Plotly Studio: Transform any dataset into an interactive data application in minutes with AI. **Sign up for early access now.**

Dash Vega Components

Dash Vega component brings the power of Dash to the popular Vega-Altair graphing library. The component is used to display **Vega-Altair charts** in your Dash app.

Install the relevant dependencies with:

```
pip install dash-vega-components altair
```

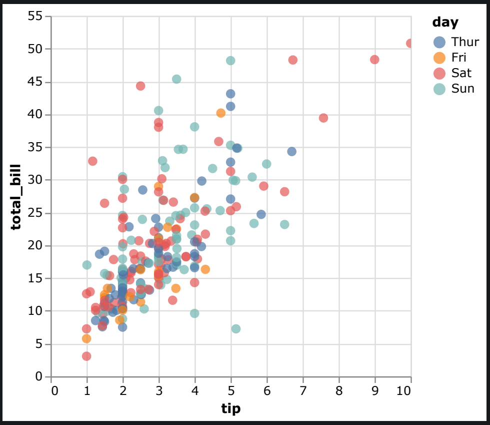
Examples in this chapter require Altair 5 or later.

Display Vega-Altair Chart in Dash

To incorporate a Vega-Altair chart into the Dash layout, convert the chart into a dictionary with `chart.to_dict()` and assign it to the `spec` prop of `dvc.Vega`.

```
from dash import Dash, html
import altair as alt
import dash_vega_components as dvc
import plotly.express as px

df = px.data.tips()
chart = (
    alt.Chart(df)
    .mark_circle(size=60)
    .encode(
        x="tip",
        y="total_bill",
        color=alt.Color("day").scale(domain=["Thur", "Fri", "Sat", "Sun"]),
        tooltip=["day", "tip", "total_bill"],
    )
    .interactive()
)


app = Dash()
app.layout = html.Div(
    [
        html.H1("Vega-Altair Chart in a Dash App"),
        dvc.Vega(
            id="altair-chart",
            opt={"renderer": "svg", "actions": False},
            spec=chart.to_dict(),
        ),
    ]
)


if __name__ == "__main__":
    app.run(debug=True)
```

Vega-Altair Chart in a Dash App



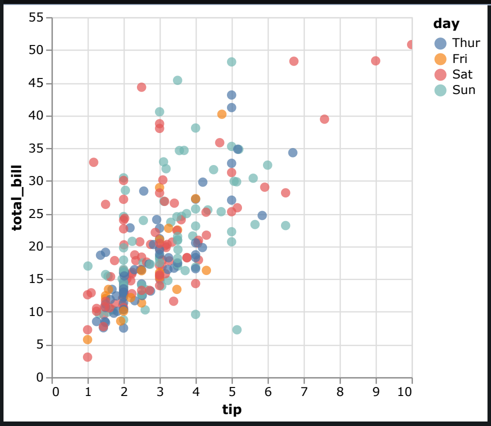
Add a Dropdown

Let's create an interactive Dash app by allowing the user to filter the dataset by day, via a dropdown. In the example below, the callback's Input is used to register the dropdown's day value selected. Then, inside the callback function, the dataset is filtered and the Vega-Altair chart is built. Finally, the chart is assigned to the `spec` prop of `dvc.Vega` by returning it at the end of the callback function.

```python
from dash import Dash, Input, Output, callback, dcc, html
import altair as alt
import dash_vega_components as dvc
import plotly.express as px

app = Dash()
app.layout = html.Div(
    [
        html.H1("Vega-Altair Chart in a Dash App"),
        dcc.Dropdown(
            options=["All", "Thur", "Fri", "Sat", "Sun"],
            value="All",
            id="origin-dropdown",
        ),
        dvc.Vega(
            id="altair-d-chart", opt={"renderer": "svg", "actions": False}, spec={}
        ),
    ]
)


@callback(
    Output(component_id="altair-d-chart", component_property="spec"),
    Input(component_id="origin-dropdown", component_property="value"),
)
def display_altair_chart(day_chosen):
    df = px.data.tips()

    if day_chosen != "All":
        df = df[df["day"] == day_chosen]

    chart = (
        alt.Chart(df)
        .mark_circle(size=60)
        .encode(
            x="tip",
            y="total_bill",
            color=alt.Color("day").scale(domain=["Thur", "Fri", "Sat", "Sun"]),
            tooltip=["day", "tip", "total_bill"],
        )
        .interactive()
    )
    return chart.to_dict()


if __name__ == "__main__":
    app.run(debug=True)
```

Vega-Altair Chart in a Dash App

All



Interact with Click Data

Once you have incorporated the Vega-Altair scatter plot into the Dash app, you might want to enhance the interactivity of the app by extracting the data of the scatter markers that were clicked by the user. For example, you could take the clicked data and build a separate graph or data table with it.

To work with click data, you need to configure the parameters of the Vega-Altair chart. Parameters are the basic building blocks used to make an Vega-Altair chart interactive. They can either be simple variables or more complex selections that map user input (for example, mouse clicks and drags) to data queries. In Vega, these are called Signals, but you can think of Signals as synonymous with parameters. To read more about the various Parameters and examples, see the official **Vega-Altair user guide**.

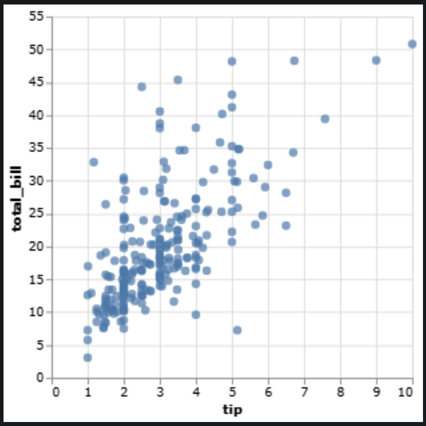
You can trigger a Dash callback based on changes in any parameter within an Vega-Altair chart:

- Specify a `name` when defining the parameter. For example, `alt.selection_point(name="my_param")`
- Add the parameter name to the `signalsToObserve` prop of the Vega component: `dvc.Vega(id="chart1", signalsToObserve=["my_param"])`. If you want to observe all signals, you can also pass `signalsToObserve=["all"]`
- Use `Input("chart1", "signalData")` in your callback to access the value of `"my_param"` and react to changes

```
from dash import Dash, Input, Output, callback, dcc, html
import altair as alt
import dash_vega_components as dvc
import plotly.express as px
import json

df = px.data.tips()
chart = (
    alt.Chart(df)
    .mark_circle(size=50)
    .encode(x="tip", y="total_bill")
    .add_params(
        alt.selection_point(fields=["tip", "total_bill"], name="selected_points")
    )
)
```

```python
app = Dash()
app.layout = html.Div(
    [
        html.H1("Interact with Click Data"),
        dvc.Vega(
            id="chart1", signalsToObserve=["selected_points"], spec=chart.to_dict()
        ),
        dcc.Markdown(id="chart1-params"),
    ]
)


@callback(
    Output("chart1-params", "children"),
    Input("chart1", "signalData"),
    prevent_initial_call=True,
)
def display_altair_width_params(params):
    return "```json\n" + json.dumps(params, indent=2) + "\n```"


if __name__ == "__main__":
    app.run(debug=True)
```

Interact with Click Data



```
{
  "selected_points": {}
}
```

Interact with Hover Data

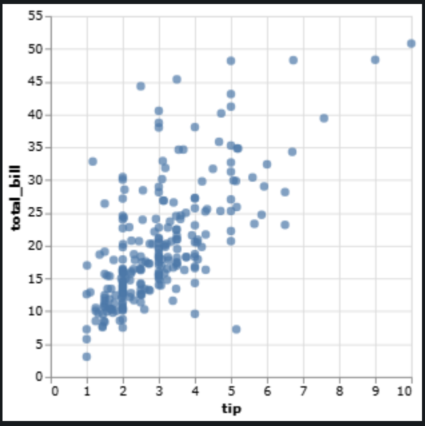
To extract the hover data of an Vega-Altair graph, add the `on="mouseover"` when defining the parameter. `alt.selection_point(fields=["tip", "total_bill"], name="selected_points", on="mouseover"))`

Here is the complete example:

```python
from dash import Dash, Input, Output, callback, dcc, html
import altair as alt
import dash_vega_components as dvc
import plotly.express as px
import json


df = px.data.tips()
chart = (
    alt.Chart(df)
```

```
        .mark_circle(size=50)
        .encode(x="tip", y="total_bill")
        .add_params(
            alt.selection_point(
                fields=["tip", "total_bill"], name="selected_points", on="mouseover"
            )
        )
    )


    app = Dash()
    app.layout = html.Div(
        [
            html.H1("Interact with Click Data"),
            dvc.Vega(
                id="chart", signalsToObserve=["selected_points"], spec=chart.to_dict()
            ),
            dcc.Markdown(id="chart-params"),
        ]
    )


    @callback(
        Output("chart-params", "children"),
        Input("chart", "signalData"),
        prevent_initial_call=True,
    )
    def display_altair_width_params(params):
        return "```json\n" + json.dumps(params, indent=2) + "\n```"


    if __name__ == "__main__":
        app.run(debug=True)
```

Interact with Click Data



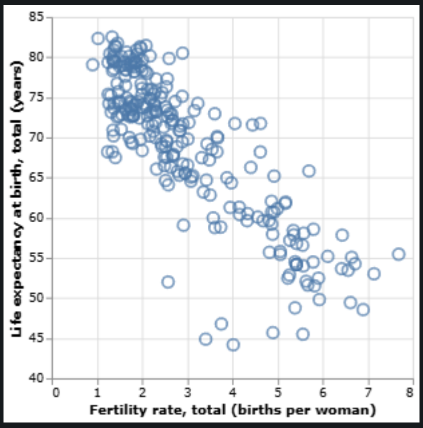
```
{
  "selected_points": {}
}
```

Box Selection Data

Vega-Altair charts also have the built-in `selection_interval` method, which can be used to fetch the values of a region selected within a chart. Call the `selection_interval` method and give it a name. For example: `alt.selection_interval(name="brush_selection")`. Then assign that name to the `signalsToObserve` prop of the Vega component: `dvc.Vega(signalsToObserve=["brush_selection"])`.

In the following example, we take the country data in the region that the user selects and display it in a DataTable.

```python
from dash import Dash, Input, Output, callback, dash_table, dcc, html
import altair as alt
import dash_vega_components as dvc
import pandas as pd
import textwrap
import json


df = pd.read_csv("https://plotly.github.io/datasets/country_indicators.csv")
df = df.pivot(
    index=["Country Name", "Year"], columns="Indicator Name", values="Value"
).reset_index()
# Select latest year and limit to 3 columns
df = df[df["Year"] == df["Year"].max()]
df = df[
    [
        "Country Name",
        "Fertility rate, total (births per woman)",
        "Life expectancy at birth, total (years)",
    ]
]


def make_chart():
    chart = (
        alt.Chart(df)
        .mark_point(size=90)
        .encode(
            alt.X("Fertility rate, total (births per woman)").scale(zero=False),
            alt.Y("Life expectancy at birth, total (years)").scale(zero=False),
            tooltip=["Country Name"],
        )
    )

    brush = alt.selection_interval(name="brush_selection")
    chart = chart.add_params(brush)
    return chart.to_dict()


def format_json(json_data):
    return "```json\n" + json.dumps(json_data, indent=2) + "\n```"


app = Dash()

app.layout = html.Div(
    [
        dvc.Vega(
            id="altair-chart1",
            spec=make_chart(),
            signalsToObserve=["brush_selection"],
        ),
```



You can read out any parameter/signal of a chart. Select a region in the chart. The numbers you see below are the lower and upper bounds of the selection and you can use them to filter a dataframe.

```
{
  "brush_selection": {}
}
```

Country Name	Fertility rate, total (births per woman)	Life expectancy at birth, total (years)
Afghanistan	6.437	57.8338292683
Albania	1.635	76.4702926829
Algeria	2.661	72.8983658537
American Samoa		
Andorra	1.18	
Angola	6.619	49.43573170729999
Antigua and Barbuda	2.1830000000000003	74.8032195122
Arab World	3.44313489066	69.2070082278
Argentina	2.4130000000000003	75.0090487805
Armenia	1.726	73.7646585366

« ‹ 1 / 27 › »

Dash Python ***> Dash Vega-Altair (Community Component)***