

[Documentation](#)

search

Search

- [rocket launch](#)

[Get started](#)

- [Installation](#)
add
- [Fundamentals](#)
add
- [First steps](#)
add
- [code](#)

[Develop](#)

- [Concepts](#)
add
- [API reference](#)
remove
 - PAGE ELEMENTS

 - [Write and magic](#)
add
 - [Text elements](#)
add
 - [Data elements](#)
add
 - [Chart elements](#)
remove
 - SIMPLE

 - [st.area_chart](#)
 - [st.bar_chart](#)
 - [st.line_chart](#)
 - [st.map](#)
 - [st.scatter_chart](#)
 - ADVANCED

 - [st.altair_chart](#)
 - [st.bokeh_chart](#)
 - [st.graphviz_chart](#)
 - [st.plotly_chart](#)
 - [st.pydeck_chart](#)
 - [st.pyplot](#)
 - [st.vega_lite_chart](#)
- [Input widgets](#)
add
- [Media elements](#)
add
- [Layouts and containers](#)
add

- [Chat elements](#)
add
 - [Status elements](#)
add
 - [Third-party components](#)*open in new*
 - APPLICATION LOGIC
-

- [Navigation and pages](#)
add
 - [Execution flow](#)
add
 - [Caching and state](#)
add
 - [Connections and secrets](#)
add
 - [Custom components](#)
add
 - [Utilities](#)
add
 - [Configuration](#)
add
 - TOOLS
-

- [App testing](#)
add
- [Command line](#)
add

- [Tutorials](#)
add
- [Quick reference](#)
add

- [web_asset](#)

[Deploy](#)

- [Concepts](#)
add
- [Streamlit Community Cloud](#)
add
- [Snowflake](#)
- [Other platforms](#)
add

- [school](#)

[Knowledge base](#)

- [FAQ](#)
- [Installing dependencies](#)
- [Deployment issues](#)

- [Home/](#)
- [Develop/](#)
- [API reference/](#)
- [Chart elements/](#)
- [st.plotly_chart](#)

st.plotly_chart



Display an interactive Plotly chart.

[Plotly](#) is a charting library for Python. The arguments to this function closely follow the ones for Plotly's `plot()` function.

To show Plotly charts in Streamlit, call `st.plotly_chart` wherever you would call Plotly's `py.plot` or `py.iplot`.

Important

You must install `plotly` to use this command. Your app's performance may be enhanced by installing `orjson` as well.

Function signature[\[source\]](#)

```
st.plotly_chart(figure_or_data, use_container_width=False, *, theme="streamlit", key=None, on_select="ignore",
                 selection_mode=('points', 'box', 'lasso'), **kwargs)
```

Parameters

figure_or_data (plotly.graph_objs.Figure, plotly.graph_objs.Data, or dict/list of plotly.graph_objs.Figure/Data)	The Plotly <code>Figure</code> or <code>Data</code> object to render. See https://plot.ly/python/ for examples of graph descriptions.
use_container_width (bool)	Whether to override the figure's native width with the width of the parent container. If <code>use_container_width</code> is <code>False</code> (default), Streamlit sets the width of the chart to fit its contents according to the plotting library, up to the width of the parent container. If <code>use_container_width</code> is <code>True</code> , Streamlit sets the width of the figure to match the width of the parent container.
theme ("streamlit" or None)	The theme of the chart. If <code>theme</code> is "streamlit" (default), Streamlit uses its own design default. If <code>theme</code> is <code>None</code> , Streamlit falls back to the default behavior of the library.
key (str)	An optional string to use for giving this element a stable identity. If <code>key</code> is <code>None</code> (default), this element's identity will be determined based on the values of the other parameters. Additionally, if selections are activated and <code>key</code> is provided, Streamlit will register the key in Session State to store the selection state. The selection state is read-only.
on_select ("ignore" or "rerun" or callable)	How the figure should respond to user selection events. This controls whether or not the figure behaves like an input widget. <code>on_select</code> can be one of the following: <ul style="list-style-type: none"> "ignore" (default): Streamlit will not react to any selection events in the chart. The figure will not behave like an input widget.

Returns

(element or dict)	If <code>on_select</code> is "ignore" (default), this command returns an internal placeholder for the chart element. Otherwise, this command returns a dictionary-like object that supports both key and attribute notation. The attributes are described by the <code>PlotlyState</code> dictionary schema.
-------------------	--

Function signature[\[source\]](#)

```
st.plotly_chart(figure_or_data, use_container_width=False, *, theme="streamlit", key=None, on_select="ignore",  
               selection_mode=('points', 'box', 'lasso'), **kwargs)
```

- "rerun": Streamlit will rerun the app when the user selects data in the chart. In this case, `st.plotly_chart` will return the selection data as a dictionary.
- A callable: Streamlit will rerun the app and execute the callable as a callback function before the rest of the app. In this case, `st.plotly_chart` will return the selection data as a dictionary.

The selection mode of the chart. This can be one of the following:

`selection_mode` ("points", "box", "lasso" or an Iterable of these)

- "points": The chart will allow selections based on individual data points.
- "box": The chart will allow selections based on rectangular areas.
- "lasso": The chart will allow selections based on freeform areas.
- An Iterable of the above options: The chart will allow selections based on the modes specified.

All selections modes are activated by default.

**kwargs (null)

Any argument accepted by Plotly's `plot()` function.

Returns

(element or dict)

If `on_select` is "ignore" (default), this command returns an internal placeholder for the chart element. Otherwise, this command returns a dictionary-like object that supports both key and attribute notation. The attributes are described by the `PlotlyState` dictionary schema.

Example

The example below comes straight from the examples at <https://plot.ly/python>. Note that `plotly.figure_factory` requires `scipy` to run.

```
import streamlit as st
import numpy as np
import plotly.figure_factory as ff

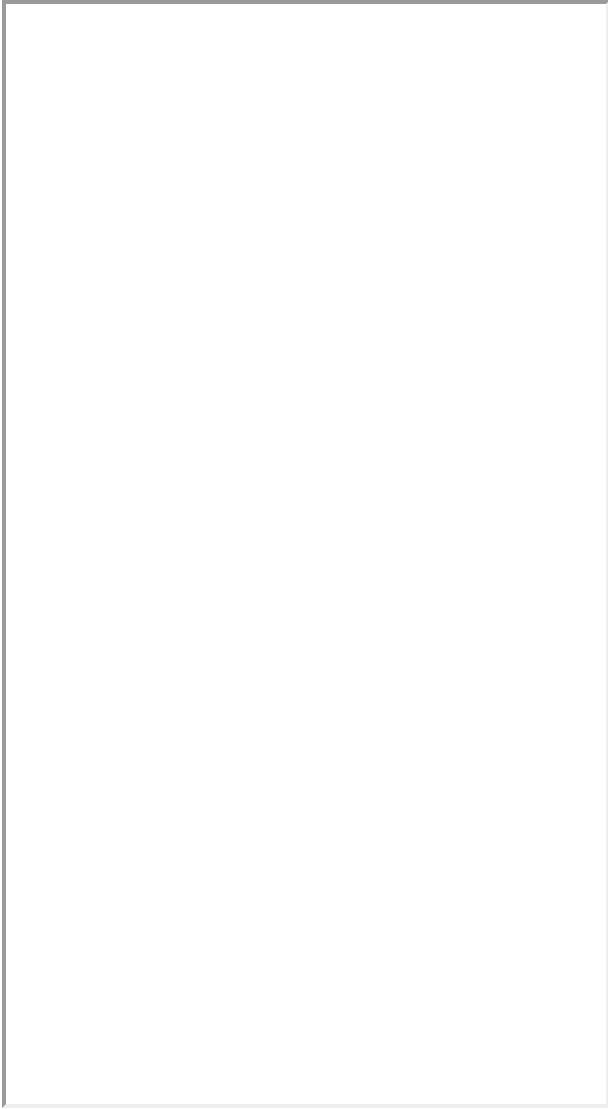
# Add histogram data
x1 = np.random.randn(200) - 2
x2 = np.random.randn(200)
x3 = np.random.randn(200) + 2

# Group data together
hist_data = [x1, x2, x3]

group_labels = ['Group 1', 'Group 2', 'Group 3']

# Create distplot with custom bin_size
fig = ff.create_distplot(
    hist_data, group_labels, bin_size=[.1, .25, .5])

# Plot!
st.plotly_chart(fig, use_container_width=True)
```



[Built with Streamlit !\[\]\(8af806fb1314382d09bc5ec5b767526c_img.jpg\)](#)
[Fullscreen open in new](#)

Chart selections



PlotlyState



Streamlit Version 

The schema for the Plotly chart event state.

The event state is stored in a dictionary-like object that supports both key and attribute notation. Event states cannot be programmatically changed or set through Session State.

Only selection events are supported at this time.

Attributes

selection (dict)	The state of the <code>on_select</code> event. This attribute returns a dictionary-like object that supports both key and attribute notation. The attributes are described by the <code>PlotlySelectionState</code> dictionary schema.
------------------	--

Example

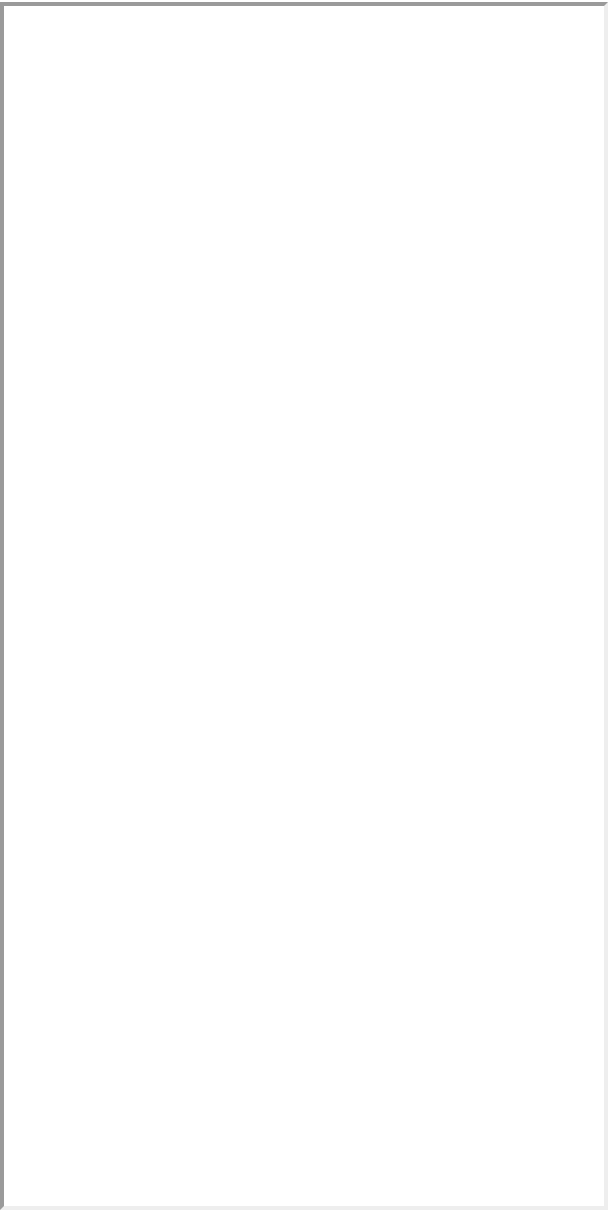
Try selecting points by any of the three available methods (direct click, box, or lasso). The current selection state is available through Session State or as the output of the chart function.


```
import streamlit as st
import plotly.express as px

df = px.data.iris() # iris is a pandas DataFrame
fig = px.scatter(df, x="sepal_width", y="sepal_length")

event = st.plotly_chart(fig, key="iris", on_select="rerun")

event
```



[Built with Streamlit](#) 
[Fullscreen open in new](#)

PlotlySelectionState



Streamlit Version Version 1.41.0 ▼

The schema for the Plotly chart selection state.

The selection state is stored in a dictionary-like object that supports both key and attribute notation. Selection states cannot be programmatically changed or set through Session State.

Attributes

<code>points</code> (list[dict[str, Any]])	The selected data points in the chart, including the data points selected by the box and lasso mode. The data includes the values associated to each point and a point index used to populate <code>point_indices</code> . If additional information has been assigned to your points, such as size or legend group, this is also included.
<code>point_indices</code> (list[int])	The numerical indices of all selected data points in the chart. The details of each identified point are included in <code>points</code> .
<code>box</code> (list[dict[str, Any]])	The metadata related to the box selection. This includes the coordinates of the selected area.
<code>lasso</code> (list[dict[str, Any]])	The metadata related to the lasso selection. This includes the coordinates of the selected area.

Example

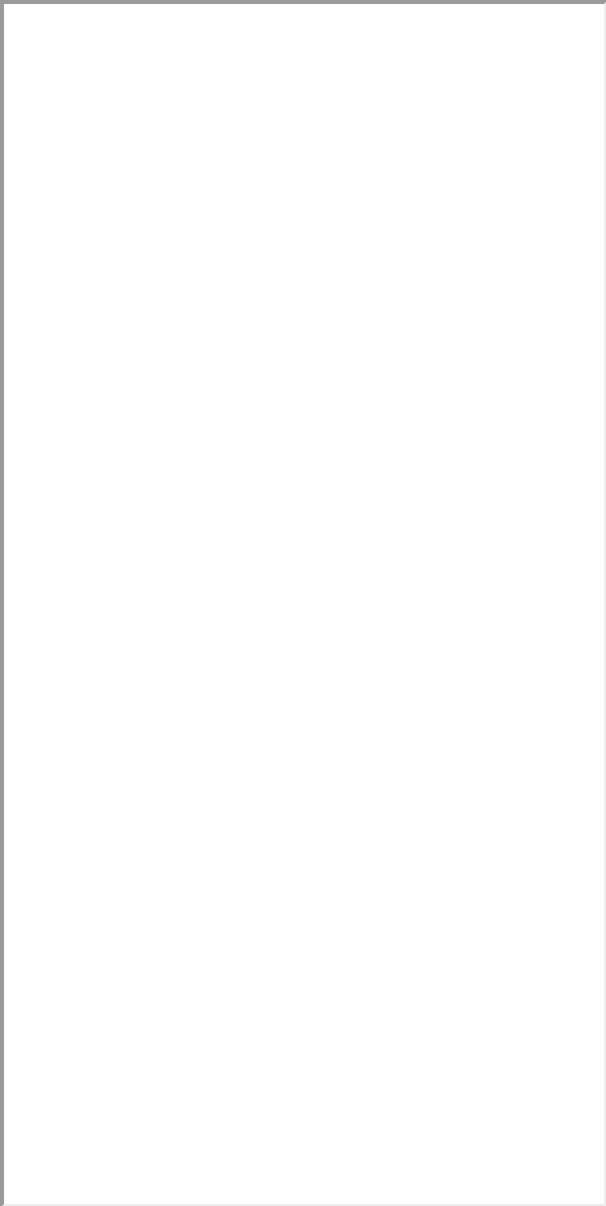
When working with more complicated graphs, the `points` attribute displays additional information. Try selecting points in the following example:

```
import streamlit as st
import plotly.express as px

df = px.data.iris()
fig = px.scatter(
    df,
    x="sepal_width",
    y="sepal_length",
    color="species",
    size="petal_length",
    hover_data=["petal_width"],
)

event = st.plotly_chart(fig, key="iris", on_select="rerun")

event.selection
```



[Built with Streamlit !\[\]\(d263118e0bfd47dc6bc704167d936b83_img.jpg\)](#)
[Fullscreen open in new](#)

This is an example of the selection state when selecting a single point:

```
{
  "points": [
    {
      "curve_number": 2,
      "point_number": 9,
      "point_index": 9,
      "x": 3.6,
      "y": 7.2,
      "customdata": [
        2.5
      ],
      "marker_size": 6.1,
      "legendgroup": "virginica"
    }
  ],
  "point_indices": [
    9
  ],
  "box": [],
  "lasso": []
}
```


Theming



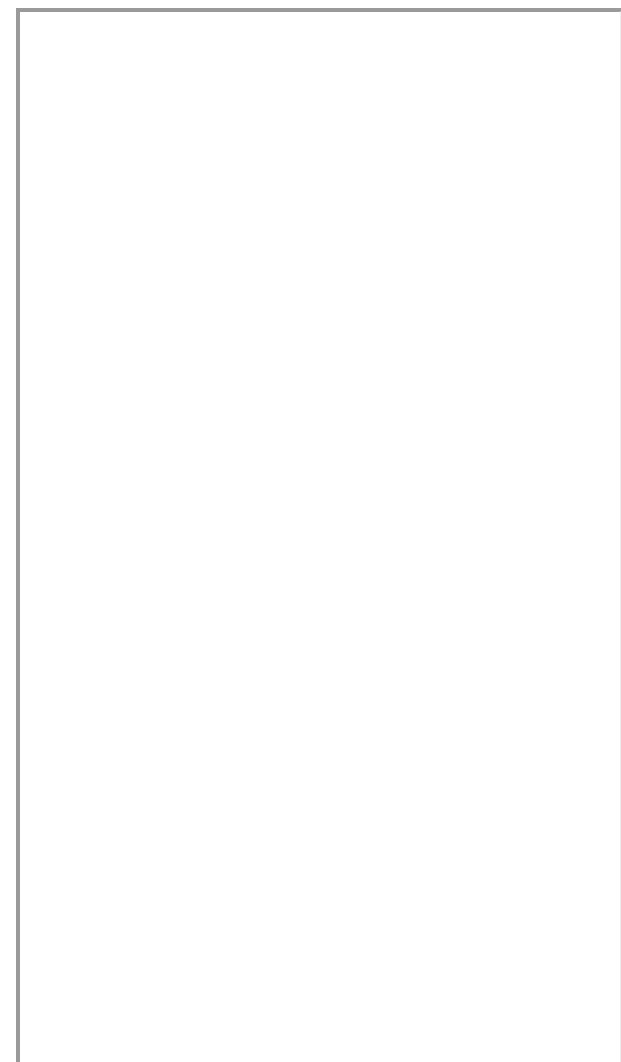
Plotly charts are displayed using the Streamlit theme by default. This theme is sleek, user-friendly, and incorporates Streamlit's color palette. The added benefit is that your charts better integrate with the rest of your app's design.

The Streamlit theme is available from Streamlit 1.16.0 through the `theme="streamlit"` keyword argument. To disable it, and use Plotly's native theme, use `theme=None` instead.

Let's look at an example of charts with the Streamlit theme and the native Plotly theme:

```
import plotly.express as px
import streamlit as st
df = px.data.gapminder()
fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp", size="pop", color="continent",
                hover_name="country", log_x=True, size_max=60, )
tab1, tab2 = st.tabs(["Streamlit theme (default)", "Plotly native theme"])
with tab1: # Use the Streamlit theme. # This is the default. So you can also omit the theme argument.
    st.plotly_chart(fig, theme="streamlit", use_container_width=True)
with tab2: # Use the native Plotly theme.
    st.plotly_chart(fig, theme=None, use_container_width=True)
```

Click the tabs in the interactive app below to see the charts with the Streamlit theme enabled and disabled.



[Built with Streamlit](#) 

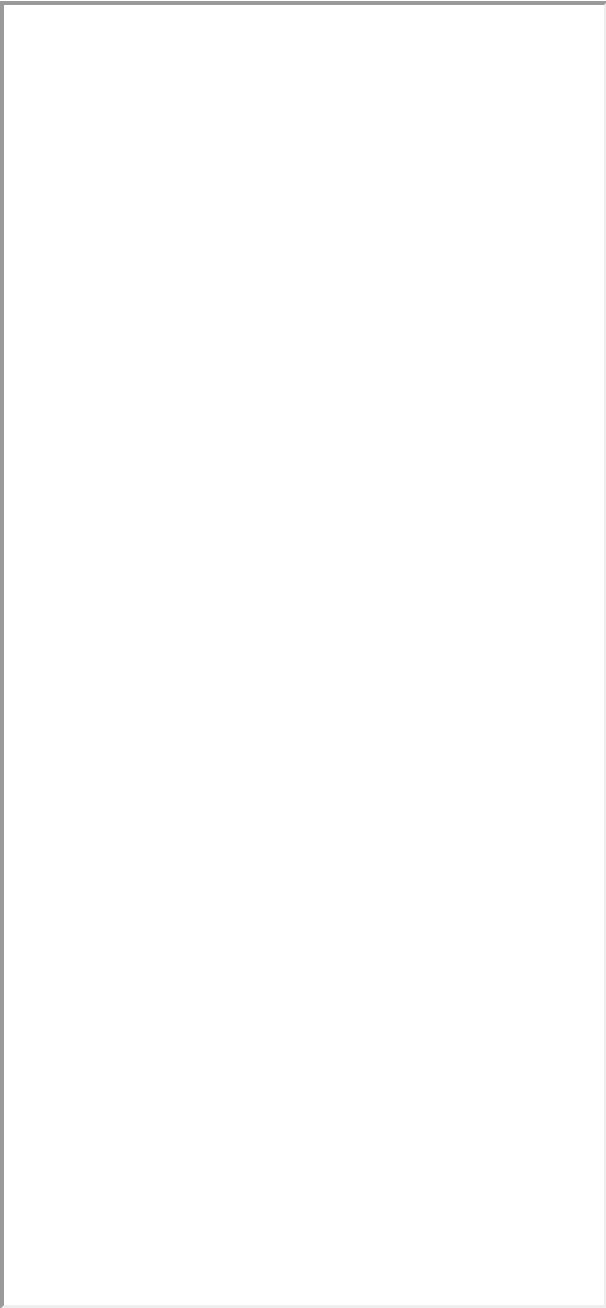
[Fullscreen open in new](#)

If you're wondering if your own customizations will still be taken into account, don't worry! You can still make changes to your chart configurations. In other words, although we now enable the Streamlit theme by default, you can overwrite it with custom colors or fonts. For example, if you want a chart line to be green instead of the default red, you can do it!

Here's an example of an Plotly chart where a custom color scale is defined and reflected:

```
import plotly.express as px
import streamlit as st
st.subheader("Define a custom colorscale")
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="sepal_length", color_continuous_scale="reds", )
tab1, tab2 = st.tabs(["Streamlit theme (default)", "Plotly native theme"])
with tab1: st.plotly_chart(fig, theme="streamlit", use_container_width=True)
with tab2: st.plotly_chart(fig, theme=None, use_container_width=True)
```

Notice how the custom color scale is still reflected in the chart, even when the Streamlit theme is enabled 📌



[Built with Streamlit](#) 📌
[Fullscreen open in new](#)

For many more examples of Plotly charts with and without the Streamlit theme, check out the [plotly.streamlit.app](#).

← [Previous: st.graphviz chart](#) [Next: st.pydeck chart](#) →
[forum](#)

Still have questions?

Our [forums](#) are full of helpful information and Streamlit experts.

