An R Markdown document converted from Applying_Fun_R.ipynb Homework Week 2: Applying Functions in R

Amogh Guthur

Applying functions in R

This notebook demonstrates various ways to apply functions in R, including built-in functions and custom functions.

Installing packages

```
# Install the 'rio' package for reading various file formats including Excel
# Commented out because it only needs to be run once
# install.packages('rio')
```

Reading data

```
# Define the URL pointing to the Excel file on GitHub
# This file contains the Fragile States Index data for 2023
linkGit="https://github.com/FundamentalsAmogh/week2_hw/raw/refs/heads/main/FSI-2023-DOWN
LOAD.xlsx"

# Use rio::import() to read the Excel file directly from the GitHub URL
# The :: syntax lets us use a function without loading the entire library
# The result is stored in a data frame called 'fragility23'
fragility23=rio::import(file = linkGit)
```

```
# Display the structure of the data frame
# str() shows the data types, number of observations, and sample values for each column
str(fragility23)
```

```
## 'data.frame':    179 obs. of  16 variables:
##  $ Country                          : chr  "Somalia" "Yemen" "South Sudan" "Congo Demo
cratic Republic" ...
##  $ Year                             : num  2023 2023 2023 2023 2023 ...
##  $ Rank                             : chr  "1st" "2nd" "3rd" "4th" ...
##  $ Total                            : num  112 109 109 107 107 ...
##  $ S1: Demographic Pressures        : num  10 9.6 9.7 9.7 7.4 9.2 8.8 9.3 9.5 8.8 ...
##  $ S2: Refugees and IDPs            : num  9 9.6 10 9.8 9.1 8.6 9.6 9.5 9 7.7 ...
##  $ C3: Group Grievance              : num  8.7 8.8 8.6 9.4 9.1 8.3 9.3 8.1 8.1 5.5 ...
##  $ E3: Human Flight and Brain Drain: num  8.6 6.4 6.5 6.4 8 8.5 7.5 6.2 7.7 8.3 ...
##  $ E2: Economic Inequality          : num  9.1 7.9 8.6 8.4 6.5 8.2 8.5 9.6 8.7 9.2 ...
##  $ E1: Economy                      : num  9.5 9.9 8.6 8.1 9.6 9.6 9.3 8.2 8.4 8.9 ...
##  $ P1: State Legitimacy             : num  9.6 9.8 9.8 9.3 10 9.4 9.4 8.9 9.1 9.9 ...
##  $ P2: Public Services              : num  9.8 9.6 9.7 9.3 9 10 8.6 10 9.6 9.8 ...
##  $ P3: Human Rights                 : num  9 9.6 8.7 9.3 9.1 8.7 9.2 9.1 8.4 8.7 ...
##  $ C1: Security Apparatus           : num  9.5 8.6 9.9 8.8 9.4 9.7 8.3 8 8.7 6.8 ...
##  $ C2: Factionalized Elites         : num  10 9.9 9.2 9.6 9.9 8.7 9.6 9.4 9.5 9.7 ...
##  $ X1: External Intervention        : num  9.1 9.2 9.2 9.1 10 7.7 8.1 9.4 7.9 9.6 ...
```

Apply square root function?

```
# Attempt to apply sqrt() to the entire data frame
# This will produce an error because sqrt() only works on numeric data
# The data frame contains character columns (Country, Rank) that can't be square-rooted
# The error=TRUE chunk option allows knitting to continue despite the error
sqrt(fragility23)
```

```
## Error in `Math.data.frame()`:
## ! non-numeric-alike variable(s) in data frame: Country, Rank
```

```
# Apply sqrt() to only columns 4 and 5, which are numeric
# The [,4:5] syntax selects all rows and only columns 4 through 5
# This works because we're only selecting numeric columns
sqrt(fragility23[,4:5])
```

```
##         Total S1: Demographic Pressures
## 1   10.578280                  3.162278
## 2   10.435516                  3.098387
## 3   10.416333                  3.114482
## 4   10.353743                  3.114482
## 5   10.348913                  2.720294
## 6   10.324728                  3.033150
## 7   10.305338                  2.966479
## 8   10.281051                  3.049590
## 9   10.227414                  3.082207
## 10  10.143964                  2.966479
## 11  10.019980                  3.130495
## 12  10.009995                  2.645751
## 13   9.974969                  2.966479
## 14   9.924717                  2.966479
## 15   9.899495                  3.098387
## 16   9.843780                  2.949576
## 17   9.803061                  2.449490
## 18   9.792855                  2.701851
## 19   9.721111                  2.774887
## 20   9.705668                  2.949576
## 21   9.695360                  2.880972
## 22   9.695360                  3.098387
## 23   9.695360                  3.000000
## 24   9.664368                  2.983287
## 25   9.581232                  2.408319
## 26   9.565563                  3.033150
## 27   9.560335                  2.863564
## 28   9.523655                  2.915476
## 29   9.513149                  2.549510
## 30   9.502631                  2.683282
## 31   9.481561                  2.983287
## 32   9.481561                  2.880972
## 33   9.428680                  2.898275
## 34   9.375500                  2.880972
## 35   9.370165                  2.846050
## 36   9.332738                  2.932576
## 37   9.327379                  2.932576
## 38   9.327379                  2.701851
## 39   9.322017                  3.049590
## 40   9.241212                  2.607681
## 41   9.230385                  2.774887
## 42   9.186947                  3.016621
## 43   9.121403                  3.098387
## 44   9.071935                  2.683282
## 45   9.066422                  2.863564
## 46   9.066422                  2.664583
## 47   9.060905                  2.720294
## 48   9.044335                  3.065942
## 49   9.038805                  3.098387
## 50   9.033272                  2.588436
## 51   9.022195                  2.898275
```

```
## 52   9.011104                2.121320
## 53   8.983318                2.190890
## 54   8.961027                2.529822
## 55   8.955445                2.846050
## 56   8.921883                2.664583
## 57   8.921883                2.626785
## 58   8.893818                2.949576
## 59   8.837420                2.792848
## 60   8.837420                2.720294
## 61   8.820431                2.810694
## 62   8.814760                2.236068
## 63   8.803408                2.949576
## 64   8.792042                2.664583
## 65   8.752143                2.932576
## 66   8.734987                2.932576
## 67   8.723531                2.863564
## 68   8.700575                2.509980
## 69   8.694826                2.387467
## 70   8.642916                2.529822
## 71   8.631338                2.949576
## 72   8.613942                2.645751
## 73   8.608136                2.915476
## 74   8.561542                2.828427
## 75   8.549854                2.756810
## 76   8.526429                1.949359
## 77   8.502941                2.469818
## 78   8.485281                2.828427
## 79   8.479387                2.167948
## 80   8.455767                2.720294
## 81   8.408329                2.626785
## 82   8.378544                2.144761
## 83   8.366600                2.236068
## 84   8.360622                2.190890
## 85   8.354639                2.626785
## 86   8.348653                2.626785
## 87   8.330666                2.588436
## 88   8.324662                2.408319
## 89   8.324662                2.607681
## 90   8.258329                2.097618
## 91   8.246211                2.469818
## 92   8.240146                2.024846
## 93   8.215838                1.949359
## 94   8.209750                2.323790
## 95   8.173127                2.073644
## 96   8.148620                2.323790
## 97   8.148620                2.073644
## 98   8.099383                2.645751
## 99   8.093207                2.588436
## 100  8.080842                2.073644
## 101  8.068457                2.167948
## 102  8.068457                2.024846
## 103  8.068457                2.549510
```

```
## 104  8.031189                2.366432
## 105  7.981228                2.387467
## 106  7.930952                2.280351
## 107  7.893035                2.720294
## 108  7.867655                2.190890
## 109  7.848567                2.408319
## 110  7.797435                2.529822
## 111  7.784600                2.024846
## 112  7.765307                2.846050
## 113  7.765307                2.049390
## 114  7.752419                2.489980
## 115  7.733046                2.236068
## 116  7.726578                2.345208
## 117  7.713624                2.428992
## 118  7.635444                2.097618
## 119  7.615773                2.024846
## 120  7.549834                1.843909
## 121  7.536577                2.024846
## 122  7.436397                2.792848
## 123  7.422937                2.024846
## 124  7.416198                2.258318
## 125  7.395945                1.673320
## 126  7.334848                1.923538
## 127  7.328028                2.024846
## 128  7.300685                2.049390
## 129  7.280110                2.121320
## 130  7.273239                2.024846
## 131  7.197222                2.323790
## 132  7.162402                2.190890
## 133  7.155418                2.097618
## 134  7.014271                2.509980
## 135  6.985700                1.870829
## 136  6.978539                2.213594
## 137  6.978539                2.049390
## 138  6.978539                2.190890
## 139  6.811755                2.387467
## 140  6.737952                1.923538
## 141  6.730527                2.280351
## 142  6.723095                1.949359
## 143  6.640783                2.190890
## 144  6.595453                1.816590
## 145  6.580274                1.788854
## 146  6.526868                2.144761
## 147  6.496153                2.428992
## 148  6.473021                1.843909
## 149  6.363961                1.732051
## 150  6.356099                1.702939
## 151  6.340347                1.788854
## 152  6.276942                1.897367
## 153  6.212890                1.702939
## 154  6.164414                1.760682
## 155  6.148170                1.612452
```

```
## 156  6.082763                  1.897367
## 157  5.865151                  1.843909
## 158  5.744563                  1.923538
## 159  5.612486                  1.581139
## 160  5.603570                  2.049390
## 161  5.522681                  2.428992
## 162  5.366563                  1.760682
## 163  5.224940                  1.897367
## 164  5.069517                  2.000000
## 165  5.049752                  1.581139
## 166  4.959839                  1.516575
## 167  4.939636                  1.897367
## 168  4.690416                  1.949359
## 169  4.582576                  1.581139
## 170  4.538722                  1.732051
## 171  4.415880                  1.549193
## 172  4.415880                  1.673320
## 173  4.347413                  1.095445
## 174  4.230839                  1.516575
## 175  4.219005                  1.549193
## 176  4.086563                  1.048809
## 177  4.000000                  1.303840
## 178  3.962323                  1.224745
## 179  3.807887                  1.183216
```

```
# Apply sqrt() to a single column using the $ accessor
# fragility23$Total extracts the 'Total' column as a vector
# sqrt() is then applied element-wise to each value in that vector
sqrt(fragility23$Total)
```

```
##   [1] 10.578280 10.435516 10.416333 10.353743 10.348913 10.324728 10.305338
##   [8] 10.281051 10.227414 10.143964 10.019980 10.009995  9.974969  9.924717
##  [15]  9.899495  9.843780  9.803061  9.792855  9.721111  9.705668  9.695360
##  [22]  9.695360  9.695360  9.664368  9.581232  9.565563  9.560335  9.523655
##  [29]  9.513149  9.502631  9.481561  9.481561  9.428680  9.375500  9.370165
##  [36]  9.332738  9.327379  9.327379  9.322017  9.241212  9.230385  9.186947
##  [43]  9.121403  9.071935  9.066422  9.066422  9.060905  9.044335  9.038805
##  [50]  9.033272  9.022195  9.011104  8.983318  8.961027  8.955445  8.921883
##  [57]  8.921883  8.893818  8.837420  8.837420  8.820431  8.814760  8.803408
##  [64]  8.792042  8.752143  8.734987  8.723531  8.700575  8.694826  8.642916
##  [71]  8.631338  8.613942  8.608136  8.561542  8.549854  8.526429  8.502941
##  [78]  8.485281  8.479387  8.455767  8.408329  8.378544  8.366600  8.360622
##  [85]  8.354639  8.348653  8.330666  8.324662  8.324662  8.258329  8.246211
##  [92]  8.240146  8.215838  8.209750  8.173127  8.148620  8.148620  8.099383
##  [99]  8.093207  8.080842  8.068457  8.068457  8.068457  8.031189  7.981228
## [106]  7.930952  7.893035  7.867655  7.848567  7.797435  7.784600  7.765307
## [113]  7.765307  7.752419  7.733046  7.726578  7.713624  7.635444  7.615773
## [120]  7.549834  7.536577  7.436397  7.422937  7.416198  7.395945  7.334848
## [127]  7.328028  7.300685  7.280110  7.273239  7.197222  7.162402  7.155418
## [134]  7.014271  6.985700  6.978539  6.978539  6.978539  6.811755  6.737952
## [141]  6.730527  6.723095  6.640783  6.595453  6.580274  6.526868  6.496153
## [148]  6.473021  6.363961  6.356099  6.340347  6.276942  6.212890  6.164414
## [155]  6.148170  6.082763  5.865151  5.744563  5.612486  5.603570  5.522681
## [162]  5.366563  5.224940  5.069517  5.049752  4.959839  4.939636  4.690416
## [169]  4.582576  4.538722  4.415880  4.415880  4.347413  4.230839  4.219005
## [176]  4.086563  4.000000  3.962323  3.807887
```

```
# Apply sqrt() to just the first value of the Total column
# The [1] index extracts only the first element
# This returns a single numeric value
sqrt(fragility23$Total[1])
```

```
## [1] 10.57828
```

Applying sum():

```
# Sum all values in columns 4 and 5 together
# This produces a single total by adding every numeric value in both columns
sum(fragility23[,4:5])
```

```
## [1] 12850.1
```

```
# Use apply() to sum each column separately
# First argument: the data (columns 4 and 5)
# Second argument: 2 means apply the function to each column (1 would mean rows)
# Third argument: the function to apply (sum)
# Result: one sum value per column
print(apply(fragility23[,4:5],2,sum))
```

```
##                   Total S1: Demographic Pressures
##                  11784.0                   1066.1
```

```
# Check the data type of the apply() result
# typeof() returns the internal storage mode
# If it's not "list", then it's a vector
typeof(apply(fragility23[,4:5],2,sum))
```

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

If you do not see **list**, then it is a vector.

```
# Use apply() to sum each row separately
# The second argument is 1, meaning apply the function across rows
# This adds columns 4 and 5 together for each country
# Result: one sum value per row (per country)
print(apply(fragility23[,4:5],1,sum))
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 121.9 118.5 118.2 116.9 114.5 115.8 115.0 115.0 114.1 111.7 110.2 107.2 108.3
##    14    15    16    17    18    19    20    21    22    23    24    25    26
## 107.3 107.6 105.6 102.1 103.2 102.2 102.9 102.3 103.6 103.0 102.3  97.6 100.7
##    27    28    29    30    31    32    33    34    35    36    37    38    39
##  99.6  99.2  97.0  97.5  98.8  98.2  97.3  96.2  95.9  95.7  95.6  94.3  96.2
##    40    41    42    43    44    45    46    47    48    49    50    51    52
##  92.2  92.9  93.5  92.8  89.5  90.4  89.3  89.5  91.2  91.3  88.3  89.8  85.7
##    53    54    55    56    57    58    59    60    61    62    63    64    65
##  85.5  86.7  88.3  86.7  86.5  87.8  85.9  85.5  85.7  82.7  86.2  84.4  85.2
##    66    67    68    69    70    71    72    73    74    75    76    77    78
##  84.9  84.3  82.0  81.3  81.1  83.2  81.2  82.6  81.3  80.7  76.5  78.4  80.0
##    79    80    81    82    83    84    85    86    87    88    89    90    91
##  76.6  78.9  77.6  74.8  75.0  74.7  76.7  76.6  76.1  75.1  76.1  72.6  74.1
##    92    93    94    95    96    97    98    99   100   101   102   103   104
##  72.0  71.3  72.8  71.1  71.8  70.7  72.6  72.2  69.6  69.8  69.2  71.6  70.1
##   105   106   107   108   109   110   111   112   113   114   115   116   117
##  69.4  68.1  69.7  66.7  67.4  67.2  64.7  68.4  64.5  66.3  64.8  65.2  65.4
##   118   119   120   121   122   123   124   125   126   127   128   129   130
##  62.7  62.1  60.4  60.9  63.1  59.2  60.1  57.5  57.5  57.8  57.5  57.5  57.0
##   131   132   133   134   135   136   137   138   139   140   141   142   143
##  57.2  56.1  55.6  55.5  52.3  53.6  52.9  53.5  52.1  49.1  50.5  49.0  48.9
##   144   145   146   147   148   149   150   151   152   153   154   155   156
##  46.8  46.5  47.2  48.1  45.3  43.5  43.3  43.4  43.0  41.5  41.1  40.4  40.6
##   157   158   159   160   161   162   163   164   165   166   167   168   169
##  37.8  36.7  34.0  35.6  36.4  31.9  30.9  29.7  28.0  26.9  28.0  25.8  23.5
##   170   171   172   173   174   175   176   177   178   179
##  23.6  21.9  22.3  20.1  20.2  20.2  17.8  17.7  17.2  15.9
```

Apply by iterating:

```
# Use lapply() to apply sum() to each column
# lapply() iterates over the columns and applies the function to each
# The 'l' in lapply stands for 'list' - it always returns a list
print(lapply(fragility23[,4:5],sum))
```

```
## $Total
## [1] 11784
##
## $`S1: Demographic Pressures`
## [1] 1066.1
```

Notice output of **lapply**:

```
# Check the internal type of the lapply() result
# This confirms that lapply() returns a list internally
typeof(lapply(fragility23[,4:5],sum))
```

```
## [1] "list"
```

```
# Check the class of the lapply() result
# class() gives the object-oriented class, which is also "list"
class(lapply(fragility23[,4:5],sum))
```

```
## [1] "list"
```

Notice output of **sapply**:

```
# Use sapply() to apply sum() to each column
# sapply() is the "simplified" version of lapply()
# It tries to simplify the result to a vector or matrix if possible
print(sapply(fragility23[,4:5],sum))
```

```
##                    Total S1: Demographic Pressures
##                   11784.0                    1066.1
```

```
# Check the class of the sapply() result
# Because the results are simple numeric values, sapply() returns a named numeric vector
class(sapply(fragility23[,4:5],sum))
```

```
## [1] "numeric"
```

Similarly:

```
# Use lapply() to apply sqrt() to each column
# Returns a list where each element is a vector of square roots for that column
print(lapply(fragility23[,4:5],sqrt))
```

```
## $Total
##   [1] 10.578280 10.435516 10.416333 10.353743 10.348913 10.324728 10.305338
##   [8] 10.281051 10.227414 10.143964 10.019980 10.009995  9.974969  9.924717
##  [15]  9.899495  9.843780  9.803061  9.792855  9.721111  9.705668  9.695360
##  [22]  9.695360  9.695360  9.664368  9.581232  9.565563  9.560335  9.523655
##  [29]  9.513149  9.502631  9.481561  9.481561  9.428680  9.375500  9.370165
##  [36]  9.332738  9.327379  9.327379  9.322017  9.241212  9.230385  9.186947
##  [43]  9.121403  9.071935  9.066422  9.066422  9.060905  9.044335  9.038805
##  [50]  9.033272  9.022195  9.011104  8.983318  8.961027  8.955445  8.921883
##  [57]  8.921883  8.893818  8.837420  8.837420  8.820431  8.814760  8.803408
##  [64]  8.792042  8.752143  8.734987  8.723531  8.700575  8.694826  8.642916
##  [71]  8.631338  8.613942  8.608136  8.561542  8.549854  8.526429  8.502941
##  [78]  8.485281  8.479387  8.455767  8.408329  8.378544  8.366600  8.360622
##  [85]  8.354639  8.348653  8.330666  8.324662  8.324662  8.258329  8.246211
##  [92]  8.240146  8.215838  8.209750  8.173127  8.148620  8.148620  8.099383
##  [99]  8.093207  8.080842  8.068457  8.068457  8.068457  8.031189  7.981228
## [106]  7.930952  7.893035  7.867655  7.848567  7.797435  7.784600  7.765307
## [113]  7.765307  7.752419  7.733046  7.726578  7.713624  7.635444  7.615773
## [120]  7.549834  7.536577  7.436397  7.422937  7.416198  7.395945  7.334848
## [127]  7.328028  7.300685  7.280110  7.273239  7.197222  7.162402  7.155418
## [134]  7.014271  6.985700  6.978539  6.978539  6.978539  6.811755  6.737952
## [141]  6.730527  6.723095  6.640783  6.595453  6.580274  6.526868  6.496153
## [148]  6.473021  6.363961  6.356099  6.340347  6.276942  6.212890  6.164414
## [155]  6.148170  6.082763  5.865151  5.744563  5.612486  5.603570  5.522681
## [162]  5.366563  5.224940  5.069517  5.049752  4.959839  4.939636  4.690416
## [169]  4.582576  4.538722  4.415880  4.415880  4.347413  4.230839  4.219005
## [176]  4.086563  4.000000  3.962323  3.807887
##
## $`S1: Demographic Pressures`
##   [1] 3.162278 3.098387 3.114482 3.114482 2.720294 3.033150 2.966479 3.049590
##   [9] 3.082207 2.966479 3.130495 2.645751 2.966479 2.966479 3.098387 2.949576
##  [17] 2.449490 2.701851 2.774887 2.949576 2.880972 3.098387 3.000000 2.983287
##  [25] 2.408319 3.033150 2.863564 2.915476 2.549510 2.683282 2.983287 2.880972
##  [33] 2.898275 2.880972 2.846050 2.932576 2.932576 2.701851 3.049590 2.607681
##  [41] 2.774887 3.016621 3.098387 2.683282 2.863564 2.664583 2.720294 3.065942
##  [49] 3.098387 2.588436 2.898275 2.121320 2.190890 2.529822 2.846050 2.664583
##  [57] 2.626785 2.949576 2.792848 2.720294 2.810694 2.236068 2.949576 2.664583
##  [65] 2.932576 2.932576 2.863564 2.509980 2.387467 2.529822 2.949576 2.645751
##  [73] 2.915476 2.828427 2.756810 1.949359 2.469818 2.828427 2.167948 2.720294
##  [81] 2.626785 2.144761 2.236068 2.190890 2.626785 2.626785 2.588436 2.408319
##  [89] 2.607681 2.097618 2.469818 2.024846 1.949359 2.323790 2.073644 2.323790
##  [97] 2.073644 2.645751 2.588436 2.073644 2.167948 2.024846 2.549510 2.366432
## [105] 2.387467 2.280351 2.720294 2.190890 2.408319 2.529822 2.024846 2.846050
## [113] 2.049390 2.489980 2.236068 2.345208 2.428992 2.097618 2.024846 1.843909
## [121] 2.024846 2.792848 2.024846 2.258318 1.673320 1.923538 2.024846 2.049390
## [129] 2.121320 2.024846 2.323790 2.190890 2.097618 2.509980 1.870829 2.213594
## [137] 2.049390 2.190890 2.387467 1.923538 2.280351 1.949359 2.190890 1.816590
## [145] 1.788854 2.144761 2.428992 1.843909 1.732051 1.702939 1.788854 1.897367
## [153] 1.702939 1.760682 1.612452 1.897367 1.843909 1.923538 1.581139 2.049390
## [161] 2.428992 1.760682 1.897367 2.000000 1.581139 1.516575 1.897367 1.949359
```

```
## [169] 1.581139 1.732051 1.549193 1.673320 1.095445 1.516575 1.549193 1.048809
## [177] 1.303840 1.224745 1.183216
```

```
# Confirm that lapply() with sqrt() also returns a list
class(lapply(fragility23[,4:5],sqrt))
```

```
## [1] "list"
```

```
# Use sapply() to apply sqrt() to each column
# sapply() simplifies the result into a matrix (rows = observations, columns = variable
s)
print(sapply(fragility23[,4:5],sqrt))
```

```
##            Total S1: Demographic Pressures
##   [1,] 10.578280                  3.162278
##   [2,] 10.435516                  3.098387
##   [3,] 10.416333                  3.114482
##   [4,] 10.353743                  3.114482
##   [5,] 10.348913                  2.720294
##   [6,] 10.324728                  3.033150
##   [7,] 10.305338                  2.966479
##   [8,] 10.281051                  3.049590
##   [9,] 10.227414                  3.082207
##  [10,] 10.143964                  2.966479
##  [11,] 10.019980                  3.130495
##  [12,] 10.009995                  2.645751
##  [13,]  9.974969                  2.966479
##  [14,]  9.924717                  2.966479
##  [15,]  9.899495                  3.098387
##  [16,]  9.843780                  2.949576
##  [17,]  9.803061                  2.449490
##  [18,]  9.792855                  2.701851
##  [19,]  9.721111                  2.774887
##  [20,]  9.705668                  2.949576
##  [21,]  9.695360                  2.880972
##  [22,]  9.695360                  3.098387
##  [23,]  9.695360                  3.000000
##  [24,]  9.664368                  2.983287
##  [25,]  9.581232                  2.408319
##  [26,]  9.565563                  3.033150
##  [27,]  9.560335                  2.863564
##  [28,]  9.523655                  2.915476
##  [29,]  9.513149                  2.549510
##  [30,]  9.502631                  2.683282
##  [31,]  9.481561                  2.983287
##  [32,]  9.481561                  2.880972
##  [33,]  9.428680                  2.898275
##  [34,]  9.375500                  2.880972
##  [35,]  9.370165                  2.846050
##  [36,]  9.332738                  2.932576
##  [37,]  9.327379                  2.932576
##  [38,]  9.327379                  2.701851
##  [39,]  9.322017                  3.049590
##  [40,]  9.241212                  2.607681
##  [41,]  9.230385                  2.774887
##  [42,]  9.186947                  3.016621
##  [43,]  9.121403                  3.098387
##  [44,]  9.071935                  2.683282
##  [45,]  9.066422                  2.863564
##  [46,]  9.066422                  2.664583
##  [47,]  9.060905                  2.720294
##  [48,]  9.044335                  3.065942
##  [49,]  9.038805                  3.098387
##  [50,]  9.033272                  2.588436
##  [51,]  9.022195                  2.898275
```

```
##  [52,]  9.011104                  2.121320
##  [53,]  8.983318                  2.190890
##  [54,]  8.961027                  2.529822
##  [55,]  8.955445                  2.846050
##  [56,]  8.921883                  2.664583
##  [57,]  8.921883                  2.626785
##  [58,]  8.893818                  2.949576
##  [59,]  8.837420                  2.792848
##  [60,]  8.837420                  2.720294
##  [61,]  8.820431                  2.810694
##  [62,]  8.814760                  2.236068
##  [63,]  8.803408                  2.949576
##  [64,]  8.792042                  2.664583
##  [65,]  8.752143                  2.932576
##  [66,]  8.734987                  2.932576
##  [67,]  8.723531                  2.863564
##  [68,]  8.700575                  2.509980
##  [69,]  8.694826                  2.387467
##  [70,]  8.642916                  2.529822
##  [71,]  8.631338                  2.949576
##  [72,]  8.613942                  2.645751
##  [73,]  8.608136                  2.915476
##  [74,]  8.561542                  2.828427
##  [75,]  8.549854                  2.756810
##  [76,]  8.526429                  1.949359
##  [77,]  8.502941                  2.469818
##  [78,]  8.485281                  2.828427
##  [79,]  8.479387                  2.167948
##  [80,]  8.455767                  2.720294
##  [81,]  8.408329                  2.626785
##  [82,]  8.378544                  2.144761
##  [83,]  8.366600                  2.236068
##  [84,]  8.360622                  2.190890
##  [85,]  8.354639                  2.626785
##  [86,]  8.348653                  2.626785
##  [87,]  8.330666                  2.588436
##  [88,]  8.324662                  2.408319
##  [89,]  8.324662                  2.607681
##  [90,]  8.258329                  2.097618
##  [91,]  8.246211                  2.469818
##  [92,]  8.240146                  2.024846
##  [93,]  8.215838                  1.949359
##  [94,]  8.209750                  2.323790
##  [95,]  8.173127                  2.073644
##  [96,]  8.148620                  2.323790
##  [97,]  8.148620                  2.073644
##  [98,]  8.099383                  2.645751
##  [99,]  8.093207                  2.588436
## [100,]  8.080842                  2.073644
## [101,]  8.068457                  2.167948
## [102,]  8.068457                  2.024846
## [103,]  8.068457                  2.549510
```

```
## [104,]  8.031189                2.366432
## [105,]  7.981228                2.387467
## [106,]  7.930952                2.280351
## [107,]  7.893035                2.720294
## [108,]  7.867655                2.190890
## [109,]  7.848567                2.408319
## [110,]  7.797435                2.529822
## [111,]  7.784600                2.024846
## [112,]  7.765307                2.846050
## [113,]  7.765307                2.049390
## [114,]  7.752419                2.489980
## [115,]  7.733046                2.236068
## [116,]  7.726578                2.345208
## [117,]  7.713624                2.428992
## [118,]  7.635444                2.097618
## [119,]  7.615773                2.024846
## [120,]  7.549834                1.843909
## [121,]  7.536577                2.024846
## [122,]  7.436397                2.792848
## [123,]  7.422937                2.024846
## [124,]  7.416198                2.258318
## [125,]  7.395945                1.673320
## [126,]  7.334848                1.923538
## [127,]  7.328028                2.024846
## [128,]  7.300685                2.049390
## [129,]  7.280110                2.121320
## [130,]  7.273239                2.024846
## [131,]  7.197222                2.323790
## [132,]  7.162402                2.190890
## [133,]  7.155418                2.097618
## [134,]  7.014271                2.509980
## [135,]  6.985700                1.870829
## [136,]  6.978539                2.213594
## [137,]  6.978539                2.049390
## [138,]  6.978539                2.190890
## [139,]  6.811755                2.387467
## [140,]  6.737952                1.923538
## [141,]  6.730527                2.280351
## [142,]  6.723095                1.949359
## [143,]  6.640783                2.190890
## [144,]  6.595453                1.816590
## [145,]  6.580274                1.788854
## [146,]  6.526868                2.144761
## [147,]  6.496153                2.428992
## [148,]  6.473021                1.843909
## [149,]  6.363961                1.732051
## [150,]  6.356099                1.702939
## [151,]  6.340347                1.788854
## [152,]  6.276942                1.897367
## [153,]  6.212890                1.702939
## [154,]  6.164414                1.760682
## [155,]  6.148170                1.612452
```

```
## [156,]  6.082763                  1.897367
## [157,]  5.865151                  1.843909
## [158,]  5.744563                  1.923538
## [159,]  5.612486                  1.581139
## [160,]  5.603570                  2.049390
## [161,]  5.522681                  2.428992
## [162,]  5.366563                  1.760682
## [163,]  5.224940                  1.897367
## [164,]  5.069517                  2.000000
## [165,]  5.049752                  1.581139
## [166,]  4.959839                  1.516575
## [167,]  4.939636                  1.897367
## [168,]  4.690416                  1.949359
## [169,]  4.582576                  1.581139
## [170,]  4.538722                  1.732051
## [171,]  4.415880                  1.549193
## [172,]  4.415880                  1.673320
## [173,]  4.347413                  1.095445
## [174,]  4.230839                  1.516575
## [175,]  4.219005                  1.549193
## [176,]  4.086563                  1.048809
## [177,]  4.000000                  1.303840
## [178,]  3.962323                  1.224745
## [179,]  3.807887                  1.183216
```

```
# Check the class of sapply() result with sqrt()
# Because each column produces multiple values, sapply() returns a matrix
class(sapply(fragility23[,4:5],sqrt))
```

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

Now our own function:

```
# Define a custom function called theOnesOK
# This function takes a data frame with country names and one numeric variable
theOnesOK = function(DF_country_and_variable) {
  # Extract the second column (the numeric variable) as a vector
  variable_values <- DF_country_and_variable[,2]

  # Calculate the mean of the variable, ignoring any NA (missing) values
  avg_value <- mean(variable_values, na.rm = TRUE)

  # Create a new vector that labels each value as "Above Average" or "Below/At Average"
  # ifelse() is a vectorized if-else: checks condition for each element
  is_above <- ifelse(variable_values > avg_value, "Above Average", "Below/At Average")

  # Add the labels as a new column called "Status" to the data frame
  DF_country_and_variable$Status <- is_above

  # Return the modified data frame with the new Status column
  return(DF_country_and_variable)
}
```

```
# Test theOnesOK function with Country and the S1 indicator columns
# The c() function creates a vector of column names to select
# The function will add a Status column showing if each country is above/below average
theOnesOK(fragility23[,c('Country','S1: Demographic Pressures')])
```

```
##                       Country S1: Demographic Pressures           Status
## 1                     Somalia                       10.0    Above Average
## 2                       Yemen                        9.6    Above Average
## 3                 South Sudan                        9.7    Above Average
## 4   Congo Democratic Republic                        9.7    Above Average
## 5                       Syria                        7.4    Above Average
## 6                 Afghanistan                        9.2    Above Average
## 7                       Sudan                        8.8    Above Average
## 8    Central African Republic                        9.3    Above Average
## 9                        Chad                        9.5    Above Average
## 10                      Haiti                        8.8    Above Average
## 11                   Ethiopia                        9.8    Above Average
## 12                    Myanmar                        7.0    Above Average
## 13                       Mali                        8.8    Above Average
## 14                     Guinea                        8.8    Above Average
## 15                    Nigeria                        9.6    Above Average
## 16                   Zimbabwe                        8.7    Above Average
## 17                      Libya                        6.0    Above Average
## 18                    Ukraine                        7.3    Above Average
## 19                    Eritrea                        7.7    Above Average
## 20                    Burundi                        8.7    Above Average
## 21                Burkina Faso                       8.3    Above Average
## 22                  Mozambique                       9.6    Above Average
## 23                    Cameroon                       9.0    Above Average
## 24                       Niger                       8.9    Above Average
## 25                     Lebanon                       5.8 Below/At Average
## 26                      Uganda                       9.2    Above Average
## 27                        Iraq                       8.2    Above Average
## 28              Congo Republic                       8.5    Above Average
## 29                   Venezuela                       6.5    Above Average
## 30                   Sri Lanka                       7.2    Above Average
## 31              Guinea Bissau                        8.9    Above Average
## 32                    Pakistan                       8.3    Above Average
## 33                     Liberia                       8.4    Above Average
## 34                   Palestine                       8.3    Above Average
## 35                       Kenya                       8.1    Above Average
## 36              Cote d'Ivoire                        8.6    Above Average
## 37                 Mauritania                        8.6    Above Average
## 38                 North Korea                       7.3    Above Average
## 39                      Angola                       9.3    Above Average
## 40                        Iran                       6.8    Above Average
## 41                  Bangladesh                       7.7    Above Average
## 42           Equatorial Guinea                       9.1    Above Average
## 43                      Malawi                       9.6    Above Average
## 44                      Rwanda                       7.2    Above Average
## 45                     Comoros                       8.2    Above Average
## 46                    Djibouti                       7.1    Above Average
## 47                        Togo                       7.4    Above Average
## 48                      Zambia                       9.4    Above Average
## 49                  Madagascar                       9.6    Above Average
## 50                       Egypt                       6.7    Above Average
## 51                Sierra Leone                       8.4    Above Average
```

```
## 52                   Turkey                  4.5 Below/At Average
## 53                   Russia                  4.8 Below/At Average
## 54                 Cambodia                  6.4    Above Average
## 55                    Nepal                  8.1    Above Average
## 56          Solomon Islands                  7.1    Above Average
## 57                 Honduras                  6.9    Above Average
## 58               Swaziland                   8.7    Above Average
## 59                 Colombia                  7.8    Above Average
## 60         Papua New Guinea                  7.4    Above Average
## 61              Philippines                  7.9    Above Average
## 62                Nicaragua                  5.0 Below/At Average
## 63              Timor-Leste                  8.7    Above Average
## 64                Guatemala                  7.1    Above Average
## 65                 Tanzania                  8.6    Above Average
## 66                  Lesotho                  8.6    Above Average
## 67                   Gambia                  8.2    Above Average
## 68                   Jordan                  6.3    Above Average
## 69           Kyrgyz Republic                 5.7 Below/At Average
## 70                     Laos                  6.4    Above Average
## 71                   Brazil                  8.7    Above Average
## 72                Tajikistan                 7.0    Above Average
## 73                    India                  8.5    Above Average
## 74                    Benin                  8.0    Above Average
## 75                     Peru                  7.6    Above Average
## 76                Azerbaijan                 3.8 Below/At Average
## 77    Bosnia and Herzegovina                 6.1    Above Average
## 78              South Africa                 8.0    Above Average
## 79                  Georgia                  4.7 Below/At Average
## 80                  Senegal                  7.4    Above Average
## 81                  Bolivia                  6.9    Above Average
## 82                     Fiji                  4.6 Below/At Average
## 83                  Algeria                  5.0 Below/At Average
## 84                  Belarus                  4.8 Below/At Average
## 85                   Mexico                  6.9    Above Average
## 86     Sao Tome and Principe                 6.9    Above Average
## 87                  Ecuador                  6.7    Above Average
## 88                Micronesia                 5.8 Below/At Average
## 89               El Salvador                 6.8    Above Average
## 90                  Morocco                  4.4 Below/At Average
## 91                 Thailand                  6.1    Above Average
## 92                   Serbia                  4.1 Below/At Average
## 93                  Armenia                  3.8 Below/At Average
## 94                  Moldova                  5.4 Below/At Average
## 95                Uzbekistan                 4.3 Below/At Average
## 96                   Bhutan                  5.4 Below/At Average
## 97                  Tunisia                  4.3 Below/At Average
## 98                Indonesia                  7.0    Above Average
## 99                    Gabon                  6.7    Above Average
## 100             Saudi Arabia                 4.3 Below/At Average
## 101                   Samoa                  4.7 Below/At Average
## 102                 Bahrain                  4.1 Below/At Average
## 103                   China                  6.5    Above Average
```

```
## 104              Turkmenistan                  5.6 Below/At Average
## 105                  Paraguay                  5.7 Below/At Average
## 106                  Maldives                  5.2 Below/At Average
## 107                     Ghana                  7.4    Above Average
## 108                   Jamaica                  4.8 Below/At Average
## 109                    Guyana                  5.8 Below/At Average
## 110        Dominican Republic                  6.4    Above Average
## 111                Kazakhstan                  4.1 Below/At Average
## 112                   Namibia                  8.1    Above Average
## 113                  Macedonia                  4.2 Below/At Average
## 114                Cape Verde                  6.2    Above Average
## 115                    Belize                  5.0 Below/At Average
## 116                  Suriname                  5.5 Below/At Average
## 117                      Cuba                  5.9 Below/At Average
## 118                   Vietnam                  4.4 Below/At Average
## 119                Montenegro                  4.1 Below/At Average
## 120                    Cyprus                  3.4 Below/At Average
## 121                   Albania                  4.1 Below/At Average
## 122                  Botswana                  7.8    Above Average
## 123                    Greece                  4.1 Below/At Average
## 124                  Malaysia                  5.1 Below/At Average
## 125          Brunei Darussalam                  2.8 Below/At Average
## 126       Antigua and Barbuda                  3.7 Below/At Average
## 127                   Grenada                  4.1 Below/At Average
## 128                Seychelles                  4.2 Below/At Average
## 129                   Romania                  4.5 Below/At Average
## 130       Trinidad and Tobago                  4.1 Below/At Average
## 131                  Bulgaria                  5.4 Below/At Average
## 132                  Mongolia                  4.8 Below/At Average
## 133                    Kuwait                  4.4 Below/At Average
## 134                   Bahamas                  6.3    Above Average
## 135                   Hungary                  3.5 Below/At Average
## 136                    Panama                  4.9 Below/At Average
## 137                      Oman                  4.2 Below/At Average
## 138                   Croatia                  4.8 Below/At Average
## 139                 Argentina                  5.7 Below/At Average
## 140                  Barbados                  3.7 Below/At Average
## 141             United States                  5.2 Below/At Average
## 142                    Poland                  3.8 Below/At Average
## 143                    Israel                  4.8 Below/At Average
## 144                     Spain                  3.3 Below/At Average
## 145                    Latvia                  3.2 Below/At Average
## 146                     Italy                  4.6 Below/At Average
## 147                     Chile                  5.9 Below/At Average
## 148            United Kingdom                  3.4 Below/At Average
## 149                     Qatar                  3.0 Below/At Average
## 150                Costa Rica                  2.9 Below/At Average
## 151            Czech Republic                  3.2 Below/At Average
## 152                 Lithuania                  3.6 Below/At Average
## 153                   Estonia                  2.9 Below/At Average
## 154                 Mauritius                  3.1 Below/At Average
## 155            Slovak Republic                  2.6 Below/At Average
```

```
## 156      United Arab Emirates                    3.6 Below/At Average
## 157                   Uruguay                    3.4 Below/At Average
## 158                     Malta                    3.7 Below/At Average
## 159               South Korea                    2.5 Below/At Average
## 160                   Belgium                    4.2 Below/At Average
## 161                     Japan                    5.9 Below/At Average
## 162                    France                    3.1 Below/At Average
## 163                  Slovenia                    3.6 Below/At Average
## 164                  Portugal                    4.0 Below/At Average
## 165                 Singapore                    2.5 Below/At Average
## 166                   Germany                    2.3 Below/At Average
## 167                   Austria                    3.6 Below/At Average
## 168                 Australia                    3.8 Below/At Average
## 169               Netherlands                    2.5 Below/At Average
## 170                    Sweden                    3.0 Below/At Average
## 171                Luxembourg                    2.4 Below/At Average
## 172                   Ireland                    2.8 Below/At Average
## 173                    Canada                    1.2 Below/At Average
## 174                   Denmark                    2.3 Below/At Average
## 175               Switzerland                    2.4 Below/At Average
## 176               New Zealand                    1.1 Below/At Average
## 177                   Finland                    1.7 Below/At Average
## 178                   Iceland                    1.5 Below/At Average
## 179                    Norway                    1.4 Below/At Average
```

```r
# Define the mystery function (explained in detail at the end of this notebook)
# This function calculates the row-wise mean of selected columns
mystery=function(DF,positionsToUse,CountryColumn='Country'){
  # Create a new data frame with only the Country column
  # drop = FALSE ensures the result stays as a data frame (not a vector)
  newDF=DF[,c(CountryColumn),drop = FALSE]

  # Create a variable to store the name of the new column
  average='average'

  # Calculate the row-wise mean for the selected columns and add as new column
  # apply() with 1 applies the function across rows
  # mean with na.rm = TRUE ignores missing values
  newDF[,average]=apply(DF[,positionsToUse],1,mean,na.rm = TRUE)

  # Return only the Country column and the new average column
  return(newDF[,c(CountryColumn,average)])
}
```

```r
# Test the mystery function with columns 4 through 6
# This calculates the average of those three columns for each country
mystery(fragility23,4:6)
```

```
##                       Country   average
## 1                     Somalia 43.633333
## 2                       Yemen 42.700000
## 3                 South Sudan 42.733333
## 4   Congo Democratic Republic 42.233333
## 5                       Syria 41.200000
## 6                 Afghanistan 41.466667
## 7                       Sudan 41.533333
## 8    Central African Republic 41.500000
## 9                        Chad 41.033333
## 10                     Haiti 39.800000
## 11                  Ethiopia 39.900000
## 12                   Myanmar 38.800000
## 13                      Mali 38.933333
## 14                    Guinea 37.833333
## 15                   Nigeria 38.000000
## 16                  Zimbabwe 37.733333
## 17                     Libya 36.533333
## 18                   Ukraine 37.733333
## 19                   Eritrea 36.266667
## 20                   Burundi 36.966667
## 21              Burkina Faso 36.833333
## 22                Mozambique 37.166667
## 23                  Cameroon 37.133333
## 24                     Niger 36.733333
## 25                   Lebanon 35.466667
## 26                    Uganda 36.533333
## 27                      Iraq 35.766667
## 28             Congo Republic 35.233333
## 29                  Venezuela 34.500000
## 30                  Sri Lanka 34.700000
## 31              Guinea Bissau 34.766667
## 32                   Pakistan 35.266667
## 33                    Liberia 34.900000
## 34                  Palestine 34.133333
## 35                      Kenya 34.400000
## 36              Cote d'Ivoire 33.900000
## 37                 Mauritania 34.633333
## 38                North Korea 32.500000
## 39                     Angola 33.833333
## 40                       Iran 32.566667
## 41                 Bangladesh 33.466667
## 42          Equatorial Guinea 33.000000
## 43                     Malawi 32.933333
## 44                     Rwanda 32.266667
## 45                    Comoros 31.966667
## 46                   Djibouti 32.133333
## 47                       Togo 31.733333
## 48                     Zambia 32.033333
## 49                 Madagascar 32.100000
## 50                      Egypt 31.266667
## 51               Sierra Leone 32.400000
```

```
## 52                   Turkey 31.300000
## 53                   Russia 30.100000
## 54                 Cambodia 30.200000
## 55                    Nepal 31.633333
## 56           Solomon Islands 30.566667
## 57                 Honduras 30.766667
## 58                Swaziland 30.500000
## 59                 Colombia 31.233333
## 60          Papua New Guinea 30.133333
## 61               Philippines 30.500000
## 62                 Nicaragua 29.266667
## 63               Timor-Leste 30.633333
## 64                 Guatemala 29.900000
## 65                  Tanzania 30.033333
## 66                   Lesotho 29.600000
## 67                    Gambia 30.033333
## 68                    Jordan 30.133333
## 69           Kyrgyz Republic 28.400000
## 70                      Laos 28.833333
## 71                    Brazil 28.900000
## 72                Tajikistan 28.333333
## 73                     India 28.900000
## 74                     Benin 29.066667
## 75                      Peru 28.400000
## 76                Azerbaijan 27.600000
## 77    Bosnia and Herzegovina 28.533333
## 78              South Africa 28.000000
## 79                   Georgia 27.700000
## 80                   Senegal 28.233333
## 81                   Bolivia 26.733333
## 82                      Fiji 26.000000
## 83                   Algeria 27.066667
## 84                   Belarus 25.966667
## 85                    Mexico 27.400000
## 86     Sao Tome and Principe 27.266667
## 87                   Ecuador 27.033333
## 88                Micronesia 26.433333
## 89               El Salvador 27.133333
## 90                   Morocco 25.900000
## 91                  Thailand 26.333333
## 92                    Serbia 26.266667
## 93                   Armenia 26.033333
## 94                   Moldova 26.266667
## 95                Uzbekistan 24.966667
## 96                    Bhutan 25.866667
## 97                   Tunisia 24.533333
## 98                 Indonesia 25.666667
## 99                     Gabon 24.933333
## 100             Saudi Arabia 24.266667
## 101                   Samoa 24.433333
## 102                 Bahrain 23.533333
## 103                   China 24.900000
```

```
## 104              Turkmenistan 24.133333
## 105                  Paraguay 24.100000
## 106                  Maldives 23.866667
## 107                     Ghana 24.400000
## 108                   Jamaica 22.966667
## 109                    Guyana 23.466667
## 110        Dominican Republic 23.266667
## 111                Kazakhstan 22.333333
## 112                   Namibia 24.000000
## 113                 Macedonia 23.600000
## 114                Cape Verde 23.300000
## 115                    Belize 22.600000
## 116                  Suriname 22.666667
## 117                      Cuba 22.866667
## 118                   Vietnam 22.200000
## 119                Montenegro 21.900000
## 120                    Cyprus 22.100000
## 121                   Albania 21.233333
## 122                  Botswana 22.066667
## 123                    Greece 21.500000
## 124                  Malaysia 21.100000
## 125         Brunei Darussalam 19.533333
## 126       Antigua and Barbuda 20.066667
## 127                   Grenada 20.000000
## 128                Seychelles 19.933333
## 129                   Romania 20.500000
## 130       Trinidad and Tobago 19.833333
## 131                  Bulgaria 20.533333
## 132                  Mongolia 19.466667
## 133                    Kuwait 19.200000
## 134                   Bahamas 19.333333
## 135                   Hungary 19.233333
## 136                    Panama 18.666667
## 137                      Oman 18.166667
## 138                   Croatia 20.100000
## 139                 Argentina 17.900000
## 140                  Barbados 17.000000
## 141             United States 17.566667
## 142                    Poland 18.300000
## 143                    Israel 16.866667
## 144                     Spain 16.233333
## 145                    Latvia 16.500000
## 146                     Italy 17.166667
## 147                     Chile 16.533333
## 148            United Kingdom 15.900000
## 149                     Qatar 14.933333
## 150                Costa Rica 16.066667
## 151            Czech Republic 15.966667
## 152                 Lithuania 15.300000
## 153                   Estonia 14.700000
## 154                 Mauritius 14.233333
## 155            Slovak Republic 14.566667
```

```
## 156      United Arab Emirates 14.000000
## 157                   Uruguay 13.200000
## 158                     Malta 13.300000
## 159               South Korea 11.700000
## 160                   Belgium 12.566667
## 161                     Japan 13.000000
## 162                    France 11.433333
## 163                  Slovenia 11.166667
## 164                  Portugal 10.366667
## 165                 Singapore  9.500000
## 166                   Germany 10.466667
## 167                   Austria 10.833333
## 168                 Australia  9.166667
## 169               Netherlands  8.633333
## 170                    Sweden  9.100000
## 171                Luxembourg  8.233333
## 172                   Ireland  7.966667
## 173                    Canada  7.366667
## 174                   Denmark  7.733333
## 175               Switzerland  7.800000
## 176               New Zealand  6.333333
## 177                   Finland  6.533333
## 178                   Iceland  6.233333
## 179                    Norway  5.866667
```

Part 2: Understanding Functions

Exercise 1: Explanation of theOnesOK2

The function `theOnesOK2` is an improved, more flexible version of `theOnesOK`:

```
# Define theOnesOK2 - a more flexible version of theOnesOK
# Parameters:
#   DF: the full data frame
#   DFvariable: the name of the numeric column to analyze (as a string)
#   CountryColumn: the name of the country column (defaults to 'Country')
theOnesOK2 = function(DF, DFvariable, CountryColumn='Country') {

  # Extract the values from the specified variable column
  # Uses bracket notation with column name string instead of position
  variable_values <- DF[,DFvariable]

  # Calculate the mean of the variable values, removing any NA values
  avg_value <- mean(variable_values, na.rm = TRUE)

  # Compare each value to the average and create status labels
  # Values strictly greater than average get "Above Average"
  # Values equal to or below average get "Below/At Average"
  is_above <- ifelse(variable_values > avg_value, "Above Average", "Below/At Average")

  # Create a dynamic column name by pasting "Status_on_" with the variable name
  # For example, if DFvariable is "Total", newname becomes "Status_on_Total"
  newname = paste('Status_on_', DFvariable)

  # Add the status labels as a new column with the dynamic name
  DF[,newname] <- is_above

  # Return a data frame with only the Country column and the new status column
  # This is cleaner output than returning the entire modified data frame
  return(DF[,c(CountryColumn, newname)])
}
```

How theOnesOK2 differs from theOnesOK:

1. **More flexible input**: theOnesOK requires a pre-selected 2-column data frame, while theOnesOK2 takes the full data frame and column names as separate arguments.
2. **Dynamic column naming**: theOnesOK2 creates a descriptive column name like “Status_on_Total” instead of just “Status”, making it clearer what variable was analyzed.
3. **Customizable country column**: theOnesOK2 has a parameter to specify which column contains country names, with a sensible default.
4. **Cleaner output**: theOnesOK2 returns only the relevant columns (Country and Status), while theOnesOK returns all original columns plus Status.

Exercise 2: Explanation of mystery function

The `mystery` function calculates the **average across multiple columns for each row** (i.e., for each country):

```r
# The mystery function explained line-by-line:
mystery = function(DF, positionsToUse, CountryColumn='Country'){

  # Create a new data frame containing only the country column
  # drop = FALSE is crucial: without it, selecting one column returns a vector
  # With drop = FALSE, we keep it as a data frame structure
  newDF = DF[,c(CountryColumn), drop = FALSE]

  # Store the string 'average' in a variable
  # This will be used as the new column name
  average = 'average'

  # This is the key line - it does several things:
  # 1. DF[,positionsToUse] selects the columns at the specified positions
  # 2. apply(..., 1, ...) applies a function to each ROW (1 = rows, 2 = columns)
  # 3. mean with na.rm = TRUE calculates the mean ignoring missing values
  # 4. The result is assigned as a new column named 'average' in newDF
  newDF[,average] = apply(DF[,positionsToUse], 1, mean, na.rm = TRUE)

  # Return a data frame with just the country name and its average score
  return(newDF[,c(CountryColumn, average)])
}
```

What the function does:

The `mystery` function takes a data frame and calculates the **row-wise mean** of specified columns. For the Fragile States Index data, this is useful for calculating a country's average score across multiple indicators.

Example usage:

When we call `mystery(fragility23, 4:6)` , it: 1. Takes columns 4, 5, and 6 (three FSI indicators) 2. For each country (row), calculates the average of those three values 3. Returns a clean two-column data frame: Country name and their average

This is essentially creating a composite score from multiple indicators for each country.