

Contents

LIMNOLOGIA CON R

Taller dictado en el marco del **IV Congreso Iberoamericano de Limnología - X Congreso Argentino de Limnología (IV CIL – CAL X)**, 27 y 28 de julio de 2023 .

El material se elaboró utilizando el paquete *bookdown* (??), basado en *R Markdown* (?) y *knitr* (?).

Docentes: Natalia Morandeira, María E. Llames, Sofía Caruso, Patricia E.García, Joaquín Cochero, Julieta Capeletti y María Victoria Quiroga.

Coordinadora: María Victoria Quiroga.

0.1 Contenido

Unidad ??: Introducción a R y RStudio. Importación y manejo de datos en R. Gráficos exploratorios básicos.

Unidad ??: Correlación, regresión múltiple.

Unidad ??: Modelos Aditivos Generalizados.

Unidad ??: Análisis multivariados I. PCA, NMDS, PERMANOVA.

Unidad ??: Análisis multivariados II. DCA, RDA, dbRDA.

Unidad ??: Paquete *metrix*. Estimar métricas (individuales y ecológicas) e índices bióticos basados en macroinvertebrados acuáticos para evaluar la calidad del agua.

Unidad ??: Paquete *DiaThor*. Estimación de información ecológica e índices bióticos para diatomeas. Paquete *optimos.prime*. Estimación de óptimo y rango de tolerancia de una especie.

Chapter 1

Introducción a R y RStudio

Sofía Carusso¹

Museo Argentino de Ciencias Naturales Bernardino Rivadavia-CONICET

& María Victoria Quiroga²

Instituto Tecnológico de Chascomús (INTECH, UNSAM-CONICET), Escuela de Bio y Nanotecnologías (UNSAM)

1.1 Instalación

Seguir las indicaciones de la página <https://posit.co/download/rstudio-desktop/> para descargar e instalar **R** y **RStudio**. Es muy importante que lo haga de manera secuencial como se indica, primero **R** y luego **RStudio**.

1.2 RStudio

La primera vez que abrimos RStudio la interfaz nos muestra tres paneles:

- *Panel izquierdo -Consola:* donde corremos el código
- *Panel derecho superior -solapa Entorno:* vemos los datos y funciones que cargamos en la sesión de R.
- *Panel derecho inferior -solapa Files:* directorio de trabajo y archivos dentro de la carpeta.
- *Panel derecho inferior -solapa: Gráficos:* visualización de plots.

¹soficarusso@gmail.com

²mvquirosa@iib.unsam.edu.ar

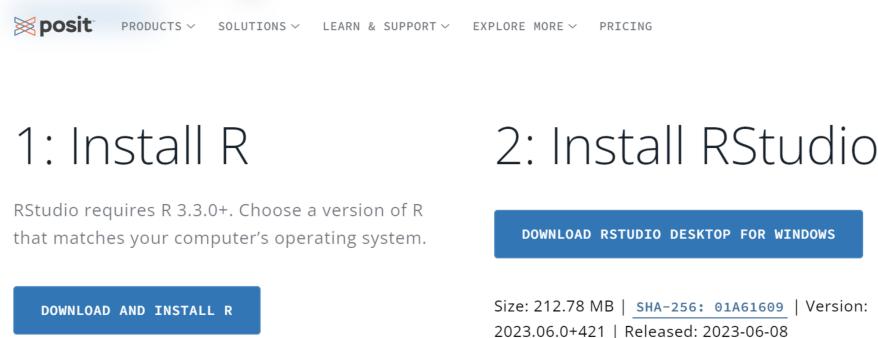


Figure 1.1: Instalación de R y RStudio.

- *Panel derecho inferior -solapa:* **Paquetes:** visualización/carga/actualización de paquetes de R.
- *Panel derecho inferior -solapa:* **Ayuda:** Recuerde que puede buscar ayuda desde esta solapa o tipeado en la consola `?nombre_del_comando`.

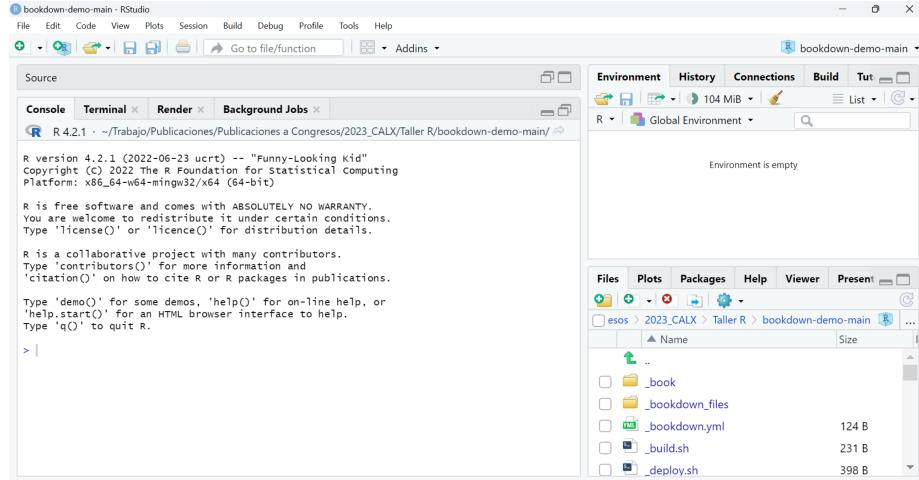


Figure 1.2: Interfaz de RStudio.

1.3 A trabajar!

1.3.1 Crear un Proyecto

1. Click en **File** (*esquina superior izquierda en Figura ??*).

2. Click en **New Project**.
3. Click en **New Directory**.
4. Click en **New Project**.
5. Escribir el nombre de la carpeta, que será el **Directorio de Trabajo** y contendrá el **Proyecto**. Se puede setear la ubicación de la carpeta haciendo click en **Browse**. En la Figura ?? se creó la carpeta LimnologiaR_U01 en el Escritorio.
6. Click en **Create Project**.

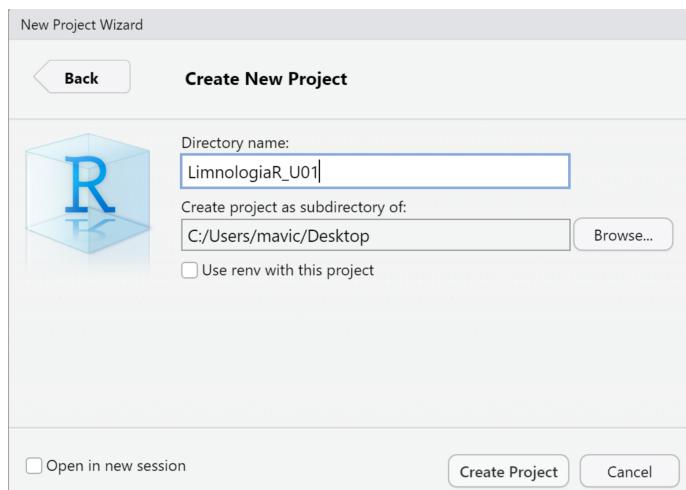


Figure 1.3: Crear un nuevo proyecto para la Unidad 1.

Ver el **Directorio de Trabajo**:

```
getwd()
```

Recomendaciones:

Crear una carpeta para cada unidad, para tener los análisis separados y ordenados.

Generar nombres de carpetas y archivos **sin** espacios, acentos o caracteres especiales.

1.3.2 Crear un R Script

1. Click en **New file**.
 2. Click en **R Script**.
- Se puede utilizar el atajo **Ctrl+Shift+N**.

Recuerde las recomendaciones para nombrar archivos al guardar el script.

El script se va a guardar como archivo **.R** en el **Directorio de Trabajo**.

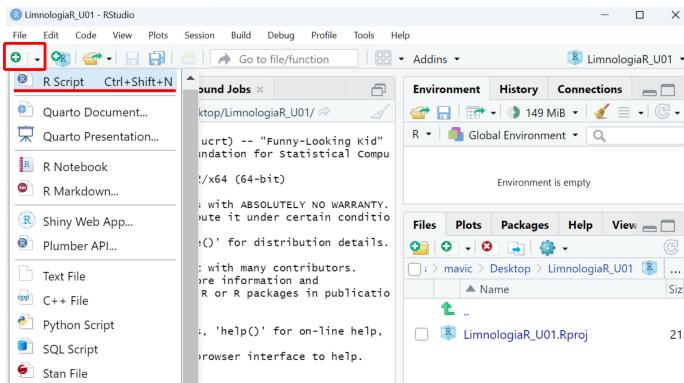


Figure 1.4: Crear un nuevo R Script.

En este **script** se escribe y guarda el código.

Una forma fácil de copiar código es utilizando el botón **Copy to clipboard** que se encuentra en la esquina superior izquierda de los **bloques de código** (Figura ??).



Figure 1.5: Copiar código desde el material del taller.

Se pega o escribe **código** en el **script** y se **guarda** haciendo click como se indica en el *recuadro 1* de la Figura ??.

El código en el **script** se **ejecuta** haciendo click en **Run the current line or selection** (*recuadro 2* en Figura ??). Podemos ejecutar de una línea por vez: nos situamos en una línea y hacemos click; o podemos ejecutar varias líneas juntas: seleccionamos el set de líneas y hacemos click. La última opción *no* se recomienda si es la primera vez que corre el código.

El código se ejecuta en la **consola** Figura ??.**2**.

Se observan los resultados en la **consola** Figura ??.**3**.

Los resultados se pueden copiar de la **consola** y pegar en el **script**. Recuerde marcar los resultados como comentarios (líneas que comienzan con #). *Atajo: seleccionar todas las líneas de resultados y presionar Ctrl+Shift+C.*

R **no** ejecuta lo que se encuentra después de #.

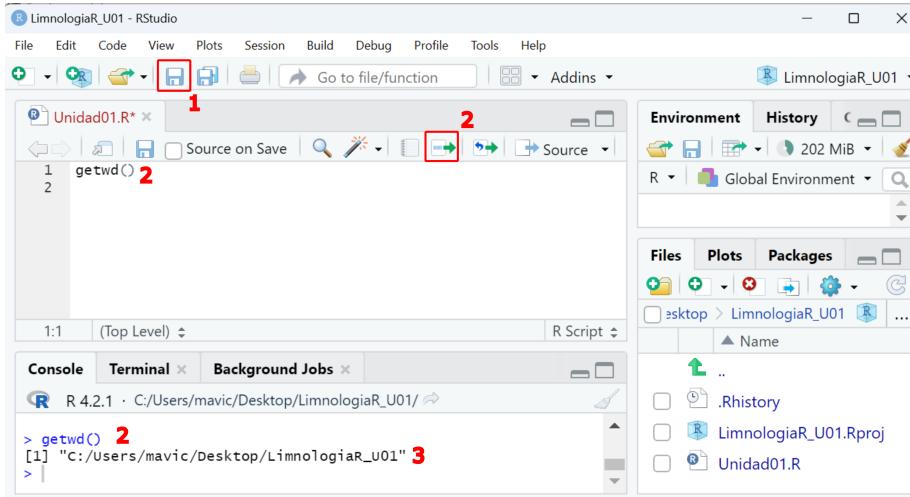


Figure 1.6: Trabajo en la interfaz de RStudio. 1- Guardar Script. 2- Ejecutar código desde Script. 3- Resultados en consola

1.3.3 Instalar y cargar paquetes

Los paquetes se instalan (por única vez) desde la **consola** con `install.packages("nombre")` o desde la interfase de la solapa **Packages**.

Los paquetes se cargan en la sesión con `library("nombre")`.

- **Buenas prácticas!** Conviene poner todas las librerías que se vayan a usar al comienzo.

1.3.4 Importar datos

Hay diversas maneras de leer datos. Podemos leer datos

- `.txt`
 - Conviene guardar la base de datos como txt (Tab delimited); donde las columnas quedan definidas por espacios.
- `.csv` Comma/Separated Values
 - Archivos de texto separados por comas. Se forma una tabla de filas y columnas

- *.xls*

Generalmente se utiliza un comando `read`, por ejemplo `read.csv("Nombre del archivo.csv")`, según el tipo de archivo que se trate (txt, xlsx, etc.)

Una vez que tenemos cargado nuestro conjunto de datos nos puede interesar realizar algunos comandos exploratorios que nos van a dar información acerca de este *data frame* (estructura de datos).

- `summary(objeto)`
 - La salida va a depender de la clase de objeto al cual lo aplicamos.
 - Útil para ver información básica sobre nuestras variables
 - Para *data frames* el summary nos devuelve el valor mínimo, máximo, la mediana y el 1er y 3er cuantil.
 - Si se aplica para salidas de modelos lineales (Ver Unidad 2), nos proporciona información acerca de los residuos, los coeficientes del modelo, el error estándar residual, R^2 , el R ajustado, el estadístico F y el p valor.
- `str(objeto)`
 - Nos devuelve cuántas observaciones tiene el data frame, información acerca de las variables (son numéricas? factores?); como esta estructurado nuestro conjunto de datos.
- `head(objeto)` y `tail(objeto)`
 - Nos da las primeras y últimas filas, respectivamente.
- `class(objeto)`
 - Devuelve qué tipo de objeto es (data frame, matrix, vector, etc).

```
data("iris")
summary(iris)
```

```
##   Sepal.Length   Sepal.Width    Petal.Length   Petal.Width
## Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
## 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
## Median :5.800   Median :3.000   Median :4.350   Median :1.300
## Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
## 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
## Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
## 
##   Species
##   setosa   :50
##   versicolor:50
```

```

##  virginica :50
##
##
##



str(iris)

## 'data.frame':   150 obs. of  5 variables:
## $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
## $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
## $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
## $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
## $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...


head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa


class(iris)

## [1] "data.frame"

```

1.3.5 Visualizar y manipular datos

Qué tipo de variables hay?

Existen variables numéricas (num), factores (que pueden tener varios niveles, factor), char o de carácter. Generalmente R detecta el tipo de variable al cargar los datos. Se puede transformar variables entre sí.

- **Observación** El comando `c()` nos permite concatenar elementos. Siempre que se selecciona más de un elemento se debe concatenar.

Por ejemplo,

```
variable <- c(1,2,1,1,1,2,2,2,1)
class(variable)

## [1] "numeric"

factor <- factor(variable, levels = c(1,2), labels=c("Nivel 1", "Nivel 2"))
```

De esta manera, transformamos una variable numérica en un factor con dos niveles, Nivel 1 y 2. Nos puede ser útil en caso de tener una variable tomada desde campo (o de encuestas) como numérica y queremos que sea categórica.

Podemos acceder a elementos particulares dentro del data frame, ya sea porque nos interesa ver ese elemento individual, para sacarlo del data frame, **para armar filtros** o para realizar ciertos análisis con una parte del data frame. Se trabaja con coordenadas (x,y), donde **x** es la fila e **y** la columna.

Si quiero todos los datos de una fila en particular, por ejemplo la 17 se escribe **iris[17,]** (seleccionamos la fila y las columnas quedan vacías porque queremos verlas todas).

De la misma manera, si queremos ver solamente una columna **iris[,3]**. También podemos llamarlo según el nombre de la misma **iris[, "Species"]**.

Para seleccionar los primeros 10 datos **iris[1:10,]** (o con el comando que ya vimos **Head()**).

Si queremos seleccionar **varias filas** simplemente las concatenamos. Entonces si escribimos **iris[c(1:5), c(2,3)]** Qué seleccionamos en este caso?

Qué pasa si quiero seleccionar una sola variable? Podemos! El signo **\$** indica que se selecciona una columna dentro del data frame que lo precede.

iris\$Species me permite ver toda la columna de la variable Especies. De esta manera, incluso podemos crear variables nuevas, asignandolas de la siguiente manera

```
iris$variable.nueva <- iris$Sepal.Length/iris$Sepal.Width
```

Qué variable nueva acabamos de crear?

- **Ojo!** Si no le asignamos el data frame a la variable nueva con el signo **\$** (**iris\$variablenueva**) el R la va a crear por fuera de nuestro data frame como un vector de los valores.

Subsets

Como lo indica el nombre, nos quedamos con una porción que seleccionemos del data frame, y sobre el cual podremos operar de manera independiente, sin que el data frame original se vea afectado.

En el ejemplo con iris, podemos crear un subset que contenga solamente a la especie iris setosa.

```
setosa <- subset(iris, iris$Species=="setosa")
```

Para aquellos que se sientan más cómodos, la librería **dplyr** permite hacer que las selecciones y filtros sea más fácil. Suma mucho a la facilidad del trabajo aprender a usarla!

```
library(dplyr)

setosa.dplyr <- iris %>%
  filter (Species %in% "setosa")
```

Hay muchas formas de hacer subsets! Es cuestión de usar aquella con la que se sientan más cómodos.

1.3.6 Gráficos exploratorios básicos

La visualización de datos es un punto clave dentro de cualquier análisis. Los gráficos son útiles para explorar datos, interpretarlos, identificar patrones, detectar outliers y constituyen una de las mejores maneras de comunicar los resultados.

¿Qué tipos de gráficos existen?

Hay una enorme diversidad, e incluso se pueden utilizar combinaciones de ellos para aumentar la información comunicada.

- **Gráfico de barras**
 - Muestra una barra para cada categoría de una variable categórica.
El alto muestra el valor observado para cada categoría (Fig 1.A)
- **Gráfico de puntos/dispersión/scatter plot**
 - Muestra la relación entre dos variables numéricas.
- **Histograma**
 - Aplica para *variables cuantitativas*. Permite ver la distribución de la variable.
- **Boxplot**

- Aplica para *variables cuantitativas*. Muestra una serie de medidas de resumen para dicha variable. Se suele graficar junto a una variable categórica para permitir la comparación entre niveles.

```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.2.2

library(ggpubr)

## Warning: package 'ggpubr' was built under R version 4.2.2

library(grid)

barra <- iris %>%
  group_by(Species) %>%
  summarise(n=n()) %>%
  ggplot(aes(Species,n, fill=Species)) +
  geom_bar(stat="identity")

scatter <- iris %>%
  ggplot(aes(Sepal.Length, Petal.Length, color=Species)) +
  geom_point()

histo <- iris %>%
  ggplot(aes(Sepal.Length, fill=Species)) +
  geom_histogram()

box <- iris %>%
  ggplot(aes(Species, Petal.Length, fill=Species)) +
  geom_boxplot()

ggarrange(barra,scatter,histo,box, ncol=2, nrow = 2, widths = c(0.5,0.5), labels=c("A"))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

¿R base o Ggplot2?

La librería en la que se suelen armar los gráficos y que nunca falta en ningún script es **ggplot2**. Sin embargo, R permite realizar gráficos nativamente sin librerías.

R base

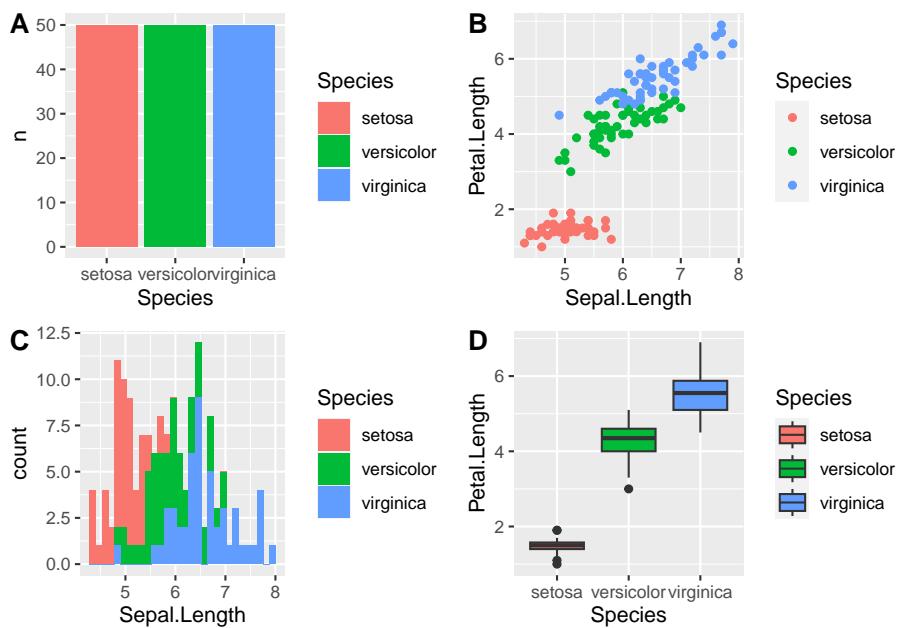
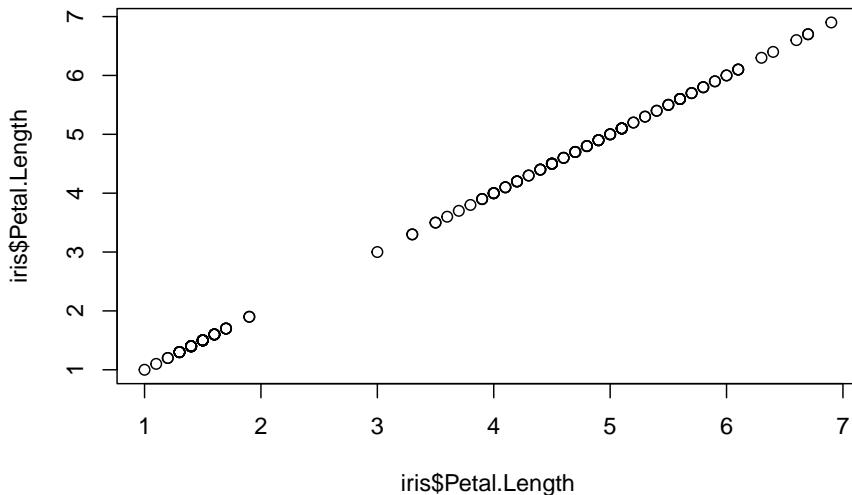


Figure 1.7: Tipos de gráficos. A) Gráfico de barras. B) Scatter plot, gráfico de puntos. C) Histograma. D) Boxplot

Las ventajas de usar R base para graficar es que es su rapidez para visualizar las relaciones entre variables, sin tener que preocuparnos por acordarnos de la sintaxis del script. Sin embargo, ggplot2 ofrece una variedad de combinaciones de customización para presentar los datos difícil de equiparar.

El comando básico es `plot()` entre dos variables. Es decir,

```
plot(iris$Petal.Length,iris$Petal.Length)
```



De ahí en adelante, se pueden agregar distintos parámetros que hagan a la customización. Por ejemplo,

```
plot(iris$Petal.Length,iris$Petal.Width,
     cex=2, # Tamaño de la forma
     pch=16, # Forma (puede ser circulo, triangulo, etc)
     xlab="Petal Length", # Titulo del eje x
     ylab="Petal Width", # Titulo del eje y
     main="Petal Width vs Petal Length of Iris", # Titulo del grafico
     col=iris$Species) # Si quiero agregarle color a los puntos. En ese caso, lo hace .
```

```
legend(x=1, y=2.4, legend=levels(iris$Species), col=c(1:3), pch=16) # Leyenda
```

Veamos como se hace el resto de los gráficos:

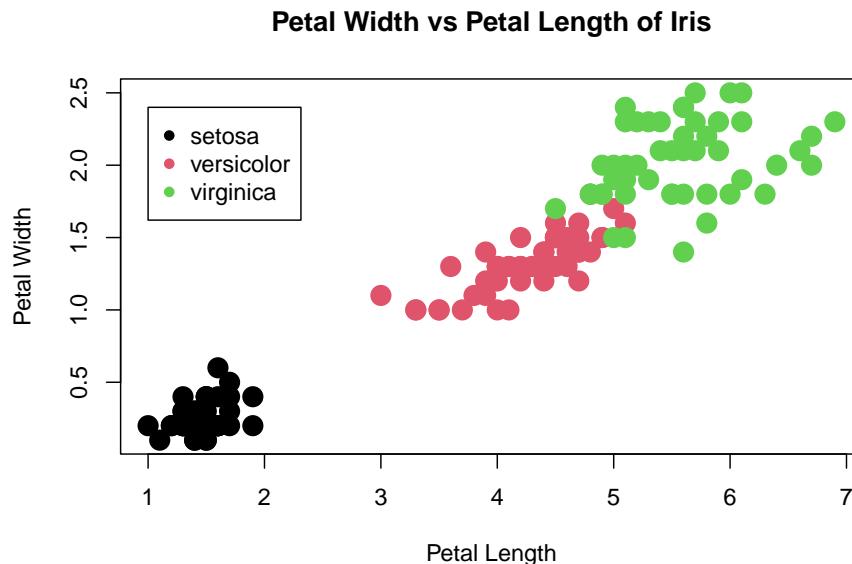


Figure 1.8: Tipos de gráficos

```
# Barra
plot(iris$Species)
```

```
# Boxplot
plot(iris$Species, iris$Sepal.Length,
     xlab="Species",
     ylab="Sepal Length",
     main="Sepal Length by Iris species",
     col="skyblue")
```

```
# Histogram
hist(iris$Sepal.Width,
     col="yellow",
     xlab="Sepal Width",
     main="Histogram of Sepal Width",
     breaks=30)
```

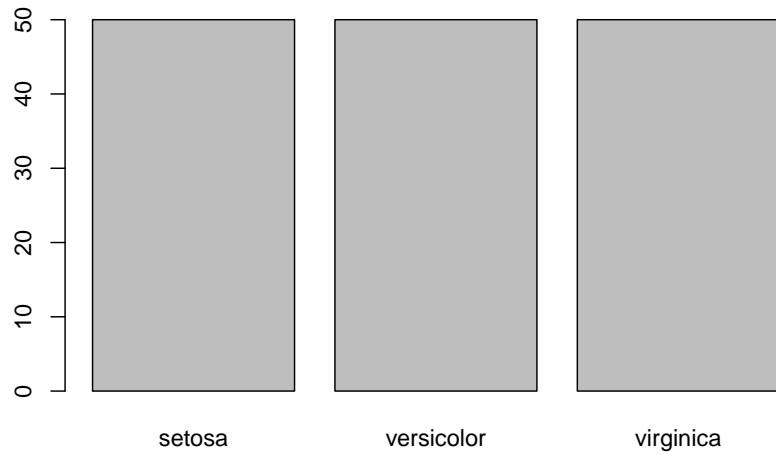


Figure 1.9: Tipos de gráficos

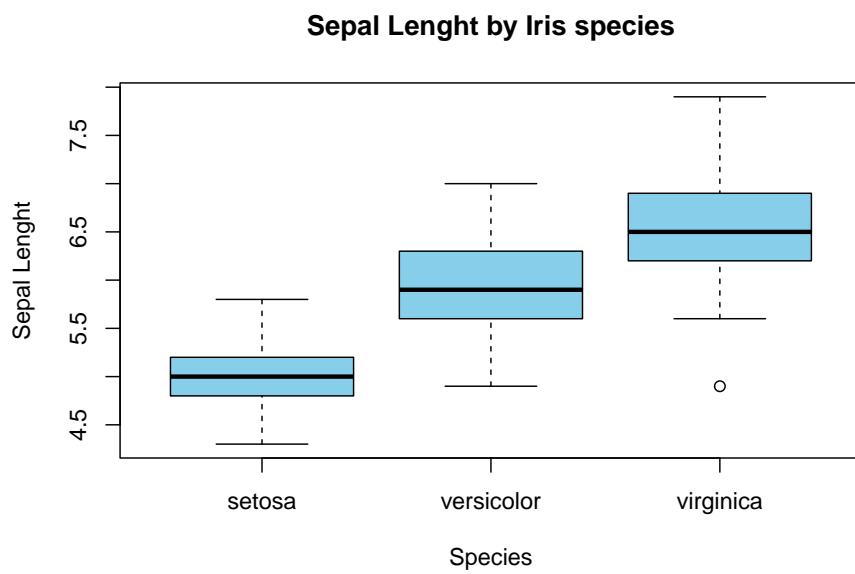


Figure 1.10: Tipos de gráficos

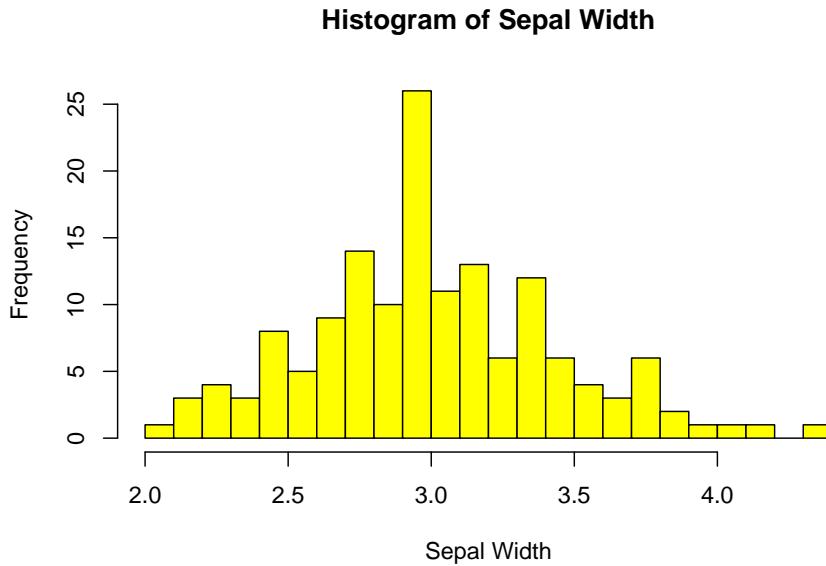


Figure 1.11: Tipos de gráficos

```
# Qué pasa si queremos plotear el data frame entero?
plot(iris)
```

```
ggplot2
```

Este paquete tiene una manera de escribirse particular en capas. Los gráficos de ggplot2 está compuesto por los datos, por un conjunto de características estéticas a tener en cuenta entre los datos junto con los aspectos visuales (*aes*, *aesthetic mapping*) y por al menos una capa que indica cómo se debe manipular cada observación (*geom_*).

```
library(ggplot2)

ggplot(iris, aes(x=Petal.Length, y=Petal.Width))+
  geom_point()
```

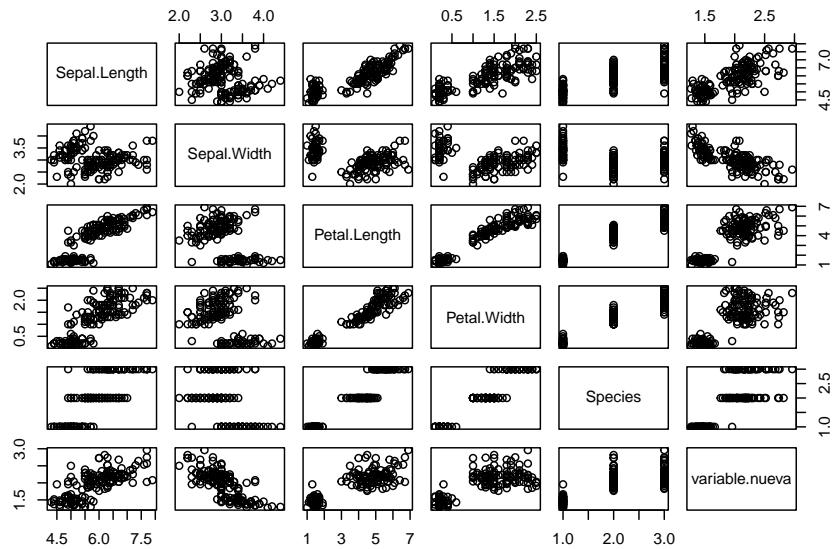
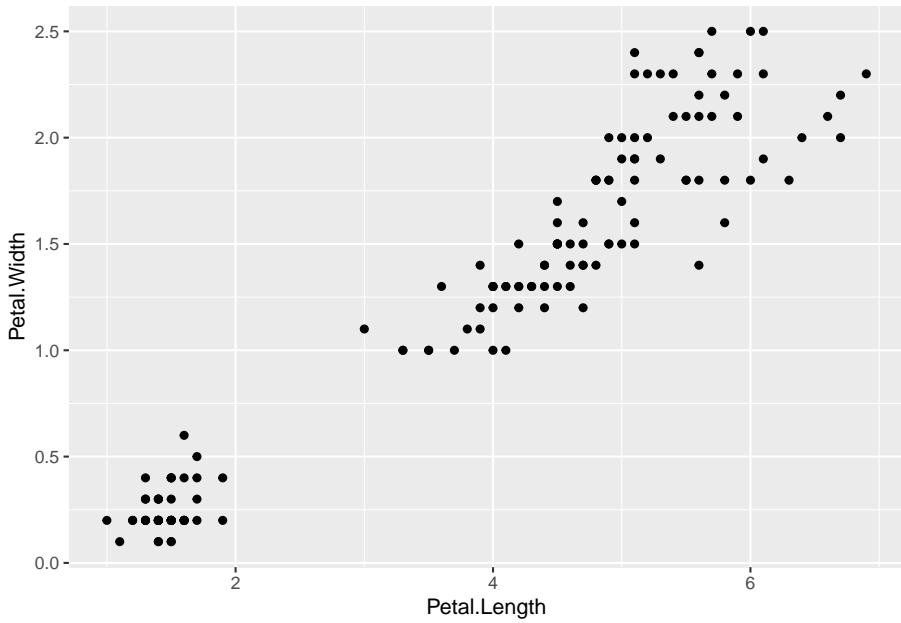


Figure 1.12: Tipos de gráficos

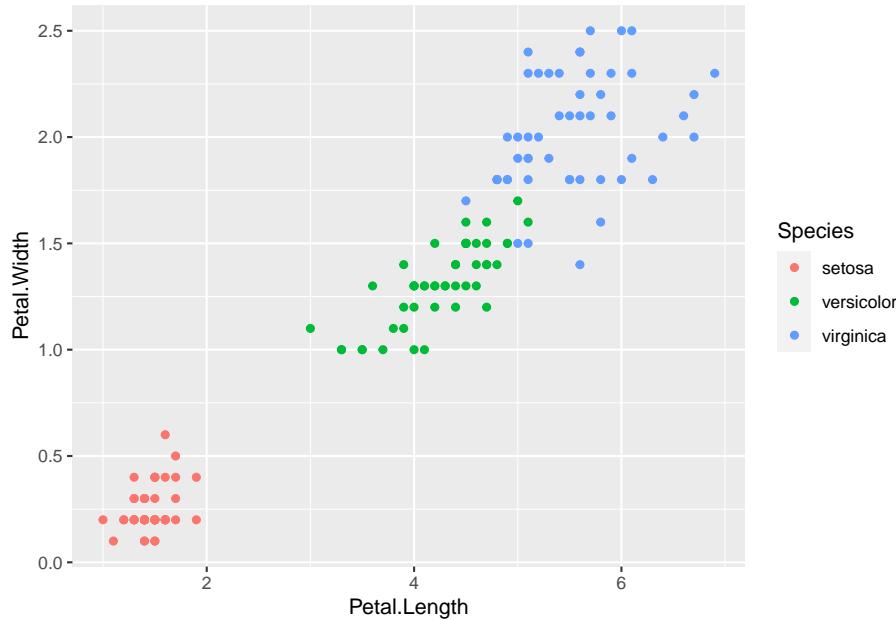


- **Buenas prácticas!** Conviene usar un renglón por capa, de esta manera

es fácil detectar errores y ver cómo se va modificando el gráfico a medida que van agregando capas.

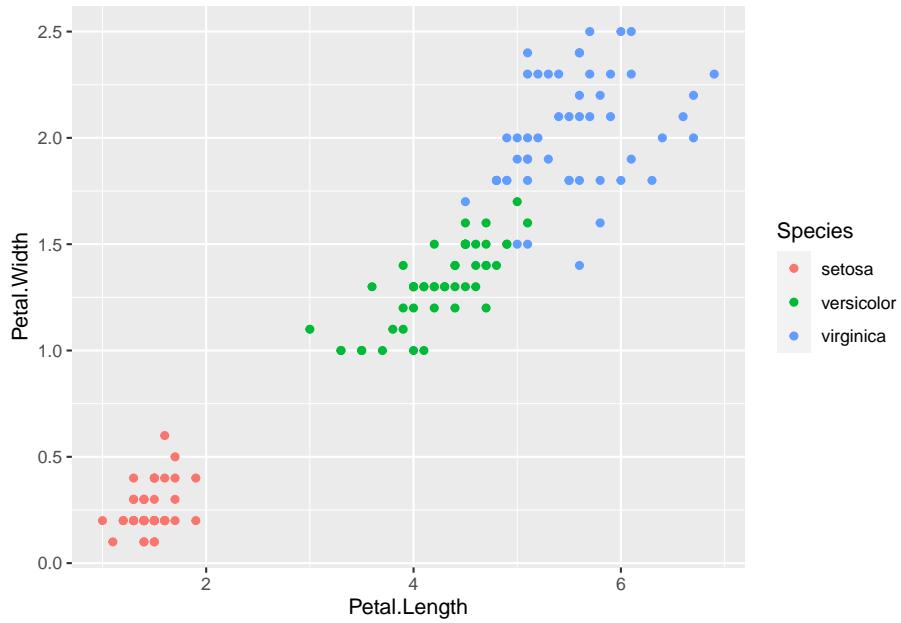
Al igual que en el plot del R base, se pueden cambiar los colores y formas. Sin embargo, esto va a estar atado a qué elemento queremos cambiar.

```
ggplot(iris, aes(x=Petal.Length, y=Petal.Width))+
  geom_point(aes(colour=Species))
```



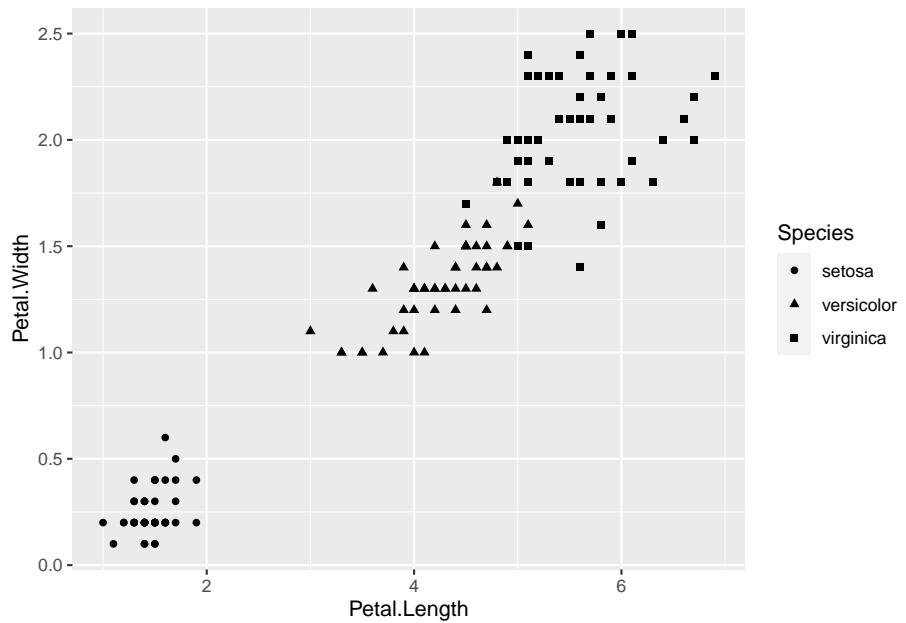
En este caso, quise cambiar el color de los puntos. Para ello, lo tengo que especificar en la capa *aes* del `geom_point`.

```
ggplot(iris, aes(x=Petal.Length, y=Petal.Width, color=Species))+  
  geom_point()
```

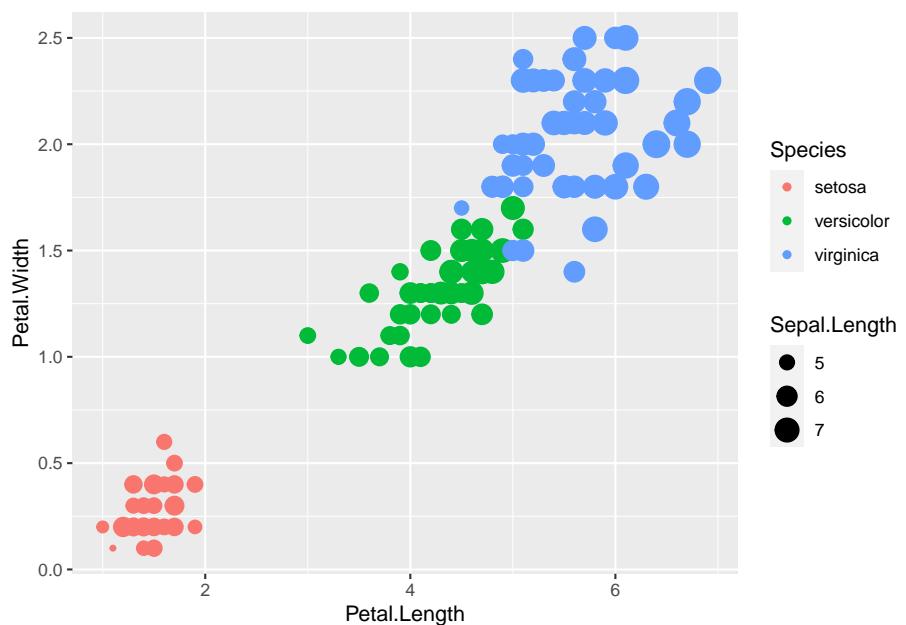


También puede ir en el aes general del gráfico. No solo puedo cambiar el color, sino también formas. O combinar todo:

```
ggplot(iris, aes(x=Petal.Length, y=Petal.Width, shape=Species))+  
  geom_point()
```



```
ggplot(iris, aes(x=Petal.Length, y=Petal.Width, size=Sepal.Length))+  
  geom_point(aes(color=Species))
```



Noten que se generan automáticamente las leyendas, según vamos cambiando

lo estético, a diferencia de R base.

Este es un primer pantallazo el mundo de ggplot. Hay mucho para aprender y en general, todo lo que quieran graficar van a poder, y de muchísimas maneras. Google es su gran amigo!

1.3.7 Recursos extra recomendados

Uso de la librería **ggpubr** para alinear plots <http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/81-ggplot2-easy-way-to-mix-multiple-graphs-on-the-same-page/>

Healy K. 2019. Data visualization: A practical Introduction **Versión libre online:**

<https://socviz.co/index.html#preface>

R Charts <https://r-charts.com/es/ggplot2/>

Cheatsheet en Code <http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3>

Chapter 2

Correlaciones y regresión múltiple

Autora de esta unidad: Natalia Morandeira

En esta Unidad vamos a abordar análisis de correlación y regresiones múltiples (modelos lineales). Las correlaciones son análisis útiles para explorar nuestros datos y para evaluar si dos variables están asociadas positivamente o negativamente. Por otro lado, en el caso de una regresión simple, tenemos una variable respuesta (o dependiente) y una variable explicativa (o independiente). Un análisis de regresión múltiple es muy adecuado cuando hemos medido muchas variables, por ejemplo, si tenemos una variable respuesta y muchas posibles variables explicativas. Nos interesa conocer cuál o cuáles variables explican la variación de la variable respuesta, y elegir el mejor modelo.

En el marco de análisis limnológicos, una situación común es tener una variable respuesta medida en cuerpos de agua y múltiples variables que creemos que pueden explicar su variación. Entre las variables respuestas, podríamos tener (de acuerdo al objetivo de nuestro estudio) la abundancia, biomasa o diversidad de especies de un dado taxón o de grupos funcionales; o la concentración de un agroquímico; o incluso variables físico-químicas del cuerpo de agua que hipotetizamos que dependen de otro factor. Entre las variables respuestas, es posible que tengamos variables físico-químicas, características morfométricas de la laguna, usos de suelo del entorno, características de la cuenca, abundancia de otro taxón que interactúe con nuestra especie de interés, etc. Nos puede interesar en primer lugar analizar si están correlacionadas nuestras potenciales variables explicativas. Luego, podemos intentar ajustar un buen modelo que explique cómo varía nuestra variable respuesta en función de varias variables explicativas. Ese es el camino que recorreremos en esta Unidad.

2.1 Caso de estudio

El sitio de estudio es la turbera de Rancho Hambre (Tierra del Fuego, Argentina), en donde entre octubre de 2009 y abril de 2010 se realizaron muestreos en lagunas (ver artículo de Quiroga *et al.* 2017, y la tesis doctoral de María Victoria Quiroga (FCEN-UBA)). En cinco lagunas (RH1 a RH5), se realizaron muestreos estacionales (octubre, diciembre, febrero y abril), es decir, tenemos un total de 20 observaciones. Contamos con dos tablas de datos:

- **Datos limnológicos físicoquímicos.** Los datos corresponden a los sitios de muestreo de orilla de los cuerpos de agua. Los parámetros limnológicos fueron estimados según lo descripto en la sección *Sampling regime, physical-chemical analyses* de Materiales y Métodos en Quiroga *et al.* (2017). Los datos físicoquímicos están disponibles en el repositorio digital de CONICET. Consideraremos a las variables medidas como variables explicativas: pH, conductividad, oxígeno disuelto, dureza total, DOC, DIN, nitrógeno total, DRP, fósforo total, ag(440), SUVA254 e índice MS; todas ellas variables numéricas. Por otro lado tenemos variables categóricas que discutiremos cómo considerar: la laguna (cinco niveles) y la fecha de muestreo (cuatro niveles).
- **Biomasa de morfotipos bacterianos.** La biomasa de bacterias heterótrofas se estimó utilizando microscopía de epifluorescencia y análisis de imágenes. Para detalles ver sección *Bacterioplankton morphotypes* de Materiales y Métodos en Quiroga *et al.* (2017). Los datos de morfotipos bacterianos están disponibles en el repositorio digital de CONICET. En la tabla se cuenta con la biomasa de seis morfotipos bacterianos distintos. Podemos tomar a la biomasa de cada morfotipo como una variable respuesta por separado, o bien considerar para un primer análisis a la biomasa total de bacterias heterótrofas como la variable a modelar.

2.2 Leer y emprolijar los datos

Siempre debemos mirar nuestra tabla de datos antes de empezar: detectar posibles errores o datos faltantes, retocar los nombres de las variables, agrupar filas y columnas si corresponde, etc. En nuestro caso además tenemos por un lado las variables físico-químicas y por el otro lado las variables biológicas, pero para los análisis es conveniente tener todos los datos en un mismo *dataframe*.

Empezamos leyendo las bases de datos en R. Es una buena práctica guardar los archivos en una carpeta llamada *data*, dentro de la carpeta de nuestro proyecto. Tenemos los datos en formato *csv* en GitHub Limno-con-R/CILCAL2023 **RH_abiotic_data.csv** y **RH_morpho_biomass.csv** (son casi iguales a los disponibles en el repositorio CONICET, se cambió un guión bajo por un

espacio en una de las tablas). Cargamos los datos indicando que se saltee (*skip*) la primera fila (¿por qué?).

```
library(readr)

datos_fq <- read_csv("data/RH_abiotic_data.csv", skip = 1)

datos_bacterias <- read_csv("data/RH_morpho_biomass.csv", skip = 1)
```

Ahora vamos a ver los datos. Recordar o anotar cuántas filas y columnas tiene cada *dataframe*.

```
datos_fq
```

```
## # A tibble: 20 x 16
##   ID    Pool Site     Date      pH Condu~1 Disso~2 Total~3   DOC   DIN Total~4
##   <chr> <chr> <chr>   <chr> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 10   RH1   shore   Octo~  4.91   13.4    NA     21.2   5.6   83.6  1870
## 2 20   RH2   shore   Octo~  5.52   8.7     NA     46.2   5.1   41.3  1980
## 3 30   RH3   shore   Octo~  4.72   15.2    NA     13.8   2.8   10    1980
## 4 40   RH4   north sh~ Octo~  5.09   11.8    NA     33.2   5.7   55.3  1650
## 5 50   RH5   shore   Octo~  4.82   5.5     NA     25.8   3.9   54.4  3410
## 6 1D   RH1   shore   Dece~  6.39   19.1    11.0   38.5   7.2   21.1  1320
## 7 2D   RH2   shore   Dece~  4.75   22.6    11.0   42.1   7.7   12.1  1870
## 8 3D   RH3   shore   Dece~  4.67   26.7    11.7   18.9   13.4  22.6  5720
## 9 4D   RH4   north sh~ Dece~  6.75   25.6    11.7   24.8   5.3   34    7480
## 10 5D  RH5   shore   Dece~  4.65   23     11.5   20.1   11    11.2  10230
## 11 1F  RH1   shore   Febr~  5.9    21.4    10.4   39.5   12.7  53    8030
## 12 2F  RH2   shore   Febr~  5.19   22.6    10.2   34     5.9   30.3  10010
## 13 3F  RH3   shore   Febr~  4.66   25.7    10.6   32     12.5  45.3  11330
## 14 4F  RH4   north sh~ Febr~  6.65   26.8    10.9   36.6   4     0    5610
## 15 5F  RH5   shore   Febr~  4.82   27.2    10.4   24.5   7.1   22.3  8250
## 16 1A  RH1   shore   Apri~  7.1    21.4    10.7   25.3   7.6   23.1  9790
## 17 2A  RH2   shore   Apri~  4.88   24     11.4   20.3   7.5   0     11110
## 18 3A  RH3   shore   Apri~  5.4    28.5    10.4   18.8   12.7  103.  8910
## 19 4A  RH4   north sh~ Apri~  6.2    28.5    11.2   27.7   5.3   43    3630
## 20 5A  RH5   shore   Apri~  5.43   27.1    11.2   24.2   11.6  73.2  3740
## # ... with 5 more variables: DRP <dbl>, `Total phosphorus` <dbl>,
## #   `ag(440)` <dbl>, SUVA254 <dbl>, `MS index` <dbl>, and abbreviated variable
## #   names 1: Conductivity, 2: `Dissolved oxygen`, 3: `Total hardness`,
## #   4: `Total nitrogen`
```

```
datos_bacterias
```

```
## # A tibble: 20 x 10
```

```

##   ID   Pool Site      Date Filam~1 Large~2 Vibri~3 Large~4 Small~5 Small~6
##   <chr> <chr> <chr>      <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 10  RH1 shore Octo~       0 16921  5315 12676 1747 25994
## 2 20  RH2 shore Octo~     1435  5645 2384  7381  888 12075
## 3 30  RH3 shore Octo~    13307 1597  392  8131  708 13222
## 4 40  RH4 north shore Octo~       0 8085  3499  7534 2028 25353
## 5 50  RH5 shore Octo~       0 1921  451  15662  666 31621
## 6 1D  RH1 shore Dece~      0 12374  9228 21906 4503 38421
## 7 2D  RH2 shore Dece~      0 22924  5933 34663 6687 77581
## 8 3D  RH3 shore Dece~      0 12309  3471 46807 8694 139006
## 9 4D  RH4 north shore Dece~  44419 76819 19692 44391 24931 156132
## 10 5D RH5 shore Dece~      0 66952 23046 134707 2949 102890
## 11 1F RH1 shore Febr~     3177  6414 7533 22218 4013 34495
## 12 2F RH2 shore Febr~      0 7263  4879 25886 4098 137049
## 13 3F RH3 shore Febr~     3886 9283 1294 17402 2522 93008
## 14 4F RH4 north shore Febr~  5860 54624 20331 33949 12469 58990
## 15 5F RH5 shore Febr~      0 21324 17482 24397 7064 106885
## 16 1A RH1 shore Apri~    12331 21098 11615 16083 6592 33478
## 17 2A RH2 shore Apri~      0 6950  4028 22525 1395 94037
## 18 3A RH3 shore Apri~      0 12394 17281 18608 4678 121090
## 19 4A RH4 north shore Apri~  0 901  2484 3455  642 50957
## 20 5A RH5 shore Apri~      0 7626  3338 17231 2016 60287
## # ... with abbreviated variable names 1: Filaments, 2: Large_rods,
## #     3: Vibrio_shaped, 4: Large_cocci, 5: Small_rods, 6: Small_cocci

```

Para unir estas tablas en una sola, debemos buscar una variable o columna que sea un indicador único de cada dato, y que sea equivalente en ambas tablas. ¿Cuál les parece que es?

Con ese indicador como nexo, haremos una **unión exterior completa** o *full_join*. Esto se debe a que queremos mantener todos los registros de nuestras tablas aunque para alguna laguna –quizás– no hayamos podido medir los parámetros físicoquímicos o no hayamos podido medir la biomasa bacteriana. Para saber más sobre funciones que permiten relacionar conjuntos de datos, recomendamos consultar el capítulo 13. Datos relacionales (en particular *13.4.1. Entendiendo las uniones*) en el libro traducido a castellano R para Ciencias de Datos, de Hadley Wickham y Garret Grolemund (2017).

Hacemos la unión e inspeccionamos la tabla (¿cuántas filas y columnas tiene?).

```

library(tidyverse)

datos_RH <- full_join(datos_bacterias, datos_fq, by = "ID")
datos_RH

## # A tibble: 20 x 25

```

```

##   ID   Pool.x Site.x    Date.x Filam~1 Large~2 Vibri~3 Large~4 Small~5 Small~6
##   <chr> <chr> <chr>    <chr>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 10  RH1    shore Octob~       0  16921    5315  12676    1747  25994    4.91
## 2 20  RH2    shore Octob~     1435  5645  2384  7381    888  12075    5.52
## 3 30  RH3    shore Octob~    13307  1597  392   8131    708  13222    4.72
## 4 40  RH4    north sh~ Octob~       0  8085  3499  7534  2028  25353
## 5 50  RH5    shore Octob~       0  1921    451   15662   666  31621
## 6 1D  RH1    shore Decem~      0  12374  9228  21906  4503  38421
## 7 2D  RH2    shore Decem~      0  22924  5933  34663  6687  77581
## 8 3D  RH3    shore Decem~      0  12309  3471  46807  8694  139006
## 9 4D  RH4    north sh~ Decem~    44419  76819  19692  44391  24931  156132
## 10 5D  RH5    shore Decem~      0  66952  23046  134707  2949  102890
## 11 1F  RH1    shore Febru~     3177  6414  7533  22218  4013  34495
## 12 2F  RH2    shore Febru~      0  7263  4879  25886  4098  137049
## 13 3F  RH3    shore Febru~     3886  9283  1294  17402  2522  93008
## 14 4F  RH4    north sh~ Febru~    5860  54624  20331  33949  12469  58990
## 15 5F  RH5    shore Febru~      0  21324  17482  24397  7064  106885
## 16 1A  RH1    shore April~    12331  21098  11615  16083  6592  33478
## 17 2A  RH2    shore April~      0  6950  4028  22525  1395  94037
## 18 3A  RH3    shore April~      0  12394  17281  18608  4678  121090
## 19 4A  RH4    north sh~ April~      0  901   2484  3455   642  50957
## 20 5A  RH5    shore April~      0  7626  3338  17231  2016  60287
## # ... with 15 more variables: Pool.y <chr>, Site.y <chr>, Date.y <chr>,
## # pH <dbl>, Conductivity <dbl>, `Dissolved oxygen` <dbl>,
## # `Total hardness` <dbl>, DOC <dbl>, DIN <dbl>, `Total nitrogen` <dbl>,
## # DRP <dbl>, `Total phosphorus` <dbl>, `ag(440)` <dbl>, SUVA254 <dbl>,
## # `MS index` <dbl>, and abbreviated variable names 1: Filaments,
## # 2: Large_rods, 3: Vibrio_shaped, 4: Large_cocci, 5: Small_rods,
## # 6: Small_cocci

```

Las variables Pool, Site y Date se repiten en ambas tablas, por eso aparecen como Pool.x y Pool.y (por ejemplo). Dado que son idénticas, tenemos la opción de eliminar estas variables de alguna de las dos tablas, o bien incluirlas como parte de la información de nexo.

```

datos_RH <- full_join(datos_bacterias, datos_fq, by = c("ID", "Pool", "Site", "Date"))
datos_RH

## # A tibble: 20 x 22
##   ID   Pool  Site  Date  Filam~1 Large~2 Vibri~3 Large~4 Small~5 Small~6    pH
##   <chr> <chr> <chr> <chr>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 10  RH1    shore Octo~       0  16921    5315  12676    1747  25994  4.91
## 2 20  RH2    shore Octo~     1435  5645  2384  7381    888  12075  5.52
## 3 30  RH3    shore Octo~    13307  1597  392   8131    708  13222  4.72
## 4 40  RH4    nort~ Octo~       0  8085  3499  7534  2028  25353  5.09

```

```

## 5 50 RH5 shore Octo~ 0 1921 451 15662 666 31621 4.82
## 6 1D RH1 shore Dece~ 0 12374 9228 21906 4503 38421 6.39
## 7 2D RH2 shore Dece~ 0 22924 5933 34663 6687 77581 4.75
## 8 3D RH3 shore Dece~ 0 12309 3471 46807 8694 139006 4.67
## 9 4D RH4 nort~ Dece~ 44419 76819 19692 44391 24931 156132 6.75
## 10 5D RH5 shore Dece~ 0 66952 23046 134707 2949 102890 4.65
## 11 1F RH1 shore Febr~ 3177 6414 7533 22218 4013 34495 5.9
## 12 2F RH2 shore Febr~ 0 7263 4879 25886 4098 137049 5.19
## 13 3F RH3 shore Febr~ 3886 9283 1294 17402 2522 93008 4.66
## 14 4F RH4 nort~ Febr~ 5860 54624 20331 33949 12469 58990 6.65
## 15 5F RH5 shore Febr~ 0 21324 17482 24397 7064 106885 4.82
## 16 1A RH1 shore Apr~ 12331 21098 11615 16083 6592 33478 7.1
## 17 2A RH2 shore Apr~ 0 6950 4028 22525 1395 94037 4.88
## 18 3A RH3 shore Apr~ 0 12394 17281 18608 4678 121090 5.4
## 19 4A RH4 nort~ Apr~ 0 901 2484 3455 642 50957 6.2
## 20 5A RH5 shore Apr~ 0 7626 3338 17231 2016 60287 5.43
## # ... with 11 more variables: Conductivity <dbl>, `Dissolved oxygen` <dbl>,
## # `Total hardness` <dbl>, DOC <dbl>, DIN <dbl>, `Total nitrogen` <dbl>,
## # DRP <dbl>, `Total phosphorus` <dbl>, `ag(440)` <dbl>, SUVA254 <dbl>,
## # `MS index` <dbl>, and abbreviated variable names 1: Filaments,
## # 2: Large_rods, 3: Vibrio_shaped, 4: Large_cocci, 5: Small_rods,
## # 6: Small_cocci

```

Ahora vamos a mejorar los nombres de las columnas. Una buena práctica es elegir siempre el mismo tipo de nomenclatura. Hay varias convenciones pero las más elegidas en el mundo de R son *snake* y *camel*. La nomenclatura *snake* implica usar guiones bajos como separador de palabras y, generalmente, todas las letras en minúscula. La nomenclatura *camel* implica, en vez de usar guiones, usar a las mayúsculas como indicador de que hay una palabra nueva, y siempre dejar a la primera palabra en minúscula. Si nuestra variable se llama “Fecha de muestreo”, la nomenclatura sería *fecha_de_muestreo* o *fechaDeMuestreo*, de acuerdo a la convención elegida.

En la tabla que tenemos, hay una mezcla de criterios: se usan guiones bajos pero a la vez las variables empiezan con mayúscula. Vamos a ajustar. Vamos a pasar todas las variables al formato *snake* con una función muy útil, que también sirve para nombres de columnas más desprolijos (por ejemplo, si tenemos otros signos de puntuación en los nombres).

```

library(janitor)

datos_RH <- clean_names(datos_RH)
datos_RH

## # A tibble: 20 x 22

```

```

##   id    pool   site   date   filam~1 large~2 vibri~3 large~4 small~5 small~6   p_h
##   <chr> <chr> <chr> <chr>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 10   RH1    shore Octo~      0    16921    5315   12676   1747   25994   4.91
## 2 20   RH2    shore Octo~     1435    5645   2384    7381    888   12075   5.52
## 3 30   RH3    shore Octo~    13307   1597    392    8131    708   13222   4.72
## 4 40   RH4    nort~ Octo~      0    8085    3499   7534    2028   25353   5.09
## 5 50   RH5    shore Octo~      0    1921    451    15662    666   31621   4.82
## 6 1D   RH1    shore Dece~      0   12374    9228   21906   4503   38421   6.39
## 7 2D   RH2    shore Dece~      0   22924    5933   34663   6687   77581   4.75
## 8 3D   RH3    shore Dece~      0   12309    3471   46807   8694   139006  4.67
## 9 4D   RH4    nort~ Dece~    44419   76819   19692   44391   24931   156132  6.75
## 10 5D  RH5    shore Dece~      0   66952   23046   134707   2949   102890  4.65
## 11 1F  RH1    shore Febr~    3177    6414    7533   22218   4013   34495   5.9
## 12 2F  RH2    shore Febr~      0   7263    4879   25886   4098   137049  5.19
## 13 3F  RH3    shore Febr~    3886    9283   1294    17402   2522   93008   4.66
## 14 4F  RH4    nort~ Febr~    5860    54624   20331   33949   12469   58990   6.65
## 15 5F  RH5    shore Febr~      0   21324   17482   24397   7064   106885  4.82
## 16 1A  RH1    shore Apri~    12331   21098   11615   16083   6592   33478   7.1
## 17 2A  RH2    shore Apri~      0   6950    4028   22525   1395   94037   4.88
## 18 3A  RH3    shore Apri~      0   12394   17281   18608   4678   121090  5.4
## 19 4A  RH4    nort~ Apri~      0    901    2484    3455    642   50957   6.2
## 20 5A  RH5    shore Apri~      0   7626    3338   17231   2016   60287  5.43
## # ... with 11 more variables: conductivity <dbl>, dissolved_oxygen <dbl>,
## #   total_hardness <dbl>, doc <dbl>, din <dbl>, total_nitrogen <dbl>,
## #   drp <dbl>, total_phosphorus <dbl>, ag_440 <dbl>, suva254 <dbl>,
## #   ms_index <dbl>, and abbreviated variable names 1: filaments, 2: large_rods,
## #   3: vibrio_shaped, 4: large_cocci, 5: small_rods, 6: small_cocci

```

El nombre de la variable pH quedó un poco raro, lo vamos a cambiar. A continuación se pide el nombre de las columnas de datos_RH, luego específicamente el de la columna 11, y luego lo cambiamos.

```
colnames(datos_RH)
```

```

## [1] "id"                  "pool"                "site"                "date"
## [5] "filaments"           "large_rods"          "vibrio_shaped"       "large_cocci"
## [9] "small_rods"          "small_cocci"         "p_h"                 "conductivity"
## [13] "dissolved_oxygen"    "total_hardness"      "doc"                 "din"
## [17] "total_nitrogen"      "drp"                 "total_phosphorus"    "ag_440"
## [21] "suva254"             "ms_index"

```

```
colnames(datos_RH)[11]
```

```
## [1] "p_h"
```

```
colnames(datos_RH) [11] <- "ph"
colnames(datos_RH)

## [1] "id"                  "pool"                 "site"                 "date"
## [5] "filaments"           "large_rods"           "vibrio_shaped"        "large_cocci"
## [9] "small_rods"           "small_cocci"          "ph"                   "conductivity"
## [13] "dissolved_oxygen"    "total_hardness"       "doc"                  "din"
## [17] "total_nitrogen"      "drp"                  "total_phosphorus"     "ag_440"
## [21] "suva254"              "ms_index"
```

Finalmente, vamos a agregar una nueva columna con la biomasa total bacteriana. Existen múltiples métodos para hacer esto, vamos a ver dos, uno con la sintaxis base de R y otro con la sintaxis de *tidyverse*. Para conocer más sobre los tipos de sintaxis de R, recomendamos la hoja de referencia Comparación de sintaxis de R, por Amelia McNamara (traducida al castellano por RLadies).

```
#Con la sintaxis base
datos_bio_opcion1 <- datos_RH #duplico el dataframe para no sobreescribirlo y comparar

datos_bio_opcion1$bio_total <- datos_bio_opcion1$filaments + datos_bio_opcion1$large_r

datos_bio_opcion1

## # A tibble: 20 x 23
##   id   pool  site  date filam~1 large~2 vibri~3 large~4 small~5 small~6   ph
##   <chr> <chr> <chr> <chr>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 10   RH1   shore Octo~     0    16921    5315   12676   1747   25994   4.91
## 2 20   RH2   shore Octo~    1435    5645   2384    7381    888   12075   5.52
## 3 30   RH3   shore Octo~   13307   1597    392    8131    708   13222   4.72
## 4 40   RH4   nort~ Octo~     0    8085    3499    7534   2028   25353   5.09
## 5 50   RH5   shore Octo~     0    1921    451    15662    666   31621   4.82
## 6 1D   RH1   shore Dece~     0   12374    9228   21906   4503   38421   6.39
## 7 2D   RH2   shore Dece~     0   22924    5933   34663   6687   77581   4.75
## 8 3D   RH3   shore Dece~     0   12309    3471   46807   8694   139006  4.67
## 9 4D   RH4   nort~ Dece~   44419   76819   19692   44391   24931  156132  6.75
## 10 5D  RH5   shore Dece~     0   66952   23046   134707  2949   102890  4.65
## 11 1F  RH1   shore Febr~   3177    6414    7533   22218   4013   34495   5.9
## 12 2F  RH2   shore Febr~     0   7263    4879   25886   4098   137049  5.19
## 13 3F  RH3   shore Febr~   3886    9283   1294    17402   2522   93008   4.66
## 14 4F  RH4   nort~ Febr~   5860    54624   20331   33949   12469   58990  6.65
## 15 5F  RH5   shore Febr~     0   21324   17482   24397   7064   106885  4.82
## 16 1A  RH1   shore Apr~  12331   21098   11615   16083   6592   33478   7.1
## 17 2A  RH2   shore Apr~     0   6950    4028   22525   1395   94037   4.88
## 18 3A  RH3   shore Apr~     0   12394   17281   18608   4678   121090  5.4
```

```

## 19 4A    RH4    nort~ Apri~      0     901    2484    3455    642    50957   6.2
## 20 5A    RH5    shore Apri~      0    7626    3338   17231    2016   60287   5.43
## # ... with 12 more variables: conductivity <dbl>, dissolved_oxygen <dbl>,
## #   total_hardness <dbl>, doc <dbl>, din <dbl>, total_nitrogen <dbl>,
## #   drp <dbl>, total_phosphorus <dbl>, ag_440 <dbl>, suva254 <dbl>,
## #   ms_index <dbl>, bio_total <dbl>, and abbreviated variable names
## #   1: filaments, 2: large_rods, 3: vibrio_shaped, 4: large_cocci,
## #   5: small_rods, 6: small_cocci

#Con la sintaxis tidyverse, la cual usa el operador _pipe_ ("la pipa") %>%
datos_bio_opcion2 <- datos_RH %>%
  mutate(bio_total = filaments + large_rods + vibrio_shaped + large_cocci + small_rods + small_cocci)

datos_bio_opcion2

## # A tibble: 20 x 23
##   id    pool   site     date bio_t~1 filam~2 large~3 vibri~4 large~5 small~6
##   <chr> <chr> <chr>   <chr> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 10    RH1    shore   Octo~  62653     0    16921    5315   12676   1747
## 2 20    RH2    shore   Octo~  29808   1435    5645    2384   7381    888
## 3 30    RH3    shore   Octo~  37357   13307   1597    392    8131   708
## 4 40    RH4    north shore Octo~  46499     0    8085    3499   7534   2028
## 5 50    RH5    shore   Octo~  50321     0    1921    451    15662   666
## 6 1D    RH1    shore   Dece~  86432     0    12374   9228   21906   4503
## 7 2D    RH2    shore   Dece~  147788    0    22924   5933   34663   6687
## 8 3D    RH3    shore   Dece~  210287    0    12309   3471   46807   8694
## 9 4D    RH4    north shore Dece~  366384   44419   76819   19692   44391   24931
## 10 5D   RH5    shore   Dece~  330544    0    66952   23046  134707   2949
## 11 1F   RH1    shore   Febr~  77850   3177    6414    7533   22218   4013
## 12 2F   RH2    shore   Febr~  179175    0    7263    4879   25886   4098
## 13 3F   RH3    shore   Febr~  127395   3886    9283   1294   17402   2522
## 14 4F   RH4    north shore Febr~  186223   5860    54624   20331   33949   12469
## 15 5F   RH5    shore   Febr~  177152    0    21324   17482   24397   7064
## 16 1A   RH1    shore   Apri~  101197   12331   21098   11615   16083   6592
## 17 2A   RH2    shore   Apri~  128935    0    6950    4028   22525   1395
## 18 3A   RH3    shore   Apri~  174051    0    12394   17281   18608   4678
## 19 4A   RH4    north shore Apri~  58439     0     901    2484    3455    642
## 20 5A   RH5    shore   Apri~  90498     0    7626    3338   17231   2016

## # ... with 13 more variables: small_cocci <dbl>, ph <dbl>, conductivity <dbl>,
## #   dissolved_oxygen <dbl>, total_hardness <dbl>, doc <dbl>, din <dbl>,
## #   total_nitrogen <dbl>, drp <dbl>, total_phosphorus <dbl>, ag_440 <dbl>,
## #   suva254 <dbl>, ms_index <dbl>, and abbreviated variable names 1: bio_total,
## #   2: filaments, 3: large_rods, 4: vibrio_shaped, 5: large_cocci,
## #   6: small_rods

```

La ventaja de la segunda opción es que podemos elegir dónde agregar la nueva columna (antes de la columna *filaments*, por ejemplo). Así que nos quedaremos con esta tabla para los siguientes pasos. De paso, borramos los dataframes que ya no necesitamos.

```
datos_RH <- datos_bio_opcion2

rm(datos_bio_opcion2)
rm(datos_bio_opcion1)
```

Un último paso es informar que las variables *pool* y *date* son categóricas (factores). En el caso de *date*, además es adecuado ordenar los niveles de los factores, tanto para algunos análisis que consideren el orden de los meses (aquí nos exceden) como para mejorar la forma en que resumimos los resultados en una tabla o en un gráfico.

```
datos_RH$pool <- factor(datos_RH$pool) #en este caso no indicamos los niveles, ya que

datos_RH$date <- factor(datos_RH$date, levels = c("October_2009", "December_2009", "Fe
```

¡Listo! Podemos empezar con los análisis.

2.3 Correlaciones

Primero vamos a realizar análisis rápidos de todos los pares de variables con la librería *GGally*.

```
library(GGally)
```

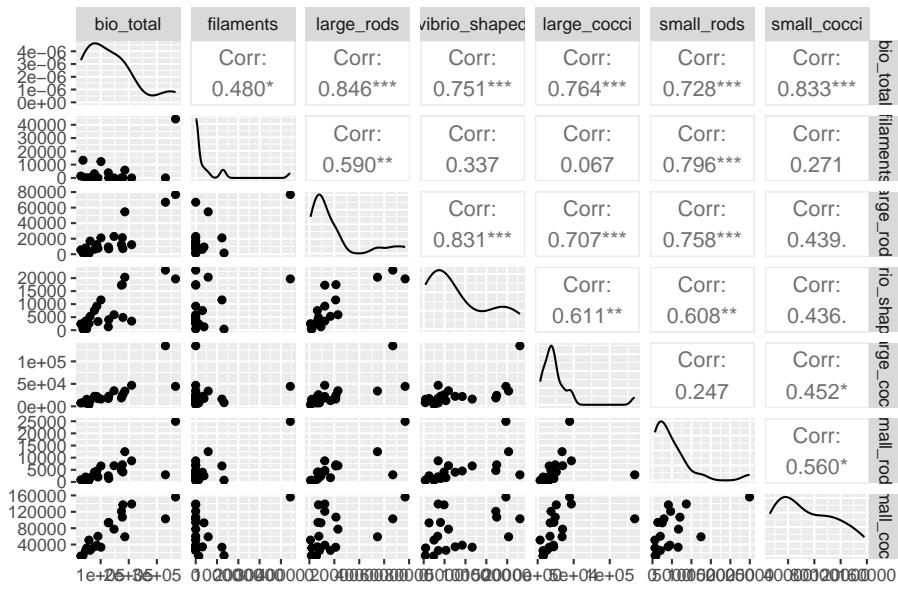
2.3.1 Entre biomasa de morfotipos bacterianos

Vamos a analizar en primer lugar las variables biológicas, para lo cual generaremos un nuevo dataframe que es un subconjunto de los anteriores.

```
datos_bacterias <- datos_RH[,5:11]

ggpairs(datos_bacterias, title="Correlograma entre la biomasa de morfotipos bacterianos")
```

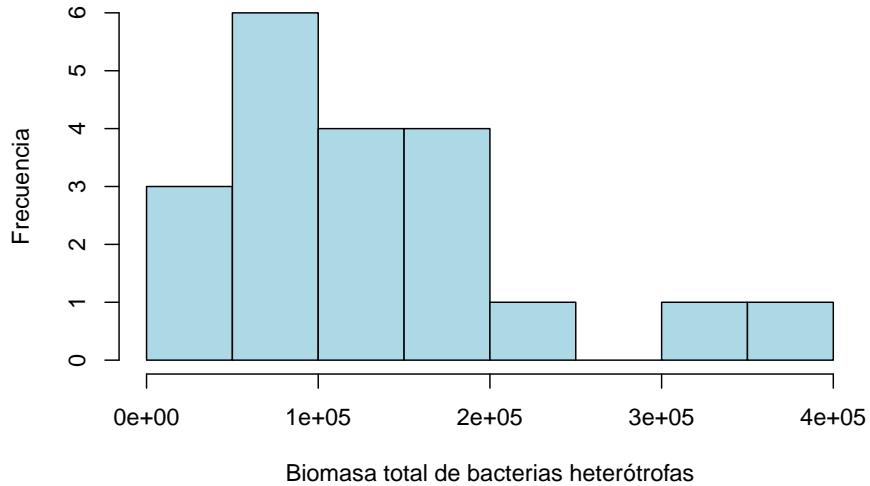
Correlograma entre la biomasa de morfotipos bacterianos



Obtenemos una matriz de gráficos, en la que cada celda expresa la correlación de un par de variables. En este caso, todas las variables son continuas, por lo que observamos que los gráficos en la parte inferior de la matriz son de dispersión (puntos).

En las diagonales, observamos histogramas suavizados para cada variable. Por ejemplo, comparemos el histograma de la biomasa total.

```
hist(datos_bacterias$bio_total, breaks=10, col="lightblue", xlab="Biomasa total de bacterias heterotróficas")
```



En la parte superior, se muestra el coeficiente de correlación para cada par de variables y asteriscos indicando su nivel de significancia. Podemos evaluar el par que nos interese con una función, para tener información numérica más detallada.

```
cor.test(datos_bacterias$vibrio_shaped, datos_bacterias$large_rods)

##
## Pearson's product-moment correlation
##
## data: datos_bacterias$vibrio_shaped and datos_bacterias$large_rods
## t = 6.3382, df = 18, p-value = 5.68e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## 0.6144454 0.9311213
## sample estimates:
## cor
## 0.8310104
```

Aquí observamos que el coeficiente de la correlación de Pearson es $r = 0.8310$, igual al que observamos en el correlograma. Además obtenemos un intervalo de confianza y un p-valor. Sin embargo, la correlación de Pearson supone que la distribución de las variables es normal. Evaluemos la normalidad de estas dos variables (deberíamos analizarlo para todas).

```
shapiro.test(datos_bacterias$vibrio_shaped)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: datos_bacterias$vibrio_shaped  
## W = 0.8493, p-value = 0.005189
```

```
shapiro.test(datos_bacterias$large_rods)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: datos_bacterias$large_rods  
## W = 0.71437, p-value = 5.866e-05
```

Dado que se rechaza el supuesto de distribución normal, tenemos dos opciones. La primera es hacer una correlación de Spearman, la cual es adecuada si suponemos que la relación entre las variables es monotónica (las variables siempre crecen o decrecen). La desventaja es que, en la fórmula de cómputo del método, las variables se ordenan en un rango y pasan a ser ordinales. Agregamos a la misma función que el método sea Spearman (por defecto es Pearson, ver la ayuda de la función con `?cor.test`). El rho es de 0.8316.

```
cor.test(datos_bacterias$vibrio_shaped, datos_bacterias$large_rods, method = "spearman")
```

```
##  
## Spearman's rank correlation rho  
##  
## data: datos_bacterias$vibrio_shaped and datos_bacterias$large_rods  
## S = 224, p-value = 2.994e-07  
## alternative hypothesis: true rho is not equal to 0  
## sample estimates:  
## rho  
## 0.8315789
```

La segunda opción es hacer una correlación de Pearson por permutaciones. Indicamos entonces el número de permutaciones a generar o dejamos el valor por defecto que es 999. Puede tardar un poco ya que tiene que permutar. El coeficiente de correlación obtenido es de 0.8310.

```
library(RVAideMemoire) #Si no encuentran esta librería en las computadoras del curso, j

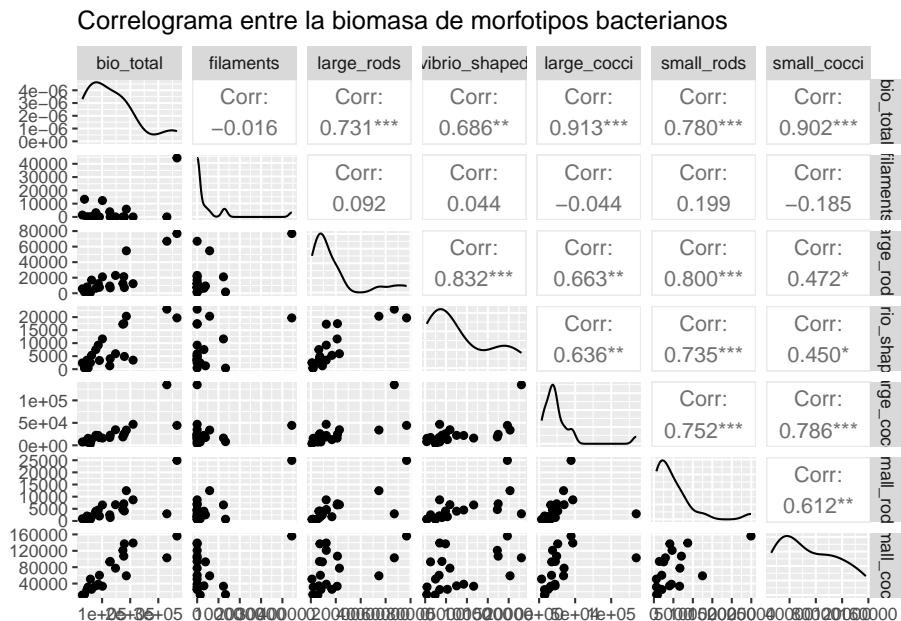
perm.cor.test(datos_bacterias$vibrio_shaped, datos_bacterias$large_rods, progress=FALSE)

## 
## Pearson's product-moment correlation - Permutation test
##
## data: datos_bacterias$vibrio_shaped and datos_bacterias$large_rods
## 999 permutations
## t = 6.3382, p-value = 0.002
## alternative hypothesis: true correlation is not equal to 0
## sample estimates:
##       cor
## 0.8310104
```

En el correlograma, vamos a cambiar qué coeficiente se muestra. Dejaremos el de Spearman. Observar que para algunos de los pares de variables hay diferencias entre los coeficientes de Pearson y Spearman.

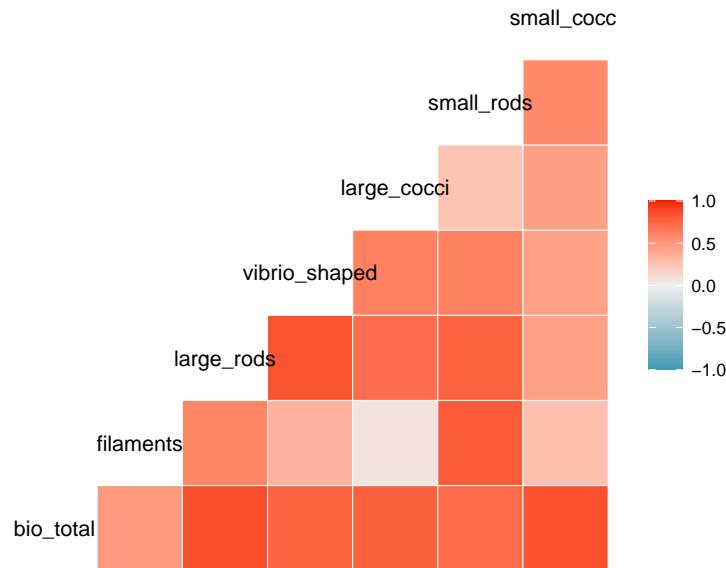
```
datos_bacterias <- datos_RH[,5:11]

ggpairs(datos_bacterias, title="Correlograma entre la biomasa de morfotipos bacterianos")
```

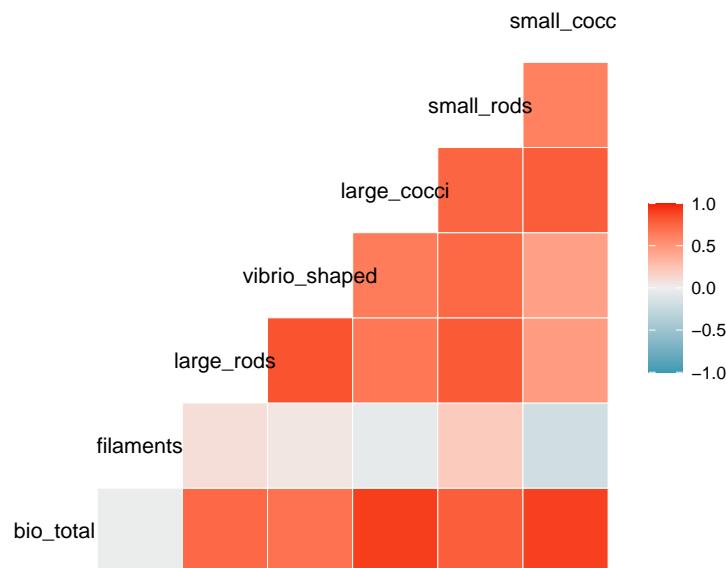


Ahora visualizaremos la correlación como un mapa de calor.

```
ggcorr(datos_bacterias, method = c("everything", "pearson"))
```



```
ggcorr(datos_bacterias, method = c("everything", "spearman"))
```



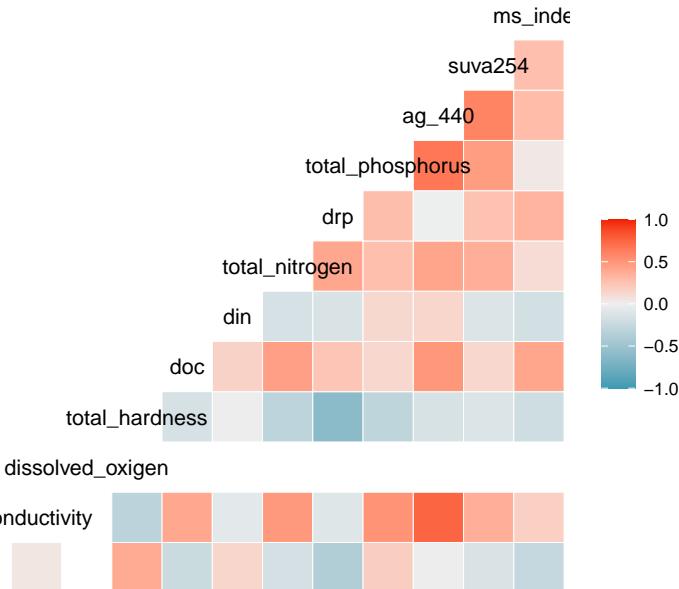
2.3.2 Entre variables abióticas

Ahora analizaremos las variables explicativas. Esto es útil ya que en el siguiente paso de realizar regresiones múltiples evitaremos colocar en el modelo dos variables muy correlacionadas entre sí, ya que son redundantes. Generamos un nuevo dataframe que es un subconjunto de los anteriores.

Podemos repetir el estudio de comparar correlaciones de Pearson, de Pearson por permutaciones y de Spearman. Aquí resumidamente pasaremos a los correlogramas visuales. ¿Qué pasa con el oxígeno disuelto? Tendremos cuidado con el uso de esta variable en los análisis siguientes.

```
datos_fq <- datos_RH[, 12:23]

ggcorr(datos_fq, method = c("everything", "spearman"))
```



Si queremos tener un resumen numérico de los coeficientes de correlación, podemos generar una matriz.

```
## Warning: package 'Hmisc' was built under R version 4.2.2

## Loading required package: lattice

## Loading required package: survival
```

```

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
## 
##     src, summarize

## The following objects are masked from 'package:base':
## 
##     format.pval, units

rcorr(as.matrix(datos_fq), type="spearman")

##          ph conductivity dissolved_oxygen total_hardness   doc   din
## ph            1.00        0.05       -0.03        0.39 -0.22  0.14
## conductivity    0.05        1.00        0.06       -0.31  0.41 -0.07
## dissolved_oxygen -0.03        0.06        1.00       -0.34 -0.03 -0.29
## total_hardness    0.39       -0.31       -0.34        1.00 -0.16  0.01
## doc            -0.22        0.41       -0.03       -0.16  1.00  0.16
## din             0.14       -0.07       -0.29        0.01  0.16  1.00
## total_nitrogen   -0.16        0.48       -0.24       -0.30  0.45 -0.15
## drp            -0.38       -0.09       -0.15       -0.60  0.24 -0.13
## total_phosphorus  0.19        0.52       -0.22       -0.29  0.13  0.13
## ag_440           -0.01        0.75       -0.32       -0.14  0.49  0.14
## suva254          -0.13        0.36       -0.63       -0.12  0.13 -0.12
## ms_index         -0.25        0.17       -0.27       -0.20  0.41 -0.18
##          total_nitrogen    drp total_phosphorus ag_440 suva254 ms_index
## ph            -0.16 -0.38        0.19 -0.01 -0.13 -0.25
## conductivity    0.48 -0.09        0.52  0.75  0.36  0.17
## dissolved_oxygen -0.24 -0.15       -0.22 -0.32 -0.63 -0.27
## total_hardness    -0.30 -0.60       -0.29 -0.14 -0.12 -0.20
## doc            0.45  0.24        0.13  0.49  0.13  0.41
## din             -0.15 -0.13        0.13  0.14 -0.12 -0.18
## total_nitrogen    1.00  0.41        0.27  0.42  0.36  0.10
## drp            0.41  1.00        0.28  0.00  0.25  0.33
## total_phosphorus  0.27  0.28        1.00  0.66  0.46  0.03
## ag_440           0.42  0.00        0.66  1.00  0.59  0.29
## suva254          0.36  0.25        0.46  0.59  1.00  0.27
## ms_index         0.10  0.33        0.03  0.29  0.27  1.00
## 
## n
##          ph conductivity dissolved_oxygen total_hardness doc din

```

```

## ph          20          20          15          20  20  20
## conductivity 20          20          15          20  20  20
## dissolved_oxygen 15          15          15          15  15  15
## total_hardness 20          20          15          20  20  20
## doc          20          20          15          20  20  20
## din          20          20          15          20  20  20
## total_nitrogen 20          20          15          20  20  20
## drp          20          20          15          20  20  20
## total_phosphorus 20          20          15          20  20  20
## ag_440        20          20          15          20  20  20
## suva254       20          20          15          20  20  20
## ms_index      20          20          15          20  20  20
##               total_nitrogen drp total_phosphorus ag_440 suva254 ms_index
## ph            20  20          20  20          20  20
## conductivity 20  20          20  20          20  20
## dissolved_oxygen 15  15          15  15          15  15
## total_hardness 20  20          20  20          20  20
## doc            20  20          20  20          20  20
## din            20  20          20  20          20  20
## total_nitrogen 20  20          20  20          20  20
## drp            20  20          20  20          20  20
## total_phosphorus 20  20          20  20          20  20
## ag_440          20  20          20  20          20  20
## suva254         20  20          20  20          20  20
## ms_index        20  20          20  20          20  20
##
## P
##               ph      conductivity dissolved_oxygen total_hardness doc
## ph            0.8450     0.9094          0.0875    0.3417
## conductivity 0.8450           0.8292          0.1788    0.0760
## dissolved_oxygen 0.9094  0.8292           0.2182    0.9142
## total_hardness 0.0875  0.1788           0.2182    0.5120
## doc            0.3417  0.0760           0.9142    0.5120
## din            0.5541  0.7813           0.2979    0.9824    0.4934
## total_nitrogen 0.5036  0.0318           0.3973    0.1926    0.0472
## drp            0.1009  0.6939           0.5901    0.0048    0.3147
## total_phosphorus 0.4266  0.0192           0.4402    0.2116    0.5765
## ag_440          0.9699  0.0001           0.2474    0.5434    0.0271
## suva254         0.5801  0.1200           0.0126    0.6090    0.5735
## ms_index        0.2850  0.4631           0.3369    0.3975    0.0714
##               din      total_nitrogen drp      total_phosphorus ag_440 suva254
## ph            0.5541  0.5036           0.1009  0.4266    0.9699  0.5801
## conductivity 0.7813  0.0318           0.6939  0.0192    0.0001  0.1200
## dissolved_oxygen 0.2979  0.3973           0.5901  0.4402    0.2474  0.0126
## total_hardness 0.9824  0.1926           0.0048  0.2116    0.5434  0.6090
## doc            0.4934  0.0472           0.3147  0.5765    0.0271  0.5735

```

```

## din           0.5348    0.5760 0.5876    0.5560 0.6267
## total_nitrogen 0.5348    0.0740 0.2464    0.0675 0.1193
## drp          0.5760 0.0740                0.2333    0.9873 0.2787
## total_phosphorus 0.5876 0.2464    0.2333    0.0015 0.0393
## ag_440        0.5560 0.0675    0.9873 0.0015    0.0057
## suva254       0.6267 0.1193    0.2787 0.0393    0.0057
## ms_index      0.4556 0.6686    0.1489 0.8907    0.2144 0.2494
##               ms_index
## ph            0.2850
## conductivity 0.4631
## dissolved_oxygen 0.3369
## total_hardness 0.3975
## doc           0.0714
## din           0.4556
## total_nitrogen 0.6686
## drp           0.1489
## total_phosphorus 0.8907
## ag_440        0.2144
## suva254       0.2494
## ms_index

```

2.4 Regresiones múltiples

Intro

hay que estandarizar?

2.4.1 Selección automática de variables

2.4.1.1 Upward: del modelo nulo, sumando variables explicativas

2.4.1.2 Forward: del modelo completo, quitando variables explicativas

2.4.2 Selección manual de variables

2.4.2.1 Modelos univariados

Todos los modelos AIC

2.4.2.2 Modelos de regresión múltiple

vif

2.4.3 Evaluación e interpretación del modelo final

Evaluación de supuestos. Si no se cumplen, permutaciones interpretación

2.5 A modo de conclusión

Chapter 3

Modelos Aditivos Generalizados

María Victoria Quiroga¹

Instituto Tecnológico de Chascomús (INTECH, UNSAM-CONICET), Escuela de Bio y Nanotecnologías (UNSAM)

3.0.1 Aclaración

En esta unidad se utiliza el paquete *ggplot2* (?) para generar algunos gráficos. Para comprender el código empleado se recomienda leer el *Capítulo 3: Data visualization* del libro *R for Data Science* (?).

3.1 ¿Porqué usar Modelos Aditivos Generalizados (GAMs)?

1. La relación entre variables predictoras (i.e., independientes) y la variable respuesta (i.e., dependiente) **no** necesita ser **lineal**.
2. **No** necesitamos conocer de antemano la forma funcional de la relación.
3. Son modelos muy flexibles que permiten la interpretación (de manera gráfica) de los efectos parciales de cada variable independiente.
4. Podemos:
 - Incluir predictores categóricos e interacciones.
 - Usar distribuciones diferentes a la normal para la variable dependiente.

¹mvquirosa@iib.unsam.edu.ar

- Incluir correlaciones entre observaciones (e.g., medidas repetidas, diseños anidados) -modelos mixtos.

3.2 ¿Qué son los GAMs?

Los GAM permiten incorporar los efectos additivos de distintas variables independientes (i.e., explicativas) utilizando funciones suaves (*smooth functions*).

$$g(\mu) = \beta_0 + f_1(x_1) + f_2(x_2) + \dots + f_n(x_n) + \varepsilon$$

Donde μ es el valor esperado de la variable dependiente y que puede presentar distribuciones de la familia exponencial, g es la función de enlace, β_0 es el intercepto, x_n son las n variables independientes y cada f_n es una función suave que se estima de manera no paramétrica. Se pueden incluir funciones suaves para interacciones, por ejemplo entre las variables independientes 1 y 2 $f(x_1, x_2)$. Si se incluyen términos aditivos aleatorios, por ejemplo la autocorrelación en series de tiempo, se generan modelos mixtos (GAMM, *generalized additive mixed model*).

Existen diferentes estrategias para ajustar una curva suave y continua a los datos. Aquí vamos a utilizar *splines*, y en particular *spline* cúbico y cúbico cíclico. De manera muy breve, un *spline* cúbico es una curva suave cúbica por tramos (Figura ??). El *spline* cúbico cíclico empieza y termina en el mismo punto. El paquete con el que vamos a trabajar *mgcv* (?) usa splines penalizados (*conventional integrated square second derivative cubic spline penalty*), donde la penalización será más grande cuanto menos suave sea la curva.

3.3 Manos a la obra!

Utilizaremos un set de datos del trabajo *The dynamics of picocyanobacteria from a hypereutrophic shallow lake is affected by light-climate and small-bodied zooplankton: a 10-year cytometric time-series analysis* publicado en *FEMS Microbiology Ecology* (?), disponibles en el Repositorio Institucional CONICET Digital.

Descargar el set de datos **data_gam.csv** de GitHub Limno-con-R/CILCAL2023. Guardar el archivo en una carpeta llamada *data*, dentro del **Directorio de Trabajo del Proyecto** que creamos para esta Unidad (ver cómo hacerlo en la Unidad ??).

Instalar los paquetes como se indica en la Unidad ?. Luego, cargarlos en la sesión

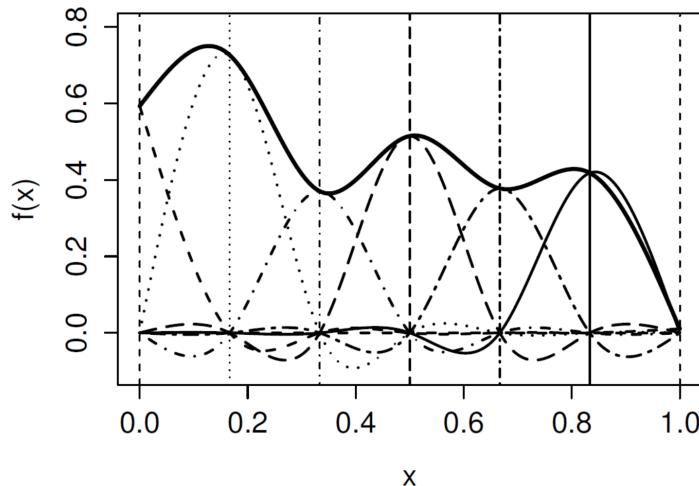


Figure 3.1: Construcción de un spline cúbico. La curva suave (línea continua gruesa) es la suma de las 5 funciones basis (líneas finas). Las líneas verticales muestran los nodos equiespaciados. Extraído de Wood (2017).

```
library(mgcv)
library(ggplot2)
library(itsadug)
library(data.table)
library(car)
library(grid)
library(lubridate)
```

Leer datos, dar formato `as.Date` a la fecha y generar las variables `Mes` y `Dias`.

```
base <- read.csv("data/data_gam.csv")
base$Fecha <- as.Date(base$Fecha, "%m/%d/%Y") # formato de fecha en inglés
base$Mes <- as.numeric(format(base$Fecha, '%m')) # generar variable Mes
base$Dia1 = rep(base$Fecha[1], nrow(base)) # fecha inicial de la serie
base$Dias <- (interval(base$Dia1, base$Fecha) %/ days(1))+1 # generar variable Dias
```

Inspección visual de primeras filas de la tabla. Para ver toda la tabla se puede utilizar `View(base)`.

```
head(base)
```

```
##           Fecha Pcy_orgml Mes      Dia1 Dias
## 1 2007-05-22    6660000   5 2007-05-22     1
```

```
## 2 2007-06-05 6830000 6 2007-05-22 15
## 3 2007-06-21 4700000 6 2007-05-22 31
## 4 2007-07-03 4680000 7 2007-05-22 43
## 5 2007-07-17 6500000 7 2007-05-22 57
## 6 2007-07-31 8110000 7 2007-05-22 71
```

Ver la estructura de los datos

```
str(base)
```

```
## 'data.frame': 225 obs. of 5 variables:
## $ Fecha : Date, format: "2007-05-22" "2007-06-05" ...
## $ Pcy_orgml: num 6660000 6830000 4700000 4680000 6500000 ...
## $ Mes : num 5 6 6 7 7 7 8 8 9 9 ...
## $ Dia1 : Date, format: "2007-05-22" "2007-05-22" ...
## $ Dias : num 1 15 31 43 57 71 85 99 113 127 ...
```

`base` es un objeto `data.frame` con 225 `obs.` observaciones o filas y 5 `variables` o columnas. Variables: `Fecha` con formato `Date`. `Pcy_orgml`, abundancia de picocianobacterias (organismos/ml) en la laguna Chascomús, como valores numéricos `num`.

Se generaron las variables `Mes` y `Dias` como números `num`. `Mes` indica el #mes (1-12) de la fecha de muestreo y `Dias` indica el #días transcurridos desde la primer fecha (considerando la primer fecha como día #1).

Grafico exploratorio de la serie temporal

```
ggplot(base, aes(x= Dias, y= Pcy_orgml)) +
  geom_point() +
  geom_line()+
  theme(legend.position = "none")
```

No se observan outliers en el gráfico.

Para datos de conteo (e.g., organismos/ml) generalmente se utiliza la distribución de Poisson, pero las series temporales de abundancia de microorganismos suelen presentar sobredispersión (Figura ??). Para considerar una varianza mayor que la media se podría utilizar la distribución binomial negativa o la *quasi-familia* quasipoisson. En este ejemplo se implementará `family=quasipoisson`.

La autocorrelación temporal de los datos se incluye utilizando una estructura autorregresiva de primer orden continua `correlation = corCAR1(form = ~ Dias)` en un modelo mixto `gamm()`.

Como primer modelado se generan funciones suaves `s()` para el efecto estacional `Mes` y el efecto interanual (*trend*) `Dias`. Se utiliza *spline* cúbico cíclico para

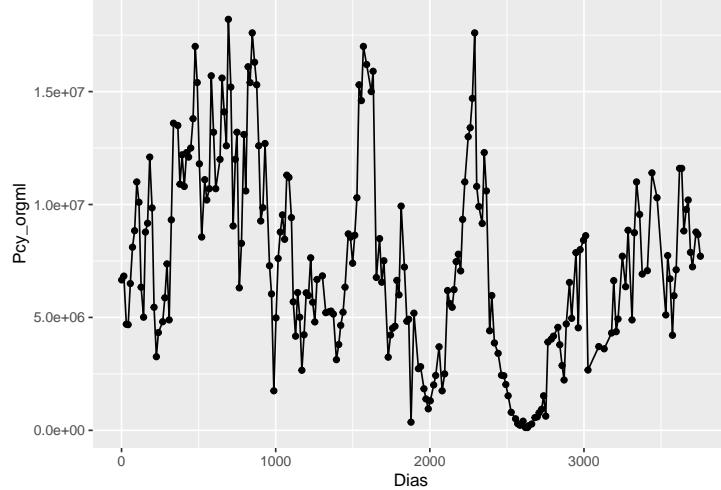


Figure 3.2: Serie temporal de abundancia (organismos/ml) de picocianobacterias.

el primer efecto `s(Mes, bs="cc")` y cúbico para el segundo efecto `s(Dias, bs="cr")`.

```
modelo1<-gamm(Pcy_orgml ~ s(Mes, bs = "cc") + s(Dias, bs="cr"),
family=quasipoisson, data = base,
correlation = corCAR1(form = ~ Dias))
```

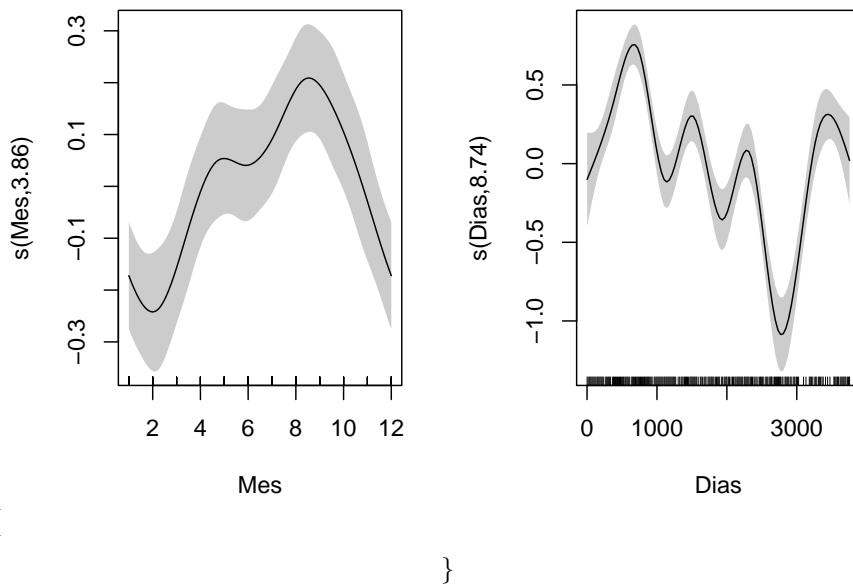
```
##  
## Maximum number of PQL iterations: 20  
  
## iteration 1  
  
## iteration 2  
  
## iteration 3  
  
## iteration 4
```

Por *default* el máximo número de iteraciones está seteado en 20 `niterPQL=20`. Si el modelo no converge se puede incrementar el número de interacciones, por ejemplo: `gamm(..., niterPQL=40)`.

Interpretación visual de los efectos parciales: curvas suaves `s()` (*smooth functions*)

```
plot(modelo1$gam, scale=0, scheme=1, pages=1)
```

\begin{figure}



\caption{Efectos parciales del modelo 1. Las curvas suaves se centraron en cero, se indican los intervalos de confianza de 95% en gris. Las líneas internas en los ejes x (Mes y Dias) representan los datos.} \end{figure}

Información del modelo

```
summary(modelo1$gam)
```

```
##  
## Family: quasipoisson  
## Link function: log  
##  
## Formula:  
## Pcy_orgml ~ s(Mes, bs = "cc") + s(Dias, bs = "cr")  
##  
## Parametric coefficients:  
##             Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 15.72716    0.02977   528.2 <2e-16 ***  
## ---
```

```

## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df      F p-value
## s(Mes) 3.858 8.000 3.674 3.46e-06 ***
## s(Dias) 8.743 8.743 23.752 < 2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) = 0.564
## Scale est. = 1.209e+06 n = 225

```

Se especifica la familia que se utilizó ## Family: quasipoisson y la función de enlace ## Link function: log. ?family muestra las familias y funciones de enlace que se utilizan por *default*. Para cambiar la función de enlace solo hay que especificarlo en el código, ejemplo family = Gamma(link = "log"). Si no se especifica una familia, gamm usa distribución gaussiana con función de enlace identidad. La ayuda ?gamm muestra todos los argumentos y los *defaults*.

Se observa que ambos términos de suavizado son significativos ($p\text{-value} < 0.05$), y el R^2_{adj} del modelo es 0.5640034.

Evaluación del modelo

```

windows()
par(mfrow=c(2,2))
gam.check(modelo1$gam, type="pearson")

##
## 'gamm' based fit - care required with interpretation.
## Checks based on working residuals may be misleading.
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##          k'  edf k-index p-value
## s(Mes) 8.00 3.86    0.93    0.14
## s(Dias) 9.00 8.74    0.31 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Los gráficos de la izquierda de la Figura ?? muestran que es correcto utilizar la *quasi familia* quasipoisson. El gráfico superior derecho muestra una dispersión bastante homogénea de los residuos, por lo que no habría mayores problemas con la varianza. El gráfico inferior derecho muestra una relación lineal positiva entre los valores observados y los predichos por el modelo. Si se logra mejorar el modelo, se observará una mejoría en este gráfico y en el R^2_{adj} del modelo.

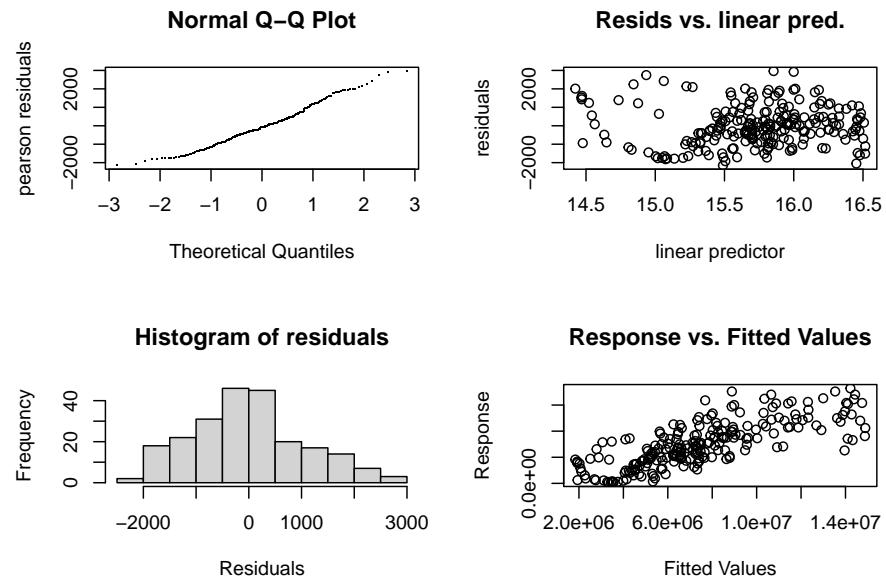


Figure 3.3: gam.check del modelo 1.

En el modelo 1 no se especificó el k en los términos de suavizado `s()`, se utilizó la opción `default k = 10`. Este argumento especifica la dimensión de las funciones *basis* del *spline*. Cuando se indica un valor de k , este determina el máximo grado de libertad permitida para ese término del modelo. Sin embargo, los grados de libertad efectivos `edf` para cada término los estima el modelo a través de la penalización, siendo el límite superior $k' = k - 1$. No especificar el argumento `k` equivale a indicar `k = 10`.

El `gam.check()` nos indica que el `k` del término `s(Dias)` del modelo 1 es muy bajo (valor de `p-value` bajo y `edf` cercano a k'). Ver `?choose.k` para mas detalles.

Para mejorar el modelo se modifica `k`. Se disminuye `k = 6` en `s(Mes)` para evitar que incrementen los `edf` de ese término, y se aumenta `k = 20` en `s(Dias)`. Además, se agrega un término de interacción `ti()` adecuando para trabajar con variables que tienen diferentes unidades (meses *versus* días).

```
modelo2<-gamm(Pcy_orgml ~ s(Mes, bs = "cc", k=6) + s(Dias, bs="cr", k=20)
               + ti(Mes,Dias, bs=c("cc","cr")), family=quasipoisson,
               data = base, correlation = corCAR1(form = ~ Dias))
```

```
##  
## Maximum number of PQL iterations: 20
```

```
## iteration 1
```

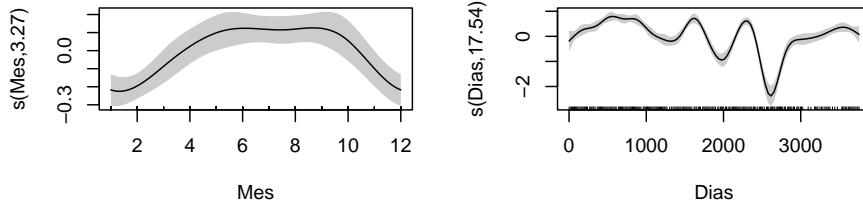
```
## iteration 2
```

```
## iteration 3
```

Interpretación visual de los efectos parciales: curvas suaves `s()` (*smooth functions*)

```
plot(modelo2$gam, scale=0, scheme=1, pages=1)
```

\begin{figure}



{

}

\caption{Efectos parciales del modelo 2. Las curvas suaves se centraron en cero, se indican los intervalos de confianza de 95% en gris. Las líneas internas en los ejes x (Mes y Dias) representan los datos.} \end{figure}

La interacción se muestra en 3D, y es difícil de interpretar visualmente. Aquí nos centramos en la interpretación visual de los efectos parciales.

Información del nuevo modelo

```
summary(modelo2$gam)

##
## Family: quasipoisson
## Link function: log
##
## Formula:
## Pcy_orgml ~ s(Mes, bs = "cc", k = 6) + s(Dias, bs = "cr", k = 20) +
##      ti(Mes, Dias, bs = c("cc", "cr"))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.64156   0.02398 652.3 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df     F p-value
## s(Mes)       3.268  4.00 9.085 <2e-16 ***
## s(Dias)      17.539 17.54 28.940 <2e-16 ***
## ti(Mes,Dias) 3.718 12.00  0.595  0.0395 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.753
## Scale est. = 5.8774e+05 n = 225
```

Los efectos parciales de `s(Mes)` y `s(Dias)` son significativos, y la interacción `ti(Mes,Dias)` es significativa. El R^2_{adj} del modelo aumentó a 0.7532274.

Evaluación del modelo

```
windows()
par(mfrow=c(2,2))
gam.check(modelo2$gam, type="pearson")
```

```
##
## 'gamm' based fit - care required with interpretation.
## Checks based on working residuals may be misleading.
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##             k'    edf k-index p-value
## s(Mes)       4.00  3.27    0.84   0.005 **
```

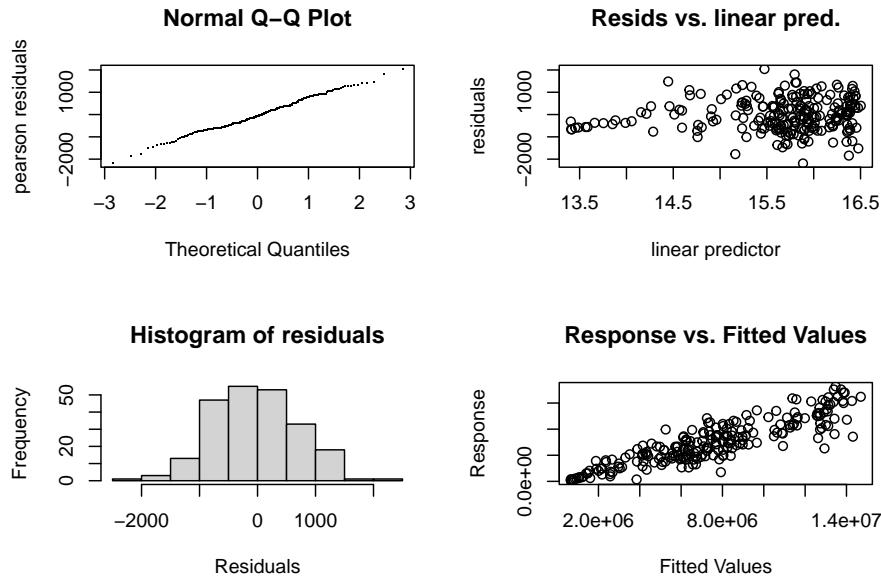


Figure 3.4: gam.check del modelo 2.

```
## s(Dias)      19.00 17.54     0.66 <2e-16 ***
## ti(Mes,Dias) 12.00  3.72     0.89   0.010 **
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

De nuevo, parece correcto utilizar la *quasi familia* quasipoisson, sin mayores problemas con la varianza. La relación lineal positiva entre los valores observados y los predichos (gráfico inferior derecho) ha mejorado, coincidiendo con el incremento del R^2_{adj} del modelo.

Los **edf** del término **s(Mes)** son cercanos a 3, muy similares al primer modelo, lo cual corrobora que la especificación de **k** en el modelo no es crítica (modelo 1 **k** = 10, modelo 2 **k** = 6), siempre que se considere un límite superior adecuado. Para **s(Dias)** el p-value sigue siendo bajo y **edf** sigue cercano a **k'**, pero **k-index** aumentó a 0.66. Se considera que no es necesario aumentar el **k** porque complejizaría el modelo, y consecuentemente la interpretación visual de la tendencia (*trend, efecto Dias*).

Se pueden comparar ambos modelos estimando su AIC

```
AIC(modelo1$lme,modelo2$lme)
```

##	df	AIC

```
## modelo1$lme 6 311.8362  
## modelo2$lme 8 228.2012
```

Efectivamente el modelo 2 presenta un AIC más bajo.

Otros gráficos

Se puede graficar con `vis.gam()`.

```

vis.gam(modelo2$gam, type="response",
        view=c("Mes", "Dias"), main="Pcy (org/ml)",
        plot.type = "contour", contour.col="black", color="terrain")
points(base$Mes,base$Dias, pch = 20) #agregar puntos de muestreo

```

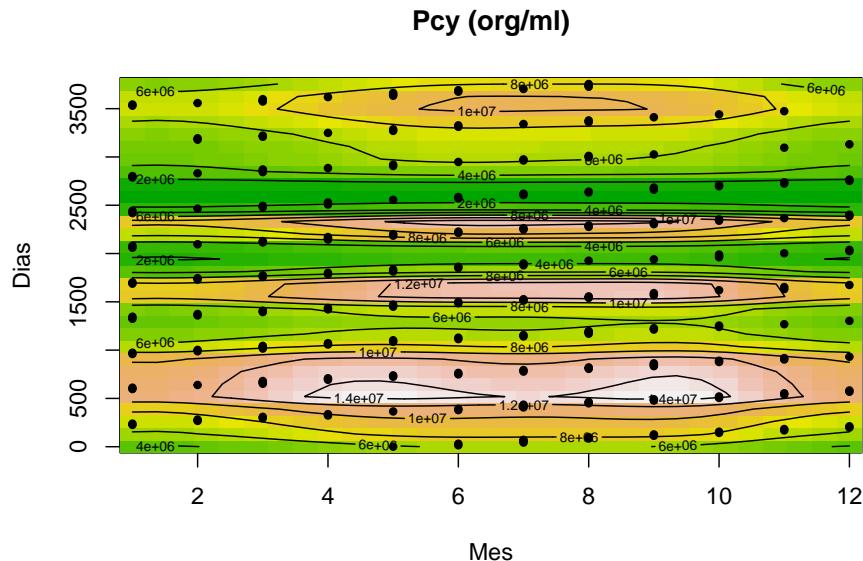


Figure 3.5: vis.gam plot del modelo 2.

Se pueden graficar los valores observados a campo y los predichos por el modelo. Cuanto mas alto sea el R^2_{adj} , mas similares serán ambos valores.

```

datas[, datos := c(rep("Observados", nrow(Base)),
                   rep("Predichos", nrow(Base)))]  
  

# generar el gráfico  

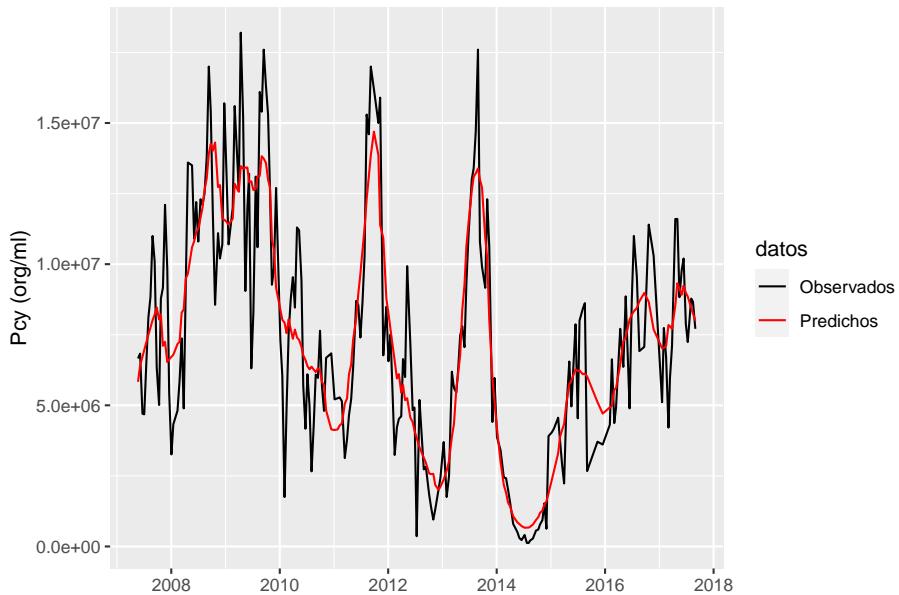
ggplot(data = datas, aes(Fecha,Pcy_orgml, group = datos))+  

  geom_line(aes(colour = datos))+  

  scale_color_manual(values=c("black","red"))+  

  labs(x = "", y = "Pcy (org/ml)")

```



3.4 Agradecimientos

Un agradecimiento muy especial para los Profesores **Gerardo Cueto** y **Adriana Pérez** de la FCEN-UBA.

Chapter 4

Análisis multivariados I

Patricia E. García¹

Instituto de Investigaciones en Biodiversidad y Medioambiente (INIBIOMA,
CONICET-UNComahue)

En esta unidad se verán ejemplos de ACP, NMDS, PERMANOVA

4.1 Análisis de Componentes Principales

Introducción: Algunos sistemas no pueden definirse por sus características individuales, sino que deben verse en conjunto. Un ejemplo muy sencillo para entender este concepto es intentar describir una cara humana con un solo rasgo: la nariz.

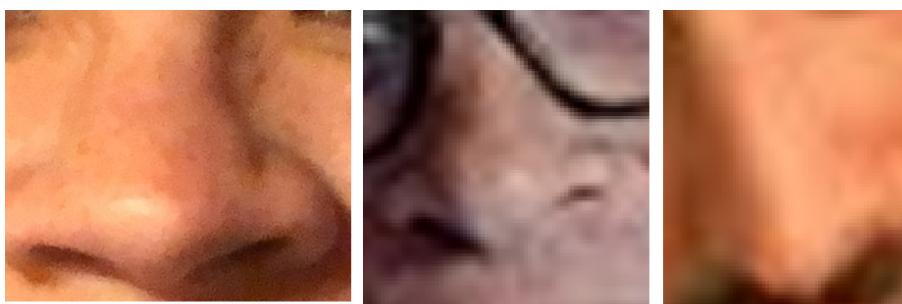


Figure 4.1: Se puede identificar una cara solo describiendo la nariz?

¹garciapatriciaelizabeth@gmail.com

Para describir y reconocer una cara es necesario observar todas sus características: la nariz, los ojos, la boca, las cejas, etc. Es decir, múltiples dimensiones.

El **análisis de componentes principales (ACP)** también conocido por siglas en inglés Principal Componentes Analysis PCA, es una poderosa herramienta estadística para describir un conjunto de datos.

Se basa en la reducción de la multidimensionalidad de una manera “*no supervisada*”.

Su principal objetivo es proyectar datos en las direcciones de **máxima varianza** y de este modo eliminar aquellas direcciones o planos que aporten menos información.

El ACP puede aplicarse antes de realizar un análisis de regresión o simplemente con fines exploratorios para ayudar a los investigadores a comprender relaciones entre las variables o descubrir patrones presentes en los datos.

4.1.1 ¿Qué tipos de datos usa una ACP?

Utiliza principalmente variable *numéricas continuas* que son las que efectivamente entran en el análisis. Estás variables se organizan en **columnas**, mientras que en las **filas** se organizan los individuos.

Individuos	Variable 1	Variable 2	Variable 3
1	15	0.6	70
2	25	0.1	50
3	24	0.2	30
4	13	0.3	55
5	23	0.5	54
6	22	0.4	31
7	19	0.1	87

Figure 4.2: Organización de los datos

Estas bases de datos son de doble entrada, por lo que es posible analizarla desde el punto de vista de los individuos, de tal manera que *individuos* que tiene características similares estarán más cerca.

Desde el punto de las *variables*, se pueden analizar las relaciones entre las mismas. El análisis considera relaciones lineales entre las variables y usa coeficientes de correlación. Es importante utilizar la matriz de correlación para observar las relaciones más importantes entre las variables.

4.1.2 ¿Cómo funciona el análisis?

El funcionamiento del análisis es muy sencillo, trata de buscar la *dirección o componente* que maximice la variación de datos. Es decir que el primer plano/componente trata de capturar la mayor dispersión de los datos.

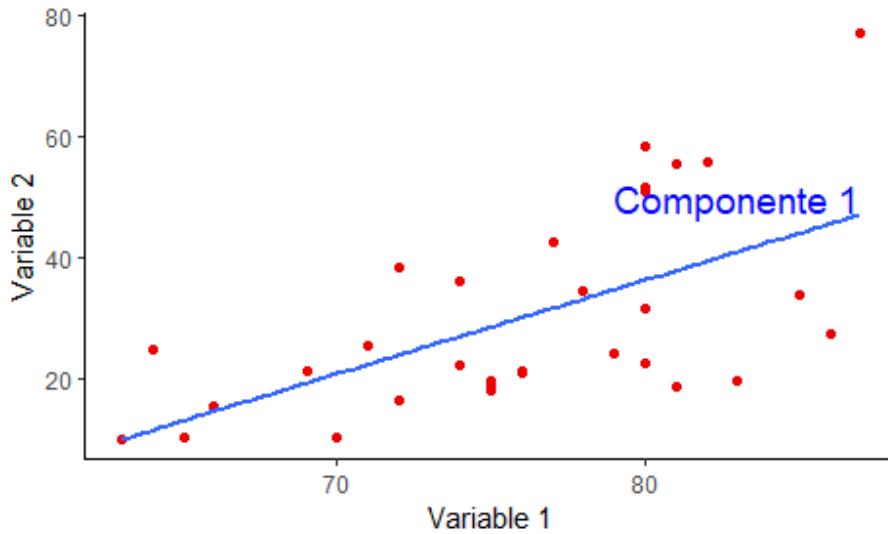
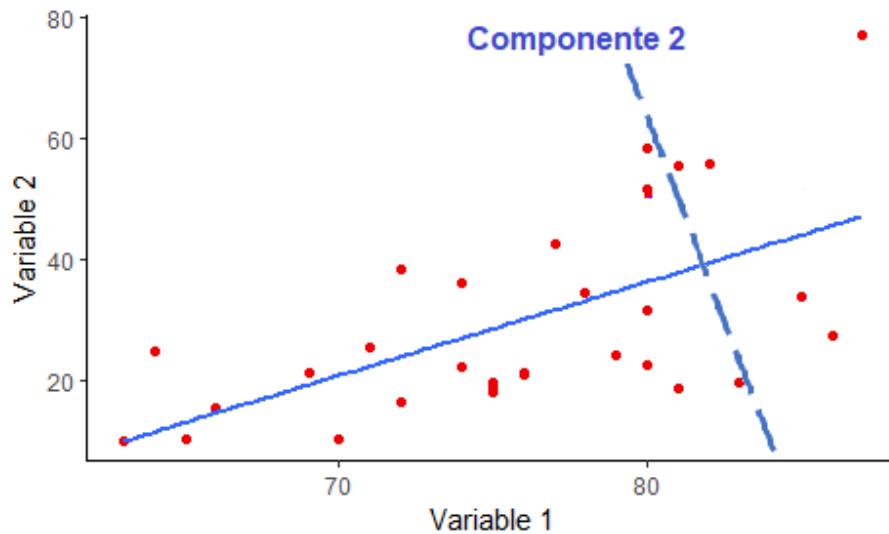


Figure 4.3: El primer componente

En el ACP, luego que se encontró el primer componente que maximizó la dispersión de los datos, el segundo componente es ortogonal, es decir a 90° .



4.1.3 Ahora manos a la obra: miremos datos

En RStudio importamos la base de datos ambientales de Castro Berman et al., 2018, disponible como **data_cursoR.xlsx** en GitHub
Limno-con-R/CILCAL2023. Recuerden guardar el archivo en una carpeta llamada *data*, dentro del **Directorio de Trabajo del Proyecto** que creamos para esta Unidad (ver cómo hacerlo en la Unidad ??).

```
## New names:  
## * `` -> `...1`
```

```
datos <- data.frame(datos, row.names = 1)
head(datos)
```

ELEVACION LATITUD LONGITUD Temp Secchi DO pH Conductivity NTU TON

```

## SL_13      62 -38.4184 -61.7594   17    44 11.8 8.55      0.32 15.7 3438
## SL_17      20 -38.9263 -61.3687    7     8 8.0 8.53      3.43 363.0 4010
## SL_18     119 -38.3474 -60.4527   14    12 9.0 8.84      4.05 110.0 4200
## SL_19      89 -38.4273 -60.3181   14    24 10.8 8.54      1.10 54.4 4738
## SL_20      64 -38.5216 -60.1404   10     9 9.0 8.44      3.25 206.0 7470
## SL_21     194 -38.0508 -60.0340   11    16 10.0 8.30      3.35 129.0 4256
##      TDON    SPM    POM     TP     TDP   Chl.a     DOC
## SL_13 2565  11.7   9.2   124    0.1   3.79   4.55
## SL_17 2890 377.5 103.4   272  114.0  68.63 10.98
## SL_18 3909  71.0  23.4  1009  833.0  54.36 253.50
## SL_19 4424  30.9  23.4   162    4.0   66.75 10.12
## SL_20 5107 186.6  62.5   727  456.0  475.36 95.43
## SL_21 3237 111.0  33.4 1365 1363.0  61.81 177.70

```

La base de datos que importamos contiene información acerca de la localización geográfica de los sitios y las variables físico-químicas.

```
summary(datos)
```

```

##      ELEVACION        LATITUD        LONGITUD       Temp
##  Min.   :-16.00   Min.   :-38.93   Min.   :-63.09   Min.   : 7.00
##  1st Qu.: 10.00  1st Qu.:-37.49  1st Qu.:-62.25  1st Qu.:15.00
##  Median : 58.50  Median :-36.84  Median :-60.23  Median :17.50
##  Mean   : 73.65  Mean   :-36.62  Mean   :-60.13  Mean   :18.52
##  3rd Qu.:104.00 3rd Qu.:-35.69  3rd Qu.:-57.98  3rd Qu.:22.25
##  Max.   :246.00  Max.   :-34.48  Max.   :-56.98  Max.   :30.00
##
##      Secchi         DO          pH      Conductivity
##  Min.   : 7.00  Min.   : 5.000  Min.   :8.000  Min.   : 0.320
##  1st Qu.:12.00 1st Qu.: 8.725  1st Qu.:8.557  1st Qu.: 1.215
##  Median :18.00  Median :10.300  Median :8.795  Median : 2.615
##  Mean   :29.44  Mean   :10.238  Mean   :8.762  Mean   : 7.968
##  3rd Qu.:37.75 3rd Qu.:11.050  3rd Qu.:8.982  3rd Qu.: 5.513
##  Max.   :140.00  Max.   :20.000  Max.   :9.400  Max.   :202.100
##
##      NTU          TON        TDON        SPM
##  Min.   : 2.70  Min.   : 2856  Min.   :1926  Min.   : 6.70
##  1st Qu.:27.90 1st Qu.: 4104  1st Qu.:3052  1st Qu.: 36.20
##  Median :83.25  Median : 4872  Median :3478  Median : 75.45
##  Mean   :88.67  Mean   : 5063  Mean   :3790  Mean   :116.83
##  3rd Qu.:110.00 3rd Qu.: 5323  3rd Qu.:4356  3rd Qu.:128.53
##  Max.   :363.00  Max.   :10830  Max.   :7235  Max.   :788.40
##
##      POM          TP        TDP        Chl.a
##  Min.   : 6.70  Min.   : 46.0  Min.   : 0.1  Min.   : 1.58

```

```

## 1st Qu.: 20.00 1st Qu.: 324.0 1st Qu.: 114.0 1st Qu.: 15.80
## Median : 29.60 Median : 533.0 Median : 302.0 Median : 52.05
## Mean    : 40.80 Mean   : 766.1 Mean   : 560.3 Mean   : 87.80
## 3rd Qu.: 47.92 3rd Qu.: 820.5 3rd Qu.: 624.0 3rd Qu.: 89.13
## Max.    :344.00 Max.   :4538.0 Max.   :4140.0 Max.   :981.06
##
##          DOC
## Min.    : 1.26
## 1st Qu.: 12.36
## Median : 19.45
## Mean   : 93.22
## 3rd Qu.: 97.58
## Max.   :1010.00
## NA's   :1

##          ELEVACION      LATITUD      LONGITUD      Temp
## Min.    :-16.00     Min.   :-38.93     Min.   :-63.09     Min.   : 7.00
## 1st Qu.: 10.00     1st Qu.:-37.49     1st Qu.:-62.25     1st Qu.:15.00
## Median : 58.50     Median :-36.84     Median :-60.23     Median :17.50
## Mean   : 73.65     Mean   :-36.62     Mean   :-60.13     Mean   :18.52
## 3rd Qu.:104.00     3rd Qu.:-35.69     3rd Qu.:-57.98     3rd Qu.:22.25
## Max.   :246.00     Max.   :-34.48     Max.   :-56.98     Max.   :30.00
##
##          Secchi        DO         pH      Conductivity
## Min.    : 7.00     Min.   : 5.000     Min.   :8.000     Min.   : 0.320
## 1st Qu.: 12.00     1st Qu.: 8.725     1st Qu.:8.557     1st Qu.: 1.215
## Median : 18.00     Median :10.300     Median :8.795     Median : 2.615
## Mean   : 29.44     Mean   :10.238     Mean   :8.762     Mean   : 7.968
## 3rd Qu.: 37.75     3rd Qu.:11.050     3rd Qu.:8.982     3rd Qu.: 5.513
## Max.   :140.00     Max.   :20.000     Max.   :9.400     Max.   :202.100
##
##          NTU          TON       TDON       SPM
## Min.    : 2.70     Min.   : 2856     Min.   :1926     Min.   : 6.70
## 1st Qu.: 27.90     1st Qu.: 4104     1st Qu.:3052     1st Qu.: 36.20
## Median : 83.25     Median : 4872     Median :3478     Median : 75.45
## Mean   : 88.67     Mean   : 5063     Mean   :3790     Mean   :116.83
## 3rd Qu.:110.00     3rd Qu.: 5323     3rd Qu.:4356     3rd Qu.:128.53
## Max.   :363.00     Max.   :10830     Max.   :7235     Max.   :788.40
##
##          POM          TP        TDP      Chl.a
## Min.    : 6.70     Min.   : 46.0     Min.   : 0.1     Min.   : 1.58
## 1st Qu.: 20.00     1st Qu.: 324.0    1st Qu.:114.0    1st Qu.: 15.80
## Median : 29.60     Median : 533.0    Median : 302.0    Median : 52.05
## Mean   : 40.80     Mean   : 766.1    Mean   : 560.3    Mean   : 87.80
## 3rd Qu.: 47.92     3rd Qu.: 820.5    3rd Qu.: 624.0    3rd Qu.: 89.13
## Max.   :344.00     Max.   :4538.0    Max.   :4140.0    Max.   :981.06

```

```
##      DOC
## Min. : 1.26
## 1st Qu.: 12.36
## Median : 19.45
## Mean   : 93.22
## 3rd Qu.: 97.58
## Max.  :1010.00
## NA's   :1
```

En resumen, tenemos 52 observaciones (individuos) y 17 variables. De las variables presentes las primeras 3 (Elevación, Longitud y Latitud) corresponden a la localización de los ambientes.

Las variables son continuas y cada una está medida de manera distintas por lo que cada una tiene su unidad y a su vez su variación. Debido a esto es necesario **estandarizar** y **centrar** las variables. La estandarización simplemente le resta a cada observación el promedio de la variable y la divide por la desviación estandar de la variable. Mientras que centrar significa que se mueve la nube de puntos de los individuos al centro de gravedad.

```
?scale # el comando para estandarizar los datos
```

```
## starting httpd help server ... done
datos.stand <- data.frame(scale(datos [4:17], center = T))
```

Para este ejemplo solo usé las variables físico-químicas, para ello utilicé los corchetes e indiqué el número de columnas que quería se estandarizaran, de la columna 4 a la columna 17: [4:17].

El comando para realizar el análisis de componentes principales , está en el paquete FactoMiner. Este paquete es muy versatil y se mantiene actualizado.

Aquí adjunto el link:

[FactoMineR](#)

```
## Warning: package 'FactoMineR' was built under R version 4.2.3
## Warning in PCA(datos.stand[-2], graph = FALSE): Missing values are imputed by
## the mean of the variable: you should use the imputePCA function of the missMDA
## package
```

¡Recibimos un **mensaje de advertencia**! Este mensaje nos indica que hay valores faltantes y que el comando va a usar el promedio de la variable en las celdas donde faltan datos.

Importante: algunos análisis son muy sensibles a la falta de datos.

¡Listo hemos realizado el análisis de ACP! El objeto “res.pca” es una lista que tiene toda la información.

```
print(res.pca)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 52 individuals, described by 13 variables
## *The results are available in the following objects:
##
##      name                description
## 1  "$eig"              "eigenvalues"
## 2  "$var"               "results for the variables"
## 3  "$var$coord"        "coord. for the variables"
## 4  "$var$cor"           "correlations variables - dimensions"
## 5  "$var$cos2"          "cos2 for the variables"
## 6  "$var$contrib"       "contributions of the variables"
## 7  "$ind"               "results for the individuals"
## 8  "$ind$coord"         "coord. for the individuals"
## 9  "$ind$cos2"          "cos2 for the individuals"
## 10 "$ind$contrib"       "contributions of the individuals"
## 11 "$call"              "summary statistics"
## 12 "$call$centre"        "mean of the variables"
## 13 "$call$ecart.type"   "standard error of the variables"
## 14 "$call$row.w"         "weights for the individuals"
## 15 "$call$col.w"         "weights for the variables"
```

4.1.3.1 Análisis de los resultados

El ACP calcula los autovalores (eigenvalues) y autovectores propios a partir de la matriz de covarianzas. El cálculo de los vectores depende de la cantidad de dimensiones de los datos. Los autovalores no son más que la magnitud de los autovectores, ambos ayudan a calcular los Componentes principales.

```
eigenvalues <- res.pca$eig# eigenvalues
head(eigenvalues[, 1:3])
```

	eigenvalue	percentage of variance	cumulative percentage of variance
## comp 1	4.1797364	32.151818	32.15182
## comp 2	3.5439409	27.261084	59.41290
## comp 3	1.3449952	10.346117	69.75902
## comp 4	1.0490855	8.069888	77.82891
## comp 5	0.9618192	7.398609	85.22752
## comp 6	0.8132415	6.255704	91.48322

En la primer columna se observa el valor del eigenvalue, en la segunda columna se observa el porcentaje de variación explicada. En la tercera columna se muestra el porcentaje de variación acumulada.

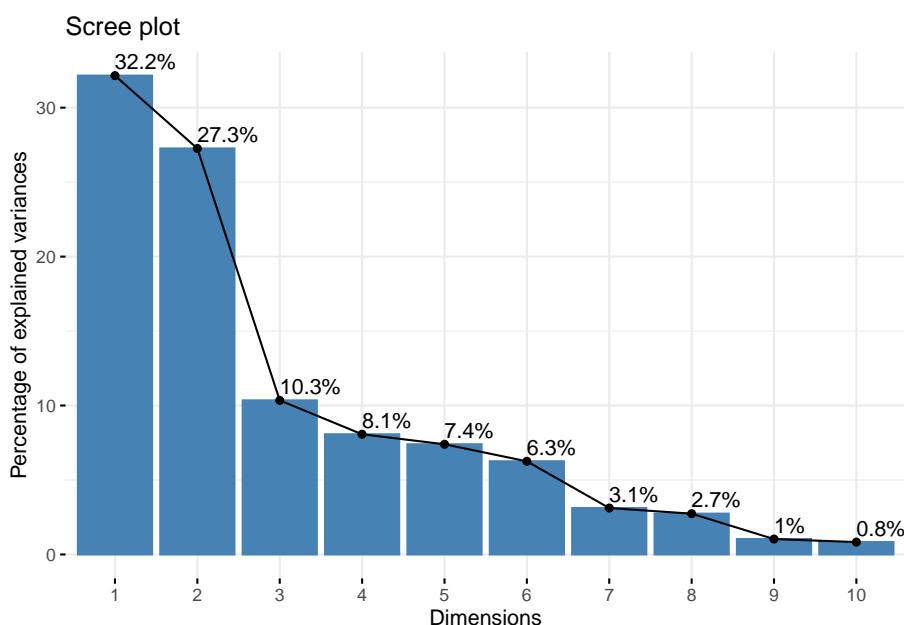
Un paquete útil para mejorar los gráficos de los análisis multivariados es **factoextra**.

```
## Warning: package 'factoextra' was built under R version 4.2.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```



Los **eigenvalues** se utilizan para determinar el número de componentes que deben conservarse. Existen dos maneras frecuentes de analizar estos eigenvalues, por un lado muchos investigadores utilizan los eigenvalues > 1 . Otra manera de determinar el número de componentes es por la cantidad de variación explicada, muchos usan $> 70\%$ de la variabilidad de los datos.

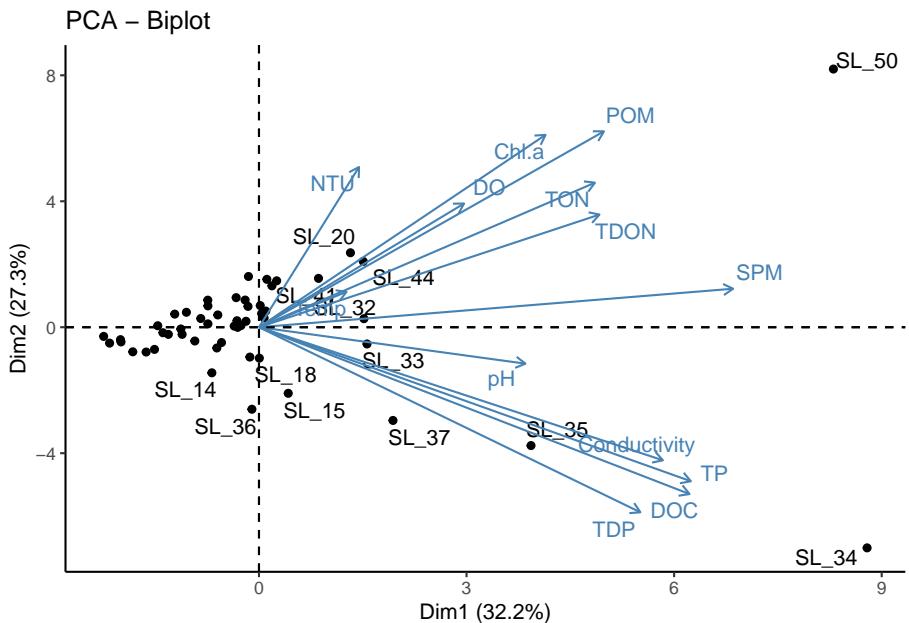
En este ejemplo, los primeros dos planos solo explican un 59.5% de la variabilidad de los datos y al usar un 3 eje se explica 69.8%. Decido usar 3 ejes para describir estos datos.

4.1.3.2 Visualizar los datos

Ahora vamos a visualizar los tres ejes. Como habíamos dicho previamente, el ACP genera gráficos de los individuos y de las variables. Actualmente es más frecuente usar gráficos en los que se observen ambos: los individuos y las variables. Este tipo de gráficos se denomina biplots.

```
?fviz_pca_biplot
fviz_pca_biplot(res.pca, repel=TRUE, invisible = "quali")+theme_classic()

## Warning: ggrepel: 39 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



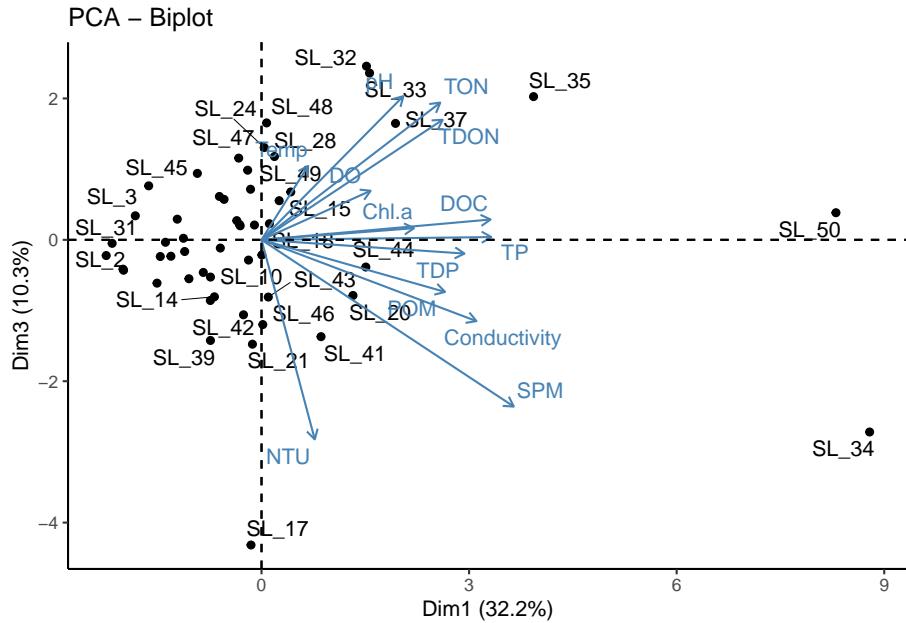
El gráfico muestra los ejes o componentes 1 y 2. A simple vista se observa que algunos individuos, como SL_34 por ejemplo es la muestra que tiene mas DOC (carbono orgánico disuelto), mientras que la muestra SL_50 es la que presentó la mayor cantidad de POM (materia orgánica particulada).

Interpretación: el primer componente (Dim1 32.%) divide a las muestras que tienen más DOC, más conductividad, más TP hacia la derecha del gráfico. El componente 2, parece que divide a las muestras de acuerdo a la POM, la clorofila a, la salinidad.

Ahora vamos a ver el componente 1 vs el componente 3.

```
fviz_pca_biplot(res.pca, axes=c(1,3), repel=TRUE, invisible = "quali")+theme_classic()
```

```
## Warning: ggrepel: 24 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



Una función importante del paquete FactoMiner, es el comando dimdesc. El mismo se utiliza para identificar las variables más significativamente asociadas a un componente.

```
dimdesc(res.pca, axes = c(1:3), proba = 0.05)
```

```
## $Dim.1
##
## Link between the variable and the continuous variables (R-square)
## =====
##          correlation      p.value
## SPM        0.8005755 1.052112e-12
## TP         0.7292919 8.700526e-10
## DOC        0.7269277 1.048135e-09
## Conductivity 0.6820821 2.578475e-08
## TDP        0.6440372 2.589279e-07
## POM        0.5824318 5.906561e-06
## TDON       0.5749478 8.275646e-06
```

```

## TON          0.5672584 1.160335e-05
## Chl.a       0.4838790 2.787599e-04
## pH          0.4499121 8.187113e-04
## DO          0.3463183 1.190357e-02
##
## $Dim.2
##
## Link between the variable and the continuous variables (R-square)
## =====
##           correlation      p.value
## POM          0.7266597 1.070368e-09
## Chl.a        0.7136585 2.879319e-09
## NTU          0.5945254 3.363780e-06
## TON          0.5361705 4.189718e-05
## DO           0.4590453 6.194896e-04
## TDON         0.4178871 2.052244e-03
## Conductivity -0.4936215 2.003631e-04
## TP           -0.5713221 9.715629e-06
## DOC          -0.6187953 1.011371e-06
## TDP          -0.6865882 1.917673e-08
##
## $Dim.3
##
## Link between the variable and the continuous variables (R-square)
## =====
##           correlation      p.value
## pH           0.4460824 9.181604e-04
## TON          0.4261880 1.631046e-03
## TDON         0.3729687 6.466085e-03
## SPM          -0.5178939 8.417748e-05
## NTU          -0.6197587 9.622302e-07

```

Estos resultados nos permiten asociar variables a los distintos ejes/componentes. Se puede inclusive establecer un criterio ya que usa coeficientes de correlación. Para este ejemplo, se puede usar las variables que tienen una correlación >0.6 con el eje. Para el componente principal 1, que explica (32.2%), las variables asociadas a este componente son DOC, TP, TDP, conductivity y SPM. Para el componente principal 2 (27.3%), está asociado a POM, clorofila a y NTU (salinidad). Finalmente, el componente principal 3 (10.3%) estaría asociado de manera negativa solo a la salinidad (NTU). El signo de la correlación indica la dirección de la correlación.

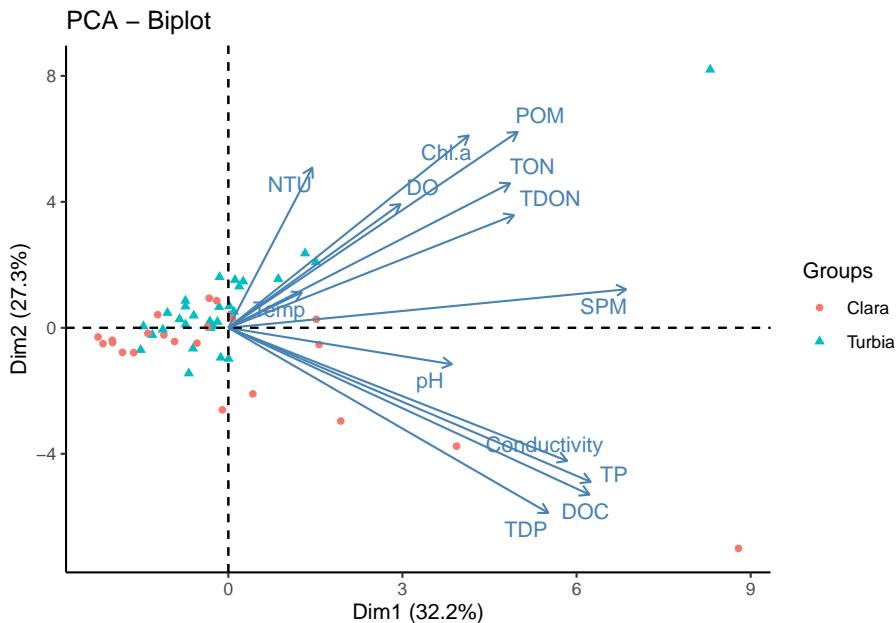
4.1.4 Crear una condición para colorear los individuos

A veces es útil crear alguna condición (variable categórica) para colorear los individuos y quizás empezar a ver patrones más claros. En este voy a usar la variable “secchi” para colorear a los individuos. Si la variable secchi es menor o igual a 20 entonces voy a llamar a la laguna como “turbia” mientras que si es mayor a 20, la voy a denominar laguna “clara”.

```
datos$condicion <- as.factor(ifelse(datos$Secchi<=20, "Turbia", "Clara"))
```

Cree una nueva variable llamada condicion

```
fviz_pca_biplot(res.pca, repel=TRUE, invisible = "quali", habillage = datos$condicion, geom = (
```



4.1.5 Refinando el modelo

El APC explica aproximadamente un 69% usando tres ejes. En el próximo paso vamos a seleccionar solo las variables disueltas: DOC, pH, TDON, DO, TDP, chla. y conductivity.

```
data.stan.sel <- datos.stand [,c(3:5,8,12:14)]
```

Vuelvo a realizar el análisis de ACP:

```
res.pca2 <- PCA(data.stan.sel, graph=FALSE)
```

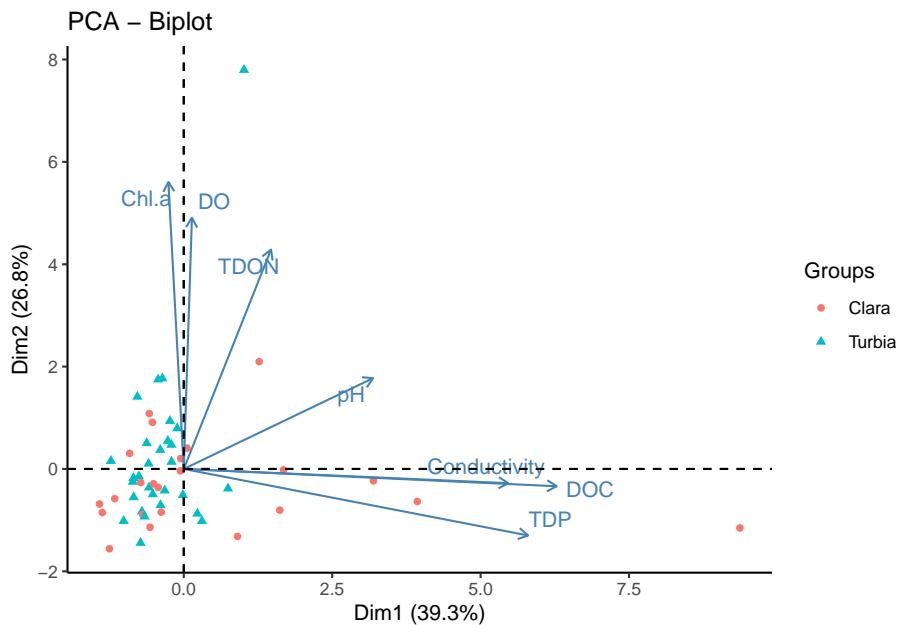
```
## Warning in PCA(data.stan.sel, graph = FALSE): Missing values are imputed by
## the mean of the variable: you should use the imputePCA function of the missMDA
## package
```

```
eigenvalues2 <- res.pca2$eig# eigenvalues
head(eigenvalues2[, 1:3])
```

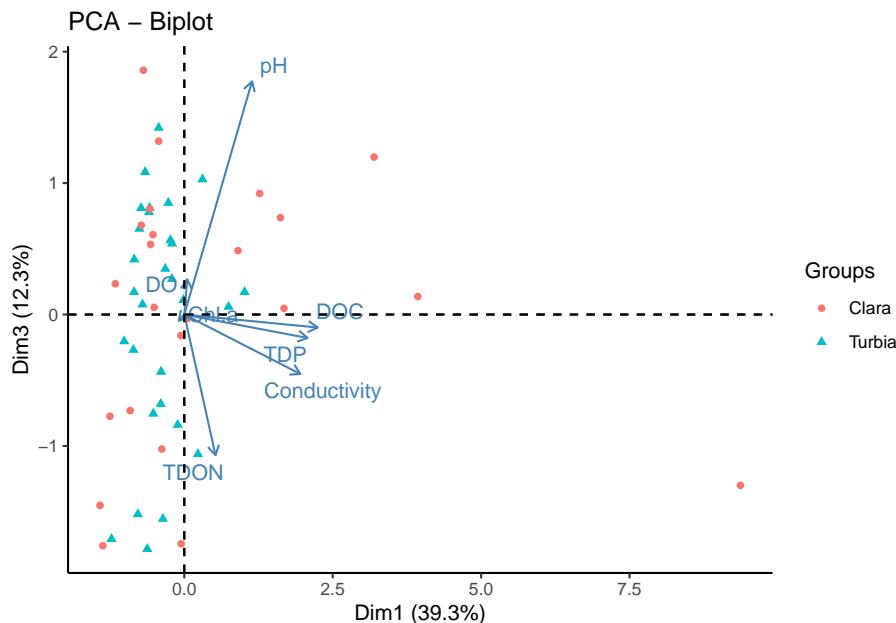
	eigenvalue	percentage of variance	cumulative percentage of variance
## comp 1	2.7477587	39.253696	39.25370
## comp 2	1.8792045	26.845778	66.09947
## comp 3	0.8578634	12.255191	78.35467
## comp 4	0.7497849	10.711212	89.06588
## comp 5	0.4279332	6.113331	95.17921
## comp 6	0.2531477	3.616395	98.79560

Biplot:

```
fviz_pca_biplot(res.pca2, repel=TRUE, invisible = "quali", habillage = datos$condicion)
```



```
fviz_pca_biplot(res.pca2, axes= c(1,3), repel=TRUE, invisible = "quali", habillage = datos$condi
```



El nuevo ACP con las variables seleccionadas, explica un 78% de la variabilidad total de los datos, teniendo en cuenta los primeros 3 ejes. Considero que este modelo refinado es mucho mejor que el anterior.

4.2 NMDS

El escalamiento multidimensional no métrico, mejor conocido por sus siglas en inglés (**Non-metric Multidimensional Scaling**), es un método de análisis estadístico multivariado que representa mediciones de *similaridad* (o disimilaridad) entre pares de objetos como distancias entre puntos de un espacio de dimensión reducida. El objetivo fundamental del **NMDS** es generar una representación gráfica de los objetos en un espacio de modo que sus posiciones relativas sean el reflejo de su proximidad. A diferencia de otros métodos de escalamientos, el NMDS utiliza ordenes de rango, por lo que es una técnica extremadamente flexible que puede adaptarse a una gran variedad de datos.

```
## Warning: package 'vegan' was built under R version 4.2.3
## Loading required package: permute
```

```
## Loading required package: lattice

## This is vegan 2.6-4
```

Vamos a usar las variables seleccionadas en el segundo ACP, es decir las variables más relacionadas con la fracción disuelta.

```
disueltos <- datos[,c(6:8,11,15:17)]
summary(disueltos)
```

```
##          DO              pH      Conductivity        TDON
##  Min.   : 5.000   Min.   :8.000   Min.   : 0.320   Min.   :1926
##  1st Qu.: 8.725   1st Qu.:8.557   1st Qu.: 1.215   1st Qu.:3052
##  Median :10.300   Median :8.795   Median : 2.615   Median :3478
##  Mean   :10.238   Mean   :8.762   Mean   : 7.968   Mean   :3790
##  3rd Qu.:11.050   3rd Qu.:8.982   3rd Qu.: 5.513   3rd Qu.:4356
##  Max.   :20.000   Max.   :9.400   Max.   :202.100  Max.   :7235
##
##          TDP             Chl.a        DOC
##  Min.   : 0.1   Min.   : 1.58   Min.   : 1.26
##  1st Qu.:114.0  1st Qu.:15.80   1st Qu.: 12.36
##  Median :302.0  Median :52.05   Median : 19.45
##  Mean   :560.3   Mean   :87.80   Mean   : 93.22
##  3rd Qu.:624.0  3rd Qu.:89.13   3rd Qu.: 97.58
##  Max.   :4140.0  Max.   :981.06  Max.   :1010.00
##                  NA's   :1
```

Algunos análisis exploratorios multivariados son sensibles a los datos faltantes. Como se puede observar, la variable DOC tiene un dato faltante. En este caso, lo mejor es sacar esa fila del análisis.

```
disueltos <- disueltos[-17,]
condicion <- data.frame(datos$condicion)
condicion <- condicion [-c(17),]## hay que sacar la misma fila de la variable condicion
```

Ahora se estandarizan los datos y luego se los convierte a distancias euclidianas para luego realizar el análisis.

```
set.seed(2306) # generar resultados reproducibles
dis.stan <- decostand(disueltos, "stand") # estandarizar: variables ambientales en diferentes escenarios
dis.dist <- vegdist(dis.stan, "euc") # distancia euclidiana
res.nmds <- metaMDS(dis.dist, trymax = 500) # NMDS
```

```
## Run 0 stress 0.08172658
## Run 1 stress 0.08190544
## ... Procrustes: rmse 0.01185868 max resid 0.06039221
## Run 2 stress 0.08129024
## ... New best solution
## ... Procrustes: rmse 0.01390953 max resid 0.05964353
## Run 3 stress 0.08801438
## Run 4 stress 0.08189141
## Run 5 stress 0.0846288
## Run 6 stress 0.08169404
## ... Procrustes: rmse 0.02381188 max resid 0.116934
## Run 7 stress 0.08336852
## Run 8 stress 0.08169061
## ... Procrustes: rmse 0.0190035 max resid 0.08992999
## Run 9 stress 0.09152335
## Run 10 stress 0.08174482
## ... Procrustes: rmse 0.01178546 max resid 0.04261408
## Run 11 stress 0.08704684
## Run 12 stress 0.08073759
## ... New best solution
## ... Procrustes: rmse 0.007340069 max resid 0.03921827
## Run 13 stress 0.08074268
## ... Procrustes: rmse 0.02379549 max resid 0.1167022
## Run 14 stress 0.08214195
## Run 15 stress 0.0819048
## Run 16 stress 0.08173675
## Run 17 stress 0.08722591
## Run 18 stress 0.08190564
## Run 19 stress 0.0877217
## Run 20 stress 0.08117769
## ... Procrustes: rmse 0.008500534 max resid 0.04099969
## Run 21 stress 0.08109837
## ... Procrustes: rmse 0.04191297 max resid 0.2040888
## Run 22 stress 0.08142552
## Run 23 stress 0.08713932
## Run 24 stress 0.08215738
## Run 25 stress 0.08117084
## ... Procrustes: rmse 0.01652584 max resid 0.07695317
## Run 26 stress 0.08088512
## ... Procrustes: rmse 0.009314285 max resid 0.04562758
## Run 27 stress 0.08161162
## Run 28 stress 0.08276532
## Run 29 stress 0.08138922
## Run 30 stress 0.08708527
## Run 31 stress 0.0814658
## Run 32 stress 0.08666174
```

```

## Run 33 stress 0.08186204
## Run 34 stress 0.08801345
## Run 35 stress 0.08087082
## ... Procrustes: rmse 0.008549763 max resid 0.04176883
## Run 36 stress 0.08143206
## Run 37 stress 0.08068018
## ... New best solution
## ... Procrustes: rmse 0.007515726 max resid 0.03743025
## Run 38 stress 0.08102709
## ... Procrustes: rmse 0.03204386 max resid 0.1556518
## Run 39 stress 0.08473582
## Run 40 stress 0.09152295
## Run 41 stress 0.08068146
## ... Procrustes: rmse 0.00815311 max resid 0.04025691
## Run 42 stress 0.09152335
## Run 43 stress 0.08245253
## Run 44 stress 0.08143243
## Run 45 stress 0.08718439
## Run 46 stress 0.08137234
## Run 47 stress 0.08720687
## Run 48 stress 0.08085884
## ... Procrustes: rmse 0.01546736 max resid 0.07652651
## Run 49 stress 0.08111344
## ... Procrustes: rmse 0.03515309 max resid 0.1708126
## Run 50 stress 0.08179221
## Run 51 stress 0.08169834
## Run 52 stress 0.08675948
## Run 53 stress 0.08797076
## Run 54 stress 0.09155446
## Run 55 stress 0.08774277
## Run 56 stress 0.08068016
## ... New best solution
## ... Procrustes: rmse 0.00136907 max resid 0.008143052
## ... Similar to previous best
## *** Best solution repeated 1 times

```

Una regla general: si el stress < 0,05 proporciona una excelente representación en dimensiones reducidas, < 0,1 es genial, < 0,2 es bueno/ok, y un stress < 0,3 proporciona una mala representación. **Para recordar:** jun stress alta es malo, un *stress bajo es bueno!*

```
res.nmds
```

```

## 
## Call:

```

```

## metaMDS(comm = dis.dist, trymax = 500)
##
## global Multidimensional Scaling using monoMDS
##
## Data:      dis.dist
## Distance: euclidean
##
## Dimensions: 2
## Stress:      0.08068016
## Stress type 1, weak ties
## Best solution was repeated 1 time in 56 tries
## The best solution was from try 56 (random start)
## Scaling: centring, PC rotation
## Species: scores missing

```

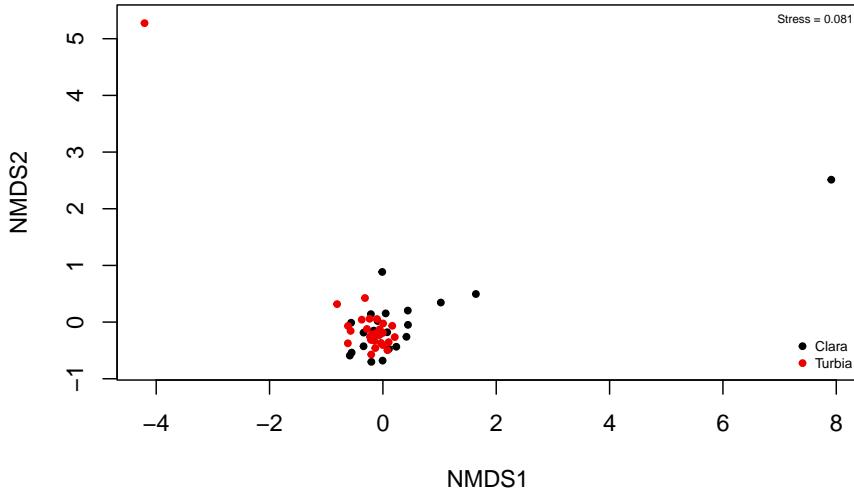
El stress es de 0.08 que es menor que 0.1 con lo cual la representación está bastante bien. Ahora vamos a graficar estos resultados:

```

{plot(res.nmds, type="n")
points(res.nmds, display = "site", cex=0.6, select=which(condicion=="Clara"), pch = 19, col="black")
points(res.nmds, display = "site", cex=0.6, select=which(condicion=="Turbia"), pch = 19, col="red2")
legend("topright", cex=0.5, box.col=NA,legend=paste("Stress =",round(res.nmds$stress, 3)))
legend("bottomright", cex=0.6, box.col=NA,
      legend=c("Clara", "Turbia"), pch=19, col=c("black","red2"))}

## species scores not available

```



El escalamiento es parecido al resultado del ACP, en donde se pueden observar que hay un solapamiento importante en el tipo de lagunas.

4.3 PERMANOVA

El análisis multivariante de permutaciones de la varianza o PerMANOVA, es una alternativa no paramétrica a la prueba de ANOVA multivariada. Es apropiado con conjuntos de múltiples variables que no cumplen los supuestos, por ejemplo, el de normalidad. Además, se puede utilizar para datos muy sesgados, ordinales o cualitativos, y en datos de comunidades ecológicas, datos de comunidades microbiana o en datos genéticos. Su funcionamiento incluye una matriz de distancias construida a partir de cualquier medida de disimilitud. Este análisis se utiliza para comparar grupos de objetos y probar con la hipótesis nula de que los centroides y la dispersión de los grupos son equivalentes. Este análisis suele acompañar a los gráficos de ordenamiento tales como el NMDS.

```
res.perma <- adonis2(dis.dist~condicion, method = "euclidean", permutations = 599)
res.perma
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
```

```
## Number of permutations: 599
##
## adonis2(formula = dis.dist ~ condicion, permutations = 599, method = "euclidean")
##          Df SumOfSqs      R2    F Pr(>F)
## condicion  1    14.03 0.04009 2.0462   0.06 .
## Residual   49   335.97 0.95991
## Total     50   350.00 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Para comparar los distintos grupos se pueden hacer análisis a posteriori, de comparaciones entre los grupos.

```
library(pairwiseAdonis)
```

```
## Loading required package: cluster
```

```
res.pos<- pairwise.adonis(dis.dist, condicion, sim.method = "euclidean", p.adjust.m = "bonferroni")
res.pos
```

```
##           pairs Df SumsOfSqs  F.Model          R2 p.value p.adjusted sig
## 1 Clara vs Turbia  1  14.03004 2.046231 0.04008584  0.048     0.048   .
```


Chapter 5

Análisis multivariados II

TECNICAS DE ORDENACION CANONICA

Maria Eugenia del R. Llames¹

Instituto Tecnológico de Chascomús (INTECH, UNSAM-CONICET), Escuela de Bio y Nanotecnologías (UNSAM)

5.1 INTRODUCCIÓN

El supuesto principal de cualquier técnica de ordenación es que los datos analizados son redundantes, es decir, contienen más variables (y dimensiones) de las necesarias para describir la información subyacente, y podemos reducir el número de estas dimensiones sin perder demasiada información. Por ejemplo, en el caso de los datos de composición de especies, algunas de las especies suelen ser ecológicamente similares (por ejemplo, especies que prefieren crecer en un hábitat húmedo en lugar de seco), lo que significa que el conjunto de datos contiene varias variables redundantes (especies) que cuentan la misma historia. O, para explicar la redundancia de otra manera, a partir de la presencia de una especie, a menudo podemos predecir la presencia de varias otras especies. En el caso de la ordenación aplicada en la matriz de variables ambientales, estas a menudo se correlacionan entre sí (por ejemplo, las mediciones del temperatura del agua a menudo se relacionan con las concentraciones de oxígeno disuelto), lo que también permite la reducción de la dimensión.

Dado que el espacio multidimensional no es fácil de mostrar, describir o simplemente imaginar, vale la pena reducirlo a unas

¹mariaellames@intech.gov.ar

pocas dimensiones principales, conservando al máximo la información. Esto también significa que si las variables individuales son completamente independientes entre sí (por ejemplo, cada especie tiene preferencias completamente diferentes), entonces es probable que la ordenación no encuentre una reducción razonable del espacio multidimensional.

Lo que hace el método de ordenación se puede formular de dos maneras alternativas:(i) busca gradientes en la composición de especies (representados generalmente por ejes de ordenación) e intenta explicar estos gradientes por variables ambientales; y/o (ii) busca la distribución de muestras en un espacio de ordenación reducido que refleje al máximo la disimilitud (= distancia) entre muestras en términos de su composición de especies.

De acuerdo al apartado anterior (**Unidad 4**), las técnicas allí presentadas permiten el análisis de una única matriz de datos de forma tal de revelar su estructura a través de un grafo construido con un conjunto reducido de ejes ortogonales (*i.e* independientes) Las variables externas que pueden estar influenciando esta estructura sólo podrán ser consideradas después del cálculo de la ordenación. De acuerdo con esta característica, se las conocen como “técnicas de ordenación indirecta” en donde uno deja que la matriz de datos exprese las relaciones entre objetos y variables sin restricción (“*unconstrained analyses*”). Por lo tanto es una forma pasiva de análisis, y el usuario interpreta los resultados de la ordenación *a posteriori*.

Las técnicas de **ordenación canónica**, por el contrario, permiten asociar dos o más conjuntos de datos en el propio proceso de ordenación. Típicamente, en ecología, estas matrices la constituyen una matriz de datos biológicos, como por ejemplo, relevamiento de especies en diferentes sitios (matriz de “respuesta”) y su matriz ambiental asociada (matriz “explicativa”) (Figura 1).

Figura 1: Esquema de las matrices utilizadas comúnmente en estudios ecológicos

En este contexto, las técnicas de ordenación canónicas se engloban dentro de las “técnicas de ordenación directa” y permiten analizar patrones entre un conjunto de datos que están relacionadas con (o pueden ser interpretadas por) otro conjunto de datos, y/o probar formalmente hipótesis estadísticas sobre la importancia de estas relaciones. En otras palabras, las técnicas de ordenación canónica exploran explícitamente las relaciones entre dos matrices: una matriz de