Chapter1

The Caveman Coder

2025-10-26

My first R script! Lines starting with # are comments - R ignores them. They are notes for humans!

Create a variable (a named box) to store a number

```r
my_favorite_number <- 7
```

Create a variable to store some text (put text in quotes!)

```r
my_city <- "Amsterdam"
```

Print the contents of the variables to the Console

```r
print(my_favorite_number)
```

```
## [1] 7
```

```r
print(my_city)
```

```
## [1] "Amsterdam"
```

Do a simple calculation

```r
result <- my_favorite_number * 3
print(result)
```

```
## [1] 21
```

Installing and Loading Essential Packages with pacman

Step 1: Install 'pacman' itself if you don't have it We use install.packages() for this one time. 'requireNamespace' checks if it's installed without loading it yet. The '!' means NOT. So, "if NOT requireNamespace pacman…"

```r
if (!requireNamespace("pacman", quietly = TRUE)) {
  install.packages("pacman")
}
```

Step 2: Load pacman using library() so we can use its functions

```r
library(pacman)
```

Step 3: use pacman::p_load() to install (if needed) and load our core mapping packages. This is the command you'll use often at the start of your scripts!

```r
print("Loading core packages (we will install if missing)...")
```

```
## [1] "Loading core packages (we will install if missing)..."
```

```r
p_load(
  sf,           # Core package for vector spatial data
  tidyverse,    # Collection including ggplot2, dplyr, readr, etc.
```

```r
  terra         # Core package for raster spatial data
)

print("Core packages sf, tidyverse, and terra are ready!")
```

```
## [1] "Core packages sf, tidyverse, and terra are ready!"
```

You might see lots of messages if packages are being installed for the first time. This is normal! It might take a few minutes. You need an internet connection.

Quick CRS Introduction Exercise

Step 1: Ensure sf is loaded

```r
pacman::p_load(sf, rnaturalearth) # Need rnaturalearth for the data
```

Step 2: Get world map data as an sf object

```r
print("Getting world map data...")
```

```
## [1] "Getting world map data..."
```

```r
world_sf <- rnaturalearth::ne_countries(
  scale = "medium", returnclass = "sf"
)
print("World data loaded.")
```

```
## [1] "World data loaded."
```

Step 3: Check the CRS of this data!

```r
print("Checking the original CRS...")
```

```
## [1] "Checking the original CRS..."
```

```r
original_crs <- sf::st_crs(world_sf)
print(original_crs)
```

```
## Coordinate Reference System:
##   User input: WGS 84
##   wkt:
## GEOGCRS["WGS 84",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["latitude",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["longitude",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     ID["EPSG",4326]]
```

Look for the EPSG code near the bottom! Should be 4326 (WGS84 Lat/Lon)

Step 4: Define a different target CRS (e.g., Robinson Projection) Robinson is often used for world maps. We use its "ESRI" code here.

```r
target_crs_robinson <- "ESRI:54030"
print(paste("Target CRS:", target_crs_robinson))
```

```
## [1] "Target CRS: ESRI:54030"
```

Step 5: Transform the data to the new CRS using st_transform()

```r
print("Transforming data to Robinson projection...")
```

```
## [1] "Transforming data to Robinson projection..."
```

```r
# The |> pipe sends world_sf to the st_transform function
world_robinson_sf <- world_sf |>
  sf::st_transform(crs = target_crs_robinson)
print("Transformation complete!")
```

```
## [1] "Transformation complete!"
```

Step 6: Check the CRS of the NEW, transformed data

```r
print("Checking the NEW CRS...")
```

```
## [1] "Checking the NEW CRS..."
```

```r
new_crs <- sf::st_crs(world_robinson_sf)
print(new_crs)
```

```
## Coordinate Reference System:
##   User input: ESRI:54030
##   wkt:
## PROJCRS["World_Robinson",
##     BASEGEOGCRS["WGS 84",
##         DATUM["World Geodetic System 1984",
##             ELLIPSOID["WGS 84",6378137,298.257223563,
##                 LENGTHUNIT["metre",1]]],
##         PRIMEM["Greenwich",0,
##             ANGLEUNIT["Degree",0.0174532925199433]]],
##     CONVERSION["World_Robinson",
##         METHOD["Robinson"],
##         PARAMETER["Longitude of natural origin",0,
##             ANGLEUNIT["Degree",0.0174532925199433],
##             ID["EPSG",8802]],
##         PARAMETER["False easting",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8807]]],
##     CS[Cartesian,2],
##         AXIS["(E)",east,
##             ORDER[1],
##             LENGTHUNIT["metre",1]],
##         AXIS["(N)",north,
##             ORDER[2],
##             LENGTHUNIT["metre",1]],
##     USAGE[
##         SCOPE["Not known."],
##         AREA["World."],
```

```
##         BBOX[-90,-180,90,180]],
##     ID["ESRI",54030]]
```

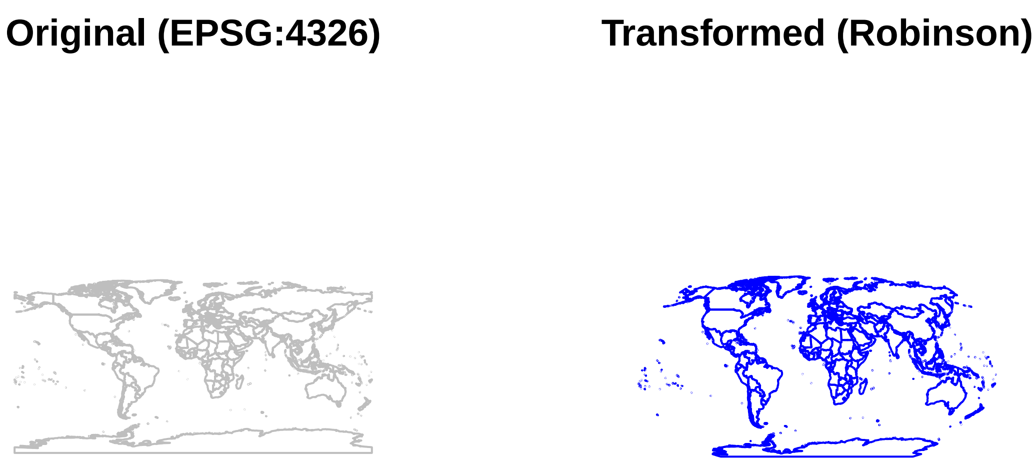
Notice the name and details are different now! It's no longer EPSG:4326.

Step 7: Quick plot comparison (using base plot for simplicity here)

```r
print( "Plotting comparison (Original vs. Transformed)..." )
```

```
## [1] "Plotting comparison (Original vs. Transformed)..."
```

```r
par(mfrow = c(1, 2)) # Arrange plots side-by-side
plot(
  sf::st_geometry(world_sf),
  main = "Original (EPSG:4326)",
  key.pos = NULL, reset = FALSE,
  border ="grey"
)
plot(
  sf::st_geometry(world_robinson_sf),
  main = "Transformed (Robinson)",
  key.pos = NULL, reset = FALSE,
  border = "blue"
)
```




```r
par(mfrow = c(1, 1)) # Reset plot layout
print("Plots displayed. Notice the shape difference!")
```

```
## [1] "Plots displayed. Notice the shape difference!"
```