st.vega_lite_chart

Streamlit Version Version 1.41.0

Display a chart using the Vega-Lite library.

Vega-Lite is a high-level grammar for defining interactive graphics.

Function signature[source]

st.vega_lite_chart(data=None, spec=None, *, use_container_width=False, theme="streamlit", key=None, on_select="ignore", selection_mode=None, **kwargs)

Parameters

data (Anything supported by st.dataframe)	Either the data to be plotted or a Vega-Lite spec containing the data (which more closely follows the Vega-Lite API).
spec (dict or None)	The Vega-Lite spec for the chart. If `spec` is `None` (default), Streamlit uses the spec passed in `data`. You cannot pass a spec to both `data` and `spec`. See https://vega.github.io/vega-lite/docs/ for more info.
use_container_width (bool)	Whether to override the figure's native width with the width of the parent container. If `use_container_width` is `False` (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 `use_container_width` is `True`, 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 `theme` is `"streamlit"` (default), Streamlit uses its own design default. If `theme` is `None`, 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 `key` is `None` (default), this element's identity will be determined based on the values of the other parameters. Additionally, if selections are activated and `key` is provided, Streamlit will register the key in Session State to store the selection state. The selection state is read-only.
on_select ("ignore", "rerun", or callable)	How the figure should respond to user selection events. This controls whether or not the figure behaves like an input widget. `on_select` can be one of the following: • `"ignore"` (default): Streamlit will not react to any selection events in the chart. The figure will not behave like an input widget. • `"rerun"`: Streamlit will rerun the app when the user selects data in the chart. In this case, `st.vega_lite_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.vega_lite_chart` will return the selection data as a dictionary.

Returns

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

Function signature[source]

st.vega_lite_chart(data=None, spec=None, *, use_container_width=False, theme="streamlit", key=None, on_select="ignore", selection_mode=None, **kwargs)

	To use selection events, the Vega-Lite spec defined in `data` or `spec` must include selection parameters from the the charting library. To learn about defining interactions in Vega-Lite, see Dynamic Behaviors with Parameters in Vega-Lite's documentation.
selection_mode (str or Iterable of str)	The selection parameters Streamlit should use. If `selection_mode` is `None` (default), Streamlit will use all selection parameters defined in the chart's Vega-Lite spec. When Streamlit uses a selection parameter, selections from that parameter will trigger a rerun and be included in the selection state. When Streamlit does not use a selection parameter, selections from that parameter will not trigger a rerun and not be included in the selection state. Selection parameters are identified by their `name` property.
**kwargs (any)	The Vega-Lite spec for the chart as keywords. This is an alternative to `spec`.

Returns

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

Example

```
import streamlit as st
import pandas as pd
import numpy as np

chart_data = pd.DataFrame(np.random.randn(200, 3), columns=["a", "b", "c"])

st.vega_lite_chart(
   chart_data,
   {
       "mark": {"type": "circle", "tooltip": True},
       "encoding": {
           "x": {"field": "a", "type": "quantitative"},
           "y": {"field": "b", "type": "quantitative"},
           "size": {"field": "c", "type": "quantitative"},
           "color": {"field": "c", "type": "quantitative"},
       },
   },
)
```

Built with Streamlit 🎈
Fullscreen *open_in_new*

Examples of Vega-Lite usage without Streamlit can be found at https://vega.github.io/vega-lite/examples/. Most of those can be easily translated to the syntax shown above.

Chart selections

VegaLiteState

Streamlit Version Version 1.41.0

The schema for the Vega-Lite 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 `on_select` event. This attribute returns a dictionary-like object that supports both key and attribute notation. The name of each Vega-Lite selection parameter becomes an attribute in the `selection` dictionary. The format of the data within each attribute is determined by the selection parameter definition within Vega-Lite.

Examples

The following two examples have equivalent definitions. Each one has a point and interval selection parameter include in the chart definition. The point selection parameter is named `"point_selection"`. The interval or box selection parameter is named `"interval_selection"`.

The follow example uses `st.altair_chart`:

```
import streamlit as st
import pandas as pd
import numpy as np
import altair as alt

if "data" not in st.session_state:
    st.session_state.data = pd.DataFrame(
        np.random.randn(20, 3), columns=["a", "b", "c"]
    )
df = st.session_state.data

point_selector = alt.selection_point("point_selection")
interval_selector = alt.selection_interval("interval_selection")
chart = (
    alt.Chart(df)
    .mark_circle()
    .encode(
        x="a",
        y="b",
        size="c",
        color="c",
        tooltip=["a", "b", "c"],
        fillOpacity=alt.condition(point_selector, alt.value(1), alt.value(0.3)),
    )
    .add_params(point_selector, interval_selector)
)

event = st.altair_chart(chart, key="alt_chart", on_select="rerun")

event
```

The following example uses `st.vega_lite_chart`:

```
import streamlit as st
import pandas as pd
import numpy as np

if "data" not in st.session_state:
    st.session_state.data = pd.DataFrame(
        np.random.randn(20, 3), columns=["a", "b", "c"]
    )

spec = {
    "mark": {"type": "circle", "tooltip": True},
    "params": [
        {"name": "interval_selection", "select": "interval"},
        {"name": "point_selection", "select": "point"},
    ],
    "encoding": {
        "x": {"field": "a", "type": "quantitative"},
        "y": {"field": "b", "type": "quantitative"},
        "size": {"field": "c", "type": "quantitative"},
        "color": {"field": "c", "type": "quantitative"},
        "fillOpacity": {
            "condition": {"param": "point_selection", "value": 1},
            "value": 0.3,
        },
    },
}

event = st.vega_lite_chart(
    st.session_state.data, spec, key="vega_chart", on_select="rerun"
)
```

`event`

Try selecting points in this interactive example. When you click a point, the selection will appear under the attribute, `"point_selection"`, which is the name given to the point selection parameter. Similarly, when you make an interval selection, it will appear under the attribute `"interval_selection"`. You can give your selection parameters other names if desired.

If you hold `Shift` while selecting points, existing point selections will be preserved. Interval selections are not preserved when making additional selections.

Built with Streamlit 🎈
Fullscreen *open_in_new*

element.add_rows

Streamlit Version Version 1.41.0

Concatenate a dataframe to the bottom of the current one.

Function signature[source]

element.add_rows(data=None, **kwargs)

Parameters

data (pandas.DataFrame, pandas.Styler, pyarrow.Table, numpy.ndarray, pyspark.sql.DataFrame, snowflake.snowpark.dataframe.DataFrame, Iterable, dict, or None)	Table to concat. Optional.
**kwargs (pandas.DataFrame, numpy.ndarray, Iterable, dict, or None)	The named dataset to concat. Optional. You can only pass in 1 dataset (including the one in the data parameter).

Example

```
import streamlit as st
import pandas as pd
import numpy as np

df1 = pd.DataFrame(
    np.random.randn(50, 20), columns=("col %d" % i for i in range(20))
)

my_table = st.table(df1)

df2 = pd.DataFrame(
    np.random.randn(50, 20), columns=("col %d" % i for i in range(20))
)

my_table.add_rows(df2)
# Now the table shown in the Streamlit app contains the data for
# df1 followed by the data for df2.
```

You can do the same thing with plots. For example, if you want to add more data to a line chart:

```
# Assuming df1 and df2 from the example above still exist...
my_chart = st.line_chart(df1)
my_chart.add_rows(df2)
# Now the chart shown in the Streamlit app contains the data for
# df1 followed by the data for df2.
```

And for plots whose datasets are named, you can pass the data with a keyword argument where the key is the name:

```
my_chart = st.vega_lite_chart(
    {
        "mark": "line",
        "encoding": {"x": "a", "y": "b"},
        "datasets": {
            "some_fancy_name": df1,  # <-- named dataset
        },
        "data": {"name": "some_fancy_name"},
    }
)
my_chart.add_rows(some_fancy_name=df2)  # <-- name used as keyword
```

Theming

Vega-Lite 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 Vega-Lite's native theme, use `theme=None` instead.

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

```
import streamlit as st from vega_datasets import data source = data.cars() chart = { "mark": "point",
"encoding": { "x": { "field": "Horsepower", "type": "quantitative", }, "y": { "field":
"Miles_per_Gallon", "type": "quantitative", }, "color": {"field": "Origin", "type": "nominal"},
"shape": {"field": "Origin", "type": "nominal"}, }, } tab1, tab2 = st.tabs(["Streamlit theme
(default)", "Vega-Lite native theme"]) with tab1: # Use the Streamlit theme. # This is the default. So
you can also omit the theme argument. st.vega_lite_chart( source, chart, theme="streamlit",
use_container_width=True ) with tab2: st.vega_lite_chart( source, chart, 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!

←Previous: st.pyplotNext: Input widgets→
forum

Still have questions?

Our forums are full of helpful information and Streamlit experts.

© 2025 Snowflake Inc. Cookie policy

forum Ask AI