Simulación ITAM

23 de octubre de 2020 2020-II

Tarea 3. Fecha de entrega: Lunes 12 de octubre de 2020

Lecturas

Continúan las mismas lecturas de la tarea pasada:

- Generating Nonhomogeneous Poisson Process
- Generating a non-homogeneous Poisson Process
- Casella y Robert, capítulo 2
- Dagpunar, Capítulos 3 y 4. Secciones 7.1-7.3

Problemas

1. Para un proceso Poisson no homogéneo con función de intensidad dada por

$$\lambda(t) = \begin{cases} 5, t \in (1,2], (3,4], \cdots \\ 3, t \in (0,1], (2,3], \cdots \end{cases}$$

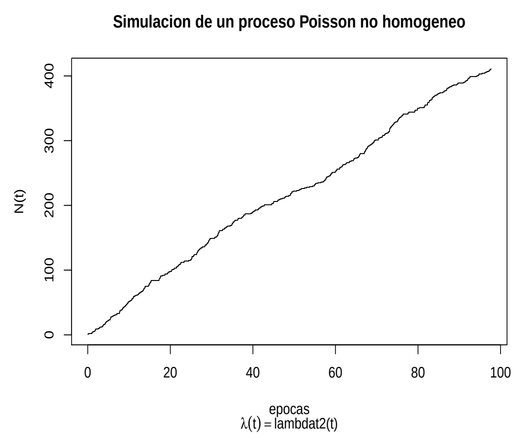
   a) Grafiquen una ejemplo del proceso considerando el intervalo de tiempo [0,100].
   b) Grafiquen el proceso hasta obtener 100 eventos
   c) Estimen la probabilidad de que el número de eventos observados en el periodo de tiempo (1.25,3] es mayor que 2.

Solución.

Para el primer problema, tenemos que definir la función pulso. Debido a que en el proceso de aceptación-rechazo se pierden algunas observaciones, el numero de observaciones necesarios para llegar al tiempo 100 se obtiene por ensayo y error, o bien, cambiar la programación y primero generar todas las exponenciales hasta que se acumule el número que necesitamos. Yo hice la otra programación.

```
lambdat2 <- function(t){
        x <- paste("","{",0,"<= t & t <",1,"}",sep="")
        for(i in seq(2,100,2)){x <- paste(x,paste("","{",i,"<= t & t <",i+1,"}",sep=""),sep="|")}
        return(ifelse(eval(parse(text=x)),3,5))
        }

lambdat2(1:10)


 [1] 5 3 5 3 5 3 5 3 5 3


poisson.nohomogeneo <- function(lambdat,n,pic=T){

        lambda <- 5           # mayoriza la función lambdat
        TT <- rexp(n,lambda)  # genera variables exponenciales para los tiempos.
        s <- cumsum(TT)       # acumula los tiempos en el vector s
        u <- runif(n)         # obten n uniformes
```

```r
        # obten los tiempos que cumplen la condición de aceptación
        ss <- s[u <= lambdat(s)/lambda]
        Ns <- 1:length(ss) # Conteo
        if(pic==T){
                   plot(ss, Ns, type = "s", xlab = "epocas", ylab = "N(t)",
                                    main = "Simulacion de un proceso Poisson no homogeneo",
                                        sub = expression(lambda(t) == paste(lambdat2,"(t)")))
        }
return(list(epocas = ss, cuenta=  Ns))
}

poisson.nohomogeneo(lambdat2,510)
```

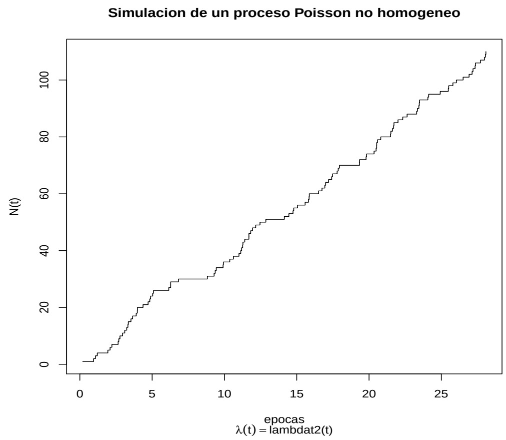



```
$epocas
  [1]  0.002420603  0.221806513  0.982950317  1.062237556  1.276548587
  [6]  1.654384501  1.784998079  1.878072520  1.918405522  2.537884161
 [11]  2.834891335  2.930744982  3.481864996  3.680761005  3.705732364
 [16]  3.826504894  4.194187573  4.197205288  4.343291102  4.435936253
 [21]  4.587741893  4.874965959  5.056068924  5.418140708  5.465558041
 [26]  5.482761575  5.598288946  5.642113323  6.033706836  6.236754431
 [31]  6.597044388  7.043487191  7.079537792  7.673514423  7.731543903
 [36]  7.735075255  7.817282832  7.841635217  8.251298049  8.346261071
 [41]  8.506004484  8.523157284  8.702001022  8.941147258  9.097550613
 [46]  9.214303114  9.388340644  9.516162253  9.635544177  9.797524429
 [51]  9.807087497 10.042739504 10.317486590 10.554337979 10.690364767
 [56] 10.749393348 10.972068935 11.038623303 11.082405517 11.279918778
 [61] 11.588960975 11.978712984 12.284901338 12.372687199 12.503917220
 [66] 12.785803775 13.011055112 13.263350368 13.453452943 13.495846178
 [71] 13.598358698 13.643507129 13.833215082 13.857061798 13.935542250
 [76] 14.677842380 14.823264673 14.843681843 14.919318630 15.011323906
 [81] 15.179707297 15.228761612 15.327173224 15.385985866 17.268543983
 [86] 17.302823324 17.372339849 17.498679269 17.576046781 17.614722202
 [91] 17.669380512 18.129093163 18.670310763 18.725931399 19.291395837
 [96] 19.399536311 19.509686187 19.855363183 20.293502824 20.352602451
[101] 20.402958969 20.924986144 20.944650197 21.472926795 21.504030149
[106] 21.655884581 21.952708783 22.035557956 22.170975885 22.511531307
[111] 22.518128657 22.617170176 23.316914401 23.401056622 24.466851778
[116] 24.889394220 25.084412825 25.138560877 25.230321602 25.245128605
[121] 25.263395793 25.587194806 25.764428360 25.800442564 26.381390806
[126] 26.386410247 26.455562766 26.556810831 26.656603080 26.710052098
[131] 26.840805751 27.001497645 27.179170497 27.253698578 27.626440794
[136] 27.769518474 28.339096207 28.396286805 28.465862781 28.817227521
[141] 28.829126632 28.985689350 29.189997209 29.217446303 29.233951926
[146] 29.388309416 29.458772199 29.509400697 29.707453863 30.712959821
[151] 30.840055120 31.212746972 31.340545343 31.441705446 31.559816863
[156] 31.571688779 31.613297357 31.718719105 31.811418328 31.835377697
[161] 31.872708371 32.649720656 32.670476051 33.080007924 33.127031842
[166] 33.442239450 33.597308207 33.835923068 34.492740531 34.876777850
[171] 34.908373117 35.013445426 35.145650540 35.171580538 35.407788566
[176] 35.508887251 35.703604102 36.328931372 36.332708634 36.495117454
[181] 37.278989292 37.382940328 37.627993110 37.737223450 37.752875590
[186] 38.089353948 38.126194890 39.548811722 39.826369744 39.950321805
[191] 40.128337382 40.506704917 40.572357176 41.217465655 41.395055289
```

```
[196] 41.518153030 41.824979248 41.960443197 42.285280373 42.720705198
[201] 42.812924920 44.507829455 44.636074420 45.006459354 45.032666072
[206] 45.041867935 46.001231852 46.023708805 46.318910627 46.613061126
[211] 47.092711855 47.622078249 47.768499785 47.992588418 48.613734649
[216] 49.011736589 49.098781113 49.103946089 49.284286758 49.436275254
[221] 49.443150863 49.780336885 50.644634618 51.165671167 51.457592984
[226] 51.758235584 52.462088418 53.006882205 53.748765455 54.467199930
[231] 54.886204269 54.901118194 55.131174250 55.205381769 55.741993500
[236] 56.501698815 57.044195952 57.256228045 57.357448704 57.616879031
[241] 57.673982414 57.695928021 57.814475931 57.892943030 58.242638745
[246] 58.605046364 58.758642050 58.876923199 58.939494942 59.174629803
[251] 59.176365283 59.934963946 60.104680276 60.119737665 60.334965302
[256] 60.383458346 61.019819820 61.071631166 61.089568748 61.546499288
[261] 61.630561117 61.724981449 61.815063425 62.377390205 62.620730207
[266] 62.711994052 63.362836831 63.385975801 63.827563999 64.350878008
[271] 64.419599917 64.469502467 64.703646198 65.280936541 65.531881613
[276] 65.699906815 65.758176338 65.845599515 65.887161655 66.011802272
[281] 66.972743992 67.032651388 67.062678025 67.209553847 67.285370326
[286] 67.371281150 67.405653938 67.542892287 67.705836704 67.804941699
[291] 67.806571420 68.021222748 68.317071753 68.496453202 68.815951235
[296] 68.833498686 69.149868867 69.316669119 69.320856391 69.405949950
[301] 69.641249351 70.379608233 70.412400565 70.467451577 70.831629807
[306] 71.271648223 71.410322641 71.410957662 71.829343894 71.990370806
[311] 72.077205033 72.487096682 72.848514490 72.866813219 73.113498853
[316] 73.115997006 73.120963827 73.173653518 73.177588206 73.292466666
[321] 73.356861575 73.587280757 73.667727235 73.755221026 73.958275395
[326] 74.134397852 74.138456518 74.296090537 74.527290685 74.974888764
[331] 75.050308882 75.177905950 75.189107087 75.328547559 75.484746658
[336] 75.518365443 75.865924369 75.949938471 76.183900496 76.359762596
[341] 76.369895730 77.548192372 77.590876941 77.767356701 79.197731812
[346] 79.266931796 79.270227015 79.850541062 79.856211839 79.888063764
[351] 80.330896968 81.473024123 81.611838974 81.668533349 81.679764108
[356] 82.278967279 82.287918626 82.336259931 82.371571785 82.698216996
[361] 82.753134088 82.762656197 82.997457642 83.337996147 83.423589553
[366] 83.426934908 83.527428949 83.736630425 83.957869443 84.141733365
[371] 84.515867469 84.735975858 84.930034395 85.283314719 85.987648070
[376] 86.303423001 86.353233296 86.783790087 86.973756712 86.974840094
[381] 87.043976262 87.416456414 87.588895063 87.950318610 88.253895029
[386] 88.580031114 89.358851074 89.611543803 89.707510558 91.001105387
[391] 91.346244753 91.444945317 91.742714596 91.997709740 92.026359058
[396] 92.174898321 92.314707524 92.563703197 92.583622475 94.205475904
[401] 94.507597378 94.759005513 94.779155833 95.473710223 96.148001704
[406] 96.444782280 96.763777555 97.204645272 97.316839919 97.486774847
[411] 97.655234147

$cuenta
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
[109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
[127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
[145] 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
[163] 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
[181] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
[199] 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
[217] 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
[235] 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
[253] 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
[271] 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
[289] 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
[307] 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
[325] 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
[343] 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
[361] 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
[379] 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
[397] 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411
```

Para obtener 100 eventos es similar, con un tamaño de muestra mucho menor:

```
poisson.nohomogeneo(lambdat2,140)
```




```
$epocas
  [1]  0.2005423  0.9511594  1.0794977  1.2061278  1.9375872  2.0914065
  [7]  2.2163527  2.6479591  2.6956737  2.7737586  2.9546000  3.1065151
 [13]  3.2428909  3.3250419  3.3513202  3.5348276  3.6498412  3.8989173
 [19]  3.9747066  3.9795309  4.3699352  4.7207248  4.8378555  4.8946141
 [25]  5.0386791  5.0955177  6.1502574  6.2813633  6.2816632  6.8178822
 [31]  8.8191510  9.2843041  9.3617960  9.4343818  9.9048131  9.9262679
 [37] 10.3680073 10.6258663 11.0047823 11.1285200 11.1887712 11.2604827
 [43] 11.2776223 11.4040113 11.6903017 11.6916601 11.8153976 11.9418554
 [49] 12.1622170 12.4591817 12.8633502 14.1422709 14.4564469 14.7420167
 [55] 14.7924067 15.0563193 15.5800479 15.8086294 15.8543964 15.8695056
 [61] 16.5081827 16.7460564 16.9424416 17.0010301 17.2028874 17.4178334
 [67] 17.4768717 17.7940281 17.8692224 17.9552102 19.3426810 19.3473924
 [73] 19.7949769 19.8299147 20.3487138 20.4879806 20.5270267 20.5305847
 [79] 20.5875126 20.8094007 21.4849621 21.5079067 21.6311372 21.7019779
 [85] 21.7227288 22.0025906 22.3316096 22.6329835 23.2941675 23.3482343
 [91] 23.4454945 23.4740605 23.4962784 24.0681618 24.1242169 24.9245059
 [97] 25.4821582 25.5090854 25.7987391 26.0406654 26.5076589 26.9203109
[103] 27.1278131 27.1927865 27.3351277 27.3639347 27.7118160 27.9838492
[109] 28.0476244 28.0822211

$cuenta
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
[109] 109 110
```

Por último, para estimar la probabilidad en el intervalo dado, lo que podemos hacer es obtener N simulaciones, y calcular la proporción de esas simulaciones que dan un valor de conteo mayor a 2 en ese intervalo.

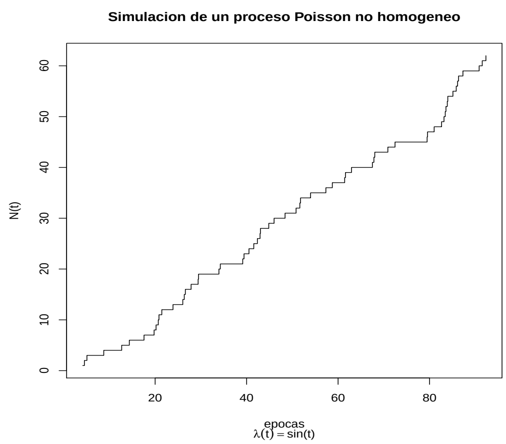
```
fr <- NULL
N <- 100000
for(i in 1:N){
      x <- poisson.nohomogeneo(lambdat2,10,pic=F);
      fr <- c(fr,ifelse(sum(1.25 < x$epocas[x$cuenta>=3] & x$epocas[x$cuenta>=3] < 3)>0,1,0))}
      sum(fr)/N

[1] 0.89354
```

$\square$

2. Simular un proceso Poisson no homogéneo con función de intensidad dada por $\lambda(t) = |sen(t)|$

Solución.

```
lambdat <- function(t) abs(sin(t))

poisson.nohomogeneo <- function(lambdat,n){

        lambda <- 1     # mayoriza la función lambdat
        TT <- rexp(n,lambda) #genera variables exponenciales para los tiempos.
        s <- cumsum(TT) # acumula los tiempos en el vector s
        u <- runif(n)   # obten n uniformes
        # obten los tiempos que cumplen la condición de aceptación
        ss <- s[u <= lambdat(s)/lambda]
        Ns <- 1:length(ss) # Conteo
        plot(ss, Ns, type = "s", xlab = "epocas", ylab = "N(t)",
                 main = "Simulacion de un proceso Poisson no homogeneo",
                 sub = expression(lambda(t) == paste("sin","(t)")))
        return(list(epocas = ss, cuenta=  Ns))
}
x <- poisson.nohomogeneo(lambdat,100)
```




```
x

$epocas
 [1]  4.167201  4.528730  5.077739  8.761156 12.684220 14.349841 17.565485
 [8] 19.771597 20.198533 20.683699 20.836393 21.453048 23.899554 26.041456
[15] 26.366326 26.601997 27.861040 29.398057 29.482458 33.942522 34.244882
[22] 39.146224 39.431486 40.512289 41.561698 42.348702 42.936012 43.021441
[29] 44.839532 46.007074 48.404352 50.812530 51.631582 51.783248 53.980656
[36] 57.326242 58.737831 61.437137 61.613889 62.934383 67.497462 67.827222
[43] 68.018054 70.877662 72.448105 79.465620 79.551360 81.005719 82.624818
[50] 83.155343 83.430593 83.583687 83.878318 83.986459 85.106463 85.832292
[57] 86.128905 86.341810 87.295337 90.845408 91.517725 92.317138

$cuenta
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
[51] 51 52 53 54 55 56 57 58 59 60 61 62
```

□

3. Un **proceso Poisson compuesto** es un proceso estocástico $\{X_t | t \geq 0\}$ que se puede representar como una suma aleatoria $X_t = \sum_{i=1}^{N_t} Y_i$, donde $\{N_t | t \geq 0\}$ es un proceso Poisson y Y_i son iid e independientes del proceso N_t. Escriban un programa para simular un proceso $\mathcal{P}(\lambda)$-$\mathcal{G}(\cdot, \cdot)$ donde Y tiene distribución gamma. Estimen la media y la varianza de X_{10} para varias elecciones de los parámetros y comparar con los valores teóricos.

Solución.

A continuación va mi propuesta de función. Supongamos un proceso Poisson homogéno con parámetro λt. De acuerdo a la teoría:

$$\mathrm{E}(X_t) = \mathrm{E}(N_t)\mathrm{E}(Y_t) = (\lambda t)(\alpha\beta)$$

y la varianza está dada por $\mathrm{Var}(X_t) = \lambda t(\beta^2\alpha(1+\alpha))$. Entonces, si elegimos $t = 10$, $\lambda = 2$, $\alpha = 1$,$\beta = 3.5$, tenemos

```
n <- 10000
Poisson.compuesto <- function(t, lambda, alfa,beta){
                 # Primero construye N(t)
                 TT <- rexp(n,lambda) # genera variables exponenciales para los tiempos.
                 s <- cumsum(TT)      # acumula los tiempos en el vector s
                 Nt <- max(which(s<=t))
                 # Ahora genera Gammas
                 Yt <- rgamma(Nt, shape=alfa,scale= beta)
                 return(Xt=sum(Yt))
}

muestra <- numeric(n)
for(i in 1:n)muestra[i] <- Poisson.compuesto(10,2,1,3.5)
mean(muestra) # valor muestral


[1] 69.63647


10*2*1*3.5        # valor teórico


[1] 70


var(muestra)  # valor muestral


[1] 479.0304


10*2*(3.5^2*1*(1+1)) # valor teórico


[1] 490
```

$\square$

4. Para un movimiento Browniano, encontrar:

   - $P(B_2 \leq 1)$
   - $E(B_4|B_1 = x)$
   - $Corr(B_{t+s}, B_s)$
   - $\mathrm{Var}(B_4|B_1)$
   - $P(B_3 \leq 5|B_1 = 2)$

   Solución.

   - Como $B_2 \sim \mathcal{N}(0, 2)$, entonces $P(B_2 \leq 1) = P(Z \leq 1/\sqrt{2}) = \Phi(1/\sqrt{2})$

     ```
     pnorm(1/sqrt(2))

     [1] 0.7602499
     ```

   - $E(B_4|B_1 = x) = E(B_4 - B_1 + B_1|B_1 = x) = E(B_4 - B_1|B_1 = x) + x = E(B_4 - B_1) + x = E(B_3)+x = x$. La penúltima desigualdad es porque $B_3 \perp\!\!\!\perp B_1$. Entonces $E(B_4|B_1 = x) = x$.

- Recordando la definición de correlación: $cor(B_{t+s}, B_s) = \frac{\text{cov}((,B)_{t+s}, B_s)}{ds(B_{t+s}ds(B_s)} = \frac{min(t+s,s)}{\sqrt{t+s}\sqrt{s}} = \frac{s}{\sqrt{s(t+s)}}$.
- $Var(B_4|B_1) = Var(B_4 - B_1 + B_1|B_1) = Var(B_4 - B_1|B_1) + Var(B_1|B_1) = Var(B_4 - B_1) = Var(B_3) = 3$.
- $P(B_3 \leq 5|B_1 = 2) = P(B_3 - B_1 \leq 5 - B_1|B_1 = 2) = P(B_3 - B_1 \leq 3) = P(B_2 \leq 3)$

```
pnorm(3,0,sqrt(2))

[1] 0.9830526
```

□

5. Supongan que la acción XYZ se vende hoy por $80 por acción y sigue un movimiento browniano geométrico con drift 0.10 y volatilidad 0.5. Encuentren la probabilidad de que en 90 días el precio de XYZ se eleve a por lo menos $100.

Solución.

Si consideramos que la unidad de valuación es 1 año, entonces, si S_t denota el precio de la acción XYZ después de t años, y considerando que 90 días es 1/4 de año,

$$\begin{aligned} P(S_{0.25} \geq 100) &= P(80e^{0.25\mu + \sigma Z_{0.25}} \geq 100) \\ &= P(0.1 * 0.25 + 0.5 * Z_{0.25} \geq \log(100/80)) \\ &= P(Z_{0.25} \geq 0.396) = 0.214 \end{aligned}$$

```
1-pnorm((log(100/80)-0.1/4)/0.5,0,sqrt(0.25))

[1] 0.214013
```

□

6. El precio de una acción se modela con un movimiento Browniano geométrico con drift $\mu = -0.25$ y volatilidad $\sigma = 0.4$. La acción actualmente se vende a $35. Supongan que hay una opción para comprar esa acción en 6 meses a $40. Encuentren la ganancia esperada de la opción.

Solución.

En este problema, sabemos que la discretización del Browniano geométrico es $S_t = S_0 e^{\mu\Delta t + \sigma\epsilon\sqrt{\Delta t}}$. En el problema dado, $S_0 = 35$, $\mu = -0.25$, $\sigma = 0.40$ y $Deltat = 0.5$. Entonces, podemos generar una muestra de tamaño n de los valores $S_{0.5} = 35 * \exp(-0.25 * 0.5 + 0.4\sqrt{0.5}\epsilon)$ y calcular el estimador de $E(max(0, S_{0.5} - 40))$ con el promedio $\frac{\sum_{i=1}^{n} max(0, \hat{S}_{0.5,i} - 40)}{n}$.

```
n <- 1000000
S05 <-NULL
S05  <- 35*exp(-0.25*0.5+0.4*sqrt(0.5)*rnorm(n,0,1))
sum(ifelse(S05-40<0,0,S05-40))/n

[1] 1.267074
```

□

7. Supongan que X,Y son iid $\mathcal{U}(0,1)$ y definan Z como

$$Z = \begin{cases} 1 & \text{si } X^2 + Y^2 \leq 1 \\ 0 & \text{en otro caso} \end{cases}$$

- Obtener $E(Z)$.
- Simulando Z, escribir un programa para estimar π.

Solución.

Para obtener el valor de $E(Z)$, basta con simular puntos en el cuadrado $[0,1] \times [0,1]$, y quedarnos con los puntos que cumplen estar dentro de la circunferencia.

```
AreaSector <- function(n){
x <- runif(n); y <- runif(n)
z <- ifelse(x^2+y^2<=1,1,0)
return(sum(z)/n)
}
AreaSector(10000)

[1] 0.7742
```

Esto corresponde a la cuarta parte de la circunferencia. Entonces, para estimar π, debemos resolver la ecuación $4E(Z) = \pi r^2 = \pi$

```
pihat <- 4*AreaSector(10000000)
pihat

[1] 3.14168
```

$\square$

8. En varios problemas de la vida real nos interesa calcular con bastante precisión probabilidades en las colas. Supongamos que nos interesa estimar $P(X > 20)$, con $X \sim \mathcal{N}(0,1)$. Pueden comprobar que simular X no servirá. La mejor forma de resolver el problema es expresar esa probabilidad como una integral y usar un cambio de variable para reescribir esa integral como una esperanza bajo la distribución $\mathcal{U}(0,1/20)$. Deduzcan una aproximación de Monte Carlo a $P(X > 20)$ junto con una estimación de error.

Solución.

El ejercicio propone hacer un cambio de variable para tener mayor control sobre la región que interesa integrar.

Noten que si definimos $u = 1/x$,

$$\begin{aligned} P(X > 20) = \int_{20}^{\infty} \phi(x)dx &= \int_{1/20}^{0} \phi(1/u)(-1/u^2)du \\ &= \int_{0}^{1/20} \phi(1/u)(1/u^2)du \\ &= 20E_U(\phi(1/u)/u^2) \end{aligned}$$

donde el valor esperado lo tomamos de $U \sim \mathcal{U}(0,1/20)$. Entonces:

```r
n <- 10000
u <- runif(n,0,1/20)
theta <- 1/(20*u^2)*dnorm(1/u,0,1)
mean(theta)
```

```
[1] 2.930535e-89
```

```r
sd(theta)
```

```
[1] 3.950035e-88
```

□

9. Usar Monte Carlo para encontrar un intervalo de confianza del 95 % para la siguiente integral:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} exp\left\{\frac{1}{2}\left[x^2+(y-1)^2-\frac{x(y-1)}{10}\right]\right\} dxdy$$

Solución.

Cabe hacer notar que como está la integral, es probable que no converja debido a que el exponente crece sin cota. El exponente debe ser con signo menos, el error es mío en este caso.

Aunque posiblemente esto pase, propondré una solución en ambos casos.

- Para el caso de la integral como está. Esta integral se puede escribir como una integral definida con el mismo límite de integración a y pensar en muestras en el cuadrado uniforme, así que si la integral es θ,

$$\hat{\theta} = lim_{a\to\infty} 4a^2 E[h(X,Y)]$$

  con $X, Y \sim \mathcal{U}(-a,a)$ independientes.

```r
theta <- function(a){
        n <- 1e7
        x <- runif(n,-a,a)
        y <- runif(n,-a,a)
        h <- exp(0.5*(x^2+(y-1)^2-x*(y-1)/10))
return(mean(4*a^2*h))
}
theta(10)
```

```
[1] 2.004761e+48
```

```r
theta(100)
```

```
[1] Inf
```

```r
theta(10000)
```

```
[1] Inf
```

- Para el caso de la integral

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} exp\left\{-\frac{1}{2}\left[x^2+(y-1)^2-\frac{x(y-1)}{10}\right]\right\} dxdy$$

  se puede usar las mismas uniformes, o notar que el kernel corresponde a una normal multivariada con media $\boldsymbol{\mu} = (0,1)$ y matriz de varianzas y covarianzas $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & -1/20 \\ -1/20 & 1 \end{pmatrix}$. Sin embargo, lo haré considerando la misma función de arriba.

```r
theta <- function(a){
        n <- 1e7
        x <- runif(n,-a,a)
        y <- runif(n,-a,a)
        h <- exp(-0.5*(x^2+(y-1)^2-x*(y-1)/10))
return(mean(4*a*a*h))
}
theta(10)

[1] 6.289264

theta(100)

[1] 6.222126

theta(1000)

[1] 8.943638
```

□

10. Otro modelo para simular precios de acciones es el siguiente modelo binomial: si S_i denota el precio en el tiempo ih donde $i = 0, 1, 2, \ldots,$ y h es un incremento de tiempo positivo. Sean μ y σ la tasa de interés y la volatilidad respectivamente. Sean

$$
\begin{aligned}
u &= \frac{1}{2}\left(e^{-\mu h} + e^{(\mu+\sigma^2)h} + \frac{1}{2}\sqrt{(e^{-\mu h} + e^{(\mu+\sigma^2)h})^2 - 4}\right) \\
\nu &= u^{-1} \\
p &= \frac{e^{\mu h} - \nu}{u - \nu}
\end{aligned}
$$

Entonces

$$S_i = X_i S_{i-1}$$

donde $X_i, i = 0, 1, \ldots$ son variables bernoulli independientes con distribución $P(X_i = u) = p$, $P(X_i = \nu) = 1 - p$ para toda i.

- Simular el precio al final de cada semana durante el siguiente año con $S_0 = 100$, $\mu = 0.2$ por año, $\sigma = 0.3$ por año, y $h = 1/52$ años.
- Supongan que hay 252 dias hábiles en un año. Hacer $h = 1/252$.
  Para cualquier realización, sea $S_{max} = max\{S_j | j = 0, 1, \ldots, 756\}$. La pérdida $= S_{max} - S_{756}$ denota la diferencia entre vender la acción en el pico de su precio durante los siguientes tres años y venderla después de tres años.
  Simular 200 realizaciones de la pérdida y construir su distribución empírica.

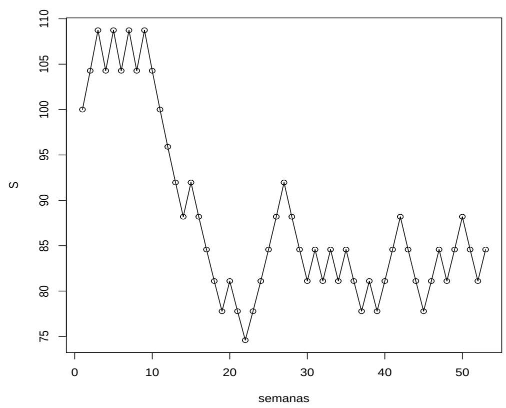
Solución.

La solución de este ejercicio es directa, de las definiciones dadas.

- El siguiente programa simula los precios al final de cada semana por el primer año. En un año hay 52 semanas.

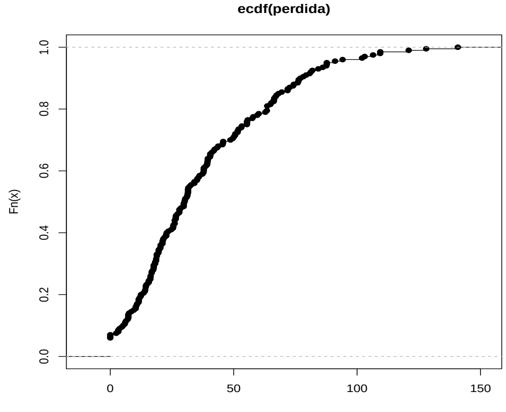
```r
h <- 1/52
mu <- 0.2
sigma2 <- 0.3^2
S0 <- 100

u <- 0.5*(exp(-mu*h) + exp((mu+sigma2)*h)) + 0.5*sqrt((exp(-mu*h) + exp((mu+sigma2)*h))^2-4)
v <- 1/u
```

```
p <- (exp(mu*h)-v)/(u-v)

#saltos bernoulli
X <- rbinom(52,1,p)
X <- ifelse(X==0,v,u)
S <- S0
for(i in 2:(length(X)+1)) S[i] <- X[i-1]*S[i-1]
plot(S,type="o",xlab="semanas")
```



- Aquí usamos los mismos parámetros que en el inciso anterior, excepto $h = 1/252$.

```
h <- 1/252
u <- 0.5*(exp(-mu*h) + exp((mu+sigma2)*h)) + 0.5*sqrt((exp(-mu*h) + exp((mu+sigma2)*h))^2-4)
v <- 1/u
p <- (exp(mu*h)-v)/(u-v)

perdida <- NULL
for(j in 1:200){ #200 realizaciones
X <- rbinom(756,1,p)
X <- ifelse(X==0,v,u)
S <- S0
for(i in 2:(length(X)+1)) S[i] <- X[i-1]*S[i-1]
perdida[j] <- max(S)- S[757]
}
plot(ecdf(perdida))
```



□

Problemas opcionales (mayor complejidad):

1. Un hospital tiene 5 ambulancias para emergencias. El área de cobertura de casos se aproxima con un circulo de diámetro 5km, con el hospital en el centro. La distribución física de los accidentes es un proceso Poisson en el espacio, con una tasa de λ casos por hora. Si una ambulancia no está disponible el paciente tiene que esperar hasta que alguna se libere. Una ambulancia siempre toma una ruta en línea recta a la escena de la emergencia y regresa al hospital. Supongan que las ambulancias viajan a una velocidad constante de ν km/hr y que sólo se requiere una ambulancia para cada emergencia.

   - Mostrar que el tiempo total de viaje de retorno (x horas) se puede muestrear tomando $x = 10\sqrt{U}/v$ donde $U \sim \mathcal{U}(0, 1)$
   - Simular el sistema para obtener para cada paciente el tiempo entre la ocurrencia de la emergencia y la llegada al hospital.

   Solución.

   - Si R denota la distancia del punto (X, Y) al origen, tenemos que encontrar primero la probabilidad de que $R \leq r$. Recordando lo que vimos del proceso Poisson en el espacial con parámetro λ en un conjunto A, primero se simula el número de puntos N en A de acuerdo a la distribución Poisson con parámetro $\lambda|A|$, luego se generan N puntos uniformemente distribuídos en A.
     Considerando que los puntos son uniformes,

     $$P(R \leq r) = \int \int_D f_{X,Y}(x, y)\, dx\, dy$$

     donde $D = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq r\}$ es el disco de diametro r. Para resolver la integral anterior, sabemos que X y Y son uniformes independientes en $(-a, a)$ y podemos convertir a coordenadas polares para facilitar la integración:

     $$\begin{aligned}
     P(R \leq r) &= \int \int_D f_{X,Y}(x, y)\, dx\, dy = \int \int_D f_X(x) f_Y(y)\, dx\, dy \\
     &= \int \int_D \frac{1}{2a} \frac{1}{2a}\, dx\, dy \\
     &= \frac{1}{4a^2} \int \int_D dx\, dy = \frac{1}{4a^2} \int_0^{2\pi} \int_0^r r dr\, d\theta \\
     &= \frac{1}{4a^2} \int_0^{2\pi} \frac{r^2}{2} d\theta = \frac{2\pi r^2}{8a^2} = \frac{\pi r^2}{4a^2}
     \end{aligned}$$

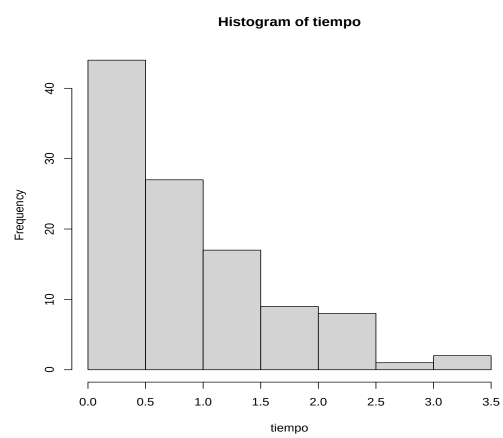
     Entonces $P(R \leq r) = P(X^2 + Y^2 \leq r^2) = \frac{\pi r^2}{4a^2}$. Ahora bien, como el punto tiene que ocurrir necesariamente en el circulo de radio 5, debemos considerar la probabilidad condicional a estar en el círculo. Entonces:

     $$P(R \leq r | R \leq 5) = \frac{P(R \leq r, R \leq 5)}{P(R \leq 5)} = \frac{\pi min(5, r)^2/4a^2}{\pi 5^2/4a^2} = (r/5)^2$$

     Por lo tanto, dado que estamos en el circulo, $F_R(r) = (r/5)^2$, por lo que $r = 5\sqrt{U}$. Como el trayecto es de ida y vuelta, se obtiene que la distancia recorrida es $2r = 10\sqrt{U}$. Por último, como el tiempo está dado por la fórmula $t = d/v$, entonces $t = 10\sqrt{U}/v$.

- El siguiente código simula el sistema. Los tiempos están dados en el vector solución. El ejercicio asume que se simulan 100 horas del proceso.

```r
simulaAmbulancias <- function(amb=5){
        #inicialmente no hay siniestros y todas las ambulancias están desocupadas.
        # q = número de pacientes esperando ambulancia (la cola)
        # b = número de ambulancias ocupadas
        # A = tiempo de la siguiente emergencia
        # amb = número de ambulancias.
        # TD[i]  = tiempo de dejada en hospital de la ambulancia i
        # TA[i] = tiempo de levantamiento de la emergencia de la ambulancia i.
        # Ar[i] = tiempo de llegada del paciente que es el j-ésimo en la cola para ambulancia.
        # clck = es el reloj de simulación
        # simtim = duración de la simulación

simtim <- 100
clock <- 0 #inicializa el reloj de simulación
b <- 0
q <- 0 #inicialmente no hay emergencias.
A <- rexp(1) #se genera una emergencia en el tiempo.
Ar <- tiempo <- NULL
TD <- rep(Inf,amb)
TA <- rep(Inf,amb)
while(clock < simtim){
clock <- min(A,TD)
if (clock==A) {
                q <- q+1
                A <- clock + rexp(1) #nueva emergencia se programa
                Ar[q] <- clock
} else {
                j <- which(TD==clock)
                tiempo <- append(tiempo,clock-TA[j])
                TA[j] <- 0
                TD[j] <- Inf
                b <- b-1
}
if (q > 0 && b < 5) {
                j <- min(which(TD==Inf)) #identifica una ambulancia libre
                TD[j] <- clock + rexp(1,1)
                TA[j] <- Ar[1]
                Ar[1] <- Inf
                Ar <- sort(Ar)
                q <- q-1
                b <- b+1
}
}
return(list(tiempo=tiempo,TA=TA,TD=TD,Ar=Ar,q=q,b=b))
}
tiempo <- simulaAmbulancias()$tiempo
hist(tiempo,breaks=10)
```



□

2. **Marginalización de Monte Carlo** es una técnica para calcular una densidad marginal cuando se simula de una densidad conjunta. Sea $(X_i, Y_i) \sim f_{XY}(x,y)$, independiente, y la correspondiente densidad marginal $f_X(x) = \int f_{XY}(x,y)dy$.

   - Sea $w(x)$ una densidad arbitraria. Mostrar que

     $$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \frac{f_{XY}(x^*, y_i)w(x_i)}{f_{XY}(x_i, y_i)} = \int\int \frac{f_{XY](x^*,y)w(x)}}{f_{XY}(x,y)} f_{XY}(x,y)\, dxdy = f_X(x^*).$$

     La fórmula anterior provee un estimado de Monte Carlo de f_X, cuando la distribución conjunta es conocida salvo una constante.
   - Sea $X|Y = y \sim \mathcal{G}(y, 1)$ y $Y \sim \exp(1)$. Usar la técnica de arriba para graficar la densidad marginal de X (pueden usar cualquier densidad). Comparar con la marginal exacta.
   - Mostrar que si se elije $w(x) = f_X(x)$ funciona para producir la distribución marginal y que es óptima en el sentido de que la varianza del estimador resultante es menor.

Solución.

- Argumentemos primero la primera igualdad. Por definición

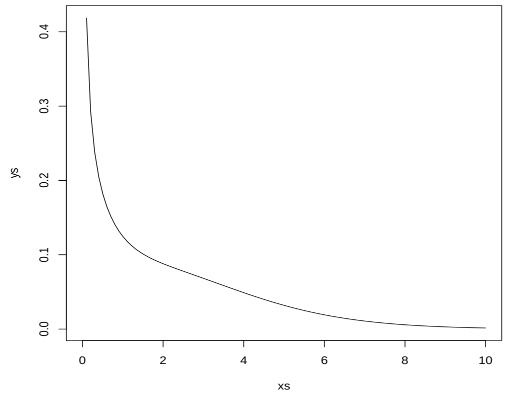
  $$\int\int \frac{f_{XY}(x^*, y)w(x)}{f_{XY}(x,y)} f_{XY}(x,y)dxdy = E_{(X,Y)}\left[\frac{f_{XY}(x^*, y)w(x)}{f_{XY}(x,y)}\right]$$

  Entonces, aplicando la Ley de los grandes números, obtenemos el primer resultado, que es aplicar a la muestra (X_i, Y_i) la función $\frac{f_{XY}(x^*,y)w(x)}{f_{XY}(x,y)}$ y promediar respecto a f_{XY}.
  Para la segunda parte, notando a la misma integral como función de $w(x)$ y aplicando Fubini:

  $$\int\int f_{XY}(x^*, y)w(x) f dxdy = \int f_{XY}(x^*, y)\left[\int w(x)dx\right] dy = \int f_{XY}(x^*, y)dy = f_X(x^*)$$

- Para aplicar el resultado, supongamos que $w(x) = \phi(x)$, la distribución normal estándar. Además, la expresión para $f_{XY}(x,y) = f(x|y)f_y(y)$ se simplifica, porque la expresión para $f_Y(y) = e^{-y}$ se cancela en el numerador y el denominador. Entonces el muestreo queda de la siguiente manera:

```
n <- 10000
y <- rexp(n,1)
x <- rgamma(n,y,1)
xs <- seq(0,10,length=100)
ys <- NULL
for (i in xs){
ys[match(i,xs)] <- mean(dgamma(i,y,1)*dnorm(x)/dgamma(x,y,1))
}
plot(xs,ys,xlim=c(0,10),type="l")
```



Sólo hay un problema para poder comparar con la marginal exacta. Con la información dada, la marginal exacta sería de la forma:

$$f(x) = \int_0^\infty \frac{1}{\Gamma(y)} x^{y-1} e^{-x} e^{-y} dy$$

La cual no se puede integrar explícitamente. Muy probablemente el autor (Robert-Casella) cometieron un error al escribir los parámetros de la densidad gamma, invirtiendo su posición.

- Si se elige $w(x) = f_X(x)$, como indica esta sección, las ecuaciones quedan como:

$$\frac{1}{n}\sum_{i=1}^{n} \frac{f_{XY}(x^*, y_i) f_X(x_i)}{f_{XY}(x_i, y_i)} = \frac{1}{n}\sum_{i=1}^{n} \frac{f_X(x^*) f_{Y|X}(y_i|x^*) f_X(x_i)}{f_X(x_i) f_{Y|X}(y_i|x_i)} = f_X(x^*) \frac{1}{n}\sum_{i=1}^{n} \frac{f_{Y|X}(y_i|x^*)}{f_{Y|X}(y_i|x_i)}$$

Entonces el promedio produce $f_X(x^*)$ por una constante que estima al 1.
Ahora, para demostrar que se reduce la varianza, descomponemos la varianza del estimador del siguiente modo:

$$\begin{aligned} Var\left(\frac{f_{XY}(x^*, y_i) f_X(x_i)}{f_{XY}(x_i, y_i)}\right) &= Var\left[E\left(\frac{f_{XY}(x^*, y_i) f_X(x_i)}{f_{XY}(x_i, y_i)} \mid x_i\right)\right] + \\ & E\left[Var\left(\frac{f_{XY}(x^*, y_i) f_X(x_i)}{f_{XY}(x_i, y_i)} \mid x_i\right)\right] \end{aligned}$$

El primer término es

$$E\left(\frac{f_{XY}(x^*, y_i) f_X(x_i)}{f_{XY}(x_i, y_i)} | x_i\right) = f_X(x^*) E\left(\frac{f_{Y|X}(y_i|x^*)}{f_{Y|X}(y_i|x_i)} | x_i\right) \frac{w(x_i)}{f_X(x_i)} = f_X(x^*)$$

que es una constante, por lo que su varianza es 0.

□