Tutorial

Shail Mirpuri (704904548)

10/23/2021

Tutorial: Basic Exploratory Data Analysis In R

Recap from Last Time

Previously we covered basic data-types, data structures and how they work on a high level in R.

We also introduced how to find data on websites such as Kaggle.

Today we will be going one step further and actually exploring datasets to answer some interesting questions!

Overview

1. What is Vectorization in R?
2. The Apply Family of Functions
3. The Tidyverse Series of Packages
4. Using Dplyr to explore data

1) What is Vectorization in R?

In R, most functions are vectorized, which means that when you apply them to a vector the function will operate on all elements within the vectors.

This means you do not need to use a loop at apply the function to each element within the vector. In fact using vectorization is a lot quicker than if you did the same operation using a loop.

```
# example of a loop and vectorization and the time difference

g <- rnorm(100000)
g_plus_4 <- numeric(100000)

# Start the clock!
ptm <- proc.time()

# Loop through the vector, adding one
for (i in 1:100000){
    g_plus_4[i] <- g[i] + 4
}

# Stop the clock
proc.time() - ptm
```

```
##    user  system elapsed
##   0.039   0.003   0.041
```

```r
# Vectorized method is much quicker
ptm <- proc.time()
g_plus_4 <- g + 4
proc.time() - ptm
```

```
##    user  system elapsed
##   0.002   0.000   0.003
```

In general, most things can be done using vectorization and you should only use loops if:

1) The order of your operations matter (i.e. the third element depends on the second element which depends on the first etc.)
2) It is much easier to do it using a loop rather than trying to figure out the vectorized approach.

Some functions in R are not directly vectorized but we still may want to apply them to several elements separately. This is where the apply family of functions comes into play.

For example the `isTrue` function.

```r
isTRUE(c(TRUE,FALSE,FALSE)) # returns only one value
```

```
## [1] FALSE
```

```r
# using apply (vapply) we can apply this function to all elements

vapply(X = c(TRUE,FALSE,FALSE), FUN = isTRUE, FUN.VALUE = logical(1))
```

```
## [1]  TRUE FALSE FALSE
```

2) The Apply Family of Functions

What is an Apply function?

An apply function takes a regular function and applies to all elements within a data structure.

There are five different apply functions: vapply, apply, tapply, lapply, sapply

We will demonstrate how apply and tapply can be used to explore data easily

The Apply Function

This function will allow to perform any computation by multiple rows/columns a lot quicker so it can extremely useful in multi-dimensional data structures

```r
# recall the matrix A
A <- matrix(1:6, nrow = 2, ncol = 3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
# let's say we want to find the sum of each column
# there are two ways we can technically do this

# the first is to manually subset each column and call the sum() function on the resultant vector

# e.g.
first_col_A <- A[,1] # using subseting to get the first column of A
sum(first_col_A)
```

```
## [1] 3
```

```r
second_col_A <- A[,2]
sum(second_col_A)
```

```
## [1] 7
```

```r
third_col_A <- A[,3]
sum(third_col_A)
```

```
## [1] 11
```

```r
# This obviously very ineffecient and imagine how long this would take us we had 10+ columns of data

# Luckily the apply function can allow us to 'apply' the sum function to all the columns in a single li
apply(A, MARGIN = 2, sum)
```

```
## [1]  3  7 11
```

The apply function requires 3 arguments `apply(X, MARGIN, FUNC)`.

`X` is the multi-dimensional structure you want to apply the function to

`MARGIN` is the level or axis you want to apply the function on (1 for ROW, 2 for COLUMN)

`FUNC` is the function you want to apply

For example if we want to find the row-wise mean of A we do the following:

```r
apply(A, MARGIN = 1, mean) # we will get a vector where the first element is the mean of the first row
```

```
## [1] 3 4
```

Using a Tapply function for EDA

Apart from performing quick computations and explorations using the apply function, the `tapply()` function is one my favorites when it comes to discovering trends/patterns/insights from a data set

Let's see how this function works!

```r
library(datasets) # easy way of loading datasets
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
# let's say we want to find the average sepal width for each species
# using the tapply function we can easily do this
tapply(iris$Sepal.Width, iris$Species, mean) # we can see that the setosa species tend to have a longer
```

```
##     setosa versicolor  virginica
##      3.428      2.770      2.974
```

Using the t-apply function, we can see that the Setosa species tend to have a longer sepal width than Versicolor and Virginca.

So now that you have seen the functionality of the t-apply function let's go over how you can use/call it.

The general format is: `tapply(X, index, function)`

`X` refers to the vector you will be applying the function to

`index` refers to what you will splitting the data by before applying the function (the index can be multiple vectors in a list as well)

`function` refers to the function you will applying after splitting the data.

Essentially, t-apply applies a specific function separately to different groups, which makes it very useful in finding underlying trends between groups.

The t-apply function can take some time to get your head around but the best way to learn and understand how it works is to use it in practice to explore and find insights from your data.

In this tutorial we will be using the olympics dataset from Kaggle to demonstrate how we can use the apply and tapply functions that we have learned to gain insight about this data.

Loading CSV data into R:

```r
df <- read.csv('olympics.csv') # make sure that the csv file is in your working directory if you want r
head(df) # this functions shows the first 5 rows of the data
```

```
##       City Year    Sport Discipline          Event                  Athlete
## 1 Montreal 1976 Aquatics     Diving 3m springboard       K\xd6HLER, Christa
## 2 Montreal 1976 Aquatics     Diving 3m springboard      KOSENKOV, Aleksandr
## 3 Montreal 1976 Aquatics     Diving 3m springboard     BOGGS, Philip George
## 4 Montreal 1976 Aquatics     Diving 3m springboard CAGNOTTO, Giorgio Franco
## 5 Montreal 1976 Aquatics     Diving   10m platform   WILSON, Deborah Keplar
## 6 Montreal 1976 Aquatics     Diving   10m platform        LOUGANIS, Gregory
##   Gender Country_Code       Country Event_gender  Medal
## 1  Women          GDR  East Germany            W Silver
## 2    Men          URS  Soviet Union            M Bronze
## 3    Men          USA United States            M   Gold
## 4    Men          ITA         Italy            M Silver
## 5  Women          USA United States            W Bronze
## 6    Men          USA United States            M Silver
```

```r
# counting the number of bronze, silver and gold medals in our dataset

table(df$Medal) # the table functions takes in a vector and summarise the value counts for the vector
```

```
##
##        Bronze   Gold Silver
##    117   5258   5042   5016
```

```r
# Using the tapply function to find the amount of medals won by each country

tapply(df$Medal, df$Country, table)
```

```
## [[1]]
##
##
## 117
##
## $Afghanistan
##
## Bronze
##      1
##
## $Algeria
##
## Bronze   Gold Silver
##      8      4      2
##
## $Argentina
##
## Bronze   Gold Silver
##     70     46     37
##
## $Armenia
##
## Bronze   Gold Silver
##      7      1      1
##
## $Australia
##
## Bronze   Gold Silver
##    312    216    270
##
## $Austria
##
## Bronze   Gold Silver
##      8      9     17
##
## $Azerbaijan
##
## Bronze   Gold Silver
##      9      4      3
##
## $Bahamas
```

```
##
## Bronze   Gold Silver
##      3      7      9
##
## $Barbados
##
## Bronze
##      1
##
## $Belarus
##
## Bronze   Gold Silver
##     53     14     25
##
## $Belgium
##
## Bronze   Gold Silver
##     20      6     15
##
## $`Bermuda*`
##
## Bronze
##      1
##
## $Brazil
##
## Bronze   Gold Silver
##    125     56    137
##
## $Bulgaria
##
## Bronze   Gold Silver
##    107     40    100
##
## $Burundi
##
## Gold
##    1
##
## $Cameroon
##
## Bronze   Gold
##      1     20
##
## $Canada
##
## Bronze   Gold Silver
##    117     76    111
##
## $Chile
##
## Bronze   Gold Silver
##     19      3      2
##
```

```
## $China
##
## Bronze   Gold Silver
##    193    234    252
##
## $Colombia
##
## Bronze   Gold Silver
##      5      1      2
##
## $`Costa Rica`
##
## Bronze   Gold Silver
##      2      1      1
##
## $`Cote d'Ivoire`
##
## Silver
##      1
##
## $Croatia
##
## Bronze   Gold Silver
##     18     31     30
##
## $Cuba
##
## Bronze   Gold Silver
##     88    152    109
##
## $`Czech Republic`
##
## Bronze   Gold Silver
##     13     10     18
##
## $Czechoslovakia
##
## Bronze   Gold Silver
##     29     27     29
##
## $Denmark
##
## Bronze   Gold Silver
##     50     78     20
##
## $Djibouti
##
## Bronze
##      1
##
## $`Dominican Republic`
##
## Bronze   Gold Silver
##      1      2      1
```

```
##
## $`East Germany`
##
## Bronze   Gold Silver
##    150    286    190
##
## $Ecuador
##
##   Gold Silver
##      1      1
##
## $Egypt
##
## Bronze   Gold Silver
##      4      1      2
##
## $Eritrea
##
## Bronze
##      1
##
## $Estonia
##
## Bronze   Gold Silver
##      6      3      3
##
## $Ethiopia
##
## Bronze   Gold Silver
##     12     15      5
##
## $Finland
##
## Bronze   Gold Silver
##     19     18     17
##
## $France
##
## Bronze   Gold Silver
##    185    154    110
##
## $Georgia
##
## Bronze   Gold Silver
##     11      5      2
##
## $Germany
##
## Bronze   Gold Silver
##    278    237    176
##
## $Ghana
##
## Bronze
```

```
##     13
##
## $Greece
##
## Bronze   Gold Silver
##     23     19     32
##
## $Guyana
##
## Bronze
##      1
##
## $`Hong Kong*`
##
##   Gold Silver
##      1      2
##
## $Hungary
##
## Bronze   Gold Silver
##    135    129    104
##
## $Iceland
##
## Bronze Silver
##      2     14
##
## $`Independent Olympic Participants (1992)`
##
## Bronze Silver
##      2      1
##
## $India
##
## Bronze   Gold Silver
##      4     17      1
##
## $Indonesia
##
## Bronze   Gold Silver
##     12      9     14
##
## $Iran
##
## Bronze   Gold Silver
##      8      7      6
##
## $Ireland
##
## Bronze   Gold Silver
##      4      4      6
##
## $Israel
##
```

```
## Bronze   Gold Silver
##      5      1      1
##
## $Italy
##
## Bronze   Gold Silver
##    178    145    163
##
## $Jamaica
##
## Bronze   Gold Silver
##     38     17     34
##
## $Japan
##
## Bronze   Gold Silver
##    182     94    112
##
## $Kazakhstan
##
## Bronze   Gold Silver
##     14      9     16
##
## $Kenya
##
## Bronze   Gold Silver
##     17     18     21
##
## $`Korea, North`
##
## Bronze   Gold Silver
##     17      9     11
##
## $`Korea, South`
##
## Bronze   Gold Silver
##    128    140    186
##
## $Kuwait
##
## Bronze
##      1
##
## $Kyrgyzstan
##
## Bronze Silver
##      2      1
##
## $Latvia
##
## Bronze   Gold Silver
##      3      2      9
##
## $Lebanon
```

```
##
## Bronze
##      1
##
## $Lithuania
##
## Bronze   Gold Silver
##     42      4      4
##
## $Macedonia
##
## Bronze
##      1
##
## $Malaysia
##
## Bronze Silver
##      3      3
##
## $Mauritius
##
## Bronze
##      1
##
## $Mexico
##
## Bronze   Gold Silver
##     22      6     10
##
## $Moldova
##
## Bronze Silver
##      3      3
##
## $Mongolia
##
## Bronze   Gold Silver
##      7      2      5
##
## $Morocco
##
## Bronze   Gold Silver
##     10      6      4
##
## $Mozambique
##
## Bronze   Gold
##      1      1
##
## $Namibia
##
## Silver
##      4
##
```

```
## $Netherlands
##
## Bronze   Gold Silver
##    151    137    140
##
## $`Netherlands Antilles*`
##
## Silver
##      1
##
## $`New Zealand`
##
## Bronze   Gold Silver
##     51     50     21
##
## $Nigeria
##
## Bronze   Gold Silver
##     25     19     38
##
## $Norway
##
## Bronze   Gold Silver
##     46     50     58
##
## $Pakistan
##
## Bronze   Gold
##     33     16
##
## $Panama
##
## Gold
##    1
##
## $Paraguay
##
## Silver
##     17
##
## $Peru
##
## Silver
##     14
##
## $Philippines
##
## Bronze Silver
##      2      1
##
## $Poland
##
## Bronze   Gold Silver
##     98     52    113
```

```
##
## $Portugal
##
## Bronze   Gold Silver
##      7      4      5
##
## $`Puerto Rico*`
##
## Bronze Silver
##      4      1
##
## $Qatar
##
## Bronze
##      2
##
## $Romania
##
## Bronze   Gold Silver
##    190    135    157
##
## $Russia
##
## Bronze   Gold Silver
##    240    192    206
##
## $`Saudi Arabia`
##
## Bronze Silver
##      1      1
##
## $Senegal
##
## Silver
##      1
##
## $Serbia
##
## Bronze Silver
##     14     15
##
## $Singapore
##
## Silver
##      3
##
## $Slovakia
##
## Bronze   Gold Silver
##      8     10     11
##
## $Slovenia
##
## Bronze   Gold Silver
```

```
##     11      4      6
##
## $`South Africa`
##
## Bronze   Gold Silver
##      7      7     10
##
## $`Soviet Union`
##
## Bronze   Gold Silver
##    297    439    285
##
## $Spain
##
## Bronze   Gold Silver
##     76     87    165
##
## $`Sri Lanka`
##
## Silver
##      1
##
## $Sudan
##
## Silver
##      1
##
## $Suriname
##
## Bronze   Gold
##      1      1
##
## $Sweden
##
## Bronze   Gold Silver
##     55     28    110
##
## $Switzerland
##
## Bronze   Gold Silver
##     28     14     37
##
## $Syria
##
## Bronze   Gold Silver
##      1      1      1
##
## $Taiwan
##
## Bronze   Gold Silver
##     12      2     26
##
## $Tajikistan
##
```

```
## Bronze Silver
##      1      1
##
## $Tanzania
##
## Silver
##      2
##
## $Thailand
##
## Bronze   Gold Silver
##     10      7      4
##
## $Togo
##
## Bronze
##      1
##
## $Tonga
##
## Silver
##      1
##
## $`Trinidad and Tobago`
##
## Bronze   Gold Silver
##      4      1      6
##
## $Tunisia
##
## Bronze   Gold
##      1      1
##
## $Turkey
##
## Bronze   Gold Silver
##     15     14     11
##
## $Uganda
##
## Bronze Silver
##      1      1
##
## $Ukraine
##
## Bronze   Gold Silver
##     78     32     38
##
## $`Unified team`
##
## Bronze   Gold Silver
##     66     92     65
##
## $`United Arab Emirates`
```

```
##
## Gold
##    1
##
## $`United Kingdom`
##
## Bronze   Gold Silver
##    188    122    157
##
## $`United States`
##
## Bronze   Gold Silver
##    481    928    583
##
## $Uruguay
##
## Silver
##      1
##
## $Uzbekistan
##
## Bronze   Gold Silver
##      8      4      5
##
## $Venezuela
##
## Bronze Silver
##      6      2
##
## $Vietnam
##
## Silver
##      2
##
## $`Virgin Islands*`
##
## Silver
##      1
##
## $`West Germany`
##
## Bronze   Gold Silver
##    126     84    135
##
## $Yugoslavia
##
## Bronze   Gold Silver
##    102     90     86
##
## $Zambia
##
## Bronze Silver
##      1      1
##
```

```
## $Zimbabwe
##
## Bronze   Gold Silver
##      1     18      4
```

3) The Tidyverse Series of Packages

The tidyverse is a collection of packages used in R for data processing.

These packages expect your data to follow the 'tidy' format, which essentially is dictated by three key rules:

1. Every column is a variable
2. Every row is an observation
3. Every cell contains a single value

There are several tidyverse packages that you may have heard of, each of these have different purposes.

We will focus on using Dpylr to explore your data easily in R.

4) Using Dplyr to explore data

When exploring data, it always helpful to have questions you want to answer or explore so you know how to go about your exploratory analysis.

Sometimes its fine to just experiment and explore the data with no real structure as this can lead you to discovering something interesting which you can dig deeper into with your analysis.

Dplyr allows you to explore data easily.

The first step to work with it is to load the package

```
# install.packages('tidyverse') # run this command if you do not have tidyverse installed on your compu
library(tidyverse) # alternatively you can just load up dplyr using 'library(dplyr)'
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
fifa <- read.csv('players_20.csv')
dim(fifa) # we have a lot of columns to work with
```

```
## [1] 18278   104
```

```r
colnames(fifa)
```

```
##   [1] "sofifa_id"                  "player_url"
##   [3] "short_name"                 "long_name"
##   [5] "age"                        "dob"
##   [7] "height_cm"                  "weight_kg"
##   [9] "nationality"                "club"
##  [11] "overall"                    "potential"
##  [13] "value_eur"                  "wage_eur"
##  [15] "player_positions"           "preferred_foot"
##  [17] "international_reputation"   "weak_foot"
##  [19] "skill_moves"                "work_rate"
##  [21] "body_type"                  "real_face"
##  [23] "release_clause_eur"         "player_tags"
##  [25] "team_position"              "team_jersey_number"
##  [27] "loaned_from"                "joined"
##  [29] "contract_valid_until"       "nation_position"
##  [31] "nation_jersey_number"       "pace"
##  [33] "shooting"                   "passing"
##  [35] "dribbling"                  "defending"
##  [37] "physic"                     "gk_diving"
##  [39] "gk_handling"                "gk_kicking"
##  [41] "gk_reflexes"                "gk_speed"
##  [43] "gk_positioning"             "player_traits"
##  [45] "attacking_crossing"         "attacking_finishing"
##  [47] "attacking_heading_accuracy" "attacking_short_passing"
##  [49] "attacking_volleys"          "skill_dribbling"
##  [51] "skill_curve"                "skill_fk_accuracy"
##  [53] "skill_long_passing"         "skill_ball_control"
##  [55] "movement_acceleration"      "movement_sprint_speed"
##  [57] "movement_agility"           "movement_reactions"
##  [59] "movement_balance"           "power_shot_power"
##  [61] "power_jumping"              "power_stamina"
##  [63] "power_strength"             "power_long_shots"
##  [65] "mentality_aggression"       "mentality_interceptions"
##  [67] "mentality_positioning"      "mentality_vision"
##  [69] "mentality_penalties"        "mentality_composure"
##  [71] "defending_marking"          "defending_standing_tackle"
##  [73] "defending_sliding_tackle"   "goalkeeping_diving"
##  [75] "goalkeeping_handling"       "goalkeeping_kicking"
##  [77] "goalkeeping_positioning"    "goalkeeping_reflexes"
##  [79] "ls"                         "st"
##  [81] "rs"                         "lw"
##  [83] "lf"                         "cf"
##  [85] "rf"                         "rw"
##  [87] "lam"                        "cam"
##  [89] "ram"                        "lm"
##  [91] "lcm"                        "cm"
##  [93] "rcm"                        "rm"
##  [95] "lwb"                        "ldm"
##  [97] "cdm"                        "rdm"
##  [99] "rwb"                        "lb"
## [101] "lcb"                        "cb"
```

```
## [103] "rcb"                        "rb"
```

Different Functionalities of Dplyr

1) `select()` - this picks variables based upon their names by columns

`select(data, ...)` essentially takes in the dataframe as the first argument and then rest of the arguments represent the column names you want to select.

```
select(fifa,nationality, dob, age, height_cm, weight_kg) %>% slice(1:10)
```

```
##    nationality        dob age height_cm weight_kg
## 1    Argentina 1987-06-24  32       170        72
## 2     Portugal 1985-02-05  34       187        83
## 3       Brazil 1992-02-05  27       175        68
## 4     Slovenia 1993-01-07  26       188        87
## 5      Belgium 1991-01-07  28       175        74
## 6      Belgium 1991-06-28  28       181        70
## 7      Germany 1992-04-30  27       187        85
## 8  Netherlands 1991-07-08  27       193        92
## 9      Croatia 1985-09-09  33       172        66
## 10       Egypt 1992-06-15  27       175        71
```

You can also use the pipe which is `%>%` and can be generated by holding CMD + SHIFT + M.

The pipe essentially passes the variable on the left as the first argument in the function on the right.

```
fifa %>% select(nationality, dob, age, height_cm, weight_kg) %>% slice(1:10)
```

```
##    nationality        dob age height_cm weight_kg
## 1    Argentina 1987-06-24  32       170        72
## 2     Portugal 1985-02-05  34       187        83
## 3       Brazil 1992-02-05  27       175        68
## 4     Slovenia 1993-01-07  26       188        87
## 5      Belgium 1991-01-07  28       175        74
## 6      Belgium 1991-06-28  28       181        70
## 7      Germany 1992-04-30  27       187        85
## 8  Netherlands 1991-07-08  27       193        92
## 9      Croatia 1985-09-09  33       172        66
## 10       Egypt 1992-06-15  27       175        71
```

You can chain these pipes together to run multiple commands together

```
fifa %>% select(dob, age, height_cm, weight_kg) %>% select(age) %>% slice(1:10)
```

```
##    age
## 1   32
## 2   34
## 3   27
## 4   26
## 5   28
```

```
## 6   28
## 7   27
## 8   27
## 9   33
## 10  27
```

Other things you can do with select:

```
fifa %>% select(dob:weight_kg) %>% slice(1:10) # use the colon to select all columns between dob and we
```

```
##           dob height_cm weight_kg
## 1  1987-06-24       170        72
## 2  1985-02-05       187        83
## 3  1992-02-05       175        68
## 4  1993-01-07       188        87
## 5  1991-01-07       175        74
## 6  1991-06-28       181        70
## 7  1992-04-30       187        85
## 8  1991-07-08       193        92
## 9  1985-09-09       172        66
## 10 1992-06-15       175        71
```

```
fifa %>% select(-dob, -weight_kg) %>% slice(1:10) # use the negative sign to select all columns EXCEPT
```

```
##    sofifa_id
## 1     158023
## 2      20801
## 3     190871
## 4     200389
## 5     183277
## 6     192985
## 7     192448
## 8     203376
## 9     177003
## 10    209331
##                                                               player_url
## 1                  https://sofifa.com/player/158023/lionel-messi/20/159586
## 2  https://sofifa.com/player/20801/c-ronaldo-dos-santos-aveiro/20/159586
## 3   https://sofifa.com/player/190871/neymar-da-silva-santos-jr/20/159586
## 4                     https://sofifa.com/player/200389/jan-oblak/20/159586
## 5                   https://sofifa.com/player/183277/eden-hazard/20/159586
## 6               https://sofifa.com/player/192985/kevin-de-bruyne/20/159586
## 7        https://sofifa.com/player/192448/marc-andre-ter-stegen/20/159586
## 8               https://sofifa.com/player/203376/virgil-van-dijk/20/159586
## 9                   https://sofifa.com/player/177003/luka-modric/20/159586
## 10                https://sofifa.com/player/209331/mohamed-salah/20/159586
##           short_name                           long_name age height_cm
## 1           L. Messi      Lionel Andrés Messi Cuccittini  32       170
## 2  Cristiano Ronaldo Cristiano Ronaldo dos Santos Aveiro  34       187
## 3          Neymar Jr       Neymar da Silva Santos Junior  27       175
## 4           J. Oblak                           Jan Oblak  26       188
## 5          E. Hazard                         Eden Hazard  28       175
```

```
## 6      K. De Bruyne                    Kevin De Bruyne  28       181
## 7     M. ter Stegen              Marc-André ter Stegen  27       187
## 8       V. van Dijk                    Virgil van Dijk  27       193
## 9         L. Modrić                        Luka Modrić  33       172
## 10         M. Salah               Mohamed  Salah Ghaly  27       175
##    nationality               club overall potential value_eur wage_eur
## 1    Argentina       FC Barcelona      94        94  95500000   565000
## 2     Portugal           Juventus      93        93  58500000   405000
## 3       Brazil Paris Saint-Germain      92        92 105500000   290000
## 4     Slovenia    Atlético Madrid      91        93  77500000   125000
## 5      Belgium        Real Madrid      91        91  90000000   470000
## 6      Belgium    Manchester City      91        91  90000000   370000
## 7      Germany       FC Barcelona      90        93  67500000   250000
## 8  Netherlands          Liverpool      90        91  78000000   200000
## 9      Croatia        Real Madrid      90        90  45000000   340000
## 10       Egypt          Liverpool      90        90  80500000   240000
##    player_positions preferred_foot international_reputation weak_foot
## 1        RW, CF, ST           Left                        5         4
## 2            ST, LW          Right                        5         4
## 3           LW, CAM          Right                        5         5
## 4                GK          Right                        3         3
## 5            LW, CF          Right                        4         4
## 6           CAM, CM          Right                        4         5
## 7                GK          Right                        3         4
## 8                CB          Right                        3         3
## 9                CM          Right                        4         4
## 10           RW, ST           Left                        3         3
##    skill_moves     work_rate           body_type real_face release_clause_eur
## 1            4    Medium/Low               Messi       Yes          195800000
## 2            5      High/Low          C. Ronaldo       Yes           96500000
## 3            5   High/Medium              Neymar       Yes          195200000
## 4            1 Medium/Medium              Normal       Yes          164700000
## 5            4   High/Medium              Normal       Yes          184500000
## 6            4     High/High              Normal       Yes          166500000
## 7            1 Medium/Medium              Normal       Yes          143400000
## 8            2 Medium/Medium              Normal       Yes          150200000
## 9            4     High/High                Lean       Yes           92300000
## 10           4   High/Medium PLAYER_BODY_TYPE_25       Yes          148900000
##
## 1                              #Dribbler, #Distance Shooter, #Crosser, #FK Specialist, #Acrobat, #Cl
## 2                                            #Speedster, #Dribbler, #Distance Shooter, #Acrobat, #Cl
## 3  #Speedster, #Dribbler, #Playmaker  , #Crosser, #FK Specialist, #Acrobat, #Clinical Finisher, #Comp
## 4
## 5
## 6                                                      #Dribbler, #Playmaker  , #Engine, #Distance Shoo
## 7
## 8                                                                                     #Tackling , #Tactic
## 9                                                                           #Dribbler, #Playmaker  , #Cros
## 10                                                                     #Speedster, #Dribbler, #Acrobat, #Cl
##    team_position team_jersey_number loaned_from     joined contract_valid_until
## 1             RW                 10             2004-07-01                 2021
## 2             LW                  7             2018-07-10                 2022
## 3            CAM                 10             2017-08-03                 2022
## 4             GK                 13             2014-07-16                 2023
```

```
## 5            LW                 7           2019-07-01               2024
## 6           RCM                17           2015-08-30               2023
## 7            GK                 1           2014-07-01               2022
## 8           LCB                 4           2018-01-01               2023
## 9           RCM                10           2012-08-01               2020
## 10           RW                11           2017-07-01               2023
##    nation_position nation_jersey_number pace shooting passing dribbling
## 1                                    NA   87       92      92        96
## 2               LS                    7   90       93      82        89
## 3               LW                   10   91       85      87        95
## 4               GK                    1   NA       NA      NA        NA
## 5               LF                   10   91       83      86        94
## 6              RCM                    7   76       86      92        86
## 7              SUB                   22   NA       NA      NA        NA
## 8              LCB                    4   77       60      70        71
## 9                                    NA   74       76      89        89
## 10              RW                   10   93       86      81        89
##    defending physic gk_diving gk_handling gk_kicking gk_reflexes gk_speed
## 1         39     66        NA          NA         NA          NA       NA
## 2         35     78        NA          NA         NA          NA       NA
## 3         32     58        NA          NA         NA          NA       NA
## 4         NA     NA        87          92         78          89       52
## 5         35     66        NA          NA         NA          NA       NA
## 6         61     78        NA          NA         NA          NA       NA
## 7         NA     NA        88          85         88          90       45
## 8         90     86        NA          NA         NA          NA       NA
## 9         72     66        NA          NA         NA          NA       NA
## 10        45     74        NA          NA         NA          NA       NA
##    gk_positioning
## 1              NA
## 2              NA
## 3              NA
## 4              90
## 5              NA
## 6              NA
## 7              88
## 8              NA
## 9              NA
## 10             NA
##
## 1  Beat Offside Trap, Argues with Officials, Early Crosser, Finesse Shot, Speed Dribbler (CPU AI Only
## 2                                      Long Throw-in, Selfish, Argues with Officials, Early Crosse
## 3                                                Power Free-Kick, Injury Free, Selfish, Early Cros
## 4
## 5                                                            Beat Offside Trap, Selfish, Finesse S
## 6                            Power Free-Kick, Avoids Using Weaker Foot, Dives Into Tackles (CPU AI Only
## 7
## 8                                                                                Diver, Avoids Using
## 9                                                                    Argues with Officials, Finesse S
## 10                                         Beat Offside Trap, Argues with Officials, Early Crosse
##    attacking_crossing attacking_finishing attacking_heading_accuracy
## 1                  88                  95                         70
## 2                  84                  94                         89
## 3                  87                  87                         62
```

```
## 4                  13                   11                           15
## 5                  81                   84                           61
## 6                  93                   82                           55
## 7                  18                   14                           11
## 8                  53                   52                           86
## 9                  86                   72                           55
## 10                 79                   90                           59
##    attacking_short_passing attacking_volleys skill_dribbling skill_curve
## 1                       92                88              97          93
## 2                       83                87              89          81
## 3                       87                87              96          88
## 4                       43                13              12          13
## 5                       89                83              95          83
## 6                       92                82              86          85
## 7                       61                14              21          18
## 8                       78                45              70          60
## 9                       92                76              87          85
## 10                      84                79              89          83
##    skill_fk_accuracy skill_long_passing skill_ball_control
## 1                 94                 92                 96
## 2                 76                 77                 92
## 3                 87                 81                 95
## 4                 14                 40                 30
## 5                 79                 83                 94
## 6                 83                 91                 91
## 7                 12                 63                 30
## 8                 70                 81                 76
## 9                 78                 88                 92
## 10                69                 75                 89
##    movement_acceleration movement_sprint_speed movement_agility
## 1                     91                    84               93
## 2                     89                    91               87
## 3                     94                    89               96
## 4                     43                    60               67
## 5                     94                    88               95
## 6                     77                    76               78
## 7                     38                    50               37
## 8                     74                    79               61
## 9                     77                    71               92
## 10                    94                    92               91
##    movement_reactions movement_balance power_shot_power power_jumping
## 1                  95               95               86            68
## 2                  96               71               95            95
## 3                  92               84               80            61
## 4                  88               49               59            78
## 5                  90               94               82            56
## 6                  91               76               91            63
## 7                  86               43               66            79
## 8                  88               53               81            90
## 9                  89               93               79            68
## 10                 92               88               80            69
##    power_stamina power_strength power_long_shots mentality_aggression
## 1             75             68               94                   48
## 2             85             78               93                   63
```

```
## 3            81             49                 84                     51
## 4            41             78                 12                     34
## 5            84             63                 80                     54
## 6            89             74                 90                     76
## 7            35             78                 10                     43
## 8            75             92                 64                     82
## 9            85             58                 82                     62
## 10           85             73                 84                     63
##    mentality_interceptions mentality_positioning mentality_vision
## 1                       40                    94               94
## 2                       29                    95               82
## 3                       36                    87               90
## 4                       19                    11               65
## 5                       41                    87               89
## 6                       61                    88               94
## 7                       22                    11               70
## 8                       89                    47               65
## 9                       82                    79               91
## 10                      55                    92               84
##    mentality_penalties mentality_composure defending_marking
## 1                   75                  96                33
## 2                   85                  95                28
## 3                   90                  94                27
## 4                   11                  68                27
## 5                   88                  91                34
## 6                   79                  91                68
## 7                   25                  70                25
## 8                   62                  89                91
## 9                   82                  92                68
## 10                  77                  91                38
##    defending_standing_tackle defending_sliding_tackle goalkeeping_diving
## 1                         37                       26                  6
## 2                         32                       24                  7
## 3                         26                       29                  9
## 4                         12                       18                 87
## 5                         27                       22                 11
## 6                         58                       51                 15
## 7                         13                       10                 88
## 8                         92                       85                 13
## 9                         76                       71                 13
## 10                        43                       41                 14
##    goalkeeping_handling goalkeeping_kicking goalkeeping_positioning
## 1                    11                  15                      14
## 2                    11                  15                      14
## 3                     9                  15                      15
## 4                    92                  78                      90
## 5                    12                   6                       8
## 6                    13                   5                      10
## 7                    85                  88                      88
## 8                    10                  13                      11
## 9                     9                   7                      14
## 10                   14                   9                      11
##    goalkeeping_reflexes   ls   st   rs   lw   lf   cf   rf   rw  lam  cam  ram
## 1                     8 89+2 89+2 89+2 93+2 93+2 93+2 93+2 93+2 93+2 93+2 93+2
```

```
## 2                     11 91+3 91+3 91+3 89+3 90+3 90+3 90+3 89+3 88+3 88+3 88+3
## 3                     11 84+3 84+3 84+3 90+3 89+3 89+3 89+3 90+3 90+3 90+3 90+3
## 4                     89
## 5                      8 83+3 83+3 83+3 89+3 88+3 88+3 88+3 89+3 89+3 89+3 89+3
## 6                     13 82+3 82+3 82+3 87+3 87+3 87+3 87+3 87+3 88+3 88+3 88+3
## 7                     90
## 8                     11 69+3 69+3 69+3 67+3 69+3 69+3 69+3 67+3 69+3 69+3 69+3
## 9                      9 77+3 77+3 77+3 84+3 83+3 83+3 83+3 84+3 86+3 86+3 86+3
## 10                    14 84+3 84+3 84+3 88+3 88+3 88+3 88+3 88+3 87+3 87+3 87+3
##      lm  lcm   cm  rcm   rm  lwb  ldm  cdm  rdm  rwb   lb  lcb   cb  rcb   rb
## 1  92+2 87+2 87+2 87+2 92+2 68+2 66+2 66+2 66+2 68+2 63+2 52+2 52+2 52+2 63+2
## 2  88+3 81+3 81+3 81+3 88+3 65+3 61+3 61+3 61+3 65+3 61+3 53+3 53+3 53+3 61+3
## 3  89+3 82+3 82+3 82+3 89+3 66+3 61+3 61+3 61+3 66+3 61+3 46+3 46+3 46+3 61+3
## 4
## 5  89+3 83+3 83+3 83+3 89+3 66+3 63+3 63+3 63+3 66+3 61+3 49+3 49+3 49+3 61+3
## 6  88+3 87+3 87+3 87+3 88+3 77+3 77+3 77+3 77+3 77+3 73+3 66+3 66+3 66+3 73+3
## 7
## 8  69+3 74+3 74+3 74+3 69+3 79+3 83+3 83+3 83+3 79+3 81+3 87+3 87+3 87+3 81+3
## 9  85+3 87+3 87+3 87+3 85+3 81+3 81+3 81+3 81+3 81+3 79+3 72+3 72+3 72+3 79+3
## 10 87+3 81+3 81+3 81+3 87+3 70+3 67+3 67+3 67+3 70+3 66+3 57+3 57+3 57+3 66+3
```

There are also some special selection functions:

- `contains()` selects columns containing a specified character string
- `starts_with()` and `ends_with()`
- `matches()` selects a column that matches a REGEX pattern
- `everything()` selects all columns
- `num_range()` selects columns from a range
- `one_of(vector of col names)` select columns where the names are stored in a vector

```r
fifa %>% select(starts_with('attacking')) %>% slice_sample(n = 5) # randomly selects 5 rows
```

```
##   attacking_crossing attacking_finishing attacking_heading_accuracy
## 1                 54                  52                         53
## 2                 68                  62                         46
## 3                 27                  64                         63
## 4                 56                  58                         52
## 5                 12                   6                         14
##   attacking_short_passing attacking_volleys
## 1                      53                35
## 2                      69                50
## 3                      57                49
## 4                      73                50
## 5                      28                 8
```

2) `filter()` allows you to filter certain rows using logical subsetting.

```r
# lets say we want rows which players have a height of above 175cm AND weight above 85 kg
fifa %>% filter(height_cm > 175, weight_kg > 85) %>% select(long_name)  %>% slice_sample(n = 5)
```

```
##                long_name
## 1              John Mary
```

```
## 2 Marco Raimondo-Metzger
## 3           Matt Lampson
## 4       George Timotheou
## 5          Matthieu Sans
```

```
# we can also find rows in which players have height above 175cm OR weight above 85
fifa %>% filter(height_cm > 175 | weight_kg > 85) %>% select(long_name)  %>% slice_sample(n = 5)
```

```
##                long_name
## 1           Ryan Sweeney
## 2 Emanuel Rodrigues Novo
## 3           Marcus Maier
## 4      Mirko Pigliacelli
## 5          Adrien Tameze
```

3) `arrange()` allows you to order by column values

```
# by default R arranges by ascending so you need to use the DESC() function to arrange by descending
fifa %>% select(long_name, age) %>% arrange(desc(age), long_name)  %>% slice(1:10)
```

```
##                             long_name age
## 1     Cristian Fernando Muñoz Hoffman  42
## 2  Hussein Omar Abdul Ghani Sulaimani  42
## 3            Cristian David Lucchetti  41
## 4                         Frode Kippe  41
## 5                    Gianluigi Buffon  41
## 6            Vitorino Hilton da Silva  41
## 7          Alberto Cifuentes Martínez  40
## 8       Claudio Miguel Pizarro Bossio  40
## 9                       Dannie Bulman  40
## 10                     Dario Dainelli  40
```

```
fifa %>% select(long_name, age) %>% arrange(long_name, desc(age)) %>% slice(1:10)  # note that the orde
```

```
##                       long_name age
## 1  A. Benjamin Chiamuloira Paes  31
## 2      A. Pimenta Flora Pimenta  20
## 3                    Aapo Halme  21
## 4                 Aaron  Lennon  32
## 5         Aaron Amadi-Holloway  26
## 6        Aaron Anthony Connolly  19
## 7           Aaron Appindangoye  27
## 8                  Aaron Barry  26
## 9              Aaron Bastiaans  17
## 10               Aaron Berzel  27
```

4) `mutate()` allows you to create new variables

```
fifa %>% mutate(overall_value = 0.5 * (value_eur + wage_eur)) %>%  select(overall_value) %>% slice(1:10)
```

```
##    overall_value
## 1       48032500
## 2       29452500
## 3       52895000
## 4       38812500
## 5       45235000
## 6       45185000
## 7       33875000
## 8       39100000
## 9       22670000
## 10      40370000
```

There are several useful functions you can use within mutate:

- `pmin()` and `pmax()` takes in multiple column names and returns the minimum/maximum value between all those columns for each row
- `cummin()` and `cummax()` Cumulative min/max
- `cumsum()`, `cumprod()`, `cummean()`
- `between()` can be used to see if values in a column are between `a` and `b`
- `lead()` and `lag()` copies values with an offset (more useful in time series data)
- `ntile()` - bins values into n buckets

5) `summarise()` allows you to create summary values for your table

```r
fifa  %>% summarise(mean_age = mean(age), mean_height = mean(height_cm), maxmimum_weight = max(weight_kg
```

```
##   mean_age mean_height maxmimum_weight
## 1 25.28329    181.3622             110
```

6) `group_by()` allows you explore how summary statistics vary between each group (group by usually needs to be combined with a `summarise()` command that tells R how to aggregate the neccessary data)

```r
fifa %>% group_by(nationality) # nothing happens unless you call the summary function
```

```
## # A tibble: 18,278 x 104
## # Groups:   nationality [162]
##    sofifa_id player_url short_name long_name   age dob   height_cm weight_kg
##        <int> <chr>      <chr>      <chr>     <int> <chr>     <int>     <int>
##  1    158023 https://s~ L. Messi   Lionel A~    32 1987~       170        72
##  2     20801 https://s~ Cristiano~ Cristian~    34 1985~       187        83
##  3    190871 https://s~ Neymar Jr  Neymar d~    27 1992~       175        68
##  4    200389 https://s~ J. Oblak   Jan Oblak    26 1993~       188        87
##  5    183277 https://s~ E. Hazard  Eden Haz~    28 1991~       175        74
##  6    192985 https://s~ K. De Bru~ Kevin De~    28 1991~       181        70
##  7    192448 https://s~ M. ter St~ Marc-And~    27 1992~       187        85
##  8    203376 https://s~ V. van Di~ Virgil v~    27 1991~       193        92
##  9    177003 https://s~ L. Modrić  Luka Mod~    33 1985~       172        66
## 10    209331 https://s~ M. Salah   Mohamed ~    27 1992~       175        71
## # ... with 18,268 more rows, and 96 more variables: nationality <chr>,
## #   club <chr>, overall <int>, potential <int>, value_eur <int>,
```

```
## #   wage_eur <int>, player_positions <chr>, preferred_foot <chr>,
## #   international_reputation <int>, weak_foot <int>, skill_moves <int>,
## #   work_rate <chr>, body_type <chr>, real_face <chr>,
## #   release_clause_eur <int>, player_tags <chr>, team_position <chr>,
## #   team_jersey_number <int>, loaned_from <chr>, joined <chr>,
## #   contract_valid_until <int>, nation_position <chr>,
## #   nation_jersey_number <int>, pace <int>, shooting <int>, passing <int>,
## #   dribbling <int>, defending <int>, physic <int>, gk_diving <int>,
## #   gk_handling <int>, gk_kicking <int>, gk_reflexes <int>, gk_speed <int>,
## #   gk_positioning <int>, player_traits <chr>, attacking_crossing <int>,
## #   attacking_finishing <int>, attacking_heading_accuracy <int>,
## #   attacking_short_passing <int>, attacking_volleys <int>,
## #   skill_dribbling <int>, skill_curve <int>, skill_fk_accuracy <int>,
## #   skill_long_passing <int>, skill_ball_control <int>,
## #   movement_acceleration <int>, movement_sprint_speed <int>,
## #   movement_agility <int>, movement_reactions <int>, movement_balance <int>,
## #   power_shot_power <int>, power_jumping <int>, power_stamina <int>,
## #   power_strength <int>, power_long_shots <int>, mentality_aggression <int>,
## #   mentality_interceptions <int>, mentality_positioning <int>,
## #   mentality_vision <int>, mentality_penalties <int>,
## #   mentality_composure <int>, defending_marking <int>,
## #   defending_standing_tackle <int>, defending_sliding_tackle <int>,
## #   goalkeeping_diving <int>, goalkeeping_handling <int>,
## #   goalkeeping_kicking <int>, goalkeeping_positioning <int>,
## #   goalkeeping_reflexes <int>, ls <chr>, st <chr>, rs <chr>, lw <chr>,
## #   lf <chr>, cf <chr>, rf <chr>, rw <chr>, lam <chr>, cam <chr>, ram <chr>,
## #   lm <chr>, lcm <chr>, cm <chr>, rcm <chr>, rm <chr>, lwb <chr>, ldm <chr>,
## #   cdm <chr>, rdm <chr>, rwb <chr>, lb <chr>, lcb <chr>, cb <chr>, rcb <chr>,
## #   rb <chr>
```

```
fifa %>% group_by(nationality) %>% summarise(mean_age = mean(age), mean_height = mean(height_cm), maxmi
```

```
## # A tibble: 75 x 5
##    nationality        mean_age mean_height maxmimum_weight number_of_players
##    <chr>                 <dbl>       <dbl>           <int>             <int>
##  1 Croatia                25.2        186.              99               126
##  2 Serbia                 25.8        185.              95               139
##  3 Bosnia Herzegovina     26.2        185.              95                66
##  4 Czech Republic         26.8        185.             102               102
##  5 Senegal                25.6        185.              95               127
##  6 Iceland                26.5        185.              88                46
##  7 Montenegro             24.8        184.              90                33
##  8 Denmark                24.3        184.              98               345
##  9 Slovenia               26.5        184.              92                61
## 10 Germany                24.4        184.             103              1216
## # ... with 65 more rows
```

You can also group by multiple columns:

```
fifa %>% group_by(nationality, team_position) %>% summarise(mean_age = mean(age), mean_height = mean(he
```

```
## `summarise()` has grouped output by 'nationality'. You can override using the `.groups` argument.
```

```
## # A tibble: 147 x 6
## # Groups:   nationality [56]
##    nationality team_position mean_age mean_height maxmimum_weight
##    <chr>       <chr>            <dbl>       <dbl>           <int>
##  1 Argentina   LM                27.8        172.              80
##  2 Colombia    LM                26.2        173.              84
##  3 Argentina   LCM               26.4        174.              83
##  4 Spain       LM                26.4        175.              85
##  5 Argentina   LB                27.3        176.              82
##  6 Saudi Arab~ RES               23          176.              90
##  7 Chile       RES               20.8        176.              83
##  8 Chile       SUB               24.6        176.              94
##  9 Argentina   RB                26          176.              82
## 10 Brazil      CAM               28.8        176.              79
## # ... with 137 more rows, and 1 more variable: number_of_players <int>
```