sjvisualizer

Release 0.0.13

Sjoerd Tilmans

Nov 22, 2023

CONTENTS

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search

`sjvisualizer.plot.bar`(*excel='', title='', sub_title='', duration=1, fps=60, record=False, output_video='output.mp4', unit='', time_indicator='year', font_color=(0, 0, 0), background_color=(255, 255, 255), colors={}, n=8*)

Function to create a bar chart race

Parameters

- **excel** (*string*) – excel file containing the data
- **title** (*string*) – title on top of the animation
- **sub_title** (*string*) – sub-title to provide extra context, displayed just under the main title
- **duration** (*integer*) – length of the animation
- **fps** (*integer*) – number of frames per second
- **record** (*string*) – should the animation be saved as an mp4? Defaults to False. If set to True, the render speed on screen is reduced, however, the playback speed of the video is correct.
- **output_video** – name of the saved video.
- **unit** – unit to be displayed in the graph
- **time_indicator** (*string, 'day', 'month', or 'year'*) – should the time format show year, month or day
- **font_color** (*tuple (R, G, B)*) – color of the texts rendered to the screen. In RGB colors.
- **background_color** (*tuple (R, G, B)*) – color of the background. In RGB colors.
- **colors** (*dict*) – 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 {}

- `n` (*integer*) – number of bars to display

`sjvisualizer.plot.line`(*excel=''*, *title=''*, *sub_title=''*, *duration=1*, *fps=60*, *record=False*, *output_video='output.mp4'*, *unit=''*, *time_indicator='year'*, *events={}*, *font_color=(0, 0, 0)*, *background_color=(255, 255, 255)*, *colors={}*)

Function to create a bar chart race

Parameters

- **excel** (*string*) – excel file containing the data
- **title** (*string*) – title on top of the animation
- **sub_title** (*string*) – sub-title to provide extra context, displayed just under the main title
- **duration** (*integer*) – length of the animation
- **fps** (*integer*) – number of frames per second
- **record** (*string*) – should the animation be saved as an mp4? Defaults to False. If set to True, the render speed on screen is reduced, however, the playback speed of the video is correct.
- **output_video** – name of the saved video.
- **unit** – unit to be displayed in the graph
- **time_indicator** (*string, 'day', 'month', or 'year'*) – should the time format show year, month or day
- **font_color** (*tuple (R, G, B)*) – color of the texts rendered to the screen. In RGB colors.
- **background_color** (*tuple (R, G, B)*) – color of the background. In RGB colors.
- **colors** (*dict*) – 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 {}

Parem 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"]

}

`sjvisualizer.plot.pie`(*excel=''*, *title=''*, *sub_title=''*, *duration=1*, *fps=60*, *record=False*, *output_video='output.mp4'*, *unit=''*, *time_indicator='year'*, *font_color=(0, 0, 0)*, *background_color=(255, 255, 255)*, *colors={}*, *sort=True*)

Function to create a bar chart race

Parameters

- **excel** (*string*) – excel file containing the data
- **title** (*string*) – title on top of the animation
- **sub_title** (*string*) – sub-title to provide extra context, displayed just under the main title
- **duration** (*integer*) – length of the animation
- **fps** (*integer*) – number of frames per second
- **record** (*string*) – should the animation be saved as an mp4? Defaults to False. If set to True, the render speed on screen is reduced, however, the playback speed of the video is correct.
- **output_video** – name of the saved video.
- **unit** – unit to be displayed in the graph
- **time_indicator** (*string, 'day', 'month', or 'year'*) – should the time format show year, month or day
- **font_color** (*tuple (R, G, B)*) – color of the texts rendered to the screen. In RGB colors.
- **background_color** (*tuple (R, G, B)*) – color of the background. In RGB colors.
- **colors** (*dict*) – 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 {}
- **sort** (*boolean*) – should the data be sorted by descending order?

sjvisualizer.plot.**stacked_area**(*excel='', title='', sub_title='', duration=1, fps=60, record=False, output_video='output.mp4', unit='', time_indicator='year', events={}, font_color=(0, 0, 0), background_color=(255, 255, 255), colors={}*)

Function to create a bar chart race

Parameters

- **excel** (*string*) – excel file containing the data
- **title** (*string*) – title on top of the animation
- **sub_title** (*string*) – sub-title to provide extra context, displayed just under the main title
- **duration** (*integer*) – length of the animation
- **fps** (*integer*) – number of frames per second
- **record** (*string*) – should the animation be saved as an mp4? Defaults to False. If set to True, the render speed on screen is reduced, however, the playback speed of the video is correct.
- **output_video** – name of the saved video.
- **unit** – unit to be displayed in the graph

- **time_indicator** (*string, 'day', 'month', or 'year'*) – should the time format show year, month or day
- **font_color** (*tuple (R, G, B)*) – color of the texts rendered to the screen. In RGB colors.
- **background_color** (*tuple (R, G, B)*) – color of the background. In RGB colors.
- **colors** (*dict*) – 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 {}

Parem 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"]

}

`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*, *record=False*, *width=2560*, *height=1440*, *file_name='output.mp4'*)

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
- **record** (*boolean*) – if set to True, the screen will be recorded, this will severely impact performance on high resolution screens
- **width** (*int*) – if record is set to True, this is the width of the window being recorded. Defaults to full screen.
- **height** (*int*) – if record is set to True, this is the height of the window being recorded. Defaults to full screen.
- **file_name** (*str*) – if record is set to True, this is the name of the output file. Defaults to output.mp4.

`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.format_value`(*number*, *decimal=3*)

`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*, *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*, *graph=None*)

`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

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 elements of this graph be sorted based on the value? 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
- `shift` (*int*) – number of pixels to shift the vertical stripe down which is used to hide the bars that fall outside of the top X. This can be used if a background image is used to avoid an ugly white bar covering the background image.
- `font_scale` (*float*) – increase or decrease the font_size. To reduce the font size by 25% set this value to 0.75.

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

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

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 softed? True/False, default is True

draw(*time*)

update(*time*)

class sjvisualizer.LineChart.**event**(*name=None*, *canvas=None*, *start_date=None*, *end_date=None*, *font_color=(0, 0, 0)*, *font_size=12*, *text_font='Microsoft JhengHei UI'*, *parent=None*, *event_color=(255, 255, 255)*)

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*, *line_width=None*, *label_at_end=True*)

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

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

- **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"

Parem 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

- **event_color** (*tuple*) – color of the event indication, default is (225,225,225)
- **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
- **line_width** (*int*) – width of the line

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=''*, *font_color=(0, 0, 0)*, *colors=None*, *font=None*, *parent=None*, *display_values=False*)

calc_position(*target_y*)

draw()

update(*x*, *y*, *draw*, *value=0*)

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

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

- **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
- **display_values** (*boolean*) – display the value of the data category at the end of the legend? default is False

- **unit** (*str*) – unit of the values visualized, default is “”

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=''*, *tick_prefix=''*)

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*, *time_indicator=False*)

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

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)
- **angle** (*float*) – rotation of the text by number of degrees

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

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 ""
- **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