Análisis Meteorológico en R

Leemos un dataset de una estación meteorológica situada en Molina de Segura, Murcia.

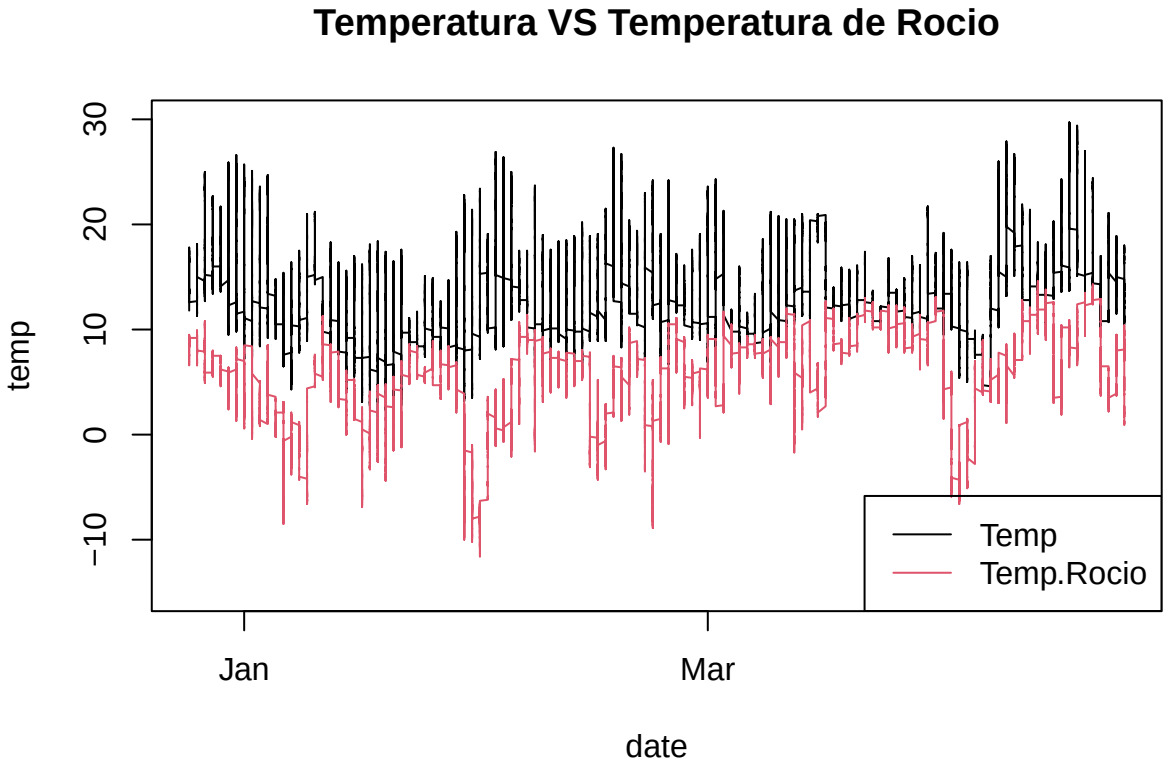
```
data = as.matrix.data.frame(data)

#Sacamos los datos pertinentes

dates_ofi = data[,1]
temp = as.numeric(data[, 3])
dew = as.numeric(data[, 6])
hum = as.numeric(data[, 10])
wind = as.numeric(data[, 12])
press = as.numeric(data[,14])
rain = as.numeric(data[ , 15])
rainrate = as.numeric(data[, 17])
solarad = as.numeric(data[, 18])
data = data.frame(dates_ofi, temp, dew, hum,wind,press,rain,rainrate, solarad )
```

Tras crear un dataframe con las variables que nos interesan, filtramos este dataframe de modo a que no presente ningun valor nulo. 'data = na.omit(data)'

```
data = na.omit(data)


#Datos modificados, sin NA
dates_ofi = data[,1]
temp = as.numeric(data[, 2])
dew = as.numeric(data[, 3])
hum = as.numeric(data[, 4])
wind = as.numeric(data[, 5])
press = as.numeric(data[,6])
rain = as.numeric(data[ , 7])
rainrate = as.numeric(data[, 8])
solarad = as.numeric(data[, 9])

date = as.Date(dates_ofi) #Transformamos las fechas en un formato más adecuado
```

Representamos la temperatura frente a la temperatura de rocío. Según dicta la teoría, si la temperatura de rocío se acerca consecuentemente al valor de la temperatura, entonces, tendremos suficiente humedad como para que llueva.

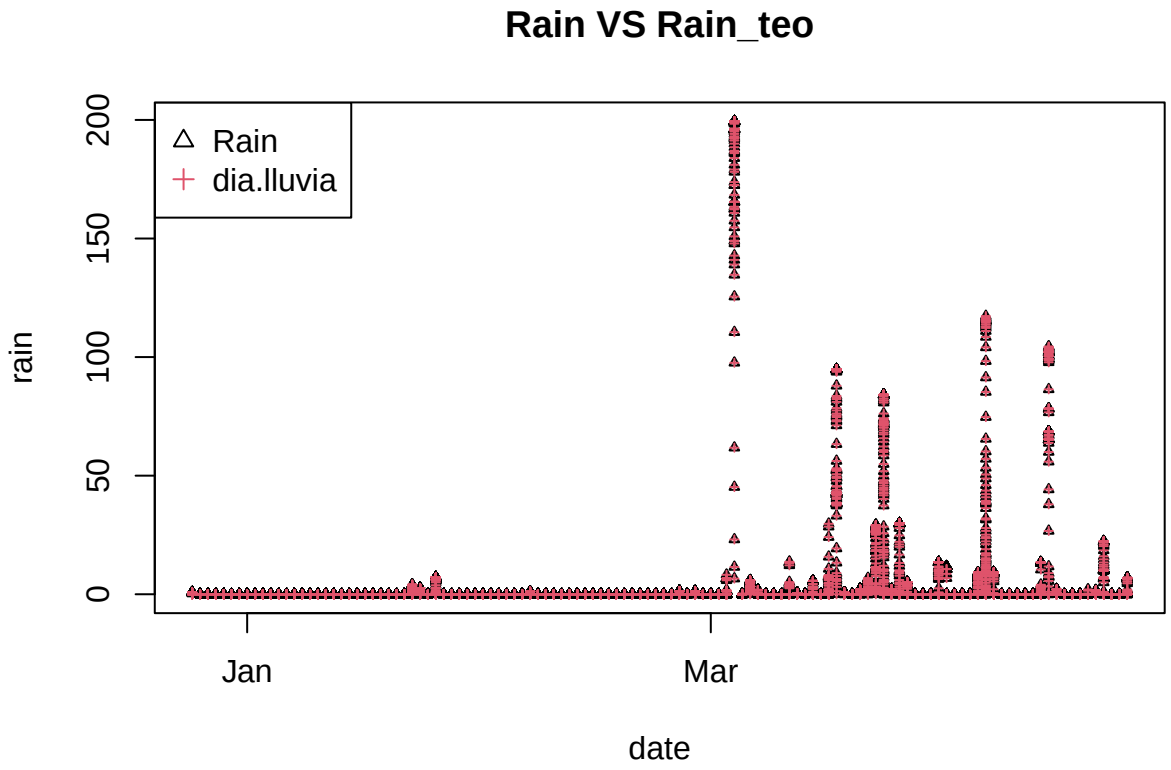
```
plot(date, temp, type = 'l', ylim = c(-15,30), main='Temperatura VS Temperatura de Rocio')
lines(date, dew, col=2)
legend('bottomright', c('Temp', 'Temp.Rocio') , lty = 1, col=c(1,2))
```



Podemos poner en juicio esta teoría con el siguiente bucle. Nos indica: Si el valor absoluto de la diferencia entre la temperatura y la temperatura de rocío es menor que 5, entonces define 'index' igual al índice en que esto ocurra. En caso contrario asigna un valor nulo a 'index'.

A continuación, definimos 'dia_lluvia' como el día en que, según la teoría, debería de llover. Definimos la variable en cuestión como un vector nulo que iremos rellenando según se cumpla la sentencia 'if' o no.

```
plot(date, rain, type = 'p', main='Rain VS Rain_teo', cex = 0.5, pch = 2)
points(date, dia_lluvia , col=2, cex = 0.4, pch = 3)
legend('topleft', c('Rain', 'dia.lluvia') , pch=c(2,3), col=c(1,2))
```



Observamos una alta coincidencia. Para ver cuánto de exacta ha sido esta coincidencia le pediremos a 'R' que nos diga cuantos elementos nulos hay en la 'prediccion'.

```
length(is.null(dia_lluvia))
```

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

Mapa Meteorológico en R

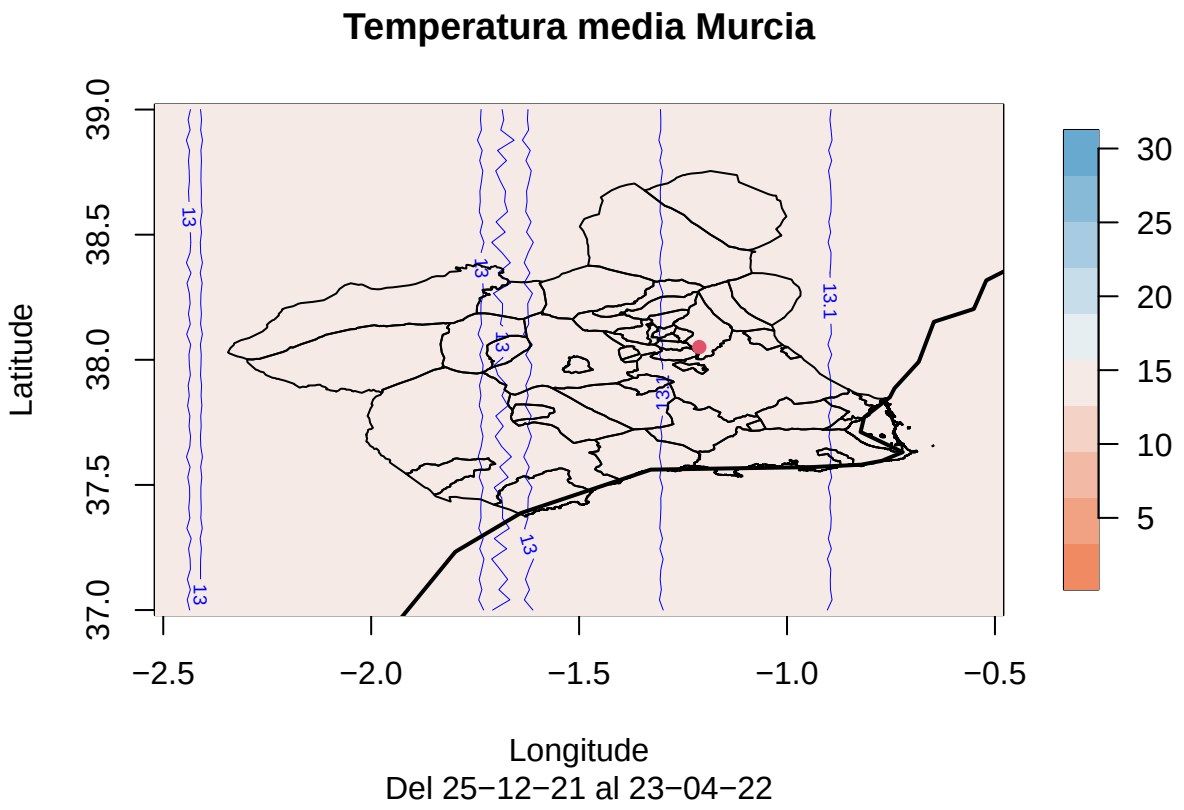
A continuación vamos a pintar un mapa meteorológico, un tanto precario debido a los datos que disponemos. Definimos un dataset (array) que contenga los datos de longitud, latitud y temperatura. A continuación los representaremos con 'image.plot' y sobrepondremos un mapa de murcia junto a sus municipios así como la ubicación de la estación meteorológica.

```
#Definimos longitud y latitud (España)
#Coordenadas estacion 38.073795, -1.164384
n = 50
t = length(temp)
p = length(press)
lons = seq(-2.5,-0.5,len=n)
lons = as.array(lons)
lats = seq(37, 39, len=n)
lats = as.array(lats)

#Si queremos estudiar la temperatura, definimos un data frame tal que
#la primera columna sea lons, la segunda lats y la tercera la temperatura

data_temp = array(c(lons,lats,temp), dim=c(n,n,t) )
data_press = array(c(lons,lats,press), dim=c(n,n,p) )

nlon = dim(lons)
```

```
nlat = dim(lats)

#Empezamos el proceso
lon2<-c(lons[lons>180]-360,lons[lons<=180])
lon2indices<- c(which(lons>180),which(lons<=180) )
temp.mean<- apply(data_temp, c(1,2),mean)
press.mean<- apply(data_press,c(1,2),mean)

toplot<-temp.mean[,nlat:1]

cols <- brewer.pal(3, "RdBu")
pal <- colorRampPalette(cols)


image.plot(lons,lats,toplot, xlab='Longitude', ylab='Latitude',
           zlim = c(min(temp), max(temp)), main='Temperatura media Murcia',
          sub = 'Del 25-12-21 al 23-04-22', col = pal(10))

toplot2<-temp.mean[lon2indices ,nlat:1]
niveles<-c(seq(5,30,0.1))

contour(lon2, lats,toplot2, levels = niveles, add = TRUE,lwd = 0.5, col= "blue")

map('world', 'Spain', add = TRUE, lwd = 2, col = 'black')
if(!'sp' %in% installed.packages()) install.packages('sp')
library('sp')
library('rgdal')
ESPAÑA <- read_rds("/Users/alexandremartinez/Downloads/gadm36_ESP_4_sp.rds")
mur <- ESPAÑA[ESPAÑA$NAME_1=="Región de Murcia",]
map(mur, add=TRUE, col='black')
```

```
## Warning in SpatialPolygons2map(database, namefield = namefield): database does
## not (uniquely) contain the field 'name'.
```

```
points(-1.21099,38.0509, pch = 16, col = 2)
```



Este mapa nos indica la temperatura media desde el 25 de diciembre de 2021 hasta el 23 de abril de 2022. Dicha temperatura resulta ser de unos 13 ºC.

Wolfram Beta en R

Todos los alumnos de ciencia hemos oido hablar de 'Wolfram Alpha' es por ello que quise elaborar una versión un tanto más básica pero efectiva para el grado de física. Una super calculadora científica de bolsillo.

```
#Calculadora



sist.ec=function(t){

  print('Antes de empezar necesita instalar: library(matlib)')
  print('__________')
  print('__________')
  print('Ahora veamos un ejemplo para usar este comando')
  A=matrix(c(1,-2,5,-7), ncol = 2, nrow = 2)
  b<- c(7,-5)
  showEqn(A, b)
  print('Tiene que escribir en la variable X la columna X del sistema, en este caso x=c(1,-2)')
  print('y así con las demás variables')
  print('Respecto a las soluciones, si es un sistema de dos ecuaciones será un vector columna tal que: c
  print('Si es un sistema de 3 variables será un vector de 3 variables')
  print('__________')
  print('__________')
  print('Empezamos')
```

```
p=readline(prompt="¿Es un sistema de 2 o 3 incógnitas?: ")

if(p==2){
  #Variable x
  x1=readline(prompt="¿Cuál es el coeficiente x de la primera ecuación?: ")
  x2=readline(prompt="¿Cuál es el coeficiente x de la segunda ecuación?: ")


  x1=as.numeric(x1)
  x2=as.numeric(x2)


  #Variable y

  y1=readline(prompt="¿Cuál es el coeficiente y de la primera ecuación?: ")
  y2=readline(prompt="¿Cuál es el coeficiente y de la segunda ecuación?: ")


  y1=as.numeric(y1)
  y2=as.numeric(y2)


  #Soluciones

  s1=readline(prompt="¿Cuál es la variable solución de la primera ecuación?: ")
  s2=readline(prompt="¿Cuál es la variable solución de la segunda ecuación?: ")


  s1=as.numeric(s1)
  s2=as.numeric(s2)

  #Operaciones
  x=c(x1,x2)
  y=c(y1,y2)
  s=c(s1,s2)
  A=matrix(c(x,y), ncol = 2, nrow = 2)
  b=s
  showEqn(A, b)
  print('Resolvemos:')
  print('__________')
  print('__________')
  Solve(A,b,verbose = TRUE, fractions = TRUE)

  print('la solución final es: ')
  Solve(A,b)


}else if(p==3){

  #Variable x
  x1=readline(prompt="¿Cuál es el coeficiente x de la primera ecuación?: ")
  x2=readline(prompt="¿Cuál es el coeficiente x de la segunda ecuación?: ")
  x3=readline(prompt="¿Cuál es el coeficiente x de la tercera ecuación?: ")
```

```
x1=as.numeric(x1)
x2=as.numeric(x2)
x3=as.numeric(x3)

#Variable y

y1=readline(prompt="¿Cuál es el coeficiente y de la primera ecuación?: ")
y2=readline(prompt="¿Cuál es el coeficiente y de la segunda ecuación?: ")
y3=readline(prompt="¿Cuál es el coeficiente y de la tercera ecuación?: ")

y1=as.numeric(y1)
y2=as.numeric(y2)
y3=as.numeric(y3)


#Variable z

z1=readline(prompt="¿Cuál es el coeficiente z de la primera ecuación?: ")
z2=readline(prompt="¿Cuál es el coeficiente z de la segunda ecuación?: ")
z3=readline(prompt="¿Cuál es el coeficiente z de la tercera ecuación?: ")

z1=as.numeric(z1)
z2=as.numeric(z2)
z3=as.numeric(z3)

#Soluciones

s1=readline(prompt="¿Cuál es la variable solución de la primera ecuación?: ")
s2=readline(prompt="¿Cuál es la variable solución de la segunda ecuación?: ")
s3=readline(prompt="¿Cuál es la variable solución de la tercera ecuación?: ")


s1=as.numeric(s1)
s2=as.numeric(s2)
s3=as.numeric(s3)

#Operaciones
x=c(x1,x2,x3)
y=c(y1,y2,y3)
z=c(z1,z2,z3)
s=c(s1,s2,s3)

B=matrix(c(x,y,z),ncol =3,nrow= 3)
c=s

print('Resolvemos por Gauss: ')
print('__________')
print('__________')

gaussianElimination(B,c,fractions = TRUE, verbose = TRUE)

print('la solución final es: ')
gaussianElimination(B,c)
```

```
  }


}
cero=function(t){




  x=seq(0,5,len=2000)
  plot(x,f(x),type='l')
  abline(h=0,col=2)
  pr=readline(prompt="¿Quiere cambiar los límites?: ")

  if(pr=='si'){

    xmin=readline(prompt="¿Defina xmin?: ")
    xmax=readline(prompt="¿Defina xmax?: ")
    xmin=as.numeric(xmin)
    xmax=as.numeric(xmax)
    x=seq(xmin,xmax,len=2000)
    plot(x,f(x),type='l')
    abline(h=0,col=2)
    x0=findZeros(f(x)~x, xmin,xmax)
    cat("Los ceros que se encuentran en el intervalo elegido son: \n")
    x0=as.matrix(x0)
    print(x0)

    abline(v=x0,col=4)
    pr=readline(prompt="¿Quiere volver a cambiar los límites?: ")

    if(pr=='si'){

      xmin=readline(prompt="¿Defina xmin?: ")
      xmax=readline(prompt="¿Defina xmax?: ")
      xmin=as.numeric(xmin)
      xmax=as.numeric(xmax)
      x=seq(xmin,xmax,len=2000)
      plot(x,f(x),type='l')
      abline(h=0,col=2)
      x0=findZeros(f(x)~x, xmin,xmax)
      cat("Los ceros que se encuentran en el intervalo elegido son: \n")
      x0=as.matrix(x0)
      print(x0)
      abline(v=x0,col=4)
      pr=readline(prompt="¿Quiere volver a cambiar los límites?: ")

      if(pr=='si'){

        xmin=readline(prompt="¿Defina xmin?: ")
        xmax=readline(prompt="¿Defina xmax?: ")
        ymin=readline(prompt="¿Defina ymin?: ")
        ymax=readline(prompt="¿Defina ymax?: ")
```

```
        xmin=as.numeric(xmin)
        xmax=as.numeric(xmax)
        ymin=as.numeric(ymin)
        ymax=as.numeric(ymax)

        x=seq(xmin,xmax,len=2000)
        plot(x,f(x),type='l', ylim=c(ymin,ymax))
        abline(h=0,col=2)
        x0=findZeros(f(x)~x, xmin,xmax)
        cat("Los ceros que se encuentran en el intervalo elegido son: \n")
        x0=as.matrix(x0)
        print(x0)
        abline(v=x0,col=4)
      }

    }

  }




}
deri=function(x){

  pp=readline(prompt="¿Respecto a qué variable quiere derivar?: ")
  d=pp
  D(x,d)


}
deri2=function(x){

  pp=readline(prompt="Elija en qué punto evaluamos la derivada: ")
  x0=as.numeric(pp)
  h=0.01
  df=(f(x0 + h) - f(x0-h))/(2*h)
  cat('La derivada en x0 es: \n', df)

}
int=function(x){

  l=readline(prompt="Límite inferior: ")
  r=readline(prompt="Límite superior: ")
  l=as.numeric(l)
  r=as.numeric(r)
  s=integrate(x, l, r)

  print('la integral es:')
  print( s)
```

```
}
int2=function(x){

  x1=readline(prompt="¿Cuál es el limite inferior de x1?: ")
  x2=readline(prompt="¿Cuál es el limite inferior de x2?: ")
  x3=readline(prompt="¿Cuál es el limite inferior de x3?: ")
  x1 = as.numeric(x1)
  x2 = as.numeric(x2)
  x3 = as.numeric(x3)

  y1=readline(prompt="¿Cuál es el limite superior de x1?: ")
  y2=readline(prompt="¿Cuál es el limite superior de x2?: ")
  y3=readline(prompt="¿Cuál es el limite superior de x3?: ")
  y1 = as.numeric(y1)
  y2 = as.numeric(y2)
  y3 = as.numeric(y3)

  xmin=c(x1,x2,x3)
  xmax=c(y1,y2,y3)

  s=adaptIntegrate(f, xmin,xmax)
  print('la integral es:')
  print( s)

}
ec.dif1=function(t){
  print('Elija las condiciones iniciales')
  x0=readline(prompt="x0=: ")
  y0=readline(prompt="y0=: ")
  x0=as.numeric(x0)
  y0=as.numeric(y0)

  y=x=NULL
  x[1]=x0
  y[1]=y0

  dx=readline(prompt="Elija precisión del paso dx=: ")
  dx=as.numeric(dx)
  n=readline(prompt="Elija iteraciones n=: ")
  n=as.integer(n)

  for( i in 1:n){

    x[i+1] = x[i] + dx
    y[i+1] = y[i] + f(x[i],y[i])*dx

  }

  plot(x,y,type = 'l')

}
deri3=function(x){
```

```
  pp=readline(prompt="¿Respecto a qué variable quiere derivar?: ")
  d=pp
  der=D(x,d)
  #dder=D(der,d)
  print('La derivada es:')
  print(der)

  print('Ahora cree una función y copie y pegue la derivada dentro de la funcion, tal que:')
  print('f=function(x){DERIVADA}')
  print('Luego compile: mini(f)')

}
mini=function(t){

  print('Vamos a usar el método de ceros de una función para encontrar el mínimo')

  cero(f(x))


}
auto=function(t){



  p=readline(prompt = '¿Cuántas dimensiones tiene su matriz?:')
  n=as.numeric(p)
  A=matrix(t, ncol=n, n)

  print('Los autovalores y autovectores son:')

  s=eigen(A)
  print(s)

}
loc=function(t){
  coord=locator(1)
  coord=as.numeric(coord)
  cat('Coordenadas: \n', coord)

  xl=readline(prompt="anote la coordenada xizquierda del primer cero: ")
  xr=readline(prompt="anote la coordenada xderecha del primer cero: ")

  xl=as.numeric(xl)
  xr=as.numeric(xr)
  n=1000

  fl=f(xl)
  fr=f(xr)

  for(i in 1:n){

    xm=(xl + xr)/2
    fm=f(xm)
```

```
    if(fm*fr<=0){
      xl=xm
      fl=fm

    }else{
      xr=xm
      fr=fm
    }
  }

  cat("x cero se encuentra en: \n")
  print(xm)
  abline(v=xm,col=4)

}
sist.nl = function(p){
  f = rep(NA, length(p0))
  f[1] = a*p[1]^(l) + b*p[2]^(u) - A # = 0
  f[2] = c*p[1]^(g) + d*p[2]^(k) - B # = 0
  f
}
ec.dif2=function(t){
  print('Elija las condiciones iniciales')
  x0=readline(prompt="x0=: ")
  y0=readline(prompt="y0=: ")
  v0=readline(prompt="v0=: ")
  x0=as.numeric(x0)
  y0=as.numeric(y0)
  v0=as.numeric(v0)


  y=x=v=NULL
  x[1]=x0
  y[1]=y0
  v[1]=v0

  dx=readline(prompt="Elija precisión del paso dx=: ")
  dx=as.numeric(dx)
  n=readline(prompt="Elija iteraciones n=: ")
  n=as.integer(n)

  for( i in 1:n){

    x[i+1] = x[i] + dx
    a=f(x[i],y[i])
    v[i+1] = v[i] + f(x[i],y[i])*dx
    y[i+1] = y[i] + v[i+1]*dx

  }

  plot(x,y,type = 'l')
```

```
}
ec.dif3=function(t){
  print('Elija las condiciones iniciales')
  x0=readline(prompt="x0=: ")
  y0=readline(prompt="y0=: ")
  v0=readline(prompt="v0=: ")
  x0=as.numeric(x0)
  y0=as.numeric(y0)
  v0=as.numeric(v0)


  y=x=v=NULL
  x[1]=x0
  y[1]=y0
  v[1]=v0

  dx=readline(prompt="Elija precisión del paso dx=: ")
  dx=as.numeric(dx)
  n=readline(prompt="Elija iteraciones n=: ")
  n=as.integer(n)

  for( i in 1:n){

    x[i+1] = x[i] + dx
    a=f(x[i],y[i])*v[i]
    v[i+1] = v[i] + f(x[i],y[i])*v[i]*dx
    y[i+1] = y[i] + v[i+1]*dx

  }

  plot(x,y,type = 'l')


}
limit=function(t){
  x <- ysym("x")
  p=readline(prompt = 'Quiere límite que tiende a infinito, ¿si o no?:')

  if(p=='si'){
    x <- ysym("x")
    print('Limite - infinito')
    lim=lim(f(x), x, -Inf)
    print(lim)
    print('-------------')
    print('Limite + infinito')
    lim2=lim(f(x), x, Inf)
    print(lim2)

  }else{


    x <- ysym("x")
    print( 'Compile: lim(f(x),x, NUMERO QUE TIENDE EL LIMITE)')
```

```
    print('Ejemplo: lim(sin(x)/x, x, 0)')
  }

}



calculadora=function(t){
  print('Instale: library(matlib), library(Deriv), library(cubature), library(mosaicCalc), library(BB)')
  print('Instale: library(Ryacas), library(mosaic) ')
  print('Indicaciones, responda todas las preguntas en minúsculas y sin tildes.')
  print('Para activar el comando, responda la palabra que está entre guiones: -palabra-')
  print('__________')
  print('__________')

  print('Esta calculadora le permitirá hacer: ')
  print(' un -sistema-(lineal), -ceros- de una funcion, -deri-var, -int-egrar, -ec.dif1-(primer orden),
  print('-ec.dif3-(segundo orden no homogénea), -minimo- de una funcion, -auto-valores y autovectores,
  print('-limit-e de una función?')

  pre=readline(prompt='¿Qué desea hacer?:')

  ##############Sistemas de ecuaciones###########

  if(pre=='sistema'){


    sist.ec=function(t){

      print('Antes de empezar necesita instalar: library(matlib)')
      print('__________')
      print('__________')
      print('Ahora veamos un ejemplo para usar este comando')
      A=matrix(c(1,-2,5,-7), ncol = 2, nrow = 2)
      b<- c(7,-5)
      showEqn(A, b)
      print('Tiene que escribir en la variable X la columna X del sistema, en este caso x=c(1,-2)')
      print('y así con las demás variables')
      print('Respecto a las soluciones, si es un sistema de dos ecuaciones será un vector columna tal q
      print('Si es un sistema de 3 variables será un vector de 3 variables')
      print('__________')
      print('__________')
      print('Empezamos')

      print('Recuerda que para coeficientes fraccionarios, los debes de escribir en su forma decimal')
      print('Es decir que si tienes 1/3, escríbelo coomo 1/3=0.3333')

      p=readline(prompt='¿Es un sistema de 2 o 3 incógnitas?:')
      p=as.numeric(p)

      if(p==2){
        #Variable x
        x1=readline(prompt="¿Cuál es el coeficiente x de la primera ecuación?: ")
```

```
  x2=readline(prompt="¿Cuál es el coeficiente x de la segunda ecuación?: ")


  x1=as.numeric(x1)
  x2=as.numeric(x2)


  #Variable y

  y1=readline(prompt="¿Cuál es el coeficiente y de la primera ecuación?: ")
  y2=readline(prompt="¿Cuál es el coeficiente y de la segunda ecuación?: ")


  y1=as.numeric(y1)
  y2=as.numeric(y2)


  #Soluciones

  s1=readline(prompt="¿Cuál es la variable solución de la primera ecuación?: ")
  s2=readline(prompt="¿Cuál es la variable solución de la segunda ecuación?: ")


  s1=as.numeric(s1)
  s2=as.numeric(s2)

  #Operaciones
  x=c(x1,x2)
  y=c(y1,y2)
  s=c(s1,s2)
  A=matrix(c(x,y), ncol = 2, nrow = 2)
  b=s
  showEqn(A, b)
  print('Resolvemos:')
  print('__________')
  print('__________')
  Solve(A,b,verbose = TRUE, fractions = TRUE)

  print('la solución final es: ')
  Solve(A,b)


}else if(p==3){

  #Variable x
  x1=readline(prompt="¿Cuál es el coeficiente x de la primera ecuación?: ")
  x2=readline(prompt="¿Cuál es el coeficiente x de la segunda ecuación?: ")
  x3=readline(prompt="¿Cuál es el coeficiente x de la tercera ecuación?: ")

  x1=as.numeric(x1)
  x2=as.numeric(x2)
  x3=as.numeric(x3)
```

```
  #Variable y

  y1=readline(prompt="¿Cuál es el coeficiente y de la primera ecuación?: ")
  y2=readline(prompt="¿Cuál es el coeficiente y de la segunda ecuación?: ")
  y3=readline(prompt="¿Cuál es el coeficiente y de la tercera ecuación?: ")

  y1=as.numeric(y1)
  y2=as.numeric(y2)
  y3=as.numeric(y3)


  #Variable z

  z1=readline(prompt="¿Cuál es el coeficiente z de la primera ecuación?: ")
  z2=readline(prompt="¿Cuál es el coeficiente z de la segunda ecuación?: ")
  z3=readline(prompt="¿Cuál es el coeficiente z de la tercera ecuación?: ")

  z1=as.numeric(z1)
  z2=as.numeric(z2)
  z3=as.numeric(z3)

  #Soluciones

  s1=readline(prompt="¿Cuál es la variable solución de la primera ecuación?: ")
  s2=readline(prompt="¿Cuál es la variable solución de la segunda ecuación?: ")
  s3=readline(prompt="¿Cuál es la variable solución de la tercera ecuación?: ")


  s1=as.numeric(s1)
  s2=as.numeric(s2)
  s3=as.numeric(s3)

  #Operaciones
  x=c(x1,x2,x3)
  y=c(y1,y2,y3)
  z=c(z1,z2,z3)
  s=c(s1,s2,s3)

  B=matrix(c(x,y,z),ncol =3,nrow= 3)
  c=s

  print('Resolvemos por Gauss: ')
  print('__________')
  print('__________')

  gaussianElimination(B,c,fractions = TRUE, verbose = TRUE)

  print('la solución final es: ')
  gaussianElimination(B,c)

}
```

```
  }
  sist.ec(t)

}


if(pre=='ceros'){


  cero=function(t){




    x=seq(0,5,len=2000)
    plot(x,f(x),type='l')
    pr=readline(prompt="¿Quiere cambiar los límites?: ")

    if(pr=='si'){

      xmin=readline(prompt="¿Defina xmin?: ")
      xmax=readline(prompt="¿Defina xmax?: ")
      xmin=as.numeric(xmin)
      xmax=as.numeric(xmax)
      x=seq(xmin,xmax,len=2000)
      plot(x,f(x),type='l')
      abline(h=0,col=2)
      x0=findZeros(f(x)~x, xmin,xmax)
      cat("Los ceros que se encuentran en el intervalo elegido son: \n")
      x0=as.matrix(x0)
      print(x0)

      abline(v=x0,col=4)
      pr=readline(prompt="¿Quiere volver a cambiar los límites?: ")

      if(pr=='si'){

        xmin=readline(prompt="¿Defina xmin?: ")
        xmax=readline(prompt="¿Defina xmax?: ")
        xmin=as.numeric(xmin)
        xmax=as.numeric(xmax)
        x=seq(xmin,xmax,len=2000)
        plot(x,f(x),type='l')
        abline(h=0,col=2)
        x0=findZeros(f(x)~x, xmin,xmax)
        cat("Los ceros que se encuentran en el intervalo elegido son: \n")
        x0=as.matrix(x0)
        print(x0)
        abline(v=x0,col=4)
        pr=readline(prompt="¿Quiere volver a cambiar los límites?: ")

        if(pr=='si'){
```

```
            xmin=readline(prompt="¿Defina xmin?: ")
            xmax=readline(prompt="¿Defina xmax?: ")
            ymin=readline(prompt="¿Defina ymin?: ")
            ymax=readline(prompt="¿Defina ymax?: ")
            xmin=as.numeric(xmin)
            xmax=as.numeric(xmax)
            ymin=as.numeric(ymin)
            ymax=as.numeric(ymax)

            x=seq(xmin,xmax,len=2000)
            plot(x,f(x),type='l', ylim=c(ymin,ymax))
            abline(h=0,col=2)
            x0=findZeros(f(x)~x, xmin,xmax)
            cat("Los ceros que se encuentran en el intervalo elegido son: \n")
            x0=as.matrix(x0)
            print(x0)
            abline(v=x0,col=4)
          }

        }

      }









  }


  print('ahora cree una función tal que: f=function(x){....}')
  print('y compile cero(f(x))')
  print('---------------')


}

#Derivada
if(pre=='deri'){


  pr=readline(prompt="¿Quiere una derivada -analitica- o -numerica-?: ")

  if(pr=='analitica'){

    print('Compile el comando: deri(expression(SU FORMULA)) ')
    print('---------------')

    deri=function(x){

      pp=readline(prompt="¿Respecto a qué variable quiere derivar?: ")
      d=pp
```

```
      D(x,d)


    }
  }else if(pr=='numerica'){


    print('Cree la funcion que quiera derivar tal que: f=function(x){SU FORMULA}')
    print('Ahora compile el comando: deri2(f(x))')




    deri2=function(x){

      pp=readline(prompt="Elija en qué punto evaluamos la derivada: ")
      x0=as.numeric(pp)
      h=0.01
      df=(f(x0 + h) - f(x0-h))/(2*h)
      cat('La derivada en x0 es: \n', df)

    }

  }



}

if(pre=='int'){

  pp=readline(prompt="¿Desea integrar de forma analitica o numerica?: ")

  if(pp=='numerica'){




    pr=readline(prompt="Su funcion tiene más de una variable ¿si o no?: ")



    if(pr=='no'){

      int=function(x){

        l=readline(prompt="Límite inferior: ")
        r=readline(prompt="Límite superior: ")
        l=as.numeric(l)
        r=as.numeric(r)
        s=integrate(x, l, r)
```

```
      print('la integral es:')
      print( s)

    }

    print('ahora cree una función tal que: f=function(x){SU FORMULA}')
    print('y compile el comando: int(f)')
    print('----------------')

  }

  if(pr=='si'){

    print('Cree su funcion. Ejemplo: f=function(x,y,z){x[1]*x[2]*x[3]} ')
    print('Ojo es importante el detalle de x[1]..., siendo x[2] correspondiente a la variable y, et
    print('Otra cosa, Este programa está preparado para hacer integrales triples. Si desea hacer una
    print('Suponga: Una integral en la variable x e y. Deberemos de poner como extremos en la integ
    print('0 y 1 para que así la integral de una constante entre el intervalo 0,1 será simpre 1')
    print('una vez creada compile int2(f)')

    int2=function(x){

      x1=readline(prompt="¿Cuál es el limite inferior de x1?: ")
      x2=readline(prompt="¿Cuál es el limite inferior de x2?: ")
      x3=readline(prompt="¿Cuál es el limite inferior de x3?: ")
      x1 = as.numeric(x1)
      x2 = as.numeric(x2)
      x3 = as.numeric(x3)

      y1=readline(prompt="¿Cuál es el limite superior de x1?: ")
      y2=readline(prompt="¿Cuál es el limite superior de x2?: ")
      y3=readline(prompt="¿Cuál es el limite superior de x3?: ")
      y1 = as.numeric(y1)
      y2 = as.numeric(y2)
      y3 = as.numeric(y3)

      xmin=c(x1,x2,x3)
      xmax=c(y1,y2,y3)

      s=adaptIntegrate(f, xmin,xmax)
      print('la integral es:')
      print( s)

    }
  }
}

if(pp=='analitica'){

  print('Compile el comando: antiD( SU FORMULA ~ LA VARIABLE QUE DESEA INTEGRAR ) ')
  print('Ejemplo: antiD(x^2*y ~ x ) ')
```

```
  }
}

if(pre=='ec.dif1'){

  ec.dif1=function(t){
    print('Elija las condiciones iniciales')
    x0=readline(prompt="x0=: ")
    y0=readline(prompt="y0=: ")
    x0=as.numeric(x0)
    y0=as.numeric(y0)

    y=x=NULL
    x[1]=x0
    y[1]=y0

    dx=readline(prompt="Elija precisión del paso dx=: ")
    dx=as.numeric(dx)
    n=readline(prompt="Elija iteraciones n=: ")
    n=as.integer(n)

    for( i in 1:n){

      x[i+1] = x[i] + dx
      y[i+1] = y[i] + f(x[i],y[i])*dx

    }

    plot(x,y,type = 'l')


  }
  ###############
  print('Metodo preparado para resolver ecuaciones tal que: dy/dx=f(x,y)')
  print('Cree una función tal que: f=function(x,y){SU FORMULA}')
  print('Compile el comando: ec.dif1(t)')
}

if(pre=='minimo'){

  print('Compile el comando: deri3(expression(SU FORMULA)) ')
  print('---------------')

  deri3=function(x){

    pp=readline(prompt="¿Respecto a qué variable quiere derivar?: ")
    d=pp
    der=D(x,d)
    #dder=D(der,d)
    print('La derivada es:')
    print(der)

    print('Ahora cree una función y copie y pegue la derivada dentro de la funcion, tal que:')
```

```
    print('f=function(x){DERIVADA}')
    print('Luego compile: mini(f)')

  }

  mini=function(t){

    print('Vamos a usar el método de ceros de una función para encontrar el mínimo')

    cero(f(x))


  }
}

if(pre=='auto'){

  print('Crea un vector con todos los elementos uno a uno columna a columna de su matriz')
  print('Luego compile el comando: auto(v)')
  print('Ejemplo:')
  B=matrix(c(1:9), ncol=3,nrow=3)
  print(B)
  print('En este caso escribiríamos un vector v=c(1,2,3,4,5,6,7,8,9)')
  print('Recuerda que la matriz debe de ser cuadrada')

  auto=function(t){



    p=readline(prompt = '¿Cuántas dimensiones tiene su matriz?:')
    n=as.numeric(p)
    A=matrix(t, ncol=n, n)

    print('Los autovalores y autovectores son:')

    s=eigen(A)
    print(s)

  }



}









if(pre=='sist.nl'){
  n=readline(prompt="¿Con cuantas ecuaciones cuenta el sistema?: ")
  n=as.integer(n)
```

```
  p0 = runif(n)
  print('Defina los coeficientes delante de cada variable y el grado de cada variable')
  print('Empezamos por los coeficientes de la primera ecuación del sistema')

  a=readline(prompt="Coef. x = ")
  b=readline(prompt="Coef. y = ")

  a=as.numeric(a)
  b=as.numeric(b)
  print('Ahora el grado de cada variable de la primera ecuación')
  l=readline(prompt="grado x = ")
  u=readline(prompt="grado y = ")
  l=as.numeric(l)
  u=as.numeric(u)

  print('Pasamos al término independiente ')
  A=readline(prompt="A = ")
  A=as.numeric(A)

  print('Repetimos con la segunda ecuación')


  c=readline(prompt="Coef. x= ")
  d=readline(prompt="Coef. y = ")

  c=as.numeric(c)
  d=as.numeric(d)
  print('Ahora el grado de cada variable de la primera ecuación')
  g=readline(prompt="grado x = ")
  k=readline(prompt="grado y = ")
  g=as.numeric(g)
  k=as.numeric(k)

  print('Pasamos al término independiente ')
  B=readline(prompt="B = ")
  B=as.numeric(B)

  sist.nl = function(p){
    f = rep(NA, length(p0))
    f[1] = a*p[1]^(l) + b*p[2]^(u) - A # = 0
    f[2] = c*p[1]^(g) + d*p[2]^(k) - B # = 0
    f
  }
  p0 = runif(n)
  print('Al fin, la solción es:')
  s=BBsolve(par=p0, fn=sist.nl)
  print(s)
  print('La solución viene dada por: $par')
}

if(pre=='ec.dif2'){

  ec.dif2=function(t){
```

```
    print('Elija las condiciones iniciales')
    x0=readline(prompt="x0=: ")
    y0=readline(prompt="y0=: ")
    v0=readline(prompt="v0=: ")
    x0=as.numeric(x0)
    y0=as.numeric(y0)
    v0=as.numeric(v0)


    y=x=v=NULL
    x[1]=x0
    y[1]=y0
    v[1]=v0

    dx=readline(prompt="Elija precisión del paso dx=: ")
    dx=as.numeric(dx)
    n=readline(prompt="Elija iteraciones n=: ")
    n=as.integer(n)

    for( i in 1:n){

      x[i+1] = x[i] + dx
      a=f(x[i],y[i])
      v[i+1] = v[i] + f(x[i],y[i])*dx
      y[i+1] = y[i] + v[i+1]*dx

    }

    M=cbind(x,y)
    print(M)
    plot(x,y,type = 'l')


  }
  ###############
  print('Metodo preparado para resolver ecuaciones tal que: (dy/dx)^2=f(x,y)')
  print('Cree una función tal que: f=function(x,y){...}')
  print('Compile: ec.dif2(t)')
}

if(pre=='ec.dif3'){

  ec.dif3=function(t){
    print('Elija las condiciones iniciales')
    x0=readline(prompt="x0=: ")
    y0=readline(prompt="y0=: ")
    v0=readline(prompt="v0=: ")
    x0=as.numeric(x0)
    y0=as.numeric(y0)
    v0=as.numeric(v0)


    y=x=v=NULL
```

```
    x[1]=x0
    y[1]=y0
    v[1]=v0

    dx=readline(prompt="Elija precisión del paso dx=: ")
    dx=as.numeric(dx)
    n=readline(prompt="Elija iteraciones n=: ")
    n=as.integer(n)

    for( i in 1:n){

      x[i+1] = x[i] + dx
      a=f(x[i],y[i])*v[i]
      v[i+1] = v[i] + f(x[i],y[i])*v[i]*dx
      y[i+1] = y[i] + v[i+1]*dx

    }

    M=cbind(x,y)
    print(M)
    plot(x,y,type = 'l')


  }
  ###############
  print('Metodo preparado para resolver ecuaciones tal que: (dy/dx)^2=f(x,y)*(dy/dx)')
  print('Cree una función tal que: f=function(x,y){...}')
  print('Compile: ec.dif3(t)')
}

if(pre=='limit'){
  x <- ysym("x")
  limit=function(t){
    x <- ysym("x")
    p=readline(prompt = 'Quiere límite que tiende a infinito, ¿si o no?:')

    if(p=='si'){
      x <- ysym("x")
      print('Limite - infinito')
      lim=lim(f(x), x, -Inf)
      print(lim)
      print('------------')
      print('Limite + infinito')
      lim2=lim(f(x), x, Inf)
      print(lim2)

    }else{


      x <- ysym("x")
      print( 'Compile: lim(f(x),x, NUMERO QUE TIENDE EL LIMITE)')
      print('Ejemplo: lim(sin(x)/x, x, 0)')
    }
```

```
    }

    print('límites para funciones de una sola variable')
    print('ahora cree una función tal que: f=function(x){....}')
    print('y compile limit(t)')
    print('---------------')


  }


  #Cerramos funcion
}
```