Tarea final

Logística

Debes empujar tu trabajo final a la carpeta *tarea_final* de la rama *tareas* a más tardar el **jueves 12 de enero hasta las 23:59**. Solo debes empujar código al repositorio.

Tu envío debe ser una carpeta con tu nombre y apellidos (ej: pedro_muñoz_gonzalez). La estructura de la carpeta debe ser la siguiente:

- *main.R*: script principal. Desde acá debes llamar a tus funciones
- *functions.R*: funciones que generes para hacer la tarea
- .Rproj (obligatorio)

Encabezado

Tu jefatura te pide un reporte sobre algunos datos de la Encuesta Suplementaria de Ingresos (ESI). Un requisito es que el trabajo sea lo más reproducible posible y que sea autocontenido, es decir, que pueda ejecutarse desde principio a fin por otros usuarios y usuarias de R. Para esta tarea, **el uso de proyectos es obligatorio**.

Al realizar la tarea, ten en mente que idealmente tu trabajo debería permitir que otra persona pueda hacer pull (o clonar) y ejecutar tu código sin tener demasiados problemas. Incluir cosas de este tipo `if (!require("dplyr")) install.packages("dplyr")` es una buena idea y significa que estás siendo amable con tus colegas.

Nota: Una manera de mejorar considerablemente la reproducibilidad es utilizar ambientes virtuales. Esto no es un requisito de la tarea y no se considerará en la evaluación, pero te recomiendo revisar de qué se trata. En R existen, al menos, 2 paquetes que implementan ambientes virtuales: *renv* y *packrat*. Si decides trabajar con un ambiente virtual, por favor, **utiliza renv y no packrat**.

Instrucción general

Considerando el volumen de datos, los ejercicios podrían resolverse mediante soluciones manuales *adhoc*. Este tipo de soluciones no se considerará correcto. Se espera que el código sea capaz de resolver la tarea para n casos.

Ejercicio 1:descargar archivos

Tu jefatura valora mucho la reproducibilidad, de modo que te entrega algunas url, que deberás usar para descargar los datos desde la página institucional del INE.

```
urls <-
c("https://www.ine.cl/docs/default-source/encuesta-suplementaria-de-ingresos/bbdd/csv_esi/2021/
           esi-2021---personas.csv?sfvrsn=d03ae552_4&download=true",
  "https://www.ine.cl/docs/default-source/encuesta-suplementaria-de-ingresos/bbdd/csv_esi/
  2020/esi-2020---personas.csv?sfvrsn=fb1f7e0c_4&download=true",
  "https://www.ine.cl/docs/default-source/encuesta-suplementaria-de-ingresos/bbdd/csv_esi/2019/
  esi-2019---personas.csv?sfvrsn=9eb52870_8&download=true",
  "https://www.ine.cl/docs/default-source/encuesta-suplementaria-de-ingresos/bbdd/csv_esi/2018/
  esi-2018---personas.csv?sfvrsn=a5de2b27_6&download=true",
  "https://www.ine.cl/docs/default-source/encuesta-suplementaria-de-ingresos/bbdd/csv_esi/2017/
```

```
esi-2017---personas.csv?sfvrsn=d556c5a1_6&download=true",
"https://www.ine.cl/docs/default-source/encuesta-suplementaria-de-ingresos/bbdd/csv_esi/2016/
esi-2016---personas.csv?sfvrsn=81beb5a_6&download=true"
)
```

- Lo primero que haremos será encontrar los nombres de los archivos dentro de las url. Utilizando expresiones regulares, crea un vector llamado *file_names* que almacene la porción de las url que contiene el nombre de los archivos que luego descargaremos (ej: *esi-2021—personas.csv*). Para esto, crea una función llamada `extract_name`. `extract_name` debe recibir una url y devolver el nombre del archivo. **¡No olvides utilizar purrr en tu solución para el vector de nombres!**
- Crea una función llamada `download_esi_data` para descargar un archivo. La función debe recibir 3 parámetros: *url*, *file_name* y *directory*.
- Usando `purrr`, las url y el vector de nombres, descarga todos los archivos en una carpeta llamada *data* en tu directorio de trabajo. **No está permitido el uso de rutas absolutas**.

Ejercicio 2: leer archivos

Ahora cargaremos los archivos en la sesión. Recuerda que no necesariamente todos los archivos tienen el mismo separador.

- Crea una función llamada `read_esi_data` que lea un archivo. La función recibe como argumento la ruta del archivo (ej: data/esi-2018—personas.csv). `read_esi_data` debe ser capaz de reconocer el tipo de separador y leer el archivo correctamente en todos los casos. Para lograr esto existen varios caminos.

Dependiendo de las versiones de tus dependencias, es posible que tengas dificultades para usar readr::read_csv. Si tienes problemas, considera utilizar otras funciones para leer archivos

Ejercicio 3: obtener datos

Tu jefatura está interesada en conocer algunas características sobre las variables de diseño y sobre la variable principal de ingresos (*ing_t_p*). Para ello, te solicita lo siguiente:

- Tabla que contenga 3 columnas: *version*, *n_personas* (*idrph*) y *n_hogares* (*id_identificacion*). En la columna *version* debes usar la siguiente estructura: esi_{año}. Ejemplo: esi_2017
- Tabla que contenga mínimo, máximo, media, mediana, p10 y p90 del factor de expansión (*fact_cal_esi*) para cada versión. Debes considerar una fila por hogar (*id_identificacion*) e incluir la columna *version* ¿Se observan algunos pesos de muestreo atípicos?
- Tabla que contenga el número de estratos (*estrato*) con una sola unidad primaria de muestro (*conglomerado*). Debes incluir la columna *version*.
- Tabla que contenga mínimo, máximo, media, mediana, p10 y p90 de los ingresos del trabajo principal (*ing_t_p*) para cada versión. Esta tabla debe ser construida a nivel persona, utilizando el factor de expansión (*fact_cal_esi*).

Ejercicio 4: mejorando el código

Tu jefatura está muy satisfecha con los resultados obtenidos, pero está preocupada por el tiempo de ejecución y te pregunta si es posible crear un código más eficiente. Para ello, te solicita comparar el tiempo de ejecución de algunas estrategias. Utiliza el paquete `microbenchmark`, cuyo uso básico se describe aquí.

Nota: No es necesario que utilices más de **5 iteraciones** para calcular el tiempo de ejecución.

Calcula el promedio de ingresos en las tablas de la ESI (*ing_t_p*) mediante las siguientes estrategias:

1. Lista de tablas: calcular promedio con herramientas de `purrr` (como en el ejercicio anterior)

2. Tablas apiladas: calcular promedio con `group_by() %>% summarise()` (apila una tabla sobre otra en un dataframe)
3. Lista de tablas: calcular promedio con herramientas de `purrr`, utilizando una función creada por ti, que utilice `data.table`.
4. Tablas apiladas: calcular promedio con `data.table`

¿Existen diferencias importantes entre las distintas estrategias? ¿Hay alguna más eficiente que otra? ¿Usar `group_by` versus `map` hace alguna diferencia?

Luego de terminar, tu jefatura está tan complacida con tu trabajo que decide regalarte un pasaje a Cancún a un hotel all-inclusive :)