Package: UnalR (via r-universe)

September 2, 2024

Type Package

Title Una implementación de funciones de uso interno

Version 1.0.0

Description Una herramienta rápida y consistente para la disposición de microdatos y la visualización de las cifras y estadísticas oficiales de la Universidad Nacional de Colombia <https://unal.edu.co>. Contiene una biblioteca de funciones gráficas, tanto estáticas como interactivas, que ofrece numerosos tipos de gráficos con una sintaxis altamente configurable y simple. Entre estos encontramos la visualización de tablas HTML, series, gráficos de barras y circulares, mapas, etc. Todo lo anterior apoyado en bibliotecas de JavaScript. English: A fast and consistent tool for the arrangement of microdata and the visualization of official figures and statistics from the National University of Colombia <https://unal.edu.co>. It includes a library of graphical functions, both static and interactive, offering numerous types of charts with a highly configurable and simple syntax. Among these, we find the visualization of HTML tables, series, bar and pie charts, maps, etc. It provides the capability to transition from the interactive to the dynamic world and from one library to another without changing function or syntax.

License GPL (>= 3)

URL https://unal.edu.co, https://estadisticas.unal.edu.co/home/, https://github.com/estadisticaun/UnalR

BugReports https://github.com/estadisticaun/UnalR/issues

Imports d3r (>= 1.1.0), dplyr (>= 1.1.4), DT (>= 0.33), dygraphs (>= 1.1.1.6), echarts4r (>= 0.4.5), fmsb (>= 0.7.6), forcats (>= 1.0.0), ggplot2 (>= 3.5.1), gt (>= 0.10.1), highcharter (>= 0.9.4), leaflet (>= 2.2.2), leaflet.extras (>= 1.0.0), plotly (>= 4.10.4), rlang (>= 1.1.3), sunburstR (>= 2.1.8), tidyr (>= 1.3.1), treemap (>= 2.4.4), cli, data.tree, ggspatial, ggrepel, gtExtras, htmltools, jsonlite, lifecycle, magrittr, maps,

scales, sf, sp, stringr, xts, zoo, grDevices, utils, methods, htmlwidgets, webshot, png, grid, gridSVG, XML

Suggests testthat (>= 3.2.1.1), cowplot, ggthemes, magick, pals, readxl, rsvg, tibble, viridis

Depends R (>= 4.3.0)

Encoding UTF-8

Language es-ES

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Config/testthat/edition 3

Config/testthat/parallel true

Repository https://estadisticaun.r-universe.dev

RemoteUrl https://github.com/estadisticaun/unalr

RemoteRef HEAD

RemoteSha 4facc18242749eb93062dc924e879da7bd930179

Contents

`Agregar` *Cree el consolidado a partir de los microdatos con los que dispone, agrupando por las variables temporales que considere necesarias*

Description

Esta función permite transformar desde la base de microdatos para obtener el consolidado final con el cual trabaja la mayoría de las funciones gráficas disponibles en `UnalR`.

Usage

```
Agregar(
  datos,
  formula,
  frecuencia,
  intervalo,
  textNA = "Sin Información",
  ask = FALSE
)
```

Arguments

`datos`	Un data frame con los microdatos.
`formula`	Fórmula en la que el primer componente especificado (*antes del* $\sim$) hace referencia a la(s) variable(s) de interés, y el segundo componente (*luego del* $\sim$) a la(s) variable(s) temporales por las cuales se quiere agrupar (*separadas por cualquier operador como* `+`, `*`, `:`, *etc.*).
`frecuencia`	Vector o lista (*dependiendo de la cantidad de variables temporales especificadas*) numérica con los periodos que debería tener cada una de éstas.
`intervalo`	Vector o lista (*dependiendo de la cantidad de variables temporales especificadas*) numérica que contiene los períodos de inicio y fin con los cuales se quiere realizar el filtro.
`textNA`	Cadena de caracteres indicando el nombre que se dará a los registros cuando éstos presenten algún dato faltante (*en la variable seleccionada más no en las variables temporales especificadas*). El valor por defecto es "Sin Información".
`ask`	Si es `TRUE` (*valor predeterminado*) mostrará un mensaje por consola preguntándole al usuario cuál periodo desea que sea el último por considerar (*esperando una respuesta por consola*). Si previamente se ha introducido el argumento `intervalo` éste quedará inhabilitado y no se ejecutará.

Value

Un data frame o tibble perteneciente a las clases "tbl_df", "tbl" y "data.frame".

Examples

```
# library(readxl)
df1 <- read_excel(read_example("TestConsolidado1.xlsx"))
df2 <- read_excel(read_example("TestConsolidado2.xlsx"))

# Seleccione para cada dataset (A, B, C y D) una de las opciones que se le muestran
# en consola (1, 2, 3 y 4) respectivamente.
Agregar(
  formula = TIPO_NIVEL ~ ANO + PERIODO,
  frecuencia = list("Year" = 2009:2013, "Period" = 1:3), df1
) -> A
Agregar(
  formula = TIPO_NIVEL ~ ANO + PERIODO,
  frecuencia = list("Year" = 2009:2013, "Period" = 1:3), df1
) -> B
Agregar(
  formula = TIPO_NIVEL ~ ANO + PERIODO,
  frecuencia = list("Year" = 2009:2013, "Period" = 1:3), df1
) -> C
Agregar(
  formula = TIPO_NIVEL ~ ANO + PERIODO,
  frecuencia = list("Year" = 2009:2013, "Period" = 1:3), df1
) -> D
all.equal(C, D)

G <- Agregar(
  formula    = TIPO_NIVEL ~ ANO + PERIODO,
  frecuencia = list("Year" = 2009:2013, "Period" = 1:2),
  intervalo  = list(c(2009, 1), c(2013, 1)),
  datos      = df1
)
H <- Agregar(
  formula    = TIPO_NIVEL ~ ANO + PERIODO,
  frecuencia = list("Year" = 2009:2013, "Period" = 1:2),
  datos      = df1,
  ask        = FALSE
)
all.equal(G, H)

# Observe las diferentes opciones que se le muestran por consola.
Agregar(formula = SEXO ~ YEAR + PERIODO, frecuencia = list("Year" = 2016:2021, "Period" = 1:3), df2)

Agregar(
  formula    = SEXO ~ YEAR + PERIODO,
  frecuencia = list("Year" = 2016:2021, "Period" = 1:3),
  intervalo  = list(c(2018, 1), c(2020, 3)),
  datos      = df2
)
Agregar(
  formula    = SEXO ~ YEAR,
  frecuencia = 2016:2021,
  intervalo  = c(2018, 2020),
```

```
  datos      = df2
)

# Observe que puede especificar más de una variable para realizar el agregado
# Se harán agregados simples y se unirán las filas en un único df
df1$SEXO <- c("Hombre", NA, NA, NA, NA, "Hombre", "Hombre",
              "Mujer", NA, NA, "Hombre", "Mujer", "Hombre", "Mujer"
              )
Agregar(
  formula    = TIPO_NIVEL + SEXO ~ ANO + PERIODO,
  frecuencia = list("Year" = 2010:2013, "Period" = 1:2),
  intervalo  = list(c(2010, 1), c(2013, 2)),
  datos      = df1
) -> A
Agregar(
  formula    = TIPO_NIVEL + SEXO ~ ANO + PERIODO,
  frecuencia = list("Year" = 2010:2013, "Period" = 1:2),
  datos      = df1
) -> B
```

`ejConsolidadoGrad` *Consolidado de Graduados*

Description

Consolidado de la población de graduados de los programas académicos en la Universidad Nacional de Colombia, cuenta con la información histórica desde el 2009-I al 2021-I.

Usage

```
ejConsolidadoGrad
```

Format

Un data frame (*data.frame, tbl_df o tbl*) con 775 filas y 5 columnas: 'Variable', 'YEAR', 'SEMESTRE', 'Clase', 'Total'.

Source

Para obtener más detalle de los metadatos consulte aquí.

Examples

```
# library(dplyr)
head(ejConsolidadoGrad)
```

`ejConsolidadoSaberPro2019` *Consolidado Saber Pro 2019*

Description

Consolidado de ejemplo de la prueba Saber Pro (*antes llamada ECAES*) del año 2019. El cual se utiliza para los ejemplos de la función `Tabla.SaberPro()`.

Usage

```
ejConsolidadoSaberPro2019
```

Format

Un data frame (*data.frame, tbl_df o tbl*) con 42 filas y 7 columnas: 'Variable', 'YEAR', 'Clase', 'n', 'Componente', 'Total', 'desv'.

Source

Para obtener más detalle de los metadatos consulte aquí.

Examples

```
# library(dplyr)
head(ejConsolidadoSaberPro2019)
```

`ejGraduados` *Muestra de Microdatos de Graduados*

Description

Muestra de microdatos de la población de graduados de los programas académicos en la Universidad Nacional de Colombia, se cuenta con una muestra de la información disponible (*del 2019-I al 2021-I*). Dicho dataset será usado en los ejemplos de las funciones `Tabla.General()` y `Plot.Treemap()`.

Usage

```
ejGraduados
```

Format

Un data frame (*data.frame, tbl_df o tbl*) con 27.830 filas y 53 columnas: 'YEAR', 'SEMESTRE', 'TIPO_NIVEL', 'NIVEL', 'DEP_NAC', 'COD_DEP_NAC', 'CIU_NAC', 'COD_CIU_NAC', 'LON_CIU_NAC', 'LAT_CIU_NAC', 'DEP_PROC', 'COD_DEP_PROC', 'CIU_PROC', 'COD_CIU_PROC', 'LON_CIU_PROC', 'LAT_CIU_PROC', 'CODS_NAC', 'CODN_NAC', 'NACIONALIDAD', 'EDAD_MOD', 'CAT_EDAD', 'SEXO', 'ESTRATO_ORIG', 'ESTRATO', 'TIPO_COL', 'PBM_ORIG', 'PBM', 'MAT_PVEZ', 'DISCAPACIDAD', 'TIPO_DISC', 'SNIES_SEDE_ADM', 'SEDE_NOMBRE_ADM', 'SNIES_SEDE_MAT', 'SEDE_NOMBRE_MAT', 'ADM_PEAMA_ANDINA', 'MOD_ADM', 'TIPO_ADM', 'PAES', 'PEAMA', 'MOV_PEAMA', 'CONVENIO', 'TIP_CONVENIO', 'SNIESU_CONVENIO', 'U_CONVENIO', 'FACULTAD', 'FACULTAD_S', 'SNIES_PROGRA', 'PROGRAMA', 'PROGRAMA_S', 'AREAC_SNIES', 'CA_CINE', 'CD_CINE', 'AREA_CINE'.

Source

Para obtener más detalle de los metadatos consulte aquí.

Examples

```
# library(dplyr)
ejGraduados[1:5, 1:10]
```

---

ejMiniAspirantesPre *Mini Microdatos de Aspirantes a Pregrado*

Description

Muestra de los microdatos (*únicamente estudiantes de pregrado*) de la población de aspirantes a cursar estudios de pregrado o postgrado en la Universidad Nacional de Colombia inscritos a través de convocatoria pública de manera regular o por medio de los programas de admisión especial existentes. Dicho dataset será usado en los ejemplos de la función `Plot.Boxplot()`.

Usage

```
ejMiniAspirantesPre
```

Format

Un data frame (*data.frame, tbl_df o tbl*) con 30.050 filas y 6 columnas: 'Serie', 'PTOTAL', 'TIPO_INS', 'INS_SEDE_NOMBRE', 'ADM_SEDE_NOMBRE', 'FACULTAD'.

Source

Para obtener más detalle de los metadatos consulte aquí.

Examples

```
# library(dplyr)
head(ejMiniAspirantesPre)
```

`ejMiniConsolidadoAsp` *Mini Consolidado de Aspirantes*

Description

Mini consolidado de la población de aspirantes a cursar estudios de pregrado o postgrado en la Universidad Nacional de Colombia inscritos a través de convocatoria pública de manera regular o por medio de los programas de admisión especial existentes -*sólo pregrado*-. Dicho dataset será usado en los ejemplos de la función `Plot.Drilldown()`.

Usage

```
ejMiniConsolidadoAsp
```

Format

Un data frame (*data.frame, tbl_df o tbl*) con 308 filas y 5 columnas: 'Variable', 'YEAR', 'SEMESTRE', 'Clase', 'Total'.

Source

Para obtener más detalle de los metadatos consulte aquí.

Examples

```
# library(dplyr)
head(ejMiniConsolidadoAsp)
```

`ejSaberPro2020` *Microdatos Saber Pro 2020*

Description

Microdatos de los resultados de la prueba Saber Pro del año 2020, obtenidos desde la página del ICFES, los cuales serán usados para los ejemplos de las funciones `Plot.Boxplot()` y `Plot.Radar()`.

Usage

```
ejSaberPro2020
```

Format

Un data frame (*data.frame, tbl_df o tbl*) con 4.896 filas y 59 columnas: 'YEAR', 'SEMESTRE', 'YEAR_MAT', 'SEMESTRE_MAT', 'TIPO_NIVEL', 'NIVEL', 'DEP_NAC', 'COD_DEP_NAC', 'CIU_NAC', 'COD_CIU_NAC', 'LON_CIU_NAC', 'LAT_CIU_NAC', 'DEP_PROC', 'COD_DEP_PROC', 'CIU_PROC', 'COD_CIU_PROC', 'LON_CIU_PROC', 'LAT_CIU_PROC', 'CODS_NAC', 'CODN_NAC', 'NACIONALIDAD', 'EDAD_MOD', 'CAT_EDAD', 'SEXO', 'ESTRATO_ORIG', 'ESTRATO', 'TIPO_COL', 'PBM_ORIG', 'PBM', 'SNIES_SEDE_ADM', 'SEDE_NOMBRE_ADM', 'SNIES_SEDE_MAT', 'SEDE_NOMBRE_MAT', 'ADM_PEAMA_ANDINA', 'MOD_ADM', 'TIPO_ADM', 'PAES', 'PEAMA', 'FACULTAD', 'FACULTAD_S', 'SNIES_PROGRA', 'PROGRAMA', 'PROGRAMA_S', 'AREAC_SNIES', 'CA_CINE', 'CD_CINE', 'AREA_CINE', 'SNP', 'PUNTAJE_GLOBAL', 'PUNT_COMP_CIUD', 'NIVEL_COMP_CIUD', 'PUNT_COMU_ESCR', 'NIVEL_COMU_ESCR', 'PUNT_INGLES', 'NIVEL_INGLES', 'PUNT_LECT_CRIT', 'NIVEL_LECT_CRIT', 'PUNT_RAZO_CUANT', 'NIVEL_RAZO_CUANT'.

Source

Para obtener más detalle de los metadatos consulte aquí.

Examples

```
# library(dplyr)
ejSaberPro2020[1:5, 1:10]
```

`Plot.Apiladas` *Cree un gráfico de barras apiladas dinámico/estático y flexible*

Description

Esta función proporciona excelentes herramientas y opciones para la visualización de un gráfico de barras apiladas con el objetivo de mostrar el tamaño relativo (*como porcentaje*) de una variable categórica, subdivididas por colores en función de un subgrupo. Dicha gráfica se va a representar usando la librería `Highcharter`, la cual usa internamente `JavaScript`.

Usage

```
Plot.Apiladas(
  datos,
  ejeX,
  valores,
  categoria,
  ano,
  periodo,
  addPeriodo = FALSE,
  colores,
  titulo = "",
  libreria = c("highcharter", "plotly"),
  estilo = NULL,
```

```
  estatico = FALSE
)
```

Arguments

datos	Un data frame, no un vector numérico.
ejeX	Una variable categórica dentro del data frame ingresado en datos.
valores	Variable numérica que contiene los valores que desea graficar.
categoria	Una variable categórica dentro del data frame ingresado en datos.
ano	Argument deprecated. This Argument still exist but will be removed in the next version.
periodo	Argument deprecated. This Argument still exist but will be removed in the next version.
addPeriodo	Argument deprecated. This Argument still exist but will be removed in the next version.
colores	Cadena de caracteres indicando los colores con los cuales se deben colorear cada una de las series correspondiente a cada nivel del argumento categoria. Si no se introduce algún vector se usará la paleta rainbow por defecto.
titulo	Cadena de caracteres indicando el título principal del plot.
libreria	Cadena de caracteres que indica el paquete con el cual se realizará el plot. Los valores permitidos son "highcharter" (*valor predeterminado*) y "plotly". Los valores se emparejarán parcialmente.
estilo	Lista compuesta por varios parámetros, los cuales van a ser usados para graficar las barras apiladas y cuyo objetivo es personalizar pequeños detalles de éste. • hc.Tema, hc.Credits, gg.Tema, gg.Legend y gg.Texto: Igual uso que en Plot.Series() • LegendTitle: Cadena de caracteres indicando un título para la leyenda (*diferentes niveles del argumento* categoria). • ply.LegendPosition: Igual uso que en Plot.Series() • ply.Credits: Igual uso que en Plot.Series() • gg.Bar: Igual uso que en Plot.Barras()
estatico	Si es FALSE (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con ggplot2.

Value

Retorna el diagrama de barras apiladas (*objeto widget de HTML*) creado. La clase del objeto retornado será un "htmlwidget" y adicionalmente pertenecerá a la clase "highchart".

Lista de argumentos de estilo

Sabemos que puede ser abrumador el número de argumentos dentro del parámetro estilo, pero es necesario si queremos ofrecer al usuario la máxima personalización dentro de cada función usando

cualquier librería. Por tal razón, a continuación, se detalla el listado completo de argumentos, usados al especificar la librería y en qué función están presentes (*marcado con una* × *si lo posee*).

Librería	**estilo$**	Plot.Series()	Plot.Barras()	Plot.Apiladas()	Plot.Boxplot()	P
—	*gg.Tema*	×	×	×	×	
l	*gg.Texto*	×	×	×	×	
l	*gg.Legend*	×		×	×	
l	*gg.Linea*	×				
l	*gg.Punto*	×				
l	*gg.Bar*		×	×		
l	*gg.VarWidth*				×	
l	*gg.OutShape*				×	
l	*gg.JitWidth*				×	
l	*gg.JitSize*				×	
l	*gg.Range*					
ggplot2	*gg.plty*					
l	*gg.plwd*					
l	*gg.cglwd*					
l	*gg.cglcol*					
l	*gg.fontsize.title*					
l	*gg.fontsize.labels*					
l	*gg.fontcolor.labels*					
l	*gg.border.lwds*					
l	*gg.border.col*					
l	*gg.lowerbound.cex.labels*					
l	*gg.force.print.labels*					
—	*gg.overlap.labels*					
»	*hc.Tema*	×	×	×	×	
l	*hc.Credits*	×	×	×	×	
highcharter	*hc.BoxInfo*	×				
l	*hc.Slider*	×				
»	*hc.borderRadius*					
•	*ply.Credits*	×	×	×	×	
◦	*ply.Legend*		×			
◦	*ply.LegendPosition*	×		×	×	
plotly	*ply.Interaction*	×			×	
◦	*ply.Relleno*					
◦	*ply.Opacidad*					
•	*ply.LegendTitle*					
dygraphs	*dyg.LegendWidth*	×				
»	*dyg.Resaltar*	×				
—	*e.Tema*					
l	*e.Credits*					
echarts4r	*e.Forma*					
l	*e.LegType*					
—	*e.LegLoc*					

Examples

```
# Ejemplo generalizado (sin uso de un consolidado como input)
# library("tibble"); library("dplyr")
set.seed(42)
Blood <- tibble(
  Quarter = sample(c("I", "II", "III", "IV"), size = 200, replace = TRUE),
  Group   = sample(
    c("O", "A", "B", "AB"), size = 200, prob = c(.5, .3, .16, .4), replace = TRUE
  ),
  Prevalence = round(runif(200)*100)
)
Plot.Apiladas(
  datos     = Blood     ,
  ejeX      = Quarter   ,
  valores   = Prevalence,
  categoria = Group     ,
  colores   = c("#FF553D", "#A5FF67", "#40D2FF", "#FFDB5C")
)


# ------------------------------------------------------------------------
Txt <- "BARRAS APILADAS EN FUNCI\u00d3N DEL NIVEL ACAD\u00c9MICO Y EL A\u00d1O"
Msj <- paste(
  "Se considera \u00fanicamente los valores obtenidos en el primer periodo",
  "acad\u00e9mico de cada a\u00f1o."
)
Plot.Apiladas(
  datos     = ejConsolidadoGrad |> filter(YEAR %in% c(2018:2020), SEMESTRE == 1),
 categoria = "NIVEL",    # Pruebe también con alguna de -> unique(ejConsolidadoGrad$Variable)
  colores   = c("#FFA700", "#C10AA1", "#01CDFE", "#00FF44", "#FF0040"),
  titulo    = Txt,
  estilo    = list(LegendTitle = "NIVEL ACAD\u00c9MICO:", hc.Tema = 4, hc.Credits = Msj)
)
Plot.Apiladas(
  datos     = ejConsolidadoGrad |> filter(YEAR %in% c(2018:2020), SEMESTRE == 1),
  categoria = "AREAC_SNIES",
    colores   = c("#D2D4DC", "#FF8ABF", "#945BC2", "#D11879",
                  "#FF7F7F", "#FFA568", "#9CFF86", "#89D8FF"),
  titulo    = "BARRAS APILADAS EN FUNCI\u00d3N DEL \u00c1REA DEL SNIES",
  libreria  = "plotly",
  estilo    = list(
    LegendTitle = "NIVEL ACAD\u00c9MICO:",
    ply.Credits = list(x = 0.5, y = 1.5, text = gsub("l p", "l\np", Msj)),
    ply.LegendPosition = list(x = 0.04, y = -0.3, orientation = "h")
  )
)
# Ejemplo usando el caso estático (ggplot2)
Plot.Apiladas(
  datos     = ejConsolidadoGrad |> filter(YEAR %in% c(2019:2021), SEMESTRE == 1),
  categoria = "NIVEL",
  colores   = c("#FFA700", "#C10AA1", "#01CDFE", "#00FF44", "#FF0040"),
  titulo    = gsub("L AC", "L\nAC", Txt),
```

```
  estatico  = TRUE,
  estilo    = list(
    LegendTitle = "NIVEL ACAD\u00c9MICO:", gg.Tema = 8,
    gg.Legend = list(legend.position = "right", legend.direction = "vertical"),
    gg.Bar    = list(width = 0.6, color = "#000000"),
    gg.Texto  = list(
      subtitle = "\u00bb\u00bb\u00bb", tag = "\u00ae",
      caption  = "Informaci\u00f3n Disponible desde 2009-1"
    )
  )
)
```

`Plot.Barras` *Cree un gráfico de barras que muestre la información de forma horizontal o vertical, para variables nominales u ordinales con dos diferentes paquetes*

Description

Esta función permite mostrar de forma interactiva (*y estática*) un gráfico de barras verticales u horizontales cuya altura/longitud es proporcional al valor de la variable (*categorías de una variable cualitativa*), lo anterior para ayudar a la creación de informes descriptivos y analíticos. Dicho diagrama se puede representar usando dos diferentes librerías que son `Highcharter` y `Plotly`, las cuales usan internamente `JavaScript`.

Usage

```
Plot.Barras(
  datos,
  valores,
  categoria,
  ano,
  periodo,
  freqRelativa = FALSE,
  ylim,
  vertical = TRUE,
  ordinal = FALSE,
  colores,
  titulo = "",
  labelX = "",
  labelY = "Número de",
  labelEje,
  addPeriodo = FALSE,
  textInfo = labelY,
  libreria = c("highcharter", "plotly"),
  estilo = NULL,
  estatico = FALSE
)
```

Arguments

datos	Un data frame, no un vector numérico.
valores	Variable numérica que contiene los valores que desea graficar.
categoria	Una variable categórica dentro del data frame ingresado en datos.
ano	Argument deprecated. This Argument still exist but will be removed in the next version.
periodo	Argument deprecated. This Argument still exist but will be removed in the next version.
freqRelativa	Si es FALSE (*valor predeterminado*) la serie graficada representará las frecuencias absolutas (*conteo*) más no las relativas (*porcentaje*).
ylim	Vector numérico que especifica el límite inferior y superior, respectivamente, del eje Y. Si no se introduce algún valor se mostrará todo el rango disponible para dicho eje.
vertical	Si es TRUE (*valor predeterminado*) indicará que la orientación del gráfico será vertical.
ordinal	Si es TRUE indicará que las categorías de la variable ingresada son ordinales (*no nominales*), esto con el fin de ordenar la disposición en el que se presentan en el eje del gráfico, el valor por defecto es FALSE.
colores	Cadena de caracteres indicando los colores con los cuales se deben colorear cada una de las series correspondiente a cada nivel del argumento categoria. Si no se introduce algún vector se usará la paleta rainbow por defecto.
titulo	Cadena de caracteres indicando el título principal del plot.
labelX	Cadena de caracteres indicando la etiqueta del eje X. Por defecto se emplea el rótulo "Periodo".
labelY	Cadena de caracteres indicando la etiqueta del eje Y.
labelEje	Cadena de caracteres indicando la etiqueta del eje X o Y (*dependiendo de la orientación del gráfico*). Por defecto se emplea el rótulo "Número de ".
addPeriodo	Argument deprecated. This Argument still exist but will be removed in the next version.
textInfo	Cadena de caracteres que especifica el texto que se escribe dentro de la caja de información al posar el cursor en alguna barra en el gráfico, producido por Highcharter, el valor por defecto es igual al de labelX.
libreria	Cadena de caracteres que indica el paquete con el cual se realizará el plot. Los valores permitidos son "highcharter" (*valor predeterminado*) y "plotly". Los valores se emparejarán parcialmente.
estilo	Lista compuesta por varios parámetros, los cuales van a ser usados de acuerdo con la librería especificada para graficar el plot y cuyo objetivo es personalizar pequeños detalles de ésta. • hc.Tema, hc.Credits, ply.Credits, gg.Tema y gg.Texto: Igual uso que en Plot.Series() • ply.Legend: Por defecto la gráfica muestra la leyenda fuera del gráfico de pie, si se introduce la cadena de texto "inside" se resumirá toda la información dentro del pie.

- `gg.Bar`: Una lista de parámetros admitidos por la función geom_bar()).

`estatico` Si es `FALSE` (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con `ggplot2`.

Details

Al usar el paquete `Highcharter` y usar las opciones de descarga, el nombre del archivo descargado será la concatenación del plot graficado y la categoría usada, así, por ejemplo, si se graficó el diagrama de barras para la categoría "Nacionalidad" el nombre será `PlotBarras_Nacionalidad.png`.

Value

Retorna el diagrama de barras (*objeto widget de HTML*) creado. La clase del objeto retornado será un "htmlwidget" y dependiendo de la librería usada pertenecerá adicionalmente a la clase "highchart" o "plotly".

Lista de argumentos de estilo

Sabemos que puede ser abrumador el número de argumentos dentro del parámetro `estilo`, pero es necesario si queremos ofrecer al usuario la máxima personalización dentro de cada función usando cualquier librería. Por tal razón, a continuación, se detalla el listado completo de argumentos, usados al especificar la librería y en qué función están presentes (*marcado con una* × *si lo posee*).

Librería	**estilo$**	Plot.Series()	Plot.Barras()	Plot.Apiladas()	Plot.Boxplot()	P
—	*gg.Tema*	×	×	×	×	
l	*gg.Texto*	×	×	×	×	
l	*gg.Legend*	×		×	×	
l	*gg.Linea*	×				
l	*gg.Punto*	×				
l	*gg.Bar*		×	×		
l	*gg.VarWidth*				×	
l	*gg.OutShape*				×	
l	*gg.JitWidth*				×	
l	*gg.JitSize*				×	
l	*gg.Range*					
ggplot2	*gg.plty*					
l	*gg.plwd*					
l	*gg.cglwd*					
l	*gg.cglcol*					
l	*gg.fontsize.title*					
l	*gg.fontsize.labels*					
l	*gg.fontcolor.labels*					
l	*gg.border.lwds*					
l	*gg.border.col*					
l	*gg.lowerbound.cex.labels*					
l	*gg.force.print.labels*					
—	*gg.overlap.labels*					
»	*hc.Tema*	×	×	×	×	

l	*hc.Credits*	×	×	×	×
highcharter	*hc.BoxInfo*	×			
l	*hc.Slider*	×			
»	*hc.borderRadius*				
•	*ply.Credits*	×	×	×	×
◦	*ply.Legend*		×		
◦	*ply.LegendPosition*	×		×	×
plotly	*ply.Interaction*	×			×
◦	*ply.Relleno*				
◦	*ply.Opacidad*				
•	*ply.LegendTitle*				
dygraphs	*dyg.LegendWidth*	×			
»	*dyg.Resaltar*	×			
—	*e.Tema*				
l	*e.Credits*				
echarts4r	*e.Forma*				
l	*e.LegType*				
—	*e.LegLoc*				

Examples

```
# Ejemplo generalizado (sin uso de un consolidado como input)
# library("tibble"); library("dplyr")
set.seed(42)
Blood <- tibble(
 Group = sample(c("O", "A", "B", "AB"), size = 200, prob = c(0.5, 0.3, 0.16, 0.4), replace = TRUE),
  RH    = sample(c("+", "-"), size = 200, replace = TRUE),
  Prevalence = round(runif(200)*100)
)
Plot.Barras(
  datos     = Blood     ,
  valores   = Prevalence,
  categoria = Group     ,
  ordinal   = TRUE      ,
  colores   = c("#FF553D", "#A5FF67", "#40D2FF", "#FFDB5C"),
  labelY    = "Prevalence"
)
Plot.Barras(
  datos     = Blood     ,
  valores   = Prevalence,
  categoria = Group     ,
  colores   = c("#FF553D", "#A5FF67", "#40D2FF", "#FFDB5C"),
  labelY    = "Prevalence",
  libreria  = "plotly"
)


# ------------------------------------------------------------------------
Msj <- "Ac\u00e1 puede ir m\u00e1s informaci\u00f3n acerca del gr\u00e1fico."
Plot.Barras(
```

```
  datos        = ejConsolidadoGrad |> filter(YEAR==2021, SEMESTRE==1),
  categoria    = "NIVEL",
  freqRelativa = TRUE,
  vertical     = TRUE,
  ordinal      = TRUE,
  colores      = c("#D7191C", "#FDAE61", "#FFFFBF", "#ABDDA4", "#2B83BA"),
  titulo       = "GRADUADOS DE ACUERDO CON EL NIVEL DE FORMACI\u00d3N (Periodo 2021-1)",
  labelY       = "Frecuencia Relativa<br>(% de graduados)",
  textInfo     = "Porcentaje de Graduados",
  libreria     = "highcharter",
  estilo       = list(hc.Tema = 2, hc.Credits = Msj)
)
# ------------------------------------------------------------------------
Txt <- "DISTRIBUCI\u00d3N DEL N\u00daMERO DE GRADUADOS POR NIVEL"
Msj <- "A\u00f1o 2020, sin segregar por semestre (considerando ambos)."
Plot.Barras(
  datos     = ejConsolidadoGrad |> filter(YEAR == 2020),
  categoria = "NIVEL",
  vertical  = FALSE,
  ordinal   = FALSE,
  colores   = c("#66C2A5", "#FC8D62", "#8DA0CB", "#E78AC3", "#A6D854"),
  titulo    = Txt,
  labelY    = "N\u00famero de Graduados",
  libreria  = "plotly",
  estilo    = list(
    ply.Credits = list(x = 0.45, y = 1.1, text = Msj), ply.Legend = FALSE
  )
)
# ------------------------------------------------------------------------
# Ejemplo usando el caso estático (ggplot2)
Plot.Barras(
  datos     = ejConsolidadoGrad |> filter(YEAR == 2020),
  categoria = "NIVEL",
  vertical  = FALSE,
  ordinal   = FALSE,
  colores   = c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3", "#FF7F00"),
  titulo    = gsub("DE GR", "DE\nGR", Txt),
  labelY    = "N\u00famero de Graduados",
  estatico  = TRUE,
  estilo    = list(
    gg.Tema  = 10,
    gg.Bar   = list(width = 0.2, color = "#000000"),
    gg.Texto = list(subtitle = gsub("A", "\nA", Msj),
                    caption  = "Informaci\u00f3n Disponible desde 2009-1",
                    tag      = "\u00ae"
    )
  )
)
```

`Plot.Boxplot` *Cree un diagrama de caja/boxplot dinámico y flexible con dos diferentes paquetes*

Description

Esta función proporciona excelentes herramientas y opciones para la visualización de un diagrama de caja y bigotes (*también conocido como boxplot*) dinámico con el objetivo de representar gráficamente una serie numérica a través de sus cuantiles. Dicho boxplot se puede representar usando dos diferentes librerías que son `Highcharter` y `Plotly`, las cuales usan internamente `JavaScript`.

Usage

```
Plot.Boxplot(
  datos,
  variable,
  grupo1,
  grupo2,
  vertical = TRUE,
  outliers = TRUE,
  jitter = FALSE,
  violin = FALSE,
  numericalVars,
  ylim,
  colores,
  sizeOutlier = 0,
  colOutlier = "#08306B",
  titulo = "",
  labelX = "Periodo",
  labelY = "",
  textBox = "",
  libreria = c("highcharter", "plotly"),
  estilo = NULL,
  estatico = FALSE
)
```

Arguments

`datos`	Un data frame, no un vector numérico.
`variable`	Una variable numérica dentro del data frame ingresado en `datos`.
`grupo1`	Una variable categórica dentro del data frame ingresado en `datos`.
`grupo2`	Otra variable categórica dentro del data frame ingresado en `datos` por si se desea segregar por otra clase el grupo principal.
`vertical`	Si es `TRUE` (*valor predeterminado*) indicará que la orientación del gráfico será vertical.
`outliers`	Si es `TRUE` (*valor predeterminado*) se mostrarán los puntos correspondientes a los datos atípicos, defínalo en `FALSE` si desea ocultar dichos puntos.

`jitter` Si es `TRUE` se agregará las observaciones de cada grupo con un poco de ruido aleatorio encima de las cajas, útil para mostrar la distribución subyacente de los datos. El valor por defecto es `FALSE`.

`violin` Si es `TRUE` se acompañará el boxplot con su diagrama de densidad, logrando ser más informativo al mostrar la distribución completa de los datos. Solo aplica para la librería `"plotly"`.

`numericalVars` Una lista (*ya sea creada con la sintaxis* `base` *o* `tidy`) con las variables numéricas dentro del data frame ingresado en `datos` con las que se desea crear un botón dinámico para desplazarse entre ellas fijando el grupo ingresado en `grupo1`.

`ylim` Vector numérico que especifica el límite inferior y superior, respectivamente, del eje `Y`. Si no se introduce algún valor se mostrará todo el rango disponible para dicho eje.

`colores` Cadena de caracteres indicando los colores con los cuales se deben colorear cada una de las trazas correspondiente a cada nivel del argumento `grupo1`. Si no se introduce algún vector se usará la paleta `rainbow` por defecto.

`sizeOutlier` Valor numérico que indica el tamaño de los puntos considerados como atípicos, por defecto se tiene un valor específico al que se le sumará el ingresado acá.

`colOutlier` Cadena de caracteres indicando el color de los puntos considerados como atípicos, por defecto se pintarán de un azul rey.

`titulo` Cadena de caracteres indicando el título principal del plot.

`labelX` Cadena de caracteres indicando la etiqueta del eje `X`. Por defecto se emplea el rótulo "Periodo".

`labelY` Cadena de caracteres indicando la etiqueta del eje `Y`.

`textBox` Cadena de caracteres indicando el nombre de la serie numérica con la que se construye las cajas, necesario únicamente si se especifica solamente el `grupo1`, para el caso en que se tenga dos grupos no tendrá ningún efecto en el plot.

`libreria` Cadena de caracteres que indica el paquete con el cual se realizará el plot. Los valores permitidos son `"highcharter"` (*valor predeterminado*) y `"plotly"`. Los valores se emparejarán parcialmente.

`estilo` Lista compuesta por varios parámetros, los cuales van a ser usados de acuerdo con la librería especificada para graficar el boxplot y cuyo objetivo es personalizar pequeños detalles de éste.

- `LegendTitle`, `hc.Tema` y `hc.Credits`: Igual uso que en `Plot.Series()`
- `ply.Interaction`, `ply.LegendPosition` y `ply.Credits`: Igual uso que en `Plot.Series()`
- `gg.Tema`, `gg.Legend` y `gg.Texto`: Igual uso que en `Plot.Series()`
- `gg.VarWidth`: Si es `FALSE` (*valor predeterminado*) se realizará un diagrama de caja estándar. Si es `TRUE`, las cajas se dibujan con anchos proporcionales a las raíces cuadradas del número de observaciones en los grupos. Para más información, consulte la función geom_boxplot()).
- `gg.OutShape`: Modifica la forma de los outliers (*datos atípicos*). Los posibles valores son un número entero entre $[1, 23]$. Para más información consulte la función geom_boxplot()).

- `gg.JitWidth`: Valor numérico que indica el tamaño del ancho en el que los puntos pueden dispersarse en cada una de las cajas El valor por defecto es `0.25`. Para más información, consulte la función geom_jitter()).
- `gg.JitSize`: Valor numérico que indica el tamaño de los jittered points. El valor por defecto es `0.4`. Para más información, consulte la función geom_jitter()).

`estatico` Si es `FALSE` (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con `ggplot2`.

Details

El argumento `numericalVars` funciona solamente con la librería `"plotly"`, pues la función de crear los botones dinámicos es procedente de dicha librería.

Value

Retorna el boxplot (*objeto widget de HTML*) creado. La clase del objeto retornado será un "htmlwidget" y dependiendo de la librería usada pertenecerá adicionalmente a la clase "highchart" o "plotly".

Lista de argumentos de estilo

Sabemos que puede ser abrumador el número de argumentos dentro del parámetro `estilo`, pero es necesario si queremos ofrecer al usuario la máxima personalización dentro de cada función usando cualquier librería. Por tal razón, a continuación, se detalla el listado completo de argumentos, usados al especificar la librería y en qué función están presentes (*marcado con una* × *si lo posee*).

Librería	**estilo$**	`Plot.Series()`	`Plot.Barras()`	`Plot.Apiladas()`	`Plot.Boxplot()`	`P`
—	*gg.Tema*	×	×	×	×	
l	*gg.Texto*	×	×	×	×	
l	*gg.Legend*	×		×	×	
l	*gg.Linea*	×				
l	*gg.Punto*	×				
l	*gg.Bar*		×	×		
l	*gg.VarWidth*				×	
l	*gg.OutShape*				×	
l	*gg.JitWidth*				×	
l	*gg.JitSize*				×	
l	*gg.Range*					
ggplot2	*gg.plty*					
l	*gg.plwd*					
l	*gg.cglwd*					
l	*gg.cglcol*					
l	*gg.fontsize.title*					
l	*gg.fontsize.labels*					
l	*gg.fontcolor.labels*					
l	*gg.border.lwds*					
l	*gg.border.col*					

l	*gg.lowerbound.cex.labels*				
l	*gg.force.print.labels*				
—	*gg.overlap.labels*				
»	*hc.Tema*	×	×	×	×
l	*hc.Credits*	×	×	×	×
highcharter	*hc.BoxInfo*	×			
l	*hc.Slider*	×			
»	*hc.borderRadius*				
•	*ply.Credits*	×	×	×	×
◦	*ply.Legend*		×		
◦	*ply.LegendPosition*	×		×	×
plotly	*ply.Interaction*	×			×
◦	*ply.Relleno*				
◦	*ply.Opacidad*				
•	*ply.LegendTitle*				
dygraphs	*dyg.LegendWidth*	×			
»	*dyg.Resaltar*	×			
—	*e.Tema*				
l	*e.Credits*				
echarts4r	*e.Forma*				
l	*e.LegType*				
—	*e.LegLoc*				

Examples

```
Txt <- "EVOLUCI\u00d3N DEL PUNTAJE EN EL EXAMEN DE ADMISI\u00d3N"
Msj <- "Aspirantes a pregrado (<i>no se incluye los datos at\u00edpicos</i>)"
Plot.Boxplot(
  datos       = ejMiniAspirantesPre,
  variable    = PTOTAL,
  grupo1      = Serie,
  outliers    = FALSE,
  ylim        = c(0, 1000),
  colores     = pals::jet(30),
  sizeOutlier = 1,
  colOutlier  = "#FF3366",
  titulo      = Txt,
  labelY      = "Puntaje",
  textBox     = "Score",
  libreria    = "highcharter",
  estilo      = list(hc.Tema = 2, hc.Credits = Msj)
)

# ------------------------------------------------------------------------
Msj2 <- paste(
  "Aspirantes a pregrado",
  "(<i>cada periodo se encuentra segregado por el tipo de admisi\u00f3n</i>)"
)
Plot.Boxplot(
  datos    = ejMiniAspirantesPre,
```

```
  variable = PTOTAL,
  grupo1   = Serie,
  grupo2   = TIPO_INS,
  outliers = TRUE,
  ylim     = c(0, 1000),
  colores  = c("#00ABFF", "#F3224B", "#FCD116", "#29DF2C"),
  titulo   = Txt,
  labelY   = "Puntaje",
  libreria = "highcharter",
  estilo   = list(LegendTitle = "Programa:", hc.Tema = 6, hc.Credits = Msj2)
)
# ------------------------------------------------------------------------
Plot.Boxplot(
  datos    = iris,
  variable = Sepal.Length,
  grupo1   = Species,
  violin   = TRUE,
  colores  = c("#FF1D58", "#FDB911", "#00E527"),
  titulo   = "BOXPLOT DE LA LONGITUD DEL S\u00c9PALO | IRIS DATASET",
  labelX   = "Especie",
  labelY   = "Longitud del S\u00e9palo",
  libreria = "plotly"
)
# ------------------------------------------------------------------------
Plot.Boxplot(
  datos       = ejMiniAspirantesPre,
  variable    = PTOTAL,
  grupo1      = Serie,
  grupo2      = TIPO_INS,
  jitter      = TRUE,
  ylim        = c(0, 1000),
  colores     = c("#00ABFF", "#F3224B", "#FCD116", "#29DF2C"),
  sizeOutlier = 0,
  colOutlier  = "#D3D3D3",
  titulo      = Txt,
  labelY      = "Puntaje",
  libreria    = "plotly",
  estilo      = list(
    LegendTitle = "Programa:", ply.Interaction = "closest",
    ply.LegendPosition = list(x = 0.16, y = -0.25, orientation = "h"),
    ply.Credits = list(x = 0.4, y = 0.95, text = Msj2)
  )
) -> Advertencia
suppressWarnings(print(Advertencia))


# ------------------------------------------------------------------------
# library(dplyr)
df <- ejSaberPro2020 |>
  select(SEDE_NOMBRE_ADM, PUNTAJE_GLOBAL, PUNT_RAZO_CUANT, PUNT_INGLES,
         PUNT_LECT_CRIT, PUNT_COMP_CIUD, PUNT_COMU_ESCR
         )
Numericas <- vars(PUNT_RAZO_CUANT, PUNT_INGLES, PUNT_LECT_CRIT, PUNT_COMP_CIUD, PUNT_COMU_ESCR)
```

```
# Numericas <- c("PUNT_RAZO_CUANT", "PUNT_INGLES", ... , "PUNT_COMU_ESCR")
misColores <- c(
  "#29ABE2", # AZUL CLARO  | Amazonia
  "#8CC63F", # VERDE       | Bogota
  "#CC241D", # ROJO        | Caribe
  "#0071BC", # AZUL VIVO   | Manizales
  "#F15A24", # NARANJA     | Medellin
  "#FBB03B", # AMARILLO    | Orinoquia
  "#93278F", # MORADO      | Palmira
  "#8A381A"  # GRIS        | Tumaco
)
Plot.Boxplot(
  datos         = df,
  variable      = PUNTAJE_GLOBAL,
  grupo1        = SEDE_NOMBRE_ADM,
  numericalVars = Numericas,
  colores       = misColores,
  libreria      = "plotly"
)


# ------------------------------------------------------------------------
# Ejemplo usando el caso estático (ggplot2)
# library(pals)
Plot.Boxplot(
  datos      = ejMiniAspirantesPre,
  variable   = PTOTAL,
  grupo1     = Serie,
  jitter     = TRUE,
  colores    = pals::turbo(30),
  colOutlier = "#CBB8FF",
  titulo     = gsub("L E", "L\nE", Txt),
  labelY     = "Puntaje",
  textBox    = "Score",
  estatico   = TRUE,
  estilo     = list(
    gg.Tema = 11, gg.Legend = list(legend.position = "none"),
    gg.Texto = list(
      subtitle = gsub("<|/|i>", "", Msj ),
      caption  = "Información Disponible desde 2008-1", tag = "\u00ae"
    ),
    gg.VarWidth = TRUE, gg.JitWidth = 0.08, gg.JitSize = 0.05
  )
)

# ------------------------------------------------------------------------
Plot.Boxplot(
  datos       = ejMiniAspirantesPre,
  variable    = PTOTAL,
  grupo1      = Serie,
  grupo2      = TIPO_INS,
  outliers    = TRUE,
  colores     = c("#00ABFF", "#F3224B", "#FCD116", "#29DF2C"),
```

```
  sizeOutlier = 2,
  colOutlier  = "#F5E8E8",
  titulo      = gsub("L E", "L\nE", Txt),
  labelY      = "Puntaje",
  textBox     = "Score",
  estatico    = TRUE,
  estilo      = list(
    LegendTitle = "NIVEL ACAD\u00c9MICO:", gg.Tema = 9, gg.OutShape = 21,
    gg.Legend = list(legend.position = "bottom", legend.direction = "horizontal"),
    gg.Texto = list(
      subtitle = gsub("<|/|i>", "", Msj2),
      caption  = "Información Disponible desde 2008-1", tag = "\u00ae"
    )
  )
)
```

`Plot.Drilldown` *Cree un gráfico profundo (*drill down*) de torta/barras dinámico y flexible*

Description

Esta función proporciona excelentes herramientas y opciones para la visualización de un gráfico drill down con el objetivo de poder inspeccionar los datos con mayor nivel de detalle, sin la necesidad de navegar o salir de él, pudiendo hacer clic en diversos elementos como columnas o sectores circulares. Dicha gráfica se va a representar usando la librería `Highcharter`, la cual usa internamente `JavaScript`.

Usage

```
Plot.Drilldown(
  datos,
  varPrincipal,
  varSecundaria,
  ano,
  periodo,
  torta = TRUE,
  vertical = TRUE,
  colores,
  colores2,
  titulo = "",
  label = "",
  textInfo = "",
  addPeriodo = TRUE,
  estilo = NULL
)
```

Arguments

datos	Un data frame, no un vector numérico.
varPrincipal	Una variable categórica dentro del data frame ingresado en datos.
varSecundaria	Otra variable categórica dentro del data frame ingresado en datos, diferente a la principal, pues se segregará a otros niveles.
ano	Igual uso que en Plot.Torta()
periodo	Igual uso que en Plot.Torta()
torta	Si es TRUE (*valor predeterminado*) el primer nivel o gráfico principal será un diagrama de torta, defínalo en FALSE si desea que éste sea un gráfico de barras.
vertical	Si es TRUE (*valor predeterminado*) indicará que tanto la orientación del gráfico principal como secundario será vertical. Solamente aplicará si el argumento torta es FALSE.
colores	Cadena de caracteres indicando los colores con los cuales se deben colorear cada una de las trazas correspondiente a cada nivel del argumento varPrincipal. Si no se introduce algún vector se usará la paleta rainbow por defecto.
colores2	Igual que colores pero aplicado al gráfico secundario.
titulo	Igual uso que en Plot.Series()
label	Cadena de caracteres indicando el agregado al que hace referencia el gráfico. Por defecto no se emplea ningún rótulo.
textInfo	Cadena de caracteres indicando el texto que aparecerá dentro de la caja de información al pasar el mouse por las diferentes columnas del gráfico de barras.
addPeriodo	Igual uso que en Plot.Torta()
estilo	Lista compuesta por varios parámetros, los cuales van a ser usados para graficar el drill down y cuyo objetivo es personalizar pequeños detalles de éste. • LegendTitle: Cadena de caracteres indicado un título para la leyenda (*diferentes niveles del argumento* varPrincipal). • hc.Tema y hc.Credits: Igual uso que en Plot.Series()

Value

Retorna el diagrama drill down (*objeto widget de HTML*) creado. La clase del objeto retornado será un "htmlwidget" y adicionalmente pertenecerá a la clase "highchart".

Examples

```
# library(dplyr)
df <- ejMiniConsolidadoAsp |>
  filter(Clase != "Sin Información", tolower(Clase) != "no aplica")
text <- "DISTRIBUCI\u00d3N DE ASPIRANTES A PREGRADO EN SITUACI\u00d3N DE DISCAPACIDAD"
Msj  <- paste(
  "Discapacidad: Deficiencia, limitaci\u00f3n de la actividad ",
  "y la restricci\u00f3n de la participaci\u00f3n."
)
Plot.Drilldown(
  datos         = df,
```

```
  varPrincipal  = "DISCAPACIDAD",
  varSecundaria = "TIPO_DISC",
  ano           = max(df$YEAR),
  periodo       = slice(df, n())$SEMESTRE,
  torta         = TRUE, # Pruebe poniendo ambos valores ahora en FALSE
  vertical      = TRUE,
  colores       = c("#FF0040", "#00FF40"),
  colores2      = c("#66C2A5", "#FC8D62", "#8DA0CB", "#E78AC3", "#A6D854", "#FFD92F"),
  titulo        = text,
  label         = "Aspirantes",
  textInfo      = "Aspirantes con discapacidades por tipo",
  addPeriodo    = TRUE,
  estilo        = list(hc.Tema = 7, hc.Credits = Msj)
)
```

`Plot.Histograma` *Cree un histograma que represente gráficamente la distribución de datos en un conjunto, facilitando la comprensión de su frecuencia y patrones con dos diferentes paquetes.*

Description

Esta función proporciona excelentes herramientas y opciones para la visualización de un histograma, con el objetivo de que pueda representar la distribución de frecuencia de datos numéricos (*variable de un conjunto de datos*) en forma de barras, donde cada barra representa la cantidad de veces que aparece un valor o rango de valores. Dicho diagrama se puede representar usando dos diferentes librerías que son `Highcharter` y `Plotly`, las cuales usan internamente `JavaScript`.

Usage

```
Plot.Histograma(
  datos,
  variable,
  color,
  bins,
  density = FALSE,
  titulo = "",
  xlim,
  ylim,
  labelX = NULL,
  labelY = "Conteo",
  libreria = c("highcharter", "plotly"),
  estilo = NULL,
  estatico = FALSE
)
```

Arguments

`datos`	Un data frame, no un objeto clase serie de tiempo o vector numérico.
`variable`	Una variable numérica dentro del data frame ingresado en `datos`.
`color`	Cadena de caracteres indicando el color de relleno de las barras para todos los rangos de valores (intervalos).
`bins`	Valor numérico que indica el número máximo de bins deseado. Este valor se utilizará en un algoritmo que decidirá el tamaño de bins óptimo para que el histograma visualice mejor la distribución de los datos.
`density`	Si es `TRUE` se agregará la curva de densidad superpuesta al histograma. El valor por defecto es `FALSE`.
`titulo`	Cadena de caracteres indicando el título principal del plot.
`xlim`	Vector numérico que especifica el límite inferior y superior, respectivamente, del eje `X`. Si no se introduce algún valor se mostrará todo el rango disponible para dicho eje.
`ylim`	Vector numérico que especifica el límite inferior y superior, respectivamente, del eje `Y`. Si no se introduce algún valor se mostrará todo el rango disponible para dicho eje.
`labelX`	Cadena de caracteres indicando la etiqueta del eje `X`. Por defecto se emplea el rótulo "Periodo".
`labelY`	Cadena de caracteres indicando la etiqueta del eje `Y`.
`libreria`	Cadena de caracteres que indica el paquete con el cual se realizará la serie. Los valores permitidos son `"highcharter"` (*valor predeterminado*), `"plotly"` o `"dygraphs"`. Los valores se emparejarán parcialmente.
`estilo`	Lista compuesta por varios parámetros, los cuales van a ser usados de acuerdo con la librería especificada para graficar el plot y cuyo objetivo es personalizar pequeños detalles de ésta. • `hc.Tema`, `ply.Credits`, `gg.Tema` y `gg.Texto`: Igual uso que en `Plot.Series()` • `ply.Density`: Una lista de parámetros admitidos por el argumento `line` de la función add_lines(). • `gg.Hist`: Una lista de parámetros admitidos por la función geom_histogram(). • `gg.Density`: Una lista de parámetros admitidos por la función geom_density().
`estatico`	Si es `FALSE` (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con `ggplot2`.

Details

- Al usar la librería `Highcharter` no se podrá superponer la curva de densidad, pues no se dispone de esta funcionalidad para dicho paquete.
- Si está usando el caso estático (`ggplot2`) y adicionalmente está graficando la curva de densidad, recuerde que el eje `Y` que predomina es el de la curva de densidad, por tal razón, si va a usar el argumento `ylim` debe recordar que quedara en la escala de $[0, 1]$.
- Tenga cuidado al usar el argumento `xlim` en el caso estático, ya que si uno de los bins se ve cortado (*no abarca el inicio y fin de éste*) no se graficará dicho intervalo.

- Al usar el paquete `Highcharter` y usar las opciones de descarga, el nombre del archivo descargado será la concatenación del plot graficado y la categoría usada, así, por ejemplo, si se graficó el diagrama de barras para la categoría "Nacionalidad" el nombre será `PlotHistograma__Nacionalidad.png`.

Value

Retorna el histograma (*objeto widget de HTML*) creado. La clase del objeto retornado será un "htmlwidget" y dependiendo de la librería usada pertenecerá adicionalmente a la clase "highchart" o "plotly".

Lista de argumentos de estilo

Sabemos que puede ser abrumador el número de argumentos dentro del parámetro `estilo`, pero es necesario si queremos ofrecer al usuario la máxima personalización dentro de cada función usando cualquier librería. Por tal razón, a continuación, se detalla el listado completo de argumentos, usados al especificar la librería y en qué función están presentes (*marcado con una* × *si lo posee*).

Librería	**estilo$**	Plot.Series()	Plot.Barras()	Plot.Apiladas()	Plot.Boxplot()	P
—	*gg.Tema*	×	×	×	×	
l	*gg.Texto*	×	×	×	×	
l	*gg.Legend*	×		×	×	
l	*gg.Linea*	×				
l	*gg.Punto*	×				
l	*gg.Bar*		×	×		
l	*gg.VarWidth*				×	
l	*gg.OutShape*				×	
l	*gg.JitWidth*				×	
l	*gg.JitSize*				×	
l	*gg.Range*					
ggplot2	*gg.plty*					
l	*gg.plwd*					
l	*gg.cglwd*					
l	*gg.cglcol*					
l	*gg.fontsize.title*					
l	*gg.fontsize.labels*					
l	*gg.fontcolor.labels*					
l	*gg.border.lwds*					
l	*gg.border.col*					
l	*gg.lowerbound.cex.labels*					
l	*gg.force.print.labels*					
—	*gg.overlap.labels*					
»	*hc.Tema*	×	×	×	×	
l	*hc.Credits*	×	×	×	×	
highcharter	*hc.BoxInfo*	×				
l	*hc.Slider*	×				
»	*hc.borderRadius*					
•	*ply.Credits*	×	×	×	×	
◦	*ply.Legend*		×			
◦	*ply.LegendPosition*	×		×	×	

plotly	*ply.Interaction*	×	×
◦	*ply.Relleno*		
◦	*ply.Opacidad*		
•	*ply.LegendTitle*		
dygraphs	*dyg.LegendWidth*	×	
»	*dyg.Resaltar*	×	
—	*e.Tema*		
l	*e.Credits*		
echarts4r	*e.Forma*		
l	*e.LegType*		
—	*e.LegLoc*		

Examples

```
Txt <- "Datos Oficiales de la Prueba Saber Pro del año 2020"
Plot.Histograma(
  datos    = ejSaberPro2020,
  variable = PUNT_RAZO_CUANT,
  color    = "#12D640",
  bins     = 100,
  titulo   = "DISTRIBUCI\u00d3N EN EL PUNTAJE DE RAZONAMIENTO CUANTITATIVO",
  ylim     = c(0, 175),
  labelX   = "Puntaje en Matem\u00e1ticas",
  labelY   = "N\u00famero de Estudiantes",
  libreria = "highcharter",
  estilo   = list(hc.Tema  = 4)
)
# ------------------------------------------------------------------------
Plot.Histograma(
  datos    = ejSaberPro2020,
  variable = PUNT_RAZO_CUANT,
  color    = "#B9ABD1",
  bins     = 80,
  density  = TRUE,
  titulo   = "DISTRIBUCI\u00d3N EN EL PUNTAJE\nDE RAZONAMIENTO CUANTITATIVO",
  ylim     = c(0, 400),
  labelX   = "Puntaje en Matem\u00e1ticas",
  labelY   = "N\u00famero de Estudiantes",
  libreria = "plotly",
  estilo   = list(
    ply.Credits = list(x = 0.5, y = 1.1, text = Txt),
    ply.Density = list(color = "#DD3380", width = 4, dash = "dot", opacity = 0.2)
  )
)
# ------------------------------------------------------------------------
# Ejemplo usando el caso estático (ggplot2)
Plot.Histograma(
  datos    = ejSaberPro2020,
  variable = PUNT_RAZO_CUANT,
  density  = TRUE,
  titulo   = "DISTRIBUCI\u00d3N EN EL PUNTAJE DE RAZONAMIENTO CUANTITATIVO",
```

```
  labelX   = "Puntaje en Matem\u00e1ticas",
  labelY   = "N\u00famero de Estudiantes",
  estatico = TRUE,
  estilo   = list(
    gg.Tema    = 6,
    gg.Hist    = list(
     binwidth = 10, fill = "#FF4040", color = "#144169", alpha = 0.5, linetype = "dashed"
    ),
   gg.Density = list(color = "#20B2AA", fill = "#40E0D0", alpha = 0.4, lwd = 1.1, linetype = 2),
    gg.Texto   = list(subtitle = "\u00bb\u00bb\u00bb", tag = "\u00ae", caption  = Txt)
  )
)
```

`Plot.Mapa` *Cree un widget para visualizar datos geográficos en un mapa interactivo usando el paquete* `Leaflet`

Description

Esta función está planeada para facilitar la creación de mapas interactivos compatible con plataformas móviles y de escritorio, además de estar diseñada pensando en la simplicidad y el rendimiento. Esta utilidad produce mapas que tienen controles para hacer zoom, desplazarse y alternar capas y puntos entre mostrar y ocultar. Igualmente, permite incrustar mapas en webs, documentos `R Markdown` y aplicaciones `Shiny`. Todo lo anterior basado enteramente en la librería `Leaflet`, la cual es la biblioteca `JavaScript` de código abierto más popular para mapas interactivos.

Usage

```
Plot.Mapa(
  datos,
  df,
  depto,
  mpio,
  variable,
  agregado = TRUE,
  zoomIslas = FALSE,
 estadistico = c("Conteo", "Promedio", "Mediana", "Varianza", "SD", "CV", "Min", "Max"),
  tipo = c("Deptos", "SiNoMpios", "Mpios", "DeptoMpio"),
  SiNoLegend,
  titulo,
  naTo0 = TRUE,
  colNA = "#EEEEEE",
  centroideMapa,
  zoomMapa = 6,
  baldosas,
  cortes,
  colores,
```

```
  showSedes = TRUE,
  colSedes,
  opacidad = 0.7,
  colBorde,
  compacto = TRUE,
  textSize = 10,
  limpio = FALSE,
  estatico = FALSE,
  estilo,
  ...
)
```

Arguments

datos	Un data frame, no un vector numérico.
df	Argument deprecated, use datos instead.
depto	Una variable numérica dentro del data frame ingresado en datos, que contiene los códigos de los departamentos, de acuerdo con la codificación de la División Político-Administrativa de Colombia (*DIVIPOLA*) dispuesta por el DANE.
mpio	Una variable numérica dentro del data frame ingresado en datos, que contiene los códigos de los municipios, de acuerdo con la codificación de la División Político-Administrativa de Colombia (*DIVIPOLA*) dispuesta por el DANE.
variable	Variable auxiliar con la cual se calculará el estadístico previamente seleccionado. Para el caso en que la estadística a calcular sea el conteo no es necesario (*no se usará*) especificar dicha variable numérica.
agregado	Si es TRUE (*valor predeterminado*) indica que el data frame ingresado se deberá agrupar porque no es un consolidado, sino que se cuenta con los microdatos. Por el contrario, si se especifica en FALSE quiere decir que cada fila es única y se cuenta con el estadístico ya calculado, además de poder ingresar únicamente uno de los argumentos depto o mpio.
zoomIslas	Valor booleano que indica si desea realizar un zoom al archipiélago de San Andrés, Providencia y Santa catalina. El valor a retornar no será un único plot sino una lista compuesta por dos figuras, una para el mapa sin las islas y otro únicamente con ellas, para que usted realice posteriormente la unión de ambos. Aplica únicamente para el caso estático y su valor por defecto es FALSE.
estadistico	Cadena de caracteres que indica el estadístico a graficar en el mapa. Los valores permitidos son "Conteo" (*valor predeterminado*), "Promedio", "Mediana", "Varianza", "SD", "CV", "Min" y "Max".
tipo	Cadena de caracteres que indica el tipo de mapa a graficar. Los valores permitidos son "Deptos", "SiNoMpios", "Mpios" y "DeptoMpio" (*valor predeterminado*). Se emparejará parcialmente.
SiNoLegend	Vector de caracteres de longitud 2 que permite editar las opciones de la etiqueta de la leyenda de los mapas "SiNoMpios". El valor por defecto es ("0", "1 o más").
titulo	Cadena de caracteres indicando la segregación que presenta el mapa y el periodo al que hace referencia éste, separados por un espacio, por ejemplo, "Admitidos 2021-I".

`naTo0` Si es `TRUE` (*valor predeterminado*) los valores introducidos como `NA` (*not available*) se cambiarán por el valor de 0. Ajústelo a `FALSE` para que no se realice tal cambio y mostrar en la caja de información la leyenda "Sin Información".

`colNA` Cadena de caracteres indicando el color que tendrá la categoría `NA` (*si esta se presenta*). El valor por defecto es un gris muy claro.

`centroideMapa` Cadena de caracteres indicando el departamento que servirá de centroide al momento de graficar el mapa. El valor por defecto es "CUNDINAMARCA". Se emparejará parcialmente.

`zoomMapa` Valor numérico que indica el nivel de zoom del mapa (*usado por la función* setView()). El valor por defecto es `6`, entre mayor sea su valor más zoom se aplicará al mapa.

`baldosas` Vector de caracteres indicando los mapas base con los que se realizará el mapa, sean los popularizados por Google Maps o por terceros. Los valores aceptados son los admitidos por la función addProviderTiles(), así mismo los valores por defecto son `c("CartoDB.Positron", "Esri.WorldStreetMap", "Esri.NatGeoWorldMap")`, algunos otros valores pueden ser:

- "Esri.DeLorme"
- "Esri.WorldTerrain"
- "Esri.WorldShadedRelief"
- "Esri.WorldPhysical"
- "Esri.OceanBasemap"
- "Esri.WorldGrayCanvas"
- "Esri.WorldImagery"
- "Stamen.Toner"
- "Stamen.TonerLite"
- "Stamen.TonerLines"
- "Stamen.Watercolor"
- "Stamen.TonerHybrid"

La lista completa la puede consultar aquí

`cortes` Vector numérico indicando los cortes con los cuales se crearán los intervalos. No aplica para el tipo de mapa "SiNoMpios", pues este es binario. Para el tipo de mapa "DeptoMpio" se debe pasar una lista de la siguiente manera `list(Deptos = c(), Mpios = c())`, pues requiere dos cortes, uno para departamentos y otro para municipios.

`colores` Vector de caracteres indicando los colores para cada uno de los intervalos con los que cuenta el mapa. Si no se introduce algún vector, se usará una paleta predeterminada dependiendo del tipo de mapa.

`showSedes` Si es `TRUE` (*valor predeterminado*) en el control de capas (*usado en la función* addLayersControl()) aparecerá el grupo destinado a mostrar o no la ubicación de las distintas sedes de la Universidad Nacional de Colombia. Ajústelo a `FALSE` para que en el control de capas no aparezca dicha opción.

`colSedes` Vector de caracteres (*de longitud 9*) indicando los colores del icono de ubicación de las distintas sedes de la Universidad Nacional de Colombia. Los

colores permitidos son los que acepta la función makeAwesomeIcon(), es decir, "red", "darkred", "lightred", "orange", "beige", "green", "darkgreen", "lightgreen", "blue", "darkblue", "lightblue", "purple", "darkpurple", "pink", "cadetblue", "white", "gray", "lightgray", "black".

`opacidad` Un número entre $[0, 1]$ que indica la opacidad de las capas.

`colBorde` Cadena de caracteres indicando el color del borde de los polígonos al momento de pasar el cursor sobre él.

`compacto` Si es `TRUE` (*valor predeterminado*) el control de capas se representará como un icono que se expande cuando se coloca el cursor sobre él. Ajústelo a `FALSE` para que el control de capas siempre aparezca en su estado expandido.

`textSize` Valor numérico que indica el tamaño del texto de las etiquetas de los municipios. El valor para los departamentos será $+2px$.

`limpio` Si es `FALSE` (*valor predeterminado*) se mostrará el MiniMapa, la barra de escala y los botones para ver en pantalla completa, retornar zoom y localización. Ajústelo a `TRUE` si desea omitir dichas herramientas adicionales al mapa.

`estatico` Si es `FALSE` (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con `ggplot2`.

`estilo` Lista compuesta por varios parámetros, los cuales van a ser usados para graficar el mapa **estático** y cuyo objetivo es personalizar pequeños detalles de éste.

- `anchoBorde`: Número decimal que indica el ancho de la línea de contorno de los polígonos del mapa. El valor por defecto es `0.5`.
- `labelX`, `labelY`: Cadena de caracteres indicando la etiqueta del eje respectivo. El valor por defecto es `"Longitud"` y `"Latitud"` respectivamente.
- `xlim`, `ylim`: Vector numérico que especifica los límites de los ejes respectivos.
- `Legend`: Igual uso que `gg.Legend` en `Plot.Series()`
- `Labs`: Igual uso que `gg.Texto` en `Plot.Series()`. Con la adición del argumento `fill`, el cual especifica el título de leyenda. El valor por defecto es `"Statistic"`. Si es `NULL`, se omitirá dicho título.
- `Theme`: Igual uso que `gg.Tema` en `Plot.Series()`
- `Style`: Cadena de caracteres que indica el tipo de mapa a graficar. Los valores permitidos son `"Intervalo"`, `"SiNo"`, `"Calor"` y `NA` (*valor predeterminado*). Se emparejará parcialmente.
- `Text`: Lista que especifica aspectos como el color y tamaño de los títulos de los polígonos. Para más información, consulte la función geom_sf_text().
- `scaleX`, `scaleY`: Vector numérico que especifica los puntos o cortes a graficar en dicho eje. Sacado de la función scale_x_continuous().

`...` Abc.

Details

Los vectores `depto` y `mpio` introducidos hacen referencia a atributos atómicos, es decir, la pareja formada por `(depto, mpio)` debe corresponder a un individuo. En los argumentos no se acepta la entrada de objetos espaciales o polígonos.

Value

Retorna el mapa (*objeto widget de HTML*) creado mediante `Leaflet`, el cual pertenece a la clase "leaflet" y "htmlwidget".

Examples

```
# library(dplyr)
df <- ejGraduados |> filter(YEAR == 2021) |>
  select(
    COD_DEP_NAC, COD_CIU_NAC, DEP_NAC, CIU_NAC, LON_CIU_NAC, LAT_CIU_NAC
  )
Plot.Mapa(
  datos    = df,
  depto    = COD_DEP_NAC,
  mpio     = COD_CIU_NAC,
  tipo     = "SiNoMpios",
  titulo   = "Graduados 2021-I",
  baldosas = c(
    "Esri.WorldPhysical", "Esri.DeLorme", "Esri.WorldShadedRelief",
    "Esri.WorldTerrain", "Esri.OceanBasemap"
  ),
  colores  = c("#10F235", "#00BCB5"),
  colSedes = rep("green", 9),
  opacidad = 0.6,
  colBorde = "#FF4D00",
  compacto = FALSE,
  textSize = 16,
  limpio   = TRUE
)
# ------------------------------------------------------------------------
Plot.Mapa(
  datos  = df,
  depto  = COD_DEP_NAC,
  mpio   = COD_CIU_NAC,
  tipo   = "DeptoMpio",
  titulo = "Graduados 2021-I",
  cortes = list(
    Deptos = c(0, 10, 20, 50, 500, Inf), Mpios = c(0, 1, 5, 10, 100, Inf)
  ),
  colores = list(
    Deptos = c("#6812F2", "#5769F6", "#F6ED0D", "#EE6115", "#EC2525"),
    Mpios  = c("#E7F15D", "#ACBD37", "#E15E32", "#A82743", "#5C323E")
  )
)


# ------------------------------------------------------------------------
# library(dplyr); library(magrittr)
ejSaberPro2020 %>%
  select(
    Code_Dept = COD_DEP_NAC,
    Code_Mun  = COD_CIU_NAC,
```

```
    Edad      = EDAD_MOD,
    PBM       = PBM_ORIG,
    ScoreGlobal   = PUNTAJE_GLOBAL,
    ScoreCompCiud = PUNT_COMP_CIUD,
    ScoreComuEscr = PUNT_COMU_ESCR,
    ScoreIngles   = PUNT_INGLES,
    ScoreLectCrit = PUNT_LECT_CRIT,
    ScoreRazCuant = PUNT_RAZO_CUANT
  ) %$%
  Plot.Mapa(
    datos         = .,
    depto         = Code_Dept,
    mpio          = Code_Mun,
    estadistico   = "Mediana",
    variable      = ScoreGlobal,
    tipo          = "DeptoMpio",
    titulo        = "P.Global 2020",
    naTo0         = FALSE,
    colNA         = "#472985",
    centroideMapa = "ANTIOQUIA",
    zoomMapa      = 8,
    cortes        = list(
      Deptos = c(0, 155, 170, 180, 185, Inf), Mpios = c(0, 50, 178, 200, 250, Inf)
    ),
    colores       = list(
      Deptos = c("#FF7D5A", "#FDBD7D", "#E5DF73", "#63D2A8", "#0055A1"),
      Mpios  = c("#E7F15D", "#ACBD37", "#E15E32", "#A82743", "#5C323E")
    ),
    showSedes     = FALSE
  )


# ------------------------------------------------------------------------
# Ejemplo usando el caso estático (ggplot2)
# library(cowplot)
Plot.Mapa(
  datos     = df,
  depto     = COD_DEP_NAC,
  mpio      = COD_CIU_NAC,
  zoomIslas = TRUE,
  tipo      = "Mpios",
  titulo    = "N\u00daM. DE GRADUADOS \u00d7 MUNICIPIO",
  colNA     = "#4ACB46",
  cortes    = c(0, 1, 10, Inf),
  colores   = c("#009CC8", "#EE6115", "#EC2525"),
  opacidad  = 0.6,
  estatico  = TRUE,
  estilo    = list(
    Style = "Intervalo", Theme = 2, anchoBorde = 0.5,
    Legend = list(legend.position = "bottom", legend.direction = "horizontal"),
   Labs  = list(subtitle = "A\u00f1o 2021", caption = "(*) Lugar de Nacimiento", tag = "\u00ae")
  )
) -> listMaps
```

```
# library(cowplot); library(ggplot2)
# |    Tema    |             Mapa             |   y   | width |
# |:----------:|:----------------------------:|:----:|:-----:|
# | 1:6, 10:11 |          San Andrés          | 0.36 |  0.06 |
# |            | Providencia y Santa Catalina | 0.39 | 0.055 |
# |      9     |             ° ° °            | 0.36 |  0.07 |
# |            |             * * *            | 0.39 | 0.065 |
# |    12:14   |             ° ° °            | 0.33 |  0.11 |
# |            |             * * *            | 0.36 |  0.10 |
ggdraw() +
  draw_plot(listMaps$M_COL) +
  draw_plot(listMaps$M_SanAndres  , x = 0.27, y = 0.36, width = 0.060) +
  draw_plot(listMaps$M_Providencia, x = 0.33, y = 0.39, width = 0.055)
# ggplot2::ggsave("COL.png", width = 12, height = 10, dpi = 550)


# ------------------------------------------------------------------------
# library(tibble)
PIB <- tibble(
  DIVIPOLA = c(91,05,81,08,11,13,15,17,18,85,19,20,27,23,25,94,95,
               41,44,47,50,52,54,86,63,66,88,68,70,73,76,97,99),
  Valor    = c(883,177837,6574,52961,301491,NA,31208,19782,4718,17810,21244,
               23244,5069,20842,73592,443,943,19837,14503,16370,41923,18259,
               18598,4253,9837,19531,1711,74737,9842,25143,116238,337,799)
)
Plot.Mapa(
  datos     = PIB,
  depto     = DIVIPOLA,
  variable  = Valor,
  agregado  = FALSE,
  zoomIslas = TRUE,
  tipo      = "Deptos",
  naTo0     = FALSE,
  colNA     = "#543619",
  titulo    = "PIB \u00d7 DEPARTAMENTO",
  cortes    = c(0, 500, 5000, 20000, 30000, Inf),
  colores   = c("#FED600", "#02D46E", "#006389", "#FA006E", "#FC553C"),
  colBorde  = "#3A0F2D",
  estatico  = TRUE,
  textSize  = 0,
  estilo    = list(
    Style = "Intervalo", Theme = 7,
    Legend = list(legend.position = "bottom", legend.direction = "horizontal"),
    Labs = list(
      fill = "PIB", subtitle = "A Precios Corrientes",
      caption = "Para el periodo de 2021P (en miles de millones de pesos)"
    )
    # scaleX = seq(-82, -64, by = 1), scaleY = seq(-4, 14, by = 1)
  )
) -> listMaps
# Caso 1:
#   Dejando solamente la isla principal (San Andrés) y omitiendo
#   las contiguas (Providencia y Santa Catalina)
```

```
ggdraw() +
  draw_plot(listMaps$M_COL) +
  draw_plot(listMaps$M_SanAndres, x = 0.26, y = 0.8, width = 0.4)
# ggsave("COL.png", width = 12, height = 10, dpi = 550)
# Caso 2:
#  Conservando todo el departamento y sus distancias reales (no se modifica su polígono espacial)
ggdraw() +
  draw_plot(listMaps$M_COL) +
  draw_plot(listMaps$M_SanAndres, x = 0.27, y = 0.35, width = 0.08)
# ggsave("COL.png", width = 12, height = 10, dpi = 550)
# ------------------------------------------------------------------------
AreaVichada <- tibble(
  MunCode = c(99001, 99524, 99624, 99773), Area = c(12409, 20141, 2018, 65674)
)
Plot.Mapa(
  datos    = AreaVichada,
  mpio     = MunCode,
  variable = Area,
  agregado = FALSE,
  tipo     = "Mpios",
  titulo   = "\u00c1REA DE LOS MUNICIPIOS DEL\nDEPARTAMENTO DE VICHADA",
  naTo0    = FALSE,
  centroideMapa = "VICHADA",
  estatico = TRUE,
  estilo   = list(
    Style = "Calor", Theme = 9,
    Labs  = list(caption = "(*) En Metros Cuadrados"),
    Text  = list(color = "#011532", size = 3)
  )
)
```

`Plot.Mundo` *Cree un mapa dinámico y flexible para visualizar datos geográficos de países*

Description

Esta función está planeada para facilitar la creación de mapas interactivos compatible con plataformas móviles y de escritorio, además de estar diseñada pensando en la simplicidad y el rendimiento. Esta utilidad produce mapas que tienen controles para hacer zoom, desplazarse y alternar capas y puntos entre mostrar y ocultar. Igualmente, permite incrustar mapas en webs, documentos `R Markdown` y aplicaciones `Shiny`. Todo lo anterior basado enteramente en la librería `Leaflet`, la cual es la biblioteca `JavaScript` de código abierto más popular para mapas interactivos.

Usage

```
Plot.Mundo(
  datos,
```

```
  paises,
  variable,
  grupo,
  tipo = c("Pais", "SiNoPais"),
  titulo,
  naTo0 = TRUE,
  colNA = "#EEEEEE",
  centroideMapa,
  zoomMapa = 2,
  baldosas,
  cortes,
  colores,
  opacidad = 0.7,
  colBorde,
  compacto = TRUE,
  textSize = 10,
  limpio = FALSE,
  estatico = FALSE,
  estilo
)
```

Arguments

datos	Un data frame, no un vector numérico.
paises	Una variable dentro del data frame ingresado en datos, que contiene los códigos de los países, de acuerdo con la codificación ISO3166, ya sea el nombre o el código alpha2 o alpha3.
variable	Variable auxiliar con la cual se calculará el estadístico previamente seleccionado. Para el caso en que la estadística a calcular sea el conteo no es necesario (*no se usará*) especificar dicha variable numérica.
grupo	Variable auxiliar con la cual se segmentará los datos, para que se grafique en un único plot divido en parcelas por dicha variable.
tipo	Cadena de caracteres que indica el tipo de mapa a graficar. Los valores permitidos son "Deptos", "SiNoMpios", "Mpios" y "DeptoMpio" (*valor predeterminado*). Se emparejará parcialmente.
titulo	Cadena de caracteres indicando la segregación que presenta el mapa y el periodo al que hace referencia éste, separados por un espacio, por ejemplo, "Admitidos 2021-I".
naTo0	Si es TRUE (*valor predeterminado*) los valores introducidos como NA (*not available*) se cambiarán por el valor de 0. Ajústelo a FALSE para que no se realice tal cambio y mostrar en la caja de información la leyenda "Sin Información".
colNA	Cadena de caracteres indicando el color que tendrá la categoría NA (*si esta se presenta*). El valor por defecto es un gris muy claro.
centroideMapa	Cadena de caracteres indicando el país que servirá de centroide al momento de graficar el mapa. El valor por defecto es "MEXICO". Se emparejará parcialmente.

`zoomMapa` Valor numérico que indica el nivel de zoom del mapa (*usado por la función* setView()). El valor por defecto es `6`, entre mayor sea su valor más zoom se aplicará al mapa.

`baldosas` Vector de caracteres indicando los mapas base con los que se realizará el mapa, sean los popularizados por Google Maps o por terceros. Los valores aceptados son los admitidos por la función addProviderTiles(), así mismo los valores por defecto son `c("CartoDB.Positron", "Esri.WorldStreetMap", "Esri.NatGeoWorldMap")`, algunos otros valores pueden ser:

- "Esri.DeLorme"
- "Esri.WorldTerrain"
- "Esri.WorldShadedRelief"
- "Esri.WorldPhysical"
- "Esri.OceanBasemap"
- "Esri.WorldGrayCanvas"
- "Esri.WorldImagery"
- "Stamen.Toner"
- "Stamen.TonerLite"
- "Stamen.TonerLines"
- "Stamen.Watercolor"
- "Stamen.TonerHybrid"

La lista completa la puede consultar aquí

`cortes` Vector numérico indicando los cortes con los cuales se crearán los intervalos. No aplica para el tipo de mapa "SiNoMpios", pues este es binario. Para el tipo de mapa "DeptoMpio" se debe pasar una lista de la siguiente manera `list(Deptos = c(), Mpios = c())`, pues requiere dos cortes, uno para departamentos y otro para municipios.

`colores` Vector de caracteres indicando los colores para cada uno de los intervalos con los que cuenta el mapa. Si no se introduce algún vector, se usará una paleta predeterminada dependiendo del tipo de mapa.

`opacidad` Un número entre $[0, 1]$ que indica la opacidad de las capas.

`colBorde` Cadena de caracteres indicando el color del borde de los polígonos al momento de pasar el cursor sobre él.

`compacto` Si es `TRUE` (*valor predeterminado*) el control de capas se representará como un icono que se expande cuando se coloca el cursor sobre él. Ajústelo a `FALSE` para que el control de capas siempre aparezca en su estado expandido.

`textSize` Valor numérico que indica el tamaño del texto de las etiquetas de los municipios. El valor para los departamentos será $+2px$.

`limpio` Si es `FALSE` (*valor predeterminado*) se mostrará el MiniMapa, la barra de escala y los botones para ver en pantalla completa, retornar zoom y localización. Ajústelo a `TRUE` si desea omitir dichas herramientas adicionales al mapa.

`estatico` Si es `FALSE` (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con `ggplot2`.

`estilo` Lista compuesta por varios parámetros, los cuales van a ser usados para graficar el mapa **estático** y cuyo objetivo es personalizar pequeños detalles de éste.

- `anchoBorde`: Número decimal que indica el ancho de la línea de contorno de los polígonos del mapa. El valor por defecto es `0.5`.
- `labelX`, `labelY`: Cadena de caracteres indicando la etiqueta del eje respectivo. El valor por defecto es `"Longitud"` y `"Latitud"` respectivamente.
- `xlim`, `ylim`: Vector numérico que especifica los límites de los ejes respectivos.
- `Legend`: Igual uso que `gg.Legend` en `Plot.Series()`
- `Labs`: Igual uso que `gg.Texto` en `Plot.Series()`. Con la adición del argumento `fill`, el cual especifica el título de leyenda. El valor por defecto es `"Statistic"`. Si es `NULL`, se omitirá dicho título.
- `Theme`: Igual uso que `gg.Tema` en `Plot.Series()`
- `Style`: Cadena de caracteres que indica el tipo de mapa a graficar. Los valores permitidos son `"Intervalo"`, `"SiNo"`, `"Calor"` y `NA` (*valor predeterminado*). Se emparejará parcialmente.
- `Text`: Lista que especifica aspectos como el color y tamaño de los títulos de los polígonos. Para más información, consulte la función geom_sf_text().
- `scaleX`, `scaleY`: Vector numérico que especifica los puntos o cortes a graficar en dicho eje. Sacado de la función scale_x_continuous().

Value

Retorna el mapa (*objeto widget de HTML*) creado mediante `Leaflet`, el cual pertenece a la clase "leaflet" y "htmlwidget".

Examples

```
LATAM <- data.frame(
  Country = c("Chile", "Venezuela", "Colombia", "Argentina", "Brazil"),
  PIB = c(1, 10, 100, 1000, 10000)*1:5
)
Plot.Mundo(
  datos    = LATAM,
  paises   = Country,
  variable = PIB,
  tipo     = "Pais",
  titulo   = "PIB 2023-Q1",
  naTo0    = FALSE,
  colNA    = "#3DBC25",
  centroideMapa = "Peru",
  zoomMapa = 3,
  cortes   = c(0, 10, 100, 1000, 10000, Inf),
  colores  = c("#DFA86A", "#FFBB00", "#FF5100", "#F20034", "#76009D"),
  opacidad = 0.5,
  colBorde = "#0E9BEE",
  compacto = TRUE,
  textSize = 4,
  limpio   = FALSE
```

```
)
# ------------------------------------------------------------------------
Plot.Mundo(
  datos    = LATAM,
  paises   = Country,
  variable = PIB,
  tipo     = "SiNoPais",
  titulo   = "PIB 2023-Q1",
  centroideMapa = "Senegal",
  colores  = c("#45C9FF", "#FF153F"),
  opacidad = 0.6,
  colBorde = "#3DBC25",
  compacto = FALSE,
  textSize = 10,
  limpio   = FALSE
)
# ------------------------------------------------------------------------
set.seed(123)
LATAM <- data.frame(
  Pais = c("Bolivia", "Colombia", "Ecuador", "Panama", "Venezuela",
           "Peru", "Argentina", "Brazil", "Chile", "Uruguay"),
  PIB  = runif(10, -10, 10),
  Bolivariano = c("Sí", "Sí", "Sí", "Sí", "Sí", "Sí", "No", "No", "No", "No")
)
Plot.Mundo(
  datos    = LATAM,
  paises   = Pais,
  variable = Bolivariano,
  tipo     = "Pais",
  titulo   = "PA\u00cdSES BOLIVARIANOS EN AM\u00c9RICA LATINA",
  naTo0    = FALSE,
  estatico = TRUE,
  estilo   = list(Style = NA, Theme = 2)
)
# ------------------------------------------------------------------------
Plot.Mundo(
  datos    = LATAM,
  paises   = Pais,
  variable = PIB,
  # grupo  = Bolivariano,
  tipo     = "Pais",
  titulo   = "PROYECCI\u00d3N DEL PIB",
  opacidad = 0.4,
  colBorde = "#876445",
  estatico = TRUE,
  estilo   = list(
    Style = "Calor",
    showISO    = list(color = "#00468A", size = 3.5, fontface = "bold.italic"),
    xlim       = c(-82, -34),
    ylim       = c(-60, 14),
    scaleX     = seq(-82, 34, by = 8),
    scaleY     = seq(-60, 14, by = 5),
    anchoBorde = 1,
```

```
    Theme = 6, Legend = list(legend.position = "bottom", legend.direction = "horizontal"),
     Labs = list(subtitle = "Para Suram\u00e9rica en el 2023",
                 caption  = "Datos simulados para el ejemplo ilustrativo",
                 tag = "\u00ae"
                 )
  )
)
# ------------------------------------------------------------------------
Territories <- data.frame(
 Country = c("RUS", "FRA", "UKR", "ESP", "SWE", "DEU", "DZA", "COD", "SDN", "LBY", "TCD", "NER"),
  Area    = rep(10^(5:0), 2)
)
Plot.Mundo(
  datos    = Territories,
  paises   = Country,
  variable = Area,
  tipo     = "Pais",
  titulo   = "ALGUNOS DE LOS PA\u00cdSES CON MAYOR \u00c1REA",
  # showISO    = list(),
  naTo0    = FALSE,
  colNA    = "#0C1C2B",
  cortes   = c(-.0001, 10, 100, 1000, 10000, Inf),
  colores  = c("#DFA86A", "#FFBB00", "#FF5100", "#F20034", "#76009D"),
  opacidad = 0.4,
  limpio   = TRUE,
  estatico = TRUE,
  estilo   = list(
    Style = "Intervalo", continente = c("Africa", "Europe"), Theme = 4
  )
)
# ------------------------------------------------------------------------
COL <- data.frame(Pais = "CO", IDH = 100000)
Plot.Mundo(
  datos    = COL,
  paises   = Pais,
  variable = IDH,
  tipo     = "Pais",
  titulo   = "",
  colores  = c("#10F235", "#00BCB5"),
  naTo0    = TRUE,
  estatico = TRUE,
  estilo   = list(Style = "SiNo", Theme = 3)
)
```

`Plot.Radar` *Cree un gráfico de radar dinámico y flexible con dos diferentes paquetes*

Description

Esta función proporciona excelentes herramientas y opciones para la visualización de un gráfico de radar (*también conocido como gráfico de araña*) dinámico con el objetivo de observar datos multivariados de forma bidimensional. Dicho radar chart o spider plot se puede representar usando dos diferentes librerías que son `Plotly` y `ECharts`, las cuales usan internamente `JavaScript`.

Usage

```
Plot.Radar(
  datos,
  categoria,
  variables,
  estadistico = c("Promedio", "Mediana", "Varianza", "SD", "CV", "Min", "Max"),
  colores,
  rango,
  ordinal = FALSE,
  titulo = "",
  libreria = c("plotly", "echarts"),
  estilo = NULL,
  estatico = FALSE
)
```

Arguments

`datos`	Un data frame, se espera en formato de microdatos no un agregado.
`categoria`	Una variable categórica dentro del data frame ingresado en `datos`.
`variables`	Lista (*ya sea creada con la sintaxis* `base` *o* `tidy`) con las variables numéricas (*mínimo tres para que se pueda realizar el gráfico*) dentro del data frame ingresado en `datos`.
`estadistico`	Cadena de caracteres que indica el estadístico a graficar. Los valores permitidos son `"Promedio"` (*valor predeterminado*), `"Mediana"`, `"Varianza"`, `"SD"`, `"CV"`, `"Min"` y `"Max"`.
`colores`	Cadena de caracteres indicando los colores con los cuales se deben colorear cada una de las trazas correspondiente a cada nivel del argumento `categoria`. Si no se introduce algún vector se usará la paleta `rainbow` por defecto.
`rango`	Vector numérico de longitud dos que indica el valor mínimo y máximo, respectivamente. Si no conoce el dominio del estadístico seleccionado omita éste parámetro, pues internamente se usará `c(0, NaN)` como rango.
`ordinal`	Si es `TRUE` indicará que las categorías de la variable ingresada son ordinales (*no nominales*), esto con el fin de ordenar la disposición en el que se presentan en el eje del gráfico, el valor por defecto es `FALSE`.
`titulo`	Cadena de caracteres indicando el título principal del plot.
`libreria`	Cadena de caracteres que indica el paquete con el cual se realizará el radar. Los valores permitidos son `"plotly"` (*valor predeterminado*) o `"echarts"`. Los valores se emparejarán parcialmente.

`estilo` Lista compuesta por varios parámetros, los cuales van a ser usados de acuerdo con la librería especificada para graficar el radar y cuyo objetivo es personalizar pequeños detalles de éste.

- `ply.LegendTitle`, `ply.LegendPosition` y `ply.Credits`: Igual uso que en `Plot.Series()`
- `ply.Relleno`: Cadena de caracteres indicando cómo se debe rellenar el área, `toself` (*valor predeterminado*) conecta los puntos de la traza de forma cerrada y superpone las áreas, mientras que `tonext` deja visible la capa más profunda y si se comparten áreas no las superpone; finalmente especifique `none` si desea ver únicamente los polígonos y que no se rellene el área dentro de ellos.
- `ply.Opacidad`: Un número entre $[0, 1]$ que indica la opacidad de los polígonos/trazos.
- `e.Credits`: Cadena de caracteres indicando el subtítulo del gráfico principal. Para mayor información, consulte la función e_title().
- `e.Forma`: Cadena de caracteres indicando el tipo de renderizado del radar, los valores admitidos son `polygon` (*valor predeterminado*) y `circle`. Para mayor información, consulte la función e_radar_opts().
- `e.Tema`: Modifica el tema con el cual se creará el gráfico. Los posibles valores son un número entero entre $[1, 14]$ el cual hace referencia a diferentes temas disponibles en dicha librería (`helianthus`, `azul`, `inspired`, `macarons`, `westeros`, `walden`, `roma`, `royal`, `fruit`, `dark`, `chalk`, `purple-passion`, `vintage` y `essos` respectivamente). El tema por defecto se logra al no ingresar valor alguno. Para más información consulte aquí.
- `e.LegType`: Cadena de caracteres indicando el tipo de leyenda, los valores admitidos son `plain` (*valor predeterminado*) y `scroll` (*útil cuando es necesario mostrar demasiados elementos*). Para mayor información consulte la función e_legend().
- `e.LegLoc`: Valor numérico o cadena de caracteres indicando la distancia entre la leyenda y el lado derecho del contenedor, puede ser expresado como un valor puntual o un valor porcentual relativo al ancho del contenedor.
- `gg.Range`: Valor booleano opcional, si se especifica en `TRUE` el rango a tomar será simétrico, en el sentido en que se tomará el mínimo y máximo global de todas las variables, uno mismo para cada una de ellas. Diferente a si se omite el parámetro `rango`, pues acá el mínimo y máximo varía para cada variable.
- `gg.plty`: Tipo de línea para los datos del gráfico. Para más detalles consulte la función radarchart().
- `gg.plwd`: Ancho de la línea para los datos del gráfico. Para más detalles consulte la función radarchart().
- `gg.cglwd`: Ancho de la línea para las grillas del radar. Para más detalles consulte la función radarchart().
- `gg.cglcol`: Color de la línea para las grillas del radar. Para más detalles consulte la función radarchart().

`estatico` Si es `FALSE` (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con `ggplot2`.

Value

Retorna el radar (*objeto widget de HTML*) creado. La clase del objeto retornado será un "htmlwidget" y dependiendo de la librería usada pertenecerá adicionalmente a la clase "plotly" o "echarts4r".

Lista de argumentos de estilo

Sabemos que puede ser abrumador el número de argumentos dentro del parámetro `estilo`, pero es necesario si queremos ofrecer al usuario la máxima personalización dentro de cada función usando cualquier librería. Por tal razón, a continuación, se detalla el listado completo de argumentos, usados al especificar la librería y en qué función están presentes (*marcado con una* × *si lo posee*).

Librería	**estilo$**	`Plot.Series()`	`Plot.Barras()`	`Plot.Apiladas()`	`Plot.Boxplot()`	P
—	*gg.Tema*	×	×	×	×	
l	*gg.Texto*	×	×	×	×	
l	*gg.Legend*	×		×	×	
l	*gg.Linea*	×				
l	*gg.Punto*	×				
l	*gg.Bar*		×	×		
l	*gg.VarWidth*				×	
l	*gg.OutShape*				×	
l	*gg.JitWidth*				×	
l	*gg.JitSize*				×	
l	*gg.Range*					
ggplot2	*gg.plty*					
l	*gg.plwd*					
l	*gg.cglwd*					
l	*gg.cglcol*					
l	*gg.fontsize.title*					
l	*gg.fontsize.labels*					
l	*gg.fontcolor.labels*					
l	*gg.border.lwds*					
l	*gg.border.col*					
l	*gg.lowerbound.cex.labels*					
l	*gg.force.print.labels*					
—	*gg.overlap.labels*					
»	*hc.Tema*	×	×	×	×	
l	*hc.Credits*	×	×	×	×	
highcharter	*hc.BoxInfo*	×				
l	*hc.Slider*	×				
»	*hc.borderRadius*					
•	*ply.Credits*	×	×	×	×	
◦	*ply.Legend*		×			
◦	*ply.LegendPosition*	×		×	×	
plotly	*ply.Interaction*	×			×	
◦	*ply.Relleno*					
◦	*ply.Opacidad*					
•	*ply.LegendTitle*					
dygraphs	*dyg.LegendWidth*	×				

»	*dyg.Resaltar*	×
—	*e.Tema*	
l	*e.Credits*	
echarts4r	*e.Forma*	
l	*e.LegType*	
—	*e.LegLoc*	

Examples

```
# library(dplyr)
Plot.Radar(
  datos     = ejSaberPro2020,
  categoria = TIPO_COL,
  variables = vars(PUNT_LECT_CRIT, PUNT_RAZO_CUANT, PUNT_INGLES),
  colores   = c("#2ACE82", "#FE2667", "#32E7C8", "#FF8D00"),
  rango     = c(0, NaN),
  estilo    = list(ply.Relleno = "tonext")
)
# ------------------------------------------------------------------------
Plot.Radar(
  datos     = ejSaberPro2020,
  categoria = SEDE_NOMBRE_ADM,
  variables = vars(
    PUNTAJE_GLOBAL, PUNT_RAZO_CUANT, PUNT_INGLES,
    PUNT_LECT_CRIT, PUNT_COMP_CIUD, PUNT_COMU_ESCR
  ),
  rango     = c(0, NaN)
)
# ------------------------------------------------------------------------
Plot.Radar(
  datos     = ejSaberPro2020,
  categoria = SEDE_NOMBRE_ADM,
  variables = vars(
    PUNTAJE_GLOBAL, PUNT_RAZO_CUANT, PUNT_INGLES,
    PUNT_LECT_CRIT, PUNT_COMP_CIUD, PUNT_COMU_ESCR
  ),
  rango     = c(0, NaN),
  libreria  = "echarts"
)
# ------------------------------------------------------------------------
misColores <- c(
  "#29ABE2", # AZUL CLARO  | Amazonia
  "#8CC63F", # VERDE       | Bogota
  "#CC241D", # ROJO        | Caribe
  "#0071BC", # AZUL VIVO   | Manizales
  "#F15A24", # NARANJA     | Medellin
  "#FBB03B", # AMARILLO    | Orinoquia
  "#93278F", # MORADO      | Palmira
  "#8A381A" # GRIS        | Tumaco
)
Msj <- "Gr\u00e1fico de radar para representar los puntajes multivariados de la prueba Saber Pro."
```

```
Plot.Radar(
  datos       = ejSaberPro2020,
  categoria   = SEDE_NOMBRE_ADM,
  variables   = vars(
    PUNTAJE_GLOBAL, PUNT_RAZO_CUANT, PUNT_INGLES,
    PUNT_LECT_CRIT, PUNT_COMP_CIUD, PUNT_COMU_ESCR
  ),
  estadistico = "SD",
  colores     = misColores,
  rango       = c(0, NaN),
  titulo      = "SPIDER PLOT",
  libreria    = "plotly",
  estilo      = list(
   ply.LegendTitle = "SEDE:", ply.LegendPosition = list(x = 0, y = -0.15, orientation = "h"),
   ply.Relleno = "tonext", ply.Opacidad = 0.8, ply.Credits = list(x = 0.8, y = -0.1, text = Msj)
  )
)
# ------------------------------------------------------------------------
Plot.Radar(
  datos       = ejSaberPro2020,
  categoria   = SEDE_NOMBRE_ADM,
  variables   = vars(
    PUNTAJE_GLOBAL, PUNT_RAZO_CUANT, PUNT_INGLES,
    PUNT_LECT_CRIT, PUNT_COMP_CIUD, PUNT_COMU_ESCR
  ),
  estadistico = "CV",
  colores     = misColores,
  rango       = c(0, 0.25),
  titulo      = "RADAR CHART",
  libreria    = "echarts",
  estilo      = list(
    e.Credits = Msj, e.Forma = "circle", e.Tema = 10,
    e.LegType = "scroll", e.LegLoc = 0
  )
)
# ------------------------------------------------------------------------
# Ejemplo usando el caso estático (fmsb)
Plot.Radar(
  datos     = ejSaberPro2020,
  categoria = TIPO_COL,
  variables = vars(
    PUNT_RAZO_CUANT, PUNT_INGLES, PUNT_LECT_CRIT, PUNT_COMP_CIUD, PUNT_COMU_ESCR
  ),
  estadistico = "SD",
  colores  = c("#89D8FF", "#9CFF86", "#FFA568", "#FF7F7F"),
  titulo   = "RADAR CHART DE LA DESVIACI\u00d3N EST\u00c1NDAR\nPOR COMPONENTE EVALUADO",
  # rango    = c(10, 40),
  estatico = TRUE,
  estilo   = list(
    gg.Range = TRUE, gg.plty = 5, gg.plwd = 4, gg.cglwd = 2, gg.cglcol = "#856AA1"
  )
)
```

`Plot.Series` *Cree una serie de tiempo dinámica/estática y flexible con tres diferentes paquetes*

Description

Esta función proporciona excelentes herramientas y opciones para la visualización de series de tiempo dinámicas con el objetivo de estudiar la evolución de una o varias variables a lo largo del tiempo. Dicha serie interactiva se puede representar usando tres diferentes librerías que son `Highcharter`, `Plotly` y `Dygraph`, las cuales usan internamente `JavaScript`.

Usage

```
Plot.Series(
  datos,
  tiempo,
  valores,
  categoria,
  freqRelativa = FALSE,
  invertir = FALSE,
  ylim,
  colores,
  titulo = "",
  labelX = "Periodo",
  labelY = "",
  libreria = c("highcharter", "plotly", "dygraphs"),
  estilo = NULL,
  estatico = FALSE
)
```

Arguments

`datos`	Un data frame, no un objeto clase serie de tiempo o vector numérico.
`tiempo`	Lista de variable(s) tanto numéricas como categóricas que se concatenaran para crear un único periodo temporal (*ordenado ascendentemente*).
`valores`	Variable numérica que contiene los valores que desea graficar.
`categoria`	Una variable categórica dentro del data frame ingresado en `datos`.
`freqRelativa`	Si es `FALSE` (*valor predeterminado*) la serie graficada representará las frecuencias absolutas (*conteo*) más no las relativas (*porcentaje*).
`invertir`	Si es `FALSE` (*valor predeterminado*) no se invertirá el eje `Y`. Establézcalo en `TRUE` si desea que en el eje `Y` el número más alto sea el más cercano al origen.
`ylim`	Vector numérico que especifica el límite inferior y superior, respectivamente, del eje `Y`. Si no se introduce algún valor se mostrará todo el rango disponible para dicho eje.

`colores` Cadena de caracteres indicando los colores con los cuales se deben colorear cada una de las series correspondiente a cada nivel del argumento `categoria`. Si no se introduce algún vector se usará la paleta `rainbow` por defecto.

`titulo` Cadena de caracteres indicando el título principal del plot.

`labelX` Cadena de caracteres indicando la etiqueta del eje `X`. Por defecto se emplea el rótulo "Periodo".

`labelY` Cadena de caracteres indicando la etiqueta del eje `Y`.

`libreria` Cadena de caracteres que indica el paquete con el cual se realizará la serie. Los valores permitidos son `"highcharter"` (*valor predeterminado*), `"plotly"` o `"dygraphs"`. Los valores se emparejarán parcialmente.

`estilo` Lista compuesta por varios parámetros, los cuales van a ser usados de acuerdo con la librería especificada para graficar la serie y cuyo objetivo es personalizar pequeños detalles de ésta.

- `LegendTitle`: Cadena de caracteres indicando un título para la leyenda (*diferentes niveles del argumento* `categorias`). Se utilizará tanto en el paquete `Highcharter` como en `Plotly`.
- `hc.Tema`: Modifica el tema con el cual se creará la serie. Los posibles valores son un número entero entre $[1, 10]$ el cual hace referencia a diferentes temas disponibles en dicha librería (`ffx`, `google`, `tufte`, `538`, `ggplot2`, `economist`, `sandsignika`, `ft`, `superheroes` y `flatdark`, respectivamente). El tema por defecto, al no ingresar valor alguno, es `hc_theme_flat()`.
- `hc.Slider`: Si es `TRUE` agrega un deslizador/navegador dinámico en la parte inferior de la serie. Proporciona herramientas para acercar y alejar partes de la serie, así como para desplazarse por el conjunto de datos. El valor por defecto es `FALSE`.
- `hc.BoxInfo`: Si es `TRUE` (*valor predeterminado*) la información concerniente a cada punto se visualiza conjuntamente en un cuadro, o de forma individual (`FALSE`) al pasar el cursor sobre él.
- `hc.Credits`: Cadena de caracteres indicando un subtítulo o etiqueta de créditos debajo del título principal.
- `ply.LegendPosition`: Lista que especifica la posición y orientación de la leyenda. Los valores por defecto la ubican centrada verticalmente a la derecha del plot, es decir, `c(x = 1, y = 0.5, orientation = "v")`.
- `ply.Interaction`: Cadena de caracteres que determina el modo de las interacciones de desplazamiento. Los valores permitidos son `"x unified"` (*valor predeterminado*), `"y unified"`, `"closest"`, `"x"`, `"y"` y `FALSE`.
- `ply.Credits`: Lista que especifica la posición y texto para añadir un subtítulo o etiqueta de créditos a la serie principal, por ejemplo, `c(x = 0.2, y = 1, text = "https://...")`.
- `dyg.LegendWidth`: Número que indica el ancho (*en píxeles*) que ocupará la leyenda. El valor por defecto es `250`.
- `dyg.Resaltar`: Si es `FALSE` (*valor predeterminado*) no se resaltará la serie en que se sitúa el cursor.
- `gg.Tema`: Modifica el tema con el cual se creará la serie. Los posibles valores son un número entero entre $[1, 11]$ el cual hace referencia a diferentes

temas disponibles para ggplot2 (theme_light, theme_bw, theme_classic, theme_linedraw, theme_gray, theme_hc, theme_pander, theme_gdocs, theme_fivethirtyeight, theme_economist y theme_solarized respectivamente). El tema por defecto, al no ingresar valor alguno, es el construido por el departamento theme_DNPE.

- gg.Legend: Lista que especifica la posición y orientación de la leyenda. Los valores por defecto la ubican verticalmente a la derecha del plot. Algunos valores aceptados para legend.position son "none", "left", "top", "right", "bottom" y c(CoordX, CoordY). Para legend.direction solo se acepta "vertical" u "horizontal".
- gg.Linea: Una lista de parámetros admitidos por la función geom_line()).
- gg.Punto: Una lista de parámetros admitidos por la función geom_point())
- gg.Texto: Una lista cuyos valores admitidos y usados son subtitle, caption y tag.
- gg.Repel: Una lista de parámetros admitidos por la función geom_text_repel())

estatico — Si es FALSE (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con ggplot2.

Details

Al usar el paquete Highcharter y usar las opciones de descarga, el nombre del archivo descargado será la concatenación del plot graficado y la categoría usada, así, por ejemplo, si se graficó la serie de tiempo para la categoría "Sede" el nombre será PlotSeries_Sede.png.

Tenga en cuenta que la librería "dygraphs" solo la podrá usar si dentro del argumento tiempo ingresa las dos variables (YEAR, SEMESTRE) para asemejar su estructura a los agregados clásicos. En caso contrario le arrojara un error.

Recuerde que puede usar más temas (*cualquiera de hecho*) de los que se proporcionan para ggplot2. Por ejemplo, los de hrbrthemes o ggtech.

Value

Retorna la serie (*objeto widget de HTML*) creada. La clase del objeto retornado será un "htmlwidget" y dependiendo de la librería usada pertenecerá adicionalmente a la clase "highchart", "plotly" o "dygraphs".

Lista de argumentos de estilo

Sabemos que puede ser abrumador el número de argumentos dentro del parámetro estilo, pero es necesario si queremos ofrecer al usuario la máxima personalización dentro de cada función usando cualquier librería. Por tal razón, a continuación, se detalla el listado completo de argumentos, usados al especificar la librería y en qué función están presentes (*marcado con una* × *si lo posee*).

Librería	estilo$	Plot.Series()	Plot.Barras()	Plot.Apiladas()	Plot.Boxplot()	P
—	*gg.Tema*	×	×	×	×	
l	*gg.Texto*	×	×	×	×	
l	*gg.Legend*	×		×	×	

l	*gg.Linea*	×			
l	*gg.Punto*	×			
l	*gg.Bar*		×	×	
l	*gg.VarWidth*				×
l	*gg.OutShape*				×
l	*gg.JitWidth*				×
l	*gg.JitSize*				×
l	*gg.Range*				
ggplot2	*gg.plty*				
l	*gg.plwd*				
l	*gg.cglwd*				
l	*gg.cglcol*				
l	*gg.fontsize.title*				
l	*gg.fontsize.labels*				
l	*gg.fontcolor.labels*				
l	*gg.border.lwds*				
l	*gg.border.col*				
l	*gg.lowerbound.cex.labels*				
l	*gg.force.print.labels*				
—	*gg.overlap.labels*				
»	*hc.Tema*	×	×	×	×
l	*hc.Credits*	×	×	×	×
highcharter	*hc.BoxInfo*	×			
l	*hc.Slider*	×			
»	*hc.borderRadius*				
•	*ply.Credits*	×	×	×	×
◦	*ply.Legend*		×		
◦	*ply.LegendPosition*	×		×	×
plotly	*ply.Interaction*	×			×
◦	*ply.Relleno*				
◦	*ply.Opacidad*				
•	*ply.LegendTitle*				
dygraphs	*dyg.LegendWidth*	×			
»	*dyg.Resaltar*	×			
—	*e.Tema*				
l	*e.Credits*				
echarts4r	*e.Forma*				
l	*e.LegType*				
—	*e.LegLoc*				

Note

A continuación, se consolida en una tabla amigable el listado, uso y disposición de todas las opciones para el parámetro `estilo`, dependiendo del tipo de gráfico (*dinámico o estático*) y la librería usada (*en el caso de que sea dinámico*).

PARÁMETRO VALOR PARÁMETRO VALOR PARÁMETRO

	*		•	gg.Tema
	*		•	gg.Legend
	TRUE		•	gg.Linea
	*		•	gg.Punto
	*		•	gg.Texto
	*		•	gg.Repel
	O	~	*highcharter*	hc.Tema
estatico	O	_	¦	hc.BoxInfo
	O	_	¦	hc.Slider
	O	_	¦	hc.Credits
	FALSE	**libreria**	*plotly*	ply.LegendPosition
	O	_	°	ply.Credits
	O	_	°	ply.Interaction
	O	_	*dygraphs*	dyg.LegendWidth
	O	~	L	dyg.Resaltar

Examples

```
# library("tibble"); library("dplyr")
set.seed(42)
Blood <- tibble(
  Year    = rep(2000:2001, each = 100),
  Quarter = sample(c("I", "II", "III", "IV"), size = 200, replace = TRUE),
  Week    = sample(c("1rt", "2nd", "3rd"), size = 200, replace = TRUE),
  Group   = sample(
    c("O", "A", "B", "AB"), size = 200, prob = c(.5, .3, .16, .4), replace = TRUE
  ),
  RH      = sample(c("+", "-"), size = 200, replace = TRUE),
  Prevalence = round(runif(200)*100)
)
Plot.Series(
  datos     = Blood,
  tiempo    = vars(Year, Quarter, Week),
  valores   = Prevalence,
  categoria = RH,
  labelX    = ""
)
Plot.Series(
  datos     = Blood,
  tiempo    = vars(Year, Quarter),
  valores   = Prevalence,
  categoria = Group,
  libreria  = "plotly"
)
# ------------------------------------------------------------------------
misColores <- c(
  "#29ABE2", # AZUL CLARO  | Amazonia
  "#8CC63F", # VERDE       | Bogota
  "#CC241D", # ROJO        | Caribe
  "#0071BC", # AZUL VIVO   | Manizales
```

```
  "#F15A24", # NARANJA     | Medellin
  "#FBB03B", # AMARILLO    | Orinoquia
  "#93278F", # MORADO      | Palmira
  "#8A381A"  # GRIS        | Tumaco
)
Msj <- "Distribuci\u00f3n de estudiantes graduados (desde el 2009-I al 2021-I) por sede."
Txt <- "EVOLUCI\u00d3N DEL N\u00daMERO DE GRADUADOS POR SEDE"
Plot.Series(
  datos        = ejConsolidadoGrad,
  categoria    = "SEDE_NOMBRE_ADM",
  freqRelativa = TRUE,
  ylim         = c(0, 75),
  colores      = misColores,
  titulo       = Txt,
  labelY       = "Frecuencia Relativa<br>(% de graduados)",
  libreria     = "highcharter",
 estilo     = list(LegendTitle = "SEDE:", hc.Tema = 10, hc.Slider = TRUE, hc.Credits = Msj)
)
# ------------------------------------------------------------------------
Plot.Series(
  datos     = ejConsolidadoGrad,
  categoria = "SEDE_NOMBRE_ADM",
  invertir  = TRUE,
  colores   = misColores,
  titulo    = Txt,
  labelY    = "N\u00famero de Graduados",
  libreria  = "plotly",
  estilo    = list(
    LegendTitle = "SEDE:", ply.Interaction = "closest",
    ply.LegendPosition = list(x = 0.16, y = -0.25, orientation = "h"),
    ply.Credits = list(x = 0.5, y = 0.1, text = Msj)
  )
)
# ------------------------------------------------------------------------
Plot.Series(
  datos     = ejConsolidadoGrad,
  categoria = "SEDE_NOMBRE_ADM",
  colores   = misColores,
  titulo    = Txt,
  labelY    = "N\u00famero de Graduados (k: miles)",
  libreria  = "dygraphs",
  estilo    = list(dyg.LegendWidth = 650, dyg.Resaltar = TRUE)
)


# ------------------------------------------------------------------------
# Agrupando para eliminar el semestre
# library("dplyr")
df <- ejConsolidadoGrad |> group_by(Variable, YEAR, Clase) |>
  summarise(Total = sum(Total, na.rm = TRUE), .groups = "drop")

Msj <- "Comportamiento anual, considerando ambos semestres (exceptuando el caso del 2021)."
Plot.Series(
```

```
  datos     = df,
  categoria = "SEXO",
  ylim      = c(1000, 6000),
  colores   = c("#3360FF", "#F30081"),
  titulo    = "EVOLUCI\u00d3N DEL N\u00daMERO DE GRADUADOS POR SEXO",
  labelX    = "A\u00f1o",
  labelY    = "N\u00famero de Graduados",
  libreria  = "highcharter",
  estilo    = list(hc.Tema = 1, hc.Credits = Msj)
)


# ------------------------------------------------------------------------
# Ejemplo usando el caso estático (ggplot2)
# library("magick"); library("cowplot")
txtA <- "EVOLUCI\u00d3N DEL N.\u00ba DE GRADUADOS \u00d7 SEDE"
txtB <- "\nComportamiento anual (exceptuando el caso del 2021)."
fig1 <- Plot.Series(
  datos        = ejConsolidadoGrad,
  categoria    = "SEDE_NOMBRE_ADM",
  freqRelativa = FALSE,
  invertir     = FALSE,
  ylim         = c(100, 2000),
  colores      = misColores,
  titulo       = txtA,
  labelY       = "N\u00famero de Graduados",
  estatico     = TRUE,
  estilo       = list(
    LegendTitle = "SEDE:", gg.Tema = 8,
    gg.Legend = list(legend.position = "bottom", legend.direction = "horizontal"),
    gg.Linea  = list(linetype = 2, size = 0.1, arrow = grid::arrow()),
    gg.Punto  = list(alpha = 0.2, shape = 21, size = 2, stroke = 5),
    gg.Texto  = list(
    subtitle = txtB, caption = "\t\t Informaci\u00f3n Disponible desde 2009-1", tag = "\u00ae"
    )
  )
)
# A continuación, se detalla el caso en el que quiera adicionar un logo a 'fig1'
# library("ggplot2"); library("magick"); require("cowplot")
URL <- "https://upload.wikimedia.org/wikipedia/commons/1/1e/UNAL_Logosimbolo.svg"
LogoUN <- magick::image_read_svg(URL)
ggdraw() +
  draw_image(LogoUN, scale = 0.15, x = 0.15, hjust = 1, halign = 1, valign = 0) +
  draw_plot(fig1 + theme(legend.background = element_blank(),
                         panel.background = element_blank(),
                         plot.background = element_blank()
                         )
  )


# ------------------------------------------------------------------------
# A continuación, se detalla el caso en el que quiera anotaciones textuales repulsivas
#   * (1) Espacio vacío que se debe respetar alrededor de la caja delimitadora
```

```
#   * (2) Espacio vacío que se debe respetar alrededor de cada punto
#   * (3) Entre más bajo más flechas, entre más distancia menos flechas
Plot.Series(
  datos        = ejConsolidadoGrad,
  categoria    = "SEDE_NOMBRE_ADM",
  freqRelativa = FALSE,
  invertir     = FALSE,
  ylim         = c(100, 2000),
  colores      = misColores,
  titulo       = "EVOLUCI\u00d3N DEL N.\u00ba DE GRADUADOS \u00d7 SEDE",
  labelY       = "N\u00famero de Graduados",
  estatico     = TRUE,
  estilo       = list(
    gg.Tema  = 1,
    gg.Repel = list(
      direction = "both", seed = 42, nudge_y = 0.25,
      arrow = arrow(length = unit(0.01, "npc")), segment.colour = "#4C716B",
      box.padding   = 0.5 ,     # (1)
      point.padding = 0.25,     # (2)
      min.segment.length = 0.45 # (3)
    )
  )
)
```

`Plot.Torta` *Cree un gráfico circular/torta/pie dinámico/estático y flexible con dos diferentes paquetes*

Description

Esta función permite mostrar de forma interactiva (*y estática*) una descripción compacta y general de una variable con sus respectivas categorías. Dicho diagrama se puede representar usando dos diferentes librerías que son `Highcharter` y `Plotly`, las cuales usan internamente `JavaScript`.

Usage

```
Plot.Torta(
  datos,
  valores,
  categoria,
  ano,
  periodo,
  colores,
  titulo = "",
  label = "",
  addPeriodo = FALSE,
  libreria = c("highcharter", "plotly"),
  estilo = NULL,
```

```
  estatico = FALSE
)
```

Arguments

datos	Un data frame, no un vector numérico.
valores	Variable numérica que contiene los valores que desea graficar.
categoria	Una variable categórica dentro del data frame ingresado en datos.
ano	Argument deprecated. This Argument still exist but will be removed in the next version.
periodo	Argument deprecated. This Argument still exist but will be removed in the next version.
colores	Cadena de caracteres indicando los colores con los cuales se deben colorear cada una de las series correspondiente a cada nivel del argumento categoria. Si no se introduce algún vector se usará la paleta rainbow por defecto.
titulo	Cadena de caracteres indicando el título principal del plot.
label	Cadena de caracteres indicando la etiqueta a la que hace referencia el plot.
addPeriodo	Argument deprecated. This Argument still exist but will be removed in the next version.
libreria	Cadena de caracteres que indica el paquete con el cual se realizará el plot. Los valores permitidos son "highcharter" (*valor predeterminado*) y "plotly". Los valores se emparejarán parcialmente.
estilo	Lista compuesta por varios parámetros, los cuales van a ser usados de acuerdo con la librería especificada para graficar la torta y cuyo objetivo es personalizar pequeños detalles de ésta. • LegendTitle, hc.Tema, hc.Credits y ply.Credits: Igual uso que en Plot.Series() • ply.Legend: Igual uso que en Plot.Barras()
estatico	Si es FALSE (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con ggplot2.

Details

Al usar el paquete Highcharter y usar las opciones de descarga, el nombre del archivo descargado será la concatenación del plot graficado y la categoría usada, así, por ejemplo, si se graficó el diagrama de pie para la categoría "Sexo" el nombre será PlotTorta_Sexo.png.

Value

Retorna el diagrama circular (*objeto widget de HTML*) creado. La clase del objeto retornado será un "htmlwidget" y dependiendo de la librería usada pertenecerá adicionalmente a la clase "highchart" o "plotly".

Lista de argumentos de estilo

Sabemos que puede ser abrumador el número de argumentos dentro del parámetro `estilo`, pero es necesario si queremos ofrecer al usuario la máxima personalización dentro de cada función usando cualquier librería. Por tal razón, a continuación, se detalla el listado completo de argumentos, usados al especificar la librería y en qué función están presentes (*marcado con una* × *si lo posee*).

Librería	**estilo$**	`Plot.Series()`	`Plot.Barras()`	`Plot.Apiladas()`	`Plot.Boxplot()`	P
—	*gg.Tema*	×	×	×	×	
l	*gg.Texto*	×	×	×	×	
l	*gg.Legend*	×		×	×	
l	*gg.Linea*	×				
l	*gg.Punto*	×				
l	*gg.Bar*		×	×		
l	*gg.VarWidth*				×	
l	*gg.OutShape*				×	
l	*gg.JitWidth*				×	
l	*gg.JitSize*				×	
l	*gg.Range*					
ggplot2	*gg.plty*					
l	*gg.plwd*					
l	*gg.cglwd*					
l	*gg.cglcol*					
l	*gg.fontsize.title*					
l	*gg.fontsize.labels*					
l	*gg.fontcolor.labels*					
l	*gg.border.lwds*					
l	*gg.border.col*					
l	*gg.lowerbound.cex.labels*					
l	*gg.force.print.labels*					
—	*gg.overlap.labels*					
»	*hc.Tema*	×	×	×	×	
l	*hc.Credits*	×	×	×	×	
highcharter	*hc.BoxInfo*	×				
l	*hc.Slider*	×				
»	*hc.borderRadius*					
•	*ply.Credits*	×	×	×	×	
◦	*ply.Legend*		×			
◦	*ply.LegendPosition*	×		×	×	
plotly	*ply.Interaction*	×			×	
◦	*ply.Relleno*					
◦	*ply.Opacidad*					
•	*ply.LegendTitle*					
dygraphs	*dyg.LegendWidth*	×				
»	*dyg.Resaltar*	×				
—	*e.Tema*					
l	*e.Credits*					
echarts4r	*e.Forma*					
l	*e.LegType*					

— *e.LegLoc*

Note

Los gráficos circulares son una forma muy mala de mostrar información. El ojo es bueno para juzgar medidas lineales y malo para juzgar áreas relativas. Un gráfico de barras o un gráfico de puntos es una forma preferible de mostrar este tipo de datos.

Examples

```
# Ejemplo generalizado (sin uso de un consolidado como input)
# library("tibble"); library("dplyr")
set.seed(42)
Blood <- tibble(
 Group = sample(c("O", "A", "B", "AB"), size = 200, prob = c(0.5, 0.3, 0.16, 0.4), replace = TRUE),
  Prevalence = round(runif(200)*100)
)
Plot.Torta(
  datos     = Blood     ,
  valores   = Prevalence,
  categoria = Group     ,
  colores   = c("#FF553D", "#A5FF67", "#40D2FF", "#FFDB5C"),
  label     = "No. of Prevalence"
)
Plot.Torta(
  datos     = Blood     ,
  valores   = Prevalence,
  categoria = Group     ,
  colores   = c("#FF553D", "#A5FF67", "#40D2FF", "#FFDB5C"),
  titulo    = "DISTRIBUTION OF BLOOD GROUPS",
  estatico  = TRUE,
  estilo    = list(gg.Tema = 6)
)
Plot.Torta(
  datos     = Blood     ,
  valores   = Prevalence,
  categoria = Group     ,
  colores   = c("#FF553D", "#A5FF67", "#40D2FF", "#FFDB5C"),
  titulo    = "DISTRIBUTION OF BLOOD GROUPS",
  estatico  = TRUE,
  estilo = list(
    gg.Tema  = 7, gg.Donut = TRUE, gg.Percent = FALSE,
    gg.Texto = list(
      subtitle = "Synthetic or fake data that resembles real-world data",
      caption  = "* Data Simulation",
      tag      = "\u00ae"
    )
  )
)
```

```
# ------------------------------------------------------------------------
col <- c("#F15A24", "#8CC63F")
Msj <- "Distribuci\u00f3n de estudiantes graduados en el primer periodo acad\u00e9mico del 2021."
Txt <- "DISTRIBUCI\u00d3N DE GRADUADOS POR MODALIDAD DE FORMACI\u00d3N"
Plot.Torta(
  datos     = ejConsolidadoGrad |> filter(YEAR==2021, SEMESTRE==1),
  categoria = "TIPO_NIVEL",
  colores   = col,
  titulo    = paste(Txt, "(Periodo 2021-1)"),
  label     = "N\u00famero de Graduados",
  libreria  = "highcharter",
  estilo    = list(
    LegendTitle = "\u00c9sta es una descripci\u00f3n para la leyenda:",
    hc.Tema = 7, hc.Credits = Msj
  )
)
# ------------------------------------------------------------------------
Msj <- "Distribuci\u00f3n hist\u00f3rica de estudiantes graduados (desde el 2009-I al 2021-I)."
Plot.Torta(
  datos     = ejConsolidadoGrad,
  categoria = "TIPO_NIVEL",
  colores   = col,
  titulo    = gsub("DOS POR", "DOS\nPOR", Txt),
  libreria  = "plotly",
  estilo    = list(
    ply.Legend = "inside", ply.Credits = list(
      x = 0.8, y = 1.1, text = paste0("<b>", Msj, "</b>")
    )
  )
)
```

`Plot.Treemap` *Cree un diagrama de árbol (*treemap*) dinámico y flexible con diversos paquetes*

Description

Esta función proporciona excelentes herramientas y opciones para la visualización de datos jerárquicos/estructurados como un conjunto de rectángulos anidados. Cada grupo está representado por un rectángulo, cuya área (*tamaño*) es proporcional a su frecuencia absoluta (*recuento*) y el color se usa para mostrar otra dimensión numérica. Usando la interactividad, es posible representar varias dimensiones/niveles: grupos, subgrupos, etc. Dicha gráfica se va a representar usando la librería `Highcharter`, `Plotly`, `d3treeR`, entre otras, las cuales usan internamente `JavaScript`.

Usage

```
Plot.Treemap(
  datos,
```

```
  variables,
  atributo,
  textFreq = "N",
  metodo = c("Classic", "Classic2", "Sunburst", "Sunburst2"),
  estadistico = c("Promedio", "Mediana", "Varianza", "SD", "CV", "Min", "Max"),
  colores,
  titulo = "",
  libreria = c("highcharter", "plotly"),
  estilo = NULL,
  estatico = FALSE
)
```

Arguments

`datos`	Un data frame, no un vector numérico.
`variables`	Una lista (*ya sea creada con la sintaxis* `base` *o* `tidy`) con las variables categóricas dentro del data frame ingresado en `datos` con las que se desea crear la jerarquía (*recuerde que esta se crea de izquierda a derecha, es decir la primera hace referencia al grupo, la segunda al subgrupo, etc.*).
`atributo`	Una variable numérica dentro del data frame ingresado en `datos`. Este es opcional y solo aplica en el caso de un nivel, es decir, cuando se ingresa únicamente una variable.
`textFreq`	Cadena de caracteres indicando el nombre que se le va a dar al recuento en cada uno de los grupos. Por defecto se emplea el rótulo "N".
`metodo`	Cadena de caracteres indicando el diseño con el cual se realizará el gráfico (*en el caso de ingresar dos niveles o más*). Los valores permitidos son `"Classic"` (*valor predeterminado*), `"Classic2"`, `"Sunburst"` y `"Sunburst2"`, así se usará las funciones d3treeR::d3tree(), d3treeR::d3tree2(), sunburst() y sund2b() respectivamente.
`estadistico`	Igual uso que en `Plot.Mapa()`
`colores`	Igual uso que en `Plot.Series()`, con algunos matices, cuando usamos `Highcharter` y nos encontramos en el caso de un nivel y especificamos un atributo es recomendable pasarle una escala de colores, pues con esto se construirá la barra horizontal. En el caso de usar el argumento `atributo` se puede ingresar el nombre de una paleta, por ejemplo "Set1".
`titulo`	Cadena de caracteres indicando el título principal del plot.
`libreria`	Igual uso que en `Plot.Torta()`, con algunos matices, pues en el caso de ingresar más de una variable categórica se omitirá dicho argumento, ya que `metodo` tomará su lugar.
`estilo`	Lista compuesta por varios parámetros, los cuales van a ser usados para graficar el treemap y cuyo objetivo es personalizar pequeños detalles de éste. • `hc.Tema` y `hc.Credits`: Igual uso que en `Plot.Series()` • `hc.borderRadius`: Un número entero positivo que indica el radio del borde de cada elemento. El valor por defecto es 0 (*rectángulos*). • `ply.Opacidad`: Igual uso que en `Plot.Radar()`

- `ply.Credits`: Igual uso que en `Plot.Series()`
- `sun.Explanation`: Cadena de caracteres indicando qué es lo que se desea ver en el centro del sunburst al pasar el mouse por los diferentes anillos (*niveles de la jerarquía*). Los valores permitidos son "All" (*valor predeterminado*), "Count" y "Percent". Solo aplica para cuando `metodo = "Sunburst"`.
- `sun.Color`: A diferencia del argumento `colores` acá puede pasar una paleta o vector de colores sin que se recorte (*más no se recicle*) éste a la longitud de categorías del nodo padre. Su uso reemplaza el funcionamiento del argumento `colores`. Solo aplica para cuando `metodo = "Sunburst"`.
- `sun.showLabels`: Si es `FALSE` (*valor predeterminado*) no se mostrará etiquetas en los cortes. Solo aplica para cuando `metodo = "Sunburst2"`.
- `sun.colorRoot`: Cadena de caracteres que indica el color del nodo raíz (*root*). Puede indicar el color con el nombre (`"red"`), código hexadecimal (`"#FF0000"`) o RGB (`rgb(1, 0, 0)`). El valor por defecto es "rojo". Solo aplica para cuando `metodo = "Sunburst2"`.
- `gg.fontsize.title`: Tamaño de la fuente del título. El valor por defecto es `14`. Para más detalles, consulte la función treemap().
- `gg.fontsize.labels`: Tamaño de la fuente de las etiquetas. Si ingresa un número especificará el tamaño para todos los niveles de agregación, por el contrario, si ingresa un vector podrá especificar el tamaño para cada nivel. El valor por defecto es `11`. Para más detalles, consulte la función treemap().
- `gg.fontcolor.labels`: Especifica los colores de la etiqueta. Ya sea una cadena de caracteres o un vector (*uno para cada nivel de agregación*). El valor por defecto es `NULL`. Para más detalles, consulte la función treemap().
- `gg.border.lwds`: Tamaño de las líneas de borde. Si ingresa un número especificará el grosor para todos los rectángulos, o un vector para especificar el grueso para cada nivel de agregación. Para más detalles, consulte la función treemap().
- `gg.border.col`: Color de los bordes dibujados alrededor de cada rectángulo, ya sea un valor único o un vector. El valor por defecto es `'#000000'`. Para más detalles, consulte la función treemap().
- `gg.lowerbound.cex.labels`: Número entre $[0, 1]$, 0 significa dibujar todas las etiquetas y 1 significa dibujar sólo las etiquetas si encajan (*considerando el* `fontsize.labels`). El valor por defecto es `0.4`. Para más detalles, consulte la función treemap().
- `gg.force.print.labels`: Si es `FALSE` (*valor predeterminado*) las etiquetas de datos no se ven obligadas a imprimirse si no encajan. Para más detalles consulte la función treemap().
- `gg.overlap.labels`: Número entre $[0, 1]$, que determina la tolerancia de superposición entre etiquetas. 0 significa que las etiquetas de los niveles inferiores no se imprimen si las etiquetas de los niveles superiores se superponen, 1 significa que las etiquetas siempre se imprimen. El valor por defecto es `0.5`. Para más detalles, consulte la función treemap().

`estatico` Si es `FALSE` (*valor predeterminado*) el gráfico a retornar será dinámico (*dependiendo de la librería seleccionada*), en caso contrario se retornará un gráfico estático construido con `ggplot2`.

Details

Si está trabajando en un `R Markdown` o un aplicativo `Shiny` no se puede usar de forma conjunta el `método = Classic` (o `Classic2`) y `método = Sunburst` (o `Sunburst2`), pues se trata de un problema interno, ya que usan versiones diferentes de `d3`, puede darle seguimiento al problema aquí. De igual forma, si utiliza la librería `sunburstR` en algunas ocasiones se le verán afectadas las tablas creadas con `DT`.

Value

Retorna el treemap (*objeto widget de HTML*) creado. La clase del objeto retornado será un "htmlwidget" y dependiendo de la librería usada pertenecerá adicionalmente a la clase "highchart", "plotly", " d3tree", "d3tree2", "sunburst" o "sund2b".

Lista de argumentos de estilo

Sabemos que puede ser abrumador el número de argumentos dentro del parámetro `estilo`, pero es necesario si queremos ofrecer al usuario la máxima personalización dentro de cada función usando cualquier librería. Por tal razón, a continuación, se detalla el listado completo de argumentos, usados al especificar la librería y en qué función están presentes (*marcado con una* × *si lo posee*).

Librería	**estilo$**	`Plot.Series()`	`Plot.Barras()`	`Plot.Apiladas()`	`Plot.Boxplot()`	`P`
—	*gg.Tema*	×	×	×	×	
l	*gg.Texto*	×	×	×	×	
l	*gg.Legend*	×		×	×	
l	*gg.Linea*	×				
l	*gg.Punto*	×				
l	*gg.Bar*		×	×		
l	*gg.VarWidth*				×	
l	*gg.OutShape*				×	
l	*gg.JitWidth*				×	
l	*gg.JitSize*				×	
l	*gg.Range*					
ggplot2	*gg.plty*					
l	*gg.plwd*					
l	*gg.cglwd*					
l	*gg.cglcol*					
l	*gg.fontsize.title*					
l	*gg.fontsize.labels*					
l	*gg.fontcolor.labels*					
l	*gg.border.lwds*					
l	*gg.border.col*					
l	*gg.lowerbound.cex.labels*					
l	*gg.force.print.labels*					
—	*gg.overlap.labels*					
»	*hc.Tema*	×	×	×	×	
l	*hc.Credits*	×	×	×	×	
highcharter	*hc.BoxInfo*	×				
l	*hc.Slider*	×				

»	*hc.borderRadius*				
•	*ply.Credits*	×	×	×	×
◦	*ply.Legend*		×		
◦	*ply.LegendPosition*	×		×	×
plotly	*ply.Interaction*	×			×
◦	*ply.Relleno*				
◦	*ply.Opacidad*				
•	*ply.LegendTitle*				
dygraphs	*dyg.LegendWidth*	×			
»	*dyg.Resaltar*	×			
—	*e.Tema*				
l	*e.Credits*				
echarts4r	*e.Forma*				
l	*e.LegType*				
—	*e.LegLoc*				

Examples

```
library(viridis)
Msj <- "Acompa\u00f1ado del Estad\u00edstico seleccionado para la Variable Edad."
Plot.Treemap(
  datos       = ejGraduados,
  variables   = SEDE_NOMBRE_MAT,
  atributo    = EDAD_MOD,
  textFreq    = "Tamaño de la Muestra",
  estadistico = "Max",
  colores     = inferno(10),
  titulo      = "TOTAL DE GRADUADOS POR SEDE DE LA UNIVERSIDAD NACIONAL",
  libreria    = "highcharter",
  estilo      = list(hc.Tema = 7, hc.borderRadius = 20, hc.Credits = Msj)
)
# ------------------------------------------------------------------------
Plot.Treemap(
  datos       = ejGraduados,
  variables   = FACULTAD,
  atributo    = EDAD_MOD,
  textFreq    = "n",
  estadistico = "CV",
  colores     = turbo(10, direction = -1),
  titulo      = "TOTAL DE GRADUADOS POR FACULTAD EN LA UNAL",
  libreria    = "plotly",
  estilo      = list(ply.Credits = list(x = 0.6, y = 1, text = Msj))
)


# ------------------------------------------------------------------------
# library(dplyr)
misColores <- c(
  "#29ABE2", # AZUL CLARO  | Amazonia
  "#8CC63F", # VERDE       | Bogota
```

```
  "#CC241D", # ROJO        | Caribe
  "#0071BC", # AZUL VIVO   | Manizales
  "#F15A24", # NARANJA     | Medellin
  "#FBB03B", # AMARILLO    | Orinoquia
  "#93278F", # MORADO      | Palmira
  "#8A381A"  # GRIS        | Tumaco
)
Plot.Treemap(
  datos     = ejGraduados,
  variables = vars(SEDE_NOMBRE_MAT, FACULTAD, PROGRAMA),
  metodo    = "Classic",
  colores   = misColores # "Set3"
)
Plot.Treemap(
  datos     = ejGraduados,
  variables = vars(SEDE_NOMBRE_MAT, FACULTAD, PROGRAMA),
  metodo    = "Classic2",
  colores   = "Set2"
)
Plot.Treemap(
  datos     = ejGraduados,
  variables = vars(SEDE_NOMBRE_MAT, FACULTAD, PROGRAMA),
  metodo    = "Sunburst",
  colores   = misColores,
  estilo    = list(sun.Explanation = "All")
)
Plot.Treemap(
  datos     = ejGraduados,
  variables = vars(SEDE_NOMBRE_MAT, FACULTAD, PROGRAMA),
  metodo    = "Sunburst",
  # colores   = misColores,
  estilo    = list(
    sun.Explanation = "All",
    sun.Color = list(range = c("#9E0142", "#D53E4F", "#F46D43", "#FDAE61",
                               "#FEE08B", "#FFFFBF", "#E6F598", "#ABDDA4",
                               "#66C2A5", "#3288BD", "#5E4FA2"
                               )
                     )
  )
)
Plot.Treemap(
  datos     = ejGraduados,
  variables = vars(SEDE_NOMBRE_MAT, FACULTAD, PROGRAMA),
  metodo    = "Sunburst2"
)
Plot.Treemap(
  datos     = ejGraduados,
  variables = vars(SEDE_NOMBRE_MAT, FACULTAD, PROGRAMA),
  metodo    = "Sunburst2",
  colores   = misColores,
  estilo    = list(sun.showLabels = TRUE, sun.colorRoot = "#EF0055")
)
```

```
# ------------------------------------------------------------------------
# Ejemplo usando el caso estático (treemap)
# library(dplyr)
Plot.Treemap(
  datos     = ejGraduados,
  variables = vars(SEDE_NOMBRE_MAT, FACULTAD),
  colores   = c("#FF3232", "#AFFF5E", "#FD6DB3", "#4CCAF2", "#FF9248", "#FBB03B"),
  titulo    = "TOTAL DE GRADUADOS \u00d7 SEDE",
  estatico  = TRUE,
  estilo    = list(
    gg.fontsize.title = 12, gg.fontsize.labels = c(15, 9),
    gg.fontcolor.labels = c("#FFFFFF", "#212020"),
    gg.border.lwds = c(4, 2), gg.border.col = c("#73095D", "#D60D4B"),
    gg.lowerbound.cex.labels = 0.3, gg.overlap.labels = 0.1
  )
)
```

---

read_example *Obtenga la ruta de los archivos de Excel para la función* Agregar()

Description

UnalR viene con algunos archivos de ejemplo en su directorio `inst/extdata`. Esta función facilita el acceso a ellos.

Usage

```
read_example(ruta = NULL)
```

Arguments

ruta	Nombre del archivo. Si es NULL, se enumerarán los archivos de ejemplo.

Value

Cadena de caracteres indicando la ruta absoluta (*no relativa a la carpeta en donde se ubique*) en donde se encuentra el archivo especificado dentro del paquete.

Examples

```
read_example("TestConsolidado1.xlsx")
```

`StaticPlot` *Guarde y presente un widget HTML renderizado como una imagen estática*

Description

Esta función permite representar un widget HTML como un objeto ráster (*imagen de mapa de bits*), útil para reproducir gráficos interactivos en archivos estáticos como un `.pdf` generado por `R Markdown`. Esta función usa internamente los paquetes `webshot`, `htmlwidgets`, `png` y `grid` para poder llevar a cabo su propósito.

Usage

```
StaticPlot(widgetHTML, height = 500, primeraVez = FALSE, ...)
```

Arguments

`widgetHTML`	Widget `HTML` a ser mostrado de forma estática.
`height`	Altura de la imagen estática a retornar.
`primeraVez`	Si es `FALSE` (*valor predeterminado*) no se instalará `PhantomJS`. Éste es necesario instalarlo una única vez, por lo cual si es la primera vez que corre la función deberá indicar el argumento con el valor `TRUE`, después de esto omita este argumento y deje su valor por defecto.
`...`	Otros parámetros concernientes a la función webshot(), sin considerar los ya usados dentro de la función (`url`, `file`, `delay`, `zoom` y `vheight`).

Details

No es necesario especificar el número del factor de zoom. El factor de zoom por defecto es 5, el cual dará como resultado cinco veces más de píxeles vertical y horizontalmente. Este valor fue seleccionado debido a que es el óptimo, un valor mayor ocasiona un tiempo de ejecución excesivamente alto y poca ganancia en cuanto a calidad, con un valor menor se tiene una pérdida considerable de calidad.

Si se especifican tanto el ancho como el alto, es probable que la imagen se distorsione. Por lo tanto, el único argumento variable será la altura de la imagen, dejando el ancho como un argumento adaptativo dependiendo del ancho disponible.

El tiempo de espera antes de tomar una captura de pantalla, en segundos, es de $1s$. Este valor es debido a que se necesita un retraso mayor para que los gráficos generados por `Highcharter` se muestren correctamente.

Value

Una imagen estática.

Examples

```
misColores <- c(
  "#29ABE2", # AZUL CLARO  | Amazonia
  "#8CC63F", # VERDE       | Bogota
  "#CC241D", # ROJO        | Caribe
  "#0071BC", # AZUL VIVO   | Manizales
  "#F15A24", # NARANJA     | Medellin
  "#FBB03B", # AMARILLO    | Orinoquia
  "#93278F", # MORADO      | Palmira
  "#8A381A"  # GRIS        | Tumaco
)
figure <- Plot.Series(
  datos     = ejConsolidadoGrad,
  categoria = "SEDE_NOMBRE_ADM",
  colores   = misColores,
  libreria  = "highcharter"
)
StaticPlot(figure, primeraVez = TRUE)

figure2 <- Plot.Torta(
  datos     = ejConsolidadoGrad,
  categoria = "SEXO",
  ano       = 2021,
  periodo   = 1,
  colores   = c("#116BEE", "#E62272"),
  titulo    = "DISTRIBUCI\u00d3N DE GRADUADOS POR SEXO",
  libreria  = "highcharter"
)
StaticPlot(figure2)
```

Tabla *Cree fácilmente un widget para visualización de tablas HTML usando el paquete* DT

Description

Esta función está diseñada para facilitar/simplificar la creación/producción de tablas para informes, presentaciones y publicaciones, produciendo un widget HTML para visualizar un data frame utilizando el paquete DT. La forma en que esta función maneja las cosas por usted significa que a menudo no tiene que preocuparse por los pequeños detalles para obtener un resultado impresionante y listo para usar.

Usage

```
Tabla(
  datos,
  df,
  rows,
```

```
  pivotCat,
  pivotVar,
  columnNames,
  filtros = FALSE,
  colFilters,
 estadistico = c("Suma", "Promedio", "Mediana", "Varianza", "SD", "CV", "Min", "Max"),
  encabezado = "Encabezados de los Niveles de la Categoría",
  leyenda = "",
  tituloPdf = NULL,
  mensajePdf = "",
  ajustarNiveles = TRUE,
  scrollX = TRUE,
  fillContainer = NULL,
  colorHead = "#FFFFFF",
  estilo,
  estatico = FALSE
)
```

Arguments

datos	Un data frame.
df	Argument deprecated, use datos instead.
rows	Una variable categórica dentro del data frame ingresado en datos.
pivotCat	Variable categórica que contiene los niveles/factores que desea pivotear como columnas. Si omite este parámetro se da por hecho que no desea pivotear nada sino graficar tal cual su data frame
pivotVar	Variable numérica que contiene los valores que desea colocar en cada celda al realizar el pivotaje.
columnNames	Vector de caracteres que especifica los nombres de las columnas de la tabla a retornar. Si no se introduce algún valor se tomará el mismo nombre de las columnas presentes en datos.
filtros	Si es FALSE (*valor predeterminado*) no se habilitará/aplicará los filtros por columna. Establézcalo en TRUE si desea generar filtros de columna automáticamente.
colFilters	Vector numérico que especifica las columnas a las cuales les desea agregar la opción de poder filtrar. Si no se introduce algún valor todas las columnas tendrán habilitada la opción de poder filtrar.
estadistico	X.
encabezado	Cadena de caracteres que describe los distintos niveles de la variable categoria.
leyenda	Cadena de caracteres que describe información adicional de la tabla, ésta se sitúa en la parte inferior de la tabla de manera centrada, dicho texto se visualizará en todas las opciones de descarga. Su valor por defecto es NULL.
tituloPdf	Cadena de caracteres que proporciona un título a la tabla al momento de generar el .pdf como al hacer clic al botón de imprimir. Su valor por defecto es el introducido en el argumento encabezado.

`mensajePdf` Cadena de caracteres que proporciona un mensaje situado entre el título y la tabla. Se visualizará tanto al generar el `.pdf` como al hacer clic al botón de imprimir.

`ajustarNiveles` Si es `TRUE` (*valor predeterminado*) se buscará optimizar el espacio entre las columnas, colocando todos los nombres de las columnas de forma horizontal y eliminando al máximo el espacio entre éstas.

`scrollX` Si es `TRUE` (*valor predeterminado*) se habilitará la propiedad Scroller para el eje X. Tenga presente que cuando su df contiene muchas columnas es de utilidad (*pues no permite que se salga la tabla por ancho*), sin embargo, asegúrese de desactivarlo cuando presente pocas columnas, pues se verá un desplazamiento de los encabezados debido a un conflicto interno.

`fillContainer` Valor booleano para indicar si desea que la tabla rellene automáticamente el elemento que lo contiene.

`colorHead` Cadena de caracteres que indica el color de fondo de la cabecera de la tabla. Puede indicar el color con el nombre (`"red"`), código hexadecimal (`"#FF0000"`) o RGB (`rgb(1, 0, 0)`). El valor por defecto es "blanco" (`"#FFFFFF"`).

`estilo` Una lista compuesta por listas las cuales en su interior contiene argumentos válidos de la función formatStyle(), esto con la finalidad de que pueda aplicar estilos CSS a la tabla, tales como color de la fuente, color de fondo, tamaño de fuente, etc. Puede encontrar mayor información de los argumentos disponibles aquí.

- `Tema`: Modifica el tema con el cual se creará la tabla Los posibles valores son un número entero entre $[1, 14]$ el cual hace referencia a diferentes temas disponibles para `gt/gtExtras` (los primeros 6 se obtienen con `opt_stylize(style = i, color = "gray")`, `gt_theme_538`, `gt_theme_dark`, `gt_theme_dot_matrix`, `gt_theme_espn`, `gt_theme_excel`, `gt_theme_guardian`, `gt_theme_nytimes` y `gt_theme_pff` respectivamente).
- `Titulo`: Cadena de caracteres indicando el título principal de la tabla.
- `Padding`: Vector numérico de longitud 2, que tiene como primera coordenada el padding vertical, es decir si aumenta o disminuye el relleno vertical. Y como segunda coordenada el padding horizontal.
- `Color`: Lista compuesta de listas, en la cual cada una de ellas detalla, qué columna se va a afectar y con qué colores de relleno, es decir, `list(columns = Columna , backgroundColor = Colores)`.

`estatico` Si es `FALSE` (*valor predeterminado*) la tabla a retornar será dinámica (*usando la librería* `DT`), en caso contrario se retornará una tabla estática construida con `gt` y `gtExtras`.

Details

Esta función se basa enteramente del paquete `DT`, el cual proporciona una interfaz para `R` a la biblioteca `DataTables` de `JavaScript`. Los data frames de `R` se pueden mostrar como tablas en páginas HTML, proporcionando opciones de filtrado, paginación, clasificación y muchas otras características en las tablas.

Esta función se basa enteramente del paquete `DT`, el cual proporciona una interfaz para `R` a la biblioteca `DataTables` de `JavaScript`. Los data frames de `R` se pueden mostrar como tablas en páginas

HTML, proporcionando opciones de filtrado, paginación, clasificación y muchas otras características en las tablas.

Al establecer `filtros = FALSE` no elimina ni modifica el filtro global (*cuadro de búsqueda en la parte superior derecha*).

Para el argumento `colFilters` recuerde que la numeración inicia en 0, es decir, la primera columna tiene asociado el índice 0, la segunda el 1, y así sucesivamente.

Value

Retorna la tabla creada mediante `DT` la cual pertenece a la clase "datatables" y "htmlwidget".

Examples

```
# library(DT); library(dplyr); library(tidyr)
# Example of R Combinations with Dot (".") and Pipe (%>%) Operator
# UnalR::Agregar(
#   datos      = UnalData::Graduados,
#   formula    = SEDE_NOMBRE_ADM ~ YEAR + SEMESTRE,
#   frecuencia = list("Year" = 2009:2022, "Period" = 1:2)
#   ) |>
#   select(-Variable) |>
#   rename(Year = YEAR, Semester = SEMESTRE, Cat = Clase) %>%
#   Tabla(
#     ., rows = vars(Year, Semester), pivotCat = Cat, pivotVar = Total
#   )
Tabla(
  datos       = ejConsolidadoGrad |> dplyr::filter(Variable == "SEDE_NOMBRE_ADM") |>
    dplyr::select(-Variable),
  rows        = vars(YEAR, SEMESTRE),
  pivotCat    = Clase,
  pivotVar    = Total,
  columnNames = c("Año", "Semestre", "Total"),
  estadistico = "Suma",
  encabezado  = "TOTAL DE ESTUDIANTES \u00d7 SEDE DE GRADUACI\u00d3N",
 leyenda    = "Distribuci\u00f3n de estudiantes graduados (desde el 2009-I al 2021-I) por sede.",
  tituloPdf   = "ESTUDIANTES GRADUADOS POR SEDE",
  colorHead   = "#8CC63F",
  estilo      = list(
    list(
      columns = "YEAR", target = "cell", fontWeight = "normal",
      backgroundColor = styleEqual(
        unique(ejConsolidadoGrad$YEAR), rainbow(13, alpha = 0.5, rev = TRUE)
      )
    ),
    list(
      columns = "SEMESTRE", target = "cell", fontWeight = "bold",
      color = styleEqual(unique(ejConsolidadoGrad$SEMESTRE), c("#EB0095", "#9D45FD"))
    )
  )
)
# ------------------------------------------------------------------------
VariosYears <- ejConsolidadoSaberPro2019         |>
```

```
  mutate(YEAR = replace(YEAR, YEAR==2019, 2020)) |>
  bind_rows(ejConsolidadoSaberPro2019)           |>
  filter(Variable == "sede") |> select(-Variable, -desv)

Msj <- "\u00c9sta es una descripci\u00f3n de la tabla diferente al valor por default."
Tabla(
  datos       = VariosYears,
  rows        = vars(YEAR, Clase, n),
  pivotCat    = Componente,
  pivotVar    = Total,
  columnNames = c("Año", "Sede", "n", "M\u00e1ximo"),
  estadistico = "Max",
  encabezado  = "PUNTAJES \u00d7 SEDE",
  leyenda     = Msj,
  colorHead   = "#F9CA00",
  estilo      = list(
    list(
      columns = "YEAR", target = "cell", fontWeight = "normal",
      backgroundColor = styleEqual(unique(VariosYears$YEAR), c("#AEF133", "#19EE9F"))
    ),
    list(
      columns = "Clase", target = "cell", fontWeight = "bold",
    color = styleEqual(unique(VariosYears$Clase), c("#42C501", "#7E10DE", "#FF6700", "#0096F2"))
    )
  )
)
# ------------------------------------------------------------------------------
Tabla(datos = datasets::mtcars)

df <- ejGraduados |>
  filter(TIPO_NIVEL == "Pregrado") |>
  group_by(YEAR, SEMESTRE, DEP_NAC, CIU_NAC, SEXO, CAT_EDAD, ESTRATO, PROGRAMA) |>
  summarise(Total = n(), .groups = "drop") |>
  mutate(across(where(is.character), \(x) replace_na(x, replace = "SIN INFO")))

Nombres <- c("<em>A\u00f1o</em>", "Semestre", "Departamento",
             "Municipio", "Sexo", "Edad", "Estrato", "Carrera", "Total"
             )
Titulo  <- paste(
  "<b>HIST\u00d3RICO DEL TOTAL DE GRADUADOS DE PREGRADO ",
  "DEPENDIENDO DE LAS VARIABLES SELECCIONADAS</b>"
)
Tabla(
  datos          = df,
  columnNames    = Nombres,
  filtros        = TRUE,
  colFilters     = 0:3,
  encabezado     = Titulo,
  leyenda        = "N\u00famero de graduados de pregrado por lugar de procedencia.",
  tituloPdf      = "Este es un t\u00edtulo provisional para el PDF",
  mensajePdf     = "Este es un mensaje provisional para el PDF",
  ajustarNiveles = TRUE,
  colorHead      = "#4CFF49",
```

```
  estilo         = list(
    list(
      columns = "YEAR", target = "cell", fontWeight = "bold",
      backgroundColor = styleEqual(unique(df$YEAR), c("#FF6400", "#01CDFE", "#FF0532"))
    ),
    list(
      columns = "SEMESTRE", target = "cell", fontWeight = "bold",
      color = styleEqual(unique(df$SEMESTRE), c("#3D3397", "#AE0421"))
    ),
    list(columns = "DEP_NAC", color = "#FFFFFF", backgroundColor = "#4D1B7B"),
    list(columns = "CIU_NAC", color = "#FFFFFF", backgroundColor = "#F59E11")
  )
)


# library(tibble)
# ------------------------------------------------------------------------
# Ejemplo Usando Directamente un Consolidado de Microdatos (Compose functions with Agregar)
set.seed(2023)
AcademyAwards <- tibble(
  year     = sample(1939:1945, 100, TRUE),
  season   = sample(1:2, 100, TRUE),
  category = sample(
    c("Best Picture", "Best Director", "Best Actor", "Best Actress", "Best Sound"),
    100, TRUE
  ),
 location = sample(c("Roosevelt Hotel", "Dolby Theatre", "NBC Century Theatre"), 100, TRUE)
)
Agregar(
  datos      = AcademyAwards,
  formula    = category + location ~ year + season,
  frecuencia = list("Year" = 1939:1945, "Period" = 1:2)
) %>%
  Tabla(., pivotCat = "location", columnNames = c("Year", "Season"),
        encabezado = "LOCATION OF CEREMONIES", scrollX = FALSE
  )


# library(gt); library(gtExtras)
# ------------------------------------------------------------------------
# Ejemplo usando el caso estático (gt)
tableGT <- Tabla(
  datos       = UnalR::ejConsolidadoGrad |> filter(Variable == "SEDE_NOMBRE_ADM"),
  rows        = vars(YEAR),
  pivotCat    = Clase,
  pivotVar    = Total,
  encabezado  = "TOTAL DE ESTUDIANTES \u00d7 SEDE DE GRADUACI\u00d3N",
  leyenda     = paste(
    "Distribuci\u00f3n de estudiantes graduados ",
    "(desde el 2009-I al 2021-I) por sede."
  ),
  colorHead   = "#8CC63F",
  estatico    = TRUE,
```

```
  estilo      = list(
    Tema = 11, Padding = c(0, 0.5), Titulo = "Summary Table:",
    Color = list(
     list(columns = "YEAR"     , backgroundColor = rainbow(12, alpha = 0.5, rev = TRUE)),
      list(columns = "Palmira"  , backgroundColor = "ggsci::red_material"),
      list(columns = "Manizales", backgroundColor = "viridis")
    )
  )
)
# ------------------------------------------------------------------------
# Ejemplo usando algunos parámetros adicionales de personalización de gt/gtExtras
#   (para ver el alcance que puede tener)
tableGT <-
  Tabla(
    datos       = UnalR::ejConsolidadoGrad |> filter(Variable == "SEDE_NOMBRE_ADM"),
    rows        = vars(YEAR, SEMESTRE),
    pivotCat    = Clase,
    pivotVar    = Total,
    estadistico = "Suma",
    encabezado  = "TOTAL DE ESTUDIANTES \u00d7 SEDE DE GRADUACI\u00d3N",
    leyenda     = paste(
      "Distribuci\u00f3n de estudiantes graduados ",
      "(desde el 2009-I al 2021-I) por sede."
    ),
    colorHead   = "#AA0000",
    estatico    = TRUE,
    estilo      = list(
      Tema = 14, Padding = c(0, 0.5), Titulo = "SUMMARY TABLE",
      Color = list(
       list(columns = "YEAR"     , backgroundColor = rainbow(12, alpha = 0.5, rev = TRUE)),
        list(columns = "SEMESTRE" , backgroundColor = c("#EB0095", "#9D45FD"))
      )
    )
  )

Win <- "<span style=\"color:green\">&#128170;</span>"
Loss <- "<span style=\"color:red\">&#128165;</span>"
tableGT |>
  # ____________________ INSERTANDO UN PIE DE PÁGINA ADICIONAL _____________________
 tab_source_note(source_note = "Source: Dirección Nacional de Planeación y Estadística (DNPE).") |>
  # _____________________ CREANDO UN GRUPO/COLECCIÓN DE FILAS ______________________
  tab_row_group(label = "< 2010"       , rows = 1:2)   |>
  tab_row_group(label = "[2010 - 2019]", rows = 3:22)  |>
  tab_row_group(label = ">= 2020"      , rows = 23:25) |>
  # __________ MODIFICANDO LA ALINEACIÓN DE CADA UNA DE LAS COLUMNAS ___________
  cols_align(align = "center", columns = Amazonía:Tumaco)  |>
  cols_align(align = "left"  , columns = where(is.factor)) |>
  # ____________________ COLOREANDO LAS CELDAS DE UNA COLUMNA _____________________
  data_color(
    columns = Statistic,
    method  = "bin",
    bins    = c(0, 3000, 4500, 10000),
    palette = c("#F44336", "#34AEC6", "#76CF44")
```

```
  ) |>
  # ____________ MODIFICANDO ASPECTOS GENERALES/GLOBALES DE LA TABLA _____________
  tab_options(
    heading.align = "right", heading.background.color = "#490948",
    table.font.size = px(12), heading.title.font.size = px(16)
  ) |>
  # _____________ CAMBIANDO EL FORMATO DE LOS VALORES NUMÉRICOS _____________
  fmt_currency(columns = c(Orinoquía:Palmira), currency = "USD") |>
  fmt_percent(columns = Tumaco, decimals = 1) |>
  # _ AÑADIENDO ALGUNOS DE LOS ESTILOS PERSONALIZADOS DISPONIBLES A LAS CELDAS _
  tab_style(
    style = cell_fill(color = "#C90076"), locations = cells_column_spanners()
  ) |>
  tab_style(
    style = list(cell_text(color = "#A5FD45", style = "italic")),
    locations = cells_body(columns = SEMESTRE, rows = SEMESTRE == "1")
  ) |>
  tab_style_body(
    style = cell_text(color = "#0CEAC0", weight = "bold"),
    columns = Amazonía,
    fn = function(x) between(x, 5, 20)
  ) |>
  text_transform(
    fn = function(x) paste(x, Win),
    locations = cells_body(columns = "Caribe", rows = Bogotá > 3*Medellín)
  ) |>
  text_transform(
    fn = function(x) paste(x, Loss),
    locations = cells_body(columns = "Caribe", rows = Bogotá <  3*Medellín)
  ) |>
  # _____________________________ MODIFICANDO LA FUENTE _____________________________
  opt_table_font(
    # font = google_font(name = "Merriweather"),
    stack = "rounded-sans",
    weight = "bolder"
  ) |>
  # ____________ OPCIONES ADICIONALES CON LIBRERÍAS COMPLEMENTARIAS ____________
  gtExtras::gt_highlight_rows(rows = 18, fill = "#FEEF05", font_weight = "bold") |>
 gtExtras::gt_add_divider(Bogotá, color = "#F94D00", style = "dotted", weight = px(4)) |>
 gtExtras::gt_plt_bar_pct(Medellín, fill = "#2A8A9C", background = "#0DC8A7", scaled = FALSE)

# Use el siguiente comando si desea guardar la tabla estática obtenida:
# gtsave(tableGT, "TablaResumen.html") # O .tex, docx
```

`Tabla.General` *Cree fácilmente un widget para visualización de tablas HTML usando el paquete* `DT`

Description

[Deprecated]

Esta función simplifica la producción de tablas para presentaciones visualmente atractivas, pues está diseñada para facilitar la creación de tablas para informes y publicaciones produciendo un widget HTML para visualizar un data frame utilizando el paquete DT.

Usage

```
Tabla.General(
  datos,
  colNames,
  filtros = FALSE,
  colFilters,
  encabezado = "",
  leyenda = "",
  tituloPdf = NULL,
  mensajePdf = "",
  ajustarNiveles = TRUE,
  scrollX = TRUE,
  colorHead = "#FFFFFF",
  estilo
)
```

Arguments

datos	Un data frame o una matriz.
colNames	Vector de caracteres que especifica los nombres de las columnas de la tabla a retornar. Si no se introduce algún valor se tomará el mismo nombre de las columnas presentes en datos.
filtros	Si es FALSE (*valor predeterminado*) no se habilitará/aplicará los filtros por columna. Establézcalo en TRUE si desea generar filtros de columna automáticamente.
colFilters	Vector numérico que especifica las columnas a las cuales les desea agregar la opción de poder filtrar. Si no se introduce algún valor todas las columnas tendrán habilitada la opción de poder filtrar.
encabezado	Igual uso que en Tabla()
leyenda	Igual uso que en Tabla()
tituloPdf	Igual uso que en Tabla()
mensajePdf	Igual uso que en Tabla()
ajustarNiveles	Igual uso que en Tabla()
scrollX	Igual uso que en Tabla()
colorHead	Igual uso que en Tabla()
estilo	Una lista compuesta por listas las cuales en su interior contiene argumentos válidos de la función formatStyle(), esto con la finalidad de que pueda aplicar estilos CSS a la tabla, tales como color de la fuente, color de fondo, tamaño de fuente, etc. Puede encontrar mayor información de los argumentos disponibles aquí.

Details

Esta función se basa enteramente del paquete DT, el cual proporciona una interfaz para R a la biblioteca DataTables de JavaScript. Los data frames de R se pueden mostrar como tablas en páginas HTML, proporcionando opciones de filtrado, paginación, clasificación y muchas otras características en las tablas.

Al establecer filtros = FALSE no elimina ni modifica el filtro global (*cuadro de búsqueda en la parte superior derecha*).

Para el argumento colFilters recuerde que la numeración inicia en 0, es decir, la primera columna tiene asociado el índice 0, la segunda el 1, y así sucesivamente.

Value

Retorna la tabla creada mediante DT la cual pertenece a la clase "datatables" y "htmlwidget".

Examples

```
Tabla.General(datos = datasets::mtcars)


# library("dplyr"); library("tidyr"); library("DT")
df <- ejGraduados |>
  filter(TIPO_NIVEL == "Pregrado") |>
  group_by(YEAR, SEMESTRE, DEP_NAC, CIU_NAC, SEXO, CAT_EDAD, ESTRATO, PROGRAMA) |>
  summarise(Total = n(), .groups = "drop") |>
  mutate(across(where(is.character), \(x) replace_na(x, replace = "SIN INFO")))

Nombres <- c("<em>A\u00f1o</em>", "Semestre", "Departamento",
             "Municipio", "Sexo", "Edad", "Estrato", "Carrera", "Total"
             )
Titulo  <- paste(
  "<b>HIST\u00d3RICO DEL TOTAL DE GRADUADOS DE PREGRADO ",
  "DEPENDIENDO DE LAS VARIABLES SELECCIONADAS</b>"
)
Tabla.General(
  datos          = df,
  colNames       = Nombres,
  filtros        = TRUE,
  colFilters     = 0:3,
  encabezado     = Titulo,
  leyenda        = "N\u00famero de graduados de pregrado por lugar de procedencia.",
  tituloPdf      = "Este es un t\u00edtulo provisional para el PDF",
  mensajePdf     = "Este es un mensaje provisional para el PDF",
  ajustarNiveles = TRUE,
  colorHead      = "#4CFF49",
  estilo         = list(
    list(
      columns = "YEAR", target = "cell", fontWeight = "bold",
      backgroundColor = styleEqual(unique(df$YEAR), c("#FF6400", "#01CDFE", "#FF0532"))
    ),
    list(
      columns = "SEMESTRE", target = "cell", fontWeight = "bold",
```

```
      color = styleEqual(unique(df$SEMESTRE), c("#3D3397", "#AE0421"))
    ),
    list(columns = "DEP_NAC", color = "#FFFFFF", backgroundColor = "#4D1B7B"),
    list(columns = "CIU_NAC", color = "#FFFFFF", backgroundColor = "#F59E11")
  )
)
```

`Tabla.SaberPro` *Cree fácilmente un widget para visualizar los resultados de la prueba Saber Pro en tablas HTML usando el paquete* `DT`

Description

[Deprecated]

Esta función está diseñada para facilitar la creación de tablas para informes y publicaciones produciendo un widget HTML para visualizar un data frame utilizando el paquete `DT`. La forma en que esta función maneja las cosas por usted significa que a menudo no tiene que preocuparse por los pequeños detalles para obtener un resultado impresionante y listo para usar.

Usage

```
Tabla.SaberPro(
  datos,
  variable,
  encabezado = "Encabezados de los Niveles de la Categoría",
  leyenda,
  tituloPdf = NULL,
  mensajePdf = "",
  ajustarNiveles = TRUE,
  scrollX = TRUE,
  colorHead = "#FFFFFF",
  colorear = FALSE,
  estilo
)
```

Arguments

`datos`	Igual uso que en `Tabla()`
`variable`	Análogo al argumento `categoria` de la función `Tabla()`
`encabezado`	Igual uso que en `Tabla()`
`leyenda`	Igual uso que en `Tabla()` con la excepción de que, si no se introduce ningún valor, el valor por defecto será una nota explicando a qué hace referencia los valores y columnas de la tabla.
`tituloPdf`	Igual uso que en `Tabla()`
`mensajePdf`	Igual uso que en `Tabla()`

ajustarNiveles	Igual uso que en Tabla()
scrollX	Igual uso que en Tabla()
colorHead	Igual uso que en Tabla()
colorear	Igual uso que en Tabla()
estilo	Una lista compuesta por dos parámetros: • PaletaYear: Vector de caracteres que especifica los colores de fondo para los años. • PaletaCategoria: Vector de caracteres que especifica los colores de fuente para las distintas categorías de la variable.

Details

Esta función se basa enteramente del paquete DT, el cual proporciona una interfaz para R a la biblioteca DataTables de JavaScript. Los data frames de R se pueden mostrar como tablas en páginas HTML, proporcionando opciones de filtrado, paginación, clasificación y muchas otras características en las tablas.

Value

Retorna la tabla creada mediante DT la cual pertenece a la clase "datatables" y "htmlwidget".

Examples

```
if (require("dplyr")) {
  VariosYears <- ejConsolidadoSaberPro2019 |>
    mutate(YEAR = replace(YEAR, YEAR==2019, 2020)) |>
    bind_rows(ejConsolidadoSaberPro2019)
}
Msj <- "\u00c9sta es una descripci\u00f3n de la tabla diferente al valor por default."
Tabla.SaberPro(
  datos      = VariosYears,
  variable   = "SEXO",
  encabezado = "PUNTAJES POR SEXO",
  leyenda    = Msj,
  colorHead  = "#FF5B5B",
  estilo     = list(
    PaletaYear = c("#F9CA00", "#F68118"),
    PaletaCategoria = c("#2458C5", "#F0006D", "#42C501")
  )
)
Tabla.SaberPro(
  datos      = VariosYears,
  variable   = "SEDE",
  encabezado = "PUNTAJES POR SEDE",
  leyenda    = Msj,
  colorHead  = "#F9CA00",
  estilo     = list(
    PaletaYear = c("#AEF133", "#19EE9F"),
    PaletaCategoria = c("#DD1C1A", "#FF6700", "#7E10DE","#0096F2", "#42C501")
  )
```

)

Index