MECH481A6: Engineering Data Analysis in R

Chapter 8 Homework: Functional Programming

Ethan Rutledge

08 December, 2023

Load packages

```
# load packages for current session
library(tidyverse)
library(purrr)
library(lubridate)
library(gridExtra) # needed for extra credit question
```

Chapter 8 Homework

This homework will give you practice at writing functions, mapping functions, and cleaning/plotting data.

When a question asks you to make a plot, remember to set a theme, title, subtitle, labels, colors, etc. It is up to you how to personalize your plots, but put in some effort and make the plotting approach consistent throughout the document. For example, you could use the same theme for all plots. I also like to use the subtitle as a place for the main summary for the viewer.

Question 1

Write a **function** named `sort_abs()`that takes a vector of numbers as input, calculates the absolute values of each entry, and then outputs that vector sorted from smallest to largest value.

```
#create a function named `sort_abs()`
sort_abs <- function(in_vector){
  abs_vector <- abs(in_vector)
  sort_abs_vector <- sort(abs_vector)
  return(sort_abs_vector)
}
```

Question 2

Modify the function `import.w.name()` to import the "date" part of the filename (in addition to the sensor ID). Create a new column variable called "date_created" with this information. Hint: you will need to apply a regex pattern like this: `"(?<=_)[:alnum:]+(?=\\.)"`

```
# create an object that tracks the file names and file paths
# see the coursebook for details
file_list <- list.files('../data/purpleair/', full.names = TRUE)
file_path <- file.path('../data/purpleair/')

# modify the import.w.name function
# hint: start with what is provided in the coursebook
import.w.name <- function(pathname){
  #create a tibble by importing the 'pathname' file
  df <- read_csv(pathname, col_names = TRUE)
  df <- df %>%
    # use stringr::str_extract & a regex to get sensor ID from file name
    # regex translation: "look for a /, then extract all letters and numbers that follow until _"
    mutate(sensor_ID = str_extract(pathname,
                                   "(?<=//)[:alnum:]+(?=_)")) %>%
    mutate(date_created=ymd(str_extract(pathname, "(?<=_)[:alnum:]+(?=\\.)"))) %>%
    # return only a few salient variables to the resultant data frame using dplyr::select
    select(UTCDateTime,
           current_temp_f,
           current_humidity,
           pressure,
           pm2_5_atm,
           sensor_ID) %>%
    na.omit() # remove NA values, which happens when sensor goes offline
  return(df)
}
```

Question 3

This question is designed to give you practice at data cleaning. First, create a pipeline that (1) uses `purrr::map_dfr()` and `import.w.name()` to read in all the the PurpleAir data files into a single data frame. Call that new data frame `PA_data_merged`. (2) Then, have the pipeline convert the character vector UTCDateTime into new column of class POSIXct using a `lubridate::` function (note - not all the indices

in UTCDateTime will parse correctly; we will address this in Question 4). Finally, (3) finish the pipeline by renaming the `current_temp_f` and `current_humidity` column names to shorter names.

```
# the map code is provided in the coursebook
PA_data_merged <- file_list %>% map_dfr(import.w.name) %>%
  mutate(UTCDateTime = ymd_hms(UTCDateTime)) %>%
  rename(curr_temp = current_temp_f, curr_hum = current_humidity)
```

Question 4

Can you find the 3 indices of UTCDateTime in PA_data_merged that failed to parse with lubridate::? Hint: use the `is.na()` function nested wihtin `which()` to return the row numbers in question. Both of these are baseR functions. Once you have the row entries identified you can `View()` them with a call to `slice()`.: normal entries in UTCDateTime are all the same number of characters nchar() or entries that failed to parse in the new date column will have NA associated with them.

```
failed <- which(is.na(PA_data_merged$UTCDateTime))
slice(PA_data_merged, failed)
```

```
## # A tibble: 0 x 6
## # i 6 variables: UTCDateTime <dttm>, curr_temp <dbl>, curr_hum <dbl>,
## #   pressure <dbl>, pm2_5_atm <dbl>, sensor_ID <chr>
```

Question 5

Create a series of EDA plots (cdf, boxplot, histogram, time series) of the `pm2_5_atm` variable from `PA_data_merged`. Use `color =` or `fill =` as an aesthetic to differentiate each sensor by `sensor_ID`. Do the data have a central tendency? Do they appear normally distributed? Do events show up in the time series? Note: the variable `pm2_5_atm` is the concentration of fine particulate matter air pollution in micrograms per cubic meter ($\mu g/m^3$).

Extra Credit

Create the EDA figures within a single plot (hint: use the `gridExtra::` package). Show only one legend and place it within the body of the CDF plot (hint: to move or remove a legend, add a call that uses a version of `theme(legend.position = ...)`).

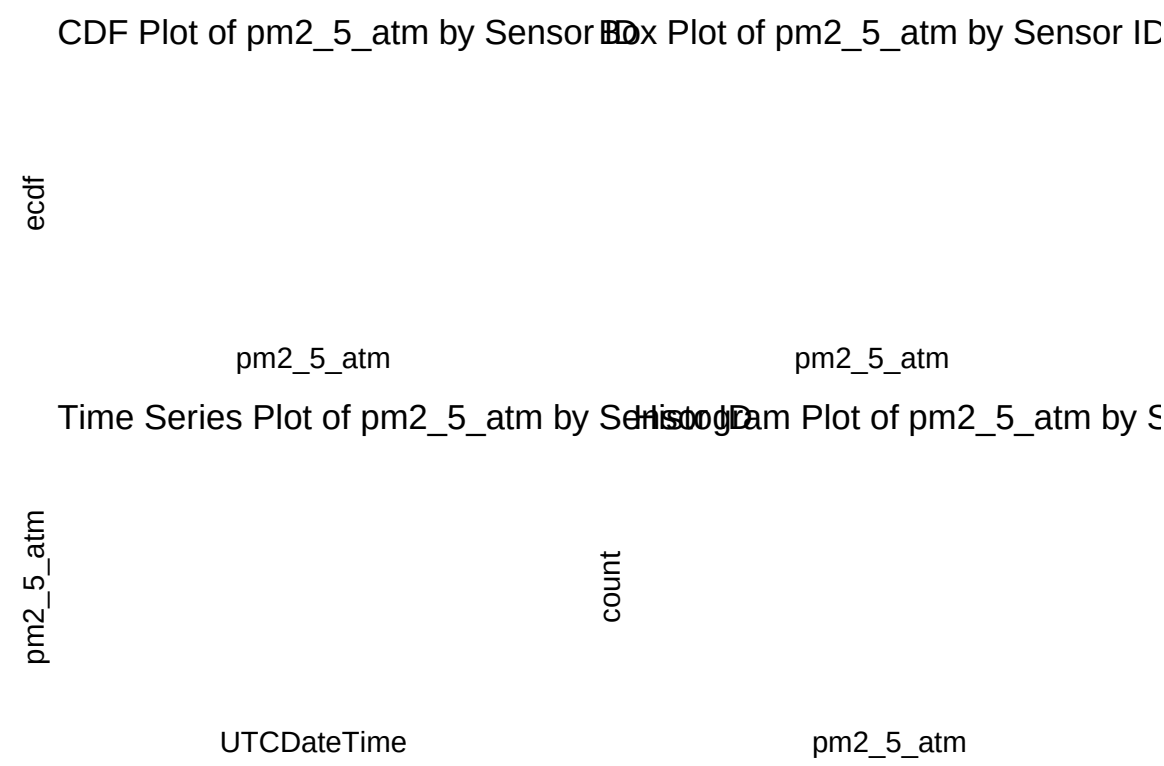
```
#cdf plot
cdf_plot <- ggplot(PA_data_merged) + geom_step(aes(x=pm2_5_atm, color=sensor_ID), stat = "ecdf") + labs

#boxplot
box_plot <- ggplot(PA_data_merged) + geom_boxplot(aes(x=pm2_5_atm, fill=sensor_ID)) + labs(title = "Box

#time-series plot
time_plot <- ggplot(PA_data_merged) + geom_line(aes(x=UTCDateTime, y=pm2_5_atm, color=sensor_ID)) + labs

#histogram
hist_plot <- ggplot(PA_data_merged) + geom_histogram(aes(x=pm2_5_atm, fill=sensor_ID), bins = 50) + labs

single_plot <- grid.arrange(cdf_plot, box_plot, time_plot, hist_plot, nrow = 2, ncol = 2)
```




```
single_plot
```

```
## TableGrob (2 x 2) "arrange": 4 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## 3 3 (2-2,1-1) arrange gtable[layout]
## 4 4 (2-2,2-2) arrange gtable[layout]
```

Appendix

```
# set global options for figures, code, warnings, and messages
knitr::opts_chunk$set(fig.width=6, fig.height=4, fig.path="../figs/", warning=FALSE, message=FALSE)
# load packages for current session
library(tidyverse)
library(purrr)
library(lubridate)
library(gridExtra) # needed for extra credit question

#create a function named `sort_abs()`
sort_abs <- function(in_vector){
  abs_vector <- abs(in_vector)
  sort_abs_vector <- sort(abs_vector)
  return(sort_abs_vector)
}

# create an object that tracks the file names and file paths
# see the coursebook for details
file_list <- list.files('../data/purpleair/', full.names = TRUE)
file_path <- file.path('../data/purpleair/')

# modify the import.w.name function
# hint: start with what is provided in the coursebook
import.w.name <- function(pathname){
  #create a tibble by importing the 'pathname' file
  df <- read_csv(pathname, col_names = TRUE)
  df <- df %>%
    # use stringr::str_extract & a regex to get sensor ID from file name
    # regex translation: "look for a /, then extract all letters and numbers that follow until _"
    mutate(sensor_ID = str_extract(pathname,
                                   "(?<=//)[:alnum:]+(?=_)")) %>%
    mutate(date_created=ymd(str_extract(pathname, "(?<=_)[:alnum:]+(?=\\.)"))) %>%
    # return only a few salient variables to the resultant data frame using dplyr::select
    select(UTCDateTime,
           current_temp_f,
           current_humidity,
           pressure,
           pm2_5_atm,
           sensor_ID) %>%
    na.omit() # remove NA values, which happens when sensor goes offline
  return(df)
}



# the map code is provided in the coursebook
PA_data_merged <- file_list %>% map_dfr(import.w.name) %>%
  mutate(UTCDateTime = ymd_hms(UTCDateTime)) %>%
  rename(curr_temp = current_temp_f, curr_hum = current_humidity)

failed <- which(is.na(PA_data_merged$UTCDateTime))
slice(PA_data_merged, failed)
```

```r
#cdf plot
cdf_plot <- ggplot(PA_data_merged) + geom_step(aes(x=pm2_5_atm, color=sensor_ID), stat = "ecdf") + labs

#boxplot
box_plot <- ggplot(PA_data_merged) + geom_boxplot(aes(x=pm2_5_atm, fill=sensor_ID)) + labs(title = "Box

#time-series plot
time_plot <- ggplot(PA_data_merged) + geom_line(aes(x=UTCDateTime, y=pm2_5_atm, color=sensor_ID)) + labs

#histogram
hist_plot <- ggplot(PA_data_merged) + geom_histogram(aes(x=pm2_5_atm, fill=sensor_ID), bins = 50) + labs

single_plot <- grid.arrange(cdf_plot, box_plot, time_plot, hist_plot, nrow = 2, ncol = 2)

single_plot
```