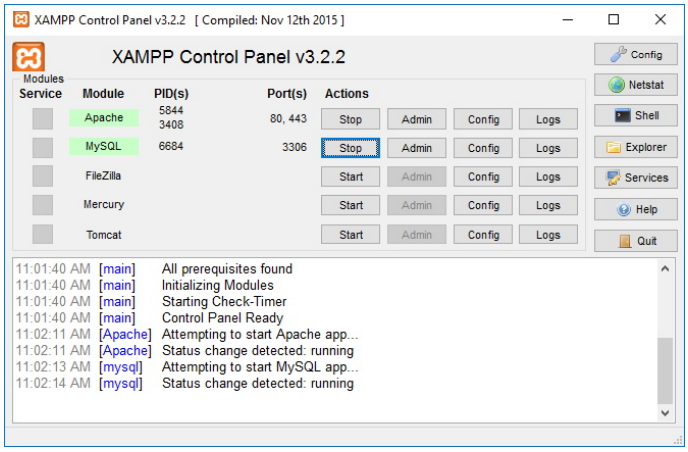
Acceso a bases de datos

En este documento daremos unas nociones básicas de cómo acceder a diferentes sistemas gestores de bases de datos desde R. En concreto nos centraremos en un ejemplo de base de datos relacional como MySQL, y en otro de base de datos No-SQL, como MongoDB. Como ocurre en muchos otros casos, necesitaremos instalar la librería adecuada para poder trabajar con la base de datos en cuestión, y después utilizar los métodos correspondientes para conectar, insertar, listar, etc.

1. Acceso a bases de datos MySQL

1.1. Instalación del servidor MySQL

Para poder trabajar con bases de datos MySQL, el primer paso será tener instalado un servidor MySQL. Podemos instalar uno de forma manual, y luego configurar los usuarios, bases de datos y tablas desde el terminal, o bien podemos disponer de un entorno algo más visual para poder gestionar el servidor y la base de datos de forma más cómoda. En nuestro caso, ya que no se trata de un curso para administrar servidores de bases de datos, optaremos por esta segunda opción a través de la herramienta XAMPP. Al instalarla dispondremos de un servidor web Apache, y un servidor MySQL. Podremos poner ambos en marcha desde la herramienta *XAMPP Manager* que viene incorporada, y poner en marcha ambos servidores.



La ventaja que tiene utilizar esta herramienta es que también disponemos de una aplicación web llamada *phpMyAdmin*, que se pone en marcha sobre Apache y nos permite gestionar las bases de datos que tengamos instaladas: crearlas, definir las tablas, hacer copia de seguridad, etc.

1.2. Instalación de las librerías de R

Para poder acceder a MySQL desde R necesitamos utilizar alguna librería externa que nos facilite los mecanismos de conexión y gestión de esa base de datos. Una de las más utilizadas es `RMySQL`, que instalamos con el correspondiente comando desde el terminal R:

```
install.packages("RMySQL")
```

Después importaremos la librería en nuestro programa:

```
library(RMySQL)
```

1.3. Operaciones con la base de datos

Veremos a continuación un ejemplo sencillo de cada una de las operaciones habituales con MySQL.

1.3.1. Conexión a una base de datos

La primera operación que tendremos que realizar será conectar con la base de datos que queramos. Supondremos que ya la tendremos previamente creada, a través de *phpMyAdmin* alguna otra herramienta. En cualquier caso, la propia librería *RMySQL* dispone de mecanismos para crear bases de datos y tablas, pero no los veremos aquí.

La instrucción para conectar es como sigue:

```
library(RMySQL)

bd = dbConnect(RMySQL::MySQL(),
    dbname="nombreBD",
    host="localhost",
    port=3306,
    user="usuario",
    password="password"
)
```

> **NOTA**: en la instrucción anterior deberemos reemplazar los valores de los parámetros por los reales. Lo normal es que conectemos desde el mismo ordenador donde está la base de datos, con lo que el *host* suele ser *localhost*, y el puerto por defecto para MySQL suele ser el 3306. Pero el usuario y contraseña pueden cambiar. La instalación por defecto de XAMPP deja un usuario *root* con contraseña vacía. Finalmente, el parámetro *dbname* apuntará al nombre de la base de datos que queramos utilizar.

1.3.2. Operaciones de inserción, borrado y actualización

Para realizar operaciones INSERT, DELETE o UPDATE, a partir del objeto obtenido al conectar, crearemos un *cursor* que nos permita acceder a la base de datos, y usaremos su instrucción `execute` para ejecutar la instrucción correspondiente. Por ejemplo, si tenemos una tabla *personas* con unos campos *nombre* y *edad*, podemos insertar una nueva persona así:

```
# ... Código anterior para conectar

sql = "INSERT INTO personas (nombre, edad) VALUES('Nacho', 45)"
resultado = dbSendQuery(bd, sql)
```

En el caso de que queramos pasar datos variables los podemos concatenar en el texto usando `paste`:

```
# ... Código anterior para conectar

nombre = "Nacho"
edad = 45

sql = paste("INSERT INTO personas (nombre, edad) VALUES('", nombre, "',", edad, ")")
resultado = dbSendQuery(bd, sql)
```

También de forma similar podemos lanzar borrados o actualizaciones, y obtener el número de filas afectadas en el resultado. En este caso usaremos el comando `dbExecute` en lugar de `dbSendQuery`:

```
sql = "UPDATE personas SET edad=50 WHERE nombre = 'Nacho'"
filas = dbExecute(resultado)
cat("Filas afectadas:", filas)
```

1.3.3. Consultas

Para lanzar consultas (SELECT) construimos la instrucción SQL de forma similar, pero usamos el método `dbFetch` para obtener los resultados, en lugar de `dbExecute`. Por ejemplo, así obtenemos las personas mayores que la edad que diga el usuario previamente:

```
edad_usuario = # ... Le pedimos la edad al usuario

sql = paste("SELECT nombre, edad FROM personas WHERE edad > ", edad_usuario)

resultado = dbSendQuery(bd, sql)

# Obtener los resultados
resultados = dbFetch(resultado)

# Imprimir los resultados
if(nrow(resultados) > 0)
{
    for(i in 1:nrow(resultados))
    {
        cat("Nombre:", resultados[i, "nombre"], "\n")
    }
}
```

1.3.4. Cerrar la conexión

Tras finalizar las operaciones necesarias en la base de datos, conviene cerrar la conexión:

```
dbDisconnect(bd)
```

> **Ejercicio 1:**
>
> Descarga este backup de una base de datos MySQL llamada *libros* e impórtalo en tu servidor. Creará una base de datos *libros* con una tabla *libros* con una serie de registros insertados. Crea un programa `LibrosMySQL.r` y prueba a hacer una consulta que muestre los libros cuyo precio sea menor que el que diga el usuario.
>
> **AYUDA:** vídeo con solución del ejercicio

2. Acceso a bases de datos MongoDB

2.1. Instalación del servidor MongoDB

Como ocurría en el caso anterior, si queremos trabajar con bases de datos MongoDB necesitamos disponer de un servidor instalado. Nuevamente, al no ser éste el propósito principal del curso, aquí os dejamos algunas nociones de cómo instalarlo y qué herramienta(s) utilizar para gestionarlo.

- Instalación de MongoDB
- Herramientas para gestionar bases de datos MongoDB

2.2. Instalación de las librerías de R

La librería que usaremos en este caso para acceder a Mongo será `mongolite`, que podemos instalar con el correspondiente comando desde el terminal de R:

```
install.packages("mongolite")
```

La incluiremos en nuestro código fuente de este modo:

```
library(mongolite)
```

2.3. Operaciones con la base de datos

2.3.1. Establecer una conexión a la base de datos MongoDB

Para conectar a una base de datos MongoDB usaremos la instrucción `mongo`. Indicaremos en el parámetro `collection` la colección a la que queremos conectar, y en el parámetro `url` la URL de acceso a la base de datos:

```
conexion <- mongo(collection='personas',
    url='mongodb://localhost:27017/nombreBD')
```

En el caso de bases de datos MongoDB no es necesario crearlas explícitamente, se crearán en cuanto añadamos contenido a sus colecciones.

2.3.2. Insertar documentos

Para insertar documentos debemos usar el método `insert` a través de la conexión anterior. Indicaremos los elementos a insertar como un vector de objetos JSON:

```
conexion$insert(c("{'nombre': 'Juan', 'edad': 30}",
    "{'nombre': 'Pepe', 'edad': 45}"))
```

2.3.3. Consultar documentos

Para consultar documentos empleamos la instrucción `find` sobre la conexión, indicando la colección. Adicionalmente se le pueden pasar parámetros de filtrado.

```
# Consultar todos los documentos de una colección
resultado1 <- conexion$find()

# Filtrar resultados simples: por nombre
resultado2 <- conexion$find(query='{"nombre": "Juan"}')

# Filtrado más avanzado: por nombre y rango de edad
resultado3 <- conexion$find(query = '{"nombre": "Juan", "edad": {"$gte": 30}}')
```

Para recorrer los elementos encontrados, hay que tener en cuenta que lo que nos devuelve *find* es un *data frame*, una tabla con filas y columnas. Podemos recorrer cada fila con un bucle indexado (de 1 al número de filas encontradas), y cada columna con su nombre.

```
for (i in 1:length(resultado1))
{
    cat("Nombre:", resultado1[i, "nombre"], "\n")
}
```

2.3.4. Actualizar documentos

Para actualizar documentos usamos la instrucción `update` en la conexión. En el parámetro `query` indicamos el criterio de filtrado para quedarnos con los documentos que queremos actualizar, y en el parámetro `update` indicamos cómo queremos dejar sus valores. Adicionalmente tenemos un parámetro `multiple` para indicar si queremos actualizar múltiples documentos (TRUE) o sólo el primero que encontremos (FALSE, valor por defecto).

```
conexion$update(query = '{"nombre": "Juan"}', update = '{"$set": {"edad": 31}}', mult
```

2.3.5. Eliminar documentos

Para eliminar documentos indicamos en la instrucción `remove` el criterio de los elementos a eliminar.

```
conexion$remove('{"nombre": "Juan"}')
```

> **NOTA:** Si no se especifica el segundo parámetro, se eliminarán TODOS los documentos de la colección.

> **Ejercicio 2**:
>
> Crea un programa `ContactosMongo.r` que cree una base *contactos* en tu servidor MongoDB, con una colección llamada *contactos*. Inserta en ella estos dos contactos con sus campos:
>
> - *nombre*: Nacho, *edad* 45, *telefono* 611223344
> - *nombre*: Ana, *edad* 35, *telefono* 699887766
>
> Después prueba a buscar los contactos que tengan más de 40 años, para ver si muestra datos correctos.
>
> **AYUDA:** vídeo con solución del ejercicio