Dinámica de trabajo de los repartidos

A lo largo del curso y en cada unidad, usted deberá completar repartidos con los denominados **ejercicios de programación**. En estos se pide al estudiante que complete un archivo de texto plano (un script de R, y por lo tanto con la extensión `.R`) con el código (i.e.: instrucciones en lenguaje R) necesarias para realizar las tareas asignadas.

Para cada ejercicio usted podrá usar un sistema de corrección automático, desarrollado para este curso, el cual le permitirá saber de forma inmediata si ha completado correctamente las instrucciones del mismo. Para que este método funcione correctamente, hay que seguir ciertos procedimientos generales que pasaremos a describir a continuación.

Archivos incluidos:

Cada repartido consta de una carpeta con archivos que usted puede bajar desde la plataforma del curso (por ejemplo este). Aquí por supuesto se encuentran los scripts de R que usted **debe modificar**, así como ciertos archivos auxiliares que usted **no debe modificar** (usualmente: `evaluar.R`, `datos`, `notas.csv` e `INSTRUCCIONES.pdf`, aunque pueden variar). Estos archivos, particularmente los dos primeros, son esenciales para que funcione el método de corrección automática que hemos desarrollado para este curso.

La carpeta con el repartido debe bajarse y descomprimirse en disco duro, creando la carpeta `rep-X` (siendo X el número de repartido). Usted deberá abrir el RStudio y seleccionar dicha carpeta como su directorio de trabajo con `setwd` (o en RStudio la combinación **Ctrl + Shift + K**)

Escribiendo el código

Cada uno de los scripts del repartido se corresponde con un ejercicio. En cada archivo se indica con presición en dónde debe usted escribir su código (o modificar lo que ya está escrito). **Cuide siempre de no escribir ninguna línea de código ajena a los propósitos del ejercicio mismo**. Por ejemplo, debe tener cuidado de no dejar escritas líneas como `source(triangulo.R)` dentro del propio archivo `triangulo.R`, ya que esto genera problemas al momento de evaluar el ejercicio (básicamente es como pedirle al archivo que se evalúe a sí mismo, lo que hace que se vuelva a evaluar a sí mismo y así sigue un "circulo vicioso", hasta que se llega a los límites de procesamiento de la máquina). Por esta razón, recomendamos usar un script aparte para este tipo de comandos, así como todos aquellos comandos usados para experimentar con posibles soluciones o simples juegos de progamación que usted quiera hacer. En el video "Guía para repartidos" mostramos un ejemplo a seguir en caso de que esta explicación no le resulte satisfactoria.

Nótese además que los cambios que se hacen al script del ejercicio son **invisibles** para R hasta el momento en que usted **guarda** el archivo a disco duro. Esta suele ser una fuente común de frustración entre principiantes (y no tanto).

Qué tipos de soluciones *sirven*

Por ejemplo, para el segundo ejercicio del repartido 1 (para el cual se debe completar el código del script `areaMax.R`), no es válido el siguiente comando:

```
i <- 72
```

Si bien para el ejemplo que se ejecuta previo al ejercicio (i.e.: las líneas de comando que aparecen en `areaMax.R` antes del código que usted debe editar) esta solución es la correcta, *no es una solución* ***general*** para el ejercicio. Es decir, si cambiáramos el ejemplo, entonces 72 ya no sería una solución correcta, ya que el valor máximo dentro del vector `a` seguramente se encuentre ubicado en otra posición del mismo. El objetivo de estos ejercicios, es que usted desarrolle soluciones *universales* para el problema en cuestión. Encontrar soluciones universales es en donde radica el **poder de la programación** como herramienta, y por lo tanto es nuestro objetivo en el curso.

Por esta razón, si usted prueba usar 72 como se muestra arriba, el sistema de corrección automática lo dará como erróneo. En general para cualquier ejercicio, el sistema de corrección automático utiliza números aleatorios para verificar que su solución es robusta respecto a los casos particulares (i.e.: diferentes valores numéricos).

Objetos nombrados

En varios ejercicios del curso usted se encontrarán con objetos en R que contienen nombres. Por ejemplo, las columnas de una matriz de datos (`data.frame` es el término correcto en R) tienen nombres. Parte de la corrección automática consiste en verificar que estos nombres estén correctos. Esto incluye: el **orden** en el que están los nombres así como la **capitalización** de las letras en los nombres (i.e.: mayúsculas y minúsculas). Por esto debe uster prestar atención a estos detalles cada vez que realice una corrección automática de los ejercicios.

Mecanismo de corrección automática:

Lo primero que debe hacer es cargar el archivo `evaluar.R` con la función `source`, como se muestra a continuación:

```
options(encoding = "utf-8")
source("evaluar.R")
```

Ejecutado correctamente, la siguiente frase debería verse en la consola:

```
Archivo de codigo fuente cargado correctamente
```

En caso de que ocurra un error o se vea otro mensaje en la consola, verifique que los archivos se descomprimieron correctamente y que usted está trabajando en la carpeta correspondiente con el comando `getwd()`.

Usted trabajará modificando los contenidos de dichos archivos con RStudio (u otro programa de su preferencia) según las consignas que se describen a continuación. Luego de terminar cada ejercicio y **guardando el archivo** correspondiente en el disco duro, usted podrá verificar rápidamente si su respuesta es correcta ejecutando el comando:

```
evaluar()
```

y además podrá en todo momento verificar su puntaje con la función `verNotas()`. Una vez terminados los ejercicios del repartido, usted deberá subir el archivo "datos" (sin extensión), incluido en la carpeta "rep-1", a la sección de entregas de la portada del curso en la plataforma EVA.

Código de Honor

Si bien animamos a que los estudiantes trabajen en equipos y que haya un intercambio fluido en los foros del curso, es fundamental que las respuestas a los cuestionarios y ejercicios de programación sean fruto del trabajo individual. En particular, consideramos importante que los estudiantes no miren el código creado por sus compañeros ya que esto supone un sabotaje a su propio proceso de aprendizaje. Como profesores estamos comprometidos a pedir tareas para las cuales hayamos dado las herramientas correctas y las explicaciones adecuadas como para que todos puedan encontrar su propio camino para resolver los ejercicios.