Basic programming in R

EVR 628- Intro to Environmental Data Science

Juan Carlos Villaseñor-Derbez (JC)

Table of contents

Part 0: Set up

1. Go to the WCPFC website here.
2. Towards the bottom of the page, look for a data set called `Aggregated data, grouped by 1°x1° latitude/longitude grids, YEAR and MONTH.`
3. Click on green link (should say 3.33 MB) to download the data to your `data/raw` folder in your EVR 628 project
4. You will have downloaded a ziped folder called `WCPFC_S_PUBLIC_BY_1x1_MM_5.zip`. Unzip it.
5. Navigate to `data/raw/WCPFC_S_PUBLIC_BY_1x1_MM_5` and open the pdf called `purse_seine_by_mon_1x1.pdf`.
6. What does the metadata say about `LAT_short` abd `LON_short`?

Part 1: Functions

Exercise 1: A function that cleans latitude / longitude data

For now, you will work with just one of the 12 data sets available in the WCPFC website. But if you had to work with all of them, you might want to build a function that deals with the

1. Start a new R script called `functions_and_loops.R`, and save it to your `scripts/processing` folder
2. Copy the code below to your new script (you can use the clipboard icon on the top right of the code chunk)
3. Run everything to make sure it works. Take some time to inspect the `wcpfc` and the `wcpfc_clean` items.
4. Look at the code. Does it look like some code is repeating? (Again. Imagine you will have to do this for other 11 data sets. And some of those will have a resolution of 5 vs 1 degree.)

```r
# SET UP ####################################################################

## Load packages ------------------------------------------------------------
library(tidyverse)
library(EVR628tools)

## Load data ----------------------------------------------------------------
wcpfc <- read_csv("data/raw/WCPFC_S_PUBLIC_BY_1x1_MM_5/WCPFC_S_PUBLIC_BY_1x1_MM.CSV")

# PROCESSING ################################################################

## Clean data ---------------------------------------------------------------
wcpfc_clean <- wcpfc |>
  select(year = yy,
         lat = lat_short,
         lon = lon_short,
         days) |>
  # WCPFC reports "the latitude of the south-west corner"
  # We need to fix that
  mutate(
    # Find if it's in the S or W hemisphere
    lat_mult = ifelse(str_detect(lat, "N"), 1, -1),
    lon_mult = ifelse(str_detect(lon, "E"), 1, -1),
    # Find if it's in the S or W hemisphere
    lat = as.numeric(str_remove_all(lat, "[:alpha:]")),
    lon = as.numeric(str_remove_all(lon, "[:alpha:]")),
    # Find if it's in the S or W hemisphere
    lat = lat_mult * lat + 0.5,
    lon = lat_mult * lon + 0.5) |>
  select(-lat_mult, -lon_mult)

wcpfc_clean
```

```
# A tibble: 193,063 x 4
    year   lat   lon  days
   <dbl> <dbl> <dbl> <dbl>
 1  1967   0.5  140.     0
 2  1967   0.5  142.     0
 3  1967  -0.5 -140.     0
 4  1967  -0.5 -142.     0
 5  1967   3.5  136.     0
 6  1967   7.5  134.     0
 7  1967   8.5  134.     0
 8  1968   0.5  140.     0
 9  1968   0.5  144.     0
10  1968  -0.5 -140.     0
# i 193,053 more rows
```

3. Let's identify the general structure:
    a. Step 1: What are we repeating?
    b. Step 2: Within that code, what is constant and what changes?
4. Let's build a function called `clean_coords()`

```
clean_coords <- function(x, res) {
  mult <- ifelse(str_detect(x, "N|E"), 1, -1)
  coord <- as.numeric(str_remove_all(x, "[:alpha:]"))
  out <- (mult * coord) + (res / 2)
  return(out)
}
```

5. Let's test the function

```
# These are equivalent to 10, -10, 10, -10 for the bottom-left corner
test_coord <- c("10N", "10S", "10E", "10W")

# Once centered, they should be 10.5, -9.5, 10.5, -9.5
clean_coords(test_coord, res = 1)
```

```
[1] 10.5 -9.5 10.5 -9.5
```

6. After confirming they perform, update the code to use our function

```r
# SET UP ########################################################################

## Load packages ----------------------------------------------------------------
library(tidyverse)
library(EVR628tools)

## User defined functions -------------------------------------------------------
clean_coords <- function(x, res) {
  mult <- ifelse(str_detect(x, "N|E"), 1, -1)
  coord <- as.numeric(str_remove_all(x, "[:alpha:]"))
  out <- (mult * coord) + (res / 2)
  return(out)
}

## Load data --------------------------------------------------------------------
wcpfc <- read_csv("data/raw/WCPFC_S_PUBLIC_BY_1x1_MM_5/WCPFC_S_PUBLIC_BY_1x1_MM.CSV")

# PROCESSING ####################################################################

## Clean data --------------------------------------------------------------------
wcpfc_clean <- wcpfc |>
  select(year = yy,
         lat = lat_short,
         lon = lon_short,
         days) |>
  # WCPFC reports "the latitude of the south-west corner"
  # We need to fix that
  mutate(
    lat = clean_coords(lat, 1),
    lon = clean_coords(lon, 1))

wcpfc_clean
```

```
# A tibble: 193,063 x 4
    year   lat   lon  days
   <dbl> <dbl> <dbl> <dbl>
 1  1967   0.5  140.     0
 2  1967   0.5  142.     0
 3  1967  -0.5  142.     0
 4  1967  -0.5  142.     0
 5  1967   3.5  136.     0
 6  1967   7.5  134.     0
 7  1967   8.5  134.     0
 8  1968   0.5  140.     0
 9  1968   0.5  144.     0
10  1968  -0.5  142.     0
# i 193,053 more rows
```

Exercise 2: A function that builds a plot

You are building 10 monographs for 10 different dive sites where lionfish are found. People are interested in knowing "how big are the lionfish in dive site X?".

1. Using the `data_lionfish` data, build a histogram of lionfish length (`total_length_m`) for all fish sampled from `site == Paraiso`. Don't worry about themes and labels for now.
2. Using the `data_lionfish` data, build a histogram of lionfish length (`total_length_m`) for all fish sampled from `site == Canones`. Don't worry about themes and labels for now.
3. Using the `data_lionfish` data, build a histogram of lionfish length (`total_length_m`) for all fish sampled from `site == Paamul`. Don't worry about themes and labels for now.
4. I think you see where I'm going. We can't use `facet_wrap` or `facet_grid` because each dive site will have it's own plot.
5. Let's identify the general structure:

    a. Step 1: What are we repeating?
    b. Step 2: Within that code, what is constant and what changes?

6. Build a new function called `site_histogram()`

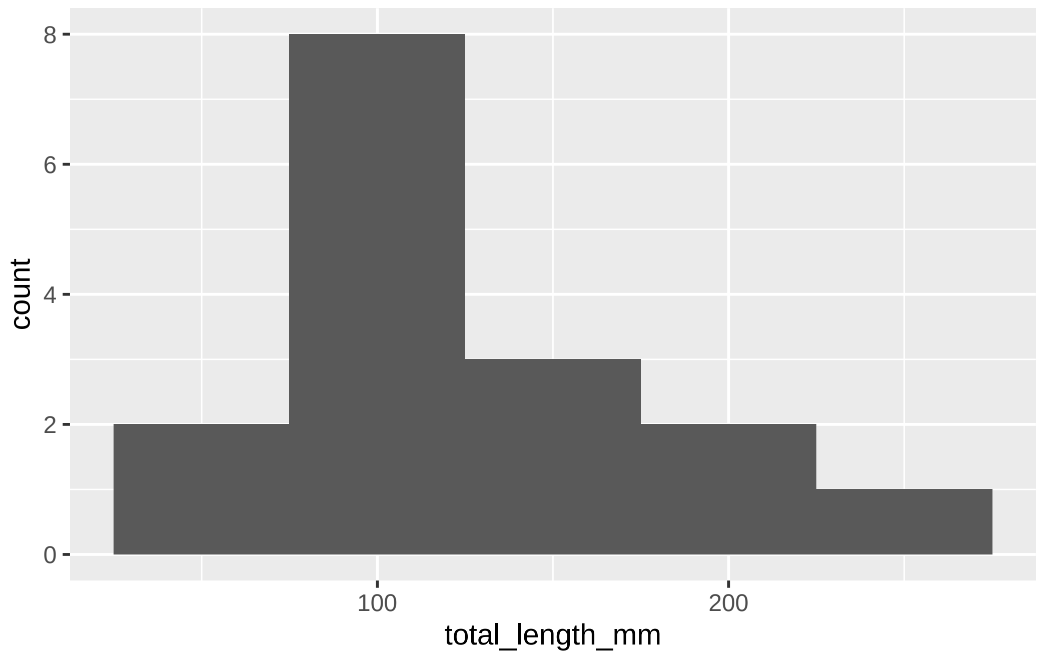
```r
data("data_lionfish")

site_histogram <- function(data, my_site) {
  # Filter my data
  inside_data <- data |>
    filter(site == my_site)

  # Build my plot
  p <- ggplot(inside_data,
              mapping = aes(x = total_length_mm)) +
    geom_histogram(binwidth = 50)

  # Return my plot
  return(p)
}

site_histogram(data = data_lionfish,
               my_site = "Paraiso")
```



Part 2: `foor` loops

We'll do this one together.

1. Build a for loop that:

    a. calls your `site_histogram()` function, and then
    b. saves it out as `results/img/<site_name>.png`

```
for(site_i in unique(data_lionfish$site)) {
  # Build plot with my function
  p <- site_histogram(data = data_lionfish,
```

```r
                      my_site = site_i)

  # Save the ploit
  ggsave(plot = p,
         filename = paste("results/img/", site_i, ".png"),
         width = 6,
         height = 4)

  # End of item i
}
```