Interactive DataViz with Python

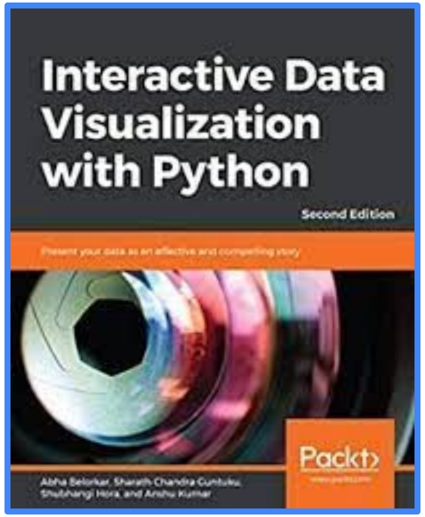
[Bokeh] [Plotly Express] [Altair]

Interactive DataViz

can react and respond to **user actions** in the moment.

Interactive data visuals are static vis that incorporate features to accept human inputs.

Interactive Data Visualization with Python.


Interactive Data Visualization with Python
Second Edition
Present your data as an effective and compelling story
Packt
www.packt.com
Abha Belorkar, Sharath Chandra Guntuku, Shubhangi Hora, and Anshu Kumar

"Interactive Data Visualization with Python"

by Abha Belorkar, Sharath Chandra Guntuku, Shubhangi Hora, Anshu Kumar

Publisher: Packt Publishing

Release Date: April 2020

ISBN: 9781800200944

On Safari

On Github

Install the libraries

plotly express install

bokeh install

altair install

chart studio install

Plotly Express

Plotly is a technical computing company in Montreal, Quebec.

Nicolas Kruchten, VP of Product, is the creator of **Plotly Express**.

SciPy 2021 talk introduces Plotly Express and Dash.

plotly.express is a built-in part of the plotly library, launched in March '19

"Makes graphics in a single function call".

Contains more than 30 functions for creating figures.

Every function uses graph objects internally and returns a *plotly.graph_objects.Figure* instance.

Bokeh

Bokeh is a Python library for creating interactive visualizations for modern web browsers.

Bryan Van de Ven is the core developer: Interview, LinkedIn, Twitter

Bryan is based in Portland, Oregon; currently the software engineer at NVIDIA.

Project was originally sponsored by Continuum Analytics (currently Anaconda, Inc.)

Altair

Altair is a **declarative** statistical visualization library based on Vega and Vega-Lite, the source is available on GitHub.

It is developed by Jake Vanderplas and Brian Granger in close collaboration with the UW Interactive Data Lab.

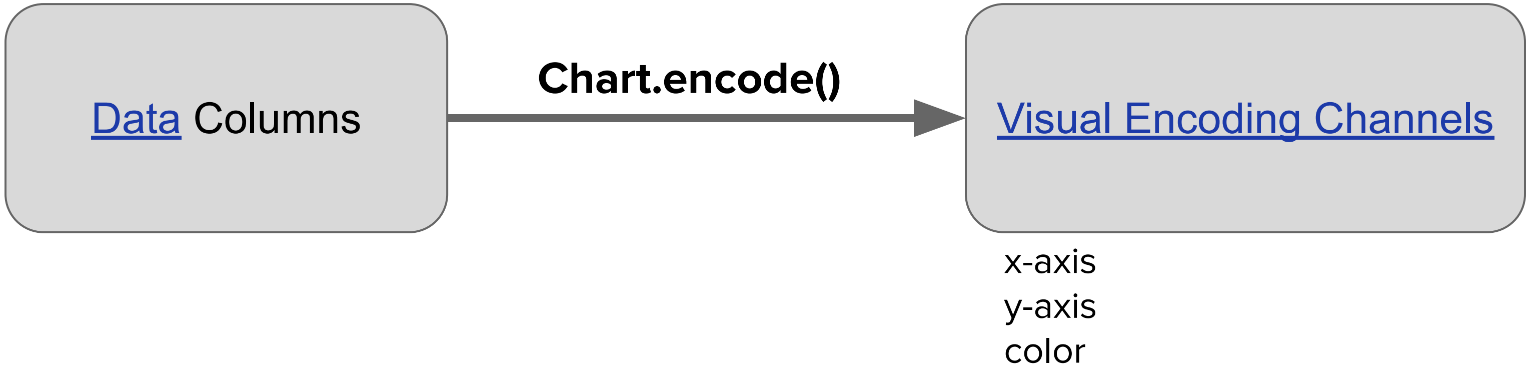
Jake is a Software Engineer at **Google** based in Seattle, WA.

Brian is a Program Manager at **AWS** based in Los Osos, CA.

Vega Online Editor

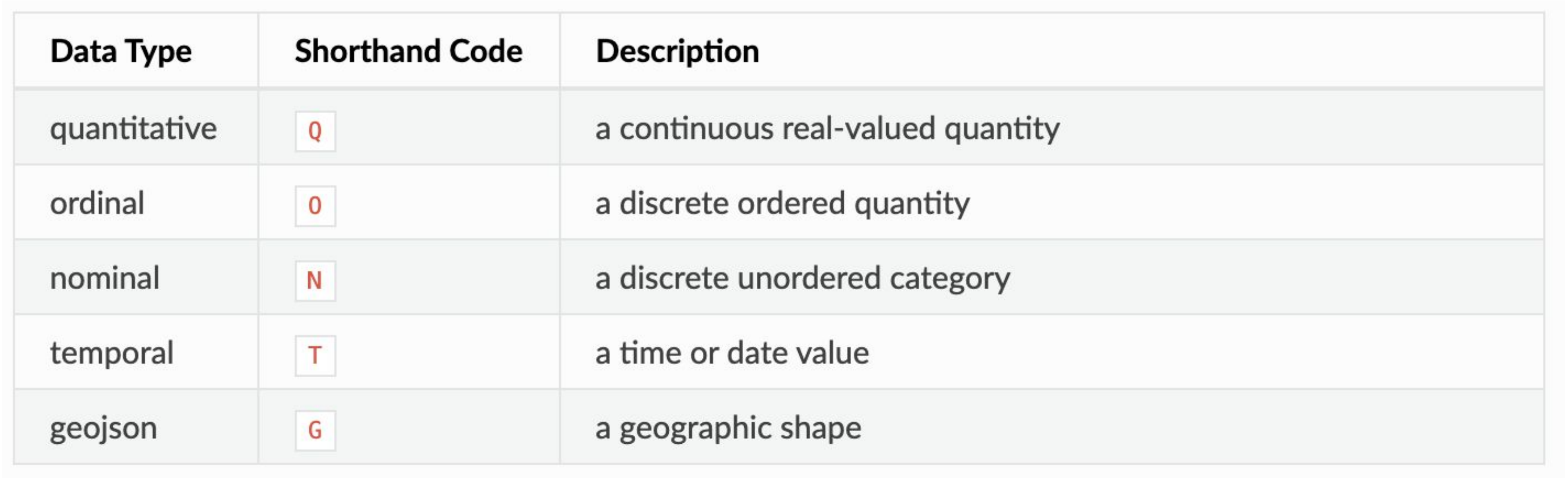
Altair Encodings

The key to creating meaningful visualizations is to **map** properties of the data to visual properties in order to effectively communicate information. In Altair, this mapping of visual properties to data columns is referred to as an encoding, and is most often expressed through the **Chart.encode()** method.



Altair Encoding Data Types

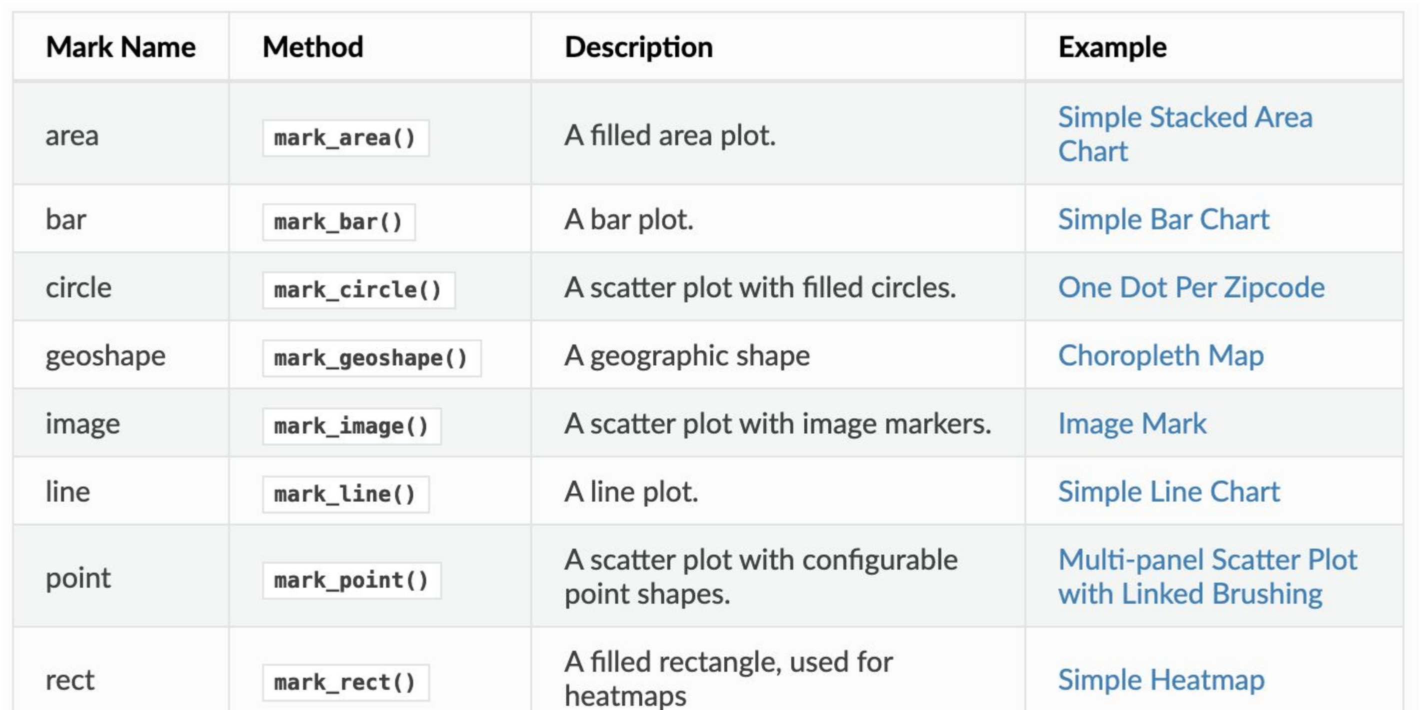
The details of any mapping depend on the type of the data.



Data Type	Shorthand Code	Description
quantitative	`Q`	a continuous real-valued quantity
ordinal	`O`	a discrete ordered quantity
nominal	`N`	a discrete unordered category
temporal	`T`	a time or date value
geojson	`G`	a geographic shape

Altair Marks

We saw in Encodings that the encode() method is used to map columns to visual attributes of the plot. The **mark property** is what specifies how exactly those attributes should be represented on the plot.



Mark Name	Method	Description	Example
area	`mark_area()`	A filled area plot.	Simple Stacked Area Chart
bar	`mark_bar()`	A bar plot.	Simple Bar Chart
circle	`mark_circle()`	A scatter plot with filled circles.	One Dot Per Zipcode
geoshape	`mark_geoshape()`	A geographic shape	Choropleth Map
image	`mark_image()`	A scatter plot with image markers.	Image Mark
line	`mark_line()`	A line plot.	Simple Line Chart
point	`mark_point()`	A scatter plot with configurable point shapes.	Multi-panel Scatter Plot with Linked Brushing
rect	`mark_rect()`	A filled rectangle, used for heatmaps	Simple Heatmap

Altair Interactions

Inherited from Vega-Lite is a declarative grammar of not just visualization, but **interaction**.

- the **selection** object **captures interactions** from the mouse or through other inputs to effect the chart. Inputs can either be events like **mouse clicks or drags**. Inputs can also be elements like a **drop-down, radio button** or **slider**.
- the **condition()** function takes the selection input and changes an element of the chart based on that input.
- the **bind** property of a selection establishes a two-way binding between the selection and an input element of your chart.

Altair Selections

Vega light currently supports two selection types:

Point

- to select multiple discrete data values;
- the first value is selected on click and additional values toggled on shift-click.

Interval

- to select a continuous range of data values on drag.

Altair supports three:

Single

to select a single discrete data value **on click.**

Multi

to select multiple discrete data value; the first value is selected on click and additional values toggled **on shift - click.**

Interval

Same as above.

Altair Example: Interactive Rectangular Brush

https://altair-viz.github.io/gallery/interactive_brush.html

This example shows how to add a simple rectangular brush to a scatter plot. By clicking and dragging on the plot, you can highlight points within the range.

Try opening the viz in Vega editor.

The **Vega editor** is a web application for authoring and testing Vega and Vega-Lite visualizations. It includes a number of example specifications that showcase both the visual encodings and interaction techniques. It is deployed at Vega Editor

Altair Filter Transform and Selection

The filter transform removes objects from a data stream based on a provided filter expression, selection, or other filter predicate.