Research Skills: Programming with R

Assignment 2

This graded set of homework assignments must be handed in on Blackboard before Wednesday January 16th, 5:00 PM. It tests your mastery of Worksheets 4 to 6. It will be graded as follows:

- 0.75 point each for Questions 1 through 7
- 1.50 point each for Questions 8 and 9
- 1.00 point in total for overall code organisation & style
- 0.75 point in total for complying with the instructions below

The guidelines for overall code organisation & style can be found in the Mini-Worksheet and the slides for Class 3. All questions are independent; copy the data sets before modifying them, and start afresh with the originals each time.

Questions 1 through 7 will be graded semi-automatically. Answer them exactly as asked, no deviations or elaborations; any exactly correct answer, irrespective of efficiency, will be worth 0.75 point, and any other, 0 point. Note that this *includes* matching the requested spellings & capitalisations exactly.

For Questions 8 and 9, partial credit will be awarded for partially correct answers; however, for full credit, your solution should not require more code than necessary, given the skills taught in the worksheets.

Other instructions:

- solve all the questions in a single R script
- use `Assigment_2_DemoScript.R`, from Blackboard, as the basis of this script
- load the `development`, `olympics` and `population` data sets exactly as shown in this demo, at the top of your script, and do not adapt the relative paths
- load the `geography` data set at the relevant question, using `read.delim()` and its options as you see fit; make sure to use a relative path
- use any function from 'base R', `dplyr`, `tidyr`, `ggplot2`, and `caret`, and no other packages
- name your script `lastname_u-number_assignment2.R`
- include your name and u-number at the top of your script
- store your final solutions in the objects `answerX` objects as described

This is an individual assignment: I accept that you will discuss it with your fellow students in general terms but directly sharing code is strictly prohibited. Suspected plagiarism will be referred to the Exam Board. Good luck!

Data Set Information

This Assignment concerns four data sets, all summarizing the world's countries; see `countries_data_descriptions.txt`.

Briefly, the `development` data set consists of a subset of the UN's World Development Indicators, collected from 2009 - 2017. The `population` data set is drawn from the same source, but it contains regions as well as countries, and it specifies their total populations for each year between 2009 and 2017.

The `olympics` data set summarizes each country's performance at the 2010 - 2016 Olympics; `Athletes` specifies the number of athletes sent, and `Medals` the number of medals won. The `NOC` column indicates the 'National Olympic Committee' that sent each `Team`; most `NOC`s only sent one, named after the corresponding country.

The `geography` data set was originally compiled from the CIA's World Factbook. It specifies countries' coastlines in kilometers and their climates as indicated by a code, where 1 = dry tropical or tundra and ice; 2 = wet tropical; 3 = temperate and 4 = dry hot summers and wet winters.

Question 1.

Create a function which accepts as its arguments a dataframe and a number. Assume that the dataframe is the `development` data set or a subset of it. The function should return a vector of countries where the `PERC_INTERNET_USERS` is at least equal to the specified number. Create the function with a meaningful name initially, then store it in an object called `answer1`. It should then be possible to call it like this:

```
answer1(development, 95)
```

```
## [1] "Iceland" "Norway"
```

Question 2.

Create a function which accepts as its arguments a dataframe and a string. Assume that the dataframe is the `olympics` data set or a subset of it, and that the string is a valid `NOC` or `Team`. The function should return all rows relating to that `NOC` or `Team`, as applicable, with the corresponding column removed. Create this function with a meaningful name initially, then store it in an object called `answer2`. It should then be possible to call it like this:

```
answer2(olympics, "SWE")
```

```
##     Team Season Year Sex Athletes Medals
## 1 Sweden Winter 2010   F       44      9
## 2 Sweden Winter 2010   M       57      9
## 3 Sweden Summer 2012   F       78      2
## 4 Sweden Summer 2012   M       55     20
## 5 Sweden Winter 2014   F       41     11
## 6 Sweden Winter 2014   M       56     38
## 7 Sweden Summer 2016   F       86     25
## 8 Sweden Summer 2016   M       64      3
```

```
answer2(olympics, "Switzerland-2")
```

```
##   NOC Season Year Sex Athletes Medals
## 1 SUI Winter 2010   F        2      0
## 2 SUI Winter 2010   M        0      0
## 3 SUI Summer 2012   F        0      0
## 4 SUI Summer 2012   M        2      0
## 5 SUI Winter 2014   F        0      0
## 6 SUI Winter 2014   M        2      0
## 7 SUI Summer 2016   F        2      0
## 8 SUI Summer 2016   M        0      0
```

Question 3.

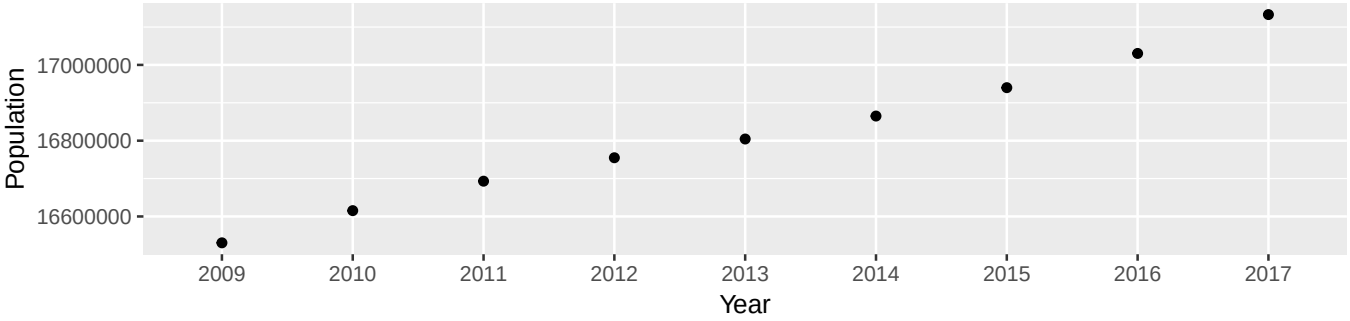
Using 160 characters of code at maximum, create a copy of the `development` data set where all the numeric columns are rounded off to two decimal places. Create this object with a meaningful name initially, then store it in an object called `answer3`. The first three rows of a correct solution look like this:

```
answer3[1:3, ]
```

```
##       COUNTRY GDP_PER_CAP CO2_PER_CAP PERC_ACCESS_ELECTRICITY ATMS_PER_1E5 PERC_INTERNET_USERS
## 1 Afghanistan      599.78        0.32                  64.03         0.78                6.22
## 2     Albania     4369.73        1.72                 100.00        34.58               54.60
## 3     Algeria     4622.60        3.45                  99.07         6.79               23.75
##   SCIENTIFIC_ARTICLES_PER_YR PERC_FEMALE_SECONDARY_EDU PERC_FEMALE_LABOR_FORCE
## 1                      37.42                    34.09                   17.65
## 2                     222.19                    48.99                   53.81
## 3                    3194.55                    51.19                   16.63
##   PERC_FEMALE_PARLIAMENT EQUAL_WORK EQUAL_PAY
## 1                  27.66         NO        NO
## 2                  19.29         NO       YES
## 3                  23.02         NO       YES
```

Question 4.

Using the `population` data set and `ggplot2`, re-create the following plot, where the points represent the population of the `"Netherlands"` over the years 2009 - 2017. Store this plot in an object called `answer4`.



Note: This plot has not been altered from its `ggplot2` defaults, and you do not have to replicate its overall dimensions.

Question 5.

Create a dataframe which consists of the columns shown below; there should be rows for all countries which occur in the `Country`, `COUNTRY` or `Team` columns of the `population`, `development` and `olympics` data sets, respectively. Only include countries with names that match exactly across all three data sets, and ensure they remain sorted alphabetically. Create this object with a meaningful name initially, then store it in an object called `answer5`.

```
answer5[1:3, ]
```

```
##       Country    X2009    X2010    X2011    X2012    X2013    X2014    X2015    X2016    X2017
## 1 Afghanistan 28004331 28803167 29708599 30696958 31731688 32758020 33736494 34656032 35530081
## 2     Albania  2927519  2913021  2905195  2900401  2895092  2889104  2880703  2876101  2873457
## 3     Algeria 35465760 36117637 36819558 37565847 38338562 39113313 39871528 40606052 41318142
##   GDP_PER_CAP CO2_PER_CAP PERC_ACCESS_ELECTRICITY ATMS_PER_1E5 PERC_INTERNET_USERS
## 1    599.7825   0.3188323                 64.03420    0.7774169            6.220034
## 2   4369.7277   1.7164619                100.00000   34.5835023           54.596542
## 3   4622.5990   3.4498462                 99.07253    6.7874244           23.746941
##   SCIENTIFIC_ARTICLES_PER_YR PERC_FEMALE_SECONDARY_EDU PERC_FEMALE_LABOR_FORCE
## 1                    37.4250                  34.08873                17.64856
## 2                   222.1875                  48.98959                53.81178
## 3                  3194.5500                  51.19323                16.62878
##   PERC_FEMALE_PARLIAMENT EQUAL_WORK EQUAL_PAY
## 1               27.65556         NO        NO
## 2               19.28889         NO       YES
## 3               23.02222         NO       YES
```

Question 6.

Using the `development` data set and `train()`, fit a logistic regression model using 10-fold cross validation; otherwise stick to the defaults. It should predict `EQUAL_WORK` based on all other variables except for `EQUAL_PAY` and `COUNTRY`. Use `set.seed(1)` before fitting this model. Store the maximum accuracy across folds in an object called `answer6`.

Question 7.

Using `caret`, create the `trn_develop` and `tst_develop` objects used in the code below. Call `set.seed(1)` before answering this question. 60% of all data in the `development` data set should be in `trn_develop`, the rest in `tst_develop`; the `EQUAL_PAY` variable should be distributed equally across the split. Ensure that the code as shown runs without problems, giving the same result. Store these objects as `answer7_trn` and `answer7_tst`, respectively.

```
pay_knn = train(EQUAL_PAY ~ GDP_PER_CAP, method = "knn", data = trn_develop,
  trControl = trainControl(method = 'cv', number = 5))
predicted_outcomes <- predict(pay_knn, tst_develop)
pay_confM <- confusionMatrix(predicted_outcomes, tst_develop$EQUAL_PAY)
pay_confM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction YES NO
##        YES  12 17
##        NO   16 20
##
##                Accuracy : 0.4923
##                  95% CI : (0.366, 0.6193)
##     No Information Rate : 0.5692
##     P-Value [Acc > NIR] : 0.9153
##
##                   Kappa : -0.0308
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.4286
##             Specificity : 0.5405
##          Pos Pred Value : 0.4138
##          Neg Pred Value : 0.5556
##              Prevalence : 0.4308
##          Detection Rate : 0.1846
##    Detection Prevalence : 0.4462
##       Balanced Accuracy : 0.4846
##
##        'Positive' Class : YES
##
```

Question 8.

Using the `olympics` and `geography` data sets, create a dataframe with columns `Country`, `Season`, `Medals`, `Coastline_KM` and `Climate`; the first six rows of a correct solution are shown below. For each country in the `olympics` data set, this dataframe should list the total number of medals won in the `"Winter"` and `"Summer"` olympics, with each `Season` in its own row, followed by the correct sum for `Medals`, and the matching `Coastline_KM` and `Climate` taken from the `geography` data set, which you will need to read into R specifically for this question.

```
answer8[1:6, ]
```

```
##       Country Season Medals Coastline_KM Climate
## 1 Afghanistan Summer      1            0       1
## 2 Afghanistan Winter      0            0       1
## 3     Albania Summer      0          362       3
## 4     Albania Winter      0          362       3
## 5     Algeria Summer      3          998       1
## 6     Algeria Winter      0          998       1
```

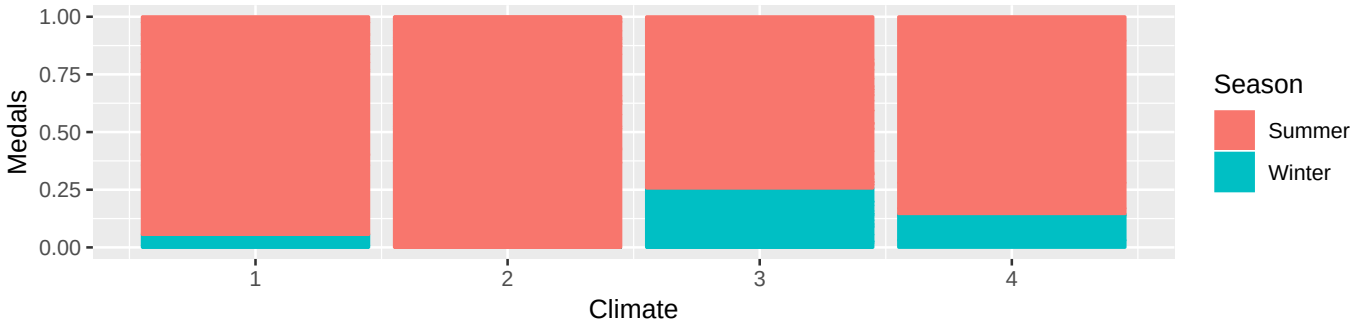
In this dataframe, the different `Teams` of each country should all be condensed into one row per country; i.e., there should be one row for `Argentina`, summing up all medals won by `Argentina`, `Argentina-1` and `Argentina-2`.

Not all countries in `olympics` have identical counterparts in `geography`; i.e., `olympics` contains `"Bahamas"` while `geography` calls it `"Bahamas, The"`. For the most part you may disregard these differences, and add in only `geography` information from country names that match exactly, but make sure your data set contains the `Coastline_KM` and `Climate` of both Koreas, as well as `"Great Britain"`, which you should match to `"United Kingdom"`.

Create this dataframe with a meaningful name initially, then store it in an object called `answer8`. One use of this dataframe would be to compare the relative number of medals won by countries with different `Climates`:

```
ggplot(answer8, aes(x = Climate, y = Medals, fill = Season, col = Season)) +
  geom_bar(stat = "identity", position = "fill")
```

```
## Warning: Removed 68 rows containing missing values (position_stack).
```



Note: You do not have to do anything with this graph, it is for illustrative purposes only!

Question 9.

Create a copy of the `development` data set where all `PERC_FEMALE_...` columns are cut into 3 discrete classes: Under 40%, the column should read `"UNDER_REPRESENTED"`, between 40% and 60% (inclusive), the column should read `"APPROX_EQUAL"`, and otherwise it should read `"OVER_REPRESENTED"`. The first three rows should look like this:

```
dev_discrete[1:3, ]
```

```
##       COUNTRY GDP_PER_CAP CO2_PER_CAP PERC_ACCESS_ELECTRICITY ATMS_PER_1E5 PERC_INTERNET_USERS
## 1 Afghanistan    599.7825   0.3188323                64.03420    0.7774169            6.220034
## 2     Albania   4369.7277   1.7164619               100.00000   34.5835023           54.596542
## 3     Algeria   4622.5990   3.4498462                99.07253    6.7874244           23.746941
##   SCIENTIFIC_ARTICLES_PER_YR PERC_FEMALE_SECONDARY_EDU PERC_FEMALE_LABOR_FORCE
## 1                    37.4250          UNDER_REPRESENTED       UNDER_REPRESENTED
## 2                   222.1875               APPROX_EQUAL            APPROX_EQUAL
## 3                  3194.5500               APPROX_EQUAL       UNDER_REPRESENTED
##   PERC_FEMALE_PARLIAMENT EQUAL_WORK EQUAL_PAY
## 1      UNDER_REPRESENTED         NO        NO
## 2      UNDER_REPRESENTED         NO       YES
## 3      UNDER_REPRESENTED         NO       YES
```

Use this dataframe to fit a `"knn"` model using 3-fold cross validation; center and scale the variables, and try out all uneven values for k ranging from 3 to 21; otherwise stick to the defaults. The model should predict `PERC_FEMALE_LABOR_FORCE` using the variables `GDP_PER_CAP`, `CO2_PER_CAP`, `PERC_INTERNET_USERS` and `SCIENTIFIC_ARTICLES_PER_YR`. Use `set.seed(1)` before fitting this model, and store it an object called `answer9`.

Note: If you cannot create the requested dataframe, fit the `"knn"` model as described predicting `EQUAL_WORK` instead.