I started using R in 2004. I started using R religiously on the day of the [annular solar eclipse in Madrid (October 3, 2005)](https://eclipse.gsfc.nasa.gov/SEmono/ASE2005/ASE2005.html) after being inspired by [David Hunter’s](https://stat.psu.edu/people/drh20) talk at [ADASS](http://www.adass.org/).

**It took me exactly 4,889 days to figure out what this vexing error means,** even though trial and error helped me move through it most every time it happened! I’m so moved by what I learned (and embarrassed that it took *so* long to make sense), that I have to share it with you.



This error happens when you’re trying to treat a function like a list, vector, or data frame. *To fix it, start treating the function like a function.*

Let’s take something that’s *very obviously a function.* I picked the [sampling distribution simulator from a 2015 blog post I wrote](https://qualityandinnovation.com/2015/03/30/sampling-distributions-and-central-limit-theorem-in-r/). Cut and paste it into your R console:

sdm.sim <- function(n,src.dist=NULL,param1=NULL,param2=NULL) {

r <- 10000 # Number of replications/samples - DO NOT ADJUST

# This produces a matrix of observations with

# n columns and r rows. Each row is one sample:

my.samples <- switch(src.dist,

"E" = matrix(rexp(n\*r,param1),r),

"N" = matrix(rnorm(n\*r,param1,param2),r),

"U" = matrix(runif(n\*r,param1,param2),r),

"P" = matrix(rpois(n\*r,param1),r),

"B" = matrix(rbinom(n\*r,param1,param2),r),

"G" = matrix(rgamma(n\*r,param1,param2),r),

"X" = matrix(rchisq(n\*r,param1),r),

"T" = matrix(rt(n\*r,param1),r))

all.sample.sums <- apply(my.samples,1,sum)

all.sample.means <- apply(my.samples,1,mean)

all.sample.vars <- apply(my.samples,1,var)

par(mfrow=c(2,2))

hist(my.samples[1,],col="gray",main="Distribution of One Sample")

hist(all.sample.sums,col="gray",main="Sampling Distribution\nof

the Sum")

hist(all.sample.means,col="gray",main="Sampling Distribution\nof the Mean")

hist(all.sample.vars,col="gray",main="Sampling Distribution\nof

the Variance")

}

The *right* thing to do with this function is use it to simulate a bunch of distributions and plot them using base R. You write the function name, followed by parenthesis, followed by each of the four arguments the function needs to work. This will generate a normal distribution with mean of 20 and standard deviation of 3, along with three sampling distributions, using a sample size of 100 and 10000 replications:

sdm.sim(100, src.dist="N", param1=20, param2=3)

(You should get four plots, arranged in a 2×2 grid.)

But what if we tried to treat sdm.sim like a list, and call the 3rd element of it? Or what if we tried to treat it like a data frame, and we wanted to call one of the variables in the column of the data frame?

> sdm.sim[3]

Error in sdm.sim[3] : object of type 'closure' is not subsettable

> sdm.sim$values

Error in sdm.sim$values : object of type 'closure' is not subsettable

SURPRISE! Object of type closure is not subsettable. This happens *because* sdm.sim is a function, and its data type is (shockingly) something called “closure”:

> class(sdm.sim)

[1] "function"

> typeof(sdm.sim)

[1] "closure"

I had read this before on Stack Overflow a whole bunch of times, but it never really clicked until I saw it like this! And now that I had a sense for why the error was occurring, turns out *it’s super easy to reproduce* with functions in base R or functions in *any*familiar packages you use:

> ggplot$a

Error in ggplot$a : object of type 'closure' is not subsettable

> table$a

Error in table$a : object of type 'closure' is not subsettable

> ggplot[1]

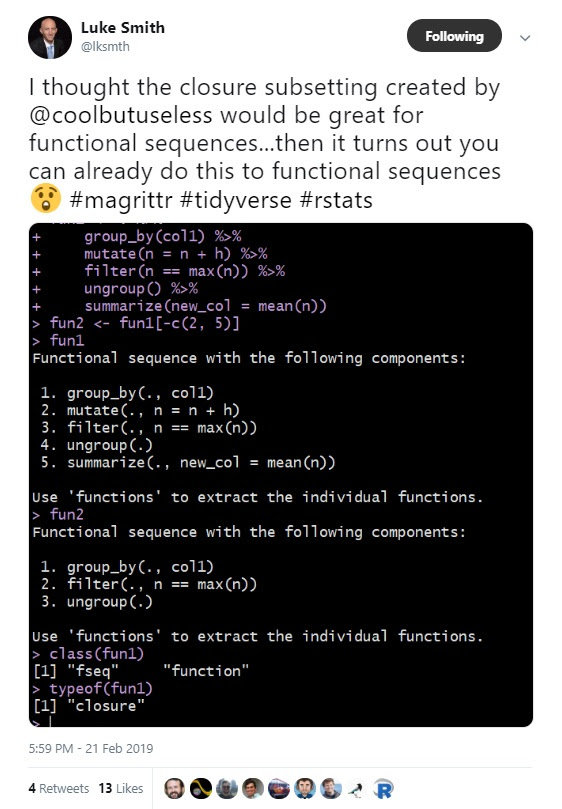
Error in ggplot[1] : object of type 'closure' is not subsettable

> table[1]

Error in table[1] : object of type 'closure' is not subsettable

As a result, if you’re pulling your hair out over this problem, check and see where in your rogue line of code you’re treating something like a non-function. Or maybe you picked a variable name Then apologize, change your notation, and move on.

But as Luke Smith pointed out, this is not true for **functional sequences** (which you can also write as functions). Functional sequences are those chains of commands you write when you’re in tidyverse mode, all strung together with %>% pipes:



Luke’s code that you can cut and paste (and try), with the top line that got cut off by Twitter, is:

fun1 <- . %>%

group\_by(col1) %>%

mutate(n=n+h) %>%

filter(n==max(n)) %>%

ungroup() %>%

summarize(new\_col=mean(n))

fun2 <- fun1[-c(2,5)]

Even though Luke’s fun1 and fun2 are of type closure, *they* ***are*** *subsettable because they contain a sequence of functions:*

> typeof(fun1)

[1] "closure"

> typeof(fun2)

[1] "closure"

> fun1[1]

Functional sequence with the following components:

1. group\_by(., col1)

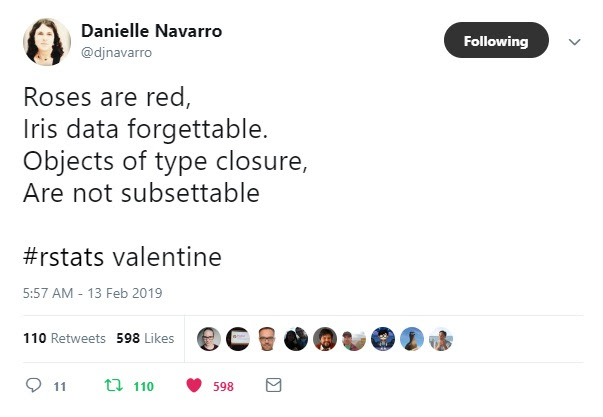
Use 'functions' to extract the individual functions.

> fun1[[1]]

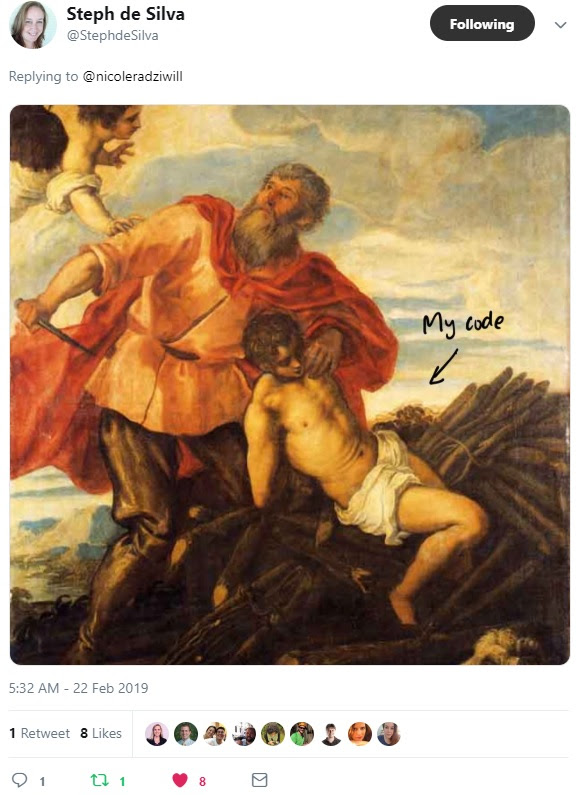
function (.)

group\_by(., col1)

Don’t feel bad! This problem has plagued all of us for many, many hours (me: years), and yet it still happens to us more often than we would like to admit. *Awareness of this issue will not prevent you from attempting things that give you this error in the future.* It’s such a popular error that there have been memes about it and sad valentines written about it:



**SCROLL DOWN PAST STEPH’S TWEET TO SEE THE JOKE!!**



(Also, if you’ve made it this far, FOLLOW THESE GOOD PEOPLE ON TWITTER: [@stephdesilva](http://twitter.com/stephdesilva) [@djnavarro](http://twitter.com/djnavarro) [@lksmth](http://twitter/com/lksmth) – They all share great information on data science in general, and R in particular. Many thanks too to all the #rstats crowd who shared in my glee last night and didn’t make me feel like an idiot for not figuring this out for ALMOST 14 YEARS. It seems so simple now.

**Steph is also a Microsoft whiz** who you should definitely hire if you need anything R+ Microsoft. Thanks to all of you!)