chartify Documentation

Release 4.0.5

Chris Halpert

Oct 12, 2023

CONTENTS

(*What's new?*).

CHAPTER

ONE

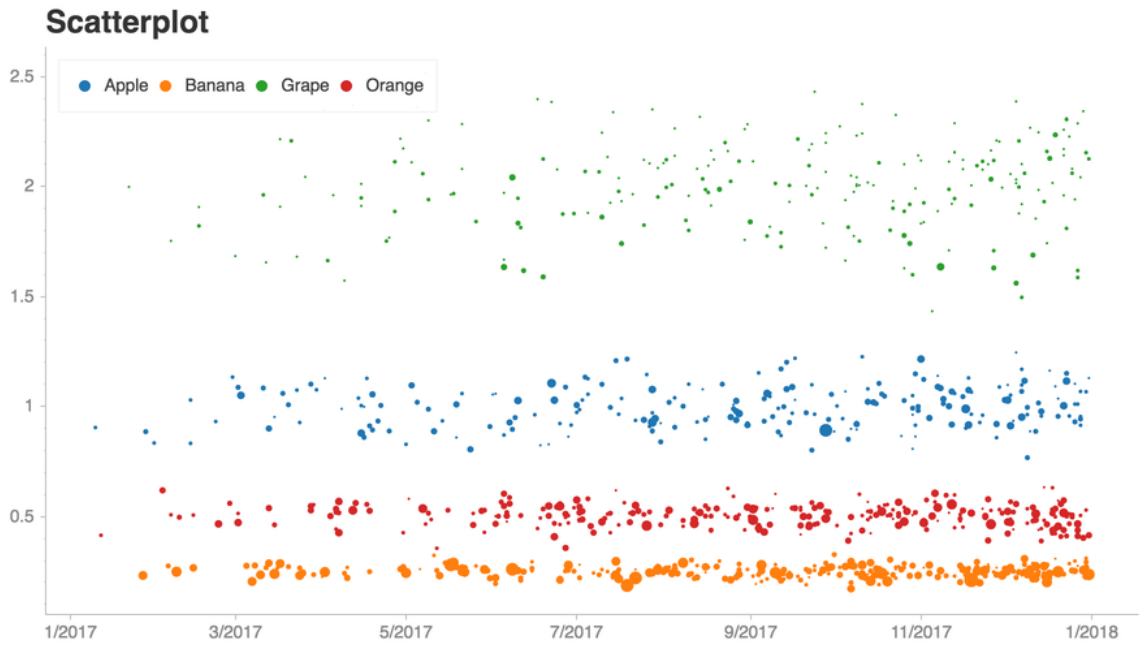
CHARTIFY

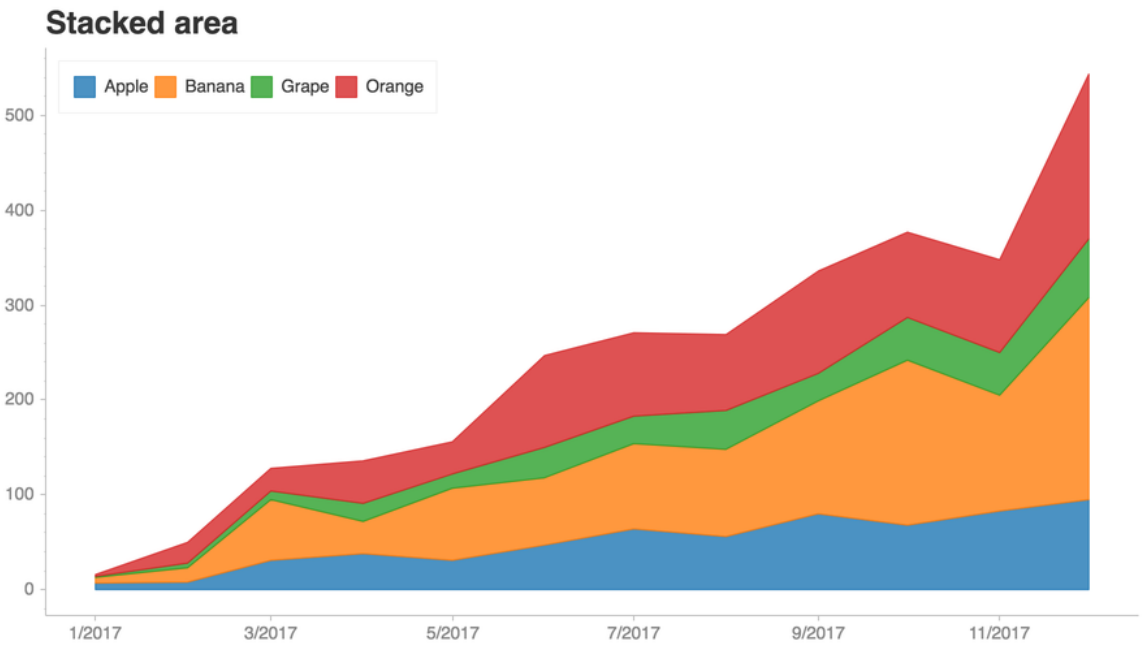
Chartify is a Python library that makes it easy for data scientists to create charts.

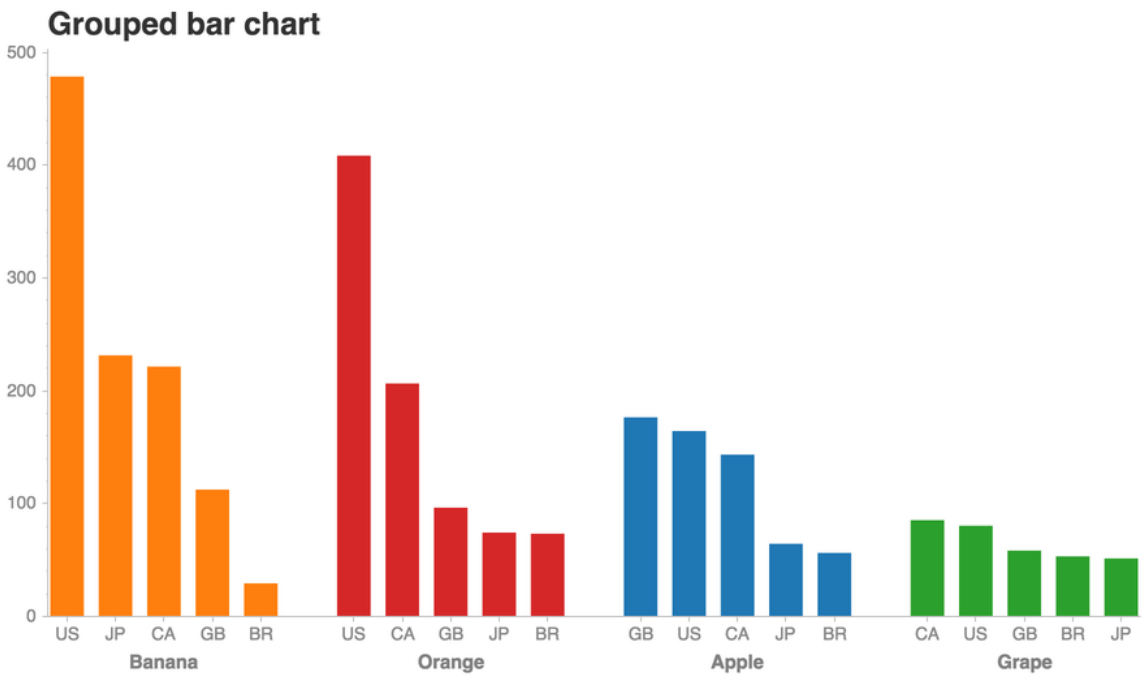
1.1 Why use Chartify?

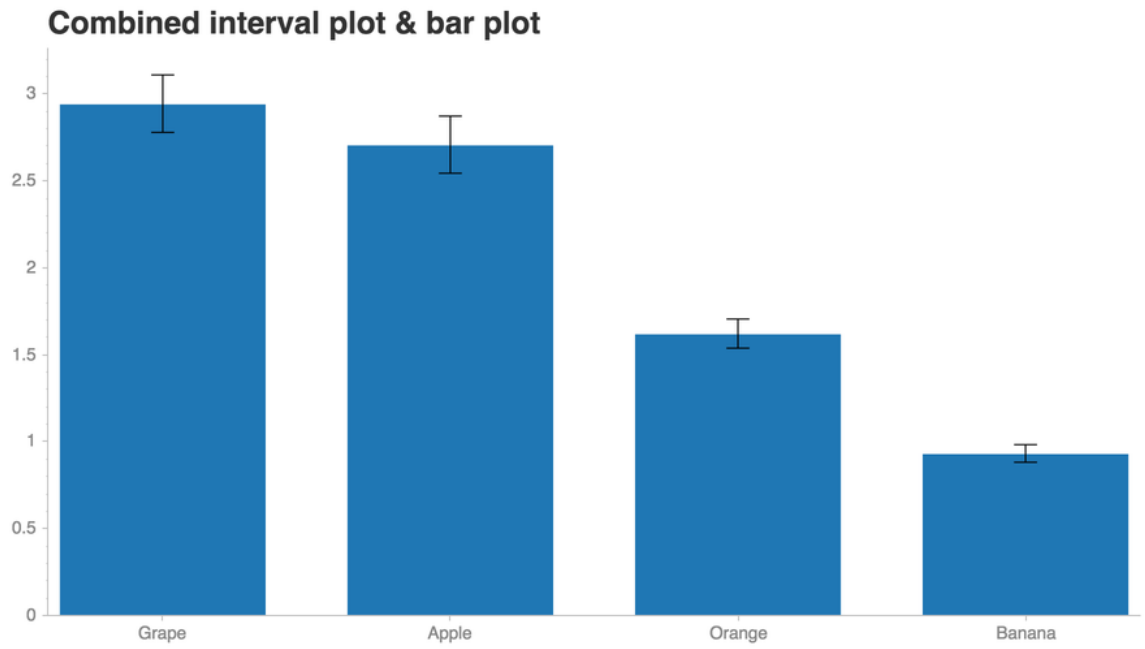
- Consistent input data format: Spend less time transforming data to get your charts to work. All plotting functions use a consistent tidy input data format.
- Smart default styles: Create pretty charts with very little customization required.
- Simple API: We've attempted to make the API as intuitive and easy to learn as possible.
- Flexibility: Chartify is built on top of Bokeh, so if you do need more control you can always fall back on Bokeh's API.

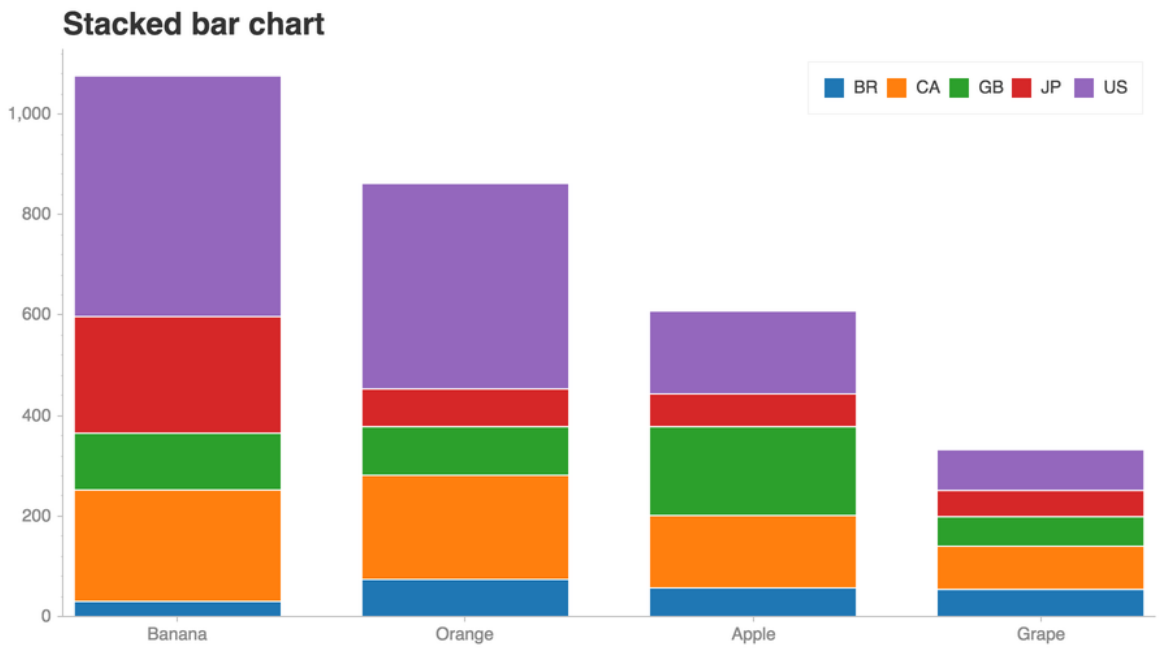
1.2 Examples

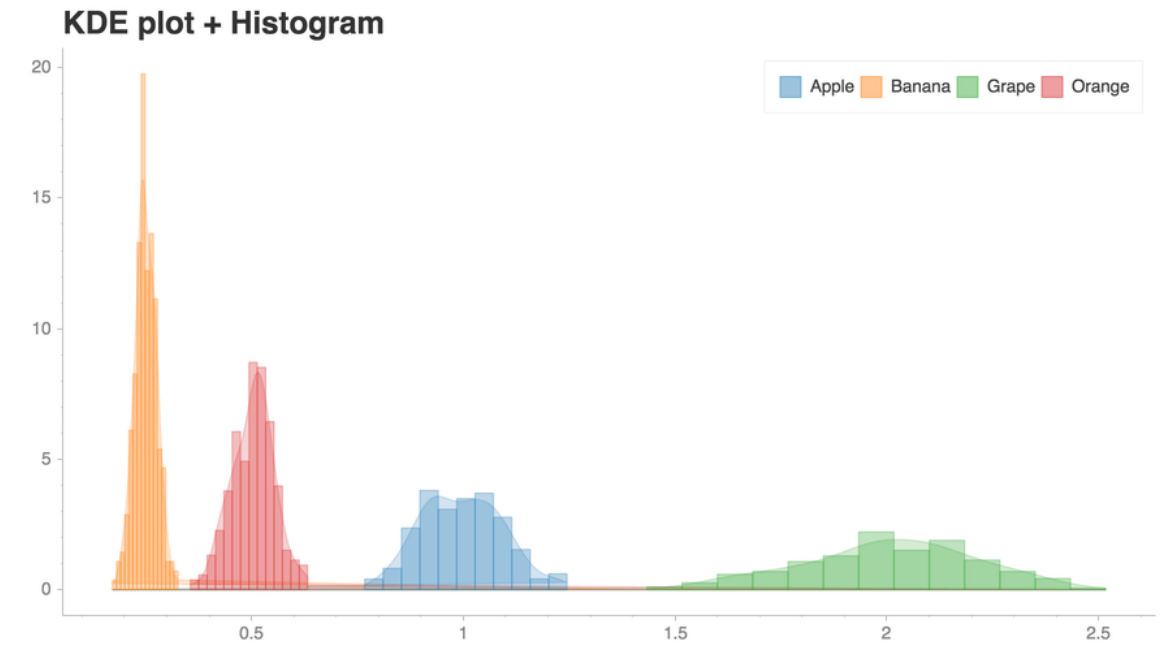

Scatterplot
Apple
Banana
Grape
Orange
2.5
2
1.5
1
0.5
1/2017
3/2017
5/2017
7/2017
9/2017
11/2017
1/2018


Stacked area
Apple
Banana
Grape
Orange
500
400
300
200
100
0
1/2017
3/2017
5/2017
7/2017
9/2017
11/2017


Grouped bar chart
500
400
300
200
100
0
US JP CA GB BR
Banana
US CA GB JP BR
Orange
GB US CA JP BR
Apple
CA US GB BR JP
Grape


Combined interval plot & bar plot
3
2.5
2
1.5
1
0.5
0
Grape
Apple
Orange
Banana





See this notebook for more examples!.

1.3 Installation

1. Chartify can be installed via pip:

```
pip3 install chartify
```

2. **Install chromedriver requirement (Optional. Needed for PNG output):**
    - Install google chrome.
    - Download the appropriate version of chromedriver for your OS here.
    - **Copy the executable file to a directory within your PATH.**
        - View directorys in your PATH variable: `echo $PATH`
        - Copy chromedriver to the appropriate directory, e.g.: `cp chromedriver /usr/local/bin`

1.4 Getting started

This tutorial notebook is the best place to get started with a guided tour of the core concepts of Chartify.

From there, check out the example notebook for a list of all the available plots.

1.5 Docs

Documentation available on chartify.readthedocs.io

1.6 Getting support

Join #chartify on spotify-foss.slack.com (Get an invite)

Use the chartify tag on StackOverflow.

1.7 Resources

- Data Visualization with Chartify

1.8 Code of Conduct

This project adheres to the Open Code of Conduct. By participating, you are expected to honor this code.

1.9 Contributing

See the contributing docs.

CHAPTER

TWO

CONTENTS:

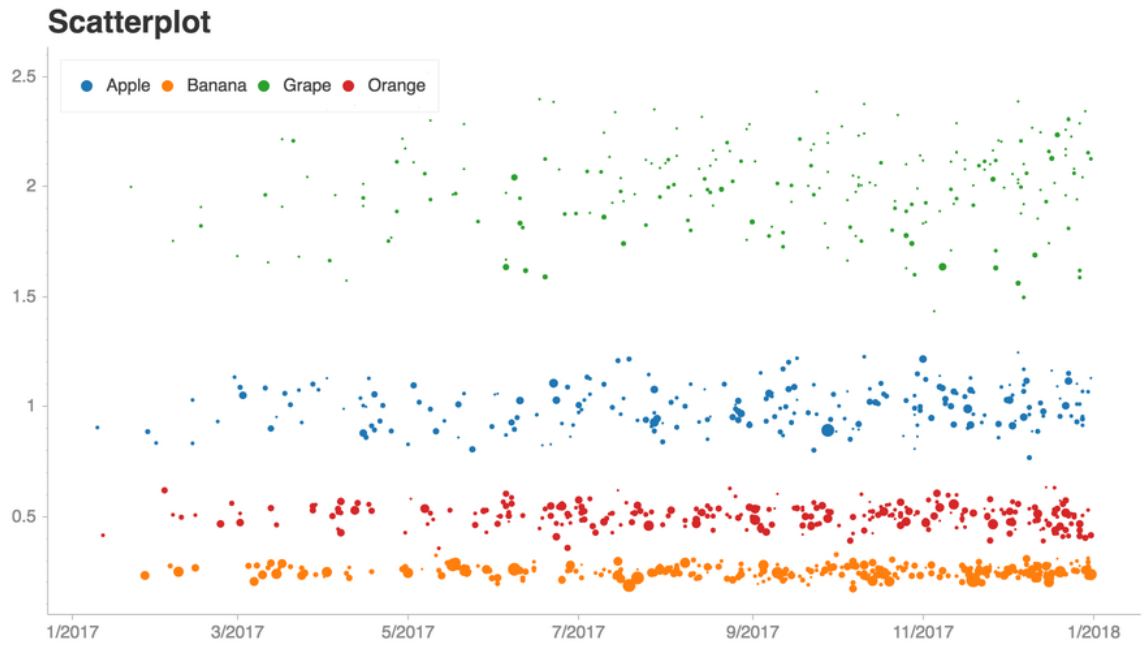
2.1 Chartify

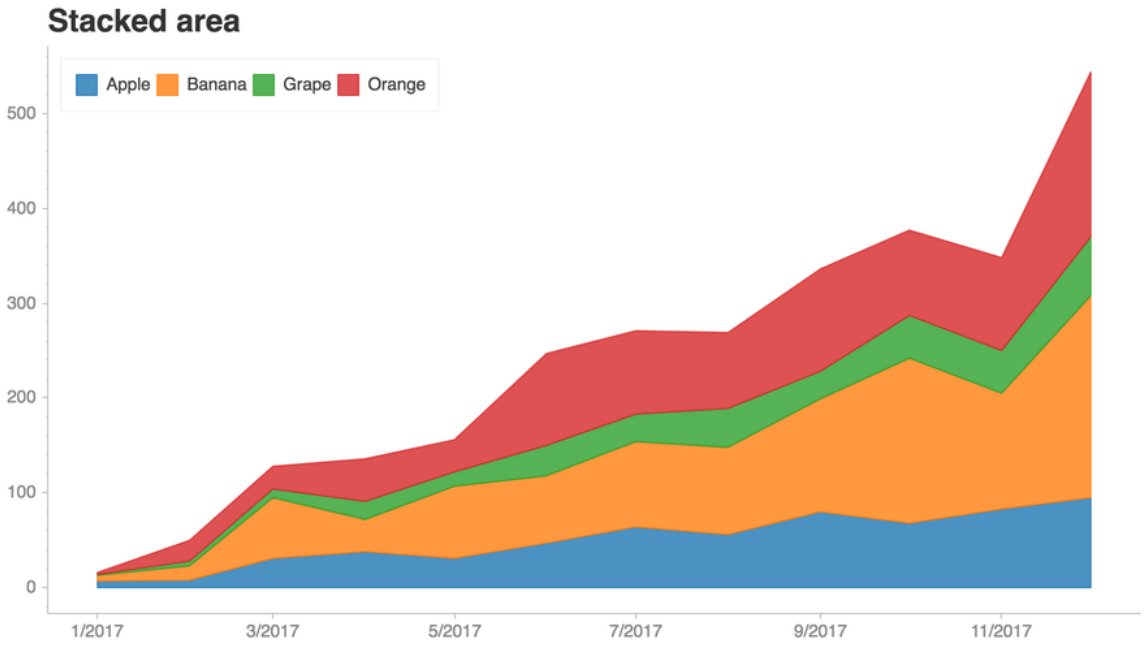
Chartify is a Python library that makes it easy for data scientists to create charts.

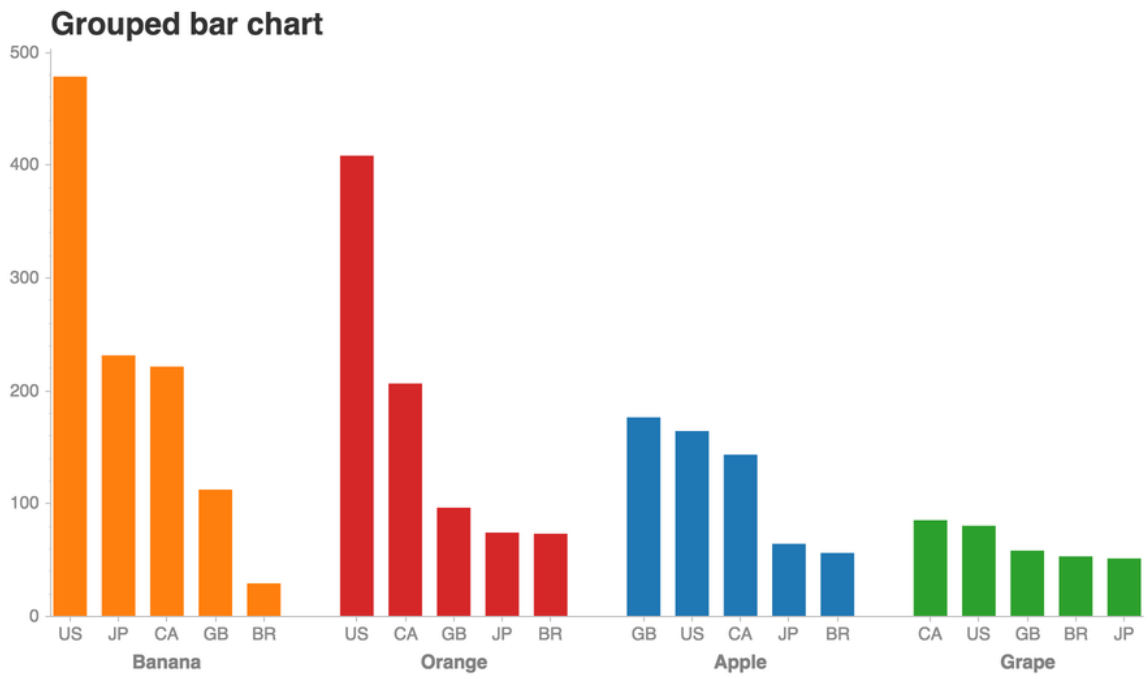
2.1.1 Why use Chartify?

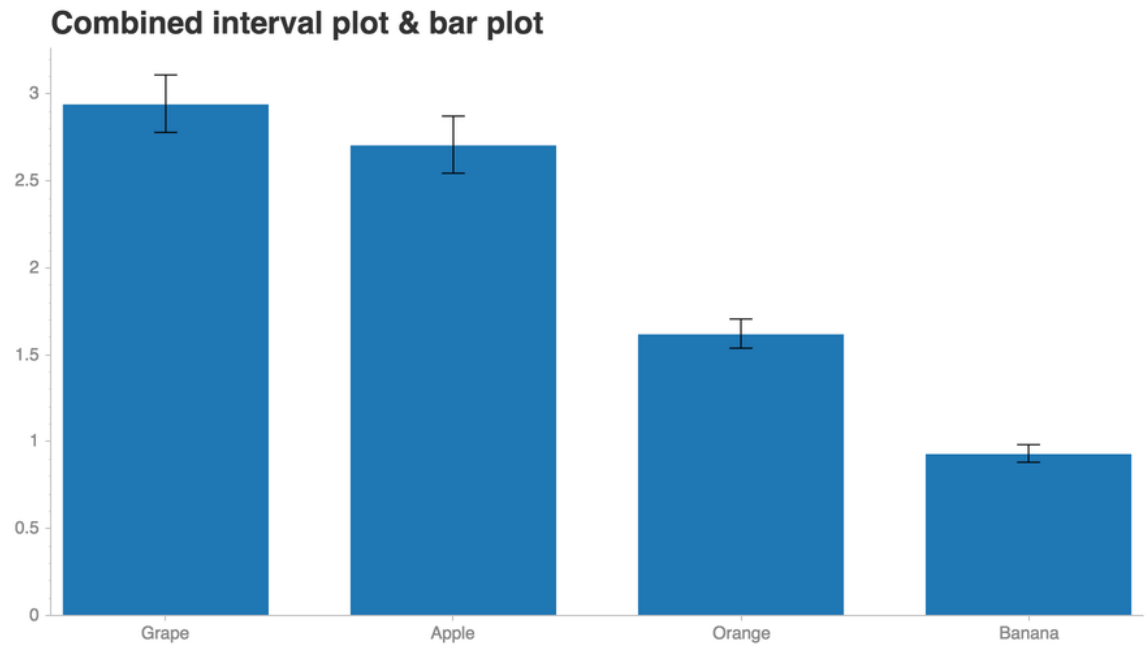
- Consistent input data format: Spend less time transforming data to get your charts to work. All plotting functions use a consistent tidy input data format.
- Smart default styles: Create pretty charts with very little customization required.
- Simple API: We've attempted to make the API as intuitive and easy to learn as possible.
- Flexibility: Chartify is built on top of Bokeh, so if you do need more control you can always fall back on Bokeh's API.

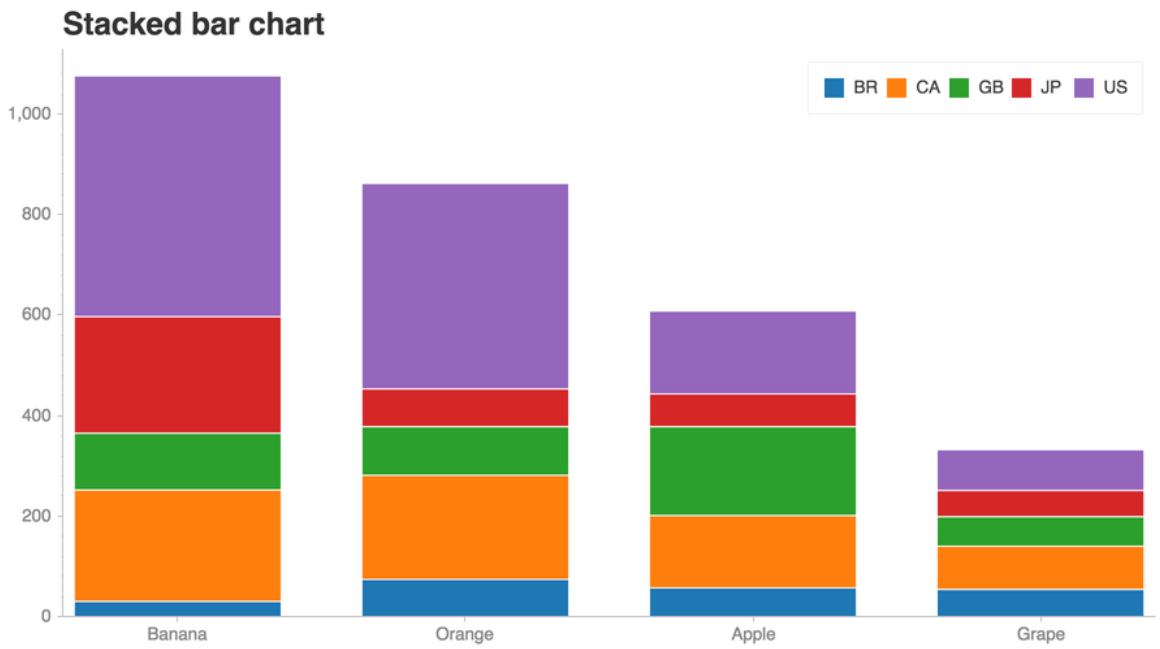
2.1.2 Examples

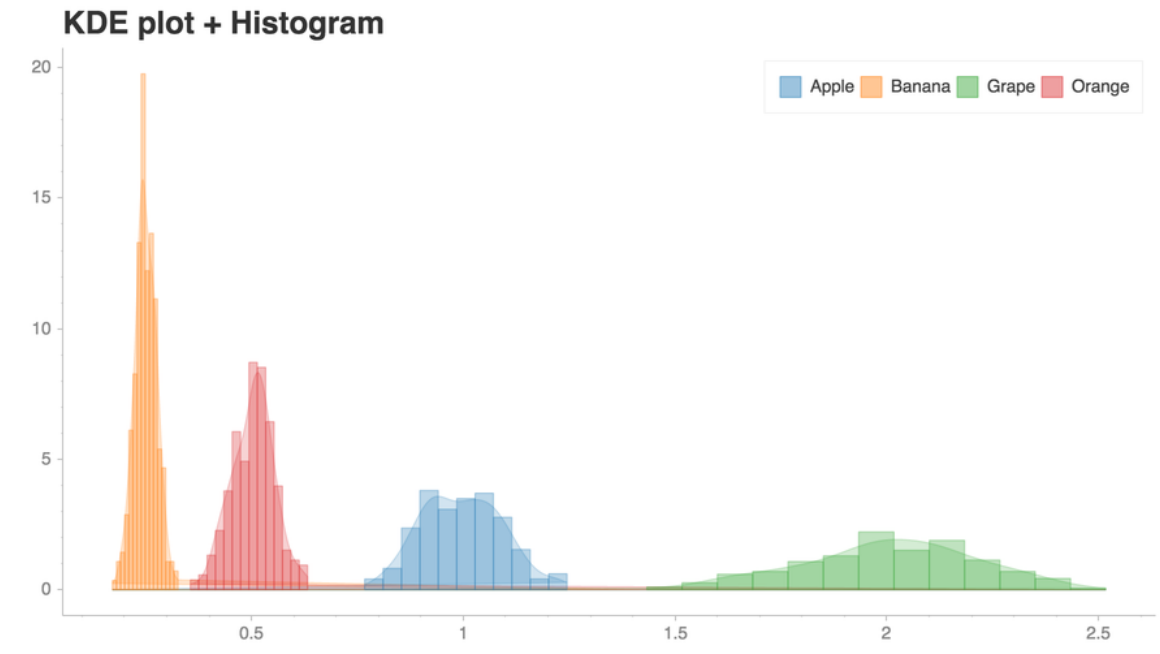





Grouped bar chart
500
400
300
200
100
0
US JP CA GB BR
Banana
US CA GB JP BR
Orange
GB US CA JP BR
Apple
CA US GB BR JP
Grape


Combined interval plot & bar plot
3
2.5
2
1.5
1
0.5
0
Grape
Apple
Orange
Banana





See this notebook for more examples!.

2.1.3 Installation

1. Chartify can be installed via pip:

```
pip3 install chartify
```

2. **Install chromedriver requirement (Optional. Needed for PNG output):**
    - Install google chrome.
    - Download the appropriate version of chromedriver for your OS here.
    - **Copy the executable file to a directory within your PATH.**
        - View directorys in your PATH variable: `echo $PATH`
        - Copy chromedriver to the appropriate directory, e.g.: `cp chromedriver /usr/local/bin`

2.1.4 Getting started

This tutorial notebook is the best place to get started with a guided tour of the core concepts of Chartify.

From there, check out the example notebook for a list of all the available plots.

2.1.5 Docs

Documentation available on chartify.readthedocs.io

2.1.6 Getting support

Join #chartify on spotify-foss.slack.com (Get an invite)

Use the chartify tag on StackOverflow.

2.1.7 Resources

- Data Visualization with Chartify

2.1.8 Code of Conduct

This project adheres to the Open Code of Conduct. By participating, you are expected to honor this code.

2.1.9 Contributing

See the contributing docs.

2.2 Installation

2.2.1 Stable release

To install chartify, run this command in your terminal:

```
$ pip install chartify
```

This is the preferred method to install chartify, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2.2 From sources

The sources for chartify can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/spotify/chartify
```

Or download the tarball:

```
$ curl  -OL https://github.com/spotify/chartify/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

2.3 Usage

To use chartify in a project:

```
import chartify
```

2.4 Chart

class chartify.**Chart**(*blank_labels=False*, *layout='slide_100%'*, *x_axis_type='linear'*, *y_axis_type='linear'*, *second_y_axis=False*)

Class Docstring

- Styling (.style)
- Plotting (.plot)
- Callouts (.callout)
- Axes (.axes)
- Bokeh figure (.figure)

property data

Return a list of dictionaries of the data that have be plotted on the chart.

Note:
The format will depend on the types of plots that have been added.

property legend_location

str: Legend location.

save(*filename*, *format='html'*)

Save the chart.

Args:
filename (str): Name of output file. format (str):

- **'html': Output chart as HTML.**
  Renders faster and allows for interactivity. Charts saved as HTML in a Jupyter notebook WILL NOT display on Github. Logos will not display on HTML charts. Recommended when drafting plots.
- **'png': Output chart as PNG.**
  Easy to paste into google slides. Recommended when the plot is in a finished state. Will render logos.
- 'svg': Output as SVG.

set_legend_location(*location*, *orientation='horizontal'*)

Set the legend location.

Args:
location (str or tuple): Legend location. One of: - Outside of the chart: 'outside_top', 'outside_bottom',

'outside_right'

- **Within the chart area: 'top_left', 'top_center',**
  'top_right', 'center_left', 'center', 'center_right', 'bottom_left', 'bottom_center', 'bottom_right'
- Coordinates: Tuple(Float, Float)
- None: Removes the legend.

orientation (str): 'horizontal' or 'vertical'

Returns:
Current chart object

set_source_label(*source*)

Set the chart data source.

Args:
source (str): Data source.

Returns:
Current chart object

set_subtitle(*subtitle*)

Set the chart subtitle.

Args:
subtitle (str): Subtitle text.

Note:
Set value to "" to remove subtitle.

Returns:
Current chart object

`set_title`(*title*)

Set the chart title.

Args:
title (str): Title text.

Returns:
Current chart object

`show`(*format='html'*)

Show the chart.

Args:

format (str):

- **'html': Output chart as HTML.**
  Renders faster and allows for interactivity. Charts saved as HTML in a Jupyter notebooks WILL NOT display on Github. Logos will not display on HTML charts. Recommended when drafting plots.
- **'png': Output chart as PNG.**
  Easy to copy+paste into slides. Will render logos. Recommended when the plot is in a finished state.
- 'svg': Output as SVG.

`property source_text`

str: Data source of the chart.

`property subtitle`

str: Subtitle text of the chart.

`property title`

str: Title text of the chart.

2.5 Plot

`class` chartify._core.plot.`PlotCategoricalXY`(*chart*, *y_range_name='default'*)

Plot functions for categorical x & y axes:

Methods:

- heatmap

`heatmap`(*data_frame*, *x_column*, *y_column*, *color_column*, *text_column=None*, *color_palette='RdBu'*, *reverse_color_order=False*, *text_color='white'*, *text_format='{:,.2f}'*, *color_value_min=None*, *color_value_max=None*, *color_value_range=100*)

Heatmap.

Args:
data_frame (pandas.DataFrame): Data source for the plot. x_column (str): Column name to plot on

the x axis. y_column (str): Column name to plot on the y axis. color_column (str): Column name of numerical type to plot on

the color dimension.

text_column (str or None): Column name of the text labels. color_palette (str, chartify.ColorPalette): Color palette to

apply to the heatmap. See chartify.color_palettes.show() for available color palettes.

reverse_color_order (bool): Reverse order of the color palette. text_color (str): Color name or hex value.

See chartify.color_palettes.show() for available color names.

text_format: Python string formatting to apply to the text labels. color_value_min (float): Minimum value for the color palette.

If None, will default to the min value of the color_column dimension.

color_value_max (float): Maximum value for the color palette.
If None, will default to the max value of the color_column dimension.

color_value_range (int): The size of the range of colors in
the color palette. A larger color range will result in greater variation among the cell colors.

`class` chartify._core.plot.**PlotNumericXY**(*chart*, *y_range_name='default'*)

Plot functions for numeric x & y axes:

Methods:

- line
- scatter
- text
- area

area(*data_frame*, *x_column*, *y_column*, *second_y_column=None*, *color_column=None*, *color_order=None*, *stacked=False*)

Area plot.

Note:

- **When a single y_column is passed: Shade area between the**
  y_values and zero.
- Use *stacked* argument for stacked areas.
- **When both y_column and second_y_column are passed:**
  Shade area between the two y_columns.

Args:

data_frame (pandas.DataFrame): Data source for the plot. x_column (str): Column name to plot on the x axis. y_column (str): Column name to plot on the y axis. second_y_column (str, optional): Column name to plot on

the y axis.

color_column (str, optional): Column name to group by on
the color dimension.

color_order (list, optional): List of values within the
'color_column' for specific sorting of the colors.

stacked (bool, optional): Stacked the areas.
Only applicable with a single y_column. Default: False.

`line`(*data_frame*, *x_column*, *y_column*, *color_column=None*, *color_order=None*, *line_dash='solid'*, *line_width=4*, *alpha=1.0*)

Line Chart.

Note:
This method will not automatically sort the x-axis. Try sorting the axis if the line graph looks strange.

Args:
data_frame (pandas.DataFrame): Data source for the plot. x_column (str): Column name to plot on the x axis. y_column (str): Column name to plot on the y axis. color_column (str, optional): Column name to group by on

the color dimension.

color_order (list, optional): List of values within the
'color_column' for specific sorting of the colors.

line_dash (str, optional): Dash style for the line. One of:

- 'solid'
- 'dashed'
- 'dotted'
- 'dotdash'
- 'dashdot'

line_width (int, optional): Width of the line alpha (float): Alpha value.

`scatter`(*data_frame*, *x_column*, *y_column*, *size_column=None*, *color_column=None*, *color_order=None*, *alpha=1.0*, *marker='circle'*)

Scatter plot.

Args:
data_frame (pandas.DataFrame): Data source for the plot. x_column (str): Column name to plot on the x axis. y_column (str): Column name to plot on the y axis. size_column (str, optional): Column name of numerical values

to plot on the size dimension.

color_column (str, optional): Column name to group by on
the color dimension.

color_order (list, optional): List of values within the
'color_column' for specific sorting of the colors.

alpha (float): Alpha value. marker (str): marker type. Valid types:

'asterisk', 'circle', 'circle_cross', 'circle_x', 'cross', 'diamond', 'diamond_cross', 'hex', 'inverted_triangle', 'square', 'square_x', 'square_cross', 'triangle', 'x', '*', '+', 'o', 'ox', 'o+'

text(*data_frame*, *x_column*, *y_column*, *text_column*, *color_column=None*, *color_order=None*, *font_size='1em'*, *x_offset=0*, *y_offset=0*, *angle=0*, *text_color=None*)

Text plot.

Args:

data_frame (pandas.DataFrame): Data source for the plot. x_column (str): Column name to plot on the x axis. y_column (str): Column name to plot on the y axis. text_column (str): Column name to plot as text labels. color_column (str, optional): Column name to group by on the

color dimension.

color_order (list, optional): List of values within the
'color_column' for specific sorting of the colors.

font_size (str, optional): Size of text. x_offset (int, optional): # of pixels for horizontal text offset.

Can be negative. Default: 0.

y_offset (int, optional): # of pixels for vertical text offset.
Can be negative. Default: 0.

angle (int): Degrees from horizontal for text rotation. text_color (str): Color name or hex value.

See chartify.color_palettes.show() for available color names. If omitted, will default to the next color in the current color palette.

class chartify._core.plot.**PlotNumericDensityXY**(*chart*, *y_range_name='default'*)

Plot functions for single density:

Methods:

- histogram
- kde

histogram(*data_frame*, *values_column*, *color_column=None*, *color_order=None*, *method='count'*, *bins='auto'*)

Histogram.

Args:

data_frame (pandas.DataFrame): Data source for the plot. values_column (str): Column of numeric values. color_column (str, optional): Column name to group by on

the color dimension.

color_order (list, optional): List of values within the
'color_column' for specific sorting of the colors.

method (str, optional): - 'count': Result will contain the number of samples at each bin. - 'density': Result is the value of the probability density

function at each bin. The PDF is normalized so that the integral over the range is 1.

- **'mass': Result is the value of the probability mass**
function at each bin. The PMF is normalized so that the value is equivalent to the sample count at each bin divided by the total count.

bins (int or sequence of scalars or str, optional):
If bins is an int, it defines the number of equal-width bins in the given range. If bins is a sequence, it defines the bin edges, including the rightmost edge, allowing for non-uniform bin widths. See numpy.histogram documentation for more details.

- **'auto':**
  Maximum of the 'sturges' and 'fd' estimators. Provides good all around performance.
- **'fd' (Freedman Diaconis Estimator)**
  Robust (resilient to outliers) estimator that takes into account data variability and data size.
- **'doane'**
  An improved version of Sturges' estimator that works better with non-normal datasets.
- **'scott'**
  Less robust estimator that that takes into account data variability and data size.
- **'rice'**
  Estimator does not take variability into account, only data size. Commonly overestimates number of bins required.
- **'sturges'**
  R's default method, only accounts for data size. Only optimal for gaussian data and underestimates number of bins for large non-gaussian datasets.
- **'sqrt'**
  Square root (of data size) estimator, used by Excel and other programs for its speed and simplicity.

kde(*data_frame*, *values_column*, *color_column=None*, *color_order=None*)

Kernel Density Estimate Plot.

Args:
data_frame (pandas.DataFrame): Data source for the plot. values_column (str): Column of numeric values. color_column (str, optional): Column name to group by on

the color dimension.

color_order (list, optional): List of values within the
'color_column' for specific sorting of the colors.

class chartify._core.plot.**PlotDensityXY**(*chart*, *y_range_name='default'*)

Plot functions for denxity X & Y:

Methods:

- hexbin

hexbin(*data_frame*, *x_values_column*, *y_values_column*, *size*, *color_palette='Blues'*, *reverse_color_order=False*, *orientation='pointytop'*, *color_value_range=10*)

Hexbin.

Args:
data_frame (pandas.DataFrame): Data source for the plot. x_values_column (str): Column of numeric values to bin into tiles. y_values_column (str): Column of numeric values to bin into tiles. size (float): Bin size for the tiles. color_palette (str, chartify.ColorPalette): Color palette to

apply to the tiles. See chartify.color_palettes.show() for available color palettes.

reverse_color_order (bool): Reverse order of the color palette. orientation (str): "pointytop" or "flat-top". Whether the hexagonal

tiles should be oriented with a pointed corner on top, or a flat side on top.

color_value_range (int): The size of the range of colors in
the color palette. A larger color range will result in greater variation among the cell colors.

class chartify._core.plot.**PlotMixedTypeXY**(*chart*, *y_range_name='default'*)

Plot functions for mixed type x & y axes:

Methods:

- bar
- bar_stacked
- lollipop
- parallel

bar(*data_frame*, *categorical_columns*, *numeric_column*, *color_column=None*, *color_order=None*, *categorical_order_by='values'*, *categorical_order_ascending=False*)

Bar chart.

Note:
To change the orientation set x_axis_type or y_axis_type argument of the Chart object.

Args:
data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

numeric_column (str): Column name to plot on the numerical axis. color_column (str, optional): Column name to group by on

the color dimension.

color_order (list, optional): List of values within the
'color_column' for specific color sort.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'values'. - 'values': Order categorical axis by the numerical

axis values. Default.

- 'labels': Order categorical axis by the categorical labels.
- **array-like object (list, tuple, np.array): New labels**
to conform the categorical axis to.

categorical_order_ascending (bool, optional): Sort order of the
categorical axis. Default False.

bar_stacked(*data_frame*, *categorical_columns*, *numeric_column*, *stack_column*, *normalize=False*, *stack_order=None*, *categorical_order_by='values'*, *categorical_order_ascending=False*)

Plot stacked bar chart.

Note:

- To change the orientation set x_axis_type or y_axis_type

argument of the Chart object. - Stacked numeric values must be all positive or all negative. To plot both positive and negative values on the same chart call this method twice. Once for the positive values and once for the negative values.

Args:

data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

numeric_column (str): Column name to plot on the numerical axis. stack_column (str): Column name to group by on the stack dimension. normalize (bool, optional): Normalize numeric dimension for

100% stacked bars. Default False.

stack_order (list, optional): List of values within the
'stack_column' dimension for specific stack sort.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'values'. - 'values': Order categorical axis by the numerical

axis values. Default.

- 'labels': Order categorical axis by the categorical labels.
- **array-like object (list, tuple, np.array): New labels**
to conform the categorical axis to.

categorical_order_ascending (bool, optional): Sort order
of the categorical axis. Default False.

`boxplot`(*data_frame*, *categorical_columns*, *numeric_column*, *color_column=None*, *color_order=None*, *categorical_order_by='labels'*, *categorical_order_ascending=True*, *outlier_marker='circle'*, *outlier_color='black'*, *outlier_alpha=0.3*, *outlier_size=15*)

Box-and-whisker plot.

Note:
To change the orientation set x_axis_type or y_axis_type argument of the Chart object.

Args:

data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

numeric_column (str): Column name to plot on the numerical axis. color_column (str, optional): Column name to group by on

the color dimension.

color_order (list, optional):

List of values within the 'color_column' for
specific color sort.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'labels'. - 'labels': Order categorical axis by the categorical labels. - array-like object (list, tuple, np.array): New labels

to conform the categorical axis to.

categorical_order_ascending (bool, optional):
Sort order of the categorical axis. Default True.

outlier_marker (str, optional): Outlier marker type. Valid types:
'asterisk', 'circle', 'circle_cross', 'circle_x', 'cross', 'diamond', 'diamond_cross', 'hex', 'inverted_triangle', 'square', 'square_x', 'square_cross', 'triangle', 'x', '*', '+', 'o', 'ox', 'o+' Default 'circle'

outlier_color (str, optional): Color name or hex value.
See chartify.color_palettes.show() for available color names. Default 'black'

outlier_alpha (float, optional): Alpha value. Default 0.3 outlier_size (float, optional): Size of outlier markers.

Default 15

`interval`(*data_frame*, *categorical_columns*, *lower_bound_column*, *upper_bound_column*, *middle_column=None*, *categorical_order_by='values'*, *categorical_order_ascending=False*, *color='black'*)

Interval.

Args:
data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

lower_bound_column (str): Column name to plot on the
numerical axis for the lower bound.

upper_bound_column (str): Column name to plot on the
numerical axis for the upper bound.

middle_column (str, optional): Column name to plot on the
numerical axis for the middle tick.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'values'. - 'values': Order categorical axis by the numerical

axis values. Default.

- 'labels': Order categorical axis by the categorical labels.
- **array-like object (list, tuple, np.array): New labels**
  to conform the categorical axis to.

categorical_order_ascending (bool, optional): Sort order of the
categorical axis. Default False.

color (str): Color name or hex value.
See chartify.color_palettes.show() for available color names.

`lollipop`(*data_frame*, *categorical_columns*, *numeric_column*, *color_column=None*, *color_order=None*, *categorical_order_by='values'*, *categorical_order_ascending=False*)

Lollipop chart.

Note:
To change the orientation set x_axis_type or y_axis_type argument of the Chart object.

Args:
data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

numeric_column (str): Column name to plot on the numerical axis. color_column (str, optional): Column name to group by on

the color dimension.

color_order (list, optional):

List of values within the 'color_column' for
specific color sort.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'values'. - 'values': Order categorical axis by the numerical axis values. - 'labels': Order categorical axis by the categorical labels. - array-like object (list, tuple, np.array): New labels

to conform the categorical axis to.

categorical_order_ascending (bool, optional):
Sort order of the categorical axis. Default False.

`parallel`(*data_frame*, *categorical_columns*, *numeric_column*, *color_column=None*, *color_order=None*, *categorical_order_by='values'*, *categorical_order_ascending=False*, *line_dash='solid'*, *line_width=4*, *alpha=1.0*)

Parallel coordinate plot.

Args:
data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

numeric_column (str): Column name to plot on the numerical axis. color_column (str, optional): Column name to group by on

the color dimension.

color_order (list, optional): List of values within the
'color_column' for specific color sort.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'values'. - 'values': Order categorical axis by the numerical axis values. - 'labels': Order categorical axis by the categorical labels. - array-like object (list, tuple, np.array): New labels

to conform the categorical axis to.

categorical_order_ascending (bool, optional):
Sort order of the categorical axis. Default False.

line_dash (str, optional): Dash style for the line. One of:

- 'solid'
- 'dashed'

- 'dotted'
- 'dotdash'
- 'dashdot'

line_width (int, optional): Width of the line alpha (float): Alpha value

scatter(*data_frame*, *categorical_columns*, *numeric_column*, *size_column=None*, *color_column=None*, *color_order=None*, *categorical_order_by='count'*, *categorical_order_ascending=False*, *alpha=1.0*, *marker='circle'*)

Scatter chart.

Note:
To change the orientation set x_axis_type or y_axis_type argument of the Chart object.

Args:
data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

numeric_column (str): Column name to plot on the numerical axis. size_column (str, optional): Column name of numerical values

to plot on the size dimension.

color_column (str, optional): Column name to group by on
the color dimension.

color_order (list, optional):

List of values within the 'color_column' for
specific color sort.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'count'. - 'count': Order categorical axis by the count of values. - 'labels': Order categorical axis by the categorical labels. - array-like object (list, tuple, np.array): New labels

to conform the categorical axis to.

categorical_order_ascending (bool, optional):
Sort order of the categorical axis. Default False.

alpha (float): Alpha value. marker (str): marker type. Valid types:

'asterisk', 'circle', 'circle_cross', 'circle_x', 'cross', 'diamond', 'diamond_cross', 'hex', 'inverted_triangle', 'square', 'square_x', 'square_cross', 'triangle', 'x', '*', '+', 'o', 'ox', 'o+'

text(*data_frame*, *categorical_columns*, *numeric_column*, *text_column*, *color_column=None*, *color_order=None*, *categorical_order_by='values'*, *categorical_order_ascending=False*, *font_size='1em'*, *x_offset=0*, *y_offset=0*, *angle=0*, *text_color=None*)

Text plot.

Args:
data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

numeric_column (str): Column name to plot on the numerical axis. text_column (str): Column name to plot as text labels. color_column (str, optional): Column name to group by on the

color dimension.

color_order (list, optional): List of values within the
'color_column' for specific color sort.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'values'. - 'values': Order categorical axis by the numerical axis

values. Default.

- 'labels': Order categorical axis by the categorical labels.
- **array-like object (list, tuple, np.array): New labels**
to conform the categorical axis to.

categorical_order_ascending (bool, optional): Sort order of the
categorical axis. Default False.

font_size (str, optional): Size of text. x_offset (int, optional): # of pixels for horizontal text offset.

Can be negative. Default: 0.

y_offset (int, optional): # of pixels for vertical text offset.
Can be negative. Default: 0.

angle (int): Degrees from horizontal for text rotation. text_color (str): Color name or hex value.

See chartify.color_palettes.show() for available color names. If omitted, will default to the next color in the current color palette.

`text_stacked`(*data_frame*, *categorical_columns*, *numeric_column*, *stack_column*, *text_column*, *normalize=False*, *stack_order=None*, *categorical_order_by='values'*, *categorical_order_ascending=False*, *font_size='1em'*, *x_offset=0*, *y_offset=0*, *angle=0*, *text_color=None*)

Text plot for use with stacked plots.

Args:
data_frame (pandas.DataFrame): Data source for the plot. categorical_columns (str or list): Column name to plot on

the categorical axis.

numeric_column (str): Column name to plot on the numerical axis. text_column (str): Column name to plot as text labels.

Note: Null text values will be omitted from the plot.

stack_column (str): Column name to group by on the stack dimension. normalize (bool, optional): Normalize numeric dimension for

100% stacked bars. Default False.

stack_order (list, optional): List of values within the
'stack_column' dimension for specific stack sort.

categorical_order_by (str or array-like, optional):
Dimension for ordering the categorical axis. Default 'values'. - 'values': Order categorical axis by the numerical

axis values. Default.

- 'labels': Order categorical axis by the categorical labels.
- **array-like object (list, tuple, np.array): New labels**
to conform the categorical axis to.

categorical_order_ascending (bool, optional): Sort order of the
categorical axis. Default False.

font_size (str, optional): Size of text. x_offset (int, optional): # of pixels for horizontal text offset.

Can be negative. Default: 0.

y_offset (int, optional): # of pixels for vertical text offset.
Can be negative. Default: 0.

angle (int): Degrees from horizontal for text rotation. text_color (str): Color name or hex value.

See chartify.color_palettes.show() for available color names. If omitted, will default to the next color in the current color palette.

2.6 Axes

Module for logic related to chart axes.

class chartify._core.axes.**BaseAxes**(*chart*)

Base class for axes.

hide_xaxis()

Hide the tick labels, ticks, and axis lines of the x-axis.

The x-axis label will remain visible, but can be removed with .axes.set_xaxis_label("")

set_xaxis_label(*label*)

Set x-axis label text.

Args:
label (string): the text for the x-axis label

Returns:
Current chart object

set_xaxis_tick_orientation(*orientation='horizontal'*)

Change the orientation or the x axis tick labels.

Args:

orientation (str or list of str):
str: 'horizontal', 'vertical', or 'diagonal' list of str: different orientation values corresponding to each level of the grouping. Example: ['horizontal', 'vertical']

property xaxis_label

Return x-axis label.

Returns:
x-axis label text

class chartify._core.axes.**CategoricalXYAxes**(*chart*)

Axis class for categorical X and Y axes.

class chartify._core.axes.**DatetimeXNumericalYAxes**(*chart*)

Axis class for datetime X and numerical Y axes.

class chartify._core.axes.**NumericalXAxis**(*chart*)

Axis class for numerical X and categorical Y axes

class chartify._core.axes.**NumericalXYAxes**(*chart*)

Axis class for numerical X and Y axes.

class chartify._core.axes.**NumericalYAxis**(*chart*)

Axis class for numerical Y and categorical X axes

class chartify._core.axes.**SecondAxis**

Class for second axis.

- Plotting (.plot)
- Axes (.axes)

class chartify._core.axes.**SecondYNumericalAxis**(*chart*)

Axis class for second Y numerical axes.

2.7 Callout

class chartify._core.callout.**Callout**(*chart*)

Class for adding callouts to the chart.

box(*top=None*, *bottom=None*, *left=None*, *right=None*, *alpha=0.2*, *color='red'*)

Add box callout to the chart.

Args:
top (numeric, optional): Top edge of the box. bottom (numeric, optional): Bottom edge of the box. left (numeric, optional): Left edge of the box. right (numeric, optional): Right edge of the box. alpha (float, optional): 0.2 color (str): Color name or hex value.

See chartify.color_palettes.show() for available color names.

Note:
The box will extend to the edge if the corresponding position argument is omitted.

Returns:
Current chart object

line(*location*, *orientation='width'*, *line_color='black'*, *line_dash='solid'*, *line_width=2*, *line_alpha=1.0*)

Add line callout to the chart.

Args:
location (numeric): orientation (str, optional): (default: 'width')

- 'width'

- 'height'

line_color (str, optional): Color name or hex value.
See chartify.color_palettes.show() for available color names.

line_dash (str, optional): Dash style for the line. One of:

- 'solid'
- 'dashed'
- 'dotted'
- 'dotdash'
- 'dashdot'

line_width (int, optional): Width of the line line_alpha (float, optional): Alpha of the line. Between 0 and 1.

Returns:
Current chart object

`line_segment`(*x_start*, *y_start*, *x_end*, *y_end*, *line_color='black'*, *line_dash='solid'*, *line_width=2*, *line_alpha=1.0*)

Add line segment callout to the chart.

Args:
x_start (numeric) y_start (numeric) x_end (numeric) y_end (numeric) line_color (str): Color name or hex value.

See chartify.color_palettes.show() for available color names.

line_dash (str, optional): Dash style for the line. One of:

- 'solid'
- 'dashed'
- 'dotted'
- 'dotdash'
- 'dashdot'

line_width (int, optional): Width of the line line_alpha (float, optional): Alpha of the line. Between 0 and 1.

Returns:
Current chart object

`text`(*text*, *x*, *y*, *text_color='black'*, *text_align='left'*, *font_size='1em'*, *angle=0*)

Add text callout to the chart.

Note:
Use `

` within text for newlines.

Args:
x (numeric): x location of the text. y (numeric, optional): y location of the text. text_color (str): Color name or hex value.

See chartify.color_palettes.show() for available color names.

text_align (str: 'left', 'right', 'center'): Text alignment. font_size (str): Font size. angle (int, 0 to 360): Angle in degrees from horizontal. Default: 0

Returns:
Current chart object

2.8 Options

class chartify._core.options.**ChartifyOptions**

get_option(*option_name*)

Return the value of the given option

set_option(*option_name*, *option_value*)

Set the default value of the specified option.

Available options:

'style.color_palette_categorical': (str)
Color palette for categorical palette types.

'style.color_palette_sequential': (str)
Color palette for sequential palette types.

'style.color_palette_diverging': (str)
Color palette for diverging palette types.

'style.color_palette_accent': (str)
Color palette for assigning color to specific values.

'style.color_palette_accent_default_color': (str)

Default color of values in the 'color_column' that
are not accented.

Default: 'light grey'

'chart.blank_labels': boolean

If False, chartify.Chart objects populate the default
chart labels with helper text.

Default: False

2.9 Style

class chartify._core.style.**Style**(*chart*, *layout*)

Contains attributes and methods for modifying the aesthetic style of the chart.

set_color_palette(*palette_type*, *palette=None*, *accent_values=None*)

Args:

palette_type:

- **'categorical': Use when the color dimension**
has no meaningful order.

- **'sequential': Use when the color dimension**
  has a sequential order.
- 'diverging'
- **'accent': Use to assign color to specific**
  values in the color dimension.

palette (color palette name, ColorPalette object, or list of colors)
See chartify.color_palettes.show() for palette & color names. Default: 'Spotify Palette'

accent_values (list or dict): List of values that should be accented or dictionary of 'value': 'color' pairs.

Only applies to 'accent' palette type.

2.10 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

2.10.1 Code of Conduct

This project adheres to the Open Code of Conduct. By participating, you are expected to honor this code.

You can contribute in many ways:

2.10.2 Types of Contributions

Report Bugs

Report bugs at https://github.com/spotify/chartify/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

chartify could always use more documentation, whether as part of the official chartify docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/spotify/chartify/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

2.10.3 Get Started!

Ready to contribute? Here's how to set up *chartify* for local development.

1. Fork the *chartify* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/chartify.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv chartify
$ cd chartify/
$ pip3 install -r requirements_dev.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 chartify tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

2.10.4 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.8, 3.9 and 3.10. Check https://github.com/spotify/chartify/actions and make sure that the tests pass for all supported Python versions.

2.10.5 Tips

The Makefile contains a bunch of useful commands for developers.

Note: You must activate the virtualenv where you installed requirements_dev.txt for these commands to work.

To run all python tests and generate code coverage:
$ make coverage

To generate documentation:
$ make docs

Use *help* to see the list of available commands:
$ make help

2.11 Credits

2.11.1 Development Lead

- Chris Halpert <chalpert@spotify.com>

2.11.2 Contributors

- Amar Patel <amar@spotify.com>
- Lynn Root <lynn@spotify.com>
- Skyler Johnson <skyler@spotify.com>

2.12 History

2.12.1 4.0.5 (2023-10-12)

- Relaxed scipy and pandas version requirements to allow verions 2.x

2.12.2 4.0.4 (2023-08-23)

- Documentation build fix
- Pin tornado requirement to reduce vulnerability

2.12.3 4.0.3 (2023-04-21)

- Require jupyter_bokeh to enable html output

2.12.4 4.0.2 (2023-03-30)

- Fix categorical_order_by check for scatter plot
- Fix categorical_order_by check for _construct_source
- Refactor category sorting in _construct_source
- Add tests for categorical_order_by
- Fix scatter plot tests that used line plots

2.12.5 4.0.1 (2023-03-24)

- Updated version requirement of pillow to avoid bug

2.12.6 4.0.0 (2023-03-23)

- Dropped support for python 3.6 and 3.7

2.12.7 3.1.0 (2023-03-22)

- Added Boxplot Chart including example in examples notebook

2.12.8 3.0.5 (2022-12-13)

- Fixed a few errors in example and tutorial notebooks
- Fixed a typo in requirements.txt

2.12.9 3.0.4 (2022-10-18)

- Updated package requirements
- Got rid of future deprecation warnings
- Bugfix related to legend for graphs with multiple groups and colors

2.12.10 3.0.2 (2020-10-21)

- Support pyyaml 5.2+

2.12.11 3.0.1 (2020-06-02)

- Reduce dependencies by switching from Jupyter to IPython.

2.12.12 3.0.0 (2020-05-29)

- Updated Python to 3.6+ and Pandas to 1.0+ (Thanks @tomasaschan!)
- Updated Bokeh to 2.0+
- Removed colour dependency to fix setup errors.

2.12.13 2.7.0 (2019-11-27)

Bugfixes:

- Updated default yaml loader to move off of deprecated method (Thanks @vh920!)
- Updated legend handling to adjust for deprecated methods in recent versions of Bokeh (Thanks for reporting @jpkoc)
- Updated license in setup.py (Thanks for reporting @jsignell)
- Bump base Pillow dependency to avoid insecure version.
- Update MANIFEST to include missing files (Thanks @toddrme2178!)

2.12.14 2.6.1 (2019-08-15)

Bugfixes:

- Moved package requirements and fixed bug that occured with latest version of Bokeh (Thanks @emschuch & @mollymzhu!)
- Fixed bug in README while generating docs (Thanks @Bharat123rox!)

2.12.15 2.6.0 (2019-03-08)

Improvements:

- Allows users to plot colors on bar charts that aren't contained in the categorical axis.

Bugfixes:

- Fixed bug that caused float types to break when plotted with categorical text plots (Thanks for finding @danela!)
- Fixed broken readme links.

2.12.16 2.5.0 (2019-02-17)

Improvements:

- Added Radar Chart

2.12.17 2.4.0 (2019-02-16)

Improvements:

- Added second Y axis plotting.
- Removed Bokeh loading notification on import (Thanks @canavandl!)
- Added support for custom Bokeh resource loading (Thanks @canavandl!)
- Added example for Chart.save() method (Thanks @david30907d!)

Bugfixes:

- Updated documentation for saving and showing svgs.
- Fixed bug that broke plots with no difference between min and max points. (Thanks for finding @fabioconcina!)

2.12.18 2.3.5 (2018-11-21)

Improvements:

- Updated docstrings (Thanks @gregorybchris @ItsPugle!)
- Added SVG output options to Chart.show() and Chart.save() (Thanks for the suggestion @jdmendoza!)

Bugfixes:

- Fixed bug that caused source label to overlap with xaxis labels.
- Fixed bug that prevented x axis orientation changes with datetime axes (Thanks for finding @simonwongwong!)
- Fixed bug that caused subtitle to disappear with *outside_top* legend location (Thanks for finding @simonwongwong!)
- Line segment callout properties will work correctly. (Thanks @gregorybchris!)

2.12.19 2.3.4 (2018-11-13)

- Updated Bokeh version requirements to support 1.0

2.12.20 2.3.3 (2018-10-24)

- Removed upper bound of Pillow dependency.

2.12.21 2.3.2 (2018-10-18)

- Stacked bar and area order now matches default vertical legend order.
- Added method for shifting color palettes.
- Added scatter plots with a single categorical axis.
- Fixed bug with text_stacked that occurred with multiple categorical levels.

2.12.22 2.3.1 (2018-09-27)

- Fix scatter plot bug that can occur due to nested data types.

2.12.23 2.3.0 (2018-09-26)

- Added hexbin plot type.
- More control over grouped axis label orientation.
- Added alpha control to scatter, line, and parallel plots.
- Added control over marker style to scatter plot.
- Added ability to create custom color palettes.
- Changed default accent color.
- Visual tweaks to lollipop plot.
- Bar plots with a few number of series will have better widths.

2.12.24 2.2.0 (2018-09-17)

- First release on PyPI.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

T

X