Data Wrangling (Data Preprocessing) Code

Practical assessment 1

Student name submitting the assessment report come here

Setup

Hide

```
# Load packages
library(kableExtra)
library(magrittr)
library(readr)
library(dplyr)
```

Student names, numbers and percentage of contributions

Group information

Student name	Student number	Percentage of contribution
Yuan-Zhi, Cai	s3886716	50%
Sukhum Boondecharak	s3940976	50%

Data Description

- Victorian water fluoridation status by postcode
- The Victorian Government is committed to improving the oral health of Victorians. A person's oral health is key to their overall general health and wellbeing. Extending community water fluoridation is important for public health. Melbourne has had fluoridated water since 1977. Other parts of Australia have had fluoridated drinking water for more than 50 years. Community water fluoridation is the most effective population-wide intervention to prevent tooth decay.
- URL of the website: https://docs.google.com/spreadsheets/d/143x2m4VvbJXKURxtrCnXjn7470kzQ109xRMq4wa2XoY/export?format=csv&id=143x2m4VvbJXKURxtrCnXjn7470kzQ109xRMq4wa2XoY&gid=0 (https://docs.google.com/spreadsheets/d/143x2m4VvbJXKURxtrCnXjn7470kzQ109xRMq4wa2XoY/export?format=csv&id=143x2m4VvbJXKURxtrCnXjn7470kzQ109xRMq4wa2XoY&gid=0)
- There are nine columns, which contain three integers, one logic, three strings, and two numeric.

Read/Import Data

Hide

```
#Read/Import Data
data_vic <- read.csv("Fluoridation by postcode - Sheet1.csv", stringsAsFactors = TRUE ) #Impo
rt data and save as data frame

head(data_vic, n = 10) # Check from the first observation up to the row we want to see by ass
igning n = 10
```

	cartodb_id <int>	melbourne <lgl>	postcode <int>	id <int>	fluoride_level <fctr>	water_company <fctr>
1	236	FALSE	3227	236	Fluoridated	Barwon Water
2	1	TRUE	3067	1	Fluoridated	City West Water
3	2	TRUE	3040	2	Fluoridated	City West Water
4	3	FALSE	3352	3	Not fluoridated	Central Highlands Water
5	4	FALSE	3465	4	Fluoridated	Central Highlands Water
6	5	FALSE	3962	5	Not fluoridated	South Gippsland Water
7	6	FALSE	3231	6	Fluoridated	Barwon Water
8	7	TRUE	3042	7	Fluoridated	City West Water
9	8	TRUE	3021	8	Fluoridated	City West Water
10	9	TRUE	3206	9	Fluoridated	South East Water

1-10 of 10 rows | 1-7 of 9 columns

Hide

```
tail(data_vic, n = 10) # Check from the bottom observation up to the row we want to see by as
signing n = 10
```

	cartodb_id <int>	melbourne <lgl>	postcode <int>	id <int>	fluoride_level <fctr>
1050	1050	TRUE	3013	1050	Fluoridated
1051	1051	FALSE	3730	1051	Fluoridated
1052	1052	FALSE	3644	1052	Not fluoridated
1053	1053	FALSE	3717	1053	Not fluoridated
1054	1054	TRUE	3139	1054	Partly fluoridated
1055	1055	FALSE	3352	1055	Fluoridated
1056	1056	TRUE	3770	1056	Fluoridated
1057	1057	FALSE	3869	1057	Fluoridated
1058	1058	TRUE	3063	1058	Fluoridated
1059	1059	FALSE	3418	1059	Not fluoridated

1-10 of 10 rows | 1-6 of 9 columns

Hide

```
data_vic <- arrange(data_vic, cartodb_id) #rearrange the ID order because it should not be 23
6 at the first row by ID

#Check the arrange is worked.
data_vic$cartodb_id[1] #Only focus on rearrange data.
```

```
[1] 1
```

Hide

```
data_vic$cartodb_id[236] #Only focus on rearrange data.
```

```
[1] 236
```

Import data and assign it to a variable called “data.vic”. We decided to use Base R instead readr package because we need to use straingsAsFactors = TRUE for levels() function.

We put the .R file and .csv file in the same working dictionary.

We noticed the id is not inorder so we used arrange() function to correct it.

Inspect and Understand

Hide

```
#Inspect and Understand
attributes(data_vic) # Check the dimensions of the data frame
```

```
$names
[1] "cartodb_id"     "melbourne"      "postcode"
[4] "id"             "fluoride_level" "water_company"
[7] "townsuburb"     "lat"            "lon"

$class
[1] "data.frame"

$row.names
   [1]    1    2    3    4    5    6    7    8    9
  [10]   10   11   12   13   14   15   16   17   18
  [19]   19   20   21   22   23   24   25   26   27
  [28]   28   29   30   31   32   33   34   35   36
  [37]   37   38   39   40   41   42   43   44   45
  [46]   46   47   48   49   50   51   52   53   54
  [55]   55   56   57   58   59   60   61   62   63
  [64]   64   65   66   67   68   69   70   71   72
  [73]   73   74   75   76   77   78   79   80   81
  [82]   82   83   84   85   86   87   88   89   90
  [91]   91   92   93   94   95   96   97   98   99
 [100]  100  101  102  103  104  105  106  107  108
 [109]  109  110  111  112  113  114  115  116  117
 [118]  118  119  120  121  122  123  124  125  126
 [127]  127  128  129  130  131  132  133  134  135
 [136]  136  137  138  139  140  141  142  143  144
 [145]  145  146  147  148  149  150  151  152  153
 [154]  154  155  156  157  158  159  160  161  162
 [163]  163  164  165  166  167  168  169  170  171
 [172]  172  173  174  175  176  177  178  179  180
 [181]  181  182  183  184  185  186  187  188  189
 [190]  190  191  192  193  194  195  196  197  198
 [199]  199  200  201  202  203  204  205  206  207
 [208]  208  209  210  211  212  213  214  215  216
 [217]  217  218  219  220  221  222  223  224  225
 [226]  226  227  228  229  230  231  232  233  234
 [235]  235  236  237  238  239  240  241  242  243
 [244]  244  245  246  247  248  249  250  251  252
 [253]  253  254  255  256  257  258  259  260  261
 [262]  262  263  264  265  266  267  268  269  270
 [271]  271  272  273  274  275  276  277  278  279
 [280]  280  281  282  283  284  285  286  287  288
 [289]  289  290  291  292  293  294  295  296  297
 [298]  298  299  300  301  302  303  304  305  306
 [307]  307  308  309  310  311  312  313  314  315
 [316]  316  317  318  319  320  321  322  323  324
 [325]  325  326  327  328  329  330  331  332  333
 [334]  334  335  336  337  338  339  340  341  342
 [343]  343  344  345  346  347  348  349  350  351
 [352]  352  353  354  355  356  357  358  359  360
 [361]  361  362  363  364  365  366  367  368  369
 [370]  370  371  372  373  374  375  376  377  378
 [379]  379  380  381  382  383  384  385  386  387
 [388]  388  389  390  391  392  393  394  395  396
 [397]  397  398  399  400  401  402  403  404  405
 [406]  406  407  408  409  410  411  412  413  414
 [415]  415  416  417  418  419  420  421  422  423
 [424]  424  425  426  427  428  429  430  431  432
```

```
[433]  433  434  435  436  437  438  439  440  441
[442]  442  443  444  445  446  447  448  449  450
[451]  451  452  453  454  455  456  457  458  459
[460]  460  461  462  463  464  465  466  467  468
[469]  469  470  471  472  473  474  475  476  477
[478]  478  479  480  481  482  483  484  485  486
[487]  487  488  489  490  491  492  493  494  495
[496]  496  497  498  499  500  501  502  503  504
[505]  505  506  507  508  509  510  511  512  513
[514]  514  515  516  517  518  519  520  521  522
[523]  523  524  525  526  527  528  529  530  531
[532]  532  533  534  535  536  537  538  539  540
[541]  541  542  543  544  545  546  547  548  549
[550]  550  551  552  553  554  555  556  557  558
[559]  559  560  561  562  563  564  565  566  567
[568]  568  569  570  571  572  573  574  575  576
[577]  577  578  579  580  581  582  583  584  585
[586]  586  587  588  589  590  591  592  593  594
[595]  595  596  597  598  599  600  601  602  603
[604]  604  605  606  607  608  609  610  611  612
[613]  613  614  615  616  617  618  619  620  621
[622]  622  623  624  625  626  627  628  629  630
[631]  631  632  633  634  635  636  637  638  639
[640]  640  641  642  643  644  645  646  647  648
[649]  649  650  651  652  653  654  655  656  657
[658]  658  659  660  661  662  663  664  665  666
[667]  667  668  669  670  671  672  673  674  675
[676]  676  677  678  679  680  681  682  683  684
[685]  685  686  687  688  689  690  691  692  693
[694]  694  695  696  697  698  699  700  701  702
[703]  703  704  705  706  707  708  709  710  711
[712]  712  713  714  715  716  717  718  719  720
[721]  721  722  723  724  725  726  727  728  729
[730]  730  731  732  733  734  735  736  737  738
[739]  739  740  741  742  743  744  745  746  747
[748]  748  749  750  751  752  753  754  755  756
[757]  757  758  759  760  761  762  763  764  765
[766]  766  767  768  769  770  771  772  773  774
[775]  775  776  777  778  779  780  781  782  783
[784]  784  785  786  787  788  789  790  791  792
[793]  793  794  795  796  797  798  799  800  801
[802]  802  803  804  805  806  807  808  809  810
[811]  811  812  813  814  815  816  817  818  819
[820]  820  821  822  823  824  825  826  827  828
[829]  829  830  831  832  833  834  835  836  837
[838]  838  839  840  841  842  843  844  845  846
[847]  847  848  849  850  851  852  853  854  855
[856]  856  857  858  859  860  861  862  863  864
[865]  865  866  867  868  869  870  871  872  873
[874]  874  875  876  877  878  879  880  881  882
[883]  883  884  885  886  887  888  889  890  891
[892]  892  893  894  895  896  897  898  899  900
[901]  901  902  903  904  905  906  907  908  909
[910]  910  911  912  913  914  915  916  917  918
[919]  919  920  921  922  923  924  925  926  927
[928]  928  929  930  931  932  933  934  935  936
[937]  937  938  939  940  941  942  943  944  945
[946]  946  947  948  949  950  951  952  953  954
```

```
 [955]  955  956  957  958  959  960  961  962  963
 [964]  964  965  966  967  968  969  970  971  972
 [973]  973  974  975  976  977  978  979  980  981
 [982]  982  983  984  985  986  987  988  989  990
 [991]  991  992  993  994  995  996  997  998  999
[1000] 1000
 [ reached getOption("max.print") -- omitted 59 entries ]
```

Hide

```
str(data_vic) # Check the dimension of the data frame, look at the number of observations, va
riables, name and class of variable
```

```
'data.frame':   1059 obs. of  9 variables:
 $ cartodb_id    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ melbourne     : logi  TRUE TRUE FALSE FALSE FALSE FALSE ...
 $ postcode      : int  3067 3040 3352 3465 3962 3231 3042 3021 3206 3971 ...
 $ id            : int  1 2 3 4 5 6 7 8 9 10 ...
 $ fluoride_level: Factor w/ 4 levels "Fluoridated",..: 1 1 3 1 3 1 1 1 1 3 ...
 $ water_company : Factor w/ 18 levels "Barwon Water",..: 3 3 2 2 14 1 3 3 13 14 ...
 $ townsuburb    : Factor w/ 1046 levels "ABBOTSFORD","ABERFELDIE",..: 1 2 3 4 5 6 7 8 9 10
...
 $ lat           : num  -37.8 -37.8 -37.4 -37.1 -38.7 ...
 $ lon           : num  145 145 144 144 146 ...
```

Hide

```
colnames(data_vic) # Check the column names
```

```
[1] "cartodb_id"     "melbourne"      "postcode"
[4] "id"             "fluoride_level" "water_company"
[7] "townsuburb"     "lat"            "lon"
```

Hide

```
names(data_vic)[6] <- "company" # Rename the column "water_company" to "company"

#check datatype of columns
#check whether a numeric object is integer or double.
typeof(data_vic$cartodb_id)
```

```
[1] "integer"
```

Hide

```
typeof(data_vic$postcode)
```

```
[1] "integer"
```

Hide

```
typeof(data_vic$id)
```

```
[1] "integer"
```

Hide

```
typeof(data_vic$lat)
```

```
[1] "double"
```

Hide

```
typeof(data_vic$lon)
```

```
[1] "double"
```

Hide

```
#check the class of the objects
class(data_vic$melbourne)
```

```
[1] "logical"
```

Hide

```
class(data_vic$fluoride_level)
```

```
[1] "factor"
```

Hide

```
class(data_vic$company)
```

```
[1] "factor"
```

Hide

```
class(data_vic$townsuburb)
```

```
[1] "factor"
```

Hide

```
#check the levels of factor variables
levels(data_vic$fluoride_level)
```

```
[1] "Fluoridated"        "Natural fluoride"
[3] "Not fluoridated"    "Partly fluoridated"
```

Hide

```
levels(data_vic$company)
```

```
 [1] "Barwon Water"
 [2] "Central Highlands Water"
 [3] "City West Water"
 [4] "Coliban Water"
 [5] "East Gippsland Water"
 [6] "Gippsland Water"
 [7] "Goulburn Valley Water"
 [8] "GWMWater"
 [9] "Lower Murray Water"
[10] "Mount Buller and Mount Stirling ARMB"
[11] "Mount Hotham ARMB"
[12] "North East Water"
[13] "South East Water"
[14] "South Gippsland Water"
[15] "Wannon Water"
[16] "Western Water"
[17] "Westernport Water"
[18] "Yarra Valley Water"
```

We only use attributes() to check the attribution, which contains the class() function for data frame so we didn't use class() again.

We use structure to make sure the number of observations and variables correctly.

We use typeof() function to check number is integer or doulbe.

We use class() function to check class of an object.

We use levels to check how many repeat words in the same column. We used stringsAsFactors so strings don't have to convert to factor, we can use levels directly.

Subsetting

Hide

```
#Subsetting
matrix_vic <- as.matrix(data_vic[1:10, 1:9]) #convert list to a matrix

rownames(matrix_vic) <- c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10") #Name the row of
the matrix
str(matrix_vic) #Check the structure of the matrix
```

```
 chr [1:10, 1:9] " 1" " 2" " 3" " 4" " 5" " 6" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:10] "1" "2" "3" "4" ...
  ..$ : chr [1:9] "cartodb_id" "melbourne" "postcode" "id" ...
```

Hide

```
attributes(matrix_vic) #Check the attributes of the matrix
```

```
$dim
[1] 10  9

$dimnames
$dimnames[[1]]
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"

$dimnames[[2]]
[1] "cartodb_id"     "melbourne"      "postcode"
[4] "id"             "fluoride_level" "company"
[7] "townsuburb"     "lat"            "lon"
```

Hide

```
class(matrix_vic) #Check the class of the matrix
```

```
[1] "matrix" "array"
```

Hide

```
matrix_vic #Make sure it's correct.
```

```
   cartodb_id melbourne postcode id
1  " 1"       "TRUE"    "3067"   " 1"
2  " 2"       "TRUE"    "3040"   " 2"
3  " 3"       "FALSE"   "3352"   " 3"
4  " 4"       "FALSE"   "3465"   " 4"
5  " 5"       "FALSE"   "3962"   " 5"
6  " 6"       "FALSE"   "3231"   " 6"
7  " 7"       "TRUE"    "3042"   " 7"
8  " 8"       "TRUE"    "3021"   " 8"
9  " 9"       "TRUE"    "3206"   " 9"
10 "10"       "FALSE"   "3971"   "10"
   fluoride_level    company
1  "Fluoridated"     "City West Water"
2  "Fluoridated"     "City West Water"
3  "Not fluoridated" "Central Highlands Water"
4  "Fluoridated"     "Central Highlands Water"
5  "Not fluoridated" "South Gippsland Water"
6  "Fluoridated"     "Barwon Water"
7  "Fluoridated"     "City West Water"
8  "Fluoridated"     "City West Water"
9  "Fluoridated"     "South East Water"
10 "Not fluoridated" "South Gippsland Water"
   townsuburb      lat         lon
1  "ABBOTSFORD"    "-37.80300" "145.0020"
2  "ABERFELDIE"    "-37.76200" "144.9010"
3  "ADDINGTON"     "-37.38333" "143.6833"
4  "ADELAIDE LEAD" "-37.08382" "143.6791"
5  "AGNES"         "-38.65778" "146.3820"
6  "AIREYS INLET"  "-38.45071" "144.1089"
7  "AIRPORT WEST"  "-37.72400" "144.8790"
8  "ALBANVALE"     "-37.74600" "144.7650"
9  "ALBERT PARK"   "-37.84200" "144.9500"
10 "ALBERTON"      "-38.61667" "146.6667"
```

We use as.matrix to convert data frame to matrix. By naming the row of the matrix, we can avoid NULL situation when using the function of structure and attributes.

Create a new Data Frame

Hide

```
#Create a new Data Frame
df_vic <- data.frame(vic_id = data_vic$id[1:10],
                     vic_postcode = data_vic$postcode[1:10]) # create a new data frame from d
ata_vic
numeric_vector <- c (1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10.1) #Create a new numeric
vector
df_vic <- cbind(df_vic, numeric_vector) #combine data frame and new vector.
df_vic #Check the new data frame is worked.
```

vic_id <int>	vic_postcode <int>	numeric_vector <dbl>
1	3067	1.1
2	3040	2.2
3	3352	3.3
4	3465	4.4
5	3962	5.5
6	3231	6.6
7	3042	7.7
8	3021	8.8
9	3206	9.9
10	3971	10.1

1-10 of 10 rows

data.frame() function to create a new data frame.