

FMDB Size Structure Walk-Through

Derek H. Ogle

Packages

Add-on packages are loaded for use with `library()`. The following packages are required for the analyses demonstrated in this document. Comments following each package are the list of functions used from the package.

```
> library(fishWiDNR) # for read.FMDB()
> library(FSA)       # for headtail(), filterD(), Summarize(), hist(), lencat(), psdVal(), psdAdd()
> library(dplyr)     # for %>%, select(), mutate(), group_by(), summarize()
> library(magrittr)  # for %<>%
```

Working with Raw Fish Data from the FM Database

Retrieving Raw Fish Data from the FM Database

Raw fish data from the FM Database is extracted and saved into a comma-separated values (CSV) file that will have a common format that you are likely familiar with. In this workshop, we will all use data retrieved for Sawyer Country. Outside of this workshop, raw fish data files should be generated using the Raw Data – Fish Raw Data report in the FMDB. You can then narrow the query down to a specific survey using the search parameters within the database. Use the Actions button – Download option (choosing CSV as download format) to save the raw data file to your specified working folder.

Setting the Working Directory (Very Important)

Each session with R/Rstudio uses a working directory (or folder) that will be examined for input files or where output files will be saved. The default working directory depends on various factors (your installation, permissions, etc.) but is almost assuredly NOT what you want it to be in most instances. Thus, the first step to loading external data into R is to set the working directory to the directory/folder that contains your data (CSV) file (described above).

The easiest way to do this is to start a script in RStudio that loads the packages described previously and then save this script in the **exact same folder/directory** that contains your data (CSV) file. With your .CSV data file and your .R script file in the same directory, you can then set the working directory in RStudio by selecting the **Session** menu, **Set Working Directory** item, and **To Source File Location ...** item. This will print a `setwd()` function with the appropriate directory/folder structure into the console pane. This function should be copied from the console and pasted into your script so that you will not have to use these menu items the next time you want to run your script.

The following is what this procedure produced for running this example script on **MY COMPUTER** (i.e., this will look different on your computer).

```
> setwd("C:/aaaWork/Web/GitHub/RcourseWiDNR2016")
```

The working directory must be set prior to the steps below, unless you are using the Projects feature of RStudio (which is not demonstrated here).

Loading Raw Fish Data CSV File

The Raw Fish data (CSV) file retrieved from the FM Database must be imported into R for use. These files may be read directly into R, but the format of the data in each column and the notation for missing values must be explicitly defined. Additionally, a month field is often needed (but does not exist in the FM Database) and species names must be converted to all lower-case with the exception of the first letter for ease of use with some R functions.

An additional hurdle for some analyses is that some records in the FM Database reflect counts of fish with a particular length or range of lengths. A snippet of Raw Fish data from the FM Database for Sawyer County is shown below. This snippet illustrates some records where the length bin for an individual fish was recorded (first row) and others where the length bin for multiple fish was recorded (second row).

	County	Waterbody.Name	Survey.Year	Number.of.Fish	Length.or.Lower.Length.IN	Length.Upper.IN
1	SAWYER	SISSABAGAMA LAKE	2010	1	11.0	11.4
2	SAWYER	SISSABAGAMA LAKE	2010	2	11.5	11.9
3	SAWYER	SISSABAGAMA LAKE	2010	1	12.0	12.4
48681	SAWYER	WINDIGO LAKE	2014	1	10.5	10.9
48682	SAWYER	WINDIGO LAKE	2014	1	7.0	7.4
48683	SAWYER	BLAISDELL LAKE	2014	0	NA	NA

The analyses demonstrated in this workshop require records for individual fish and prefers more precise length measurements. Thus, these records need to be expanded (repeated) to represent individual fish and a random length within the length bin needs to be defined for each fish. For example, if two fish were recorded between 11.5 and 11.9 inches, then it is necessary to create two fish that will have random lengths between 11.5 and 11.9 inches.

The `read.FMDB()` function in the `fishWiDNR` package is designed to read the FM Database CSV file, appropriately set the format of the variables, add a month variable (in `Mon`) if `addMonth=TRUE` (the default), and change the species names to the appropriate case (in `Species1`) if `addSpecies=TRUE` (the default). Additionally, the counts of fish will be expanded if `expandCounts=TRUE` (**NOT** the default). By default the expanded lengths will be in inches, but this can be changed to mm with `whichLengths="mm"`.

The example below (assumes that `setwd()` was set previously) reads the Sawyer County data (CSV) file (and expands the counts) with `read.FMDB()` and then examines the structure (with `str()`) and first and last few rows of the data.frame in R (with `headtail()`).

```
> d <- read.FMDB("SAWYER_fish_raw_data_012915.csv",expandCounts=TRUE)
Successfully read SAWYER_fish_raw_data_012915.csv.
Set variable classes.
Created months in 'Mon'.
Renamed species in 'Species1'.
Results messages from expandCounts():
13 rows had zero or no counts in Number.of.Fish.
38448 rows had an individual measurement.
10222 rows with multiple measurements were expanded to 94790 rows of individual measurements.

> str(d)
'data.frame': 133251 obs. of 56 variables:
 $ County          : Factor w/ 1 level "SAWYER": 1 1 1 1 1 1 1 1 1 1 ...
 $ Waterbody.Name  : Factor w/ 86 levels "ALDER CREEK",...: 73 52 67 85 35 9 35 60 9 9 ...
 $ WBIC            : Factor w/ 86 levels "99999","1835700",...: 69 77 70 6 54 58 54 53 58 ..
 $ Survey.Year     : int  2010 2010 2011 2012 2012 2012 2012 2012 2012 ...
 $ Station.Name    : Factor w/ 112 levels "ALDER CREEK BELOW BEAVER POND",...: 97 76 89 11..
 $ Swims.Station.Id: int  10005657 10002492 10005658 10005544 10005605 10005611 10005605 ..
 $ Site.Seq.No     : int  123111 122805 123040 121911 122533 109201 122533 122923 109201 ..
 $ Survey.Seq.No   : int  104342960 104342961 195135342 274195365 274195372 257736123 274..
 $ Survey.Begin.Date: POSIXct, format: "2010-09-26" "2010-10-18" "2011-10-02" ...
 $ Survey.End.Date : POSIXct, format: "2010-09-27" "2010-10-20" "2011-10-02" ...
 $ Survey.Status    : Factor w/ 3 levels "DATA ENTRY COMPLETE",...: 2 2 2 2 2 2 2 2 2 ...
 $ Data.Entry.Name  : chr  "warwir" "warwir" "warwir" "warwir" ...
```

```

$ Visit.Fish.Seq.No      : int  687017 687020 693862 698904 698912 698213 698931 698908 699165 ..
$ Visit.Type            : Factor w/ 2 levels "ELECTROFISHING",...: 1 2 1 2 2 2 2 2 2 ...
$ Gear                  : Factor w/ 7 levels "BACKPACK SHOCKER",...: 2 4 2 4 4 3 4 4 3 ...
$ Sample.Date           : POSIXct, format: "2010-09-27" "2010-10-19" "2011-10-02" ...
$ Substation.Name       : Factor w/ 299 levels " IBI"," STUKY BAY (INDEX)",...: 271 NA 123 NA N..
$ Target.Species        : Factor w/ 14 levels "ALL SPECIES",...: 14 3 14 6 6 7 6 6 7 7 ...
$ Fish.Data.Seq.No      : int  9022342 9022358 9343440 9610979 9612673 9553472 9612713 9611371..
$ Net.Number            : chr  NA "2" NA "3" ...
$ Species.Code          : Factor w/ 73 levels "A01J","A02","A03",...: 73 73 73 73 73 73 73 73 7..
$ Species               : Factor w/ 73 levels "AMERICAN BROOK LAMPREY",...: 43 43 43 43 43 43 4..
$ Length.or.Lower.Length.IN: num  NA NA NA NA NA NA NA NA NA NA ...
$ Length.Upper.IN       : num  NA NA NA NA NA NA NA NA NA NA ...
$ Length.or.Lower.Length.MM: num  NA NA NA NA NA NA NA NA NA NA ...
$ Length.Upper.MM       : num  NA NA NA NA NA NA NA NA NA NA ...
$ Weight.Pounds         : num  NA NA NA NA NA NA NA NA NA NA ...
$ Weight.Grams          : num  NA NA NA NA NA NA NA NA NA NA ...
$ Gender                : Factor w/ 3 levels "F","M","U": NA NA NA NA NA NA NA NA NA NA ...
$ Disease               : Factor w/ 0 levels: NA NA NA NA NA NA NA NA NA NA ...
$ Injury.Type           : Factor w/ 1 level "DEAD": NA NA NA NA NA NA NA NA NA NA ...
$ Age..observed.annuli. : int  NA NA NA NA NA NA NA NA NA NA ...
$ Edge.Counted.Desc     : Factor w/ 1 level "Yes": NA NA NA NA NA NA NA NA NA NA ...
$ Age.Structure          : Factor w/ 3 levels "OTOLITH","SCALE",...: NA NA NA NA NA NA NA NA NA ..
$ Mark.Given            : Factor w/ 7 levels "AN","LP","LV",...: NA NA NA NA NA NA NA NA NA ..
$ Mark.Found            : Factor w/ 9 levels "AN","BC","LP",...: NA NA NA NA NA NA NA NA NA ..
$ Second.Mark.Found     : Factor w/ 1 level "PIT": NA NA NA NA NA NA NA NA NA NA ...
$ Tag.Number.Given      : chr  NA NA NA NA ...
$ Second.Tag.Number.Given : chr  NA NA NA NA ...
$ Tag.Number.Found      : chr  NA NA NA NA ...
$ Second.Tag.Number.Found : chr  NA NA NA NA ...
$ YOY                   : Factor w/ 2 levels "N","Y": NA NA NA NA NA NA NA NA NA NA ...
$ Entry.Date            : POSIXct, format: "2010-12-09" "2010-12-09" "2011-10-06" ...
$ Last.Update.Date      : POSIXct, format: NA NA NA ...
$ Data.Ent.Name         : chr  "prattf" "prattf" "prattf" "warwir" ...
$ Last.Update.Name      : chr  NA NA NA NA ...
$ Invalid.Species       : chr  NA NA NA NA ...
$ Non.Standard.Bin      : chr  NA NA NA NA ...
$ Length.Unit.Error     : chr  NA NA NA NA ...
$ Length.Outside.Range  : chr  NA NA NA NA ...
$ Count.Outside.Range   : chr  NA NA NA NA ...
$ Status.Code           : chr  NA NA NA NA ...
$ Mon                   : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<...: 9 10 10 4 4 4 4 4 4 4 ..
$ Species1              : Factor w/ 73 levels "American Brook Lamprey",...: 43 43 43 43 43 43 4..
$ Len                   : num  NA NA NA NA NA NA NA NA NA NA ...
$ lennote               : chr  "Observed length" "Observed length" "Observed length" "Observe"..

```

```
> headtail(d,n=2)
```

	County	Waterbody.Name	WBIC	Survey.Year		Station.Name
1	SAWYER	TIGER CAT FLOWAGE	2435000	2010	TIGER CAT FLOWAGE_GENERAL	LAKE STATION
2	SAWYER	NELSON LAKE	2704200	2010	NELSON LAKE_GENERAL	LAKE STATION
133250	SAWYER	WINDIGO LAKE	2046600	2014	WINDIGO LAKE_GENERAL	LAKE STATION
133251	SAWYER	WINDIGO LAKE	2046600	2014	WINDIGO LAKE_GENERAL	LAKE STATION

	Swims.Station.Id	Site.Seq.No	Survey.Seq.No	Survey.Begin.Date	Survey.End.Date
1	10005657	123111	104342960	2010-09-26	2010-09-27
2	10002492	122805	104342961	2010-10-18	2010-10-20
133250	10005544	121911	515077184	2014-10-16	2014-10-17
133251	10005544	121911	515077184	2014-10-16	2014-10-17

	Survey.Status	Data.Entry.Name	Visit.Fish.Seq.No	Visit.Type
1	DATA ENTRY COMPLETE AND PROOFED	warwir	687017	ELECTROFISHING
2	DATA ENTRY COMPLETE AND PROOFED	warwir	687020	NETTING

133250	DATA ENTRY COMPLETE AND PROOFED				spooner_treaty	723742	ELECTROFISHING	
133251	DATA ENTRY COMPLETE AND PROOFED				spooner_treaty	723742	ELECTROFISHING	
	Gear	Sample.Date	Substation.Name	Target.Species	Fish.Data.Seq.No	Net.Number		
1	BOOM SHOCKER	2010-09-27	UPPER TWIN	WALLEYE	9022342	<NA>		
2	FYKE NET	2010-10-19	<NA>	BLACK CRAPPIE	9022358	2		
133250	BOOM SHOCKER	2014-10-17	GLIFWC	WALLEYE	10711295	<NA>		
133251	BOOM SHOCKER	2014-10-17	GLIFWC	WALLEYE	10711295	<NA>		
	Species.Code	Species	Length.or.Lower.Length.IN	Length.Upper.IN				
1	Z98	NO FISH CAPTURED	NA	NA				
2	Z98	NO FISH CAPTURED	NA	NA				
133250	X22	WALLEYE	18	18.4				
133251	X22	WALLEYE	18	18.4				
	Length.or.Lower.Length.MM	Length.Upper.MM	Weight.Pounds	Weight.Grams	Gender	Disease.		
1	NA	NA	NA	NA	<NA>	<NA>		
2	NA	NA	NA	NA	<NA>	<NA>		
133250	457.2	467.36	NA	NA	<NA>	<NA>		
133251	457.2	467.36	NA	NA	<NA>	<NA>		
	Injury.Type	Age..observed.annuli.	Edge.Counted.Desc	Age.Structure	Mark.Given	Mark.Found		
1	<NA>	NA	<NA>	<NA>	<NA>	<NA>		
2	<NA>	NA	<NA>	<NA>	<NA>	<NA>		
133250	<NA>	NA	<NA>	<NA>	<NA>	<NA>		
133251	<NA>	NA	<NA>	<NA>	<NA>	<NA>		
	Second.Mark.Found	Tag.Number.Given	Second.Tag.Number.Given	Tag.Number.Found				
1	<NA>	<NA>	<NA>	<NA>				
2	<NA>	<NA>	<NA>	<NA>				
133250	<NA>	<NA>	<NA>	<NA>				
133251	<NA>	<NA>	<NA>	<NA>				
	Second.Tag.Number.Found	YOY	Entry.Date	Last.Update.Date	Data.Ent.Name	Last.Update.Name		
1	<NA>	<NA>	2010-12-09	<NA>	prattf	<NA>		
2	<NA>	<NA>	2010-12-09	<NA>	prattf	<NA>		
133250	<NA>	<NA>	2015-01-21	<NA>	spooner_treaty	<NA>		
133251	<NA>	<NA>	2015-01-21	<NA>	spooner_treaty	<NA>		
	Invalid.Species	Non.Standard.Bin	Length.Unit.Error	Length.Outside.Range	Count.Outside.Range			
1	<NA>	<NA>	<NA>	<NA>	<NA>			
2	<NA>	<NA>	<NA>	<NA>	<NA>			
133250	<NA>	<NA>	<NA>	<NA>	<NA>			
133251	<NA>	<NA>	<NA>	<NA>	<NA>			
	Status.Code	Mon	Species1	Len	lennote			
1	<NA>	Sep	No Fish Captured	NA	Observed length			
2	<NA>	Oct	No Fish Captured	NA	Observed length			
133250	<NA>	Oct	Walleye	18.4	Expanded length			
133251	<NA>	Oct	Walleye	18.4	Expanded length			

Manipulating data.frames for Use

Selecting a Subset of Variables

This data.frame contains a large number of variables, many of which you are likely not interested in for a particular analysis. The data.frame can be reduced to a smaller number of variables with `select()` from the `dplyr` package. This function takes the original data.frame as the first argument followed by the names of the variables that you wish to retain as additional arguments. The code below reduces the original data.frame to only six variables.

```
> d <- select(d, Species1, Waterbody.Name, Gear, Survey.Year, Mon, Len)
```

The `%>%` from the `dplyr` package and `%<>%` from the `magrittr` package can simplify this code. The `%>%` operator “pipes” the data.frame on the left of the operator into the first argument of the function to the right of the operator. For example, the code above could be written as follows.

```
> d <- d %>% select(Species1, Waterbody.Name, Gear, Survey.Year, Mon, Len)
```

The `%<>%` operator also “pipes” the data.frame on the left of the operator into the first argument of the function to the right of the operator, but it **ALSO** assigns the result from the right of the operator to the object to the left of the operator. Thus, the code from above is simplified as follows.

```
> d %<>% select(Species1, Waterbody.Name, Gear, Survey.Year, Mon, Len)
> headtail(d, n=2)
```

	Species1	Waterbody.Name	Gear	Survey.Year	Mon	Len
1	No Fish Captured	TIGER CAT FLOWAGE	BOOM SHOCKER	2010	Sep	NA
2	No Fish Captured	NELSON LAKE	FYKE NET	2010	Oct	NA
133250	Walleye	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	18.4
133251	Walleye	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	18.4

Adding Variables

One often needs to construct a new variable from other variables and add this new variable to the data.frame. A variable can be added to a data.frame with `mutate()` from the `dplyr` package. This function takes the data.frame as the first argument and an argument of the form `newvar=expression`, where `newvar` is the name for the new variable and `expression` describes how the new variable is constructed. For example, the following code adds the natural log of length to the data.frame (note that the `%<>%` was used to pipe `d` into the first argument of `mutate()` so that it appears that `mutate()` only has one argument).

```
> d %<>% mutate(loglen=log(Len))
> headtail(d)
```

	Species1	Waterbody.Name	Gear	Survey.Year	Mon	Len	loglen
1	No Fish Captured	TIGER CAT FLOWAGE	BOOM SHOCKER	2010	Sep	NA	NA
2	No Fish Captured	NELSON LAKE	FYKE NET	2010	Oct	NA	NA
3	No Fish Captured	SPIDER LAKE	BOOM SHOCKER	2011	Oct	NA	NA
133249	Largemouth Bass	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	12.1	2.493205
133250	Walleye	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	18.4	2.912351
133251	Walleye	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	18.4	2.912351

Adding Length Categories

Length categories are the variable most often added to a data.frame. The `lencat()` function from the FSA package provides a flexible mechanism for adding this variable. When used within `mutate()`, `lencat()` takes the name of the variable with the length data as the first argument and the width of the length categories in `w=`. This function will choose “smart” starting values for the categories, but you can also set the starting values with `startcat=` (not demonstrated here). The following code creates a new variable called `Len1` that contains 1-in length categories derived from the `Len` variable.

```
> d %<>% mutate(Len1=lencat(Len,w=1))
> headtail(d)
```

	Species1	Waterbody.Name	Gear	Survey.Year	Mon	Len	loglen	Len1
1	No Fish Captured	TIGER CAT FLOWAGE	BOOM SHOCKER	2010	Sep	NA	NA	NA
2	No Fish Captured	NELSON LAKE	FYKE NET	2010	Oct	NA	NA	NA
3	No Fish Captured	SPIDER LAKE	BOOM SHOCKER	2011	Oct	NA	NA	NA
133249	Largemouth Bass	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	12.1	2.493205	12
133250	Walleye	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	18.4	2.912351	18
133251	Walleye	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	18.4	2.912351	18

Specific length categories (“Stock”, “Quality”, etc.) have been defined for fisheries management purposes for many game species. These length categories can be viewed for a species with `psdVal()` from the FSA package. The first argument to this function is the name of the species. The default is to return the lengths in mm, but lengths in inches may be returned by including `units="in"`. These length categories are shown below for Largemouth Bass and Walleye. [Note that the “substock” value is also returned for completeness.]

```
> psdVal("Largemouth Bass",units="in")
substock    stock    quality preferred memorable    trophy
      0         8        12         15         20         25

> psdVal("Walleye",units="in")
substock    stock    quality preferred memorable    trophy
      0        10        15         20         25         30
```

Thus, for example, Largemouth Bass that are 12 inches and longer are considered “quality” fish, whereas Largemouth Bass 20 inches and longer are categorized as “memorable.”

A variable that contains these categories for a single species can be added to a data.frame with `lencat()` and the `breaks=` argument (*this is demonstrated below for PSD calculations*). However, it is more efficient to add a variable to a data.frame with these categories for **ALL** species for which the categories have been defined. This is accomplished with `psdAdd()` from the FSA package, which when used with `mutate()`, requires the name of the variable with the length measurements as the first argument, the name of the variable with the species names as the second argument, and, if the measurements are in inches, `units="in"`. Note that `psdAdd()` will, by default, return a message for each species for which these length categories have not been defined. The `verbose=FALSE` argument used below will suppress these messages (and is only done here to save space).

```
> d %<>% mutate(Lcat=psdAdd(Len,Species1,units="in",verbose=FALSE))
> headtail(d)
```

	Species1	Waterbody.Name	Gear	Survey.Year	Mon	Len	loglen	Len1	Lcat
1	No Fish Captured	TIGER CAT FLOWAGE	BOOM SHOCKER	2010	Sep	NA	NA	NA	<NA>
2	No Fish Captured	NELSON LAKE	FYKE NET	2010	Oct	NA	NA	NA	<NA>
3	No Fish Captured	SPIDER LAKE	BOOM SHOCKER	2011	Oct	NA	NA	NA	<NA>
133249	Largemouth Bass	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	12.1	2.493205	12	quality
133250	Walleye	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	18.4	2.912351	18	quality
133251	Walleye	WINDIGO LAKE	BOOM SHOCKER	2014	Oct	18.4	2.912351	18	quality

Selecting a Subset of Individuals

The current data.frame contains information from Sawyer County for many years, seasons, species, water bodies, etc. A particular analysis may require you to restrict the data to a species, a water body, a species in a waterbody, a species in a waterbody within a year, etc. Subsets of a data.frame can be constructed with `filterD()` from the `FSA` package (which is simply a modification of `filter()` from the `dplyr` package). This function requires the original data.frame as the first argument followed by expressions that describe the condition for a subset. Multiple conditions (arguments) are joined with an “and” (such that both conditions must be true).

For example, the code below constructs a new data.frame (`Spr`) from `d` that contains only fish that were captured in April, May, or June of 2013. It is generally a good habit to examine the structure of a data.frame following filtering to make sure that it contains the individuals of interest. What evidence is there in this structure that the filtering was successful?

```
> Spr <- filterD(d, Survey.Year==2013, Mon %in% c("Apr", "May", "Jun"))
> str(Spr)
'data.frame': 18903 obs. of 9 variables:
 $ Species1      : Factor w/ 22 levels "Black Bullhead",...: 3 1 21 1 7 22 21 21 12 14 ...
 $ Waterbody.Name: Factor w/ 14 levels "BLACK DAN LAKE",...: 4 1 1 1 9 9 9 9 9 9 ...
 $ Gear          : Factor w/ 2 levels "BOOM SHOCKER",...: 2 2 2 2 2 2 2 2 2 2 ...
 $ Survey.Year   : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ Mon           : Ord.factor w/ 2 levels "May"<"Jun": 1 1 1 1 1 1 1 1 1 1 ...
 $ Len           : num  NA NA NA NA NA NA NA NA NA NA NA ...
 $ loglen        : num  NA NA NA NA NA NA NA NA NA NA NA ...
 $ Len1          : num  NA NA NA NA NA NA NA NA NA NA NA ...
 $ Lcat          : Factor w/ 6 levels "substock","stock",...: NA NA NA NA NA NA NA NA NA NA ...
```

Three other subsets are created below for later use. Can you describe the data in each of these data.frames?

```
> BGSpr <- filterD(Spr, Species1=="Bluegill")
> BGSprLC <- filterD(BGSpr, Waterbody.Name=="LAKE CHETAC", Gear=="BOOM SHOCKER")
> SprLC <- filterD(Spr, Waterbody.Name=="LAKE CHETAC")
```


Summary Statistics and Graphics

Simple Summaries

Frequencies of individuals are created with `xtabs()` which takes a formula of the form `~rows` or `~rows+cols`, where `rows` and `cols` generically represent the variables to form the rows and columns, respectively, of the frequency table. The `data.frame` that contains the `rows` and `cols` variables must be given in `data=`.

The following two uses of `xtabs()` produce frequency tables of the number of fish of each type captured from Lake Chetac and the number of Bluegill by month and waterbody in spring samples from 2013.

```
> xtabs(~Species1,data=SprLC)
```

```
Species1
  Black Crappie      Bluegill      Bowfin Largemouth Bass      Northern Pike      Pumpkinseed
            3619            589            10            274            40            36
    Rock Bass Smallmouth Bass      Walleye      Yellow Perch
            2            10            72            2222
```

```
> xtabs(~Mon+Waterbody.Name,data=BGSpr)
```

```
Waterbody.Name
Mon  BLACK DAN LAKE CONNORS LAKE DURPHEE LAKE GREEN LAKE LAKE CHETAC LAKE CHIPPEWA
May      599            90            603            0            589            746
Jun        0            108            0            144            0            0
Waterbody.Name
Mon  LAKE OF THE PINES LOWER CLAM LAKE MOOSE LAKE ROUND LAKE WHITEFISH LAKE
May      213            35            1            414            72
Jun        90            0            0            0            0
```

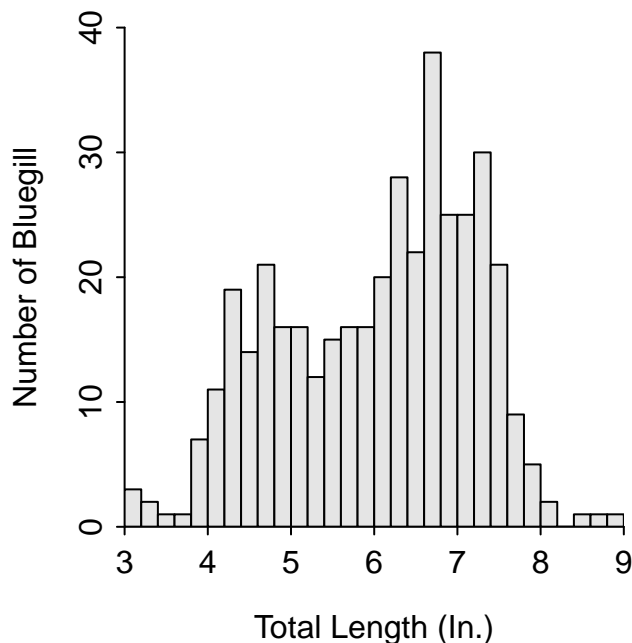
Summary statistics for a quantitative variable are efficiently computed with `Summarize()` (note the capital S) from the FSA package. This function takes a formula of the form `~quant` as the first argument, where `quant` generically represents the name of a quantitative variable, and the `data.frame` that contains that variable in `data=`. The number of decimals for the returned statistics is optionally controlled with `digits=`. The code below summarizes the lengths of Bluegill captured in Spring samples from Lake Chetac.

```
> Summarize(~Len,data=BGSprLC,digits=2)
```

```
      n  nvalid    mean     sd    min     Q1  median     Q3    max percZero
398.00  398.00    5.98    1.16    3.00    5.00    6.20    6.90    8.90    0.00
```

A histogram of a quantitative variable is constructed with `hist()` from the FSA package. The first two arguments to `hist()` are exactly the same as the first two arguments to `Summarize()`. The cutoffs for the bins used in the histogram may be defined with `breaks=`. For fisheries purposes, the cutoffs are usually evenly spaced and, thus, can be easily constructed as a sequence of values. A simple sequence of values is constructed with `seq()`, which takes the starting value of the sequence as the first argument, the ending value as the last argument, and the step width for the sequence as the last argument. For example, the code in the `breaks=` below constructs a sequence of numbers that starts with 3, ends with 9, and has steps of 0.2 (i.e., 3, 3.2, 3.4, etc.). The x and y axes of the histogram are labeled with a string in `xlab=` and `ylab=`, respectively. The numerical limits of the x and y axes are controlled by including a vector of two values that represent the minimum and maximum values for the axes in `xlim=` and `ylim=`, respectively. The code below constructs the length frequency histogram for Bluegill captured from Lake Chetac in Spring, 2013.

```
> hist(~Len,data=BGSprLC,breaks=seq(3,9,0.2),
      xlab="Total Length (In.)",ylab="Number of Bluegill",
      xlim=c(3,9),ylim=c(0,40))
```

Multiple Summaries

Numerical summaries can be efficiently computed for multiple groups using a combination of `group_by()` and `summarize()` (note the lower-case s) from the `dplyr` package. The `group_by` function creates groups for the data.frame given in its first argument based on the groups given in its ensuing arguments. The `group_by()` function below will create groups based on `Waterbody.Name` (i.e., by lakes or rivers) for the `BGSpr` data.frame (which is “piped” into the first argument of `group_by()` with `%>%`). The `summarize()` function then creates summaries defined by the user for each group. A wide variety of summaries can be defined, but the example below uses `n()` to count the number of individuals (i.e., sample size), `mean()` to compute an average, `sd()` to compute a standard deviation, and `min()` and `max()` to compute the minimum and maximum values. The `n()` function does not require any arguments, whereas the other functions require the name of a quantitative variable as the first argument. Also note that these functions use `na.rm=TRUE` to remove missing values from the variable (otherwise, for example, a mean will not be computed). Note that the `sum(!is.na(Len))` code is a “trick” used to count the number of non-missing values in the variable (a “valid” sample size). The result from `summarize()` is “piped” into `as.data.frame()` to remove the grouping structure (which is not needed after the summarization) and other attributes that were added to the result. Thus, the code below computes the sample size, valid sample size, and mean, standard deviation, minimum, and maximum lengths of Bluegill in all water bodies sampled in Spring, 2013.

```
> BGSpr %>%
  group_by(Waterbody.Name) %>%
  summarize(n=n(), valid_n=sum(!is.na(Len)),
            meanLen=mean(Len, na.rm=TRUE), sdLen=sd(Len, na.rm=TRUE),
            minLen=min(Len, na.rm=TRUE), maxLen=max(Len, na.rm=TRUE) ) %>%
  as.data.frame()
```

	Waterbody.Name	n	valid_n	meanLen	sdLen	minLen	maxLen
1	BLACK DAN LAKE	599	241	4.352697	0.9151520	2.1	7.0
2	CONNORS LAKE	198	108	5.155556	1.1018534	1.7	7.0
3	DURPHEE LAKE	603	574	6.603136	0.5071123	1.4	7.9
4	GREEN LAKE	144	144	6.567361	1.1392446	2.8	8.4
5	LAKE CHETAC	589	400	5.979250	1.1819420	2.0	8.9
6	LAKE CHIPPEWA	746	181	5.758011	1.1447001	3.7	8.0
7	LAKE OF THE PINES	303	90	5.000000	1.1646478	1.7	6.8
8	LOWER CLAM LAKE	35	35	4.554286	1.0042096	2.7	6.2
9	MOOSE LAKE	1	0	NaN	NaN	NA	NA
10	ROUND LAKE	414	309	5.070874	1.3018442	1.8	8.7
11	WHITEFISH LAKE	72	67	4.392537	1.3614067	2.1	7.4

The code below is similar to that above except that it uses all Spring catches grouped by fish species within water bodies, and the mean and standard deviations were rounded to two decimal places. In addition, the result was “piped” into `write.csv` which will write the results to the filename given in the first argument (in the current working directory). Note that `row.names=FALSE` was used in `write.csv` to suppress the writing of unneeded row numbers to the file.

```
> Spr %>%
  group_by(Waterbody.Name,Species1) %>%
  summarize(n=n(),valid_n=sum(!is.na(Len)),
            meanLen=round(mean(Len,na.rm=TRUE),2),sdLen=round(sd(Len,na.rm=TRUE),2),
            minLen=min(Len,na.rm=TRUE),maxLen=max(Len,na.rm=TRUE) ) %>%
  as.data.frame() %>%
  write.csv("LenSum_Sawyer_Spr13.csv",row.names=FALSE)
```

PSD Calculations

Single Waterbody and Species (version 1)

The PSD values are calculated in three steps – (1) find the frequency of individuals between each category with `xtabs()` (as described above), (2) compute a reverse cumulative summary with `rcumsum()` from the `xtabs()` result to find the frequency of individuals in each category (i.e., each category is the number of fish of that size **or greater**), and (3) divide each reverse cumulative frequency by the number of stock-length fish and multiply by 100. These calculations are shown below for the Lake Chetac Bluegill.

```
> ( freq <- xtabs(~Lcat,data=BGSprLC) )
Lcat
  stock  quality preferred
    170    223         5

> ( rcum <- rcumsum(freq) )
  stock  quality preferred
    398    228         5

> rcum["stock"]
stock
 398

> rcum/rcum["stock"]*100
  stock  quality preferred
100.000000  57.286432   1.256281
```

Thus, the PSD-Q for Lake Chetac Bluegill in Spring, 2013 is 57.

Single Waterbody and Species (version 2)

Lengths other than those defined for a species may be of particular interest to the fisheries biologist. For example, one may be interested in PSD-7, which is the percentage of stock-length Bluegill that are 7 inches or longer. To perform this calculation, 7 inches must first be added to the vector of length categories returned by `psdVal()` by including the `addLens=` argument. This vector is then given to `breaks=` in `lencat()` to create a new length category variable that will include a category for 7 inches. In this example, `use.names=TRUE` was used so that category names (“stock”, “quality”, etc.) are used rather than numbers (“3”, “6”, etc.) and `drop.levels=TRUE` was used to drop categories for which no fish were found in the data.frame (for example, “trophy” will be dropped here because no trophy-length fish were captured).

```

> ( brks <- psdVal("Bluegill",units="in",addLens=7) )
substock      stock      quality      7 preferred memorable trophy
      0          3          6          7          8          10          12

> BGSPrLC %<>% mutate(Lcat2=lencat(Len,breaks=brks,use.names=TRUE,drop.levels=TRUE))
> headtail(BGSPrLC)
  Species1 Waterbody.Name      Gear Survey.Year Mon Len  loglen Len1  Lcat  Lcat2
1  Bluegill LAKE CHETAC BOOM SHOCKER 2013 May 4.0 1.386294 4 stock stock
2  Bluegill LAKE CHETAC BOOM SHOCKER 2013 May 4.7 1.547563 4 stock stock
3  Bluegill LAKE CHETAC BOOM SHOCKER 2013 May 4.7 1.547563 4 stock stock
396 Bluegill LAKE CHETAC BOOM SHOCKER 2013 May 5.6 1.722767 5 stock stock
397 Bluegill LAKE CHETAC BOOM SHOCKER 2013 May 6.6 1.887070 6 quality quality
398 Bluegill LAKE CHETAC BOOM SHOCKER 2013 May 6.6 1.887070 6 quality quality

```

This new variable is then summarized as shown above (code below also demonstrates how to round the final result to one decimal place).

```

> ( freq <- xtabs(~Lcat2,data=BGSPrLC) )
Lcat2
      stock      quality      7 preferred
      170      133      90      5

> ( rcum <- rcumsum(freq) )
      stock      quality      7 preferred
      398      228      95      5

> round(rcum/rcum["stock"]*100,1)
      stock      quality      7 preferred
    100.0      57.3      23.9      1.3

```

Thus, the PSD-7 for Lake Chetac Bluegill in Spring, 2013 is 24.

Multiple Waterbodies and Single Species

Frequencies can be computed for multiple water bodies using `xtabs()` as described above (note that a new variable was added to the data.frame of Spring-captured Bluegill from multiple water bodies that included the 7 inch category.)

```

> BGSPr %<>% mutate(Lcat2=lencat(Len,breaks=brks,use.names=TRUE,drop.levels=TRUE))
> ( freq <- xtabs(~Waterbody.Name+Lcat2,data=BGSPr) )
      Waterbody.Name      Lcat2
substock stock quality  7 preferred
BLACK DAN LAKE      5      227      7      2      0
CONNORS LAKE      6      73      28      1      0
DURPHEE LAKE      1      36      414 123      0
GREEN LAKE      2      30      49 55      8
LAKE CHETAC      1      170      133 90      6
LAKE CHIPPEWA      0      101      44 35      1
LAKE OF THE PINES      7      66      17      0      0
LOWER CLAM LAKE      1      30      4      0      0
MOOSE LAKE      0      0      0      0      0
ROUND LAKE      13      221      49 20      6
WHITEFISH LAKE      8      50      4      5      0

```

However, it is not as simple to convert these frequencies into PSD values because the reverse cumulative sum must be computed for each **row** of the frequency table. The `apply()` function can be used to “apply” a function to each row or column of a matrix. The `apply()` function takes the matrix (or frequency table, in this case) as the first argument and the function to “apply” is given in `FUN=`. The function will be applied to the rows if `MARGIN=1` and to the columns if `MARGIN=2`. This calculation is demonstrated below.

```
> # apply() result has wrong orientation, only partial results shown
> apply(freq,FUN=rcumsum,MARGIN=1)
```

	Waterbody.Name					
Lcat2	BLACK DAN LAKE	CONNORS LAKE	DURPHEE LAKE	GREEN LAKE	LAKE CHETAC	LAKE CHIPPEWA
substock	241	108	574	144	400	181
stock	236	102	573	142	399	181
quality	9	29	537	112	229	80
7	2	1	123	63	96	36
preferred	0	0	0	8	6	1

Unfortunately, the results from `apply()` are not oriented to meet our needs (i.e., water bodies are in columns rather than rows). The orientation of the result can be “transposed” with `t()` as shown below (and note that this result is assigned to the `rcum` object).

```
> ( rcum <- t(apply(freq,FUN=rcumsum,MARGIN=1)) )
```

	Lcat2					
Waterbody.Name	substock	stock	quality	7	preferred	
BLACK DAN LAKE	241	236	9	2	0	
CONNORS LAKE	108	102	29	1	0	
DURPHEE LAKE	574	573	537	123	0	
GREEN LAKE	144	142	112	63	8	
LAKE CHETAC	400	399	229	96	6	
LAKE CHIPPEWA	181	181	80	36	1	
LAKE OF THE PINES	90	83	17	0	0	
LOWER CLAM LAKE	35	34	4	0	0	
MOOSE LAKE	0	0	0	0	0	
ROUND LAKE	309	296	75	26	6	
WHITEFISH LAKE	67	59	9	5	0	

Another problem here is that these results contain the substock-length fish. These fish can be removed by eliminating the first column in the `rcum` table.

```
> rcum <- rcum[,-1]
```

The PSD values are then computed as before.

```
> round(rcum/rcum[, "stock"]*100,1)
```

	Lcat2				
Waterbody.Name	stock	quality	7	preferred	
BLACK DAN LAKE	100	3.8	0.8	0.0	
CONNORS LAKE	100	28.4	1.0	0.0	
DURPHEE LAKE	100	93.7	21.5	0.0	
GREEN LAKE	100	78.9	44.4	5.6	
LAKE CHETAC	100	57.4	24.1	1.5	
LAKE CHIPPEWA	100	44.2	19.9	0.6	
LAKE OF THE PINES	100	20.5	0.0	0.0	
LOWER CLAM LAKE	100	11.8	0.0	0.0	
MOOSE LAKE	NaN	NaN	NaN	NaN	
ROUND LAKE	100	25.3	8.8	2.0	
WHITEFISH LAKE	100	15.3	8.5	0.0	

Multiple Species in a Single Waterbody

Similar code can be used to summarize multiple species in a single waterbody. However, there is no simple way to use “other” lengths for individual species (e.g., it is not simple to include a 7 inch category for Bluegill and a 14 inch category for Largemouth Bass). Thus, this summary can only be computed for the main length categories (which are in the LCat variable for the data.frames created further above).

```
> ( freq <- xtabs(~Species1+Lcat,data=SprLC) )
```

Species1	Lcat				
	substock	stock	quality	preferred	memorable
Black Crappie	28	453	52	14	1
Bluegill	1	170	223	6	0
Bowfin	0	0	0	0	0
Largemouth Bass	19	81	112	62	0
Northern Pike	0	20	10	9	1
Pumpkinseed	0	16	20	0	0
Rock Bass	0	0	0	0	0
Smallmouth Bass	1	2	5	2	0
Walleye	8	17	9	20	17
Yellow Perch	34	257	91	3	0

```
> ( rcum <- t(apply(freq,FUN=rcumsum,MARGIN=1)) )
```

Species1	Lcat				
	substock	stock	quality	preferred	memorable
Black Crappie	548	520	67	15	1
Bluegill	400	399	229	6	0
Bowfin	0	0	0	0	0
Largemouth Bass	274	255	174	62	0
Northern Pike	40	40	20	10	1
Pumpkinseed	36	36	20	0	0
Rock Bass	0	0	0	0	0
Smallmouth Bass	10	9	7	2	0
Walleye	71	63	46	37	17
Yellow Perch	385	351	94	3	0

```
> rcum <- rcum[,-1]
```

```
> round(rcum/rcum[, "stock"]*100,1)
```

Species1	Lcat			
	stock	quality	preferred	memorable
Black Crappie	100	12.9	2.9	0.2
Bluegill	100	57.4	1.5	0.0
Bowfin	NaN	NaN	NaN	NaN
Largemouth Bass	100	68.2	24.3	0.0
Northern Pike	100	50.0	25.0	2.5
Pumpkinseed	100	55.6	0.0	0.0
Rock Bass	NaN	NaN	NaN	NaN
Smallmouth Bass	100	77.8	22.2	0.0
Walleye	100	73.0	58.7	27.0
Yellow Perch	100	26.8	0.9	0.0

Reproducibility Information

- **Compiled Date:** Sun Mar 06 2016
- **Compiled Time:** 1:06:56 PM
- **R Version:** R version 3.2.3 (2015-12-10)
- **System:** Windows, i386-w64-mingw32/i386 (32-bit)
- **Base Packages:** base, datasets, graphics, grDevices, methods, stats, utils
- **Required Packages:** knitr, fishWiDNR, FSA, dplyr, magrittr and their dependencies (assertthat, car, DBI, digest, evaluate, formatR, gdata, gplots, graphics, grDevices, highr, Hmisc, lazyeval, markdown, methods, plotrix, plyr, R6, Rcpp, sciplot, stats, stringr, tools, utils, yaml)
- **Other Packages:** dplyr_0.4.3, fishWiDNR_0.0.6, FSA_0.8.6, knitr_1.12.3, magrittr_1.5
- **Loaded-Only Packages:** assertthat_0.1, DBI_0.3.1, digest_0.6.9, evaluate_0.8, formatR_1.2.1, gdata_2.17.0, gtools_3.5.0, htmltools_0.3, lazyeval_0.1.10, lubridate_1.5.0, parallel_3.2.3, plyr_1.8.3, R6_2.1.2, Rcpp_0.12.3, rmarkdown_0.9.2, stringi_1.0-1, stringr_1.0.0, tools_3.2.3, yaml_2.1.13
- **Links:** [Script](#) / [RMarkdown](#)