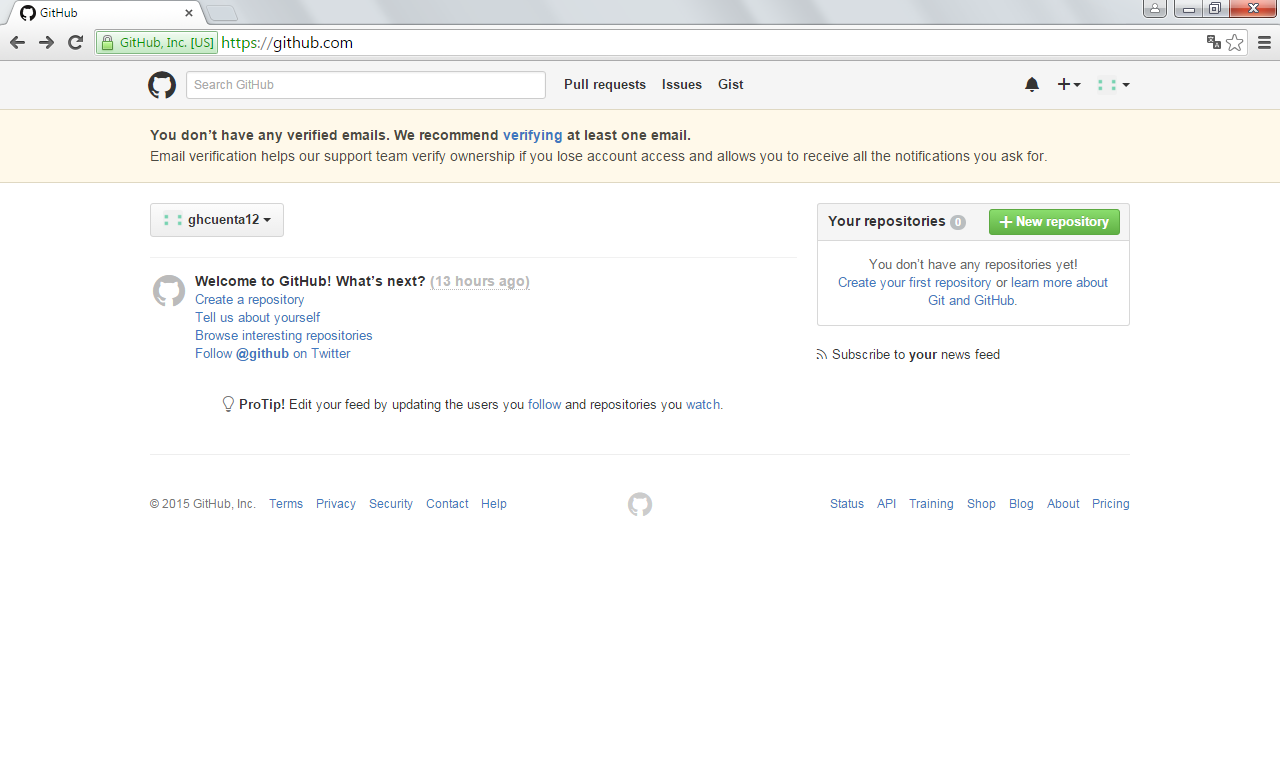
|  |  |
| --- | --- |
| UPIICSA.jpg | **INSTITUTO POLITÉCNICO NACIONAL**  **UNIDAD PROFESIONAL INTERDISCIPLINARIA**  **DE INGENIERÍA Y CIENCIAS SOCIALES**  **Y ADMINISTRATIVAS**  ***MAESTRÍA EN CIENCIAS EN INFORMÁTICA***  **MATERIA:** ALMACENAMIENTO DE DATOS Y SU ADMINISTRACIÓN  **PROFESOR:** M. EN C. EDUARDO RENÉ RODRÍGUEZ ÁVILA  **TAREA 5:** ANÁLISIS EXPLORATORIO  **ALUMNO:** CARLOS DANIEL MARTINEZ ORTIZ |

**ANÁLISIS EXPLORATORIO**

**PRIMERA PARTE**

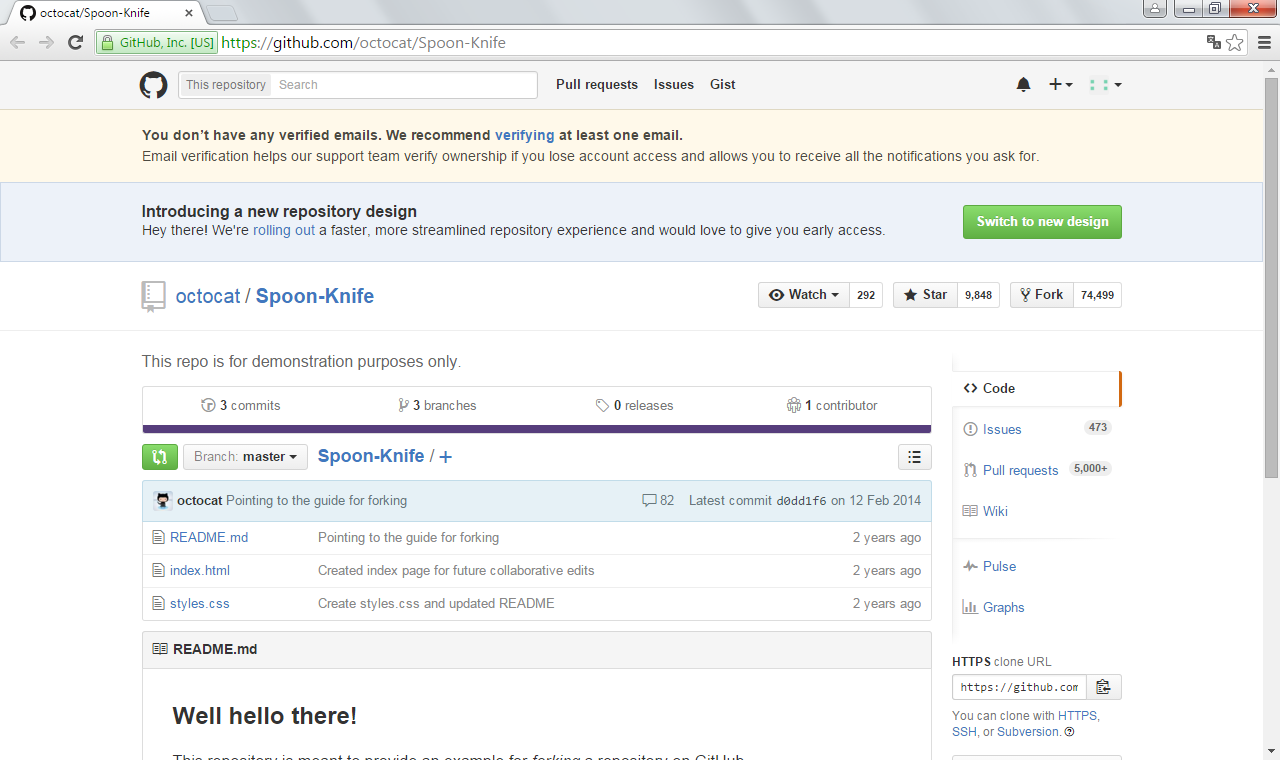
**1. Siga el ejemplo sugerido en la página del “GitHub Bootcamp” sobre creación de repositorios (*click* en la imagen inferior para ir al artículo; en la página de GitHub esta liga corresponde a la que lleva el tercer cuadro de esta imagen y que aparece en dicho sitio) y proceda con el “*fork*” del repositorio Spoon-Knife que en ésta se indica.**

Nos firmamos con nuestro usuario y password.



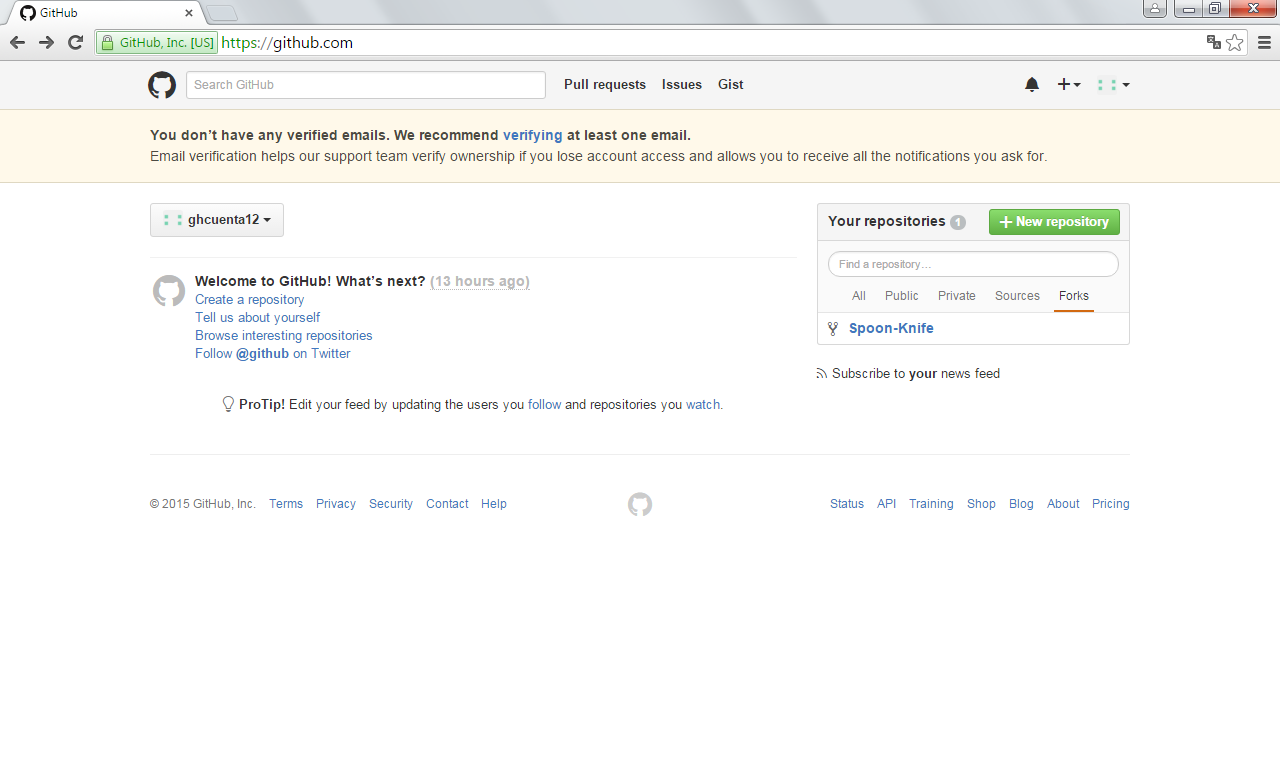
**2. Tome evidencia de la creación del repositorio bifurcado ya creado en su cuenta.**

Vamos al repositorio [**octocat/Spoon-Knife**](https://github.com/octocat/Spoon-Knife), una vez posicionados en el repositorio, damos “**Fork**” como lo vemos en la siguiente imagen.



Damos click en **Fork**

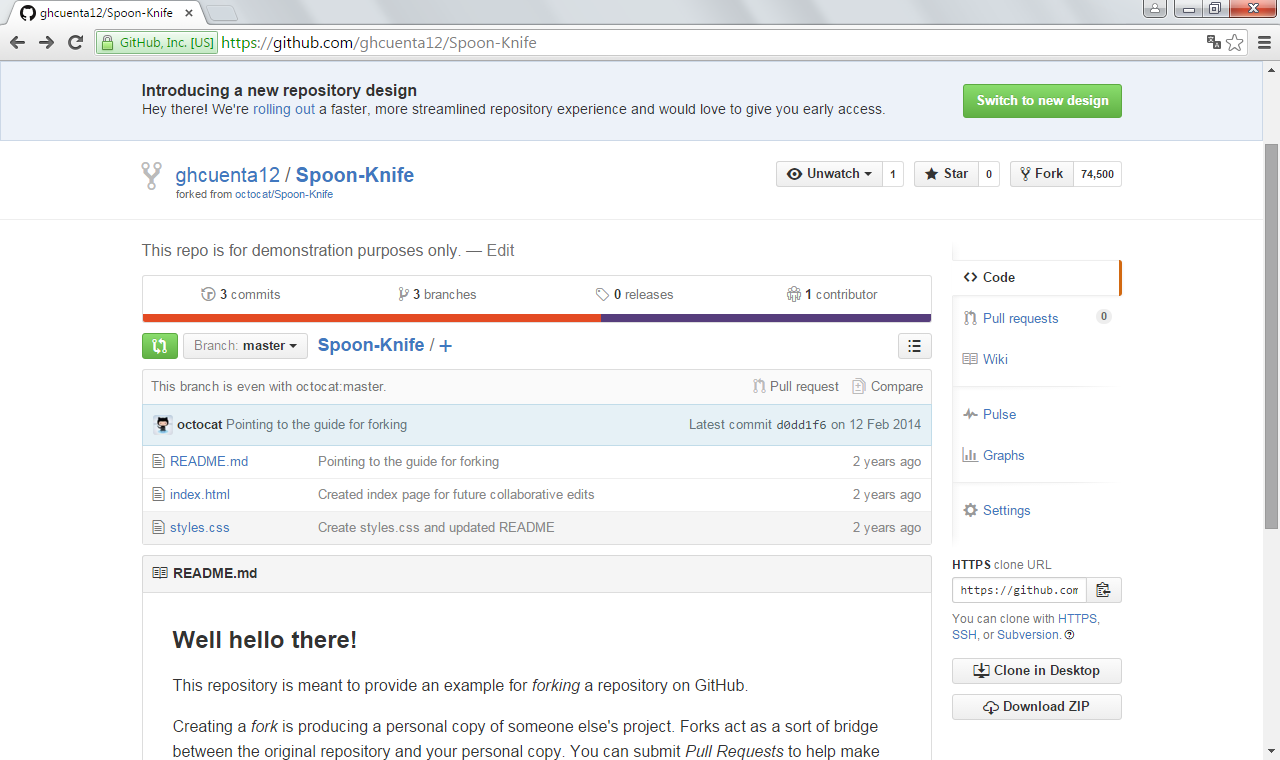
Vamos a la página principal de nuestra cuenta y ya vemos el repositorio **Spoon-Knife**.



Repositorio **Spoon-Knife**

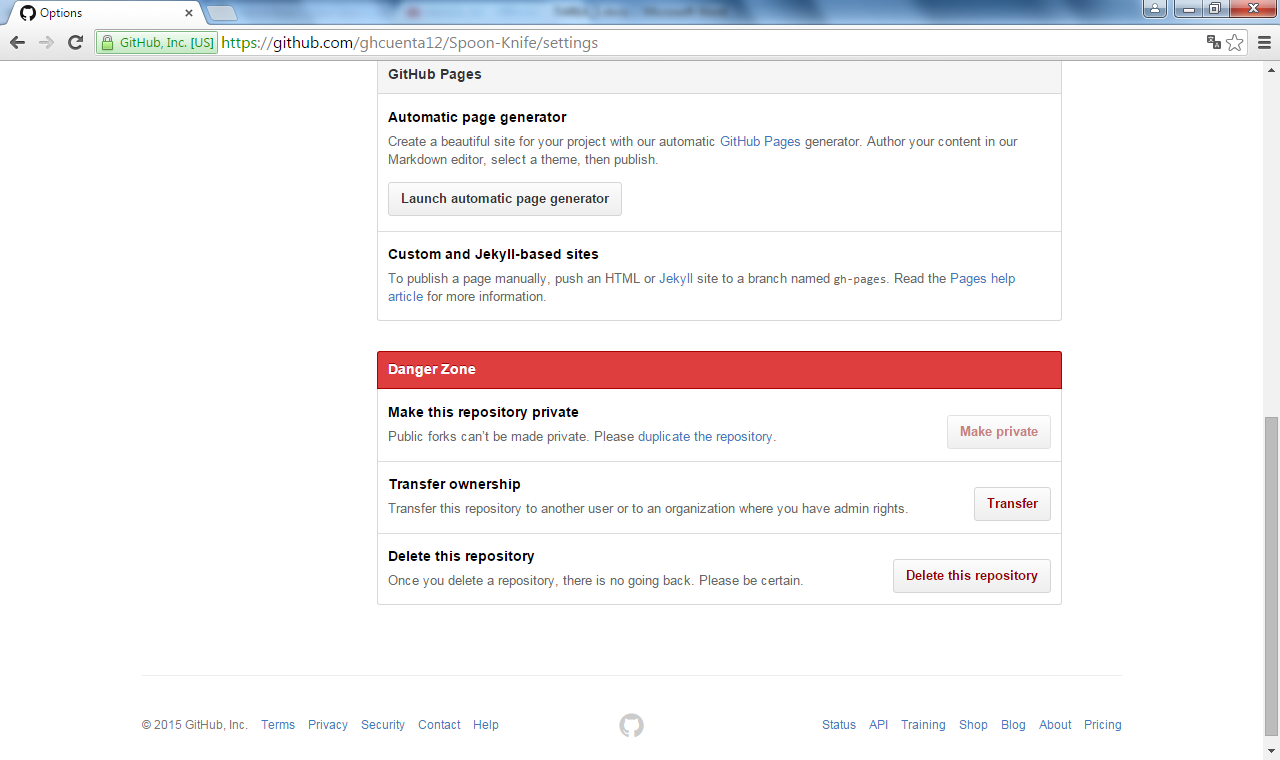
**3. Investigue como eliminarlo y proceda con su borrado.**

Nos posicionamos dentro de nuestra bifurcación del repositorio **Spoon-Knife** y damos click en “**Settings**”.



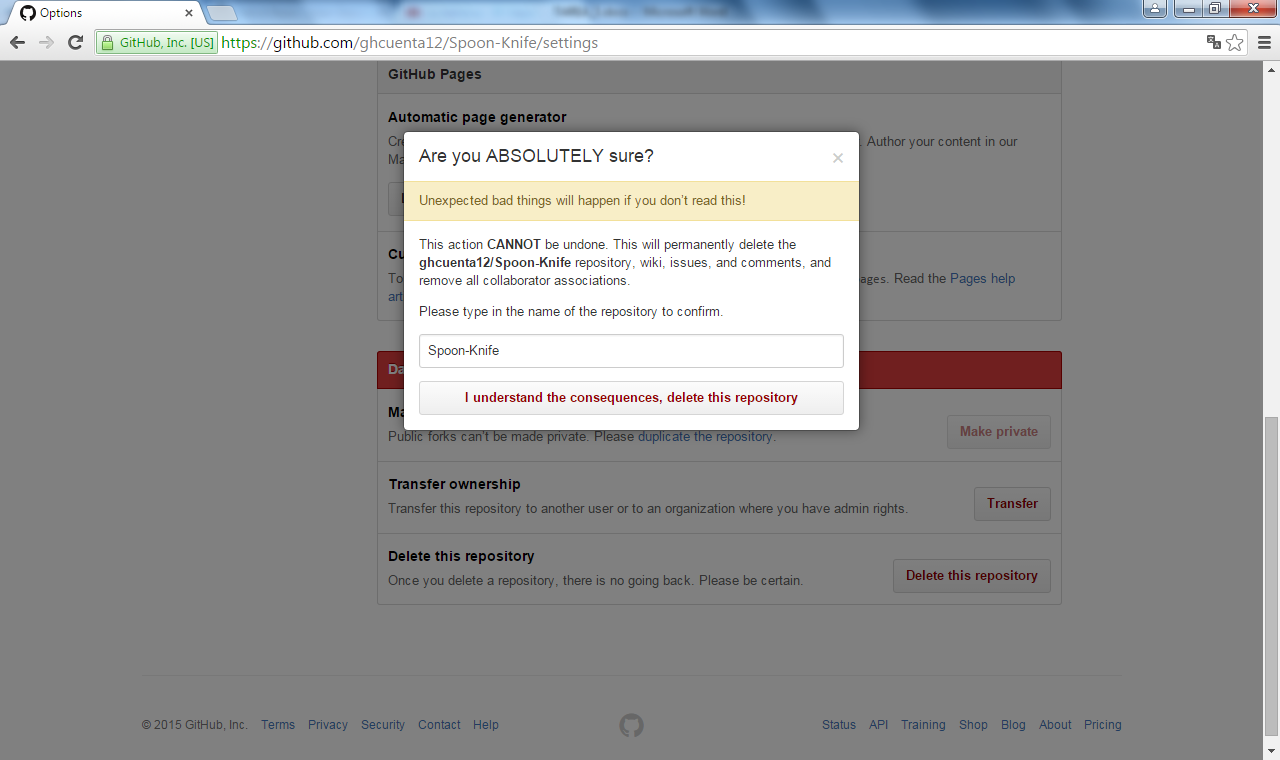
Click en **Settings**

Nos posicionamos al final de la página y damos click en “**Delete this repository**”.



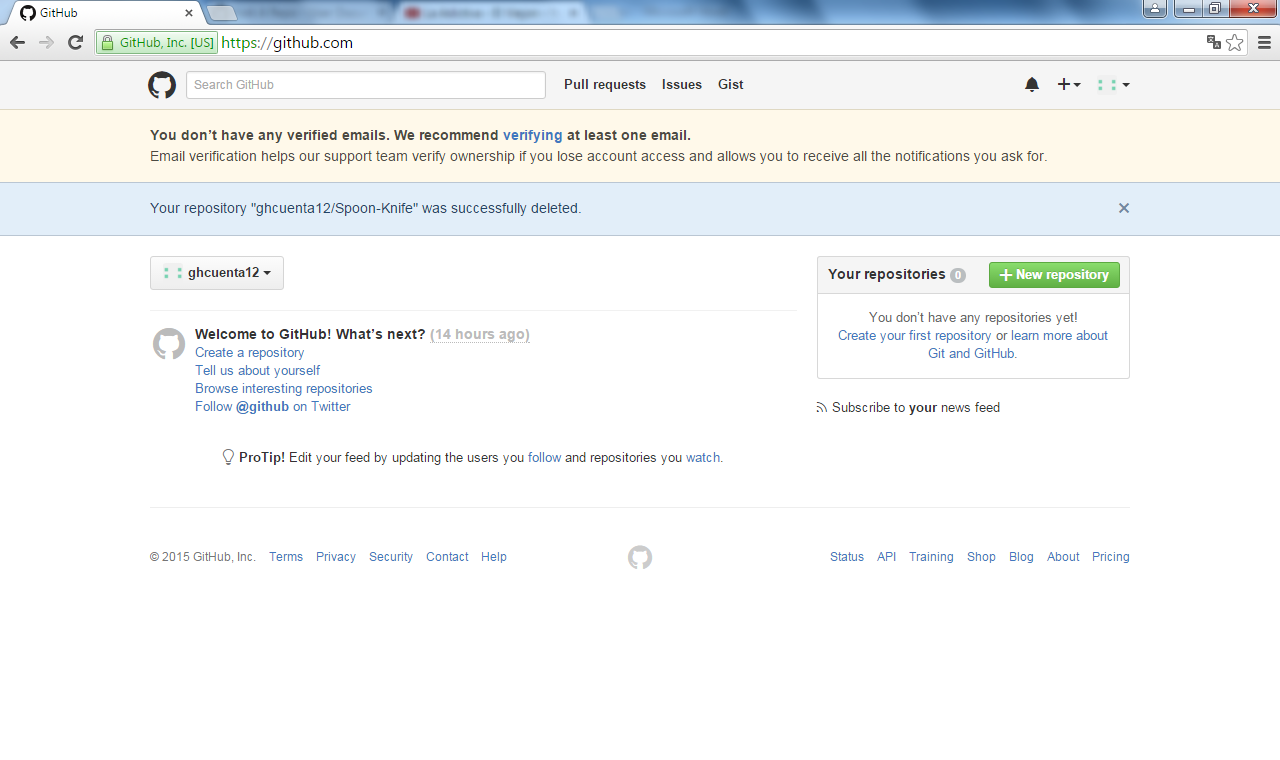
Para borrar dar Click

Tenemos que confirmar el nombre del repositorio para poder continuar a eliminarlo.



Escribimos el nombre

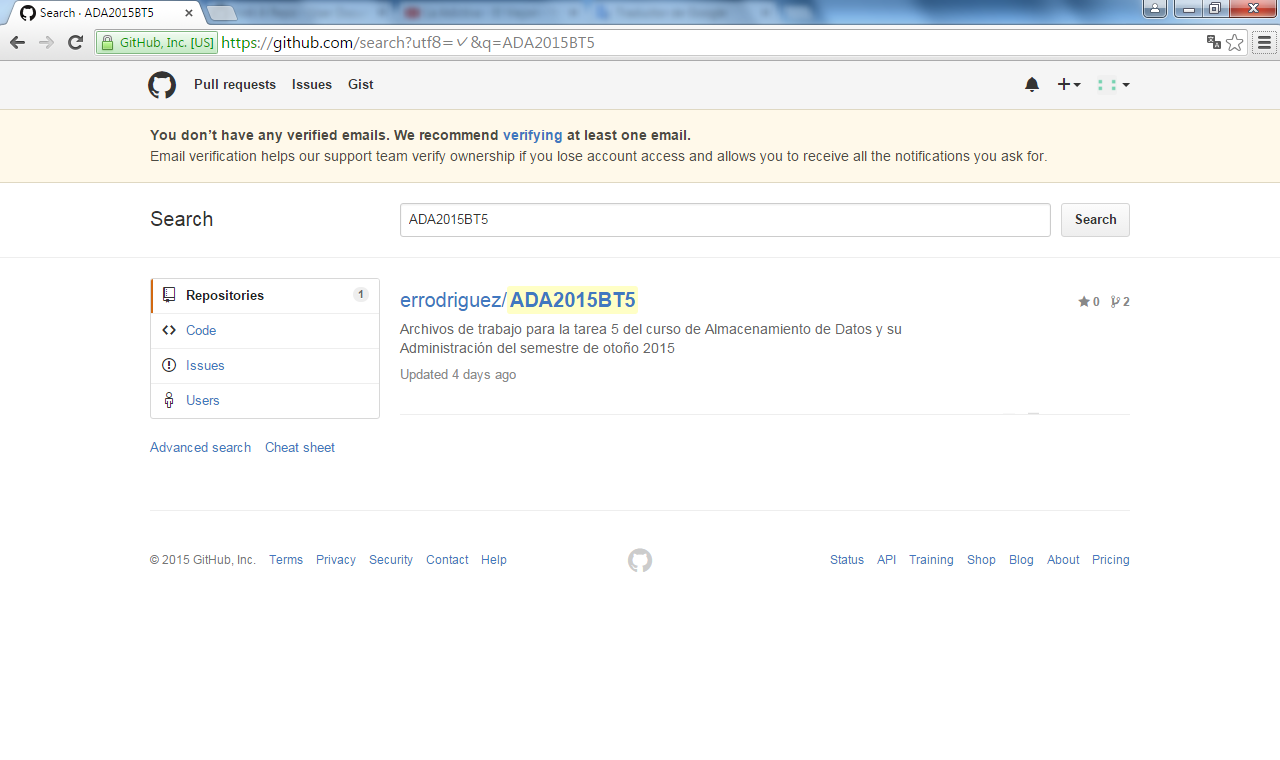
Vamos a la página principal de nuestra cuenta y ya vemos que ya no está repositorio **Spoon-Knife**.



**SEGUNDA PARTE**

**1. Busque el repositorio ADA2015BT5 de la cuenta errodriguez y bifúrquelo. Esto en el browser. Los siguientes pasos serán en una ventana de MSDOS o shell Unix, según sea el caso.**

Desde la página principal de nuestra cuenta buscamos el repositorio **ADA2015BT5** y entramos al repositorio.



Una vez dentro del repositorio **ADA2015BT5**, damos click en “**Fork**”. Una vez realizada ésta acción ya la bifurcamos en nuestra cuenta.



Damos click en **Fork**

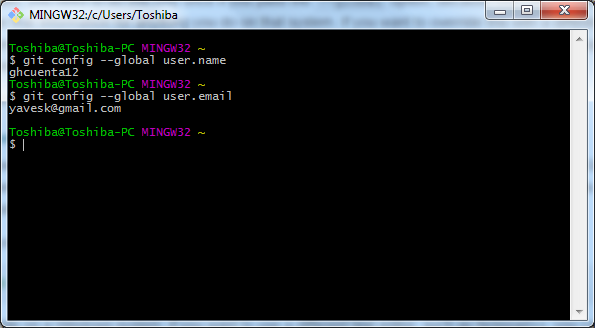
**2. Si Ud. ya ha configurado localmente a git en su equipo pase al siguiente punto, en otro caso continúe leyendo. Proceda con la configuración general, de primera vez (**[**first- time setup**](https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup)**). Muestre comandos y resultados.**

**IDENTIDAD**

Comandos para ver nuestro usuario e email.

**$** git config --global user.name

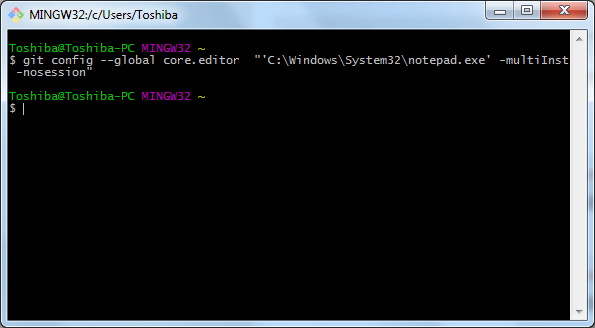
**$** git config --global user.email



**EDITOR**

Comandos para establecer nuestro editor predeterminado. En nuestro caso establecemos **notepad.exe**.

**$** git config --global core.editor

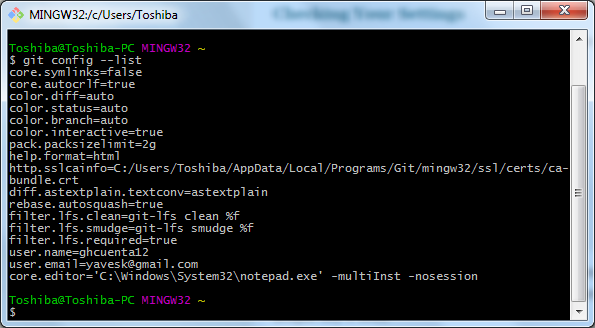


**3. Liste la configuración global con que termina este paso.**

**REVISIÓN DE LA CONFIGURACIÓN**

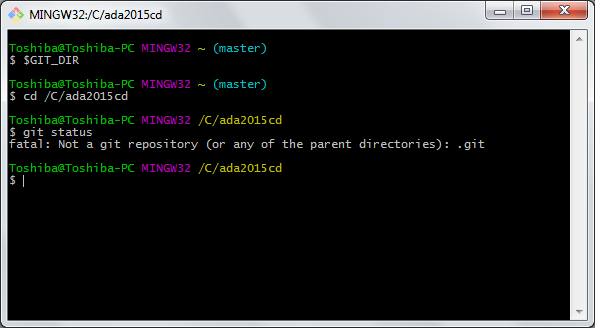
Comando para ver las variables de configuración.

**$** git config --list



**4. Ubique una ruta en su equipo donde desea trabajar. Cree un directorio de ser necesario. Proporcione evidencia de que este directorio no es un depósito git.**

Creamos un directorio en “**C:\ada2015cd**” y verificamos si es un depósito git.



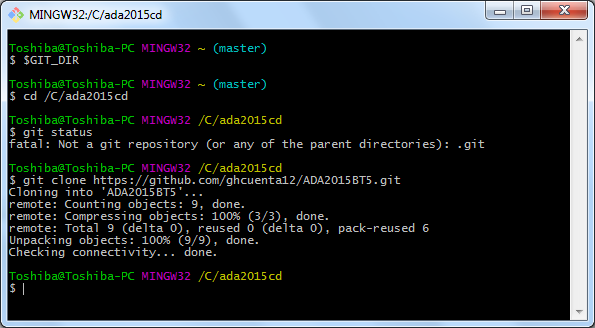
Error de que no es un depósito git

**5. En la ruta seleccionada proceda con la clonación de repositorio bifurcado en el primer paso de esta sección. El URL a usar debe aparecer en la página del repositorio. Buscar:**

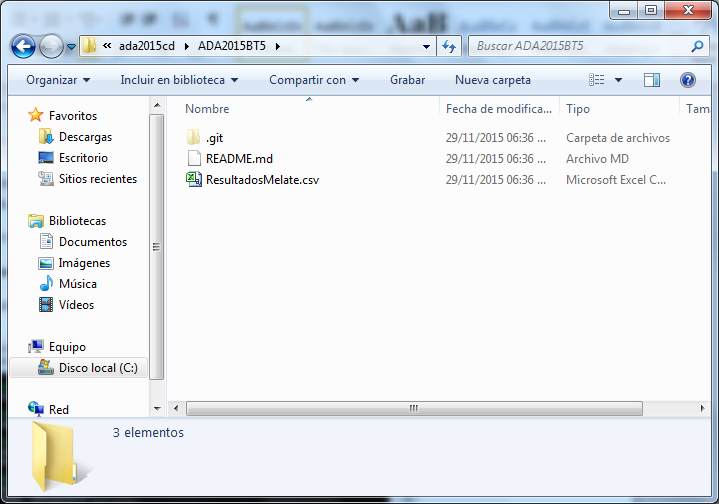
Clonamos el repositorio bifurcado de la siguiente ubicación:

**https://github.com/ghcuenta12/ADA2015BT5.git**

A continuación vemos el resultado del clon en la consola de comandos.



Vemos que en nuestro directorio ya se ha clonado el repositorio **ADA2015BT5**.



**6. Investigue al respecto de la diferencia entre clonar y bifurcar un repositorio. Indique si son o no parte de la funcionalidad de git o si alguna es propia de GitHub.**

Cuando hacemos una **bifurcación**, se crea un nuevo repositorio en nuestra cuenta con una URL diferente, entonces habrá 2 repositorios git idénticos independientes que pueden evolucionar de forma totalmente autónoma. Los cambios que se hacen en la bifurcación no se transmiten automáticamente al repositorio original, lo mismo sucede inversamente. Mientras tanto cuando **clonamos** un repositorio, nos bajamos una copia a nuestra máquina, podemos hacer modificaciones a nuestra copia y cuando realizamos un “push”, estaremos modificando el repositorio original que hemos clonado. El **clone** es una Función de **git** y la función **Fork** es de **gitHub**.

Con git **clone** podemos copiar el código de un repositorio de forma local, pero no estará conectado al repositorio de origen. Con **Fork** podremos crear una copia en remoto de un repositorio git en nuestra cuenta de **GitHub** y de esa forma será más fácil clonar de forma local para hacer pull y push de los cambios realizados.

Para mas información puedes verificar la documentación oficial

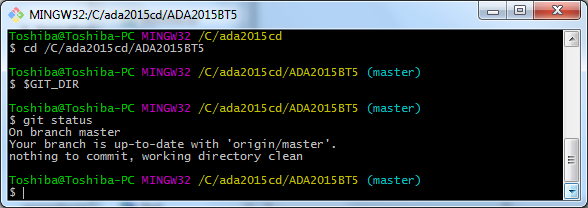
**Github Fork :** <https://help.github.com/articles/fork-a-repo/>

**Git Clone :** <http://git-scm.com/docs/git-clone>

**7. El clonado deberá crear un directorio nuevo con el nombre del repositorio. Proporcione evidencia de que éste es un almacén git.**

Ejecutamos el siguiente comando, para verificar que es un repositorio git, en caso contrario nos mandará un mensaje como en el punto 4 “**fatal: Not a git repository (or any of the parent directories): .git**”:

**$ git status**



**TERCERA PARTE**

**Usando**RStudio**:**

**1. Fije como directorio de trabajo el directorio creado en la segunda parte de este trabajo. Tome evidencia.**

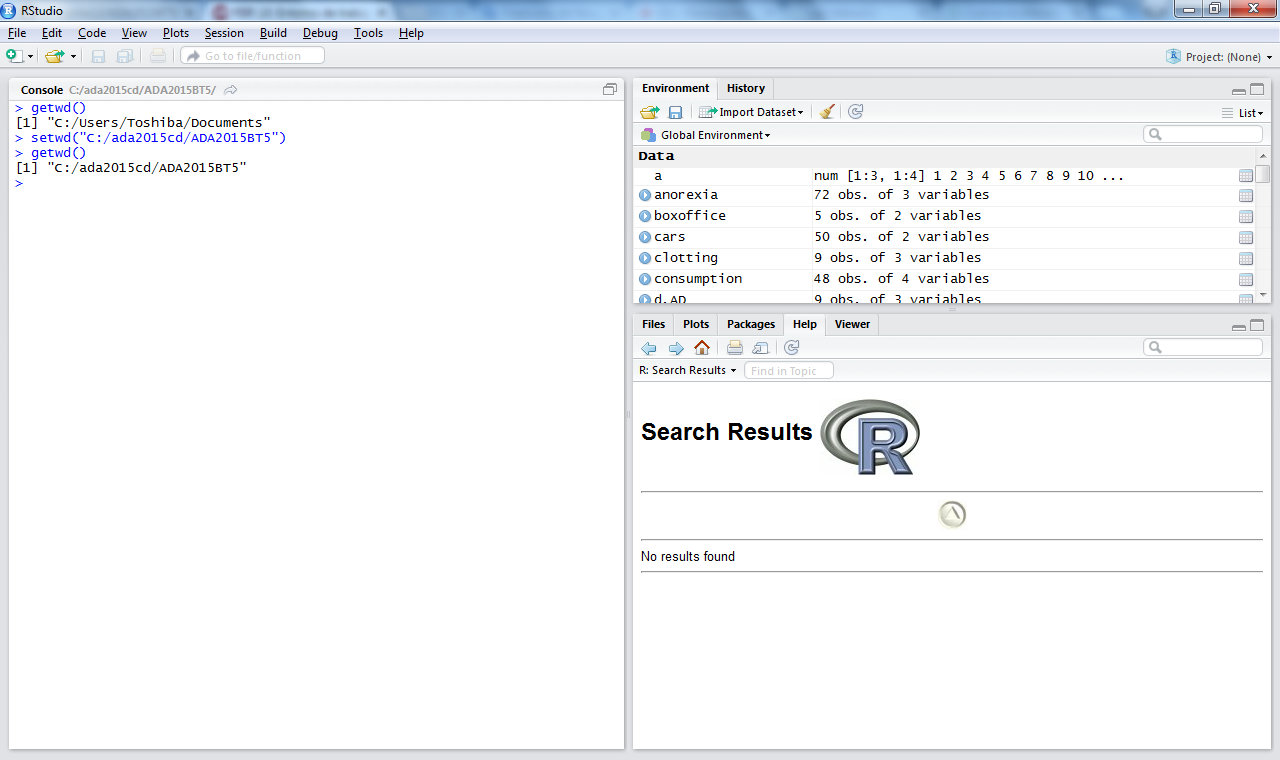
Con el comando:

Actualizamos el directorio de trabajo

**> setwd("C:/ada2015cd/ADA2015BT5")**

Muestra el directorio de trabajo

**> getwd()**

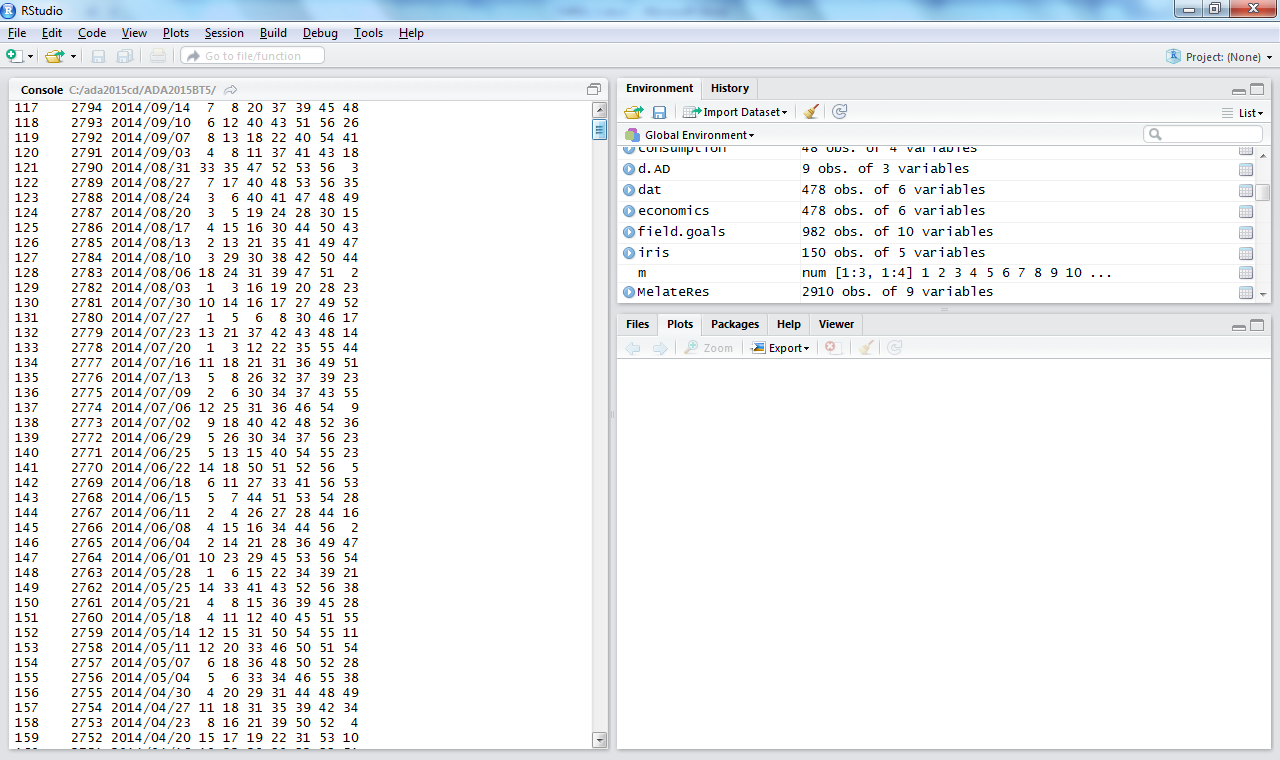


**2. Pruebe primero la ventana de la consola de RStudio las instrucciones necesarias para las siguientes acciones:**

* Lea el archivo CSV dentro del repositorio git (tip: una forma es mediante la función read.csv)

Ejecutamos el siguiente comando para leer el archivo:

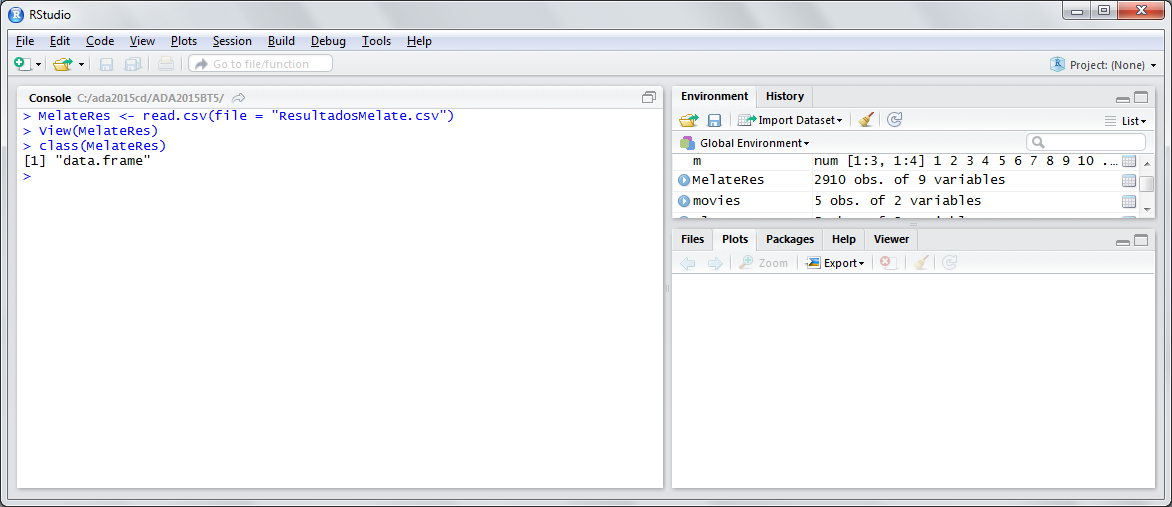
**> read.csv(file = "ResultadosMelate.csv")**



* Haga una primera exploración del archivo mostrando:
* Encabezados.
* Estructura del archivo y algunos registros.
* Un resumen (summary).
* Dimensiones del archivo.
* El archivo cuenta con varias columnas. Seis de ellas son de particular importancia. Haga un histograma por cada una.
* Opcional: cualquier otra acción sencilla exploratoria del contenido del archivo es bienvenida.

Para trabajar más fácilmente con el archivo lo guardamos en un **data.frame** llamado **MelateRes**.

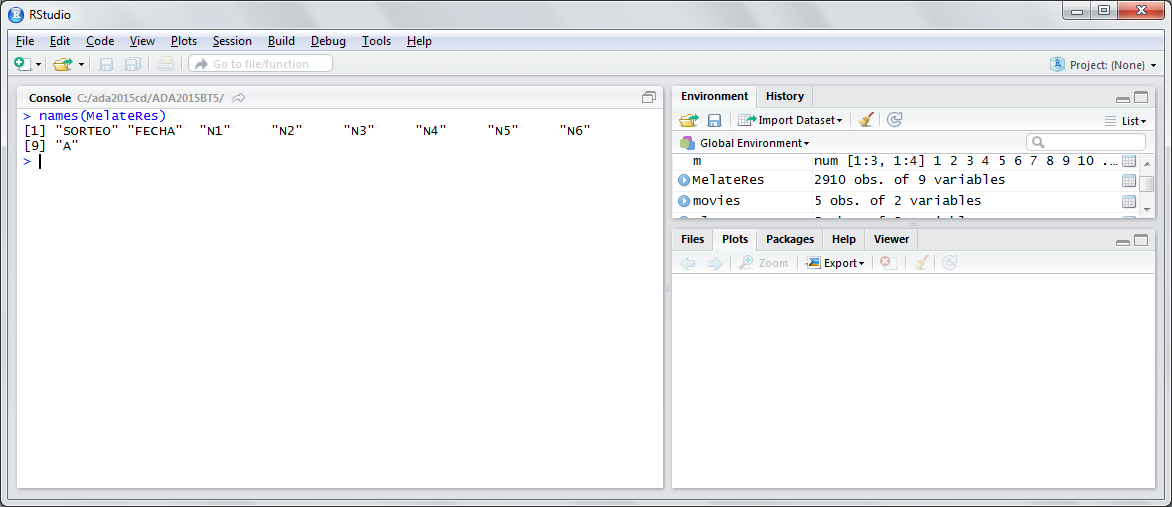
**> MelateRes <- read.csv(file = "ResultadosMelate.csv")**



**Encabezados**

Ejecutamos el siguiente comando para ver el nombre de las columnas de la **data.frame**.

**> names(MelateRes)**

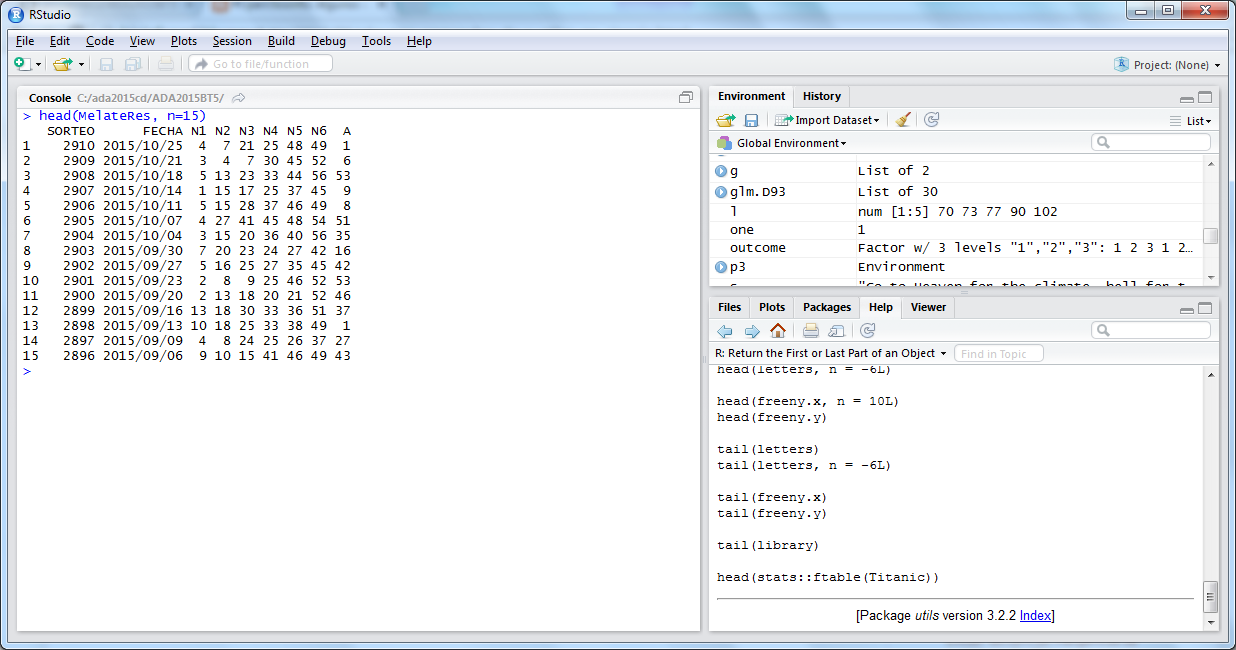


**Estructura del archivo y algunos registros**

Ejecutamos el siguiente comando para ver el nombre de las columnas de la **data.frame**.

Con el comando **head** visualizamos solo los primero 15 registros del **data.frame**, con su estructura de cada columna.

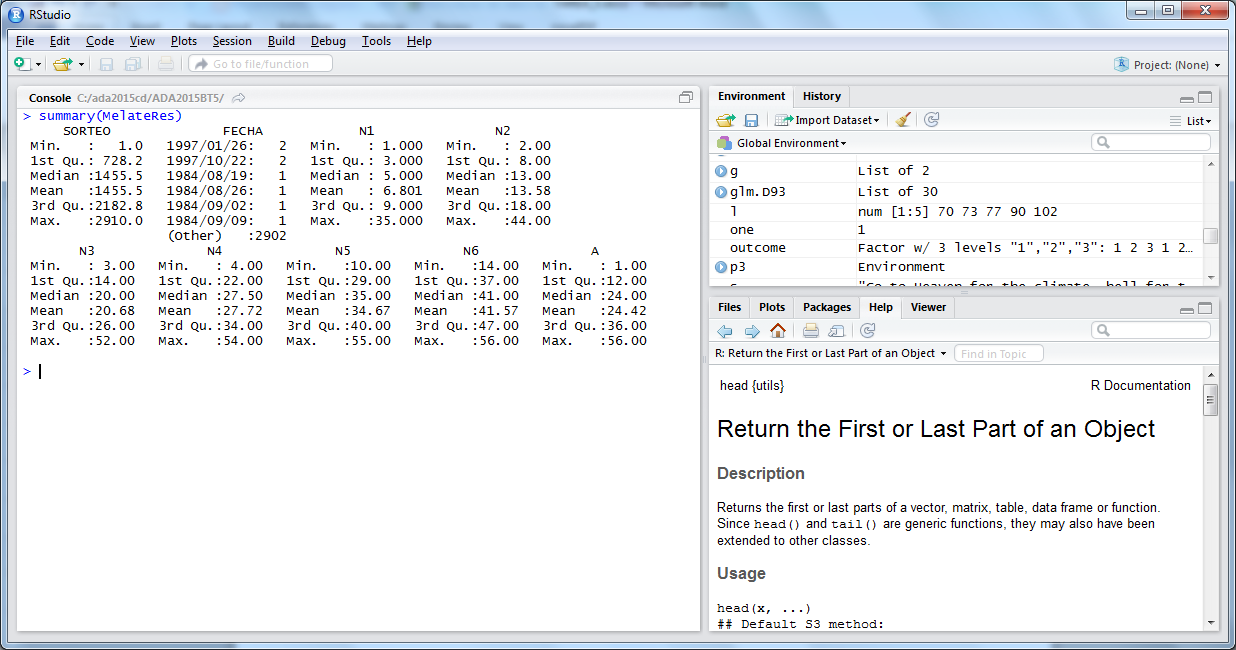
**> head(MelateRes, n=15)**



**Summary**

Ejecutamos el siguiente comando para ver un resumen de los datos del **data.frame MelateRes**.

**> summary(MelateRes)**



Al ejecutar este comando nos arroja la información de cada columna con las siguientes variables: **Valor mínimo, 1er cuartil**, **mediana**, **media**, **3er cuartil** y **valor máximo**.

**Dimensiones del archivo**

Ejecutamos el siguiente comando para ver las dimensiones de **MelateRes**.

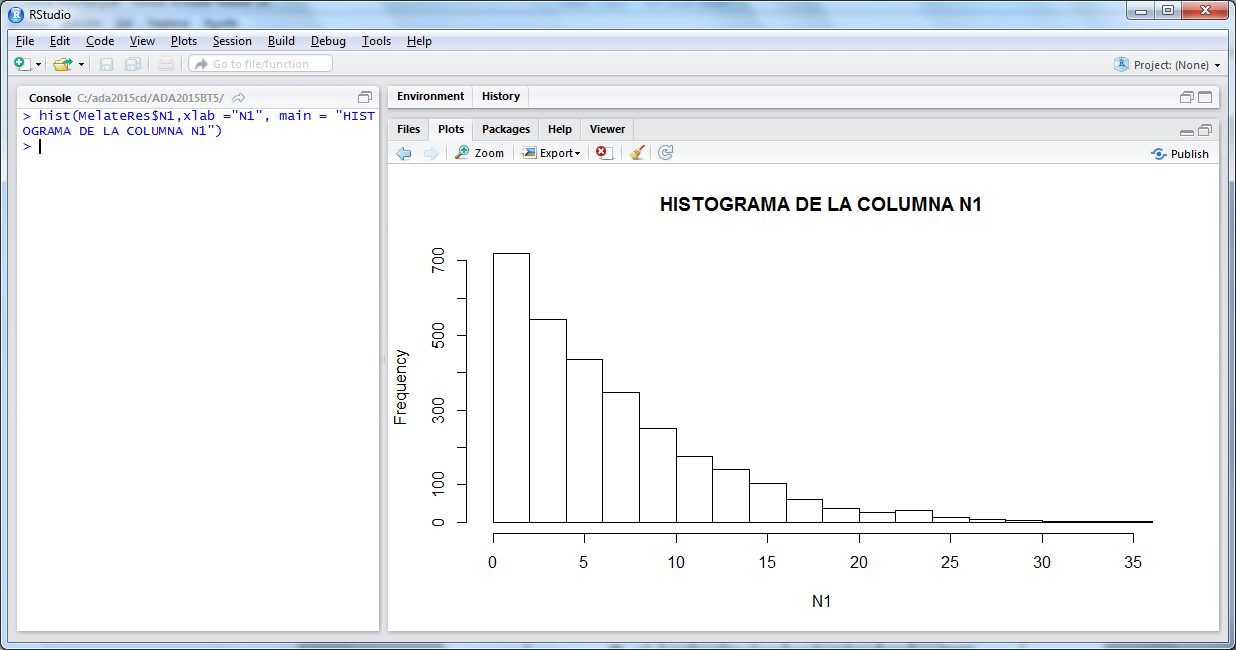
**> dim(MelateRes)**

Nos indica que existen 2910 observaciones (Filas) y 9 atributos (Columnas).

**Histogramas.**

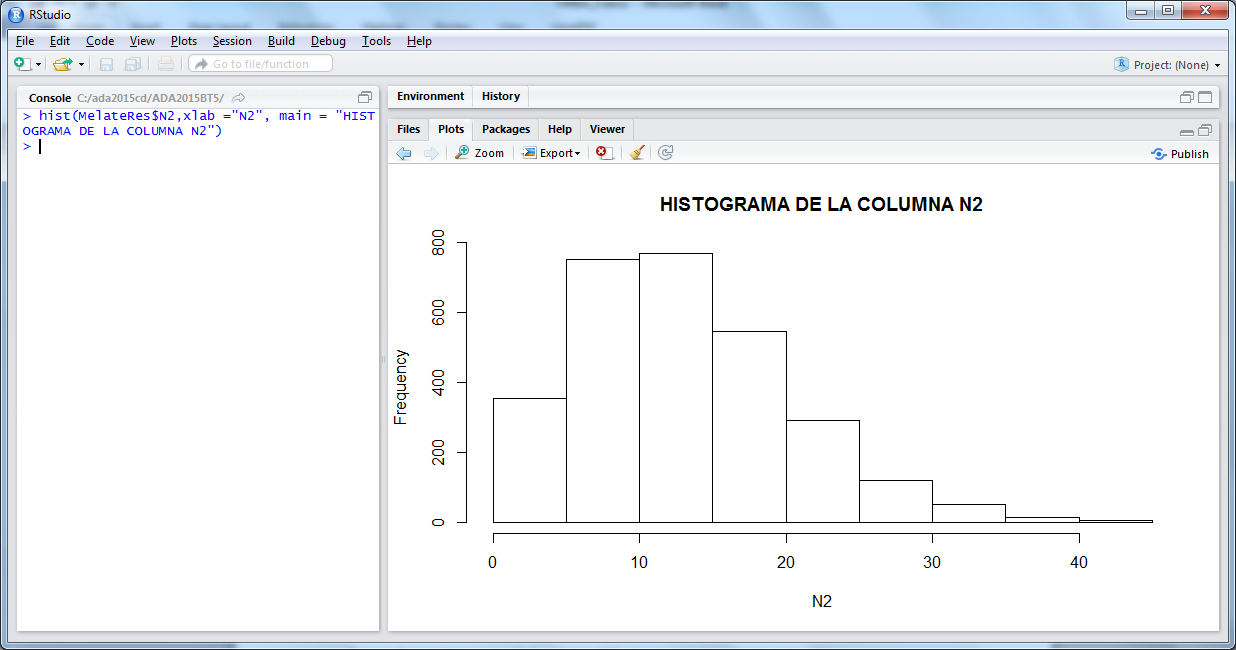
Para realizar el Histograma de la columna **N1**, ejecutamos el siguiente comando:

**> hist(MelateRes$N1,xlab ="N1", main = "HISTOGRAMA DE LA COLUMNA N1")**



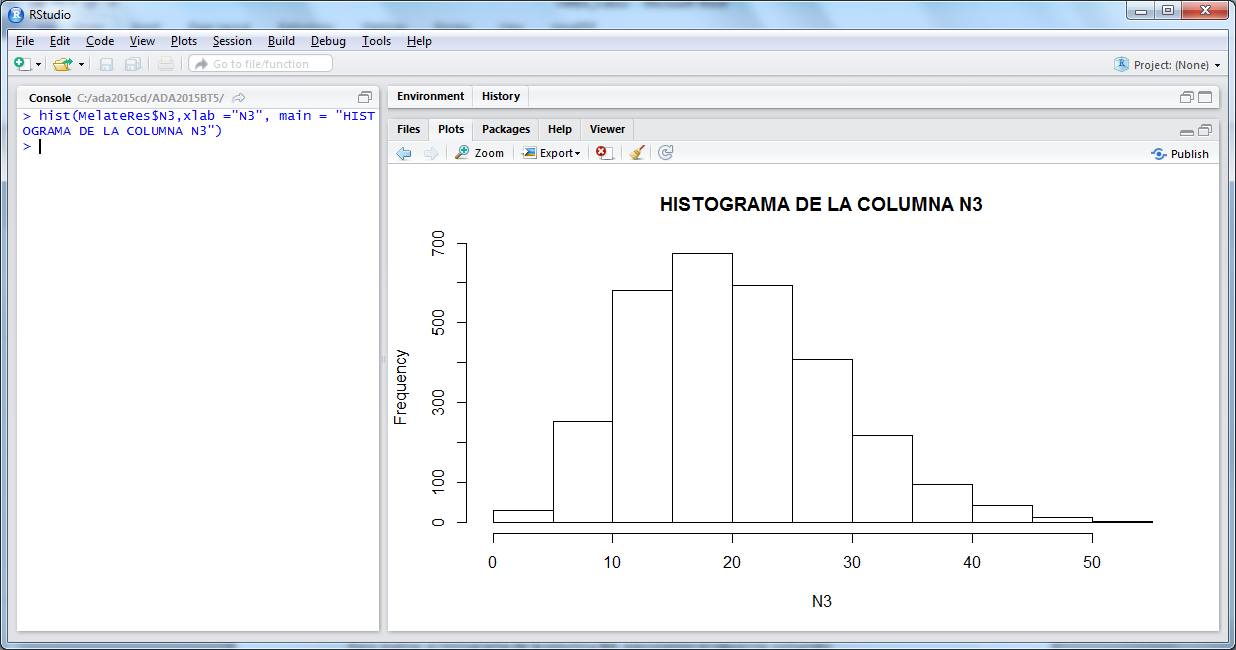
Para realizar el Histograma de la columna **N2**, ejecutamos el siguiente comando:

**> hist(MelateRes$N1,xlab ="N2", main = "HISTOGRAMA DE LA COLUMNA N2")**



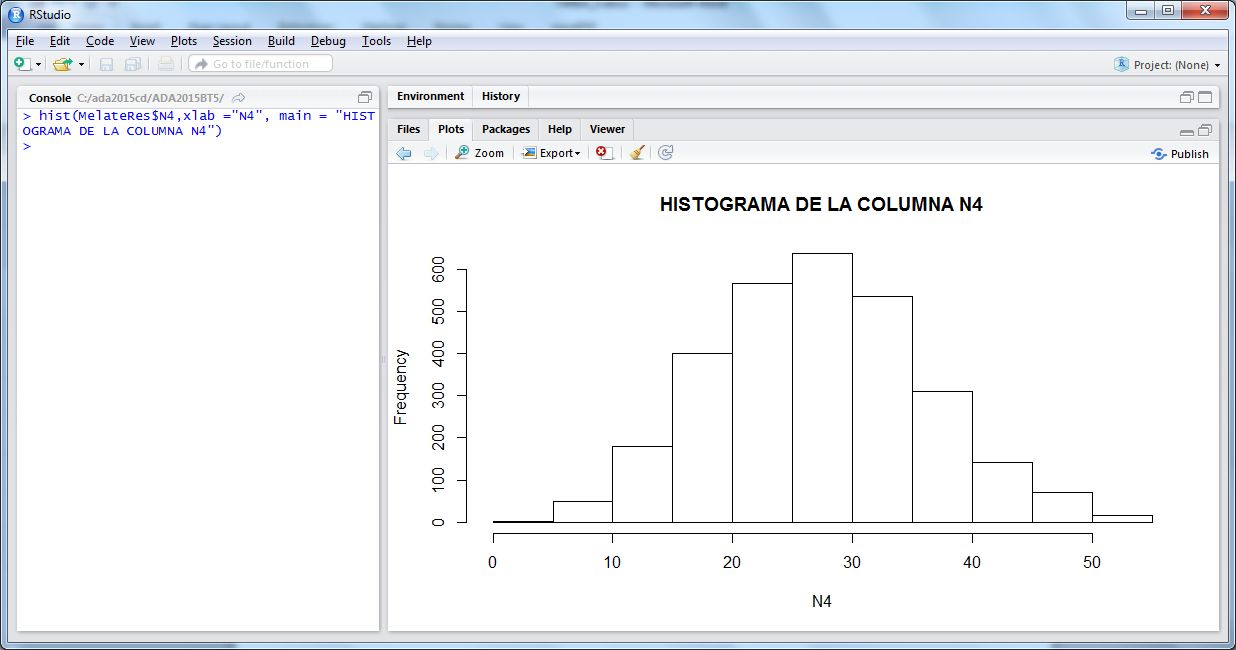
Para realizar el Histograma de la columna **N3**, ejecutamos el siguiente comando:

**> hist(MelateRes$N3,xlab ="N3", main = "HISTOGRAMA DE LA COLUMNA N3")**



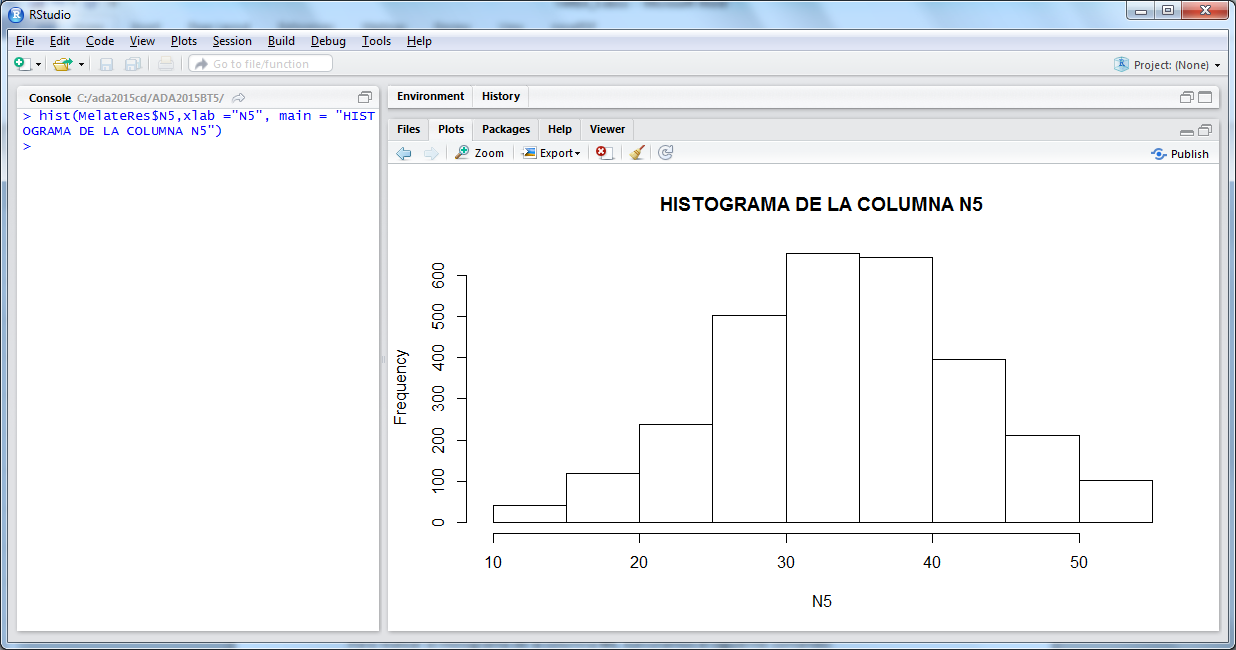
Para realizar el Histograma de la columna **N4**, ejecutamos el siguiente comando:

**> hist(MelateRes$N4,xlab ="N4", main = "HISTOGRAMA DE LA COLUMNA N4")**



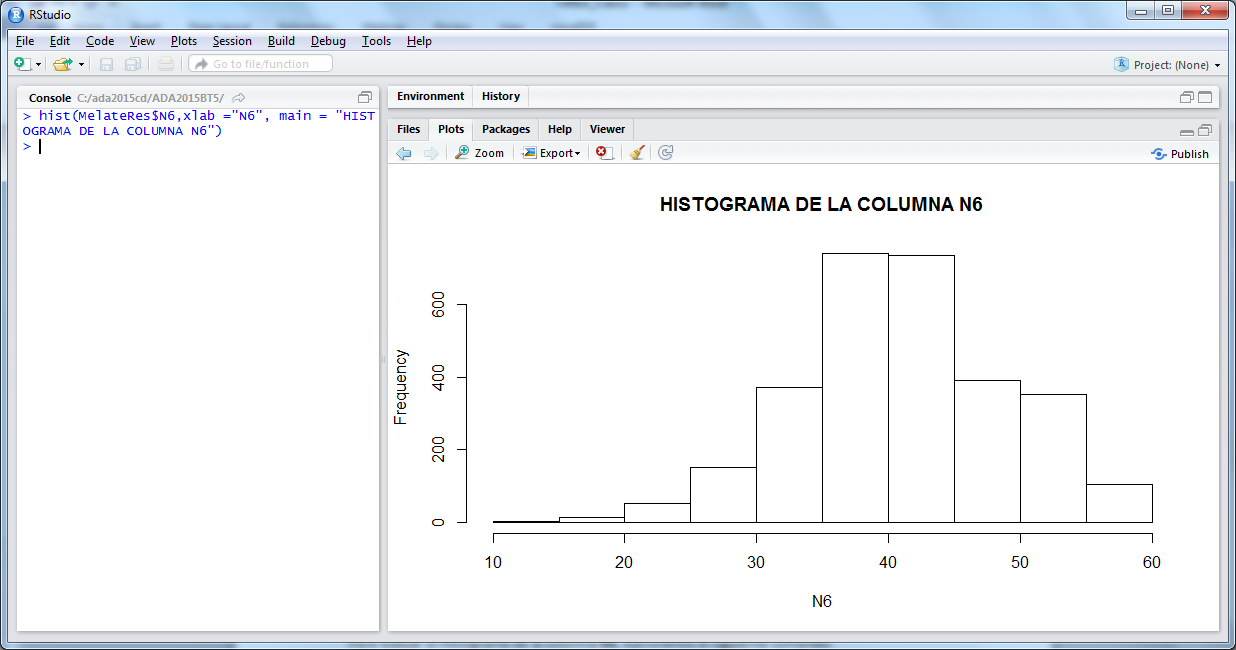
Para realizar el Histograma de la columna **N5**, ejecutamos el siguiente comando:

**> hist(MelateRes$N5,xlab ="N5", main = "HISTOGRAMA DE LA COLUMNA N5")**



Para realizar el Histograma de la columna **N6**, ejecutamos el siguiente comando:

**> hist(MelateRes$N6,xlab ="N6", main = "HISTOGRAMA DE LA COLUMNA N6")**

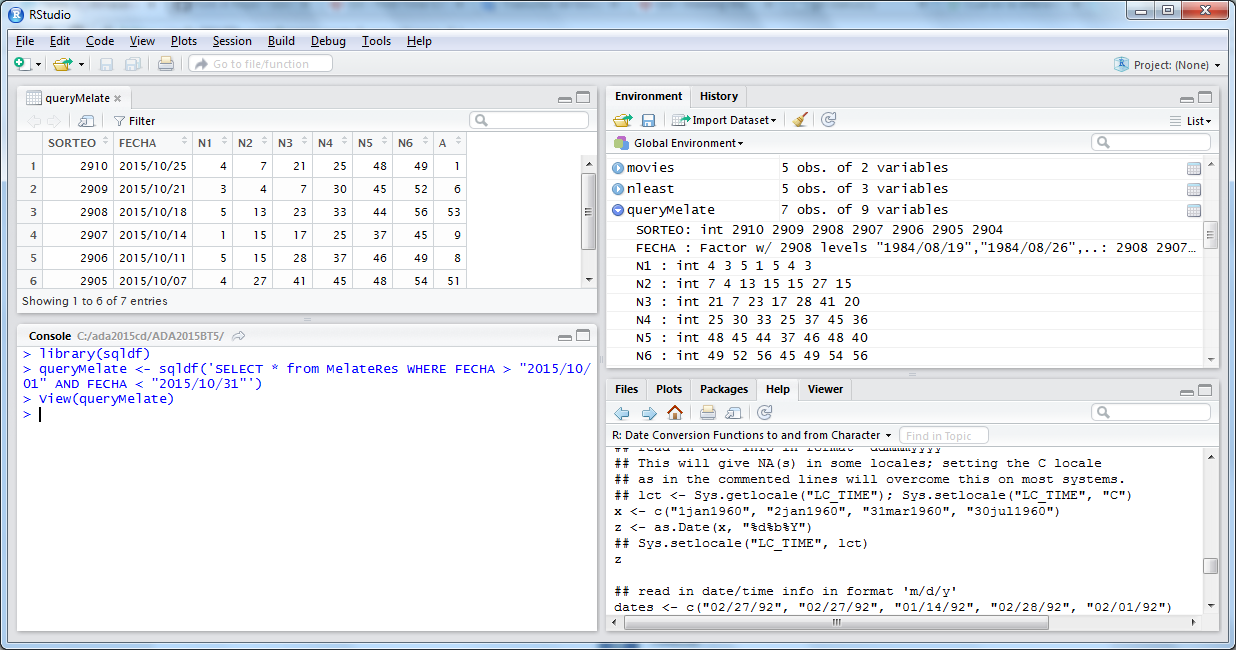


**Opcional**

Como me gustan las base de datos, he ejemplificado una consulta con el archivo que es en ocasiones es más fácil que usar funciones trabajando directamente con **data.frame**. Primero debemos cargar la librería para poder utilizar sus funciones. La consulta la guardamos en un **data.frame** llamado **queryMelate**.

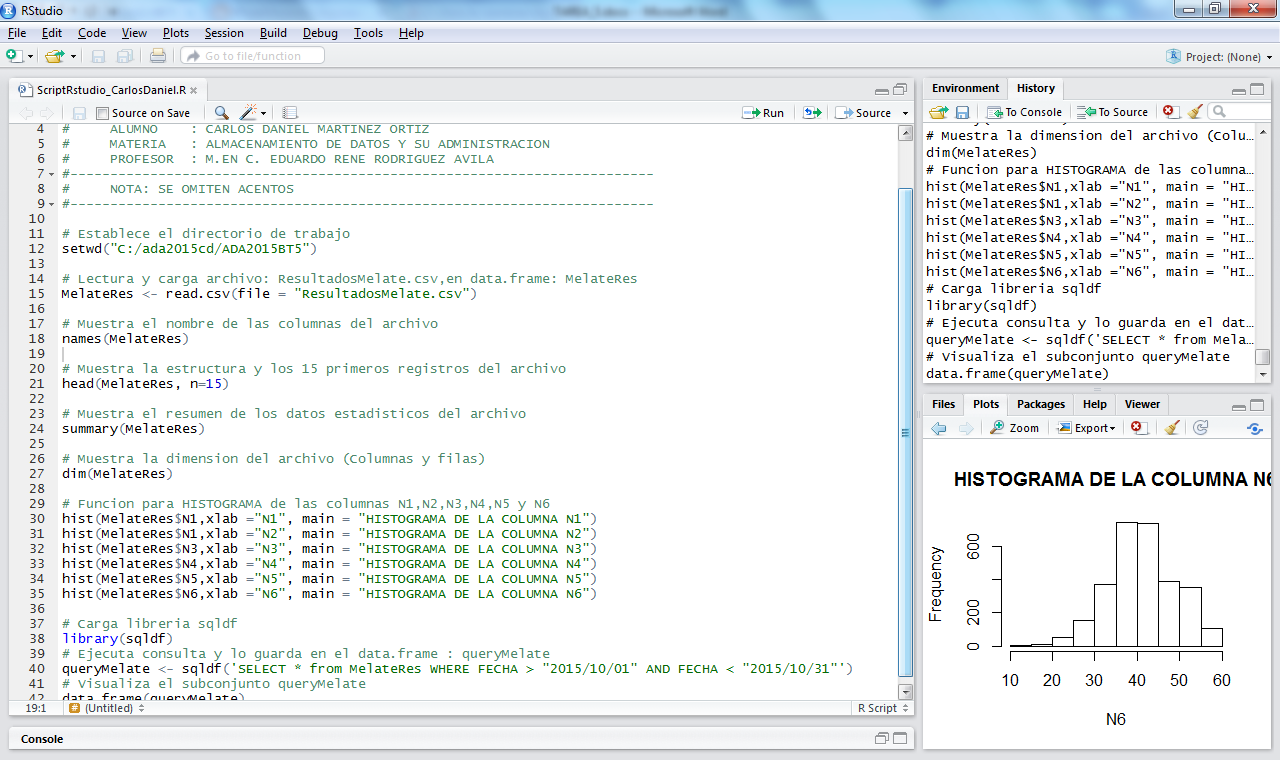
**> library(sqldf)**

**> queryMelate <- sqldf('SELECT \* from MelateRes WHERE FECHA > "2015/10/01" AND FECHA < "2015/10/31"')**



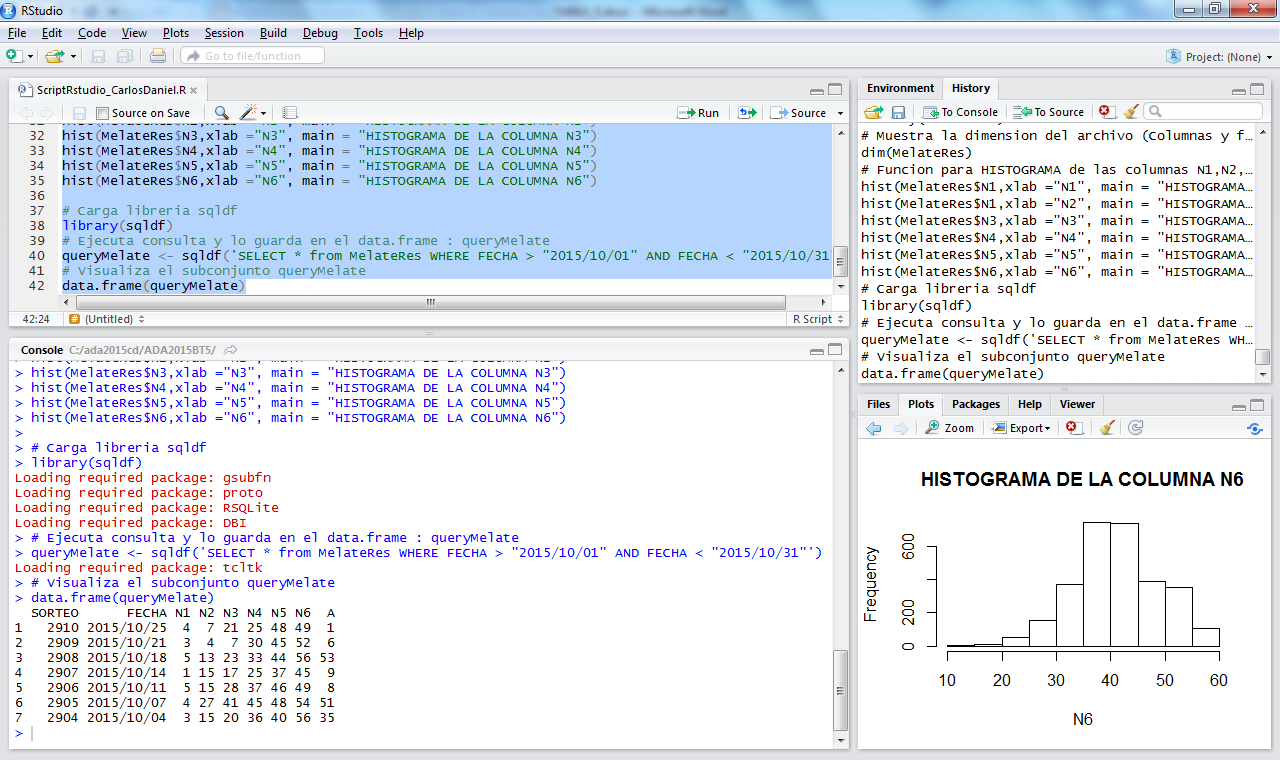
**3. Desde el menú de RStudio, cree un nuevo *script* R (File -> New File -> R Script) en el que colocará las instrucciones manualmente probadas en el paso previo.**

A continuación se muestra el Script que creé llamado “**ScriptRstudio\_CarlosDaniel.R**”.



**4. Ejecute el script para validar su funcionamiento.**

A continuación muestro la ejecución completa del Script.



**5. Cuando considere que su script está terminado agréguelo  a la lista de monitoreo de git y sincronícelo con su repositorio. Tome evidencia del proceso.**

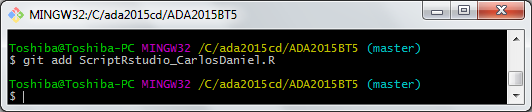
Para sincronizar ejecutamos los siguientes comandos:

git add

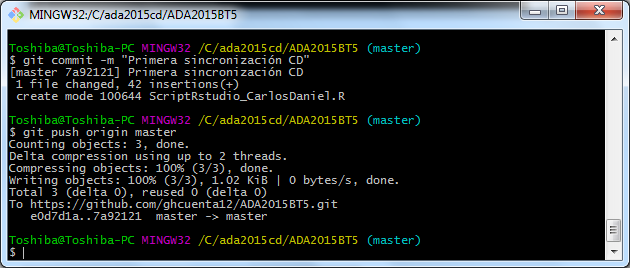
git commit -m "Comentario"

git push origin master

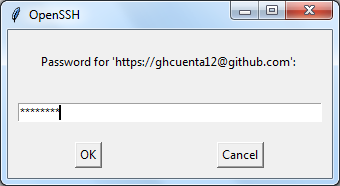
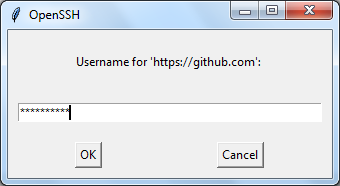
**Agregamos el archivo**



**Hacer commit y enviar cambios.**

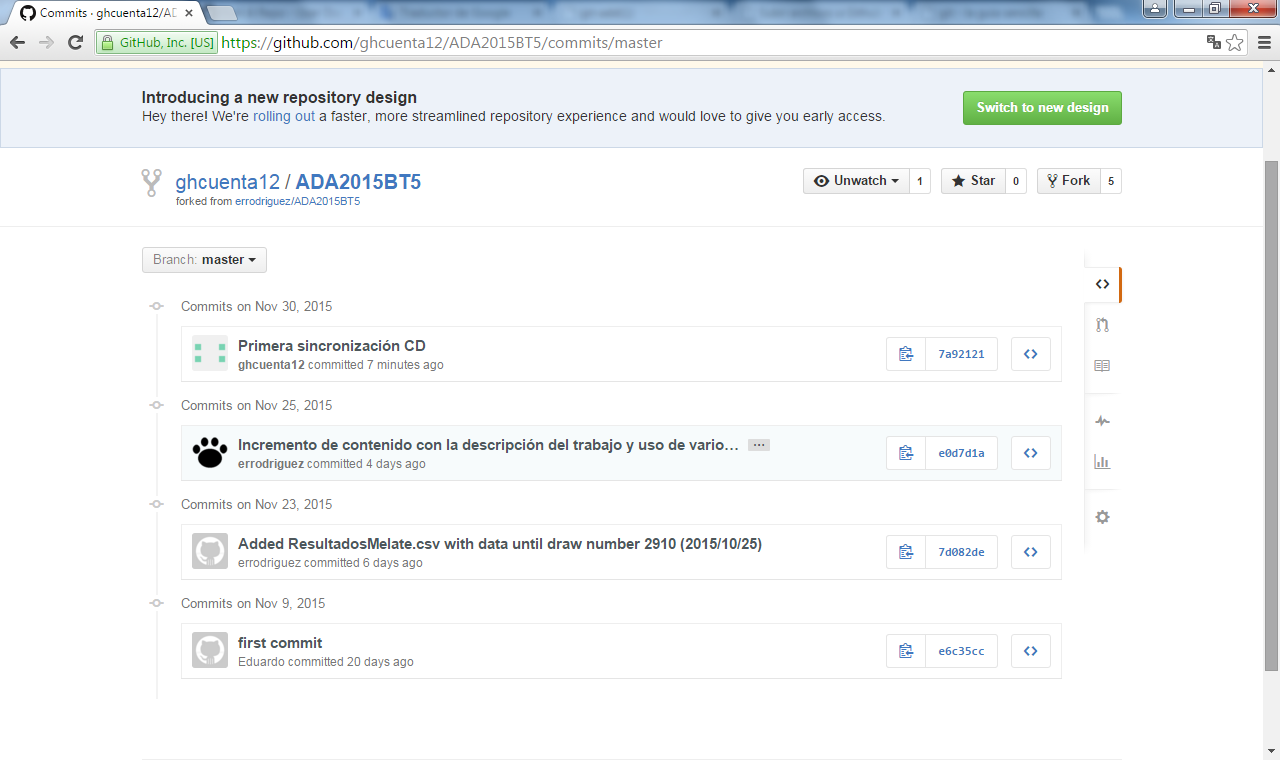


Ventanas donde nos pide ingresar usuario y password.



**6. De *click* en su contador de *commits.***

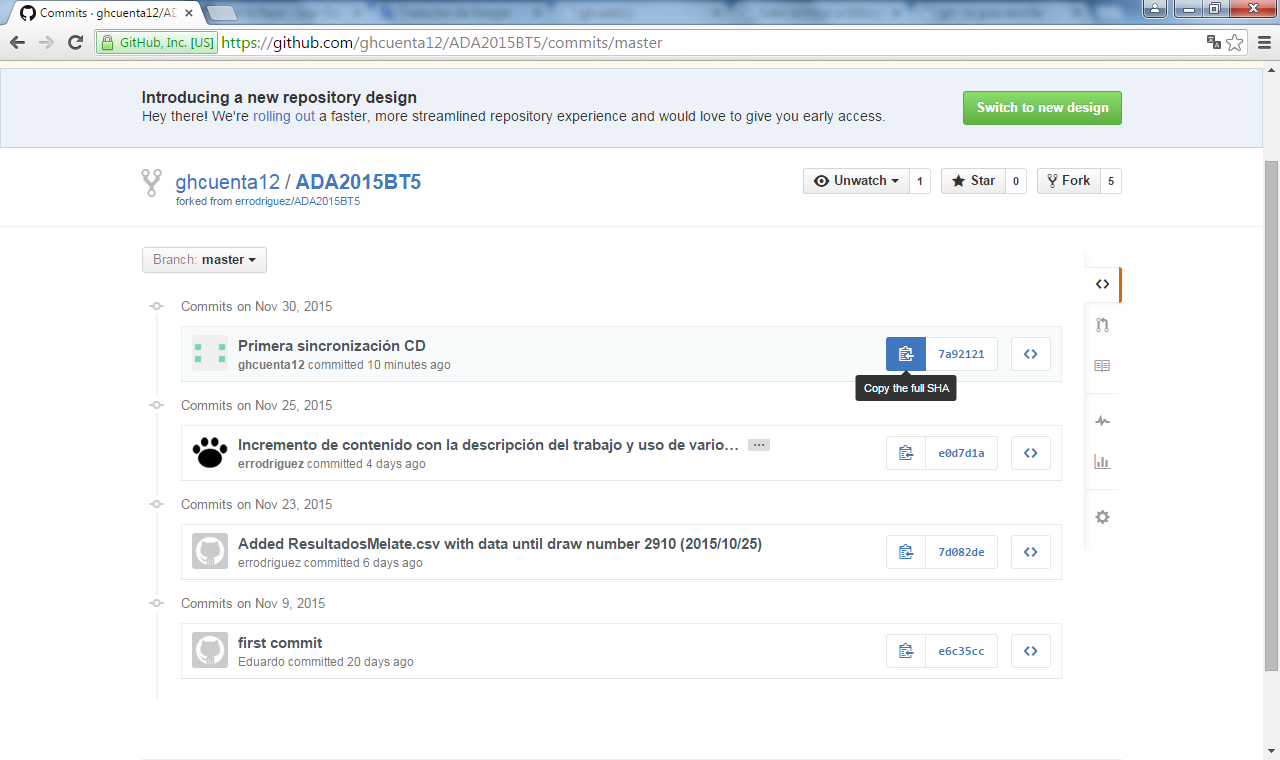
Observamos el commit realizado.



**Commit** realizado

**7. De la página resultante copie el *hash* SHA del último *commit* de su *script* y documéntelo en su trabajo.**

El hash SHA es: **7a92121d41f48252d22fab4a4c1f21a02cfcd879**



**8. Exprese sus observaciones, opiniones o conclusiones sobre el archivo explorado y sus resultados. Documente su script (comentarios) y en el documento a presentar.**

**9. Sincronice el documento que presentará con el repositorio. Indique que es el 1er commit en su texto de sincronización.**

La sincronización ya se realizó con el texto “**Primera sincronización del Documento**”.

**10. Agregue el URL de su repositorio al documento como una referencia bibliográfica y sincronice de nuevo (2do commit). Indique las razones en su mensaje de sincronización.**

Modificación del script, agregando la URL del Repositorio.