azad_arshad_RA

azad

10/04/2022

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(writexl)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(stargazer)
```

```
##
## Please cite as:

##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

Data Cleaning and Manipulation

##1

```r
#1 load dataset:"vehicle_pucc_certificates.csv"

pucc <- read.csv("./data/vehicle_pucc_certificates.csv")
```

##2

```r
#2 total number of vehicles present

tot_veh <- length(unique(pucc$vehicle_number))
tot_veh
```

```
## [1] 1000
```

```r
# Dimensions of the datafile
dim_pucc <- dim(pucc)
dim_pucc
```

```
## [1] 14000    12
```

```r
# column that uniquely identifies the each row
len_pucc_n <- length(unique(pucc$pucc_number))
len_pucc_n
```

```
## [1] 14000
```

```r
## column `pucc_number` uniquely identifies the each rows.
```

##3

```r
# vehicle count summary

vehicle_count_summary_df <- pucc %>% select(vehicle_number, pucc_number) %>% group_by(vehicle_number) %

vehicle_count_summary_df %>% writexl::write_xlsx("./azad_arshad_output/vehicle_count_summary.xlsx")
```

##4

```r
# converting the string `timestamp` into POSIXct `ts2`.
pucc$ts2 <- ymd_hms(pucc$timestamp)

# extracting year, month, and date from `ts2`

pucc$test_year <- year(pucc$ts2)
pucc$test_month <- month(pucc$ts2)
pucc$test_date <- mday(pucc$ts2)
```

##5

```r
# variable `vehicle_test_result` denotes the outcome of the PUC test.

vehicle_test_result_year_summary_df <- pucc %>% select(vehicle_test_result, test_year) %>% group_by(vehi
```

```
## `summarise()` has grouped output by 'vehicle_test_result'. You can override
## using the `.groups` argument.
```

```r
vehicle_test_result_year_summary_df %>% writexl::write_xlsx("./azad_arshad_output/vehicle_test_result_y
```

##6

```r
# number of pollution test conducted in 2021

pucc %>% filter(test_year == "2021") %>% summarise(n())
```

```
##    n()
## 1 2000
```

```r
# no of unique diesel that took test in 2015
pucc$vehicle_FuelType <- tolower(pucc$vehicle_fuel_type)
pucc %>% filter(test_year == "2015" & vehicle_FuelType == "diesel") %>% summarise(n())
```

```
##   n()
## 1  33
```

##7

```r
# a) percentage of missing data in variable `vehicle_engine_stroke`

missin_vehicle_engine_stroke <- 100*sum(is.na(pucc$vehicle_engine_stroke))/length(pucc$vehicle_engine_st
missin_vehicle_engine_stroke
```

```
## [1] 22.62857
```

```r
# b) replacing missing values with `Missing`


pucc$vehicle_engine_stroke <- ifelse(is.na(pucc$vehicle_engine_stroke), "Missing", pucc$vehicle_engine_s

# c) replaing "4 S" and "4-Stroke" with "4_str"; "2 S" and "2-Stroke" with "2_str".

pucc$vehicle_engine_stroke <- str_replace_all(pucc$vehicle_engine_stroke, c("4 S"="4_str", "4-Stroke"="4
```

##8

```r
# load dataset:"pollution_checking_centers.csv"

pollution_cc_df <- read.csv("./data/pollution_checking_centers.csv")
```

```r
# a) merging it with "vehicle_pucc_certificates" dataset

df_merged <- pucc %>% left_join(pollution_cc_df, by="center_code")

## number of datapoints left unmerged:

sum(!unique(pucc$center_code) %in% unique(pollution_cc_df$center_code))
```

```
## [1] 454
```

Data Exploration and Visualization

##1

```r
# histogram of pollution test counts
##
png("./azad_arshad_output/vehicle_count_hist.png", width = 500, height = 500)
h <- hist(vehicle_count_summary_df$tot_count)
text(h$mids,h$counts,labels=h$counts, adj=c(0.5, -0.5))
dev.off()
```

```
## pdf
##   2
```

##2

```r
## summarize the number of vehicle by fuel type and year.

## data cleaning
### alternate wordings

pucc$vehicle_FuelType <- str_replace_all(pucc$vehicle_FuelType, c("lpg/petrol"="petrol/lpg", "petlpg"="

### probably mean the same

pucc$vehicle_FuelType <- str_replace_all(pucc$vehicle_FuelType, c("cng only"="cng", "lpg only"="lpg"))

veh_count_fuel_df <- pucc %>% select(test_year, vehicle_FuelType, pucc_number) %>% group_by(test_year, 
```

```
## `summarise()` has grouped output by 'test_year'. You can override using the
## `.groups` argument.
```

```r
## Plot Bar graphs

pucc$test_year <- as.factor(pucc$test_year)
veh_count_fuel_plot <- ggplot(veh_count_fuel_df,
       aes(x = test_year, y = count,
           fill = vehicle_FuelType)) +
  geom_bar(stat = "identity", position=position_dodge())+
  ggtitle(label = "number of vehicles tested in each year by their fuel type") +
```

```
  labs(x = "year", y = "test count") +
theme(axis.text.x = element_text(angle = 0, hjust = 1)) +
geom_text(aes(label = count), vjust = -0.2, size = 2.5,
position = position_dodge(0.9))

ggsave("./azad_arshad_output/vehicle_count_fuel_type_year.png", width = 9, height = 5, units = "in", dp
```

##3

```
## average number of tests conducted in each month over seven years.

pucc$test_month <- as.factor(pucc$test_month)

test_avg_per_month_df <- pucc %>% select(test_year, test_month, pucc_number) %>% group_by(test_year, tes
```

```
## `summarise()` has grouped output by 'test_year'. You can override using the
## `.groups` argument.
```

```
test_avg_per_month_df$test_month <- as.factor(test_avg_per_month_df$test_month)

## line plot

png("./azad_arshad_output/test_avg_per_month_plot", width = 500, height = 500)

ggplot(test_avg_per_month_df)+geom_point(aes(test_month, count_month))+geom_line(aes(test_month, count_m
  labs(x = "test_month", y = "count_month")+
theme(axis.text.x = element_text(angle = 0, hjust = 1))
dev.off()
```

```
## pdf
##   2
```

Estimation and Causal Inference

##1

```
## creating dummies

pucc$vehicle_test_result_dummy <- ifelse(pucc$vehicle_test_result == "Pass", 1, 0)

pucc$vehicle_FuelType_dummy <- ifelse(pucc$vehicle_FuelType == "diesel", 1, 0)
```

##2

```
## OLS

test_reg_ols <- lm(vehicle_test_result_dummy ~ vehicle_FuelType_dummy, pucc)
test_reg_ols
```

```
##
## Call:
## lm(formula = vehicle_test_result_dummy ~ vehicle_FuelType_dummy,
##     data = pucc)
##
## Coefficients:
##            (Intercept)  vehicle_FuelType_dummy
##                0.94930                 0.03407
```

```
stargazer(test_reg_ols, type = "text",
          title = "impact of vehicle fuel type on pucc test result: an OLS regression", out = "./azad_a
```

```
##
## impact of vehicle fuel type on pucc test result: an OLS regression
## ================================================
##                            Dependent variable:
##                        ---------------------------
##                         vehicle_test_result_dummy
## ------------------------------------------------
## vehicle_FuelType_dummy          0.034***
##                                  (0.007)
##
## Constant                         0.949***
##                                  (0.002)
##
## ------------------------------------------------
## Observations                      14,000
## R2                                 0.002
## Adjusted R2                        0.002
## Residual Std. Error         0.214 (df = 13998)
## F Statistic              23.998*** (df = 1; 13998)
## ================================================
## Note:                  *p<0.1; **p<0.05; ***p<0.01
```

##3

```
# Interpretation of OLS regression.
```

##4

Normally when independent variable(s) are qualitative and/or quantitative and the dependent variable is a quantitative one, it makes sense to use OLS. When the dependent variable is also a binary one, this leads to a problems. We can not interpret the Beta coefficients as giving the rate of change of `test result` for a unit change in `vehicle fuel type` as varibale `test result` takes just 2 values. 1 and 0.

The solution is to use a linear probability model instead of OLS. We can interpret the Beta coefficient as the change in the probability that `test result=1`, when the `vehicle fuel` type changes.

Additional Questions

1. As long as the codes are clean and each chunk follows the other sensibly there is not much change in cleaning and analysis. Though there exists certain ways in which the performance of codes could be

optimised. For R, parallel progamming could be helpful but again that requires a change in the way the code has to be written. Certain packages such as `apache` in R, could also be employed to improve the calculation time and load on the system. Analysis of bigger dataset necessitates the use of a stronger system, more RAM and faser processor. Use of package such as `apache` could be a wayaround such reqirement.

2. As the complexity of the codes and the project rises it become necessary to implement a system of organization of the codes. The best practices to follow are

    1. Comment everything. Not just what is in the codes but also the why part. The codes should be self sufficient in the sense that if anyone(including the authors) read the code in the future, it is should make sense.
    2. Uniform and clear nomenclature: The variable and the dataset should follow a consistent pattern in their names. All the datasets names could end with a `_df` in the code. Whereas for the variable, it should be descriptive. A variable name such as `hh_male_count` is better than `householdmalecount`. Also, abberviations should be avoided i.e, nothing like `hhmc`.
    3. A version control system such as *Github* could be used. It helps to keep track of dataset and the changes in the codes. It is also helpful in the case of inadvertent file deletion corruption.