EP1: Sistema de Atendimento com Clientes Impacientes

Erica Mayumi Kanashiro nUSP: 9761698, Eder Gabriel da Trindade Félix nUSP:9778515

10/6/2019

Subproblema 1

Programa para simular um sistema de atendimento com clientes impaciantes. Descriçãohttp://www.each.usp.br/lauretto/ACH2138_2019.

Considerando o cenário $T = 50, n = 5, \lambda = 3, \mu = 0.5$ a implementação incrementa N em passos de 100 (começando com N = 100) e para quando a amplitude do intervalo de 95% de confiança para W for menor do que 0.005 (ou seja, | LimiteSuperior − LimiteInferior | = 2 × 1.96 × se W < 0 .005), onde se W = sd (W) / √ N é erro padrão de W.

O exercício propõe a simular um atendimento de guichê com uma fila de espera, onde clientes impacientes permanecem ou saem da filas. Gerando a cada iteração de N=20000 um valores para: (a)n=5,lambda=3, mi=0.5 (b)n=4,=3, =0.5 Para o funcionamento do ep, que simula o atendimento de guiches com clientes impacientes que podem desistir de esperar na fila, dependendo do tamanho da fila de espera, que cresce quando os guiches estão ocupados e mais clientes chegam. As seguintes variáveis foram utilizadas. Diversos trechos seguem a orientação dada pelo enunciado e materiais fornecidos pelo professor. As seguintes variáveis foram utilizadas.

```
InterTempo<-50 # Intervalo de tempo total sobre o qual se deseja calcular as médias de aceitações e rej
N<-20000 #Numero de repetições da simulação
n<-5 # número de linhas ou guiches
lambda<-3 # modela a taxa de entrada de clientes ( ), parâmetro da distribuição exponencial
mi<-0.5 #taxa de atendimentos a clientes por cada guichê (th: tempo de atendimento de cada linha)
```

vetores guardando resultados

```
TM<-rep(0,N) # Tempo de Espera
R<-rep(0,N) # tamanho da fila
X<-rep(0,N) # Requisições atendidas
Y<-rep(0,N) # Requisições rejeitadas
W<-rep(0,N) #Proportção Requisições Rejeitadas
```

Algoritmo Geral

```
#Calculando todas as N repetições
for(i in 1:N){
Tr<-0 #instante de chegada do último cliente até o momento, Inicialmente tr<-0
k<-0 # k: contador de clientes que entraram na fila até o momento;
ctcheg<-c(0,(N/10))#vetor de tamanho variável em que ctcheg[k]>0 denota o instante em que o k-ésimo cli
x<-0 #contador de clientes já atendidos; inicialmente, x=0
y<-0 #contador de clientes que forma embora sem entrar na fila; inicialmente y = 0
r<-0 #comprimento atual da fila; inicialmente, r=0
w<-0 # proporção de clientes que foram embora:  w=y/(x+y+r)
tm<-0 #tempo máximo de permanência dentre todos os clientes atendidos até o momento
guiche<-rep(0,n) # Iniciando os guiches disponíveis

#Calculando as variações em T
while(Tr<= InterTempo){
  z<-rexp(1, lambda) # Variância de atendimento com poisson
```

```
  Tr<-Tr+z #Tempo do cliente mais a variavel aleatoria de tempo gerada
  k<-k+1 #cliente entra na fila
  if (k>length(ctcheg)) {
    ctcheg = c(ctcheg, rep(0,1))
  }
  ctcheg[k] <- Tr #

  while (min(guiche)<=Tr & x<k) {
    #enquanto houver mais clientes chegando que atendidos
    # e enquanto houver um guiche com menos tempo requerido que o tempo de chegada
    guicheAtendente = which.min(guiche)
    taxaAtend = rexp(1, mi)
    x = x+1
    guiche[guicheAtendente] = max(c(guiche[guicheAtendente],ctcheg[x])) + taxaAtend # descrito enunciad
    tm = max(c(tm, guiche[guicheAtendente]-ctcheg[x])) # descrito enunciado do ep
  }

  # Comprimento atual da fila (precisa descontar o cliente que acabou de chegar)
  r = max(c(0, (k-1)-x))
    #não achou guiche
    # Comprimento atual da fila (precisa descontar o cliente que acabou de chegar)
    r<-max(c(0, (k-1)-x))  #atribuindo r = max{0, (k - 1) - x} // não considera o último cliente
    #Gerando atributo para verificar se o cliente irá embora
    pr<-r/(r+n)
    s<-rbinom(1,1,pr) #s  Ber (pr) onde pr = r /(r + n)
    if (s==1) {
      k<-k-1 #Se s = 1, atribua k = k - 1; y = y + 1
      y<-y+1
    }
    r<-k-x
}
R[i] <-r
TM[i]<-tm
X[i] <-x
Y[i] <-y
W[i] <-y/(x+y+r)

}
```

Os resultados:

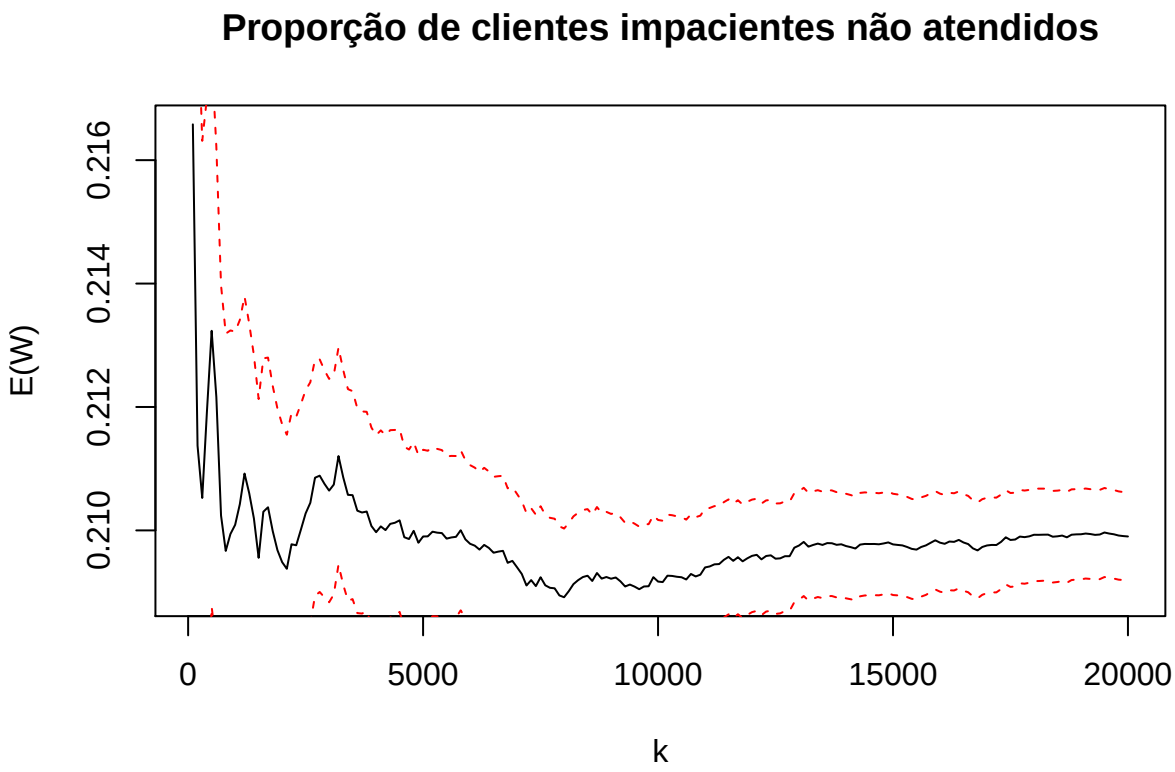
```
#Calcular a média parcial para cada k
centos<-N/100 # quantos blocos de 100 iterações
mparcialW<-rep(0,centos) # array com media amostral de 100 iterações não atendidos
mparcialAtend<-rep(0,centos) # array com media amostral de 100 iterações atendidos
mIntConf<-rep(0,centos) # array com media amostral de 100 iterações atendidos
mTM<-rep(0,centos)
sW <- 0 #desvio padrao
sTM <- 0 #desvio padrao tempo máximo
K<-rep(0,centos) # array com media amostral de 100 iterações
LIw<-rep(0,centos)
LSw<-rep(0,centos)
LItm<-rep(0,centos)
LStm<-rep(0,centos)
```

```
errorTM <-0
totalsw<-rep(0,centos)

for(j in 1:centos){
 K[j]<-j*100
 sW<-sd(W[1: K[j] ])
 totalsw[j]<-sW
 mparcialW[j]<-sum(W[1: K[j]])/ K[j]
 mparcialAtend[j]<-sum(TM[1: K[j]])/ K[j]
 #error <- qnorm(0.975)*sW/sqrt(K[j])
 error <- qt(0.95,df=K[j]-1)*sW/sqrt(K[j]) #Defininto o intervalo de confiança para W
 mIntConf[j]<-error
 LIw[j]<-mparcialW[j]-error
 LSw[j]<-mparcialW[j]+error

 sTM<-sd(TM[1: K[j] ])
 mTM[j]<-sum(TM[1: K[j]])/ K[j]
 errorTM <-  qt(0.95,df=K[j]-1)*sTM/sqrt(K[j]) #Defininto o intervalo de confiança para TM
 LItm[j]<-mTM[j]-errorTM
 LStm[j]<-mTM[j]+errorTM
}
```

1:

Um gráfico de linha no qual, para cada k {100, 200, . . . , N }, sejam apresentadas as médias parciais e seus respectivos intevalos de 95% de confiança para Wk

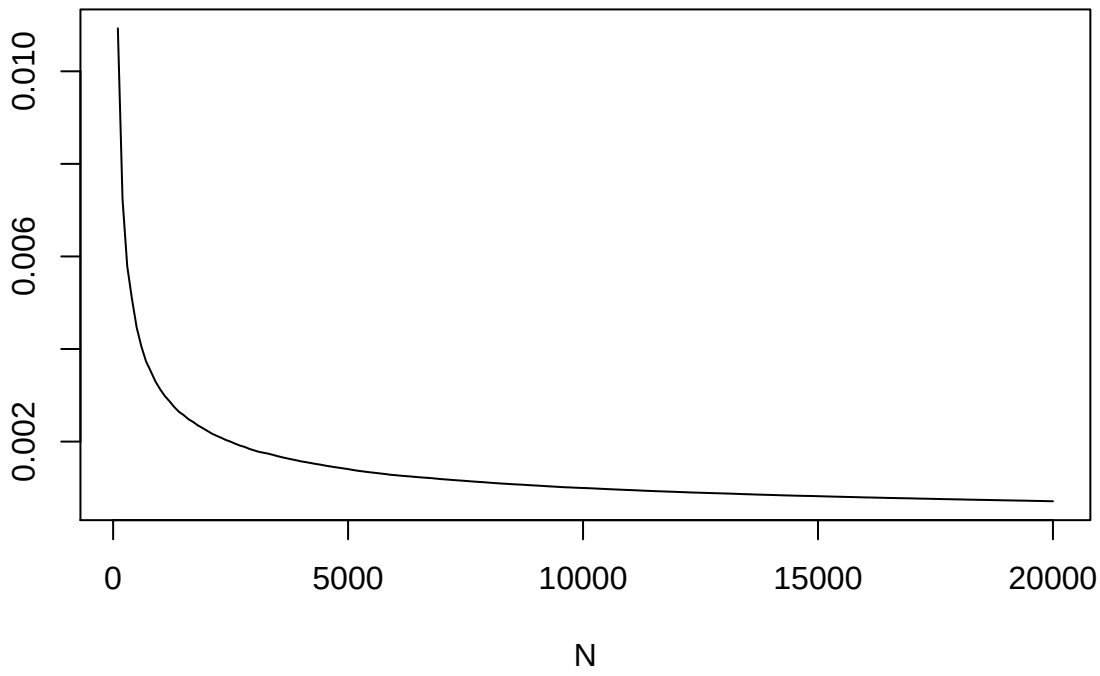
```
#imprimir gráfico de linha para cada k c
plot(K, mparcialW, type='l', xlab='k', ylab='E(W)', main='Proporção de clientes impacientes não atendid
lines(K,LIw, col="red", type = "l",lty=2)
lines(K,LSw, col="red", type = "l",lty=2)
```



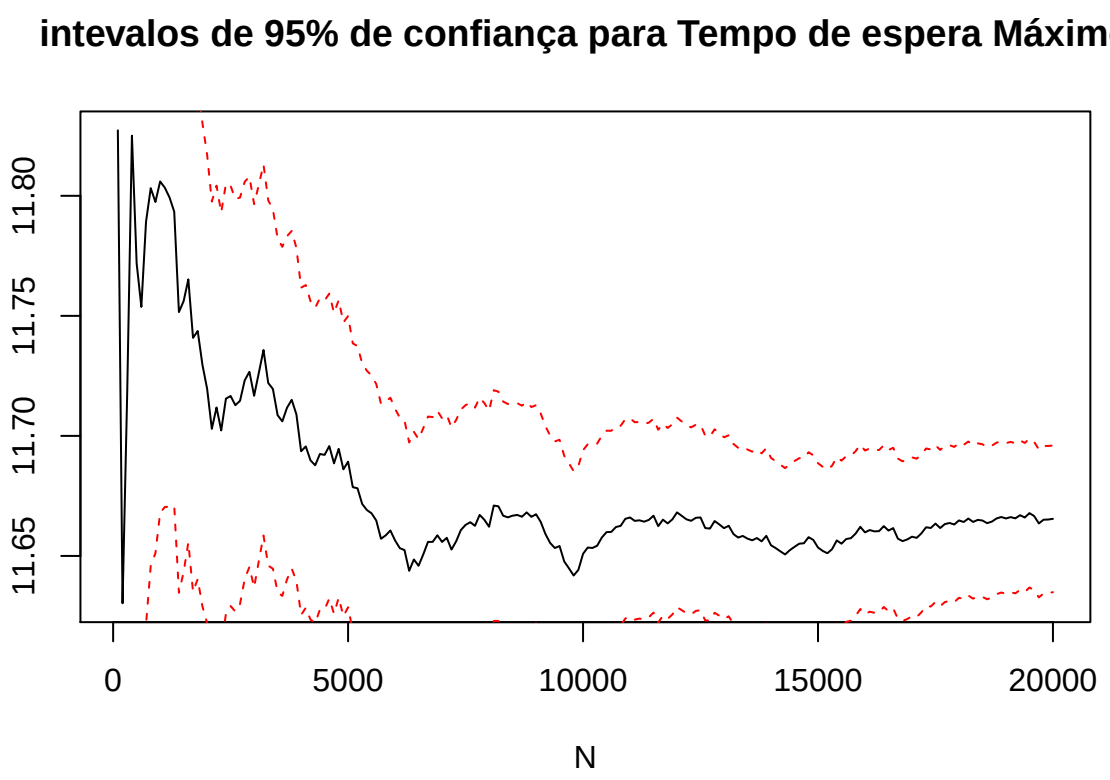
2:

Um gráfico de linha no qual, para cada k {100, 200, . . . , N }, sejam apresentadas as médias parciais e seus respectivos intevalos de 95% de confiança para TMk

```
plot(K, mIntConf, type='l', xlab='N', ylab='', main='amplitude do intervalo de 95% de confiança para W'
```




```
plot(K, mTM, type='l', xlab='N', ylab='', main='intevalos de 95% de confiança para Tempo de espera Máxi
lines(K,LItm, col="red", type = "l",lty=2)
lines(K,LStm, col="red", type = "l",lty=2)
```



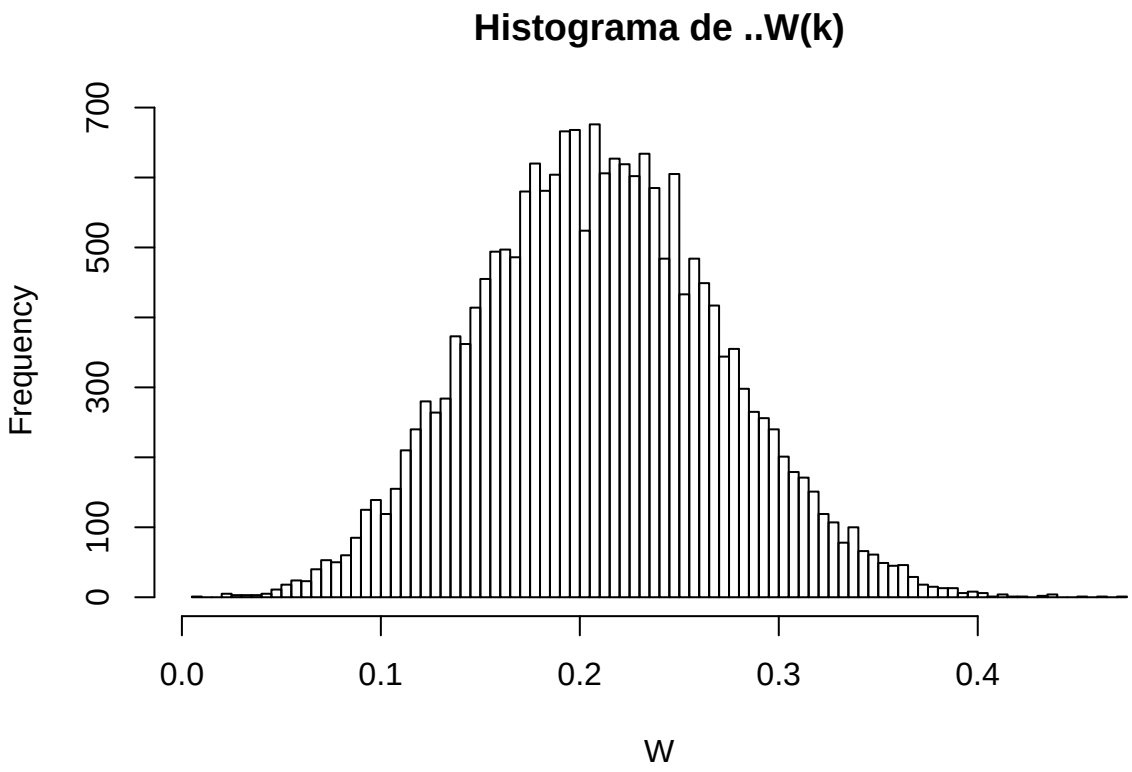
3:

Os histogramas de W e de tm obtidos nas N iterações

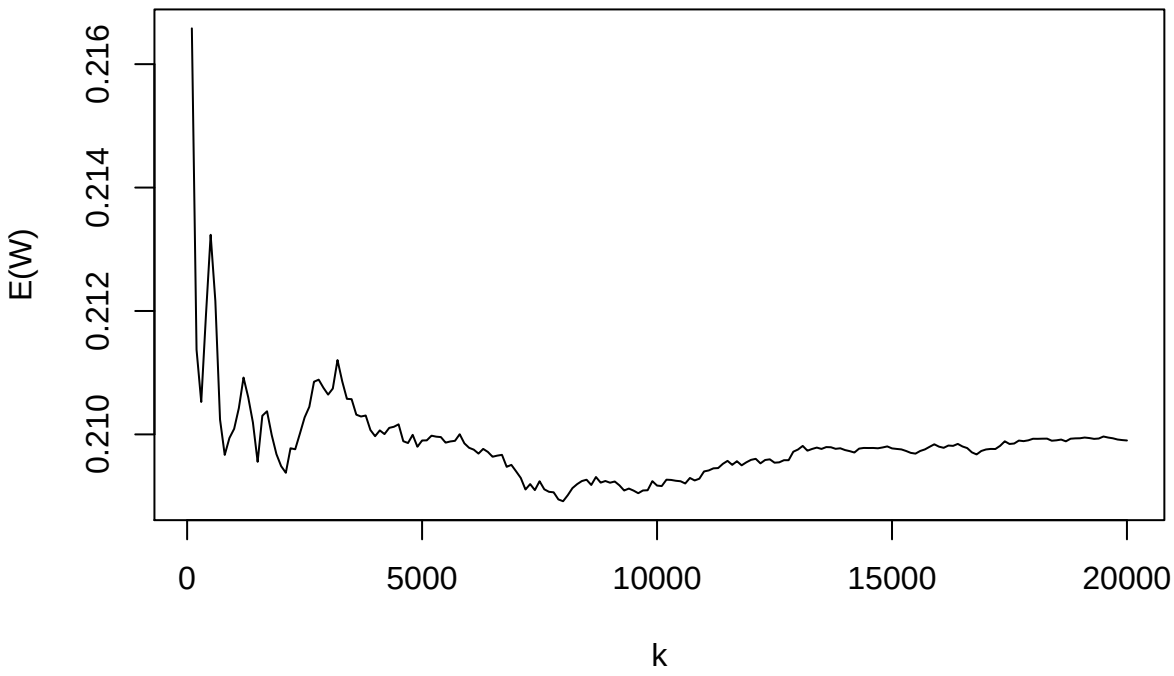
```
#imprimir o histograma de W
hist(W,main='Histograma de  W(k)',90)
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## conversion failure on 'Histograma de  W(k)' in 'mbcsToSbcs': dot
## substituted for <ce>
```

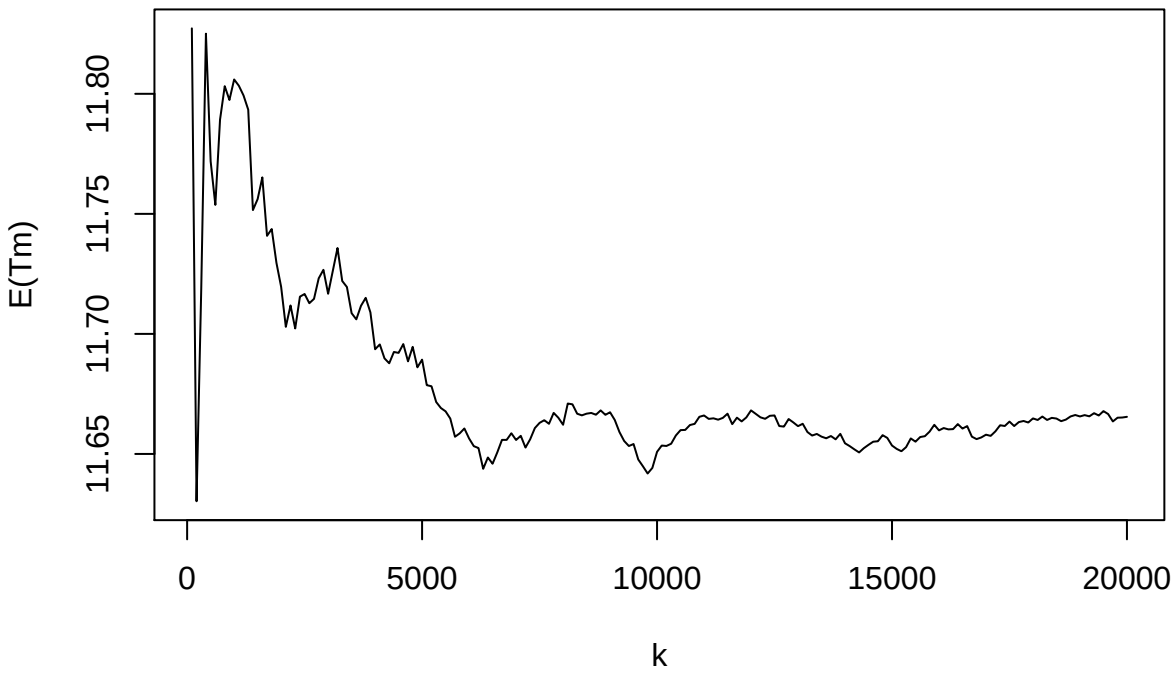
```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## conversion failure on 'Histograma de  W(k)' in 'mbcsToSbcs': dot
## substituted for <bc>
```




```
plot(K, mparcialW, type='l', xlab='k', ylab='E(W)', main='', ylim=c(min(mparcialW), max(mparcialW)))
```



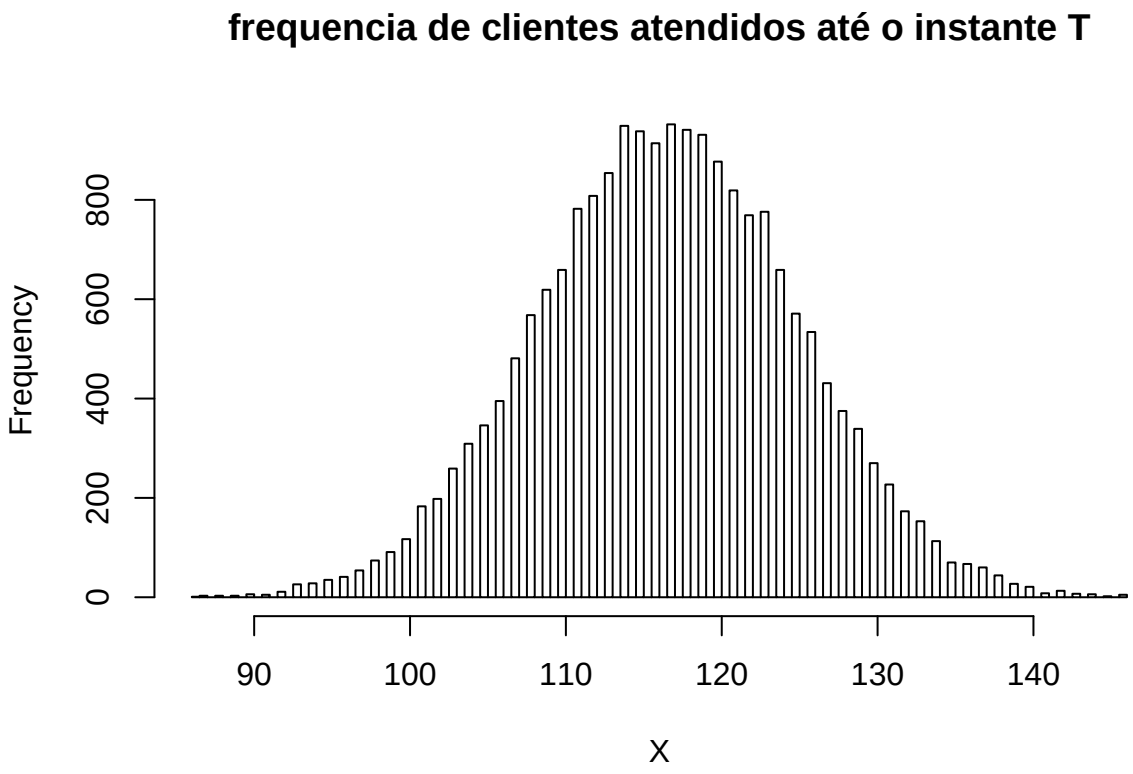

```
plot(K, mparcialAtend, type='l', xlab='k', ylab='E(Tm)', main='', ylim=c(min(mparcialAtend), max(mparcia
```



4:

Imprimir as médias finais Xk , Yk , Wk , TMk nas N iterações

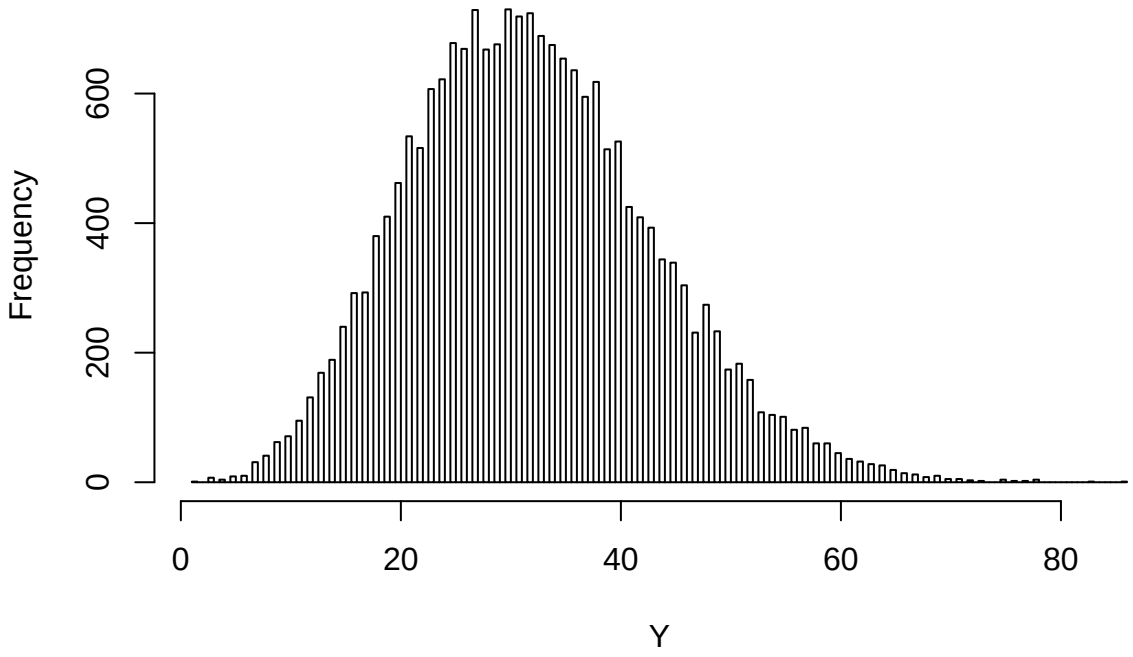
```
hist(X,150, main='frequencia de clientes atendidos até o instante T') #ilustra a frequencia de clientes
```




```
print(mean(X))
```

```
## [1] 116.7484
```

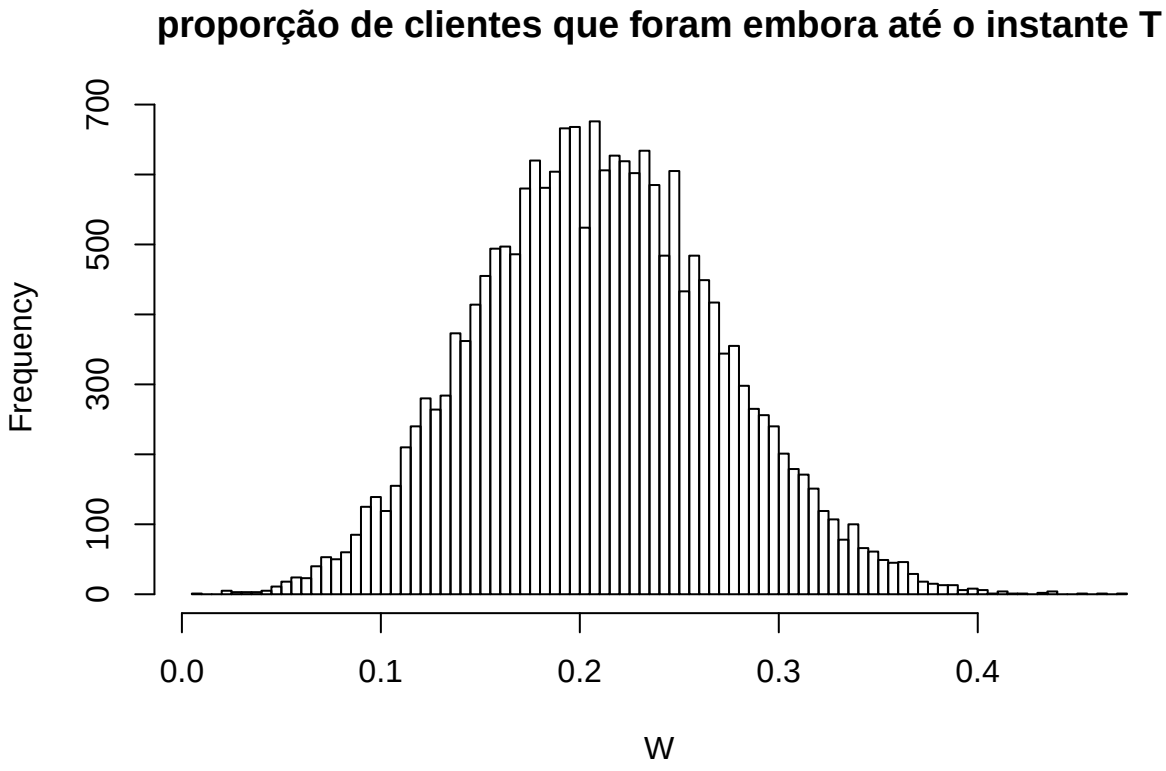
```
hist(Y,150, main='frequencia de clientes que foram embora até o instante T') #ilustra a frequencia de c
```




```
print(mean(Y))
```

```
## [1] 32.1597
```

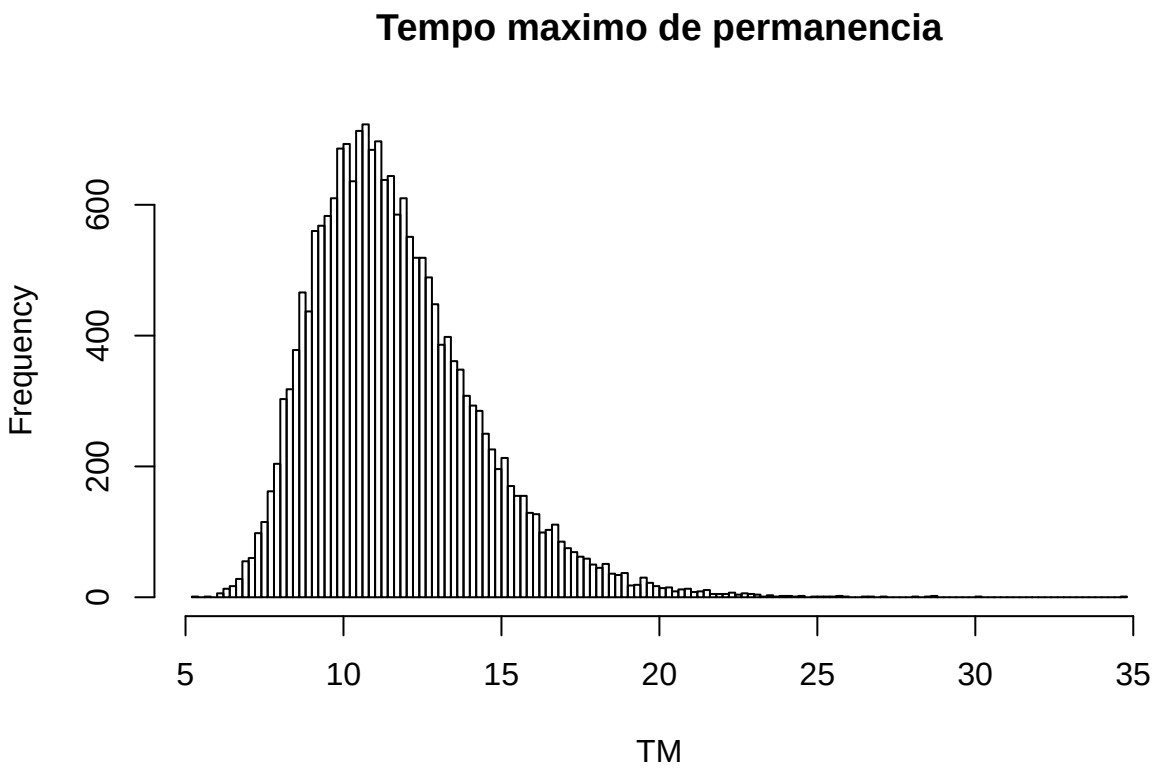
```
hist(W,150, main='proporção de clientes que foram embora até o instante T') #ilustra a frequencia da pr
```




```
print(mean(W))
```

```
## [1] 0.2099013
```

```
hist(TM,150, main='Tempo maximo de permanencia') #ilustra a frequencia do tempo maximo de permanencia d
```




```
print(mean(TM))
```

```
## [1] 11.66546
```

```
#Outros
#hist(R, main='Tamanho da fila até o instante T') #ilustra a frequencia de do tamnho da fila até o
```

5:

Imprimir Pr(tm > 13)

```
#Calculando a probabilidade de TM > 13
pacientes <- sum(TM>13)
pp <-pacientes/sum(X)#dado que proporção de clientes atendidos
sprintf("%1.4f%%", pp)
```

```
## [1] "0.0022%"
```

6:

Imprimir o valor de w s para o qual Pr(w > w s) < 5%. Note que esse valor corresponde ao quantil 0.95 dos valores simulados de w.

```
ws <- sum(W>mean(W)+error) + sum(W<mean(W)-error)
total <-sum(X)+sum(Y)
wsr <-ws/total
#6
sprintf("%1.4f%%", wsr)
```

```
## [1] "0.0067%"
```

SUBPROBLEMA 2

Baseado na informação da média, se dependesse somente da variável de quantidade de guiches uma solução simples seria calcular quanto cada guiche consegue atender no tempo T para preencher o intervalo e manter a taxa W menor que 20%.

```
gMax = 10
TempMax<-rep(0,gMax)
Atendidos<-rep(0,gMax)
Natendidos <-rep(0,gMax)
PropNAtend<-rep(0,gMax)
limPropNAtend<-rep(0,gMax)
nGuiches<-rep(0,gMax)

for(g in 1:gMax){

# Simulacao de Sistema de atendimento com clientes impacientes
InterTempo<-60 # Intervalo de tempo total sobre o qual se deseja calcular as médias de aceitações e rej
N<-20000 #Numero de repetições da simulação
n<-g # número de guiches
lambda<-3 # para taxa de entrada de clientes
mi<-0.5 #taxa de atendimentos a clientes por cada guichê

#vetores guardando resultados
TM<-rep(0,N)
R<-rep(0,N)
X<-rep(0,N)
Y<-rep(0,N)
W<-rep(0,N)

#Calculando todas as N repetições
for(i in 1:N){
 Tc<-0 #instante de chegada do último cliente até o momento, Inicialmente tc<-0
 k<-0 # k: contador de clientes que entraram na fila até o momento;
 ctcheg<-c(0,(N/10))#vetor de tamanho variável em que ctcheg[k]>0 denota o instante em que o k-ésimo cl
 x<-0 #contador de clientes já atendidos; inicialmente, x=0
 y<-0 #contador de clientes que forma embora sem entrar na fila; inicialmente y = 0
 r<-0 #comprimento atual da fila; inicialmente, r=0
 w<-0 # proporção de clientes que foram embora:  w=y/(x+y+r)
 tm<-0 #tempo máximo de permanência dentre todos os clientes atendidos até o momento
 guiche<-rep(0,n) # Iniciando os guiches disponíveis

 #Calculando as variações em T
 while(Tc<= InterTempo){
   z<-rexp(1, lambda) # Variância de atendimento com poisson
   Tc<-Tc+z #Tempo do cliente mais a variavel aleatoria de tempo gerada
   k<-k+1 #cliente entra na fila
   if (k>length(ctcheg)) {
     ctcheg = c(ctcheg, rep(0,1))
   }
   ctcheg[k] <- Tc #

   while (min(guiche)<=Tc & x<k) {
```

```
    #enquanto houver mais clientes chegando que atendidos
    # e enquanto houver um guiche com menos tempo requerido que o tempo de chegada
    guicheAtendente = which.min(guiche)
    taxaAtend = rexp(1, mi)
    x = x+1
    guiche[guicheAtendente] = max(c(guiche[guicheAtendente],ctcheg[x])) + taxaAtend # descrito enuncia
    tm = max(c(tm, guiche[guicheAtendente]-ctcheg[x])) # descrito enunciado do ep
  }

  # Comprimento atual da fila (precisa descontar o cliente que acabou de chegar)
  r = max(c(0, (k-1)-x))
    #não achou guiche
    # Comprimento atual da fila (precisa descontar o cliente que acabou de chegar)
    r<-max(c(0, (k-1)-x))  #atribuindo r = max{0, (k - 1) - x} // não considera o último cliente
    #Gerando atributo para verificar se o cliente irá embora
    pr<-r/(r+n)
    s<-rbinom(1,1,pr) #s  Ber (pr) onde pr = r /(r + n)
    if (s==1) {
      k<-k-1 #Se s = 1, atribua k = k - 1; y = y + 1
      y<-y+1
    }
    r<-k-x

 }
 R[i] <-r
 TM[i]<-tm
 X[i] <-x
 Y[i] <-y
 W[i] <-y/(x+y+r)

 }

#Calcular a média parcial para cada k
centos<-N/100 # quantos blocos de 100 iterações
mparcialW<-rep(0,centos) # array com media amostral de 100 iterações não atendidos
mparcialAtend<-rep(0,centos) # array com media amostral de 100 iterações atendidos
mIntConf<-rep(0,centos) # array com media amostral de 100 iterações atendidos
mTM<-rep(0,centos)
sW <- 0 #desvio padrao
sTM <- 0 #desvio padrao tempo máximo
K<-rep(0,centos) # array com media amostral de 100 iterações
LIw<-rep(0,centos)
LSw<-rep(0,centos)
LItm<-rep(0,centos)
LStm<-rep(0,centos)
errorTM <-0
totalsw<-rep(0,centos)

for(j in 1:centos){
 K[j]<-j*100
 sW<-sd(W[1: K[j] ])
 totalsw[j]<-sW
 mparcialW[j]<-sum(W[1: K[j]])/ K[j]
```

```
 mparcialAtend[j]<-sum(TM[1: K[j]])/ K[j]
 #error <- qnorm(0.975)*sW/sqrt(K[j])
 error <- qt(0.975,df=K[j]-1)*sW/sqrt(K[j])
 mIntConf[j]<-error
 LIw[j]<-mparcialW[j]-error
 LSw[j]<-mparcialW[j]+error

 sTM<-sd(TM[1: K[j] ])
 mTM[j]<-sum(TM[1: K[j]])/ K[j]
 errorTM <-  qt(0.975,df=K[j]-1)*sTM/sqrt(K[j])
 LItm[j]<-mTM[j]-errorTM
 LStm[j]<-mTM[j]+errorTM
}

nGuiches[g] = g
 TempMax[g] = mean(TM)
 Atendidos[g] = mean(X)
 Natendidos[g] = mean(Y)
 PropNAtend[g] = mean(W)
 limPropNAtend[g] = error

}

B<- rbind(nGuiches,Atendidos,Natendidos,PropNAtend,limPropNAtend,TempMax)

minA <- which((PropNAtend+limPropNAtend)*100 <= 20)[1]
print(B)
```

```
##                       [,1]         [,2]         [,3]         [,4]
## nGuiches      1.000000e+00 2.000000e+00 3.000000e+00 4.000000e+00
## Atendidos     3.097120e+01 6.148425e+01 9.085035e+01 1.175769e+02
## Natendidos    1.442174e+02 1.144868e+02 8.622795e+01 6.032045e+01
## PropNAtend    7.951485e-01 6.307743e-01 4.738047e-01 3.306475e-01
## limPropNAtend 4.167854e-04 6.435856e-04 7.924279e-04 8.434112e-04
## TempMax       2.095761e+01 1.515917e+01 1.326398e+01 1.242864e+01
##                       [,5]         [,6]         [,7]         [,8]
## nGuiches      5.000000e+00 6.000000e+00 7.000000e+00 8.000000e+00
## Atendidos     1.398844e+02 1.565094e+02 1.677955e+02 1.742361e+02
## Natendidos    3.905100e+01 2.304360e+01 1.251555e+01 6.237150e+00
## PropNAtend    2.131489e-01 1.252443e-01 6.765102e-02 3.359253e-02
## limPropNAtend 7.815060e-04 6.502865e-04 4.797860e-04 3.266301e-04
## TempMax       1.202874e+01 1.180021e+01 1.172190e+01 1.163798e+01
##                       [,9]        [,10]
## nGuiches      9.000000e+00 1.000000e+01
## Atendidos     1.779501e+02 1.796838e+02
## Natendidos    2.884200e+00 1.259850e+00
## PropNAtend    1.545111e-02 6.719811e-03
## limPropNAtend 2.061358e-04 1.277214e-04
## TempMax       1.160576e+01 1.155313e+01
```

Quantidade mínima de atendentes:

Pr(W 20%) 0.95 , utilizando o intervalo de confiança calculado para W de cada simulação de 1 a 10 guiches.

```r
print(minA)
```

```
## [1] 6
```