Estimacion de la precipitacion en el valle de Mexico usando datos de pluviometros y radar meteorologico

August 18, 2022

1 Ejemplo práctico: estimación de la precipitación en el valle de México usando datos de pluviómetros y radar meteorológico.

Autores: Daniel Vázquez-Ramírez[(1)] , Martín Díaz-Viera[(2)], Van Huong Le[(1)]

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

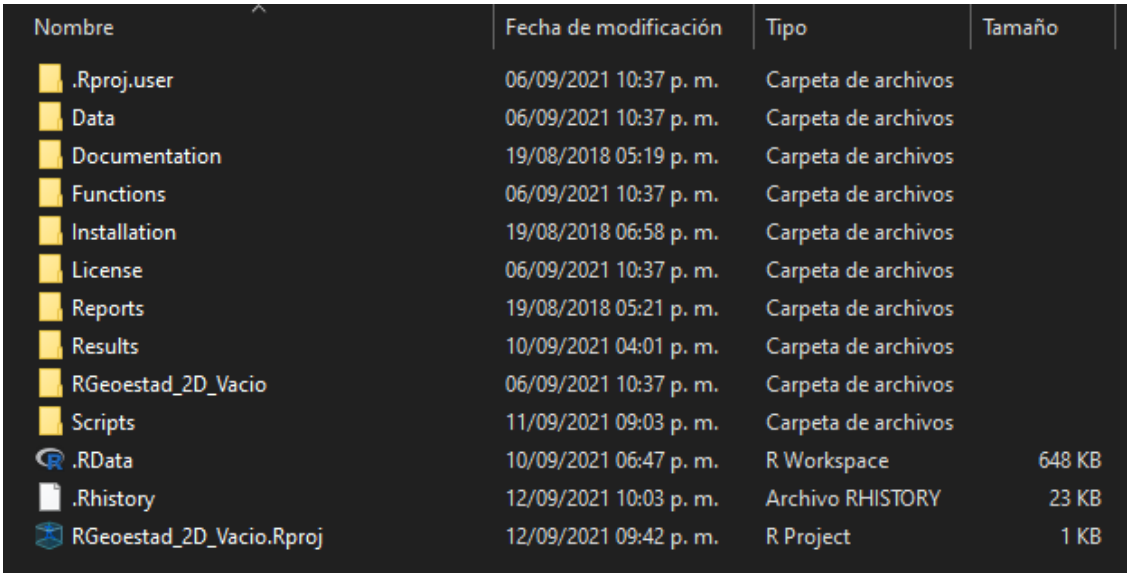
2 Introducción.

El siguiente ejemplo forma parte del artículo publicado por (Diaz,Et.Al,2009), donde se presenta una modificación del procedimiento geoestadístico de estimación espacial de la precipitación introducido por Krajewski (1987), el cual aplica el método de cokriging ordinario, combinando imágenes de radar meteorológico con datos de pluviómetros. Aquí, a diferencia del procedimiento de Krajewski, se incluye en el método de cokriging ordinario la dependencia espacial conjunta de radar-pluviómetros mediante un modelo de corregionalización lineal. La metodología propuesta es probada usando datos de pluviógrafos y de radar de una tormenta ocurrida en el valle de la ciudad de México.

2.1 Abrir el proyecto en R Studio.

El proyecto precargado lo pueden descargar en la página del curso http://www.esmg-mx.org/activities/courses/geoestadistica. Para el análisis exploratorio descargamos la Clase Práctica 1: Análisis Exploratorio de Datos. Se descomprime el archivo y podemos ver que el proyecto se compone de un conjunto de carpetas; la carpeta "Functions" donde encontraran las funciones que usaremos durante la clase práctica y la carpeta "Scripts" donde encontrarán los archivos "Getting_Started_script.R" y "00 Lluvia_AED_2D.R". El primer script contiene las instrucciones para definir directorios, instalar los paquetes, cargar librerías y funciones, el segundo script contiene las instrucciones del ejemplo para hacer el análisis exploratorio. Les recomendamos que el archivo "lluvia.txt" lo guarden en la carpeta "Data".

Para abrir el proyecto en R Studio deben dar doble clic en "RGeoestad_2D_Vacio.Rproj".



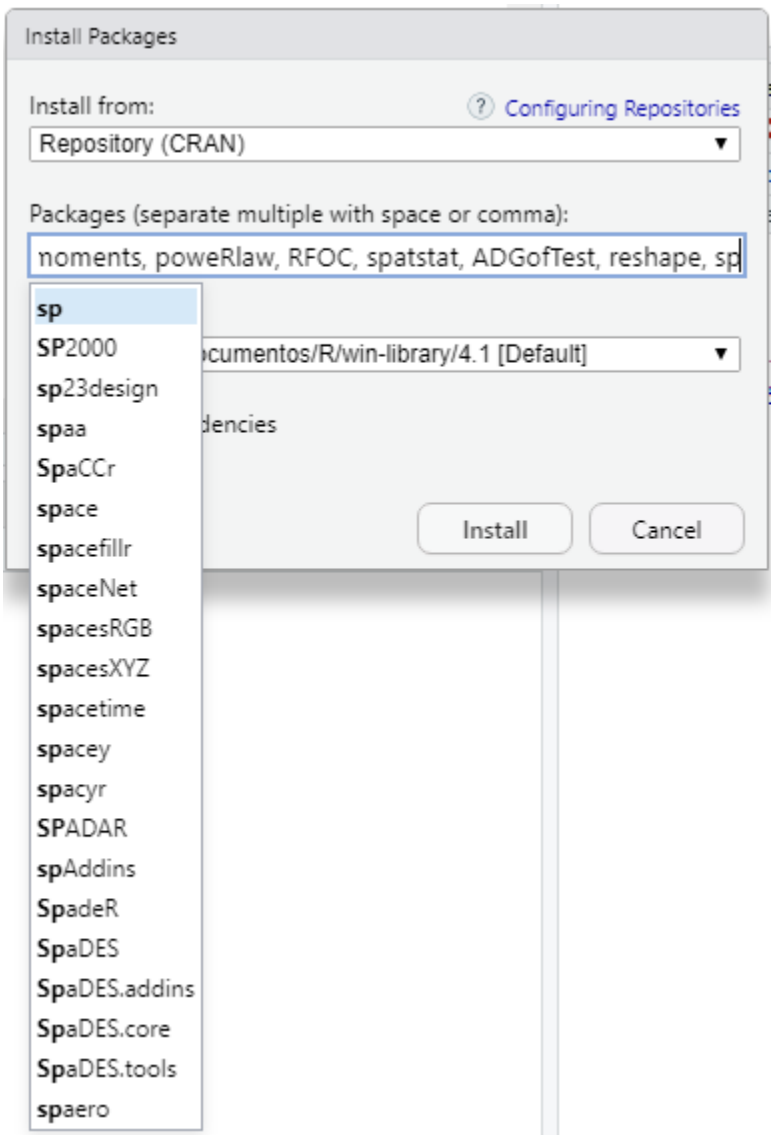
Nombre	Fecha de modificación	Tipo	Tamaño
.Rproj.user	06/09/2021 10:37 p. m.	Carpeta de archivos	
Data	06/09/2021 10:37 p. m.	Carpeta de archivos	
Documentation	19/08/2018 05:19 p. m.	Carpeta de archivos	
Functions	06/09/2021 10:37 p. m.	Carpeta de archivos	
Installation	19/08/2018 06:58 p. m.	Carpeta de archivos	
License	06/09/2021 10:37 p. m.	Carpeta de archivos	
Reports	19/08/2018 05:21 p. m.	Carpeta de archivos	
Results	10/09/2021 04:01 p. m.	Carpeta de archivos	
RGeoestad_2D_Vacio	06/09/2021 10:37 p. m.	Carpeta de archivos	
Scripts	11/09/2021 09:03 p. m.	Carpeta de archivos	
.RData	10/09/2021 06:47 p. m.	R Workspace	648 KB
.Rhistory	12/09/2021 10:03 p. m.	Archivo RHISTORY	23 KB
RGeoestad_2D_Vacio.Rproj	12/09/2021 09:42 p. m.	R Project	1 KB

NOTA: es importante que el proyecto se abra usando "RGeoestad_2D_Vacio.Rproj", si abren solo los scripts no podrán acceder a los directorios.

2.2 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).



La segunda opción es usando scripts, para eso abrimos el script "Getting_Started_script.R" y ejecutar las siguientes lineas.

[ ]:
```
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:

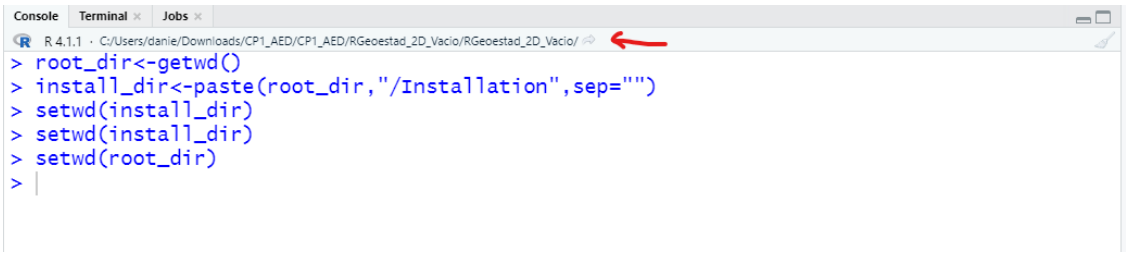
[73]:
```
root_dir<-getwd()
setwd(root_dir)

#### 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.

[74]:
```
root_dir<-getwd()

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

setwd(function_dir)
```

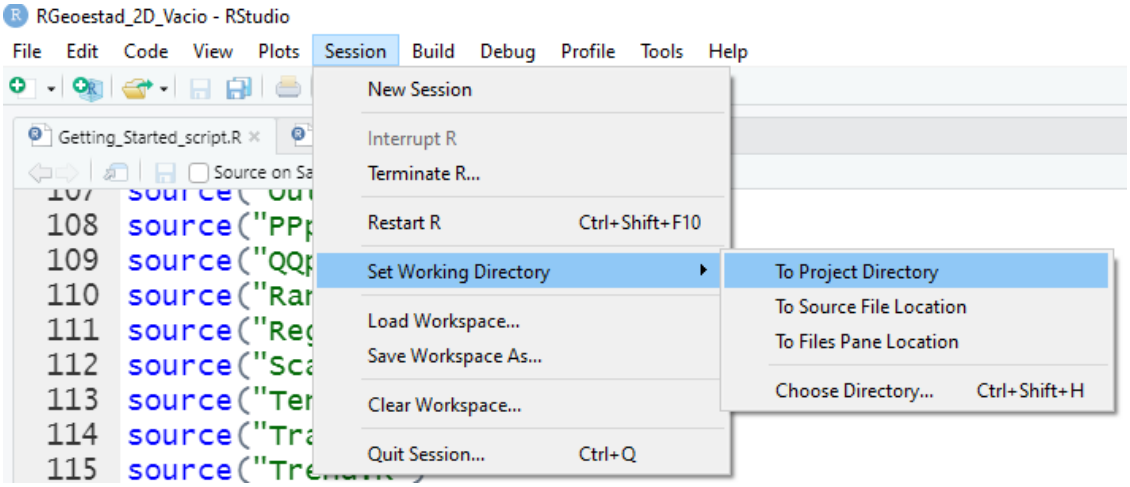
```
source("AllModel.R", encoding='ISO-8859-1')
source("BestModel.R")
source("CDF.R")
source("CoKrigingOrd.R")
source("CoKrigingOrdAnis.R")
source("CrossValidation.R")
source("CrossValidation2.R")
source("CrossVariograma.R")
source("DEspacial.R", encoding='ISO-8859-1')
source("Estadisticas.R")
source("EyeModel.R", 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)
```

Para saber si todos los directorios fueron cargados adecuadamente, veremos que el directorio final mostrado en la consola es "RGeoestad_2D_Vacio" como se muestra en la siguiente imagen.



De lo contrario, se debe poner en la consola la instrucción "setwd(root_dir)" o bien se puede redirigir

el directorio en la barra de menus>Session>Set Working Directory>To Project Directory.



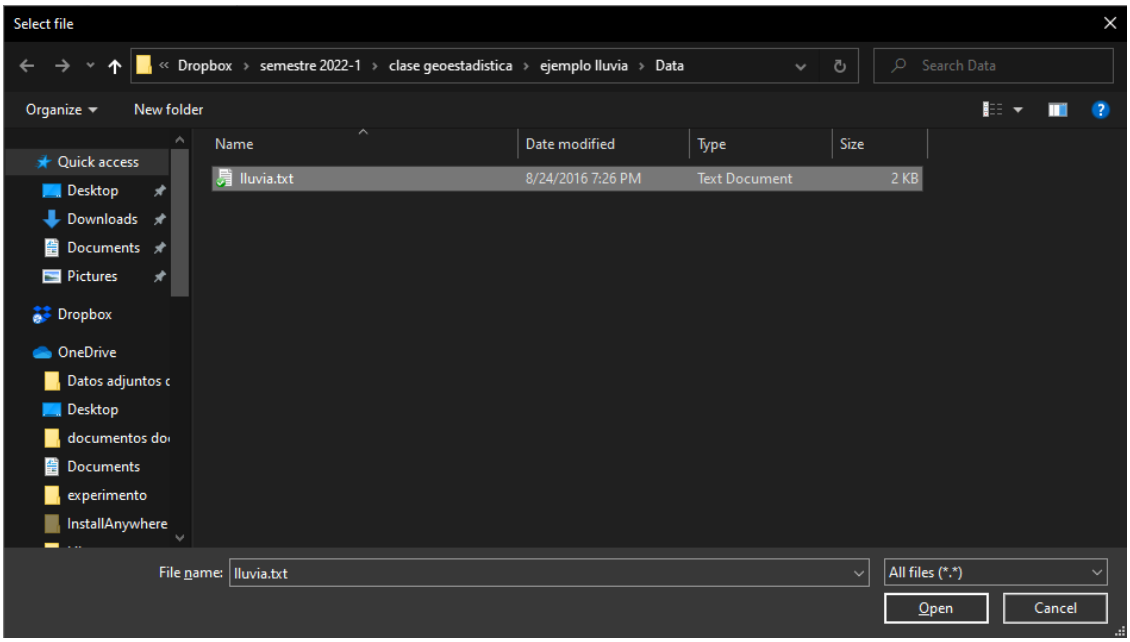
Despues de cargar las funciones podemos continuar con el análisis exploratorio.

Es importante recordar que cada vez que se abre el proyecto en R Studio se deben cargar solo las librerias como se muestra en la siguiente imagen.

```
40
41 library(Rcpp)
42 library(maps)
43 library(mapproj)
44 library(actuar)
45 library(fields)
46 library(fitdistrplus)
47 library(geoR)
48 library(gstat)
49 library(MASS)
50 library(moments)
51 library(poweRlaw)
52 library(RFOC)
53 library(spatstat)
54 library(ADGofTest)
55 library(reshape)
56 library(sp)
```

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



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

NOTA: solo en este notebook se da la instrucción directa de la localización del archivo, en R studio se puede usar el comando file.choose() sin problemas

[75]:
```
Data_File <- read.table(file='Data/lluvia.txt',header=TRUE,na.strings="-999.25")
```

Si desea abrir un archivo con información delimitada por comas (.csv) se cambia la instrucción por: "Data_File <- read.csv(file=file.choose(),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 err
"Warning message in dir.create(paste(getwd(),"/Results/AED", sep = "")) already exists"

[76]:
```
dir.create(paste(getwd(),"/Results/AED", sep=""))
```

```
Warning message in dir.create(paste(getwd(), "/Results/AED", sep = "")):
"'/home/danielvr/Dropbox/Semestre 2023-1/ejemplo lluvia/Results/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, es necesario verificar 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 obtenidos de los pluviómetros (Pluv_mm) y del radar meteorológico (Radar_mm), los cuales tienen una distribución espacial en coordenadas UTM.

Después de cargar el archivo con la información y asignarle el nombre "Data_File", se necesitan las variables aleatorias y su posición espacial. Esas columnas pueden extraerse de la siguiente forma:

[77]:
```
XCoord<-Data_File$UTM_X_m      #Coordenada UTM en x
YCoord<-Data_File$UTM_Y_m      #Coordenada UTM en y
Radar_mm<-Data_File$Radar_mm   #variable con la información del radar␣
 ↪metereológico
Pluv_mm<-Data_File$Pluv_mm     #variable con la información de los pluviómetros
```

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

[78]:
```
XCoord_Stat<-Estadisticas(XCoord)
YCoord_Stat<-Estadisticas(YCoord)
Radar_mm_Stat<-Estadisticas(Radar_mm)
Pluv_mm_Stat<-Estadisticas(Pluv_mm)
```

3.1 Análisis estadístico univariado.

Para la interpretación estadística univariada comenzaremos dos elementos: la tabla con los valores estadísticos y el histograma con boxplot. La tabla con los estadígrafos la obtenemos usando la función "Val_Estadisticos" y la guardamos en la carpeta "AED" con la instrucción "write.csv".

[79]:
```
Data_File_Stat <- Val_Estadisticos(Data_File)
write.csv(Data_File_Stat , file = "Results/AED/Data_File_Stat.csv")  #esta linea␣
 ↪sirve para guardar los resultados en un archivo csv
print(Data_File_Stat[,3:4])
```

```
                  Radar_mm  Pluv_mm
No_muestras       50.00000 50.00000
Minimo             0.18000  0.25000
Cuartil_1er        0.87750  0.31250
Mediana            1.42500  1.00000
Media              1.83500  1.47000
Cuartil_3er        2.37500  1.93750
Maximo             7.79000  7.75000
Rango              7.61000  7.50000
Rango_Intercuartil 1.49750  1.62500
Varianza           2.47352  2.57561
Desv_Estandar      1.57274  1.60487
Simetria           2.02492  2.46929
Curtosis           7.99412 10.05416
```

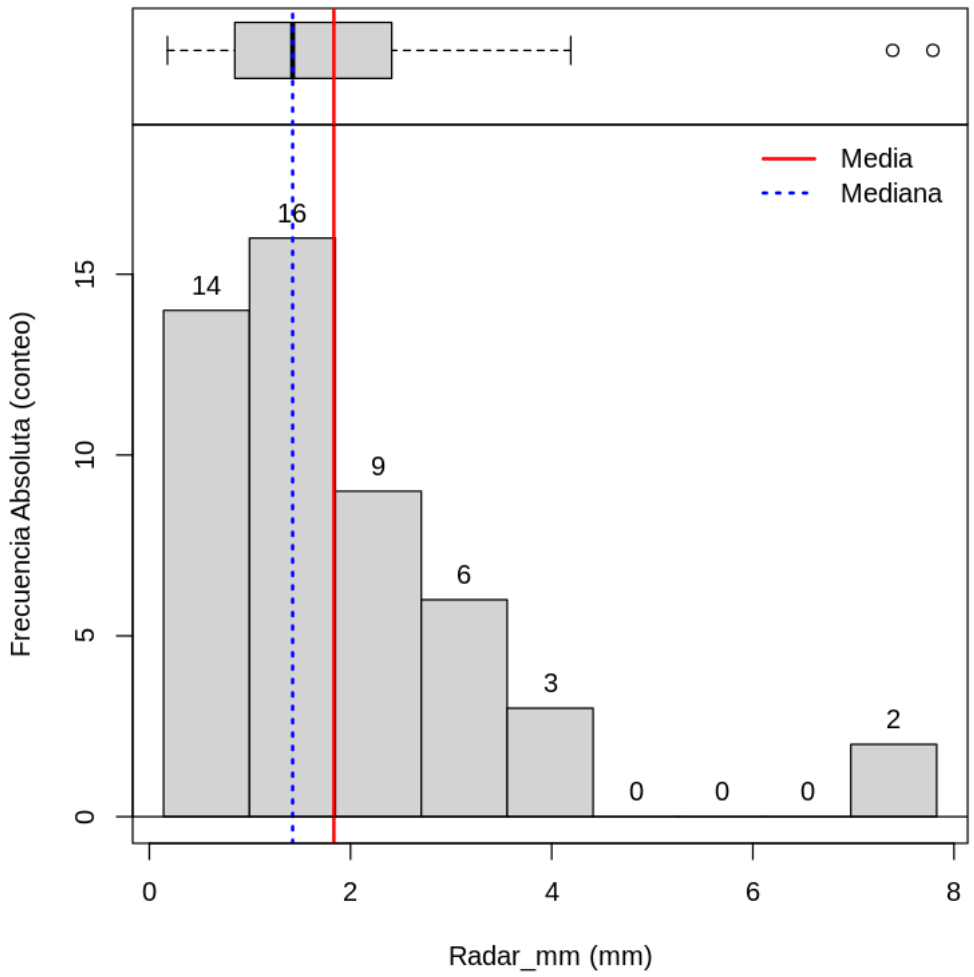
3.1.1 Análisis estadístico univariado para el radar meteorológico (Radar_mm).

El histograma con boxplot se grafica usando la función "HistBoxplot", esta necesita que se asigne el vector con los valores (x), su valor esperado (mean) y su mediana (median). Si observan la instrucción para obtener el histograma para el radar meteorológico, el valor esperado y la mediana la toma de la tabla que se generó con la función "Estadisticas". Por ejemplo, la instrucción "Radar_mm_Stat[5,2]" indica que el valor esperado se encuentra en la fila 5 columna 2. también se indica el número de bins que tendrá el histograma, en este caso es "nbin = 9", es importante recordar que el número por default que encontrarán es 9, pero puede cambiarse según las necesidades del usuario.

La función "HistBoxplot" puede generar dos tipos de histograma: de frecuencia absoluta y de frecuencia relativa. Para obtener el histograma de frecuencia absoluta se indica que "AbsFreq = TRUE, PercentFreq = FALSE" y para el histograma de frecuencia relativa se indicar que "AbsFreq = FALSE, PercentFreq = TRUE".

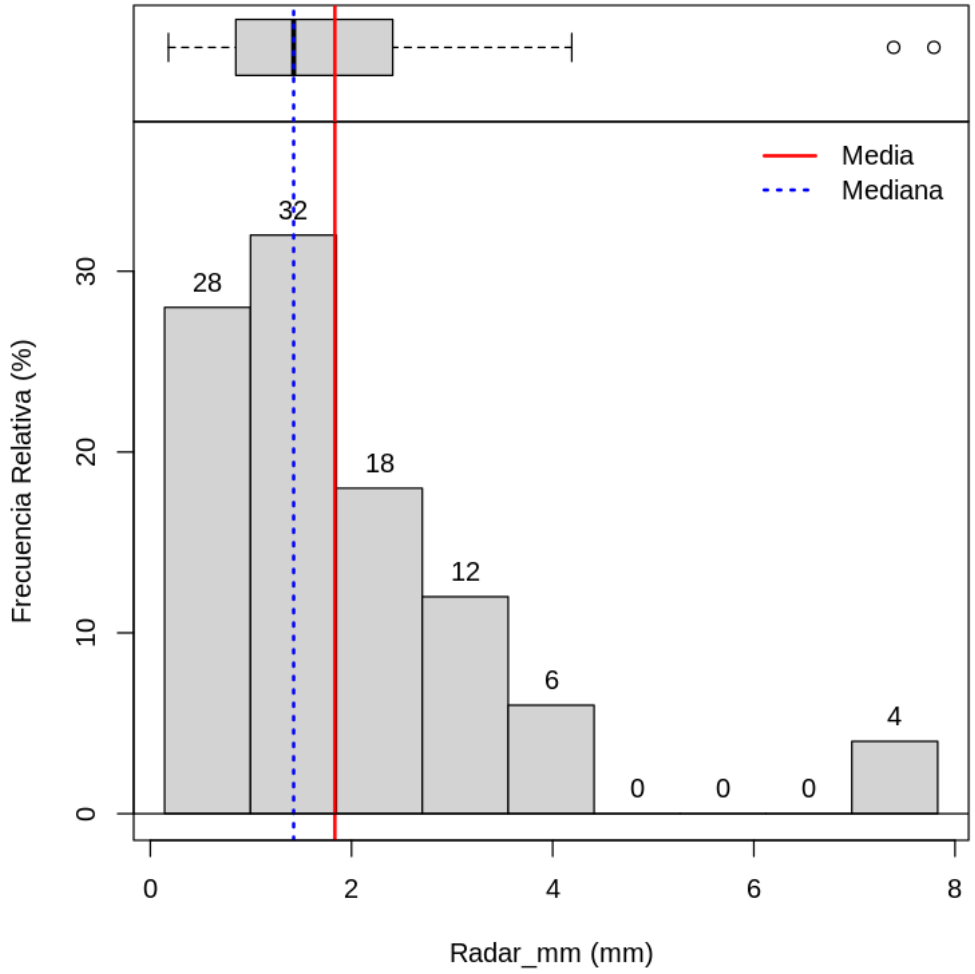
El histograma para el radar meteorológico (Radar_mm) con frecuencia absoluta es:

[80]:
```
HistBoxplot(x=Radar_mm, mean = Radar_mm_Stat[5,2], median = Radar_mm_Stat[4,2],␣
↪main ="",
            xlab = "Radar_mm (mm)", ylab = "Frecuencia Absoluta (conteo)",␣
↪AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



Y el histograma del radar meteorológico con frecuencia relativa es:

[81]:
```
HistBoxplot(x=Radar_mm, mean = Radar_mm_Stat[5,2], median = Radar_mm_Stat[4,2],␣
↪main ="",
            xlab = "Radar_mm (mm)", ylab = "Frecuencia Relativa (%)", AbsFreq =␣
↪FALSE, PercentFreq = TRUE,
            nbin = 9)
```



Analizando los histogramas y los estadígrafos de la variable del radar meteorológico (Radar_mm) tiene una diferencia entre la media y la mediana de 0.47, su coeficiente de asimetría es de 2.46929, lo cual significa que la variable es asimétrica. Esto se confirma con los histogramas, los cuales muestran que la asimetría es positiva. También podemos notar que el boxplot muestra dos valores atípicos localizados en el extremo derecho. El valor de la curtosis es de 7.99412, lo cual nos indica que es leptocúrtica.

Para saber cuáles son esos valores atípicos usamos la función "OutliersPos".

[82]:
```
Radar_mm_outliers<-OutliersPos(Radar_mm)
Data_File[Radar_mm_outliers,c(1,2,3)]
```

A data.frame: 2 × 3

	UTM_X_m <int>	UTM_Y_m <int>	Radar_mm <dbl>
32	482135	2129478	7.79
36	497898	2129469	7.39

El resultado indica que la muestra 32 y 36 son los valores atípicos, por lo tanto, se elimina esos valores de la variable creando una nueva variable que llamaremos "Radar_mm_out". También se calcula calcular los estadísticos para esta nueva variable.

NOTA: Eliminar los outliers no significa que no se usen más adelante, los valores atípicos son necesarios para saber si conviene o no eliminarlos y para la estimación del variograma, para ello se genera un nuevo vector. Por ejemplo, el vector "Radar_mm" es el vector original, el vector "Radar_mm_out" contiene todos los valores del vector original excepto aquellos que se marcaron como valores atípicos.

NOTA: Si desea utilizar el vector sin los valores atípicos, también deberá eliminar las coordenadas de los valores atípicos detectados, de lo contrario, no se podrá seguir con el análisis variográfico.

Para retirar los valores atípicos y sus coordenadas hay que generar una nueva tabla de la siguiente forma: primero hay que unir las columnas de las coordenadas (XCoord,YCoord) y la variable que se está analizando (Radar_mm) usando la función "cbind"

[83]:
```
Radar_DF<-cbind(XCoord,YCoord,Radar_mm)
```

Después hay que convertir esa tabla en un data frame. Para esto se usa la función "data.frame", además, se debe indicar cuales son las filas a retirar, para esto se usa el argumento "Radar_DF[-c(Radar_mm_outliers),]", en este caso "-c" indica que se retiran las filas con los números indicados en "Radar_mm_outliers" en la tabla antes generada, la cual se llama "Radar_DF"

[84]:
```
Radar_mm_out_DF<-data.frame(Radar_DF[-c(Radar_mm_outliers),])
```

Generado este nuevo dataframe, podemos llamar a la columna de la variable sin los valores atípicos usando la instrucción "$"

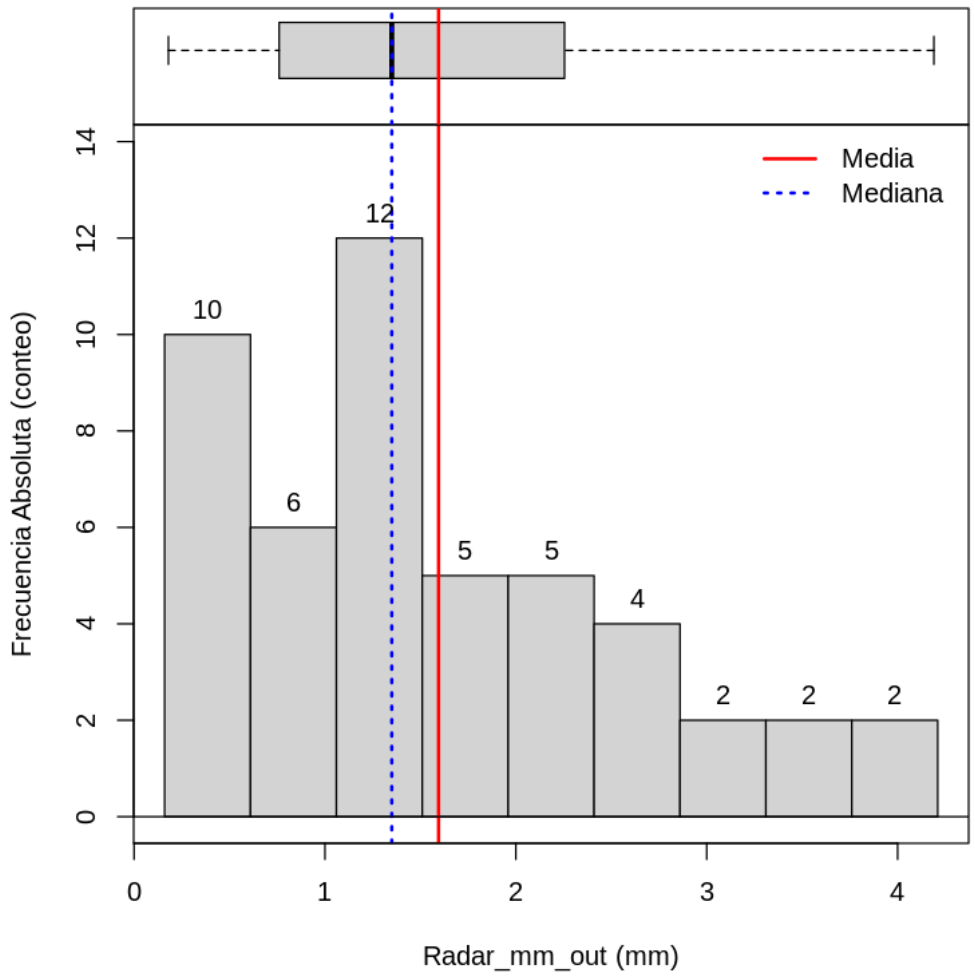
[85]:
```
Radar_mm_out<-Radar_mm_out_DF$Radar_mm
Radar_mm_out_Stat<-Estadisticas(Radar_mm_out)
Radar_mm_out_Stat
```

A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	48.0000
minimos	Minimum	0.1800
cuantiles1	1st. Quartile	0.8050
medianas	Median	1.3500
medias	Mean	1.5952
cuantiles3	3rd. Quartile	2.2475
maximos	Maximum	4.1900
rangos	Rank	4.0100
rangosInt	Interquartile Rank	1.4425
varianzas	Variance	1.1090
desvs	Standard Deviation	1.0531
CVs	Variation Coeff.	0.6602
simetrias	Skewness	0.6937
curtosiss	Kurtosis	2.7723

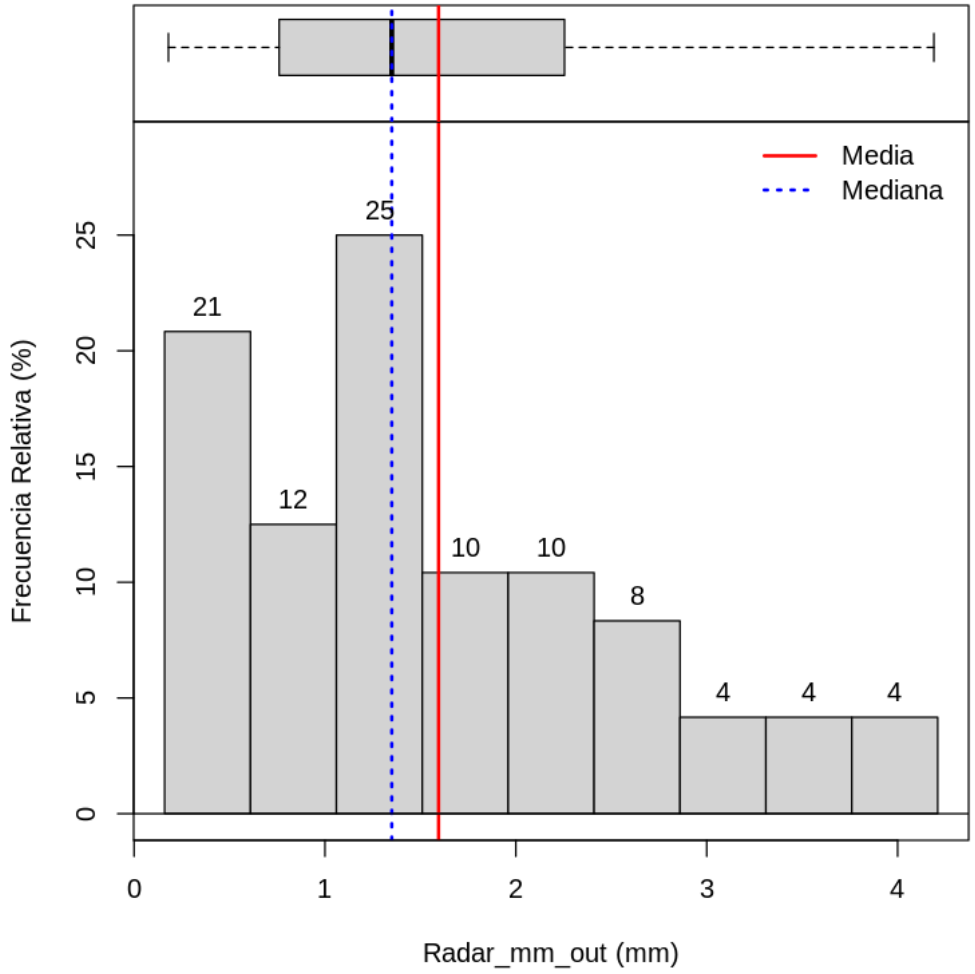
Volvemos a graficar el histograma para los valores obtenidos con el radar meteorológico sin valores atípicos (Radar_mm_out) con frecuencia absoluta.

[86]:
```
HistBoxplot(x=Radar_mm_out, mean = Radar_mm_out_Stat[5,2], median =␣
↪Radar_mm_out_Stat[4,2], main ="",
            xlab = "Radar_mm_out (mm)", ylab = "Frecuencia Absoluta (conteo)",␣
↪AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



Y el histograma para los valores obtenidos con el radar meteorológico sin valores atípicos (Radar_mm_out) con frecuencia relativa.

[87]:
```
HistBoxplot(x=Radar_mm_out, mean = Radar_mm_out_Stat[5,2], median =␣
↪Radar_mm_out_Stat[4,2], main ="",
            xlab = "Radar_mm_out (mm)", ylab = "Frecuencia Relativa (%)",␣
↪AbsFreq = FALSE, PercentFreq = TRUE,
            nbin = 9)
```



Analizando la diferencia entre me media y la mediana de esta nueva variable es de 0.2452 y su asimetría es de 0.6937. En principio esto nos indica que la asimetría disminuyó, sin embargo, la variable sigue mostrando asimetría positiva. Con el caso de la curtosis, el valor pasó de 7.99412 a 2.7723, con lo cual consideramos que es planicúrtica.

Ahora podemos observar que los boxplot no muestran nuevos valores atípicos para la variable del radar meteorológico sin valores atípicos (Radar_mm_out). Esto lo podemos confirmar usando la función "OutliersPos".

[88]:
```
Radar_mm_out_outliers<-OutliersPos(Radar_mm_out)
print(Radar_mm_out_outliers)
```

```
numeric(0)
```

Transformación de variable para el radar meteorológico (Radar_mm). 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 razones más comunes para aplicar una transformación son:

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

Las transformaciones más comunes son:

Para Asimetrías positivas:

- Raíz cuadrada $v_{at} = \sqrt{v_a}$
- Logarítmica $v_{at} = Log(v_a)$
- Recíproca $v_{at} = \frac{1}{v_a}$

Para asimetrías negativas:

- Potencias $v_{at} = v_a^n$
- Arcseno $v_{at} = arcsen(v_a)$
- Exponencial $v_{at} = exp(v_a)$

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

NOTA: El usuario debe saber si la transformación puede ser usada en la variable aleatoria. Por ejemplo, si la variable aleatoria contiene muestras con valor cero, entonces no podrá usar las transformaciones logarítmica o reciproca. Si desea usar la transformación arcseno, la variable aleatoria debe tener componente trigonométrica.

Dado que la variable del radar meteorológico (Radar_mm) presenta asimetría positiva, se van a usar dos tipos de transformaciones: raíz cuadrada y logarítmica.

Transformación de raíz cuadrada

Empezaremos con la transformación de raíz cuadrada usando el vector "Radar_mm", la cual se hace de la siguiente forma:

[89]:
```
Data_File$Radar_mm_Sqrt<-sqrt(Radar_mm)
Radar_mm_Sqrt <- Data_File$Radar_mm_Sqrt
```

Ya que obtenemos la variable aleatoria transformada por raíz cuadrada (Radar_mm_Sqrt), debemos obtener sus estadígrafos.

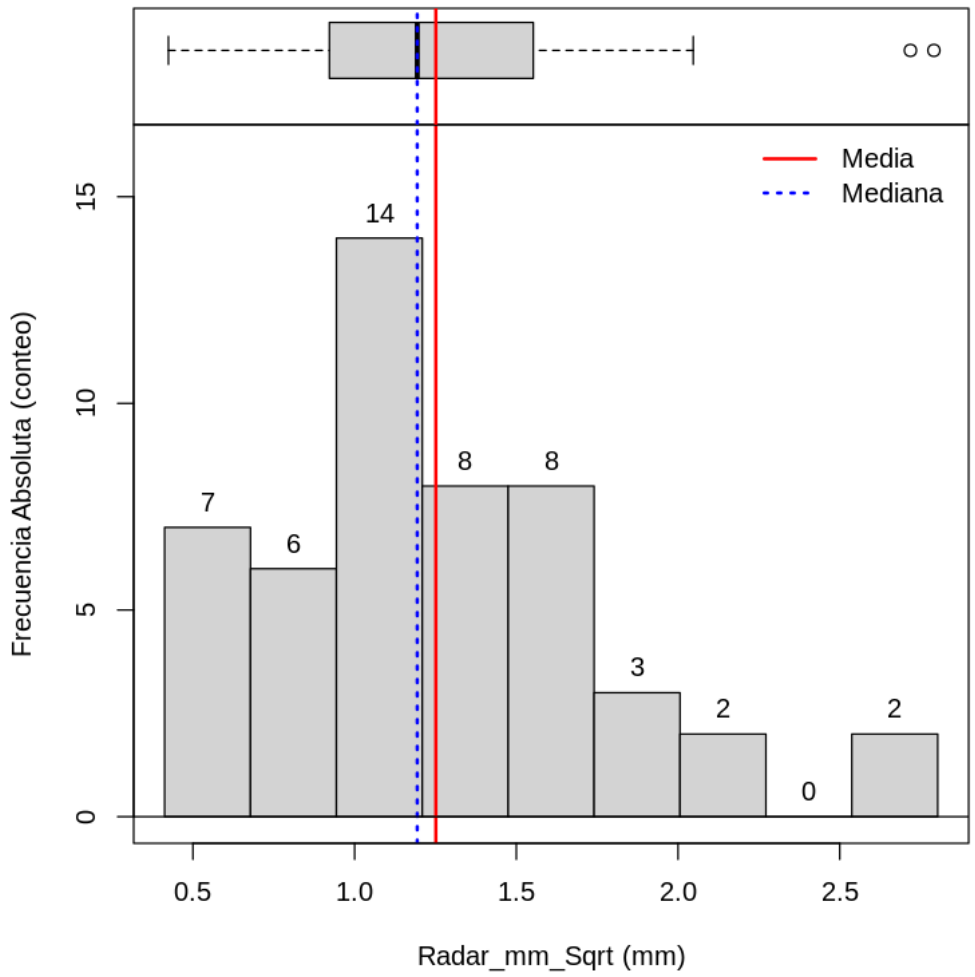
[90]:
```
# Estadística básica
Radar_mm_Sqrt_Stat<-Estadisticas(Radar_mm_Sqrt) #estos valores los necesitamos␣
↪para poder graficar los histogramas

# Estadística básica comparativa entre (Radar_mm) y la transformación␣
↪(Radar_mm_Sqrt)
Data_File_Stat <- Val_Estadisticos(Data_File)
Data_File_Stat[,c(3,5)]
```

A data.frame: 13 × 2

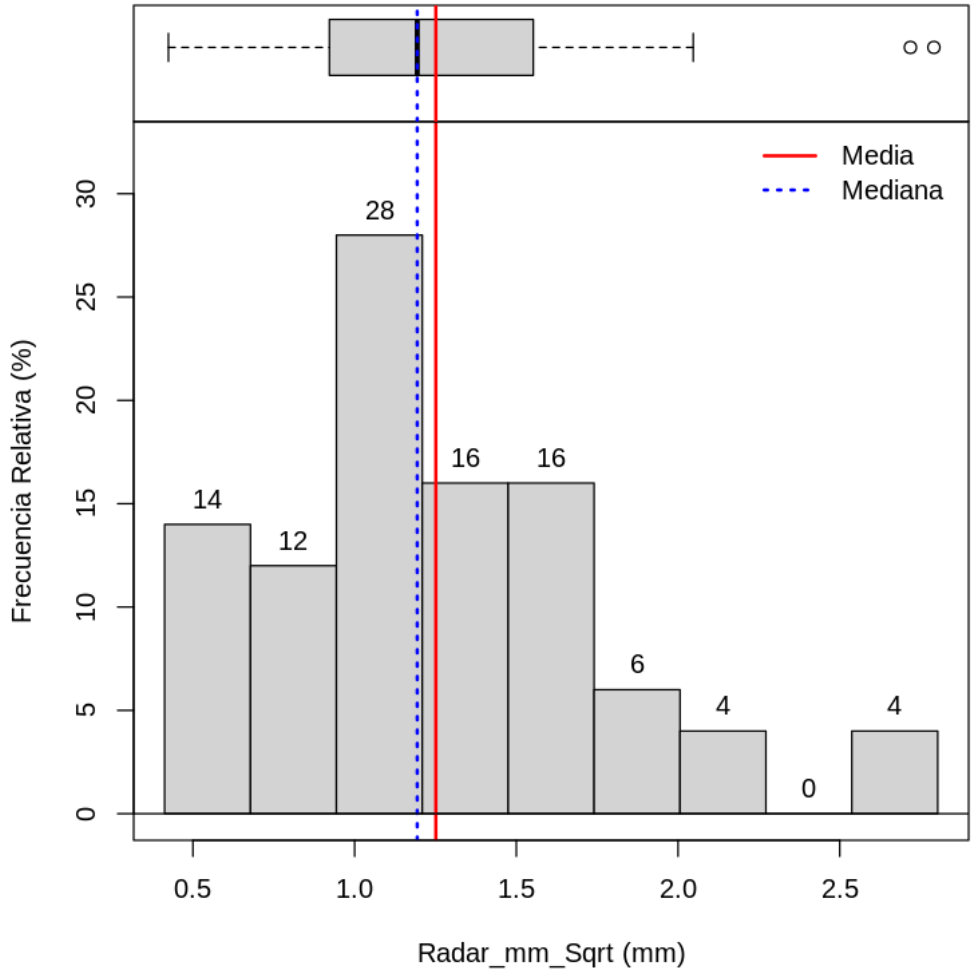
	Radar_mm <dbl>	Radar_mm_Sqrt <dbl>
No_muestras	50.00000	50.00000
Minimo	0.18000	0.42426
Cuartil_1er	0.87750	0.93641
Mediana	1.42500	1.19364
Media	1.83500	1.25106
Cuartil_3er	2.37500	1.54098
Maximo	7.79000	2.79106
Rango	7.61000	2.36679
Rango_Intercuartil	1.49750	0.60456
Varianza	2.47352	0.27536
Desv_Estandar	1.57274	0.52474
Simetria	2.02492	0.74939
Curtosis	7.99412	3.86947

Y graficamos los histogramas con frecuencia absoluta y relativa de la transformación raíz cuadrada.

[91]:
```
HistBoxplot(x=Radar_mm_Sqrt, mean = Radar_mm_Sqrt_Stat[5,2], median =␣
↪Radar_mm_Sqrt_Stat[4,2], main ="",
            xlab = "Radar_mm_Sqrt (mm)", ylab = "Frecuencia Absoluta (conteo)",␣
↪AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



[92]:
```
HistBoxplot(x=Radar_mm_Sqrt, mean = Radar_mm_Sqrt_Stat[5,2], median =␣
↪Radar_mm_Sqrt_Stat[4,2], main ="",
            xlab = "Radar_mm_Sqrt (mm)", ylab = "Frecuencia Relativa (%)",␣
↪AbsFreq = FALSE, PercentFreq = TRUE,
            nbin = 9)
```



Si observamos la diferencia entre la media y la mediana de la variable transformada (Radar_mm_Sqrt) podemos notar que es de 0.05742, lo cual podemos considerar como muy bajo en comparación de la diferencia obtenida de la variable sin transformar que es de 0.47, sin embargo, observamos que el boxplot muestra la presencia de dos valores atípicos, los cuales debemos retirar y verificar que no afecten la supuesta simetría que hemos logrado con la transformación.

[93]:
```
Radar_mm_Sqrt_outliers<-OutliersPos(Radar_mm_Sqrt)
Data_File[Radar_mm_Sqrt_outliers,c(1,2,5)]
```

A data.frame: 2 × 3

	UTM_X_m <int>	UTM_Y_m <int>	Radar_mm_Sqrt <dbl>
32	482135	2129478	2.791057
36	497898	2129469	2.718455

Ahora que se sabe cuáles son los valores atípicos, se debe generar una nueva tabla para retirar los valores atípicos junto con sus coordenadas, para esto se usan las siguientes líneas:

[94]:
```
Radar_sqrt_DF<-cbind(XCoord,YCoord,Radar_mm_Sqrt)
```

[95]:
```
Radar_mm_sqrt_out_DF<-data.frame(Radar_sqrt_DF[-c(Radar_mm_Sqrt_outliers),])
```

Se calculan los estadígrafos

[281]:
```
Radar_mm_Sqrt_out<-Radar_mm_sqrt_out_DF$Radar_mm_Sqrt
Radar_mm_Sqrt_out_stat<-Estadisticas(Radar_mm_Sqrt_out)
Radar_mm_Sqrt_out_stat
```

A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	48.0000
minimos	Minimum	0.4243
cuantiles1	1st. Quartile	0.8961
medianas	Median	1.1618
medias	Mean	1.1884
cuantiles3	3rd. Quartile	1.4992
maximos	Maximum	2.0469
rangos	Rank	1.6227
rangosInt	Interquartile Rank	0.6031
varianzas	Variance	0.1868
desvs	Standard Deviation	0.4322
CVs	Variation Coeff.	0.3637
simetrias	Skewness	0.0273
curtosiss	Kurtosis	2.2704

[97]:
```
Comparative_sqrt<-cbind(Radar_mm_Stat[2],Radar_mm_out_Stat[2],Radar_mm_Sqrt_Stat[2],Radar_mm_Sq
colnames(Comparative_sqrt)<-c("Radar_mm","Radar_mm_out","Radar_mm_Sqrt","Radar_mm_Sqrt_out")
Comparative_sqrt
```

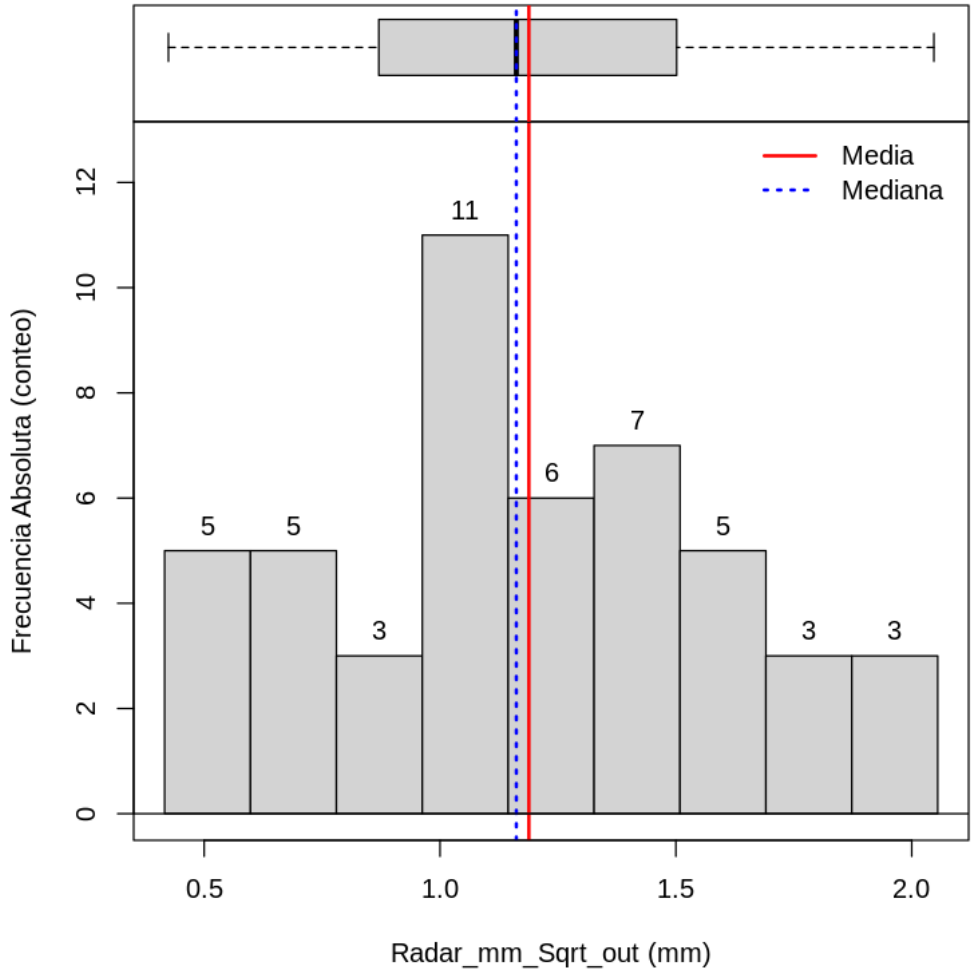
A data.frame: 14 × 4

	Radar_mm <dbl>	Radar_mm_out <dbl>	Radar_mm_Sqrt <dbl>	Radar_mm_Sqrt_out <dbl>
muestras	50.0000	48.0000	50.0000	48.0000
minimos	0.1800	0.1800	0.4243	0.4243
cuantiles1	0.8775	0.8050	0.9364	0.8961
medianas	1.4250	1.3500	1.1936	1.1618
medias	1.8350	1.5952	1.2511	1.1884
cuantiles3	2.3750	2.2475	1.5410	1.4992
maximos	7.7900	4.1900	2.7911	2.0469
rangos	7.6100	4.0100	2.3668	1.6227
rangosInt	1.4975	1.4425	0.6046	0.6031
varianzas	2.4735	1.1090	0.2754	0.1868
desvs	1.5727	1.0531	0.5247	0.4322
CVs	0.8571	0.6602	0.4194	0.3637
simetrias	2.0249	0.6937	0.7494	0.0273
curtosiss	7.9941	2.7723	3.8695	2.2704

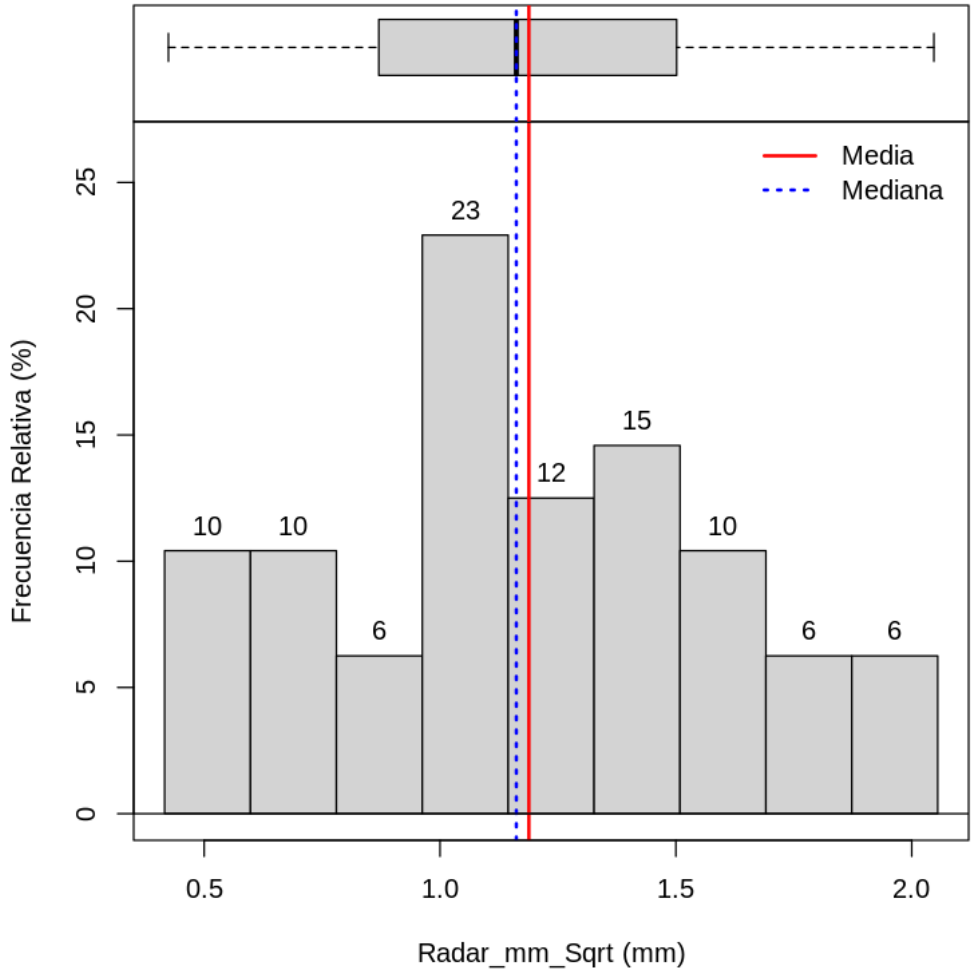
Con los resultados obtenidos al retirar los dos valores atípicos podemos notar que la diferencia entre la media y la mediana pasaron de 0.05742 a 0.0266, podemos considerar que la transformación cumplió su propósito al reducir de manera significativa la asimetría, sin embargo, comparando las muestras transformadas con y sin valores atípicos podría no justificar el retiro de los valores atípicos ya que es muy pequeña la diferencia.

Y graficamos sus histogramas.

[98]:
```
HistBoxplot(x=Radar_mm_Sqrt_out, mean = Radar_mm_Sqrt_out_stat[5,2], median =␣
↪Radar_mm_Sqrt_out_stat[4,2], main ="",
            xlab = "Radar_mm_Sqrt_out (mm)", ylab = "Frecuencia Absoluta␣
↪(conteo)", AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



[99]:
```
HistBoxplot(x=Radar_mm_Sqrt_out, mean = Radar_mm_Sqrt_out_stat[5,2], median =␣
↪Radar_mm_Sqrt_out_stat[4,2], main ="",
            xlab = "Radar_mm_Sqrt (mm)", ylab = "Frecuencia Relativa (%)",␣
↪AbsFreq = FALSE, PercentFreq = TRUE,
            nbin = 9)
```



Confirmamos que no existan más valores atípicos.

[100]:
```
OutliersPos(Radar_mm_Sqrt_out)
```

Transformación logarítmica

Ahora probaremos usando la transformación logarítmica. La cual se hace de la siguiente forma:

[101]:
```
Data_File$Radar_mm_Log<-log(Radar_mm)
Radar_mm_Log <- Data_File$Radar_mm_Log
```

y al igual que la transformada por raíz cuadrada, obtenemos sus valores estadísticos.

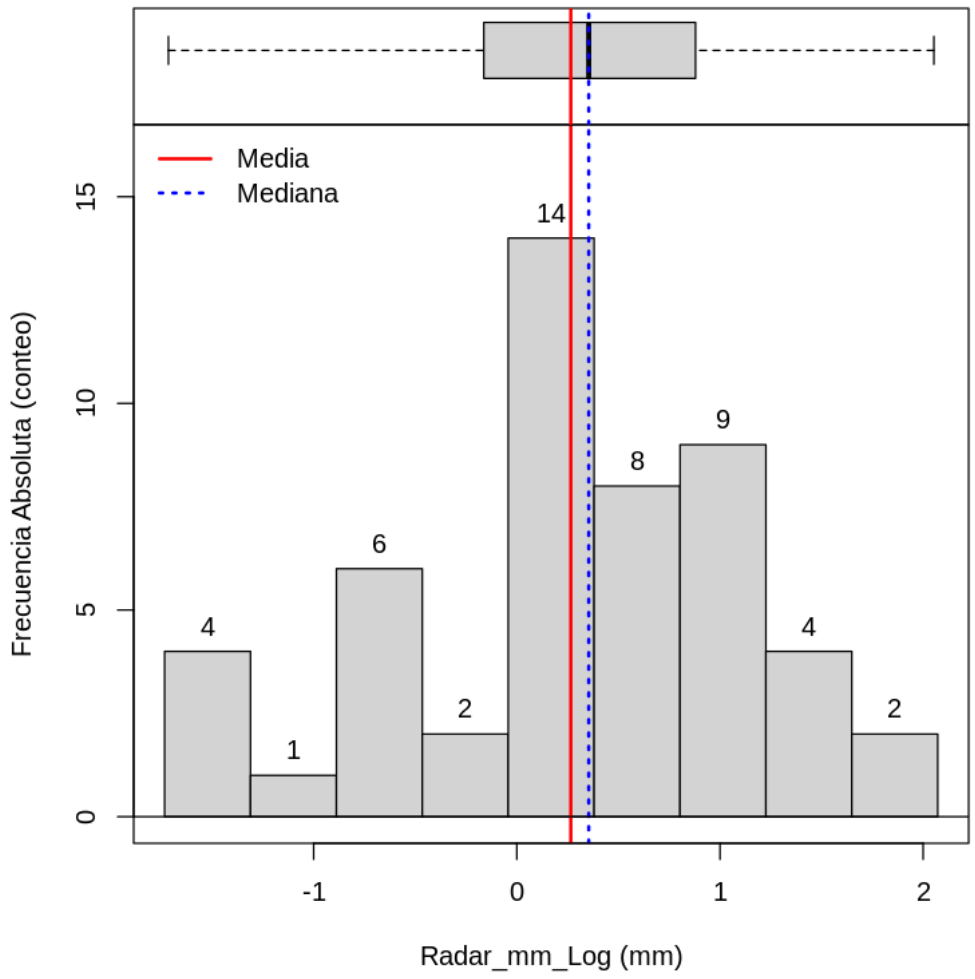
[102]:
```
# Estadística básica
Radar_mm_Log_Stat<-Estadisticas(Radar_mm_Log)

# Estadística básica comparativa entre (Radar_mm), la transformación␣
↪(Radar_mm_Sqrt) y (Radar_mm_Log)
Data_File_Stat <- Val_Estadisticos(Data_File)
Data_File_Stat[,c(3,5,6)]
```

A data.frame: 13 × 3

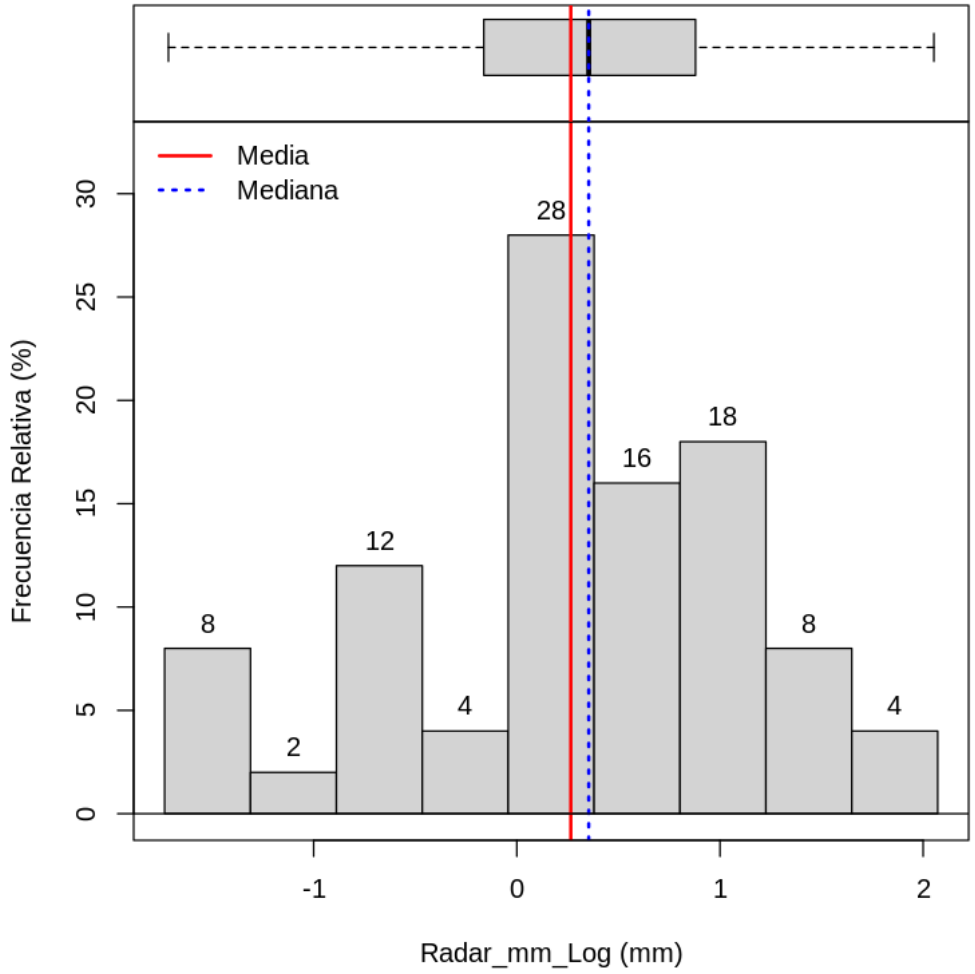
	Radar_mm <dbl>	Radar_mm_Sqrt <dbl>	Radar_mm_Log <dbl>
No_muestras	50.00000	50.00000	50.00000
Minimo	0.18000	0.42426	-1.71480
Cuartil_1er	0.87750	0.93641	-0.13209
Mediana	1.42500	1.19364	0.35387
Media	1.83500	1.25106	0.26569
Cuartil_3er	2.37500	1.54098	0.86467
Maximo	7.79000	2.79106	2.05284
Rango	7.61000	2.36679	3.76764
Rango_Intercuartil	1.49750	0.60456	0.99676
Varianza	2.47352	0.27536	0.79776
Desv_Estandar	1.57274	0.52474	0.89317
Simetria	2.02492	0.74939	-0.46105
Curtosis	7.99412	3.86947	2.87206

Y graficamos sus respectivos histogramas.

[103]:
```
HistBoxplot(x=Radar_mm_Log, mean = Radar_mm_Log_Stat[5,2], median =␣
↪Radar_mm_Log_Stat[4,2], main ="",
            xlab = "Radar_mm_Log (mm)", ylab = "Frecuencia Absoluta (conteo)",␣
↪AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



[104]:
```
HistBoxplot(x=Radar_mm_Log, mean = Radar_mm_Log_Stat[5,2], median =␣
↪Radar_mm_Log_Stat[4,2], main ="",
            xlab = "Radar_mm_Log (mm)", ylab = "Frecuencia Relativa (%)",␣
↪AbsFreq = FALSE, PercentFreq = TRUE,
            nbin = 9)
```



La diferencia entre la media y la mediana de la transformación logarítmica es de 0.08818, de forma aparente esta diferencia es mayor comparada con la transformación por raíz cuadrada. El boxplot indica que no existen valores atípicos, sin embargo, se usará la función para detectar los valores atípicos.

[105]:
```
Radar_mm_Log_outliers<-OutliersPos(Radar_mm_Log)
Data_File[Radar_mm_Log_outliers,c(1,2,6)]
```

A data.frame: 2 × 3

	UTM_X_m <int>	UTM_Y_m <int>	Radar_mm_Log <dbl>
22	462178	2133934	-1.714798
49	503149	2123937	-1.660731

Como podemos ver, la función ha detectado dos valores atípicos, los cuales retiraremos y obtenemos

sus valores estadísticos.

[106]:
```
Radar_mm_log_DF<-cbind(XCoord,YCoord,Radar_mm_Log)
Radar_mm_log_out_DF<-data.frame(Radar_mm_log_DF[-c(Radar_mm_Log_outliers),])
```

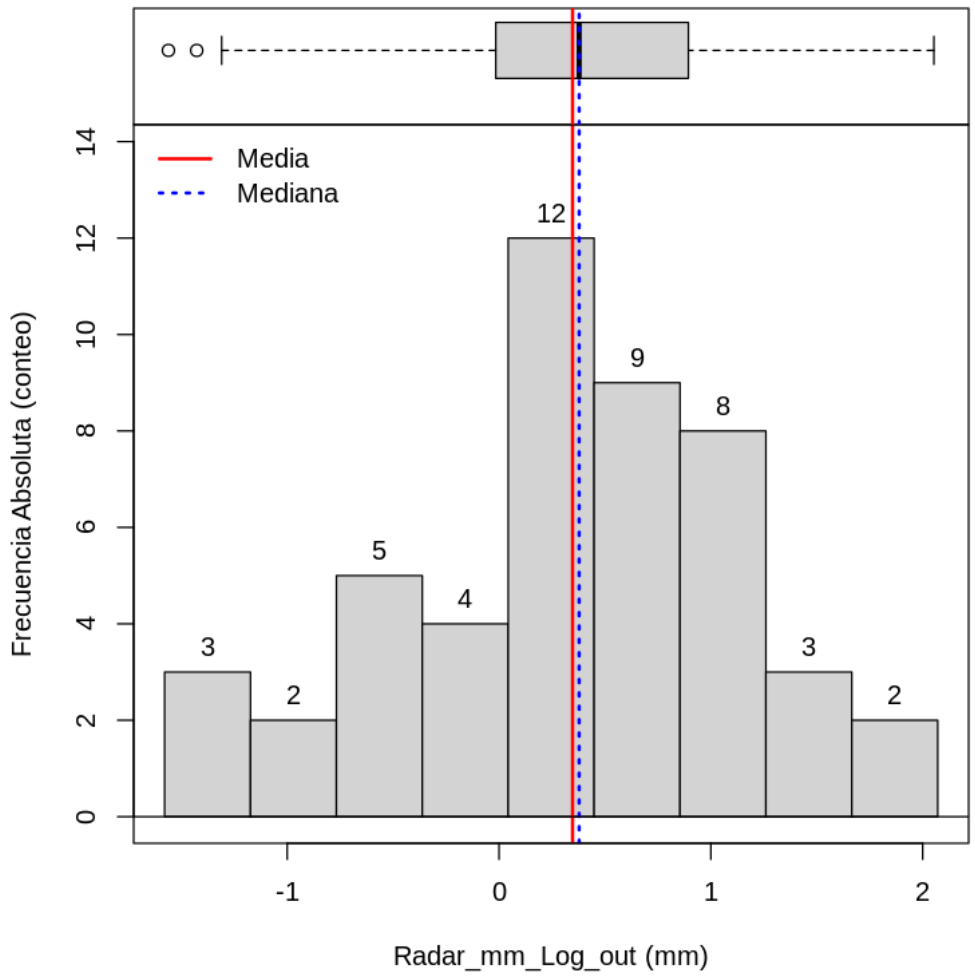
[107]:
```
Radar_mm_Log_out<-Radar_mm_log_out_DF$Radar_mm_Log

# Estadística básica
Radar_mm_Log_out_Stat<-Estadisticas(Radar_mm_Log_out)
Radar_mm_Log_out_Stat
```

A data.frame: 14 × 2

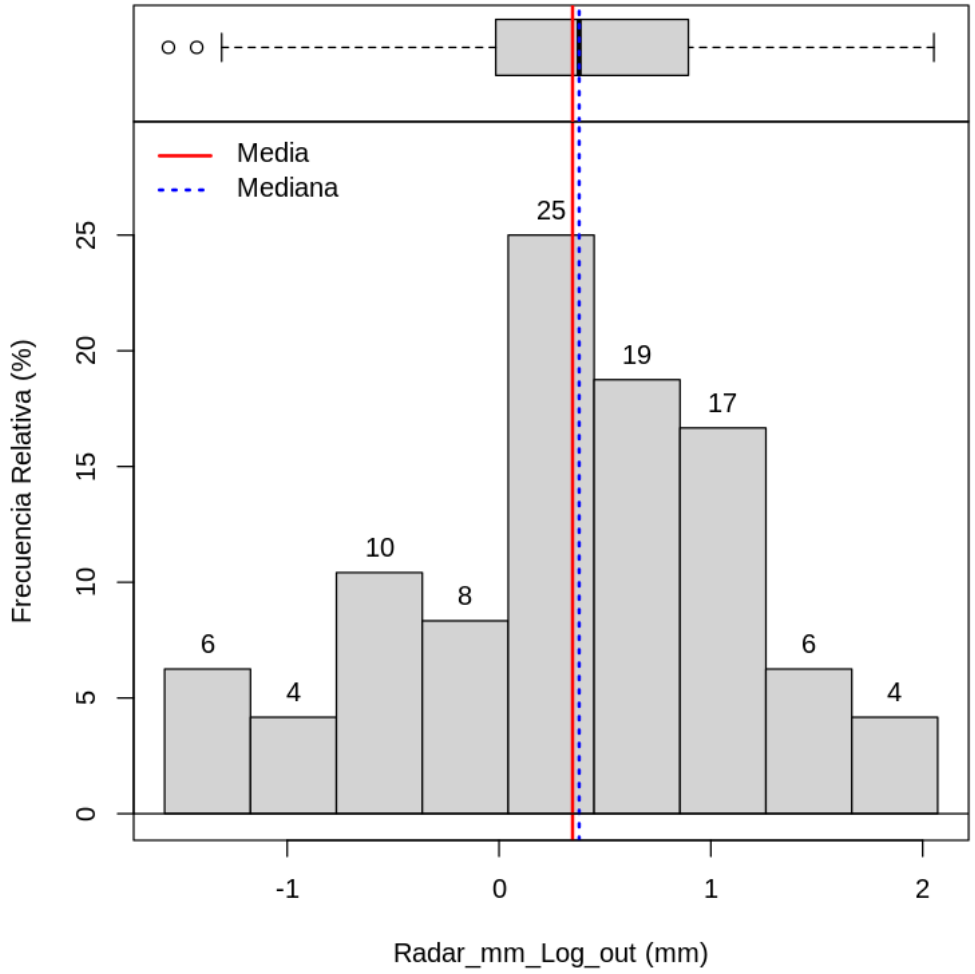
	Statistics <chr>	Values <dbl>
muestras	n	48.0000
minimos	Minimum	-1.5606
cuantiles1	1st. Quartile	-0.0027
medianas	Median	0.3784
medias	Mean	0.3471
cuantiles3	3rd. Quartile	0.8868
maximos	Maximum	2.0528
rangos	Rank	3.6135
rangosInt	Interquartile Rank	0.8895
varianzas	Variance	0.6625
desvs	Standard Deviation	0.8140
CVs	Variation Coeff.	2.3451
simetrias	Skewness	-0.3448
curtosiss	Kurtosis	2.9797

Y graficamos sus histogramas:

[108]:
```
HistBoxplot(x=Radar_mm_Log_out, mean = Radar_mm_Log_out_Stat[5,2], median =␣
↪Radar_mm_Log_out_Stat[4,2], main ="",
            xlab = "Radar_mm_Log_out (mm)", ylab = "Frecuencia Absoluta␣
↪(conteo)", AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



[109]:
```
HistBoxplot(x=Radar_mm_Log_out, mean = Radar_mm_Log_out_Stat[5,2], median =␣
↪Radar_mm_Log_out_Stat[4,2], main ="",
            xlab = "Radar_mm_Log_out (mm)", ylab = "Frecuencia Relativa (%)",␣
↪AbsFreq = FALSE, PercentFreq = TRUE,
            nbin = 9)
```



La diferencia entre la media y la mediana de la transformación logarítmica quitando los valores atípicos detectados pasó de 0.08818 a 0.0313, sin embargo, el boxplot muestran dos valores atípicos localizados a la izquierda del gráfico. Por lo tanto, retiraremos esos nuevos valores atípicos.

[110]:
```
Radar_mm_Log_outliers2<-OutliersPos(Radar_mm_Log_out)
Radar_mm_log_out_DF[Radar_mm_Log_outliers2,c(1,2,3)]
```

A data.frame: 2 × 3

	XCoord <dbl>	YCoord <dbl>	Radar_mm_Log <dbl>
45	467433	2135030	-1.427116
48	472686	2135022	-1.560648

Y hacemos un analisis estadístico de este nuevo conjunto de valores.

[111]:
```
Radar_mm_log_out_DF2<-data.
↪frame(Radar_mm_log_out_DF[-c(Radar_mm_Log_outliers2),])
```

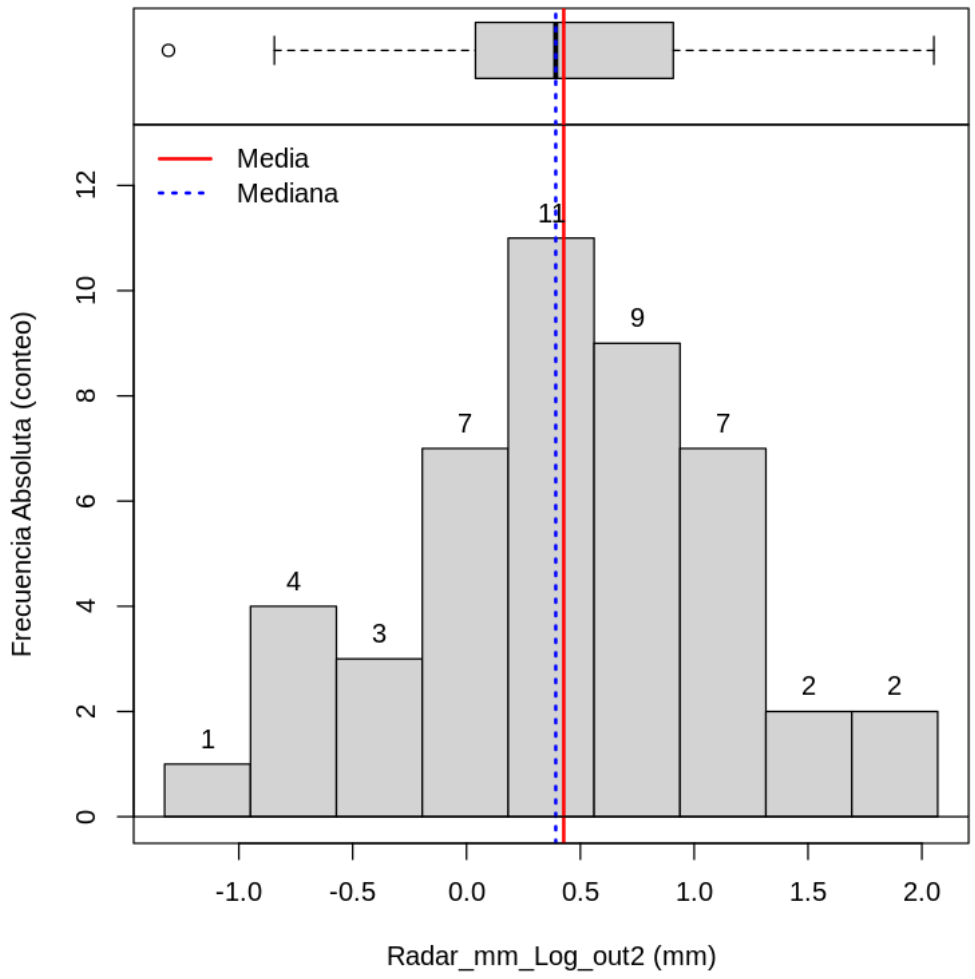
[112]:
```
# Variable without distributional outliers 2
Radar_mm_Log_out2<-Radar_mm_log_out_DF2$Radar_mm_Log

#2º análisis sin valores atípicos
# Estadística básica
Radar_mm_Log_out2_Stat<-Estadisticas(Radar_mm_Log_out2)
Radar_mm_Log_out2_Stat
```

A data.frame: 14 × 2

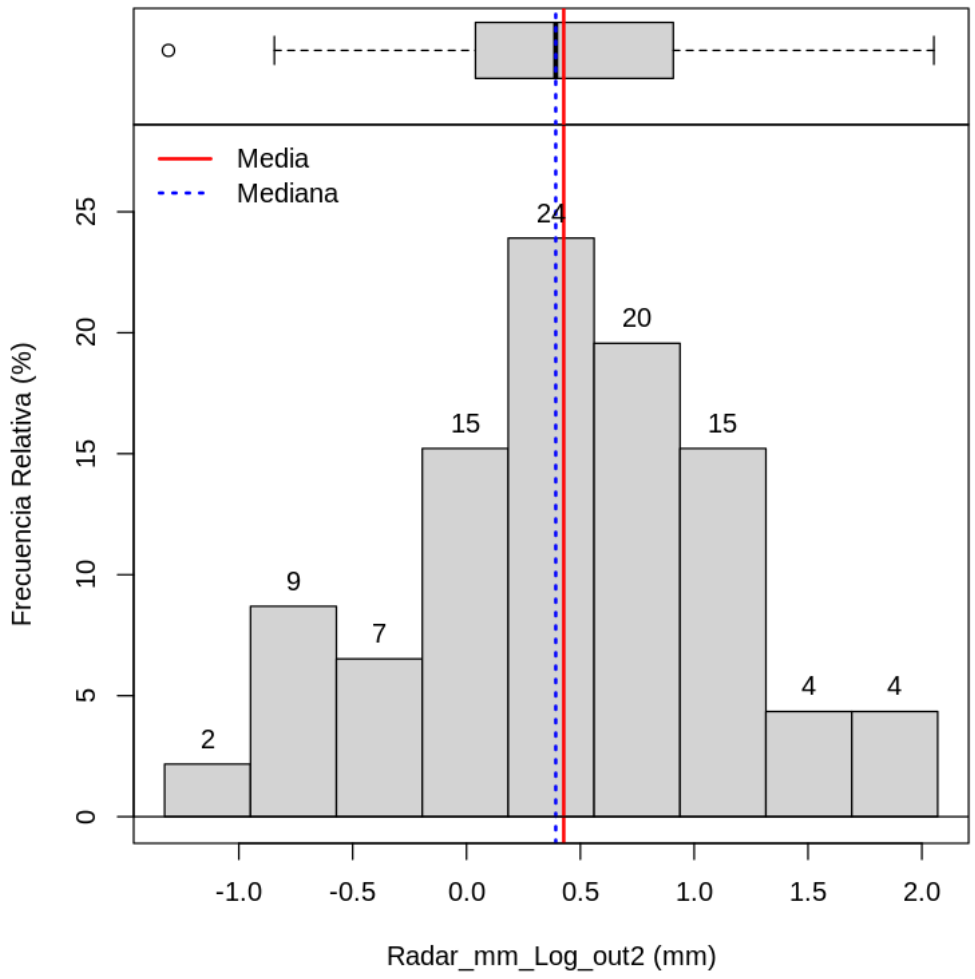
	Statistics <chr>	Values <dbl>
muestras	n	46.0000
minimos	Minimum	-1.3093
cuantiles1	1st. Quartile	0.0440
medianas	Median	0.3920
medias	Mean	0.4271
cuantiles3	3rd. Quartile	0.9011
maximos	Maximum	2.0528
rangos	Rank	3.3622
rangosInt	Interquartile Rank	0.8571
varianzas	Variance	0.5346
desvs	Standard Deviation	0.7312
CVs	Variation Coeff.	1.7118
simetrias	Skewness	-0.1103
curtosiss	Kurtosis	2.9190

Y graficamos su respectivo histograma.

[113]:
```
HistBoxplot(x=Radar_mm_Log_out2, mean = Radar_mm_Log_out2_Stat[5,2], median =␣
↪Radar_mm_Log_out2_Stat[4,2], main ="",
            xlab = "Radar_mm_Log_out2 (mm)", ylab = "Frecuencia Absoluta␣
↪(conteo)", AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



[114]:
```
HistBoxplot(x=Radar_mm_Log_out2, mean = Radar_mm_Log_out2_Stat[5,2], median =␣
↪Radar_mm_Log_out2_Stat[4,2], main ="",
            xlab = "Radar_mm_Log_out2 (mm)", ylab = "Frecuencia Relativa (%)",␣
↪AbsFreq = FALSE, PercentFreq = TRUE,
            nbin = 9)
```



[115]:
```
Comparative_log<-cbind(Radar_mm_Stat[2],Radar_mm_out_Stat[2],Radar_mm_Sqrt_Stat[2],Radar_mm_Sqr
↪Radar_mm_Log_Stat[2],
                       Radar_mm_Log_out_Stat[2],Radar_mm_Log_out2_Stat[2])
colnames(Comparative_log)<-c("Radar_mm","Radar_mm_out","Radar_mm_Sqrt","Radar_mm_Sqrt_out","Rad
Comparative_log
```

A data.frame: 14 × 7

	Radar_mm <dbl>	Radar_mm_out <dbl>	Radar_mm_Sqrt <dbl>	Radar_mm_Sqrt_out <dbl>
muestras	50.0000	48.0000	50.0000	48.0000
minimos	0.1800	0.1800	0.4243	0.4243
cuantiles1	0.8775	0.8050	0.9364	0.8961
medianas	1.4250	1.3500	1.1936	1.1618
medias	1.8350	1.5952	1.2511	1.1884
cuantiles3	2.3750	2.2475	1.5410	1.4992
maximos	7.7900	4.1900	2.7911	2.0469
rangos	7.6100	4.0100	2.3668	1.6227
rangosInt	1.4975	1.4425	0.6046	0.6031
varianzas	2.4735	1.1090	0.2754	0.1868
desvs	1.5727	1.0531	0.5247	0.4322
CVs	0.8571	0.6602	0.4194	0.3637
simetrias	2.0249	0.6937	0.7494	0.0273
curtosiss	7.9941	2.7723	3.8695	2.2704

La diferencia entre la media y la mediana de la transformación logarítmica quitando los valores atípicos detectados pasó de 0.0313 a 0.0351, es decir, aumentó el valor, sin ser significativo este aumento, con lo cual podemos considerar que hacer el segundo retiro de valores atípicos es innecesario.

3.1.2 Análisis estadístico univariado para los datos obtenidos de pluviómetros (Pluv_mm).

Ahora se hace el mismo análisis estadístico a los datos obtenidos de los pluviómetros (Pluv_mm). Empezamos obteniendo los valores estadísticos.

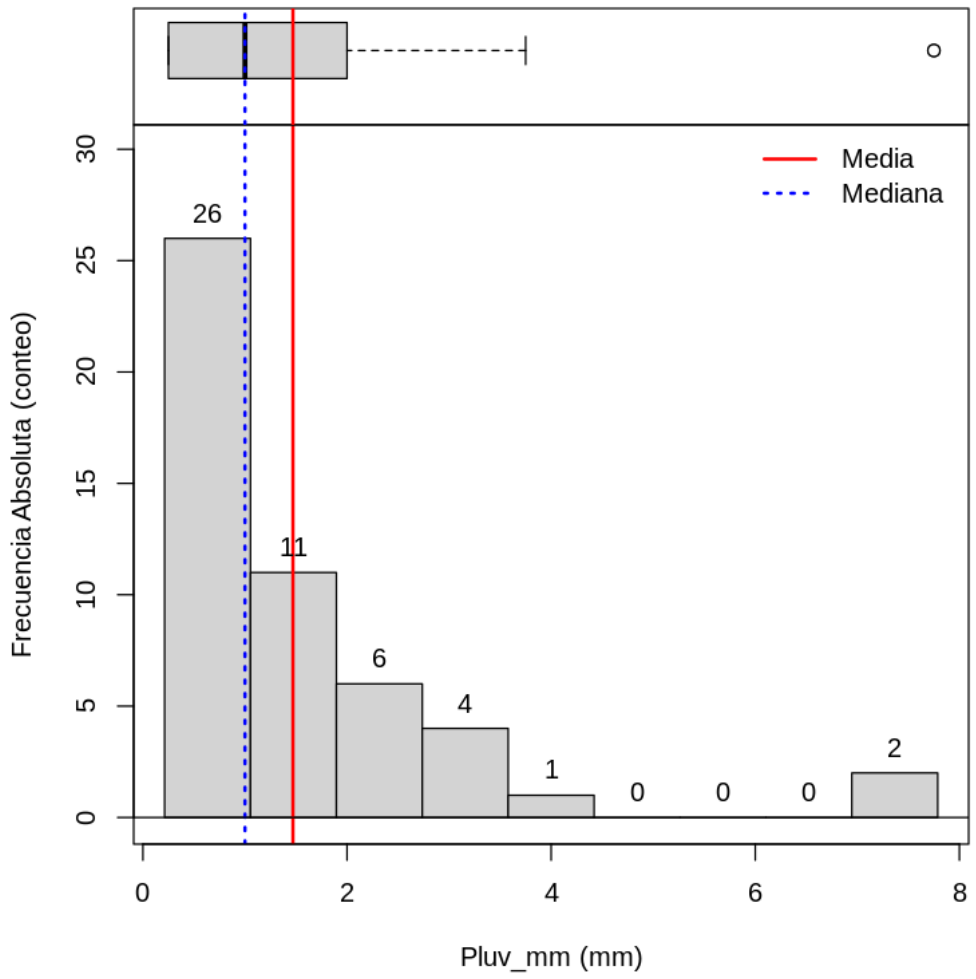
[116]:
```
Pluv_mm_Stat <- Estadisticas(Pluv_mm)
Pluv_mm_Stat
```

A data.frame: 14 × 2

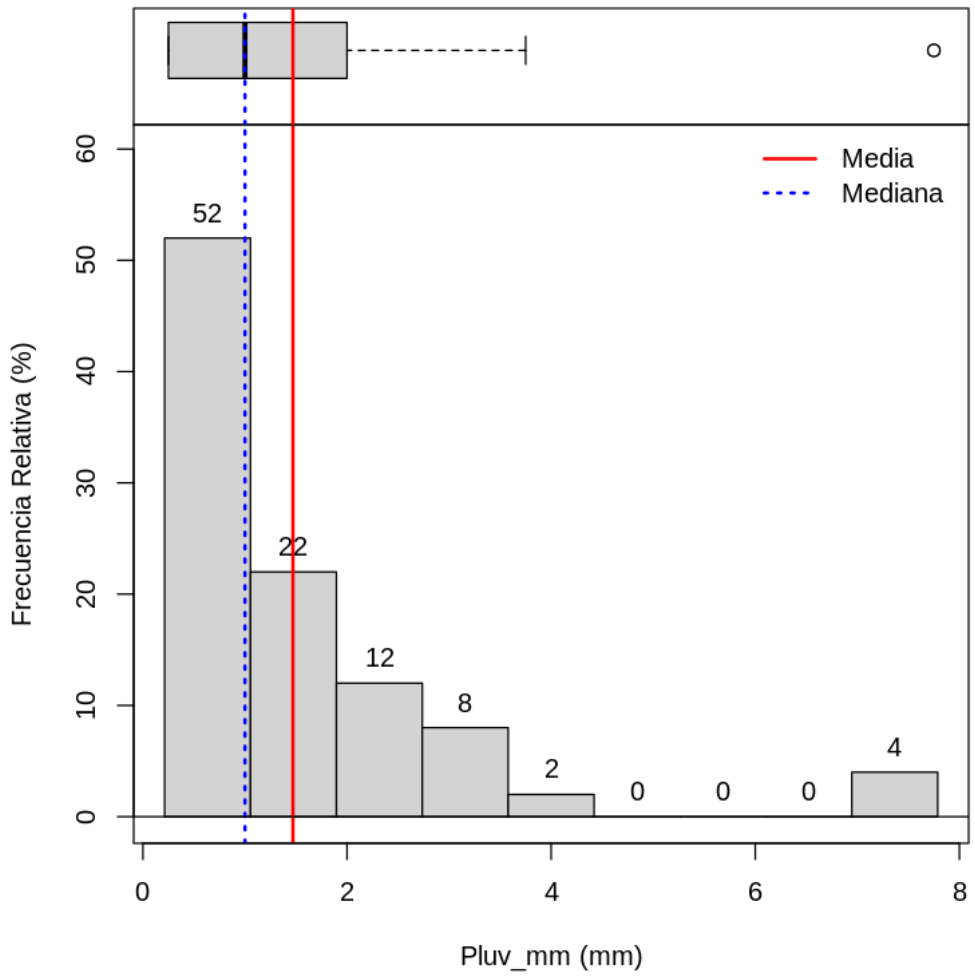
	Statistics <chr>	Values <dbl>
muestras	n	50.0000
minimos	Minimum	0.2500
cuantiles1	1st. Quartile	0.3125
medianas	Median	1.0000
medias	Mean	1.4700
cuantiles3	3rd. Quartile	1.9375
maximos	Maximum	7.7500
rangos	Rank	7.5000
rangosInt	Interquartile Rank	1.6250
varianzas	Variance	2.5756
desvs	Standard Deviation	1.6049
CVs	Variation Coeff.	1.0917
simetrias	Skewness	2.4693
curtosiss	Kurtosis	10.0542

Y su respectivo histograma.

[117]:
```
HistBoxplot(x=Pluv_mm, mean = Pluv_mm_Stat[5,2], median = Pluv_mm_Stat[4,2],␣
↪main ="",
            xlab = "Pluv_mm (mm)", ylab = "Frecuencia Absoluta (conteo)",␣
↪AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



[118]:
```
HistBoxplot(x=Pluv_mm, mean = Pluv_mm_Stat[5,2], median = Pluv_mm_Stat[4,2],␣
↪main ="",
            xlab = "Pluv_mm (mm)", ylab = "Frecuencia Relativa (%)", AbsFreq =␣
↪FALSE, PercentFreq = TRUE,
            nbin = 9)
```



Se puede notar que la diferencia entre la media y la mediana es de 0.47, lo cual nos indica que la variable es asimétrica positiva y el histograma nos confirma esta información.

Transformación logarítmica.

Analizando el histograma se observa un fuerte crecimiento en la parte izquierda, por lo tanto es recomendable usar la transformación logarítmica.

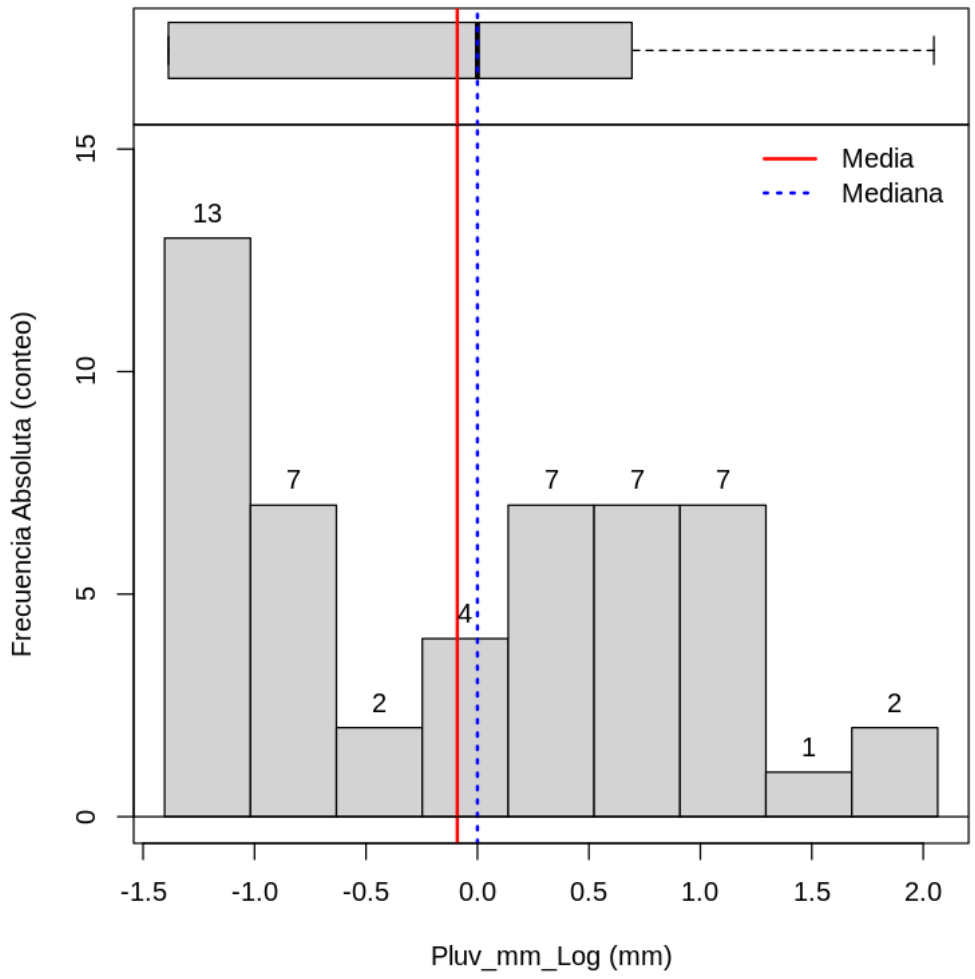
[119]:
```
Pluv_mm_Log <- log(Pluv_mm)
Pluv_mm_Log_Stat <- Estadisticas(Pluv_mm_Log)
Pluv_mm_Log_Stat
```

A data.frame: 14 × 2

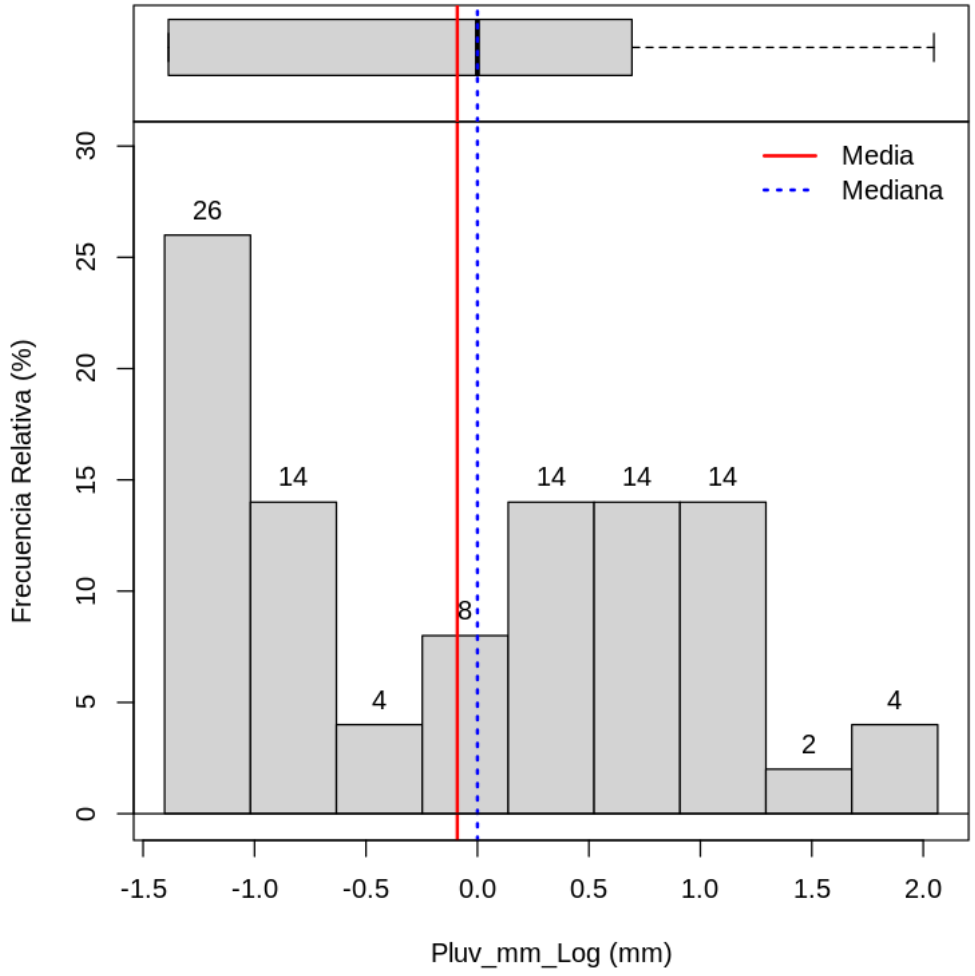
	Statistics <chr>	Values <dbl>
muestras	n	50.0000
minimos	Minimum	-1.3863
cuantiles1	1st. Quartile	-1.2130
medianas	Median	0.0000
medias	Mean	-0.0903
cuantiles3	3rd. Quartile	0.6598
maximos	Maximum	2.0477
rangos	Rank	3.4340
rangosInt	Interquartile Rank	1.8728
varianzas	Variance	1.0004
desvs	Standard Deviation	1.0002
CVs	Variation Coeff.	-11.0754
simetrias	Skewness	0.1124
curtosiss	Kurtosis	2.0120

Los histogramas de la transformación logarítmica son los siguientes:

[120]:
```
HistBoxplot(x=Pluv_mm_Log, mean = Pluv_mm_Log_Stat[5,2], median =␣
↪Pluv_mm_Log_Stat[4,2], main ="",
            xlab = "Pluv_mm_Log (mm)", ylab = "Frecuencia Absoluta (conteo)",␣
↪AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



[121]:
```
HistBoxplot(x=Pluv_mm_Log, mean = Pluv_mm_Log_Stat[5,2], median =␣
↪Pluv_mm_Log_Stat[4,2], main ="",
            xlab = "Pluv_mm_Log (mm)", ylab = "Frecuencia Relativa (%)", AbsFreq␣
↪= FALSE, PercentFreq = TRUE,
            nbin = 9)
```



La diferencia entre la media y la mediana pasó de 0.47 a 0.09, lo cual es bajo. El boxplot no muestra valores atípicos, sin embargo, nos aseguraremos que esto sea cierto usando la función "OutliersPos".

[122]:
```
Pluv_mm_Log_outliers<-OutliersPos(Pluv_mm_Log)
print(Pluv_mm_Log_outliers)
```

```
numeric(0)
```

No se encontraron valores atípicos, por lo tanto, aquí se termina el análisis univariado de esta variable.

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:

$$\tau = \frac{\textit{Número de pares concordantes - Número de pares discordante}}{\binom{n}{2}}$$

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 (Radar_mm , Pluv_mm) y el método que se desea usar, este puede ser Pearson, Spearman o Kendall.

[123]:
```
cor(Radar_mm , Pluv_mm, method = "pearson")
```

0.941048488234573

[124]:
```
cor(Radar_mm , Pluv_mm, method = "spearman")
```

0.848045300430134

[125]:
```
cor(Radar_mm , Pluv_mm, method = "kendall")
```

0.718715084238533

Respecto al coeficiente de Pearson, su valor es de 0.9410, lo cual podemos considerar como cuasi-lineal. Sin embargo, los valores de la correlación de Spearman (0.8480) y Kendall (0.7187) indican que el modelo no es cuasi-lineal, por lo que es posible que la dependencia lineal esté alterada por valores atípicos.

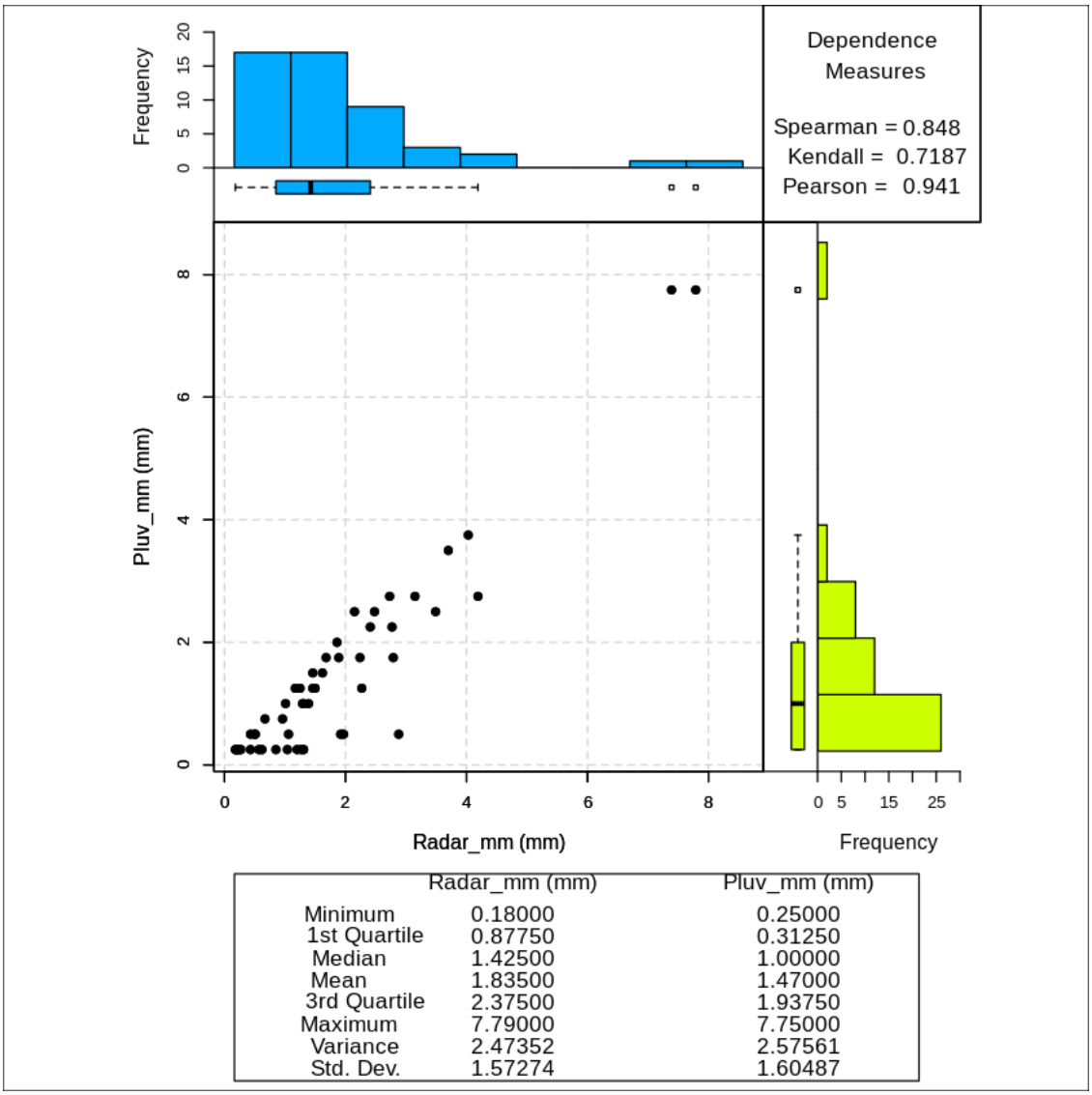
3.2.2 Diagrama de dispersión.

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

- Variables (Radar_mm , Pluv_mm)
- 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)

[126]:
```
ScatterPlot(Radar_mm , Pluv_mm, 9,
            Xmin = Radar_mm_Stat[2,2], Xmax = Radar_mm_Stat[7,2],
            Ymin = Pluv_mm_Stat[2,2],Ymax = Pluv_mm_Stat[7,2],
            XLAB = "Radar_mm (mm)", YLAB = "Pluv_mm (mm)")
```



	Radar_mm (mm)	Pluv_mm (mm)
Minimum	0.18000	0.25000
1st Quartile	0.87750	0.31250
Median	1.42500	1.00000
Mean	1.83500	1.47000
3rd Quartile	2.37500	1.93750
Maximum	7.79000	7.75000
Variance	2.47352	2.57561
Std. Dev.	1.57274	1.60487

Podemos notar en el gráfico de dispersión que hay dos pares atípicos localizados en la esquina superior derecha, los cuales podrían ser la causa de la cuasi-linearidad que indica la medida de dependencia de Pearson. Estos valores coinciden con los valores atípicos que se detectaron durante el análisis univariado, Sin embargo, es importante señalar que los valores atípicos encontrados en el análisis exploratorio univariado no son necesariamente valores atípicos en el análisis exploratorio

bivariado, se debe evaluar la conveniencia de retirar cada par.

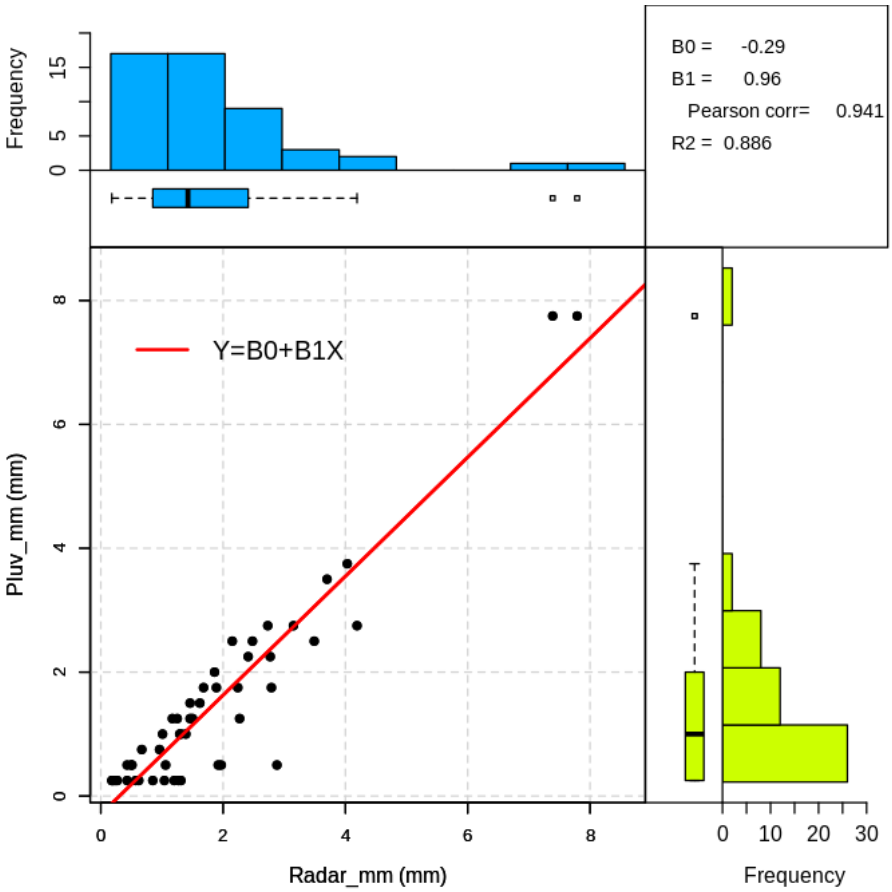
3.2.3 Analisis de regresión lineal.

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 (Radar_mm , Pluv_mm)
- 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)

[127]:
```
scaterplotReg(Radar_mm , Pluv_mm, 9,
              Xmin = Radar_mm_Stat[2,2], Xmax = Radar_mm_Stat[7,2],
              Ymin = Pluv_mm_Stat[2,2],Ymax = Pluv_mm_Stat[7,2],
              XLAB = "Radar_mm (mm)", YLAB = "Pluv_mm (mm)")
```



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

[128]:
```
# Linear Regression
X<-Radar_mm
Y<-Pluv_mm

linear_regression <-lm(Y ~ X)

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

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

(Intercept): -0.292098951631834

X: 0.960271908246231

Con los parámetros de la recta B_0 y B_1, hacemos el calculo de los residuos.

[129]:
```
# Regression line and Residual Calculation
Y_Regression <- linear_regression$fitted.values
Y_Residual <- linear_regression$residuals
```

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

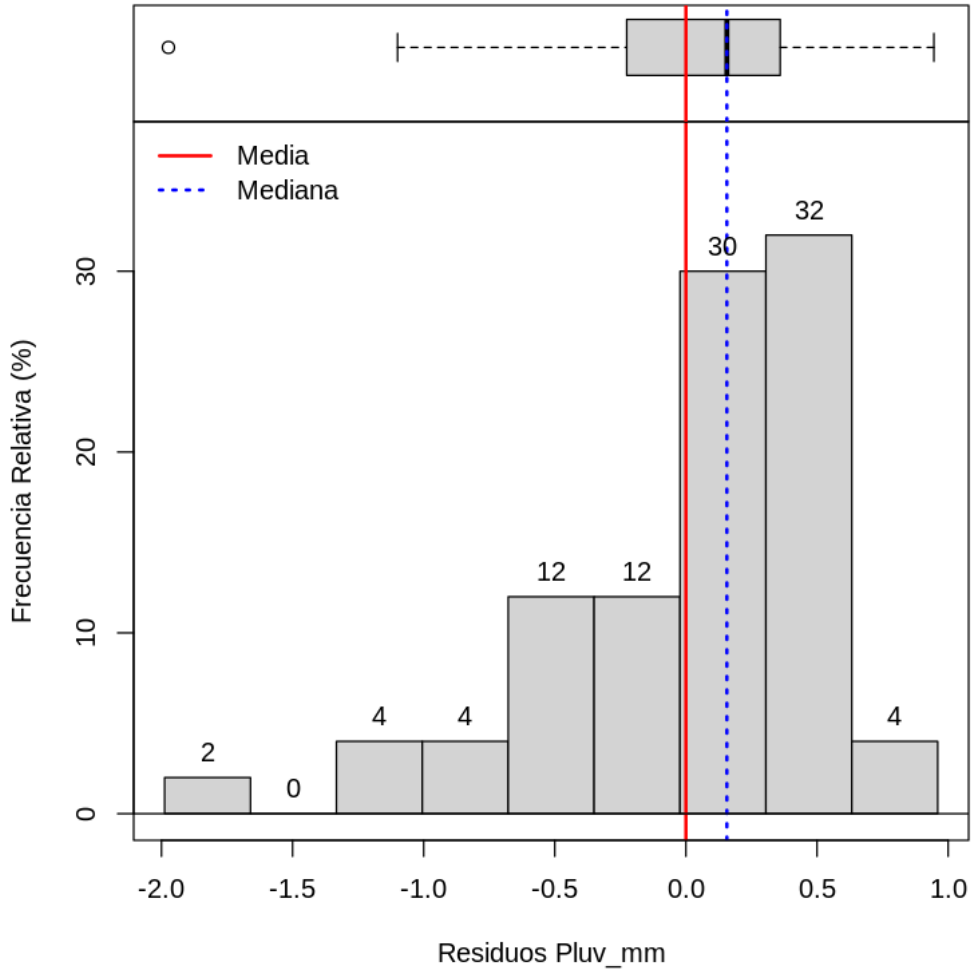
[130]:
```
Y_Residual_Stat<-Estadisticas(Y_Residual)
write.csv(Y_Residual_Stat , file = "Results/AED/Pluv_mm_Residual_Stat.csv")
Y_Residual_Stat
```

A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	5.000000e+01
minimos	Minimum	-1.973500e+00
cuantiles1	1st. Quartile	-1.988000e-01
medianas	Median	1.562000e-01
medias	Mean	0.000000e+00
cuantiles3	3rd. Quartile	3.552000e-01
maximos	Maximum	9.457000e-01
rangos	Rank	2.919200e+00
rangosInt	Interquartile Rank	5.540000e-01
varianzas	Variance	2.947000e-01
desvs	Standard Deviation	5.429000e-01
CVs	Variation Coeff.	-2.186936e+16
simetrias	Skewness	-1.347400e+00
curtosiss	Kurtosis	5.138200e+00

Y tambien necesitamos obtener el histograma de estos residuos.

[131]:
```
HistBoxplot(x=Y_Residual, mean = Y_Residual_Stat[5,2], median =␣
↪Y_Residual_Stat[4,2], main ="",
            xlab = "Residuos Pluv_mm", ylab = "Frecuencia Relativa (%)", AbsFreq␣
↪= FALSE, PercentFreq = TRUE )
```



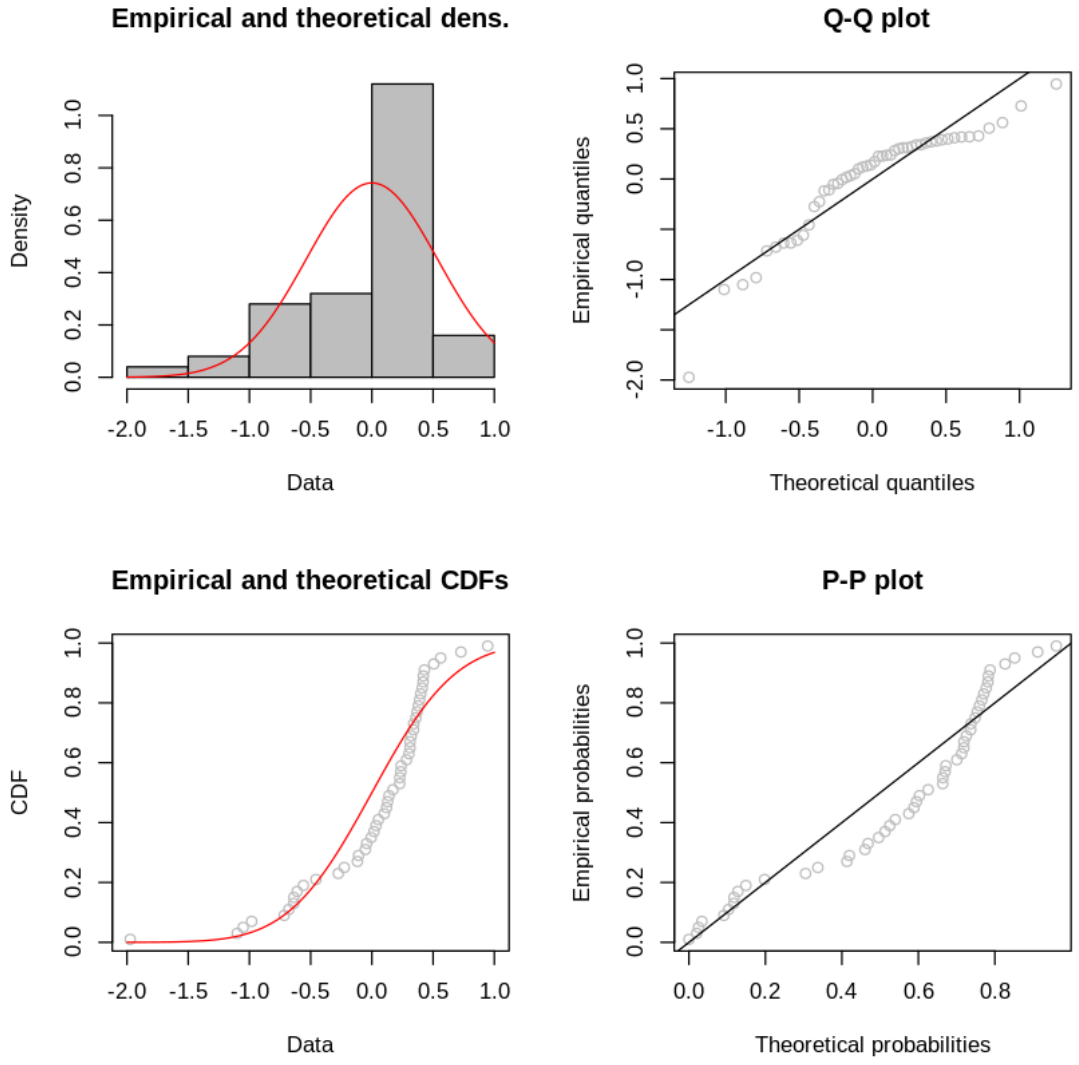
Al analizar los valores estadísticos y el histograma de los residuos, se observa que el valor esperado es de 0, su varianza es de 0.2947 y la diferencia entre la media y la mediana es de 0.1562, lo cual nos indica que tiene asimetría negativa, por lo tanto, los residuos no cumplen con todas las condiciones 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 empirica 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)

[132]:
```
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 unas pocas muestras están en la recta. La grafica comparativa entre las funciones de distribución acumulativas empírica y teórica (figura inferior izquierda) no muestran un buen ajuste Y en el caso del grafico P-P plot (figura inferior derecha) solo las muestras de la esquina inferior izquierda y unas pocas de la parte central 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.

[133]:
```
FD_HT_Residual_normal<-FitDistr2_Residual_normal$x
FD_HT_Residual_normal
```

A data.frame: 2 × 5

Method <chr>	Significance level <chr>	P-value <chr>	Statistical <chr>	Decision <chr>
Kolmogorov-Smirnov	0.05	0.136	0.1605	No rechazo H0
Anderson-Darling	0.05	0.07826	2.13	No rechazo H0

Bajo la prueba de Kolmogorov-Smirnov podemos ver que la normalidad de los residuos es de no rechazo. Mientras que la prueba de Anderson-Darling no rechaza la hipótesis de normalidad. Sin embargo, los valores obtenidos de estas pruebas son muy cercanos al nivel de significancia, por lo que podríamos reforzar la evidencia de no normalidad de la variable.

[134]:
```
FD_FP_Residual_normal<-FitDistr2_Residual_normal$y
FD_FP_Residual_normal
```

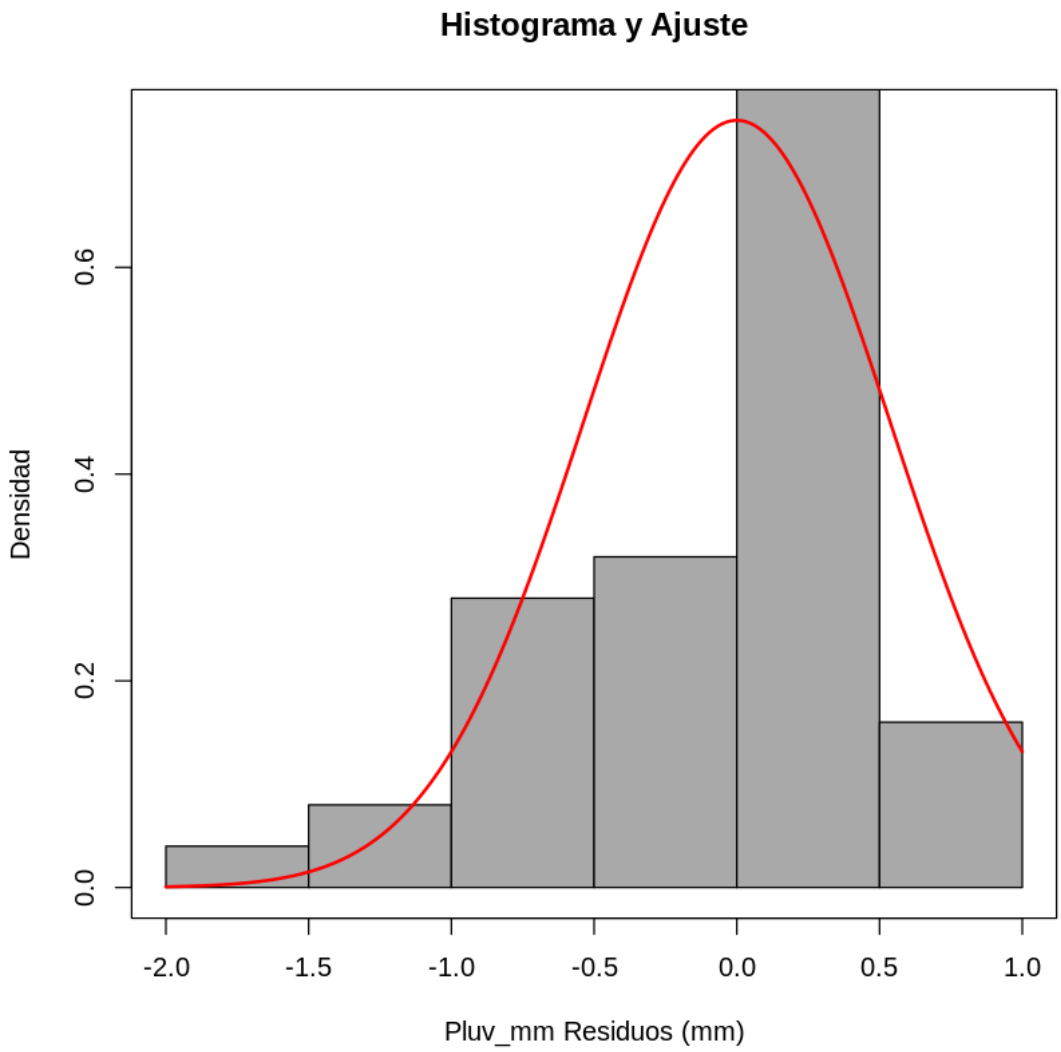
A data.frame: 4 × 1

	Normal <dbl>
Mean	-2.482389e-17
Standard deviation	5.374263e-01
Maximum likelihood	-3.989875e+01
AIC	8.379750e+01

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)

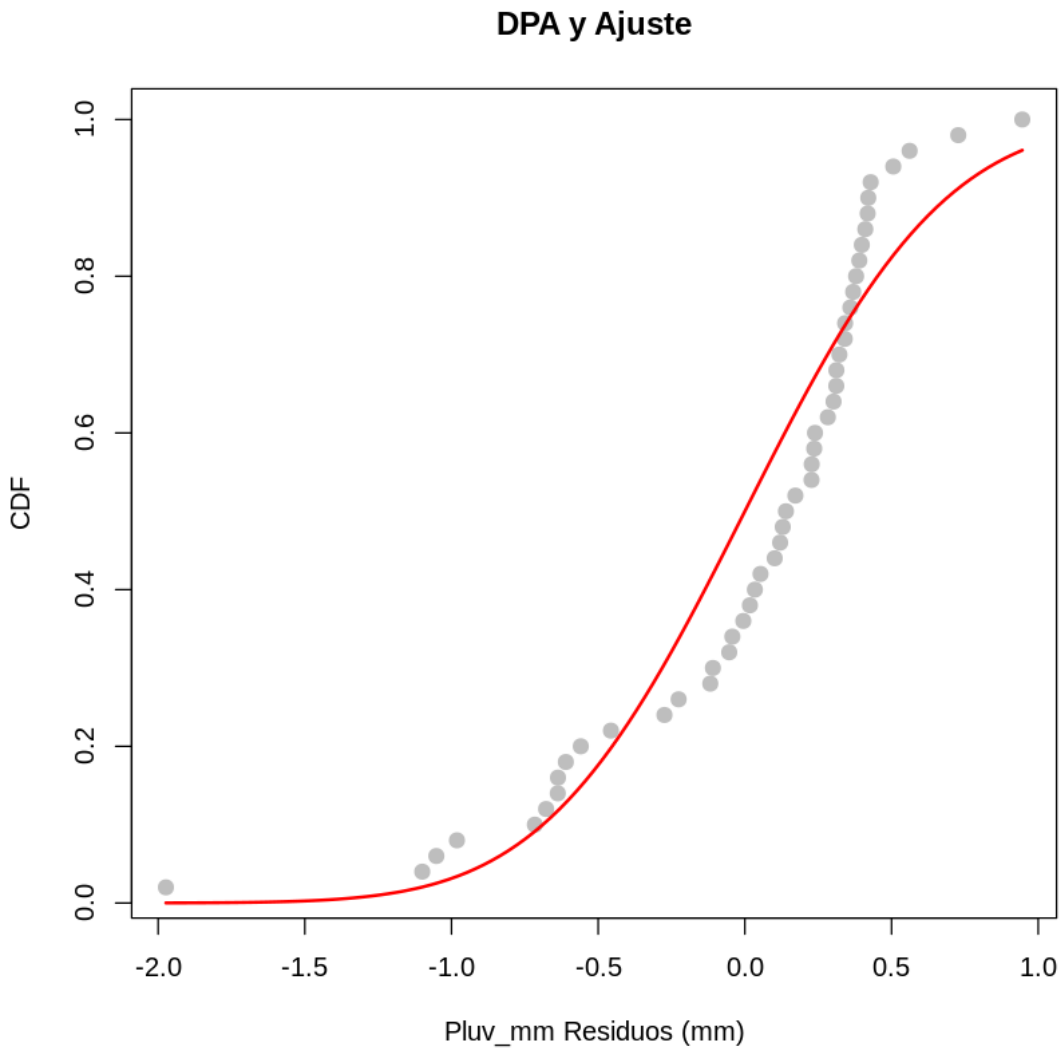
[135]:
```
PARA_Residual_normal <- list(mean = as.numeric(FD_FP_Residual_normal[1,1]),
                             sd = as.numeric(FD_FP_Residual_normal[2,1]))

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



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)

[136]:
```
CDF(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray",␣
↪main = "DPA y Ajuste",
    xlab = "Pluv_mm Residuos (mm)", lcol = "red", lwd = 2)
```

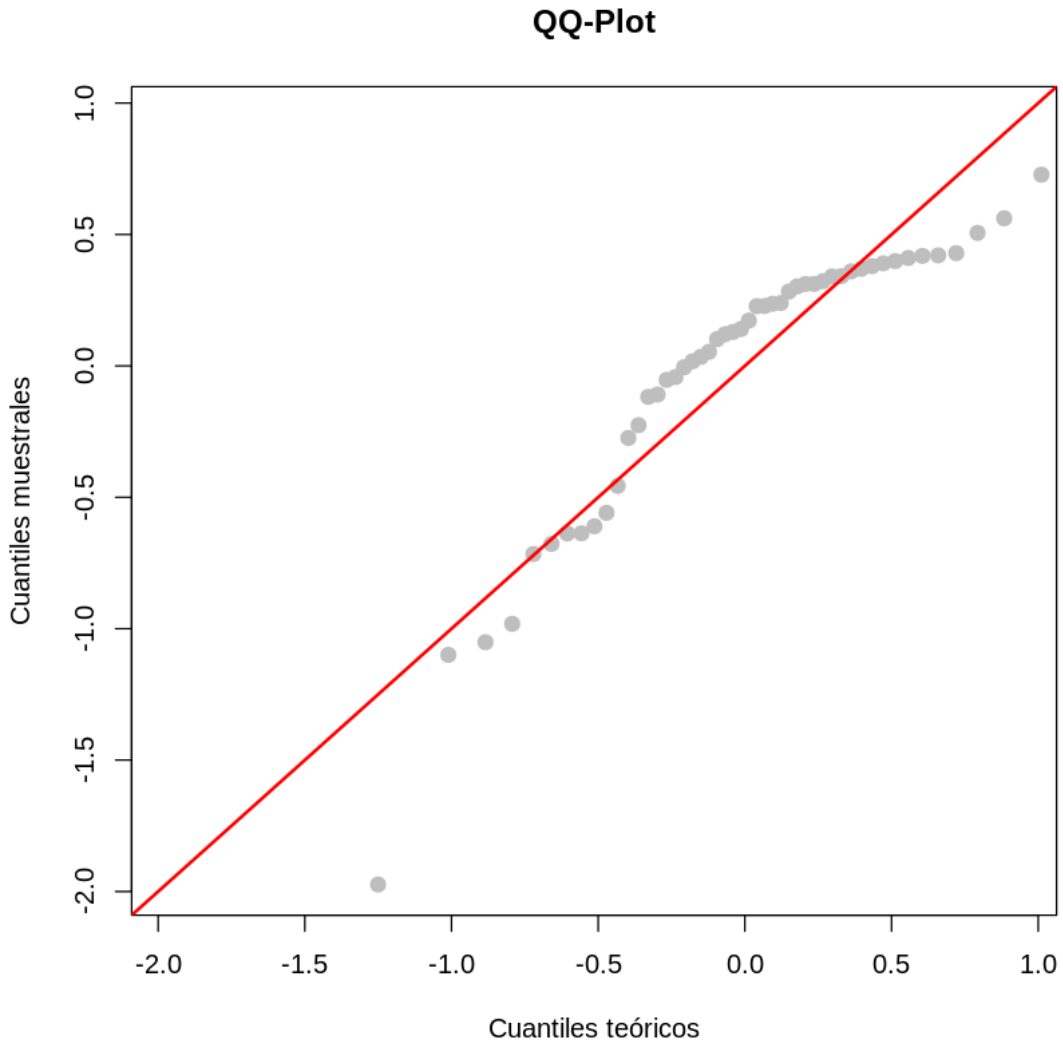


El Grafico cuantil-cuantil (Q-Q plot) se puede graficar usando la función "QQplot", esta requiere de los siguientes parámetros:

- 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)

[137]:
```
QQplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col =␣
↪"gray", main = "QQ-Plot",
      xlab = "Cuantiles teóricos", lcol = "red", lwd = 2)
```

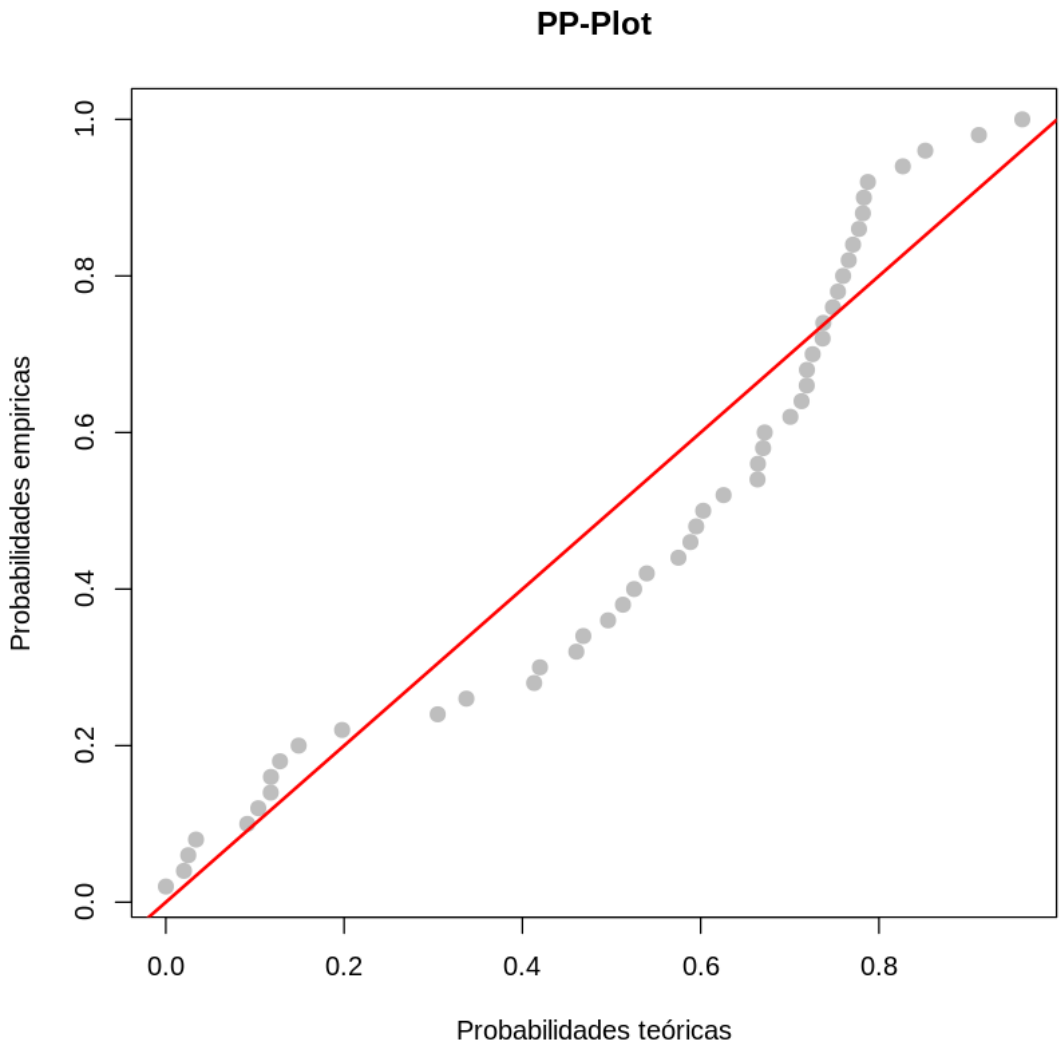


El Grafico percentil-percentil (P-P plot) se puede graficar usando la función "PPplot", esta requiere de los siguientes parámetros:

- 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)

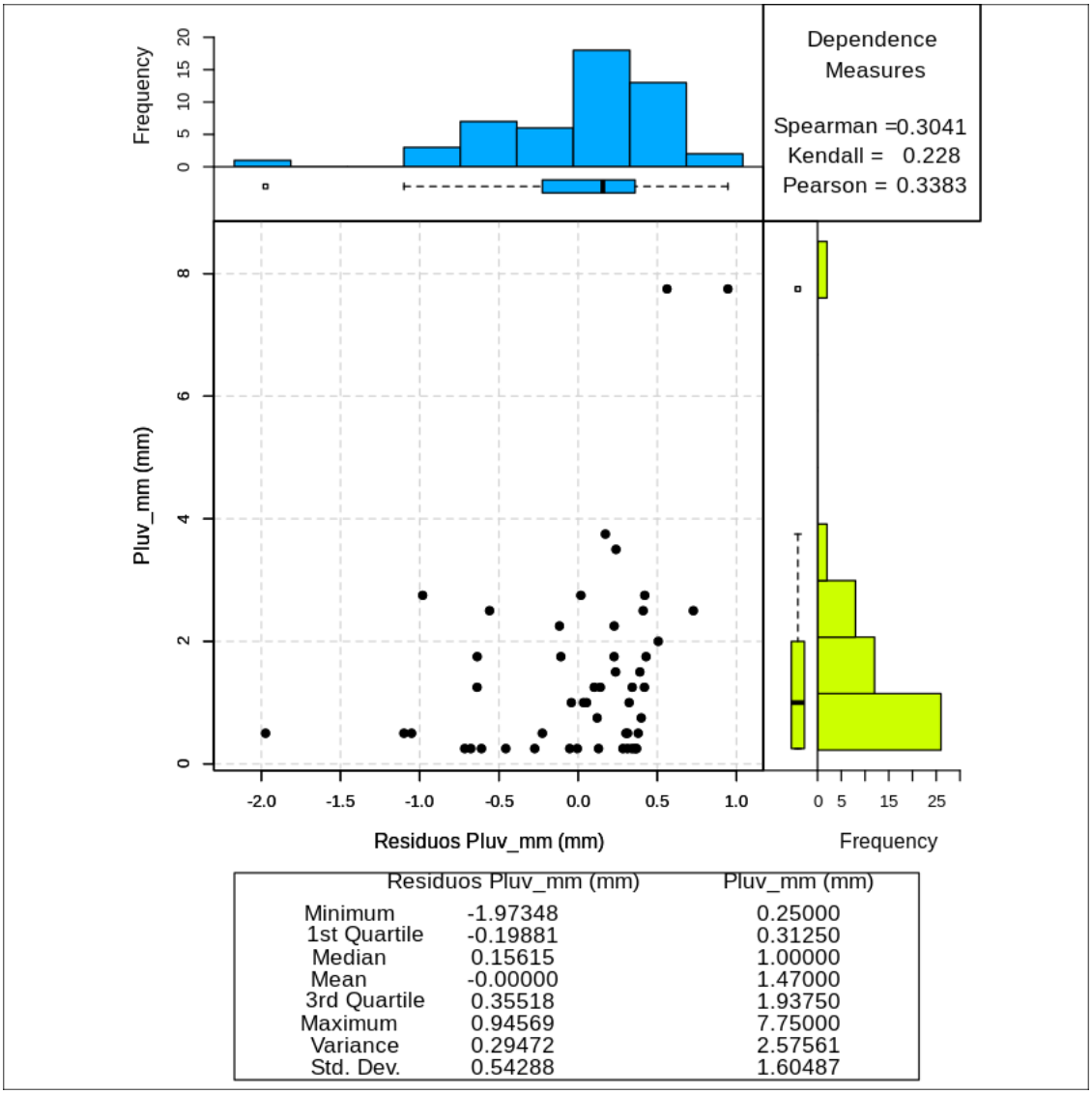
[138]:
```
PPplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col =␣
↪"gray", main = "PP-Plot",
      xlab = "Probabilidades teóricas", lcol = "red", lwd = 2)
```



3.2.4 Análisis bivariado: Y vs Y residual.

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

[139]:
```
X<-Y_Residual    # Y_Residual is the independent variable

Y<-Pluv_mm       # Pluv_mm is the dependent variable

ScatterPlot(X, Y, 9,
            Xmin = Y_Residual_Stat[2,2], Xmax = Y_Residual_Stat[7,2],
            Ymin = Pluv_mm_Stat[2,2],Ymax = Pluv_mm_Stat[7,2], XLAB = "Residuos␣
↪Pluv_mm (mm)", YLAB = "Pluv_mm (mm)")
```



	Residuos Pluv_mm (mm)	Pluv_mm (mm)
Minimum	-1.97348	0.25000
1st Quartile	-0.19881	0.31250
Median	0.15615	1.00000
Mean	-0.00000	1.47000
3rd Quartile	0.35518	1.93750
Maximum	0.94569	7.75000
Variance	0.29472	2.57561
Std. Dev.	0.54288	1.60487

En este grafico podemos notar que la medida de dependencia lineal de Pearson es de 0.339, Spearman es de 0.3041 y Kendall es de 0.228. Esto nos da como conclusión que la regresión lineal no cumple con la condición no correlación entre los residuos y las muestras del pluviómetro.

3.3 Análisis estadístico bivariado con variables transformadas.

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

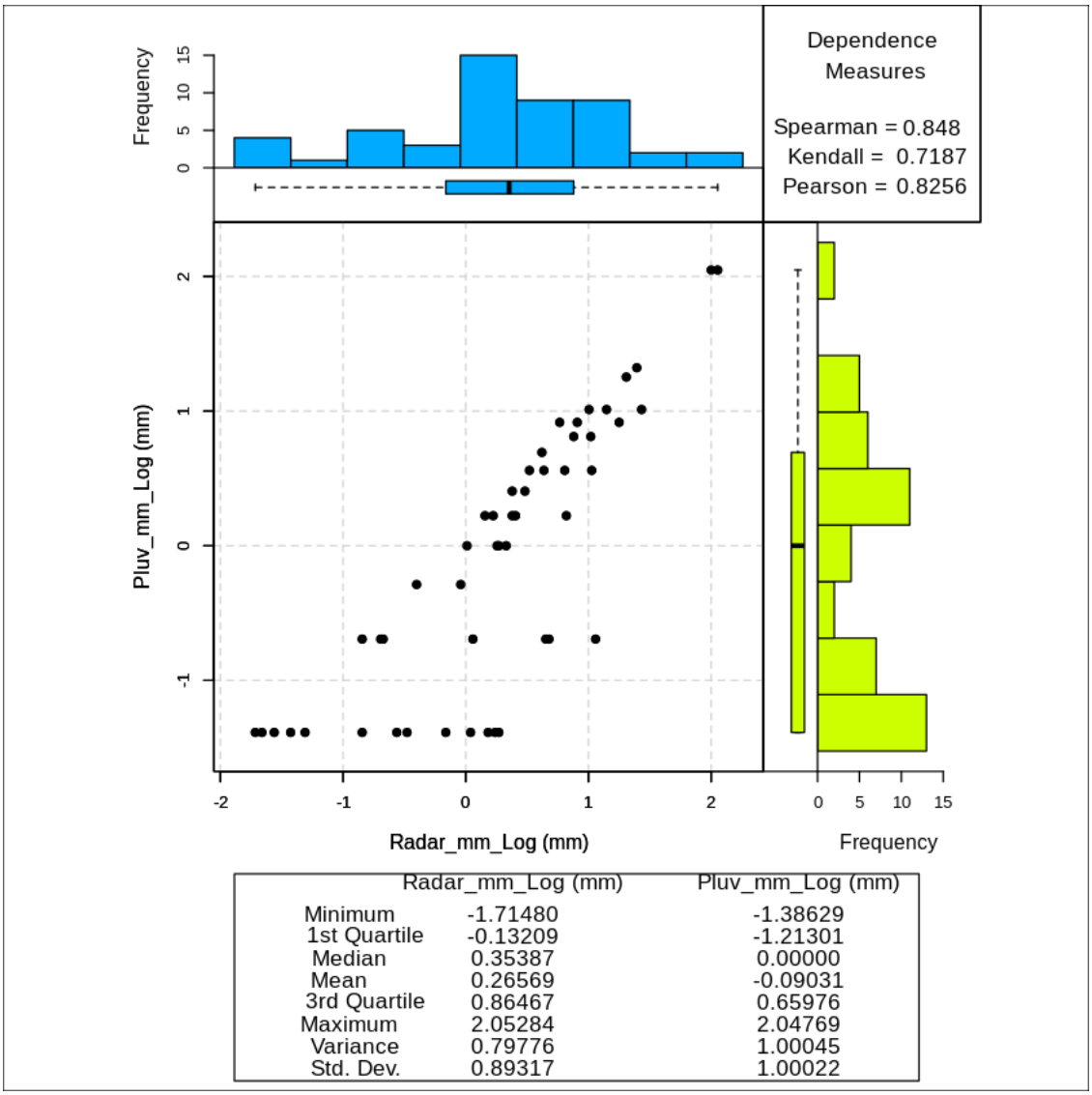
3.3.1 Grafico de dispersión.

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

- Spearman (0.848)
- Kendall (0.7187)
- Pearson (0.8256)

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 0.9410 a 0.8256, 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.

[140]:
```
ScatterPlot(Radar_mm_Log , Pluv_mm_Log, 9,
            Xmin = Radar_mm_Log_Stat[2,2], Xmax = Radar_mm_Log_Stat[7,2],
            Ymin = Pluv_mm_Log_Stat[2,2],Ymax = Pluv_mm_Log_Stat[7,2],
            XLAB = "Radar_mm_Log (mm)", YLAB = "Pluv_mm_Log (mm)")
```

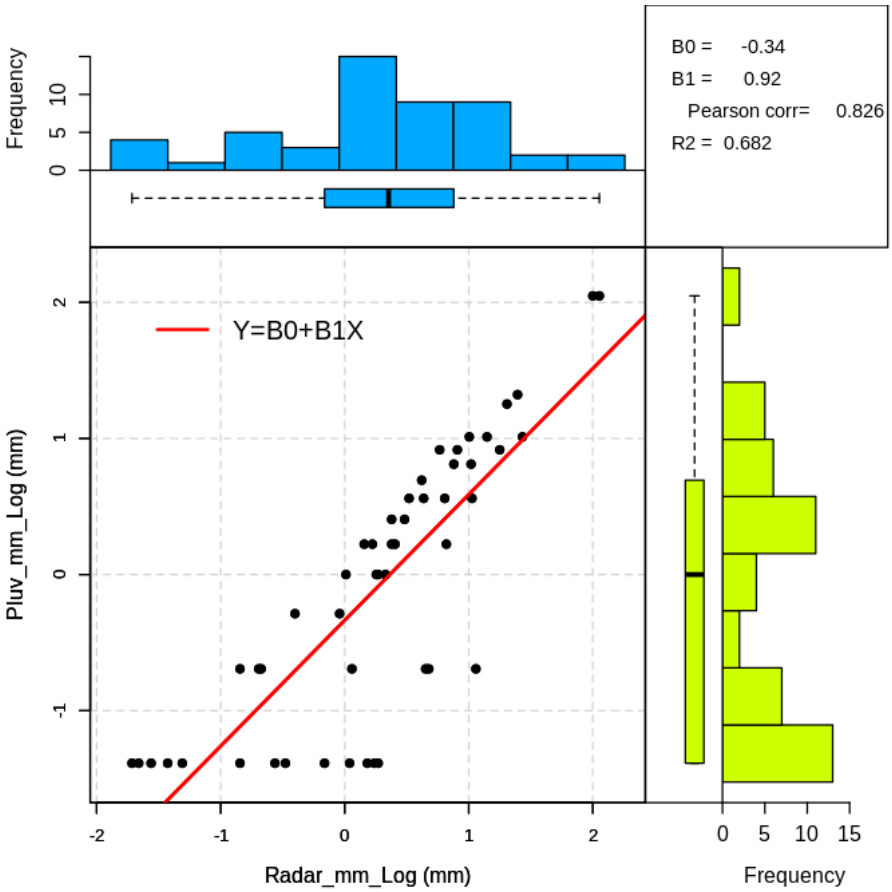


	Radar_mm_Log (mm)	Pluv_mm_Log (mm)
Minimum	-1.71480	-1.38629
1st Quartile	-0.13209	-1.21301
Median	0.35387	0.00000
Mean	0.26569	-0.09031
3rd Quartile	0.86467	0.65976
Maximum	2.05284	2.04769
Variance	0.79776	1.00045
Std. Dev.	0.89317	1.00022

3.3.2 Análisis de regresión lineal.

Ahora obtendremos los valores de la regresión lineal.

[141]:
```
scaterplotReg(Radar_mm_Log , Pluv_mm_Log, 9,
              Xmin = Radar_mm_Log_Stat[2,2], Xmax = Radar_mm_Log_Stat[7,2],
              Ymin = Pluv_mm_Log_Stat[2,2],Ymax = Pluv_mm_Log_Stat[7,2],
              XLAB = "Radar_mm_Log (mm)", YLAB = "Pluv_mm_Log (mm)")
```



los valores de la regresión lineal son:

[142]:
```
# Linear Regression
X<-Radar_mm_Log
Y<-Pluv_mm_Log

linear_regression <-lm(Y ~ X)

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

(Intercept): -0.335971802729908

X: 0.924607667074211

Y calculamos sus estadígrafos e histograma.

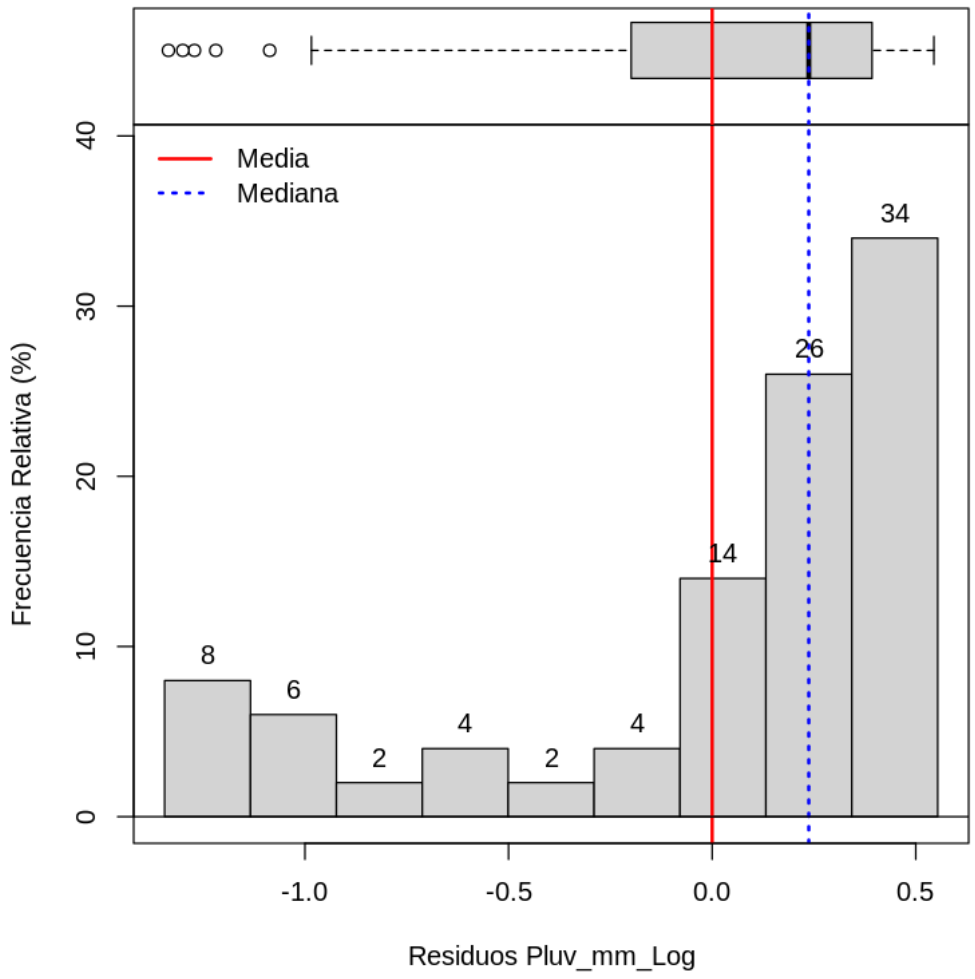
[144]:
```
# Regression line and Residual Calculation
Y_Regression <- linear_regression$fitted.values
Y_Residual <- linear_regression$residuals

Y_Residual_Stat<-Estadisticas(Y_Residual)
write.csv(Y_Residual_Stat , file = "Results/AED/Pluv_mm_Log_Residual_Stat.csv")
Y_Residual_Stat
```

A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	5.000000e+01
minimos	Minimum	-1.335200e+00
cuantiles1	1st. Quartile	-1.624000e-01
medianas	Median	2.373000e-01
medias	Mean	0.000000e+00
cuantiles3	3rd. Quartile	3.924000e-01
maximos	Maximum	5.445000e-01
rangos	Rank	1.879700e+00
rangosInt	Interquartile Rank	5.548000e-01
varianzas	Variance	3.184000e-01
desvs	Standard Deviation	5.643000e-01
CVs	Variation Coeff.	-1.393905e+17
simetrias	Skewness	-1.264700e+00
curtosiss	Kurtosis	3.224800e+00

Y también se grafican sus histogramas.

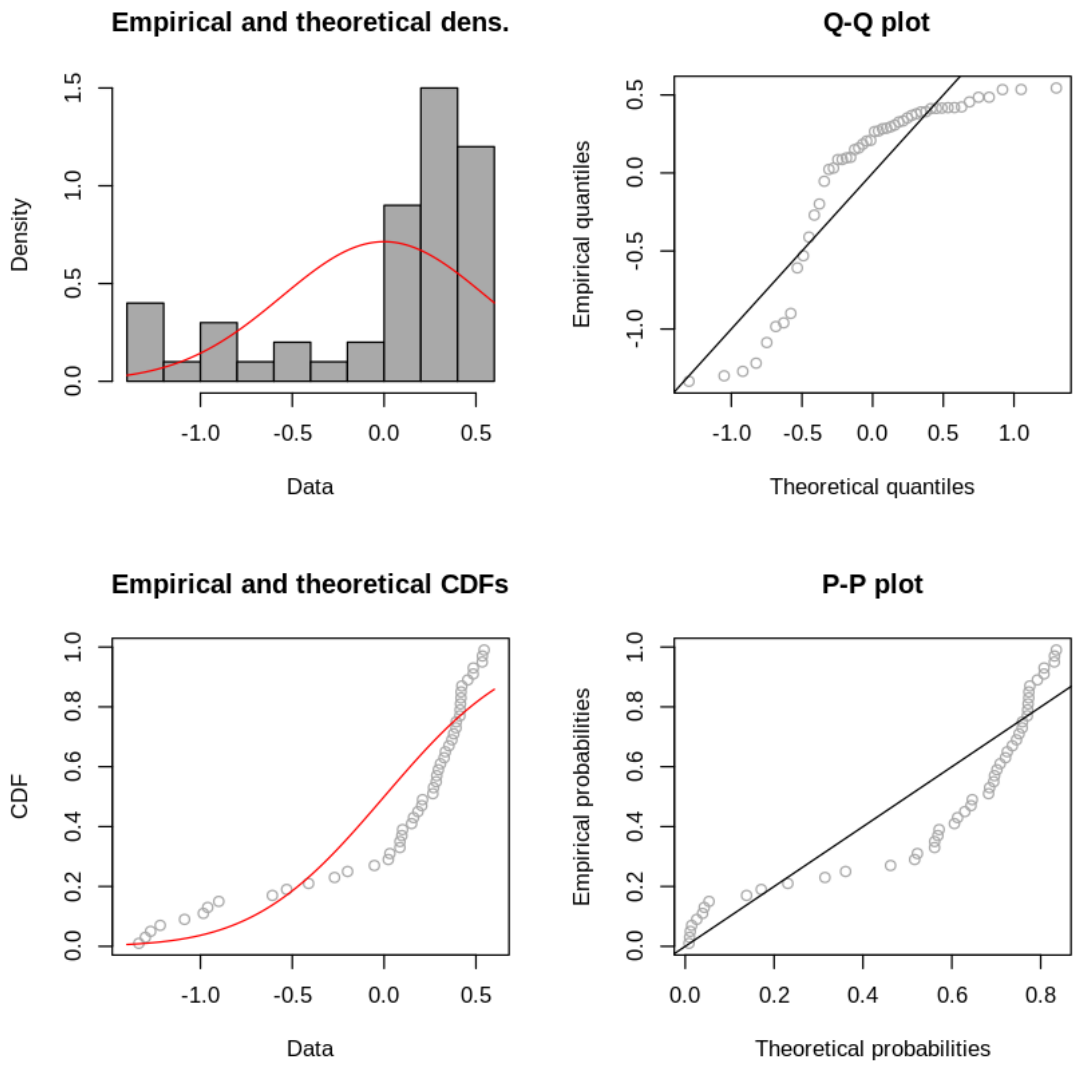
[145]:
```
HistBoxplot(x=Y_Residual, mean = Y_Residual_Stat[5,2], median =␣
↪Y_Residual_Stat[4,2], main ="",
            xlab = "Residuos Pluv_mm_Log", ylab = "Frecuencia Relativa (%)",␣
↪AbsFreq = FALSE, PercentFreq = TRUE )
```



En este caso podemos ver que el histograma presenta asimetría negativa con cuatro valores atípicos localizados a la izquierda. La diferencia entre la media y la mediana es de 0.2373, la cual es alta.

[146]:
```
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 a los extremos del histograma. El grafico Q-Q plot (figura superior derecha) también muestra que pocas muestras están en la recta. La grafica comparativa entre las funciones de distribución acumulativas empírica y teórica (figura inferior izquierda) no muestra un buen ajuste y en el caso del grafico P-P plot (figura inferior derecha) hay muy pocas muestras cercanas a la recta. Con esto podemos concluir que los residuos no cumplen con los requisitos de la regresión lineal.

[147]:
```
FD_HT_Residual_normal<-FitDistr2_Residual_normal$x
FD_HT_Residual_normal
```

A data.frame: 2 × 5

Method <chr>	Significance level <chr>	P-value <chr>	Statistical <chr>	Decision <chr>
Kolmogorov-Smirnov	0.05	0.004802	0.2412	Rechazo H0
Anderson-Darling	0.05	0.006424	4.282	Rechazo H0

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

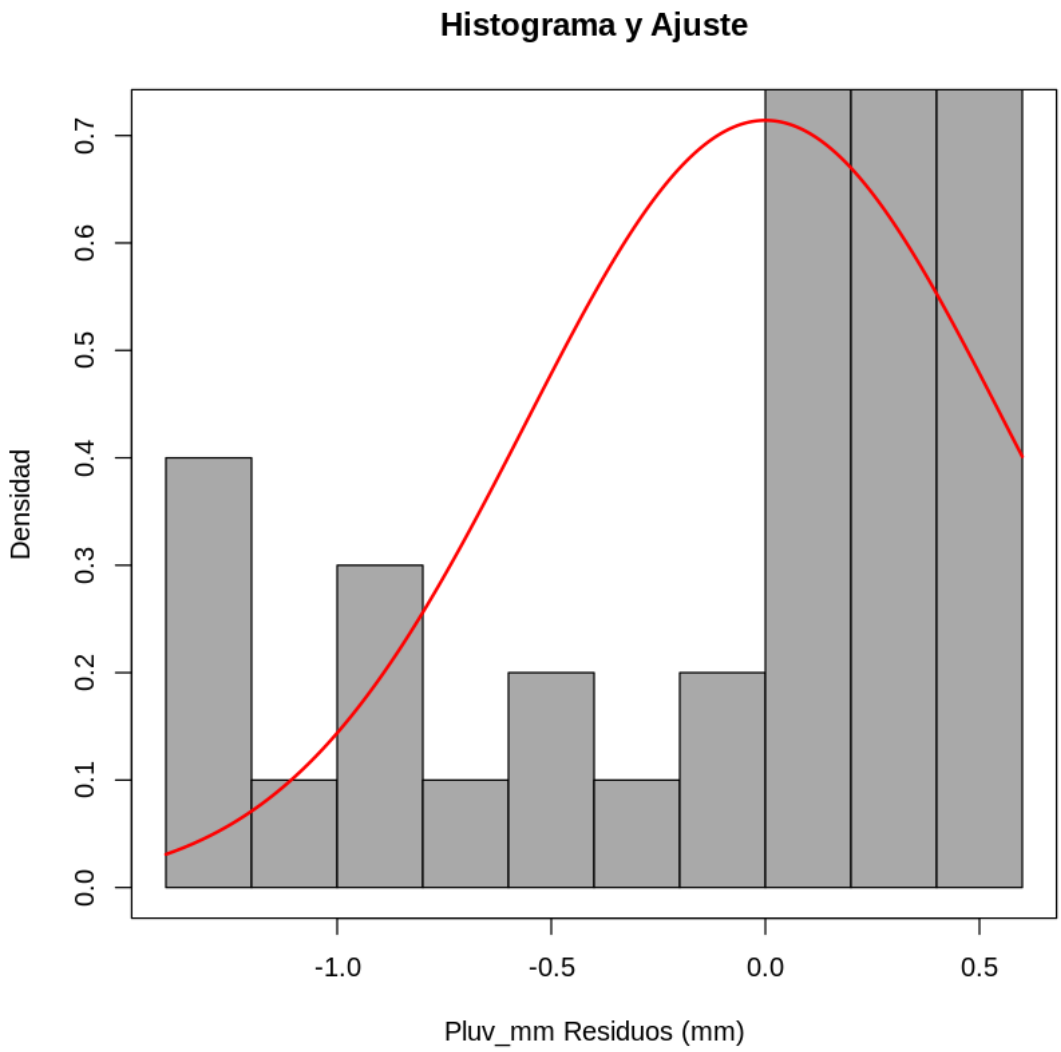
[148]:
```
FD_FP_Residual_normal<-FitDistr2_Residual_normal$y
FD_FP_Residual_normal
```

A data.frame: 4 × 1

	Normal <dbl>
Mean	-4.048411e-18
Standard deviation	5.586384e-01
Maximum likelihood	-4.183428e+01
AIC	8.766856e+01

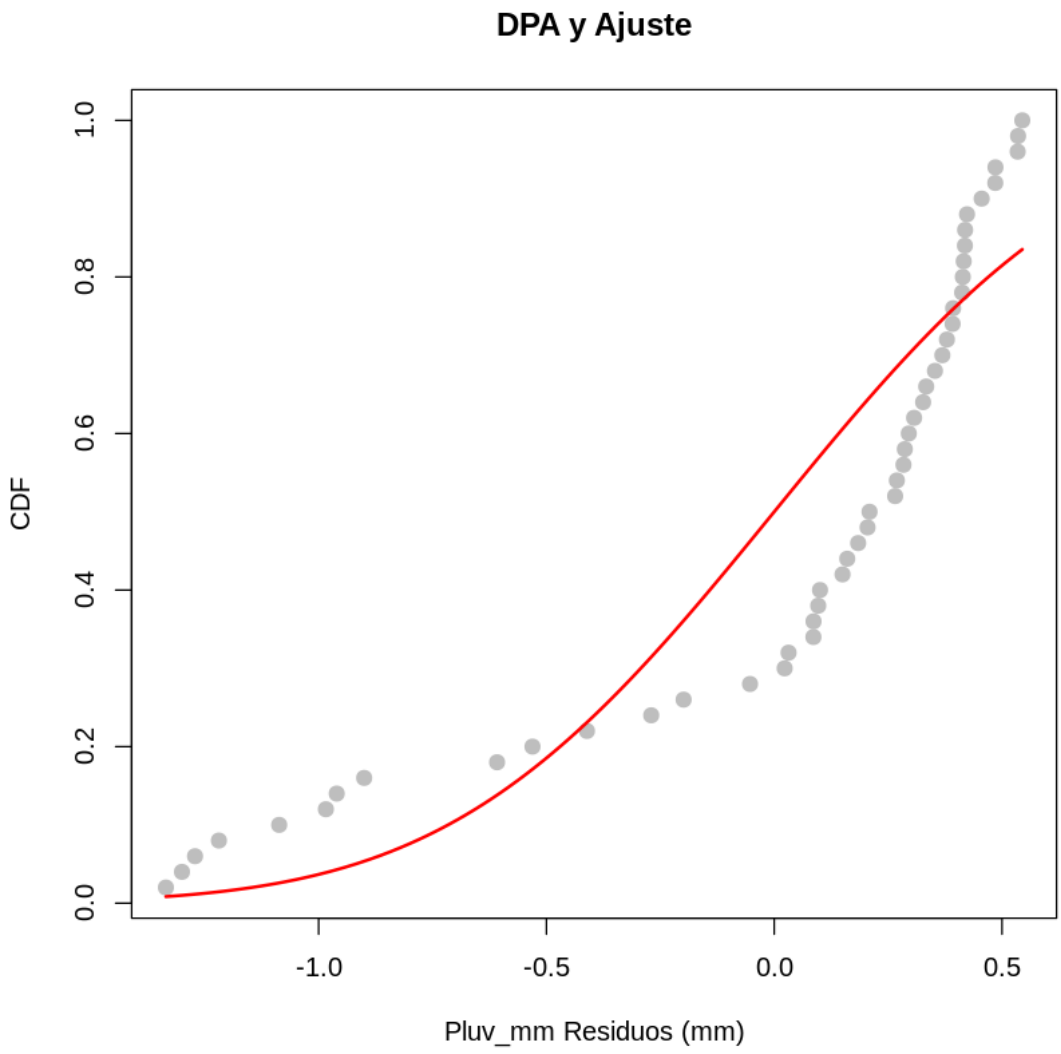
Graficamos las comparativas entre el histograma y la funcion de distribución normal.

[149]:
```
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 = "Pluv_mm Residuos␣
↪(mm)",
          ylab = "Densidad", colCurve =  "red", col = "darkgray")
```



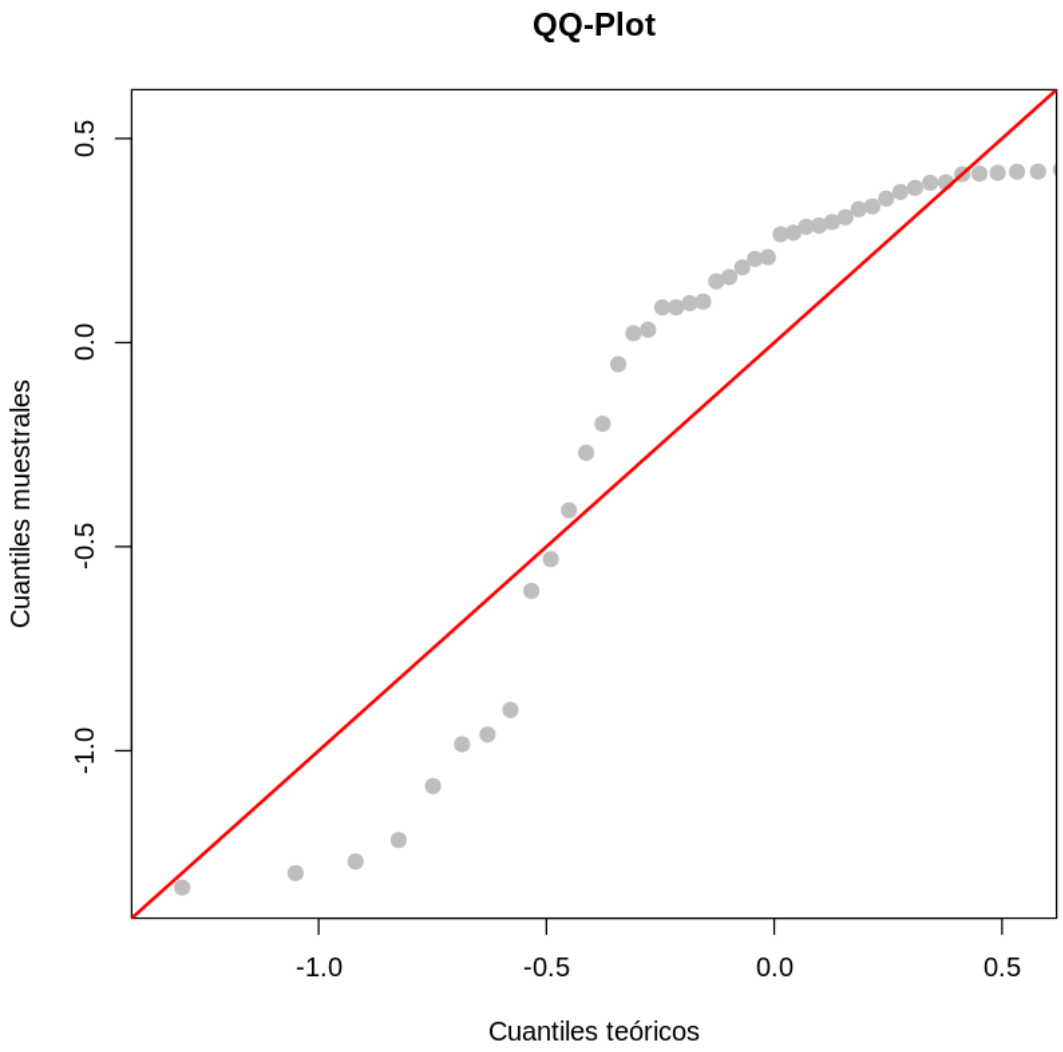
Graficamos las comparativas entre la función de distribución acumulativa empírica y la función de distribución acumulativa normal.

[150]:
```
CDF(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col = "gray",␣
↪main = "DPA y Ajuste",
    xlab = "Pluv_mm Residuos (mm)", lcol = "red", lwd = 2)
```



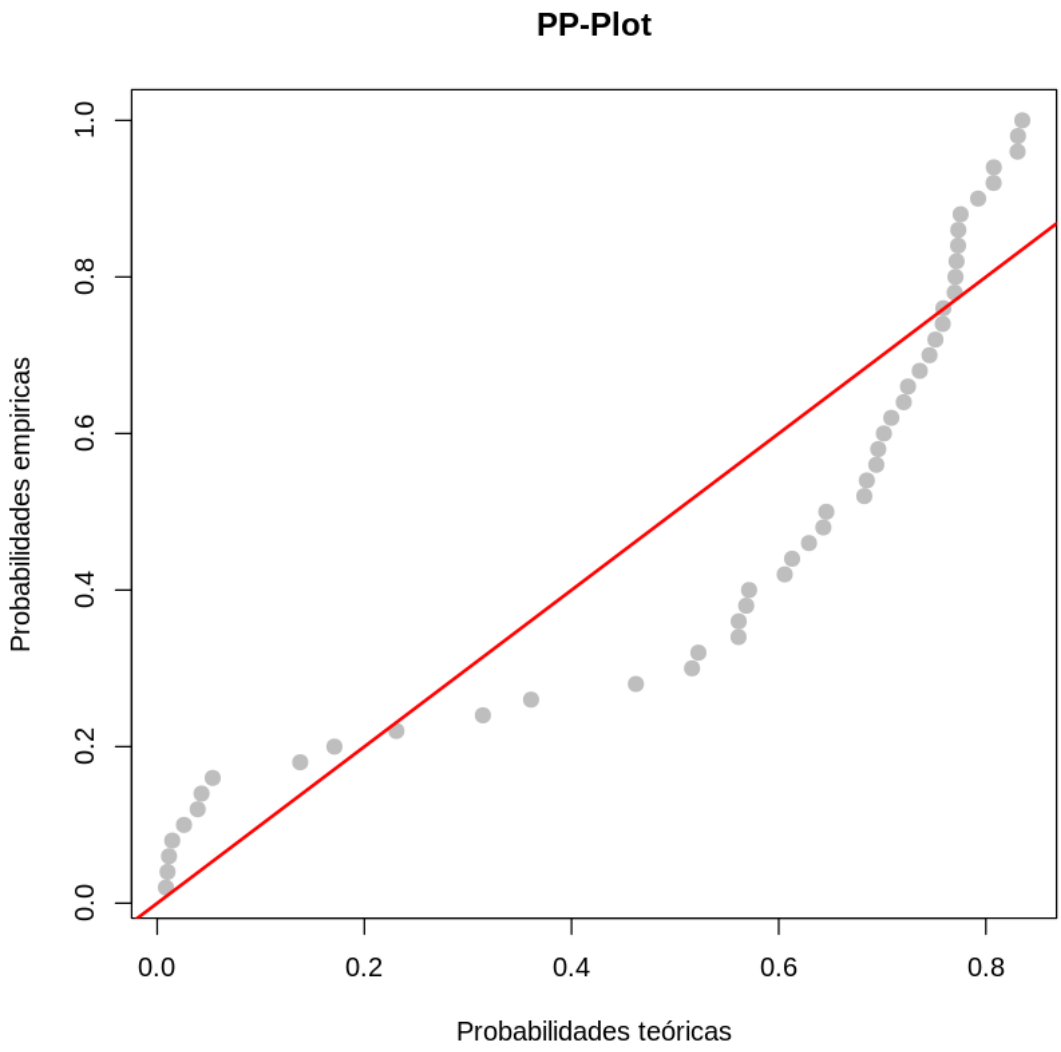
El Grafico cuantil-cuantil (Q-Q plot).

[151]:
```
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).

[152]:
```
PPplot(x = Y_Residual, distr = "norm", para = PARA_Residual_normal, col =␣
↪"gray", main = "PP-Plot",
      xlab = "Probabilidades teóricas", lcol = "red", lwd = 2)
```

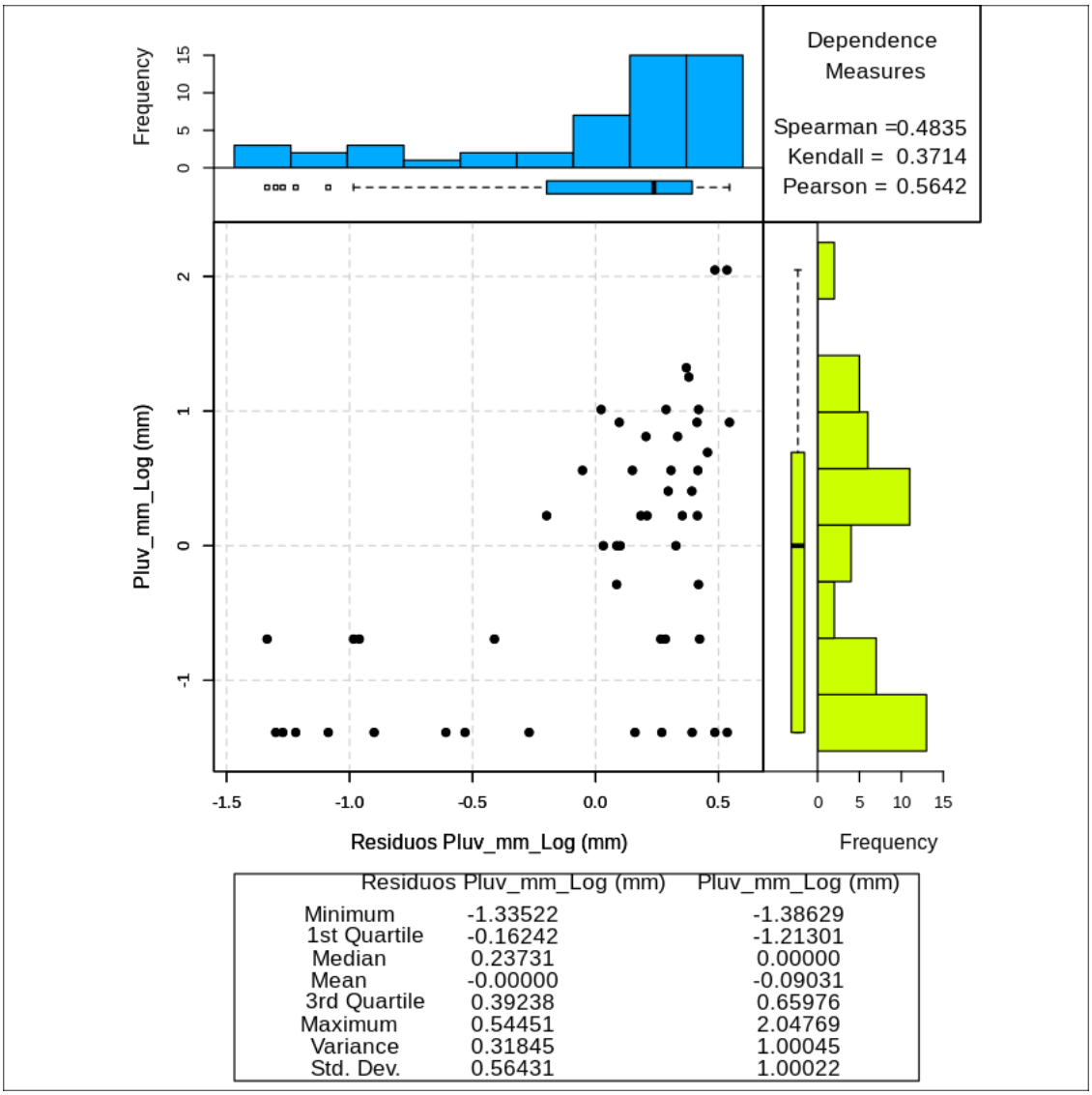


3.3.3 Análisis bivariado: Y vs Y residual.

[153]:
```
X<-Y_Residual      # Y_Residual es la variable independiente
Y<-Pluv_mm_Log     # Pluv_mm_Log es la variable dependiente
```

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

[154]:
```
ScatterPlot(X, Y, 9,
            Xmin = Y_Residual_Stat[2,2], Xmax = Y_Residual_Stat[7,2],
            Ymin = Pluv_mm_Log_Stat[2,2],Ymax = Pluv_mm_Log_Stat[7,2],
            XLAB = "Residuos Pluv_mm_Log (mm)", YLAB = "Pluv_mm_Log (mm)")
```



El resultado de las medidas de dependencia muestra que la dependencia es considerable. Pearson tiene un valor de 0.5702, Spearman es de 0.4971 y Kendall es de 0.3818. Con esta última prueba podemos concluir que la regresión lineal usando las variables aleatorias transformadas no cumplió con ninguna de las condiciones.

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

4 Análisis variográfico.

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

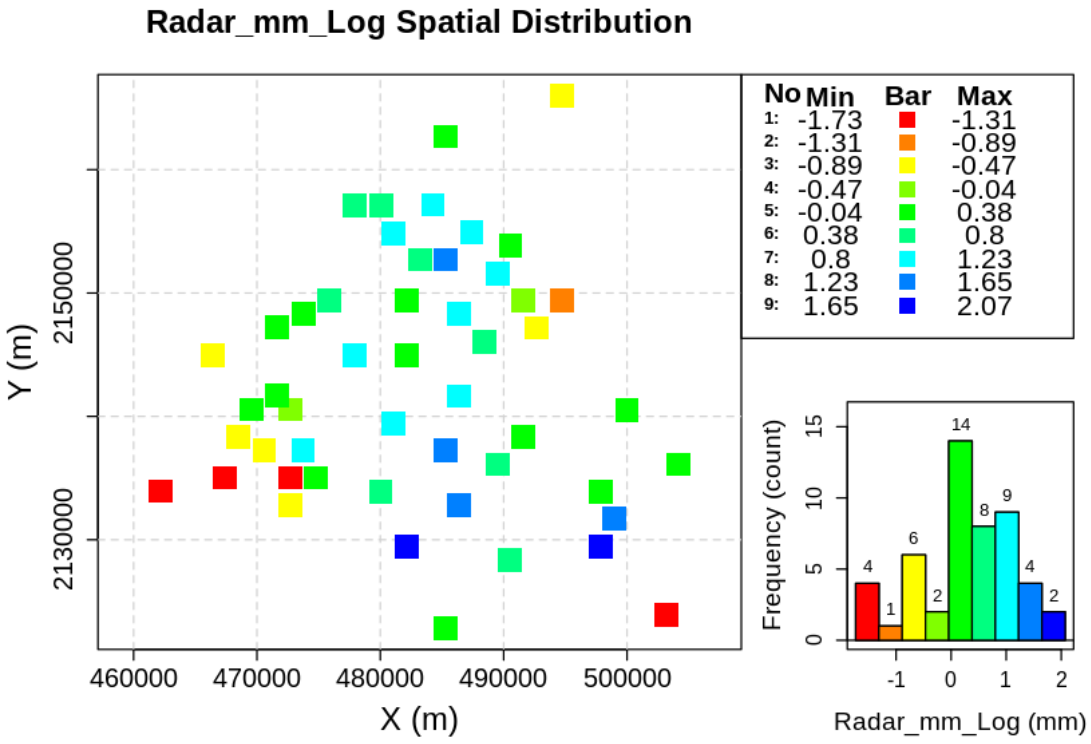
4.1 Análisis variográfico variable Radar_mm_Log

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 los pluviómetros con transformada logarítmica (Pluv_mm_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:

[155]:
```
DEspacial(XCoord, YCoord, Radar_mm_Log,n_bins=9,
          'X (m)', 'Y (m)', 'Radar_mm_Log (mm)', 'Radar_mm_Log Spatial␣
↪Distribution')
```



El gráfico que obtenemos nos da la distribución espacial de las muestras con puntos de distintos colores, siendo los colores rojo y azul los valores extremos. A la derecha del grafico tenemos el histograma que por default tiene nueve intervalos (bins).

4.1.1 Análisis de tendencia de la variable Radar_mm_Log.

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:

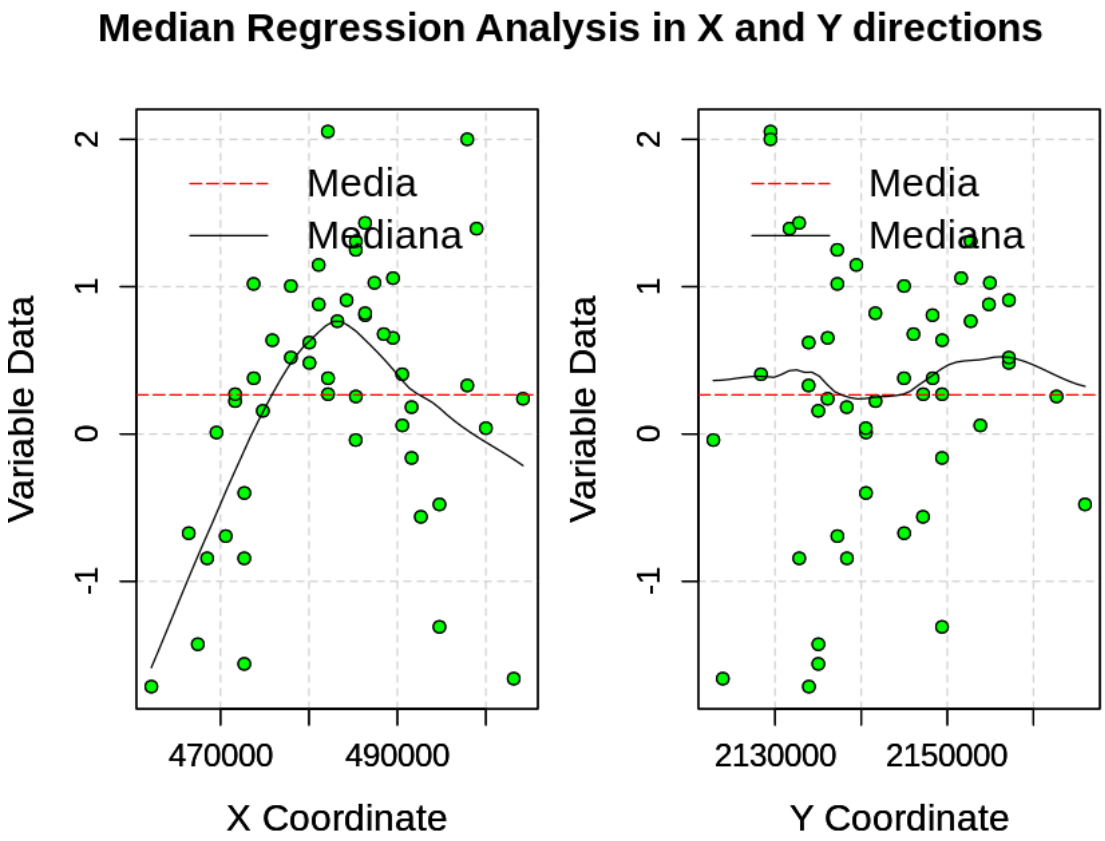
- Si la regresión es paralela a la línea de la media entonces la variable no tiene tendencia
- Si la regresión no es paralela entonces la variable podría tener algún grado de tendencia.

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 Radar_mm_Log.

[156]:
```
GDirecciones(XCoord, YCoord, Radar_mm_Log)
```



Analizando el resultado de la regresión podemos notar que en el eje de coordenadas X tiene una curva ascendente, llega a su punto más alto a la mitad del gráfico y después desciende, lo cual podemos considerar como un indicio de tendencia. Con el caso del eje de coordenadas Y vemos que su regresión es cercana a la línea del valor esperado lo cual nos indica que su nivel de tendencia es muy bajo. Esto lo confirmaremos cuando se estime los variogramas direccionales.

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.

[157]:
```
N_lags<-10
N_lags
```

10

Calculamos la distancia mínima entre muestras usando la función "dist" y "min", el argumento "Data_File[,1:2]" indica que se está usando las coordenadas. La distancia mínima de este ejemplo es de 1525.79.

[158]:
```
DistMin<-min(dist(Data_File[,1:2])) # Minimum distance in data
DistMin
```

1525.79979027394

La distancia máxima entre muestras se calcula usando la función "dist" y "max", en este caso es de 45696.15

[159]:
```
DistMax<-max(dist(Data_File[,1:2])) # Maximum distance in data
DistMax
```

45696.1566217554

El valor de intervalo se calcula dividiendo entre dos el valor de la distancia máxima y después dividir el valor resultante entre el número de intervalos. En este caso el valor de intervalo es de 2284.8

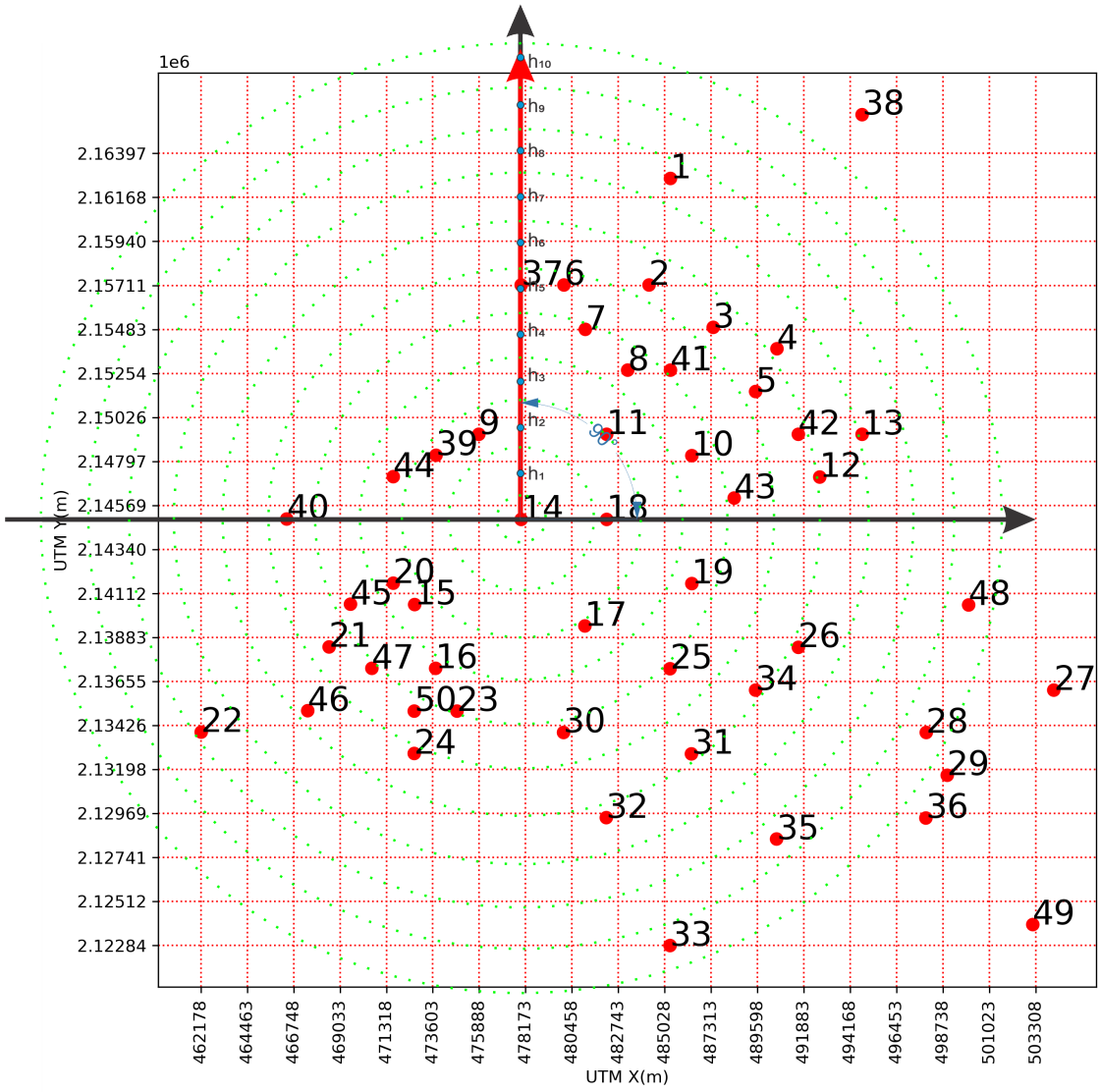
[160]:
```
lag_value<- (DistMax/2)/N_lags # DistMin
lag_value
```

2284.80783108777

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

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

Ya que tenemos estos valores podemos estimar el variograma experimental adireccional, el cual tiene por dirección 0^o y ángulo de tolerancia de 90^o. 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 (Pluv_mm_Log)
- Una dirección (Direccion) y su tolerancia angular (Tol), como en este caso es variograma adireccional la dirección es de 0^{o} y su tolerancia es de 90^{o}.
- 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 clásico o método de momentos, el cual es:

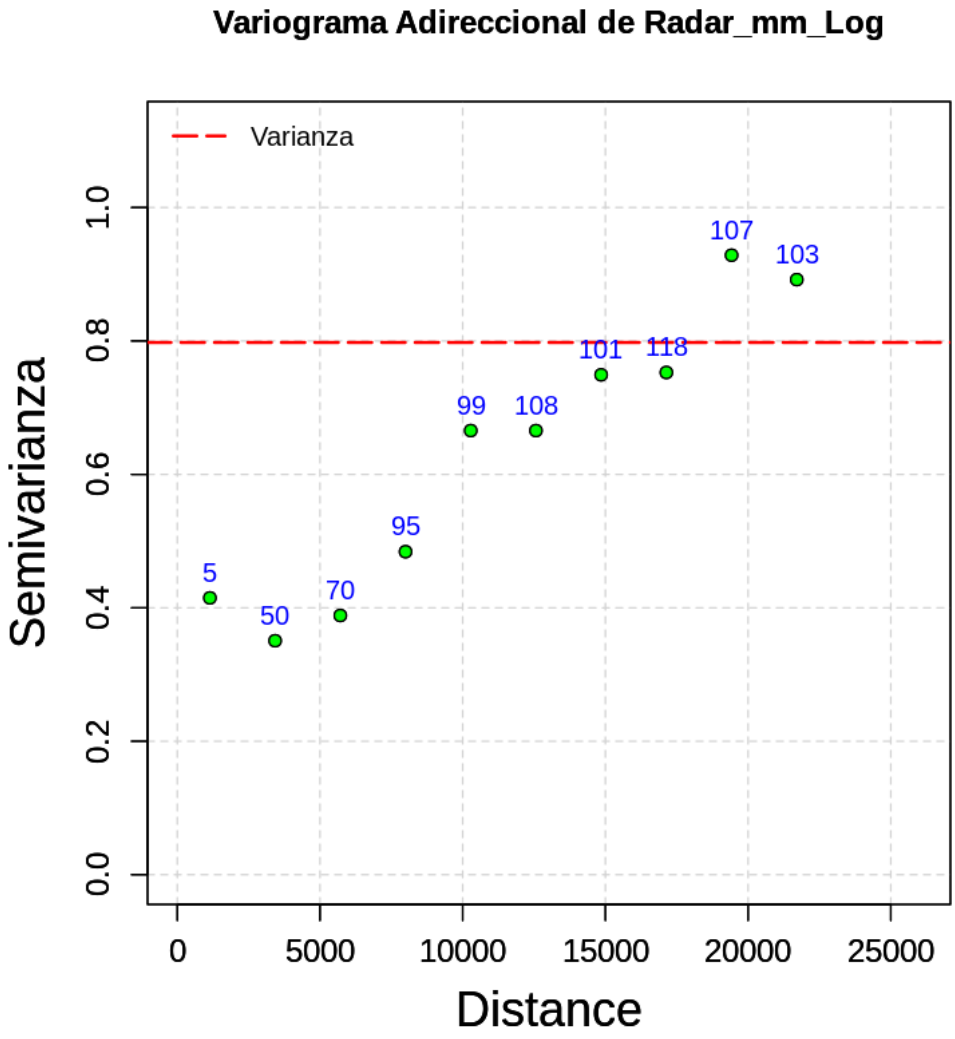
$$\gamma^{*}(\underline{h}) = \frac{1}{2N(\underline{h})} \sum_{i=1}^{N(\underline{h})} [Z(\underline{x}_i + \underline{h}) - Z(\underline{x}_i)]^2$$

[161]:
```
Radar_mm_Log_VarioEstimation<-Variograma(XCoord, YCoord, Radar_mm_Log,␣
↪Direccion=0, Tol=90,
                                         NIntervalos=N_lags, Lags=lag_value,␣
↪Npares=1,
                                         "Variograma Adireccional de␣
↪Radar_mm_Log", xlab="Distance")
Radar_mm_Log_VarioEstimation
```

```
variog: computing omnidirectional variogram
```

A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
5	1142.404	0.4148882
50	3427.212	0.3506868
70	5712.020	0.3884817
95	7996.827	0.4841424
99	10281.635	0.6656620
108	12566.443	0.6656483
101	14851.251	0.7492239
118	17136.059	0.7527673
107	19420.867	0.9285426
103	21705.674	0.8919122



Observando el resultado del variograma experimental adireccional se observa 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:

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

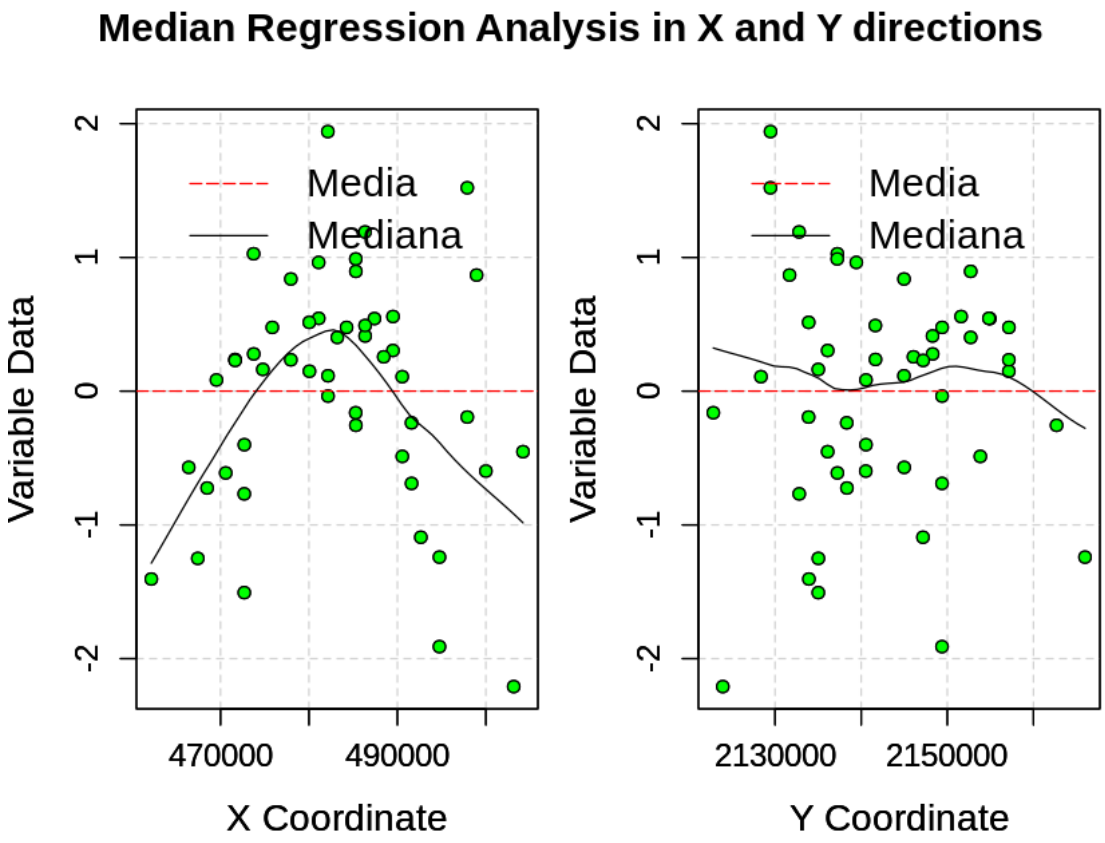
La transformación polinomial se puede hacer usando la función "Trend", la cual necesita los siguientes parámetros:

- Vector de las coordenadas (XCoord, YCoord)
- La variable (Pluv_mm_Log)
- El grado del polinomio (pol_degree).

[162]:
```
pol_degree=1
Radar_mm_Log_Detrended_1<-Trend(XCoord, YCoord,
                                Radar_mm_Log, pol_degree)
```

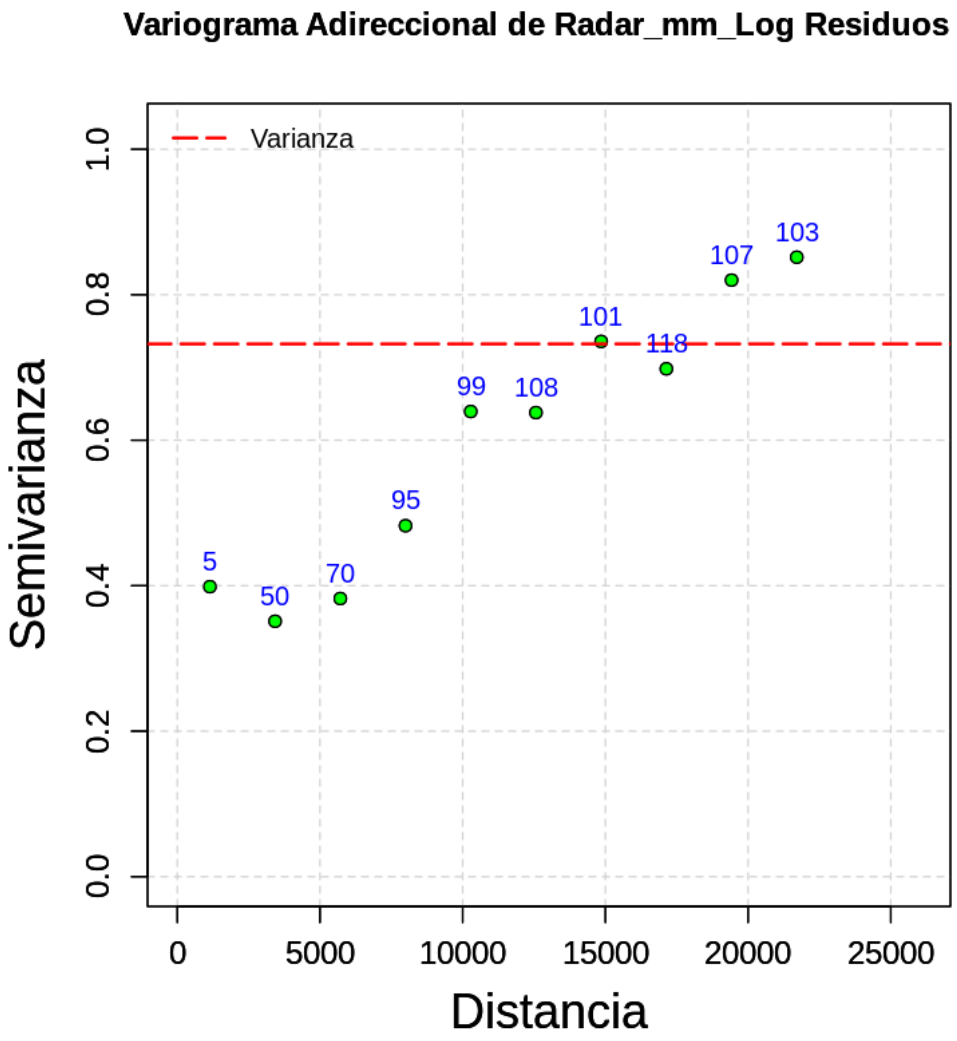
Con esta transformación volvemos a graficar la regresión de la mediana y el variograma experimental adireccional.

[163]:
```
GDirecciones(Radar_mm_Log_Detrended_1[,1], Radar_mm_Log_Detrended_1[,2],␣
↪Radar_mm_Log_Detrended_1[,3])
```



[164]:
```
Radar_mm_Log_Detrended_1_VarioEstimation<-Variograma(Radar_mm_Log_Detrended_1[,1],␣
↪Radar_mm_Log_Detrended_1[,2],
                                                    ␣
↪Radar_mm_Log_Detrended_1[,3], 0, 90, N_lags, lag_value, 1,
                                                      "Variograma Adireccional de␣
↪Radar_mm_Log Residuos 1")
```

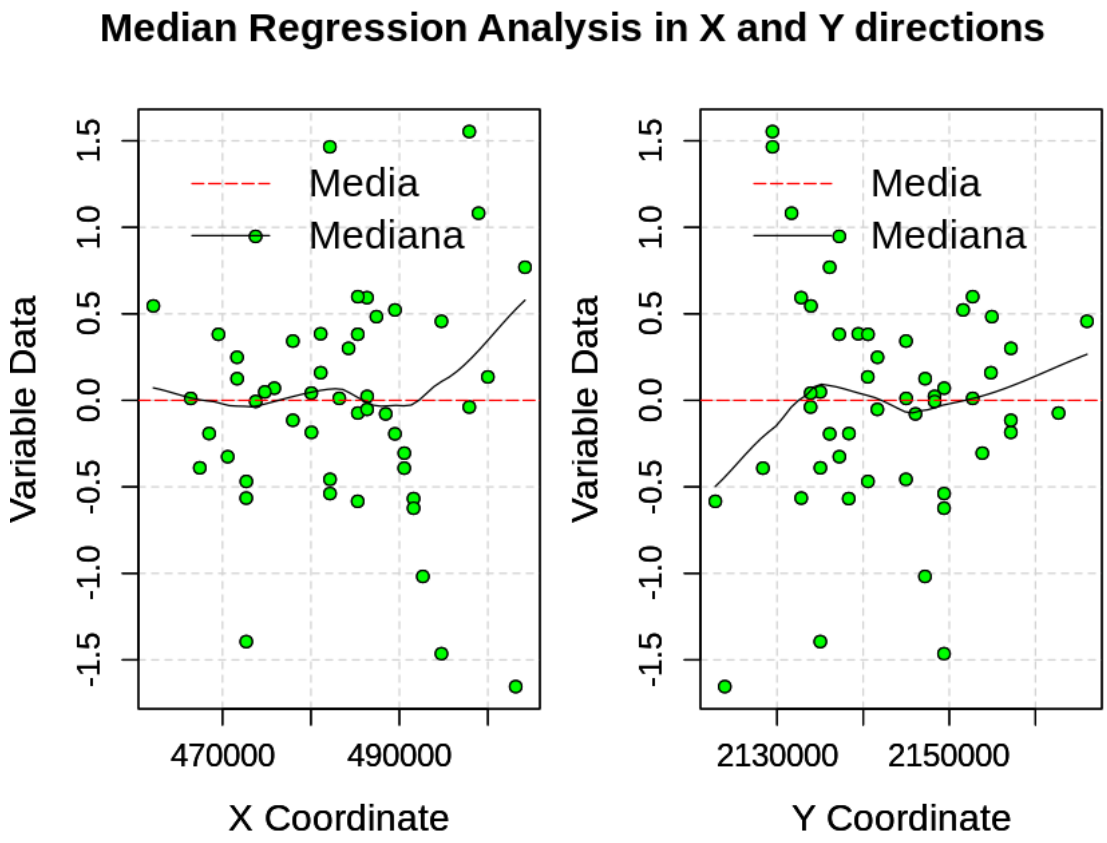
```
variog: computing omnidirectional variogram
```



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

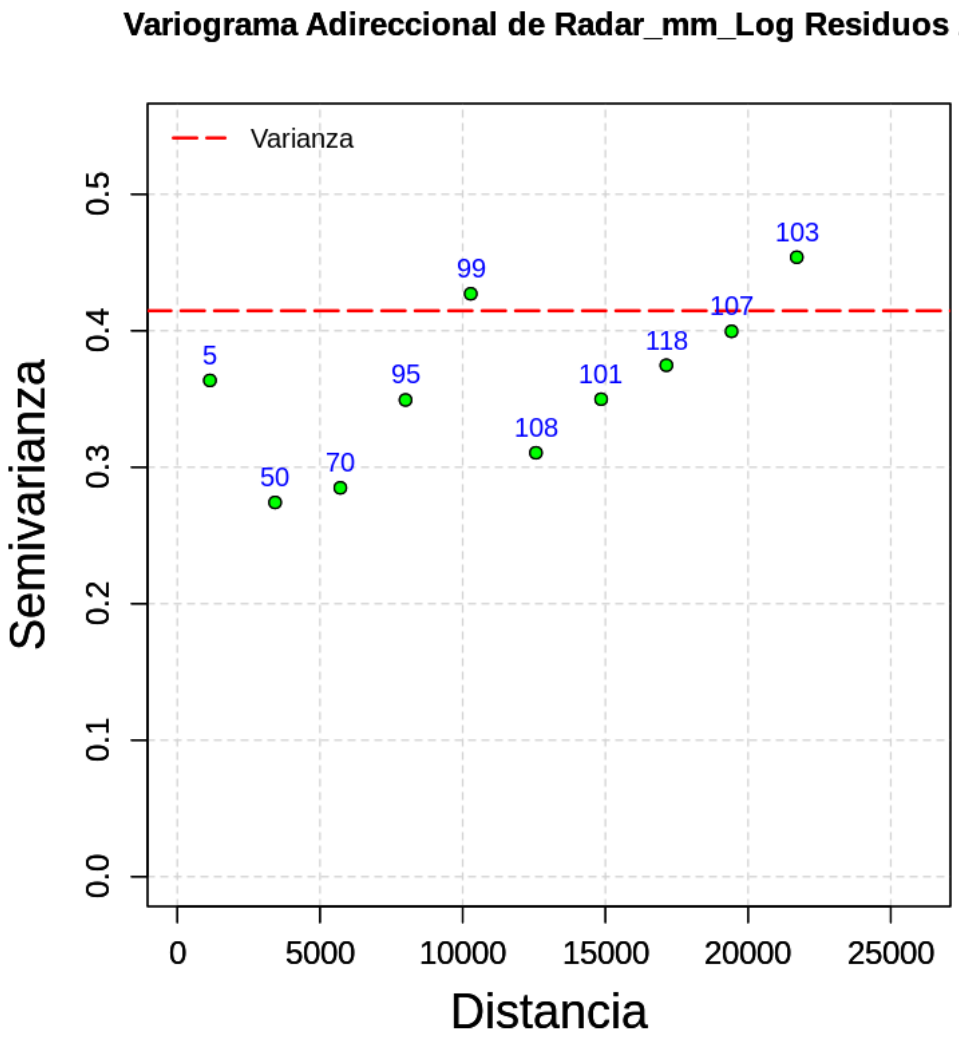
[165]:
```
pol_degree=2
Radar_mm_Log_Detrended_2<-Trend(XCoord, YCoord,
                                Radar_mm_Log, pol_degree)
```

[166]:
```
GDirecciones(Radar_mm_Log_Detrended_2[,1], Radar_mm_Log_Detrended_2[,2],␣
↪Radar_mm_Log_Detrended_2[,3])
```



[167]:
```
Radar_mm_Log_Detrended_2_VarioEstimation<-Variograma(Radar_mm_Log_Detrended_2[,1],␣
↪Radar_mm_Log_Detrended_2[,2],
                                                    ␣
↪Radar_mm_Log_Detrended_2[,3], 0, 90, N_lags, lag_value, 1,
                                                      "Variograma Adireccional de␣
↪Radar_mm_Log Residuos 2")
```

```
variog: computing omnidirectional variogram
```



4.1.2 Modelado variográfico unidimensional de la variable Radar_mm_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.

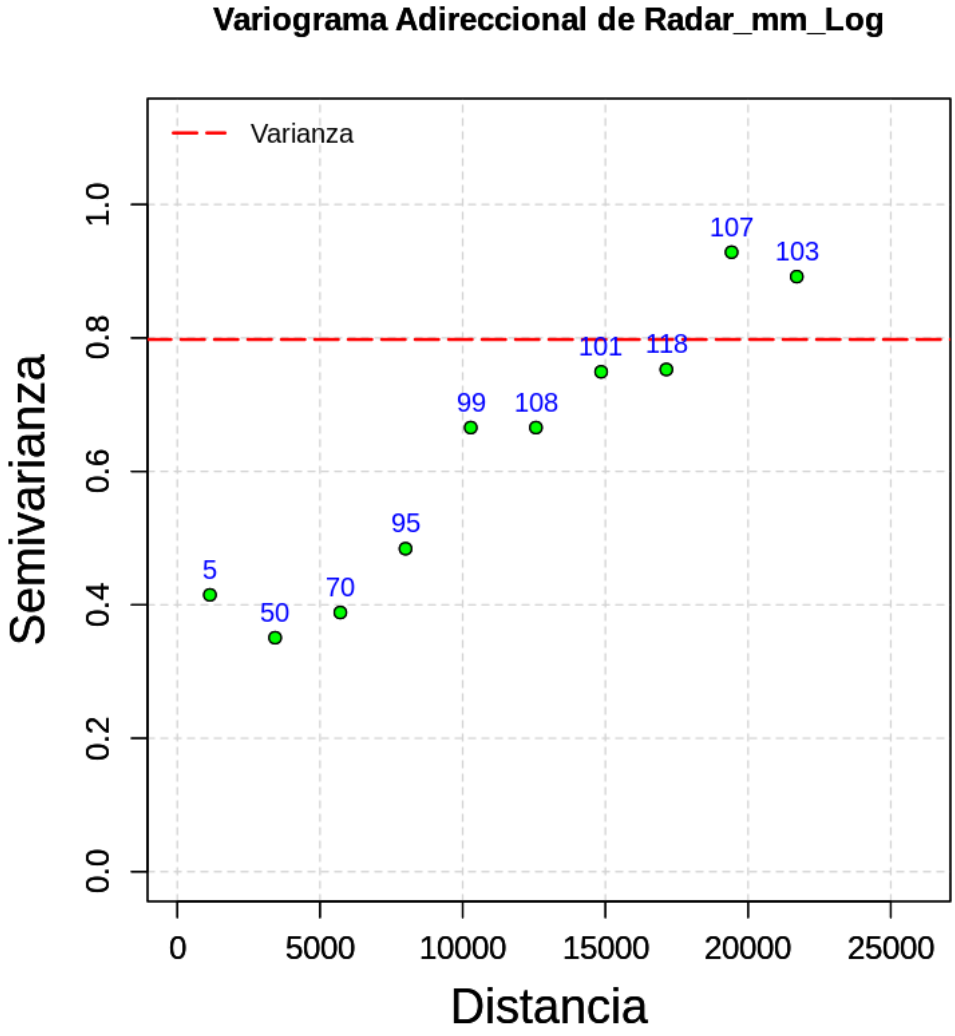
El variograma adireccional se calcula de la siguiente forma:

[168]:
```
Radar_mm_Log_VarioEstimation<-Variograma(XCoord, YCoord,
                                         Radar_mm_Log, 0, 90, N_lags, lag_value,␣
↪1,
                                         "Variograma Adireccional de␣
↪Radar_mm_Log")
Radar_mm_Log_VarioEstimation
```

```
variog: computing omnidirectional variogram
```

A data.frame: 10 × 3

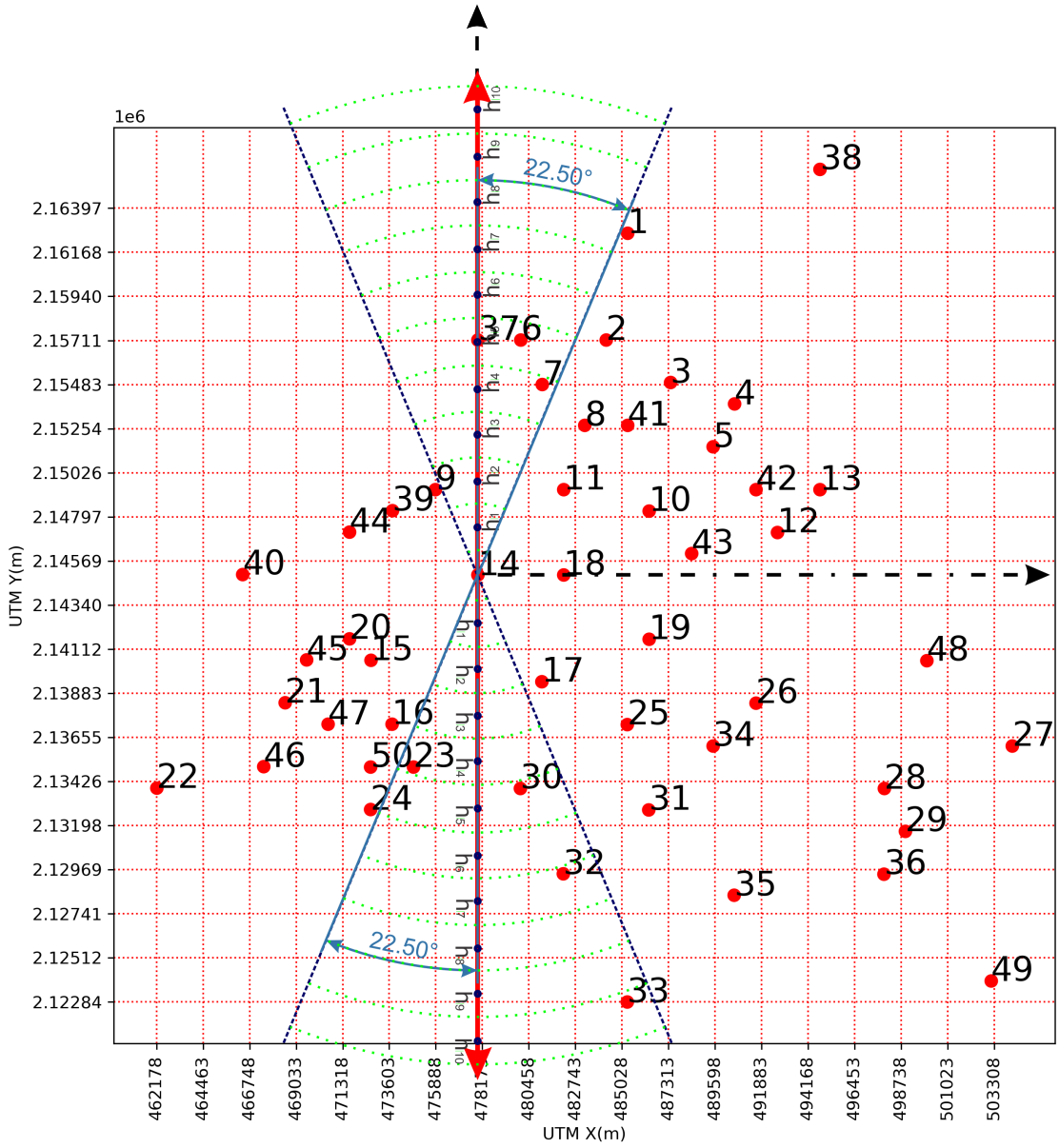
Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
5	1142.404	0.4148882
50	3427.212	0.3506868
70	5712.020	0.3884817
95	7996.827	0.4841424
99	10281.635	0.6656620
108	12566.443	0.6656483
101	14851.251	0.7492239
118	17136.059	0.7527673
107	19420.867	0.9285426
103	21705.674	0.8919122



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

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

Dado estos parámetros, se puede considerar las siguientes graficas que ilustran la forma de cálculo del variograma con los parámetros antes mencionados, para la dirección de vector 0^o es:



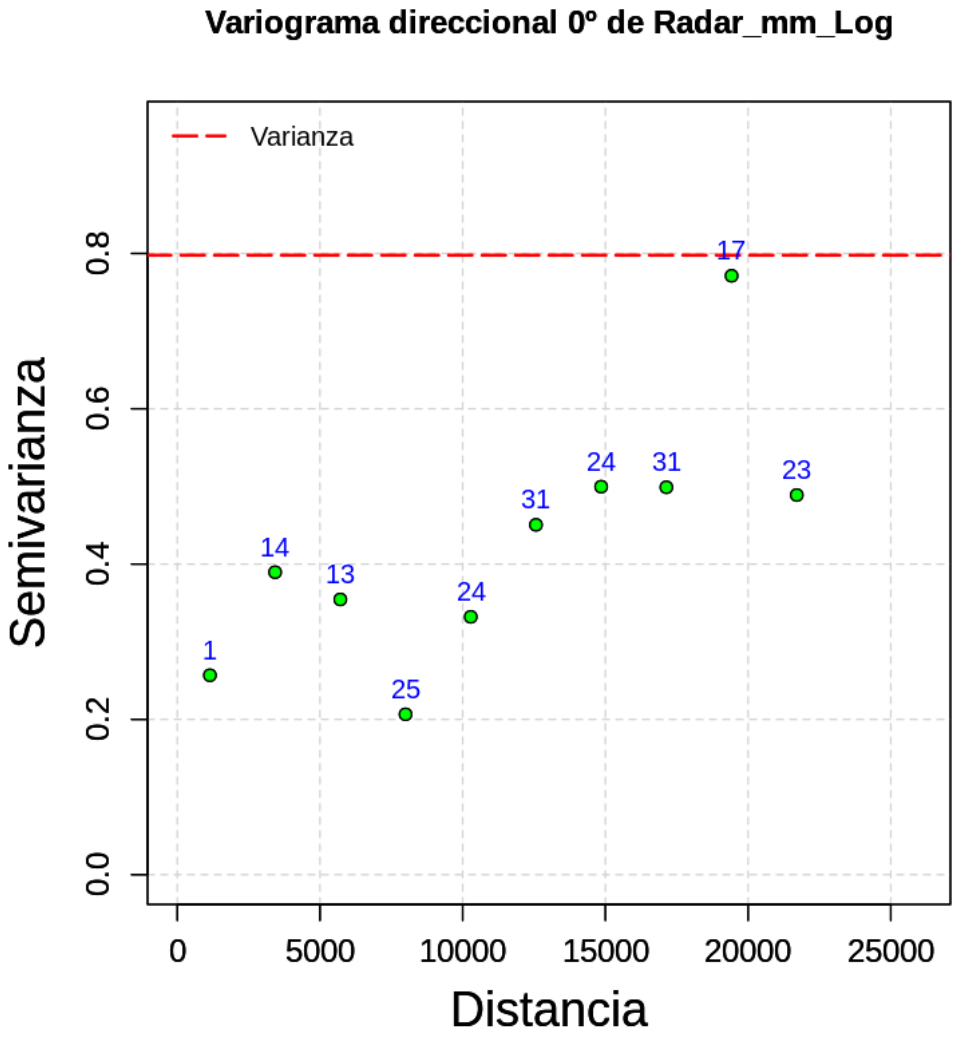
Nota: Esta imagen solo muestra el computo con respecto a un punto, el cálculo completo tiene que ser cubierto con todos los puntos

[169]:
```
Radar_mm_Log_VarioEstimation_0<-Variograma(XCoord, YCoord,
                                           Radar_mm_Log, 0, 22.5, N_lags,␣
↪lag_value, 1,
                                             "Variograma direccional 0º de␣
↪Radar_mm_Log")
Radar_mm_Log_VarioEstimation_0
```

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

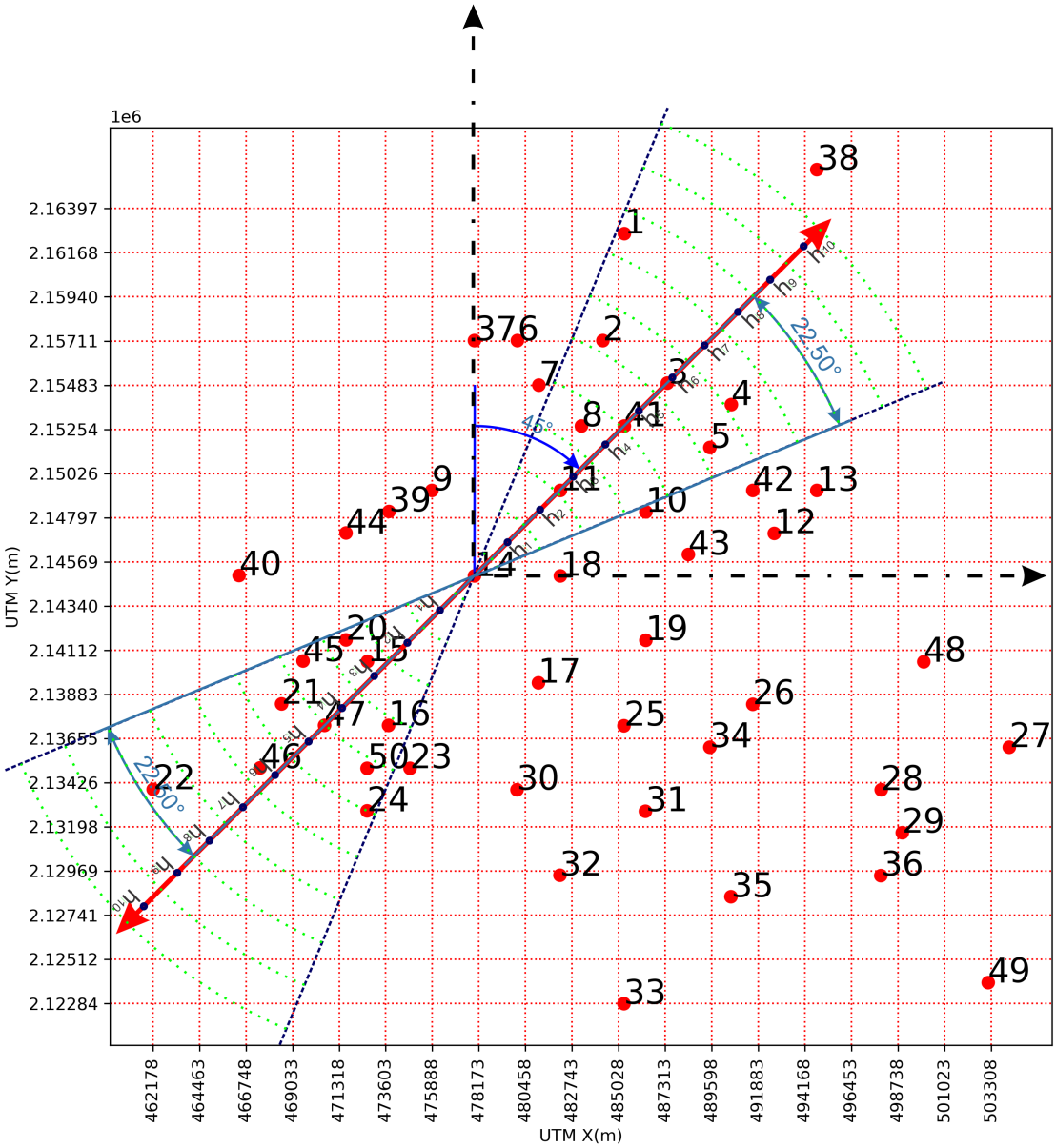
A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
1	1142.404	0.2568134
14	3427.212	0.3895325
13	5712.020	0.3544580
25	7996.827	0.2065631
24	10281.635	0.3320694
31	12566.443	0.4505446
24	14851.251	0.4996824
31	17136.059	0.4989373
17	19420.867	0.7712610
23	21705.674	0.4889112



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

Calculamos el variograma direccional 45^{o}

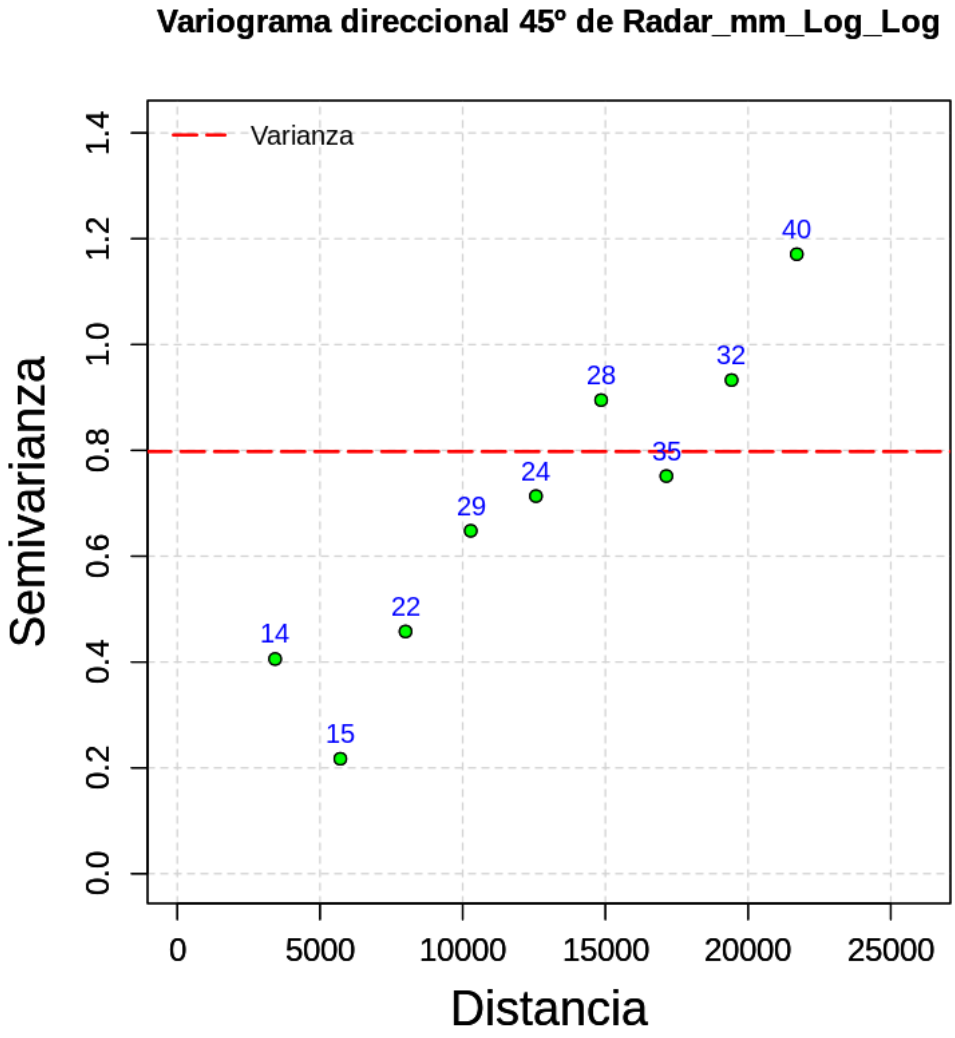


[170]:
```
Radar_mm_Log_VarioEstimation_45<-Variograma(XCoord, YCoord,
                                            Radar_mm_Log, 45, 22.5, N_lags,␣
↪lag_value, 1,
                                              "Variograma direccional 45º de␣
↪Radar_mm_Log")
Radar_mm_Log_VarioEstimation_45
```

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

A data.frame: 9 × 3

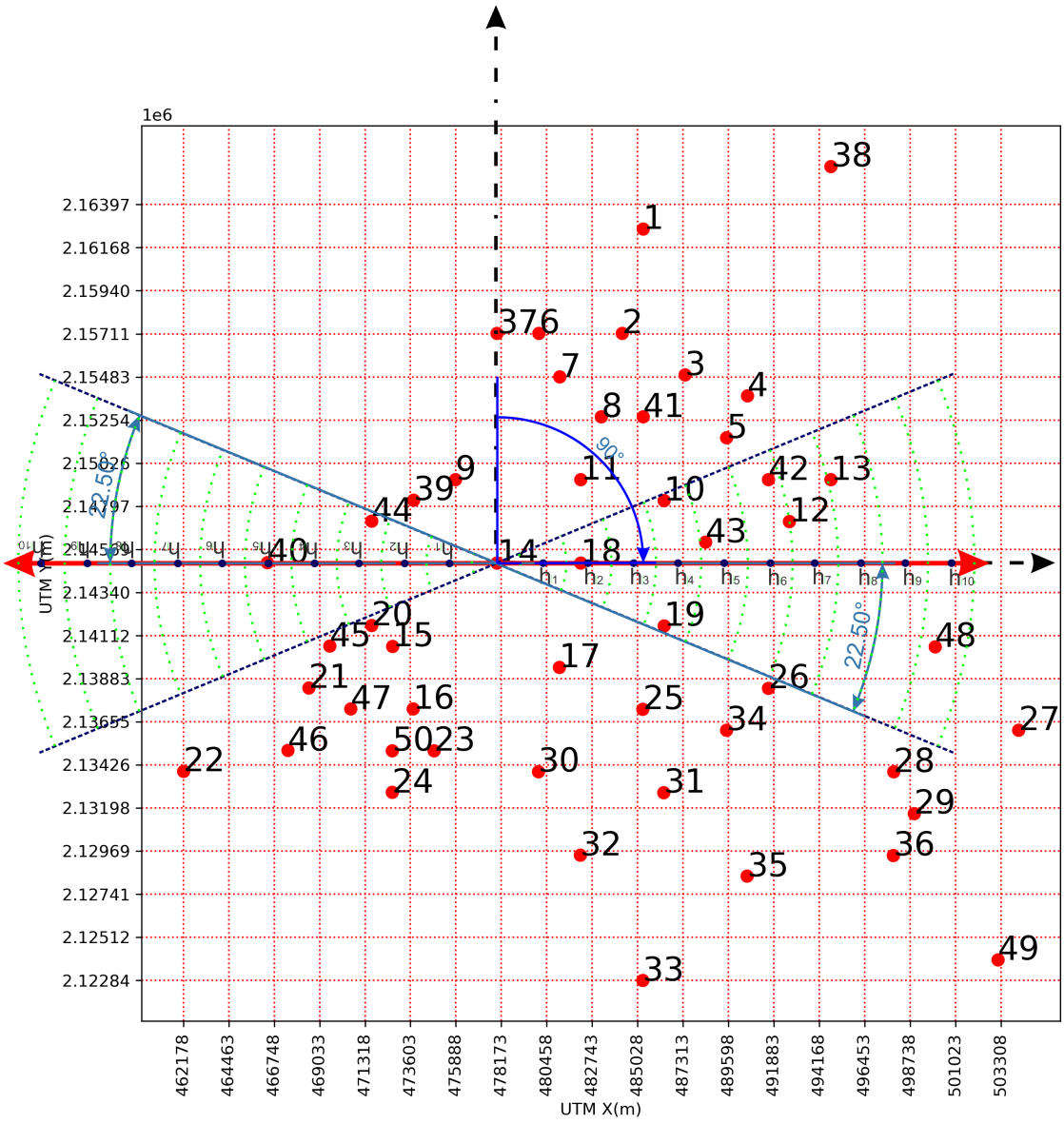
Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
14	3427.212	0.4056996
15	5712.020	0.2171602
22	7996.827	0.4577728
29	10281.635	0.6480578
24	12566.443	0.7136076
28	14851.251	0.8950589
35	17136.059	0.7515032
32	19420.867	0.9329158
40	21705.674	1.1707838



Con el variograma direccional a 45º notamos el mismo problema detectado en el variograma direccional de 0º, solo tres intervalos tienen un número de pares superior a 30, por lo tanto, en esta

dirección no se puede juzgar si existe algún tipo de anisotropía.

Probamos con el variograma direccional de 90º

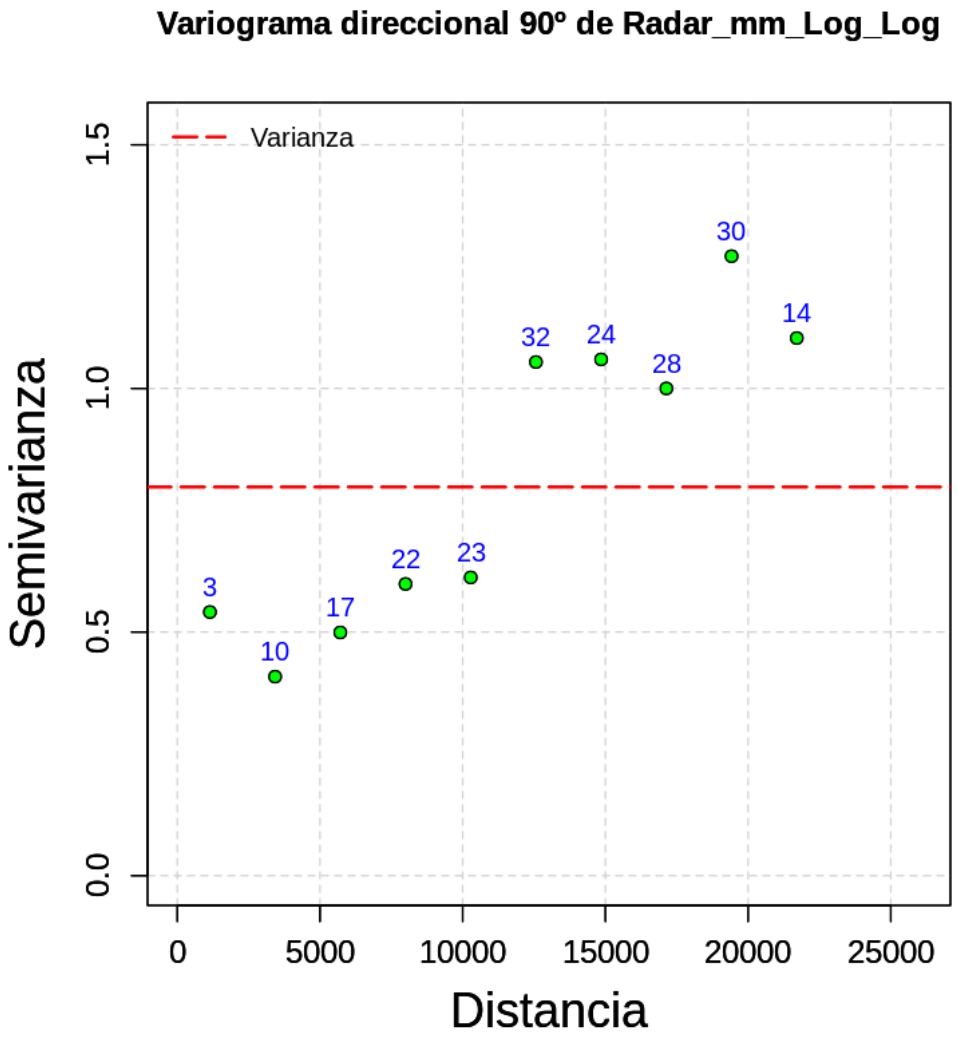


[171]:
```
Radar_mm_Log_VarioEstimation_90<-Variograma(XCoord, YCoord,
                                            Radar_mm_Log, 90, 22.5, N_lags,␣
↪lag_value, 1,
                                              "Variograma direccional 90º de␣
↪Radar_mm_Log")
Radar_mm_Log_VarioEstimation_90
```

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

A data.frame: 10 × 3

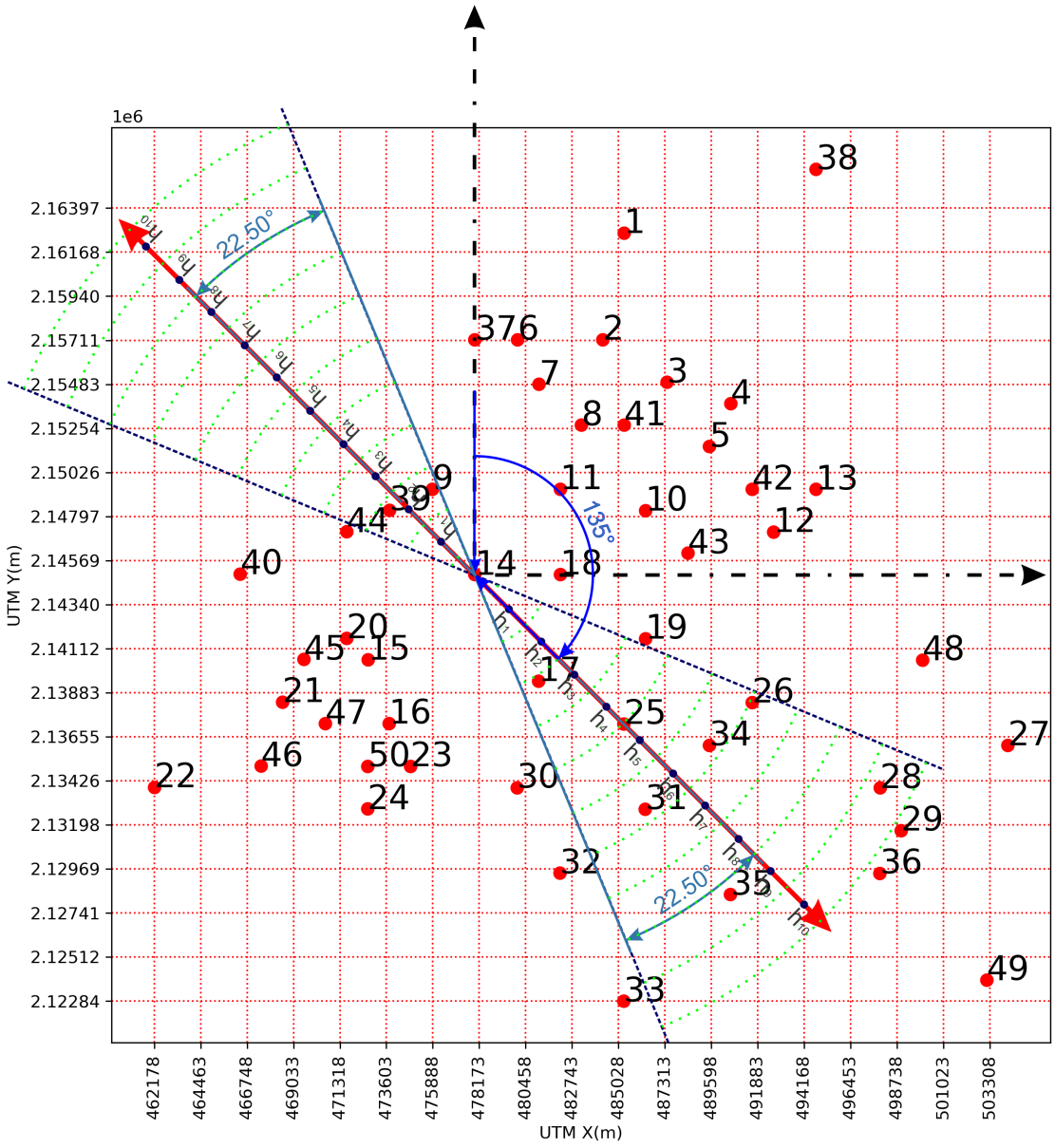
Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
3	1142.404	0.5410586
10	3427.212	0.4084904
17	5712.020	0.4991839
22	7996.827	0.5986924
23	10281.635	0.6120796
32	12566.443	1.0541704
24	14851.251	1.0595848
28	17136.059	1.0000044
30	19420.867	1.2713758
14	21705.674	1.1033700



Con el variograma direccional de 90º seguimos obteniendo intervalos mal estimados, solo dos tiene

más de 30 pares y por lo tanto no se puede juzgar si existe anisotropía en esta dirección.

Seguimos con el variograma direccional de 135º.

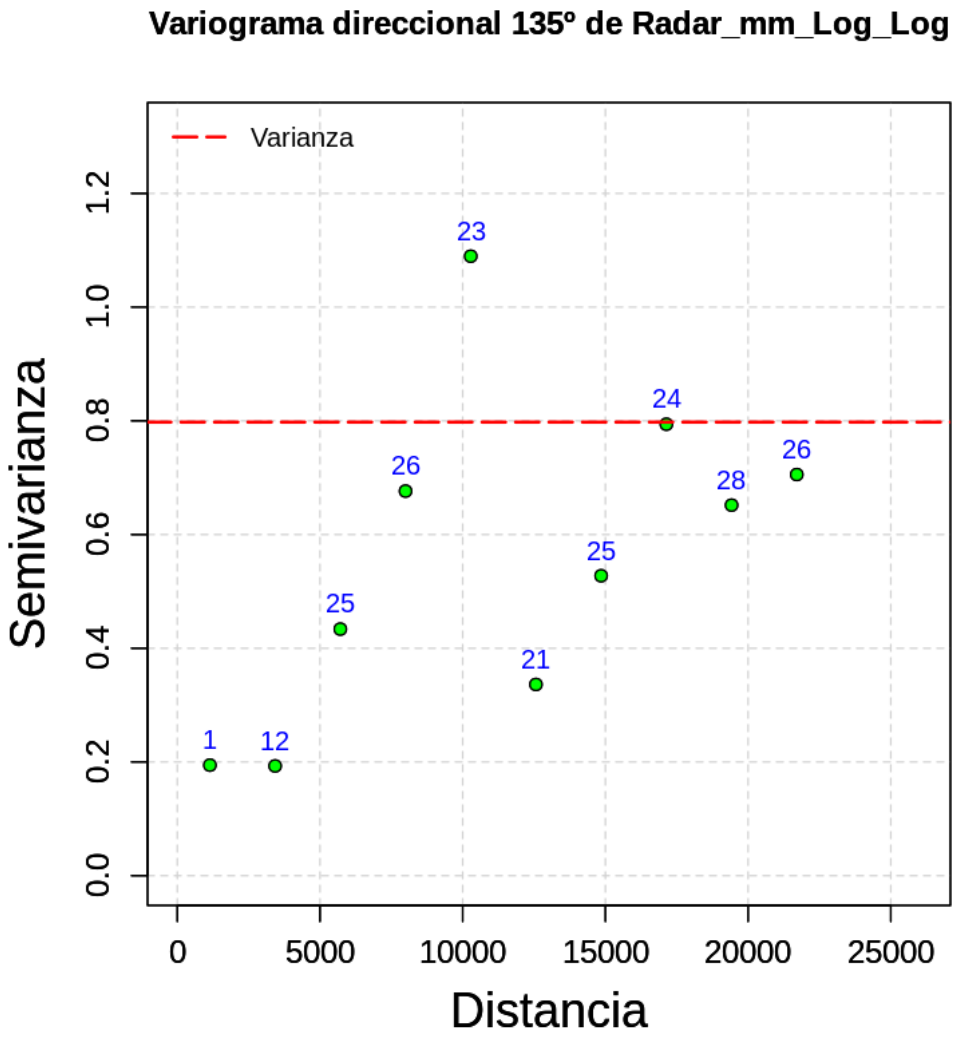


[172]:
```
Radar_mm_Log_VarioEstimation_135<-Variograma(XCoord, YCoord,
                                             Radar_mm_Log, 135, 22.5, N_lags,␣
↪lag_value, 1,
                                                 "Variograma direccional 135º de␣
↪Radar_mm_Log")
Radar_mm_Log_VarioEstimation_135
```

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

A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
1	1142.404	0.1944516
12	3427.212	0.1930156
25	5712.020	0.4336894
26	7996.827	0.6764315
23	10281.635	1.0895375
21	12566.443	0.3363382
25	14851.251	0.5275022
24	17136.059	0.7940313
28	19420.867	0.6517157
26	21705.674	0.7055181



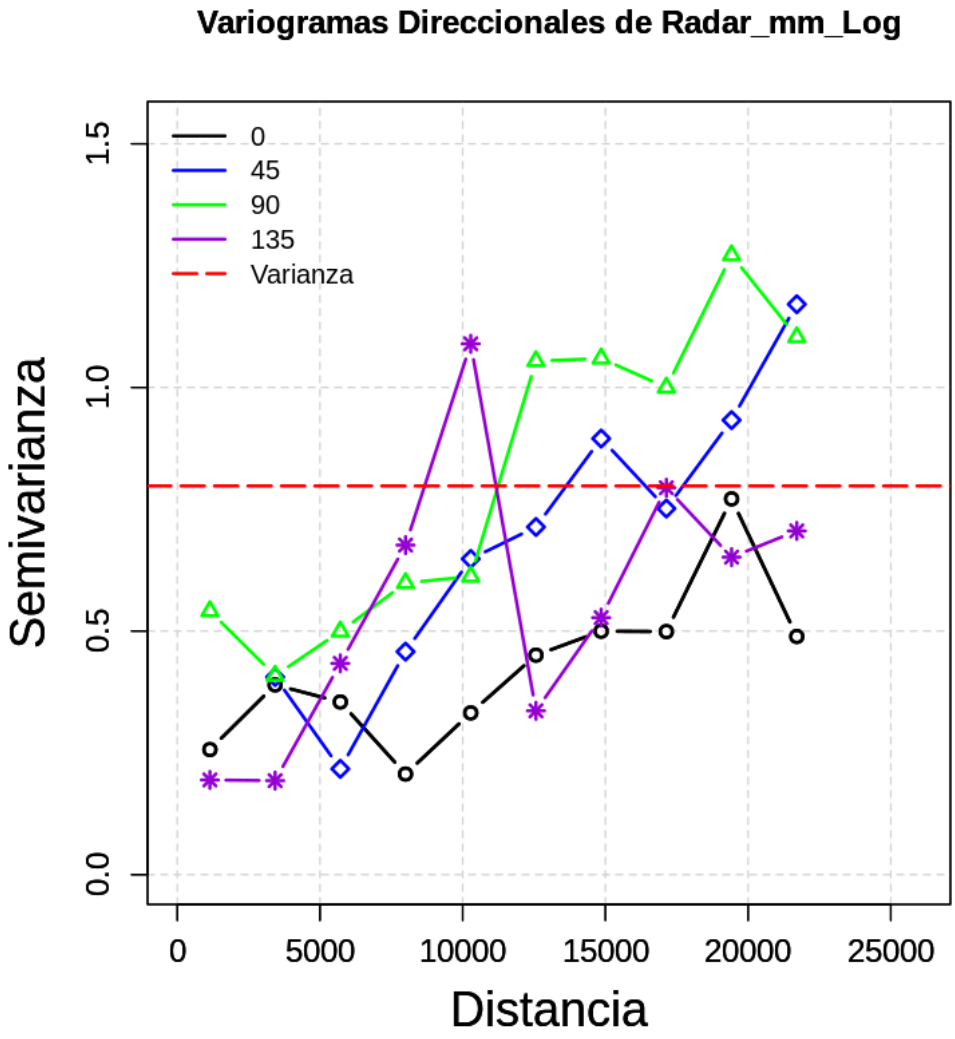
El variograma direccional a 135º no tiene intervalos con un número de pares superior a 30, de

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

Para visualizar los cuatro variogramas direccionales en una sola imagen usamos la función "Variograma4D". Esta función necesita * Vector de las coordenadas (XCoord, YCoord) * La variable (Pluv_mm_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

[173]:
```
Radar_mm_Log_VarioEstimation4D<-Variograma4D(XCoord, YCoord,
                                             Radar_mm_Log, 0, 45, 90, 135, 22.5,␣
↪N_lags, lag_value, 1,
                                                 "Variogramas Direccionales de␣
↪Radar_mm_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)
```



Si colocan en la consola "Radar_mm_Log_VarioEstimation4D", pueden obtener los pares de la estimación en los cuatro variogramas direccionales.

[174]:
```
Radar_mm_Log_VarioEstimation4D
```

$Zero A matrix: 10 × 3 of type dbl

1	1142.404	0.2568134
14	3427.212	0.3895325
13	5712.020	0.3544580
25	7996.827	0.2065631
24	10281.635	0.3320694
31	12566.443	0.4505446
24	14851.251	0.4996824
31	17136.059	0.4989373
17	19420.867	0.7712610
23	21705.674	0.4889112

$FortyFive A matrix: 9 × 3 of type dbl

14	3427.212	0.4056996
15	5712.020	0.2171602
22	7996.827	0.4577728
29	10281.635	0.6480578
24	12566.443	0.7136076
28	14851.251	0.8950589
35	17136.059	0.7515032
32	19420.867	0.9329158
40	21705.674	1.1707838

$Ninety A matrix: 10 × 3 of type dbl

3	1142.404	0.5410586
10	3427.212	0.4084904
17	5712.020	0.4991839
22	7996.827	0.5986924
23	10281.635	0.6120796
32	12566.443	1.0541704
24	14851.251	1.0595848
28	17136.059	1.0000044
30	19420.867	1.2713758
14	21705.674	1.1033700

$OneThertyFive A matrix: 10 × 3 of type dbl

1	1142.404	0.1944516
12	3427.212	0.1930156
25	5712.020	0.4336894
26	7996.827	0.6764315
23	10281.635	1.0895375
21	12566.443	0.3363382
25	14851.251	0.5275022
24	17136.059	0.7940313
28	19420.867	0.6517157
26	21705.674	0.7055181

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.

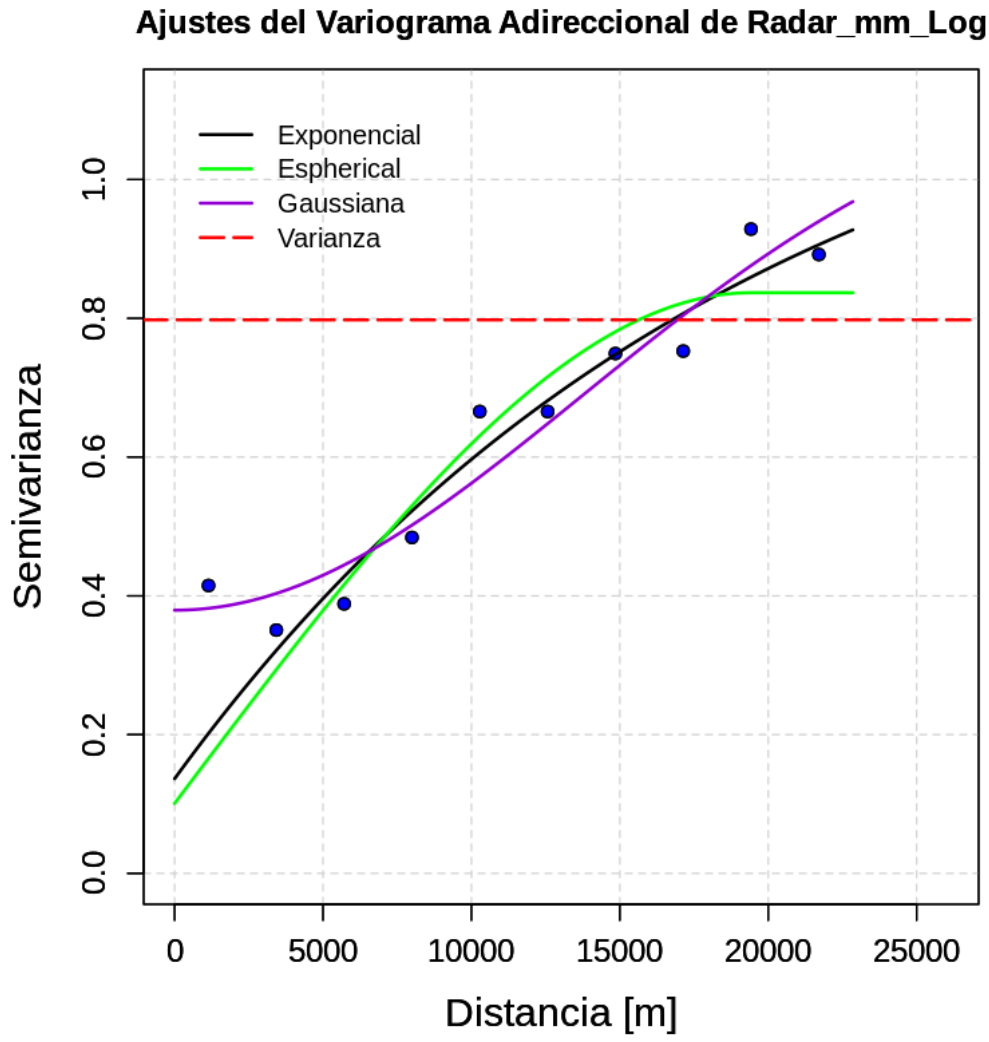
Ahora que sabemos cuál es el mejor variograma experimental, procedemos a ajustar un modelo de variograma autorizado. Para hacer el ajuste automático usamos la función "AllModel", esta función necesita los siguientes parámetros:

- Vector de las coordenadas (XCoord, YCoord)
- La variable (Pluv_mm_Log)
- La dirección del vector el cual es de 0º
- Su ángulo de tolerancia (90º)
- 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:

[175]:
```
Radar_mm_Log_AllModelVarioFit<-AllModel(XCoord, YCoord,
                                    Radar_mm_Log, 0, 90, N_lags, lag_value, 1,
                                        "Ajustes del Variograma Adireccional de␣
↪Radar_mm_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
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is exponential
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is spherical
variofit: weights used: npairs
variofit: minimisation function used: optim
variofit: covariance model used is gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
```



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

[176]:
```
Radar_mm_Log_AllModelVarioFit
```

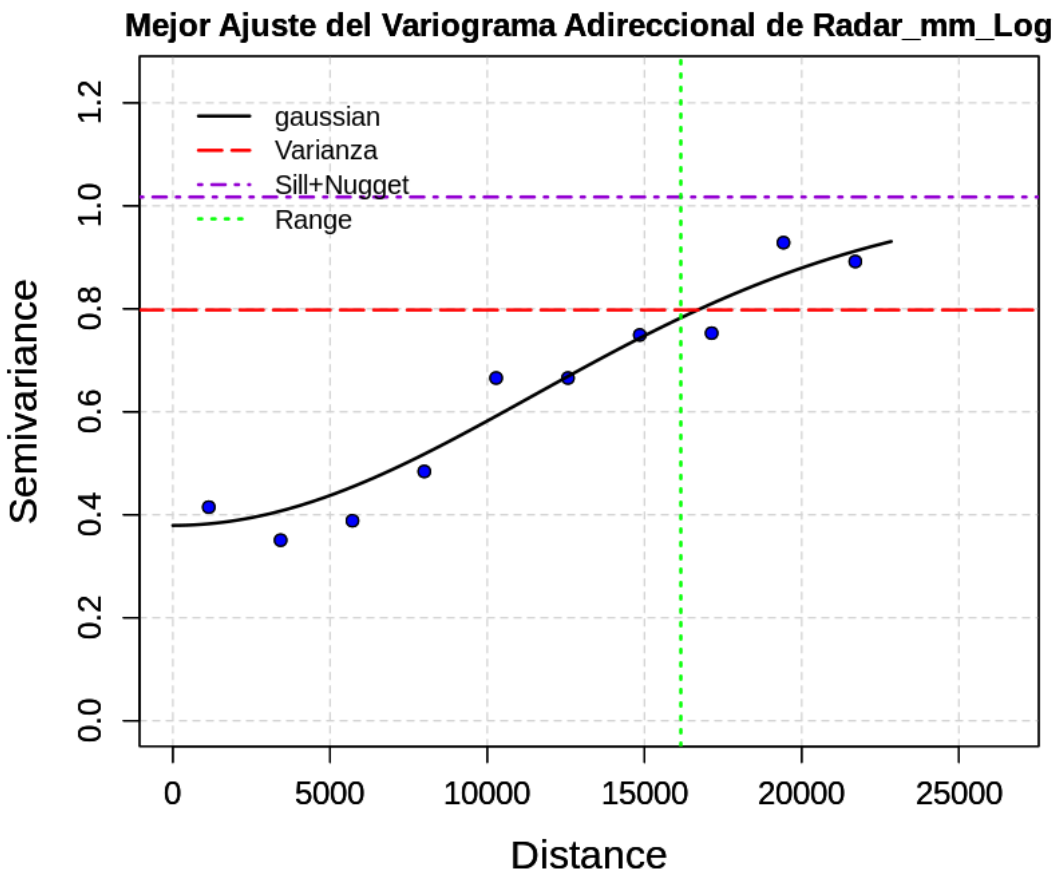
A matrix: 3 × 4 of type dbl

	Nugget	Meseta+Nugget	Alcance	SCE
exponential	0.1364152	1.2801046	19420	0.06095025
spherical	0.1007615	0.8369067	19420	0.08917023
gaussian	0.3793155	1.1649810	19420	0.04857662

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

[177]:
```
Radar_mm_Log_BestModelVarioFit<-BestModel(XCoord, YCoord,
                                          Radar_mm_Log, 0, 90, N_lags, lag_value, 1,␣
↪"Mejor Ajuste del Variograma Adireccional de Radar_mm_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
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 gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
```



Model	Nugget	Sill+Nugget	Range	MSE
gaussian	0.3793	1.0172	16156.7744	0.0228

[178]:
```
Radar_mm_Log_BestModelVarioFit
```

A matrix: 1 × 6 of type dbl

	Nugget	Meseta+Nugget	Alcance	SCE	MaxY	MinY
gaussian	0.3793155	1.017185	16156.77	0.02275588	0.9285426	0.35068

Como podemos notar, el mejor modelo según el ajuste automático es Gaussiano, sin embargo, debemos tomar en cuenta que el ajuste dio importancia a los pares del primer intervalo, dando como resultado un ajuste automático poco confiable, por lo tanto, se recomienda usar otro modelo y ajustarlo al variograma experimental.

Para hacer el ajuste manual usamos la función "EyeModel", esta función necesita los siguientes parámetros: * Vector de las coordenadas (XCoord, YCoord) * La variable (Pluv_mm_Log) * La dirección del vector (0^o) * Su ángulo de tolerancia (90^o) * 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

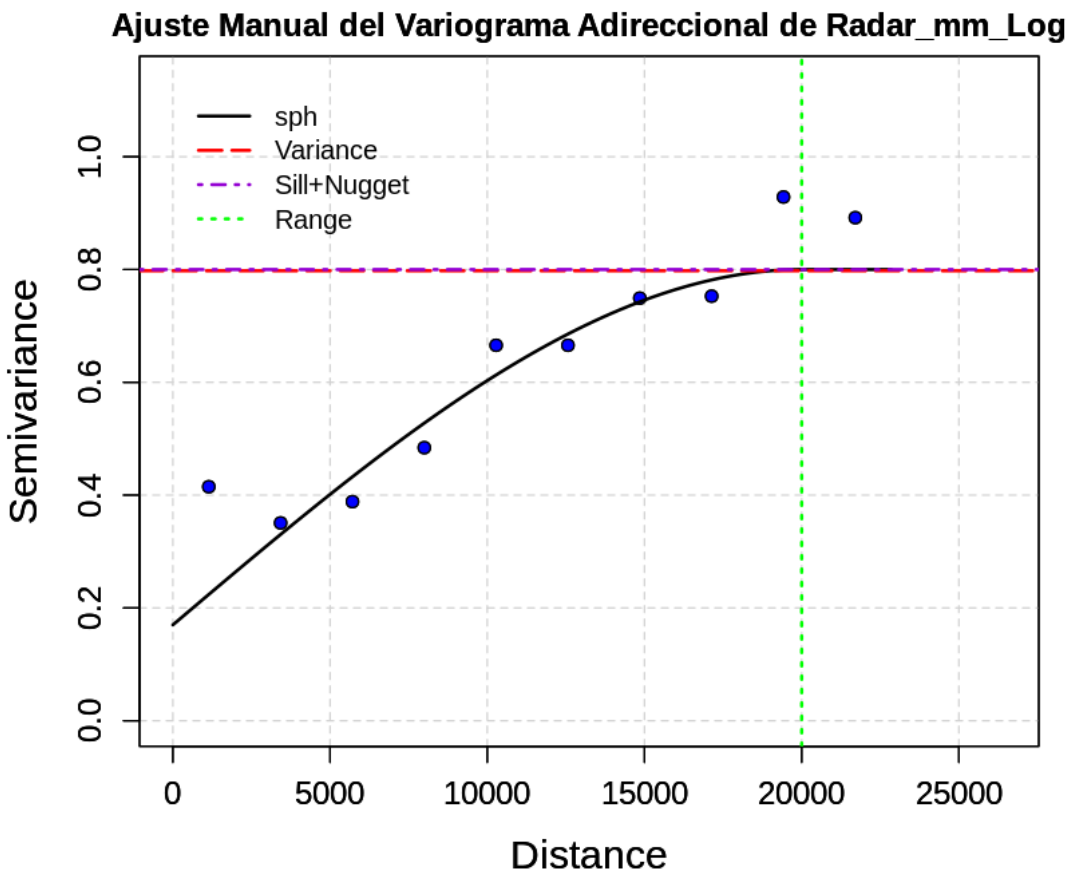
Después ingresamos: * Valor de nugget (nugget) * Valor de meseta más nugget (sill_and_nugget) * Alcance (rank).

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

[179]:
```
#modelos de variograma (1- exponential, 2- spherical, 3- gaussian)
Radar_mm_Log_vario_model<- 2
Radar_mm_Log_nugget<- 0.17
Radar_mm_Log_sill_and_nugget<- 0.8
Radar_mm_Log_rank <- 20000
```

[180]:
```
Radar_mm_Log_EyeModelVarioFit<-EyeModel(XCoord, YCoord,
                                        Radar_mm_Log, 0, 90, N_lags, lag_value, 1,
                                        Radar_mm_Log_vario_model,␣
↪Radar_mm_Log_nugget, Radar_mm_Log_sill_and_nugget, Radar_mm_Log_rank,
                                        "Ajuste Manual del Variograma Adireccional␣
↪de Radar_mm_Log")
```

```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.1700	0.8000	20000.0000	0.0699

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

El valor esperado

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

La varianza.

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

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

- Vectores de posicionamiento (XCoord, YCoord)
- La variable aleatoria (Pluv_mm_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.

[182]:
```
Radar_mm_Log_CrossValid<- CrossValidation(XCoord, YCoord,
                                          Radar_mm_Log, Radar_mm_Log_vario_model,␣
 ↪Radar_mm_Log_nugget, Radar_mm_Log_sill_and_nugget, Radar_mm_Log_rank,␣
 ↪MaxAnis=0, proporcion=1)
Radar_mm_Log_CrossValid
```

A data.frame: 50 × 5

	X	Y	Z	Z*	Z-Z*
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	485303	2162682	0.254642218	0.352984840	-0.09834262
2	484253	2157150	0.908258560	0.730428148	0.17783041
3	487403	2154937	1.026041596	0.696232351	0.32980924
4	490552	2153830	0.058268908	0.475526532	-0.41725762
5	489502	2151601	1.057790294	0.291970492	0.76581980
6	480054	2157142	0.482426149	0.679475914	-0.19704976
7	481104	2154828	0.879626748	0.714304052	0.16532270
8	483203	2152713	0.765467842	0.919633893	-0.15416605
9	475855	2149399	0.636576829	0.549815930	0.08676090
10	486353	2148284	0.806475866	0.659838986	0.14663688
11	482155	2149394	0.270027137	0.909578537	-0.63955140
12	492651	2147174	-0.562118918	-0.262311361	-0.29980756
13	494751	2149386	-1.309333320	-0.178534564	-1.13079876
14	477955	2144973	1.004301609	0.579839305	0.42446230
15	472694	2140554	-0.400477567	0.263859026	-0.66433659
16	473739	2137233	1.018847320	-0.335098564	1.35394588
17	481095	2139437	1.147402453	0.944101510	0.20330094
18	482154	2144968	0.378436436	0.918223025	-0.53978659
19	486348	2141645	0.819779831	0.724613505	0.09516633
20	471645	2141662	0.223143551	-0.116467472	0.33961102
21	468489	2138348	-0.843970070	-0.755178970	-0.08879110
22	462178	2133934	-1.714798428	-0.909562969	-0.80523546
23	474787	2135019	0.157003749	-0.074810123	0.23181387
24	472682	2132809	-0.843970070	-0.642469262	-0.20150081
25	485294	2137220	1.249901736	1.013812848	0.23608889
26	491597	2138323	0.182321557	0.532650300	-0.35032874
27	504199	2136108	0.239016900	0.287775835	-0.04875893
28	497899	2133895	0.329303747	0.972046642	-0.64274289
29	498949	2131682	1.393766376	0.768036213	0.62573016
30	480038	2133906	0.620576488	1.138496000	-0.51791951
31	486341	2132793	1.432700734	1.163643938	0.26905680
32	482135	2129478	2.052840860	0.499893708	1.55294715
33	485282	2122836	-0.040821995	0.811892049	-0.85271404
34	489495	2136111	0.652325186	0.896370063	-0.24404488
35	490542	2128365	0.405465108	1.071784413	-0.66631931
36	497898	2129469	2.000127735	0.393537991	1.60658974
37	477955	2157144	0.518793793	0.467687279	0.05110651
38	494751	2165983	-0.478035801	0.005459968	-0.48349577
39	473755	2148298	0.378436436	0.456673988	-0.07823755
40	466407	2144991	-0.673344553	-0.178927589	-0.49441696
41	485303	2152711	1.308332820	0.823899946	0.48443287
42	491602	2149387	-0.162518929	-0.073873177	-0.08864575
43	488451	2146070	0.678033543	0.324831718	0.35320182
44	471654	2147195	0.270027137	0.184363844	0.08566329
45	469543	2140559	0.009950331	-0.482923471	0.49287380
46	467433	2135030	-1.427116356	-1.187709652	-0.23940670
47	470588	2137238	-0.693147181	-0.634857603	-0.05828958
48	500000	2140533	-0.039220713	-0.114295578	0.15351629
49	503149	2123937	-1.660731207	1.108763511	-2.76949472
50	472686	2135022	-1.560647748	-0.228727300	-1.33192045

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.

[183]:
```
Radar_mm_Log_CrossValid_Stat <- Val_Estadisticos(Radar_mm_Log_CrossValid[1:
↪102,c(3,4,5)])
Radar_mm_Log_CrossValid_Stat
```

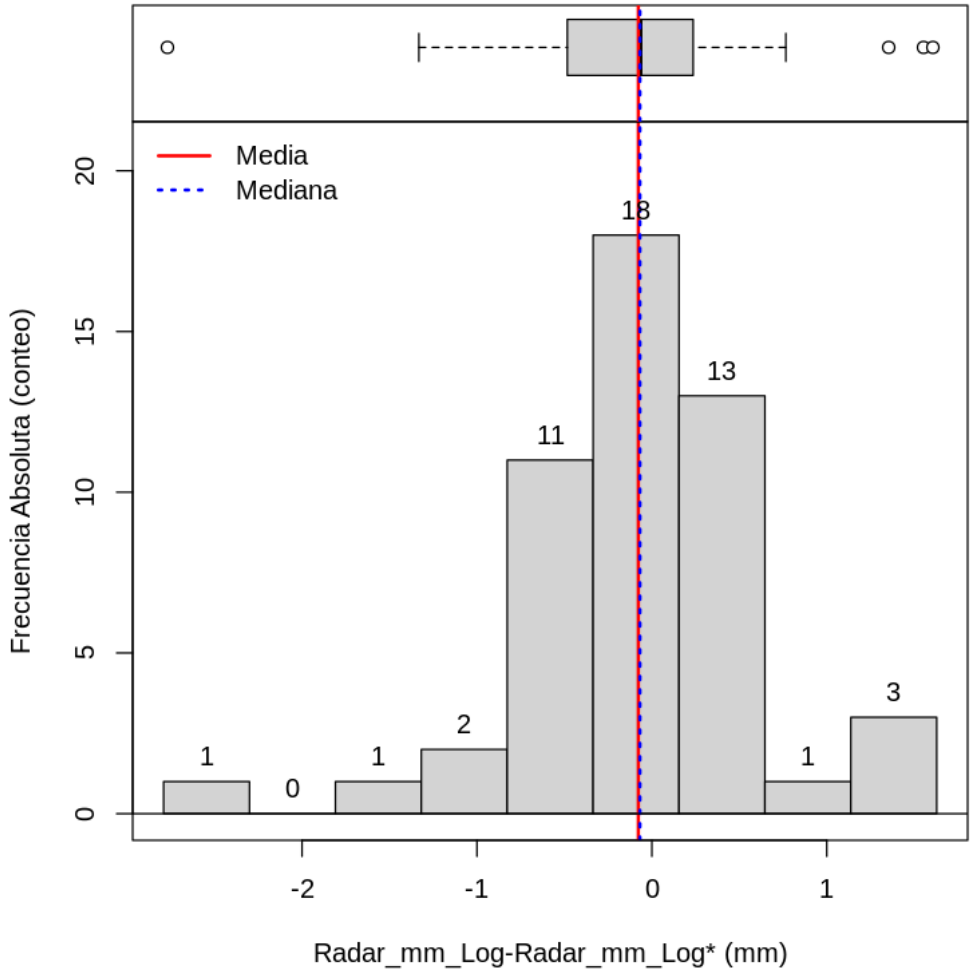
A data.frame: 13 × 3

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	50.00000	50.00000	50.00000
Minimo	-1.71480	-1.18771	-2.76949
Cuartil_1er	-0.13209	-0.10442	-0.46694
Mediana	0.35387	0.47161	-0.06826
Media	0.26569	0.34313	-0.07743
Cuartil_3er	0.86467	0.80093	0.23502
Maximo	2.05284	1.16364	1.60659
Rango	3.76764	2.35135	4.37608
Rango_Intercuartil	0.99676	0.90535	0.70196
Varianza	0.79776	0.34445	0.49699
Desv_Estandar	0.89317	0.58690	0.70497
Simetria	-0.46105	-0.68543	-0.63447
Curtosis	2.87206	2.65471	6.57202

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.

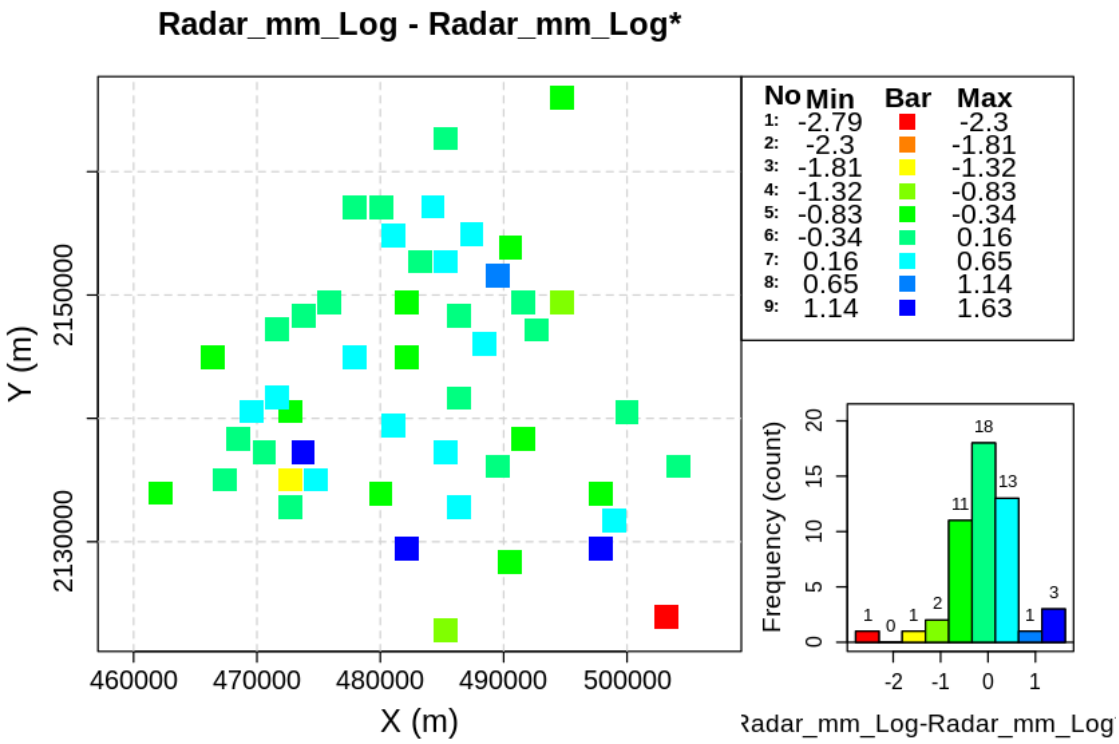
Ahora graficamos el histograma con los errores.

[184]:
```
HistBoxplot(x=Radar_mm_Log_CrossValid[,5], mean =␣
↪Radar_mm_Log_CrossValid_Stat[5,3], median = Radar_mm_Log_CrossValid_Stat[4,3],␣
↪main ="",
            xlab = "Radar_mm_Log-Radar_mm_Log* (mm)", 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".

[185]:
```
DEspacial(Radar_mm_Log_CrossValid[,1], Radar_mm_Log_CrossValid[,2],␣
↪Radar_mm_Log_CrossValid[,5],n_bins=9,
          'X (m)', 'Y (m)', 'Radar_mm_Log-Radar_mm_Log* (mm)', 'Radar_mm_Log -␣
↪Radar_mm_Log*')
```

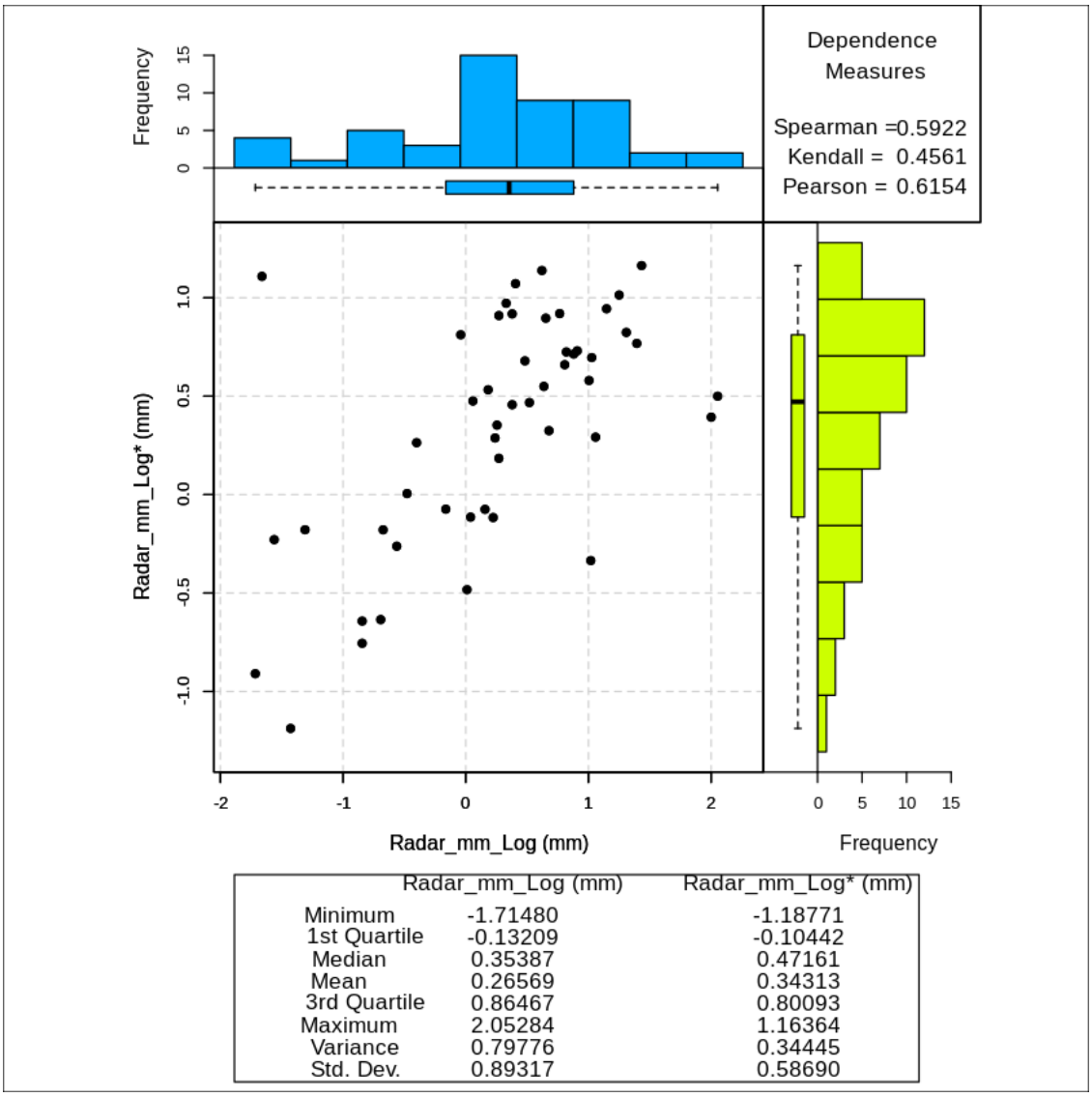


De este grafico podemos notar que el valor atípico negativo marcado en color rojo está en la frontera, por lo que es difícil saber si es un valor atípico. Mientras que los valores atípicos marcados en azul fuerte probablemente si sean valores atípicos espaciales, en especial el ubicado a la izquierda, ya que los valores vecinos tienen valores más pequeños en comparación.

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

[186]:
```
# Radar_mm_Log is the independent variable
X<-Radar_mm_Log_CrossValid[,3]
# Radar_mm_Log* is the dependent variable
Y<-Radar_mm_Log_CrossValid[,4]
```

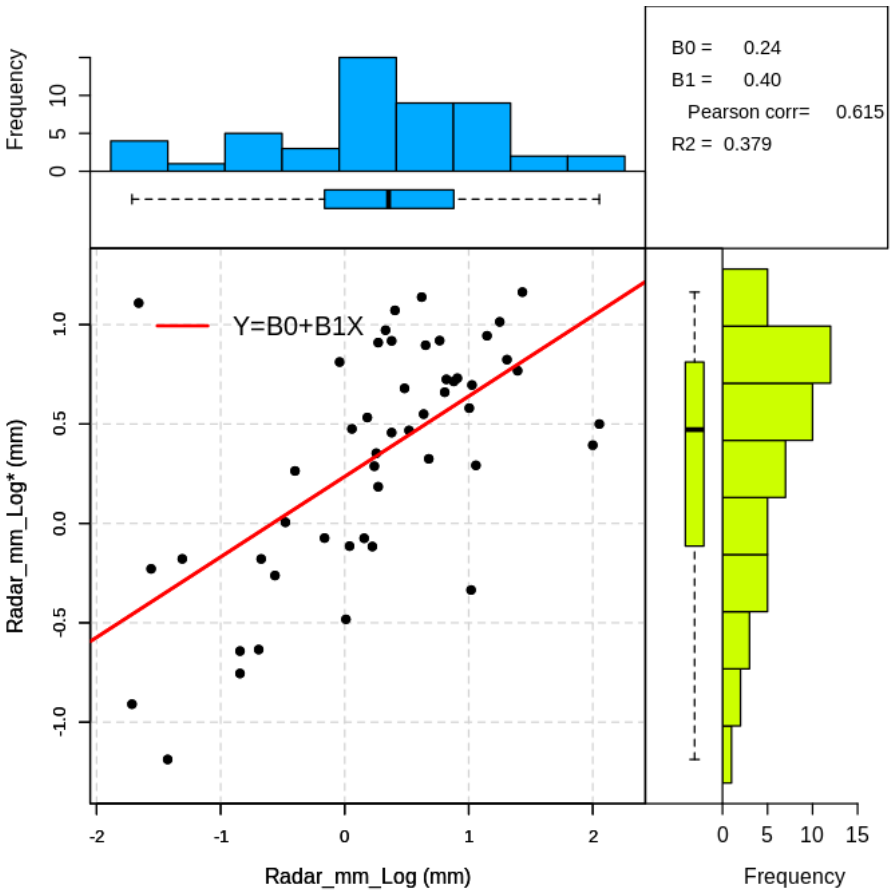
[187]:
```
ScatterPlot(Radar_mm_Log_CrossValid[,3] , Radar_mm_Log_CrossValid[,4], 9,
            Xmin = Radar_mm_Log_CrossValid_Stat[2,1], Xmax =␣
↪Radar_mm_Log_CrossValid_Stat[7,1],
            Ymin = Radar_mm_Log_CrossValid_Stat[2,2],Ymax =␣
↪Radar_mm_Log_CrossValid_Stat[7,2],
            XLAB = "Radar_mm_Log (mm)", YLAB = "Radar_mm_Log* (mm)")
```



	Radar_mm_Log (mm)	Radar_mm_Log* (mm)
Minimum	-1.71480	-1.18771
1st Quartile	-0.13209	-0.10442
Median	0.35387	0.47161
Mean	0.26569	0.34313
3rd Quartile	0.86467	0.80093
Maximum	2.05284	1.16364
Variance	0.79776	0.34445
Std. Dev.	0.89317	0.58690

[188]:
```
scaterplotReg(Radar_mm_Log_CrossValid[,3] , Radar_mm_Log_CrossValid[,4], 9,
            Xmin = Radar_mm_Log_CrossValid_Stat[2,1], Xmax =␣
↪Radar_mm_Log_CrossValid_Stat[7,1],
            Ymin = Radar_mm_Log_CrossValid_Stat[2,2],Ymax =␣
↪Radar_mm_Log_CrossValid_Stat[7,2],
```

```
XLAB = "Radar_mm_Log (mm)", YLAB = "Radar_mm_Log* (mm)")
```



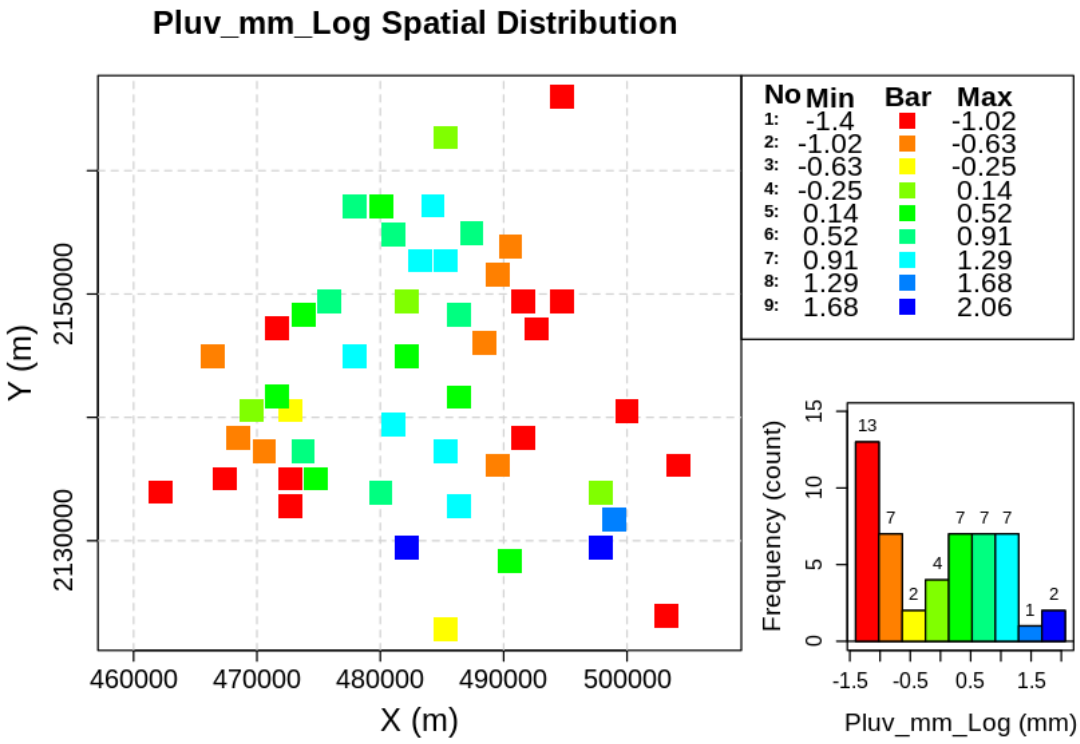
Observando los resultados del grafico de dispersión notamos que la dependencia no es lo suficientemente alta para ofrecer una buena representación de la variable aleatoria a partir del variograma propuesto.

4.2 Análisis variografico variable Pluv_mm_Log

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

[189]:

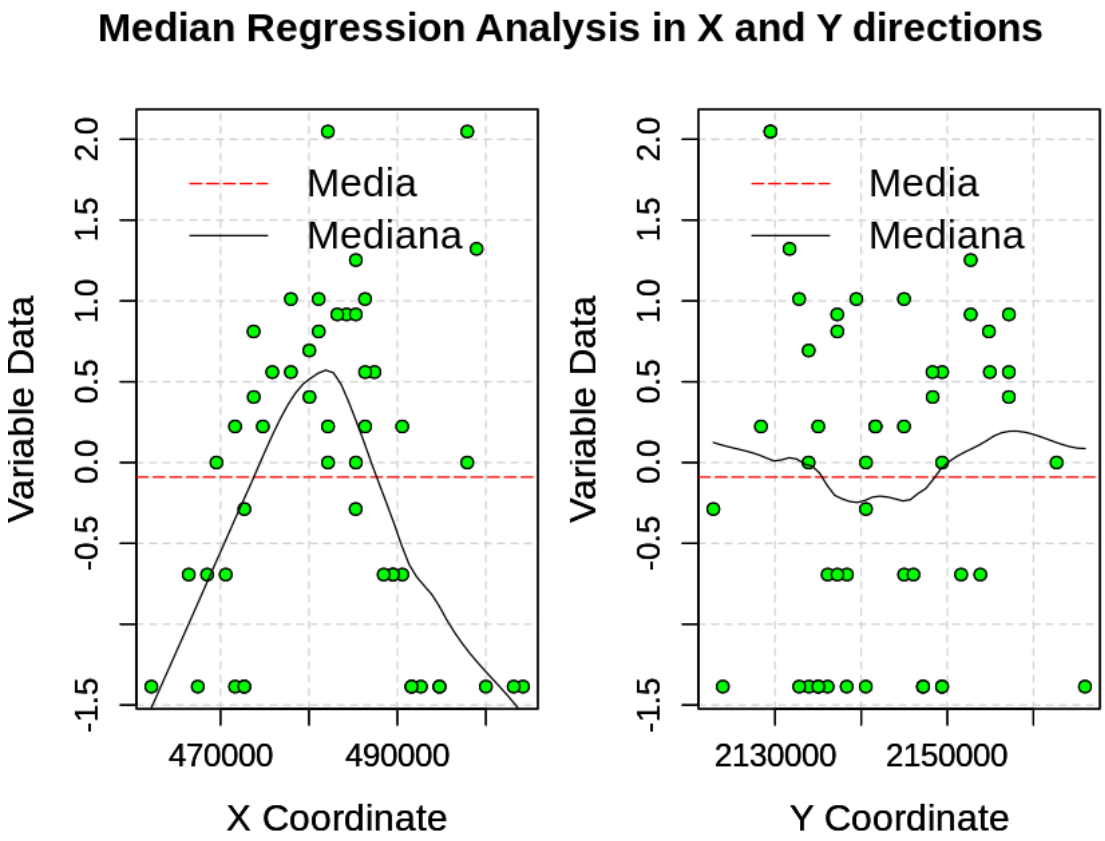
```
DEspacial(XCoord, YCoord, Pluv_mm_Log,n_bins=9,
          'X (m)', 'Y (m)', 'Pluv_mm_Log (mm)', 'Pluv_mm_Log Spatial␣
↪Distribution ')
```



4.2.1 Análisis de tendencia.

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

[190]:
```
GDirecciones(XCoord, YCoord, Pluv_mm_Log)
```



Respecto al análisis de tendencia usando la regresión de la mediana, se puede notar el mismo 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.

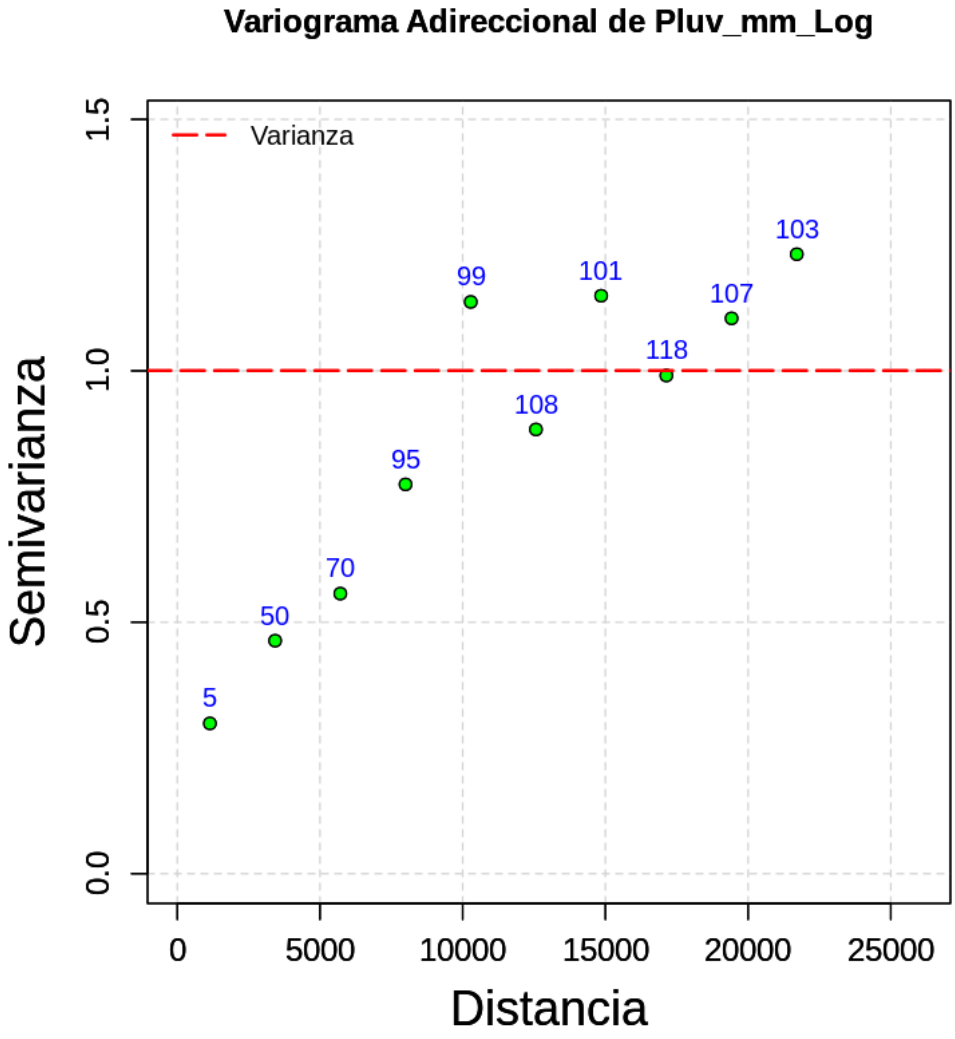
[191]:
```
N_lags<-10
DistMin<-min(dist(Data_File[,1:2])) # Minimum distance in data
DistMax<-max(dist(Data_File[,1:2])) # Maximum distance in data
lag_value<- (DistMax/2)/N_lags # DistMin
```

Y calculamos el variograma.

[192]:
```
Pluv_mm_Log_VarioEstimation<-Variograma(XCoord, YCoord,
```

```
                                        Pluv_mm_Log, 0, 90, 1*N_lags,␣
↪lag_value, 1,
                                       "Variograma Adireccional de Pluv_mm_Log")
```

```
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.

[193]:
```
Pluv_mm_Log_VarioEstimation
```

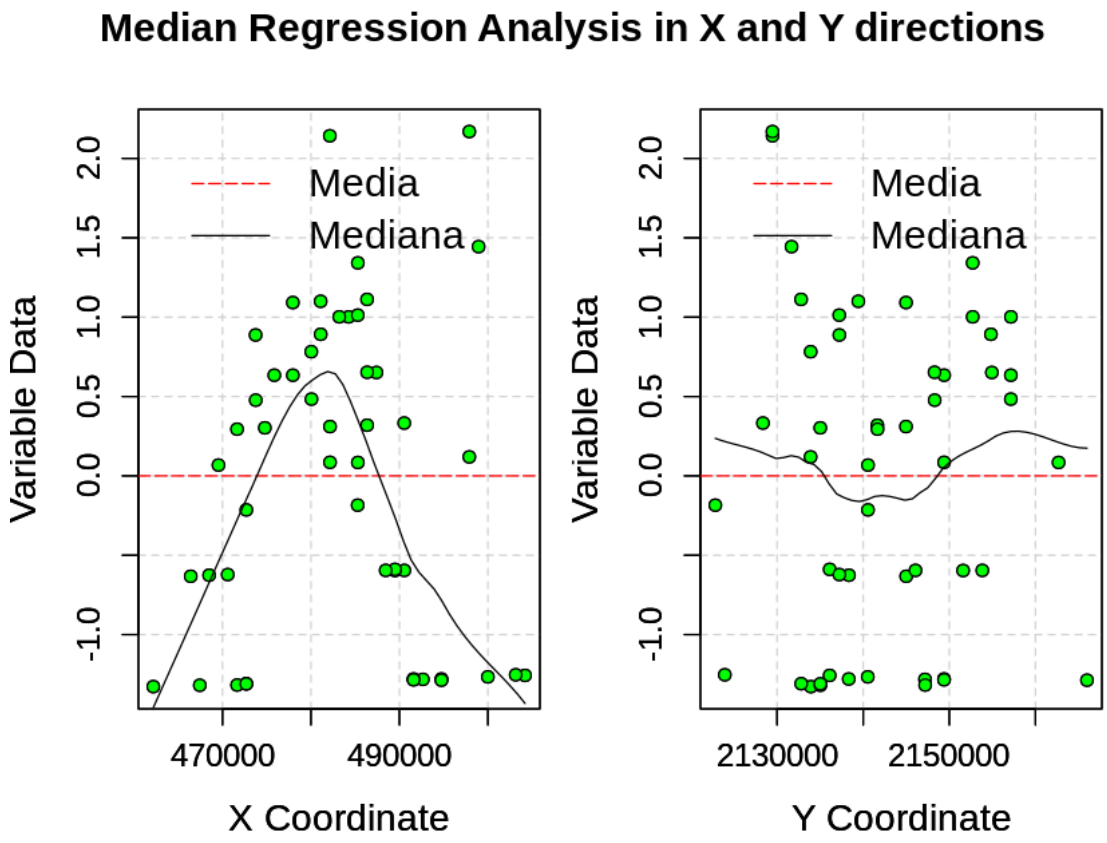
A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
5	1142.404	0.2988209
50	3427.212	0.4633333
70	5712.020	0.5569420
95	7996.827	0.7741376
99	10281.635	1.1369521
108	12566.443	0.8833747
101	14851.251	1.1492244
118	17136.059	0.9906020
107	19420.867	1.1042933
103	21705.674	1.2318517

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"

[194]:
```
pol_degree=1
Pluv_mm_Log_Detrended_1<-Trend(XCoord, YCoord,
                               Pluv_mm_Log, pol_degree)
```

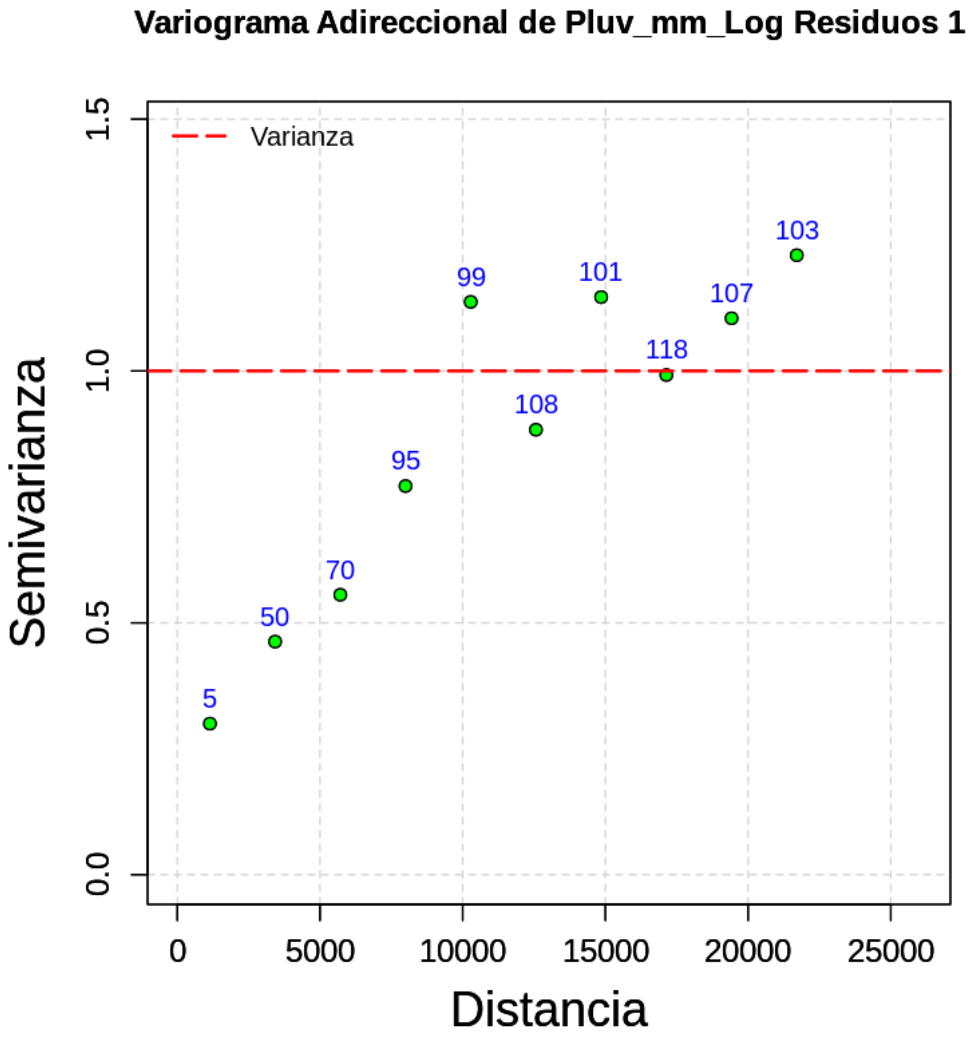
[195]:
```
GDirecciones(Pluv_mm_Log_Detrended_1[,1], Pluv_mm_Log_Detrended_1[,2],␣
↪Pluv_mm_Log_Detrended_1[,3])
```



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.

[196]:
```
Pluv_mm_Log_Detrended_1_VarioEstimation<-Variograma(Pluv_mm_Log_Detrended_1[,1],␣
↪Pluv_mm_Log_Detrended_1[,2],
                                                    ␣
↪Pluv_mm_Log_Detrended_1[,3], 0, 90, N_lags, lag_value, 1,
                                                      "Variograma Adireccional de␣
↪Pluv_mm_Log Residuos 1")
```

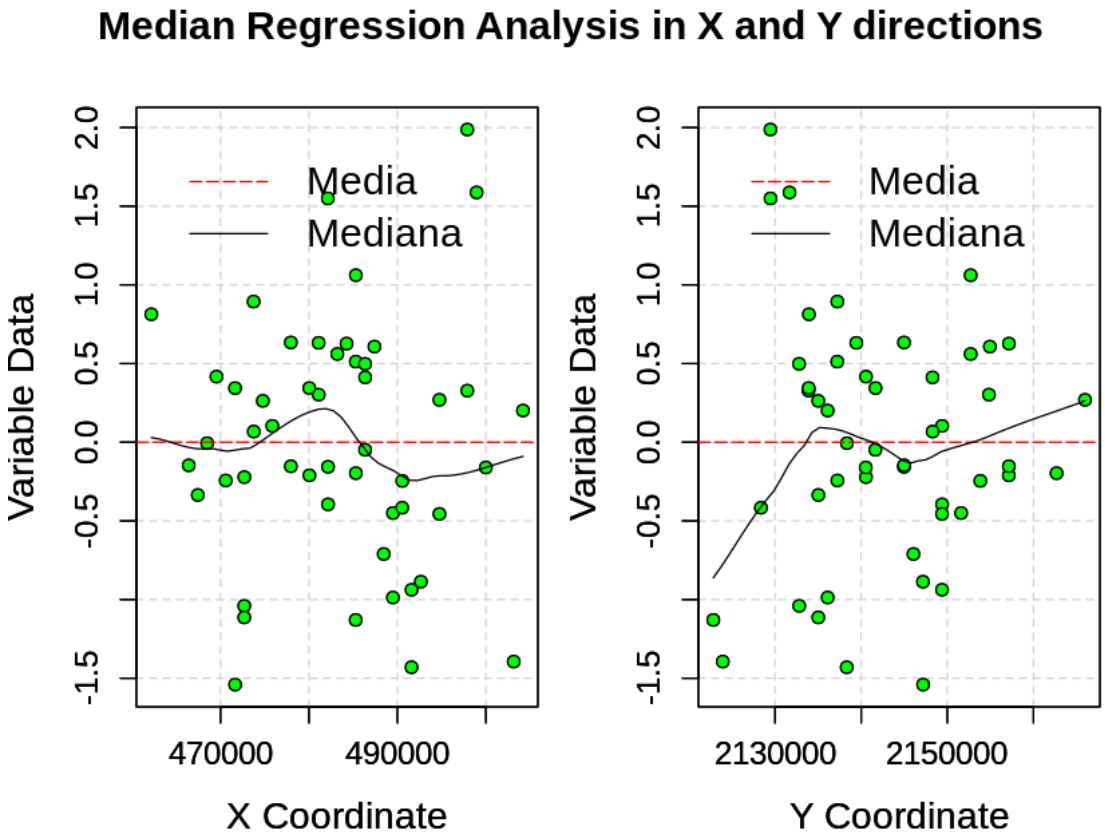
```
variog: computing omnidirectional variogram
```



Para usar la transformación de segundo orden.

[197]:
```
pol_degree=2
Pluv_mm_Log_Detrended_2<-Trend(XCoord, YCoord,
                               Pluv_mm_Log, pol_degree)
```

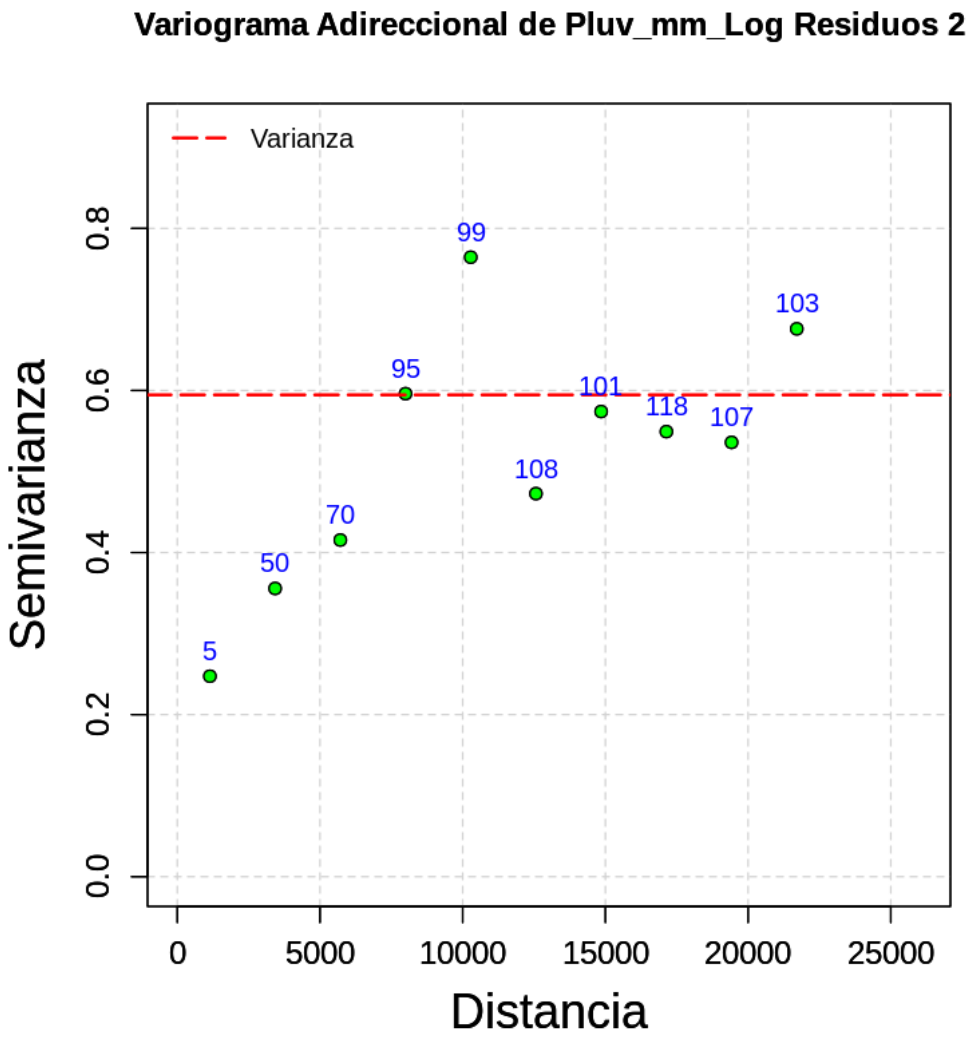
[198]:
```
GDirecciones(Pluv_mm_Log_Detrended_2[,1], Pluv_mm_Log_Detrended_2[,2],␣
↪Pluv_mm_Log_Detrended_2[,3])
```



Usando la transformada de segundo orden podemos notar que la coordenada X tuvo una mejora, sin embargo la coordenada Y empeoró su regresión.

[199]:
```
Pluv_mm_Log_Detrended_2_VarioEstimation<-Variograma(Pluv_mm_Log_Detrended_2[,1],␣
↪Pluv_mm_Log_Detrended_2[,2],
                                                    ␣
↪Pluv_mm_Log_Detrended_2[,3], 0, 90, N_lags, lag_value, 1,
                                                      "Variograma Adireccional de␣
↪Pluv_mm_Log Residuos 2")
```

```
variog: computing omnidirectional variogram
```

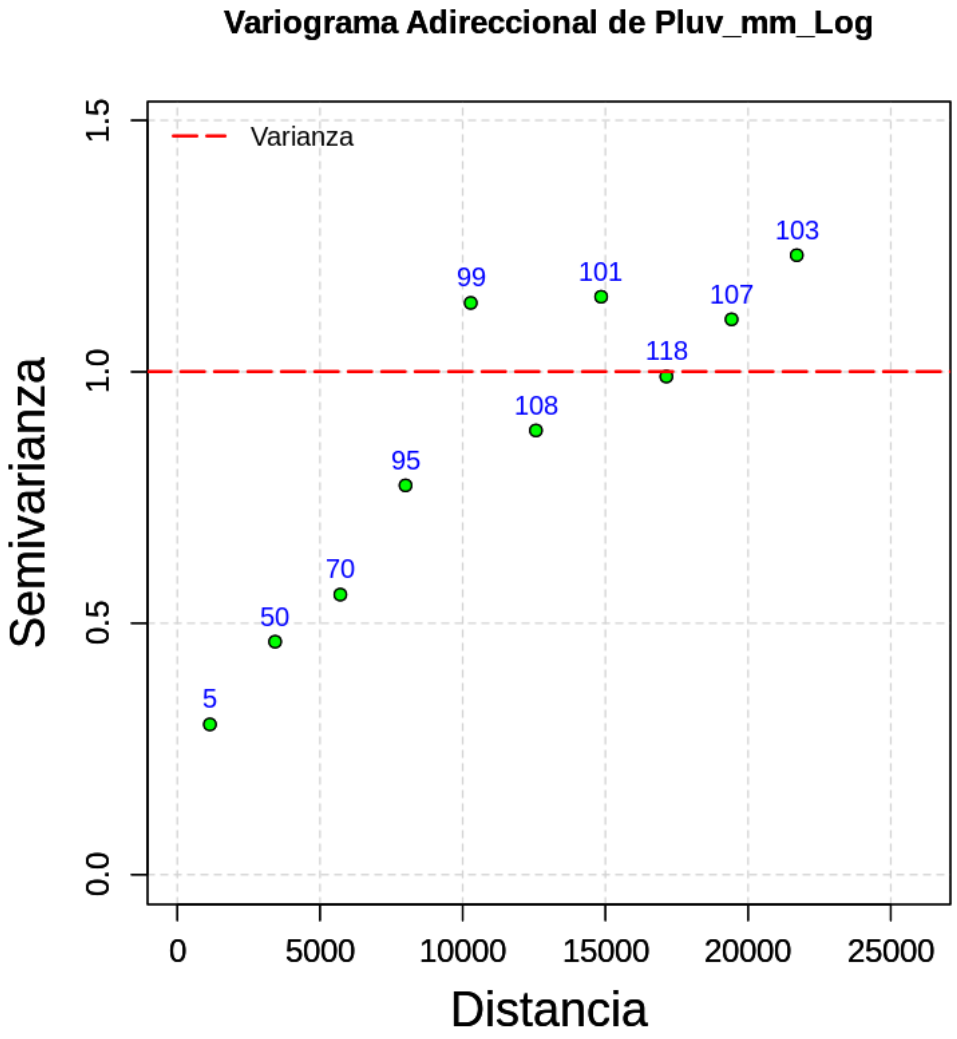


El variograma que se obtuvo con la transformación de segundo orden empeoro. Este ejemplo sirve para notar el efecto de la transformación, si se usa de forma incorrecta podríamos tener una peor estimación del variograma experimental.

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

[200]:
```
Pluv_mm_Log_VarioEstimation<-Variograma(XCoord, YCoord,
                                        Pluv_mm_Log, 0, 90, N_lags, lag_value,␣
↪1,
                                       "Variograma Adireccional de Pluv_mm_Log")
```

```
variog: computing omnidirectional variogram
```



[201]:
```
Pluv_mm_Log_VarioEstimation
```

A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
5	1142.404	0.2988209
50	3427.212	0.4633333
70	5712.020	0.5569420
95	7996.827	0.7741376
99	10281.635	1.1369521
108	12566.443	0.8833747
101	14851.251	1.1492244
118	17136.059	0.9906020
107	19420.867	1.1042933
103	21705.674	1.2318517

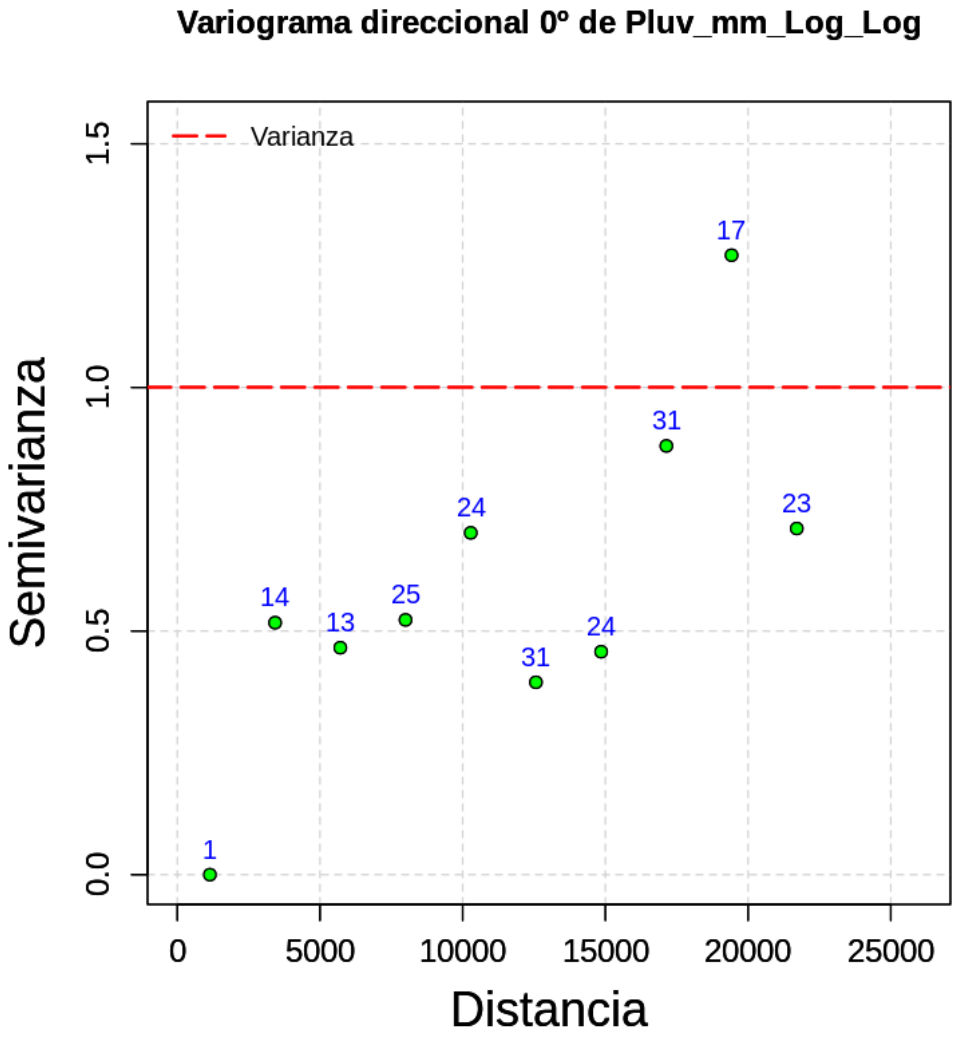
Ahora que sabemos cuál es el mejor variograma procedemos a calcular los variogramas direccionales y determinar si existe algún tipo de anisotropía. Comenzamos con la dirección 0º.

[202]:
```
Pluv_mm_Log_VarioEstimation_0<-Variograma(XCoord, YCoord,
                                          Pluv_mm_Log, 0, 22.5, N_lags,␣
↪lag_value, 1, "Variograma direccional 0º de Pluv_mm_Log_Log")
Pluv_mm_Log_VarioEstimation_0
```

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

A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
1	1142.404	0.0000000
14	3427.212	0.5171901
13	5712.020	0.4657970
25	7996.827	0.5229390
24	10281.635	0.7015614
31	12566.443	0.3949000
24	14851.251	0.4575810
31	17136.059	0.8799773
17	19420.867	1.2713896
23	21705.674	0.7104754



En esta dirección se puede observar que este variograma no está bien estimado, solo dos intervalos superan los 30 pares.

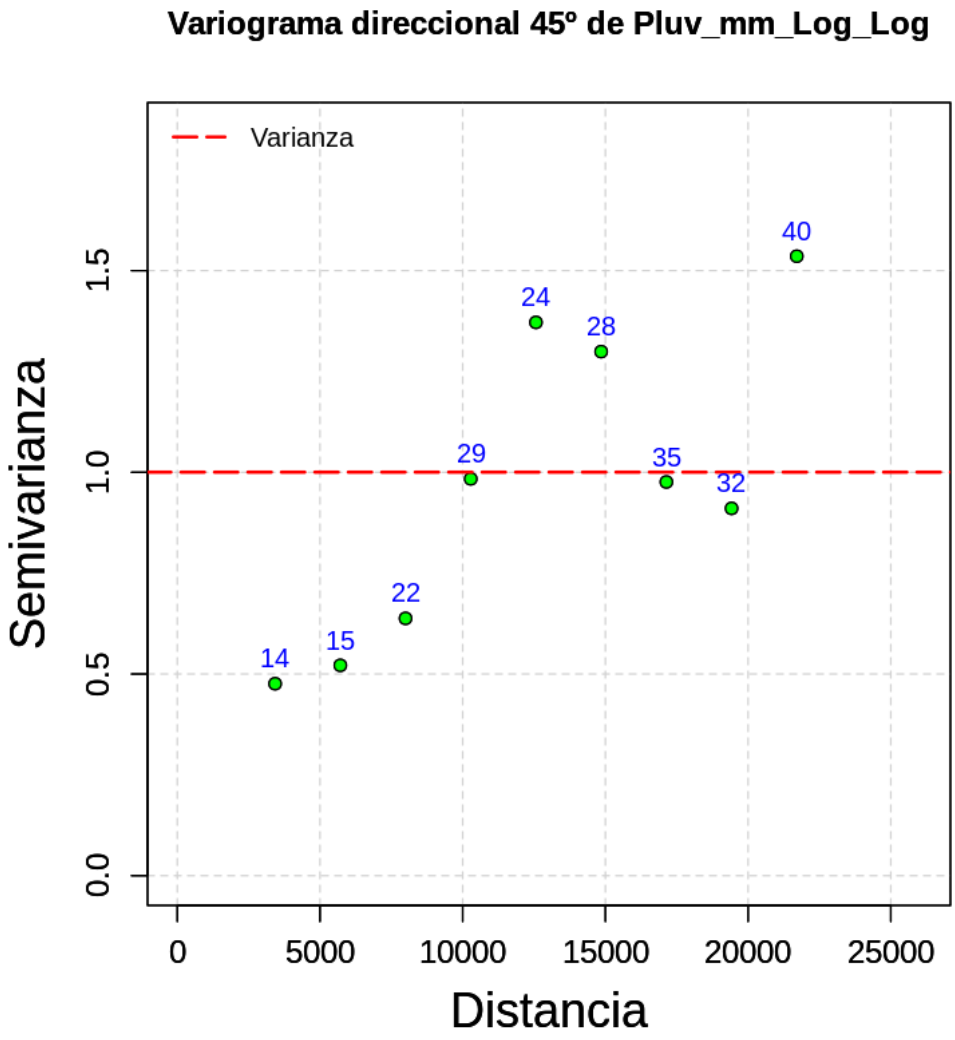
Seguimos con la dirección de 45º.

[203]:
```
Pluv_mm_Log_VarioEstimation_45<-Variograma(XCoord, YCoord,
                                           Pluv_mm_Log, 45, 22.5, N_lags,␣
 ↪lag_value, 1, "Variograma direccional 45º de Pluv_mm_Log_Log")
Pluv_mm_Log_VarioEstimation_45
```

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

A data.frame: 9 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
14	3427.212	0.4759408
15	5712.020	0.5214920
22	7996.827	0.6379057
29	10281.635	0.9838092
24	12566.443	1.3717237
28	14851.251	1.2991847
35	17136.059	0.9760720
32	19420.867	0.9106024
40	21705.674	1.5357749



El variograma con dirección 45º tampoco está bien estimado, solo tres intervalos superan los 30 pares.

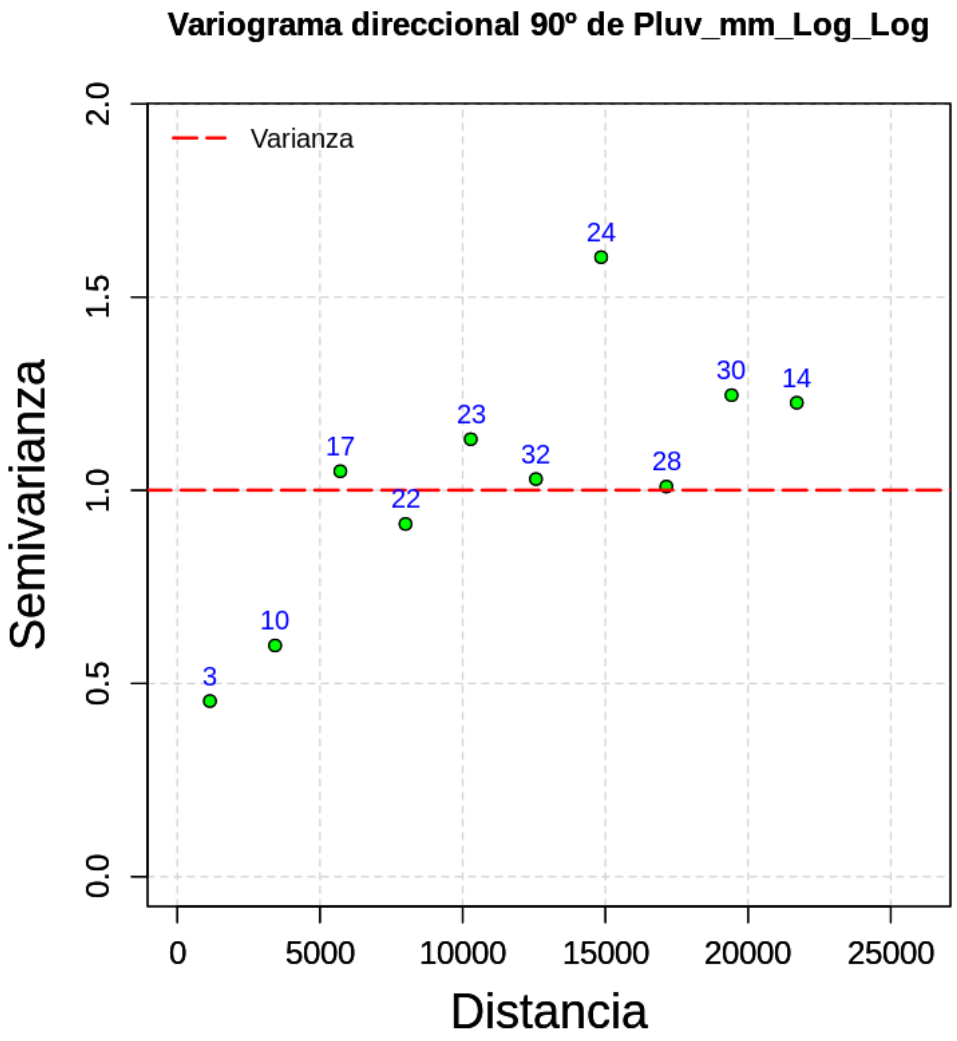
Calculamos el variograma con dirección 90º.

[204]:
```
Pluv_mm_Log_VarioEstimation_90<-Variograma(XCoord, YCoord,
                                           Pluv_mm_Log, 90, 22.5, N_lags,␣
 ↪lag_value, 1, "Variograma direccional 90º de Pluv_mm_Log_Log")
Pluv_mm_Log_VarioEstimation_90
```

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

A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
3	1142.404	0.4545444
10	3427.212	0.5983756
17	5712.020	1.0493852
22	7996.827	0.9129011
23	10281.635	1.1322249
32	12566.443	1.0291239
24	14851.251	1.6033039
28	17136.059	1.0096557
30	19420.867	1.2462150
14	21705.674	1.2266749



En este caso podemos notar que el variograma no está bien estimado, solo dos intervalos tienen más de 30 pares.

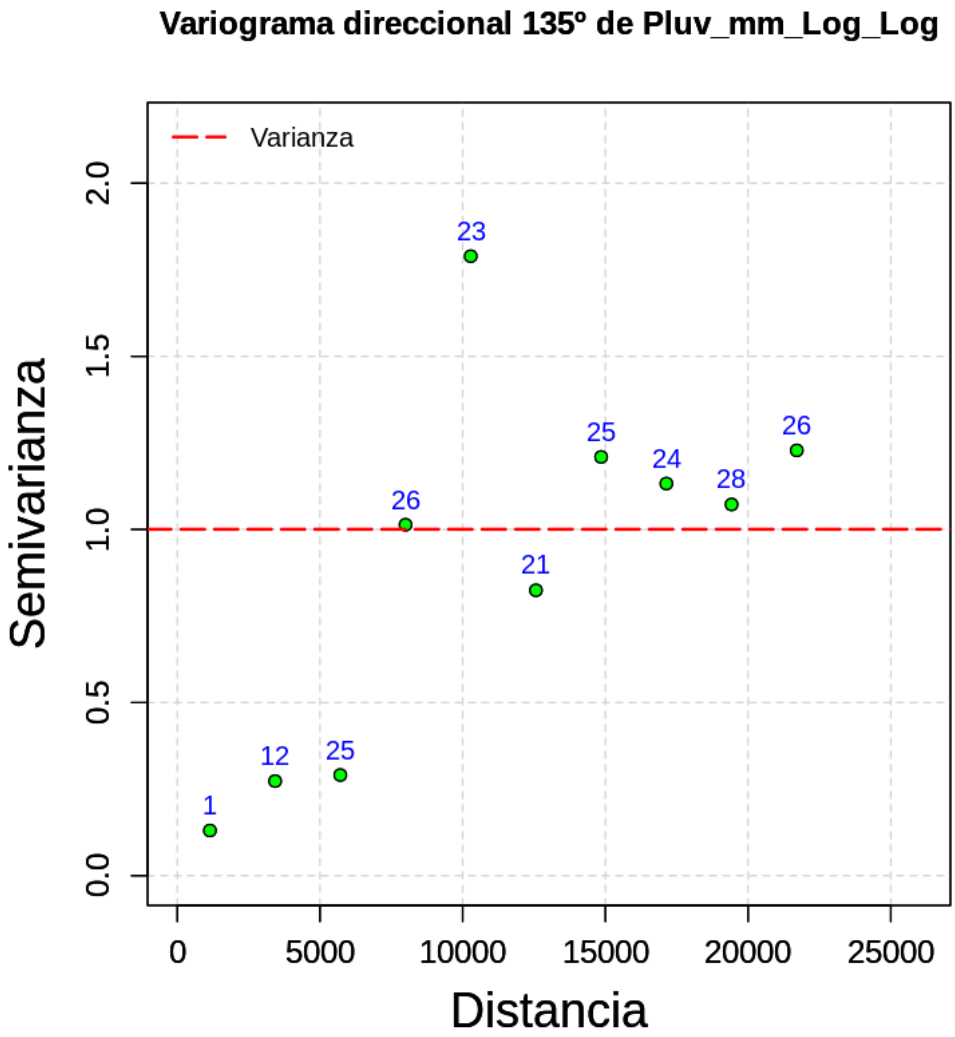
Calculamos la dirección 135º.

[205]:
```
Pluv_mm_Log_VarioEstimation_135<-Variograma(XCoord, YCoord,
                                            Pluv_mm_Log, 135, 22.5, N_lags,␣
 ↪lag_value, 1, "Variograma direccional 135º de Pluv_mm_Log_Log")
Pluv_mm_Log_VarioEstimation_135
```

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

A data.frame: 10 × 3

Npares <dbl>	Lags <dbl>	Semivarianzas <dbl>
1	1142.404	0.1304714
12	3427.212	0.2732561
25	5712.020	0.2907460
26	7996.827	1.0135327
23	10281.635	1.7890932
21	12566.443	0.8242492
25	14851.251	1.2093302
24	17136.059	1.1324523
28	19420.867	1.0721439
26	21705.674	1.2282827



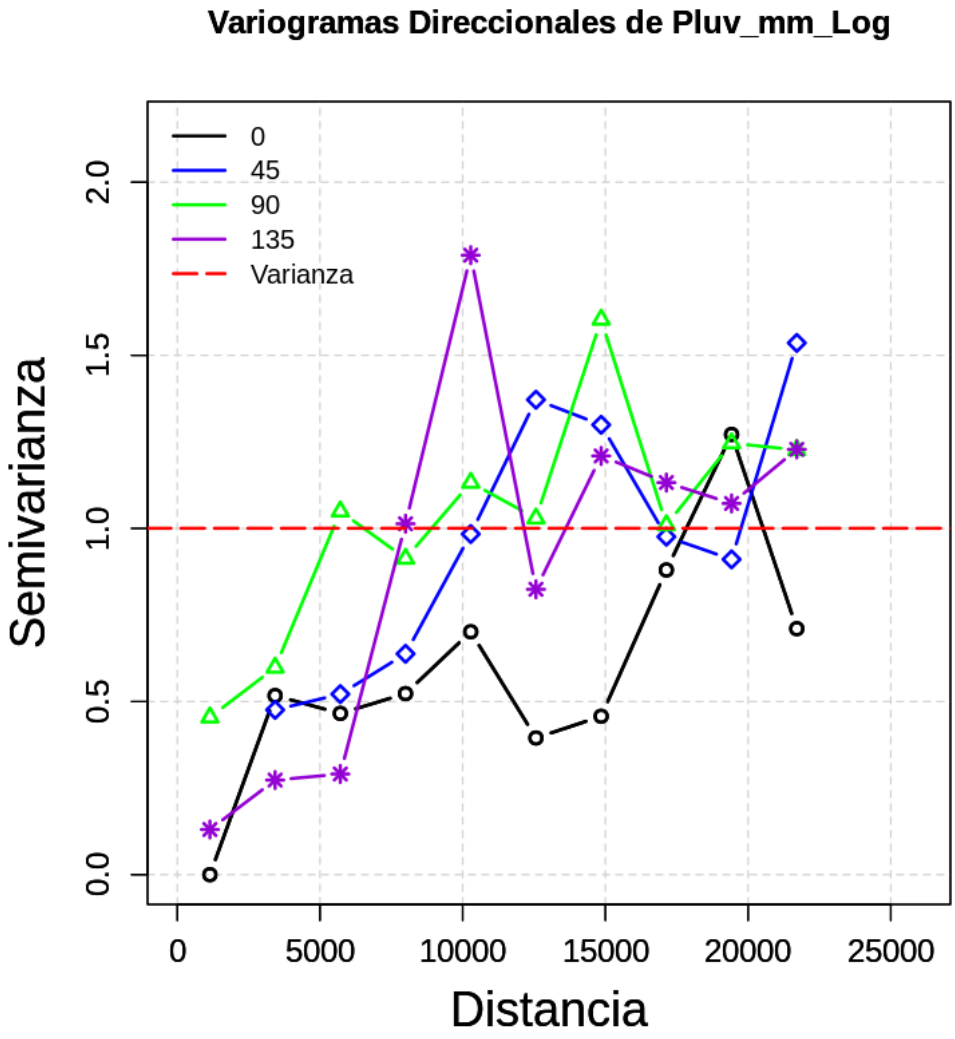
Con este caso podemos notar que es el peor variograma experimental, no hay intervalo que tenga

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.

[206]:
```
Pluv_mm_Log_VarioEstimation4D<-Variograma4D(XCoord, YCoord,
                                            Pluv_mm_Log, 0, 45, 90, 135, 22.5,␣
↪N_lags, lag_value, 1,
                                               "Variogramas Direccionales de␣
↪Pluv_mm_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)
```



[207]:
```
Pluv_mm_Log_VarioEstimation4D
```

$Zero A matrix: 10 × 3 of type dbl

1	1142.404	0.0000000
14	3427.212	0.5171901
13	5712.020	0.4657970
25	7996.827	0.5229390
24	10281.635	0.7015614
31	12566.443	0.3949000
24	14851.251	0.4575810
31	17136.059	0.8799773
17	19420.867	1.2713896
23	21705.674	0.7104754

$FortyFive A matrix: 9 × 3 of type dbl

14	3427.212	0.4759408
15	5712.020	0.5214920
22	7996.827	0.6379057
29	10281.635	0.9838092
24	12566.443	1.3717237
28	14851.251	1.2991847
35	17136.059	0.9760720
32	19420.867	0.9106024
40	21705.674	1.5357749

$Ninety A matrix: 10 × 3 of type dbl

3	1142.404	0.4545444
10	3427.212	0.5983756
17	5712.020	1.0493852
22	7996.827	0.9129011
23	10281.635	1.1322249
32	12566.443	1.0291239
24	14851.251	1.6033039
28	17136.059	1.0096557
30	19420.867	1.2462150
14	21705.674	1.2266749

$OneThertyFive A matrix: 10 × 3 of type dbl

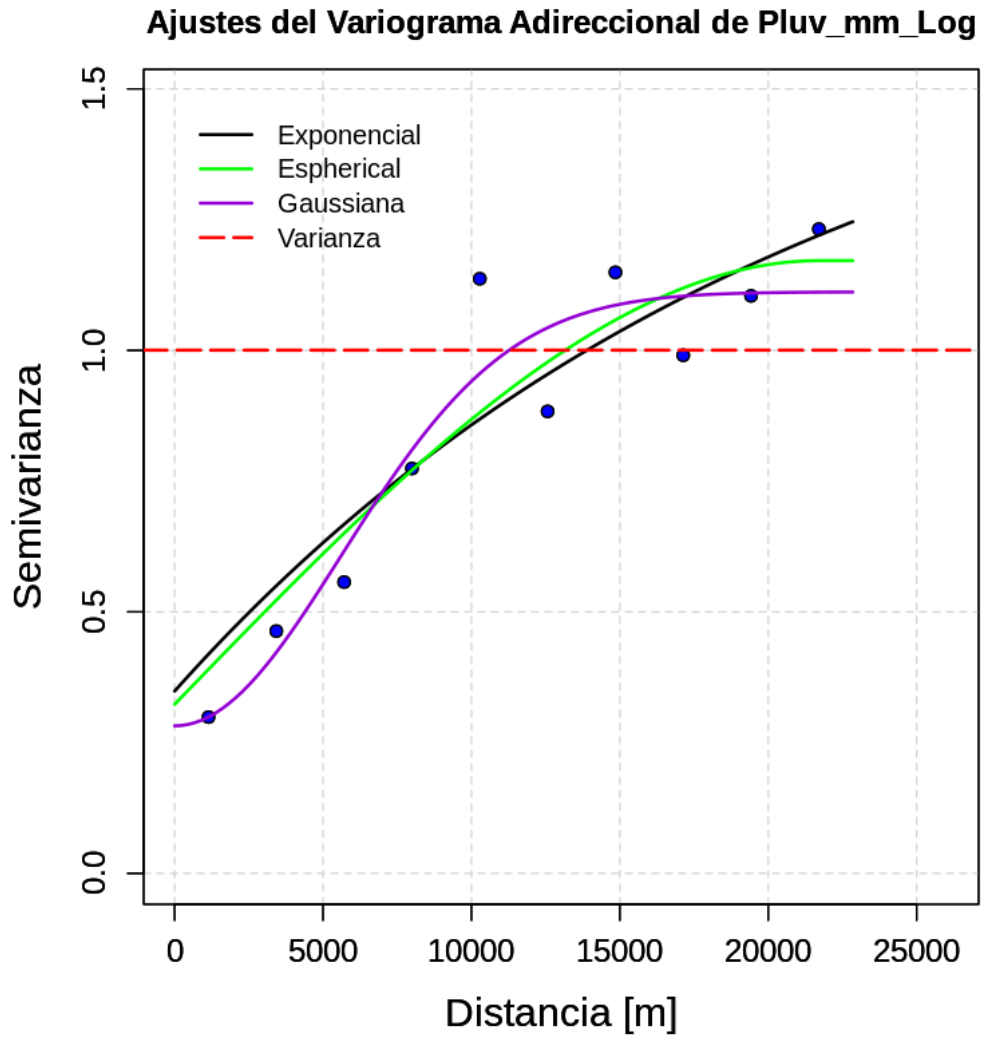
1	1142.404	0.1304714
12	3427.212	0.2732561
25	5712.020	0.2907460
26	7996.827	1.0135327
23	10281.635	1.7890932
21	12566.443	0.8242492
25	14851.251	1.2093302
24	17136.059	1.1324523
28	19420.867	1.0721439
26	21705.674	1.2282827

Ahora estimamos los modelos de variograma autorizados:

[208]:
```
Pluv_mm_Log_AllModelVarioFit<-AllModel(XCoord, YCoord,
                                       Pluv_mm_Log, 0, 90, N_lags, lag_value, 1,
                                          "Ajustes del Variograma Adireccional de␣
 ↪Pluv_mm_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
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
```



[209]:
```
Pluv_mm_Log_AllModelVarioFit
```

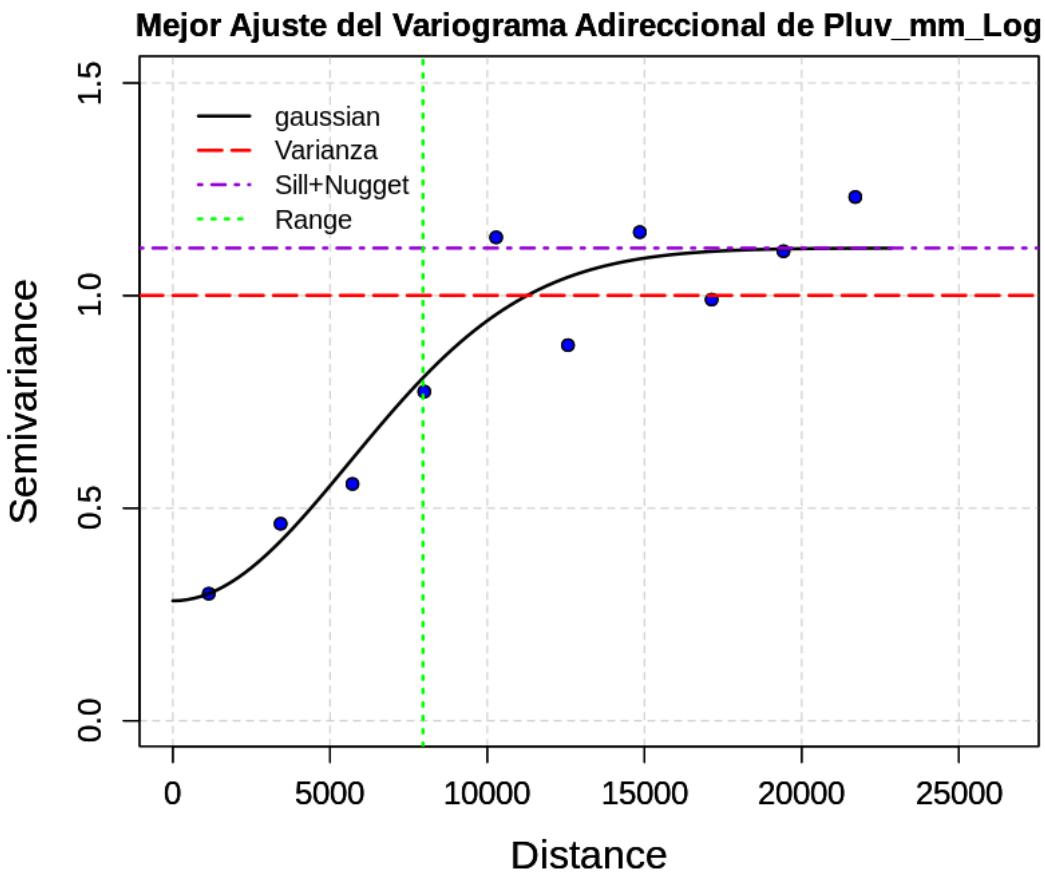
A matrix: 3 × 4 of type dbl

	Nugget	Meseta+Nugget	Alcance	SCE
exponential	0.3484807	1.727137	21705.000	0.14093110
spherical	0.3232237	1.171616	21705.000	0.12636551
gaussian	0.2820670	1.111793	7955.107	0.09853298

Dados los resultados del ajuste automático podemos ver que sus errores no tienen diferencias significativas. Pero podemos considerar que el mejor modelo estimado es el gaussiano.

[210]:
```
Pluv_mm_Log_BestModelVarioFit<-BestModel(XCoord, YCoord,
                                         Pluv_mm_Log, 0, 90, N_lags, lag_value, 1,
                                            "Mejor Ajuste del Variograma␣
 ↪Adireccional de Pluv_mm_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
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 gaussian
variofit: weights used: npairs
variofit: minimisation function used: optim
```



Model	Nugget	Sill+Nugget	Range	MSE
gaussian	0.2821	1.1118	7955.1073	0.0962

[211]:
```
Pluv_mm_Log_BestModelVarioFit
```

A matrix: 1 × 6 of type dbl

	Nugget	Meseta+Nugget	Alcance	SCE	MaxY	MinY
gaussian	0.282067	1.111793	7955.107	0.09623212	1.231852	0.2988209

Si analizamos el modelo Gaussiano propuesto en este ajuste automático, podemos notar que el modelo es forzado a ajustarse, indicando un valor de nugget que no corresponde al sugerido por el variograma experimental, por lo tanto, se recomienda usar otro modelo.

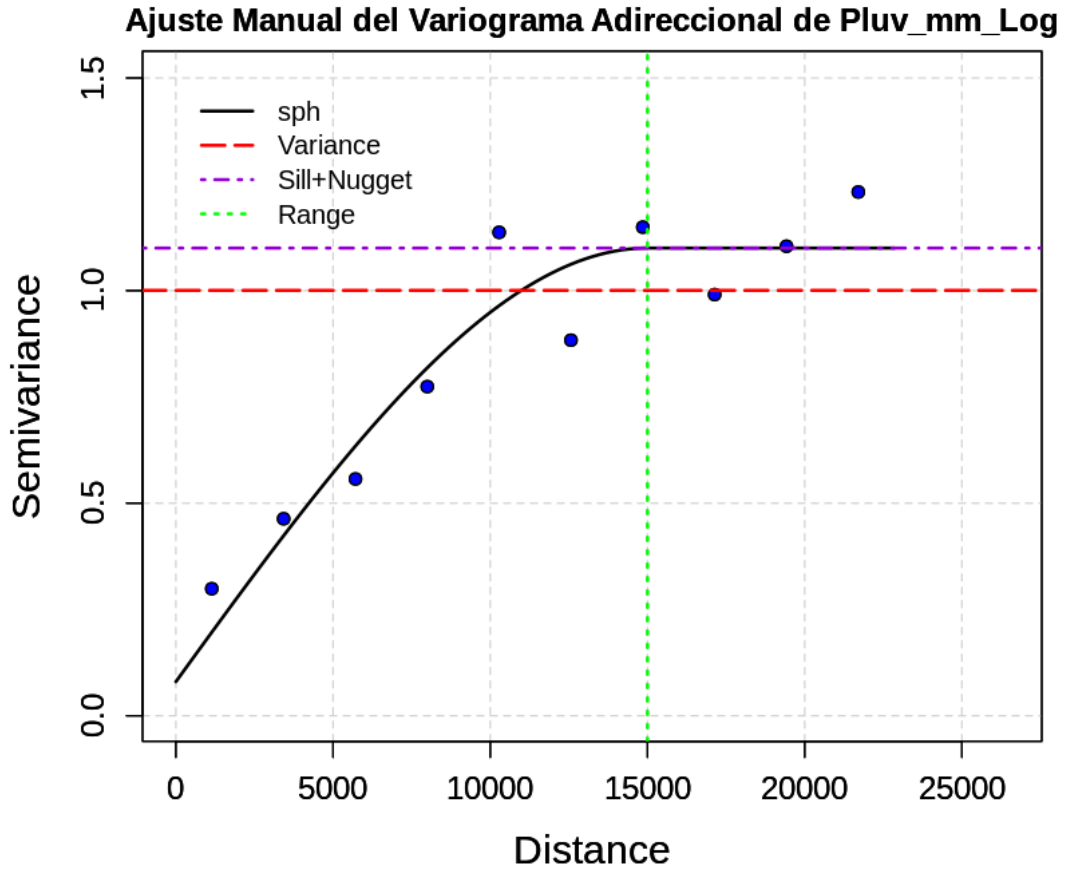
Ajustamos el variograma a un modelo esférico de forma manual y obtenemos lo siguiente:

[212]:
```
#modelos de variograma (1- exponential, 2- spherical, 3- gaussian)
vario_model<- 2 # 2
nugget<- 0.08 # 0.08
```

```
sill_and_nugget<- 1.1 # 1.1
rank <- 15000 # 15000
```

[213]:
```
Pluv_mm_Log_EyeModelVarioFit<-EyeModel(XCoord, YCoord,
                                       Pluv_mm_Log, 0, 90, N_lags, lag_value, 1,
                                       vario_model, nugget, sill_and_nugget, rank,
                                          "Ajuste Manual del Variograma␣
 ↪Adireccional de Pluv_mm_Log")
```

```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0800	1.1000	15000.0000	0.1135

Ya que tenemos el mejor ajuste calculamos la validación cruzada.

[214]:
```
Pluv_mm_Log_CrossValid<- CrossValidation(XCoord, YCoord,
                                         Pluv_mm_Log, vario_model, nugget,␣
 ↪sill_and_nugget, rank,
                                            MaxAnis=0, proporcion=1)
Pluv_mm_Log_CrossValid
```

A data.frame: 50 × 5

	X	Y	Z	Z*	Z-Z*
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	485303	2162682	0.0000000	-0.06425495	0.064254949
2	484253	2157150	0.9162907	0.66992096	0.246369776
3	487403	2154937	0.5596158	0.37333787	0.186277916
4	490552	2153830	-0.6931472	-0.45605034	-0.237096845
5	489502	2151601	-0.6931472	-0.40868722	-0.284459962
6	480054	2157142	0.4054651	0.69411959	-0.288654483
7	481104	2154828	0.8109302	0.78943165	0.021498564
8	483203	2152713	0.9162907	0.91233182	0.003958910
9	475855	2149399	0.5596158	0.36395926	0.195656524
10	486353	2148284	0.5596158	-0.18148616	0.741101946
11	482155	2149394	0.0000000	0.87415516	-0.874155163
12	492651	2147174	-1.3862944	-1.43082205	0.044527693
13	494751	2149386	-1.3862944	-1.42033062	0.034036255
14	477955	2144973	1.0116009	0.45532103	0.556279877
15	472694	2140554	-0.2876821	0.38421625	-0.671898320
16	473739	2137233	0.8109302	-0.29213702	1.103067233
17	481095	2139437	1.0116009	0.91224194	0.099358971
18	482154	2144968	0.2231436	0.70952495	-0.486381394
19	486348	2141645	0.2231436	-0.11692038	0.340063933
20	471645	2141662	0.2231436	-0.20986848	0.433012029
21	468489	2138348	-0.6931472	-0.69034992	-0.002797265
22	462178	2133934	-1.3862944	-1.06255275	-0.323741610
23	474787	2135019	0.2231436	-0.18754665	0.410690199
24	472682	2132809	-1.3862944	-0.86699160	-0.519302758
25	485294	2137220	0.9162907	0.51481184	0.401478893
26	491597	2138323	-1.3862944	-0.79385931	-0.592435051
27	504199	2136108	-1.3862944	-0.44771027	-0.938584089
28	497899	2133895	0.0000000	0.42504901	-0.425049010
29	498949	2131682	1.3217558	0.74941584	0.572339996
30	480038	2133906	0.6931472	1.35820262	-0.665055441
31	486341	2132793	1.0116009	0.70209707	0.309503843
32	482135	2129478	2.0476928	0.41162297	1.636069876
33	485282	2122836	-0.2876821	1.20108895	-1.488771021
34	489495	2136111	-0.6931472	-0.13854155	-0.554605632
35	490542	2128365	0.2231436	0.85146851	-0.628324957
36	497898	2129469	2.0476928	0.66157096	1.386121879
37	477955	2157144	0.5596158	0.31525677	0.244359015
38	494751	2165983	-1.3862944	-0.09925321	-1.287041155
39	473755	2148298	0.4054651	-0.32413095	0.729596061
40	466407	2144991	-0.6931472	-0.77855229	0.085405114
41	485303	2152711	1.2527630	0.56169490	0.691068070
42	491602	2149387	-1.3862944	-1.07646061	-0.309833747
43	488451	2146070	-0.6931472	-0.41000945	-0.283137730
44	471654	2147195	-1.3862944	0.21496174	-1.601256104
45	469543	2140559	0.0000000	-0.41345917	0.413459172
46	467433	2135030	-1.3862944	-1.23270845	-0.153585908
47	470588	2137238	-0.6931472	-0.67653129	-0.016615890
48	500000	2140533	-1.3862944	-1.26583321	-0.120461151
49	503149	2123937	-1.3862944	0.73600233	-2.122296689
50	472686	2135022	-1.3862944	-0.46695691	-0.919337450

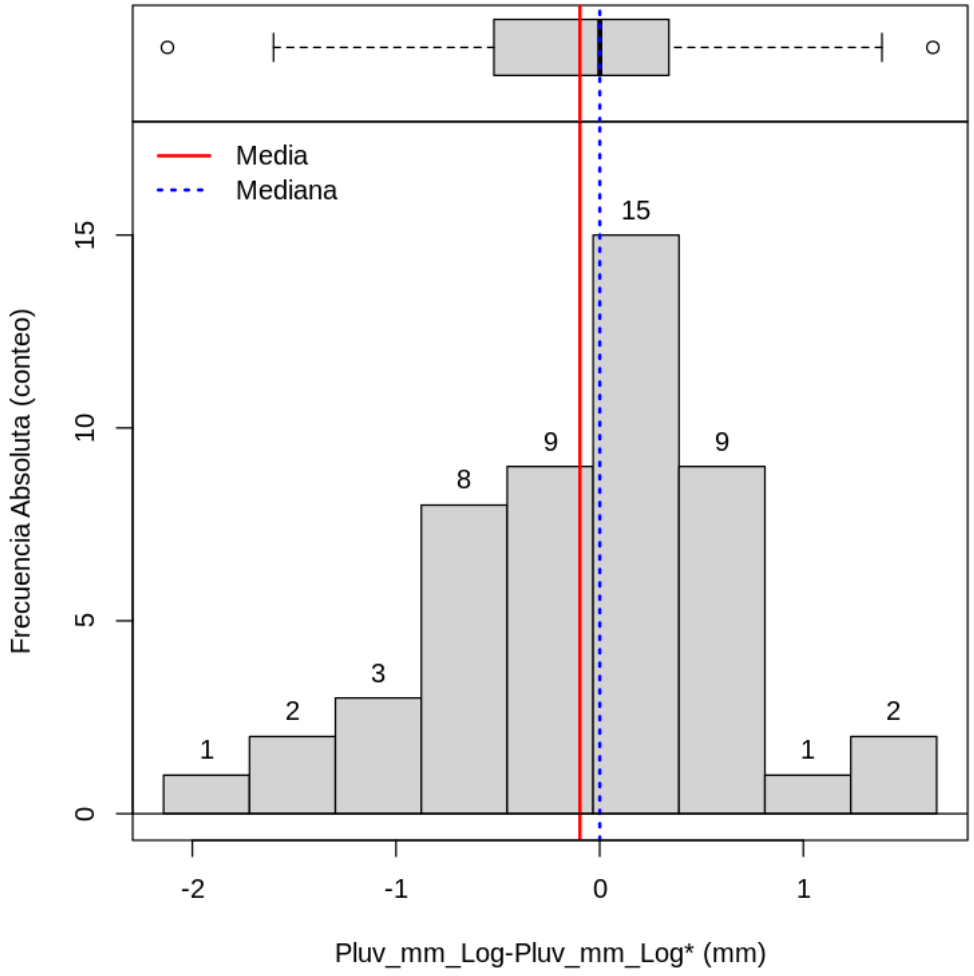
[215]:
```
Pluv_mm_Log_CrossValid_Stat<- Val_Estadisticos(Pluv_mm_Log_CrossValid[,c(3,4,5)])
Pluv_mm_Log_CrossValid_Stat
```

A data.frame: 13 × 3

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	50.00000	50.00000	50.00000
Minimo	-1.38629	-1.43082	-2.12230
Cuartil_1er	-1.21301	-0.45397	-0.51107
Mediana	0.00000	-0.08175	0.00058
Media	-0.09031	0.00660	-0.09691
Cuartil_3er	0.65976	0.66783	0.33242
Maximo	2.04769	1.35820	1.63607
Rango	3.43399	2.78902	3.75837
Rango_Intercuartil	1.87277	1.12180	0.84350
Varianza	1.00045	0.53289	0.51560
Desv_Estandar	1.00022	0.72999	0.71806
Simetria	0.11239	-0.25271	-0.30157
Curtosis	2.01197	2.07699	3.72457

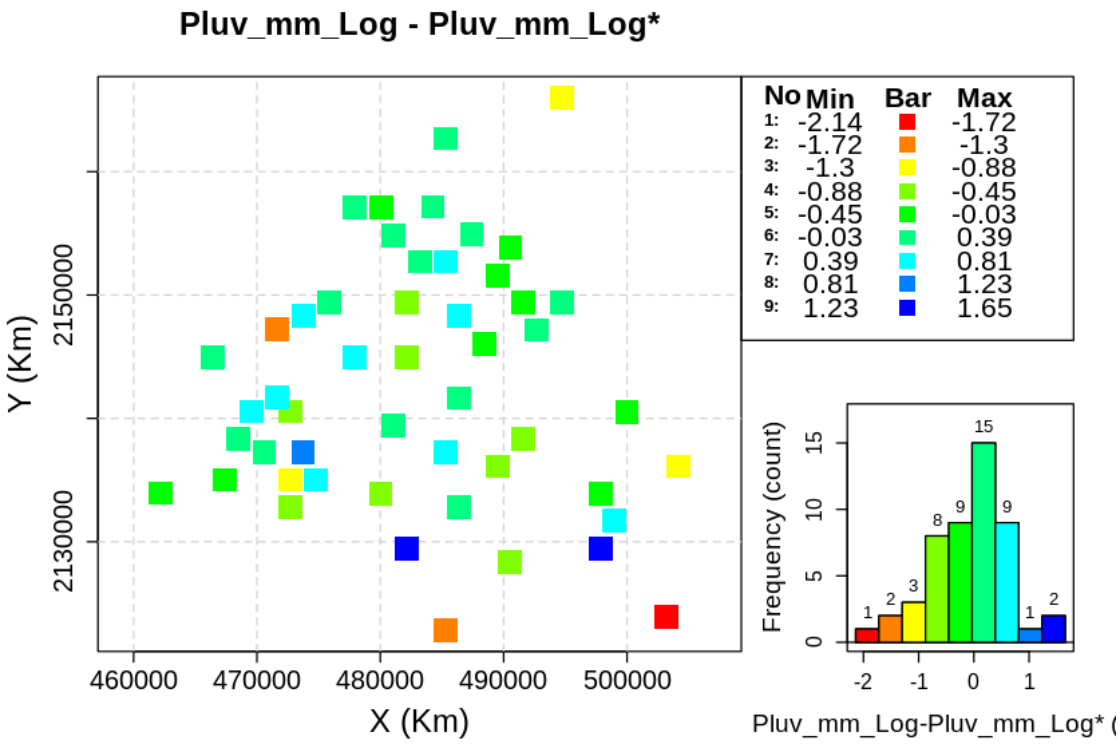
Respecto a los estadígrafos que obtenemos podemos notar que el valor esperado es cercano a cero, pero no lo suficiente para decidir si es un buen ajuste. Mientras que la varianza no es lo suficientemente pequeña.

[216]:
```
HistBoxplot(x=Pluv_mm_Log_CrossValid[,5], mean =␣
↪Pluv_mm_Log_CrossValid_Stat[5,3],
            median = Pluv_mm_Log_CrossValid_Stat[4,3], main ="",
            xlab = "Pluv_mm_Log-Pluv_mm_Log* (mm)", ylab = "Frecuencia Absoluta␣
↪(conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
```



Respecto a su histograma podemos notar que tiene valores atípicos en los extremos.

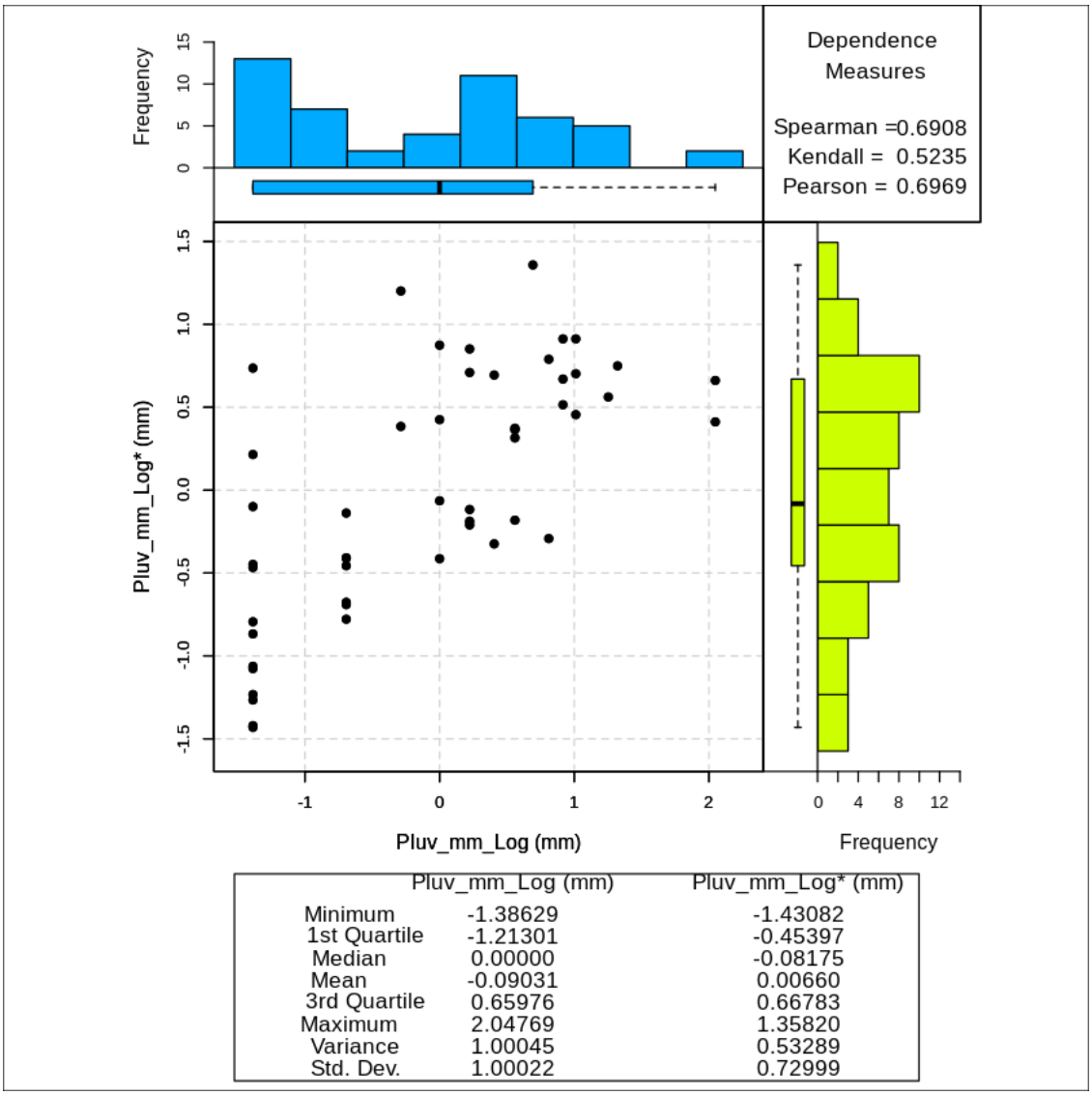
[217]:
```
DEspacial(Pluv_mm_Log_CrossValid[,1], Pluv_mm_Log_CrossValid[,2],␣
↪Pluv_mm_Log_CrossValid[,5],n_bins=9,
          'X (m)', 'Y (m)', 'Pluv_mm_Log-Pluv_mm_Log* (mm)', 'Pluv_mm_Log -␣
↪Pluv_mm_Log*')
```



Los valores atípicos mostrados en el grafico espacial coinciden con los valores atípicos localizados en el análisis de la variable radar, en especial los tres que están debajo de la cota 2130000

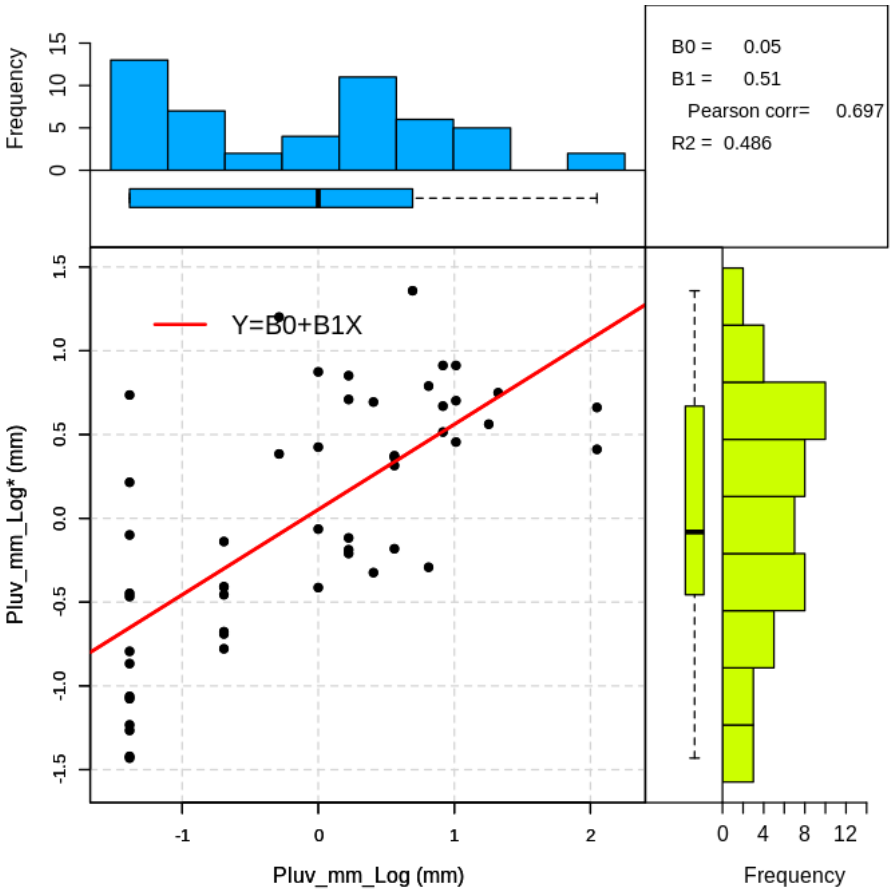
[218]:
```
# Pluv_mm_Log is the independent variable
X<-Pluv_mm_Log_CrossValid[,3] # the same as Pluv_mm_Log
# Pluv_mm_Log* is the dependent variable
Y<-Pluv_mm_Log_CrossValid[,4]
```

[219]:
```
ScatterPlot(Pluv_mm_Log, Pluv_mm_Log_CrossValid[,4], 9,
            Xmin = Pluv_mm_Log_Stat[2,2], Xmax = Pluv_mm_Log_Stat[7,2],
            Ymin = Pluv_mm_Log_CrossValid_Stat[2,2],Ymax =␣
↪Pluv_mm_Log_CrossValid_Stat[7,2],
            XLAB = "Pluv_mm_Log (mm)", YLAB = "Pluv_mm_Log* (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.

[220]:
```
scaterplotReg(Pluv_mm_Log, Pluv_mm_Log_CrossValid[,4], 9,
              Xmin = Pluv_mm_Log_Stat[2,2], Xmax = Pluv_mm_Log_Stat[7,2],
              Ymin = Pluv_mm_Log_CrossValid_Stat[2,2],Ymax =␣
↪Pluv_mm_Log_CrossValid_Stat[7,2],
              XLAB = "Pluv_mm_Log (mm)", YLAB = "Pluv_mm_Log* (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.

[221]:
```
dir.create(paste(getwd(),"/Results/EstimacionEspacial", sep=""))
```

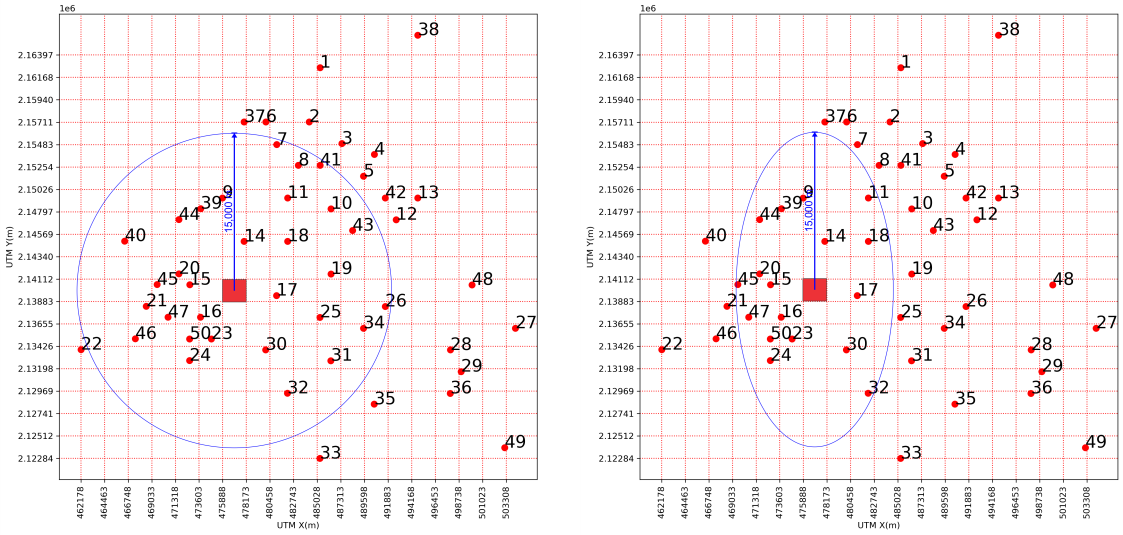
```
Warning message in dir.create(paste(getwd(), "/Results/EstimacionEspacial", sep
= "")):
“'/home/danielvr/Dropbox/Semestre 2023-1/ejemplo
lluvia/Results/EstimacionEspacial' already exists”
```

5.1 Estimación espacial usando kriging ordinario en la variable Radar_mm_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 variográma adireccional que consideramos isotrópico, por lo tanto, los valores que necesitamos del variograma son los siguientes:

[222]:
```
######## Radar_mm_Log
#modelos de variograma (1- exponential, 2- spherical, 3- gaussian)
Radar_mm_Log_vario_model<- 2
Radar_mm_Log_nugget<- 0.17
Radar_mm_Log_sill_and_nugget<- 0.8
Radar_mm_Log_rank <- 20000
```

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

[223]:
```
minPoints<-4
maxPoints<-10
```

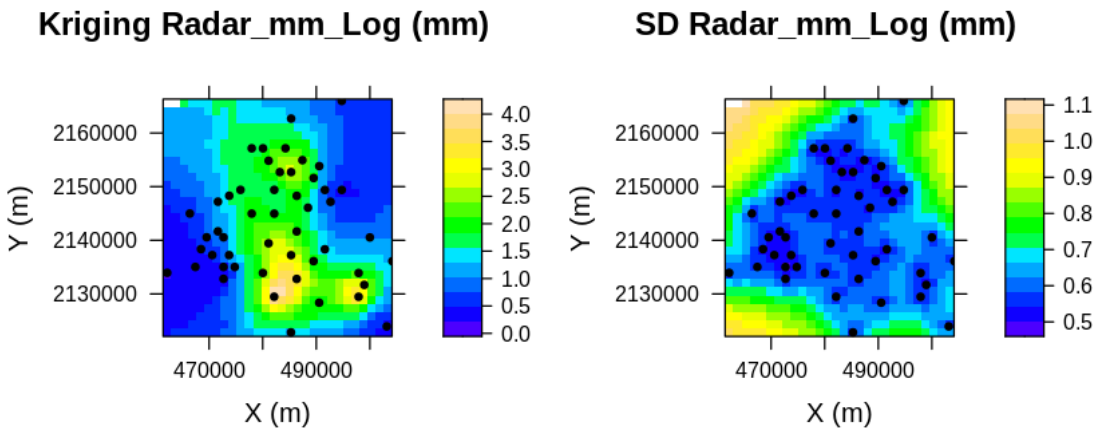
Ya que ingresamos los parámetros iniciales del variograma y número de puntos procedemos a hacer la estimación usando kriging ordinario. Esto lo hacemos usando la función "KrigingOrd". Esta función se estructura de la siguiente forma:

- Las coordenadas (CoorX, CoorY), la variable a usar (Prop1) que en este caso son las muestras obtenidas en el radar con transformación logarítmica (Radar_mm_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 (Radar_mm_Log_vario_model), valor del nugget (Radar_mm_Log_nugget), valor de la meseta más el nugget (Radar_mm_Log_sill_and_nugget) y el alcance (Radar_mm_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).

[224]:
```
Radar_mm_Log_OrdKrig <- KrigingOrd(CoorX = XCoord, CoorY = YCoord,
                                   Prop1 = Radar_mm_Log, Modelo =␣
↪Radar_mm_Log_vario_model, Nugget = Radar_mm_Log_nugget,
```

```
                              SillYNugget = Radar_mm_Log_sill_and_nugget, Alcance␣
↪= Radar_mm_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 (m)",
                              NameY="Y (m)", Titulo1="Kriging Radar_mm_Log (mm)",
                              Titulo2="SD Radar_mm_Log (mm)")
```

```
[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 "Radar_mm_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.

[225]:
```
Radar_mm_Log_OrdKrig
```

A matrix: 810 × 4 of type dbl

V1	V2	V3	V4
462178.0	2122836	0.3807191	1.0056316
463703.8	2122836	0.6513284	0.9727598
465229.6	2122836	0.6350795	0.9675436
466755.4	2122836	0.6090698	0.9611977
468281.2	2122836	0.6489124	0.9547559
469807.0	2122836	0.6067122	0.9299016
471332.8	2122836	0.6787958	0.9180130
472858.6	2122836	0.7732874	0.9032500
474384.4	2122836	0.9930957	0.8897243
475910.2	2122836	1.1150180	0.8653273
477436.0	2122836	1.2414374	0.8358788
478961.8	2122836	1.4306585	0.8014923
480487.6	2122836	1.4905495	0.7588172
482013.4	2122836	1.4725823	0.7077236
483539.2	2122836	1.4667607	0.6442298
485065.0	2122836	1.2927449	0.5609579
486590.8	2122836	1.4158241	0.6178495
488116.6	2122836	1.5299755	0.6758584
489642.4	2122836	1.2365387	0.7116253
491168.2	2122836	1.2005001	0.7382912
492694.0	2122836	1.1602312	0.7562419
494219.8	2122836	1.1023989	0.7652767
495745.6	2122836	1.0213370	0.7643058
497271.4	2122836	0.9155103	0.7530349
498797.2	2122836	0.7520732	0.7307808
500323.0	2122836	0.5836539	0.6962661
501848.8	2122836	0.4396870	0.6507814
503374.6	2122836	0.3499923	0.6293062
462178.0	2124362	0.3202458	0.9667039
463703.8	2124362	0.3143048	0.9572643
⋮	⋮	⋮	⋮
498797.2	2164033	0.6788844	0.7922850
500323.0	2164033	0.6899736	0.8469460
501848.8	2164033	0.6867658	0.8948266
503374.6	2164033	0.6914366	0.9362773
465229.6	2165558	1.6217380	1.0760143
466755.4	2165558	1.5444166	1.0336635
468281.2	2165558	1.5289204	1.0183501
469807.0	2165558	1.5172081	1.0007904
471332.8	2165558	1.5840866	0.9773470
472858.6	2165558	1.6058440	0.9583204
474384.4	2165558	1.5726149	0.9312199
475910.2	2165558	1.5407702	0.9062373
477436.0	2165558	1.5188026	0.8758481
478961.8	2165558	1.3556129	0.8374402
480487.6	2165558	1.2908461	0.8057845
482013.4	2165558	1.2686785	0.7701819
483539.2	2165558	1.2293526	0.7345463
485065.0	2165558	1.1755590	0.7091805
486590.8	2165558	1.1117305	0.7045957
488116.6	2165558	1.0401851	0.7103429
489642.4	2165558	0.9704935	0.7090575

Sí ejecutamos la siguiente línea solo nos muestra los primeros seis renglones de la tabla.

[226]:
```
head(Radar_mm_Log_OrdKrig)
```

A matrix: 6 × 4 of type dbl

V1	V2	V3	V4
462178.0	2122836	0.3807191	1.0056316
463703.8	2122836	0.6513284	0.9727598
465229.6	2122836	0.6350795	0.9675436
466755.4	2122836	0.6090698	0.9611977
468281.2	2122836	0.6489124	0.9547559
469807.0	2122836	0.6067122	0.9299016

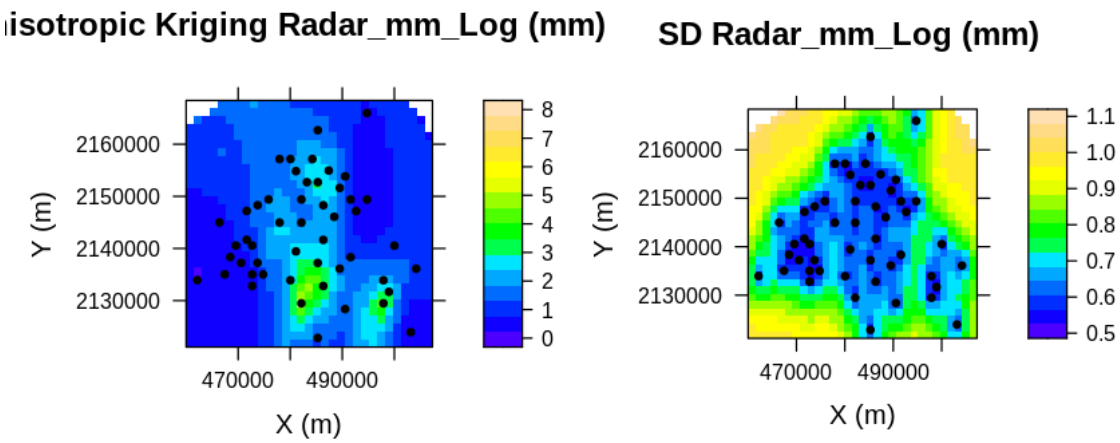
En caso de que usemos un variograma anisotrópico debemos usar la función "KrigingOrdAnis", esta función necesita los siguientes parámetros:

- Las coordenadas (CoorX, CoorY), la variable a usar (Prop1) que en este caso son las muestras obtenidas en los pluviómetros con transformación logarítmica (Radar_mm_Log)
- La información del variograma, que en este caso es el modelo (Radar_mm_Log_vario_model), valor del nugget (Radar_mm_Log_nugget), valor de la meseta más el nugget (Radar_mm_Log_sill_and_nugget) y el alcance (Radar_mm_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).

[227]:
```
Radar_mm_Log_OrdKrigAnis <- KrigingOrdAnis(CoorX = XCoord, CoorY = YCoord,
                Prop1 = Radar_mm_Log, Modelo = Radar_mm_Log_vario_model, Nugget =␣
↪Radar_mm_Log_nugget,
                SillYNugget = Radar_mm_Log_sill_and_nugget, Alcance =␣
↪Radar_mm_Log_rank,
                minPar = minPoints, maxPar = maxPoints,
                malla = DistMin, InvT = 1, MaxAnis = 0, proporcion = 0.5,
                NameX="X (m)",
                NameY="Y (m)", Titulo1="Anisotropic Kriging Radar_mm_Log (mm)",
                Titulo2="SD Radar_mm_Log (mm)")
```

```
[using ordinary kriging]
```

```
Warning message in sqrt(kriging$var1.var):
"NaNs produced"
```



5.2 Estimación espacial usando kriging ordinario en la variable Pluv_mm_Log

Al igual que la variable radar, para la estimación espacial usando kriging ordinario para la variable Pluv_mm_Log 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:

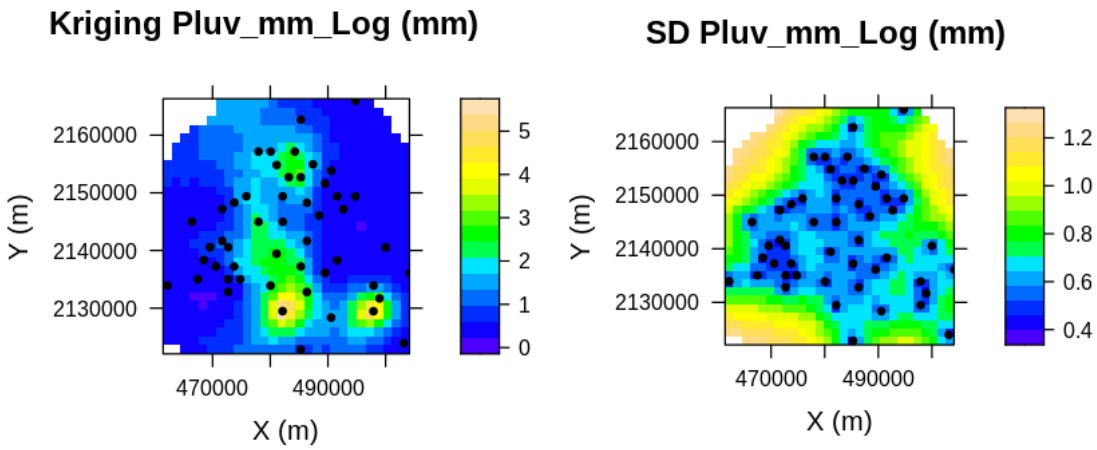
[228]:
```
Pluv_mm_Log_vario_model<- 2
Pluv_mm_Log_nugget<- 0.08
Pluv_mm_Log_sill_and_nugget<- 1.1
```

```
Pluv_mm_Log_rank <- 15000
```

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

[229]:
```
Pluv_mm_Log_OrdKrig <- KrigingOrd(CoorX = XCoord, CoorY = YCoord,
                                  Prop1 = Pluv_mm_Log, Modelo =␣
↪Pluv_mm_Log_vario_model, Nugget = Pluv_mm_Log_nugget,
                                  SillYNugget = Pluv_mm_Log_sill_and_nugget,␣
↪Alcance = Pluv_mm_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 (m)",
                                  NameY="Y (m)", Titulo1="Kriging Pluv_mm_Log␣
↪(mm)",
                                  Titulo2="SD Pluv_mm_Log (mm)")
```

```
[using ordinary kriging]
```



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

[230]:
```
head(Pluv_mm_Log_OrdKrig)
```

A matrix: 6 × 4 of type dbl

V1	V2	V3	V4
465229.6	2122836	0.2500000	1.249798
466755.4	2122836	0.3720087	1.234793
468281.2	2122836	0.3604417	1.233255
469807.0	2122836	0.8629939	1.182966
471332.8	2122836	0.6924827	1.150607
472858.6	2122836	0.8631012	1.151633

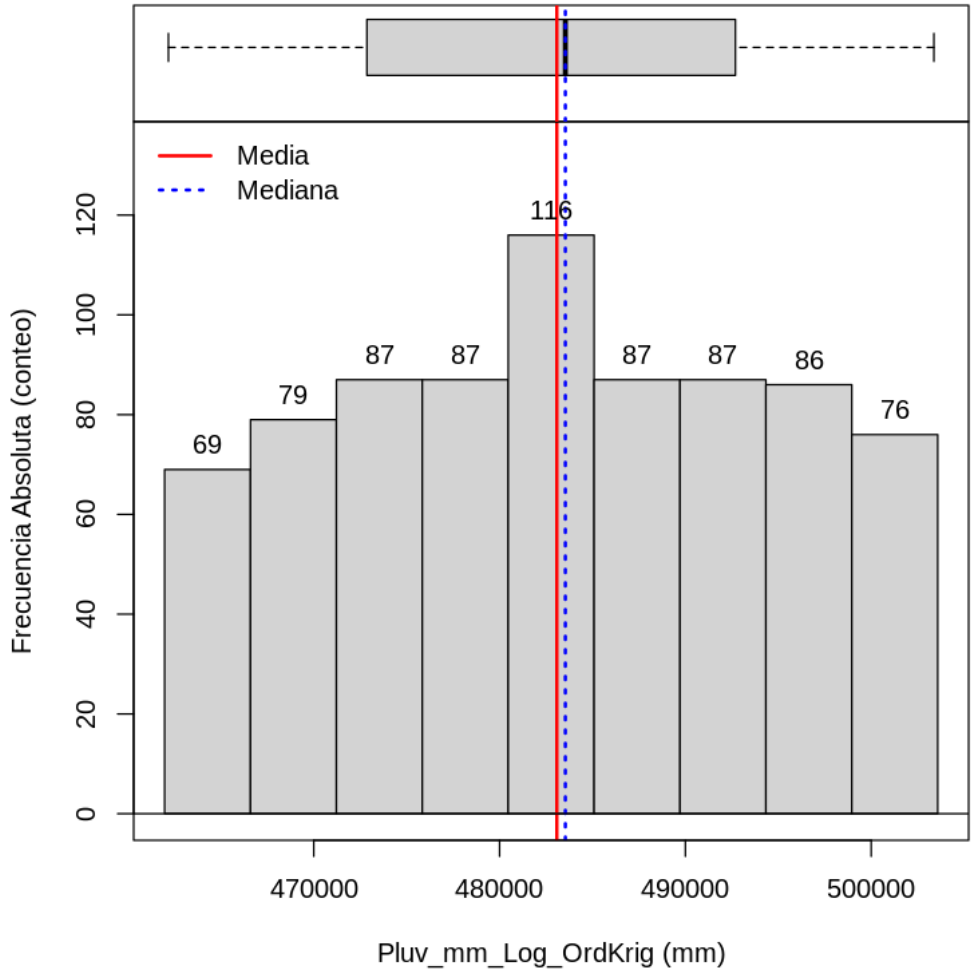
Calculamos los estadígrafos del kriging ordinario.

[231]:
```
Pluv_mm_Log_OrdKrig_stat<-Estadisticas(Pluv_mm_Log_OrdKrig[,1])
Pluv_mm_Log_OrdKrig_stat
```

A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	7.740000e+02
minimos	Minimum	4.621780e+05
cuantiles1	1st. Quartile	4.728586e+05
medianas	Median	4.835392e+05
medias	Mean	4.830819e+05
cuantiles3	3rd. Quartile	4.926940e+05
maximos	Maximum	5.033746e+05
rangos	Rank	4.119659e+04
rangosInt	Interquartile Rank	1.983540e+04
varianzas	Variance	1.426669e+08
desvs	Standard Deviation	1.194432e+04
CVs	Variation Coeff.	2.470000e-02
simetrias	Skewness	-2.160000e-02
curtosiss	Kurtosis	1.842900e+00

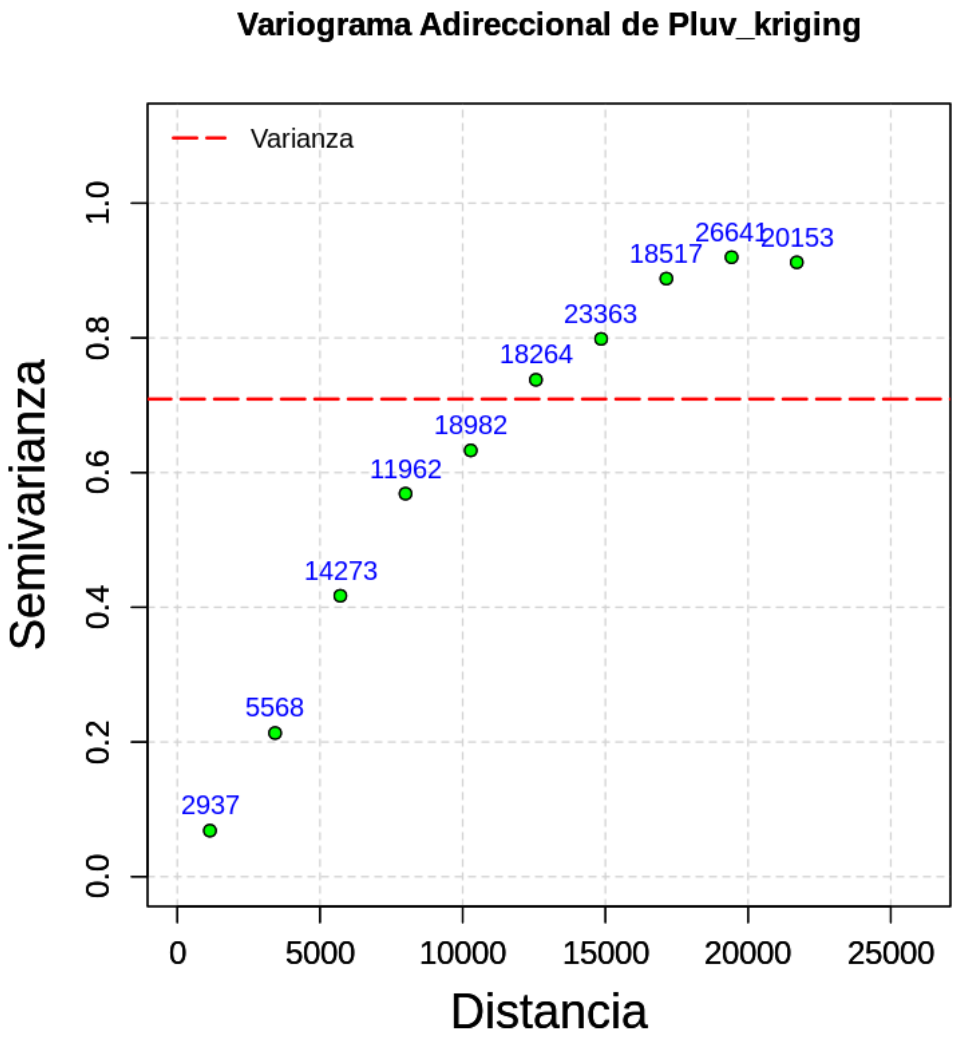
[232]:
```
HistBoxplot(x=Pluv_mm_Log_OrdKrig[,1], mean = Pluv_mm_Log_OrdKrig_stat[5,2],␣
↪median = Pluv_mm_Log_OrdKrig_stat[4,2], main ="",
            xlab = "Pluv_mm_Log_OrdKrig (mm)", ylab = "Frecuencia Absoluta␣
↪(conteo)", AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



También podemos obtener su variograma y evaluar cuanto se ajusta el resultado de la estimación usando kriging

[233]:
```
Pluv_ordikrig_1_Log_VarioEstimation<-Variograma(Pluv_mm_Log_OrdKrig[,1],␣
↪Pluv_mm_Log_OrdKrig[,2],
                                                Pluv_mm_Log_OrdKrig[,3], 0, 90,␣
↪1*N_lags, lag_value, 1, "Variograma Adireccional de Pluv_kriging")
```

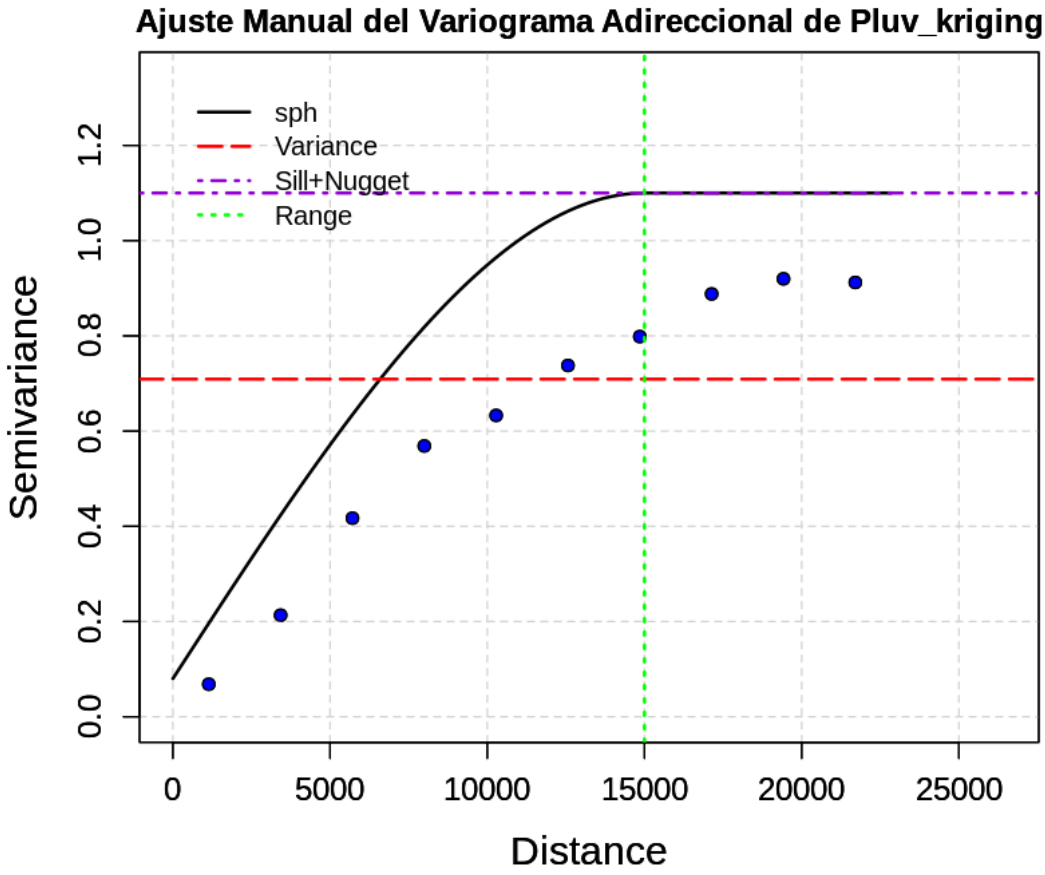
```
variog: computing omnidirectional variogram
```



[234]:
```
Pluv_mm_Log_vario_model<- 2
Pluv_mm_Log_nugget<- 0.08
Pluv_mm_Log_sill_and_nugget<- 1.1
Pluv_mm_Log_rank <- 15000

Pluv_SGS_1_EyeModelVarioFit<-EyeModel(Pluv_mm_Log_OrdKrig[,1],␣
↪Pluv_mm_Log_OrdKrig[,2],
                                      Pluv_mm_Log_OrdKrig[,3], 0, 90, N_lags,␣
↪lag_value, 1,
                                      Pluv_mm_Log_vario_model,␣
↪Pluv_mm_Log_nugget, Pluv_mm_Log_sill_and_nugget, Pluv_mm_Log_rank,
```

```
                                                    "Ajuste Manual del Variograma Adireccional␣
↪de Pluv_kriging")
```

```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0800	1.1000	15000.0000	0.5887

5.3 Estimación espacial usando Co-kriging ordinario

5.3.1 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 corregionalización 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 corregionalización 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 (radar y pluviómetros) 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)

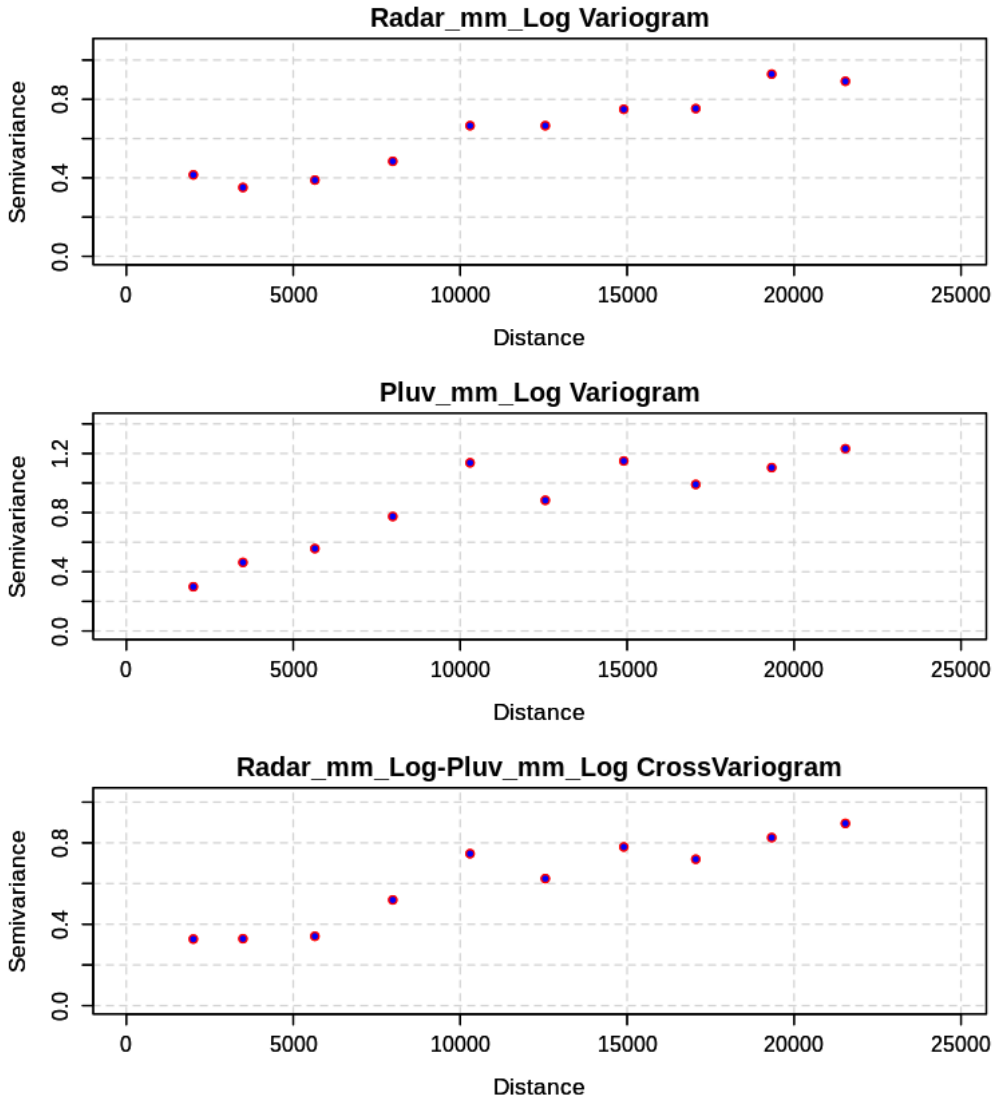
[235]:
```
N_lags<-10
DistMin<-min(dist(Data_File[,1:2])) # Minimum distance in data
DistMax<-max(dist(Data_File[,1:2])) # Maximum distance in data
lag_value<-max((DistMax/2)/N_lags, DistMin)
```

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 en los pluviómetros con transformación logarítmica (Pluv_mm_Log) y la imagen del Radar con transformación logarítmica (Radar_mm_Log)
- 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 0^{o} y su tolerancia es de 90^{o}.
- 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 corregionalización dará valores negativos

[236]:
```
Radar_mm_Log_Pluv_mm_Log_CrossVario<-CrossVariograma(CoorX = XCoord, CoorY =␣
↪YCoord,
                                                     P1 = Radar_mm_Log, P2 = Pluv_mm_Log,
                                                     NInt = N_lags, lags = lag_value,
                                                     Direccion = 0, Tol = 90,
                                                     NomP1 = 'Radar_mm_Log Variogram',
                                                     NomP2 = 'Pluv_mm_Log Variogram',
                                                     NomP1P2 = 'Radar_mm_Log-Pluv_mm_Log␣
↪CrossVariogram')
```



Esto nos da como resultado una imagen donde podemos ver tres variogramas: el de la variable Radar_mm_Log, variable Pluv_mm_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 Pluv_mm_Log.

[237]:
```
# Radar_mm_Log
Radar_mm_Log_vario_model<- 2
Radar_mm_Log_nugget<- 0.17
Radar_mm_Log_sill_minus_nugget<- 0.63
Radar_mm_Log_rank <- 15000
```

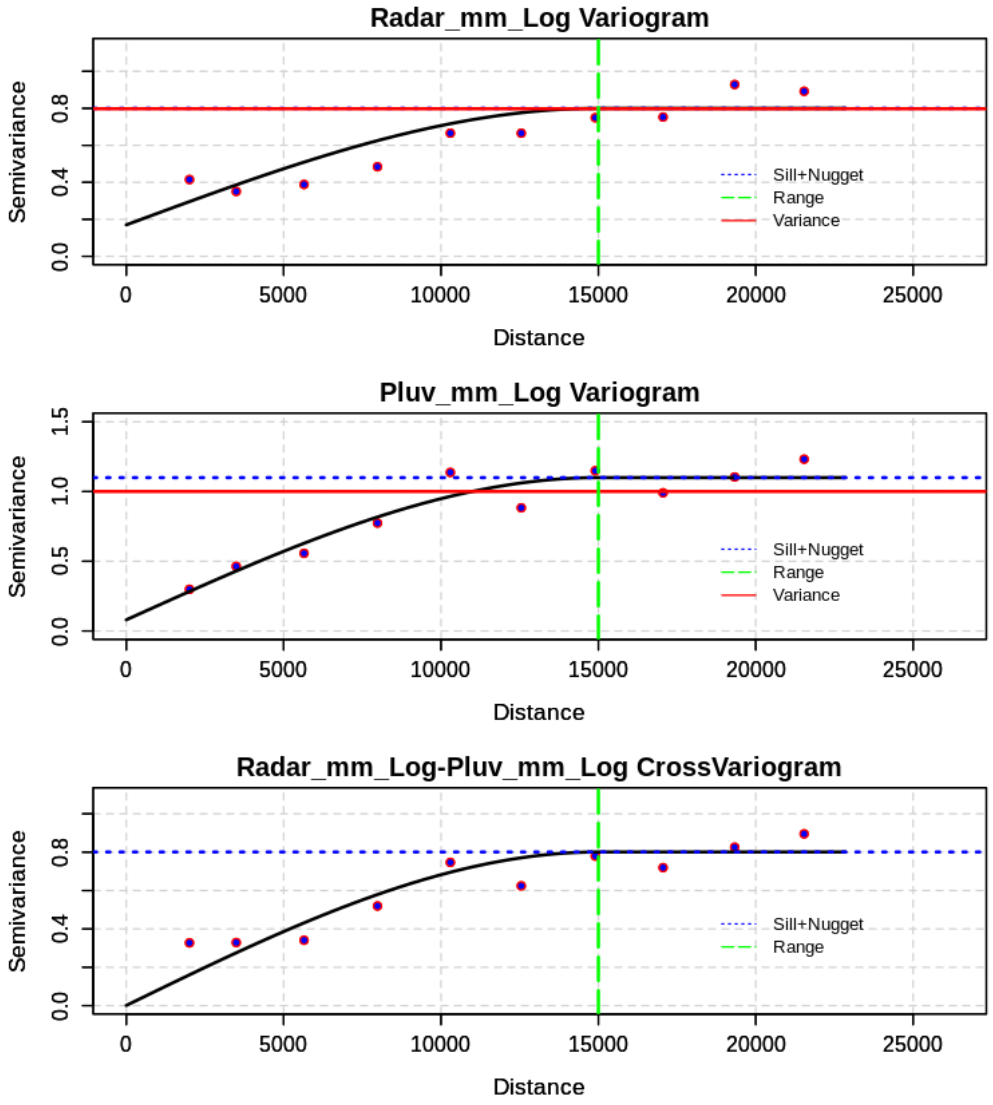
[238]:
```
# Pluv_mm_Log
Pluv_mm_Log_vario_model<- 2
Pluv_mm_Log_nugget<- 0.08
Pluv_mm_Log_sill_minus_nugget<- 1.02
Pluv_mm_Log_rank <- 15000
```

[239]:
```
# Cross-variogram
Radar_mm_Log_Pluv_mm_Log_vario_model<- 2 # spherical
Radar_mm_Log_Pluv_mm_Log_nugget<- 0.001
Radar_mm_Log_Pluv_mm_Log_sill_minus_nugget<- 0.8
Radar_mm_Log_Pluv_mm_Log_rank <- 15000
```

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 (Radar_mm_Log, Pluv_mm_Log) que en este caso son las muestras obtenidas en los pluviómetros con transformación logarítmica (Pluv_mm_Log) y la imagen del Radar con transformación logarítmica (Radar_mm_Log)
- 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 0^{o} y su tolerancia es de 90^{o}.
- Los valores de cada variograma que colocamos en las lineas anteriores (Radar_mm_Log_Pluv_mm_Log_vario_model, Radar_mm_Log_sill_minus_nugget, Pluv_mm_Log_sill_minus_nugget, Radar_mm_Log_Pluv_mm_Log_sill_minus_nugget, Radar_mm_Log_nugget, Pluv_mm_Log_nugget, Radar_mm_Log_Pluv_mm_Log_nugget, Radar_mm_Log_Pluv_mm_Log_rank)
- Títulos para cada variograma ('Radar_mm_Log Variogram', 'Pluv_mm_Log Variogram', 'Radar_mm_Log-Pluv_mm_Log CrossVariogram')

[240]:
```
ModelVariogram(XCoord, YCoord,
               Radar_mm_Log, Pluv_mm_Log,
               N_lags, lag_value, 0, 90,
               Radar_mm_Log_Pluv_mm_Log_vario_model,
               Radar_mm_Log_sill_minus_nugget, Pluv_mm_Log_sill_minus_nugget,
               Radar_mm_Log_Pluv_mm_Log_sill_minus_nugget, Radar_mm_Log_nugget,␣
↪Pluv_mm_Log_nugget,
               Radar_mm_Log_Pluv_mm_Log_nugget, Radar_mm_Log_Pluv_mm_Log_rank,
               'Radar_mm_Log Variogram', 'Pluv_mm_Log Variogram',␣
↪'Radar_mm_Log-Pluv_mm_Log CrossVariogram')
```



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

$$\begin{pmatrix} \gamma_{PP}(\underline{h}) & \gamma_{PR}(\underline{h}) \\ \gamma_{RP}(\underline{h}) & \gamma_{RR}(\underline{h}) \end{pmatrix} = \begin{pmatrix} 0.08 & 0.001 \\ 0.001 & 0.17 \end{pmatrix} \gamma_0(\underline{h}) + \begin{pmatrix} 1.02 & 0.8 \\ 0.8 & 0.63 \end{pmatrix} \gamma_1(\underline{h})$$

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

[241]:
```
NugetMatrix <- matrix(c(Pluv_mm_Log_nugget,
                        Radar_mm_Log_Pluv_mm_Log_nugget,
                        Radar_mm_Log_Pluv_mm_Log_nugget,
                        Radar_mm_Log_nugget), ncol = 2)
```

```
det(NugetMatrix)
```

0.013599

$$det\begin{pmatrix}0.08 & 0.001\\ 0.001 & 0.17\end{pmatrix} = 0.013599 > 0$$

el determinante es positivo

[242]:
```
SemiVariMatrix1 <- matrix(c(Pluv_mm_Log_sill_minus_nugget,
                      Radar_mm_Log_Pluv_mm_Log_sill_minus_nugget,
                      Radar_mm_Log_Pluv_mm_Log_sill_minus_nugget,
                      Radar_mm_Log_sill_minus_nugget), ncol = 2)
det(SemiVariMatrix1)
```

0.00260000000000001

$$det\begin{pmatrix}1.02 & 0.8\\ 0.8 & 0.63\end{pmatrix} = 0.0026 > 0$$

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 (Radar_mm_Log, Pluv_mm_Log) que en este caso son las muestras obtenidas en los pluviómetros con transformación logarítmica (Pluv_mm_Log) y la imagen del Radar con transformación logarítmica (Radar_mm_Log)
- 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.

[243]:
```
Pluv_mm_Log_CrossValid2<- CrossValidation2(XCoord, YCoord,
                                           Pluv_mm_Log, Radar_mm_Log,
                                           N_lags, lag_value,␣
↪Radar_mm_Log_Pluv_mm_Log_vario_model,
                                           Pluv_mm_Log_sill_minus_nugget,␣
↪Radar_mm_Log_sill_minus_nugget,
                                          ␣
↪Radar_mm_Log_Pluv_mm_Log_sill_minus_nugget, Pluv_mm_Log_nugget,
                                           Radar_mm_Log_nugget,␣
↪Radar_mm_Log_Pluv_mm_Log_nugget,
                                           Radar_mm_Log_Pluv_mm_Log_rank)
```

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

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

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

[244]:
```
head(Pluv_mm_Log_CrossValid2)
```

A data.frame: 6 × 5

	X <dbl>	Y <dbl>	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
1	485303	2162682	0.0000000	-0.09802348	0.09802348
2	484253	2157150	0.9162907	0.67624411	0.24004662
3	487403	2154937	0.5596158	0.64957678	-0.08996099
4	490552	2153830	-0.6931472	-0.31209350	-0.38105368
5	489502	2151601	-0.6931472	0.29077563	-0.98392281
6	480054	2157142	0.4054651	0.42329067	-0.01782556

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"

[245]:
```
Pluv_mm_Log_CrossValid2_Sta <-␣
↪Val_Estadisticos(Pluv_mm_Log_CrossValid2[,c(3,4,5)])
Pluv_mm_Log_CrossValid2_Sta
```

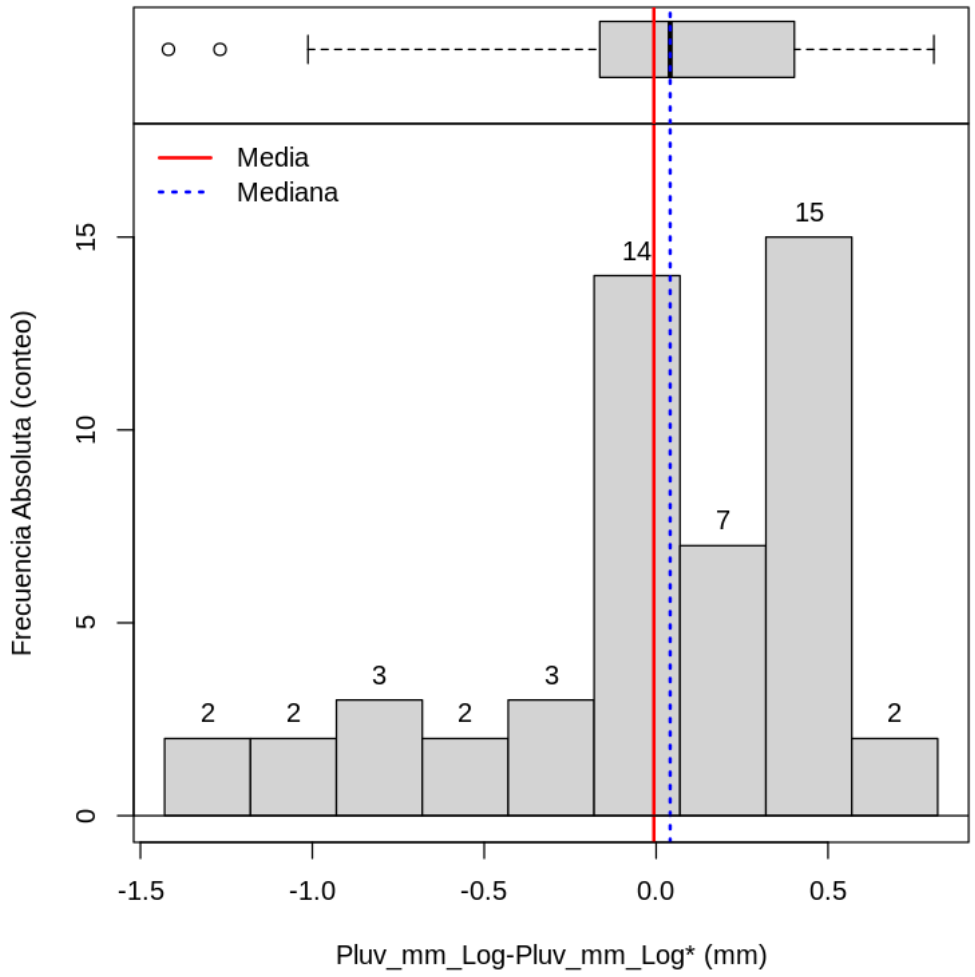
A data.frame: 13 × 3

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	50.00000	50.00000	50.00000
Minimo	-1.38629	-2.19414	-1.41856
Cuartil_1er	-1.21301	-0.56034	-0.14604
Mediana	0.00000	0.14359	0.04081
Media	-0.09031	-0.08377	-0.00654
Cuartil_3er	0.65976	0.59301	0.39143
Maximo	2.04769	1.56279	0.80785
Rango	3.43399	3.75693	2.22641
Rango_Intercuartil	1.87277	1.15334	0.53747
Varianza	1.00045	0.80625	0.26143
Desv_Estandar	1.00022	0.89792	0.51130
Simetria	0.11239	-0.50851	-1.00479
Curtosis	2.01197	2.65372	3.42112

También graficamos su histograma

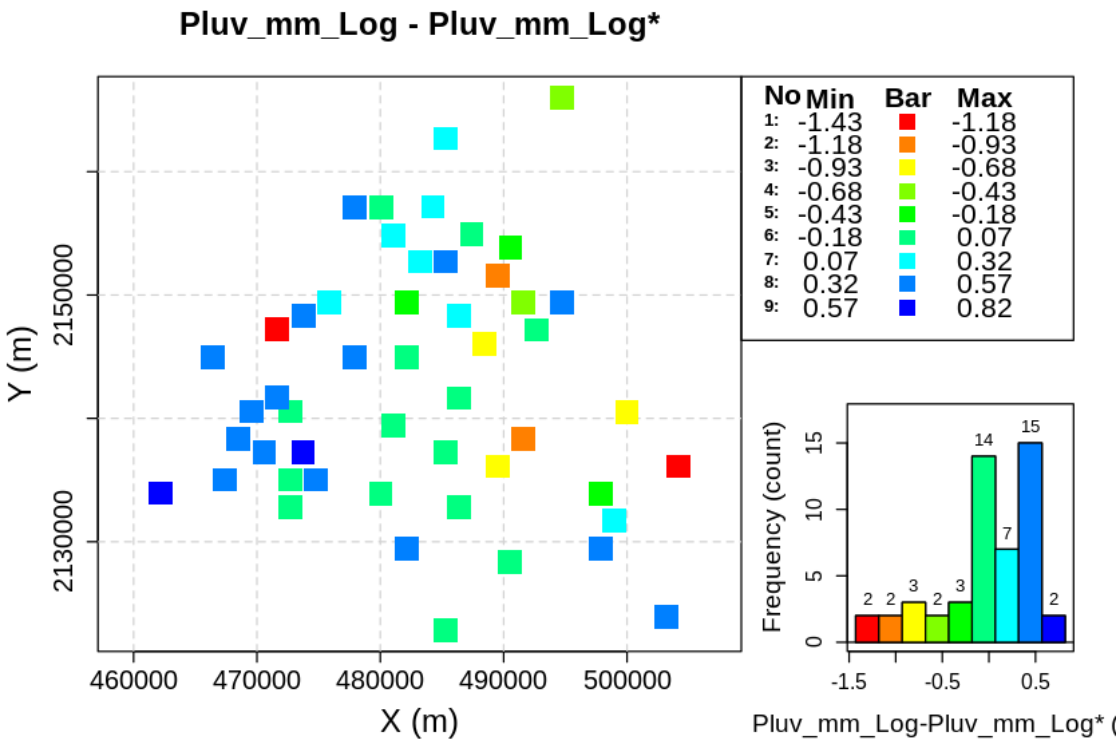
[246]:
```
HistBoxplot(x=Pluv_mm_Log_CrossValid2[,5], mean =␣
↪Pluv_mm_Log_CrossValid2_Sta[5,3],
            median = Pluv_mm_Log_CrossValid2_Sta[4,3],
            main ="", xlab = "Pluv_mm_Log-Pluv_mm_Log* (mm)", ylab = "Frecuencia␣
↪Absoluta (conteo)",
```

```
            AbsFreq = TRUE, PercentFreq = FALSE )
```



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

[247]:
```
DEspacial(Pluv_mm_Log_CrossValid2[,1], Pluv_mm_Log_CrossValid2[,2],␣
↪Pluv_mm_Log_CrossValid2[,5],n_bins=9,
          'X (m)', 'Y (m)', 'Pluv_mm_Log-Pluv_mm_Log* (mm)', 'Pluv_mm_Log -␣
↪Pluv_mm_Log*')
```



5.3.2 Estimación usando co-kriging ordinario para la variable Pluv_mm_Log

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

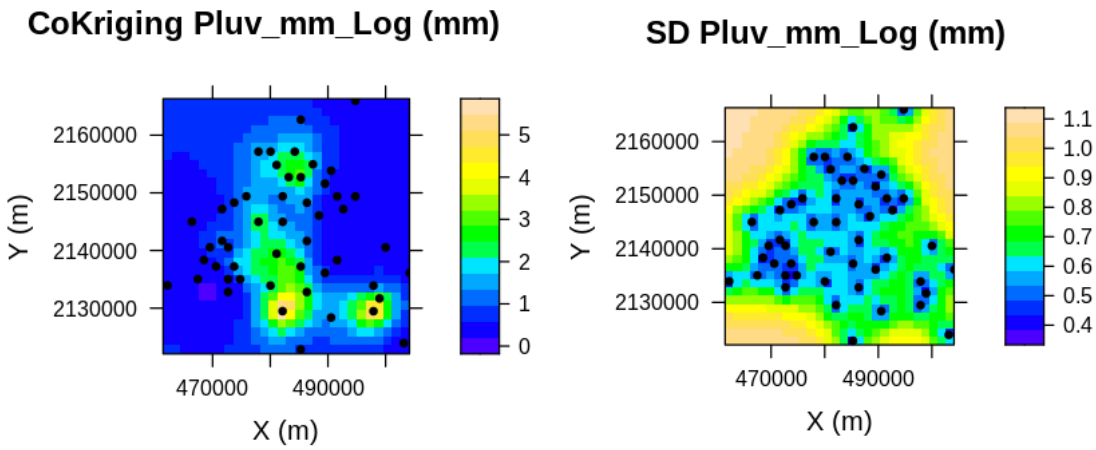
- Las coordenadas (CoorX, CoorY), las variables a usar (Radar_mm_Log, Pluv_mm_Log) que en este caso son las muestras obtenidas en los pluviómetros con transformación logarítmica (Pluv_mm_Log) y la imagen del Radar con transformación logarítmica (Radar_mm_Log)
- La información de los variogramas simples 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).

[248]:
```
Pluv_mm_Log_CoKrig <- CoKrigingOrd(XCoord, YCoord,
                                   Pluv_mm_Log, Radar_mm_Log,
                                   Radar_mm_Log_Pluv_mm_Log_vario_model,
                                   Pluv_mm_Log_sill_minus_nugget,␣
 ↪Radar_mm_Log_sill_minus_nugget,
                                   Radar_mm_Log_Pluv_mm_Log_sill_minus_nugget,
                                   Pluv_mm_Log_nugget, Radar_mm_Log_nugget,
                                   Radar_mm_Log_Pluv_mm_Log_nugget,␣
 ↪Radar_mm_Log_Pluv_mm_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 (m)",
                                   NameY="Y (m)", Titulo1="CoKriging Pluv_mm_Log␣
 ↪(mm)",
                                   Titulo2="SD Pluv_mm_Log (mm)")
```

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



6 Simulación secuencial Gaussiana condicional

Ya que exploramos las ventajas y desventajas del kriging y cokriging, podemos comparar sus resultados con otros métodos, en este caso la simulación. Las diferencias que hay entre los métodos de estimación (kriging) y la simulación son:

Estimaciones

- Dependen fuertemente número de puntos y de su distribución espacial
- No requieren de mucho esfuerzo de computo

Simulaciones

- No dependen tan fuertemente del número de puntos y de su distribución espacial

- Son más demandantes computacionalmente

El tipo de simulación que se presentará es la simulación secuencial Gaussiana condicional, esta puede ser perfeccionada agregándole todo una suerte de información cualitativa disponible del fenómeno real. Por ejemplo, si usamos los datos de los pluviómetros entonces podemos disponer del variograma y su distribución espacial.

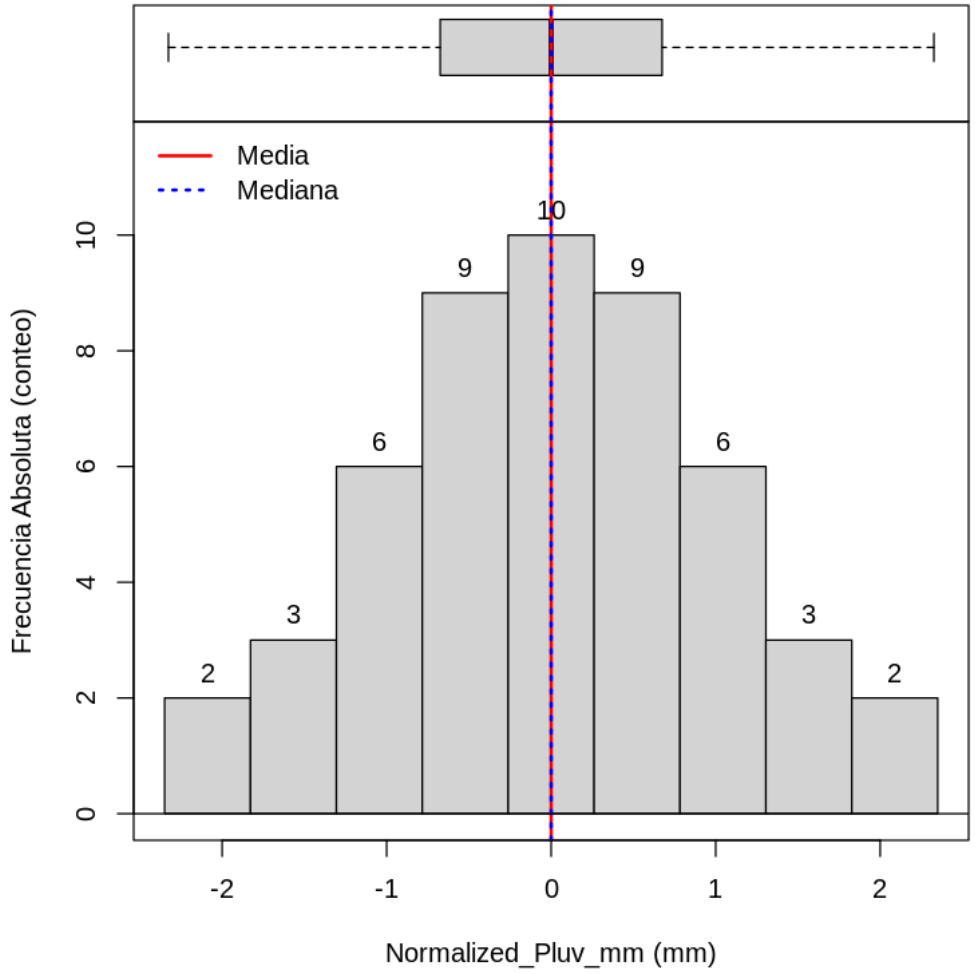
Para usar este tipo de simulación necesitamos transformar la variable que nos interesa simular, en este caso es "Pluv_mm", esto lo hacemos usando la función "nscore". Al vector resultante le calculamos sus estadígrafos y su histograma.

[249]:
```
Normalized_Pluv <- nscore(Pluv_mm)
Normalized_Pluv_Stat <- Estadisticas(Normalized_Pluv$nscore)
Normalized_Pluv_Stat
```

A data.frame: 14 × 2

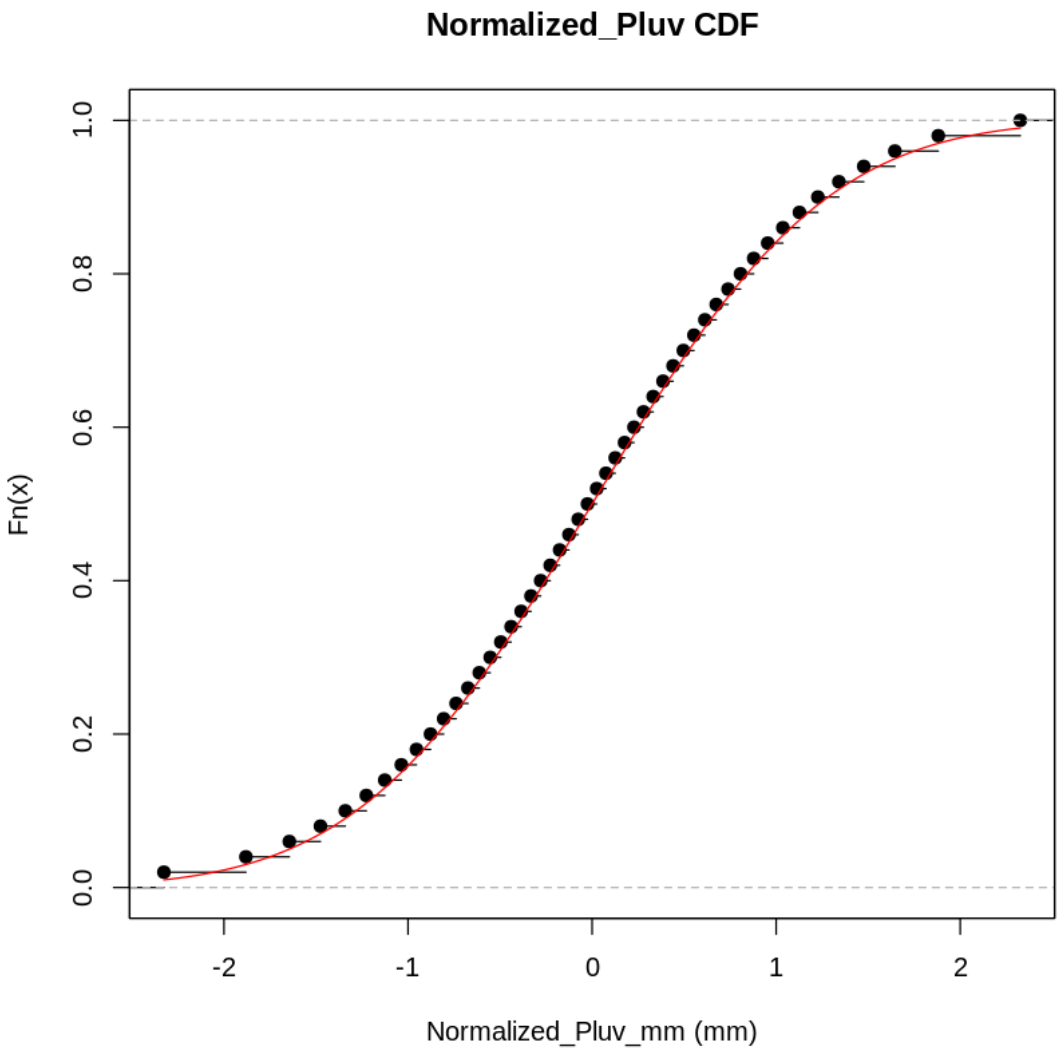
	Statistics <chr>	Values <dbl>
muestras	n	5.000000e+01
minimos	Minimum	-2.326300e+00
cuantiles1	1st. Quartile	-6.591000e-01
medianas	Median	0.000000e+00
medias	Mean	0.000000e+00
cuantiles3	3rd. Quartile	6.591000e-01
maximos	Maximum	2.326300e+00
rangos	Rank	4.652700e+00
rangosInt	Interquartile Rank	1.318100e+00
varianzas	Variance	9.948000e-01
desvs	Standard Deviation	9.974000e-01
CVs	Variation Coeff.	-5.620351e+16
simetrias	Skewness	0.000000e+00
curtosiss	Kurtosis	2.729300e+00

[250]:
```
HistBoxplot(x=Normalized_Pluv$nscore, mean = Normalized_Pluv_Stat[5,2], median =␣
↪Normalized_Pluv_Stat[4,2], main ="",
            xlab = "Normalized_Pluv_mm (mm)", ylab = "Frecuencia Absoluta␣
↪(conteo)", AbsFreq = TRUE, PercentFreq = FALSE,
            nbin = 9)
```



También graficamos la función de distribución acumulativa de las observaciones normalizadas.

[251]:
```
plot(ecdf(Normalized_Pluv$nscore), main="Normalized_Pluv CDF",xlab =␣
 ↪"Normalized_Pluv_mm (mm)",
     xlim=c(Normalized_Pluv_Stat[2,2], Normalized_Pluv_Stat[7,2]), ylim=c(0,1))
par(new=TRUE)
curve(pnorm, from = Normalized_Pluv_Stat[2,2], to = Normalized_Pluv_Stat[7,2],
      ylim=c(0,1),col=("red"),xaxt='n',yaxt='n',ann=FALSE)
```



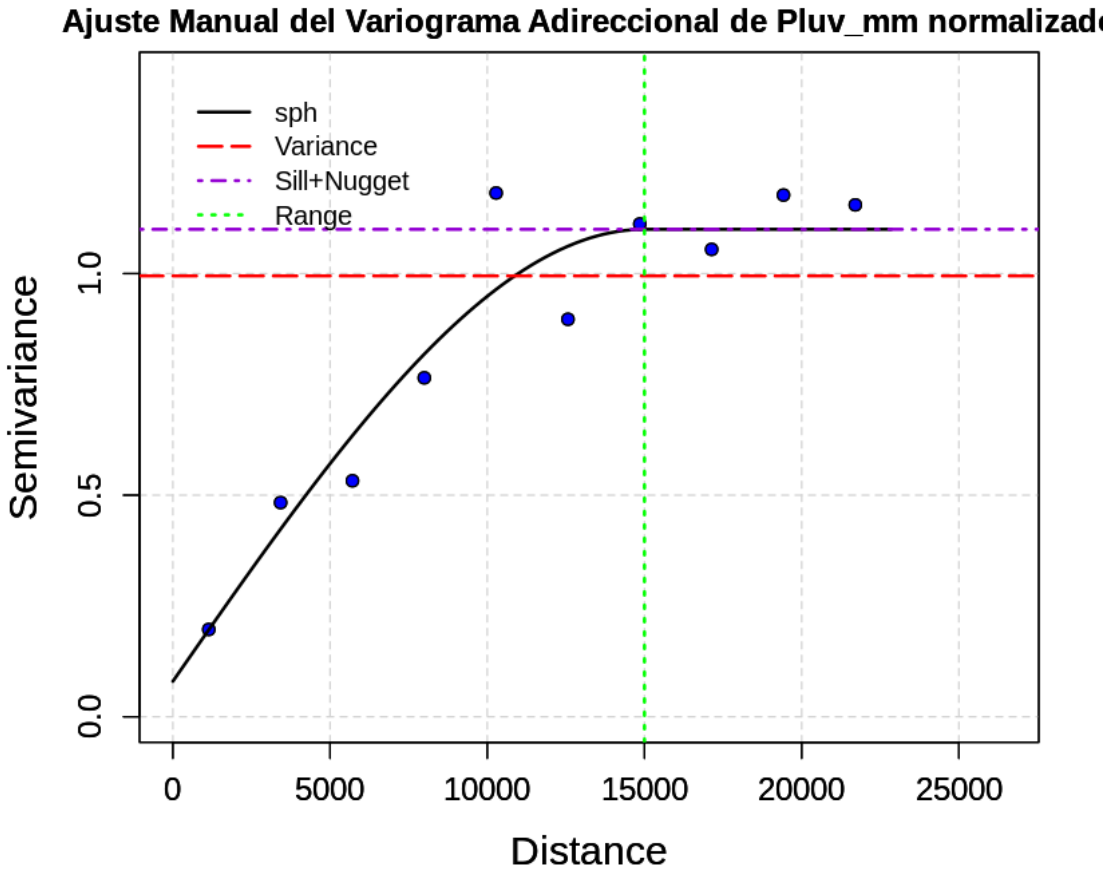
Despues debemos hacer el análisis variográfico para saber cuanto cambia el modelo de variograma propuesto.

[252]:
```
N_lags <- 10
lag_value <- ((DistMax/2)/N_lags)
Normalized_Pluv_vario_model<- 2
Normalized_Pluv_nugget<- 0.08
Normalized_Pluv_sill_and_nugget<- 1.1
Normalized_Pluv_rank <- 15000
```

[253]:
```
Normalized_Pluv_EyeModelVarioFit<-EyeModel(XCoord, YCoord,
                                           Normalized_Pluv$nscore, 0, 90,␣
↪N_lags, lag_value, 1,
```

```
                                                    Normalized_Pluv_vario_model,␣
↪Normalized_Pluv_nugget,
                                                    Normalized_Pluv_sill_and_nugget,␣
↪Normalized_Pluv_rank,
                                                    "Ajuste Manual del Variograma␣
↪Adireccional de Pluv_mm normalizado")
```

```
variog: computing omnidirectional variogram
```



Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0800	1.1000	15000.0000	0.1025

Si comparamos el modelo de variograma obtenido con las muestras con transformación logarítmica y el modelo de variograma con las muestras normalizadas no tienen gran diferencia. El error del primero es de 0.1135 y el segundo es de 0.1025, ambos casos tienen los mismos valores en sus parámetros. Para confirmar si el modelo es bueno hay que realizar la validación cruzada.

[254]:
```
Normalized_Pluv_mm_CrossValid<- CrossValidation(XCoord, YCoord,
                                        Normalized_Pluv$nscore,␣
 ↪Normalized_Pluv_vario_model,
                                        Normalized_Pluv_nugget,␣
 ↪Normalized_Pluv_sill_and_nugget,
                                        Normalized_Pluv_rank, MaxAnis=0,␣
 ↪proporcion=1)
Normalized_Pluv_mm_CrossValid
```

A data.frame: 50 × 5

	X	Y	Z	Z*	Z-Z*
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	485303	2162682	-0.12566135	-0.036030698	-0.089630649
2	484253	2157150	0.87789630	0.614404802	0.263491493
3	487403	2154937	0.43991317	0.491000101	-0.051086936
4	490552	2153830	-0.61281299	-0.429077658	-0.183735333
5	489502	2151601	-0.55338472	-0.260236417	-0.293148302
6	480054	2157142	0.33185335	0.681074572	-0.349221226
7	481104	2154828	0.73884685	0.761316532	-0.022469683
8	483203	2152713	0.95416525	0.957217968	-0.003052715
9	475855	2149399	0.49585035	0.395288476	0.100561871
10	486353	2148284	0.55338472	-0.005608546	0.558993266
11	482155	2149394	-0.07526986	0.877415468	-0.952685330
12	492651	2147174	-2.32634787	-1.336176778	-0.990171096
13	494751	2149386	-1.88079361	-1.653432169	-0.227361439
14	477955	2144973	1.12639113	0.359896017	0.766495112
15	472694	2140554	-0.22754498	0.423079612	-0.650624588
16	473739	2137233	0.80642125	-0.014064539	0.820485786
17	481095	2139437	1.22652812	0.896232549	0.330295571
18	482154	2144968	0.07526986	0.775225573	-0.699955710
19	486348	2141645	0.12566135	-0.007866443	0.133527789
20	471645	2141662	0.17637416	-0.094681099	0.271055263
21	468489	2138348	-0.49585035	-0.415759327	-0.080091021
22	462178	2133934	-1.64485363	-0.747356873	-0.897496754
23	474787	2135019	0.22754498	-0.000949713	0.228494690
24	472682	2132809	-1.47579103	-0.409916894	-1.065874134
25	485294	2137220	1.03643339	0.701975237	0.334458153
26	491597	2138323	-1.34075503	-0.701839280	-0.638915754
27	504199	2136108	-1.22652812	-0.016422187	-1.210105933
28	497899	2133895	-0.02506891	0.755191985	-0.780260893
29	498949	2131682	1.64485363	0.895617723	0.749235904
30	480038	2133906	0.67448975	1.363608949	-0.689119199
31	486341	2132793	1.34075503	0.775359314	0.565395719
32	482135	2129478	1.88079361	0.485645787	1.395147821
33	485282	2122836	-0.17637416	1.182614255	-1.358988420
34	489495	2136111	-0.43991317	-0.013257537	-0.426655629
35	490542	2128365	0.27931903	1.089634466	-0.810315431
36	497898	2129469	2.32634787	0.942957543	1.383390331
37	477955	2157144	0.61281299	0.248814442	0.363998549
38	494751	2165983	-1.12639113	-0.145706924	-0.980684205
39	473755	2148298	0.38532047	-0.141682060	0.527002526
40	466407	2144991	-0.38532047	-0.661634710	0.276314244
41	485303	2152711	1.47579103	0.574524329	0.901266699
42	491602	2149387	-1.03643339	-1.434233797	0.397800407
43	488451	2146070	-0.33185335	-0.628634770	0.296781424
44	471654	2147195	-0.95416525	0.260289712	-1.214454965
45	469543	2140559	0.02506891	-0.256166783	0.281235691
46	467433	2135030	-0.87789630	-1.102998452	0.225102157
47	470588	2137238	-0.27931903	-0.385627469	0.106308435
48	500000	2140533	-0.80642125	-1.494223324	0.687802077
49	503149	2123937	-0.73884685	0.955181700	-1.694028550
50	472686	2135022	-0.67448975	-0.385285104	-0.289204646

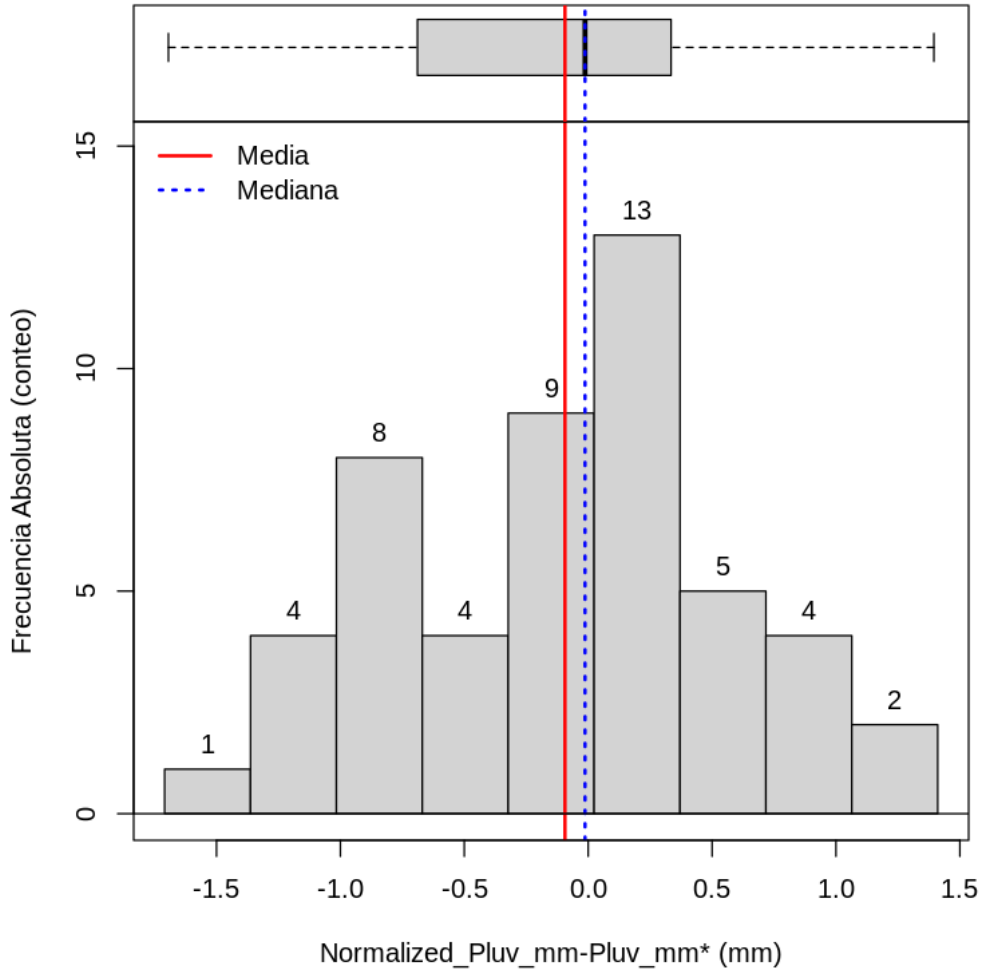
[255]:
```
Normalized_Pluv_mm_CrossValid_Sta<-␣
↪Val_Estadisticos(Normalized_Pluv_mm_CrossValid[,c(3,4,5)])
Normalized_Pluv_mm_CrossValid_Sta
```

A data.frame: 13 × 3

	Z <dbl>	Z* <dbl>	Z-Z* <dbl>
No_muestras	50.00000	50.00000	50.00000
Minimo	-2.32635	-1.65343	-1.69403
Cuartil_1er	-0.65907	-0.38554	-0.67950
Mediana	0.00000	-0.00328	-0.01276
Media	0.00000	0.09369	-0.09369
Cuartil_3er	0.65907	0.74189	0.33342
Maximo	2.32635	1.36361	1.39515
Rango	4.65270	3.01704	3.08918
Rango_Intercuartil	1.31814	1.12743	1.01291
Varianza	0.99481	0.55315	0.50131
Desv_Estandar	0.99740	0.74374	0.70804
Simetria	0.00000	-0.51859	-0.14159
Curtosis	2.72928	2.60729	2.47855

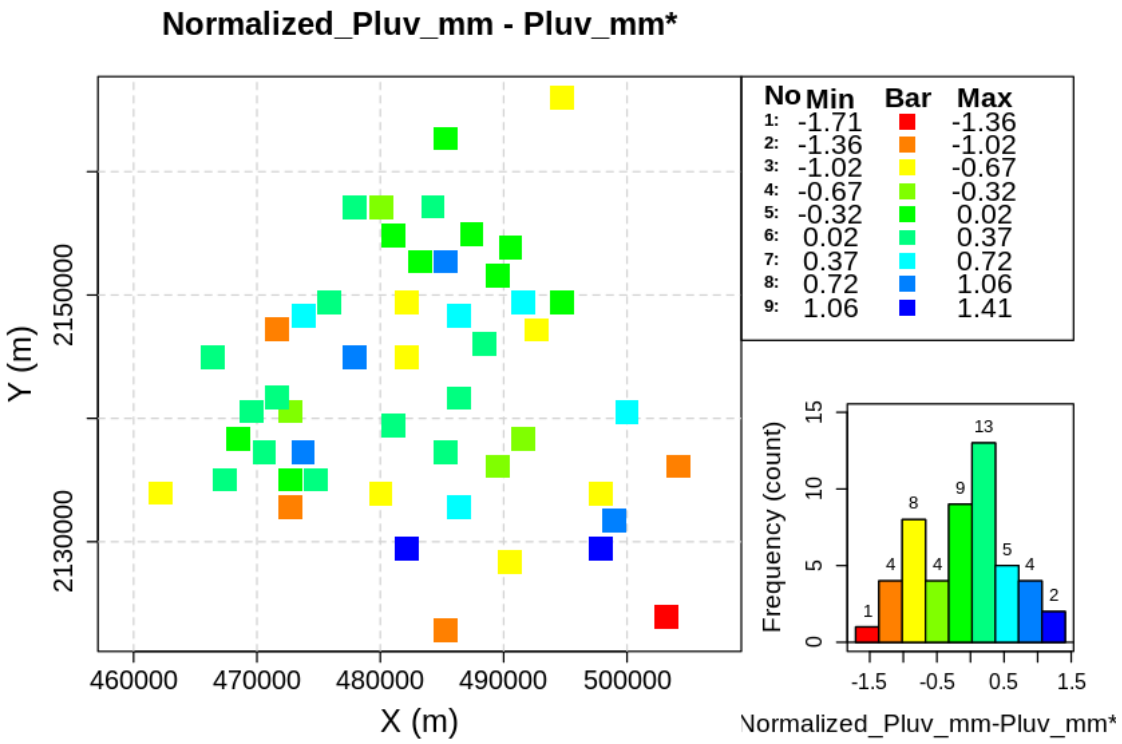
Evaluando los estadígrafos podemos notar que la media es cercana a cero y su varianza es de 0.5, incluso si los comparamos con los resultados de la validación cruzada con las muestras con transformación logarítmica es muy similar.

[256]:
```
HistBoxplot(x=Normalized_Pluv_mm_CrossValid[,5], mean =␣
↪Normalized_Pluv_mm_CrossValid_Sta[5,3],
            median = Normalized_Pluv_mm_CrossValid_Sta[4,3], main ="",
            xlab = "Normalized_Pluv_mm-Pluv_mm* (mm)", ylab = "Frecuencia␣
↪Absoluta (conteo)",
            AbsFreq = TRUE, PercentFreq = FALSE )
```



Respecto al histograma podemos ver que aparentemente no hay valores atípicos. Sin embargo, al analizar el grafico de la distribución espacial podemos notar que los valores atípicos espaciales son los mismos que los reportados con la transformación logarítmica.

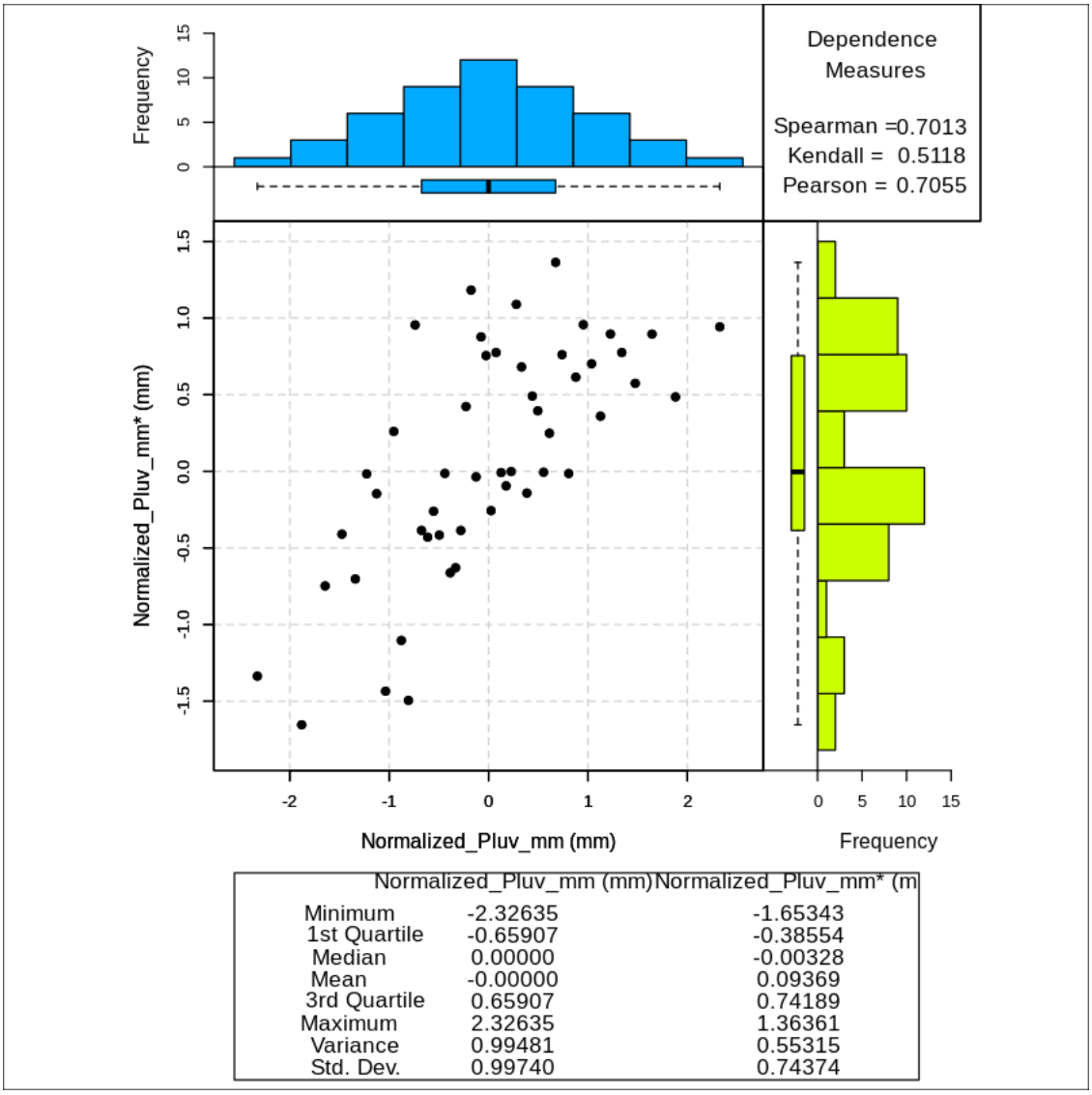
[257]:
```
DEspacial(Normalized_Pluv_mm_CrossValid[,1], Normalized_Pluv_mm_CrossValid[,2],␣
↪Normalized_Pluv_mm_CrossValid[,5],
          n_bins=9,'X (m)', 'Y (m)', 'Normalized_Pluv_mm-Pluv_mm* (mm)',␣
↪'Normalized_Pluv_mm - Pluv_mm*')
```



[258]:
```
# Pluv_mm_Log is the independent variable
X<-Normalized_Pluv_mm_CrossValid[,3] # the same as Pluv_mm_Log
# Pluv_mm_Log* is the dependent variable
Y<-Normalized_Pluv_mm_CrossValid[,4]
```

[259]:
```
Normalized_Pluv_mm_CrossValid_stat <-␣
↪Estadisticas(Normalized_Pluv_mm_CrossValid[,4])

ScatterPlot(Normalized_Pluv$nscore, Normalized_Pluv_mm_CrossValid[,4], 9,
            Xmin = Normalized_Pluv_Stat[2,2], Xmax = Normalized_Pluv_Stat[7,2],
            Ymin = Normalized_Pluv_mm_CrossValid_stat[2,2],Ymax =␣
↪Normalized_Pluv_mm_CrossValid_stat[7,2],
```

```
        XLAB = "Normalized_Pluv_mm (mm)", YLAB = "Normalized_Pluv_mm* (mm)")
```



	Normalized_Pluv_mm (mm)	Normalized_Pluv_mm* (m
Minimum	-2.32635	-1.65343
1st Quartile	-0.65907	-0.38554
Median	0.00000	-0.00328
Mean	-0.00000	0.09369
3rd Quartile	0.65907	0.74189
Maximum	2.32635	1.36361
Variance	0.99481	0.55315
Std. Dev.	0.99740	0.74374

Respecto a los resultados del scatterplot entre la muestra normalizada y la estimada por kriging durante la validación cruzada tiene valores de dependencia muy similares al reportado con la transformación logarítmica.

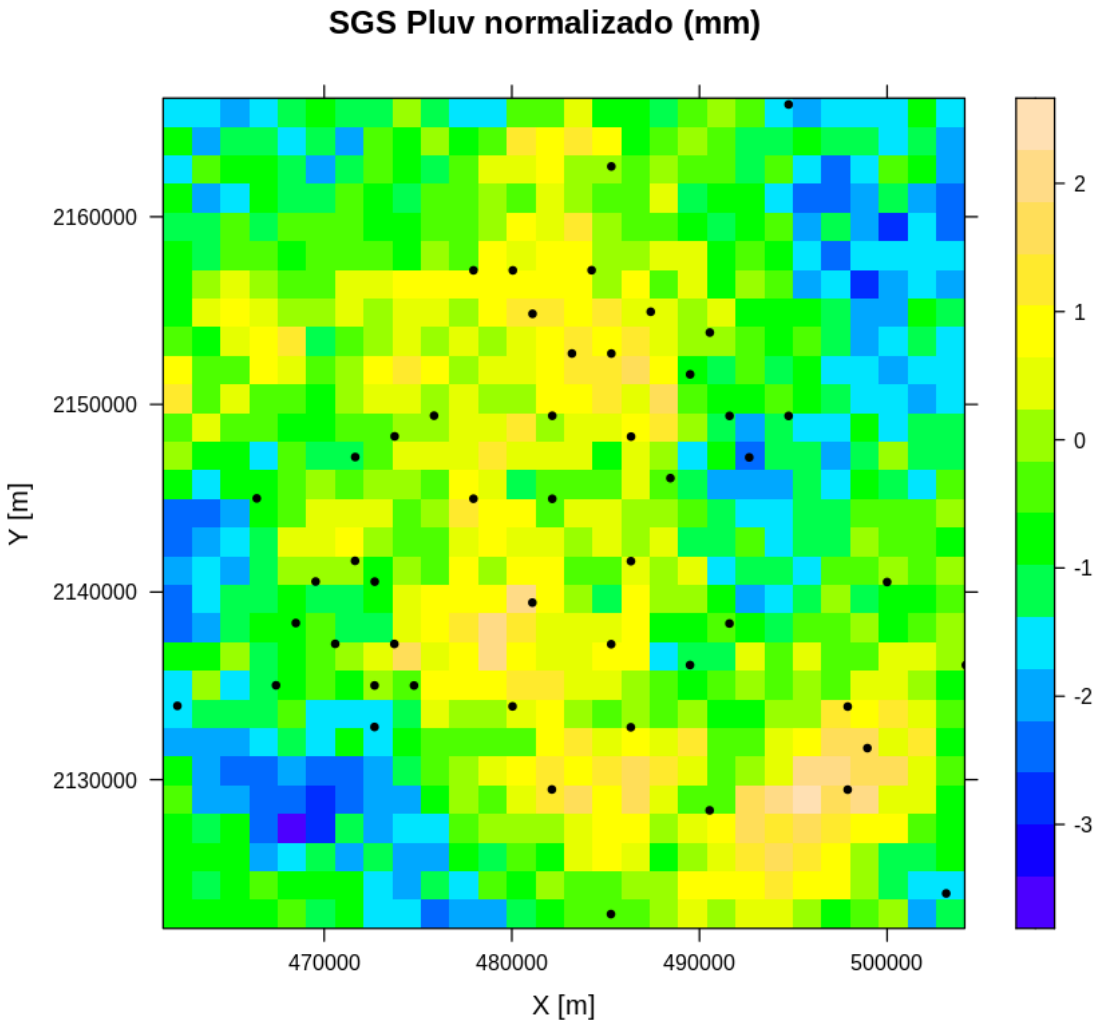
Ya que tenemos el modelo de variograma podemos hacer la simulación condicional. Para implementar la simulación se usa la función "SGS". Esta función se estructura de la siguiente forma:

- Las coordenadas (CoorX, CoorY), la variable a usar (Prop1) que en este caso son las muestras obtenidas de los pluviómetros con transformación logarítmica (Pluv_mm_Log)
- La información del variograma, que en este caso es el modelo (Pluv_mm_Log_vario_model), valor del nugget (Pluv_mm_Log_nugget), valor de la meseta más el nugget (Pluv_mm_Log_sill_and_nugget) y el alcance (Pluv_mm_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 podríamos causar la creación de artefactos.
- Se indica el tipo de transformación que tiene la variable (InvT). Dado que la variable tiene transformación normal solo se asigna 0.
- Las leyendas del gráfico: para el eje X (NameX), para el eje Y "NameY" y el título de la imagen resultante de la simulación "Titulo1".
- Por último, indicamos el número de simulaciones que deseamos calcular (n_sim)

[260]:
```
Normalized_Pluv_SGS <- SGS(XCoord, YCoord,
                           Prop1 = Normalized_Pluv$nscore, Modelo =␣
↪Normalized_Pluv_vario_model, Nugget = Normalized_Pluv_nugget,
                           SillYNugget = Normalized_Pluv_sill_and_nugget,␣
↪Alcance = Normalized_Pluv_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=0, NameX="X [m]",
                           NameY="Y [m]", Titulo1="SGS Pluv normalizado (mm)",␣
↪n_sim = 10)
```

```
drawing 10 GLS realisations of beta...
[using conditional Gaussian simulation]
```



Si ejecutamos el vector "Normalized_Pluv_SGS" podemos ver que su estructura esta distribuida de la siguiente forma: las dos primeras columnas contienen las coordenadas de las muestras simuladas y a partir de la tercera columna tenemos las simulaciónes, en este caso se hicieron 10.

[261]:
```
Normalized_Pluv_SGS
```

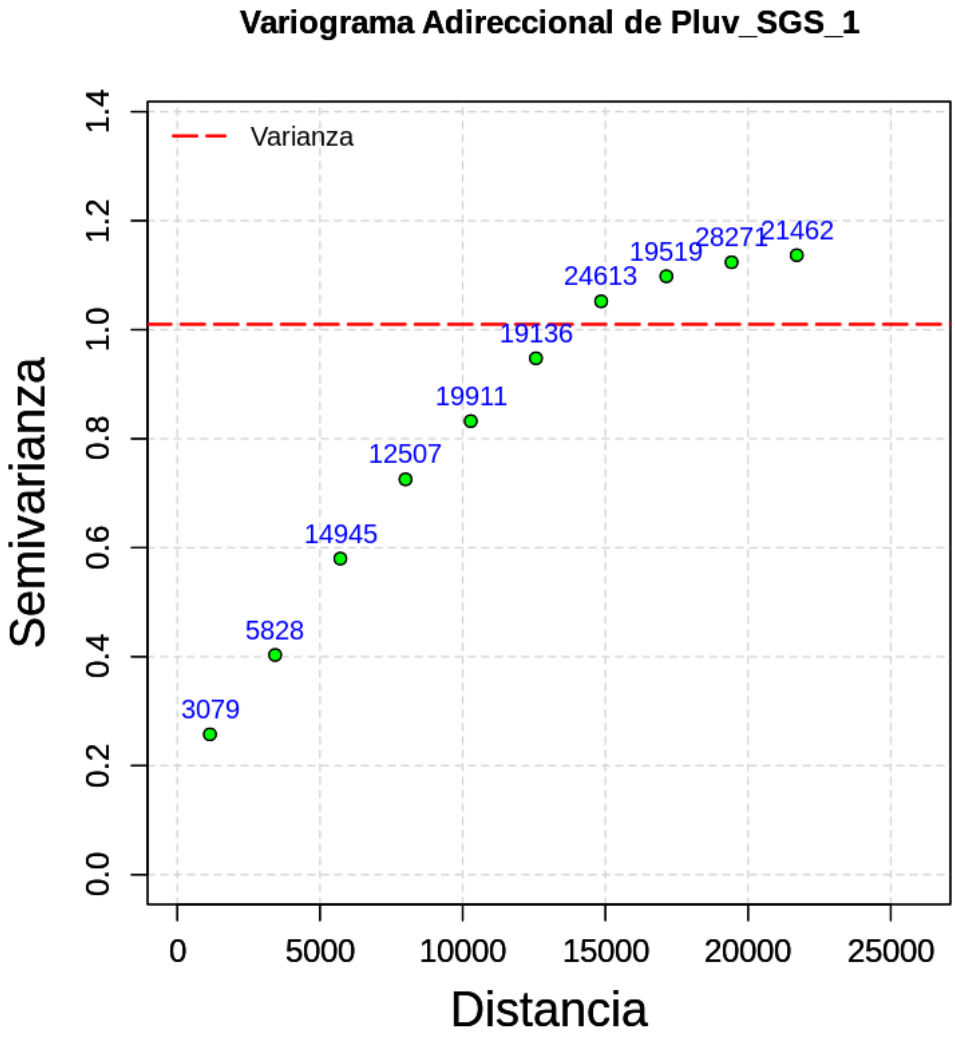
A matrix: 812 × 12 of type dbl

	Var1	Var2	sim1	sim2	sim3	sim4
1	462178.0	2122836	-0.66299105	-0.8623175025	0.43215951	-2.527539
2	463703.8	2122836	-0.62352318	-0.2331287116	-0.11634643	-2.778054
3	465229.6	2122836	-0.94393462	-0.1454488039	-0.78333366	-3.275978
4	466755.4	2122836	-0.84493256	-0.6005839705	-1.42271006	-2.974512
5	468281.2	2122836	-0.40814364	0.2467078567	-2.06416965	-3.232894
6	469807.0	2122836	-1.32952034	-0.3239098489	-1.15473807	-3.138960
7	471332.8	2122836	-0.59410715	-0.6116937399	-1.53655005	-2.719784
8	472858.6	2122836	-1.57488728	0.3948004246	-1.06966329	-3.301563
9	474384.4	2122836	-1.64972818	-1.1044524908	-0.77907926	-1.875341
10	475910.2	2122836	-2.40519738	-1.0904313326	-1.16386724	-1.436118
11	477436.0	2122836	-1.83235276	-0.7179629803	-0.69149256	-1.958147
12	478961.8	2122836	-1.89206326	-0.2125950456	-0.59041154	-0.992233
13	480487.6	2122836	-1.29280066	-0.7528117299	-0.94774723	-0.648917
14	482013.4	2122836	-0.92788184	-0.0004439472	-0.50119311	-0.129016
15	483539.2	2122836	-0.31511635	0.4698901176	-0.55882978	0.3668007
16	485065.0	2122836	-0.43807772	-0.4285125434	-0.65224499	0.1568494
17	486590.8	2122836	-0.31417567	-0.1232054606	-0.22558691	-0.693955
18	488116.6	2122836	-0.04689626	-0.4659109712	0.21209151	-0.126740
19	489642.4	2122836	0.56043839	-0.1478317529	0.12360626	-0.607318
20	491168.2	2122836	0.08262041	0.5109724998	-0.17461169	-1.110500
21	492694.0	2122836	0.34524608	0.3247115314	0.33214921	-0.044754
22	494219.8	2122836	0.35374534	1.3434544802	1.08320522	-0.379166
23	495745.6	2122836	0.12076803	1.8482322693	0.52402174	0.5606424
24	497271.4	2122836	-0.61410999	1.6708080769	0.96525407	-0.557697
25	498797.2	2122836	-0.28430641	0.4282688498	0.73021787	0.3432040
26	500323.0	2122836	0.01839113	0.4065772295	0.99286199	0.6713569
27	501848.8	2122836	-2.15336299	-0.4264852107	0.08134577	-0.124925
28	503374.6	2122836	-1.34062517	-1.3997641802	0.57208639	-0.920535
29	462178.0	2124362	-0.95227933	-0.2143409699	-0.23590222	-2.450924
30	463703.8	2124362	-1.30040109	-0.2377137393	-1.32318449	-2.648911
⋮	⋮	⋮	⋮	⋮	⋮	⋮
783	501848.8	2164033	-1.22457302	-2.76219463	0.62261993	0.3847231
784	503374.6	2164033	-2.04023552	-2.41813421	0.66615820	-0.907326
785	462178.0	2165558	-1.47274482	-1.94758236	-0.31800875	0.6025177
786	463703.8	2165558	-1.72589016	-1.56009316	-1.59042108	-0.378592
787	465229.6	2165558	-1.85491824	-1.43481600	-1.63112819	0.4339723
788	466755.4	2165558	-1.39998436	-1.70790720	-1.78206277	0.3514278
789	468281.2	2165558	-1.30195570	-1.01659739	-1.55221665	-0.053375
790	469807.0	2165558	-0.87032700	-1.17021906	-1.24992681	-0.324634
791	471332.8	2165558	-1.26371551	-0.71321857	-0.99826181	-0.836254
792	472858.6	2165558	-1.27258551	-0.38649076	-1.45071185	0.0920708
793	474384.4	2165558	0.01587747	-0.08162922	-1.13832903	-0.025418
794	475910.2	2165558	-1.24203873	0.41565406	-0.89283055	-0.287942
795	477436.0	2165558	-1.38789654	0.68534666	-0.67483383	-0.191480
796	478961.8	2165558	-1.76719141	0.81927031	0.25300467	-0.679222
797	480487.6	2165558	-0.56323665	1.57101691	-0.13370070	-1.538374
798	482013.4	2165558	-0.36029133	0.97984082	0.14458247	-0.584907
799	483539.2	2165558	0.30196869	0.18411501	0.51668388	-0.685760
800	485065.0	2165558	-0.59568805	-0.32398453	0.33004847	-1.236028
801	486590.8	2165558	-0.64077097	-0.37582004	0.08257899	-0.215682
802	488116.6	2165558	-1.30058086	-0.77497673	0.79398817	0.5012320
803	489642.4	2165558	-0.38448897	0.03495660	-0.40494359	0.7490157

Obtenemos el variograma resultante de una de las simulaciones para comprobar que la simulación se basa en el modelo de variograma propuesto.

[262]:
```
Pluv_SGS_1_Log_VarioEstimation<-Variograma(Normalized_Pluv_SGS[,1],␣
↪Normalized_Pluv_SGS[,2],
                                           Normalized_Pluv_SGS[,3], 0, 90,␣
↪1*N_lags, lag_value, 1,
                                           "Variograma Adireccional de␣
↪Pluv_SGS_1")
```

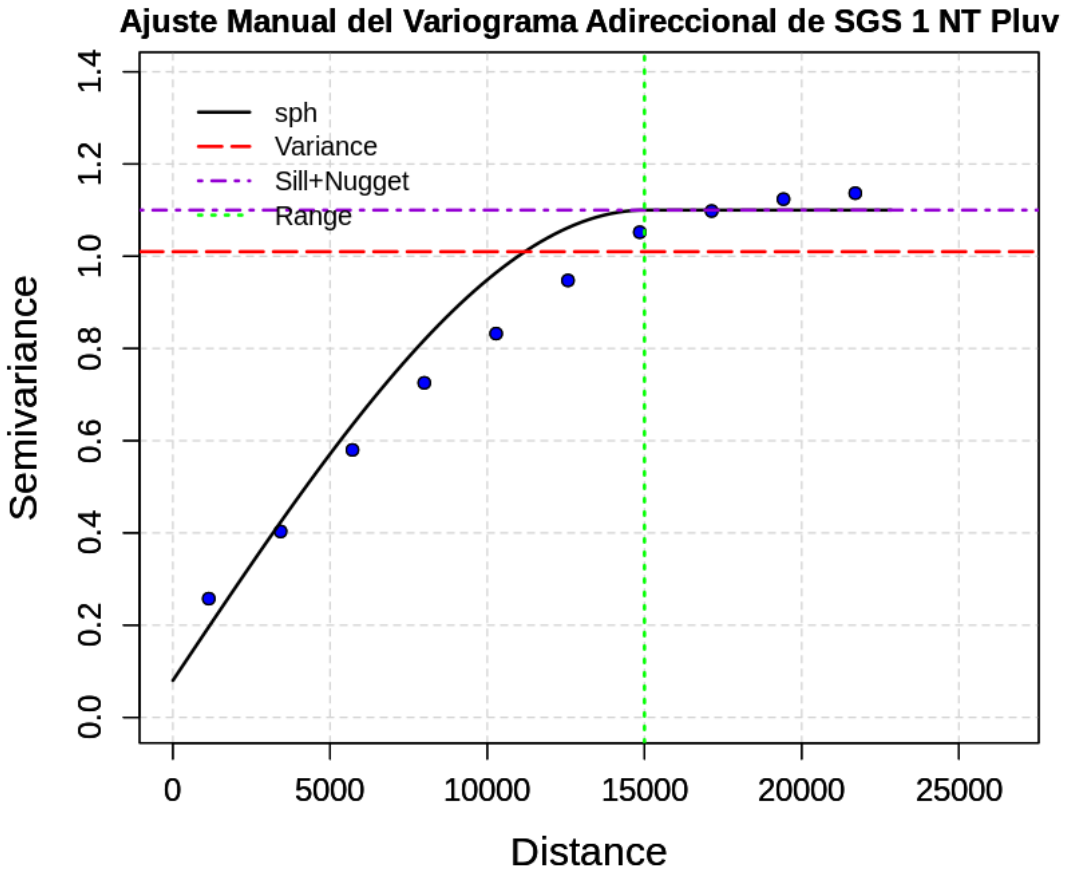
```
variog: computing omnidirectional variogram
```



[263]:
```
SGS_Pluv_vario_model<- 2
SGS_Pluv_nugget<- 0.08
SGS_Pluv_sill_and_nugget<- 1.1
SGS_Pluv_rank <- 15000
```

[264]:
```
Pluv_SGS_1_EyeModelVarioFit<-EyeModel(Normalized_Pluv_SGS[,1],␣
↪Normalized_Pluv_SGS[,2],
                                      Normalized_Pluv_SGS[,3], 0, 90, N_lags,␣
↪lag_value, 1,
                                      SGS_Pluv_vario_model, SGS_Pluv_nugget,␣
↪SGS_Pluv_sill_and_nugget,
                                      SGS_Pluv_rank,
                                      "Ajuste Manual del Variograma Adireccional␣
↪de SGS 1 NT Pluv")
```

```
variog: computing omnidirectional variogram
```

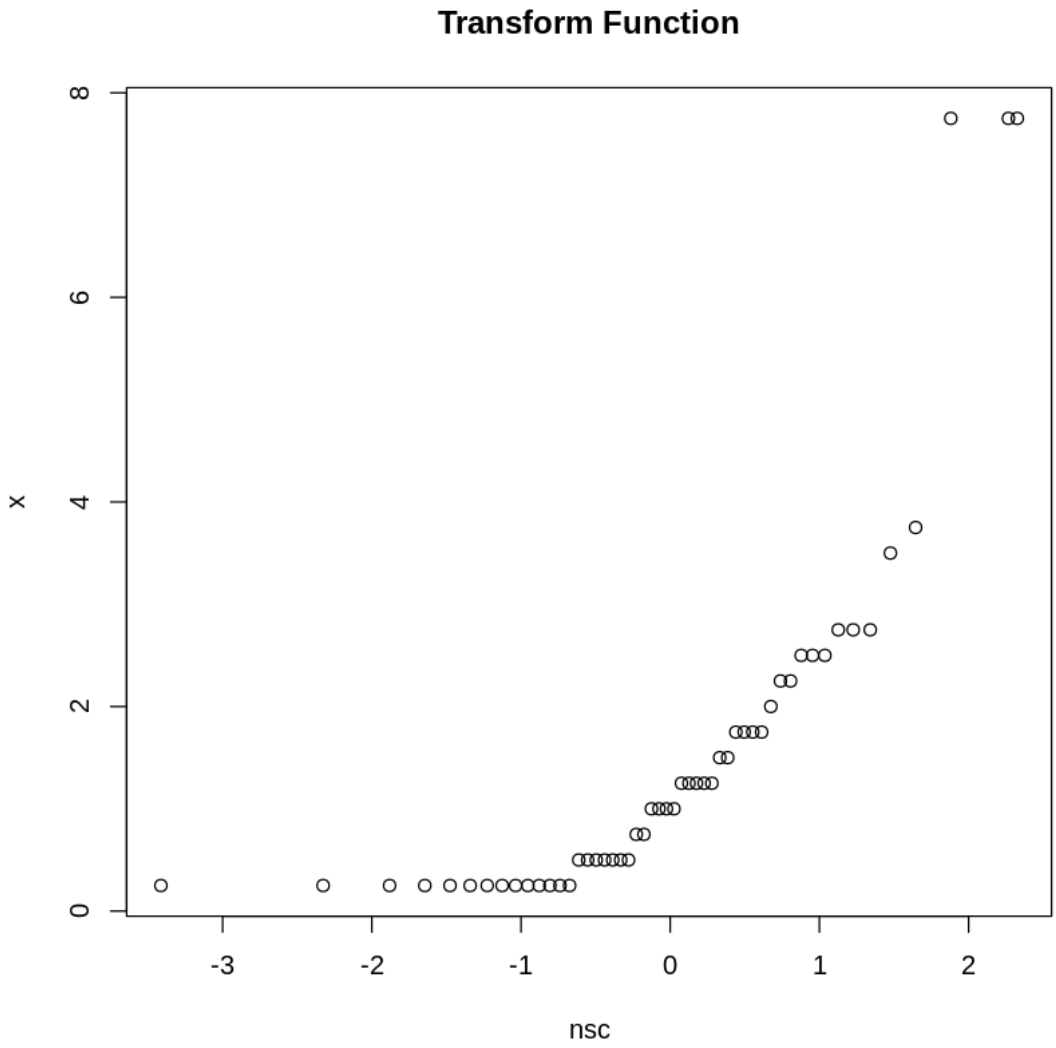


Model	Nugget	Sill+Nugget	Range	MSE
sph	0.0800	1.1000	15000.0000	0.0505

Ya que tenemos las simulaciones hay que invertir los valores simulados para obtener los resultados en el dominio original. para esto usamos la función "backtr", esta nos pide los siguientes requisitos:

- El vector con las muestras simuladas
- Las muestras normalizadas
- Indicar si la función tiene cola (tails)
- Indicar si se requiere el grafico de la distribución acumulativa, si el valor es TRUE se mostrará un gráfico con las muestras transformadas

[265]:
```
Pluv_SGS_1 <- backtr(Normalized_Pluv_SGS[,3], Normalized_Pluv, tails='none',␣
↪draw=TRUE)
```



Ya que tenemos las muestras en el dominio original podemos obtener su estadígrafos e histograma, los cuales podemos comparar con las muestras originales:

[266]:
```
Pluv_SGS_1_Stat <- Estadisticas(Pluv_SGS_1)
Pluv_SGS_1_Stat  # estadígrafos de la simulación 1
Pluv_mm_Stat  #estadígrafos originales
```

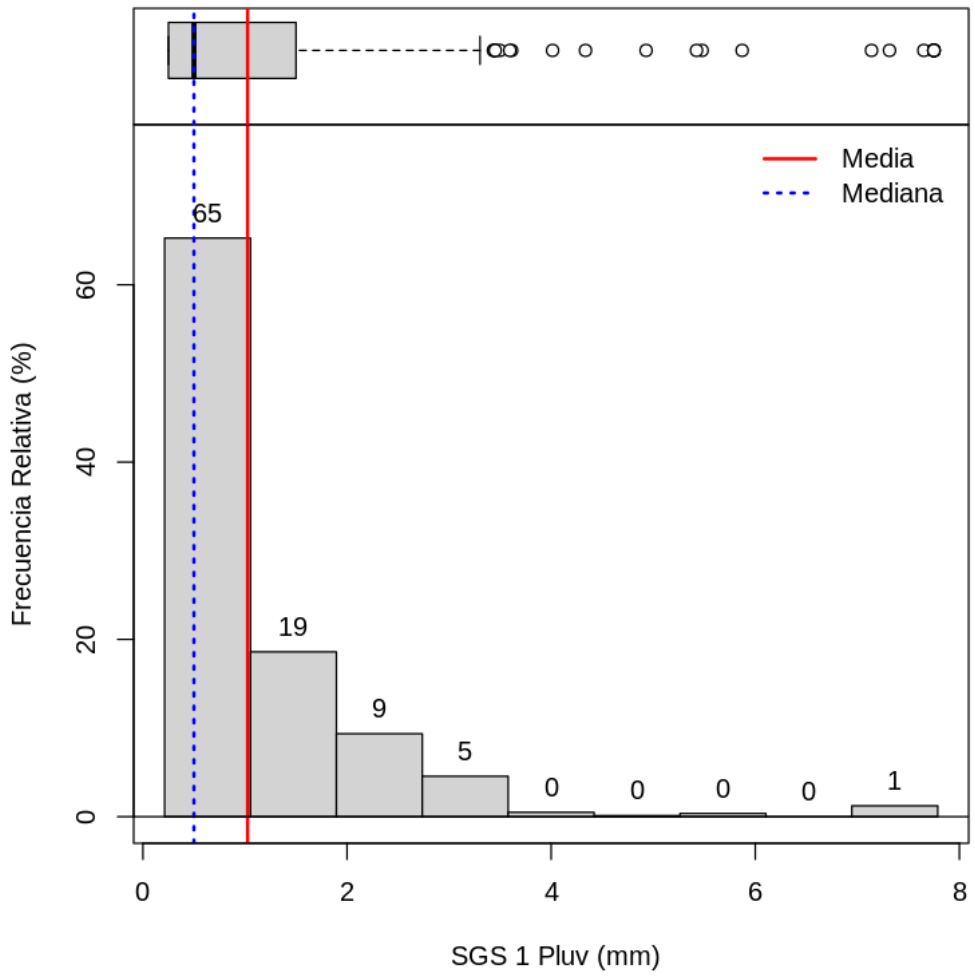
A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	812.0000
minimos	Minimum	0.2500
cuantiles1	1st. Quartile	0.2500
medianas	Median	0.5000
medias	Mean	1.0258
cuantiles3	3rd. Quartile	1.5000
maximos	Maximum	7.7500
rangos	Rank	7.5000
rangosInt	Interquartile Rank	1.2500
varianzas	Variance	1.3397
desvs	Standard Deviation	1.1575
CVs	Variation Coeff.	1.1283
simetrias	Skewness	2.8903
curtosiss	Kurtosis	15.1015

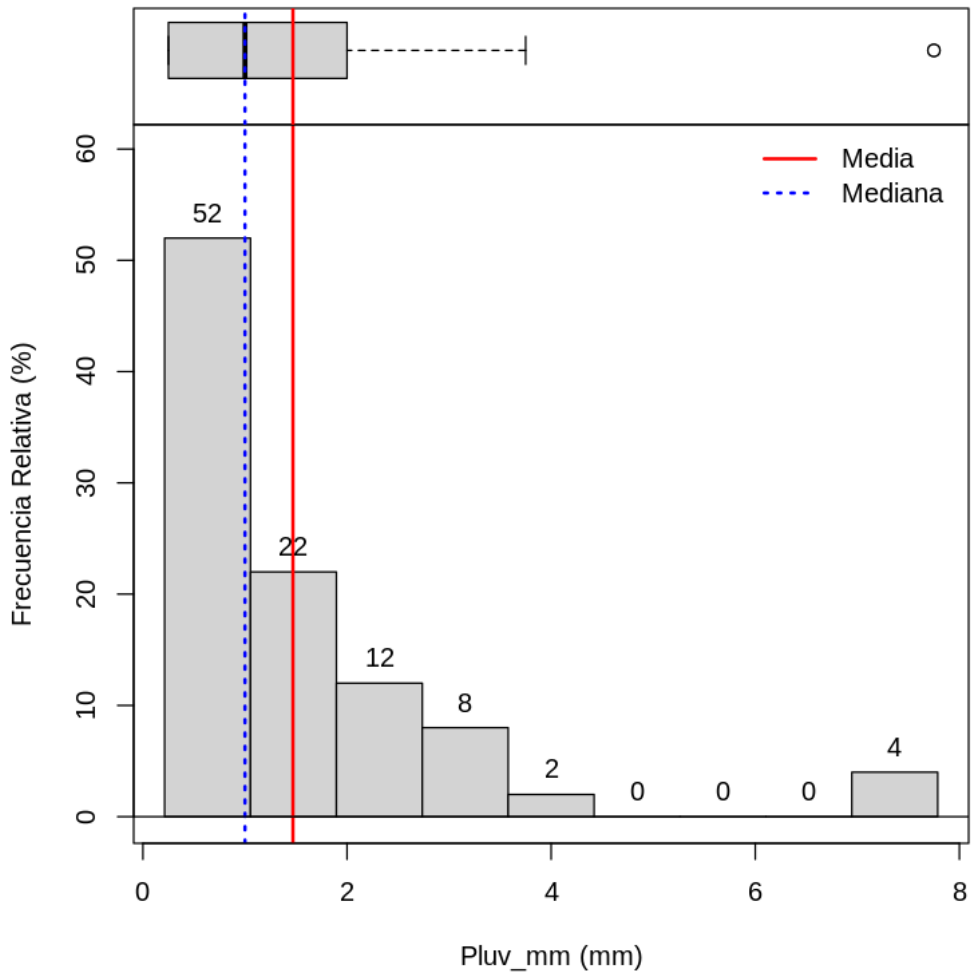
A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	50.0000
minimos	Minimum	0.2500
cuantiles1	1st. Quartile	0.3125
medianas	Median	1.0000
medias	Mean	1.4700
cuantiles3	3rd. Quartile	1.9375
maximos	Maximum	7.7500
rangos	Rank	7.5000
rangosInt	Interquartile Rank	1.6250
varianzas	Variance	2.5756
desvs	Standard Deviation	1.6049
CVs	Variation Coeff.	1.0917
simetrias	Skewness	2.4693
curtosiss	Kurtosis	10.0542

[267]:
```
HistBoxplot(x=Pluv_SGS_1, mean = Pluv_SGS_1_Stat[5,2], median =␣
↪Pluv_SGS_1_Stat[4,2], main ="",
            xlab = " SGS 1 Pluv (mm)", ylab = "Frecuencia Relativa (%)", AbsFreq␣
↪= FALSE, PercentFreq = TRUE,
            nbin =9)
```



[268]:
```
HistBoxplot(x=Pluv_mm, mean = Pluv_mm_Stat[5,2], median = Pluv_mm_Stat[4,2],␣
↪main ="",
            xlab = "Pluv_mm (mm)", ylab = "Frecuencia Relativa (%)", AbsFreq =␣
↪FALSE, PercentFreq = TRUE,
            nbin = 9)
```



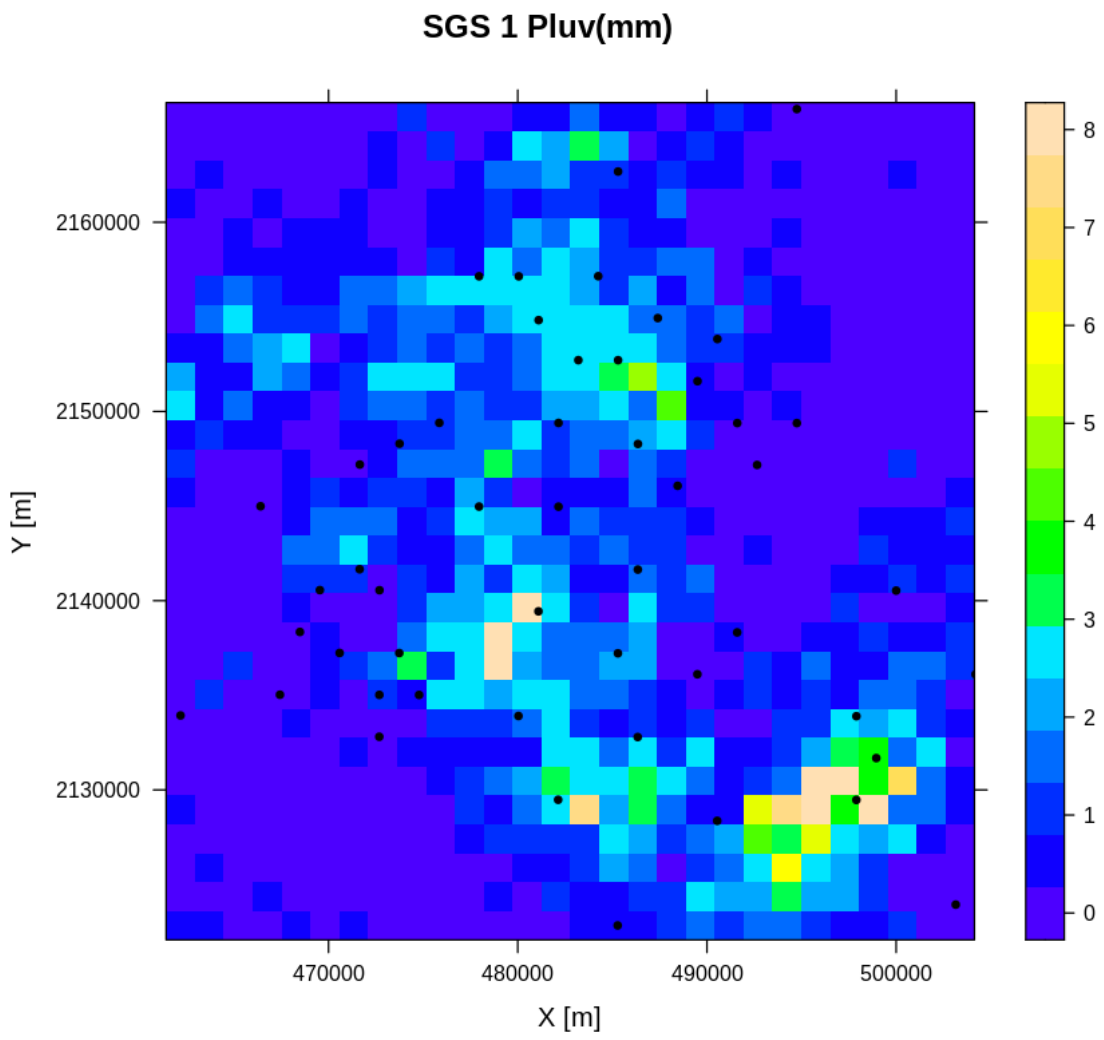
Comparando los estadígrafos podemos ver que son muy similares, respecto al histograma su forma es muy similar, salvo por las muestras ubicadas a la derecha.

Para graficar la simulación 1 usamos el grafico "spplot", el cual necesita lo siguiente:

- Los vectores de las coordenadas y la simulación en formato de tabla (dataframe)
- La variable simulada (c(variable simulada)), esta variable debe estar nombrada.
- Títulos (Titulo1) y nombres de los ejes (NameX, NameY)
- Paleta de colores (col.regions)
- Barra de escalas (scales)
- Ubicación de las muestras condicionantes (sp.layout)

[269]:
```
DatosOrg <- as.data.frame(cbind(XCoord, YCoord, Pluv_mm))
colnames(DatosOrg) <- c("X", "Y", "V1")
```

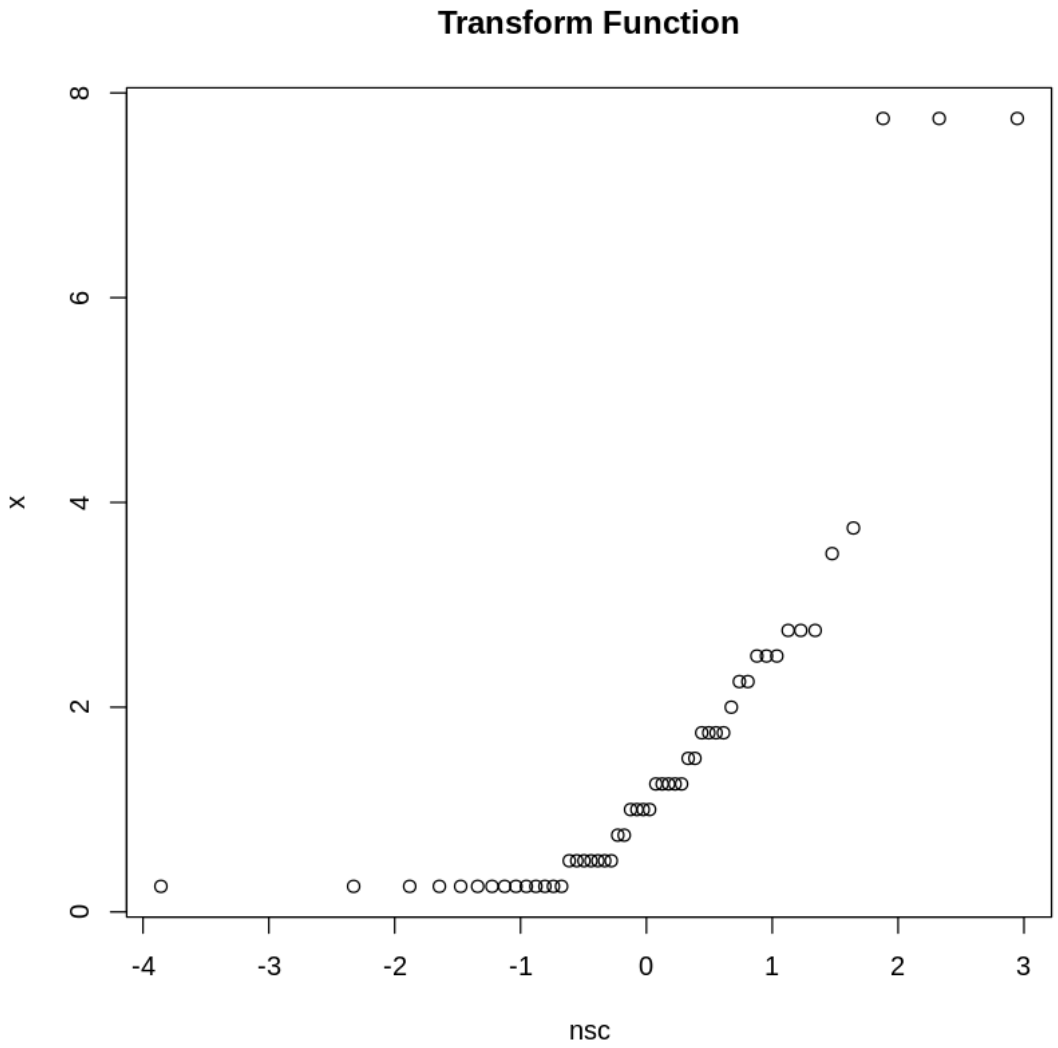
```
coordinates(DatosOrg) <- ~X + Y
Salida <- as.data.frame(cbind(Normalized_Pluv_SGS[,1], Normalized_Pluv_SGS[,2],␣
 ↪Pluv_SGS_1))
Salida1<-Salida
gridded(Salida) <- ~V1+V2
NameX="X [m]"
NameY="Y [m]"
Titulo1="SGS 1 Pluv(mm)"
spplot(Salida, c("Pluv_SGS_1"), main=Titulo1, xlab = NameX , ylab = NameY , col.
 ↪regions=topo.colors,
       do.log = TRUE,
       key.space=list(x = 0.1, y = 0.95, corner = c(0, 1)),
       scales=list(draw = TRUE),
       sp.layout = list("sp.points", DatosOrg, pch = 20, col = "black"))
```

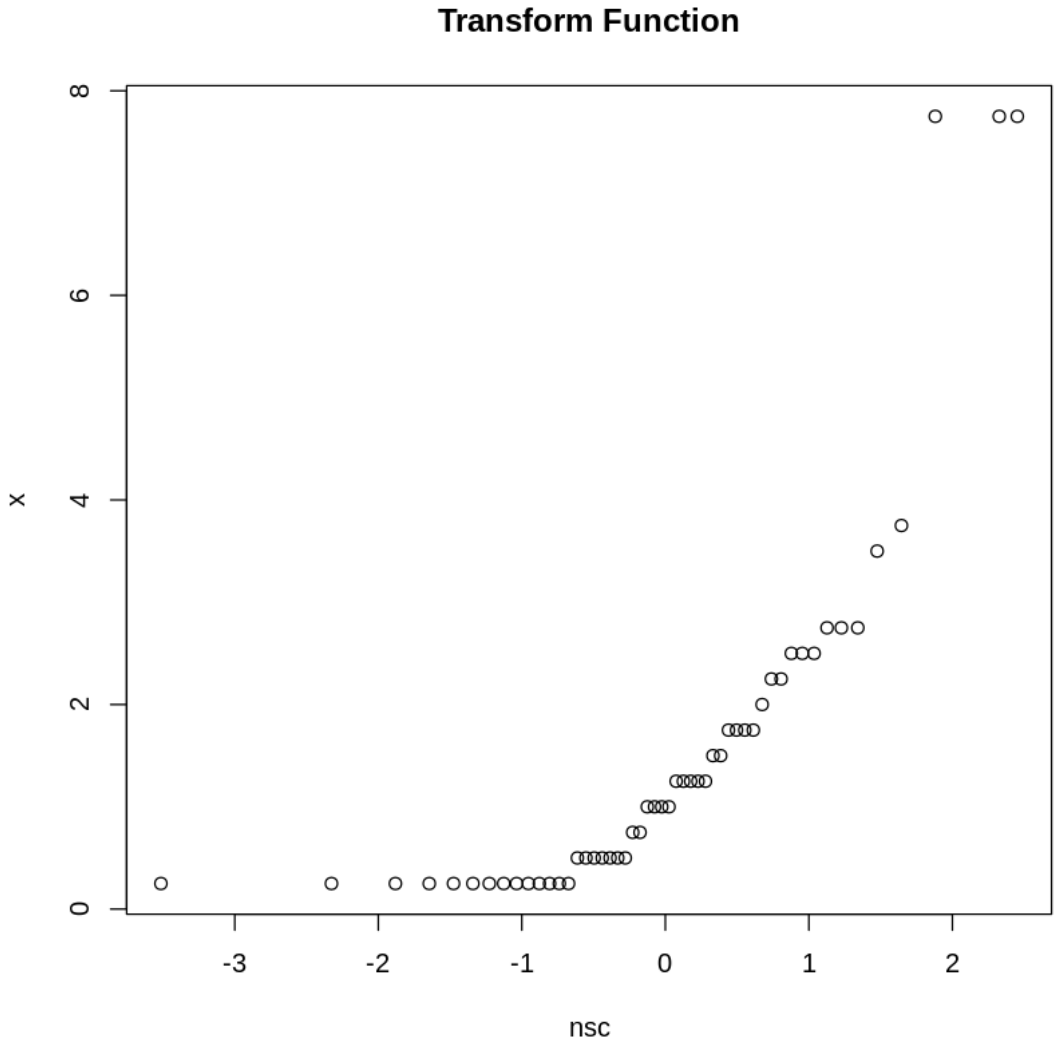


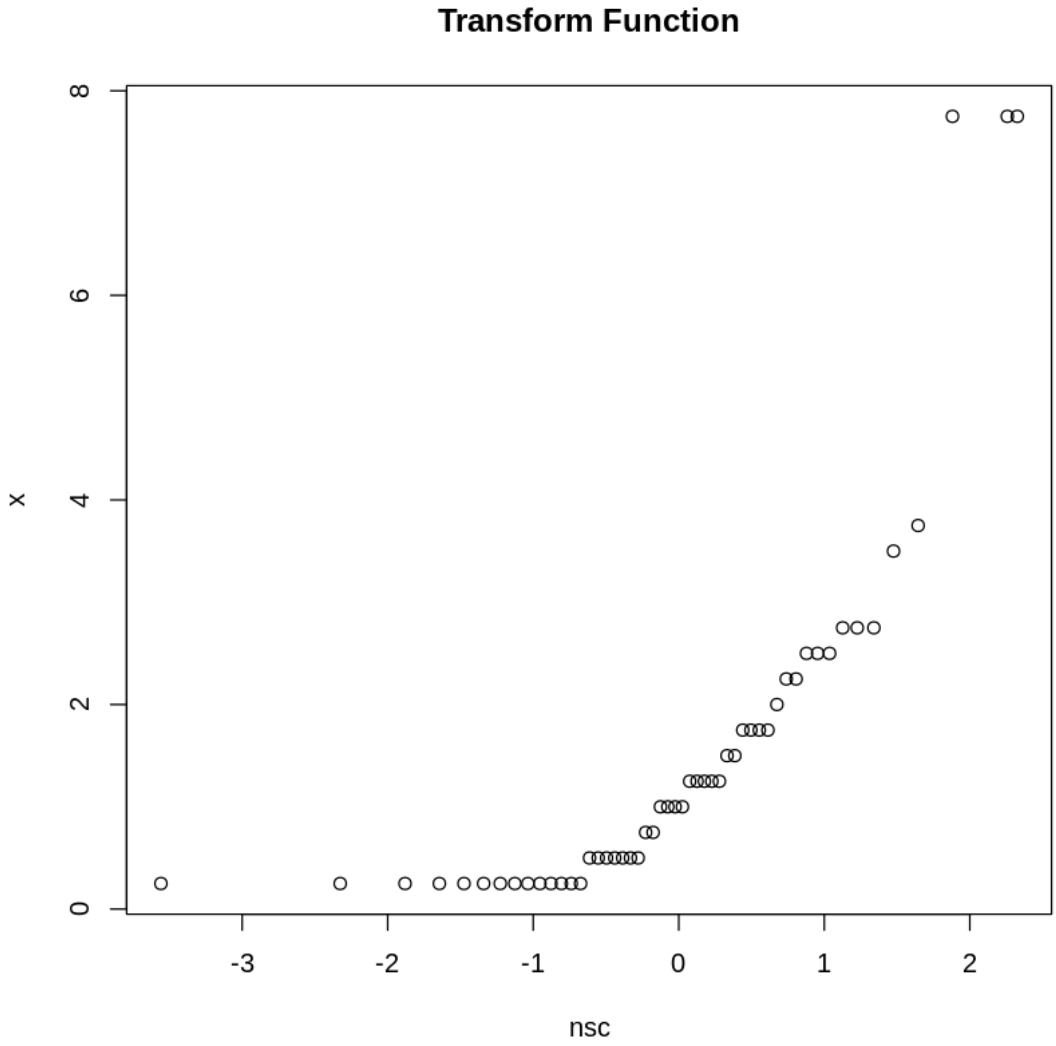
También necesitamos la distribución espacial de la media, desviación estándar y cuantiles de las 10 simulaciones. Eso lo hacemos de la siguiente forma:

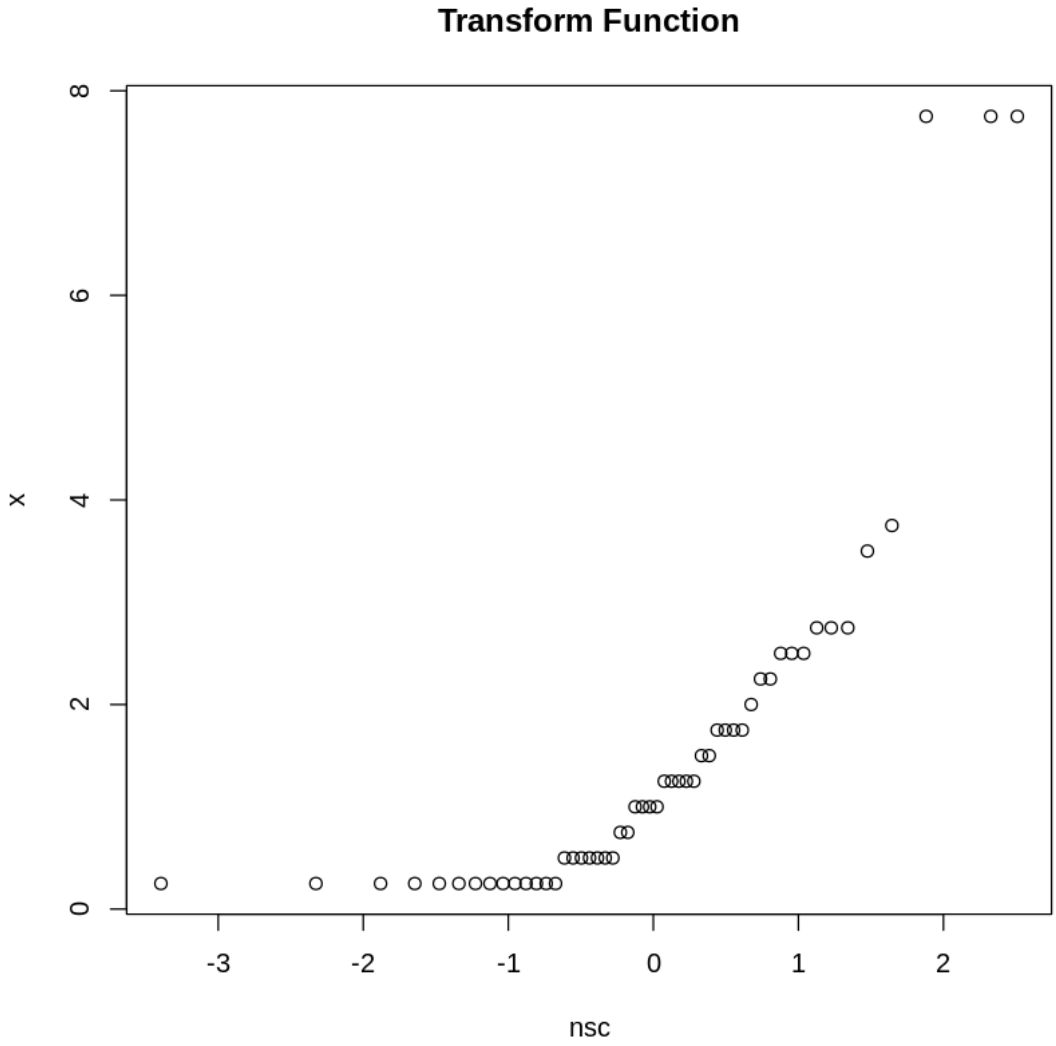
Primero obtenemos la transformada inversa de las 10 simulaciones usando el siguiente ciclo

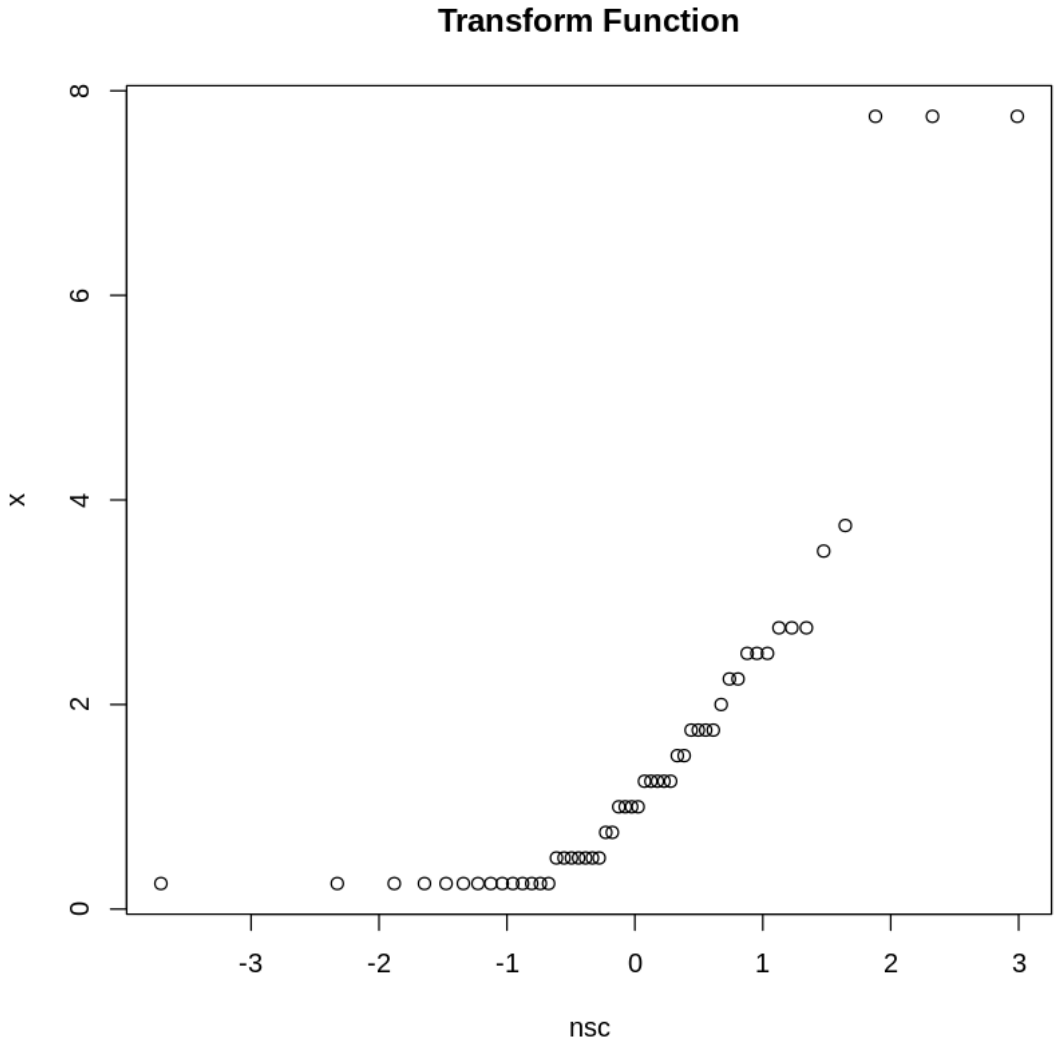
[270]:
```
Pluv_SGS_nsim = Pluv_SGS_1
for (i in seq(from= 4, to = 102, by =1)) {
  sim_back = backtr(Normalized_Pluv_SGS[,i], Normalized_Pluv, tails='none',␣
 ↪draw=TRUE)
  Pluv_SGS_nsim = cbind(Pluv_SGS_nsim, sim_back)
}
```

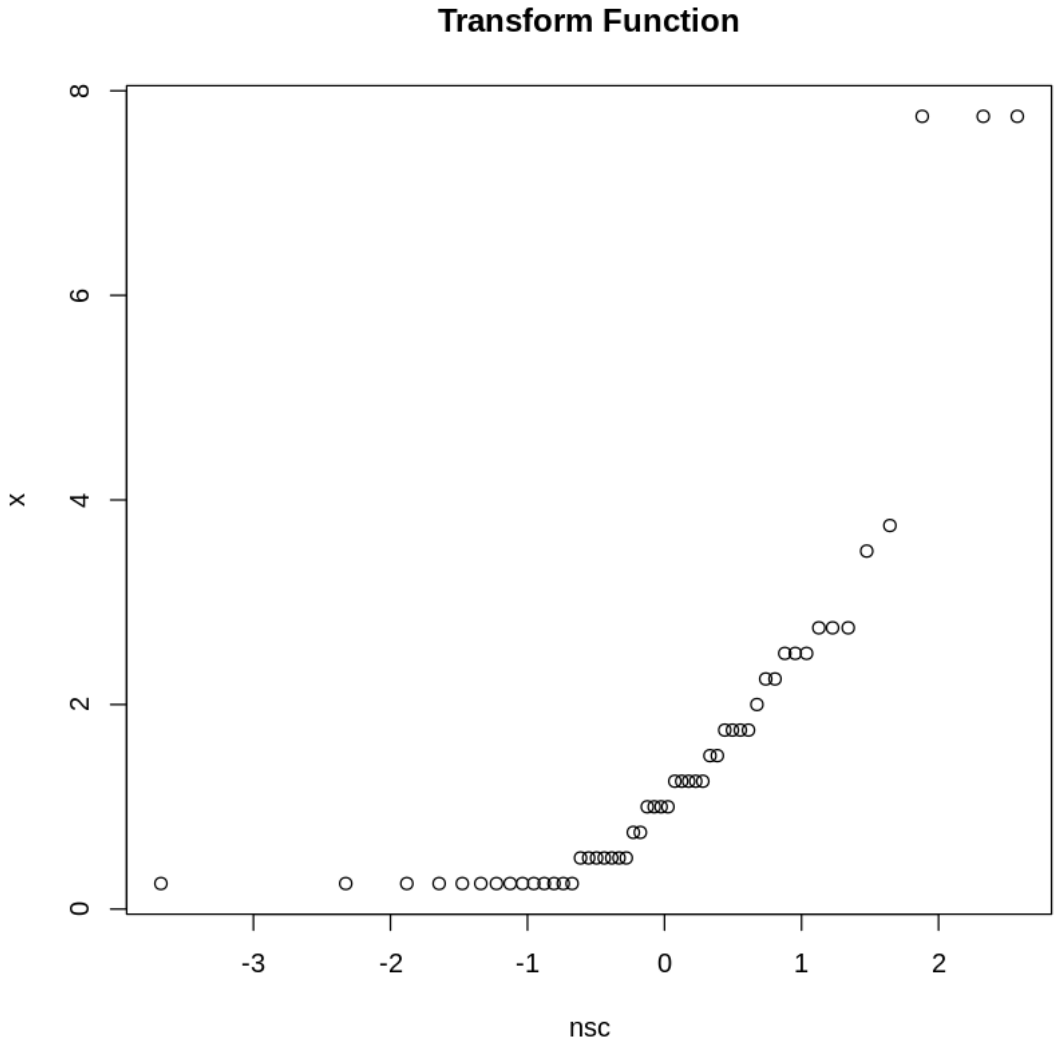


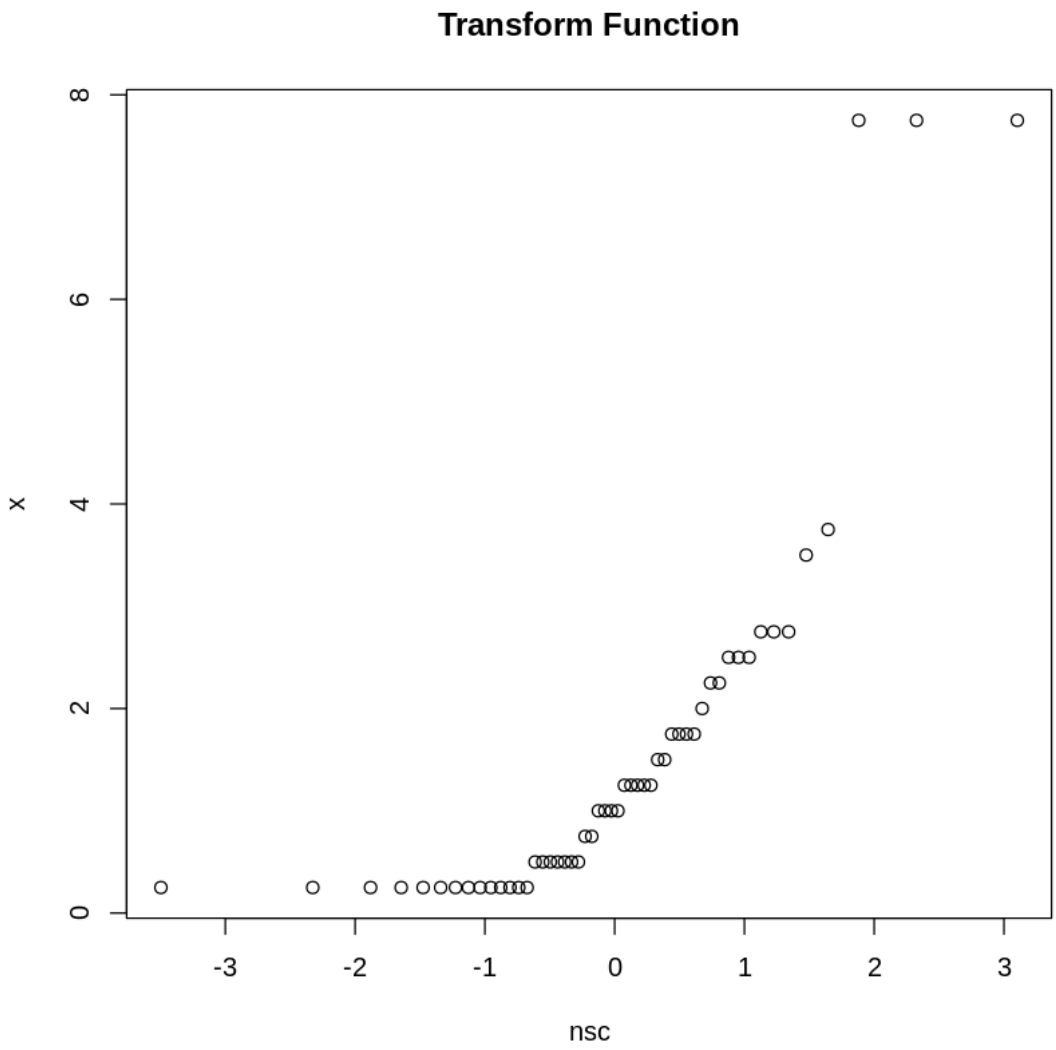

Transform Function
x
nsc
0
2
4
6
8
-3
-2
-1
0
1
2


Transform Function
x
nsc
0
2
4
6
8
-3
-2
-1
0
1
2

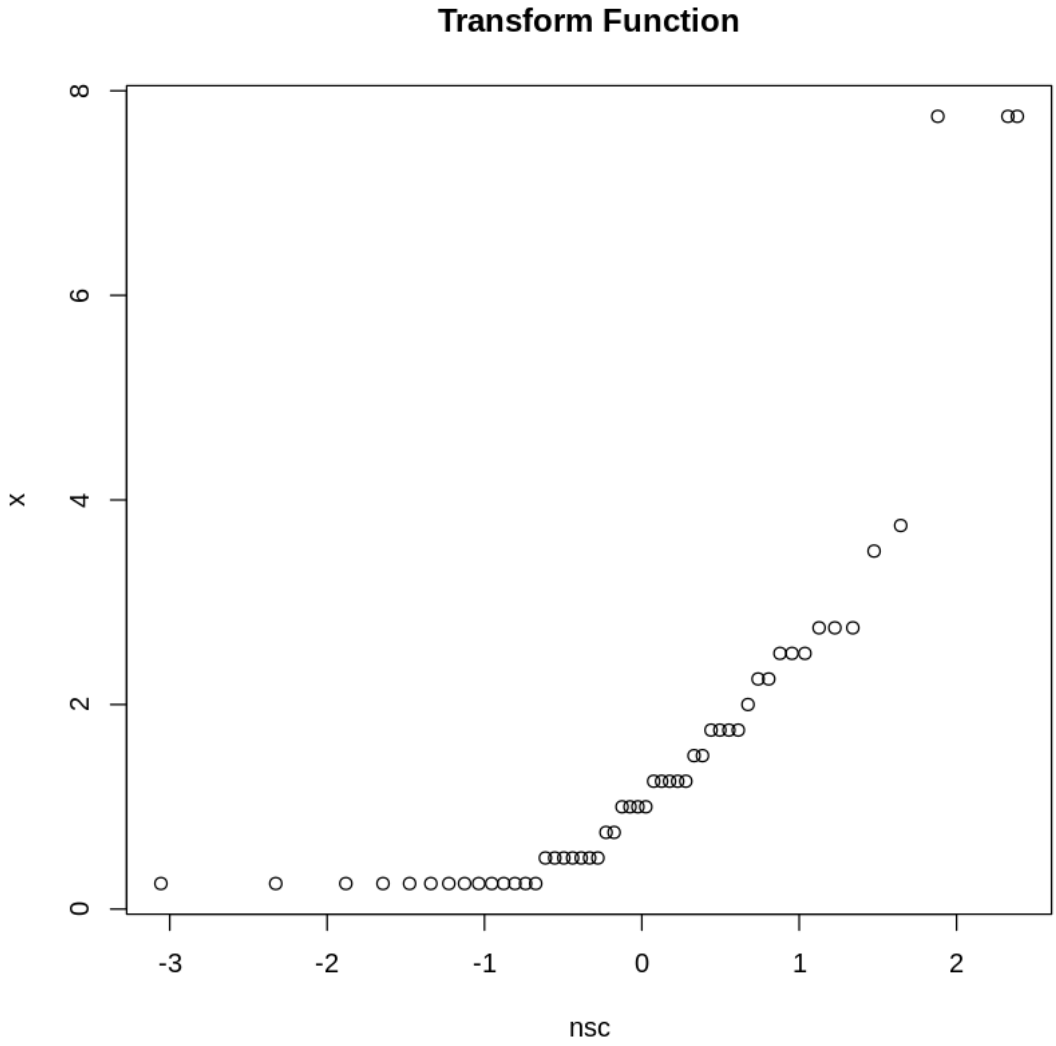

Transform Function
x
nsc
0
2
4
6
8
-3
-2
-1
0
1
2

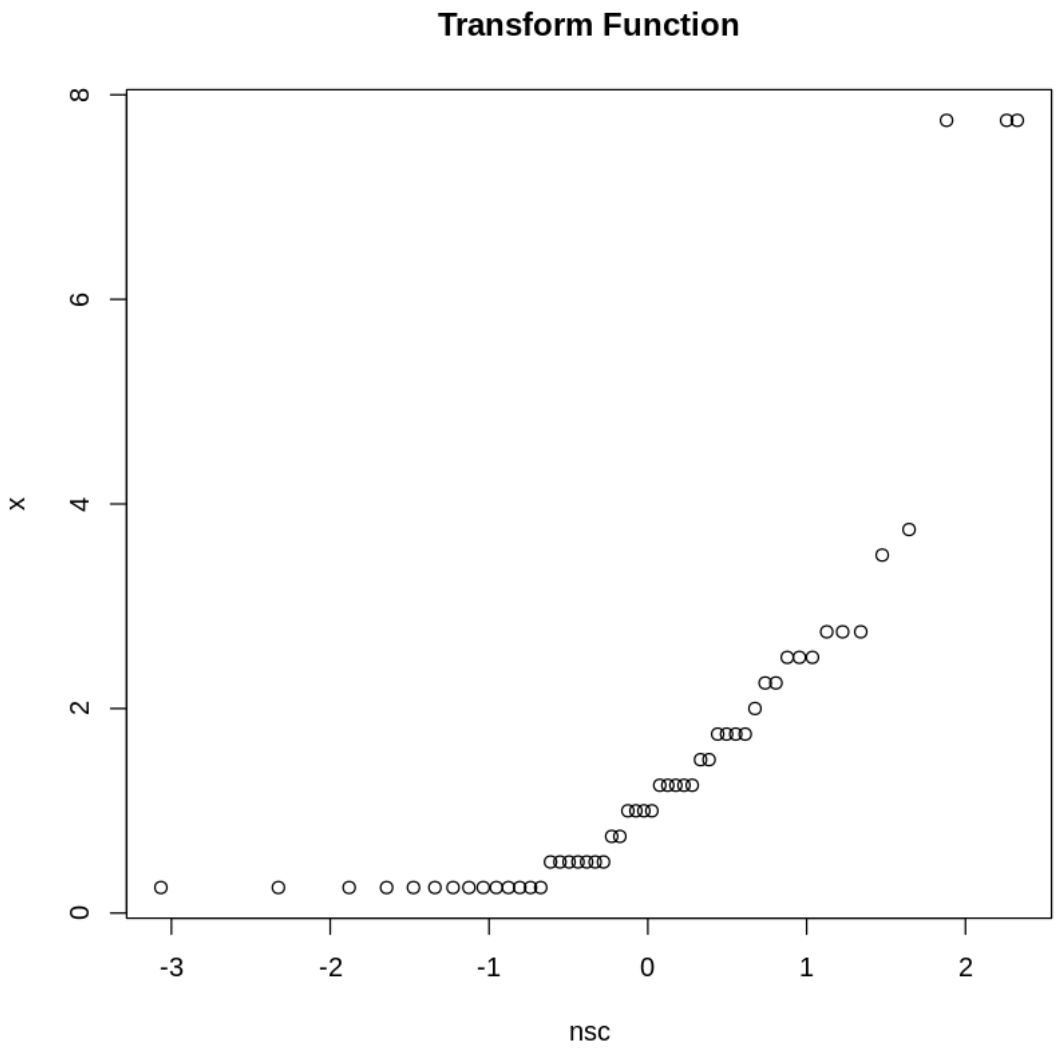

Transform Function
x
nsc
0
2
4
6
8
-3
-2
-1
0
1
2
3


Transform Function
x
nsc
0
2
4
6
8
-3
-2
-1
0
1
2




```
Error in Normalized_Pluv_SGS[, i]: subscript out of bounds
Traceback:
```


Transform Function
x
nsc
0
2
4
6
8
-3
-2
-1
0
1
2



Para obtener el promedio de la propiedad estadística que nos interesa usamos la función “apply”

[271]:
```
Pluv_SGS_nsim_mean <- apply(Pluv_SGS_nsim, 1, mean)
Pluv_SGS_nsim_sd <- apply(Pluv_SGS_nsim, 1, sd)
Pluv_SGS_nsim_quantiles <- apply(Pluv_SGS_nsim, 1, quantile)
Pluv_SGS_nsim_range <- Pluv_SGS_nsim_quantiles[5,] - Pluv_SGS_nsim_quantiles[1,]
```

[272]:
```
Pluv_SGS_nsim_mean_Stat <- Estadisticas(Pluv_SGS_nsim_mean)
Pluv_SGS_nsim_mean_Stat
```

A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	812.0000
minimos	Minimum	0.2500
cuantiles1	1st. Quartile	0.5000
medianas	Median	0.8615
medias	Mean	1.1953
cuantiles3	3rd. Quartile	1.6499
maximos	Maximum	5.7964
rangos	Rank	5.5464
rangosInt	Interquartile Rank	1.1499
varianzas	Variance	0.8694
desvs	Standard Deviation	0.9324
CVs	Variation Coeff.	0.7801
simetrias	Skewness	1.5051
curtosiss	Kurtosis	5.4419

[273]:
```
Pluv_SGS_nsim_sd_Stat <- Estadisticas(Pluv_SGS_nsim_sd)
Pluv_SGS_nsim_sd_Stat
```

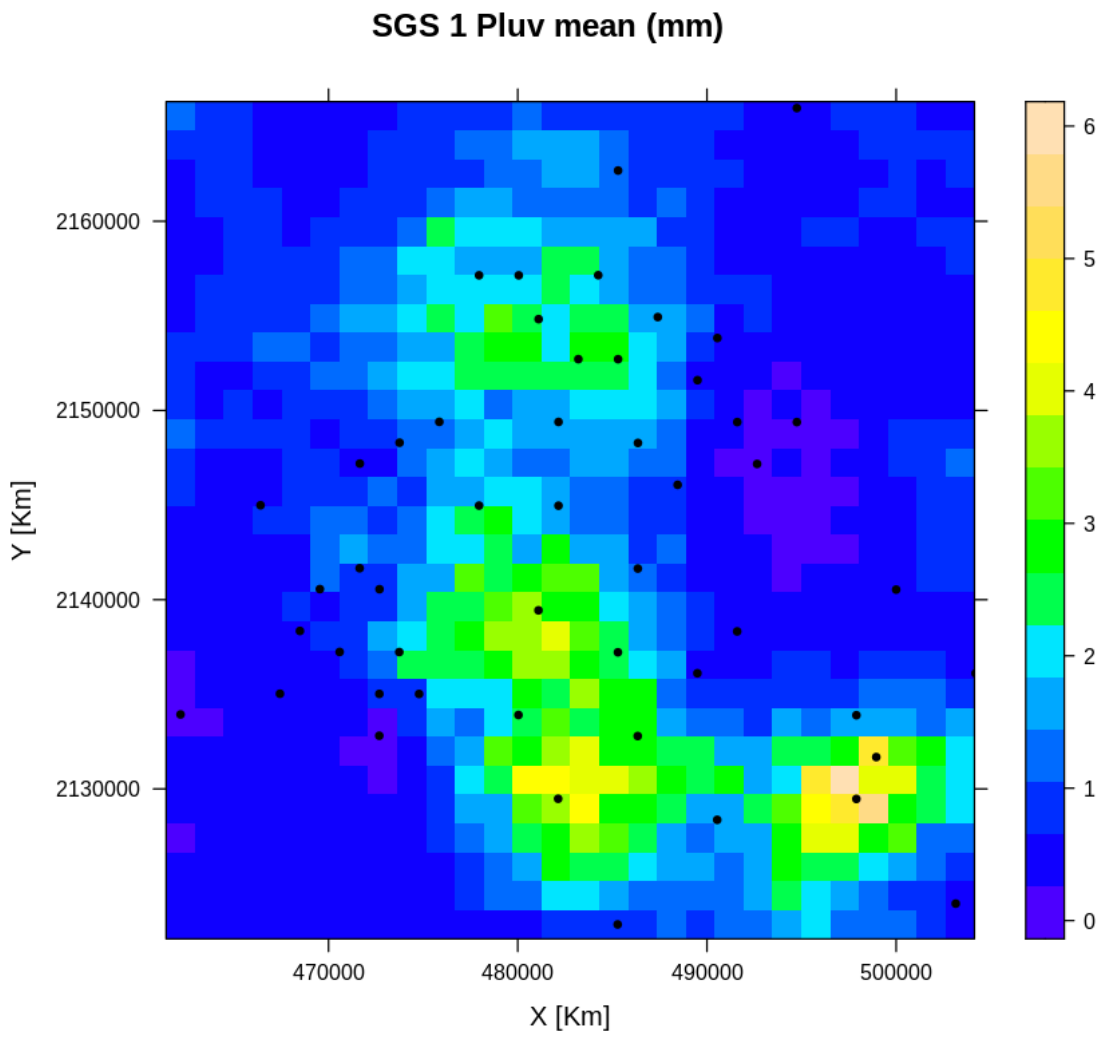
A data.frame: 14 × 2

	Statistics <chr>	Values <dbl>
muestras	n	812.0000
minimos	Minimum	0.0000
cuantiles1	1st. Quartile	0.3872
medianas	Median	0.6519
medias	Mean	0.7858
cuantiles3	3rd. Quartile	0.9233
maximos	Maximum	2.8635
rangos	Rank	2.8635
rangosInt	Interquartile Rank	0.5361
varianzas	Variance	0.3633
desvs	Standard Deviation	0.6027
CVs	Variation Coeff.	0.7670
simetrias	Skewness	1.3430
curtosiss	Kurtosis	4.3379

Para obtener el mapa usamos la función "spplot", los parámetros son los mismos, solo cambia el vector que nos interesa, en este caso es la media.

[274]:
```
DatosOrg <- as.data.frame(cbind(XCoord, YCoord, Pluv_mm))
colnames(DatosOrg) <- c("X", "Y", "V1")
coordinates(DatosOrg) <- ~X + Y
Salida <- as.data.frame(cbind(Normalized_Pluv_SGS[,1], Normalized_Pluv_SGS[,2],␣
 ↪Pluv_SGS_nsim_mean))
Salida1<-Salida
gridded(Salida) <- ~V1+V2
NameX="X [Km]"
```

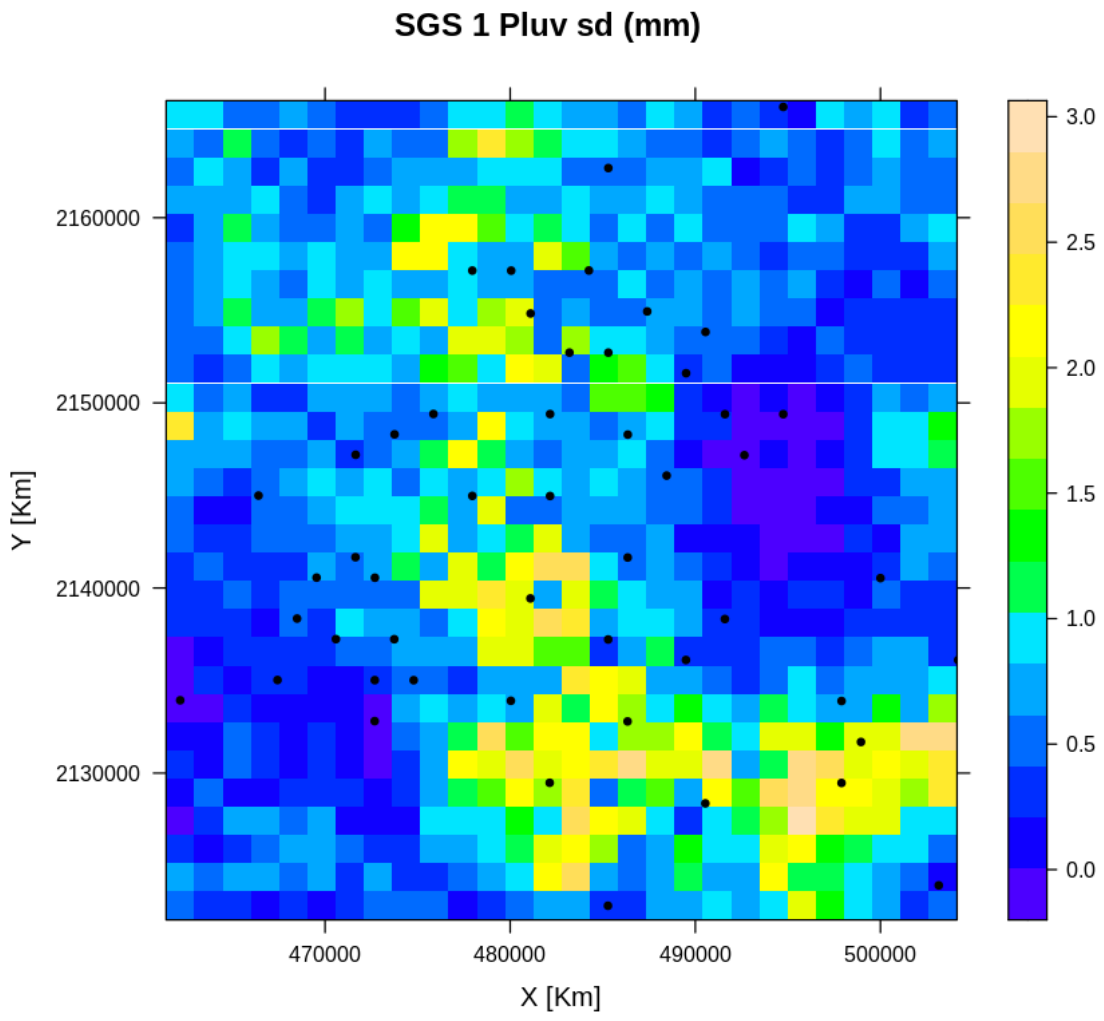
```
NameY="Y [Km]"
Titulo1="SGS 1 Pluv mean (mm)"
spplot(Salida, c("Pluv_SGS_nsim_mean"), main=Titulo1, xlab = NameX , ylab =␣
↪NameY , col.regions=topo.colors,
       do.log = TRUE,
       key.space=list(x = 0.1, y = 0.95, corner = c(0, 1)),
       scales=list(draw = TRUE),
       sp.layout = list("sp.points", DatosOrg, pch = 20, col = "black"))
```



Para obtener el mapa de la desviación estándar usamos las siguientes instrucciones:

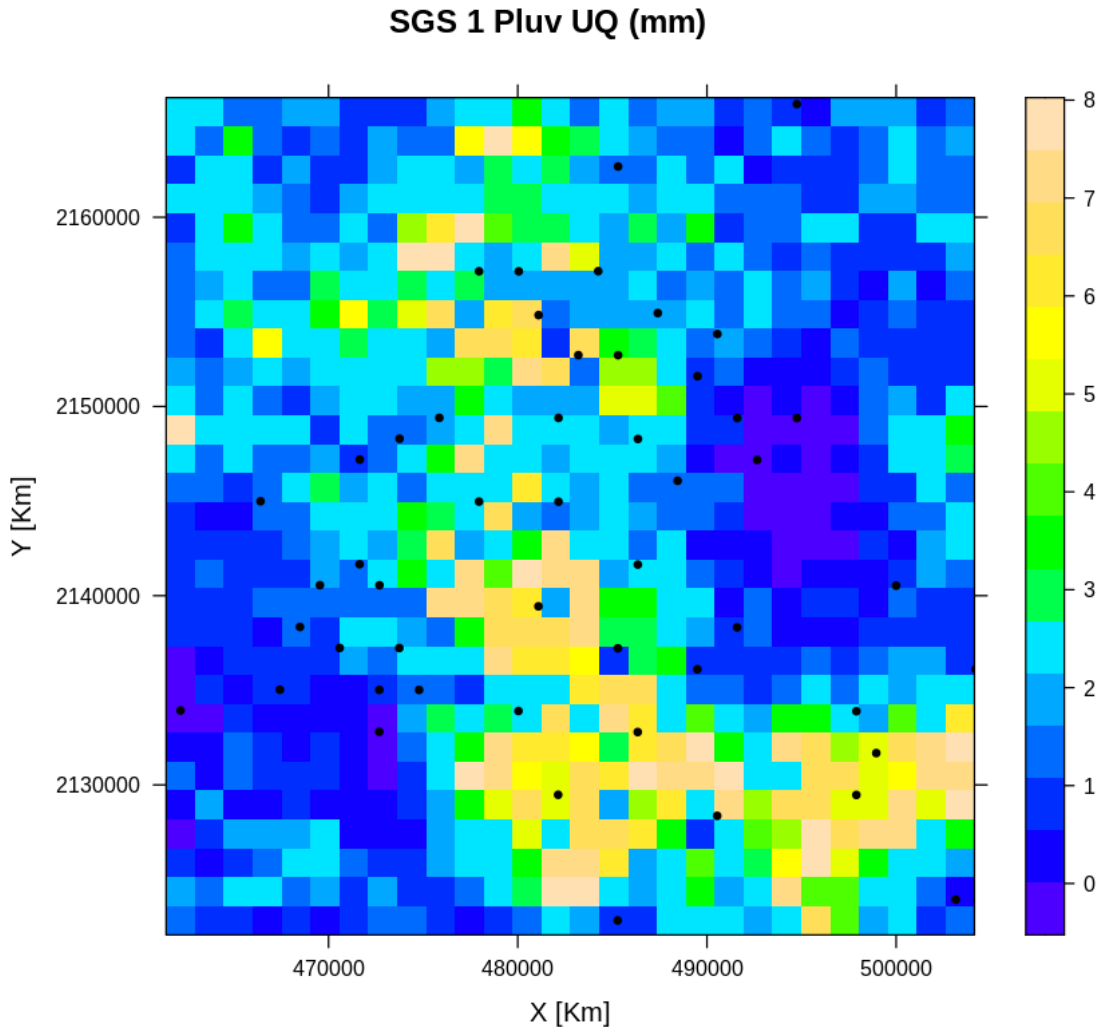
[275]:
```
DatosOrg <- as.data.frame(cbind(XCoord, YCoord, Pluv_mm))
colnames(DatosOrg) <- c("X", "Y", "V1")
```

```
coordinates(DatosOrg) <- ~X + Y
Salida <- as.data.frame(cbind(Normalized_Pluv_SGS[,1], Normalized_Pluv_SGS[,2],␣
 ↪Pluv_SGS_nsim_sd))
Salida1<-Salida
gridded(Salida) <- ~V1+V2
NameX="X [Km]"
NameY="Y [Km]"
Titulo1="SGS 1 Pluv sd (mm)"
spplot(Salida, c("Pluv_SGS_nsim_sd"), main=Titulo1, xlab = NameX , ylab = NameY␣
 ↪, col.regions=topo.colors,
       do.log = TRUE,
       key.space=list(x = 0.1, y = 0.95, corner = c(0, 1)),
       scales=list(draw = TRUE),
       sp.layout = list("sp.points", DatosOrg, pch = 20, col = "black"))
```



y del rango es el siguiente:

[276]:
```
DatosOrg <- as.data.frame(cbind(XCoord, YCoord, Pluv_mm))
colnames(DatosOrg) <- c("X", "Y", "V1")
coordinates(DatosOrg) <- ~X + Y
Salida <- as.data.frame(cbind(Normalized_Pluv_SGS[,1], Normalized_Pluv_SGS[,2],␣
↪Pluv_SGS_nsim_range))
Salida1<-Salida
gridded(Salida) <- ~V1+V2
NameX="X [Km]"
NameY="Y [Km]"
Titulo1="SGS 1 Pluv UQ (mm)"
spplot(Salida, c("Pluv_SGS_nsim_range"), main=Titulo1, xlab = NameX , ylab =␣
↪NameY , col.regions=topo.colors,
       do.log = TRUE,
       key.space=list(x = 0.1, y = 0.95, corner = c(0, 1)),
       scales=list(draw = TRUE),
       sp.layout = list("sp.points", DatosOrg, pch = 20, col = "black"))
```



Para obtener el resultado de las 10 simulaciones debemos promediarlas y después estimar el variograma y generar los gráficos de pixel igual que el caso de una simulación.

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

[277]:
```
save.image() #important line, this is for save the R workspace, this include␣
↪variables and results
```

7 Bibliografia

Diaz-Viera, Martin & Herrera-Zamarrón, Graciela Del Socorro & Valdes-Manzanilla, Arturo. (2009). A Linear Coregionalization Model For Spatial Rainfall Estimation In The Mexico City

Valley Combining Rain Gages Data And Meteorological Radar Images. Ingeniería hidráulica en México. XXIV. 63-90.

KRAJEWSKI, W.F. (1987) Cokriging radar-rainfall and rain gage data.J. Geophys. Res. Vol. 92, pp. 9571-9580.