lista

Lucas Matheus - 211029049

2022-04-21

Questão 1

Com base no banco de dados nycflights13::flights, execute os comandos a seguir, a fim de transformar o banco de dados em uma lista por companhia aerea (carrier), e responda os itens abaixo utilizando a lista banco e as funcionalidades do pacotepurr.

```
banco <- nycflights13::flights%>% split(.$carrier)
```

a) A partir da listabanco, selecione as colunas que contém a palavra 'delay' para cada companhia aerea(carrier). Retorne o resultado como uma lista.

```
lista<-map(banco,~select(., ends_with("delay")))

lista
```

```
## $`9E`
## # A tibble: 18,460 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1         0        11
##  2        -9        -2
##  3        -3        -2
##  4        -6        -1
##  5        -8        -5
##  6         0        -5
##  7         6         5
##  8         0        13
##  9        -8        -8
## 10        -6       -33
## # ... with 18,450 more rows
##
## $AA
## # A tibble: 32,729 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1         2        33
##  2        -2         8
##  3        -1        31
##  4        -4       -12
```

```
##  5        13         5
##  6        -2        -3
##  7        -1        14
##  8         0        48
##  9        -4         4
## 10        -3       -10
## # ... with 32,719 more rows
##
## $AS
## # A tibble: 714 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -1       -10
##  2        -7       -19
##  3        -3       -41
##  4         3         1
##  5        -1       -18
##  6         2        -9
##  7         0         1
##  8        -7       -29
##  9         0       -19
## 10       -12       -12
## # ... with 704 more rows
##
## $B6
## # A tibble: 54,635 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -1       -18
##  2        -5        19
##  3        -3        -8
##  4        -2        -2
##  5        -2        -3
##  6         0        -4
##  7         0        -7
##  8         1        -6
##  9         3         4
## 10         0       -21
## # ... with 54,625 more rows
##
## $DL
## # A tibble: 48,110 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -6       -25
##  2        -8        -8
##  3        -4        -8
##  4         0        -9
##  5        -7       -33
##  6         0        -9
##  7        -5        -8
##  8        -5       -18
##  9        -3       -14
## 10        -2         5
```

```
## # ... with 48,100 more rows
##
## $EV
## # A tibble: 54,173 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -3       -14
##  2        -6        29
##  3        24        12
##  4        -6        -7
##  5        -4       -26
##  6        -2        23
##  7        -8        -2
##  8        -4       -18
##  9         0        -4
## 10         0       -14
## # ... with 54,163 more rows
##
## $F9
## # A tibble: 685 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -2        32
##  2       -14        -6
##  3        -8        18
##  4        -2        -1
##  5         0         0
##  6       123        98
##  7        -1        -3
##  8        61        36
##  9         0        -5
## 10        -4        -5
## # ... with 675 more rows
##
## $FL
## # A tibble: 3,260 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -3        10
##  2         4        17
##  3        -2         2
##  4        -8         8
##  5       -10        -5
##  6        -3         6
##  7        -8         6
##  8         0        14
##  9       -11         2
## 10       -10        -7
## # ... with 3,250 more rows
##
## $HA
## # A tibble: 342 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
```

```
##  1        -3       -14
##  2         9        -5
##  3        14       -26
##  4         0       -14
##  5        -2       -11
##  6        79        28
##  7       102        50
##  8         1       -26
##  9      1301      1272
## 10        -1       -41
## # ... with 332 more rows
##
## $MQ
## # A tibble: 26,397 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1         0        12
##  2        -3        16
##  3         8        32
##  4        -6        10
##  5        -9        27
##  6        39        49
##  7       -10        -6
##  8       -10        -4
##  9       101       137
## 10        -4       -13
## # ... with 26,387 more rows
##
## $OO
## # A tibble: 32 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        67       107
##  2        -6        -5
##  3        13        27
##  4        -8       -24
##  5         4        -6
##  6         1         3
##  7        -9       -20
##  8       131       157
##  9       -10         3
## 10       154       140
## # ... with 22 more rows
##
## $UA
## # A tibble: 58,665 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1         2        11
##  2         4        20
##  3        -4        12
##  4        -2         7
##  5        -2       -14
##  6        -1        -8
```

```
##  7         0       -17
##  8        11        14
##  9        -4         1
## 10        -2        29
## # ... with 58,655 more rows
##
## $US
## # A tibble: 20,536 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -8         3
##  2        -3         0
##  3        -1        -9
##  4        -2       -11
##  5        -7        -4
##  6        -4        10
##  7        -2        -5
##  8        -7       -29
##  9        -5        11
## 10        -1       -15
## # ... with 20,526 more rows
##
## $VX
## # A tibble: 5,162 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -2         2
##  2        -1       -26
##  3        -1        -2
##  4         2        -6
##  5         1       -22
##  6         3        -2
##  7         3       -26
##  8        -3       -17
##  9        -3       -40
## 10         0        -5
## # ... with 5,152 more rows
##
## $WN
## # A tibble: 12,275 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -1       -19
##  2        -1        33
##  3        -3        26
##  4         4        30
##  5        -2        15
##  6        -3         7
##  7        -1       -14
##  8        -5         5
##  9        -2        -4
## 10         0        23
## # ... with 12,265 more rows
##
```

```
## $YV
## # A tibble: 601 x 2
##    dep_delay arr_delay
##        <dbl>     <dbl>
##  1        -7       -20
##  2       -11       -23
##  3        -5       -13
##  4        89        75
##  5        -8       -15
##  6        -5       -18
##  7        -6        -1
##  8        -3       -22
##  9        -7         5
## 10        -3       -16
## # ... with 591 more rows
```

```
class(lista)
```

```
## [1] "list"
```

b) A partir da lista obtida na letra A, calcule a média dos atrasos de chegada e de saída para cada companhia aerea (carrier). Retorne o resultado como um dataframe.

```
arr <- lista %>%
  lapply(function(x) x = x$arr_delay)%>%
  map_dfc(mean, na.rm = T)
arr<- as.data.frame(t(arr))




dep <- lista %>%
  lapply(function(x) x = x$dep_delay)%>%
  map_dfc(mean, na.rm = T)
dep<-as.data.frame(t(dep))



arr_dep<-cbind(arr,dep)
arr_dep
```

```
##            V1        V1
## 9E  7.3796692 16.725769
## AA  0.3642909  8.586016
## AS -9.9308886  5.804775
## B6  9.4579733 13.022522
## DL  1.6443409  9.264505
## EV 15.7964311 19.955390
## F9 21.9207048 20.215543
## FL 20.1159055 18.726075
## HA -6.9152047  4.900585
```

```
## MQ 10.7747334 10.552041
## OO 11.9310345 12.586207
## UA  3.5580111 12.106073
## US  2.1295951  3.782418
## VX  1.7644644 12.869421
## WN  9.6491199 17.711744
## YV 15.5569853 18.996330
```

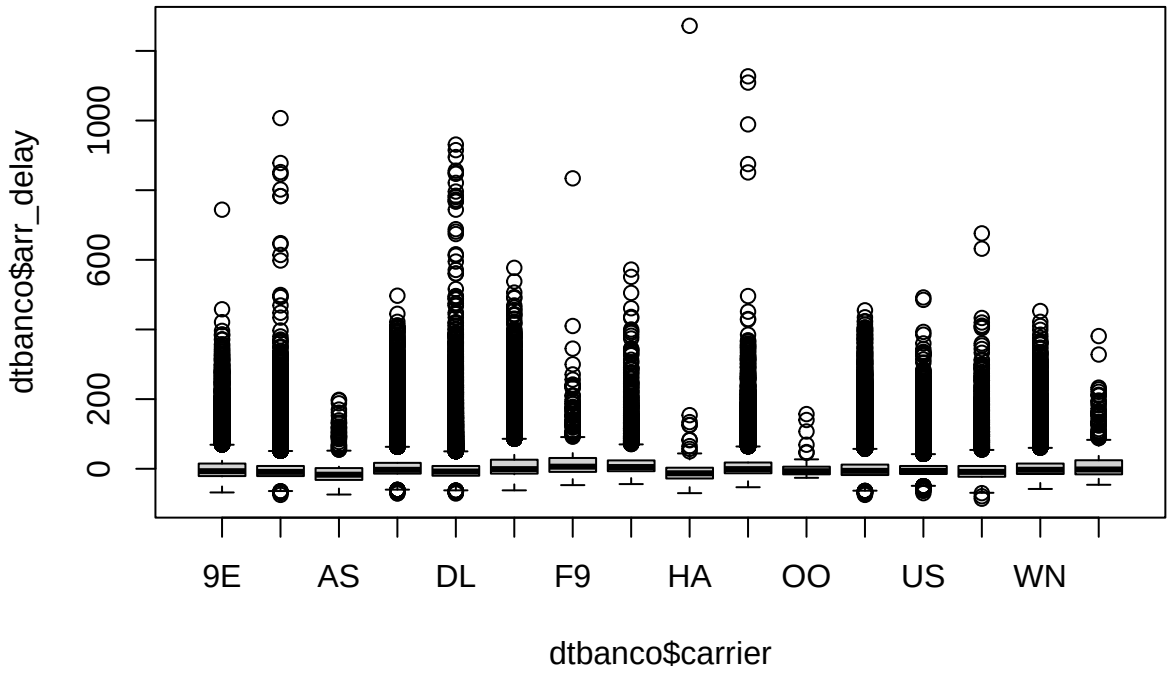
c) Crie uma função para repetir a tarefa da letra B, mas permita ao usuário escolher a operação que seráefetuada nas colunas (média, mínimo, summary, etc . . .). Permita ao usuário escolher se na.rm = T/F.Retorne o resultado como uma lista.

```
function(lista,operação,TF){
    arr <- lista %>%
  lapply(function(x) x = x$arr_delay)%>%
  map_dfc(mean, na.rm = T)
arr<- as.data.frame(t(arr))%>%
  rename(arr = V1)

dep <- lista %>%
  lapply(function(x) x = x$dep_delay)%>%
  map_dfc(mean, na.rm = T)
dep<-as.data.frame(t(dep))%>%
  rename(dep =V1)

arr_dep<-cbind(arr,dep)
   return(as.list(df))
}
```

```
## function(lista,operação,TF){
##     arr <- lista %>%
##   lapply(function(x) x = x$arr_delay)%>%
##   map_dfc(mean, na.rm = T)
## arr<- as.data.frame(t(arr))%>%
##   rename(arr = V1)
##
## dep <- lista %>%
##   lapply(function(x) x = x$dep_delay)%>%
##   map_dfc(mean, na.rm = T)
## dep<-as.data.frame(t(dep))%>%
##   rename(dep =V1)
##
## arr_dep<-cbind(arr,dep)
##    return(as.list(df))
## }
```

d)A partir da lista inicial banco, faça um boxplot da coluna 'arr_delay' para cada companhia aérea (utilizeas funçõesmapeboxplot).

```
x <- as.list(banco)
dtbanco <-map_df(x,~.x)
y <- boxplot(dtbanco$arr_delay~dtbanco$carrier)
```



e)Note que, além do gráfico, a função boxplot retorna uma série de informações relevantes. Com base nisso,retorne quantos outliers na variável 'arr_delay' cada companhia aérea registrou. Retorne o resultado como um vetor

```
limitesup <- quantile(dtbanco$arr_delay,0.975, na.rm = TRUE)
limiteinf <- quantile(dtbanco$arr_delay,0.025, na.rm = TRUE)
x <-which(dtbanco$arr_delay < limiteinf | dtbanco$arr_delay > limitesup)
```

Questão 2

Usando uma cópia da base de dados 'lahmansbaseballdb.sqlite' em formato SQLite (fontehttp://www.seanlahman.com/baseball-archive/statistics/), responda usando a linguagem SQL:

```
conmy = dbConnect(RSQLite::SQLite(),db = "lahmansbaseballdb.sqlite")
conmy
```

```
## <SQLiteConnection>
##   Path: C:\Users\Lucas\Documents\lahmansbaseballdb.sqlite
##   Extensions: TRUE
```

a) Liste as 10 maiores porcentagens de rebatidas por tentativa (batting average). Batting average pode sercalculada dividindo o número de rebatidas (hits, coluna H na tabela Batting) pelo número de tentativas (at-bats, coluna AB na tabela Batting). Inclua o nome do jogador e o ano de ocorrência, além da porcentagematingida. Elimine dos cálculos os jogadores com menos de 50 tentativas.

```
dbListFields(conmy, "batting")
```

```
##  [1] "ID"        "playerID"  "yearID"    "stint"     "teamID"    "team_ID"
##  [7] "lgID"      "G"         "G_batting" "AB"        "R"         "H"
## [13] "2B"        "3B"        "HR"        "RBI"       "SB"        "CS"
## [19] "BB"        "SO"        "IBB"       "HBP"       "SH"        "SF"
## [25] "GIDP"
```

```
dbListFields(conmy, "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] "birth_date"     "debut_date"     "finalgame_date" "death_date"
```

```
dbGetQuery(conmy, statement=
              "SELECT batting.H*100 / batting.AB AS battingaverage, people.nameFirst, batting.yearID
                      FROM batting, people
                      WHERE batting.AB >= 50 AND batting.playerID=people.playerID
                      GROUP BY people.nameFirst, batting.yearID
                      ORDER BY battingaverage DESC
                      LIMIT 10")
```

```
##    battingaverage nameFirst yearID
## 1              49      Levi   1871
## 2              43       Cal   1871
## 3              43      Hugh   1894
## 4              43      Ross   1872
## 5              43      Ross   1873
## 6              43       Tip   1887
## 7              42      Jack   1923
## 8              42       Nap   1901
## 9              42    Rogers   1924
## 10             42      Ross   1876
```

b)Liste as 10 maiores porcentagens de rebatidas por tentativa (batting average). Inclua o nome do jogador,o ano de ocorrência e o nome do college/school (obtido da tabela 'collegeplaying') que o jogador estava noano, além da porcentagem atingida. Elimine dos cálculos os jogadores com menos de 50 tentativas.

```
dbGetQuery(conmy, statement=
              "SELECT batting.H*100 / batting.AB AS battingaverage, people.nameFirst, batting.yearID, c
                      FROM batting, people, collegeplaying
                      WHERE batting.AB >= 50 AND collegeplaying.playerID=people.playerID AND batting.yea
                      GROUP BY people.nameFirst, batting.yearID,collegeplaying.schoolID
                      ORDER BY battingaverage DESC
                      LIMIT 10" )
```

```
##    battingaverage nameFirst yearID   schoolID
## 1              40    Hughie   1896    stbonny
## 2              39      Fred   1894      brown
## 3              39      Ryan   2005   virginia
## 4              38     Frank   1884    cornell
## 5              38    Hughie   1895    stbonny
## 6              36      John   1895    stbonny
## 7              35    Hughie   1897    stbonny
## 8              35      John   1928 mryvilletn
## 9              33     Jimmy   1893  holycross
## 10             33      John   1894    stbonny
```

c)

```
valores <- function(dad, digits = 2) {

  dad <- dad %>%
    na.omit()

  valores <- data.frame(
   Valores = c(
      mean(dad),
      sd(dad),
      var(dad),
      min(dad),
      max(dad),
      quantile(dad,.5),

    )
  )

  rownames(valores) <- c(
    "Média", "Desvio Padrão", "variância",
    "Mediana", "Máximo",
    "Máximo"
  )
```

```
  return(valores)
}
```

d) Liste as 5 college/school com maiores MÉDIAS de porcentagens de rebatidas por tentativa (média daescola com base nos jogadores que estavam jogando lá no ano). Inclua o nome do college/school (obtidoda 'collegeplaying') e a porcentagem média atingida. Elimine dos cálculos os jogadores com menos de 50tentativas.

```
dbGetQuery(conmy, statement=
        "SELECT AVG(batting.H*100 / batting.AB) AS battingaverage, collegeplaying.schoolID
              FROM batting, collegeplaying
              WHERE collegeplaying.playerID = batting.playerID AND batting.yearID = collegeplaying.year
              GROUP BY collegeplaying.schoolID, batting.yearID, collegeplaying.playerID
              ORDER BY battingaverage DESC
              LIMIT 5")
```

```
##   battingaverage schoolID
## 1             40  stbonny
## 2             39    brown
## 3             39 virginia
## 4             38  stbonny
## 5             36  stbonny
```

e) Dentre as college/school com maiores MÉDIAS de porcentagens de rebatidas por tentativa (média daescola com base nos jogadores que estavam jogando lá no ano), liste aquelas que tem University no nome(10 melhores). Inclua o nome do college/school (obtido da 'collegeplaying') e a porcentagem média atingida.Elimine dos cálculos os jogadores com menos de 50 tentativas.

```
dbGetQuery(conmy, statement=
        "SELECT AVG(batting.H*100 / batting.AB) AS battingaverage, schools.name_full
              FROM batting, schools, collegeplaying
              WHERE batting.AB >= 50 AND schools.name_full LIKE '%University%' AND collegeplaying.schoo
              GROUP BY collegeplaying.schoolID, batting.yearID, collegeplaying.playerID
              ORDER BY battingaverage DESC
              LIMIT 10")
```

```
##    battingaverage                  name_full
## 1              40 St. Bonaventure University
## 2              39           Brown University
## 3              39     University of Virginia
## 4              38 St. Bonaventure University
## 5              36 St. Bonaventure University
## 6              35 St. Bonaventure University
## 7              33 St. Bonaventure University
## 8              32      University of Alabama
## 9              32      Georgetown University
## 10             32 St. Bonaventure University
```

Questão 3

Usando uma cópia da base de dados 'lahmansbaseballdb.sqlite' em formato SQLite (fontehttp://www.seanlahman.com/baseball-archive/statistics/), responda usando a linguagem SQL:

```
conmy = dbConnect(RSQLite::SQLite(),db = "lahmansbaseballdb.sqlite")
conmy
```

```
## <SQLiteConnection>
##   Path: C:\Users\Lucas\Documents\lahmansbaseballdb.sqlite
##   Extensions: TRUE
```

a) Liste o jogador com maior salário de cada ano

```
dbGetQuery(conmy,statement=
             "select yearID,max(salary),playerID from salaries group by yearID")
```

```
##    yearID max(salary)  playerID
## 1    1985     2130300 schmimi01
## 2    1986     2800000 fostege01
## 3    1987     2127333 schmimi01
## 4    1988     2340000 smithoz01
## 5    1989     2766667 hershor01
## 6    1990     3200000 yountro01
## 7    1991     3800000 strawda01
## 8    1992     6100000 bonilbo01
## 9    1993     6200000 bonilbo01
## 10   1994     6300000 bonilbo01
## 11   1995     9237500 fieldce01
## 12   1996     9237500 fieldce01
## 13   1997    10000000 belleal01
## 14   1998    14936667 sheffga01
## 15   1999    11949794 belleal01
## 16   2000    15714286 brownke01
## 17   2001    22000000 rodrial01
## 18   2002    22000000 rodrial01
## 19   2003    22000000 rodrial01
## 20   2004    22500000 ramirma02
## 21   2005    26000000 rodrial01
## 22   2006    21680727 rodrial01
## 23   2007    23428571 giambja01
## 24   2008    28000000 rodrial01
## 25   2009    33000000 rodrial01
## 26   2010    33000000 rodrial01
## 27   2011    32000000 rodrial01
## 28   2012    30000000 rodrial01
## 29   2013    29000000 rodrial01
## 30   2014    26000000 greinza01
## 31   2015    32571000 kershcl01
## 32   2016    33000000 kershcl01
```

b) Acrescente a idade do jogador na planilha da letra A

```
dbGetQuery(conmy, statement=
        "SELECT AVG(batting.H*100 / batting.AB) AS battingaverage, schools.name_full
              FROM batting, schools, collegeplaying
              WHERE batting.AB >= 50 AND schools.name_full LIKE '%University%' AND collegeplaying.schoo
              GROUP BY collegeplaying.schoolID, batting.yearID, collegeplaying.playerID
              ORDER BY battingaverage DESC
              LIMIT 10")
```

```
##    battingaverage                 name_full
## 1              40 St. Bonaventure University
## 2              39           Brown University
## 3              39     University of Virginia
## 4              38 St. Bonaventure University
## 5              36 St. Bonaventure University
## 6              35 St. Bonaventure University
## 7              33 St. Bonaventure University
## 8              32      University of Alabama
## 9              32      Georgetown University
## 10             32 St. Bonaventure University
```

c) Filtre os jogadores com idade igual ou maior do que 30 anos na planilha da letra B. Comente o resultado.

```
dbGetQuery(conmy,statement =
             "SELECT salaries.playerID, salaries.yearID, 2016-people.birthYear AS idade
              FROM people, salaries
              WHERE idade >=30 AND salaries.playerID=people.playerID
              GROUP BY salaries.yearID
              HAVING MAX(salary)")
```

```
##     playerID yearID idade
## 1  schmimi01   1985    67
## 2  fostege01   1986    68
## 3  schmimi01   1987    67
## 4  smithoz01   1988    62
## 5  hershor01   1989    58
## 6  yountro01   1990    61
## 7  strawda01   1991    54
## 8  bonilbo01   1992    53
## 9  bonilbo01   1993    53
## 10 bonilbo01   1994    53
## 11 fieldce01   1995    53
## 12 fieldce01   1996    53
## 13 belleal01   1997    50
## 14 sheffga01   1998    48
## 15 belleal01   1999    50
## 16 brownke01   2000    51
## 17 rodrial01   2001    41
```

```
## 18 rodrial01   2002    41
## 19 rodrial01   2003    41
## 20 ramirma02   2004    44
## 21 rodrial01   2005    41
## 22 rodrial01   2006    41
## 23 giambja01   2007    45
## 24 rodrial01   2008    41
## 25 rodrial01   2009    41
## 26 rodrial01   2010    41
## 27 rodrial01   2011    41
## 28 rodrial01   2012    41
## 29 rodrial01   2013    41
## 30 greinza01   2014    33
## 31 verlaju01   2015    33
## 32 greinza01   2016    33
```