Lista 3 - MAE399

Samira Tokunaga, Yiying Lin, André Vinícius

April 21, 2019

Este relatório é o terceiro de uma série de análises dos dados do site https://www.citibikenyc.com/, que contém informações sobre o sistema de bicicletas para transporte público de Nova Yorque. Desta vez, a tarefa era apresentar o comportamento do número de bicicletas em cirulação em função da hora do dia. Foi criada uma função em R que prepara os dados para esta análise - seu funcionamento será explicado adiante. Também foram necessários alguns tratamentos prévios nos dados. Todo esse processo será aqui apresentado passo a passo, e como conclusão, teremos quais são as variáveis que influenciam no crescimento ou decrescimento da função.

#Download dos dados:

```
bikeurl <- "https://s3.amazonaws.com/tripdata/201807-citibike-tripdata.csv.zip"
unzippedfile <- "201807-citibike-tripdata.csv"
if (!file.exists(unzippedfile))
{
  destfile <- "bikesjul.csv.zip"
  if (!file.exists(destfile))
  {
    download.file(bikeurl, destfile = destfile, method = "curl")
    downloadtime <- date()
  }
  unzip(destfile)
}
dadosbike <- read.csv(unzippedfile)
```

#Bibliotecas

```
library(dplyr) # uso geral
library(lubridate) # manipular datas e horas
```

#Redução dos dados e tratamento das variáveis de tempo

```
#selecionar apenas as colunas que serão usadas
dadosbike <- dadosbike %>% select(tripduration, starttime, stoptime, usertype)
#transformar as strings em variáveis de data e hora - lubridate ymd_hms
dadosbike$starttime <- ymd_hms(dadosbike$starttime)
dadosbike$stoptime <- ymd_hms(dadosbike$stoptime)
#criar uma coluna para as semanas do mês - lubridate week
dadosbike <- dadosbike %>% mutate(stopweek = week(stoptime))
#criar um vetor com as semanas do mês
weeks <- unique(dadosbike$stopweek)
#identificar qual é a segunda semana
myweek <- weeks[2]
#selecionar apenas as linhas da semana desejada
dadosbike <- dadosbike %>% filter(stopweek == myweek)
```

#Excluindo outliers - Questão 1

```
x <- quantile(dadosbike$tripduration, probs = c(0.995))
paste("x =", x)
```

```
## [1] "x = 6982.01500000001"
```

```
dadosbike <- dadosbike %>% filter(tripduration <= x)
```

#Questões 2 e 3 Os passos para responder esta questão estão abaixo, nos comentários do código:

```
#criar uma variável que identifica o dia da semana em que a bike foi devolvida
dadosbike <- dadosbike %>% mutate(weekday = wday(stoptime))
circulating <- dadosbike %>%
   #filtar as viagens em que o dia de início é diferente do dia de devolução
   filter(day(starttime) != day(stoptime)) %>%
   #agrupar pela variável do dia da semana
   group_by(weekday) %>%
   #contar
   count()

#resultado das transformações
circulating
```

```
## # A tibble: 7 x 2
## # Groups:   weekday [7]
##   weekday     n
##     <dbl> <int>
## 1       1   283
## 2       2   170
## 3       3   176
## 4       4   217
## 5       5   343
## 6       6   165
## 7       7   271
```

```
#estatísticas dos dados
summary(circulating$n)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   165.0   173.0   217.0   232.1   277.0   343.0
```

#Questões 4 e 5 #Contando as bicicletas em circulação - código da função

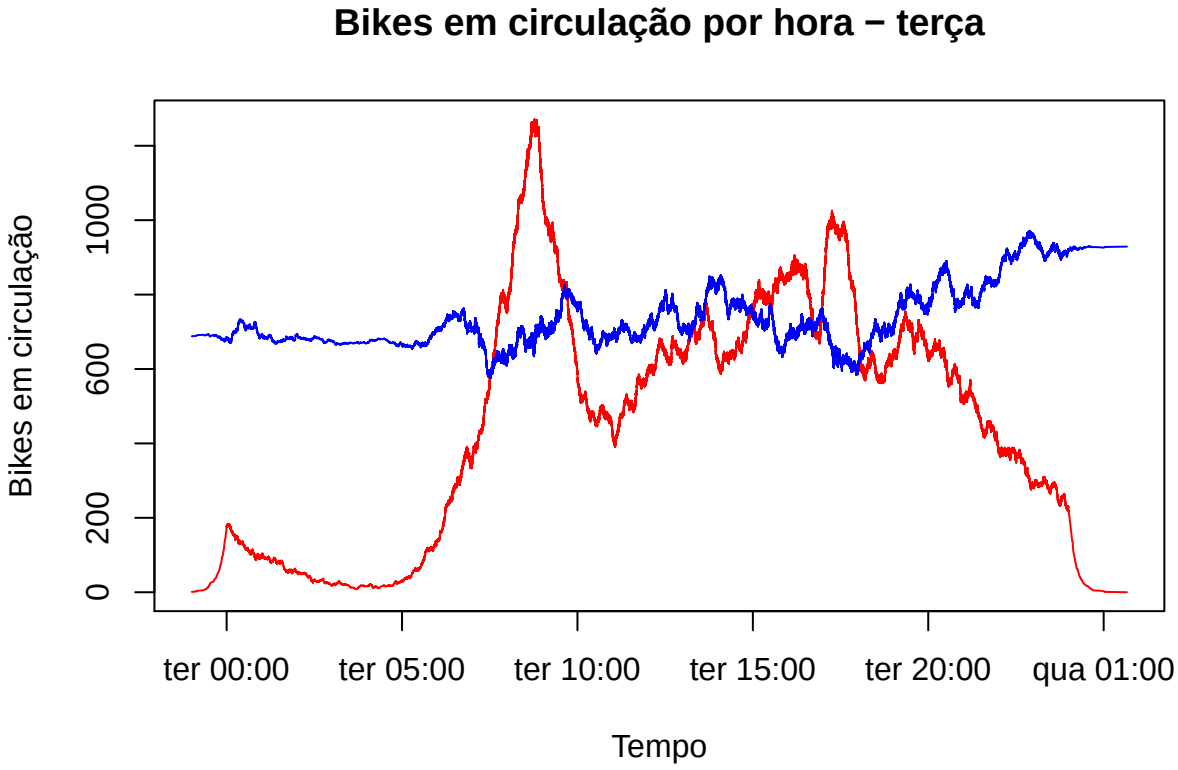
Aqui, apresentamos a função que trata os dados que temos até agora e conta as bicicletas em circulação. O tratamento dos dados é feito criando primeiro duas tabelas, umas com os começos de viagem e outra com os finais. Cada tabela recebe um vetor que identifica se é uma chegada ou retirada: menos uma bicicleta em circulação, ou mais uma, respectivamente(variável start/stop“). Antes de juntar as duas tabelas, é necessário colocar o mesmo nome para todas as colunas. No caso, só foi necessário alterar o nome da primeira coluna. Após juntar os dados, basta ordená-los em relação à coluna”time“, e teremos a ordem de chegada e partida das bikes. Para contar quantas bikes estão circulando, foi criado um loop que soma a variável”start/stop" à quantidade obtida na soma anterior(variável “summation”)
O caráter de análise deste relatório tem como principal elemento o trecho do código que funciona como simulador do comportamento esperado. Apesar de termos um Processo de Poisson para as retiradas e devoluções, e apesar de termos uma distribuição exponencial para o intervalo entre as movimentações de bikes, descobrimos que o fator chave para o crescimento ou decrescimento da função é a proporção entre as retiradas e devoluções. Explicaremos adiante, com os dados.

```
count_bikes <- function(#dataset com os dados
                        bikedataset,
                        #lógico para dados já no formato ideal
                        dataistidy = TRUE,
                        #lógico para simular comportamento esperado dos dados
                        simulate_expected = FALSE){
```

```
  if(!dataistidy){

      #select only the starts
      starts <- bikedataset %>%
      select(starttime, usertype) %>%
      # set a variable that sums 1(plus one bike running)
      mutate("start/stop" = 1)
      #set the same time variable name to bind the tables
      names(starts)[1] <- "time"

      #select only the stops
      stops <- bikedataset %>%
      select(stoptime, usertype) %>%
      # set a variable that sums -1(minus one bike running)
      mutate("start/stop" = -1)
      #set the same time variable name to bind the tables
      names(stops)[1] <- "time"

      #bind the tables
      bikedataset <- rbind(starts, stops)
      #sort the table by time
      bikedataset <- bikedataset[order(bikedataset$time),]
  }

  #sum the running and stopping bikes in the order they happen
  #set the first value
  bikedataset$summation[1] <- bikedataset$`start/stop`[1]
  for(i in 2:nrow(bikedataset)){
    #loop that sums the line value with its previous value
    bikedataset$summation[i] <- bikedataset$`start/stop`[i] +
  }
  #simulating an expected behavior
  if(simulate_expected){
    #set the first value as the mean of the original data
    reference <- mean(bikedataset$summation)
    #a vector of a random walk is what is expected for the starts and stops
    bikedataset$expected <- sample(c(1, -1),
                            size = nrow(bikedataset),
                            replace = TRUE,
                            prob = c(0.5, 0.5))#here is where the magic happens
    #initializing a vector for the expected sum of bikes
    bikedataset <- bikedataset %>% mutate(expected_sum = 0)
    #set the first value
    bikedataset$expected_sum[1] <- reference
    for(i in 2:nrow(bikedataset)){
      #loop that sums the line value with its previous value
      bikedataset$expected_sum[i] <- bikedataset$expected[i] +
                                     bikedataset$expected_sum[i-1]
    }
  }
  #return the new table
  return(bikedataset)
}
```

#Conclusão #Comportamento da função N(t) = Número de Bicicletas em Circulação Enquanto tentávamos criar o simulador da função, notamos que o número de bicicletas retiradas e devolvidas era o mesmo. Deduzimos então que a probabilidade de ser retirada uma bike era a mesma de ser devolvida. Por isso que o simulador toma como valor de início a média de bicicletas em circulação e usa um passeio aleatório(random walk) para definir as retiradas e devoluções. Se a probabilidade é a mesma, e queremos testar a homogeneidade da função, devemos ter uma variável que oscila em torno de uma média. E aí é que entra o detalhe: a homogeneidade tem pouco a ver com as taxas da Poisson ou da Exponencial. Ela depende na verdade da ordem de retiradas e devoluções, que pode ser expressa num intervalo como a proporção entre uma e outra. Por mais que, no final do dia, todas as bicicletas retiradas sejam devolvidas, se num intervalo de tempo temos mais retiradas do que devoluções a quantidade em circulação aumenta. Invertendo as proporções, a quantidade diminui. O efeito da Exponencial-Poisson na verdade é de potencializar a diferença. Observe abaixo que no período entre 0h e 6h, as devoluções têm grande vantagem. Mesmo assim a velocidade de decaimento é baixa, porque os intervalos de tempo entre as movimentações são maiores. Já no período de alteração mais drástica da função, as retiradas ganham por pouco, mas o volume de movimentações multiplica essa diferença. Portanto, gostaria de destacar aqui como o comportamento da simulação homogênea fica mais errático nos períodos mais movimentados.

```
tuesday <- dadosbike %>%
  filter(weekday == 3 | wday(starttime) == 3) %>%
  select(starttime, stoptime, usertype)
tuesday <- count_bikes(tuesday,
                       FALSE,
                       TRUE)
plot(tuesday$time,
     tuesday$summation,
     type="l",
     col="red",
     main = "Bikes em circulação por hora - terça",
     xlab = "Tempo",
     ylab = "Bikes em circulação")
lines(tuesday$time,tuesday$expected_sum,col="blue")
```




```
summary(tuesday$summation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     565     678     688     836    1271
```

```
#8oh to 6h from tuesday summary
summary(tuesday[hour(tuesday$time) >= 0 & hour(tuesday$time) < 6,]$summation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   56.00   93.00   91.94  122.00  216.00
```

```
counting <- tuesday %>%
  filter(hour(tuesday$time) >= 0 & hour(tuesday$time) < 6) %>%
  count(`start/stop` == 1)
counting
```

```
## # A tibble: 2 x 2
##   `\`start/stop\` == 1`     n
##   <lgl>                 <int>
## 1 FALSE                  1821
## 2 TRUE                   1565
```

```
#stop/start proportion
counting$n/sum(counting$n)
```

```
## [1] 0.5378027 0.4621973
```

```
#6h to 12h from tuesday summary
summary(tuesday[hour(tuesday$time) >= 6 & hour(tuesday$time) < 12,]$summation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     136     494     761     751     992    1271
```

```r
counting <- tuesday %>%
  filter(hour(tuesday$time) >= 6 & hour(tuesday$time) < 12) %>%
  count(`start/stop` == 1)
counting
```

```
## # A tibble: 2 x 2
##   `\`start/stop\` == 1`     n
##   <lgl>                 <int>
## 1 FALSE                 18220
## 2 TRUE                  18675
```

```r
#stop/start proportion
counting$n/sum(counting$n)
```

```
## [1] 0.4938339 0.5061661
```

```r
#12h to 18h from tuesday summary
summary(tuesday[hour(tuesday$time) >= 12 & hour(tuesday$time) < 18,]$summation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     580     669     759     770     852    1026
```

```r
counting <- tuesday %>%
  filter(hour(tuesday$time) >= 12 & hour(tuesday$time) < 18) %>%
  count(`start/stop` == 1)
counting
```

```
## # A tibble: 2 x 2
##   `\`start/stop\` == 1`     n
##   <lgl>                 <int>
## 1 FALSE                 21083
## 2 TRUE                  21176
```

```r
#stop/start proportion
counting$n/sum(counting$n)
```

```
## [1] 0.4988996 0.5011004
```

```r
#18h to 24h from tuesday summary
summary(tuesday[hour(tuesday$time) >= 18 & hour(tuesday$time) < 24,]$summation)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   443.0   596.0   549.7   651.0   754.0
```

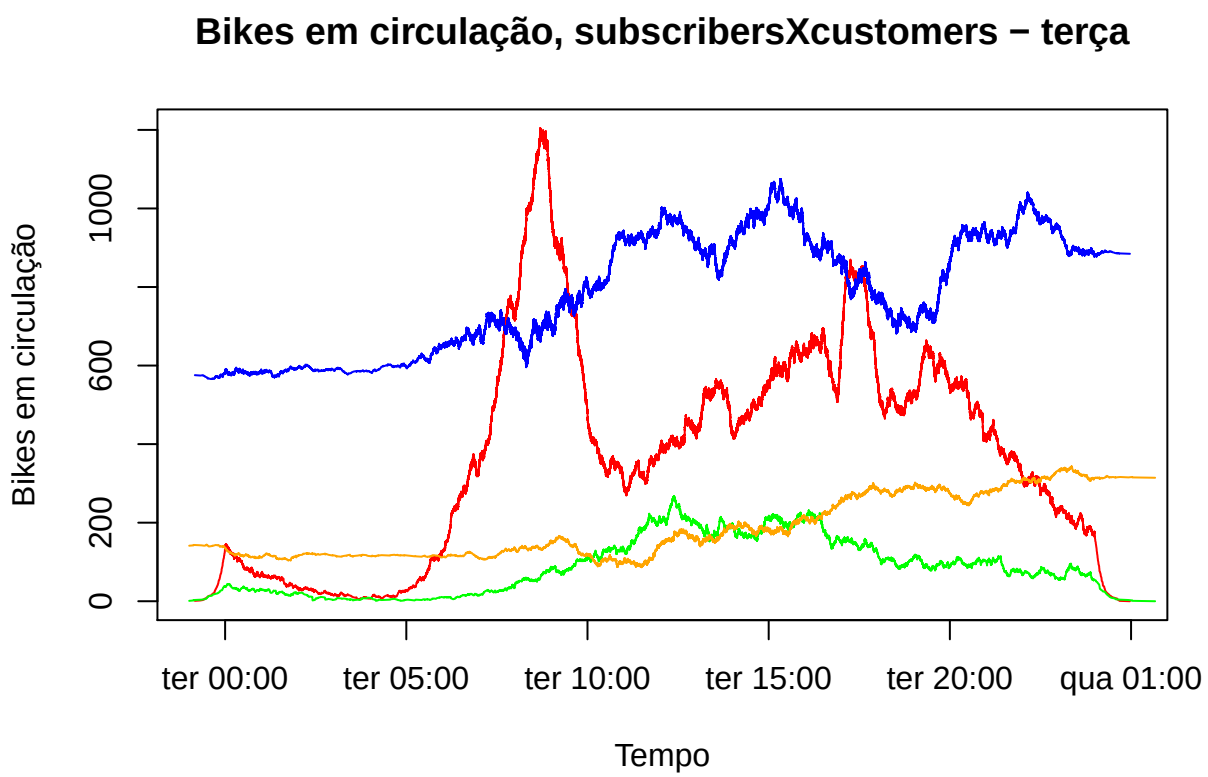
```r
counting <- tuesday %>%
  filter(hour(tuesday$time) >= 18 & hour(tuesday$time) < 24) %>%
  count(`start/stop` == 1)
counting
```

```
## # A tibble: 2 x 2
##   `\`start/stop\` == 1`     n
##   <lgl>                 <int>
## 1 FALSE                 13778
## 2 TRUE                  13486
```

```r
#stop/start proportion
counting$n/sum(counting$n)
```

```
## [1] 0.505355 0.494645
```

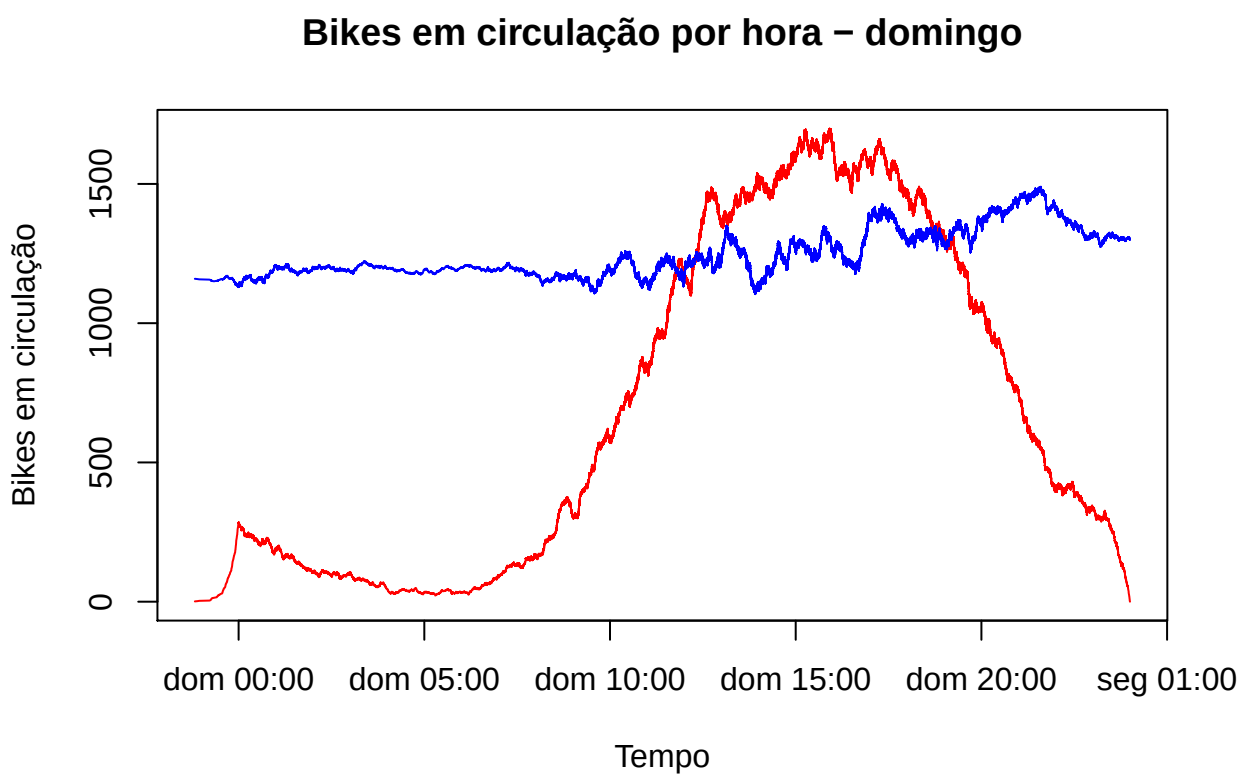
```r
tuesday_subscriber <- tuesday %>%
  filter(usertype == "Subscriber") %>%
  mutate(usertype = NULL)
tuesday_subscriber <- count_bikes(tuesday_subscriber,
                                  simulate_expected = TRUE)
tuesday_customer <- tuesday %>%
  filter(usertype == "Customer") %>%
  mutate(usertype = NULL)
tuesday_customer <- count_bikes(tuesday_customer,
                                simulate_expected = TRUE)
plot(tuesday_subscriber$time,
     tuesday_subscriber$summation,
     type="l",
     col="red",
     main = "Bikes em circulação, subscribersXcustomers - terça",
     xlab = "Tempo",
     ylab = "Bikes em circulação")
lines(tuesday_subscriber$time,tuesday_subscriber$expected_sum,col="blue")
lines(tuesday_customer$time,tuesday_customer$summation,col="green")
lines(tuesday_customer$time,tuesday_customer$expected_sum,col="orange")
```



Por último, os dados de domingo comparados aos de sábado. O comportamento da função N(t) não é homogêneo nem durante o dia, nem durante a semana, e não é similar entre os "usertypes".

```r
sunday <- dadosbike %>%
  filter((weekday == 1 | wday(starttime) == 1) & wday(stoptime) != 2) %>%
  select(starttime, stoptime, usertype)
sunday <- count_bikes(sunday,
                      FALSE,
                      TRUE)
```

```
plot(sunday$time,
     sunday$summation,
     type="l",
     col="red",
     main = "Bikes em circulação por hora - domingo",
     xlab = "Tempo",
     ylab = "Bikes em circulação")
lines(sunday$time,sunday$expected_sum,col="blue")
lines(tuesday$time,tuesday$summation,col="green")
lines(tuesday$time,tuesday$expected_sum,col="orange")
```




```
sunday_subscriber <- sunday %>%
  filter(usertype == "Subscriber") %>%
  mutate(usertype = NULL)
sunday_subscriber <- count_bikes(sunday_subscriber,
                                 simulate_expected = TRUE)
sunday_customer <- sunday %>%
  filter(usertype == "Customer") %>%
  mutate(usertype = NULL)
sunday_customer <- count_bikes(sunday_customer,
                               simulate_expected = TRUE)
plot(sunday_subscriber$time,
     sunday_subscriber$summation,
     type="l",
     col="red",
     main = "Bikes em circulação, subscribersXcustomers - domingo",
     xlab = "Tempo",
     ylab = "Bikes em circulação")
lines(sunday_subscriber$time,sunday_subscriber$expected_sum,col="blue")
```

```
lines(sunday_customer$time,sunday_customer$summation,col="green")
lines(sunday_customer$time,sunday_customer$expected_sum,col="orange")
```