Renats Trubins

Ola Sallnas

2014-09-01

Version 1

# Guidelines for setting up the EFDM-model for a national setting

Contents

General structure and functioning 1  
Recommendations for defining factors (classes) 3  
State space parameters 4  
Initial state 5  
Activity probabilities 6  
Transition probabilities 6  
Transition probabilities for final felling areas 7  
Transition probabilities for thinning areas 8  
Transition probabilities for no-management areas 9  
Output generation 11

## General structure and functioning

The EFDM is built around a basic matrix structure defining a number of fixed states, between which “units” of forest move over time. The core of the matrix structure is, let us call it, the dynamic state-space, which is defined by two variables: volume-class and age-class. We can, and should, set-up different dynamic state-spaces for different forests, as illustrated by figure 1. Here we have created a “pack” of state-spaces together constituting the total state-space. The “dynamic state space is as noted before defined by the variables or dimensions age-class and volume-class, while the different “forestry types” could be defined by the modeler using variables as site quality, geographical region, species (group), owner type etc. All dynamics in the model are taking place inside the different “dynamic state-spaces” – no “natural” movements between the forestry types are intended. But an area can move from one forestry type to another, for example, as a result of forest regeneration with a different tree species.

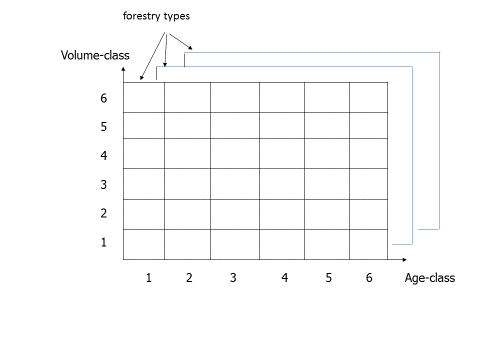


Figure1. Matrix structure of the EFDM

At every simulation step, the distribution of the forest area in the state space changes driven by the transition probabilities. Figure 2 shows how the transition is realized within one time step in EFDM. First, the areas in each cell of the state-space are split according to the activity probabilities. Essentially, it means splitting the state-space into three, one for each activity. In the next step, the distributions of the areas in the activity-specific state-space are modified according to transition probabilities for that activity. At last the activity-specific state-spaces are merged back into one. The resulting distribution of forest area in the state-space describes the state of the forest after one time step.

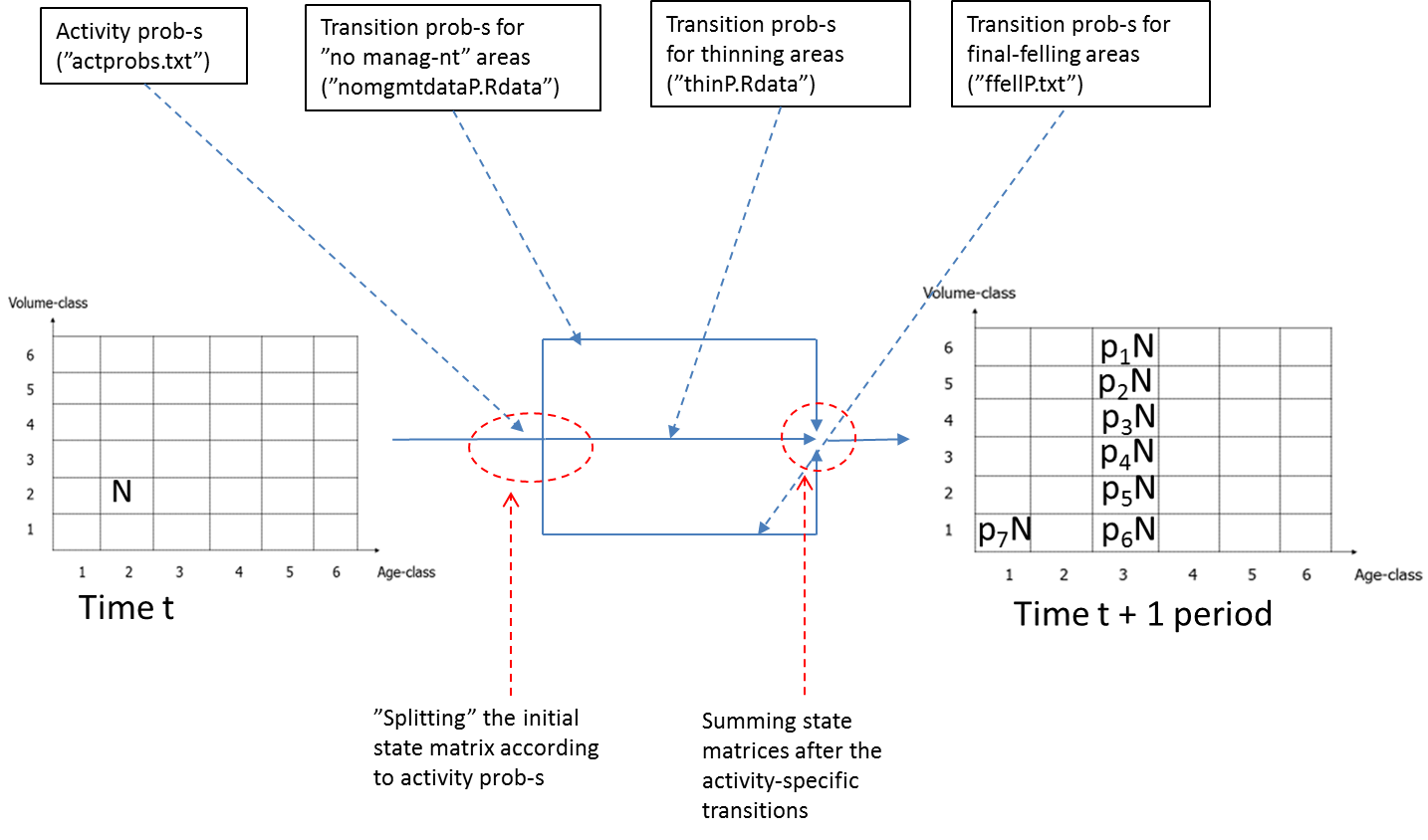


Figure 2. One time step in EFDM. In this illustration, the initial state only contains N ha in one single age-volume class. The “p’s” in the state at t+1 stand for the proportions of N falling into the different age-volume classes. Transition probabilities used in connection to specific activities (no management, thin, fell) are in fact also proportions like that. Note that the transitions are constrained to the age-class following the initial age-class (simulation time steps and age classes are equal in length) and the age class “1” which stands for bare land (in the case of final felling).

## Recommendations for defining factors (classes)

Vol-class and age-class should be used to form the dynamic state space while the choice of the other factors is up to the modeler. Generally speaking you could say that it makes sense to separate between different “forestry types” if they can be expected to either growth differently or to be managed differently. The number of “forestry types” could from a technical view point be large, but as we will see later, the larger the matrix - the poorer the estimate of the parameters of the model.

The factor “age” should preferably be set up so that the width of the age-classes coincides with the time between the two “observations’ of the NFI-plots used for estimating the model. In that way we do not need to estimate “age-transitions”. The number of age-classes should be big enough to let the state-space span all forests existing today and those which can occur during simulations.

The factor “vol” demands additional considerations. According to our experience the number of volume-class could very well be 10-15. There is not much use in increasing the number much over that, but we must have a total volume span in the models that implies that very seldom forests grow and hit the “roof” of the volume dimension. Probably it is also good if the probability of growing “out” of a state is of the same magnitude for young and older forests, which means that we should define our volume-classes so that they reflect the growth pattern of a forest.

In a Swedish case study this was carried out using a general Chapman-Richards growth function as support. In figure 3 the procedure is outlined. The growth function basically distributes the total volume reached at relative age=100 (which is defined to be the time for MAI culmination) over relative age. By splitting the time span in terms of relative age 0-100 in the number of volume classes decided on, you get from the curve the corresponding volumes in terms of relative production. These can be used to define the volume classes.

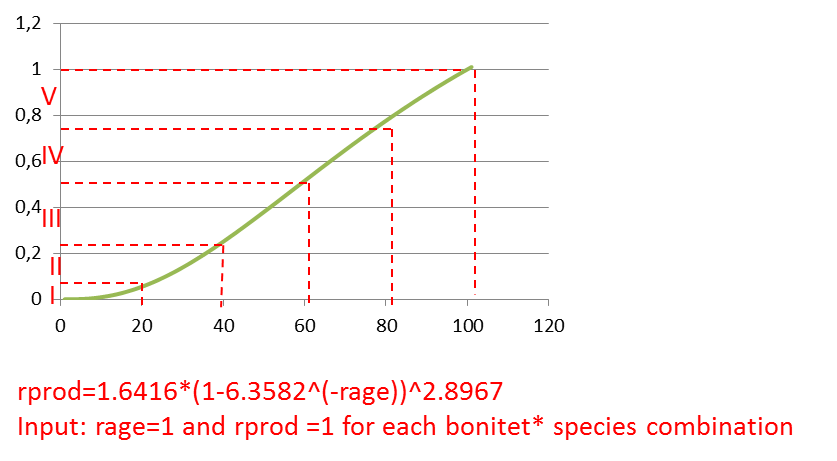


Figure 3. Using a growth function to define volume class limits

### Additional considerations regarding volume class width

When the volume classes are defined according to the growth pattern, the plots will be more concentrated in and close to the “diagonal” cells of the state-space. In principle, it means that the number of volume classes should be equal to the number of time periods (age classes) that it takes for MAI to reach its maximum. Of course, this case is theoretic – it would only occur if the growth function used for defining class limits was 100 % true. In reality, there will anyway be plots of the same age class but belonging to different volume classes. Even more so since we are using same number of volume classes for all site qualities and species.

Anyway, a “broad” volume class (i.e. wider that the actual one-period growth) implies that the plots will be distributed over a several age-volume classes instead of one age volume class (v1a1 … v1an instead of v1a1). “Stretching” volume classes over several age classes is not good for several reasons. It can lead to some problems in the following model specification and reduce the information that the model would be otherwise able to “carry” in the analyses. Following disadvantages of “broad” volume classes can be named:

1. The model might be less sensitive to modifications of final felling age. Changing age class of final felling might have no effect on the harvested volume because it still will be the mean of the same volume class.
2. The above applies also to thinning timings
3. It might complicate the definition of the thinning. That means even one-class drop might be too large for a realistic thinning representation

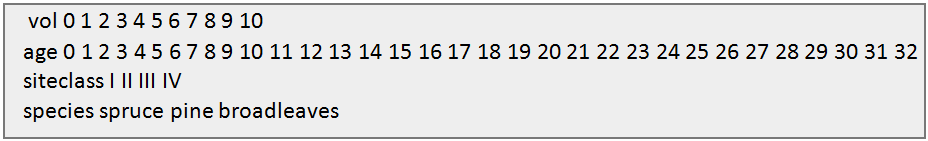
Narrowing volume classes below one-time-step-growth does not seem to have other disadvantages than reduction of the number of plots per class (because of more classes) available for the estimation of the transition probabilities which is, of course, a very serious limitation.

## State space parameters

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 4 is an example of what such a file could look like. In the Finnish example we are working with 11 volume classes, “named’ 0-10, and 33 age classes, named 0-32. Together these two form the dynamic state space having 11\*32 = 352 different states. The two variables “siteclass” and “species” together define 12 different forestry types.

  
Figure 4. “factors.txt”

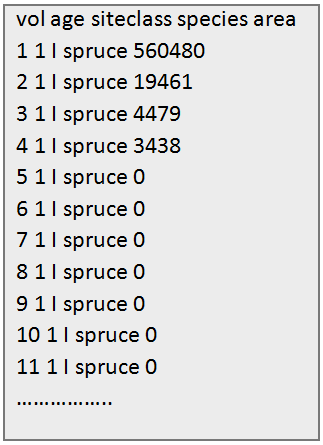
It should however be noted that in “factors.txt” you only give the number and the “names” of the factor levels. The definitions – for example the limits between the volume classes – must be decided when classifying the initial raw data.

## Initial state

Files and functions involved:

* initstate.txt

This file contains a description of the initial state in the forest. It should contain one record per class in the state-space and give the area that is found in that class. When a simulation is initialized this data is used to fill the state-space with forest area.

  
Figure 5. “initstate.txt”

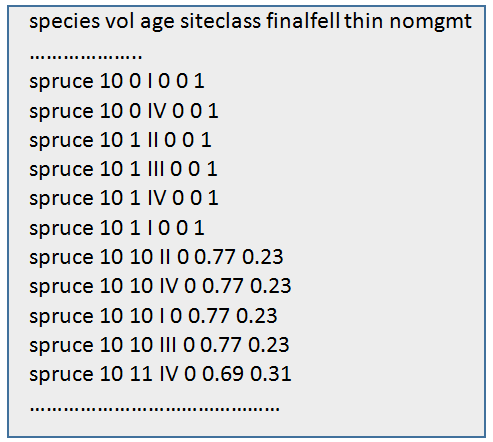
## Activities

Files and functions involved:

* actprobs.txt

Activity probabilities are essentially proportions of the area in a state-space cell that must be subject to the respective activities within a given simulation period. In the course of simulation, the program will create activity-specific state spaces and split the areas according to the proportions. There should be one line per “forestry type”\*state-space cell and that the probabilities on one line should sum to 1. Figure 6 illustrates how the “actprobs.txt” file could look like.

By means of activity probabilities we define forest management. We can try to define realistic management based on expert-knowledge and inventory records. Note, that EFDM package presently does not contain any functionality for estimating activity probabilities from inventory data.

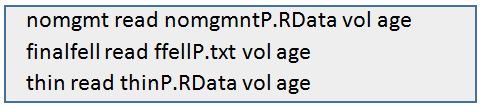
  
Figure 6. “actprobs.txt”

## Transition probabilities

Files and functions involved:

* activities.txt

The transition probabilities for forest areas subject to different activities such as “no-management”, thinning and final felling can be either read from files or estimated (automatically) at the start of every simulation. In the former case, the estimation has to be done beforehand and the results saved in a file. In the latter case, data must be accessible for the estimation function. The words “read” or “estimate” in the “activities.txt” specify whether the program has to do reading or to run the estimation function (figure 7). Note, that in the current version, estimation “on the fly” is only possible for “no-management”. Further, depending on the “keyword”, either the name of the file to be read or the name of the input file for the estimation function needs to be specified. At last the names of the “changing” factors need to be specified. The order must be “vol” “age”. The entries should be separated by spaces.

  
Figure 7. “activities.txt”

Technically, the matrices for different forestry types are “packed” into multidimensional arrays. The probability matrices (or arrays) for the different activities must be, however, separate. The matrices must have the dimensions of (nvol\*nage ) x (nvol\*nage ) where nvol and nage are the numbers of volume and age classes respectively. The matrix is made of “nvol\*nvol” blocks arranged in a “nage \*nage” matrix.

## Transition probabilities for final felling areas

Files and functions involved:

* ffellP.txt

These “probabilities” produce only one type of transition – after final felling the whole area always goes to age-volume class 1-1 (see figure 8). Therefore no statistical estimation is needed. The file needs to be written in accordance with the dimensions of the state space. EFDM package at this stage does not offer any function for producing this file but rather the users need to prepare this file on their own. Since these probabilities will always be read from a file, the “keyword” “read” must not be changed for final felling in the “activities.txt” file.



Figure 8. A compressed view of a transition probability matrix for final felling areas. The symbols a1 to a33 and v1 to v10 denote age and volume classes. The matrix is made of nvol\*nvol blocks arranged on nage \*nage matrix where nvol and nage are the numbers of volume and age classes respectively.

## Transition probabilities for thinning areas

Files and functions involved:

* testthinP.RData
* hackfunctions.r
  + makethinP(...)

At this stage of development of EFDM, a file containing the probabilities for thinning areas (like “testthinP.RData”) must be produced before running the model. Such file can be produced using “makethinP(..)” function from “hackfunctions.r” This means also that the “keyword” “read” must not be changed for thinnings in the “activities.txt” file.

In EFDM, we use an assumption that after partial wood volume removal, the forest grows in the same manner as any other forest in the same volume-age class. Eventual thinning effects on subsequent growth are not modelled.

The arguments for makethinP(…) are “inputfilename”, “voldrops” and “resultfilename”. “Inputfilename” refers to the file containing no-management transition probabilities (“nomgmtdataP.Rdata”). EFDM users need to produce such file based on their own data with the help of the pre.estimate(…) function described in one of the next sections. “Voldrops” refers to a vector which specifies how many volume classes to drops to reflect the direct effect of thinning (just the volume decrease, no effect on growth as such). The length of the vector corresponds to the number of volume classes. “Resultfilename” must be consistent with the filename specified in the “activities.txt”. You should use the extension “.RData”.

## Transition probabilities for “no management” areas

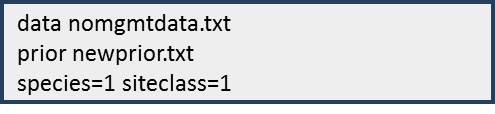
Files and functions involved:

* nomgmtdataP.Rdata
* estiminput.txt
* nomgmtdata.txt
* newprior.txt
* hackfunctions.r
  + neutralprior(…)
  + pre.estimate(…)

Transition probabilities for “no management” areas can be read from a file (like “nomgmtdataP.Rdata”) or estimated automatically in the course of every simulation. In the latter case, EFDM calls the function “estimatetransprobs (…)” (from “efdmestim.r” file) to conduct the estimation. In this case 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 produced beforehand and written to a file. This can be done using the “pre.estimate(…)” function from “hackfunctions.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. Whether the probabilities should be estimated in the course of the simulation or read from a file must be specified in “activities.txt”.

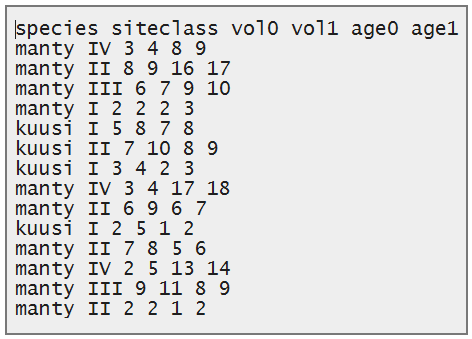
### Setting parameters for the estimation procedure

The file “estiminput.txt” (figure 9) is a control file used by the functions “estimatetransprobs (…)” and “pre.estimate (…)”.The first line contains the name of the file with the data for estimation (“nomgmtdata.txt”). The second line contains the name of the file with “priors” (like “newprior.txt”) needed for the estimation procedure. If no priors can be produced the filename can be replaced with the string “uninformative”. A file with simple priors can be produced using the function “neutralprior(…)” from “hackfunctions.r”. 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 “prior weights” used in the estimation procedure. Their use will be described elsewhere.

  
Figure 9. “estiminput.txt”

### Input data

As input to the estimation procedure is needed a data set giving the state of 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”. Figure 10 shows a fragment of the “nomgmtdata.txt” with the input data for the estimation procedure used in the Finnish example. This data is produced by classifying raw data.

  
Figure 10. “nomgmtdata.txt” (a fragment)

In the Swedish case, the raw data did not include volume at two points in time but rather volume and the five-year increment. The five-years-ago volume was therefore derived by subtracting the increment from the present volume. All plots which had activities were excluded (since we are estimating “no management” probabilities). The plots were then classified according to the chosen class definitions at times “0” and “-5”. Figure 11 summarizes the raw data description and the estimation procedure in the Swedish case.

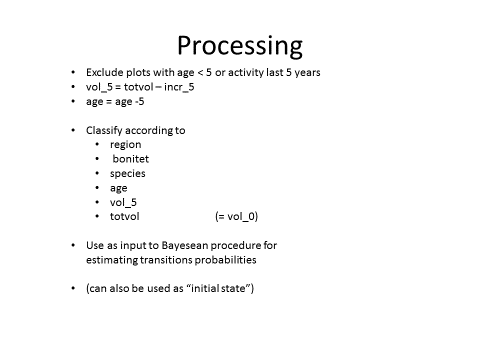
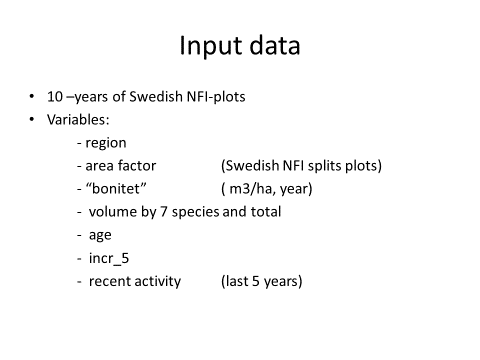


Figure 11. Estimation of “no management” transition probabilities in the Swedish case study.

To produce a file with no-management transition probabilities the function “pre.estimate (…)” (in hackfunctions.r) can be used. A file with no-management probabilities is needed, for example, when you use the makeathinP(…) function (also in hackfunctions.r) to compute the transition probabilities for thinning areas.

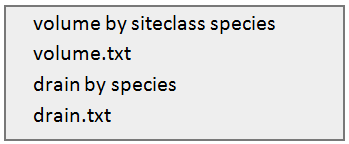
## Output generation

Files and functions involved:

### outputrequests.txt

* volume.txt
* drain.txt
* rawoutput.txt
* resultstates.txt
* efdmoutput.r
  + outputcalc(…)
  + efdmplot(…)

Current version of EFDM includes functionality for compiling outputs such as standing volume and drain for classes based on “non-changing” variables. The volume-results show the volume before every simulation period, while the drain-results show the drain ­under every simulation period. Classes can be defined by a single or combinations of the non-changing variables, for example, species, site index or species and site index. The users can specify what output should be compiled in file “outputrequests.txt”. Figure 12 shows how the contents of the file “outputrequests.txt” could look like. Every odd row specifies one output item that should be compiled. Every following even row specifies the name of the file where the coefficients necessary for the calculation should be taken. There are only two coefficient files, “volume.txt” and “drain.txt”. The outputs are not limited to two items. There could be another pair of rows requesting, for example, “drain by species siteclass” (followed by “drain.txt” in the next row). The entries in the file must be separated by spaces. The outputs are written to files with corresponding names. Plots are also produced. This is done by the functions outputcalc(…) and efdmplot(…) in efdmoutput.r. The functions are activated internally during the model run.

  
Figure 12. “outputrequests.txt”

The files “volume.txt” and “drain.txt” contain the coefficients for converting volume classes to m3. Figure 13 shows a portion of the “volume.txt” file used in the Finnish example. The coefficients are the means of the volume classes calculated as *ci = (vcli – vcli-1)/2* where *ci* is the coefficient for a class *i*, *vcl* is the (upper) limit for the class*.* For the highest volume class whose upper limit is “open” the coefficient can be some 1.20\* *vcli-1.*  The choice of the constant depends on where the actual class limits are in relation to the observed range of the volumes.

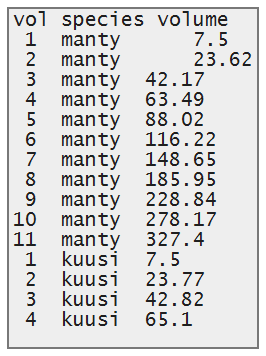
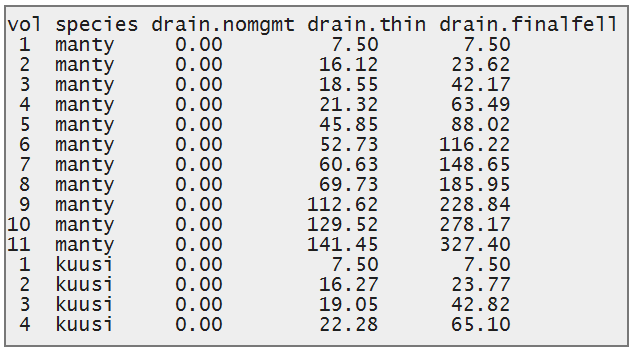
  
Figure 13. “volume.txt”

Figure 14 shows a portion of the “drain.txt” file used in the Finnish example. Here the coefficients are given activity-wise. Note, however, that for no-management, the coefficients are zeros since obviously there is no drain. 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 before growing again. Recall, that this was specified as the “voldrops” argument the the makethinp(…) function. In the Finnish example, thinning implied a drop of one volume class. Therefore, the coefficients for drain in thinning arecalculated as *cthi= ci – ci-1* where *ci* and  *ci-1* are the class means and thus equal to the respective coefficients for final felling.

  
Figure 14. “drain.txt”

It is, of course, possible to extract more outputs than only those named above. For example, drain from separately thinning and final felling, age or volume class distribution – all that can be derived from files “rawoutput.txt” and “resultstates.txt” using R or other suitable software. It is just that there are as yet no ready functions for that in the EFDM package.