

Grading guidelines ECON3170/4170 Fall 2022

I.

a)

```
library(tidyverse)
library(readxl)

vdem <- read_csv('vdem.csv')
summary(vdem$year)
vdem %>%
  filter(year==2021) %>%
  nrow()
```

The year variable ranges from 1789 to 2021. There were 179 countries present in the data in 2021.

b)

```
vdem %>%
  filter(country_text_id %in% c('NOR','USA')) %>%
  ggplot(aes(x=year,y=v2x_libdem,color=country_name)) +
  geom_line()
```

If the candidate produces two separate plots, this should give some but not full credit.

c)

```
vdem %>%
  filter(year>=1950) %>%
  group_by(year) %>%
  summarise(p50=median(v2x_libdem, na.rm=T),
            p25=quantile(v2x_libdem, probs = .25, na.rm=T),
            p75=quantile(v2x_libdem, probs = .75, na.rm=T)) %>%
  ggplot(aes(x=year)) +
  geom_line(aes(y=p50), color='red', size=1.5) +
  geom_line(aes(y=p25)) +
  geom_line(aes(y=p75))
```

Should present everything in one diagram for full score. Should get some credit for only median.

d)

```
vdem %>%
  filter(year %in% c(1980, 1990, 2000, 2010, 2020)) %>%
  select(country_name,year,v2x_libdem) %>%
  pivot_wider(names_from = year,
              names_prefix = 'demo_',
              values_from = v2x_libdem)
```

A working pivot should give a good score. Need proper variable names for full score.

e)

```
fh <- read_excel('FreedomHouse.xlsx')
names(fh) <- c('country', 'score_status', 'polrights', 'civillib')
fh <- fh %>%
  mutate(score = as.numeric(word(score_status,1)),
         status = word(score_status,2,-1))
summary(fh$score)
Mean score is 55.71, the range is 1-100.
```

f)

```
# Quick and dirty - gives 169 matches
vdem %>%
  filter(year == 2021) %>%
  inner_join(fh, by = c('country_name' = 'country'))

# Cleaner using country codes - gives 173 matches
library(countrycode)
fh$countrycode <- countrycode(fh$country, origin = 'country.name',
                             destination = 'iso3c')
fh <- fh %>%
  mutate(territory = str_sub(country, -1, -1) == '*',
         countrycode = ifelse(territory, NA, countrycode))
demo <- vdem %>%
  filter(year == 2021) %>%
  inner_join(fh, by = c('country_text_id' = 'countrycode'))

demo %>%
  ggplot(aes(x=v2x_libdem, y=score)) +
  geom_point() +
  labs(x = 'V-Dem index',
       y = 'Freedom House score')
```

Should comment on the strong but not perfect positive association

Should get some score for matching on names without any transformation, but not full score. Should also get some deduction for not removing territories (which gives some artificial matches).

II.

a)

```
library(tidyverse)
library(tidymodels)

chocolate <- read_delim("chocolate.tsv",
                        delim = "\t")
table(chocolate$Rating)
chocolate %>%
  ggplot(aes(x=as.factor(Rating))) +
  geom_bar() +
  labs(x='Rating', y='Frequency')
```

There are several ways to plot the data. The simplest is to use plot on table, or can use ggplot.

b)

```
ch <- chocolate %>%
  select(Rating, Cocoa_Percent, Review_Date)

set.seed(1234)
ch_split <- initial_split(ch, prop = 0.8)
ch_train <- training(ch_split)
ch_test <- testing(ch_split)

ols_fit <- linear_reg() %>%
  set_engine("lm") %>%
  fit(Rating~., data = ch_train)
```

The linear model fits parameters to minimize the sum of errors

$$\sum (Rating_i - \beta_0 - \beta_1 Cocoa_i - \beta_2 Date_i)^2,$$

hence it assumes a linear structure.

Acceptable to use other approaches than TidyModels. Caret Ok, fully manual also but lots of care has to be taken. Should remember to set the seed for replicability for full score.

c)

```
rf_fit <- rand_forest() %>%
  set_mode("regression") %>%
  set_engine("ranger") %>%
  fit(Rating~., data = ch_train)
```

Should give an explanation for a tree model (sequentially split the data according to one explanatory variable). A random forest works on a bootstrapped sample (so bagging) and a random selection of the explanatory variables. This was covered in quite some detail in the lectures.

Only providing code with no explanation of RF should give about 25% score

d)

```
pred <- predict(ols_fit,ch_test) %>%
  rename(ols_pred = .pred) %>%
  bind_cols(ch_test)
pred <- predict(rf_fit,ch_test) %>%
  rename(rf_pred = .pred) %>%
  bind_cols(pred)
pred %>%
  mutate(esq_ols=(Rating-ols_pred)^2,
         esq_rf =(Rating- rf_pred)^2) %>%
  summarise(mse_ols=mean(esq_ols),
            mse_rf =mean(esq_rf))
```

The models perform about equally (and not very well), with the linear model doing slightly better at MSE=.204 vs .206.

e)

The splitting of the sample makes it possible to study the prediction performance of various models out of sample. Studying the prediction on the training sample would be in-sample performance. This works to see if the model fits the data, but cannot detect over-fitting. As many ML models are susceptible to over-fitting, this is potentially a serious problem.

f)

```
ch2 <- chocolate %>%
  select(Rating,Cocoa_Percent,Review_Date,Bean_Origin) %>%
  group_by(Been_Origin) %>%
  mutate(n=n()) %>%
  mutate(Been_Origin = as.factor(if_else(n>20,Bean_Origin,"Other"))) %>%
  select(!n)
```

```
set.seed(1234)
ch2_split <- initial_split(ch2, prop = 0.8)
ch2_train <- training(ch2_split)
ch2_test <- testing(ch2_split)
```

```
ols_fit <- linear_reg() %>%
  set_engine("lm") %>%
  fit(Rating~., data = ch2_train)
```

```
pred <- predict(ols_fit,ch2_test) %>%
  rename(ols_pred = .pred) %>%
  bind_cols(ch2_test)
```

```

rf_fit <- rand_forest() %>%
  set_mode("regression") %>%
  set_engine("ranger") %>%
  fit(Rating~., data = ch2_train)

pred <- predict(rf_fit, ch2_test) %>%
  rename(rf_pred = .pred) %>%
  bind_cols(pred)

pred %>%
  mutate(esq_ols=(Rating-ols_pred)^2,
         esq_rf =(Rating- rf_pred)^2) %>%
  summarise(mse_ols=mean(esq_ols),
            mse_rf =mean(esq_rf))

```

We get MSE 0.204 for OLS, 0.203 for RF. Not a huge impact, and hardly any help for the linear model. Now RF does better, probably due to some non-linear pattern.

III.

a)

```
library(tidyverse)
```

```
Y <- 10
```

```
r <- 0.05
```

```
gamma = .4
```

```
beta = .96
```

```
U <- function(c1,c2) {  
  return((c1^(1-gamma))/(1-gamma) + beta*(c2^(1-gamma))/(1-gamma))  
}
```

```
c1 <- seq(from = .1, to = Y-.1, by = .01)
```

```
c2 <- (1+r)*(Y-c1)
```

```
Util <- U(c1,c2)
```

```
tibble(c1,c2,Util) %>%  
  ggplot(aes(x=c1,y=Util)) +  
  geom_line()
```

Candidates should get some score (1/3) for coding the utility function. If a candidate computes utility without using the budget constraint, should get 2/3.

b)

```
u <- function(c1) {  
  c2 <- (1+r)*(Y-c1)  
  return(-U(c1,c2))  
}  
optim(2,u, method = 'BFGS')
```

Find optimum 5.07.

Necessary to include the BC in a function. Newton, quasi-Newton and Nelder-Mead discussed in lectures. For (quasi-)Newton, tries to approximate the function with a quadratic and maximize this in each step. Nelder-Mead makes a simplex ("triangle" – actually only two points in a one dimensional problem), crawls around and increases/shrinks the size.

Should not get full score without some intuition for algorithm.

c)

```
N=1e3
```

```
lambda = .9
```

```
R = rexp(N, rate = lambda)
```

```
Eu <- function(c1) {  
  c2 <- R*(Y-c1)  
  return(-mean(U(c1,c2)))  
}  
Eu(5)
```

d)

```
optim(2, Eu, method = 'BFGS')
```

Find optimum 5.03.

To be able to maximize properly, need to make sure that the draws of R remain the same for each call to Eu. This has not been covered in detail in class, so most decent attempts should be rewarded some points.