Advanced Visualization

Trendlines

Error Bars

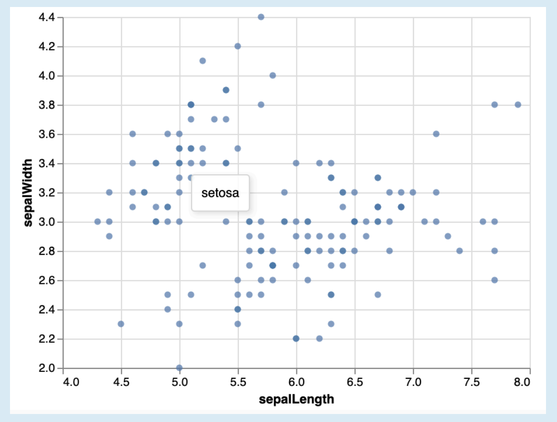
Confidence Intervals

Interactivity

The highlighted codes are the key to the certain function

Tooltips

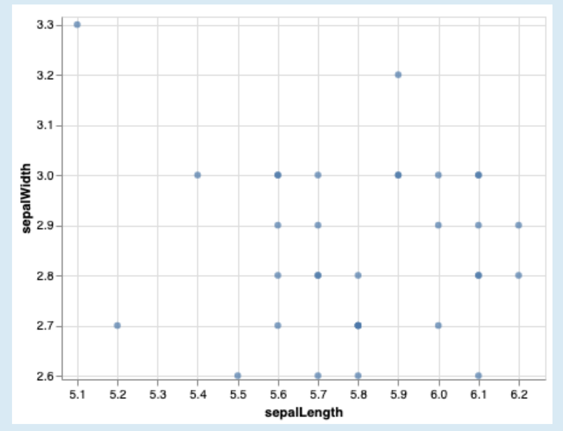
```
alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    tooltip='species')
```



The tooltips allow us to hover over the points to see the information in a tooltip.
*Multiple fields can be included, replacing the highlighted grammar with `tooltip='tooltip_1', 'tooltip_2'`

Panning and zooming

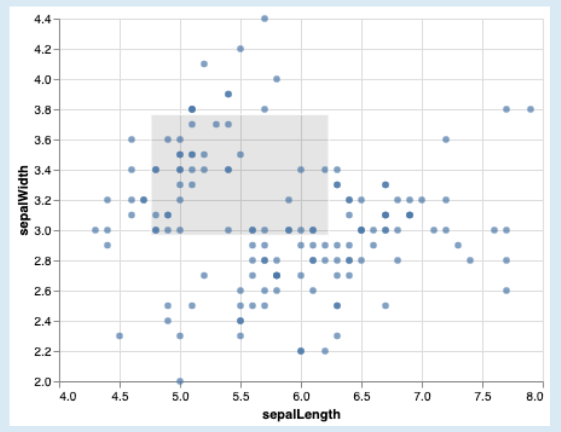
```
alt.Chart(setosa_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False))
).interactive()
```



The `.interactive` allow us to pan or zoom the figure. The left figure shows the effect of zooming.

Interval selection

```
brush = alt.selection_interval()
alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False))
).add_selection(brush)
```

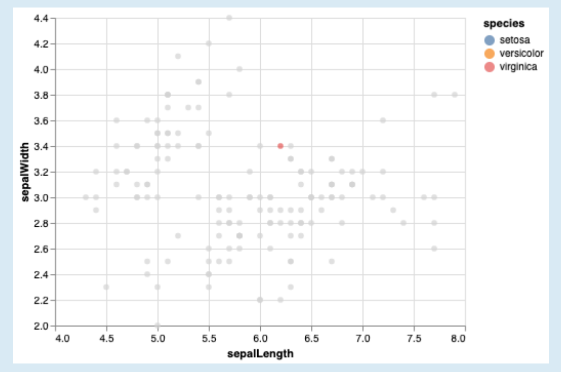


We can drag and drop with the mouse to create an interval of selected points.

*We could change along which dimensions the selection is active by the argument `selection_interval(encodings=['<x> or <y>'])`

Click selection and highlight

```
click = alt.selection_multi()
alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    color = alt.condition(click, 'species',
                          alt.value('lightgray'))
).add_selection(click)
```

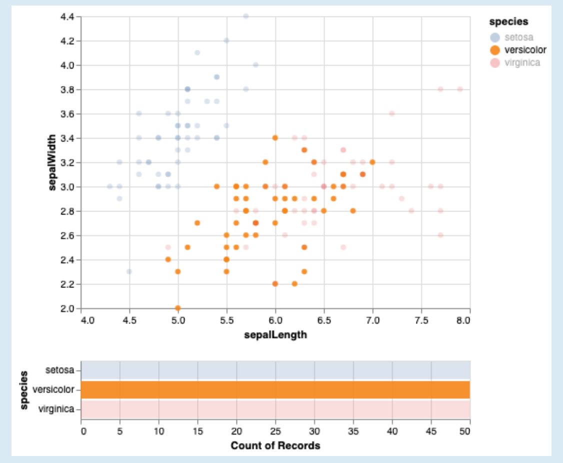


We can click with the mouse to select points and change the grammar of color to highlight the points.

*Highlight can also be combined with interval selections.

Legend selection

```
brush = alt.selection_interval()
click = alt.selection_multi(fields=['species'],
                            bind='legend')

points = (alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    color=alt.condition(brush, 'species',
                        alt.value('lightgray')),
    opacity=alt.condition(click,
                          alt.value(0.9), alt.value(0.2)))
.add_selection(brush))

bars = (alt.Chart(iris_data).mark_bar().encode(
    x='count()',
    y='species',
    color='species',
    opacity=alt.condition(click,
                          alt.value(0.9), alt.value(0.2))))

(points & bars).add_selection(click)
```



We can specify that we want to bind it to the legend. We also need to add the selection to the combined chart instead of to the bar chart or the point chart since the legend belongs to both of them.

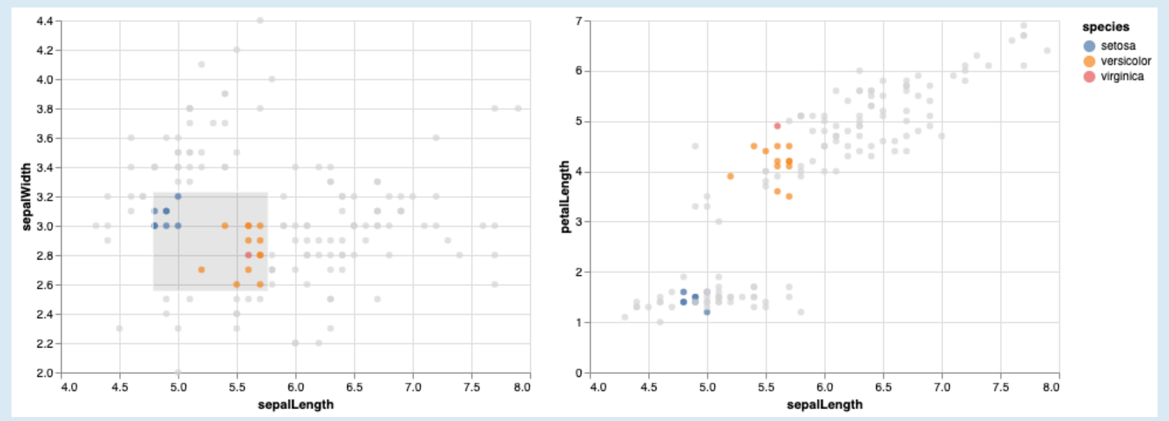
Material from: (1) Joel Östblom, DSCI 531 Lecture Notes (2) Vega-Altair https://altair-viz.github.io/

Advanced Visualization

Linking selections across plots

Link with interval

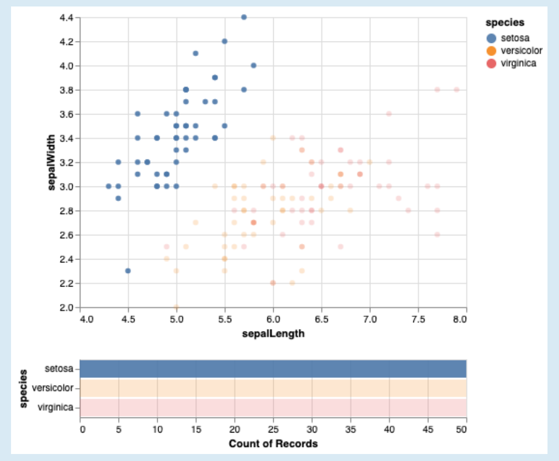
```
brush = alt.selection_interval(resolve='union')
points = (alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    color=alt.condition(brush, 'species',
                        alt.value('lightgray')))
.add_selection(brush))

points | points.encode(y='petalLength')
```



* The default of the `resolve` argument is '`global`'. In order that each subplot gets its own selection and that points within any section are highlighted within all plots, we can use '`union`'. In order that only points that fall within the intersection of *all* the selections are highlighted, we can use '`interaction`'.

Link with both of interval selections and click

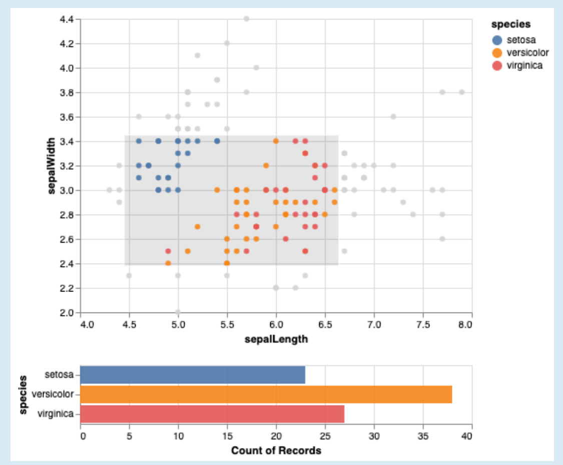
```
brush = alt.selection_interval()
click = alt.selection_multi(fields=['species'])

points = (alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    color=alt.condition(brush, 'species',
                        alt.value('lightgray')),
    opacity=alt.condition(click,
                          alt.value(0.9), alt.value(0.2)))
.add_selection(brush))

bars = (alt.Chart(iris_data).mark_bar().encode(
    x='count()',
    y='species',
    color='species',
    opacity=alt.condition(click,
                          alt.value(0.9), alt.value(0.2)))
.add_selection(click))

points & bars
```



We can link the charts together. For the bar chart selector, we need to specify which field/column we should select on.

Filter data based on selection & Binding element to selection event

```
brush = alt.selection_interval()
click = alt.selection_multi(fields=['species'],
                            bind='legend')

points = (alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    color=alt.condition(brush, 'species',
                        alt.value('lightgray')),
    opacity=alt.condition(click,
                          alt.value(0.9), alt.value(0.2)))
.add_selection(brush))

bars = (alt.Chart(iris_data).mark_bar().encode(
    x='count()',
    y='species',
    color='species',
    opacity=alt.condition(click,
                          alt.value(0.9), alt.value(0.2)))
   .transform_filter(brush))

(points & bars).add_selection(click)
```

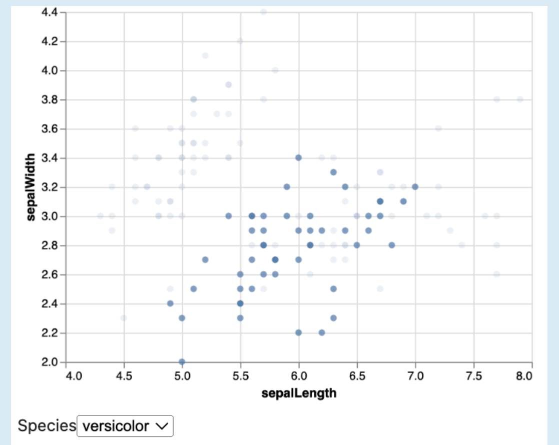


We can filter the data based on a selection, by adding `transform_filter(brush)` to the bar plot.

* To bind certain element, we need to use the argument `bind=<binding_element>` in the selection to determine the binded element.

Dropdowns

```
species_s = sorted(iris_data['species'].unique())
dropdown = alt.binding_select(name='Species',
                              options=species_s)

select_species = alt.selection_single(
    fields=['species'],
    bind=dropdown,
    init={'species': 'versicolor'})

alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    tooltip='species',
    opacity=alt.condition(select_species,
                          alt.value(0.7), alt.value(0.1))
).add_selection(select_species)
```

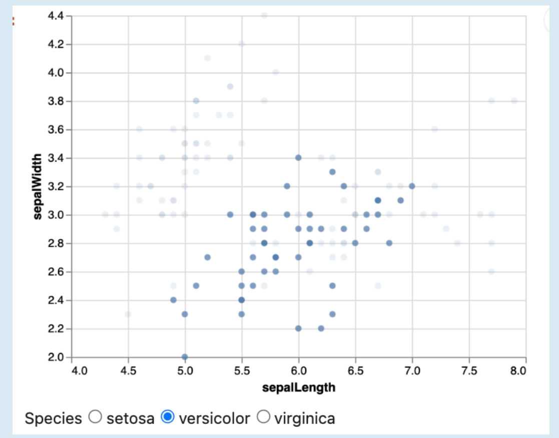


We can create a dropdown selection widget by `alt.binding_select` to let us choose categories without coloring the points.

* We can use the argument `name=<name>` to give the dropdown a nice name.
* We can use the argument `init=<{column: category}>` to set the default value for the selection.

Radio buttons

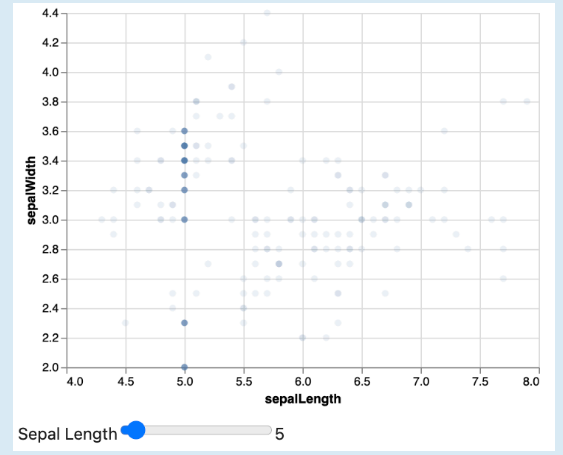
```
species_r = sorted(iris_data['species'].unique())
radio_species = alt.binding_radio(name='Species',
                                  options=species_r)

select_species = alt.selection_single(
    fields=['species'],
    bind=radio_species)

alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    tooltip='species',
    opacity=alt.condition(select_species,
                          alt.value(0.7), alt.value(0.1))
).add_selection(select_species)
```



Similar to dropdown selection, we can create a radio button selection widget by `alt.binding_radio` to let us choose categories without coloring the points.

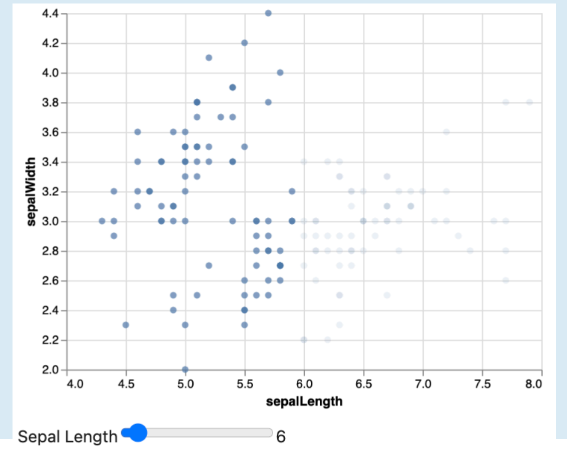
Sliders

```
slider = alt.binding_range(name='Sepal Length')
select_rating = alt.selection_single(
    fields=['sepalLength'],
    bind=slider)

alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    tooltip='species',
    opacity=alt.condition(select_rating,
                          alt.value(0.7), alt.value(0.1))
).add_selection(select_rating)
```



We can create a slider button selection widget by `alt.binding_range` to let us choose categories or continuous variables without coloring the points.

```
slider = alt.binding_range(name='Sepal Length')
select_rating = alt.selection_single(
    fields=['sepalLength'],
    bind=slider)

alt.Chart(iris_data).mark_circle().encode(
    x=alt.X('sepalLength', scale=alt.Scale(zero=False)),
    y=alt.Y('sepalWidth', scale=alt.Scale(zero=False)),
    tooltip='species',
    opacity=alt.condition(
        alt.datum.sepalLength < select_rating.sepalLength,
        alt.value(0.7), alt.value(0.1))
).add_selection(select_rating)
```



We can use `alt.datum` for making comparisons of bigger and smaller than.

Material from: (1) Joel Östblom, DSCI 531 Lecture Notes (2) Vega-Altair https://altair-viz.github.io/