Material Extra de POST

Estructura tu código científico

Un ejemplo de cómo estructurar tu código

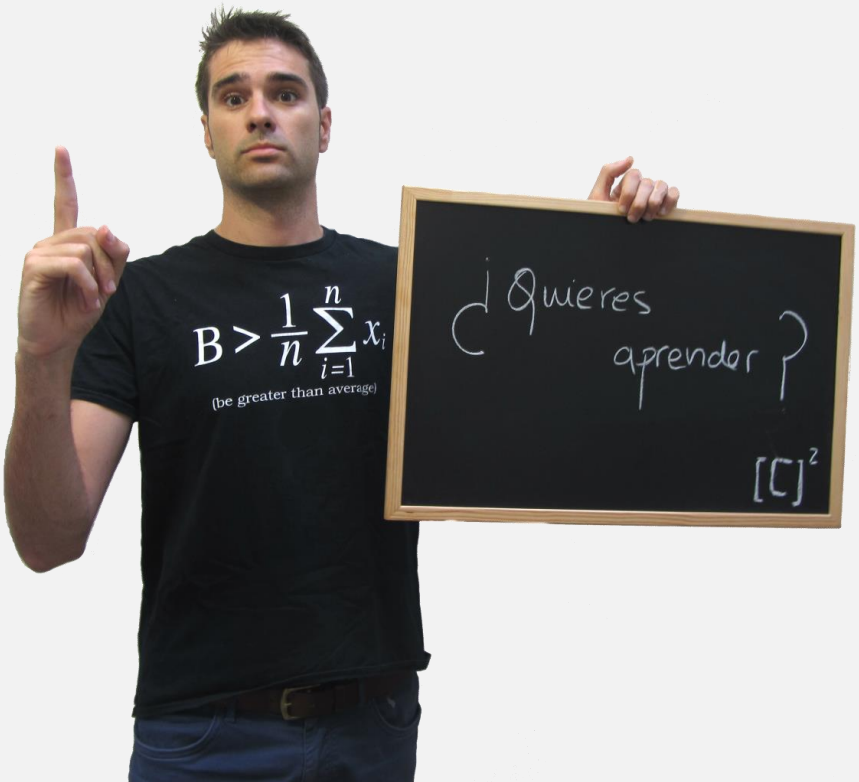
```
#*********************************************************************************
#*********************************************************************************
#            MATERIAL DESCARGABLE DE CONCEPTOS CLAROS
#*********************************************************************************
#*********************************************************************************
# Autor: Jordi Ollé Sánchez
# Fecha: 27/09/2016
# E-mail: jordi@conceptosclaros.com
# Explicación: Este código te permite ver la estructura de un script
# Tiene en cuenta la estructura siguiente:

# ESTRUCTURA DEL CÓDIGO
#*********************************************************************************
# 1.1 INSTALAR PAQUETES DE FUNCIONES
# 1.2 CARGAR PAQUETES O CREAR FUNCIONES
# 2. IMPORTAR/LEER DATOS
# 3. VARIABLES DE ENTRADA
# 4. METODOLOGÍA
# 5. EXPORTAR RESULTADOS

# ¿QUÉ HACE ESTE CÓDIGO?
#*********************************************************************************
# Este código te permite leer un archivo csv separado por ;
# Te permite escojer una de las variables numéricas de la tabla de datos
# Calcula la tabla de frecuencias y dibuja el histograma según un número
# de BINS especificado.

# INPUTS DISPONIBLES
#*********************************************************************************
#   varNum: la columna de la tabla de datos que quieres leer
#   Nbins: el número de barras de tu histograma

# OUTPUTS
#*********************************************************************************
#   "Tabla Frecuencias.xlsx": tabla de frecuencias en formato Excel
```



Estructura tu código científico

Este es un material extra para completar la información del POST: "Los 5 puntos necesarios que todo código científico debe tener". Vas a encontrar un código de ejemplo en R dónde te ejemplifico con más detalle la estructura de un código científico o "script".

Vemos el ejemplo: [Post]-Estructura Codigo Cientifico.R

Los puntos del código son:

0. **Encabezado**
1. **Instalar paquetes de funciones**
2. **Cargar paquetes de funciones o crear funciones**
3. **Variables de entrada**
4. **Metodología de procesado de datos (Estadística, mates, data analytics,…)**
5. **Exportar resultados**

Punto 0: Encabezado

En el encabezado te he puesto:

A. **Datos característicos del código:**
   - El autor
   - La fecha
   - El contacto
   - Mini explicación de cuál es el **propósito del código.**

     ¡¡¡SUPER IMPORTANTE PENSAR EN EL PROPÓSITO DEL CÓDIGO!!!

B. **¿Qué hace este código?** Aquí explico que hace el código. Esto es importante para poder recordar en un futuro que ~~coño~~ hacia el código.

C. **Inputs disponibles:** las variables inputs del código y para qué sirven. Son los únicos parámetros que deberías cambiar.

D. **Outputs:** los outputs que va a generar el script.

Aquí el encabezado que vas a encontrar:

```
#******************************************************************
#******************************************************************
#          MATERIAL DESCARGABLE DE CONCEPTOS CLAROS
#******************************************************************
#******************************************************************
# Autor: Jordi Ollé Sánchez
# Fecha: 27/09/2016
```

```
#                 E-mail: jordi@conceptosclaros.com
# Explicación: Este código te permite ver la estructura de un script
# Tiene en cuenta la estructura siguiente:

# ESTRUCTURA DEL CÓDIGO
#***********************************************************************
# 1.1 INSTALAR PAQUETES DE FUNCIONES
# 1.2 CARGAR PAQUETES O CREAR FUNCIONES / INICIALIZACIÓN
# 2. IMPORTAR/LEER DATOS
# 3. VARIABLES DE ENTRADA
# 4. METODOLOGÍA
# 5. EXPORTAR RESULTADOS

# ¿QUÉ HACE ESTE CÓDIGO?
#***********************************************************************
# Este código te permite leer un archivo csv separado por ;
# Te permite escojer una de las variables numéricas de la tabla de datos
# Calcula la tabla de frecuencias y dibuja el histograma según un número
# de BINS especificado.

# INPUTS DISPONIBLES
#***********************************************************************
#   varNum: la columna de la tabla de datos que quieres leer
#   Nbins: el número de barras de tu histograma

# OUTPUTS
#***********************************************************************
#   "Tabla Frecuencias.xlsx": tabla de frecuencias en formato Excel
#   Plots de los histogramas en RStudio
```

Punto 1.1: Instalar paquetes de funciones

En este punto debemos instalas los paquetes de funciones sino los tenemos.
En R es necesario verificar si tenemos instalados los paquetes. Si no es así se instalar automáticamente.
Los paquetes de funciones utilizados para este script son: ggplot2, plotly, xlsx, scales.

En caso de otros softwares cómo Matlab o Excel eres tu quien debe instalar previamente los paquetes de funciones antes de crear el código. Deberás configurarlos y probarlos.

```
#***********************************************************************
# 1.1 INSTALAR PAQUETES DE FUNCIONES
#***********************************************************************
# Lista de paquetes de funciones a instalar
.packages = c("ggplot2", "plotly", "xlsx","scales")

# Instala los paquetes sinó los tienes instalados
.inst <- .packages %in% installed.packages()
if(length(.packages[!.inst]) > 0) install.packages(.packages[!.inst])
```

Punto 1.2: Crear paquetes o crear funciones / inicialización de variables

Una vez tienes los paquetes de funciones instalados hay que pasar a cargarlos o configurarlos.

En caso de R usaremos estas líneas para cargar los paquetes de la variable .packages.

```
# Carga los paquetes sinó los tienes cargados
lapply(.packages, require, character.only=TRUE)
```

Para Matlab o Excel normalmente este proceso es automático cuando instalas paquetes de funciones.

Una vez tienes las funciones instaladas y cargadas es momento de inicializar las variables que vas a utilizar.

Normalmente se utiliza matrices o vectores de zeros para inicializar variables numéricas.

En este apartado también es momento de crear funciones DIY que después vas a utilizar.

En Matlab puedes crear funciones y después llamarlas sin problema.

En R también lo puedes hacer cómo yo lo he hecho aquí:

```
#**********************************************************************
# 1.2 CARGAR PAQUETES O CREAR FUNCIONES
#**********************************************************************
# Carga los paquetes sinó los tienes cargados
lapply(.packages, require, character.only=TRUE)

# Funciones "DIY"
# Función para encontrar la moda
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

Punto 2: Importar/leer datos

Es momento de leer/importar o incluso crearte tus propios datos de entrada a tu particular fábrica (tu código).

Con una simple línea puedes leer un archivo .csv con valors separados por punto y coma.

```
#**********************************************************************
# 2. IMPORTAR/LEER DATOS
#**********************************************************************
# Recuerda que debes configurar antes tu working directory ;) y asegúrate que apunta al fichero Excel
mydata <- read.table("Tabla de Datos Ejemplo.csv", header=TRUE,sep=";")
```

Punto 3: Variables de entrada

Tú eres quien decide cuáles y cuántas variables de entrada tendrá tu código y qué función realizará cada una.

Por ejemplo en mi caso he escogido dos variables:

```
#**********************************************************************
# 3. VARIABLES DE ENTRADA
#**********************************************************************
# Input1: #selecciona la variable numérica que quieras, por ejemplo la número 3: "Edad"
varNum <- 3
# Input2: #selecciona el número de clases que quieres obtener
Nbins <- 20
```

Punto 4: Metodología de procesado de datos

Tu gran momento. Demuestra tus conocimientos en mates, estadística, "data analytics" etc... Demuestra quién eres. Es momento de fabricar, transformar tu materia prima (datos) en algo útil para ti (datos transformados) y responder tus preguntas o solucionar lo que querías. Muy importante que tú código tenga un objetivo claro que habrás escrito en el encabezado (**TE LO HE REMARCADO EN ROJO AL PRINCIPIO**), y que los datos que transformes sirvan para el propósito u objetivo del código.

Te dejo con la metodología de este ejemplo:

Estoy transformando los datos leídos en una tabla de frecuencias clasificándolos según un número de clases o BINS (variable de entrada Nbins) para después dibujar un histograma. Si quieres saber más sobre el histograma aquí un post muy interesante.

```
#**********************************************************************
# 4. METODOLOGÍA
#**********************************************************************
# Preparamos los inputs para ponerlos en los plots
x <- mydata[,varNum] #guardamos la variable x (variable numérica)
binwidth <- (max(x)-min(x))/(Nbins-1) #el ancho de las clases

# Histograma utilizando ggplot para obtener los datos para dibujar el histograma
histograma <- ggplot(mydata,aes(x=mydata[,varNum])) +
  geom_histogram(binwidth = binwidth,colour="black", fill="#e2746a") +
  xlab(names(mydata)[varNum]) + ylab("Frecuencia absoluta (nº de repeticiones)") +
  ggtitle(paste("Histograma de la variable ", names(mydata)[varNum],sep = "")) +
  theme_minimal() # creamos el histograma y nos lo guardamos en la variable "histograma"

pg <- ggplot_build(histograma) #guardamos los datos del histograma (la tabla de frecuencias)

# Calculamos la tabla de frecuencias
valoresMinClase <- pg$data[[1]]$xmin
valoresMaxClase <- pg$data[[1]]$xmax
valoresMedioClase <- pg$data[[1]]$x
frecuenciaAbsoluta <- pg$data[[1]]$count
frecuenciaAbsolutaAcumulada <- cumsum(frecuenciaAbsoluta)
frecuenciaRelativa <- pg$data[[1]]$count/sum(pg$data[[1]]$count)
frecuenciaRelativaAcumulada <- cumsum(frecuenciaRelativa)
tablaFrecuencias <- data.frame(Valores.Min.Clase = valoresMinClase,
                               Valores.Max.Clase = valoresMaxClase,
                               Valores.Medio.Clase = valoresMedioClase,
                               Frecuencia.Absoluta = frecuenciaAbsoluta,
                               Frecuencia.Absoluta.Acumulada = frecuenciaAbsolutaAcumulada,
                               Frecuencia.Relativa = frecuenciaRelativa,
                               Frecuencia.Relativa.Acumulada = frecuenciaRelativaAcumulada)
```

Punto 5: Exportar datos/ crear gráficos

Finalmente, el momento de poner relucir tu trabajo. El momento esperado. Puedes hacer dos cosas:

Guardar tus datos en el formato que mejor te vaya. (En este ejemplo guardo en un fichero Excel la tabla de frecuencias).

Crear los gráficos que mejor te vayan para decir lo que quieres decir. En este ejemplo he creado una serie de histogramas con distintas funciones de R:

```
#**********************************************************************
# 5. EXPORTAR RESULTADOS
#**********************************************************************
# Plot con la función base hist
hist(x) #no especificamos nº de clases, por defecto 10
hist(x,Nbins) #especificamos nº de clases = Nbins

# Histograma utilizando ggplot
histograma + scale_x_continuous(breaks=round(seq(min(pg$data[[1]]$xmin),max(pg$data[[1]]$xmax),by=binwidth),digits = 1))
#corregimos los valores de las clases en el eje horizontal

# Histograma con plotly
ggplotly(histograma +
scale_x_continuous(breaks=round(seq(min(pg$data[[1]]$xmin),max(pg$data[[1]]$xmax),by=binwidth),digits = 2))) # convertimos
ggplot to plotly

# Exportamos la tabla de frecuencias a un excel
library(xlsx)
write.xlsx(tablaFrecuencias, "Tabla Frecuencias.xlsx")
```

El histograma output con plotly de R ☺