CT manager

Script „Image-manual knots organizer“

The objective is to create from the CT manual output (.txt files) two new databases: one with various measurements per knot (multiple rows per knot), and one with only one measurement per knot (one row per knot). In the input file you will see two rows in the beginning. The first row represents the identification of a tree and log, and the second one contains the total number of knots from this log. The following rows are always a sequence of four rows. Every group of four rows represents measurements from a unique knot. Knowing that, do the following:

1. Import the text (.txt) file.

2. Join the first and second cells from the first row, to have the complete “name” of the file.

3. From the joint name, extract the tree ID (always a 3-digit number, ex: 623, 716, 751, 790, 821 and 903), and the log ID (always after the tree ID and an underline symbol, ex: 1, 2, 4, 5, 6, 7, 8, 9, 10).

4. Split the file in groups of four rows, beginning from the 3rd row, so the first group will be all columns from row 3 to row 6 (including also rows 3 and 6). Stop when you find a row with only blank cells.

5. Create three variables (columns) of identification:
    a. The first one called “Tree” with the tree ID extracted in step 3.
    b. The second one called “Log” with the log ID extracted in step 3.
    c. The third one called “Knot” with the knot ID, which is composed by sequential numbers to differ between each other. Begin from 1 onwards. Do not reset this sequence, meaning that independently of the file imported; it should always increase, not allowing two knots with the same Knot ID.

For every group of four rows, do the following:

Database 1: “Multi_imgManual”

6. For every column in the 2nd row different from 0, attribute fixed sequential values beginning with 20 in intervals of 20, so the series will be 20, 40, 60, 80… in a new paired variable called “RadialPosition_mm”. To be clear: the total number of rows you should produce in your database for each knot is equal to the number of columns in the 2nd row that are different from 0. Once you have this number, apply the fixed sequence and fill the cells.

7. Create a new variable called “DiameterV_mm” and store in it the values of the 3rd row, so the values formerly organized in a row, will now be stored in a column.

8. Create a new variable called “DiameterH_mm” and store in it the values of the 2nd row, so the values formerly organized in a row, will now be stored in a column.

9. Create a new variable called “MeanDiameter_mm” and store in it the mean between “DiameterV_mm” and “DiameterH_mm” for each row.
10. Sum the value of the third cell in the first row to every cell different from zero in the 4th row. Take the new values of the 4th row and store in the database under a new variable called “Zposition_mm”.

11. Create a new variable called “Sound_pos” based on:
    If the “RadialPosition_mm” is greater than the value of the 4th column of the 1st row (input file), return 0. Else return 1. Do this for all rows.

12. **FIRST PRODUCT:** Save database 1 as “Multi_imgMan” in the first tab of an .xls file called “Manual_KnotData”.

Database 2: “One_imgManual”

13. Create the identification variables, as described in step 5.

14. Extract the value of the 4th cell in the first row and store it under the variable called “DKB_mm”.
    If the 5th column of the 1st row is different from -1, store the value of the 4th column in the first row this cell. Else return 0.

15. Create a new variable called “KE_mm” and store in it the following value:
    If the 5th column of the 1st row is different from -1, store the value of this cell. Else store the value of the 4th column in the first row.

16. Extract the value of the 7th cell in the first row and store it under the variable called “Azimuth_deg”.

17. Create a new variable called “Sound_knot”, which stands for sound/dead status, based on:
    If the value in the 5th column of the first row is equal to -1, return 1. Else return 0. Do this for all knots.

18. **SECOND PRODUCT:** Save database 2 as “One_imgMan” in the second tab of the .xls file created in step 10.

---

2 Script „CT-automatic knots organizer“

The objective here is to create from the CT automatic output (.csv files) two new databases and one table: one database with various measurements per knot (multiple rows per knot), one database with only one measurement per knot (one row per knot) and a table with results of a checking process. In this input file, each row represents a unique knot, and each column is a parameter for that specific knot that will be used to calculate other variables for that particular knot.

1. Import the knotsParametric@Dgl_LogID.csv file.

2. From the **FILE NAME**, extract the tree ID (always a 3-digit number after “Dgl_”, ex: 623, 716, 751, 790, 821 and 903), and the log ID (always after the tree ID and an underline symbol, ex: 1, 2, 4, 5, 6, 7, 8, 9, 10).

3. Count the number of rows in the file. Perform the following checking process: If there is a row with a zero in any of the 5th, 6th or 7th columns, exclude this row. Count all remaining rows. Based on this checking process, generate a table according to step 4.

4. **FIRST PRODUCT:** Provide a table with: number of rows in total, number of excluded rows, number of remaining rows, and a percentage of usable rows (remaining rows/total rows*100). Each row represents a knot, so please call these output parameters respectively: “Knots in total”, “Excluded knots”, “Remaining knots”, “Percentage of usable knots”.

5. Create three variables (columns) of identification:
   a. The first one called “Tree” with the tree ID extracted in step 2.
   b. The second one called “Log” with the log ID extracted in step 2.
   c. The third one called “Knot” with the knot ID, which is composed by sequential numbers to differ between each other. Begin from 1 onwards. Do not reset this sequence, meaning that independently of the file imported; it should always increase, not allowing two knots with the same Knot ID.

For every row, do the following:

Database 1: “Multi_CT”

6. Create a variable called “RadialPosition_mm” and attribute to it fixed sequential values beginning with 20 in intervals of 20, so the series will be 20, 40, 60, 80… until the limit established by the value on the 8th column for each knot.

7. For each “RadialPosition_mm” value, calculate the variable “Diameter_mm”. The calculation is as follows:

$$D_i = abs\{\tan[A_i + (B_i * \sqrt[4]{R_i})] * 2R_i\}$$

D_i = “Diameter_mm” for the row i
abs = return the absolute value
tan = trigonometric function: tangent
A_i = value in the 1st column for the row i
B_i = value in the 2nd column for the row i
R_i = “RadialPosition_mm” value established in step 3, for the row i

8. For each “RadialPosition_mm” value, calculate the variable “Zposition_mm”. The calculation is as follows:

$$Z_i = E_i + \left(F_i * \sqrt{R_i} + G_i * R_i\right)$$

Z_i = “Zposition_mm” for the row i
E_i = value in the 5th column for the row i
F_i = value in the 6th column for the row i
G_i = value in the 7th column for the row i
R_i = “RadialPosition_mm” value established in step 3, for the row i

9. For each “RadialPosition_mm” value, calculate the variable “Azimuth_deg”. The calculation is as follows:

$$A_i = \frac{(C_i + D_i * \log(R_i)) * 360}{2 * \pi}$$

A_i = "Azimuth_deg" for the row i
C_i = value in the 3th column for the row i
D_i = value in the 4th column for the row i
log = base 10 logarithm mathematical function
R_i = "RadialPosition_mm" value established in step 3, for the row i

10. Create a new variable called "Sound_pos" based on:
If "RadialPosition_mm" is greater or equal to the 9th column for the row i, return 0. Else return 1. Do this for all rows. Remember: the row i in the input file will be the same for all the rows representing the same knot in the output database.

11. Import the barkPolar@Dgl_LogID.csv, borderPolar@Dgl_LogID.csv and sapPolar@Dgl_LogID.csv files (comma is the column separator). These files measure different boundaries within a slice (see Figure 1). These files are organized in the following way: there are always a total of 360 columns, which represent azimuth (or angular) positions, meaning that we have data for every degree within 360 degrees. Each row represents 5 mm of the longitudinal position of a log. So, by multiplying the number of a row by 5, we obtain the longitudinal position in mm within a log. The total number of rows may vary between logs, since each log may have a total length between 3 and 5 meters. Finally, the number that appears in every cell is the radial distance between the pith and the threshold for a region within the stem (colored lines in Fig.1, depending on the file) at a specific angular position (column) and at a specific height (row).

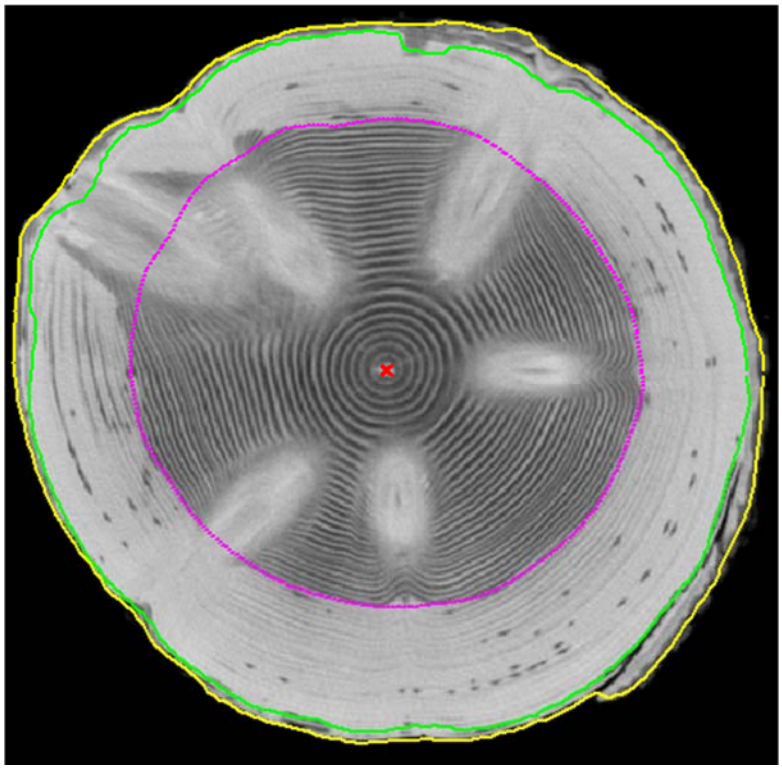

Figure 1. CT-slice of a Douglas-fir log. The red X in the middle is the pith location, the yellow line is the outer border of the log (borderPolar file), the green line is the wood-bark border (barkPolar file) and the pink line is the heartwood-sapwood border (sapPolar file).

12. Create a new variable (column) called "HSlimit_mm", in which you will store the pink boundary value (Fig.1) at each knot position, as follows:
    - For each row in your database (corresponding to a knot position) you will take the value of "Zposition_mm" and round it to the nearest round number (without decimals). Based on this value you should be able to find the correct **row** in the "**sapPolar**" file. The correct row is the one that represents the same longitudinal

position (Zposition_mm) in mm (so you should multiply the number of the row by 5, so you have the longitudinal position instead of the sequential number of the slice).
- For each row in your database (corresponding to a knot position) you will take the value of "Azimuth_deg" and round it to the nearest round number (without decimals). Based on this value you should be able to find the correct **column** in the "**sapPolar**" file. The correct column is the one that represents the same azimuth (angular) position (Azimuth_deg) in degrees (one of the 360 columns should be chosen).
- If you followed the last two steps you have the coordinates (a column and a row) to a specific cell. Store the value of this cell under the "HSlimit_mm" variable, in the correspondent knot position.

13. Create a new variable (column) called "WBlimit_mm", in which you will store the green boundary value (Fig.1) at each knot position, as follows:
    - For each row in your database (corresponding to a knot position) you will take the value of "Zposition_mm" and round it to the nearest round number (without decimals). Based on this value you should be able to find the correct **row** in the "**barkPolar**" file. The correct row is the one that represents the same longitudinal position (Zposition_mm) in mm (so you should multiply the number of the row by 5, so you have the longitudinal position instead of the sequential number of the slice).
    - For each row in your database (corresponding to a knot position) you will take the value of "Azimuth_deg" and round it to the nearest round number (without decimals). Based on this value you should be able to find the correct **column** in the "**barkPolar**" file. The correct column is the one that represents the same azimuth (or angular) position (Azimuth_deg) in degrees (one of the 360 columns should be chosen).
    - If you followed the last two steps you have the coordinates (a column and a row) to a specific cell. Store the value of this cell under the "WBlimit_mm" variable, in the correspondent knot position.

14. Create a new variable (column) called "Outlimit_mm", in which you will store the pink boundary value (Fig.1) at each knot position, as follows:
    - For each row in your database (corresponding to a knot position) you will take the value of "Zposition_mm" and round it to the nearest round number (without decimals). Based on this value you should be able to find the correct **row** in the "**borderPolar**" file. The correct row is the one that represents the same longitudinal position (Zposition_mm) in mm (so you should multiply the number of the row by 5, so you have the longitudinal position instead of the sequential number of the slice).
    - For each row in your database (corresponding to a knot position) you will take the value of "Azimuth_deg" and round it to the nearest round number (without decimals). Based on this value you should be able to find the correct **column** in the "**borderPolar**" file. The correct column is the one that represents the same azimuth (angular) position (Azimuth_deg) in degrees (one of the 360 columns should be chosen).
    - If you followed the last two steps you have the coordinates (a column and a row) to a specific cell. Store the value of this cell under the "Outlimit_mm" variable, in the correspondent knot position.

15. Create a new variable called "Heartwood_point", in which you will store the result of the following for each row: If the "RadialPosition_mm" is greater than "HSlimit_mm", return 0. Else return 1.

16. **SECOND PRODUCT:** Save database 1 as "Multi_CT" in the first tab of a .xls file called "CT_KnotData".

Note: The variables will always have the same number of measurements within a knot. So, for example, if "Diameter_mm" has only 4 measurements in a given knot, the "Zposition_mm" will also have 4 values.

Database 2: "One_CT"

17. Based on the file "KnotParametrics.csv", create the three identification variables, as described in step 5.

18. Calculate the azimuth of the knot, following the formula in step 10, using the maximum value of "RadialPosition_mm" for each knot as input, and store it under the variable called "Azimuth_deg".

19. Extract the value of the 9th column and store it under the variable called "DKB_mm".

20. Extract the value of the 8th column and store it under the variable called "KE_mm".

21. Create a new variable called “Sound_knot” based on:
    If the value in “KE_mm” is larger than “DKB_mm”, return 0. Else return 1. Do this for all knots.

22. Create a new variable called “C_mm” which will be the distance between the beginning of the log until the starting point of each knot. This value is directly given by the algorithm and is stored in the 5th column of the input file “KnotsParametric.csv”. Do this for all knots.

23. Calculate the three variables “HSlimit_mm”, “WBlimit_mm” and “Outlimit_mm”, using only the row containing the maximum “RadialPosition_mm” per knot. So, instead of generating several values per knot, you will now search the value correspondent to the maximum radial position.

24. **THIRD PRODUCT:** Save database 2 as “One_CT” in the second tab of the .xls file generated in step 13.