# Intro to R

## Week 1

### Monday

On Monday we did a brief introduction to R. We looked at making functions, variables, and played around with string substitution and printing out a little message.

```
function1 <- function(name, years){
  age <- 365*years
  final <- paste("Hello", name, "you are", age, "days old!")
  print(final)
}

function2 <- function(name, years){
  age <- 365*years
  sprintf("Hello %s, you are %d days old!", name, age)
}

function1("Ephraim", 24)
```

```
## [1] "Hello Ephraim you are 8760 days old!"
```

```
function2("Ephraim", 24)
```

```
## [1] "Hello Ephraim, you are 8760 days old!"
```

```
players <- read.csv("../data/players.csv")
games_details <- read.csv("../data/games_details.csv")
games <- read.csv("../data/games.csv")
teams <- read.csv("../data/teams.csv")
ranking <- read.csv("../data/ranking.csv")
```

### Tuesday

Today we'll continue learning R and playing around with the data a little bit.

Let's start by making a function that says something about you depending on which name you put in.

```
teams$mean_arena <- mean(teams$ARENACAPACITY, na.rm = TRUE)
# hist(teams$ARENACAPACITY, breaks=12, col="red")
print(teams[teams$ARENACAPACITY == 0 , "ABBREVIATION"])
```

```
## [1] NA    NA    "ORL" NA    NA
```

We learned about tidyverse and ggplot and the powerful syntax they provide.

```
library("ggplot2")
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## v purrr   0.3.4
```

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

We showed how things could be accomplished in both base R and in tidyverse.

```
teams$mean_size <- mean(teams$ARENACAPACITY, na.rm = TRUE)
```

```
ranking <- ranking %>% separate(STANDINGSDATE, c("year","month", "day"))
```

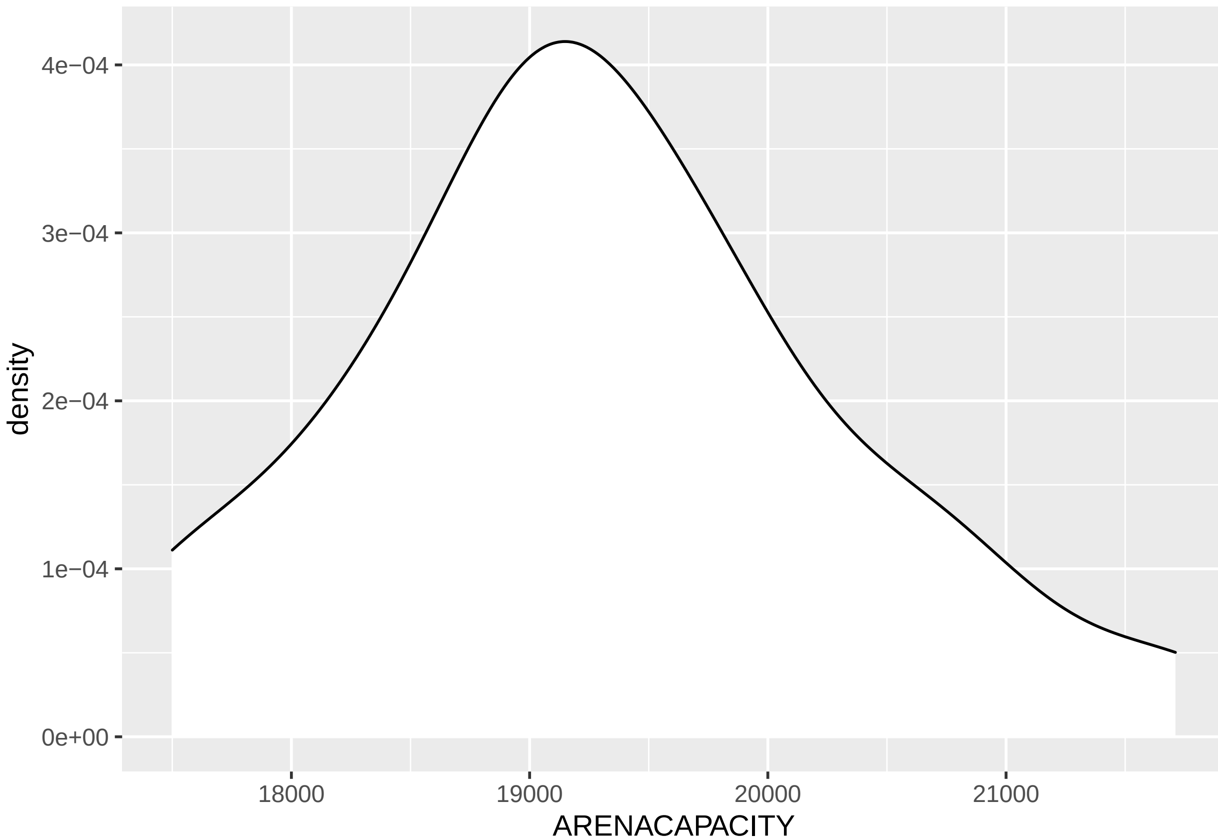
We played around with arena information and learned about missing values and adding new variables.

```
teams %>% select(NICKNAME, ARENA, ARENACAPACITY) %>% mutate(mean = mean(ARENACAPACITY, na.rm = TRUE))
```

```
##          NICKNAME                      ARENA ARENACAPACITY     mean
## 1           Hawks           State Farm Arena         18729 18553.31
## 2         Celtics                  TD Garden         18624 18553.31
## 3        Pelicans        Smoothie King Center           NA 18553.31
## 4           Bulls              United Center         21711 18553.31
## 5       Mavericks    American Airlines Center        19200 18553.31
## 6         Nuggets               Pepsi Center         19099 18553.31
## 7         Rockets              Toyota Center         18104 18553.31
## 8        Clippers             Staples Center         19060 18553.31
## 9          Lakers             Staples Center         19060 18553.31
## 10           Heat      AmericanAirlines Arena        19600 18553.31
## 11          Bucks               Fiserv Forum         17500 18553.31
## 12   Timberwolves              Target Center         19356 18553.31
## 13           Nets            Barclays Center            NA 18553.31
## 14         Knicks       Madison Square Garden        19763 18553.31
## 15          Magic               Amway Center             0 18553.31
## 16         Pacers     Bankers Life Fieldhouse        18345 18553.31
## 17          76ers          Wells Fargo Center           NA 18553.31
## 18           Suns Talking Stick Resort Arena            NA 18553.31
## 19 Trail Blazers                Moda Center          19980 18553.31
## 20          Kings           Golden 1 Center          17500 18553.31
## 21          Spurs               AT&T Center          18694 18553.31
## 22        Thunder     Chesapeake Energy Arena        19163 18553.31
## 23        Raptors          Scotiabank Arena          19800 18553.31
## 24           Jazz     Vivint Smart Home Arena        20148 18553.31
## 25      Grizzlies                 FedExForum         18119 18553.31
## 26        Wizards          Capital One Arena         20647 18553.31
## 27        Pistons        Little Caesars Arena        21000 18553.31
## 28        Hornets            Spectrum Center         19026 18553.31
## 29      Cavaliers         Quicken Loans Arena        20562 18553.31
## 30       Warriors              Chase Center          19596 18553.31
```

```
teams %>% filter(ARENACAPACITY != 0) %>%
  ggplot(aes(x=ARENACAPACITY)) +
  geom_density(color="black", fill="white", na.rm = TRUE, binwidth=1000)
```

```
## Warning: Ignoring unknown parameters: binwidth
```




Plot arena size

```
ranking <- ranking %>% filter(month == "03") %>% group_by(TEAM_ID) %>%  mutate(mean_w = mean(W, na.rm =
# ranking %>% sort(mean_w, decreasing = TRUE)

combined_games_details <- merge(games_details,games,by="GAME_ID")
```

Now we can look at the mean difference in home vs away points.

```
mean_home <- mean(games$PTS_home, na.rm = TRUE)
mean_away <- mean(games$PTS_away, na.rm = TRUE)
mean_home - mean_away
```

```
## [1] 2.811924
```

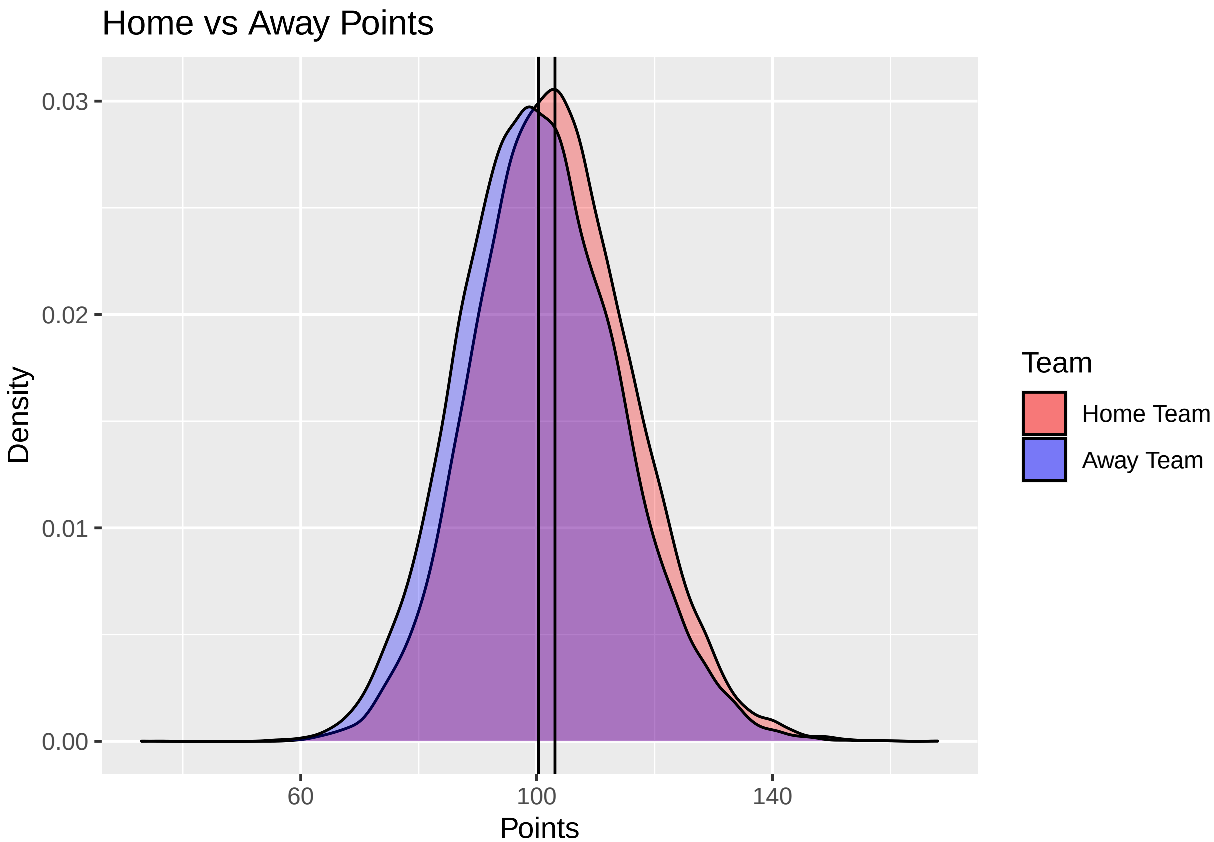
We clearly see above that the home team does marginally better. This is what many of us might expect. But the result is not that large. Do we know if this could be just by chance? In other words, what is the probability that the mean difference is instead 0?

In Statistics we have a way of testing this called a t-test. We'll come back to this.

```
ggplot() +
  geom_density(data = games, aes(x = PTS_home, fill = "home"), alpha = 0.3) +
  geom_density(data = games, aes(x = PTS_away, fill = "away"), alpha = 0.3) +
  scale_colour_manual(name ="Team", values = c("home" = "red", "away" = "blue"), labels=c("home" = "Home
  scale_fill_manual(name ="Team", values = c("home" = "red", "away" = "blue"), labels=c("home" = "Home '
  geom_vline(xintercept=mean_home) + geom_vline(xintercept=mean_away) + labs(title="Home vs Away Points
        x ="Points", y = "Density")
```

```
## Warning: Removed 99 rows containing non-finite values (stat_density).
```

```
## Removed 99 rows containing non-finite values (stat_density).
```




We started looking at season stats for players.

```
# games_details %>% group_by(PLAYER_ID) %>% summarise(mean_plus_minus = mean(PLUS_MINUS, na.rm = TRUE))

games_details %>% group_by(PLAYER_ID, PLAYER_NAME) %>% summarise(mean_plus_minus = mean(PLUS_MINUS, na.
```

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

```
## # A tibble: 2,580 x 4
## # Groups:   PLAYER_ID [2,575]
##    PLAYER_ID PLAYER_NAME     mean_plus_minus games
##        <int> <chr>                     <dbl> <int>
##  1    200970 Renaldo Major             15        1
##  2   1628996 Sagaba Konate             12        1
##  3   1628431 V.J. Beachem               9        1
##  4   1629236 Jonathan Stark             9        1
##  5   1627776 Isaiah Miles               8        1
##  6   1629746 Christ Koumadje            8        1
##  7   1629166 Jeff Roberson              7.5      2
##  8      1721 Keon Clark                 7        2
##  9    201939 Stephen Curry              6.56   980
## 10    203560 E.J. Singler               6.33     3
## # ... with 2,570 more rows
```

```
games_details %>% group_by(PLAYER_ID, PLAYER_NAME) %>% select(PLAYER_ID, PLAYER_NAME, PLUS_MINUS) %>% mu
```

```
## # A tibble: 719 x 4
## # Groups:   PLAYER_ID, PLAYER_NAME [719]
##    PLAYER_ID PLAYER_NAME     mean_plus_minus num_games
##        <int> <chr>                     <dbl>     <int>
##  1      2544 LeBron James              5.16       1651
##  2      2738 Andre Iguodala            2.30       1423
##  3      2594 Kyle Korver               2.25       1409
##  4      2730 Dwight Howard             2.11       1382
##  5      2546 Carmelo Anthony           0.848      1368
##  6      1713 Vince Carter              0.889      1329
##  7    101108 Chris Paul                4.57       1309
##  8      2225 Tony Parker               4.27       1303
##  9      1717 Dirk Nowitzki             3.78       1280
## 10      2548 Dwyane Wade               2.53       1260
## # ... with 709 more rows
```

```
games_details %>% group_by(PLAYER_ID, PLAYER_NAME) %>% summarise(mean_plus_minus = mean(PLUS_MINUS, na.:
```

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

```
## # A tibble: 576 x 4
## # Groups:   PLAYER_ID [576]
##    PLAYER_ID PLAYER_NAME    mean_plus_minus num_games
##        <int> <chr>                    <dbl>     <int>
##  1    201939 Stephen Curry             6.56       980
##  2    202695 Kawhi Leonard             6.03       733
##  3    203110 Draymond Green            5.86       832
##  4    202691 Klay Thompson             5.80       788
##  5      1495 Tim Duncan                5.77      1128
##  6    203954 Joel Embiid               5.52       366
##  7      2544 LeBron James              5.16      1651
##  8      1938 Manu Ginobili             5.09      1199
##  9       959 Steve Nash                4.64       822
## 10   1628369 Jayson Tatum              4.62       426
## # ... with 566 more rows
```

## Chef Curry

Here we played around with merging data from different dataframes.

```
players = games_details %>% select(-c("TEAM_ID", "TEAM_CITY", "PLAYER_ID", "COMMENT"))

games_date = games[,c("GAME_DATE_EST", "GAME_ID", "SEASON")]

# stats = steph.merge(games_date, on="GAME_ID", how="left")

# stats <- steph %>% left_join(games_date, by = c("GAME_ID"))

stats <- left_join(players, games_date, by = c("GAME_ID"))


head(stats)
```

```
##    GAME_ID TEAM_ABBREVIATION        PLAYER_NAME     NICKNAME START_POSITION
```

```
## 1 22101005               MIN    Anthony Edwards      Anthony              F
## 2 22101005               MIN    Jaden McDaniels        Jaden              F
## 3 22101005               MIN Karl-Anthony Towns Karl-Anthony              C
## 4 22101005               MIN      Malik Beasley        Malik              G
## 5 22101005               MIN   D'Angelo Russell     D'Angelo              G
## 6 22101005               MIN           Naz Reid          Naz
##     MIN FGM FGA FG_PCT FG3M FG3A FG3_PCT FTM FTA FT_PCT OREB DREB REB AST STL
## 1 36:22   4  10  0.400    3    8   0.375   4   4   1.00    0    8   8   5   3
## 2 23:54   6   8  0.750    1    3   0.333   1   1   1.00    2    4   6   0   0
## 3 25:17   4   9  0.444    1    3   0.333   6   8   0.75    1    9  10   0   0
## 4 30:52   4   9  0.444    4    9   0.444   0   0   0.00    0    3   3   1   1
## 5 33:46   3  13  0.231    1    6   0.167   7   7   1.00    0    6   6   9   1
## 6 23:56   3   8  0.375    1    2   0.500   4   4   1.00    3    7  10   1   3
##   BLK TO PF PTS PLUS_MINUS GAME_DATE_EST SEASON
## 1   1  1  1  15          5    2022-03-12   2021
## 2   2  2  6  14         10    2022-03-12   2021
## 3   0  3  4  15         14    2022-03-12   2021
## 4   0  1  4  12         20    2022-03-12   2021
## 5   0  5  0  14         17    2022-03-12   2021
## 6   2  1  1  11         -7    2022-03-12   2021
```

We then learned about how to chain commands together in the tidyverse (or dplyr) syntax using pipes. We learned how to group datasets by variables of interest and perform computations within these groups.

```
seasonal_stats <- stats %>% group_by(SEASON, PLAYER_NAME) %>% summarise(PTS = mean(PTS, na.rm=TRUE)) %>%
```

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

```
seasonal_stats
```

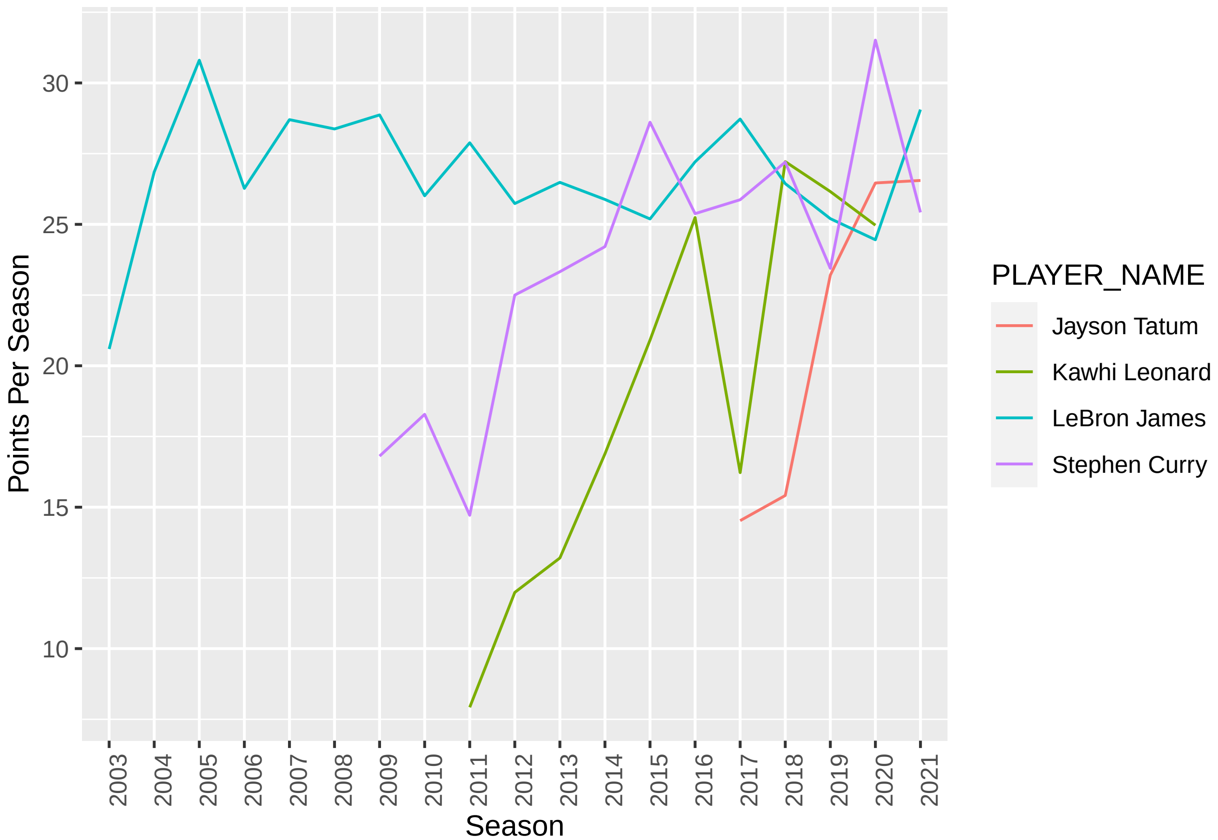
```
## # A tibble: 10,707 x 3
## # Groups:   SEASON [19]
##    SEASON PLAYER_NAME        PTS
##     <int> <chr>            <dbl>
##  1   2003 A.J. Guyton      4
##  2   2003 Aaron McKie      9.16
##  3   2003 Aaron Williams   5.89
##  4   2003 Ademola Okulaja  2
##  5   2003 Adonal Foyle     3.11
##  6   2003 Adrian Griffin   0.579
##  7   2003 Al Harrington   12.6
##  8   2003 Alan Henderson   4
##  9   2003 Alex Garcia      1.5
## 10   2003 Alex Scales     13.5
## # ... with 10,697 more rows
```

Last weekends homework.

```
select_players <- function (players) {
 seasonal_stats %>% filter(PLAYER_NAME %in% players) %>%
  ggplot(aes(x=factor(SEASON), y=PTS, group=PLAYER_NAME, fill=PLAYER_NAME, colour=PLAYER_NAME)) + geom_l
    theme( axis.text.x=element_text(angle=90) )
}
```

```
select_players(players=c("Stephen Curry", "LeBron James", "Jayson Tatum", "Kawhi Leonard"))
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

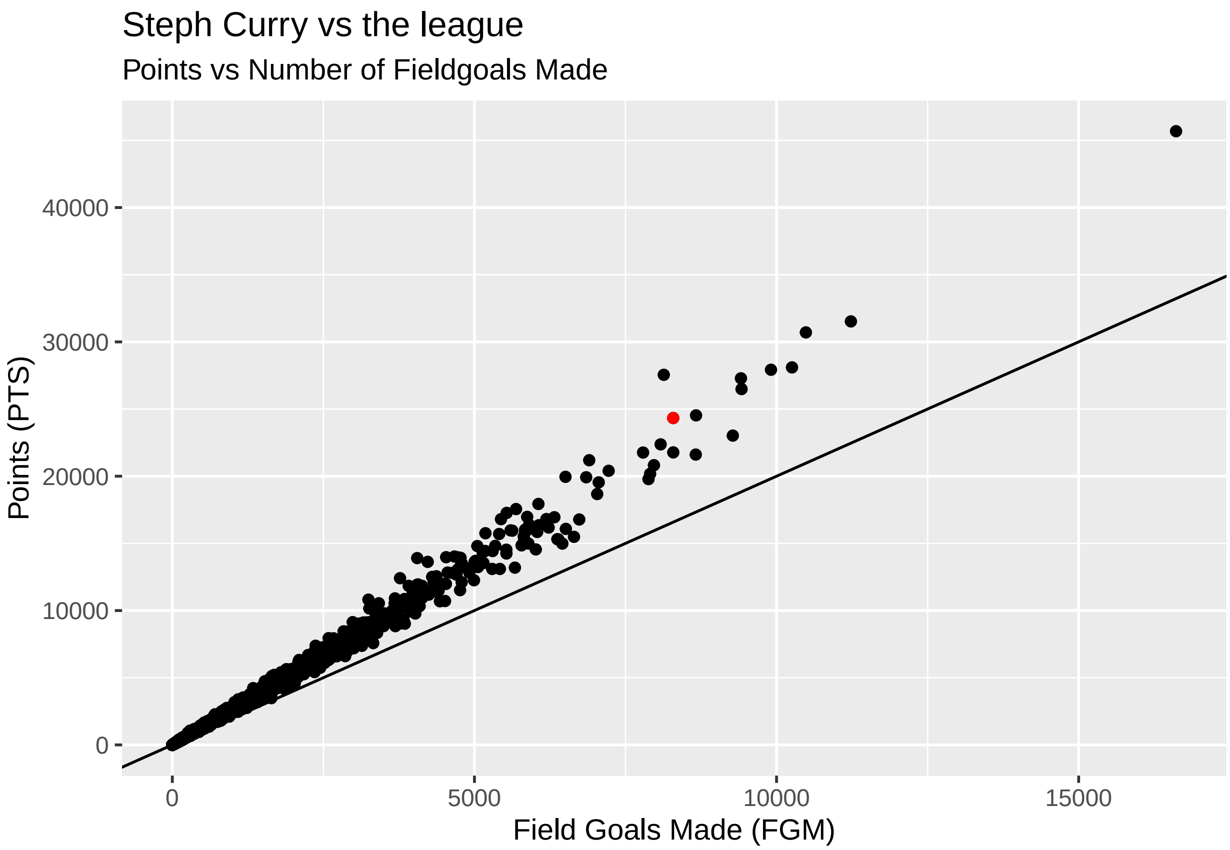



## Curry vs the rest of the league

Here we learned how to summarise (collapse the data) across multiple columns or variables at once. We then plotted performance for the rest of the team against a player of our choice.

```
all_players <- games_details %>% select(PLAYER_NAME, FGM, FG_PCT, FG3_PCT, PTS, FG3M, FG3A, FTM, FT_PCT

ggplot() + geom_point(data=all_players, aes(x=FGM, y=PTS)) + geom_point(data=all_players[all_players$PL
```


Steph Curry vs the league
Points vs Number of Fieldgoals Made
40000
30000
20000
10000
0
Points (PTS)
0
5000
10000
15000
Field Goals Made (FGM)