**Allometric**

URL: [GitHub - brycefrank/allometric: A native R package of allometric equations for trees.](https://github.com/brycefrank/allometric)

Date: January 2021

Author:

Description: allometric is an R package for predicting tree attributes with allometric models. Thousands of allometric models exist in the scientific and technical forestry literature, and allometric is a platform for archiving and using this vast array of models in a robust and structured format.

Allometric not only enables the use of allometric models for analysis, it also provides a structured language for adding models to the package. If you are interested in helping the developer in this process please refer to the Installing a Model vignette.

In total, allometric contains 1590 models from 38 publications. More details about this tool can be found on the URL link provided above.

Installation:

1. You need to install devtools and rtools for successful installation of this tool, however, just with the devtools it can work fine, if you have some issues with rtools installation. Then, run the code below.

devtools::install\_github("brycefrank/allometric")

Getting started:

Check on the proper installation of the models by running the following code:

library(allometric)

install\_models()

after running this install\_models() just check with the following code for more help information to know more about the considered parameters for all the allometric models kept in this tool, based on this you can easily choose the model which you need to work with:

head(allometric\_models)

Finding and selecting models:

allometric\_models is a dataframe containing all the models. To choose model based on your requirements, you need to run a query and then unnest the model using following code:

unnested\_models <- unnest\_models(allometric\_models, cols = "family\_name")

unnested\_models

In this the model is unnested based on the column “family\_name”, so you can also choose other columns.

after unnesting, you can filter the model using dplyr::filter function. For example:

brackett\_alnus\_vol <- unnested\_models %>%

 dplyr::filter(

 family\_name == "Brackett", model\_type == "stem volume",

 genus == "Alnus"

 )

brackett\_alnus\_vol

In this code, the focus is to find a volume model for the genus Alnus, "Brackett" is an author or co-author.

**Select the model using id**:

brackett\_alnus\_mod <- brackett\_alnus\_vol %>% select\_model("f21028ef")

or by row Index:

brackett\_alnus\_mod <- brackett\_alnus\_vol %>% select\_model(1)

brackett\_alnus\_mod will be used for the prediction

if you need to know the input parameters or more information how to fit the model, type following line:

brackett\_alnus\_mod

**Predict using the Selected Model:**

dsob [in]: diameter of the stem, outside bark at breast height

hst [ft]: total height of the stem

predict(brackett\_alnus)mod, dsob, hst)

for example:

predict(brackett\_alnus\_mod, 12, 65)

#> 22.2347 [ft^3]

We can also use values from our dataframe, you need to give input of your dataframe. First make a dataframe of your data. for example:

my\_trees <- data.frame(dias = c(12, 15, 20), heights = c(65, 75, 100))

predict(brackett\_alnus\_mod, my\_trees$dias, my\_trees$heights)

or even you can use the dplyr, do not forget to install it first using the code:

install.packages(“dplyr”)

library(dplyr)

my\_trees %>%

 mutate(vols = predict(brackett\_alnus\_mod, dias, heights))

here mutate function adds new variables that are functions of existing variables. Forexample, running the above code would look like:

#> dias heights vols

#> 1 12 65 22.23470 [ft^3]

#> 2 15 75 39.80216 [ft^3]

#> 3 20 100 94.20053 [ft^3]

So, it added a new column of a new variable named **vols**