Container.GetMultipleAttributes() and Container.SetMultipleAttributes() should be used (VPH stands for the module VisumPy.helpers, which is delivered with Visum, Container for any object container, e.g. Visum.Net.Persons).
* When reading and writing very large amounts of data   
   visum\_utilities.SetMulti(container, attribute, values, active\_only=False, chunk\_size=abm\_settings.chunk\_size\_trips) should be used. If the data for a large number of objects are retrieved at once, this may exceed the maximum transfer size of Pywin32.
* When filtering a set of objects, the Container.GetFilteredSet() method should be used, which directly returns a container with the filtered set of objects. This is to be preferred to the Container.FilteredBy() method, which is evaluated only when the filtered set of objects is actually used, but then again with each call of this set.

This efficient transfer is implemented in each step of the ABM script.

1. PopulationSim is part of the ActivitySim project and was created in collaboration with Oregon DOT (Department of Transportation). See also https://activitysim.github.io/populationsim/ [↑](#footnote-ref-2)
2. The PopulationSim software also offers functions that simulate population evolution. [↑](#footnote-ref-3)
3. To a very small extent, this is also possible with data-based generation. [↑](#footnote-ref-4)
4. If the generation is model-based, it must also be calibrated. [↑](#footnote-ref-5)