Lab 13: Relational Databases

Name: Rufus Petrie

This week's agenda: practicing SQLite queries, performing simple computations and joins, and testing our understanding by writing equivalent R code for these database manipulations.

Lahman baseball database

Thanks to Sean Lahman, extensive baseball data is freely available from the 1871 season all the way to the current season. We're going ot use a SQLite version of the baseball database put together by Jeff Knecht, at https://github.com/jknecht/baseball-archive-sqlite. The most recent SQLite database was recently updated to include the 2016 season. It has been posted to the class website at http://www.stat.cmu.edu/~ryantibs/statcomp/data/lahman2016.sqlite. Download this file (it's about 50 MB) and save it in the working directory for your lab.

Practice with SQL data extraction

- **1a.** Install the packages `DBI`, `RSQLite` if you haven't done so already, and load them into your R session. Using `dbDriver()`, `dbConnect()`, set up a connection called `con` the SQLite database stored in lahman2016.sqlite. Then, use `dbListTables()` to list the tables in the database.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(DBI)
library(RSQLite)
setwd("C:/Users/Rufus/Documents/R")
drv = dbDriver("SQLite")
con = dbConnect(drv, dbname="lahman2016.sqlite")
dbListTables(con)
```

```
##  [1] "AllstarFull"         "Appearances"         "AwardsManagers"
##  [4] "AwardsPlayers"       "AwardsShareManagers" "AwardsSharePlayers"
##  [7] "Batting"             "BattingPost"         "CollegePlaying"
## [10] "Fielding"            "FieldingOF"          "FieldingOFsplit"
## [13] "FieldingPost"        "HallOfFame"          "HomeGames"
## [16] "Managers"            "ManagersHalf"        "Master"
```

```
## [19] "Parks"               "Pitching"            "PitchingPost"
## [22] "Salaries"            "Schools"             "SeriesPost"
## [25] "Teams"               "TeamsFranchises"     "TeamsHalf"
```

- **1b.** Using `dbReadTable()`, grab the table named "Batting" and save it as a data frame in your R session, called `batting`. Check that `batting` is indeed a data frame, and that it has dimension 102816 x 24.

```
batting <- dbReadTable(con, "Batting")
class(batting)
```

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

```
dim(batting)
```

```
## [1] 102816     24
```

- **1c.** Remove `eval=FALSE` from the preamble in the R code chunks below. Then, after each SQL query (each call to `dbGetQuery()`), explain in words what is being extracted, and write one line of base R code (sometimes you might need two lines) to get the same result using the `batting` data frame.

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "ORDER BY yearID",
                      "LIMIT 10"))
```

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

```
# Select five columns from batting, sort by yearID, and display the first 10
batting[order(batting$yearID), c("playerID", "yearID", "AB", "H", "HR")][1:10,]
```

```
##       playerID yearID  AB  H HR
## 154  abercda01   1871   4  0  0
## 546   addybo01   1871 118 32  0
## 1218 allisar01   1871 137 40  0
## 1239 allisdo01   1871 133 44  2
## 2139 ansonca01   1871 120 39  0
## 2455 armstbo01   1871  49 11  0
## 4164 barkeal01   1871   4  1  0
## 4313 barnero01   1871 157 63  0
## 4402 barrebi01   1871   5  1  0
## 4519 barrofr01   1871  86 13  0
```

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "ORDER BY HR DESC",
                      "LIMIT 10"))
```

```
##     playerID yearID  AB   H HR
```

```
## 1  bondsba01   2001 476 156 73
## 2  mcgwima01   1998 509 152 70
## 3   sosasa01   1998 643 198 66
## 4  mcgwima01   1999 521 145 65
## 5   sosasa01   2001 577 189 64
## 6   sosasa01   1999 625 180 63
## 7  marisro01   1961 590 159 61
## 8   ruthba01   1927 540 192 60
## 9   ruthba01   1921 540 204 59
## 10  foxxji01   1932 585 213 58
```

```
# Same as last one but order by HRs instead
batting[order(-batting$HR), c("playerID", "yearID", "AB", "H", "HR")][1:10,]
```

```
##        playerID yearID  AB   H HR
## 8376  bondsba01   2001 476 156 73
## 60534 mcgwima01   1998 509 152 70
## 87306  sosasa01   1998 643 198 66
## 60535 mcgwima01   1999 521 145 65
## 87309  sosasa01   2001 577 189 64
## 87307  sosasa01   1999 625 180 63
## 57291 marisro01   1961 590 159 61
## 80905  ruthba01   1927 540 192 60
## 80899  ruthba01   1921 540 204 59
## 30076  foxxji01   1932 585 213 58
```

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "WHERE HR > 55",
                      "ORDER BY HR DESC"))
```

```
##     playerID yearID  AB   H HR
## 1  bondsba01   2001 476 156 73
## 2  mcgwima01   1998 509 152 70
## 3   sosasa01   1998 643 198 66
## 4  mcgwima01   1999 521 145 65
## 5   sosasa01   2001 577 189 64
## 6   sosasa01   1999 625 180 63
## 7  marisro01   1961 590 159 61
## 8   ruthba01   1927 540 192 60
## 9   ruthba01   1921 540 204 59
## 10  foxxji01   1932 585 213 58
## 11 greenha01   1938 556 175 58
## 12 howarry01   2006 581 182 58
## 13 gonzalu01   2001 609 198 57
## 14 rodrial01   2002 624 187 57
## 15 griffke02   1997 608 185 56
## 16 griffke02   1998 633 180 56
## 17 wilsoha01   1930 585 208 56
```

```
# Don't limit, keep people with a high HR count
df <- batting[batting$HR>55, c("playerID", "yearID", "AB", "H", "HR")]
df[order(-df$HR),]
```

```
##         playerID yearID  AB   H HR
## 8376   bondsba01   2001 476 156 73
```

```
## 60534  mcgwima01   1998 509 152 70
## 87306   sosasa01   1998 643 198 66
## 60535  mcgwima01   1999 521 145 65
## 87309   sosasa01   2001 577 189 64
## 87307   sosasa01   1999 625 180 63
## 57291  marisro01   1961 590 159 61
## 80905   ruthba01   1927 540 192 60
## 80899   ruthba01   1921 540 204 59
## 30076   foxxji01   1932 585 213 58
## 35076  greenha01   1938 556 175 58
## 43064  howarry01   2006 581 182 58
## 34020  gonzalu01   2001 609 198 57
## 79060  rodrial01   2002 624 187 57
## 35490  griffke02   1997 608 185 56
## 35491  griffke02   1998 633 180 56
## 100186 wilsoha01   1930 585 208 56
```

```
dbGetQuery(con, paste("SELECT playerID, yearID, AB, H, HR",
                      "FROM Batting",
                      "WHERE yearID >= 1990 AND yearID <= 2000",
                      "ORDER BY HR DESC",
                      "LIMIT 10"))
```

```
##     playerID yearID  AB   H HR
## 1  mcgwima01   1998 509 152 70
## 2   sosasa01   1998 643 198 66
## 3  mcgwima01   1999 521 145 65
## 4   sosasa01   1999 625 180 63
## 5  griffke02   1997 608 185 56
## 6  griffke02   1998 633 180 56
## 7  mcgwima01   1996 423 132 52
## 8  fieldce01   1990 573 159 51
## 9  anderbr01   1996 579 172 50
## 10 belleal01   1995 546 173 50
```

```
df <- batting[(batting$yearID>=1990 & batting$yearID<=2000), c("playerID", "yearID", "AB", "H", "HR")]
df[order(-df$HR),][1:10,]
```

```
##        playerID yearID  AB   H HR
## 60534 mcgwima01   1998 509 152 70
## 87306  sosasa01   1998 643 198 66
## 60535 mcgwima01   1999 521 145 65
## 87307  sosasa01   1999 625 180 63
## 35490 griffke02   1997 608 185 56
## 35491 griffke02   1998 633 180 56
## 60531 mcgwima01   1996 423 132 52
## 28345 fieldce01   1990 573 159 51
## 1774  anderbr01   1996 579 172 50
## 5741  belleal01   1995 546 173 50
```

- **1d.** Replicate the computations in the last question on more time, now using `dplyr` verbs and pipes.

```
# Query 1
batting %>%
  select(playerID, yearID, AB, H, HR) %>%
  arrange(yearID) %>%
  head(10)
```

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

```
batting %>%
  select(playerID, yearID, AB, H, HR) %>%
  arrange(-HR) %>%
  head(10)
```

```
##     playerID yearID  AB   H HR
## 1  bondsba01   2001 476 156 73
## 2  mcgwima01   1998 509 152 70
## 3   sosasa01   1998 643 198 66
## 4  mcgwima01   1999 521 145 65
## 5   sosasa01   2001 577 189 64
## 6   sosasa01   1999 625 180 63
## 7  marisro01   1961 590 159 61
## 8   ruthba01   1927 540 192 60
## 9   ruthba01   1921 540 204 59
## 10  foxxji01   1932 585 213 58
```

```
batting %>%
  select(playerID, yearID, AB, H, HR) %>%
  filter(HR>55) %>%
  arrange(-HR)
```

```
##     playerID yearID  AB   H HR
## 1  bondsba01   2001 476 156 73
## 2  mcgwima01   1998 509 152 70
## 3   sosasa01   1998 643 198 66
## 4  mcgwima01   1999 521 145 65
## 5   sosasa01   2001 577 189 64
## 6   sosasa01   1999 625 180 63
## 7  marisro01   1961 590 159 61
## 8   ruthba01   1927 540 192 60
## 9   ruthba01   1921 540 204 59
## 10  foxxji01   1932 585 213 58
## 11 greenha01   1938 556 175 58
## 12 howarry01   2006 581 182 58
## 13 gonzalu01   2001 609 198 57
## 14 rodrial01   2002 624 187 57
## 15 griffke02   1997 608 185 56
## 16 griffke02   1998 633 180 56
## 17 wilsoha01   1930 585 208 56
```

```
batting %>%
  select(playerID, yearID, AB, H, HR) %>%
  filter(yearID>=1990 & yearID<=2000) %>%
  arrange(-HR) %>%
  head(10)
```

```
##     playerID yearID  AB   H HR
## 1  mcgwima01   1998 509 152 70
## 2   sosasa01   1998 643 198 66
## 3  mcgwima01   1999 521 145 65
## 4   sosasa01   1999 625 180 63
## 5  griffke02   1997 608 185 56
## 6  griffke02   1998 633 180 56
## 7  mcgwima01   1996 423 132 52
## 8  fieldce01   1990 573 159 51
## 9  anderbr01   1996 579 172 50
## 10 belleal01   1995 546 173 50
```

Practice with SQL computations

- **2a.** As before, remove `eval=FALSE` from the preamble in the following R code chunks. Then, after each SQL query, explain in words what is being extracted, and write one line of base R code to get the same result using the `batting` data frame. Hint: often you'll have to use `na.rm=TRUE` to deal with NA values, for example `mean(x, na.rm=TRUE)` computes the mean of a vector `x` after removing any NA values.

```
dbGetQuery(con, paste("SELECT AVG(HR)",
                      "FROM Batting"))
```

```
##    AVG(HR)
## 1 2.813599
```

```
# Select the average homerun count of all observations
mean(batting$HR, na.rm=TRUE)
```

```
## [1] 2.813599
```

```
dbGetQuery(con, paste("SELECT SUM(HR)",
                      "FROM Batting"))
```

```
##   SUM(HR)
## 1  289283
```

```
# Select total homerun count from all observations
sum(batting$HR)
```

```
## [1] 289283
```

```
dbGetQuery(con, paste("SELECT playerID, yearID, teamID, MAX(HR)",
                      "FROM Batting"))
```

```
##    playerID yearID teamID MAX(HR)
## 1 bondsba01   2001    SFN      73
```

```
# Select information about the player with the most homeruns
batting[batting$HR == max(batting$HR, na.rm=TRUE),][,c("playerID", "yearID", "teamID", "HR")]
```

```
##       playerID yearID teamID HR
```

```
## 8376 bondsba01   2001    SFN 73
```

```
dbGetQuery(con, paste("SELECT AVG(HR)",
                      "FROM Batting",
                      "WHERE yearID >= 1990"))
```

```
##    AVG(HR)
## 1 3.585889
```

```
# Select average homerun count from 1990 onwards
mean(batting[batting$yearID>=1990,][,"HR"])
```

```
## [1] 3.585889
```

- **2b.** Again, after each SQL query explain in words what is being extracted, and write one line (or two lines) of R code to get the same result using the `batting` data frame. You may use base R, `plyr`, `dplyr`, pipes, or whatever means you want.

```
dbGetQuery(con, paste("SELECT teamID, AVG(HR)",
                      "FROM Batting",
                      "WHERE yearID >= 1990",
                      "GROUP BY teamID",
                      "LIMIT 5"))
```

```
##   teamID  AVG(HR)
## 1    ANA 3.928783
## 2    ARI 3.610227
## 3    ATL 3.822374
## 4    BAL 3.958401
## 5    BOS 3.621142
```

```
# Select average player homerun count by team after the year 1990
batting %>%
  filter(yearID>=1990) %>%
  group_by(teamID) %>%
  summarize(avghr = mean(HR)) %>%
  head(5)
```

```
## # A tibble: 5 x 2
##   teamID avghr
##   <chr>  <dbl>
## 1 ANA     3.93
## 2 ARI     3.61
## 3 ATL     3.82
## 4 BAL     3.96
## 5 BOS     3.62
```

```
dbGetQuery(con, paste("SELECT teamID, AVG(HR)",
                      "FROM Batting",
                      "WHERE yearID < 1960",
                      "GROUP BY teamID",
                      "ORDER BY AVG(HR) DESC",
                      "LIMIT 5"))
```

```
##   teamID  AVG(HR)
## 1    ML1 5.029412
## 2    SFN 4.616438
## 3    LAN 3.950617
## 4    NYP 3.882353
```

```
## 5    BSP 3.176471
```

```
# Select average player homerun count by team before 1960 and order
batting %>%
  filter(yearID<1960) %>%
  group_by(teamID) %>%
  summarize(avghr = mean(HR)) %>%
  arrange(-avghr) %>%
  head((5))
```

```
## # A tibble: 5 x 2
##   teamID avghr
##   <chr>  <dbl>
## 1 ML1     5.03
## 2 SFN     4.62
## 3 LAN     3.95
## 4 NYP     3.88
## 5 BSP     3.18
```

```
dbGetQuery(con, paste("SELECT teamID, yearID, AVG(HR)",
                      "FROM Batting",
                      "WHERE yearID == 1991 OR yearID == 1992",
                      "GROUP BY teamID, yearID",
                      "ORDER BY AVG(HR) DESC",
                      "LIMIT 15"))
```

```
##    teamID yearID  AVG(HR)
## 1     DET   1992 5.352941
## 2     DET   1991 5.097561
## 3     CIN   1991 4.100000
## 4     CHN   1991 4.076923
## 5     NYA   1992 4.075000
## 6     TOR   1992 4.075000
## 7     BAL   1991 4.047619
## 8     MIN   1991 4.000000
## 9     BAL   1992 3.894737
## 10    NYA   1991 3.675000
## 11    CHA   1991 3.657895
## 12    TEX   1991 3.612245
## 13    SDN   1992 3.461538
## 14    OAK   1991 3.456522
## 15    SFN   1991 3.439024
```

```
# Select avg player hr count from 1991/1992 grouped by team/year and sorted
batting %>%
  filter(yearID==1991 | yearID==1992) %>%
  group_by(teamID, yearID) %>%
  summarize(avghr=mean(HR)) %>%
  arrange(-avghr) %>%
  head(15)
```

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

```
## # A tibble: 15 x 3
## # Groups:   teamID [12]
##    teamID yearID avghr
##    <chr>   <int> <dbl>
```

```
##  1 DET      1992  5.35
##  2 DET      1991  5.10
##  3 CIN      1991  4.1
##  4 CHN      1991  4.08
##  5 NYA      1992  4.08
##  6 TOR      1992  4.08
##  7 BAL      1991  4.05
##  8 MIN      1991  4
##  9 BAL      1992  3.89
## 10 NYA      1991  3.68
## 11 CHA      1991  3.66
## 12 TEX      1991  3.61
## 13 SDN      1992  3.46
## 14 OAK      1991  3.46
## 15 SFN      1991  3.44
```

More practice with computations

- **3a.** Use a SQL query on the "Batting" table to calculate each player's average number of hits (`H`) over the seasons they played, and display the players with the 10 highest hit averages, along with their hit averages. Hint: `AVG()`, `GROUP BY`, `ORDER BY`.

```
dbGetQuery(con, paste("SELECT playerID, AVG(H)",
                      "FROM Batting",
                      "GROUP BY playerID",
                      "ORDER BY AVG(H) DESC",
                      "LIMIT 10"))
```

```
##     playerID   AVG(H)
## 1  puckeki01 192.0000
## 2   canoro01 184.1667
## 3  cabremi01 179.9286
## 4  abreujo02 179.0000
## 5  suzukic01 178.2353
## 6  burkeje01 178.1250
## 7  pujolal01 176.5625
## 8  sislege01 175.7500
## 9   cobbty01 174.5417
## 10 altuvjo01 174.3333
```

- **3b.** Calculate the same as in the last question, but now display all players whose hit averages are above 170. Hint: `HAVING`.

```
dbGetQuery(con, paste("SELECT playerID, AVG(H)",
                      "FROM Batting",
                      "GROUP BY playerID",
                      "HAVING AVG(H) >170",
                      "ORDER BY AVG(H) DESC"))
```

```
##     playerID   AVG(H)
## 1  puckeki01 192.0000
## 2   canoro01 184.1667
## 3  cabremi01 179.9286
## 4  abreujo02 179.0000
## 5  suzukic01 178.2353
```

```
## 6  burkeje01 178.1250
## 7  pujolal01 176.5625
## 8  sislege01 175.7500
## 9   cobbty01 174.5417
## 10 altuvjo01 174.3333
## 11 jeterde01 173.2500
## 12 markani01 171.7273
## 13 ashburi01 171.6000
## 14 dimagjo01 170.3077
## 15  rosepe01 170.2400
```

- **3c.** Calculate the same as in the last question, but now display for all players with hit averages above 170—-in addition to the player's ID and his batting average—the last year in which each player played.

```
dbGetQuery(con, paste("SELECT playerID, AVG(H), MAX(yearID)",
                      "FROM Batting",
                      "GROUP BY playerID",
                      "HAVING AVG(H) >170",
                      "ORDER BY AVG(H) DESC"))
```

```
##     playerID   AVG(H) MAX(yearID)
## 1  puckeki01 192.0000        1995
## 2   canoro01 184.1667        2016
## 3  cabremi01 179.9286        2016
## 4  abreujo02 179.0000        2016
## 5  suzukic01 178.2353        2016
## 6  burkeje01 178.1250        1905
## 7  pujolal01 176.5625        2016
## 8  sislege01 175.7500        1930
## 9   cobbty01 174.5417        1928
## 10 altuvjo01 174.3333        2016
## 11 jeterde01 173.2500        2014
## 12 markani01 171.7273        2016
## 13 ashburi01 171.6000        1962
## 14 dimagjo01 170.3077        1951
## 15  rosepe01 170.2400        1986
```

Practice with SQL join operations

- **4a.** Using `JOIN`, merge the "Batting" and "Salaries" tables based on matching the `yearID`, `playerID` pairs. Display the year, player, salary, and number of hits for the first 10 records.

```
salaries = dbReadTable(con, "Salaries")
dbGetQuery(con, paste("SELECT yearID, playerID, salary, H",
                      "FROM Batting LEFT JOIN Salaries USING(yearID, playerID)",
                      "LIMIT 10"))
```

```
##    yearID  playerID  salary   H
## 1    2004 aardsda01  300000   0
## 2    2006 aardsda01      NA   0
## 3    2007 aardsda01  387500   0
## 4    2008 aardsda01  403250   0
## 5    2009 aardsda01  419000   0
## 6    2010 aardsda01 2750000   0
## 7    2012 aardsda01  500000   0
```

```
## 8    2013 aardsda01      NA   0
## 9    2015 aardsda01      NA   0
## 10   1954 aaronha01      NA 131
```

- **4b.** Building off of the code from the end of lecture, which does something similar, compute the average salaries for the players with the top 10 highest hit averages.

```
dbGetQuery(con, paste("SELECT playerID, AVG(salary), AVG(H)",
                      "FROM Batting JOIN Salaries USING(yearID, playerID)",
                      "GROUP BY playerID",
                      "ORDER BY AVG(H) DESC",
                      "LIMIT 10"))
```

```
##     playerID AVG(salary)   AVG(H)
## 1  altuvjo01     1685240 197.0000
## 2  puckeki01     2708182 194.4545
## 3  bettsmo01      540250 194.0000
## 4  seageco01      510000 193.0000
## 5   canoro01    11806527 186.8182
## 6  lindofr01      540300 182.0000
## 7  jeterde01    13927268 181.7368
## 8  cabremi01    13457902 179.9286
## 9  abreujo02     9110889 179.0000
## 10 suzukic01    10713617 178.2353
```

- **4c.** Compute the hit averages for the players with the top 10 highest salaries. Hint: this should only require a very small tweak to the code you wrote for the last question.

```
dbGetQuery(con, paste("SELECT playerID, AVG(salary), AVG(H)",
                      "FROM Batting JOIN Salaries USING(yearID, playerID)",
                      "GROUP BY playerID",
                      "ORDER BY AVG(salary) DESC",
                      "LIMIT 10"))
```

```
##     playerID AVG(salary)      AVG(H)
## 1  tanakma01    22000000   0.3333333
## 2  rodrial01    18109830 141.5909091
## 3  howarry01    15525500 130.9000000
## 4  teixema01    14735938 116.3750000
## 5  jeterde01    13927268 181.7368421
## 6  fieldpr01    13901318 148.0000000
## 7  sabatcc01    13642857   1.6666667
## 8  hamelco01    13536364   9.2727273
## 9  cabremi01    13457902 179.9285714
## 10 mauerjo01    13232692 140.4615385
```

- **4d.** Using the "Fielding" table, list the 10 worst (highest) number of errors (`E`) committed by a player in a season, only considering the year 1990 and later. In addition to the number of errors, list the year and player ID for each record.

```
fielding = dbReadTable(con, "Fielding")
dbGetQuery(con, paste("SELECT playerID, yearID, E",
                      "FROM Fielding",
                      "WHERE yearID>=1990",
                      "ORDER BY E DESC",
                      "LIMIT 10"))
```

```
##     playerID yearID  E
```

```
## 1  offerjo01   1992 42
## 2  offerjo01   1993 37
## 3  valenjo03   1996 37
## 4  valenjo03   2000 36
## 5  carusmi01   1998 35
## 6  offerjo01   1995 35
## 7  semiema01   2015 35
## 8  desmoia01   2010 34
## 9  reynoma01   2008 34
## 10 cordewi01   1993 33
```

- **4e.** By appropriately merging the "Fielding" and "Salaries" tables, list the salaries for each record that you extracted in the last question. Then, answer the following question: what was the highest salary paid to a player who made at least 30 errors in a season, after 1990?

```
dbGetQuery(con, paste("SELECT playerID, yearID, salary, E",
                      "FROM Fielding JOIN Salaries USING(yearID, playerID)",
                      "WHERE yearID>=1990 AND E>30",
                      "ORDER BY salary DESC",
                      "LIMIT 5"))
```

```
##    playerID yearID  salary  E
## 1 furcara01   2003 2200000 31
## 2 offerjo01   1995 1600000 35
## 3 valenjo03   2000 1320000 36
## 4 zeileto01   1993 1025000 33
## 5 davisru01   1998 1000000 32
```

All about the money

- **5a.** Use a SQL query on the "Salaries" table to compute the payroll (total of salaries) for each team in the year 2010, and display the 3 teams with the highest payrolls. Do the same, but display the 3 teams with the lowest payroll (ouch!).

```
dbGetQuery(con, paste("SELECT teamID, sum(salary)",
                      "FROM Salaries",
                      "WHERE yearID==2010",
                      "GROUP BY teamID",
                      "ORDER BY sum(salary) DESC",
                      "LIMIT 3"))
```

```
##   teamID sum(salary)
## 1    NYA   206333389
## 2    BOS   162447333
## 3    CHN   146609000
```

```
dbGetQuery(con, paste("SELECT teamID, sum(salary)",
                      "FROM Salaries",
                      "WHERE yearID==2010",
                      "GROUP BY teamID",
                      "ORDER BY sum(salary)",
                      "LIMIT 3"))
```

```
##   teamID sum(salary)
## 1    PIT    34943000
```

```
## 2    SDN    37799300
## 3    TEX    55250544
```

- **5b.** Use a SQL query to compute the total payroll for each team, added up over the years between 1985 and 2016. Hint: `dbGetQuery()` actually returns a data frame. You should have a data frame of dimension 46 x 2, and the 2 columns should display the team ID and the payroll. Check that your data frame has the right dimensions and display its first 10 rows. Then, answer: what team has the highest total payroll? The lowest payroll? Where do the Pirates rank?

```
dbGetQuery(con, paste("SELECT teamID, sum(salary)",
                      "FROM Salaries",
                      "WHERE yearID>=1985 AND yearID<=2016",
                      "GROUP BY teamID",
                      "ORDER BY sum(salary) DESC",
                      "LIMIT 10"))
```

```
##    teamID sum(salary)
## 1     NYA  3495871291
## 2     BOS  2802350096
## 3     LAN  2453558703
## 4     PHI  2153028800
## 5     DET  2138358918
## 6     NYN  2117310904
## 7     ATL  2023226325
## 8     SFN  2004454588
## 9     TEX  1989872010
## 10    CHN  1975712625
```

The Yankees had the highest payroll. The Rays had the lowest payroll. The Pirates had the 27th highest payroll in this time.

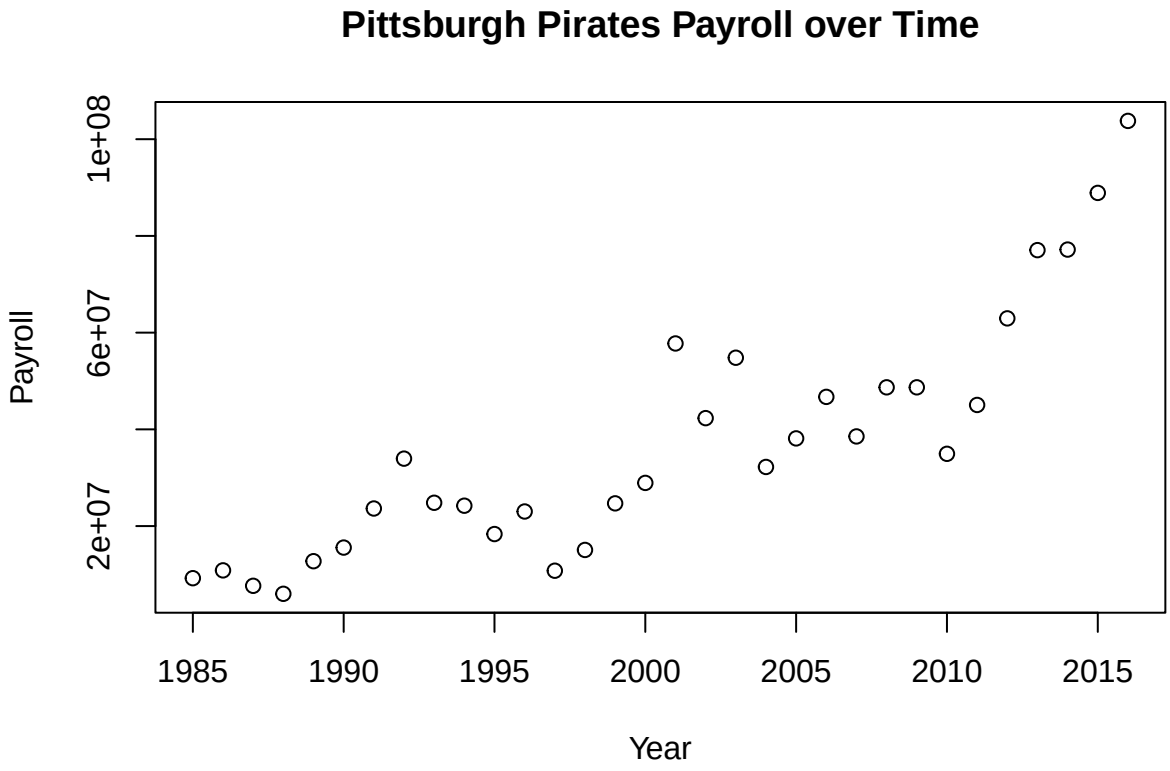
- **5c.** Use a SQL query to compute the payroll for each team, separately for each year in between 1985 and 2016. Hint: `GROUP BY` can take two arguments, separated by a comma. You should have a data frame of dimension 918 x 3, and the 3 columns should be display the team ID, year, and payroll. Check that your data frame has the proper dimensions, and display its last 10 rows.

```
dbGetQuery(con, paste("SELECT teamID, yearID, sum(salary)",
                      "FROM Salaries",
                      "WHERE yearID>=1985 AND yearID<=2016",
                      "GROUP BY teamID, yearID",
                      "ORDER BY sum(salary)",
                      "LIMIT 10"))
```

```
##    teamID yearID sum(salary)
## 1     TEX   1987      880000
## 2     SEA   1987     2263500
## 3     SEA   1985     4613000
## 4     TEX   1988     5342131
## 5     MIN   1985     5764821
## 6     SEA   1986     5958309
## 7     PIT   1988     5998500
## 8     CHA   1988     6390000
## 9     MIN   1987     6397500
## 10    CLE   1985     6551666
```

- **5d.** Plot the Pittsburgh Pirates' payroll over time (i.e., over the years 1985 to 2016), with appropriately labeled axes and an appropriate title. What is the trend that you see?

```
pit <- dbGetQuery(con, paste("SELECT teamID, yearID, sum(salary)",
                      "FROM Salaries",
                      "WHERE yearID>=1985 AND yearID<=2016",
                      "GROUP BY teamID, yearID",
                      "ORDER BY sum(salary)"))
pit <- pit[pit$teamID=="PIT",]
pit <- pit[order(pit$yearID),]
plot(pit$yearID, pit$`sum(salary)`,
     main = "Pittsburgh Pirates Payroll over Time", ylab="Payroll", xlab="Year")
```



The Pirates' payroll has increased on average over time, but there have been a lot of years where it has decreased (the trend changes direction like 7 times).

- **Challenge.** On a single plot, display the payrolls over time (i.e., over the years 1985 to 2016) for 8 teams of your choosing. Make sure that their payroll curves are distinguishable (by color, line type, some combo, you choose). Make sure that the y limit is properly set (so the extremes of all curves are properly contained within the plotting region). Use appropriately labeled axes, an appropriate title, and an informative legend.
- **Challenge.** To make these plots more sensible, we need to adjust for inflation. Find data on the average consumer price index (CPI) over the years 1985 to 2016, and use this to adjust the payrolls for inflation and reproduce your plot from Q2d. Comment on the changes.

Batting averages (optional)

- **6a.** Use a SQL query to calculate the top 10 best batting averages achieved by a player in any season after 1940. Note: batting average is the number of hits (`H`) divided by number of at bats (`AB`) achieved by a player in a given season, but (let's say) it is only defined for players that have at least 400 at bats in that season. Your resulting data frame from the SQL query should be 10 x 3, with the 3 columns displaying the playerID, yearID, and batting average.
- **6b.** Compute batting averages as described above, but now plot a histogram of all of these battings averages (aggregated over all players and all seasons after 1940), with an appropriate title. Use a large value of the `breaks` argument to get a good sense of the shape of the histogram. Does this look like a normal distribution to you? What is the estimated mean and the standard deviation? Overlay the normal density curve on top of your histogram, with the appropriate mean and variance, and comment on how it fits. Perform a rigorous hypothesis test for normality of batting averages here; you might consider using `ks.test()`.
- **6c.** For the computed batting averages in the last question, separate out the batting averages before and after 1985. Plot two overlaid histograms, using transparent colors, for the batting averages before and after 1985. Set an appropriate title and informative legend. Do the distributions look different? If so, how? Perform a rigorous hypothesis test for the difference in distributions here; you might again consider using `ks.test()`.
- **6d.** Modifying your last SQL query so that you also extract, in addition to the batting averages, the number of home runs (for all players and all seasons after 1940). Produce a scatterplot of the number of home runs versus the batting average, with appropriate axes labels and an appropriate title. What does the general trend appear to be? Overlay the least squares regression line on top of your plot. What could go wrong with using this regression line to predict a player's home run total from their batting average?