Package ‘BibPlots’

May 6, 2026

Title Plot Functions for Use in Bibliometrics

Version 0.0.8

Description Currently, the package provides several functions for plotting and analyzing bibliometric data (JIF, Journal Impact Factor, and paper percentile values), beamplots with citations and percentiles, and three plot functions to visualize the result of a reference publication year spectroscopy (RPYS) analysis performed in the free software 'CRExplorer' (see <`http://crexplorer.net`>). Further extension to more plot variants is planned.

Depends R (>= 3.1.2)

License EUPL

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Robin Haunschild [aut, cre]

Maintainer Robin Haunschild <`R.Haunschild@fkf.mpg.de`>

Repository CRAN

Date/Publication 2021-10-12 15:20:05 UTC

Contents

`beamplot` *Create a beamplot using raw citations from a WoS download*

Description

Create a beamplot using raw citations from a WoS download. Use the format "Other File Format –> Tab-delimited (Win, UTF-8)" and provide the downloaded file name. a simple weighting of citation counts is also available for comparison of older with newer publications.

Usage

```
beamplot(wos_file, do_weight = FALSE, ...)
```

Arguments

`wos_file`	is the file name of the downloaded WoS export in the format Tab-delimited (Win, UTF-8).
`do_weight`	is a boolean to spcify if citation counts should be weighted with their age. The older the publication, the smaller the weight. The weight depends on on the difference between the year until that citations are counted (i.e., the current calendar year in the case of WoS downloads) and the publication year. A weighting factor of 1 is used for a difference of 0, 1/2 for a difference of 1, ..., and 1/11 for differences of ten or more.
`...`	further parameters passed to stripchart.

Details

beamplot(wos_file="WoS_savedrecs.txt", do_weight=boolean) Only the argument wos_file is mandatory. The argument do_weight is optional and FALSE by default.

Literature:

- Haunschild, R., Bornmann, L., & Adams, J. (2019). R package for producing beamplots as a preferred alternative to the h index when assessing single researchers (based on downloads from Web of Science), Scientometrics, DOI 10.1007/s11192-019-03147-3, preprint: https://arxiv.org/abs/1905.09095

Examples

```
## Not run: beamplot("WoS_savedrecs.txt")
```

`beamplot_scopus` *Create a beamplot using raw citations from a Scopus download*

Description

Create a beamplot using raw citations from a Scopus download. Use the CSV/Excel format and provide the downloaded file name. A simple weighting of citation counts is also available for comparison of older with newer publications.

Usage

```
beamplot_scopus(scopus_file, do_weight = FALSE, ...)
```

Arguments

`scopus_file`	is the file name of the downloaded Scopus export in the format CSV/Excel.
`do_weight`	is a boolean to spcify if citation counts should be weighted with their age. The older the publication, the smaller the weight. The weight depends on on the difference between the year until that citations are counted (i.e., the current calendar year in the case of Scopus downloads) and the publication year. A weighting factor of 1 is used for a difference of 0, 1/2 for a difference of 1, ..., and 1/11 for differences of ten or more.
`...`	further parameters passed to stripchart.

Details

beamplot_scopus(scopus_file="Scopus.csv", do_weight=boolean) Only the argument scopus_file is mandatory. The argument do_weight is optional and FALSE by default.

Literature:

- Haunschild, R., Bornmann, L., & Adams, J. (2019). R package for producing beamplots as a preferred alternative to the h index when assessing single researchers (based on downloads from Web of Science), Scientometrics, DOI 10.1007/s11192-019-03147-3, preprint: https://arxiv.org/abs/1905.09095

Examples

```
## Not run: beamplot_scopus("Scopus.csv")
```

`DAMBibPlot` *Create a difference against mean plot using journal and paper percentile values*

Description

Provide journal and paper percentile values in a data frame, e.g. df, and the function call DAMBibPlot(df) creates the difference against mean plot. DAMBibPlot takes some optional arguments to modify its behaviour, see arguments and details.

Usage

```
DAMBibPlot(
  df,
  off_set = 0,
  print_stats = TRUE,
  do_plot = TRUE,
  digits = 1,
  ...
)
```

Arguments

`df`	data frame with journal and paper percentiles
`off_set`	determines the location of additional plotted information (number of points in each quadrant), values between 0 and 40 might be useful (optional parameter). The default value is 0.
`print_stats`	boolean variable (optional parameter) which determines if the additional statistical values are printed to the R console (T: yes print, F: no do not print). The default value is T.
`do_plot`	boolean variable (optional parameter) which determines if the difference against mean plot is actually produced (T: yes plot, F: no do not plot). The default value is T.
`digits`	integer value to determine the number of desired digits after the decimal point for statistical values (optional parameter). The default value is 1.
`...`	additional arguments to pass to the plot function

Details

DAMBibPlot(df=data_frame, off_set=numeric_value, print_stats=boolean, do_plot=boolean) Only the argument df is necessary. All other aruments are optional.

Literature:

- Bland, J. M., & Altman, D. G. (1986). Statistical Methods for Assessing Agreement between Two Methods of Clinical Measurement. Lancet, 1(8476), 307-310, https://www.ncbi.nlm.nih.gov/pubmed/2868172

Cleveland, W. S. (1985). The elements of graphing data. Monterey, CA: Wadsworth Advanced Books and Software.

- Bornmann, L., & Haunschild, R. (2017). Plots for visualizing paper impact and journal impact of single researchers in a single graph, DOI: 10.1007/s11192-018-2658-1, preprint: https://arxiv.org/abs/1707.04050

An example data frame is provided as `example_researcher` in the package. It can be used to create a difference against mean plot using default values.

Examples

```
data(example_researcher)

DAMBibPlot(example_researcher)
```

`example_researcher` *Example data set from publication for scatter plot and difference against mean plot*

Description

Contains the data set (`example_researcher`).

`inv_perc_beamplot` *Create a beamplot using inverted percentile values*

Description

Create a beamplot using inverted percentile values.

Usage

```
inv_perc_beamplot(rd, au_name = "Example Researcher", ...)
```

Arguments

`rd`	is a dataframe with two columns: (i) publication year and (ii) inverted percentile value with one row per paper/dataset.
`au_name`	is the name of the researcher this beamplot belongs to.
`...`	further parameters passed to stripchart.

Details

inv_perc_beamplot(rd, au_name='Name of researcher') Only the rd is argument mandatory. It has to be a dataframe with two columns: (i) publication year and (ii) inverted percentile value with one row per paper/dataset.

Literature:

- Haunschild, R., Bornmann, L., & Adams, J. (2019). R package for producing beamplots as a preferred alternative to the h index when assessing single researchers (based on downloads from Web of Science), Scientometrics, DOI 10.1007/s11192-019-03147-3, preprint: https://arxiv.org/abs/1905.09095 - Bornmann, L. & Marx, W. (2014a). Distributions instead of single numbers: percentiles and beam plots for the assessment of single researchers. Journal of the American Society of Information Science and Technology, 65(1), 206–208 - Bornmann, L. & Marx, W. (2014b). How to evaluate individual researchers working in the natural and life sciences meaningfully? A proposal of methods based on percentiles of citations. Scientometrics, 98(1), 487-509. DOI: 10.1007/s11192-013-1161-y. - Bornmann, L., & Haunschild, R. (2018). Plots for visualizing paper impact and journal impact of single researchers in a single graph. Scientometrics, 115(1), 385-394. DOI: 10.1007/s11192-018-2658-1.

Examples

```
## Not run: inv_perc_beamplot(rd, au_name='Name of researcher')
```

`jpscatter` *Create a scatter plot using journal and paper percentile values*

Description

Provide journal and paper percentile values in a data frame, e.g. df, and the function call jpscatter(df) creates the scatter plot. The function jpscatter takes some optional arguments to modify its behaviour, see arguments and details.

Usage

```
jpscatter(df, off_set = 0, print_stats = TRUE, do_plot = TRUE, digits = 1, ...)
```

Arguments

`df`	data frame with journal and paper percentiles
`off_set`	determines the location of additional plotted information (number of points in each quadrant), values between 0 and 40 might be useful (optional parameter). The default value is 0.
`print_stats`	boolean variable (optional parameter) which determines if the additional statistical values are printed to the R console (T: yes print, F: no do not print). The default value is T.
`do_plot`	boolean variable (optional parameter) which determines if the scatter plot is actually produced (T: yes plot, F: no do not plot). The default value is T.

`digits`	integer value to determine the number of desired digits after the decimal point for statistical values (optional parameter). The default value is 1.
`...`	additional arguments to pass to the plot function

Details

jpscatter(df=data_frame, off_set=numeric_value, print_stats=boolean, do_plot=boolean, digits=integer)
Only the argument df is necessary. All other aruments are optional.

Literature:

- Bornmann, L., & Haunschild, R. (2017). Plots for visualizing paper impact and journal impact of single researchers in a single graph, DOI: 10.1007/s11192-018-2658-1, preprint: https://arxiv.org/abs/1707.04050

An example data frame is provided as `example_researcher` in the package. It can be used to create a scatter plot using default values.

Examples

```
data(example_researcher)

jpscatter(example_researcher)
```

`ncr_comp` *Create a spectrogram using data from the free software CRExplorer*

Description

Provide the contents of CSV files from the 'CRExplorer' in data frames, e.g. df1 and df2, and the function call ncr_comp(df1, df2, py1, py2) creates a plot with both sets of NCR values. Here, py1 and py2 are the lowest and highest publication year to be used in the plot. The function ncr_comp takes some optional arguments to modify its behaviour, see arguments and details.

Usage

```
ncr_comp(
  df1,
  df2,
  py1,
  py2,
  col_cr = "red",
  smoothing = TRUE,
  par_pch = 20,
  ...
)
```

Arguments

`df1`	data frame 1 with reference publication year and number of cited references, e. g., as exported from the CRExplorer (File > Export > CSV (Graph)).
`df2`	data frame 2 with reference publication year and number of cited references, e. g., as exported from the CRExplorer (File > Export > CSV (Graph)).
`py1`	determines lowest reference publication year which should be shown in the graph.
`py2`	determines highest reference publication year which should be shown in the graph.
`col_cr`	character color name value to determine color of the line and points of the number of cited references (optional parameter). The default value is "red".
`smoothing`	boolean variable (optional parameter) which determines if the lines of the spectrogram are smoothed or not. (T: yes apply smoothing, F: no do not apply smoothing). The default value is T.
`par_pch`	integer value to set the point type (optional parameter). The default value is 20.
`...`	additional arguments to pass to the plot, points, and lines functions.

Details

ncr_comp <- function(df1, df2, py1, py2, col_cr = "red", smoothing = TRUE, par_pch = 20, ...) Only the arguments df1, df2, py1, and py2 are necessary. All other aruments are optional. Please use the function `legend` to add a user-defined legend The solid curve represents the data from df1 and the dotted curve represents the data from df2.

Literature:

- Thor, A., Bornmann, L., Marx, W., Haunschild, R., Leydesdorff, L., & Mutz, Ruediger (2017). Website of the free software 'CRExplorer', http://www.crexplorer.net

---

`perc_beamplot` *Create a beamplot using percentile values*

Description

Create a beamplot using percentile values.

Usage

```
perc_beamplot(rd, au_name = "Example Researcher", ...)
```

Arguments

`rd`	is a dataframe with two columns: (i) publication year and (ii) percentile value with one row per paper/dataset.
`au_name`	is the name of the researcher this beamplot belongs to.
`...`	further parameters passed to stripchart.

Details

perc_beamplot(rd, au_name='Name of researcher') Only the rd is argument mandatory. It has to be a dataframe with two columns: (i) publication year and (ii) percentile value with one row per paper/dataset.

Literature:

- Haunschild, R., Bornmann, L., & Adams, J. (2019). R package for producing beamplots as a preferred alternative to the h index when assessing single researchers (based on downloads from Web of Science), Scientometrics, DOI 10.1007/s11192-019-03147-3, preprint: https://arxiv.org/abs/1905.09095 - Bornmann, L. & Marx, W. (2014a). Distributions instead of single numbers: percentiles and beam plots for the assessment of single researchers. Journal of the American Society of Information Science and Technology, 65(1), 206–208 - Bornmann, L. & Marx, W. (2014b). How to evaluate individual researchers working in the natural and life sciences meaningfully? A proposal of methods based on percentiles of citations. Scientometrics, 98(1), 487-509. DOI: 10.1007/s11192-013-1161-y. - Bornmann, L., & Haunschild, R. (2018). Plots for visualizing paper impact and journal impact of single researchers in a single graph. Scientometrics, 115(1), 385-394. DOI: 10.1007/s11192-018-2658-1.

Examples

```
## Not run: perc_beamplot(rd, au_name='Name of researcher')
```

`rpys` *Create a spectrogram using data from the free software CRExplorer*

Description

Provide the contents of the CSV (Graph) file from the 'CRExplorer' in a data frame, e.g. df, and the function call rpys(df, py1, py2) creates the spectrogram. Here, py1 and py2 are the lowest and highest publication year to be used in the plot. The function rpys takes some optional arguments to modify its behaviour, see arguments and details.

Usage

```
rpys(
  df,
  py1 = min(df$Year),
  py2 = max(df$Year),
  col_cr = "red",
  col_med = "blue",
  smoothing = TRUE,
  par_pch = 20,
  plot_NCR = TRUE,
  plot_Med = TRUE,
  ...
)
```

Arguments

`df`	data frame with reference publication year, number of cited references, and median deviation as exported from the CRExplorer (File > Export > CSV (Graph)).
`py1`	determines lowest reference publication year which should be shown in the graph (optional parameter).
`py2`	determines highest reference publication year which should be shown in the graph (optional parameter).
`col_cr`	character color name value to determine color of the line and points of the number of cited references (optional parameter). The default value is "red".
`col_med`	character color name value to determine color of the line and points of the median deviation (optional parameter). The default value is "blue".
`smoothing`	boolean variable (optional parameter) which determines if the lines of the spectrogram are smoothed or not. (T: yes apply smoothing, F: no do not apply smoothing). The default value is T.
`par_pch`	integer value to set the point type (optional parameter). The default value is 20.
`plot_NCR`	boolean variable (optional parameter) which determines the NCR curve should be plotted.
`plot_Med`	boolean variable (optional parameter) which determines the median deviation curve should be plotted.
`...`	additional arguments to pass to the plot, points, and lines functions.

Details

rpys(df=data_frame, py1=integer_value, py2=integer_value, smoothing=boolean, col_cr=character_color_name, col_med=character_color_name, par_pch=integer, plot_NCR=boolean, plot_Med=boolean, ...) Only the argument df is necessary. All other aruments are optional.

Literature:

- Thor, A., Bornmann, L., & Haunschild, R. (2021). Website of the free software 'CRExplorer', http://www.crexplorer.net - Thor, A., Bornmann, L., & Haunschild, R. (2018). CitedReferencesExplorer (CRExplorer) manual. Retrieved December 19, 2019, from https://andreas-thor.github.io/cre/manual.pdf

An example data frame is provided as `rpys_example_data` in the package. It can be used to create an example spectrogram.

Examples

```
data(rpys_example_data)

rpys(rpys_example_data, 1935, 2010)
```

---

rpys_bl *Create a spectrogram with bars and lines using data from the free software CRExplorer*

Description

Provide the contents of the CSV (Graph) file from the 'CRExplorer' in a data frame, e.g. df, and the function call rpys_bl(df) creates a spectrogram. Previously, you should use the function rpys for a plain line graph to determin the proper parameters, e.g., x_offset and x_range. Determination of the proper x_offset and x_range is a bit tricky. Usage of a wrong value of x_range will cause an error. Usage of a wrong value of x_offset will produce a plot. However, the line for the median deviation and the bars might not be at the proper location. First, adjust x_range if necessary, and second, adjust x_offset so that the x axis is properly aligned with the line and bars. Comapare the plot from rpys_bl with your data and the plot from the function rpys. The function rpys_bl takes some optional arguments to modify its behaviour, see arguments and details.

Usage

```
rpys_bl(
  df,
  py1 = min(df$Year),
  py2 = max(df$Year),
  x_range = py2 - py1 + 1,
  col_cr = "grey",
  col_med = "blue",
  col_ol = "red",
  smoothing = TRUE,
  par_mar = c(5, 5, 1, 5),
  x_offset = 0,
  x_min = py1,
  x_max = py2,
  x_step1 = 10,
  x_step2 = 5,
  y1_min = 0,
  y1_max = max(df$NCR),
  y1_step = (max(df$NCR) - min(df$NCR))/5,
  y2_min = min(df$Median.5),
  y2_max = max(df$Median.5),
  y2_step = (max(df$Median.5) - min(df$Median.5))/5,
  lx = median(df$Year),
  ly = median(df$Median.5),
  pl_offset = (max(df$NCR) - min(df$NCR))/50,
  bar_border = "white",
  outliers = 2,
  lpos = 3,
  pl_cex = 0.9,
  TFmin = py1,
```

```
  TFmax = py2,
  plot_NCR = TRUE,
  plot_Med = TRUE,
  ...
)
```

Arguments

`df`	data frame with reference publication year, number of cited references, and median deviation as exported from the CRExplorer (File > Export > CSV (Graph)).
`py1`	determines lowest reference publication year which should be shown on the x axis (optional parameter). The default is the minimum RPY.
`py2`	determines highest reference publication year which should be shown on the x axis (optional parameter). The default is the maximum RPY.
`x_range`	is the range of the x axis (optional parameter). The default is py2-py1+1.
`col_cr`	is a character color name value to determine color of the bars of the number of cited references (optional parameter). The default value is "grey".
`col_med`	is a character color name value to determine color of the line of the median deviation (optional parameter). The default value is "blue".
`col_ol`	is a character color name value to determine color of the outlier labels (optional parameter). The default value is "red".
`smoothing`	boolean variable (optional parameter) which determines if the lines of the spectrogram are smoothed or not. (T: yes apply smoothing, F: no do not apply smoothing). The default value is T.
`par_mar`	integer vector to set the margins (optional parameter). The default value is c(5, 5, 1, 5).
`x_offset`	determines the x axis offset to adjust the median deviation curve properly (optional parameter). The default is 0.
`x_min`	determines lowest reference publication year which should be shown on the x axis (optional parameter). The default is the minimum RPY.
`x_max`	determines highest reference publication year which should be shown on the x axis (optional parameter). The default is the maximum RPY.
`x_step1`	is the interval of major x tics (optional parameter).
`x_step2`	is the interval of minor x tics (optional parameter).
`y1_min`	is the minimum left y axis value (optional parameter).
`y1_max`	is the maximum left y axis value (optional parameter).
`y1_step`	is the interval left y axis (optional parameter).
`y2_min`	is the minimum right y axis value (optional parameter).
`y2_max`	is the maximum right y axis value (optional parameter).
`y2_step`	is the interval right y axis (optional parameter).
`lx`	is the x position of the legend (optional parameter).
`ly`	is the y position of the legend according to the right y axis (optional parameter).

`pl_offset`	is the offset of the year label (optional parameter).
`bar_border`	is the color around the bars (optional parameter).
`outliers`	is an integer that indicates if outliers should be detected (optional parameter): (0: no outlier detection, 1: outliers are detected and marked, 2: only extreme outliers are detected and marked)
`lpos`	is an integer that determines the position of the outlier year label around the point (optional parameter). Values of 1, 2, 3, and 4, respectively indicate positions below, to the left of, above, and to the right of the specified coordinates.
`pl_cex`	is the cex value of the year labels (optional parameter).
`TFmin`	is the first year that should be used for outlier detection according to Tukey's fences.
`TFmax`	is the last year that should be used for outlier detection according to Tukey's fences.
`plot_NCR`	boolean variable (optional parameter) which determines the NCR curve should be plotted.
`plot_Med`	boolean variable (optional parameter) which determines the median deviation curve should be plotted.
`...`	additional arguments to pass to the plot function.

Details

rpys_bl(df=data_frame, py1=integer_value, py2=integer_value, x_range=integer_value, smoothing=boolean, col_cr=character_color_name, col_med=character_color_name, col_ol=character_color_name, par_mar=integer_vector, plot_NCR=boolean, plot_Med=boolean, x_offset=integer_value, x_min=integer_value, x_max=integer_value, x_step1=integer_value, x_step2=integer_value, y1_min=integer_value, y1_max=integer_value, y1_step=integer_value, y2_min=integer_value, y2_max=integer_value, y2_step=integer_value, lx=integer_value, ly=integer_value, pl_offset=integer_value, bar_border=string_value, outliers=integer_value, lpos=integer_value, pl_cex=floating_point_value, TFmin=integer_value,TFmax=integer_value, ...) Only the argument df is necessary. All other aruments are optional, but many should be provided to produce nice plots.

Literature:

- Thor, A., Bornmann, L., & Haunschild, R. (2021). Website of the free software 'CRExplorer', http://www.crexplorer.net - Thor, A., Bornmann, L., & Haunschild, R. (2018). CitedReferencesExplorer (CRExplorer) manual. Retrieved December 19, 2019, from https://andreas-thor.github.io/cre/manual.pdf - Tukey, J. W. (1977). Exploratory data analysis. Boston, MA, USA: Addison-Wesley Publishing Company.

An example data frame is provided as `rpys_example_data` in the package. It can be used to create an example spectrogram.

Examples

```
data(rpys_example_data)

rpys_bl(rpys_example_data)

rpys_bl(rpys_example_data, x_min=1930, x_max=2020, x_range=91, x_offset=1, lx=1926, ly=135,
y1max=300, y1_step=50, y2_min=-150, y2_max=150, y2_step=25, lpos=1)
```

```
rpys_bl(rpys_example_data, py1=1930, py2=2020, x_offset=1, lx=1926, ly=135, y1max=300,
y1_step=50, y2_min=-150, y2_max=150, y2_step=25, lpos=1)
```

`rpys_example_data` *Example data set for the rpys function*

Description

Contains the data sets (`rpys_example_data`).

Index