CWRU DSCI351-351m-451: Lab Exercise LE4 SOLUTION

EDA with Tidyverse vs. Base R

Prof.:Roger French, Prof. Paul Leu, TAs: Raymond Wieser, Sameera Nalin Venkat, Mingxuan Li

18 October, 2021

Contents

4.0.1 LE4, 10 points, 5 questions.

4.0.1.1 Lab Exercise (LE) 4 Tidyverse is a set of R packages that make our lives easier when handling unclean data. This lab exercise will highlight the advantages of using these functions as opposed to base R functions. Remember that your first step in this lab exercise is to library in the tidyverse package!

4.0.2 LE4-1. dplyr functions: (2.5 points)

4.0.2.1 LE4-1a (1/2 point) Define what these tidyverse functions do:

- select()
- filter()
- mutate()
- arrange()
- group_by()
- summarise()
  - funny: summarize() is the same function, for Amer. vs. British
- glimpse()
- tibble()

4.0.2.2 ANSWER (define functions listed above) ->

- select() : given a dataframe, it picks out the columns specified, and only displays them, -> i.e. allows us to pick the column of interest
- filter() : given a dataframe, it picks out the rows based on conditions, -> i.e. allows us to pick rows of interest
- mutate() : uses the information in a row to create another row.
- arrange() : arranges the data /rows according to a column's order. For eg: arranging the records by date.
- group_by() : groups the data using a similar values of a column, so that an operation can be done on it after.
- summarise() : on the grouped data, for the numerical columns, if there is a single value for. eg mean, average, max can be used to describe that group, it can be obtained using summarise. Summarise gives one value for a group, and therefore, gives a meaningful value of interest for a group.
  - funny: summarize() is the same function, for Amer. vs. British
- glimpse() : This is like a transposed version of print: columns run down the page, and data runs across. This makes it possible to see every column in a data frame. It's a little like str applied to a data frame but it tries to show you as much data as possible. (And it always shows the underlying data, even when applied to a remote data source.)
- tibble() : constructs a data frame. Character vectors are not coerced to factor.List-columns are expressly anticipated and do not require special tricks. Column names are not modified.Inner names in columns are left unchanged.

4.0.2.3 LE4-1b (1 point)

Show an example of each tidyverse function used on the Palmer Penguins dataset:

```
library(ggplot2)

library(tidyverse)
```

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

## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     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()
```

```
library(palmerpenguins)
library(hrbrthemes)
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.

##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and

##       if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```
data(penguins)

class(penguins) # already in the right format
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
## deplyr functions subscript
# select, filter, mutate

sel_dt_dp <- penguins %>%
  select(species, body_mass_g, sex)

fil_dt_dp <- penguins %>%
  filter(species == "Adelie")

mut_dt_dp <- penguins %>%
  mutate(bill_sum = bill_length_mm + bill_depth_mm)

# arrange, groupby, summarise

arr_dt_dp <- penguins %>%
  arrange(flipper_length_mm)

group_dt_dp <- penguins %>%
  group_by(sex, species)
## even though we have grouped all sex and species together, unless we do an
## operation on this we cannot observe the grouping

summ_dt_dp <- penguins %>%
  group_by(species) %>%
  summarise( mean(flipper_length_mm))

# glimpse and tibble

glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
```

```
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex               <fct> male, female, female, NA, female, male, female, male~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

```
tibb_dt_dp <- tibble(x = 1:5, y = 1)
```

4.0.2.4 LE4-1c (1 point) Execute the same examples that you created in 4-1b using base R code:

```
## base R functions : subscript r
# select, filter, mutate substitutes in r

sel_dt_r <- penguins[, c("species", "body_mass_g", "sex")]
fil_dt_r <- subset(penguins, species == "Adelie")
mut_dt_r <- penguins
mut_dt_r$bill_sum <- mut_dt_r$bill_length_mm + mut_dt_r$bill_depth_mm


# arrange, groupby, summarise substitutes in r
arr_dt_r <- penguins[order(penguins$flipper_length_mm),] ## always address a
#column using $
group_dt_r <- penguins[order(penguins$sex, penguins$species), ]
summ_dt_r <-aggregate(penguins$flipper_length_mm, list(penguins$species),
                      FUN=mean)

# glimpse and tibble  substitutes in r
tibb_dt_r <- data.frame(
   x = c(1:5),
   y = 1
)

structure(penguins)
```

```
## # A tibble: 344 x 8
##    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
##  1 Adelie  Torgersen           39.1          18.7               181        3750
##  2 Adelie  Torgersen           39.5          17.4               186        3800
##  3 Adelie  Torgersen           40.3          18                 195        3250
##  4 Adelie  Torgersen           NA            NA                  NA          NA
##  5 Adelie  Torgersen           36.7          19.3               193        3450
##  6 Adelie  Torgersen           39.3          20.6               190        3650
##  7 Adelie  Torgersen           38.9          17.8               181        3625
##  8 Adelie  Torgersen           39.2          19.6               195        4675
##  9 Adelie  Torgersen           34.1          18.1               193        3475
## 10 Adelie  Torgersen           42            20.2               190        4250
## # ... with 334 more rows, and 2 more variables: sex <fct>, year <int>
```

```
## unused
# summ_dt_r <- as.data.frame(t(sapply(X = split(x = penguins[which(penguins$species %in% c("Adelie","Ch
#     f = penguins$species[which(penguins$species %in% c("Adelie","Chinstrap","Gentoo"))],
#     drop = TRUE),
#   FUN = function(x) {apply(x, 2, mean)})))
```

4.0.3 LE4-2. Tuberculosis in different countries (3 points)

Tidyverse contains several tables containing information about

- TB cases in various countries,
- in different data formats.

We will use tidyverse and base R to manipulate these tables.

Parts a, b, and c will all require two versions of your answers

- one using base R, and
- one using tidyverse.

Also it will be useful

- to review the topic of `Joins` in R4DS

In table2,

- each row represents a (country, year, variable) combination.
- The column count contains
  - the values of variables cases and population
  - in separate rows.

Since table2 is a tibble

- a modified, or "enhanced" form of a dataframe
- its best to use the `glimpse` or `print` command on it
- And you can limit the number of rows that `print` will show
  - to n = 10, for example
  - with `print(table2, n = 10)`

```
data(table2)
glimpse(table2)
```

```
## Rows: 12
## Columns: 4
## $ country <chr> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", "B~
## $ year    <int> 1999, 1999, 2000, 2000, 1999, 1999, 2000, 2000, 1999, 1999, 20~
## $ type    <chr> "cases", "population", "cases", "population", "cases", "popula~
## $ count   <int> 745, 19987071, 2666, 20595360, 37737, 172006362, 80488, 174504~
```

Table 4 is split into two tables,

- one table for each variable.

The table table4a contains

- the values of cases and

Table4b contains

- the values of population.

Within each table,

- each row represents a country,
- each column represents a year,
- and the cells are the value of the table's variable
  - for that country and year.

```
data(table4a, table4b)
glimpse(table4a)
```

```
## Rows: 3
## Columns: 3
## $ country <chr> "Afghanistan", "Brazil", "China"
## $ `1999`  <int> 745, 37737, 212258
## $ `2000`  <int> 2666, 80488, 213766
```

```
glimpse(table4b)
```

```
## Rows: 3
## Columns: 3
## $ country <chr> "Afghanistan", "Brazil", "China"
## $ `1999`  <int> 19987071, 172006362, 1272915272
## $ `2000`  <int> 20595360, 174504898, 1280428583
```

4.0.3.1 LE4-2a (1 point) Compute the rate for table2, and table4a + table4b.

You will need to perform four operations:

1. Extract the number of TB cases per country per year.
2. Extract the matching population per country per year.
3. Divide cases by population, and multiply by 10000.
4. Store back in the appropriate place.

```
## use the cases from table 4a for cases

cntry <- c("Afghanistan","Brazil","China")
year <- c("1999", "2000")

## if we use table 4a and table 4b  the output should bein table 4c
## format :
## country : 1999 : 2000
## Afghanistan : rate1 : rate2
## 2 more rows


cases_1999 <- c(1:3)
cases_2000 <- c(1:3)
pop_1999  <- c(1:3)
pop_2000 <- c(1:3)

### find this and use it as column
rate_1999 <- c(1:3)
rate_2000 <- c(1:3)

for (i in 1:3){
cases_1999[i] <- table4a %>%
  filter(country == cntry[i])%>%
  select ("1999")
}

for (i in 1:3){
cases_2000[i] <- table4a %>%
  filter(country == cntry[i])%>%
  select ("2000")
}
```

```r
for (i in 1:3){
pop_1999[i] <- table4b %>%
  filter(country == cntry[i])%>%
  select ("1999")
}

for (i in 1:3){
pop_2000[i] <- table4b %>%
  filter(country == cntry[i])%>%
  select ("2000")
}

for (i in 1:3){
  rate_1999[i] = cases_1999[[i]]/pop_1999[[i]] * 10000
  rate_2000[i] = cases_2000[[i]]/pop_2000[[i]] * 10000
}

table4c <- data.frame(country = c("Afghanistan", "Brazil", "China"),
                      rate_1999 = rate_1999,
                      rate_2000 = rate_2000
                     )
##### rate from table 4 - > table 4c
head(table4c)
```

```
##       country rate_1999 rate_2000
## 1 Afghanistan  0.372741  1.294466
## 2      Brazil  2.193930  4.612363
## 3       China  1.667495  1.669488
```

```r
## table2

case_pop <- table2 %>%
  filter(country == "Afghanistan") %>%
  select(count)

rate_1999_an <- (case_pop$count[1]/case_pop$count[2] ) * 10000
rate_2000_an <- (case_pop$count[3]/case_pop$count[4] ) * 10000

case_pop <- table2 %>%
  filter(country == "Brazil") %>%
  select(count)

rate_1999_br <- (case_pop$count[1]/case_pop$count[2] ) * 10000
rate_2000_br <- (case_pop$count[3]/case_pop$count[4] ) * 10000

case_pop <- table2 %>%
  filter(country == "China") %>%
  select(count)

rate_1999_ch <- (case_pop$count[1]/case_pop$count[2] ) * 10000
rate_2000_ch <- (case_pop$count[3]/case_pop$count[4] ) * 10000

rate <- rbind(rate_1999_an, rate_2000_an, rate_1999_br, rate_2000_br,
              rate_1999_ch, rate_2000_ch)
```

```r
rate <- as.data.frame(rate)


table2new <- data.frame(country = c("Afghanistan", "Afghanistan",
                                    "Brazil", "Brazil",
                                    "China", "China"),
                        year = c(1999, 2000),
                        type = "rate",
                        count = rate$V1)

## rate from table 2  -> table 2
table2 <- full_join(table2, table2new)
```

```
## Joining, by = c("country", "year", "type", "count")
```

```r
tail(table2)
```

```
## # A tibble: 6 x 4
##   country      year type  count
##   <chr>       <dbl> <chr> <dbl>
## 1 Afghanistan  1999 rate  0.373
## 2 Afghanistan  2000 rate  1.29
## 3 Brazil       1999 rate  2.19
## 4 Brazil       2000 rate  4.61
## 5 China        1999 rate  1.67
## 6 China        2000 rate  1.67
```

4.0.3.2 LE4-2b (1 point) Which representation is easiest to work with?

4.0.3.3 ANSWER -> Both of them were hard to work with. It was easier to work with the second one because it was the same dataframe

Which is hardest? Why?

4.0.3.4 ANSWER -> The first one was harder because it took me a while to even realise where the appropriate place was, which was a new table. Dealing with columns that are numerical is problematic, the select column name does not really reflect what we are selecting so its very confusing, to remember what we have picked.

4.0.3.5 LE4-2c (1 point) The difficulty of working with these representations is why we use tidy data.

Combine these tables into a single tidy dataframe:

```r
tdy_fr <-table2 %>% spread(type, count) %>%
  mutate(rate = cases/population *10000)

glimpse(tdy_fr)
```

```
## Rows: 6
## Columns: 5
## $ country    <chr> "Afghanistan", "Afghanistan", "Brazil", "Brazil", "China", ~
## $ year       <dbl> 1999, 2000, 1999, 2000, 1999, 2000
## $ cases      <dbl> 745, 2666, 37737, 80488, 212258, 213766
## $ population <dbl> 19987071, 20595360, 172006362, 174504898, 1272915272, 12804~
## $ rate       <dbl> 0.372741, 1.294466, 2.193930, 4.612363, 1.667495, 1.669488
```

4.0.4 LE4-3. Useful EDA Plots (1.5 points)

This is an example of a plot generated using geom_violin in ggplot.

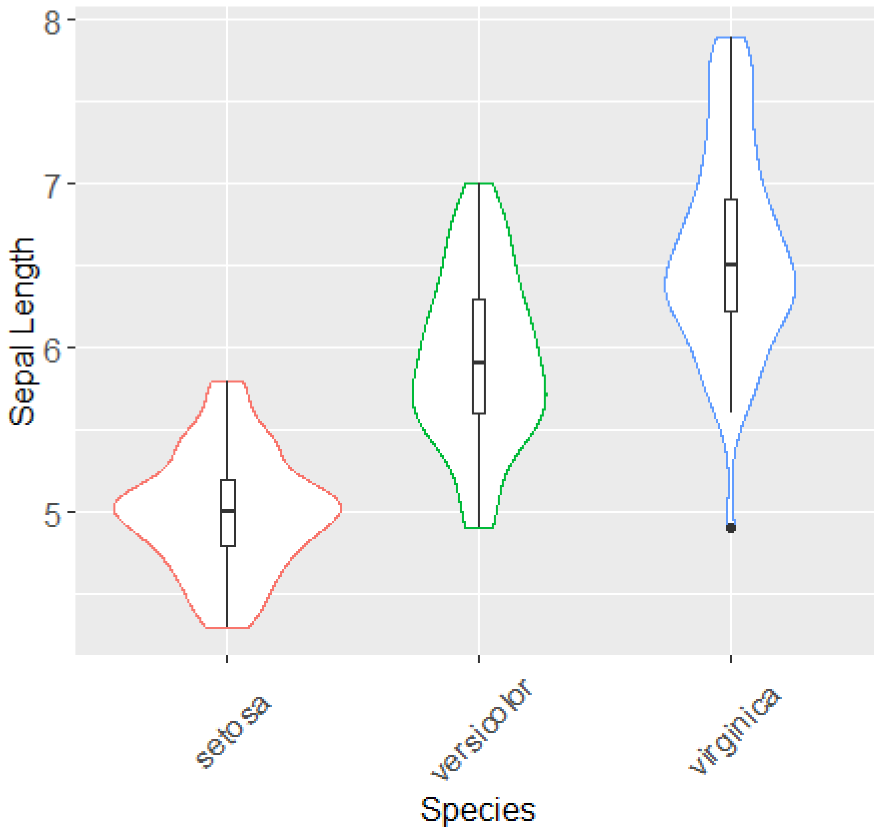


Figure 1: Violin plot generated by Raymond Wieser (SDLE) using iris dataset.

4.0.4.1 LE4-3a (3/4 point)

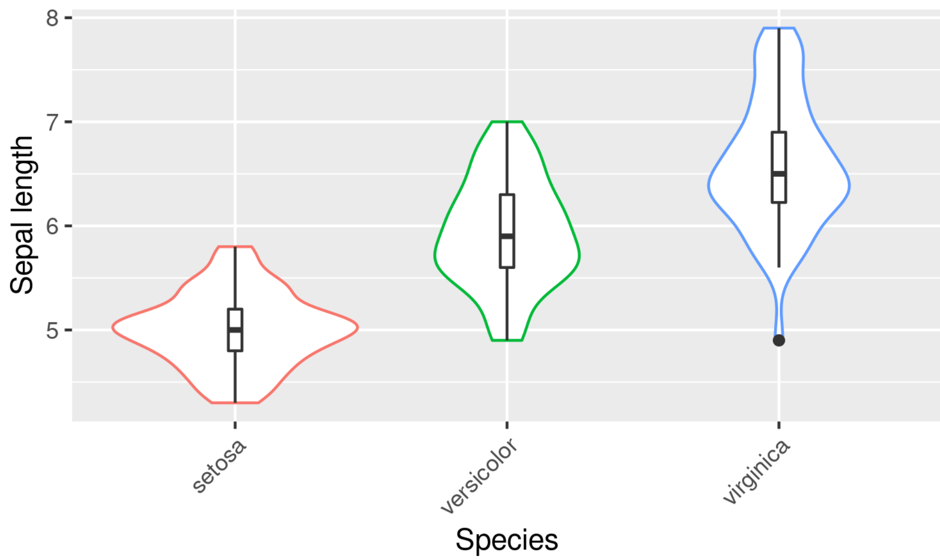
Recreate this plot using ggplot as best you can.

- It does not have to be identical,
  - but your plot should be professionally presentable and
  - contain all the relevant information.

You should use the theme() function in ggplot

- to set custom axis labels.

```
p <- ggplot(iris, aes(x = Species, y = Sepal.Length) ) +
  labs(y = "Sepal length")

p + geom_violin( aes(color = as.factor(Species)) , show.legend = FALSE)+
  geom_boxplot(width = 0.05)+ theme( ## use theme()  to set custom axis labels.
  axis.text.x = element_text( angle = 45, hjust = 1)
)
```




```
# +
## sepal length, and Species is 45 degrees
## should have box plot as well, and colored edges
## boxplot size should be pretty small - size  width =
```

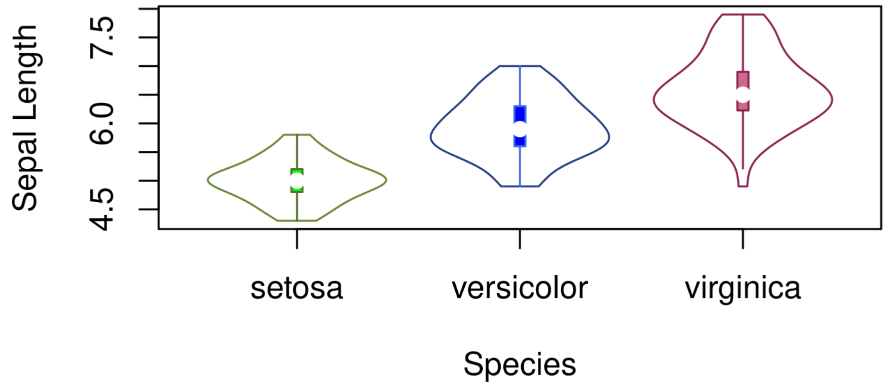
4.0.4.2 LE4-3b (3/4 point) Recreate the same plot using base R as best you can.

- It does not have to be identical,
- but your plot should be professionally presentable and
  - contain all the relevant information
- hint: checkout the package vioplot.

```
library(vioplot)
```

```
## Loading required package: sm

## Package 'sm', version 2.2-5.7: type help(sm) for summary information

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
vioplot(iris$Sepal.Length[iris$Species=="setosa"],
        iris$Sepal.Length[iris$Species=="versicolor"],
        iris$Sepal.Length[iris$Species=="virginica"],
        names=c("setosa", "versicolor", "virginica"),
        col=c("white", "white", "white"),
        rectCol=c("green", "blue", "palevioletred3"),
        lineCol=c("darkolivegreen", "royalblue", "violetred4"),
        border=c("darkolivegreen4", "royalblue4", "violetred4"),
        xlab="Species", ylab="Sepal Length")
```



4.0.5 LE4-4 EDA of sports salaries (2 points)

There is a dataset of basketball player's salaries from the 1984 through the 2017 season. The season is defined by what year it starts in.

We'll use tidyverse throughout this problem.

And lets use a ggplot theme for our plots,

- so they look like a newspaper or magazine of your choice.

ggplot2 has built in themes

ggthemes package has more interesting ones.

- You can read about ggplot themes in R4DS Ch 28.6
- but the more famous themes are in the ggthemes package.

```
library(tidyverse)
library(ggthemes)
```

4.0.5.1 Data assembly and check (1/2 point) So first read in the data, its in two .csv files for players and for salaries.

- The players file is like a key file to identify the players

You'll want to combine these into a single dataframe.

```
# read in data

file_names <- list.files(path = "./data" )

read_files <- file_names %>%
map(~read_csv(file.path('./data',.))) ### does this function internally convert it to a tibble ?
```

```
## Rows: 5100 Columns: 6

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Song, Artist, Lyrics
## dbl (3): Rank, Year, Source

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

## Rows: 4685 Columns: 24
```

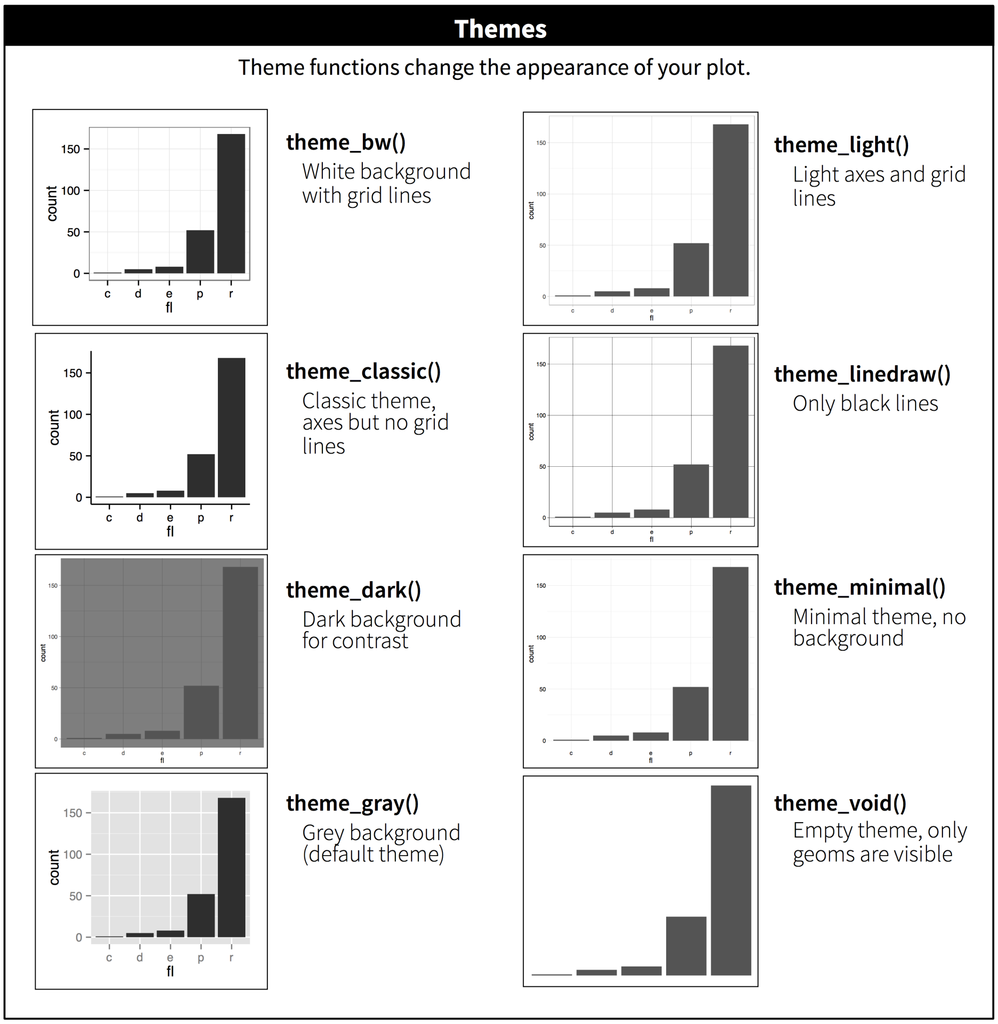


Figure 2: ggplot2 built-in themes

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (20): id, birthDate, birthPlace, career_FG%, career_FG3%, career_FT%, ca...
## dbl  (4): career_AST, career_G, career_PTS, career_WS

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

## Rows: 14163 Columns: 7

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): league, player_id, season, team
## dbl (3): salary, season_end, season_start

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
players <- as.data.frame(read_files[2]) %>%
  rename(player_id = id)

### rename the column id it might be causing problems.

salaries <- as.data.frame(read_files[3])

### combine players -"id" to salaries - "player_id"
### Use the join that has all the columns of both, but we want the information
### pertaining only to players whose salaries are available ?
### Not given in the question, ask TA ?

### since we are interested in the salaries of players, it is important to have
## all players that have a salaries,LEFT join with salaries
#### issue 1 :Error: Can't rename columns that don't exist.
###x Column `ID` doesn't exist.
#### solved 1: rename id to something else, maybe its a key word and messes up
#### operation

play_sal <- left_join(salaries, players, by =  "player_id")

## since the salaries and players have the same observations, that  means
### all the players in salaries were valid.

## look at the combined data
# glimpse(play_sal)
# row.has.na <- apply(play_sal, 1, function(x){any(is.na(x))})

## There are some values that have na, but they seem like Okay choices
## supposing if a player is in high school, he cannot have a college
```

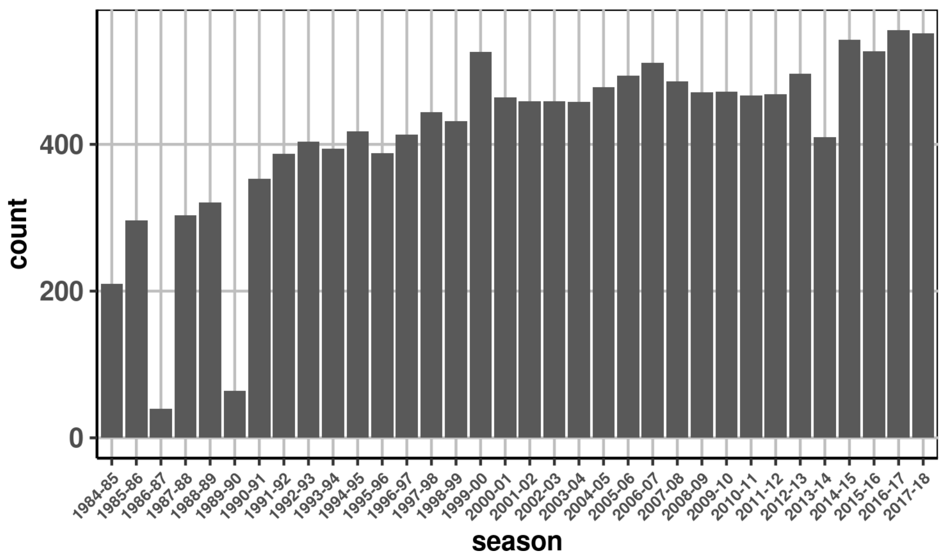
Since each observation is one player's salary for 1 year.

How many players salaries do we have for each year?

```r
# no. of records by season
## season - year, number of salaries
sal_by_year <- play_sal %>%
```

```
  group_by(season) %>%
  tally(n= "count")


### you can make another one to check if wht your histogram makes is correct
##### histogram can work with the original data
```

```
t <- theme(panel.background = element_rect(fill = "white", colour = "black"), panel.grid.major = element
        axis.text = element_text(size = 10, face = "bold"),
        axis.text.x = element_text(size = 6, face = "bold", angle = 45, hjust = 1),
        axis.title = element_text(size = 10, face = "bold"),
        strip.text = element_text(size = 12
                              , face = "bold"),
        text = element_text(size = 18), panel.spacing = unit(1, "lines"), legend.position = "top")
# plot it
ggplot(data = play_sal, aes(season)) +
  geom_histogram(stat = "count") +  theme_classic() +
  t
```



ANSWER (how many players salaries do we have for each year?)

What do you notice from the plot of players by season?

I can think of two relevant things to notice.

ANSWER (What do you notice from the plot of players by season?)

General trend : The number of players with salaries have gradually increased as years have passed by. Particular : There have been outliers in some years, and either the number has dipped drastically (1986 - 1987) & (1989 - 1990) from normal or increased (1999-2000)(2013-1014.

Now lets only consider the years from 2000 onward.

```
# we only look at salaries from 2000 onwards
# drop and rename some columns

year = "2000"
sal_from_2000 <- play_sal %>%
  filter(season_start >= year)
```

```
## distinct(sal_from_2000, season_start)
```

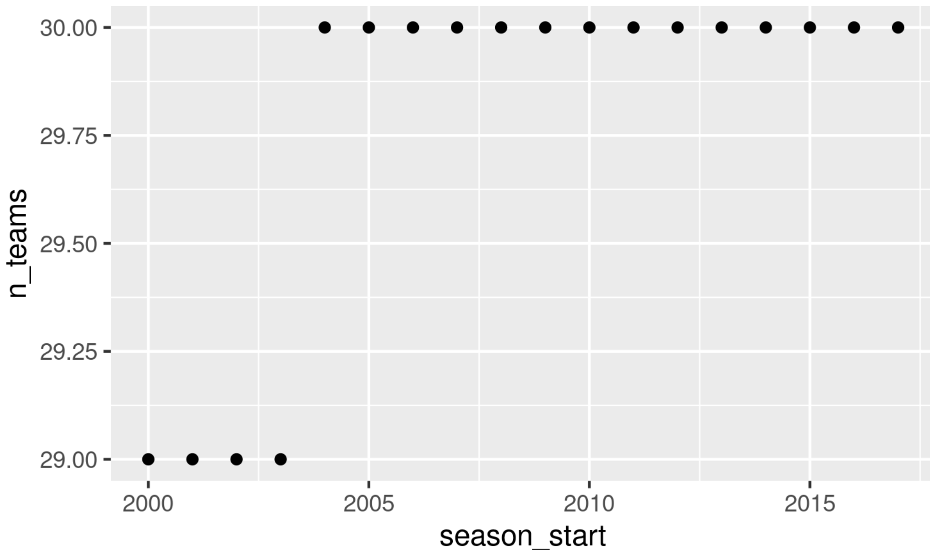
Next plot how many teams there are in each season since 2000.

```
# count no. of teams by year
distinct(sal_from_2000, team)
```

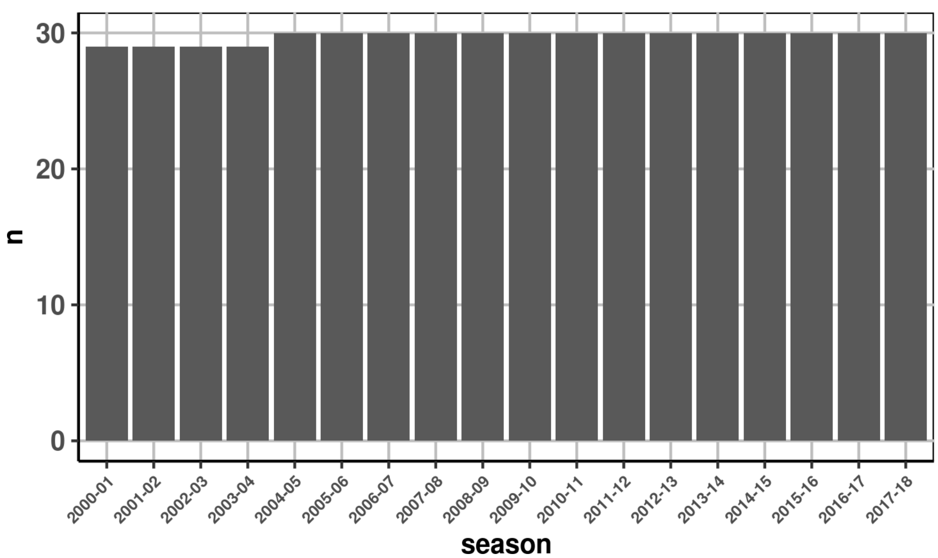
```
##                               team
## 1                Vancouver Grizzlies
## 2                     Denver Nuggets
## 3                   Dallas Mavericks
## 4                      Atlanta Hawks
## 5             Portland Trail Blazers
## 6                   Sacramento Kings
## 7              Oklahoma City Thunder
## 8                    Detroit Pistons
## 9               Los Angeles Clippers
## 10                   Toronto Raptors
## 11                   New York Knicks
## 12                     Brooklyn Nets
## 13                   New Jersey Nets
## 14                 Memphis Grizzlies
## 15                        Miami Heat
## 16             Golden State Warriors
## 17                   Houston Rockets
## 18                 Charlotte Bobcats
## 19                     Orlando Magic
## 20            Minnesota Timberwolves
## 21                 San Antonio Spurs
## 22              New Orleans Pelicans
## 23                Philadelphia 76ers
## 24                Washington Wizards
## 25               New Orleans Hornets
## 26                   Milwaukee Bucks
## 27                     Chicago Bulls
## 28                    Indiana Pacers
## 29               Seattle SuperSonics
## 30                    Boston Celtics
## 31                         Utah Jazz
## 32                      Phoenix Suns
## 33 New Orleans/Oklahoma City Hornets
## 34                 Charlotte Hornets
## 35               Cleveland Cavaliers
## 36                Los Angeles Lakers
```

```
sal_from_2000_teams <- sal_from_2000 %>%
  group_by(season_start)%>%
  summarise(n_teams = n_distinct(team))

# check <- sal_from_2000 %>%
#   group_by(season)%>%
#   summarise(n())

################################################################### how to check ?
```

```
ggplot(sal_from_2000_teams, aes(season_start,n_teams )) +
  geom_point()
```




```
sal_from_2000 %>%
     group_by(season) %>%
     summarise(n = n_distinct( team)) %>%
     ggplot(., aes(x = season, y=n)) +
            geom_bar(stat='identity') +  theme_classic() +
  t
```




```
#  "The heights of the bars commonly represent one of two things: either a count
#  of cases in each group, or the values in a column of the data frame.
# By default, geom_bar uses stat="bin". This makes the height of each bar
# equal to the number of cases in each group, and it is incompatible with
# mapping values to the y aesthetic. If you want the heights of the bars
# to represent values in the data, use stat="identity" and map a value to
# the y aesthetic."
```

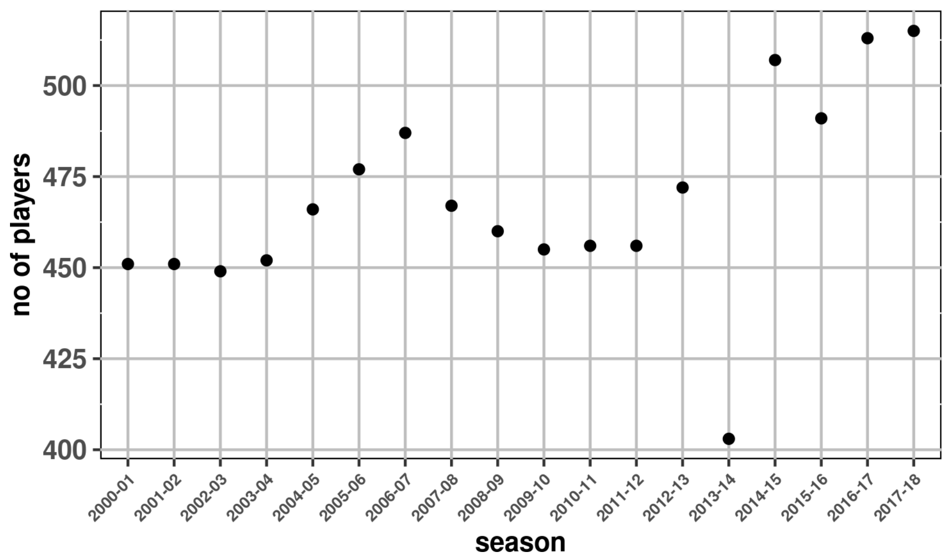
What do you notice from this EDA plot? Does this make sense?

ANSWER (What do you notice from this EDA plot and does it make sense?) -> Yes, according to the NBA expansion from wikipedia : 1995–2004 - 29 2004–present: 30 There have been 29 and 30 teams, from 2000 onwards.

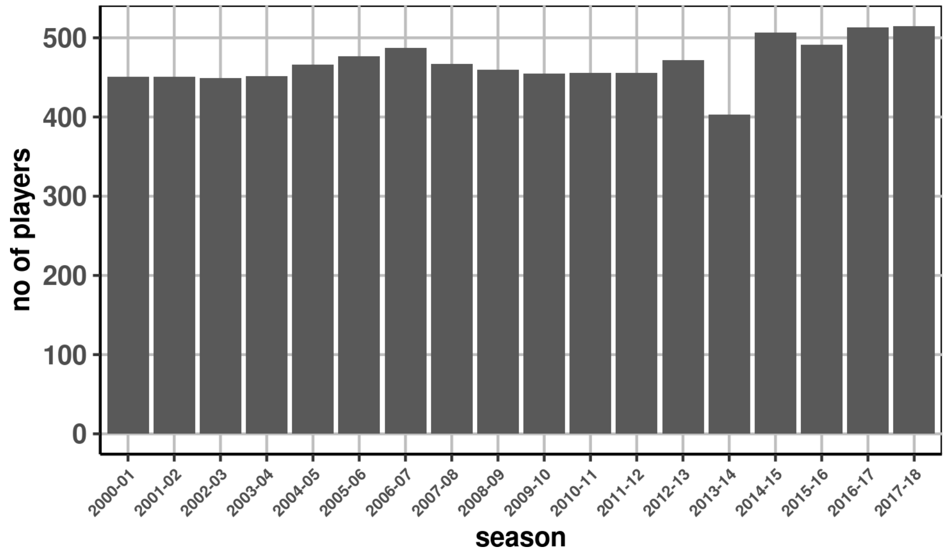
- NBA Expansion

```
# count no. of players by year

###
sal_from_2000_players <- sal_from_2000 %>%
  group_by(season)%>%
  summarise(n_players = n_distinct(player_id))



################################################################### how to check ?
```

```
l <-  labs(y= "no of players")

ggplot(sal_from_2000_players, aes(season, n_players )) +
  geom_point()+ t + l
```




```
sal_from_2000_players %>%
          ggplot(., aes(x = season, y = n_players)) +
            geom_bar(stat='identity' ) +  theme_classic() +
  t +l
```

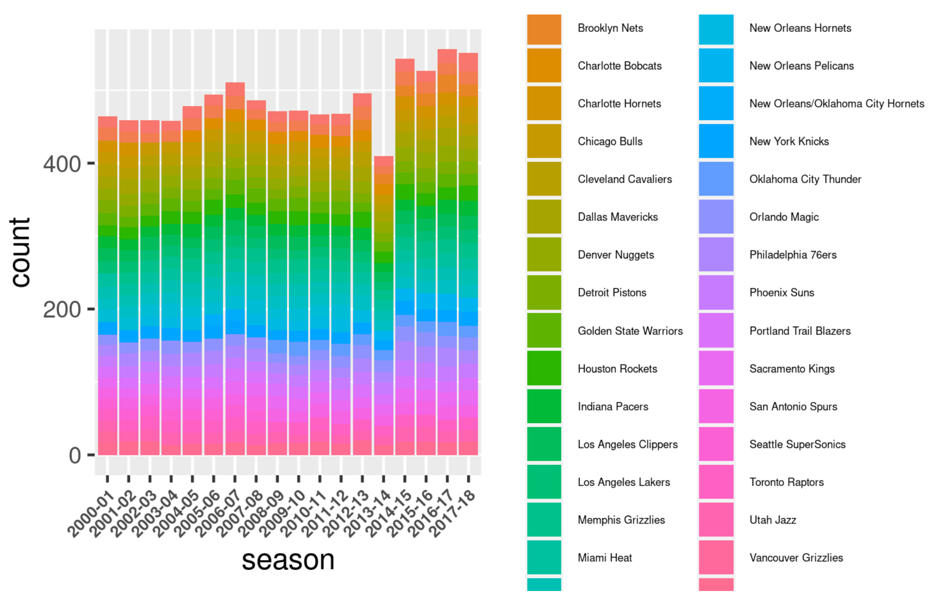


Can you make it a stacked histogram plot

- So that is shows players per team.
  - And sums the teams up
- To show the total players for each year on the y axis.

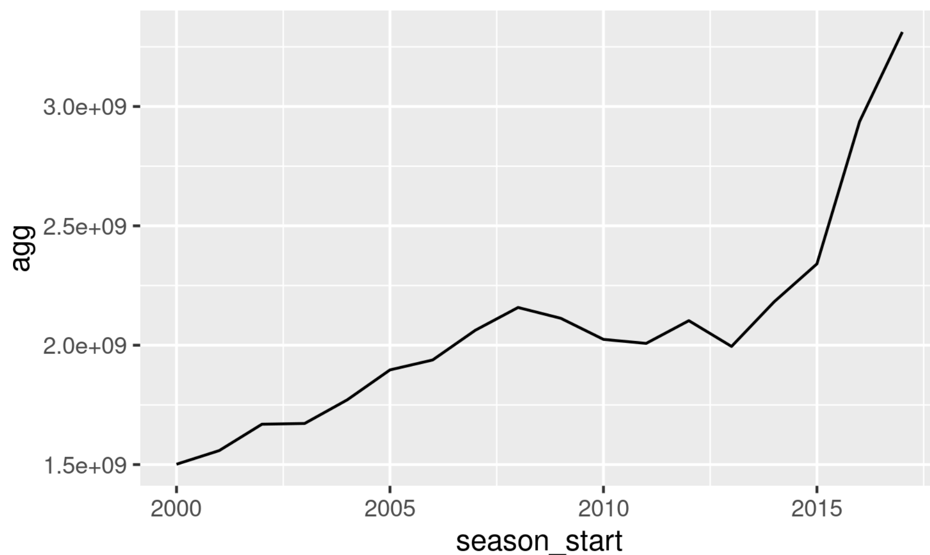
```
ggplot(data = sal_from_2000, aes(x = season, fill = team)) +
  geom_histogram(stat = "count", show.legend = TRUE)+
  theme(legend.text = element_text(size = 4) ) +
  theme(legend.key.size = unit(0.5, 'cm') ,
        axis.text.x = element_text(size = 6,
                                   face = "bold", angle = 45, hjust = 1))
```

4.0.5.2 Now lets look at the total salary by year. (1/2 point)




```
## hide the legend
## theme
```

```r
# total salary by year

# sal_from_2000 <- sal_by_year %>%
#   as.numeric(salaries)

sal_from_2000$salary <- as.numeric(sal_from_2000$salary)# Convert one variable to numeric



sal_from_2000_salaries <- sal_from_2000 %>%
  group_by(season_start) %>%
  summarise(agg = sum(salary))


### doesn't seem to work because salaries is a character.


# ggplot(data = sal_from_2000_salaries, aes(x = season_start )) + geom_histogram()

ggplot(sal_from_2000_salaries, aes(season_start,agg )) +
  geom_line()
```



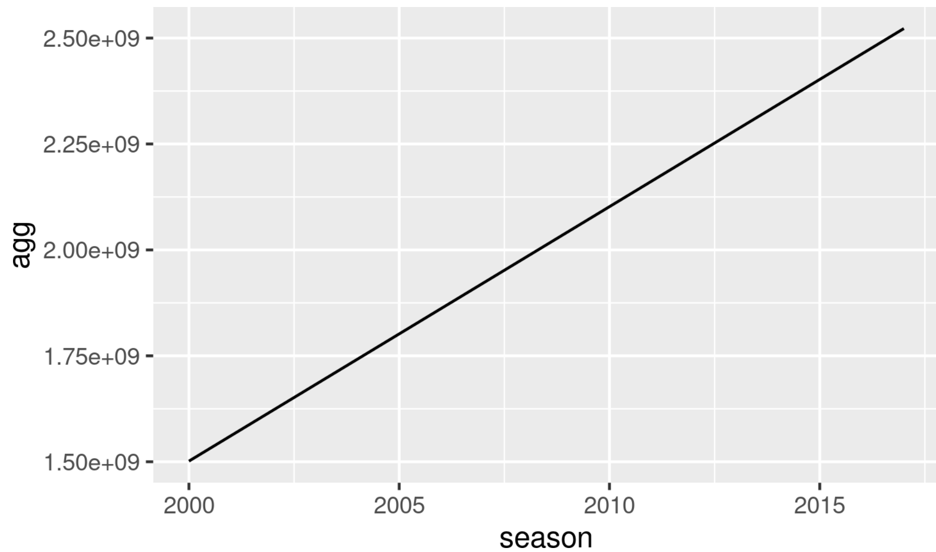
So the salary pool is growing each year.

- Is this as expected? - For example is it just the effect of inflation - Typically inflation is 4% a year (or maybe 2%)
- Or is something else going on?

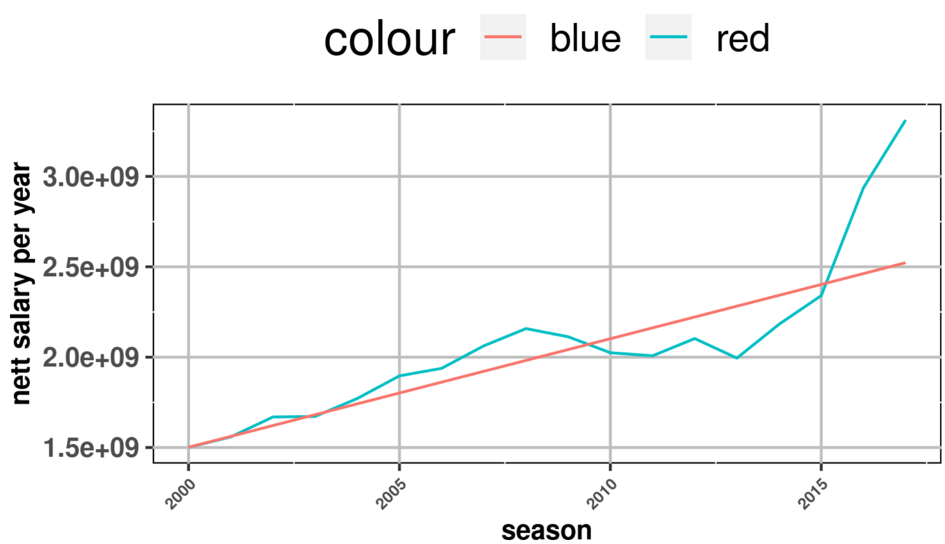
ANSWER (Is this as expected or is there something else going on?) -> It is comparable from 2000-2003, but then it increases slightly until 2010, where salaries reduce, and then after 2015, the salaries increase again.

```r
# compare with constant inflation

agg = c(1:18)
for (i in 1: 18){
  sd = 1501509015
  agg[i] = sd + sd *(i-1)*4/100
```

```
}

constant_inflation <- data.frame(season = c(2000 : 2017),
                                 agg = agg)

ggplot(constant_inflation, aes(season,agg )) +
  geom_line()
```




```
## let us join the columns
sal_comp <-  full_join(sal_from_2000_salaries, constant_inflation ,
                       by=c("season_start" ="season"))

ggplot(sal_comp, aes(x = season_start ), show.legend = FALSE) +
  geom_line(aes(y = agg.x, colour = "red")) +
  geom_line(aes(y = agg.y, colour = "blue")) +
  labs(x = "season", y = "nett salary per year")+ t
```



How is the salary pool of each team?

- Are they all comparable?

- Or are their large differences?

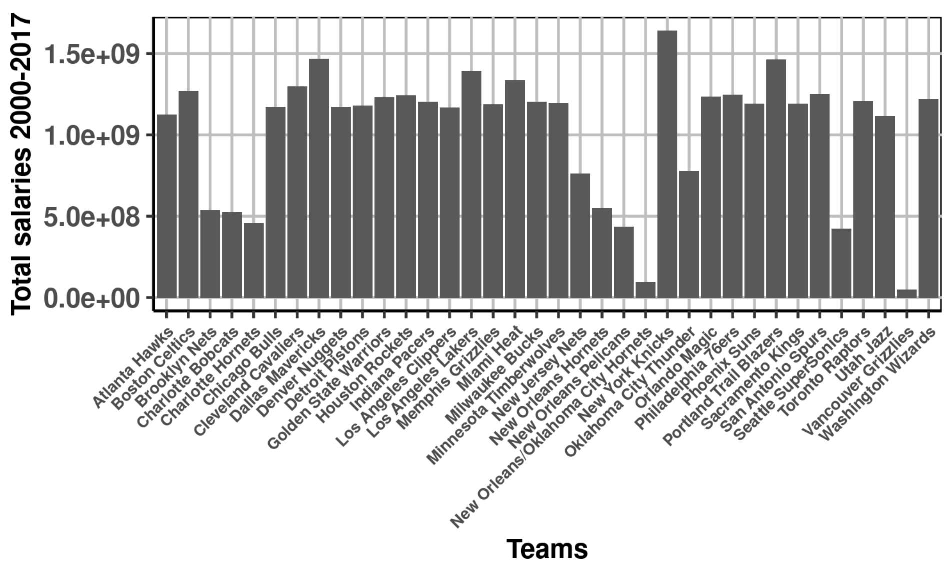
```r
# total salary by year by team

df2000_sal_team <- sal_from_2000 %>%
  group_by(season_start, team)%>%
  summarise(tot_sal = sum(salary)) %>%
  arrange(season_start, desc(tot_sal))
```

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

```r
# total salary by team
df2000_sal_team2 <- sal_from_2000 %>%
  group_by( team)%>%
  summarise(tot_sal = sum(salary)) %>%
  arrange(desc(tot_sal))

summary(df2000_sal_team2)
```

```
##      team              tot_sal
##  Length:36          Min.   :4.890e+07
##  Class :character   1st Qu.:7.752e+08
##  Mode  :character   Median :1.194e+09
##                     Mean   :1.035e+09
##                     3rd Qu.:1.244e+09
##                     Max.   :1.643e+09
```

```r
df2000_sal_team2 %>%
          ggplot(., aes(x = team, y = tot_sal)) +
            geom_bar(stat='identity' ) +  theme_classic() +
  t + labs(x =  "Teams", y = "Total salaries 2000-2017")
```



ANSWER (How is the salary pool for each team?) ->

If we sum all the salaries/ per team after 2000:

Min : Vancouver Grizzlies: 4.890e+07 ; Max :
New York Knicks :1.643e+09 ; Median : 1.194e+09 ; Mean :1.035e+09 ;

Mean and Median are somewhat different.

4.0.5.3 Now lets compare among the teams (1 point)

- Do some teams always spend more than others?

To do this, lets rank teams by salary within each year.

- And the small ranks, paying out more salary in that year.
  - So team ranked 1 pays the most salary
  - And the team ranked 2nd pays less than them.

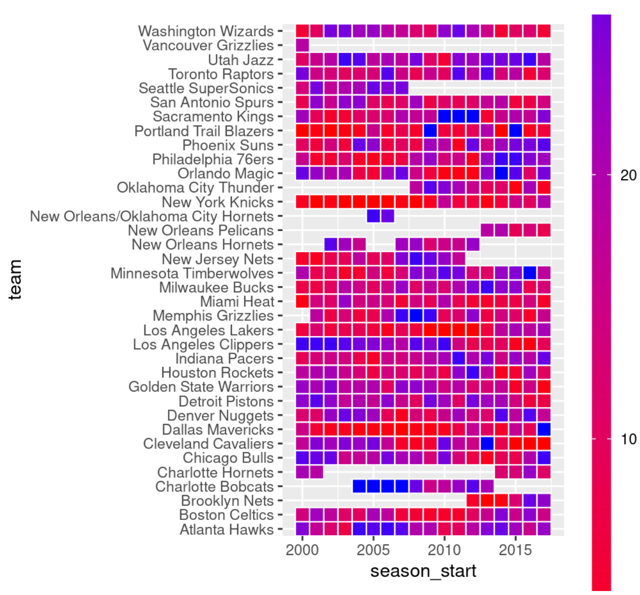
Do this with a heatmap, with season/year as the x-axis

- And the teams only the y-axis
- And a teams rank, is by color
  - Say from Red (rank 1) to Blue (lowest rank)
  - With Red as Rank 1 and Blue as the lowest ranked team salary pool

```
# team ranking comparison by total salary by year
df2000_sal_team <- sal_from_2000 %>%
  group_by(season_start, team)%>%
  summarise(tot_sal = sum(salary)) %>%
  arrange(season_start, desc(tot_sal)) %>%
  mutate(rank = dense_rank(desc(tot_sal)))
```

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

```
# Heatmap
p<- ggplot(df2000_sal_team, aes(season_start, team, fill = rank)) +
  geom_tile(colour="white", size = 0.25)  +
  scale_fill_gradient(low = "red", high = "blue")+ coord_fixed() +
  guides(fill = guide_colourbar(barwidth = 0.5,
                                barheight = 20))+
  scale_y_discrete(expand=c(0,0))+
  theme_grey(base_size = 7)

p
```




```
# ggsave(p,filename="heatmap-basic.png")
```

ANSWER (Do some teams always spend more than others?) -> NewYork Kicks, Los Angeles lakers, Dallas

Mavericks, Miami heat have consistly have had high salaries, while Utah Jazz have been more judicious about spending.

Now make a table of the top paid player for each year from 2000 onward.

```
# top paid player in each year

highest_sal_year <- sal_from_2000 %>%
  group_by(season_start) %>%
  top_n(salary, n = 1)%>%
  select (salary, season_start, player_id, team) %>%
  arrange(season_start)

head(highest_sal_year)
```

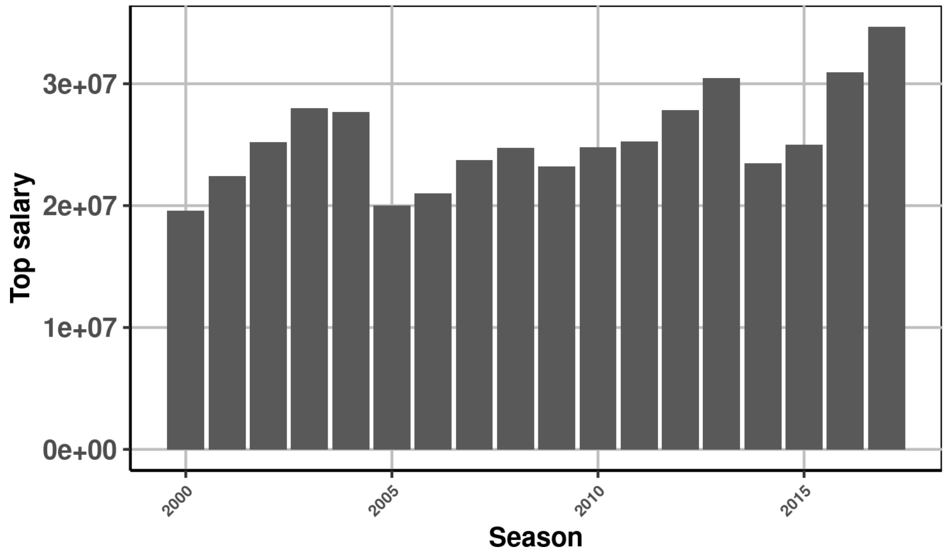
```
## # A tibble: 6 x 4
## # Groups:   season_start [6]
##     salary season_start player_id team
##      <dbl>        <dbl> <chr>     <chr>
## 1 19610000         2000 garneke01 Minnesota Timberwolves
## 2 22400000         2001 garneke01 Minnesota Timberwolves
## 3 25200000         2002 garneke01 Minnesota Timberwolves
## 4 28000000         2003 garneke01 Minnesota Timberwolves
## 5 27696430         2004 onealsh01 Miami Heat
## 6 20000000         2005 onealsh01 Miami Heat
```

```
highest_check <- sal_from_2000 %>%
  group_by(season_start) %>%
  summarise(max_sal = max(salary))


### join this with original
# new <- left_join(highest_sal_year, sal_from_2000,
#                               by = c("salary" = "max_sal")) %>%
#   select(name, salary, season_start)

# result <- sal_from_2000 %>%
#              group_by(season_start , name) %>%
#              filter(max == max(salary)) %>%
#              arrange(season_start,name, max)
#
# season_start
```

Now make a plot of the top salary in each year.

- Is this what you expected?

```
# Basic histogram
highest_sal_year %>%
          ggplot(., aes(x = season_start, y = salary)) +
            geom_bar(stat='identity' ) +  theme_classic() +
  t + labs(x =  "Season", y = "Top salary")
```



ANSWER (Is this what you expected?) -> 2015 -2017 Cleveland Caveliers and Golden State Warriors were spending a lot according to the heat maps, and therefor the highest salaries comes from them. 2010 - 2015 : Los Angeles Lakers were in red according to heat map, the highest salary is from them

2000 - 2003 : Minnesota Timberwolves were in red and salries reflect so.

I would say in general if the highest salry came from a team in a given year they were spending a lot on salaries in general (According to heat map)

4.0.6 LE4-5 Creating Extensible and Flexible Code (1 point)

- In LE2-3c-d, you created word clouds for Elton John and Eminem
- Let's modify that code so that it is more flexible and extensible.
  - The modified code will work with an arbitrary list of artists.

- The dataset for this assignment is a collection of
  - the information and lyrics from every top 100 billboard song since 1965

Write a function, named `GenerateWordCloud` that creates a wordcloud

- for each artist in an arbitrarily chosen list of artists
  - with max_wordcloud_words
- So this function needs to work for 1, 2, 3 or more artists
- The word cloud should not include any stop_words
  - as in LE#2
- Write your code below for the `GenerateWordCloud` function

```
# load in the dataset
library(tidytext)
library(tm)  # the Text Mining Package
```

4.0.6.1 LE4-a Define your `GenerateWordCloud` function

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
```

```
##
##     annotate
```

```
library(NLP) # the Natural Language Processing package
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(dplyr)
billboard_df <- read.csv('./data/billboard_lyrics_1964-2015.csv') %>%
  as.data.frame()


# Use VCorpus(VectorSource(word) in wordcloud to eliminate warnings
```

```
GenerateWordCloud <- function(artists, billboard_df, max_wordcloud_words) {
  # This function generates word clouds for each artist in the list artists
   # with the maximum number of words max_wordcloud_words
   #
   # Write your code here for the GenerateWordCloud function
   #

 tbl_df(billboard_df)
billboard_df$Lyrics <- as.character(billboard_df$Lyrics)
#
#
   artist_songs <- billboard_df %>%
  filter(Artist == artists) %>%
   select (Artist, Song, Lyrics)

   artist_songs$Lyrics <- as.character(artist_songs$Lyrics)
   artist_word <- artist_songs %>%
   unnest_tokens(output = word, input = Lyrics)

   # print(artist_word)

 # artist : use this info to get only the songs from that artist.


 wordcloud(words = artist_word$word, min.freq = 1,
          max_wordcloud_words = max_wordcloud_words, random.order=FALSE,
          rot.per=0.35, colors=brewer.pal(8, "Dark2"))

}
```

4.0.6.2 LE4-5b Now test your function On 2 artists

```r
max_wordcloud_words <- 30
artists <- c("elton john","eminem")
GenerateWordCloud(artists, billboard_df, max_wordcloud_words)
```

Test your function on 1 artist

```r
max_wordcloud_words <- 40
artists <- c("elton john")
GenerateWordCloud(artists, billboard_df, max_wordcloud_words)
```

And on 3 artists

```r
max_wordcloud_words <- 20
artists <- c("madonna", "elton john", "eminem")
GenerateWordCloud(artists, billboard_df, max_wordcloud_words)
```

4.0.7 Links

http://www.r-project.org

http://rmarkdown.rstudio.com/

Chris Davis: https://data.world/datadavis/nba-salaries