Week 2.1 - Accessing data using R

In this week, we will use basketball data downloaded from NBA.com to demonstrate how to import data into R, how to clean up data before conducting any data analyses, as well how to describe and summarize data.

Importing data into R

Before we import the dataset into R markdown, we need to first import the R packages that we will use to analyze the data.

- tidyverse: contains the essential R packages for importing data, data manipulation, data analyses, and plotting. Packages included in the tidyverse are ggplot2, dplyr, tidyr, readr, purr, tibble, stringr, and forcats.

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

```
NBA_Teams = read.csv("~/Google Drive/Sports Analytics Moocs/MOOC 1 - Foundations of sports analytics/We
```

In our data repository, we have a dataset that contains NBA team information. Let's import this dataset into R Markdown. We can take a quick look at the data we imported by displaying the dataset.

```
head(NBA_Teams)
```

```
##   X ABBREVIATION        CITY            FULL_NAME         ID  NICKNAME
## 1 0          ATL     Atlanta        Atlanta Hawks 1610612737     Hawks
## 2 1          BOS      Boston       Boston Celtics 1610612738   Celtics
## 3 2          CLE   Cleveland  Cleveland Cavaliers 1610612739 Cavaliers
## 4 3          NOP New Orleans New Orleans Pelicans 1610612740  Pelicans
## 5 4          CHI     Chicago        Chicago Bulls 1610612741     Bulls
## 6 5          DAL      Dallas     Dallas Mavericks 1610612742 Mavericks
##           STATE YEAR_FOUNDED
## 1       Atlanta         1949
## 2 Massachusetts         1946
## 3          Ohio         1970
## 4     Louisiana         2002
## 5      Illinois         1966
## 6         Texas         1980
```

This dataset provides some basic information of the NBA teams.

For a dataset, each row represents an observation, i.e., a team in this dataset and each column represents a variable which contains information of a characteristics of the observation. A variable can take different values in different situations. The number of observation in a dataset represents the size of our sample and the number of variables represents the richness of information in our dataset.

```
dim(NBA_Teams)
```

We can use the "dim" function in Python to see how many variables and observations in our dataset.

```
## [1] 30  8
```

We can see that there are 30 observations (rows) and 8 variables (columns).

Renaming Variables

We can rename a variable using the "rename" function in the dplyr package. The first variable is unnamed, let's rename it to be "TEAM_NUMBER"; let's also rename "ID" to "TEAM_ID."

```
NBA_Teams = NBA_Teams %>% rename(TEAM_NUMBER = X, TEAM_ID = ID)
head(NBA_Teams)
```

```
##   TEAM_NUMBER ABBREVIATION        CITY            FULL_NAME    TEAM_ID
## 1           0          ATL     Atlanta        Atlanta Hawks 1610612737
## 2           1          BOS      Boston       Boston Celtics 1610612738
## 3           2          CLE   Cleveland  Cleveland Cavaliers 1610612739
## 4           3          NOP New Orleans New Orleans Pelicans 1610612740
## 5           4          CHI     Chicago        Chicago Bulls 1610612741
## 6           5          DAL      Dallas     Dallas Mavericks 1610612742
##    NICKNAME         STATE YEAR_FOUNDED
## 1     Hawks       Atlanta         1949
## 2   Celtics Massachusetts         1946
## 3 Cavaliers          Ohio         1970
## 4  Pelicans     Louisiana         2002
## 5     Bulls      Illinois         1966
## 6 Mavericks         Texas         1980
```

```
tail(NBA_Teams)
```

```
##    TEAM_NUMBER ABBREVIATION       CITY          FULL_NAME    TEAM_ID  NICKNAME
## 25          24          TOR    Toronto    Toronto Raptors 1610612761   Raptors
## 26          25          UTA       Utah          Utah Jazz 1610612762      Jazz
## 27          26          MEM    Memphis  Memphis Grizzlies 1610612763 Grizzlies
## 28          27          WAS Washington Washington Wizards 1610612764   Wizards
## 29          28          DET    Detroit    Detroit Pistons 1610612765   Pistons
## 30          29          CHA  Charlotte  Charlotte Hornets 1610612766   Hornets
##                   STATE YEAR_FOUNDED
## 25              Ontario         1995
## 26                 Utah         1974
## 27            Tennessee         1995
## 28 District of Columbia         1961
## 29             Michigan         1948
## 30       North Carolina         1988
```

Self Test

- Rename "FULL_NAME" to "TEAM_NAME"

```
NBA_Teams = NBA_Teams %>% rename(TEAM_NAME = FULL_NAME)
head(NBA_Teams)
```

```
##   TEAM_NUMBER ABBREVIATION        CITY            TEAM_NAME    TEAM_ID
## 1           0          ATL     Atlanta        Atlanta Hawks 1610612737
## 2           1          BOS      Boston       Boston Celtics 1610612738
## 3           2          CLE   Cleveland  Cleveland Cavaliers 1610612739
## 4           3          NOP New Orleans New Orleans Pelicans 1610612740
## 5           4          CHI     Chicago        Chicago Bulls 1610612741
## 6           5          DAL      Dallas     Dallas Mavericks 1610612742
##    NICKNAME         STATE YEAR_FOUNDED
## 1     Hawks       Atlanta         1949
## 2   Celtics Massachusetts         1946
## 3 Cavaliers          Ohio         1970
## 4  Pelicans     Louisiana         2002
## 5     Bulls      Illinois         1966
## 6 Mavericks         Texas         1980
```

Dropping Variables (columns)

To drop a variable, i.e., to delete a column, we can use the "select" command.

- We need to provide the name of the variable with a minus which is used to drop the columns by name

The variable "TEAM_NUMBER" has little meaning, let's drop it.

```
NBA_Teams = NBA_Teams %>% select(-TEAM_NUMBER)
head(NBA_Teams)
```

```
##   ABBREVIATION        CITY            TEAM_NAME    TEAM_ID  NICKNAME
## 1          ATL     Atlanta        Atlanta Hawks 1610612737     Hawks
## 2          BOS      Boston       Boston Celtics 1610612738   Celtics
## 3          CLE   Cleveland  Cleveland Cavaliers 1610612739 Cavaliers
## 4          NOP New Orleans New Orleans Pelicans 1610612740  Pelicans
## 5          CHI     Chicago        Chicago Bulls 1610612741     Bulls
## 6          DAL      Dallas     Dallas Mavericks 1610612742 Mavericks
##           STATE YEAR_FOUNDED
## 1       Atlanta         1949
## 2 Massachusetts         1946
## 3          Ohio         1970
## 4     Louisiana         2002
## 5      Illinois         1966
## 6         Texas         1980
```

Next we will work on game level data. Import the game level dataset from our data repository. - We can display just first six rows of the dataset using the "head" command.

```
Games = read_csv("~/Google Drive/Sports Analytics Moocs/MOOC 1 - Foundations of sports analytics/Week 2/
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   TEAM_ABBREVIATION = col_character(),
##   TEAM_NAME = col_character(),
```

```
##   GAME_DATE = col_character(),
##   MATCHUP = col_character(),
##   WL = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
head(Games)
```

```
## # A tibble: 6 x 28
##   SEASON_ID TEAM_ID TEAM_ABBREVIATI~ TEAM_NAME GAME_ID GAME_DATE MATCHUP WL
##       <dbl>   <dbl> <chr>            <chr>       <dbl> <chr>     <chr>   <chr>
## 1     22019  1.61e9 WAS              Washingt~  1.02e9 8/5/2019  WAS @ ~ <NA>
## 2     22019  1.61e9 LVA              Las Vega~  1.02e9 8/5/2019  LVA vs~ <NA>
## 3     22019  1.61e9 LAS              Los Ange~  1.02e9 8/1/2019  LAS vs~ W
## 4     22019  1.61e9 CON              Connecti~  1.02e9 8/1/2019  CON vs~ W
## 5     22019  1.61e9 PHO              Phoenix ~  1.02e9 8/1/2019  PHO @ ~ L
## 6     22019  1.61e9 LVA              Las Vega~  1.02e9 8/1/2019  LVA @ ~ L
## # ... with 20 more variables: MIN <dbl>, PTS <dbl>, FGM <dbl>, FGA <dbl>,
## #   FG_PCT <dbl>, FG3M <dbl>, FG3A <dbl>, FG3_PCT <dbl>, FTM <dbl>, FTA <dbl>,
## #   FT_PCT <dbl>, OREB <dbl>, DREB <dbl>, REB <dbl>, AST <dbl>, STL <dbl>,
## #   BLK <dbl>, TOV <dbl>, PF <dbl>, PLUS_MINUS <dbl>
```

Upon importing the game data, we notice that the first five games are not NBA games, instead, they are WNBA games. Indeed, this dataset contains NBA games, WNBA games, NBA 2K (simulation video) games.

Dropping observations (rows)

```
Games = Games[-1,]
head(Games)
```

To drop an observation, we can use the index number on the left to specify the row we want to drop.

```
## # A tibble: 6 x 28
##   SEASON_ID TEAM_ID TEAM_ABBREVIATI~ TEAM_NAME GAME_ID GAME_DATE MATCHUP WL
##       <dbl>   <dbl> <chr>            <chr>       <dbl> <chr>     <chr>   <chr>
## 1     22019  1.61e9 LVA              Las Vega~  1.02e9 8/5/2019  LVA vs~ <NA>
## 2     22019  1.61e9 LAS              Los Ange~  1.02e9 8/1/2019  LAS vs~ W
## 3     22019  1.61e9 CON              Connecti~  1.02e9 8/1/2019  CON vs~ W
## 4     22019  1.61e9 PHO              Phoenix ~  1.02e9 8/1/2019  PHO @ ~ L
## 5     22019  1.61e9 LVA              Las Vega~  1.02e9 8/1/2019  LVA @ ~ L
## 6     22019  1.61e9 NYL              New York~  1.02e9 8/1/2019  NYL @ ~ L
## # ... with 20 more variables: MIN <dbl>, PTS <dbl>, FGM <dbl>, FGA <dbl>,
## #   FG_PCT <dbl>, FG3M <dbl>, FG3A <dbl>, FG3_PCT <dbl>, FTM <dbl>, FTA <dbl>,
## #   FT_PCT <dbl>, OREB <dbl>, DREB <dbl>, REB <dbl>, AST <dbl>, STL <dbl>,
## #   BLK <dbl>, TOV <dbl>, PF <dbl>, PLUS_MINUS <dbl>
```

More often, we will drop observations based on certain conditions. For example, Las Vegas Aces is a women's basketball team. If we are only going to focus on men's basketball games, we will drop all the games played by Las Vegas Aces. In this case, we can use the filter function from the dplyr package. We can specify our TEAM_NAME variables to be not equal to "Las Vegas Aces."

```
Games = Games %>% filter(TEAM_NAME !="Las Vegas Aces")
head(Games)
```

```
## # A tibble: 6 x 28
```

```
##   SEASON_ID TEAM_ID TEAM_ABBREVIATI~ TEAM_NAME GAME_ID GAME_DATE MATCHUP WL
##       <dbl>   <dbl> <chr>            <chr>       <dbl> <chr>     <chr>   <chr>
## 1     22019  1.61e9 LAS              Los Ange~  1.02e9 8/1/2019  LAS vs~ W
## 2     22019  1.61e9 CON              Connecti~  1.02e9 8/1/2019  CON vs~ W
## 3     22019  1.61e9 PHO              Phoenix ~  1.02e9 8/1/2019  PHO @ ~ L
## 4     22019  1.61e9 NYL              New York~  1.02e9 8/1/2019  NYL @ ~ L
## 5     22019  1.61e9 DAL              Dallas W~  1.02e9 8/1/2019  DAL vs~ W
## 6     22019  1.61e9 IND              Indiana ~  1.02e9 7/31/2019 IND vs~ W
## # ... with 20 more variables: MIN <dbl>, PTS <dbl>, FGM <dbl>, FGA <dbl>,
## #   FG_PCT <dbl>, FG3M <dbl>, FG3A <dbl>, FG3_PCT <dbl>, FTM <dbl>, FTA <dbl>,
## #   FT_PCT <dbl>, OREB <dbl>, DREB <dbl>, REB <dbl>, AST <dbl>, STL <dbl>,
## #   BLK <dbl>, TOV <dbl>, PF <dbl>, PLUS_MINUS <dbl>
```

Self Test

- Drop all the Phoenix Mercury games

```
Games = Games %>% filter(TEAM_NAME !="Phoenix Mercury")
head(Games)
```

```
## # A tibble: 6 x 28
##   SEASON_ID TEAM_ID TEAM_ABBREVIATI~ TEAM_NAME GAME_ID GAME_DATE MATCHUP WL
##       <dbl>   <dbl> <chr>            <chr>       <dbl> <chr>     <chr>   <chr>
## 1     22019  1.61e9 LAS              Los Ange~  1.02e9 8/1/2019  LAS vs~ W
## 2     22019  1.61e9 CON              Connecti~  1.02e9 8/1/2019  CON vs~ W
## 3     22019  1.61e9 NYL              New York~  1.02e9 8/1/2019  NYL @ ~ L
## 4     22019  1.61e9 DAL              Dallas W~  1.02e9 8/1/2019  DAL vs~ W
## 5     22019  1.61e9 IND              Indiana ~  1.02e9 7/31/2019 IND vs~ W
## 6     22019  1.61e9 ATL              Atlanta ~  1.02e9 7/31/2019 ATL @ ~ L
## # ... with 20 more variables: MIN <dbl>, PTS <dbl>, FGM <dbl>, FGA <dbl>,
## #   FG_PCT <dbl>, FG3M <dbl>, FG3A <dbl>, FG3_PCT <dbl>, FTM <dbl>, FTA <dbl>,
## #   FT_PCT <dbl>, OREB <dbl>, DREB <dbl>, REB <dbl>, AST <dbl>, STL <dbl>,
## #   BLK <dbl>, TOV <dbl>, PF <dbl>, PLUS_MINUS <dbl>
```

Merging Dataframes

We will only focus on NBA games. We could merge the NBA_Teams and Games datasets to filter out NBA games.

```
NBA_Games = left_join(NBA_Teams, Games, by=c('TEAM_ID', 'TEAM_NAME'))
head(NBA_Games)
```

Teams are identified by the TEAM_ID. So, let's merge the datasets by TEAM_ID. Since the variable "TEAM_NAME" is also present in both datasets, we could also include this variable as a criteria to merge the datasets so that in our new dataset, there is no duplicate variables.

```
##   ABBREVIATION    CITY     TEAM_NAME    TEAM_ID NICKNAME   STATE YEAR_FOUNDED
## 1          ATL Atlanta Atlanta Hawks 1610612737    Hawks Atlanta         1949
## 2          ATL Atlanta Atlanta Hawks 1610612737    Hawks Atlanta         1949
## 3          ATL Atlanta Atlanta Hawks 1610612737    Hawks Atlanta         1949
## 4          ATL Atlanta Atlanta Hawks 1610612737    Hawks Atlanta         1949
## 5          ATL Atlanta Atlanta Hawks 1610612737    Hawks Atlanta         1949
## 6          ATL Atlanta Atlanta Hawks 1610612737    Hawks Atlanta         1949
##   SEASON_ID TEAM_ABBREVIATION    GAME_ID GAME_DATE     MATCHUP   WL MIN PTS FGM
## 1     22019               ATL 1521900072 7/12/2019   ATL @ SAS    W 201  80  27
```

```
## 2     22019              ATL 1521900060 7/11/2019   ATL @ WAS    L 200  71  26
## 3     22019              ATL 1521900042  7/9/2019 ATL vs. IND    W 202  87  31
## 4     22019              ATL 1521900023  7/7/2019 ATL vs. MIN <NA>  21   6   2
## 5     22019              ATL 1521900023  7/7/2019 ATL vs. MIN    L 178  60  18
## 6     22019              ATL 1521900013  7/6/2019   ATL @ MIL    L 201  83  25
##   FGA FG_PCT FG3M FG3A FG3_PCT FTM FTA FT_PCT OREB DREB REB AST STL BLK TOV PF
## 1  79  0.342    9   32   0.281  17  20  0.850   13   23  36  14  15   3  12 24
## 2  68  0.382   12   29   0.414   7  10  0.700    9   28  37  19  10   8  22 25
## 3  60  0.517    8   21   0.381  17  24  0.708    7   27  34  17   5   5  18 21
## 4   8  0.250    1    3   0.333   1   2  0.500    1    2   3   1   0   1   4  2
## 5  62  0.290    4   22   0.182  20  32  0.625    9   27  36   7   7  10  18 28
## 6  73  0.342   10   32   0.313  23  26  0.885    9   30  39  13  11   6  13 21
##   PLUS_MINUS
## 1        8.0
## 2       -5.0
## 3       18.2
## 4        0.0
## 5      -24.0
## 6        2.0
```

Understanding and cleaning the merged dataset

As you can tell, the merged dataset has a lot more variables and R cannot fit all of them in the screen.

```
colnames(NBA_Games)
```

We can obtain the list of variables using the "colnames" command. This provides us a full list of variables in our dataset.

```
##  [1] "ABBREVIATION"      "CITY"              "TEAM_NAME"
##  [4] "TEAM_ID"           "NICKNAME"          "STATE"
##  [7] "YEAR_FOUNDED"      "SEASON_ID"         "TEAM_ABBREVIATION"
## [10] "GAME_ID"           "GAME_DATE"         "MATCHUP"
## [13] "WL"                "MIN"               "PTS"
## [16] "FGM"               "FGA"               "FG_PCT"
## [19] "FG3M"              "FG3A"              "FG3_PCT"
## [22] "FTM"               "FTA"               "FT_PCT"
## [25] "OREB"              "DREB"              "REB"
## [28] "AST"               "STL"               "BLK"
## [31] "TOV"               "PF"                "PLUS_MINUS"
```

Data Cleaning The variable "ABBREVIATION" AND "TEAM_ABBREVIATION" carry the same information and it is not necessary to keep both of them. - Delete "ABBREVIATION"

```
NBA_Games = NBA_Games %>% select(-ABBREVIATION)
```

Self Test

- Find the number of observations and the number of variables in the dataset

```
dim(NBA_Games)
```

```
## [1] 18421    32
```

The merged dataset is sorted by the criteria we use to merge the datasets. Thus, the NBA_Games dataset is currently sorted by "TEAM_ID." We may be interested to sort the data by other criteria, for example, the date of the game.

We can do so by using the "arrange" function in combination with the function desc() in the dplyr package. In our dataset, "GAME_ID" is created based on the date of the game. We can sort the games by "GAME_ID" and display the 20 most recent games.

```
NBA_Games = NBA_Games %>% arrange(desc(GAME_ID))
head(NBA_Games, 20)
```

```
##           CITY           TEAM_NAME    TEAM_ID  NICKNAME     STATE YEAR_FOUNDED
## 1  San Antonio   San Antonio Spurs 1610612759     Spurs     Texas         1976
## 2         Utah           Utah Jazz 1610612762      Jazz      Utah         1974
## 3    Cleveland Cleveland Cavaliers 1610612739 Cavaliers      Ohio         1970
## 4      Memphis   Memphis Grizzlies 1610612763 Grizzlies Tennessee         1995
## 5    Cleveland Cleveland Cavaliers 1610612739 Cavaliers      Ohio         1970
## 6         Utah           Utah Jazz 1610612762      Jazz      Utah         1974
## 7  San Antonio   San Antonio Spurs 1610612759     Spurs     Texas         1976
## 8      Memphis   Memphis Grizzlies 1610612763 Grizzlies Tennessee         1995
## 9         Utah           Utah Jazz 1610612762      Jazz      Utah         1974
## 10     Memphis   Memphis Grizzlies 1610612763 Grizzlies Tennessee         1995
## 11   Cleveland Cleveland Cavaliers 1610612739 Cavaliers      Ohio         1970
## 12 San Antonio   San Antonio Spurs 1610612759     Spurs     Texas         1976
## 13     Atlanta       Atlanta Hawks 1610612737     Hawks   Atlanta         1949
## 14        Utah           Utah Jazz 1610612762      Jazz      Utah         1974
## 15 San Antonio   San Antonio Spurs 1610612759     Spurs     Texas         1976
## 16     Memphis   Memphis Grizzlies 1610612763 Grizzlies Tennessee         1995
## 17        Utah           Utah Jazz 1610612762      Jazz      Utah         1974
## 18     Memphis   Memphis Grizzlies 1610612763 Grizzlies Tennessee         1995
## 19     Atlanta       Atlanta Hawks 1610612737     Hawks   Atlanta         1949
## 20 San Antonio   San Antonio Spurs 1610612759     Spurs     Texas         1976
##    SEASON_ID TEAM_ABBREVIATION    GAME_ID GAME_DATE     MATCHUP WL MIN PTS FGM
## 1      22019               SAS 1621900006  7/3/2019   SAS @ UTA  L 200  81  29
## 2      22019               UTA 1621900006  7/3/2019 UTA vs. SAS  W 199  84  30
## 3      22019               CLE 1621900005  7/3/2019   CLE @ MEM  L 201  68  22
## 4      22019               MEM 1621900005  7/3/2019 MEM vs. CLE  W 199  81  31
## 5      22019               CLE 1621900004  7/2/2019   CLE @ UTA  L 200  71  24
## 6      22019               UTA 1621900004  7/2/2019 UTA vs. CLE  W 200  86  34
## 7      22019               SAS 1621900003  7/2/2019 SAS vs. MEM  W 199  99  34
## 8      22019               MEM 1621900003  7/2/2019   MEM @ SAS  L 202  84  31
## 9      22019               UTA 1621900002  7/1/2019 UTA vs. MEM  L 200  68  29
## 10     22019               MEM 1621900002  7/1/2019   MEM @ UTA  W 201  85  28
## 11     22019               CLE 1621900001  7/1/2019 CLE vs. SAS  L 200  89  38
## 12     22019               SAS 1621900001  7/1/2019   SAS @ CLE  W 200  97  34
## 13     22018               ATL 1621800006  7/5/2018   ATL @ UTA  L 201  87  30
## 14     22018               UTA 1621800006  7/5/2018 UTA vs. ATL  W 200  92  32
## 15     22018               SAS 1621800005  7/5/2018 SAS vs. MEM  W 201  94  32
## 16     22018               MEM 1621800005  7/5/2018   MEM @ SAS  L 200  87  33
## 17     22018               UTA 1621800004  7/3/2018 UTA vs. MEM  L 200  92  34
## 18     22018               MEM 1621800004  7/3/2018   MEM @ UTA  W 198  95  39
## 19     22018               ATL 1621800003  7/3/2018 ATL vs. SAS  L 200  81  31
## 20     22018               SAS 1621800003  7/3/2018   SAS @ ATL  W 200 103  36
##    FGA FG_PCT FG3M FG3A FG3_PCT FTM FTA FT_PCT OREB DREB REB AST STL BLK TOV PF
## 1   70  0.414    7   17   0.412  16  26  0.615    9   29  38  12   5   3  11 14
```

```
## 2   74  0.405   11   31   0.355  13  19  0.684   13   31  44  15   6   5  15 25
## 3   60  0.367   10   35   0.286  14  19  0.737    3   26  29  15   9   3  17 12
## 4   74  0.419   10   28   0.357   9  13  0.692   11   36  47  19   8   5  14 19
## 5   74  0.324    8   29   0.276  15  22  0.682    6   27  33  13   6   0   6 14
## 6   70  0.486    6   25   0.240  12  16  0.750   11   34  45  18   3   4  14 18
## 7   76  0.447   11   27   0.407  20  23  0.870   10   35  45  24  10   7  18 24
## 8   79  0.392    9   32   0.281  13  15  0.867   10   29  39  17   6   5  17 18
## 9   79  0.367    4   24   0.167   6  13  0.462   14   27  41  16  10   7  19 26
## 10  65  0.431   10   22   0.455  19  27  0.704    8   32  40  14  14   4  20 19
## 11  85  0.447    9   36   0.250   4   6  0.667   14   17  31  22  11   3  14 26
## 12  54  0.630    5   12   0.417  24  34  0.706    5   26  31  18   8   2  22 15
## 13  86  0.349   10   34   0.294  17  26  0.654   15   33  48  16  11   5  17 25
## 14  72  0.444    9   24   0.375  19  30  0.633    8   36  44  21  11   7  19 25
## 15  69  0.464   15   25   0.600  15  22  0.682    6   29  35  20   6   2  14 13
## 16  69  0.478    9   19   0.474  12  17  0.706    6   26  32  16   7   6  11 24
## 17  71  0.479    7   22   0.318  17  22  0.773    8   35  43  19   4   3  15 23
## 18  83  0.470    6   18   0.333  11  15  0.733    9   27  36  18   5   4   5 17
## 19  90  0.344    7   33   0.212  12  22  0.545   14   29  43  12   9   2  12 25
## 20  70  0.514    9   21   0.429  22  28  0.786    6   45  51  24   9  10  17 23
##    PLUS_MINUS
## 1        -7.2
## 2         5.8
## 3       -13.0
## 4        13.0
## 5       -15.0
## 6        14.6
## 7        16.6
## 8       -15.8
## 9        -6.6
## 10        7.4
## 11       -8.0
## 12        7.0
## 13        2.2
## 14       -1.4
## 15        3.2
## 16       -3.2
## 17       -7.0
## 18        8.4
## 19      -26.2
## 20       25.8
```

Missing Values

Before we move on to doing any data analyses, we usually need to check if there is any missing value, that is, the source may have failed to collect some information.

```
sapply(NBA_Games, function(x) sum(is.na(x)))
```

We can use "sum(is.na(x))" which will return the number of missing values for one column. To apply this to all of the columns in the dataset, we need to use "sapply."

```
##              CITY         TEAM_NAME           TEAM_ID          NICKNAME
##                 0                 0                 0                 0
##             STATE     YEAR_FOUNDED         SEASON_ID TEAM_ABBREVIATION
```

```
##                0                0                0                0
##          GAME_ID        GAME_DATE          MATCHUP               WL
##                0                0                0                7
##              MIN              PTS              FGM              FGA
##                0                0                0                0
##           FG_PCT             FG3M             FG3A          FG3_PCT
##                2                0                0                3
##              FTM              FTA           FT_PCT             OREB
##                0                0                3                0
##             DREB              REB              AST              STL
##                0                0                0                0
##              BLK              TOV               PF        PLUS_MINUS
##                0                0                0                0
```

There are missing values in variable WL, FG_PCT, FG3_PCT, and FT_PCT.

Detecting missing values We can use the is.na() function to detect where the missing values are.

```
head(!is.na(NBA_Games))
```

```
##      CITY TEAM_NAME TEAM_ID NICKNAME STATE YEAR_FOUNDED SEASON_ID
## [1,] TRUE      TRUE    TRUE     TRUE  TRUE         TRUE      TRUE
## [2,] TRUE      TRUE    TRUE     TRUE  TRUE         TRUE      TRUE
## [3,] TRUE      TRUE    TRUE     TRUE  TRUE         TRUE      TRUE
## [4,] TRUE      TRUE    TRUE     TRUE  TRUE         TRUE      TRUE
## [5,] TRUE      TRUE    TRUE     TRUE  TRUE         TRUE      TRUE
## [6,] TRUE      TRUE    TRUE     TRUE  TRUE         TRUE      TRUE
##      TEAM_ABBREVIATION GAME_ID GAME_DATE MATCHUP   WL  MIN  PTS  FGM  FGA
## [1,]              TRUE    TRUE      TRUE    TRUE TRUE TRUE TRUE TRUE TRUE
## [2,]              TRUE    TRUE      TRUE    TRUE TRUE TRUE TRUE TRUE TRUE
## [3,]              TRUE    TRUE      TRUE    TRUE TRUE TRUE TRUE TRUE TRUE
## [4,]              TRUE    TRUE      TRUE    TRUE TRUE TRUE TRUE TRUE TRUE
## [5,]              TRUE    TRUE      TRUE    TRUE TRUE TRUE TRUE TRUE TRUE
## [6,]              TRUE    TRUE      TRUE    TRUE TRUE TRUE TRUE TRUE TRUE
##      FG_PCT FG3M FG3A FG3_PCT  FTM  FTA FT_PCT OREB DREB  REB  AST  STL  BLK
## [1,]   TRUE TRUE TRUE    TRUE TRUE TRUE   TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [2,]   TRUE TRUE TRUE    TRUE TRUE TRUE   TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [3,]   TRUE TRUE TRUE    TRUE TRUE TRUE   TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [4,]   TRUE TRUE TRUE    TRUE TRUE TRUE   TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [5,]   TRUE TRUE TRUE    TRUE TRUE TRUE   TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [6,]   TRUE TRUE TRUE    TRUE TRUE TRUE   TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##       TOV   PF PLUS_MINUS
## [1,] TRUE TRUE       TRUE
## [2,] TRUE TRUE       TRUE
## [3,] TRUE TRUE       TRUE
## [4,] TRUE TRUE       TRUE
## [5,] TRUE TRUE       TRUE
## [6,] TRUE TRUE       TRUE
```

Handling Missing Values

There are two main approaches to handle missing values. - First, we can simply drop the observations with missing value.

```
NBA_Games = NBA_Games %>% filter(!is.na(FG_PCT))
dim(NBA_Games)
```

Drop observations with missing value in the variable "FG_PCT"

```
## [1] 18419    32
```

- Second, we can replace the missing values with valid values (Imputation), such as mean and median.

```
col_means <- lapply(NBA_Games, mean, na.rm = TRUE)
NBA_Games <- replace_na(NBA_Games, col_means)

sapply(NBA_Games, function(x) sum(is.na(x)))
```

We can use lapply to first get the column means, and then replace all the NA values with the column means using the tidyr package's "replace_na" function.

```
##               CITY          TEAM_NAME            TEAM_ID           NICKNAME
##                  0                  0                  0                  0
##              STATE      YEAR_FOUNDED          SEASON_ID TEAM_ABBREVIATION
##                  0                  0                  0                  0
##            GAME_ID          GAME_DATE            MATCHUP                 WL
##                  0                  0                  0                  5
##                MIN                PTS                FGM                FGA
##                  0                  0                  0                  0
##             FG_PCT              FG3M               FG3A           FG3_PCT
##                  0                  0                  0                  0
##                FTM                FTA             FT_PCT               OREB
##                  0                  0                  0                  0
##               DREB                REB                AST                STL
##                  0                  0                  0                  0
##                BLK                TOV                 PF         PLUS_MINUS
##                  0                  0                  0                  0
```

Creating variables

We can create a variable equals to the total number of goals made.

```
NBA_Games$GM = NBA_Games$FGM + NBA_Games$FG3M + NBA_Games$FTM
```

Self Test

- Create a variable called "GA" equals to the total number of goals attempted.

```
NBA_Games$GA = NBA_Games$FGA + NBA_Games$FG3A + NBA_Games$FTA
```

Create variables based on conditions

- We can create a variable conditional on the value of another variable.

For example, we can create a variable "RESULT" that equals to 'W' if the team won the game and 'L' otherwise. The result of the game can be captured in the points of the team receive, whether it was positive or negative.

```
NBA_Games$RESULT = ifelse(NBA_Games$PLUS_MINUS > 0, 'W', 'L')
```

We will now drop this newly created "RESULT" variable.

```
NBA_Games = NBA_Games %>% select(-RESULT)
```

Create a variable within group

In the dataset, each game has two observations, one represents the statistics of the home team, one represents those of the away team. Both observations have the same GAME_ID. We can create a variable "POINT_DIFF" that equals the difference between the points earned by the two teams.

We will first sort the data not only by the "GAME_ID" but also by the result "WL".

```
NBA_Games = NBA_Games %>% arrange(GAME_ID, WL)
NBA_Games = NBA_Games %>% group_by(GAME_ID) %>%
    mutate(POINT_DIFF = PTS - lag(PTS)) %>% ungroup()
```

The "POINT_DIFF" variable only has the point difference for the winning team, we need to impute the point difference for the losing team as well.

```
impute.mean <- function(x) replace(x, is.na(x), mean(x, na.rm = TRUE))

NBA_Games = NBA_Games %>% group_by(GAME_ID) %>%
    mutate(POINT_DIFF = impute.mean(POINT_DIFF)) %>% ungroup()
```

- We can also drop all observations with missing value in at least one variable using the "drop_na()" command in the tidyr package.

```
NBA_Games = drop_na(NBA_Games)
dim(NBA_Games)
```

```
## [1] 17779    35
```

Creating new dataframe

Create a new dataframe that aggregates information by group Sometimes we may want to work with season level data rather than team level data. We can create a new dataset that includes aggregate information of team statistics in each season.

```
NBA_Team_Stats = NBA_Games %>%
  select(TEAM_ID, SEASON_ID, PTS, FGM, FGA, FG_PCT, FG3M, FG3A,
         FG3_PCT, FTM, FTA, FT_PCT, OREB, DREB, REB, AST, STL,
         BLK, TOV, PF, PLUS_MINUS) %>%
  group_by(TEAM_ID, SEASON_ID) %>%
  summarise_all(list(sum)) %>% ungroup()
head(NBA_Team_Stats)
```

```
## # A tibble: 6 x 21
##   TEAM_ID SEASON_ID   PTS   FGM   FGA FG_PCT  FG3M  FG3A FG3_PCT   FTM   FTA
##     <dbl>     <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
## 1  1.61e9     12013   523   193   459   2.52    34   131    1.58   103   140
## 2  1.61e9     12014   708   247   554   3.12    66   182    2.55   148   197
## 3  1.61e9     12015   652   226   548   2.90    59   176    2.34   141   175
## 4  1.61e9     12016   686   261   593   3.08    50   159    2.14   114   153
## 5  1.61e9     12017   480   167   410   2.04    51   156    1.61    95   125
## 6  1.61e9     12018   563   206   445   2.31    63   189    1.66    88   124
## # ... with 10 more variables: FT_PCT <dbl>, OREB <dbl>, DREB <dbl>, REB <dbl>,
## #   AST <dbl>, STL <dbl>, BLK <dbl>, TOV <dbl>, PF <dbl>, PLUS_MINUS <dbl>
```

Notice that the newly created dataset has two levels of index, the "TEAM_ID" and "SEASON_ID"

```
NBA_Team_Stats = NBA_Team_Stats %>% ungroup()
```

If we want to convert these two indexes back as variables, we can use the "ungroup" command.

We can create a variable that equals to the total number of observations within a specified group using the count() command.

- Create a variable that equals to the total number of games played by a team in each season, name this variable "GAME_COUNT".

```
NBA_Game_Count = NBA_Games %>% count(TEAM_ID, SEASON_ID, name = "GAME_COUNT")
head(NBA_Game_Count)
```

```
## # A tibble: 6 x 3
##      TEAM_ID SEASON_ID GAME_COUNT
##        <dbl>     <dbl>      <int>
## 1 1610612737     12013          6
## 2 1610612737     12014          7
## 3 1610612737     12015          7
## 4 1610612737     12016          7
## 5 1610612737     12017          5
## 6 1610612737     12018          5
```

Saving data

We can save a dataframe by exporting the edited dataframe to csv file using the "write.csv" command.

- Save merged data as a csv file *We can use the "row.names=FALSE" command to save the data without adding the index as a column in the csv file*

```
write.csv(NBA_Games, "NBA_Games.csv", row.names=FALSE)
```