Principles of effective data visualization

Nikos Tsalikidis, CERTH

CERTH SmartWins Training Sessions: Day2
23 April 2024
Thessaloniki

Funded by the European Union

This project has received funding from the European Union's Horizon research and innovation programme under grant agreement No 101078997

Basic notions of statistical representations

The statistical concepts and procedures can be divided into two categories corresponding to the two main areas of statistics:

- Descriptive statistics: make data more comprehensible and interpretable by representing them in tables, charts, and diagrams, and to summarize them by computing certain characteristic measures.
- Inferential statistics: draw inferences about the data generating process, like estimating the parameters of the process or selecting a model that fits it.

Tabular to Graphical Representations

- Tabular Representations (contingency table)

Table A.3 A contingency table for two attributes A and B

	a_1	a_2	a_3	a_4	$\sum$
b_1	8	3	5	2	18
b_2	2	6	1	3	12
b_3	4	1	2	7	14
$\sum$	14	10	8	12	44

- Graphical Representations (pole/stick/bar chart, area and volume charts, frequency polygons , line chart, mosaic chart, pie and stripe chart, histogram, histogram)

Basic graphical representations

Fig. A.4 A mosaic chart for the contingency table of Table A.3

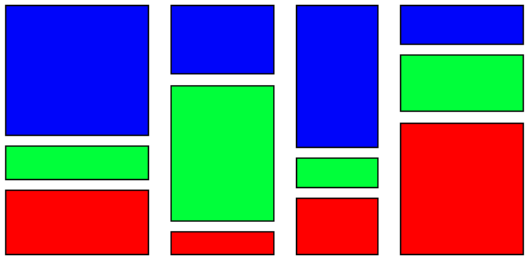
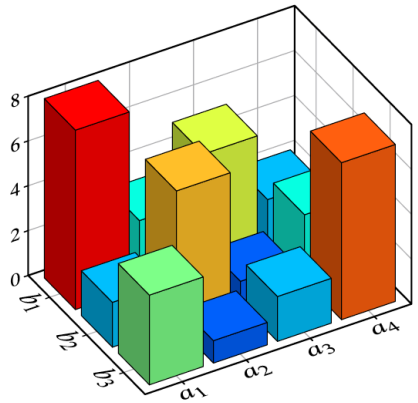


Fig. A.5 A bar chart for the contingency table of Table A.3



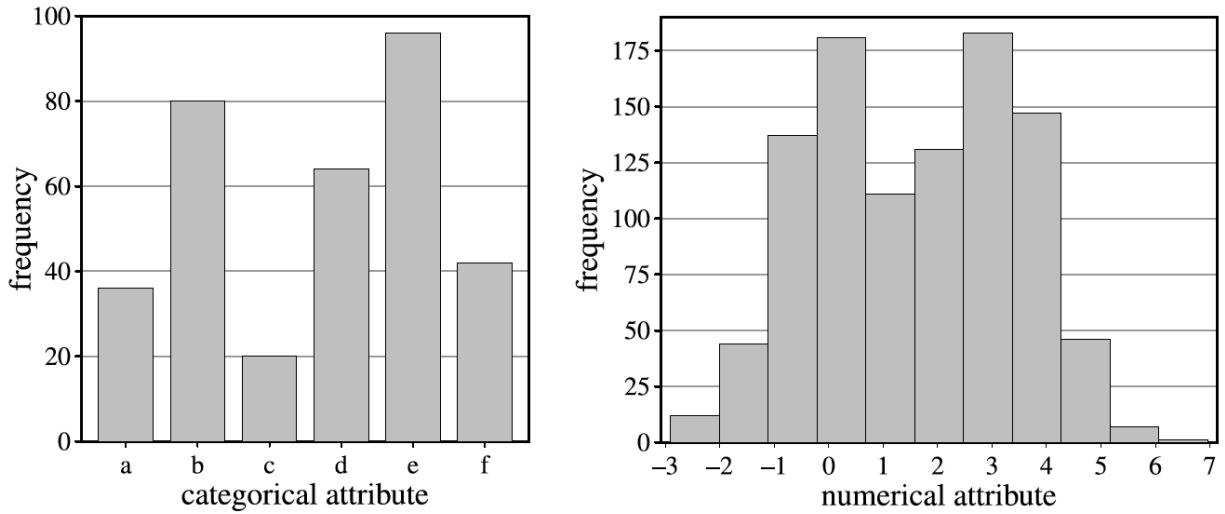
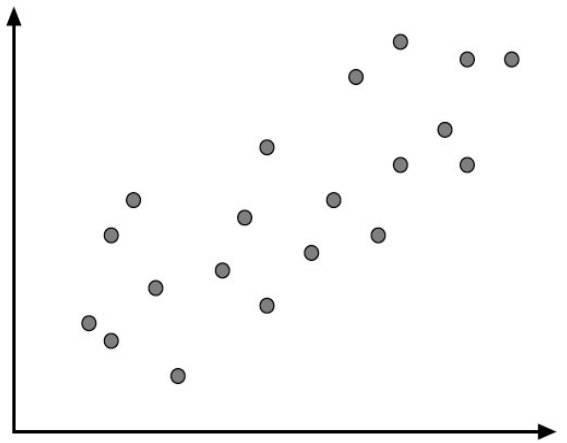


Fig. 4.2 A bar chart (categorical attribute, *left*) and a histogram (numerical attribute, *right*)

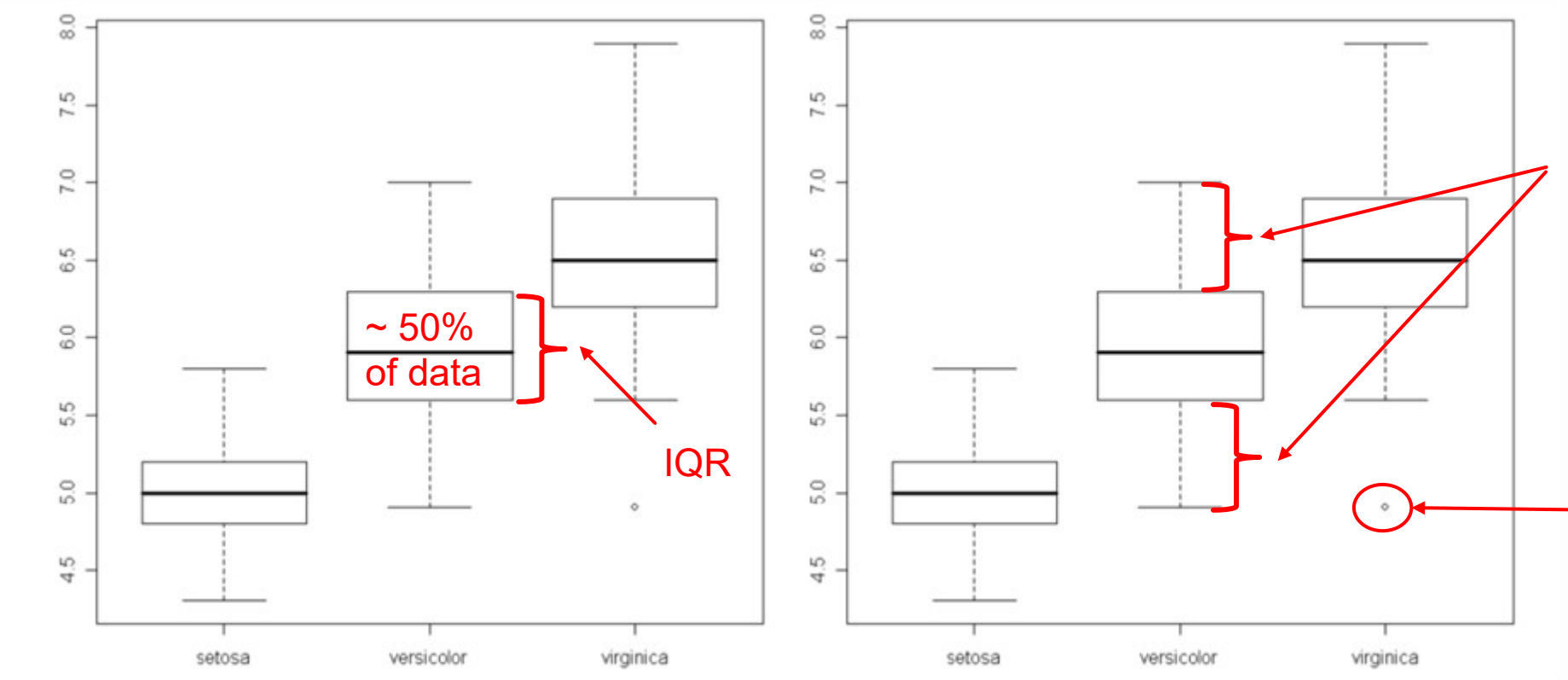


Location Measures

- **Mode** An attribute value is called the (empirical) mode of a data set if it is the value that occurs most frequently in the sample
- **Median** is an attribute value such that half of the sample values are less than it, and the other half is greater.
- **Quantile** is an attribute value such that a certain fraction p, 0 <p< 1, of the sample values is less than this attribute value. The median in particular is the (empirical) $\frac{1}{2}$-quantile of a data set.
- The **mean** of a sample is the arithmetic mean of the sample values.

Boxplots

Boxplots are a very compact way to visualize and summarize the main characteristics of a numeric attribute, through the median, the *IQR*, and possible outliers



Practical examples

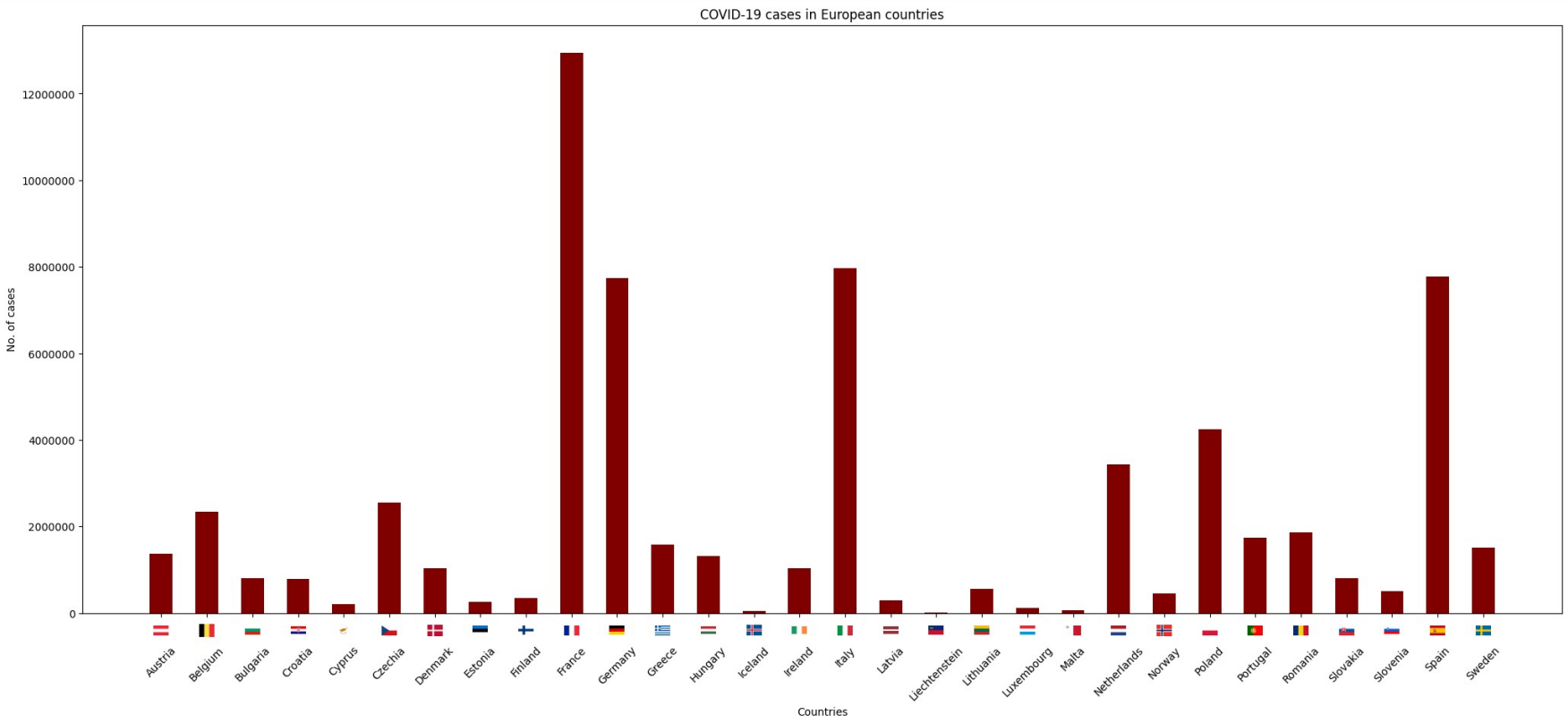
Bar plot w/ custom labels #1

	Country	COVID cases (Until 2022)
0	Austria	1376485
1	Belgium	2347164
2	Bulgaria	799943
3	Croatia	785033
4	Cyprus	215271
5	Czechia	2562235
6	Denmark	1030638
7	Estonia	260396
8	Finland	347276
9	France	12934982
10	Germany	7743228
11	Greece	1592460
12	Hungary	1318093
13	Iceland	42530
14	Ireland	1042212
15	Italy	7971068
16	Latvia	293395
17	Liechtenstein	6915
18	Lithuania	555971
19	Luxembourg	118925
20	Malta	61925
21	Netherlands	3432119
22	Norway	460530
23	Poland	4248559
24	Portugal	1734343
25	Romania	1866102
26	Slovakia	815720
27	Slovenia	512793
28	Spain	7771367
29	Sweden	1509230

- Goal: Create a bar plot but include the country flags as labels
- First step: Download all flags from : https://hampusborgos.github.io/country-flags/
- 2nd step: create functions to match EU countries to flags

```
from matplotlib.offsetbox import OffsetImage,AnnotationBbox
def get_flag(name):
    path = "…/flags/{}.png".format(name)
    im = plt.imread(path)
    return im

def offset_image(coord, name, ax):
    img = get_flag(name)
    im = OffsetImage(img, zoom=0.72)
    im.image.axes = ax

    ab = AnnotationBbox(im, (coord, 0),  xybox=(0., -16.), frameon=False,
                        xycoords='data',  boxcoords="offset points", pad=0)

    ax.add_artist(ab)
```

Bar plot w/ custom labels #1




```
fig, ax =
plt.subplots(1,1,figsize=(25,10))

ax.bar(range(len(df_sum['country'])),
df_sum['cases'],
width=0.5,align="center", color
='maroon')

ax.set_xticks(range(len(df_sum['country']
)))

ax.set_xticklabels(df_sum['country'])

ax.tick_params(axis='x', which='major',
pad=26)

ax.ticklabel_format(style='plain',
axis='y')

plt.xticks(rotation=45)

plt.xlabel("Countries")

plt.ylabel("No. of cases")

plt.title("COVID-19 cases in European
countries")


for i, c in
enumerate((df_sum['country'])):

    offset_image(i, c, ax)

plt.show()
```

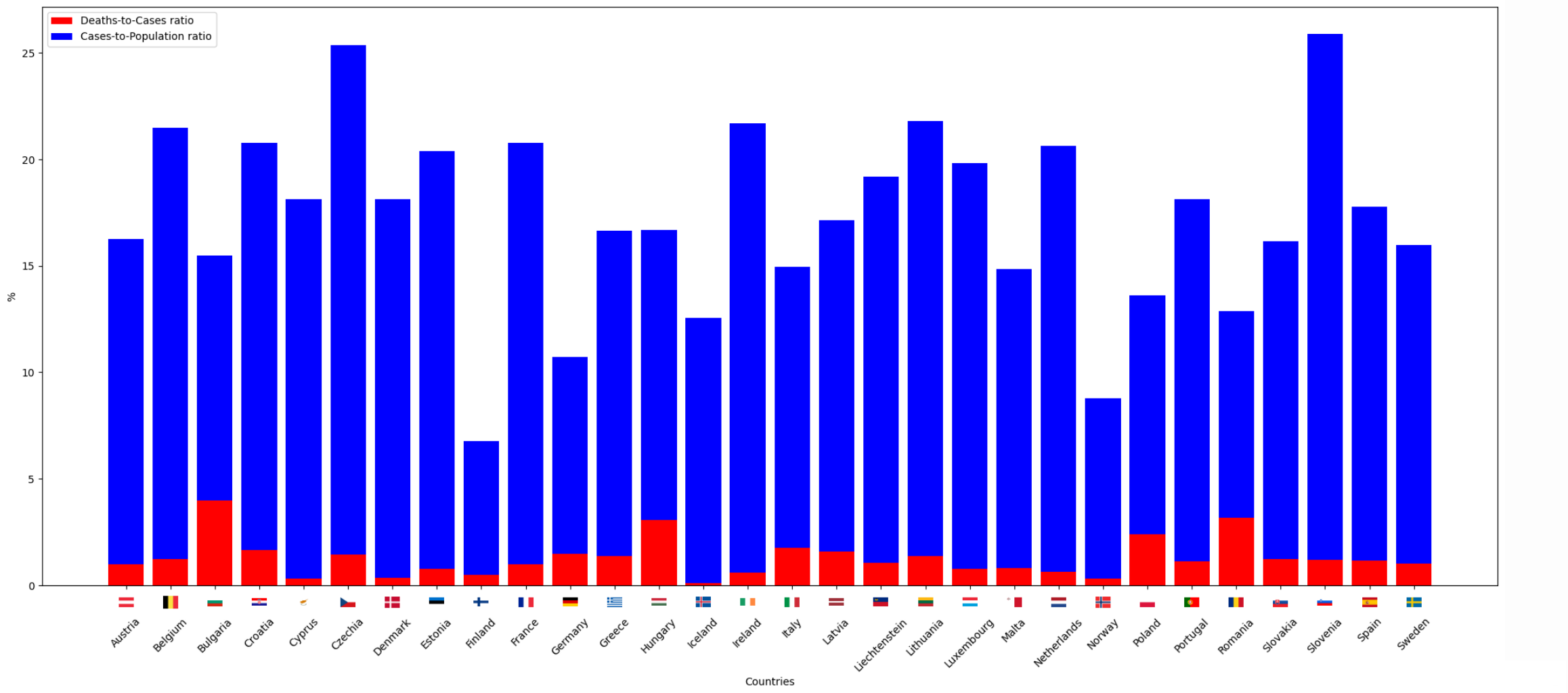
Bar plot w/ custom labels #2

```
df_pop=df.groupby('country')
['cases','deaths','population'].sum().reset_index()

df_pop['div_pop']=(df_pop['cases']/
df_pop['population'])*100

df_pop['div_death']=(df_pop['deaths']/df_pop['cases'])*100

df_pop
```

	country	cases	deaths	population	div_pop	div_death
0	Austria	1376485	13436	9000000	15.294278	0.976109
1	Belgium	2347164	28566	11589000	20.253378	1.217043
2	Bulgaria	799943	31838	6948000	11.513284	3.980034
3	Croatia	785033	12983	4105257	19.122627	1.653816
4	Cyprus	215271	669	1207000	17.835211	0.310771
5	Czechia	2562235	36765	10708000	23.928231	1.434880
6	Denmark	1030638	3433	5792000	17.794164	0.333095
7	Estonia	260396	1967	1326535	19.629787	0.755388
8	Finland	347276	1688	5540720	6.267705	0.486069

```
fig, ax = plt.subplots(1,1,figsize=(25,10))

#fig = plt.figure(figsize = (30, 10))

ax.bar(range(len(df_pop['country'])), df_pop['div_death'],
color='r')

ax.bar(range(len(df_pop['country'])),
df_pop['div_pop'],bottom=df_pop['div_death'], color='b')


ax.set_xticks(range(len(df_sum['country'])))

ax.set_xticklabels(df_sum['country'])

ax.tick_params(axis='x', which='major', pad=26)

ax.ticklabel_format(style='plain', axis='y')

plt.xticks(rotation=45)

plt.xlabel("Countries")

plt.ylabel("%")

plt.legend(["Deaths-to-Cases ratio", "Cases-to-Population ratio"])


for i, c in enumerate((df_sum['country'])):

    offset_image(i, c, ax)

plt.show()
```

Bar plot w/ custom labels #2



Plotly library

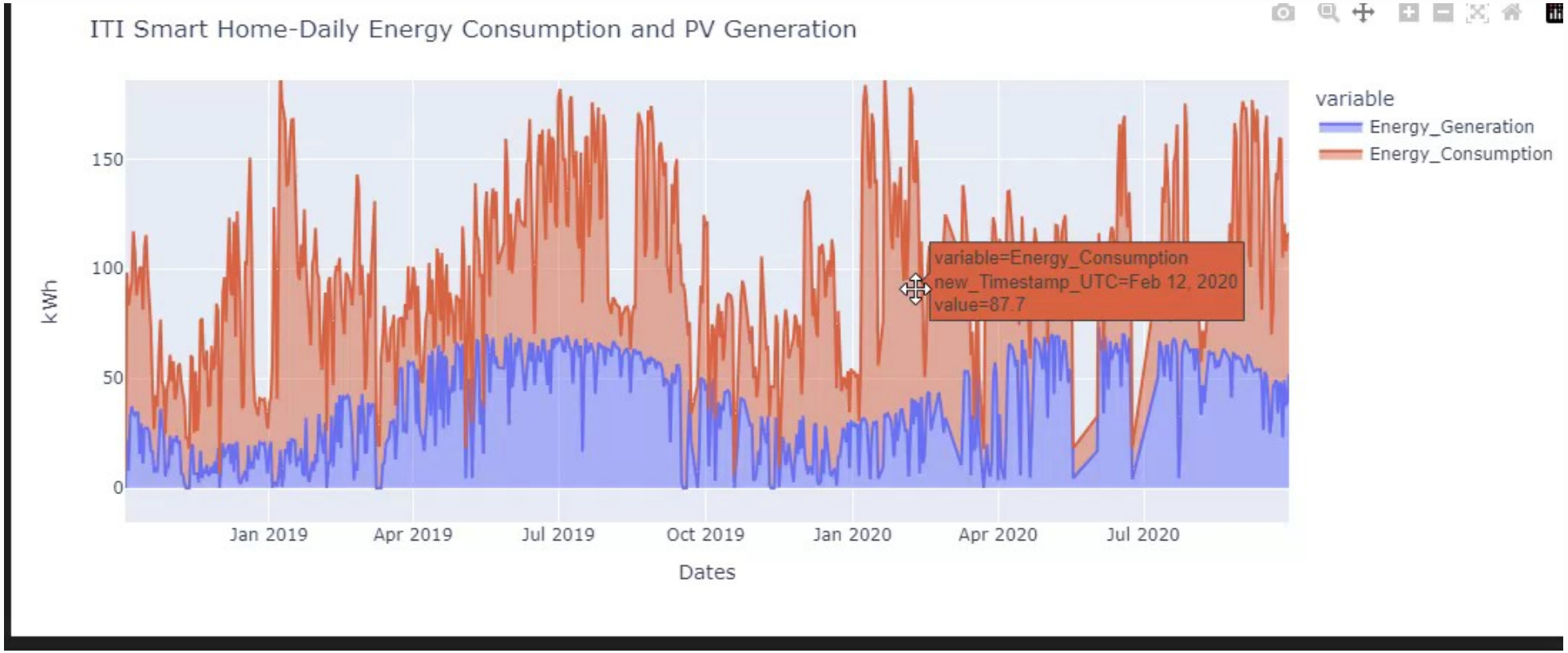
https://plotly.com/

- Interactive Visualization: Plotly is a Python graphing library that creates interactive, publication-quality graphs and figures. It offers a wide variety of chart types including scatter plots, line plots, bar charts, heatmaps, and more.
- Open Source: open-source, used freely in both commercial and non-commercial projects
- Integration: integrates w/ Pandas, NumPy, and SciPy, making it easy to visualize data stored in various formats.

CERTH-ITI Smart Home data

new_Timestamp_UTC	Temperature_v1	Relative_Humidity_v1	Wind_Speed_v1	Clouds_v1	Energy_Consumption	Energy_Generation
2018-10-03	20.533333	69.083333	4.200000	14.583333	8.20	0.000000
2018-10-04	20.741667	55.770833	6.583875	23.958333	76.00	22.275999
2018-10-05	18.853125	72.000000	5.167312	33.072917	75.25	8.021500
2018-10-06	18.597917	65.416667	5.450625	15.104167	57.60	32.574000
2018-10-07	18.686458	68.083333	4.969875	15.364583	57.00	37.189001
...	...	...	...	...	...	...
2020-09-26	21.666667	67.963542	1.287500	58.427083	82.00	22.879875
2020-09-27	21.640625	50.656250	0.868125	14.093750	71.40	49.215500
2020-09-28	22.796875	71.250000	1.204688	50.562500	70.30	37.588500
2020-09-29	21.737500	60.531250	1.262500	36.406250	75.70	39.407500
2020-09-30	19.257143	48.857143	1.468335	25.392857	64.50	51.850250

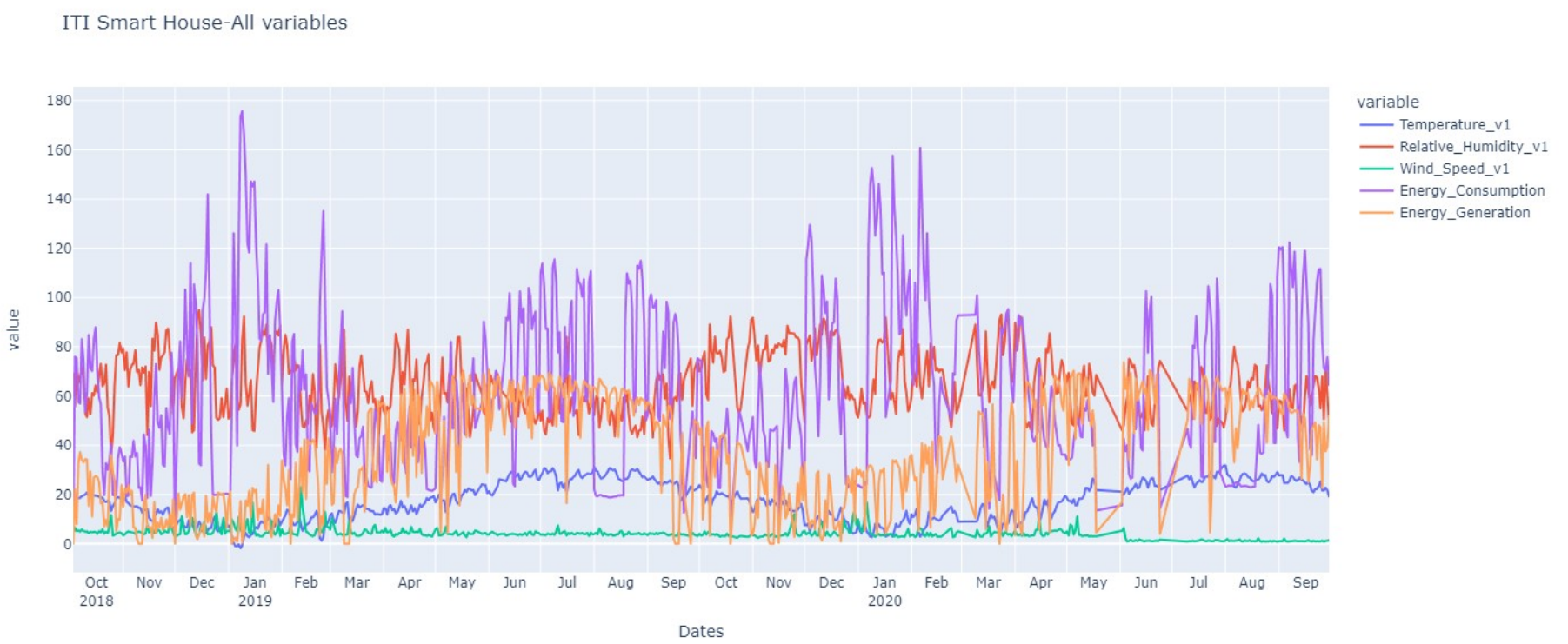
Energy consumption and generation plot




```
fig = px.area(df_cons_weh
, x='new_Timestamp_UTC', y=[ 'Energy_Generation','Energy_Consumption'])
fig.update_layout(title_text='ITI Smart Home-Daily Energy Consumption and PV
Generation', xaxis_title='Dates', yaxis_title='kWh')
fig.show()
```

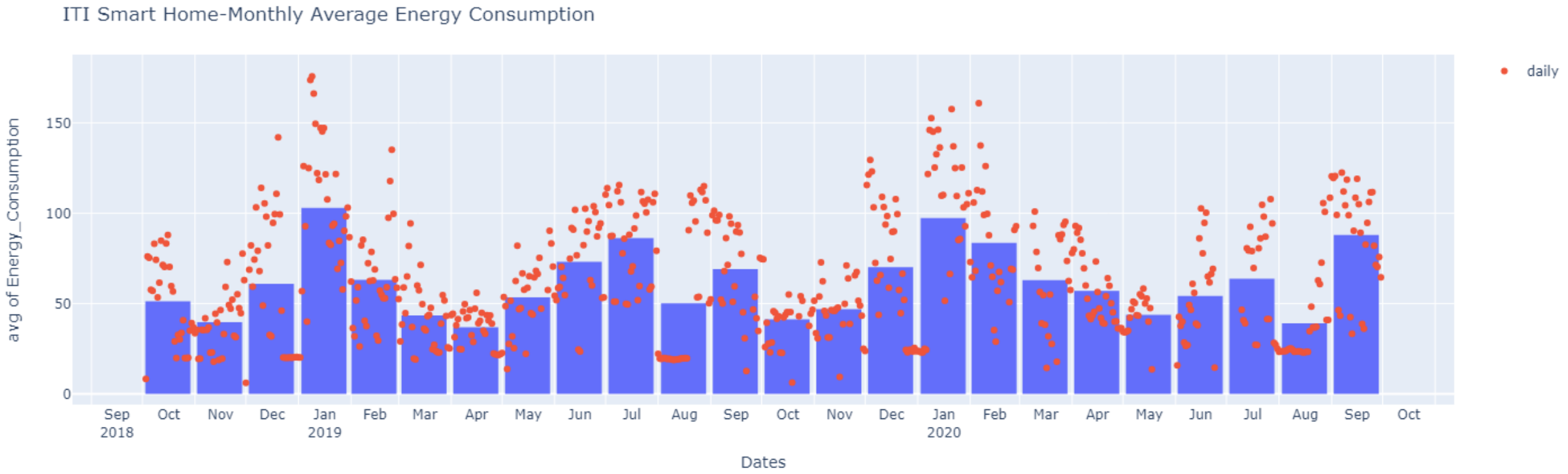
All variables plot

```
df = df_cons_weh

columns_shown =
df.drop(['Clouds_v1'],
axis=1).columns

fig = px.line(df,
x=df_cons_weh.index,
y=columns_shown,

hover_data={"new_Timestamp_UTC"},

title='ITI Smart House-All
variables')

fig.update_xaxes(

    dtick="M1",

    tickformat="%b\n%Y",

    ticklabelmode="period")

fig.update_layout(

    height=600,  # set the height
in pixels

    width=1400, # set the width in
pixels\

    xaxis_title='Dates'

)

fig.show()
```



Monthly and daily averages plot

```
fig = px.histogram(df, x=df_cons_weh['new_Timestamp_UTC'], y=df_cons_weh['Energy_Consumption'], histfunc="avg")
fig.update_traces(xbins_size="M1")
fig.update_xaxes(showgrid=True, ticklabelmode="period", dtick="M1", tickformat="%b\n%Y")
fig.update_layout(bargap=0.1, title_text='ITI Smart Home-Monthly Average Energy Consumption', xaxis_title='Dates')
fig.add_trace(go.Scatter(mode="markers", x=df_cons_weh['new_Timestamp_UTC'], y=df_cons_weh['Energy_Consumption'], name="daily"))
fig.show()
```

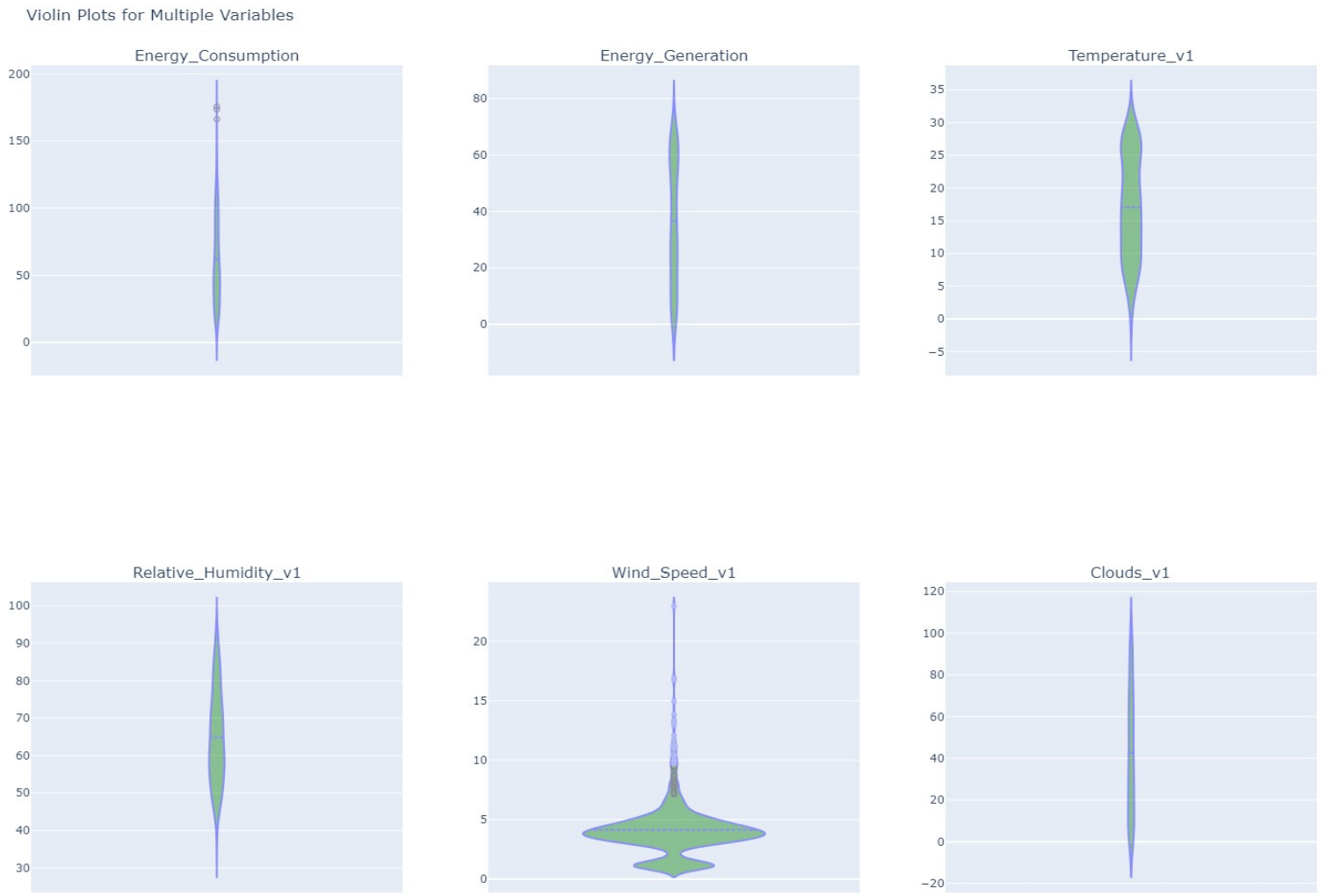


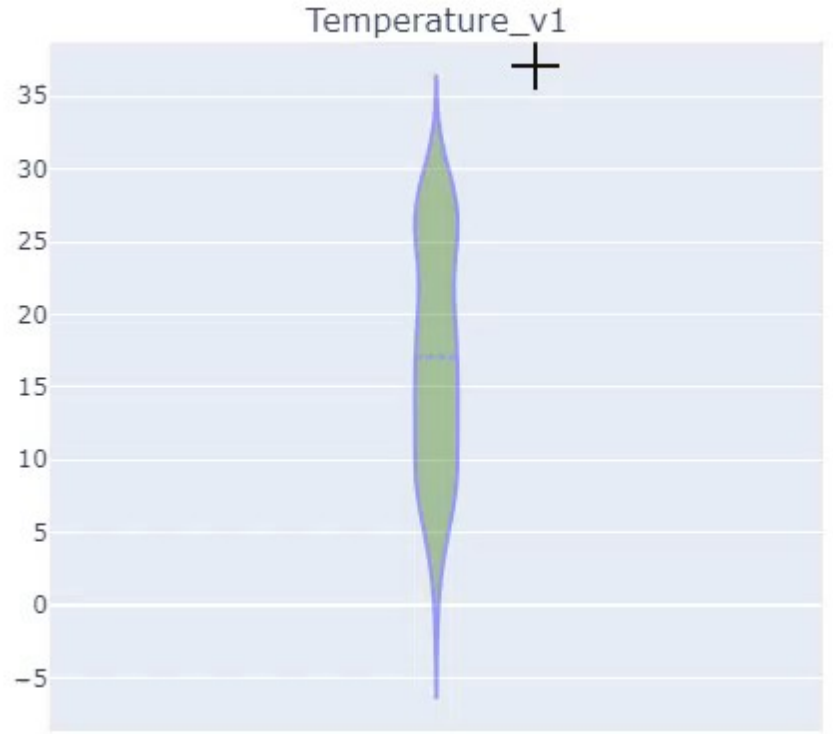
Violin plots

```
from plotly.subplots import make_subplots

import plotly.graph_objects as go

# Specify the columns for which you want to create violin plots

variables= ["Energy_Consumption", "Energy_Generation", "Temperature_v1",
"Relative_Humidity_v1", 'Wind_Speed_v1', 'Clouds_v1']

variables1 = ["Energy_Consumption", "Energy_Generation", "Temperature_v1"]

variables2 = [ "Relative_Humidity_v1", 'Wind_Speed_v1', 'Clouds_v1']

# Create a subplot with multiple violin plots

fig = make_subplots(rows=2, cols=len(variables1),

                    subplot_titles=variables)


for i, variable in enumerate(variables1, start=1):

    violin_fig = px.violin(df_cons_weh, y=variable, box=False,
 facet_col_spacing=0.7, width=900)

    violin_trace = violin_fig['data'][0]

    fig.add_trace(violin_trace, row=1, col=i)


for i, variable in enumerate(variables2, start=1):

    violin_fig = px.violin(df_cons_weh, y=variable, box=False,
facet_col_spacing=0.7,width=900)

    violin_trace = violin_fig['data'][0]

    fig.add_trace(violin_trace, row=2, col=i)
```

```
# Update traces outside of the loop for individual color settings
fig.update_traces(
    fillcolor='green',
    line_color='blue',
    marker_line_outliercolor='black',
    box_fillcolor='red',
    opacity=0.4,
    meanline_visible=True,box_visible=False

)

fig.update_layout(
    boxgap=0, boxgroupgap=0,
    height=1100,  # Adjust the height as needed
    width=1500,  # Adjust the width as needed
    title_text="Violin Plots for Multiple Variables",
    showlegend=False
)
fig.show()
```

Violin plots





Boosting Research for a Smart and Carbon Neutral Built Environment with Digital Twins – **SmartWins**

Project Partners

ktu kaunas university of technology 1922

CERTH CENTRE FOR RESEARCH & TECHNOLOGY HELLAS

POLITECNICO MILANO 1863

Contecht

INNOTROPE

Funded by the European Union

This project has received funding from the European Union's Horizon research and innovation programme under grant agreement No 101078997

SmartWins