Homework 3_Suixin Jiang

Suixin Jiang

11/10/2019

Exercise 1

Q1: Import data

```
BCC <- read_csv(file = './BPD_Part_1_Victim_Based_Crime_Data.csv')
```

```
## Parsed with column specification:
## cols(
##   CrimeDate = col_character(),
##   CrimeTime = col_character(),
##   CrimeCode = col_character(),
##   Location = col_character(),
##   Description = col_character(),
##   `Inside/Outside` = col_character(),
##   Weapon = col_character(),
##   Post = col_double(),
##   District = col_character(),
##   Neighborhood = col_character(),
##   Longitude = col_double(),
##   Latitude = col_double(),
##   `Location 1` = col_character(),
##   Premise = col_character(),
##   crimeCaseNumber = col_logical(),
##   `Total Incidents` = col_double()
## )
```

Q2: Convert dates and times to date classes

```
BCC %>%
  mutate(CrimeDate = parse_date(CrimeDate, format = '%m/%d/%Y'),
         CrimeTime = parse_time(CrimeTime, format = '%H:%M:%S')) %>%
  subset(!is.na(CrimeTime)) ->
  BCC
```

```
## Warning: 5751 parsing failures.
## row col           expected actual
##  48  -- time like %H:%M:%S   0454
##  77  -- time like %H:%M:%S   2247
##  82  -- time like %H:%M:%S   2132
##  88  -- time like %H:%M:%S   2053
## 134  -- time like %H:%M:%S   1159
## ... ... .................. ......
## See problems(...) for more details.
```

Q3 Separate ‘Location 1’

```
BCC$`Location 1` <- gsub('[()]', '', BCC$`Location 1`)
BCC %>%
  separate('Location 1', into = c('LocationLat', 'LocationLon'), sep = ',') ->
  BCC
```

Q4 Determine the percent of crimes

Up to 14% of crimes happened during midnight to 4:00 am.

```
BCC %>%
  filter(CrimeTime >= 0 & CrimeTime <= 60*60*4) %>%
  count()/nrow(BCC)
```

```
##           n
## 1 0.1394631
```

Exercise 2

Q1

The ‘Baby names’ data set has four data frames – applicants, babynames, births, and lifetables.

The primary key for ‘babynames::applicants’ is (‘year’, ‘sex’).

The primary key for ‘babynames::babynames’ is (‘year’, ‘sex’, ‘name’).

The primary key for ‘babynames::births’ is (‘year’).

The primary key for ‘babynames::lifetables’ is (‘x’, ‘sex’, ‘year’).

```
library(babynames)
```

```
## Warning: package 'babynames' was built under R version 3.5.3
```

```
babynames::applicants %>%
  count(year, sex) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 0
```

```
babynames::babynames %>%
  count(year, sex, name) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 0
```

```
babynames::births %>%
  count(year) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 0
```

```
babynames::lifetables %>%
  count(x, sex, year) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 0
```

Q2

The ‘NASA weather’ data set has five data frames – atoms, borders, elev, glaciers, and storms.

The primary key for ‘nasaweather::atmos’ is (‘lat’, ‘long’, ‘year’, ‘month’).

‘nasaweather::borders’ does not have a primary key.

The primary key for ‘nasaweather::elel’ is (‘long’, ‘lat’, ‘elev’).

The primary key for ‘nasaweather::glaciers’ is (‘id’).

The primary key for ‘nasaweather::storms’ is (‘hour’, ‘lat’, ‘long’, ‘seasday’).

```
library(nasaweather)
```

```
##
## Attaching package: 'nasaweather'

## The following object is masked from 'package:dplyr':
##
##     storms
```

```
nasaweather::atmos %>%
  count(lat, long, year, month) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 0
```

```
nasaweather::elev %>%
  count(long, lat, elev) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 0
```

```
nasaweather::glaciers %>%
  count(id) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 0
```

```
nasaweather::storms %>%
  count(hour, lat, long, seasday) %>%
  filter(n > 1) %>%
  nrow()
```

```
## [1] 0
```

Exercise 3

Q1 Load data frames

```
library(Lahman)
```

```
## Warning: package 'Lahman' was built under R version 3.5.3
```

```
data("Batting")
names(Batting)
```

```
##  [1] "playerID" "yearID"   "stint"    "teamID"   "lgID"     "G"
##  [7] "AB"       "R"        "H"        "X2B"      "X3B"      "HR"
## [13] "RBI"      "SB"       "CS"       "BB"       "SO"       "IBB"
## [19] "HBP"      "SH"       "SF"       "GIDP"
```

```
data("Fielding")
names(Fielding)
```

```
##  [1] "playerID" "yearID"   "stint"    "teamID"   "lgID"     "POS"
##  [7] "G"        "GS"       "InnOuts"  "PO"       "A"        "E"
## [13] "DP"       "PB"       "WP"       "SB"       "CS"       "ZR"
```

```
data("Master")
names(Master)
```

```
##  [1] "playerID"     "birthYear"    "birthMonth"   "birthDay"
##  [5] "birthCountry" "birthState"   "birthCity"    "deathYear"
##  [9] "deathMonth"   "deathDay"     "deathCountry" "deathState"
## [13] "deathCity"    "nameFirst"    "nameLast"     "nameGiven"
## [17] "weight"       "height"       "bats"         "throws"
## [21] "debut"        "finalGame"    "retroID"      "bbrefID"
## [25] "deathDate"    "birthDate"
```

```
data("People")
names(People)
```

```
##  [1] "playerID"     "birthYear"    "birthMonth"   "birthDay"
##  [5] "birthCountry" "birthState"   "birthCity"    "deathYear"
##  [9] "deathMonth"   "deathDay"     "deathCountry" "deathState"
## [13] "deathCity"    "nameFirst"    "nameLast"     "nameGiven"
## [17] "weight"       "height"       "bats"         "throws"
## [21] "debut"        "finalGame"    "retroID"      "bbrefID"
## [25] "deathDate"    "birthDate"
```

```
data("Pitching")
names(Pitching)
```

```
##  [1] "playerID" "yearID"   "stint"    "teamID"   "lgID"     "W"
##  [7] "L"        "G"        "GS"       "CG"       "SHO"      "SV"
## [13] "IPouts"   "H"        "ER"       "HR"       "BB"       "SO"
## [19] "BAOpp"    "ERA"      "IBB"      "WP"       "HBP"      "BK"
## [25] "BFP"      "GF"       "R"        "SH"       "SF"       "GIDP"
```

```
data("Salaries")
names(Salaries)
```

```
## [1] "yearID"   "teamID"   "lgID"     "playerID" "salary"
```

```
data("Teams")
names(Teams)
```

```
##  [1] "yearID"         "lgID"           "teamID"         "franchID"      
##  [5] "divID"          "Rank"           "G"              "Ghome"         
##  [9] "W"              "L"              "DivWin"         "WCWin"         
## [13] "LgWin"          "WSWin"          "R"              "AB"            
## [17] "H"              "X2B"            "X3B"            "HR"            
## [21] "BB"             "SO"             "SB"             "CS"            
## [25] "HBP"            "SF"             "RA"             "ER"            
## [29] "ERA"            "CG"             "SHO"            "SV"            
## [33] "IPouts"         "HA"             "HRA"            "BBA"           
## [37] "SOA"            "E"              "DP"             "FP"            
## [41] "name"           "park"           "attendance"     "BPF"           
## [45] "PPF"            "teamIDBR"       "teamIDlahman45" "teamIDretro"
```

Q2 Player names within the teams that headed to World Series

```
Teams %>%
  select(yearID, teamID, LgWin) %>%
  filter(yearID >= 1903, teamID == 'BOS', LgWin == 'Y') ->
  Boston_team
Fielding %>%
  select(playerID, yearID, teamID) %>%
  filter(yearID >= 1903, teamID == 'BOS') %>%
  unique() ->
  Boston_player
Boston_player_lgwin <- left_join(Boston_team, Boston_player, by = 'yearID')
Boston_player_lgwin_name <- left_join(Boston_player_lgwin, People, by = 'playerID')
Boston_player_lgwin_name %>%
  select(nameFirst, nameLast, yearID) %>%
  arrange(nameLast) ->
  bpln
head(bpln, 10)
```

```
##    nameFirst nameLast yearID
## 1    Alfredo   Aceves   2013
## 2      Jerry    Adair   1967
## 3      Terry    Adams   2004
## 4        Sam    Agnew   1916
## 5        Sam    Agnew   1918
## 6       Nick  Altrock   1903
## 7        Abe  Alvarez   2004
## 8      Jimmy Anderson   2004
## 9      Ernie   Andres   1946
## 10       Kim   Andrew   1975
```

Q3

Total salary for each player in each year.

```
Salaries_aggregate <- aggregate(Salaries$salary,
                                by=list(player=Salaries$playerID, year=Salaries$yearID),
                                FUN=sum)
```

```
Salaries_aggregate %>%
  rename(salary = x) ->
  Salaries_aggregate
head(Salaries_aggregate, 10)
```

```
##       player year  salary
## 1  ackerji01 1985  170000
## 2  agostju01 1985  147500
## 3  aguaylu01 1985  237000
## 4  alexado01 1985  875000
## 5  allenne01 1985  750000
## 6  almonbi01 1985  255000
## 7  anderal02 1985   62500
## 8  anderla02 1985  250500
## 9  andujjo01 1985 1030000
## 10 armasto01 1985  915000
```

Total number of at bats and hits for each player in each year.

```
Batting_AB <- aggregate(Batting$AB,
                        by=list(player=Batting$playerID, year=Batting$yearID),
                        FUN=sum)
Batting_AB %>%
  rename(AB = x) ->
  Batting_AB
Batting_H <- aggregate(Batting$H,
                       by=list(player=Batting$playerID, year=Batting$yearID),
                       FUN=sum)
Batting_H %>%
  rename(H = x) ->
  Batting_H
Batting_aggregate <- full_join(Batting_AB, Batting_H, by = c('player','year'))
head(Batting_aggregate, 10)
```

```
##       player year  AB  H
## 1  abercda01 1871   4  0
## 2   addybo01 1871 118 32
## 3  allisar01 1871 137 40
## 4  allisdo01 1871 133 44
## 5  ansonca01 1871 120 39
## 6  armstbo01 1871  49 11
## 7  barkeal01 1871   4  1
## 8  barnero01 1871 157 63
## 9  barrebi01 1871   5  1
## 10 barrofr01 1871  86 13
```

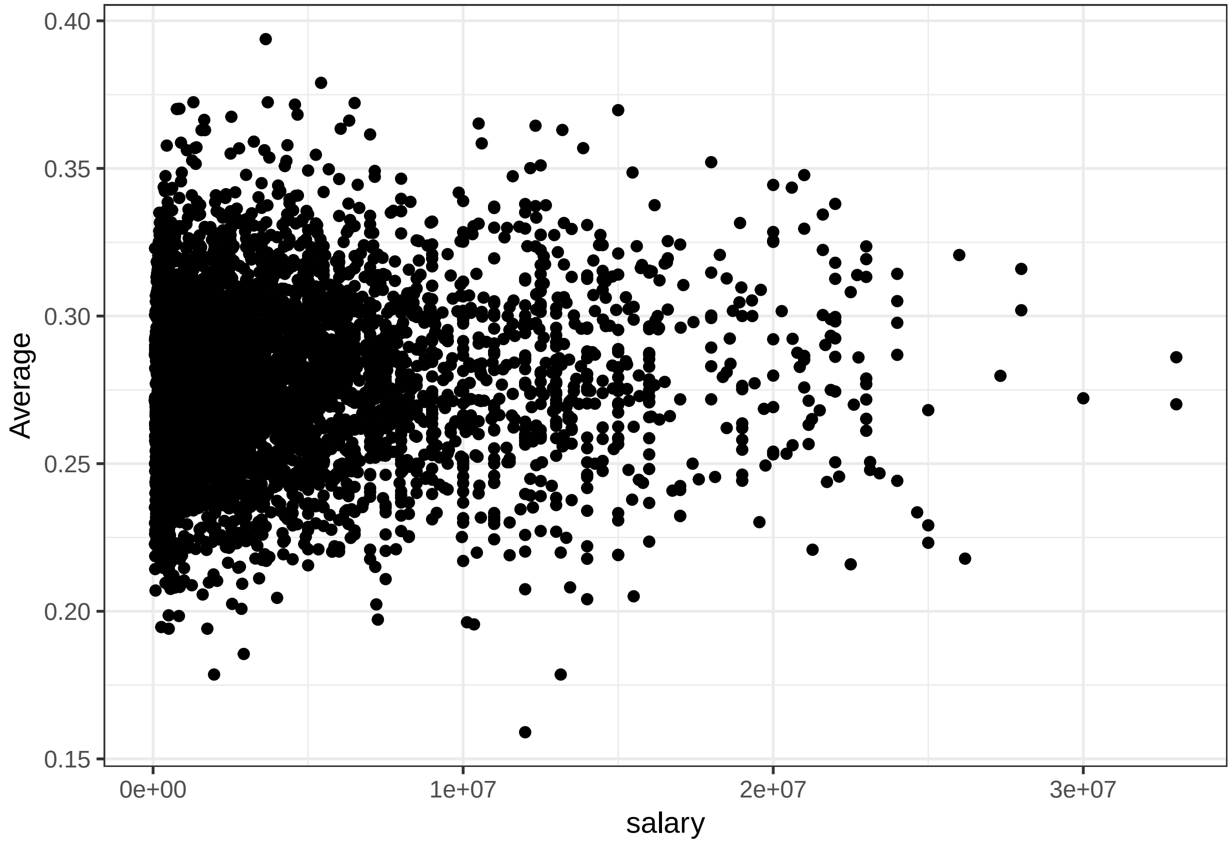
Q4

It seems like batting average is not much relating to salary, players with high batting average are do not always having a high salary, players got high paid may due to other defensive performance.

When considering the time, it is clear that today's players had made more money than before, which may be due to improved athletic performance, increased club investment and league advertising sponsorship.

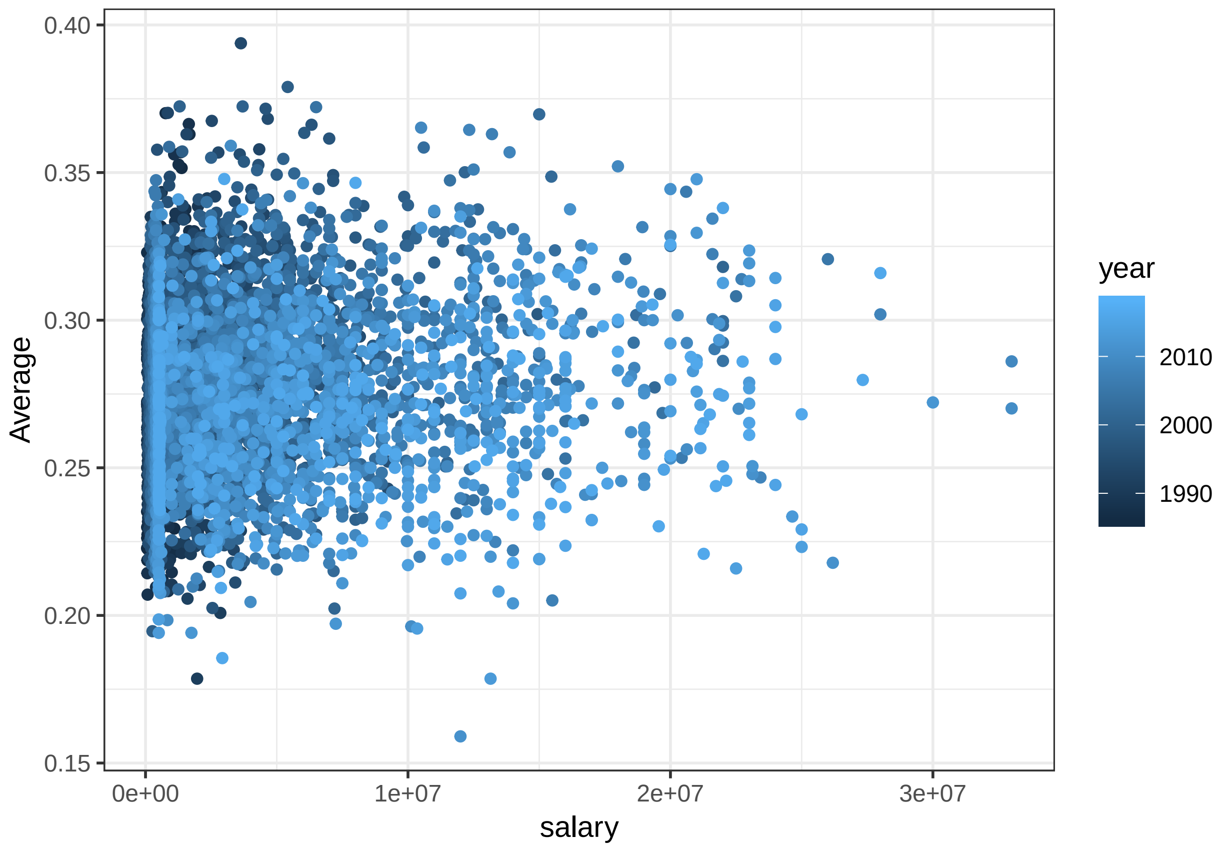
```
Batting_Salary <- left_join(Batting_aggregate, Salaries_aggregate, by = c('player', 'year'))
Batting_Salary %>%
  filter(year > 1985, AB >= 400) %>%
  mutate(Average = H/AB) ->
  Batting_Salary
ggplot(Batting_Salary, aes(x = salary, y = Average)) +
  geom_point() +
  theme_bw()
```

```
## Warning: Removed 492 rows containing missing values (geom_point).
```



```
ggplot(Batting_Salary, aes(x = salary, y = Average, color = year)) +
  geom_point() +
  theme_bw()
```

```
## Warning: Removed 492 rows containing missing values (geom_point).
```



Q5 Salaries of players named 'John' in even numbered years after 1985

```
People %>%
  select(playerID, nameFirst, nameLast) %>%
  filter(nameFirst == 'John') ->
  John
John_Salary <- left_join(John, Salaries, by = 'playerID')
John_Salary %>%
  select(yearID, nameFirst, nameLast, salary) %>%
  filter(yearID > 1985 & yearID %% 2 ==0) %>%
  arrange(desc(salary)) ->
  John_Salary_evenyears
head(John_Salary_evenyears,10)
```

```
##    yearID nameFirst nameLast   salary
## 1    2010      John   Lackey 18700000
## 2    2016      John   Lackey 16000000
## 3    2012      John   Lackey 15950000
## 4    2016      John    Danks 15750000
## 5    2014      John   Lackey 15250000
## 6    2014      John    Danks 14250000
## 7    2008      John   Smoltz 14000000
## 8    2004      John   Smoltz 11666667
## 9    2006      John   Smoltz 11000000
## 10   2000      John   Smoltz  8500000
```

Exercise 4

Q1 Load data

```
asw <- read_table(file = './acceptable_scrabble_words.txt')
```

```
## Parsed with column specification:
## cols(
##   word = col_character()
## )
```

Q2 Number of words either begin or end in 'X' is 885.

There are 309 words start in 'X' and 577 words end in 'X'. 'XEROX' is the only word starts and ends in 'X'. So the number is 390 + 577 - 1 = 885.

```
asw %>%
  filter(str_detect(word, '^X')) %>%
  select(word) ->
  X_start
nrow(X_start)
```

```
## [1] 309
```

```
asw %>%
  filter(str_detect(word, 'X$')) %>%
  select(word) ->
  X_end
nrow(X_end)
```

```
## [1] 577
```

```
common <- intersect(X_start$word, X_end$word)
common
```

```
## [1] "XEROX"
```

Q3 Number of words contain all of the vowels is 3476.

```
asw %>%
  filter(str_detect(word, 'A')) ->
  A
asw %>%
  filter(str_detect(word, 'E')) ->
  E
asw %>%
  filter(str_detect(word, 'I')) ->
  I
asw %>%
  filter(str_detect(word, 'O')) ->
  O
asw %>%
  filter(str_detect(word, 'U')) ->
  U
AE <- inner_join(A, E, by = 'word')
AEI <- inner_join(AE, I, by = 'word')
```

```r
AEIO <- inner_join(AEI, O, by = 'word')
AEIOU <- inner_join(AEIO, U, by = 'word')
head(AEIOU, 10)
```

```
## # A tibble: 10 x 1
##    word
##    <chr>
##  1 ABOIDEAU
##  2 ABOIDEAUS
##  3 ABOIDEAUX
##  4 ABOITEAU
##  5 ABOITEAUS
##  6 ABOITEAUX
##  7 ABORTUARIES
##  8 ABSOLUTISE
##  9 ABSOLUTISED
## 10 ABSOLUTISES
```

Q4 Shortest words that contain all of the vowels.

```r
sw <- AEIOU$word
shortest_word <- sw[nchar(sw)==min(nchar(sw))]
shortest_word
```

```
## [1] "DOULEIA" "EULOGIA" "MIAOUED" "MOINEAU" "SEQUOIA"
```

Q5 Still meaningful words after switching of the positions of the first and the last letters.

21285 words still meaningful.

```r
first_letter   <- substr(asw$word, 1, 1)
middle_letters <- substr(asw$word, 2, (str_length(asw$word)-1))
last_letter    <- substr(asw$word, str_length(asw$word), str_length(asw$word))
new_words      <- data.frame(paste(last_letter, middle_letters, first_letter, sep = ''))
names(new_words)[1] <- 'word'
valid_words    <- data.frame(intersect(asw$word, new_words$word))
names(valid_words)[1] <- 'word'
nrow(valid_words)
```

```
## [1] 21285
```

Q6 There are 1694 words have different first and last letters after switching.

```r
fl <- substr(valid_words$word, 1, 1)
ll <- substr(valid_words$word, str_length(valid_words$word), str_length(valid_words$word))
summary(fl != ll)
```

```
##    Mode   FALSE    TRUE
## logical   19591    1694
```

Q7 The longest words where the first and last letters are differnet.

```r
different_words <- as.character(valid_words$word)[which(fl != ll)]
longest_words <- different_words[nchar(different_words)==max(nchar(different_words))]
longest_words
```

```
## [1] "DECOMMISSIONER" "DEMYTHOLOGISER" "DEMYTHOLOGIZER" "RECOMMISSIONED"
## [5] "REMYTHOLOGISED" "REMYTHOLOGIZED"
```

Q8

```r
writeLines(c("Do not have an idea!!"))
```

```
## Do not have an idea!!
```