Altair
Cheat sheet

No installation required

- We will work with Google Collab, no installation required.

But in case you would like to install Altair locally, you also can do so:

- Go to Altair installation instructions

https://altair-viz.github.io/getting_started/installation.html

- Work with Anaconda and JupyterLab

Fundamental concepts in Altair

- Chart: the fundamental object in Altair. It takes a data frame as argument
- (graphical) Mark: the geometric shape we want to use to represent the data. Ex. Point, bar, line…
- Encoding channel: a relation between a field in the data frame and an attribute of a mark (x,y, colour, shape, size, tooltip…).
- Data types:
  - nominal (unordered, categorical values):N, (ex. Computers, Smartphones, Printers)
  - quantitative (real-valued number):Q, (ex. Inhabitants in a country; Budget)
  - ordinal (rank-ordered data):O, (ex. S, M, L, XL)
  - or temporal (date/time data):T, (for temporal data types ex. alt.X('month(graduationYear):T')

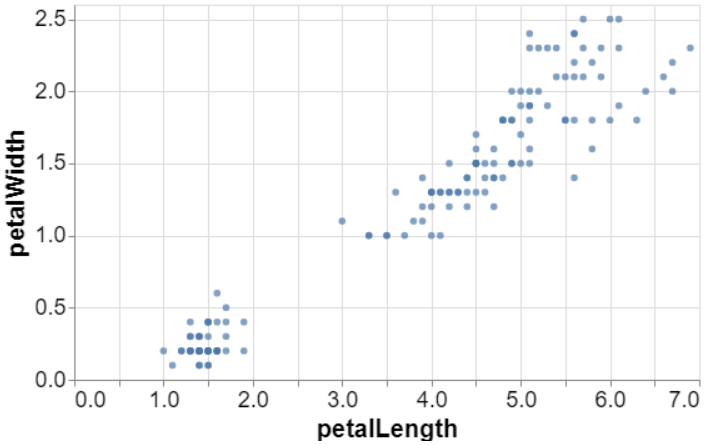
Simple pattern

the *Chart* object takes in the data (df)

the *mark* type specifies the geometric shape we want in our chart

the *encoding* object specifies the mapping between columns in our data and visual aspects of the chart

```
alt.Chart(df).mark_circle().encode(
    x='petalLength',
    y='petalWidth'
)
```



Source: [Introduction to Altair by Allan Campopiano]

Marks

- Mark:arc() – Wedges in a circle, for donut, pie or radial charts
- mark_area() - Filled areas defined by a top-line and a baseline.
- mark_bar() - Rectangular bars.
- mark_circle() - Scatter plot points as filled circles.
- mark_line() - Connected line segments.
- mark_point() - Scatter plot points with configurable shapes.
- mark_rect() - Filled rectangles, useful for heatmaps.
- mark_rule() - Vertical or horizontal lines spanning the axis.
- mark_square() - Scatter plot points as filled squares.
- mark_text() - Scatter plot points represented by text.
- mark_tick() - Vertical or horizontal tick marks.

For a complete list, and links to examples, see the Altair marks documentation

Encoding channels (I)

- Position encodings: map the data with coordinates
  - x: Horizontal (x-axis) position of the mark.
  - y: Vertical (y-axis) position of the mark.
- Mark encodings: map the data onto properties of the mark itself
  - size: Size of the mark. May correspond to area or length, depending on the mark type.
  - color: Mark color, specified as a legal CSS color.
  - opacity: Mark opacity, ranging from 0 (fully transparent) to 1 (fully opaque).
  - shape: Plotting symbol shape for point marks.
  - order: Mark ordering, determines line/area point order and drawing order.

Encoding channels (II)

- Text encodings
  - text: show the value as text
  - tooltip: Tooltip text to display upon mouse hover over the mark.
- Facet encodings: map the data onto views of the data separated into sub charts
  - column: Facet the data into horizontally-aligned subplots.
  - row: Facet the data into vertically-aligned subplots.

For a complete list of available channels, see the Altair encoding documentation.

Source: Visualization curriculum

Shorthand *vs* Longhand

- Longhand used when you want to customize some details

Ex.

```
alt.Chart(df).mark_circle().encode(
    x=alt.X('petalLength’,
            scale=alt.Scale(domain=[-4,10])),
    y=alt.Y('petalWidth’,
            axis=alt.Axis(labelColor='red')),
    color=alt.Color('species’,
            legend=alt.Legend(labelColor='blue'))
)
```

Longhand additional examples

```
alt.X(… axis=alt.Axis(title=“Precipitation in USA”)
alt.Color(…, legend=None)
Tooltip = [
      alt.Tooltip('country:N'),
       alt.Tooltip('fertility:Q'),
]
alt.Chart(….
).properties(width=135, height=135)
```

Aggregation functions

- average(field)
- count
- min, max
- median, stdev
- Also with time: year, yearmonth, yearmonthday, quarter

We can also bin (group) values

```
alt.X('field_name:Q',bin=True)   or
alt.X('field_name:Q',bin=alt.BinParams(maxbins=20))
```

Composition of charts

- Layering
  chart1 + chart2
- Horizontal concatenation
  chart1 | chart2
- Vertical concatenation
  chart1 & chart2

Column and row encodings channels: set of sub-plots

ex. alt.Column('cluster:N') will generate as many charts as clusters

Interaction

- Panning and zooming

  alt.Chart (..

).interactive()

Exporting / Conversion

- chart.to_json()
  - Outputs a Vega-Lite specification
- chart.save('chart.html')