GraphLib

Release 1.0

U64053

Apr 29, 2020

CONTENTS:

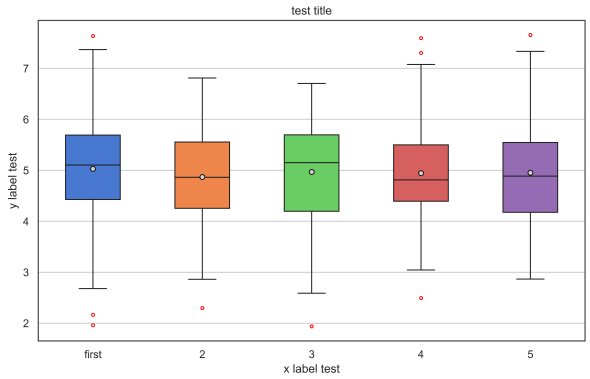
This library is intended for graph representation. Check the graphical index for the graphs available.

CHAPTER

ONE

GRAPH INDEX

You can find here a index of the available graphs.



boxPlot

test header

	F	FH	Total landings	FS landings
S1	15000	21000	42000	10000
S2	12000	15000	11000	2396
All	27000	36000	53000	12396

test header

	AR land	RL	T&G	Bounces
S1	10	100	8000	19000
S2	1	600	9500	2000
All	11	700	17500	21000

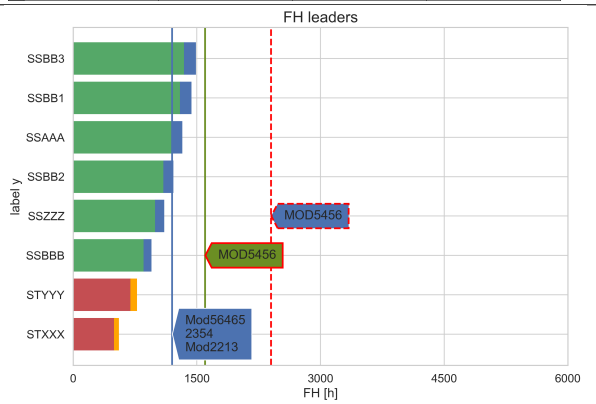
test header

	Probe Deploy	In-flight Deploy	MNT probe Deploy	Probe contacts
S1	not available	4000	not available	not available
S2	not available	700	not available	not available
All	not available	4700	not available	not available

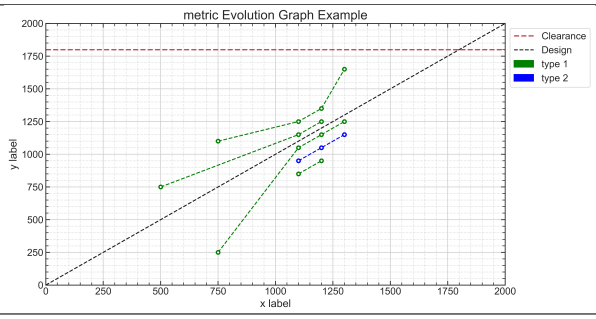
test header

	Total Cycles	In-flight Cycles	MNT Cycles	Parachute
S1	45000	37000	800	550
S2	16000	11000	444	333
All	61000	48000	1244	883

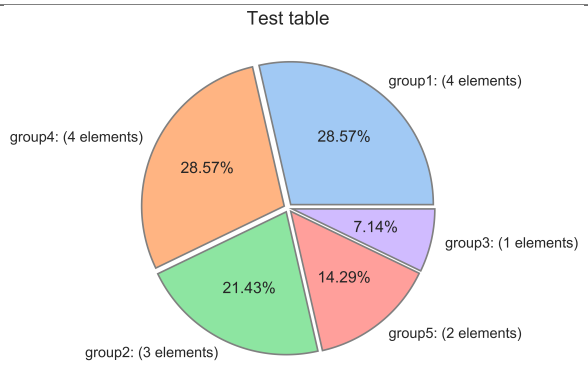
enhancedTable



hBarPlot



metricsEvolution



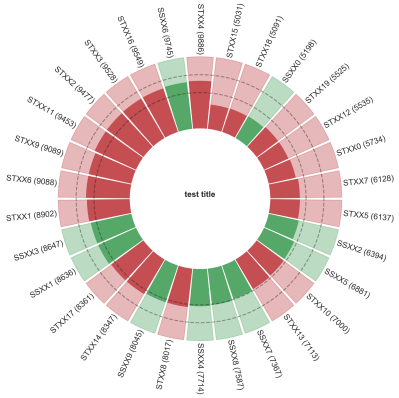
*footnote for test

ocurrPieChart

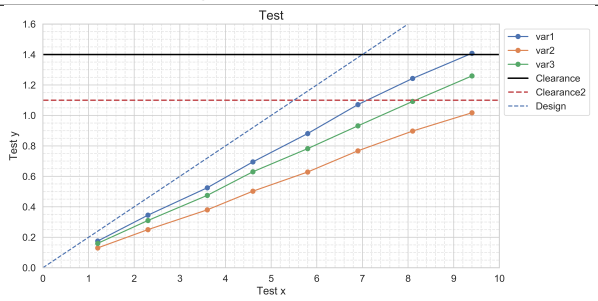
Tranche	Batch	Block	S1	S2	Total
T1	1	1A	0	2	2
		1B	1	1	2
		1C	0	1	1
	2	2B	2	1	3
	3	2C	0	1	1
		2D	1	0	1
T2	2	8A	1	1	2
	3		0	1	1
		8B	2	1	3
		8C	1	2	3
		8D	1	0	1
T3	4	20	3	4	7
		30	2	1	3
			14	16	30

*footnote for test

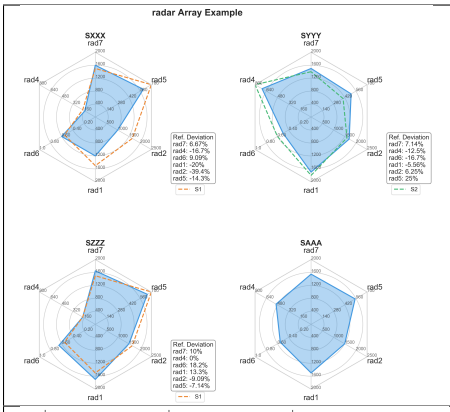
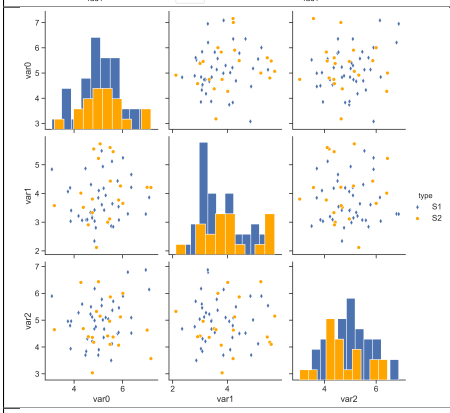
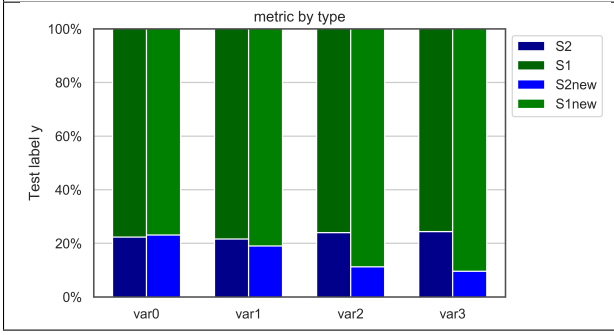
ocurrTable

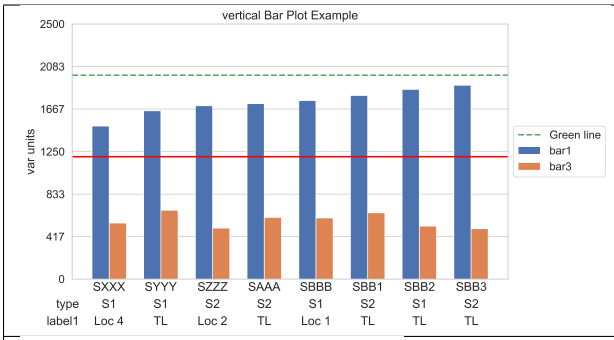
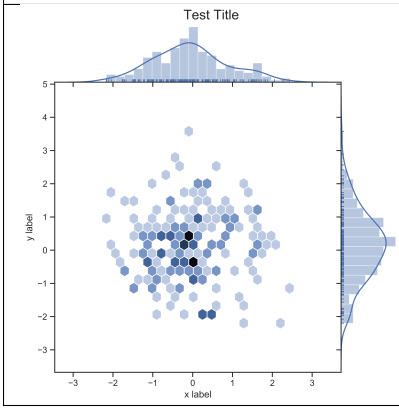


polarBarPlot



progressPlot

	radarArray
	scatterPlotMatrix
	stackedBars

	verticalBarPlot
	scattHist

CHAPTER

TWO

BOXPLOT

@author: U64053

boxPlot.**boxPlotter**(*df*, *graphTitle*, *boxCols*, *save=True*, ***kwargs*)

Represents a box plot from the data of a dataframe.

Parameters

df [pandas DataFrame] dataframe containing the data to plot.

graphTitle [string] title of the plot, used as fileName if not provided.

boxCols [list of strings] Columns of df to plot in the boxplots.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

tickLabels [list of strings, optional] Labels for the box plot, must have the same lenght as boxCols. If not provided, boxCols is used.

colors [list of strings, optional] Colors to color the boxplos, must have the same lenght as boxCols. If not provided the default palette is used.

xLabel [string, optional] Label for the x axis.

yLabel [string, optional] Label for the y axis.

showMeans [boolean, optional, default is True] Show mean marker.

showFliers [boolean, optional, default is True] Show fliers markers.

notch [boolean, optional, default False] Activate notched boxes.

xSize [float, optional, default is 10.] X dimension of the figure in inches.

ySize [float, optional, default is 6.] Y dimension of the figure in inches.

yLim [list of floats (min, max), optional] It is used to set y axis limits.

cappkw [dict, optional, {“color”: “k”}] Specifies the style of the caps.

boxeskw [dict, optional, default is {“color”:”k”}] Specifies the style of the box.

whiskerskw [dict, optional, default is {“color”:”k”}] Specifies the style of the whiskers.

flierskw [dict, optional] Specifies the style of the fliers. The default is {“marker”:’.’, “markeredgecolor”:”k”}.

medianskw [dict, optional, default {“color”:”k”}] Specifies the style of the median.

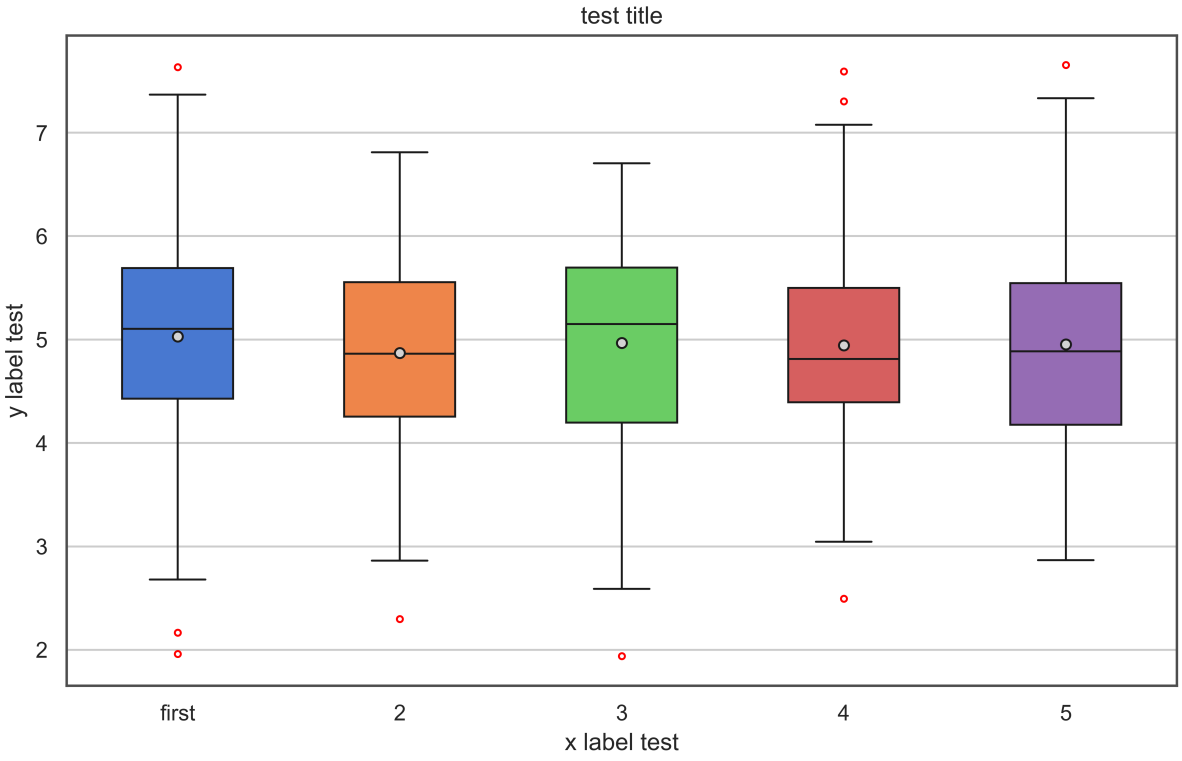
meankw [dict, optional] Specifies the style of the mean. The default is {"marker":"o", "markerfacecolor":"lightgrey", "markeredgecolor":"k", "markersize":"5"}.

palette [seaborn compatible palette, default "deep"] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] File name to save, if not provided graphTitle will be used. Non alphanumeric values will be deleted.

Examples

```
>>> data = {"loc" + str(x): np.random.normal(5, 1, 80) for x in range(1, 17)}
>>> data["AC"] = ["AC0" + str(ac) for ac in range(1, 81)]
>>> df = pd.DataFrame(data)
>>> boxPlotter(df, "test title",
...            ["loc1", "loc2", "loc3", "loc4", "loc5"],
...            yLabel="y label test",
...            xLabel="x label test",
...            tickLabels=["first", 2, 3, 4, 5],
...            flierskw = {"marker":'.', "markeredgecolor":"red"},
...            palette = "muted")
```



CHAPTER

THREE

ENHANCEDTABLE

@author: u64053

enhancedTable.**enhancedTable**(*df*, *fileName*, *rowLabelCol*, *save=True*, ***kwargs*)

Represent a table or several tables. Options to add a total row, header, labels...

Parameters

df [pandas Dataframe] Dataframe containing the data.

fileName [string] Filename to use to save the table, non alphanumeric characters will be removed.

rowLabelcol [string] Name of the dataframe column of df to use as label values.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

labelStyle [string, optional, default "label",] Style of the labels, "label" or "column".

totalRow [boolean, optional, default False] If true add a new row with the total per colum.

header [string, optional, default False] If it has a value add a header cell with that value.

tableNum [integer, optional, default 1.] Number of tables to split the data in.

layout [string, optional, default "vertical"] Layout to place the tables relative to each others, can be 'horizontal' or 'vertical'.

fontSize [integer, optional, default False.] Size of the font, if False, let the code decide the best horizontal fit.

xSize [float, optional, default 18] Max horizontal size of the figure, in inches.

ySize [float, optional, default 8.] Max vertical size of the figure, in inches.

rowLabelColor [string, optional, default "grey"] Color to use for the labels, named colors or HEX values.

colLabelColor [string, optional, default "grey"] Color to use for the labels, named colors or HEX values.

minimalistic [boolean, optional, default False.] If true, most of the cell lines are hidden.

palette [seaborn palette or compatible, optional.] Seaborn compatible palette, can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

Examples

```
>>> data={"Type":["S1", "S2"],
...       "F":[15000, 12000],
...       "FH":[21000, 15000],
...       "Total landings":[42000, 11000],
...       "FS landings":[10000, 2396],
...       "AR land":  [10, 1],
...       "RL": [100, 600],
...       "T&G": [8000, 9500],
...       "Bounces":[19000, 2000],
...       "Test Deploy":["not available", "not available"],
...       "In-flight  Deploy":[4000, 700],
...       "MNT probe Deploy":["not available", "not available"],
...       "Contacts":["not available", "not available"],
...       "Total Cycles":[45000, 16000],
...       "In-flight Cycles":[37000, 11000],
...       "MNT Cycles":[800, 444],
...       "Antena":[550, 333]}
>>> df = pd.DataFrame.from_dict(data)
>>> kw = dict(labelStyle="label",
...           totalRow=True,
...           header="test header",
...           tableNum=4,
...           layout="vertical",
...           fontSize=18,
...           xSize=16,
...           ySize=12,
...           rowLabelColor="grey",
...           colLabelColor='b')
>>> fig = enhancedTable(df, "test table", rowLabelCol="Type",
...                     save=True, **kw)
```

	test header			
	F	FH	Total landings	FS landings
S1	15000	21000	42000	10000
S2	12000	15000	11000	2396
All	27000	36000	53000	12396

	test header			
	AR land	RL	T&G	Bounces
S1	10	100	8000	19000
S2	1	600	9500	2000
All	11	700	17500	21000

	test header			
	Probe Deploy	In-flight Deploy	MNT probe Deploy	Probe contacts
S1	not available	4000	not available	not available
S2	not available	700	not available	not available
All	not available	4700	not available	not available

	test header			
	Total Cycles	In-flight Cycles	MNT Cycles	Parachute
S1	45000	37000	800	550
S2	16000	11000	444	333
All	61000	48000	1244	883

CHAPTER

FOUR

HORIZONTALBARPLOT

@author: u64053

horizontalBarPlot.**hBarPlot**(*df*, *yCol*, *xCols*, *xLabel*, *graphTitle*, *save=True*, ***kwargs*)

Represents a horizontal bar graph with vertical lines, anotations and text under the graph.

Parameters

df [pandas dataframe] Dataframe with the data to represent.

yCol [string] df column name to represent in the y axis.

xCols [string or list of strings] df column or column names to represent in the x axis. If multiple columns are used, stacked bars will be used.

xLabel [string] label for the x axis.

graphTitle [string] Title of the graph, will be used as file name if not provided.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

colorCol [string, optional, default False] df col to use as label to choose color. If False all bars are colored equally.

colorDict [list of dictionaries, default False] List with as many elements as xCols with dicts with colorCol label:color to use to color the bars. Example: [{"S1": "g", "S2": "r"}, {"S1": "b","S2": "orange"}]

xLim [float, optional, default False] Limit of the x axis in x data units.

nLim [integer, optional, default False.] Limit of the number of elemnents of yCol displayed in the y axis.

ascending [boolean, optional, default False.] False for higher values, True of lower values.

vLabels [list of lists] List of lists with annotations parameters in the format: [[x1, y1, text1, color1, style1] , [x2, y2, text2, color2, style2] , . . .]

- x : float, x in data units where the vertical line is.
- y : integer, y bar position to place the annotation.
- text : string, text to display in the annotation.
- color : string, rgb color for the line and annotation.
- style : string, ("regular", "highlighted", "dashed", "regularD").

yLabel [string, optional.] Label for the y axis.

xSize [float, optional, default 12] Width of the figure in inches.

ySize [float, optional, default 9] Height in inches.

fontScale [float, optional, default 1] Factor scale for the text.

xTicksNum [integer, optional.] Number of x axis ticks.

palette [seaborn compatible palette, optional] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] If not provided graphTitle will be used. Non alphanumeric values will be deleted.

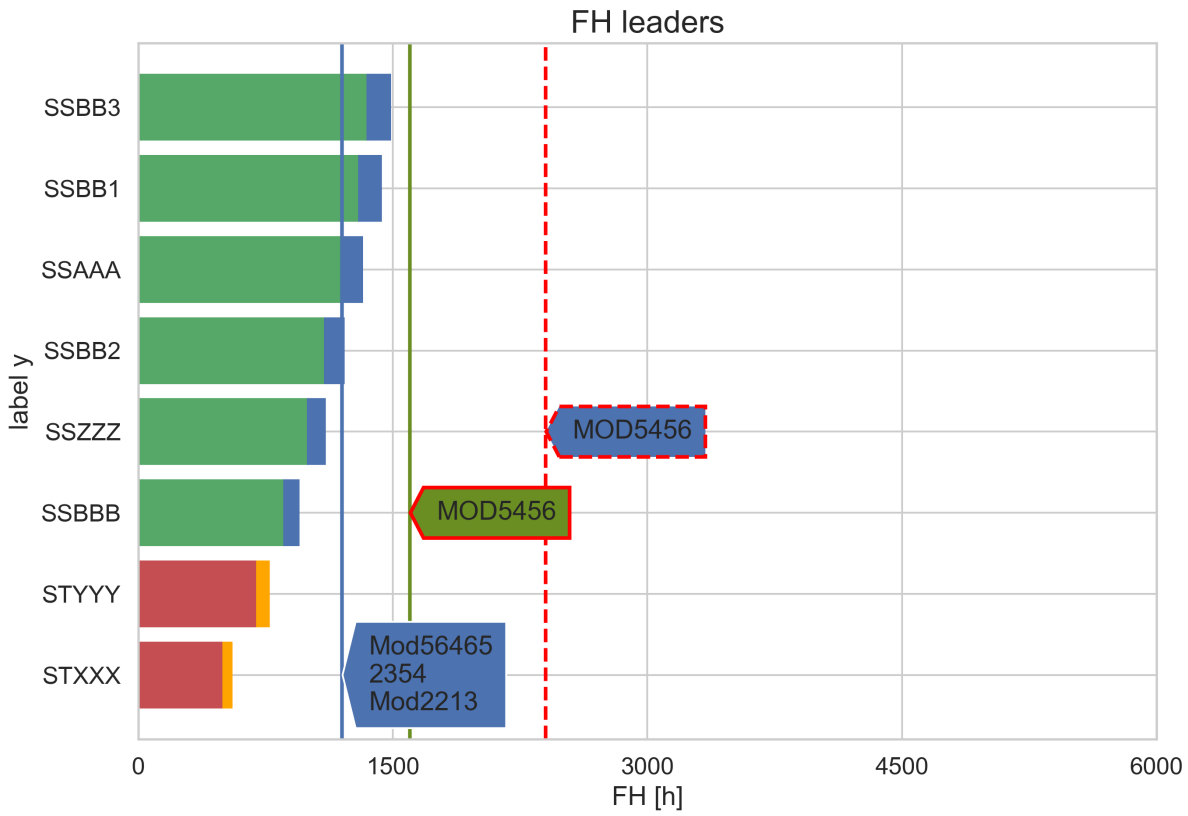
orderCol [string, optional, default false.] Column to use to order the df. If false use the first element of xCols

yElems [list of strings, optional] List with the elements of the ycol you wish to represent. If not provided, all elements are represented. It is not recomended to use it with the kw nLim.

subText [string, optional.] If not False, add text to the bottom of the graph.

Examples

```
>>> data={"AC":["STXXX", "STYYY", "SSZZZ", "SSAAA",
...             "SSBBB", "SSBB1", "SSBB2", "SSBB3"],
...       "Type": ["S2", "S2", "S1", "S1", "S1", "S1", "S1", "S1"],
...       "FH":[500, 700, 1000, 1200, 860, 1300, 1100, 1350],
...       "FH Period":[50, 70, 100, 120, 86, 130, 110, 135]}
>>> df = pd.DataFrame.from_dict(data)
>>> v_labels=[[1200, 0, 'adf5646asdf2213', "b", "regular"],
...           [1600, 2, 'asd5456', "#6B8E23", "highlighted"],
...           [2400, 3, 'ads5456', "b", "dashed"]]
>>> keywords=dict(xLim=6000,
...               vLabels=v_labels,
...               yLabel="label y",
...               colorCol="Type",
...               xSize=10,
...               ySize=7,
...               colorDict=[{"S1":"g", "S2":"r"},
...                          {"S1":"b","S2":"orange"}],
...               fontScale=1.2,
...               xTicksNum=4,
...               patette="deep")
>>> hBarPlot(df, "AC", ["FH", "FH Period"],
...          "FH [h]", "FH leaders", **keywords)
```


FH leaders
label y
SSBB3
SSBB1
SSAAA
SSBB2
SSZZZ
SSBBB
STYYY
STXXX
MOD5456
MOD5456
Mod56465
2354
Mod2213
0
1500
3000
4500
6000
FH [h]

CHAPTER

FIVE

METRICSEVOLUTION

@author: u64053

metricsEvolution.**metricEvolutionGraph**(*data*, *graphTitle*, *save=True*, ***kwargs*)

Represents a metric evolution graph to show the progress of multiple elements colored by a label value. Elements don't need to have the same number of points. Can represent one line (design reference line) and a horizontal line (clearance line).

Parameters

data [list of list,] list of lists, each list containing 3 elements, list with the x, y, and label. For example [[[x1, x2, x3], [y1, y2, y3], label1], [[x1, x2, x3], [y1, y2, y3], label2], ...] Label values will be used to choose a color for the lines.

graphTitle [string] Title of the graph, will be used as filename if fileName kwarg is not provided

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

fontScale [float, optional, default 1.] Global scalation factor for every text of the figure.

legendLoc [string, optional, default "best".] Legend location, accepts standart legend locations for matplotlib, plus "out top right", "out center right", "out lower right" for placing the legend out of the plotting area and "none" for hidding the legend.

xSize [float, optional, default 12] Horizontal size of the figure, in inches.

ySize [float, optional, default 8] Vertical size of the figure, in inches.

gridMinor [boolean, optional, default True.] If True, minor grid is displayed.

xLabel [string, optional.] Label for the x axis.

yLabel [string, optional.] Label for the y axis.

xLim [float, optional.] Right limit of the x axis in x data units.

yLim [float, optional] Top limit of the y axis in y data units.

designLine [boolean, default False.] If True, the design reference line is represented.

designSlope [float, optional, default 1] Slope for the design line.

designIntercept [float, optional,default 0.] Intercept for the design line.

designColor [string, optional, default black.] Named color or hex code for the design reference line.

designLabel [string, optional] Label of the design reference line, it will be displayed in the legend.

clearanceLine [boolean, optional, default False] If True, the clearance line is represented.

clearancey [float, optional, default 0] Location in y units for the horizontal line

clearanceColor [string, optional, default r] Named color or hex code for the clearance horizontal line.

clearanceLabel [string, optional] Label of the clearance line, it will be displayed in the legend.

palette [seaborn compatible palette, optional] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

colorDict [dictionaries label: color, optional] Dict with label:color to use to color the progression lines, if not provided, the palette will be used.

fileName [string, optional.] If not provided graphTitle will be used. Non alphanumeric values will be deleted.

markerShape [string, optional, default "o"] Shape of the marker. Check matplotlib.markers for the possible values.

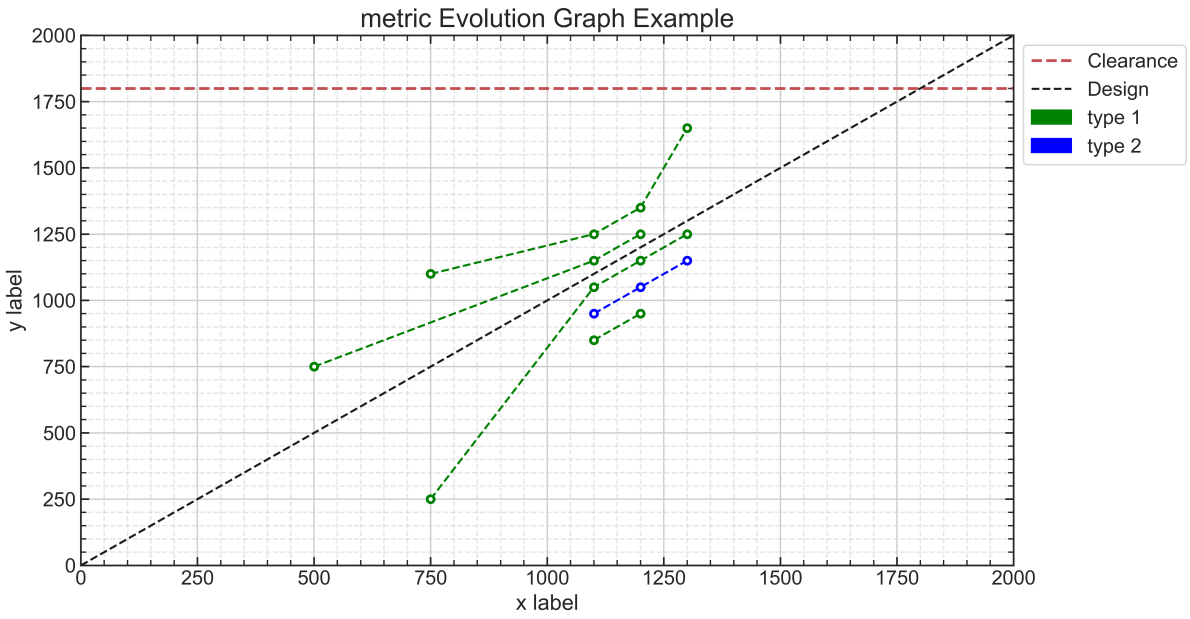
markerFill [string, optional, default "w"] Color to fill the marker, named color or hex code, use "none" for empty circles and None to fill the circles with the color of the edges.

markerSize [float, optional, default 5] Size of the markers.

markerEdgeWidth [float, optional, default 2] Linewidth ot the edges of the markers.

Examples

```
>>> dataIn = [[[750, 1100, 1200, 1300], [1100, 1250, 1350, 1650], "type 1"],
...           [[500, 1100, 1200], [750, 1150, 1250], "type 1"],
...           [[750, 1100, 1200, 1300], [250, 1050, 1150, 1250], "type 1"],
...           [[1100, 1200, 1300], [950, 1050, 1150], "type 2"],
...           [[1100, 1200], [850, 950], "type 1"]]
>>> metricEvolutionGraph(dataIn, "metric Evolution Graph Example",
...                      gridMinor=True, legendLoc="out top right",
...                      xSize=12, ySize=7,
...                      xLabel="x label", yLabel="y label",
...                      xLim=2000, yLim=2000,
...                      designLine=True, designSlope=1,
...                      designIntercept=0, designColor="k",
...                      designLabel="Design", clearanceLine=True,
...                      clearancey=1800,  clearanceColor="r",
...                      clearanceLabel="Clearance",
...                      markerSize=5, markerFill="w",
...                      markerShape="o",
...                      colorDict={"type 1": "green", "type 2": "blue"},
...                      palette="deep")
```


metric Evolution Graph Example
y label
x label
0
250
500
750
1000
1250
1500
1750
2000
Clearance
Design
type 1
type 2

CHAPTER

SIX

OCURRENCEPIECHART

@author: u64053

ocurrencePieChart.**ocurrPieChart**(*df*, *countCol*, *graphTitle*, *save=True*, ***kwargs*)

Generates a pie chart according to the occurence of values in a column of a dataframe.

Parameters

df [pandas dataframe] Dataframe containing the data.

countCol [string] Column of the df to count the ocurrence of the values.

graphTitle [string,] title of the graph and name of the file if fileName is not provided.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

sufix [string, optional] sufix to the labels

xSize [float, optional, default 8,] max horizontal size of the figure, in inches.

ySize [float, optional, default 8,] max vertical size of the figure, in inches.

fontScale [float, optional, default 1,] global font scalation factor for every text of the figure.

linewidth [float, default 0,] Contour line width for the pie slices in pixels, if 0 no line is displayed.

linecolor [strings, optional, default "k",] Colors to use for the contour lines to use, hex value or named color.

shadow [boolean, optional, default False] If True, shadows are displayed for the pie.

explode [float, optional, default 0,] exploding factor for the pie chart, it separates the pie slices. If 0, no explosion is performed.

footnote [string, optional] footnote text to display

foonoteSize [float, optionial, default 18.] Font size of the footnote.

footnoteColor [string, optional, default "grey"] colors the text of the footnote, hex value or named color.

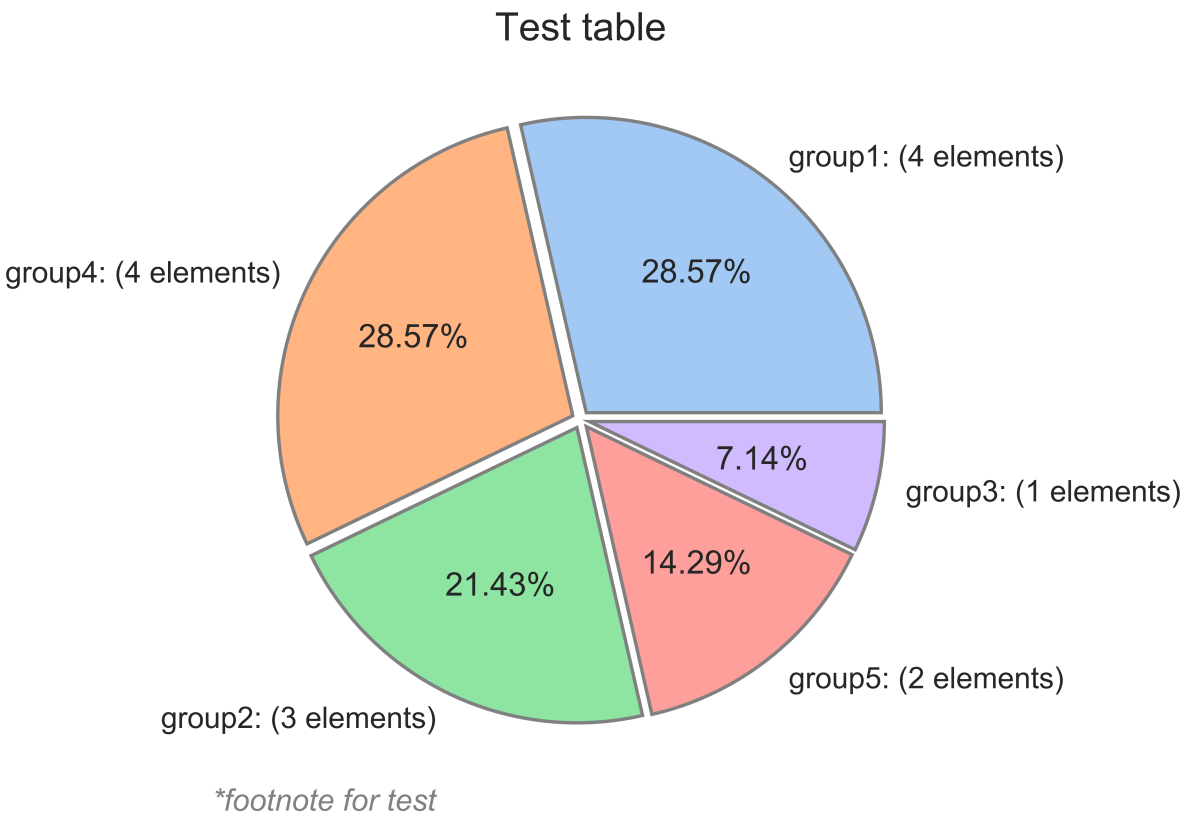
footnoteStyle [string, optional, default "italic"] style for the footnote, valid values: 'normal', 'italic', 'oblique'.

palette [seaborn compatible palette, optional] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] if not provided graphHeader will be used. Non alphanumeric values will be deleted.

Examples

```
>>> df = pd.DataFrame(
...                 {'element': ['elemXX', 'elemX1', 'elemX2',
...                              'elemX3', 'elemX4', 'elemX5',
...                              'elemX6', 'elemX7', 'elemX8',
...                              'elemX9', 'elem10', 'elem11',
...                              'elem12', 'elem13'],
...                  'groups': ['group1','group1','group1','group1',
...                             'group2','group2','group2','group3',
...                             'group4','group4','group4','group4',
...                             'group5','group5']})
>>> ocurrPieChart(df,
...                countCol='groups',
...                graphTitle="Test table",
...                sufix=" elements",
...                fontScale=1.6,
...                palette="pastel",
...                linewidth=2,
...                linecolor="grey",
...                explode=0.03,
...                footnote="*footnote for test ",
...                footnoteStyle='italic',
...                fileName= "ocurr Pie Chart example"
...                )
```



CHAPTER

SEVEN

OCURRTABLE

@author: u64053

ocurrTable.**ocurrenceTable**(*df*, *groupColumns*, *countColumn*, *fileName*, *save=True*, ***kwargs*)

Represents ocurrence and prints a grouped table from a python dataframe.

It Aggregates the diferent values of the Dataframes groupColumns to count occurrence in countColumn.

It allows to add totals by column or row.

Parameters

df [pandas dataframe] Dataframe containing the data.

groupColumns [list of strings] df columns names to display as grouped columns.

countColumn [string] df column name to count ocurrence.

fileName [string] filename to use to save the table, non alphanumeric characters will be removed.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

colColor [string, optional, default "grey"] Colors of the header row, hex value or named color.

fontSize [float, optional,] Font size of the table

xSize [float, optional, default 8,] max horizontal size of the figure, in inches.

ySize [float, optional, default 8,] max vertical size of the figure, in inches.

rowColoring [boolean, optional, default True] if True the cells are colored following the palette provided.

totalRow [boolean, optional, default False,] if True a "total per row" column will be generated.

totalCol [boolean, optional, default False,] if True a "total per column" row will be generated.

footnote [string, optional] footnote text to display

footnoteStyle [string, optional, default "italic"] style for the footnote, valid values: 'normal', 'italic', 'oblique'.

footnoteColor [string, optional, default "grey"] colors the text of the footnote, hex value or named color.

palette [seaborn compatible palette, optional] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

Examples

```
>>> df = pd.DataFrame({
...             "Tranche":["T1","T1","T1","T1","T1","T1","T1","T1","T1","T1",
...                        "T2","T2","T2","T2","T2","T2","T2","T2","T2","T2",
...                        "T3","T3","T3","T3","T3","T3","T3","T3","T3","T3"],
...             "Batch":[1,1,1,1,1,2,2,2,3,3,
...                      2,2,3,3,3,3,3,3,3,3,
...                      4,4,4,4,4,4,4,4,4,4],
...             "Block":["1A","1A","1B","1B","1C","2B","2B","2B","2C","2D",
...                      "8A","8A","8A","8B","8B","8B","8C","8C","8C","8D",
...                      "20","30","30","30","20","20","20","20","20","20"],
...             "Conf":["S2","S2","S2","S1","S2","S1","S2","S1","S2","S1",
...                     "S2","S1","S2","S1","S2","S1","S2","S1","S2","S1",
...                     "S2","S1","S2","S1","S2","S1","S2","S1","S2","S1"]
...             })
>>>  ocurrenceTable(df,
...                  groupColumns=["Tranche","Batch","Block"],
...                  colColor="grey",
...                  countColumn="Conf",
...                  palette="deep",
...                  fontSize=18,
...                  xSize=8,
...                  ySize=8,
...                  totalRow=True,
...                  totalCol=True,
...                  fileName="test ocurrence table",
...                  footnote="*footnote for test"
...                  )
```

Tranche	Batch	Block	S1	S2	Total
T1	1	1A	0	2	2
		1B	1	1	2
		1C	0	1	1
	2	2B	2	1	3
	3	2C	0	1	1
		2D	1	0	1
T2	2	8A	1	1	2
	3		0	1	1
		8B	2	1	3
		8C	1	2	3
		8D	1	0	1
T3	4	20	3	4	7
		30	2	1	3
			14	16	30

*footnote for test

CHAPTER

EIGHT

POLARPLOT

@author: u64053

polarPlot.**polarBarPlot**(*data*, *labelCol*, *valueCol*, *colorCol*, *graphTitle*, *save=True*, ***kwargs*)

It represents a polar bar plot with radial labels and title in the center.

Parameters

data [pandas dataframe] Dataframe containing the data.

labelCol [string] Name of the dataframe column containing the texts of the labels.

valueCol [string] Name of the dataframe column containing the data.

colorCol [string,] Name of the dataframe column containing the labels to use to color code the bars.

graphTitle [string] Title to place in the center of the graph. If no fileName is provided, it is used to name the file.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

sort [boolean, optional, default True] If true, order the dataframe in valueCol order.

xSize [float, optional, default 8] max horizontal size of the figure, in inches.

ySize [float, optional, default 8.] max vertical size of the figure, in inches.

barColors [dict, optional, default False.] Dict with label:color with label as colorCol column values and values as colors in hex values. If false the current palette is used.

clearance [float, optional] In data units, clearance of the data. Used to represend behind bars with alpha.

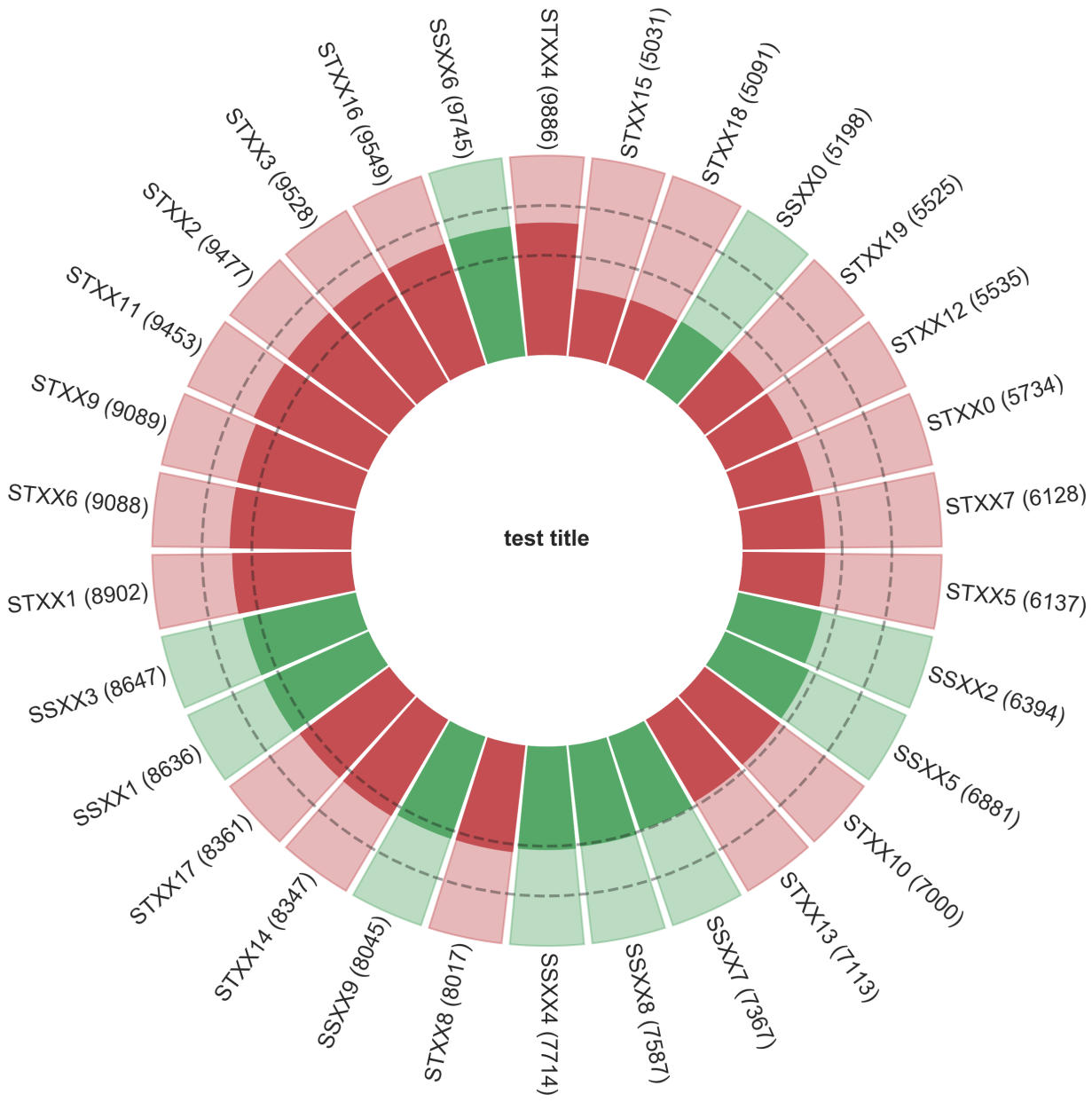
referLine [list of floats, optional] References lines in fraction of max data units or fraction of clearance if clearance is provided. Example:[0.25, 0.5, 0.75]

palette [seaborn compatible palette, optional] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] if not provided graphTitle will be used. Non alphanumeric values will be deleted.

Examples

```
>>> data = pd.DataFrame.from_dict(
...             {"element":["SSXX{}".format(x) for x in range(10)]
...                        + ["STXX{}".format(x) for x in range(20)],
...              "Type":["S1"]*10 + ["S2"]*20,
...              "magnitude": list(np.random.randint(5000,
...                                                  10000, size=(30)))})
>>> kwargs = dict(palette = "deep",
...               xSize = 8,
...               ySize = 8,
...               colors = False,
...               fileName = "polar Bar Plot Example",
...               clearance = 15000,
...               referLine = [0.5, 0.75],
...               barColors = {"S1": "g", "S2": "r"},
...               fontScale = 1)
>>> polarBarPlot(data, "element", "magnitude",
...              "Type", "test title", **kwargs)
```


test title
STXX4 (9886)
STXX15 (5031)
STXX18 (5091)
SSXX0 (5198)
STXX19 (5525)
STXX12 (5535)
STXX0 (5734)
STXX7 (6128)
STXX5 (6137)
SSXX2 (6394)
SSXX5 (6881)
STXX10 (7000)
STXX13 (7113)
SSXX7 (7367)
SSXX8 (7587)
SSXX4 (7714)
STXX8 (8017)
SSXX9 (8045)
STXX14 (8347)
STXX17 (8361)
SSXX1 (8636)
SSXX3 (8647)
STXX1 (8902)
STXX6 (9088)
STXX9 (9089)
STXX11 (9453)
STXX2 (9477)
STXX3 (9528)
STXX16 (9549)
SSXX6 (9745)

CHAPTER

NINE

PROGRESSPLOT

@author: u64053

progressPlot.**progressPlot**(*df*, *xCol*, *yCols*, *graphTitle*, *save=True*, ***kwargs*)

Plot progression lines from the data of a dataframe. To represent multiple columns as dependant variables and one column as independant variable. Allows to cumulate values, represent horizontal lines and a design reference line.

Parameters

df [pandas dataframe] Datafrane with the data to represent.

xCol [string] df column name to represent in the x axis.

yCols [list of strings] df columns names to represent in the x axis.

graphTitle [string] Title of the graph. If no fileName is provided, it is used to name the file.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

cumulative [boolean, optional, default False] If True cumulate the input dataframe.

xLabel [string, optional] Label for the y axis.

yLabel [string, optional] Label for the y axis.

legendLoc [string, optional, default "best"] Legend location, accepts standart legend locations for matplotlib, plus "out top right","out center right", "out lower right" for placing the legend out of the plotting area and "none" for hidding the legend.

yLim [float, optional] Limit of the y axis in y data units.

xLim [float, optional] Limit of the x axis in x data units.

xSize [float, optional default 10] Width of the figure in inches.

ySize [float, optional, default 6] Height in inches.

fontScale [float, optional, default 1.2] Scale factor for the font size.

xTicksNum [float, optinal] Number of ticks in the x axis.

yTicksNum [float, optional,] Number of ticks in the y axis.

hLines [list of lists, optional] Used to display horizontal lines, one list for each line. Each list containing a 3 or 4 elements list for each line:

- first, the y cordinate for the horizontal line,
- second, the color for the horizontal line,
- third, linestyle for the line, i.e. "-", "–", "-.-"
- fourth, label value to display in the legend (optional)

gridMinor [boolean, optional, default True] If True, minor grid is displayed.

designLine [boolean, optional, default False] If True, the design reference line is represented.

designSlope [float, optional, default 1] Slope for the design line.

designIntercept [float, optional, default 0] Intercept for the design line.

designColor [string, optional, default "k",] named color or hex code for the design reference line.

designLabel [string, optional] Label for the design reference line, it will be displayed in the legend.

markerShape [string, optional, default "o"] Shape of the marker. Check matplotlib.markers for the possible values.

markerFill [string, optional, default "w",] color to fill the marker, named color or hex code, use "none" for transparent circles and None to fill the circles with the color of the edges.

markerSize [float, optional, default 5] Size of the markers.

markerEdgeWidth [float, optional, default 2] linewidth ot the edges of the markers.

palette [seaborn compatible palette,] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] If not provided graphTitle will be used. Non alphanumeric values will be deleted.

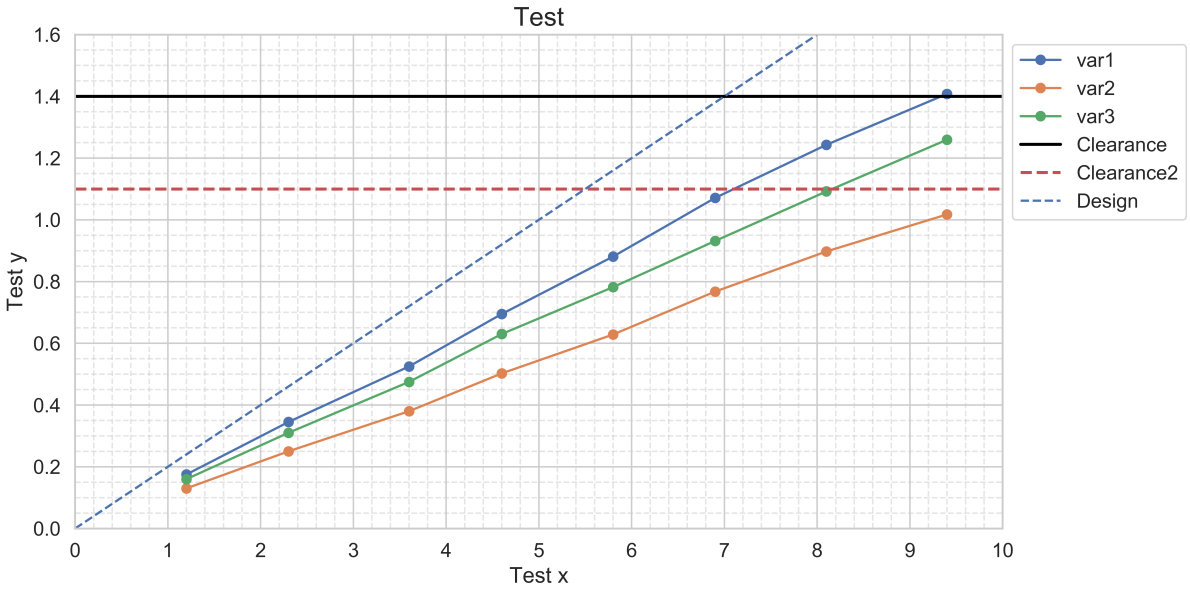
Examples

```
>>> df_test = pd.DataFrame.from_dict(
...                 {"var0":[1.2, 1.1, 1.3, 1, 1.2, 1.1, 1.2, 1.3],
...                  "var1":[0.175,0.17,0.18,0.17,0.186,0.19,0.172,0.165],
...                  "var2":[0.13,0.12,0.13,0.1225,0.126,0.139,0.13,0.12],
...                  "var3":[0.1600, 0.1500, 0.1650, 0.1550,
...                          0.1520, 0.1495, 0.1605, 0.1675]})
>>> progressPlot(
...             df=df_test,
...             xCol="var0",
...             yCols=["var1", "var2", "var3"],
...             graphTitle="Test",
...             yLabel="Test y",
...             xLabel="Test x",
...             legendLoc ="out top right",
...             cumulative=True,
...             yLim=1.6,
...             xLim=10,
...             xTicksNum=10,
...             yTicksNum=8,
...             hLines=[[1.4, "black", "-", "Clearance"],
...                     [1.1, "r", "--", "Clearance2"]],
```

(continues on next page)

(continued from previous page)

```
...             xSize=11,
...             ySize=6,
...             gridMinor=True,
...             designLine=True,
...             designSlope=0.2,
...             designColor="b",
...             designLabel="Design"
...             )
```



CHAPTER

TEN

RADARARRAY

@author: u64053

radarArray.**radarArrayGraph**(*df*, *titleCol*, *radiusCol*, *graphHeader*, *save=True*, ***kwargs*)

It represents an array of radarGraphs, with the possibility to use custom axes for each variable. Can represent a reference value and deviation form reference.

Parameters

df [pandas dataframe] Dataframe containing the data.

titleCol [string,] Dataframe column header containing the elements of the graph, one plot will be made for each element in this column and the value of this colum will be used as title.

radiusCol [list of strings,] variables to represent in the radius of the plots.

graphHeader: string Header to display for the graph, will be used for naming the saved file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

numPlotCols [int, optional] For the array of plots, number of columns. If not provided a layout as squared as posible will be used.

radTicksNum [int, optional, default 4] Number of circular grid lines.

xSize [float, optional, default 12] Max horizontal size of the figure, in inches.

ySize [float, optional, default 12] max vertical size of the figure, in inches.

multiScale [boolean, optional, default False.] If True, activates multiple escale mode for the radial axis of the plots.

radLim [float, optional,] if *multiScale* is False, maxiumun value for the radial axes.

radiusMax [list of floats, optional] if *multiScale* is True, maximun value for each of the axes, use the order defined in *radiusCol*.

fontScale [float, optional, default 1] global scalation factor for every text of the figure.

radTextSize [string, optional, default “large”] relative font size of the labels of the axes. Valid values are ‘xx-small’, ‘x-small’, ‘small’, ‘medium’, ‘large’, ‘x-large’, ‘xx-large’

scaleTextSize [string, optional, default “small”] relative font size of the labels of the plot grid. Valid values are ‘xx-small’, ‘x-small’, ‘small’, ‘medium’, ‘large’, ‘x-large’, ‘xx-large’

titleTextSize [string, optional, default “large”,] Relative font size of the title of each plot. Valid values are ‘xx-small’, ‘x-small’, ‘small’, ‘medium’, ‘large’, ‘x-large’, ‘xx-large’

headerTextSize [string, optional, default "x-large"] relative font size of the header of the figure. Valid values are 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'

headerVerPad [float, optional, default 0.08] vertical padding for the header of the figure.

color [string, optional, default False,] if false chooses the first color of the current palette.

reference [list of strings, optional, default False,] if not false, value in the dataframe in the col titleCol to use as a reference, this register will be used as a reference and not represented as a separated plot.

referCol [string, optional] Header of the col of the dataframe to use as key to add references. If not provided, all the itemps in *reference* are allways displayed.

referColor [list of strings, optional,] Colors to use to represent reference values. Must have more elements than reference.

devTable [boolean, optional, default False] If true, a table with de deviation from the reference is displayed.

legTitle [string, optional, default "Ref. Deviation"] text to display as the title of the deviation table.

palette [seaborn compatible palette,] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] File name to save, if not provided graphHeader will be used. Non alphanumeric values will be deleted.

Examples

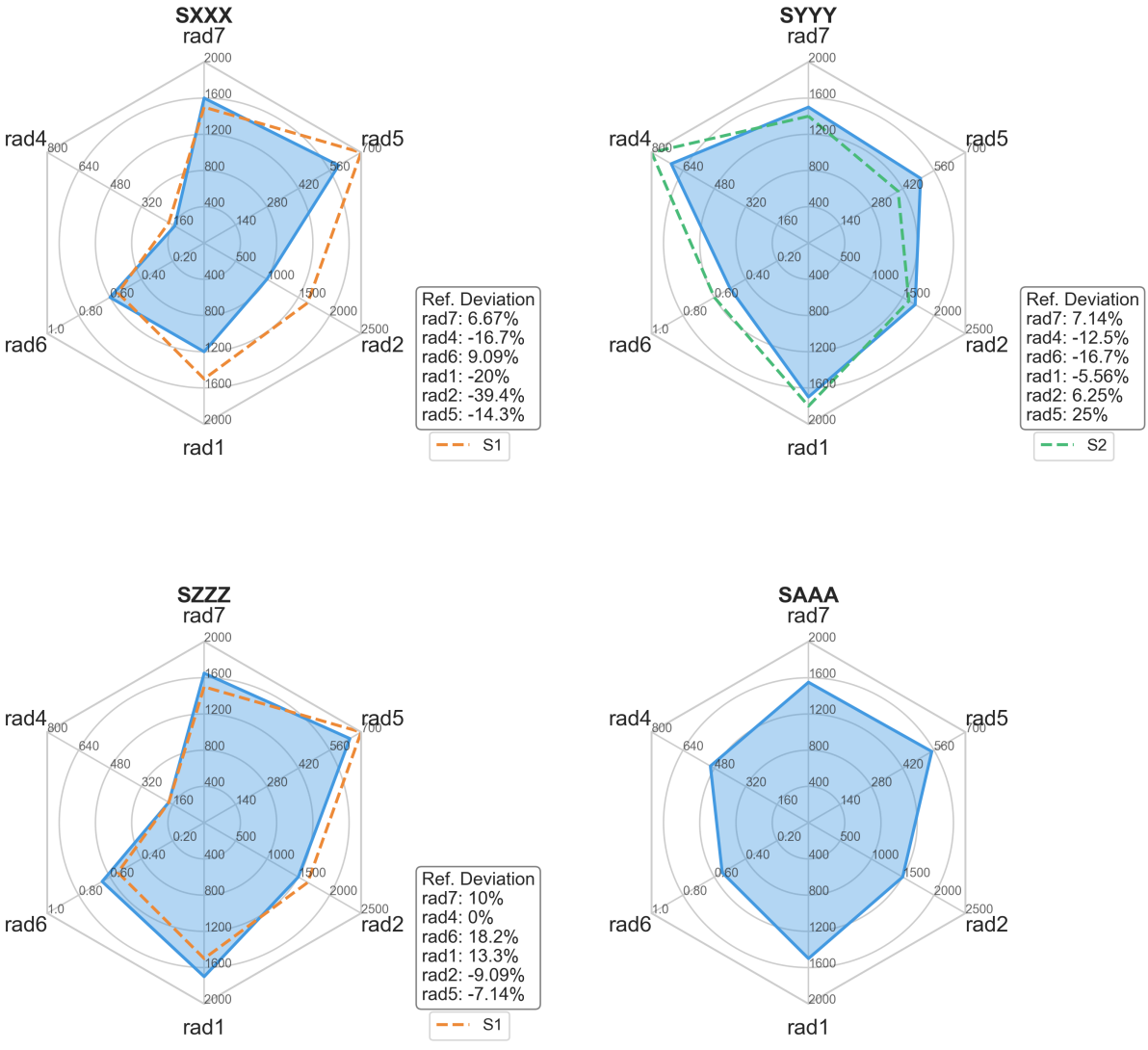
```
>>> data={"AC": ["S1","S2","SXXX","SYYY","SZZZ","SAAA",
...               "SBBB","SBB1","SBB2","SBB3","SB34","SB67"],
...       "rad1": [1500, 1800, 1200, 1700, 1700, 1500,
...                 1860, 1900, 1720, 1650, 650, 1234],
...       "rad2": [1650, 1600, 1000, 1700, 1500, 1500,
...                 1860, 1900, 1720, 1650, 650 ,1234],
...       "rad4": [180, 800, 150, 700, 180, 500,
...                 160, 900, 720, 650, 650, 1234],
...       "rad5": [700, 400, 600, 500, 650, 550,
...                 520, 495, 605, 675, 650, 1234],
...       "rad6":[0.550, 0.600, 0.600, 0.500, 0.650,
...               0.55, 0.52, 0.495, 0.605, 0.675, 0.65, 0.1234],
...       "rad7":[1500, 1400, 1600, 1500, 1650, 1550,
...               1520, 1495, 1605, 1675, 650, 1234],
...       "type":["na", "na", "S1", "S2", "S1",
...               "no data", "S1", "no data", "S1", "S2", "S1", "S2"]}
>>> df = pd.DataFrame.from_dict(data)
>>> radarArrayGraph(
...         df.head(6),
...         "AC",
...         ["rad7", "rad4", "rad6", "rad1", "rad2", "rad5"],
...         'radar Array Example',
...         numPlotCols=2,
...         radTicksNum=5,
...         xSize=12,
...         ySize=10,
...         radLim=False,
...         multiScale=True,
```

(continues on next page)

(continued from previous page)

```
...         devTable = True,
...         fontScale=1,
...         radiusMax=[2000, 800, 1, 2000, 2500, 700],
...         radTextSize="large",
...         scaleTextSize="x-small",
...         titleTextSize="large",
...         headerTextSize="x-large",
...         headerVerPad=0.08,
...         color='#4299E1',
...         reference=["S1", "S2"],
...         referCol="type",
...         palette=['#ED8936', '#48BB78', '#F56565', '#9F7AEA',
...                  '#38B2AC', '#ED64A6', '#A0AEC0',
...                  '#ECC94B', '#4299E1'])
```

radar Array Example



CHAPTER

ELEVEN

SCATTERPLOTMATRIX

@author: u64053

scatterPlotMatrix.**scatterMatrix**(*df*, *graphTitle*, *groupCol*, *correCols=None*, *save=True*, ***kwargs*)

Function to represent a scatter matrix with hist or kde in the diagonal

Parameters

df [pandas dataframe] Dataframe to plot.

graphTitle [string] Title for the plot, used as filename.

groupCol [string] Name of the df column to use to separate the df in groups.

correCols [list of strings, optional] List of the df columns to plot, if None all columns except groupCol are used.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [seaborn PairGrid object] figure that stores the generated graph.

Other Parameters

colors [Dict{label:color}, optional] If provided, labels of the groupCol column and the color to use. The default is False.

markers [string or list of strings, optional] markers to use for the different groups, use the same if only a marker is provided.

diagKind [string, optional, default "hist"] Values: "hist", "kde" or "auto" the plot to use for the diagonal.

fontScale [float, optional, default 1.2] Scale factor for the font size.

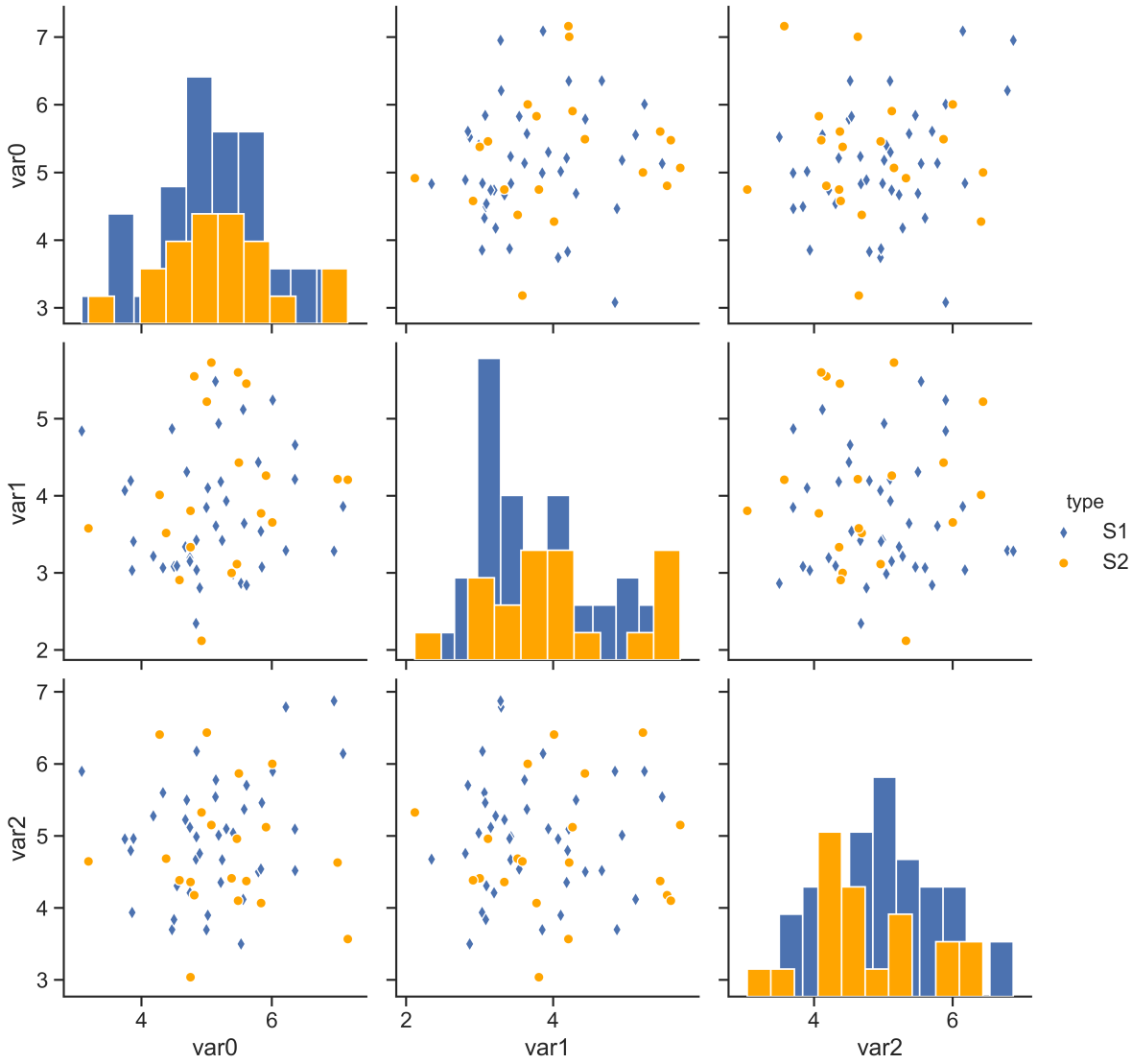
palette [seaborn compatible palette, default "deep"] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] File name to save, if not provided graphTitle will be used. Non alphanumeric values will be deleted.

{plot, diag, grid}_kwsdicts [dicts, optional] Dictionaries of keyword arguments. plot_kws are passed to the bivariate plotting function, diag_kws are passed to the diagonal plotting function, and grid_kws are passed to the PairGrid constructor

Examples

```
>>> df = pd.DataFrame.from_dict(
...         {"type": ["S1", "S2", "S1"]*20,
...          "var0": np.random.normal(5, 1, 60),
...          "var1": np.random.triangular(2, 3, 6, 60),
...          "var2": np.random.normal(5, 0.8, 60)})
>>> scatterMatrix(df, graphTitle="scatter plot", groupCol="type",
...               correCols=['var0', 'var1', 'var2'],
...               colors={"S1": "b", "S2": "orange"},
...               markers=["d", "o"], diagKind='hist',
...               fontScale=1.2)
```



CHAPTER

TWELVE

SCATTHIST

@author: u64053

scattHist.**scatterHist**(*df*, *xCol*, *yCol*, *graphTitle*, *save=True*, ***kwargs*)

Represents a center scatter plot of the kind "hex", "scatter" or "kde", and a top and right (border) histogram plots.

Parameters

df [pandas dataframe] Dataframe with the data to represent.

xCol [string] df column name to represent in the x axis.

yCol [string] df column name to represent in the y axis.

graphTitle [string] title of the graph and filename for saving.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [seaborn JoinGrid object.] object that stores the generated graph.

Other Parameters

cKind [string, {"hex", "kde", "scatter"}, optional, default "scatter"] Kind of plot to use for the center plot. The options are "hex" for a hexagonal grid plot, "kde" for a kernel density estimate plot (or level surfaces) and "scatter" for the typical scatter plot.

xLabel [string, optional] Label for the x axis. If not provided xCol is used.

yLabel [string, optional] Label for the y axis. If not provided yCol is used.

size [float, optional, default 7] Size of the figure in inches.

centerPropor [float, optional, default 5] Relative size of the center to the margins.

plotSeparation [float, optional, default 0.1] Separation between the border plots and the center plot.

cColor [string, optional] hex value or named color to use in the center plot.

cKdeFill [boolean, optional, default True] If true it fills the center plot kde surfaces, if False, draw only the borders. Only if cKind is "kde"

cHexGrid [integer, optional, default 20.] Number of hexagonal grids along a axis for the center plot. Only if cKind is "hex".

cHexGridColor [string, optional, default "w"] hex value or named color to use in the center plot for the hex grid borders. Only if cKind is "hex"

bColor [string, optional] hex value or named color to use in the border plots.

bBins [integer, optional, default 20] Number of bins (columns) of the border plots.

bRug [boolean, optional, default True] It switches on and off the rug representation in the border plot.

bKde [boolean, optional, default True] It switches on and off the kde line representation in the border plots.

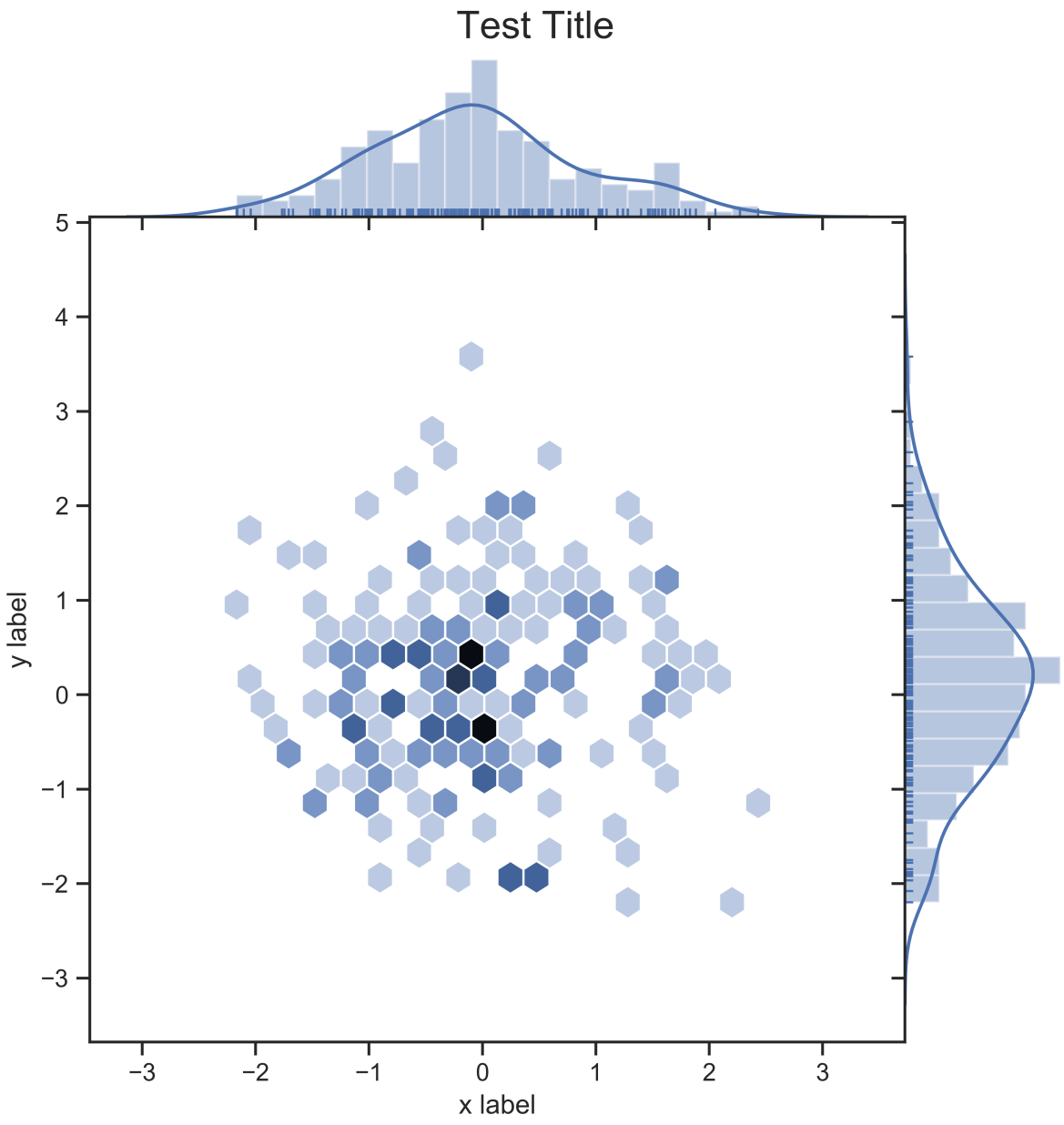
bHist [boolean, optional, default True] It switches on and off the histogram representation in the border plots.

palette [seaborn compatible palette, optional] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] If not provided graphTitle will be used. Non alphanumeric values will be deleted.

Examples

```
>>> rs = np.random.RandomState()
>>> x = rs.normal(size=200)
>>> y = rs.normal(size=200)
>>> df =pd.DataFrame({"X_coord": x, "Y_coord": y})
>>> scatterHist(df, xCol="X_coord", yCol="Y_coord",
...             graphTitle="Test Title", save=True,
...             cKind="hex", fileName="scatt Hist Test",
...             xLabel="x label", yLabel="y label")
```


Test Title
y label
x label
5
4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

CHAPTER

THIRTEEN

STACKEDBARS

@author: u64053

stackedBars.**stackedbars**(*dflist*, *repreCols*, *groupCol*, *graphTitle*, *save=True*, ***kwargs*)

Function to represent stacked bar graph normaliced to 100%. Data from multiple datasets with the same columns. A column will be used to aggregate the rows in multiple groups.

Parameters

dflist [list of pandas DataFrames] list of dataframes containing the data, must all have the same columns.

repreCols [list of strings] list of string with the dataframe columns to represent.

groupCol [string] column of the dataframe to use

graphTitle [string] title of the graph and filename for saving.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

yLabel [string, optional.] label for the y axis.

yTicksNum [integer, optional] Number of ticks in the y axis.

width [float, optional.] width betwwen 0 and 1 of the bars.

colorDict [dict, optional.] dictionary with label:color being labels, groupCol column elements.

legendLoc [string, optional, default "best"] Legend location, accepts standart legend locations for matplotlib, plus "out top right","out center right", "out lower right" for placing the legend out of the plotting area and "none" for hidding the legend.

inverse_order [boolean, optional.] inverse the order of the bars.

xSize [float, optional.] x size of the figure in inches.

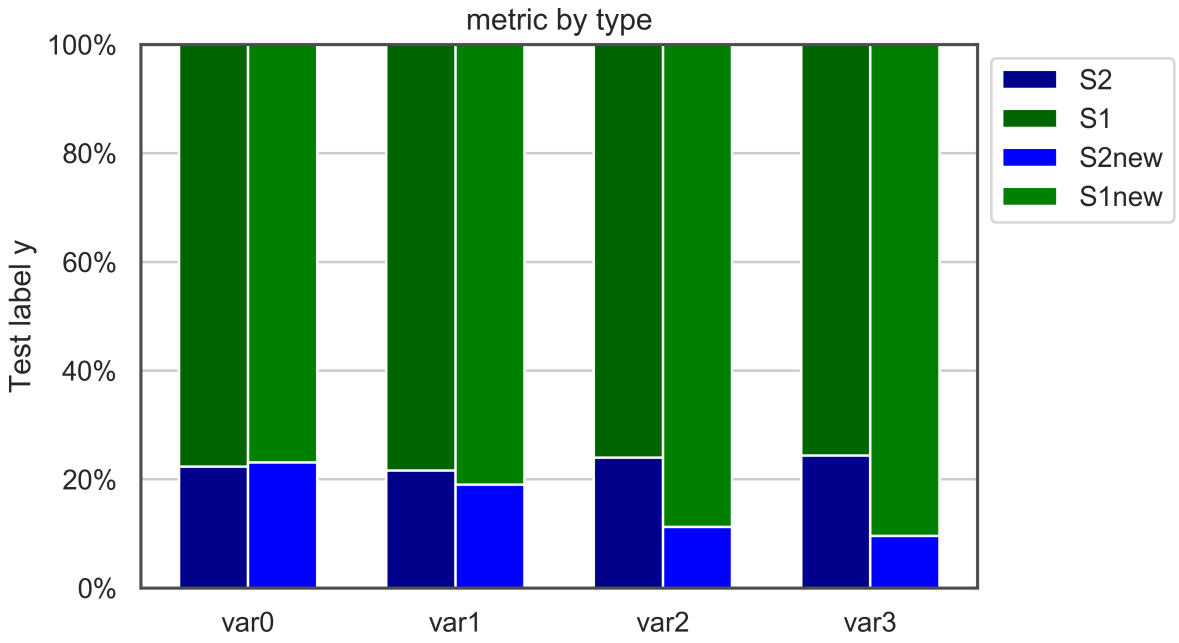
ySize [float, optional.] y size of the figure in inches.

palette [seaborn compatible palette,] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] File name to save, if not provided graphHeader will be used. Non alphanumeric values will be deleted.

Examples

```
>>> df_test1 = pd.DataFrame.from_dict(
...              {"type":["S1", "S2", "S1", "S2", "S1", "S1", "S1", "S1"],
...               "var0":[1.2, 1.1, 1.3, 1, 1.2, 1.1, 1.2, 1.3],
...               "var1":[4, 5, 8, 3, 5, 2, 7, 3],
...               "var2":[0.13, 0.12, 0.13,0.12, 0.12, 0.13, 0.13, 0.12],
...               "var3":[0.16, 0.15, 0.16, 0.15, 0.15, 0.14, 0.16, 0.16]})
>>> df_test2 = pd.DataFrame.from_dict(
...              {"type":["S1new", "S2new", "S1new", "S2new",
...                       "S1new", "S1new", "S1new", "S1new"],
...               "var0":[2.2, 1.1, 1.3, 2, 1.2, 2.1, 1.2, 2.3],
...               "var1":[4, 5, 6, 3, 5, 6, 7, 6],
...               "var2":[0.13, 0.16, 0.63,0.12, 0.12, 0.6, 0.13, 0.6],
...               "var3":[1.16, 0.16, 0.16, 0.15, 0.15, 1.14, 0.16, 0.16]})
>>> dflist = [df_test1, df_test2]
>>> stackedbars(dflist,
...             repreCols=['var0', 'var1', 'var2', 'var3'],
...             groupCol="type",
...             graphTitle="metric by type",
...             save = True,
...             normalize=True,
...             yLabel= "Test label y",
...             colorDict = {"S1":"darkgreen", "S1new": "green",
...                          "S2":"darkblue", "S2new": "blue"},
...             inverse_order=False,
...             legendLoc="out top right",
...             fileName="stacked Bars Example")
```



CHAPTER

FOURTEEN

STANDARTCOLORPALETTE

@author: u64053

standartColorPalette.**stdColor**(*colorCod*, *series=False*, *color=False*, *reverse=False*)

Function to generate a color or color palette from the standart TEASP color database using the color code chosen.

If a color and a series is selected the color code is given. If only a series is selected, all the colors in that serie are given in order: [“INDIGO”, “ORANGE”, “GREEN”, “RED”, “PURPLE”, “TEAL”, “PINK”, “GRAY”, “YELLOW”, “BLUE”]

If only a color is selected, a monocrome palette is given, from light to dark.

Parameters

colorCod [string.] Color code to use, valid inputs “hex”,”dec”,”rgb”.

series [integer, optional.] from lighter to darker, color brightness. Valid inputs: [100, 200, 300, 400, 500, 600, 700, 800, 900]

color [string, optional.] Name of the color to select. Valid values [“INDIGO”, “ORANGE”, “GREEN”, “RED”, “PURPLE”, “TEAL”, “PINK”, “GRAY”, “YELLOW”, “BLUE”]

reverse [boolean, optional.] If True reverse the order of the color palette.

Returns

colorOut [string or list of strings.] Color or list with the colors in the desired format.

Examples

Create a list of hex or RGB values of the color series 500:

```
>>> listHex = stdColor('hex', series=500, reverse=False)
>>> listRGB = stdColor('rgb', series=500, reverse=False)
```

Create a list of decimal values of shades of Blue:

```
>>> stdColor('dec', color="BLUE", reverse=False)
```

Obtain the RGB color code of the Blue of the series 500:

```
>>> blueRGB = stdColor('rgb', color="BLUE", series=500)
```

CHAPTER

FIFTEEN

VERTICALBARPLOT

@author: u64053

verticalBarPlot.**vBarPlot**(*df*, *xCol*, *barCols*, *yLabel*, *graphTitle*, *save=True*, ***kwargs*)

It represents a vertical appilated bar graph with legend, and a table at the bottom

Parameters

df [pandas dataframe] Dataframe with the data to represent.

xCol [string] df column name to represent in the x axis.

barCols [list of strings] df columns names to represent in the y axis as bars.

yLabel [string] Label for the y axis.

graphTitle [string] Title of the graph. Will be used as a filename if fileName is not provided.

save [boolean, optional] If True, the figure is saved into a file.

Returns

fig [matplotlib figure object.] figure that stores the generated graph.

Other Parameters

legendLoc [string, optional, default "best".] Legend location, accepts standart legend locations for matplotlib, plus "out top right","out center right", "out lower right" for placing the legend out of the plotting area and "none" for hidding the legend.

yLim [float, optional] Limit of the y axis in y data units.

tableCols [list of strings, optional, default False,] List of df cols headers to use as rows in the table under the plot. If not provided, the table will not be generated

tableEdge [string, optional, default "open",] Border style for the table valid values are: 'open', 'closed', 'horizontal', 'vertical'

width [float, optional] Width of the bars between 0 and 1.

xSize [float, optional default 10] Width of the figure in inches.

ySize [float, optional, default 6] Height in inches.

fontScale [float, optional, default 1.2] Global scale factor for the font size.

yTicksNum [float, optional] Number of ticks in the y axis.

orderBy [string, optional] *df* column header name to use to order x axis in ascending order.

hLines [list of lists, optional] Used to display horizontal lines, each list containing a 3 elements, one list for each line:

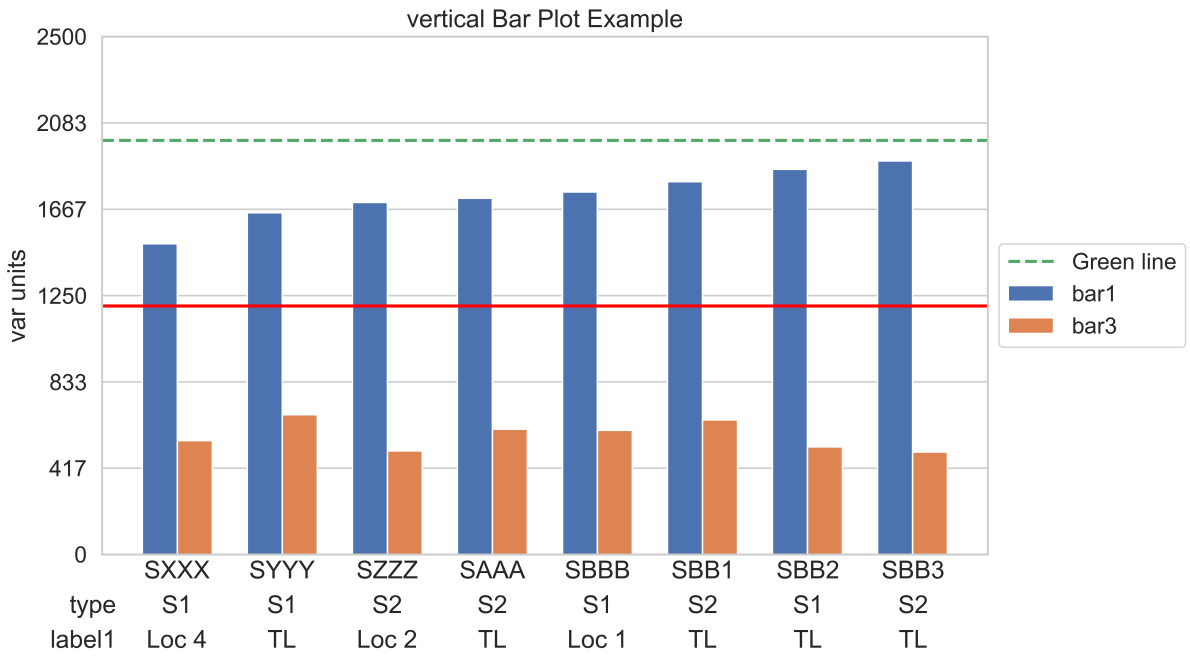
- first, the y cordinate for the horizontal line
- second, the color for the horizontal line
- third, linestyle to use for the line
- fourth, label of the line, to use in the legend (optional)

palette [seaborn compatible palette,] Can be the name of a seaborn palette, a seaborn palette or a list interpretable as a palette by seaborn.

fileName [string, optional] File name to save, if not provided graphTitle will be used. Non alphanumeric values will be deleted.

Examples

```
>>> data = {"AC": ["SXXX","SYYY","SZZZ","SAAA","SBBB","SBB1","SBB2","SBB3"],
...         "type": ["S1","S2","S2","S1","S1","S2","S2","S1"],
...         "bar1": [1750,1700,1800,1500,1860,1900,1720,1650],
...         "bar2": [1750,1700,1800,1500,1860,1900,1720,1650],
...         "bar3": [600,500,650,550,520,495,605,675],
...         "label1": ["Loc 1","Loc 2","TL","Loc 4","TL","TL","TL","TL"],
...         "label2": ["Loc 1","Loc 2","TL","Loc 4","TL","TL","TL","TL"]}
>>> df_test = pd.DataFrame.from_dict(data)
>>> vBarPlot(df=df_test,
...          xCol="AC",
...          barCols=["bar1", "bar3"],
...          yLabel="var units",
...          graphTitle="vertical Bar Plot Example",
...          legendLoc="out center right",
...          yLim= 2500,
...          tableCols=["type", "label1"],
...          tableEdge="open",
...          width=False,
...          xSize=10,
...          ySize=6,
...          fontScale=1.3,
...          yTicksNum=6,
...          orderBy="bar1",
...          hLines=[[1200, "red","-"], [2000, "g","--" ,"Green line"]]
...          )
```


vertical Bar Plot Example
var units
2500
2083
1667
1250
833
417
0
SXXX SYYY SZZZ SAAA SBBB SBB1 SBB2 SBB3
type S1 S1 S2 S2 S1 S2 S1 S2
label1 Loc 4 TL Loc 2 TL Loc 1 TL TL TL
Green line
bar1
bar3

CHAPTER

SIXTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX