- 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.scatter_chart

st.scatter_chart

Streamlit Version Version 1.41.0

Display a scatterplot chart.

This is syntax-sugar around `st.altair_chart`. The main difference is this command uses the data's own column and indices to figure out the chart's Altair spec. As a result this is easier to use for many "just plot this" scenarios, while being less customizable.

If `st.scatter_chart` does not guess the data specification correctly, try specifying your desired chart using `st.altair_chart`.

Function signature[source]

st.scatter_chart(data=None, *, x=None, y=None, x_label=None, y_label=None, color=None, size=None, width=None, height=None, use_container_width=True)

Parameters	
data (Anything supported by st.dataframe)	Data to be plotted.
x (str or None)	Column name or key associated to the x-axis data. If `x` is `None` (default), Streamlit uses the data index for the x-axis values.
y (str, Sequence of str, or None)	Column name(s) or key(s) associated to the y-axis data. If this is `None` (default), Streamlit draws the data of all remaining columns as data series. If this is a `Sequence` of strings, Streamlit draws several series on the same chart by melting your wide-format table into a long-format table behind the scenes.
x_label (str or None)	The label for the x-axis. If this is `None` (default), Streamlit will use the column name specified in `x` if available, or else no label will be displayed.
y_label (str or None)	The label for the y-axis. If this is `None` (default), Streamlit will use the column name(s) specified in `y` if available, or else no label will be displayed.
color (str, tuple, Sequence of str, Sequence of tuple, or None)	The color of the circles representing each datapoint. This can be: • None, to use the default color. • A hex string like "#ffaa00" or "#ffaa0088". • An RGB or RGBA tuple with the red, green, blue, and alpha components specified as ints from 0 to 255 or floats from 0.0 to 1.0. • The name of a column in the dataset where the color of that datapoint will come from. If the values in this column are in one of the color formats above (hex string or color tuple), then that color will be used. Otherwise, the color will be automatically picked from the default palette. For example: if the dataset has 1000 rows, but this column only contains the values "adult", "child", and "baby", then those 1000 datapoints be shown using three colors

	from the default palette. But if this column only contains floats or ints, then those 1000 datapoints will be shown using a colors from a continuous color gradient. Finally, if this column only contains the values "#ffaa00", "#f0f", "#0000ff", then then each of those 1000 datapoints will be assigned "#ffaa00", "#f0f", or "#0000ff" as appropriate. If the dataframe is in wide format (that is, y is a Sequence of columns), this can also be: • A list of string colors or color tuples to be used for each of the series in the chart. This list should have the same length as the number of y values (e.g. `color=["#fd0", "#f0f", "#04f"]` for three series).
size (str, float, int, or None)	The size of the circles representing each point. This can be: • A number like 100, to specify a single size to use for all datapoints. • The name of the column to use for the size. This allows each datapoint to be represented by a circle of a different size.
width (int or None)	Desired width of the chart expressed in pixels. If `width` is `None` (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 `width` is greater than the width of the parent container, Streamlit sets the chart width to match the width of the parent container. To use `width`, you must set `use_container_width=False`.
height (int or None)	Desired height of the chart expressed in pixels. If `height` is `None` (default), Streamlit sets the height of the chart to fit its contents according to the plotting library.
use_container_width (bool)	Whether to override `width` with the width of the parent container. If `use_container_width` is `True` (default), Streamlit sets the width of the chart to match the width of the parent container. If `use_container_width` is `False`, Streamlit sets the chart's width according to `width`.

Examples

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

chart_data = pd.DataFrame(np.random.randn(20, 3), columns=["a", "b", "c"])

st.scatter_chart(chart_data)
```

Built with Streamlit 🎈
Fullscreen *open_in_new*

You can also choose different columns to use for x and y, as well as set the color dynamically based on a 3rd column (assuming your dataframe is in long format):

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

chart_data = pd.DataFrame(
    np.random.randn(20, 3), columns=["col1", "col2", "col3"]
)
chart_data["col4"] = np.random.choice(["A", "B", "C"], 20)

st.scatter_chart(
    chart_data,
    x="col1",
    y="col2",
    color="col4",
    size="col3",
)
```

Built with Streamlit 🎈

Fullscreen *open_in_new*

Finally, if your dataframe is in wide format, you can group multiple columns under the y argument to show multiple series with different colors:

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

chart_data = pd.DataFrame(
    np.random.randn(20, 4), columns=["col1", "col2", "col3", "col4"]
)

st.scatter_chart(
    chart_data,
    x="col1",
    y=["col2", "col3"],
    size="col4",
    color=["#FF0000", "#0000FF"],  # Optional
)
```

[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
```

←Previous: st.mapNext: st.altair_chart→

forum

Still have questions?

Our forums are full of helpful information and Streamlit experts.