Table of Contents

Requirements

This program requires Blender, Python (version 3.0 or higher), R, HandBrake, and HandBrakeCLI, as well as some special treatment of HandbrakeCLI. You will also need to install two R libraries, which is best done with RStudio run in Administrative mode; an explanation of how to do this is at the end of this section.

If you are missing any of these, or don't know whether you have them or not, please use the following links to install the requirements.

- Blender: https://www.blender.org/download/
- Python: https://www.python.org/downloads/
    - Note: When installing, click "Customize" before clicking "Install" and make sure "Unix command-line tools" is checked. If you forget to do this part, Python recommends simply running the installer again.
- R: https://courses.edx.org/courses/UTAustinX/UT.7.01x/3T2014/56c5437b88fa43cf828bff5371c6a924/
    - Note: This program assumes version 3.6.0, which is the current version as of writing this manual. If you download or are using another version, please see Advanced Adjustments: Changing the R version for how to check your current version and ensure the program uses it.
- Handbrake: https://handbrake.fr/rotation.php?file=HandBrake-1.2.2-x86_64-Win_GUI.exe
- HandbrakeCLI: https://handbrake.fr/rotation.php?file=HandBrakeCLI-1.2.2-win-x86_64.zip
    - Note: You may notice that HandBrake and HandBrakeCLI are very similar. HandBrakeCLI is required for the program to run, and HandBrake is required for HandBrakeCLI to work; both downloads are necessary.

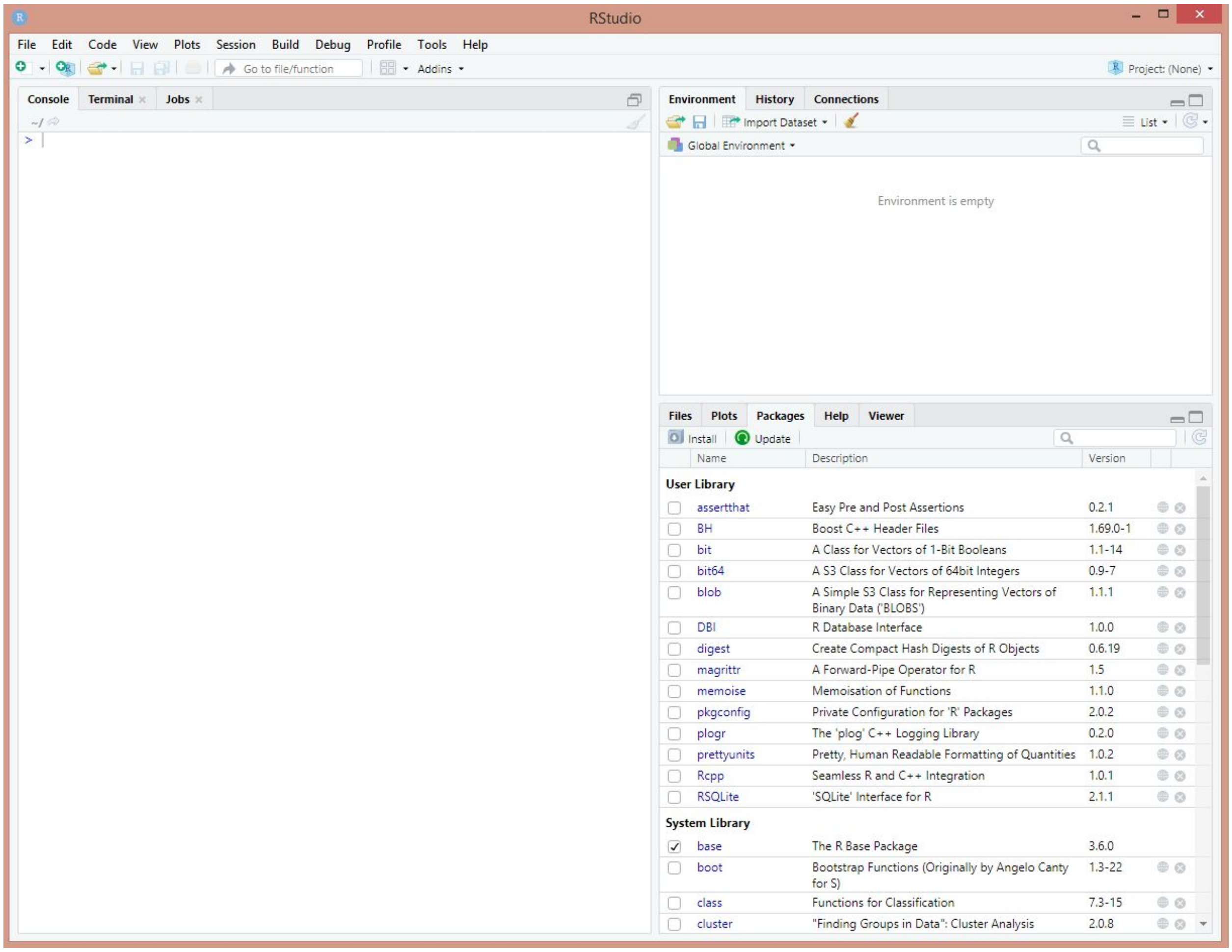
Before running the program, HandBrakeCLI must be in your Program Files. You will need to unzip the downloaded files (you can install Winzip here, simply click Try it Free), and then open a second window of the File Explorer; you can do this by right clicking the icon on the system tray at the bottom of your screen, and selecting "File Explorer." Your new window should be on the This PC page by default; double-click the Local Disk (C:).

Drag HandBrakeCLI from its unzipped folder and drop it onto “Program Files” (not “Program Files x86”), so that it is put inside that folder.

You will need to give Administrator permissions for this, as all changes to Program Files requires this. Once done, you can check that it is in the correct location by opening the “Program Files” folder and scrolling to the bottom to make sure it is there.

Windows does not allow R to install libraries when run from the Command Prompt, and so you must install two libraries, DBI and RSQLite, before running the program. To do so, you will need to run R or RStudio in Administrative mode; RStudio is the more intuitive route, and the process with RStudio will be outlined here.

To begin, you will need RStudio and Rtools. RStudio can be found here; select the free version. Rtools can be found here; the best version is typically highlighted in green. Once both are installed, navigate to RStudio in your programs. Right click on the icon, and select “Run as Administrator.” Windows will prompt you to allow the program to make changes to your computer; select yes, and input the Administrator password if necessary. RStudio will open, and appear as below:



	Name	Description	Version
	User Library		
□	assertthat	Easy Pre and Post Assertions	0.2.1
□	BH	Boost C++ Header Files	1.69.0-1
□	bit	A Class for Vectors of 1-Bit Booleans	1.1-14
□	bit64	A S3 Class for Vectors of 64bit Integers	0.9-7
□	blob	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.1.1
□	DBI	R Database Interface	1.0.0
□	digest	Create Compact Hash Digests of R Objects	0.6.19
□	magrittr	A Forward-Pipe Operator for R	1.5
□	memoise	Memoisation of Functions	1.1.0
□	pkgconfig	Private Configuration for 'R' Packages	2.0.2
□	plogr	The 'plog' C++ Logging Library	0.2.0
□	prettyunits	Pretty, Human Readable Formatting of Quantities	1.0.2
□	Rcpp	Seamless R and C++ Integration	1.0.1
□	RSQLite	'SQLite' Interface for R	2.1.1
	System Library		
☑	base	The R Base Package	3.6.0
□	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-22
□	class	Functions for Classification	7.3-15
□	cluster	"Finding Groups in Data": Cluster Analysis	2.0.8

Click on the left box, and type in the following, then hit enter:

```
        install.packages(“DBI”)
```

Once the “>” appears again, type in the following and hit enter again:

```
        install.packages(“RSQLite”)
```

The second package may take longer to install than the first; there are a number of dependencies. Speed of installation will depend on your internet connection. Once you see the “>” again, the installation is complete, and you can close RStudio. This process only needs to be completed once, and then you will be able to run AutoTrees.

Note: this manual assumes that your .db is formatted in such a way that there is a table called cohorts, with columns cohortID, year, species, trees, diameter, and height. The R script’s SQL calls may need to be edited if this is not the case.

Running AutoTrees

I. Once you have downloaded and unzipped AutoTrees_v2_Windows (you can install Winzip here, simply click Try it Free), copy your .db file and paste it into the AutoTrees_v2_Windows.
    - Note: this program relies on all of its files being kept in the same folder. Please move all files in the folder together if you wish to change the location of the program.

II. Open the Start menu with the Windows key, and then click on the downwards arrow in the bottom left corner of the screen to navigate to your programs. Scroll all the way to the right, and open “Command Prompt” under “Windows System”.
    - Note: this is for Windows 8. Other Windows versions will also have Command Prompt in the system tools, under System Tools, Utilities, or a similar title.

III. Type “py -3 ” and then the path to AutoTrees_v2.py; if you downloaded AutoTrees_v2_Windows in your Downloads folder, the path will be “./Downloads/AutoTrees_v2_Windows/AutoTrees_v2.py”. Once you have typed in the path, press enter.
    - Note: paths are case sensitive; “downloads” is different from “Downloads”.

IV. You will be prompted to enter the name of the .db file you placed into the folder. Type it in, including “.db” at the end, and press enter.
    - Note: this is case sensitive, like the paths in step III.

V. The program will now run on its own. The Terminal will add information to the screen as the run progresses; please do not close, type into, or interact with the Terminal window until the process is complete, signified by the last line “HandBrake has exited.” and Terminal giving you an entry line with your computer name and username once more.
    - Note: An unresponsive Blender window with a greyed out screen will appear while the Blender portions runs; this is normal. You can minimize the window if it is in the way, and Blender will close on its own when it is finished working. Do not use Task Manager to close Blender. If Windows prompts you to close Blender for being unresponsive, do not close the program. Blender will return to responsiveness when the script it is running ends, and then it will close.
    - Note: You may see “Warning message: All arguments to RSQLite Driver are ignored.” This is normal and is not a problem.
    - Note: You may see "Error: Not freed memory blocks: 1, total unfreed memory 0.000076 MB” when the Blender portion of the program completes. This is normal, even if it has a slightly different number.
    - Note: When HandBrakeCLI begins to run, you may notice it run through a few quick checks and give errors or say certain actions failed or are unavailable; this is HandBrake’s process of checking what file it is using, and is normal.

VI. The process may take a long time to complete, depending on the size of your dataset; anywhere from half an hour to overnight for larger projects. You will need to leave your computer on and Terminal running, even if it appears to not be doing anything; for certain phases of the program, it will not output information as it works, and these are typically the longest phases.

VII. Once the program has finished running, signified by the last line “HandBrake has exited.” and Terminal giving you an entry line with your computer name and username once more, you will see a video file called “final.mp4” with your animation. Please move the file out of AutoTrees_v2_Windows and rename it to whatever you wish.
   - Note: Should you not remove final.mp4 from AutoTrees_v2_Windows, the next time AutoTrees_v2.py runs it with either overwrite the old final.mp4 or fail to create a new version. It is important that you move your old file, whether you wish to keep it or not, due to the chance of HandBrake running into an error overwriting it.

VIII. The process is complete. If you would like to run the program again on a different dataset, simply repeat the process outlined here. If you would like to tweak the final animation, or an error has occurred during the Blender phase, please see Advanced Adjustments.

Advanced Adjustments

- Changing the R version

  AutoTrees v2 is designed to work with R 3.6.0 by default. This is due to how R is installed in the Program Files; each version is named differently, and so it must be called by that name.

  To check your version of R, open Command Prompt as described in step II of Running AutoTrees, type in the following, and hit enter:

  ```
  dir “C:\Program Files\R”
  ```

  Command Prompt will list the files kept in the R folder; typically they will be in the format R-x.x.x depending on version. Keep track of the name for the R version you have that you would like to use.

  Next, right click on AutoTrees_v2.py, select “Open with…”, “More options”, and then scroll down to Notepad. Uncheck the box above if you don’t want Notepad to become the default program for Python files, and then click on Notepad to open AutoTrees_v2.py.

  Scroll down to the following lines:

```
rcmd = '\"C:\\Program Files\\R\\R-3.6.0\\bin\\RScript\" ' + basedir + '\\unpack_v2.R' +
' ' + str(basedir) + ' ' + dbfile
subprocess.run(rcmd.split())
```

  Change “R-3.6.0” to the name of the version of R you would like to use, and then save AutoTrees_v2.py. It will now use your version of R.

  Note: It is recommended that you use the most recent version of R available to you if you do not have 3.6.0. unpack_v2.R has not been tested with any other versions of R; older versions may not support all features of the script, and major changes in new versions could cause unexpected interactions.

- Changing the animation models

  animation_v2.py uses Trees.blend, the provided model, by default. To change what model is used, you will need a .blend file (as the script is not made to handle other file types) and the path to that file. If you are unsure of how to build a path, you can put your .blend file into the same folder as the project files; you will only need the file name Then. In addition, you will need to open the .blend file in blender, right click on the model you want to use, and look at the name in the bottom left corner. This is your model name.

  The first step is to open animation_v2.py in Notepad. Right click on animation_v2.py, select “Open with…”, “More options”, and then scroll down to Notepad. Uncheck the box above if you don’t want Notepad to become the default program for Python files, and then click on Notepad to open animation_v2.py.

  Once animation_v2.py is open, scroll down to the section “`#--- Model data`”. You will want to change the variable “blendfile”; if your new .blend file is in the same folder as the program files, you will only need to change Trees.blend to the name of your file. (Remember to include the .blend at the end!) If you would like to keep the .blend file in a different folder, you will need to replace the entirety of “`basedir + '\\Trees.blend'`” with the path. Put single quotes (') on either side of the path, and replace any “\” with “\\”, then go to “obj”. Change tree to the model name you found in Blender; remember it must be surrounded by single quotes. Save animation_v2.py, and you are ready to run AutoTrees_v2.py with your new model.

  > Note: you may find that you need to adjust model scaling with your new model. See next section for details on how to do this. It is recommended that you run the program once to get an idea of how much the scale will need to change.

- Changing model scale

  If the models in your animation are not the size you would like them to be, either too large or two small, you can adjust their scale using two variables in animation_v2.py.

  The first step is to open animation_v2.py in Notepad. Right click on animation_v2.py, select “Open with…”, “More options”, and then scroll down to Notepad. Uncheck the box above if you don’t want Notepad to become the default program for Python files, and then click on Notepad to open animation_v2.py.

  Once animation_v2.py is open, scroll down to section “`#--- Scaling`”. You will see three variables: scale, xmod, and zmod. It is recommended that you use xmod and zmod, and leave scale alone. xmod affects the width and length of models, while zmod affects the height; larger numbers make larger models, and vice versa. Whole numbers are not necessary, and fractions less than 1 may be necessary to get particularly large models down to size. Make sure to save any changes to animation_v2.py before running AutoTrees_v2 again; you may need to go back and forth a few times to perfect the scaling.

> Note: it may be tempting to instead change the camera radius. This is not advised, as a larger radius will lose trees in the distance, and a smaller radius may lose trees along the sides of the animation. Changing scale is a much safer adjustment.

- Adjusting colors used

  There are 24 colors provided by the script to map to individual tree species. These are provided in a list called *colors*, where each color is listed as three numbers in brackets (ex. [1.0,1.0,1.0]). You can add more colors to the list by simply adding more sets of three numbers in brackets, with commas between each set as per the rest of the list, but there are several considerations.

I. Blender uses a variation on RGB. You can find the standard RGB values for a particular color using tools, such as w3school's easy to use widget here; you will note that the values for R, G, and B (red, green, and blue) are between 0 and 255. Blender uses a scale of 0 to 1.0 instead; and so you will need to take the standard values and divide them by 255 to see what you should be typing into the .py file. Note that these numbers tend to be very long decimals, but you can shorten them to the third decimal place.

   - Example 1: Bright red, with RBG values (255,0,0), would be typed in as [1.0,0,0]. 255/255 = 1.0, 0/255 = 0.

   - Example 2: A medium grey with values (144,144,144), would be typed in as [0.565,0.565,0.565]. 144/255 is 0.56470…., but we can round it to 0.565.

II. Colors will need to be selected to ensure that trees are visually distinct. There are already 24 colors provided, over a wide range of colors; pastel, medium, and dark variations of grey, red, pink, blue, teal, green, yellow, and orange were selected, to create a wide range of colors, but this means it may be difficult to find distinct colors from these. You may find that creating your own list is easier; either by hand, or using a site such as this one. (To use the linked site, click the Refine tab, adjust Threshold in the top right corner to roughly the amount of colors you need (it will not be precise), and then go to the Results tab to see a list of swatches and values.)

III. The following is a listing of the colors included by default in the .py script. Note the swatches may be off due to limitations of Google Docs (it does not represent darks or pastels well). The swatches are just to give a basic idea of what is represented.

Color swatch	RGB	Blender entry
⬜	rgb(255,255,255)	[1.0,1.0,1.0]
▨	rgb(161,161,161)	[0.633,0.633,0.633]
⬛	rgb(0,0,0)	[0.0,0.0,0.0]
⬛	rgb(42,2,42)	[0.168,0.008,0.168]
🟪	rgb(169,42,169)	[0.665,0.0168,0.665]

🟪	rgb(178,60,178)	[0.701,0.235,0.701]
🟫	rgb(56,2,2)	[0.220,0.008,0.008]
🟥	rgb(218,11,11)	[0.855,0.044,0.044]
🟥	rgb(207,66,66)	[0.815,0.262,0.262]
⬛	rgb(6,2,53)	[0.023,0.010,0.209]
🟦	rgb(15,6,162)	[0.060,0.023,0.638]
🟦	rgb(60,45,184)	[0.235,0.175,0.723]
⬛	rgb(2,40,42)	[0.008,0.156,0.168]
🟦	rgb(5,159,171)	[0.022,0.624,0.672]
🟦	rgb(51,199,207)	[0.200,0.780,0.814]
⬛	rgb(2,42,3)	[0.008,0.168,0.010]
🟩	rgb(6,175,8)	[0.024,0.687,0.032]
🟦	rgb(48,171,171)	[0.188,0.672,0.672]
⬛	rgb(43,47,2)	[0.171,0.185,0.007]
🟨	rgb(178,194,4)	[0.701,0.761,0.017]
🟨	rgb(178,184,75)	[0.701,0.723,0.292]
🟫	rgb(42,20,2)	[0.168,0.080,0.007]
🟧	rgb(207,88,3)	[0.815,0.347,0.010]
🟫	rgb(166,97,23)	[0.651,0.381,0.093]