TreeSnatcher Plus

Tutorial 3 Preprocessing tools

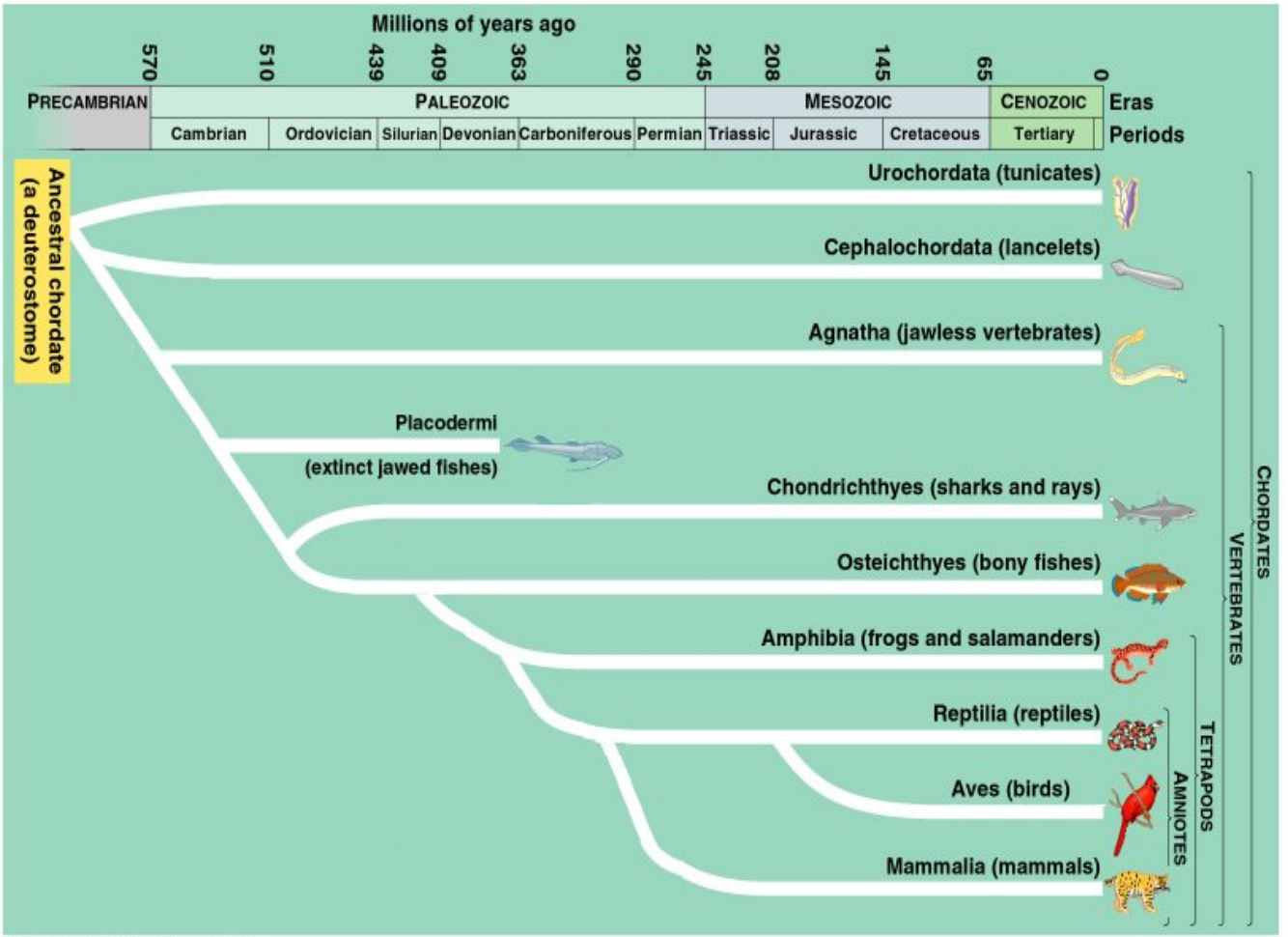
What we want

We are interested in the tree depicted in this image. As the tree is not rectangular, we select **Freeform** as tree type. It is our aim to modify the image in a way that the tree itself is black and the background is white. In most cases, there is not only one way to achieve this.

Please load the image.

Click **Whole Image**, then **Global Threshold**, try different threshold values and preview them.

If the image looks like the next one, accept it.

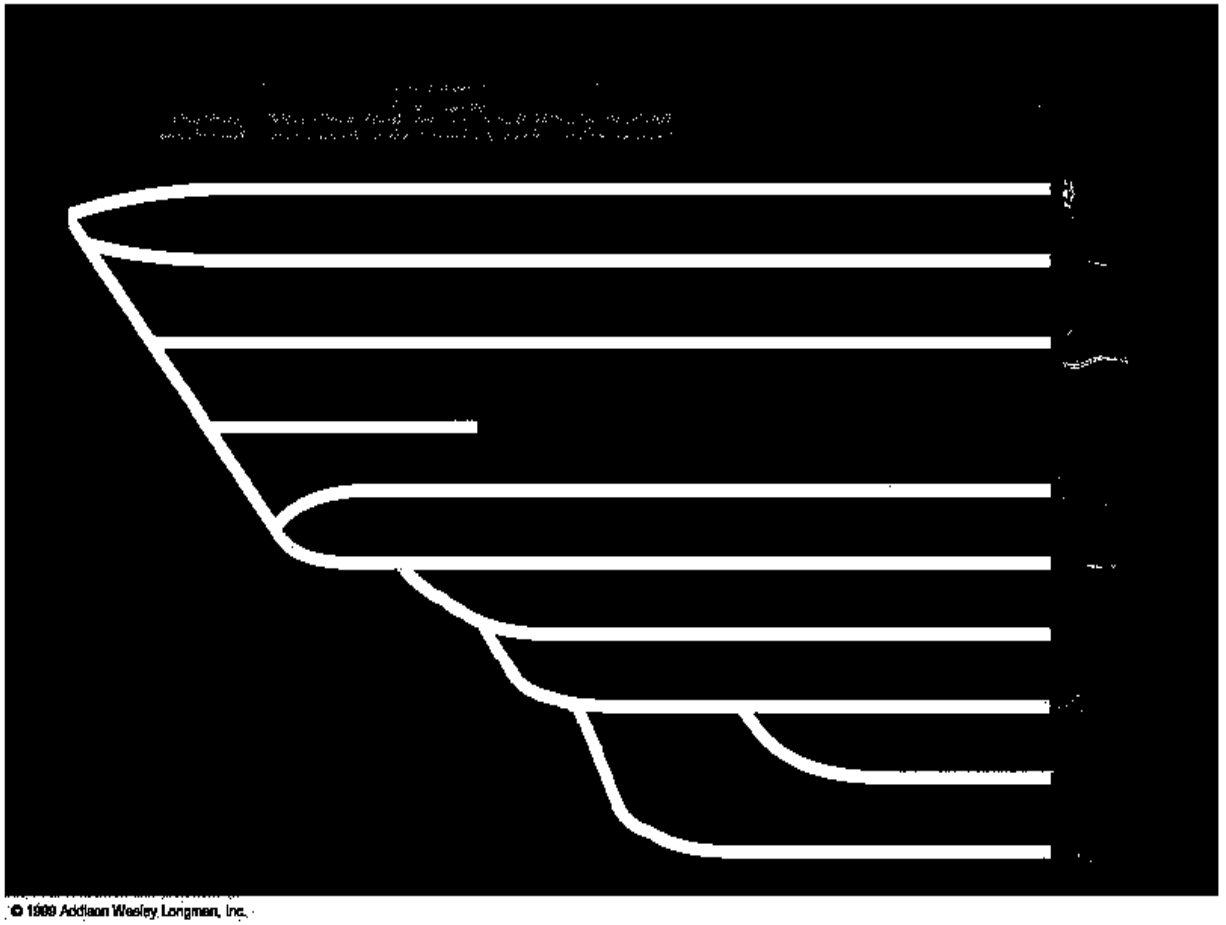


The image is binarized now. In a binarized image, there are only two shades – black and white. For the next tasks in TreeSnatcher Plus, the area which contains the tree must be binarized.

You cannot decide whether an image is binarized by visual inspection alone: The RGB color (0, 0, 0) is binary black, but (0, 1, 0) isn't. The RGB color (255, 255, 255) is binary white, but (253, 255, 254) isn't*. Also keep in mind that other preprocessing operations convert a binarized image back into a non-binary image.

Notice that the tree is clearly separated from the background.

Click **Invert**.
Now the tree is black and the background is white as needed.



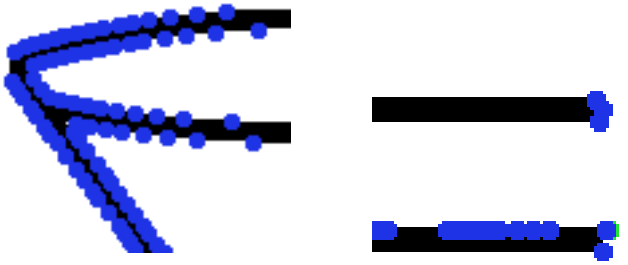
* This is not the absolute truth. A real binary image contains only the shades 0 and 1. As TreeSnatcher Plus is programmed in JAVA, we use the RGB color model.

Click with the right mouse button somewhere into the tree, then select **Flood Foreground** from the pop-up menu. Locate a suitable position on a path with the magnifier.

Starting from the position you have right-clicked, TreeSnatcher Plus colors all black pixels that can be reached from there in blue. By this, you signal the program where the tree is.

Click **Locate Nodes**.

What follows is not what we want. TreeSnatcher has stubbornly used its heuristics to choose candidate positions for inner nodes and tips. To make matters worse, there are almost only inner nodes at improper positions.

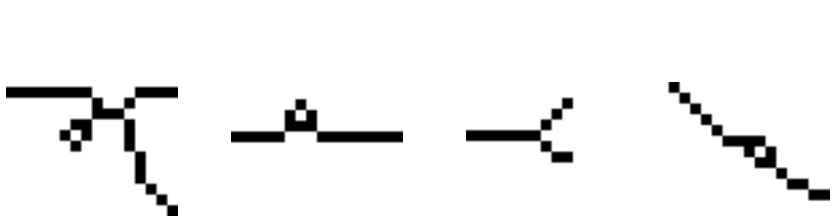


What has gone wrong?
Before we can place the nodes we must first use the preprocessing tool **Thin** to further prepare the foreground for the heuristics.

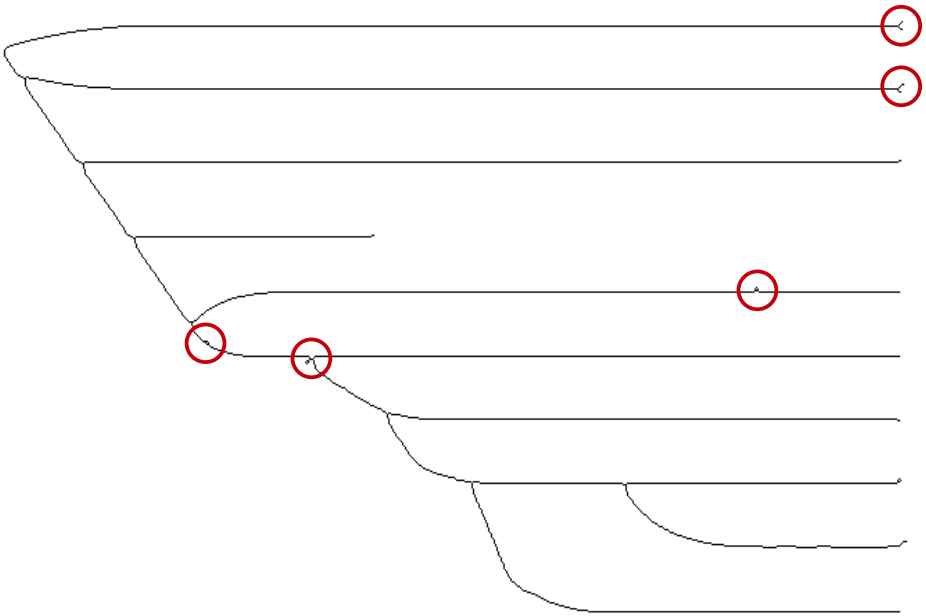
Click **Undo**, then **Thin**, then repeat **Flood Foreground,** and activate **Flood User Drawings**.

The **Thin** technique tries to thin the whole image foreground as much as possible while preserving connected components. Unfortunately it still preserves some undesired pixels. Correct the image as shown below with **Pencil** and **Rubber**.

Before correction:



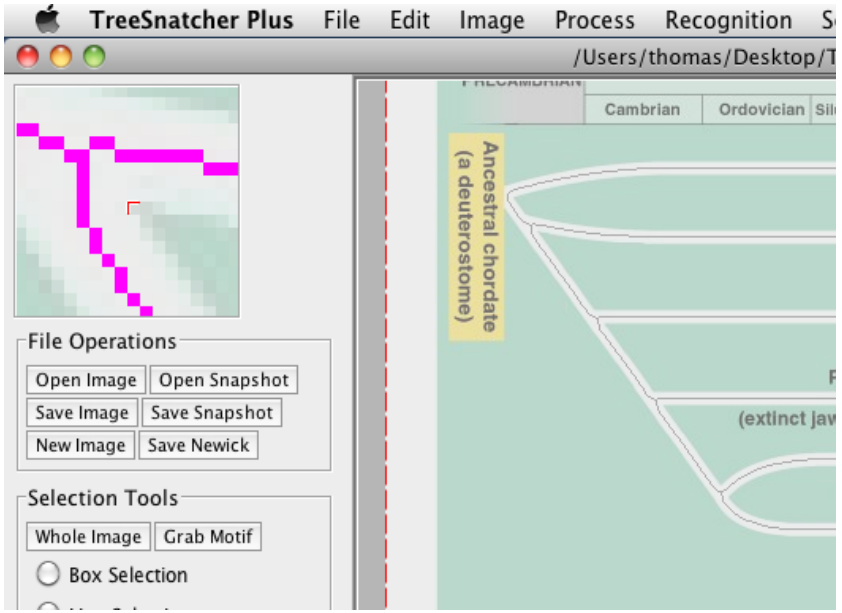
After correction:

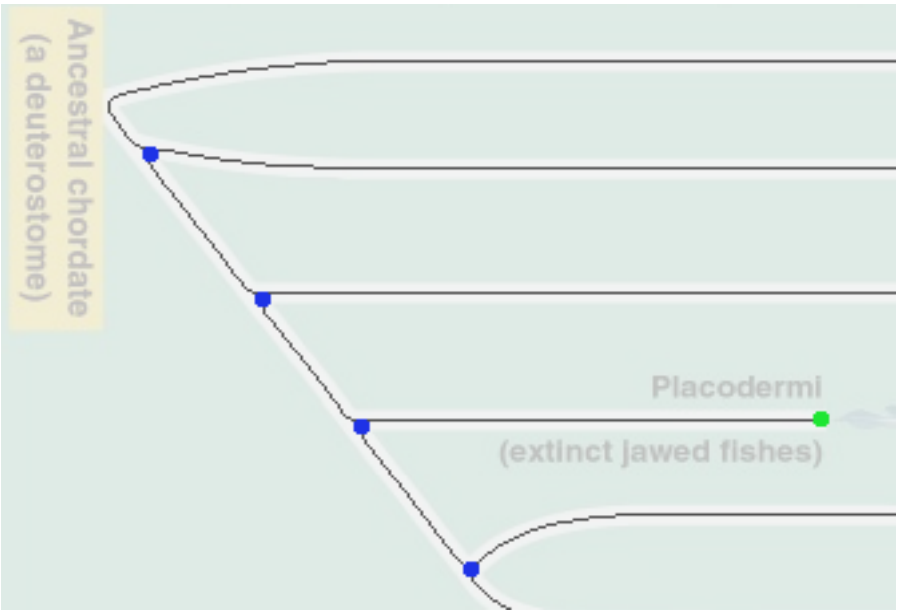


As soon as you started editing the pixels, the program flooded the foreground from the last seed position. You can mark the seed position if you select **Mark TreeFlood Seed.**

Look at the leftmost image: By removing one pixel you decoupled a group of pixels from the rest of the flooded foreground. For a better overview, click **Extract Foreground:** Only the flooded foreground is kept.

For reference, you can adjust the **Blend Ratio** between the original image and the modified image. Also select **Magnifier Settings/Show Source Image** in the menu to adopt the blend ratio for the magnifier view. You can additionally highlight all black pixels in the magnifier with the combination **Highlight Foreground** and **Select Foreground Color**.

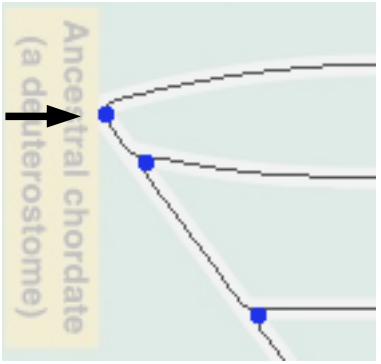




Click **Locate Nodes** again. There should be much less nodes now.

You select a node by moving the mouse over it. The selected node flashes red.
In the magnifier, nodes are represented by a red dot if selected, green otherwise.
If there are still odd nodes, select them and click **Remove Node** in the pop-up menu.

We want another branching position in the upper left part of the image. Therefore, right-click at the insertion position and select **Add Inner Node** from the pop-up menu. Select the new node and click **Use Node as Origin** from the pop-up menu. The program will build the Newick expression recursively from this node.

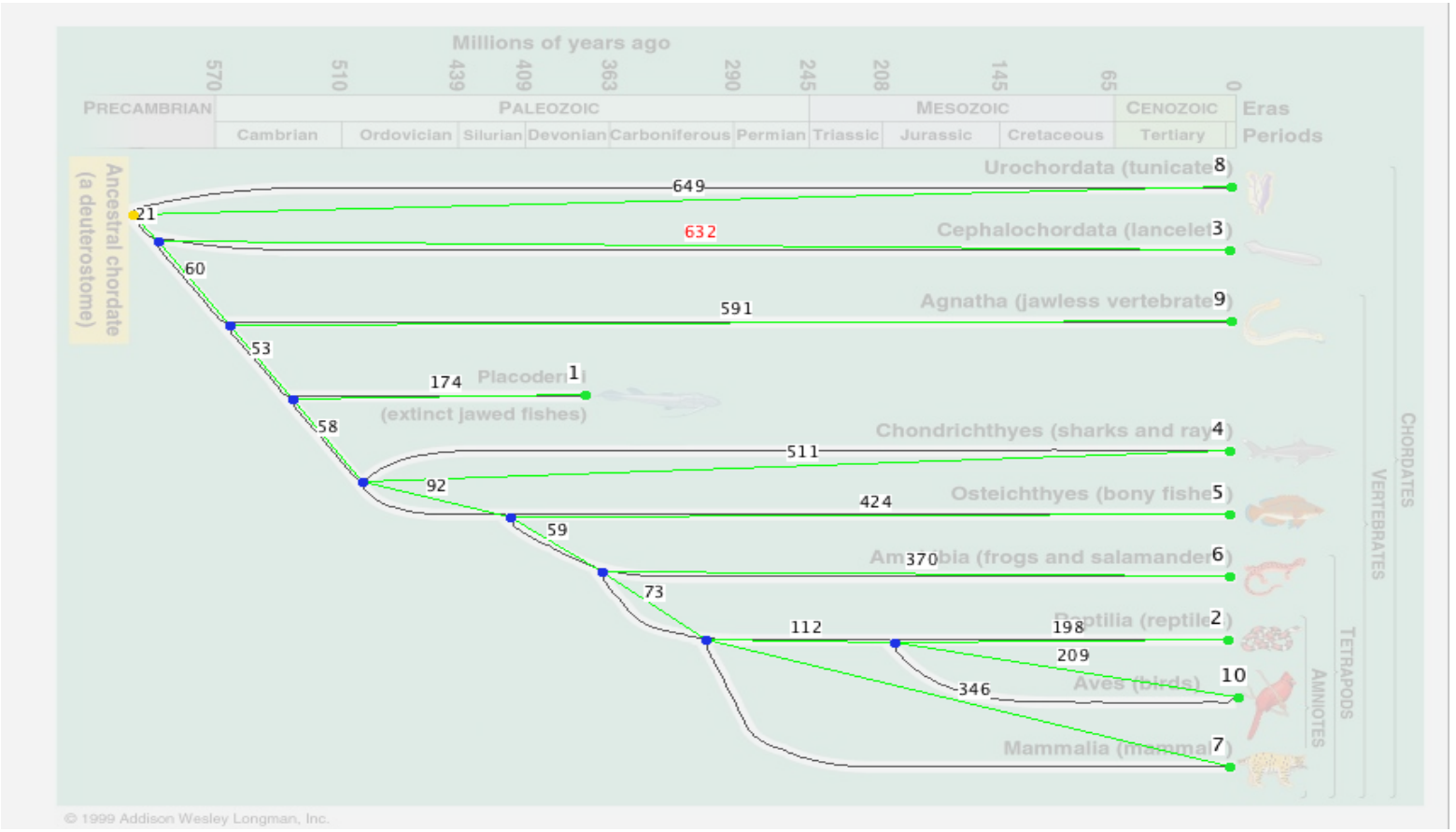


It should be clear by now that you can only place nodes on foreground (binary black) pixels. This is sensible as a branch is defined by means of a pair of nodes linked by a foreground path.

Click **Save Snapshot**. This preserves what you have done so far. Everything apart from the undo buffer is saved. Using **Load Snapshot**, you can restore the state of work later.

Click **Calculate Branches,** then **Toggle Newick View.**
You should be presented the frame with the Newick tree expression for the tree.
If TreeSnatcher Plus informs you that an error occurred, you should search for nodes at illogical positions.

Click **Save image**. In the dialog, select **Source and Current image** and **Branches with lengths**, Check **Inner Nodes and tips** and **Species Names**. Press **Done**. Select a file. You have now saved an image that contains separate layers of information.



This is the resulting Newick representation. The default taxon names correspond to those in the image.

```
(8:649, (3:632, (9:591, (1:174, (4:511, (5:424, (6:370, (7:346, (2:198,
10:209):112):73):59):92):58):53):60):21);
```