
sjvisualizer

Release 0.0.8

Sjoerd Tilmans

Jul 05, 2023

CONTENTS

1	Indices and tables	1
	Index	13

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

`sjvisualizer.Canvas.calc_spacing(value, current_spacing, n)`

class `sjvisualizer.Canvas.canvas`(*width=None, height=None, bg=(255, 255, 255), colors={}*)

Canvas to which all the graphs will be drawn

Parameters

bg (*tuple of length 3 with integers*) – Background color in RGB, defaults to (255, 255, 255) (white)

add_logo(*logo*)

Helper function to add a logo

Parameters

logo – image name of your logo, absolute or relative path

:type str

add_sub_plot(*sub_plot*)

Function to add sub plots to this canvas

Parameters

sub_plot (`sjvisualizer.Canvas.sub_plot`) – sub_plot object

add_sub_title(*text, color=(0, 0, 0)*)

Helper function to add a sub title to your animation.

Parameters

- **text** (*str*) – sub title to be displayed at the top of the visualization
- **color** (*tuple of length 3 with integers*) – sub title color in RGB, defaults to (0, 0, 0) black

add_time(*df, time_indicator='year', color=(150, 150, 150)*)

Helper function to add a timestamp to the visualization

Parameters

- **df** (`pandas.DataFrame`) – pandas dataframe that holds the timestamps as the index
- **time_indicator** (*str*) – determine the format of the timestamp, possible values: “day”, “month”, “year”, defaults to “year”

- **color** (*tuple of length 3 with integers*) – text color in RGB, defaults to (150, 150, 150)

add_title(*text, color=(0, 0, 0)*)

Helper function to add a title to your animation.

Parameters

- **text** (*str*) – title to be displayed at the top of the visualization
- **color** (*tuple of length 3 with integers*) – title color in RGB, defaults to (0, 0, 0) black

play(*df=None, fps=30*)

Main loop of the animation. This function will orchestrate the animation for each time step set in the pandas df

Parameters

- **df** (*pandas.DataFrame*) – pandas data frame to be animated
- **fps** (*int*) – frame rate of the animation, defaults to 30 frames per second

set_decimals(*decimals*)

update(*time*)

Update function that gets called every frame of the animation.

Parameters

- **time** (*datetime object*) – time object that corresponds to the frame

`sjvisualizer.Canvas.format_date`(*time, time_indicator, format='Europe'*)

`sjvisualizer.Canvas.hex_to_rgb`(*h*)

`sjvisualizer.Canvas.load_image`(*path, x, y, root, name*)

class `sjvisualizer.Canvas.sub_plot`(*canvas=None, width=None, height=None, x_pos=None, y_pos=None, start_time=None, text=None, df=None, multi_color_df=None, anchor='c', sort=True, colors={}, root=None, display_percentages=True, display_label=True, title=None, invert=False, origin='s', display_value=True, font_color=(0, 0, 0), back_ground_color=(255, 255, 255), events={}, time_indicator='year', number_of_bars=None, unit="", x_ticks=4, y_ticks=4, log_scale=False, only_show_latest_event=True, allow_decrease=True, format='Europe', draw_points=True, area=True, color_bar_color=[[100, 100, 100], [255, 0, 0]], **kwargs*)

Basic sub_plot class from which all chart types are inherited

Parameters

- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the plot in pixels
- **height** (*int*) – height of the plot in pixels
- **x_pos** (*int*) – the x location of the top left pixel in this plot
- **y_pos** (*int*) – the y location of the top left pixel in this plot
- **font_color** (*tuple of length 3 with integers*) – font color

```

load_image()

save_colors()

set_root(root)

update(time)

sjvisualizer.Canvas.truncate(n, decimals=1)

class sjvisualizer.BarRace.bar(name=None, canvas=None, color=None, root=None, target_y=0, x=100,
                               size=10, width=0, radius=0, value=0, unit=None, display_value=True,
                               multi_colors=None, color_data=None, font_color=(0, 0, 0), mode=None,
                               colors=None, decimal_places=1, font_scale=1)

    delete()

    draw(target_y=0, width=0, img=None, value=0, color_data=None)

    update(target_y=0, width=0, value=0, color_data=None)

class sjvisualizer.BarRace.bar_race(canvas=None, width=None, height=None, x_pos=None,
                                     y_pos=None, start_time=None, text=None, df=None,
                                     multi_color_df=None, anchor='c', sort=True, colors={}, root=None,
                                     display_percentages=True, display_label=True, title=None,
                                     invert=False, origin='s', display_value=True, font_color=(0, 0, 0),
                                     back_ground_color=(255, 255, 255), events={},
                                     time_indicator='year', number_of_bars=None, unit="", x_ticks=4,
                                     y_ticks=4, log_scale=False, only_show_latest_event=True,
                                     allow_decrease=True, format='Europe', draw_points=True,
                                     area=True, color_bar_color=[[100, 100, 100], [255, 0, 0]],
                                     **kwargs)

```

Class to construct a bar race

Parameters

- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the plot in pixels, default depends on screen resolution
- **height** (*int*) – height of the plot in pixels, default depends on screen resolution
- **x_pos** (*int*) – the x location of the top left pixel in this plot, default depends on screen resolution
- **y_pos** (*int*) – the y location of the top left pixel in this plot, default depends on screen resolution
- **df** (*pandas.DataFrame*) – pandas dataframe that holds the data
- **colors** – dictionary that holds color information for each of the data categories. The key of the dict should

correspond to the name of the data category (column). The value of the dict should be the RGB values of the color:

```

{
    "United States": [
        23, 60, 225
    ]
}

```

}, default is {}

Parameters

- **unit** (*str*) – unit of the values visualized, default is “”
- **back_ground_color** – color of the background. To hide bars that fall outside of the top X, a square is drawn

at the bottom of the visualization. Typically you want this square to match the color of the background. Default is (255,255,255) :type back_ground_color: tuple of length 3 with integers

Parameters

- **font_color** (*tuple of length 3 with integers*) – font color, default is (0,0,0)
- **sort** (*boolean*) – should the values of this plot be sorted? True/False, default is True
- **number_of_bars** (*int*) – number of bars to display in the animation, default is 10 unless you have less than 10 data categories

draw(*time*)

update(*time*)

class sjvisualizer.BarRace.**bar_stripes**(*canvas, y_min, y_max, row, x, width, height, number_of_bars, invert, allow_decrease=True*)

draw(*row*)

update(*row*)

class sjvisualizer.PieRace.**pie**(*name=None, canvas=None, x1=0, y1=0, x2=0, y2=0, start=0, extent=0, color=None, root=None, display_percentages=True, display_label=True, colors=None, load_img=True, font_color=(0, 0, 0), scale_label=True*)

draw(*start=0, extent=0*)

update(*target_start=0, target_extent=0*)

class sjvisualizer.PieRace.**pie_plot**(*canvas=None, width=None, height=None, x_pos=None, y_pos=None, start_time=None, text=None, df=None, multi_color_df=None, anchor='c', sort=True, colors={}, root=None, display_percentages=True, display_label=True, title=None, invert=False, origin='s', display_value=True, font_color=(0, 0, 0), back_ground_color=(255, 255, 255), events={}, time_indicator='year', number_of_bars=None, unit='', x_ticks=4, y_ticks=4, log_scale=False, only_show_latest_event=True, allow_decrease=True, format='Europe', draw_points=True, area=True, color_bar_color=[[100, 100, 100], [255, 0, 0]], **kwargs*)

Class to construct a pie chart race

Parameters

- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the plot in pixels, default depends on screen resolution
- **height** (*int*) – height of the plot in pixels, default depends on screen resolution

- **x_pos** (*int*) – the x location of the top left pixel in this plot, default depends on screen resolution
- **y_pos** (*int*) – the y location of the top left pixel in this plot, default depends on screen resolution
- **df** (*pandas.DataFrame*) – pandas dataframe that holds the data
- **colors** – dictionary that holds color information for each of the data categories. The key of the dict should

correspond to the name of the data category (column). The value of the dict should be the RGB values of the color:

```
{
    "United States": [
        23, 60, 225
    ]
}, default is {}
```

Parameters

back_ground_color – color of the background. To hide bars that fall outside of the top X, a square is drawn

at the bottom of the visualization. Typically you want this square to match the color of the background. Default is (255,255,255) :type back_ground_color: tuple of length 3 with integers

Parameters

- **font_color** (*tuple of length 3 with integers*) – font color, default is (0,0,0)
- **sort** (*boolean*) – should the values of this plot be sorted? True/False, default is True

draw(*time*)

update(*time*)

```
class sjvisualizer.LineChart.event(name=None, canvas=None, start_date=None, end_date=None,
                                   color=(225, 225, 225), font_color=(0, 0, 0), font_size=12,
                                   text_font='Microsoft JhengHei UI', parent=None)
```

draw()

update(*date*)

```
class sjvisualizer.LineChart.line(name=None, canvas=None, value=0, unit=None, font_color=(0, 0, 0),
                                  colors=None, time=None, xaxis=None, yaxis=None, chart=None,
                                  draw_points=False)
```

draw(*value, time*)

remove_points()

update(*value, time*)

```
class sjvisualizer.LineChart.line_chart(canvas=None, width=None, height=None, x_pos=None,
                                         y_pos=None, start_time=None, text=None, df=None,
                                         multi_color_df=None, anchor='c', sort=True, colors={},
                                         root=None, display_percentages=True, display_label=True,
                                         title=None, invert=False, origin='s', display_value=True,
                                         font_color=(0, 0, 0), back_ground_color=(255, 255, 255),
                                         events={}, time_indicator='year', number_of_bars=None,
                                         unit="", x_ticks=4, y_ticks=4, log_scale=False,
                                         only_show_latest_event=True, allow_decrease=True,
                                         format='Europe', draw_points=True, area=True,
                                         color_bar_color=[[100, 100, 100], [255, 0, 0]], **kwargs)
```

Class to construct an animated area graph

Parameters

- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the plot in pixels, default depends on screen resolution
- **height** (*int*) – height of the plot in pixels, default depends on screen resolution
- **x_pos** (*int*) – the x location of the top left pixel in this plot, default depends on screen resolution
- **y_pos** (*int*) – the y location of the top left pixel in this plot, default depends on screen resolution
- **df** (*pandas.DataFrame*) – pandas dataframe that holds the data
- **colors** – dictionary that holds color information for each of the data categories. The key of the dict should

correspond to the name of the data category (column). The value of the dict should be the RGB values of the color:

```
{
    "United States": [
        23, 60, 225
    ]
}, default is {}
```

Parameters

- **font_color** (*tuple of length 3 with integers*) – font color, default is (0,0,0)
- **font_size** (*int*) – font size, in pixels
- **draw_points** (*boolean*) – if set to True, the script will draw markers for each line, this may impact performance
- **time_indicator** (*str*) – format of the timestamp, “day”, “month”, “year”, default is “year”

Param events

dictionary to add additional context to the line chart. For example to indicate events in time.
Example:

```
events = {
    "{EVENT NAME}": [{"START DATE DD/MM/YYYY", "END DATE DD/MM/YYYY"}, "Event 1"]:
```

```
[“28/01/2017”, “28/01/2018”], “Event 2”: [“28/01/2019”, “28/01/2020”], “Last event”: [“28/05/2020”,
“28/01/2021”]
```

```
} :type events: dict
```

Parameters

draw_all_events (*boolean*) – by default only the label will be added to the most recent event.
Set this value to True to keep the labels for all events

```
draw(time)
```

```
update(time)
```

```
class sjvisualizer.Total.total(canvas=None, width=None, height=None, x_pos=None, y_pos=None,
                               start_time=None, text=None, df=None, multi_color_df=None, anchor='c',
                               sort=True, colors={}, root=None, display_percentages=True,
                               display_label=True, title=None, invert=False, origin='s',
                               display_value=True, font_color=(0, 0, 0), back_ground_color=(255, 255,
                               255), events={}, time_indicator='year', number_of_bars=None, unit="",
                               x_ticks=4, y_ticks=4, log_scale=False, only_show_latest_event=True,
                               allow_decrease=True, format='Europe', draw_points=True, area=True,
                               color_bar_color=[[100, 100, 100], [255, 0, 0]], **kwargs)
```

```
draw(time)
```

```
update(time)
```

```
class sjvisualizer.Legend.elem(name=None, canvas=None, y=0, unit=None, font_color=(0, 0, 0),
                                colors=None, font=None, parent=None)
```

```
calc_position(target_y)
```

```
draw()
```

```
update(x, y, draw)
```

```
class sjvisualizer.Legend.legend(canvas=None, width=None, height=None, x_pos=None, y_pos=None,
                                  start_time=None, text=None, df=None, multi_color_df=None,
                                  anchor='c', sort=True, colors={}, root=None, display_percentages=True,
                                  display_label=True, title=None, invert=False, origin='s',
                                  display_value=True, font_color=(0, 0, 0), back_ground_color=(255, 255,
                                  255), events={}, time_indicator='year', number_of_bars=None, unit="",
                                  x_ticks=4, y_ticks=4, log_scale=False, only_show_latest_event=True,
                                  allow_decrease=True, format='Europe', draw_points=True, area=True,
                                  color_bar_color=[[100, 100, 100], [255, 0, 0]], **kwargs)
```

Class to construct an animated area graph

Parameters

- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the plot in pixels, default depends on screen resolution
- **height** (*int*) – height of the plot in pixels, default depends on screen resolution
- **x_pos** (*int*) – the x location of the top left pixel in this plot, default depends on screen resolution
- **y_pos** (*int*) – the y location of the top left pixel in this plot, default depends on screen resolution

- **df** (*pandas.DataFrame*) – pandas dataframe that holds the data
- **colors** – dictionary that holds color information for each of the data categories. The key of the dict should

correspond to the name of the data category (column). The value of the dict should be the RGB values of the color:

```
{
    "United States": [
        23, 60, 225
    ]
}, default is {}
```

Parameters

- **font_color** (*tuple of length 3 with integers*) – font color, default is (0,0,0)
- **font_size** (*int*) – font size, in pixels
- **sort** (*boolean*) – should the elements of this graph be sorted based on the value? default is True

draw(*time*)

update(*time*)

```
class sjvisualizer.Axis.axis(canvas, x=0, y=0, length=1000, width=1000, orientation='horizontal', n=3,
                             allow_decrease=False, tick_length=0, is_log_scale=False, is_date=False,
                             color=(50, 50, 50), font_size=20, text_font='Microsoft JhengHei UI',
                             time_indicator='year', line_tickness=3, ticks_only=True, unit='')
```

calc_positions(*value*)

draw(*min=0, max=0*)

update(*min=0, max=0*)

```
sjvisualizer.Axis.calculate_nice_ticks(min_val, max_val, num_ticks, is_log_scale=False)
```

```
class sjvisualizer.Axis.tick(canvas, axis=None, length=0, label_pos='s')
```

draw(*value=0*)

update(*value=0, draw=True, l=0*)

```
class sjvisualizer.DataHandler.DataHandler(excel_file=None, number_of_frames=0, log_scale=False)
```

Class to handle the data, and interpolate values between each data point

Parameters

- **excel_file** (*str*) – source Excel file to get the data
- **number_of_frames** (*int*) – number of frames in your animation. Typically you want to aim for 60*FPS*Duration

```
class sjvisualizer.DataHandler.SizeCompareDataHandler(excel_file=None, number_of_frames=0,
                                                       area=True)
```

```
class sjvisualizer.Date.date(canvas=None, width=None, height=None, x_pos=None, y_pos=None,
    start_time=None, text=None, df=None, multi_color_df=None, anchor='c',
    sort=True, colors={}, root=None, display_percentages=True,
    display_label=True, title=None, invert=False, origin='s', display_value=True,
    font_color=(0, 0, 0), back_ground_color=(255, 255, 255), events={},
    time_indicator='year', number_of_bars=None, unit="", x_ticks=4, y_ticks=4,
    log_scale=False, only_show_latest_event=True, allow_decrease=True,
    format='Europe', draw_points=True, area=True, color_bar_color=[[100, 100,
    100], [255, 0, 0]], **kwargs)
```

Use this to add a timestamp to your visualization.

Parameters

- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the timestamp in pixels (doesn't change the font size), default depends on screen resolution
- **height** (*int*) – height of the timestamp in pixels, this settings also changes the font size, default depends on screen resolution
- **x_pos** (*int*) – the x location of the top left pixel of the timestamp, default depends on screen resolution
- **y_pos** (*int*) – the y location of the top left pixel of the timestamp, default depends on screen resolution
- **prefix** (*str*) – text to prefix the timestamp, default is “
- **time_indicator** (*str*) – format of the timestamp, “day”, “month”, “year”, default is “year”
- **font_color** (*tuple of length 3 with integers*) – font color, default is (0,0,0)

draw(*time*)

update(*time*)

```
class sjvisualizer.StaticImage.static_image(canvas=None, width=None, height=None, x_pos=None,
    y_pos=None, start_time=None, text=None, df=None,
    multi_color_df=None, anchor='c', sort=True, colors={},
    root=None, display_percentages=True,
    display_label=True, title=None, invert=False, origin='s',
    display_value=True, font_color=(0, 0, 0),
    back_ground_color=(255, 255, 255), events={},
    time_indicator='year', number_of_bars=None, unit="",
    x_ticks=4, y_ticks=4, log_scale=False,
    only_show_latest_event=True, allow_decrease=True,
    format='Europe', draw_points=True, area=True,
    color_bar_color=[[100, 100, 100], [255, 0, 0]], **kwargs)
```

Use this to add static images to your visualization.

Parameters

- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the image in pixels
- **height** (*int*) – height of the image in pixels
- **x_pos** (*int*) – the x location of the top left pixel of this image
- **y_pos** (*int*) – the y location of the top left pixel of this image

- **file** (*str*) – file location of the image you want to add the canvas, only png files are support
- **on_top** (*boolean*) – set this to True to always draw this image on top

draw(*args, **kwargs)

update(*args, **kwargs)

```
class sjvisualizer.StaticText.static_text(canvas=None, width=None, height=None, x_pos=None,
                                          y_pos=None, start_time=None, text=None, df=None,
                                          multi_color_df=None, anchor='c', sort=True, colors={},
                                          root=None, display_percentages=True, display_label=True,
                                          title=None, invert=False, origin='s', display_value=True,
                                          font_color=(0, 0, 0), back_ground_color=(255, 255, 255),
                                          events={}, time_indicator='year', number_of_bars=None,
                                          unit="", x_ticks=4, y_ticks=4, log_scale=False,
                                          only_show_latest_event=True, allow_decrease=True,
                                          format='Europe', draw_points=True, area=True,
                                          color_bar_color=[[100, 100, 100], [255, 0, 0]], **kwargs)
```

Class to add a static text to the visualization

Parameters

- **text** (*str*) – text to be displayed, for example a title
- **anchor** (*str*) – Anchors are used to define where text is positioned relative to a reference point. Possible values correspond wind directions: NW N NE W CENTER E SW S SE
- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the plot in pixels, default depends on screen resolution
- **height** (*int*) – height of the text, closely resembles font size
- **x_pos** (*int*) – the x location of the top left pixel in this plot, default depends on screen resolution
- **y_pos** (*int*) – the y location of the top left pixel in this plot, default depends on screen resolution
- **font_color** (*tuple of length 3 with integers*) – font color, default is (0,0,0)

draw(*args, **kwargs)

update(*args, **kwargs)

```
class sjvisualizer.StackedBarChart.bar_graph_y_tick(canvas, value, max_value, width, height, x_pos,
                                                    y_pos, unit, font_size, font_color=(0, 0, 0))
```

draw(max_value, fraction=0)

update(max_value, fraction)

```
class sjvisualizer.StackedBarChart.stacked_bar_chart(canvas=None, width=None, height=None,
x_pos=None, y_pos=None, start_time=None,
text=None, df=None, multi_color_df=None,
anchor='c', sort=True, colors={}, root=None,
display_percentages=True,
display_label=True, title=None, invert=False,
origin='s', display_value=True, font_color=(0,
0, 0), back_ground_color=(255, 255, 255),
events={}, time_indicator='year',
number_of_bars=None, unit="", x_ticks=4,
y_ticks=4, log_scale=False,
only_show_latest_event=True,
allow_decrease=True, format='Europe',
draw_points=True, area=True,
color_bar_color=[[100, 100, 100], [255, 0,
0]], **kwargs)
```

Class to construct an animated stack bar chart

Parameters

- **canvas** (*tkinter.Canvas*) – tkinter canvas to draw the graph to
- **width** (*int*) – width of the plot in pixels, default depends on screen resolution
- **height** (*int*) – height of the plot in pixels, default depends on screen resolution
- **x_pos** (*int*) – the x location of the top left pixel in this plot, default depends on screen resolution
- **y_pos** (*int*) – the y location of the top left pixel in this plot, default depends on screen resolution
- **df** (*pandas.DataFrame*) – pandas dataframe that holds the data
- **colors** – dictionary that holds color information for each of the data categories. The key of the dict should

correspond to the name of the data category (column). The value of the dict should be the RGB values of the color:

```
{
    "United States": [
        23, 60, 225
    ]
}, default is {}
```

Parameters

- **unit** (*str*) – unit of the values visualized, default is ""
- **font_color** (*tuple of length 3 with integers*) – font color, default is (0,0,0)
- **number_of_bars** (*int*) – number of horizontal bars to display in the animation, default is 10.

draw(*time*)

draw_y_ticks(*time*)

update(*time*)

class `sjvisualizer.StackedBarChart.stacked_bar_graph_bar`(*canvas, number, number_of_bars, data, colors, max_value, width, height, x_pos, y_pos*)

draw()

update(*current_max_value*)

INDEX

A

`add_logo()` (*sjvisualizer.Canvas.canvas* method), 1
`add_sub_plot()` (*sjvisualizer.Canvas.canvas* method), 1
`add_sub_title()` (*sjvisualizer.Canvas.canvas* method), 1
`add_time()` (*sjvisualizer.Canvas.canvas* method), 1
`add_title()` (*sjvisualizer.Canvas.canvas* method), 2
`axis` (class in *sjvisualizer.Axis*), 8

B

`bar` (class in *sjvisualizer.BarRace*), 3
`bar_graph_y_tick` (class in *sjvisualizer.StackedBarChart*), 10
`bar_race` (class in *sjvisualizer.BarRace*), 3
`bar_stripes` (class in *sjvisualizer.BarRace*), 4

C

`calc_position()` (*sjvisualizer.Legend.elem* method), 7
`calc_positions()` (*sjvisualizer.Axis.axis* method), 8
`calc_spacing()` (in module *sjvisualizer.Canvas*), 1
`calculate_nice_ticks()` (in module *sjvisualizer.Axis*), 8
`canvas` (class in *sjvisualizer.Canvas*), 1

D

`DataHandler` (class in *sjvisualizer.DataHandler*), 8
`date` (class in *sjvisualizer.Date*), 8
`delete()` (*sjvisualizer.BarRace.bar* method), 3
`draw()` (*sjvisualizer.Axis.axis* method), 8
`draw()` (*sjvisualizer.Axis.tick* method), 8
`draw()` (*sjvisualizer.BarRace.bar* method), 3
`draw()` (*sjvisualizer.BarRace.bar_race* method), 4
`draw()` (*sjvisualizer.BarRace.bar_stripes* method), 4
`draw()` (*sjvisualizer.Date.date* method), 9
`draw()` (*sjvisualizer.Legend.elem* method), 7
`draw()` (*sjvisualizer.Legend.legend* method), 8
`draw()` (*sjvisualizer.LineChart.event* method), 5
`draw()` (*sjvisualizer.LineChart.line* method), 5
`draw()` (*sjvisualizer.LineChart.line_chart* method), 7
`draw()` (*sjvisualizer.PieRace.pie* method), 4
`draw()` (*sjvisualizer.PieRace.pie_plot* method), 5

`draw()` (*sjvisualizer.StackedBarChart.bar_graph_y_tick* method), 10
`draw()` (*sjvisualizer.StackedBarChart.stacked_bar_chart* method), 11
`draw()` (*sjvisualizer.StackedBarChart.stacked_bar_graph_bar* method), 12
`draw()` (*sjvisualizer.StaticImage.static_image* method), 10
`draw()` (*sjvisualizer.StaticText.static_text* method), 10
`draw()` (*sjvisualizer.Total.total* method), 7
`draw_y_ticks()` (*sjvisualizer.StackedBarChart.stacked_bar_chart* method), 11

E

`elem` (class in *sjvisualizer.Legend*), 7
`event` (class in *sjvisualizer.LineChart*), 5

F

`format_date()` (in module *sjvisualizer.Canvas*), 2

H

`hex_to_rgb()` (in module *sjvisualizer.Canvas*), 2

L

`legend` (class in *sjvisualizer.Legend*), 7
`line` (class in *sjvisualizer.LineChart*), 5
`line_chart` (class in *sjvisualizer.LineChart*), 5
`load_image()` (in module *sjvisualizer.Canvas*), 2
`load_image()` (*sjvisualizer.Canvas.sub_plot* method), 2

M

module
 sjvisualizer, 12
 sjvisualizer.Axis, 8
 sjvisualizer.BarRace, 3
 sjvisualizer.Canvas, 1
 sjvisualizer.DataHandler, 8
 sjvisualizer.Date, 8
 sjvisualizer.Legend, 7
 sjvisualizer.LineChart, 5

sjvisualizer.PieRace, 4
 sjvisualizer.StackedBarChart, 10
 sjvisualizer.StaticImage, 9
 sjvisualizer.StaticText, 10
 sjvisualizer.Total, 7

P

pie (class in sjvisualizer.PieRace), 4
 pie_plot (class in sjvisualizer.PieRace), 4
 play() (sjvisualizer.Canvas.canvas method), 2

R

remove_points() (sjvisualizer.LineChart.line method), 5

S

save_colors() (sjvisualizer.Canvas.sub_plot method), 3
 set_decimals() (sjvisualizer.Canvas.canvas method), 2
 set_root() (sjvisualizer.Canvas.sub_plot method), 3
 SizeCompareDataHandler (class in sjvisualizer.DataHandler), 8
 sjvisualizer
 module, 12
 sjvisualizer.Axis
 module, 8
 sjvisualizer.BarRace
 module, 3
 sjvisualizer.Canvas
 module, 1
 sjvisualizer.DataHandler
 module, 8
 sjvisualizer.Date
 module, 8
 sjvisualizer.Legend
 module, 7
 sjvisualizer.LineChart
 module, 5
 sjvisualizer.PieRace
 module, 4
 sjvisualizer.StackedBarChart
 module, 10
 sjvisualizer.StaticImage
 module, 9
 sjvisualizer.StaticText
 module, 10
 sjvisualizer.Total
 module, 7
 stacked_bar_chart (class in sjvisualizer.StackedBarChart), 10
 stacked_bar_graph_bar (class in sjvisualizer.StackedBarChart), 12
 static_image (class in sjvisualizer.StaticImage), 9
 static_text (class in sjvisualizer.StaticText), 10

sub_plot (class in sjvisualizer.Canvas), 2

T

tick (class in sjvisualizer.Axis), 8
 total (class in sjvisualizer.Total), 7
 truncate() (in module sjvisualizer.Canvas), 3

U

update() (sjvisualizer.Axis.axis method), 8
 update() (sjvisualizer.Axis.tick method), 8
 update() (sjvisualizer.BarRace.bar method), 3
 update() (sjvisualizer.BarRace.bar_race method), 4
 update() (sjvisualizer.BarRace.bar_stripes method), 4
 update() (sjvisualizer.Canvas.canvas method), 2
 update() (sjvisualizer.Canvas.sub_plot method), 3
 update() (sjvisualizer.Date.date method), 9
 update() (sjvisualizer.Legend.elem method), 7
 update() (sjvisualizer.Legend.legend method), 8
 update() (sjvisualizer.LineChart.event method), 5
 update() (sjvisualizer.LineChart.line method), 5
 update() (sjvisualizer.LineChart.line_chart method), 7
 update() (sjvisualizer.PieRace.pie method), 4
 update() (sjvisualizer.PieRace.pie_plot method), 5
 update() (sjvisualizer.StackedBarChart.bar_graph_y_tick method), 10
 update() (sjvisualizer.StackedBarChart.stacked_bar_chart method), 11
 update() (sjvisualizer.StackedBarChart.stacked_bar_graph_bar method), 12
 update() (sjvisualizer.StaticImage.static_image method), 10
 update() (sjvisualizer.StaticText.static_text method), 10
 update() (sjvisualizer.Total.total method), 7