¿Qúe puedo incluir en mis documentos?

Miguel Equihua

Xalapa, Ver., 28 de febrero, 2025

Visita a la galería de Quarto

Platiquemos sobre tus intereses

Tu Blog contendrá alguno o varios de los siguientes componentes:

- Tablas de datos.
- Gráficas.
- Mapas.
- Análisis de datos geográficos.
- Análisis de series de tiempo.
- Modelos matemáticos.
- Análisis estadísticos.
- Análisis de datos moleculares.
- Análisis de imágenes.
- Análisis de audio.
- Alguna otra cosa que sea de tu interés.

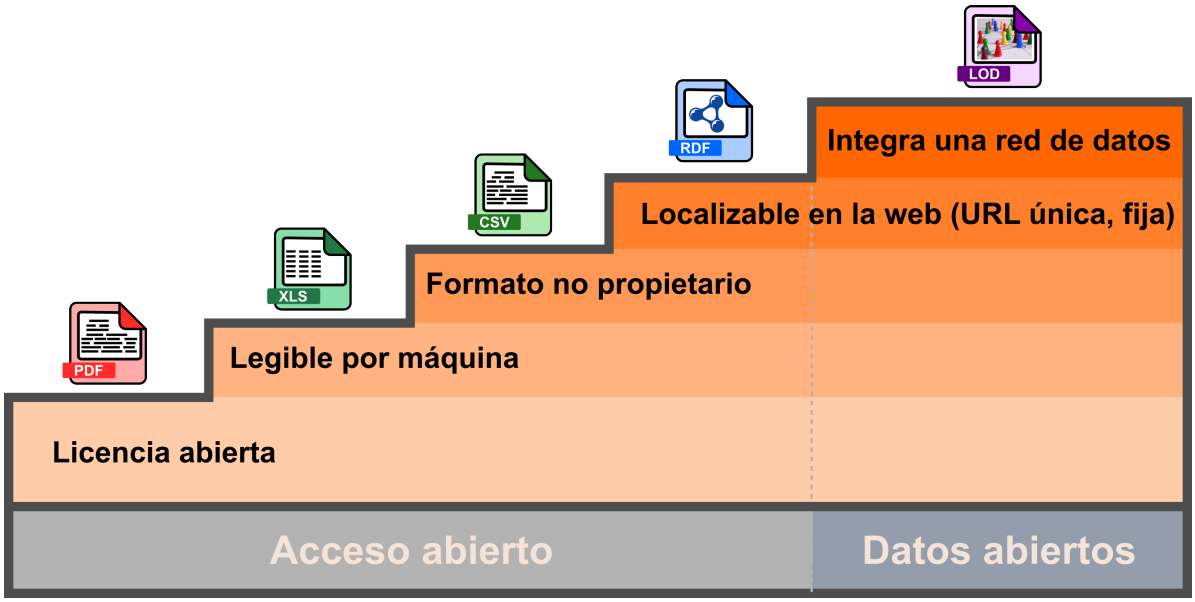
Sugiero hacer una reflexión sobre esto y considerar también de qué manera se obtendrán los datos, procesarán y presentarán.

Algunas ideas para animar la charla

rvest helps you scrape (or harvest) data from web pages. It is designed to work with magrittr to make it easy to express common web scraping tasks, inspired by …

Obtención de Datos Abiertos

Las fuentes de datos, especialmente hoy, pueden ser muy variadas. Desde los datos que conseguimos directamente en campo a partir de mediciones directas o encuestas, hasta los datos que podemos obtener de *fuentes de datos abiertos*. Considero que será de tu interés explorar las distintas formas de interacción que las fuentes de datos implican para nuestros procesos de producción *científica reproducible*.



Manejo de claves confidenciales

Un tema importante a cuidar es preservar la confidencialidad de claves, **tokens** y otras formas de identificación personal que puede implicar el proceso de acceso a datos en línea. Así que veremos eso como primer asunto. Queda claro que debemos evitar por todos los medios evitar poner esa información en carpetas o código que pueden acabar siendo registradas en Github en nuestro repositorio público. Haberte vacunado con `usethis::git_vaccinate()` ayuda en gran medida, pero desde luego no es remplazo a estar atentos a lo que estamos haciendo. La estrategia de registro de datos confidenciales que te propongo es la biblioteca `keyring`. Esta biblioteca accede al sistema de almacenamiento de credenciales de tu máquina desde *R*. La describen como una **API** *Independiente de la Plataforma* para acceder al depósito de credenciales del sistema operativo de tu máquina. Este sitio explica que es una API (Application Programming Interface). Actualmente `keyring` soporta: **Keychain** en *macOS*, **Credential Store** wn *Windows*, **the Secret Service API** en *Linux*, soluciones simples (sin plataforma específica) desarrollados con variables de sistema o archivos encristalados e incluso ofrece la posibilidad de desarrollar algunas soluciones propias con sencillez. En nuestro caso, básicamente usaremos dos funciones de esta biblioteca. Primero y desde la pantalla de

Consola ejecuta:

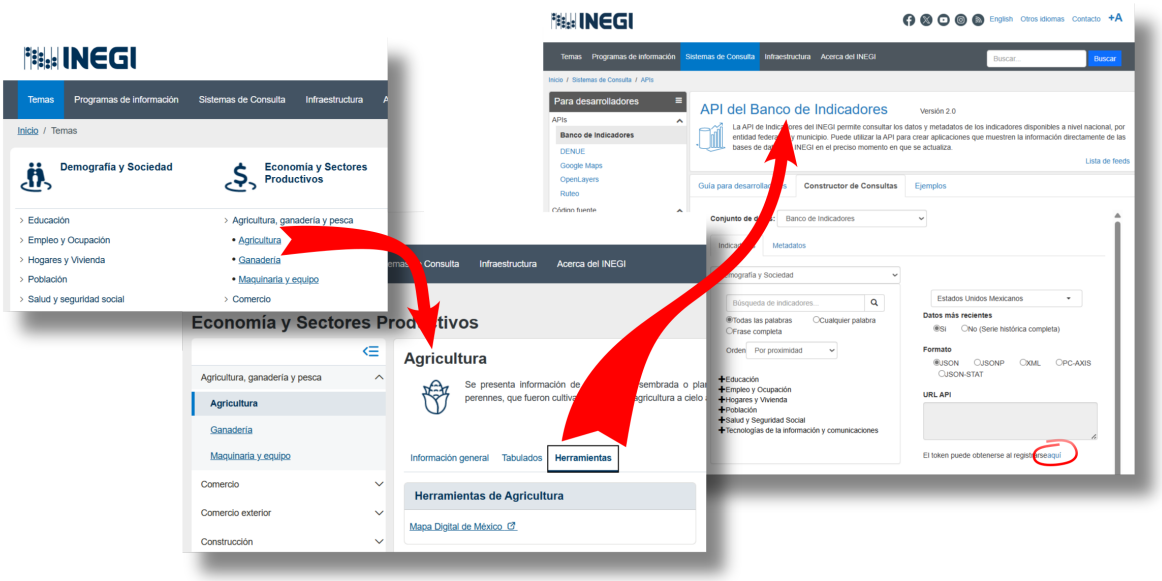
```
key_set(service = "[dale un nombre]", username = "[el que desees]")
```

Esto hará el registro de tus credenciales en tu máquina, fuera de la vista. A partir de ese momento y mientras no elimines el registro explícitamente, estarán disponibles los datos que hayas registrado y los podrás obtener con:

```
key_get(service = "[dale un nombre]", username = "[el que desees]")
```

Esta última línea recupera los datos confidenciales, así que deberás cuidar usar las credenciales de inmediato y procurar no guardarlas y menos desplegarlas o habilitar medios para mostrarlas, durante el proceso.

Veamos un primer ejemplo con **INEGI**. Te sugiero ir a https://www.inegi.org.mx/servicios/api_indicadores.html al constructor de consultas, en donde **INEGI** nos muestra un ejemplo de como acceder a los datos abiertos que *compilan*, *mantienen* y *custodian*. Deberás obtener un **token** personal, el mismo sitio de **INEGI** te dirá como obtenerlo. El ejemplo muestra como obtener datos de la **serie histórica** del indicador de la **Población total** de los Estados Unidos Mexicanos, en idioma español, en formato *JSON*. Una vez que los obtengamos mostraremos los datos en tablas y gráficas.



Lo primero que haremos es preparar el acceso a los datos con el **token** confidencial y obtenemos los datos, sin haber registrado la *URL* de acceso, pues como viste arriba, incluye tu **token**, así que habrá que manejarla con seguridad. El resultado de este *código* es una estruuctra de datos que ya no contiene información confidencial.

La siguiente tarea que haremos ahora es simplemente arreglar los datos y ponerlos de la manera que requiero. Los datos son actualmente una base de datos *JSON*, que es una estructura parecida a un diccionario jerárquico, que tiene una etiqueta seguida de los datos que le corresponden. Aquí te muestro un fragmento de los datos de **INEGI** que obtuvimos. Es la sección etiquetada como *Header*. Podemos ver que esta etiqueta tiene como contenido los datos *Name* y *email*, a su vez con sus respectivos datos.

```
descrip <- fromJSON(datosGenerales[[1]])$Header
prettify(toJSON(descrip))
```

```
{
    "Name": "Datos compactos BISE",
    "Email": "atencion.usuarios@inegi.org.mx"
}
```

Ahora haremos algunas operaciones para arreglar los datos del *JSON* en una tabla de tipo *data.frame* (*tibble* si optamos por una variante actual) en *R*. Los datos que nos interesan son los

que están en la etiqueta *Series* y dentro de estas *Series* están las listas de *OBSERVATIONS*, que en este caso son 15.

```
# junta todo en un gran texto corrido
flujoDatos <- paste(datosGenerales, collapse = " ")

# Obtiene la lista de observaciones
flujoDatos <- fromJSON(flujoDatos) # Convierte al JSON a una lista de R
flujoDatos <- flujoDatos$Series # Toma la sublista Series
flujoDatos <- flujoDatos[[1]]["OBSERVATIONS"]

cat("\nNúmero de observaciones: ", length(flujoDatos[[1]]),
    "\n\nDatos en cada observación:\n",
    paste("   ", names(flujoDatos[[1]][[1]]), collapse = "\n"), sep = "")
```

```
Número de observaciones: 15

Datos en cada observación:
    TIME_PERIOD
    OBS_VALUE
    OBS_EXCEPTION
    OBS_STATUS
    OBS_SOURCE
    OBS_NOTE
    COBER_GEO
```

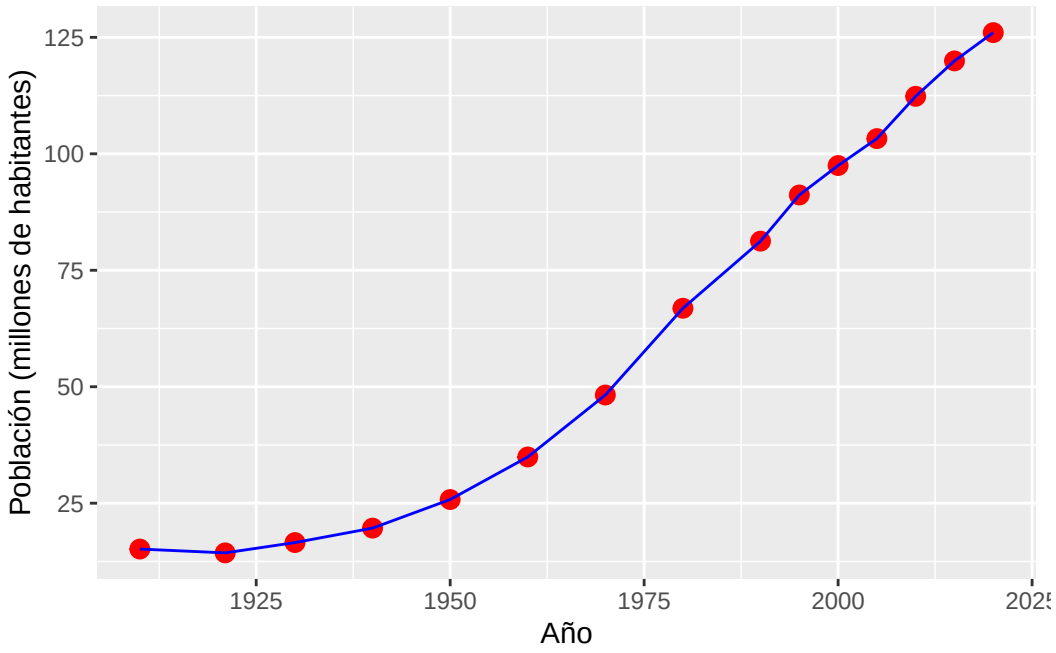
Ahora convierto lista de listas en un tabla con los datos de población y año de censado.

```
df1 <- flujoDatos[[1]] %>%
      sapply(., c) %>%
      t() %>%
      as_tibble() %>%
      select(TIME_PERIOD, OBS_VALUE) %>%
      mutate(TIME_PERIOD = as.integer(TIME_PERIOD),
             OBS_VALUE = as.integer(OBS_VALUE))


df1 %>% flextable() %>%
        colformat_int(j = 1, big.mark = "") %>%
        set_header_labels(TIME_PERIOD = "Año", OBS_VALUE = "Población")
```

Año	Población
1910	15,160,369
1921	14,334,780
1930	16,552,722
1940	19,653,552
1950	25,791,017
1960	34,923,129
1970	48,225,238
1980	66,846,833
1990	81,249,645
1995	91,158,290
2000	97,483,412
2005	103,263,388
2010	112,336,538
2015	119,938,473
2020	126,014,024

Ahora podemos ver los datos como una gráfica apropiada para pulicación. Nos ayudará la biblioteca `ggplot2`

```
library(ggplot2)

ggplot(df1, aes(x = TIME_PERIOD, y = OBS_VALUE / 1000000)) +
  geom_point(color = "red", size = 3, show.legend = FALSE) +
  geom_line(color = "blue", show.legend = FALSE) +
  ylab("Población (millones de habitantes)") +
  xlab("Año")
```



Contar historias con números

Esta es una tarea usualmente complicada. A lo mejor la lectura de lo que propone Stephen Few, nos puede ayudar a profundizar esta reflexión.