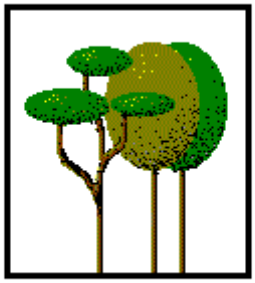
User's Manual for NOTHOPACK:

A Growth and Yield Simulator for Nothofagus Second Growth Forests



Prepared by

Salvador A. Gezan, Paulo C. Moreno, Sebastian Palmas and Alicia Ortega

April 24, 2021

User's Manual for NOTHOPACK: A Growth and Yield Simulator for Nothofagus Second Growth Forests

Bibliographical Reference

Gezan, S.A., Moreno, P.C., Palmas, S., and A. Ortega. 2021. NOTHOPACK: A Growth and Yield Simulator for Nothofagus Second Growth Forests. Version 1.0. Universidad Austral de Chile, Valdivia, Chile.

Authors Email Addresses

Salvador A. Gezan forestats.sg@gmail.com
Paulo C. Moreno paulo.moreno@ciep.cl
Sebastian Palmas palmasforest@gmail.com
Alicia Ortega aortega@uach.cl

Companion Resources

This R library and associated documentation can be accessed at:
https://github.com/sgezan

Acknowledgements

Initial funding for this research come from FONDEF (Chile), grant D97I1065. P. Moreno and S. Palmas were funded by scholarships and grants from the School of Forest Resources and Conservation, IFAS, University of Florida, USA. Several institutions collaborated with data, time, and resources to the success of this project, including: INFOR (Hans Grosse, Luis Barrales), Universidad Austral (Pablo Donoso, Cristian Estades), and personal from CONAF, Inversines Cranefield Chile Ltda., Forestal Cholguan S.A., Forestal Mininco S.A., Forestal Tornagaleones S.A. and Forestal Valdivia S.A., all companies that supported use from 1997 to 2000.

The authors greatly acknowledge the team of students, technicians and faculty that helped on the establishment and measurement of the majority of the plots used in this study, including Marcelo Farias, Ernesto Venegas, Soledad Munoz, Christian Salas-Eljatib, to name just a few of our helpers. A critical consultant for this project was Dr. James Fleweeling. In addition, we want to thank our current institutions VSN International (United Kingdom), CIEP (Chile), and Universidad Austral (Chile) for allowing us to dedicate time to this exciting package.

Contents

Chapter 1

Introduction to NOTHOPACK

1.1 Overview

NOTHOPACK is a growth model (G&Y) simulator for second-growth forests of *Nothofagus obliqua* (roble), *N. alpina* (raulí), and *N. dombeyi* (coihue), which are among the most important native mixed forests in Chile. They form the RORACO forest type that is present approximately between the 36° and 42° S latitudes in both the Chilean Andes and the coastal mountain range with some fragments in Argentina. At the present, the RORACO forest type covers 1.96 million hectares, around 10% of the native forested area of Chile.

This simulator is the product of a large FONDEF project (D97I1065) that started in 1996 at the Universidad Austral under the direction of Dr. Alicia Ortega, financed by the Chilean Government. Under this 3-year project, several efforts were done to collect data for RORACO from previously studies. However, the largest effort was the development of two stratified sampling networks with temporary and permanent plots established across the complete range of this forest type in Chile. Efforts to build this simulator have continued over the years mainly at the Universidad Austral (Chile) and at the University of Florida (United States). Hence, NOTHOPACK is the result of combining all these resources and efforts into a single product, that has been put togehter into an R library to be used by foresters, and researchers to facilitate and improve the management and sustainability of this important Chilean forest type.

This G&Y model has several properties, but the main ones are:

- Inventory processing
- Whole-stand level simulation
- Individual-tree level simulation
- Compatibility (stand-tree) simulation
- Thinning simulation

In this manual we present some of the structure of NOTHOPACK and we illustrate its use with several examples.

NOTHOPACK is a tool that is provided as it is. We have done many efforts to use the datasets available as much as possible, and on cases with limited information, we have considered some simplifying assumptions. However, NOTHOPACK will not necessarely be free of errors or, in some cases, inconsistencies. We believe this G&Y should be *dynamic* and improved as more information becomes available (particularly remeasurement of permanent plots), and as our understanding of this natural resources improves. For this reason, we consider NOTHOPACK as a research tool, and we are happy to recieve your comments and contributions to make this G&Y better and more useful to all users; hence, feel free to contact any of the author with your suggestions and recommendations.

1.2 Getting Started

NOTHOPACK is an R package that can be downloaded from https://github.com/sgezan/Nothopack. In order to install NOTHOPACK you need:

```
install.packages("devtools")
devtools::install_github("sgezan/Nothopack")
```

Alternatively, if you have the zip file associated with this package, you can install it by using the command:

```
install.packages(path, repos = NULL, type = "source")
```

where `path` is the location and name of the ZIP file to install, for example: `'C:/Desktop/Nothopack_1.0.0.zip'`.

Anohter option is to install NOTHOPACK directly from RStudio by first going to the menu: `>Tools/Install Packages...`, in the `Install From` select `'Package Archive File (.zip; tar.gz)'`, and then you will have to search for the location of the file on your computer and select it. Finally, you just need to click on `Install`.

Once installed, to load this library in R you run the command:

```
library(Nothopack)
```

Now you are ready to use it! A good way to get started is to look at the help directly from this library, you can type for example:

```
help(core_module)
```

In the next sections we will present how to run some of your analyses for an array of different examples and conditions. In these cases, we will use the console in RStudio to perform all analyses.

Chapter 2

Reading a Stand or a Tree-list: `input_module`

One of the most important modules from NOTHOPACK is `input_module()`, which process input information required for all current or future stand- or tree-level calculations, or their simulations. NOTHOPACK accepts as input information from both stand- or tree-level. This function is always the first step to start performing your simulations, but also you can use it to complete missing information for a given plot. For example, by providing age and dominant height, you will be supplied with the corresponding site index.

The module `input_module()` generates a new object with several elements that are critical input to be used for downstream simulations or calculations in other modules. However, it can be used on itself as it perform required calculations that help to have a full numerical description of the stand under evaluation.

2.1 Stand-level Information

The basic code to process stand-level information for `input_module()` is:

```
input_module(ZONE, AD, AF, HD, SI, N, BA, type = "stand")
```

This function requires several inputs that specifie charactersitics of the stand. The first element, `ZONE` identifies the corresponding growth zone of the plot for the RORACO forest type. This map can be found in Figure 1. These growth zones were defined according to edaphoclimatic variables together with the actual distribution of this forest type.

`AD` corresponds to the dominant age (in years) of the stand. This is defined as the age of the dominant trees measured at breast height. Note that this definition is considered as it will close to the age of establishment of these stands after a disruption. `AF` is the final dominant age (in years) for the requested simulation. You can also supply site index, `SI` (m) or dominant height, `HD` (m). These are calculated based on the height of 100 tallest trees in a stand. `N` and `BA` are vectors of tree density (trees/ha) and basal are (m^2/ha) of the stand,

respectively, where the order of the cohorts are: Rauli, Roble, Coigue and other species.

Finally, we indicate the type of input information (and also simulation) requested, which in this case corresponds to `'stand'`, as this is the level of data we are providing in `N` and `BA`. There are other input arguments, but we will discuss those later as they are needed for other operations.

In order to illustrate this function, we will consider stand-level information for the stand presented in table below.

Table 2.1: Plot information for a stand dominated by Rauli located in growth zone 1, that is 28 years old (dominant age) and with a dominant height of 28.0 m.

Code	Cohort	BA	N
1	Rauli	36.5	464
2	Roble	2.8	23
3	Coigue	1.6	16
4	Other	2.4	48

The above plot data can be read by NOTHOPACK as:

```
BA <- c(36.5, 2.8, 1.6, 2.4)
N <- c(464, 23, 16, 48)
plot.inf <- input_module(ZONE = 1, AD = 28, AF = 28, HD = 18.5,
    N = N, BA = BA, type = "stand")
```

Note that in the above case, we did not specified `SI`. After running `input_module()`, the object `plot.inf` has a list of elements of interest, which can be called with the command:

```
ls(plot.inf)
```

```
##  [1] "AD"          "AF"          "area"        "ATHIN"
##  [5] "BARp"        "comptype"    "data.sim"    "ddiam"
##  [9] "DDist"       "DOM.SP"      "FT.thin"     "HD"
## [13] "IADBH_model" "NHA_model"   "PBAN"        "PNHAN"
## [17] "QD_ba"       "SDIP"        "SI"          "sp.table"
## [21] "T_model"     "tree.list"   "type"        "V_model"
## [25] "ZONE"
```

The function `input_module()` completes all required calculations (including stand-level volume estimation). Hence, this function can be considered as an inventory generator providing all relevant information for the year of interest based on those supplied stand-level parameters. For example, we can access to the stand table, by using:

```
plot.inf$sp.table
```

```
##   SPECIES   N   BA     QD    VTHA
## 1       1 464 36.5 31.648 252.147
## 2       2  23  2.8 39.370  19.343
## 3       3  16  1.6 35.682  11.053
## 4       4  48  2.4 25.231  16.579
## 5       0 551 43.3 31.632 299.122
```

In addition, we have the elements from `plot.inf$data.sim` that provide with the required summary to use later on the simulations. Note from here, that there are several stand parameters, some of the important ones are: `SI` = 13.48 and `SDIP` = 65.02, which corresponds to site index, obtained from the dominant age and height, and the stand-density index (%) based on the dominant specie, respectively. In addition, the dominant specie. `DOM.SP` is identified as 1 corresponding to Rauli.

Another important element from this list i the that has important summary information is the object `data.sim` that summarize all information of the stand, and this is the table from collecting the simulated data, that in this case corresponds to only `AGE = 28`, as there are no simulations carried out yet.

```
plot.inf$data.sim
```

```
##   AGE   HD NHA     QD   BA NHAN NHA99  BAN BA99    PBAN
## 1  28 18.5 551 31.632 43.3  503    48 40.9  2.4 0.94457
##     PNHAN    SI   SDIP   VTOT
## 1 0.91289 13.48 65.019 299.12
```

2.2 Generating a Diameter Distribution

The above stand-level plot data can be further extended to generate a diameter distribution. NOTHOPACK has incorporated equations based on the method of parameter recovery to generate these diameter distributions based on the work by Gezan and Moreno (200X). This function generates a diameter distribution for each of the *Nothofagus* cohorts, and is later used to calculate stand parameters for each of their diameter classes.

As an example, lets use the above code by adding `ddiam = TRUE` as shown below

```
BA <- c(36.5, 2.8, 1.6, 2.4)
N <- c(464, 23, 16, 48)
plot.inf <- input_module(ZONE = 1, AD = 28, HD = 18.5, AF = 28,
    N = N, BA = BA, type = "stand", ddiam = TRUE)
```

The diameter distribution for all cohorts is stored in `plot.inf$DDist`. For instance, the diameter distribution combined for all species can be printed using:

```
plot.inf$DDist[5, , ]
```

```
##    DBH_ll DBH_ul D_class H_class       N      BA       VT
## 1       5     10     7.5  3.5576  19.232 0.11008  0.09354
## 2      10     15    12.5  8.6628  30.061 0.42982  1.13454
## 3      15     20    17.5 12.1946  56.713 1.52117  5.88615
## 4      20     25    22.5 14.5828  86.577 3.78640 17.57876
## 5      25     30    27.5 16.1997 105.575 6.86413 35.27888
## 6      30     35    32.5 17.2943 103.655 9.39696 51.37775
## 7      35     40    37.5 18.0413  81.939 9.88879 56.24896
## 8      40     45    42.5 18.5594  51.936 8.06141 47.01127
## 9      45     50    47.5 19.1043  11.941 2.35457 13.44148
## 10     50     55    52.5 19.9028   2.898 0.73133  3.80970
## 11     55     60    57.5 19.7090   0.477 0.15535  0.79974
## 12     60     65    62.5  0.0000   0.000 0.00000  0.00000
## 13     65     70    67.5  0.0000   0.000 0.00000  0.00000
## 14     70     75    72.5  0.0000   0.000 0.00000  0.00000
## 15     75     80    77.5  0.0000   0.000 0.00000  0.00000
## 16     80     85    82.5  0.0000   0.000 0.00000  0.00000
## 17     85     90    87.5  0.0000   0.000 0.00000  0.00000
## 18     90     95    92.5  0.0000   0.000 0.00000  0.00000
```

This diameter distribution is presented by diameter classes of 5 cm, and you can see the average height, number of trees, basal area and total volume for each of the classes. It is important to indicate that the total volume in this table, based on the column `VT` corresponds to 232.66, which differs from the one reported earlier of 299.12, this occurs because for the diameter distribution volumes are calculated using the available taper equations, instead of the stand-level volume equations.

If there is interest to obtain the tables for other species, the number 5 from above can be replaced. The levels considered here are: $1: Rauli, 2: Roble, 3: Coigue, 4: Other, 5: All$.

2.3 Processing an Inventory Plot

One of the main practical uses of NOTHOPACK is to process a RORACO inventory plot. This is done using the same function `input_module()` and then allowing this module to perform all required calculations. However, in this case instead of using stand-level input we provide to this function inventory data as a complete tree-list, which is supplied to R as a `data.frame` with the following columns: *ID*: unique tree ID number, *SPECIES*: species code ($1: Rauli, 2: Roble, 3: Coigue, 4: Others$), *DBH*: diameter at breast height (in cm), *HT*: total tree height (in m) and *SS*: sociological status of each tree in the stand. The column for *HT* can have some missing values and these will be completed by fitting a height-diameter model to the available data. For *SS*, if they are missing they will be estimated proportional to the basal area of larger trees (BAL, m^2). Note that *SS* is considered as a continuous

variable based on the usual classes: 1 : *Dominant*, 2 : *Codominant*, 3 : *Intermediate*, 4 : *Surpressed*; therefore, larger values are associated with trees with increased crown competition.

As indicated, the inventory plot provided to NOTHOPACK with a tree-list will process the information to obtain all stand parameters and completes all missing information (*e.g.*, `SI`) based on tree- or stand-level data. In addition, there will be estimation of volume for each of the trees on the list, and a stand table will be provided with this summary. The idea is that this output can be used directly to plan forest management such as thinning or other opperations.

In addition, as part of NOTHOPACK the output from `input_module()` with tree-level data will be the main input for further tree-level simulations, if required.

The library NOTHOPACK provides with an example of a tree-list that we will use for illustration. This tree-list example is named `plot_example` and it contains 46 trees and from an inventory plot with an area of 500 m^2. The structure of this data frame and the first six rows are shown below.

```
str(plot_example)
```

```
## 'data.frame':    46 obs. of  6 variables:
##  $ ID     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ SPECIES: int  2 2 2 2 2 2 2 2 2 2 ...
##  $ DBH    : num  24 23.6 35.5 15.9 13.5 ...
##  $ HT     : num  NA NA 17.5 8.64 NA NA 15 NA NA NA ...
##  $ SS     : logi  NA NA NA NA NA NA ...
##  $ FT     : logi  NA NA NA NA NA NA ...
```

```
head(plot_example)
```

```
##   ID SPECIES   DBH    HT SS FT
## 1  1       2 24.00    NA NA NA
## 2  2       2 23.60    NA NA NA
## 3  3       2 35.47 17.50 NA NA
## 4  4       2 15.90  8.64 NA NA
## 5  5       2 13.50    NA NA NA
## 6  6       2 23.60    NA NA NA
```

This tree-list can be entered into `input_module()` by specifying `'type = tree'` together with the specification of the data frame under `'tree.list = plot_example`. Note that when using a tree-level list to specify the stand, there is no need to specify the number of trees (`N`), basal area (`BA`) or the dominant height (`HD`) as these will be calculated from the tree list. In addition, calculations of the individual tree volume, together with a stand table and a diameter distribution will be automatically calculated and stored in the respective object.

```
tree_inv <- input_module(ZONE = 2, AD = 28, AF = 28, area = 500,
    type = "tree", tree.list = plot_example, T_model = 1)
```

```
## Warning: Some tree heights were estimated using model fitted
## with data (across all species)
```

```
## Warning: Verify that these are good estimates of total
## height.
```

```
## Warning: Sociological Status (SS) was completed for all (or
## some) trees in plot if missing.
```

```
## Warning: Dominant height (HD) missing, using the one
## obtained from the HT tree data
```

As seen in the above warning messages, this function also completes missing information from the inventory (total height and sociological status as seen here).

After running the above code, we have an updated tree-list table that is stored in `tree_inv$tree.list`. The first six rows are presented below

```
head(tree_inv$tree.list)
```

```
##   ID SPECIES   DBH     HT     SS FT  DBH0     VIND
## 1  1       2 24.00 16.827 3.1140 20 24.00 0.285632
## 2  2       2 23.60 16.739 3.1791 20 23.60 0.275382
## 3  3       2 35.47 17.500 1.4784 20 35.47 0.594898
## 4  4       2 15.90  8.640 3.8351 20 15.90 0.064188
## 5  5       2 13.50 13.203 3.9304 20 13.50 0.074701
## 6  6       2 23.60 16.739 3.2442 20 23.60 0.275382
```

There are a few additional columns added corresponding to *DBH0*, which is used to keep track of the individual tree diameter increments, and *VIND*, which is the current individual stem volumen for each tree.

The function `input_module()` will also automatically calculate summary statistics by species in a similar way as shown before when we used the stand-level input.

```
tree_inv$sp.table
```

```
##   SPECIES   N       BA      QD    VTHA
## 1       1  60  1.08580 15.1794   6.964
## 2       2 780 38.91528 25.2039 245.218
## 3       3   0  0.00000  0.0000   0.000
## 4       4  80  0.31314  7.0596   1.490
## 5       0 920 40.31422 23.6206 253.672
```

In addition, we can request the diameter distribution from the complete stand or by each species. Here the code is:

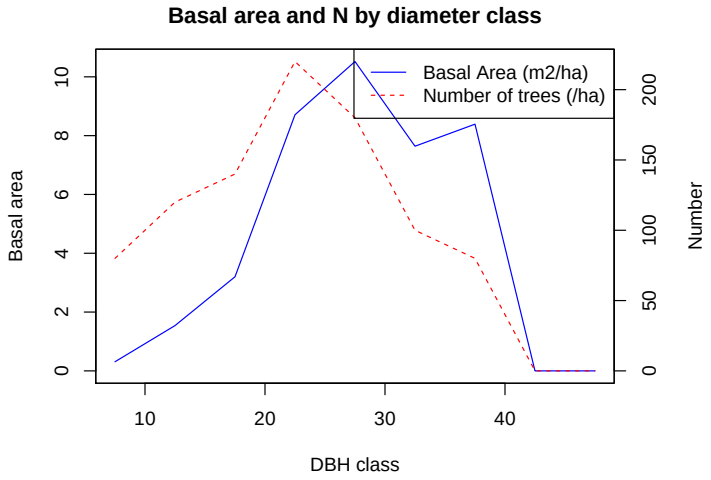
```
# Overall diameter distribution
tree_inv$DDist[5, , ]
# Diameter distribution for N. alpina (1)
tree_inv$DDist[1, , ]
```

For example, for the all trees, but only a few of the diameter classes this is:

```
tree_inv$DDist[5, 1:8, ]
```

```
##   DBH_ll DBH_ul D_class H_class   N    BA     VT
## 1      5     10     7.5    7.76  80  0.31  1.490
## 2     10     15    12.5   12.63 120  1.54  7.931
## 3     15     20    17.5   14.60 140  3.20 18.279
## 4     20     25    22.5   17.30 220  8.71 56.801
## 5     25     30    27.5   17.40 180 10.52 67.563
## 6     30     35    32.5   17.64 100  7.64 48.147
## 7     35     40    37.5   18.49  80  8.39 53.461
## 8     40     45    42.5    0.00   0  0.00  0.000
```

The above diameter distribution can be plotted; for example, by showing the basal area and number of trees, as presented in the figure below.



Finally, the stand object also stores additional stand summary information such as quadratic diameter (QD), basal area of *Nothofagus* (BAN), and site index (SI) in a table object. All this information will be used for reports or in downstream simulations.

```
tree_inv$data.sim
```

```
##   AGE     HD NHA     QD     BA NHAN NHA99    BAN    BA99
## 1  28 18.394 920 23.621 40.314  840    80 40.001 0.31314
##      PBAN   PNHAN    SI  SDIP   VTOT
## 1 0.99223 0.91304 13.11 91.37 253.67
```

Chapter 3

Running a Stand-level Simulation: core-module

In this section we will review a couple of examples on how to run a stand-level simulation. First we will show an example on how to run a simple stand-level simulation and in the second example, we will perform another stand-level simulation but this will be complemented with the generation of a diameter distribution.

3.1 Simple Stand-Level Simulation

Probably the most common use of NOTHOPACK is related to performing a simple stand-level simulation, in this case well consider a case without thinning, no generation of diameter distribution.

The following simulation function will simulate annual stand-level parameters from the speficied initial stand age `AD` to the final age `AF`, together with other critical information such as ZONE and HD (or SI) and the stand table from the conditions of interest. An example is shown below:

```
BA <- c(36.5, 2.8, 1.6, 2.4)
N <- c(464, 23, 16, 48)
stand.input <- input_module(ZONE = 2, AD = 28, HD = 18.5, AF = 33,
    N = N, BA = BA)
```

We started with the function `input_module()` that prepares the stand. We will first, review the simulation summary data with

```
stand.input$data.sim
```

```
##   AGE   HD NHA     QD   BA NHAN NHA99  BAN BA99    PBAN
## 1  28 18.5 551 31.632 43.3  503    48 40.9  2.4 0.94457
##     PNHAN   SI   SDIP   VTOT
## 1 0.91289 13.9 65.019 299.12
```

and the initial stand table by species is

```
stand.input$sp.table
```

```
##   SPECIES   N   BA     QD    VTHA
## 1       1 464 36.5 31.648 252.147
## 2       2  23  2.8 39.370  19.343
## 3       3  16  1.6 35.682  11.053
## 4       4  48  2.4 25.231  16.579
## 5       0 551 43.3 31.632 299.122
```

The function that runs the simulation is `core_module()`. As part of NOTHOPACK logic, all of the necessary information for the simulations is inside the stand object created by `input_module()`, and hence `core_module()` will take this object, in our example `stand.input`, as the starting point for the simulation. Now, we are ready to perform our simulations with:

```
sims.stand <- core_module(input = stand.input)
```

The object created by `core_module()` (in this case named `sims.stand`) is a list with all of the simulation results. To begin with, we can present the summary table by specie at the simulation age AD of 33 years.

```
sims.stand$sp.table
```

```
##   SPECIES       N      BA     QD    VTHA
## 1       1 459.274 44.1455 34.983 342.972
## 2       2  22.766  3.3865 43.520  26.310
## 3       3  15.837  1.9351 39.443  15.034
## 4       4  43.750  2.4339 26.614  18.909
## 5       0 541.626 51.9010 34.930 403.225
```

And, a table with the summary stand-level parameters by year that can be used to assess patterns and calculate rates. This is obtained as:

```
sims.stand$data.sim
```

```
##   AGE    HD    NHA     QD     BA   NHAN  NHA99    BAN
## 1  28 18.50 551.00 31.632 43.300 503.00 48.000 40.900
## 2  29 19.02 549.45 32.300 45.021 502.28 47.163 42.614
## 3  30 19.52 547.72 32.963 46.743 501.41 46.319 44.329
## 4  31 20.02 545.84 33.623 48.464 500.37 45.468 46.043
## 5  32 20.51 543.81 34.278 50.184 499.20 44.612 47.756
## 6  33 20.98 541.63 34.930 51.901 497.88 43.750 49.467
##     BA99    PBAN   PNHAN   SI   SDIP   VTOT
## 1 2.4000 0.94457 0.91289 13.9 65.019 299.12
## 2 2.4072 0.94653 0.91416 13.9 66.777 319.19
## 3 2.4142 0.94835 0.91543 13.9 68.505 339.52
## 4 2.4210 0.95005 0.91670 13.9 70.205 360.44
## 5 2.4275 0.95163 0.91796 13.9 71.874 381.75
## 6 2.4339 0.95311 0.91923 13.9 73.514 403.23
```

Note that in this table, you can see the changes on parameters for the complete stand (*e.g.* BA) and for all *Nothofagus* species aggregated (*e.g.* BAN).

In the current simulation presented before, all internal prediction equations only create simulations at the stand-level, without any dissagreggation. For example, in this case we do not have a diameters distribution as this was not requested. This is because the default behavior, specified in `input_module()`, is to not generate a diameter distribution (*i.e.* `ddiam=FALSE`)

3.2 Stand-level Simulation with the generation of a diameter distribution

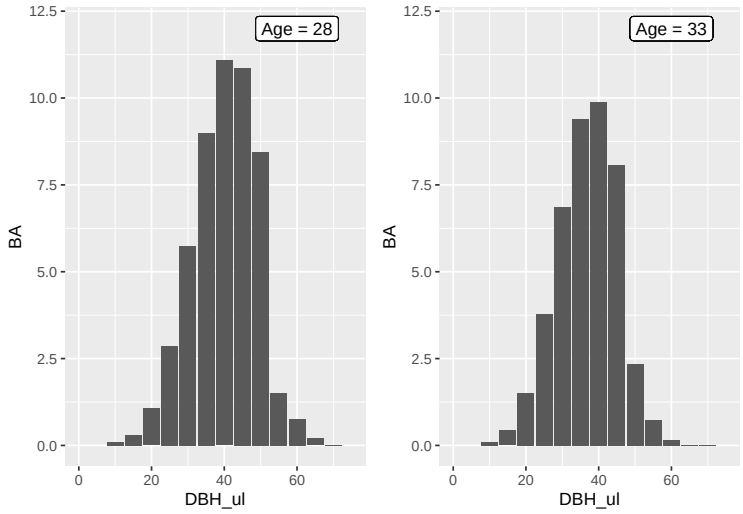
As indicated earlier, it is possible to generate a diameter distribution. This is obtained based on the method of parameter recovery using stand-level parameters. In NOTHOPACK this is generated for each year of simulation by specifying in `input_module()` that a diameter distribution is desired (*i.e.* `ddiam = TRUE`), this is shown in the following code

```
stand.input <- input_module(ZONE = 2, AD = 28, HD = 18.5, AF = 33,
    N = N, BA = BA, type = "stand", ddiam = TRUE, T_model = 1,
    comp = FALSE)
```

Note that, to show the available simulation options, we also changed some of the settings of this simulation. In this case, the selected taper model from all zones (`T_model=2`) is changed to zone specific (`T_model=1`).

Then, we can call `core_module()` and for illustration we are showing the distribution for basal area for ages 28 and 33 in our example.

```
sims.stand <- core_module(input = stand.input)
```



Chapter 4

Running a Tree-level Simulation

Tree-level simulations offer greater details on the simulated stand as in this case for NOTHOPACK diameter and height increment of each tree is modelled in an annual basis. In this section, we will show an example to run this type of simulation using the same data that was presented before as part of the processing inventory, namely:

```
str(plot_example)
```

```
## 'data.frame':    46 obs. of  6 variables:
##  $ ID     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ SPECIES: int  2 2 2 2 2 2 2 2 2 2 ...
##  $ DBH    : num  24 23.6 35.5 15.9 13.5 ...
##  $ HT     : num  NA NA 17.5 8.64 NA NA 15 NA NA NA ...
##  $ SS     : logi  NA NA NA NA NA NA ...
##  $ FT     : logi  NA NA NA NA NA NA ...
```

```
head(plot_example)
```

```
##   ID SPECIES   DBH    HT SS FT
## 1  1       2 24.00    NA NA NA
## 2  2       2 23.60    NA NA NA
## 3  3       2 35.47 17.50 NA NA
## 4  4       2 15.90  8.64 NA NA
## 5  5       2 13.50    NA NA NA
## 6  6       2 23.60    NA NA NA
```

To start we need to create an stand object using `input_module`. We will simulate from an initial age of 28 to 33. Note that `input_module` requires the specification `type='tree'` to allow a tree-level simulation. In this example we will use taper model 2. The code used is:

```
plot.input <- input_module(ZONE = 1, AD = 28, AF = 33, type = "tree",
    area = 500, tree.list = plot_example, T_model = 2)
```

Another key aspect is the use of `tree.list` where we specify the inventory data for the plot to simulate, and this has to have the specified column names.

Now, we can proceed to perform the tree-level simulation to $AF = 33$, with

```
tree.sim <- core_module(input = plot.input)
```

After the simulation, the tree.list from the new object is the *grown* tree.list from the original plot. We can, for instance review the growth prediction for any specific tree in the stand.

```
# tree #1: initial characteristics
head(plot.input$tree.list)
```

```
##   ID SPECIES   DBH     HT     SS FT  DBH0     VIND
## 1  1       2 24.00 16.827 3.1140 20 24.00 0.290415
## 2  2       2 23.60 16.739 3.1791 20 23.60 0.280420
## 3  3       2 35.47 17.500 1.4784 20 35.47 0.578748
## 4  4       2 15.90  8.640 3.8351 20 15.90 0.061964
## 5  5       2 13.50 13.203 3.9304 20 13.50 0.078372
## 6  6       2 23.60 16.739 3.2442 20 23.60 0.280420
```

```
# tree #1: final characteristics
head(tree.sim$tree.list)
```

```
##   ID SPECIES    DBH     HT     SS     FT   DBH0     VIND
## 1  1       2 25.320 18.957 3.1140 19.054 25.072 0.367066
## 2  2       2 24.883 18.844 3.1791 19.035 24.641 0.353844
## 3  3       2 38.245 20.020 1.4784 19.784 37.715 0.766181
## 4  4       2 16.670 10.076 3.8351 18.749 16.525 0.081961
## 5  5       2 14.151 14.315 3.9304 18.723 14.029 0.094096
## 6  6       2 24.861 18.834 3.2442 18.998 24.624 0.353103
```

It is clear from the above table the changes and growth in each of the trees. An important element is the column FT this is the expansion factor that originally is 20 trees/ha, and due to mortality, this has lowered to values between 18 and 20; therefore representing the change in number of trees. It is important to note that NOTHOPACK does not consider ingrowth in the simulations.

We can now look at the change in stand-level conditions for these simulations between initial and final age.

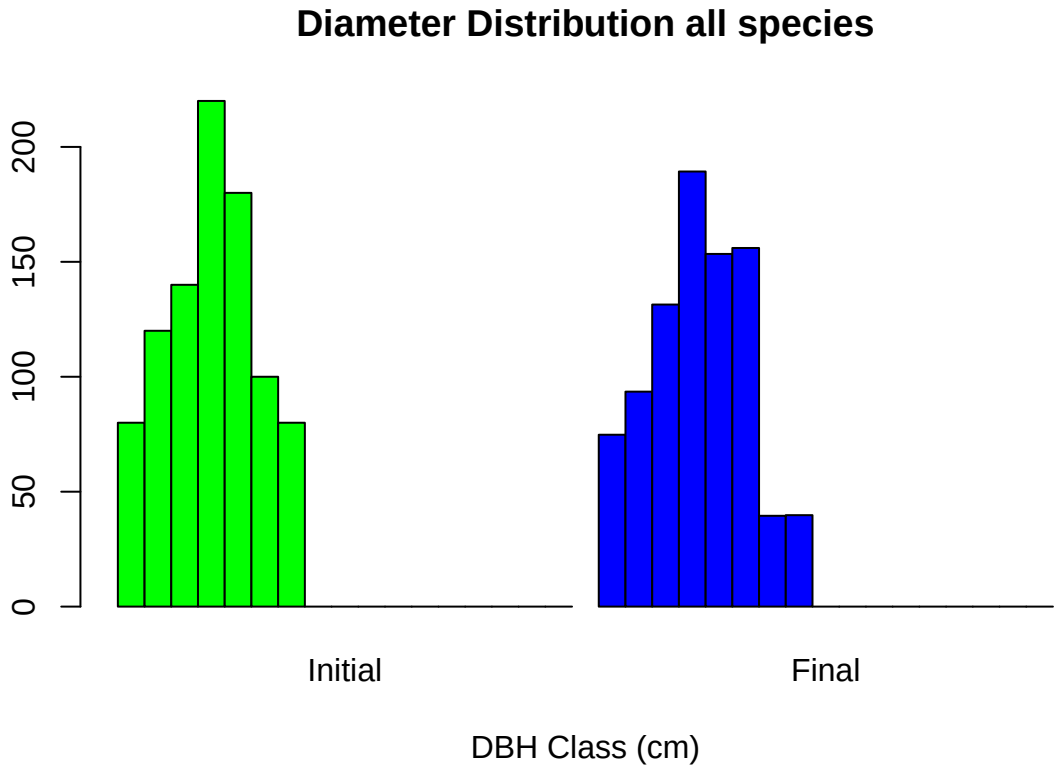
```
plot.input$sp.table
```

```
##   SPECIES   N       BA      QD    VTHA
## 1       1  60  1.08580 15.1794   6.836
## 2       2 780 38.91528 25.2039 245.444
## 3       3   0  0.00000  0.0000   0.000
## 4       4  80  0.31314  7.0596   1.040
## 5       0 920 40.31422 23.6206 253.320
```

```
tree.sim$sp.table
```

```
##   SPECIES       N       BA      QD    VTHA
## 1       1  56.283  1.12620 15.9616   7.759
## 2       2 746.713 42.61596 26.9566 304.010
## 3       3   0.000  0.00000  0.0000   0.000
## 4       4  74.768  0.32299  7.4164   1.069
## 5       0 877.763 44.06516 25.2822 312.838
```

Recall that SPECIES = 0 represents the total. We can now compare the histogram of the diameter distribution at AD (green) and then AF (blue) as shown below:



Chapter 5

Compatibility of Stand- and Tree-level Simulations

Often combining stand-level simulations with tree-level simulations offers increased predictability. The reason is because stand-level equations are more accurate on the long term but they do not have the granularity required that tree-level models have; however, the later are often only accurate for a few years (5-10) and then they suffer from important bias. For this reason, compatibility models that grow trees using the tree-level model but then modify them to match the paramters produced by stand-level models is always a good option.

NOTHOPACK has incorporated this option under two compatibility algorithms: proportional yield (PY) and proportional growth (PG). In this section, we will simulate the same stand considered earlier under `plot_example` with the compatibility method, and we will compare their results with the previous simulation. As expected, this needs to start with some plot inventory, and as before, it all starts with the `input_model` but here we will specify `type='comp'` to request compatibility, and then we run the `core_module`.

We will be running and comparing the two compatibility algorithms: proportional yield and proportional growth, as shown below:

```
# Compatibility simulation with proportional yield (PY)
plot.py <- input_module(ZONE = 1, AD = 28, AF = 38, type = "comp",
    area = 500, tree.list = plot_example, T_model = 2, comptype = "PY")
comp.py.sim <- core_module(input = plot.py)

# Compatibility simulation with proportional growth (PG)
plot.pg <- input_module(ZONE = 1, AD = 28, AF = 38, type = "comp",
    area = 500, tree.list = plot_example, T_model = 2, comptype = "PG")
comp.pg.sim <- core_module(input = plot.pg)
```

We will only see the simulations for PY, as the difference between both algorithms is minimal. Lets start by looking at the tree list at AF = 33, followed by the specie summary table:

```
head(comp.py.sim$tree.list)
```

```
##   ID SPECIES    DBH     HT     SS     FT   DBH0    VIND
## 1  1       2 28.681 20.865 3.1140 17.879 28.238 0.50721
## 2  2       2 28.180 20.728 3.1791 17.869 27.746 0.48874
## 3  3       2 44.219 22.334 1.4784 19.513 43.368 1.08983
## 4  4       2 18.782 11.176 3.8351 17.254 18.511 0.11509
## 5  5       2 15.950 15.029 3.9304 17.248 15.717 0.12386
## 6  6       2 28.124 20.704 3.2442 17.760 27.698 0.48647
```

```
comp.py.sim$sp.table
```

```
##   SPECIES       N       BA      QD    VTHA
## 1       1  51.879  1.32883 18.0589   9.665
## 2       2 707.361 53.43660 31.0137 406.851
## 3       3   0.000  0.00000  0.0000   0.000
## 4       4  68.669  0.37787  8.3703   1.231
## 5       0 827.910 55.14329 29.1212 417.747
```

When these results are compared to the tree simulations from the previous chapter, so differences are clear, but recall that in compatibility the stand-level equations are the ones that are used for the compatibility.

Chapter 6

Implementing a Thinning

Thinning is a critical component of management of *Nothofagus* second-growth forests. The simulator NOTHOPACK does include a module to allow for thinning. For the stand-level thinning, no additional information was available of thinning results, hence, the original stand-level models are considered without any specific modifications. Therefore, these results should be taken with care, particularly for this type of simulations.

In contrast, for the thinning done using tree-level simulations, there is more robust information. Here, individual tree growth is based on the use of tree-level equations that respond to changes on the levels of tree-to-tree competition, and not on original information generated by operational thinning. However, they are more flexible to model changes on levels of competition at this high granularity. In any case, results from this module should be considered with care, and we consider that reasonable simulations will be obtained with levels no higher than 5-10% of basal area removal from thinning.

NOTHOPACK uses, as part of the input data, the option `ATHIN` as part of `input_module`. This option will specify tha age of the thinning (on years), and it will require a few additional parameters depending if the simulation is done at the stand- or tree-level.

6.1 Thinning at the stand-level

In this example we simulate an stand with two different types of thinning strategies: a) with an equal thinning rate for every class and b) with a reduction of basal area to 90%.

The thinning specifications/parameters are all specified in the `input_module` as part of the input. For stand-level, thinning can be defined as a percentage of total basal area to be removed using `BARp` and the thinning age is specified with `ATHIN`. Note that this thinning age can be anywere betwee AD and AF, as the simulator will stop at the required age to perform the thinning.

Below, we specify a stand using their stand-level information. In this case the initial age of simulation is from 28 years until 45 years, with a thinning implemented at age 30. Note that

the basal area to remove is BARp = 10, and that this thinning is not changing the quadratic diameter; hence, Qd_ba = 1, which is the ration of quadratic diameter of the stand before against after thinning.

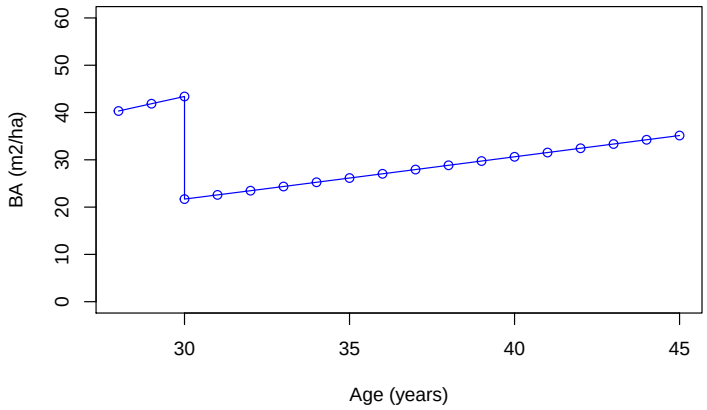
```
BA <- c(1.09, 38.92, 0, 0.31)
N <- c(60, 780, 0, 80)
stand.input <- input_module(ZONE = 2, AD = 28, AF = 45, ATHIN = 30,
    HD = 18.4, N = N, BA = BA, type = "stand", ddiam = FALSE,
    BARp = 50, QD_ba = 1)
# Simulating stand
sims.thin <- core_module(input = stand.input)
```

The object `sims.thin` stores information for all simulated years as well as a pre-thinning and a post-thinning stand-level characteristics at the thinning age (in this case 30). A few years are shown below:

```
sims.thin$data.sim[1:8, ]
```

```
##   AGE    HD    NHA     QD     BA   NHAN  NHA99    BAN
## 1  28 18.40 920.00 23.622 40.320 840.00 80.000 40.010
## 2  29 18.99 911.65 24.180 41.862 833.15 78.502 41.551
## 3  30 19.56 903.05 24.735 43.393 826.06 76.989 43.081
## 4  30 19.56 451.53 24.735 21.696 413.03 38.494 21.541
## 5  31 20.11 451.92 25.223 22.580 413.78 38.137 22.424
## 6  32 20.65 452.19 25.705 23.467 414.42 37.765 23.310
## 7  33 21.18 452.34 26.184 24.357 414.96 37.380 24.200
## 8  34 21.69 452.37 26.659 25.250 415.39 36.980 25.092
##      BA99    PBAN   PNHAN    SI   SDIP   VTOT
## 1 0.31000 0.99231 0.91304 13.12 91.380 278.78
## 2 0.31097 0.99257 0.91389 13.12 93.580 298.28
## 3 0.31190 0.99281 0.91475 13.12 95.715 318.02
## 4 0.15595 0.99281 0.91475 13.12 47.857 169.53
## 5 0.15640 0.99307 0.91561 13.12 49.237 181.05
## 6 0.15684 0.99332 0.91648 13.12 50.603 192.85
## 7 0.15726 0.99354 0.91736 13.12 51.955 204.94
## 8 0.15767 0.99376 0.91825 13.12 53.293 217.18
```

Clearly, there are two records for age 30, one before thinning and another after. The other elements are changing according to the thinning specifications, and from this point, the stand follows a different trajectory. For illustration, we visualize the plot dynamics of the thinned plot for basal area below.



6.2 Thinning at the tree-level

As with stand-level, all thinning specifications/parameters must be indicated in the `input_module`. In this case, a vector of thinning decision for each tree in a parallel ways as the that provided in the `tree.list` by including a vector, of the same length, under the option `FT.thin`. This vectro has a 0 if the tree is to be removed and 1 is if going to be kept after thinning. This additional flexibility allows for any type of thinning desired (below, above, or by specific species). As before, it is also required to indicate the thinning age with `ATHIN`.

In this example we will implement a tree-level thinning, and only for simplicity, we

The thinning vector should be the length of the tree.list and should have values of 0 to trees to be thinned and 1 for trees to be kept in the stand.

In this example, the thinning vector is named `FT.thin` and we specify a group of six trees with a value of 0; hence, they will the removed at age 30, when the thinning is planned.

```
FT.thin <- rep(1, length(plot_example$FT))
FT.thin[c(8, 11, 17, 31, 33, 38)] <- 0
```

The above vector is then added the simulation presented below:

```
# Processing the plot-level data
plot.input <- input_module(ZONE = 1, AD = 28, AF = 40, type = "tree",
    ATHIN = 30, FT.thin = FT.thin, area = 500, tree.list = plot_example,
    T_model = 2)
# Simulating to age AF by tree-level with thinning
sims.tree.thin <- core_module(input = plot.input)
```

```
## [1] "Tree thinning resulted in QD_ba = 0.99"
## [1] "Tree thinning resulted in BARp = 11.34"
```

The simulation returns messages with information about the results of the thinning. In this case, the proposed tree-level thinning vector, `FT.thin`, resulted in a reduction of 11.34% of the basal and a ratio of quadratic diameter of stand before against after thinning of 0.99.

The final tree-list with thinning is presented below, and the user can verify that the residual trees are there, but in contrast with a simulation without thinning these present larger diameters (note that IDs and FT vlaues will be slightly different).

```
head(sims.tree.thin$tree.list, 8)
```

```
##   ID SPECIES    DBH     HT     SS     FT   DBH0    VIND
## 1  1       2 26.873 21.557 3.1140 18.126 26.668 0.47302
## 2  2       2 26.399 21.411 3.1791 18.112 26.198 0.45538
## 3  3       2 41.519 23.152 1.4784 19.524 41.073 1.03462
## 4  4       2 17.582 11.656 3.8351 17.572 17.462 0.10804
## 5  5       2 14.921 15.406 3.9304 17.506 14.820 0.11308
## 6  6       2 26.341 21.383 3.2442 18.009 26.146 0.45303
## 7  7       2 28.458 19.902 2.9754 18.279 28.233 0.47320
## 8  9       2 18.931 18.202 3.7741 17.669 18.799 0.21128
```

As we did with the stand-level simulations, we can also see the information for all simulated years as well as a pre-thinning and a post-thinning stand-level characteristics at the thinning age below:

```
sims.tree.thin$data.sim[1:8, ]
```

```
##   AGE     HD    NHA     QD     BA   NHAN  NHA99    BAN
## 1  28 18.394 920.00 23.621 40.314 840.00 80.000 40.001
## 2  29 19.030 911.65 23.966 41.124 832.69 78.961 40.808
## 3  30 19.650 903.24 24.304 41.903 825.32 77.916 41.585
## 4  30 19.650 785.60 24.537 37.149 707.69 77.916 36.831
## 5  31 20.260 780.14 24.863 37.875 702.99 77.144 37.554
## 6  32 20.860 774.59 25.182 38.578 698.23 76.362 38.254
## 7  33 21.450 768.98 25.495 39.257 693.41 75.573 38.930
## 8  34 22.030 763.31 25.803 39.914 688.53 74.777 39.585
##      BA99    PBAN   PNHAN SI   SDIP   VTOT
## 1 0.31314 0.99223 0.91304 13 91.370 253.32
## 2 0.31576 0.99232 0.91339 13 92.413 265.54
## 3 0.31803 0.99241 0.91374 13 93.389 277.57
## 4 0.31803 0.99144 0.90082 13 82.330 245.20
## 5 0.32117 0.99152 0.90112 13 83.291 256.16
## 6 0.32402 0.99160 0.90142 13 84.201 267.04
## 7 0.32658 0.99168 0.90172 13 85.062 277.81
## 8 0.32888 0.99176 0.90204 13 85.876 288.46
```

Finally, we can also visualize the stand growth over the years based on the above simulation for basal area:

Chapter 7

Closing Remarks

Some of the capabilities of the growth model (G&Y) simulator for *Nothofagus* second-growth forests NOTHOPACK was presented in this document. However, there are more details that were not presented here but they can be of interest for the user. For example, there are several taper equations incorporated into this model that can be used to estimate volume for different products of interest; and these are completemented by functions that calculate stem diameter of height for specific trees.

In addition, there are several parametrized height-diameter equations for different zones that require stand-level parameters that can be used to estimate missing heights with better precision than local height-diameter functions.

We also have allowed the user to select among a variaty of different models for different aspects. For example, there are several mortality models that can be considered, and it is expected that some will provide better results than others, depending on the specific conditions. Similarly, we have allowed for different stand-level volume equations, and individual annual increment in DBH models. In all cases, we recommend that the user tests different options and, with the use of real information, verifies and selects the model that better represents their data. We have selected a few options by default, but given the variable nature of the system, and the wide geographical range of conditions for these forests, we suspect some models will be more accurate.

Finally, we like to indicate that this G&Y simulators is a dynamic tool that was produced with limited filed observation and considering a few biological, and simplifying, assumptions. As more understanding of these forests is generated and more field information and trials are available, these models can, and should, be modified and improved. It is for that reason that we have considered this library NOTHOPACK as open source available for all scientific and forest community.