

Length-frequency data for TropFishR

Tobias K. Mildenerberger

2020-03-09

#Background Length-frequency data (LFQ) is collected by measuring the length of focal fish species onboard of the fishing vessel, the landing site or the fish market. Concerning fish there are three different lengths which can be measured: standard length (SL), fork length (FL), and total length (TL), for more information see [here](#). The big advantage of LFQ data is that only a representative subsample of the total catch is needed, meaning that not every fish in the catch has to be measured if the subsample is random but representative. The sample should be representative of the whole stock under investigation and for all seasons of a year. Furthermore, enough samples of a range of different fish lengths have to be measured in order for the methods of **TropFishR** to work - a reference value is around 50 fish measurements per month. Assuming that collected data are entered in some spreadsheet or text format, this document will illustrate how different arrangements of data can be read into R and converted to the LFQ list required by all methods within **TropFishR**. No manual data input in R is required, The data used in the examples below were generated randomly.

#LFQ import There are three main ways in which you can arrange your data in a spreadsheet or text file and easily import it into R and convert it to a LFQ list (Fig. 1-3). Each of these 3 methods works evenly well with a 'csv', 'xls', or 'txt' file. For simplification, I will present all three methods with 'csv' files and consequently demonstrate how to use a 'xlsx' and a 'txt' file. This vignette requires to load the **TropFishR** package, `require(TropFishR)`.

##Data arrangement 1 The first data arrangement assumes that each measurement was inserted into a specific cell in the spreadsheet, together with associated date when this measurement was done.

The data in this arrangement can be loaded into R by following code.

```
lfq1 <- read.csv2("lfq1.csv")

lfq1$date <- as.Date(lfq1$date, format = "%d.%m.%Y")

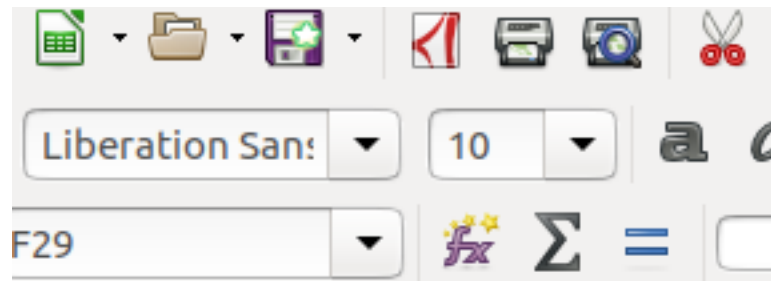
lfq1new <- lfqCreate(data = lfq1, Lname = "length", Dname = "date")

plot(lfq1new, Fname = "catch")
```

It is always important to convert the date information into the R class 'Date' which can be done by the function `as.Date()`. The second argument in that function displays the format of the date in your data, '%d' stand for the day, '%m' for the month, and '%Y' for the year with the century, e.g. 2014, without the century it would be '%y', e.g. 14. The separator, here '.', should be the same as in your data (see other examples below). The function `lfqCreate()` is a handy function, which allows you to convert the dataframe into a list in the right format. The arguments 'Lname' and 'Dname' define the names of the columns in your dataframe with the length measurements and the dates, respectively. They should correspond to `colnames(lfq1)`. Besides the arguments above it allows for many other adjustments, such as setting the bin size or aggregating the data monthly, see `?lfqCreate` for more information.

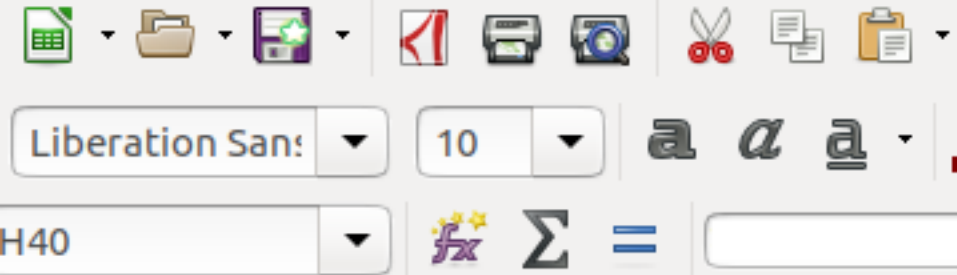
##Data arrangement 2 The second data arrangement represents the case with an additional column which contains the number of times the length was measured at respective date.

Data of this arrangement can be loaded into R by following code.



	A	B	C
1	date	length	
2	15.01.2016	10	
3	15.01.2016	25	
4	15.01.2016	50	
5	15.01.2016	36	
6	15.01.2016	43	
7	15.01.2016	34	
8	15.01.2016	12	
9	15.01.2016	14	
10	15.01.2016	45	
11	15.01.2016	32	
12	15.01.2016	10	
13	15.01.2016	25	
14	15.01.2016	50	
15	15.01.2016	36	
16	15.01.2016	43	
17	15.01.2016	34	
18	15.01.2016	12	

Figure 1: Data arrangement 1. A column with dates and a column with length measurements.



	A	B	C	D
1	DATE	Length_CM	Frequency	
2	15/01/16	10	3	
3	15/01/16	25,5	5	
4	15/01/16	50,5	2	
5	15/01/16	36	6	
6	15/01/16	43,5	10	
7	15/01/16	34	23	
8	15/01/16	12	12	
9	15/01/16	14	2	
10	15/01/16	45	3	
11	15/01/16	32	4	
12	15/01/16	10	11	
13	15/01/16	25	20	
14	15/01/16	50	1	
15	15/01/16	36	2	
16	15/01/16	43,5	4	
17	15/01/16	34	5	
18	15/01/16	12	2	

Figure 2: Data arrangement 2. A column with dates, one with length measurements, and one with the frequency of the length at respective date.

```
lfq2 <- read.csv2("lfq2.csv")

lfq2$date <- as.Date(lfq2$date, format = "%d/%m/%y")

lfq2new <- lfqCreate(data = lfq2, Lname = "Length_CM", Dname = "DATE", Fname = "Frequency")

plot(lfq2new, Fname = "catch")
```

Be aware of the different date format, the different column headers or names and the additional argument in the function `lfqCreate()` called 'Fname' displaying the name of the column with the frequency information.

Data arrangement 3 The last data arrangement presented here, assumes that the data was already sorted into length classes within your spreadsheet. The first column does now represent the length classes and the number of times each length class was observed at respective sampling times is in the subsequent columns, where each column represent one sampling time (here aggregated per month). In this example, the information about the date is contained within the header of the not-length-class columns (cp Fig. 3). This is however not necessary and the date information can also be added manually in R.

Data in this arrangement have to be treated a little differently:

```
lfq3 <- read.csv2("lfq3.csv")

dates <- colnames(lfq3)[-1]
dates <- strsplit(dates, "X")
dates <- unlist(lapply(dates, function(x) x[2]))
dates <- as.Date(dates, "%Y-%d-%m")

lfq3new <- list(dates = dates,
               midLengths = lfq3$lengthClass,
               catch = as.matrix(lfq3[,-1]))
class(lfq3new) <- "lfq"
lfq3new$catch[is.na(lfq3new$catch)] <- 0

plot(lfq3new, Fname = "catch")
```

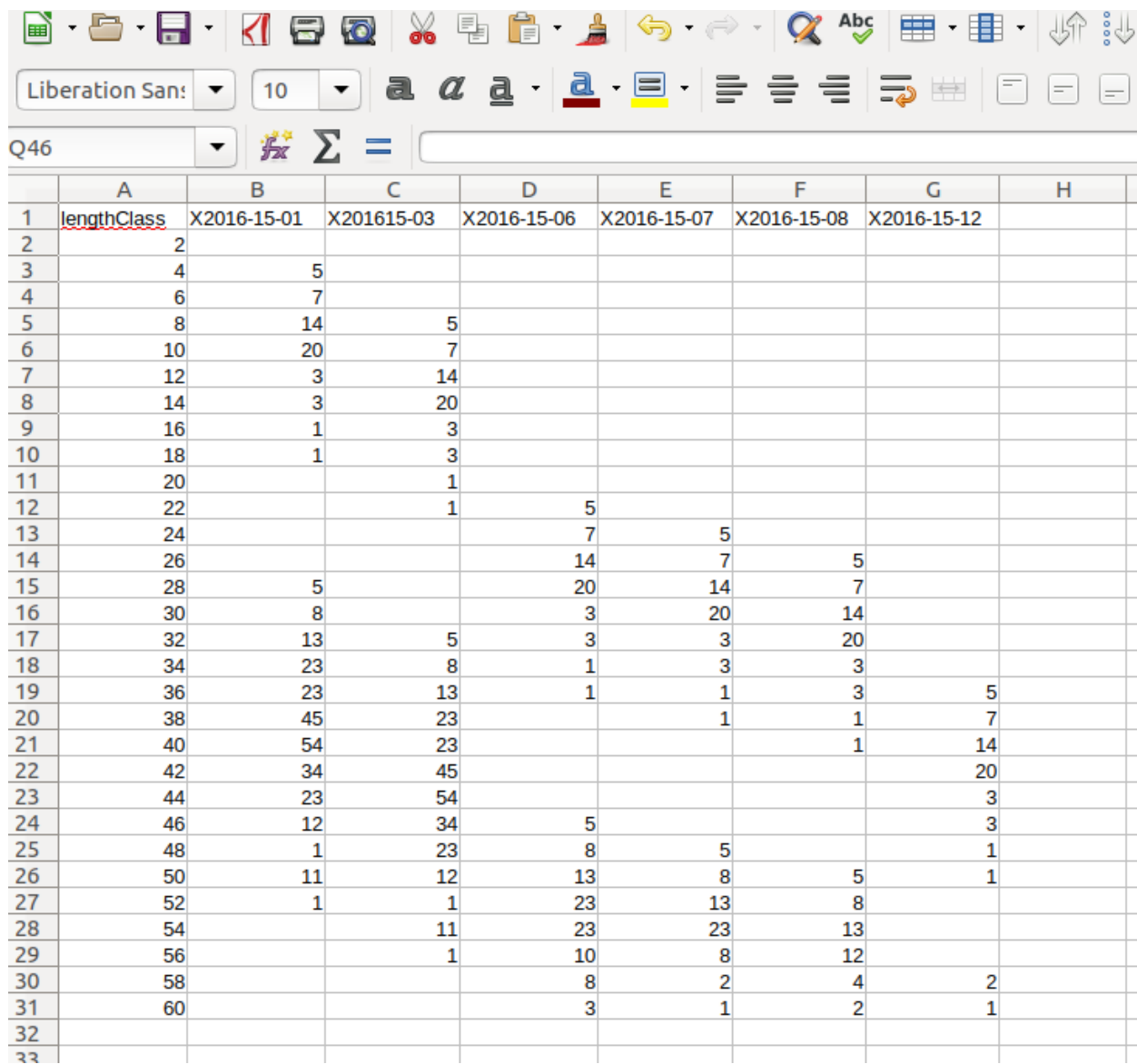
In this case, the function `lfqCreate()` can not be used, instead the data is arranged into a list manually. Extracting the dates involves extra steps to separate the actual dates from the 'X'. Next, the dates, mid lengths and the catch matrix can be directly arranged together with the function `list()`. It is of crucial importance to use the exact names and spelling (incl. case sensitivity) in the list ('dates', 'midLengths', 'catch'). with the function `class()` the TropFishR internal class 'lfq' can be assigned to the created list. Missing values in the catch matrix (NA, NaN) should be put to 0.

Other file formats Of course, your data has not to be in the csv format, it can just as well be 'odt', 'xls', 'xlsx', 'txt' or any text file. Other formats require different functions to load the data into R (beside `read.csv2()`). The following code demonstrates this for a 'xlsx' and 'txt' file.

```
lfq2a <- read.table("lfq2.txt", sep="\t", dec=',', fileEncoding="latin1", skipNul=TRUE)

install.packages("openxlsx")
require(openxlsx)
lfq3a <- read.xlsx("lfq3.xlsx", sheet = 1)
```

The function `read.table()` allows to read and load many different file formats into R, The arguments 'sep' and 'dec' control the separator between columns (here tabstop) and the decimal separator (here comma), respectively. Additional arguments like 'fileEncoding' and 'skipNul' might be necessary to use for controlling the encoding of the file and if nuls should be skipped. More information about the arguments can be viewed with `?read.table`. There are many different R packages available to read 'xls' and 'xlsx' files into R, one



	A	B	C	D	E	F	G	H
1	lengthClass	X2016-15-01	X201615-03	X2016-15-06	X2016-15-07	X2016-15-08	X2016-15-12	
2	2							
3	4	5						
4	6	7						
5	8	14	5					
6	10	20	7					
7	12	3	14					
8	14	3	20					
9	16	1	3					
10	18	1	3					
11	20		1					
12	22		1	5				
13	24			7	5			
14	26			14	7	5		
15	28	5		20	14	7		
16	30	8		3	20	14		
17	32	13	5	3	3	20		
18	34	23	8	1	3	3		
19	36	23	13	1	1	3	5	
20	38	45	23		1	1	7	
21	40	54	23			1	14	
22	42	34	45				20	
23	44	23	54				3	
24	46	12	34	5			3	
25	48	1	23	8	5		1	
26	50	11	12	13	8	5	1	
27	52	1	1	23	13	8		
28	54		11	23	23	13		
29	56		1	10	8	12		
30	58			8	2	4	2	
31	60			3	1	2	1	
32								
33								

Figure 3: Data arrangement 3. A column with the length classes and subsequent columns with the frequency of observed lengths for several sampling times.

of the fastest is `openxlsx`, but it does not allow to read in ‘xls’ files. For this you might consider using the package ‘xlsx’. These functions allow you to read in any sheet of your ‘xlsx’ file by using the ‘sheet’ argument.

#Conclusion This vignette demonstrates the first step in the application of **TropFishR**, the arrangement of your length measurements in a spreadsheet (or text file) and the import into R. Nevertheless, this step is important and several requests motivated me to write this vignette. How you can use **TropFishR** for the further analysis of your data is demonstrated in the detailed **tutorial**.

#Author’s comment I hope this short vignette is of help. Any comment or question can be addressed to me via **email** or you can post an issue at **GitHub**. You can follow the development of **TropFishR** on **ResearchGate**. All bold words are links.