

# Ejemplo GAERM

August 21, 2022

## 1 Ejemplo práctico: estimación de la porosidad y la permeabilidad.

Autores: Daniel Vázquez-Ramírez<sup>(1)</sup> , Martín Díaz-Viera<sup>(2)</sup>, Van Huong Le<sup>(1)</sup>

- 1) Posgrado en Ciencias de la Tierra (UNAM)
- 2) Instituto Mexicano del Petróleo

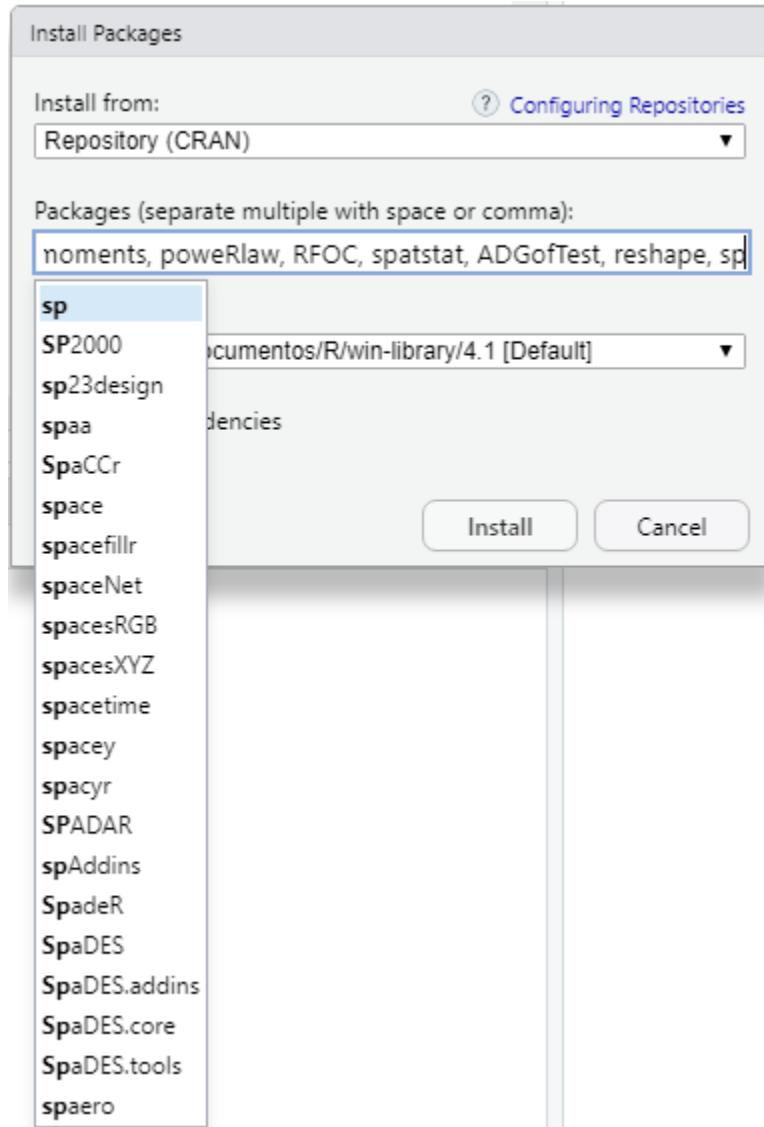
## 2 Introducción.

El siguiente ejemplo muestra la forma de estimar la porosidad y la permeabilidad obtenida de diferentes núcleos, al igual del caso de la estimación de la precipitación en el valle de México, se usará kriging y cokriging ordinario. El proyecto precargado lo pueden descargar en la página del curso <http://www.esmg-mx.org/activities/courses/geoestadistica> en el enlace " Apuntes de la clase práctica porosidad"

### 2.1 Carga de bibliotecas 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.

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 (Para mayores detalles sobre como instalar Las bibliotecas use el siguiente link [https://www.esmg-mx.org/media/courses/geoestadistica/practicas/R\\_and\\_Rstudio\\_Instalation.pdf](https://www.esmg-mx.org/media/courses/geoestadistica/practicas/R_and_Rstudio_Instalation.pdf)).



La segunda opción es usando scripts, para eso abrimos el script “Getting\_Started\_script.R” y ejecutar las siguientes líneas.

```
[ ]: 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("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")
install.packages("splancs")
install.packages("RandomFieldsUtils")
install.packages("devtools")

#set back to root work directory

```

Despues de instalar las bibliotecas debemos cargarlos de la siguiente forma:

```
[3]: 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)
```

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

```
[4]: root_dir<-getwd()

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

```

setwd(function_dir)

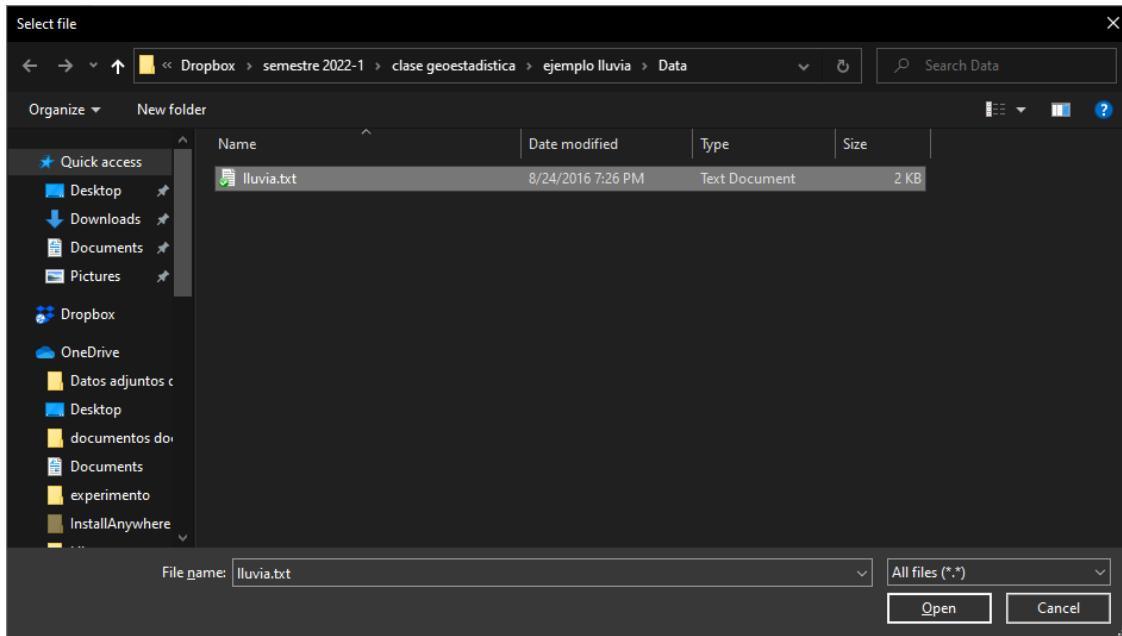
source("AllModel.R", encoding='ISO-8859-1')
source("BestModel.R")
source("CDF.R")
source("CoKrigingOrd.R")
source("CoKrigingOrdAnis.R")
source("CrossValidation.R")
source("CrossValidation2.R")
source("CrossVariograma.R")
source("DEspacial.R", encoding='ISO-8859-1')
source("Estadisticas.R")
source("EyeModel.R", encoding='ISO-8859-1')
source("FitDistribution.R", encoding='ISO-8859-1')
source("GDirecciones.R", encoding='ISO-8859-1')
source("HistBoxplot.R")
source("HistModel.R")
source("hist2.R")
source("KrigingOrd.R", encoding='ISO-8859-1')
source("KrigingOrdAnis.R", encoding='ISO-8859-1')
source("ModelVariogram.R")
source("nscore.R")
source("OutliersPos.R")
source("PPplot.R")
source("QQplot.R")
source("ScatterPlot.R")
source("scaterplotReg.R")
source("SGS.R")
source("Tendencia.R")
source("Trend.R")
source("Val_Estadisticos.R", encoding='ISO-8859-1')
source("Validacion.R", encoding='ISO-8859-1')
source("Variograma.R", encoding='ISO-8859-1')
source("Variograma4D.R", encoding='ISO-8859-1')

setwd(root_dir)

```

## 2.2 Carga de datos.

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

[5]: `Data_File_Burb <- read.csv(file='Data/BURB.csv', header=T, na.strings="-999.25")`

Para ordenar los resultados necesitamos crear una carpeta que usemos específicamente para el análisis exploratorio de datos (AED), ahí se almacenarán tablas e imágenes, esto lo hacemos con el comando "dir.create", donde le indicaremos la ruta donde se creará la carpeta AED.

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

[6]: `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 = "")):  
 "'/home/danielvr/Dropbox/Semestre 2023-1/ejemplo GAERM/ejemplo  
 GAERM/Results/Burb' already exists"

[7]: `# 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 = "")):  
 "'/home/danielvr/Dropbox/Semestre 2023-1/ejemplo GAERM/ejemplo  
 GAERM/Results/Burb/AED' already exists"

### 3 Análisis 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 análisis variográfico 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:

```
[8]: 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 estadígrafos, esto lo podemos calcular usando la función “estadísticas”. Es importante mencionar que los valores calculados en este paso se usarán en los gráficos.

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

#### 3.1 Análisis estadístico univariado.

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

Para obtener la tabla con los valores estadísticos, usamos la función “Val\_Estadisticos”.

```
[10]: 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=""))
       #esta linea sirve para guardar los resultados en un archivo csv
       print(Data_File_Burb_Stat[,3:4])
```

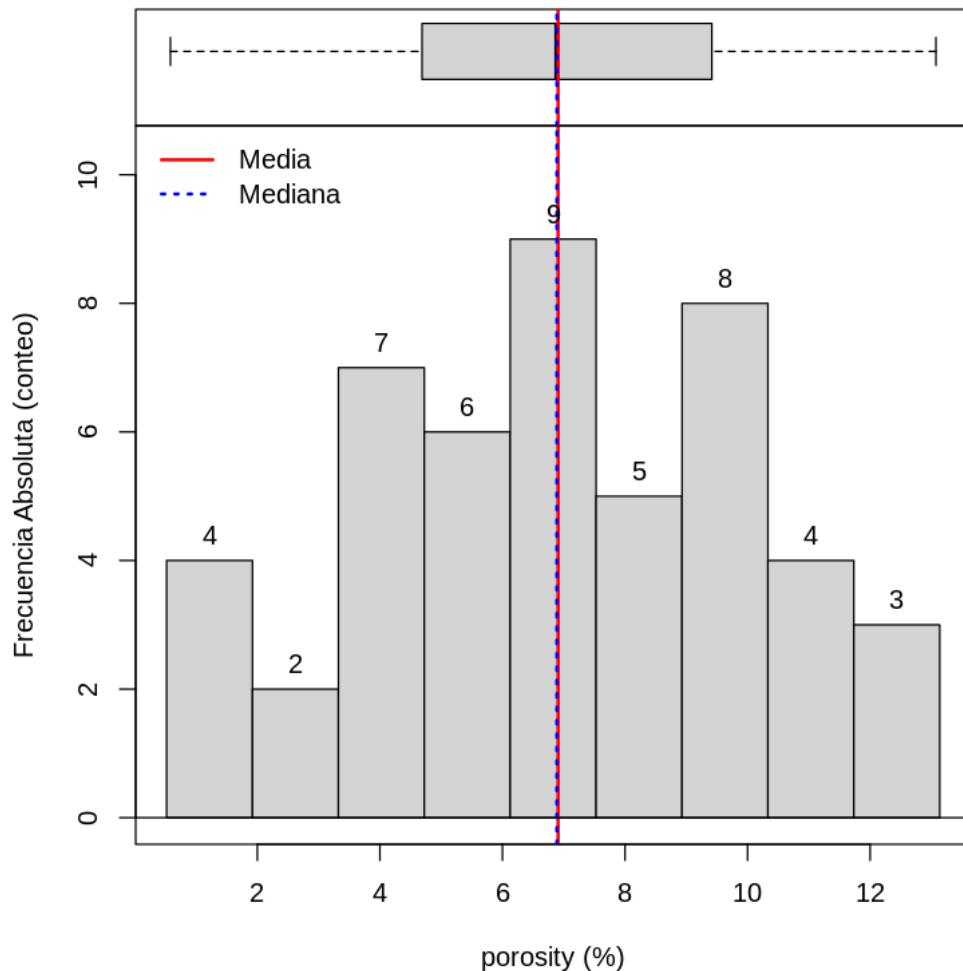
	k_mD	phi_percent
No_muestras	4.800000e+01	48.00000
Minimo	7.400000e+00	0.58200
Cuartil_1er	1.002450e+03	4.70125
Mediana	3.482205e+03	6.88800
Media	6.818245e+03	6.90763
Cuartil_3er	7.743625e+03	9.31300
Maximo	3.634740e+04	13.07300

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

### 3.1.1 Análisis estadístico univariado para la porosidad (phi\_per).

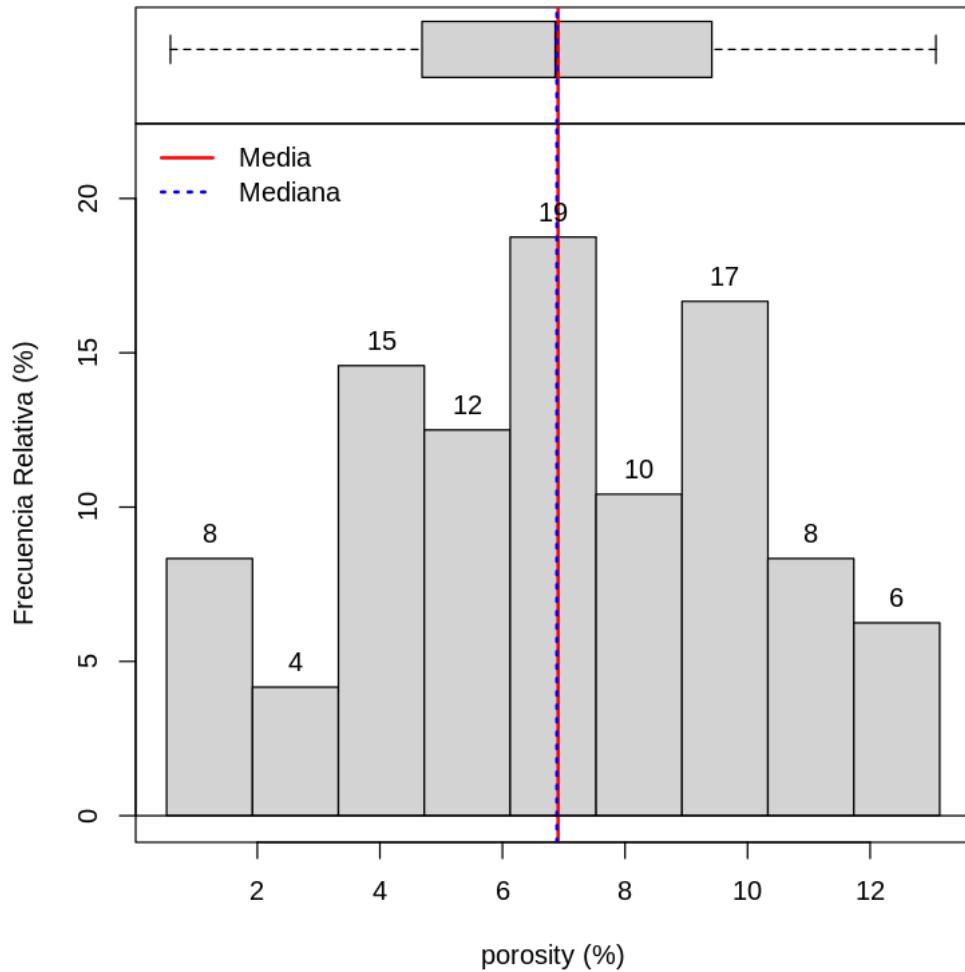
El histograma para la porosidad (phi\_per) con frecuencia absoluta es:

```
[11]: 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, PercentFreq = FALSE,
  nbin = 9)
```



Y el histograma de la porosidad con frecuencia relativa es:

```
[12]: HistBoxplot(x=phi_per, mean = phi_per_Stat[5,2], median = phi_per_Stat[4,2],  
                   main = "",  
                   xlab = "porosity (%)", ylab = "Frecuencia Relativa (%)", AbsFreq =  
                   FALSE, PercentFreq = TRUE,  
                   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”.

```
[13]: 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.

### 3.1.2 Análisis estadístico univariado para la permeabilidad (perm\_mD).

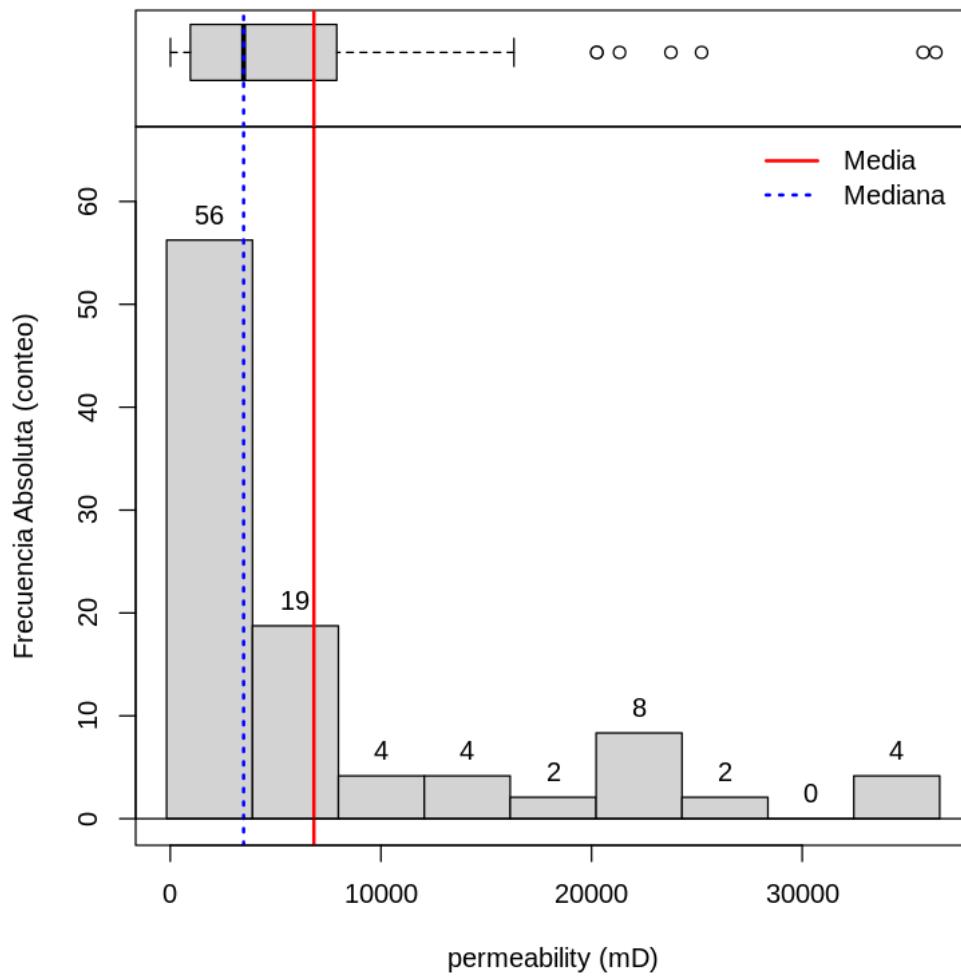
Ahora se hace el mismo análisis estadístico a los datos obtenidos de la permeabilidad (perm\_mD). Empezamos obteniendo los valores estadísticos.

```
[14]: perm_mD_Stat <- Estadisticas(perm_mD)
perm_mD_Stat
```

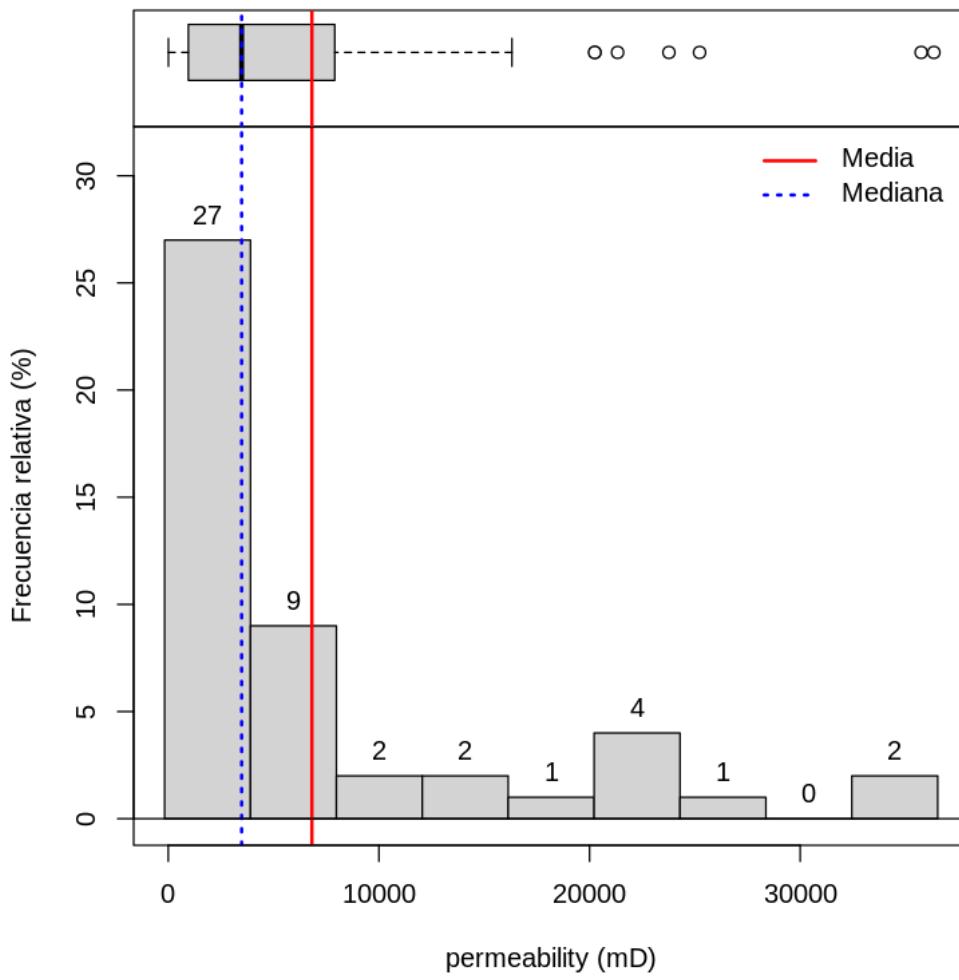
	Statistics <chr>	Values <dbl>
muestras	n	4.800000e+01
minimos	Minimum	7.400000e+00
cuantiles1	1st. Quartile	1.002450e+03
medianas	Median	3.482205e+03
medias	Mean	6.818245e+03
cuantiles3	3rd. Quartile	7.743625e+03
maximos	Maximum	3.634740e+04
rangos	Rank	3.634000e+04
rangosInt	Interquartile Rank	6.741175e+03
varianzas	Variance	8.353271e+07
desvs	Standard Deviation	9.139623e+03
CVs	Variation Coeff.	1.340500e+00
simetrias	Skewness	1.835800e+00
curtosiss	Kurtosis	5.626000e+00

Y su respectivo histograma.

```
[15]: 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 = FALSE, PercentFreq = TRUE,  
  ↪nbin = 9)
```



```
[16]: 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, PercentFreq = FALSE,  
  ↪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 quasi lineales
- Conveniencia.

Las transformaciones más comunes son:

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

Analizando el histograma podemos notar un fuerte crecimiento en la parte izquierda, por lo tanto usaremos la transformación logarítmica.

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

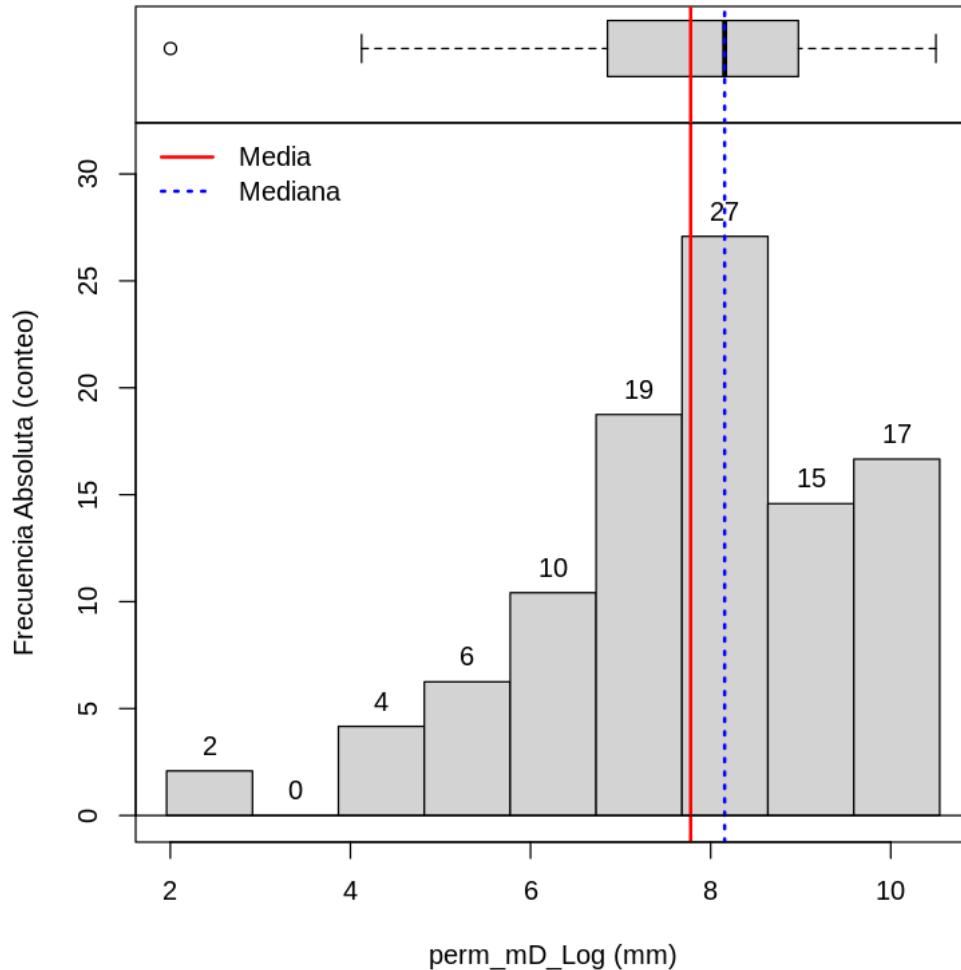
Y calculamos sus estadígrafos.

```
[18]: 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
```

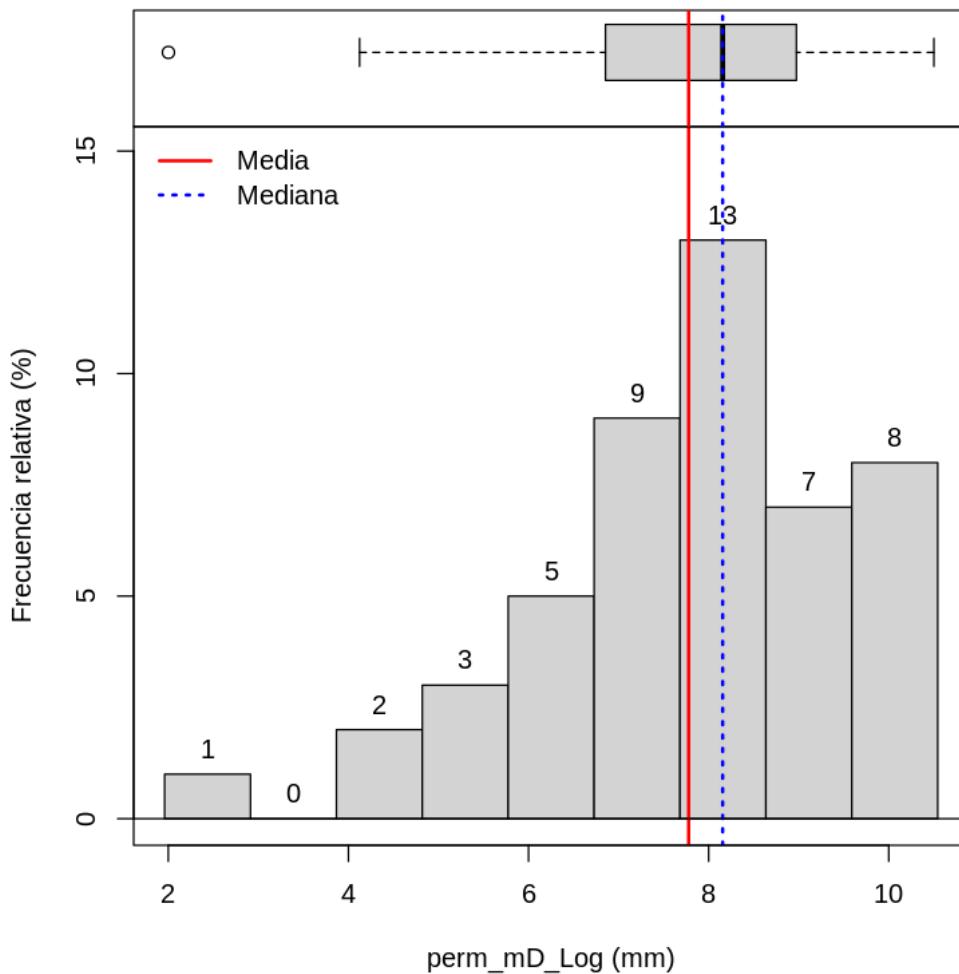
	Statistics	Values
	<chr>	<dbl>
muestras	n	48.0000
minimos	Minimum	2.0015
cuantiles1	1st. Quartile	6.9063
medianas	Median	8.1554
medias	Mean	7.7777
cuantiles3	3rd. Quartile	8.9540
maximos	Maximum	10.5009
rangos	Rank	8.4994
rangosInt	Interquartile Rank	2.0477
varianzas	Variance	3.2156
desvs	Standard Deviation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

Los histogramas de la transformación logarítmica son los siguientes:

```
[19]: HistBoxplot(x=perm_mD_Log, mean = perm_mD_Log_Stat[5,2], median =  
perm_mD_Log_Stat[4,2], main ="",  
xlab = "perm_mD_Log (mm)", ylab = "Frecuencia Absoluta (conteo)",  
AbsFreq = FALSE, PercentFreq = TRUE,  
nbin = 9)
```



```
[20]: HistBoxplot(x=perm_mD_Log, mean = perm_mD_Log_Stat[5,2], median =  
perm_mD_Log_Stat[4,2], main ="",  
xlab = "perm_mD_Log (mm)", ylab = "Frecuencia relativa (%)", AbsFreq =  
TRUE, PercentFreq = FALSE,  
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 gráfico, nos aseguraremos que esto sea cierto usando la función "OutliersPos".

```
[21]: 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 necesitaremos pa-

```
[22]: perm_mD_Log_DF<-cbind(XCoord,YCoord,perm_mD_Log)
perm_mD_Log_out_DF<-data.frame(perm_mD_Log_DF[-c(perm_mD_Log_outliers),])
perm_mD_Log_out<-perm_mD_Log_out_DF$perm_mD_Log
```

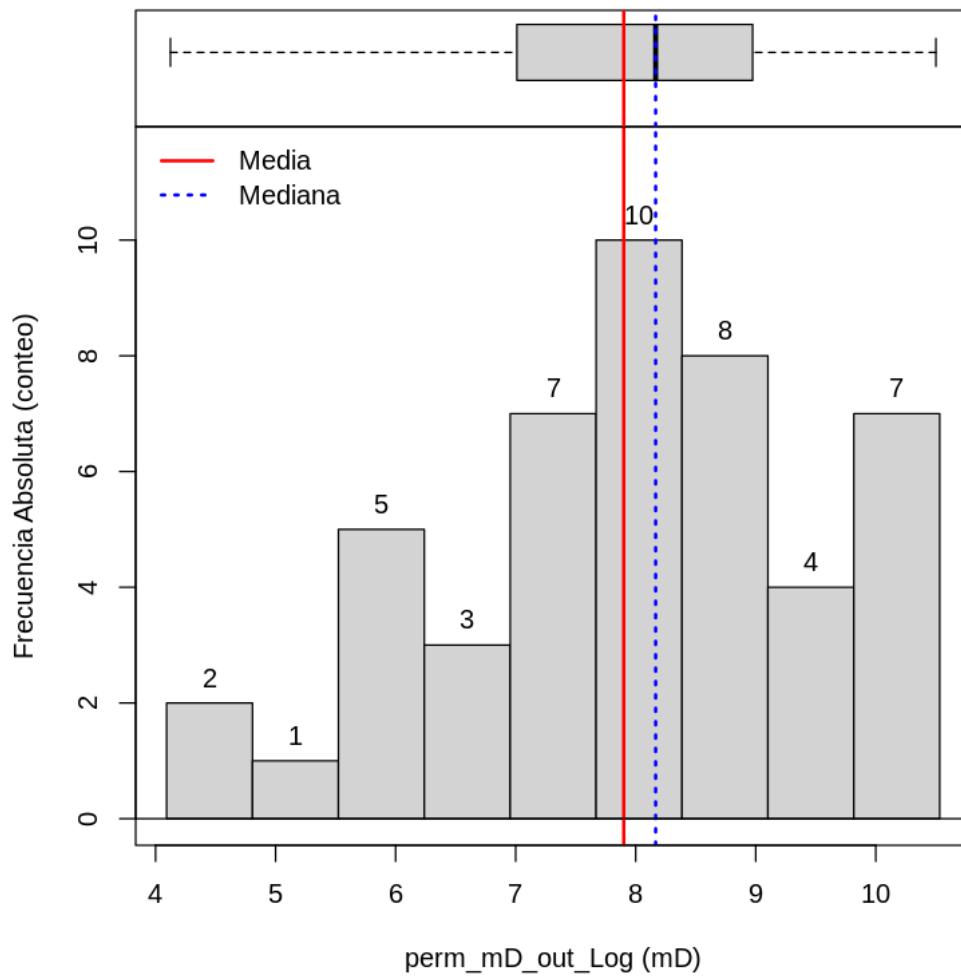
Después de retirar los valores atípicos, calculamos los estadígrafos de esta nueva variable aleatoria.

```
[23]: 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
```

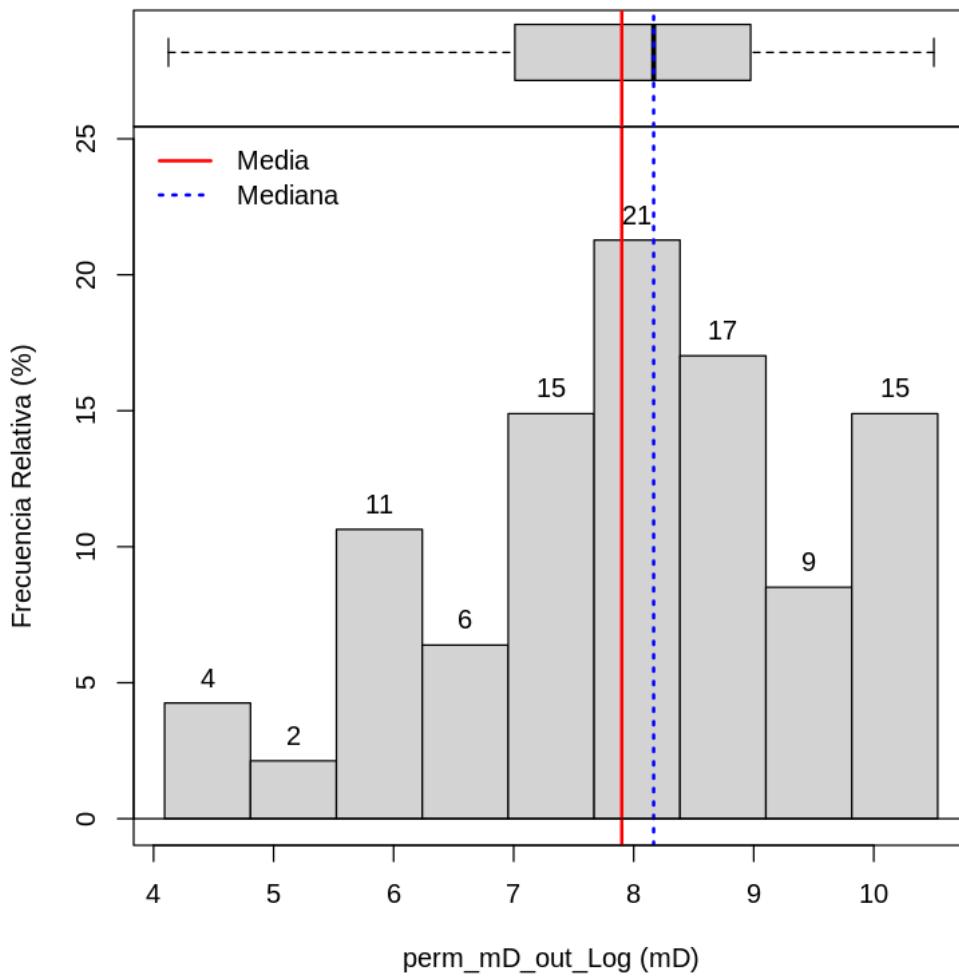
	Statistics <chr>	Values <dbl>
muestras	n	47.0000
minimos	Minimum	4.1239
cuantiles1	1st. Quartile	7.0100
medianas	Median	8.1663
medias	Mean	7.9006
A data.frame: 14 × 2	cuantiles3	3rd. Quartile
	maximos	8.9744
	rangos	Maximum
	rangosInt	10.5009
	varianzas	6.3770
	desvs	Interquartile Rank
	CVs	1.9643
	simetrias	Variance
	curtosiss	2.5447
		Standard Deviation
		0.2019
		Skewness
		Kurtosis

También graficamos sus respectivos histogramas.

```
[24]: HistBoxplot(x=perm_mD_Log_out, mean = perm_mD_Log_out_Stat[5,2], median =_
→perm_mD_Log_out_Stat[4,2], main ="",
xlab = "perm_mD_out_Log (mD)", ylab = "Frecuencia Absoluta_
→(conteo)", AbsFreq = TRUE, PercentFreq = FALSE,
nbin = 9)
```



```
[25]: HistBoxplot(x=perm_mD_Log_out, mean = perm_mD_Log_out_Stat[5,2], median = perm_mD_Log_out_Stat[4,2], main = "", xlab = "perm_mD_out_Log (mD)", ylab = "Frecuencia Relativa (%)", AbsFreq = FALSE, PercentFreq = TRUE, 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”.

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

```
numeric(0)
```

### 3.2 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:

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

### 3.2.1 Cálculo de grados de dependencia

Para calcular las medidas de dependencia usamos la función “cor”, esta necesita tres elementos: dos variables, en este caso (phi\_per , perm\_mD) y el método que se desea usar, este puede ser Pearson, Spearman o Kendall.

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

0.716875024758622

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

0.865935735996526

```
[29]: 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.

### 3.2.2 Diagrama de dispersión.

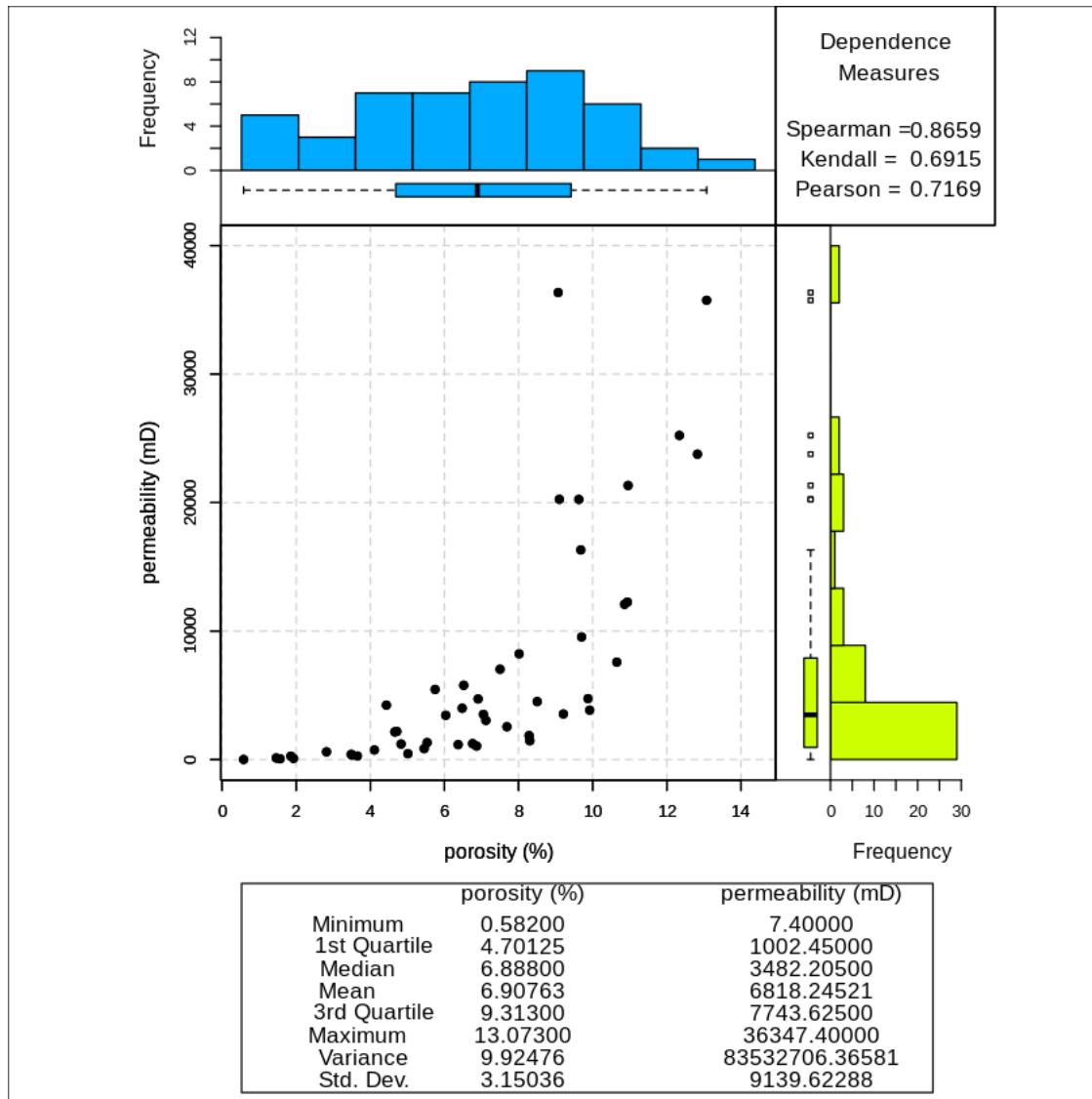
El diagrama de dispersión se grafica usando la función “ScatterPlot”, esta función requiere de los siguientes parámetros:

- Variables (phi\_per , perm\_mD)
- Número de intervalos para los histogramas, en este caso 9
- Valores mínimos y máximos de las variables analizadas (Xmin, Xmax, Ymin, Ymax), estos se pueden obtener de los estadígrafos calculados de cada variable

- Leyendas para el eje X (XLAB) y el eje Y (YLAB)

El diagrama de dispersión se grafica de la siguiente manera:

```
[30]: 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)")
```



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.

### 3.2.3 Análisis de regresión lineal.

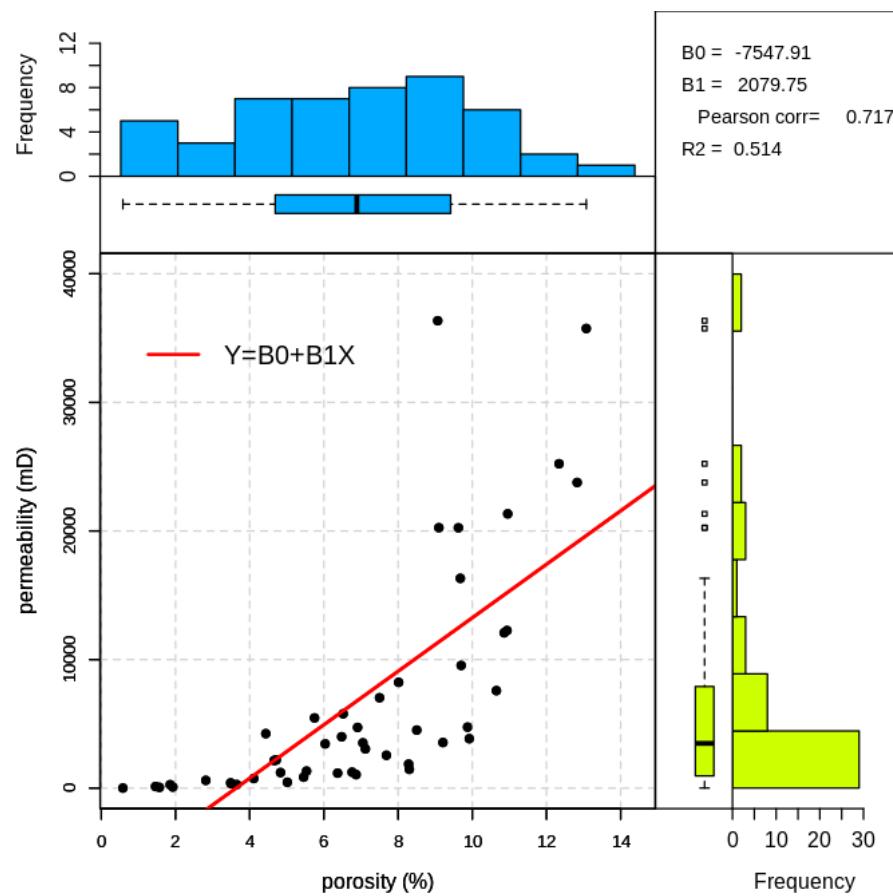
Como se mencionó en clase, la regresión trata de establecer relaciones funcionales entre variables aleatorias, en este caso, la relación se establece con una recta. Para hacer el análisis necesitamos de los parámetros de la recta y el análisis de residuos.

Para obtener este gráfico se usa la función “scaterplotReg”, la cual necesita de los siguientes parámetros:

- Variables (phi\_per , perm\_mD)
- Número de intervalos para los histogramas, en este caso 9
- Valores mínimos y máximos de las variables analizadas (Xmin, Xmax, Ymin, Ymax), estos se pueden obtener de los estadígrafos calculados de cada variable
- Leyendas para el eje X (XLAB) y el eje Y (YLAB)

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

```
[31]: 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 se requiere de los valores de la regresión lineal y su error cuadrático. Para hacer la regresión lineal se usa la función “lm”, esta necesita los siguientes parámetros:

- Indicar las variables X y Y

```
[32] : 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 cálculo de los residuos.

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

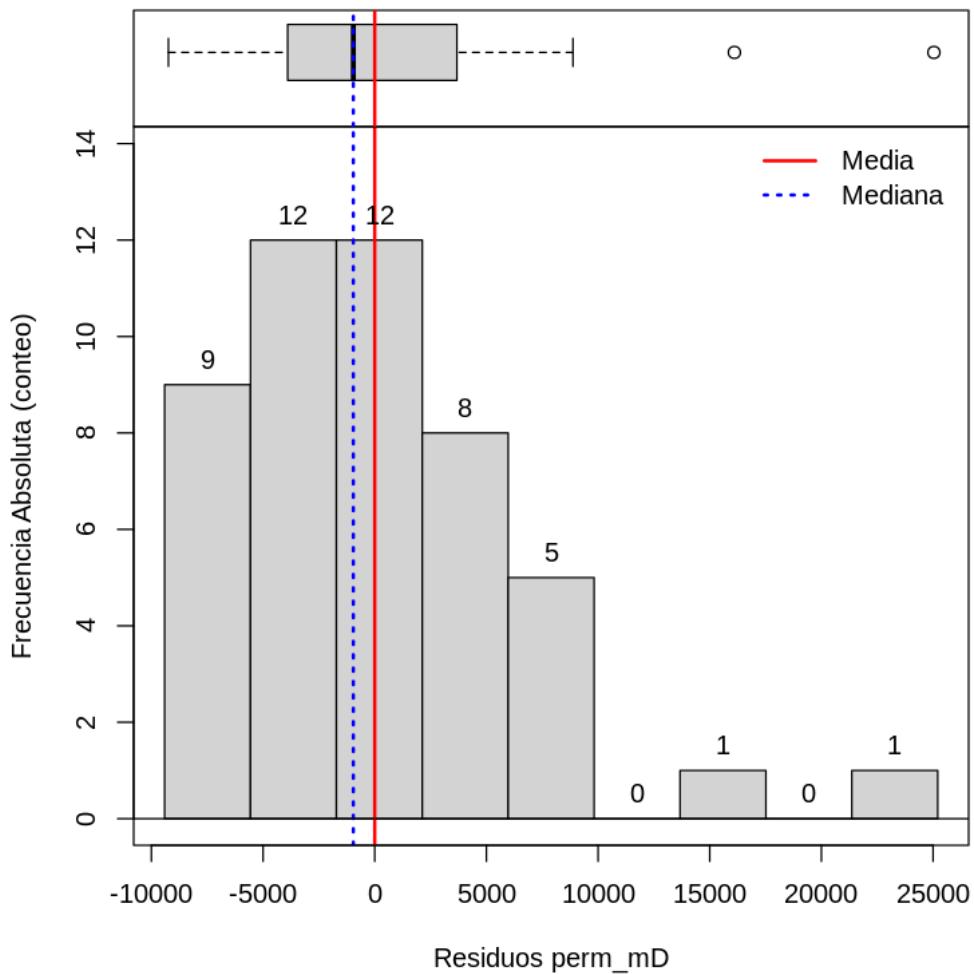
Ya que tenemos calculados los residuos necesitamos obtener sus valores estadísticos.

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

	Statistics <chr>	Values <dbl>
	muestras	4.800000e+01
	minimos	-9.237240e+03
	cuantiles1	-3.748058e+03
	medianas	-9.578251e+02
	medias	0.000000e+00
A data.frame: 14 × 2	cuantiles3	3.650538e+03
	maximos	2.503611e+04
	rangos	3.427335e+04
	rangosInt	7.398596e+03
	varianzas	4.060443e+07
	desvs	6.372160e+03
	CVs	1.197399e+17
	simetrias	1.496600e+00
	curtosiss	6.718600e+00

Y también necesitamos obtener el histograma de estos residuos.

```
[35]: HistBoxplot(x=Y_Residual, mean = Y_Residual_Stat[5,2], median =  
→Y_Residual_Stat[4,2], main ="",  
xlab = "Residuos perm_mD", ylab = "Frecuencia Absoluta (conteo)",  
→AbsFreq = TRUE, PercentFreq = FALSE )
```



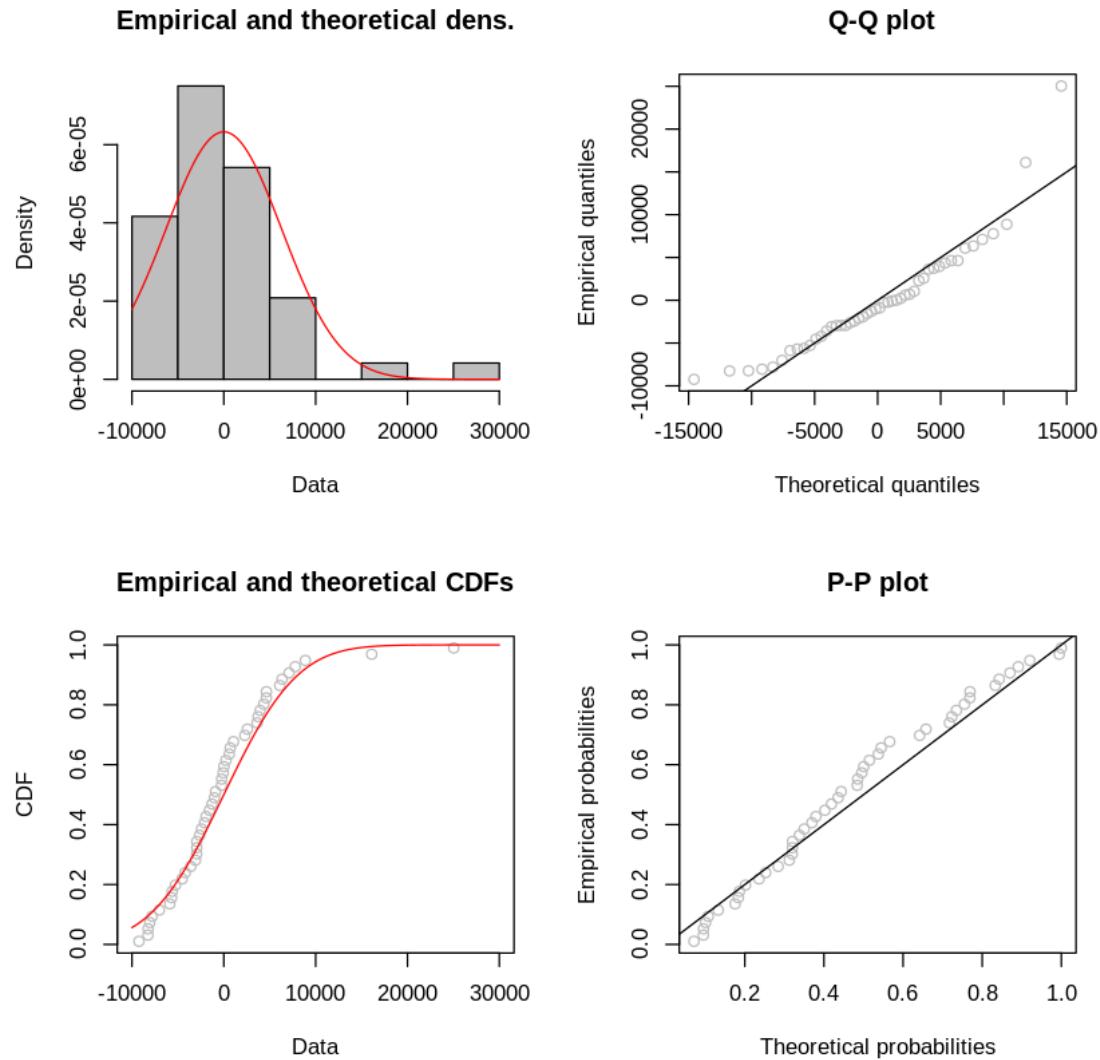
Si analizamos los valores estadísticos y el histograma de los residuos, podemos notar que el valor esperado es de -0.0016, lo cual se podría considerar cercano a cero, su varianza es de 0.2947 y la diferencia entre la media y la mediana es de 0.1564, lo cual nos indica que tiene asimetría negativa, por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

Para confirmar que los residuos no cumplen con las condiciones de la regresión lineal hay que usar los gráficos de comparación empírica y teórica, estos son el histograma, grafico cuantil-cuantil (Q-Q), percentil-percentil (P-P) y de función acumulativa. Para esto se usa la función “FitDistribution”, esta requiere de los siguientes parámetros:

- Vector de los residuos (data)
- Tipo de distribución de probabilidad, en este caso normal (norm)
- Número de intervalos para el histograma (BREAKS)
- Color para los intervalos del histograma (col)

```
[36]: FitDistr2_Residual_normal<-FitDistribution(data = Y_Residual, DISTR="norm",
→BREAKS = "Sturges", col = "gray", DistName = "Normal")
```

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 una prueba 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:

$$D = mx|F_n(x) - F_{va}(x)|$$

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:

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

El valor p es una probabilidad que mide la evidencia en contra de la hipótesis nula. Un valor p más pequeño proporciona una evidencia más fuerte en contra de la hipótesis nula. Esta hipótesis se usa para determinar si los datos siguen una distribución normal.

Si  $p \leq a$  donde a es el nivel de significancia la decisión es rechazar la hipótesis nula y concluir que sus datos no siguen una distribución normal.

Si  $p > a$  donde a es el nivel de significancia la decisión es no rechazar la hipótesis nula y concluir que sus datos no tienen suficiente evidencia para concluir que los datos no siguen una distribución normal.

```
[37] : FD_HT_Residual_normal<-FitDistr2_Residual_normal$x
FD_HT_Residual_normal
```

	Method <chr>	Significance level <chr>	P-value <chr>	Statistical <chr>	Decision <chr>
A data.frame: 2 × 5	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 gráfico Q-Q plot los valores no están muy alejados de la recta esperada.

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

```
[38] : FD_FP_Residual_normal<-FitDistr2_Residual_normal$y
FD_FP_Residual_normal
```

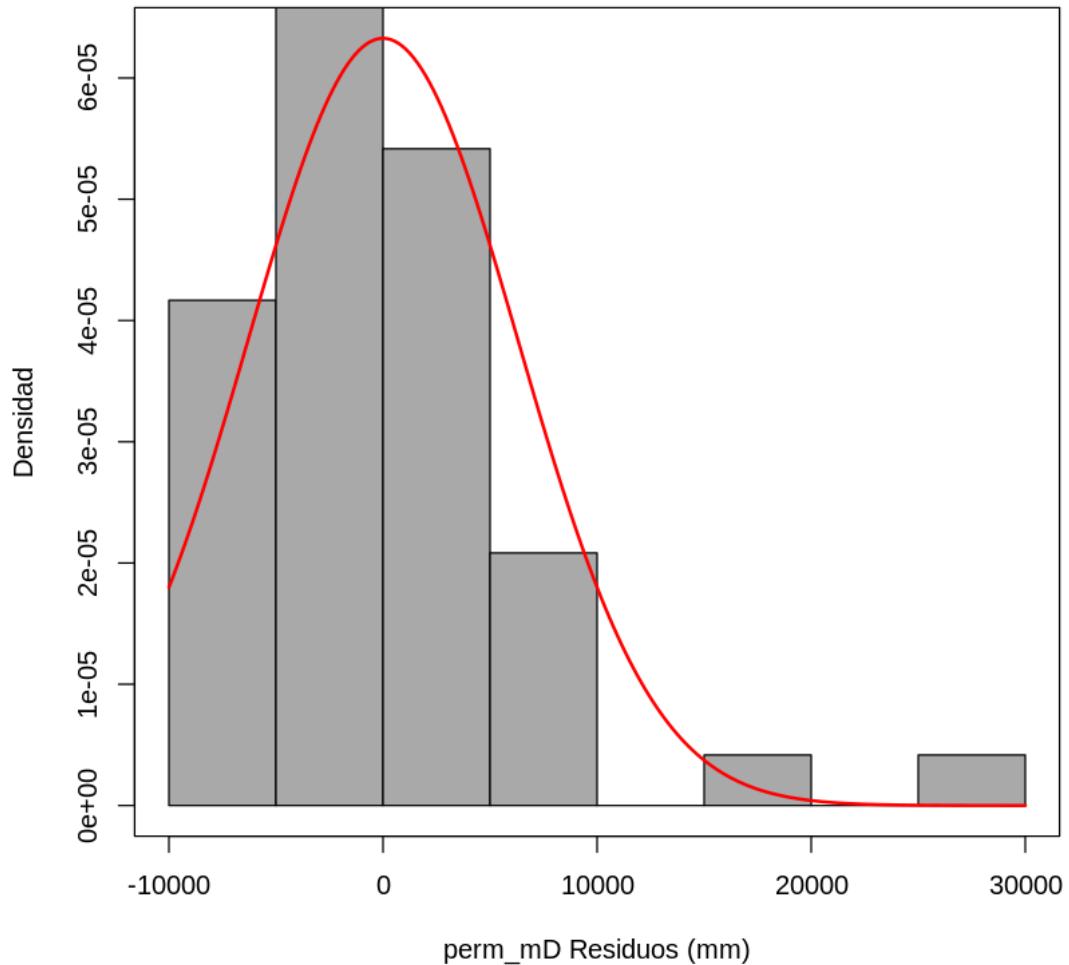
	Normal <dbl>
A data.frame: 4 × 1	Mean 5.321669e-14
	Standard deviation 6.305434e+03
	Maximum likelihood -4.880691e+02
	AIC 9.801381e+02

Los siguientes gráficos son los mismos que se analizaron en el vector “FitDistr2\_Residual\_normal”, tenemos el histograma la función de distribución normal. Este se puede graficar usando la función “Histmodel”, los parámetros que necesita son los siguientes:

- Vector de los residuos (x)
- Tipo de distribución (distr), en este caso normal
- Parámetros de la distribución de probabilidad seleccionada (para), en este caso esos fueron estimados durante el análisis de los residuos y deben ingresarse en formato de lista
- Número de intervalos (breaks), en este caso se usa el método de Sturges
- Se indica si el histograma es de frecuencia absoluta (freq), por default es FALSE
- Se indica el título del gráfico (main)
- Se indica las leyendas de los ejes X (xlab) y Y (ylab)
- Se indica el color de la curva de la función de densidad (colCurve)
- Se indica el color de los intervalos del histograma (col)

```
[39]: PARA_Residual_normal <- list(mean = as.numeric(FD_FP_Residual_normal[1,1]), sd =  
    ↪as.numeric(FD_FP_Residual_normal[2,1]))  
  
HistModel(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, breaks =  
    ↪"Sturges", freq = FALSE,  
    main ="Histograma y Ajuste", xlab = "perm_mD Residuos (mm)",  
    ylab = "Densidad", colCurve =  "red", col = "darkgray")
```

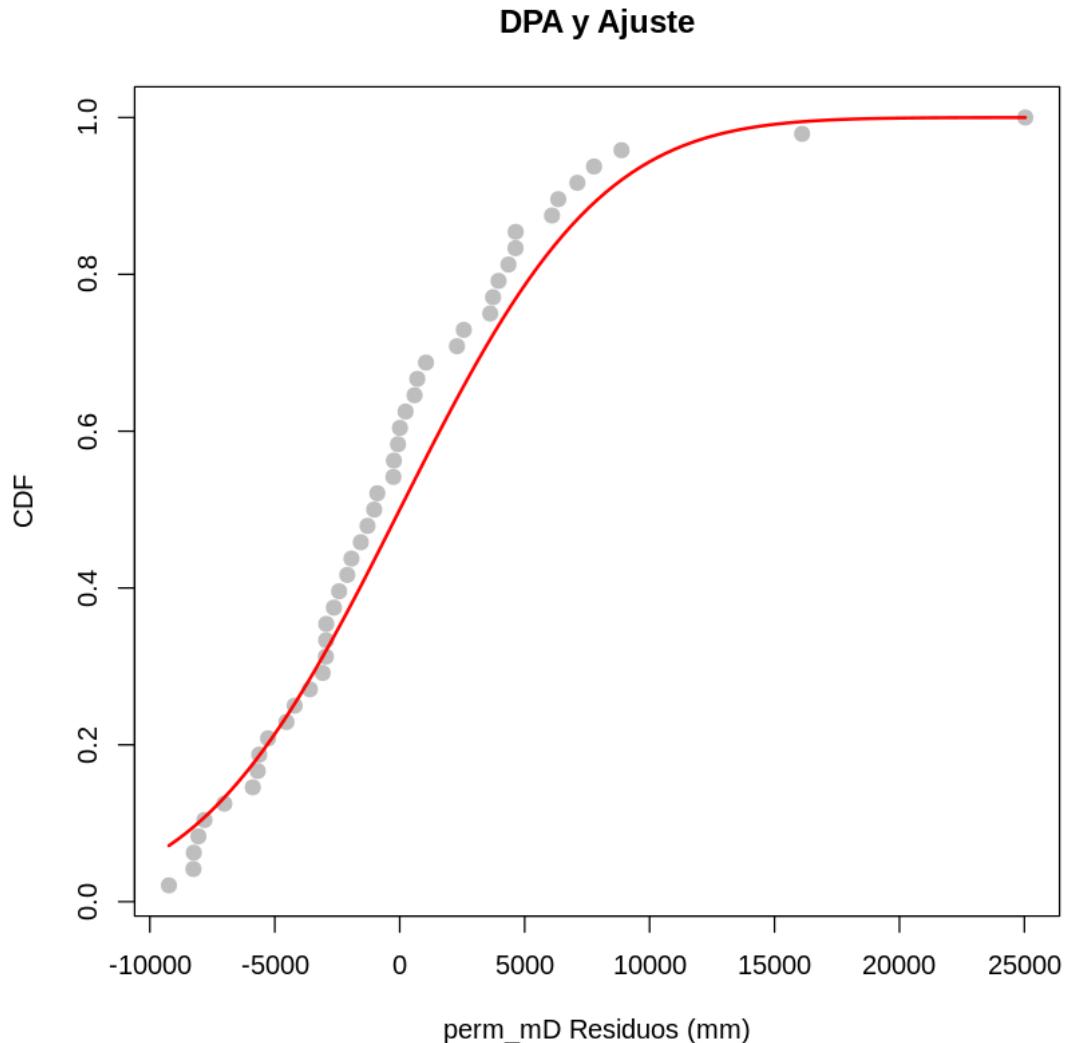
## Histograma y Ajuste



Se grafica su función de distribución acumulativa. Para esto se usa la función “CDF”, esta requiere de los siguientes parámetros:

- Vector de los residuos (x)
- Tipo de distribución (distr), en este caso normal
- Parámetros de la distribución de probabilidad seleccionada (para), en este caso esos fueron estimados durante el análisis de los residuos y deben ingresarse en formato de lista
- Se indica el color de la función empírica (col)
- Se indica el título del gráfico (main)
- Se indica la leyenda del eje X (xlab)
- Se indica el color de la curva de la función teórica (lcol)
- Se indica el grosor de la línea de la función teórica (lwd)

```
[40]: CDF(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray",  
         main = "DPA y Ajuste",  
         xlab = "perm_mD Residuos (mm)", lcol = "red", lwd = 2)
```

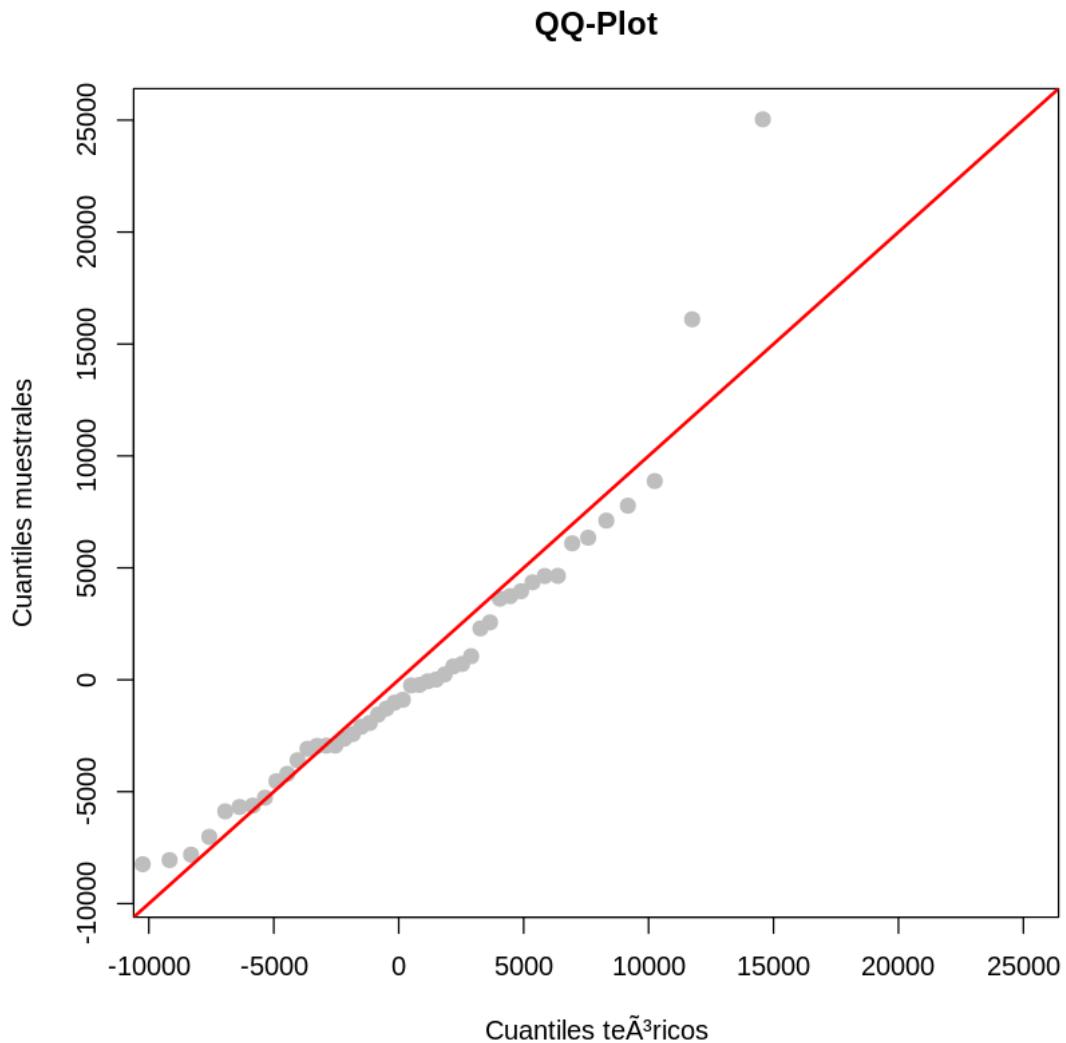


El Grafico cuantil-cuantil (Q-Q plot) se puede graficar usando la función “QQplot”, esta requiere de los siguientes parámetros:

- Vector de los residuos (x)
- Tipo de distribución (distr), en este caso normal
- Parámetros de la distribución de probabilidad seleccionada (para), en este caso esos fueron estimados durante el análisis de los residuos y deben ingresarse en formato de lista
- Se indica el color de la función empírica (col)
- Se indica el título del gráfico (main)
- Se indica la leyenda del eje X (xlab)

- Se indica el color de la curva de la función teórica (lcol)
- Se indica el grosor de la línea de la función teórica (lwd)

```
[41]: QQplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray",
           main = "QQ-Plot", xlab = "Cuantiles teóricos",
           lcol = "red", lwd = 2)
```

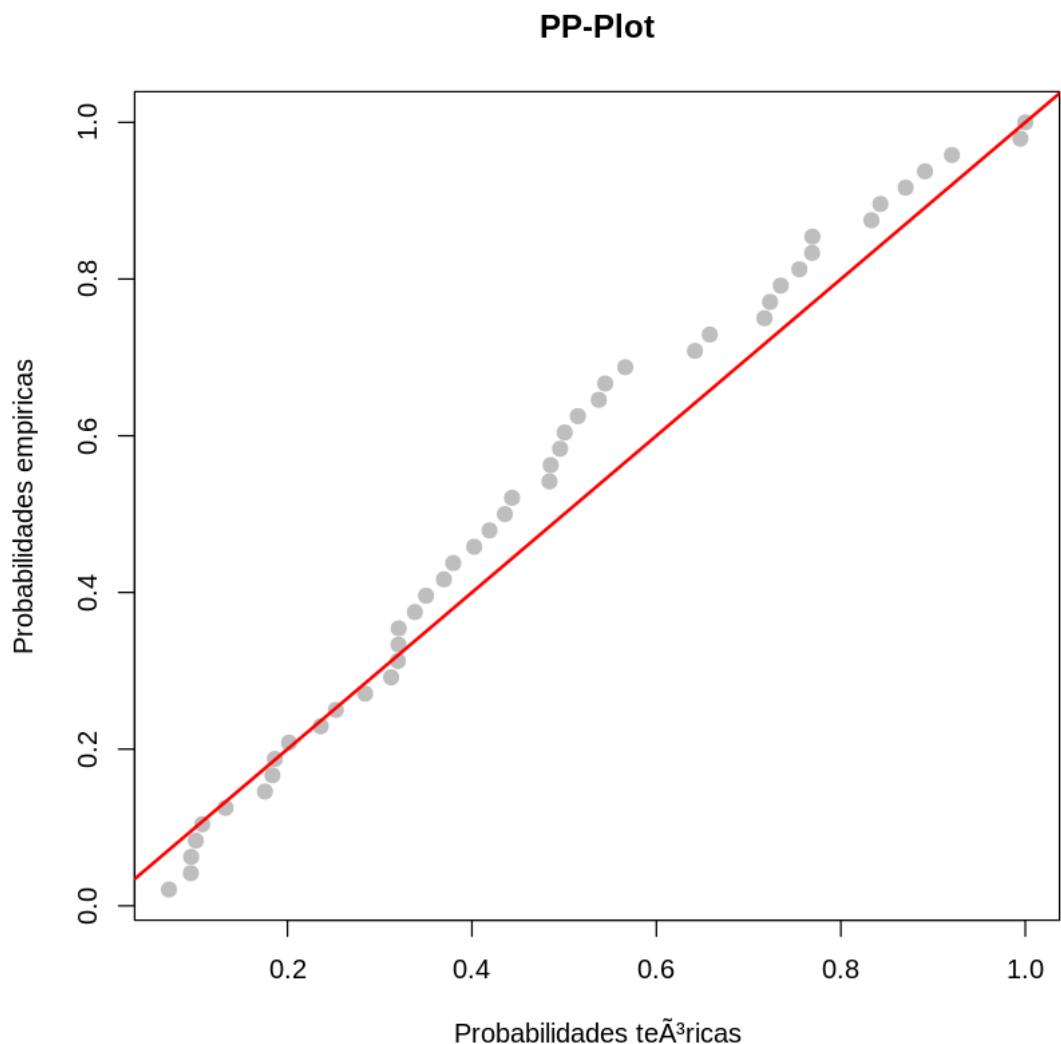


El Grafico percentil-percentil (P-P plot) se puede graficar usando la función “PPplot”, esta requiere de los siguientes parámetros:

- Vector de los residuos (x)
- Tipo de distribución (distr), en este caso normal
- Parámetros de la distribución de probabilidad seleccionada (para), en este caso esos fueron estimados durante el análisis de los residuos y deben ingresarse en formato de lista

- Se indica el color de la función empírica (col)
- Se indica el título del gráfico (main)
- Se indica la leyenda del eje X (xlab)
- Se indica el color de la curva de la función teórica (lcol)
- Se indica el grosor de la línea de la función teórica (lwd)

```
[42]: PPplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main = "PP-Plot", xlab = "Probabilidades teóricas", lcol = "red", lwd = 2)
```



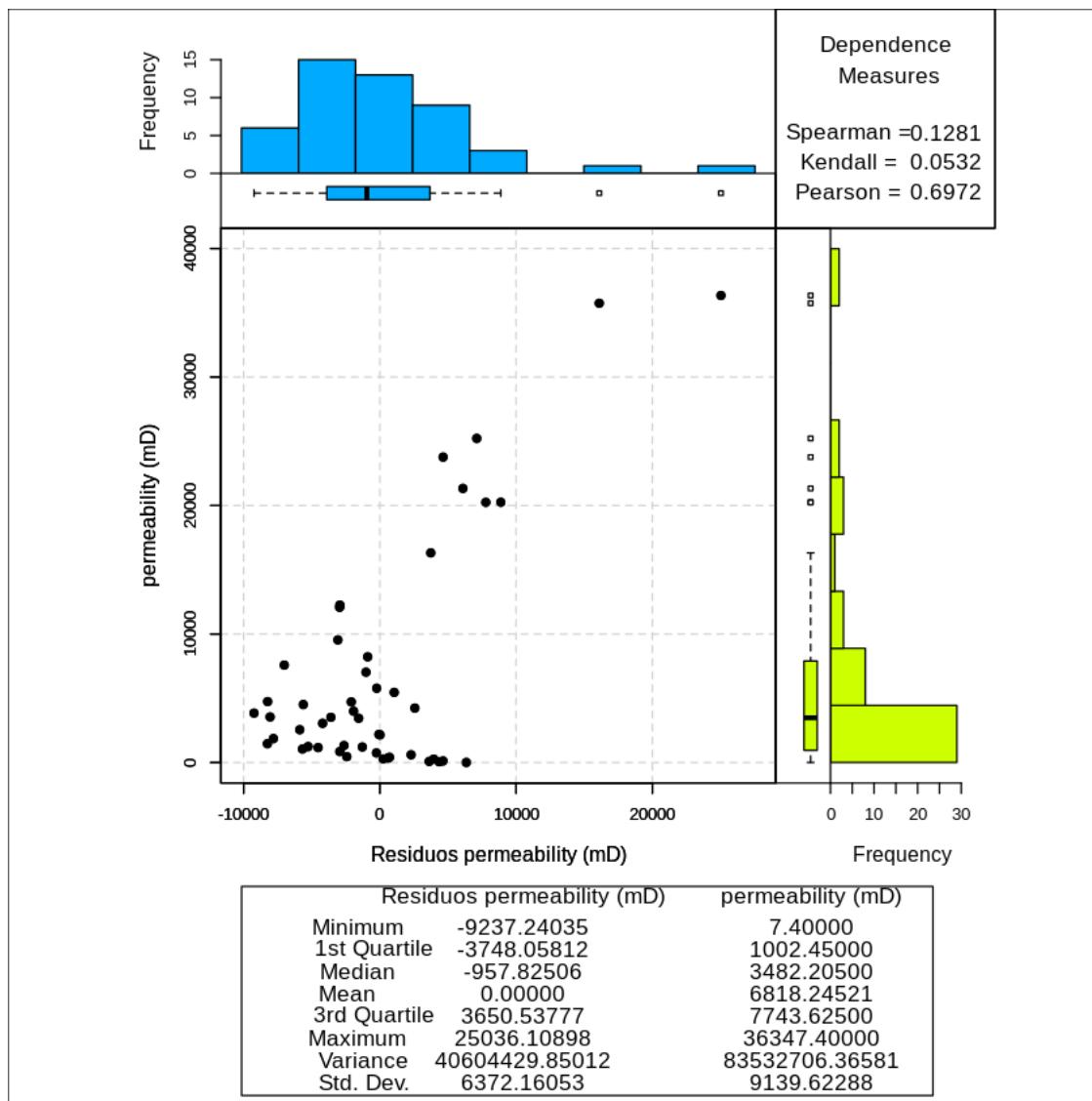
### 3.2.4 Análisis bivariado: Y vs Y residual.

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

```
[43]: 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_U
            →permeability (mD)", YLAB = "permeability (mD)")
```



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 de la 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.

### 3.3 Análisis estadístico bivariado con variables transformadas.

Ahora haremos el caso donde las variables (phi\_per, perm\_mD\_Log) 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 de la permeabilidad.

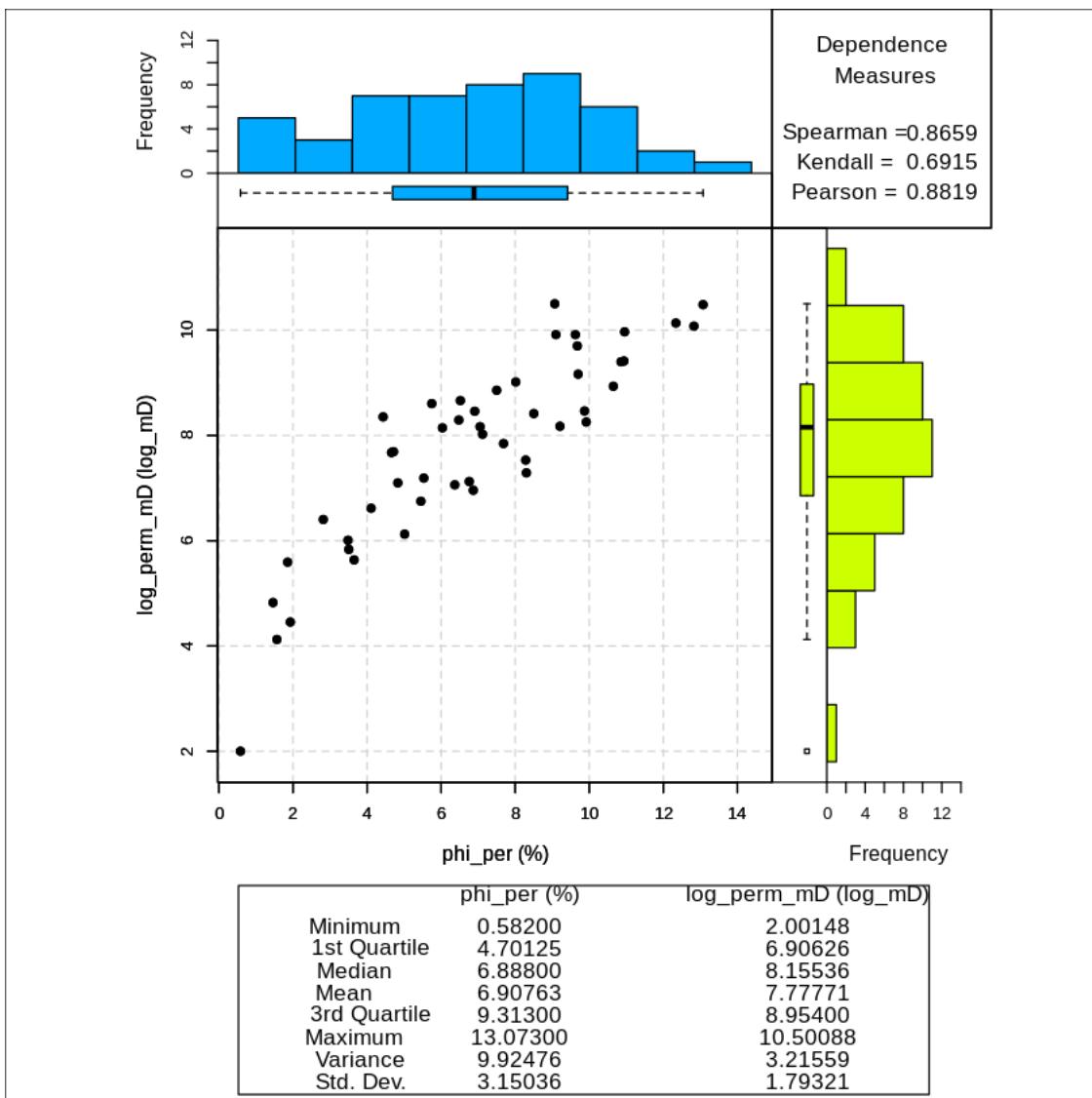
#### 3.3.1 Grafico de dispersión.

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.

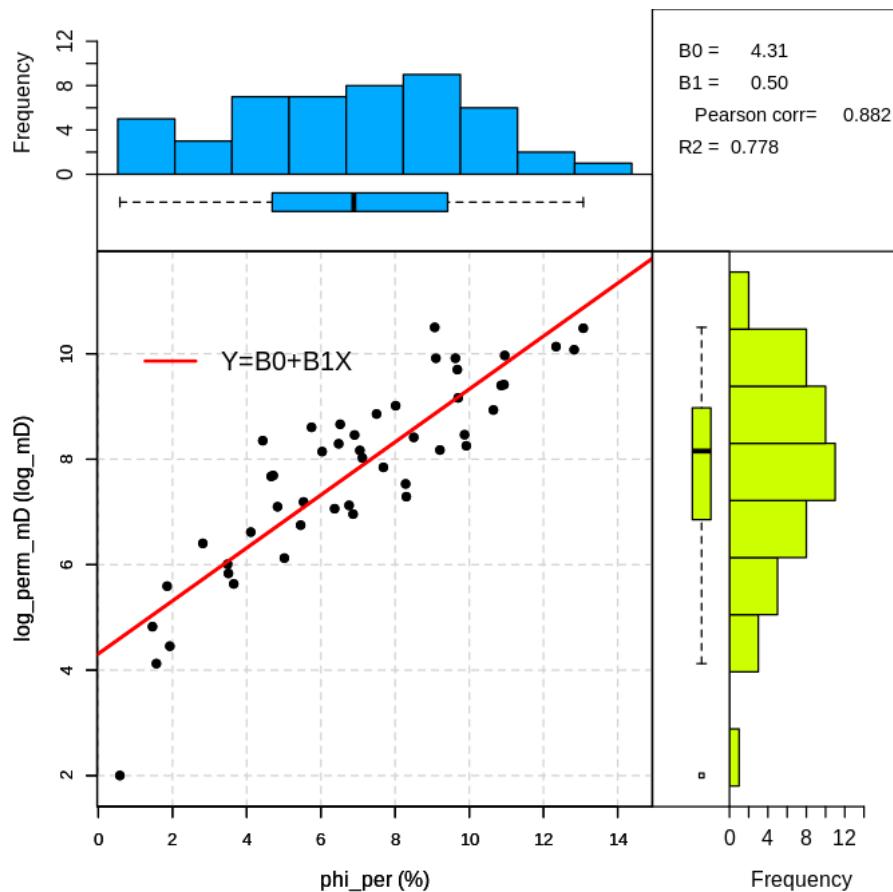
```
[44]: ScatterPlot(phi_per , perm_mD_Log, 9,  
                  Xmin = phi_per_Stat[2,2] , Xmax = phi_per_Stat[7,2] ,  
                  Ymin = perm_mD_Log_Stat[2,2] , Ymax = perm_mD_Log_Stat[7,2] ,  
                  XLAB = "phi_per (%)", YLAB = "log_perm_mD (log_mD)")
```



### 3.3.2 Análisis de regresión lineal.

Ahora obtendremos los valores de la regresión lineal .

```
[45]: 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

```
[46]: 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

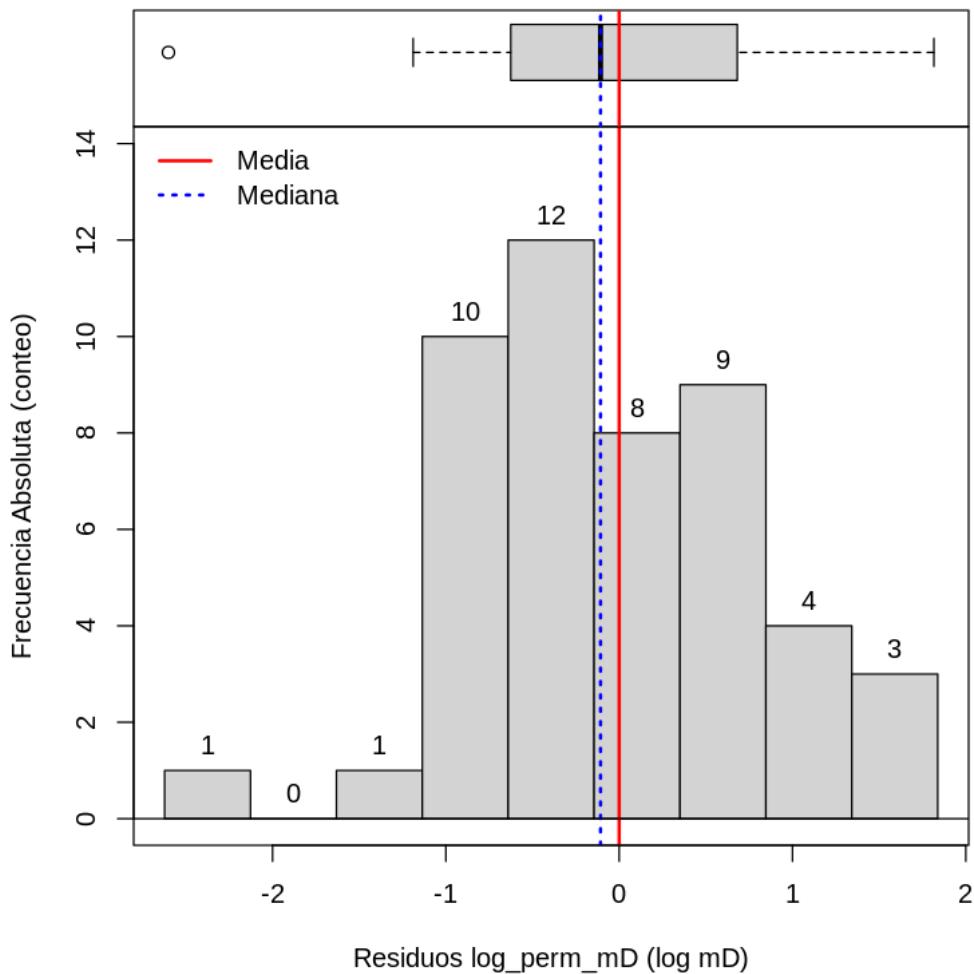
X: 0.501986643878787

Y calculamos sus estadígrafos e histograma.

```
[47]: 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
```

	Statistics <chr>	Values <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
A data.frame: 14 × 2	cuantiles3	3rd. Quartile
	maximos	6.815000e-01
	rangos	Maximum
	rangosInt	1.815900e+00
	varianzas	Rank
	desvs	Interquartile Rank
	CVs	7.146000e-01
	simetrias	Standard Deviation
	curtosiss	Variation Coeff.
		Skewness
		Kurtosis

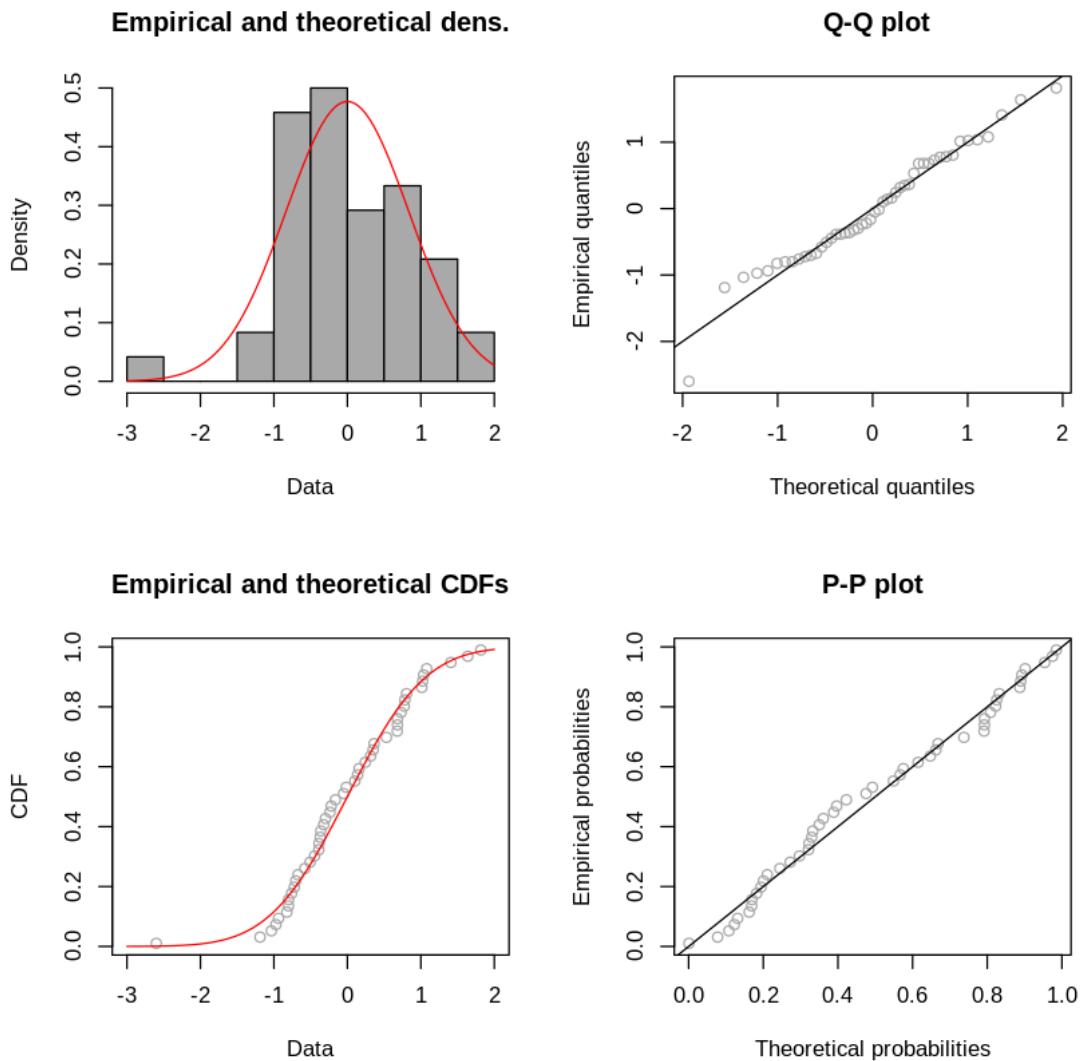
```
[48]: HistBoxplot(x=Y_Residual, mean = Y_Residual_Stat[5,2], median =  
→Y_Residual_Stat[4,2], main ="",  
xlab = "Residuos log_perm_mD (log mD)", ylab = "Frecuencia Absoluta  
→(conteo)", AbsFreq = TRUE, PercentFreq = FALSE )
```



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.

```
[49]: FitDistr2_Residual_normal<-FitDistribution(data = Y_Residual, DISTR="norm",
  ↪BREAKS = "Sturges", col = "darkgray", DistName = "Normal")
```

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 que casi 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.

```
[50] : FD_HT_Residual_normal<-FitDistr2_Residual_normal$x
FD_HT_Residual_normal
```

	Method	Significance level	P-value	Statistical	Decision
	<chr>	<chr>	<chr>	<chr>	<chr>
A data.frame: 2 × 5	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.

```
[51]: FD_FP_Residual_normal<-FitDistr2_Residual_normal$y
FD_FP_Residual_normal
```

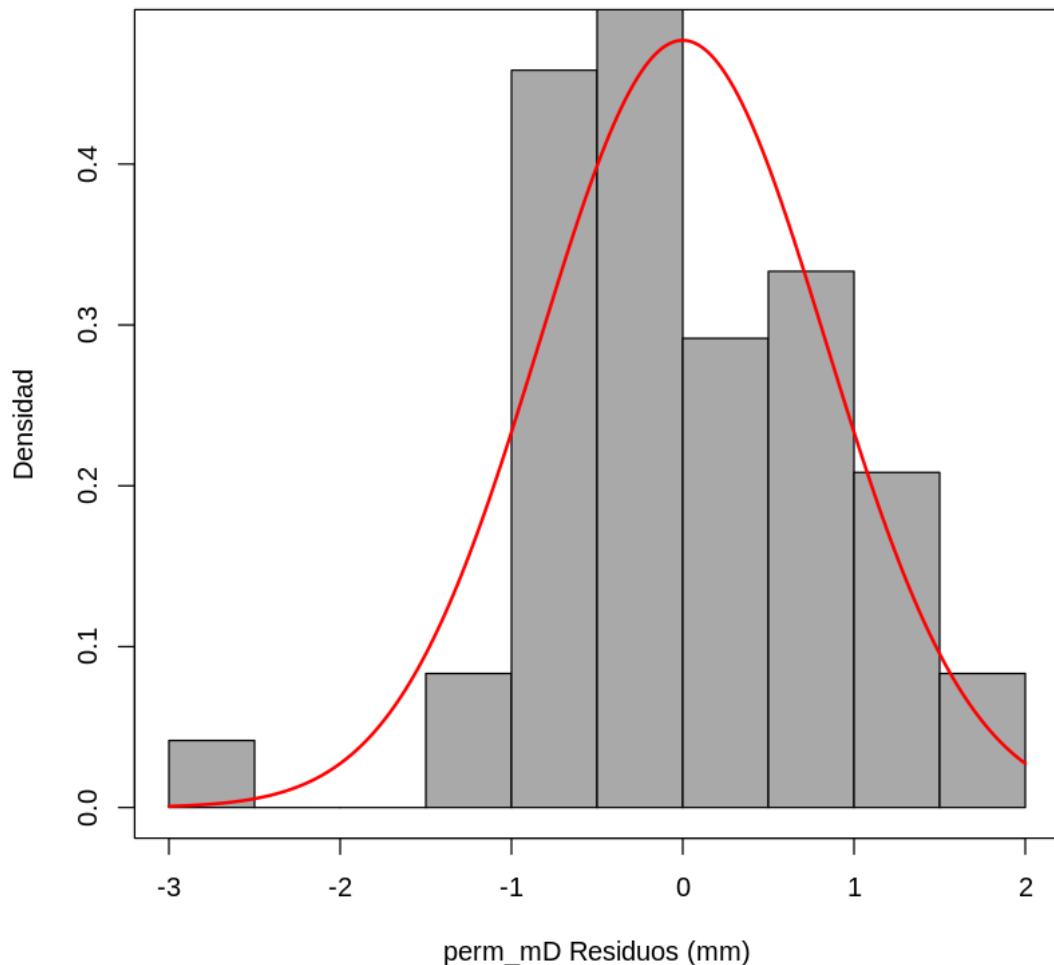
		Normal
		<dbl>
A data.frame: 4 × 1	Mean	-1.033380e-19
	Standard deviation	8.365121e-01
	Maximum likelihood	-5.954036e+01
	AIC	1.230807e+02

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

```
[52]: PARA_Residual_normal <- list(mean = as.numeric(FD_FP_Residual_normal[1,1]),
                                 sd = as.numeric(FD_FP_Residual_normal[2,1]))

HistModel(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, breaks = "Sturges",
          freq = FALSE,
          main ="Histograma y Ajuste", xlab = "perm_mD Residuos (mm)",
          ylab = "Densidad", colCurve = "red", col = "darkgray")
```

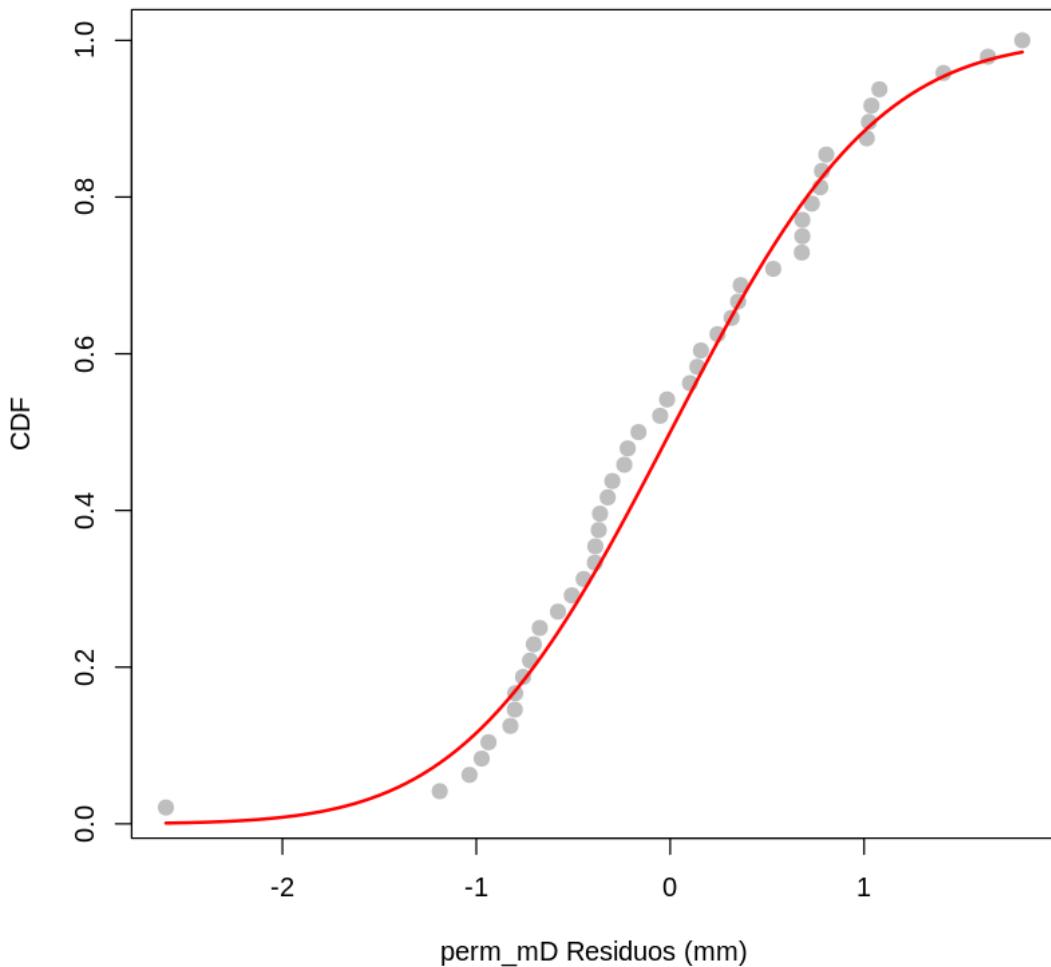
## Histograma y Ajuste



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

```
[53]: CDF(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray",  
        main = "DPA y Ajuste",  
        xlab = "perm_mD Residuos (mm)", lcol = "red", lwd = 2)
```

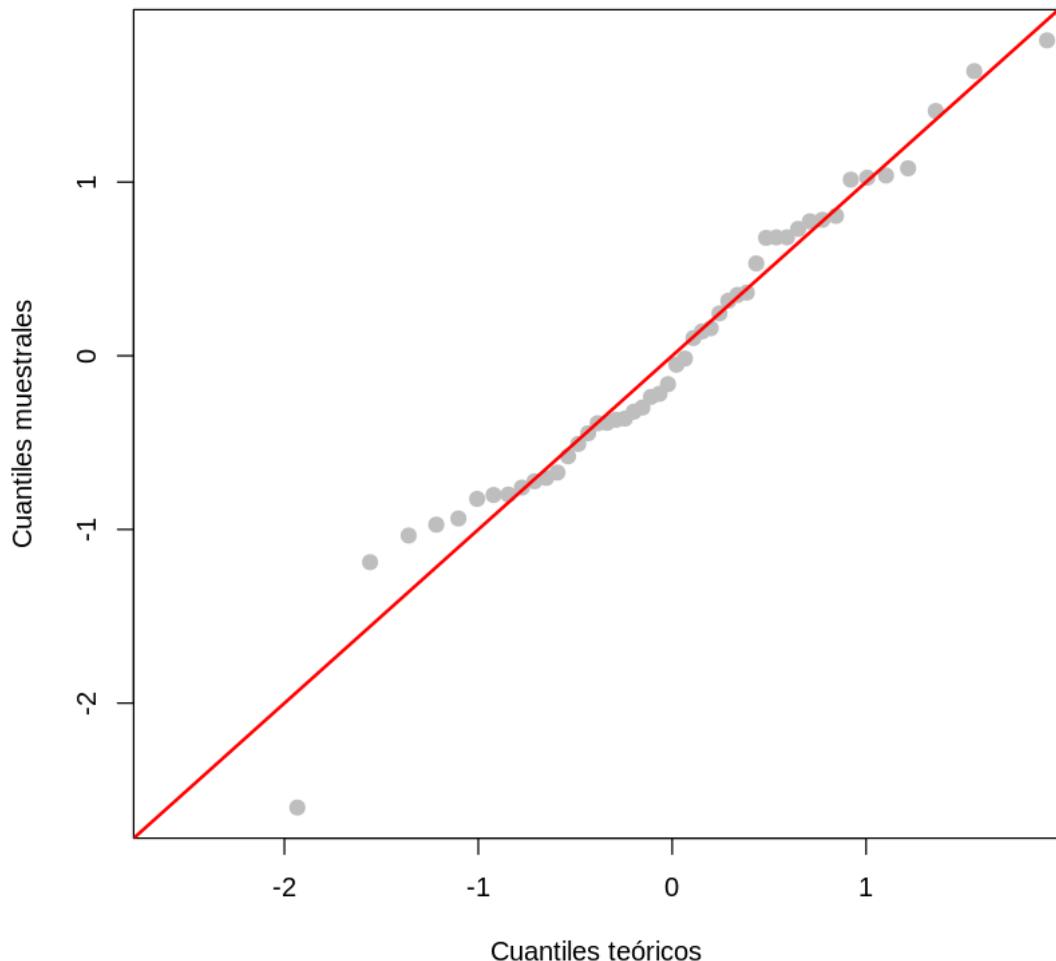
## DPA y Ajuste



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

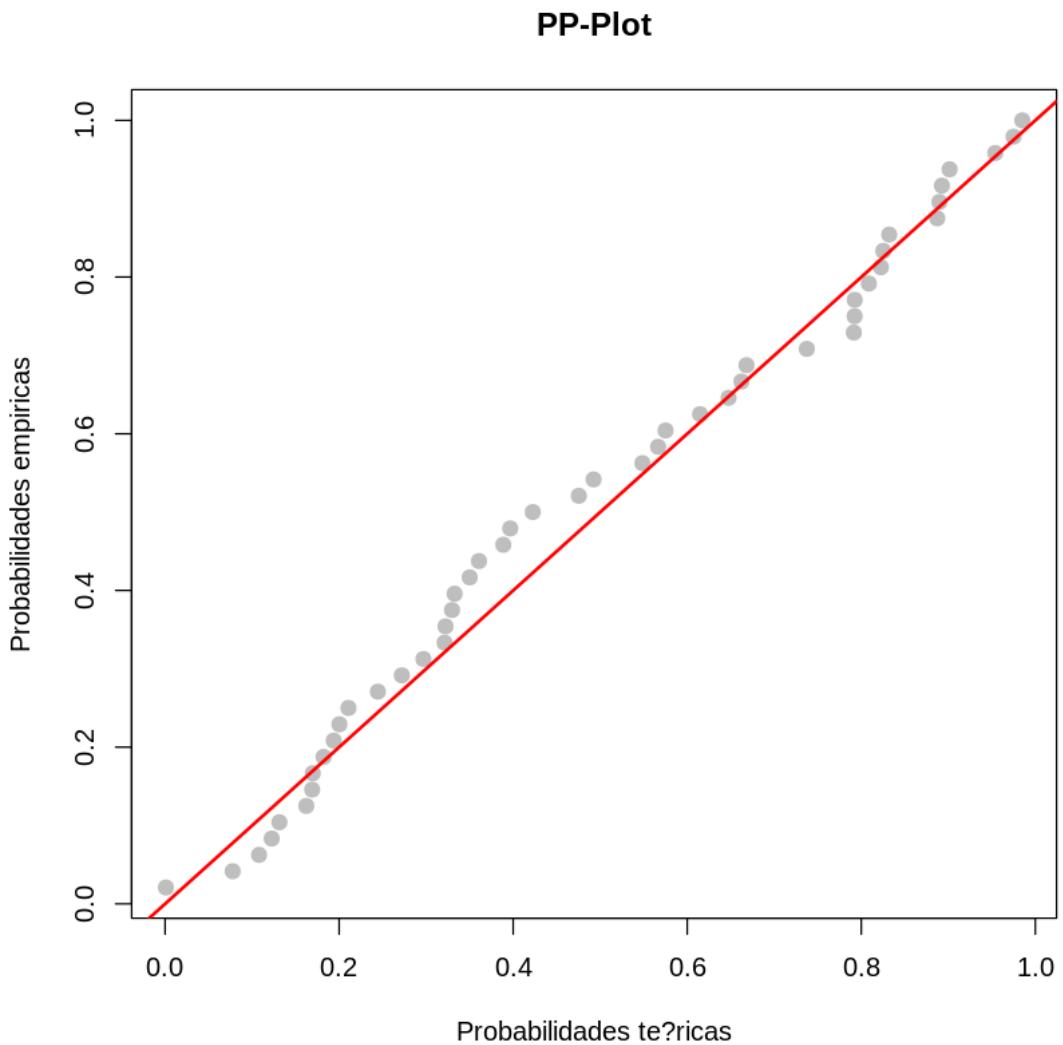
```
[54]: QQplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main = "QQ-Plot", xlab = "Cuantiles teóricos", lcol = "red", lwd = 2)
```

### QQ-Plot



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

```
[55]: PPplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray", main = "PP-Plot", xlab = "Probabilidades teóricas", lcol = "red", lwd = 2)
```

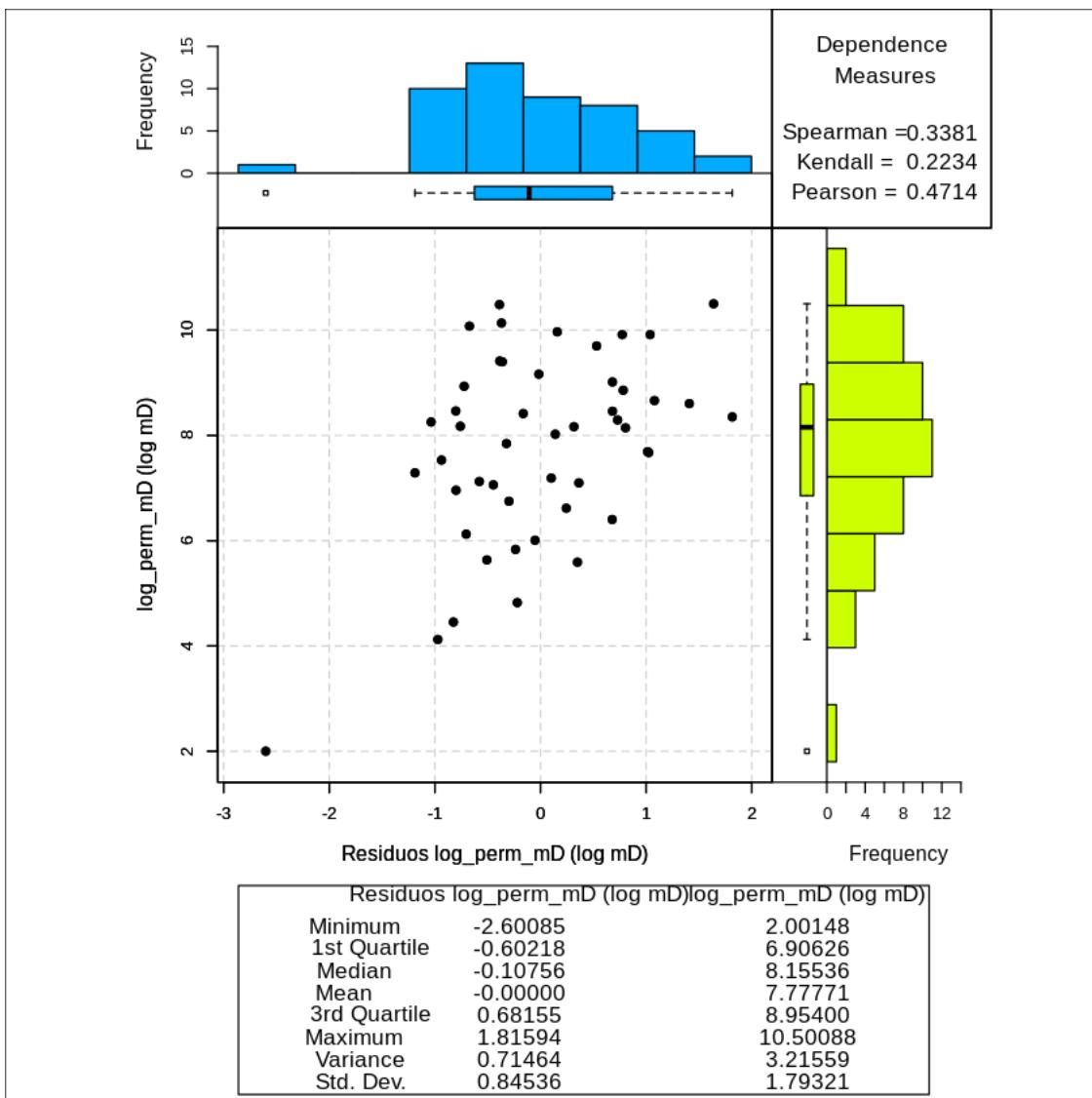


### 3.3.3 Análisis bivariado: Y vs Y residual.

```
[56]: X<-Y_Residual      # Y_Residual es la variable independiente
       Y<-perm_mD_Log    # perm_mD_Log es la variable dependiente
```

Ahora Analizaremos la dependencia entre la variable de porosidad (perm\_mD) con transformación logarítmica y sus residuos. Se hace el grafico de dispersión.

```
[57]: 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 log_perm_mD (log mD)",
                  YLAB = "log_perm_mD (log mD)")
```



El resultado de las medidas de dependencia muestra que sus valores son considerables. 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.

#### 4 Análisis variográfico.

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

## 4.1 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 los siguientes parámetros:

- Vector de las coordenadas (XCoord,YCoord)
- la variable que usaremos que en este caso es la variable de la permeabilidad con transformada logarítmica (perm\_mD\_Log)
- Número de intervalos para el histograma (n\_bins)
- Leyendas para el eje X, eje Y, histograma y título de la imagen

Cuando ejecutamos esta función obtenemos la siguiente imagen:

```
[58]: root_dir<-getwd()

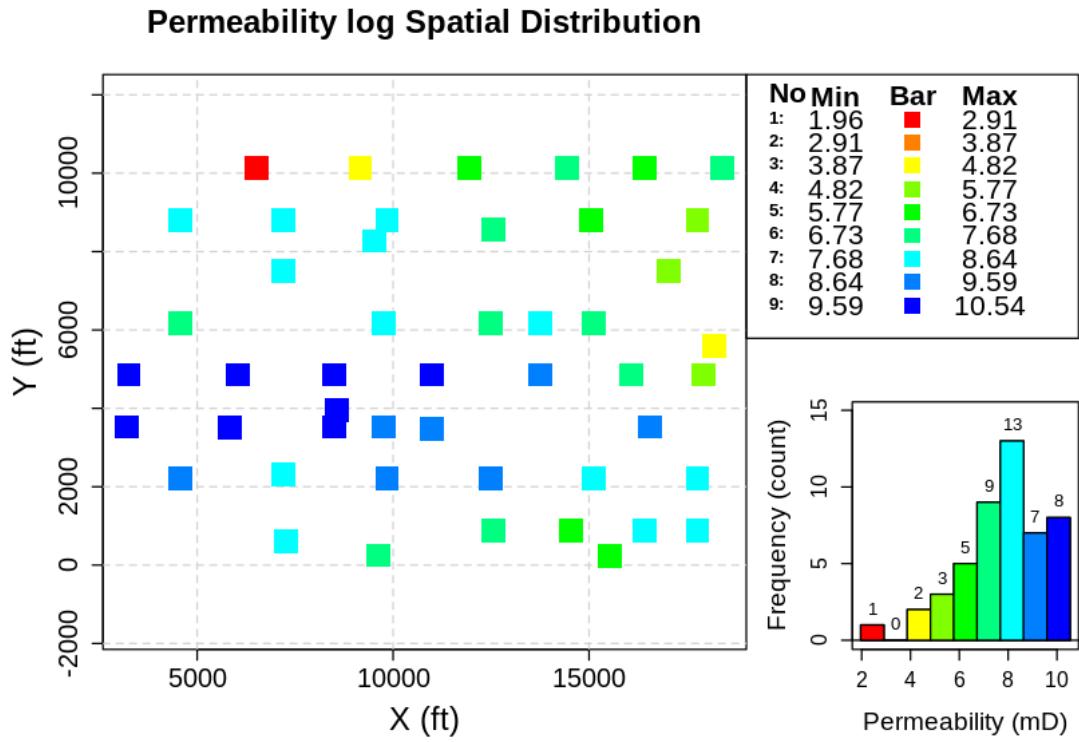
# Creates a folder to store results for a case
# dir.create(paste(getwd(),"/Results/Burb", sep=""))
#
# result_dir<-paste(root_dir,"/Results/Burb",sep="")

# 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 = "")):  
“'/home/danielvr/Dropbox/Semestre 2023-1/ejemplo GAERM/ejemplo  
GAERM/Results/Burb/AV' already exists”

```
[59]: DEspacial(XCoord, YCoord, perm_mD_Log,n_bins=9,
              'X (ft)', 'Y (ft)', 'Permeability (mD)', 'Permeability log Spatial  
↪Distribution')
```



#### 4.1.1 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 regresión no es paralela entonces la variable podría tener algún grado de tendencia.

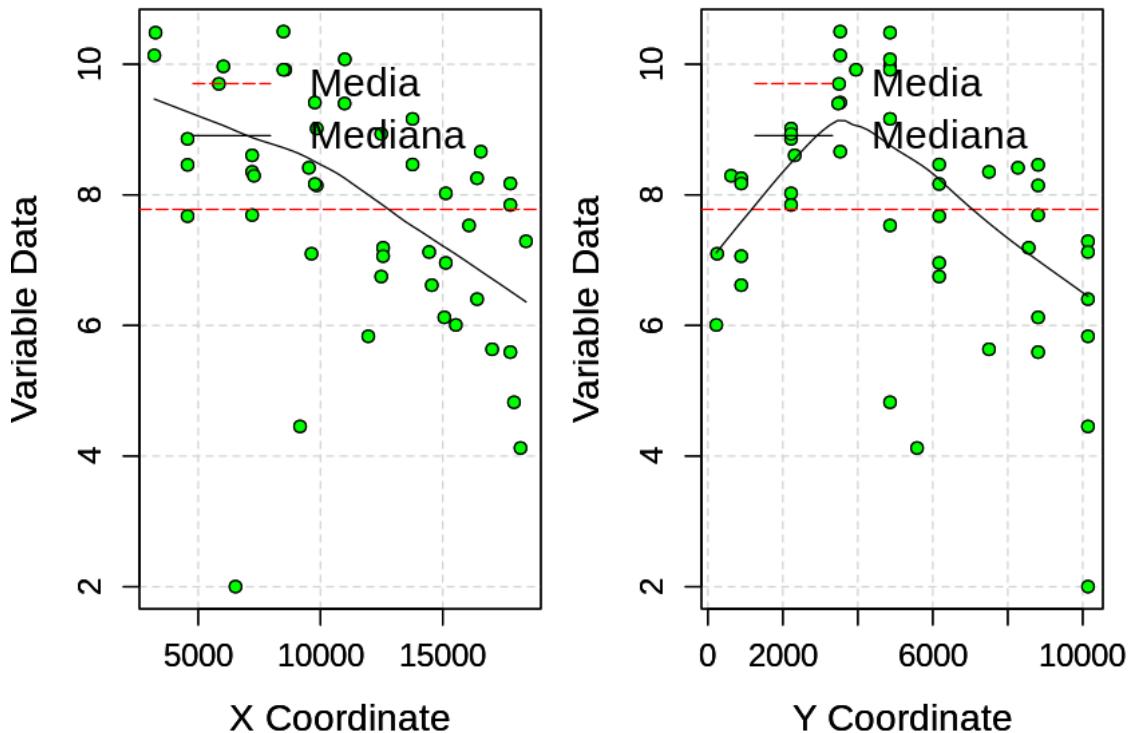
Cabe aclarar que esta prueba solo nos da indicios de la posible tendencia, mas no es determinante. El variograma es la prueba fuerte que nos indica si la variable tiene tendencia.

Para el análisis de regresión de la mediana usamos la función “GDirecciones”, esta requiere de los siguientes parámetros:

- Vector de las coordenadas (XCoord, YCoord)
- La variable a analizar, en este caso perm\_mD\_Log.

[60]: `GDirecciones(XCoord, YCoord, perm_mD_Log)`

## Median Regression Analysis in X and Y directions



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.

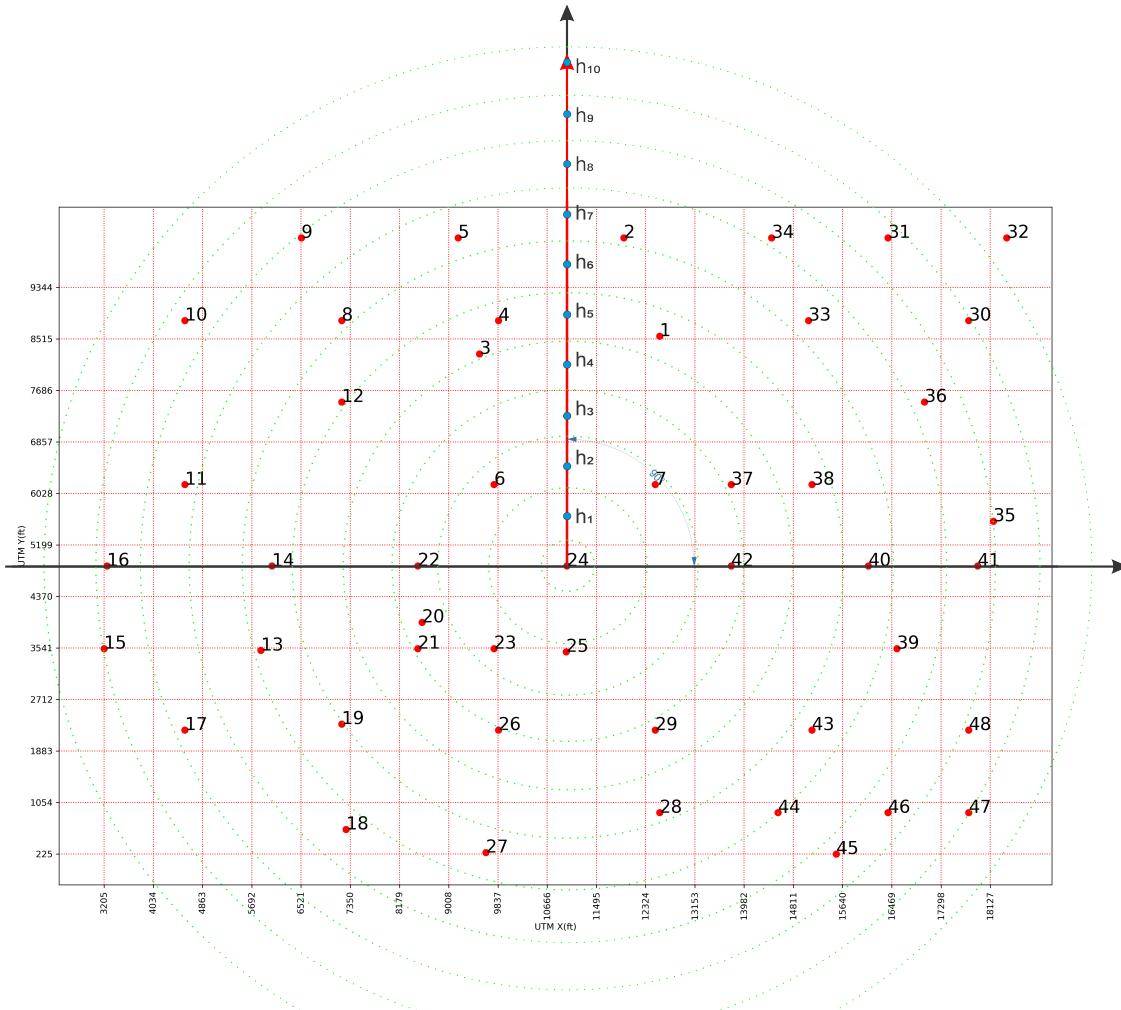
```
[61]: X_rng<-XCoord_Stat[8,2]
Y_rng<-YCoord_Stat[8,2]
N_lags<-10
lag_value <- sqrt(X_rng*X_rng+Y_rng*Y_rng)/(2*N_lags)
DistMin<-min(dist(Data_File_Burb[,1:2])) # Minimum distance in data
DistMax<-max(dist(Data_File_Burb[,1:2])) # Maximum distance in data
lag_value<-max((DistMax/2)/N_lags, DistMin)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

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  $(\text{DistMax}/2)/\text{N\_lags}$  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  $(\text{DistMax}/2)/\text{N\_lags}$  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  $0^\circ$  y ángulo de tolerancia de  $90^\circ$ .

Ya que tenemos estos valores podemos estimar el variograma experimental adireccional, el cual tiene por dirección  $0^\circ$  y ángulo de tolerancia de  $90^\circ$ . De forma gráfica el cálculo del variograma se puede ver de la siguiente forma:



Para estimar el variograma adireccional se usa la función “Variograma”, Esta función necesita:

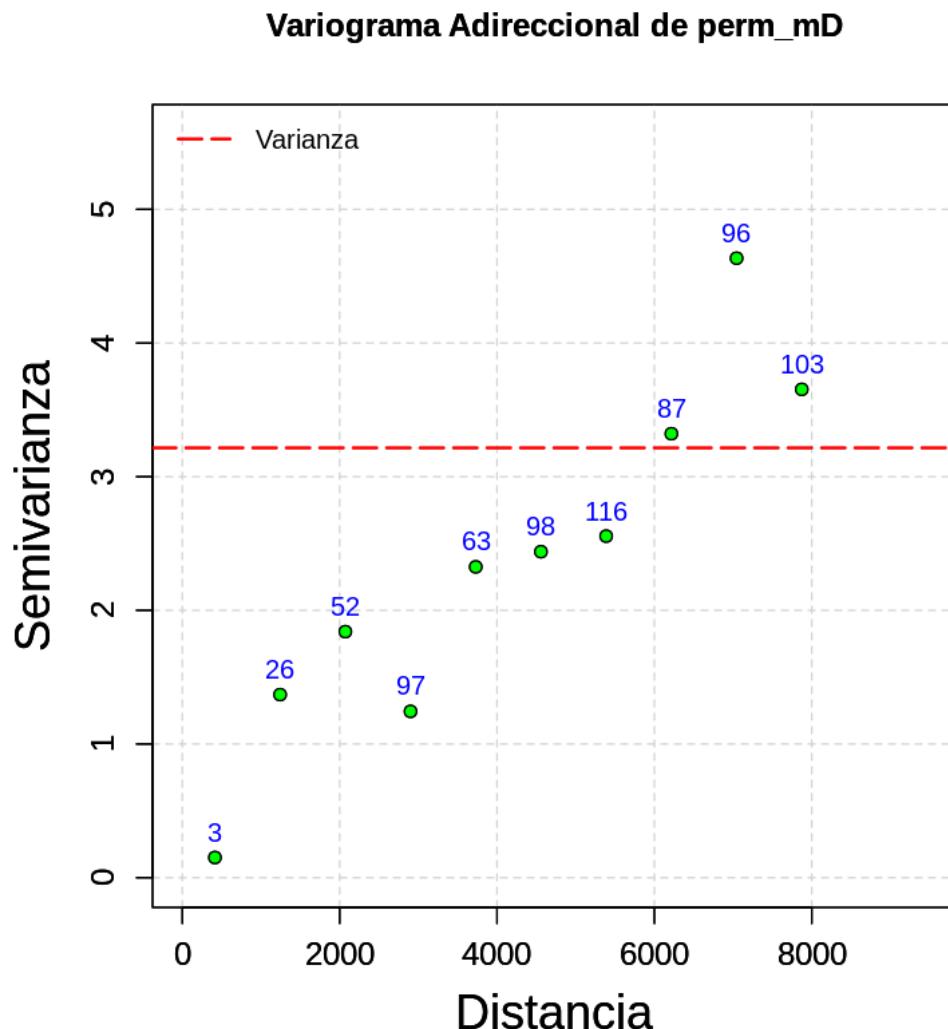
- Coordenadas (XCoord, YCoord)
- Variable (perm\_mD\_Log)
- Una dirección (Direccion) y su tolerancia angular (Tol), como en este caso es variograma adireccional la dirección es de  $0^{\circ}$  y su tolerancia es de  $90^{\circ}$ .
- Número de intervalos (N\_lags)
- Valor del intervalo (lag\_value)
- 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:

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

```
[62]: perm_mD_Log_VarioEstimation<-Variogramma(XCoord, YCoord,
                                                perm_mD_Log, 0, 90, N_lags, lag_value, ↵
                                                ↵1, "Variogramma Adireccional de perm_mD")
```

variog: computing omnidirectional variogram



```
[63]: perm_mD_Log_VarioEstimation
write.csv(perm_mD_Log_VarioEstimation, file = paste(av_dir,"/
→perm_mD_Vario_Adireccional.csv",sep=""))
```

	Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
A data.frame: 10 × 3	3	414.3761	0.1509602
	26	1243.1283	1.3694949
	52	2071.8806	1.8407988
	97	2900.6328	1.2442231
	63	3729.3850	2.3253647
	98	4558.1372	2.4387534
	116	5386.8895	2.5553160
	87	6215.6417	3.3214789
	96	7044.3939	4.6345450
	103	7873.1461	3.6533460

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 polinomial se puede hacer usando la función “Trend”, la cual necesita los siguientes parámetros:

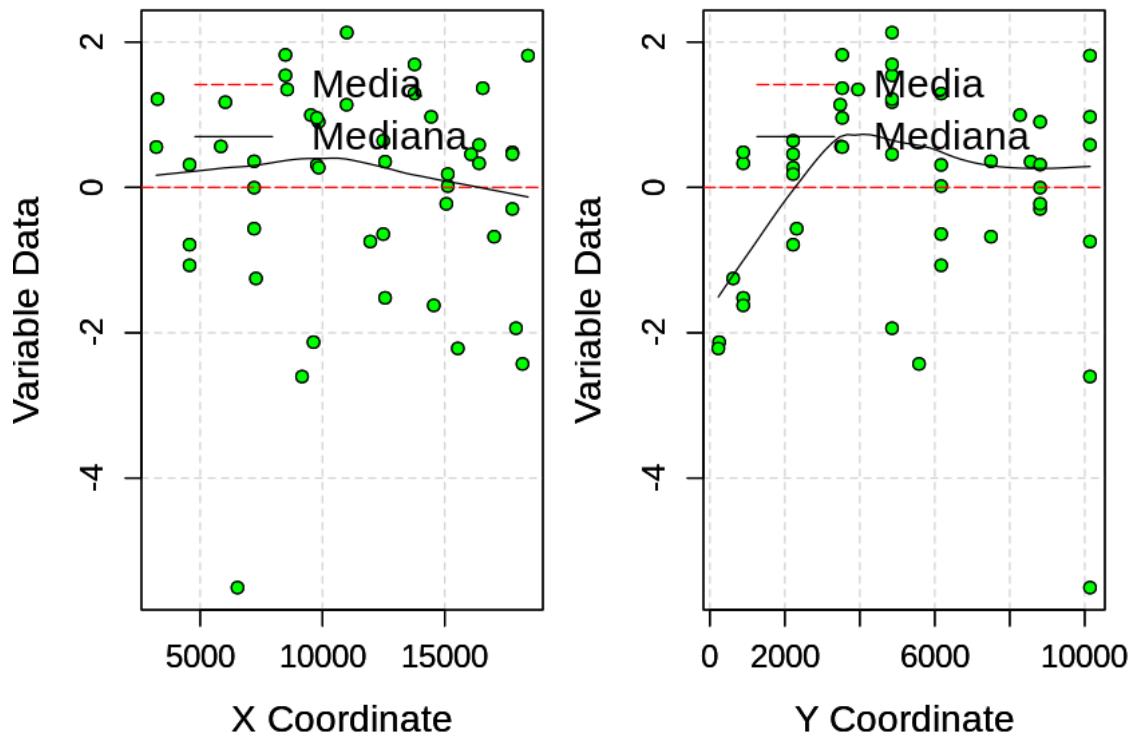
- Vector de las coordenadas (XCoord, YCoord)
- La variable (perm\_mD\_Log)
- El grado del polinomio (pol\_degree).

```
[64]: pol_degree=1
perm_mD_Log_Detrended_1<-Trend(XCoord, YCoord,
                                     perm_mD_Log, pol_degree)
```

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

```
[65]: GDirecciones(perm_mD_Log_Detrended_1[,1], perm_mD_Log_Detrended_1[,2],  
                   perm_mD_Log_Detrended_1[,3])
```

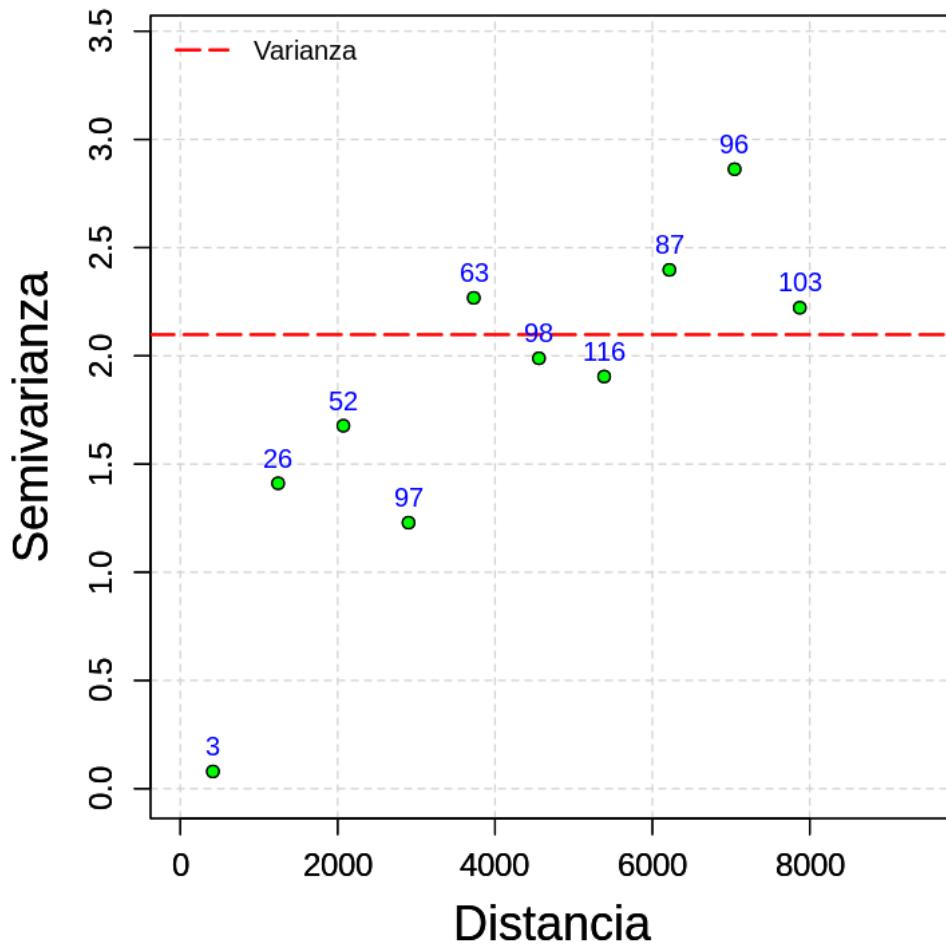
## Median Regression Analysis in X and Y directions



```
[66]: perm_mD_Log_Detrended_1_VarioEstimation<-Variograma(perm_mD_Log_Detrended_1[,1],  
           ↪perm_mD_Log_Detrended_1[,2],  
           ↪perm_mD_Log_Detrended_1[,3], 0, 90, N_lags, lag_value, 1,  
           "Variograma Adireccional de"  
           ↪perm_mD_Log_Residuos_1")
```

variog: computing omnidirectional variogram

### Variograma Adireccional de perm\_mD\_Log Residuos 1

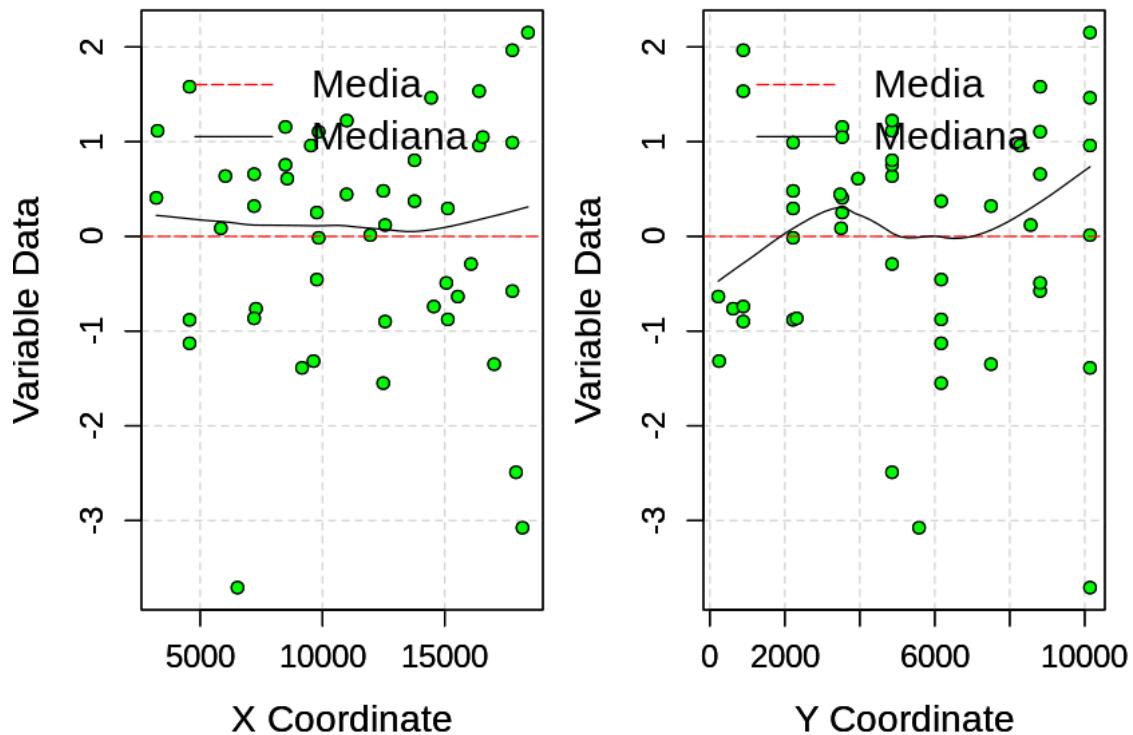


La transformación de segundo orden se hace de la siguiente forma.

```
[67]: pol_degree=2  
perm_mD_Log_Detrended_2<-Trend(XCoord, YCoord,  
perm_mD_Log, pol_degree)
```

```
[68]: GDirecciones(perm_mD_Log_Detrended_2[,1], perm_mD_Log_Detrended_2[,2],  
perm_mD_Log_Detrended_2[,3])
```

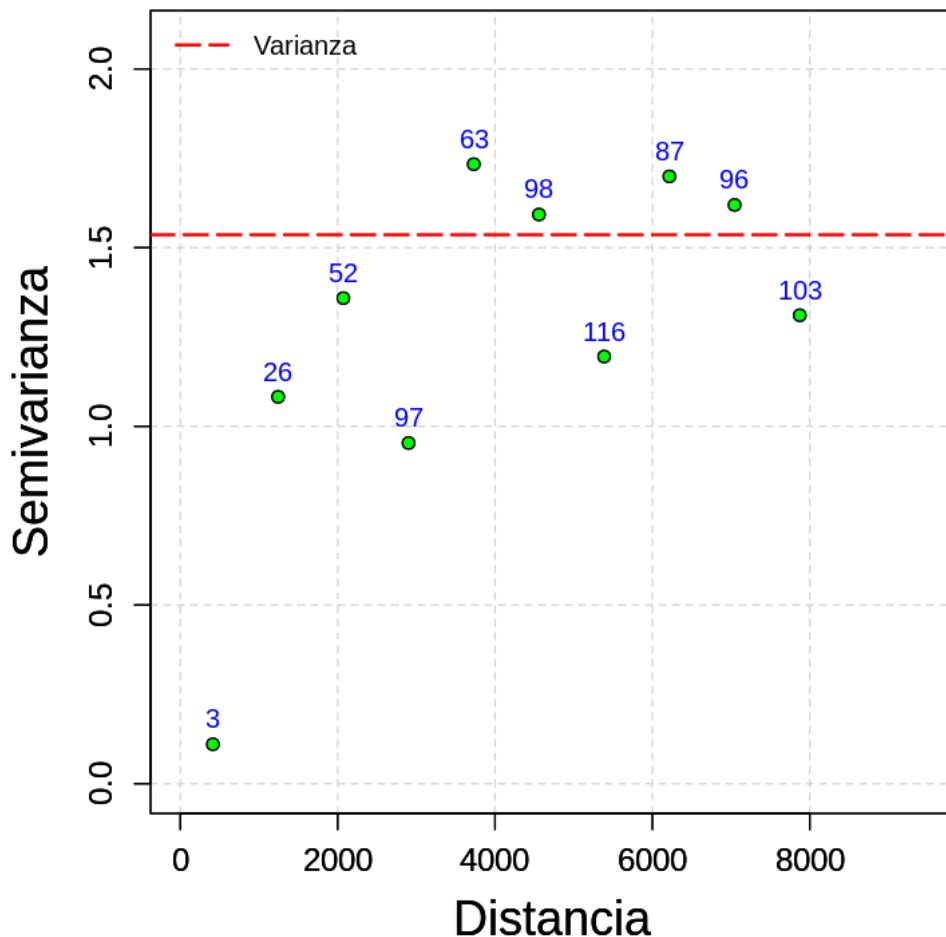
## Median Regression Analysis in X and Y directions



```
[69]: perm_mD_Log_Detrended_2_VarioEstimation<-Variograma(perm_mD_Log_Detrended_2[,1],  
           ↪perm_mD_Log_Detrended_2[,2],  
           ↪perm_mD_Log_Detrended_2[,3], 0, 90, N_lags, lag_value, 1,  
           "Variograma Adireccional de  
           ↪perm_mD_Log_Residuos_2")
```

variog: computing omnidirectional variogram

### Variograma Adireccional de perm\_mD\_Log Residuos 2



#### 4.1.2 Modelado variográfico unidimensional de la variable perm\_mD\_Log

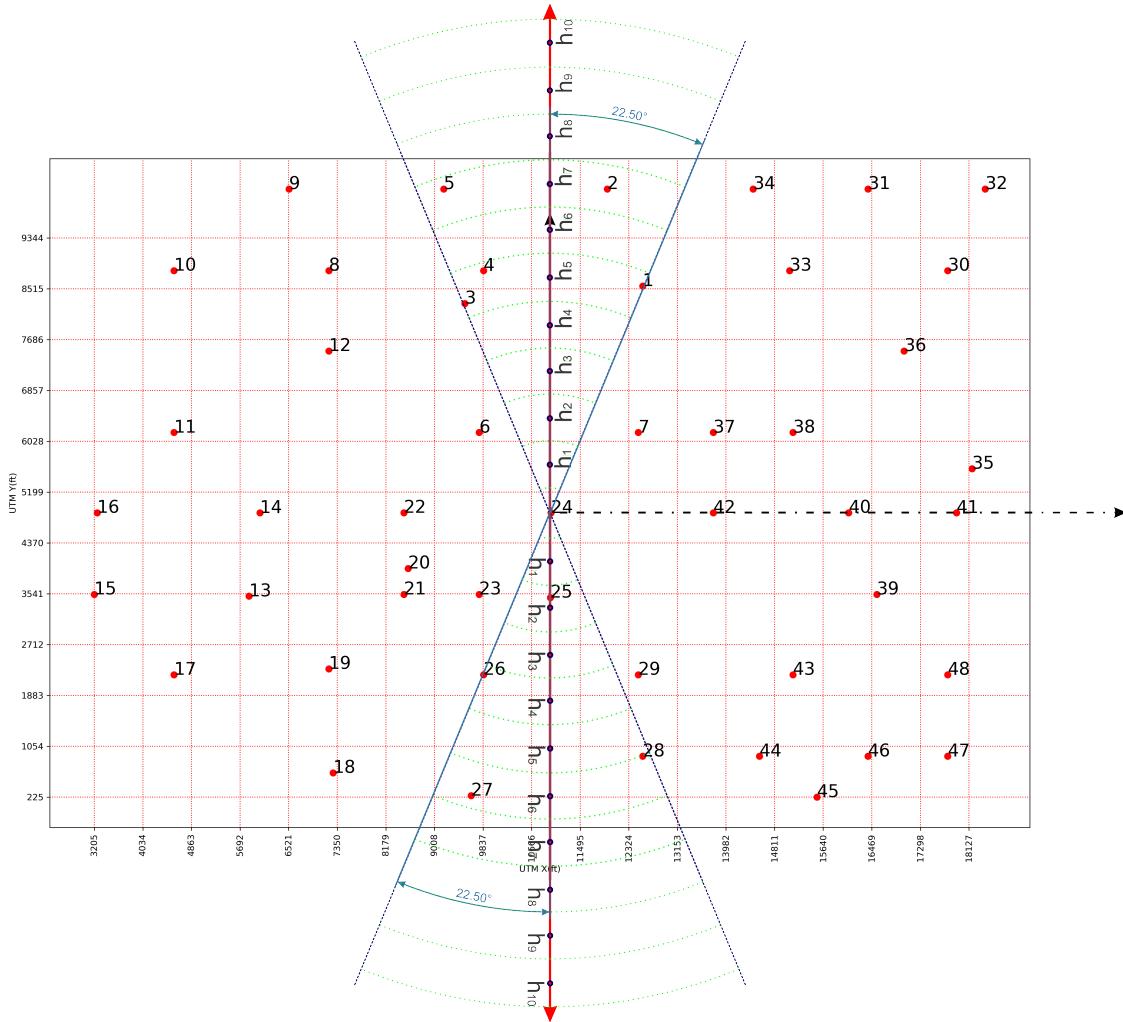
Después del análisis de tendencia, podemos empezar con el modelado del variograma. Por lo que empezaremos con el cálculo de los variogramas direccionales, esto para determinar si hay anisotropía.

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

- Coordenadas (XCoord, YCoord)
- Variable (perm\_mD\_Log)
- Dirección del vector, los cuales son:  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  y  $135^\circ$
- Valor de la tolerancia angular, la cual es de  $22.5^\circ$
- Número de intervalos (N\_lags)
- Valor del intervalo (lag\_value)
- Número de pares mínimo, por default es 1

- Título del gráfico.

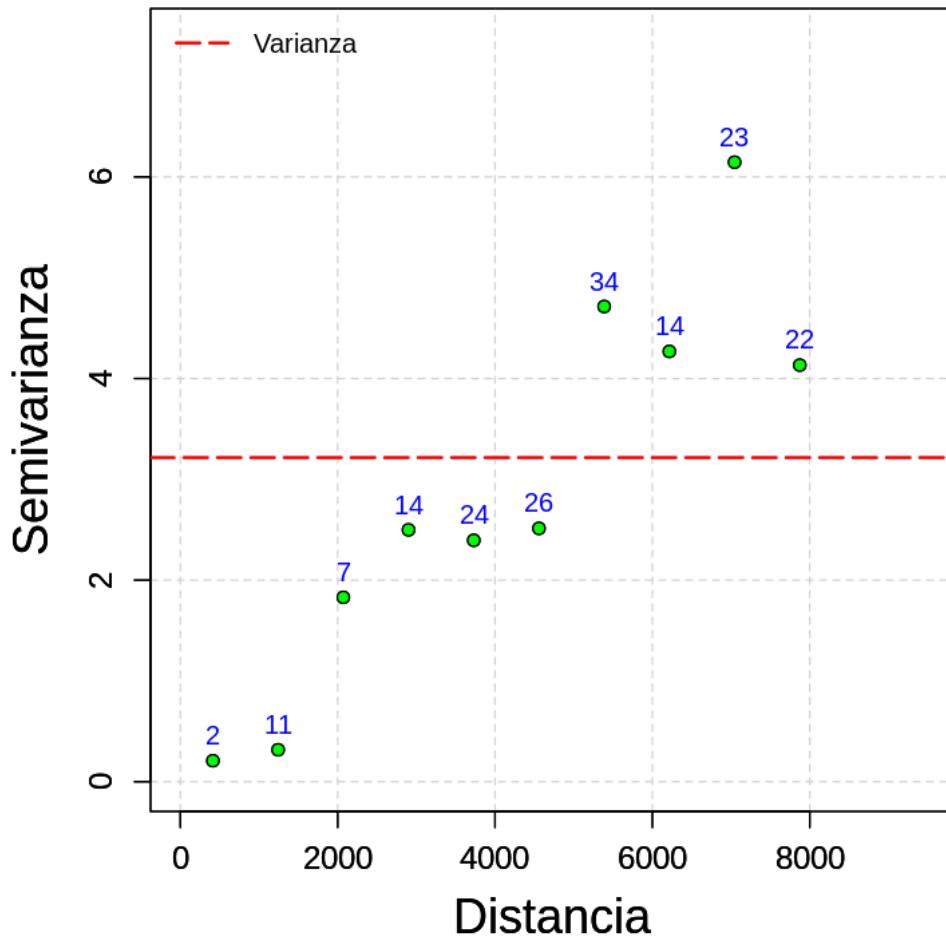
Dado estos parámetros, podemos poner como ejemplo la siguiente grafica que ilustra la forma de cálculo del variograma con los parámetros antes mencionados, para la dirección de vector  $0^{\circ}$  es:



```
[70]: perm_mD_Log_VarioEstimation_dir0<-Variograma(XCoord, YCoord,
                                                 perm_mD_Log, 0, 22.5, N_lags,
                                                 lag_value, 1,
                                                 "Variograma de perm_mD_Log en la"
                                                 "direccion de 0°")
```

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

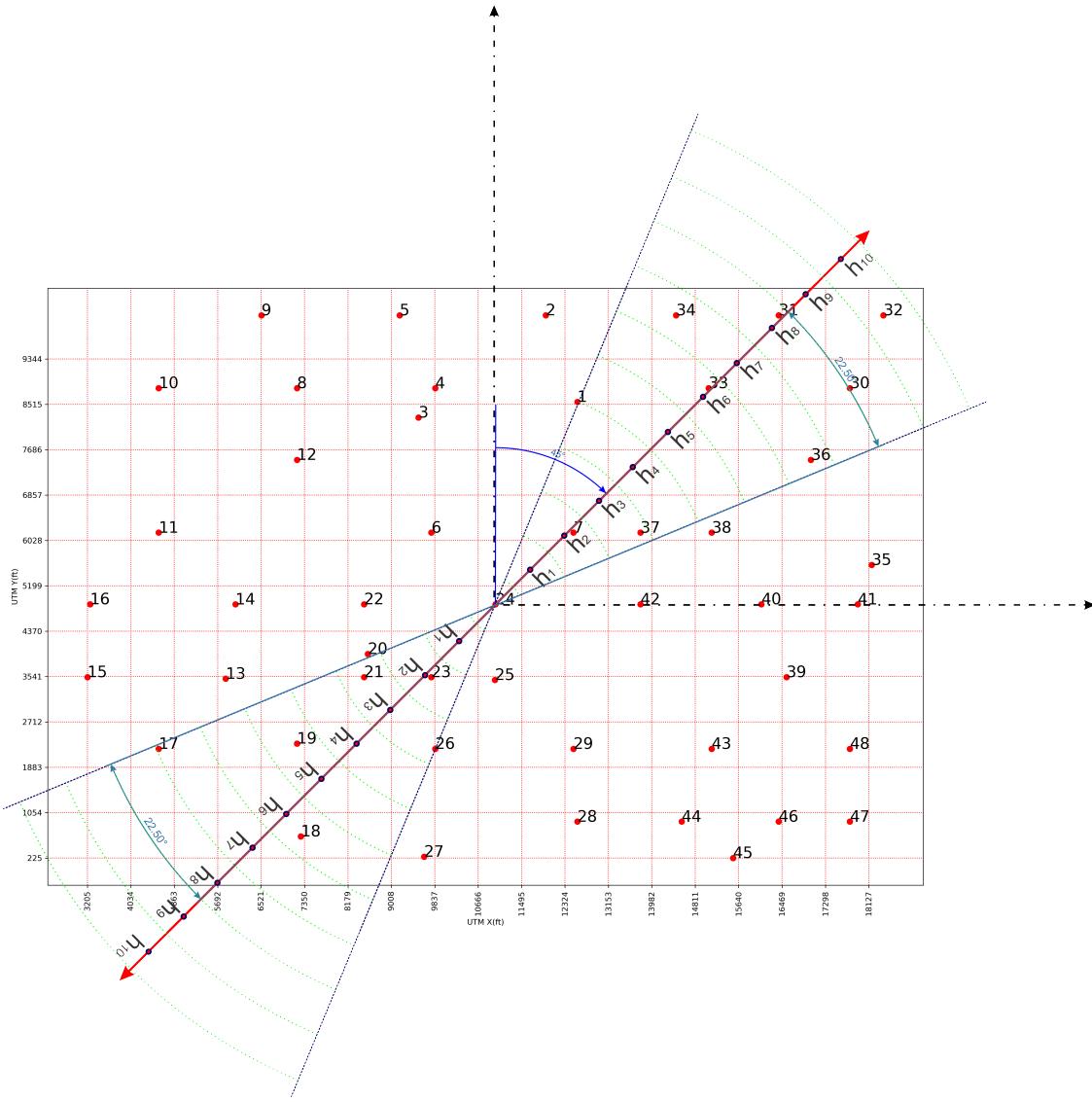
### Variograma de perm\_mD\_Log en la direccion de 0°



```
[71]: write.csv(perm_mD_Log_VarioEstimation_dir0, file = paste(av_dir,"/perm_mD_Vario_direccion0.csv",sep=""))
```

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

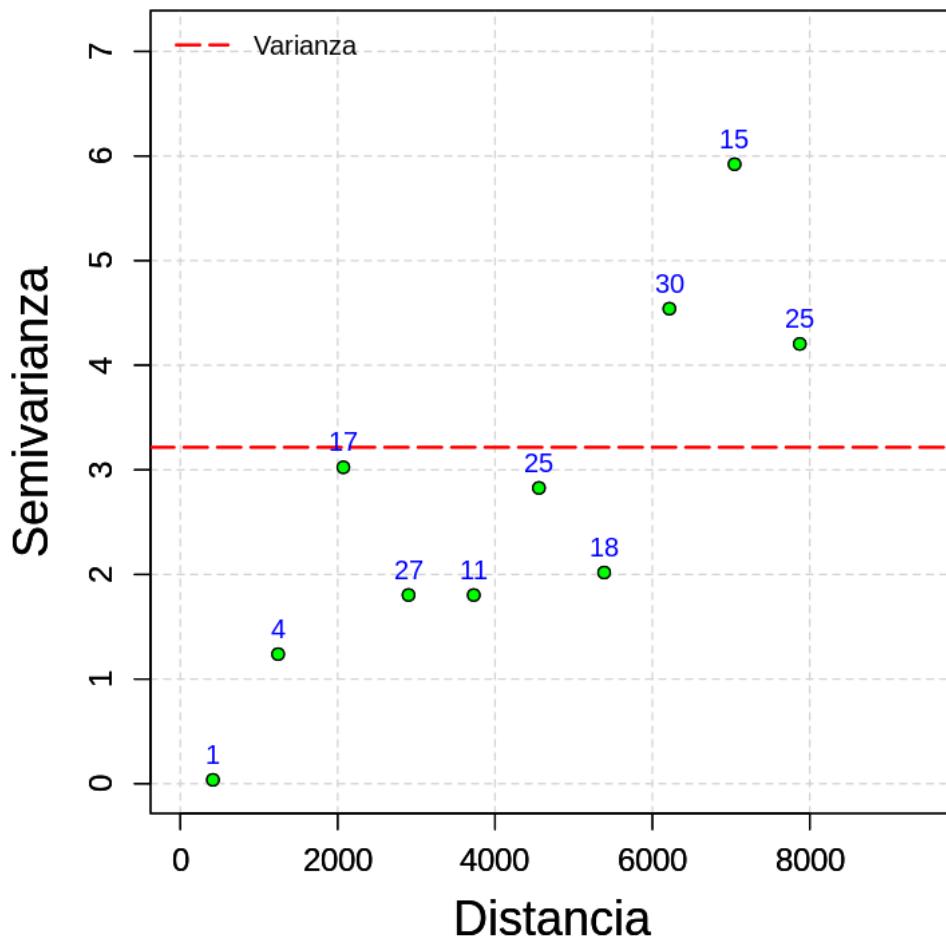
Calculamos el variograma direccional  $45^\circ$



```
[72]: perm_mD_Log_VarioEstimation_dir45<-Variogramma(XCoord, YCoord,
perm_mD_Log, 45, 22.5, N_lags,
lag_value, 1,
"Variograma de perm_mD_Log en la dirección de 45º")
```

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

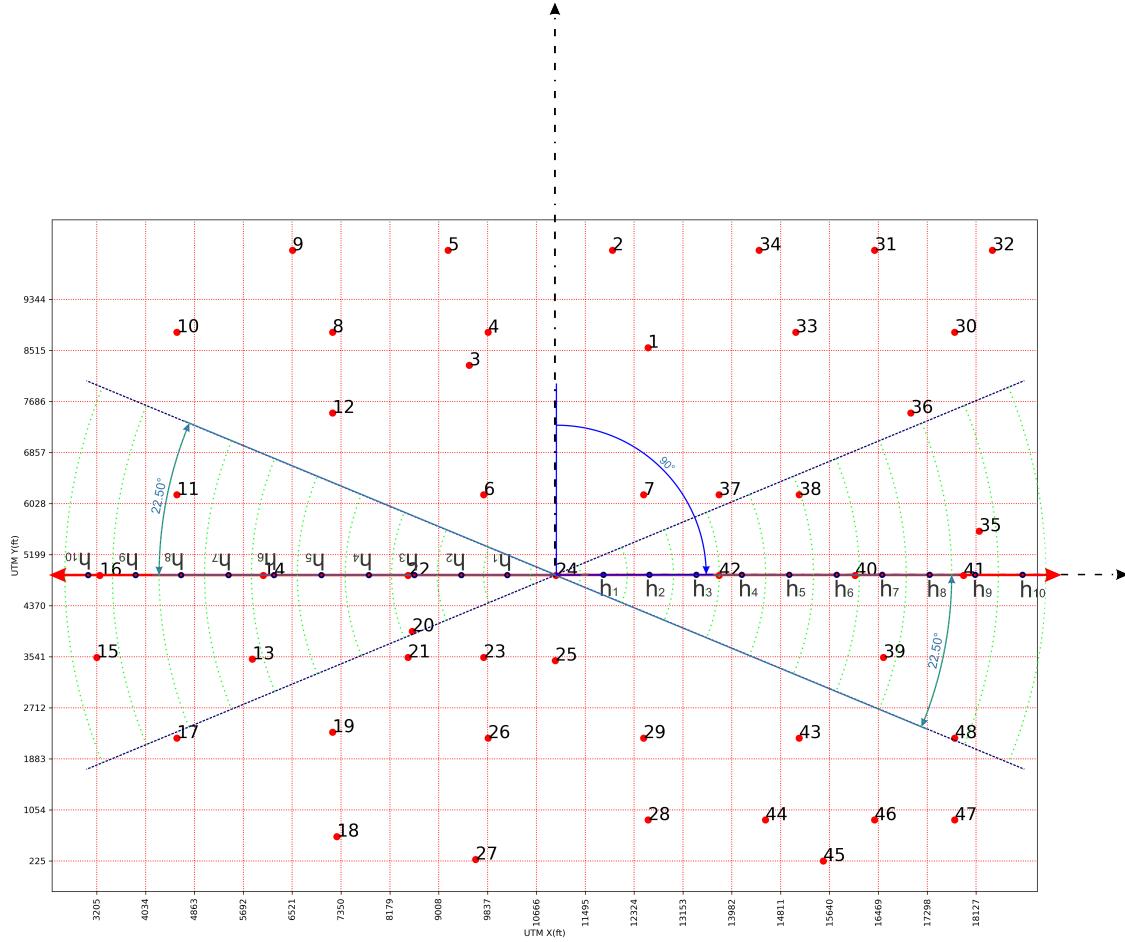
### Variograma de perm\_mD\_Log en la dirección de 45°



```
[73]: write.csv(perm_mD_Log_VarioEstimation_dir45, file = paste(av_dir,"/perm_mD_Vario_direccion45.csv",sep=""))
```

Con el variograma direccional a 45° notamos el mismo problema detectado en el variograma direccional de 0°, 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.

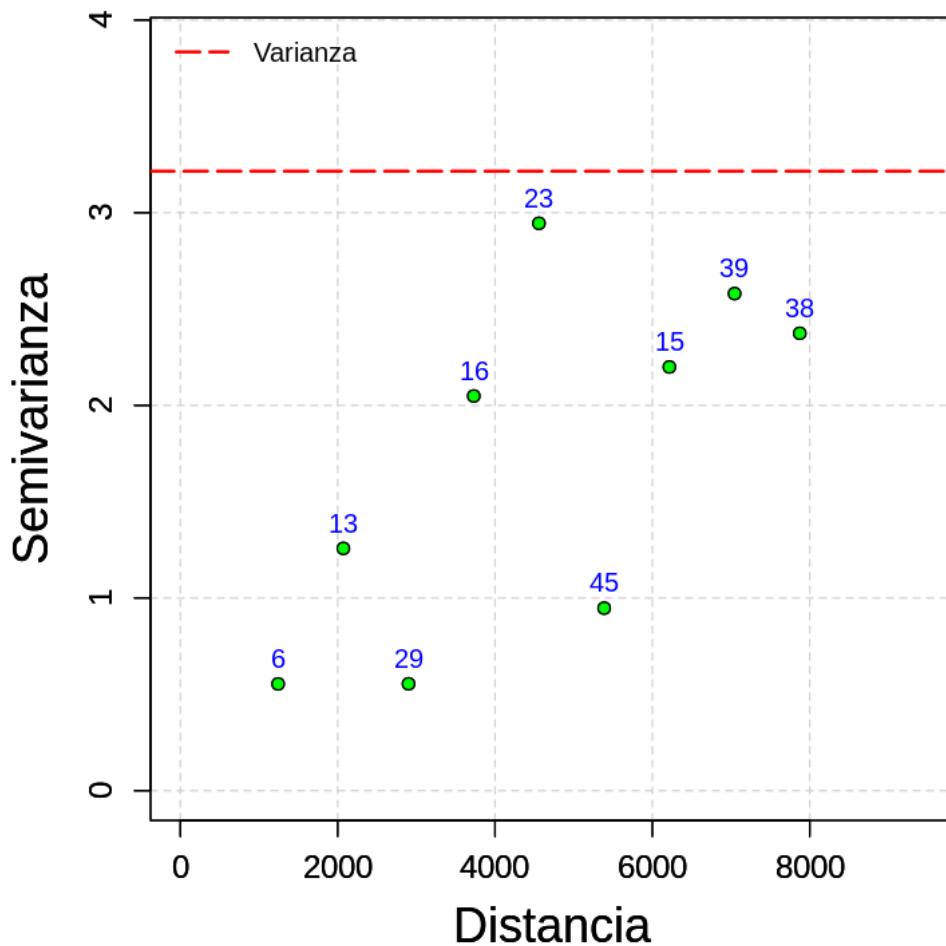
Probamos con el variograma direccional de 90°



```
[74]: perm_mD_Log_VarioEstimation_dir90<-Variograma(XCoord, YCoord,
                                                    perm_mD_Log, 90, 22.5, N_lags,
                                                    lag_value, 1,
                                                    "Variograma de perm_mD_Log en la"
                                                    "direccion de 90°")
```

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

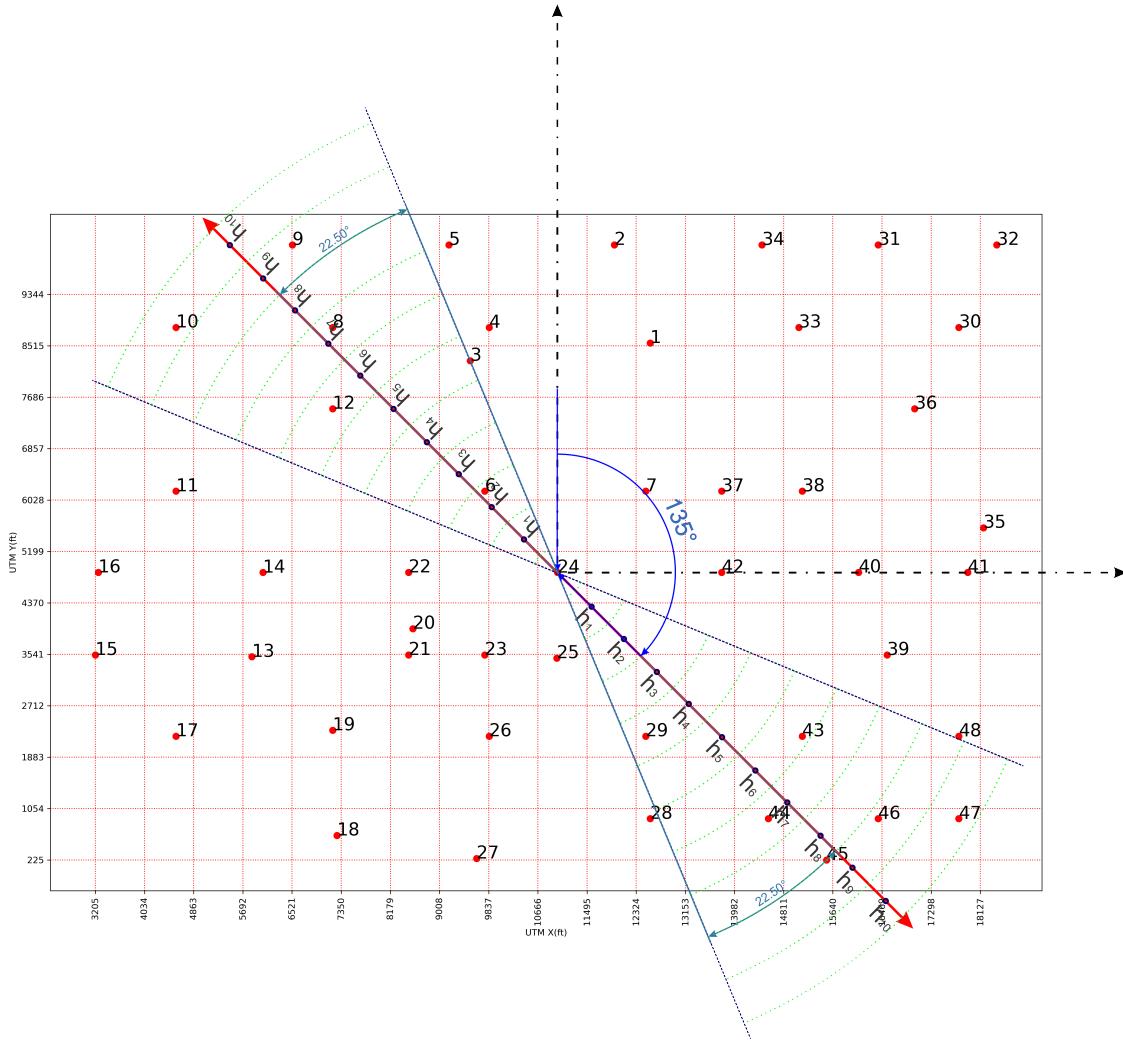
### Variograma de perm\_mD\_Log en la dirección de 90°



```
[75]: write.csv(perm_mD_Log_VarioEstimation_dir90, file = paste(av_dir,"/perm_mD_Vario_direccion90.csv",sep=""))
```

Con el variograma direccional de 90° 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.

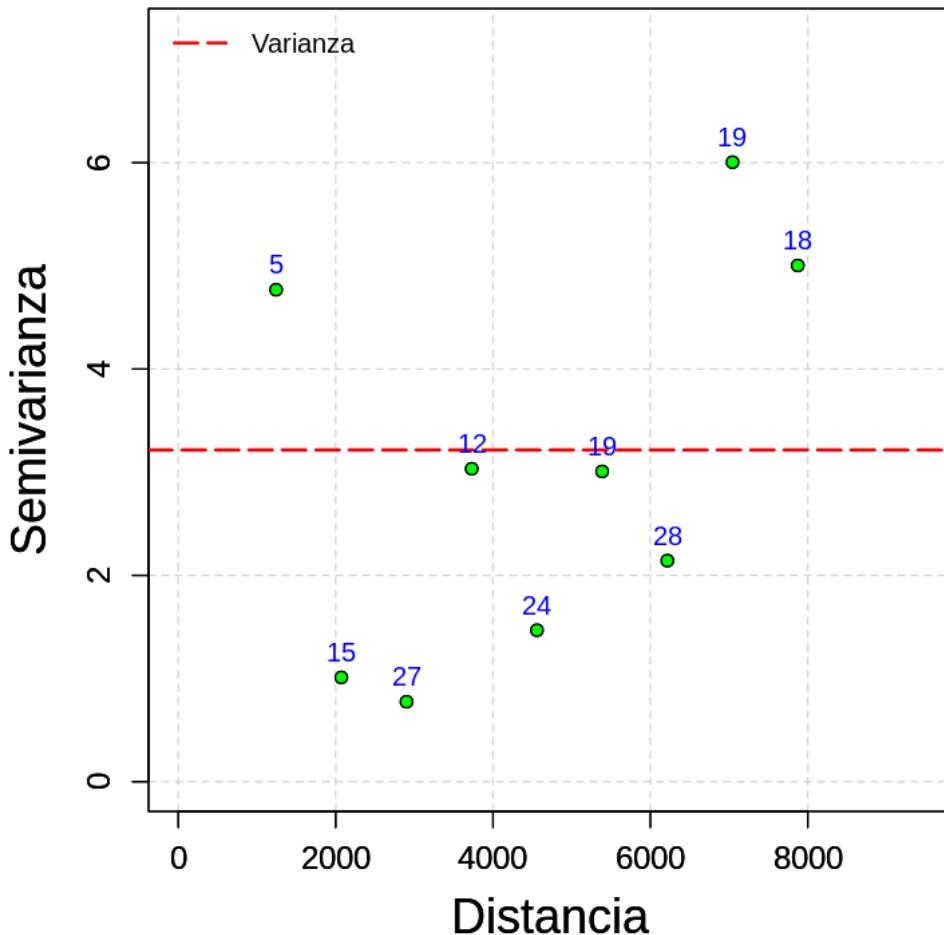
Seguimos con el variograma direccional de 135°.



```
[76]: perm_mD_Log_VarioEstimation_dir135<-Variogramma(XCoord, YCoord,
                                                       perm_mD_Log, 135, 22.5, N_lags,
                                                       lag_value, 1,
                                                       "Variogramma de perm_mD_Log en la"
                                                       →direccion de 135°")
```

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

### Variograma de perm\_mD\_Log en la dirección de 135°



```
[77]: write.csv(perm_mD_Log_VarioEstimation_dir135, file = paste(av_dir,"/
→perm_mD_Vario_direccion135.csv",sep=""))
```

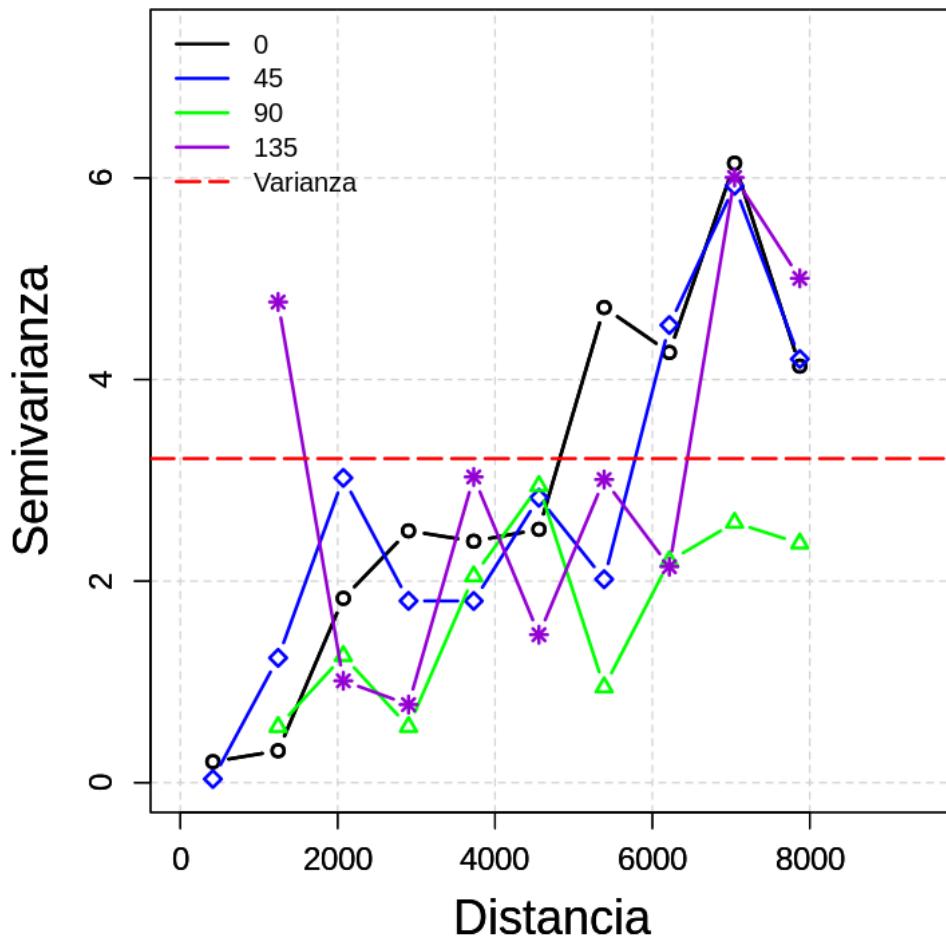
El variograma direccional a 135° no tiene intervalos con un número de pares superior a 30, de todos los casos direccionales este es el peor y al igual que los demás, no se puede juzgar si existe anisotropía.

Para visualizar los cuatro variogramas direccionales en una sola imagen usamos la función “Variograma4D”. Esta función necesita \* Vector de las coordenadas (XCoord, YCoord) \* La variable (perm\_mD\_Log) \* 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) \* Número mínimo de pares, el cual por default es uno. \* Titulo de la imagen

```
[78]: perm_mD_Log_VarioEstimation4D<-Variograma4D(XCoord, YCoord,
                                                    perm_mD_Log, 0, 45, 90, 135, 22.5,
                                                    N_lags, lag_value, 1,
                                                    "Variogramas Direccionales de",
                                                    perm_mD_Log")
```

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)

### Variogramas Direccionales de perm\_mD\_Log



Si colocan en la consola “perm\_mD\_VarioEstimation4D”, pueden obtener los pares de la estimación en los cuatro variogramas direccionales.

```
[79]: perm_mD_Log_VarioEstimation4D  
      write.csv(perm_mD_Log_VarioEstimation4D, file = paste(av_dir,"/  
      ↪perm_mD_VarioEstimation4D.csv",sep=""))
```

\$Zero	A matrix: 10 × 3 of type dbl	2	414.3761	0.2081300
		11	1243.1283	0.3167509
		7	2071.8806	1.8291463
		14	2900.6328	2.4988258
		24	3729.3850	2.3952091
		26	4558.1372	2.5133831
		34	5386.8895	4.7147252
		14	6215.6417	4.2686884
		23	7044.3939	6.1462901
		22	7873.1461	4.1329232
\$FortyFive	A matrix: 10 × 3 of type dbl	1	414.3761	0.03662079
		4	1243.1283	1.23830150
		17	2071.8806	3.02390975
		27	2900.6328	1.80358029
		11	3729.3850	1.80297727
		25	4558.1372	2.82690635
		18	5386.8895	2.01827413
		30	6215.6417	4.54043961
		15	7044.3939	5.92152265
		25	7873.1461	4.20327918
\$Ninety	A matrix: 9 × 3 of type dbl	6	1243.128	0.5542313
		13	2071.881	1.2577436
		29	2900.633	0.5547653
		16	3729.385	2.0486393
		23	4558.137	2.9451322
		45	5386.889	0.9475378
		15	6215.642	2.1994109
		39	7044.394	2.5803364
		38	7873.146	2.3741641
\$OneThertyFive	A matrix: 9 × 3 of type dbl	5	1243.128	4.768803
		15	2071.881	1.010692
		27	2900.633	0.774860
		12	3729.385	3.033498
		24	4558.137	1.468299
		19	5386.889	3.007782
		28	6215.642	2.142953
		19	7044.394	6.005036
		18	7873.146	5.003895

```

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

1. write.csv(perm_mD_Log_VarioEstimation4D, file = paste(av_dir,
  .      "/perm_mD_VarioEstimation4D.csv", sep = ""))
2. eval.parent(Call)
3. eval(expr, p)
4. eval(expr, p)
5. utils:::write.table(perm_mD_Log_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 used",
  .                     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[duplicated(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 =
stringsAsFactors)
  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 (mirln && nrows[i] > OL) {
        .
        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] > OL && (nr%%nrows[i] == OL)) {
            .
            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 > OL]))
    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, 0.208129959792529, 0.316750946755461, 1.829146303687,

```

```

. 2.49882577825612, 2.39520909623014, 2.51338313699251, 4.71472516140141,
. 4.26868839085334, 6.1462900584639, 4.13292317278246), 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, 0.0366207900826558, 1.23830149543311,
. 3.02390974886301, 1.80358028512656, 1.8029772736876, 2.82690635107213,
. 2.01827413110267, 4.54043961140359, 5.9215226500334, 4.20327917867764
. ), 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, 0.554231342418576, 1.25774355931735,
. 0.554765315815191, 2.04863930313629, 2.94513223492395, 0.94753780037439,
. 2.19941093092259, 2.58033644821637, 2.37416410699823), 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, 4.76880274797831, 1.01069202110633,
. 0.774860025286425, 3.03349837681268, 1.46829878566707, 3.00778214794749,
. 2.14295272069518, 6.00503620333989, 5.00389538893742), dim = c(9L,
. 3L)), check.names = FALSE, fix.empty.names = TRUE, stringsAsFactors = FALSE)
11. stop(gettextf("arguments imply differing number of rows: %s",
.     paste(unique(nrows), collapse = ", ")), domain = NA)

```

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. También 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 los siguientes parámetros:

- Vector de las coordenadas (XCoord, YCoord)
- La variable (perm\_mD\_Log)
- La dirección del vector el cual es de  $0^\circ$
- Su ángulo de tolerancia ( $90^\circ$ )
- Número de intervalos (N\_lags)
- 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:

```
[80]: perm_mD_Log_AllModelVarioFit<-AllModel(XCoord, YCoord,
                                                perm_mD_Log, 0, 90, N_lags, lag_value, 1,
```

"Ajustes del Variograma Adireccional de la  
→perm\_mD\_Log")

```
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

Warning message in variofit(Nuevo, ini = c(modelo_Exp00$cov.pars[1],
modelo_Exp00$cov.pars[2]), :
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Exp00$cov.pars[1],
modelo_Exp00$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Exp00$cov.pars[1],
modelo_Exp00$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

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

Warning message in variofit(Nuevo, ini = c(modelo_Sph00$cov.pars[1],
modelo_Sph00$cov.pars[2]), :
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Sph00$cov.pars[1],
modelo_Sph00$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Sph00$cov.pars[1],
modelo_Sph00$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

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

Warning message in variofit(Nuevo, ini = c(modelo_Exp0$cov.pars[1],
modelo_Exp0$cov.pars[2]), :
```

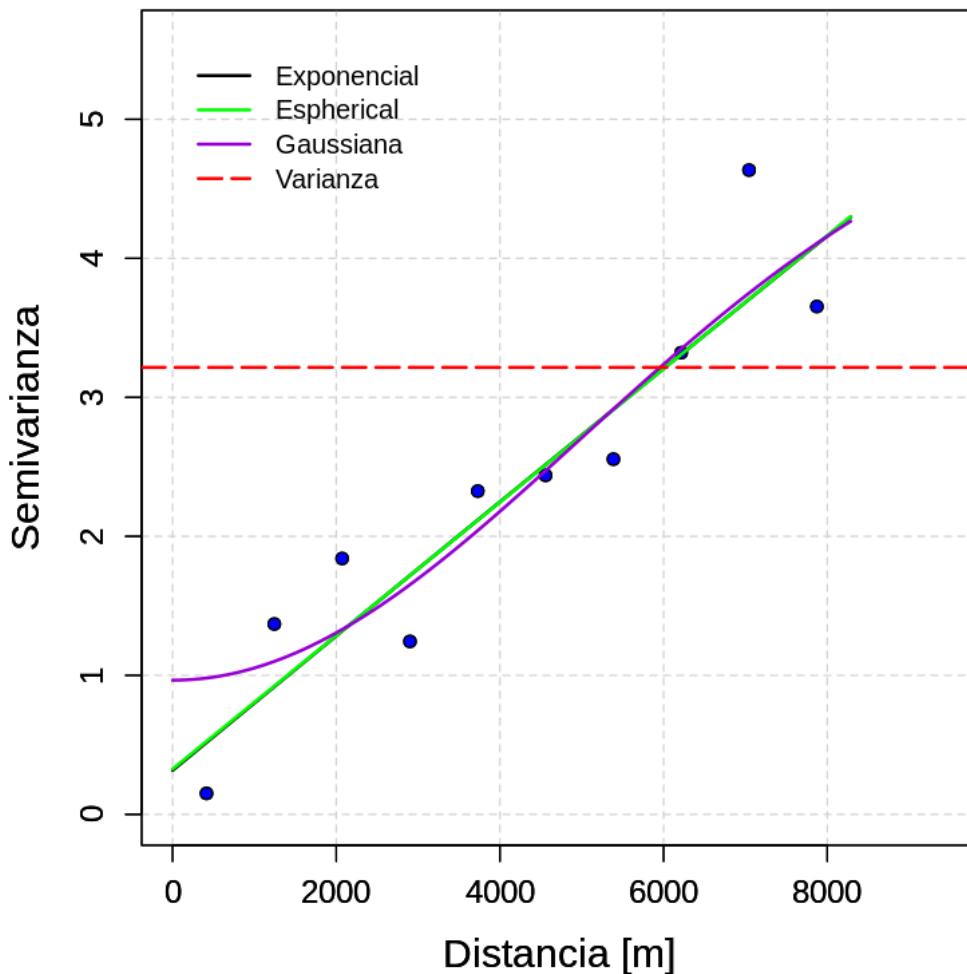
```
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Exp0$cov.pars[1],
modelo_Exp0$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Exp0$cov.pars[1],
modelo_Exp0$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

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

Warning message in variofit(Nuevo, ini = c(modelo_Sph0$cov.pars[1],
modelo_Sph0$cov.pars[2]), :
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Sph0$cov.pars[1],
modelo_Sph0$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Sph0$cov.pars[1],
modelo_Sph0$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

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

### Ajustes del Variograma Adireccional de perm\_mD\_Log



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.

```
[81]: perm_mD_Log_AllModelVarioFit
write.csv(perm_mD_Log_AllModelVarioFit, file = paste(av_dir, "/",
→perm_mD_AllModelVarioFit.csv", sep = ""))
```

		Nugget	Meseta+Nugget	Alcance	SCE
A matrix: 3 × 4 of type dbl	exponential	0.3167553	212.110284	436756.711	2.075355
	spherical	0.3264074	59.598537	185234.804	2.075597
	gaussian	0.9652377	5.370461	7043.999	30124.301121

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

```
[82]: perm_mD_Log_BestModelVarioFit<-BestModel(XCoord, YCoord,
                                                perm_mD_Log, 0, 90, N_lags, lag_value, 1,
                                                "Mejor Ajuste del Variograma Adireccional"
                                                →de perm_mD_Log")
```

```
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

Warning message in variofit(Nuevo, ini = c(modelo_Exp00$cov.pars[1],
modelo_Exp00$cov.pars[2]), :
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Exp00$cov.pars[1],
modelo_Exp00$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Exp00$cov.pars[1],
modelo_Exp00$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

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

Warning message in variofit(Nuevo, ini = c(modelo_Sph00$cov.pars[1],
modelo_Sph00$cov.pars[2]), :
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Sph00$cov.pars[1],
modelo_Sph00$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Sph00$cov.pars[1],
modelo_Sph00$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

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
```

```

Warning message in variofit(Nuevo, ini = c(modelo_Exp0$cov.pars[1],
modelo_Exp0$cov.pars[2]), :
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Exp0$cov.pars[1],
modelo_Exp0$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Exp0$cov.pars[1],
modelo_Exp0$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

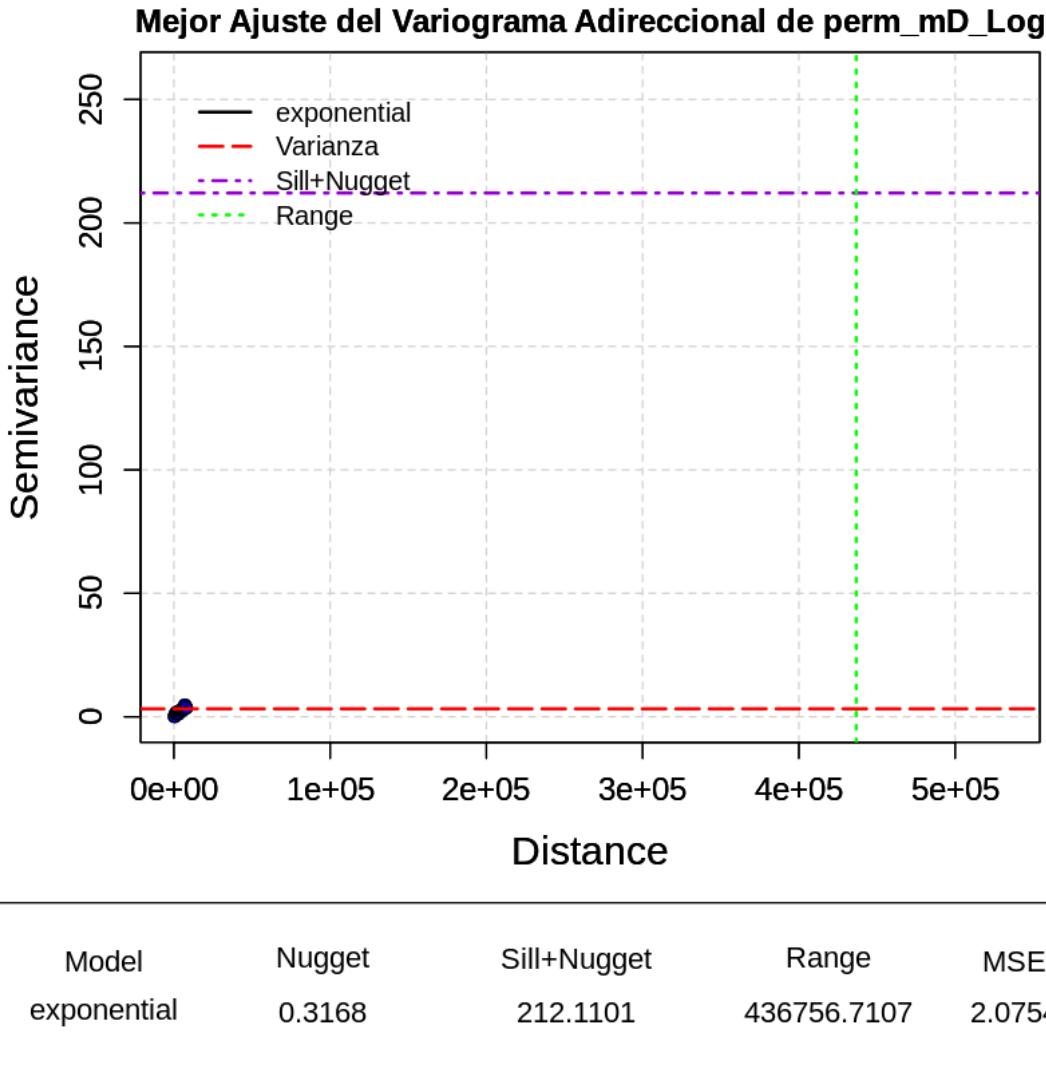
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim

Warning message in variofit(Nuevo, ini = c(modelo_Sph0$cov.pars[1],
modelo_Sph0$cov.pars[2]), :
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Sph0$cov.pars[1],
modelo_Sph0$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo_Sph0$cov.pars[1],
modelo_Sph0$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

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

Warning message in variofit(Nuevo, ini = c(modelo$cov.pars[1],
modelo$cov.pars[2]), :
“unreasonable initial value for sigmasq (too high)”
Warning message in variofit(Nuevo, ini = c(modelo$cov.pars[1],
modelo$cov.pars[2]), :
“unreasonable initial value for sigmasq + nugget (too high)”
Warning message in variofit(Nuevo, ini = c(modelo$cov.pars[1],
modelo$cov.pars[2]), :
“unreasonable initial value for phi (too high)”

```



[83]: perm\_mD\_Log\_BestModelVarioFit

A matrix: 1 × 6 of type dbl	Nugget	Meseta+Nugget	Alcance	SCE	MaxY	MinY
exponential	0.3167553	212.1101	436756.7	2.075357	4.634545	0.15096

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 los siguientes parámetros:

- \* Vector de las coordenadas (XCoord, YCoord)
- \* La variable (perm\_mD)
- \* La dirección del vector ( $0^\circ$ )
- \* Su ángulo de tolerancia ( $90^\circ$ )
- \* 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: \* Valor de nugget (perm\_mD\_nugget) \* Valor de meseta más nugget (perm\_mD\_sill\_and\_nugget) \* Alcance (perm\_mD\_rank).

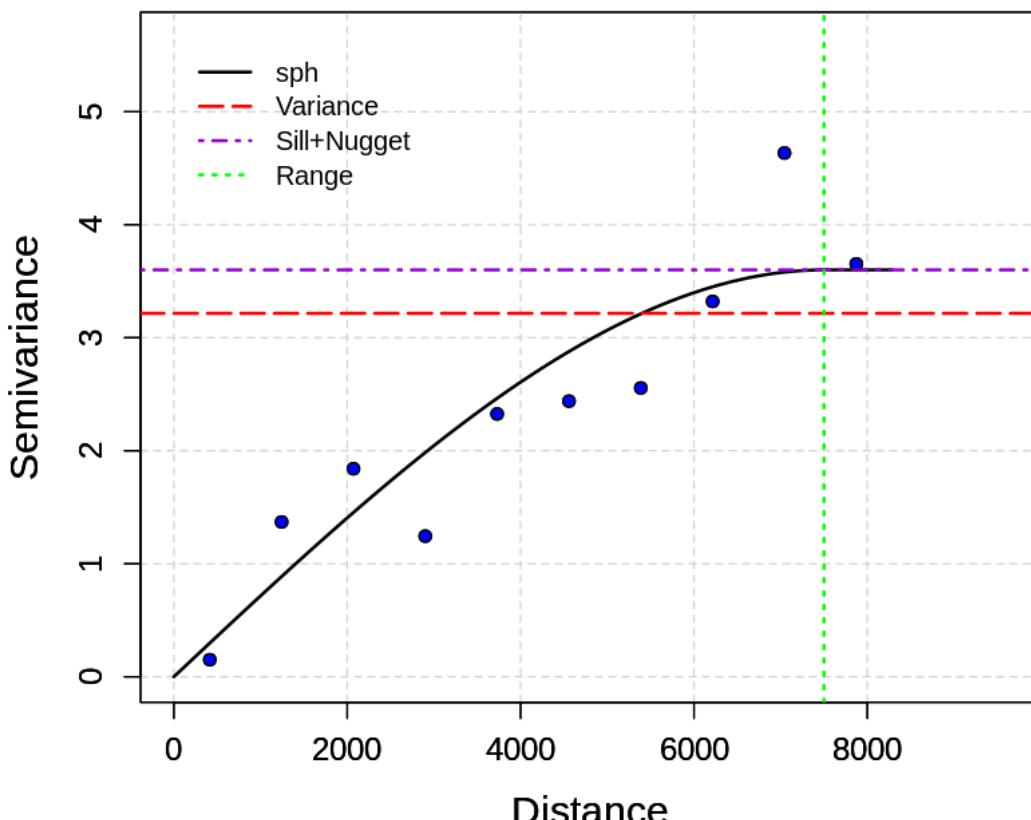
Estos valores se van cambiando bajo el criterio del usuario, el objetivo es lugar un ajuste con el menor error posible, pero con un ajuste adecuado, sin exagerar en el uso del nugget y dando prioridad a los intervalos con mayor número de pares.

```
[84] : perm_mD_Log_vario_model<- 2  
perm_mD_Log_nugget<- 0  
perm_mD_Log_sill_and_nugget<- 3.6  
perm_mD_Log_rank <- 7500
```

```
[85] : perm_mD_Log_EyeModelVarioFit<-EyeModel(XCoord, YCoord,  
perm_mD_Log, 0, 90, N_lags, lag_value, 1,  
perm_mD_Log_vario_model, perm_mD_Log_nugget,  
perm_mD_Log_sill_and_nugget,  
perm_mD_Log_rank,  
"Ajuste Manual del Variograma Adireccional"  
"de perm_mD")
```

variog: computing omnidirectional variogram

### Ajuste Manual del Variograma Adireccional de perm\_mD



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0000	3.6000	7500.0000	2.7253

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

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

La varianza.

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

Para realizar la validación cruzada usamos la función “CrossValidation”, la cual necesita los siguientes parámetros:

- Vectores de posicionamiento (XCoord, YCoord)
- La variable aleatoria (perm\_mD\_Log)
- Modelo de variograma (vario\_model)
- Valor de nugget (nugget)
- El valor de meseta más nugget (sill\_and\_nugget)
- El alcance (rank)
- Valores de anisotropía geométrica: 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 debe estar en el intervalo [0,1] y se calcula como la razón del eje menor dado el eje mayor.

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

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.

```
[87]: perm_mD_Log_CrossValid<- CrossValidation(XCoord, YCoord,
                                                perm_mD_Log, perm_mD_Log_vario_model,
                                                ↪perm_mD_Log_nugget,
                                                perm_mD_Log_sill_and_nugget,
                                                perm_mD_Log_rank, MaxAnis=0, proporcion=1)
```

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 ( $Z - Z^*$ )

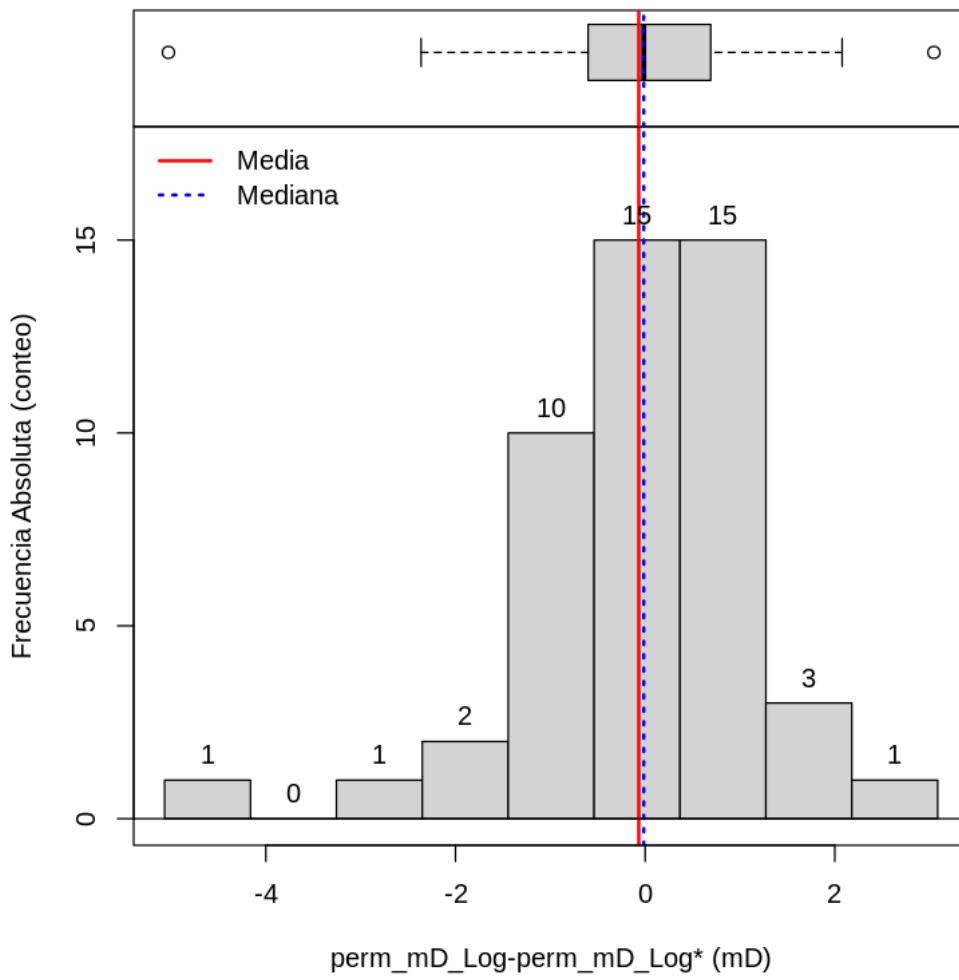
ya que tenemos la validación cruzada calculamos los estadígrafos.

```
[88]: perm_mD_Log_CrossValid_Sta <- Val_Estadisticos(perm_mD_Log_CrossValid[1:
                                                ↪102,c(3,4,5)])
write.csv(perm_mD_Log_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.

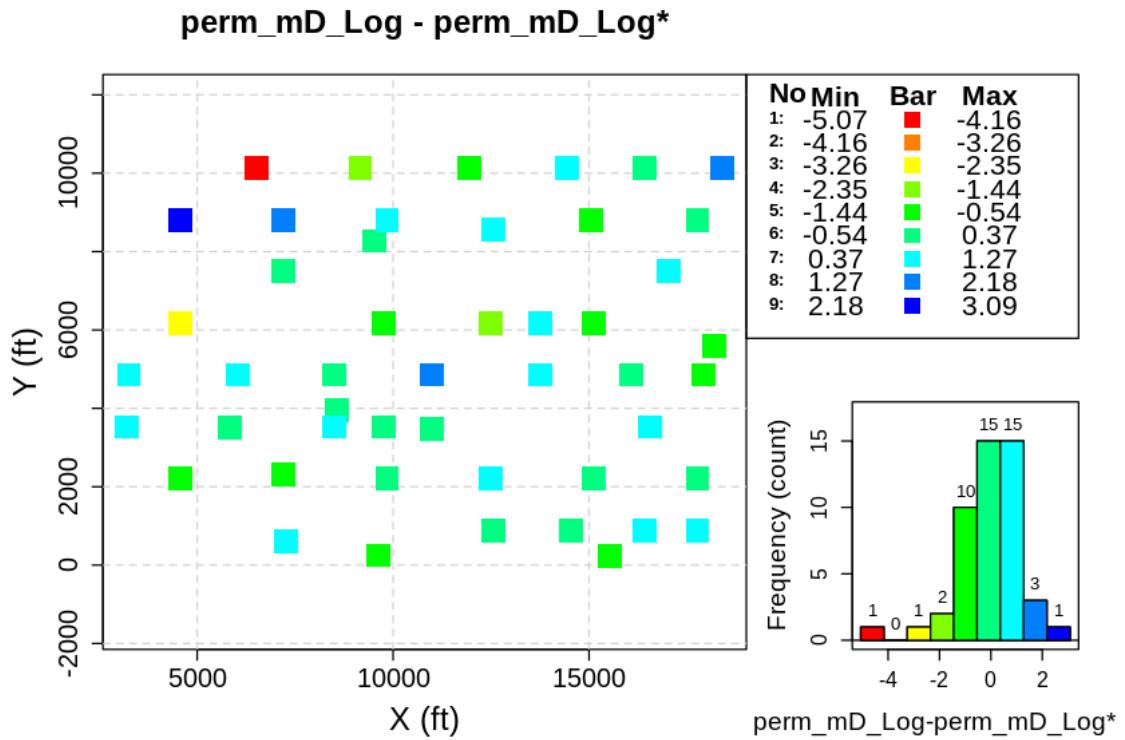
Ahora graficamos el histograma con los errores.

```
[89]: Histogram(x=perm_mD_Log_CrossValid[,5], mean = perm_mD_Log_CrossValid_Sta[5,3],
                median = perm_mD_Log_CrossValid_Sta[4,3], main ="",
                xlab = "perm_mD_Log-perm_mD_Log* (mD)", ylab = "Frecuencia Absoluta",
                ↪(conteo),
                AbsFreq = TRUE, PercentFreq = FALSE )
```



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”.

```
[90]: DEspacial(perm_mD_Log_CrossValid[,1], perm_mD_Log_CrossValid[,2],  
    ↪perm_mD_Log_CrossValid[,5], n_bins=9,  
    'X (ft)', 'Y (ft)', 'perm_mD_Log-perm_mD_Log* (mD)', 'perm_mD_Log -  
    ↪perm_mD_Log*')
```

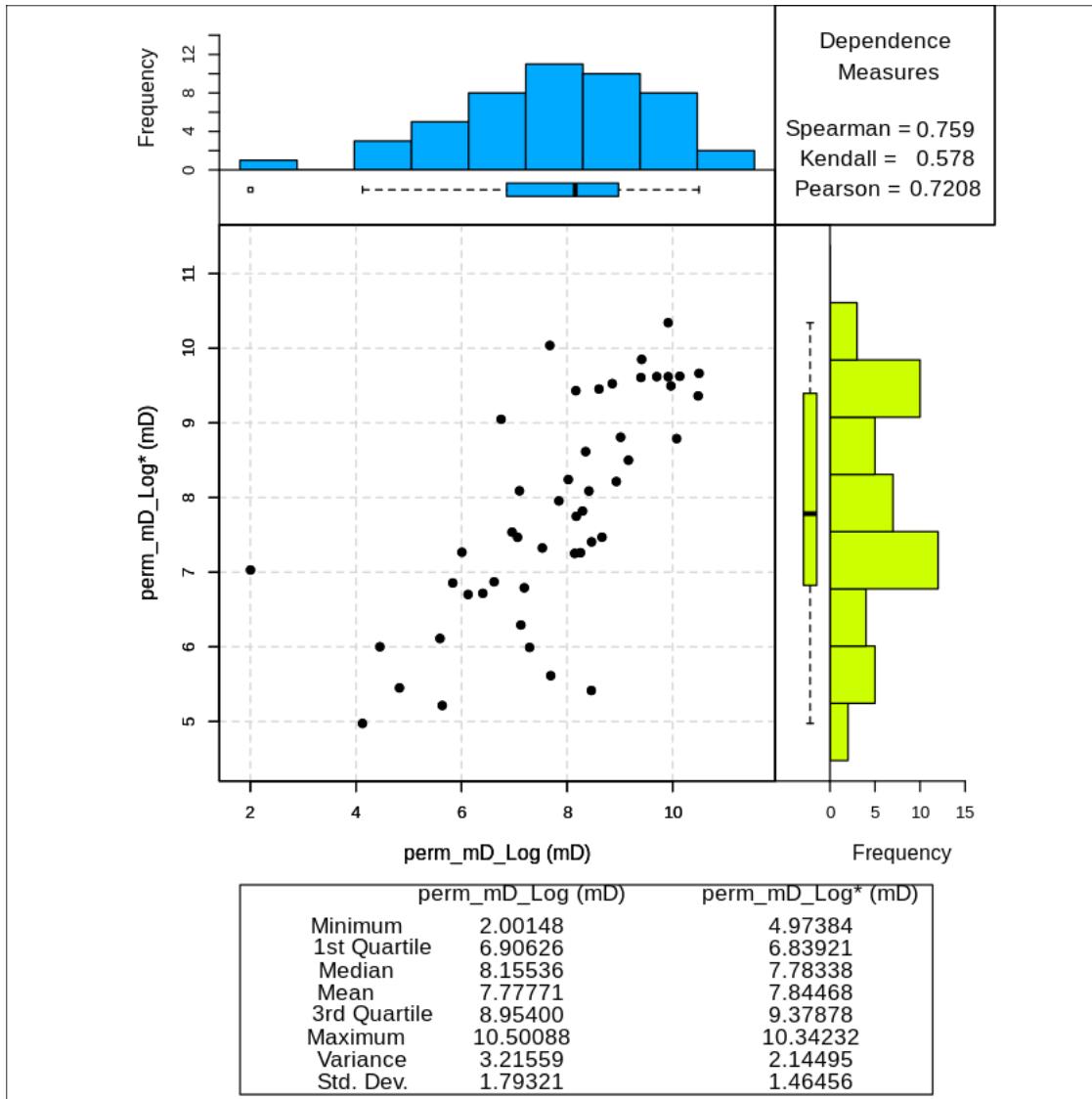


De este grafico podemos notar que el valor atípico negativo marcado en color rojo y azul fuerte estan en la frontera, por lo que es difícil saber si son valores atípicos.

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  $Z$  entonces podríamos esperar que la dependencia sea alta.

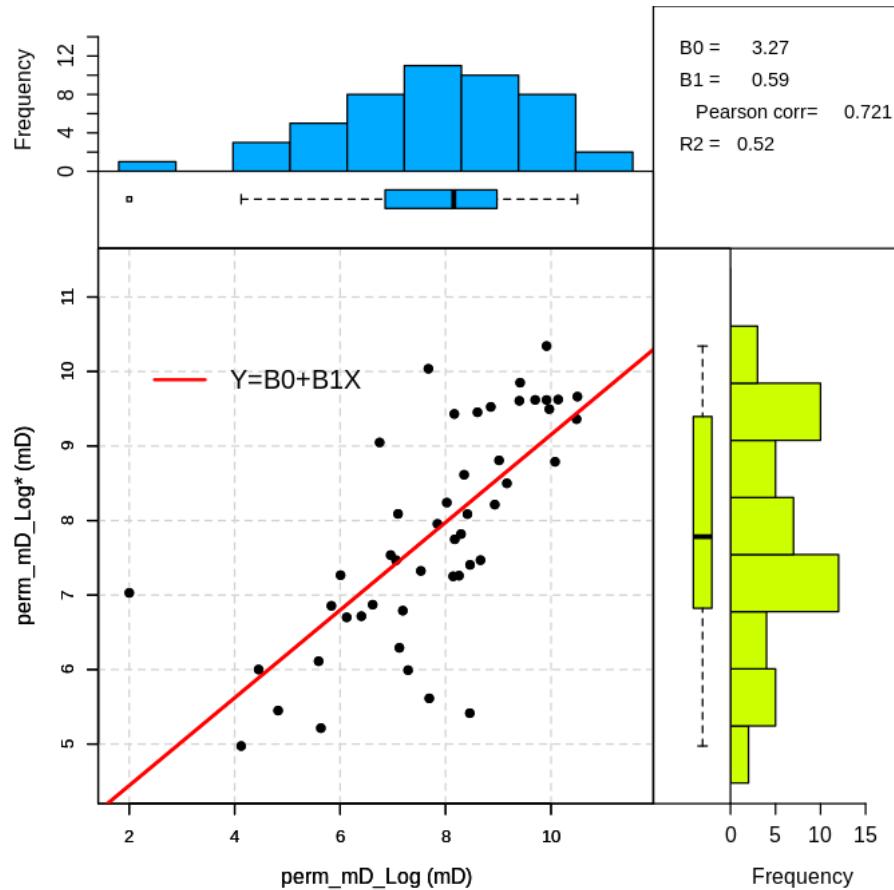
```
[91]: # perm_mD is the independent variable
X<-perm_mD_Log_CrossValid[,3]
# perm_mD* is the dependent variable
Y<-perm_mD_Log_CrossValid[,4]
```

```
[92]: ScatterPlot(perm_mD_Log_CrossValid[,3] , perm_mD_Log_CrossValid[,4] , 9,
                  Xmin = perm_mD_Log_CrossValid_Sta[2,1] , Xmax = perm_mD_Log_CrossValid_Sta[7,1] ,
                  Ymin = perm_mD_Log_CrossValid_Sta[2,2] , Ymax = perm_mD_Log_CrossValid_Sta[7,2] ,
                  XLAB = "perm_mD_Log (mD)" , YLAB = "perm_mD_Log* (mD)")
```



```
[93]: scaterplotReg(perm_mD_Log_CrossValid[,3] , perm_mD_Log_CrossValid[,4] , 9,
                  Xmin = perm_mD_Log_CrossValid_Sta[2,1] , Xmax = perm_mD_Log_CrossValid_Sta[7,1] ,
                  Ymin = perm_mD_Log_CrossValid_Sta[2,2] , Ymax = perm_mD_Log_CrossValid_Sta[7,2] ,
```

```
XLAB = "perm_mD_Log (mD)", YLAB = "perm_mD_Log* (mD)"
```

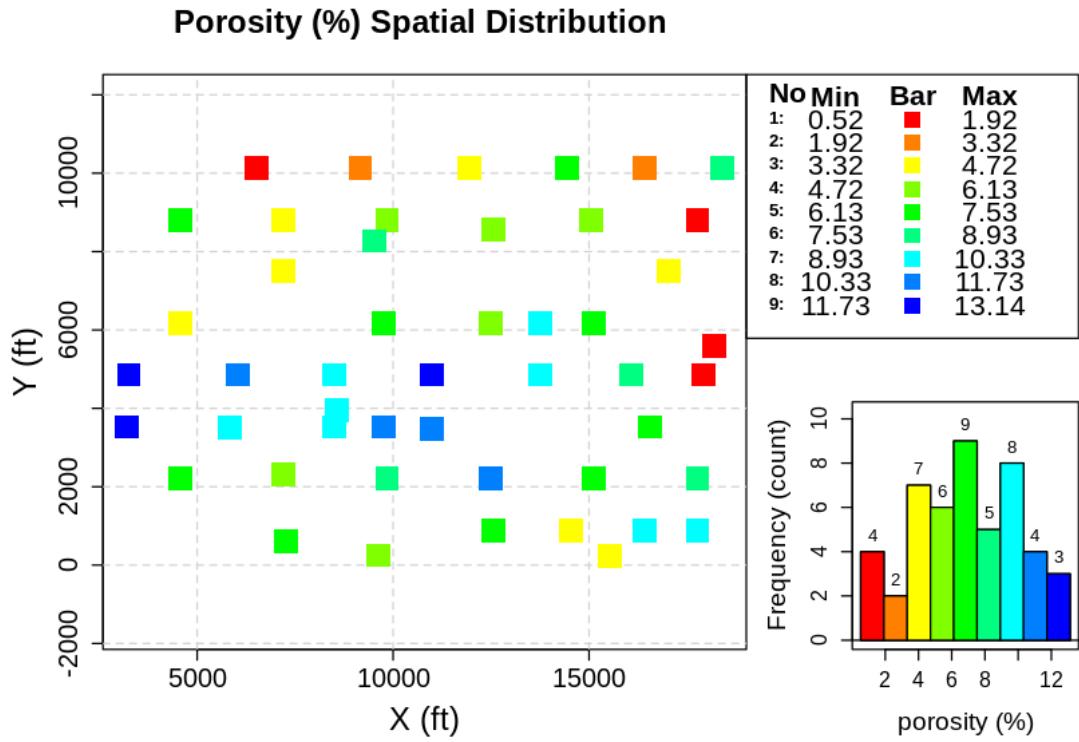


Observando los resultados del grafico de dispersión notamos que la dependencia es lo suficientemente alta para ofrecer una buena representación de la variable aleatoria a partir del variograma propuesto.

## 4.2 Análisis variográfico variable phi\_per

Ahora toca hacer el análisis variográfico de la variable `phi_per`. Primero obtenemos su distribución espacial.

```
[94]: DEspacial(XCoord, YCoord, phi_per, n_bins=9,
              'X (ft)', 'Y (ft)', 'porosity (%)', 'Porosity (%) Spatial Distribution')
       ↵')
```

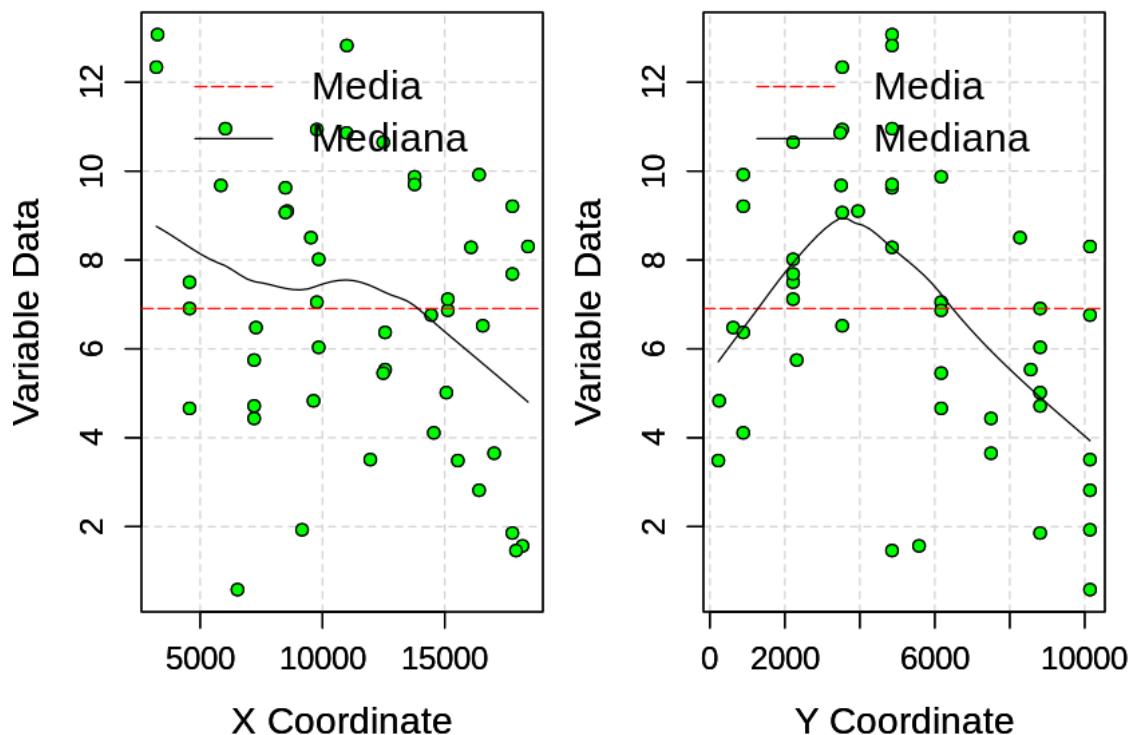


#### 4.2.1 Análisis de tendencia.

Al igual que la variable anterior, se hace el análisis de estacionaridad usando la función “Gdirecciones”.

```
[95]: GDirecciones(XCoord, YCoord, phi_per)
```

## Median Regression Analysis in X and Y directions



Respecto al análisis de tendencia usando la regresión de la mediana, se puede un efecto de tendencia en la coordenada X, por lo que veremos si este efecto se manifiesta en el variograma.

Estimamos el valor del intervalo y su número.

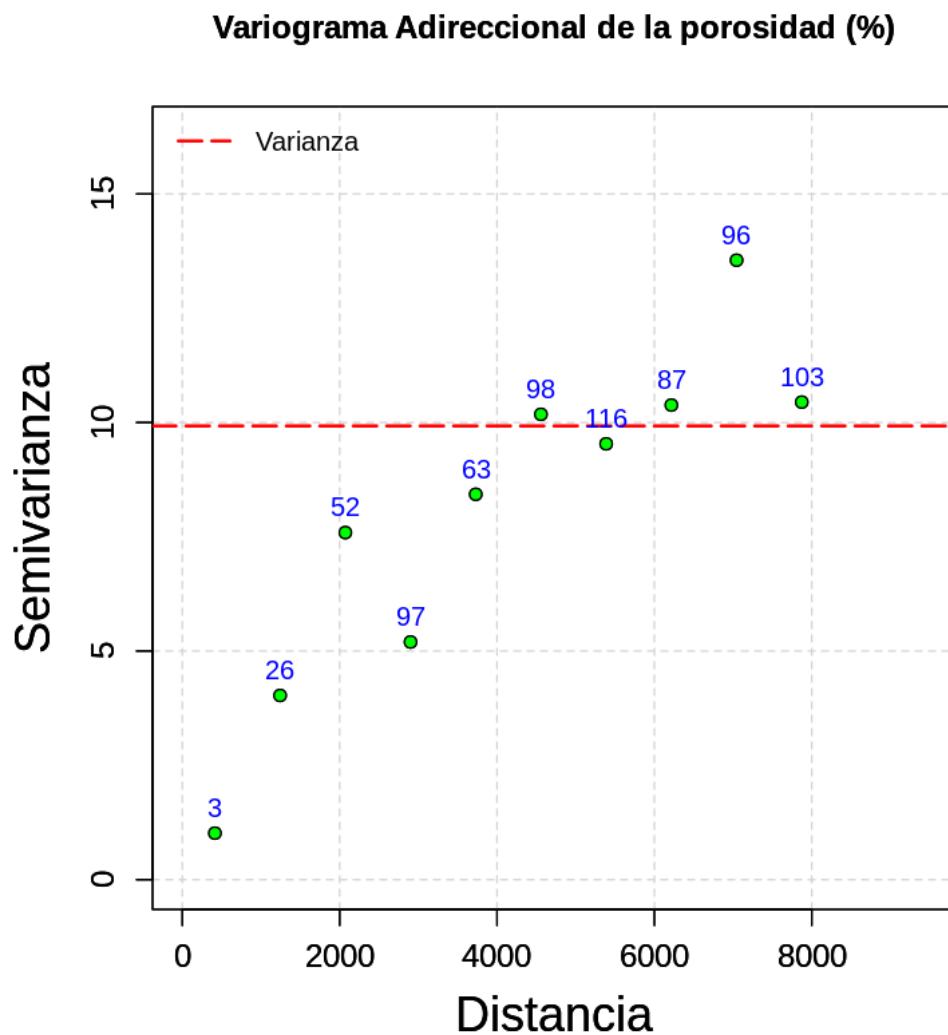
```
[96]: N_lags<-10
DistMin<-min(dist(Data_File_Burb[,1:2])) # Minimum distance in data
DistMax<-max(dist(Data_File_Burb[,1:2])) # Maximum distance in data
lag_value<- (DistMax/2)/N_lags # DistMin
lag_value
```

828.752224733062

Y calculamos el variograma adireccional.

```
[97]: phi_per_VarioEstimation<-Variogramma(XCoord, YCoord,
                                              phi_per, 0, 90, 1*N_lags, lag_value, 1,
                                              "Variogramma Adireccional de la porosidad"
                                              ↪"(%)")
```

variog: computing omnidirectional variogram



El variograma que se obtuvo muestra una buena estimación excepto en el primer intervalo, además podemos notar que el modelo se acota, por lo tanto, no hay tendencia.

```
[98]: phi_per_VarioEstimation
```

	Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
3	414.3761	1.018002	
26	1243.1283	4.029808	
52	2071.8806	7.592128	
97	2900.6328	5.198855	
63	3729.3850	8.429592	
98	4558.1372	10.179204	
116	5386.8895	9.533690	
87	6215.6417	10.381373	
96	7044.3939	13.549973	
103	7873.1461	10.446132	

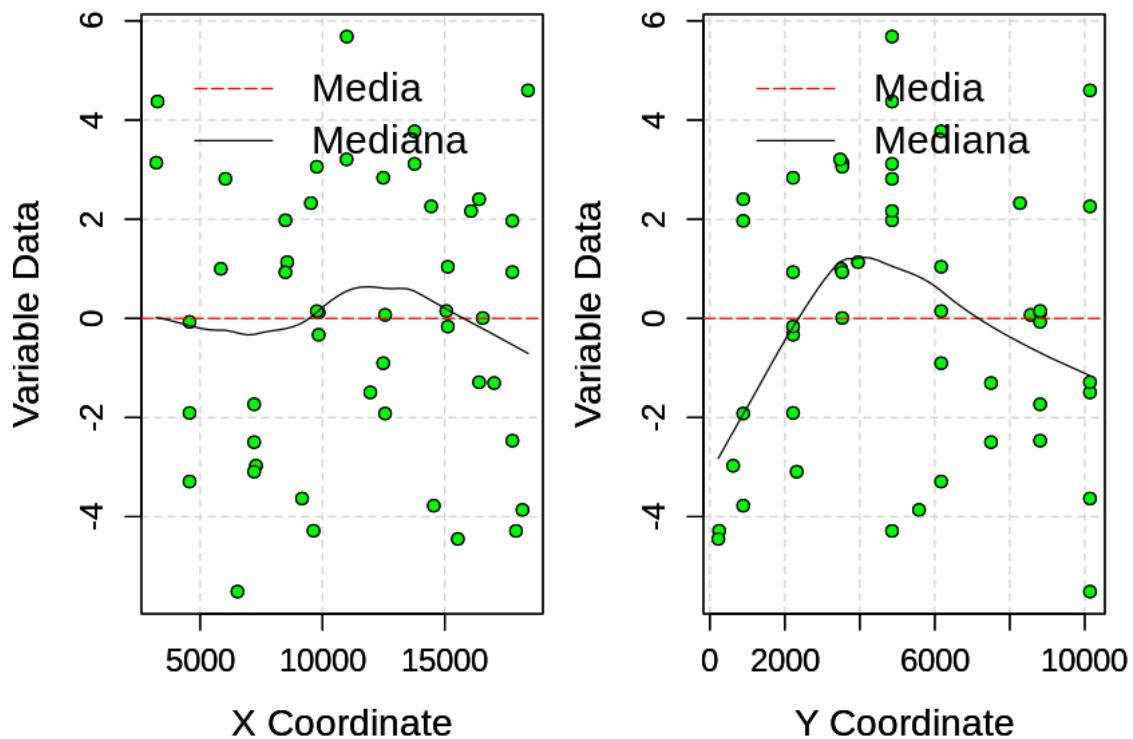
A data.frame: 10 × 3

Aquí recordamos que, en caso de que el variograma muestre tendencia, podemos usar la transformación polinomial. Al igual que la variable anterior, esto se hace usando la función “Trend”

```
[99]: pol_degree=1
phi_per_Detrended_1<-Trend(XCoord, YCoord,
                                phi_per, pol_degree)

[100]: GDirecciones(phi_per_Detrended_1[,1], phi_per_Detrended_1[,2], ↴
                     phi_per_Detrended_1[,3])
```

## Median Regression Analysis in X and Y directions

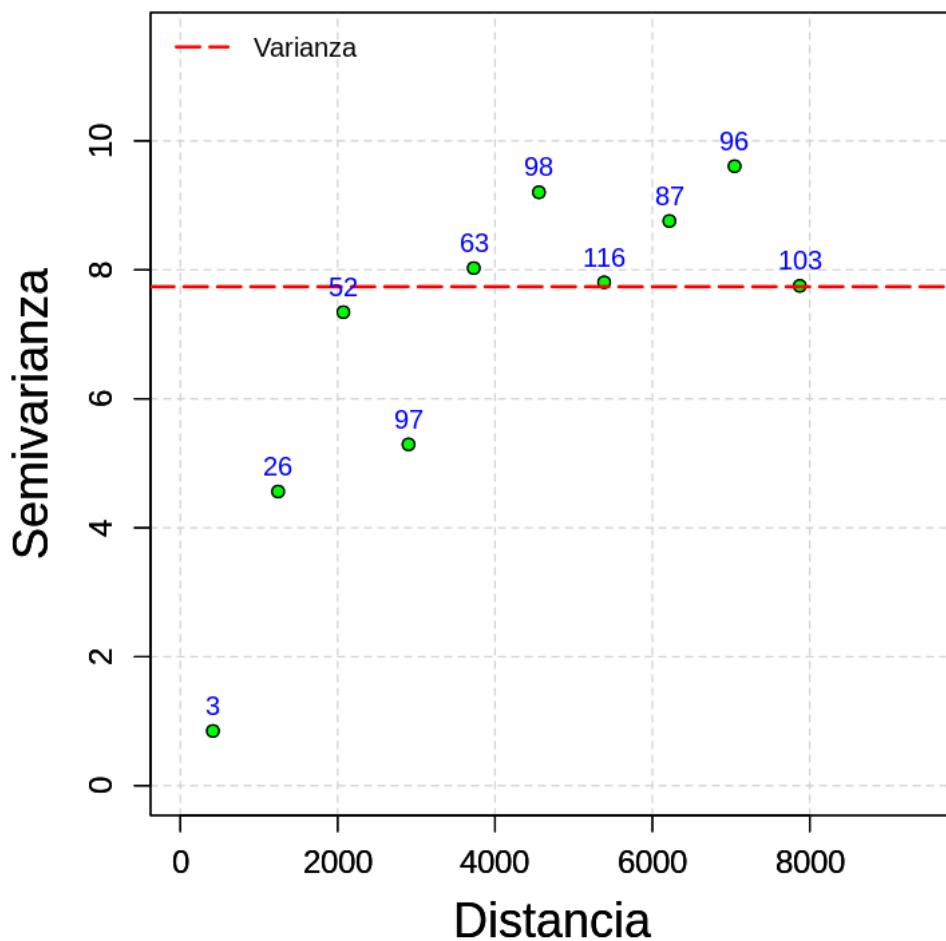


Al igual que la variable anterior, notamos la presencia de tendencia en la dirección X, veremos que sucede con el variograma estimado con la transformación de primer orden.

```
[101]: phi_per_Detrended_1_VarioEstimation<-Variograma(phi_per_Detrended_1[,1],  
          ↪phi_per_Detrended_1[,2],  
          ↪90, N_lags, lag_value, 1,  
          ↪"Variograma Adireccional de  
          ↪la porosidad (%) Residuos 1")
```

variog: computing omnidirectional variogram

### Variograma Adireccional de la porosidad (%) Residuos 1

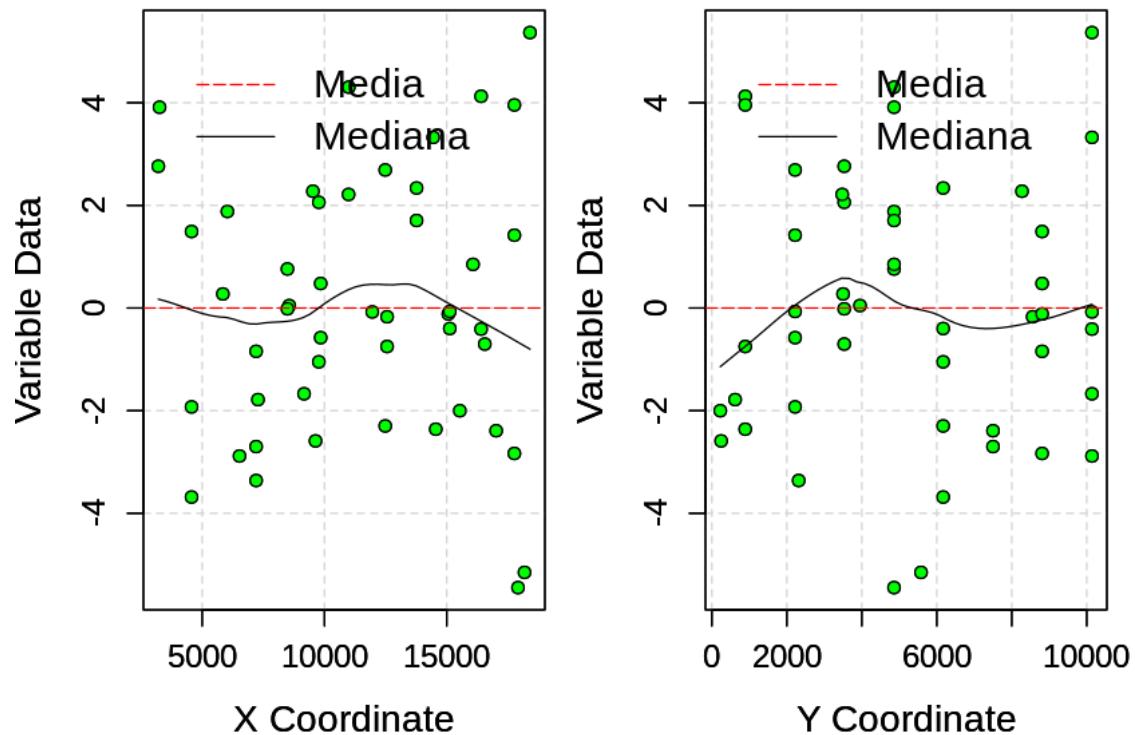


Para usar la transformación de segundo orden.

```
[102]: pol_degree=2  
phi_per_Detrended_2<-Trend(XCoord, YCoord,  
                                phi_per, pol_degree)
```

```
[103]: GDirecciones(phi_per_Detrended_2[,1], phi_per_Detrended_2[,2],  
                   phi_per_Detrended_2[,3])
```

## Median Regression Analysis in X and Y directions

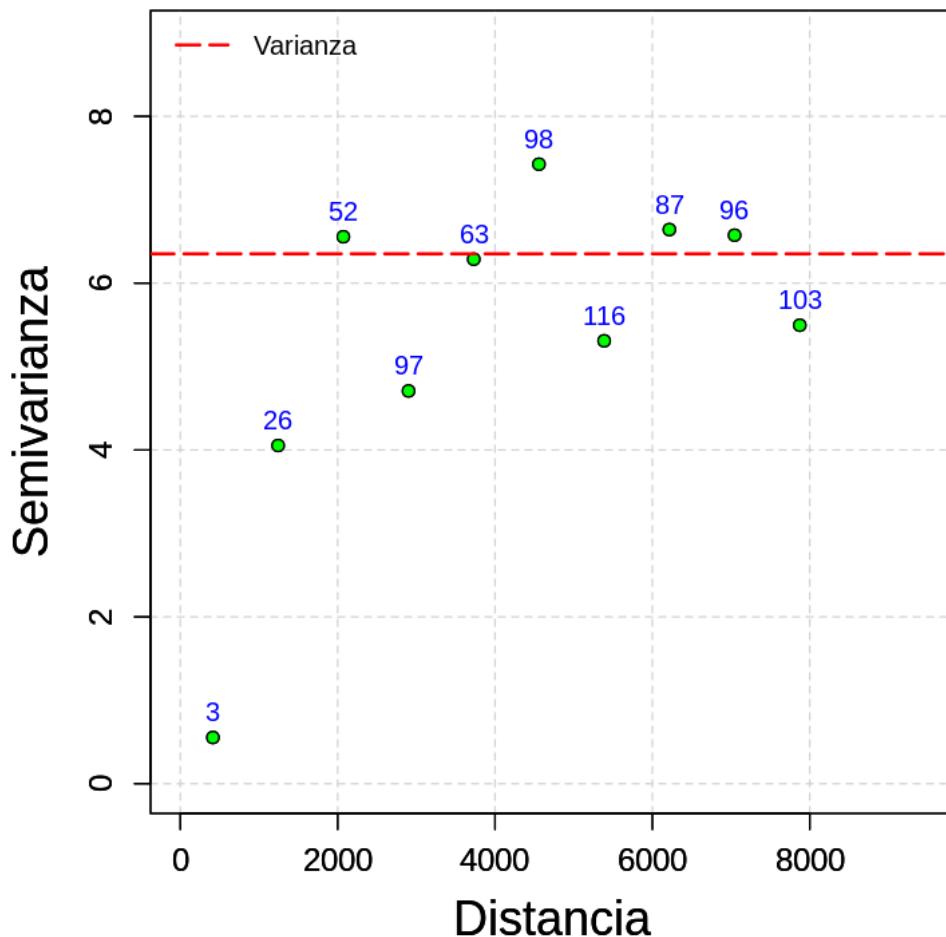


Usando la transformada de segundo orden podemos notar que ambas coordenadas fueron mejoradas.

```
[104]: phi_per_Detrended_2_VarioEstimation<-Variograma(phi_per_Detrended_2[,1],  
         ~phi_per_Detrended_2[,2],  
         ~90, N_lags, lag_value, 1,  
         phi_per_Detrended_2[,3], 0,  
         "Variograma Adireccional de  
         ~la porosidad (%) Residuos 2")
```

variog: computing omnidirectional variogram

## Variograma Adireccional de la porosidad (%) Residuos 2



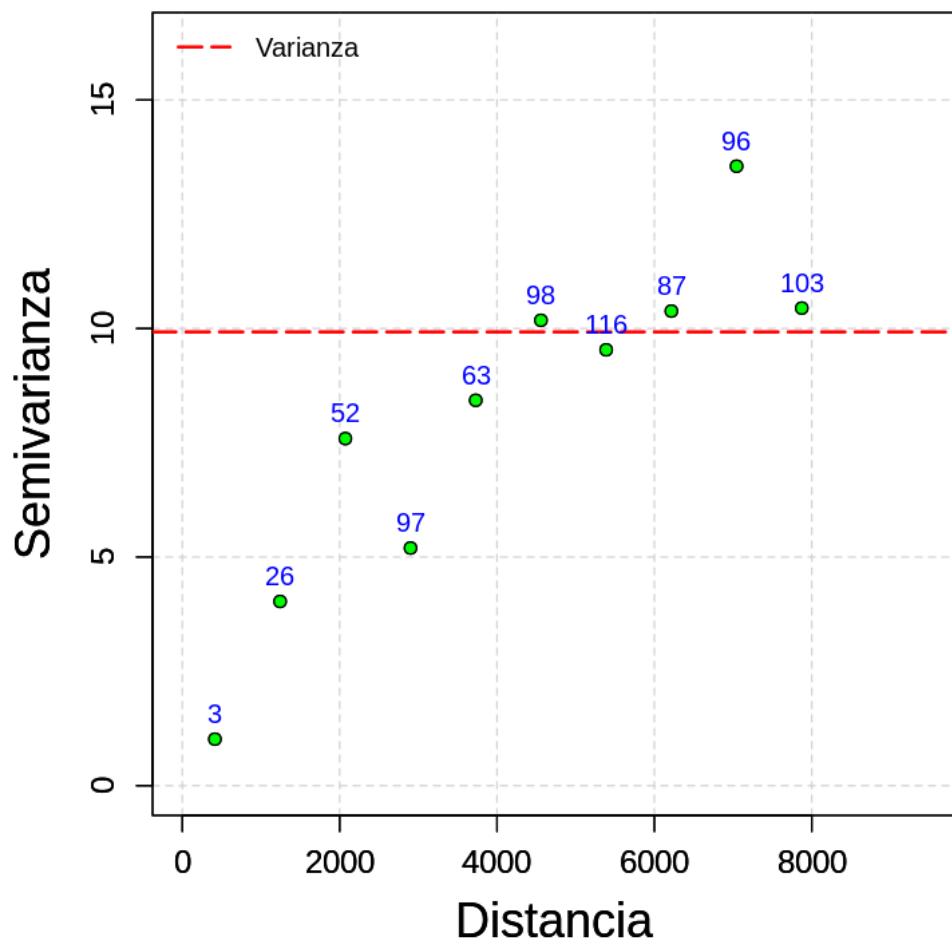
El variograma que se obtuvo con la transformación de segundo orden empeoro. Este ejemplo sirve para notar el efecto de la transformación, si se usa de forma incorrecta podríamos tener una peor estimación del variograma experimental.

Dados los resultados se decidió usar el variograma adireccional para hacer el ajuste de modelo.

```
[105]: phi_per_VarioEstimation<-Variograma(XCoord, YCoord,
                                         phi_per, 0, 90, N_lags, lag_value, 1,
                                         "Variograma Adireccional de la porosidad
                                         →(%)")
```

variog: computing omnidirectional variogram

### Variograma Adireccional de la porosidad (%)



```
[106]: phi_per_VarioEstimation
```

	Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
A data.frame: 10 × 3	3	414.3761	1.018002
	26	1243.1283	4.029808
	52	2071.8806	7.592128
	97	2900.6328	5.198855
	63	3729.3850	8.429592
	98	4558.1372	10.179204
	116	5386.8895	9.533690
	87	6215.6417	10.381373
	96	7044.3939	13.549973
	103	7873.1461	10.446132

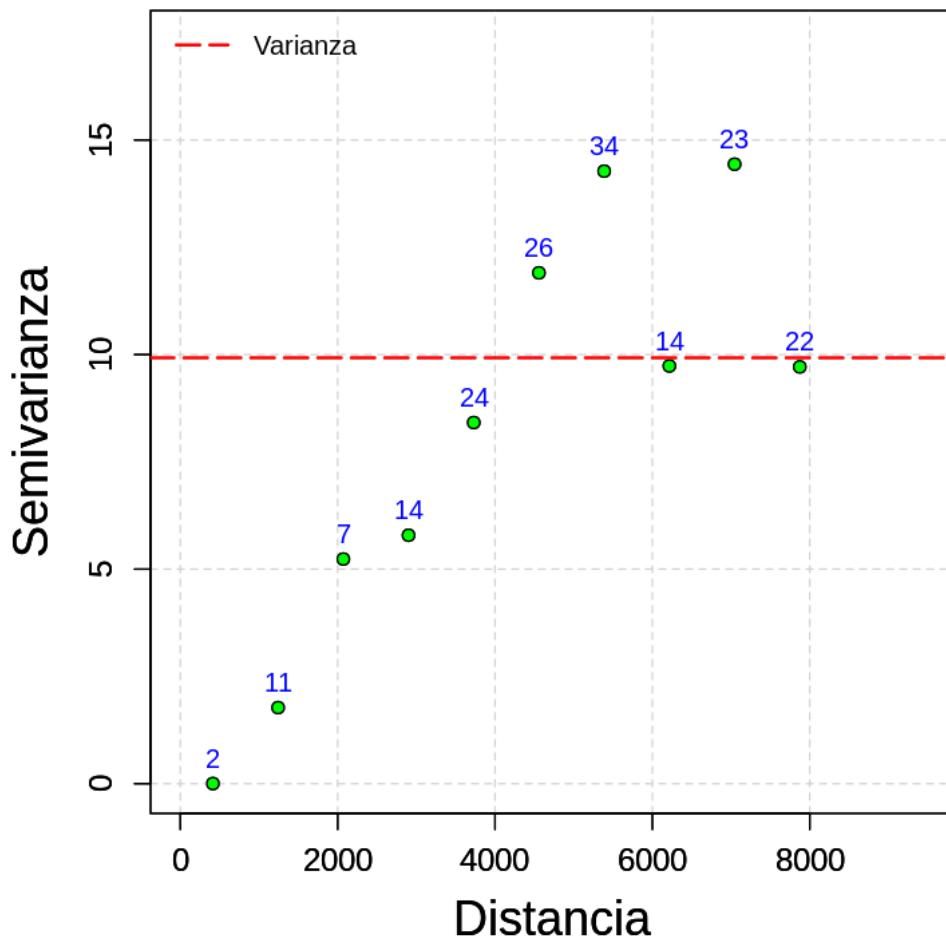
Ahora que sabemos cuál es el mejor variograma procedemos a calcular los variogramas direccionales y determinar si existe algún tipo de anisotropía. Comenzamos con la dirección 0º.

```
[107]: phi_per_VarioEstimation_0<-Variograma(XCoord, YCoord,
                                              phi_per, 0, 22.5, N_lags, lag_value, 1,
                                              "Variograma direccional 0º de la↓
                                              →porosidad(%))"
phi_per_VarioEstimation_0
```

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

	Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
A data.frame: 10 × 3	2	414.3761	0.00301225
	11	1243.1283	1.77355686
	7	2071.8806	5.23544907
	14	2900.6328	5.79010668
	24	3729.3850	8.41596610
	26	4558.1372	11.90553417
	34	5386.8895	14.27497454
	14	6215.6417	9.73564586
	23	7044.3939	14.43718441
	22	7873.1461	9.71383941

### Variograma direccional 0º de la porosidad(%)



En esta dirección se puede observar que este variograma no está bien estimado, solo un intervalo supera los 30 pares.

Seguimos con la dirección de 45º.

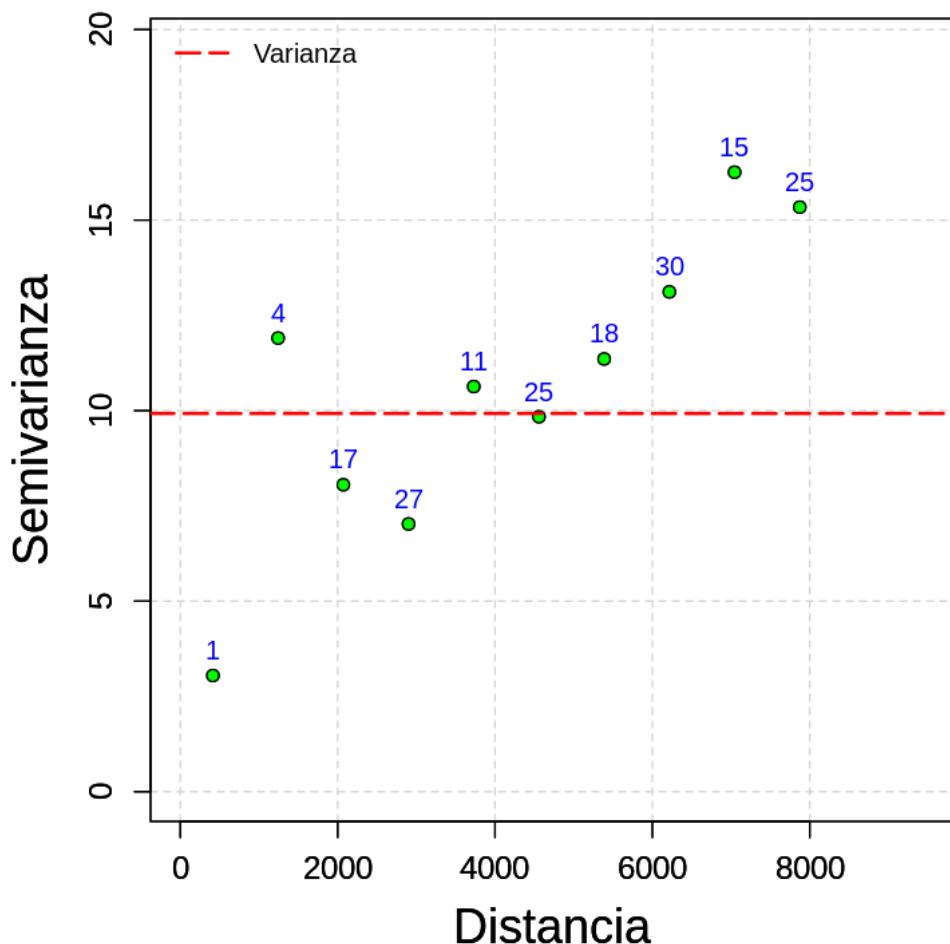
```
[108]: phi_per_VarioEstimation_45<-Variograma(XCoord, YCoord,
                                              phi_per, 45, 22.5, N_lags, lag_value, 1,
                                              "Variograma direccional 45º de laper
                                              →porosidad(%))")
phi_per_VarioEstimation_45
```

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

A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
1	414.3761	3.047981
4	1243.1283	11.903782
17	2071.8806	8.054144
27	2900.6328	7.025295
11	3729.3850	10.631665
25	4558.1372	9.840083
18	5386.8895	11.355219
30	6215.6417	13.115817
15	7044.3939	16.256710
25	7873.1461	15.339386

### Variograma direccional 45º de la porosidad(%)



El variograma con dirección 45º tampoco está bien estimado, solo un intervalo supera los 30 pares.

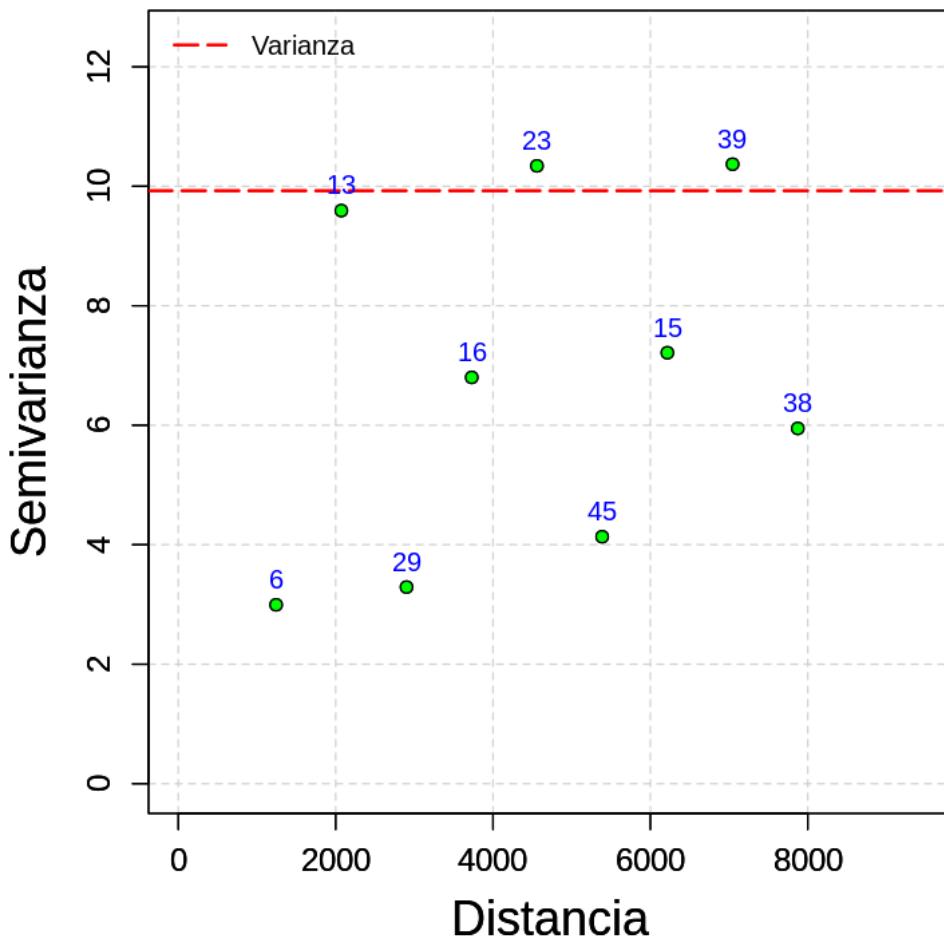
Calculamos el variograma con dirección 90º.

```
[109]: phi_per_VarioEstimation_90<-Variograma(XCoord, YCoord,
                                              phi_per, 90, 22.5, N_lags, lag_value, 1,
                                              "Variograma direccional 90º de la_"
                                              →porosidad(%))
phi_per_VarioEstimation_90
```

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

	Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
A data.frame: 9 × 3	6	1243.128	2.994113
	13	2071.881	9.592904
	29	2900.633	3.290863
	16	3729.385	6.800735
	23	4558.137	10.341227
	45	5386.889	4.133571
	15	6215.642	7.213451
	39	7044.394	10.369248
	38	7873.146	5.946595

### Variograma direccional 90° de la porosidad(%)



En este caso podemos notar que el variograma no está bien estimado, solo un intervalo tiene más de 30 pares.

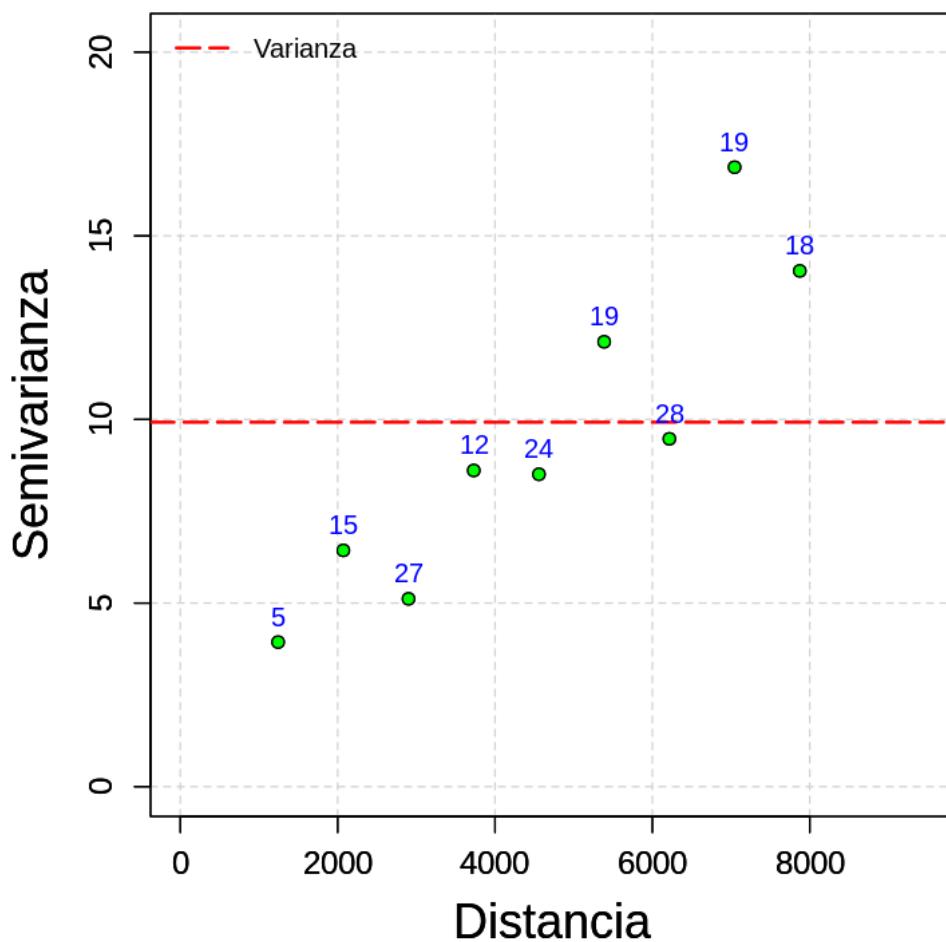
Calculamos la dirección 135°.

```
[110]: phi_per_VarioEstimation_135<-Variogramma(XCoord, YCoord,
                                                 phi_per, 135, 22.5, N_lags, lag_value,
                                                 1,
                                                 "Variograma direccional 135° de la"
                                                 &porosidad(%))
```

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

	Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
	5	1243.128	3.937217
	15	2071.881	6.434287
	27	2900.633	5.115165
A data.frame: 9 × 3	12	3729.385	8.610087
	24	4558.137	8.506993
	19	5386.889	12.113380
	28	6215.642	9.471576
	19	7044.394	16.867941
	18	7873.146	14.043993

### Variograma direccional 135° de la porosidad(%)



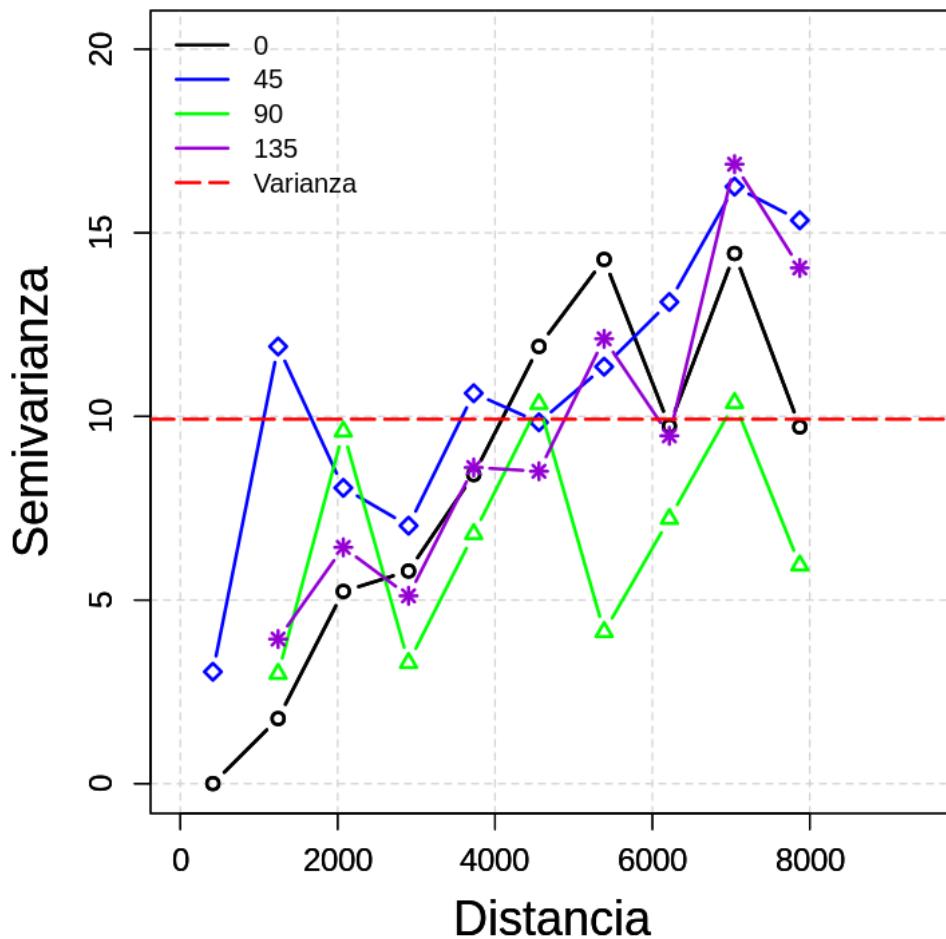
Con este caso podemos notar que es el peor variograma experimental, no hay intervalo que tenga más de 30 pares.

Debido a los malos resultados se recomienda usar el variograma adireccional y considerar que el los variogramas direccionales no permiten establecer si existe algún tipo de anisotropía, por lo tanto, consideraremos que el variograma es isotrópico.

```
[111]: phi_per_VarioEstimation4D<-Variograma4D(XCoord, YCoord,
                                              phi_per, 0, 45, 90, 135, 22.5, N_lags, □
                                              ↵lag_value, 1,
                                              "Variogramas Direccionales de la"
                                              ↵porosidad(%))
```

```
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)
```

### Variogramas Direccionales de la porosidad(%)



```
[112]: phi_per_VarioEstimation4D
```

2	414.3761	0.00301225
11	1243.1283	1.77355686
7	2071.8806	5.23544907
14	2900.6328	5.79010668
24	3729.3850	8.41596610
26	4558.1372	11.90553417
34	5386.8895	14.27497454
14	6215.6417	9.73564586
23	7044.3939	14.43718441
22	7873.1461	9.71383941

```

1   414.3761  3.047981
4   1243.1283 11.903782
17  2071.8806 8.054144
27  2900.6328 7.025295
11  3729.3850 10.631665
25  4558.1372 9.840083
18  5386.8895 11.355219
30  6215.6417 13.115817
15  7044.3939 16.256710
25  7873.1461 15.339386

6   1243.128  2.994113
13  2071.881  9.592904
29  2900.633  3.290863
16  3729.385  6.800735
$Ninety A matrix: 9 × 3 of type dbl 23  4558.137  10.341227
45  5386.889  4.133571
15  6215.642  7.213451
39  7044.394  10.369248
38  7873.146  5.946595

5   1243.128  3.937217
15  2071.881  6.434287
27  2900.633  5.115165
12  3729.385  8.610087
$OneThertyFive A matrix: 9 × 3 of type dbl 24  4558.137  8.506993
19  5386.889  12.113380
28  6215.642  9.471576
19  7044.394  16.867941
18  7873.146  14.043993

```

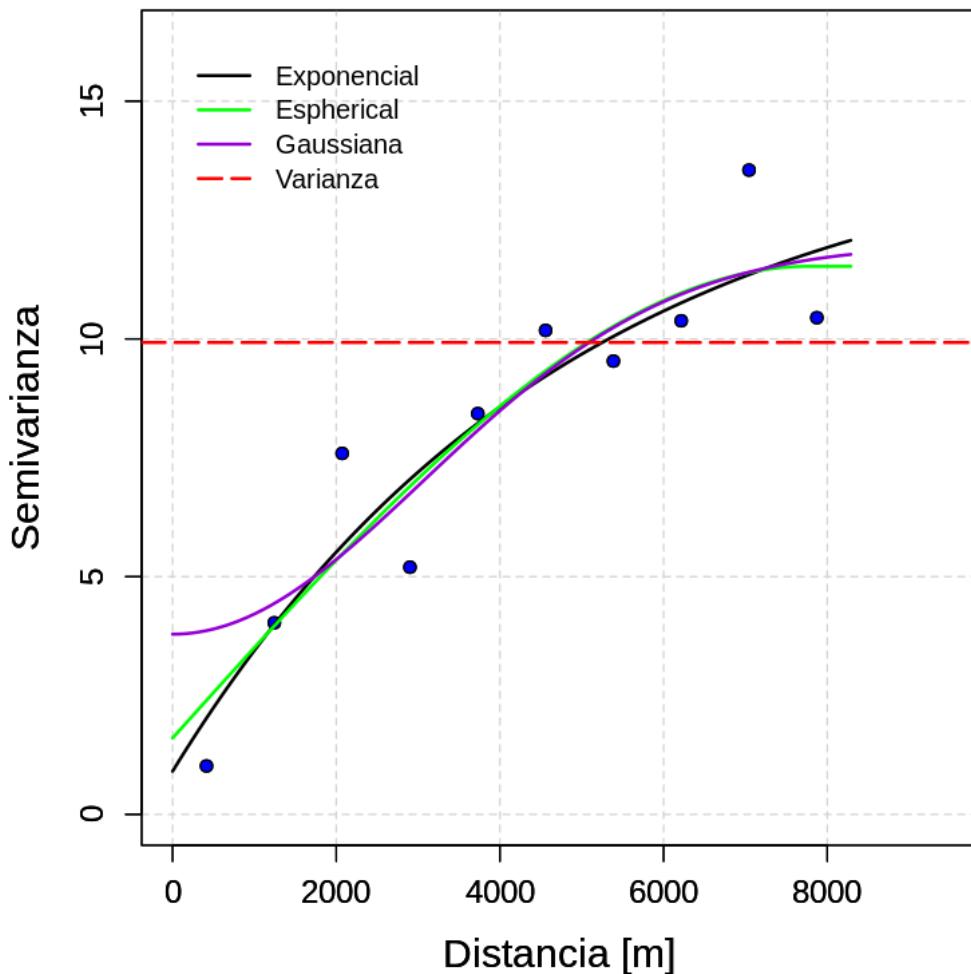
Ahora estimamos los modelos de variograma autorizados:

```
[113]: phi_per_AllModelVarioFit<-AllModel(XCoord, YCoord,
                                         phi_per, 0, 90, N_lags, lag_value, 1,
                                         "Ajustes del Variograma Adireccional de la porosidad(%)")


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
```

### Ajustes del Variograma Adireccional de la porosidad(%)



```
[114]: phi_per_AllModelVarioFit
```

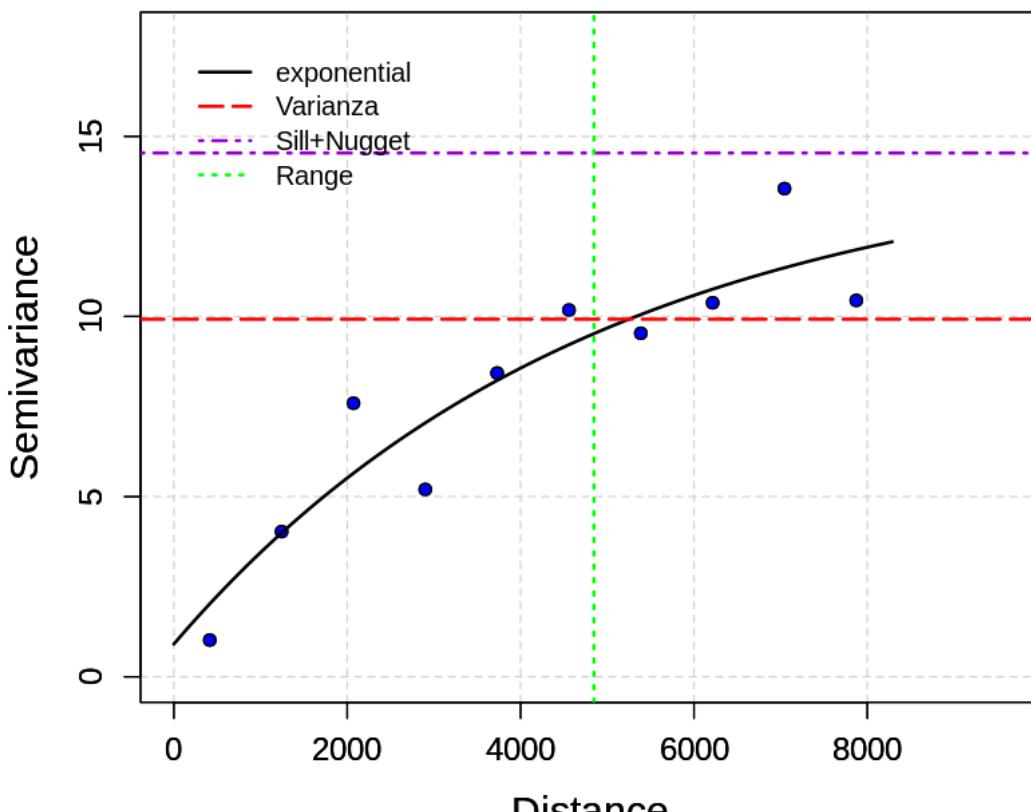
		Nugget	Meseta+Nugget	Alcance	SCE
A matrix: 3 × 4 of type dbl	exponential	0.9080503	14.53796	4846.160	16.39316
	spherical	1.6054352	11.52603	7775.276	16.72357
	gaussian	3.7896798	11.98895	4335.085	58.83218

El resultado de este ajuste automatico muestra que los modelos exponencial y esferico son los que tienen menor error.

```
[115]: phi_per_BestModelVarioFit<-BestModel(XCoord, YCoord,  
                                              phi_per, 0, 90, N_lags, lag_value, 1,  
                                              "Mejor Ajuste del Variograma  
→Adireccional de la porosidad (%)" )
```

```
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
```

### Mejor Ajuste del Variograma Adireccional de la porosidad (%)



Model	Nugget	Sill+Nugget	Range	MSE
exponential	0.9081	14.5380	4846.1603	16.3932

```
[116]: phi_per_BestModelVarioFit
```

A matrix: 1 × 6 of type dbl	Nugget	Meseta+Nugget	Alcance	SCE	MaxY	MinY
exponential	0.9080503	14.53796	4846.16	16.39316	13.54997	1.018002

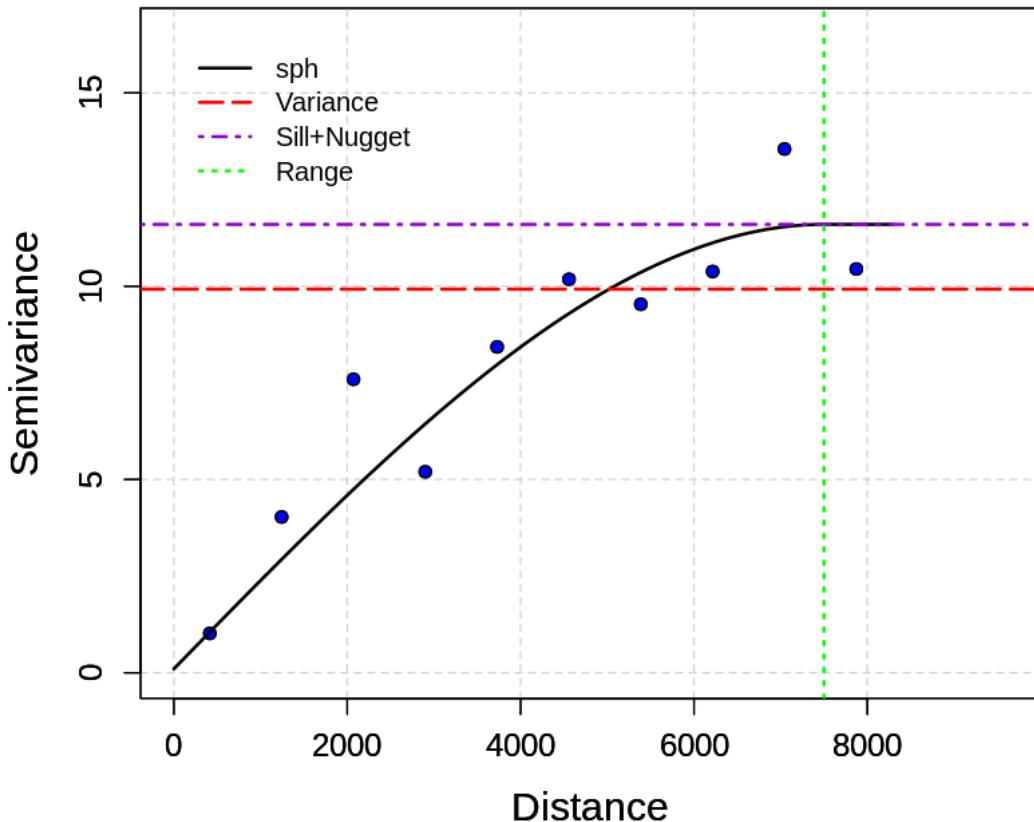
Ajustamos el variograma a un modelo esférico de forma manual y obtenemos lo siguiente:

```
[117]: #modelos de variograma (1- exponential, 2- spherical, 3- gaussian)
vario_model<- 2 # 2
nugget<- 0.1 # 0.08
sill_and_nugget<- 11.6 # 1.1
rank <- 7500 # 15000b
```

```
[118]: phi_per_EyeModelVarioFit<-EyeModel(XCoord, YCoord,
                                         phi_per, 0, 90, N_lags, lag_value, 1,
                                         vario_model, nugget, sill_and_nugget, rank,
                                         "Ajuste Manual del Variograma",
                                         "→Adireccional de la porosidad(%)" )
```

variog: computing omnidirectional variogram

### Ajuste Manual del Variograma Adireccional de la porosidad(%)



Ya que tenemos el mejor ajuste calculamos la validación cruzada.

```
[119]: phi_per_CrossValid<- CrossValidation(XCoord, YCoord,
                                             phi_per, vario_model, nugget,
                                             "→sill_and_nugget, rank,
```

```
MaxAnis=0, proporcion=1)  
phi_per_CrossValid
```

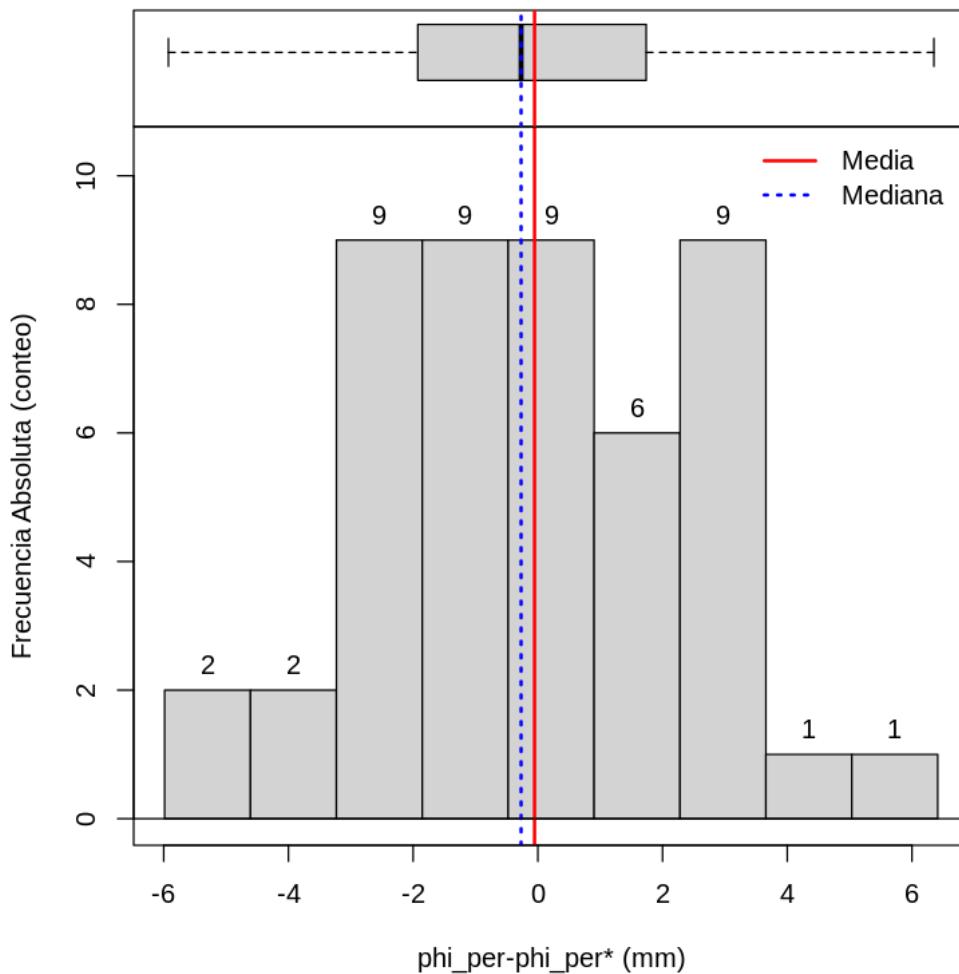
	X <dbl>	Y <dbl>	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
1	12561	8558	5.533	5.668759	-0.13575937
2	11955	10140	3.508	5.218015	-1.71001473
3	9527	8273	8.503	5.726954	2.77604586
4	9845	8810	6.034	6.435881	-0.40188060
5	9165	10140	1.929	3.812239	-1.88323879
6	9770	6170	7.051	9.705413	-2.65441269
7	12485	6170	5.454	10.377800	-4.92379999
8	7205	8810	4.715	2.955867	1.75913338
9	6525	10141	0.582	4.687265	-4.10526519
10	4565	8810	6.908	3.183656	3.72434414
11	4565	6170	4.660	10.584353	-5.92435329
12	7205	7500	4.434	6.639971	-2.20597138
13	5845	3500	9.678	9.015651	0.66234938
14	6029	4860	10.955	8.214510	2.74049047
15	3205	3530	12.339	10.892186	1.44681444
16	3255	4860	13.073	9.754764	3.31823597
17	4565	2220	7.500	9.998811	-2.49881076
18	7280	620	6.479	4.762273	1.71672718
19	7205	2316	5.748	8.034545	-2.28654477
20	8559	3952	9.100	9.595122	-0.49512158
21	8485	3530	9.068	8.765665	0.30233484
22	8485	4860	9.625	8.917950	0.70704971
23	9770	3530	10.936	9.985099	0.95090075
24	10995	4860	12.826	9.301874	3.52412555
25	10985	3477	10.859	11.520951	-0.66195069
26	9845	2220	8.015	8.640772	-0.62577161
27	9633	247	4.831	6.798357	-1.96735737
28	12560	890	6.370	6.777066	-0.40706561
29	12485	2220	10.650	8.111978	2.53802160
30	17765	8810	1.856	4.872963	-3.01696326
31	16405	10140	2.818	6.019598	-3.20159798
32	18405	10140	8.303	1.951523	6.35147730
33	15065	8810	5.015	5.116216	-0.10121618
34	14445	10140	6.760	4.014159	2.74584133
35	18180	5580	1.566	2.098742	-0.53274192
36	17020	7500	3.651	2.455596	1.19540378
37	13765	6170	9.873	6.974979	2.89802099
38	15125	6170	6.868	7.932685	-1.06468481
39	16555	3530	6.520	7.132191	-0.61219096
40	16075	4860	8.284	5.796812	2.48718753
41	17915	4860	1.461	3.539454	-2.07845397
42	13765	4860	9.700	9.795519	-0.09551899
43	15125	2220	7.118	7.796928	-0.67892780
44	14552	890	4.110	5.521579	-1.41157869
45	15535	225	3.486	6.887227	-3.40122686
46	16405	890	9.920	6.262535	3.65746546
47	17765	890	9.209	8.262487	0.94651309
48	17765	2220	7.065	7.584849	0.10015100

```
[120]: phi_per_CrossValid_Stat<- Val_Estadisticos(phi_per_CrossValid[,c(3,4,5)])
phi_per_CrossValid_Stat
```

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	48.00000	48.00000	48.00000
Minimo	0.58200	1.95152	-5.92435
Cuartil_1er	4.70125	5.19257	-1.90427
Mediana	6.88800	6.93110	-0.26882
Media	6.90763	6.96041	-0.05279
A data.frame: 13 × 3	Cuartil_3er	9.31300	8.94238
	Maximo	13.07300	11.52095
	Rango	12.49100	9.56943
	Rango_Intercuartil	4.61175	3.74981
	Varianza	9.92476	6.29876
	Desv_Estandar	3.15036	2.50973
	Simetria	-0.05236	-0.22998
	Curtosis	2.28307	2.16024

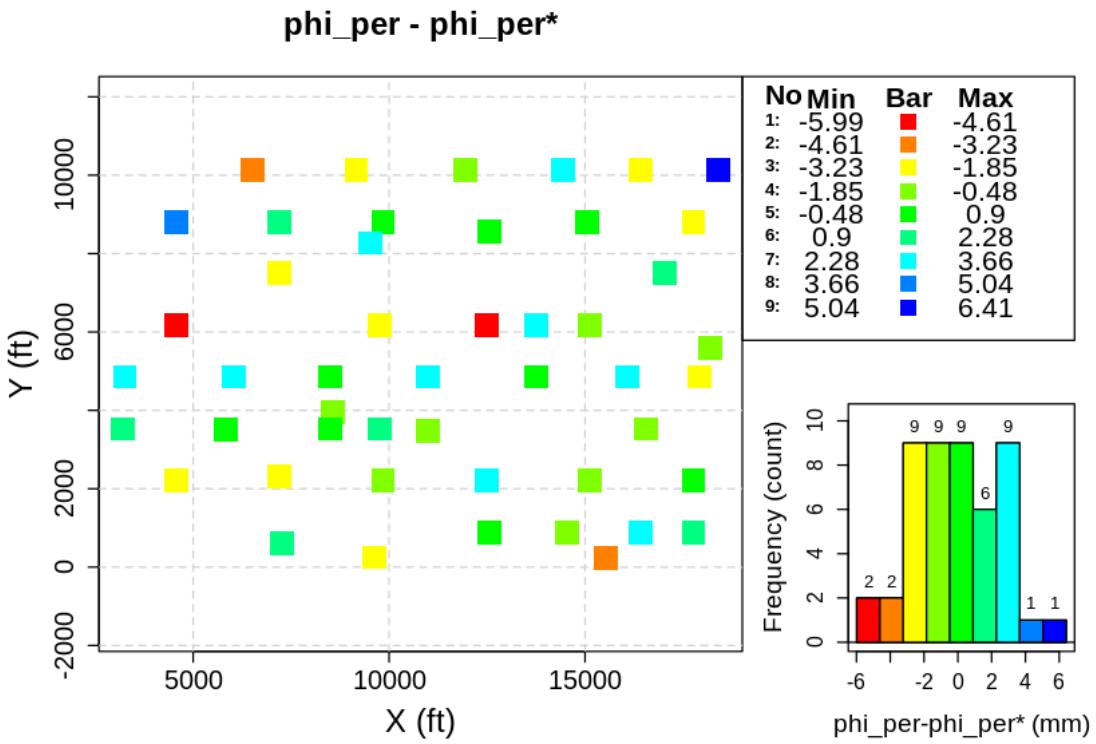
Respecto a los estadígrafos que obtenemos podemos notar que el valor esperado es cercano a cero, pero no lo suficiente para decidir si es un buen ajuste. Mientras que la varianza no es lo suficientemente pequeña.

```
[121]: HistBoxplot(x=phi_per_CrossValid[,5], mean = phi_per_CrossValid_Stat[5,3],
                  median = phi_per_CrossValid_Stat[4,3], main ="",
                  xlab = "phi_per-phi_per* (mm)", ylab = "Frecuencia Absoluta",
                  conteo",
                  AbsFreq = TRUE, PercentFreq = FALSE )
```



Según el boxplot mostrado sobre el histograma, no hay valores atípicos. Ahora hay que analizar si hay valores atípicos espaciales.

```
[122]: DEspacial(phi_per_CrossValid[,1], phi_per_CrossValid[,2],  
    ↪phi_per_CrossValid[,5], n_bins=9,  
    'X (ft)', 'Y (ft)', 'phi_per-phi_per* (mm)', 'phi_per - phi_per*')
```



En el histograma se muestran tres posibles valores atípicos: dos rojos y uno azul. La muestra azul está en la frontera, por lo que es difícil decidir si este es un valor atípico. En cambio, las muestras marcadas en rojo si pudieran considerarse como valores atípicos, ya que las muestras vecinas tienen valores muy diferentes.

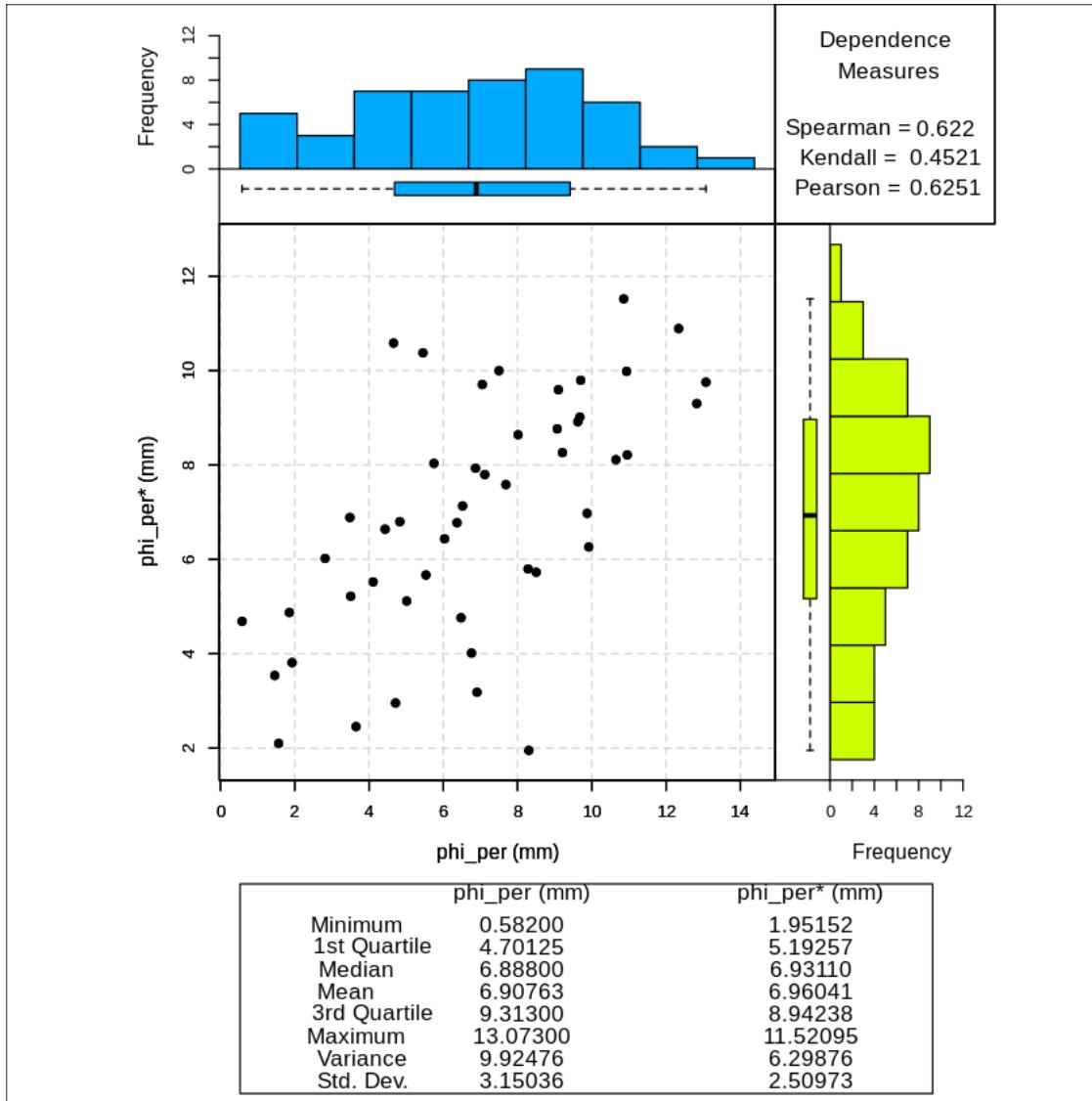
```
[123]: # phi_per is the independent variable
X<-phi_per_CrossValid[,3]
# phi_per* is the dependent variable
Y<-phi_per_CrossValid[,4]
```

```
[124]: ScatterPlot(phi_per, phi_per_CrossValid[, 4], 9,
                  Xmin = phi_per_Stat[2,2], Xmax = phi_per_Stat[7,2],
```

```

Ymin = phi_per_CrossValid_Stat[2,2], Ymax =_
↪phi_per_CrossValid_Stat[7,2],
XLAB = "phi_per (mm)", YLAB = "phi_per* (mm)"

```

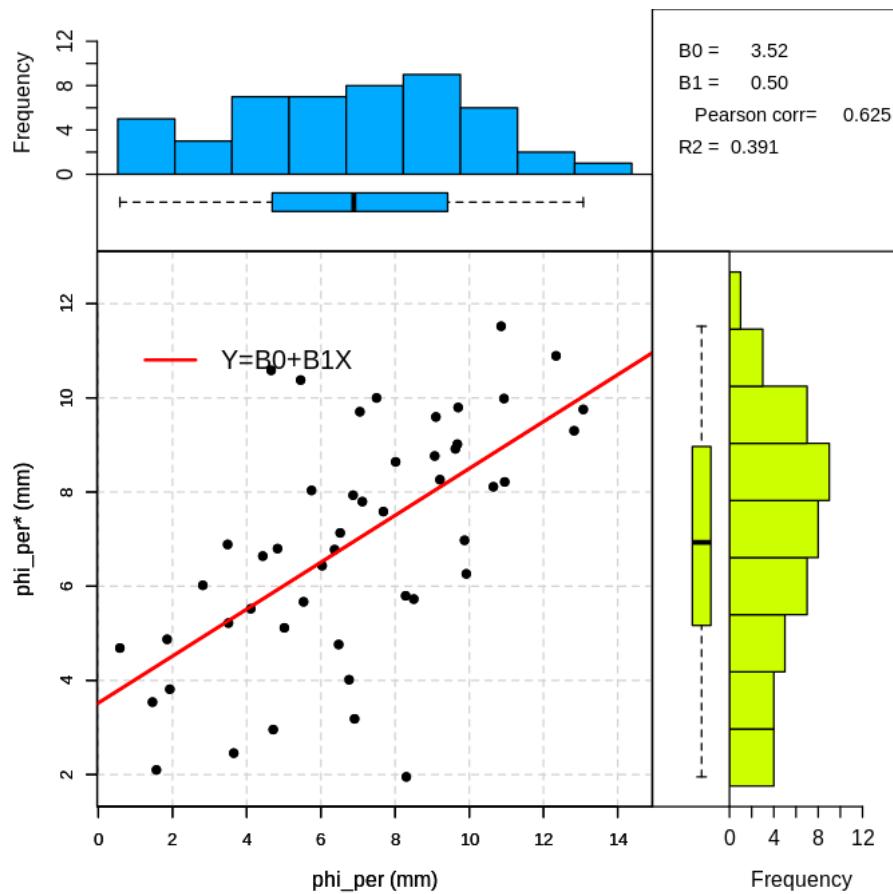


Respecto a su grafico de dispersión podemos notar que los valores no tienen una buena dependencia, por lo tanto podemos considerar que la aproximación no es buena, pero podria ser aceptable.

```

[125]: scatterplotReg(phi_per, phi_per_CrossValid[,4], 9,
                      Xmin = phi_per_Stat[2,2], Xmax = phi_per_Stat[7,2],
                      Ymin = phi_per_CrossValid_Stat[2,2], Ymax =_
↪phi_per_CrossValid_Stat[7,2],
                      XLAB = "phi_per (mm)", YLAB = "phi_per* (mm)")

```



## 5 Estimación espacial

Ahora que ya tenemos el análisis variográfico, haremos la estimación espacial usando el método de kriging ordinario y cokriging.

Al igual que la sección de análisis variográfico, crearemos una nueva carpeta donde se guardarán las gráficas que obtengamos usando la siguiente línea.

```
[126]: dir.create(paste(getwd(),"/Results/EstimacionEspacial", sep=""))
```

```
Warning message in dir.create(paste(getwd(), "/Results/EstimacionEspacial", sep = "")):  

'"/home/danielvr/Dropbox/Semestre 2023-1/ejemplo GAERM/ejemplo  

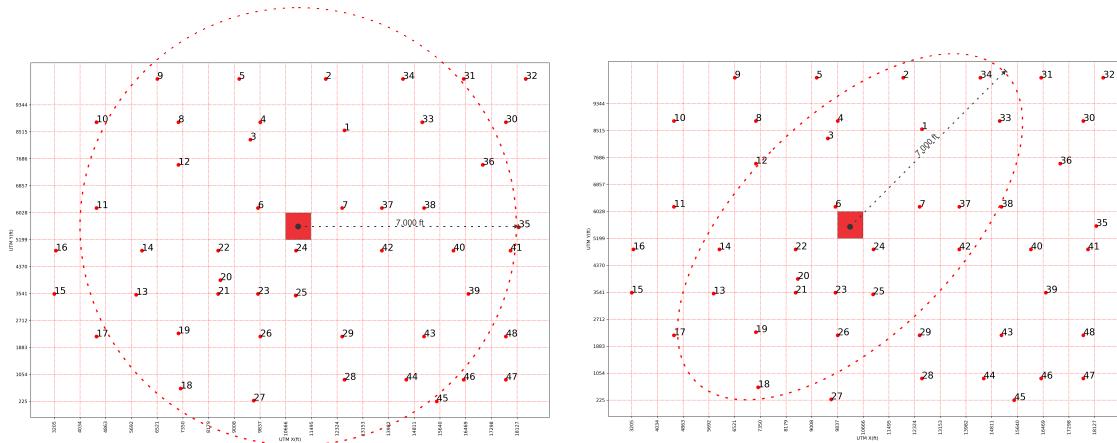
GAERM/Results/EstimacionEspacial' already exists"
```

## 5.1 Estimación espacial usando kriging ordinario en la variable perm\_mD\_Log

Si seguimos los pasos que vimos en clase sobre los aspectos prácticos del Kriging (presentación CG6 a partir de la diapositiva 36), necesitamos tres elementos para hacer la estimación usando kriging ordinario. La primera es establecer la **malla de estimación**, la cual se recomienda debe ser aproximadamente igual a la distancia mínima de separación. En este ejercicio esa distancia ya se calculó en el análisis variográfico y está almacenada en la variable “DistMin”.

La segunda es establecer la **vecindad de búsqueda**, esta nos permite determinar los puntos vecinos potenciales para la estimación. Esta distancia se determina tomando el alcance del variograma, sin embargo, como se vio en el análisis variográfico, el variograma puede ser isotrópico o anisotrópico.

- Para el caso isotrópico (siguiente imagen a la izquierda): tomar una circunferencia con centro en el punto a estimar y radio igual o menor al alcance del variograma.
- Para el caso anisotrópico (siguiente imagen a la derecha): tomar una elipse con centro en el punto a estimar y semiejes iguales o menores a los alcances del variograma anisotrópico.



En nuestro caso usamos un variograma adireccional que consideramos isotrópico, por lo tanto, los valores que necesitamos del variograma son los siguientes:

```
[127]: perm_mD_Log_vario_model<- 2  
perm_mD_Log_nugget<- 0  
perm_mD_Log_sill_and_nugget<- 3.6  
perm_mD_Log_rank <- 7500
```

La tercera condición es establecer el **número de puntos mínimo y máximo** para hacer la estimación.

Recordando los aspectos prácticos del kriging podemos establecer que:

- mínimo de puntos entre 4 y 6
- máximo de puntos entre 10 y 25

En este caso el intervalo se estableció entre 4 y 10, los cuales se ingresan en las variables “minPoints” para mínimo de puntos y “maxPoints” para máximo de puntos

```
[128]: minPoints<-4
maxPoints<-10
```

Ya que ingresamos los parámetros iniciales del variograma y número de puntos procedemos a hacer la estimación usando kriging ordinario. Esto lo hacemos usando la función “KrigingOrd”. Esta función se estructura de la siguiente forma:

- Las coordenadas (CoorX, CoorY), la variable a usar (Prop1) que en este caso son las muestras obtenidas en de la permeabilidad con transformación logarítmica (perm\_mD\_Log) (NOTA: el vector debe contener toda la muestra, incluidos los valores atípicos. Si el variograma fue estimado usando muestras transformadas, entonces Prop1 debe usar las muestras transformadas)
- La información del variograma, que en este caso es el modelo (perm\_mD\_Log\_vario\_model), valor del nugget (perm\_mD\_Log\_nugget), valor de la meseta más el nugget (perm\_mD\_Log\_sill\_and\_nugget) y el alcance (perm\_mD\_Log\_rank)
- El número de puntos mínimo (minPoints) y máximo (maxPoints) que previamente establecimos.
- Definimos el tamaño del espacio de trabajo. Para el eje X usamos (Xmin, Xmax) y para el eje Y usamos (Ymin, Ymax), estos valores los encontramos en los estadígrafos que calculamos de las coordenadas.
- Establecemos el tamaño de la celda, esto lo hacemos con las variables (TX,TY), como podemos observar, el tipo de celda en la siguiente función está determinada por la distancia mínima “DistMin” en ambas direcciones para que obtengamos celdas cuadradas ya que esa es la resolución mínima que tenemos. Si lo deseamos podemos poner diferentes distancias, sin embargo se debe justificar esa decisión, ya que podemos causar la creación de artefactos.
- Indicamos el tipo de transformación que tiene la variable que estamos usando. El parámetro invertir transformada “InvT” puede considerar tres tipos de transformaciones para asimetrías positivas: 1 si es logarítmica, 2 si es raíz cuadrada y 3 si es inversa. si la variable no tiene transformación o se usó otro tipo de transformación solo ponemos 0 (NOTA: en caso de usar otro tipo de transformación, las muestras obtenidas en la estimación kriging deben ser transformadas según la función. Por ejemplo, si la muestra fue diagnosticada con asimetría negativa y fue tratada con una transformación de potencia al cuadrado, entonces las muestras obtenidas con la estimación kriging deben transformarse usando la función raíz cuadrada, la cual se considera como función inversa de la potencia al cuadrado).
- Por último ingresamos las leyendas del gráfico: para el eje X (NameX), para el eje Y “NameY”, el título de la imagen resultante del kriging “Titulo1” y el título de la imagen obtenida de la desviación estándar (Titulo2).

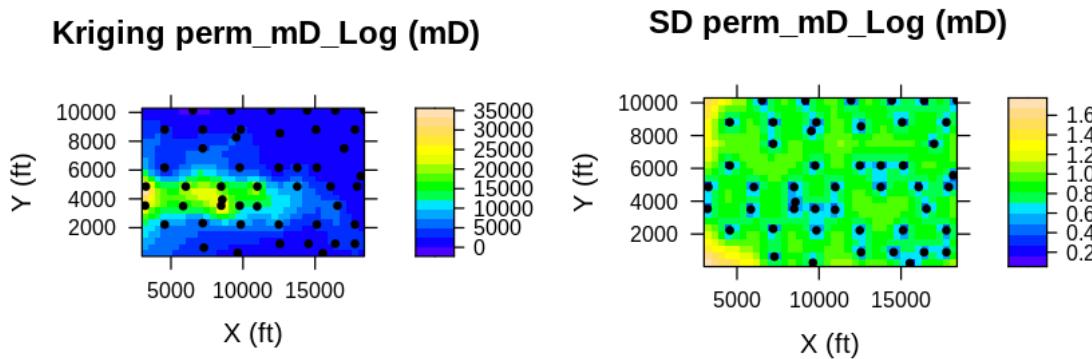
```
[129]: perm_mD_Log_OrdKrig <- KrigingOrd(CoordX = XCoord, CoordY = YCoord,
                                         Prop1 = perm_mD_Log, Modelo = u
                                         perm_mD_Log_vario_model, Nugget = perm_mD_Log_nugget,
                                         SillYNugget = perm_mD_Log_sill_and_nugget, Alcanceu
                                         = perm_mD_Log_rank,
                                         minPar = minPoints, maxPar = maxPoints,
                                         Xmin = XCoord_Stat[2,2], Xmax = XCoord_Stat[7,2],
```

```

Ymin = YCoord_Stat[2,2], Ymax = YCoord_Stat[7,2],
TX=DistMin, TY=DistMin, InvT=1, NameX="X (ft)",
NameY="Y (ft)", Titulo1="Kriging perm_mD_Log (mD)",
Titulo2="SD perm_mD_Log (mD)")

```

[using ordinary kriging]



El resultado es la imagen anterior, a la izquierda tenemos la estimación que se obtuvo usando kriging ordinario y a la derecha su desviación estándar, si analizamos estas imágenes podemos notar que las estimaciones con la menor desviación estándar se encuentran en el centro, esto se debe a su abundancia de puntos vecinos, mientras que las estimaciones con desviación estándar alta se encuentran a las orillas debido a la pobre distribución de muestras en esas zonas, si ejecutamos la línea “perm\_mD\_Log\_OrdKrig” en la consola, obtenemos una tabla dividida en cuatro columnas con la información de la estimación: la columna V1 muestra las coordenadas en X, la columna V2

muestra las coordenadas en Y, la columna V3 muestra los valores estimados por kriging ordinario y la columna V4 muestra los valores de la desviación estándar del valor estimado.

[130] : perm\_mD\_Log\_OrdKrig

V1	V2	V3	V4
3205.000	225	6564.1583	1.6712848
3633.439	225	6034.4106	1.6002744
4061.878	225	5948.7815	1.5319824
4490.317	225	5454.4003	1.4690111
4918.756	225	5028.7537	1.4091211
5347.195	225	4664.2566	1.3439834
5775.634	225	4346.3074	1.2608864
6204.073	225	4077.1886	1.1447719
6632.512	225	3813.3630	0.9811438
7060.951	225	3586.5525	0.7841697
7489.390	225	3230.9791	0.7557741
7917.829	225	2763.0054	0.8855367
8346.268	225	2359.7309	0.9447951
8774.707	225	1956.6776	0.8989154
9203.146	225	1558.6199	0.7158846
9631.585	225	1208.6969	0.1778635
10060.024	225	1274.3155	0.7283378
10488.463	225	1313.8251	0.9459075
10916.902	225	1298.4450	1.0487985
11345.342	225	1187.1949	1.0711274
11773.781	225	1065.2374	1.0294119
12202.220	225	903.3535	0.9479946
12630.659	225	791.1460	0.9045890
13059.098	225	712.7450	0.9457558
13487.537	225	644.2580	0.9782553
13915.976	225	594.8612	0.9461487
14344.415	225	532.5467	0.8607214
14772.854	225	478.4102	0.7727386
15201.293	225	437.7921	0.6096990
15629.732	225	481.6525	0.3560890
:	:	:	:
5775.634	10079.1	47.81297	0.8964469
6204.073	10079.1	18.53871	0.6422551
6632.512	10079.1	10.84300	0.4060778
7060.951	10079.1	23.60878	0.7563706
7489.390	10079.1	43.01856	0.8975176
7917.829	10079.1	63.87540	0.9367294
8346.268	10079.1	82.15429	0.8789415
8774.707	10079.1	93.83829	0.6852337
9203.146	10079.1	103.26074	0.3137633
9631.585	10079.1	173.19694	0.7225343
10060.024	10079.1	252.84818	0.8914814
10488.463	10079.1	318.10061	0.9580595
10916.902	10079.1	358.91780	0.9388612
11345.342	10079.1	371.33075	0.8138733
11773.781	10079.1	365.61521	0.5019751
12202.220	10079.1	415.37471	0.5650339
12630.659	10079.1	528.83209	0.8228619
13059.098	10079.1	657.02466	0.9181090
13487.537	10079.1	811.85729	0.9038935
13915.976	10079.1	1024.71240	0.7670517
14344.415	10079.1	1170.05315	0.3988932

A matrix: 864 × 4 of type dbl

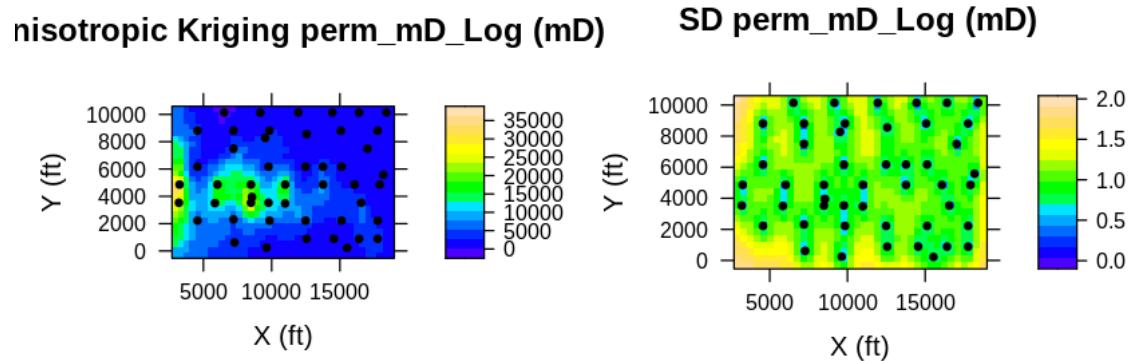
En caso de que usemos un variograma anisotrópico debemos usar la función “KrigingOrdAnis”, esta función necesita los siguientes parámetros:

- Las coordenadas (CoorX, CoorY), la variable a usar (Prop1) que en este caso son las muestras obtenidas de la permeabilidad con transformación logarítmica (perm\_mD\_Log)
- La información del variograma, que en este caso es el modelo (perm\_mD\_Log\_vario\_model), valor del nugget (perm\_mD\_Log\_nugget), valor de la meseta más el nugget (perm\_mD\_Log\_sill\_and\_nugget) y el alcance (perm\_mD\_Log\_rank)
- El número de puntos mínimo (minPoints) y máximo (maxPoints) que previamente establecimos.
- Establecemos el tamaño de la celda, esto lo hacemos con las variables (malla), en este caso la función no permite ser flexible con las medidas de la celda, por lo que siempre será cuadrada.
- Indicamos el tipo de transformación que tiene la variable que estamos usando. El parámetro “InvT” puede considerar tres tipos de transformaciones positivas: 1 si es logarítmica, 2 si es raíz cuadrada y 3 si es inversa. si la variable no tiene transformación solo ponemos 0.
- Indicamos los valores del variograma anisotrópico: ángulo de máxima anisotropía (MaxAnis) y su proporción (proporcion). Estos valores son calculados en el análisis variográfico.
- Por último ingresamos las leyendas del gráfico: para el eje X (NameX), para el eje Y “NameY”, el título de la imagen resultante del kriging “Titulo1” y el título de la imagen obtenida de la desviación estándar (Titulo2).

```
[131]: perm_mD_Log_OrdKrigAnis <- KrigingOrdAnis(CoorX = XCoord, CoorY = YCoord,
                                                 Prop1 = perm_mD_Log, Modelo = perm_mD_Log_vario_model, Nugget = ↴perm_mD_Log_nugget,
                                                 SillYNugget = perm_mD_Log_sill_and_nugget, Alcance = ↴perm_mD_Log_rank,
                                                 minPar = minPoints, maxPar = maxPoints,
                                                 malla = DistMin, InvT = 1, MaxAnis = 0, proporcion = 0.5,
                                                 NameX="X (ft)",
                                                 NameY="Y (ft)", Titulo1="Anisotropic Kriging perm_mD_Log (mD)",
                                                 Titulo2="SD perm_mD_Log (mD)")
```

[using ordinary kriging]

Warning message in sqrt(kriging\$var1.var):  
“NaNs produced”



## 6 Estimación espacial usando kriging ordinario en la variable phi\_per

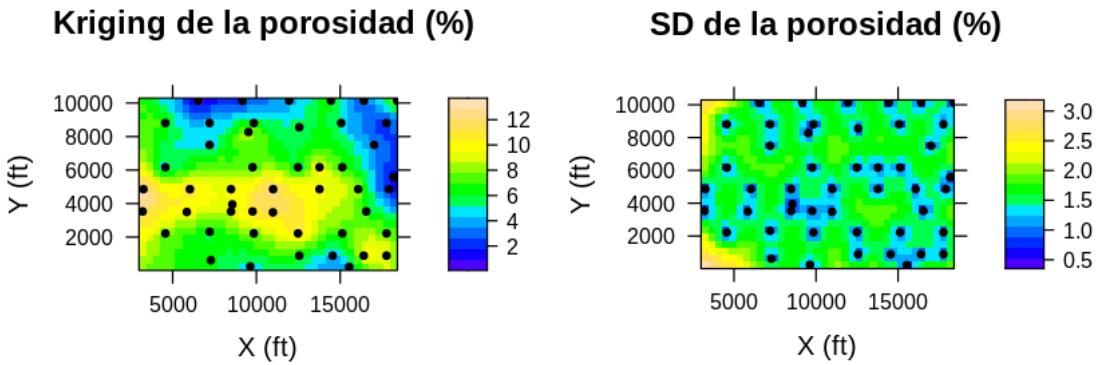
Al igual que la variable de permeabilidad, para la estimación espacial usando kriging ordinario para la variable phi\_per necesitamos los mismos elementos: dimensiones de la malla, variograma y número de puntos. El primer y tercer punto son iguales en ambas variables, solo cambia los valores del variograma, los cuales son los siguientes:

```
[132]: phi_per_vario_model<- 2
phi_per_nugget<- 0.1
phi_per_sill_and_nugget<- 11.6
phi_per_rank <- 7500
```

ya que tenemos los valores del variograma, usamos la función “KrigingOrd” y estimamos.

```
[133]: phi_per_OrdKrig <- KrigingOrd(CoorX = XCoord, CoorY = YCoord,
                                         Prop1 = phi_per, Modelo = u
                                         →phi_per_vario_model, Nugget = phi_per_nugget,
                                         SillyNugget = phi_per_sill_and_nugget,u
                                         →Alcance = phi_per_rank,
                                         minPar = minPoints, maxPar = maxPoints,
                                         Xmin = XCoord_Stat[2,2], Xmax = u
                                         →XCoord_Stat[7,2],
                                         Ymin = YCoord_Stat[2,2], Ymax = u
                                         →YCoord_Stat[7,2],
                                         TX=DistMin, TY=DistMin, InvT=0, NameX="X
                                         →(ft)",
                                         NameY="Y (ft)", Titulo1="Kriging de la
                                         →porosidad (%)"",
                                         Titulo2="SD de la porosidad (%)")
```

[using ordinary kriging]



Si analizamos la imagen resultante podemos ver la desviación estándar solo es buena cerca de los puntos donde se encuentran las muestras.

```
[134]: phi_per_OrdKrig
```

V1	V2	V3	V4
3205.000	225	7.878995	3.0120420
3633.439	225	7.603817	2.8865166
4061.878	225	7.650310	2.7658950
4490.317	225	7.354060	2.6544906
4918.756	225	7.091314	2.5484828
5347.195	225	6.855178	2.4334230
5775.634	225	6.667408	2.2873776
6204.073	225	6.527225	2.0846563
6632.512	225	6.444084	1.8015014
7060.951	225	6.364946	1.4650262
7489.390	225	6.183651	1.4147420
7917.829	225	5.903516	1.6320294
8346.268	225	5.642155	1.7321146
8774.707	225	5.412508	1.6531899
9203.146	225	5.140029	1.3436769
9631.585	225	4.864575	0.5461397
10060.024	225	5.165901	1.3664104
10488.463	225	5.450343	1.7361008
10916.902	225	5.657275	1.9134956
11345.342	225	5.656180	1.9525036
11773.781	225	5.649940	1.8812450
12202.220	225	5.403649	1.7420283
12630.659	225	5.185236	1.6675009
13059.098	225	4.883052	1.7360989
13487.537	225	4.504750	1.7901748
13915.976	225	4.146628	1.7339469
14344.415	225	3.830772	1.5866807
14772.854	225	3.616826	1.4347735
15201.293	225	3.593696	1.1622169
15629.732	225	4.087248	0.7675116
:	:	:	:
5775.634	10079.1	2.461093	1.6512646
6204.073	10079.1	1.506334	1.2216114
6632.512	10079.1	0.907196	0.8425091
7060.951	10079.1	1.384073	1.4082024
7489.390	10079.1	1.764146	1.6479957
7917.829	10079.1	2.011333	1.7155716
8346.268	10079.1	2.159306	1.6171840
8774.707	10079.1	2.187298	1.2906832
9203.146	10079.1	2.169885	0.7090057
9631.585	10079.1	2.697444	1.3519571
10060.024	10079.1	3.088645	1.6385247
10488.463	10079.1	3.348153	1.7531469
10916.902	10079.1	3.495507	1.7205818
11345.342	10079.1	3.571762	1.5077274
11773.781	10079.1	3.566151	0.9928946
12202.220	10079.1	3.913321	1.0926409
12630.659	10079.1	4.470321	1.5212314
13059.098	10079.1	5.020096	1.6836260
13487.537	10079.1	5.610328	1.6596474
13915.976	10079.1	6.185576	1.4275081
14344.415	10079.1	6.553755	0.8326028

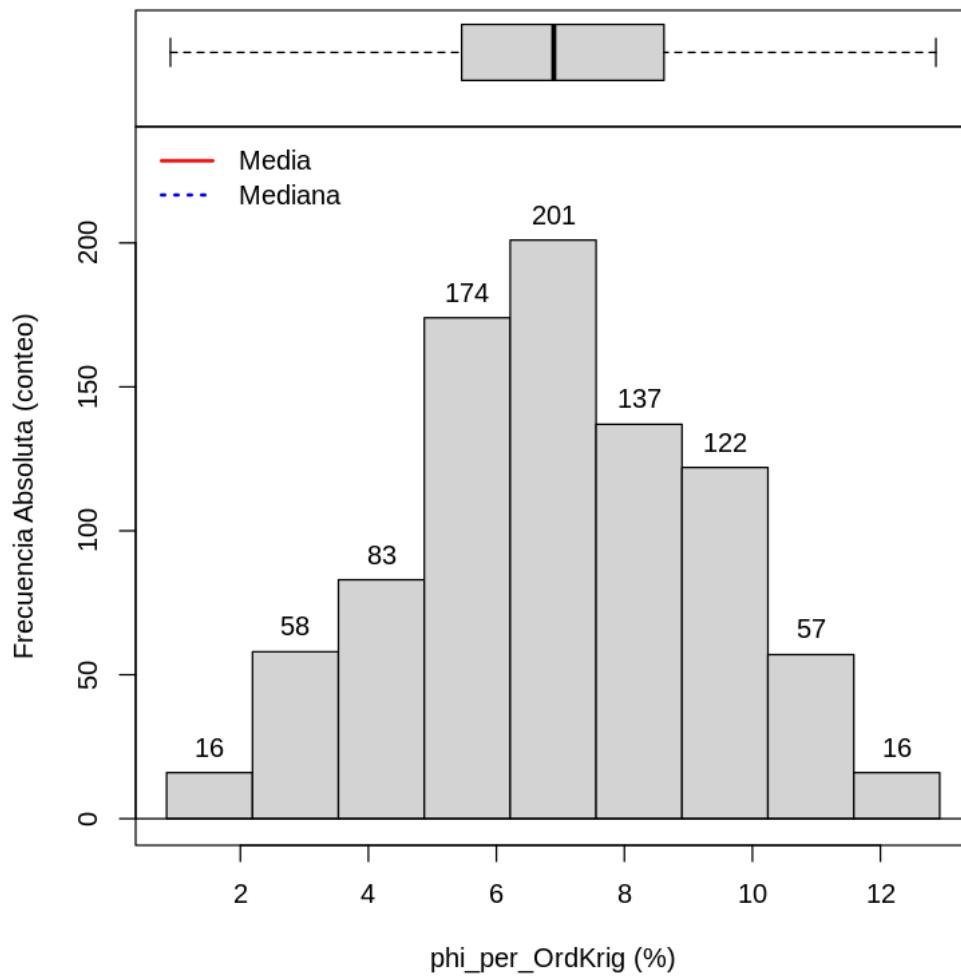
A matrix: 864 × 4 of type dbl

Calculamos los estadígrafos del kriging ordinario.

```
[135]: phi_per_OrdKrig_stat<-Estadisticas(phi_per_OrdKrig[,1])  
phi_per_OrdKrig_stat
```

	Statistics <chr>	Values <dbl>
	muestras	8.640000e+02
	minimos	3.205000e+03
	cuantiles1	6.953842e+03
	medianas	1.070268e+04
	medias	1.070268e+04
A data.frame: 14 × 2	cuantiles3	1.445152e+04
	maximos	1.820037e+04
	rangos	1.499537e+04
	rangosInt	7.497683e+03
	varianzas	1.983214e+07
	desvs	4.453329e+03
	CVs	4.161000e-01
	simetrias	0.000000e+00
	curtosiss	1.798100e+00

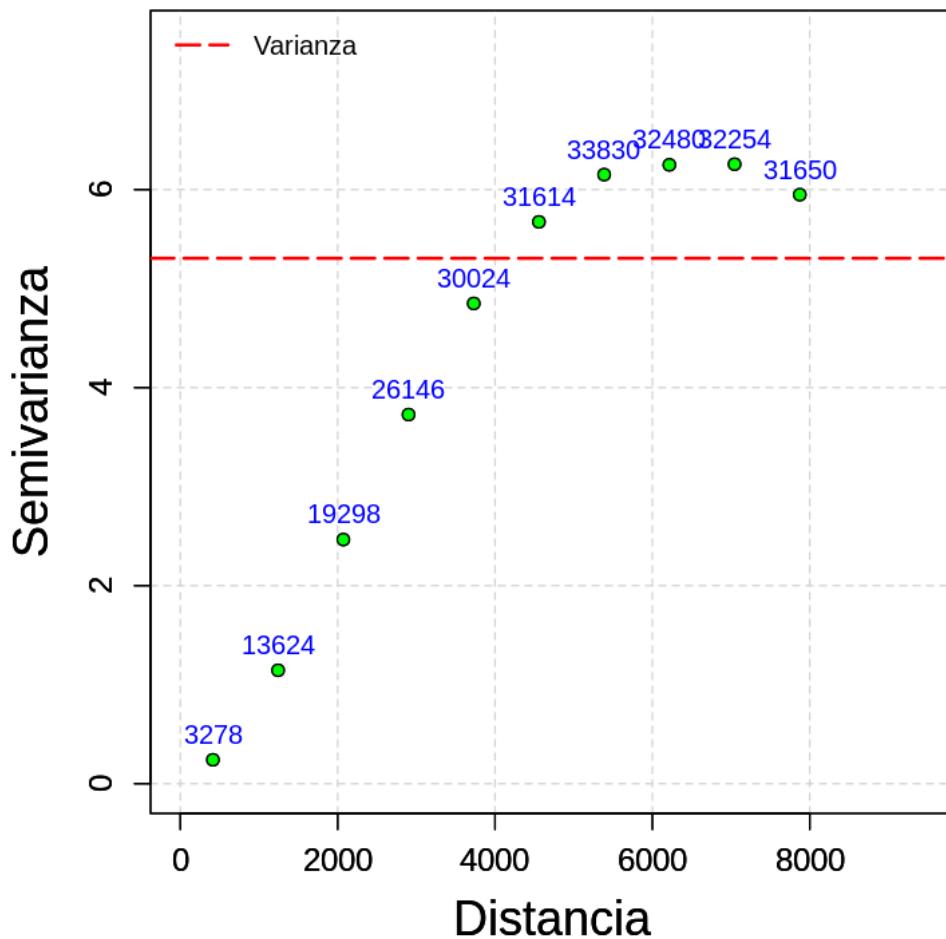
```
[136]: HistBoxplot(x=phi_per_OrdKrig[,3], mean = phi_per_OrdKrig_stat[5,2], median =  
                   phi_per_OrdKrig_stat[4,2],  
                   main = "",  
                   xlab = "phi_per_OrdKrig (%)", ylab = "Frecuencia Absoluta (conteo)",  
                   AbsFreq = TRUE,  
                   PercentFreq = FALSE,  
                   nbin = 9)
```



```
[137]: phi_per_ordikrig_1_Log_VarioEstimation<-Variograma(phi_per_OrdKrig[,1],  
          ↪phi_per_OrdKrig[,2],  
          phi_per_OrdKrig[,3], 0, 90, 1*N_lags,  
          ↪lag_value, 1,  
          "Variograma Adireccional de  
          ↪porosidad kriging (%)")
```

variog: computing omnidirectional variogram

### Variograma Adireccional de porosidad kriging (%)

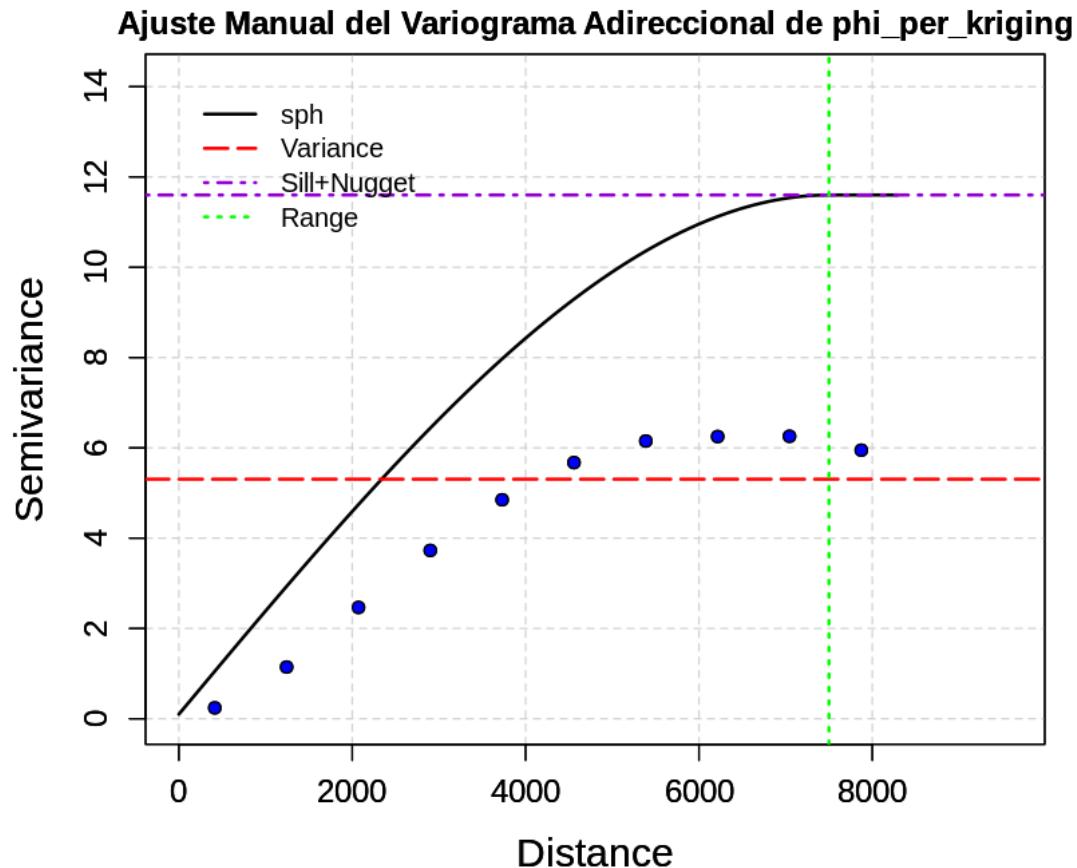


```
[138]: phi_per_vario_model<- 2
phi_per_nugget<- 0.1
phi_per_sill_and_nugget<- 11.6
phi_per_rank <- 7500

phi_per_krig_EyeModelVarioFit<-EyeModel(phi_per_OrdKrig[,1], phi_per_OrdKrig[,2],
                                             phi_per_OrdKrig[,3], 0, 90, N_lags,
                                             lag_value, 1,
                                             phi_per_vario_model, phi_per_nugget,
                                             phi_per_sill_and_nugget,
                                             phi_per_rank,
```

"Ajuste Manual del Variograma Adireccional  
de phi\_per\_kriging")

variog: computing omnidirectional variogram



## 6.1 Estimación espacial usando Co-kriging ordinario

### 6.1.1 Análisis variográfico bivariado

Para obtener la estimación por co-kriging debemos hacer un análisis variográfico bivariado. Si revisamos lo aprendido en la clase, la manera más aceptada es usar un modelo de corregralización lineal.

Este modelo consiste en:

- Modelar cada semivariograma simple y semivariograma cruzado individualmente
- Determinar el número de estructuras anidadas de manera que sea mínimo (es deseable que sea cuanto más tres)
- Comprobar que todos los determinantes de los menores de orden dos son no negativos.
- Verificar que todas las matrices de correacionalización sean positivas semidefinidas, en caso contrario hacer los cambios necesarios hasta satisfacer la condición o volver al paso 2.

Para cumplir con el primer y segundo paso vamos a estimar tres variogramas dos para cada variable (porosidad y permeabilidad) y el variograma cruzado.

Comenzamos con definir los parámetros básicos del semivariograma: número de intervalos(N\_lags), distancia mínima (DistMin), distancia máxima (DistMax) y el valor del intervalo (lag\_value)

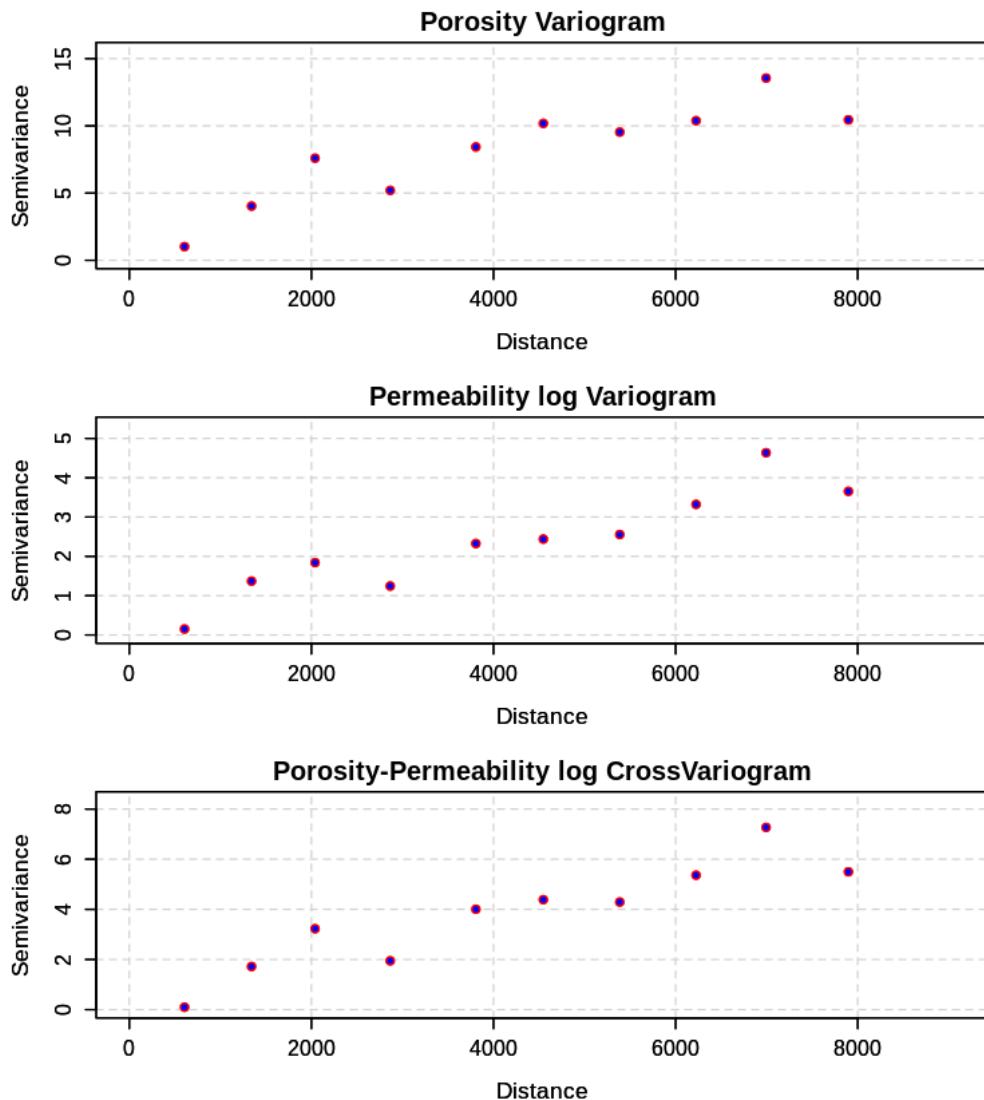
```
[140]: N_lags<-10
DistMin<-min(dist(Data_File_Burb[,1:2])) # Minimum distance in data
DistMax<-max(dist(Data_File_Burb[,1:2])) # Maximum distance in data
lag_value<-max((DistMax/2)/N_lags, DistMin)
```

Para estimar el variograma cruzado usamos la función “CrossVariograma”, esta función necesita los siguientes parámetros:

- Las coordenadas (CoorX, CoorY), las variables a usar (P1,P2) que en este caso son las muestras obtenidas de la permeabilidad con transformación logarítmica (perm\_mD\_Log) y la porosidad (phi\_per)
- Los parámetros para estimar el semivariograma: número de intervalos (NInt), el valor del intervalo (lags), 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.
- Títulos de las imágenes para la primera variable (NomP1), segunda variable (NomP2) y variograma cruzado (NomP1P2)

**Nota:** Es importante mencionar que en caso de que la dependencia de las variables sea negativa, entonces una de las variables debe ser multiplicada por -1, de lo contrario el modelo de correacionalización dará valores negativos

```
[159]: phi_per_perm_mD_Log_CrossVario<-CrossVariograma(CoorX = XCoord, CoorY = YCoord,
                                                       P1 = phi_per, P2 = perm_mD_Log,
                                                       NInt = N_lags, lags = lag_value,
                                                       Direccion = 0, Tol = 90,
                                                       NomP1 = 'Porosity Variogram',
                                                       NomP2 = 'Permeability log
                                                               ↪Variogram',
                                                       NomP1P2 = 'Porosity-Permeability
                                                               ↪log CrossVariogram')
```



Esto nos da como resultado una imagen donde podemos ver tres variogramas: el de la variable phi\_per, variable perm\_mD\_Log y el variograma cruzado.

Para estimar el modelo de variograma debemos ingresar los datos de cada modelo, para cada variograma simple iniciamos conservando el valor de la meseta y el nugget, el alcance y modelo de variograma serán determinados por la variable que nos interesa estimar mediante co-kriging, en este caso perm\_mD\_Log.

```
[142]: # porosity
phi_per_vario_model<- 2
phi_per_nugget<- 0.1
phi_per_sill_minus_nugget<- 11.6
phi_per_rank <- 7500
```

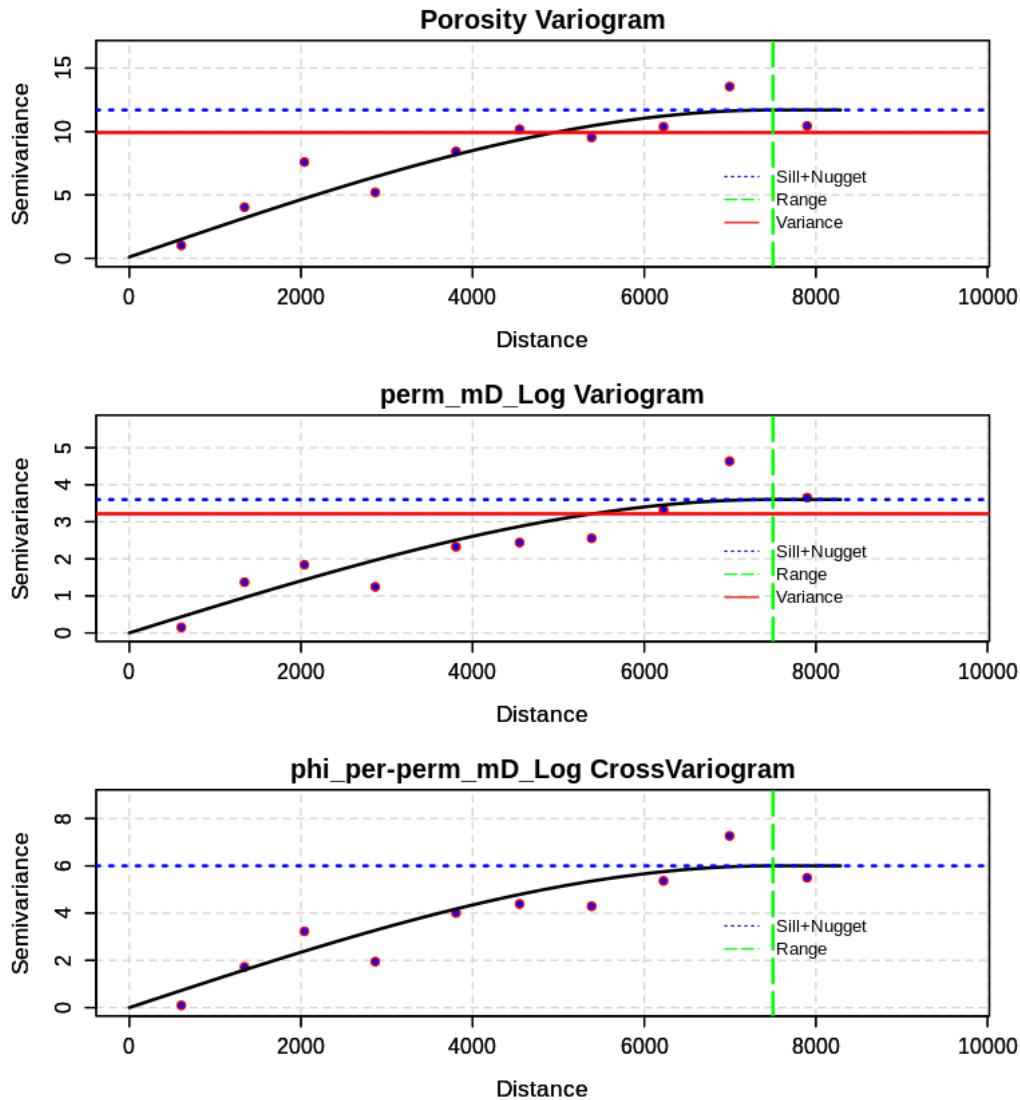
```
[152]: # permeability
perm_mD_Log_vario_model<- 2
perm_mD_Log_nugget<- 0.001
perm_mD_Log_sill_minus_nugget<- 3.6
perm_mD_Log_rank <- 7500
```

```
[155]: # Cross-variogram
phi_per_perm_mD_Log_vario_model<- 2 # spherical
phi_per_perm_mD_Log_nugget<- 0
phi_per_perm_mD_Log_sill_minus_nugget<- 6
phi_per_perm_mD_Log_rank <- 7500
```

Para obtener el modelo del variograma cruzado usamos la función “ModelVariogram”, esta función necesita los siguientes parámetros:

- Las coordenadas (CoorX, CoorY), las variables a usar (phi\_per, perm\_mD\_Log) que en este caso son las muestras obtenidas de la permeabilidad con transformación logarítmica (perm\_mD\_Log) y las muestra de la porosidad (phi\_per)
- Los parámetros para estimar el semivariograma: número de intervalos (N\_lags), el valor del intervalo (lag\_value), 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.
- Los valores de cada variograma que colocamos en las líneas anteriores (phi\_per\_perm\_mD\_Log\_vario\_model, phi\_per\_sill\_minus\_nugget, perm\_mD\_Log\_sill\_minus\_nugget, phi\_per\_perm\_mD\_Log\_sill\_minus\_nugget, phi\_per\_nugget, perm\_mD\_Log\_nugget, phi\_per\_perm\_mD\_Log\_nugget, phi\_per\_perm\_mD\_Log\_rank)
- Títulos para cada variograma ('phi\_per Variogram', 'perm\_mD\_Log Variogram', 'phi\_per\_perm\_mD\_Log CrossVariogram')

```
[160]: ModelVariogram(XCoord, YCoord,
                      phi_per, perm_mD_Log,
                      N_lags, lag_value, 0, 90,
                      phi_per_perm_mD_Log_vario_model,
                      phi_per_sill_minus_nugget, perm_mD_Log_sill_minus_nugget,
                      phi_per_perm_mD_Log_sill_minus_nugget, phi_per_nugget,
                      perm_mD_Log_nugget,
                      phi_per_perm_mD_Log_nugget, phi_per_perm_mD_Log_rank,
                      'Porosity Variogram', 'perm_mD_Log Variogram',
                      'phi_per-perm_mD_Log CrossVariogram')
```



Ya que estimamos los variogramas, debemos verificar que los determinantes del nugget y meseta sean positivas semidefinidas, es decir

$$\begin{pmatrix} \gamma_{PP}(h) & \gamma_{PR}(h) \\ \gamma_{RP}(h) & \gamma_{RR}(h) \end{pmatrix} = \begin{pmatrix} 0.001 & 0.001 \\ 0.001 & 0.1 \end{pmatrix} \gamma_0(h) + \begin{pmatrix} 3.6 & 6 \\ 6 & 11.6 \end{pmatrix} \gamma_1(h)$$

Para hacer esto usamos la función “det” para cada caso, la cual requiere que los valores sean ingresados en una matriz.

```
[157]: NuggetMatrix <- matrix(c(perm_mD_Log_nugget,
                                phi_per_perm_mD_Log_nugget,
                                phi_per_perm_mD_Log_nugget,
                                phi_per_nugget), ncol = 2)
```

```
det(NuggetMatrix)
```

1e-04

$$\det \begin{pmatrix} 0.001 & 0.001 \\ 0.001 & 0.1 \end{pmatrix} = 0.0001 > 0$$

el determinante es positivo

```
[158]: SemiVariMatrix1 <- matrix(c(perm_mD_Log_sill_minus_nugget,
                                    phi_per_perm_mD_Log_sill_minus_nugget,
                                    phi_per_perm_mD_Log_sill_minus_nugget,
                                    phi_per_sill_minus_nugget), ncol = 2)
det(SemiVariMatrix1)
```

5.76

$$\det \begin{pmatrix} 3.6 & 6 \\ 6 & 11.6 \end{pmatrix} = 5.76 > 0$$

el determinante es positivo

ya que sabemos que los determinantes son positivos semidefinidos debemos hacer la validación cruzada, al igual que en el análisis variográfico, el método consiste en estimar por Cokriging los valores en los puntos muestrales usando el procedimiento de leave one out.

Esto lo podemos hacer usando la función “CrossValidation2”. la cual necesita de los siguientes parámetros:

- Las coordenadas (XCoor, YCoor), las variables a usar (phi\_per, perm\_mD\_Log) que en este caso son las muestras obtenidas de la permeabilidad con transformación logarítmica (perm\_mD) y la porosidad (phi\_per)
- Los parámetros para estimar el semivariograma: número de intervalos (N\_lags), el valor del intervalo (lag\_value)
- Los valores de nugget, meseta, alcance y modelo de cada variograma simple y cruzado.

```
[161]: perm_mD_Log_CrossValid2<- CrossValidation2(XCoord, YCoord,
                                                    phi_per, perm_mD_Log,
                                                    N_lags, lag_value,
                                                    phi_per_perm_mD_Log_vario_model,
                                                    phi_per_sill_minus_nugget, u
                                                    ↵perm_mD_Log_sill_minus_nugget,
                                                    ↵phi_per_nugget, perm_mD_Log_nugget,
                                                    ↵phi_per_perm_mD_Log_rank
                                                    )
```





[162]: perm\_mD\_Log\_CrossValid2

	X <dbl>	Y <dbl>	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
1	12561	8558	5.533	6.3469665	-0.81396655
2	11955	10140	3.508	3.4501054	0.05789456
3	9527	8273	8.503	6.4296901	2.07330990
4	9845	8810	6.034	7.8794102	-1.84541015
5	9165	10140	1.929	1.4874549	0.44154508
6	9770	6170	7.051	7.5804773	-0.52947730
7	12485	6170	5.454	6.4800874	-1.02608737
8	7205	8810	4.715	6.3707735	-1.65577346
9	6525	10141	0.582	-3.8186439	4.40064388
10	4565	8810	6.908	8.2296883	-1.32168831
11	4565	6170	4.660	6.7152771	-2.05527709
12	7205	7500	4.434	6.3926386	-1.95863857
13	5845	3500	9.678	9.0666400	0.61135998
14	6029	4860	10.955	9.0050946	1.94990541
15	3205	3530	12.339	11.7769743	0.56202567
16	3255	4860	13.073	11.5477875	1.52521249
17	4565	2220	7.500	9.0464074	-1.54640735
18	7280	620	6.479	5.9017875	0.57721250
19	7205	2316	5.748	7.0270703	-1.27907031
20	8559	3952	9.100	9.0638210	0.03617901
21	8485	3530	9.068	10.3313896	-1.26338957
22	8485	4860	9.625	9.2873687	0.33763127
23	9770	3530	10.936	9.2770596	1.65894042
24	10995	4860	12.826	11.5092632	1.31673680
25	10985	3477	10.859	11.1317403	-0.27274028
26	9845	2220	8.015	9.1752781	-1.16027806
27	9633	247	4.831	5.0232839	-0.19228393
28	12560	890	6.370	5.9850838	0.38491625
29	12485	2220	10.650	9.3057415	1.34425845
30	17765	8810	1.856	4.1073189	-2.25131891
31	16405	10140	2.818	5.5054609	-2.68746092
32	18405	10140	8.303	4.7382637	3.56473632
33	15065	8810	5.015	4.1222013	0.89279868
34	14445	10140	6.760	5.5141075	1.24589252
35	18180	5580	1.566	0.6046826	0.96131739
36	17020	7500	3.651	3.1514765	0.49952352
37	13765	6170	9.873	8.8974670	0.97553304
38	15125	6170	6.868	6.7633419	0.10465811
39	16555	3530	6.520	9.0658139	-2.54581389
40	16075	4860	8.284	6.1694949	2.11450515
41	17915	4860	1.461	2.3903694	-0.92936940
42	13765	4860	9.700	10.9361014	-1.23610139
43	15125	2220	7.118	7.5693710	-0.45137097
44	14552	890	4.110	5.1569319	-1.04693194
45	15535	225	3.486	4.6370473	-1.15104732
46	16405	890	9.920	7.7696959	2.15030414
47	17765	890	9.209	8.7641836	0.44481635
48	17765	2220	7.685	7.4463442	0.23865582

A data.frame: 48 × 5

El resultado es una tabla donde 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, estimando el valor con el método de co-kriging usando el variograma cruzado propuesto ( $Z^*$ ), la fila 5 es la diferencia entre la variable y los valores estimados ( $Z - Z^*$ )

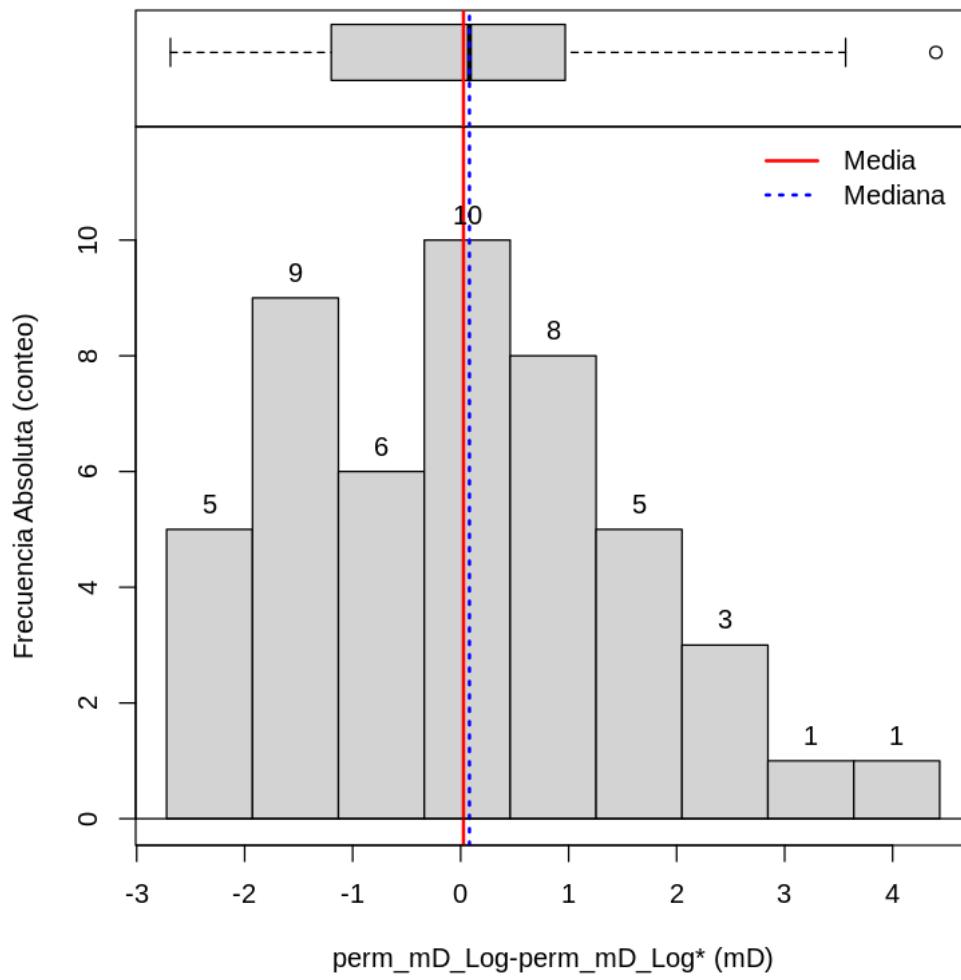
Y también podemos obtener sus estadígrafos usando la función “Val\_Estadisticos”

```
[163]: perm_mD_Log_CrossValid2_Sta <-  
        Val_Estadisticos(perm_mD_Log_CrossValid2[,c(3,4,5)])  
perm_mD_Log_CrossValid2_Sta
```

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	48.00000	48.00000	48.00000
Minimo	0.58200	-3.81864	-2.68746
Cuartil_1er	4.70125	5.41833	-1.17923
Mediana	6.88800	6.89521	0.08128
Media	6.90763	6.88157	0.02605
A data.frame: 13 × 3	Cuartil_3er	9.31300	9.06432
	Maximo	13.07300	11.77697
	Rango	12.49100	15.59562
	Rango_Intercuartil	4.61175	3.64599
	Varianza	9.92476	9.35499
	Desv_Estandar	3.15036	3.05859
	Simetria	-0.05236	-0.95268
	Curtosis	2.28307	4.70749

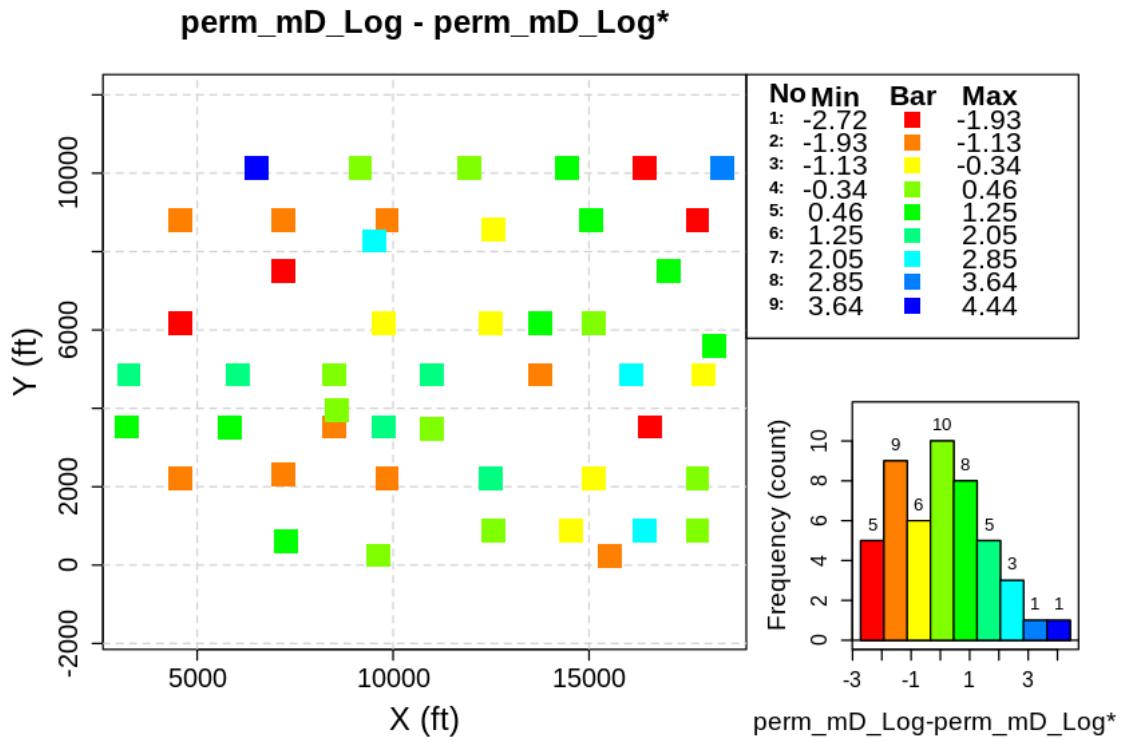
También graficamos su histograma

```
[165]: HistBoxplot(x=perm_mD_Log_CrossValid2[,5], mean =  
                    perm_mD_Log_CrossValid2_Sta[5,3],  
                    median = perm_mD_Log_CrossValid2_Sta[4,3],  
                    main = "", xlab = "perm_mD_Log-perm_mD_Log* (mD)", ylab = "Frecuencia",  
                    Absoluta (conteo),  
                    AbsFreq = TRUE, PercentFreq = FALSE )
```



Y evaluamos los posibles valores atípicos, a diferencia de la estimación por kriging, podemos notar que hay un valor atípico localizado a la derecha del histograma y su ubicación espacial (puntos rojos) es muy distinta.

```
[167]: DEspacial(perm_mD_Log_CrossValid2[,1], perm_mD_Log_CrossValid2[,2],  
    ↪perm_mD_Log_CrossValid2[,5], n_bins=9,  
    'X (ft)', 'Y (ft)', 'perm_mD_Log - perm_mD_Log* (mD)', 'perm_mD_Log -  
    ↪perm_mD_Log*')
```



### 6.1.2 Estimación usando co-kriging ordinario para la variable perm\_mD\_Log

Ya que tenemos el modelo de corregionalización podemos hacer la estimación usando co-kriging ordinario. Para hacerlo usamos la función “CoKrigingOrd”

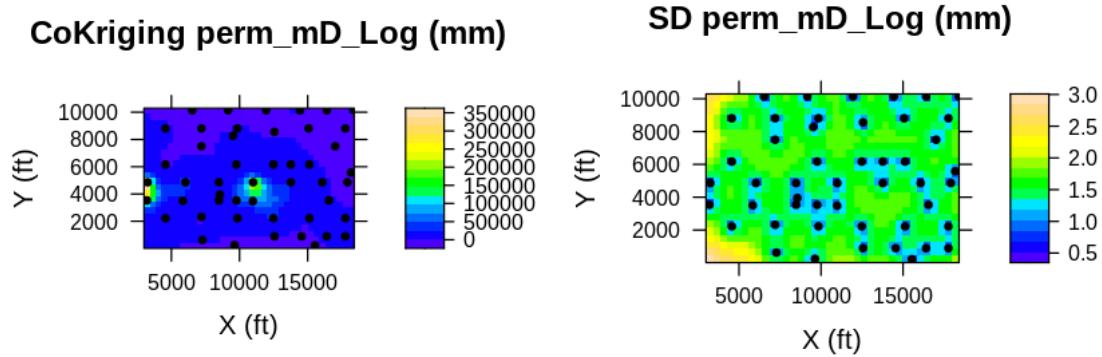
- Las coordenadas (CoorX, CoorY), las variables a usar (phi\_per, perm\_mD\_Log) que en este caso son las muestras obtenidas de la permeabilidad con transformación logarítmica (perm\_mD\_Log) y la porosidad (phi\_per)
- La información de los variogramas simples y cruzado
- El número de puntos mínimo (minPoints) y máximo (maxPoints) que previamente establecimos.
- Definimos el tamaño del espacio de trabajo. Para el eje X usamos (Xmin, Xmax) y para el

eje Y usamos (Ymin, Ymax), estos valores los encontramos en los estadígrafos que calculamos de las coordenadas.

- Establecemos el tamaño de la celda, esto lo hacemos con las variables (TX,TY), como podemos observar, el tipo de celda en la siguiente función está determinada por la distancia mínima “DistMin” en ambas direcciones para que obtengamos celdas cuadradas ya que esa es la resolución mínima que tenemos. Si lo deseamos podemos poner diferentes distancias, sin embargo se debe justificar esa decisión, ya que podríamos causar la creación de artefactos.
- Indicamos el tipo de transformación que tiene la variable que estamos usando. El parámetro “InvT” puede considerar tres tipos de transformaciones positivas: 1 si es logarítmica, 2 si es raíz cuadrada y 3 si es inversa. si la variable no tiene transformación solo ponemos 0.
- Por último ingresamos las leyendas del gráfico: para el eje X (NameX), para el eje Y “NameY”, el título de la imagen resultante del kriging “Titulo1” y el título de la imagen obtenida de la desviación estándar (Titulo2).

```
[169]: perm_mD_Log_CoKrig <- CoKrigingOrd(XCoord, YCoord,
                                         phi_per, perm_mD_Log,
                                         phi_per_perm_mD_Log_vario_model,
                                         phi_per_sill_minus_nugget, □
                                         ↪perm_mD_Log_sill_minus_nugget,
                                         phi_per_perm_mD_Log_sill_minus_nugget, □
                                         ↪phi_per_nugget, perm_mD_Log_nugget,
                                         phi_per_perm_mD_Log_nugget, □
                                         ↪phi_per_perm_mD_Log_rank,
                                         minPar = minPoints, maxPar = maxPoints,
                                         Xmin = XCoord_Stat[2,2], Xmax = XCoord_Stat[7,2],
                                         Ymin = YCoord_Stat[2,2], Ymax = YCoord_Stat[7,2],
                                         TX=DistMin, TY=DistMin, InvT=1, NameX="X (ft)",
                                         NameY="Y (ft)", Titulo1="CoKriging perm_mD_Log
                                         ↪(mm)",
                                         Titulo2="SD perm_mD_Log (mm)")
```

Linear Model of Coregionalization found. Good.  
[using ordinary cokriging]



Con este último ejercicio se concluye este trabajo, en caso de tener dudas por favor comunícate con los instructores.

```
[170]: save.image() #important line, this is for save the R workspace, this include
         ↪variables and results
```