

---

# **sjvisualizer**

***Release 0.0.5***

**Sjoerd Tilmans**

**Feb 01, 2023**



# CONTENTS

<b>1</b>	<b>Indices and tables</b>	<b>1</b>
	<b>Index</b>	<b>9</b>



## 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={}*)

Bases: `object`

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)
```

Bases: object

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=0, font\_scale=1*)

Bases: object

**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*)

Bases: sub\_plot

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

corespond 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)
```

Bases: object

**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)
```

Bases: object

**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)
```



Bases: `sub_plot`

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

**background\_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 `background_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.DataHandler.DataHandler`(*excel\_file=None, number\_of\_frames=0, log\_scale=False*)

Bases: `object`

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)
```

Bases: object

```
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)
```

Bases: sub\_plot

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)
```

Bases: sub\_plot

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)
```

Bases: *sub\_plot*

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)



## 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

## B

bar (*class in sjvisualizer.BarRace*), 3  
 bar\_race (*class in sjvisualizer.BarRace*), 3  
 bar\_stripes (*class in sjvisualizer.BarRace*), 4

## C

calc\_spacing() (*in module sjvisualizer.Canvas*), 1  
 canvas (*class in sjvisualizer.Canvas*), 1

## D

DataHandler (*class in sjvisualizer.DataHandler*), 5  
 date (*class in sjvisualizer.Date*), 6  
 delete() (*sjvisualizer.BarRace.bar method*), 3  
 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*), 6  
 draw() (*sjvisualizer.PieRace.pie method*), 4  
 draw() (*sjvisualizer.PieRace.pie\_plot method*), 5  
 draw() (*sjvisualizer.StaticImage.static\_image method*), 7  
 draw() (*sjvisualizer.StaticText.static\_text method*), 7

## F

format\_date() (*in module sjvisualizer.Canvas*), 2

## H

hex\_to\_rgb() (*in module sjvisualizer.Canvas*), 2

## L

load\_image() (*in module sjvisualizer.Canvas*), 2  
 load\_image() (*sjvisualizer.Canvas.sub\_plot method*), 3

## M

module  
   sjvisualizer, 7  
   sjvisualizer.BarRace, 3  
   sjvisualizer.Canvas, 1  
   sjvisualizer.DataHandler, 5  
   sjvisualizer.Date, 6  
   sjvisualizer.PieRace, 4  
   sjvisualizer.StaticImage, 6  
   sjvisualizer.StaticText, 7

## P

pie (*class in sjvisualizer.PieRace*), 4  
 pie\_plot (*class in sjvisualizer.PieRace*), 4  
 play() (*sjvisualizer.Canvas.canvas method*), 2

## 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*), 5  
 sjvisualizer  
   module, 7  
 sjvisualizer.BarRace  
   module, 3  
 sjvisualizer.Canvas  
   module, 1  
 sjvisualizer.DataHandler  
   module, 5  
 sjvisualizer.Date  
   module, 6  
 sjvisualizer.PieRace  
   module, 4  
 sjvisualizer.StaticImage  
   module, 6  
 sjvisualizer.StaticText  
   module, 7  
 static\_image (*class in sjvisualizer.StaticImage*), 6  
 static\_text (*class in sjvisualizer.StaticText*), 7  
 sub\_plot (*class in sjvisualizer.Canvas*), 2

## T

`truncate()` (*in module sjvisualizer.Canvas*), 3

## U

`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*), 6

`update()` (*sjvisualizer.PieRace.pie method*), 4

`update()` (*sjvisualizer.PieRace.pie\_plot method*), 5

`update()` (*sjvisualizer.StaticImage.static\_image method*), 7

`update()` (*sjvisualizer.StaticText.static\_text method*), 7