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.

Este documento puede considerarse como una contraparte escrita al video "Guía para repartidos".

ADVERTENCIA: leer este documento puede ahorrar *muchos* dolores de cabeza a la hora de resolver los ejercicios del curso.

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 estos archivos se corresponde con un ejercicio del repartido. Se trata de archivos de texto plano (equivalentes a un `.txt`, pero con una extensión diferente que sirve para que los programas que lo abren reconozcan la sintaxis). En cada archivo se indica con presición en dónde debe usted escribir su código (o modificar lo que ya está escrito). Para que el sistema de corrección funcione bien, debe evitar cambiar el texto de todas las líneas de texto del archivo ajenas a este espacio, incluyendo las que se usan para indicar el inicio y el final del mismo. Nótese además que esto no impide **ampliar o reducir** el espacio que

usted tiene para escribir, subiendo o bajando cualquiera de las "líneas límite" del archivo.

Cuide siempre de no escribir en estos archivos 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).

Problemas de redondeo/aproximación numérica

Ocasionalmente puede ocurrir que una solución que plantea un estudiante tenga diferencias ínfimas con los valores esperados y resultan en ejercicios corregidos como incompletos. Estas diferencias pueden estar en el orden de 10^{-99}, por lo que a simple vista no se pueden ver (R nos muestra valores redondeados al 5 decimal por defecto). El origen de estas divergencias es la necesidad de la computadora de redondear los números, debido a que tiene capacidades finitas y los números pueden tener infinitos decimales.

En general tratamos de que esto no sea un problema para la corrección de ejercicios, pero no puede descartarse que en algún caso hallamos pasado un problema así sin verlo. Si está *convencido* de que esto es así, por favor comuníquelo a través de los foros bajo la etiqueta "fe de erratas".

Debe notarse además que estas diferencias pueden además surgir en base a la forma en que uno escribe las ecuaciones, sobre todo en el caso de que se usen operaciones que "amplifiquen" dichas diferencias, como puede ser la exponenciación. El siguiente es un ejemplo concreto:

```
a <- (1/7)^100
b <- (1/(7^100))
```

Es claro que los valores de `a` y `b` deberían ser idénticos, ya que se obtuvieron con expresiones matemáticamente equivalentes (i.e.: $(1/7)^{100}$ y $1/(7^{100})$). Sin embargo cuando buscamos verificar esto, nos sorprendemos:

```
a == b
```

```
## [1] FALSE
```

```
a - b
```

```
## [1] -1.714e-99
```

Aquí la evaluación de `a == b` debería dar `TRUE` y no `FALSE`, y la última línea debería devolver un cero. Lo que ocurre es que el orden en el que se realizan las operaciones **sí** altera el producto en este caso, a causa de los problemas de redondeo que antes mencionábamos. Nótese que para crear `a`, *primero* se calcula 1/7 (lo que resulta en un valor redondeado) y *luego* se eleva el resultado a la 100. Contrariamente, para el caso de `b`, R primero eleva 7 a la 100 (lo que debería dar un resultado exacto), y recién entonces evalúa la división (aquí es otra vez un valor redondeado).

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.

Evaluaciones de métodos

En algunos ejercicios (la minoría) se pide que el estudiante resuelva un problema utilizando un método en particular (por ejemplo, usando el operador `%*%` en el 3.a del repartido 1). En estos casos el sistema de evaluación sólo dará como correcto al ejercicio si el usuario usa efectivamente el método en cuestión (i.e.: utiliza el operador `%*%` correctamente en la línea correspondiente).

Se evalúan varios objetos

En muchos casos el algoritmo de evaluación no se limita a evaluar si el "objeto final" de un script está correcto, si no que evalúa también objetos (y/o pasos) intermedios. Por ejemplo, en el ejercicio 1.b del repartido 1, la corrección evalúa los tres objetos, `i`, `sol` y `amax`.

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-X", a la sección de entregas de la portada del curso en la plataforma EVA.

Sistema de puntaje

Cada ejercicio en un repartido vale un punto, sin valores intermedios. Es decir, el resultado es binario: 0 o 1 punto por ejercicio. Los repartidos tienen ejercicios **obligatorios** pero también ejercicios *opcionales*. El puntaje total de cada repartido se calcula como el porcentaje de puntos obtenidos respecto al total de ejercicios *obligatorios* del mismo. Por lo tanto, es posible obtener notas mayores al 100%. Por ejemplo: si hay 6 ejercicios obligatorios y 2 opcionales, entonces 6 puntos equivalen a un 100% y 8 puntos a 133% ($nota = (puntos/6) \cdot 100$).

Nótese que de todas formas, en la nota final del curso no se permitirán porcentajes superiores al 100%, en acordancia con el sistema de notas de la UdelaR. Este sistema es simplemente una forma de motivar y poder subir el promedio general de cada estudiante.

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.