Técnicas y Herramientas Modernas 1

Curso de programación en R

true true true true true

Performance de algoritmos

1. Generar un vector secuencia
2. Implementación de una serie Fibonacci
3. Diseñar un algoritmo para la pesadilla de Gauss
4. Ordenación de un vector por método burbuja
5. Progresión geométrica del COVid-19
6. Algoritmo de funciones estadísticas (media y varianza)
7. Caso de estudio de r-pubs o rblogers
8. Generación de vector secuencia

En éste caso se generará un vector cuyas componentes van sumándose de a tres a medida que se avanza en el mismo. El vector estará compuesto de los números del 0 al 999 y mostrará como resultado sus primeras seis componentes

```
A <- 0
for (i in 2:1000)
{
  A[i] <- (A[i-1] +3)

  }
head (A)
```

```
## [1]  0  3  6  9 12 15
```

2.Implementación de serie Fibonacci

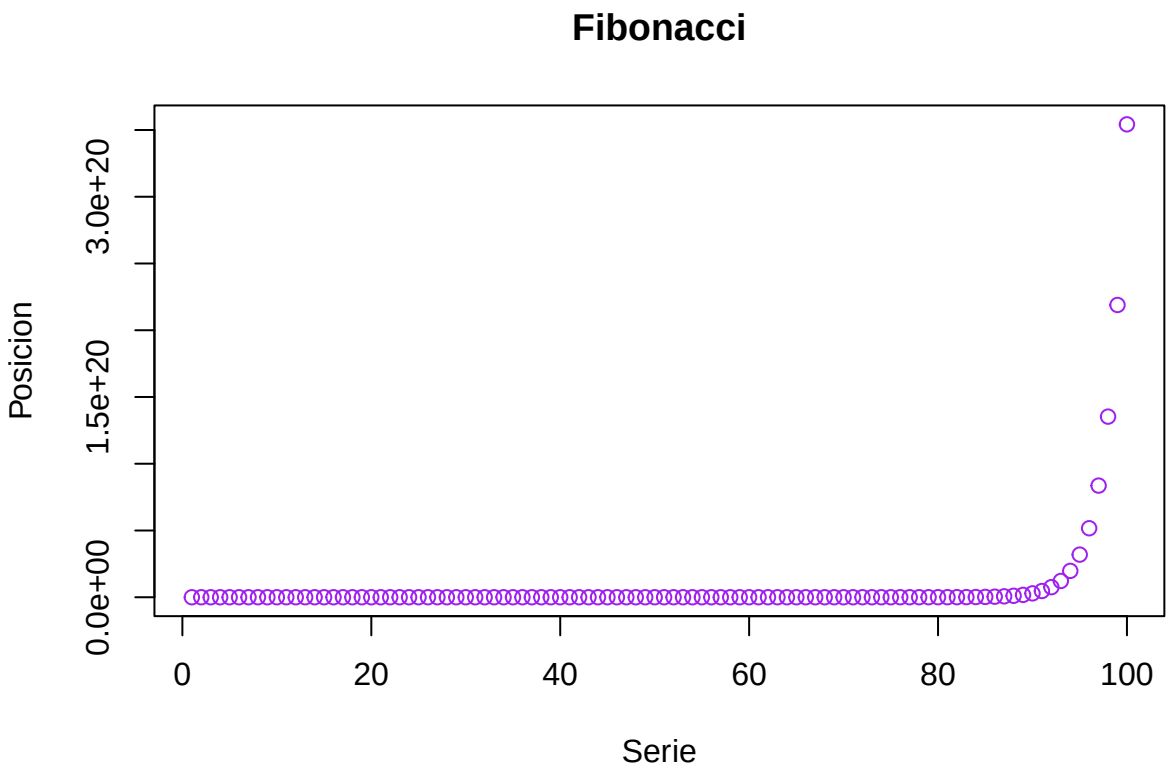
En el siguiente ejemplo se creará y graficará un vector de cien componentes, las cuales seguirán una serie de Fibonacci.La misma consiste en que, siendo la primer componente “0” y la segunda “1”, las siguientes serán la suma de las dos anteriores. Ejemplo inicio de serie Fibonacci: 0;1; 1; 2; 3; 5; 8; 13; 21; 34;...

```
A <- 0
for (i in 3:100){
  A[1] <- 1
  A[2] <- 1
  A[i] <- (A[i-2]+A[i-1])
}
print(A, scientific = FALSE)
```

```
##   [1] 1.000000e+00 1.000000e+00 2.000000e+00 3.000000e+00 5.000000e+00
##   [6] 8.000000e+00 1.300000e+01 2.100000e+01 3.400000e+01 5.500000e+01
##  [11] 8.900000e+01 1.440000e+02 2.330000e+02 3.770000e+02 6.100000e+02
##  [16] 9.870000e+02 1.597000e+03 2.584000e+03 4.181000e+03 6.765000e+03
##  [21] 1.094600e+04 1.771100e+04 2.865700e+04 4.636800e+04 7.502500e+04
##  [26] 1.213930e+05 1.964180e+05 3.178110e+05 5.142290e+05 8.320400e+05
##  [31] 1.346269e+06 2.178309e+06 3.524578e+06 5.702887e+06 9.227465e+06
##  [36] 1.493035e+07 2.415782e+07 3.908817e+07 6.324599e+07 1.023342e+08
##  [41] 1.655801e+08 2.679143e+08 4.334944e+08 7.014087e+08 1.134903e+09
##  [46] 1.836312e+09 2.971215e+09 4.807527e+09 7.778742e+09 1.258627e+10
##  [51] 2.036501e+10 3.295128e+10 5.331629e+10 8.626757e+10 1.395839e+11
##  [56] 2.258514e+11 3.654353e+11 5.912867e+11 9.567220e+11 1.548009e+12
##  [61] 2.504731e+12 4.052740e+12 6.557470e+12 1.061021e+13 1.716768e+13
##  [66] 2.777789e+13 4.494557e+13 7.272346e+13 1.176690e+14 1.903925e+14
##  [71] 3.080615e+14 4.984540e+14 8.065155e+14 1.304970e+15 2.111485e+15
##  [76] 3.416455e+15 5.527940e+15 8.944394e+15 1.447233e+16 2.341673e+16
##  [81] 3.788906e+16 6.130579e+16 9.919485e+16 1.605006e+17 2.596955e+17
##  [86] 4.201961e+17 6.798916e+17 1.100088e+18 1.779979e+18 2.880067e+18
##  [91] 4.660047e+18 7.540114e+18 1.220016e+19 1.974027e+19 3.194043e+19
##  [96] 5.168071e+19 8.362114e+19 1.353019e+20 2.189230e+20 3.542248e+20
```

```
#  B <- 0
# for (i in 2:100)
# {
#   B[i] <- (B[i-1] +1)
#
#   }

plot(A,main="Fibonacci", xlab="Serie", ylab="Posicion",col="purple")
```



Aproximación numero áureo. El número áureo es una constante numérica que se obtiene a partir de dividir un elemento de la serie de Fibonacci con su elemento anterior. A mayor posición del elemento en la serie, mejor es la aproximación al número

```
c<- A[99]/A[98]
d<- A[50]/A[49]

print(c)
```

```
## [1] 1.618034
```

```
print(d)
```

```
## [1] 1.618034
```

3. Implementación de algoritmo para resolver la "pesadilla de Gauss" En el siguiente ejemplo se desarrollará un algoritmo capáz de sumar entre sí todos los elementos de una serie numérica a partir de que el usuario ingrese el último elemento de la misma (tamaño de la serie

```
A <- readline(prompt="Ingrese el último número de la serie: ")
```

```
## Ingrese el último número de la serie:
```

```
B <- (as.numeric(A)*(as.numeric(A)+1)/2)
print(paste("La suma de todos los elementos de la serie es:", B))
```

```
## [1] "La suma de todos los elementos de la serie es: NA"
```

4.Generación de algoritmo para desarrollo de método de ordenamiento “burbuja”, el cual revisa cada elemento del vector a ordenar con el siguiente, intercambiándolos de posición si están en el orden equivocado

```
# x <- sample(-100:100, size = 1000, replace = TRUE)
library(tictoc)
set.seed(123)
x <- rnorm(100,50,25)
t1 <- Sys.time()
tic()
burbuja <- function(x){



n <- length(x) #cuantos numeros dentro de x
for(j in 1:(n-1)){ #j hasta n-1 porque si no buscaria un elemento que no existiria
  for(i in 1:(n-j)){ #barre los numeros que todavia no estan ordenados, va yendo con los desordenados q
    if(x[i]>x[i+1]){
      temp<-x[i]
      x[i]<-x[i+1]
      x[i+1]<-temp
}
}
}
return(x)
}
toc()
```

```
## 0.01 sec elapsed
```

```
t2 <- Sys.time()
res<-burbuja(x)
#Muestra obtenida
x
```

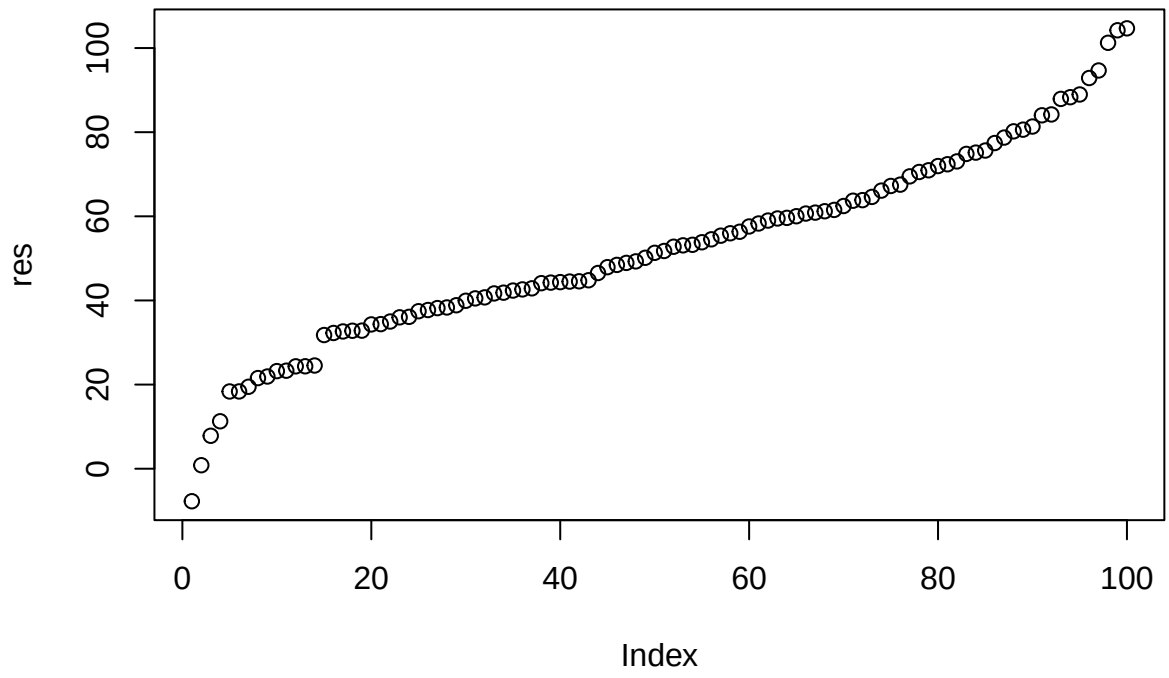
```
##   [1]  35.9881088  44.2455628  88.9677079  51.7627098  53.2321934  92.8766247
##   [7]  61.5229051  18.3734691  32.8286787  38.8584507  80.6020449  58.9953457
##  [13]  60.0192863  52.7670679  36.1039716  94.6728284  62.4462620   0.8345711
##  [19]  67.5338975  38.1802148  23.3044074  44.5506271  24.3498888  31.7777193
##  [25]  34.3740183   7.8326672  70.9446761  53.8343279  21.5465766  81.3453730
##  [31]  60.6616055  42.6232129  72.3781415  71.9533372  70.5395270  67.2160064
##  [37]  63.8479413  48.4522072  42.3509334  40.4882250  32.6323255  44.8020680
##  [43]  18.3650912 104.2238991  80.1990500  21.9222854  39.9278791  38.3336162
##  [49]  69.4991280  47.9157733  56.3329628  49.2863311  48.9282386  84.2150571
##  [55]  44.3557254  87.9117651  11.2811799  64.6153437  53.0963561  55.3985392
##  [61]  59.4909871  37.4419137  41.6698154  24.5356154  23.2052193  57.5882160
##  [67]  61.2052445  51.3251057  73.0566867 101.2521171  37.7242208  -7.7292219
##  [73]  75.1434631  32.2699809  32.7997846  75.6392842  42.8806748  19.4820572
```

```
##  [79]  54.5325870  46.5277159  50.1441046  59.6320100  40.7334992  66.1094137
##  [85]  44.4878360  58.2945491  77.4209753  60.8795373  41.8517104  78.7201905
##  [91]  74.8375964  63.7099240  55.9682934  34.3023481  84.0163112  34.9935103
##  [97] 104.6833248  88.3152657  44.1074910  24.3394775
```

```
#Muestra Ordenada
res
```

```
##   [1]  -7.7292219   0.8345711   7.8326672  11.2811799  18.3650912  18.3734691
##   [7]  19.4820572  21.5465766  21.9222854  23.2052193  23.3044074  24.3394775
##  [13]  24.3498888  24.5356154  31.7777193  32.2699809  32.6323255  32.7997846
##  [19]  32.8286787  34.3023481  34.3740183  34.9935103  35.9881088  36.1039716
##  [25]  37.4419137  37.7242208  38.1802148  38.3336162  38.8584507  39.9278791
##  [31]  40.4882250  40.7334992  41.6698154  41.8517104  42.3509334  42.6232129
##  [37]  42.8806748  44.1074910  44.2455628  44.3557254  44.4878360  44.5506271
##  [43]  44.8020680  46.5277159  47.9157733  48.4522072  48.9282386  49.2863311
##  [49]  50.1441046  51.3251057  51.7627098  52.7670679  53.0963561  53.2321934
##  [55]  53.8343279  54.5325870  55.3985392  55.9682934  56.3329628  57.5882160
##  [61]  58.2945491  58.9953457  59.4909871  59.6320100  60.0192863  60.6616055
##  [67]  60.8795373  61.2052445  61.5229051  62.4462620  63.7099240  63.8479413
##  [73]  64.6153437  66.1094137  67.2160064  67.5338975  69.4991280  70.5395270
##  [79]  70.9446761  71.9533372  72.3781415  73.0566867  74.8375964  75.1434631
##  [85]  75.6392842  77.4209753  78.7201905  80.1990500  80.6020449  81.3453730
##  [91]  84.0163112  84.2150571  87.9117651  88.3152657  88.9677079  92.8766247
##  [97]  94.6728284 101.2521171 104.2238991 104.6833248
```

```
plot(res)
```




```
tiempo <- t2 - t1
print(tiempo)
```

```
## Time difference of 0.005985975 secs
```

Función RStudio para ordenamiento de vectores y posterior comparación del tiempo de ejecución de ambos métodos

```
set.seed(123)
y <- sample(-100:100, size = 1000, replace = TRUE)
b <- 0
b <- sort(y)

x
```

```
##   [1]  35.9881088  44.2455628  88.9677079  51.7627098  53.2321934  92.8766247
##   [7]  61.5229051  18.3734691  32.8286787  38.8584507  80.6020449  58.9953457
##  [13]  60.0192863  52.7670679  36.1039716  94.6728284  62.4462620   0.8345711
##  [19]  67.5338975  38.1802148  23.3044074  44.5506271  24.3498888  31.7777193
##  [25]  34.3740183   7.8326672  70.9446761  53.8343279  21.5465766  81.3453730
##  [31]  60.6616055  42.6232129  72.3781415  71.9533372  70.5395270  67.2160064
##  [37]  63.8479413  48.4522072  42.3509334  40.4882250  32.6323255  44.8020680
##  [43]  18.3650912 104.2238991  80.1990500  21.9222854  39.9278791  38.3336162
##  [49]  69.4991280  47.9157733  56.3329628  49.2863311  48.9282386  84.2150571
##  [55]  44.3557254  87.9117651  11.2811799  64.6153437  53.0963561  55.3985392
```

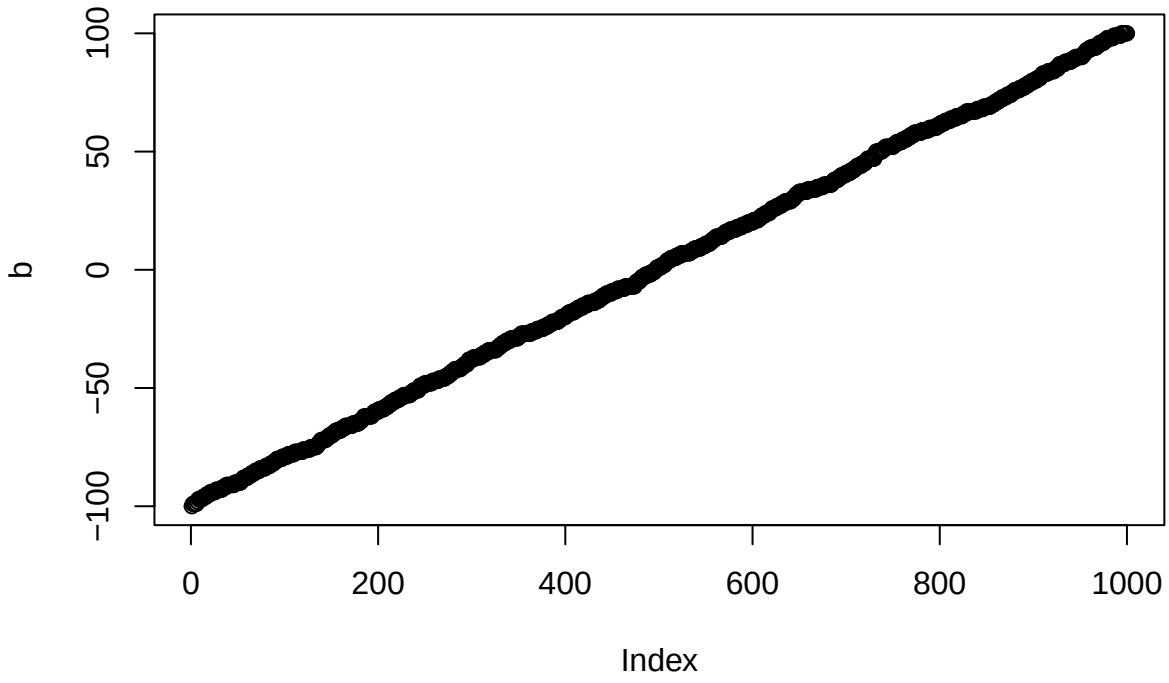
```
##  [61]  59.4909871  37.4419137  41.6698154  24.5356154  23.2052193  57.5882160
##  [67]  61.2052445  51.3251057  73.0566867 101.2521171  37.7242208  -7.7292219
##  [73]  75.1434631  32.2699809  32.7997846  75.6392842  42.8806748  19.4820572
##  [79]  54.5325870  46.5277159  50.1441046  59.6320100  40.7334992  66.1094137
##  [85]  44.4878360  58.2945491  77.4209753  60.8795373  41.8517104  78.7201905
##  [91]  74.8375964  63.7099240  55.9682934  34.3023481  84.0163112  34.9935103
##  [97] 104.6833248  88.3152657  44.1074910  24.3394775
```

```
b
```

```
##    [1] -100  -99  -99  -99  -99  -99  -98  -97  -97  -97  -97  -97  -96  -96
##   [15]  -96  -96  -95  -95  -95  -95  -94  -94  -94  -94  -94  -94  -93  -93
##   [29]  -93  -93  -93  -93  -93  -92  -92  -92  -92  -91  -91  -91  -91  -91
##   [43]  -91  -91  -91  -91  -90  -90  -90  -90  -90  -90  -90  -89  -89  -88
##   [57]  -88  -88  -88  -88  -87  -87  -87  -87  -86  -86  -86  -86  -85  -85
##   [71]  -85  -85  -85  -84  -84  -84  -84  -84  -84  -83  -83  -83  -83  -83
##   [85]  -82  -82  -82  -82  -81  -81  -81  -80  -80  -80  -80  -80  -80  -79
##   [99]  -79  -79  -79  -79  -79  -78  -78  -78  -78  -78  -78  -78  -77  -77
##  [113]  -77  -77  -77  -77  -77  -77  -77  -76  -76  -76  -76  -76  -76  -76
##  [127]  -76  -75  -75  -75  -75  -75  -75  -75  -74  -74  -73  -73  -72  -72
##  [141]  -72  -72  -72  -72  -71  -71  -71  -70  -70  -70  -70  -69  -69  -69
##  [155]  -68  -68  -68  -68  -68  -68  -67  -67  -67  -67  -66  -66  -66  -66
##  [169]  -66  -66  -66  -66  -65  -65  -65  -65  -65  -65  -65  -64  -64  -64
##  [183]  -63  -63  -62  -62  -62  -62  -62  -62  -62  -62  -62  -61  -61  -60
##  [197]  -60  -60  -60  -60  -59  -59  -59  -59  -59  -59  -58  -58  -58  -58
##  [211]  -57  -57  -57  -56  -56  -56  -56  -55  -55  -55  -55  -55  -54  -54
##  [225]  -54  -54  -53  -53  -53  -53  -53  -53  -53  -53  -52  -52  -52  -51
##  [239]  -51  -51  -51  -51  -51  -50  -49  -49  -49  -49  -49  -48  -48  -48
##  [253]  -48  -48  -48  -48  -48  -47  -47  -47  -47  -47  -47  -47  -46  -46
##  [267]  -46  -46  -46  -46  -46  -45  -45  -45  -45  -44  -44  -44  -43  -43
##  [281]  -43  -42  -42  -42  -42  -42  -42  -42  -41  -41  -41  -40  -40  -40
##  [295]  -40  -39  -38  -38  -38  -38  -38  -37  -37  -37  -37  -37  -37  -37
##  [309]  -37  -36  -36  -36  -36  -35  -35  -35  -35  -34  -34  -34  -34  -34
##  [323]  -34  -34  -34  -34  -33  -33  -33  -32  -32  -32  -31  -31  -31  -31
##  [337]  -30  -30  -30  -30  -30  -29  -29  -29  -29  -29  -29  -29  -29  -28
##  [351]  -28  -28  -27  -27  -27  -27  -27  -27  -27  -27  -27  -27  -27  -26
##  [365]  -26  -26  -26  -26  -26  -25  -25  -25  -25  -25  -25  -25  -24  -24
##  [379]  -24  -24  -24  -23  -23  -23  -23  -22  -22  -22  -22  -22  -22  -22
##  [393]  -21  -21  -21  -20  -20  -20  -20  -20  -19  -19  -19  -18  -18  -18
##  [407]  -18  -18  -18  -17  -17  -17  -17  -16  -16  -16  -16  -16  -15  -15
##  [421]  -15  -15  -15  -14  -14  -14  -14  -14  -14  -14  -14  -13  -13  -13
##  [435]  -13  -13  -12  -12  -12  -11  -11  -11  -11  -10  -10  -10  -10  -10
##  [449]  -10   -9   -9   -9   -9   -9   -9   -8   -8   -8   -8   -8   -8   -8
##  [463]   -8   -7   -7   -7   -7   -7   -7   -7   -7   -7   -7   -7   -6   -5
##  [477]   -5   -5   -5   -4   -4   -3   -3   -3   -3   -2   -2   -2   -2   -2
##  [491]   -2   -1   -1   -1   -1    0    0    1    1    1    1    1    2    2
##  [505]    2    2    3    3    4    4    4    4    5    5    5    5    5    5
##  [519]    6    6    6    6    6    7    7    7    7    7    7    7    7    7
##  [533]    7    8    8    8    8    9    9    9    9    9    9    9   10   10
##  [547]   10   10   10   11   11   11   11   11   12   12   12   13   13   13
##  [561]   14   14   14   14   14   14   14   15   15   15   16   16   16   16
##  [575]   16   17   17   17   17   17   17   17   18   18   18   18   18   18
##  [589]   19   19   19   19   19   19   20   20   20   20   20   20   21   21
##  [603]   21   21   21   21   22   22   22   23   23   23   23   24   24   24
```

```
##  [617]   24   24   25   25   26   26   26   26   26   27   27   27   27   27
##  [631]   28   28   28   28   29   29   29   29   29   29   29   30   30   30
##  [645]   31   31   32   32   32   33   33   33   33   33   33   33   33   33
##  [659]   34   34   34   34   34   34   34   34   34   34   35   35   35   35
##  [673]   35   35   35   36   36   36   36   36   36   36   36   36   37   37
##  [687]   38   38   38   38   38   39   39   39   40   40   40   40   40   41
##  [701]   41   41   41   41   42   42   42   42   43   43   43   44   44   44
##  [715]   44   44   45   45   45   45   46   46   47   47   47   47   47   47
##  [729]   47   47   49   50   50   50   50   50   50   50   51   51   51   52
##  [743]   52   52   52   52   52   52   52   52   53   53   53   54   54   54
##  [757]   54   54   54   55   55   55   55   55   56   56   56   56   57   57
##  [771]   57   57   58   58   58   58   58   58   58   58   59   59   59   59
##  [785]   59   59   59   59   60   60   60   60   60   60   60   60   61   61
##  [799]   61   61   62   62   62   62   62   63   63   63   63   63   63   64
##  [813]   64   64   64   64   64   65   65   65   65   65   65   65   66   66
##  [827]   66   66   67   67   67   67   67   67   67   67   67   67   67   68
##  [841]   68   68   68   68   68   68   69   69   69   69   69   69   69   69
##  [855]   70   70   70   70   71   71   71   71   72   72   72   72   73   73
##  [869]   73   73   73   74   74   74   74   74   75   75   75   76   76   76
##  [883]   76   76   76   77   77   77   77   77   78   78   78   78   79   79
##  [897]   79   79   80   80   80   80   80   81   81   81   81   82   82   83
##  [911]   83   83   83   83   83   84   84   84   84   84   84   84   85   85
##  [925]   85   85   86   87   87   87   87   87   87   88   88   88   88   88
##  [939]   88   88   89   89   89   89   90   90   90   90   90   90   90   90
##  [953]   91   91   92   92   93   93   93   93   94   94   94   94   94   94
##  [967]   94   95   95   95   96   96   96   96   96   97   97   97   98   98
##  [981]   98   98   98   98   98   99   99   99   99   99   99   99   99  100
##  [995]  100  100  100  100  100  100
```

```
plot(b)
```




```
library(microbenchmark)
microbenchmark(
  b,
  burbuja
)
```

```
## Unit: nanoseconds
##     expr min lq mean median  uq  max neval
##        b   0  0   51      0 100 1200   100
##  burbuja   0  0   56      0 100 2400   100
```

5. Progresión geométrica del Covid-19

```
library(readr)

location <- getwd()
setwd(location)
casos_a <- read_delim("casos.csv", ";", escape_double = FALSE, trim_ws = TRUE, skip = 1)
```

```
## Rows: 34 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (1): Fecha
## dbl (1): Casos
```

```
## lgl (1): E_P+1
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
m <- length(casos_a$Casos)
F <- (casos_a$Casos[2:m])/(casos_a$Casos[1:m-1])

#Estadisticos de F

mean(F,na.rm = TRUE)
```

```
## [1] 1.350739
```

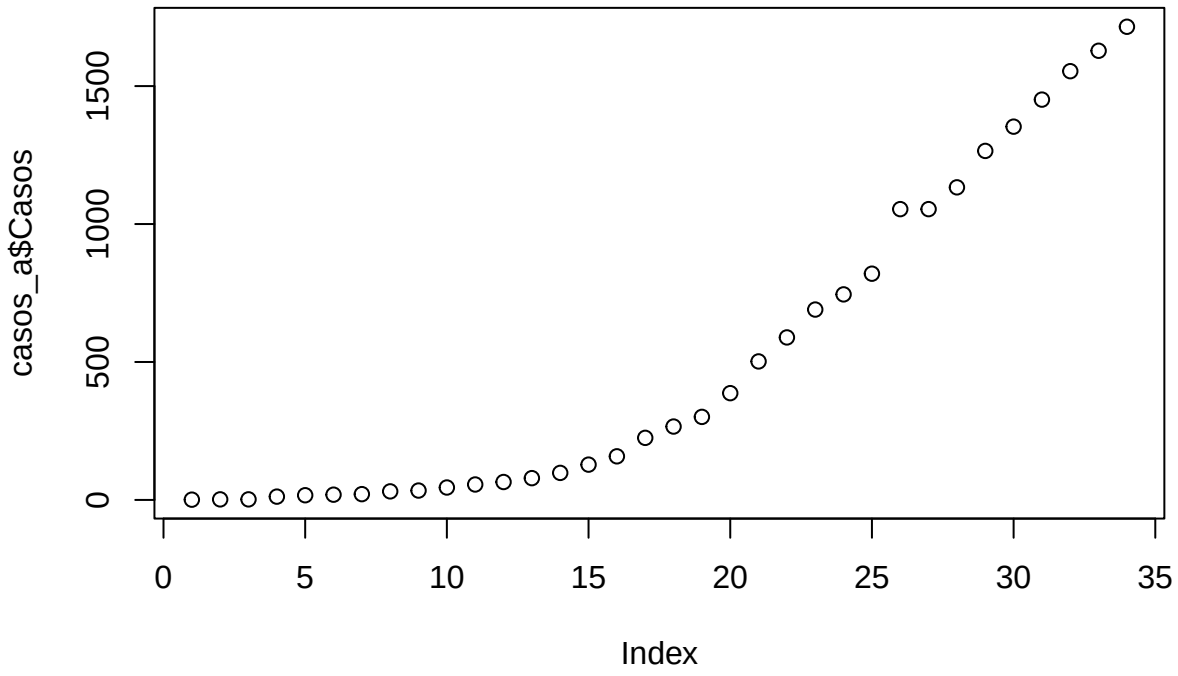
```
sd(F,na.rm = TRUE)
```

```
## [1] 0.8554107
```

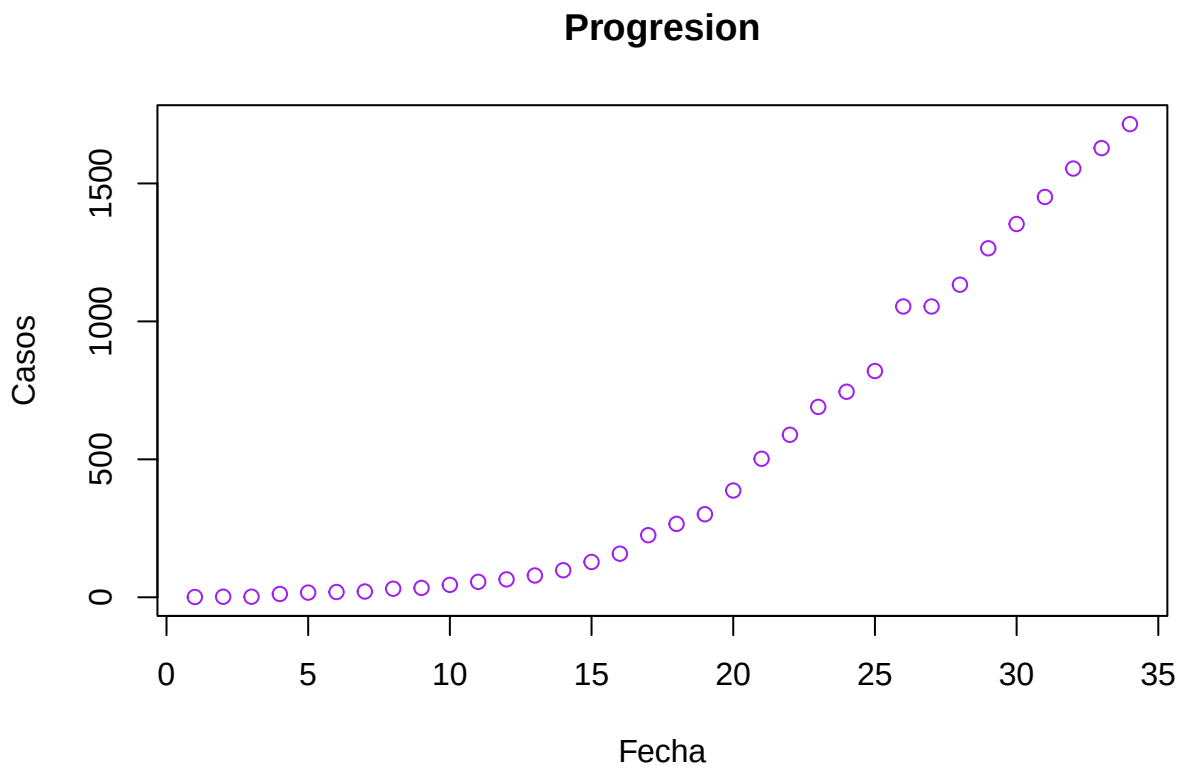
```
var(F,na.rm = TRUE)
```

```
## [1] 0.7317275
```

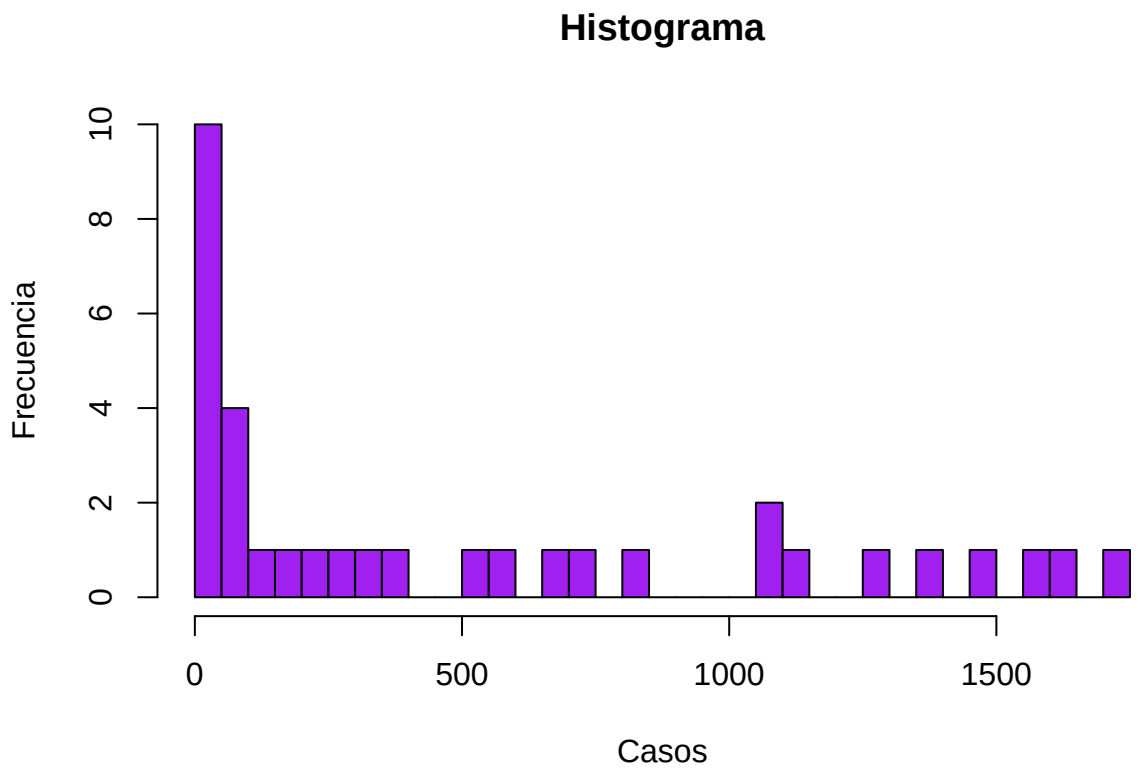
```
plot(casos_a$Casos)
```



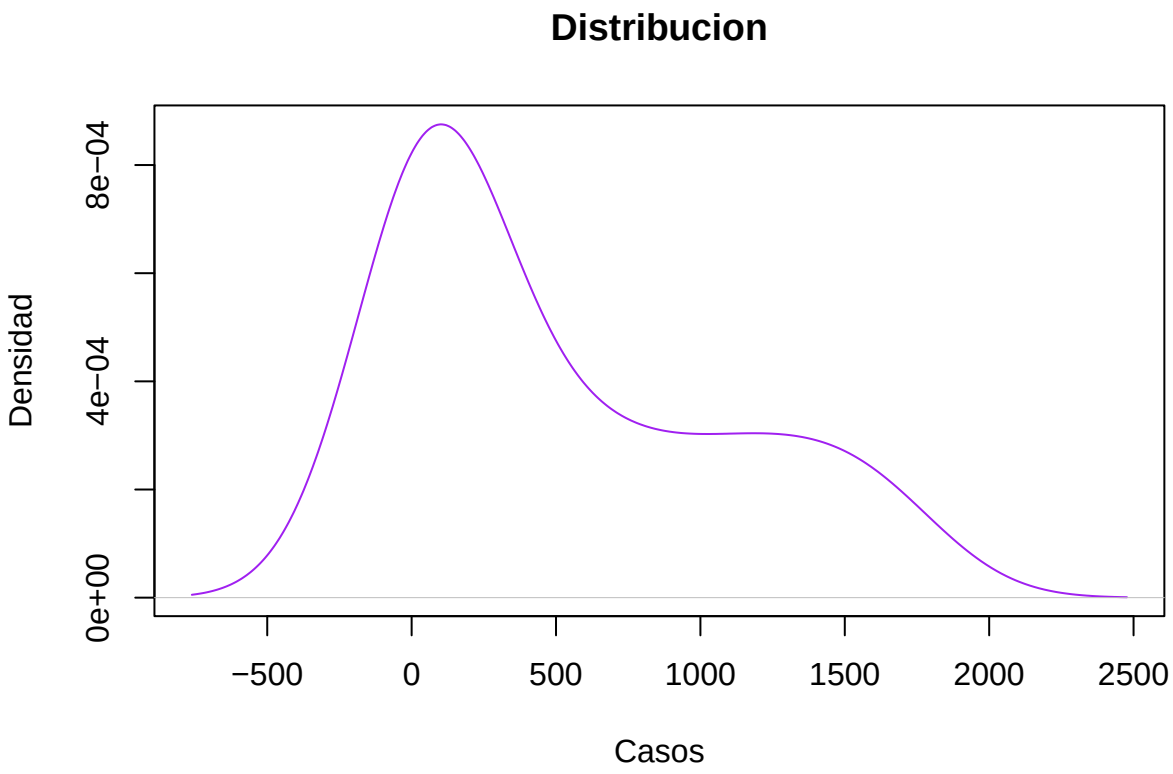
```
plot(casos_a$Casos, main="Progresion", xlab="Fecha", ylab="Casos",col="purple")
```




```
hist(casos_a$Casos,breaks = 50,main="Histograma", xlab="Casos", ylab="Frecuencia",col="purple")
```




```
plot(density(na.omit(casos_a$Casos)), main="Distribucion", xlab="Casos", ylab="Densidad",col="purple")
```


Distribucion
Densidad
0e+00
4e−04
8e−04
−500
0
500
1000
1500
2000
2500
Casos