

Forest Modeling Exercise

Process-based and empirical modeling exercises

Olalla Díaz-Yáñez¹, Laura Dobor², Katarina Merganicova³ and Mats Nieberg⁴

1. olalla.diaz@usys.ethz.ch (ETH Zurich)
2. dobor@fld.czu.cz (CZU Prague)
3. merganicova@fld.czu.cz (CZU Prague)
4. mats.nieberg@pik-potsdam.de (PIK Potsdam | EFI)



Interdisciplinary Summer School
Ljubljana 2023
Repository available in GitHub

Table of contents

1	Overview	1
1.1	Data and model	4
2	Process based	5
2.1	iLand model	5
2.1.1	General	5
2.1.2	Input and outputs of iLand	5
2.1.3	How to work with the model	6
2.2	Excercises	6
2.2.1	Getting started	7
2.2.2	Explore input files	8
2.2.3	Run the model	11
2.2.4	Working with the output	11
2.2.5	Run alternative scenarios	12
2.2.6	Question A - Group 1	13
2.2.7	Question B - Group 1	23
2.2.8	Question A - Group 2	33
2.2.9	Question B - Group 2	43
2.2.10	Question A - Group 3	52
2.2.11	Question B - Group 3	62
2.2.12	Question A - Group 4	71
2.2.13	Question B - Group 4	81
3	Empirical modeling	91
3.1	Data	91
3.1.1	Data description	92
3.2	Species description	93
3.2.1	Species 1 - Bryophytes	93
3.2.2	Species 2 - Great spotted woodpecker	93
3.2.3	Species 3 - Eurasian treecreeper	93
3.3	Models	94
3.3.1	Generalized Linear Models (GLMs)	94
3.3.2	Boosted regression trees (BRTs)	94

Table of contents

3.4	The exercise	95
3.4.1	The project folder	95
3.4.2	Download the data	95
3.4.3	Explore the data	96
3.4.4	Question C - Group 1	99
3.4.5	Question D - Group 1	104
3.4.6	Question C - Group 2	108
3.4.7	Question D - Group 2	113
3.4.8	Question C - Group 3	117
3.4.9	Question D - Group 3	125
3.4.10	Question C - Group 4	134
3.4.11	Question D - Group 4	143
4	References and further reading	153
5	Appendix - Initialize trees for the model based on inventory data	155
5.1	Inventory data	155
5.2	iLand needs	155
5.2.1	Read in the inventory data to R visualize it. Read also a table with species latin names and short names in iLand.	155
5.2.2	Calculate some statistics for populating remaining areas:	157
5.2.3	Generate trees for each plot	158
5.2.4	Lets remove the trees in the middle, and put back the real trees from the inventory	162
5.2.5	Give height and plot final layout	163
5.2.6	Make files for iLand:	163

1 Overview

This document contains the instructions and solutions of the modeling group assignments. First, you will find the section for the process model approach (Section 2) and after that the section for the empirical modelling (Section 3). This document is very long because it describes all the exercises and solutions step by step.

You will work per groups in the four groups established at the beginning of the week. Each of the four groups will be divided internally in four subgroups (A-D). Each subgroup (A-D) inside each group (1-4) will address one question that will have to be answered using one approach. In the table below you can find a link to the section to each question e.g. group 1 subgroup A will answer the questions “**How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 1?**” using the approach based on the model iLand and will have the full instructions and solution in Section 2.2.6.

Please, always ask your coach if you do not know what is the section where you should be working in.

Group	Subgroup Question	Question	Approach	Go to
1	A	How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 1?	iLand	Section 2.2.6
1	B	How the species distribution and total living biomass C content changing in time on Plot 1?	iLand	Section 2.2.8

1 Overview

Group	Subgroup Question	Question	Approach	Go to
1	C	Does a more diverse forest in structure and composition have more Bryophytes species?	GLM	Section 3.4.4
1	D	Is the number of Bryophytes species affected by forest management type and the forest structural diversity?	GLM	Section 3.4.5
2	A	How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 2?	iLand	Section 2.2.8
2	B	How the species distribution and total living biomass C content changing in time on Plot 2?	iLand	Section 2.2.9
2	C	Does a more diverse forest in structure and composition have more bird species?	GLM	Section 3.4.6

1 Overview

Group	Subgroup Question	Question	Approach	Go to
2	D	Is the number of bird species affected by forest management type and the forest structural diversity?	GLM	Section 3.4.7
3	A	How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 3?	iLand	Section 2.2.10
3	B	How the species distribution and total living biomass C content changing in time on Plot 3?	iLand	Section 2.2.11
3	C	Is the presence of the Great spotted woodpecker affected by forest density?	BRT	Section 3.4.8
3	D	Is the presence of the Great spotted woodpecker affected by forest diversity?	BRT	Section 3.4.9

Group	Subgroup Question	Question	Approach	Go to
4	A	How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 4?	iLand	Section 2.2.12
4	B	How the species distribution and total living biomass C content changing in time on Plot 4?	iLand	Section 2.2.13
4	C	Is the presence of the Eurasian treecreeper affected by forest density?	BRT	Section 3.4.10
4	D	Is the presence of the Eurasian treecreeper affected by forest management?	BRT	Section 3.4.11

1.1 Data and model

We provide all data, model and scripts for the excercises. However if you will use the model iLand for your research in the future please get in contact with the model developers at the Technical University München: Werner Rammer (werner.rammer@tum.de) and Rupert Seidl (rupert.seidl@tum.de). Regarding the biodiversity data used in the empirical modeling processed in the emprical modeling part, in case of further usage please contact Jeňýk Hofmeister at the Czech University of Life Sciences Prague (jenyk.hofmeister@email.cz).

2 Process based

2.1 iLand model

2.1.1 General

iLand is an ecosystem model that simulates forest landscape dynamics, including growth and regeneration, disturbance dynamics, and management in a spatially explicit manner. The main entity in the model is a tree, for which the demographic processes are simulated. Processes at the stand and landscape scale constrain the dynamics of individual trees and thus allow for the scaling of tree-scale processes to large areas. The model explicitly simulates tree competition for resources such as light, water, and nutrients. A light use efficiency approach is used to simulate the production physiology. Carbon starvation is used as a process oriented indicator of tree stress, which can result from competition for resources as well as suboptimal environmental conditions for tree growth (e.g., drought).

iLand's mechanistic representation of forest disturbances and vegetation dynamics, as well as the climatic sensitivity of these processes, makes it well suited for the research of disturbance dynamics under climate change. Additionally, flexible implementation of management operations, which include planting after harvests or natural disturbances, thinning, harvesting, and post-disturbance salvaging, allows for testing the effects of various disturbance management strategies.

A detailed model documentation is available on iLand WIKI page: <https://iland-model.org/iLand+Hub>

2.1.2 Input and outputs of iLand

iLand needs information in specific formats about environment (mainly climate and soil) and trees. Daily climate data is needed (<https://iland-model.org/ClimateData>) for minimum, maximum temperature, precipitation, vapor pressure deficit and radiation. Soil depth and soil texture (sand, silt, clay %) information are needed. The soil is represented as a one-layer bucket, no depth-dependent information is needed. The model needs species-specific parameters (<https://iland-model.org/species+parameter>) that are describing the behavior of each tree species (growth, mortality, competition etc.). Currently there are 31 species parametrized for iLand, and the species specific parameters

2 Process based

are stored in a database that is an input for the model. The parameters were previously calibrated for European conditions (species_param_europe.sqlite). iLand uses short codes for species derived from the latin names (e.g. Picea Abies - piab, Fagus Sylvatica - fasy, ..) Information on the initial forest is also required to start a simulation.

As iLand is a landscape scale model, both climate, soil and tree information need to be set spatially. For larger areas environmental maps can be set using a 100x100m resolution grid, a so called resource grid with resource units, and tree information can be set by maximum 10x10m resolution grid on a stand grid using stand IDs. There are more options how to initialize a landscape in iLand depending on our spatial scales and available data (<https://iland-model.org/initialization>).

In our case now we will use iLand in a stand-scale mode (called torus), where we simulate only one 100x100m pixel, 1 ha forest area. We will use single-tree initialization method, where we set the x-y coordinates of the trees inside the 100x100m area, tree species, dbh and height of each tree. From inventory we have data for the a circle with 13.82m radius. We generated trees for the 1ha area based on the plot-data. See details later.

The output of the model is an sqlite database file. The output produced by the model is highly versatile and needs to be set in advance of the simulation which output tables we would like to get.

2.1.3 How to work with the model

Simulations are driven by one xml file, the project file. After starting iland.exe, we should call the project file, load the initial forest and then run the simulation for the required length of period. iLand comes with a graphical interface where you can visualize single trees and many other things: <https://iland-model.org/iLand+viewer>

2.2 Exercises

Your group will work with the plot which you measured and worked with in the previous days (Plot1-Plot4). The 2 process based subgroups will do the same modeling experiment, just addressing different questions based on the results (see the 2 questions below).

You will simulate the growth and development using the process-based model, iLand. The task is to simulate your study plot for 100 years under reference conditions and one/two simplified scenarios that you could define upon your ideas. You can define scenarios assuming different temperature, precipitation and CO2 concentration levels (see more details how to do it later below). The questions that you should address forming 2 sub-groups are the following:

2 Process based

- A) How biodiversity indices are changing in time and across the simulated scenario(s)?
(Calculate indices)
- B) How the species distribution and total living biomass C content changing in time?
Compare 0 year and 100 year status in the reference case and in the case of your scenario(s)!

2.2.1 Getting started

Navigate to this folder and download the materials for the process-based modeling exercise: https://drive.google.com/drive/folders/1o2_K3ZN-SnPR44UzZoHj9lFZQu-vU3wg?usp=drive_link

Go to the *iLand_simulations* folder. Here you will find several folders that are the part of the model, others are containing input data for the model and one folder is dedicated for the outputs. Additionally, you will find *iland.exe* and *.xml* files. The *.xml* file is called *project file*. This file is driving the settings of simulations, such as input/output file names, output variables and many other settings. This file is what you need to run in the model.

The xml file structure follows a tree structure, e.g:

```
<root>
  <child>
    <subchild>
      ...
    </subchild>
  </child>
</root>
```

The iLand project file consists of five main sections:

- system: settings like file path, database locations, or logging
- model: main settings of the models: extent (world), site specific setting, used climate, model initialization and management
- modules: settings related to the disturbance modules of iLand
- output: definition of which outputs should be created by the model
- user: this section can be used for user-defined settings

More details about the project file structure and meaning of each record can be found here: <https://iland-model.org/project+file>

We prepared one project file for each group (for each simulation plot). You can right away run it, and then copy-paste and modify it to specify your own scenarios for alternative simulations. You can open a project file in any text editor, but we recommend to use Notepad++ for this (<https://notepad-plus-plus.org/downloads/>).

2.2.2 Explore input files

The following input files are needed and prepared for you to run the model: This is all prepared, YOU DO NOT HAVE TO CHANGE ANYTHING:

- Species parameter file: database/species_param_europe.sqlite Containing the parameters for each species that are driving growth, mortality, establishment and other processes. <https://iland-model.org/species+parameter>

Set in project file:

```
<path>
<home></home>
<database>database</database> <!------- HERE
    <lip>lip</lip>
    <temp>temp</temp>
    <script>scripts</script>
    <init>init</init>
    <output>output</output>
</path>
...
<database>
    <in>species_param_europe.sqlite</in> <!------- HERE
    <out>Output_plot1_4deg_30percdrier.sqlite</out>
    <climate>E-OBSv27_Roznik_46.05_14.45_1961-1990.sqlite</climate>
</database>
```

- Climate file: database/E-OBSv27_Roznik_46.05_14.45_1961-1990.sqlite
Daily climate data for the area based on the E-OBS dataset. <https://iland-model.org/ClimateData> https://surfobs.climate.copernicus.eu/dataaccess/access_eobs.php#datafiles We are using the same climate input for all plots. The E-OBS data is representative for a 0.1x0.1 degree resolution gridbox

Set in project file:

```
<database>
    <in>species_param_europe.sqlite</in>
    <out>Output_plot1_4deg_30percdrier.sqlite</out>
    <climate>E-OBSv27_Roznik_46.05_14.45_1961-1990.sqlite</climate> <!------- HERE
</database>
```

- Enviromental file: gis/Environment.txt
The structure of this file is very flexible, more or less columns can be set. If we would have a landscape with larger area with different environments, we could set

2 Process based

more lines into this file. For now we have only one simulation pixel with 1ha area. Here we set which climate table to use from the climate file (there could be more tables, but now we have only one), what is the soil texture, soil depth. The column names are referring to specific lines in the project file.

For example: if we set here “model.site.pctSand”, the model will use this sand % value instead of the value that we have in the project file under:

```
<model>
...
  <site>
    <availableNitrogen>84</availableNitrogen> <!-- kg/ha/yr -->
    <soilDepth>38</soilDepth> <!-- in cm -->
    <pctSand>9</pctSand>
    <pctSilt>53</pctSilt>
    <pctClay>38</pctClay>
  ...
</site>
...
</model>
```

Set in project file:

```
<environmentGrid>gis/environment_grid.asc</environmentGrid>
<environmentFile>gis/Environment.txt</environmentFile> <!-- HERE
```

- Environment grid: `gis/environment_grid.asc` This would specify the map of different environments. Now we have only 1 pixel, with the id of 1. In the previous file we had also id column, this specifies we want to use that environment for our pixel.

Set in project file:

```
<environmentGrid>gis/environment_grid.asc</environmentGrid> <!-- HERE
<environmentFile>gis/Environment.txt</environmentFile>
```

- Tree initialization file: `init/Tree_init_plot_1.txt` This is the list of the tree with x,y coordinates, species, dbh, height and optionally with age. (age=0 means, there is no data on age, model will generate) There are other options to set initial trees/landscape (e.g. using distributions, number of trees in a given dbh range, or using outputs of another simulation) More details: <https://iland-model.org/initialize+trees>

```

    <initialization>
    <type>single</type>
    <mode>unit</mode>
    ...
    <file>Tree_init_plot_1.txt</file> <!-- HERE
    <saplingFile></saplingFile>
    ...
</initialization>

```

Here we initialize trees taller than 4 meters. Trees smaller than 4m are handled as sapling cohorts in separated input file. Now we do not put there saplings and we do not have a sapling file.

We assume now no-management scenario for all simulations. Generally iLand has a very sophisticated so called “agent based management engine” functionality for large landscapes, but also simple management activities are possible for stand-scale simulations. However, for the sake of simplicity in these examples we do not simulate any management activities. This need to keep in mind during the interpretation of the results.

NOTE: in the graphical interface you also wont see trees smaller than 4m, because they are not simulated as individuals rather in cohorts. But as regeneration will work, after a while as they grow above 4m, they become individual trees and we can see them also visually and they have own properties (dbh, height, age,...<https://iland-model.org/tree+variables>). More details on saplings: <https://iland-model.org/sapling+growth+and+competition>

Setting the output tables: In the output section you can set which tables you want to enable. Here you can find more info on the structure and variables of each table: <https://iland-model.org/Outputs>

To answer the questions the most straight forward way is to look the *landscape* output and use the *total_carbon_kg* column (total carbon in living biomass (aboveground compartments and roots) of all living trees (including regeneration layer) (kg/ha)). This output table aggregates on the level of landscape x species. Values are always aggregated per hectare. The output is created after the growth of the year. We pre-set the project file to have this output. Additionally we will work with the *tree* output, where individual trees are recorded with position.

```

<output>

<tree>
<enabled>true</enabled>
</tree>

...

```

```
<landscape>
<enabled>true</enabled>
</landscape>

</output>
```

2.2.3 Run the model

- Double click on `iland.exe` and wait until the graphical interface opens.
- Load the required project file on the left-hand-side under *iLand project file* by browsing on your computer after clicking on the folder icon next to the box.
- After you selected a project file, click on the “Create Model” on the top-left part of the window with an icon of a Globe.
- Wait until the simulation area shown in the middle.
- Try out some visualizations on the right hand-side “Visualization options”. e.g individual trees + color by species. Here you can explore the initial stage of the 1 ha simulation area.
- To run the model click on Run Model icon on the top and give 100 years to simulate.
- On the bottom of the window you can follow how fast the model is running and see if it is finished.
- Stage of the last year remains in the simulation area, and you can go to the folder *output* to check your output file.
- When you want to run another simulation, first click on “Destroy”, then start the process again by “Creating Model” for the same project file or loading new project file.

2.2.4 Working with the output

The output database has a *.sqlite* extension. You can explore outputs:

- Open and browse it in the *DB Browser* software that was on the software list. (<https://sqlitebrowser.org/>)
- Load the file into R, where you can make additional data analyses needed to answer the questions in the exercise.

```
file <-
  here::here("model/iLand_simulations/output/Output_plot1.sqlite")

sqlite.driver <- RSQLite::dbDriver("SQLite")
db1 <-
```

```

    RSQLite::dbConnect(sqlite.driver, dbname = file) # connect to the file
    tabs <- RSQLite::dbListTables(db1) # explore the tables in the file
    print(tabs)

    landscape <- RSQLite::dbReadTable(db1, "landscape")

    RSQLite::dbDisconnect(db1)

    summary(landscape)
    head(landscape)

```

There are different output tables in iLand and all have specific structure, column and spatial representation of the records. The landscape output gives information on the whole landscape, but here we will work with one 1 ha pixel that is our “whole” landscape for now. Here species specific information is also available for each year. More about the meaning of the columns in the landscape output you can find here: https://iland-model.org/Outputs#Landscape_aggregates_per_species Not always all outputs are produced, in the project file we can set which output tables we want to enable, and those will be created.

2.2.5 Run alternative scenarios

The prepared simulation is using the 30years climate of 1961-1990, and repeating during the simulation. (The model copy paste the 30year time series after each other to fill up the simulation period) For CO2 concentration 400ppm is given. Your task is to run 2 extra alternative scenarios where you modify the output data. For example assuming 800ppm, and 4degree warming, or keep CO2 on 400ppm and decrease precipitation by 20%.

All of these modification can be done in the project file directly, you do not need to modify input data files!

For this, go to the climate section and modify the values there, and save as the project file as your alternative scenario.

```

<climate>
<co2concentration>400</co2concentration> <!------- HERE
    <tableName>Roznik</tableName>
<batchYears>30</batchYears>
    <temperatureShift>4</temperatureShift> <!------- HERE
    <precipitationShift>0.8</precipitationShift> <!------- HERE
    <randomSamplingEnabled>>false</randomSamplingEnabled>

```

```
<randomSamplingList></randomSamplingList>
</climate>
```

Note that the same CO₂ concentration will be used during the whole simulation period (there is a possibility to give annually changing CO₂ concentration using an external file, but for now we won't use it for simplicity). The temperature shift is applied by the model on each day's temperature values. The precipitation shift is a scaler that the model applies on each day's precipitation amount. (0.8 gives 20% decrease)

DO NOT FORGET TO CHANGE THE OUTPUT FILE NAME AS WELL for your alternative scenario, otherwise it will overwrite the previous output. You can do it in this section in the beginning of the project file:

```
<database>
<in>species_param_europe.sqlite</in>
<out>Output_plot1_4deg_20percldrier.sqlite</out> <!------- HERE
<climate>E-OBSv27_Roznik_46.05_14.45_1961-1990.sqlite</climate>
</database>
```

Running the alternative scenario follows the same steps as before.

2.2.6 Question A - Group 1

- How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 1?

Run the model as it is described in the previous chapters to have at least 2 simulations completed: one with reference conditions and one scenario with some changes in the environment. Check if you have the output files in the output folder: *iLand_simulations/output/*. If you manage to simulate 2 scenarios, you can expand the code to have file3, and 3rd from each variable, or make the comparison in 2 step always comparing one scenario to the reference conditions at the time.

Open the *summerSchoolExercise.Rproj* project in R studio and start working there.

2.2.6.1 Read the tree output table

Read in the *tree* output table from two outputs you have for your plot:

2 Process based

```
path1 <- "model/iLand_simulations/output/Output_plot1.sqlite"
path2 <-
  "model/iLand_simulations/output/Output_plot1_4deg_20percdrier.sqlite"

file1 <- here::here(path1)
file2 <- here::here(path2)

# Give some name for the three simulations.
name1 <- "reference"
name2 <- "4deg_20percdrier"

# Read in data using the RSQLite package
sqlite.driver <- RSQLite::dbDriver("SQLite")
db1 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file1) # connect to the file1

tables.in.the.file <-
  RSQLite::dbListTables(db1) # explore the tables in the file1
print(tables.in.the.file)
```

```
[1] "carbon"           "carbonflow"       "dynamicstand"
[4] "landscape"        "landscape_removed" "runinfo"
[7] "stand"            "tree"
```

```
# We will work with "tree" table and tree-scale data:
tree1 <- RSQLite::dbReadTable(db1, "tree")
RSQLite::dbDisconnect(db1) # disconnect to the file1

# READ IN DATA FROM THE SECOND FILE: -----
db2 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file2) # connect to the file2
tree2 <- RSQLite::dbReadTable(db2, "tree")
RSQLite::dbDisconnect(db2)
```

Merge the data from the two files together and make a column which tells which comes from which simulation, and study the table. The column “run” will support the plotting where we can use facets in ggplot.

```
tree <- rbind(tree1 |> dplyr::mutate(run = name1) ,
              tree2 |> dplyr::mutate(run = name2))

head(tree)
```

2 Process based

	year	ru	rid	species	id	x	y	dbh	height	basalArea	volume_m3	age
1	0	0	1	fasy	1	79	51	69.45111	39.46735	0.3788334	6.474022	493
2	0	0	1	fasy	2	69	91	68.95086	39.18307	0.3733956	6.335131	489
3	0	0	1	fasy	3	59	83	68.42681	38.88527	0.3677414	6.191780	486
4	0	0	1	piab	4	99	19	68.36907	45.96828	0.3671210	7.138516	492
5	0	0	1	fasy	5	63	61	66.87022	38.00070	0.3512007	5.778762	475
6	0	0	1	fasy	6	55	61	65.80000	37.39252	0.3400492	5.505722	467
	leafArea_m2	foliageMass	stemMass	branchMass	fineRootMass	coarseRootMass						
1	322.7507	29.34097	3099.621	378.6909	22.00573	337.2626						
2	318.8086	28.98260	3049.604	372.4465	21.73695	331.6535						
3	314.7003	28.60912	2997.693	365.9680	21.45684	325.8348						
4	295.8482	69.61134	2192.064	365.2582	52.20850	526.4032						
5	302.6273	27.51158	2846.413	347.1027	20.63368	308.8961						
6	294.4398	26.76726	2744.921	334.4585	20.07544	297.5477						
	lri	lightResponse	stressIndex	reserve_kg	treeFlags	run						
1	0.2646746		0	51.34669	0	reference						
2	0.5870413		0	50.71954	0	reference						
3	0.3931382		0	50.06596	0	reference						
4	1.0000000		0	121.81984	0	reference						
5	0.3406834		0	48.14526	0	reference						
6	0.4422086		0	46.84270	0	reference						

2.2.6.2 Visualize trees in year 0 and year 100!

To explore a bit the simulation, plot the tree locations at the beginning and the end of the simulation! For plotting we use the colors for the species, and the size of a circles to show dbh differences. Here we divided dbh with the value 20 just for visualization, not to have too huge circles shading each others completely, but still see the trees.

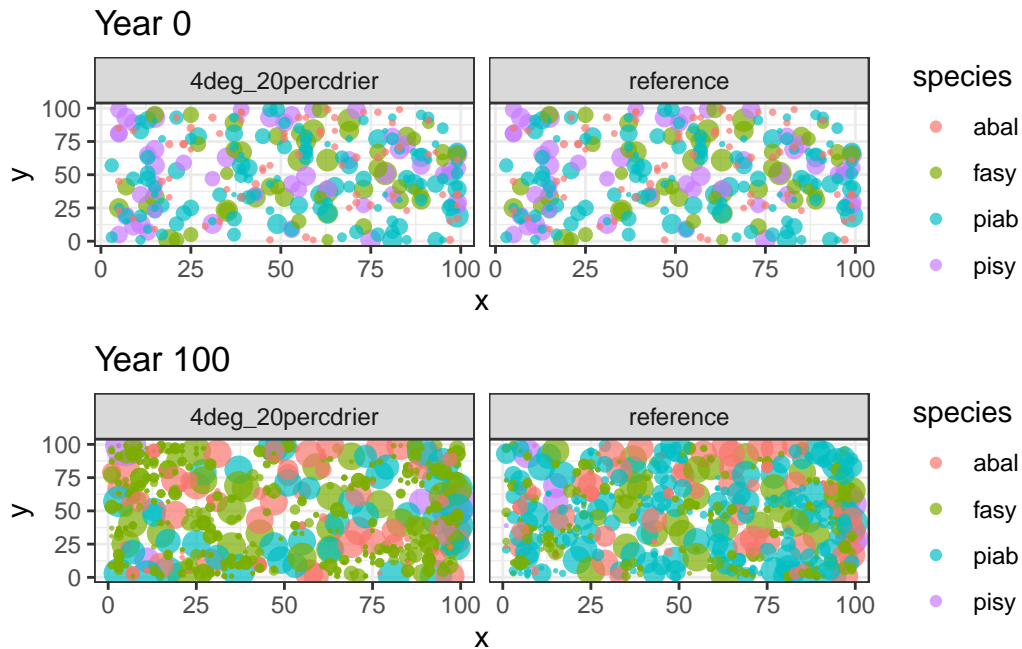
```
tree0 <- tree |> dplyr::filter(year == 0)
g1 <-
  ggplot2::ggplot(tree0, ggplot2::aes(x = x, y = y, color = species)) +
  ggplot2::geom_point(size = tree0$dbh / 20, alpha = 0.7) +
  ggplot2::ggtitle("Year 0") +
  ggplot2::facet_wrap(~ run) + ggplot2::theme_bw()

tree100 <- tree |> dplyr::filter(year == 100)
g2 <-
  ggplot2::ggplot(tree100, ggplot2::aes(x = x, y = y, color = species)) +
  ggplot2::geom_point(size = tree100$dbh / 20, alpha = 0.7) +
  ggplot2::ggtitle("Year 100") +
```

2 Process based

```
ggplot2::facet_wrap(~ run) + ggplot2::theme_bw()

gridExtra::grid.arrange(g1, g2, ncol = 1)
```



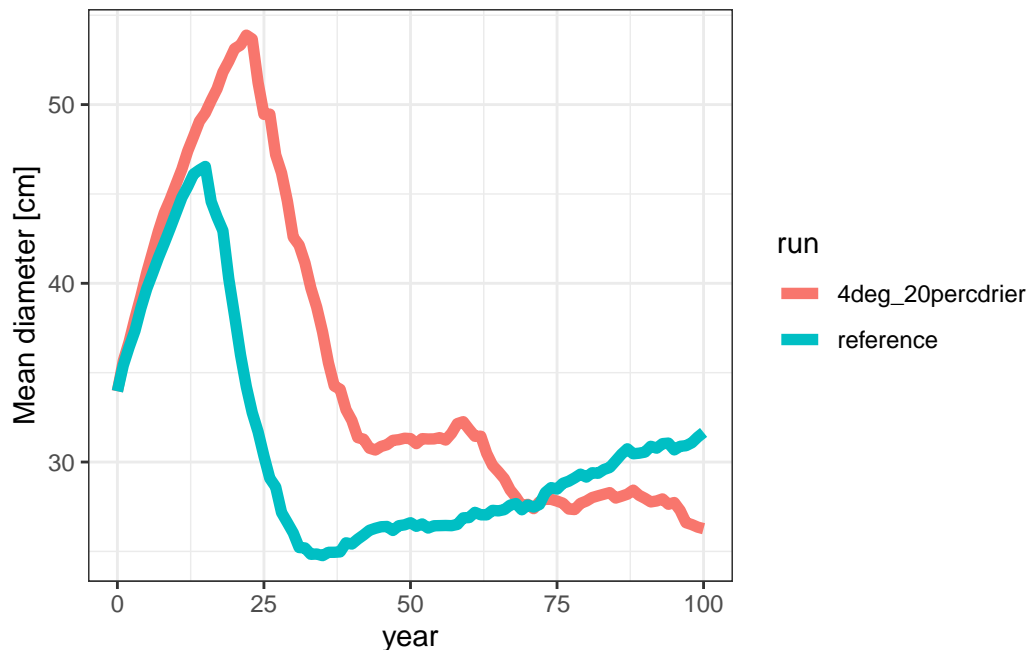
We can see that the species composition has changed and the forest become more dense. We can even identify the same trees at the same location.

2.2.6.3 Visualize some changes in time!

```
sum.table <- tree |> dplyr::group_by(year, run) |>
  dplyr::summarise(
    N = dplyr::n(),
    MD = mean(dbh, na.rm = TRUE),
    #mean diameter
    BA = sum(basalArea)
  ) #basal area

ggplot2::ggplot(sum.table, ggplot2::aes(x = year,
                                         y = MD, color = run)) +
  ggplot2::geom_line(lwd = 2) +
  ggplot2::ylab("Mean diameter [cm]") + ggplot2::theme_bw()
```

2 Process based



We can see that there is an initial increase in mean dbh, then a drop after 20/25 years. This is due to the growing regeneration layer that is produced by the seeds of the existing trees. In the initial year we do not have regeneration layer in the simulations (it is possible to put there, but we do not have it now). And until the small trees grow up to higher than 4 m they are not appearing in the individual *tree* output as they are handled in cohorts.

2.2.6.4 Species diversity

To study biodiversity aspects, we can calculate biodiversity indicators. Let's use the *adiv* package for species diversity. First we need to change the database structure to have the number of trees for each species in different columns (`pivot_wider`) without any additional columns for the *adiv* package *speciesdiv* function. We work first only with tree data from one simulation (*tree1*). You can find the documentation of the *adiv* package here: "[documents/adiv.pdf](#)"

```
# First we need to change the
tlong1 <- tree1 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
```

2 Process based

```
      values_from = 'N',
      values_fill = 0
    ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

head(tlong1)
```

```
# A tibble: 6 x 4
  abal  fasy  piab  pisy
  <int> <int> <int> <int>
1     68    50   120    49
2     68    49   115    49
3     68    48   111    48
4     68    48   110    47
5     66    48   107    46
6     65    48   105    45
```

Then we apply the “speciesdiv” function and we add back the year and run columns to have the information.

```
div1 <- data.frame(activ::speciesdiv(tlong1)) |>
  dplyr::mutate(year = unique(tree1$year), run = name1)
head(div1)
```

```
  richness GiniSimpson  Simpson  Shannon  Margalef Menhinick  McIntosh year
1         4   0.7095388 3.442800 1.312005 0.5300838 0.2361125 0.4899779    0
2         4   0.7131369 3.485983 1.318098 0.5320701 0.2386200 0.4938655    1
3         4   0.7150017 3.508792 1.321063 0.5341147 0.2412091 0.4960613    2
4         4   0.7150505 3.509394 1.320996 0.5348097 0.2420910 0.4962264    3
5         4   0.7162956 3.524796 1.323410 0.5369369 0.2447960 0.4978275    4
6         4   0.7169397 3.532816 1.324555 0.5383914 0.2466506 0.4987181    5

run
1 reference
2 reference
3 reference
4 reference
5 reference
6 reference
```

We make the same steps for the other simulation results (tree2).

2 Process based

```
tlong2 <- tree2 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

div2 <- data.frame(adiv::speciesdiv(tlong2)) |>
  dplyr::mutate(year = unique(tree2$year), run = name2)
```

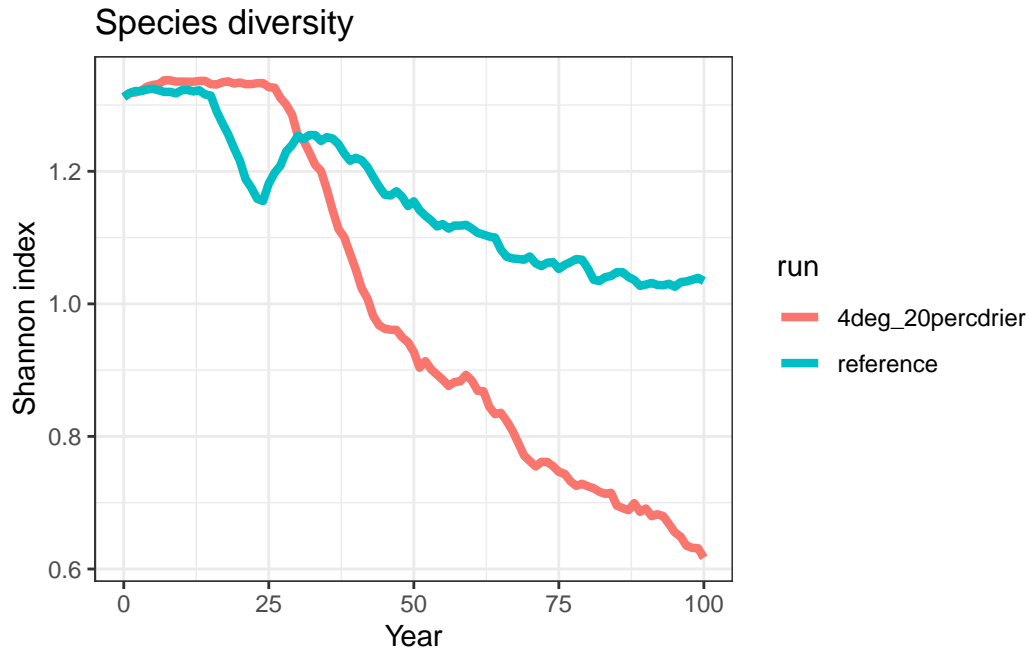
We merge them all together and make a plot based on Shannon index.

```
tlong2 <- tree2 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

div2 <- data.frame(adiv::speciesdiv(tlong2)) |>
  dplyr::mutate(year = unique(tree2$year), run = name2)

div <- rbind(div1, div2)

ggplot2::ggplot(div, ggplot2::aes(year, Shannon , color = run)) +
  ggplot2::geom_line(lwd = 1.5) +
  ggplot2::ggtitle("Species diversity") +
  ggplot2::labs(x = "Year", y = "Shannon index") +
  ggplot2::theme_bw()
```



We see that the values of Shannon index are first stable and then tend to decrease, while the reduction under the 4deg_20percdrier is more substantial than under the reference scenario. Shannon index is calculated as follows:

$$S = - \sum (p_i * \log(p_i))$$

Where p_i is the relative abundance of the species i , calculated as the ratio of the abundance of the species i and the abundance of all species. Shannon-index is 0 when we have only one species at the stand.

2.2.6.5 Spatial diversity

We can calculate some additional biodiversity indices targeting spatial diversity using the *treepat* package developed by Francesco Chianucci (fchianucci@gmail.com). You can find a short description of the package here: [“documents/treepat_package.pdf”](#)

You can install the package using devtools:

```
devtools::install_gitlab('fchianucci/treepat')
```

Then, we are ready to use it! Let's calculate these two indices:

2 Process based

- Diameter differentiation (Gadow, 1993): Spatial size inequality defined as the mean of the ratio of smaller and larger plant sizes in the nearest neighbors of a tree. The value of the index increases with increasing average size difference between neighboring trees. 0 is implying that neighboring trees have equal size.
- Mingling (Aguirre et al., 2003): One very intuitive extension of taxonomic species diversity (either richness or abundance) is considering spatial mingling, namely how plants of the same (con-specific neighbors) or different (hetero-specific neighbors) species are arranged in space. The mingling index calculates the proportion of the k nearest neighbors that do not belong to the same species as the reference tree. For example, with four neighbors, the mingling attribute can assume five values, ranging from 0 (all trees are of the same species) to 1 (all trees belong to different species).

Calculate the indices based on the reference run, then based on the scenario. For each index we add extra columns with the index name, and run name for further analyses. The max.k parameter is telling how many neighboring trees we want to account for.

```
# diameter differentiation:
# mingling
diff <-
  data.frame(
    treespat::DIFF(
      tree1,
      .x = x,
      .y = y,
      .mark = dbh,
      xmax = 100,
      ymax = 100,
      max.k = 4,
      shape = 'square',
      .groups = c('year')
    ) |>
    dplyr::mutate(index = "DIFF", name = "Diameter differentiation", run =
      name1)
  )
head(diff)
```

	year	DIFF index	name	run
1	0	0.4317331	DIFF Diameter differentiation	reference
2	1	0.4104274	DIFF Diameter differentiation	reference
3	2	0.3988705	DIFF Diameter differentiation	reference
4	3	0.3903934	DIFF Diameter differentiation	reference
5	4	0.3775628	DIFF Diameter differentiation	reference

6 5 0.3595558 DIFF Diameter differentiation reference

```
ming <-
  data.frame(
    treespat::MING(
      tree1,
      .x = x,
      .y = y,
      .species = species,
      xmax = 100,
      ymax = 100,
      max.k = 4,
      shape = 'square',
      .groups = c('year')
    ) |>
    dplyr::mutate(index = "MING", name = "Mingling", run = name1)
  )

indices1 <- rbind(diff |> dplyr::rename(value = DIFF),
  ming |> dplyr::rename(value = MINGLING))
```

We calculated each index as separate variable and then merged them into indices1. Here I renamed the column names to have it all “value”, for easier plotting later on. (each line has the information on which run and which index is the value) We do the same for the scenario simulation, we merge them together and make the plotting.

```
diff <- data.frame(treespat::DIFF(tree2, .x = x, .y = y, .mark = dbh,
  xmax = 100, ymax = 100, max.k = 4, shape = 'square',
  .groups = c('year'))) |>
  dplyr::mutate(index = "DIFF", name = "Diameter differentiation",
    run = name2))

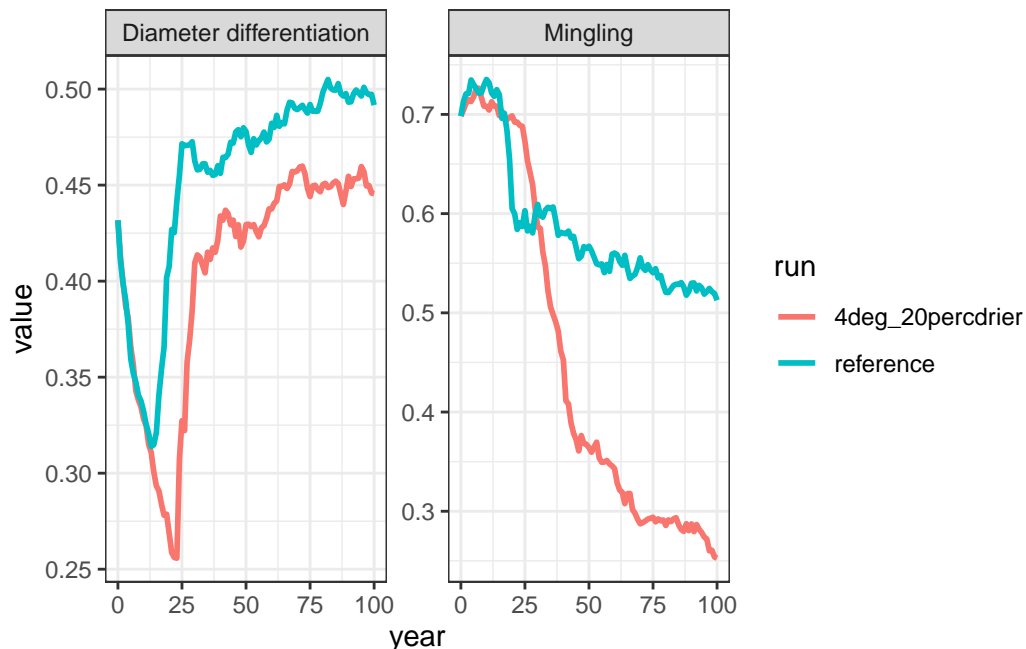
ming <- data.frame(treespat::MING(tree2, .x = x, .y = y, .species = species,
  xmax = 100, ymax = 100, max.k = 4, shape = 'square', .groups = c('year'))) |>
  dplyr::mutate(index = "MING", name = "Mingling", run = name2))

indices2 <- rbind(diff |> dplyr::rename(value = DIFF),
  ming |> dplyr::rename(value = MINGLING))

indices <- rbind(indices1, indices2)

ggplot2::ggplot(indices, ggplot2::aes(x = year, y = value, color = run)) +
  ggplot2::geom_line(lwd = 1) +
```

```
ggplot2::facet_wrap( ~ name, ncol = 2, scales = "free") +
ggplot2::theme_bw()
```



The diameter differentiation increases with the increasing average difference in diameter between neighboring trees. The value 0 indicates that neighboring trees have equal diameters. The development shows that the index values decrease during the first 20/25 years, after which it steeply increases due to the occurrence of ingrowth and then the values are stabilized. The values for the reference climate scenario are slightly higher than for the climate change scenario indicating slower growth under drier and warmer conditions and/or the differences in tree species composition (proportion of tree species). The mingling index is studying the neighboring trees regarding their species. In both development there is a decrease in the index after the occurrence of regeneration. Using the climate change scenario the index shows steeper decrease.

2.2.7 Question B - Group 1

- How are the species distribution and total living biomass C content changing in time on Plot 1? Compare 0 year and 100 year status in the reference case and in the case of your scenario(s)!

Run the model as it is described in the previous chapters to have at least 2 simulations completed: one with reference conditions and one scenario with some changes in the

environment. Check if you have the output files in the output folder: *iLand_simulations/output/*. If you manage to simulate 2 scenarios, you can expand the code to have file3, and 3rd from each variable, or make the comparison in 2 step always comparing one scenario to the reference conditions at the time. Open the *summerSchoolExercise.Rproj* project in R studio and start working there.

2.2.7.1 Read the landscape output table

Read in the *landscape* output table from two outputs you have for your plot:

```
path1 <- "model/iLand_simulations/output/Output_plot1.sqlite"
path2 <-
  "model/iLand_simulations/output/Output_plot1_4deg_20percdrier.sqlite"

file1 <- here::here(path1)
file2 <- here::here(path2)

# Give some name for the three simulations.
name1 <- "reference"
name2 <- "4deg_20percdrier"

# Read in data using the RSQLite package
sqlite.driver <- RSQLite::dbDriver("SQLite")

# We will work with "landscape" table and tree-scale data:
db1 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file1) # connect to the file1
landscape1 <- RSQLite::dbReadTable(db1, "landscape")
RSQLite::dbDisconnect(db1) # disconnect to the file1

db2 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file2) # connect to the file2
landscape2 <- RSQLite::dbReadTable(db2, "landscape")
RSQLite::dbDisconnect(db2)
```

Merge the data from the three files together and make a column which tells which comes from which simulation, and study the table. The column “run” will support the plotting where we can use facets in ggplot.

```
landscape <- rbind(landscape1 |> dplyr::mutate(run = name1),
  landscape2 |> dplyr::mutate(run = name2))
```

2 Process based

```
head(landscape)
```

	year	area	area_100m	species	count_ha	dbh_avg_cm	height_avg_m	volume_m3
1	0	1	1	abal	68	12.55456	10.04365	4.33641
2	0	1	1	fasy	50	49.86496	28.33702	136.89579
3	0	1	1	piab	120	32.73141	22.00713	150.34005
4	0	1	1	pisy	49	50.45049	28.66693	126.09230
5	1	1	1	abal	68	14.15550	11.39836	6.12901
6	1	1	1	fasy	49	50.95480	28.84830	141.11347

	total_carbon_kg	gwl_m3	basal_area_m2	NPP_kg	NPPabove_kg	LAI
1	2264.490	4.33641	0.8659201	0.000	0.000	0.09848037
2	49667.453	136.89579	10.2322163	0.000	0.000	0.94517453
3	45424.641	150.34005	12.1157843	0.000	0.000	1.22598624
4	42580.564	126.09230	9.9010286	0.000	0.000	1.13237721
5	2784.089	6.12901	1.0937620	1657.542	1128.888	0.11701639
6	52091.802	143.51049	10.4297469	9855.392	6472.199	0.96978453

	cohort_count_ha	run
1	0	reference
2	0	reference
3	0	reference
4	0	reference
5	31	reference
6	54	reference

2.2.7.2 Visualize changes in time

We can see that the output is given for each year and each species in our 1ha area. Plot the living carbon (total_carbon_kg) in time, coloring by species. We give a unified species coloring in the beginning. Plot number of trees and mean diameter. Carbon and number of trees can be plotted in an additive way, but for mean dbh we do line plot per species.

```
cols.all=c( "rops"="#e0e0e0","acpl"="#A9A9A9","alin"="#696969","alvi"="#2e2e2e",
  "bepe"="#fadbfb","casa"="#7eadfb","coav"="#20c6b6","tipl"="#645394",
  "ulgl"="#311432","saca"="#D8BFD8","soar"="#DDA0DD","soau"="#BA55D3",
  "pice"="#D27D2D","pini"="#a81c07","algl"="#2ECBE9","tico"="#128FC8",
  "potr"="#00468B","poni"="#5BAEB7","frex"="#fe9cb5","cabe"="#fe6181",
  "acps"="#fe223e","lade"="#FFFE71","abal"="#FFD800","pisy"="#A4DE02",
  "fasy"="#76BA1B","piab"="#006600","quro"="#FF7F00","qupe"="#FF9900",
  "qupu"="#CC9900")
```

2 Process based

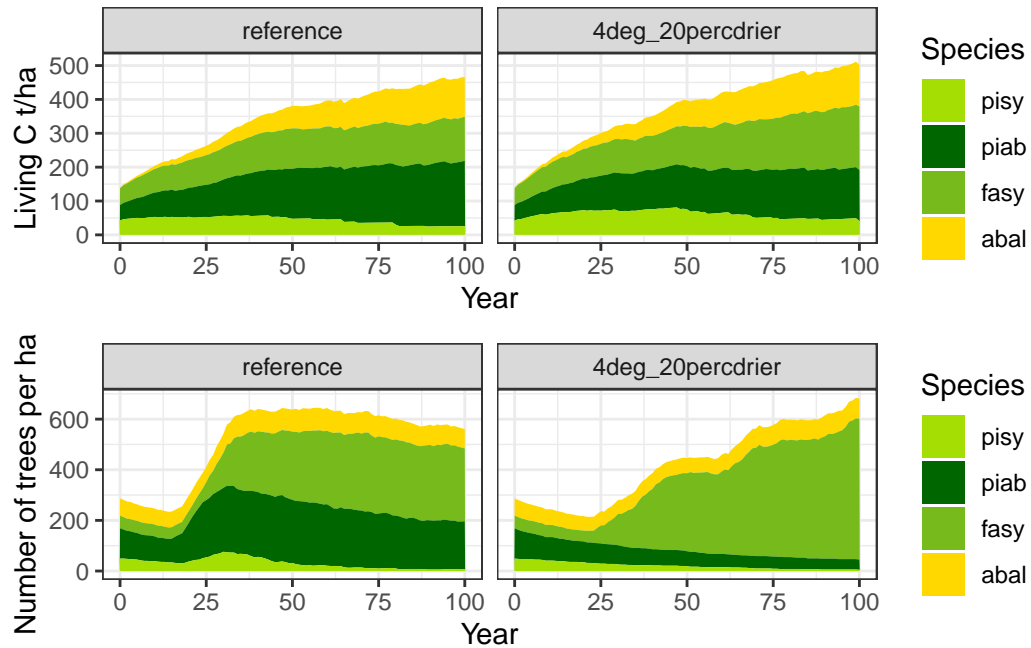
```
# We set here the order of the run categories for the table, to have the first
#run first and then the second. (left-right of the plots)
landscape$run <- factor(landscape$run, levels = c(name1, name2))

# Plot the living carbon content, to have tonnes/ha, we divide total_carbon_kg by 1000.
g1 <- ggplot2::ggplot(landscape, ggplot2::aes(year, total_carbon_kg / 1000 ,
                                              fill = factor(species))) +
  ggplot2::geom_area() +
  ggplot2::scale_fill_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Living C t/ha", fill = "Species") +
  ggplot2::theme_bw()

g2 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, count_ha , fill = factor(species))) +
  ggplot2::geom_area() +
  ggplot2::scale_fill_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Number of trees per ha", fill = "Species") +
  ggplot2::theme_bw()

gridExtra::grid.arrange(g1, g2, ncol = 1)
```

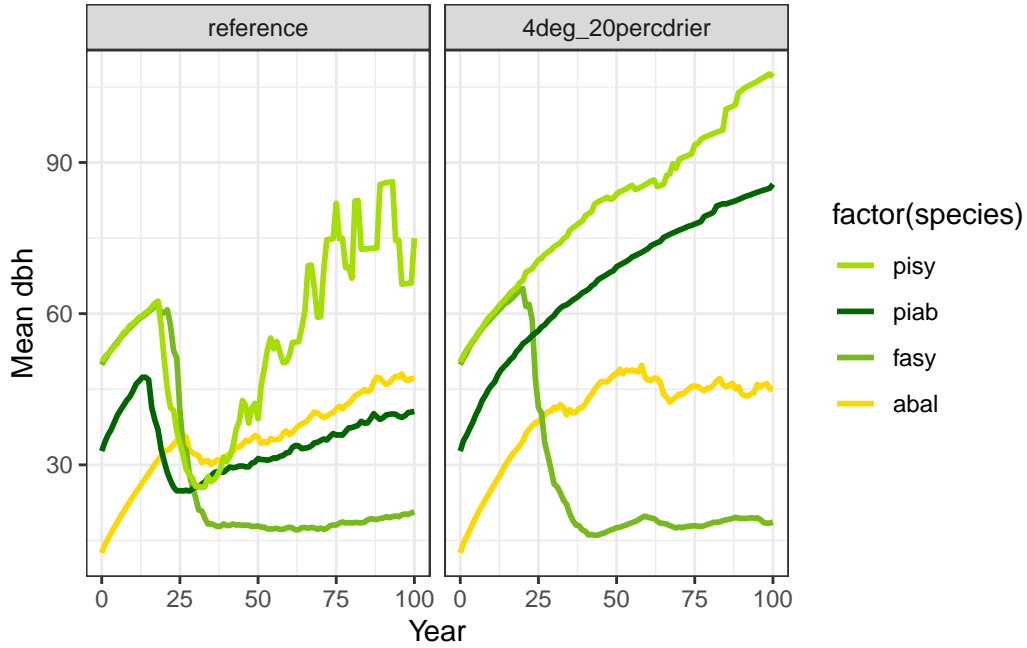
2 Process based



```
g3 <- ggplot2::ggplot(landscape,
                      ggplot2::aes(year, dbh_avg_cm , color = factor(species))) +
  ggplot2::geom_line(lwd = 1) +
  ggplot2::scale_color_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Mean dbh", fill = "Species") +
  ggplot2::theme_bw()

print(g3)
```

2 Process based



Note that we do not have management intervention, and in the model trees shorter than 4m are not included in these outputs. However as they grow taller than 4m, model start to handle them as individual trees and including in these outputs which we are looking now.

At the end of the simulated 100 years, the total simulated carbon stock in living biomass under climate change scenario was slightly higher than under the reference climate. The proportion of beech and pine increased under drier and warmer climate, while the proportion of spruce was lower, since spruce is in general more susceptible to dry conditions. The changes in the number of trees of individual tree species were more substantial than the changes in carbon stock. The number of spruce trees was substantially reduced under climate change, while under reference climate it was doubled after 30 years and then remain stable until the end of simulation. In contrast, the number of beech trees increased under both scenarios, but the increase was more pronounced under climate change scenario due to the reduction of spruce trees.

Mean dbh of individual species reflected the development of the number of trees. The increase in the number of trees caused the reduction of mean dbh, e.g. of beech under both scenarios between 20 and 35 simulation years. The species-specific graphs for mean dbh indicates that some species do not have successful regeneration under the climate change scenario. Spruce and pine dbh decreased only under reference scenario, when we observed the increase of the number of trees, while under climate change scenario the mean dbh of spruce or pine was growing indicating the growth of remaining trees of the species at the plot. In the case of fir, we observed the continuous increase of dbh due

to the more or less stable number of trees at the plots. While the mean dbh of fir was growing during the whole simulation period, under drier and warmer climate mean dbh of fir did not substantially change after 70 years.

2.2.7.3 Assess the stored carbon amount

Calculate the stored C amount in the initial year (year==0) and the last year (year==100)!

```
livingC0 <-
  data.frame(
    landscape |> dplyr::filter(year == 0) |>
      dplyr::group_by(run) |>
      dplyr::summarize(sum.livingC = sum(total_carbon_kg / 1000))
  )
print(livingC0)
```

	run	sum.livingC
1	reference	139.9371
2	4deg_20percdrier	139.9371

```
livingC100 <-
  data.frame(
    landscape |> dplyr::filter(year == 100) |>
      dplyr::group_by(run) |>
      dplyr::summarize(sum.livingC = sum(total_carbon_kg / 1000))
  )
print(livingC100)
```

	run	sum.livingC
1	reference	465.9313
2	4deg_20percdrier	502.2416

The initial conditions are same for the runs, but they are ending up at different C levels.

Calculate the stored C amount PER SPECIES in the initial year (year==0) and the last year (year==100)


```
species.livingC0 <-
  data.frame(
    landscape |> dplyr::filter(year == 0) |>
      dplyr::group_by(run, species) |>
      dplyr::summarize(livingC = sum(total_carbon_kg / 1000))
  )
print(species.livingC0)
```

	run	species	livingC
1	reference	abal	2.26449
2	reference	fasy	49.66745
3	reference	piab	45.42464
4	reference	pisy	42.58056
5	4deg_20percdrier	abal	2.26449
6	4deg_20percdrier	fasy	49.66745
7	4deg_20percdrier	piab	45.42464
8	4deg_20percdrier	pisy	42.58056

```
species.livingC100 <-
  data.frame(
    landscape |> dplyr::filter(year == 100) |>
      dplyr::group_by(run, species) |>
      dplyr::summarize(livingC = sum(total_carbon_kg / 1000))
  )
print(species.livingC100)
```

	run	species	livingC
1	reference	abal	117.08641
2	reference	fasy	130.23479
3	reference	piab	192.96180
4	reference	pisy	25.64826
5	4deg_20percdrier	abal	123.25491
6	4deg_20percdrier	fasy	187.43320
7	4deg_20percdrier	piab	150.38585
8	4deg_20percdrier	pisy	41.16762

2.2.7.4 Assess the species proportions

Calculate the species proportions based on the stored C amount in the initial year (year==0) and the last year (year==100) For this we need the total C amount and the species-specific C amount.

2 Process based

```
LC0 <- dplyr::left_join(species.livingC0, livingC0, by = "run")
LC0 <-
  data.frame(LC0 |> dplyr::mutate(spec.prop = livingC / sum.livingC, year =
                                0))
print(LC0)
```

	run	species	livingC	sum.livingC	spec.prop	year
1	reference	abal	2.26449	139.9371	0.01618219	0
2	reference	fasy	49.66745	139.9371	0.35492686	0
3	reference	piab	45.42464	139.9371	0.32460745	0
4	reference	pisy	42.58056	139.9371	0.30428349	0
5	4deg_20percdrier	abal	2.26449	139.9371	0.01618219	0
6	4deg_20percdrier	fasy	49.66745	139.9371	0.35492686	0
7	4deg_20percdrier	piab	45.42464	139.9371	0.32460745	0
8	4deg_20percdrier	pisy	42.58056	139.9371	0.30428349	0

```
LC100 <- dplyr::left_join(species.livingC100, livingC100, by = "run")
LC100 <-
  data.frame(LC100 |> dplyr::mutate(spec.prop = livingC / sum.livingC, year =
                                   100))
print(LC100)
```

	run	species	livingC	sum.livingC	spec.prop	year
1	reference	abal	117.08641	465.9313	0.25129545	100
2	reference	fasy	130.23479	465.9313	0.27951504	100
3	reference	piab	192.96180	465.9313	0.41414220	100
4	reference	pisy	25.64826	465.9313	0.05504730	100
5	4deg_20percdrier	abal	123.25491	502.2416	0.24540961	100
6	4deg_20percdrier	fasy	187.43320	502.2416	0.37319331	100
7	4deg_20percdrier	piab	150.38585	502.2416	0.29942930	100
8	4deg_20percdrier	pisy	41.16762	502.2416	0.08196777	100

Put the two tables together and visualize the results!

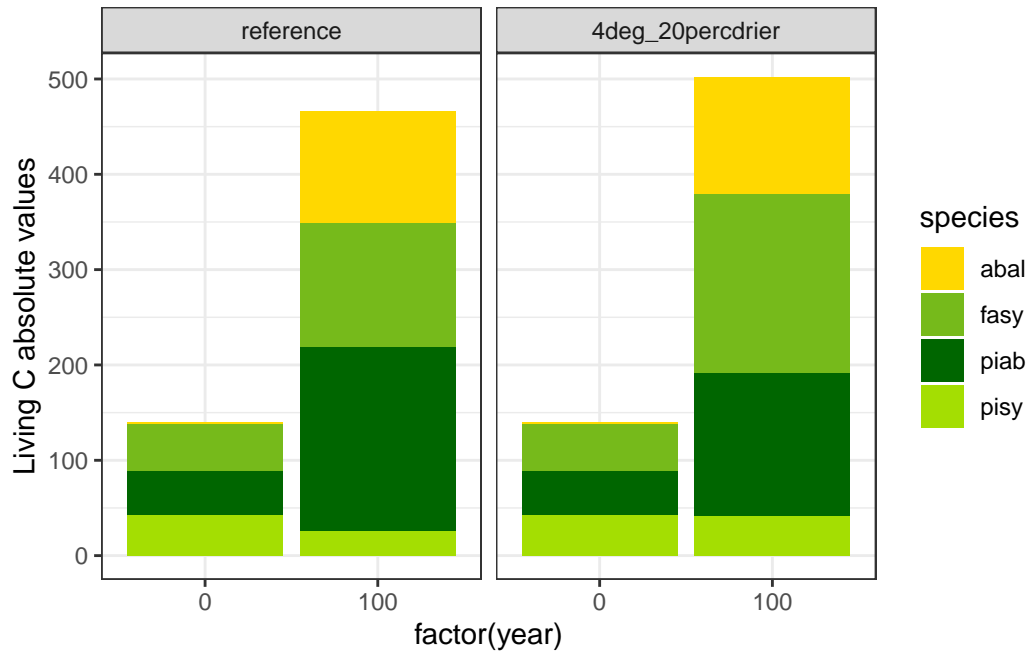
```
LC <- rbind(LC0, LC100)

LC$run <- factor(LC$run, levels = c(name1, name2))

ggplot2::ggplot(LC, ggplot2::aes(fill = species, y = livingC, x = factor(year))) +
  ggplot2::geom_bar(position = "stack", stat = "identity") +
  ggplot2::scale_fill_manual(values = cols.all) +
```

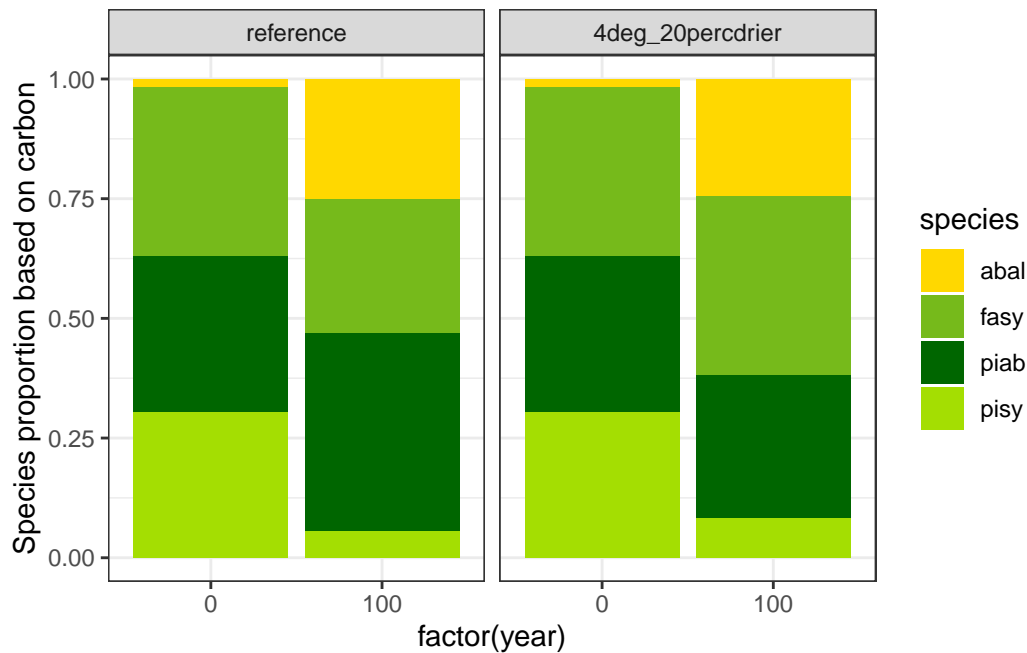
2 Process based

```
ggplot2::facet_wrap( ~ run) +  
ggplot2::ylab("Living C absolute values") +  
ggplot2::theme_bw()
```



```
ggplot2::ggplot(LC, ggplot2::aes(fill = species, y = spec.prop,  
                                x = factor(year))) +  
ggplot2::geom_bar(position = "stack", stat = "identity") +  
ggplot2::scale_fill_manual(values = cols.all) +  
ggplot2::facet_wrap( ~ run) +  
ggplot2::ylab("Species proportion based on carbon") +  
ggplot2::theme_bw()
```

2 Process based



The living C stock tripled over 100 years due to the accumulation of biomass in trees and no management interventions. Moreover, the proportion of individual tree species changed with fir substantially increasing its share. The living C stock of pine was reduced under the reference climate or remained unchanged under the climate change scenario.

2.2.8 Question A - Group 2

- How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 2?

Run the model as it is described in the previous chapters to have at least 2 simulations completed: one with reference conditions and one scenario with some changes in the environment. Check if you have the output files in the output folder: *iLand_simulations/output/*. If you manage to simulate 2 scenarios, you can expand the code to have file3, and 3rd from each variable, or make the comparison in 2 step always comparing one scenario to the reference conditions at the time.

Open the *summerSchoolExercise.Rproj* project in R studio and start working there.

2.2.8.1 Read the tree output table

Read in the *tree* output table from two outputs you have for your plot:

2 Process based

```
path1 <- "model/iLand_simulations/output/Output_plot2.sqlite"
path2 <-
  "model/iLand_simulations/output/Output_plot2_4deg_20percdrier.sqlite"

file1 <- here::here(path1)
file2 <- here::here(path2)

# Give some name for the three simulations.
name1 <- "reference"
name2 <- "4deg_20percdrier"

# Read in data using the RSQLite package
sqlite.driver <- RSQLite::dbDriver("SQLite")
db1 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file1) # connect to the file1

tables.in.the.file <-
  RSQLite::dbListTables(db1) # explore the tables in the file1
print(tables.in.the.file)
```

```
[1] "carbon"           "carbonflow"       "dynamicstand"
[4] "landscape"        "landscape_removed" "runinfo"
[7] "stand"            "tree"
```

```
# We will work with "tree" table and tree-scale data:
tree1 <- RSQLite::dbReadTable(db1, "tree")
RSQLite::dbDisconnect(db1) # disconnect to the file1

# READ IN DATA FROM THE SECOND FILE: -----
db2 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file2) # connect to the file2
tree2 <- RSQLite::dbReadTable(db2, "tree")
RSQLite::dbDisconnect(db2)
```

Merge the data from the two files together and make a column which tells which comes from which simulation, and study the table. The column “run” will support the plotting where we can use facets in ggplot.

```
tree <- rbind(tree1 |> dplyr::mutate(run = name1) ,
              tree2 |> dplyr::mutate(run = name2))
```

```
head(tree)
```

	year	ru	rid	species	id	x	y	dbh	height	basalArea	volume_m3	age
1	0	0	1	piab	1	65	43	85.63804	57.57916	0.5760011	14.029073	616
2	0	0	1	qupe	2	9	47	78.57278	40.23281	0.4848798	8.856667	1072
3	0	0	1	piab	3	95	39	78.37173	52.69362	0.4824016	10.752442	564
4	0	0	1	piab	4	95	91	76.88074	51.69115	0.4642212	10.150361	553
5	0	0	1	piab	5	79	75	74.49680	50.08829	0.4358782	9.235103	536
6	0	0	1	qupe	6	89	49	71.96641	36.85005	0.4067706	6.805241	982
	leafArea_m2	foliageMass	stemMass	branchMass	fineRootMass	coarseRootMass						
1	420.3857	98.91427	3815.740	613.1350	74.18570	986.7341						
2	394.8285	41.56089	3702.413	365.5260	31.17067	1160.0538						
3	366.0804	86.13657	3067.633	500.0222	64.60242	770.4860						
4	355.2738	83.59383	2925.986	478.4131	62.69537	730.2825						
5	338.2381	79.58543	2707.714	444.9787	59.68908	668.8427						
6	340.0686	35.79670	3147.565	311.7678	26.84752	936.2913						
	lri	lightResponse	stressIndex	reserve_kg	treeFlags	run						
1	0.4567858		0	0	173.09998	0 reference						
2	0.2441241		0	0	72.73156	0 reference						
3	1.0000000		0	0	150.73898	0 reference						
4	1.0000000		0	0	146.28922	0 reference						
5	0.4540371		0	0	139.27451	0 reference						
6	0.2951740		0	0	62.64423	0 reference						

2.2.8.2 Visualize trees in year 0 and year 100!

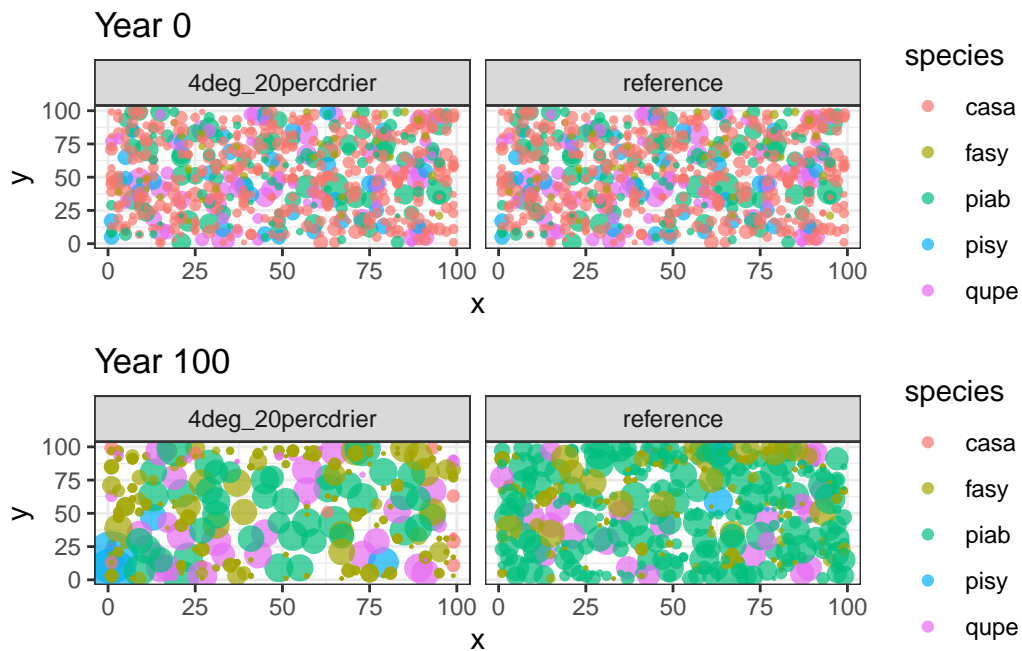
To explore a bit the simulation, plot the tree locations at the beginning and the end of the simulation! For plotting we use the colors for the species, and the size of a circles to show dbh differences. Here we divided dbh with the value 20 just for visualization, not to have too huge circles shading each others completely, but still see the trees.

```
tree0 <- tree |> dplyr::filter(year == 0)
g1 <-
  ggplot2::ggplot(tree0, ggplot2::aes(x = x, y = y, color = species)) +
  ggplot2::geom_point(size = tree0$dbh / 20, alpha = 0.7) +
  ggplot2::ggtitle("Year 0") +
  ggplot2::facet_wrap(~ run) + ggplot2::theme_bw()

tree100 <- tree |> dplyr::filter(year == 100)
g2 <-
```

2 Process based

```
ggplot2::ggplot(tree100, ggplot2::aes(x = x, y = y, color = species)) +  
  ggplot2::geom_point(size = tree100$dbh / 20, alpha = 0.7) +  
  ggplot2::ggtitle("Year 100") +  
  ggplot2::facet_wrap(~ run) + ggplot2::theme_bw()  
  
gridExtra::grid.arrange(g1, g2, ncol = 1)
```



We can see that the species composition has changed and the dbh of the trees seems to be larger, than in the initial year. We can even identify the same trees at the same location.

2.2.8.3 Visualize some changes in time!

Visualize some changes in time, for example the mean diameter of the trees!

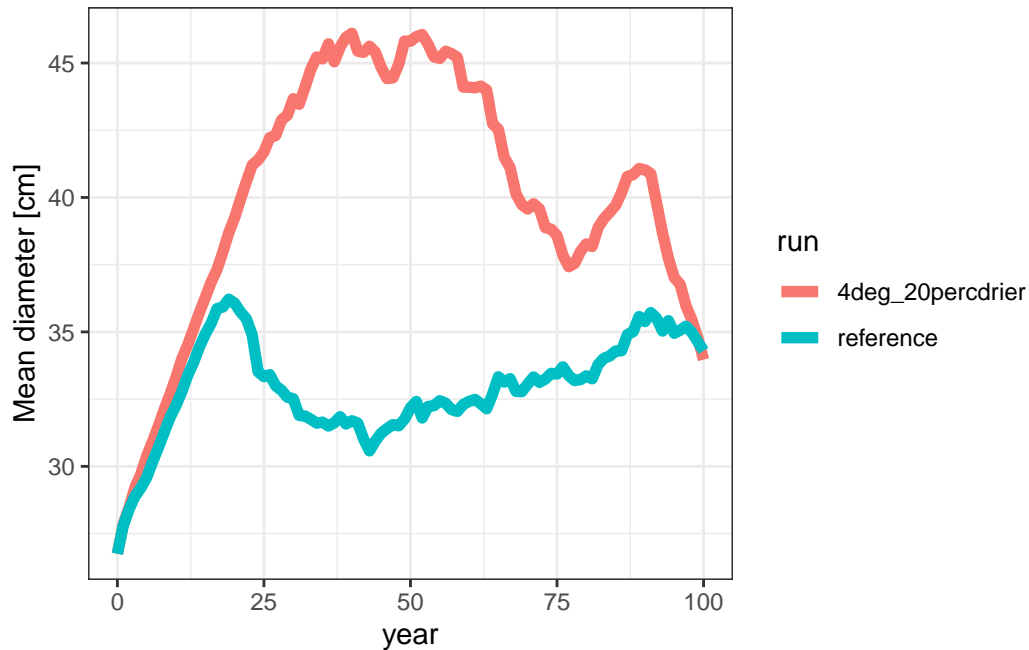
```
sum.table <- tree |> dplyr::group_by(year, run) |>  
  dplyr::summarise(  
    N = dplyr::n(),  
    MD = mean(dbh, na.rm = TRUE),  
    #mean diameter  
    BA = sum(basalArea)
```

```

) #basal area

ggplot2::ggplot(sum.table, ggplot2::aes(x = year, y = MD, color = run)) +
  ggplot2::geom_line(lwd = 2) +
  ggplot2::ylab("Mean diameter [cm]") + ggplot2::theme_bw()

```



We can see that there is an initial increase in mean dbh, then a drop after 20/25 years under reference conditions. Under the climate change scenario, the mean dbh is growing for longer time and has the drop after 60 years. The dbh in the last year are very similar however the development of the simulations were different. The drop is due to the growing regeneration layer that is produced by the seeds of the existing trees. In the initial year we do not have regeneration layer in the simulations (it is possible to put there, but we do not have it now). And until the small trees grow up to higher than 4 m they are not appearing in the individual *tree* output as they are handled in cohorts.

2.2.8.4 Species diversity

To study biodiversity aspects, we can calculate biodiversity indicators. Let's use the *adiv* package for species diversity. First we need to change the database structure to have the number of trees for each species in different columns (*pivot_wider*) without any additional columns for the *adiv* package *speciesdiv* function. We work first only

2 Process based

with tree data from one simulation (tree1). You can find the documentation of the adiv package here: “[documents/adiv.pdf](#)”

```
# First we need to change the
tlong1 <- tree1 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

head(tlong1)
```

```
# A tibble: 6 x 5
  casa fasy piab pisy qupe
<int> <int> <int> <int> <int>
1   385   34  118   32   68
2   358   34  117   30   64
3   335   33  115   30   63
4   319   33  114   30   63
5   310   32  112   29   63
6   303   31  112   29   62
```

Then we apply the “speciesdiv” function and we add back the year and run columns to have the information.

```
div1 <- data.frame(adiv::speciesdiv(tlong1)) |>
  dplyr::mutate(year = unique(tree1$year), run = name1)
head(div1)
```

```
  richness GiniSimpson Simpson Shannon Margalef Menhinick McIntosh year
1         5    0.5836227 2.401668 1.162161 0.6195048 0.1981072 0.3693616    0
2         5    0.5929556 2.456735 1.177198 0.6248128 0.2036157 0.3773673    1
3         5    0.6039255 2.524778 1.196665 0.6293160 0.2083333 0.3867710    2
4         5    0.6136885 2.588585 1.214413 0.6322962 0.2114775 0.3951745    3
5         5    0.6159951 2.604134 1.217670 0.6346568 0.2139802 0.3973226    4
6         5    0.6185478 2.621560 1.221342 0.6363349 0.2157659 0.3996270    5
run
```

2 Process based

1 reference
2 reference
3 reference
4 reference
5 reference
6 reference

We make the same steps for the other simulation results (tree2).

```
tlong2 <- tree2 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

div2 <- data.frame(adiv::speciesdiv(tlong2)) |>
  dplyr::mutate(year = unique(tree2$year), run = name2)
```

We merge them all together and make a plot based on Shannon index.

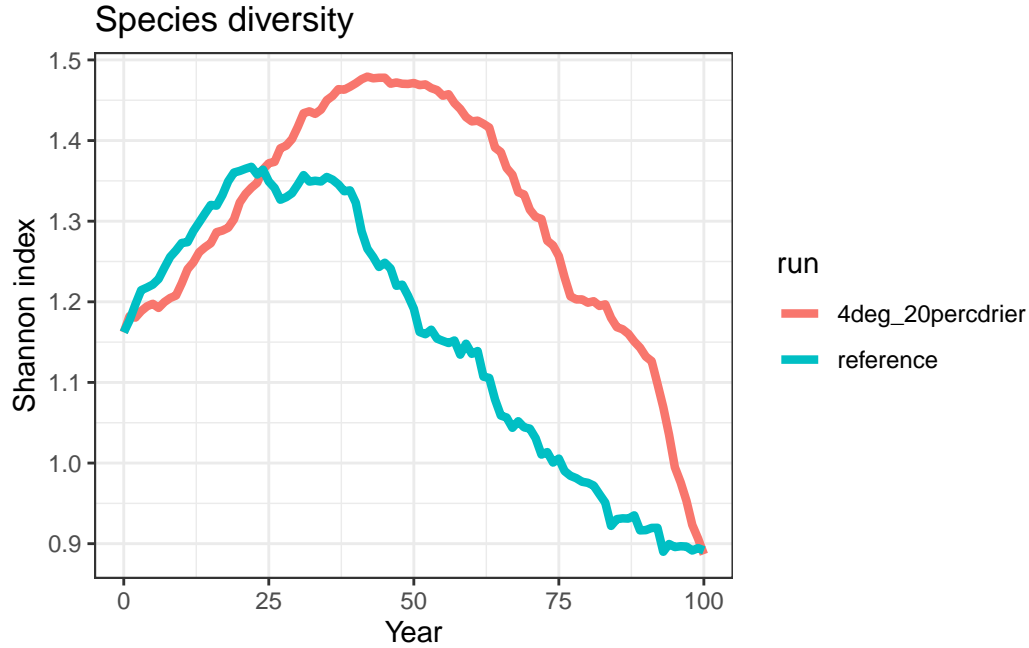
```
tlong2 <- tree2 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

div2 <- data.frame(adiv::speciesdiv(tlong2)) |>
  dplyr::mutate(year = unique(tree2$year), run = name2)

div <- rbind(div1, div2)

ggplot2::ggplot(div, ggplot2::aes(year, Shannon , color = run)) +
```

```
ggplot2::geom_line(lwd = 1.5) +
ggplot2::ggtitle("Species diversity") +
ggplot2::labs(x = "Year", y = "Shannon index") +
ggplot2::theme_bw()
```



We see that the values of Shannon index are first increasing and then tend to decrease, while the reduction under the 4deg_20percdrier is starting later than under the reference scenario. Shannon index is calculated as follows:

$$S = - \sum (p_i * \log(p_i))$$

Where p_i is the relative abundance of the species i , calculated as the ratio of the abundance of the species i and the abundance of all species. Shannon-index is 0 when we have only one species at the stand.

2.2.8.5 Spatial diversity

We can calculate some additional biodiversity indices targeting spatial diversity using the *treemap* package developed by Francesco Chianucci (fchianucci@gmail.com). You can find a short description of the package here: “[documents/treemap_package.pdf](#)”

You can install the package using devtools:

2 Process based

```
devtools::install_gitlab('fchianucci/treespat')
```

Then, we are ready to use it! Let's calculate these two indices:

- Diameter differentiation (Gadow, 1993): Spatial size inequality defined as the mean of the ratio of smaller and larger plant sizes in the nearest neighbors of a tree. The value of the index increases with increasing average size difference between neighboring trees. 0 is implying that neighboring trees have equal size.
- Mingling (Aguirre et al., 2003): One very intuitive extension of taxonomic species diversity (either richness or abundance) is considering spatial mingling, namely how plants of the same (con-specific neighbors) or different (hetero-specific neighbors) species are arranged in space. The mingling index calculates the proportion of the k nearest neighbors that do not belong to the same species as the reference tree. For example, with four neighbors, the mingling attribute can assume five values, ranging from 0 (all trees are of the same species) to 1 (all trees belong to different species).

Calculate the indices based on the reference run, then based on the scenario. For each index we add extra columns with the index name, and run name for further analyses. The max.k parameter is telling how many neighboring trees we want to account for.

```
library(treespat)

# diameter differentiation:
# mingling

diff <- data.frame(treespat::DIFF(tree1, .x = x, .y = y, .mark = dbh, xmax = 100,
                                ymax = 100, max.k = 4, shape='square', .groups=c('year')) |>
              dplyr::mutate(index="DIFF", name="Diameter differentiation", run=name1))
head(diff)
```

	year	DIFF index		name	run
1	0	0.4027813	DIFF	Diameter differentiation	reference
2	1	0.3793126	DIFF	Diameter differentiation	reference
3	2	0.3728755	DIFF	Diameter differentiation	reference
4	3	0.3672772	DIFF	Diameter differentiation	reference
5	4	0.3621167	DIFF	Diameter differentiation	reference
6	5	0.3617524	DIFF	Diameter differentiation	reference

2 Process based

```
ming <- data.frame(treespat::MING(tree1, .x = x, .y = y, .species = species,
  xmax = 100, ymax = 100, max.k = 4, shape='square', .groups=c('year')) |>
  dplyr::mutate(index="MING", name="Mingling", run=name1))

indices1<-rbind(diff |> dplyr::rename(value=DIFF),
  ming |> dplyr::rename(value=MINGLING ) )
```

We calculated each index as separate variable and then merged them into indices1. Here I renamed the column names to have it all “value”, for easier plotting later on. (each line has the information on which run and which index is the value) We do the same for the scenario simulation, we merge them together and make the plotting.

```
diff <- data.frame(treespat::DIFF(tree2, .x = x, .y = y, .mark = dbh,
  xmax = 100, ymax = 100, max.k = 4,
  shape = 'square', .groups = c('year')) |>
  dplyr::mutate(index = "DIFF",
    name = "Diameter differentiation", run = name2))

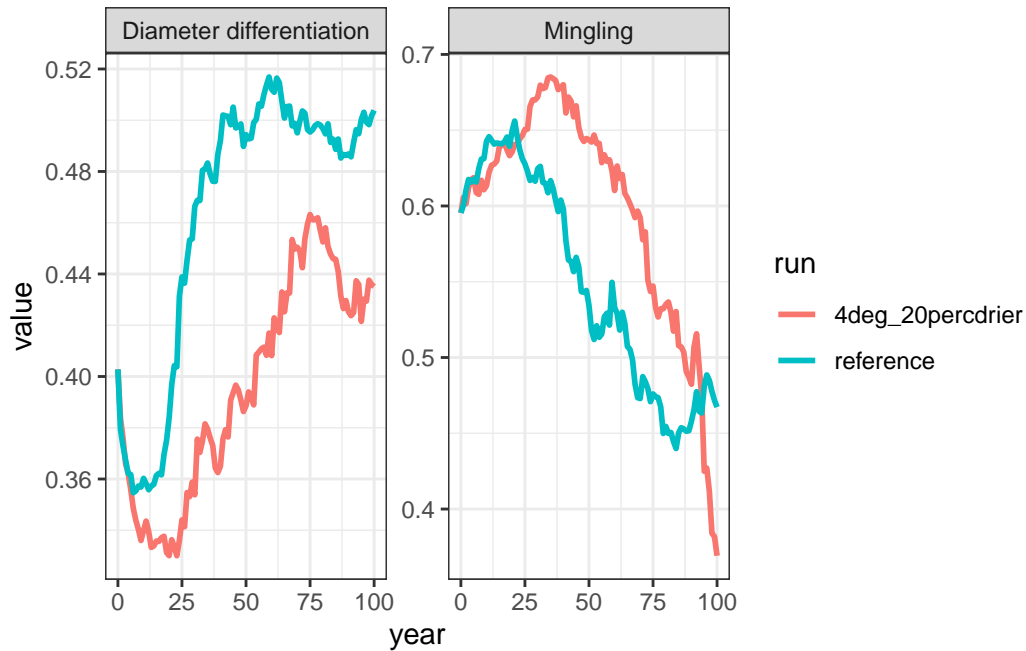
ming <- data.frame(treespat::MING(tree2, .x = x, .y = y, .species = species,
  xmax = 100, ymax = 100, max.k = 4, shape = 'square', .groups = c('year')) |>
  dplyr::mutate(index = "MING", name = "Mingling", run = name2))

indices2 <- rbind(diff |> dplyr::rename(value = DIFF),
  ming |> dplyr::rename(value = MINGLING ) )

indices <- rbind(indices1, indices2)

ggplot2::ggplot(indices, ggplot2::aes(x = year, y = value, color = run)) +
  ggplot2::geom_line(lwd = 1) +
  ggplot2::facet_wrap( ~ name, ncol = 2, scales = "free") +
  ggplot2::theme_bw()
```

2 Process based



The diameter differentiation increases with the increasing average difference in diameter between neighboring trees. The value 0 indicates that neighboring trees have equal diameters. The development shows that the index values decrease during the first circa 10 years, after which it steeply increases due to the occurrence of ingrowth and then the values are stabilized. The increase is not that steep for the climate change scenario that is in accordance with the mean dbh graph. The values for the reference climate scenario are higher than for the climate change scenario indicating slower growth under drier and warmer conditions and/or the differences in tree species composition (proportion of tree species). The mingling index is studying the neighboring trees regarding their species. In both development there is a decrease in the index after the occurrence of regeneration. Using the climate change scenario the index shows steeper decrease and ending up at lower values after 100 years.

2.2.9 Question B - Group 2

- How are the species distribution and total living biomass C content changing in time on Plot 2? Compare 0 year and 100 year status in the reference case and in the case of your scenario(s)!

Run the model as it is described in the previous chapters to have at least 2 simulations completed: one with reference conditions and one scenario with some changes in the environment. Check if you have the output files in the output folder: *iLand_simulations/output/*. If you manage to simulate 2 scenarios, you can expand the code to have

file3, and 3rd from each variable, or make the comparison in 2 step always comparing one scenario to the reference conditions at the time. Open the *summerSchoolExercise.Rproj* project in R studio and start working there.

2.2.9.1 Read the landscape output table

Read in the *landscape* output table from two outputs you have for your plot:

```
path1 <- "model/iLand_simulations/output/Output_plot2.sqlite"
path2 <-
  "model/iLand_simulations/output/Output_plot2_4deg_20percdrier.sqlite"

file1 <- here::here(path1)
file2 <- here::here(path2)

# Give some name for the three simulations.
name1 <- "reference"
name2 <- "4deg_20percdrier"

# Read in data using the RSQLite package
sqlite.driver <- RSQLite::dbDriver("SQLite")

# We will work with "landscape" table and tree-scale data:
db1 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file1) # connect to the file1
landscape1 <- RSQLite::dbReadTable(db1, "landscape")
RSQLite::dbDisconnect(db1) # disconnect to the file1

db2 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file2) # connect to the file2
landscape2 <- RSQLite::dbReadTable(db2, "landscape")
RSQLite::dbDisconnect(db2)
```

Merge the data from the three files together and make a column which tells which comes from which simulation, and study the table. The column “run” will support the plotting where we can use facets in ggplot.

```
landscape <- rbind(landscape1 |> dplyr::mutate(run = name1) ,
  landscape2 |> dplyr::mutate(run = name2))

head(landscape)
```

2 Process based

	year	area	area_100m	species	count_ha	dbh_avg_cm	height_avg_m	volume_m3	
1	0	1	1	casa	385	21.73276	18.052312	139.177831	
2	0	1	1	fasy	34	12.91628	7.340003	1.600727	
3	0	1	1	piab	118	34.75135	23.365242	215.003187	
4	0	1	1	pisy	32	42.76543	24.300133	49.013727	
5	0	1	1	qupe	68	40.62035	20.799453	118.797078	
6	1	1	1	casa	358	22.49524	18.644885	138.166090	
	total_carbon_kg		gwl_m3	basal_area_m2	NPP_kg	NPPabove_kg	LAI		
1	73246.102		139.177831	16.1066092	0.000	0.000	1.92032457		
2	1636.821		1.600727	0.4644299	0.000	0.000	0.06445078		
3	58083.681		215.003187	14.4589881	0.000	0.000	1.36948128		
4	18250.374		49.013727	4.6099225	0.000	0.000	0.51041807		
5	56880.735		118.797078	10.0244600	0.000	0.000	0.94624017		
6	73118.915		141.336043	15.8220002	8338.849	5192.833	1.87298787		
	cohort_count_ha		run						
1			0	reference					
2			0	reference					
3			0	reference					
4			0	reference					
5			0	reference					
6			0	reference					

2.2.9.2 Visualize changes in time

We can see that the output is given for each year and each species in our 1ha area. Plot the living carbon (total_carbon_kg) in time, coloring by species. We give a unified species coloring in the beginning. Plot number of trees and mean diameter. Carbon and number of trees can be plotted in an additive way, but for mean dbh we do line plot per species.

```
cols.all=c( "rops"="#e0e0e0", "acpl"="#A9A9A9", "alin"="#696969", "alvi"="#2e2e2e",
  "bepe"="#fADFAD", "casa"="#7eeadf", "coav"="#20c6b6", "tipl"="#645394",
  "ulgl"="#311432", "saca"="#D8BFD8", "soar"="#DDA0DD", "soau"="#BA55D3",
  "pice"="#D27D2D", "pini"="#a81c07", "algl"="#2ECBE9", "tico"="#128FC8",
  "potr"="#00468B", "poni"="#5BAEB7", "frex"="#fe9cb5", "cabe"="#fe6181",
  "acps"="#fe223e", "lade"="#FFFE71", "abal"="#FFD800", "pisy"="#A4DE02",
  "fasy"="#76BA1B", "piab"="#006600", "quro"="#FF7F00", "qupe"="#FF9900",
  "qupu"="#CC9900")

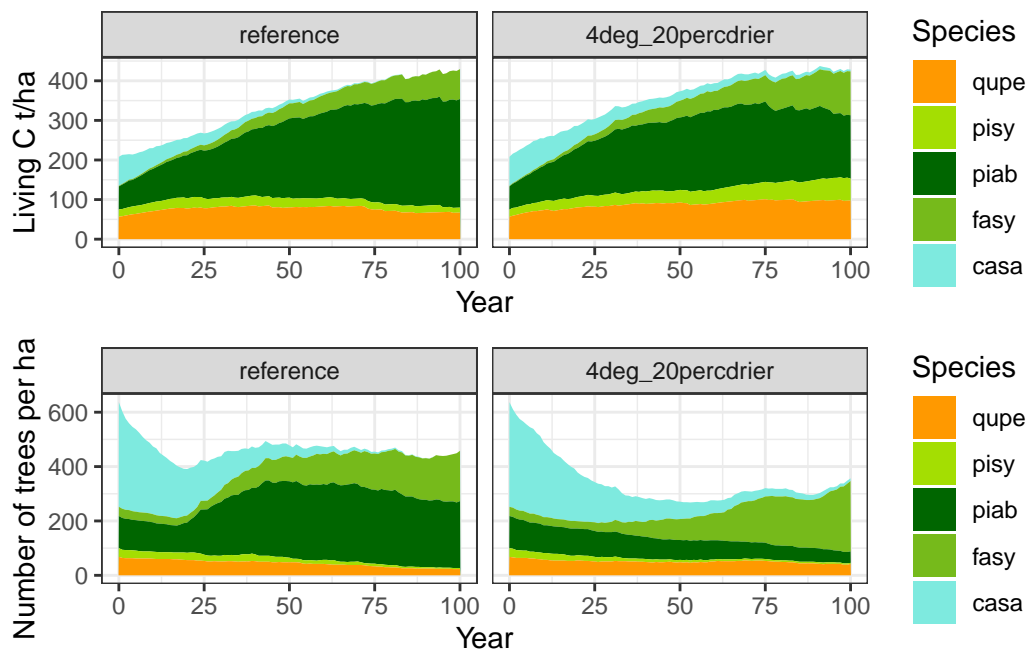
# We set here the order of the run categories for the table, to have the first
#run first and then the second. (left-right of the plots)
landscape$run <- factor(landscape$run, levels = c(name1, name2))
```


2 Process based

```
# Plot the living carbon content, to have tonnes/ha, we divide total_carbon_kg by 1000.
g1 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, total_carbon_kg / 1000 , fill = factor(
  ggplot2::geom_area() +
  ggplot2::scale_fill_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE))) +
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Living C t/ha", fill = "Species") +
  ggplot2::theme_bw()

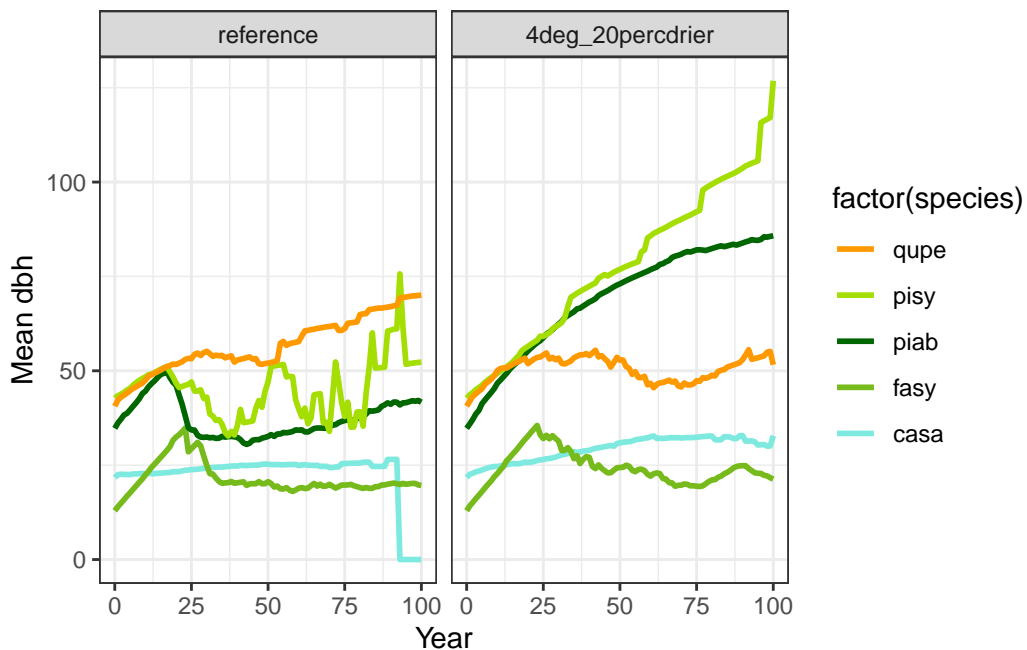
g2 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, count_ha , fill = factor(species))) +
  ggplot2::geom_area() +
  ggplot2::scale_fill_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Number of trees per ha", fill = "Species") +
  ggplot2::theme_bw()

gridExtra::grid.arrange(g1, g2, ncol = 1)
```



2 Process based

```
g3 <-
  ggplot2::ggplot(landscape,
    ggplot2::aes(year, dbh_avg_cm , color = factor(species))) +
  ggplot2::geom_line(lwd = 1) +
  ggplot2::scale_color_manual(values = cols.all,
    guide = ggplot2::guide_legend(reverse = TRUE)) +
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Mean dbh", fill = "Species") +
  ggplot2::theme_bw()
print(g3)
```



Note that we do not have management intervention, and in the model trees shorter than 4m are not included in these outputs. However as they grow taller than 4m, model start to handle them as individual trees and including in these outputs which we are looking now.

Castanea sativa gradually decreased during the simulation period under both scenarios. The reduction resulted from the mortality of chestnut trees, indicating their aging. By the end of the simulation period, chestnut almost completely disappeared from the stand due to the increased number of trees of other species at the plot, and hence higher inter-tree competition, especially under the reference climate, which hindered the chestnut regeneration and its survival. Mean dbh of chestnut is zero at the end of the simulation

period under the reference climate indicating that the species completely disappeared from the stand due to the high stand density, and inter-tree competition.

2.2.9.3 Assess the stored carbon amount

Calculate the stored C amount in the initial year (year==0) and the last year (year==100)!

```
livingC0 <- data.frame(
  landscape |> dplyr::filter(year == 0) |>
    dplyr::group_by(run) |>
    dplyr::summarize(sum.livingC = sum(total_carbon_kg / 1000))
)
print(livingC0)
```

```
      run sum.livingC
1      reference    208.0977
2 4deg_20percdrier    208.0977
```

```
livingC100 <- data.frame(
  landscape |> dplyr::filter(year == 100) |>
    dplyr::group_by(run) |>
    dplyr::summarize(sum.livingC = sum(total_carbon_kg / 1000))
)
print(livingC100)
```

```
      run sum.livingC
1      reference    429.5477
2 4deg_20percdrier    428.2694
```

The initial conditions are same for the runs, but they are ending up at different C levels.

Calculate the stored C amount PER SPECIES in the initial year (year==0) and the last year (year==100)

```
species.livingC0 <-
  data.frame(
    landscape |> dplyr::filter(year == 0) |>
      dplyr::group_by(run, species) |>
      dplyr::summarize(livingC = sum(total_carbon_kg / 1000))
  )
print(species.livingC0)
```

2 Process based

	run	species	livingC
1	reference	casa	73.246102
2	reference	fasy	1.636821
3	reference	piab	58.083681
4	reference	pisy	18.250374
5	reference	qupe	56.880735
6	4deg_20percdrier	casa	73.246102
7	4deg_20percdrier	fasy	1.636821
8	4deg_20percdrier	piab	58.083681
9	4deg_20percdrier	pisy	18.250374
10	4deg_20percdrier	qupe	56.880735

```
species.livingC100 <-  
  data.frame(  
    landscape |> dplyr::filter(year == 100) |>  
      dplyr::group_by(run, species) |>  
      dplyr::summarize(livingC = sum(total_carbon_kg / 1000))  
  )  
print(species.livingC100)
```

	run	species	livingC
1	reference	casa	0.000000
2	reference	fasy	75.469157
3	reference	piab	274.331862
4	reference	pisy	12.701789
5	reference	qupe	67.044909
6	4deg_20percdrier	casa	4.546279
7	4deg_20percdrier	fasy	110.708431
8	4deg_20percdrier	piab	159.773037
9	4deg_20percdrier	pisy	55.439370
10	4deg_20percdrier	qupe	97.802295

2.2.9.4 Assess the species proportions

Calculate the species proportions based on the stored C amount in the initial year (year==0) and the last year (year==100) For this we need the total C amount and the species-specific C amount.

```
LC0 <- dplyr::left_join(species.livingC0, livingC0, by = "run")  
LC0 <-  
  data.frame(LC0 |> dplyr::mutate(spec.prop = livingC / sum.livingC, year =
```

2 Process based

```
print(LC0)
0))
```

	run	species	livingC	sum.livingC	spec.prop	year
1	reference	casa	73.246102	208.0977	0.351979370	0
2	reference	fasy	1.636821	208.0977	0.007865637	0
3	reference	piab	58.083681	208.0977	0.279117344	0
4	reference	pisy	18.250374	208.0977	0.087700985	0
5	reference	qupe	56.880735	208.0977	0.273336665	0
6	4deg_20percdrier	casa	73.246102	208.0977	0.351979370	0
7	4deg_20percdrier	fasy	1.636821	208.0977	0.007865637	0
8	4deg_20percdrier	piab	58.083681	208.0977	0.279117344	0
9	4deg_20percdrier	pisy	18.250374	208.0977	0.087700985	0
10	4deg_20percdrier	qupe	56.880735	208.0977	0.273336665	0

```
LC100 <- dplyr::left_join(species.livingC100, livingC100, by = "run")
LC100 <-
  data.frame(LC100 |> dplyr::mutate(spec.prop = livingC / sum.livingC, year =
    100))
print(LC100)
```

	run	species	livingC	sum.livingC	spec.prop	year
1	reference	casa	0.000000	429.5477	0.00000000	100
2	reference	fasy	75.469157	429.5477	0.17569447	100
3	reference	piab	274.331862	429.5477	0.63865282	100
4	reference	pisy	12.701789	429.5477	0.02957015	100
5	reference	qupe	67.044909	429.5477	0.15608256	100
6	4deg_20percdrier	casa	4.546279	428.2694	0.01061547	100
7	4deg_20percdrier	fasy	110.708431	428.2694	0.25850184	100
8	4deg_20percdrier	piab	159.773037	428.2694	0.37306666	100
9	4deg_20percdrier	pisy	55.439370	428.2694	0.12944975	100
10	4deg_20percdrier	qupe	97.802295	428.2694	0.22836629	100

Put the two tables together and visualize the results!

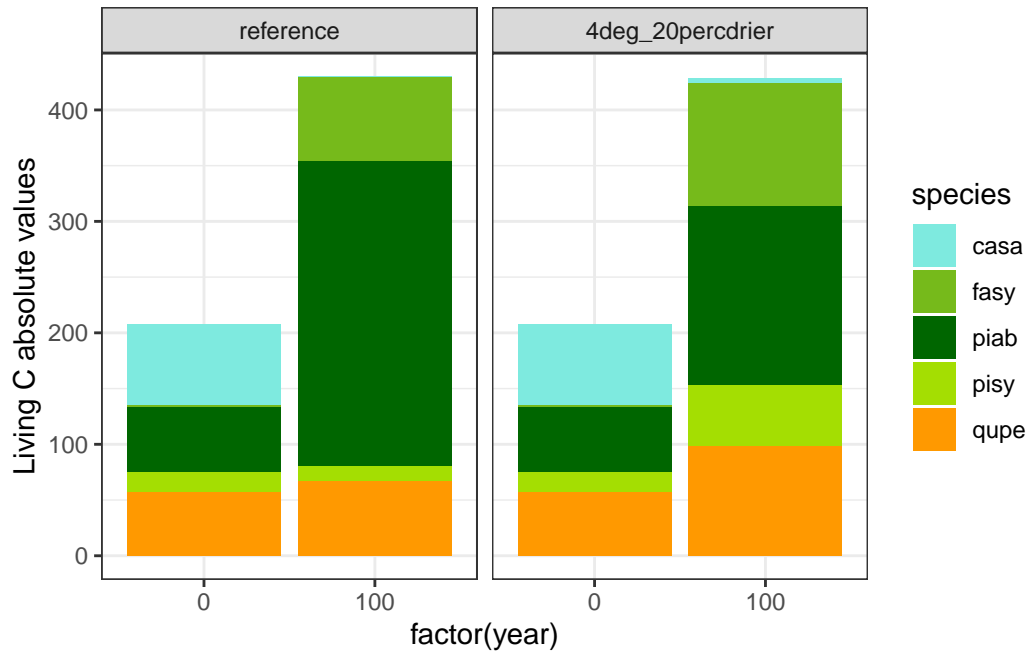
```
LC <- rbind(LC0, LC100)

LC$run <- factor(LC$run, levels = c(name1, name2))

ggplot2::ggplot(LC, ggplot2::aes(fill = species, y = livingC , x = factor(year))) +
  ggplot2::geom_bar(position = "stack", stat = "identity") +
```

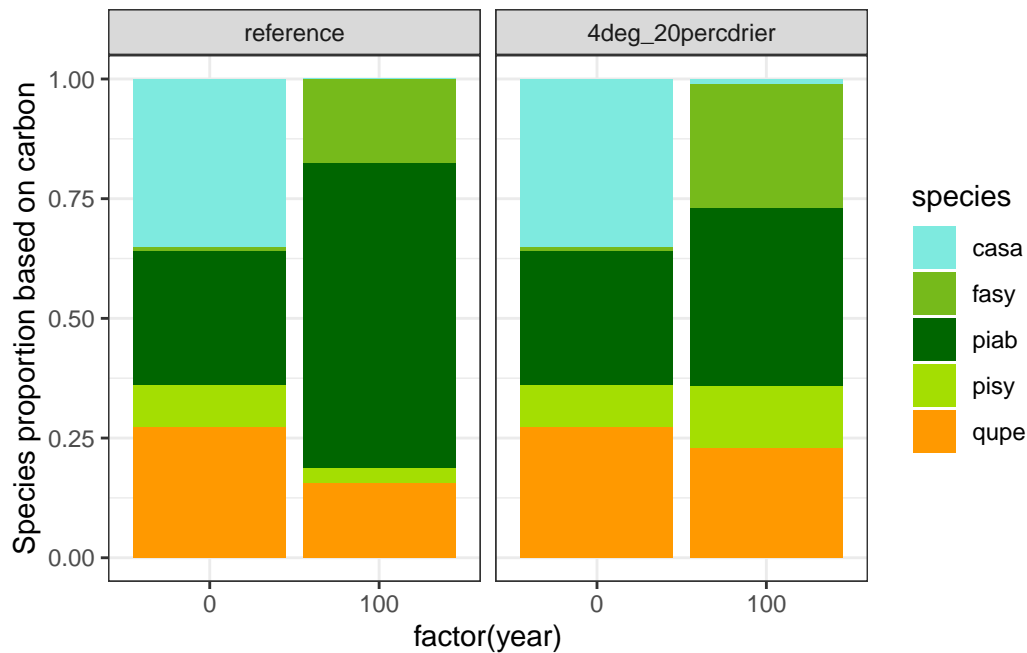
2 Process based

```
ggplot2::scale_fill_manual(values = cols.all) +
ggplot2::facet_wrap( ~ run) +
ggplot2::ylab("Living C absolute values") +
ggplot2::theme_bw()
```



```
ggplot2::ggplot(LC, ggplot2::aes(fill = species, y = spec.prop,
                                   x = factor(year))) +
ggplot2::geom_bar(position = "stack", stat = "identity") +
ggplot2::scale_fill_manual(values = cols.all) +
ggplot2::facet_wrap( ~ run) +
ggplot2::ylab("Species proportion based on carbon") +
ggplot2::theme_bw()
```

2 Process based



The living carbon stock doubled over 100 years due to the accumulation of biomass at the plot irrespective of the climate scenario. The comparison of the results between the two climate scenarios showed the difference in the tree species share. The living C stock of spruce was lower under the climate change reflecting its susceptibility to dry conditions, while the living C stock of *Quercus pubescens*, *Fagus sylvatica* and *Pinus sylvestris* were higher. Additionally, we see a significant increase in the share of the shade-tolerant beech at the expense of light-demanding chestnut.

2.2.10 Question A - Group 3

- How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 3?

Run the model as it is described in the previous chapters to have at least 2 simulations completed: one with reference conditions and one scenario with some changes in the environment. Check if you have the output files in the output folder: *iLand_simulations/output/*. If you manage to simulate 2 scenarios, you can expand the code to have file3, and 3rd from each variable, or make the comparison in 2 step always comparing one scenario to the reference conditions at the time.

Open the *summerSchoolExercise.Rproj* project in R studio and start working there.

2.2.10.1 Read the tree output table

Read in the *tree* output table from two outputs you have for your plot:

```
path1 <- "model/iLand_simulations/output/Output_plot3.sqlite"
path2 <-
  "model/iLand_simulations/output/Output_plot3_4deg_20percdrier.sqlite"

file1 <- here::here(path1)
file2 <- here::here(path2)

# Give some name for the three simulations.
name1 <- "reference"
name2 <- "4deg_20percdrier"

# Read in data using the RSQLite package
sqlite.driver <- RSQLite::dbDriver("SQLite")
db1 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file1) # connect to the file1

tables.in.the.file <-
  RSQLite::dbListTables(db1) # explore the tables in the file1
print(tables.in.the.file)

[1] "carbon"          "carbonflow"      "dynamicstand"
[4] "landscape"       "landscape_removed" "runinfo"
[7] "stand"           "tree"

# We will work with "tree" table and tree-scale data:
tree1 <- RSQLite::dbReadTable(db1, "tree")
RSQLite::dbDisconnect(db1) # disconnect to the file1

# READ IN DATA FROM THE SECOND FILE: -----
db2 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file2) # connect to the file2
tree2 <- RSQLite::dbReadTable(db2, "tree")
RSQLite::dbDisconnect(db2)
```

Merge the data from the two files together and make a column which tells which comes from which simulation, and study the table. The column “run” will support the plotting where we can use facets in ggplot.

2 Process based

```
tree <- rbind(tree1 |> dplyr::mutate(run = name1) ,
              tree2 |> dplyr::mutate(run = name2))

head(tree)
```

	year	ru	rid	species	id	x	y	dbh	height	basalArea	volume_m3	age
1	0	0	1	piab	1	7	41	80.47096	54.10505	0.5085905	11.639825	579
2	0	0	1	piab	2	57	63	77.85299	52.34484	0.4760367	10.540342	560
3	0	0	1	fasy	3	33	45	77.32411	43.94139	0.4695910	8.934730	549
4	0	0	1	fasy	4	79	97	76.44166	43.43991	0.4589338	8.632307	542
5	0	0	1	piab	5	99	35	74.65018	50.19142	0.4376748	9.292261	537
6	0	0	1	fasy	6	11	77	74.61653	42.40273	0.4372804	8.028635	530
	leafArea_m2	foliageMass	stemMass	branchMass	fineRootMass	coarseRootMass						
1	381.4915	89.76272	3273.843	531.3649	67.32204	829.4560						
2	362.3074	85.24881	3017.901	492.4429	63.93661	756.3417						
3	387.3896	35.21723	3946.903	484.7826	26.41292	432.6764						
4	379.9038	34.53671	3846.260	472.1521	25.90253	421.3067						
5	339.3250	79.84119	2721.455	447.0886	59.88089	672.6917						
6	364.6130	33.14663	3642.679	446.6253	24.85997	398.3362						
	lri	lightResponse	stressIndex	reserve_kg	treeFlags	run						
1	0.5557720		0	0	157.08476	0 reference						
2	0.3826686		0	0	149.18542	0 reference						
3	0.4321580		0	0	61.63016	0 reference						
4	0.7127665		0	0	60.43925	0 reference						
5	1.0000000		0	0	139.72208	0 reference						
6	0.3028383		0	0	58.00661	0 reference						

2.2.10.2 Visualize trees in year 0 and year 100!

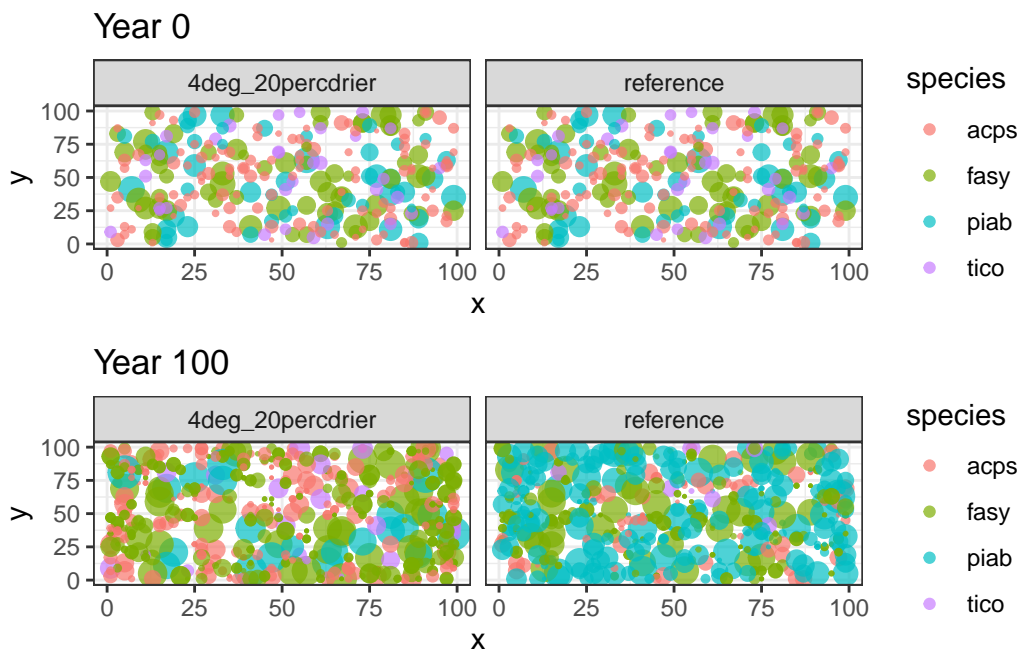
To explore a bit the simulation, plot the tree locations at the beginning and the end of the simulation! For plotting we use the colors for the species, and the size of a circles to show dbh differences. Here we divided dbh with the value 20 just for visualization, not to have too huge circles shading each others completely, but still see the trees.

```
tree0 <- tree |> dplyr::filter(year == 0)
g1 <-
  ggplot2::ggplot(tree0, ggplot2::aes(x = x, y = y, color = species)) +
  ggplot2::geom_point(size = tree0$dbh / 20, alpha = 0.7) +
  ggplot2::ggtitle("Year 0") +
  ggplot2::facet_wrap(~ run) + ggplot2::theme_bw()
```

2 Process based

```
tree100 <- tree |> dplyr::filter(year == 100)
g2 <-
  ggplot2::ggplot(tree100, ggplot2::aes(x = x, y = y, color = species)) +
  ggplot2::geom_point(size = tree100$dbh / 20, alpha = 0.7) +
  ggplot2::ggtitle("Year 100") +
  ggplot2::facet_wrap(~ run) + ggplot2::theme_bw()

gridExtra::grid.arrange(g1, g2, ncol = 1)
```



We can see that the species composition is somewhat changed and the forest become more dense. We can even identify the same trees at the same location.

2.2.10.3 Visualize some changes in time!

Visualize some changes in time, for example the mean diameter of the trees!

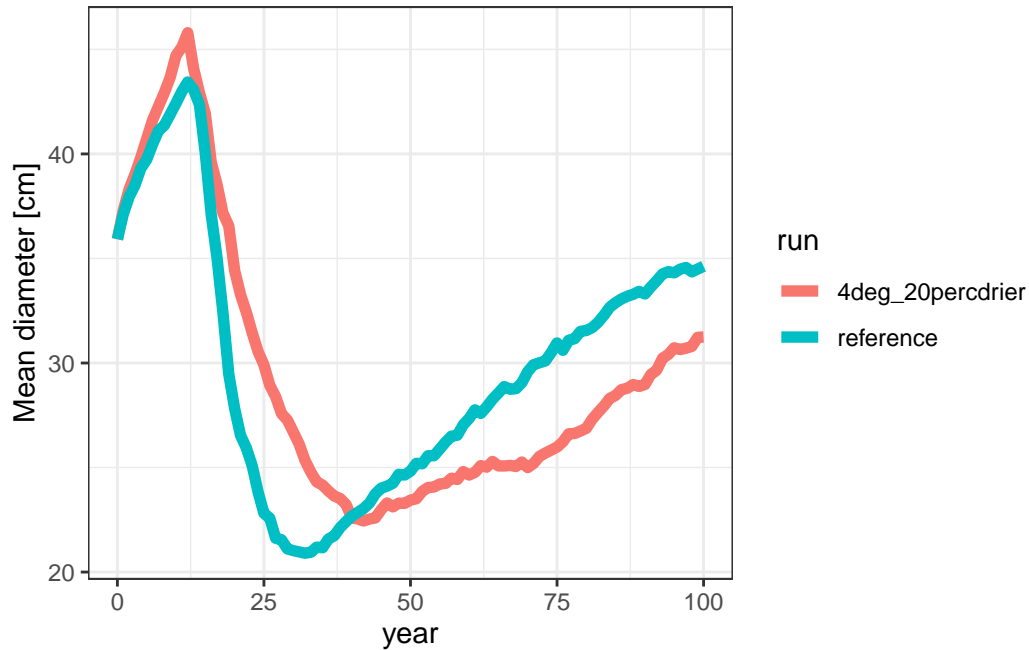
```
sum.table <- tree |> dplyr::group_by(year, run) |>
  dplyr::summarise(
    N = dplyr::n(),
    MD = mean(dbh, na.rm = TRUE),
    #mean diameter
```

```

    BA = sum(basalArea)
  ) #basal area

ggplot2::ggplot(sum.table, ggplot2::aes(x = year, y = MD, color = run)) +
  ggplot2::geom_line(lwd = 2) +
  ggplot2::ylab("Mean diameter [cm]") + ggplot2::theme_bw()

```



We can see that there is an initial increase in mean dbh, then a drop after 20/25 years. This is due to the growing regeneration layer that is produced by the seeds of the existing trees. In the initial year we do not have regeneration layer in the simulations (it is possible to put there, but we do not have it now). And until the small trees grow up to higher than 4 m they are not appearing in the individual *tree* output as they are handled in cohorts.

2.2.10.4 Species diversity

To study biodiversity aspects, we can calculate biodiversity indicators. Let's use the *adiv* package for species diversity. First we need to change the database structure to have the number of trees for each species in different columns (*pivot_wider*) without any additional columns for the *adiv* package *speciesdiv* function. We work first only with tree data from one simulation (*tree1*). You can find the documentation of the *adiv* package here: "[documents/adiv.pdf](#)"

2 Process based

```
# First we need to change the
tlong1 <- tree1 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

head(tlong1)
```

```
# A tibble: 6 x 4
  acps fasy piab tico
<int> <int> <int> <int>
1   113    67   49   32
2   110    67   49   28
3   108    67   49   27
4   106    66   47   27
5   104    65   47   27
6   103    64   46   26
```

Then we apply the “speciesdiv” function and we add back the year and run columns to have the information.

```
div1 <- data.frame(activ::speciesdiv(tlong1)) |>
  dplyr::mutate(year = unique(tree1$year), run = name1)
head(div1)
```

```
  richness GiniSimpson Simpson Shannon Margalef Menhinick McIntosh year
1         4    0.6963785 3.293574 1.282865 0.5391300 0.2475938 0.4786064    0
2         4    0.6935024 3.262668 1.274467 0.5417769 0.2509823 0.4762610    1
3         4    0.6939255 3.267178 1.274173 0.5429419 0.2524778 0.4768591    2
4         4    0.6938000 3.265839 1.274493 0.5449263 0.2550307 0.4770630    3
5         4    0.6955241 3.284332 1.277850 0.5461435 0.2566001 0.4789300    4
6         4    0.6936853 3.264617 1.274063 0.5477988 0.2587385 0.4774250    5

  run
1 reference
2 reference
```

2 Process based

3 reference
4 reference
5 reference
6 reference

We make the same steps for the other simulation results (tree2).

```
tlong2 <- tree2 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

div2 <- data.frame(adiv::speciesdiv(tlong2)) |>
  dplyr::mutate(year = unique(tree2$year), run = name2)
```

We merge them all together and make a plot based on Shannon index.

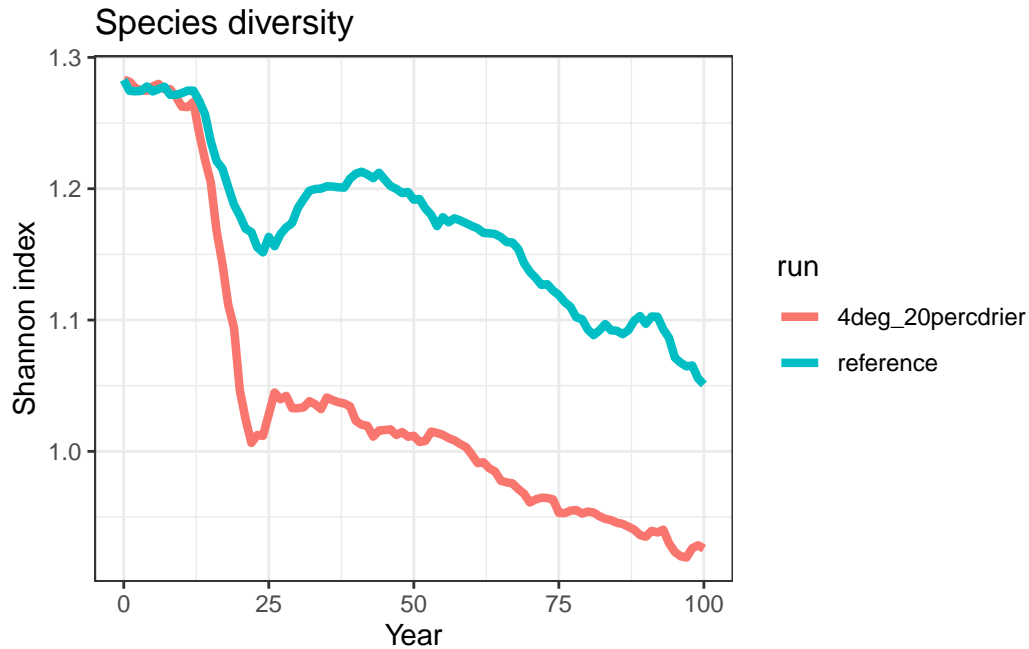
```
tlong2 <- tree2 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

div2 <- data.frame(adiv::speciesdiv(tlong2)) |>
  dplyr::mutate(year = unique(tree2$year), run = name2)

div <- rbind(div1, div2)

ggplot2::ggplot(div, ggplot2::aes(year, Shannon , color = run)) +
  ggplot2::geom_line(lwd = 1.5) +
  ggplot2::ggtitle("Species diversity") +
```

```
ggplot2::labs(x = "Year", y = "Shannon index") +
ggplot2::theme_bw()
```



We see that the values of Shannon index are first stable and then tend to decrease, while the reduction under the 4deg_20percdrier is more substantial than under the reference scenario. Shannon index is calculated as follows:

$$S = - \sum (p_i * \log(p_i))$$

Where p_i is the relative abundance of the species i , calculated as the ratio of the abundance of the species i and the abundance of all species. Shannon-index is 0 when we have only one species at the stand.

2.2.10.5 Spatial diversity

We can calculate some additional biodiversity indices targeting spatial diversity using the *treespac* package developed by Francesco Chianucci (fchianucci@gmail.com). You can find a short description of the package here: “[documents/treespac_package.pdf](#)”

You can install the package using devtools:

```
devtools::install_gitlab('fchianucci/treespat')
```

Then, we are ready to use it! Let's calculate these two indices:

- Diameter differentiation (Gadow, 1993): Spatial size inequality defined as the mean of the ratio of smaller and larger plant sizes in the nearest neighbors of a tree. The value of the index increases with increasing average size difference between neighboring trees. 0 is implying that neighboring trees have equal size.
- Mingling (Aguirre et al., 2003): One very intuitive extension of taxonomic species diversity (either richness or abundance) is considering spatial mingling, namely how plants of the same (con-specific neighbors) or different (hetero-specific neighbors) species are arranged in space. The mingling index calculates the proportion of the k nearest neighbors that do not belong to the same species as the reference tree. For example, with four neighbors, the mingling attribute can assume five values, ranging from 0 (all trees are of the same species) to 1 (all trees belong to different species).

Calculate the indices based on the reference run, then based on the scenario. For each index we add extra columns with the index name, and run name for further analyses. The `max.k` parameter is telling how many neighboring trees we want to account for.

```
# diameter differentiation:
# mingling

diff <- data.frame(treespat::DIFF(tree1, .x = x, .y = y, .mark = dbh, xmax = 100,
                                ymax = 100, max.k = 4, shape='square', .groups=c('year'))) |>
  dplyr::mutate(index="DIFF", name="Diameter differentiation", run=name1))
head(diff)
```

	year	DIFF index	name	run
1	0	0.3938630	DIFF Diameter differentiation	reference
2	1	0.3825122	DIFF Diameter differentiation	reference
3	2	0.3765717	DIFF Diameter differentiation	reference
4	3	0.3749870	DIFF Diameter differentiation	reference
5	4	0.3681887	DIFF Diameter differentiation	reference
6	5	0.3639637	DIFF Diameter differentiation	reference

```
ming <- data.frame(treespat::MING(tree1, .x = x, .y = y, .species = species,
                                xmax = 100, ymax = 100, max.k = 4, shape='square', .groups=c('year'))) |>
  dplyr::mutate(index="MING", name="Mingling", run=name1))

indices1<-rbind(diff |> dplyr::rename(value=DIFF),
  ming |> dplyr::rename(value=MINGLING ) )
```

2 Process based

We calculated each index as separate variable and then merged them into indices1. Here I renamed the column names to have it all “value”, for easier plotting later on. (each line has the information on which run and which index is the value) We do the same for the scenario simulation, we merge them together and make the plotting.

```
diff <- data.frame(treespat::DIFF(tree2, .x = x, .y = y, .mark = dbh,
                                xmax = 100, ymax = 100, max.k = 4,
                                shape = 'square', .groups = c('year')) |>
  dplyr::mutate(index = "DIFF", name = "Diameter differentiation",
               run = name2))

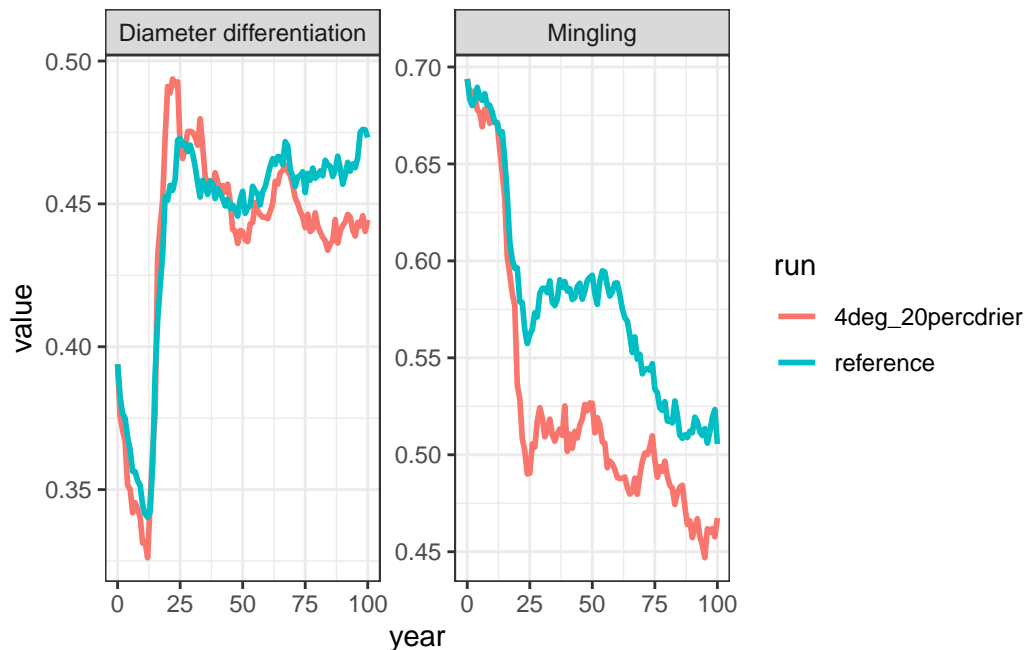
ming <- data.frame(treespat::MING(tree2, .x = x, .y = y,
                                .species = species,
                                xmax = 100, ymax = 100, max.k = 4,
                                shape = 'square', .groups=c('year')) |>
  dplyr::mutate(index = "MING", name = "Mingling",
               run = name2))

indices2 <- rbind(diff |> dplyr::rename(value = DIFF),
                 ming |> dplyr::rename(value = MINGLING))

indices <- rbind(indices1, indices2)

ggplot2::ggplot(indices, ggplot2::aes(x = year, y = value, color = run)) +
  ggplot2::geom_line(lwd = 1) +
  ggplot2::facet_wrap( ~ name, ncol = 2, scales = "free") +
  ggplot2::theme_bw()
```


2 Process based



The diameter differentiation increases with the increasing average difference in diameter between neighboring trees. The value 0 indicates that neighboring trees have equal diameters. The development shows that the index values decrease during the first 20/25 years, after which it steeply increases due to the occurrence of ingrowth and then the values are stabilized. The values for the reference climate scenario are slightly higher than for the climate change scenario indicating slower growth under drier and warmer conditions and/or the differences in tree species composition (proportion of tree species). The mingling index is studying the neighboring trees regarding their species. In both development there is a decrease in the index after the occurrence of regeneration. Using the climate change scenario the index shows steeper decrease.

2.2.11 Question B - Group 3

- How are the species distribution and total living biomass C content changing in time on Plot 3? Compare 0 year and 100 year status in the reference case and in the case of your scenario(s)!

Run the model as it is described in the previous chapters to have at least 2 simulations completed: one with reference conditions and one scenario with some changes in the environment. Check if you have the output files in the output folder: *iLand_simulations/output/*. If you manage to simulate 2 scenarios, you can expand the code to have file3, and 3rd from each variable, or make the comparison in 2 step always comparing one

scenario to the reference conditions at the time. Open the *summerSchoolExercise.Rproj* project in R studio and start working there.

2.2.11.1 Read the landscape output table

Read in the *landscape* output table from two outputs you have for your plot:

```
path1 <- "model/iLand_simulations/output/Output_plot3.sqlite"
path2 <-
  "model/iLand_simulations/output/Output_plot3_4deg_20percdrier.sqlite"

file1 <- here::here(path1)
file2 <- here::here(path2)

# Give some name for the three simulations.
name1 <- "reference"
name2 <- "4deg_20percdrier"

# Read in data using the RSQLite package
sqlite.driver <- RSQLite::dbDriver("SQLite")

# We will work with "landscape" table and tree-scale data:
db1 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file1) # connect to the file1
landscape1 <- RSQLite::dbReadTable(db1, "landscape")
RSQLite::dbDisconnect(db1) # disconnect to the file1

db2 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file2) # connect to the file2
landscape2 <- RSQLite::dbReadTable(db2, "landscape")
RSQLite::dbDisconnect(db2)
```

Merge the data from the three files together and make a column which tells which comes from which simulation, and study the table. The column “run” will support the plotting where we can use facets in ggplot.

```
landscape <- rbind(landscape1 |> dplyr::mutate(run = name1) ,
  landscape2 |> dplyr::mutate(run = name2))

head(landscape)
```

```
year area area_100m species count_ha dbh_avg_cm height_avg_m volume_m3
```

2 Process based

1	0	1	1	acps	113	21.62995	18.52183	41.57766	
2	0	1	1	fasy	67	52.05965	29.58422	218.19995	
3	0	1	1	piab	49	50.26190	33.79384	182.14630	
4	0	1	1	tico	32	30.63841	23.38127	21.38005	
5	1	1	1	acps	110	22.55262	19.19236	43.91456	
6	1	1	1	fasy	67	53.02260	30.03274	226.37490	
total_carbon_kg		gwl_m3		basal_area_m2		NPP_kg	NPPabove_kg	LAI	
1	19316.78		41.57766		4.693063		0.000	0.000	0.4800021
2	74974.41		218.19995		15.193077		0.000	0.000	1.3766679
3	45666.89		182.14630		10.724078		0.000	0.000	0.9377907
4	11816.98		21.38005		2.379258		0.000	0.000	0.2673509
5	20284.37		44.17968		4.895663		3936.955	2581.190	0.4935337
6	79317.75		226.37490		15.670265		12642.176	8302.325	1.4294626
cohort_count_ha		run							
1	0		reference						
2	0		reference						
3	0		reference						
4	0		reference						
5	27		reference						
6	49		reference						

2.2.11.2 Visualize changes in time

We can see that the output is given for each year and each species in our 1ha area. Plot the living carbon (total_carbon_kg) in time, coloring by species. We give a unified species coloring in the beginning. Plot number of trees and mean diameter. Carbon and number of trees can be plotted in an additive way, but for mean dbh we do line plot per species.

```
cols.all=c( "rops"="#e0e0e0","acpl"="#A9A9A9","alin"="#696969","alvi"="#2e2e2e",
  "bepe"="#fADFAD","casa"="#7eeadf","coav"="#20c6b6","tipl"="#645394",
  "ulgl"="#311432","saca"="#D8BFD8","soar"="#DDA0DD","soau"="#BA55D3",
  "pice"="#D27D2D","pini"="#a81c07","algl"="#2ECBE9","tico"="#128FC8",
  "potr"="#00468B","poni"="#5BAEB7","frex"="#fe9cb5","cabe"="#fe6181",
  "acps"="#fe223e","lade"="#FFFE71","abal"="#FFD800","pisy"="#A4DE02",
  "fasy"="#76BA1B","piab"="#006600","quro"="#FF7F00","qupe"="#FF9900",
  "qupu"="#CC9900")
```

```
# We set here the order of the run categories for the table, to have the first
#run first and then the second. (left-right of the plots)
landscape$run <- factor(landscape$run, levels = c(name1, name2))
```

2 Process based

```
# Plot the living carbon content, to have tonnes/ha, we divide total_carbon_kg by 1000.
g1 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, total_carbon_kg / 1000 ,
                                           fill = factor(species))) +

  ggplot2::geom_area() +
  ggplot2::scale_fill_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +

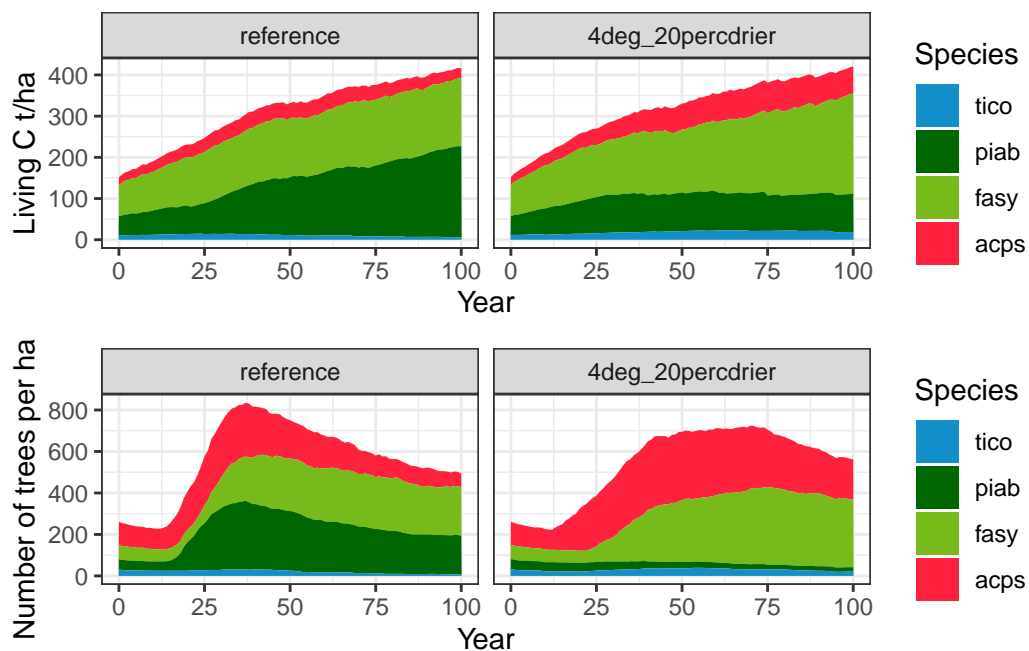
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Living C t/ha", fill = "Species") +
  ggplot2::theme_bw()

g2 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, count_ha,
                                           fill = factor(species))) +

  ggplot2::geom_area() +
  ggplot2::scale_fill_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +

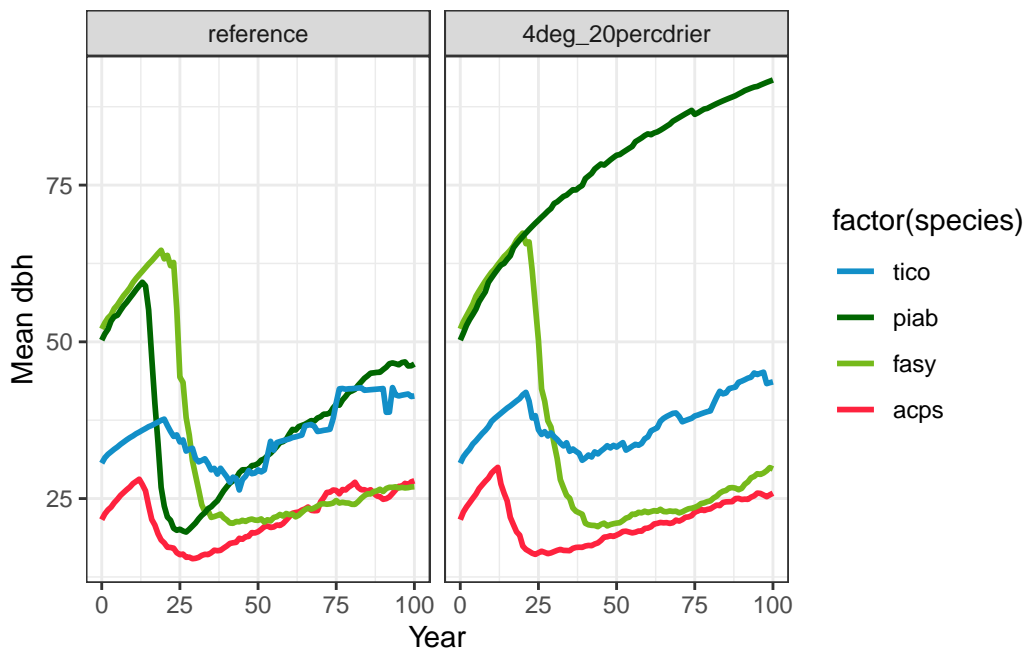
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Number of trees per ha",
               fill = "Species") +
  ggplot2::theme_bw()

gridExtra::grid.arrange(g1, g2, ncol = 1)
```



2 Process based

```
g3 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, dbh_avg_cm ,
                                           color = factor(species))) +
  ggplot2::geom_line(lwd = 1) +
  ggplot2::scale_color_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Mean dbh", fill = "Species") +
  ggplot2::theme_bw()
print(g3)
```



Note that we do not have management intervention, and in the model trees shorter than 4m are not included in these outputs. However as they grow taller than 4m, model start to handle them as individual trees and including in these outputs which we are looking now.

Beech profited from drier and warmer conditions at the expense of spruce. *Acer pseudo-platanus* and *Tilia cordata* also slightly increased their proportion in the living C stock and number of trees. The increase of the number of trees indicates the success of the regeneration under reference conditions, however this increase is much slower under the climate change scenario.

Mean dbh of individual species reflected the development of the number of trees. The increase in the number of trees caused the reduction of mean dbh, e.g. of beech under

both scenarios between 20 and 35 simulation years. The species-specific graphs for mean dbh indicates that some species do not have successful regeneration under the climate change scenario. Spruce dbh decreased only under reference scenario, when we observed the increase of the number of trees, while under climate change scenario the mean dbh of spruce was growing indicating the growth of remaining trees of the species at the plot.

2.2.11.3 Assess the stored carbon amount

Calculate the stored C amount in the initial year (year==0) and the last year (year==100)!

```
livingC0 <- data.frame(
  landscape |> dplyr::filter(year == 0) |>
    dplyr::group_by(run) |>
    dplyr::summarize(sum.livingC = sum(total_carbon_kg / 1000))
)
print(livingC0)
```

	run	sum.livingC
1	reference	151.7751
2	4deg_20percdrier	151.7751

```
livingC100 <- data.frame(
  landscape |> dplyr::filter(year == 100) |>
    dplyr::group_by(run) |>
    dplyr::summarize(sum.livingC = sum(total_carbon_kg / 1000))
)
print(livingC100)
```

	run	sum.livingC
1	reference	416.3909
2	4deg_20percdrier	420.1472

The initial conditions are same for the runs, but they are ending up at different C levels.

Calculate the stored C amount PER SPECIES in the initial year (year==0) and the last year (year==100)

```
species.livingC0 <-
  data.frame(
    landscape |> dplyr::filter(year == 0) |>
      dplyr::group_by(run, species) |>
      dplyr::summarize(livingC = sum(total_carbon_kg / 1000))
  )
print(species.livingC0)
```

	run	species	livingC
1	reference	acps	19.31678
2	reference	fasy	74.97441
3	reference	piab	45.66689
4	reference	tico	11.81698
5	4deg_20percdrier	acps	19.31678
6	4deg_20percdrier	fasy	74.97441
7	4deg_20percdrier	piab	45.66689
8	4deg_20percdrier	tico	11.81698

```
species.livingC100 <-
  data.frame(
    landscape |> dplyr::filter(year == 100) |>
      dplyr::group_by(run, species) |>
      dplyr::summarize(livingC = sum(total_carbon_kg / 1000))
  )
print(species.livingC100)
```

	run	species	livingC
1	reference	acps	25.042637
2	reference	fasy	164.555979
3	reference	piab	221.267535
4	reference	tico	5.524713
5	4deg_20percdrier	acps	65.167214
6	4deg_20percdrier	fasy	243.374438
7	4deg_20percdrier	piab	92.649456
8	4deg_20percdrier	tico	18.956110

2.2.11.4 Assess the species proportions

Calculate the species proportions based on the stored C amount in the initial year (year==0) and the last year (year==100) For this we need the total C amount and the species-specific C amount.

2 Process based

```
LC0 <- dplyr::left_join(species.livingC0, livingC0, by = "run")
LC0 <-
  data.frame(LC0 |> dplyr::mutate(spec.prop = livingC / sum.livingC,
                                year = 0))
print(LC0)
```

	run	species	livingC	sum.livingC	spec.prop	year
1	reference	acps	19.31678	151.7751	0.1272724	0
2	reference	fasy	74.97441	151.7751	0.4939837	0
3	reference	piab	45.66689	151.7751	0.3008854	0
4	reference	tico	11.81698	151.7751	0.0778585	0
5	4deg_20percdrier	acps	19.31678	151.7751	0.1272724	0
6	4deg_20percdrier	fasy	74.97441	151.7751	0.4939837	0
7	4deg_20percdrier	piab	45.66689	151.7751	0.3008854	0
8	4deg_20percdrier	tico	11.81698	151.7751	0.0778585	0

```
LC100 <-
  dplyr::left_join(species.livingC100, livingC100, by = "run")
LC100 <-
  data.frame(LC100 |> dplyr::mutate(spec.prop = livingC / sum.livingC,
                                   year = 100))
print(LC100)
```

	run	species	livingC	sum.livingC	spec.prop	year
1	reference	acps	25.042637	416.3909	0.06014214	100
2	reference	fasy	164.555979	416.3909	0.39519594	100
3	reference	piab	221.267535	416.3909	0.53139383	100
4	reference	tico	5.524713	416.3909	0.01326809	100
5	4deg_20percdrier	acps	65.167214	420.1472	0.15510567	100
6	4deg_20percdrier	fasy	243.374438	420.1472	0.57925991	100
7	4deg_20percdrier	piab	92.649456	420.1472	0.22051665	100
8	4deg_20percdrier	tico	18.956110	420.1472	0.04511778	100

Put the two tables together and visualize the results!

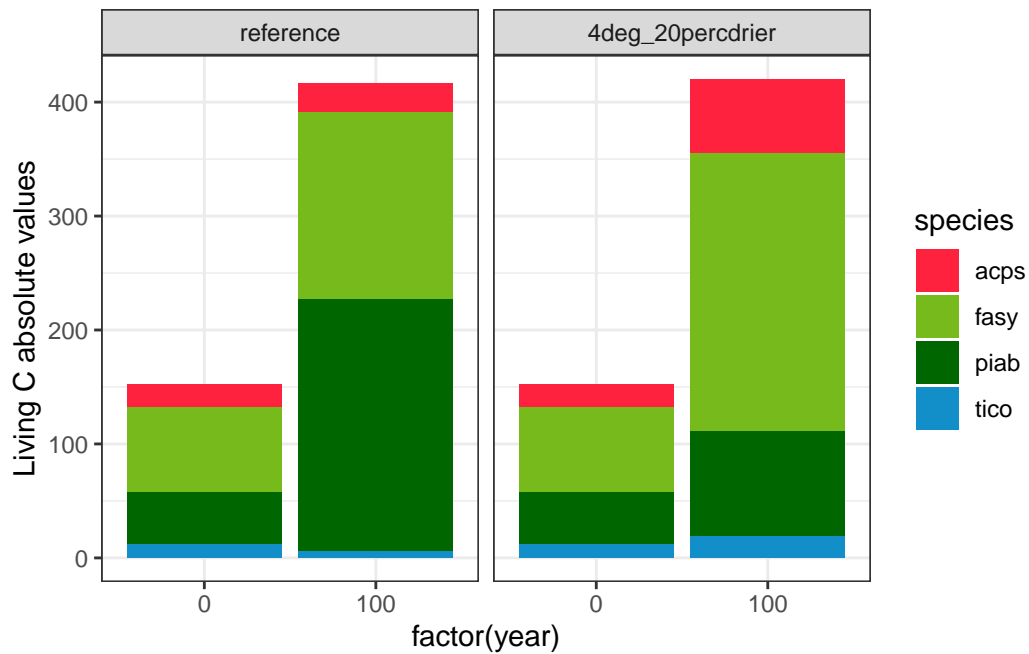
```
LC <- rbind(LC0, LC100)

LC$run <- factor(LC$run, levels = c(name1, name2))

ggplot2::ggplot(LC, ggplot2::aes(fill = species, y = livingC,
                                  x = factor(year))) +
```

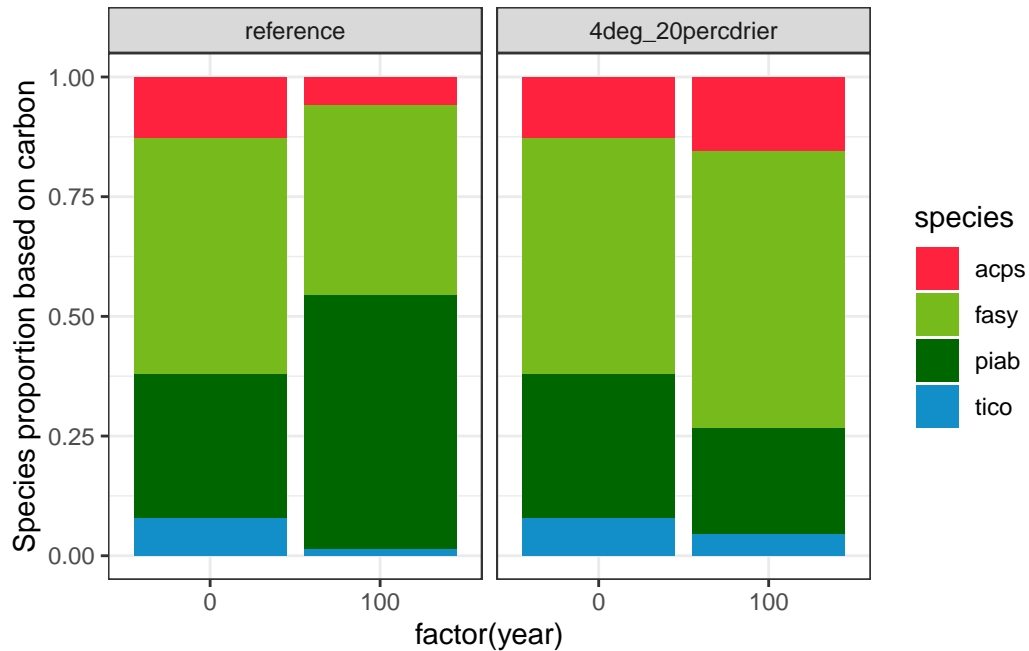

2 Process based

```
ggplot2::geom_bar(position = "stack", stat = "identity") +  
ggplot2::scale_fill_manual(values = cols.all) +  
ggplot2::facet_wrap( ~ run) +  
ggplot2::ylab("Living C absolute values") +  
ggplot2::theme_bw()
```



```
ggplot2::ggplot(LC, ggplot2::aes(fill = species, y = spec.prop,  
                                x = factor(year))) +  
ggplot2::geom_bar(position = "stack", stat = "identity") +  
ggplot2::scale_fill_manual(values = cols.all) +  
ggplot2::facet_wrap( ~ run) +  
ggplot2::ylab("Species proportion based on carbon") +  
ggplot2::theme_bw()
```

2 Process based



The living C stock more than doubled over 100 years due to the accumulation of biomass in trees and no management interventions. Moreover, the proportion of individual tree species changed depending on the driving climatic conditions. Under reference conditions the proportion of spruce has grown, under climate change the proportion of beech.

2.2.12 Question A - Group 4

- How are biodiversity indices changing in time and across the simulated scenario(s) on Plot 4?

Run the model as it is described in the previous chapters to have at least 2 simulations completed: one with reference conditions and one scenario with some changes in the environment. Check if you have the output files in the output folder: *iLand_simulations/output/*. If you manage to simulate 2 scenarios, you can expand the code to have file3, and 3rd from each variable, or make the comparison in 2 step always comparing one scenario to the reference conditions at the time.

Open the *summerSchoolExercise.Rproj* project in R studio and start working there.

Read in the *tree* output table from two outputs you have for your plot:

2 Process based

```
path1 <- "model/iLand_simulations/output/Output_plot4.sqlite"
path2 <-
  "model/iLand_simulations/output/Output_plot4_4deg_20percdrier.sqlite"

file1 <- here::here(path1)
file2 <- here::here(path2)

# Give some name for the three simulations.
name1 <- "reference"
name2 <- "4deg_20percdrier"

# Read in data using the RSQLite package
sqlite.driver <- RSQLite::dbDriver("SQLite")
db1 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file1) # connect to the file1

tables.in.the.file <-
  RSQLite::dbListTables(db1) # explore the tables in the file1
print(tables.in.the.file)
```

```
[1] "carbon"           "carbonflow"       "dynamicstand"
[4] "landscape"        "landscape_removed" "runinfo"
[7] "stand"            "tree"
```

```
# We will work with "tree" table and tree-scale data:
tree1 <- RSQLite::dbReadTable(db1, "tree")
RSQLite::dbDisconnect(db1) # disconnect to the file1

# READ IN DATA FROM THE SECOND FILE: -----
db2 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file2) # connect to the file2
tree2 <- RSQLite::dbReadTable(db2, "tree")
RSQLite::dbDisconnect(db2)
```

Merge the data from the two files together and make a column which tells which comes from which simulation, and study the table. The column “run” will support the plotting where we can use facets in ggplot.

```
tree <- rbind(tree1 |> dplyr::mutate(run = name1) ,
              tree2 |> dplyr::mutate(run = name2))
```

```
head(tree)
```

	year	ru	rid	species	id	x	y	dbh	height	basalArea	volume_m3	age
1	0	0	1	fasy	1	49	65	98.48459	55.96636	0.7617745	18.46041	699
2	0	0	1	fasy	2	51	15	94.27415	53.57367	0.6980316	16.19252	669
3	0	0	1	fasy	3	97	47	94.07941	53.46301	0.6951508	16.09238	668
4	0	0	1	fasy	4	63	7	91.67390	52.09602	0.6600568	14.88928	651
5	0	0	1	fasy	5	53	57	90.70000	51.54258	0.6461070	14.41977	644
6	0	0	1	fasy	6	13	95	89.37351	50.78876	0.6273466	13.79631	634
	leafArea_m2	foliageMass	stemMass	branchMass	fineRootMass	coarseRootMass						
1	584.4382	53.13074	6802.456	845.6093	39.84806	758.3801						
2	542.6002	49.32729	6165.430	764.7609	36.99547	685.2726						
3	540.6961	49.15419	6136.807	761.1324	36.86565	681.9930						
4	517.4045	47.03678	5789.331	717.1134	35.27758	642.2182						
5	508.0949	46.19045	5651.843	699.7122	34.64284	626.5004						
6	495.5272	45.04793	5467.530	676.3990	33.78595	605.4482						
	lri	lightResponse	stressIndex	reserve_kg	treeFlags	run						
1	0.6396535		0	0	92.97880	0 reference						
2	0.4253728		0	0	86.32275	0 reference						
3	1.0000000		0	0	86.01984	0 reference						
4	0.4545080		0	0	82.31436	0 reference						
5	0.6055760		0	0	80.83328	0 reference						
6	1.0000000		0	0	78.83388	0 reference						

2.2.12.1 Visualize trees in year 0 and year 100!

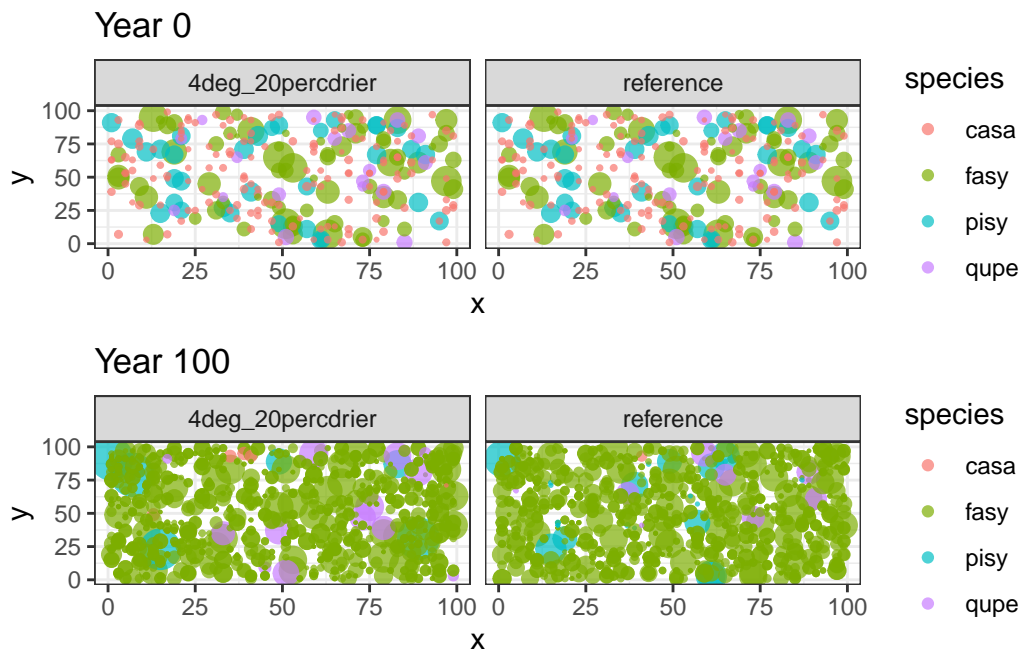
To explore a bit the simulation, plot the tree locations at the beginning and the end of the simulation! For plotting we use the colors for the species, and the size of a circles to show dbh differences. Here we divided dbh with the value 20 just for visualization, not to have too huge circles shading each others completely, but still see the trees.

```
tree0 <- tree |> dplyr::filter(year == 0)
g1 <-
  ggplot2::ggplot(tree0, ggplot2::aes(x = x, y = y, color = species)) +
  ggplot2::geom_point(size = tree0$dbh / 20, alpha = 0.7) +
  ggplot2::ggtitle("Year 0") +
  ggplot2::facet_wrap(~ run) + ggplot2::theme_bw()

tree100 <- tree |> dplyr::filter(year == 100)
g2 <-
```

2 Process based

```
ggplot2::ggplot(tree100, ggplot2::aes(x = x, y = y, color = species)) +  
  ggplot2::geom_point(size = tree100$dbh / 20, alpha = 0.7) +  
  ggplot2::ggtitle("Year 100") +  
  ggplot2::facet_wrap(~ run) + ggplot2::theme_bw()  
  
gridExtra::grid.arrange(g1, g2, ncol = 1)
```



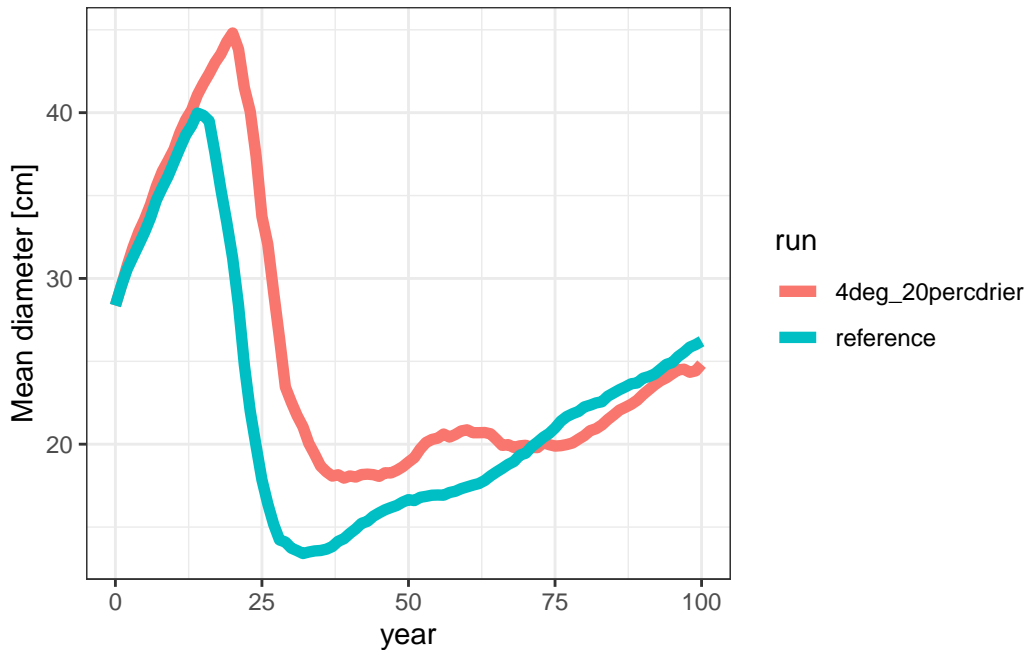
We can see that the species composition is somewhat changed and the forest become more dense. We can even identify the same trees at the same location.

2.2.12.2 Visualize some changes in time!

Visualize some changes in time, for example the mean diameter of the trees!

```
sum.table <- tree |> dplyr::group_by(year, run) |>  
  dplyr::summarise(  
    N = dplyr::n(),  
    MD = mean(dbh, na.rm = TRUE), #mean diameter  
    BA = sum(basalArea)#basal area  
  )
```

```
ggplot2::ggplot(sum.table, ggplot2::aes(x = year, y = MD, color = run)) +
  ggplot2::geom_line(lwd = 2) +
  ggplot2::ylab("Mean diameter [cm]") + ggplot2::theme_bw()
```



We can see that there is an initial increase in mean dbh, then a drop after 20/25 years. This is due to the growing regeneration layer that is produced by the seeds of the existing trees. In the initial year we do not have regeneration layer in the simulations (it is possible to put there, but we do not have it now). And until the small trees grow up to higher than 4 m they are not appearing in the individual *tree* output as they are handled in cohorts.

2.2.12.3 Species diversity

To study biodiversity aspects, we can calculate biodiversity indicators. Let's use the *adiv* package for species diversity. First we need to change the database structure to have the number of trees for each species in different columns (*pivot_wider*) without any additional columns for the *adiv* package *speciesdiv* function. We work first only with tree data from one simulation (*tree1*). You can find the documentation of the *adiv* package here: "[documents/adiv.pdf](#)"

2 Process based

```
# First we need to change the
tlong1 <- tree1 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

head(tlong1)
```

```
# A tibble: 6 x 4
  casa fasy pisy qupe
<int> <int> <int> <int>
1   152    70   33   18
2   147    70   32   18
3   143    69   32   18
4   139    68   30   18
5   139    68   30   18
6   137    68   30   18
```

Then we apply the “speciesdiv” function and we add back the year and run columns to have the information.

```
div1 <- data.frame(activ::speciesdiv(tlong1)) |>
  dplyr::mutate(year = unique(tree1$year), run = name1)
head(div1)
```

```
  richness GiniSimpson Simpson Shannon Margalef Menhinick McIntosh year
1         4    0.6052946 2.533535 1.109706 0.5348097 0.2420910 0.3956925    0
2         4    0.6092385 2.559105 1.115644 0.5369369 0.2447960 0.3993293    1
3         4    0.6131053 2.584683 1.122656 0.5387598 0.2471208 0.4028815    2
4         4    0.6129335 2.583535 1.122122 0.5413928 0.2504897 0.4030962    3
5         4    0.6129335 2.583535 1.122122 0.5413928 0.2504897 0.4030962    4
6         4    0.6154135 2.600195 1.126170 0.5421632 0.2514778 0.4053326    5

  run
1 reference
2 reference
```

2 Process based

3 reference
4 reference
5 reference
6 reference

We make the same steps for the other simulation results (tree2).

```
tlong2 <- tree2 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

div2 <- data.frame(adiv::speciesdiv(tlong2)) |>
  dplyr::mutate(year = unique(tree2$year), run = name2)
```

We merge them all together and make a plot based on Shannon index.

```
tlong2 <- tree2 |>
  dplyr::group_by(year, species) |> dplyr::summarise(N = dplyr::n()) |>
  tidyr::pivot_wider(
    id_cols = 'year',
    names_from = 'species',
    values_from = 'N',
    values_fill = 0
  ) |>
  dplyr::ungroup() |>
  dplyr::select(-year)

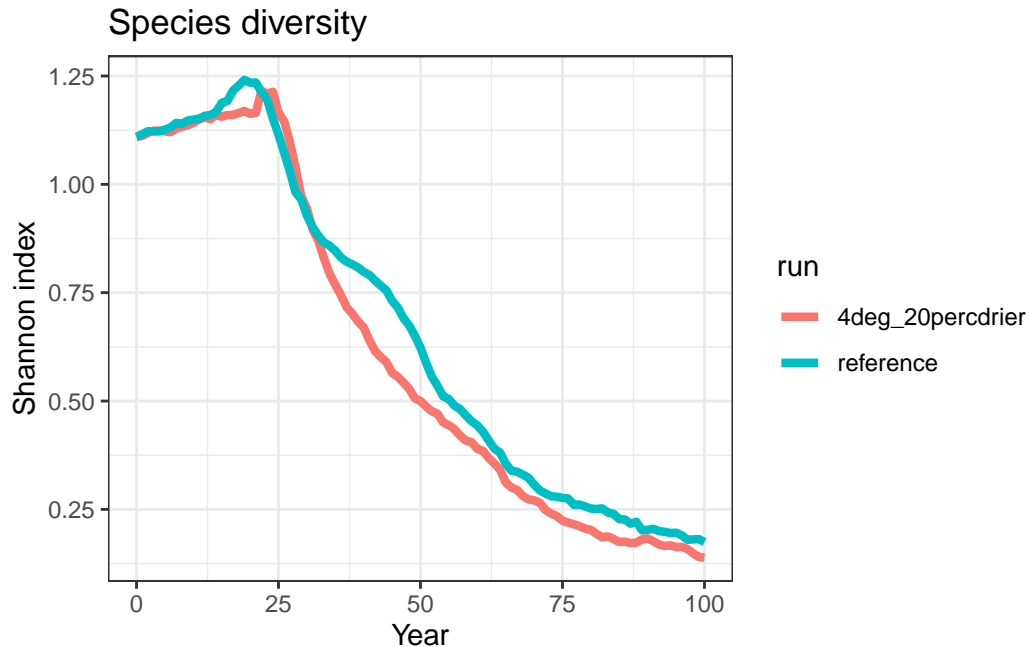
div2 <- data.frame(adiv::speciesdiv(tlong2)) |>
  dplyr::mutate(year = unique(tree2$year), run = name2)

div <- rbind(div1, div2)

ggplot2::ggplot(div, ggplot2::aes(year, Shannon , color = run)) +
  ggplot2::geom_line(lwd = 1.5) +
  ggplot2::ggtitle("Species diversity") +
```



```
ggplot2::labs(x = "Year", y = "Shannon index") +
ggplot2::theme_bw()
```



We see that the values of Shannon index are first stable and then tend to decrease, while the reduction under the 4deg_20percdrier is more substantial than under the reference scenario. Shannon index is calculated as follows:

$$S = - \sum (p_i * \log(p_i))$$

Where p_i is the relative abundance of the species i , calculated as the ratio of the abundance of the species i and the abundance of all species. Shannon-index is 0 when we have only one species at the stand.

2.2.12.4 Spatial diversity

We can calculate some additional biodiversity indices targeting spatial diversity using the *treespac* package developed by Francesco Chianucci (fchianucci@gmail.com). You can find a short description of the package here: “[documents/treespac_package.pdf](#)”

You can install the package using devtools:

```
devtools::install_gitlab('fchianucci/treespat')
```

Then, we are ready to use it! Let's calculate these two indices:

- Diameter differentiation (Gadow, 1993): Spatial size inequality defined as the mean of the ratio of smaller and larger plant sizes in the nearest neighbors of a tree. The value of the index increases with increasing average size difference between neighboring trees. 0 is implying that neighboring trees have equal size.
- Mingling (Aguirre et al., 2003): One very intuitive extension of taxonomic species diversity (either richness or abundance) is considering spatial mingling, namely how plants of the same (con-specific neighbors) or different (hetero-specific neighbors) species are arranged in space. The mingling index calculates the proportion of the k nearest neighbors that do not belong to the same species as the reference tree. For example, with four neighbors, the mingling attribute can assume five values, ranging from 0 (all trees are of the same species) to 1 (all trees belong to different species).

Calculate the indices based on the reference run, then based on the scenario. For each index we add extra columns with the index name, and run name for further analyses. The `max.k` parameter is telling how many neighboring trees we want to account for.

```
# diameter differentiation:
# mingling

diff <- data.frame(treespat::DIFF(tree1, .x = x, .y = y, .mark = dbh, xmax = 100,
                                ymax = 100, max.k = 4, shape='square', .groups=c('year'))) |>
  dplyr::mutate(index="DIFF", name="Diameter differentiation", run=name1))
head(diff)
```

	year	DIFF index	name	run
1	0	0.4874231	DIFF Diameter differentiation	reference
2	1	0.4752327	DIFF Diameter differentiation	reference
3	2	0.4665189	DIFF Diameter differentiation	reference
4	3	0.4701394	DIFF Diameter differentiation	reference
5	4	0.4668349	DIFF Diameter differentiation	reference
6	5	0.4663594	DIFF Diameter differentiation	reference

```
ming <- data.frame(treespat::MING(tree1, .x = x, .y = y, .species = species,
                                xmax = 100, ymax = 100, max.k = 4, shape='square', .groups=c('year'))) |>
  dplyr::mutate(index="MING", name="Mingling", run=name1))

indices1<-rbind(diff |> dplyr::rename(value=DIFF),
  ming |> dplyr::rename(value=MINGLING ) )
```

2 Process based

We calculated each index as separate variable and then merged them into indices1. Here I renamed the column names to have it all “value”, for easier plotting later on. (each line has the information on which run and which index is the value) We do the same for the scenario simulation, we merge them together and make the plotting.

```
diff <- data.frame(treespat::DIFF(tree2, .x = x, .y = y, .mark = dbh,
  xmax = 100, ymax = 100, max.k = 4, shape = 'square',
  .groups = c('year')) |>
  dplyr::mutate(index = "DIFF", name = "Diameter differentiation",
    run = name2))

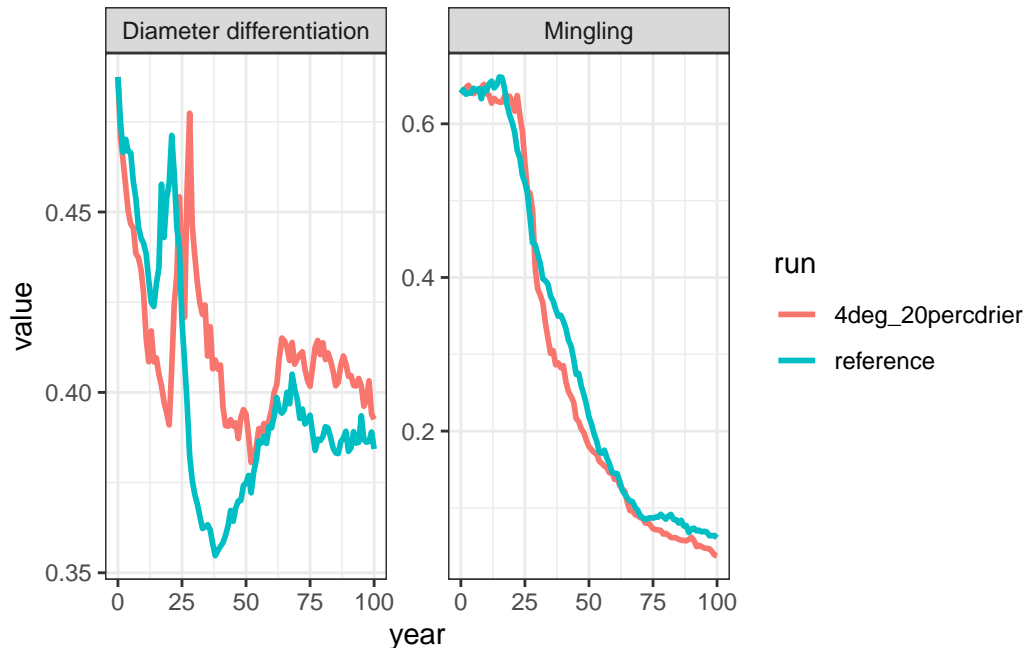
ming <- data.frame(treespat::MING(tree2, .x = x, .y = y, .species = species,
  xmax = 100, ymax = 100, max.k = 4, shape = 'square',
  .groups = c('year')) |>
  dplyr::mutate(index = "MING", name = "Mingling", run = name2))

indices2 <- rbind(diff |> dplyr::rename(value = DIFF),
  ming |> dplyr::rename(value = MINGLING))

indices <- rbind(indices1, indices2)

ggplot2::ggplot(indices, ggplot2::aes(x = year, y = value, color = run)) +
  ggplot2::geom_line(lwd = 1) +
  ggplot2::facet_wrap(~ name, ncol = 2, scales = "free") +
  ggplot2::theme_bw()
```

2 Process based



The diameter differentiation increases with the increasing average difference in diameter between neighboring trees. The value 0 indicates that neighboring trees have equal diameters. The development shows that the index values decrease during the first 20/25 years, after which it steeply increases due to the occurrence of ingrowth and then the values are stabilized. The values for the reference climate scenario are slightly higher than for the climate change scenario indicating slower growth under drier and warmer conditions and/or the differences in tree species composition (proportion of tree species). The mingling index is studying the neighboring trees regarding their species. In both development there is a decrease in the index after the occurrence of regeneration. Using the climate change scenario the index shows steeper decrease.

2.2.13 Question B - Group 4

- How are the species distribution and total living biomass C content changing in time on Plot 4? Compare 0 year and 100 year status in the reference case and in the case of your scenario(s)!

Run the model as it is described in the previous chapters to have at least 2 simulations completed: one with reference conditions and one scenario with some changes in the environment. Check if you have the output files in the output folder: *iLand_simulations/output/*. If you manage to simulate 2 scenarios, you can expand the code to have file3, and 3rd from each variable, or make the comparison in 2 step always comparing one

scenario to the reference conditions at the time. Open the *summerSchoolExercise.Rproj* project in R studio and start working there.

2.2.13.1 Read the landscape output table

Read in the *landscape* output table from two outputs you have for your plot:

```
path1 <- "model/iLand_simulations/output/Output_plot4.sqlite"
path2 <-
  "model/iLand_simulations/output/Output_plot4_4deg_20percdrier.sqlite"

file1 <- here::here(path1)
file2 <- here::here(path2)

# Give some name for the three simulations.
name1 <- "reference"
name2 <- "4deg_20percdrier"

# Read in data using the RSQLite package
sqlite.driver <- RSQLite::dbDriver("SQLite")

# We will work with "landscape" table and tree-scale data:
db1 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file1) # connect to the file1
landscape1 <- RSQLite::dbReadTable(db1, "landscape")
RSQLite::dbDisconnect(db1) # disconnect to the file1

db2 <-
  RSQLite::dbConnect(sqlite.driver, dbname = file2) # connect to the file2
landscape2 <- RSQLite::dbReadTable(db2, "landscape")
RSQLite::dbDisconnect(db2)
```

Merge the data from the three files together and make a column which tells which comes from which simulation, and study the table. The column “run” will support the plotting where we can use facets in ggplot.

```
landscape <- rbind(landscape1 |> dplyr::mutate(run = name1),
  landscape2 |> dplyr::mutate(run = name2))

head(landscape)
```

```
year area area_100m species count_ha dbh_avg_cm height_avg_m volume_m3
```

2 Process based

1	0	1	1	casa	152	12.96761	10.77155	9.465222
2	0	1	1	fasy	70	48.97035	27.82864	260.513423
3	0	1	1	pisu	33	52.05737	29.57999	91.870356
4	0	1	1	qupe	18	34.83232	17.83572	15.683124
5	1	1	1	casa	147	13.72713	11.31040	10.643544
6	1	1	1	fasy	70	50.20535	28.46379	268.968996
	total_carbon_kg		gwl_m3	basal_area_m2	NPP_kg	NPPabove_kg	LAI	
1	8041.268		9.465222	2.099428	0.000	0.00	0.3009711	
2	80391.775		260.513423	15.847768	0.000	0.00	1.3974759	
3	30694.307		91.870356	7.063300	0.000	0.00	0.8112550	
4	10327.265		15.683124	1.791511	0.000	0.00	0.1830485	
5	8550.914		10.987314	2.266547	2834.498	1765.12	0.3118849	
6	84739.791		268.968996	16.383716	13024.190	8553.20	1.4546398	
	cohort_count_ha		run					
1	0		reference					
2	0		reference					
3	0		reference					
4	0		reference					
5	0		reference					
6	229		reference					

2.2.13.2 Visualize changes in time

We can see that the output is given for each year and each species in our 1ha area. Plot the living carbon (total_carbon_kg) in time, coloring by species. We give a unified species coloring in the beginning. Plot number of trees and mean diameter. Carbon and number of trees can be plotted in an additive way, but for mean dbh we do line plot per species.

```
cols.all=c( "rops"="#e0e0e0","acpl"="#A9A9A9","alin"="#696969","alvi"="#2e2e2e",
  "bepe"="#fADFAD","casa"="#7eeADF","coav"="#20c6b6","tipl"="#645394",
  "ulgl"="#311432","saca"="#D8BFD8","soar"="#DDA0DD","soau"="#BA55D3",
  "pice"="#D27D2D","pini"="#a81c07","algl"="#2ECBE9","tico"="#128FC8",
  "potr"="#00468B","poni"="#5BAEB7","frex"="#fe9cb5","cabe"="#fe6181",
  "acps"="#fe223e","lade"="#FFFE71","abal"="#FFD800","pisy"="#A4DE02",
  "fasy"="#76BA1B","piab"="#006600","quro"="#FF7F00","qupe"="#FF9900",
  "qupu"="#CC9900")
```

```
# We set here the order of the run categories for the table, to have the first
#run first and then the second. (left-right of the plots)
landscape$run <- factor(landscape$run, levels = c(name1, name2))
```

2 Process based

```
# Plot the living carbon content, to have tonnes/ha, we divide total_carbon_kg by 1000.
g1 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, total_carbon_kg / 1000,
                                           fill = factor(species))) +

  ggplot2::geom_area() +
  ggplot2::scale_fill_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +

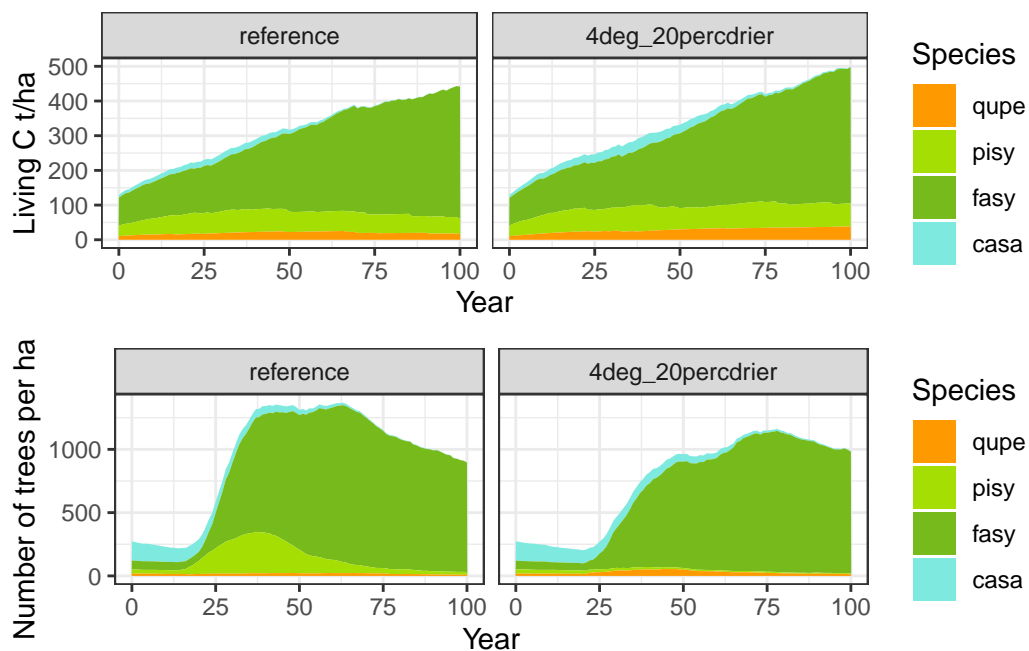
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Living C t/ha", fill = "Species") +
  ggplot2::theme_bw()

g2 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, count_ha,
                                           fill = factor(species))) +

  ggplot2::geom_area() +
  ggplot2::scale_fill_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse = TRUE)) +

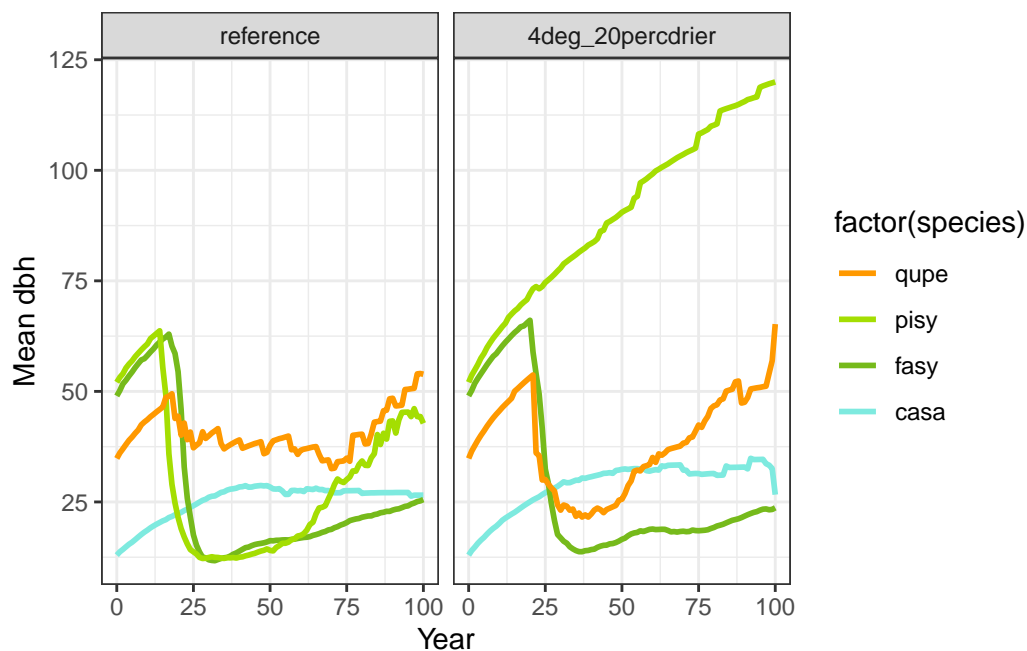
  ggplot2::facet_wrap( ~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Number of trees per ha", fill = "Species") +
  ggplot2::theme_bw()

gridExtra::grid.arrange(g1, g2, ncol = 1)
```



2 Process based

```
g3 <-
  ggplot2::ggplot(landscape, ggplot2::aes(year, dbh_avg_cm,
                                           color = factor(species))) +
  ggplot2::geom_line(lwd = 1) +
  ggplot2::scale_color_manual(values = cols.all,
                             guide = ggplot2::guide_legend(reverse =
TRUE))) +
  ggplot2::facet_wrap(~ run, nrow = 1) +
  ggplot2::labs(x = "Year", y = "Mean dbh", fill = "Species") +
  ggplot2::theme_bw()
print(g3)
```



Note that we do not have management intervention, and in the model trees shorter than 4m are not included in these outputs. However as they grow taller than 4m, model start to handle them as individual trees and including in these outputs which we are looking now.

At the end of the simulated 100 years, the total simulated carbon stock in living biomass under climate change scenario was slightly higher than under the reference climate. The proportion of beech has increased under both simulations. The number of trees increased in both cases after circa 25 years, but the increase was steeper under reference conditions. This increase caused by the regeneration.

Mean dbh of individual species reflected the development of the number of trees. The increase in the number of trees caused the reduction of mean dbh, e.g. of beech under both scenarios between 20 and 35 simulation years. The species-specific graphs for mean dbh indicates that some species do not have successful regeneration under the climate change scenario. Pine dbh decreased only under reference scenario, when we observed the increase of the number of trees, while under climate change scenario the mean dbh of pine was growing indicating the growth of remaining trees of the species at the plot. The small proportion of chestnut almost decrease to 0 under both scenarios, its dbh development does not show any successful regeneration processes.

2.2.13.3 Assess the stored carbon amount

Calculate the stored C amount in the initial year (year==0) and the last year (year==100)!

```
livingC0 <- data.frame(
  landscape |> dplyr::filter(year == 0) |>
    dplyr::group_by(run) |>
    dplyr::summarize(sum.livingC = sum(total_carbon_kg / 1000))
)
print(livingC0)
```

	run	sum.livingC
1	reference	129.4546
2	4deg_20percdrier	129.4546

```
livingC100 <- data.frame(
  landscape |> dplyr::filter(year == 100) |>
    dplyr::group_by(run) |>
    dplyr::summarize(sum.livingC = sum(total_carbon_kg / 1000))
)
print(livingC100)
```

	run	sum.livingC
1	reference	441.6930
2	4deg_20percdrier	499.1578

The initial conditions are same for the runs, but they are ending up at different C levels.

Calculate the stored C amount PER SPECIES in the initial year (year==0) and the last year (year==100)

2 Process based

```
species.livingC0 <- data.frame(  
  landscape |> dplyr::filter(year == 0) |>  
    dplyr::group_by(run, species) |>  
    dplyr::summarize(livingC = sum(total_carbon_kg / 1000))  
)  
print(species.livingC0)
```

	run	species	livingC
1	reference	casa	8.041268
2	reference	fasy	80.391775
3	reference	pisy	30.694307
4	reference	qupe	10.327265
5	4deg_20percdrier	casa	8.041268
6	4deg_20percdrier	fasy	80.391775
7	4deg_20percdrier	pisy	30.694307
8	4deg_20percdrier	qupe	10.327265

```
species.livingC100 <-  
  data.frame(  
    landscape |> dplyr::filter(year == 100) |>  
      dplyr::group_by(run, species) |>  
      dplyr::summarize(livingC = sum(total_carbon_kg / 1000))  
  )  
print(species.livingC100)
```

	run	species	livingC
1	reference	casa	0.2319933
2	reference	fasy	380.6997702
3	reference	pisy	44.2202121
4	reference	qupe	16.5410152
5	4deg_20percdrier	casa	1.5847763
6	4deg_20percdrier	fasy	392.4993454
7	4deg_20percdrier	pisy	66.8578057
8	4deg_20percdrier	qupe	38.2158285

2.2.13.4 Assess the species proportions

Calculate the species proportions based on the stored C amount in the initial year (year==0) and the last year (year==100) For this we need the total C amount and the species-specific C amount.

2 Process based

```
LC0 <- dplyr::left_join(species.livingC0, livingC0, by = "run")
LC0 <-
  data.frame(LC0 |> dplyr::mutate(spec.prop = livingC / sum.livingC, year =
                                0))
print(LC0)
```

	run	species	livingC	sum.livingC	spec.prop	year
1	reference	casa	8.041268	129.4546	0.06211650	0
2	reference	fasy	80.391775	129.4546	0.62100355	0
3	reference	pisy	30.694307	129.4546	0.23710477	0
4	reference	qupe	10.327265	129.4546	0.07977518	0
5	4deg_20percdrier	casa	8.041268	129.4546	0.06211650	0
6	4deg_20percdrier	fasy	80.391775	129.4546	0.62100355	0
7	4deg_20percdrier	pisy	30.694307	129.4546	0.23710477	0
8	4deg_20percdrier	qupe	10.327265	129.4546	0.07977518	0

```
LC100 <- dplyr::left_join(species.livingC100, livingC100, by = "run")
LC100 <-
  data.frame(LC100 |> dplyr::mutate(spec.prop = livingC / sum.livingC, year =
                                   100))
print(LC100)
```

	run	species	livingC	sum.livingC	spec.prop	year
1	reference	casa	0.2319933	441.6930	0.0005252366	100
2	reference	fasy	380.6997702	441.6930	0.8619103723	100
3	reference	pisy	44.2202121	441.6930	0.1001152679	100
4	reference	qupe	16.5410152	441.6930	0.0374491232	100
5	4deg_20percdrier	casa	1.5847763	499.1578	0.0031749006	100
6	4deg_20percdrier	fasy	392.4993454	499.1578	0.7863232430	100
7	4deg_20percdrier	pisy	66.8578057	499.1578	0.1339412339	100
8	4deg_20percdrier	qupe	38.2158285	499.1578	0.0765606224	100

Put the two tables together and visualize the results!

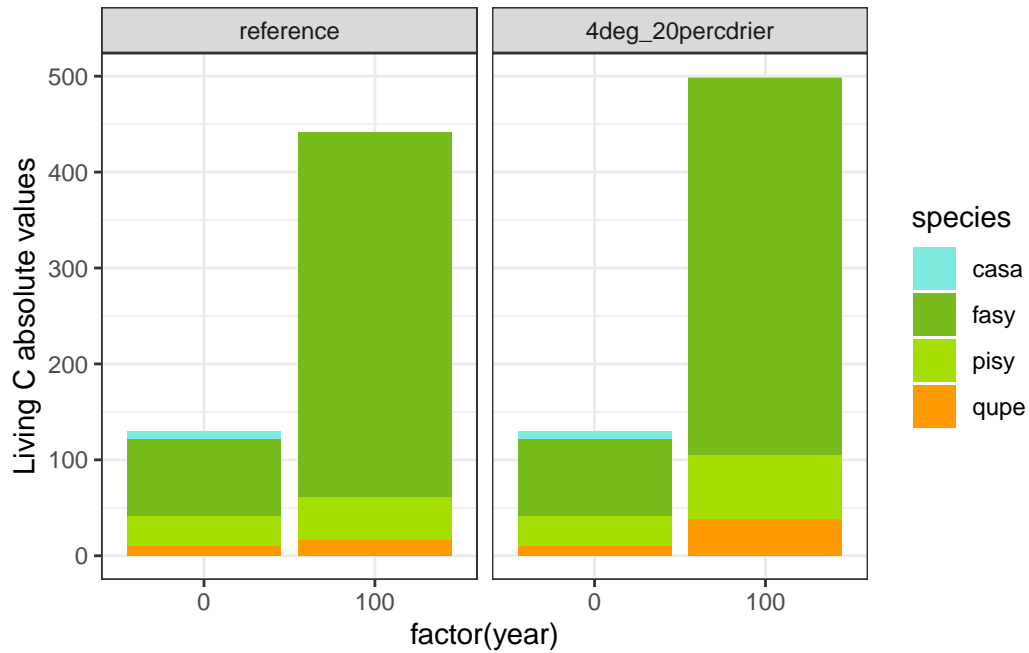
```
LC <- rbind(LC0, LC100)

LC$run <- factor(LC$run, levels = c(name1, name2))

ggplot2::ggplot(LC, ggplot2::aes(fill = species, y = livingC,
                                x = factor(year))) +
  ggplot2::geom_bar(position = "stack", stat = "identity") +
```

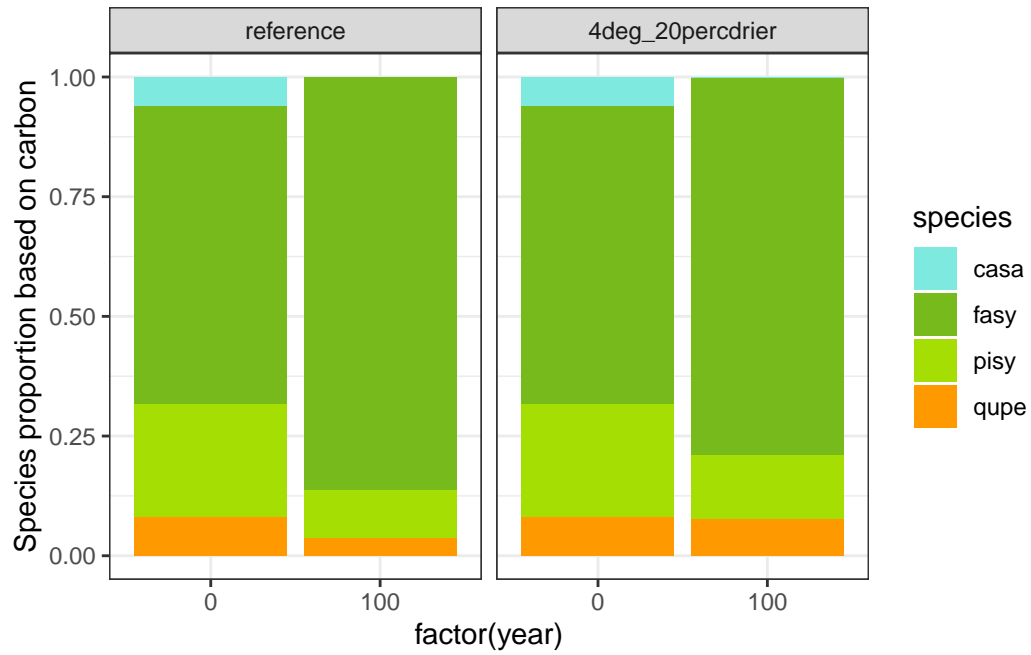
2 Process based

```
ggplot2::scale_fill_manual(values = cols.all) +
ggplot2::facet_wrap( ~ run) +
ggplot2::ylab("Living C absolute values") +
ggplot2::theme_bw()
```



```
ggplot2::ggplot(LC, ggplot2::aes(fill = species, y = spec.prop,
                                   x = factor(year))) +
ggplot2::geom_bar(position = "stack", stat = "identity") +
ggplot2::scale_fill_manual(values = cols.all) +
ggplot2::facet_wrap( ~ run) +
ggplot2::ylab("Species proportion based on carbon") +
ggplot2::theme_bw()
```

2 Process based



The living C stock tripled over 100 years due to the accumulation of biomass in trees and no management interventions and reached higher level under the climate change scenario. Both cases the proportion of beech increased.

3 Empirical modeling

In the empirical modelling we will cover two examples of creating two models based on observed data. We will use two approaches Generalized Linear Models (GLMs) and Boosted Regression trees (BRTs). Please take all the results with a grain of salt, we are making very generalized statements from a limited data set, and we are not getting into the details of how each of the models should be assessed; the goal here is that you learn how observed data can be used to create models that help you to understand the data and relationships better.

3.1 Data

We will work with one dataset derived from the inventory developed to assess forest structure and deadwood properties of six representative forest areas in the Czech Republic (Hošek, 2022). This dataset collects observations of the presence /absence of certain species of biodiversity importance across 99 plots.

The data was collected from square sampling plots (2500 m² each) in six forested Czech Republic regions. These regions are representative of the main bio-regions and elevation range of forests, considering their importance in territorial representation, forestry, and ecology. The inventory sampled for biodiversity variables such as the presence/absence of different species of birds, Tracheophyta, bryophytes, fungi, lichens, and beetles.

The dataset has been generously provided by Jeňýk Hofmeister at the Czech University of Life Sciences Prague (jenyk.hofmeister@email.cz) to be used in the context of this exercise. It should not be used for other purposes or be further distributed. If you want to use this data or learn more about it, please get in touch with Jeňýk Hofmeister here: jenyk.hofmeister@email.cz.

The data you can download for this exercise is an aggregated summary of the original data, some aspects have been modified from the observed data, so the results you will obtain will only partially match the observed reality. This data is very similar in structure and observed variables to the one you collected in this summer school. The idea is to move from data collection, analysis to this part, where you use the data to create a model.

3.1.1 Data description

These are the variables available in the data

- **longitud**: longitude of the plot location
- **latitude**: latitude of the plot location
- **forestManagementType**: forest management type applied in this plot
- **forestStructure**: current forest structure in the plot
- **slope**: slope of the plot
- **A.pseudoplatanus**: proportion of this tree species in the plot in volume
- **F.sylvatica**: proportion of this tree species in the plot in volume
- **L.decidua**: proportion of this tree species in the plot in volume
- **Q.robur**: proportion of this tree species in the plot in volume
- **S.aucuparia** : proportion of this tree species in the plot in volume
- **B.pendula**: proportion of this tree species in the plot in volume
- **P.abies**: proportion of this tree species in the plot in volume
- **P.sylvestris**: proportion of this tree species in the plot in volume
- **F.excelsior**: proportion of this tree species in the plot in volume
- **A.alba**: proportion of this tree species in the plot in volume
- **A.platanoides**: proportion of this tree species in the plot in volume
- **T.cordata**: proportion of this tree species in the plot in volume
- **S.racemosa**: proportion of this tree species in the plot in volume
- **U.glabra**: proportion of this tree species in the plot in volume
- **S.nigra**: proportion of this tree species in the plot in volume
- **P.alba**: proportion of this tree species in the plot in volume
- **U.minor**: proportion of this tree species in the plot in volume
- **S.caprea**: proportion of this tree species in the plot in volume
- **C.betulus**: proportion of this tree species in the plot in volume
- **P.nigra**: proportion of this tree species in the plot in volume
- **Q.petraea**: proportion of this tree species in the plot in volume
- **S.torminalis**: proportion of this tree species in the plot in volume
- **A.campestre**: proportion of this tree species in the plot in volume
- **P.strobus**: proportion of this tree species in the plot in volume
- **Q.rubra**: proportion of this tree species in the plot in volume
- **volAllha**: total volume in the plot
- **GiniDBH**: Gini index calculated to assess the forest structural diversity in diameter sizes, higher values indicate more structural heterogeneity; lower values indicate more homogeneous stands
- **ShannonIndexTreeSpp**: The Shannon index is way to measure the diversity of tree species in the plot
- **Tracheophyta_rich**: Species richness across Tracheophyta species
- **Birds_rich**: Species richness across Birds species
- **Bryophytes_rich** : Species richness across Bryophytes species
- **Fungi_rich**: Species richness across Fungi species

- **Lichens_rich**: Species richness across Lichens species
- **Beetles_rich**: Species richness across Beetles species
- **dendrocoposMajor**: presence / absence of *Dendrocopos major*
- **certhia**: presence / absence of *Certhia familiaris*
- **bryophitaNumObs**: number of observed Bryophytes species
- **birdNumObs**: number of observed Bird species
- **PlotID**: ID number for the plot

3.2 Species description

3.2.1 Species 1 - Bryophytes

Bryophytes constitute an important and permanent component of the forest flora and diversity. They colonize various substrates, which are unsuitable for vascular plants, because of low light intensity or low nutrient level, such as deadwood, bark, rocks, and open soil. They provide shelter habitats, food, and nest material for many animals.

In forests, different ecological guilds of Bryophytes can be distinguished by the substrate on which they are growing, including terricolous, lignicolous, corticolous and saxicolous species that occur on soil, deadwood, bark of living trees and shrubs, or rocks, respectively. As diversity and quality of these substrates is affected by forest management, Bryophytes are suitable indicators for the effect of management on forest conditions. Especially typical woodland Bryophytes, which are strictly depending on forest conditions. It is interesting to better understand the relation of forest management effects on Bryophytes and some studies have already demonstrated their sensitivity to management practices.

3.2.2 Species 2 - Great spotted woodpecker

The great spotted woodpecker (*Dendrocopos major*) is a medium-sized woodpecker with pied black and white plumage and a red patch on the lower belly. It is found in a wide variety of woodlands, broadleaf, coniferous or mixed forests. The great spotted woodpecker spends much of its time climbing trees. It a quite generalist bird species.

3.2.3 Species 3 - Eurasian treecreeper

The Eurasian treecreeper or common treecreeper (*Certhia familiaris*) is a small passerine bird. It prefers mature trees, and in most of Europe, it tends to be found mainly in coniferous forest, especially spruce and fir.

3.3 Models

3.3.1 Generalized Linear Models (GLMs)

We propose to use a generalized linear model (GLM) to understand the abundance of different Bryophytes species in the 99 plots. Count data often conform to a Poisson distribution; in this case, we have a count of the number of species recorded at each plot.

Fitting a Poisson GLM in R is similar to analyzing covariance (or linear model), except that we now need to use the `glm` function. To run a GLM, we need to provide one extra piece of information beyond that required for a linear model: the family of models we want to use. In this case, we want a Poisson family, `family=poisson`.

3.3.2 Boosted regression trees (BRTs)

Boosted regression trees (BRT) are a combination of two powerful statistical techniques: boosting and regression trees. Boosting is a machine learning technique similar to model averaging, where the results of several competing models are merged. Unlike model averaging, however, boosting uses a forward, stage-wise procedure, where tree models are fitted iteratively to a subset of the training data. Subsets of the training data used at each iteration of the model fit are randomly selected without replacement, where the proportion of the training data used is determined by the modeler, this is defined with the “bag fraction” parameter. This procedure, known as stochastic gradient boosting, introduces an element of stochasticity that improves model accuracy and reduces overfitting (Elith et al., 2008).

The BRT model calibration is defined by four parameters:

- the **learning rate** (or shrinkage parameter): The learning rate determines the contribution of each new tree to the growing model, and it is always substantially lower than 1, higher values being related to faster learning.
- the **bag fraction**: The bag fraction provides information on which fraction of the entire data should be drawn randomly to fit the new tree. This parameter includes a random probabilistic component, making each run model different, and is aimed at improving model accuracy, speed of model creation, and the reduction of overfitting (Friedman et al., 2008).
- the **tree complexity**: Tree complexity controls the number of fitted interactions among variables, and determines the number of splits in each tree; for example, a value of 1 will present only one split, meaning that the model does not consider interactions; a value of 2 will result in two splits, and two interactions.

- The **number of trees** required for optimal prediction: The optimal number of trees is selected based on the three previous parameters. The values fitted by the final model are computed as the sum of all of the predictions of the trees, multiplied by their respective learning rates.

3.4 The exercise

3.4.1 The project folder

All the project is available in a github repository. You can download the project with this document, code, the **.rproj** and the correct folder structure in here [2].

[2] <https://github.com/oldiya/summerSchoolExercise>

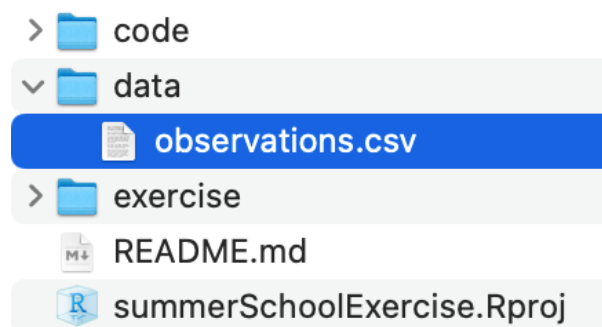
In this project you will find a folder called **code** where the codes for each question are stored. A folder called **data** is where you have to store the data that you have downloaded. A folder called **exercise** contains this document and all the required files to build it.

3.4.2 Download the data

You can download the data in here [1]

[1] <https://polybox.ethz.ch/index.php/s/qERYKjzmFTr81Sq>

Our recommendation is that you follow this folder structure and you put the data in the folder **data** of the project and the **.rproj** into the main folder. If you decide to organize things in a different way then you will have to change the path to the code provided to load the data in the section “Explore the data”



3.4.3 Explore the data

You can load the data in R by doing:

```
observations <- read.csv(here::here("data/observations.csv"))
```

Once you have obtained the data, your next step is to explore it. It is important to carefully analyze the structure of the data and understand its meaning. Each variable must be examined closely, along with the data structure. Remember that each row in this dataset represents one plot, identified by its `PlotID`, and each column represents one variable.

You can explore each of the variables by doing:

```
str(observations)
```

```
'data.frame':  99 obs. of  45 variables:
 $ X                : int  1 2 3 4 5 6 7 8 9 10 ...
 $ longitud          : num  13.5 13.5 13.5 13.6 13.6 ...
 $ latitude          : num  49.5 49.5 49.5 49.5 49.5 ...
 $ forestManagementType: chr  "simple clearcutting" "simple clearcutting" "simple clearcutting" ...
 $ forestStructure    : chr  "even-aged" "even-aged" "even-aged" "even-aged" ...
 $ slope             : num  16.85 6.51 4.91 5.55 14.55 ...
 $ A.pseudoplatanus  : int  0 0 0 0 0 4 1 7 16 0 ...
 $ F.sylvatica        : int  61 0 0 100 0 82 85 91 75 0 ...
 $ L.decidua          : int  38 0 0 0 4 6 0 0 0 0 ...
 $ Q.robur            : int  0 1 0 0 0 0 0 0 0 0 ...
 $ S.aucuparia        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ B.pendula          : int  0 0 1 0 28 0 0 0 0 0 ...
 $ P.abies            : int  0 99 93 0 30 0 0 0 8 99 ...
 $ P.sylvestris       : int  0 0 7 0 1 0 0 0 0 0 ...
 $ F.excelisior       : int  0 0 0 0 38 0 0 0 0 0 ...
 $ A.alba             : int  0 0 0 0 0 4 0 0 0 1 ...
 $ A.platanoides      : int  0 0 0 0 0 2 2 2 0 0 ...
 $ T.cordata          : int  0 0 0 0 0 2 12 0 0 0 ...
 $ S.racemosa         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ U.glabra           : int  0 0 0 0 0 0 0 0 1 0 ...
 $ S.nigra            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ P.alba             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ U.minor            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ S.caprea           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ C.betulus          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ P.nigra            : int  0 0 0 0 0 0 0 0 0 0 ...
```

3 Empirical modeling

```
$ Q.petraea      : int  0 0 0 0 0 0 0 0 0 0 ...
$ S.torminalis   : int  0 0 0 0 0 0 0 0 0 0 ...
$ A.campestre    : int  0 0 0 0 0 0 0 0 0 0 ...
$ P.strobus      : int  0 0 0 0 0 0 0 0 0 0 ...
$ Q.rubra        : int  0 0 0 0 0 0 0 0 0 0 ...
$ volAllha       : num  592 377 438 615 194 ...
$ GiniDBH        : num  0.448 0.416 0.12 0.133 0.378 ...
$ ShannonIndexTreeSpp : num  0.68 0.07 0.28 0 1.28 0.78 0.54 0.39 0.75 0.06 ...
$ Tracheophyta_rich : num  0.218 0.233 0.209 0.189 0.354 ...
$ Birds_rich     : num  0.358 0.46 0.485 0.307 0.383 ...
$ Bryophytes_rich : num  0.0876 0.0876 NA 0.2629 0.3505 ...
$ Fungi_rich     : num  0.199 0.133 0.114 0.218 0.262 ...
$ Lichens_rich   : num  0.0659 0.1976 0.1318 0.1537 0.1537 ...
$ Beetles_rich   : num  0.16 0.16 0.234 0.258 0.221 ...
$ dendrocoposMajor : int  1 1 1 0 1 1 1 1 1 1 ...
$ certhia        : int  0 0 0 0 0 0 1 0 1 0 ...
$ bryophitaNumObs : int  1 1 0 3 4 5 7 4 7 1 ...
$ birdNumObs     : int  14 18 19 12 15 14 12 14 18 18 ...
$ PlotID         : int  1 2 3 4 5 6 7 8 9 10 ...
```

You have a description of each of the variables in the section Section 3.1.1.

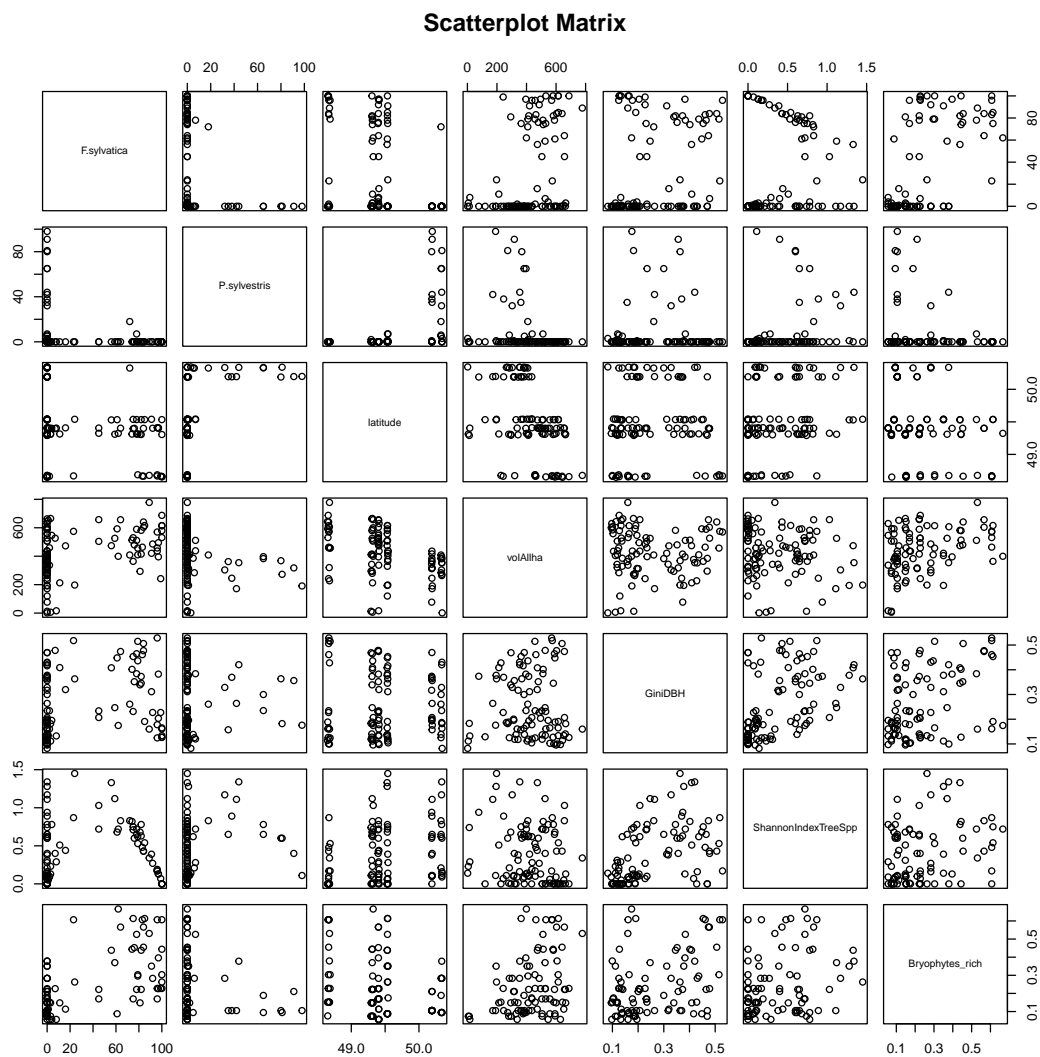
You could do further analysis by exploring the data correlations and you can even plot them to explore their values and ranges better. During this process you should start thinking:

- What am I trying to understand with this model? (your question)
- What variables are important to answer my question?

You could for example explore the behavior of the variables by plotting a scatterplot:

```
pairs(
  ~ F.sylvatica + P.sylvestris + latitude + volAllha + GiniDBH +
    ShannonIndexTreeSpp + Bryophytes_rich,
  data = observations,
  main = "Scatterplot Matrix"
)
```

3 Empirical modeling



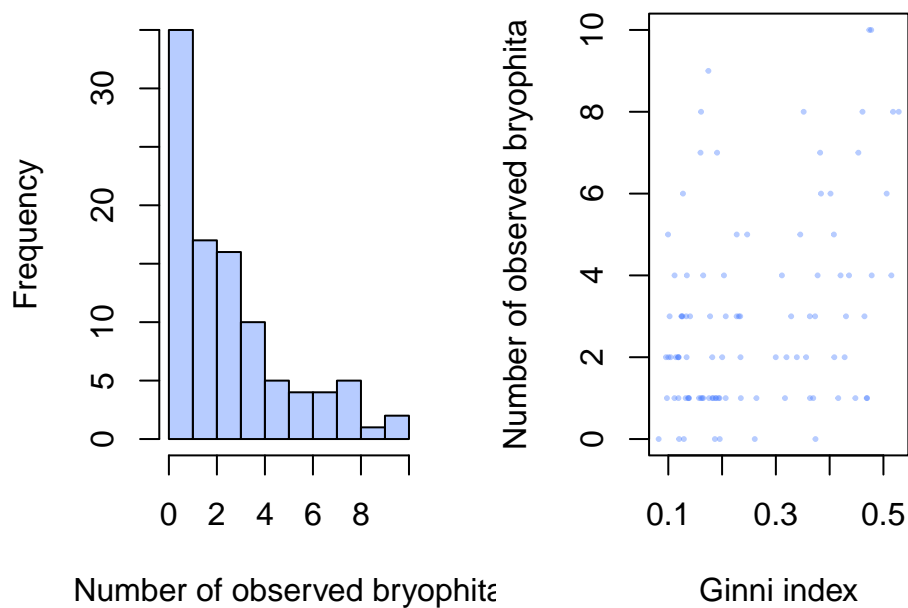
You can also create individual scatterplots or histograms for a variable of interest, for example:

```
#Divide the screen in 1 line and 3 columns
par(mfrow = c(1, 2), oma = c(0, 2, 0, 0))

#Make the margin around each graph a bit smaller
par(mar = c(4, 4, 2, 2))
# Histogram and Scatterplot
hist(
  observations$bryophitaNumObs,
```

3 Empirical modeling

```
main = "",  
breaks = 10,  
col = rgb(0.3, 0.5, 1, 0.4) ,  
xlab = "Number of observed bryophita"  
)  
plot(  
  y = observations$bryophitaNumObs,  
  x = observations$GiniDBH,  
  main = "" ,  
  pch = 20,  
  cex = 0.4,  
  col = rgb(0.3, 0.5, 1, 0.4),  
  xlab = "Ginni index",  
  ylab = "Number of observed bryophita"  
)
```



3.4.4 Question C - Group 1

- Does a more diverse forest in structure and composition have more Bryophytes species?

3.4.4.1 Fitting a Poisson GLM in R

During your data exploration, you should have selected your response variable, `bryophitaNumObs`. In this case, since we are trying to understand forest structure and composition, you should also choose explanatory variables for that, such as `GiniDBH`, which indicates the forest structural diversity in diameter sizes; higher values indicate more structural heterogeneity, and lower values indicate more homogeneous stands, or the `ShannonIndexTreeSpp` which assesses the diversity of tree species in the plot.

We could start by only looking at how the forest structural diversity affects the number of Bryophyte species that we have in a plot. You can create a GLM model with `bryophitaNumObs` as the response variable and `GiniDBH` as an explanatory variable. You will use the function `glm` in R and the `family = poisson`. You can see how to create and see this model here:

```
bryoModel1 <- glm(bryophitaNumObs ~ GiniDBH,
                  family = poisson,
                  data = observations)

summary(bryoModel1)
```

Call:

```
glm(formula = bryophitaNumObs ~ GiniDBH, family = poisson, data = observations)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	0.4627	0.1388	3.333	0.000859	***
GiniDBH	2.2797	0.4221	5.401	6.64e-08	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 188.58 on 98 degrees of freedom
 Residual deviance: 159.92 on 97 degrees of freedom
 AIC: 424.2

Number of Fisher Scoring iterations: 5

Here you can see that the coefficient table produced by a GLM is very similar to a linear model. The intercept tells us the estimated value of the response variable when the continuous explanatory variables (here, just the Gini index) has a value of 0. We

3 Empirical modeling

then also have coefficients describing the slope of the relationship with our continuous explanatory variables. We can see here that the number of Bryophytes species appears to show a positive relationship with the Gini index, which means that increasing structural diversity in tree diameter sizes has a positive relationship with the number of Bryophytes species in the plot.

We are also interested in understanding the relationship of the number of Bryophytes species and the tree species diversity; we can now try to add this variable into the model and see if it helps us to understand things. You can do that by adding the variable `ShannonIndexTreeSpp` to the model:

```
bryoModel2 <- glm(bryophitaNumObs ~ GiniDBH + ShannonIndexTreeSpp,
                  family = poisson,
                  data = observations)

summary(bryoModel2)
```

Call:

```
glm(formula = bryophitaNumObs ~ GiniDBH + ShannonIndexTreeSpp,
     family = poisson, data = observations)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	0.45277	0.14060	3.220	0.00128 **
GiniDBH	2.17411	0.46413	4.684	2.81e-06 ***
ShannonIndexTreeSpp	0.09209	0.16202	0.568	0.56977

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 188.58 on 98 degrees of freedom
Residual deviance: 159.60 on 96 degrees of freedom
AIC: 425.88

Number of Fisher Scoring iterations: 5

Here we can see that the variable Shannon index also has a positive relationship with the number of Bryophytes species in the plot, but its effect is much smaller. We can also observe that this variable is not significant. This does not mean it is wrong to add this variable because we want to understand its effect, but keeping this variable in the model depends on what you're trying to do and what "reality" is. Adding variables that

3 Empirical modeling

are not significant will not help your model (particularly your estimates) but also might not matter much (e.g., predictions). However, removing actual variables can create a useless model even if they don't meet significance.

Variable selection is a long and complicated topic. Some general rules of thumb include: (1) Include the variable if it is of interest, (2) Include the variable if you have some prior knowledge that it should be relevant. This can be misleading because it's a confirmation bias, but in most cases, this makes sense. (3) If you want a model that can generalize many cases, you should favor fewer variables.

3.4.4.2 Explanatory Power of the model

When we ran linear models, we used the coefficient of determination, or R^2 to assess how much of the variability in our response variable is explained by a given model. R^2 is based on the sums of squares of our model, and so cannot be calculated for GLMs. Instead, we can calculate the deviance explained by our model:

```
# Extract the null and residual deviance from the model
dev.null <- bryoModel11$null.deviance
dev.resid <- bryoModel11$deviance

# Calculate the deviance explained by the model
dev.explained <- (dev.null - dev.resid) / dev.null

# Round to 3 decimal places
dev.explained <- round(dev.explained, 3)

dev.explained
```

```
[1] 0.152
```

Variability in forest structure (Gini index) explain 15% of the variation in Bryophytes species richness in this study. That is an ok explanatory power for a very simple model of a complex ecological system (many factors determine the species richness for Bryophytes and we are attempting to explain everything with just one variable).

3.4.4.3 Model Assumptions

For Poisson GLMs, there is one further assumption that we have not encountered before. If data follow a Poisson distribution, then the mean of the distribution is equal to the variance. Accordingly, a Poisson distribution is represented by just one parameter λ , which describes both the mean and the variance of the distribution.

3 Empirical modeling

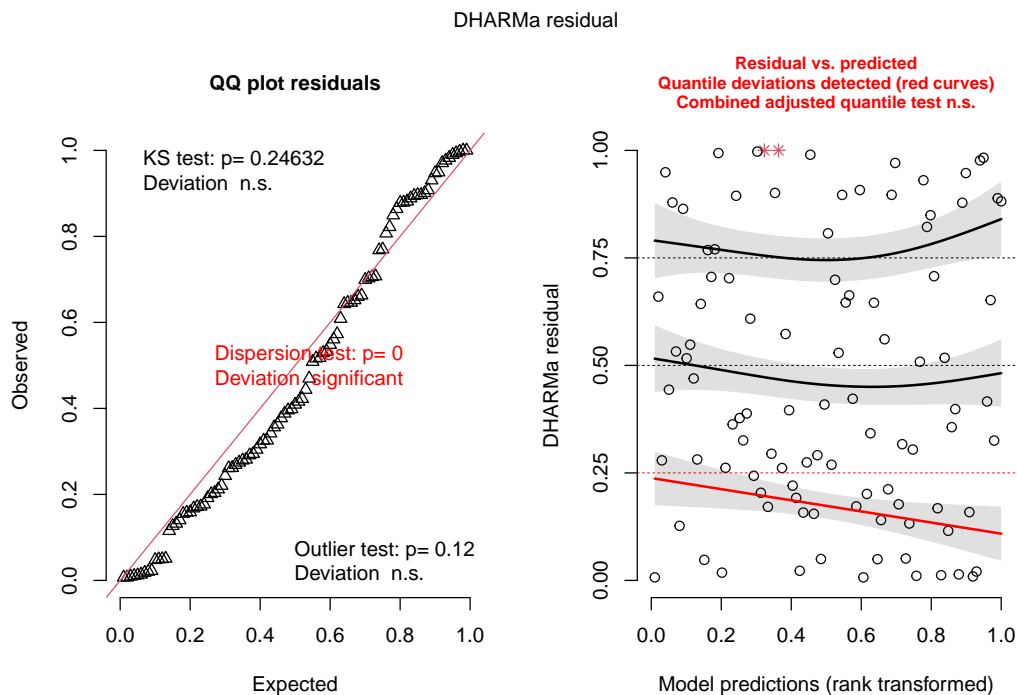
Count data in ecology are often **overdispersed**, where the variance is greater than the mean. This violates the assumption of a Poisson GLM, and means that any statistics that we calculate from the model may be unreliable.

We can get look whether a model is over-dispersed by inspecting the model summary as you did in Section 3.4.4.1. As a rule of thumb, if the response variable conforms to a true Poisson distribution, we expect the residual deviance to be approximately equal to the residual degrees of freedom. If the deviance is much greater than the degrees of freedom, this indicates over-dispersion. This is the case in our models (see outputs from running the code in Section 3.4.4.1).

To check the model assumptions in a GLM is not as straight forward as with a linear model. This is because classical residuals are not expected to behave in the same way for GLMs. We can use the DHARMA package in R for working with GLMs, which uses a simulation-based approach to compare the residuals from the actual model with the expectation if the model is behaving normally:

```
# Simulate residuals
simResids <- DHARMA::simulateResiduals(bryoModel1)

# Generate plots to compare the model residuals to expectations
plot(simResids)
```



3 Empirical modeling

These plots show us that this model is not behaving as we would expect in terms of homogeneity of variance and distribution of residuals. A follow up to this would be to try alternatives to deal with over-dispersed count data in GLMs such as fit a quasi-Poisson GLM or a negative binomial GLM. Unfortunately we do not have time to continue in this exercise.

3.4.5 Question D - Group 1

- Is the number of Bryophytes species affected by forest management type and the forest structural diversity?

3.4.5.1 Fitting a Poisson GLM in R

Fitting a Poisson GLM in R is very similar to fitting an analysis of covariance (or linear model), except that now we need to use the `glm` function. To run a GLM, we need to provide one extra piece of information beyond that needed for a linear model: the family of model we want to use. In this case, we want a Poisson family.

We could start by only looking at how the forest structural diversity affects the number Bryophytes species that we have in a plot. You can do this by creating a GLM model which has `bryophitaNumObs` as response variable and `GiniDBH` as explanatory variable. For that you will use the function `glm` in R and use the `family = poisson`. You can see how to create and see this model here:

```
bryoModel1 <- glm(bryophitaNumObs ~ GiniDBH,
                  family = poisson,
                  data = observations)

summary(bryoModel1)
```

Call:

```
glm(formula = bryophitaNumObs ~ GiniDBH, family = poisson, data = observations)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	0.4627	0.1388	3.333	0.000859	***
GiniDBH	2.2797	0.4221	5.401	6.64e-08	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

3 Empirical modeling

Null deviance: 188.58 on 98 degrees of freedom
Residual deviance: 159.92 on 97 degrees of freedom
AIC: 424.2

Number of Fisher Scoring iterations: 5

Here you can see that the coefficient table produced by a GLM is very similar to a linear model. The intercept tells us the estimated value of the response variable when the continuous explanatory variables (here just Gini index) have a value of 0. We then also have coefficients describing the slope of the relationship with our continuous explanatory variables. We can see here that the bryophita numbers appears to show a positive relationship with Gini index, which means that increasing structural diversity in trees diameter sizes has a positive relationship with the number of bryophita species in the plot.

We are also interested in understanding the relationship of the number of observed Bryophytes species and forest management, we can now try to add this variable into the model.

```
bryoModel2 <- glm(bryophitaNumObs ~ forestManagementType,  
                  family = poisson,  
                  data = observations)  
  
summary(bryoModel2)
```

Call:

```
glm(formula = bryophitaNumObs ~ forestManagementType, family = poisson,  
     data = observations)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	0.9651	0.1543	6.254	3.99e-10
forestManagementTypesimple clearcutting	-0.1335	0.1750	-0.763	0.445
forestManagementTypeunmanaged	0.7633	0.1821	4.192	2.76e-05

```
(Intercept) ***  
forestManagementTypesimple clearcutting  
forestManagementTypeunmanaged ***  
---
```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

3 Empirical modeling

(Dispersion parameter for poisson family taken to be 1)

```
Null deviance: 188.58  on 98  degrees of freedom
Residual deviance: 141.54  on 96  degrees of freedom
AIC: 407.82
```

```
Number of Fisher Scoring iterations: 5
```

The intercept here tells us the estimated value of the response variable when the reference groups in our grouping (categorical) variables (here for retention clear-cutting). We then also have coefficients describing the slope of the relationship with our continuous explanatory variables, and coefficients giving the estimated difference in the response variable for non-reference groupings. We can see here that number of bryophytes species appears to show a negative relationship with simple clearcutting, and appears to have a positive relationship with unmanaged and retention clear-cutting.

The type simple clearcutting appears to be non significant, which only tells us about the pairwise differences between the levels. To test whether the categorical predictor, as a whole, is significant is equivalent to testing whether there is any heterogeneity in the means of the levels of the predictor. When there are no other predictors in the model, this is a classical ANOVA problem.

3.4.5.2 Explanatory Power of the model

When we run linear models, we use the coefficient of determination, or R^2 to assess how much of the variability in our response variable is explained by a given model. R^2 is based on the sums of squares of our model, and so it cannot be calculated for GLMs. Instead, we can calculate the the deviance explained by our model:

```
# Extract the null and residual deviance from the model
dev.null <- bryoModel1>null.deviance
dev.resid <- bryoModel1$deviance

# Calculate the deviance explained by the model
dev.explained <- (dev.null - dev.resid) / dev.null

# Round to 3 decimal places
dev.explained <- round(dev.explained, 3)

dev.explained
```

```
[1] 0.152
```

3 Empirical modeling

Variability in forest structure (Gini index) explains 15% of the variation in bryophita species richness in this study system. That is an ok explanatory power for a very simple model of a complex ecological system (many factors determine the species richness of bryophitas and we are attempting to explain everything with one variable).

3.4.5.3 Model Assumptions

For Poisson GLMs, there is one further assumption that we have not encountered before. If data follow a Poisson distribution, then the mean of the distribution is equal to the variance. Accordingly, a Poisson distribution is represented by just one parameter λ , which describes both the mean and the variance of the distribution.

Count data in ecology are often *overdispersed*, where the variance is greater than the mean. This violates the assumption of a Poisson GLM, and means that any statistics that we calculate from the model may be unreliable.

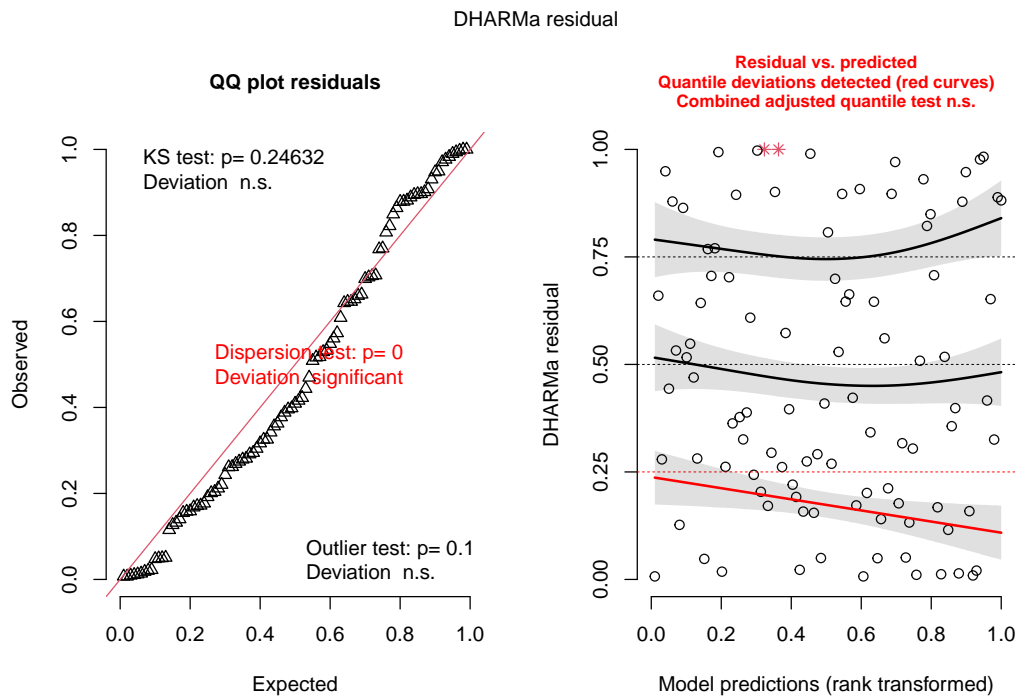
We can get look whether a model is over-dispersed by inspecting the model summary as you did in Section 3.4.5.1. As a rule of thumb, if the response variable conforms to a true Poisson distribution, we expect the residual deviance to be approximately equal to the residual degrees of freedom. If the deviance is much greater than the degrees of freedom, this indicates over-dispersion. This is the case in our models (see outputs from running the code in Section 3.4.5.1).

To check the model assumptions in a GLM is not as straight-forward as with a linear model. This is because classical residuals are not expected to behave in the same way for GLMs. We can use the DHARMA package in R for working with GLMs, which uses a simulation-based approach to compare the residuals from the actual model with the expectation if the model is behaving normally.

```
# Simulate residuals
simResids <- DHARMA::simulateResiduals(bryoModel1)

# Generate plots to compare the model residuals to expectations
plot(simResids)
```

3 Empirical modeling



These plots show us that this model is not behaving as we would expect in terms of homogeneity of variance and distribution of residuals. A follow up to this would be to try alternatives to deal with over-dispersed count data in GLMs such as fit a quasi-Poisson GLM or a negative binomial GLM. Unfortunately we do not have to continue in this exercise.

3.4.6 Question C - Group 2

- Does a more diverse forest in structure and composition have more bird species?

3.4.6.1 Fitting a Poisson GLM in R

During your data exploration you should have selected your response variable: `birdNumObs`. In this case since we are trying to understand forest structure and composition you should also select explanatory variables for that such as `GiniDBH` which indicates the forest structural diversity in diameter sizes, higher values indicate more structural heterogeneity and lower values indicate more homogeneous stands, or the `ShannonIndexTreeSp` which assess the diversity of tree species in the plot.

3 Empirical modeling

We could start by only looking at how the forest structural diversity affects the number bird species that we have in a plot. You can do this by creating a GLM model which has `birdNumObs` as response variable and `GiniDBH` as explanatory variable. For that you will use the function `glm` in R and use the `family = poisson`. You can see how to create and see this model here:

```
birdModel1 <- glm(birdNumObs ~ GiniDBH,
                  family = poisson,
                  data = observations)

summary(birdModel1)
```

Call:

```
glm(formula = birdNumObs ~ GiniDBH, family = poisson, data = observations)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.63827	0.05636	46.809	<2e-16 ***
GiniDBH	0.42725	0.19030	2.245	0.0248 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 59.277 on 98 degrees of freedom
Residual deviance: 54.280 on 97 degrees of freedom
AIC: 511.57

Number of Fisher Scoring iterations: 4

Here you can see that the coefficient table produced by a GLM is very similar to a linear model. The intercept tells us the estimated value of the response variable when the continuous explanatory variables (here just Gini index) has a value of 0. We then also have coefficients describing the slope of the relationship with our continuous explanatory variables. We can see here that the number of bird species appears to show a positive relationship with Gini index, which means that increasing structural diversity in trees diameter sizes has a positive relationship with the number of bird species in the plot.

We are also interested in understanding the relationship of the number of bird species and the trees species diversity, we can now try to add this variable into the model and see if it help us to understand things. You can do that by adding the variable `ShannonIndexTreeSpp` into the model

3 Empirical modeling

```
birdModel2 <- glm(birdNumObs ~ GiniDBH + ShannonIndexTreeSpp,  
                  family = poisson,  
                  data = observations)  
  
summary(birdModel2)
```

Call:

```
glm(formula = birdNumObs ~ GiniDBH + ShannonIndexTreeSpp, family = poisson,  
     data = observations)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.63499	0.05665	46.513	<2e-16 ***
GiniDBH	0.33323	0.21661	1.538	0.124
ShannonIndexTreeSpp	0.06923	0.07444	0.930	0.352

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 59.277 on 98 degrees of freedom
Residual deviance: 53.419 on 96 degrees of freedom
AIC: 512.71

Number of Fisher Scoring iterations: 4

Here we can see that the variable Shannon index also has a positive relationship with the number of bird species in the plot but its effect is much smaller. We can also observe that this variable is not significant and that including this variable also made Gini index variable not significant. This does not mean that is wrong to add this variable, because we want to understand its effect, but keeping this variable in the model or not depends on what you're trying to do, and what “reality” is. Adding variables that are not needed will not help your model (particularly your estimates), but also might not matter much (e.g. predictions). However, removing variables that are real, even if they don't meet significance, can create a useless model.

Variable selection is a long and complicated topic. some general rules of thumb include: (1) Include the variable if it is of interest, (2) Include the variable if you have some prior knowledge that it should be relevant. This can be misleading, because it's a confirmation bias, but in most cases this makes sense. (3) If you want a model that can generalize to many cases, you should favor fewer variables.

3.4.6.2 Explanatory Power of the model

When we ran linear models, we used the coefficient of determination, or R^2 to assess how much of the variability in our response variable is explained by a given model. R^2 is based on the sums of squares of our model, and so cannot be calculated for GLMs. Instead, we can calculate the the deviance explained by our model:

```
# Extract the null and residual deviance from the model
dev.null <- birdModel1$null.deviance
dev.resid <- birdModel1$deviance

# Calculate the deviance explained by the model
dev.explained <- (dev.null - dev.resid) / dev.null

# Round to 3 decimal places
dev.explained <- round(dev.explained, 3)

dev.explained
```

```
[1] 0.084
```

Variability in forest structure (Gini index) explain 8% of the variation in bird species richness in this study. That is an ok explanatory power for a very simple model of a complex ecological system (many factors determine the species richness for birds and we are attempting to explain everything with just one variable).

3.4.6.3 Model Assumptions

For Poisson GLMs, there is one further assumption that we have not encountered before. If data follow a Poisson distribution, then the mean of the distribution is equal to the variance. Accordingly, a Poisson distribution is represented by just one parameter λ , which describes both the mean and the variance of the distribution.

Count data in ecology are often **overdispersed**, where the variance is greater than the mean. This violates the assumption of a Poisson GLM, and means that any statistics that we calculate from the model may be unreliable.

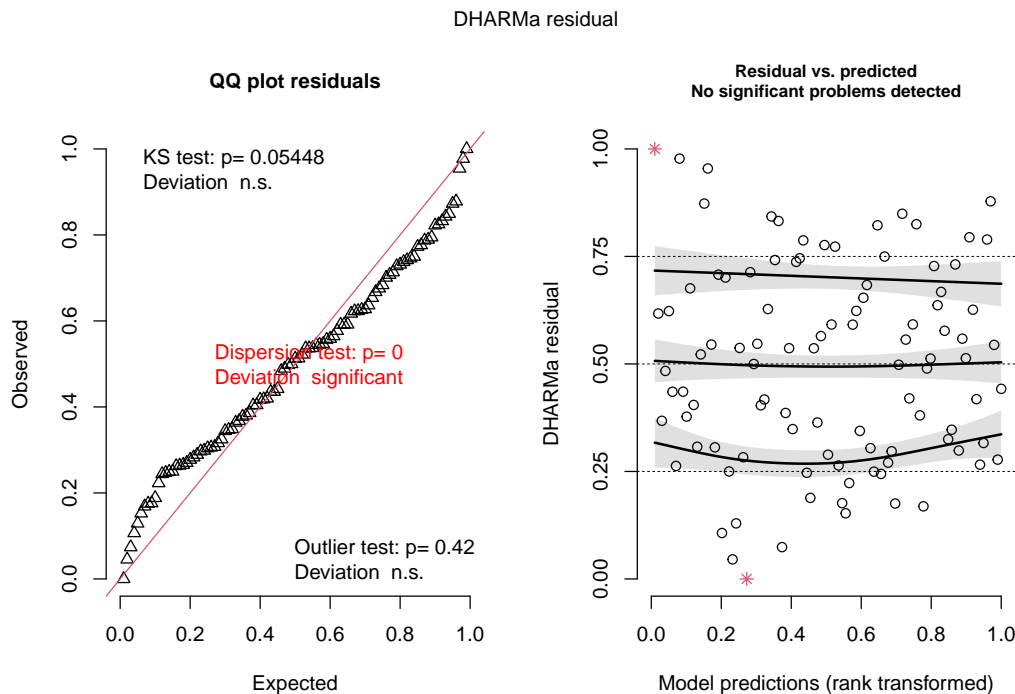
We can get look whether a model is over-dispersed by inspecting the model summary as you did in Section 3.4.6.1. As a rule of thumb, if the response variable conforms to a true Poisson distribution, we expect the residual deviance to be approximately equal to the residual degrees of freedom. If the deviance is much greater than the degrees of freedom, this indicates over-dispersion. This is the case in our models (see outputs from running the code in Section 3.4.6.1).

3 Empirical modeling

To check the model assumptions in a GLM is not as straight-forward as with a linear model. This is because classical residuals are not expected to behave in the same way for GLMs. We can use the DHARMA package in R for working with GLMs, which uses a simulation-based approach to compare the residuals from the actual model with the expectation if the model is behaving normally:

```
# Simulate residuals
simResids <- DHARMA::simulateResiduals(birdModel11)

# Generate plots to compare the model residuals to expectations
plot(simResids)
```



These plots show us that this model is not behaving as we would expect in terms of homogeneity of variance and distribution of residuals. A follow up to this would be to try alternatives to deal with over-dispersed count data in GLMs such as fit a quasi-Poisson GLM or a negative binomial GLM. Unfortunately we do not have time to continue in this exercise.

3.4.7 Question D - Group 2

- Is the number of bird species affected by forest management type and the forest structural diversity?

3.4.7.1 Fitting a Poisson GLM in R

Fitting a Poisson GLM in R is very similar to fitting an analysis of covariance (or linear model), except that now we need to use the `glm` function. To run a GLM, we need to provide one extra piece of information beyond that needed for a linear model: the family of model we want to use. In this case, we want a Poisson family.

We could start by only looking at how the forest structural diversity affects the number bird species that we have in a plot. You can do this by creating a GLM model which has `birdNumObs` as response variable and `GiniDBH` as explanatory variable. For that you will use the function `glm` in R and use the `family = poisson`. You can see how to create and see this model here:

```
birdModel1 <- glm(birdNumObs ~ GiniDBH,
                  family = poisson,
                  data = observations)

summary(birdModel1)
```

Call:

```
glm(formula = birdNumObs ~ GiniDBH, family = poisson, data = observations)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.63827	0.05636	46.809	<2e-16 ***
GiniDBH	0.42725	0.19030	2.245	0.0248 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 59.277 on 98 degrees of freedom
 Residual deviance: 54.280 on 97 degrees of freedom
 AIC: 511.57

Number of Fisher Scoring iterations: 4

3 Empirical modeling

Here you can see that the coefficient table produced by a GLM is very similar to a linear model. The intercept tells us the estimated value of the response variable when the continuous explanatory variables (here just Gini index) have a value of 0. We then also have coefficients describing the slope of the relationship with our continuous explanatory variables. We can see here that the bird numbers appears to show a positive relationship with Gini index, which means that increasing structural diversity in trees diameter sizes has a positive relationship with the number of bird species in the plot.

We are also interested in understanding the relationship of the number of observed bird species and forest management, we can now try to add this variable into the model.

```
birdModel2 <- glm(birdNumObs ~ forestManagementType,  
                  family = poisson,  
                  data = observations)  
  
summary(birdModel2)
```

Call:

```
glm(formula = birdNumObs ~ forestManagementType, family = poisson,  
     data = observations)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.83688	0.06052	46.873	<2e-16
forestManagementTypesimple clearcutting	-0.11744	0.06850	-1.714	0.0865
forestManagementTypeunmanaged	-0.06429	0.08338	-0.771	0.4407

```
(Intercept) ***  
forestManagementTypesimple clearcutting .  
forestManagementTypeunmanaged  
---
```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

```
Null deviance: 59.277 on 98 degrees of freedom  
Residual deviance: 56.205 on 96 degrees of freedom  
AIC: 515.49
```

Number of Fisher Scoring iterations: 4

The intercept here tells us the estimated value of the response variable when the reference groups in our grouping (categorical) variables (here for retention clear-cutting). We

3 Empirical modeling

then also have coefficients describing the slope of the relationship with our continuous explanatory variables, and coefficients giving the estimated difference in the response variable for non-reference groupings. We can see here that number of bird species appears to show a negative relationship with simple clearcutting, unmanaged and a positive relationship with retention clear-cutting.

The type simple clearcutting and unmanaged appear to be no significant, which only tell us about the pairwise differences between the levels. To test whether the categorical predictor, as a whole, is significant is equivalent to testing whether there is any heterogeneity in the means of the levels of the predictor. When there are no other predictors in the model, this is a classical ANOVA problem.

3.4.7.2 Explanatory Power of the model

When we ran linear models, we used the coefficient of determination, or R^2 to assess how much of the variability in our response variable is explained by a given model. R^2 is based on the sums of squares of our model, and so cannot be calculated for GLMs. Instead, we can calculate the the deviance explained by our model:

```
# Extract the null and residual deviance from the model
dev.null <- birdModel1$null.deviance
dev.resid <- birdModel1$deviance

# Calculate the deviance explained by the model
dev.explained <- (dev.null - dev.resid) / dev.null

# Round to 3 decimal places
dev.explained <- round(dev.explained, 3)

dev.explained
```

```
[1] 0.084
```

Variability in forest structure (Gini index) explain 15% of the variation in bird species richness in this study system. That is an ok explanatory power for a very simple model of a complex ecological system (many factors determine the species richness and we are attempting to explain everything with one variable).

3.4.7.3 Model Assumptions

For Poisson GLMs, there is one further assumption that we have not encountered before. If data follow a Poisson distribution, then the mean of the distribution is equal to the

3 Empirical modeling

variance. Accordingly, a Poisson distribution is represented by just one parameter λ , which describes both the mean and the variance of the distribution.

Count data in ecology are often *overdispersed*, where the variance is greater than the mean. This violates the assumption of a Poisson GLM, and means that any statistics that we calculate from the model may be unreliable.

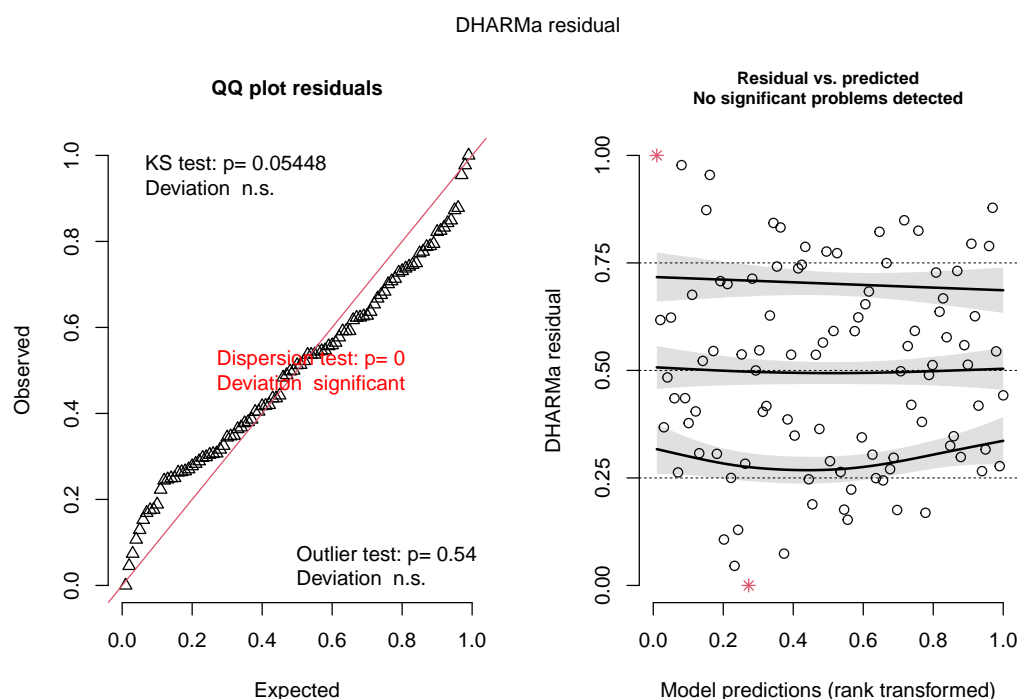
We can get look whether a model is over-dispersed by inspecting the model summary as you did in Section 3.4.7.1. As a rule of thumb, if the response variable conforms to a true Poisson distribution, we expect the residual deviance to be approximately equal to the residual degrees of freedom. If the deviance is much greater than the degrees of freedom, this indicates over-dispersion. This is the case in our models (see outputs from running the code in Section 3.4.7.1).

To check the model assumptions in a GLM is not as straight forward as with a linear model. This is because classical residuals are not expected to behave in the same way for GLMs. We can use the DHARMA package in R for working with GLMs, which uses a simulation-based approach to compare the residuals from the actual model with the expectation if the model is behaving normally.

```
# Simulate residuals
simResids <- DHARMA::simulateResiduals(birdModel1)

# Generate plots to compare the model residuals to expectations
plot(simResids)
```

3 Empirical modeling



These plots show us that this model is not behaving as we would expect in terms of homogeneity of variance and distribution of residuals. A follow up to this would be to try alternatives to deal with over-dispersed count data in GLMs such as fit a quasi-Poisson GLM or a negative binomial GLM. Unfortunately we do not have to continue in this exercise.

3.4.8 Question C - Group 3

- Is the presence of the Great spotted woodpecker affected by forest density?

3.4.8.1 Fitting a BRT in R

In this case we want to assess the occurrence of certain species across the plots. In other words, we want to assess what is the probability of a certain species with biodiversity interest to be present in a plot based on the variables that describe the forest of that plot. In this case the response variable is `dendrocoposMajor` that represents if the Great spotted woodpecker has been observed in this plot or not.

Then you need to select some variables of interest, after you have explored the data you can decide which variables you want to use to fit this model. We are proposing to select the following variables:

3 Empirical modeling

- `latitude` as proxy for plot location or/and climate
- `forestManagementType` to assess if different management types have different impact in the presence / absence of Great spotted woodpecker.
- `volAllha` that is the total volume in the plot, as a proxy of how dense the plot is. Higher volumes will mean that the forest is more dense.
- `GiniDBH` showing how homogeneous the plot is in trees diameters. A value closer to 1 indicate a more structural heterogeneity, lower values indicate more homogeneous plots.

You can create a vector `selVar` in which you add the names of the selected variables. Then you only take those variables from the data that you will use to create the model.

```
# Select variables from the dataset for the model
selVar <- c(
  "dendrocoposMajor",
  "latitude",
  "forestManagementType",
  "volAllha",
  "GiniDBH",
  "ShannonIndexTreeSpp"
)

# Filter the dataset to the selected variables
modelDataSel <- observations[, colnames(observations) %in% selVar]
```

Unfortunately the amount of that we have in this dataset it is not enough to fit a BRT model for these variables. We are going to do an obviously wrong thing for the shake of being able to demonstrate how to fit a BRT model. In the next code you are going to repeat the same dataset multiple times:

```
modelDataSel <- rbind(modelDataSel, modelDataSel, modelDataSel,
                      modelDataSel, modelDataSel)
```

Now it is important to assess if the variables have the right categories. Variables should be type numeric or factor.

```
summary(modelDataSel)
```

latitude	forestManagementType	volAllha	GiniDBH
Min. :48.65	Length:495	Min. : 1.681	Min. :0.08209
1st Qu.:49.31	Class :character	1st Qu.:319.920	1st Qu.:0.13684

3 Empirical modeling

```
Median :49.40   Mode  :character   Median :434.729   Median :0.20358
Mean   :49.49                                     Mean   :425.694   Mean   :0.25683
3rd Qu.:50.19                                     3rd Qu.:558.355   3rd Qu.:0.37368
Max.   :50.34                                     Max.   :777.882   Max.   :0.52852
ShannonIndexTreeSpp dendrocoposMajor
Min.   :0.000   Min.   :0.0000
1st Qu.:0.060   1st Qu.:0.0000
Median :0.280   Median :1.0000
Mean   :0.392   Mean   :0.7071
3rd Qu.:0.680   3rd Qu.:1.0000
Max.   :1.450   Max.   :1.0000
```

```
# Two variables are character, we assign to factor instead:
modelDataSel$forestManagementType <-
  as.factor(modelDataSel$forestManagementType)
```

In the next step you can see how you can run the model with the selected variables and model parameters. You have a description of the models parameters in the Section 3.3.2 . In this example we are going to use the default parameters for the calibration, where learning rate = 0.01 and tree complexity = 1 and cross-validation = 10-fold. However, the bag fraction is changed from the default value, 0.75, to 0.5. As a family we used the Bernoulli family, because we are predicting presence/absence per plot. These data have 495 plots, comprising 350 presence records for the Great spotted woodpecker. You can check these numbers by doing:

```
table(modelDataSel$dendrocoposMajor)
```

```
0    1
145 350
```

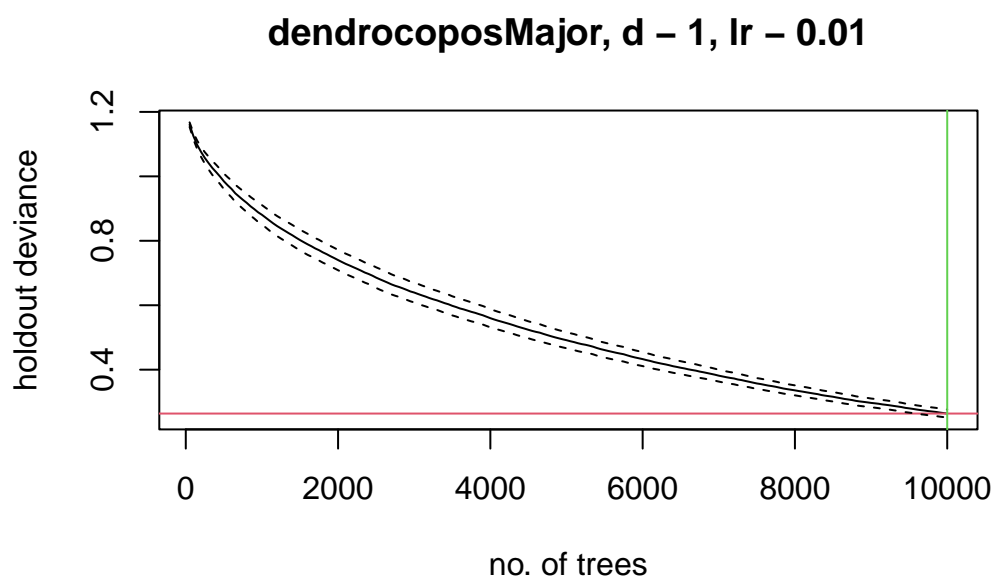
As a first guess you could decide there are enough data to model interactions of reasonable complexity, and a lr of about 0.01 could be a reasonable starting point. You can use the model creation function that steps forward and identifies the optimal number of trees (nt) by doing this:

```
family <- "bernoulli"
tc = 1    # tree complexity
lr = 0.01 # learning rate-shrinkage
bag = 0.5 # bag fraction

modelBRT <- dismo::gbm.step(
```

3 Empirical modeling

```
data = modelDataSel,  
#indices of predictor variables in data  
gbm.x = 1:5,  
#index of response variable in data:  
gbm.y = 6,  
family = family,  
tree.complexity = tc,  
learning.rate = lr,  
bag.fraction = bag  
)
```



Running a model such as that described above writes progress reports to the screen, makes a graph, and returns an object containing a number of components. The R console results reports a brief model summary all the values are also retained in the model object.

The model is built with the default 10-fold cross-validation (CV). In the plotted graph the solid black curve is the mean, and the dotted curves 1 standard error, for the changes in predictive deviance (i.e., as measured on the excluded folds of the CV). The red line shows the minimum of the mean, and the green line the number of trees at which that occurs. The final model that is returned in the model object is built on the full data set, using the number of trees identified as optimal.

3 Empirical modeling

Ideally, you should invest time in modifying the parameters and find the parameters that provide the models with the minimum deviance resulting from the best combination of bag, tree complexity and learning rate values. For the shake of limited timing, we will only test here the default values.

3.4.8.2 Model behaviour

You can summarize the model parameters used and the cross validation statistics from the fitted model by doing this:

```
# We make a table with the summary statistics
results <- data.frame(
  # Model parameters
  Tree.Complexity = modelBRT$gbm.call$tree.complexity,
  Learning.Rate = modelBRT$gbm.call$learning.rate,
  Bag.Fraction = modelBRT$gbm.call$bag.fraction,
  Interaction.depth = modelBRT$interaction.depth,
  Shrinkage = modelBRT$shrinkage,
  N.trees = modelBRT$n.trees,

  # Cross validation statistics

  ## mean total deviance
  Deviance = modelBRT$self.statistics$mean.resid,
  # mean residual deviance

  AUC = modelBRT$self.statistics$discrimination,
  # training data AUC score

  Corr = modelBRT$self.statistics$correlation,
  # training data correlation

  ## Cross Validation statistics

  # We calculate each statistic within each fold (at the identified optimal number
  # of trees that is calculated on the mean change in predictive deviance over all folds
  # then present here the mean and standard error of those fold-based statistics.

  devianceCV = modelBRT$cv.statistics$deviance.mean,
  # estimated cv deviance
  devianceCVse = modelBRT$cv.statistics$deviance.se,
  # estimated cv deviance se
```

3 Empirical modeling

```
CorrCV = modelBRT$cv.statistics$correlation.mean,  
#cv correlation  
CorrCVse = modelBRT$cv.statistics$correlation.se,  
#cv correlation se  
  
AUCcv = modelBRT$cv.statistics$discrimination.mean,  
# cv AUC score  
AUCcvSE = modelBRT$cv.statistics$discrimination.se  
) # cv AUC score se  
  
print(t(results))
```

```
          [,1]  
Tree.Complexity  1.000000e+00  
Learning.Rate    1.000000e-02  
Bag.Fraction     5.000000e-01  
Interaction.depth 1.000000e+00  
Shrinkage        1.000000e-02  
N.trees          1.000000e+04  
Deviance         2.171324e-01  
AUC              1.000000e+00  
Corr             9.750307e-01  
devianceCV       2.638418e-01  
devianceCVse     1.255729e-02  
CorrCV           9.578769e-01  
CorrCVse         4.999275e-03  
AUCcv            1.000000e+00  
AUCcvSE          0.000000e+00
```

3.4.8.3 Model output analysis

We can look at the relative contribution of each of the predictor variables. The measures are based on the number of times the variable is selected for splitting, weighted by the improvement of the model as a result of each split averaged across all trees. The relative contribution of each of the variables is scaled so the sum is 100%, with higher numbers indicating stronger influence in the response.

```
# Variables contribution  
modelBRT$contributions
```

3 Empirical modeling

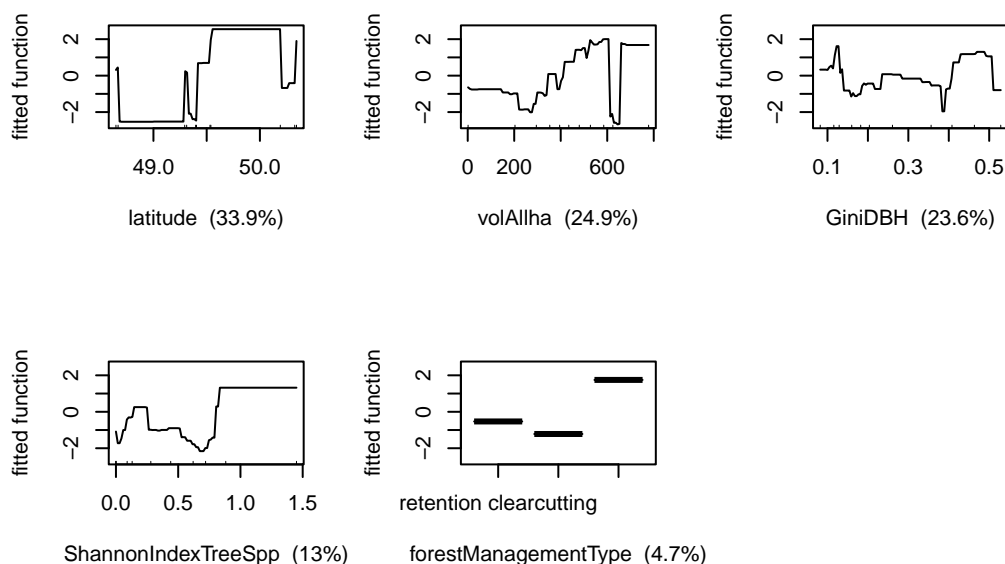
	var	rel.inf
latitude	latitude	33.869099
volAllha	volAllha	24.903641
GiniDBH	GiniDBH	23.570631
ShannonIndexTreeSpp	ShannonIndexTreeSpp	12.978776
forestManagementType	forestManagementType	4.677853

Here we can see that the two variables with the highest influence in the response are `latitude`, and `volAllhsa`.

Now we can evaluate the model behavior via partial dependence plots, showing the effect of each of the variables on the response by accounting for the average effects of all other predictors in the model:

```
dismo::gbm.plot(
  modelBRT,
  n.plots = 6,
  plot.layout = c(2, 3),
  write.title = F
)
```

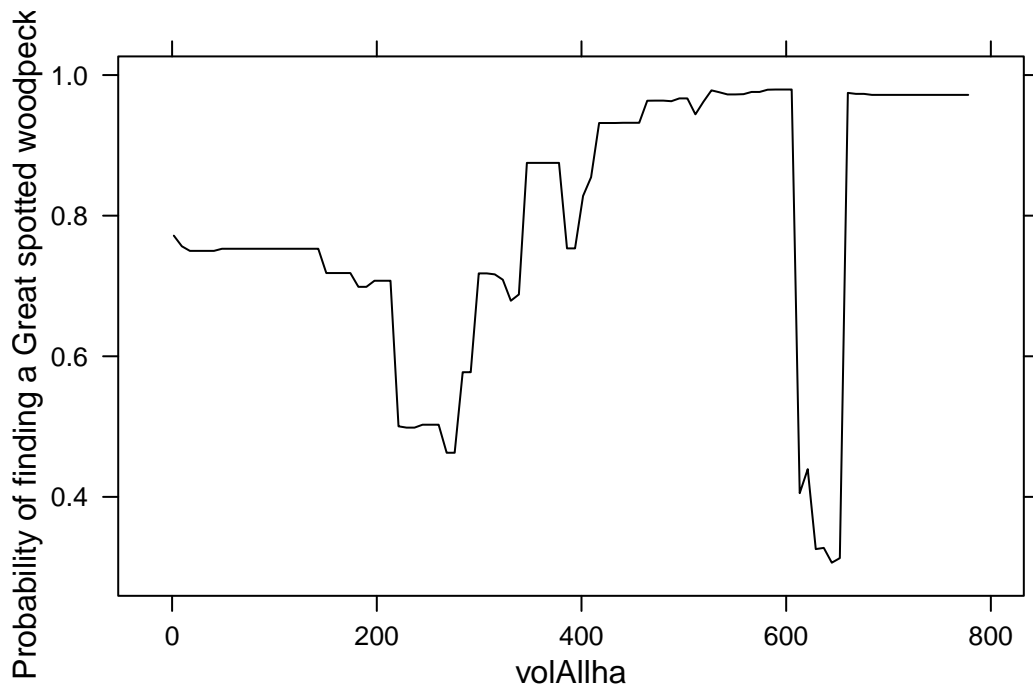
Warning in `dismo::gbm.plot(modelBRT, n.plots = 6, plot.layout = c(2, 3), :`
reducing no of plotted predictors to maximum available (5)



3 Empirical modeling

In partial dependence plots the predictions are on the scale of $f(x)$. In this case, for the Bernoulli loss the returned value is on the log odds scale. You can see how this plot will look by plotting with the function from the package `gbm` and using the type “response”. Since we are interested in finding out if the forest density has an impact on the presence of the Great spotted woodpecker we can have a look to the plot density variable `volAllha` :

```
gbm::plot.gbm(modelBRT, i.var = 3, type = "response",  
               ylab = "Probability of finding a Great spotted woodpecker ")
```

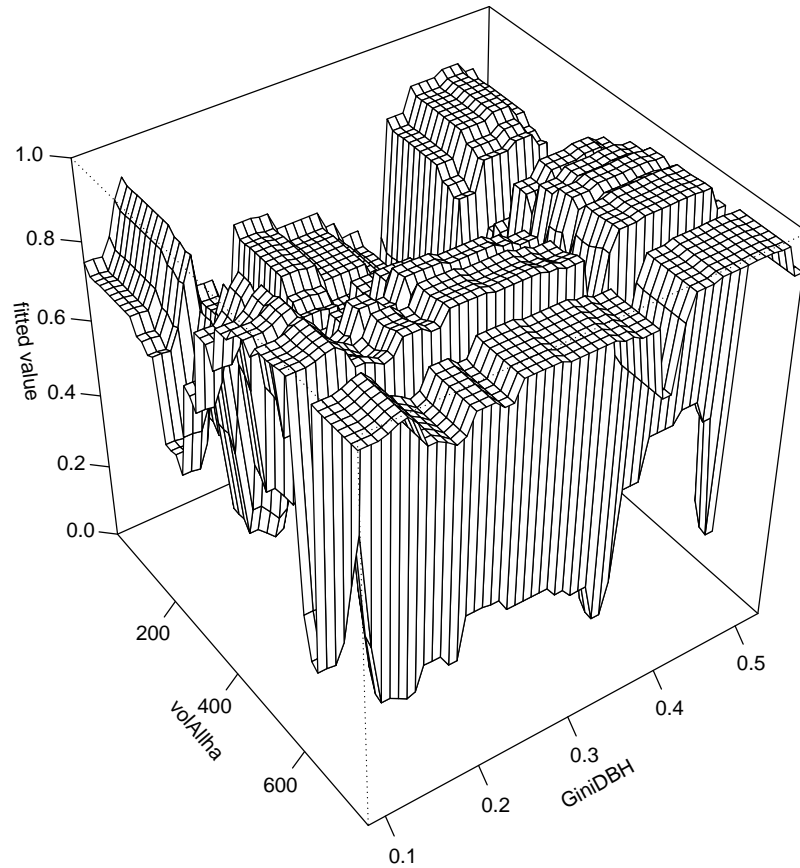


It seems that there is an increase in probability of finding a Great spotted woodpecker with higher forest densities, but the trend it is not very clear. We could also analyse the interaction effects, of density and for example overall bird richness. The model predictions can be obtained for each pair of predictor variables, setting all other predictors to their means.

To plot this pairwise interactions we have to do:

```
dismo::gbm.perspec(modelBRT, 3, 4)
```

3 Empirical modeling



Here we can see that both increasing bird diversity and forest density provides the highest probabilities for finding the Great spotted woodpecker.

3.4.9 Question D - Group 3

- Is the presence of the Great spotted woodpecker affected by forest diversity?

3.4.9.1 Fitting a BRT in R

In this case we want to assess the occurrence of certain species across the plots. In other words, we want to assess what is the probability of a certain species with biodiversity interest to be present in a plot based on the variables that describe the forest of that plot. In this case the response variable is `dendrocoposMajor` that represents if the Great spotted woodpecker has been observed in this plot or not.

Then you need to select some variables of interest, after you have explored the data you can decide which variables you want to use to fit this model. We are proposing to select the following variables:

- `latitude` as proxy for plot location or/and climate
- `forestManagementType` to assess if different management types have different impact in the presence / absence of Great spotted woodpecker.
- `volAllha` that is the total volume in the plot, as a proxy of how dense the plot is. Higher volumes will mean that the forest is more dense.
- `GiniDBH` showing how homogeneous the plot is in trees diameters. A value closer to 1 will mean that indicate more structural heterogeneity, lower values indicate more homogeneous plots.

You can create a vector `selVar` in which you add the names of the selected variables. Then you only take those variables from the data that you will use to create the model.

```
# Select variables from the dataset for the model
selVar <- c(
  "dendrocoposMajor",
  "latitude",
  "forestManagementType",
  "volAllha",
  "GiniDBH",
  "ShannonIndexTreeSpp"
)

# Filter the dataset to the selected variables
modelDataSel <- observations[, colnames(observations) %in% selVar]
```

Unfortunately the amount of that we have in this dataset it is not enough to fit a BRT model for these variables. We are going to do an obviously wrong thing for the shake of being able to demonstrate how to fit a BRT model. In the next code you are going to repeat the same dataset multiple times:

3 Empirical modeling

```
modelDataSel <- rbind(modelDataSel, modelDataSel, modelDataSel, modelDataSel,
                      modelDataSel)
```

Now it is important to assess if the variables have the right categories. Variables should be type numeric or factor.

```
summary(modelDataSel)
```

latitude	forestManagementType	volAllha	GiniDBH
Min. :48.65	Length:495	Min. : 1.681	Min. :0.08209
1st Qu.:49.31	Class :character	1st Qu.:319.920	1st Qu.:0.13684
Median :49.40	Mode :character	Median :434.729	Median :0.20358
Mean :49.49		Mean :425.694	Mean :0.25683
3rd Qu.:50.19		3rd Qu.:558.355	3rd Qu.:0.37368
Max. :50.34		Max. :777.882	Max. :0.52852
ShannonIndexTreeSpp dendrocoposMajor			
Min. :0.000	Min. :0.0000		
1st Qu.:0.060	1st Qu.:0.0000		
Median :0.280	Median :1.0000		
Mean :0.392	Mean :0.7071		
3rd Qu.:0.680	3rd Qu.:1.0000		
Max. :1.450	Max. :1.0000		

```
# Two variables are character, we assign to factor instead:
modelDataSel$forestManagementType <-
  as.factor(modelDataSel$forestManagementType)
```

In the next step you can see how you can run the model with the selected variables and model parameters. You have a description of the models parameters in the Section 3.3.2 . In this example we are going to use the default parameters for the calibration, where learning rate = 0.01 and tree complexity = 1 and cross-validation = 10-fold. However, the bag fraction is changed from the default value, 0.75, to 0.5. As a family we used the Bernoulli family, because we are predicting presence/absence per plot. These data have 495 plots, comprising 350 presence records for the Great spotted woodpecker. You can check these numbers by doing:

```
table(modelDataSel$dendrocoposMajor)
```

```
0 1
145 350
```

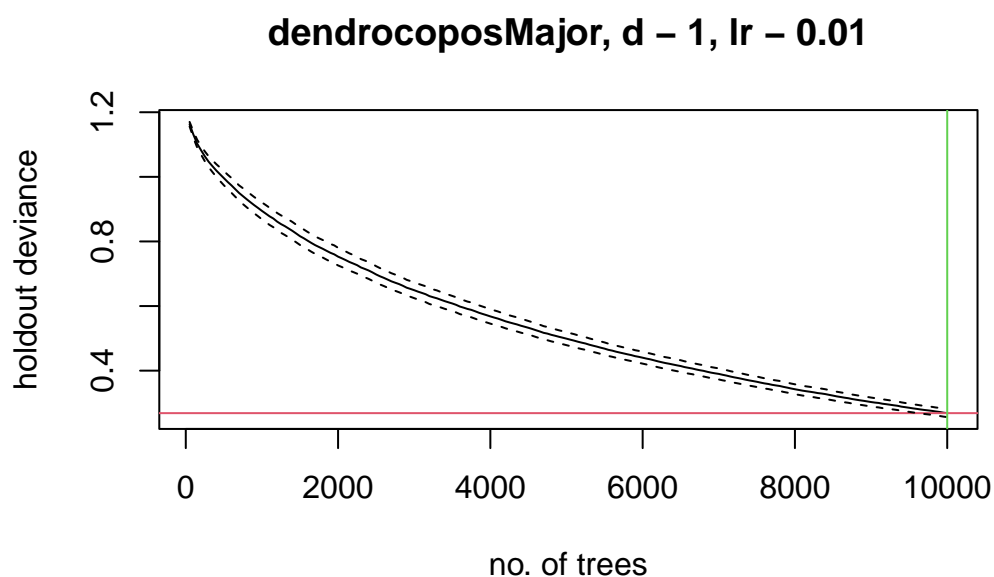
3 Empirical modeling

As a first guess you could decide there are enough data to model interactions of reasonable complexity, and a lr of about 0.01 could be a reasonable starting point. You can use the model creation function that steps forward and identifies the optimal number of trees (nt) by doing this:

```
family <- "bernoulli"

tc = 1    # tree complexity
lr = 0.01 # learning rate-shrinkage
bag = 0.5 # bag fraction

modelBRT <- dismo::gbm.step(
  data = modelDataSel,
  #indices of predictor variables in data
  gbm.x = 1:5,
  #index of response variable in data:
  gbm.y = 6,
  family = family,
  tree.complexity = tc,
  learning.rate = lr,
  bag.fraction = bag
)
```



3 Empirical modeling

Running a model such as that described above writes progress reports to the screen, makes a graph, and returns an object containing a number of components. The R console results reports a brief model summary all the values are also retained in the model object.

The model is built with the default 10-fold cross-validation (CV). In the plotted graph the solid black curve is the mean, and the dotted curves 1 standard error, for the changes in predictive deviance (i.e., as measured on the excluded folds of the CV). The red line shows the minimum of the mean, and the green line the number of trees at which that occurs. The final model that is returned in the model object is built on the full data set, using the number of trees identified as optimal.

Ideally, you should invest time in modifying the parameters and find the parameters that provide the models with the minimum deviance resulting from the best combination of bag, tree complexity and learning rate values. For the shake of limited timing, we will only test here the default values.

3.4.9.2 Model behaviour

You can summarize the model parameters used and the cross validation statistics from the fitted model by doing this:

```
# We make a table with the summary statistics
results <- data.frame(
  # Model parameters
  Tree.Complexity = modelBRT$gbm.call$tree.complexity,
  Learning.Rate = modelBRT$gbm.call$learning.rate,
  Bag.Fraction = modelBRT$gbm.call$bag.fraction,
  Interaction.depth = modelBRT$interaction.depth,
  Shrinkage = modelBRT$shrinkage,
  N.trees = modelBRT$n.trees,

  # Cross validation statistics

  ## mean total deviance
  Deviance = modelBRT$self.statistics$mean.resid,
  # mean residual deviance

  AUC = modelBRT$self.statistics$discrimination,
  # training data AUC score

  Corr = modelBRT$self.statistics$correlation,
  # training data correlation
```

3 Empirical modeling

```
## Cross Validation statistics

# We calculate each statistic within each fold (at the identified optimal number
# of trees that is calculated on the mean change in predictive deviance over all folds)
# then present here the mean and standard error of those fold-based statistics.

devianceCV = modelBRT$cv.statistics$deviance.mean,
# estimated cv deviance
devianceCVse = modelBRT$cv.statistics$deviance.se,
# estimated cv deviance se

CorrCV = modelBRT$cv.statistics$correlation.mean,
# cv correlation
CorrCVse = modelBRT$cv.statistics$correlation.se,
# cv correlation se

AUCcv = modelBRT$cv.statistics$discrimination.mean,
# cv AUC score
AUCcvSE = modelBRT$cv.statistics$discrimination.se
) # cv AUC score se

print(t(results))
```

```
          [,1]
Tree.Complexity  1.000000e+00
Learning.Rate    1.000000e-02
Bag.Fraction     5.000000e-01
Interaction.depth 1.000000e+00
Shrinkage        1.000000e-02
N.trees          1.000000e+04
Deviance         2.172949e-01
AUC              1.000000e+00
Corr             9.748868e-01
devianceCV       2.684618e-01
devianceCVse     1.232031e-02
CorrCV           9.549340e-01
CorrCVse         6.449939e-03
AUCcv            9.988600e-01
AUCcvSE          1.140000e-03
```

3.4.9.3 Model output analysis

We can look at the relative contribution of each of the predictor variables. The measures are based on the number of times the variable is selected for splitting, weighted by the improvement of the model as a result of each split averaged across all trees. The relative contribution of each of the variables is scaled so the sum is 100%, with higher numbers indicating stronger influence in the response.

```
# Variables contribution
modelBRT$contributions
```

	var	rel.inf
latitude	latitude	33.522861
volAllha	volAllha	24.941379
GiniDBH	GiniDBH	24.143051
ShannonIndexTreeSpp	ShannonIndexTreeSpp	12.828783
forestManagementType	forestManagementType	4.563926

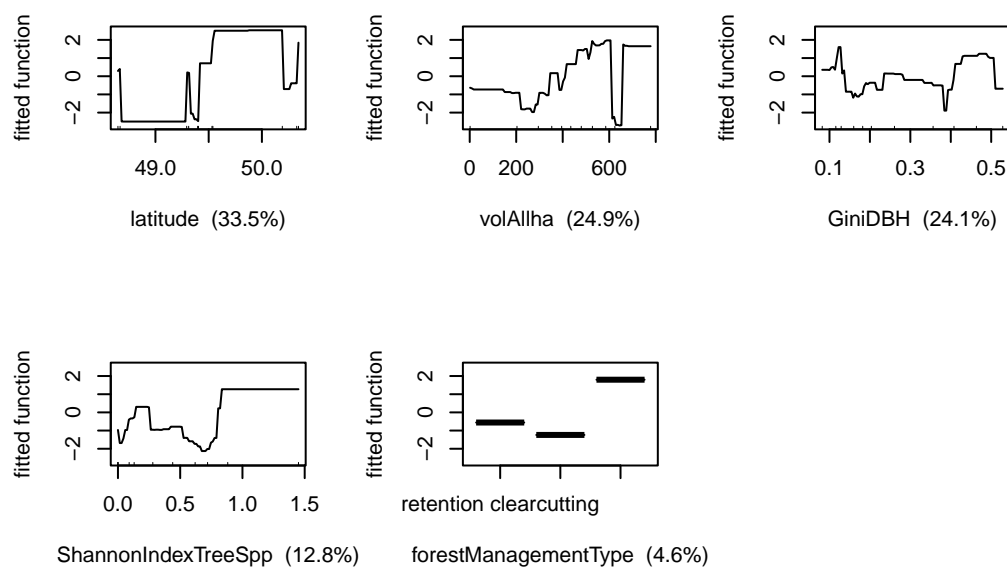
Here we can see that the two variables with the highest influence in the response are `latitude`, and `volAllha`.

Now we can evaluate the model behavior via partial dependence plots, showing the effect of each of the variables on the response by accounting for the average effects of all other predictors in the model:

```
dismo::gbm.plot(
  modelBRT,
  n.plots = 6,
  plot.layout = c(2, 3),
  write.title = F
)
```

```
Warning in dismo::gbm.plot(modelBRT, n.plots = 6, plot.layout = c(2, 3), :
reducing no of plotted predictors to maximum available (5)
```

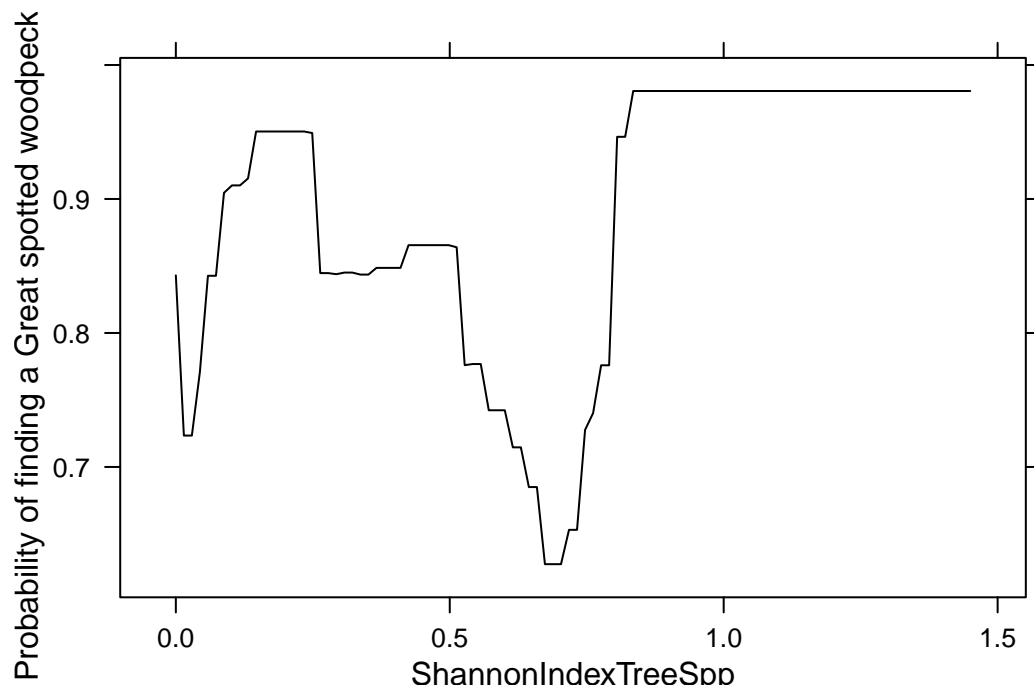
3 Empirical modeling



In this partial dependence plots the predictions are on the scale of $f(x)$. In this case, for the Bernoulli loss the returned value is on the log odds scale. You can see how this plot will look by plotting with the function from the package **gbm** and using the type “response”. Since we are interesting in finding out if the forest diversity has an impact in the presence of the Great spotted woodpecker we can have a look to the plot density variable `volAllha` :

```
gbm::plot.gbm(modelBRT,
  i.var = 5,
  type = "response",
  ylab = "Probability of finding a Great spotted woodpecker ")
```

3 Empirical modeling

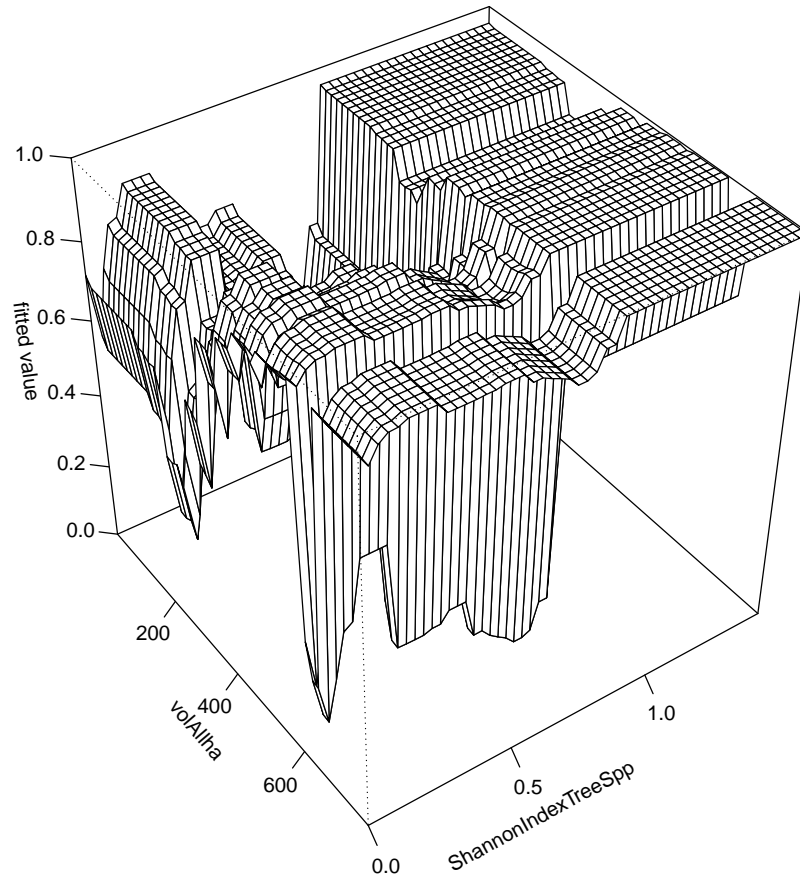


It seems that there is an increase in probability of finding a Great spotted woodpecker with higher forest densities, but the trend it is not very clear. We could also analyse the interaction effects, of forest diversity and for example forest density. The model predictions can be obtained for each pair of predictor variables, setting all other predictors to their means.

To plot this pairwise interactions we have to do:

```
dismo::gbm.perspec(modelBRT, 3, 5)
```


3 Empirical modeling



Here we can not see a very clear combined behavior between forest diversity and forest structural diversity in respect to the probabilities for finding the Great spotted woodpecker.

3.4.10 Question C - Group 4

- Is the presence of the Eurasian treecreeper affected by forest density?

3.4.10.1 Fitting a BRT in R

In this case we want to assess the occurrence of certain species across the plots. In other words, we want to assess what is the probability of a certain species with biodiversity interest to be present in a plot based on the variables that describe the forest of that plot. In this case the response variable is `certhia` that represents if the Eurasian treecreeper has been observed in this plot or not.

Then you need to select some variables of interest, after you have explored the data you can decide which variables you want to use to fit this model. We are proposing to select the following variables:

- `latitude` as proxy for plot location or/and climate
- `forestManagementType` to assess if different management types have different impact in the presence / absence of Great spotted woodpecker.
- `volAllha` that is the total volume in the plot, as a proxy of how dense the plot is. Higher volumes will mean that the forest is more dense.
- `GiniDBH` showing how homogeneous the plot is in trees diameters. A value closer to 1 will mean that indicate more structural heterogeneity, lower values indicate more homogeneous plots.

You can create a vector `selVar` in which you add the names of the selected variables. Then you only take those variables from the data that you will use to create the model.

```
# Select variables from the dataset for the model
selVar <- c(
  "certhia",
  "latitude",
  "forestManagementType",
  "volAllha",
  "GiniDBH",
  "ShannonIndexTreeSpp"
)

# Filter the dataset to the selected variables
modelDataSel <- observations[, colnames(observations) %in% selVar]
```

Unfortunately the amount of that we have in this dataset it is not enough to fit a BRT model for these variables. We are going to do an obviously wrong thing for the shake of being able to demonstrate how to fit a BRT model. In the next code you are going to repeat the same dataset multiple times:

3 Empirical modeling

```
modelDataSel <- rbind(modelDataSel, modelDataSel, modelDataSel, modelDataSel,  
                      modelDataSel)
```

Now it is important to assess if the variables have the right categories. Variables should be type numeric or factor.

```
summary(modelDataSel)
```

latitude	forestManagementType	volAllha	GiniDBH
Min. :48.65	Length:495	Min. : 1.681	Min. :0.08209
1st Qu.:49.31	Class :character	1st Qu.:319.920	1st Qu.:0.13684
Median :49.40	Mode :character	Median :434.729	Median :0.20358
Mean :49.49		Mean :425.694	Mean :0.25683
3rd Qu.:50.19		3rd Qu.:558.355	3rd Qu.:0.37368
Max. :50.34		Max. :777.882	Max. :0.52852
ShannonIndexTreeSpp	certhia		
Min. :0.000	Min. :0.0000		
1st Qu.:0.060	1st Qu.:0.0000		
Median :0.280	Median :1.0000		
Mean :0.392	Mean :0.5859		
3rd Qu.:0.680	3rd Qu.:1.0000		
Max. :1.450	Max. :1.0000		

```
# Two variables are character, we assign to factor instead:  
modelDataSel$forestManagementType <-  
  as.factor(modelDataSel$forestManagementType)
```

In the next step you can see how you can run the model with the selected variables and model parameters. You have a description of the models parameters in the Section 3.3.2 . In this example we are going to use the default parameters for the calibration, where learning rate = 0.01 and tree complexity = 1 and cross-validation = 10-fold. However, the bag fraction is changed from the default value, 0.75, to 0.5. As a family we used the Bernoulli family, because we are predicting presence/absence per plot. These data have 495 plots, comprising 290 presence records for the Eurasian treecreeper. You can check these numbers by doing:

```
table(modelDataSel$certhia)
```

```
0 1  
205 290
```

3 Empirical modeling

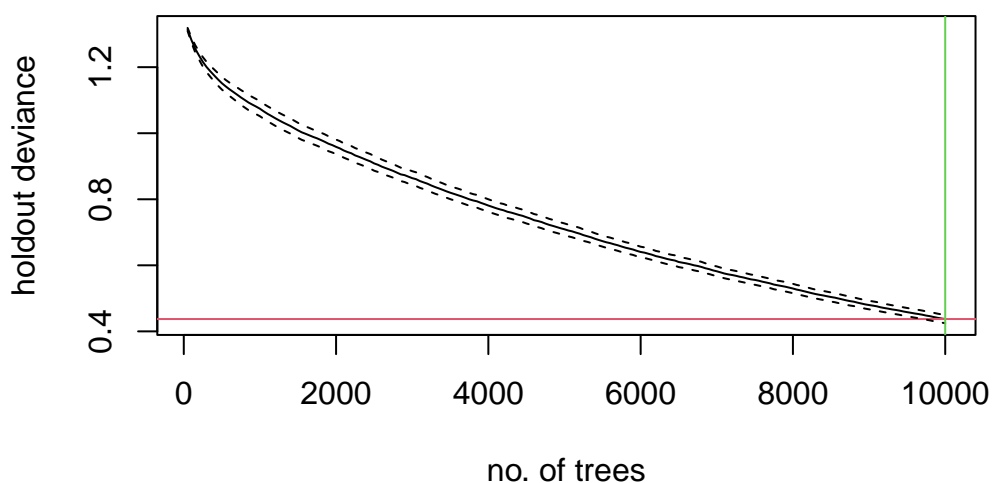
As a first guess you could decide there are enough data to model interactions of reasonable complexity, and a lr of about 0.01 could be a reasonable starting point. You can use the model creation function that steps forward and identifies the optimal number of trees (nt) by doing this:

```
family <- "bernoulli"

tc = 1    # tree complexity
lr = 0.01 # learning rate-shrinkage
bag = 0.5 # bag fraction

modelBRT <- dismo::gbm.step(
  data = modelDataSel,
  #indices of predictor variables in data
  gbm.x = 1:5,
  #index of response variable in data:
  gbm.y = 6,
  family = family,
  tree.complexity = tc,
  learning.rate = lr,
  bag.fraction = bag
)
```

certhia, d – 1, lr – 0.01



3 Empirical modeling

Running a model such as that described above writes progress reports to the screen, makes a graph, and returns an object containing a number of components. The R console results reports a brief model summary all the values are also retained in the model object.

The model is built with the default 10-fold cross-validation (CV). In the plotted graph the solid black curve is the mean, and the dotted curves 1 standard error, for the changes in predictive deviance (i.e., as measured on the excluded folds of the CV). The red line shows the minimum of the mean, and the green line the number of trees at which that occurs. The final model that is returned in the model object is built on the full data set, using the number of trees identified as optimal.

Ideally, you should invest time in modifying the parameters and find the parameters that provide the models with the minimum deviance resulting from the best combination of bag, tree complexity and learning rate values. For the shake of limited timing, we will only test here the default values.

3.4.10.2 Model behaviour

You can summarize the model parameters used and the cross validation statistics from the fitted model by doing this:

```
# We make a table with the summary statistics
results <- data.frame(
  # Model parameters
  Tree.Complexity = modelBRT$gbm.call$tree.complexity,
  Learning.Rate = modelBRT$gbm.call$learning.rate,
  Bag.Fraction = modelBRT$gbm.call$bag.fraction,
  Interaction.depth = modelBRT$interaction.depth,
  Shrinkage = modelBRT$shrinkage,
  N.trees = modelBRT$n.trees,

  # Cross validation statistics

  ## mean total deviance
  Deviance = modelBRT$self.statistics$mean.resid,
  # mean residual deviance

  AUC = modelBRT$self.statistics$discrimination,
  # training data AUC score

  Corr = modelBRT$self.statistics$correlation,
  # training data correlation
```

3 Empirical modeling

```
## Cross Validation statistics

# We calculate each statistic within each fold (at the identified optimal number
# of trees that is calculated on the mean change in predictive deviance over all folds)
# then present here the mean and standard error of those fold-based statistics.

devianceCV = modelBRT$cv.statistics$deviance.mean,
# estimated cv deviance
devianceCVse = modelBRT$cv.statistics$deviance.se,
# estimated cv deviance se

CorrCV = modelBRT$cv.statistics$correlation.mean,
# cv correlation
CorrCVse = modelBRT$cv.statistics$correlation.se,
# cv correlation se

AUCcv = modelBRT$cv.statistics$discrimination.mean,
# cv AUC score
AUCcvSE = modelBRT$cv.statistics$discrimination.se
) # cv AUC score se

print(t(results))
```

```
          [,1]
Tree.Complexity  1.000000e+00
Learning.Rate    1.000000e-02
Bag.Fraction     5.000000e-01
Interaction.depth 1.000000e+00
Shrinkage        1.000000e-02
N.trees          1.000000e+04
Deviance         3.634722e-01
AUC              1.000000e+00
Corr             9.538741e-01
devianceCV       4.372956e-01
devianceCVse     1.238398e-02
CorrCV           9.235549e-01
CorrCVse         5.046297e-03
AUCcv            9.990100e-01
AUCcvSE          7.043437e-04
```

3.4.10.3 Model output analysis

We can look at the relative contribution of each of the predictor variables. The measures are based on the number of times the variable is selected for splitting, weighted by the improvement of the model as a result of each split averaged across all trees. The relative contribution of each of the variables is scaled so the sum is 100%, with higher numbers indicating stronger influence in the response.

```
# Variables contribution
modelBRT$contributions
```

	var	rel.inf
GiniDBH	GiniDBH	30.738508
latitude	latitude	25.141662
volAllha	volAllha	23.659849
ShannonIndexTreeSpp	ShannonIndexTreeSpp	19.152892
forestManagementType	forestManagementType	1.307088

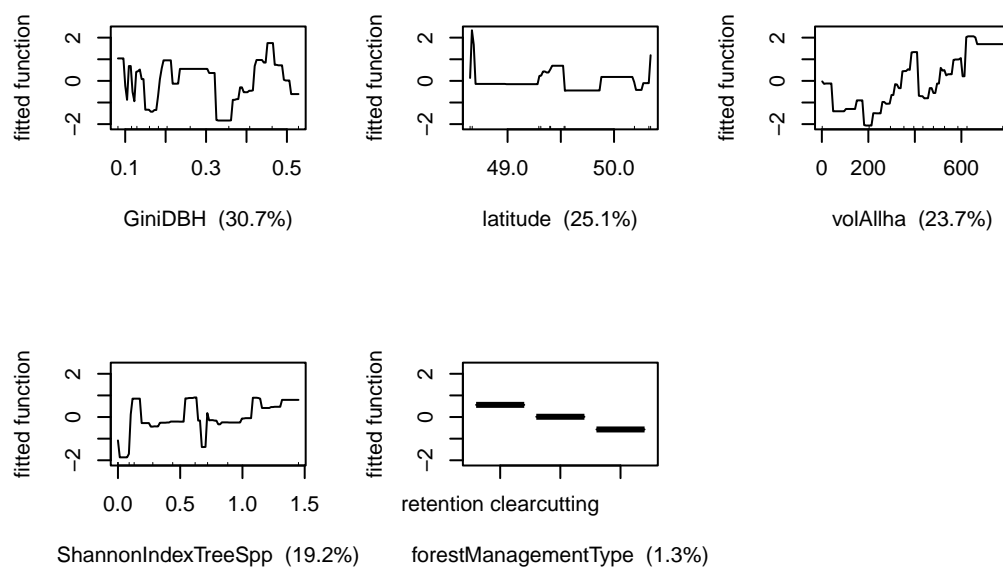
Here we can see that the two variables with the highest influence in the response are GiniDBH and volAllha.

Now we can evaluate the model behavior via partial dependence plots, showing the effect of each of the variables on the response by accounting for the average effects of all other predictors in the model:

```
dismo::gbm.plot(
  modelBRT,
  n.plots = 6,
  plot.layout = c(2, 3),
  write.title = F
)
```

Warning in dismo::gbm.plot(modelBRT, n.plots = 6, plot.layout = c(2, 3), :
reducing no of plotted predictors to maximum available (5)

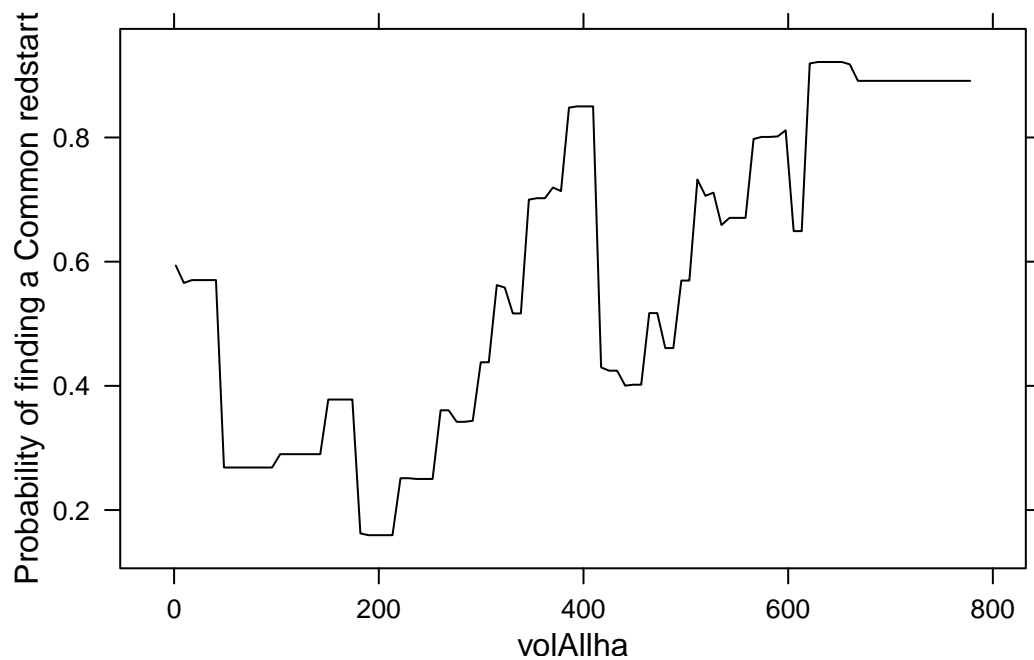
3 Empirical modeling



In this partial dependence plots the predictions are on the scale of $f(x)$. In this case, for the Bernoulli loss the returned value is on the log odds scale. You can see how this plot will look by plotting with the function from the package **gbm** and using the type “response”. Since we are interesting in finding out if the forest density has an impact in the presence of the Eurasian treecreeper affected we can have a look to the plot density variable `volAllha` :

```
gbm::plot.gbm(modelBRT,  
  i.var = 3,  
  type = "response",  
  ylab = "Probability of finding a Common redstart")
```


3 Empirical modeling

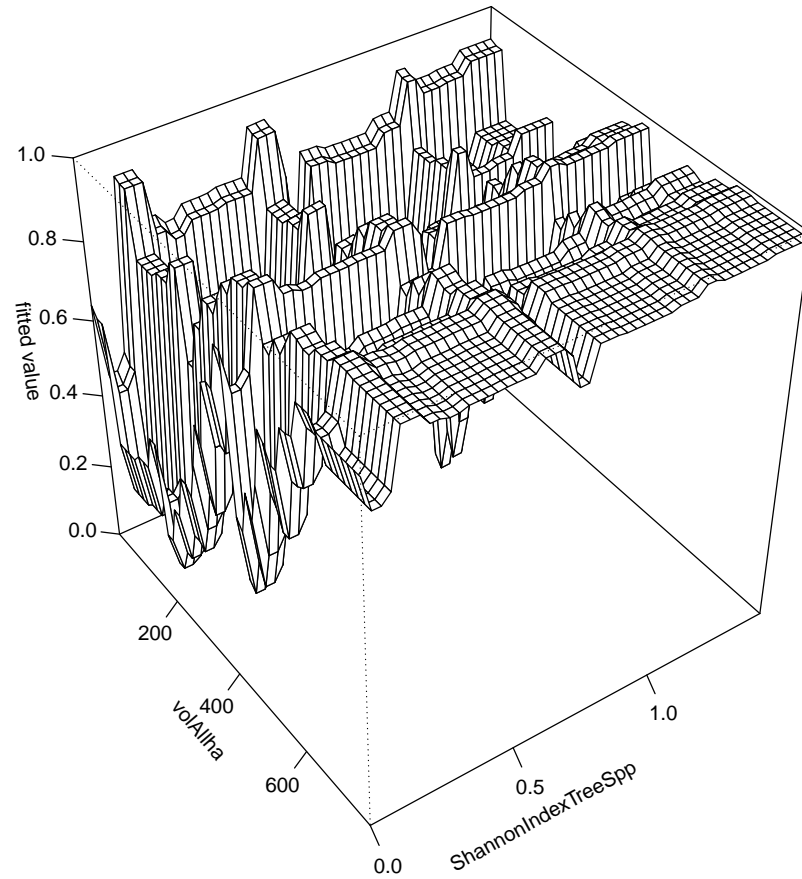


It seems that there is an increase in probability of finding a Eurasian treecreeper with higher forest densities. We could also analyse the interaction effects, of forest diversity and for example forest density. The model predictions can be obtained for each pair of predictor variables, setting all other predictors to their means.

To plot this pairwise interactions we have to do:

```
dismo::gbm.perspec(modelBRT, 3, 5)
```

3 Empirical modeling



It seems that the highest chances to see a Eurasian treecreeper is in dense forests with highest tree diversity.

3.4.11 Question D - Group 4

- Is the presence of the Eurasian treecreeper affected by forest management?

3.4.11.1 Fitting a BRT in R

In this case we want to assess the occurrence of certain species across the plots. In other words, we want to assess what is the probability of a certain species with biodiversity interest to be present in a plot based on the variables that describe the forest of that plot. In this case the response variable is `certhia` that represents if the Eurasian treecreeper has been observed in this plot or not.

Then you need to select some variables of interest, after you have explored the data you can decide which variables you want to use to fit this model. We are proposing to select the following variables:

- `latitude` as proxy for plot location or/and climate
- `forestManagementType` to assess if different management types have different impact in the presence / absence of Great spotted woodpecker.
- `volAllha` that is the total volume in the plot, as a proxy of how dense the plot is. Higher volumes will mean that the forest is more dense.
- `GiniDBH` showing how homogeneous the plot is in trees diameters. A value closer to 1 will mean that indicate more structural heterogeneity, lower values indicate more homogeneous plots.

You can create a vector `selVar` in which you add the names of the selected variables. Then you only take those variables from the data that you will use to create the model.

```
# Select variables from the dataset for the model
selVar <- c(
  "certhia",
  "latitude",
  "forestManagementType",
  "volAllha",
  "GiniDBH",
  "ShannonIndexTreeSpp"
)

# Filter the dataset to the selected variables
modelDataSel <- observations[, colnames(observations) %in% selVar]
```

Unfortunately the amount of that we have in this dataset it is not enough to fit a BRT model for these variables. We are going to do an obviously wrong thing for the shake of being able to demonstrate how to fit a BRT model. In the next code you are going to repeat the same dataset multiple times:

3 Empirical modeling

```
modelDataSel <- rbind(modelDataSel, modelDataSel, modelDataSel, modelDataSel,  
                      modelDataSel)
```

Now it is important to assess if the variables have the right categories. Variables should be type numeric or factor.

```
summary(modelDataSel)
```

latitude	forestManagementType	volAllha	GiniDBH
Min. :48.65	Length:495	Min. : 1.681	Min. :0.08209
1st Qu.:49.31	Class :character	1st Qu.:319.920	1st Qu.:0.13684
Median :49.40	Mode :character	Median :434.729	Median :0.20358
Mean :49.49		Mean :425.694	Mean :0.25683
3rd Qu.:50.19		3rd Qu.:558.355	3rd Qu.:0.37368
Max. :50.34		Max. :777.882	Max. :0.52852
ShannonIndexTreeSpp	certhia		
Min. :0.000	Min. :0.0000		
1st Qu.:0.060	1st Qu.:0.0000		
Median :0.280	Median :1.0000		
Mean :0.392	Mean :0.5859		
3rd Qu.:0.680	3rd Qu.:1.0000		
Max. :1.450	Max. :1.0000		

```
# Two variables are character, we assign to factor instead:  
modelDataSel$forestManagementType <-  
  as.factor(modelDataSel$forestManagementType)
```

In the next step you can see how you can run the model with the selected variables and model parameters. You have a description of the models parameters in the Section 3.3.2 . In this example we are going to use the default parameters for the calibration, where learning rate = 0.01 and tree complexity = 1 and cross-validation = 10-fold. However, the bag fraction is changed from the default value, 0.75, to 0.5. As a family we used the Bernoulli family, because we are predicting presence/absence per plot. These data have 495 plots, comprising 290 presence records for the Eurasian treecreeper. You can check these numbers by doing:

```
table(modelDataSel$certhia)
```

```
0 1  
205 290
```

3 Empirical modeling

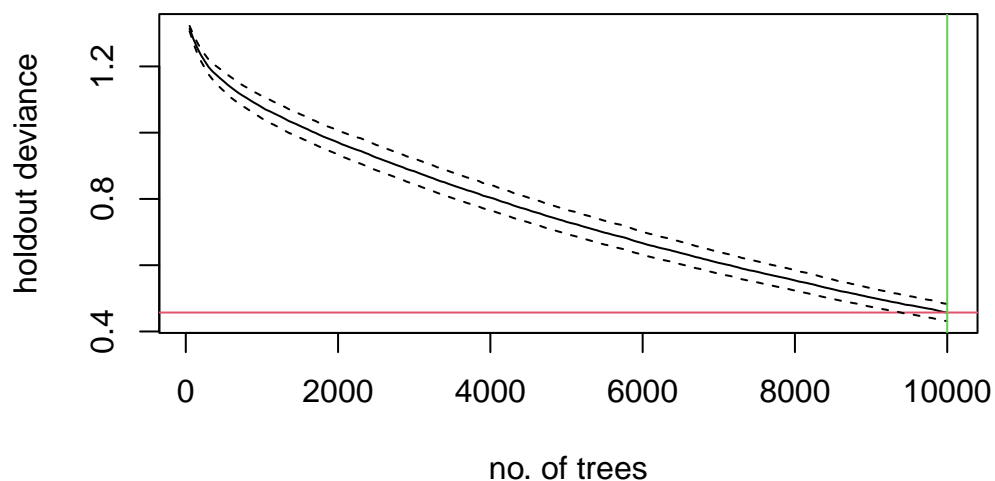
As a first guess you could decide there are enough data to model interactions of reasonable complexity, and a lr of about 0.01 could be a reasonable starting point. You can use the model creation function that steps forward and identifies the optimal number of trees (nt) by doing this:

```
family <- "bernoulli"

tc = 1    # tree complexity
lr = 0.01 # learning rate-shrinkage
bag = 0.5 # bag fraction

modelBRT <- dismo::gbm.step(
  data = modelDataSel,
  #indices of predictor variables in data
  gbm.x = 1:5,
  #index of response variable in data:
  gbm.y = 6,
  family = family,
  tree.complexity = tc,
  learning.rate = lr,
  bag.fraction = bag
)
```

certhia, d – 1, lr – 0.01



3 Empirical modeling

Running a model such as that described above writes progress reports to the screen, makes a graph, and returns an object containing a number of components. The R console results reports a brief model summary all the values are also retained in the model object.

The model is built with the default 10-fold cross-validation (CV). In the plotted graph the solid black curve is the mean, and the dotted curves 1 standard error, for the changes in predictive deviance (i.e., as measured on the excluded folds of the CV). The red line shows the minimum of the mean, and the green line the number of trees at which that occurs. The final model that is returned in the model object is built on the full data set, using the number of trees identified as optimal.

Ideally, you should invest time in modifying the parameters and find the parameters that provide the models with the minimum deviance resulting from the best combination of bag, tree complexity and learning rate values. For the shake of limited timing, we will only test here the default values.

3.4.11.2 Model behaviour

You can summarize the model parameters used and the cross validation statistics from the fitted model by doing this:

```
# We make a table with the summary statistics
results <- data.frame(
  # Model parameters
  Tree.Complexity = modelBRT$gbm.call$tree.complexity,
  Learning.Rate = modelBRT$gbm.call$learning.rate,
  Bag.Fraction = modelBRT$gbm.call$bag.fraction,
  Interaction.depth = modelBRT$interaction.depth,
  Shrinkage = modelBRT$shrinkage,
  N.trees = modelBRT$n.trees,

  # Cross validation statistics

  ## mean total deviance
  Deviance = modelBRT$self.statistics$mean.resid,
  # mean residual deviance

  AUC = modelBRT$self.statistics$discrimination,
  # training data AUC score

  Corr = modelBRT$self.statistics$correlation,
  # training data correlation
```

3 Empirical modeling

```
## Cross Validation statistics

# We calculate each statistic within each fold (at the identified optimal number
# of trees that is calculated on the mean change in predictive deviance over all folds)
# then present here the mean and standard error of those fold-based statistics.

devianceCV = modelBRT$cv.statistics$deviance.mean,
# estimated cv deviance
devianceCVse = modelBRT$cv.statistics$deviance.se,
# estimated cv deviance se

CorrCV = modelBRT$cv.statistics$correlation.mean,
# cv correlation
CorrCVse = modelBRT$cv.statistics$correlation.se,
# cv correlation se

AUCcv = modelBRT$cv.statistics$discrimination.mean,
# cv AUC score
AUCcvSE = modelBRT$cv.statistics$discrimination.se
) # cv AUC score se

print(t(results))
```

```
          [,1]
Tree.Complexity  1.000000e+00
Learning.Rate    1.000000e-02
Bag.Fraction     5.000000e-01
Interaction.depth 1.000000e+00
Shrinkage        1.000000e-02
N.trees          1.000000e+04
Deviance         3.667425e-01
AUC              1.000000e+00
Corr             9.527998e-01
devianceCV       4.571432e-01
devianceCVse     2.593891e-02
CorrCV           9.107790e-01
CorrCVse         1.267956e-02
AUCcv            9.927200e-01
AUCcvSE          4.487335e-03
```

3.4.11.3 Model output analysis

We can look at the relative contribution of each of the predictor variables. The measures are based on the number of times the variable is selected for splitting, weighted by the improvement of the model as a result of each split averaged across all trees. The relative contribution of each of the variables is scaled so the sum is 100%, with higher numbers indicating stronger influence in the response.

```
# Variables contribution
modelBRT$contributions
```

	var	rel.inf
GiniDBH	GiniDBH	30.363854
latitude	latitude	25.262063
volAllha	volAllha	24.038196
ShannonIndexTreeSpp	ShannonIndexTreeSpp	19.146713
forestManagementType	forestManagementType	1.189174

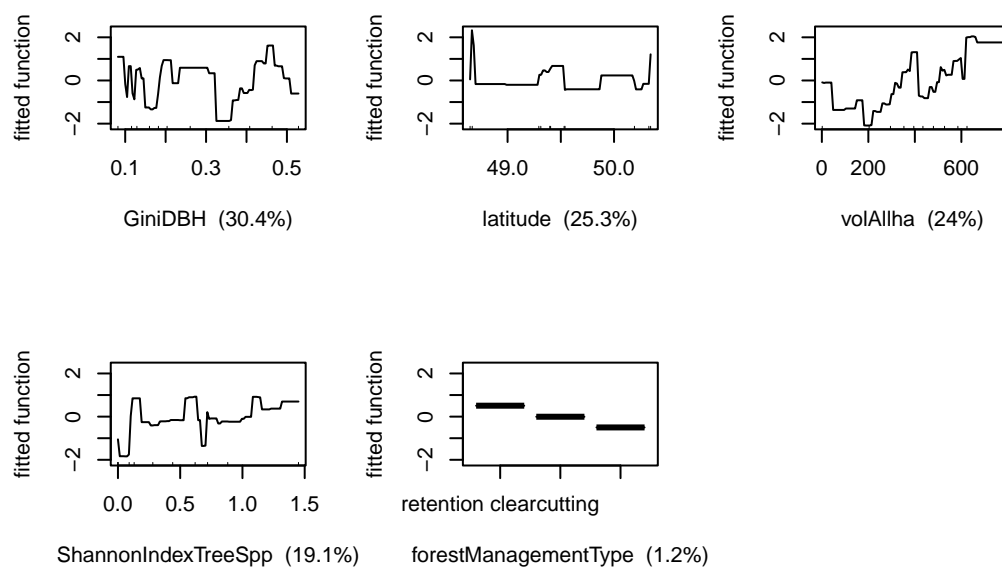
Here we can see that the two variables with the highest influence in the response are GiniDBH and volAllha.

Now we can evaluate the model behavior via partial dependence plots, showing the effect of each of the variables on the response by accounting for the average effects of all other predictors in the model:

```
dismo::gbm.plot(
  modelBRT,
  n.plots = 6,
  plot.layout = c(2, 3),
  write.title = F
)
```

Warning in dismo::gbm.plot(modelBRT, n.plots = 6, plot.layout = c(2, 3), :
reducing no of plotted predictors to maximum available (5)

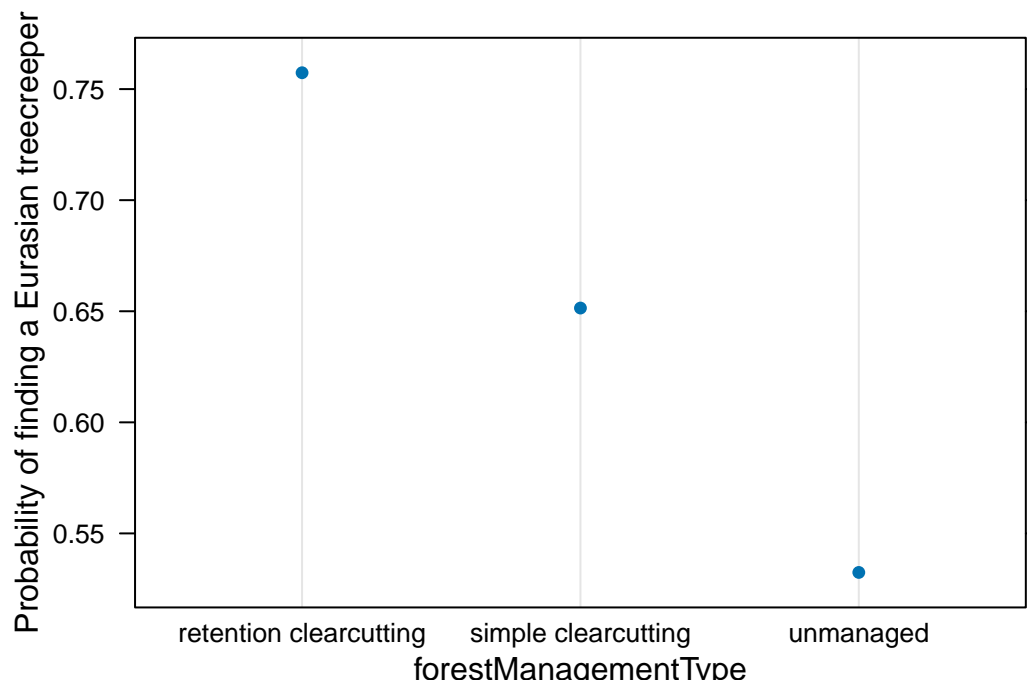
3 Empirical modeling



In this partial dependence plots the predictions are on the scale of $f(x)$. In this case, for the Bernoulli loss the returned value is on the log odds scale. You can see how this plot will look by plotting with the function from the package `gbm` and using the type “response”. Since we are interesting in finding out if the forest density has an impact in the presence of the Eurasian treecreeper affected we can have a look to the plot density variable `volAllha` :

```
gbm::plot.gbm(modelBRT,
  i.var = 2,
  type = "response",
  ylab = "Probability of finding a Eurasian treecreeper")
```

3 Empirical modeling

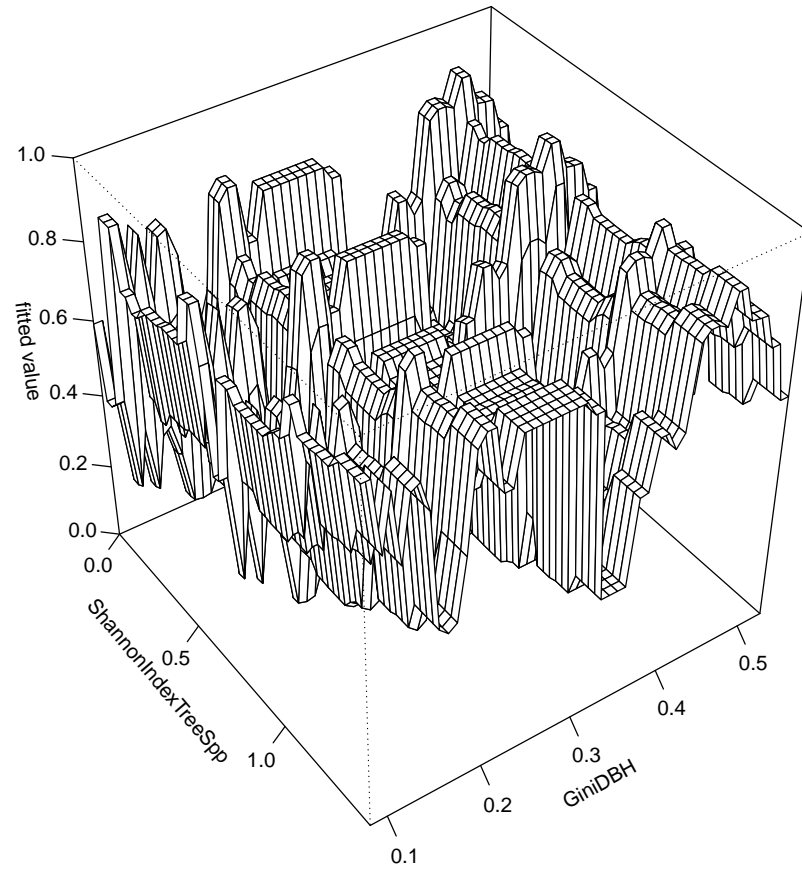


It seems that there is a small difference in the probability of finding the Eurasian treecreeper different management strategies. We could also analyse the interaction effects, of forest diversity and forest structural diversity. The model predictions can be obtained for each pair of predictor variables, setting all other predictors to their means.

To plot this pairwise interactions we have to do:

```
dismo::gbm.perspec(modelBRT, 5, 4)
```

3 Empirical modeling



It seems that there are not clear patterns in the combined effect of forest structure diversity and forest tree species diversity.

4 References and further reading

Aguirre, O., Hui, G., von Gadow, K. and Jiménez, J., 2003. An analysis of spatial forest structure using neighbourhood-based variables. *Forest ecology and management*, 183 (1-3), pp.137-145.

Elith, J., J. R. Leathwick, and T. Hastie. 2008. "A Working Guide to Boosted Regression Trees." *Journal of Animal Ecology* 77 (4): 802–13. <https://doi.org/10.1111/j.1365-2656.2008.01390.x>.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. n.d. "ADDITIVE LOGISTIC REGRESSION: A STATISTICAL VIEW OF BOOSTING."

Gadow, K.V., Zhang, C.Y., Wehenkel, C., Pommerening, A., Corral-Rivas, J., Korol, M., Myklush, S., Hui, G.Y., Kiviste, A. and Zhao, X.H., 2012. Forest structure and diversity. *Continuous cover forestry*, pp.29-83.

Hošek, Jan. n.d. "Forest Structure and Dead Wood Properties in Six Representative Forest Areas in the Czech Republic." 165

On iLand model:

Seidl, R., Rammer, W., Scheller, R.M., Spies, T.A., 2012. An individual-based process model to simulate landscape-scale forest ecosystem dynamics. *Ecol. Model.* 231, 87-100.

Seidl, R., Rammer, W., Blennow, K. 2014 Simulating wind disturbance impacts on forest landscapes: Tree-level heterogeneity matters. *Environmental Modelling and Software* 51, 1-11.

Rammer, W., Seidl, R., 2015 Coupling human and natural systems: Simulating adaptive management agents in dynamically changing forest landscapes. *Global Environmental Change*, 35, 475-485.

Rammer, W., Seidl, R., 2015 Coupling human and natural systems: Simulating adaptive management agents in dynamically changing forest landscapes. *Global Environmental Change*, 35, 475-485.

On Applications:

Thom, D., Rammer, W., Garstenauer, R., & Seidl, R. ,2018 Legacies of past land use have a stronger effect on forest carbon exchange than future climate change in a temperate forest landscape. *Biogeosciences*, 15(18), 5699–5713.

4 References and further reading

- Silva Pedro, M., Rammer, W., & Seidl, R., 2017 Disentangling the effects of compositional and structural diversity on forest productivity. *Journal of Vegetation Science*, 28(3), 649–658.
- Dobor, L., Hlásny, T., Rammer, W., Barka, I., Trombik, J., Pavlenda, P., Seidl, R., 2018 Post-disturbance recovery of forest carbon in a temperate forest landscape under climate change. *Agricultural and Forest Meteorology*, 263, 308–322.
- Albrich, K., Rammer, W., Thom, D., & Seidl, R., 2018 Trade-offs between temporal stability and level of forest ecosystem services provisioning under climate change. *Ecological Applications*, 28(7), 1884–1896.
- Dobor, L., Hlásny, T., Rammer, W., Zimová, S., Barka, I., & Seidl, R., 2020 Spatial configuration matters when removing windfelled trees to manage bark beetle disturbances in Central European forest landscapes. *Journal of Environmental Management*, 254
- Honkaniemi, J., Rammer, W., & Seidl, R. (2020). Norway spruce at the trailing edge: the effect of landscape configuration and composition on climate resilience. *Landscape Ecology*, 35(3), 591–606.
- Sommerfeld, A., Rammer, W., Heurich, M., Hilmers, T., Müller, J., & Seidl, R. (2020). Do bark beetle outbreaks amplify or dampen future bark beetle disturbances in Central Europe? *Journal of Ecology*, July, 1–13.

5 Appendix - Initialize trees for the model based on inventory data

5.1 Inventory data

We had inventory data prior the summer school. These plots were established in 2007 for monitoring of urban forest Rožnik. Those four plots were already remeasured by NFI teams (for yearly training) at the beginning of April 2023. The radius of the plots is 13.8m and trees $dbh \geq 10$ were measured.

5.2 iLand needs

We need to produce tree list with tree positions, species, dbh and height for a 1ha area, that is the smallest unit in the model that can be individually simulated. We will use a so called “torus” when the simulation space is treated as a torus, where any influence (e.g. a light influence pattern) leaving on one side again enters at the opposite site. This is especially useful for small simulated areas to provide a continuous environment without edge effects.

Idea: use the measured trees as the middle of our 1ha area, keep the trees and locations as they are, but populate the rest of the 1ha randomly based on the characteristics of the measured plot.

This script is using these input files which are placed here: iLand_simulations/init/trees/ -testPlot_data_Roznik.xlsx -species_names.csv

And making outputs as text files for each four plots with tree lists that are directly can go into model iLand as initialization file.

5.2.1 Read in the inventory data to R visualize it. Read also a table with species latin names and short names in iLand.

5 Appendix - Initialize trees for the model based on inventory data

```
a <-
  openxlsx::read.xlsx(here::here("model/iLand_simulations/init/trees/testPlot_data_Roz"),
  head(a)
```

	plotID	treeID	treeSp	dbh	azimuth	h_dist.(dm)	X	Y
1	3	1	Fagus sylvatica	65.8	29.0	118.3	5.736914	10.3496665
2	3	2	Pinus sylvestris	56.7	91.5	66.0	6.597738	-0.1727679
3	3	3	Pinus sylvestris	46.2	151.5	122.0	5.821337	-10.7215688
4	3	4	Pinus sylvestris	48.6	165.0	78.3	2.025259	-7.5583696
5	3	5	Fagus sylvatica	41.5	192.7	85.0	-1.863868	-8.2931294
6	3	6	Fagus sylvatica	50.5	197.0	83.7	-2.446177	-8.0010831

	height.(dm)
1	338.0
2	309.3
3	289.3
4	294.5
5	338.8
6	332.0

```
# read table with latin and iLand names:
sp <-
  utils::read.table(
    here::here("model/iLand_simulations/init/trees/species_names.csv"),
    header = T,
    sep = ";"
  )

# change the column names to have the latin name column similar name as in "a"
colnames(sp) <- c("id", "iLand_name", "treeSp")

# add iLand names as a new column:
a <- dplyr::left_join(a, sp, by = "treeSp")

g1 <- ggplot2::ggplot(a, ggplot2::aes(X, Y, color = iLand_name)) +
  ggplot2::geom_point(size = a$dbh / 10) +
  ggplot2::facet_wrap(~ plotID, nrow = 2) +
  ggplot2::theme_bw() +
  ggplot2::theme(
    panel.grid.major = ggplot2::element_blank(),
    panel.grid.minor = ggplot2::element_blank(),
```

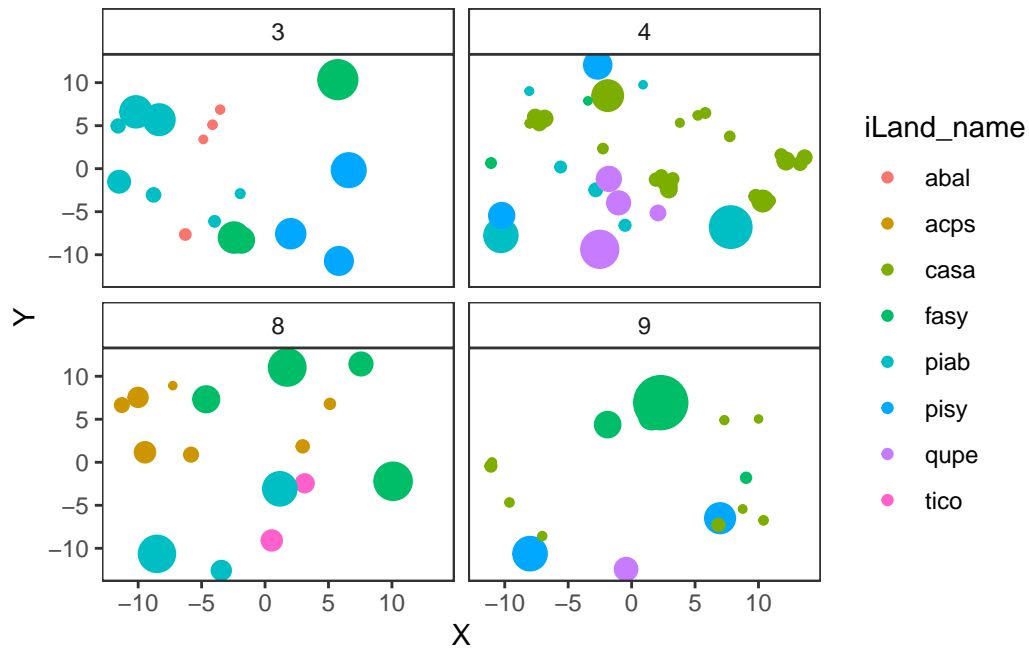
5 Appendix - Initialize trees for the model based on inventory data

```

panel.background = ggplot2::element_blank(),
strip.background = ggplot2::element_rect(fill = "white")
)

print(g1)

```



5.2.2 Calculate some statistics for populating remaining areas:

Not all records have height, so we calculate H:D ratios for each tree that has.

```

# Area of the plots:
r <- 13.82 #m, radius of the circular plot
plot.area <- r * r * pi #m2

tree.stats <- a |> dplyr::group_by(plotID, treeSp, iLand_name) |>
  dplyr::summarize(
    mean.dbh = mean(dbh),
    min.dbh = min(dbh),
    max.dbh = max(dbh),
    sd.dbh = sd(dbh) ,
    n = dplyr::n()
  )

```


5 Appendix - Initialize trees for the model based on inventory data

```
) |>
  dplyr::mutate(n.perha = n * 10000 / plot.area)

tree.stats$sd.dbh[is.na(tree.stats$sd.dbh) == T] <- 10

# Look the H:D ratio for each trees:
a <- a |> dplyr::mutate(hdratio = 10 * `height.(dm)` / dbh)

# Species specific H:D ratio ranges (to calculate height for trees which are not
#given:
hdratios <- a |> dplyr::group_by(treeSp, iLand_name) |>
  dplyr::summarize(
    mean.hdratio = mean(hdratio, na.rm = T),
    min.hdratio = min(hdratio, na.rm = T),
    max.hdratio = max(hdratio, na.rm = T),
    n = dplyr::n()
  )

# abies alba was missing hd ratios, there were no height for any abal trees:
hdratios$mean.hdratio[hdratios$iLand_name == "abal"] <- 80

print(hdratios)
```

```
# A tibble: 8 x 6
# Groups:   treeSp [8]
  treeSp          iLand_name mean.hdratio min.hdratio max.hdratio      n
  <chr>          <chr>          <dbl>      <dbl>      <dbl> <int>
1 Abies alba      abal              80         Inf        -Inf      4
2 Acer pseudoplatanus acps          85.6        78.2        98.4      7
3 Castanea sativa  casa          83.1        62.6       109.     32
4 Fagus sylvatica  fasy          56.8        31.4        81.6     13
5 Picea abies      piab          67.2        46.2       85.5     17
6 Pinus sylvestris pisy          56.8        49.8        62.6      7
7 Quercus petraea  qupe          51.2        43.3        59.1      5
8 Tilia cordata    tico          76.3        70.2        82.4      2
```

5.2.3 Generate trees for each plot

- Using the number of trees/ha as required number of trees.
- Generate dbh that normally distributed using mean dbh and stdev

5 Appendix - Initialize trees for the model based on inventory data

- Randomly placed: x,y coordinates -50m -> 50m. (inventory data had 0,0 coordinate in the middle, so we kept this “coordinate system” for now)

```
ids <- unique(a$plotID)
tree.list <- c()
for (i in 1:4) {
  # go on plots

  a1 <- tree.stats |> dplyr::filter(plotID == ids[i])

  for (j in 1:length(a1$treeSp)) {
    # go on species
    spec1 <- a1[j, ]
    ntree <- round(spec1$n.perha)
    print(paste(
      "We need ",
      ntree,
      spec1$iLand_name ,
      "trees per ha,for plot id: ",
      ids[i]
    ))

    # tree list for one species:

    trees <- data.frame(
      plotID = rep(ids[i], ntree),
      species = rep(spec1$iLand_name, ntree),
      dbh = truncnorm::rtruncnorm(ntree, 5, 100,
                                   spec1$mean.dbh, spec1$sd.dbh),
      X = runif(ntree) * 100 - 50,
      Y = runif(ntree) * 100 - 50
    )

    tree.list <- rbind(tree.list, trees)
  }
}
```

```
[1] "We need 67 abal trees per ha,for plot id: 3"
[1] "We need 50 fasy trees per ha,for plot id: 3"
[1] "We need 117 piab trees per ha,for plot id: 3"
[1] "We need 50 pisy trees per ha,for plot id: 3"
```

5 Appendix - Initialize trees for the model based on inventory data

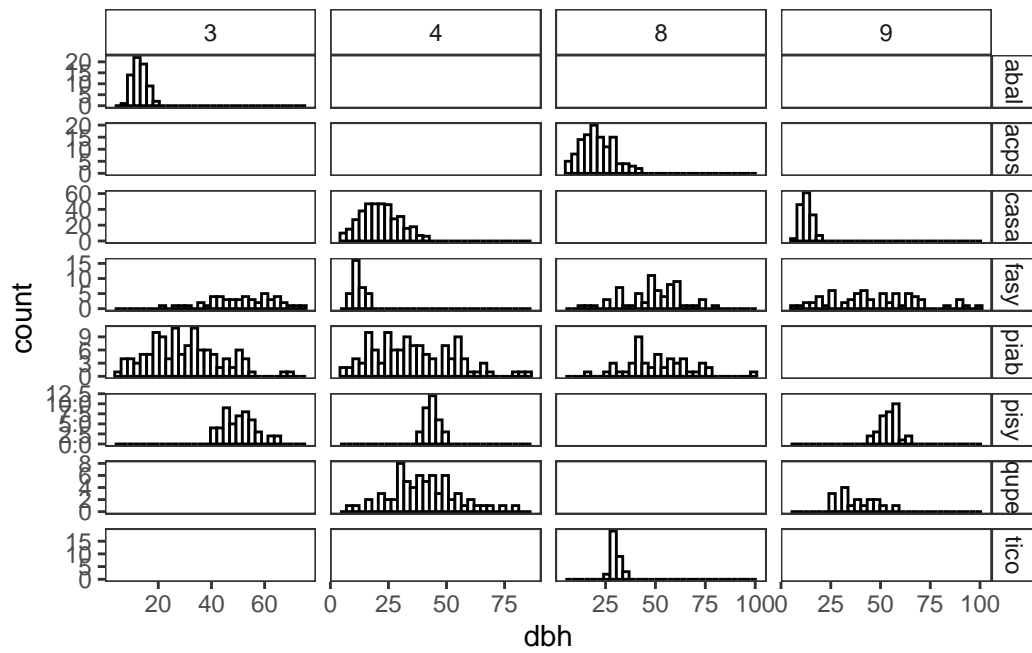
```
[1] "We need 383 casa trees per ha,for plot id: 4"
[1] "We need 33 fasy trees per ha,for plot id: 4"
[1] "We need 117 piab trees per ha,for plot id: 4"
[1] "We need 33 pisy trees per ha,for plot id: 4"
[1] "We need 67 qupe trees per ha,for plot id: 4"
[1] "We need 117 acps trees per ha,for plot id: 8"
[1] "We need 67 fasy trees per ha,for plot id: 8"
[1] "We need 50 piab trees per ha,for plot id: 8"
[1] "We need 33 tico trees per ha,for plot id: 8"
[1] "We need 150 casa trees per ha,for plot id: 9"
[1] "We need 67 fasy trees per ha,for plot id: 9"
[1] "We need 33 pisy trees per ha,for plot id: 9"
[1] "We need 17 qupe trees per ha,for plot id: 9"
```

```
g2 <- ggplot2::ggplot(tree.list, ggplot2::aes(dbh)) +
  ggplot2::geom_histogram(color = "black", fill = "white") +
  ggplot2::facet_grid(species ~ plotID, scales = "free") +
  ggplot2::theme_bw() + ggplot2::theme(
    panel.grid.major = ggplot2::element_blank(),
    panel.grid.minor = ggplot2::element_blank(),
    panel.background = ggplot2::element_blank(),
    strip.background = ggplot2::element_rect(fill = "white")
  )

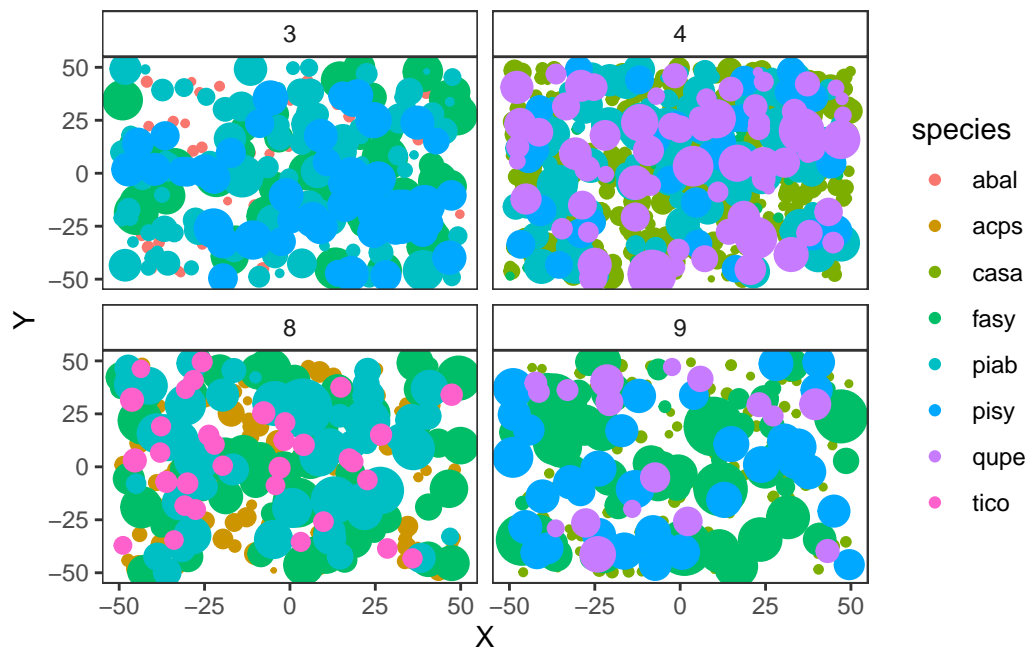
g3 <- ggplot2::ggplot(tree.list, ggplot2::aes(X, Y, color = species)) +
  ggplot2::geom_point(size = tree.list$dbh / 10) +
  ggplot2::facet_wrap( ~ plotID, nrow = 2) +
  ggplot2::xlim(-50, 50) + ggplot2::ylim(-50, 50) +
  ggplot2::theme_bw() + ggplot2::theme(
    panel.grid.major = ggplot2::element_blank(),
    panel.grid.minor = ggplot2::element_blank(),
    panel.background = ggplot2::element_blank(),
    strip.background = ggplot2::element_rect(fill = "white")
  )

print(g2)
```

5 Appendix - Initialize trees for the model based on inventory data



```
print(g3)
```



5.2.4 Lets remove the trees in the middle, and put back the real trees from the inventory

```
# Calculate the distance for each tree from the center point:
tree.list <- tree.list |> dplyr::mutate(distance = sqrt(X * X + Y * Y))

# Look which we want to keep: where distance is larger than the plot radius was:
cutted.tree.list <- tree.list |> dplyr::filter(distance > r)

g4 <- ggplot2::ggplot(cutted.tree.list,
                      ggplot2::aes(X, Y, color = species)) +
  ggplot2::geom_point(size = cutted.tree.list$dbh / 10) +
  ggplot2::facet_wrap(~ plotID, nrow = 2) +
  ggplot2::xlim(-50, 50) + ggplot2::ylim(-50, 50) +
  ggplot2::theme_bw() + ggplot2::theme(
    panel.grid.major = ggplot2::element_blank(),
    panel.grid.minor = ggplot2::element_blank(),
    panel.background = ggplot2::element_blank(),
    strip.background = ggplot2::element_rect(fill = "white")
  )

# Check our inventory trees:
trees.circle = data.frame(
  plotID = a$plotID,
  species = a$iLand_name,
  dbh = a$dbh,
  X = a$X,
  Y = a$Y,
  distance = sqrt(a$X * a$X + a$Y * a$Y)
)

g5 <- ggplot2::ggplot(trees.circle, ggplot2::aes(X, Y, color = species)) +
  ggplot2::geom_point(size = trees.circle$dbh / 10) +
  ggplot2::facet_wrap(~ plotID, nrow = 2) +
  ggplot2::xlim(-50, 50) + ggplot2::ylim(-50, 50) +
  ggplot2::theme_bw() + ggplot2::theme(
    panel.grid.major = ggplot2::element_blank(),
    panel.grid.minor = ggplot2::element_blank(),
    panel.background = ggplot2::element_blank(),
    strip.background = ggplot2::element_rect(fill = "white")
  )
```

5 Appendix - Initialize trees for the model based on inventory data

```
# put them all together:
all.trees <- rbind(trees.circle, cutted.tree.list)
```

5.2.5 Give height and plot final layout

```
# add hd ratio to each record and calculate height:
all.trees2 <-
  dplyr::left_join(all.trees, hdratios[, c("iLand_name", "mean.hdratio")],
    by = c("species" = "iLand_name")) |>
  dplyr::mutate(height = dbh * mean.hdratio / 100)

summary(all.trees2$height)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
3.405	11.929	19.103	20.108	26.403	65.877

```
g6 <- ggplot2::ggplot(all.trees, ggplot2::aes(X, Y, color = species)) +
  ggplot2::geom_point(size = all.trees$dbh / 10) +
  ggplot2::facet_wrap(~ plotID, nrow = 2) +
  ggplot2::xlim(-50, 50) + ggplot2::ylim(-50, 50) +
  ggplot2::theme_bw() + ggplot2::theme(
    panel.grid.major = ggplot2::element_blank(),
    panel.grid.minor = ggplot2::element_blank(),
    panel.background = ggplot2::element_blank(),
    strip.background = ggplot2::element_rect(fill = "white")
  )
```

5.2.6 Make files for iLand:

Coordinates need to be shifted: 0, 0 is the bottom left corner. So we shift everything back by 50m.

```
# Make the tree list for iLand:
#https://iland-model.org/initialize+trees
#
## this is a sample tree input file
# note that the height is given in meter
# the 2nd tree has no age
# x;y;species;dbh;height;age
```

5 Appendix - Initialize trees for the model based on inventory data

```
# 31.11;8.01;piab;42.2;29.7;120
# 21.11;15.11;piab;52.8;33.5;0
# 11.51;25.11;fasy;55;35;100

plotids <- unique(all.trees2$plotID)
for (i in 1:4) {
  d <- all.trees2 |> dplyr::filter(plotID == plotids[i])

  # order them by dbh!
  d <- d[order(d$dbh, decreasing = T), ]

  dat <-
    data.frame(
      x = d$X + 50,
      y = d$Y + 50,
      species = d$species,
      dbh = d$dbh,
      height = d$height,
      age = 0
    )

  out <-
    paste0(here::here("model/iLand_simulations/init/trees/Tree_init_plot_"),
           i,
           ".txt")
  write.table(
    dat,
    out,
    sep = ";",
    col.names = T,
    row.names = F,
    quote = F
  )
}
```