IST5128 - Take Home Exam (Due Date: 28/11/2022)

- PLEASE READ THE LAST PAGE OF THESE QUESTIONS TO LEARN ABOUT OTHER DETAILS ABOUT TAKE-HOME EXAM
- This take home exam consists of two parts.
  - Part A is about writing some code to convert hex color codes to rgb color codes.
  - Part B is about a portion of V-Dem dataset where you will manipulate and visualize data.

Part A: MORE ABOUT COLOR CODES (Due Date: 28/11/2022)

Hex Codes

- Hex codes is used to give a unique code to each color.
- Hex code is a six digit expression where,
  - First two digits denote red values
  - Third and fourth digits denote green values
  - Fifth and sixth digits denote blue values
- Hex codes are given in base of 16.
- So values between 0 to 9 is given as their original numbers but values between 10-15 is denoted with letters A to F.
- Thus, 10 = A, 11=B, 12=C, 13=D, 14=E and 15=F in a hex expression.
- It is common to start hex expressions with a hashtag:
- `#A5B468` is an example of hex code.
- `A5` denotes the red color, `B4` denotes the green color and `68` denotes the blue color.
- All the colors are obtained with the combination of red, green and blue colors.

Part A: MORE ABOUT COLOR CODES (Due Date: 28/11/2022)

RGB Codes

- RGB is another expression to denote colors.
- In RGB, red green and blue intensity is denoted with values between 0 and 255.
- The function `col2rgb()` can be used to convert a hex expression to rgb in R.
- For example for the hex expression `#A5B468`

```
> col2rgb("#A5B468")
      [,1]
red    165
green  180
blue   104
```

and (165,180,104) is the RGB expression of the same color.

- In the first part of this take home exam you will reverse this process and write some codes to convert rgb codes to hex codes.

Part A (Due Date: 28/11/2022)

1. **(10 pt)**
   - Start by writing a function `create_rgb()` that creates rgb codes at the desired size.
   - You can use `sample` to achieve this.
   - Your output should be in `(r,g,b)` format.
   - For example when you call the function `create_rgb()` your output should give something like (it doesn't have to be exactly the same though);

```
> create_rgb(n=3)

[1] "(65,0,199)"   "(16,120,183)" "(188,62,54)"

> set.seed(100)
> Q1 <- create_rgb(n=6)
> Q1

[1] "(201,246,101)" "(111,242,216)" "(249,205,235)" "(150,255,213)"
[5] "(197,3,54)"    "(69,97,134)"
```

Part A (Due Date: 28/11/2022)

2. **(5 pt)** Write a helper function `find_quotient()` that finds quotient when a dividend and divisor is provided.

```
> find_quotient(201,5)
[1] 40
> find_quotient(40,10)
[1] 4
> find_quotient(85,7)
[1] 12
```

Part A (Due Date: 28/11/2022)

3. **(5 pt)** Write a helper function `find_remainder()` that finds remainder when a dividend and divisor is provided.

```
> find_remainder(201,5)
[1] 1
> find_remainder(40,10)
[1] 0
> find_remainder(85,7)
[1] 1
```

Part A (Due Date: 28/11/2022)

4. (10 pts)

- Write a function `extract_rgb()` such that when you input a vector given in Q1, the result will be a dataframe with three columns
  - where each row represents a different color codes and
  - where each column represents red, green and blue color codes respectively
- You can use regular expressions or/and suitable `stringr` functions and then convert your result to a matrix or dataframe

```
> Q1
[1] "(201,246,101)" "(111,242,216)" "(249,205,235)" "(150,255,213)"
[5] "(197,3,54)"    "(69,97,134)"
> Q4 <- extract_rgb(Q1)
> Q4
    R   G   B
1 201 246 101
2 111 242 216
3 249 205 235
4 150 255 213
5 197   3  54
6  69  97 134
```

Part A (Due Date: 28/11/2022)

5. **(5 pts)**
   - To convert rgb values to hex codes you have to find quotient and remainder of rgb values in base of 16.
   - To do that you have to divide each color value by 16.
   - For example if your rgb color code is `(201,246,101)`
     - for 201/16; quotient is 12 and remainder is 9 which denotes the first two digits (red color part) of hex code.
     - for 246/16; quotient is 15 and remainder is 6 which denotes the third and fourth digits (green color part) of hex code.
     - for 101/16; quotient is 6 and remainder is 5 denotes the fifth and sixth digits (blue color part) of hex code
   - Luckily, you have written two helper codes in `Q2` and `Q3`
   - Now use those helper functions on the `Q4` with the `apply` family to get the quotient and the remainder.

Part A (Due Date: 28/11/2022)

```
> Q4
    R   G   B
1 201 246 101
2 111 242 216
3 249 205 235
4 150 255 213
5 197   3  54
6  69  97 134
```

```
> Q5a_quotient
      R  G  B
[1,] 12 15  6
[2,]  6 15 13
[3,] 15 12 14
[4,]  9 15 13
[5,] 12  0  3
[6,]  4  6  8
```

```
> Q5b_remainder
      R  G  B
[1,]  9  6  5
[2,] 15  2  8
[3,]  9 13 11
[4,]  6 15  5
[5,]  5  3  6
[6,]  5  1  6
```

Part A (Due Date: 28/11/2022)

6. **(5 pts)**
   - For the values 0 to 9 we have no problem but for the values 10 to 16 we have to replace them with A to F.
   - Write a helper function `replace_10_16()` that replaces 10 to 16 with A to F

```
> replace_10_16(c(12,15,6))
[1] "C" "F" "6"
> replace_10_16(c(3,4,5))
[1] "3" "4" "5"
> replace_10_16(c(13,15,10))
[1] "D" "F" "A"
```

Part A (Due Date: 28/11/2022)

7. **(5 pt)** Extend this function and use `apply` to `Q5a_quotient` and `Q5b_remainder` to get the suitable output.

```
> Q4
    R   G   B
1 201 246 101
2 111 242 216
3 249 205 235
4 150 255 213
5 197   3  54
6  69  97 134
```

```
> Q5a_quotient
     R  G  B
[1,] 12 15  6
[2,]  6 15 13
[3,] 15 12 14
[4,]  9 15 13
[5,] 12  0  3
[6,]  4  6  8
> Q5b_remainder
     R  G  B
[1,]  9  6  5
[2,] 15  2  8
[3,]  9 13 11
[4,]  6 15  5
[5,]  5  3  6
[6,]  5  1  6
```

```
> Q7a_quotient
     R   G   B
[1,] "C" "F" "6"
[2,] "6" "F" "D"
[3,] "F" "C" "E"
[4,] "9" "F" "D"
[5,] "C" "0" "3"
[6,] "4" "6" "8"
> Q7b_remainder
     R   G   B
[1,] "9" "6" "5"
[2,] "F" "2" "8"
[3,] "9" "D" "B"
[4,] "6" "F" "5"
[5,] "5" "3" "6"
[6,] "5" "1" "6"
```

Part A (Due Date: 28/11/2022)

8. **(10 pts)**
   - Combine `Q7a_quotient` and `Q7b_remainder` to get the final hexcodes.
   - Don't forget to add # in front of expression

```
> Q1
[1] "(201,246,101)" "(111,242,216)" "(249,205,235)" "(150,255,213)"
[5] "(197,3,54)"    "(69,97,134)"
> Q8
[1] "#C9F665" "#6FF2D8" "#F9CDEB" "#96FFD5" "#C50336" "#456186"
```

Part B: V-Dem Database (Due Date: 28/11/2022)

- You are given a portion of V-DEM database.
- Full data and information can be found on https://www.v-dem.net/
- The data is in R data format with the `.rds` extension.
- Load the dataset with the `readRDS()` function and assign it to `df`.

```
> str(df)
'data.frame':        693 obs. of  15 variables:
 $ country_name      : chr  "Mexico" "Mexico" "Mexico" "Mexico" ...
 $ year              : num  2011 2012 2013 2014 2015 ...
 $ v2x_polyarchy     : num  0.652 0.649 0.623 0.623 0.633 0.636 0.63 0.674 0.669 0.647 ...
 $ v2x_libdem        : num  0.459 0.455 0.418 0.42 0.419 0.424 0.43 0.453 0.438 0.412 ...
 $ v2x_partipdem     : num  0.404 0.402 0.371 0.37 0.39 0.39 0.402 0.433 0.402 0.389 ...
 $ v2x_delibdem      : num  0.54 0.541 0.496 0.494 0.488 0.49 0.474 0.47 0.406 0.387 ...
 $ v2x_regime        : num  2 2 2 2 2 2 2 2 2 2 ...
 $ v2x_accountability: num  1.004 0.985 0.896 0.901 0.912 ...
 $ v2x_diagacc       : num  0.952 0.953 0.829 0.838 0.872 ...
 $ v2x_corr          : num  0.636 0.662 0.769 0.769 0.742 0.742 0.701 0.689 0.601 0.586 ...
 $ v2x_gender        : num  0.747 0.76 0.773 0.774 0.775 0.775 0.794 0.829 0.835 0.823 ...
 $ v2x_rule          : num  0.581 0.559 0.457 0.457 0.47 0.47 0.474 0.474 0.541 0.514 ...
 $ v2xcs_ccsi        : num  0.818 0.823 0.758 0.758 0.758 0.758 0.785 0.853 0.883 0.817 ...
 $ v2xcl_disc        : num  0.767 0.767 0.808 0.808 0.808 0.808 0.856 0.924 0.924 0.825 ...
 $ v2xca_academ      : num  0.941 0.941 0.933 0.939 0.935 0.943 0.936 0.906 0.906 0.863 ...
```

Part B: V-Dem Database (Due Date: 28/11/2022)

- `country_name`: This portion of the data consists 63 countries.
- `year`: This portion of the data has the year range 2011 to 2021.
- `v2x_regime`: Regimes of the World index is a categorical variable where
  - 0:Closed autocracy,
  - 1:Electoral autocracy,
  - 2:Electoral democracy,
  - 3:Liberal democracy
- `v2x_polyarchy`: Electoral democracy index
- `v2x_libdem`: Liberal democracy index
- `v2x_partipdem`: Participatory democracy index
- `v2x_delibdem`: Deliberative democracy index
- `v2x_accountability`: Accountability index
- `v2x_diagacc`: Diagonal accountability index
- `v2x_corr`: Political corruption index
- `v2x_gender`: Women political empowerment index
- `v2x_rule`: Rule of law index
- `v2x_ccsi`: Core civil society index
- `v2x_disc`: Freedom of discussion
- `v2x_academ`: Academic Freedom Index

Part B (Due Date: 28/11/2022)

9. **(5 pts)**
   - Find average Polyarch index for each country for the given years.
   - Sort your result from highest to lowest.

```
> print(Q9,n=15)
# A tibble: 63 x 2
   country_name mean_polyarchy
   <chr>                  <dbl>
 1 Sweden                 0.919
 2 Denmark                0.913
 3 Costa Rica             0.907
 4 Switzerland            0.898
 5 Norway                 0.896
 6 New Zealand            0.894
 7 Belgium                0.893
 8 Uruguay                0.891
 9 Portugal               0.890
10 France                 0.888
11 Germany                0.887
12 Australia              0.879
13 Finland                0.877
14 Chile                  0.875
15 Netherlands            0.873
# ... with 48 more rows
```

Part B (Due Date: 28/11/2022)

10. **(5 pts)**
    - Create a new variable called `democracy_index` by taking average all the variables except `country_name`, `year` and `regime`.
    - Round new variable to 3 digits.

```
> head(Q10 %>% select(country_name, year, democracy_index),n=15)
```

	country_name	year	democracy_index
1	Mexico	2011	0.708
2	Mexico	2012	0.708
3	Mexico	2013	0.678
4	Mexico	2014	0.679
5	Mexico	2015	0.683
6	Mexico	2016	0.686
7	Mexico	2017	0.693
8	Mexico	2018	0.744
9	Mexico	2019	0.730
10	Mexico	2020	0.675
11	Mexico	2021	0.625
12	Sweden	2011	1.017
13	Sweden	2012	1.022
14	Sweden	2013	1.008
15	Sweden	2014	1.006

Part B (Due Date: 28/11/2022)

11. **(10 pts)**
    - Use `group_by()` and `mutate()` together to create a year-based rank for each country named `democracy_rank()`.
    - Each ranking should be made separately for each country.
    - Use function `rank()` to obtain rankings.

Part B (Due Date: 28/11/2022)

```
> head(Q11 %>% select(country_name, year, democracy_index, democracy_rank),n=22)
```

	country_name	year	democracy_index	democracy_rank
1	Mexico	2011	0.708	3.5
2	Mexico	2012	0.708	3.5
3	Mexico	2013	0.678	9.0
4	Mexico	2014	0.679	8.0
5	Mexico	2015	0.683	7.0
6	Mexico	2016	0.686	6.0
7	Mexico	2017	0.693	5.0
8	Mexico	2018	0.744	1.0
9	Mexico	2019	0.730	2.0
10	Mexico	2020	0.675	10.0
11	Mexico	2021	0.625	11.0
12	Sweden	2011	1.017	3.0
13	Sweden	2012	1.022	1.0
14	Sweden	2013	1.008	6.0
15	Sweden	2014	1.006	7.0
16	Sweden	2015	1.019	2.0
17	Sweden	2016	1.014	4.0
18	Sweden	2017	1.013	5.0
19	Sweden	2018	0.998	8.0
20	Sweden	2019	0.989	10.5
21	Sweden	2020	0.990	9.0
22	Sweden	2021	0.989	10.5

Part B (Due Date: 28/11/2022)

12. **(5 pts)** Create a new variable `Year_Group` by dividing the year into two groups: Between 2010-2015 and 2016-2021. You can use `mutate()` and `case_when()` together.

```
> head(Q12 %>% select(country_name, year, Year_Group),n=15)
```

	country_name	year	Year_Group
1	Mexico	2011	2010-2015
2	Mexico	2012	2010-2015
3	Mexico	2013	2010-2015
4	Mexico	2014	2010-2015
5	Mexico	2015	2010-2015
6	Mexico	2016	2016-2021
7	Mexico	2017	2016-2021
8	Mexico	2018	2016-2021
9	Mexico	2019	2016-2021
10	Mexico	2020	2016-2021
11	Mexico	2021	2016-2021
12	Sweden	2011	2010-2015
13	Sweden	2012	2010-2015
14	Sweden	2013	2010-2015
15	Sweden	2014	2010-2015

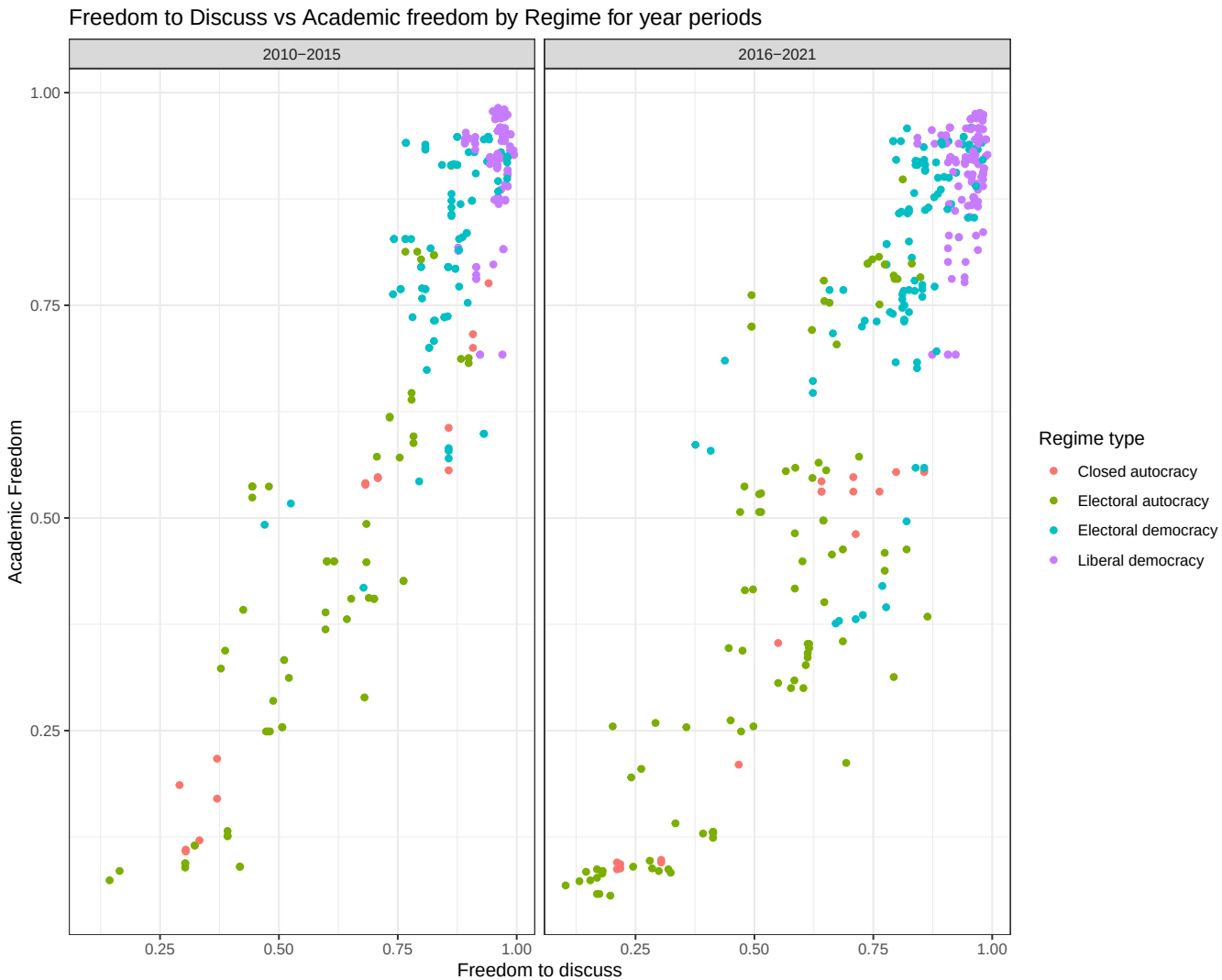
Part B (Due Date: 28/11/2022)

13. **(5 pts)** Change the variable regime into a categorical variable, where
    `0:Closed autocracy`, `1:Electoral autocracy`,
    `2:Electoral democracy`, `3:Liberal democracy`

```
> head(Q13$v2x_regime)
[1] Electoral democracy Electoral democracy Electoral democracy
[4] Electoral democracy Electoral democracy Electoral democracy
4 Levels: Closed autocracy Electoral autocracy ... Liberal democracy
```

Part B (Due Date: 28/11/2022)

14. **(10 pts)** Create the following graph



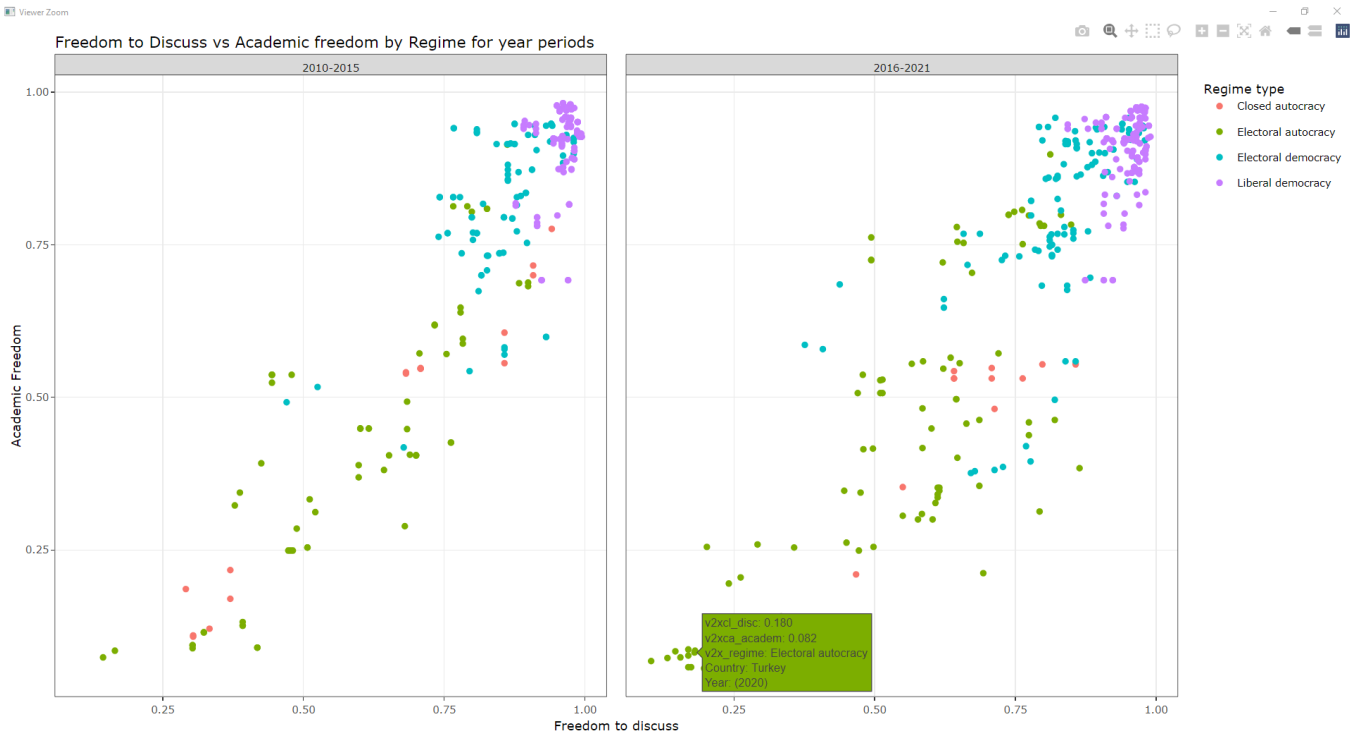
Part B (Due Date: 28/11/2022)

15. **(5 pt)**
    - Convert the graph you created in `Q14` into an interactive graph with `plotly`.
    - You can use the following template to create your own interactive graph.

```
library(ggplot2)
library(plotly)
# After writing your ggplot code assign it to an object
p <- ggplot(...)
# Then use the function plotly to create interactive plot
ggplotly(p)
```

    - By default your graph will give the values for the freedom to discuss, academic freedom and regime type when you hover over a point.
    - To add additional info to the plotly graph, add the argument `text = paste("text1:", var1,"<br1>", "text2:", var2)` inside the `aes` argument in your `ggplot` code.
    - By using the info above add the of the country and the year to the plotly graph.

Part B (Due Date: 28/11/2022)



Part B (Due Date: 28/11/2022)

- For your take home exam to be graded you have to do two thing.
  1. Copy all your codes to the exam answer sheet provided to you. Don't copy outputs, figures etc. to the exam answer sheet. Then print it out, sign and deliver to the lecturer until the deadline. You should bring the take home exam personally as signature will be taken as a proof.
  2. You should also upload your take home exam (programming script + answer sheet) to YTU online campus system until the deadline.
- If you fail to submit any of the item 1 and 2, your take home exam won't be graded.
- You don't have to use R, but it is recommended. You can also use other programming languages like Python, Matlab, Jupyter Notebook etc.
- The announcement date for the take home exam is **14/11/2022 15:00** and deadline for the homework is **28/11/2022 15:00**.
- You can ask your friends or to the lecturer to get help but don't copy somebody else's code.