I think that I have learnt and forgotten, and then learnt about this feature of R a few times in the past 4 years. The idea (I think), is this:

1. R allows you to pass functions as arguments
2. Functions can be modified inside a function

So what the hell does that mean?

Well, I think I can summarise it down to this *crazy piece of magic:*

my\_fun <- function(x, fun){

fun(x)

}

Now we can pass in some input, and *any* function.

Let’s take the storms data from dplyr.

library(dplyr)

##

## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':

##

## filter, lag

## The following objects are masked from 'package:base':

##

## intersect, setdiff, setequal, union

storms

## # A tibble: 10,010 x 13

## name year month day hour lat long status category wind pressure

##

## 1 Amy 1975 6 27 0 27.5 -79 tropi… -1 25 1013

## 2 Amy 1975 6 27 6 28.5 -79 tropi… -1 25 1013

## 3 Amy 1975 6 27 12 29.5 -79 tropi… -1 25 1013

## 4 Amy 1975 6 27 18 30.5 -79 tropi… -1 25 1013

## 5 Amy 1975 6 28 0 31.5 -78.8 tropi… -1 25 1012

## 6 Amy 1975 6 28 6 32.4 -78.7 tropi… -1 25 1012

## 7 Amy 1975 6 28 12 33.3 -78 tropi… -1 25 1011

## 8 Amy 1975 6 28 18 34 -77 tropi… -1 30 1006

## 9 Amy 1975 6 29 0 34.4 -75.8 tropi… 0 35 1004

## 10 Amy 1975 6 29 6 34 -74.8 tropi… 0 40 1002

## # … with 10,000 more rows, and 2 more variables: ts\_diameter ,

## # hu\_diameter

Let’s take the mean of wind:

my\_fun(storms$wind, mean)

## [1] 53.495

And, we can also do the standard deviation, or the variance, or the median

my\_fun(storms$wind, sd)

## [1] 26.21387

my\_fun(storms$wind, var)

## [1] 687.1668

my\_fun(storms$wind, median)

## [1] 45

**Why would you want to do this?**

Let’s say you want to summarise the storms data further, for each month.

We take storms, group my month, then take the mean for month.

storms %>%

group\_by(month) %>%

summarise(wind\_summary = mean(wind))

## # A tibble: 10 x 2

## month wind\_summary

##

## 1 1 45.7

## 2 4 44.6

## 3 5 36.3

## 4 6 37.8

## 5 7 41.2

## 6 8 52.1

## 7 9 58.0

## 8 10 54.6

## 9 11 52.5

## 10 12 47.9

You could repeat the code again you could vary mean to be, say sd

storms %>%

group\_by(month) %>%

summarise(wind\_summary = sd(wind))

## # A tibble: 10 x 2

## month wind\_summary

##

## 1 1 9.08

## 2 4 5.94

## 3 5 9.57

## 4 6 13.4

## 5 7 19.1

## 6 8 26.0

## 7 9 28.2

## 8 10 25.3

## 9 11 22.0

## 10 12 14.6

Over the years, every time I repeat some code like this, I have felt a tug somewhere in my brain – a little spidey sense saying (something like): “Don’t repeat yourself, Nick”.

We can avoid repeating ourselves by using the template from earlier here in dplyr. We want to manipulate the summary (mean) used – so you could also take the median, variance, etc.

We can write the following:

storms\_wind\_summary <- function(fun){

storms %>%

group\_by(month) %>%

summarise(wind\_summary = fun(wind))

}

And now we can pass the function name, say, mean.

storms\_wind\_summary(mean)

## # A tibble: 10 x 2

## month wind\_summary

##

## 1 1 45.7

## 2 4 44.6

## 3 5 36.3

## 4 6 37.8

## 5 7 41.2

## 6 8 52.1

## 7 9 58.0

## 8 10 54.6

## 9 11 52.5

## 10 12 47.9

Or, any other function!

storms\_wind\_summary(sd)

## # A tibble: 10 x 2

## month wind\_summary

##

## 1 1 9.08

## 2 4 5.94

## 3 5 9.57

## 4 6 13.4

## 5 7 19.1

## 6 8 26.0

## 7 9 28.2

## 8 10 25.3

## 9 11 22.0

## 10 12 14.6

storms\_wind\_summary(var)

## # A tibble: 10 x 2

## month wind\_summary

##

## 1 1 82.5

## 2 4 35.3

## 3 5 91.5

## 4 6 180.

## 5 7 365.

## 6 8 678.

## 7 9 793.

## 8 10 638.

## 9 11 482.

## 10 12 213.

storms\_wind\_summary(median)

## # A tibble: 10 x 2

## month wind\_summary

##

## 1 1 50

## 2 4 45

## 3 5 35

## 4 6 35

## 5 7 37.5

## 6 8 45

## 7 9 50

## 8 10 50

## 9 11 50

## 10 12 45

We could even make our own!

range\_diff <- function(x){

diff(range(x))

}

storms\_wind\_summary(range\_diff)

## # A tibble: 10 x 2

## month wind\_summary

##

## 1 1 25

## 2 4 15

## 3 5 35

## 4 6 70

## 5 7 130

## 6 8 140

## 7 9 145

## 8 10 145

## 9 11 120

## 10 12 50

Looks like there was a pretty huge range in July through to November!

Pretty neat, eh? You can manipulate the function itself!

**Going slightly further**

The above was an example demonstrating *how* you can manipulate a function being passed.

But, there are other ways to do this with dplyr that I might use instead.  
We could use summarise\_at here, to specify a function in a different, equivalent, way.

storms\_wind\_summary <- function(fun){

storms %>%

group\_by(month) %>%

summarise\_at(.vars = vars(wind),

.funs = list(fun))

}

storms\_wind\_summary(mean)

## # A tibble: 10 x 2

## month wind

##

## 1 1 45.7

## 2 4 44.6

## 3 5 36.3

## 4 6 37.8

## 5 7 41.2

## 6 8 52.1

## 7 9 58.0

## 8 10 54.6

## 9 11 52.5

## 10 12 47.9

storms\_wind\_summary(median)

## # A tibble: 10 x 2

## month wind

##

## 1 1 50

## 2 4 45

## 3 5 35

## 4 6 35

## 5 7 37.5

## 6 8 45

## 7 9 50

## 8 10 50

## 9 11 50

## 10 12 45

What if we want to provide many functions? Say, the mean, median, sd, variance, all together, how they belong?

We can do this.

This is done by passing dots (ellipsis) ... to the function. This allows for any number of inputs.

storms\_wind\_summary <- function(...){

storms %>%

group\_by(month) %>%

summarise\_at(.vars = vars(wind),

.funs = list(...))

}

storms\_wind\_summary(median, mean, max)

## # A tibble: 10 x 4

## month fn1 fn2 fn3

##

## 1 1 50 45.7 55

## 2 4 45 44.6 50

## 3 5 35 36.3 60

## 4 6 35 37.8 80

## 5 7 37.5 41.2 140

## 6 8 45 52.1 150

## 7 9 50 58.0 160

## 8 10 50 54.6 160

## 9 11 50 52.5 135

## 10 12 45 47.9 75

**What’s the point of this?**

So, this might not be the most useful summary of the storms data…and writing functions like this might not be the most general usecase. dplyr provides some amazingly flexible syntax to summarise data. Sometimes the answer isn’t writing a function, and you want to be mindful of replicating the flexibility of dplyr itself.

That said, with a task like this, or any section of code, I really think it can be useful to wrap them in a function, which *describes more broadly what that section does*. And, with features like what I wrote about here, I think that you can more clearly and flexible wrap up these features for your own use.

R is flexible enough to make that quite straightforward, and I think that is pretty darn neat!

**Fin**

Go forth, and use the power of functions in your work!

**PS**

Upon reflection, I’m pretty sure [Mitchell O’Hara-Wild](https://www.njtierney.com/post/2019/09/29/unexpected-function/mitchoharawild?) was the one who helped really solidify this into my brain. Thanks, Mitch!