Tarea individual 3

Ciencia de Datos con R

Entrega

Esta tarea tiene que estar disponible en su repositorio de GitHub en la carpeta Tareas . Asegurate que tanto Mauro como yo seamos colaboradoras de tu proyecto privado STAT_NT. Recordar seleccionar en en opciones de proyecto, codificación de texto UTF-8. La tarea debe ser realizada en RMarkdown, la tarea es individual por lo que cada uno tiene que escribir su propia versión de la misma. El repositorio debe contener el archivo .Rmd con la solución de la tarea y los archivos que sean necesarios para su reproducibilidad que se evaluará.

Se espera que para resolver esta tarea se utilicen los paquetes contenidos dentro del `tidyverse`, si bien puede utilizar otros paquetes la tarea se puede realizar de forma satisfactoria únicamente con los paquetes vistos hasta el momento en el curso.

Como sabe `tidyverse` comprende una variedad de paquetes enfocados a la manipulación de datos, en este momento al estar en una instancia de aprendizaje (y también puede considerarse una buena practica) NO cargue el paquete `tidyverse` en su totalidad sino que al invocar las funciones utilice la siguiente sintaxis:

```
# NO USAR:

library(tidyverse)

# SI USAR:

dplyr::filter()

tidyr::pivot_longer()

paquete::funcion()
```

Para cumplir esto debe de averiguar para cada función que desee utilizar el paquete correspondiente dentro del `tidyverse`.

Ejercicio 1: Otra vez datos de COVID

1. Ingrese a Catalogo de Datos abiertos y busque el dataset **Vacunación por Covid-19**.

   Una vez allí encontrará diferentes archivos, en esta instancia nos concentraremos en el **Historico de actos vacunales**.

   Como puede ver existen diferentes formatos del mismo archivo, elija un formato en el cual se sienta cómoda para trabajar y guardelo en el directorio donde se encuentra trabajando en la tarea.

   Nota: El primer archivo es un archivo Excel, por algún error de la página no muestra el logo correspondiente.

2. Resuma el archivo descargado (incluyendo TODAS las variables) agregando por:

(Transformar el resumen por dia a uno por mes y por anio) (usar lubridate)

```
- Año
- Mes/Año

Para esta parte del ejercicio deberá de extraer el año de la columna fecha y armar el código *aniomes*.

**Pista**: Puede ayudarse de la siguiente formula:

$$ aniomes = anio*100 + mes$$
```

En el promedio se pierde las variables FechaDate y Fecha

3. Seleccione solo las columnas referidas al total de dosis por laboratorio.

```
Descarte el total de dosis general, por departamento y rango etario. **No seleccione las columnas por e
encontrar un patrón que le permita elegir las columnas de interés de forma resumida.
```

4. Obtenga el total de cada columna seleccionada.

5. En el formato actual, ¿es posible operar de forma ordenada?. En caso negativo realice las transformaciones correspondientes.

No esta ordenada porque tenemos dos variables por columna (numero de dosis y laboratorio de la vacuna)
y por lo tanto, cada fila corresponde a 3 observaciones en vez de una

6. Explique e incluya a la transformación de sus datos el siguiente chunk:

```
dplyr::mutate(
    dosis = factor(
      as.numeric(
        gsub(
          "([0-9]+).*$",
          "\\1",
          acto
        )
      ),
      ordered = TRUE
    ),
    laboratorio = sapply(
      X = acto,
      FUN = function(x) {
        strsplit(
            x,
            split = "_"
          )[[1]][3]
      }
    )
  )
```

Donde acto es una columna con los strings "**1era__dosis__sinovac**", "**2da__dosis__pfizer**", etc.

Con dplyr::mutate se crea dos variables: dosis y laboratorio. Para "dosis" se agarra el String de la variable acto y saca la primera secuencia de numeros dentro del String: Luego lo convierte en tipo numerico con as.numeric Los numeros se convierten en factores y se les indica que son ordenados con el parametro ordered que se le asigna TRUE. Para "laboratorio" se aplica una funcion a toda la columna de acto. La funcion saca del String de acto el nombre del laboratorio. Para eso usa la funcion strsplit que divide el string en partes utilizando el carácter "_" como separador. Luego, se accede al tercer elemento de la lista resultante, que corresponde al nombre del laboratorio, y se asigna a la variable "laboratorio".

7. Replique los siguientes gráficos e interprete brevemente.

Se puede ver que la vacuna preferida para las primeras dosis fueron Sinovac + Astrazeneca , pero luego para las sigueintes se prefirió Pfizer

Lo mismo que en el anterior grafico, aparte de que se puede ver como se abandonó el uso de las vacunas de otros laboratorios que no fueran Pfizer para la 3ra y 4ta dosis

8. Investigue el gráfico de Mosaico. Explique por qué podemos resumir los dos gráficos anteriores utilizando este tipo de visualización.

Con un grafico de mosaico se puede visualizar en un eje las proporciones , como en el primer grafico y en el otro eje las cantidades absolutas como en el segundo grafico

Ejercicio 2: Obteniendo datos del Banco Mundial como una experta

Parte 1: Producto Bruto Interno

1. Ejecute el siguiente código y explique brevemente que se realiza en cada chunk.

```
library(here)

dir <- here()
```

Carga la libreria here y asigna a "dir" la direccion de la raiz del repositorio

```
indicador <- "NY.GDP.PCAP.PP.KD"

query <- list(
  indicador = indicador,
  url = paste0(
    "https://api.worldbank.org/v2/es/indicator/",
    indicador,
    "?downloadformat=excel"
  ),
  destfile = here(
    dir,
    paste0(
      indicador,
      ".xls"
    )
  ),
  method = "curl"
)

if (!file.exists(query$destfile)) {

  do.call(
    download.file,
    args = query
  )

}
```

asigna a "indicador" un String que contiene un codigo crea una lista llamada "query" que contiene el codigo, la url a la que se va a hacer la peticion, el destino del archivo a descargar que se obtiene usando here, y el metodo usado para la descarga Si no existe previamente un archivo con el mismo nombre y ruta que el que se quiere descargar, se ejecuta la funcion download.file con la lista query para descargar el archivo

2. Con ayuda de las liberías vistas en el curso para leer archivos de Excel, lea el archivo descargado anteriormente.

```
**Pistas**:

- Busque como leer una sola hoja y seleccionarla por su nombre. **Pista** el archivo tiene 3 hojas, dos
- Investigue la manera de leer una matriz de datos de un libro utilizando rangos. Para ver la importanc
```

Note que al comienzo de la hoja se genera de forma automática información al descargar el archivo esto NO debe incluirlo, solo debe de considerar la información luego de la fila cuatro y las variables a importar son las siguientes:

- **Country Name**
- **Country Code**
- **Indicator Name**
- **Indicator Code**
- **Años (desde 1960 a 2021)**

3. Con lo aprendido en la Tarea 1 y en el curso, renombre de forma programática los nombres de las columnas obteniendo el siguiente resultado (solo se muestran los primeros valores):

Pista: Debe remplazar los espacios por un guión bajo y convertir el texto a minúsculas.

4. Explique brevemente por que los datos no cumplen con el concepto de `tidydata` visto en el curso.

Porque cada fila no corresponde con una sola observacion, sino que cada fila corresponde a todas las observaciones de un mismo pais (que son mas de una)

5. Transforme de forma conveniente el objeto para poder manipularlo de forma ordenada.
6. Filtre la información solo para **Argentina**, **Chile**, **Paraguay** y **Uruguay**.
7. Remplace los valores faltantes siguiendo la siguiente lógica:
   - Si la variable es del tipo númerico remplace con valor mas cercano por país y cree una nueva variable indicadora que valga 1 si el dato fue imputado.
   - En caso contrario remplazar con el string **SIN_DATO**
8. Defina brevemente la variable con la cual estamos trabajando, una pregunta de interés y un gráfico que intente responderla. Puede trabajar con los datos filtrados o con todos los datos disponibles.

   Al crear el gráfico recuerde que el mismo debe ser autocontenido, es decir debe incluir un titulo (utilizando fig.cap), etiquetas adecuadas en los ejes y una proporción acorde al tipo de gráfico.

   Para obtener mas información, puede buscar el indicador por su código (*NY.GDP.PCAP.PP.KD*) dentro del catalogo de datos del Banco Mundial.

   La variable es PBI per capita es la suma de todos los bienes y servicios finales producidos por un país en un año, dividido por la población estimada para mediados del mismo año. Como afecto la crisis sanitaria del 2020 a los cuatro paises de datos_banco5 En el grafico se puede observar un decrecimiento del PIB de 2019 a 2020 en los 4 paises pero se ve que en los 4 paises ya habia un deteriodo del PIB previo a la pandemia, la cual solo hizo que se asentuara.

Parte 2: Automatizando la importación de datos

Generalize la importación de datos anterior, como ayuda siga el siguiente esquema:

Debe de crear una función donde debe de incorporar la siguiente lógica:

1. Creación de la consulta al Banco Mundial, utilizando como argumento el código del indicador. Cada indicador tiene un nombre y un código interno, en esta situación necesitamos obtener el código y no su nombre. Puede revisar el formato en esta lista, seleccione uno, ingrese a detalles en la esquina superior derecha al gráfico y revise el campo **ID** o también en el link generado al ingresar al sitio web.
2. Descarga de archivo
3. Carga del archivo en formato 'sucio'
4. Limpieza de datos:

- Limpieza del nombre de las variables
- Incorporar el concepto de tidydata
- Imputación de datos faltantes

5. Filtrar ciudades

En el siguiente bosquejo puede encontrar una función por cada parte o lógica que debe de cumplir la función, es decir, debe de crear las siguientes funciones:

- `make_query` para la parte 1.
- `download_file` para la parte 2.
- `load_file` para la parte 3.
- `tidy_data` para la parte 4.
- `filter_country` para la parte 5.

Revise los argumentos sugeridos y reflexione sobre la inclusión o no de valores por defecto.

```
tidy_wdi <- function(
  country = "all",
  indicador,
  destfolder = here::here(
    "Tarea_3",
    "data"
  ),
  sheet_name = "REVISAR",
  range = "REVISAR"
) {

  # Generar la consulta al Banco mundial. Necesitamos:

  # El nombre del indicador como argumento y la función debe
  # contener devolver la MISMA lista del ejercicio 1.
  # Por temas de compatibilidad, utilice el método curl e
  # interpole con el indicador la misma url del ejercicio anterior.
  # Es decir, el Banco Mundial sigue el siguiente patrón:
  # https://api.worldbank.org/v2/es/indicator/{Indicador}?downloadformat=excel
```

```
  query <- make_query(
    indicador = indicador
  )

  # Incorporar la lógica de la descarga del archivo.
  # En el caso de existir el archivo NO se debe de descargar.

  download_file(
    query = query
  )

  # Carga de los archivos en formato "sucio"

  df <- load_file(
    filename = query$destfile,
    sheet_name = sheet_name,
    range = range
  )

  # Limpieza de datos:
    # Incluir la limpieza de los nombres del data.frame
      # (crear una función clean_names(df)
      # puede ser una buena idea)
    # Convertir el archivo siguiendo
    # el concepto de Tidy Data
    # Imputación de datos faltantes (
      # crear una función imputar_valores_faltantes()
      # ) puede ser una buena idea

  df <- tidy_data(df)

  # En el caso que se quiera filtrar países
  # (el argumento country debe ser distinto de 'all')
  # filtrar el data.frame

  df <- df  %>%
    filter_country(
      country
    )


  return(df)


}

prueba <- tidy_wdi(indicador = indicador, sheet_name = "Data", range = readxl::cell_limits(c(4, NA), c(
```

Parte 3: Función final

1. Investigue sobre el paquete WDI y compare con lo realizado en el ejercicio 2. Describa brevemente las funciones del paquete y reflexione sobre la utilidad sobre este tipo de herramientas.
2. Intente modificar la función `tidy_wdi` de forma que admita descargar múltiples indicadores. Puede utilizar funciones de la familia `apply` o investigar el paquete `purrr` contenido dentro del `tidyverse`.

Pista: En el caso de utilizar `purrr` revise la función `map_dfr` y en el caso de utilizar `apply` considere que la salida debe ser un data.frame.

¡Felicitaciones!

Esta a unos simples pasos de crear su primer paquete, en concreto un SDK (Software Development Kit) de la API (Application Programming Interfaces) del Banco Mundial. Simplemente le falta modificar lo siguiente (en el caso que le interese) y empaquetar sus funciones:

- Al crear la llamada al banco mundial, deberá de modificarlo para incorporar los métodos de solicitud HTTP, como pueden ser:
  - GET
  - POST
  - PUT
  - Un largo etc...

  Puede utilizar el paquete **httr** para utilizar estos verbos, mas información acá y revisar la documentación de la API con las llamadas disponibles aquí
- No descargará un archivo. Al hacer la solicitud, el servidor del Banco Mundial le devolverá la respuesta en formato **JSON** con toda la información, puede trabajar con este tipo de objetos con el paquete **jsonlite** para luego parsear los datos en un `data.frame` rectangular, mas información aquí
- En la misma llamada puede aplicar filtros de años y ciudades y no deberá de incorporar la lógica del filtrado con **dplyr**.

En esencia de esto trata la implementación del paquete **WDI** o **spotifyr**. Puede comprobarlo en los siguientes repositorios de Github

- WDI.
- spotifyr