The R Mensuration Package User's Guide

J. H. Gove
USDA Forest Service, Northern Research Station, 271 Mast Road, Durham, NH 03824 USA
(603) 868-7667; e-mail: jgove@fs.fed.us

Tuesday 9th October, 2018
11:30am

Contents

1 Introduction

This is a user's guide with some examples of use and a little "theory" background for the `Mensuration` package in R.

Begin by loading the package (alternatively, the `library` method may also be used)...

```
R> require(Mensuration)
```

Once this is loaded you can call up the overall package help at the R command prompt with `package?Mensuration`. Each routine in the package is described and demonstrated in the following sections. For more information on individual functions than is presented here (i.e., what each argument in the function definition means), please see the individual help files by using the `?functionName` syntax at the R prompt.

In most of the examples below, we have used the `args` function in R to show what the arguments are for a particular function before its first use. This saves having to type the arguments explicitly

when they are in order of appearance in the function definition; if they are not in order, then you must also include the argument names so that R can match them correctly.

The R routines described and illustrated in § 2 comprise a few very simple functions for easily building stand tables. These methods are put to use in the following two sections. First, § 3 describes a routine that can be used to find the value of q for a stand when the basal area and maximum DBH are known—the so-called BDq method. All of the routines are then applied in a very user-friendly interface that facilitates the study of q distributions in an interactive manner. A little background "theory" is provided for those readers that are so inclined—and may be skipped as desired. The following section, § 4, presents a sketch of the Chapman-Richards distribution and provides another interactive R routine that can be used to explore the relationship between stand growth parameters and the resulting diameter distribution under this model. Lastly, a useful little utility that facilitates unit conversions from within R is described.[1]

2 Handy Stand Table Routines

The following routines may make handling diameter classes and building very simple stand tables a bit easier.

2.1 Routine: `dbhClassLimits`

This routine will simply generate a data frame containing the lower, mid, and upper DBH class limits based on the arguments passed (diameter class width, etc.). It is important to note that the first two arguments are the *midpoint* diameters for the smallest and largest class. The respective lower and upper limits are then calculated based on these and the DBH class width.

Let's look at a couple simple examples...

```
R> args(dbhClassLimits)


function (dbh.low = 1, dbh.up = 20, dbh.width = 1, forceIntegerWidth = TRUE,
    adjustLowest = NULL, classNames = c("dbh.low", "dbh.mid",
        "dbh.hi"), runQuiet = TRUE, ...)
NULL


R> (dlims = dbhClassLimits(1, 5.23, 1))
```

[1] Since writing this, a very useful package, **udunits2**, has arrived in CRAN that provides a good deal of flexibility for unit conversions within R.

```
  dbh.low dbh.mid dbh.hi
1     0.5       1    1.5
2     1.5       2    2.5
3     2.5       3    3.5
4     3.5       4    4.5
5     4.5       5    5.5


R> (dbhClassLimits(2, 10, 2, adjustLowest=1.0))


  dbh.low dbh.mid dbh.hi
1       0       2      3
2       3       4      5
3       5       6      7
4       7       8      9
5       9      10     11
```

The first example gives us the bounds and midpoint diameters for one-inch classes as we could expect. If you want the lowest boundary to go down to zero (thus the lowest class is lop-sided/wider on the low end) to get all size trees, then use the `adjustLowest` argument as in the second example.

One other feature that should get very little use (unless one, for example, might want to work with metric equivalents) is the `forceIntegerWidths` argument. When this is set to `TRUE`, the routine forces the class *widths* to be whole numbers; this is the default as shown above. If you request it to be `FALSE`, then you can also request bounds that are real valued. For example...

```
R> (dbhClassLimits(2.54, 10, 2.54, force=FALSE))


  dbh.low dbh.mid dbh.hi
1    1.27    2.54   3.81
2    3.81    5.08   6.35
3    6.35    7.62   8.89
```

The above example shows how we could get metric limits in cm corresponding to one-inch classes. Note that the upper midpoint of 10cm is not large enough to include the true midpoint, so the routine returns results to the next lowest class (try setting it to, e.g., 11 instead of 10 and see what happens).

2.2 Routine: groupDBH

This function takes a list of raw diameters and a set of limits from **dbhClassLimits** and then groups the raw diameters into these classes. It really is not meant to be used as a general function, it is called from **standTable**, for example, to do most of the dirty work. An example to illustrate its use follows, but also see the examples in the help file for more details.

```
R> args(groupDBH)


function (dbh, classLimits)
NULL


R> set.seed(1324)           #for replicability
R> dbh = rexp(20, .45)      #some diameters
R> #dlims = dbhClassLimits() #default limits
R> gd = groupDBH(dbh, dlims)


Warning in groupDBH(dbh, dlims):  Some diameters fell outside the class limits passed;
these have been discarded.


R> print(rbind(gd$dbh.grp, dbh), digits=3)


    [,1]  [,2]   [,3]  [,4] [,5]  [,6]  [,7] [,8] [,9] [,10] [,11] [,12] [,13]
    5.00    NA     NA 1.000   NA 1.000    NA   NA   NA  2.00    NA  1.00  3.00
dbh 5.01 0.452 0.0891 0.607 9.54 0.933 0.417 0.47 6.48  2.12 0.459  1.26  3.48
    [,14] [,15] [,16] [,17] [,18] [,19] [,20]
     1.00  2.00  1.00  1.00  5.00 1.000 1.000
dbh  1.43  1.65  1.37  1.34  5.19 0.562 0.626
```

Note that we just drew some random diameters from an exponential distribution. Some of them lie outside the extent of the class limits that we determined in the first example of § 2.1, so a warning was issued to this affect and these are set to **NA**. Examine the output above and the limits given in **dlims** to see that the raw diameters were classified correctly.

2.3 Routine: `standTable`

This routine combines the above two functions in the sense that it takes a vector of raw diameters and arguments specifying the diameter class limits (to be passed on to `dbhClassLimits` through the `...` argument) and builds a stand table with number of trees by DBH class and associated basal area. Both English and metric units are supported.

For example, we draw some random diameters from an exponential distribution again, and then build the stand table with two-inch classes...

```
R> args(standTable)


function (dbh, wantBA = TRUE, isEnglish = TRUE, runQuiet = TRUE,
    ...)
NULL


R> set.seed(1282)
R> dbh = rexp(100, 0.6)          #100 dbhes
R> print(range(dbh), digits=3)


[1] 0.00791 7.18993


R> st = standTable(dbh, dbh.low = 1, dbh.up = max(dbh), dbh.width = 2)
R> print(st, digits=3)


  dbh.low dbh.mid dbh.hi trees    ba
1       0       1      2    69 0.270
2       2       3      4    23 1.123
3       4       5      6     6 0.901
4       6       7      8     2 0.553
```

Setting `dbh.up` to the maximum DBH just ensures that all diameters are included in the table.

3 "q" Distribution Routines

The package has some facility for working with the q distribution. This distribution was evidently first developed by Meyer (1952) based on earlier work (Meyer and Stevenson, 1943) and, contrary

to what has been stated in texts and the literature, is not due to de Liocourt (1898) as recently demonstrated by Kerr (2014). Please consult the help files on the routines for documentation on arguments, return values, and other examples of usage.

3.1 Routine: `findq`

This routine will find the q-value for a stand with a desired level of basal area and number of trees in the maximum DBH class. There are probably simpler ways to do this, but here the idea behind what the `findq` method does.

Let $N_{d_{\max}}$ be the desired target number of trees in the largest DBH class, with $i = 1, \ldots, n_d$ DBH classes beginning from the 1-inch class (see Davis and Johnson, 1987, p. 61 for another version); then

$$\begin{aligned} N &= N_{d_{\max}} + N_{d_{\max}} q + N_{d_{\max}} q^2 + \ldots + N_{d_{\max}} q^{n_d - 1} \\ &= N_{d_{\max}} \sum_{i=1}^{n_d} q^{i-1} \end{aligned} \tag{1}$$

and therefore

$$B = N_{d_{\max}} \sum_{i=1}^{n_d} b_i q^{i-1} \tag{2}$$

where the b_i are the midpoint basal areas for the ith class. To find the q-value corresponding to a given level of basal area, simply re-arrange (2) and solve with a root finder (i.e., `uniroot`); *viz.*,

$$\frac{B}{N_{d_{\max}}} - \sum_{i=1}^{n_d} b_i q^{i-1} = 0$$

This is demonstrated in the following snippet...

```
R> dcWidth = 1
R> args(findq)


function (dbh.low = 0.5, dbh.up = 19.5, dc.width = 1, B = 80,
    N.max = 2, units = c("English", "metric"), runQuiet = FALSE,
    ...)
NULL


R> fq = findq(dc.width = dcWidth)
```

```
Results...
------------------
Minimum dbh = 0.5
Maximum dbh = 19.5
Basal area/acre = 80
Trees in the 19.5 inch class = 2
1-inch q value = 1.18
...objective at q root =  -0.0010083258
...iterations = 14
...approximate precision = 6.1035156e-05
------------------


R> (q = fq$qValue)


[1] 1.1839539


R> qTable = fq$qTable
R> print(qTable[1:5,], digits=2)


  dbh.low dbh.mid dbh.hi  N     B
1       0     0.5      1 49 0.067
2       1     1.5      2 42 0.513
3       2     2.5      3 35 1.203
4       3     3.5      4 30 1.992
5       4     4.5      5 25 2.781
```

The output above shows the q-value solution and shows the first few rows of the stand table for the desired stand. This routine is used, for example, in the `qDistnShiny` application discussed in § 3.2.

3.2 Routine: `qDistnShiny`

This began as a simple experiment to learn the Shiny (`http://shiny.rstudio.com/`) interface before coding a more complicated project (see § 4.1). It turns out to be somewhat useful in its own right in looking at different q structures, so it was put into this package for others who might want to use it. You must be sure to have installed the `Shiny` package on your system, it is simple to do this within **R** via, e.g.,

```
R> install.packages('shiny')
```

You need install this packages only once. Then it is a simple matter to run the application as...

```
R> qDistnShiny()
```

The application runs in your web browser. Whenever possible, please try to exit the application using the exit button in the web page, this will make sure everything is closed up correctly. If you get in a bind (like some error) then use either escape or control-c back at your **R** command-line session to stop the application. Please note that your **R** command-line session will be unavailable until you exit the application (see `?qDistnShiny` for help). English[2] units are the default, but metric can be displayed quite easily using `qDistnShiny('metric')` to start the routine.

An optional package can also be loaded that enables JavaScript `DataTables`[3], which are much nicer and more flexible than the default `html` tables. To enable `DataTables` you also need to install the **R** `DT` pacakge; *viz.*,

```
R> install.packages('DT')
```

Once this is done, the `DataTables` become the default display, but one can request the `html` table, if desired, by starting `qDistnShiny` with the optional argument `useDT=FALSE`.

3.2.1 Mathematical development

This section can be skipped if desired, though it is not difficult—just go to § 3.2.2 to skip to an example of `qDistnShiny`. All that it does is document how the continuous densities that are overlayed on top of the bar charts are calculated for those that are interested in the detials.

We begin with Meyer and Stevenson's original scaled exponential formulation. When I say scaled exponential, this comes automatically from the regression setting so it is a "numbers density" rather than a probability density: the former integrates to N and the latter to one, over the support of the distribution. See Appendix A for more information on the relation of Meyer and Stevenson's model to q (the quotient q can produce either a numbers or probability density, depending on how it is used; foresters normally just like to work in terms of the numbers density—the stand table).

Now, what has prompted the following is the fact that the `qDistnShiny` function cuts off the full negative exponential distribution (i.e., truncates it), but we do not normally think of it in those

[2] To be precise, U. S. Customary units, as developed from the old "English" units (`https://en.wikipedia.org/wiki/United_States_customary_units`).

[3] See `http://rstudio.github.io/DT/` for more information.

terms when working with q, because we are thinking in terms of BDq, and the total number of trees in a stand is reflected in this triplet (basal area, tree of maximum diameter, q). But when one needs to overlay a full and correctly scaled numbers or probability density onto a bar chart or histogram, we must be able to back out the total number of trees that would be in an imaginary stand under the exponential, were trees allowed to get infinitely large in diameter.[4]

Let's begin with the numbers density as given in (A.2); the number of trees in the ith DBH class $[d_{l_i}, d_{u_i}]$ with midpoint d_i is...

$$\begin{aligned} N_i &= K \int_{d_{l_i}}^{d_{u_i}} \exp\left(-\frac{\ln q}{w} d\right) \mathrm{d}d \\ N_i &\approx K \exp\left(-\frac{\ln q}{w} d_i\right) \\ &= K \exp\left(-a\, d_i\right) \end{aligned} \tag{3}$$

with...

$$a = \frac{\ln q}{w} \tag{4}$$

and where w is the diameter class width, K is the scale factor, and N_i is the number of trees in the class. This is Meyer and Stevenson's regression formulation. Note that in a BDq application, we know everything in equation (3) but the scale factor, K. This is because N_i is calculated via the creation of the q-distribution for all DBH classes $i = 1, \ldots, n_d$ by knowing BDq (i.e, via (1)). So all we have to do is invert (3) to find the scale factor; *viz.*,

$$K = \frac{N_i}{\exp\left(-a\, d_i\right)} \tag{5}$$

Note that this works best when the diameter class size is small, say one-inch classes. The fact that it works at all is due to the q-based proportionality in the N_i between DBH classes.

We can look at this in R using the results from `findq` above; e.g.,

```
R> K = with(qTable, N[1]/exp(-log(q) * dbh.mid[1]) )
R> K


[1] 53.83426
```

Aside: R's `with` statement is a very helpful language statement that simply creates a temporary new "environment" within R to hold the components of a `list` object—in this particular case, that

[4] An alternative would be to use the truncated distribution, but this weights the probability or numbers density differently so it is not used here.

translates to the columns of the `qTable` data frame—such that we can now refer to them simply by their names, and not with the sometimes cumbersome $ (e.g., `qTable$N`) notation.[5]

Now we can use this information to make a numbers density plot which is very similar to what is found in the `qDistnShiny` application. For example,

```
R> pdf(file.path(getwd(), 'figures', 'qPlot.pdf'))
R> barplot(qTable$N, dcWidth, 0.0, col='grey90',
+         names.arg = qTable$dbh.mid, cex.lab=1.6, cex.axis=1.6,
+         cex.names=1.6, xlab='DBH', ylab='Number of trees'
+        )
R> ndcl = nrow(qTable)
R> dd = with(qTable, seq(dbh.mid[1], dbh.hi[ndcl], length.out=100))
R> lines(dd, K*exp(-log(q)*dd), col='blue', lty='dashed', lwd=1.4)
R> dev.off()


pdf
  2
```

The result is presented in Figure 1a.

Now we consider the basal area-size distribution (BASD) representation that corresponds to the q-distribution in Figure 1a. We begin with determining the total population size, N, including all trees in the stand larger than the $d_{\max}$ limit; this will eventually allow us to determine the total basal area over all size classes. We know that[6]

$$\begin{aligned} N &= K \int_0^{\infty} \exp\left(-a\, d\right)\, \mathrm{d}d \\ &= \frac{K}{a} \end{aligned} \tag{6}$$

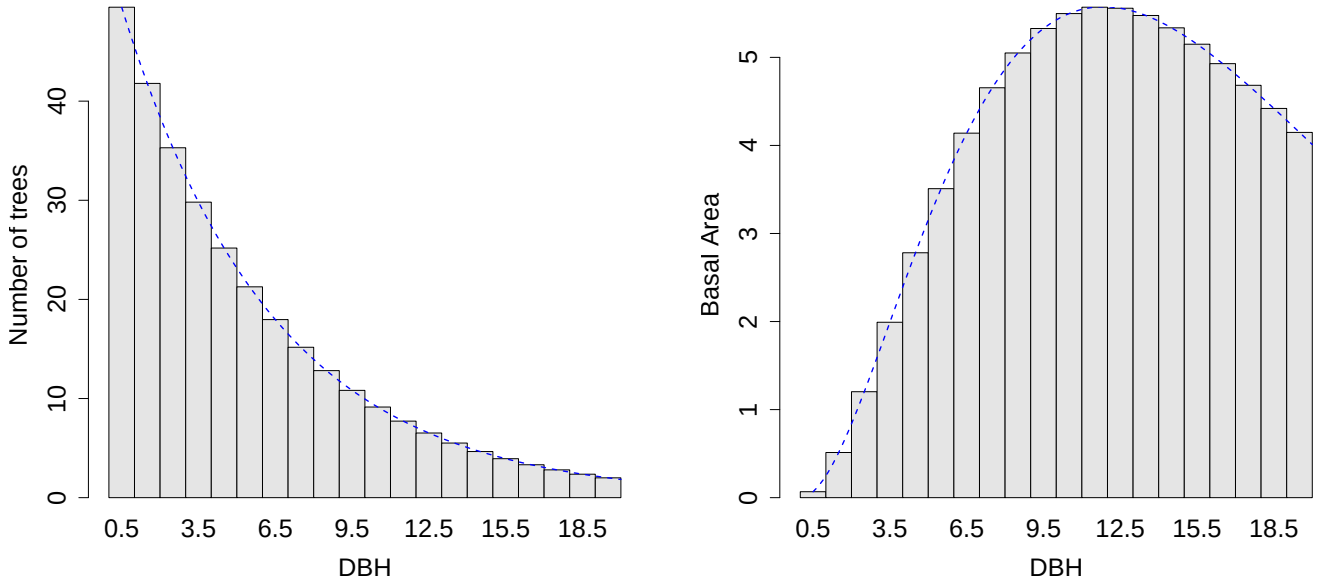
which works out to the usual scaled exponential PDF if we substitute $K = Na$; i.e.,

$$N_i = Na \int_{d_{l_i}}^{d_{u_i}} \exp\left(-a\, d\right)\, \mathrm{d}d$$

Knowing this quantity allows us to find the total basal area, B, over all diameters (including those larger than $d_{\max}$), which is required to properly scale the basal area-DBH curve when plotted. The

[5] There is much more to `with` than what I have described here, but this is why we use it in such simple circumstances as above; see `?with` for more information.

[6] For those interested, let: $u = -ad$ so that $\mathrm{d}u = -a\,\mathrm{d}d$, then $-\frac{K}{a}\int_0^{\infty}\exp(-a\,d)\,\mathrm{d}d = -\frac{K}{a}\int_0^{\infty}\exp(u)\,\mathrm{d}u = -\frac{K}{a}\left(\exp(-a\cdot\infty) - \exp(a\cdot 0)\right) = -\frac{K}{a}(0-1)$.



(a) Exponential (q) Diameter Distribution

(b) Basal Area-size Distribution

Figure 1: The (a) q distribution (bars) with exponent function (dashed) and (b) q-based basal area-size distribution (bars) with theoretical gamma (dashed).

theory behind this is given in Gove and Patil (1998), but the relevant formula is...

$$\begin{aligned} B &= \bar{D}_q^2 N\kappa \\ &= \mu_2' N\kappa \\ &= \frac{2}{a^2} N\kappa \end{aligned} \tag{7}$$

where $\bar{D}_q^2 = \mu_2' = \frac{2}{a^2}$ is the second raw moment of the exponential distribution and κ is the usual basal area conversion factor. Then, given our exponential numbers density parameterization, it follows that the basal area-DBH distribution is a gamma distribution, the size-biased distribution on diameter of order two: $D_2^* \sim \text{Gamma}(1/a, 3)$ (Gove and Patil, 1998).

Putting this all together, and making sense of the math by translating it into **R** we have...

```
R> pdf(file.path(getwd(), 'figures', 'qBAPlot.pdf'))
R> barplot(qTable$B, dcWidth, 0.0, col='grey90',
+         names.arg = qTable$dbh.mid, cex.lab=1.6, cex.axis=1.6,
+         cex.names=1.6, xlab='DBH', ylab='Basal Area'
```

```
+         )
R> (a = log(q)/dcWidth)        #exponential rate parameter

[1] 0.16885963

R> (N = K/a)                   #All trees over all diameters

[1] 318.81071

R> (kappa = pi/(4*144))        #ba conversion factor

[1] 0.0054541539

R> (B = 2/a^2 * N * kappa)     #total basal area over all diameters

[1] 121.96595

R> basd = B * dgamma(dd, shape=3, scale= 1/a)  #ba-size distribution curve
R> lines(dd, basd, col='blue', lty='dashed', lwd=1.4)
R> dev.off()

pdf
  2
```

The result is presented in Figure 1b, where the bar plot shows the basal area derived from the corresponding q distribution up to the largest class specified and the dashed line is the theoretical gamma distribution overlaid for comparison. Note that the whole diameter distribution over all diameters (not just to the upper limit) carries significantly more basal area (122 $\text{ft}^2\text{ac}^{-1}$) than the actual stand does (80 $\text{ft}^2\text{ac}^{-1}$), which is why we needed to account for all that extra when plotting the curve component to the figure in order for it to scale correctly.

Meyer and Stevenson's numbers density (3) is already scaled to take into consideration all trees over all diameters. But the number of trees in the portion shown in Figure 1a to $d_{\max}$ is 307.6, whereas the total number over the entire range of diameters to d_{∞} is 318.8. Again the latter was required to scale the BASD density curve correctly.

In summary, the above has detailed the theory and calculations that are used within the `qDistnShiny` application and hopefully removed any of the mystery behind what is being done within the application behind the scenes.

3.2.2 An example

As explained above, when the `qDistnShiny` application is run at the R command line prompt, it takes you to your web browser and displays the interface shown in Figure 2. The application is very simple to use. The sliders in the left-hand panel allow one to change the different parameters for the stand and resulting distribution display. Underneath that is a textual summary for the current display. The middle panel displays the stand diameter distribution (top) and the BASD (bottom). The far right panel displays the stand table using the `DataTable` display for added flexibility. The latter can be re-arranged to show a portion of the table (10 classes are displayed in the figure), or all diameter classes at once using the "Show xx entries" drop-down at the top of the panel. One can copy the results from this table and paste them into other programs as desired. One can also "print" the page to a file to save it in a format supported by the browser, or use a program that does screen capture to save the screen or portions thereof to a file.

Note that as one changes the sliders, the figures, table, and textual results get updated immediately. Also note in the figures that the small red "rug" mark delineates the quadratic mean stand diameter. Finally, as mentioned above, it is best to use the "Exit Application" button at the top-right of the display to close the application and clean up making R ready for subsequent work. Returns values from the function are detailed in the help file (`?qDistnShiny`); however, it may be worth mentioning that the stand table for the last setting before exit, as well as the associated stand parameters, can be saved into your workspace if desired as described in the help file.

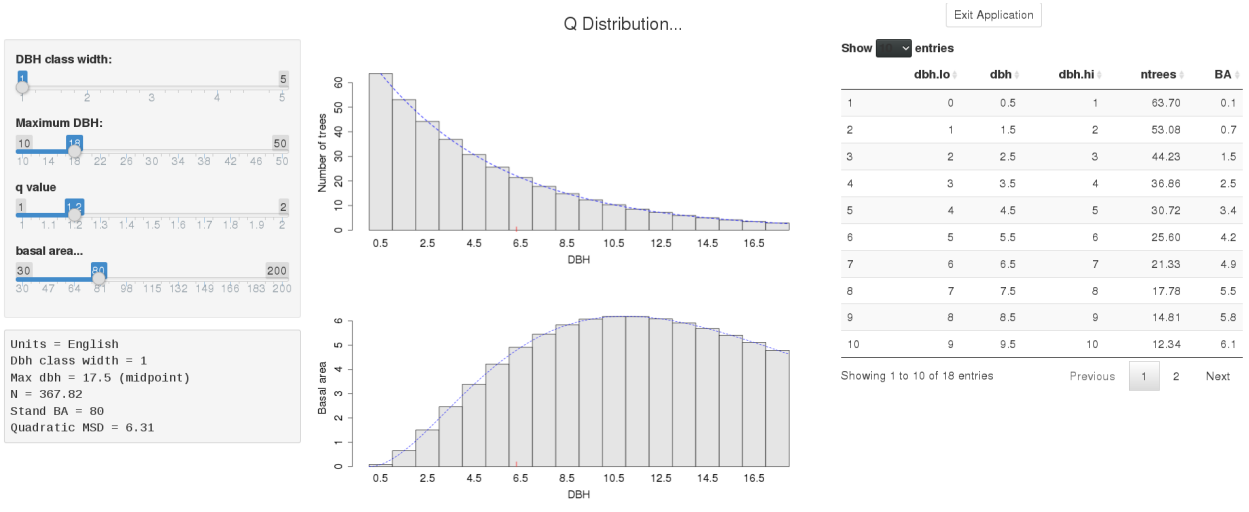


Figure 2: Screenshot from Firefox of the `qDistnShiny` application.

The help file also discusses the different arguments that can be passed to the application. For example, one can shift the DBH distribution left so that it is aligned to include all trees down to zero DBH by using `shiftZero=TRUE` when starting the application. Note that when you use this option, the maximum DBH as given by the slider will be larger than the upper diameter class limit, because the former is always to the nearest integer DBH, and shifting the lower limit to include zero also adjusts the upper class limit down accordingly.

4 The Chapman-Richards Distribution

The Chapman-Richards distribution derives from the basic equilibrium size-structured numbers density (SSD) given by...

$$n(d) = \frac{R}{g(d)} \exp\left(-\int_{d_0}^{d} \frac{m(d')}{g(d')}\,\mathrm{d}d'\right) \qquad d > d_0 \tag{8}$$

where both mortality (M) and recruitment (R) are constants. However, we let growth, $g(d)$, be given by the Chapmpan-Richards (CR) function (Pienaar and Turnbull, 1973)

$$\frac{\mathrm{d}d}{\mathrm{d}t} = g(d) = \eta d^m - \gamma d \qquad 0 \leq d < d_\infty \tag{9}$$

where m, η, and γ are model parameters to be estimated from the data. The asymptotic maximum DBH (d_∞) is found by setting (9) to zero and solving:

$$\begin{aligned} \eta d^m &= \gamma d \\ {d_\infty}^{m-1} &= \frac{\gamma}{\eta} \\ d_\infty &= \left(\frac{\gamma}{\eta}\right)^{\frac{1}{m-1}} \end{aligned} \tag{10}$$

This is the same as what Pienaar and Turnbull (1973, p. 5) found, though they looked at it as the asymptotic limit to volume (rather than diameter) and approached it from the volume yield function in time (rather than diameter).

Gove (2017) showed that the numbers density (8) has closed-form under these vital rate assumptions. That is, the equilibrium numbers density solution to (8) with constant recruitment R, constant mortality M, and CR growth (9) is given as

$$n(d) = \frac{R}{\eta d^m - \gamma d}\left(\frac{d^{m-1}\left(\eta {d_0}^{m-1} - \gamma\right)}{{d_0}^{m-1}\left(\eta d^{m-1} - \gamma\right)}\right)^{\frac{M}{\gamma(m-1)}} \qquad 0 \leq d < d_\infty \tag{11}$$

We will refer to (11) as the Chapman-Richards distribution (CRD). The CRD is very convenient, and avoids numerical integration of the survival function integral in the exponential of (8) (not

really a technical problem now a days), allowing one the ability to manipulate this density to look deeper into relationships perhaps with other densities, The derivation of the above density and its link to the well-known generalize beta distribution (e.g., Ducey and Gove, 2015) are detailed in Gove et al. (2018) and Gove et al. (2019). A Shiny (`http://shiny.rstudio.com/`) application, much like the one shown above for the q distribution has been developed for the CRD and is shown in more detail in § 4.1.

4.1 Routine: `crDistnShiny`

This Shiny application runs in your web browser just like `qDistnShiny`. The details of how to install `shiny` and associated `DT` packages are given in § 3.2; installation only has to be done once for either application. To run the application from the command line, simply type

```
R> crDistnShiny()
```

or add any arguments as detailed in the help page `?crDistnShiny`. The application uses the CR distribution in (11) with sliders that allow you to change the parameter values for vital rates M, R, and CR parameters m, η and γ. The CR growth function and associated distributions are updated as the parameters are changed. An example is shown in Figure 3. The left-hand panel in Figure 3

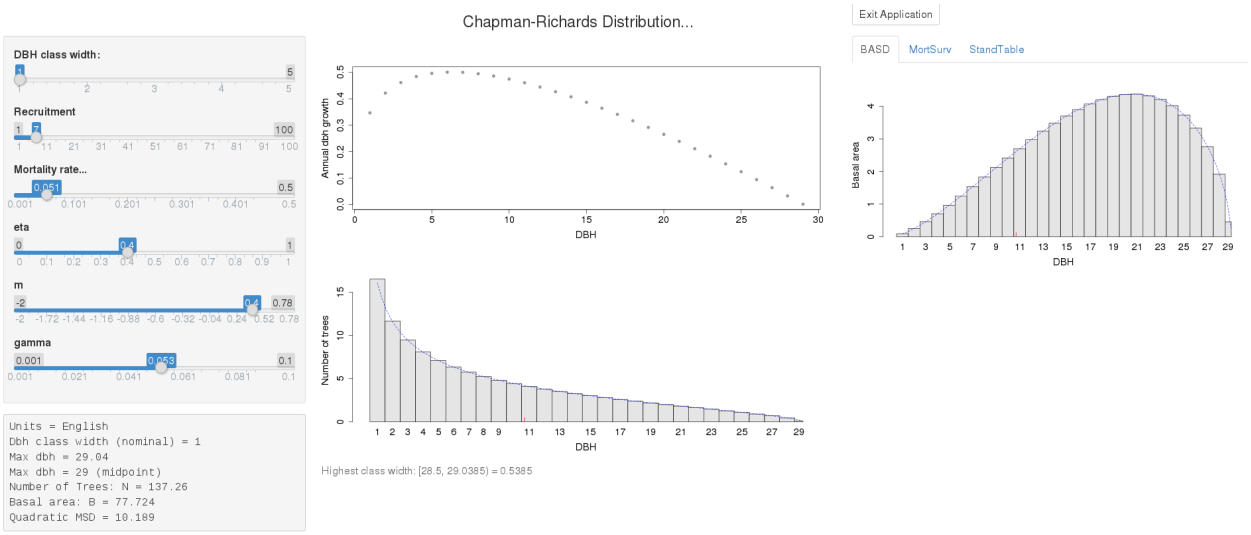


Figure 3: Screenshot from Firefox of the `crDistnShiny` application.

shows the sliders, with summary information on the stand below. The middle panel displays the CR growth model (top) and the CRD (bottom) associated with the settings of the parameters to the left. Finally, the right-most panel allows one to switch between the BASD, which is the default, a graph of the survival function and mortality density function, or a stand table representing the

results of all the graphs in tabular form. The application will display notes below the graphs in the second and third panels as appropriate. These inform the user of, e.g., the size of the upper DBH class limit, which will rarely be the nominal width because of the asymptotic d_∞ limit for the class.

Shiny essentially takes over your R workspace terminal session when you run this application, and the values returned from the normal function call are specifically meant to interact with Shiny; in other words they are not available for returning information from the function call in the normal way. Therefore, like the `qDistnShiny` application, there is an argument that allows saving the results from a run into your `.GlobalEnv` workspace. Setting the argument `returnVar` to a character variable name in the call to `crDistnShiny` will place an R list of results in your workspace. For example, using `returnVar='crStuff'` will create a new R list by the name of `crStuff` with the final results from the run. Returned values contained in the list object from the function are detailed in the help file (`?crDistnShiny`).

Please note that it is possible to request parameter values that cause errors. These will not crash the application, simply change them to the range that no longer throws an error to the screen. Because of the interaction of the parameters it is very difficult to determine different combinations that may cause problems, but it often happens when the CR growth parameters `eta` (η) and/or `m` are low and `gamma` (γ) is large, resulting in only one DBH class that is larger than the entire range for DBH. If you choose to run the application with ranges for the CRD parameters that are different from the defaults (allowing larger ranges), expect problems to happen more frequently. Note that the above problem could be caught, or the default ranges restricted by code, but it is probably better to allow this to happen so one can see how the resulting distribution is sensitive to the different parameter combinations than to overly restrict the results.

4.1.1 The survival and mortality functions

The "MortSurv" tab in the application shows the survival and mortality functions. Details of these with reference to tree diameter distributions can be found in Gove (2017) with specific application to the negative exponential distribution, more details are provided in the references therein. The survival graph (not shown here, run the application to see an example) gives the probability of surviving from d_0 to d. The graph presented has points delineating the upper limit of each DBH class. It is denoted as $S(d)$ in the time-invariant case, and is the exponential term in (8) (e.g., Gove, 2017; Gove et al., 2018, 2019).

The mortality function is a probability density function, $\mathfrak{m}(d)$. Thus the function $\mathfrak{M}(d_l, d_u) = \int_{d_l}^{d_u} \mathfrak{m}(d)\,\mathrm{d}d$, gives the probability of mortality between $[d_l, d_u]$. It is quite straightforward to show in the case of the CRD that the mortality density, $\mathfrak{m}(d)$, and the numbers density $n(d)$ are proportional

and differ only by a scale factor, N; i.e.,

$$n(d) = \frac{R}{g(d)} S(d)$$

$$\mathfrak{m}(d) = \frac{M}{g(d)} S(d)$$

which implies that the mortality density is equal to the PDF $n(d)/N$ derived from the numbers density, since $R = MN$. Hence, under the CRD, the mortality and numbers densities will always look alike. However, it is still instructive to note how the survival function changes relative to the mortality density as one changes parameters in the model; e.g., try changing mortality, M, as a simple example, while keeping all other parameters constant.

4.1.2 The basal area-size distribution

The BASD is detailed in Gove et al. (2018, 2019) for the CRD. There we show that it is simply a size-biased version or order $\alpha = 2$ of the generalized beta distribution of the first kind (GB_1). Note that the BASD also has a nice closed form based on the CR growth model and the GB_1 as given in Ducey and Gove (2015).

5 Utility Routines

5.1 Routine: `Units`

The `Units` method is simply a wrapper for the GNU `units` program (`https://www.gnu.org/software/units/`). Please note that this GNU program must be installed on your system before you can use the function described here (see the above website for details).

Please see the help for the GNU program for details on its use. The examples that follow show how to use it from within R based on `Units`. Note that there is a `units` function in the base package in R, therefore the one here is capitalized to avoid any possible ambiguity when calling the method...

```
R> Units(10, 'm', 'ft')$conv

[1] 32.808399

R> Units(c(25, 50), 'm^2/hectare', 'ft^2/acre')
```

```
                          systemCommand orig  conv
1 units -1 '25m^2/hectare' 'ft^2/acre'   25 108.9
2 units -1 '50m^2/hectare' 'ft^2/acre'   50 217.8


R> Units(c(250, 500), 'ft^3/acre', 'm^3/hectare')


                           systemCommand orig      conv
1 units -1 '250ft^3/acre' 'm^3/hectare'  250 17.493113
2 units -1 '500ft^3/acre' 'm^3/hectare'  500 34.986226
```

The first example shows how to get a simple conversion from meters to feet. The second and third illustrate that the object actually returned from units is a data frame with columns...

1. `systemCommand`: The actual command that was run in the GNU units program.
2. `orig`: The value in the orginal units.
3. `conv`: The converted value to the new units.

These examples show how you can get more than one conversion at a time, and if you run `example(Units)` at the R prompt, you will also see that it is possible to mix unit conversions in a given call as well.

A Relationship of q to the Scaled Exponential

The following model is due to Meyer and Stevenson (1943)

$$N = K \exp(-ad) \tag{A.1}$$

We will call it the scaled exponential model for reasons that are described in the main document (see § 3.2.1). Some of the following appears in many places (for example, see Davis and Johnson, 1987, p. 60), but it sometimes helps to have things like this handy for reference and repetition is not a bad thing.

So, recalling that w is the DBH class width, (A.1) implies that...

$$\begin{aligned} q &= \frac{K \exp(-ad)}{K \exp(-a(d+w))} \\ &= \exp(-ad) \cdot \exp(a(d+w)) \\ &= \exp(-ad + ad + aw) \\ &= \exp(aw) \end{aligned}$$

and taking the logarithm of both sides we have

$$\begin{aligned} aw &= \ln q \\ a &= \frac{\ln q}{w} \end{aligned}$$

then substituting this result back into (A.1) gives...

$$N = K \exp\left(-\frac{\ln q}{w} d\right) \tag{A.2}$$

References

L. S. Davis and K. N. Johnson. *Forest Management*. McGraw-Hill, 3rd edition, 1987. 6, 18

F. de Liocourt. De l'amenagement des sapinières. Technical report, Bulletin trimestriel, Société forestière de Franche-Comtè et Belfort, juillet 1898. (English translation: url-http://www.snr.missouri.edu/forestry/larsen.html). 6

M. J. Ducey and J. H. Gove. Size-biased distributions in the generalized beta distribution family, with applications to forestry. *Forestry*, (88):143–151, 2015. 15, 17

J. H. Gove. A demographic study of the exponential distribution applied to uneven-aged forests. *Forestry*, 90:18–31, 2017. 14, 16

J. H. Gove and G. P. Patil. Modeling the basal area-size distribution of forest stands: a compatible approach. *Forest Science*, 44(2):285–297, 1998. 11

J. H. Gove, T. B. Lynch, and M. J. Ducey. Notes on the Chapman-Richards size-structured distribution. Unpublished R vignette, USDA Forest Service, 2018. 15, 16, 17

J. H. Gove, T. B. Lynch, and M. J. Ducey. The Chapman-Richards distribution and its relationship to the generalized beta. *PLOS ONE*, 2019. In review. 15, 16, 17

G. Kerr. The management of sliver fir forests: de Liocourt (1898) revisited. *Forestry*, 87:29–28, 2014. 6

H. A. Meyer. Structure, growth, and drain in balanced uneven-aged forests. *Journal of Forestry*, 50:85–92, 1952. 5

H. A. Meyer and D. D. Stevenson. The structure and growth of virgin beech-birch-maple-hemlock forests in northern Pennsylvania. *Journal of Agricultural Research*, 67(12):465–484, 1943. 5, 8, 9, 12, 18

L. V. Pienaar and K. J. Turnbull. The Chapman-Richards generalization of Von Bertalanffy's growth model for basal area growth and yield in even-aged stands. *Forest Science*, 19:2–22, 1973. 14