Ejemplo práctico: estimación de la porosidad y la permeabilidad.

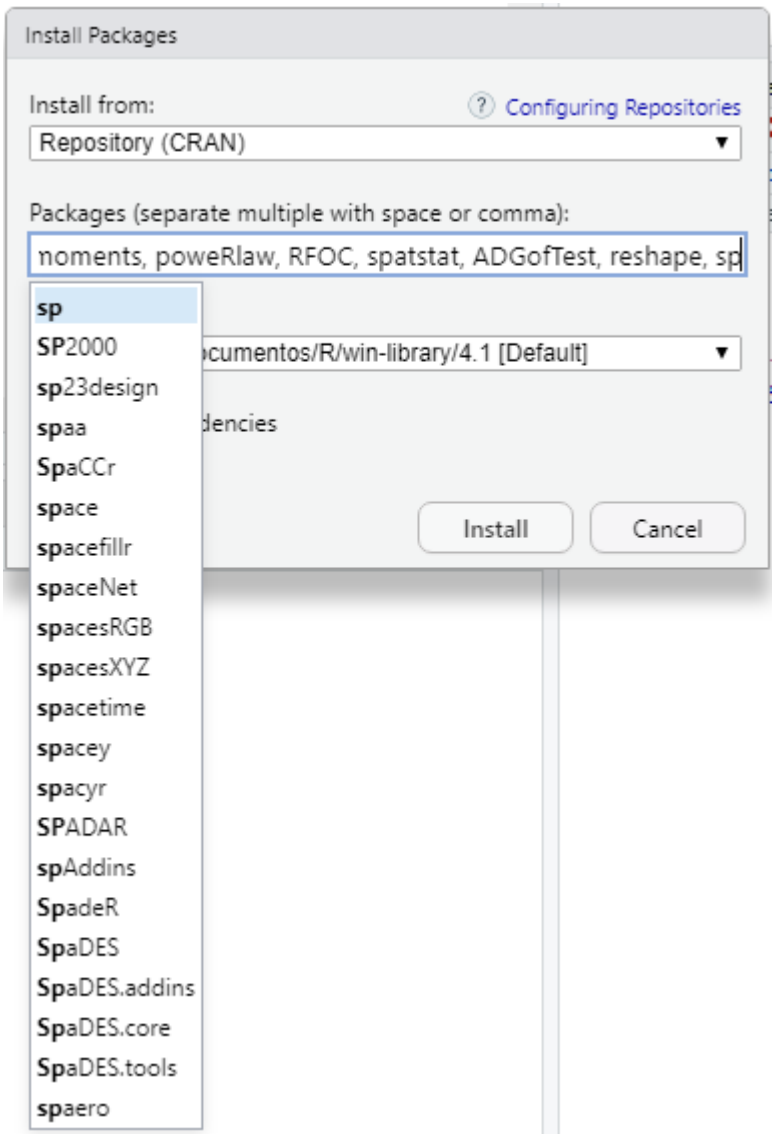
Introducción.

Carga de paqueterias y funciones.

Para obtener la estimación espacial debemos instalar en R Studio los siguientes paquetes: Rcpp, maps, mapproj, actuar, fields, fitdistrplus, geoR, gstat, MASS, moments, poweRlaw, RFOC, spatstat, ADGofTest, reshape, sp.

NOTA: la versión de R recomendada es 4.1.1

Hay dos formas de instalar estos paquetes: la primera opción es ir a la barra de menús en la interfaz de R Studio, dar click en tools>install Packages. En el renglón Packages pondrán los nombres de los paquetes separados por coma y después dan click en install



La segunda opción es usando la consola, para eso debemos copiar las siguientes líneas en un script y ejecutarlo o copiar línea por línea en la consola de R Studio.

In [1]:

```
root_dir<-getwd()

#install_dir- installation directory

install_dir<-paste(root_dir,"/Installation",sep="")

setwd(install_dir)

install.packages("Rcpp")
install.packages("maps")
install.packages("mapproj")
install.packages("actuar")
install.packages("fields")
install.packages("fitdistrplus")
install.packages("geoR")
install.packages("gstat")
install.packages("MASS")
install.packages("moments")
install.packages("poweRlaw")
install.packages("RFOC")
install.packages("spatstat")
install.packages("ADGofTest")
install.packages("reshape")
install.packages("sp")

#set back to root work directory
```

```
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
  There is a binary version available but the source version is later:
     binary source needs_compilation
Rcpp  1.0.6  1.0.7              TRUE

  Binaries will be installed
package 'Rcpp' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'maps' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'mapproj' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
Warning message:
"package 'actuar' is not available (for R version 3.6.1)"Installing package into 'E:/OneDr
ive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
  There is a binary version available but the source version is later:
       binary source needs_compilation
fields   11.6   12.5              TRUE

  Binaries will be installed
package 'fields' successfully unpacked and MD5 sums checked
```

```
The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
  There is a binary version available but the source version is later:
            binary source needs_compilation
fitdistrplus  1.1-3  1.1-5             FALSE

installing the source package 'fitdistrplus'

Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'geoR' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'gstat' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'MASS' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'moments' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'poweRlaw' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'RFOC' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
also installing the dependencies 'spatstat.geom', 'spatstat.core', 'spatstat.linnet', 'spa
tstat.utils'

  There are binary versions available but the source versions are later:
               binary source needs_compilation
spatstat.geom    2.1-0  2.2-2              TRUE
spatstat.core    2.1-2  2.3-0              TRUE
spatstat.linnet  2.1-1  2.3-0              TRUE
spatstat.utils   2.1-0  2.2-0              TRUE
spatstat         2.1-0  2.2-0              TRUE

  Binaries will be installed
package 'spatstat.geom' successfully unpacked and MD5 sums checked
```

```
package 'spatstat.core' successfully unpacked and MD5 sums checked
package 'spatstat.linnet' successfully unpacked and MD5 sums checked
package 'spatstat.utils' successfully unpacked and MD5 sums checked
package 'spatstat' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'ADGofTest' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'reshape' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
Installing package into 'E:/OneDrive/Documentos/R/win-library/3.6'
(as 'lib' is unspecified)
package 'sp' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\danie\AppData\Local\Temp\RtmpERJ58l\downloaded_packages
```

Despues de instalar los paquetes debemos cargarlos de la siguiente forma:

In [1]:

```r
root_dir<-getwd()

setwd(root_dir)

#### Load Packages ####
library(actuar)
library(Rcpp)
library(maps)
library(mapproj)
library(fields)
library(fitdistrplus)
library(geoR)
library(gstat)
library(MASS)
library(moments)
library(poweRlaw)
library(RFOC)
library(spatstat)
library(ADGofTest)
library(reshape)
library(sp)
```

```
Attaching package: 'actuar'


The following objects are masked from 'package:stats':

    sd, var


The following object is masked from 'package:grDevices':

    cm


Loading required package: spam
```

```
Loading required package: dotCall64

Loading required package: grid

Spam version 2.7-0 (2021-06-25) is loaded.
Type 'help( Spam)' or 'demo( spam)' for a short introduction
and overview of this package.
Help for individual functions is also obtained by adding the
suffix '.spam' to the function name, e.g. 'help( chol.spam)'.


Attaching package: 'spam'


The following objects are masked from 'package:base':

    backsolve, forwardsolve


Loading required package: viridis

Loading required package: viridisLite


Attaching package: 'viridis'


The following object is masked from 'package:maps':

    unemp


See https://github.com/NCAR/Fields for
 an extensive vignette, other supplements and source code

Loading required package: MASS

Loading required package: survival

------------------------------------------------------------
 Analysis of Geostatistical Data
 For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
 geoR version 1.8-1 (built on 2020-02-08) is now loaded
------------------------------------------------------------


Loading required package: spatstat.data

Loading required package: spatstat.geom

spatstat.geom 2.2-2


Attaching package: 'spatstat.geom'


The following object is masked from 'package:MASS':

    area


The following object is masked from 'package:actuar':
```

```
    discretise


Loading required package: spatstat.core

Loading required package: nlme

Loading required package: rpart

spatstat.core 2.3-0


Attaching package: 'spatstat.core'


The following object is masked from 'package:gstat':

    idw


Loading required package: spatstat.linnet

spatstat.linnet 2.3-0


spatstat 2.2-0       (nickname: 'That's not important right now')
For an introduction to spatstat, type 'beginner'
```

Comprobamos que todos los paquetes hayan sido cargados, si es asi, cargaremos las funciones. Estas nos permitirán obtener los graficos, modelos, etc.

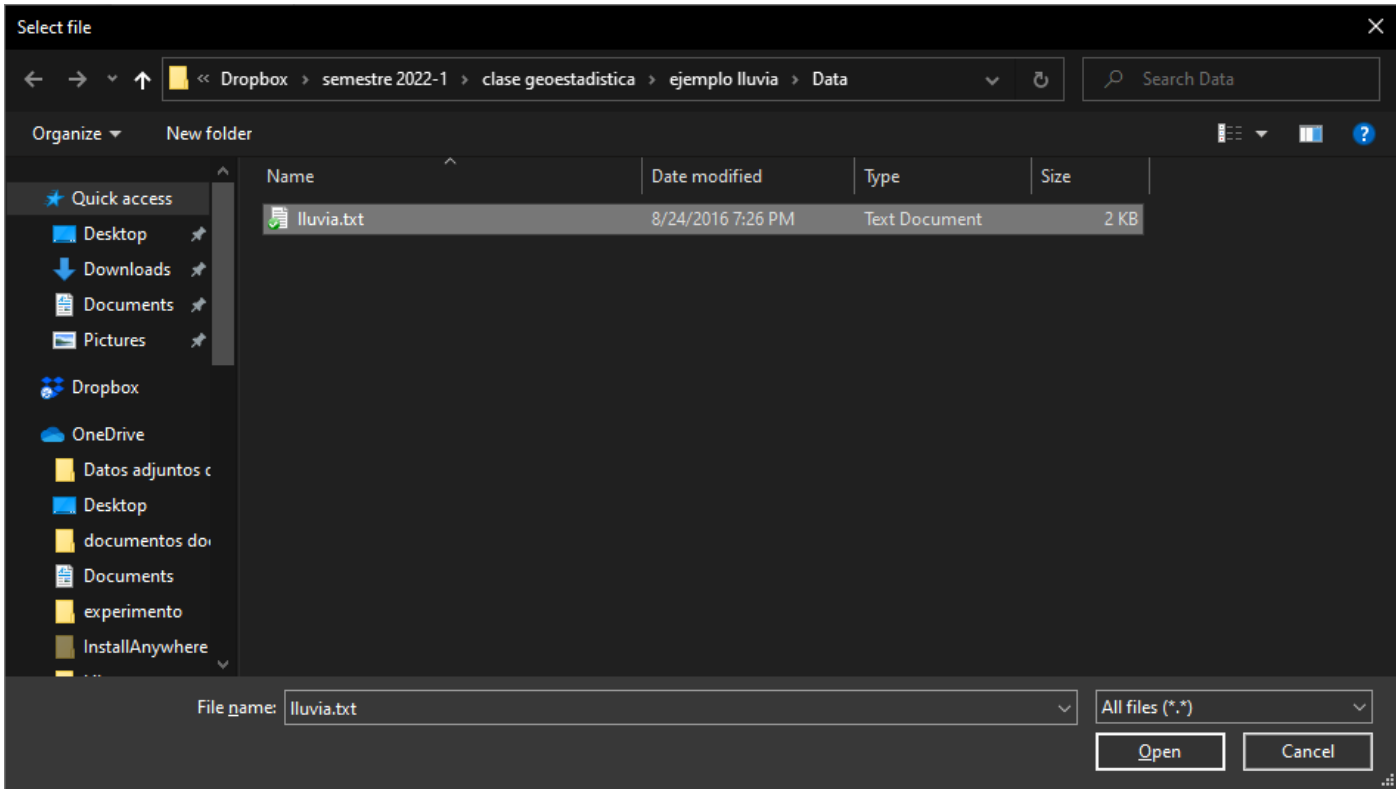
In [2]:

```
#root_dir<-getwd()

function_dir<-paste(root_dir,"/Functions",sep="")

setwd(function_dir)

source("AllModel.R", encoding='ISO-8859-1')
source("BasicStats.R", encoding='ISO-8859-1')
source("BestModel.R", encoding='ISO-8859-1')
source("BestModel.R")
source("BestModelName.R")
source("CDF.R")
source("CoKrigingOrd.R")
source("CoKrigingOrdAnis.R")
source("CrossValidation.R")
source("CrossValidation2.R")
source("CrossVariograma.R")
source("DEspacial.R", encoding='ISO-8859-1')
source("Distance.R")
source("Estadisticas.R")
source("EyeModel.R", encoding='ISO-8859-1')
source("FitDistribution.R", encoding='ISO-8859-1')
source("GDEspacial.R", encoding='ISO-8859-1')
source("GDirecciones.R", encoding='ISO-8859-1')
source("GNormal.R", encoding='ISO-8859-1')
source("hist2.R")
source("HistBoxplot.R")
source("HistModel.R")
source("KrigingOrd.R", encoding='ISO-8859-1')
source("KrigingOrdAnis.R", encoding='ISO-8859-1')
source("ModelVariogram.R")
source("Modelo.R")
source("Outliers.R")
source("OutliersCount.R")
source("OutliersCountTwo.R")
source("OutliersPos.R")
source("OutliersTwo.R")
source("PPplot.R")
source("QQplot.R")
source("RangoParams.R")
source("Regresion.R")
source("ScatterPlot.R")
source("Tendencia.R")
source("Transformacion.R")
source("Trend.R")
source("Val_Estadisticos.R", encoding='ISO-8859-1')
source("Validacion.R", encoding='ISO-8859-1')
source("ValidacionCross.R", encoding='ISO-8859-1')
source("Variograma.R")
source("Variograma4D.R", encoding='ISO-8859-1')
source("hist2.R")
source("scaterplot.R")
source("scaterplotReg.R")
#source("PlotGridCells.R")

setwd(root_dir)
```

Carga de datos.

Ahora tenemos que cargar los datos de cada variable con su respectiva posicion espacial en coordenadas UTM. Es importante que cada columna tenga su propio encabezado, así será fácil localizarlos e indexarlos.

Para seleccionar el archive que contiene la información que necesitamos, ejecutamos el comando “read.table”, el cual contiene las siguientes tres instrucciones: file=file.choose(), esta instrucción indica que quieres seleccionar el archivo usando una ventana emergente similar a la mostrada en la siguiente imagen:



header=TRUE indica que las columnas tienen encabezado y na.strings="-999.25" es una condicional para que cualquier celda nula sea llenada con el número -999.25.

In [3]:
```
Data_File_Burb <- read.csv(file=file.choose(),header=T,na.strings="-999.25")
```

Para ordenar los resultados necesitamos crear una carpeta que usemos especificamente para el analísis exploratorio de datos (AED), ahi se almacenarán tablas e imagenes, esnto lo hacemos con el comando "dir.create", donde le indicaremos la ruta donde se creará la carpeta AED.

Nota: no es necesario ejecutar esta linea mas de una vez, de lo contrario R Studio mostrará "Warning message in dir.create(paste(getwd(), "/Results/AED", sep = "")) already exists"

In [4]:
```
dir.create(paste(getwd(),"/Results/Burb", sep=""))

result_dir<-paste(root_dir,"/Results/Burb",sep="")
```

```
Warning message in dir.create(paste(getwd(), "/Results/Burb", sep = "")):
"'C:\Users\danie\Dropbox\semestre 2022-1\clase geoestadistica\ejemplo GAERM\Results\Burb'
already exists"
```

In [5]:
```
# Creates a folder to store results for AED
dir.create(paste(getwd(),"/Results/Burb/AED", sep=""))

aed_dir<-paste(result_dir,"/AED",sep="")
```

```
Warning message in dir.create(paste(getwd(), "/Results/Burb/AED", sep = "")):
"'C:\Users\danie\Dropbox\semestre 2022-1\clase geoestadistica\ejemplo GAERM\Results\Burb\A
ED' already exists"
```

Analisis exploratorio de datos.

Como se mostró en clase, el objetivo del análisis exploratorio es examinar las variables aleatorias disponibles y establecer si estas cumplen con los supuestos que requiere la estimación. Por lo tanto, debemos verificar su normalidad, linealidad, homocedasticidad, identificar los valores atípicos (outliers) y evaluar el impacto que tendrán estos valores durante el analisis variografico y por supuesto, la estimación.

Para este ejemplo las variables son los valores de la porosidad (phi_per) y la permeabilidad (perm_mD), los cuales tienen una distribución espacial en coordenadas UTM.

Después de cargar el archivo con la información y asignarle el nombre "Data_File_Burb", necesitamos las variables aleatorias y su posición espacial. Esto lo podemos hacer de la siguiente forma:

In [6]:
```
XCoord<-Data_File_Burb$Easting_ft       #Coordenada UTM en x
YCoord<-Data_File_Burb$Northing_ft      #Coordenada UTM en y
phi_per<-Data_File_Burb$phi_percent     #variable con la información de la porosidad
perm_mD<-Data_File_Burb$k_mD            #variable con la información de la permeabilidad
```

Ya que tenemos las variables necesitamos saber sobre sus estadigrafos, esto lo podemos calcular usando la función "estadisticas". Es importante mencionar que los valores calculados en este paso se usarán en los graficos.

In [7]:
```
XCoord_Stat<-Estadisticas(XCoord)
YCoord_Stat<-Estadisticas(YCoord)
phi_per_Stat<-Estadisticas(phi_per)
perm_mD_Stat<-Estadisticas(perm_mD)
```

Analisis estadístico univariado.

Para la interpretacion estadística univariada comenzaremos dos elementos: la tabla con los valores estadísticos y el histograma con boxplot.

Para obtener la tabla con los valores estadisticos, usamos la funcion "Val_Estadisticos".

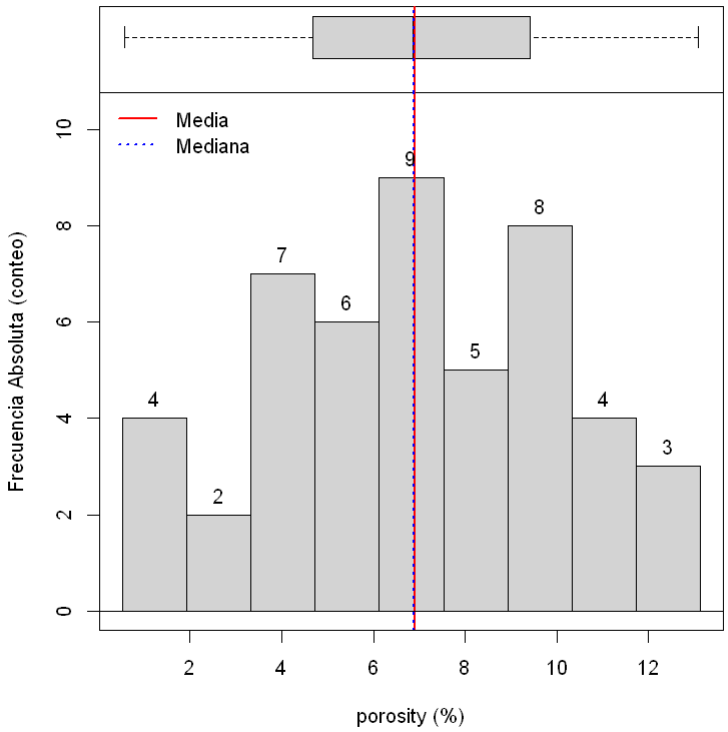
In [8]:
```
Data_File_Burb_Stat <- Val_Estadisticos(Data_File_Burb)
write.csv(Data_File_Burb_Stat , file = paste(aed_dir,"/Data_File_Burb_Stat.csv",sep=""))
print(Data_File_Burb_Stat[,3:4])
```

```
                          k_mD phi_percent
No_muestras       4.800000e+01    48.00000
Minimo            7.400000e+00     0.58200
Cuartil_1er       1.002450e+03     4.70125
Mediana           3.482205e+03     6.88800
Media             6.818245e+03     6.90763
Cuartil_3er       7.743625e+03     9.31300
Maximo            3.634740e+04    13.07300
Rango             3.634000e+04    12.49100
Rango_Intercuartil 6.741175e+03     4.61175
Varianza          8.353271e+07     9.92476
Desv_Estandar     9.139623e+03     3.15036
Simetria          1.835790e+00    -0.05236
Curtosis          5.626030e+00     2.28307
```

Analisis estadístico univariado para la porosidad (phi_per).

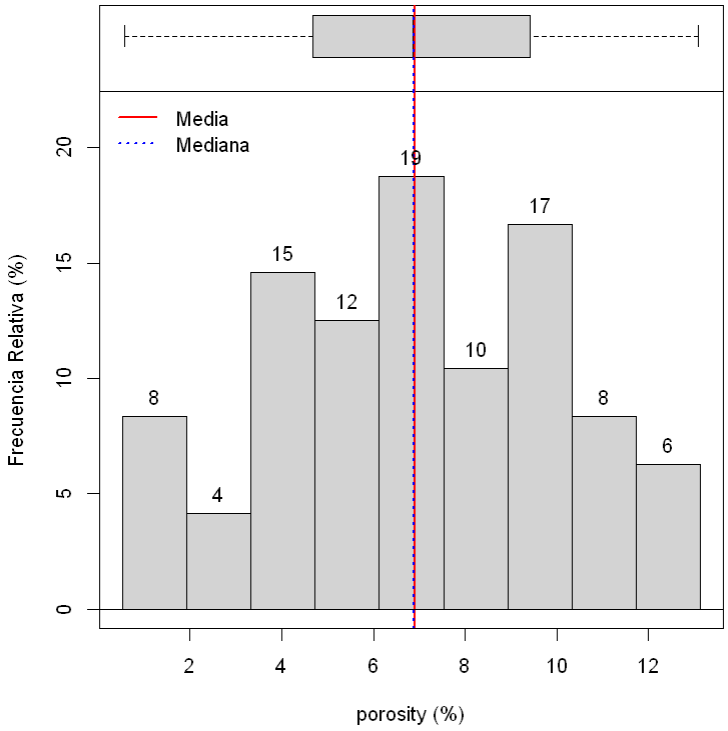
El histograma para la porosidad (phi_per) con frecuencia absoluta es:

In [9]:
```
HistBoxplot(x=phi_per, mean = phi_per_Stat[5,2], median = phi_per_Stat[4,2], main ="",
            xlab = "porosity (%)", ylab = "Frecuencia Absoluta (conteo)", AbsFreq = TRUE,
            nbin = 9)
```



Y el histograma de la porosidad con frecuencia relativa es:

In [10]:
```
HistBoxplot(x=phi_per, mean = phi_per_Stat[5,2], median = phi_per_Stat[4,2], main ="",
            xlab = "porosity (%)", ylab = "Frecuencia Relativa (%)", AbsFreq = FALSE, Per
            nbin = 9)
```



Analizando los histogramas y los estadígrafos de la variable de la porosidad (phi_per) tiene una diferencia entre la media y la mediana de 0.0.1963, su coeficiente de asimetría es de -0.05236, lo cual significa que la variable es ligeramente asimétrica. Esto se confirma con los histogramas, los cuales muestran que la asimetría es positiva. También podemos notar que el boxplot muestra no muestra valores atípicos. El valor de la curtosis es de 2.28307, lo cual nos indica que es planicúrtica.

Para saber si hay valores atípicos que no logramos ver con el boxplot usamos la función "OutliersPos".

In [11]:
```
phi_per_outliers<-OutliersPos(phi_per)
Data_File_Burb[phi_per_outliers,c(1,2,3)]
```

A data.frame: 0 × 3

Easting_ft	Northing_ft	k_mD
<int>	<int>	<dbl>

Como podemos notar, no existen valores atípicos. Dado que la variable aleatoria tiene asimetría ligera, no es necesario hacer alguna transformación.

Análisis estadístico univariado para la permeabilidad (perm_mD).

Ahora se hace el mismo análisis estadístico a los datos obtenidos de la permeabilidad (perm_mD). Empezamos obteniendo los valores estadísticos.

In [12]:
```
perm_mD_Stat <- Estadisticas(perm_mD)
perm_mD_Stat
```

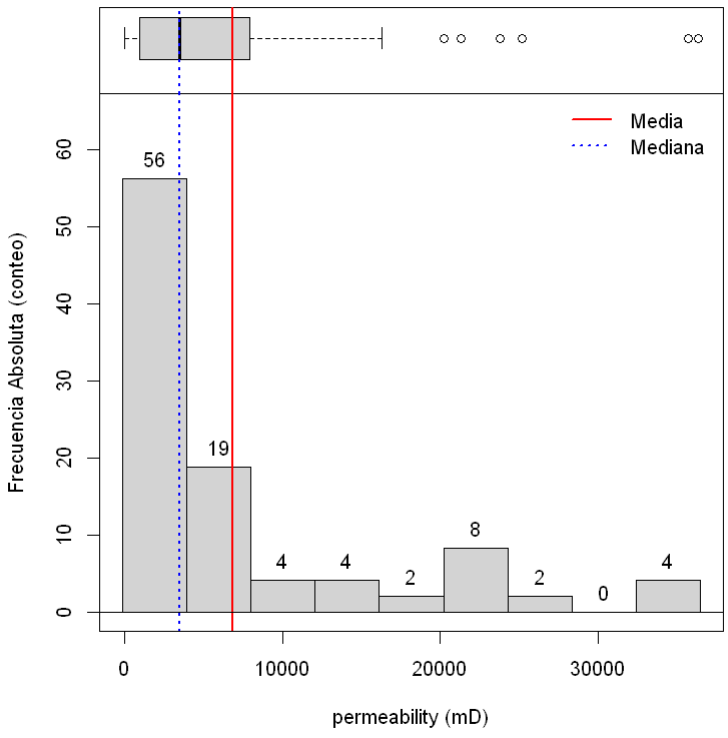
A data.frame: 14 × 2

Statistics	Values
<chr>	<dbl>

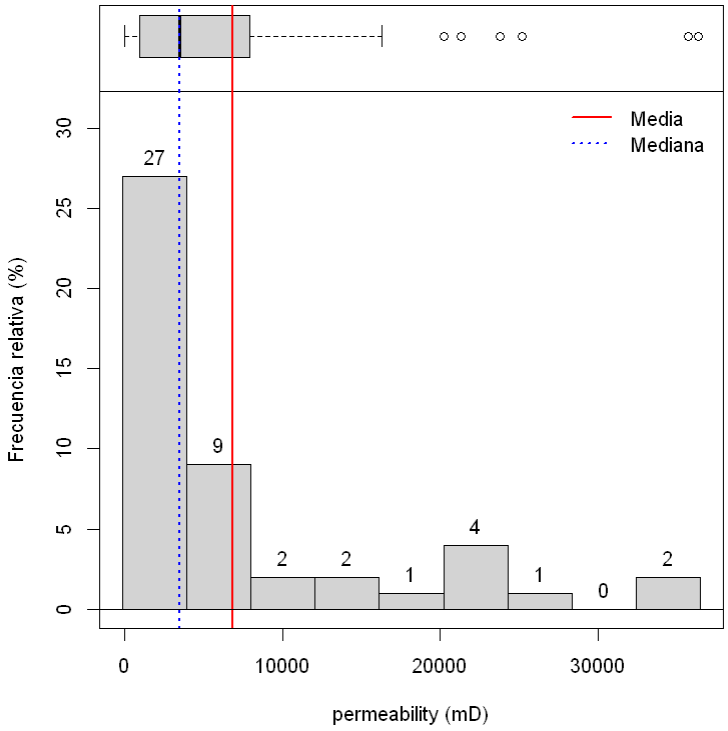
	Statistics	Values
	<chr>	<dbl>
muestras	n	4.800000e+01
minimos	Minimum	7.400000e+00
cuantiles1	1st. Quartile	1.002450e+03
medianas	Median	3.482205e+03
medias	Mean	6.818245e+03
cuantiles3	3rd. Quartile	7.743625e+03
maximos	Maximum	3.634740e+04
rangos	Rank	3.634000e+04
rangosInt	Interquartile Rank	6.741175e+03
varianzas	Variance	8.353271e+07
desvs	Standard Deviation	9.139623e+03
CVs	Variation Coeff.	1.340500e+00

Y su respectivo histograma.

In [13]:
```
HistBoxplot(x=perm_mD, mean = perm_mD_Stat[5,2], median = perm_mD_Stat[4,2], main ="",
            xlab = "permeability (mD)", ylab = "Frecuencia Absoluta (conteo)", AbsFreq = T
            nbin = 9)
```



In [14]:
```
HistBoxplot(x=perm_mD, mean = perm_mD_Stat[5,2], median = perm_mD_Stat[4,2], main ="",
            xlab = "permeability (mD)", ylab = "Frecuencia relativa (%)", AbsFreq = TRUE,
            nbin = 9)
```



Podemos notar que la diferencia entre la media y la mediana es de 3336.4, lo cual nos indica que la variable es asimétrica positiva y el histograma nos confirma esta información, también podemos notar que hay seis valores atípicos localizados a la derecha del boxplot.

Transformación de variable para la permeabilidad (perm_mD).

Dado que no se logró obtener la normalidad en esta variable, podemos usar alguna transformación.

En estadística, la transformación de datos es la aplicación de una función matemática determinista a cada punto en un conjunto de datos, es decir, cada punto de datos z_i se reemplaza con el valor transformado $y_i = f(z_i)$, donde f es una función.

Las transformaciones generalmente se aplican para que los datos parezcan cumplir más con los supuestos de un procedimiento de inferencia estadística que se aplicará, o para mejorar la interpretabilidad o la apariencia de los gráficos. Las transformaciones generalmente se aplican para que los datos parezcan cumplir más con los supuestos de un procedimiento de inferencia estadística que se aplicará, o para mejorar la interpretabilidad o la apariencia de los gráficos.

Las razones más comunes para aplicar una transformación son:

- Reducir la asimetría.
- Lograr relaciones de dependencia lineales o cuasi lineales
- Conveniencia.

Las transformaciones más comunes son:

Asimetrías positivas	**Ecuación**	**Asimetrías negativas**	**Ecuación**
Raíz cuadrada	$v_{at} = \sqrt{v_a}$	Potencias	$v_{at} = v_a^n$

Asimetrías positivas	Ecuación	Asimetrías negativas	Ecuación
Logarítmica	$v_{at} = Log(v_a)$	Arcseno	$v_{at} = arcsen(v_a)$
Recíproca	$v_{at} = \frac{1}{v_a}$	Exponencial	$v_{at} = exp(v_a)$

Donde v_a es la variable aleatoria y v_{at} es la variable aleatoria transformada.

Transformación logarítmica.

In [15]:
```
Data_File_Burb$perm_mD_Log <- log(perm_mD)
perm_mD_Log <- log(perm_mD)
```

Y calculamos sus estadígrafos.

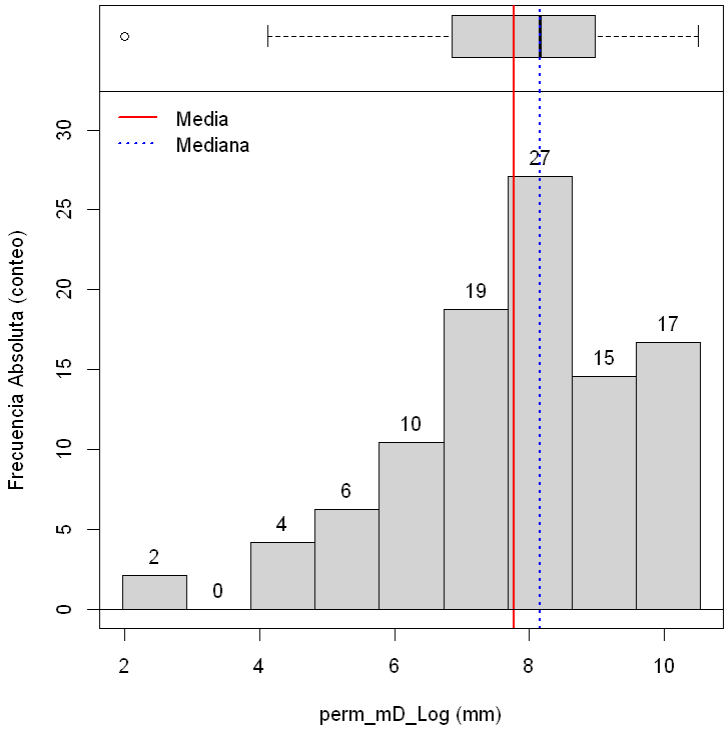
In [16]:
```
perm_mD_Log_Stat <- Estadisticas(perm_mD_Log)
write.csv(perm_mD_Log_Stat , file = paste(aed_dir,"/perm_mD_Log_Stat.csv",sep=""))
perm_mD_Log_Stat
```

A data.frame: 14 × 2

	Statistics	Values
	<chr>	<dbl>
muestras	n	48.0000
minimos	Minimum	2.0015
cuantiles1	1st. Quartile	6.9063
medianas	Median	8.1554
medias	Mean	7.7777
cuantiles3	3rd. Quartile	8.9540
maximos	Maximum	10.5009
rangos	Rank	8.4994
rangosInt	Interquartile Rank	2.0477
varianzas	Variance	3.2156
desvs	Standard Deviation	1.7932
CVs	Variation Coeff.	0.2306
simetrias	Skewness	-0.8484
curtosiss	Kurtosis	3.8748

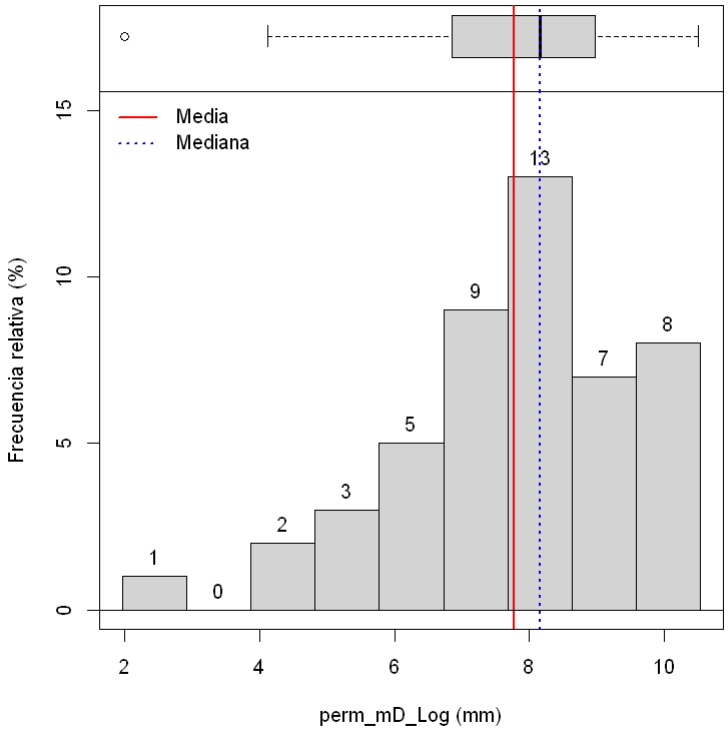
Los histogramas de la transformación logarítmica son los siguientes:

In [17]:
```
HistBoxplot(x=perm_mD_Log, mean = perm_mD_Log_Stat[5,2], median = perm_mD_Log_Stat[4,2], r
            xlab = "perm_mD_Log (mm)", ylab = "Frecuencia Absoluta (conteo)", AbsFreq = FA
            nbin = 9)
```



In [18]:

```
HistBoxplot(x=perm_mD_Log, mean = perm_mD_Log_Stat[5,2], median = perm_mD_Log_Stat[4,2], r
            xlab = "perm_mD_Log (mm)", ylab = "Frecuencia relativa (%)", AbsFreq = TRUE, b
            nbin = 9)
```



La diferencia entre la media y la mediana pasó de 3336.4 a 0.38, lo cual es muy bajo, El histograma presenta asimetría negativa. El boxplot muestra valores atípicos a la izquierda del grafico, nos aseguraremos que esto sea cierto usando la función "OutliersPos".

In [19]:
```
perm_mD_Log_outliers<-OutliersPos(perm_mD_Log)
Data_File_Burb[perm_mD_Log_outliers,c(1,2,3)]
```

A data.frame: 1 × 3

	Easting_ft	Northing_ft	k_mD
	<int>	<int>	<dbl>
9	6525	10141	7.4

Como podemos ver, hay un valor atípico, el cual vamos a retirar.

NOTA: retirar los valores atípicos no significa que no usaremos más adelante, los necesitar

In [20]:
```
perm_mD_Log_out<-perm_mD_Log[-perm_mD_Log_outliers]
```

Después de retirar los valores atípicos, calculamos los estadígrafos de esta nueva variable aleatoria.

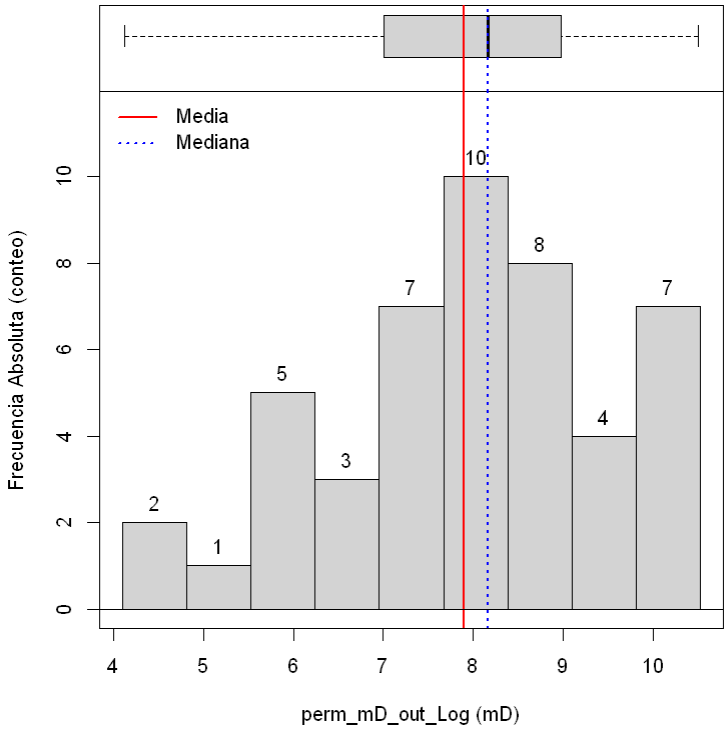
In [21]:
```
perm_mD_Log_out_Stat<-Estadisticas(perm_mD_Log_out)
write.csv(perm_mD_Log_out_Stat , file = paste(aed_dir,"/perm_mD_Log_out_Stat.csv",sep="")
perm_mD_Log_out_Stat
```

A data.frame: 14 × 2

	Statistics	Values
	<chr>	<dbl>
muestras	n	47.0000
minimos	Minimum	4.1239
cuantiles1	1st. Quartile	7.0100
medianas	Median	8.1663
medias	Mean	7.9006
cuantiles3	3rd. Quartile	8.9744
maximos	Maximum	10.5009
rangos	Rank	6.3770
rangosInt	Interquartile Rank	1.9643
varianzas	Variance	2.5447
desvs	Standard Deviation	1.5952
CVs	Variation Coeff.	0.2019
simetrias	Skewness	-0.4224
curtosiss	Kurtosis	2.6013

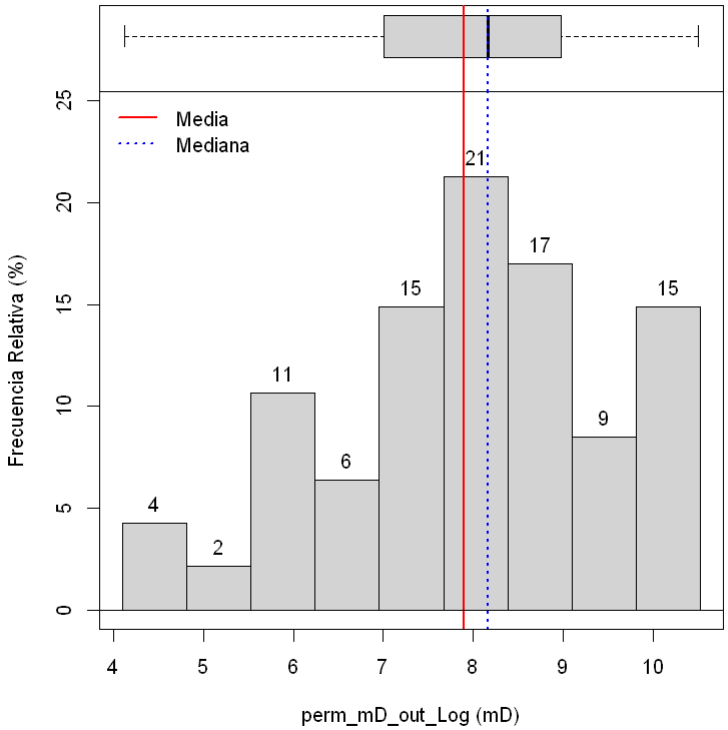
También graficamos sus respectivos histogramas.

In [22]:
```
HistBoxplot(x=perm_mD_Log_out, mean = perm_mD_Log_out_Stat[5,2], median = perm_mD_Log_out
            xlab = "perm_mD_out_Log (mD)", ylab = "Frecuencia Absoluta (conteo)", AbsFreq
            nbin = 9)
```



In [23]:

```
HistBoxplot(x=perm_mD_Log_out, mean = perm_mD_Log_out_Stat[5,2], median = perm_mD_Log_out_
            xlab = "perm_mD_out_Log (mD)", ylab = "Frecuencia Relativa (%)", AbsFreq = FA
            nbin = 9)
```



La diferencia entre la media y la mediana pasó de 0.38 a 0.27, lo cual es bajo, El histograma presenta asimetría positiva. El boxplot no muestra valores atípicos, nos aseguraremos que esto sea cierto usando la función "OutliersPos".

In [24]:
```
perm_mD_Log_out_outliers2<-OutliersPos(perm_mD_Log_out)
print(perm_mD_Log_out_outliers2)
```

```
numeric(0)
```

Análisis estadístico bivariado.

Como pudimos notar durante el análisis exploratorio univariado, necesitamos de dos elementos para interpretar las características estadísticas de una variable: un histograma y una tabla con los valores estadísticos. Con el caso del análisis exploratorio bivariado necesitamos un diagrama de dispersión o scatterplot y los grados de dependencia.

Un diagrama de dispersión es una gráfica compuesta por pares de valores de dos variables aleatorias (x_i, y_i).

Los grados de dependencia se miden usando el coeficiente de correlación lineal de Pearson:

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} == \frac{Cov(X,Y)}{\sqrt{(Var(X)Var(Y))}}$$

El coeficiente de correlación de Spearman:

$$\rho = 1 - \frac{6\sum D^2}{N(N^2-1)}$$

Y el coeficiente de correlación de Kendall:

$$\tau = \frac{número\ de\ pares\ concordantes - número\ de\ pares\ discordantes}{\binom{n}{2}}$$

Cálculo de grados de dependencia

Para calcular estos grados de dependencia lo hacemos de la siguiente forma:

In [25]:
```
cor(phi_per , perm_mD, method = "pearson")
```

0.716875024758622

In [26]:
```
cor(phi_per , perm_mD, method = "spearman")
```

0.865935735996526

In [27]:
```
cor(phi_per , perm_mD, method = "kendall")
```

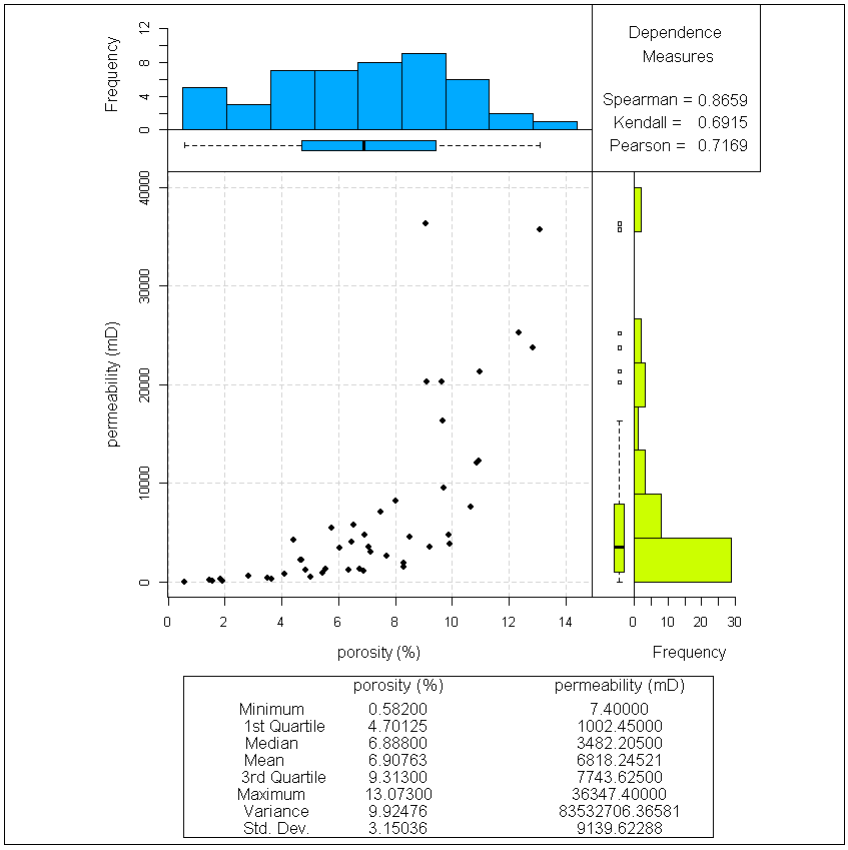
0.691489361702128

Respecto al coeficiente de Pearson, su valor es de 0.7168. Sin embargo, los valores de la correlación de Spearman (0.8659) y Kendall (0.6914) indican que el modelo indica que el modelo tiene buena dependencia.

Diagrama de dispersión.

El diagrama de dispersión se grafica de la siguiente manera:

In [28]:
```
ScatterPlot(phi_per , perm_mD, 9,
            Xmin = phi_per_Stat[2,2], Xmax = phi_per_Stat[7,2],
            Ymin = perm_mD_Stat[2,2],Ymax = perm_mD_Stat[7,2],
            XLAB = "porosity (%)", YLAB = "permeability (mD)")
```



	porosity (%)	permeability (mD)
Minimum	0.58200	7.40000
1st Quartile	4.70125	1002.45000
Median	6.88800	3482.20500
Mean	6.90763	6818.24521
3rd Quartile	9.31300	7743.62500
Maximum	13.07300	36347.40000
Variance	9.92476	83532706.36581
Std. Dev.	3.15036	9139.62288

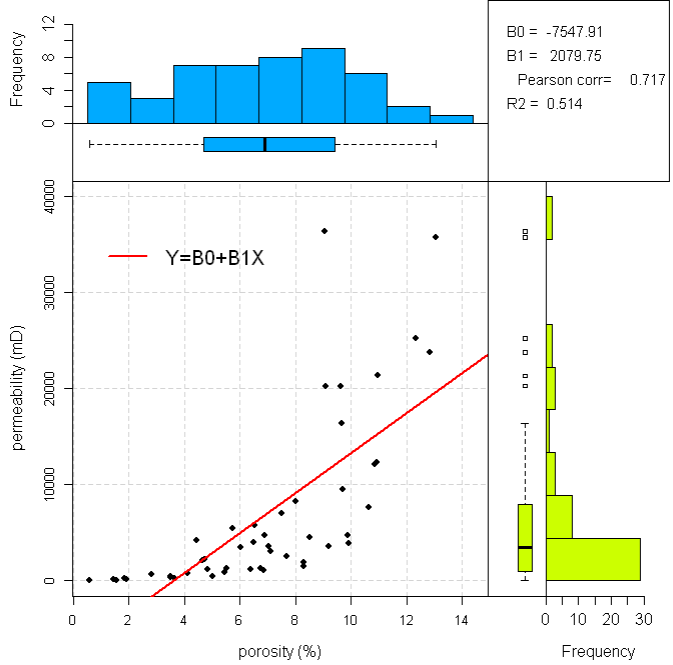
Podemos notar en el gráfico de dispersión que hay valores atípicos localizados en la esquina superior derecha, pero solo se manifiestan en el histograma de la permeabilidad, por el momento no se omitirán esos valores con el fin de saber si para el análisis exploratorio bivariado se podría considerar como valores atípicos a omitir.

Análisis de regresión lineal.

Como se mencionó en clase, la regresión trata de establecer relaciones funcionales entre variables aleatorias, en este caso, la relación se establece con una recta. Para hacer el análisis necesitamos de los parámetros de la recta y el análisis de residuos.

El gráfico de dispersión con línea de regresión se genera de la siguiente forma:

In [29]:
```
scaterplotReg(phi_per , perm_mD, 9,
              Xmin = phi_per_Stat[2,2], Xmax = phi_per_Stat[7,2],
              Ymin = perm_mD_Stat[2,2],Ymax = perm_mD_Stat[7,2],
              XLAB = "porosity (%)", YLAB = "permeability (mD)")
```



De este grafico nos interesa saber los valores de la regresión lineal y su error cuadratico, con el caso de los parámetros B_o y B_1, copiamos sus valores de la siguiente forma:

In [30]:
```
X<-phi_per
Y<-perm_mD

linear_regression <-lm(Y ~ X)

B0 <- linear_regression$coefficients[1]
B0
B1 <- linear_regression$coefficients[2]
B1
```

(Intercept): -7547.90678034324

X: 2079.75273537237

Ya que tenemos los parámetros de la recta, hacemos el calculo de los residuos.

In [31]:
```
Y_Regression <- linear_regression$fitted.values
Y_Residual <- linear_regression$residuals
```

Ya que tenemos calculados los residuos necesitamos obtener sus valores estadísticos.

In [32]:
```
Y_Residual_Stat<-Estadisticas(Y_Residual)
write.csv(Y_Residual_Stat , file = paste(aed_dir,"/perm_mD_Residual_Stat.csv",sep=""))
Y_Residual_Stat
```

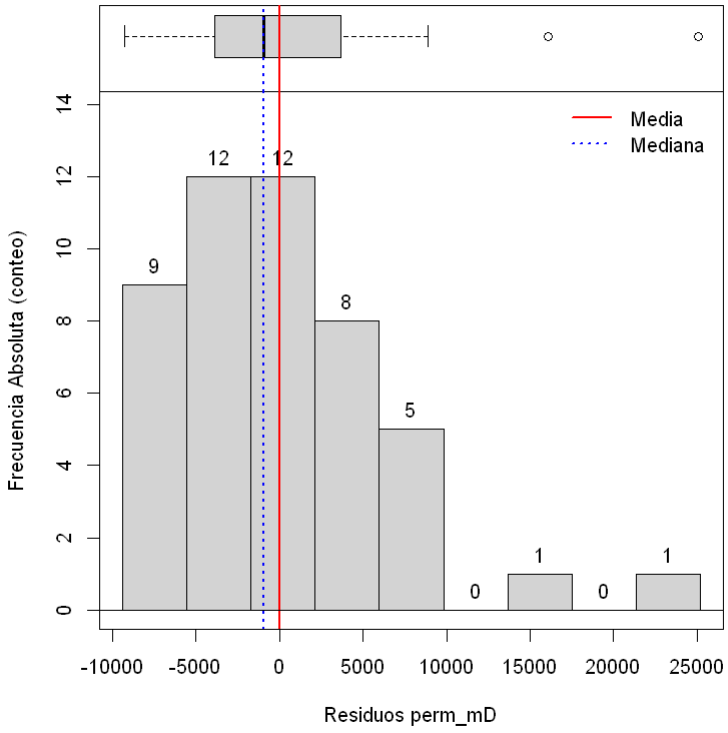
A data.frame: 14 × 2

	Statistics	Values
	<chr>	<dbl>
muestras	n	4.800000e+01

	Statistics	Values
	<chr>	<dbl>
minimos	Minimum	-9.237240e+03
cuantiles1	1st. Quartile	-3.748058e+03
medianas	Median	-9.578251e+02
medias	Mean	0.000000e+00
cuantiles3	3rd. Quartile	3.650538e+03
maximos	Maximum	2.503611e+04
rangos	Rank	3.427335e+04
rangosInt	Interquartile Rank	7.398596e+03
varianzas	Variance	4.060443e+07
desvs	Standard Deviation	6.372160e+03

Y tambien necesitamos obtener el histograma de estos residuos.

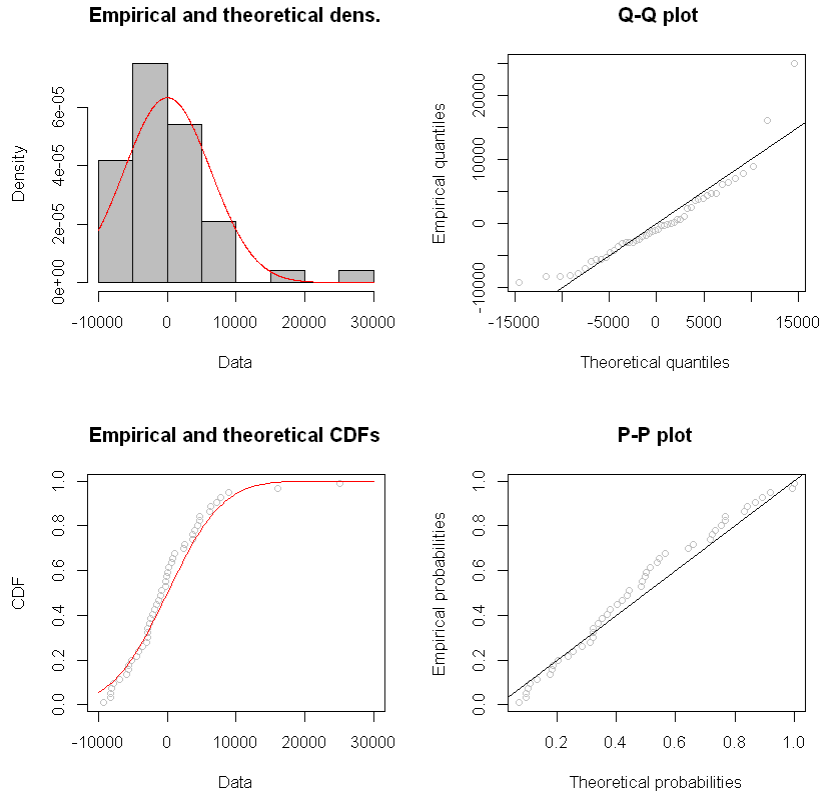
In [33]:
```
HistBoxplot(x=Y_Residual, mean = Y_Residual_Stat[5,2], median = Y_Residual_Stat[4,2], mai
            xlab = "Residuos perm_mD", ylab = "Frecuencia Absoluta (conteo)", AbsFreq = TR
```



Si analizamos los valores estadísticos y el histograma de los residuos, podemos notar que el valor esperado es de -0.0016, lo cual se podria considerar cercano a cero, su varianza es de 0.2947 y la diferencia entre la media y la mediana es de 0.1564, lo cual nos indica que tiene asimetría negativa, por lo tanto, los residuos no cumplen con todas las condiciones que demanda la regresión lineal.

In [34]:
```
FitDistr2_Residual_normal<-FitDistribution(data = Y_Residual, DISTR="norm", BREAKS = "Stu
```

```
Warning message in hist.default(data, breaks = breaks, plot = FALSE, ...):
"argument 'col' is not made use of"
```



Para confirmar que los residuos no cumplen con las condiciones de la regresión lineal podemos sobreponer el histograma con la distribución normal (figura superior izquierda), ahí podemos ver que una de las barras del histograma sobrepasa a la función de distribución. El grafico Q-Q plot (figura superior derecha) también muestra que solo las muestras de la parte central están en la recta, las muestras localizadas a los extremos están lejos de la recta esperada. La grafica comparativa entre las funciones de distribución acumulativas empírica y teórica (figura inferior izquierda) muestran un ajuste aceptable, pero no es ideal y en el caso del grafico P-P plot (figura inferior derecha) casi todas las muestras se posicionan cerca de la recta.

Ahora debemos aplicar un test de normalidad, en este caso tenemos dos opciones: hipótesis de Kolmogorov-Smirnov y la hipótesis de Anderson-Darling.

La hipótesis de Kolmogorov-Smirnov se usa para contrastar la hipótesis de normalidad, el estadístico de prueba es la máxima diferencia:

$$D = m\acute{a}x|F_n(x) - F_{va}(x)|$$

Donde $F_n(x)$ es la función de distribución paramétrica, en este caso la función normal. Y $F_{va}(x)$ es la función de la variable aleatoria.

La hipótesis de Anderson-Darling es una prueba no paramétrica que se basa en la comparación de las muestras $\mathbf{Y}$ y la función de distribución de probabilidad teórica $\mathbf{F}$. Su fórmula es:

$$S = \sum_{k=1}^{N} \frac{2k-1}{N}[ln(F(Y_k)) + ln(1 - F(Y_{N+1-k}))]$$

In [35]:

```
FD_HT_Residual_normal<-FitDistr2_Residual_normal$x
FD_HT_Residual_normal
```

A data.frame: 2 × 5

Nombre	Nivel de significancia	P-valor	Estad?stico	Decisi?n

<chr>	<chr>	<chr>	<chr>	<chr>
Kolmogorov-Smirnov	0.05	0.4362	0.1222	No rechazo H0
Anderson-Darling	0.05	0.4035	0.9167	No rechazo H0

Bajo la prueba de Kolmogorov-Smirnov podemos ver que la normalidad de los residuos es de no rechazo, es decir, se aprueba su hipótesis de normalidad, esto se puede deber a que en el grafico Q-Q plot los valores no están muy alejados de la recta esperada.

Mientras que la prueba de Anderson-Darling no rechaza la hipotesis de normalidad.

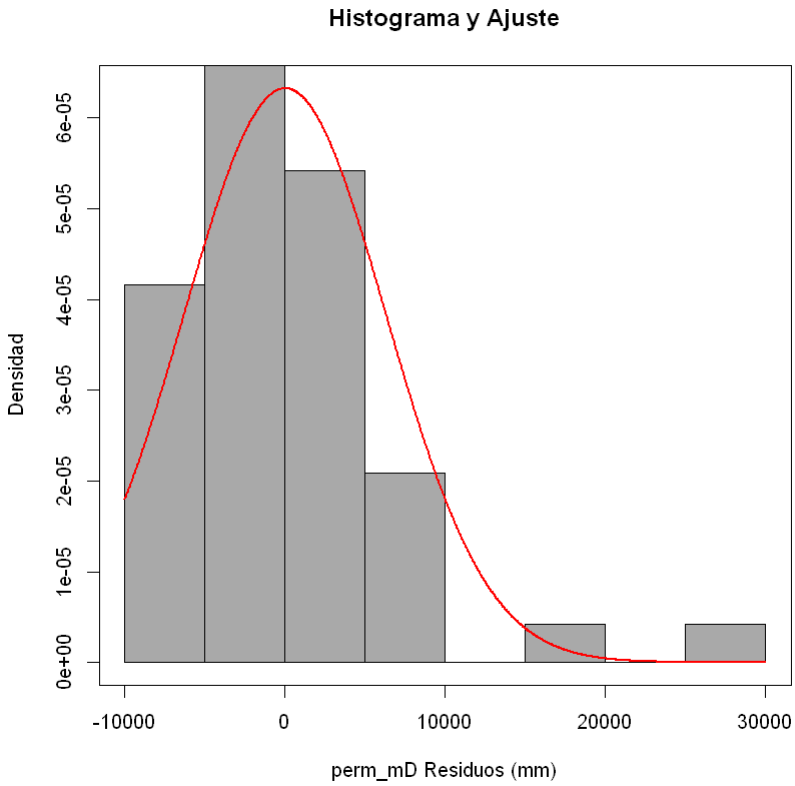
In [36]:
```
FD_FP_Residual_normal<-FitDistr2_Residual_normal$y
FD_FP_Residual_normal
```

A data.frame: 4 × 1

	Normal
	<dbl>
Media	-6.438183e-13
Desviaci?n est?ndar	6.305434e+03
M?xima Verosimilitud	-4.880691e+02
AICC	9.801381e+02

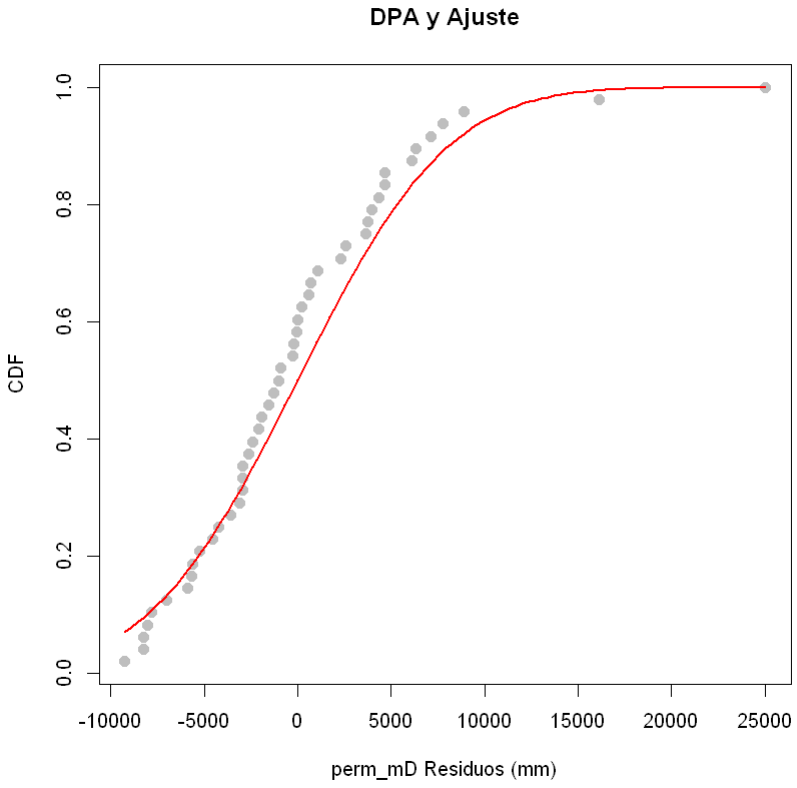
Los siguientes gráficos son los mismos que se analizaron en el vector "FitDistr2_Residual_normal", tenemos el histograma la funcion de distribución normal.

In [37]:
```
PARA_Residual_normal <- list(mean = as.numeric(FD_FP_Residual_normal[1,1]), sd = as.numer
HistModel(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, breaks = "Sturges"
          ylab = "Densidad", colCurve =  "red", col = "darkgray")
```



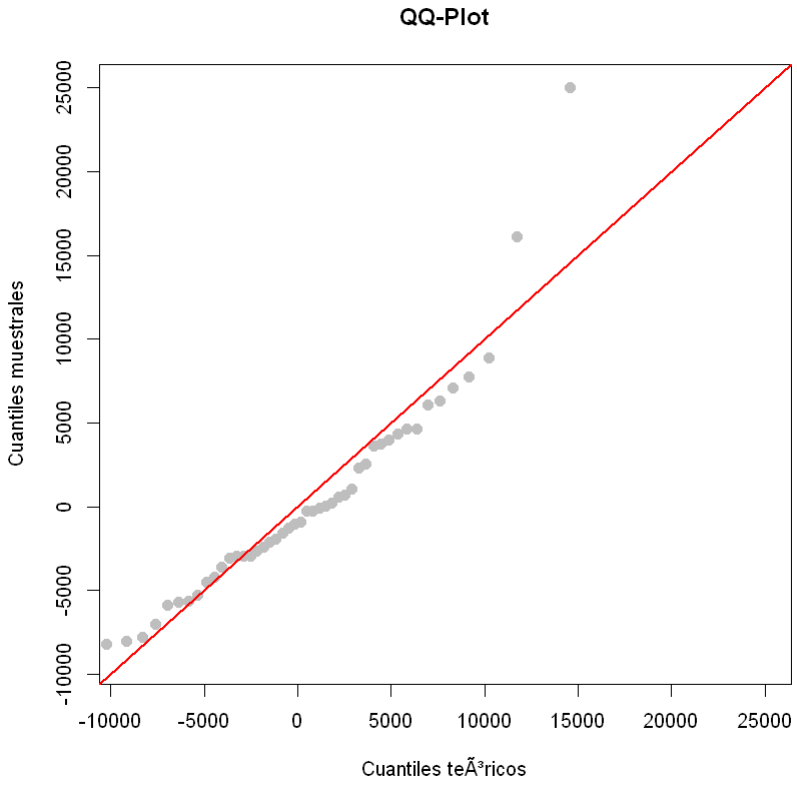
Graficamos su función de distribución acumulativa.

In [38]:
```
CDF(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main = "DPA
    lcol = "red", lwd = 2)
```



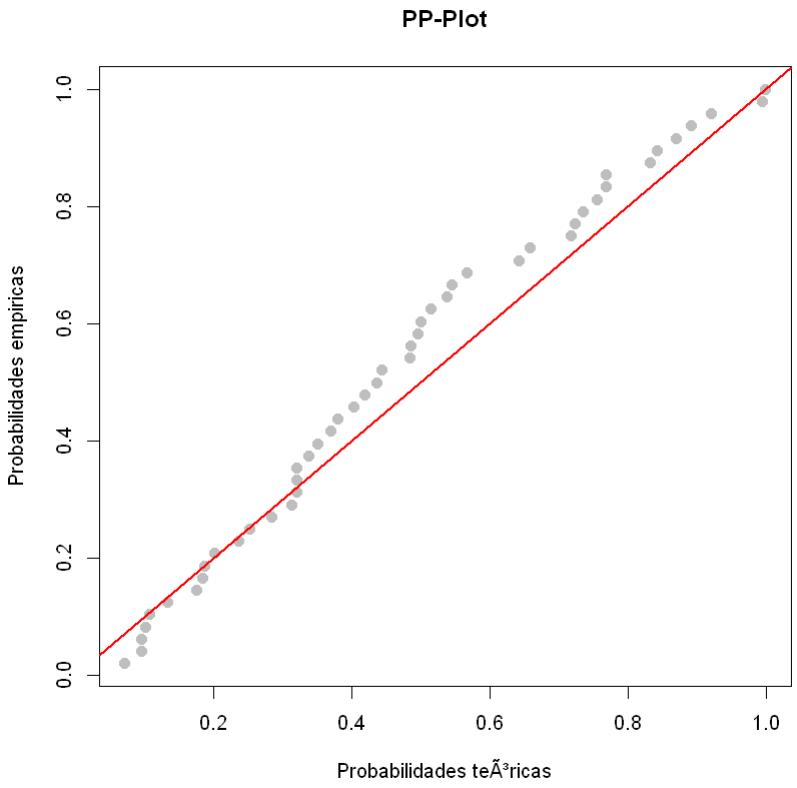
El Grafico cuantil-cuantil (Q-Q plot).

In [39]:
```
QQplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main =
       lcol = "red", lwd = 2)
```



El Grafico percentil-percentil (P-P plot).

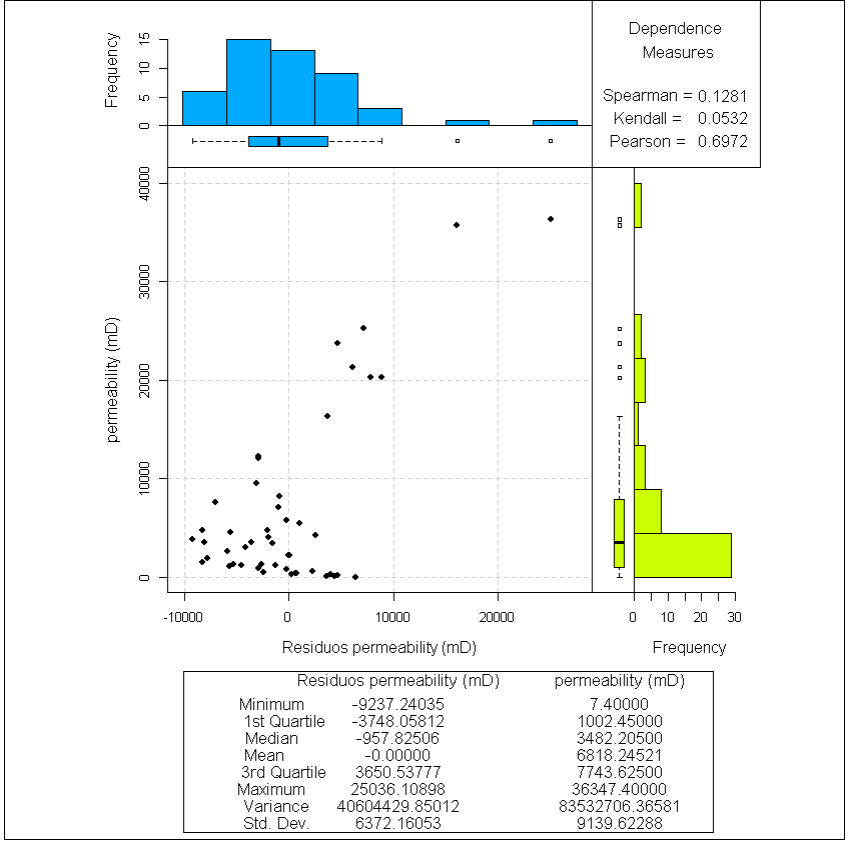
In [40]:
```
PPplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main =
       lcol = "red", lwd = 2)
```



Analisis bivariado: Y vs Y residual.

Ahora necesitamos evaluar si los valores obtenidos del pluviómetro tienen una relación con los residuos, para esto necesitamos el gráfico de dispersión. El cual generamos de la siguiente forma:

In [41]:
```
X<-Y_Residual   # Y_Residual is the independent variable

Y<-perm_mD      # perm_mD is the dependent variable

ScatterPlot(X, Y, 9,
           Xmin = Y_Residual_Stat[2,2], Xmax = Y_Residual_Stat[7,2],
           Ymin = perm_mD_Stat[2,2],Ymax = perm_mD_Stat[7,2], XLAB = "Residuos permeabili
```



	Residuos permeability (mD)	permeability (mD)
Minimum	-9237.24035	7.40000
1st Quartile	-3748.05812	1002.45000
Median	-957.82506	3482.20500
Mean	-0.00000	6818.24521
3rd Quartile	3650.53777	7743.62500
Maximum	25036.10898	36347.40000
Variance	40604429.85012	83532706.36581
Std. Dev.	6372.16053	9139.62288

En este grafico podemos notar que la medida de dependencia lineal de Pearson es de 0.1281, Spearman es de 0.0532 y Kendall es de 0.6972. Aquí podemos notar una fuerte discrepancia en los resultados dela medida de dependencia; para Spearman y Kendall tenemos una baja dependencia que podríamos considerar que la regresión cumple con su propósito, pero Pearson muestra que su dependencia es buena.

Análisis estadístico bivariado con variables transformadas.

Ahora haremos el caso donde las variables (Radar_mm, Pluv_mm) tienen transformada logarítmica. Se escogió que las variables aleatorias usen esta transformación por los resultados que se tuvieron con la variable aleatoria de las muestras obtenidas con el pluviómetro.

Grafico de dispersión.

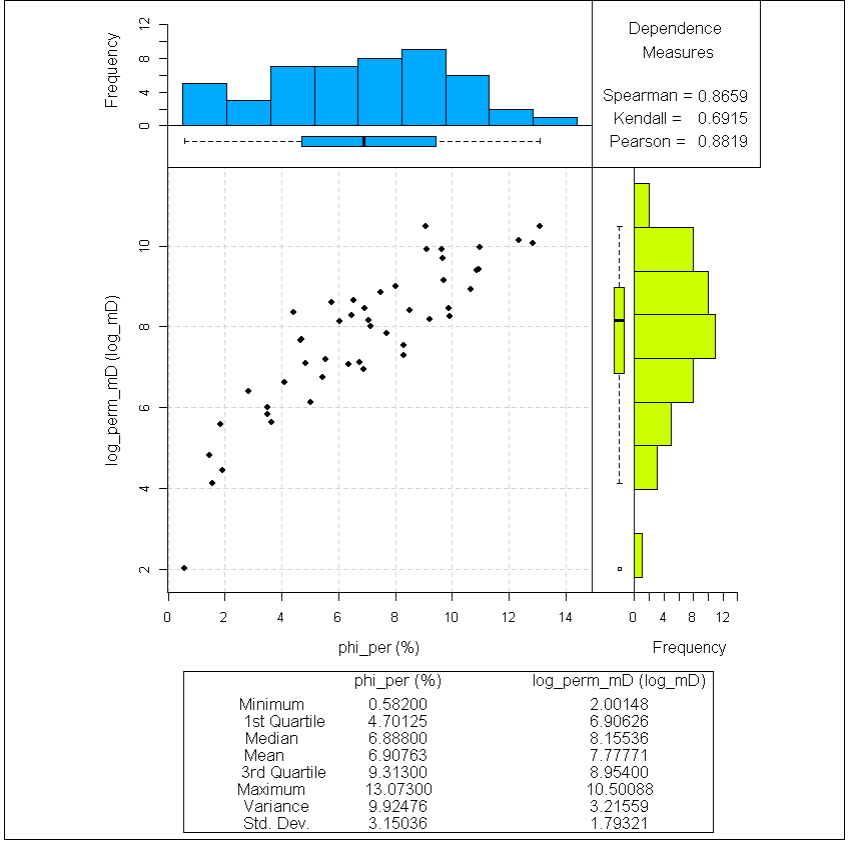
En el grafico de dispersión podemos notar que las medidas de dependencia tienen los siguientes valores:

- Spearman (0.8659)
- Kendall (0.6915)
- Pearson (0.8819)

Si comparamos las medidas de dependencia de este análisis estadístico bivariado con el anterior análisis podemos notar que el coeficiente de correlación de Pearson cambió de 0.7168 a 0.8659, mientras que los coeficientes de Spearman y Kendall se mantuvieron. esto muestra una gran ventaja al usar los coeficientes de Spearman y Kendall ya que no se ven afectados ante transformaciones.

In [42]:

```
ScatterPlot(phi_per , perm_mD_Log, 9,
            Xmin = phi_per_Stat[2,2], Xmax = phi_per_Stat[7,2],
            Ymin = perm_mD_Log_Stat[2,2],Ymax = perm_mD_Log_Stat[7,2],
            XLAB = "phi_per (%)", YLAB = "log_perm_mD (log_mD)")
```



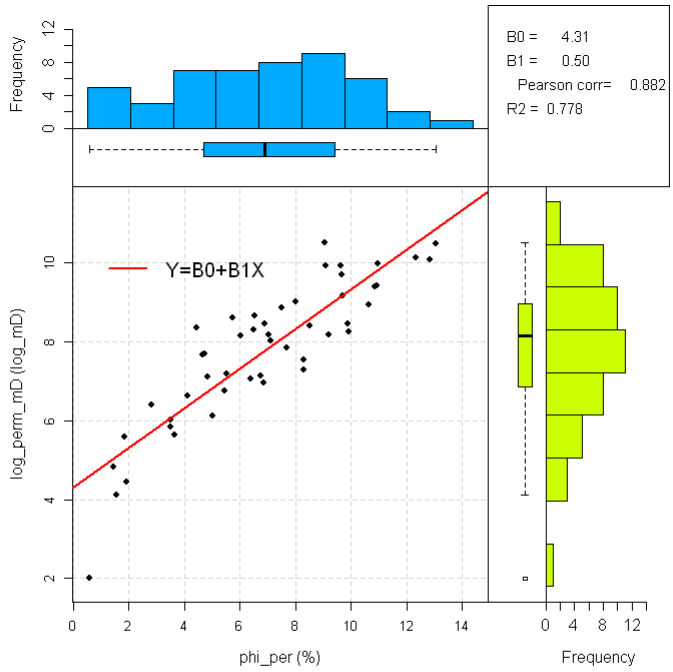
	phi_per (%)	log_perm_mD (log_mD)
Minimum	0.58200	2.00148
1st Quartile	4.70125	6.90626
Median	6.88800	8.15536
Mean	6.90763	7.77771
3rd Quartile	9.31300	8.95400
Maximum	13.07300	10.50088
Variance	9.92476	3.21559
Std. Dev.	3.15036	1.79321

Analisis de regresión lineal.

Ahora obtendremos los valores de la regresión lineal .

In [43]:

```
scaterplotReg(phi_per , perm_mD_Log, 9,
              Xmin = phi_per_Stat[2,2], Xmax = phi_per_Stat[7,2],
              Ymin = perm_mD_Log_Stat[2,2],Ymax = perm_mD_Log_Stat[7,2],
              XLAB = "phi_per (%)", YLAB = "log_perm_mD (log_mD)")
```



Procedemos a calcular los residuos y los valores de los parámetros

In [44]:
```
X<-phi_per
Y<-perm_mD_Log

linear_regression <-lm(Y ~ X)

# Linear Regression Parameters
B0 <- linear_regression$coefficients[1]
B0
B1 <- linear_regression$coefficients[2]
B1
```

(Intercept): 4.3101719606354

X: 0.501986643878787

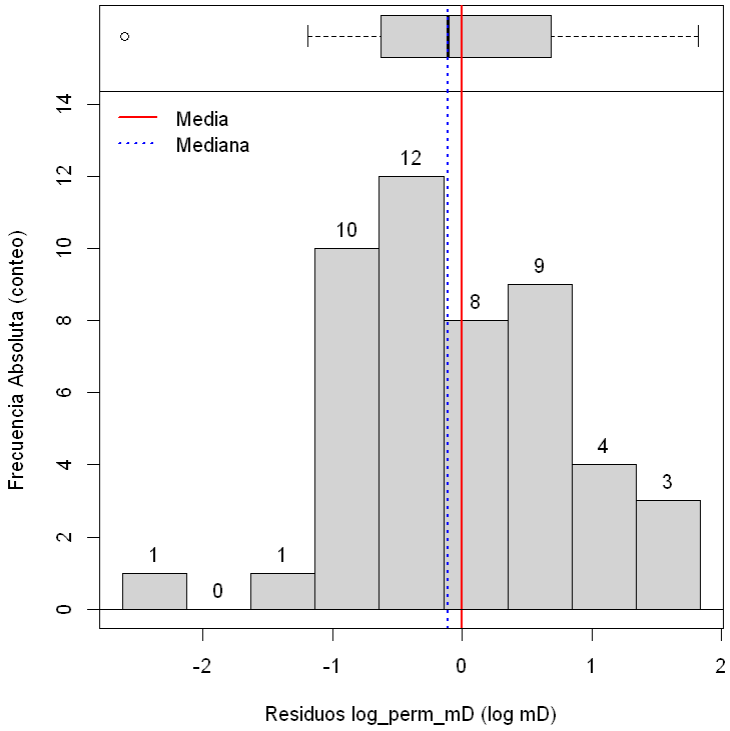
Y calculamos sus estadígrafos e histograma.

In [45]:
```
Y_Regression <- linear_regression$fitted.values
Y_Residual <- linear_regression$residuals

Y_Residual_Stat<-Estadisticas(Y_Residual)
write.csv(Y_Residual_Stat , file = paste(aed_dir,"/perm_mD_Log_Residual_Stat.csv",sep=""))
Y_Residual_Stat
```

A data.frame: 14 × 2

	Statistics	Values
	<chr>	<dbl>
muestras	n	4.800000e+01
minimos	Minimum	-2.600800e+00
cuantiles1	1st. Quartile	-6.022000e-01
medianas	Median	-1.076000e-01
medias	Mean	0.000000e+00
cuantiles3	3rd. Quartile	6.815000e-01
maximos	Maximum	1.815900e+00
rangos	Rank	4.416800e+00
rangosInt	Interquartile Rank	1.283700e+00
varianzas	Variance	7.146000e-01
desvs	Standard Deviation	8.454000e-01
CVs	Variation Coeff.	1.605428e+16
simetrias	Skewness	-1.820000e-01
curtosiss	Kurtosis	3.495000e+00

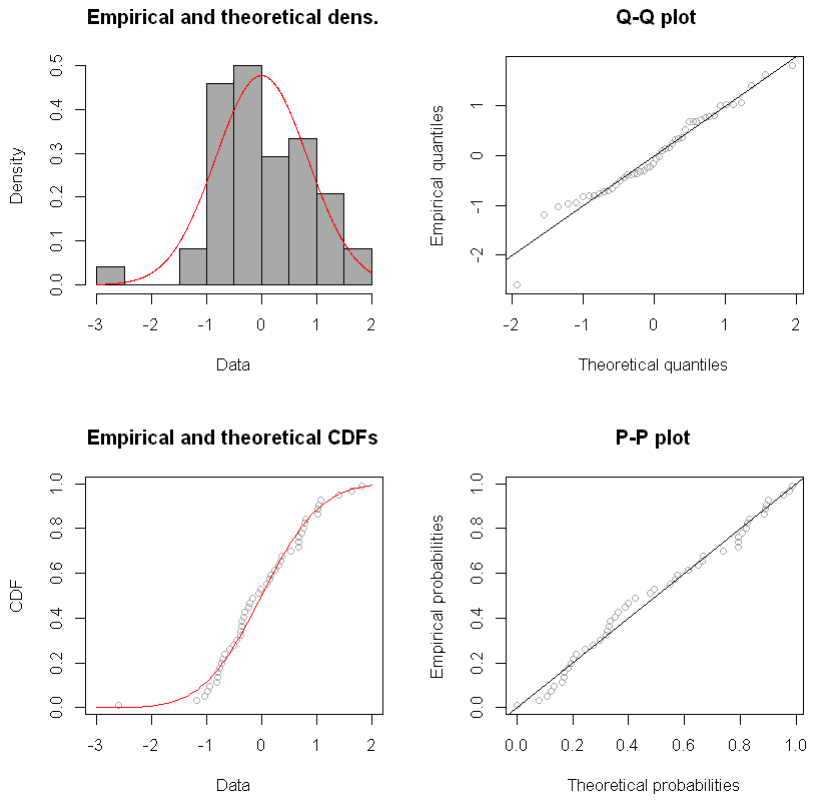
In [46]:
```
HistBoxplot(x=Y_Residual, mean = Y_Residual_Stat[5,2], median = Y_Residual_Stat[4,2], mai
            xlab = "Residuos log_perm_mD (log mD)", ylab = "Frecuencia Absoluta (conteo)"
```



En este caso podemos ver que el histograma presenta asimetría positiva con un valor atípico localizado a la izquierda. La diferencia entre la media y la mediana es de 0.11, la cual es baja.

In [47]:
```
FitDistr2_Residual_normal<-FitDistribution(data = Y_Residual, DISTR="norm", BREAKS = "Stu
```

```
Warning message in hist.default(data, breaks = breaks, plot = FALSE, ...):
"argument 'col' is not made use of"
```



Analizando los residuos podemos notar que el histograma con la distribución normal (figura superior izquierda) nos muestra que los residuos no son normales, en especial los valores localizados al centro del histograma. El

grafico Q-Q plot (figura superior derecha) también muestra quecasi todas las muestras están cerca de la recta. La grafica comparativa entre las funciones de distribución acumulativas empírica y teórica (figura inferior izquierda) muestra un buen ajuste y en el caso del grafico P-P plot (figura inferior derecha) hay muy pocas muestras alejadas a la recta. Con esto podemos concluir que los residuos posiblemente cumplen con los requisitos de la regresión lineal.

In [48]:
```
FD_HT_Residual_normal<-FitDistr2_Residual_normal$x
FD_HT_Residual_normal
```

A data.frame: 2 × 5

Nombre	Nivel de significancia	P-valor	Estad?stico	Decisi?n
<chr>	<chr>	<chr>	<chr>	<chr>
Kolmogorov-Smirnov	0.05	0.8672	0.08312	No rechazo H0
Anderson-Darling	0.05	0.8338	0.4142	No rechazo H0

Las pruebas de normalidad de Kolmogorov-Smirnov y Anderson-Darling confirman lo analizado con los graficos.

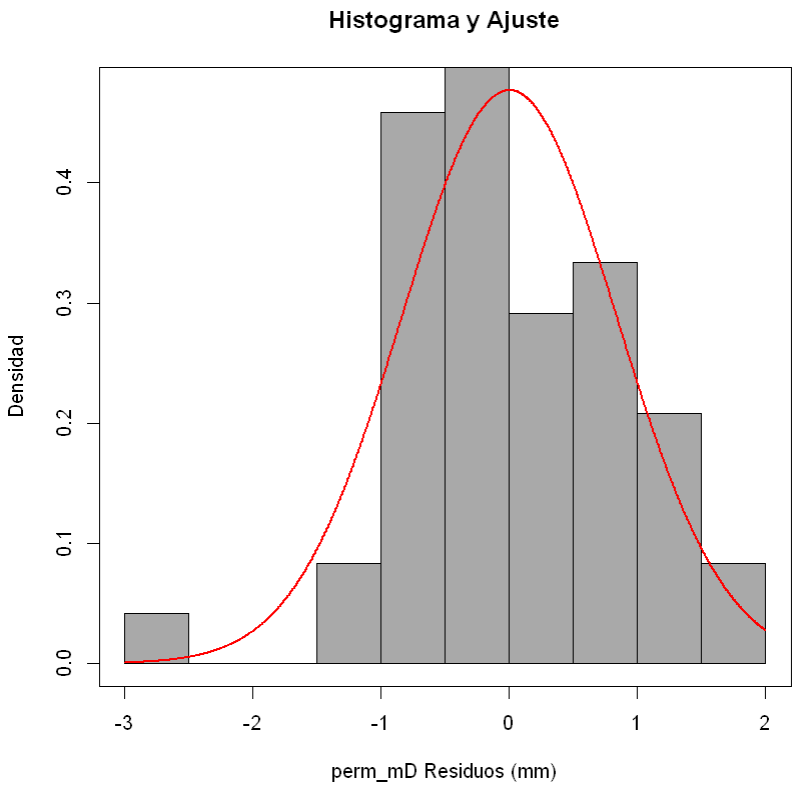
In [49]:
```
FD_FP_Residual_normal<-FitDistr2_Residual_normal$y
FD_FP_Residual_normal
```

A data.frame: 4 × 1

	Normal
	<dbl>
Media	5.265665e-17
Desviaci?n est?ndar	8.365121e-01
M?xima Verosimilitud	-5.954036e+01
AICC	1.230807e+02

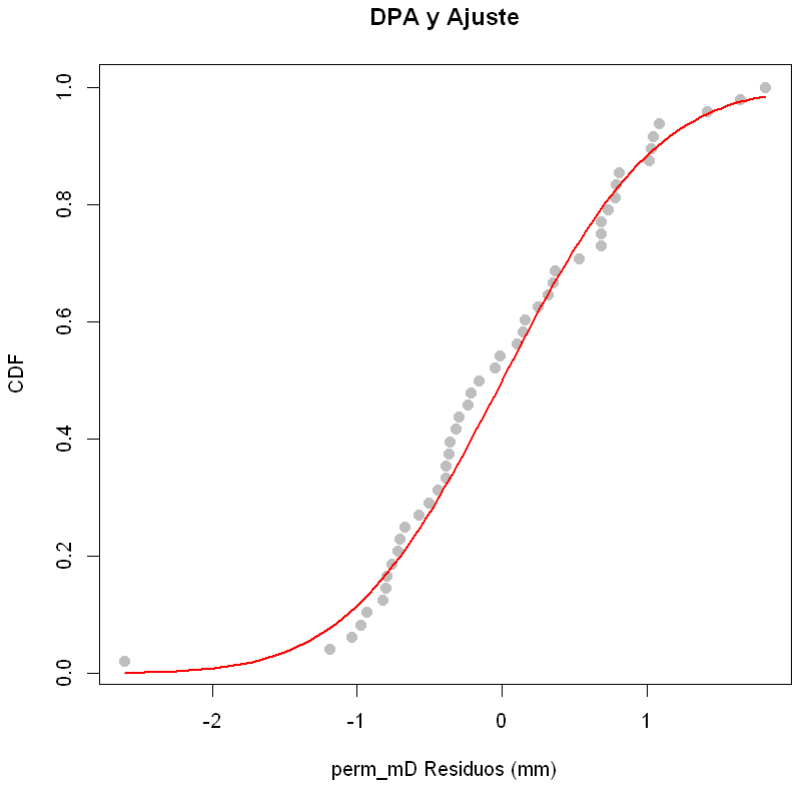
Graficamos las comparativas entre el histrograma y la funcion de distribución normal.

In [50]:
```
PARA_Residual_normal <- list(mean = as.numeric(FD_FP_Residual_normal[1,1]), sd = as.numer
HistModel(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, breaks = "Sturges"
          ylab = "Densidad", colCurve =  "red", col = "darkgray")
```



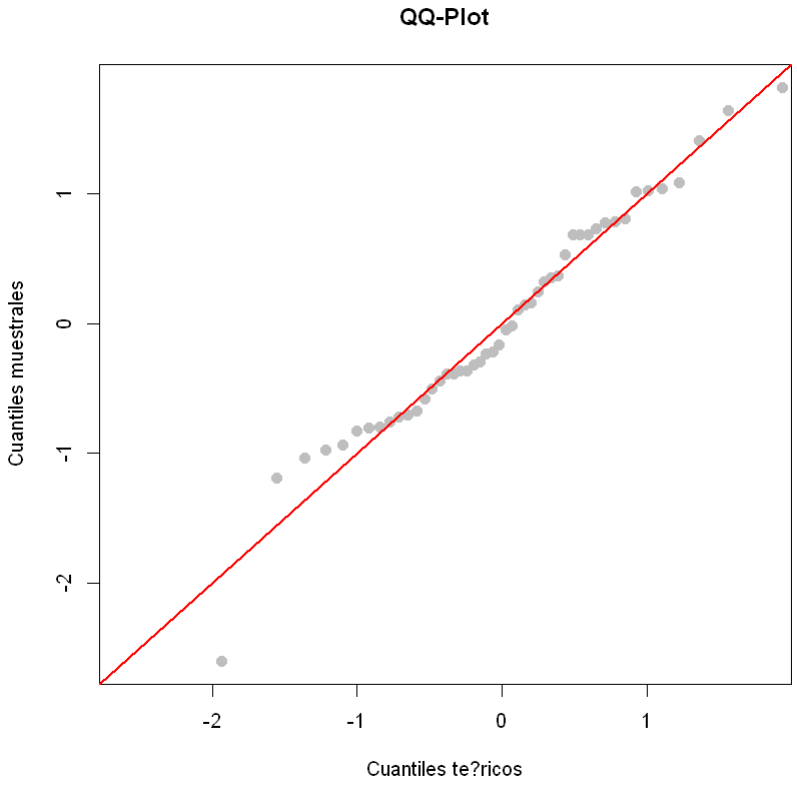
Graficamos las comparativas entre la función de distribución acumulativa empírica y la función de distribución acumulativa normal.

In [51]:
```
CDF(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main = "DP
    lcol = "red", lwd = 2)
```



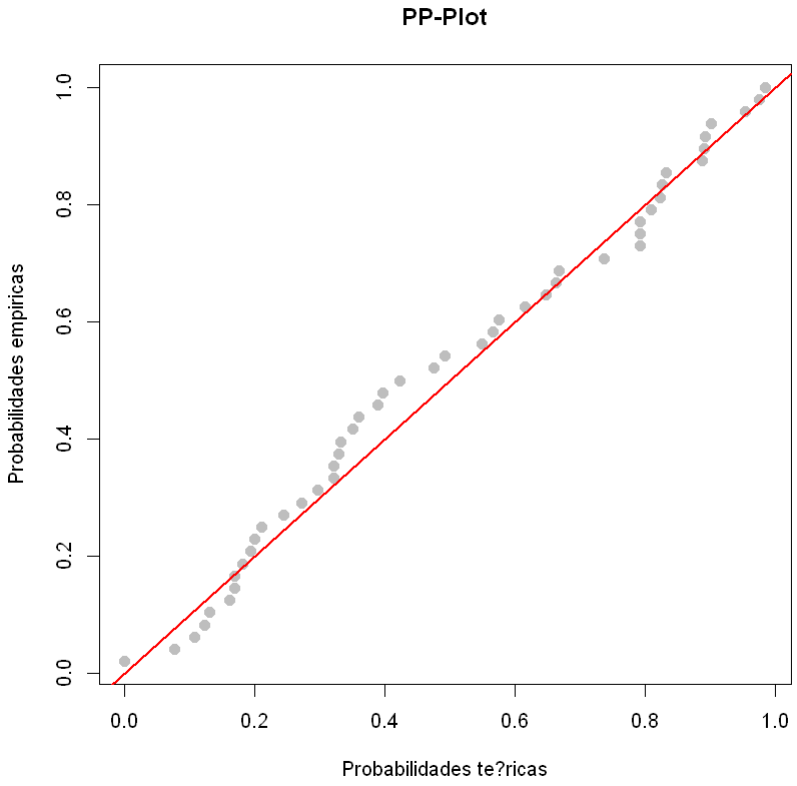
El Grafico cuantil-cuantil (Q-Q plot).

In [52]:
```
QQplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main =
       lcol = "red", lwd = 2)
```



El Grafico Percentil-percentil (P-P plot).

In [53]:
```
PPplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main =
       lcol = "red", lwd = 2)
```

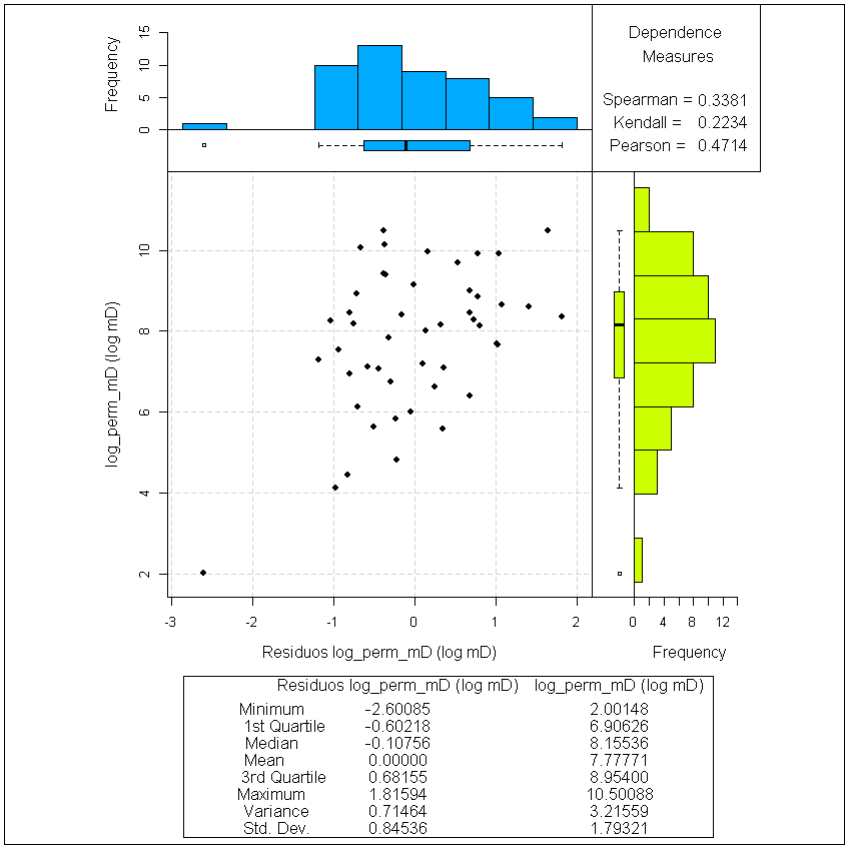


Analisis bivariado: Y vs Y residual.

In [54]:
```
X<-Y_Residual      # Y_Residual es la variable invependiente
Y<-perm_mD_Log     # perm_mD_Log es la variable dependiente
```

Ahora Analizaremos la dependencia entre la variable aleatoria con transformación logarítmica y sus residuos. Se hace el grafico de dispersión.

In [55]:
```
ScatterPlot(X, Y, 9,
            Xmin = Y_Residual_Stat[2,2], Xmax = Y_Residual_Stat[7,2],
            Ymin = perm_mD_Log_Stat[2,2],Ymax = perm_mD_Log_Stat[7,2], XLAB = "Residuos lo
```



	Residuos log_perm_mD (log mD)	log_perm_mD (log mD)
Minimum	-2.60085	2.00148
1st Quartile	-0.60218	6.90626
Median	-0.10756	8.15536
Mean	0.00000	7.77771
3rd Quartile	0.68155	8.95400
Maximum	1.81594	10.50088
Variance	0.71464	3.21559
Std. Dev.	0.84536	1.79321

El resultado de las medidas de dependencia muestra que la dependencia es considerable. Pearson tiene un valor de 0.4714, Spearman es de 0.3381 y Kendall es de 0.4714. Con esta última prueba podemos concluir que la regresión lineal usando las variables aleatorias transformadas no cumplió con todas las condiciones.

Con el caso de las muestras obtenidas de los pluviómetros, podemos ver que los valores atípicos se localizan en los mismos lugares de los valores atípicos obtenidos del radar meteorológico, por lo que debemos tomar en cuenta esta información para el análisis variografico.

Análisis variografico.

Ahora que sabemos cuál es el comportamiento de las variables aleatorias y escogimos la mejor opción continuamos con el análisis vario gráfico.

Análisis variográfico variable perm_mD

Al igual que el análisis exploratorio, necesitamos saber cómo están distribuidos los valores de la variable, para esto usamos la función "DEspacial". Esta función necesita saber las coordenadas (XCoord,YCoord), la variable

que usaremos que en este caso es la variable (perm_mD); cuando ejecutamos esta función obtenemos la siguiente imagen:

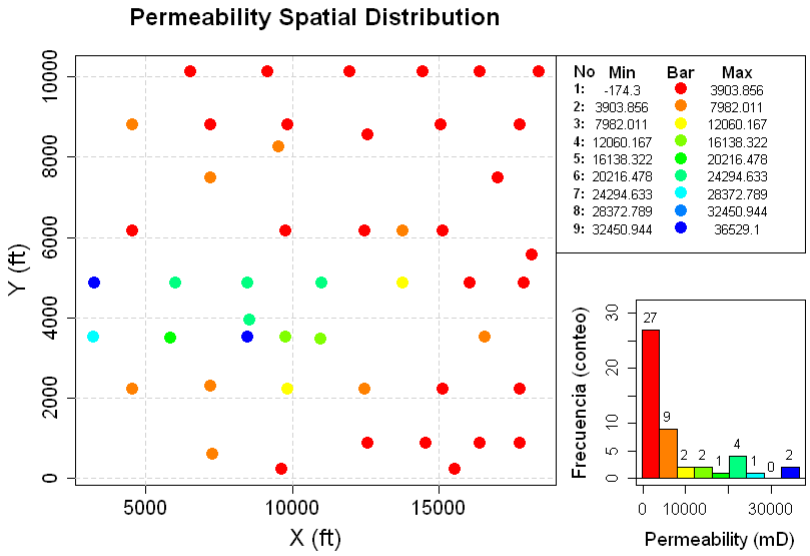
In [56]:

```
root_dir<-getwd()

# Creates a folder to store results for a case
# dir.create(paste(getwd(),"/Results/Burb", sep=""))
#
# result_dir<-paste(root_dir,"/Results/Burb",sep="")

# Creates a folder to store results for Variography Analysis (AV)
dir.create(paste(getwd(),"/Results/Burb/AV", sep=""))

av_dir<-paste(root_dir,"/Results/Burb/AV",sep="")
```

```
Warning message in dir.create(paste(getwd(), "/Results/Burb/AV", sep = "")):
"'C:\Users\danie\Dropbox\semestre 2022-1\clase geoestadistica\ejemplo GAERM\Results\Burb\AV' already exists"
```

In [57]:

```
DEspacial(XCoord, YCoord, perm_mD,
          'X (ft)', 'Y (ft)', 'Permeability (mD)', 'Permeability Spatial Distribution')
```



El grafico que obtenemos nos da la distribución espacial de las muestras con puntos de distintos colores, siendo los colores rojo y azul los valores extremos. A la derecha del grafico tenemos el histograma que por default tiene nueve intervalos (bins), si deseamos cambiar el número de intervalos debemos abrir el archivo que contiene la función DEspacial, el cual es DEspacial.R localizado en la carpeta "Functions", después deben ir a la línea 16 y cambiar el valor nbins como se muestra en la siguiente imagen:

```
DEspacial <-
function (CoorX, CoorY, P1, XLAB="X", YLAB="Y", NameP1= deparse(substitute(P1)), MainTitle = "DistribuciÃ³n espacial",
          cex.lab = 1.8, cex.axis = 1.5, cex.main= 1.8,
          AbsFreq = TRUE,Id = FALSE, IdCol = "black",
          # col = gray.colors(64, start = 0.3, end = 0.9, gamma = 2.2),
          Grid=NULL, breaks = NULL, TextPar=list(col="black", cex=1),
          win = NULL)
{
  library(reshape) # for the function "sort_df"
  library(fields) # for the function "as.image"

  Datos <- as.data.frame(cbind(CoorX, CoorY, P1))
  DatosOrd <- sort_df(Datos, vars = "P1") # "sort_df" sorts "Datos" in ascending order for "P1"

  if (is.null(breaks)) {
    nbins = 9
    Xmax <- max(P1)
    Xmin <- min(P1)
```

Análisis de tendencia de la variable perm_mD.

Para determinar si la variable es estacionaria usamos dos fuentes de información:

- Análisis de regresión de la mediana en la dirección X y en la dirección Y
- Estimar el variograma.

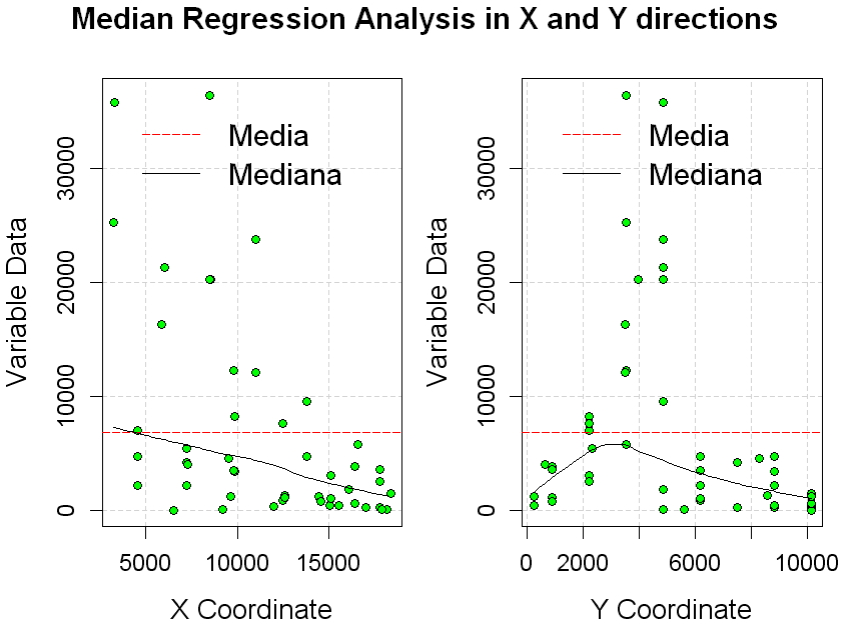
El gráfico de regresión de la mediana nos informa si existe algún indicio de tendencia siguiendo el siguiente criterio:

- Si la regresión es paralela a la línea de la media entonces la variable no tiene tendencia
- Si la regresion no es paralela entonces la variable podría tener algún grado de tendencia.

Cabe aclarar que esta prueba solo nos da indicios de la posible tendencia, mas no es determinante. El variograma es la prueba fuerte que nos indica si la variable tiene tendencia.

Para el análisis de regresión de la mediana usamos la función "GDirecciones", la cual necesita las coordenadas (XCoord, YCoord) y la variable a analizar, en este caso perm_mD.

In [58]:
```
GDirecciones(XCoord, YCoord, perm_mD)
```



Analizando el resultado de la regresión podemos notar que en el eje de coordenadas X tiene una linea descendente, lo cual podemos considerar como un indicio de tendencia. Con el caso del eje de coordenadas Y vemos que su regresión no cruza con la linea del valor esperado y puede ser indicio de tendencia. Esto lo confirmaremos cuando se estime el variograma experimental.

Para estimar el variograma experimental necesitamos:

- número de intervalos (lags)
- distancia mínima (DistMin)
- distancia máxima (DistMax)
- valor de intervalo (lag value)

Definimos el número de intervalos que deseamos usar. Si la distribución espacial de la variable esta muestreada en una malla regular, entonces usamos la información de dicha malla para definir el valor y numero de intervalo; de lo contrario hay que calcular la distancia máxima y mínima de las muestras. Para este caso el número que seleccionamos es 10.

In [59]:
```
X_rng<-XCoord_Stat[8,2]
Y_rng<-YCoord_Stat[8,2]
N_lags<-10
lag_value <- sqrt(X_rng*X_rng+Y_rng*Y_rng)/(2*N_lags)
DistMin<-min(dist(Data_File_Burb[,1:2])) # Minimum distance in data
DistMax<-max(dist(Data_File_Burb[,1:2])) # Minimum distance in data
lag_value<-max((DistMax/2)/N_lags, DistMin)
```

Ahora hay que comparar el valor de la distancia mínima (DistMin) con el valor de intervalo (lag_value).

- Si el valor de intervalo calculado con (DistMax/2)/N_lags es menor a la distancia mínima entonces no usamos el valor de intervalo debido a que no hay pares que cumplan con esa distancia, en su lugar usamos el valor de la distancia mínima.

- Si el valor del intervalo calculado con (DistMax/2)/N_lags es mayor a la distancia mínima, entonces podemos decidir cualquiera de los dos valores

Ya que tenemos estos valores podemos estimar el variograma experimental adireccional, el cual tiene por dirección 0º y ángulo de tolerancia de 90º.
Para estimar el variograma adireccional se usa la función "Variograma", Esta función necesita:

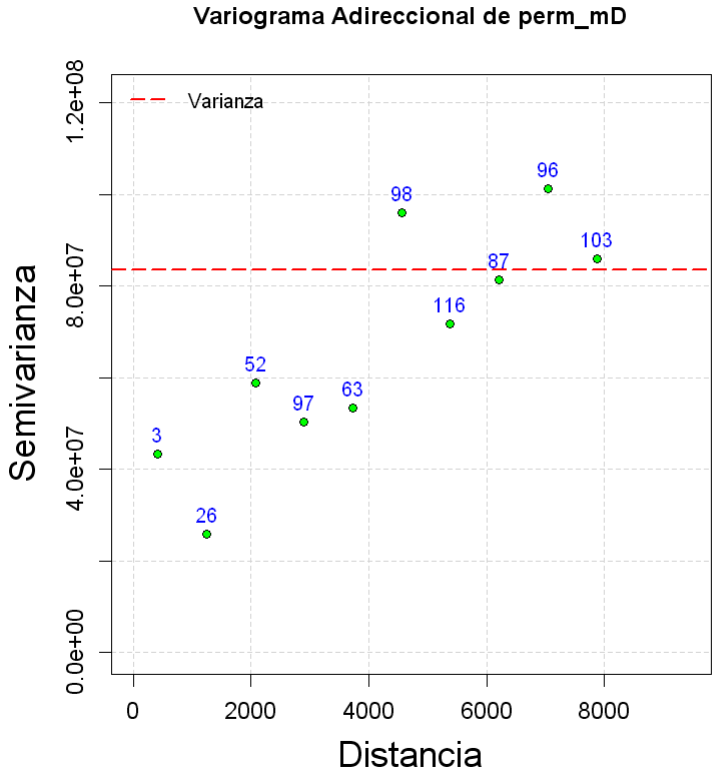
- Coordenadas (XCoord, YCoord)
- Variable (perm_mD)
- Una dirección (Direccion) y su tolerancia angular (Tol), como en este caso es variograma adireccional la dirección es de 0º y su tolerancia es de 90º.
- Número de intervalos (N_lags)
- Valor del intervalo (lag_value)
- Número de pares mínimo, por default es 1
- Titulo del gráfico.

La función "Variograma" usa el estimador clasico o método de momentos, el cual es:

$$\gamma^*(\underline{h}) = \frac{1}{2N(\underline{h})} \sum_{i=1}^{N(\underline{h})} [Z(\underline{x}_i + \underline{h}) - Z(\underline{x}_i)]^2$$

In [75]:
```
perm_mD_VarioEstimation<-Variograma(XCoord, YCoord,
                                    perm_mD, 0, 90, N_lags, lag_value, 1, "Variograma
```

```
variog: computing omnidirectional variogram
```



In [61]:
```
perm_mD_VarioEstimation
write.csv(perm_mD_VarioEstimation, file = paste(av_dir,"/perm_mD_Vario_Adireccional.csv",
```

A data.frame: 10 × 3

Npares	Lags	Semivarianzas
<dbl>	<dbl>	<dbl>
3	414.3761	43357869
26	1243.1283	25802752
52	2071.8806	58723392
97	2900.6328	50352988
63	3729.3850	53330721
98	4558.1372	95975552
116	5386.8895	71704539
87	6215.6417	81426837
96	7044.3939	101149039
103	7873.1461	85880480

Observando el resultado del variograma experimental adireccional podemos notar que no hay evidencias de tendencia, ya que el variograma crece hasta que se acota en la varianza, lo cual indica que esta variable al menos cumple con la hipótesis intrínseca.

Si por alguna razón la variable presenta evidencias de tendencia, entonces hay que aplicar una transformación polinomial.

La transformación polinomial de primer orden tiene la siguiente forma:

$$Z_1(x) = m_1(x) + R_1(x)$$

La transformación polinomial de segundo orden es:

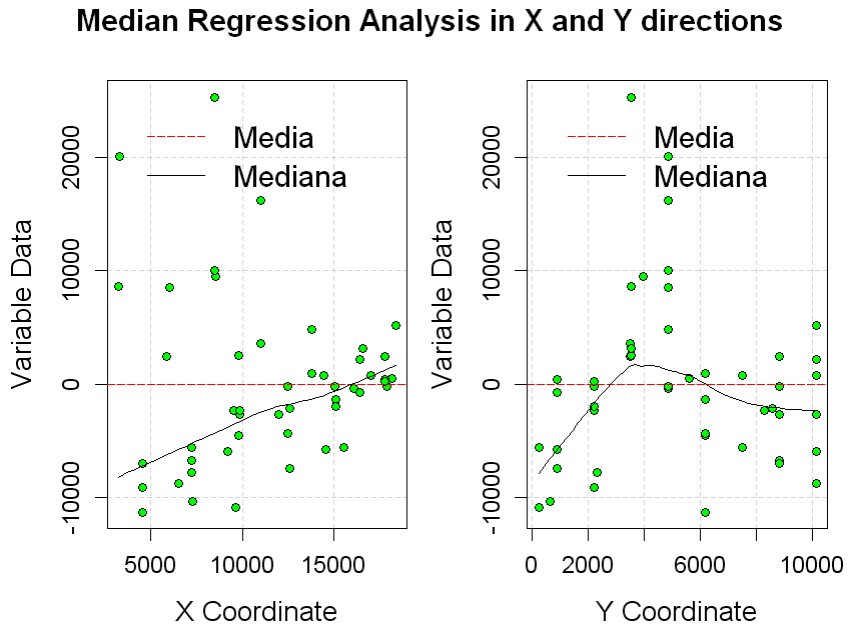
$$Z_2(x) = m_2(x) + R_2(x)$$

La transformación la hacemos usando la función "Trend", la cual necesita las coordenadas (XCoord, YCoord), la variable (perm_mD) y el grado del polinomio (pol_degree).

In [62]:
```
pol_degree=1
perm_mD_Detrended_1<-Trend(XCoord, YCoord,
                           perm_mD, pol_degree)
```

Con esta transformación volvemos a graficar la regresión de la mediana y el variograma experimental adireccional.

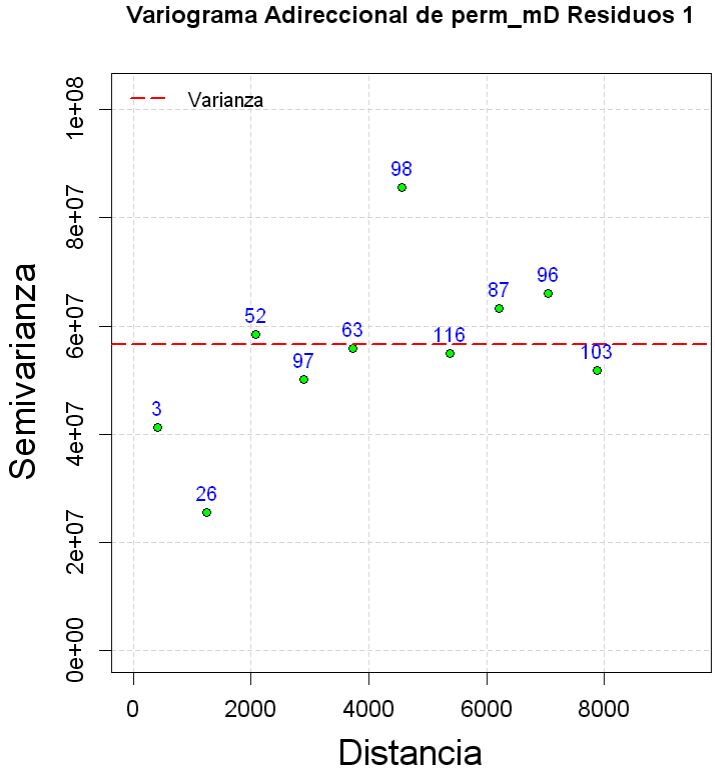
In [63]:
```
GDirecciones(perm_mD_Detrended_1[,1], perm_mD_Detrended_1[,2], perm_mD_Detrended_1[,3])
```



In [64]:

```
perm_mD_Detrended_1_VarioEstimation<-Variograma(perm_mD_Detrended_1[,1], perm_mD_Detrende
                                                perm_mD_Detrended_1[,3], 0, 90, N_la
                                                "Variograma Adireccional de perm_mD
```

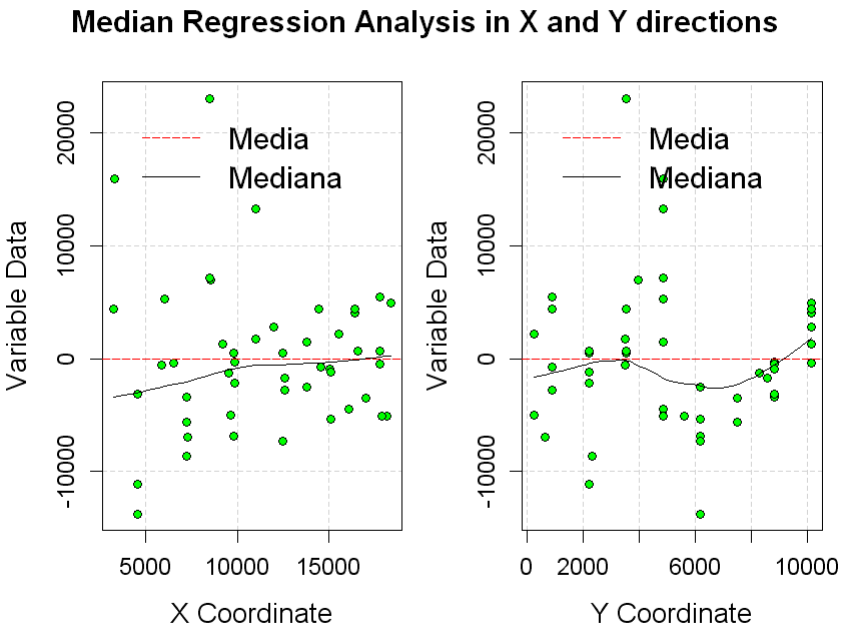
```
variog: computing omnidirectional variogram
```



La transformación de segundo orden se hace de la siguiente forma.

In [65]:
```
pol_degree=2
perm_mD_Detrended_2<-Trend(XCoord, YCoord,
                                  perm_mD, pol_degree)
```

In [66]:
```
GDirecciones(perm_mD_Detrended_2[,1], perm_mD_Detrended_2[,2], perm_mD_Detrended_2[,3])
```



In [67]:
```
perm_mD_Detrended_2_VarioEstimation<-Variograma(perm_mD_Detrended_2[,1], perm_mD_Detrende
                                                perm_mD_Detrended_2[,3], 0, 90, N_la
                                                "Variograma Adireccional de perm_mD
```

```
variog: computing omnidirectional variogram
```

Variograma Adireccional de perm_mD Residuos 2

Modelado variográfico unidimensional de la variable perm_mD

Después del análisis de tendencia, podemos empezar con el modelado del variograma. Por lo que empezaremos con el cálculo de los variogramas direccionales, esto para determinar si hay anisotropía.

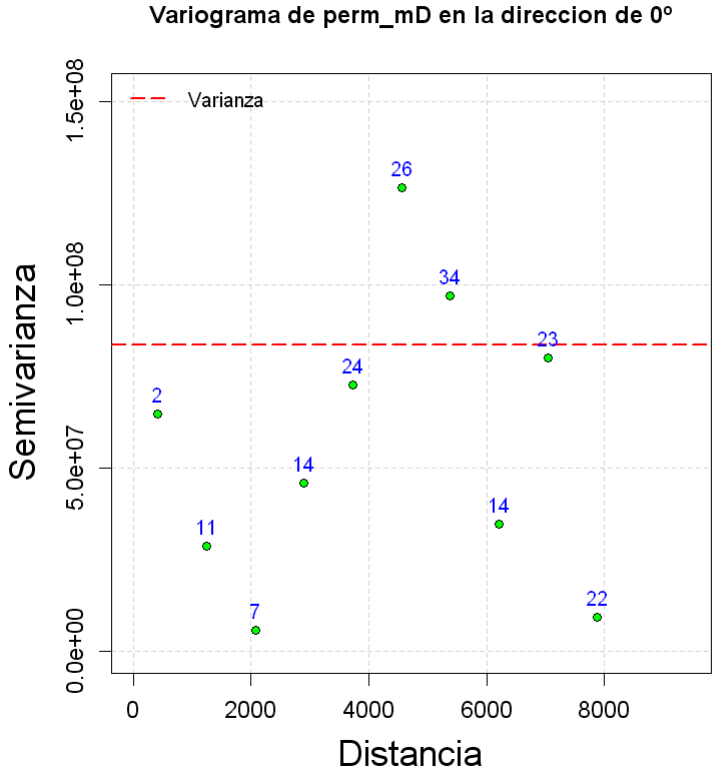
Para calcular los variogramas direccionales solo cambiamos los parámetros en la función variograma:

- Coordenadas (XCoord, YCoord)
- Variable (Pluv_mm_Log)
- Dirección del vector, los cuales son: 0º, 45º, 90º y 135º
- Valor de la tolerancia angular, la cual es de 22.5º
- Número de intervalos (N_lags)
- Valor del intervalo (lag_value)
- Número de pares mínimo, por default es 1
- Titulo del gráfico.

Dado estos parámetros, podemos poner como ejemplo la siguiente grafica que ilustra la forma de cálculo del variograma con los parámetros antes mencionados, para la dirección de vector 0º es:

In [68]:
```
perm_mD_VarioEstimation_dir0<-Variograma(XCoord, YCoord,
                                         perm_mD, 0, 22.5, N_lags, lag_value, 1, "Variogr
```

```
variog: computing variogram for direction = 0 degrees (0 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
```



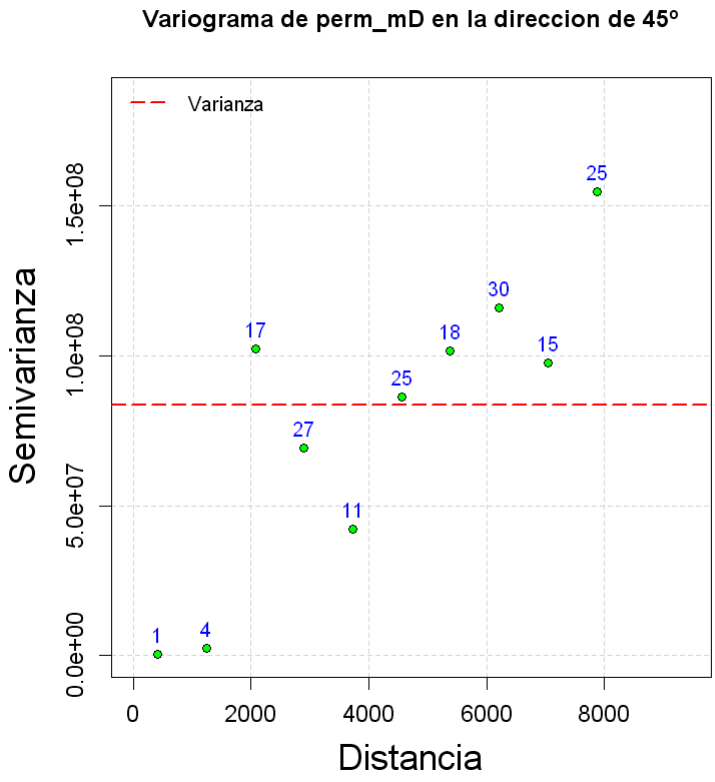
In [69]:
```
write.csv(perm_mD_VarioEstimation_dir0, file = paste(av_dir,"/perm_mD_Vario_direccio0.csv
```

Para el caso del variograma direccional 0º podemos notar que el variograma experimental en la dirección 0º no está bien estimado, solo un intervalo tiene más de 30 pares. Por lo tanto, no se puede juzgar si esta dirección presenta anisotropía.

Calculamos el variograma direccional 45º

In [70]:
```
perm_mD_VarioEstimation_dir45<-Variograma(XCoord, YCoord,
                                          perm_mD, 45, 22.5, N_lags, lag_value, 1, "Variog
```

```
variog: computing variogram for direction = 45 degrees (0.785 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
```



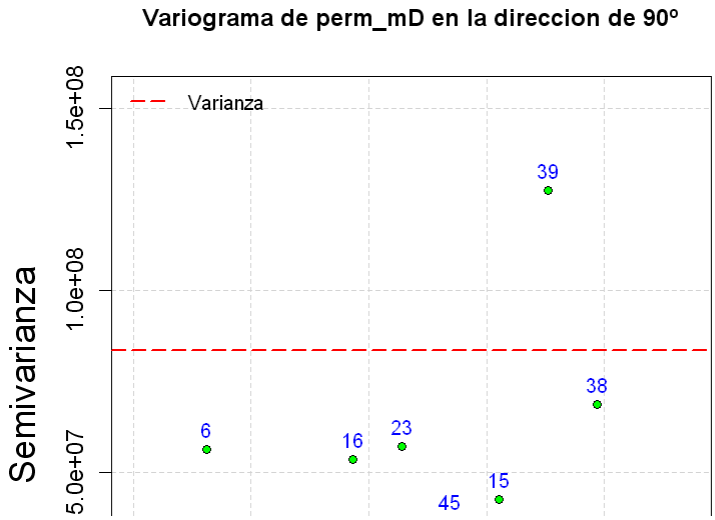
In [71]:
```
write.csv(perm_mD_VarioEstimation_dir45, file = paste(av_dir,"/perm_mD_Vario_direccio45.cs
```

Con el variograma direccional a 45º notamos el mismo problema detectado en el variograma direccional de 0º, solo un intervalo tiene un número de pares superior a 30, por lo tanto, en esta dirección no se puede juzgar si existe algún tipo de anisotropía.

Probamos con el variograma direccional de 90º

In [72]:
```
perm_mD_VarioEstimation_dir90<-Variograma(XCoord, YCoord,
                                          perm_mD, 90, 22.5, N_lags, lag_value, 1, "Variog
```

```
variog: computing variogram for direction = 90 degrees (1.571 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
```



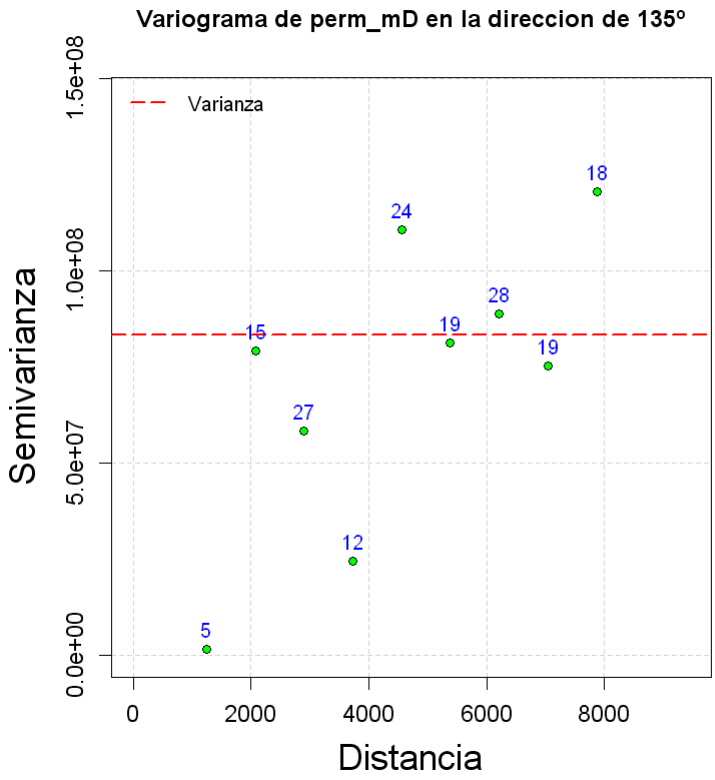
In [73]:
```
write.csv(perm_mD_VarioEstimation_dir90, file = paste(av_dir,"/perm_mD_Vario_direccio90.cs
```

Con el variograma direccional de 90º seguimos obteniendo intervalos mal estimados, solo dos tiene más de 30 pares y por lo tanto no se puede juzgar si existe anisotropía en esta dirección.

Seguimos con el variograma direccional de 135º.

In [74]:
```
perm_mD_VarioEstimation_dir135<-Variograma(XCoord, YCoord,
                                           perm_mD, 135, 22.5, N_lags, lag_value, 1, "Vario
```

```
variog: computing variogram for direction = 135 degrees (2.356 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
```



In [80]:
```
write.csv(perm_mD_VarioEstimation_dir135, file = paste(av_dir,"/perm_mD_Vario_direccio135
```

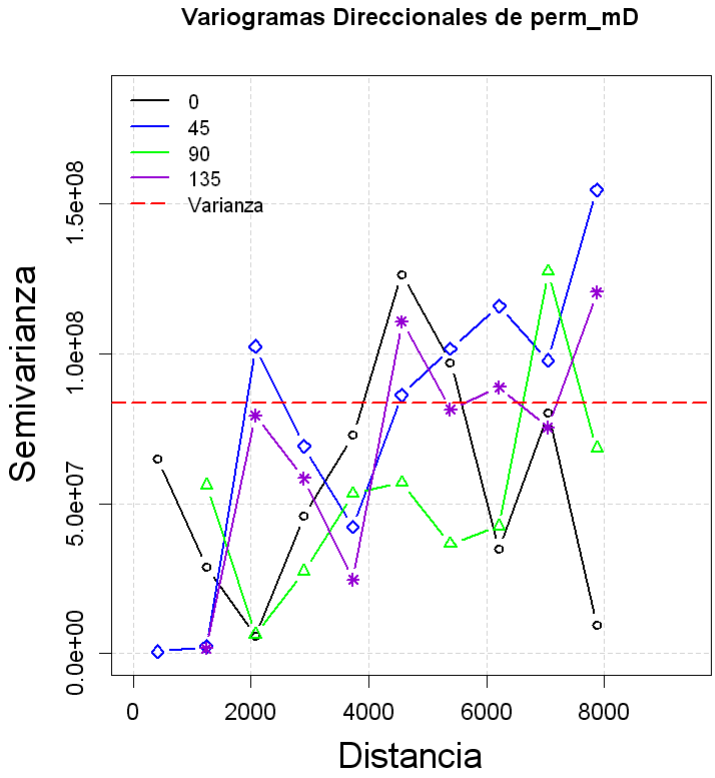
El variograma direccional a 135º no tiene intervalos con un número de pares superior a 30, de todos los casos

direccionales este es el peor y al igual que los demás, no se puede juzgar si existe anisotropía.

Para visualizar los cuatro variogramas direccionales en una sola imagen usamos la función "Variograma4D". Esta función necesita de las coordenadas (XCoord, YCoord), la variable (perm_mD) las direcciones del vector en grados (0, 45, 90, 135), el ángulo de tolerancia (22.5), el número de intervalos (N_lags), el valor del intervalo (lag_value), el número mínimo de pares, el cual por default es uno.

In [70]:
```
perm_mD_VarioEstimation4D<-Variograma4D(XCoord, YCoord,
                                        perm_mD, 0, 45, 90, 135, 22.5, N_lags, lag_value
```

```
variog: computing variogram for direction = 0 degrees (0 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 45 degrees (0.785 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 90 degrees (1.571 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 135 degrees (2.356 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
```



Si colocan en la consola "perm_mD_VarioEstimation4D", pueden obtener los pares de la estimación en los cuatro variogramas direccionales.

In [71]:
```
perm_mD_VarioEstimation4D
write.csv(perm_mD_VarioEstimation4D, file = paste(av_dir,"/perm_mD_VarioEstimation4D.csv"
```

$Zero

A matrix: 10 × 3 of type dbl

2	414.3761	64750364
11	1243.1283	28755245
7	2071.8806	5724445
14	2900.6328	45843993
24	3729.3850	72759016
26	4558.1372	126347699

34 5386.8895 97059734

14 6215.6417 34739560

23 7044.3939 80081235

22 7873.1461 9397970

$FortyFive

A matrix: 10 × 3 of type dbl

1 414.3761 572878.1

4 1243.1283 2320975.4

17 2071.8806 102360387.4

27 2900.6328 69134525.4

11 3729.3850 42216734.2

25 4558.1372 86186198.6

18 5386.8895 101636450.7

30 6215.6417 115825532.8

15 7044.3939 97763871.6

25 7873.1461 154618969.6

$Ninety

A matrix: 9 × 3 of type dbl

6 1243.128 56094073

13 2071.881 6430924

29 2900.633 27468938

16 3729.385 53356753

23 4558.137 56949656

45 5386.889 36504983

15 6215.642 42374285

39 7044.394 127421340

38 7873.146 68497417

$OneThertyFive

A matrix: 9 × 3 of type dbl

5 1243.128 1743104

15 2071.881 79321111

27 2900.633 58488613

12 3729.385 24627240

24 4558.137 110669454

19 5386.889 81342908

28 6215.642 88835739

19 7044.394 75397314

18 7873.146 120586555

```
Error in (function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE, : argum
ents imply differing number of rows: 10, 9
Traceback:

1. write.csv(perm_mD_VarioEstimation4D, file = paste(av_dir, "/perm_mD_VarioEstimation4D.c
```

```
sv",
 .     sep = ""))
2. eval.parent(Call)
3. eval(expr, p)
4. eval(expr, p)
5. utils::write.table(perm_mD_VarioEstimation4D, file = paste(av_dir,
 .     "/perm_mD_VarioEstimation4D.csv", sep = ""), col.names = NA,
 .     sep = ",", dec = ".", qmethod = "double")
6. data.frame(x)
7. as.data.frame(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
8. as.data.frame.list(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
9. do.call(data.frame, c(x, alis))
10. (function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE,
  .     fix.empty.names = TRUE, stringsAsFactors = FALSE)
  . {
  .     data.row.names <- if (check.rows && is.null(row.names))
  .         function(current, new, i) {
  .             if (is.character(current))
  .                 new <- as.character(new)
  .             if (is.character(new))
  .                 current <- as.character(current)
  .             if (anyDuplicated(new))
  .                 return(current)
  .             if (is.null(current))
  .                 return(new)
  .             if (all(current == new) || all(current == ""))
  .                 return(new)
  .             stop(gettextf("mismatch of row names in arguments of 'data.frame', item %
d",
  .                 i), domain = NA)
  .         }
  .     else function(current, new, i) {
  .         if (is.null(current)) {
  .             if (anyDuplicated(new)) {
  .                 warning(gettextf("some row.names duplicated: %s --> row.names NOT use
d",
  .                   paste(which(duplicated(new)), collapse = ",")),
  .                   domain = NA)
  .                 current
  .             }
  .             else new
  .         }
  .         else current
  .     }
  .     object <- as.list(substitute(list(...)))[-1L]
  .     mirn <- missing(row.names)
  .     mrn <- is.null(row.names)
  .     x <- list(...)
  .     n <- length(x)
  .     if (n < 1L) {
  .         if (!mrn) {
  .             if (is.object(row.names) || !is.integer(row.names))
  .                 row.names <- as.character(row.names)
  .             if (anyNA(row.names))
  .                 stop("row names contain missing values")
  .             if (anyDuplicated(row.names))
  .                 stop(gettextf("duplicate row.names: %s", paste(unique(row.names[duplic
ated(row.names)]),
  .                   collapse = ", ")), domain = NA)
  .         }
  .         else row.names <- integer()
  .         return(structure(list(), names = character(), row.names = row.names,
  .             class = "data.frame"))
  .     }
```

```
  .     vnames <- names(x)
  .     if (length(vnames) != n)
  .         vnames <- character(n)
  .     no.vn <- !nzchar(vnames)
  .     vlist <- vnames <- as.list(vnames)
  .     nrows <- ncols <- integer(n)
  .     for (i in seq_len(n)) {
  .         xi <- if (is.character(x[[i]]) || is.list(x[[i]]))
  .             as.data.frame(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactor
s)
  .         else as.data.frame(x[[i]], optional = TRUE)
  .         nrows[i] <- .row_names_info(xi)
  .         ncols[i] <- length(xi)
  .         namesi <- names(xi)
  .         if (ncols[i] > 1L) {
  .             if (length(namesi) == 0L)
  .                 namesi <- seq_len(ncols[i])
  .             vnames[[i]] <- if (no.vn[i])
  .                 namesi
  .             else paste(vnames[[i]], namesi, sep = ".")
  .         }
  .         else if (length(namesi)) {
  .             vnames[[i]] <- namesi
  .         }
  .         else if (fix.empty.names && no.vn[[i]]) {
  .             tmpname <- deparse(object[[i]], nlines = 1L)[1L]
  .             if (startsWith(tmpname, "I(") && endsWith(tmpname,
  .                 ")")) {
  .                 ntmpn <- nchar(tmpname, "c")
  .                 tmpname <- substr(tmpname, 3L, ntmpn - 1L)
  .             }
  .             vnames[[i]] <- tmpname
  .         }
  .         if (mirn && nrows[i] > 0L) {
  .             rowsi <- attr(xi, "row.names")
  .             if (any(nzchar(rowsi)))
  .                 row.names <- data.row.names(row.names, rowsi,
  .                   i)
  .         }
  .         nrows[i] <- abs(nrows[i])
  .         vlist[[i]] <- xi
  .     }
  .     nr <- max(nrows)
  .     for (i in seq_len(n)[nrows < nr]) {
  .         xi <- vlist[[i]]
  .         if (nrows[i] > 0L && (nr%%nrows[i] == 0L)) {
  .             xi <- unclass(xi)
  .             fixed <- TRUE
  .             for (j in seq_along(xi)) {
  .                 xi1 <- xi[[j]]
  .                 if (is.vector(xi1) || is.factor(xi1))
  .                   xi[[j]] <- rep(xi1, length.out = nr)
  .                 else if (is.character(xi1) && inherits(xi1, "AsIs"))
  .                   xi[[j]] <- structure(rep(xi1, length.out = nr),
  .                     class = class(xi1))
  .                 else if (inherits(xi1, "Date") || inherits(xi1,
  .                   "POSIXct"))
  .                   xi[[j]] <- rep(xi1, length.out = nr)
  .                 else {
  .                   fixed <- FALSE
  .                   break
  .                 }
  .             }
  .             if (fixed) {
```

```
  .                 vlist[[i]] <- xi
  .                 next
  .             }
  .         }
  .         stop(gettextf("arguments imply differing number of rows: %s",
  .             paste(unique(nrows), collapse = ", ")), domain = NA)
  .     }
  .     value <- unlist(vlist, recursive = FALSE, use.names = FALSE)
  .     vnames <- as.character(unlist(vnames[ncols > 0L]))
  .     if (fix.empty.names && any(noname <- !nzchar(vnames)))
  .         vnames[noname] <- paste0("Var.", seq_along(vnames))[noname]
  .     if (check.names) {
  .         if (fix.empty.names)
  .             vnames <- make.names(vnames, unique = TRUE)
  .         else {
  .             nz <- nzchar(vnames)
  .             vnames[nz] <- make.names(vnames[nz], unique = TRUE)
  .         }
  .     }
  .     names(value) <- vnames
  .     if (!mrn) {
  .         if (length(row.names) == 1L && nr != 1L) {
  .             if (is.character(row.names))
  .                 row.names <- match(row.names, vnames, 0L)
  .             if (length(row.names) != 1L || row.names < 1L ||
  .                 row.names > length(vnames))
  .                 stop("'row.names' should specify one of the variables")
  .             i <- row.names
  .             row.names <- value[[i]]
  .             value <- value[-i]
  .         }
  .         else if (!is.null(row.names) && length(row.names) !=
  .             nr)
  .             stop("row names supplied are of the wrong length")
  .     }
  .     else if (!is.null(row.names) && length(row.names) != nr) {
  .         warning("row names were found from a short variable and have been discarded")
  .         row.names <- NULL
  .     }
  .     class(value) <- "data.frame"
  .     if (is.null(row.names))
  .         attr(value, "row.names") <- .set_row_names(nr)
  .     else {
  .         if (is.object(row.names) || !is.integer(row.names))
  .             row.names <- as.character(row.names)
  .         if (anyNA(row.names))
  .             stop("row names contain missing values")
  .         if (anyDuplicated(row.names))
  .             stop(gettextf("duplicate row.names: %s", paste(unique(row.names[duplicated
(row.names)]),
  .                 collapse = ", ")), domain = NA)
  .         row.names(value) <- row.names
  .     }
  .     value
  . })(Zero = structure(c(2, 11, 7, 14, 24, 26, 34, 14, 23, 22, 414.376112366532,
  . 1243.12833709959, 2071.88056183266, 2900.63278656572, 3729.38501129878,
  . 4558.13723603184, 5386.8894607649, 6215.64168549797, 7044.39391023103,
  . 7873.14613496409, 64750363.7125, 28755244.5977273, 5724445.13893571,
  . 45843992.6980821, 72759015.9851521, 126347699.341577, 97059733.5960294,
  . 34739560.4772929, 80081235.4345413, 9397970.11109545), .Dim = c(10L,
  . 3L)), FortyFive = structure(c(1, 4, 17, 27, 11, 25, 18, 30, 15,
  . 25, 414.376112366532, 1243.12833709959, 2071.88056183266, 2900.63278656572,
  . 3729.38501129878, 4558.13723603184, 5386.8894607649, 6215.64168549797,
  . 7044.39391023103, 7873.14613496409, 572878.08, 2320975.3583625,
```

```
. 102360387.392709, 69134525.3740574, 42216734.2263773, 86186198.571726,
. 101636450.696858, 115825532.774858, 97763871.5896033, 154618969.565888
. ), .Dim = c(10L, 3L)), Ninety = structure(c(6, 13, 29, 16, 23,
. 45, 15, 39, 38, 1243.12833709959, 2071.88056183266, 2900.63278656572,
. 3729.38501129878, 4558.13723603184, 5386.8894607649, 6215.64168549797,
. 7044.39391023103, 7873.14613496409, 56094073.2441667, 6430924.34296154,
. 27468937.9371914, 53356753.3325063, 56949656.4527, 36504982.6211367,
. 42374284.9785167, 127421339.849288, 68497416.9739092), .Dim = c(9L,
. 3L)), OneThertyFive = structure(c(5, 15, 27, 12, 24, 19, 28,
. 19, 18, 1243.12833709959, 2071.88056183266, 2900.63278656572,
. 3729.38501129878, 4558.13723603184, 5386.8894607649, 6215.64168549797,
. 7044.39391023103, 7873.14613496409, 1743103.83369, 79321110.6289367,
. 58488612.6796352, 24627239.9944583, 110669453.884794, 81342908.2416605,
```

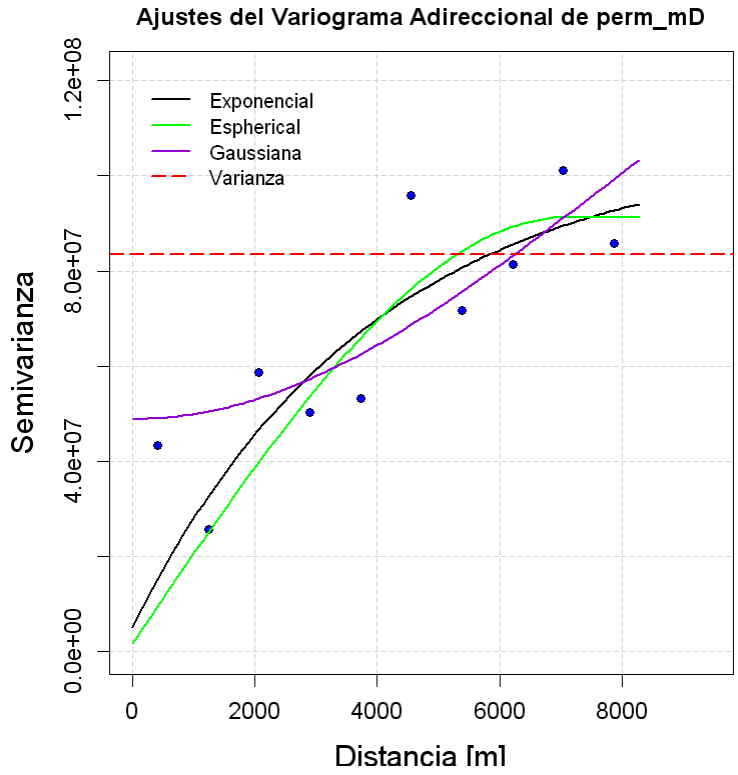
Si comparamos los resultados obtenidos de los variogramas direccionales con el variograma adireccional, podemos notar que la mejor opción es usar el variograma adireccional ya que es el único que está bien estimado. Tambien podemos considerar que la variable es isotrópica.

Cabe aclarar que en el caso de que los variogramas direccionales estén bien estimados, entonces se debe ajustar un modelo de variograma autorizado y así determinar el tipo de anisotropía (geométrica o zonal) que podemos encontrar.

Ahora que sabemos cuál es el mejor variograma experimental, procedemos a ajustar un modelo de variograma autorizado. Para hacer el ajuste automático usamos la función "AllModel", esta función necesita de las coordenadas (XCoord, YCoord), la variable (perm_mD), el dirección del vector el cual es de 0º, su ángulo de tolerancia (90º), número de intervalos (N_lags) y valor de intervalo (lag_value). El resultado de usar la función "AllModel" es un gráfico que nos mostrara tres tipos de modelos validos: exponencial, esférico y Gaussiano:

In [79]:
```
perm_mD_AllModelVarioFit<-AllModel(XCoord, YCoord,
                                    perm_mD, 0, 90, N_lags, lag_value, 1, "Ajustes del Va
```

```
variog: computing omnidirectional variogram
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
```



También obtenemos una tabla con los valores calculados de cada modelo, donde el valor del error (SCE) nos indica cual es el mejor modelo en función del menor error.

In [81]:
```
perm_mD_AllModelVarioFit
write.csv(perm_mD_AllModelVarioFit, file = paste(av_dir,"/perm_mD_AllModelVarioFit.csv",s
```

A matrix: 3 × 4 of type dbl

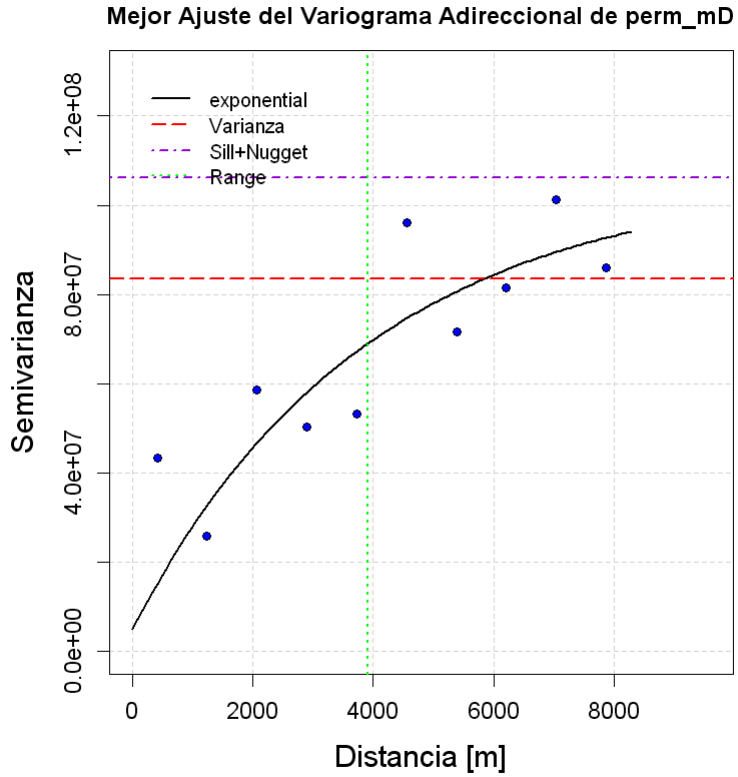
	Nugget	Meseta+Nugget	Alcance	SCE
exponential	5050911	106219929	3911.094	1.969296e+15
spherical	1666401	91260999	7082.350	2.405089e+15
gaussian	48876925	175128356	11031.622	1.360523e+16

Para graficar el mejor modelo de variograma usamos la función "BestModel", esta función es similar a la función "AllModel", solo nos muestra el mejor modelo y los valores de sus parámetros

In [82]:
```
perm_mD_BestModelVarioFit<-BestModel(XCoord, YCoord,
                                     perm_mD, 0, 90, N_lags, lag_value, 1, "Mejor Ajuste
```

```
variog: computing omnidirectional variogram
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
```

```
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
```



In [83]:

```
perm_mD_BestModelVarioFit
```

A matrix: 1 × 6 of type dbl

	Nugget	Meseta+Nugget	Alcance	SCE	MaxY	MinY
exponential	5050911	106219929	3911.094	1.969297e+15	101149039	25802752

Como podemos notar, el mejor modelo según el ajuste automático es Exponencial, sin embargo, podemos probar un ajuste manual.

Para hacer el ajuste manual usamos la función "EyeModel", esta función necesita de las coordenadas (XCoord, YCoord), la variable (perm_mD), la dirección del vector (0º) y su ángulo de tolerancia (90º), el número de intervalos (N_lags), valor de intervalo (lag_value). Ahora de forma manual necesitamos ingresar la información a los siguientes parámetros: modelo de variograma (vario_model) que usaremos, en este caso tenemos las tres opciones numeradas de la siguiente forma:

- 1- Exponencial
- 2- Esférico
- 3- Gaussiano

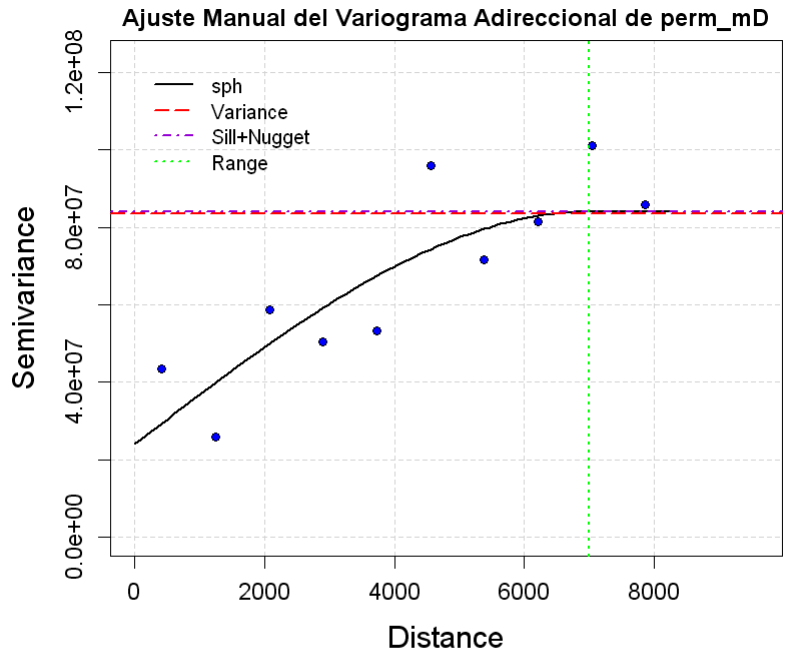
después ingresamos el valor de nugget (nugget), el valor de meseta más nugget (sill_and_nugget) y el alcance (rank). Estos valores se van cambiando bajo el criterio del usuario, el objetivo es lugar un ajuste con el menor error posible, pero con un ajuste adecuado, sin exagerar en el uso del nugget.

In [84]:
```
vario_model<- 2
nugget<- 24000000
sill_and_nugget<- 84000000
rank <- 7000
```

In [85]:
```
perm_mD_vario_model<- 2
perm_mD_nugget<- 24000000
perm_mD_sill_and_nugget<- 84000000
perm_mD_rank <- 7000
```

In [86]:
```
perm_mD_EyeModelVarioFit<-EyeModel(XCoord, YCoord,
                                   perm_mD, 0, 90, N_lags, lag_value, 1,
                                   perm_mD_vario_model, perm_mD_nugget, perm_mD_sill_and_
                                   "Ajuste Manual del Variograma Adireccional de perm_mD
```

```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	24000000.0000	84000000.0000	7000.0000	107863446013301

Para comprobar si el ajuste propuesto es válido realizamos la validación cruzada. Si los valores estimados (Z^*) son cercanos a los valores observados (Z) entonces la diferencia entre los valores observados y los valores estimados deben cumplir los siguientes criterios:

el valor esperado

$$\frac{1}{n}\sum_{i=1}^{n}\{Z(\underline{x}_i) - Z^*(\underline{x}_i)\} \qquad \text{cercano a } 0$$

La varianza.

$$\frac{1}{n}\sum_{i=1}^{n}\{Z(\underline{x}_i)-Z^*(\underline{x}_i)\}^2 \qquad \text{pequeño}$$

Para realizar la validación cruzada usamos la función "CrossValidation", la cual necesita los vectores de posicionamiento (XCoord, YCoord), la variable aleatoria (perm_mD), el modelo de variograma (vario_model), valor de nugget (nugget), el valor de meseta más nugget (sill_and_nugget), el alcance (rank) y los valores de anisotropia geometrica: el valor de máxima anisotropía (MaxAnis) el cual corresponde al valor del ángulo del vector del semivariograma y la relación de anisotropía (proporción), esta proporción se calcula en función de

$$\lambda = \frac{A}{B}$$

El variograma con el eje mayor (mayor alcance) el cual nombramos como A y el variograma con el eje menor (menor alcance) el cual nombramos B. Dado que este ejemplo isotrópico, el valor de máxima anisotropía es cero y la relación de anisotropía es uno.

In [88]:
```
perm_mD_CrossValid<- CrossValidation(XCoord, YCoord,
                                     perm_mD, vario_model, nugget, sill_and_nugget, rank
```

Lo que obtenemos con la validación cruzada es una tabla. Las filas 1 y 2 tienen la información de las coordenadas, la fila 3 tiene los valores de la variable (Z) , la fila 4 muestra los valores estimados con el método de validación cruzada conocido como leave one out, estimando el valor con el método de kriging usando el variograma propuesto ($Z*$), la fila 5 es la diferencia entre la variable y los valores estimados ($Z–Z*$)

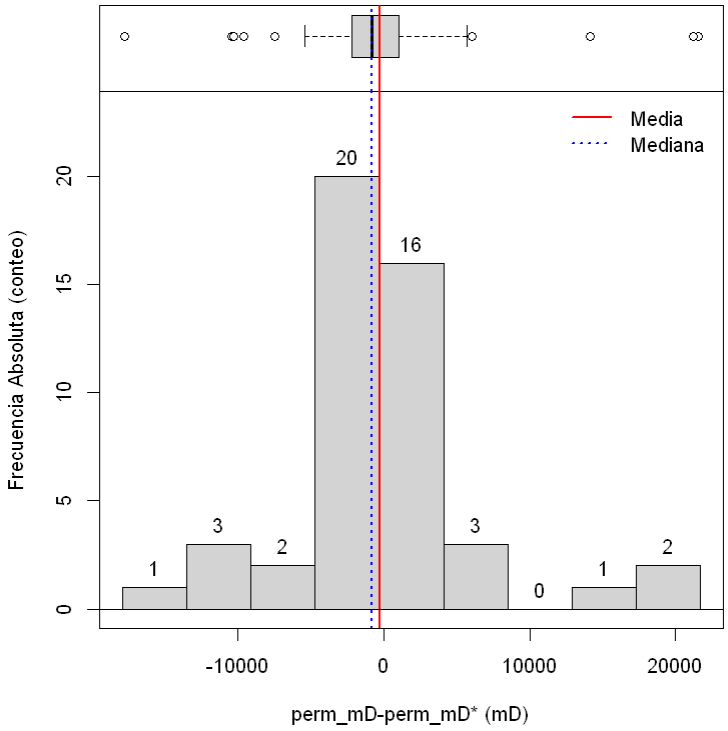
ya que tenemos la validación cruzada calculamos los estadígrafos.

In [89]:
```
perm_mD_CrossValid_Sta <- Val_Estadisticos(perm_mD_CrossValid[1:102,c(3,4,5)])
write.csv(perm_mD_CrossValid_Sta, file = paste(av_dir,"/perm_mD_CrossValid_Sta.csv",sep="
```

Al ver los estadígrafos notamos que la diferencia del valor esperado $Z-Z^*$ es cercana a cero mientras que la varianza $Z-Z^*$ no es tan pequeña.

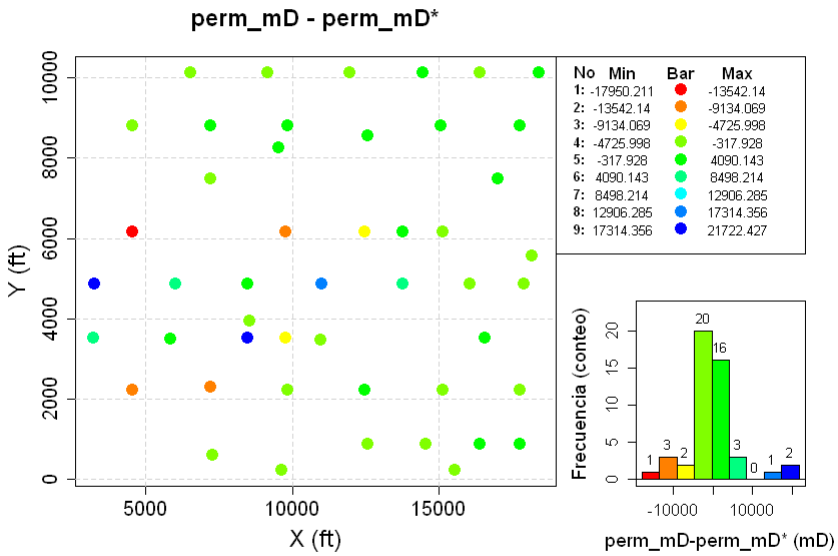
Ahora graficamos el histograma con los errores.

In [90]:
```
HistBoxplot(x=perm_mD_CrossValid[,5], mean = perm_mD_CrossValid_Sta[5,3], median = perm_ml
            xlab = "perm_mD-perm_mD* (mD)", ylab = "Frecuencia Absoluta (conteo)", AbsFre
```



Si analizamos el histograma obtenido con la diferencia entre los valores estimados (Z^*) y los valores observados (Z) podemos cuatro valores atípicos, tres a la derecha del boxplot y uno a la izquierda. Para saber cuál es la ubicación de esos valores atípicos graficamos su distribución espacial con la función "DEspacial".

In [91]:
```
DEspacial(perm_mD_CrossValid[,1], perm_mD_CrossValid[,2], perm_mD_CrossValid[,5],
          'X (ft)', 'Y (ft)', 'perm_mD-perm_mD* (mD)', 'perm_mD - perm_mD*')
```



De este grafico podemos notar que el valor atípico negativo marcado en color rojo está en la frontera, por lo que es difícil saber si es un valor atípico. Mientras que los valores atípicos marcados en azul fuerte probablemente

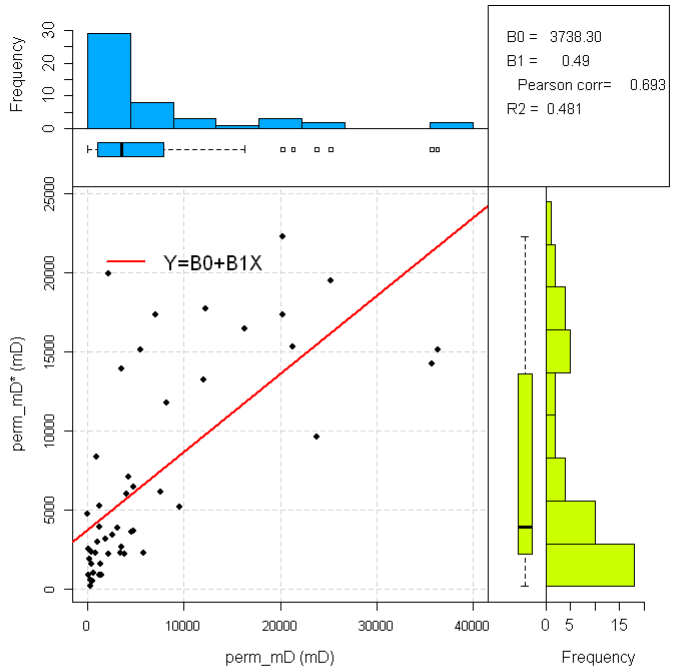
si sean valores atípicos espaciales, en especial el ubicado a la izquierda, ya que los valores vecinos tienen valores más pequeños en comparación.

Los siguiente es saber si el modelo propuesto refleja adecuadamente la relación espacial de los datos, esto lo podemos saber usando un gráfico de dispersión. Si los datos estimados Z^* son cercanos a los valores reales $\mathbf{Z}$ entonces podríamos esperar que la dependencia sea alta.

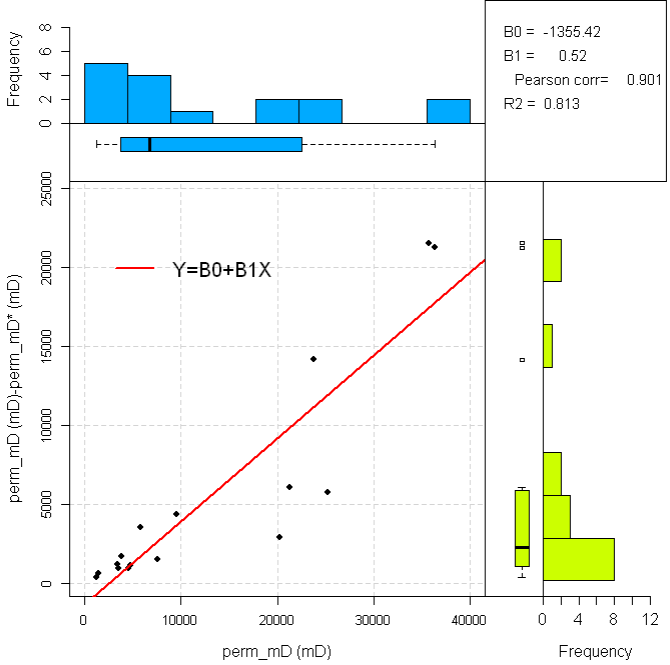
In [92]:
```
pos1 <- which(perm_mD_CrossValid[,5] > 1.3)
pos0 <- which(perm_mD_CrossValid[,5] < -2)
```

In [93]:
```
# perm_mD is the independent variable
X<-perm_mD_CrossValid[,3]
# perm_mD* is the dependent variable
Y<-perm_mD_CrossValid[,4]
```

In [94]:
```
scaterplotReg(perm_mD_CrossValid[,3] , perm_mD_CrossValid[,4], 9,
              Xmin = perm_mD_CrossValid_Sta[2,1], Xmax = perm_mD_CrossValid_Sta[7,1],
              Ymin = perm_mD_CrossValid_Sta[2,2],Ymax = perm_mD_CrossValid_Sta[7,2],
              XLAB = "perm_mD (mD)", YLAB = "perm_mD* (mD)")
```



In [95]:
```
scaterplotReg(perm_mD_CrossValid[,3] , perm_mD_CrossValid[,5], 9,
              Xmin = perm_mD_CrossValid_Sta[2,1], Xmax = perm_mD_CrossValid_Sta[7,1],
              Ymin = perm_mD_CrossValid_Sta[2,2],Ymax = perm_mD_CrossValid_Sta[7,2],
              XLAB = "perm_mD (mD)", YLAB = "perm_mD (mD)-perm_mD* (mD)")
```



Observando los resultados del grafico de dispersión notamos que la dependencia no es lo suficientemente alta para ofrecer una buena representación de la variable aleatoria a partir del variograma propuesto.

In [ ]: