Ejercicios de programación IV: Funciones

[IMSER 2012]

Archivos incluidos:

El archivo con los ejercicios del práctico debe bajarse y descomprimirse en disco duro, creando la carpeta `rep-2` (nota: no debe dentro de ningún disco, partición o carpeta protegida a la escritura, como puede ser un disco duro externo de backup). Usted deberá abrir el RStudio y seleccionar dicha carpeta como su directorio de trabajo con `setwd` o en RStudio la combinación **Ctrl + Shift + K**. En esta carpeta se encuentran algunos archivos que usted deberá modificar:

- `triangulo.R`
- `educacion.R`
- `cambiaPares.R`
- `radio.R`
- `distancias.R`

Adicionalmente los siguientes archivos son necesarios, pero **no deben ser modificados** para que el método de calificación automático funcione correctamente:

- `datos`
- `evaluar`
- `notas.csv`
- `edu.data.rda`
- `HandbookSpanish.pdf`
- `INSTRUCCIONES-caca.pdf`

Mecanismo de corrección:

Lo primero que debe hacer es cargar el archivo evaluar.R con la función `source` y la codificación de caracteres "UTF-8" (lo cual afecta a la función `evaluar` en particular), de la siguiente manera:

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

Si usted ha ejecutado todos los pasos anteriores 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()`. Tenga siempre en cuenta que, a **menos que sea indicado** por la letra del ejercicio, las soluciones deben ser genéricas y por lo tanto deben obtenerse con el código de los scripts en lugar de ser valores fijos. Usualmente se utilizan valores generados de forma aleatoria para las correcciones automáticas. Los objetos que son evaluados en la corrección automática estarán indicados con un asterísco en las instrucciones de cada script. Nótese además que en los archivos **se indica claramente en dónde se inicia y dónde finaliza su código** y que debe respetar esta organización para que la corrección de los ejercicios funcione bien.

Al finalizar

Una vez terminados y guardados los archivos de los ejercicios del repartido, usted deberá ejecutar `evaluar()` y seleccionar la última opción ("Todos") y luego 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. Este archivo se podrá reemplazar con uno más nuevo, en caso de que desee corregir algún error; en caso de querer que el archivo sea corregido antes de la fecha de entrega, puede cambiarle el nombre a "datos-finalizado", pero en ese caso la nota no se cambiará de ahí en adelante.

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.

---

1. Triángulos, volumen II

Script: triangulo.R

Si hacemos un poco de memoria recordaremos el ejercicio 1 del repartido I. En el mismo tomábamos los catetos de un triángulo rectángulo y escribíamos el código necesario para calcular el valor de la hipotenusa y el área del mismo. Como recordaremos, el valor de la hipotenusa se calcula como:

$$hip = \sqrt{cat.op^2 + cat.ad^2}$$

mientras que el área del triángulo es:

$$A = \frac{cat.op \cdot cat.ad}{2}$$

A partir de los valores de los catetos y de la hipotenusa es posible calcular el valor de los restantes ángulos del triángulo, siendo el ángulo adyacente

$$\alpha_{ad} = arccos\left(\frac{cat.ad}{hip}\right)$$

y el ángulo opuesto

$$\alpha_{op} = arccos\left(\frac{cat.op}{hip}\right)$$

En el presente ejercicio debemos escribir el código de una función llamada `triangulo` que, a partir de los mismos datos de las funciones `area` e `hipot` de aquella ocasión, o sea de los catetos de un triángulo rectángulo, calcule la

hipotenusa, el área y los ángulos adyacente y opuesto del triángulo. La salida de la función deberá ser una lista con los objetos denominados de la siguiente manera: `hipotenusa`, `area`, `angulo.adyacente` y `angulo.opuesto`, en ese orden. Los ángulos deberán estar expresados en grados. En R, la salida de las funciones trigonométricas, como `asin` y `acos` están expresadas en radianes, por lo que se deberá hacer la transformación correspondiente, teniendo en cuenta que π radianes equivalen a 180^o.

A modo de ejemplo, la salida deseada al evaluar esta función con el cateto adyacente y el opuesto valiendo respectivamente 4 y 3, sería la siguiente:

```
triangulo(4, 3)
$hipotenusa
[1] 5

$area
[1] 6

$angulo.adyacente
[1] 36.8699

$angulo.opuesto
[1] 53.1301
```

---

2. Educación

Script: educacion.R

La Organización de la Naciones Unidas (ONU) en el año 2011 presentó un informe basado en indicadores utilizados para medir y supervisar los Objetivos de Desarrollo del Milenio (ODM).

> "La Declaración del Milenio de las Naciones Unidas de 2000, basada en las conferencias mundiales de las Naciones Unidas durante el decenio de 1990, representó un fuerte compromiso con el derecho al desarrollo, la paz y la seguridad, la igualdad de género, la erradicación de las numerosas dimensiones de la pobreza y el desarrollo humano sostenible. En la Declaración, adoptada por 147 jefes de Estado y 189 Estados, se incorporaban lo que ha llegado a conocerse con el nombre de 'ocho objetivos de desarrollo del milenio', incluidas 18 metas con plazos cronológicos delimitados" (Fuente).

Para este ejercicio se utilizarán tres de los indicadores que han sido propuestos para cumplir con el objetivo 2 de los ODM Lograr la enseñanza primaria universal:

- Tasa Neta de Matriculación en la Enseñanza Primaria (TM).
- Porcentaje de alumnos que comienzan el primer grado y llegan al quinto grado (PA).
- Tasa de alfabetización de personas entre 15 y 24 años (TA).

Las tres variables están expresadas en porcentaje. El objetivo es realizar un cálculo grueso del porcentaje de niños que completan la educación primaria (PC) para ciertos paises, a partir de TA y PA, usando la fórmula: $PC = (TM \cdot PA)/100$. Finalmente se hará una regresión entre la tasa de alfabetización y el porcentaje de conclusión de primaria calculado.

El archivo "edu.data.rda" contiene una data.frame llamada `edu.data` con datos de varios años para ciertos países, usando las abreviaciones de arriba para cada indicador (importar a R con `load` o los botones de RStudio). Los datos fueron extraidos de la portada oficial de la ONU para los indicadores de los OMD.

Entradas: ud. deberá crear una función llamada `educación`, la cual deberá aceptar los siguientes argumentos, en este orden (si quiere puede asignar valores por defecto a uno o a varios de estos argumentos):

- `x`: data.frame con los datos de entrada (`edu.data`).
- `tmcol`: columnas correspondientes al indicador TM (en números o nombres, debería ser irrelevante).
- `pacol`: columnas correspondientes al indicador PA (en números o nombres, debería ser irrelevante).
- `tacol`: columna correspondiente al indicador TA (en número o nombre, debería ser irrelevante).

Acciones: la función deberá realizar los siguiente pasos:

1. Calcular, por país/fila, el promedio de los valores de TM y de PA.
2. Calcular con estos el valor promedio de PC para cada país.
3. Agregar estos tres valores calculados como columnas a la data.frame de entrada (`x`).
4. Realizar una regresión lineal de TA en función de PC.

5. Devolver los valores calculados y los datos más importantes de la regresión.

Nota: los cálculos de promedios deben hacerse dejando de lado los valores `NA`.

Salida: `educacion` debe devolver un objeto de clase `list`, el cual debe contener los siguientes objetos (y en el mismo orden):

- `coeficientes`: los coeficientes de la regresión de TA en función de PC.
- `p.valor`: el p.valor de la regresión correspondiente a los coeficientes (ver salida de `summary` para obtenerlos).
- `r2`: el coeficiente de determinación (R^2) de la regresión (el cual puede calcularse usando una fórmula o la función `summary`).
- `datos`: data.frame con los datos originales más las columnas correspondientes a los promedios de TM, PA y PC, en ese orden.

Nota: la función `summary` además de imprimir en la consola datos útiles de una regresión, también genera una **salida invisible** (ver lección 5.3).

El siguiente es un ejemplo obtenido con los datos y una función `educacion` completada (la figura incluida también se generó con esta función, pero para la tarea es opcional):

```
load("edu.data.rda")
x <- educacion(edu.data, 1:5, 6:10, 11)
```

```
x$coeficientes
```

```
## (Intercept)          PC
##     75.0948      0.2679
```

```
x$p.valor
```

```
## (Intercept)          PC
##   2.329e-10   6.259e-05
```

```
x$r2
```

```
## [1] 0.7803
```

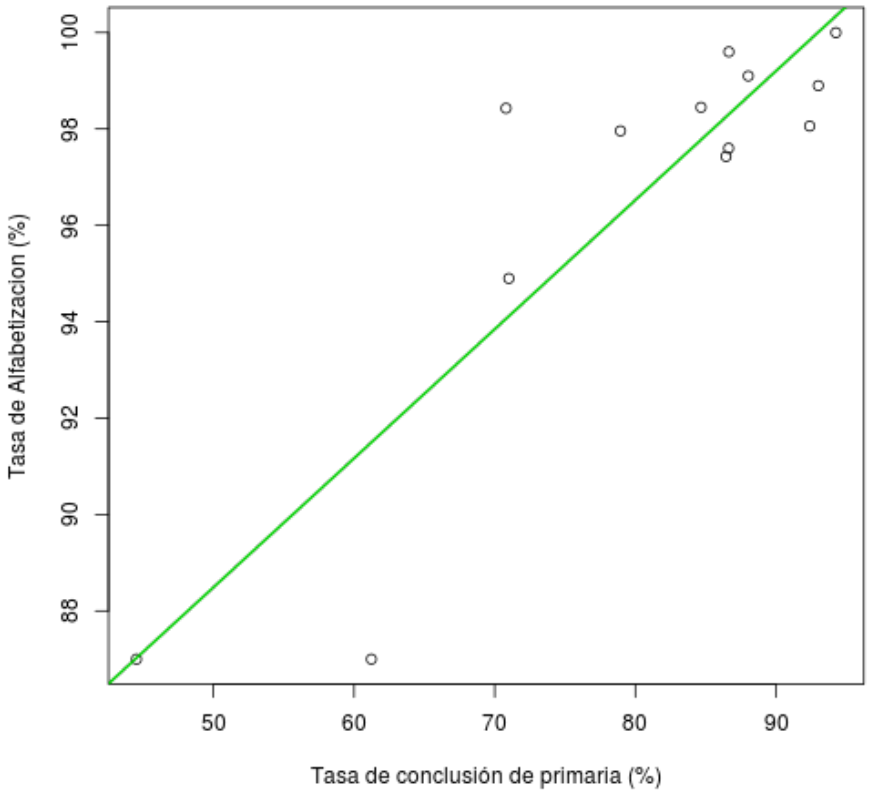

Figure 1: regresión de TA en función de PC (salida gráfica opcional de la función educacion)

3. Funciones en problemas

En este ejercicio se propone arreglar el código de algunas funciones simples (y otra no tan simple). Para esto la idea es utilizar los métodos y conceptos vertidos en la lección 5.4: **Depuración de funciones**.

3.a Cambiador de valores en subíndices pares...

Script: cambiaPares.R

Esta sencilla función toma un vector `x` y cambia aquellos valores que se encuentran en las posiciones pares del mismo (`x[2]`, `x[4]`, etc.), usando como sustitutos los elementos del vector `subs` (el segundo argumento). Un ejemplo de salida de esta función puede ser el siguiente:

```
cambiaPares(1:6, NA)
```

```
## [1]  1 NA  3 NA  5 NA
```

El objetivo de este ejercicio es arreglar el código de la función contenida en el script asociado de forma tal que ejecute correctamente su tarea.

3.b Radios

Script: radio.R

La función radio toma como argumento el valor `r`, un número cualquiera, y calcula tres valores asociados con circunferencias y esferas: el perímetro (P), área (A; de la circunferencia) y volumen (V; de la esfera), utilizando las fórmulas:

$$P = 2 \cdot \pi \cdot r$$
$$A = \pi \cdot r^2$$
$$V = \frac{4 \cdot \pi \cdot r^3}{3}$$

La función está pensada para generar una salida invisible, al mismo tiempo que imprimir en la consola los resultados obtenidos (nótese el uso de la función `cat` para este cometido). El siguiente es un ejemplo de salida de la función:

```
x <- radio(5)
```

```
## Perímetro: 31.42
## Área:      78.54
## Volumen:   523.6
```

```
x
```

```
##      P      A      V
##  31.42  78.54 523.60
```

El objetivo de este ejercicio es arreglar el código de la función contenida en el script asociado de forma tal que ejecute correctamente su tarea.

3.c Extra: encuentra distancias

Script: distancias.R

La función `distancias`, escrita en el archivo "distancias.R", busca realizar una tarea parecida a la que ya se hizo en el Repartido I del curso: calcular las distancias de un punto a un conjunto de coordenadas (ej. 1.c). En este caso se utiliza un algoritmo alternativo para calcular las distancias, el cual es más robusto dadas las limitaciones de las computadoras.

El significado de los argumentos de la función están explicados en el propio archivo, así como varios de los pasos internos, a través del uso de comentarios. Al igual que en ejercicios anteriores, el objetivo es corregir los errores que tiene el archivo para que la función `distancias` cumpla su tarea correctamente. El siguiente es un ejemplo de cómo debería ser la salida de la función (incluyendo la figura):

```
pts <- matrix(rnorm(20), ncol = 2)
x <- distancias(pts, p = c(0.3, -0.1))
```

```
## d.max = 3.26 - punto: -0.25 3.12
## d.min = 0.65 - punto: 0.93 -0.26
```

```
x
```

```
## $dists
##  d.max  d.min
## 3.2629 0.6493
##
## $posiciones
## i.max i.min
##     5    10
##
## $puntos
##              x       y
## d.max -0.2500  3.1162
```

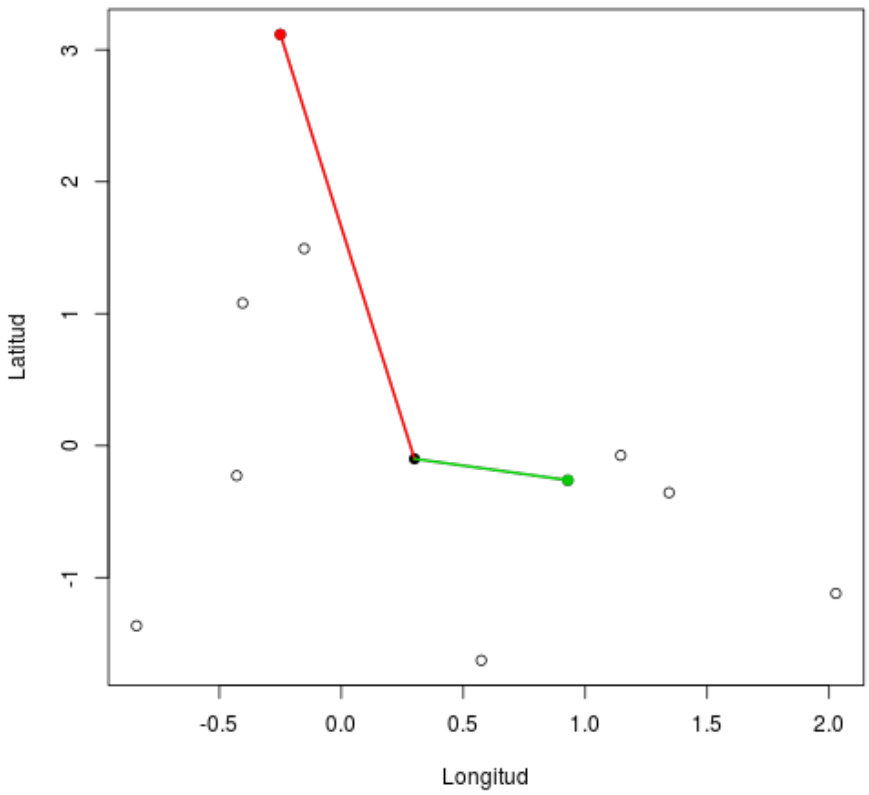

Figure 2: salida gráfica de la función distancias

```
## d.min  0.9289 -0.2618
##
## $centro
## [1]  0.3 -0.1
##
```