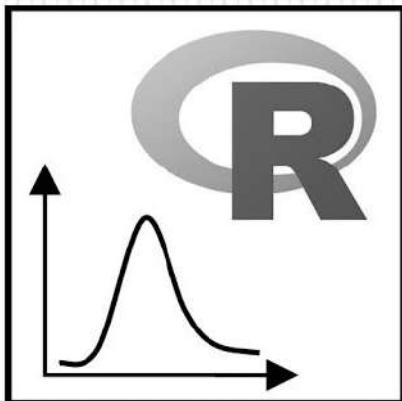


Lima, 10-11 marzo 2020

Análisis de series hidrológicas en R

Hydrological series analysis in R

Pedro Rau, PhD
Hidrólogo



✓ **Instructor:**

Pedro Rau, PhD

- Profesor a tiempo completo- Universidad de Ingeniería y Tecnología UTEC - Departamento de Ingeniería Ambiental e Investigador Principal del Centro de Investigación y Tecnología del Agua CITA
- Hidrólogo investigador en redes internacionales de colaboración científica (Francia, Ecuador, EEUU, Reino Unido)
- Revisor para revistas científicas indexadas internacionales.

Co-Instructores:

Víctor Rojas: Especialista en Data Science

Erick Claros: Especialista en Hidráulica y Recursos Hídricos

Froilan Rodas: Especialista en Hidrología y Recursos Hídricos

✓ **E.mail:** prau@utec.edu.pe

✓ **Sitio web:** <http://pedroraub.blogspot.com>

✓ **Repositorio del curso:** https://github.com/hydrocodes/Series_hidro_R

✓ **Requisitos recomendados:** Hidrología y Estadística básicos

Objetivos del curso

- Brindar las bases teóricas del estudio de una serie de tiempo hidrológica (variables: precipitación, temperatura y caudales).
- Punto de inicio para los interesados en el sistema R y su entorno estadístico y geoestadístico.
- Desarrollar capacidades para el empleo correcto del software libre en la hidrología.

Metodología « Hands-on »

- Bibliografía base
- Planteamiento de ejercicios
- Creación/tipeo/interpretación de códigos
*.R con semi comentarios
- Resolución de bugs
- Finalización de códigos *.R con comentarios completos al detalle (evaluado)

Evaluación

Clases teóricas + prácticas

- Asistencia:
16 hr presenciales
- Ejercicios

Total: 16 hrs lectivas

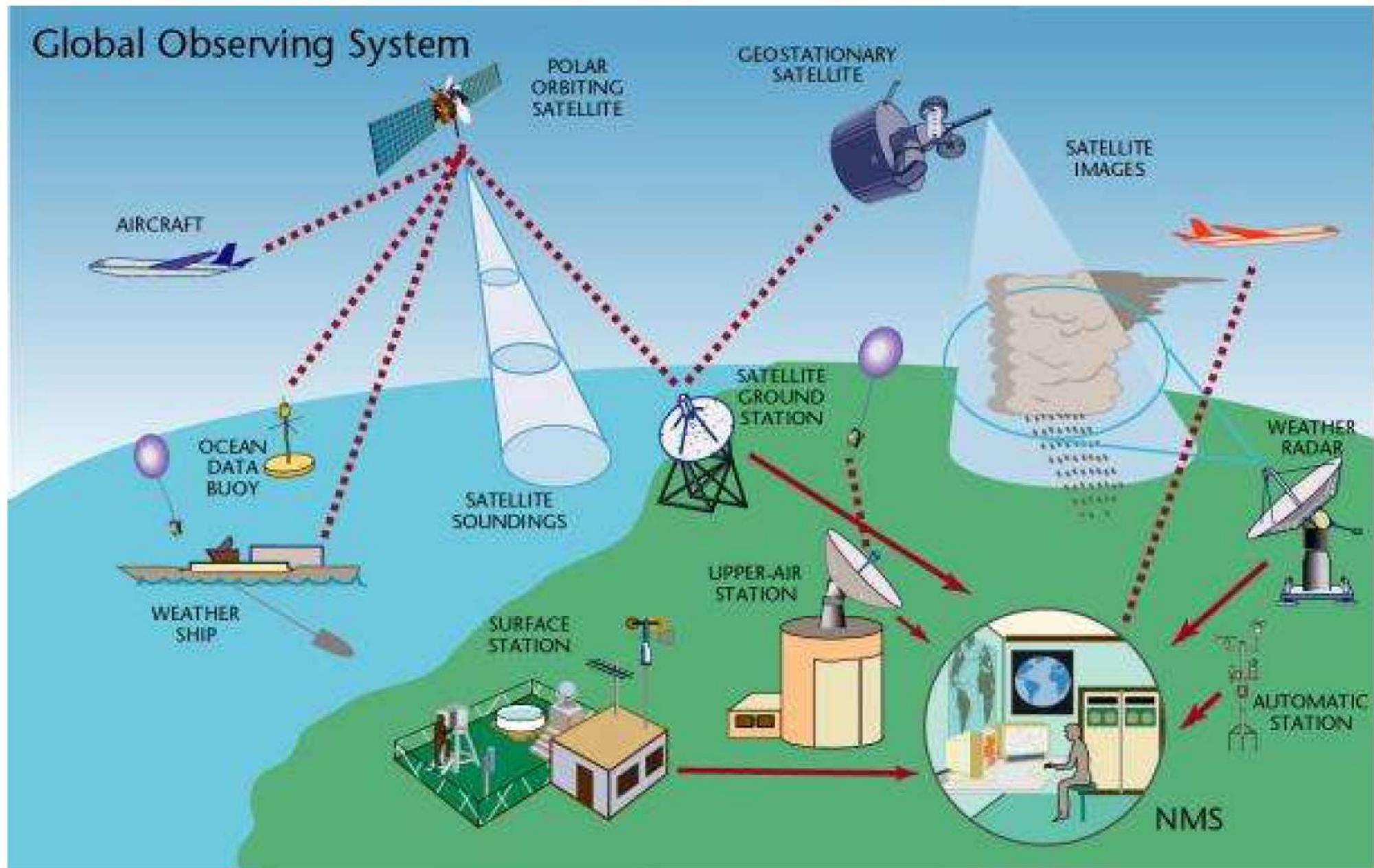
Temáticas

1. Introducción a las series de tiempo hidrológicas
2. Entorno R y Rstudio
3. Análisis exploratorio de datos
4. Tratamiento de datos
5. Introducción a las series espacio-temporales y el big data en hidrología
6. Aplicaciones geoespaciales en R

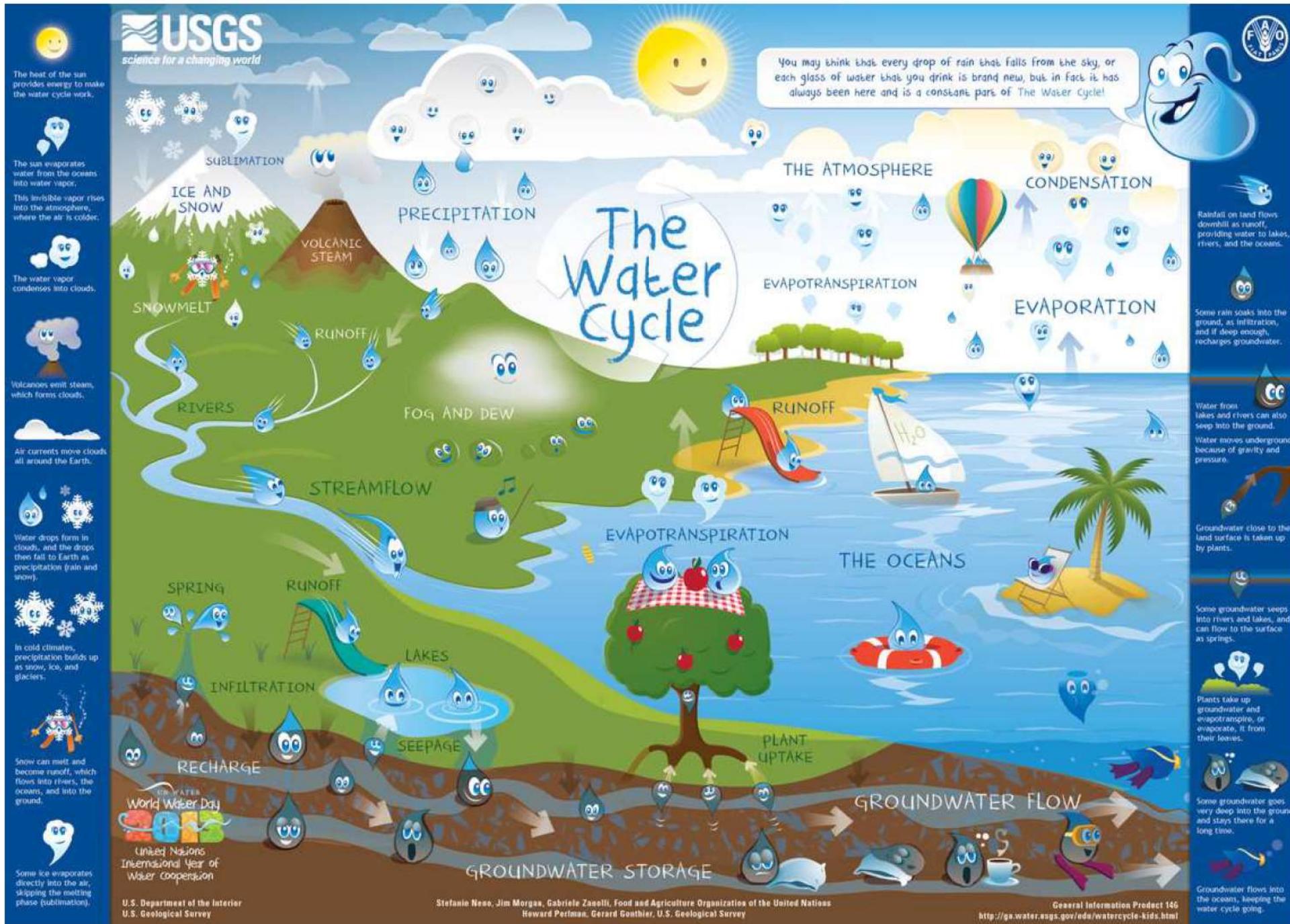
1. Introducción a las series de tiempo hidrológicas



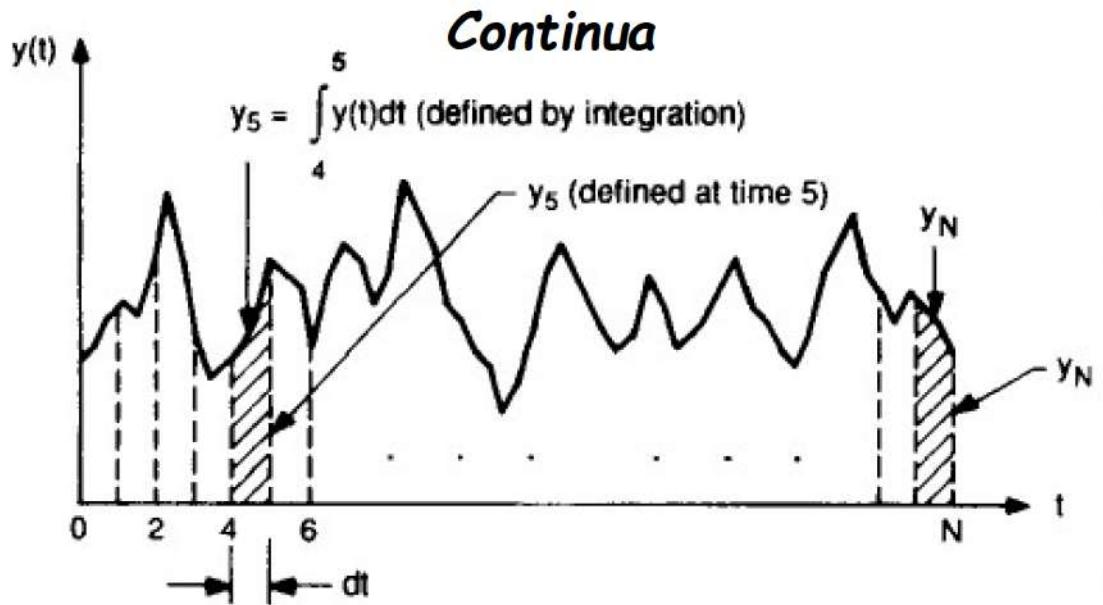
1.1 Sistema de observación global



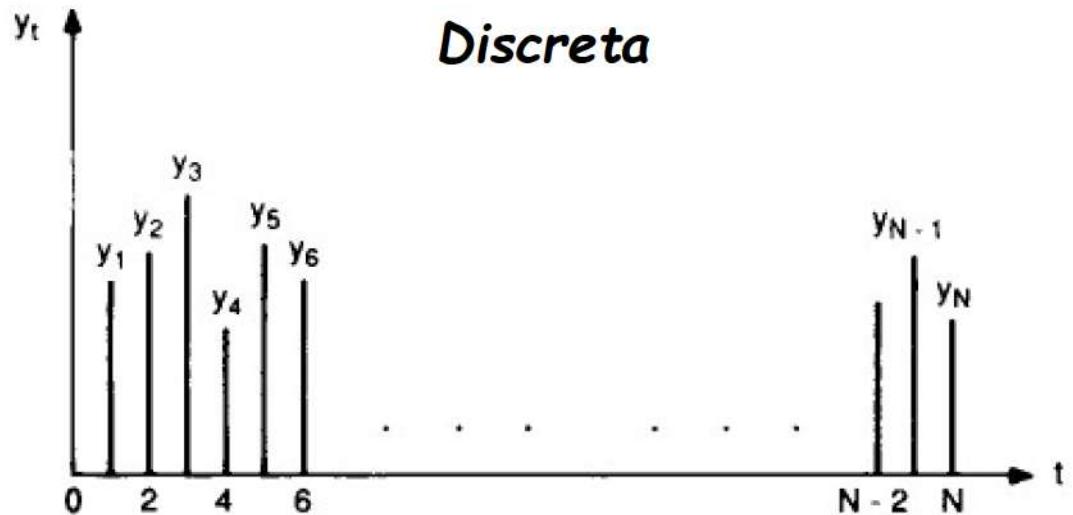
1.2 El ciclo hidrológico



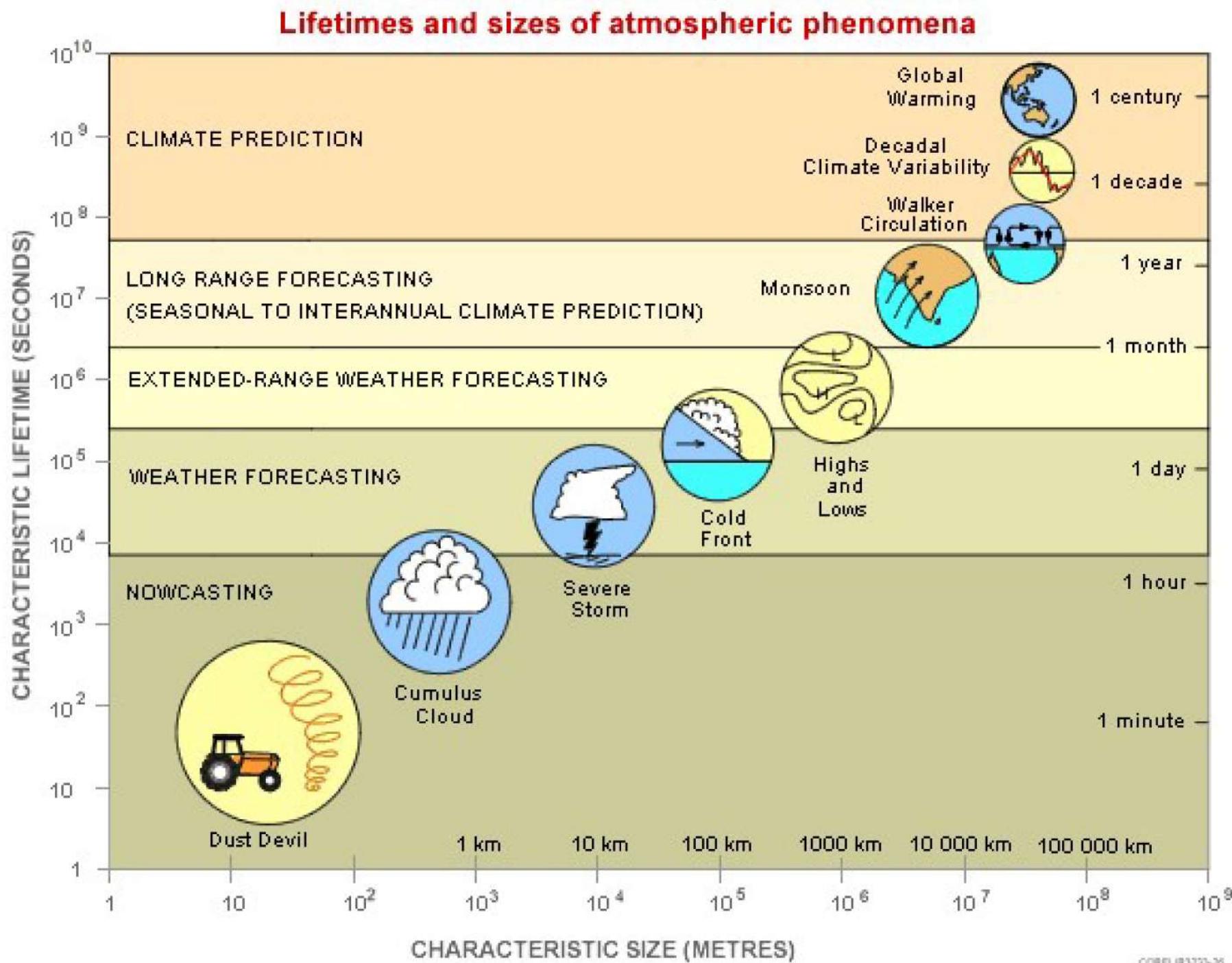
1.3 Tipos o categorías de series de tiempo



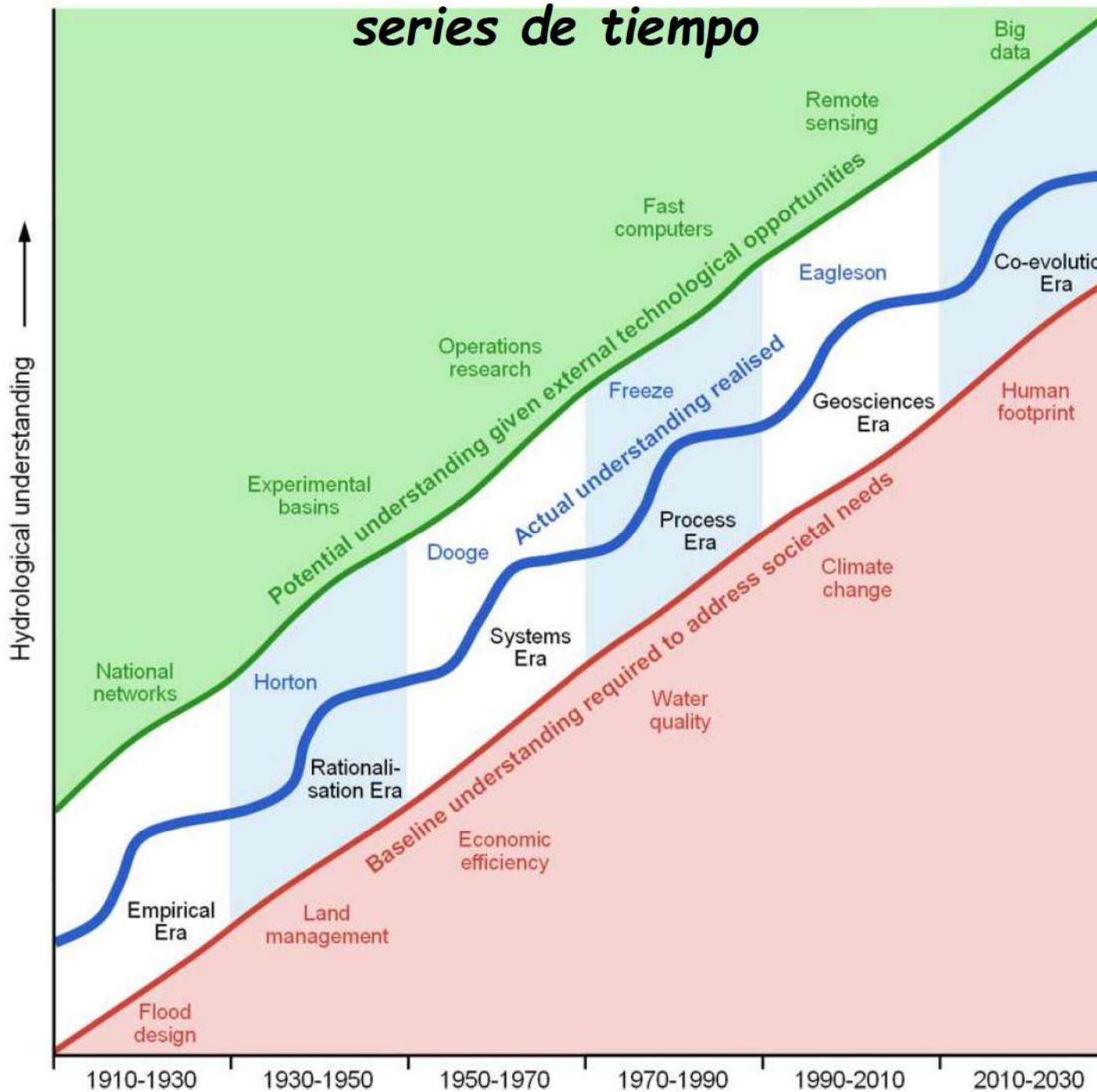
- ✓ Unicas
- ✓ Multiples
- ✓ Independientes, no correlacionadas; autocorrelacionados o dependientes
- ✓ Intermitentes
- ✓ De conteo
- ✓ Regulares o irregulares en espacios de tiempo
- ✓ Estacionarias (sin tendencia, ni salto ni ciclicidad / periodicidad) y no-estacionarias



1.4 Escalas de tiempo espacio-temporales



1.5 Evolución de la hidrología y el análisis de sus series de tiempo

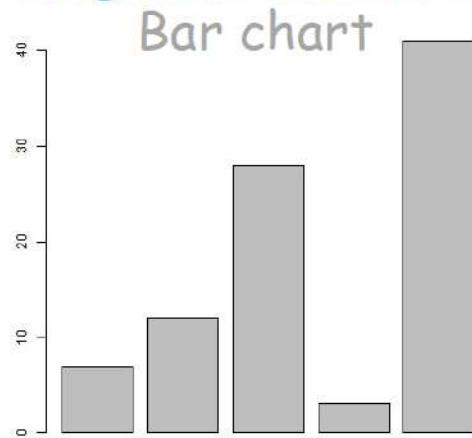


Sivapalan & Blöschl (2017)

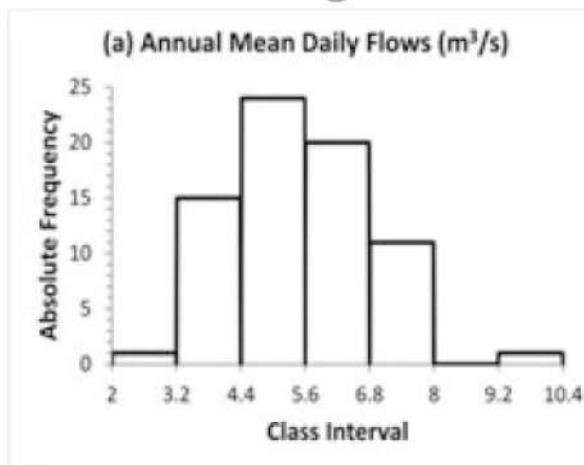
1.6 Análisis preliminar de datos hidrológicos

a. Representación gráfica

Diagrama de barras
Bar chart



Histograma
Histogram



Polígonos de frecuencia
Frequency Polygon

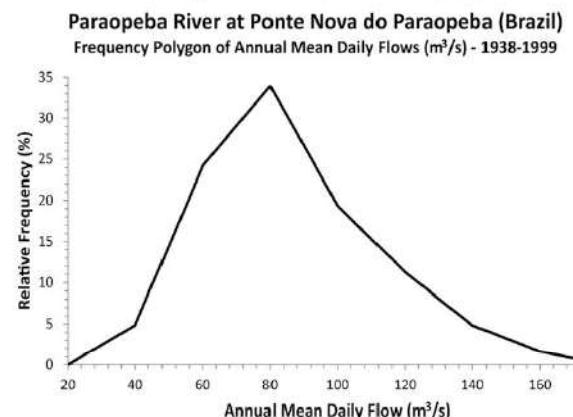
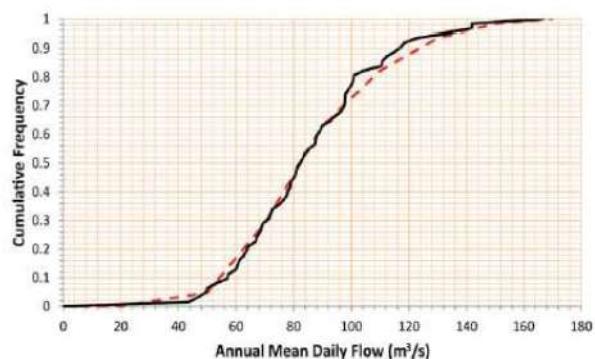


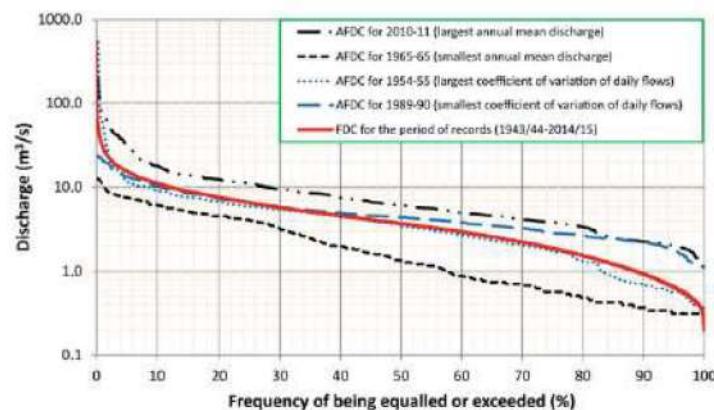
Diagrama de puntos
Dot chart



Frecuencia relativa acumulada
Cumulative Relative Frequency Diagram



Curvas de duración
Duration curves



b. Estadísticos descriptivos

c. Métodos exploratorios

Parámetro de la población

1. Punto medio

Media aritmética

$$\mu = E(X) = \int_{-\infty}^{\infty} xf(x) dx$$

Estadística de la muestra

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

Mediana

$$x \text{ tal que } F(x) = 0.5$$

Valor de la información en el 50o. percentil

Media geométrica

$$\text{antilog } [E(\log x)]$$

$$\left(\prod_{i=1}^n x_i \right)^{1/n}$$

2. Variabilidad

Varianza

$$\sigma^2 = E[(x - \mu)^2]$$

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Desviación estándar

$$\sigma = \{E[(x - \mu)^2]\}^{1/2}$$

$$s = \left[\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \right]^{1/2}$$

Coeficiente de variación

$$CV = \frac{\sigma}{\mu}$$

$$CV = \frac{s}{\bar{x}}$$

3. Simetría

Coeficiente de asimetría (oblicuidad)

$$\gamma = \frac{E[(x - \mu)^3]}{\sigma^3}$$

$$C_s = \frac{n \sum_{i=1}^n (x_i - \bar{x})^3}{(n-1)(n-2)s^3}$$

Chow (1994)

Diagrama de cajas

Box plot

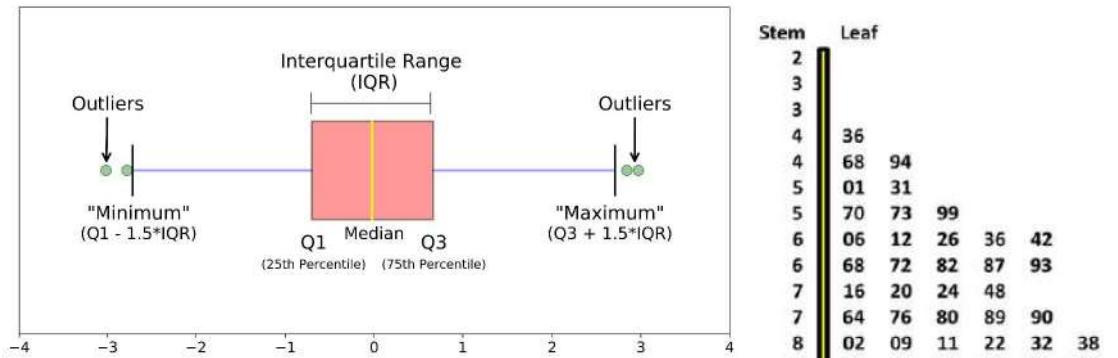
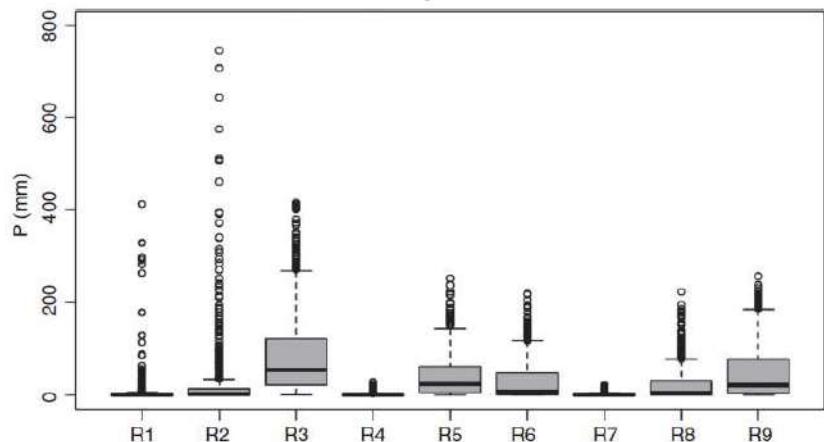


Diagrama de
tallos y hojas
Stem-and-leaf
Diagram

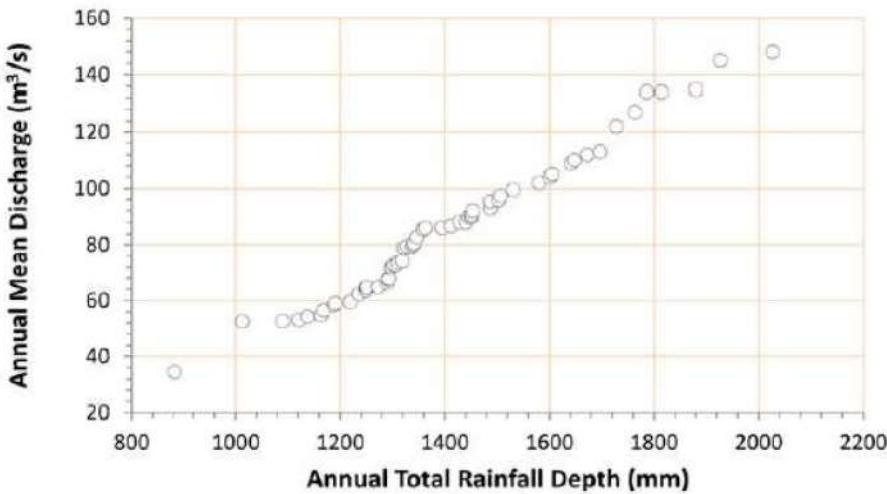
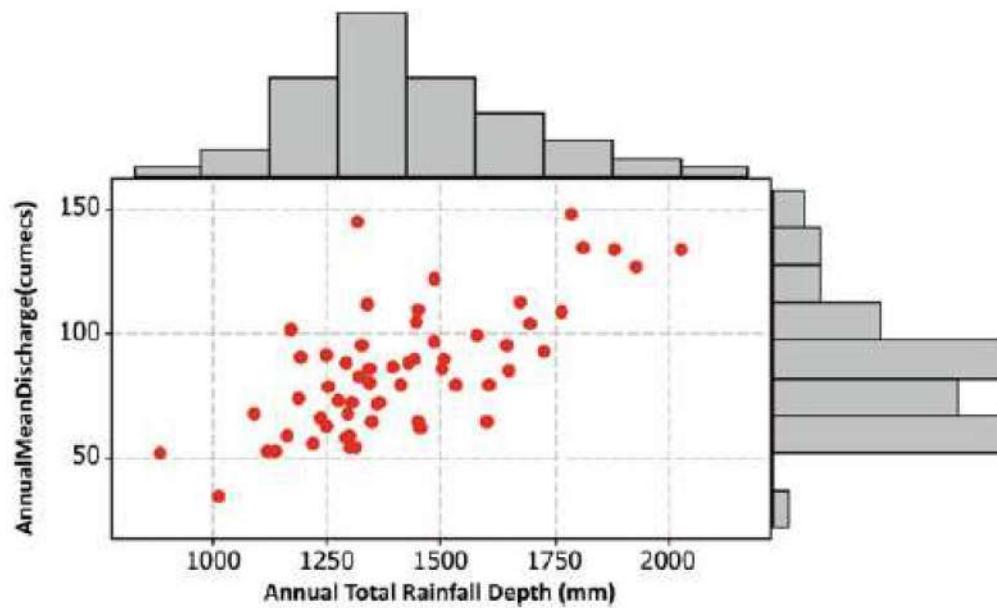
Rau et al (2017)

Naghettini (2017)

d. Asociación de datos

Gráfico de dispersión
Scatter plot

Diagrama Cuantil-Cuantil
Empirical Quantile-
Quantile Diagram
Q-Q Plot



Naghettini (2017)

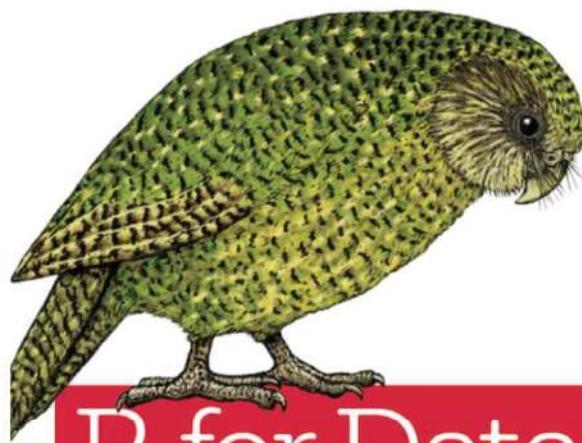


2. *Entorno R y Rstudio*



Programming language

O'REILLY®



R for Data Science

VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA

Hadley Wickham &
Garrett Grolemund

<https://r4ds.had.co.nz/>

<https://github.com/jrnold/r4ds-exercise-solutions/blob/master/README.md>

The Art of Data Science

A Guide for Anyone Who Works with Data



Roger D. Peng & Elizabeth Matsui

<https://bookdown.org/rdpeng/artofdatascience/>

<https://github.com/waldronlab/The-Art-of-Data-Science/blob/master/README.md>

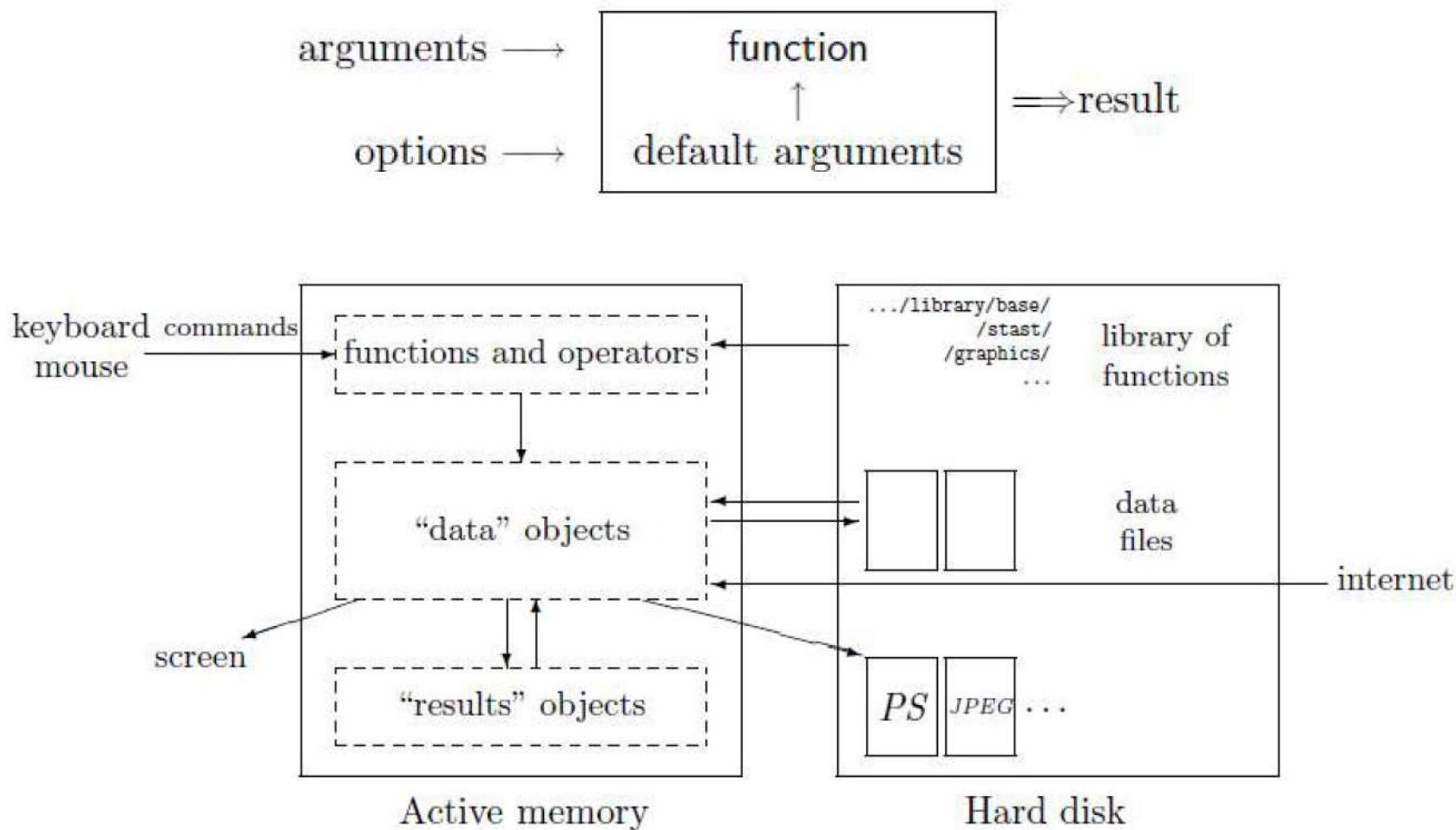
R-3.6.2 for Windows (32/64 bit) - Diciembre 2019



<https://cran.r-project.org/bin/windows/base/>

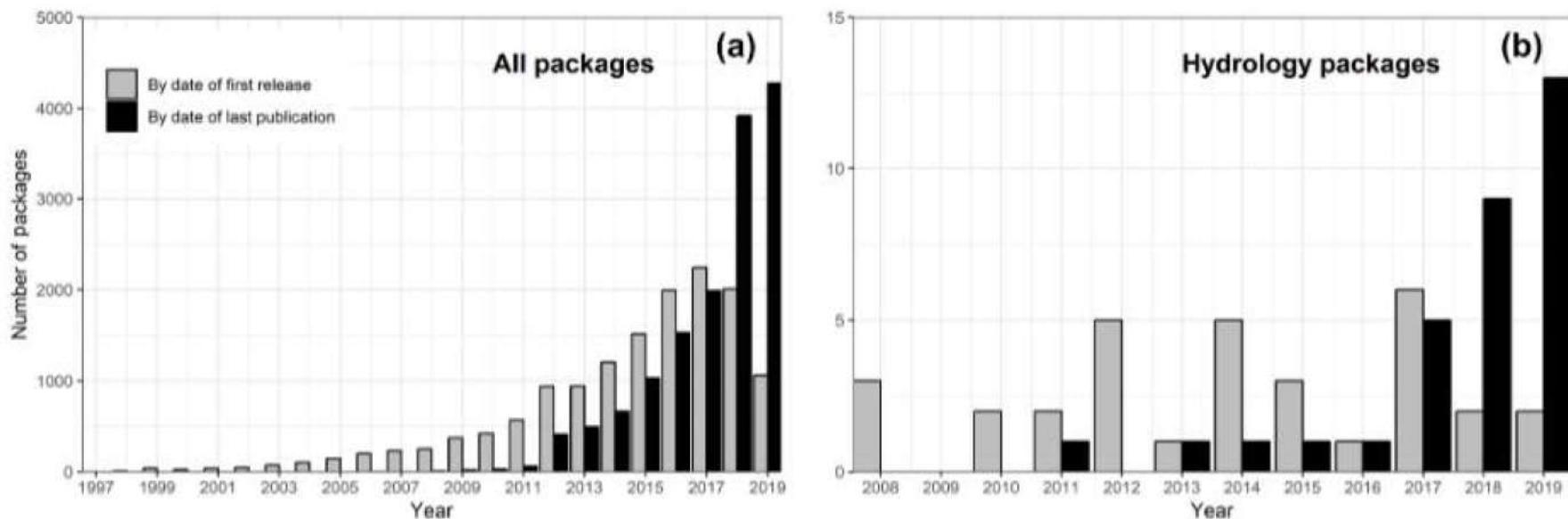
- ✓ Sistema para el análisis estadístico y graficación (Ihaka y Gentleman, 1996)
- ✓ Lenguaje interpretado, no es un lenguaje compilado
- ✓ Fácil e intuitivo, estrictamente « no es un lenguaje de programación »

Paradis (2002)

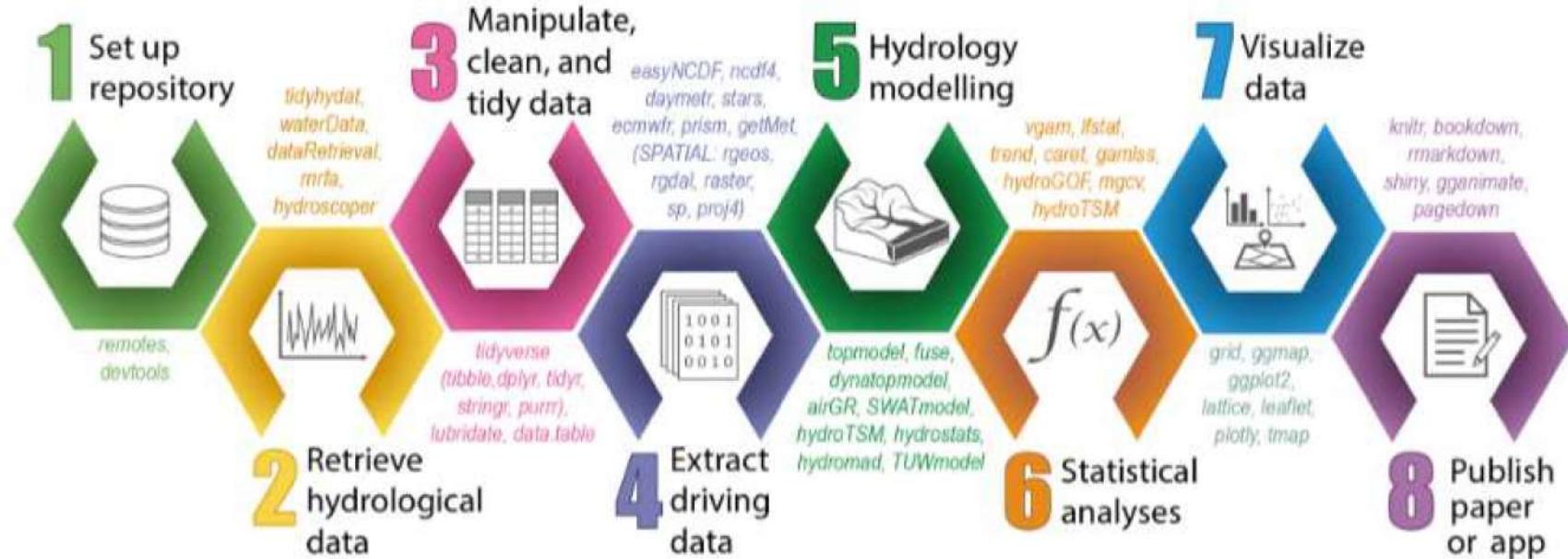


The Comprehensive R Archive Net-work (CRAN)

<https://cran.r-project.org>



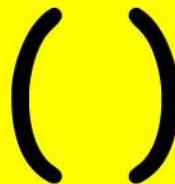
Tipico Flujo de trabajo en R en hidrologia





« Clases » de objetos básicos o “atomic” en R:

- numeric (real numbers)
- integer
- complex
- logical (True/False)



« Estructuras o tipos » de objetos en R para representar a los datos

object	modes	several modes possible in the same object?	
vector	numeric, character, complex <i>or</i> logical	No	<code>c()</code>
factor	numeric <i>or</i> character	No	<code>factor()</code>
array	numeric, character, complex <i>or</i> logical	No	<code>array()</code>
matrix	numeric, character, complex <i>or</i> logical	No	<code>matrix()</code>
data frame	numeric, character, complex <i>or</i> logical	Yes	<code>data.frame()</code>
ts	numeric, character, complex <i>or</i> logical	No	<code>ts()</code>
list	numeric, character, complex, logical, function, expression, ...	Yes	<code>list()</code>



Convirtiendo clases de objetos

Conversion to	Function	Rules
numeric	as.numeric	FALSE → 0 TRUE → 1 "1", "2", ... → 1, 2, ... "A", ... → NA
logical	as.logical	0 → FALSE other numbers → TRUE "FALSE", "F" → FALSE "TRUE", "T" → TRUE other characters → NA
character	as.character	1, 2, ... → "1", "2", ... FALSE → "FALSE" TRUE → "TRUE"

Funciones de probabilidad

Distribución/función	función
Gausse (normal)	rnorm(n, mean=0, sd=1)
exponencial	rexp(n, rate=1)
gamma	rgamma(n, shape, scale=1)
Poisson	rpois(n, lambda)
Weibull	rweibull(n, shape, scale=1)
Cauchy	rcauchy(n, location=0, scale=1)
beta	rbeta(n, shape1, shape2)
'Student' (<i>t</i>)	rt(n, df)
Fisher-Snedecor (<i>F</i>)	rf(n, df1, df2)
Pearson (χ^2)	rchisq(n, df)
binomial	rbinom(n, size, prob)
geométrica	rgeom(n, prob)
hypergeométrica	rhyper(nn, m, n, k)
logística	rlogis(n, location=0, scale=1)
lognormal	rlnorm(n, meanlog=0, sdlog=1)
binomial negativa	rnbnom(n, size, prob)
uniforme	runif(n, min=0, max=1)
Estadístico de Wilcoxon's	rwilcox(nn, m, n), rsignrank(nn, n)

Operadores en R

Operators				
Arithmetic		Comparison		Logical
+	addition	<	lesser than	! x logical NOT
-	subtraction	>	greater than	x & y logical AND
*	multiplication	<=	lesser than or equal to	x && y id.
/	division	>=	greater than or equal to	x y logical OR
^	power	==	equal	x y id.
%%	modulo	!=	different	xor(x, y) exclusive OR
%%/	integer division			

RStudio Desktop 1.2.5033

<https://rstudio.com/products/rstudio/download/#download>



Entorno de desarrollo integrado para el lenguaje de programación R

The screenshot shows the RStudio desktop application interface. On the left, the 'Editor de código' (Code Editor) contains R script code for extracting data from a CMIP5 file and plotting it. Below it, the 'Consola de resultados' (Results Console) displays the output of the R commands, including the resulting data frame 'df'. In the center, the 'Area de objetos e historial' (Environment and History) shows the global environment with various objects like 'vals', 'Variab', 'Variables', and 'xy'. At the bottom right, a 'Figuras y archivos' (Figures and Archives) panel displays a time-series plot of 'Tmin (°C/d)' over years from 2005 to 2030.

Editor de código
Consola de scripts

```
# Extracción de una variable del CMIP5 (o cualquier archivo .NC)
library(raster)

# Comando brick lee todas las capas contenidas en el archivo nc, Ejm: pr:precipitacion
b <- brick('C:/1_PHESE_SU2E/3_SIG/ccmodels/hadgem2-es/tasmin_Amon_HadGEM2-es_rcp45_r1i1p1_20c001.nc')

# Asignando indices
idx <- setZ(b)

# Indicar coordenadas y la latitud
coords <- c(x=c(-85.0, -11.8), y=c(1.0, 2.0), z=c(1.0, 1.0), lat=y, lon=x, depth=z)
valo <- extract(b, coords, df=TRUE, values=TRUE)

# Filtrar fecha y guardar en un archivo dataframe
df <- data.frame(valo[, 1], valo[, 2])
rownames(df) <- NULL
names(df) <- c('date', 'value')

# Dar un vistazo al archivo
head(df)
```

Consola R
Consola de resultados

```
> # Fijar fechas y datos en un solo archivo dataframe
> df <- data.frame(idx, t(vals)[-1,])
> rownames(df) <- NULL
> names(df) <- c('date', 'value')
>
> # Dar un vistazo al archivo
> head(df)
  date
1 2005-12-16 14.42845
2 2006-01-15 16.42248
3 2006-02-14 16.47473
4 2006-03-16 16.47473
5 2006-04-14 16.47473
6 2006-05-15 13.08361
>
> # Plotear la serie
> plot(df, type="l", xlab="Años", ylab="Tmin (°C/d)")
```

Area de objetos e historial

Figuras y archivos

A line plot titled 'Figuras y archivos' showing the minimum temperature (Tmin) in degrees Celsius per day (°C/d) from 2005 to 2030. The x-axis is labeled 'Años' (Years) and ranges from 2005 to 2030. The y-axis is labeled 'Tmin (°C/d)' and ranges from 10 to 18. The plot shows a highly seasonal pattern with significant annual fluctuations, where temperatures drop sharply in winter and rise in summer.

A. Objetos, estructuras y control en el entorno Rstudio

Ejercicio A.1

En la consola de resultados, efectuar las siguientes operaciones:

- 1) $3^2 - 5 * 9 * (25 - 18)$
- 2) Crear una serie consecutiva del 1 al 8
- 3) Crear una serie consecutiva de 8 letras comenzando por "a"
- 4) Asignar la operación $8 * 5^2$ a un objeto de nombre *value*
- 5) Visualizar el objeto *value* creado
- 6) Crear el objeto *p*, almacenando una secuencia desde 5 hasta 15 de 2 en 2
- 7) Visualizar *p*
- 8) Almacenar los siguientes valores de lluvia anual: 20, 21.2, 49, 10.5, 15.1, 19, 31 y 20.2 en el objeto *rain* de tipo vector.
- 9) Visualizar *rain*
- 10) Obtener el 5to elemento de *rain*
- 11) Almacenar los siguientes valores anuales de índice SOI El Niño: 0.2, 0.4, 1.5, -0.1, 0.15, 0.3, 0.9 y 1.1 en el objeto *soi* de tipo vector.
- 12) Visualizar *soi*.

Respuestas A.1

```
> 3^2-5*9*(25-18)
[1] -306
> 1:8
[1] 1 2 3 4 5 6 7 8
> letters[1:8]
[1] "a" "b" "c" "d" "e" "f" "g" "h"
> value<-8*5^2 #Asignamos una operación al objeto value
> value #Escribimos el objeto "value"
[1] 200
> p<-seq(from=5, to=15, by=2) #secuencia desde 5 hasta 15 de 2 en 2
> print (p)
> rain<-c(20, 21.2, 49, 10.5, 15.1, 19, 31, 20.2) #Uso de función c(combine)
asignar datos al objeto rain
> rain #Escribimos el objeto "rain"
[1] 20.0 21.2 49.0 10.5 15.1 19.0 31.0 20.2
> rain[5] #Obtenemos el 5to elemento del objeto rain
[1] 15.1
> soi<-c(0.2, 0.4, 1.5, -0.1, 0.15, 0.3, 0.9, 1.1) # Ejemplo de valores del índice
SOI para El Niño
> soi #Escribimos el objeto "soi"
[1] 0.20 0.40 1.50 -0.10 0.15 0.30 0.90 1.10
```

<code>plot(x)</code>	plot of the values of <code>x</code> (on the <code>y</code> -axis) ordered on the <code>x</code> -axis
<code>plot(x, y)</code>	bivariate plot of <code>x</code> (on the <code>x</code> -axis) and <code>y</code> (on the <code>y</code> -axis)
<code>sunflowerplot(x, y)</code>	id. but the points with similar coordinates are drawn as a flower which petal number represents the number of points
<code>pie(x)</code>	circular pie-chart
<code>boxplot(x)</code>	"box-and-whiskers" plot
<code>stripchart(x)</code>	plot of the values of <code>x</code> on a line (an alternative to <code>boxplot()</code> for small sample sizes)
<code>coplot(x~y z)</code>	bivariate plot of <code>x</code> and <code>y</code> for each value (or interval of values) of <code>z</code>
<code>interaction.plot(f1, f2, y)</code>	if <code>f1</code> and <code>f2</code> are factors, plots the means of <code>y</code> (on the <code>y</code> -axis) with respect to the values of <code>f1</code> (on the <code>x</code> -axis) and of <code>f2</code> (different curves); the option <code>fun</code> allows to choose the summary statistic of <code>y</code> (by default <code>fun=mean</code>)
<code>matplot(x,y)</code>	bivariate plot of the first column of <code>x</code> vs. the first one of <code>y</code> , the second one of <code>x</code> vs. the second one of <code>y</code> , etc.
<code>dotchart(x)</code>	if <code>x</code> is a data frame, plots a Cleveland dot plot (stacked plots line-by-line and column-by-column)
<code>fourfoldplot(x)</code>	visualizes, with quarters of circles, the association between two dichotomous variables for different populations (<code>x</code> must be an array with <code>dim=c(2, 2, k)</code> , or a matrix with <code>dim=c(2, 2)</code> if <code>k = 1</code>)
<code>assocplot(x)</code>	Cohen-Friendly graph showing the deviations from independence of rows and columns in a two dimensional contingency table
<code>mosaicplot(x)</code>	'mosaic' graph of the residuals from a log-linear regression of a contingency table
<code>pairs(x)</code>	if <code>x</code> is a matrix or a data frame, draws all possible bivariate plots between the columns of <code>x</code>
<code>plot.ts(x)</code>	if <code>x</code> is an object of class "ts", plot of <code>x</code> with respect to time, <code>x</code> may be multivariate but the series must have the same frequency and dates
<code>ts.plot(x)</code>	id. but if <code>x</code> is multivariate the series may have different dates and must have the same frequency
<code>hist(x)</code>	histogram of the frequencies of <code>x</code>
<code>barplot(x)</code>	histogram of the values of <code>x</code>
<code>qqnorm(x)</code>	quantiles of <code>x</code> with respect to the values expected under a normal law
<code>qqplot(x, y)</code>	quantiles of <code>y</code> with respect to the quantiles of <code>x</code>
<code>contour(x, y, z)</code>	contour plot (data are interpolated to draw the curves), <code>x</code> and <code>y</code> must be vectors and <code>z</code> must be a matrix so that <code>dim(z)=c(length(x), length(y))</code> (<code>x</code> and <code>y</code> may be omitted)
<code>filled.contour (x, y, z)</code>	id. but the areas between the contours are coloured, and a legend of the colours is drawn as well
<code>image(x, y, z)</code>	id. but the actual data are represented with colours
<code>persp(x, y, z)</code>	id. but in perspective
<code>stars(x)</code>	if <code>x</code> is a matrix or a data frame, draws a graph with segments or a star where each row of <code>x</code> is represented by a star and the columns are the lengths of the segments
<code>symbols(x, y, ...)</code>	draws, at the coordinates given by <code>x</code> and <code>y</code> , symbols (circles, squares, rectangles, stars, thermometers or "boxplots") which sizes, colours, etc, are specified by supplementary arguments
<code>termplot(mod.obj)</code>	plot of the (partial) effects of a regression model (<code>mod.obj</code>)

Ploteando gráficos

`add=FALSE` if TRUE superposes the plot on the previous one (if it exists)

`axes=TRUE` if FALSE does not draw the axes and the box

`type="p"` species the type of plot, "`p`": points, "`|`": lines, "`b`": points connected by lines, "`o`": id. but the lines are over the points, "`h`": vertical lines, "`s`": steps, the data are represented by the top of the vertical lines, "`S`": id. But the data are represented by the bottom of the vertical lines

`xlim=`, `ylim=` species the lower and upper limits of the axes, for example with `xlim=c(1, 10)` or `xlim=range(x)`

`xlab=`, `ylab=` annotates the axes, must be variables of mode character

`main=` main title, must be a variable of mode character

`sub=` sub-title (written in a smaller font)

Ejercicio A.2

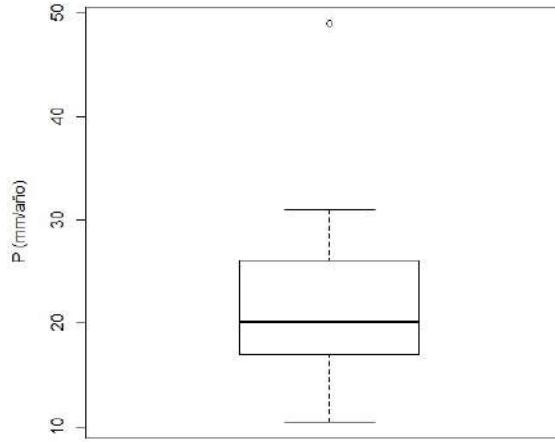
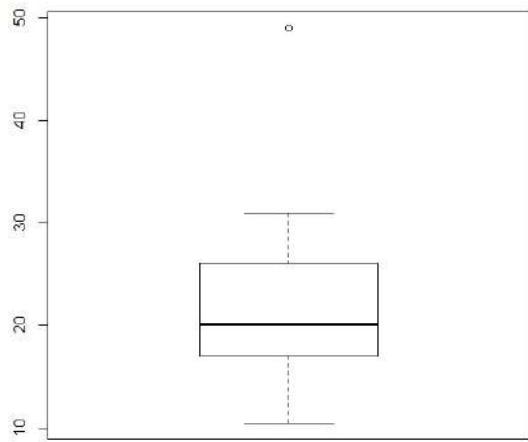
En la consola de resultados, efectuar las siguientes operaciones:

- 1) Obtener la media, mediana, desviación estándar, varianza del objeto rain (del ejercicio A.1)
- 2) Visualizar un resumen de los estadísticos notables
- 3) Plotear un diagrama de cajas con los valores del objeto rain. interpretar
- 4) Plotear lo anterior con una etiqueta en el eje vertical indicando: P (mm/año)

Respuestas A.2

```
> mean(rain)
[1] 23.25
> median(rain)
[1] 20.1
> sd(rain)
[1] 11.91781
> var(rain)
[1] 142.0343
> summary(rain)
   Min. 1st Qu. Median Mean 3rd Qu. Max.
10.50 18.02 20.10 23.25 23.65 49.00
```

```
> boxplot(rain) #Grafico de cajas
> boxplot(rain, ylab="P (mm/año)") #Grafico de cajas y etiqueta
```



Ejercicio A.3

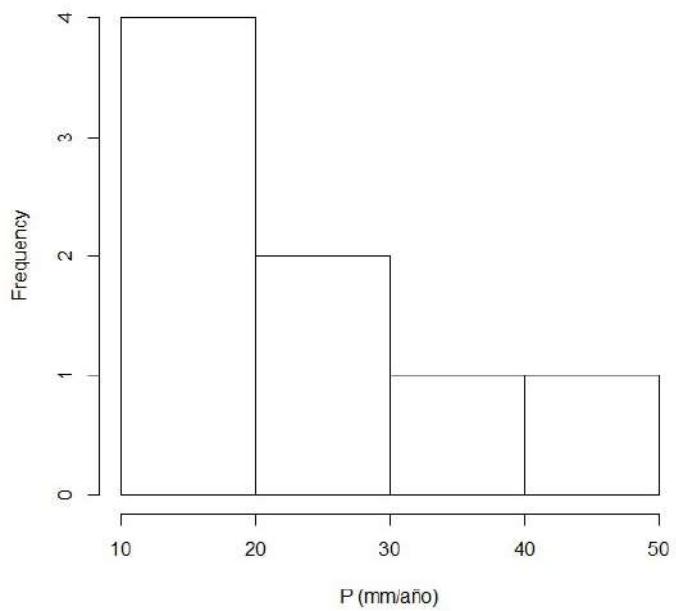
En la consola de resultados, efectuar las siguientes operaciones:

- 1) Plotear un histograma de frecuencias para rain, empleando la frecuencia.
- 2) Plotear un histograma de frecuencias para rain, empleando la densidad
- 3) Agregar al histograma una curva de distribución normal en color rojo con la media y desviación estimados anteriormente
- 4) Agregar en un solo grafico (1filas x 2 columnas), el grafico de cajas anterior y el histograma
- 5) Separar los graficos anteriores
- 6) Calcular la covarianza entre la lluvia y el índice soi
- 7) Calcular el coeficiente de correlación entre la lluvia y el índice soi
- 8) Calcular la ecuación de regresión lineal entre la lluvia y soi
- 9) Plotear un grafico de dispersión entre la lluvia y el índice soi
- 10) Agregar la línea de tendencia al grafico anterior.

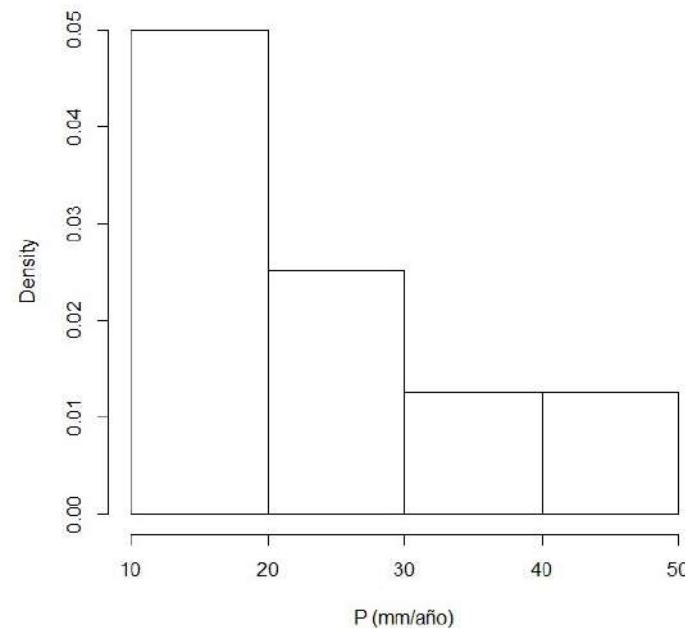
Respuestas A.3

```
> hist(rain, main="Precipitacion anual", xlab="P (mm/año)", freq=F) # Grafico de histograma, F(False) no usar frecuencia
> curve(dnorm(x, mean(rain, na.rm = T), sd(rain, na.rm = T)), add = TRUE, col="red") # Agregar curva distribucion r normal en color rojo
> split.screen(c(1,2)) #usar este comando para ordenar el espacio de figuras, aquí 2 figuras en una fila
> screen(2) #usar este comando antes de plotear la segunda figura
> close.screen(all=TRUE) # comando para regresar a las condiciones iniciales de una figura por ventana
> cov(rain,soi) #covarianza
[1] 5.650357
> cor(rain,soi) #coeficiente de correlación r
[1] 0.8620822
> lm(soi ~ rain) #ecuación de regresión lineal
Call:
lm(formula = soi ~ rain)
Coefficients:
(Intercept)      rain
-0.36867     0.03978
> plot(rain,soi,main="ENSO-rainfall relationship",xlab="rainfall", ylab="SOI index")
> abline(lm(soi~rain))
```

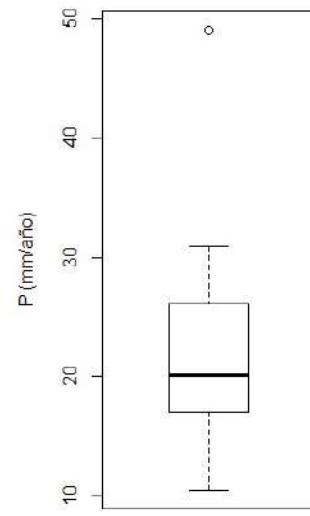
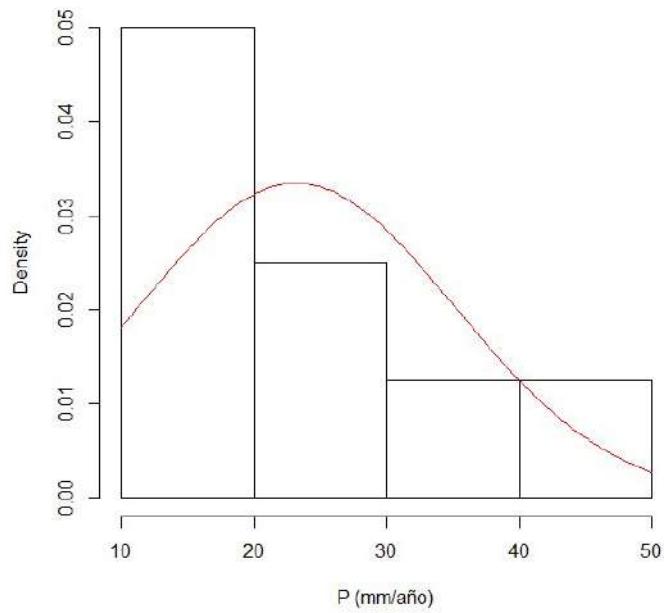
Precipitacion anual



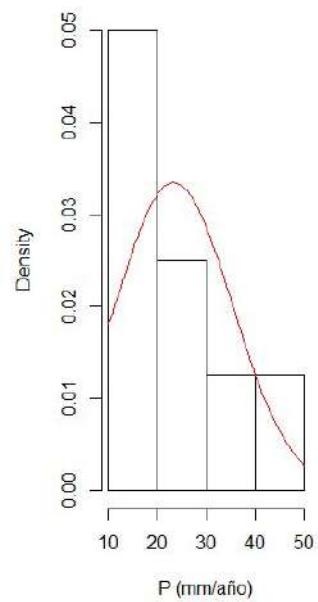
Precipitacion anual



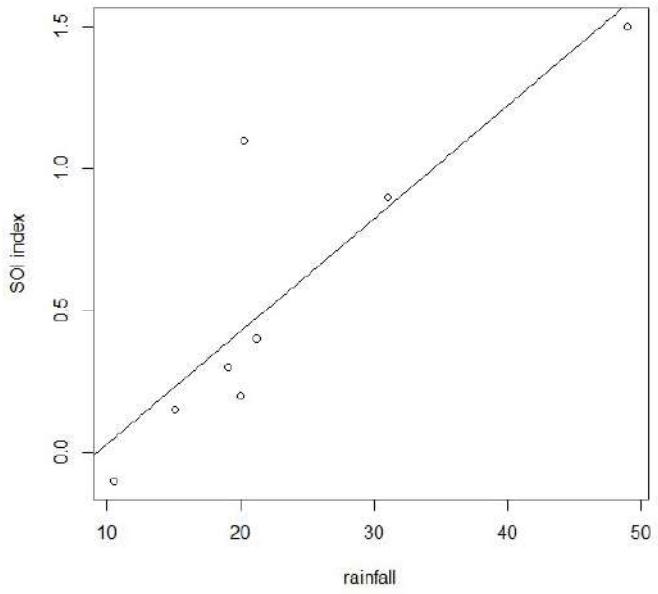
Precipitacion anual



Precipitacion anual



ENSO-rainfall relationship



Ejercicio A.4

En la consola de scripts:

- a) Crear una cadena de caracteres con el mensaje "Esto es una prueba". Mostrar el mensaje, identificar la clase de objeto, la longitud del objeto y el numero de caracteres.
- b) Crear un objeto numérico entero "x"; y otro real "y". Plotear la salida: "el valor de x =" y verificar la clase de objeto.
- c) Crear dos objetos lógicos: TRUE almacenado como "a" y FALSE almacenado como "b". Realizar la operación lógicas de conjunción, disyunción, negación y verificar las clases de objeto.
- d) Crear dos objetos complejos: $1+2i$; $-3i$. Sumar ambos objetos y verificar la clase del objeto.
- e) Crear tres listas de objetos, una conteniendo 1, 2, 3, 4; una segunda lista conteniendo: 1, hola, $1+i$; y una tercera lista conteniendo una secuencia continua del 1 al 10.
- f) Crear un vector con 3 elementos vacíos, detectar los elementos vacíos. Reemplazar el primer elemento del vector por el valor de 11. Visualizar el vector. Verificar si hay elementos vacíos.
- g) Crear una matriz vacía de 2 files y 4 columnas, verificar sus dimensiones. Reemplazar el valor de la ubicación fila 1 y columna 4 por el valor de 22.
- h) Crear una matriz de 3×4 con elementos secuenciales de 2 hasta 24 de dos en dos. Verificar el numero de filas y de columnas. Verificar la clase de objetos almacenados.

Respuestas A.4

```
cadena <- "Esto es una prueba"
```

```
cadena
```

```
typeof(cadena)
```

```
length(cadena)
```

```
nchar(cadena)
```

```
x <- 5
```

```
y <- 4.5
```

```
cat("x=", x)
```

```
typeof(x)
```

```
print("y=", y)
```

```
typeof(y)
```

```
a <- TRUE
```

```
b <- FALSE
```

```
m <- a | b
```

```
n <- a & b
```

```
p <- !a
```

```
typeof(p)
```

```
c1 <- 1 +2i
```

```
typeof(c1)
```

```
c2 <- -3i
```

```
c3 = c1 +c2
```

```
C3
```

```
lista1 <- list(1,2,3,4)
```

```
lista1
```

```
lista2 <- list(1,"hola", 1+1i)
```

```
lista2
```

```
length(lista2)
```

```
vector_nan = c(NA, NA, NA)
```

```
vector_nan
```

```
is.na(vector_nan)
```

```
vector_nan[1]<-11
```

```
vector_nan
```

```
is.na(vector_nan)
```

```
anyNA(vector_nan)
```

```
array2 =matrix(1:12, 3, 4)*2
```

```
array2
```

```
ndim <- dim(array2)
```

```
nrows <- ndim[1]
```

```
ncols <- ndim[2]
```

```
nrows
```

```
ncols
```

```
typeof(ndim)
```

```
class(ndim)
```

Ejercicio A.5

En la consola de scripts:

- a) En una región, se tienen los datos del índice de aridez para años continuos desde 1980: 0.5, 0.45, 0.74, 0.51, 0.78, 0.66, 0.63, 0.82, 0.67, 0.41. Automatizar la clasificación indicando si se trata de una región húmeda o árida. (usar la secuencia *if, else*)
- b) Mostrar los índices anteriores con su secuencia de años respectivos iniciando con la palabra "Año..." (usar la secuencia *for*)
- c) Mostrar los índices anteriores con su secuencia de años respectivos (usar la secuencia *while*)

Respuestas A.5

```
ia <- c(0.5, 0.45, 0.74, 0.51, 0.78, 0.66, 0.63, 0.82, 0.67, 0.41)
n_elem <- length(ia)
```

```
media <- mean(ia)
cat("los datos son de tipo:", class(ia))
if (media >= 0.65){
  cat("\n\nRegion Humeda")
}else{cat("\n\nRegion Arida")}
```

```
for(i in 1:n_elem){
  cat("\n", "Año", i+1979, "-->", ia[i])
}
```

```
i <- 1
while (i <= n_elem) {
  cat("\n", i+1979, "-->", ia[i])
  i = i+1
}
```

Ejercicio A.6

En la consola de scripts:

- 1) Crear una función capaz de calcular la velocidad de un río ancho por el método de Manning: datos de ingreso el tirante, la pendiente y el coeficiente de Manning en el S.I.
- 2) Calcular la velocidad para un tirante de 1.5 m, pendiente de 1% y un coeficiente de Manning de 0.035.
- 3) Calcular la velocidad para un tirante de 1.5 m, pendiente de 0.5% y un coeficiente de Manning de 0.035

Respuestas A.6

$$V = \frac{1}{n} R_h^{\frac{2}{3}} \cdot S^{\frac{1}{2}}$$

V: Velocidad

Rh: Radio Hidraulico

S: Pendiente

n: Coeficiente de Manning

```
vel.manning<- function(y,s,n) #creando la funcion vel.manning  
con tres argumentos  
{ result <- y^(2/3)*s^0.5/n #definiendo las operaciones  
print(result) }
```

```
vel.manning(1.5,0.01,0.035) # calculo de la velocidad en m/s  
[1] 3.743916
```

```
vel.manning(1.5,0.005,0.035) # calculo de la velocidad en m/s  
[1] 2.647349
```

3. Análisis exploratorio de datos



My code doesn't work

```

read.table(file, header = FALSE, sep = "", quote = "\"\"", dec = ".", row.names,
col.names, as.is = FALSE, na.strings = "NA", colClasses = NA, nrows = -1, skip = 0,
check.names = TRUE, fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip =
TRUE, comment.char = "#")
read.csv(file, header = TRUE, sep = ",", quote = "\"\"", dec = ".", fill = TRUE, ...)

```

Leyendo archivos, bases de datos

“Data Frame”

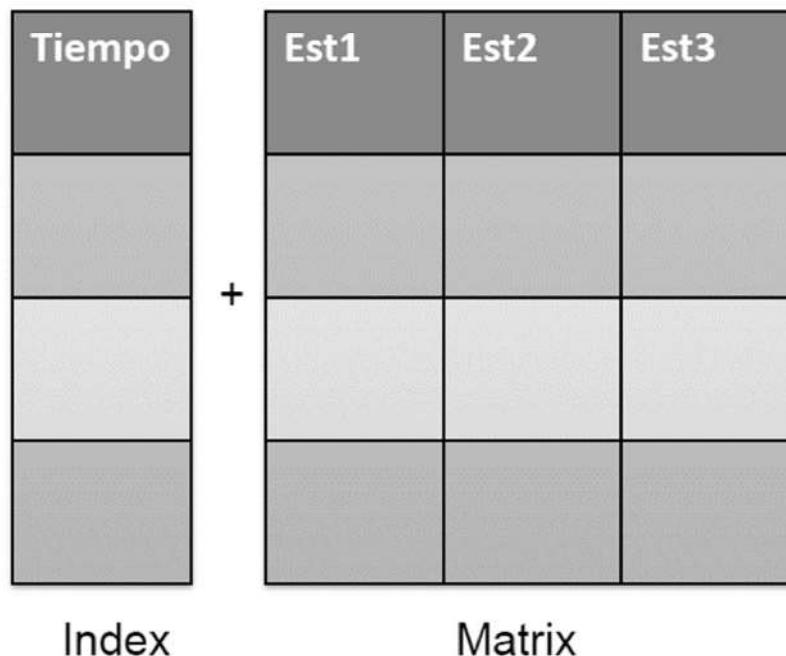
file	the name of the file (within "" or a variable of mode character), possibly with its path (the symbol \ is not allowed and must be replaced by /, even under Windows), or a remote access to a file of type URL (http://...)
header	a logical (FALSE or TRUE) indicating if the file contains the names of the variables on its first line
sep	the field separator used in the file, for instance <code>sep="\t"</code> if it is a tabulation
quote	the characters used to cite the variables of mode character
dec	the character used for the decimal point
row.names	a vector with the names of the lines which can be either a vector of mode character, or the number (or the name) of a variable of the file (by default: 1, 2, 3, ...)
col.names	a vector with the names of the variables (by default: V1, V2, V3, ...)
as.is	controls the conversion of character variables as factors (if FALSE) or keeps them as characters (TRUE); <code>as.is</code> can be a logical, numeric or character vector specifying the variables to be kept as character
na.strings	the value given to missing data (converted as NA)
colClasses	a vector of mode character giving the classes to attribute to the columns
nrows	the maximum number of lines to read (negative values are ignored)
skip	the number of lines to be skipped before reading the data
check.names	if TRUE, checks that the variable names are valid for R
fill	if TRUE and all lines do not have the same number of variables, “blanks” are added
strip.white	(conditional to <code>sep</code>) if TRUE, deletes extra spaces before and after the character variables
blank.lines.skip	if TRUE, ignores “blank” lines
comment.char	a character defining comments in the data file, the rest of the line after this character is ignored (to disable this argument, use <code>comment.char = ""</code>)

Z's Ordered Observations

Zoo y xts

```
zoo(x = NULL, order.by = index(x), frequency = NULL, calendar =  
getOption("zoo.calendar", TRUE))  
# S3 method for zoo  
print(x, style = , quote = FALSE, ...)
```

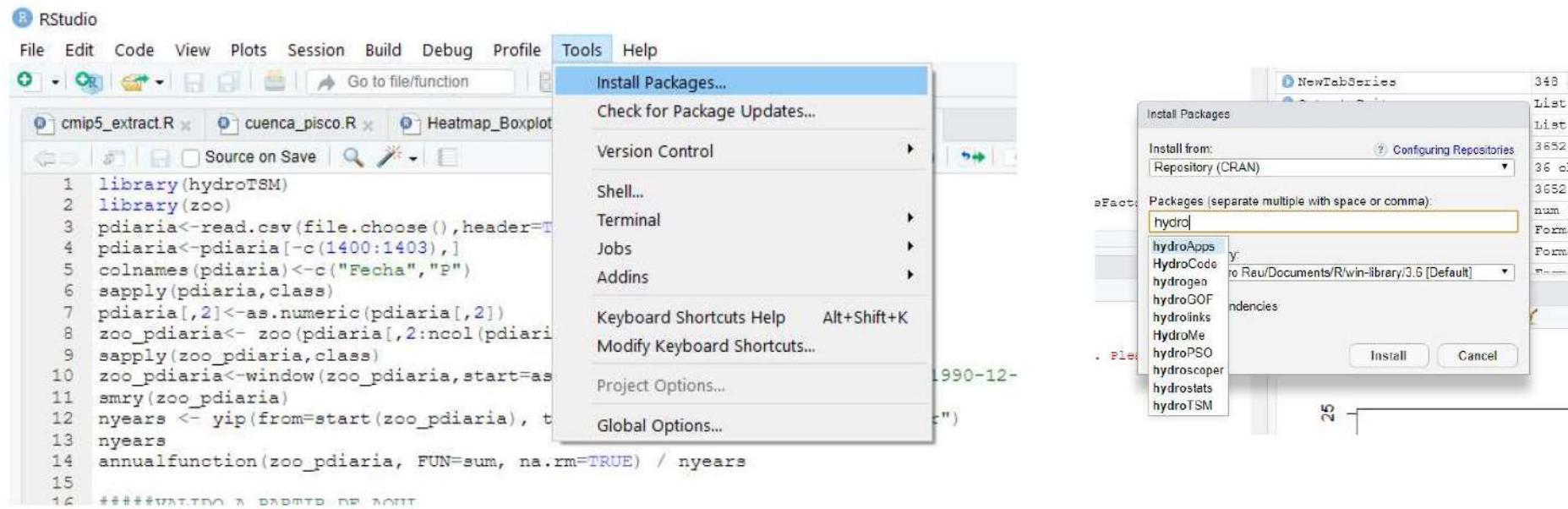
```
xts(x = NULL, order.by = index(x), frequency = NULL, unique = TRUE, tzone =  
Sys.getenv("TZ"), ...)  
is.xts(x)
```



B. Explorando datos de precipitación mensual almacenados en una matriz de una sola estación

Uso de las librerías: *lattice*, *zoo*, *hydroTSM*

- Instalar librerías TOOLS - INSTALL PACKAGES: *lattice*, *zoo*, *hydroTSM*
- Descargar archivo "pmensual.txt"
<https://drive.google.com/file/d/1Z1k6QOpFeyaOBvRyzgb5kH4Od8pLOpKS/view?usp=sharing>
- Revisión de archivo texto (guardado desde el excel)



Ejercicio B

En la consola de scripts, crear un código que realice lo siguiente:

- 1) Leer el archivo de precipitaciones mensuales "pmensual.txt" almacenados en formato matriz de 13 columnas (año mes1 mes 2 mes 12) y 45 filas (nombre valor1 valor 2 ...).
- 2) Plotear con puntos y línea continua la serie de enero y sus años respectivos.
- 3) Plotear toda la serie de tiempo
- 4) Agregar una curva de tendencia
- 5) Plotear un gráfico de calor "heatmap"

Grabar y dar nombre al script: Analisis_mensual.R

Respuestas B

```
library("lattice") #llamar librería lattice
library("hydroTSM") #llamar librería TSM
pmensual<-read.table(file.choose(), header = T) #lectura de archivo con una ventana para seleccionar el
#archivo, "T" para contener datos en la cabecera como etiquetas
typeof(pmensual)
class(pmensual)
names(pmensual)
prec_ENE<-pmensual$ENE #Visualización de la serie de tiempo:
yr<-pmensual$Año
plot(yr, prec_ENE, type = "b")
#Visualización de la serie de tiempo:
pmensual<-read.table(file.choose(), header = F) #lectura de archivo con ruta directa, "F" no considerar
#la cabecera como datos
datos<-pmensual[2:45,2:13]
datos_vector<-as.vector(t(datos)) #convirtiendo a vector lineal
datos_ts<-stats::ts(datos_vector, start=c(1964, 1), end=c(2010, 12), frequency=12)
plot.ts(datos_ts, col="black", main="Monthly Rainfall time series", ylab="P [mm]", xlab="Year")
lines(lowess(time(datos_ts), datos_ts), col="blue", lwd=2) #agregar curva de tendencia
```

#Diagrama de calor o HeatMap:

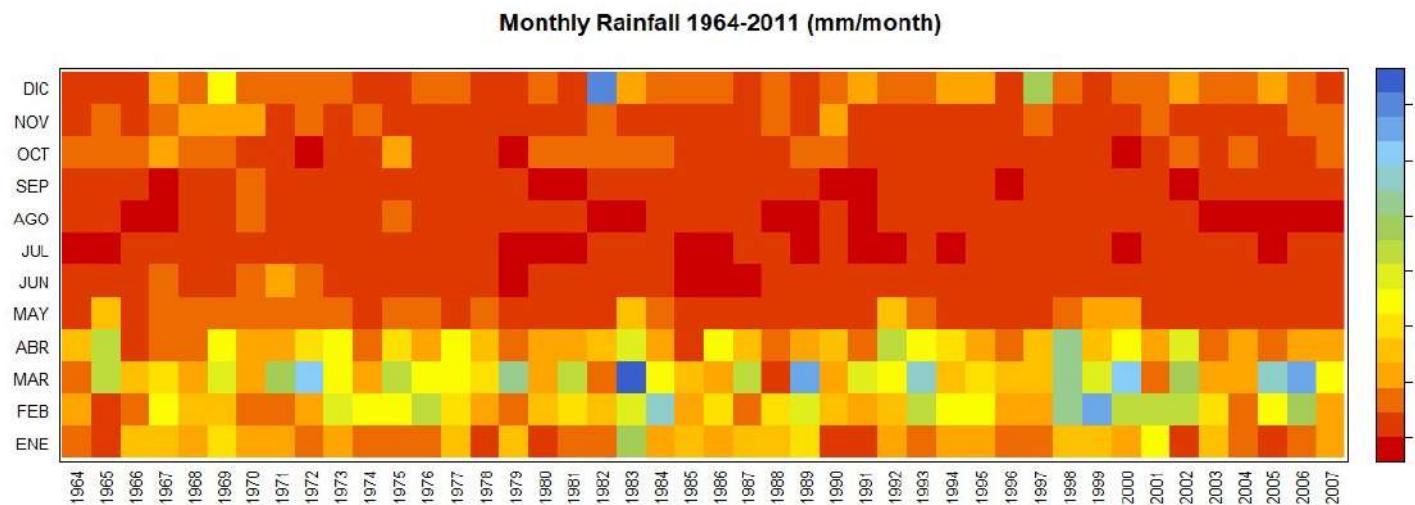
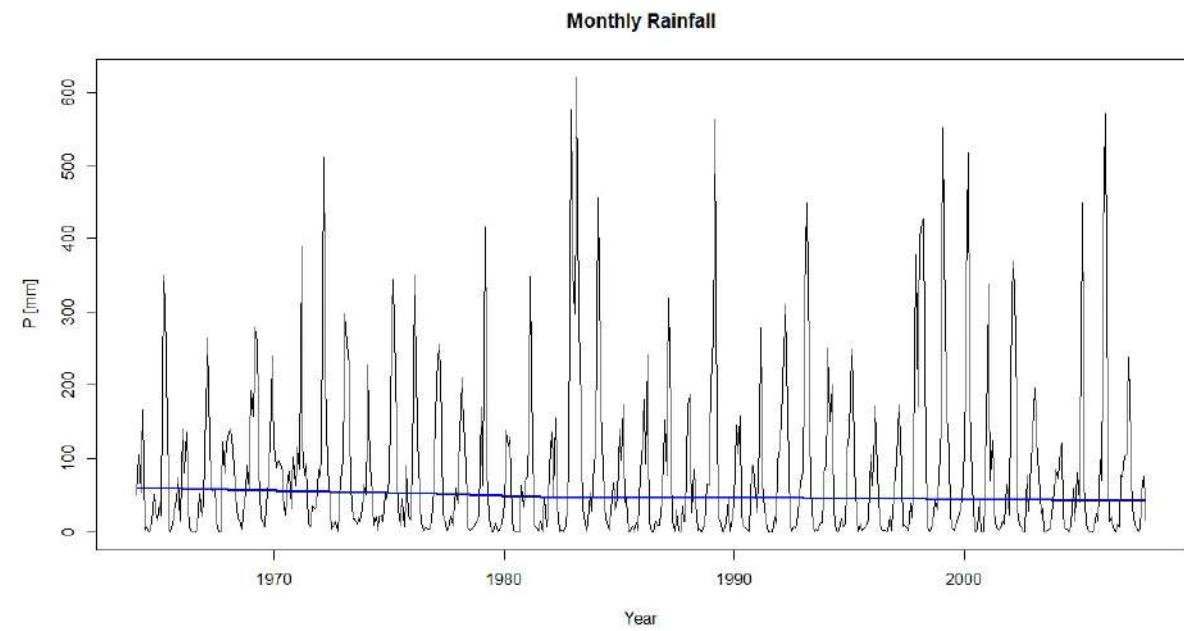
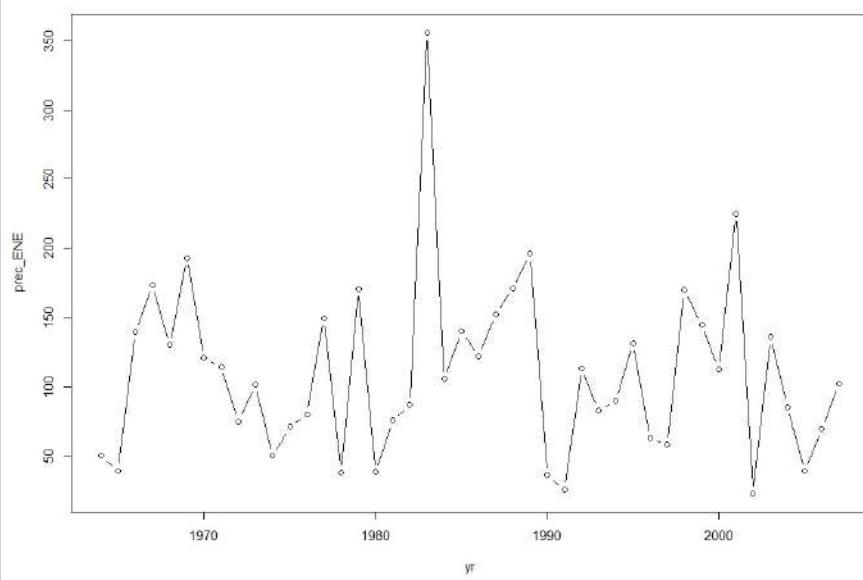
```
lluvia<-pmensual[2:45,2:13] #lectura de solo datos de lluvia, sin etiquetas de años, ni meses (desde 1964 al
#2007)
```

```
meses<-pmensual[1:1,2:13] #lectura de la cabecera de meses en la primera fila
```

```
colnames(lluvia)<-unlist(meses) #desagrega los nombres de los meses y se asigna a la matriz
```

```
rownames(lluvia)<-pmensual[2:45,1:1] #desagrega los nombres de los meses y se asigna a la matriz
```

```
matrixplot(lluvia, ColorRamp="Precipitation",main="Monthly Rainfall 1964-2011 (mm/month)")
```



C. Explorando datos de precipitación diaria de varias estaciones almacenados en una matriz

Uso de las librerías: easypackages, xts, lattice, ggplot2

- Instalar librerías TOOLS - INSTALL PACKAGES: *easypackages*, *xts*, *lattice*, *ggplot2*
- Descargar archivo "p_diarias.csv"
<https://drive.google.com/file/d/1qj6I51SjhNIUfhq56Fw8kK4v3g1ttYq8/view?usp=sharing>
- Revisión de archivo csv (separados por comas con formato de fecha %Y-%M-%D)

Ejercicio C

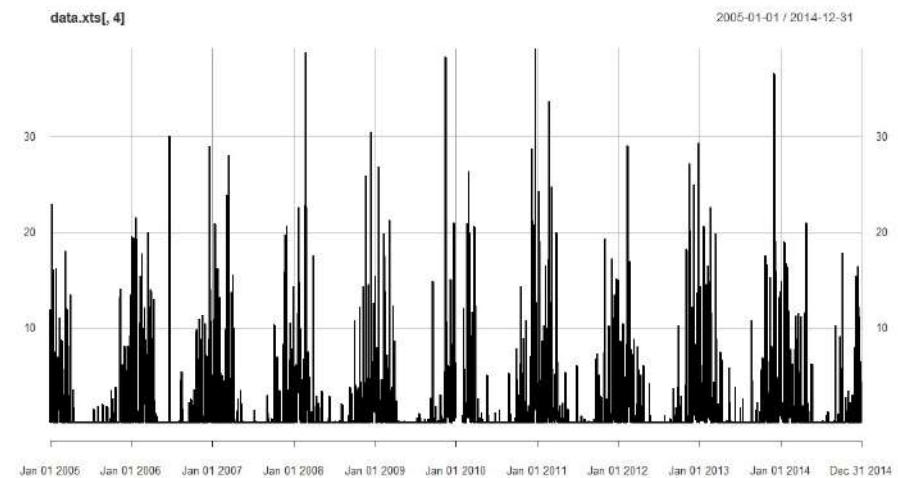
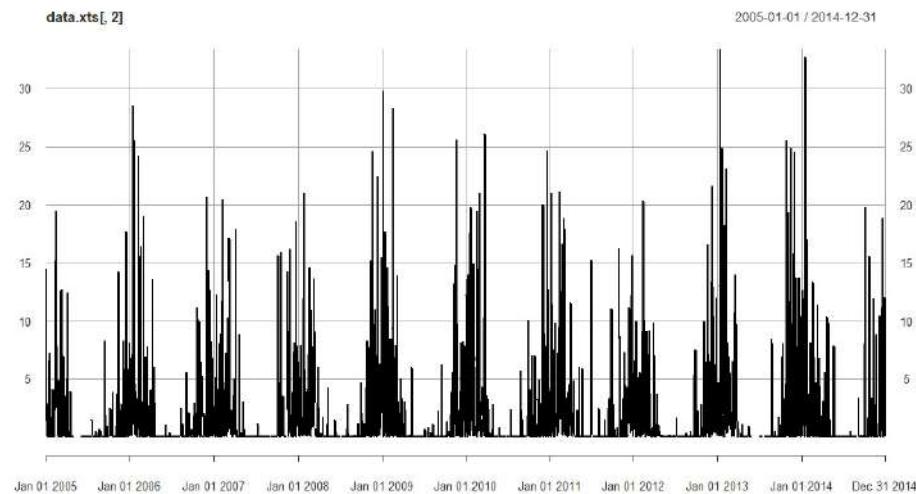
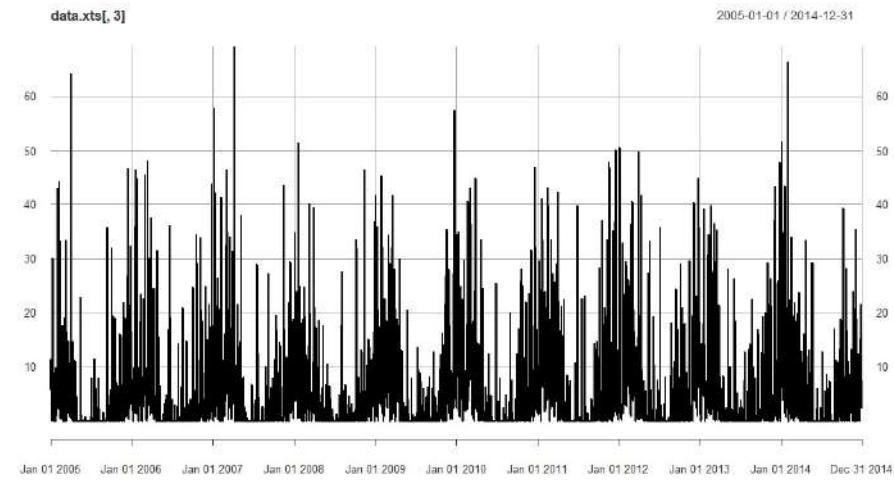
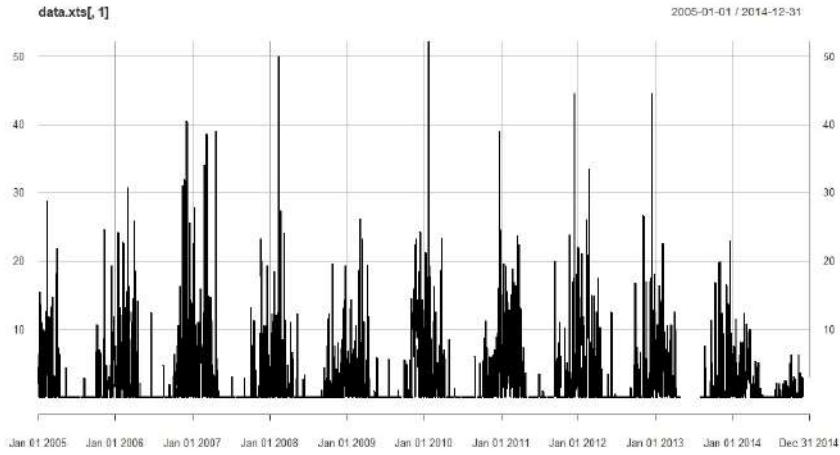
En la consola de scripts, crear un código que realice lo siguiente:

- 1) Leer el archivo de precipitaciones diarias "p_diarias.csv" almacenados en formato matriz con 4 estaciones (identificadores 101, 102, 103, 104) desde el 1Ene2005 al 31Dic2014
 - 2) Plotear las series de tiempo para cada estación
 - 3) Plotear las series de tiempo en conjunto con la misma escala vertical
 - 4) Plotear un boxplot de las estaciones en conjunto
 - 5) Plotear 3 histogramas correspondientes a 3 estaciones
- Grabar y dar nombre al script: *Analisis_diario.R*

Respuestas C

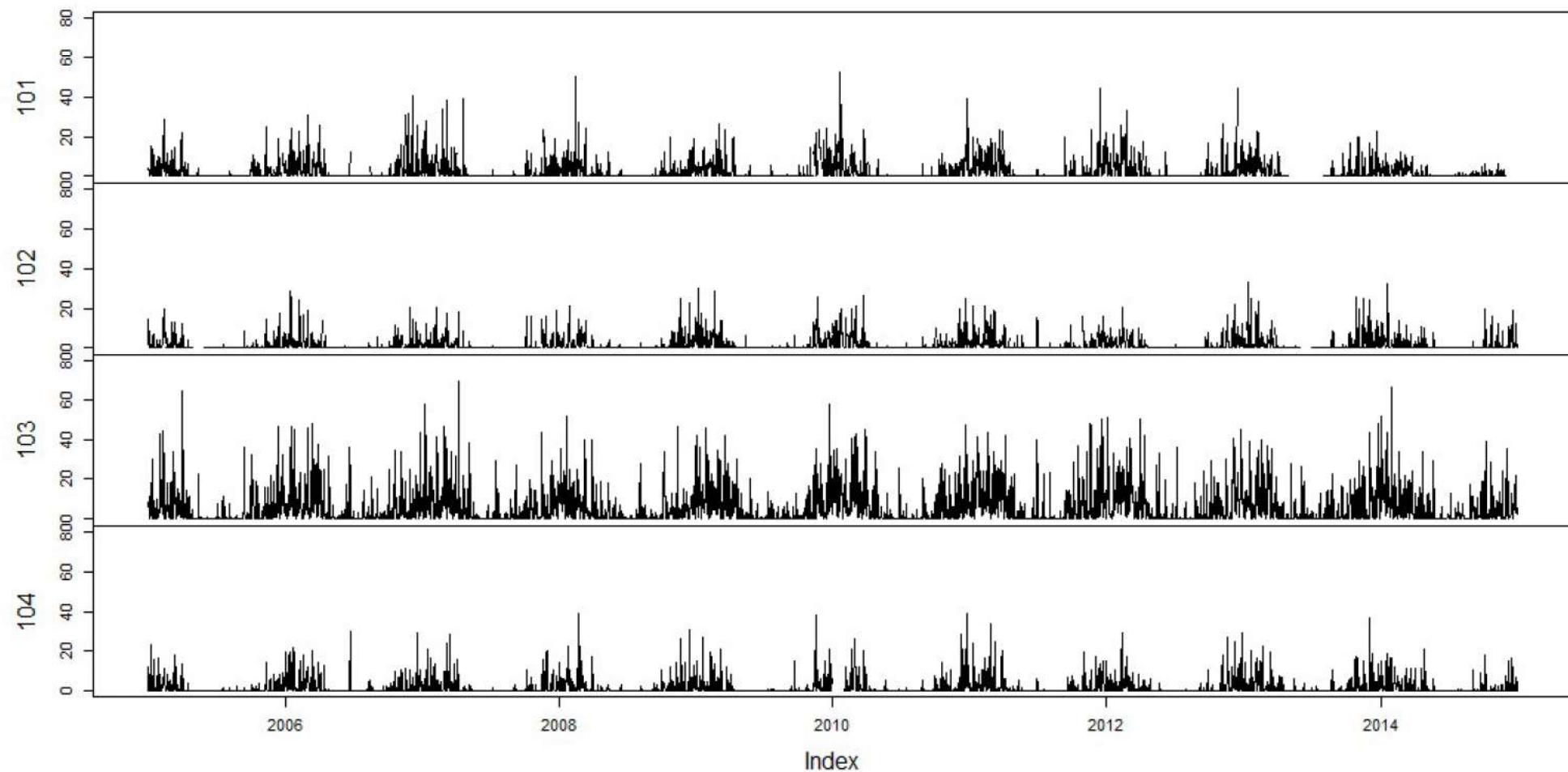
```
library(easypackages)
library(xts)
library(lattice)
library(ggplot2)
pdiaria <- read.csv(file.choose(),header=TRUE, check.names = F, stringsAsFactors = F)
str(pdiaria) #para ver la estructura del objeto
idx <- as.Date(pdiaria[,1]) #formato fecha a la 1era columna
data.matrix <- pdiaria[,-1] #formato matriz al resto de columnas a excepcion de la 1era columna
data.xts <- xts(data.matrix, order.by = idx ) #crear un objeto xts (eXtended time series)
str(data.xts)
plot(data.xts)
plot(data.matrix[,1], type="l")
plot(data.xts[,2])

# convertir a un objeto zoo
data.zoo <- as.zoo(data.xts)
str(data.zoo)
plot(data.zoo, main = "Series de tiempo de precipitación")
summary(data.zoo)
max(data.zoo, na.rm = T)
plot(data.zoo, main = "Series de tiempo de precipitación", ylim = c(0,80))
xyplot(data.xts,xlab = "Fecha",ylab = "Precipitación [mm/dia]",ylim=c(0,100))
autoplot(data.xts[,1:2]) +theme_bw() +xlab('Fecha') +ylab('Precipitación [mm/dia]')
boxplot(coredata(data.xts))
hist(coredata(data.xts[,1]), freq = T) # cantidad de datos por clase
histogram(coredata(data.xts[,1])) # porcentaje de datos por clase
```

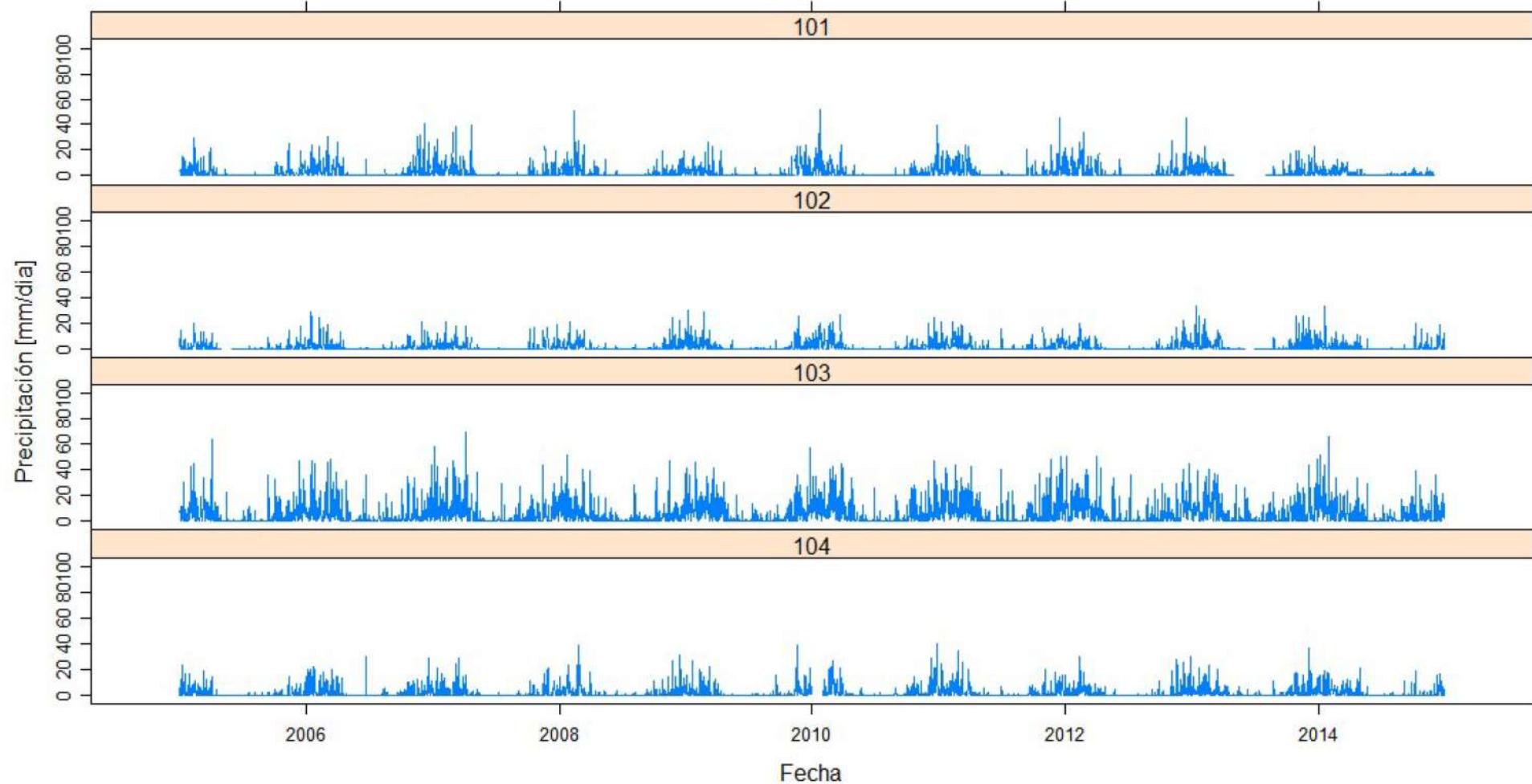


Ploteo simple

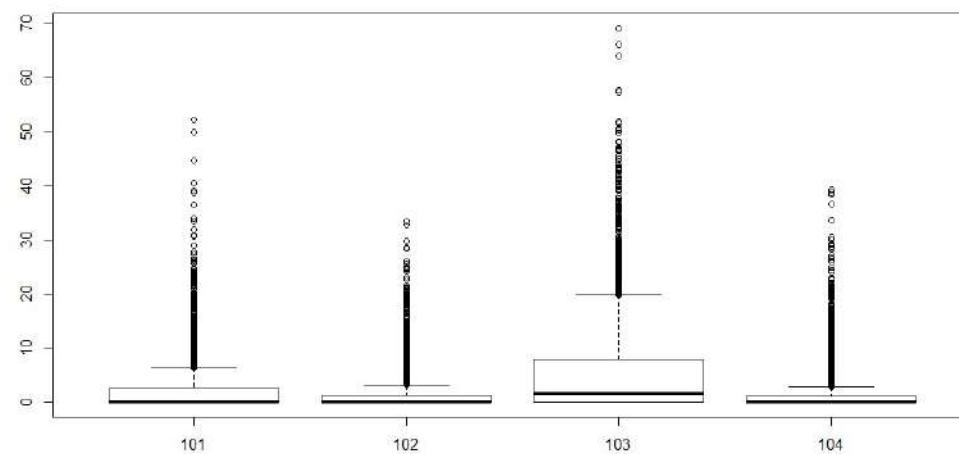
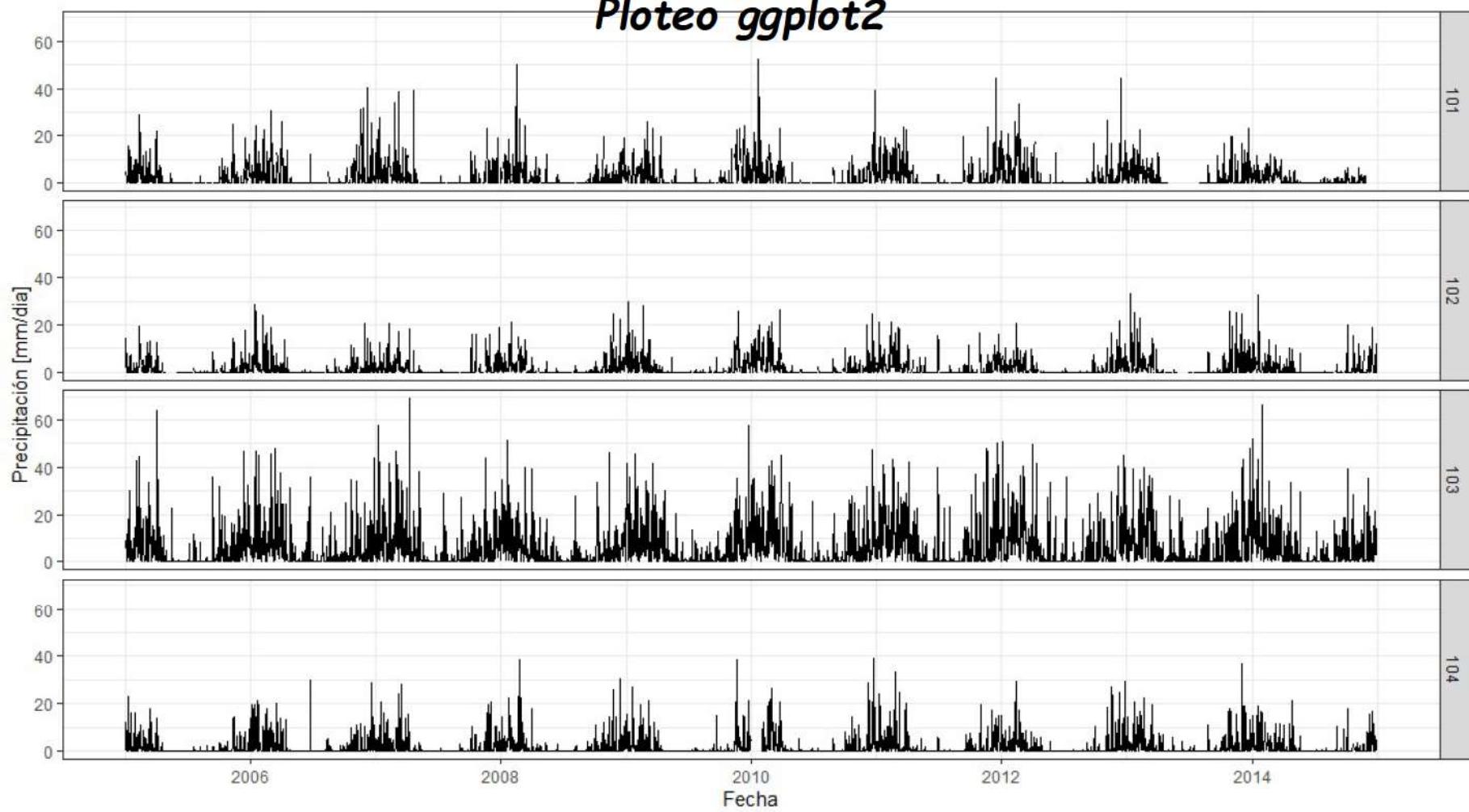
Series de tiempo de precipitación

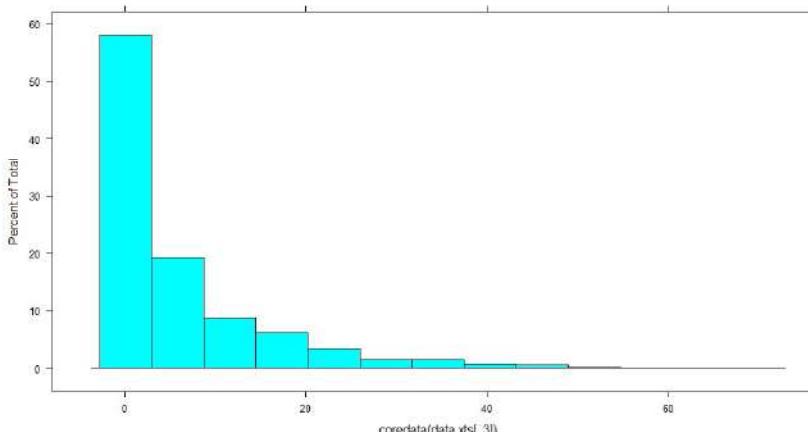
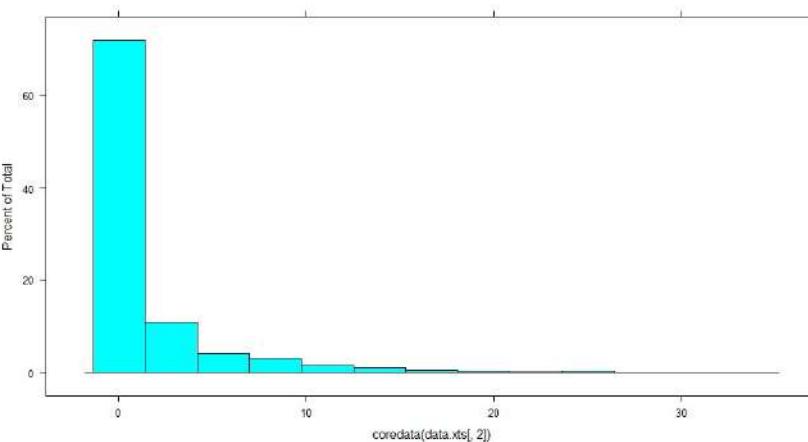
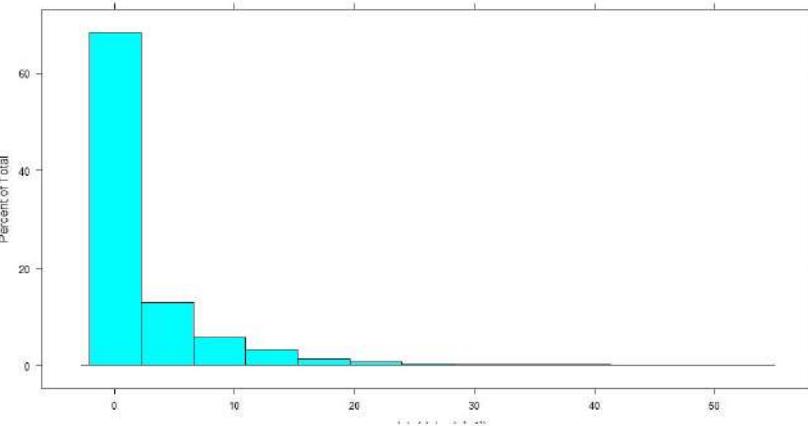
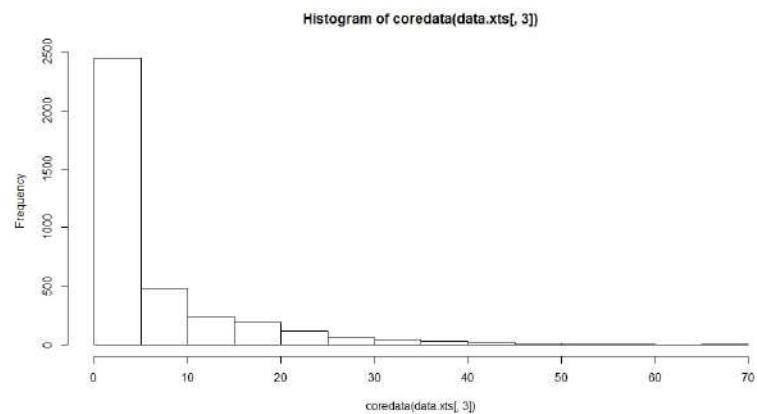
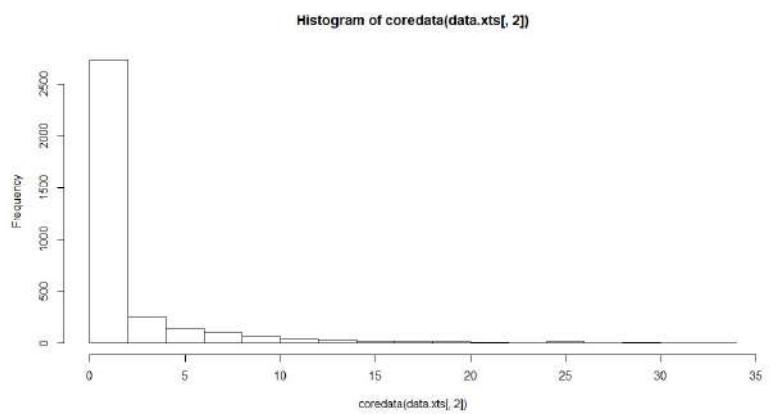
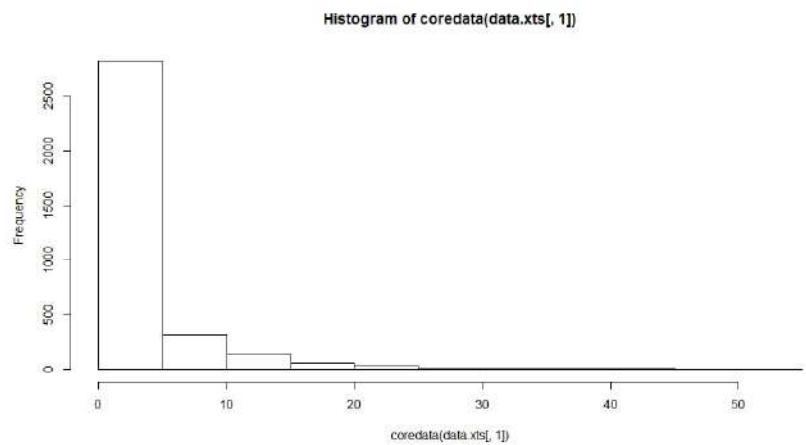


Ploteo lattice

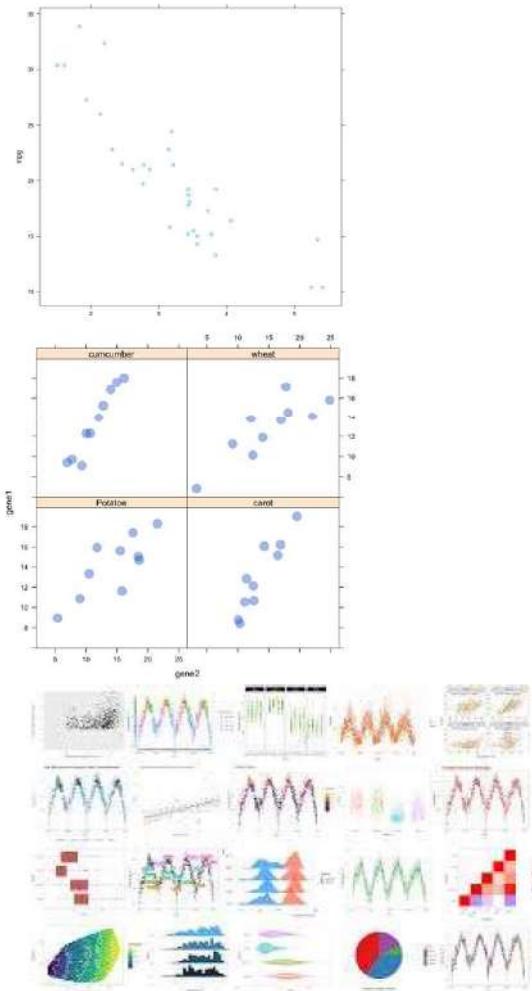


Ploteo ggplot2





4. Tratamiento de datos



D. Convirtiendo datos diarios a mensuales y anuales

Ejercicio D

En la consola de scripts, crear un código que realice lo siguiente:

- 1) Leer el archivo de precipitaciones diarias "p_diarias.csv" almacenados en formato matriz con 4 estaciones (identificadores 101, 102, 103, 104) desde el 1Ene2005 al 31Dic2014
- 2) La estación 103 se encuentra completa, convertir sus datos diarios en mensuales y éstos en anuales. Plotear las series de tiempo, los datos mensuales en formato linea y los anuales en formato barras. Plotear todas las estaciones en conjunto y analizar el vacío de información.
- 3) Plotear un boxplot para los datos mensuales de cada estacion
- 4) Plotear un boxplot estacional de la estacion 103

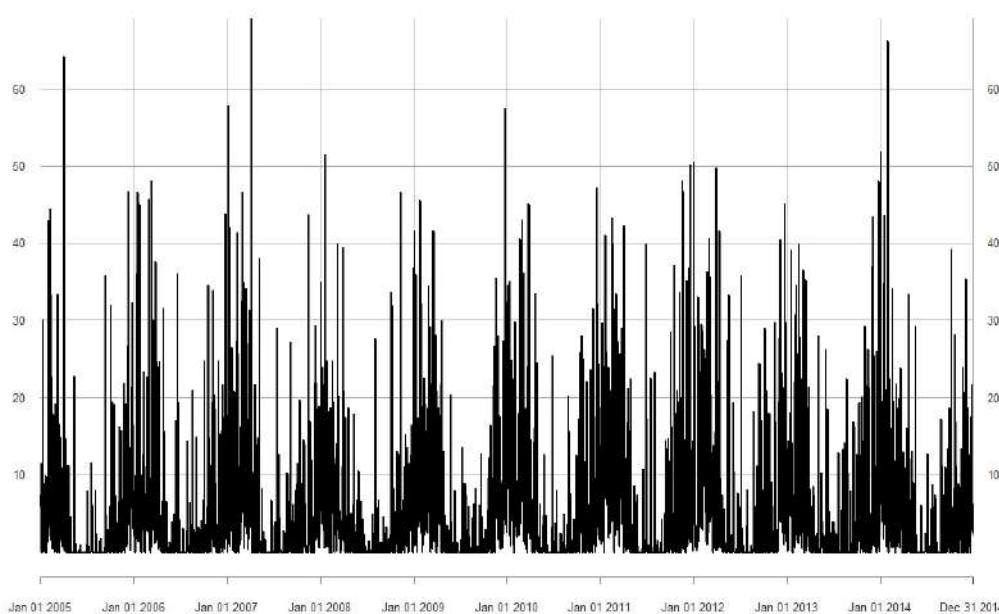
Respuestas D

```
library(easypackages)
library(xts)
library(lattice)
library(ggplot2)
pdiaria <- read.csv(file.choose(), header=TRUE, check.names = F, stringsAsFactors = F)
str(pdiaria)
idx <- as.Date(pdiaria[,1])
data.matrix <- pdiaria[,-1]
data.xts <- xts(data.matrix, order.by = idx )
str(data.xts)
plot(data.xts)
plot(data.matrix[,1], type="l")
plot(data.xts[,3])
#Convirtiendo a mensuales la estacion 103
data.monthly <- apply.monthly(data.xts[,3], FUN = sum)
plot(data.monthly)
#Todas las estaciones a mensuales
data.monthly <- apply.monthly(data.xts, FUN=apply, MARGIN = 2, sum)
xyplot(data.monthly, ylim = c(0,600))
#Convirtiendo a anuales la estacion 103
data.anual <- apply.yearly(data.xts[,3], FUN = sum)
barplot(data.anual)
#Todas las estaciones a anuales
data.anual <- apply.yearly(data.monthly, FUN=apply, 2, sum)
xyplot(data.anual)

boxplot(coredata(data.monthly)) # datos mensuales por estacion
# boxplot con estacionalidad de Lluvia para la estacion 103
boxplot(matrix(coredata(data.monthly[,3]), nrow=nrow(data.monthly)/12, ncol=12, byrow=T),
col="gray", main=c(paste(names(data.monthly[,3])), "Prec mensual [mm]"))
```

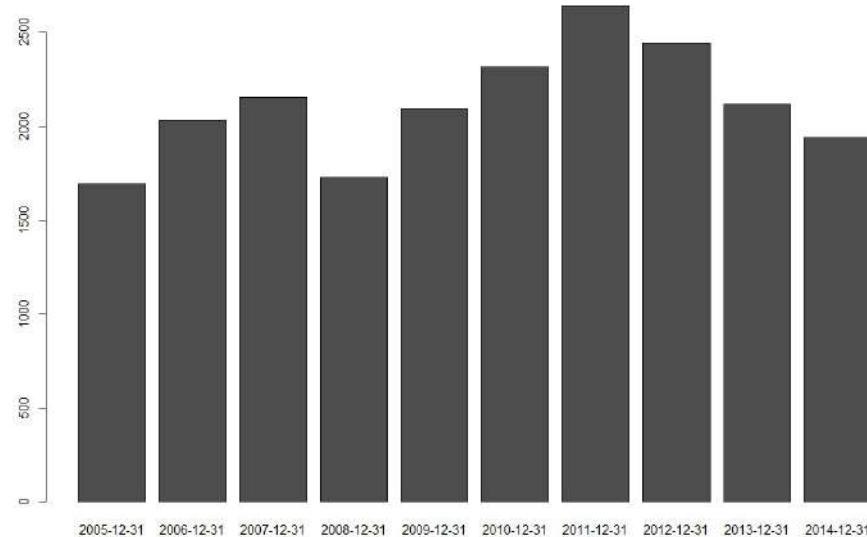
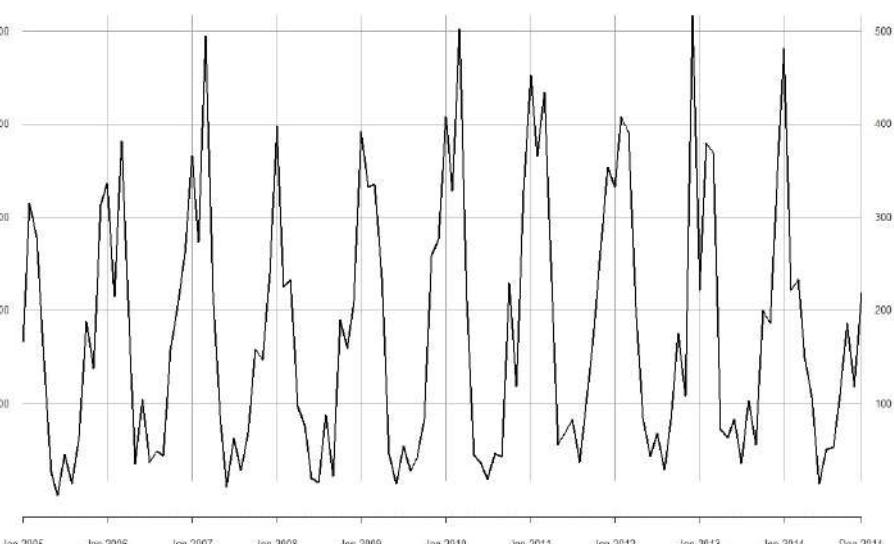
`data.xts[, 3]`

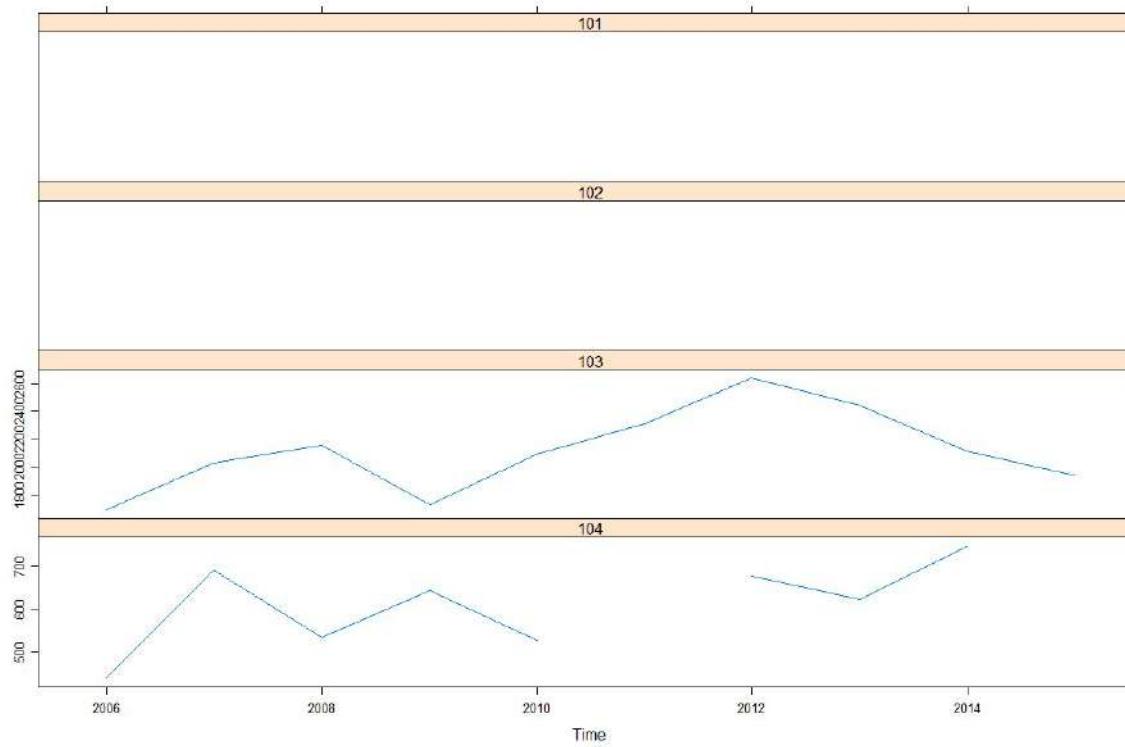
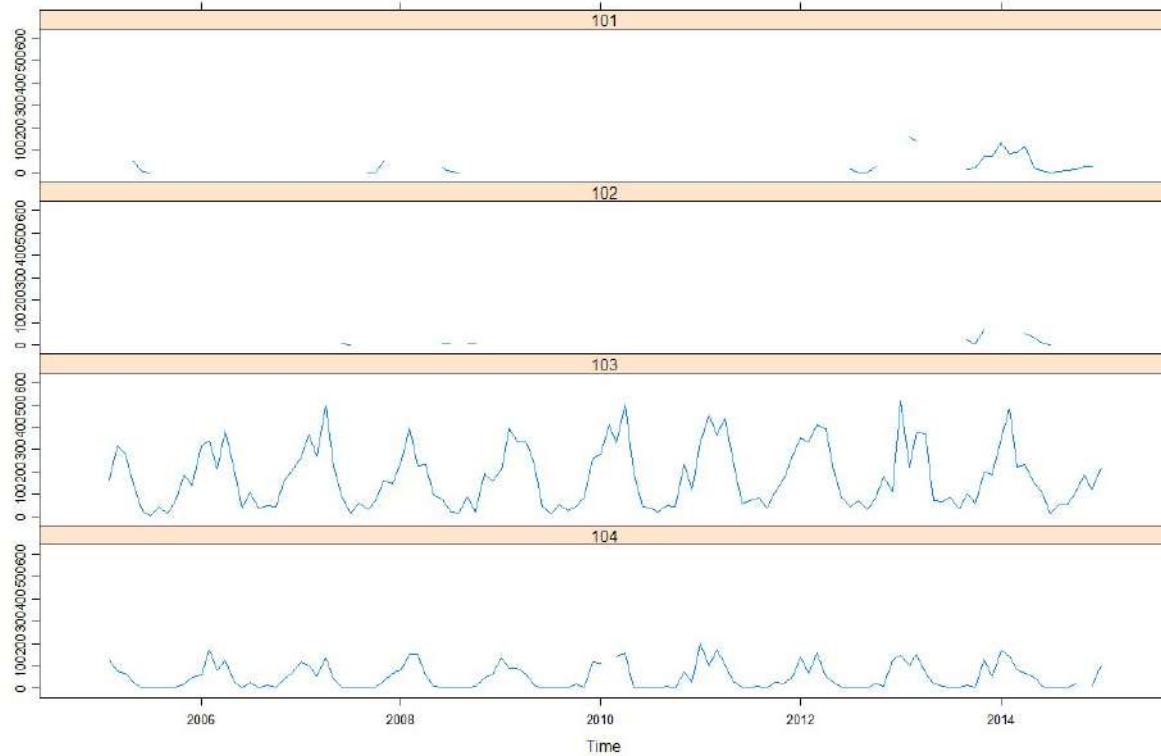
2006-01-01 / 2014-12-31

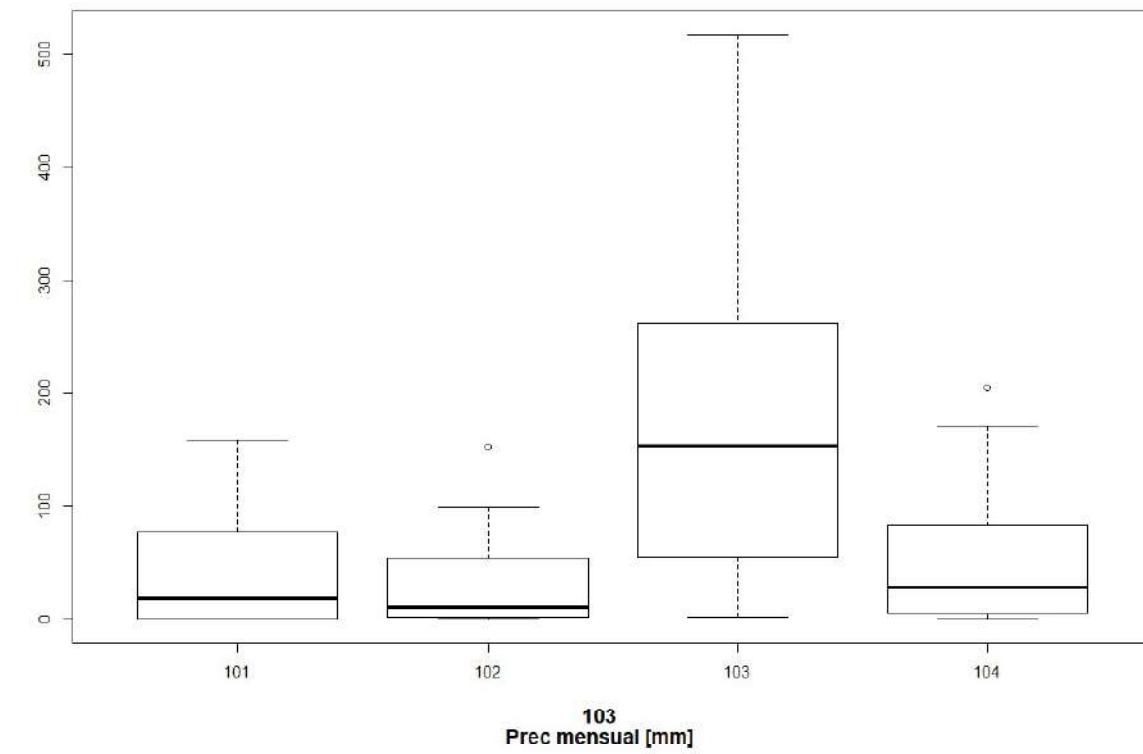


`data.monthly`

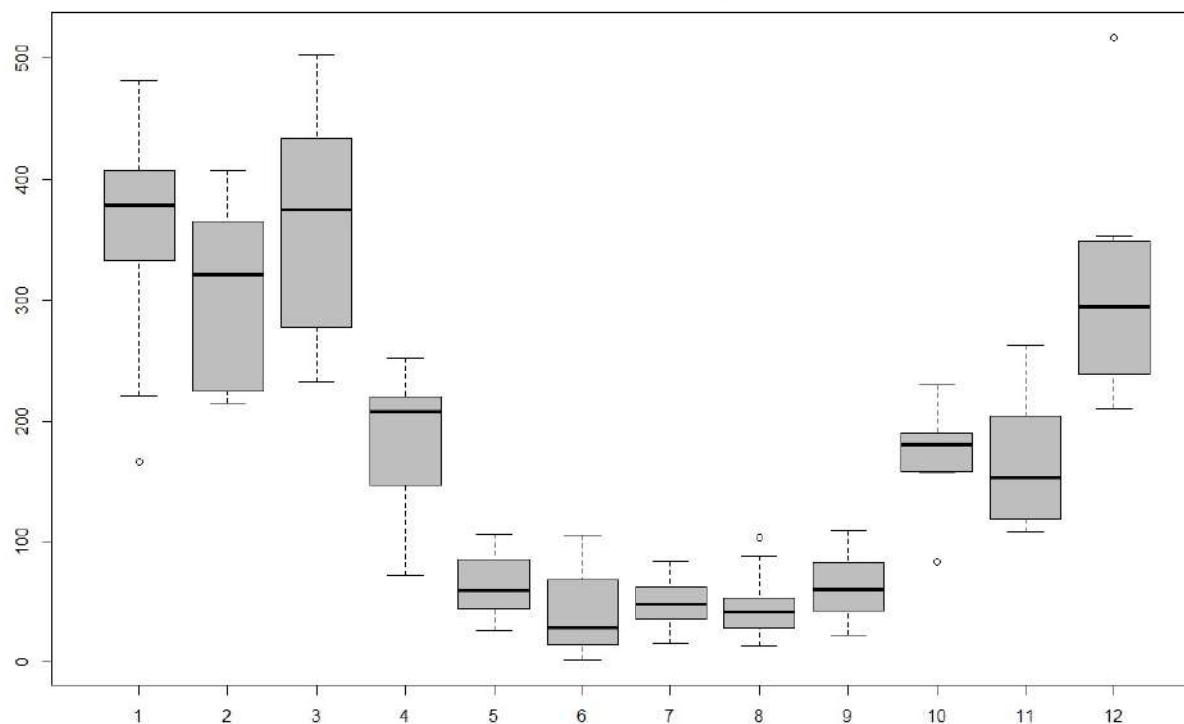
2005-01-31 / 2014-12-31







103
Prec mensual [mm]



E. Completación de datos

Ejercicio E

Crear un código que realice lo siguiente:

- 1) De la base de datos anterior, plotear en un mismo gráfico, boxplots e histogramas de frecuencias para la estacion 103 desde el nivel diario hasta el nivel annual.
- 2) Plotear en un mismo gráfico, boxplots e histogramas de frecuencias para la estacion 103 desde el nivel mensual hasta el nivel annual.
- 3) Visualizar los datos vacios en cada estación en sus series de tiempo
- 4) Completar los datos de forma forzada para la estación 101, con las opciones: Interpolación, Filtro de Kalman y Promedio de toda la serie.
- 5) Completar los datos de todas las estaciones, con la metodología de correlaciones cruzadas.

Respuestas E

```
library(easypackages)
library(xts)
library(lattice)
library(ggplot2)
pdiaria <- read.csv(file.choose(), header=TRUE, check.names = F, stringsAsFactors = F)
str(pdiaria)
idx <- as.Date(pdiaria[,1])
data.matrix <- pdiaria[,-1]
data.xts <- xts(data.matrix, order.by = idx )
str(data.xts)
plot(data.xts)
plot(data.matrix[,1], type="l")
plot(data.xts[,3])
```

#Ploteando la ubicacion de los valores vacios

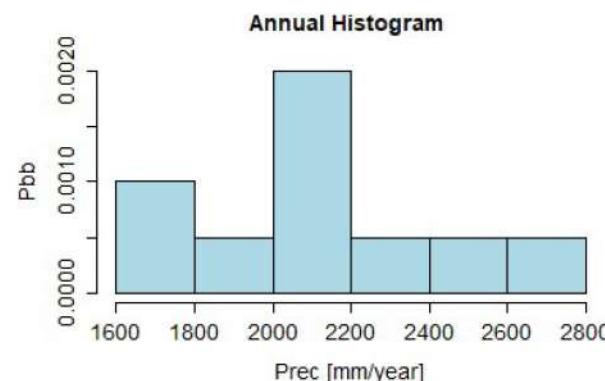
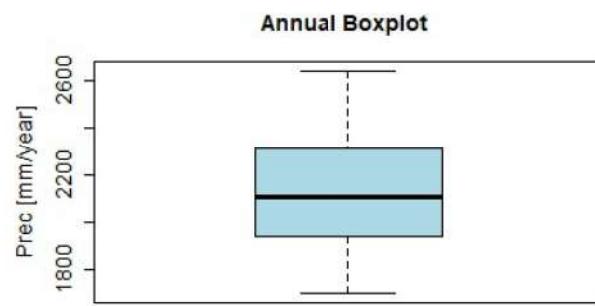
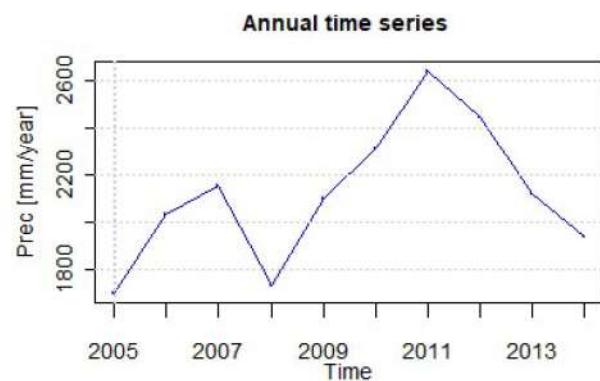
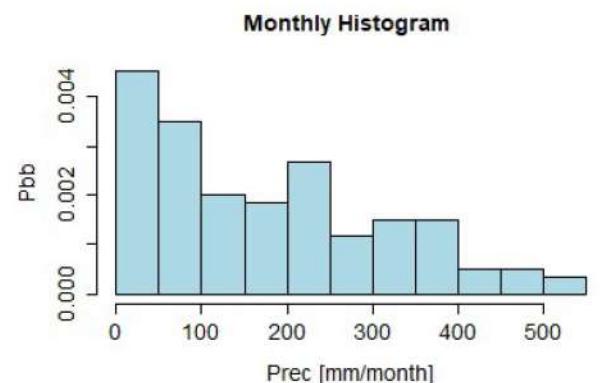
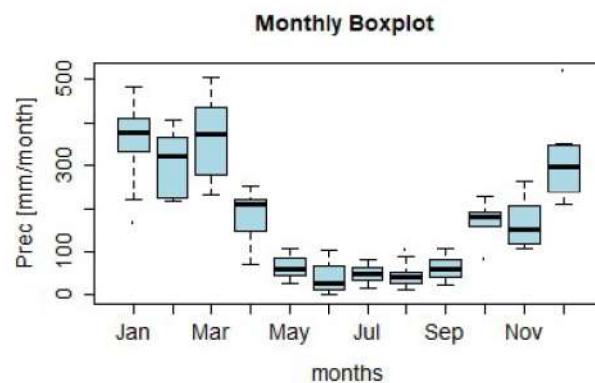
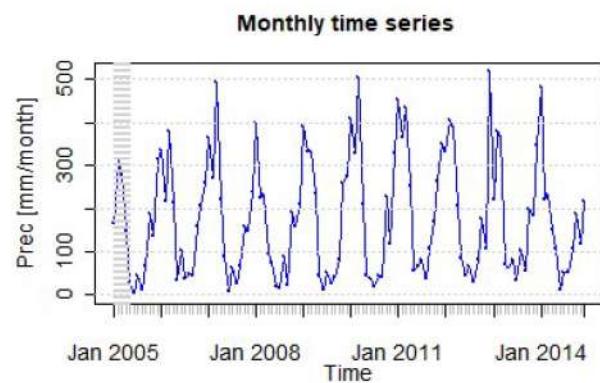
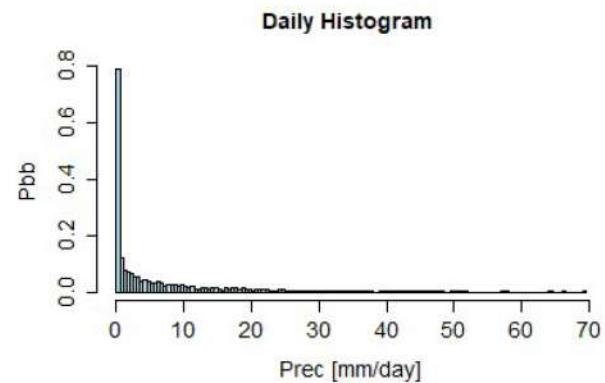
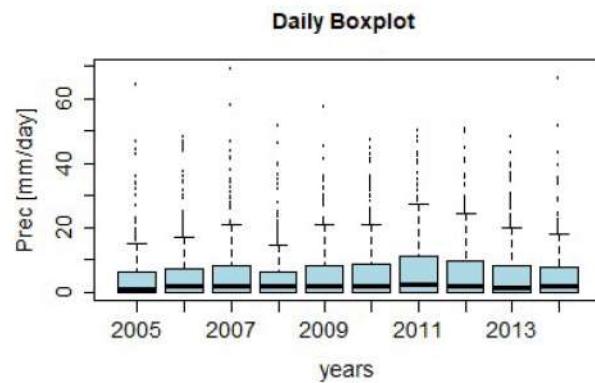
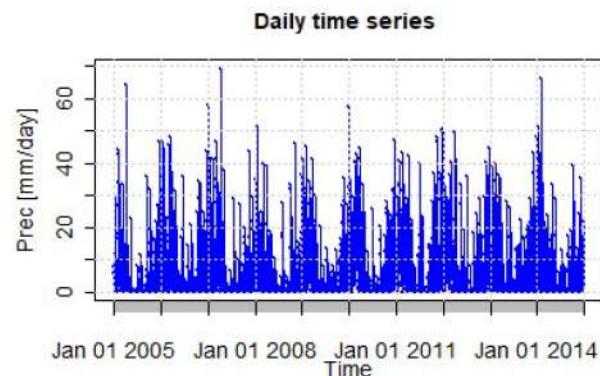
```
library(lubridate)
library(imputeTS)
library(fracdiff)
library(hydroGOF)
library(plyr)
plotNA.distribution(data.xts[,1], ylab = "P", xlab = "Nro datos")
plotNA.distribution(data.xts[,2], ylab = "P", xlab = "Nro datos")
plotNA.distribution(data.xts[,3], ylab = "P", xlab = "Nro datos")
plotNA.distribution(data.xts[,4], ylab = "P", xlab = "Nro datos")
```

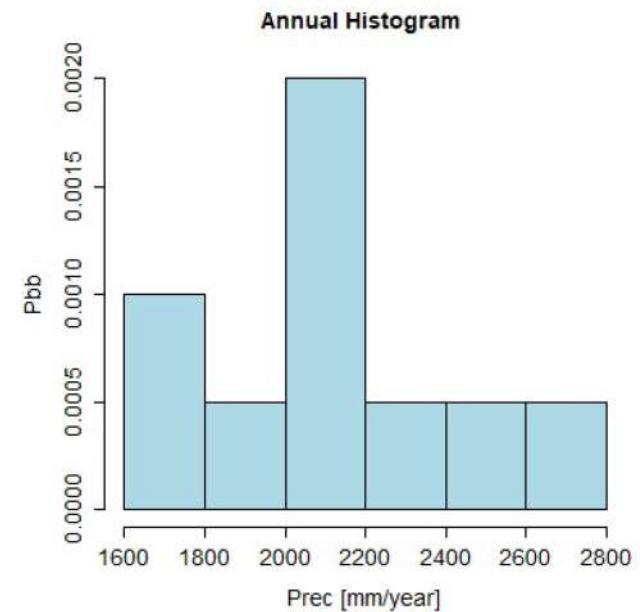
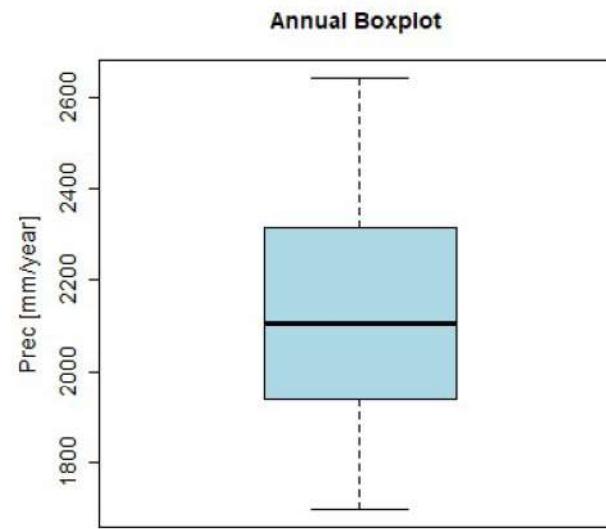
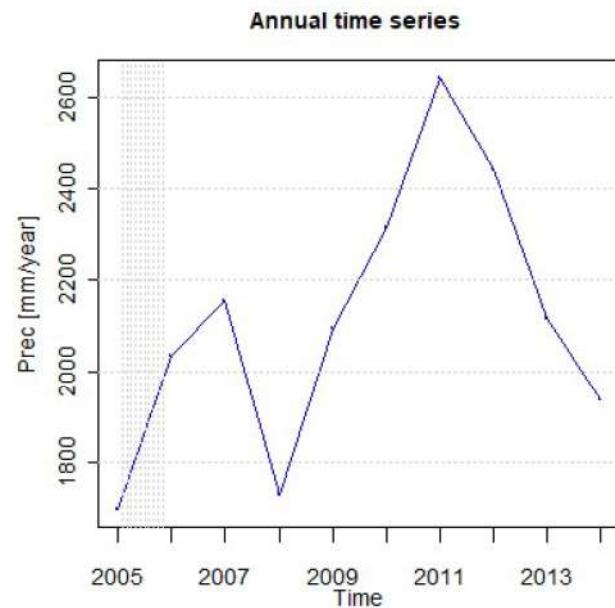
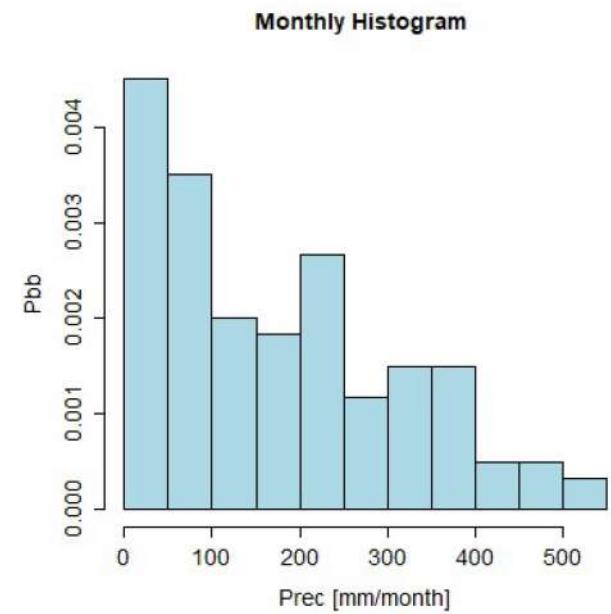
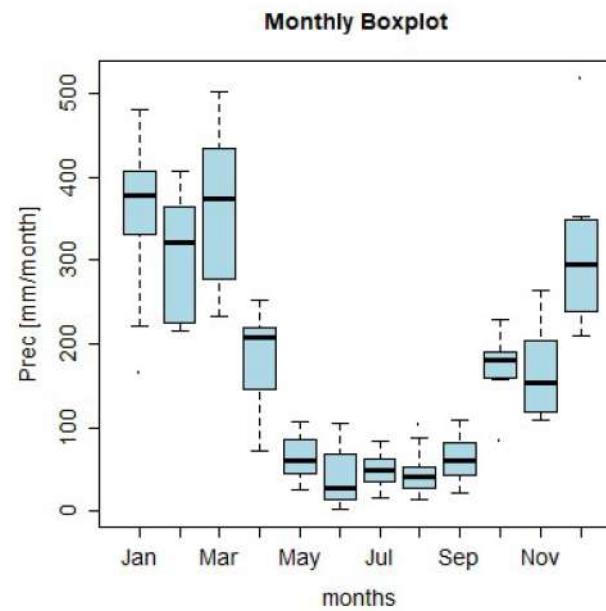
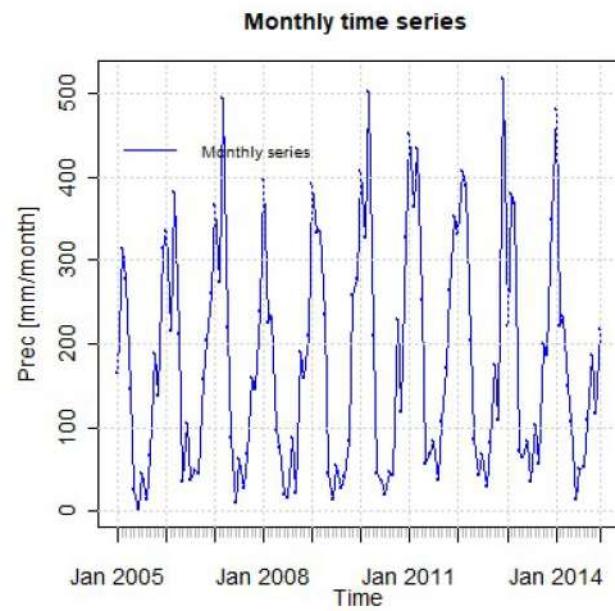
#Completacion forzada de datos

```
p_int<-na.interpolation(data.xts) #interpolation
p_kal<-na.kalman(data.xts) #Filtro de Kalman
p_mn<-na.mean(data.xts) #promedio total
plot(p_int[,1])
plot(p_kal[,1])
plot(p_mn[,1])
```

```
#Completacion de datos mediante correlaciones cruzadas
library(corrplot)
library(cutoffR)
#Correlacion cruzada
cor.cruzada.mensual <- cor(as.matrix(data.monthly),use="complete")
cor.test(data.monthly[,1],data.monthly[,2])
summary(cor.cruzada.mensual)
min(cor.cruzada.mensual)
#Grafico de matriz de correlaciones entre estaciones
corrplot(cor.cruzada.mensual, method="number",type = c("lower"),mar = c(4, 2,
3, 4))
data.cutoff <- data.frame(data.monthly,date=index(data.monthly),check.names =
FALSE)

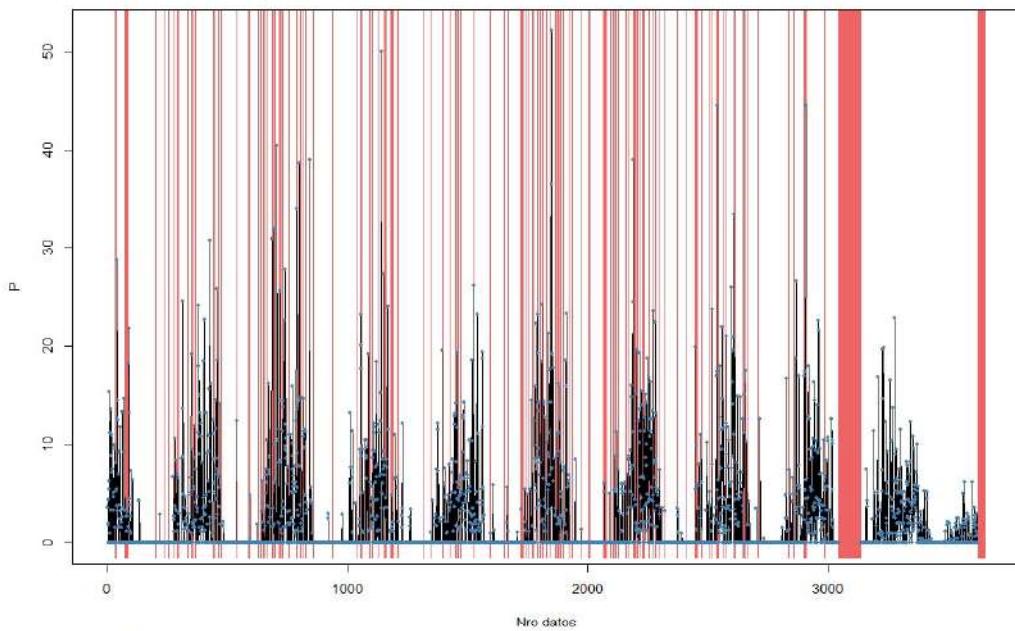
data.cutoff.comp <- cutoff(data = data.cutoff,method = c("correlation"),corr =
"spearman",cutoff = 0.7) #para tmin:cutoff = 0.8, tmax:cutoff = 0.7
data.mensual.comp <- as.xts(data.cutoff.comp,order.by = index(data.monthly))
xyplot(data.monthly,ylim=c(0,600)) # mensual
xyplot(data.mensual.comp,ylim=c(0,600))
library(latticeExtra)
st.sin <- xyplot(data.monthly,ylim=c(0,600),col="red", lwd=2)
st.com <- xyplot(data.mensual.comp,ylim=c(0,600),col="blue", lwd=2)
st.com + st.sin
```



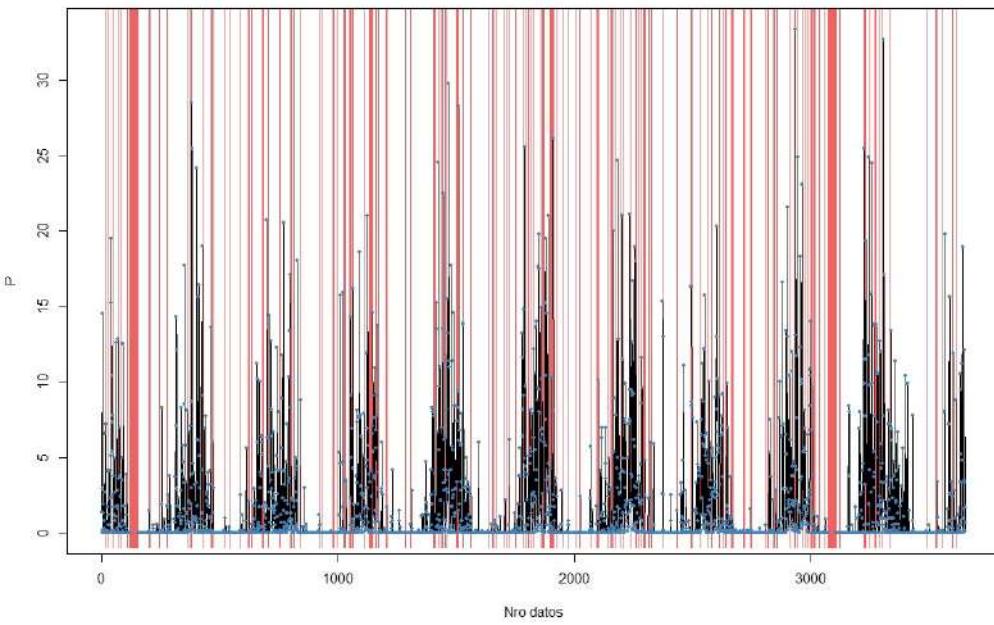


101

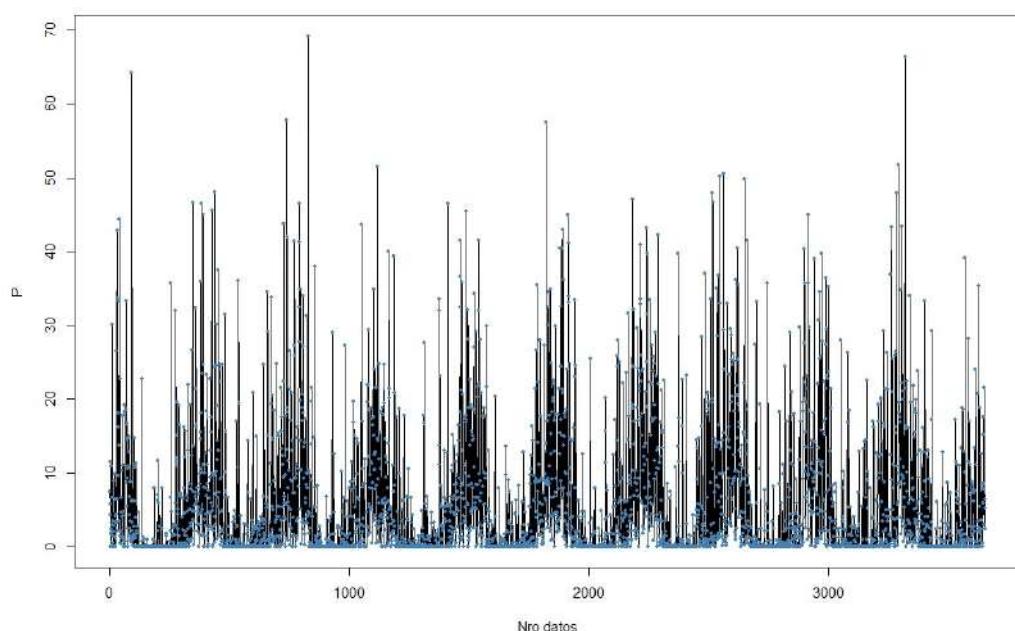
Distribution of NAs

**102**

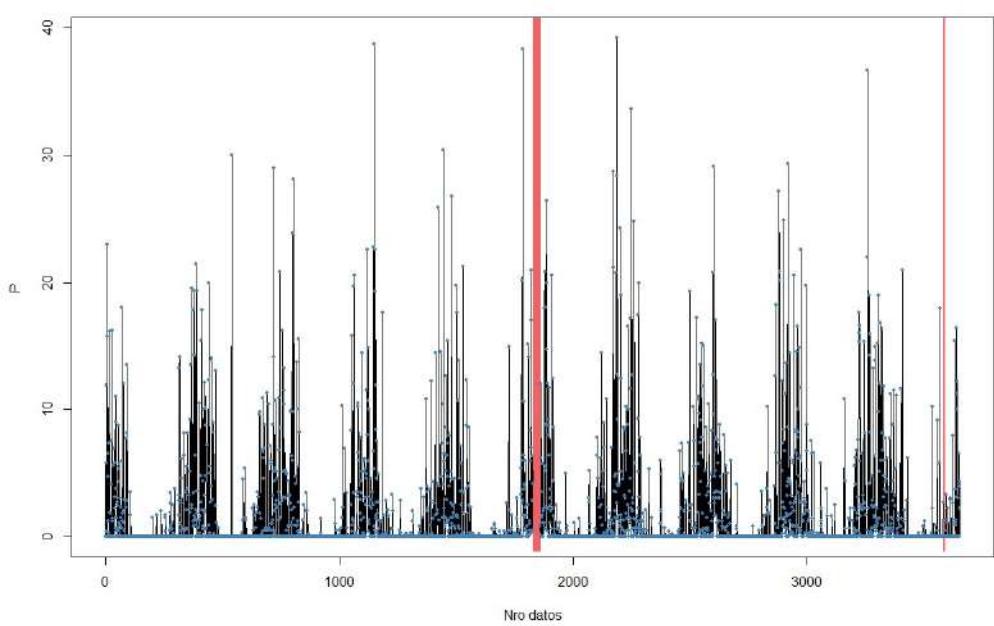
Distribution of NAs

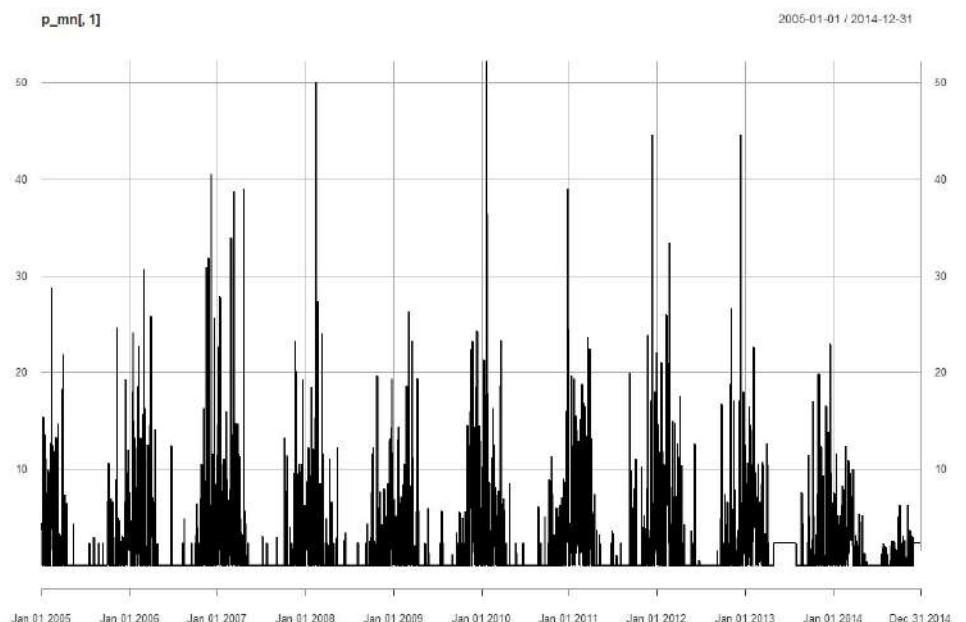
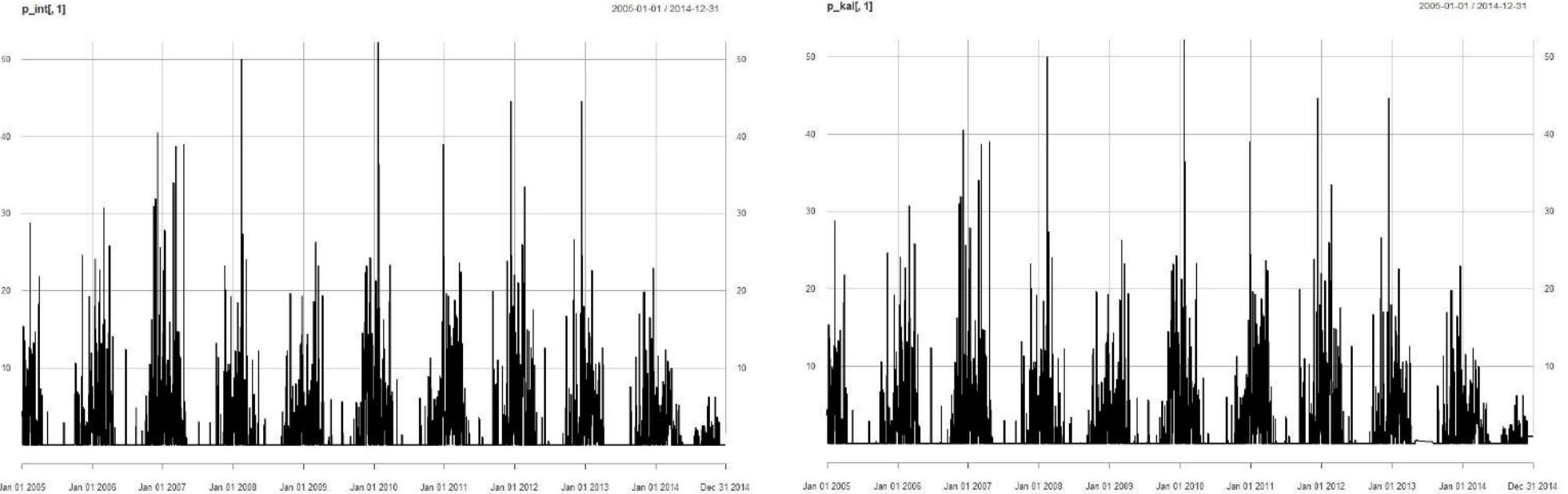
**103**

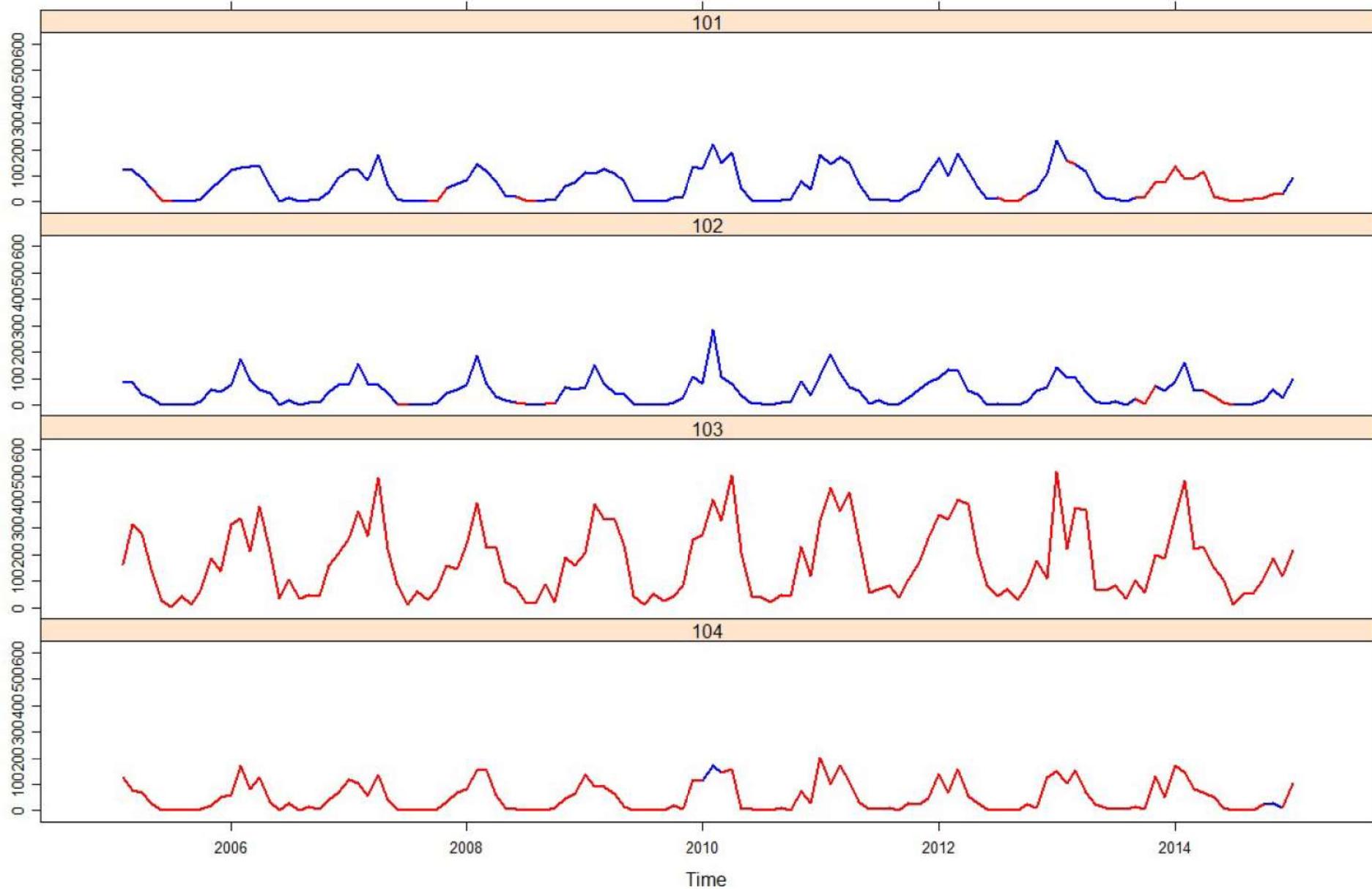
Distribution of NAs

**104**

Distribution of NAs

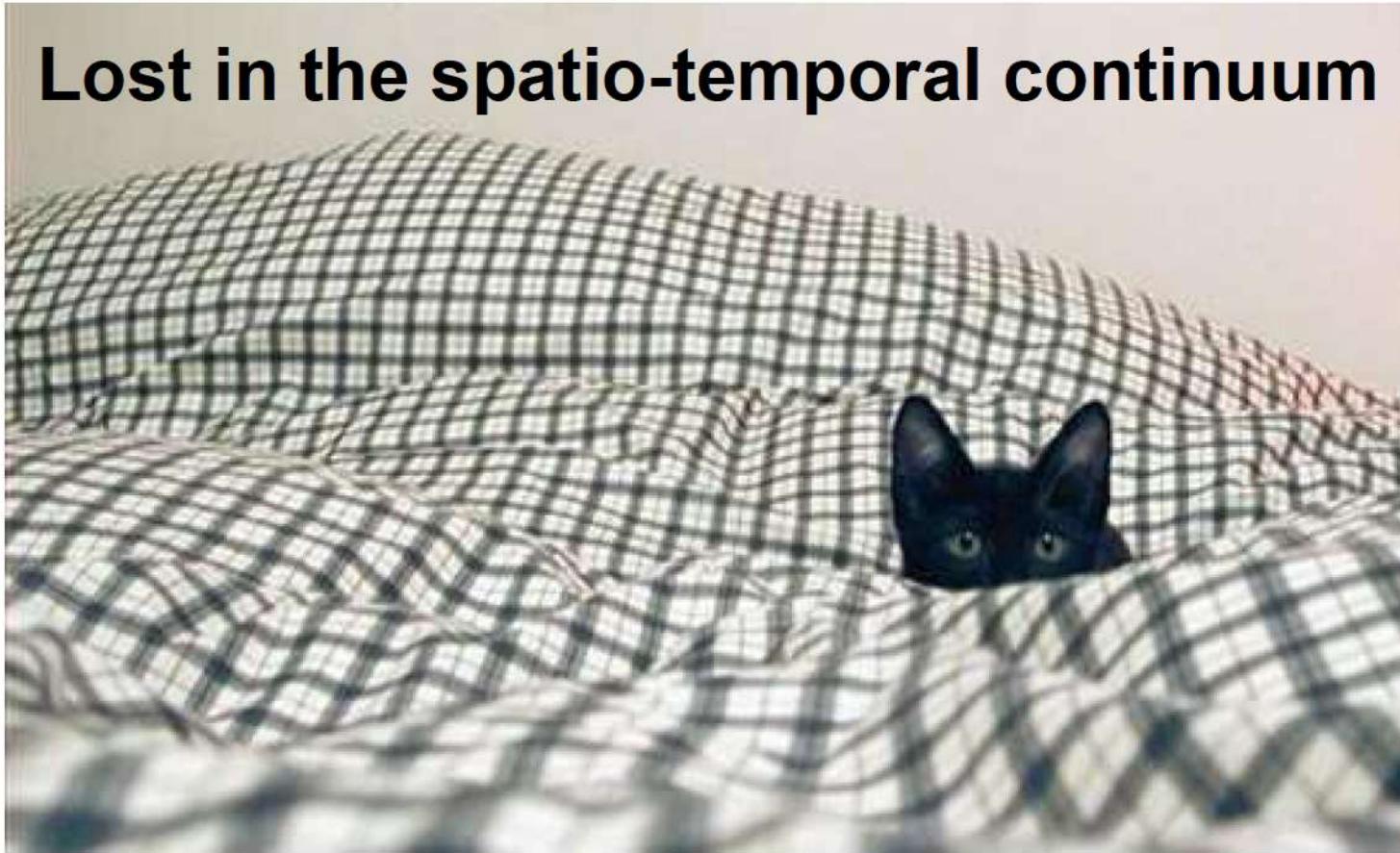




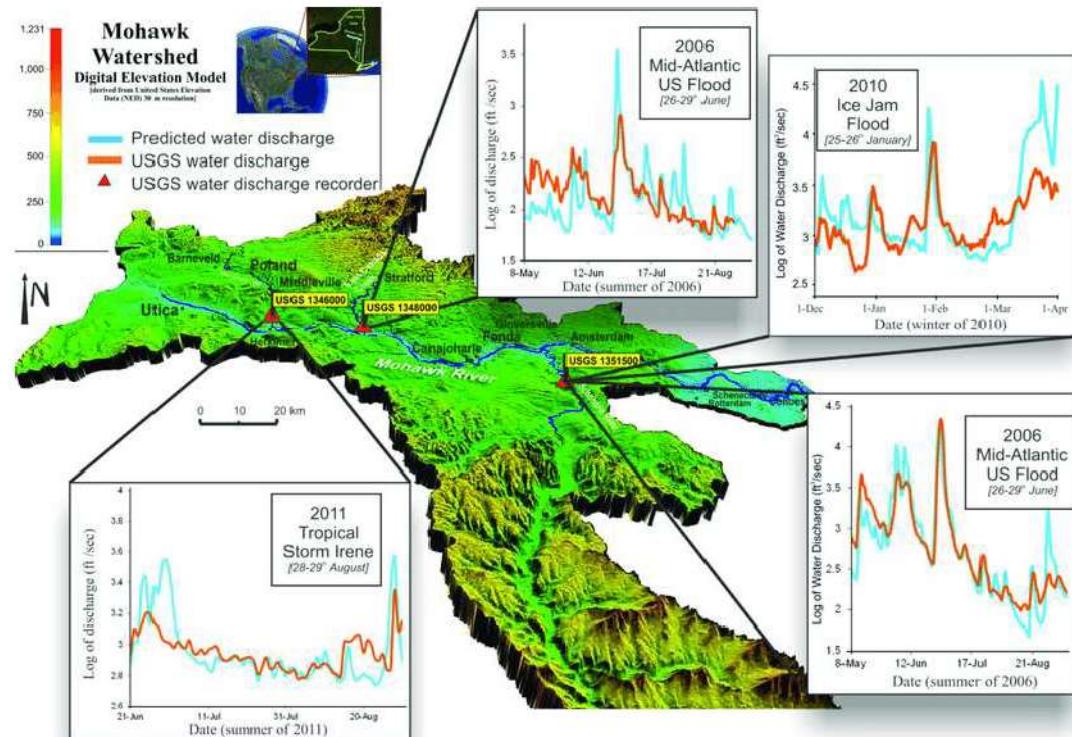
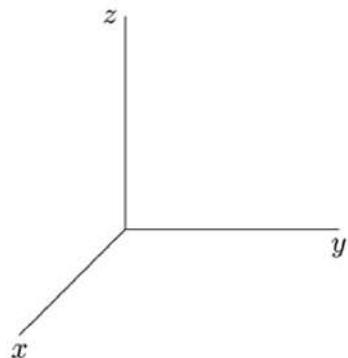


Datos completados en azul

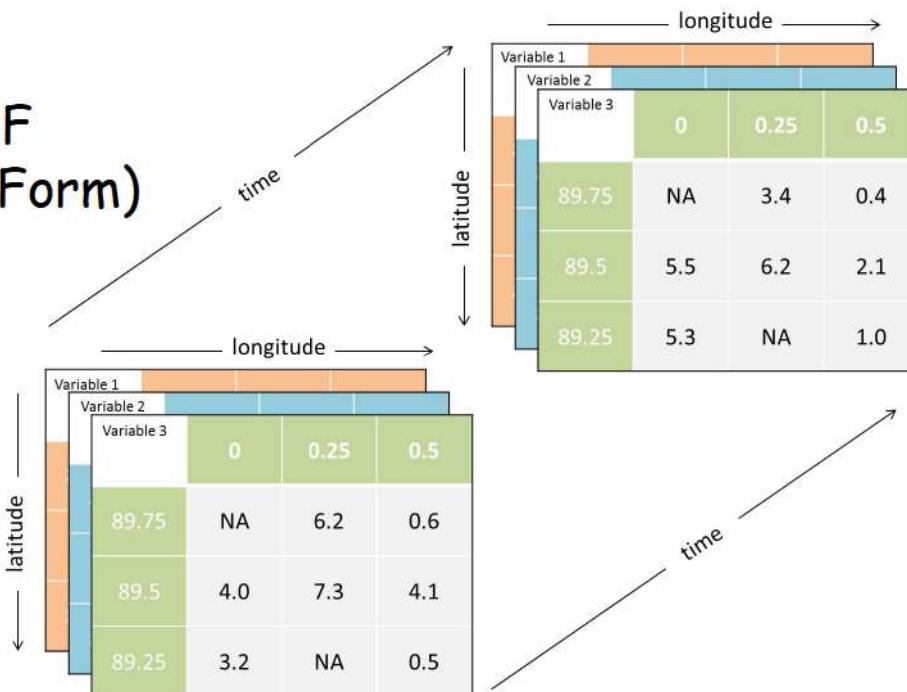
5. Introducción a las series espacio-temporales y el big data en hidrología



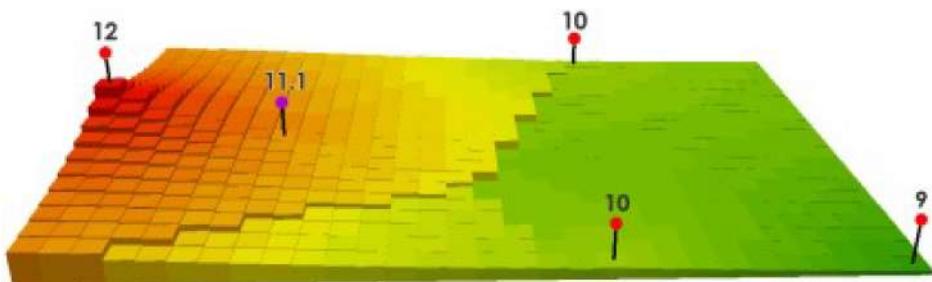
5.1. Espacialidad y temporalidad en hidrología



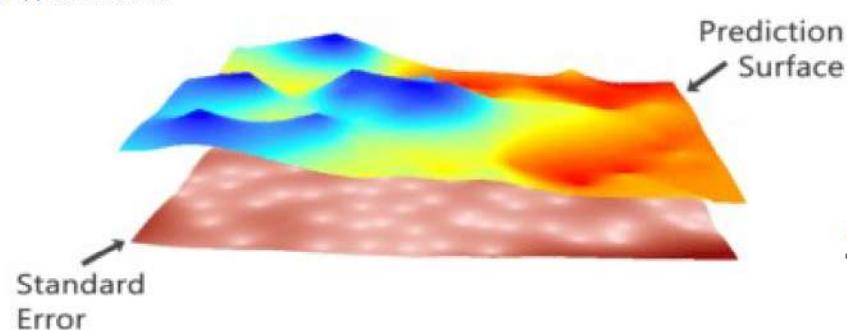
Estructura datos NetCDF
 (Network Common Data Form)



5.2. Métodos básicos y modelos



<https://acolita.com>



Los 2 primeros momentos deben ser deducidos de los datos

✓ **Estacionaridad**

$$E[(z(\mathbf{x}) - m)(z(\mathbf{x}') - m)] = R(\mathbf{x} - \mathbf{x}')$$

✓ **Isotropía**

$$E[(z(\mathbf{x}) - m)(z(\mathbf{x}') - m)] = R(|\mathbf{x} - \mathbf{x}'|)$$

$$|\mathbf{x} - \mathbf{x}'| = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + (x_3 - x'_3)^2}$$

Primeros dos momentos

Valor esperado (E) de z en la ubicación \mathbf{x}

$$E[z(\mathbf{x})] = P_1 z(\mathbf{x}; 1) + P_2 z(\mathbf{x}; 2) + \dots = \sum_i P_i z(\mathbf{x}; i)$$

✓ **Función promedio**

$$m(\mathbf{x}) = E[z(\mathbf{x})]$$

✓ **Función covarianza**

$$R(\mathbf{x}, \mathbf{x}') = E[(z(\mathbf{x}) - m(\mathbf{x}))(z(\mathbf{x}') - m(\mathbf{x}'))]$$

Coeficiente de correlación entre $z(\mathbf{x})$ y $z(\mathbf{x}')$

$$\rho(\mathbf{x}, \mathbf{x}') = \frac{R(\mathbf{x}, \mathbf{x}')}{\sigma(\mathbf{x}) \sigma(\mathbf{x}')}$$

Modelo lineal de predicción de $z(\mathbf{x}')$, con $z(\mathbf{x})$ como observado

$$m(\mathbf{x}') + \rho(\mathbf{x}, \mathbf{x}') \frac{\sigma(\mathbf{x}')}{\sigma(\mathbf{x})} [z(\mathbf{x}) - m(\mathbf{x})]$$

Modelo intrínseco

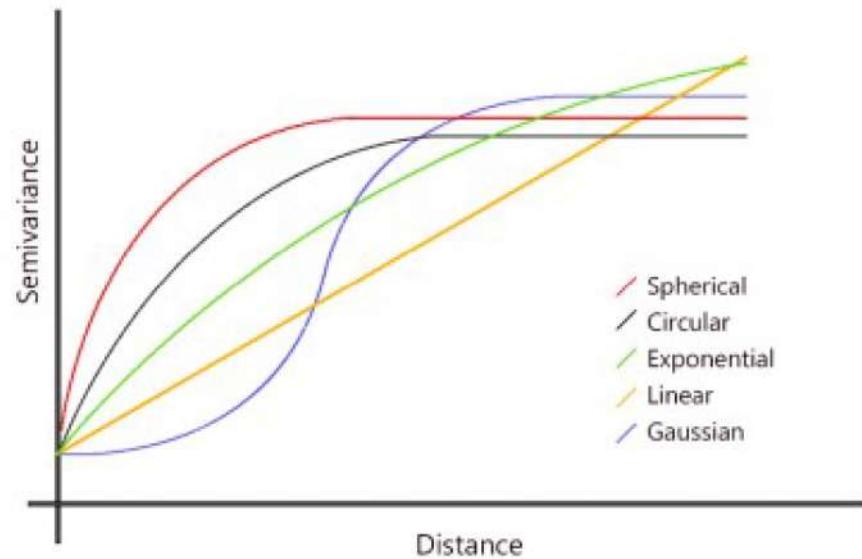
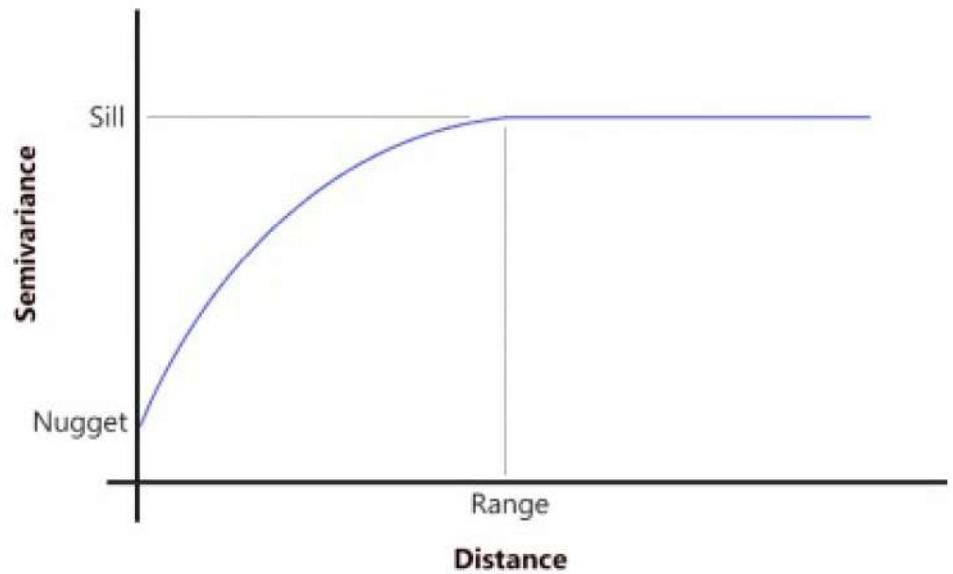
$$E[z(\mathbf{x}) - z(\mathbf{x}')] = 0$$

variograma $2\gamma(\mathbf{h}) = E[(z(\mathbf{x}) - z(\mathbf{x}'))^2]$

semivariograma

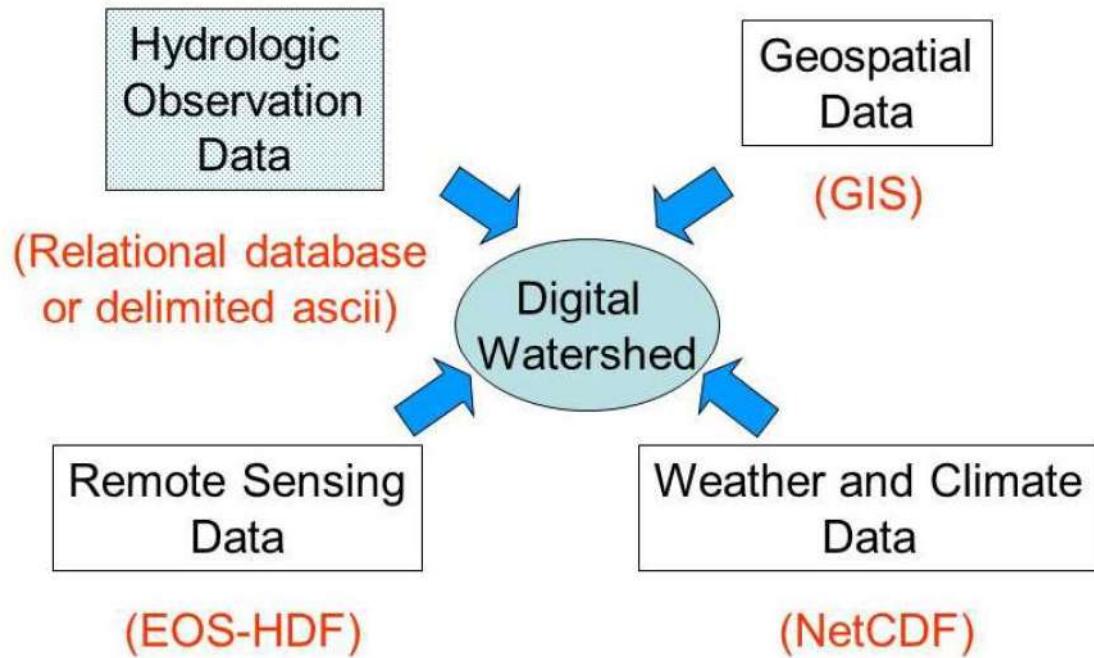
$$2\gamma(\mathbf{h}) = 2R(\mathbf{0}) - 2R(\mathbf{h})$$

El semivariograma cuantifica la "autocorrelación", graficando la varianza de todos los pares de datos según la distancia. Entre puntos cercanos tendrán una pequeña semivarianza. Mientras que puntos lejanos tendrán menos relacionadas y una alta semivarianza. A cierta distancia (rango), la autocorrelación se vuelve independiente.

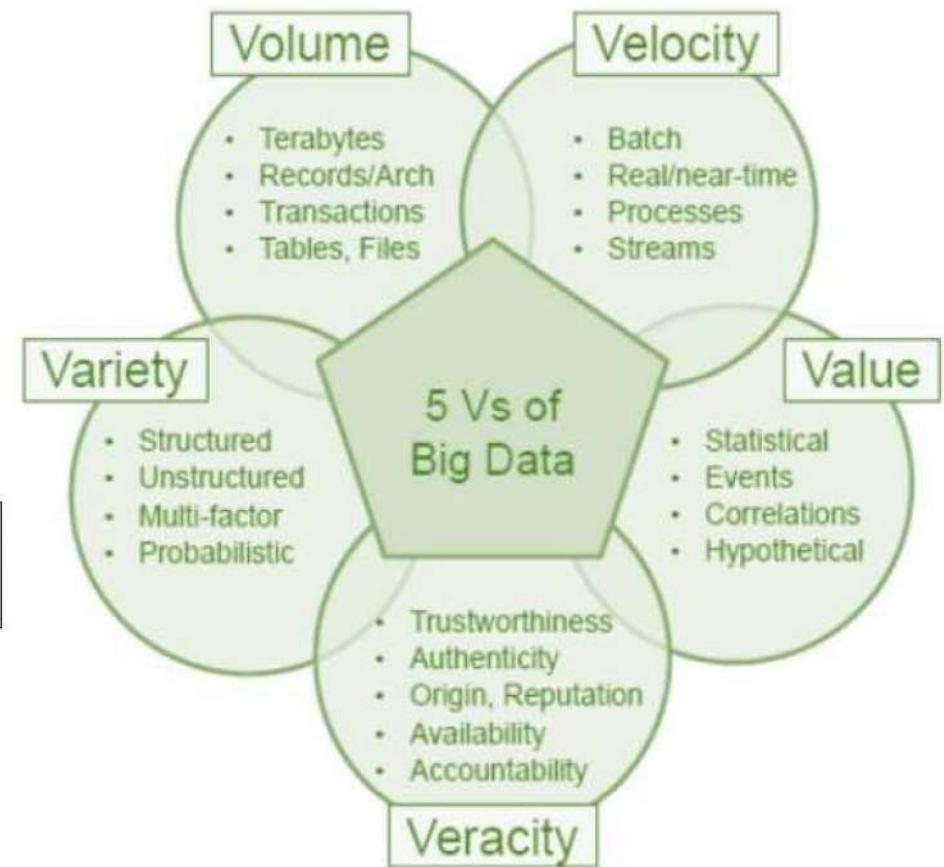


- ✓ Análisis exploratorio de datos
- ✓ Eliminar tendencias
- ✓ Evitar el eje principal de anisotropía
- ✓ Ajuste del modelo
- ✓ Aplicación de Kriging

5.3. Big data en hidrología



Maidment (1999) CE 394K.2 Surface Water Hydrology



6. Aplicaciones geoespaciales en R

Spatio-temporal R environment



F. Lectura de datos grillados netcdf (*.nc)

-Instalar librería TOOLS - INSTALL PACKAGES: *raster*

-Descargar archivos <https://esgf-node.llnl.gov/search/cmip5/>

pr_Amon_HadGEM2-ES_rcp45_r1i1p1_200512-203011

tasmax_Amon_HadGEM2-ES_rcp45_r1i1p1_200512-203011,

tasmin_Amon_HadGEM2-ES_rcp45_r1i1p1_200512-203011

Ejercicio F

Extraer las series de tiempo de precipitación mensual, temperatura mensual máxima y mínima desde el modelo de circulación general HadGEM2-ES para la cuenca media del río Rímac.

Considerar el escenario cuasi pesimista RCP 4.5 de emisión de gases de efecto invernadero para el periodo 2005 al 2030.

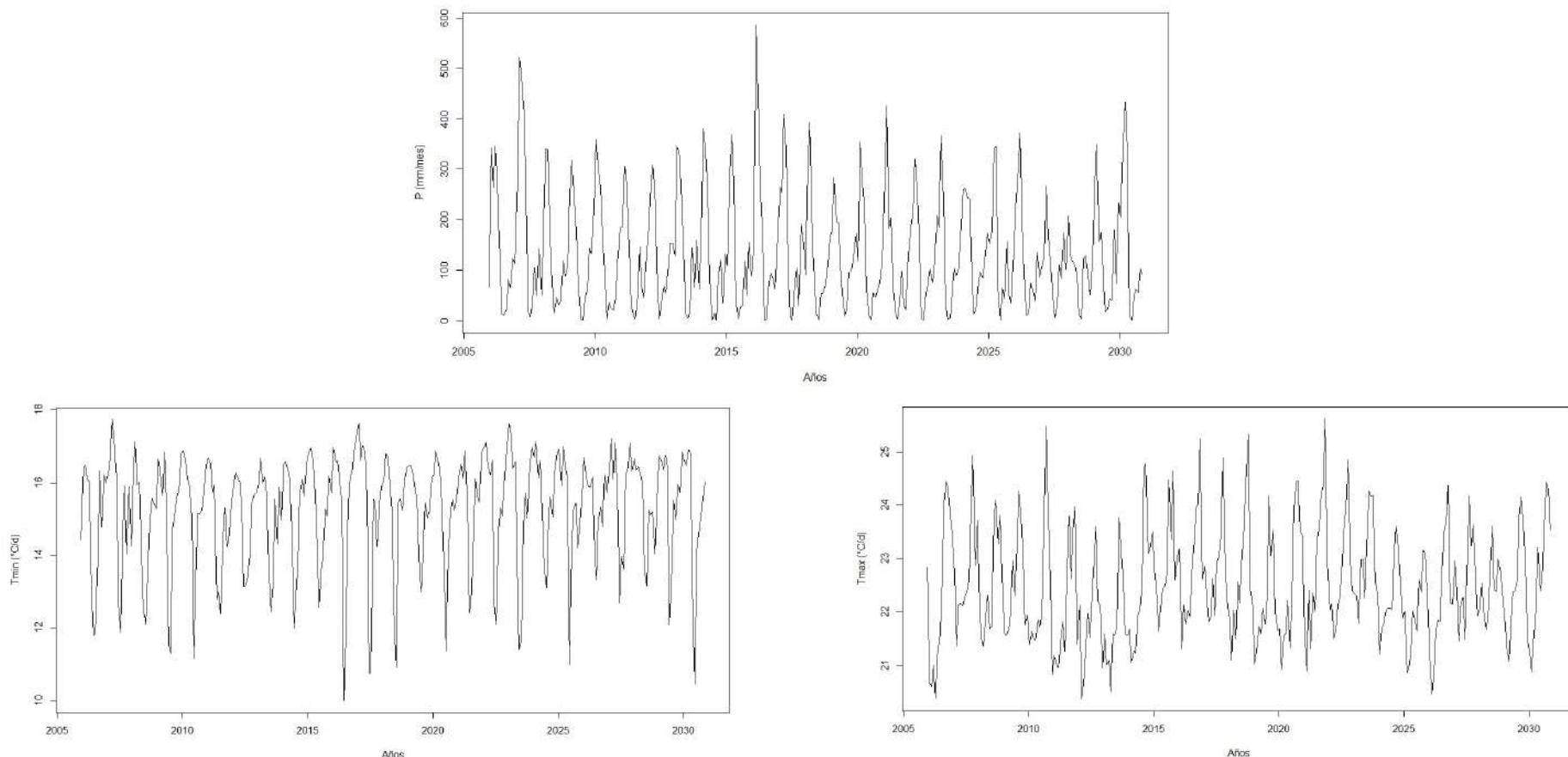
Respuestas F

```
library(raster)
# Comando brick lee las capas contenidas en el archivo ncdf *.nc, (i.e. pr: precipitación mensual).
# Archivo que contiene parte del registro histórico y proyección de precipitación a corto plazo
# (2005 - 2030) del modelo HadGEM2-ES para el escenario RCP4.5
b <- brick("C:/lab3/pr_Amon_HadGEM2-ES_rcp45_r1i1p1_200512-203011.nc", varname = "pr")
b # Para ver especificaciones del raster obtenido. Ejm su resolución es: 1.875° x 1.25° !
plot(b) # Visualización de la precipitación global
# Asignando índices
idx <- getZ(b)
# Indicar coordenadas y extraer los valores y convertirlos de kg/m2/s a mm/mes
# En general formato de longitud GCM = longitud WGS84 (ej. Google Earth) + 360°
# Ejemplo para la cuenca media del Rimac.
coords <- matrix(c(283.6, -11.8), ncol = 2) # 283.6 es la longitud (-76.4° en WGS84) y -11.8 la latitud
vals <- extract(b, coords, df=T)*86400*30.5 # Extrayendo las coordenadas y conversion
# Fijar fechas y datos en un solo archivo dataframe
df <- data.frame(idx, t(vals)[-1,])
rownames(df) <- NULL
names(df) <- c('date','value')
# Verificar el archivo
head(df)
# Plotear la serie
plot(df, type="l", xlab="Años", ylab="P (mm/mes)")
```

```

# Hacer los siguientes cambios al código anterior
# Lectura de la temperatura mínima: "tasmin" (e.g. tasmax para la temperatura maxima)
b <- brick("C:/lab3/hadgem2-es/tasmin_Amon_HadGEM2-ES_rcp45_r1i1p1_203012-
205511.nc", varname = "tasmin")
# Conversión de datos K/d a °C/d
vals <- extract(b, coords, df=T)-273.15
# Plotear con las etiquetas correctas
plot(df, type="l", xlab="Años", ylab="Tmin (°C/d)")

```



G. Extracción de datos netcdf para una cuenca hidrográfica

- Instalar librería TOOLS - INSTALL PACKAGES: easypackages, sp, rgdal, raster, lattice, latticeExtra, ncdf4
- Revisión de archivos: Cuenca del río Chillón (chillon.shp y archivos SIG anexos); producto grillado de precipitación mensual PISCO (Peruvian Interpolate data of the SENAMHI's Climatological and hydrological Observations, SENAMHI) desde 1981 al 2016 (PISCOV3-MONTHLY.nc).

Ejercicio G

Extraer la serie de tiempo de precipitación mensual agregado para toda la cuenca del río Chillón, desde el producto rasterizado PISCO para el periodo 1981 al 2016. Estandarizar el trabajo en el sistema de coordenadas geográficas WGS84.

Respuestas G

```
#Lectura de un polígono en formato shape
library(easypackages)
library(sp)
library(rgdal)
library(raster)
library(lattice)
library(latticeExtra)
library(ncdf4)
# Definiendo un directorio de trabajo donde se encuentren los archivos input
descargados
setwd("C:/lab2")
# Leyendo el polígono chillon.shp (layer: chillon) desde la carpeta "files" con data
source name (dsn)
cuenca.shape <- readOGR(dsn="files", layer="chillon")
# Conociendo el tipo de objeto después de la importación
class(cuenca.shape)
# Visualizando la cuenca con color cyan, se aprecia que se encuentra en coordenadas
geográficas
plot(cuenca.shape, axes=T, col=c("cyan"))
# Conociendo el contenido del polígono importado
head(cuenca.shape@data)
```

```
# Transformando a UTM zona 18, WGS84 y visualizando la conversión  
cuenca.utm <- spTransform(cuenca.shape, CRS("+proj=utm +zone=18 +ellps=WGS84 +south  
+datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0"))  
plot(cuenca.utm, axes=T, asp=1)  
# Reproyectando a Geograficas WGS84 y visualizando la reconversión  
cuenca.wgs <- spTransform(cuenca.utm, CRS("+proj=longlat +ellps=WGS84"))  
plot(cuenca.wgs, axes=T, asp=1)
```

####METODO 1

```
# Extrayendo datos y visualizando el producto ráster PISCO  
r <- stack("PISCOV3-MONTHLY.nc") # El archivo *.nc debe estar en la carpeta lab2, no en  
subcarpetas  
# Visualizando la precipitación espacial del primer mes de 1981.  
plot(r[[1]])  
# Ploteando la cuenca del río Chillón dentro del mapa de precipitaciones  
plot(cuenca.wgs, add=T)  
# Delimitando el área de estudio al cuadrante que ocupa el río Chillón  
r.chillon <- crop(r, cuenca.wgs, snap="out")  
# Ploteando el primer mes de 1981 en el cuadrante que ocupa el río Chillón  
plot(r.chillon[[1]])  
# Delimitando el área de estudio a la cuenca del río Chillón  
r.chillon <- mask(r.chillon, cuenca.wgs)  
# Ploteando los meses de enero a diciembre de 1981  
plot(r.chillon[[1:12]])  
# Ploteando todos los meses del ráster (431 datos) delimitado por la cuenca del río Chillón  
spplot(r.chillon, col.regions = rev(terrain.colors(100)))
```

####METODO2

Extrayendo datos y visualizando

Pisco.prec.brick <- brick("PISCOV3-MONTHLY.nc") # El archivo *.nc debe estar en la carpeta lab2, no en subcarpetas

Pisco.prec.brick

nlayers(Pisco.prec.brick)

Ploteando los primeros 12 meses de 1981 de la precipitación para todo el Perú

plot(Pisco.prec.brick[[1:12]])

Extrayendo los datos y promediando todas las grillas de la cuenca del Chillón

pp.cuenca.mensual <- extract(Pisco.prec.brick, cuenca.wgs, fun=mean)

colnames(pp.cuenca.mensual) <- 1:ncol(pp.cuenca.mensual)

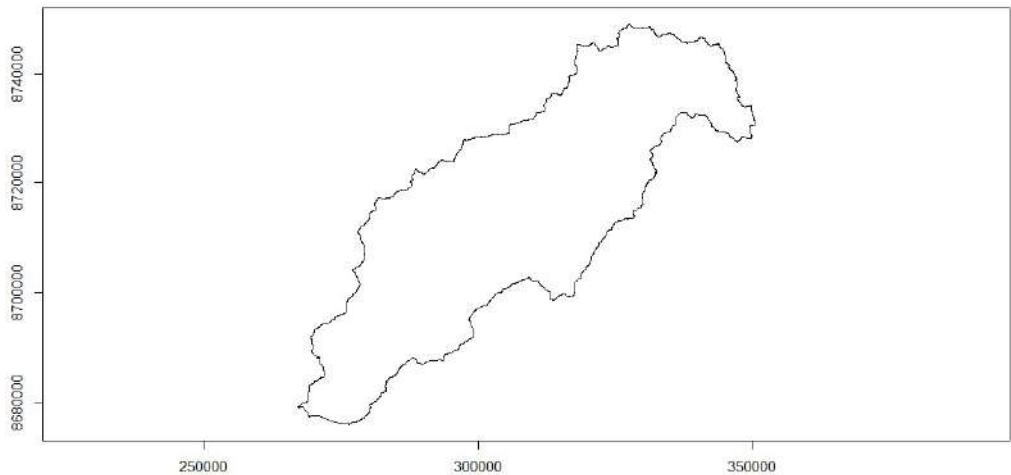
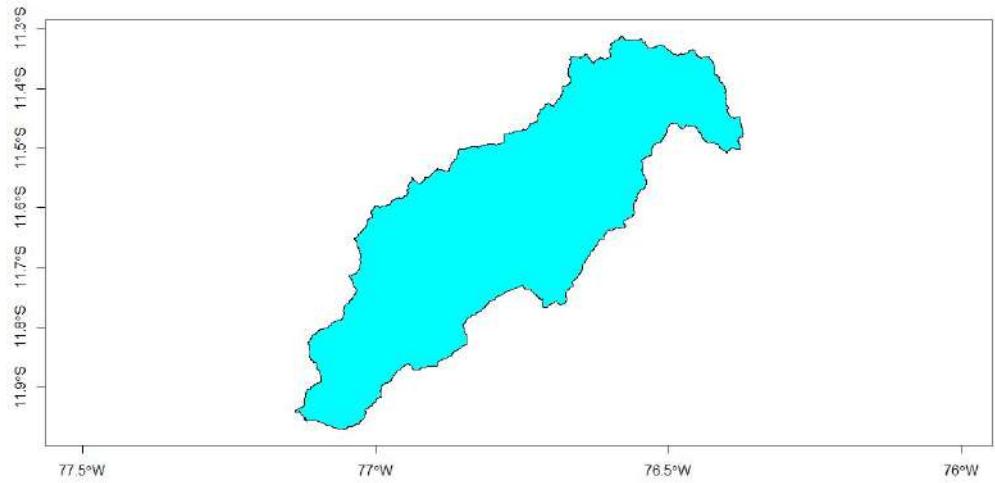
Visualizando los 431 datos promediados

View(pp.cuenca.mensual)

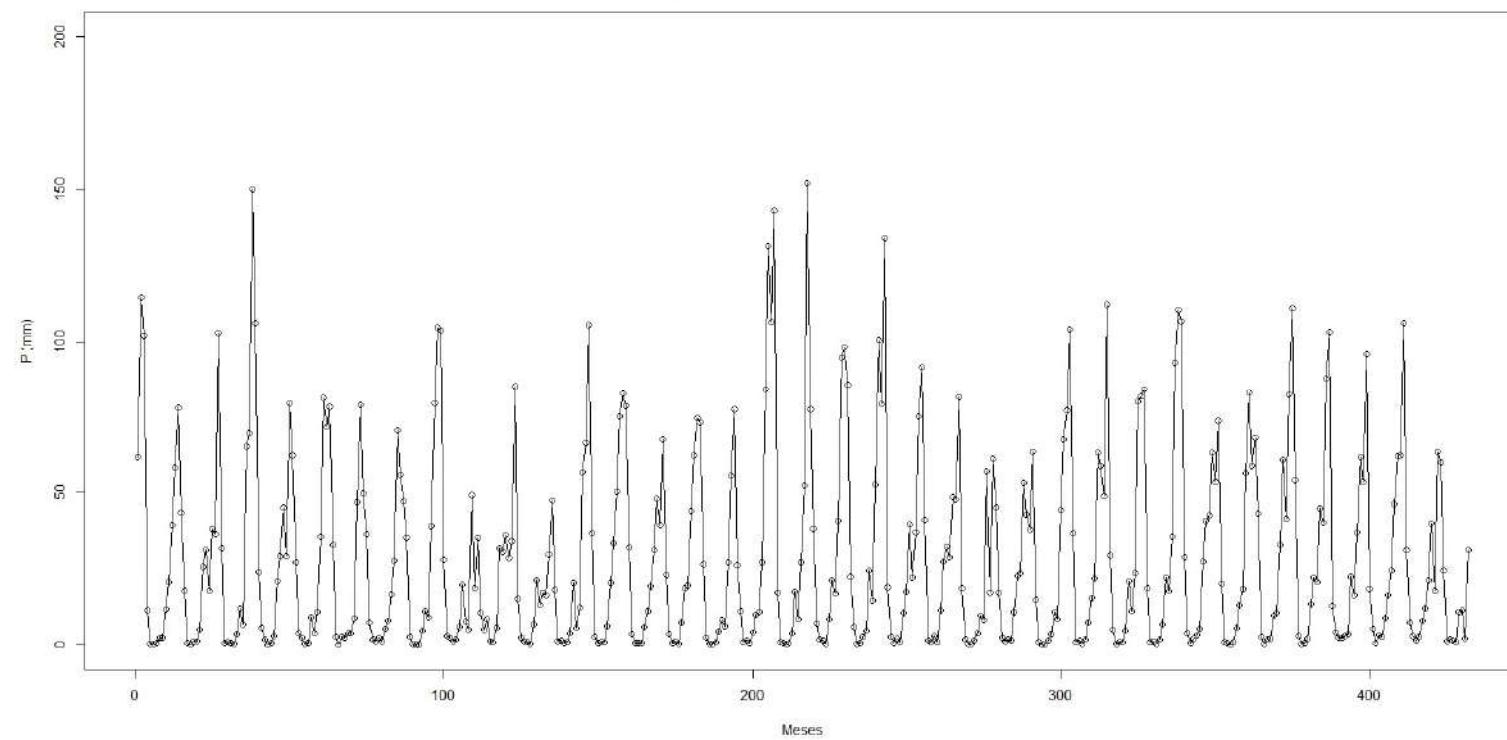
range(pp.cuenca.mensual)

Ploteando la serie de los 431 valores de precipitación mensual promedio areal

plot(pp.cuenca.mensual[1,], type="o", col="1", ylim=c(0,200), ylab="P (mm)", xlab = "Meses", main="Precipitacion promedio areal - Chillón (mm)")

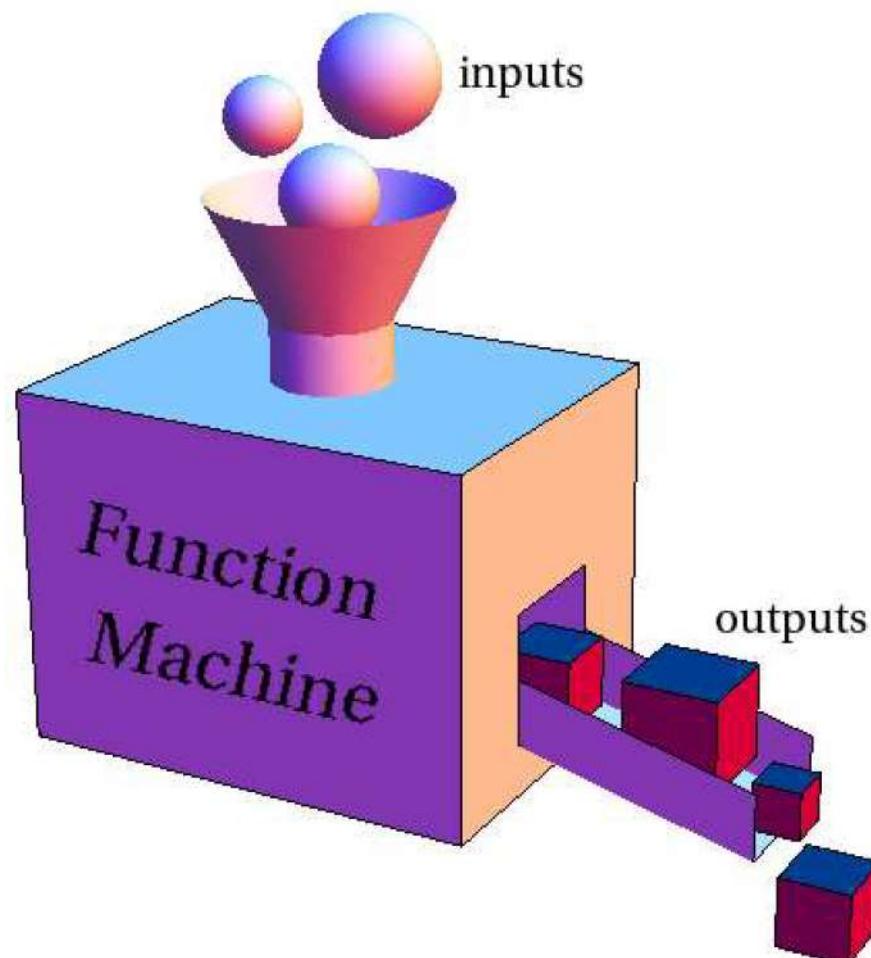


Precipitacion promedio areal - Chillon (mm)



7. Inside the libraries

7.1 Tratamiento de funciones



Funciones



Es un objeto en R, que puede tomar como entrada algunos objetos (llamados argumentos de función) y regresa un objeto de salida.

Definición de una función en R

```
name <- function(variables) {  
}
```

```
Nombre_de_función <- function (argumentos) {  
    #operaciones de la función  
}
```

Argumentos de funciones

Formales : Se incluyen en la definición de la función



Consultar argumentos con:
`args()` o `formals()`

Coincidentes : Coincidencia exacta
: Coincidencia parcial
: Coincidencia por posición

Argumento “...” : se usa cuando no se sabe el numero de argumentos
: se usa cuando algunas variables ingresaran a otras funciones

Ejercicio H

- A) Realizar la función "consecutivo", donde la variable de entrada sea "x" y retorne "x+1"
- B) Realizar la función "ope_arit", con variables de entrada "x", "y" y "operacion", el cual debe ser capaz de realizar las cuatro operaciones aritméticas entre "x" y "y" donde "operacion" puede ser "suma", "resta", "multiplicación" o "división". Y probar para $\text{ope_arit}(x = 20, y = 5, \text{operacion} = \text{"suma"})$
- C) Sobre la ultima función responder: ¿los resultados de: $\text{ope_arit}(y = 3, x = 15, \text{"resta"})$, $\text{ope_arit}(15, 3, \text{"resta"})$ y $\text{ope_arit}(\text{operacion} = \text{"resta"}, 15, 3)$ son iguales?
- D) Realizar la función a_mayúscula, donde la cantidad de variables (tipo caracteres) de entrada sea indeterminado y sea capaz de concatenarlos y convertirlos a mayuscula., pruebe para:
`a_mayusculas("Congreso", "Nacional", "del", "Agua").`
- E) Implementar la función numeros_por_vocales, la cual recibe un numero arbitrario de argumentos, y sea capaz de concatenar con espacios en blanco, además se reemplace las vocales "aeiou" por los números "12345" y prueba para: `numeros_por_vocales("Capa", "Nivel", "orden")`.
- F) Implementar la función "operador_binario", el cual recibe una función, y dos argumentos "x" y "y", y regresa la evaluación de la función pasándole como primer argumento a "x" y como segundo a "y", nota: el orden de los argumentos es importante; probar para: `operador_binario(`%/%`, 7, 2)`.

Respuesta H

```
> consecutivo <- function(x) { #Nombre y argumentos
  x + 1 #Operación
}
consecutivo(3) #Ejecución de la función
formals(consecutivo) #Verificar las variables
> ope_arit <- function(x,y,operacion){
  if (operacion == "suma") {
    return(x+y) #la función return, regresa resultados especificados
  }
  if (operacion == "resta") {
    return(x-y)
  }
  if (operacion == "multiplicacion") {
    return(x*y)
  } else (operacion == "division")
  return(x/y)
}
ope_arit(x = 20,y = 5,operacion = "suma") [1] 25
> ope_arit(15,3,"resta") [1] 12
ope_arit(y = 3,x = 15,"resta") [1] 12
ope_arit(operacion = "resta",15,3) [1] 12
```

Respuesta H (continuación)

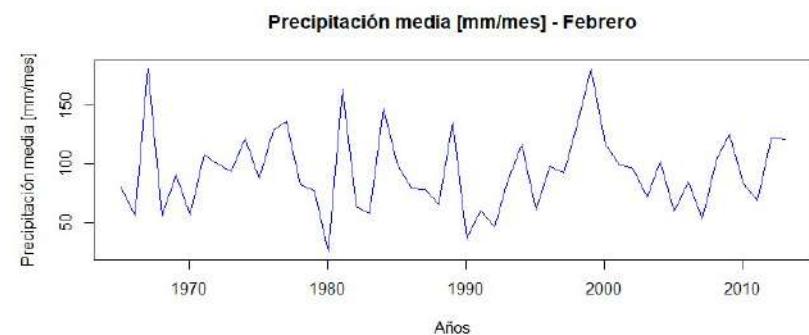
```
➤ a_mayusculas <- function(...){  
  frase <- paste(...) # La funcion paste() concatena los elementos (...)  
  toupper(frase) # La funcion toupper() convierte a mayúscula  
}  
a_mayusculas("congreso","nacional","del","agua")  
[1] "CONGRESO NACIONAL DEL AGUA"  
➤ numeros_por_vocales <- function(...){  
  # Nota 1: tolower(x) regresa la cadena x en minusculas.  
  # Nota 2: chartr(old, new, x) Traduce los caracteres especificados en old  
  # por los caracteres especificados en new que se encuentren en la cadena x.  
  # Ejemplo chartr("si", "5y", "sistemas") evalua a "5y5tema5".  
  chartr("aeiou","12345",tolower(paste(...,sep = " ")))  
}  
numeros_por_vocales("Capa","Nivel","orden")  
[1] "c1p1 n3v2l 4rd2n"  
➤ operador_binario <- function(fun, x, y){  
  fun(x,y)  
}  
operador_binario(`%/%` ,7,2) # `%/ %` permite obtener la división entera  
[1] 3
```

Ejercicio I

- A) Leer el archivo "dfP.Rdata" el cual contiene las precipitaciones de la cuenca de Pativilca.
- B) Visualizar el contenido del archivo
- C) Adicionar una columna llamada "MONTHS" y que contenga el nombre del mes de acuerdo a la fecha de la columna "dates"
- D) Filtrar los datos para el mes de "Enero" y almacenar en el objeto "df_enero".
- E) Realizar una función "prom_mes", con las siguiente características:
 - El mes será la variable de entrada.
 - Para dicho mes se debe calcular la precipitación media, considerar un promedio aritmético de las 13 estaciones.
 - Como salida de la función es un vector que contenga los valores de la precipitación media para el mes elegido de cada año, y que sea capaz de mostrar una gráfica de dichos valores, así como se muestra en el ejemplo.
- F) Ejecutar la función: prom_mes(mes = "Abril")

Ejemplo: prom_mes(mes=febrero)

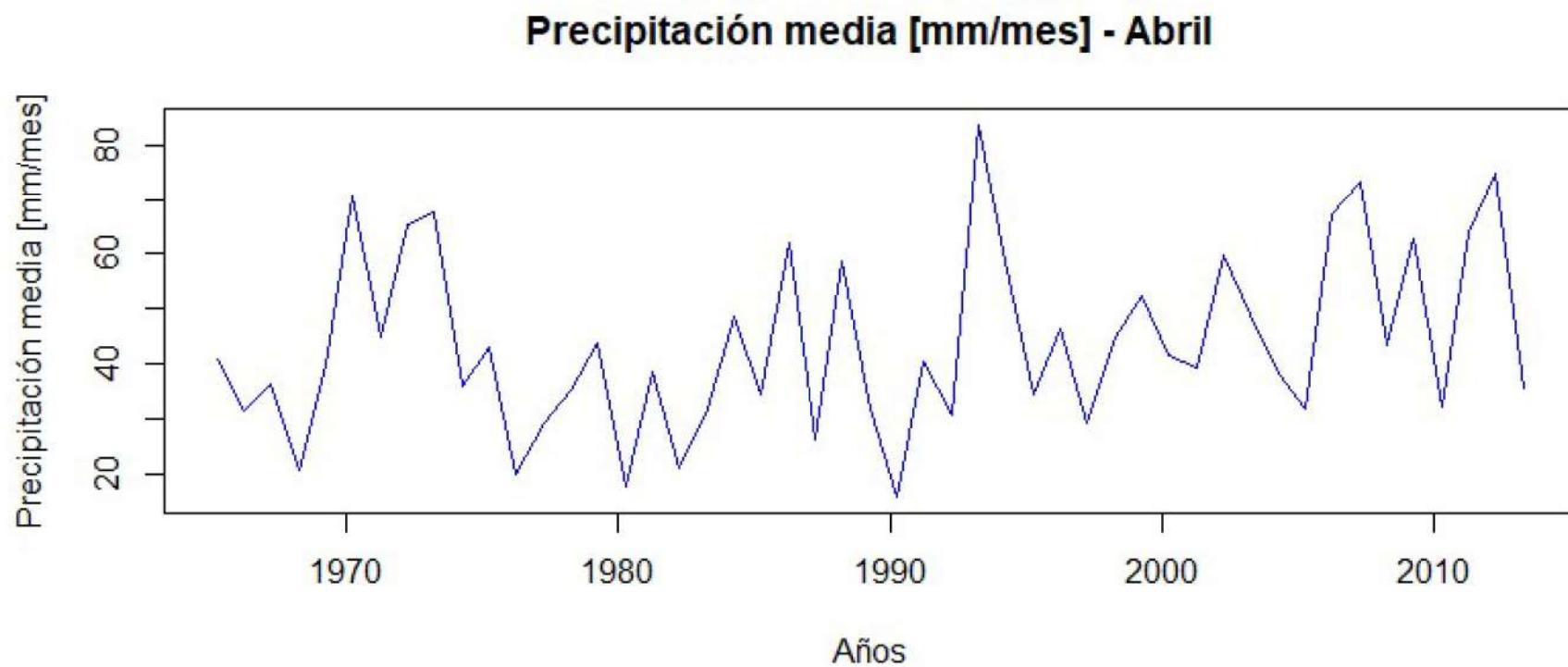
```
> prom_mes(mes = "Febrero")
[1] 80.53077 56.20769 180.79231 56.30769 90.64538
[12] 128.08462 136.07692 82.89923 77.20769 25.40769
[23] 78.49231 65.80769 135.01538 36.32308 60.59231
[34] 132.81538 181.46154 117.17692 99.47692 96.51538
[45] 125.26923 83.48308 68.61231 122.46154 120.72308
```



Respuesta I

```
> load("dfP.Rdata") #Cargar archivo tipo Rdata
> View(dfP)          #Visualizar el archivo en formato tabla
> dfP$MONTHS <- months(dfP[,1]) #Adicionar una columna que contenga el
  nombre del mes
> df_enero <- subset(dfP,dfP$MONTHS == "Enero") #Filtrado de datos
> prom_mes <- function(mes){ #nombre y variable de entrada
  prom <- vector() #Vector que almacenará los datos
  df_mes <- subset(dfP,months(dfP[,1]) == mes) #Filtrado para el mes elegido
  for (i in seq(1,length(dfP[,1])/12,1)) { #Secuencia de adición
    prom[i] <- mean(as.numeric(df_mes[i,2:14])) #cálculo del promedio de las
  precipitaciones
  }
  plot(df_mes[,1],prom,type = "l",col="blue",xlab = "Años",ylab = "Precipitación
  media [mm/mes]",
       main = paste("Precipitación media [mm/mes]",mes,sep = " - "))
  return(prom) #Retorno de la variable de salida
}
> prom_mes(mes = "Abril") #Ejecución de la función
```

```
> prom_mes(mes = "Abril") #Ejecución de la función
[1] 40.96154 31.55385 36.43846 20.97692 39.68462 70.42308 44.93846 65.53077 67.43077
[10] 36.10000 43.28462 20.07692 29.24615 35.19231 43.78462 17.83846 38.72308 21.13077
[19] 31.82308 48.67308 34.46923 62.04615 26.23846 58.82308 31.85385 15.93077 40.69231
[28] 31.04615 83.73077 58.80000 34.63846 46.48462 29.55385 44.98462 52.54615 41.71538
[37] 39.28462 59.66154 49.12308 38.39231 32.16154 67.22308 73.24615 43.56923 62.79231
[46] 32.38462 63.72308 74.54615 35.58462
```



Funciones apply, lapply y sapply

Función apply

Description

Returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

Usage

```
apply(X, MARGIN, FUN, ...)
```

Arguments

X an array, including a matrix.

MARGIN a vector giving the subscripts which the function will be applied over. E.g., for a matrix 1 indicates rows, 2 indicates columns. c(1, 2) indicates rows and columns. Where X has named dimnames, it can be a character vector selecting dimension names.

FUN the function to be applied: see ‘Details’. In the case of functions like +, %*%, etc., the function name must be backquoted or quoted.

... optional arguments to FUN.

Notas:

- Son un grupo de funciones vectorizadas y su ejecución se lleva en paralelo.
- Evitan el uso explícito de bucles (for, while y repeat).
- Actúan sobre una entrada: lista, matriz o vector
- Opera sobre subconjunto de datos.

Tipos de funciones como argumento

1. Funciones de agregación

Operan sobre un conjunto de datos y regresa un valor único (sum, mean, etc.)

2. Funciones de transformación

Extracción de subconjuntos

3. Funciones vectorizadas

Opera sobre Listas, vectores, matrices y arreglos.

Ejercicio J

- A) Hacer que la semilla de generación de números aleatorios sea el mismo en todas las computadoras.
- B) Generar una matriz de 5x8 con números aleatorios que tengan una distribución normal y asignar al objeto "x"
- C) Obtener la suma de cada columna de la matriz generada usando la función apply
- D) Verificar con la función sum para la primera columna de la matriz

Respuesta J

- `set.seed(100) #Se usa para generar los mismo aleatorios en todas las computadoras`
- `x <- matrix(rnorm(40),nrow = 5,ncol = 8) #genera matriz de 5x8`
- `apply(x, 2, sum) #uso de la funcion apply`
- `sum(x[,1]) #para comprobar`

```
> x
[1,] -0.50219235 [,1] 0.3186301 [,2] 0.08988614 [,3] -0.02931671 [,4] -0.4380900 [,5] -0.4384506 [,6] -0.09111356 [,7] -0.2217942
[2,] 0.13153117 -0.5817907 [,2] 0.09627446 [,3] -0.38885425 [,4] 0.7640606 [,5] -0.7202216 [,6] 1.75737562 [,7] 0.1829077
[3,] -0.07891709 0.7145327 -0.20163395 [,3] 0.51085626 [,4] 0.2619613 [,5] 0.2309445 [,6] -0.13792961 [,7] 0.4173233
[4,] 0.88678481 -0.8252594 0.73984050 [,4] -0.91381419 [,5] 0.7734046 [,6] -1.1577295 [,7] -0.11119350 [,8] 1.0654023
[5,] 0.11697127 -0.3598621 0.12337950 [,5] 2.31029682 [,6] -0.8143791 [,7] 0.2470760 [,8] -0.69001432
`->
> apply(x, 2, sum) #uso de la funcion apply
[1] 0.5541778 -0.7337494 0.8477467 1.4891679 0.5469574 -1.8383811 0.7271246 2.4140411
```

Función lapply

Notas:

- Se utiliza para aplicar una función a cada elemento de una lista y obtener otra lista como resultado.
- A diferencia de apply esta recibe como tercer parámetro los [parámetros de la función](#) que se usan como argumento.
- Opera sobre dataframes, listas o vectores.

Ejercicio K

- A) Generar una lista a partir de tres matriz A, B y C, donde A sea una matriz de 3x3 y contenga números enteros consecutivos del 1 al 9, y B otra de 4x3 y contenga números enteros consecutivos del 4 al 15; y C de 3x2 y contenga números enteros consecutivos del 8 al 13. asignar al objeto "lista".
- B) Extraer la segunda columna de todas las matrices antes generadas.

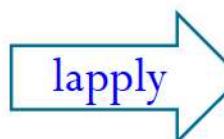
Respuesta K

```
> A <- matrix(1:9,3,3)
B <- matrix(4:15,4,3)
C <- matrix(8:13,3,2)
lista <- list(A,B,C)
> lapply(lista,"[,2]") #"[,2" se usa para extraer
```

```
> A
 [,1] [,2] [,3]
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
> B
 [,1] [,2] [,3]
[1,] 4 8 12
[2,] 5 9 13
[3,] 6 10 14
[4,] 7 11 15
> C
 [,1] [,2]
[1,] 8 11
[2,] 9 12
[3,] 10 13
```



```
> lista
[[1]]
 [,1] [,2] [,3]
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
[[2]]
 [,1] [,2] [,3]
[1,] 4 8 12
[2,] 5 9 13
[3,] 6 10 14
[4,] 7 11 15
[[3]]
 [,1] [,2]
[1,] 8 11
[2,] 9 12
[3,] 10 13
```



```
> lapply(lista,"[,2]")
[[1]]
[1] 4 5 6
[[2]]
[1] 8 9 10 11
[[3]]
[1] 11 12 13
```

Función sapply

Notas:

- Similar a lapply con la diferencia que retorna un vector.

Ejercicio L

- Generar una lista a partir de tres matriz A, B y C, donde A sea una matriz de 3x3 y contenga números enteros consecutivos del 1 al 9, y B otra de 4x3 y contenga números enteros consecutivos del 4 al 15; y C de 3x2 y contenga números enteros consecutivos del 8 al 13. asignar al objeto "lista".
- Extraer la segunda columna y la primera fila de todas las matrices antes generadas. Usando lapply y sapply
- Comparar los resultados

Respuesta L

```
> A <- matrix(1:9,3,3)
B <- matrix(4:15,4,3)
C <- matrix(8:13,3,2)
lista <- list(A,B,C)
> lapply(lista,"[,1,2) #"[ se usa para extraer
> sapply(lista,"[,1,2)
> Sapply trata de similiar la salida de lapply
```

```
> A
 [,1] [,2] [,3]
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
> B
 [,1] [,2] [,3]
[1,] 4 8 12
[2,] 5 9 13
[3,] 6 10 14
[4,] 7 11 15
> C
 [,1] [,2]
[1,] 8 11
[2,] 9 12
[3,] 10 13
```

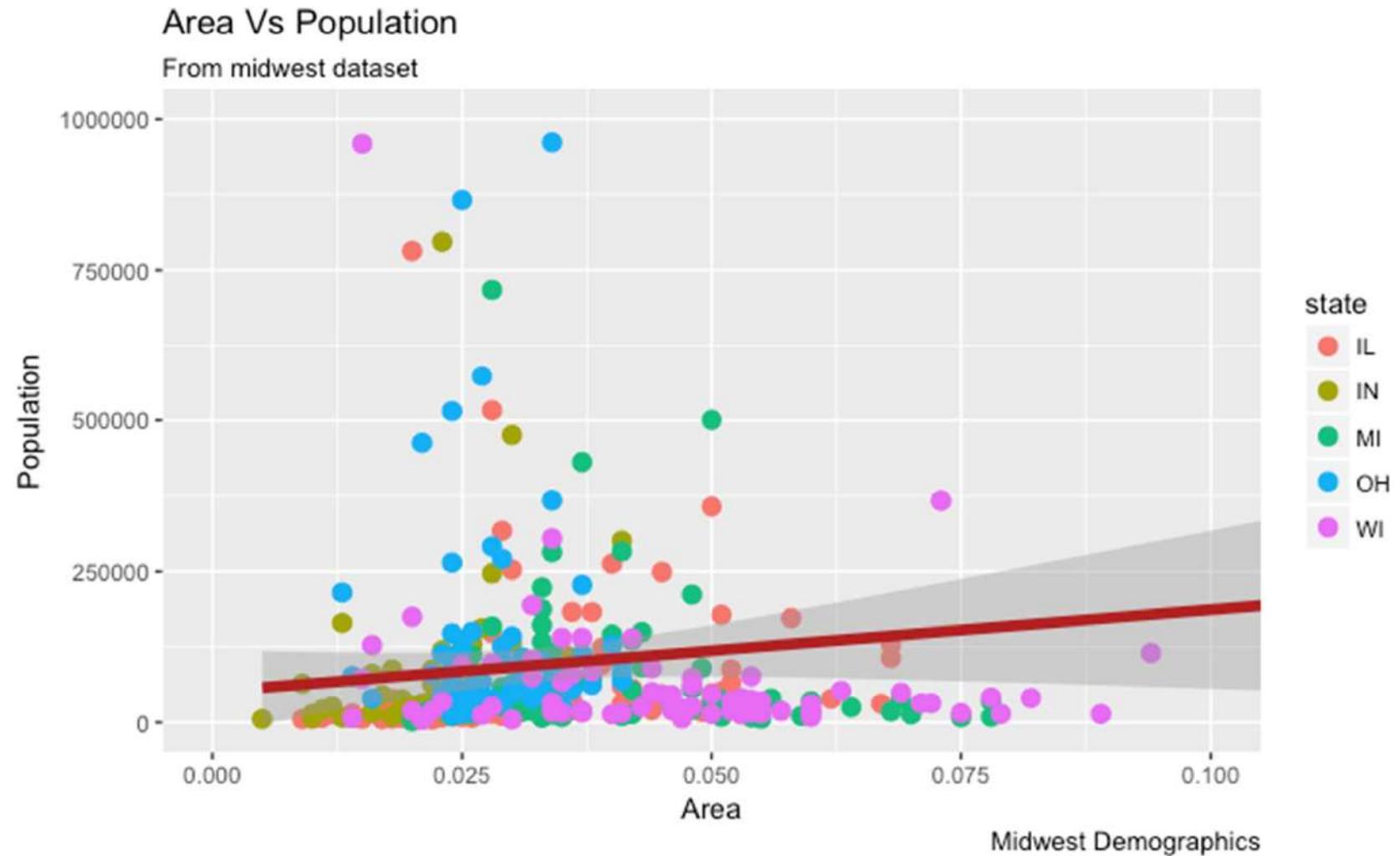


```
> lista
[[1]]
 [,1] [,2] [,3]
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
[[2]]
 [,1] [,2] [,3]
[1,] 4 8 12
[2,] 5 9 13
[3,] 6 10 14
[4,] 7 11 15
[[3]]
 [,1] [,2]
[1,] 8 11
[2,] 9 12
[3,] 10 13
```

```
> lapply(lista,"[,1,2)
[[1]]
[1] 4
[[2]]
[1] 8
[[3]]
[1] 11
```

```
> sapply(lista,"[,1,2)
[1] 4 8 11
```

7.2 Visualización en R



tidyverse

dplyr

tidyr

Permiten manipular dataframes

`filter (data, condición)`

permite filtrar filas

`mutate(data, var = ...)`

permite crear una variable

`arrange (data, var =)`

permite ordenar el df

de acuerdo a una variable

`select (data,nombre de campos)` selecciona campos

`gather(data,key,value,vars a sintetizar)`

sintetiza el df, uniendo varias variables en una sola

Anexo - Beyond the library

ggplot2

`ggplot(data, ...)` + CAPA 1 + CAPA 2 + CAPA 3 + ...

Geometría:

- + `geom_point()`
- + `geom_text()`
- + `geom_line()`
- + `geom_area()`
- + `geom_density()`
- + `geom_histogram()`
- + `geom_bar()`
- + `geom_boxplot()`

Escalas:

- + `scale_x_date()`
- + `scale_x_datetime()`
- + `scale_x_log10()`
- + `scale_x_reverse()`

Etiquetas:

- + `ggtitle()`
- + `xlab()`
- + `ylab()`

Otros:

- + `stat_ecdf`

Temas:

- + `theme_bw()`
- + `theme_classic()`
- + `theme_minimal()`



ggplot2

`ggplot (data, aes()) + capa1 (aes()) + ...`

Opción 1: global

Opción 2: local

La **data** ingresada siempre es un **data frame** !

Se deben llamar los elementos a usar (campos del data frame) mediante la función **aes()**

Múltiples gráficos ?

- | `facet_grid(. ~ fl)` *en columnas*
- | `facet_grid(year ~ .)` *en filas*
- | `facet_grid(year ~ fl)` *en filas y columnas*
- | `facet_wrap(~ fl)` *en arreglo rectangular*

M. Generando gráficos muy flexibles

Ejercicio M

En la consola de scripts, crear un código que realice lo siguiente:

- 1) Cargar la base de datos dfP de dfP.Rdata, correspondientes a las precipitaciones mensuales en estaciones de la Cuenca de Pativilca.
- 2) Crear el campo 'mes' en el data frame dfP.
- 3) Plotear un boxplot por meses para la estación CHIQUIAN.
- 3) Plotear un histograma para la estación de CAJATAMBO.
- 4) Plotear un diagrama de distribución de densidad para la estación CAJATAMBO
- 5) Plotear un scatterplot entre las estaciones CHIQUIAN y CAJATAMBO
- 6) Plotear la serie de tiempo para la estación CHIQUIAN

- 7) Reordenar dfP considerando como nuevos campos dates,mes,Estacion y Precipitacion

Respuestas M

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(ggrepel)

Sys.setenv(TZ='GMT')

load(file.choose())
load('D:/dfP.Rdata')          # se puede leer de ambas formas
class(dfP$dates)
str(dfP)

##### BOXPLOT #####
p <- ggplot(dfP)
p
dfP$mes <- format(dfP$dates,'%b')
#dfP <- dfP %>% mutate(mes=format(dates,'%b'))

p <- ggplot(dfP,aes(mes,CHIQUIAN)) + geom_boxplot() +
  xlab('Precipitacion') + ylab('Meses')
p

p <- ggplot(dfP,aes(reorder(mes,dates),CHIQUIAN)) + geom_boxplot() +
  xlab('Precipitacion') + ylab('Meses')
p

p + scale_y_continuous(trans='sqrt')
p + scale_y_continuous(trans='sqrt') + ggtitle('Estacion CHIQUIAN')
p + scale_y_continuous(trans='sqrt') + ggtitle('Estacion CHIQUIAN') + theme_bw()
```

Respuestas M (continuación)

```
theme_set(theme_bw())

##### HISTOGRAM #####
ggplot(dfP,aes(CAJATAMBO)) +
  geom_histogram(col='blue',fill = 'white',binwidth = 5) + ylab('Número de meses')

##### SMOOTH DENSITY PLOT #####
ggplot(dfP,aes(CAJATAMBO,fill=mes)) +
  geom_density(alpha=0.25 ,bw=15)

##### SCATTER PLOT #####
dfP <- dfP %>% mutate(mes=reorder(mes,dates))
#dfP$mes <- reorder(dfP$mes,dfP$dates)
ggplot(dfP, aes(CHIQUIAN,CAJATAMBO)) +geom_point()

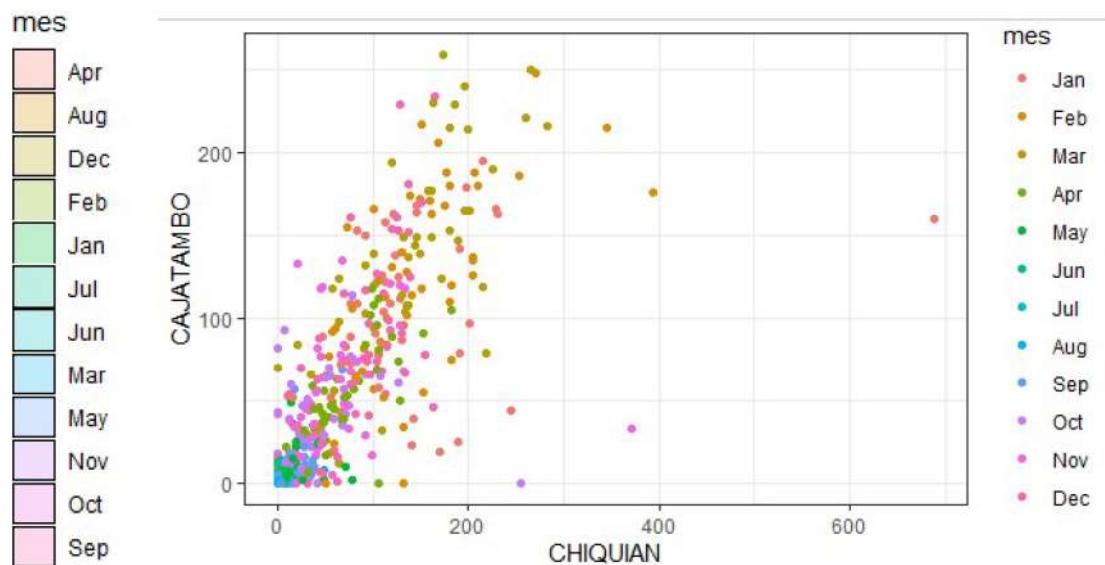
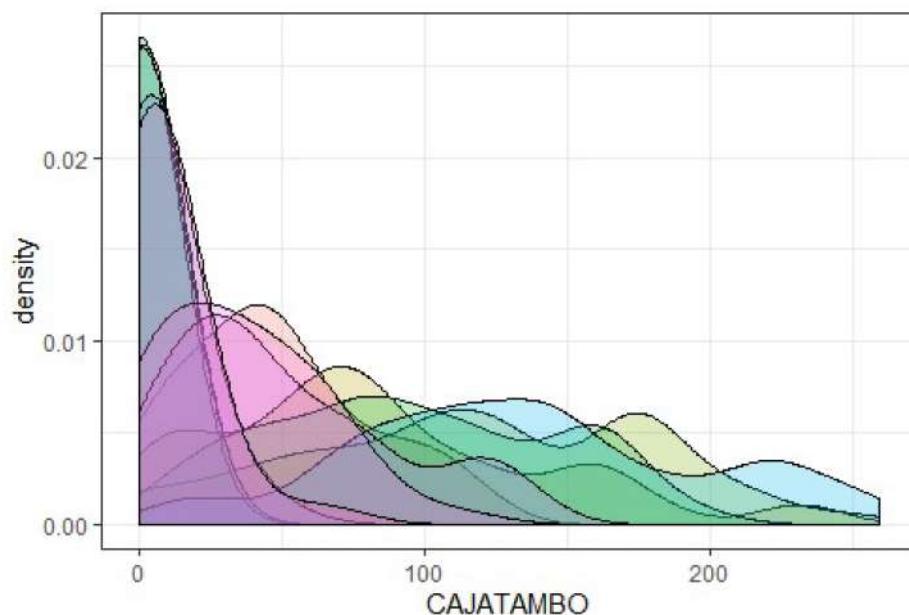
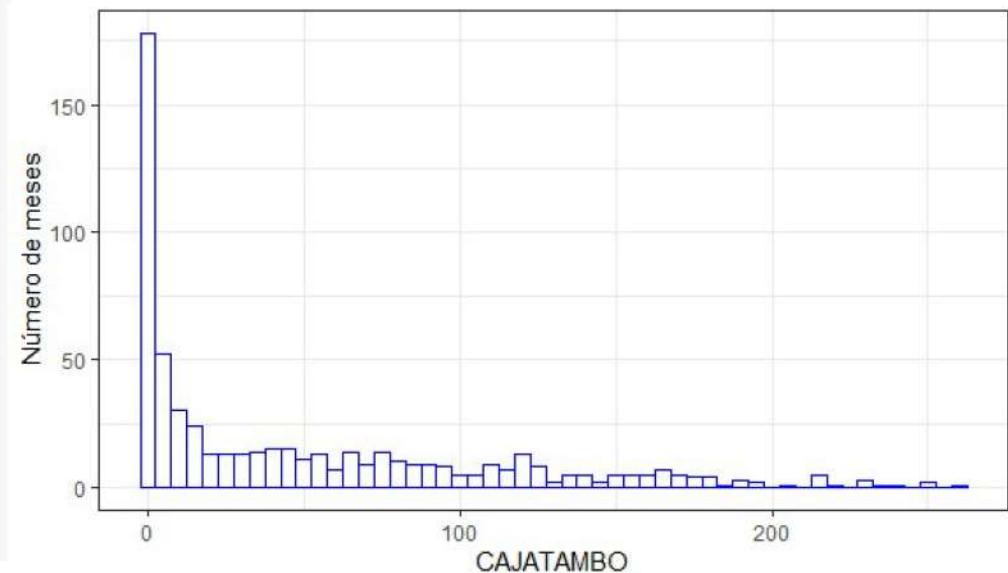
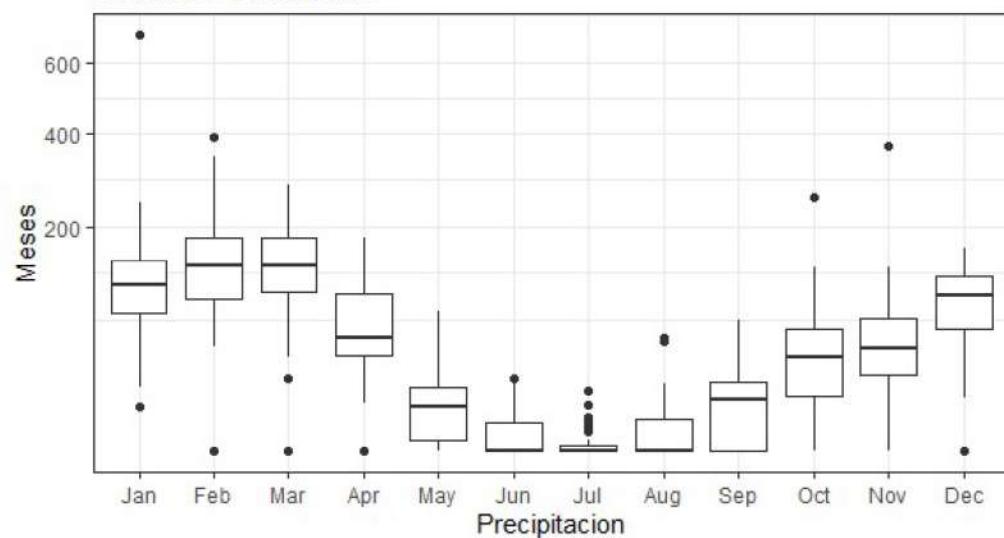
ggplot(dfP, aes(CHIQUIAN,CAJATAMBO)) +geom_point(aes(col=mes))

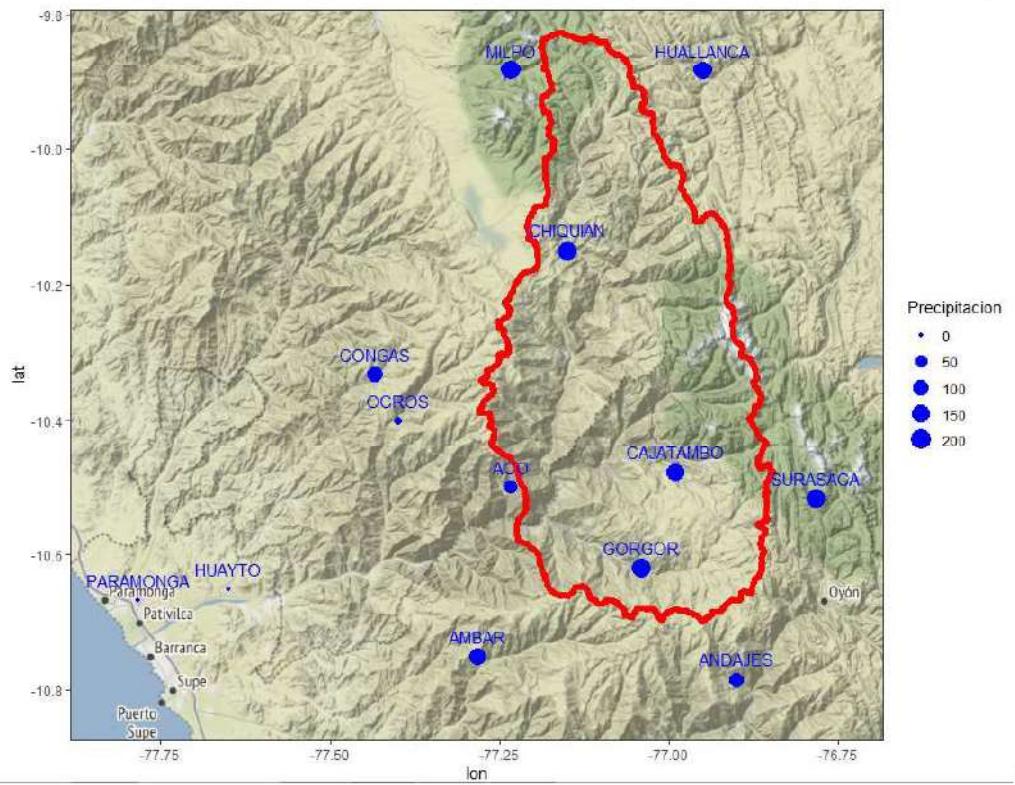
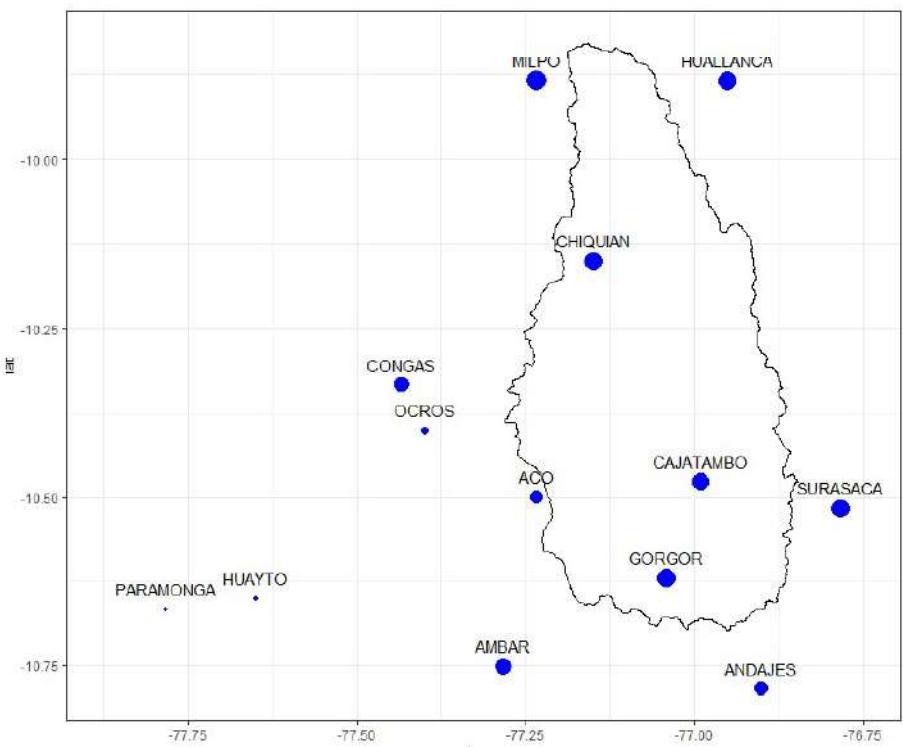
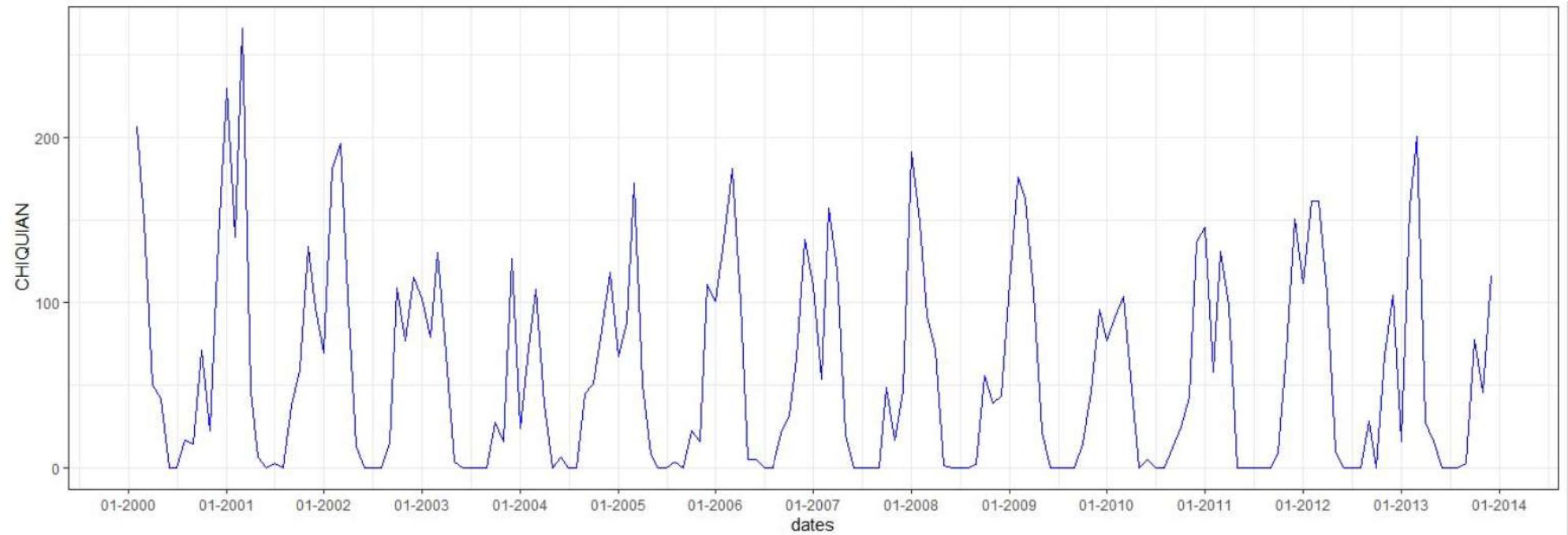
##### TIME SERIES #####
ggplot(dfP,aes(dates,CHIQUIAN)) + geom_line()

dfP %>% filter(dates>=as.POSIXct('2000-01-01')) %>%
  ggplot(aes(dates,CHIQUIAN)) + geom_line() +
  scale_x_datetime(date_breaks = "1 years",date_labels="%m-%Y")

##### Reordenando data frame #####
dfP0 <- dfP %>% gather(key = 'Estacion',value='Precipitacion',2:14)
```

Estacion CHIQUIAN





Respuestas extra

```
library(ggmap)
load('D:/staP.Rdata')
cca <- readOGR('cca_pativilca.shp')

map <- ggplot() + geom_polygon(data=cca, aes(x = long, y = lat),
                                 colour = "black",fill=NA)
staP$Precipitacion <- t(dfP[3,-c(1,15)])
map + geom_point(data=staP,aes(lon,lat,size=Precipitacion),col='blue') +
  geom_text(data=staP,aes(lon,lat,label=sta),nudge_y = 0.03) + coord_fixed() +
  xlim(c(-77.875,-76.75))

#sbbox <- make_bbox(lon = c(-77.875,-76.75), lat = c(-10.75,-9.75), f = .1)
sbbox <- make_bbox(lon = staP$lon, lat = staP$lat, f = .1)

map <- get_map(location = sbbox, maptype = "satellite", source = "google")

ggmap(map) + geom_polygon(data=cca, aes(x = long, y = lat),
                           colour = "red",size=2 ,fill=NA) +
  geom_point(data=staP,aes(lon,lat,size=Precipitacion),col='blue') +
  geom_text(data=staP,aes(lon,lat,label=sta),nudge_y = 0.03) + coord_fixed()
```

N. Practicando ggplot

Ejercicio N

En la consola de scripts, crear un código que realice lo siguiente:

- 1) Plotear un boxplot separando por estaciones
- 2) Plotear la serie de tiempo a partir del año 2000 de todas las estaciones en 1 solo gráfico.
- 3) Plotear la distribución de probabilidad de todas las estaciones en 1 solo gráfico.
- 5) Plotear un scatterplot entre las estaciones SURASACA y MILPO con datos a partir del año 2012

gganimate

Generando
giffs

Permiten generar gifs.

`transition_reveal()`

los datos aparecen
gradualmente

`transition_time()`

transición entre diferentes
estados en el tiempo

`transition_states()`

transición entre diferentes
estados de la data

O. Generando gifs

Ejercicio O

En la consola de scripts, crear un código que realice lo siguiente:

- 1) Crear una animación donde se vayan mostrando secuencialmente en el tiempo las series de tiempo de precipitación
- 2) Guardar el objeto como gif.

Respuestas O

```
library(gganimate)
```

animación de estaciones en un solo plano

```
dfPO %>% filter(dates > as.POSIXct('2010-01-01')) %>%
```

```
  ggplot(aes(dates,Precipitacion,col=Estacion)) + geom_line() + transition_reveal(dates)
```

animación de estaciones separadas

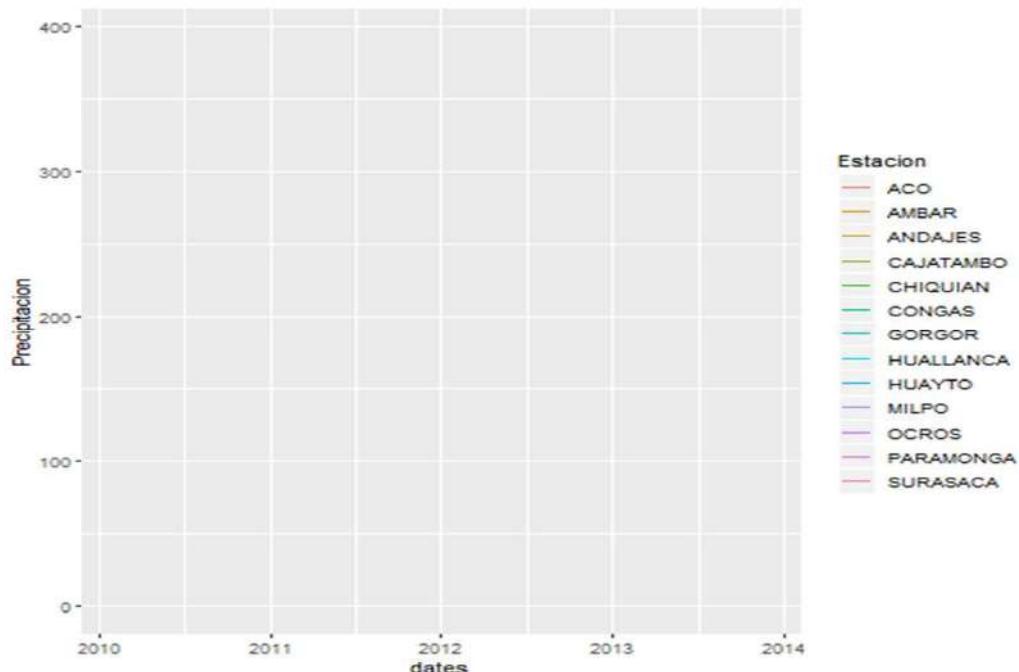
```
gif <- dfPO %>% filter(dates > as.POSIXct('2012-01-01')) %>%
```

```
  ggplot(aes(dates,Precipitacion,col=Estacion)) + geom_line() +  
  facet_wrap(~ Estacion) + transition_reveal(dates)
```

```
gif
```

guardado de animación

```
anim_save(filename = 'gif.gif',gif)
```



P. Interpolación espacial de la precipitación

- Instalar librería TOOLS - INSTALL PACKAGES: rgdal, gstat
- Revisión de archivos: countries.shp; límites de países de Europa central (Austria, Eslovenia, Hungría, Croacia, Bosnia-Herzegovina, Serbia y Montenegro)
- Archivo GSOD_2008.csv de precipitaciones del Global Surface Summary of the Day.

Ejercicio P

Interpolar la precipitación de todas las estaciones de la región de estudio para el día 01/05/2008. Emplear los métodos de Thiessen, IDW y Kriging.

Analizar el variograma estimado a partir de los datos.
Estandarizar el trabajo en el sistema de coordenadas geográficas WGS84 - 33N.

Ejercicio basado en Hengl et al (2000), d'après Ballari (2015)

http://spatial-analyst.net/book/rainfall_mapping.R

Respuestas P

```
library(rgdal)
library(gstat)
#definir directorio de trabajo
setwd("D:/2_Courses/R_Hidrologia")
# Cargar los datos de countries.shp a una variable llamada limite
# Cargar los datos de GSOD_2008.csv a una variable llamada PREC.2008
limite <- readOGR(dsn="files", layer="countries")
PREC.2008<-read.csv(file.choose(),header=TRUE, sep=";")
names(PREC.2008)
#STN: nombre de estación
#WBAN: identificador del weather bureau air force navy
#TEMPC: temperatura diaria acumulada (celcius)
#TEMP.count: número de observaciones usadas para calcular la temperatura media
#PREC: precipitación total diaria acumulada (mm)
#LAT y LON: coordenadas en latitud longitud, wgs84
#DATE: fecha de observación
str(PREC.2008)

# Formatear los datos de fechas
PREC.2008$DATE <- as.Date(PREC.2008$DATE)
# Seleccionar datos de un día en específico (1 de mayo 2008 - 01/05/2008) y cuyos
valores no sean vacíos
PREC.20080501 <- subset(PREC.2008, PREC.2008$DATE==as.Date("2008-05-
01")&!is.na(PREC.2008$PREC))
str(PREC.20080501)
```

```
head(PREC.20080501)
#Asignar coordenadas para convertir en Spatial.Points.Data.Frame
coordinates(PREC.20080501) <- ~LON+LAT # otra opción sería c(LON,LAT)
#Asignar sistema de referencia
proj4string(PREC.20080501) <- CRS("+proj=longlat +datum=WGS84")
#Definir sistema de referencia UTM y convertir
utm33 <- "+proj=utm +zone=33 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"
PREC.20080501.XY <- spTransform(PREC.20080501, CRS(utm33))
#En los datos hay una estación duplicada, de manera que se eliminará
table(duplicated(PREC.20080501.XY$STATION.NAME))
PREC.20080501.XY <- remove.duplicates(PREC.20080501.XY)
hist(PREC.20080501.XY$PREC)

# Se aplicará log para mejorar la normalización de los datos.
#log1p = log(x+1) se utiliza para evitar tener valores negativos resultado de log
hist(log1p(PREC.20080501.XY$PREC))
#Visualizar
PREC.plt1<- bubble(PREC.20080501.XY, "PREC", col="black", pch=21, main="PREC 2008-05-01", sp.layout=list(list("sp.polygons", col="grey", fill="transparent", limite), list("sp.points", col="black", pch="+", cex=1.2, subset(PREC.20080501.XY, PREC.20080501.XY$PREC==0))))
```

PREC.plt1

#Variograma empírico

```
PREC.ve.d <- variogram(log1p(PREC)~1, PREC.20080501.XY)
```

```
plot(PREC.ve.d, pl = T)
```

```
PREC.ve.d
```

variograma inicial

```
PREC.vi.d <- vgm(nugget=0, model="Exp", range=sqrt(diff(PREC.20080501.XY@bbox["LON",])^2 + diff(PREC.20080501.XY@bbox["LAT",])^2)/4, psill=var(log1p(PREC.20080501.XY$PREC)))
```

Regla general para evitar determinar parámetros visualmente

Nugget = 0

partial-sill = varianza total de los datos

Range = 1/4 de la distancia maxima, diagonal del bounding box

```
PREC.vi.d
```

#Ajuste del variograma (teórico)

```
PREC.vt.d <- fit.variogram(PREC.ve.d, model=PREC.vi.d)
```

```
PREC.vt.d
```

```
plot(PREC.ve.d, pl = T, model = PREC.vt.d)
```

Preparar grilla para predicción

```
PREC.20080501.XY_grid = spsample(PREC.20080501.XY, type = "regular", cellsize = c(1000,1000))
```

#spsample selecciona una muestra regular de puntos en la extensión geográfica de PREC.20080501.XY.

#Los puntos estarán espaciados uno de otro 1000m.

```
class(PREC.20080501.XY_grid)
```

```
gridded(PREC.20080501.XY_grid) = TRUE
```

```
class(PREC.20080501.XY_grid)
```

#Thiessen

```
thiessen.d = krige(PREC ~ 1, PREC.20080501.XY, PREC.20080501.XY_grid, nmax = 1)
pts.s <- list("sp.points", PREC.20080501.XY, col="white",pch=20)
spplot(thiessen.d, "var1.pred", asp=1, at= seq(0,16,1), zlim=c(0,16),
col.regions=gray(seq(0.9,0.1,l=30)),sp.layout = list(pts.s),main="Thiessen")
```

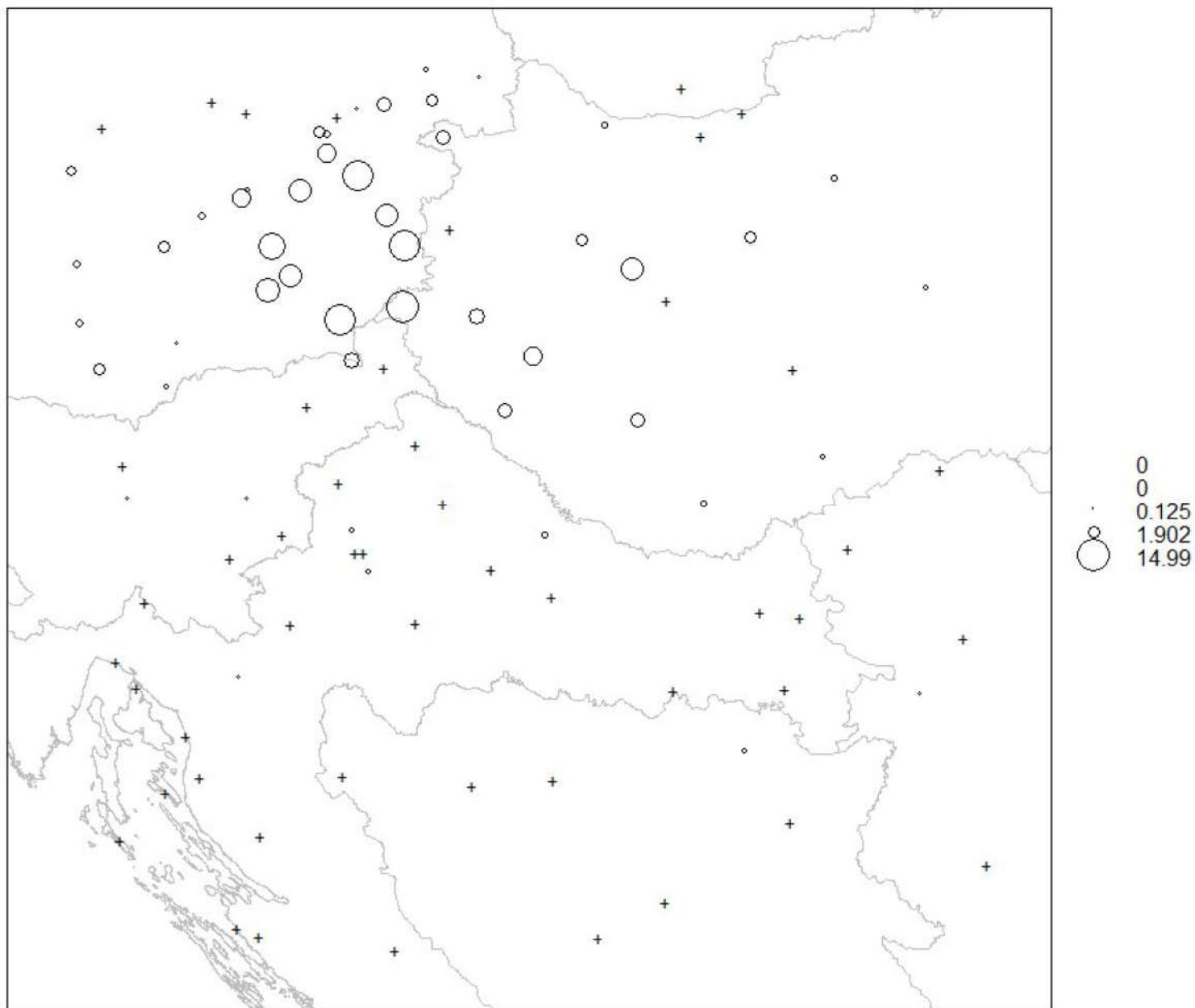
#Inverso de la distancia (idw)

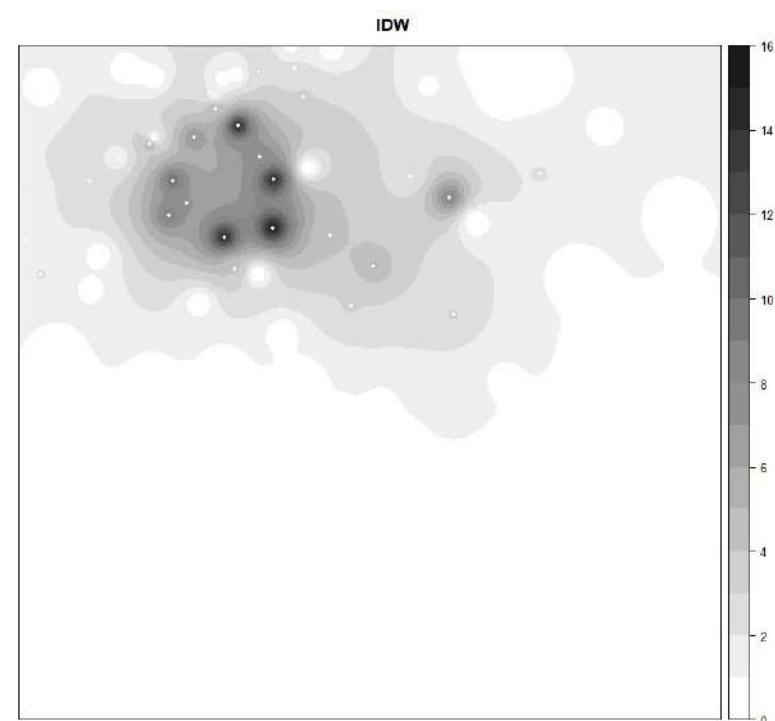
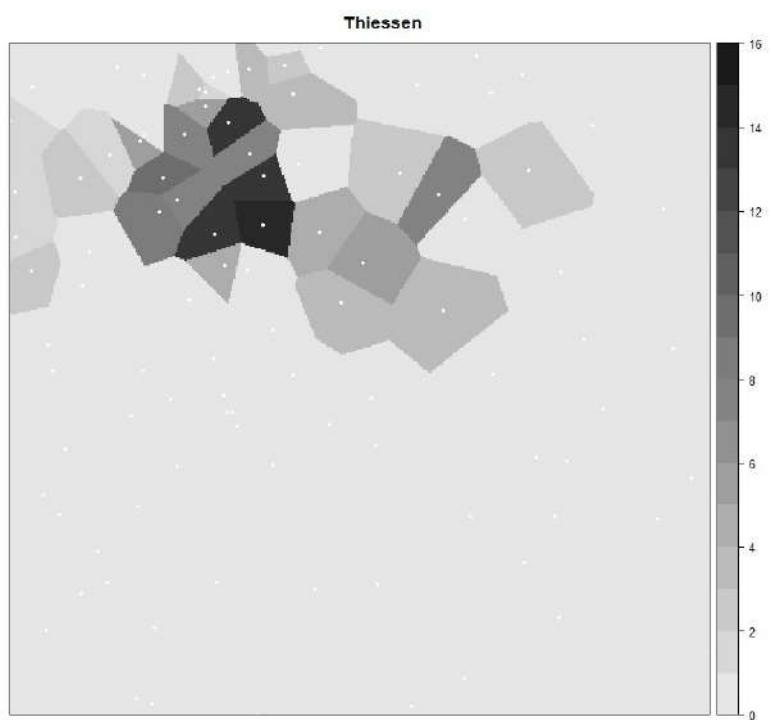
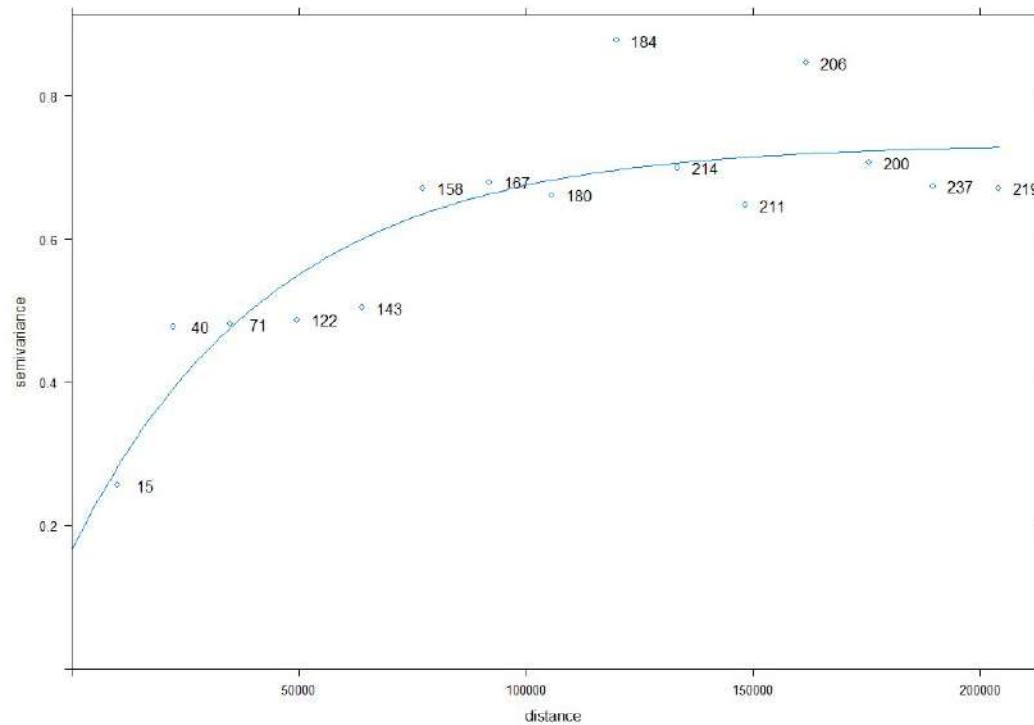
```
idw.d = idw(PREC ~ 1, PREC.20080501.XY, PREC.20080501.XY_grid)
spplot(idw.d, "var1.pred", asp=1, at= seq(0,16,1), zlim=c(0,16),
col.regions=gray(seq(1,0.1,l=30)),sp.layout = list(pts.s),main="IDW")
```

Ordinary kriging

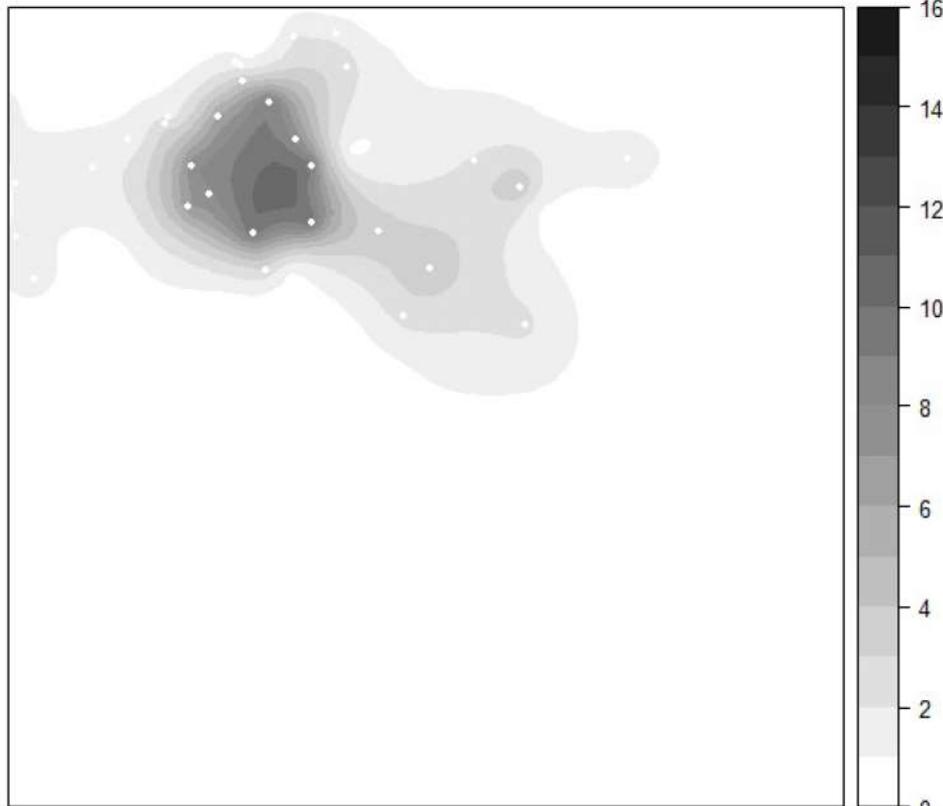
```
ok.d <- krige(log1p(PREC) ~ 1, locations = PREC.20080501.XY, newdata =
PREC.20080501.XY_grid, model = PREC.vt.d)
ok.d$PREC.pred <- expm1(ok.d$var1.pred)# Antilog para volver a valores de
precipitación
par(mfrow=c(2,1))
print(spplot(ok.d, "PREC.pred", asp=1, at= seq(0,16,1), zlim=c(0,16),
col.regions=gray(seq(1,0.1,l=30)),main="OK, diaria 01/05/2008", sp.layout = list(pts.s),
zlim=c(0,16)), split=c(1,1,2,1), more=TRUE)
print(spplot(ok.d, "PREC.pred", asp=1, col.regions=gray(seq(1,0.1,l=30)), main="Varianza
OK, diaria 01/05/2008", sp.layout = list(pts.s)), split=c(2,1,2,1), more=FALSE)
```

PREC 2008-05-01

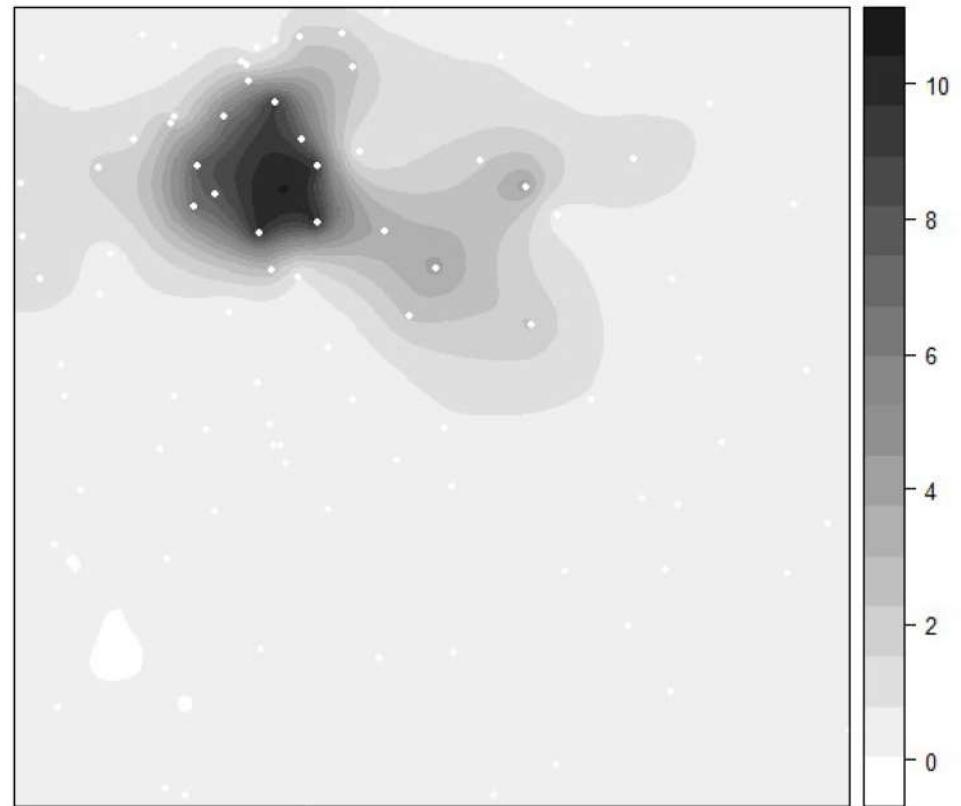




OK, diaria 01/05/2008



Varianza OK, diaria 01/05/2008



Referencias

- Ballari D. 2015. Herramientas avanzadas de investigación - Análisis espacial con R. Curso Maestria en ecohidrologia. Universidad de Cuenca. Ecuador.
- Chow V, Maidment D, Mays L. 1994. Hidrologia Aplicada. McGraw-Hill.
- Hengl T, Aghakouchak A, Pecev Tadic M. 2010. Methods and Data Sources for Spatial Prediction of Rainfall (in Rainfall: State of the Science) / Script rainfall_mapping. R http://spatial-analyst.net/book/rainfall_mapping.R
- Ihaka R. & Gentleman R. 1996. R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5: 299-314.
- Kitanidis P. 1996. Geostatistics (in Handbook of Hydrology). McGraw-Hill Education.
- Naghettini M. 2017. Fundamentals of Statistical Hydrology. Springer.
- Paradis E. 2002. R for beginners. Institut de Sciences de l'évolution. Université de Montpellier. France
- Rau P, Bourrel L, Labat D, et al. 2017. Regionalization of rainfall over the Peruvian Pacific slope and coast. *International Journal of Climatology* 37(1):143-158.
- RStudio Team (2015). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA
- Salas J. 1996. Analysis and modelling of hydrologic time series (in Handbook of Hydrology). McGraw-Hill Education.
- Shen, C. 2018. Deep learning: A next-generation big-data approach for hydrology, *Eos*, 99
- Sivapalan M, Blöschl G. 2017. The growth of hydrological understanding: Technologies, ideas, and societal needs shape the field. *Water Resources Research*, 53, 8137-8146
- Slater L, Thirel G, Harrigan S et al. 2019. Using R in hydrology: a review of recent developments and future directions. *Hydrol. Earth Syst. Sci.*, 23, 2939-2963
- Wickham H, Grolemund G. 2016. R for data science. O'Reilly Media, Inc.