Stat 243: Problem Set 3, Due Wednesday Oct-05

September 21, 2016

Instructions

Please turn in (1) a copy on paper, as this makes it easier for us to handle AND (2) an electronic copy through bCourses so we can run your code if needed.

You may need to learn some R functionality that you don't know and we haven't covered yet. Use piazza to your advantage to get suggestions, and chat with me or the GSI.

All your work should be done using either shell commands or R code (depending on the specifications of each problem) that you save in a file and turn in as part of your solution to the assignment. So you can't say "I downloaded the data from such-and-such website" or "I clicked the link of the file ..."; you need to provide us the code that we could run to repeat what you did.

You can use an `.R` (R script) file, or an `.Rmd` (Rmarkdown) file for your source code.

Introduction

This assignment covers data manipulation, and provides practice in downloading and manipulating data using some of the data importing functions in R. We'll use data sets from the *International Best Track Archive for Climate Stewardship* (IBTrACS) website which provides tropical cyclone best track data worldwide.

We'll be working with IBTrACS-WMO data files. A sample of what this format looks like is available in the following link:

https://www.ncdc.noaa.gov/ibtracs/index.php?name=wmo-samples#wmo

In particular, we'll be using the data files in WMO standard format, organized by year:

ftp://eclipse.ncdc.noaa.gov/pub/ibtracs/v03r08/wmo/wmo_format/year

The WMO data files are in a fixed-width format. The description of the fields (i.e. the columns) in this format is in the following pdf file:

ftp://eclipse.ncdc.noaa.gov/pub/ibtracs/documents/Best-Track_WMO-format_Revised2002.pdf

When I was importing and processing the WMO files for the first time, I found a couple of inconsistent issues. After re-reading the description of the WMO format, I noticed two details that are not well specified in such description:

- Position **1-9** refers to the cyclone identification code: e.g. `01 SI2005`. This code is actually composed by 1) 2 digit numbers in order within the cyclone season, followed by 2) a blank space, followed by 3) two letters indicating the basin code, followed by 4) the year code. The basin codes are:
  - `EP`: east pacific ocean
  - `NA`: north atlantic ocean

  - `NI`: north indian ocean
  - `SA`: south pacific ocean
  - `SI`: south indian ocean
  - `SP`: south pacific ocean
  - `WP`: west pacific ocean

- Position **30** is the latitude indicator.

Problem 1

The goal of this first problem is to write a function, say `read_wmo()`, which takes a given year (e.g. 2005), and returns a *processed* data frame with all the variables described in the pdf of the WMO format.

Storm Names. The names of the storms should not contain trailing spaces. For example, the names of the first storm in 2005 is `"PHOEBE      "`, which has 4 spaces after the last `E`. The processed name should be `"PHOEBE"`.

Missing Values. Depending on the field, missing values—referred to as `"no report"` or `"unknown"` in the WMO format description—are codified in different ways. Your produced data frame should reflect those values as `NA`.

Categorical Variables. Your data frame should show category names of those categorical fields instead of numbers. For example, the value at position 36 represents *Longitude indicator*, which can take values 1 (west longitude) or 2 (east longitude). The produced data frame should show something like `"west"` or `"east"` instead of just 1 or 2.

Longitude and Latitude. The processed longitude values should range from 0 to 180 for `east` longitude, and negative degrees (0, -180) for `west` longitude. Likewise, the processed latitude values should range from 0 to 90 for `north` latitude, and negative degrees (0, -90) for `south` latitude.

To write the function `read_wmo()`, you will also have to create other auxiliary functions that will process some of the fields: to replace missing values with `NA`, or to replace numbers with text for categorical variables, or to process longitude and latitude values.

Here's a general strategy on how you could approach this challenge:

- Download a copy of one of the `.wmo` files (e.g. year 2005) that will be your working sample.
- In case you are not able to download such file from the indicated url, you can find a copy inside the `data/` folder of the course's github repo: `Year.2005.ibtracs_wmo.v03r08.wmo`
- Focus first on what will be the body of your function `read_wmo()`.
- Because WMO format is basically a fixed-width format, you should use the function `read.fwf()`—check its documentation and google for some examples on how to use it.
- Carefully define the width of the columns (for the argument `widths`).
- Carefully define the classes of the columns (for the argument `colClasses`).
- Choose descriptive names for the variables (for the argument `col.names`).
- Start small: begin by reading just a few lines from the downloaded `.wmo` file to play with the `read.fwf()` function.

- Write small functions to handle those variables that need some processing. For instance, you can create a `fix_latitude_indicator()` function that takes the column for *Latitude Indicator*, and converts 1 into `"north"`, and 2 into `"south"`.
- Include descriptions for the auxiliary functions, briefly documenting what each function does, what's the input, and what's the returned value.
- Once your code works for a few lines, try it on the entire file.
- Then start thinking about encapsulating your code for the main function `read_wmo()`. This function should call the other auxiliary functions.
- Ideally (but not mandatory), the body of your `read_wmo()` function should be formed by no mre than 10 lines of code.
- Check that `read_wmo()` works with a couple of different years.
- Include some stop condition in case the user provides an incorrect year (e.g. 2050). In such a case the function should stop with an error message.

Problem 2

Once you have generated a processed version of the data, you will need to create two more variables:

- Create a variable for *Basin* with the initials of the basin.
- Create a variable for *Wind Intensity*, which will be used for plotting purposes, by categorizing maximum average wind speed according to the levels given below. You can use the function `cut()` applied to the values of maximum average wind speed:
  - 0 <= 10
  - 10 <= 30
  - 30 <= 55
  - 55 <= 70
  - 70 <= 85
  - 85 <= 100
  - 100 <= 120
  - > 120

Export a clean data set in CSV format with the following variables (in this order):

- Cyclone identification
- Basin
- Storm Name
- Year
- Month
- Day
- Hour
- Latitude indicator
- Latitude
- Longitude indicator
- Longitude

- Maximum average wind speed
- Central pressure
- Wind intensity

Problem 3

With the data frame that you use to export the CSVC file, use the variables *Longitude*, and *Latitude* to produce a map of the world with the storm trajectories. For reference purposes, you can find links to maps (by year) of storm tracks in the following link:

https://www.ncdc.noaa.gov/ibtracs/index.php?name=browse

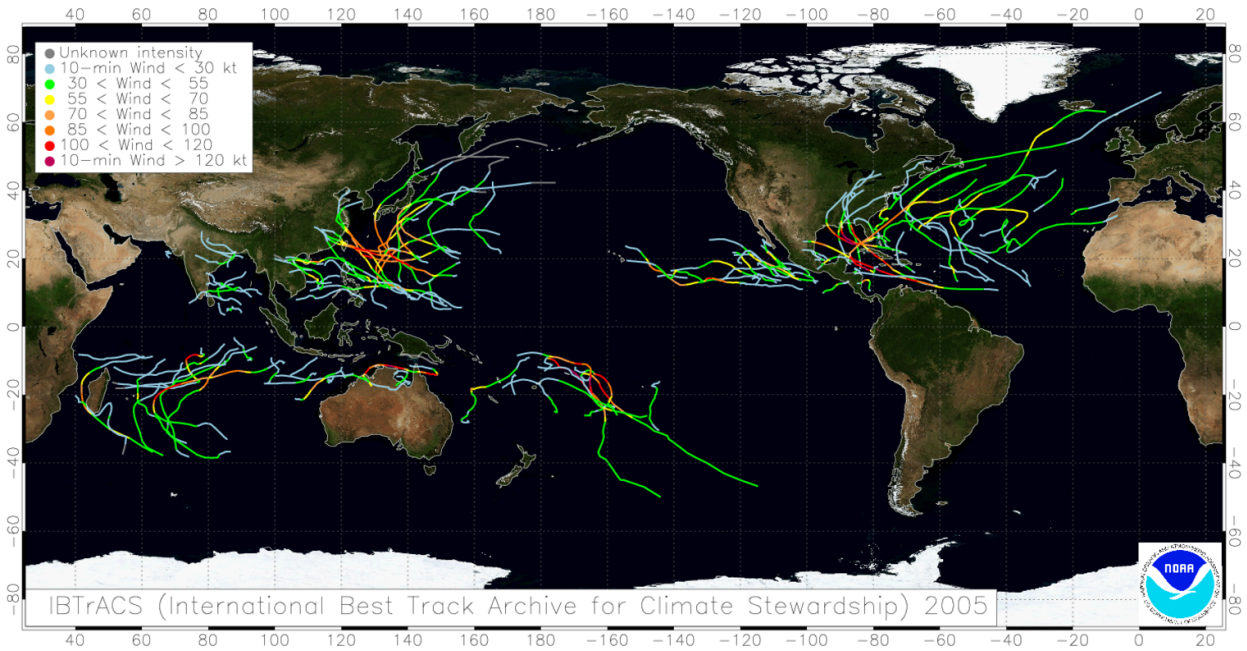

Figure 1: Map of storms, 2005

You can try the following code (extremely basic map):

```
dat <- read_wmo(2005)

# very basic map
library(maps)
map("world")
points(dat$longitude, dat$latitude, pch = ".", cex = 2, col = "blue")
```

A nicer plot can be obtained with the package `"ggplot2"` (you will have to install it if you haven't done so before). I'm using these variables:

- `latitude` (latitude)

- `longitude` (longitude)
- `wind_intensity` (wind intensity)

I also added a variable `group` by combining the `id` (cyclone identification) with the `londitude_id` (longitude indicator). I'm using this variable to properly display the trajectories of individual storms with `geom_path()`.

```
library(ggplot2)
world <- map_data("world")

# initialize "empty" ggplot object
p <- ggplot()
# add map layer
p <- p + geom_polygon(data = world,
                      aes(x=long, y=lat, group = group),
                      colour="gray90", fill="grey70" )
# for grouping purposes of storm trajectories
dat$group <- factor(paste0(dat$id, dat$longitude_id))
# add storm trajectories
p + geom_path(
  data = dat,
  aes(x = longitude, y = latitude, color = wind_intensity, group = group),
  size = 1, alpha = 0.8)
```

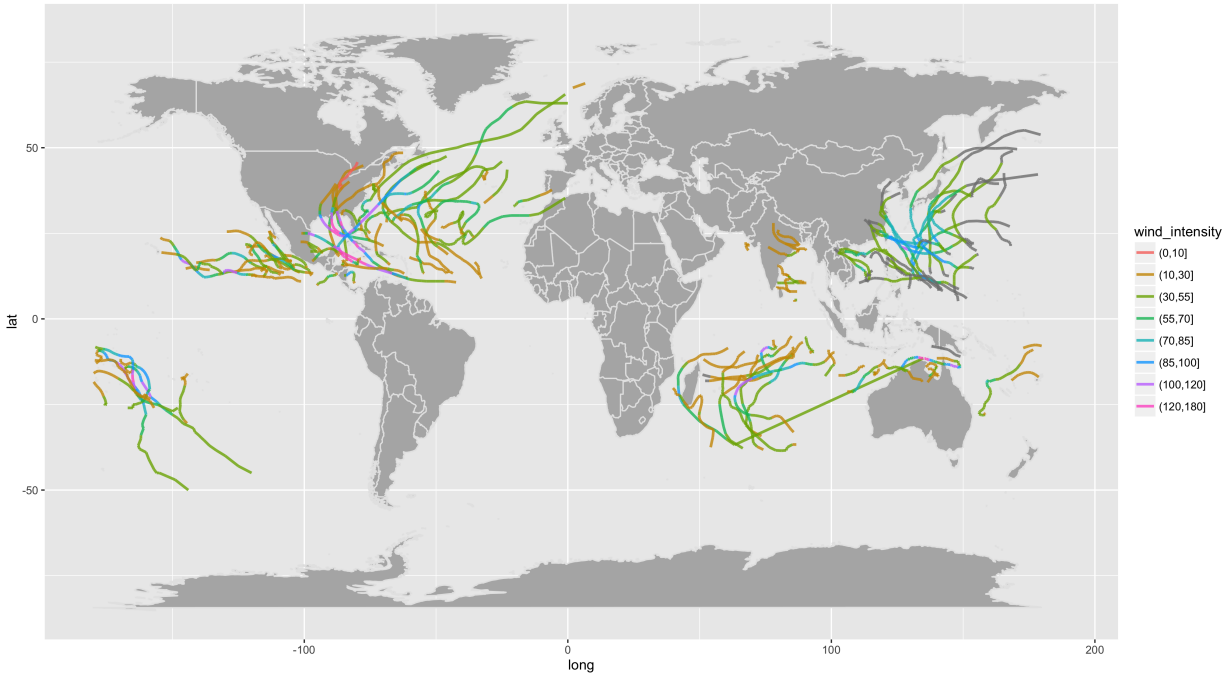

Figure 2: Map of storms in 2005, with ggplot

Problem 4

This problem involves using bash commands. The idea is to practice manipulating a data file. All of the following questions can be answered in R. You could compare your bash solutions with your R solutions.

Use the WMO data file for 2005 to answer the following questions:

- How many (unique) storms were in the WMO file for 2005?
- Which storm has the most recorded values (i.e. lines)?
- Display the number of recorded values per basin in decreasing order.
- How would you subset all the lines for basin `NA` (North Atlantic) and save it into a file `NA-storms-2005.csv`?
- Write a for-loop to subset the lines for each basin in a separate file.

Problem 5

Write an R function, `sample_file()`, which takes a year number (e.g. 2010) and an arbitrary set of indices of observations of a `.wmo` file (e.g. `indices = c(5, 9, 1000, 5000)`) and which returns a data frame (a character vector of unprocessed fields would also be ok) with just those rows.

The function should NOT read the entire `.wmo` file into memory and then subset. This is for sampling very large files. You should practice with the WMO data files for one year. One way this could be used is to take a random sample of observations from a file, based on using `sample()` to determine the indices.