

Guidelines for setting up EFDM for uneven-aged forests

In the context of the Specific Contract 15

Contents

1.	EFDM - structure and functioning	1
2.	Defining state space	3
3.	Initial state.....	5
4.	Activities	6
5.	Transition probability matrices	8
6.	Transitions under final fellings	9
7.	Transitions under thinnings and calamities	9
8.	Transitions under no-management.....	12
9.	Estimating transition probabilities for no-management.....	13
10.	Generating outputs	15
11.	Annex 1: checklists for correct spelling of variables' names.....	19
12.	Annex 2: a way to model regeneration	20
	References.....	21

1. EFDM - structure and functioning

EFDM is based on a matrix structure - a state space that defines a number of fixed states, between which the modelled units move over time. EFDM software is highly generic and allows, in principle, for any number of state-space dimensions. It is convenient, however, to separate the state space within which the dynamics occurs from the factors which are assumed to remain static over time. It means we can, and should, set-up different dynamic state-spaces for different forests, as illustrated by figure 1, in which the dynamic state-space is two-dimensional. The different “forestry types” can be defined by the modeler using variables such as site quality, geographical region, species (group), owner (or management) type etc. since (or if) no “natural” transitions between them are intended. It makes sense to separate between different “forestry types” if they can be expected to either grow differently or to be managed differently.

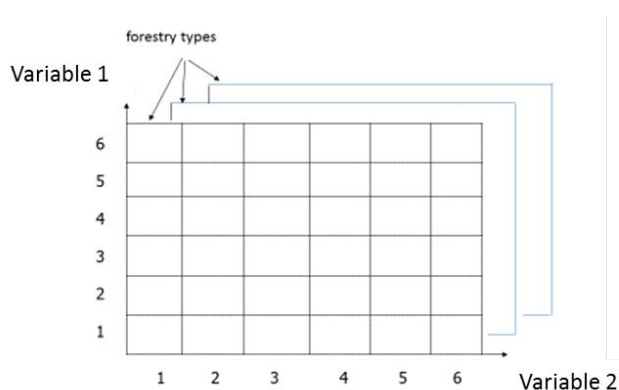


Figure 1. A two-dimensional dynamic state-space and multiple “forestry types”

Even-aged forests have been successfully modelled by area-based matrix models with the dynamic state spaces defined by volume and age. This type of area-based matrix models has showed itself less capable of representing the dynamics of un-evenaged forests. Instead, tree-based diameter class concepts have been widely used. In this project, we are testing a new area-based concept for un-evenaged forests using an enhanced version of the EFDM software. The concept is an area based matrix model in which the dynamic state space is defined by volume and stem number per hectare (Sallnäs & et al. 2015).

Technically, the software “rolls out” the state-space as a one-dimensional vector (regardless of the number of dynamic factors), which allows for realizing the transitions through multiplications with transition probability matrices. The transition probability matrices are square with a side dimension equal to the product of the numbers of classes of the dynamic state space dimensions. For example, 10 stem number classes and 15 volume classes mean that the transition matrix will have a size of 150 times 150.

Figure 2 shows how the sequence of the operations is carried out by EFDM at every simulation step. First, the state-space is sliced into one state-space for each activity. Each state-space is filled with area fractions according to the activity probabilities (step 1). Then the distribution of the forest area

in each of the activity specific state-spaces is modified according to activity-specific transition probabilities by a matrix times vector multiplication (step 2). Finally, the activity-specific state-spaces are merged back into one (step 3). The resulting distribution of forest area in the common state-space describes the state of the forest after the simulated time step.

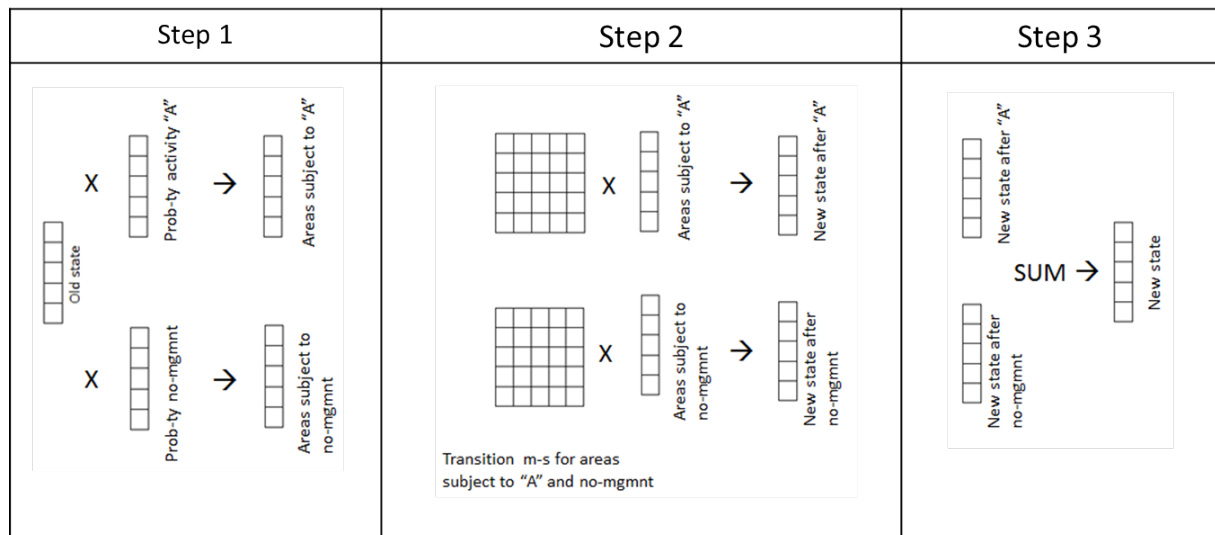


Figure 2. Simulation steps in EFDM

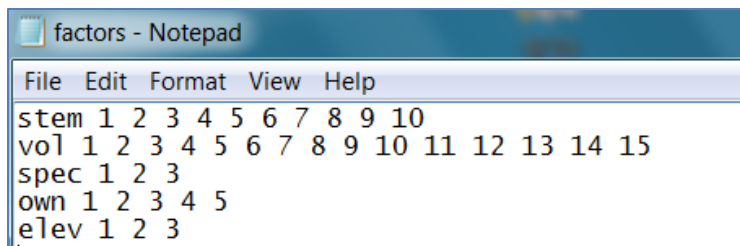
The following sections contain references to and examples from the EFDM package set-up on Austrian data. The names of control and input files can, of course, be freely chosen by the users.

2. Defining state-space

Files and functions involved:

- factors.txt

All information about the size and definition of the state-space should be entered into the file “factors.txt”. Figure 3 is an example of what such a file could look like. In the Austrian example we are working with 15 volume classes named 1,2 ... 15, and 10 stem number classes, named 1,2 ... 10. Together they form the dynamic state space having $10 \cdot 15 = 150$ different states. The static dimensions are species, owner type and elevation class with 3, 5 and 3 levels respectively, defining together 45 “forestry types”. The names of the factors levels do not need to be numeric, character strings can be used as well. Note that the last line in the file must end with “carriage return” for the program to run properly.



```

factors - Notepad
File Edit Format View Help
stem 1 2 3 4 5 6 7 8 9 10
vol 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
spec 1 2 3
own 1 2 3 4 5
elev 1 2 3

```

Figure 3. “factors.txt”

It should however be noted that in “factors.txt” you only give the number and the “names” of the factor levels. The definitions – for example the limits between the volume classes – must be decided when classifying the raw data.

The classes in the dynamic state space should be defined with consideration of the intended simulation time step, which in turn will depend on the time step in the inventory data available. Volume class width could be increasing throughout or follow an s-shaped curve with the class-width increasing to a maximum and then decreasing. In even-aged models both approaches have been successfully applied. In any case, the range of the volume classes must be large enough that forests hit the “roof” of the volume dimension under realistic management only very seldom. Stem class width should be increasing throughout. Tables 1 and 2 show the volume and stem number class limits applied in the tests with Austrian data.

Table 1: Stem classes in the Austrian test

Stem number:	(lower bound, no/ha)
class 1	0
class 2	200
class 3	300
class 4	450
class 5	600
class 6	1000
class 7	1500
class 8	2250
class 9	3500
class 10	5000

Table 2: Volume classes in the Austrian test

Volume:	(lower bound,m3/ha)
class 1	0
class 2	63
class 3	92
class 4	134
class 5	188
class 6	259
class 7	347
class 8	452
class 9	575

class 10	710
class 11	852
class 12	994
class 13	1124
class 14	1232
class 15	1309

When defining the classes, one should bear in mind that more classes mean more possible transitions. Having a state-space of 10 times 15 means 150 different states. The transition matrix describes the probability of a transition from each state to all other state and consequently has the size 150 times 150, which means 150^2 entries. Most of these are 0, but the number of the non-zero entries is still large. Ideally there would be several data points for each transition in order to estimate the transition probabilities properly, but of course this is not realistic. The Bayesian estimator is one way to cope with the lack of data.

Using more classes means that there is less data to estimate the probability for any given transition. A balance between enough classes to get a fine enough description of the forest and few enough classes to still have enough data for the estimation is necessary.

3. Initial state

Files and functions involved:

- initstate.txt

The initial state is described in a file containing area records for every class in the state-space, as defined above. It is up to the user to find a way to produce such a file based on available data. When a simulation is initialized, this file is used to fill the state-space with forest area. The file contains one line for every possible combination of the factors in the file "factors.txt". Note that even classes with zero area have to be present in the file for the model to function correctly. Figure 4 shows a snapshot.

```

initstate - Notepad
File Edit Format View Help
elev own spec vol stem area
1 1 1 1 1 52
1 1 1 1 2 3
1 1 1 1 3 0
1 1 1 1 4 7
1 1 1 1 5 12
1 1 1 1 6 6
1 1 1 1 7 8
1 1 1 1 8 4
1 1 1 1 9 1
1 1 1 1 10 0
1 1 1 2 1 8

```

Figure 4. "initstate.txt"

4. Activities

Files and functions involved:

- actprobs.txt
- activities.txt

Activities imply different transitions in the state-space. Activities can be thinnings, final fellings, no action at all or calamities. The impact of an activity is defined as a certain transition in the state space. For example, a thinning might imply a movement of one volume class and one stem number class down. Final felling will imply a movement down to the lowest volume class and the lowest stem number class, considered bare land. Calamity might imply a stronger impact than a thinning.

Calamities require a few extra words. Calamities have not been considered in the previous applications of EFDM on even-aged forests. However, in the Austrian data used for the tests of the un-evenaged model concept a significant number of negative volume transitions not caused by cuttings were observed. Ignoring the calamities in simulations would make the results misleading. One way to model the calamities in EFDM is to define one or several types of calamities as "activities" in the sense above.

Activity probabilities are essentially proportions of the area in a state-space cell that must be subject to specific activities within a simulation step. The activities are disjoint within one simulation step. Activity probabilities must be recorded in a file containing one line for every factor combination, like in the initial state file, and the probabilities for the different activities. The sum of the probabilities must be equal to one. In the example shown in Figure 5 there are five activities: two types of thinning, final felling, no-management, and calamity.

It is up to the case study teams to either estimate the activity and calamity probabilities for their model tests from inventory data or derive them from expert knowledge and experience. .

```

File Edit Format View Help
"elev" "own" "spec" "vol" "stem" "thHigh" "thLow" "ff" "nomgmt" "cal"
"1" 1 1 1 1 1 0 0 0 1 0
"2" 1 1 1 1 2 0 0 0 1 0
"3" 1 1 1 1 3 0 0 0 1 0
"4" 1 1 1 1 4 0 0 0 1 0
"5" 1 1 1 1 5 0 0 0 1 0
"6" 1 1 1 1 6 0 0 0 1 0
"7" 1 1 1 1 7 0 0 0 1 0
"8" 1 1 1 1 8 0 0 0 1 0
"9" 1 1 1 1 9 0 0 0 1 0
"10" 1 1 1 1 10 0 0 0 1 0
"11" 1 1 1 2 1 0 0 0 0.987 0.013
"12" 1 1 1 2 2 0 0 0 0.98 0.02
"13" 1 1 1 2 3 0 0 0 0.986 0.014
"14" 1 1 1 2 4 0 0 0 0.983 0.017

```

Figure 5. Activity probability file

Activities also must be “described” in a control file containing references and parameters that are needed by EFDM to load or produce correct transition matrices for the portions of the forest under specific activities (Figure 6).

```

File Edit Format View Help
nomgmt read nomgmtP.RData stem vol
ffel read ffelP.txt stem vol
thHigh read thHigh_nocalP.txt stem vol
thLow read thLow_nocalP.txt stem vol
cal read calamp.RData stem vol

```

Figure 6. “activities.txt”

This file "activities.txt" tells the program whether the transition probabilities for the different activities must be read from a file or estimated in the course of model runs. If they are to be estimated, the third parameter in the line is the name of the control file for the estimation function which will be described in more detail later. For the activities other than “nomgmt”, “read” is currently the only possible option. “Read” must be followed by the name of the file with the transition probabilities. Every line is concluded with the names of the dynamic state space dimensions. Note that the order in which the dynamic state space dimensions are mentioned must be the same as in the “factors.txt”.

5. Transition probability matrices

Files and functions involved:

- `ffellP.txt`
- `thHigh_nocalP.txt` (or `.RData`)
- `thLow_nocalP.txt` (or `.RData`)
- `calamP.RData`
- `nomgmtP.RData`

Transition probability matrices for the different activities (including no management) are stored in files and must be available prior to running a simulation (except no-management, for which estimation during model runs is also possible). There must be one file for each activity.

The transition probability matrices are square with a side dimension equal to the product of the numbers of classes of the dynamic state space dimensions. For example, 10 stem number classes and 15 volume classes mean that the size of a transition matrix will be 150 times 150.

It is very important to make sure that the “nesting order” in the transition matrices corresponds to the order of the factor combinations in the state space which is based on “factors.txt”. If, for example, stem number is on the first line of “factors.txt” and volume is on the second, the transitions probability matrices should be seen as $vol \times vol$ with nested $stem \times stem$ matrices. If you use the EFDM functionality (`pre.estimate(..)`, see following chapters) for estimating the no-management transition probabilities from your pair data, the program takes care of this.

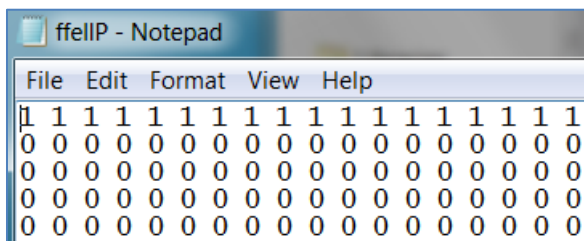
The files can be in “.RData” or “.txt” format. If you use “`efdmsetuptools`” functionality (`estimatetransprobs(..)`, `pre.estimate(..)`, `makeanewP(..)`) for estimating the probabilities and writing the files, the files will be in “.RData” format. Some of the transition probability files in the example based on Austrian data have been compiled using older versions of EFDM and therefore are in the “.txt” format. In case of final felling in which one and the same transition matrix is used for all “forestry types” there must not be any header and only a single matrix in the file (see the next chapter).

6. Transitions under final fellings

Files and functions involved:

- ffellP.txt

Transition probability matrix for forest under final felling is simple. All areas move to the state-space class which is considered “bare land” (Figure 7).



1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7. Transition probabilities for final felling

7. Transitions under thinnings and calamities

Files and functions involved:

- thHigh_nocalP.txt (or .RData)
- thLow_nocalP.txt (or .RData)
- calamP.RData
- efdmsetuptools.r
 - o makeanewP(...)

In EFDm, we assume that after partial wood volume removal due to an activity, the forest grows in the same manner as any other forest in the state-space cell. Eventual thinning effects on subsequent growth are not modelled explicitly. The same applies for calamities.

Thus, the transition probability matrices for thinnings or calamities must combine the transitions due the direct impact of the activity and the transitions due the consequent growth. This allows for using a single vector times matrix multiplication in the model core.

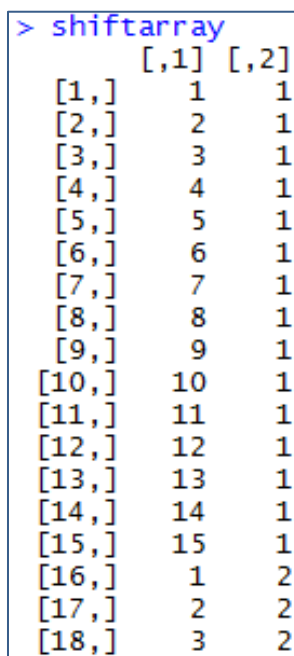
The function “makeanewP(...)” of the “efdmsetuptools.r” performs such combination. The function’s arguments are “inputfilename”, “shiftarray”, “shiftdims” and “resultfilename”. The “inputfilename” refers to the name of the file with transition probabilities for no management (in our nomenclature,

“nomgmtP.RData”). The “shiftarray” describes the direct effect of an activity, such as thinning or calamity, by specifying a new state for each old state (defined as a combination of different levels of the dynamic dimensions). It must be a table with the number of rows equal to the product of the number of levels of the dynamic dimensions and the number of columns equal to the number of the dynamic dimensions.

When running the function this table must be in the R workspace, specifying a file name won't do. The contents of the table depends on your definition of the activity i.e. how the activity changes the volume and the stem number classes of the plots. To better understand this, you could create a table using the following code line:

```
shiftarray <- cbind(rep(1:nvol,times=nstem),rep(1:nstem, each=nvol))
```

the resulting table is shown in Figure 8 below.



	[,1]	[,2]
[1,]	1	1
[2,]	2	1
[3,]	3	1
[4,]	4	1
[5,]	5	1
[6,]	6	1
[7,]	7	1
[8,]	8	1
[9,]	9	1
[10,]	10	1
[11,]	11	1
[12,]	12	1
[13,]	13	1
[14,]	14	1
[15,]	15	1
[16,]	1	2
[17,]	2	2
[18,]	3	2

Figure 8. “shiftarray” table when activity has no effect

The table in figure 8 can be read as:

First line: Move from volume class 1, stem number class 1 to volume class 1, stem number class 1.

Second line: Move from volume class 2, stem number class 1 to volume class 2, stem number class 1.

16th line: Move from volume class 1, stem number class 2 to volume class 1, stem number class 2, and so on.

The elements of the matrix refer to the 'to' part in the sentence above. The part 'from' part is determined by the row number and is fixed. The rows cover the entire dynamic state space, first running through all volume classes in the first stem number class, then through all volume classes in the second stem number class, and so on.

As you see, in this example the volume and stem number classes remain the same, in other words there is no activity effect.

If an activity is supposed to move the element of the state vector one volume class down but remain in the same stem number class, the code for 'shiftarray' would be:

```
shiftarray <- cbind(rep(c(1,1:(nvol-1)),times=nstem),rep(1:nstem, each=nvol))
```

```
> shiftarray
```

	[,1]	[,2]
[1,]	1	1
[2,]	1	1
[3,]	2	1
[4,]	3	1
[5,]	4	1
[6,]	5	1
[7,]	6	1
[8,]	7	1
[9,]	8	1
[10,]	9	1
[11,]	10	1
[12,]	11	1
[13,]	12	1
[14,]	13	1
[15,]	14	1
[16,]	1	2
[17,]	1	2
[18,]	2	2

Figure 9. “shiftarray” table when activity has the effect of one volume class drop and no effect on the stem number class.

Care needs to be taken that the volume class 1 is not moved to volume class 0 which does not exist (in our example). So this activity needs to keep volume class 1 in volume class 1, which is done by replacing 'rep(1:nvol...)' not by rep(0:nvol-1...), but instead by rep(c(1,1:nvol-1)...). The resulting matrix (Figure 9) is:

The first line is as above. The second line is read as:

Move from volume class 2, stem number class 1 to volume class 1, stem number class 1.

The “shiftdims” is n-element vector giving the number of levels for each of the dynamic dimensions. In the “resultfilename” you must use “.RData” extension. For more technical information on the function arguments see the programmer’s comments in the script.

Within one transition probability file, the direct effects' component of the transitions is uniform across all the "forestry types" (sub-state spaces as defined by the static factors). However, you can define several different activities with separate transition probability files and steer their allocation over the different "forestry types" by the probabilities in "actprobs.txt".

8. Transitions under no-management

Files and functions involved:

- nomgmtP.RData
- estimatetransprobs(...)
- pre.estimate(...)

Transition probabilities for no management areas can be read from a previously produced file (like "nomgmtdataP.Rdata") or estimated automatically in the course of every simulation run (this is specified in "activities.txt"). In the latter case, EFDM calls the function "estimatetransprobs(...)" (from "efdmestim.r" file) to conduct the estimation. Then the probability matrices are not saved to a file but stay as an object in the workspace only during the work session.

The probabilities can be also estimated beforehand and written to a file. If there is a need to produce the transition probability files for other activities than no-management, a file with no-management transition probabilities is required as an input.

The next chapter describes in more detail how to set up the transition probability estimation in EFDM.

9. Estimating transition probabilities for no-management

Files and functions involved:

- nomgmtdataP.Rdata
- estiminput.txt
- pairedata.txt
- efdmsetuptools.r
 - o pre.estimate(...)

The transition probabilities for no-management can be estimated from data and written to a file with the help of the “pre.estimate (...)” function from “efdmsetuptools.r”. The function “pre.estimate(...)” is an envelope function which uses the `estimatetransprobs(...)` to do the actual job but it makes sure the resulting object is saved to a file.

Control file “*estiminput.txt*”

The “estiminput.txt” (figure 10) is a control file used by the functions “`estimatetransprobs(...)`” and “`pre.estimate(...)`”. The first line must contain the name of the file with the data for estimation (“nomgmtdata.txt”).

In the second line the method of obtaining “prior” for the Bayesian estimation algorithm must be specified. If it is “uninformative”, the prior will be produced by the function `estimatetransprobs(...)` in the course of the estimation. The “uninformative” prior in this case is a unit matrix with row and column number equal to the product of the numbers of levels of the dynamics state space dimensions. It means the “prior-option” if no observations are available is to stay in the same state-space class. The priors can be read also from a file in “.txt” format. Then instead of “uninformative” you need to specify the file name.

The third line contains the names of the non-changing dimensions of the total state space defining what we called “forestry types”. The order in which these dimensions are listed is from more to less influential. The associated numerals are the “weights” used in the estimation procedure.

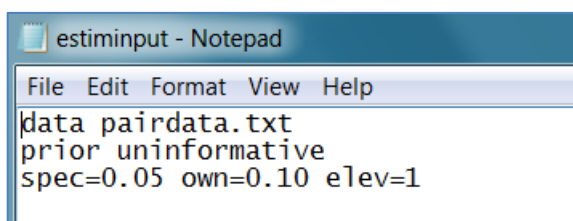


Figure 10. “estiminput.txt”

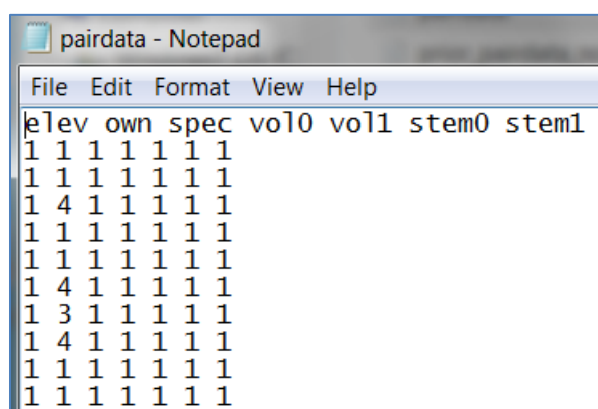
Estimation approach and the static factor weights

The Bayesian algorithm works in a recursive manner. At the first round, observations of transitions

(as counts of plots) are pooled from across all the static dimensions of the state-space into one transition matrix and added to the prior's matrix. If the estimation procedure stopped at this stage, there would be no difference between transition probability matrices of all the different "forestry types" as defined by the static dimensions. At the second round, observations are pooled across all the static dimensions but one (the first entry in the 3-rd row of "estiminput.txt") and then multiplied by the weight assigned to this dimension, after which the previously accumulated observations (i.e. from the first round) are added to each of the now several (the first static dimensions has several levels!) transition matrices. At the third round, observations are pooled across all the static dimensions but two (the first and the second entries in the 3-rd row of "estiminput.txt") into now yet more transition matrices. The algorithm proceeds in the same manner until the last static dimension. Thus, the weights allow for scaling the "contributions" of every round of the procedure to the final pool of observations. By scaling the "contributions" we can make the effects of any of the static factors on the differences between transition matrices (forest growth) of the different "forestry types" more or less pronounced. Note that the weight assigned to the first factors scales its own effect while the weight associated with the second factors scales the combined effect of the first and the second factor, the weight associated with the third factor – the combined effect of the three factors. Therefore, the order, in which the factor names appear in the "estiminput.txt", is from more to less influential. There are also some explanatory comments in the "efdmestim.r".

Input data

As input to the estimation procedure, you need a data set giving the state of the individual plots (or other units) at two different points of time, where the "state" is expressed in terms of classes according to definitions in "factors.txt". The data is produced by classifying raw data. Each line represents one plot. Two columns are needed for each variable of the dynamic state-space, so in our case "vol0" and "vol1" denote the volume class at the two time points and "stem0" and "stem1" the stem number class. Figure 11 shows a fragment of the "pairedata.txt" with the input data for the estimation procedure used in the Austrian example. All plots which had activities were excluded (since we are estimating "no management" probabilities).



elev	own	spec	vol0	vol1	stem0	stem1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	4	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	4	1	1	1	1	1
1	3	1	1	1	1	1
1	4	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

Figure 11. "pairedata.txt" (a fragment)

10. Generating outputs

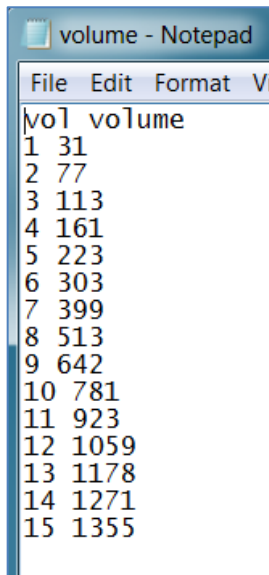
Files and functions involved:

- outputrequests.txt
- volume.txt
- drain.txt
- rawoutput.txt
- resultstates.txt
- efdmoutput.r
 - o outputcalc(...)

The current version of EFDM includes functionality for compiling outputs per simulation period for classes based on “non-changing” variables. The outputs (provided that the dynamic state-space is based on stem number and volume classes) can be anything that can be derived from either standing volume/stem number (the “stock” type) or harvested/lost in calamity volume/stem number (the “drain” type). For that, it is only necessary to supply the program with coefficients for converting area to volume/stem number (or their derivatives’) values. The conversion coefficients are multiplied with the columns in either “resultstates.txt” (“stock” type) or “rawoutput.txt” (“drain” type) produced by the model in the course of the simulation.

The “stock” type outputs show the state before every simulation period, while the “drain” type outputs show the flow under every simulation period.

For outputs of the “stock” type, such as volume, biomass, carbon etc. the column in the file must have the same name as the output variable, e.g. “biomass”. Figure 12 shows a snapshot of the “volume.txt” file used in the Austrian example. The coefficients for volume are calculated as $c_i = (vcl_i - vcl_{i-1})/2 + vcl_{i-1}$ where c_i is the coefficient for a class i , vcl_i is the (upper) limit for the class. For the highest volume class whose upper limit is “open” the coefficient can be some $1.20 * vcl_{i-1}$. The choice of the constant depends on where the actual class limits are in relation to the observed range of the volumes.



vol	volume
1	31
2	77
3	113
4	161
5	223
6	303
7	399
8	513
9	642
10	781
11	923
12	1059
13	1178
14	1271
15	1355

Figure 12. “volume.txt”

For outputs of the “drain” type, such as harvested total volume, timber or pulpwood volumes, the columns in the coefficient file must be named using the “activity suffixes”. Figure 13 shows the “drain.txt” used in the Austrian study with the column names “drain.thHigh”, “drain.thLow”, “drain.ffel”, “drain.cal” and “drain.nomgmt”. Note that for no-management and for calamities coefficients are zero. The coefficient for calamities is zero in this file because there would a separate files for calamities in order to keep track of those volume separately. For final felling, the coefficients are the same as in “volume.txt” since all volume is removed. For thinning, the coefficients depend on the volume class limits and on the definition of thinning – how many volume classes should a thinned area drop due to the removed stock. In our example, “thLow” was defined as a drop of one volume class and “thHigh” as a drop of two classes. Therefore, the coefficients for drain in “thLow” are calculated as $cth_i = c_i - c_{i-1}$ and in “thHigh” as $cth_i = c_i - c_{i-2}$ where c_i , c_{i-1} , and c_{i-2} are the class means and thus equal to the respective coefficients for final felling.


```

drain - Notepad
File Edit Format View Help
"vol" "drain.thHigh" "drain.thLow" "drain.ffe1" "drain.cal" "drain.nomgmt"
"1" 1 0 0 31 0 0
"2" 2 0 46 77 0 0
"3" 3 82 36 113 0 0
"4" 4 84 48 161 0 0
"5" 5 110 62 223 0 0
"6" 6 142 80 303 0 0
"7" 7 176 96 399 0 0
"8" 8 210 114 513 0 0
"9" 9 243 129 642 0 0
"10" 10 268 139 781 0 0
"11" 11 281 142 923 0 0
"12" 12 278 136 1059 0 0
"13" 13 255 119 1178 0 0
"14" 14 212 93 1271 0 0
"15" 15 177 84 1355 0 0

```

Figure 13. "drain.txt"

The desired outputs are specified in a control file "outputrequests.txt" (Figure 14). Every odd row specifies one output item to be compiled. The outputs can be calculated as total or as per hectare values, which is steered by the first word in the line ("TOTAL" or "PERHA"). The classes for which outputs are calculated can be defined by a single or combinations of the non-changing factors, for example species, elevation class or species and elevation class. Note that it is not possible to calculate any output without specifying at least one "by" variable. Every even row in "outputrequests.txt" specifies the name of the file with the coefficients necessary for the calculation.

```

outputrequests - Notepad
File Edit Format View Help
TOTAL drain by elev
drain.txt
PERHA drain by elev
drain.txt
TOTAL volume by elev
volume.txt
PERHA volume by elev
volume.txt

```

Figure 14. "outputrequests.txt"

In figure 14, four output items are requested. Note that the number of output items is not limited to four. There could be another pair of rows requesting drain or volumes for a different combination of non-changing variables, for example, drain by species elevation class. The entries in the file must be separated by spaces. The last row in the file must end with a "carriage return" (i.e. press "enter" button after typing the last character in the last line).

The outputs are written to files named according to output requests, e.g. “PERHA_volume_by_own_by_elev.txt”. Diagrams are generated as well. This is done by the functions `outputcalc(...)` and `efdmplot(...)` in `efdmoutput.r`. The functions are activated internally during the model run.

For other possible outputs, such as volume of different roundwood assortments, biomass or carbon storage respective coefficient files are needed. Note that for every single output variable, there must be a separate coefficient file. So, to obtain timber and pulpwood volumes, you need one file with coefficients for timber and another file with coefficients for pulpwood. The results are then compiled in separate output files and illustrated by separate diagrams. To merge the diagrams or join the results in one table, you need to resort to the general functionality of R or other software. Note also that the coefficients need to be defined so that the conversion takes place directly from volume classes (rather than m^3 !) to the desired output variable. This means that when preparing the coefficient files you will need to do some calculations.

11. Annex 1: checklists for correct spelling of variables' names

A misspelled column name in any of the input files will cause an error. As the number of instances when a variable or an activity name need to be recorded is rather large so is the risk for misspellings. As EFDM currently does not include any functionality for checking the consistency of the column names across the input files, great care is required when preparing the model inputs. The lists below can be used as check-lists when preparing model inputs or searching for an error when it has already occurred.

Activity names appear in

- activities file (as inputs)
- activity probabilities file (as column names)
- coefficients file(s) for drain (in column names)

Dynamic state-space dimension names appear in

- initial state file (as column names)
- factors file (as inputs)
- as an argument into the function `pre.estimate(...)`
- activity probabilities files (as column names)
- activities file (as inputs)
- coefficients file(s) for drain as column names (only volume; as column name)
- coefficients file for standing volume (only volume; as column name)

Other (non-changing) dimension names appear in

- initial state file (as column names)
- in factors file (as inputs)
- in activity probabilities file (as column names)
- in estimation function control file (as inputs)
- coefficients file(s) for drain as column names (might and might not; as column name)
- coefficients file for standing volume (might and might not; as column name)

12. Annex 2: a way to model regeneration in EFDM

When final felling is applied, area moves to a state-space cell which we consider bare land. It could be the “youngest” cell in the state-space, from which the area starts to “travel” through the state-space according to the no-management transition probabilities. However, it might be that in this way you are underestimating the time in reality needed for forest regeneration. One way to deal with this problem is to introduce a special class for bare land (“0-0” further in the text). The “speed” of regeneration can then be modified by how “fast” the final felled areas pass through this class.

For example, in the Swedish case study on even-aged forests (Specific Contract 14), we defined the transition probabilities after final felling (ffelP.txt) in such manner that 80 % of final felled area lands in the 0-0 class and the remaining 20% lands at once in the 1-1 (volume-age) class. At the next time step, all-area from 0-0 class moves to 1-1 class (i.e. “transition probability” 100%). This means that 20 % of final felled area is regenerated within 5 years and 80 % is regenerated within 10 years, counting from the year of final felling, which is realistic. Note that the transition 0-0 to 1-1 was not estimated from “pairedata” and is thus fully “user defined”. The transitions from 1-1 and further take place according to the probabilities estimated from “pairedata”. “Regenerated” means that a, by silvicultural standards, sufficient number of tree seedlings/ saplings is found growing on the site so that the site can be assigned stand mean age and total volume values.

In order to implement such a solution, you need, among other things, to make sure that the transition probabilities from “0-0” in the no-management transition probability file are what you want them to be. In the Swedish case study it was done in the following way. Since, the “pairedata” contained no pairs involving the class “0-0”, the probability values for transitions from “0-0” were fully determined by the priors file for the estimation procedure. Thus, in the priors file cell corresponding to the transition from “0-0” to “1-1” I entered “1” while making sure that transitions from “0-0” to any other classes were set to zeros. The rest of the priors file remained unchanged, as written by the *neutralprior()* function. Then the *pre.estimate()* function was used in the usual way to write a no-management probability file.

If the 10 year time span is not sufficient for realistic description of regeneration, there are at least two options of how to extend it. One way is to set the transition probability from 0-0 class to 1-1 class to less than 100%, the rest remaining in 0-0. (However, it means that this class will never be completely “emptied” (even if final fellings are stopped). Another way is to introduce one more bare land age class (let us call it “0A”). Then there will be two classes for bare land: “0-0” and “0-0A”. Having two bare land classes would allow for regenerating within 5, within 10 and within 15 years from final felling.

The current version of the code is not suitable for defining regeneration differently for different “forestry types” (unless you set up and run three “clones” of the model separately), because one and the same final felling transition probability matrix (file) is used for all “forestry types”.

References

1. Packalen, T., Sallnäs, O., Sirkiä, S., Korhonen, K., Salminen, O., Vidal, C., Robert, N., Colin, A., Belouard, T., Schadauer, K., Berger, A., Rego, F., Louro, G., Camia, A., Räty, M., San-Miguel, J. 2014. The European Forestry Dynamics Model: Concept, design and results of first case studies. Publications Office of the European Union, EUR 27004 doi10.2788/153990
2. Sallnäs, O., Berger, A., Räty, M., and Trubins, R. (2015) An Area-Based Matrix Model for Uneven-Aged Forests, Forests (accepted for publication).