FishR 101 – Functions

Homework

Instructor: Emily Markowitz (Emily.Markowitz@noaa.gov)

February 08, 2021

Contents

Questions:

First, load your libraries!

```
library(tidyverse)
library(here)
```

NOTE: For this homework, also refer to the PDF version. There are helpful graphics included in the PDF version!

1 If-else statements

1.1 Performance: was this a good trawl?

If the weather is `excellent` or `good`, the peformance of the trawl is `0` if the weather is `fair`, the peformance of the trawl is `1` if the weather is `poor`, the peformance of the trawl is `-1`

Write the following as an:

- if() / if()}else{} / if()}else if(){} statement (whichever you see fit),
- in an ifelse() function,
- using dplyr::if_else(), and
- using dplyr::case_when()

Test your scripts whith `weather <- "good"` and solve for the parameter.

Which do you think is the most sensible approach?

> Hint: dplyr depends (and ifelse() can use)) on data.frames (and other types of tables) so you might use this data.frame:

```
dat<-data.frame(weather = c("excellent", "good", "fair", "poor"))
```

```
## [1] 0
```

```
## [1] 0
```

```
## [1] 0
```

```
## [1] 0
```

```
## [1]  0  0  1 -1
```

weather	performance
excellent	0
good	0
fair	1
poor	-1

weather	performance
excellent	0
good	0
fair	1
poor	-1

weather	performance
excellent	0
good	0
fair	1
poor	-1

2 For loops

2.1 Improve the following code by putting it into a for loop!

Let's use a for loop to estimate the average of squaring the result of a roll of a die. Where...

```
# Let's use a for loop to estimate the average the result of a roll of a die.
nsides = 6
ntrials = 1000
```

A non-loop version of this for the first variable would be:

```
trials <- c()
j <- 1
trials <- c(trials, sample(1:nsides,1))
trials
```

```
## [1] 3
```

```
# once you write your loop, you can use the following to calcuale the average the result of a roll of a
mean(trials) # NOTE: because we are taking a random sample (sample()) you will not get the same answer
```

```
## [1] 3
```

```
## [1] 3 3 2 3 2 3
```

```
## [1] 3.45954
```

3 Functions

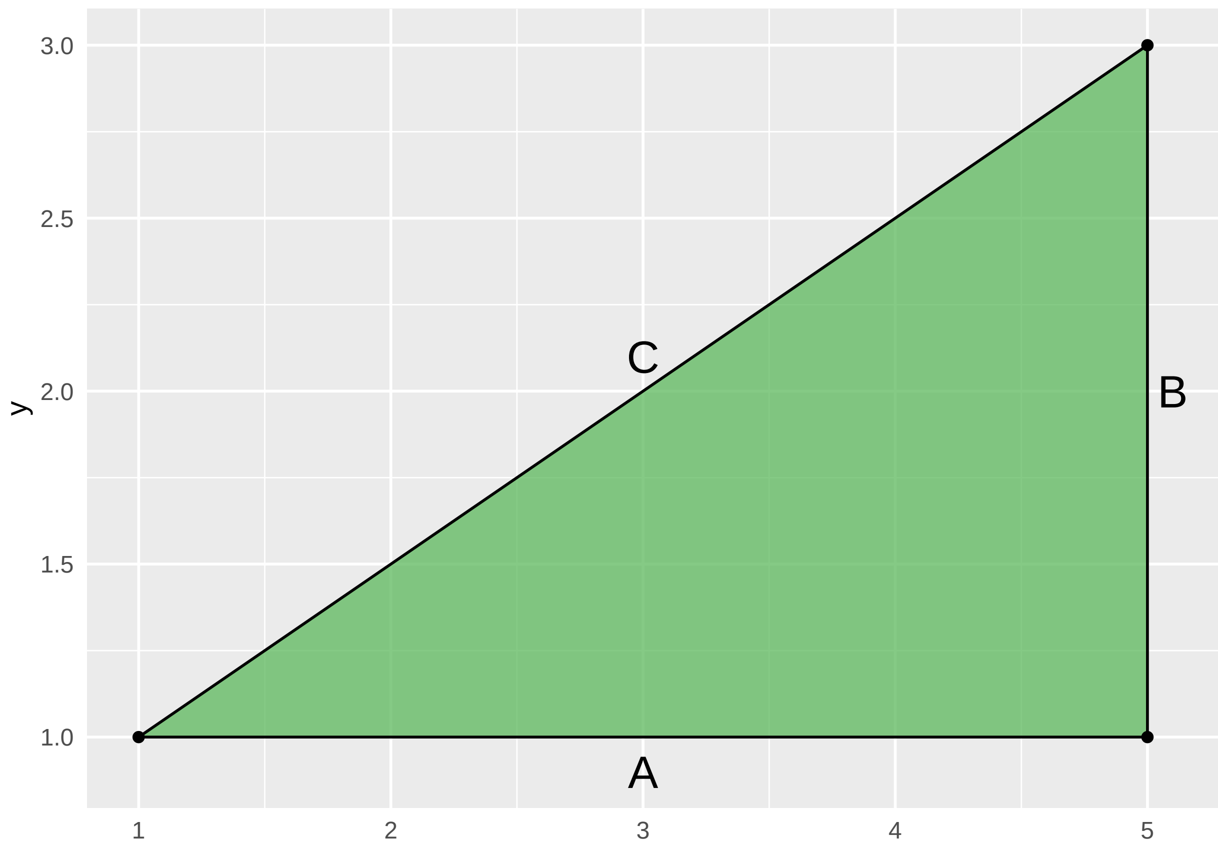
Though you can use funtions to automate processes like we did in the coursework, I am going to take a slightly different take here and focus of using functions for, well, mathimatical functions!

3.1 The Pythagorean Theorem

$$a^2 + b^2 = c^2$$

Where:

- a is the length of leg A
- b is leg length of leg B
- c is the length of leg C, otherwise known as the hypotenuse



If $a = 5$ and $b = 3$, solve for c and include {roxygen2} skeletons!

```
## [1] 5.830952
```

3.2 Newton's Universal Law of Gravitation

$$F = G\frac{m_1 m_2}{d^2}$$

Where:

- F = force
- G = gravitational constant $(6.67430 * 10^{-11})$
- m_1 = mass of object 1
- m_2 = mass of object 2

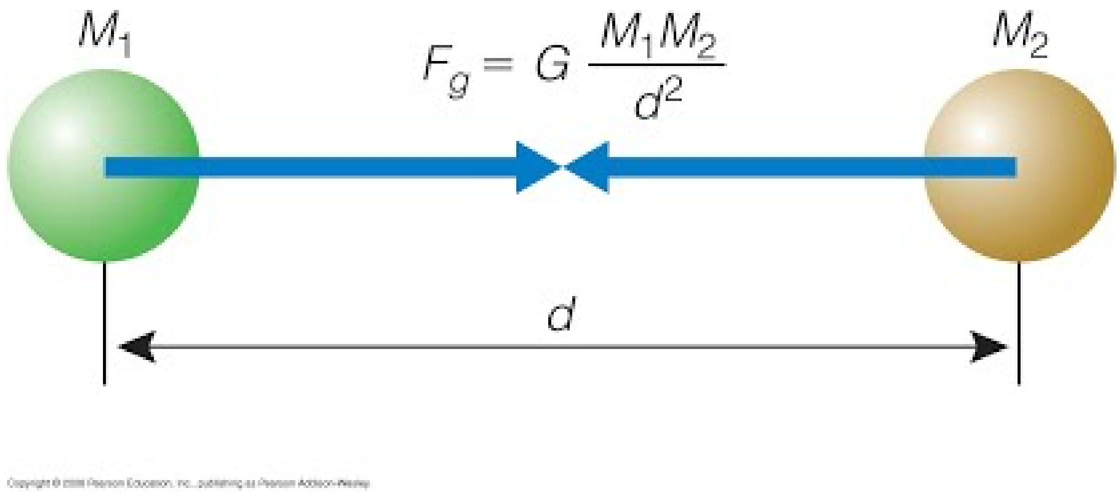


Figure 1: Newton's Universal Law of Gravitation

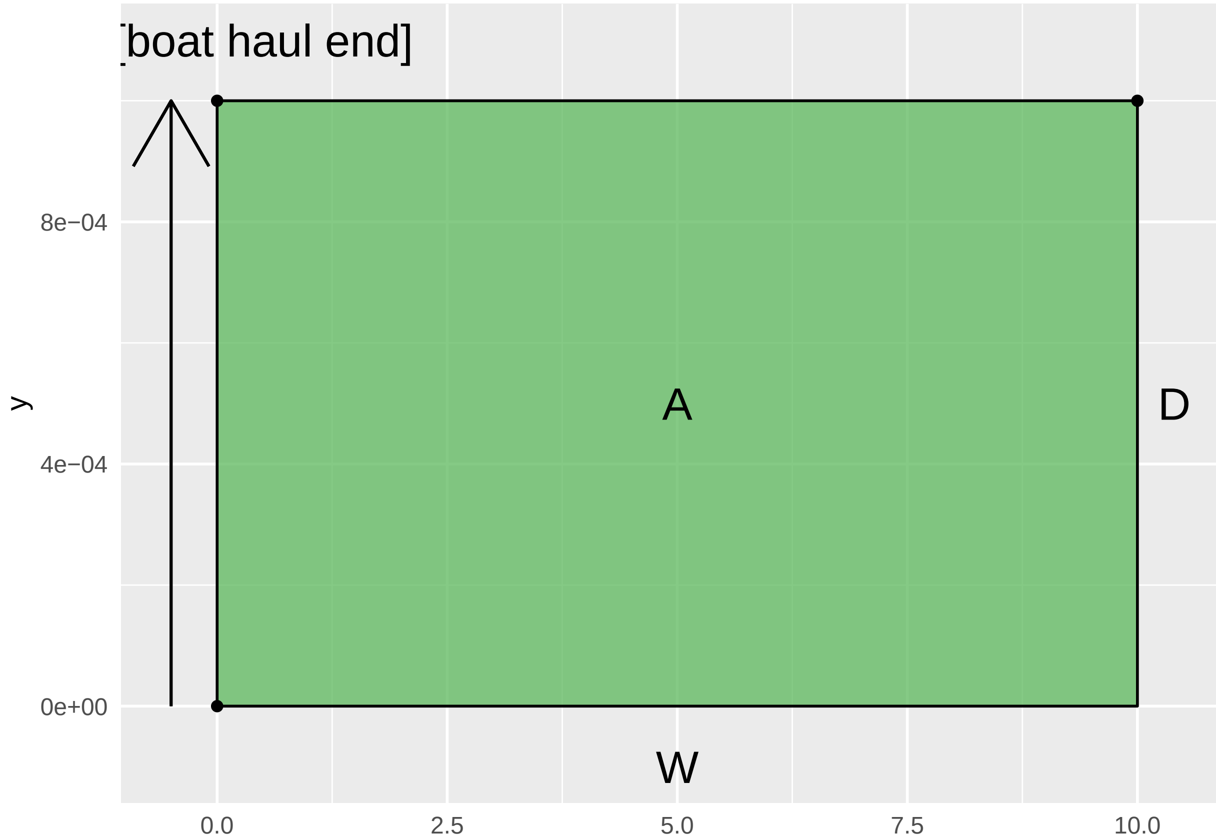
- r = distance between centers of the masses

Let the mass of object 1 (m_1) = 5, mass of object 2 (m_2) = 3, and distance between the two masses (d) = 2. Solve for force (F) and include {roxygen2} skeletons:

> Hint! The Gravitational Constant Shouldn't change, unless you are testing this out on other planets. To save time, add the gravitational constant to where you call your arguments in your function.

```
## [1] 5.005725e-10
```

3.3 Area Swept (from our surveys!)



This is the area sampled for each observation when we survey. Often we can assume this number is low (e.g., 0.001). In our case, we need to add some conversions to make this useful to the survey outputs. Here I'll write without variables to make it easier to read:

$$AreaSwept(km^2) = DistanceFished(hectare) * (NetWidth(m) * 0.001(\frac{km}{m})) * 100(\frac{km}{hectare})$$

But for writing our function, we'll simplify to the core of this equation:

$$area = distance * width$$

3.3.1 Write a function for this equation and solve for $AreaSwept(km^2)$. Use a0 = Area, d0 = Distance (.001), w0 = Width (10). Solve for Area (a0) and include {roxygen2} skeletons.

```
## [1] 0.001
```

3.3.2 Now we will apply this function to some real data! Here I've combined two datasets, *haul* and *catch* for Eastern Bering Sea data.

Here is *catch* and *haul* joined:

```r
EBS<-read.csv(here("data", "haul_catch.csv"))
EBS$X<-NULL
EBS <- subset(x = EBS,
              subset = (YEAR == 2017),
              select = c("YEAR", "SPECIES_CODE", "WEIGHT", "NUMBER_FISH",
                         "REGION", "VESSEL", "HAUL",
                         "STRATUM", "PERFORMANCE", "START_TIME",
                         "DURATION", "DISTANCE_FISHED", "NET_WIDTH", "NET_HEIGHT"))
kable(head(EBS))
```

	YEAR	SPECIES_CODE	WEIGHT	NUMBER_FISH	REGION	VESSEL	HAUL	STRATUM	PEF
53757	2017	471	129.200	25	BS	162	126	62	
53758	2017	21390	0.060	1	BS	162	127	43	
53759	2017	10285	96.964	99	BS	162	25	31	
53760	2017	10112	4.600	19	BS	162	30	50	
53761	2017	10120	20.972	20	BS	162	6	10	
53762	2017	21368	2.000	1	BS	162	42	31	

```
##       YEAR       SPECIES_CODE        WEIGHT          NUMBER_FISH         REGION              VESSEL
##  Min.   :2017   Min.   :  320   Min.   :   0.002   Min.   :    1.0   Length:4698         Min.   : 94.(
##  1st Qu.:2017   1st Qu.:10120   1st Qu.:   3.741   1st Qu.:    4.0   Class :character   1st Qu.: 94.(
##  Median :2017   Median :10270   Median :  19.225   Median :   17.0   Mode  :character   Median : 94.(
##  Mean   :2017   Mean   :18154   Mean   : 105.054   Mean   :  318.6                      Mean   :123.9
##  3rd Qu.:2017   3rd Qu.:21438   3rd Qu.:  77.986   3rd Qu.:  134.2                      3rd Qu.:162.(
##  Max.   :2017   Max.   :81742   Max.   :6699.338   Max.   :28160.0                      Max.   :162.(
##                                                    NA's   :2
##   PERFORMANCE        START_TIME            DURATION        DISTANCE_FISHED   NET_WIDTH        NET_HEIGHT
##  Min.   :0.00000   Length:4698         Min.   :0.2470   Min.   :1.337   Min.   :13.61   Min.   :1.302
##  1st Qu.:0.00000   Class :character   1st Qu.:0.5120   1st Qu.:2.768   1st Qu.:16.31   1st Qu.:1.955
##  Median :0.00000   Mode  :character   Median :0.5180   Median :2.827   Median :16.92   Median :2.135
##  Mean   :0.07264                      Mean   :0.5144   Mean   :2.815   Mean   :16.97   Mean   :2.129
##  3rd Qu.:0.00000                      3rd Qu.:0.5240   3rd Qu.:2.897   3rd Qu.:17.62   3rd Qu.:2.337
##  Max.   :4.50000                      Max.   :0.6090   Max.   :3.439   Max.   :20.12   Max.   :2.962
##
```

3.4 Catch Per Unit Effort (CPUE)

CPUE is calculated by dividing the catch of each fishing trip by the number of hours fished during that trip. This gives CPUE in units of kilograms per hour.

$$CPUE = \frac{Catch_{trips}(kg)}{time_{trips}(hr)}$$

3.4.1 Write a function for this equation and solve for $CPUE$. Use catch = catch from the survey (catch = EBS$NUMBER_{F}ISH), time = timeinhoursfromthesurvey(EBS$DURATION), trips = number of trips taken during survey (EBS$HAUL). Solve for CPUE (CPUE) and include {roxygen2} skeleton.

> Hint: This question is meant to challenge you. You'll need to use an for loop to cycle through unique trips.

TRIP	CPUE
126	239.4721
127	277.7416
25	816.5289
30	795.5340
6	785.0103
42	1642.2331