Data 550: Data Visualization I

Lecture 8: Layouts and Interactivity

Dr. Irene Vrbik

University of British Columbia Okanagan

Overview

By the end of the lecture you will be able to:

- Layout plots in panels of a figure grid.
- Create selections within a chart.
- Link selections between chart to highlight and filter data.

Suggested readings from Fundamentals of Data Visualization.

- Section 29 on telling a story

Introduction

- While data visualization plays a key role in EDA, for example, we often create charts for the purpose of *communication*.
- As data scientists, we would like to successfully convey insights to our audience with a clear *story*.
- A single (static) visualization will rarely tell an entire story
- Rather than trying to encode all of our findings into a single visualization, it is often a better idea to separate complex visualizations into a collection of simpler figures.

Story telling

How to tell a story

There are standard patterns for storytelling, e.g. Opening (O)–Challenge (C)–Action (A)–Resolution (R) format[1]

A story | Elements

Stephen Hawking was diagnosed with motor neuron disease at age 21—one year into his PhD—and was given two years to live. Hawking did not accept this predicament and started pouring all his energy into doing science. Hawking ended up living to be 76, became one of the most influential physicists of his time, and did all of his seminal work while being severely disabled. - Wilke, 29 Telling a story and making a point

1. for standard forms of story telling for scientists and analysts see: Schimel, J. 2011. *Writing Science: How to Write Papers That Get Cited and Proposals That Get Funded*.

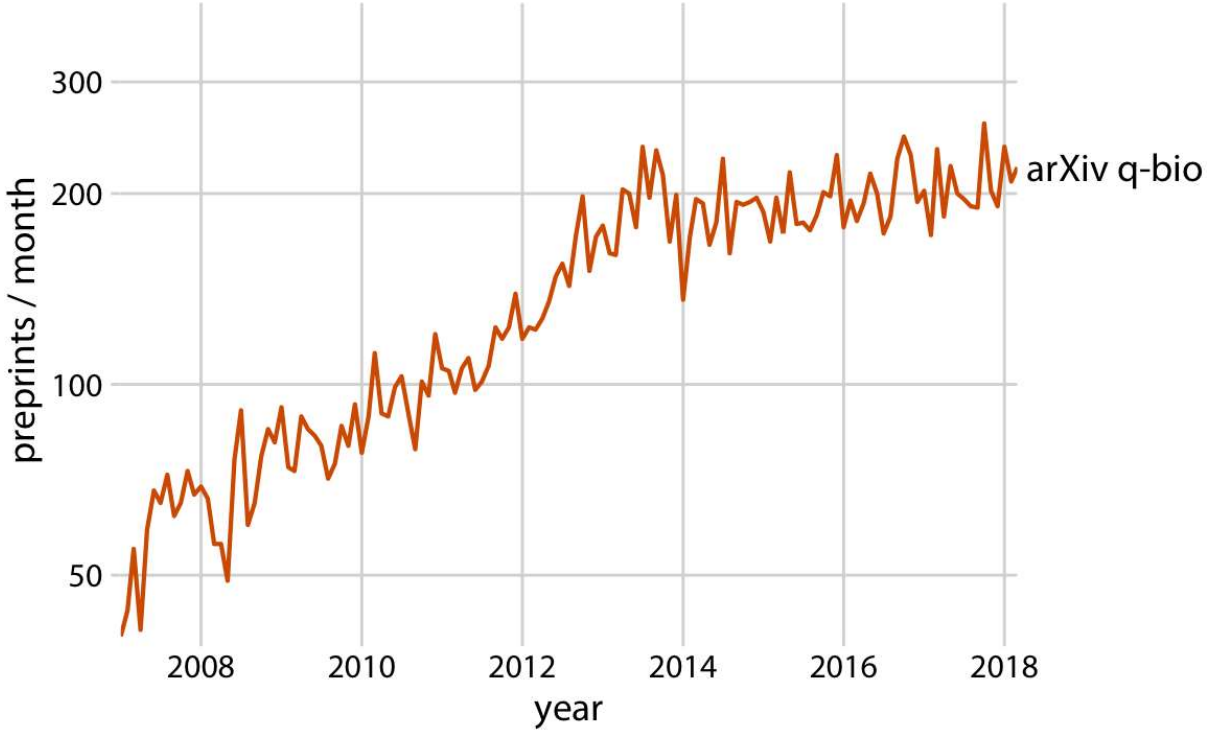
Telling a story with data

- Peak your audience's interest, e.g. start with a striking figure
- Make the reader curious about the next figure, e.g. an important problem that begs for a solution
- Present facts in an engaging way without hyperbole, e.g. reveal data that challenges the reader's previous knowledge
- Tailor it to the audience you have in mind, e.g. be at the appropriate technical difficulty and detail

The Opening (O)

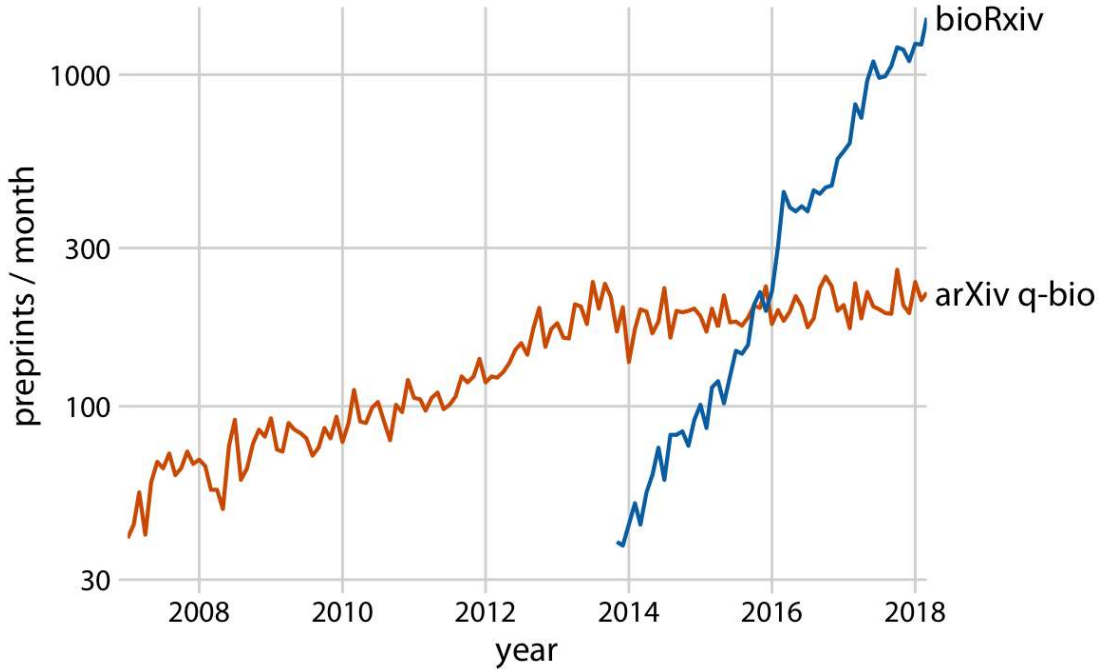
- *Preprints* are drafts of manuscripts, i.e. before formal peer review and official publication.
- In the early 1990s, physicists created a preprint server, arXiv.org, which served as an efficient central repository to store and distribute manuscript drafts.
- Shortly after it was established, arXiv.org started to branch out and become popular in related quantitative fields.
- Let's investigate the preprint submissions to the quantitative biology (q-bio) section of arXiv.org.

The Challenge (C)



Growth in monthly submissions to the quantitative biology (q-bio) section of the preprint server arXiv.org. A sharp transition in the rate of growth can be seen around 2014. While growth was rapid up to 2014, almost no growth occurred from 2014 to 2018. Note that the y axis is logarithmic, so a linear increase in y corresponds to exponential growth in preprint submissions. Data source: Jordan Anaya prepubmed.org, Ex source: Ch 29 Wilke

The Resolution (R)



The leveling off of submission growth to q-bio coincided with the introduction of the bioRxiv server. Shown are the growth in monthly submissions to the q-bio section of the general-purpose preprint server arxiv.org and to the dedicated biology preprint server bioRxiv. The bioRxiv server went live in November 2013, and its submission rate has grown exponentially since. It seems likely that many scientists who otherwise would have submitted preprints to q-bio chose to submit to bioRxiv instead.

Schimel Quotes

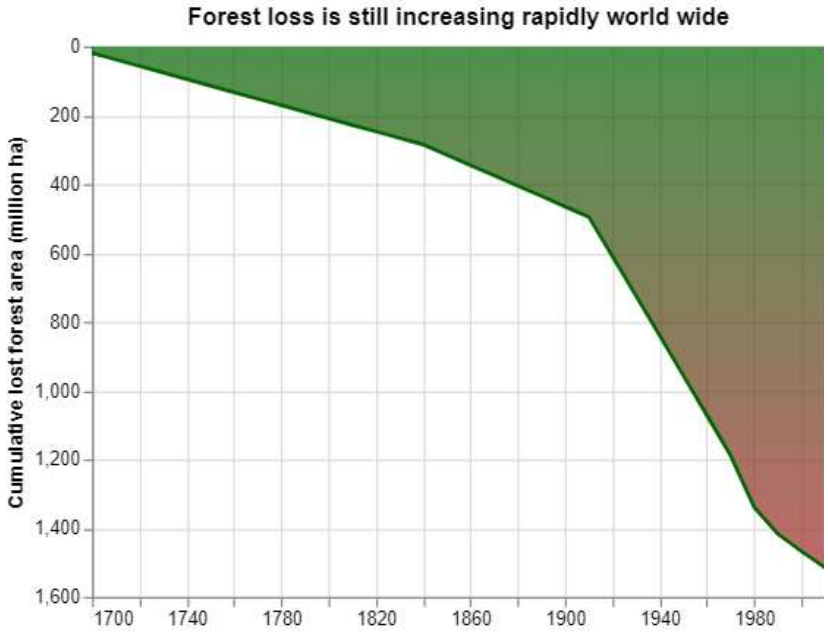
> To tell a good story in science, you must assess your data and evaluate the possible explanations — which are most consistent with existing knowledge and theory?

> It is the author's job to make the reader's job easy.

Schimel, J. 2011. *Writing Science: How to Write Papers That Get Cited and Proposals That*

A tale of deforestation

Plot Code

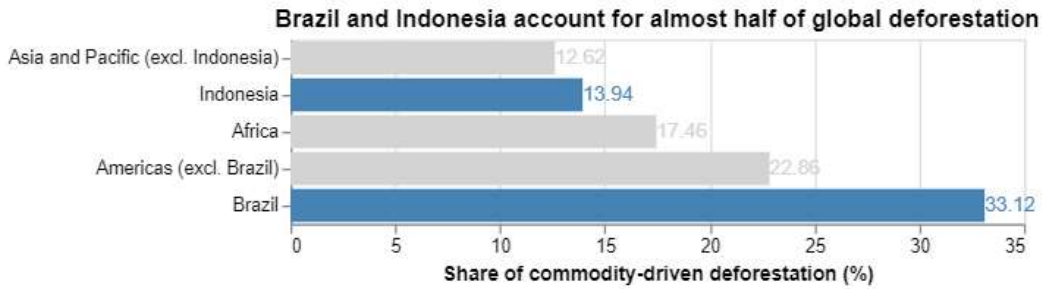


The total global forest loss during the last 300 years is about 1.5 billion hectares (that's almost twice the area of Australia!)

Permanent loss of forest (or "deforestation") is a big problem that leads to species extinction, land erosion, and accelerates climate change.

The main players

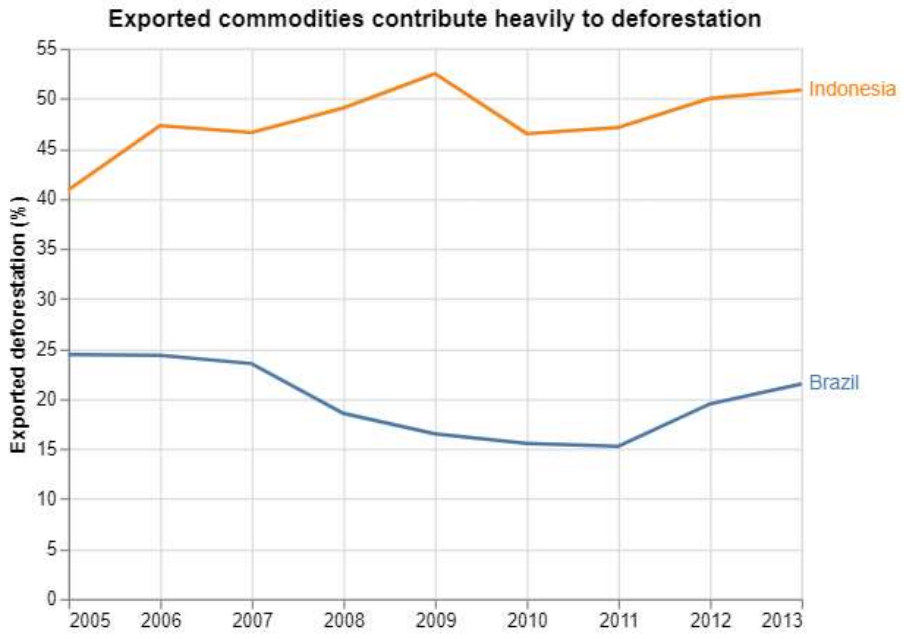
Plot Code



Brazil and Indonesia account for roughly half of the commodity-driven deforestation around the world.

Percentage of exported deforestation

Plot Code



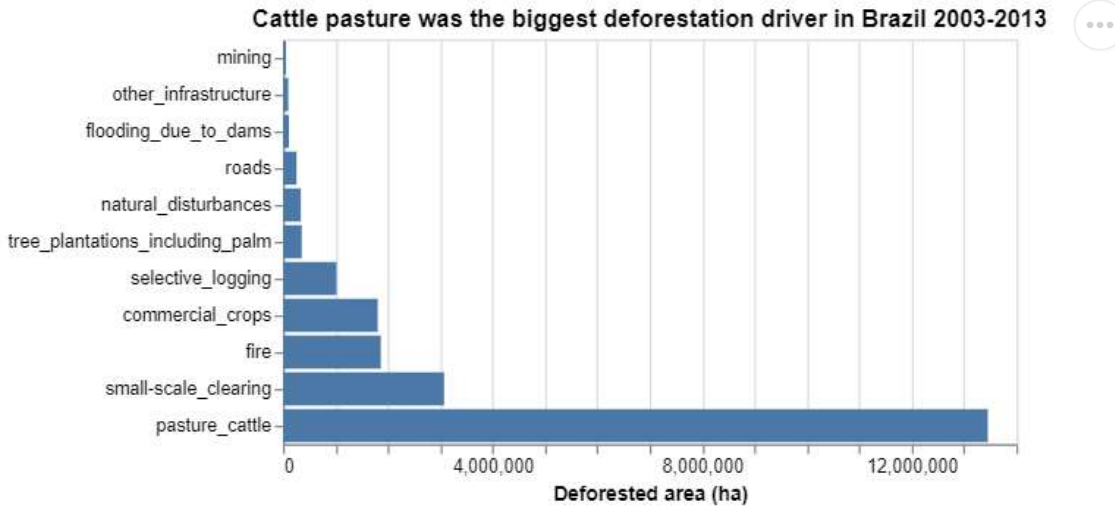
A large part of the deforestation in both countries is driven by the demand from other countries.

This is most striking in Indonesia, where the materials from around half the country's deforestation ends up in exported goods.

What are they exporting?

Exported commodities (Brazil)

Plot Code



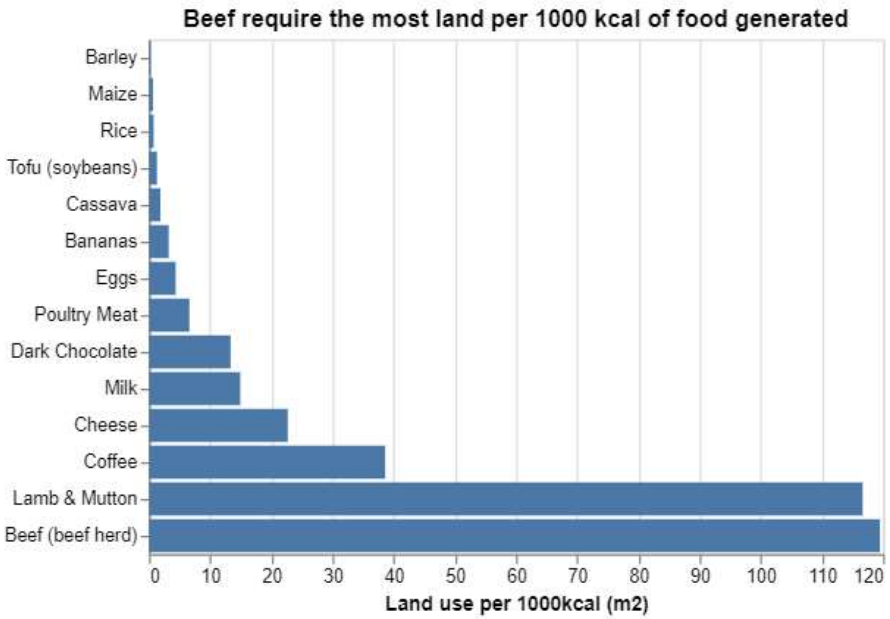
The biggest driver of deforestation in Brazil between the years 2003 and 2013 is pasture cattle.

Perhaps our demand for food has simply increased and cattle herds are an effective use of land to create food for human consumption?

Note: we do not have the data for how much of each commodity is exported

Land use efficiency

Plot Code



Raising cattle for the purpose of beef production is one of the most inefficient ways to produce food for human consumption.

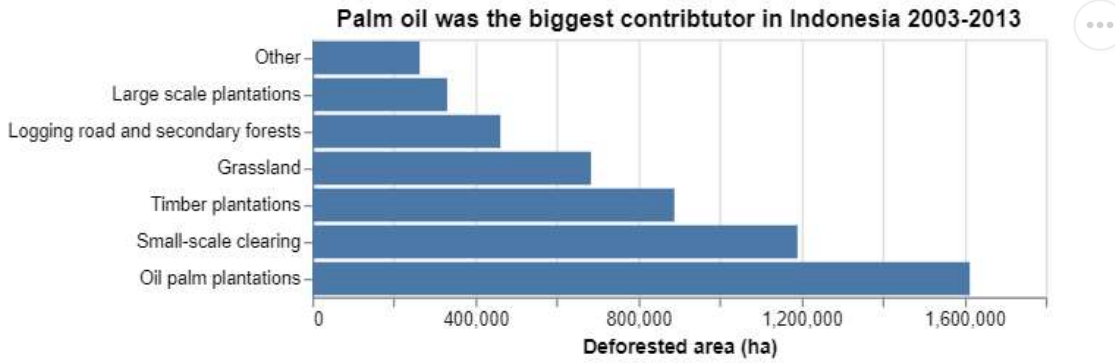
Is the solution to decrease the global demand for beef?

Comments (no resolution chart)

- The inefficient land use coupled with the high demand for beef are major contributors to the deforestation in Brazil.
- When telling a story, it is important to not jump to conclusions or suggest that solutions to problems are simpler than what they might really be.
- Changing beef for other products could be beneficial, but it could also be challenging and costly up front to implement.
- In telling this story, it would be important to also include these uncertainties and potential challenges.

Exported commodities (Indonesia)

Plot Code



The biggest drivers of deforestation in Indonesia between the years 2003 and 2013 is palm oil.

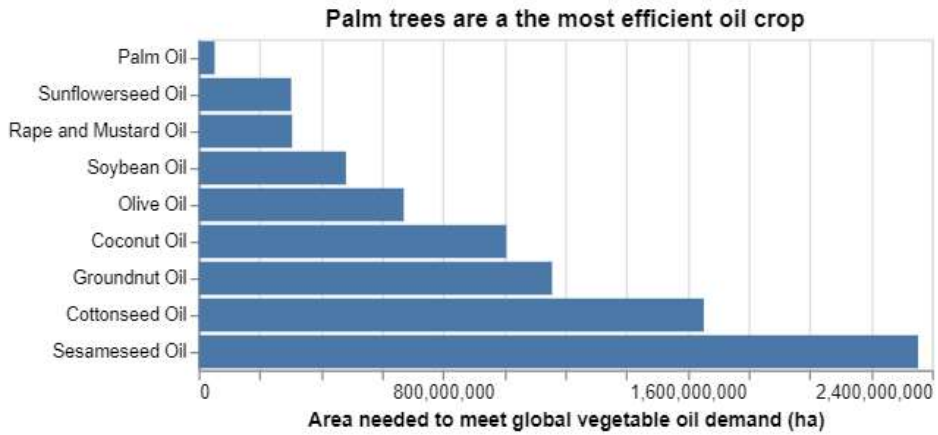
Secondary contributors are small scale clearings, timber plantations, and grasslands.

Replace this palm oil crop with a more efficient alternative?

Note: we do not have the data for how much of each commodity is exported

Land use efficiencies (Oil)

Plot Code



Palm oil is the most efficient use of land for vegetable oil production

For example, if we wanted to replace it with olive oil, we would need almost 10x the land to do so!

So keep clearing Indonesias forest to make palm oil?

Comments

- Things are not so simple. The type of land where the oil crop grows also matters.
- Therefore it is important that palm oil is grown in a *sustainable* way that is compatible with preserving the tropical forests in Indonesia.
- Just as with pasture cattle in Brazil, the story of palm oil is more complex then we can illustrate in a single visualization.

For more read Why boycotting palm oil could do more harm than good

Summary

1. **Peak interest**: illustrating the scale of the problem
2. **Evoke curiosity**: delved deeper into both areas that have been completely new to them and topics they might have read about previously.
3. **Present facts without hyperbole**: state any caveats where our visualizations did not intend to reveal the full complexity of the situation, but highlight a few main findings.

This example was based on this article. If you want to learn more on this topic visit

Figure Composition

Figure Composition

- In previous lectures we have seen how we can layout plots together in a panel, instead of showing only one at a time.
- These compound charts were created via stacking, layering, faceting, and repeating.
- We'll see how we can use these to customize our grid

Compound Charts

Taken from the the Altair documentation for compound charts

class	functional form	operator form	reference
`LayerChart`	`alt.layer(chart1, chart2)`	`chart1 + chart2`	Layered Charts
`HConcatChart`	`alt.hconcat(chart1, chart2)`	`chart1 | chart2`	Horizontal Concatenation
`VConcatChart`	`alt.vconcat(chart1, chart2)`	`chart1 & chart2`	Vertical Concatenation

class	method form	reference
`RepeatChart`	`chart.repeat(row, column)`	Repeated Charts
`FacetChart`	`chart.facet(facet, row, column)`	Faceted Charts

In R you can create similar grid panels for plotting using `plot_grid()` from the **cowplot** library created by Claus O. Wilke (*yes that is the author of one of our texts Fundamentals of*

Penguins

Data Info

```
penguins = pd.read_csv('data/penguins.csv').dropna(subset=['sex', 'species'])
penguins
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_r
0	Adelie	Torgersen	39.1	18.7	181	3750
1	Adelie	Torgersen	39.5	17.4	186	3800
2	Adelie	Torgersen	40.3	18.0	195	3250
3	Adelie	Torgersen	36.7	19.3	193	3450
4	Adelie	Torgersen	39.3	20.6	190	3650
...	...	...	...	...	...	...
337	Gentoo	Biscoe	47.2	13.7	214	4925
338	Gentoo	Biscoe	46.8	14.3	215	4850
339	Gentoo	Biscoe	50.4	15.7	222	5750
340	Gentoo	Biscoe	45.2	14.8	212	5200

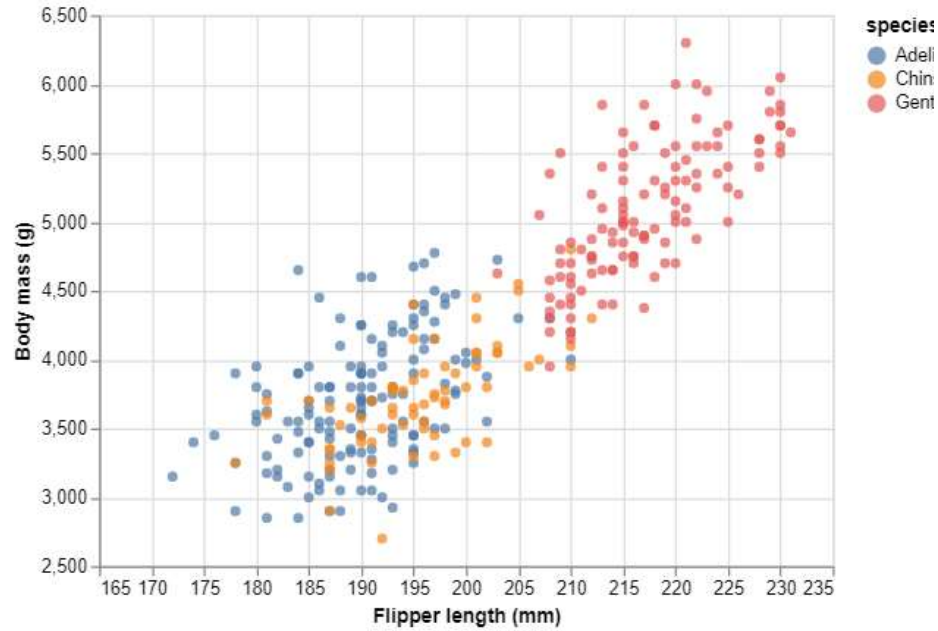
	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_r
341	Gentoo	Biscoe	49.9	16.1	213	5400

334 rows × 7 columns

Penguins Plots

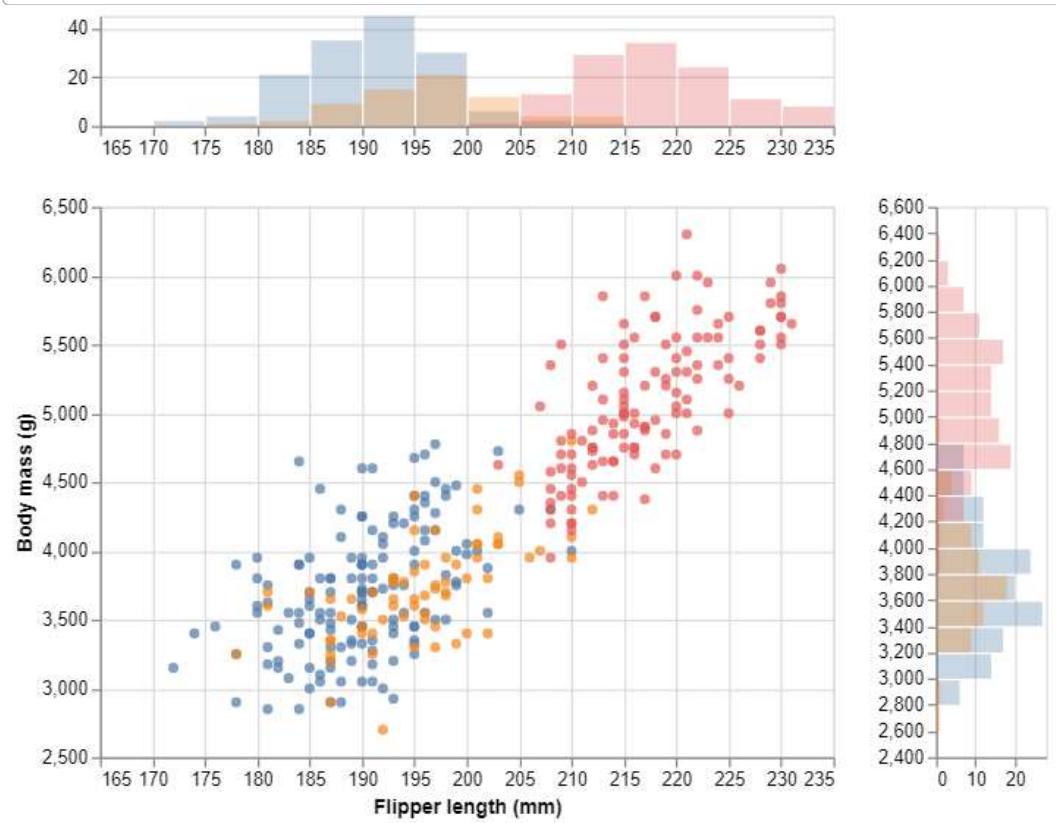
Scatter plot | Top Histogram | Right Histogram

```
xscale = alt.Scale(domain=(169, 234))
yscale = alt.Scale(domain=(2556, 6444))


points = alt.Chart(penguins
).mark_circle().encode(
x = alt.X('flipper_length_mm',
  scale=xscale, title='Flipper length (mm)'
y = alt.Y('body_mass_g', title='Body mass (
  scale=yscale),
color='species',
)

points
```



Marginal Histograms

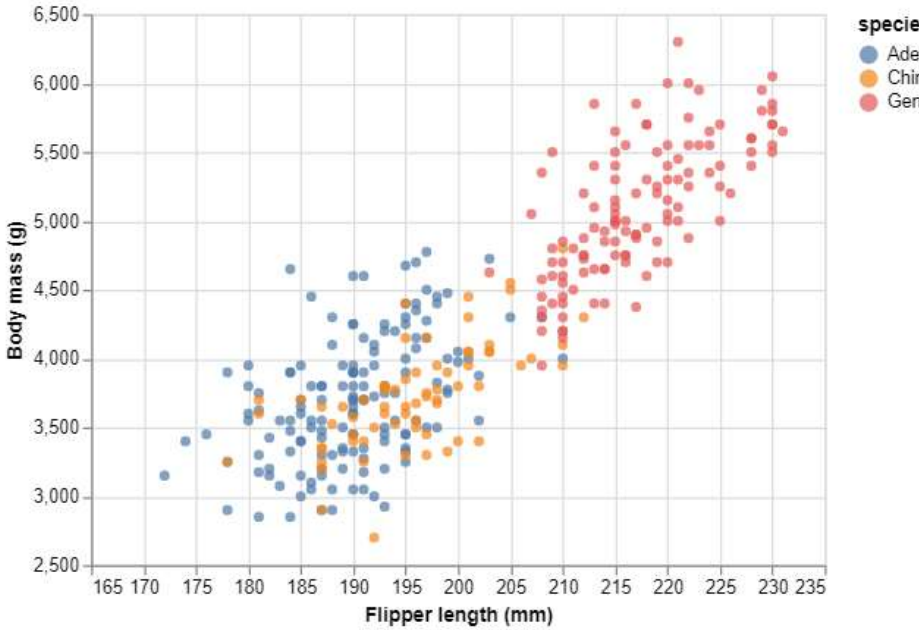
```
1 top_hist & (points | right_hist)
```



Penguin Plots 2

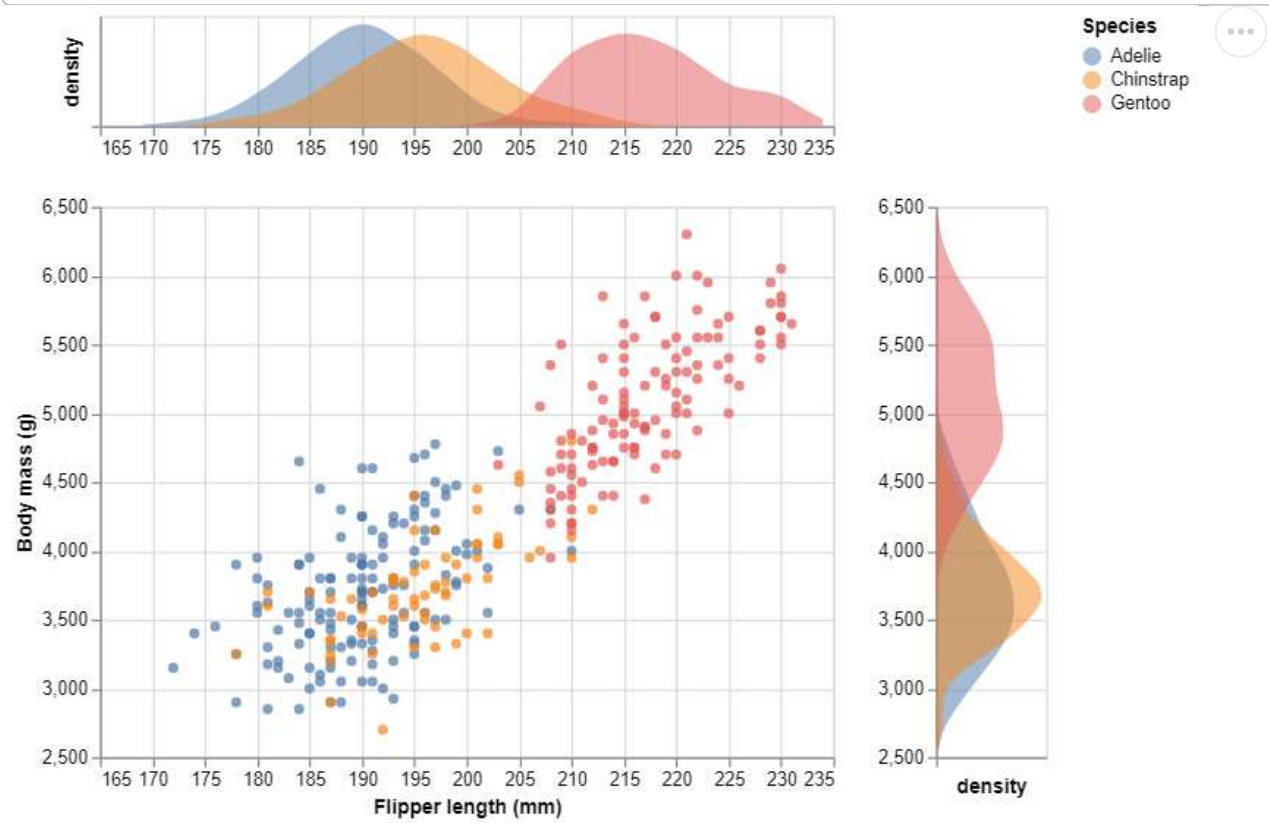
Scatter plot | Flipper KDE | Body Mass KDE

```
xscale = alt.Scale(domain=(169, 23
yscale = alt.Scale(domain=(2556, 6

points = alt.Chart(penguins
).mark_circle().encode(
x = alt.X('flipper_length_mm',
  title='Flipper length (mm)',
  scale=xscale),
y = alt.Y('body_mass_g',
  title='Body mass (g)',
  scale=yscale),
color='species',
)

points
```



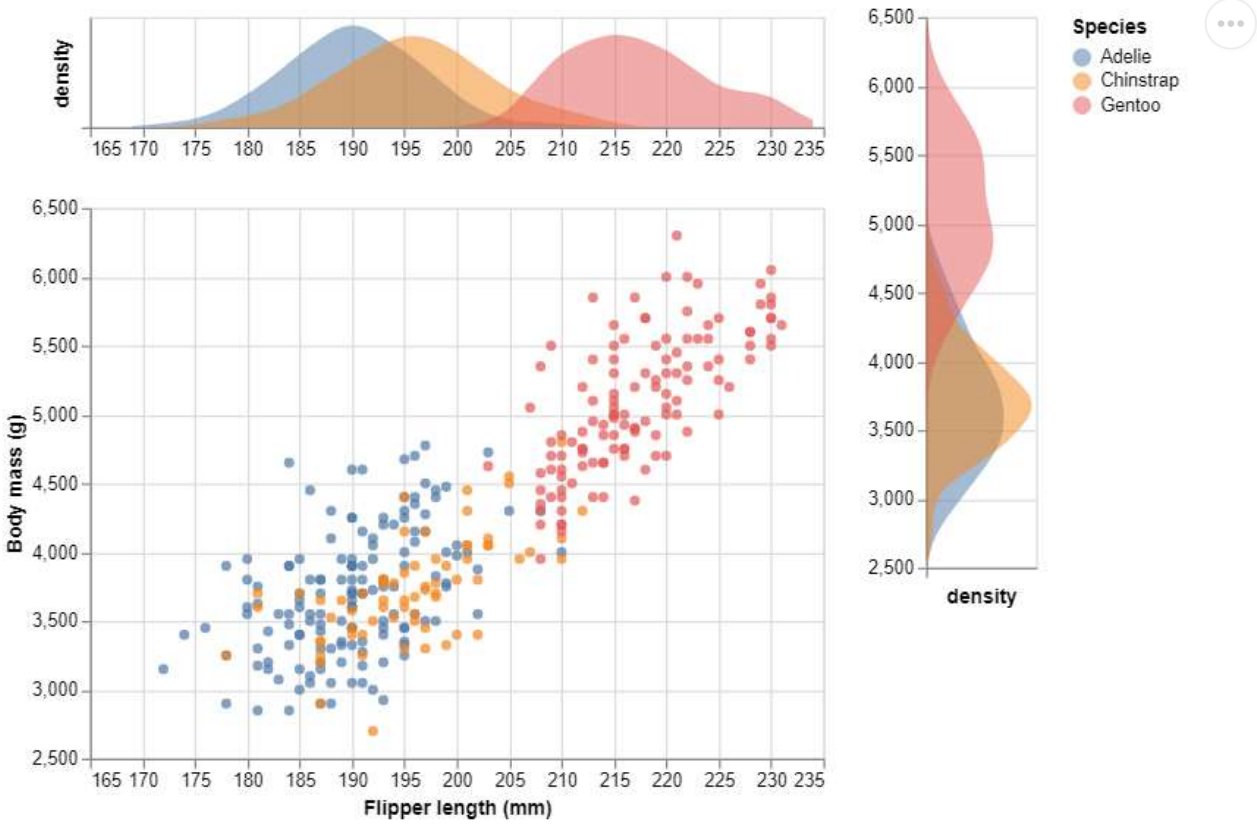
Marginal KDE

```
1 top_KDE & (points | right_KDE)
```



Importance of Parentheses

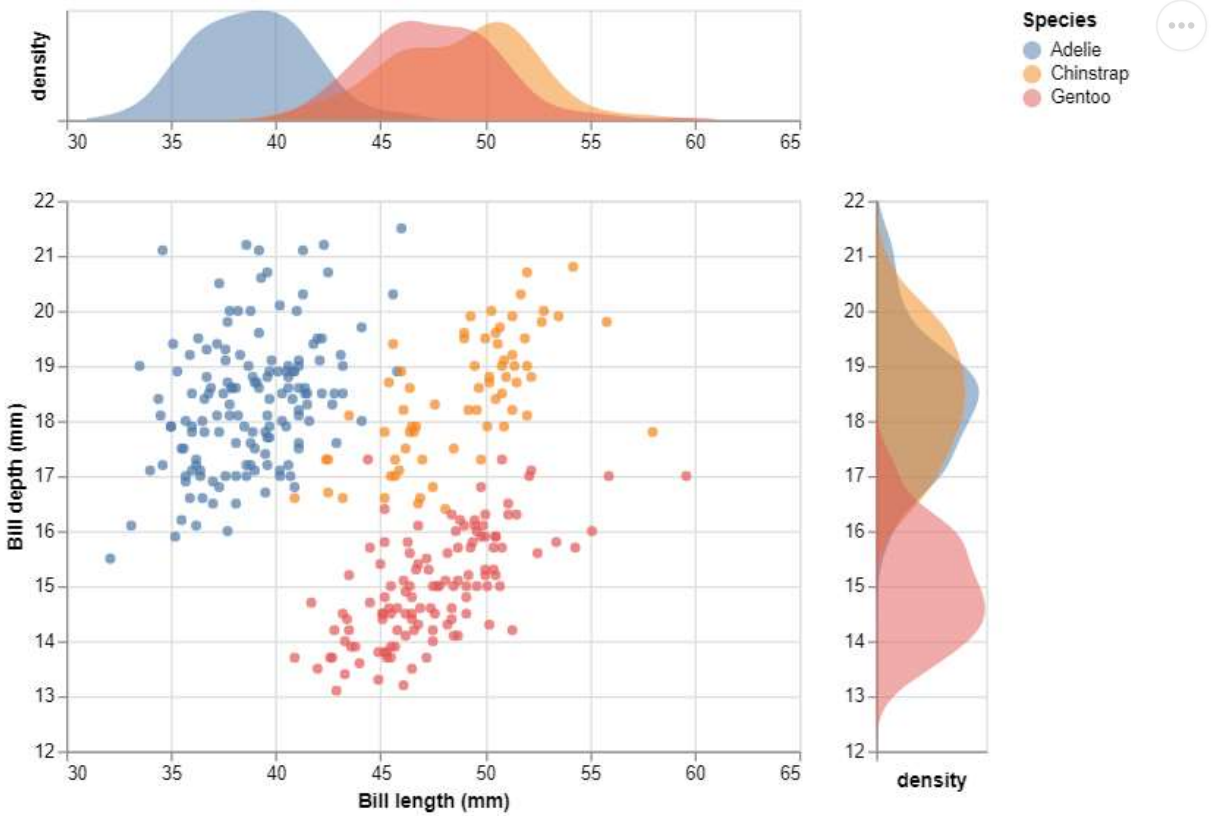
```
1 top_KDE & points | right_KDE
```



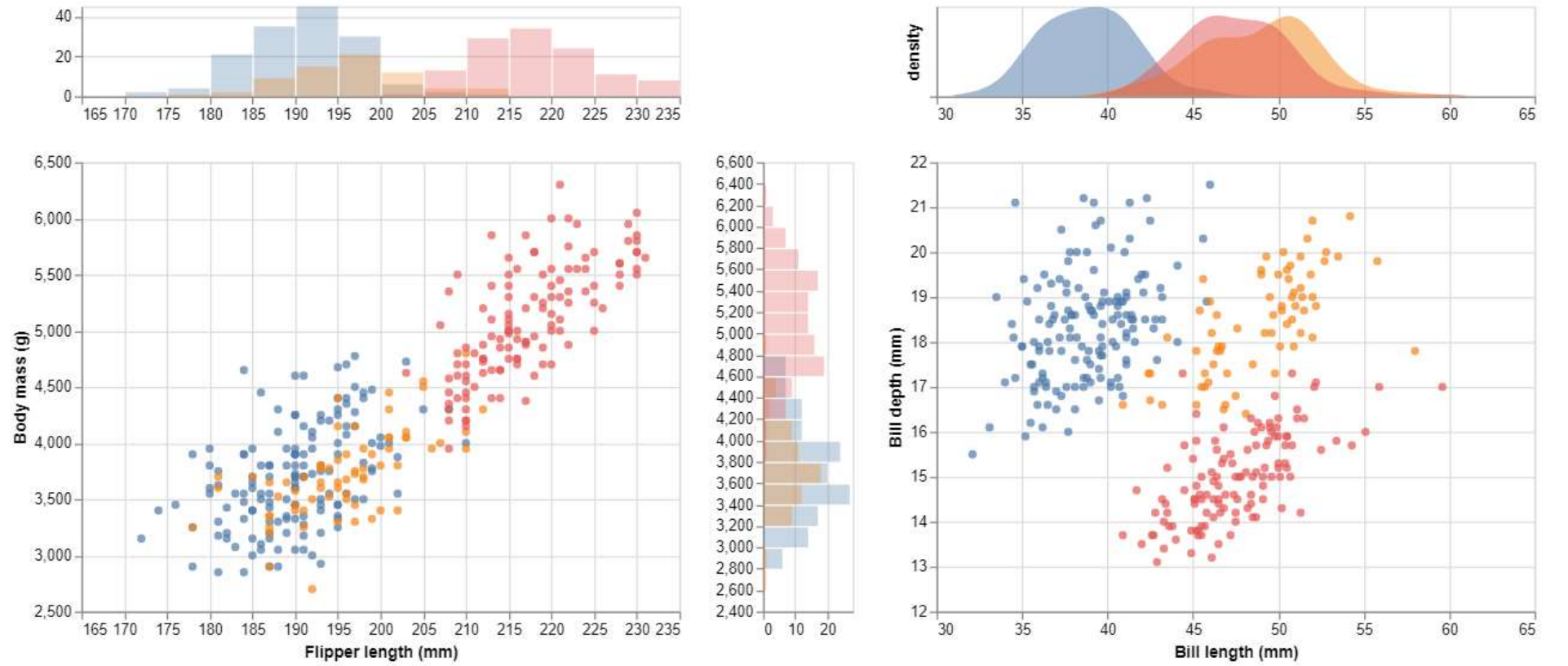
Penguin Plots 2

```
xscale_bills = alt.Scale(domain=(31, 61))
yscale_bills = alt.Scale(domain=(12, 22))


points_bills = alt.Chart(penguins
).mark_circle().encode(
x = alt.X('culmen_length_mm', title='Bill length (mm)', scale = xscale_bill
y = alt.Y('culmen_depth_mm', title='Bill depth (mm)', scale = yscale_bills)
color='species',
)
points_bills


top_KDE_bills = alt.Chart(penguins).transform_density(
  'culmen_length_mm',
  groupby=['species'],
  extent= xscale_bills.domain,
  as_=['culmen_length_mm', 'density']
) mark_area(opacity=0.5).encode(
```

Penguin Plots 2



```
1 ((top_hist & (points | right_hist)) | (top_KDE_bills & (points_bills | righ
```



Making Charts Interactive

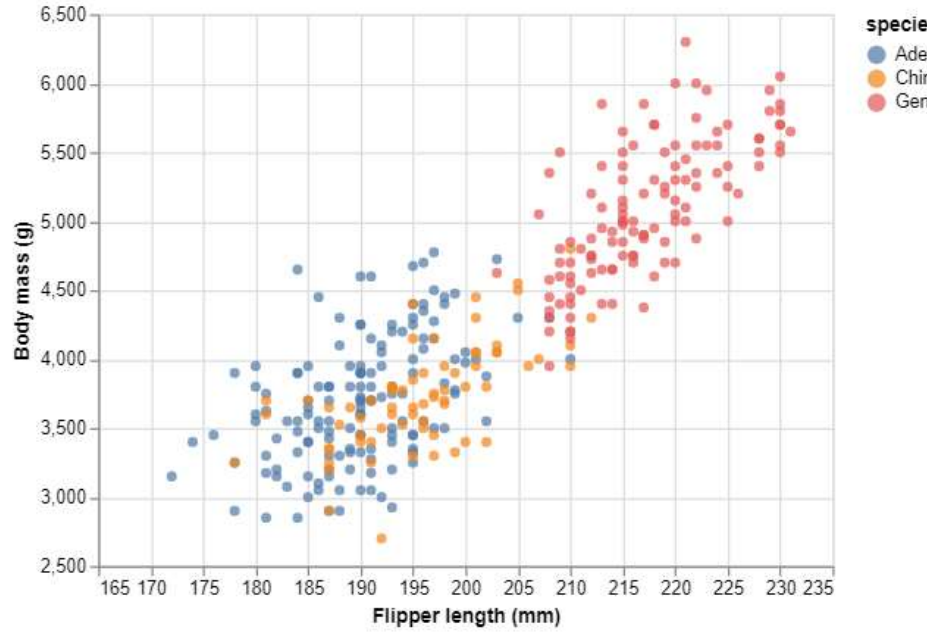
Interactivity

- One of the unique features of Altair is that it goes beyond defining a grammar of graphics, and also defines a grammar of interactivity.
- Interactive chart elements can often be used by your audience to answer additional questions that might come up as they are studying your visualizations.
- We've seen `tooltip` for displaying data details when users hover over marks and the `interactive` method which enables panning and zooming to our charts.

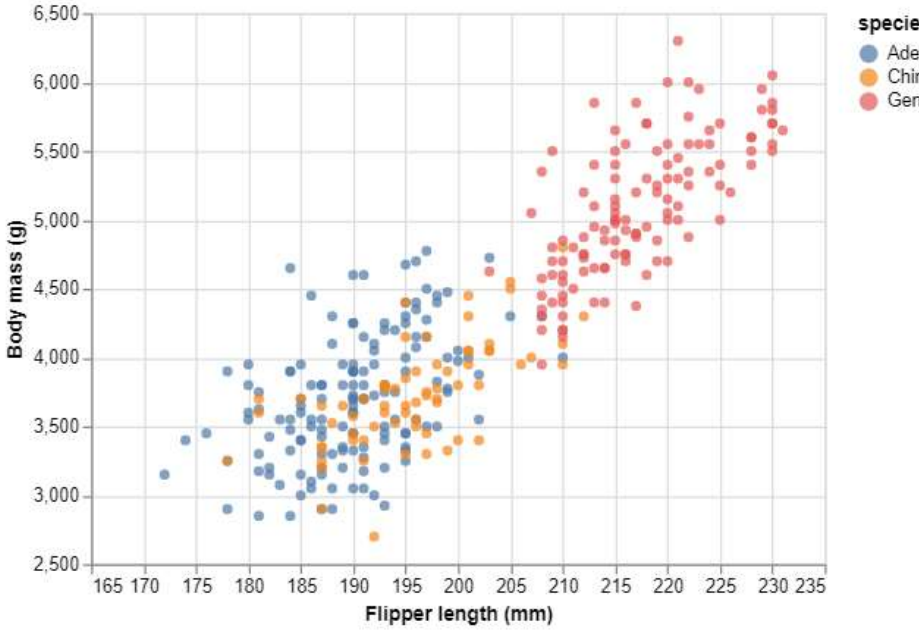
See an example that uses both in the Altair Example gallery

Recall

```
points.encode(
  tooltip=['sex','island']
).interactive()
```




```
scales = alt.selection_interval(bi
points.encode(
  tooltip=['sex','island']
).add_selection(scales)
```



Grammar of *Interactivity*

- the `selection()` function captures interactions from the mouse (e.g. clicks or drags) or through other inputs to effect the chart (eg. drop-down menu, sliders).
- the `condition()` function takes the selection input and changes an element of the chart based on that input.
- the `bind` property establishes a two-way binding between the selection and an input element of your chart.

Interactive charts can use one or more of these elements to create rich interactivity between the viewer and the data.

Selection Types

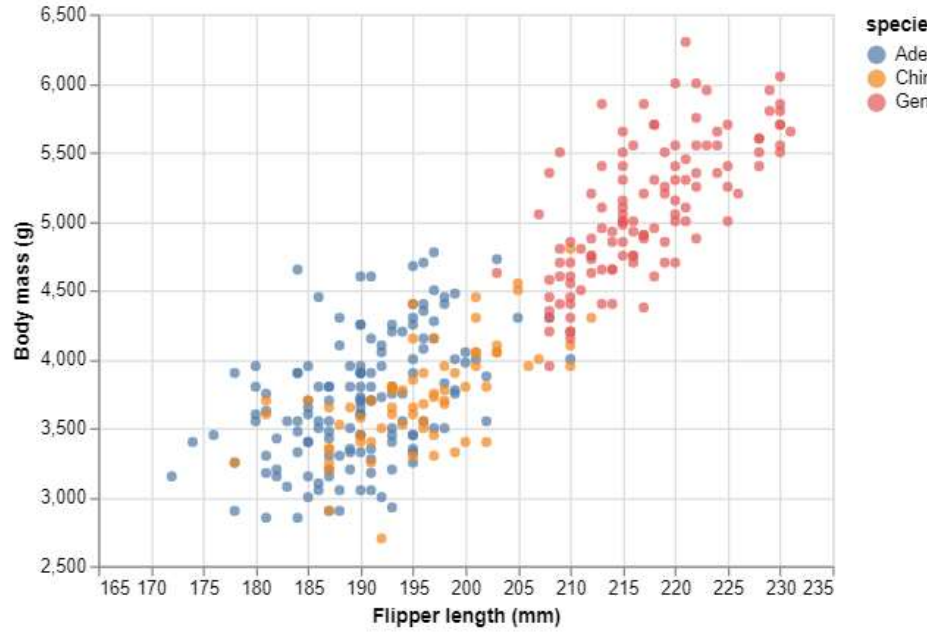
1. `selection_interval()` allows you to select chart elements by clicking and dragging.
2. `selection_single()` allows you to select a single chart element at a time using mouse actions.
3. `selection_multi()` allows for multiple chart objects to be selected at once (by default add/remove by selecting while holding the *shift* key)

We'll start with intervals and return to the other options later.

Can also be specified `selection(type =`*interval/single/multi*`)`

Selection

```
1 brush = alt.selection_interval()
2 points.add_selection(brush)
```



You will see now that you can make selection boxes (which can be dragged across the chart)

We can now bind this brush to our chart by setting the `selection` property

*this doesn't actually **do** anything until we reference it in some way within the chart.*

Note that selecting points with an interval is often referred to as "brushing" in data

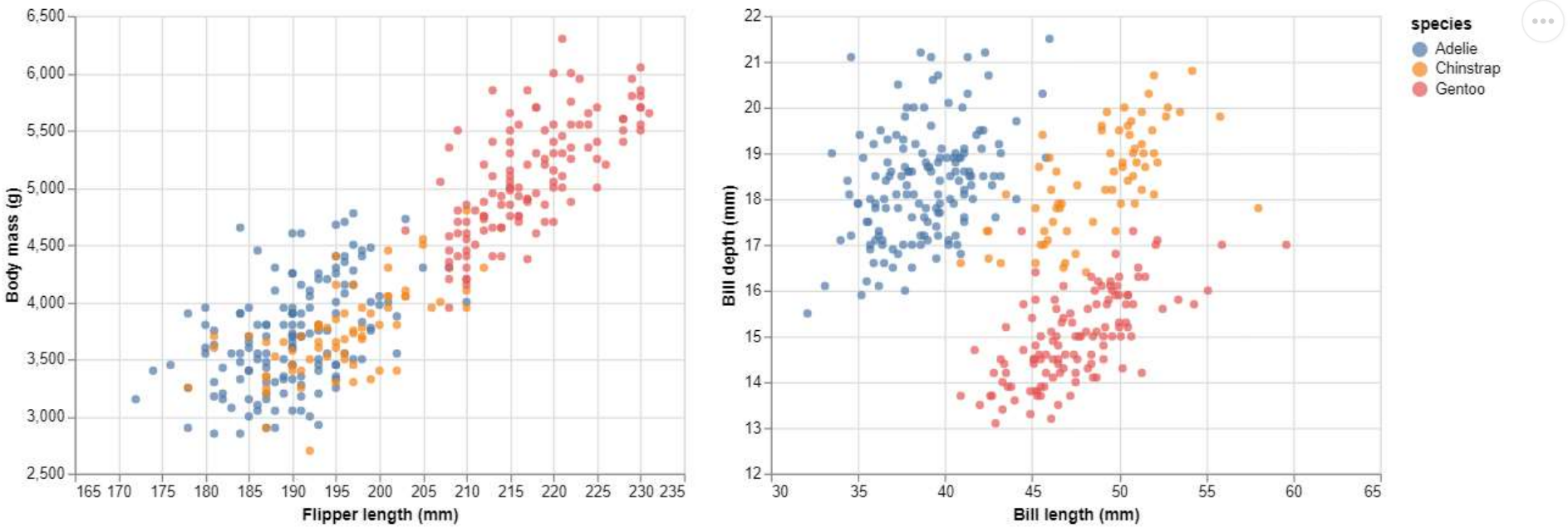
Conditions: Making the chart respond

Linking Charts

This approach becomes even more powerful when the selection behavior is tied across multiple views of the data within a compound chart.

```
chart2 = points_bills.encode(
  color=alt.condition(
    brush, # condition
    'species',  # if True
    alt.value('lightgray') # if False
  )
).add_selection(brush)


chart | chart2
```

Linking Charts



Select Interval Option

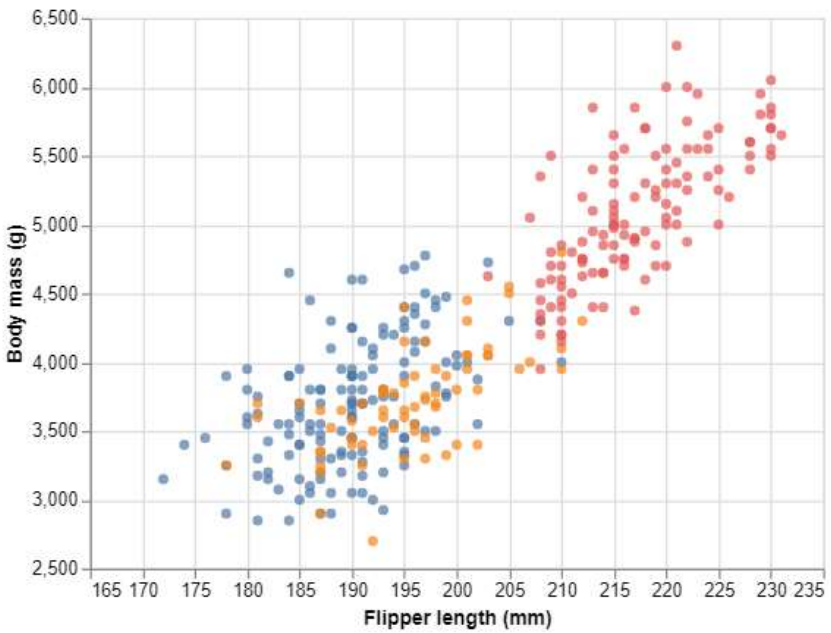
In addition to `bind='scales'`[1], `selection_interval` takes a few additional arguments;

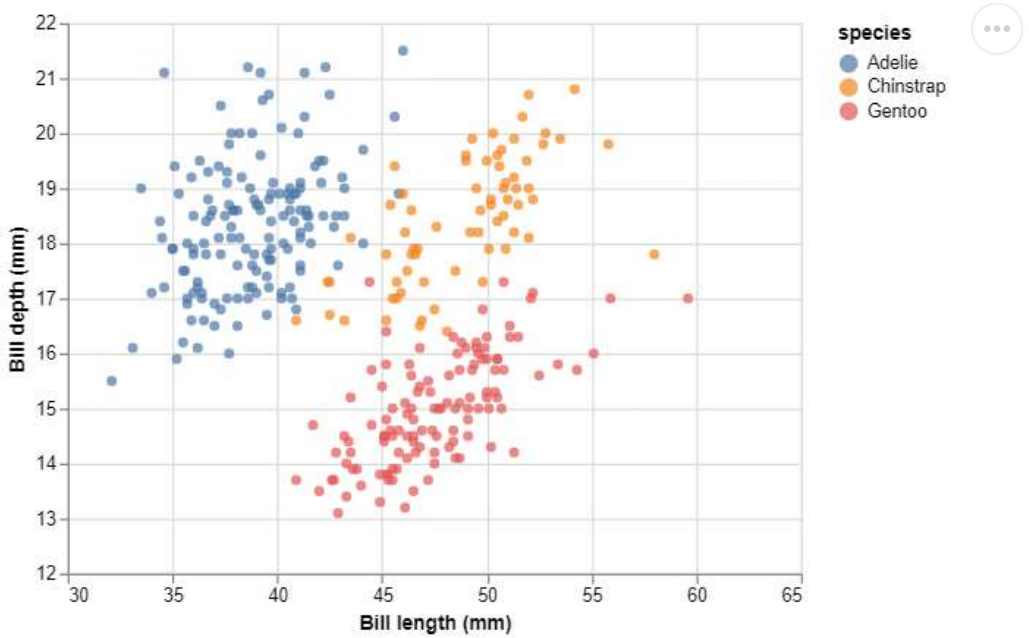
- `resolve='intersect'` creates separate intervals for each chart and selects only points in the intersection
- `encodings=['x']` selections are made only according to the x-axis
- `nearest` by setting to `True` we can select points on mouseover rather than on click.

1. this is equivalent to `chart.interactive()`

Intersection

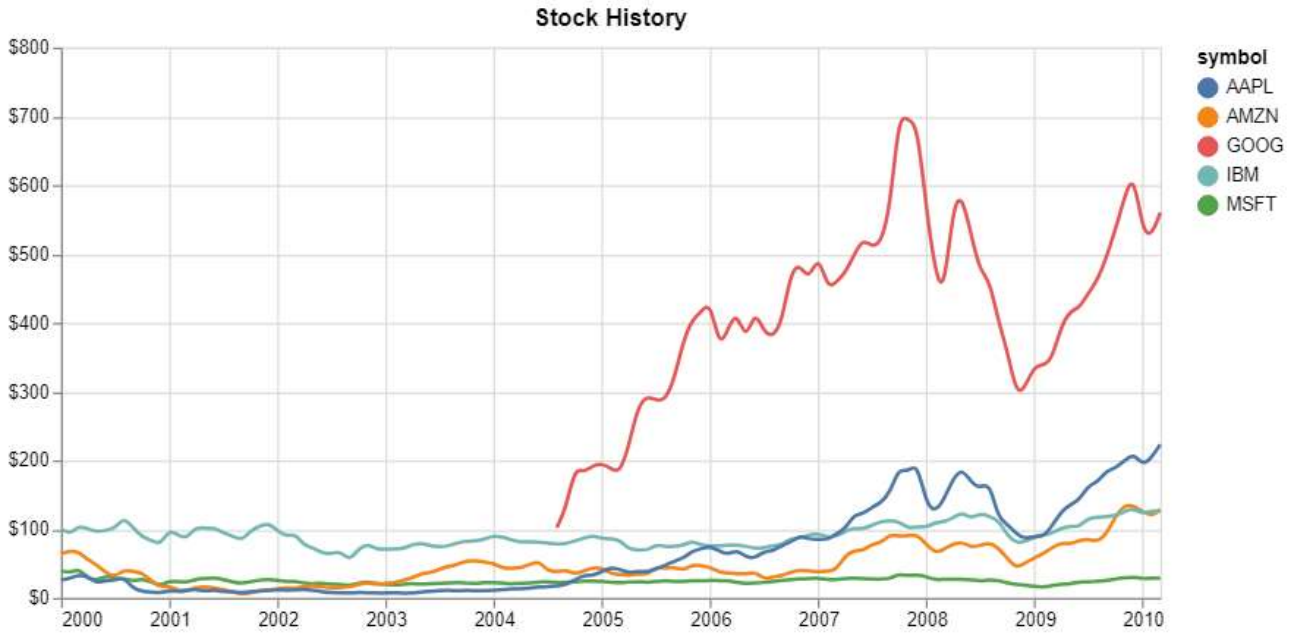
▶ Code





Selection with x Encoding

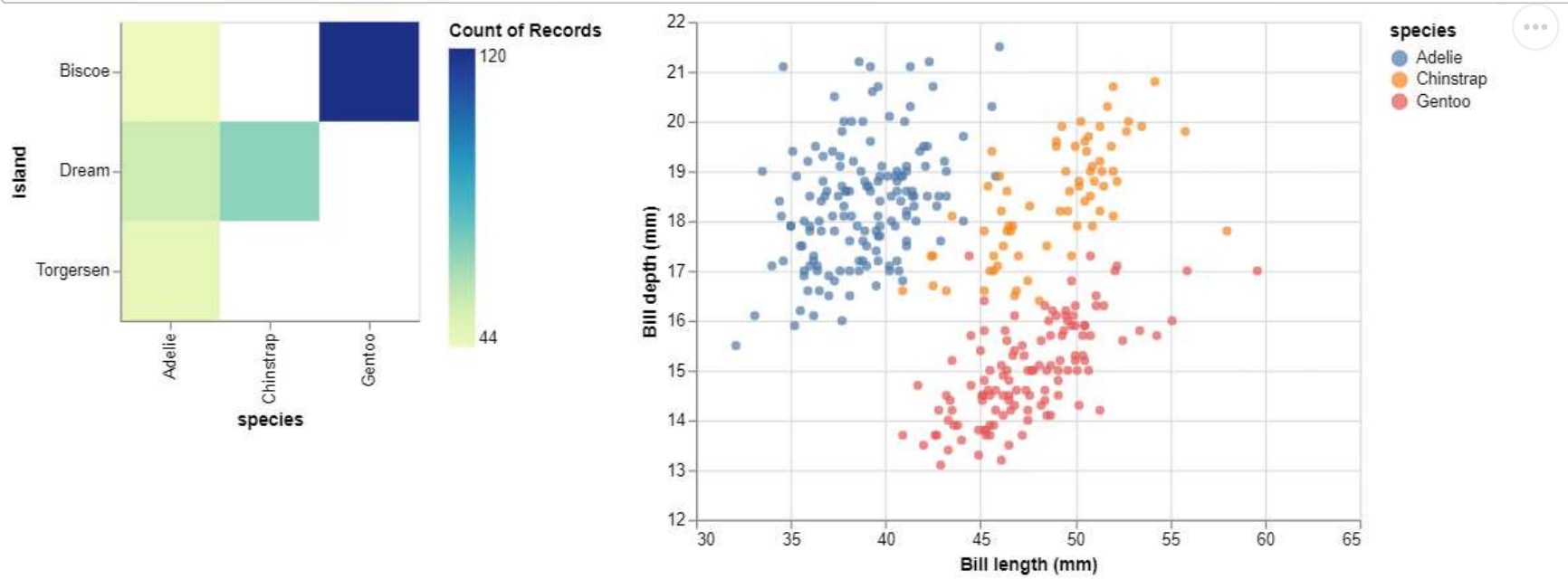
▶ Code



Interval Selection

Plot | Heat Map | Scatter plot

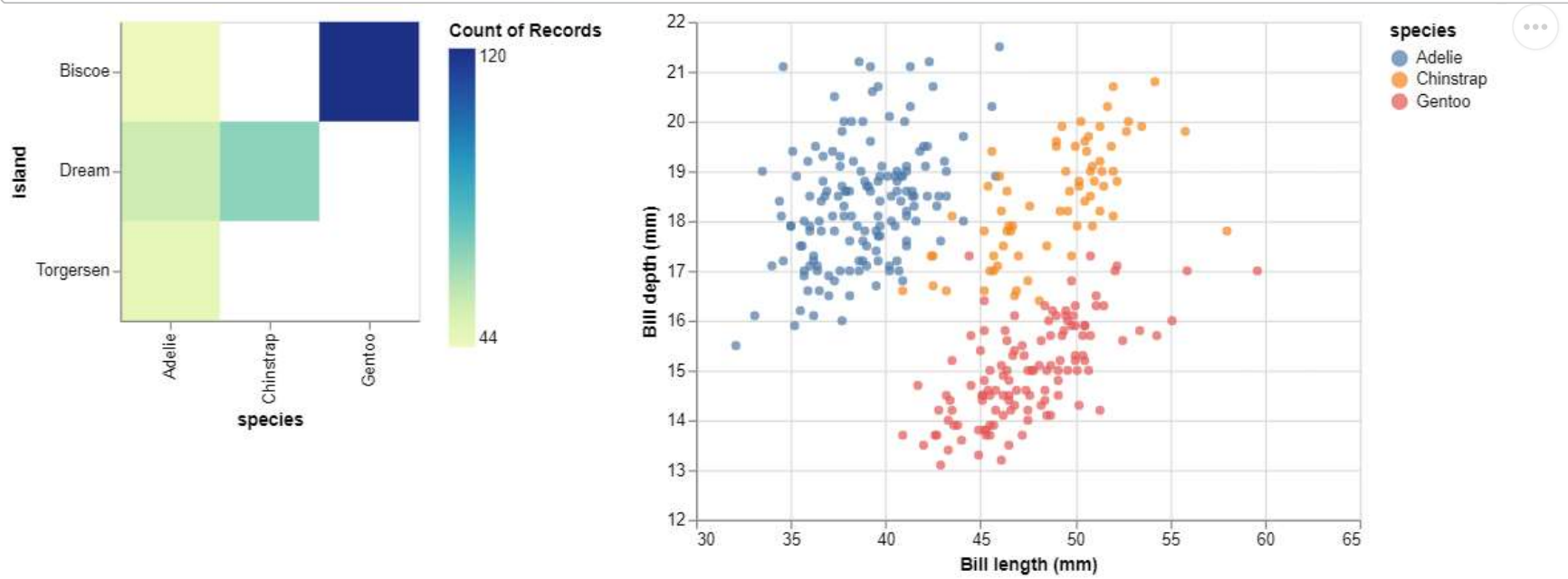
```
1 int1 | int2
```



Single Selection

Plot | Heat Map | Scatter plot

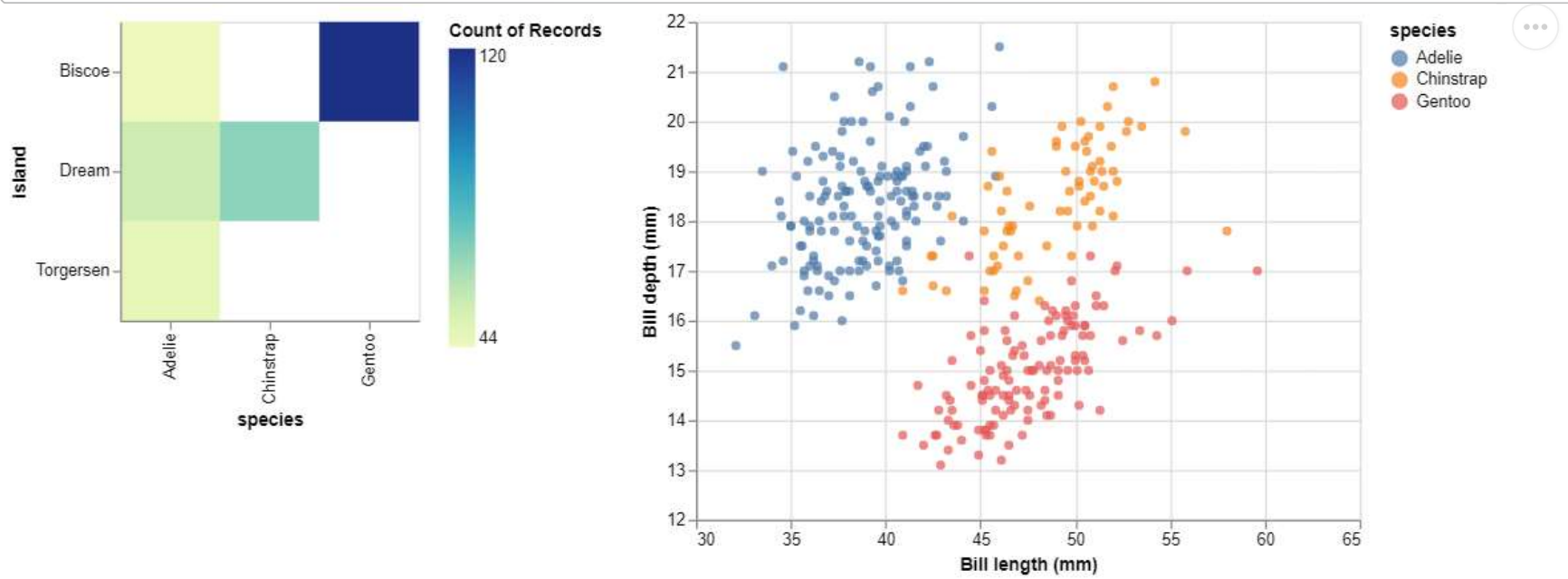
```
1 sing1 | sing2
```



Multiple Selection

Plot | Heat Map | Scatter plot

```
1 multi1 | multi2
```

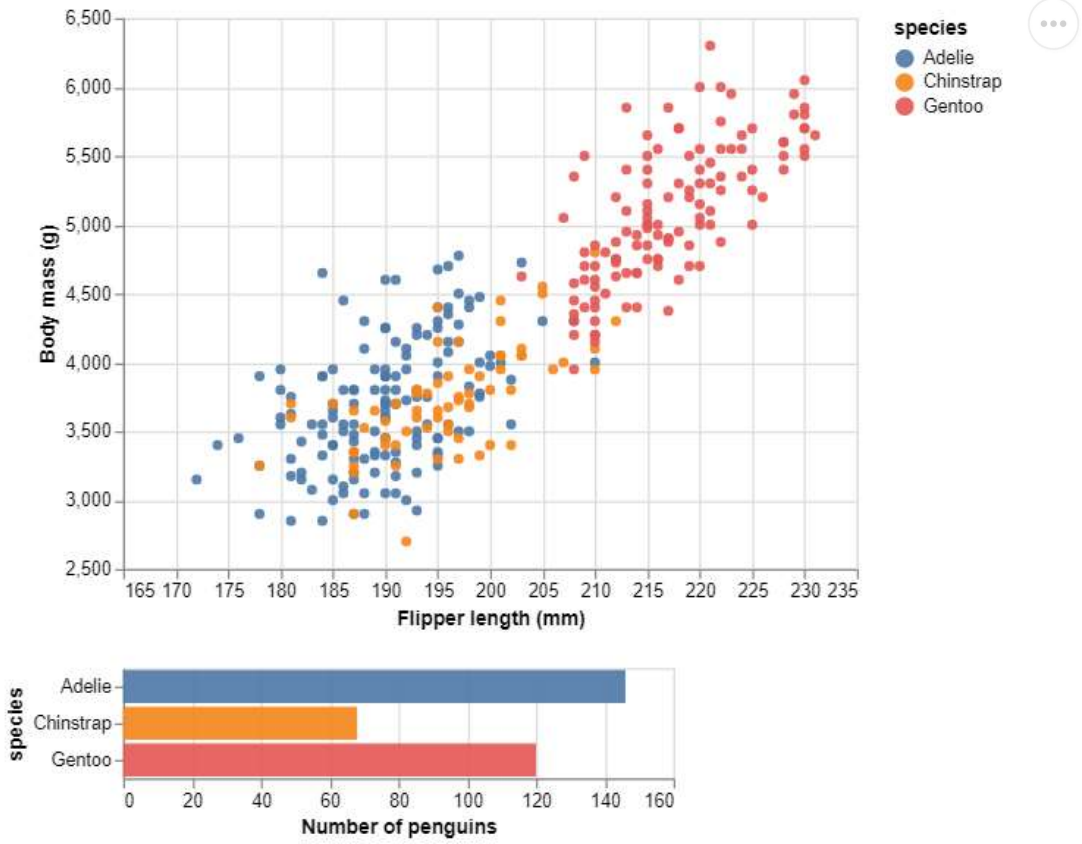


Comments

- For any but the simplest selections, the user needs to think about exactly what is targeted by the selection
- This can be controlled with either the `fields` or `encodings` arguments.
- These control what data properties are used to determine which points are part of the selection.

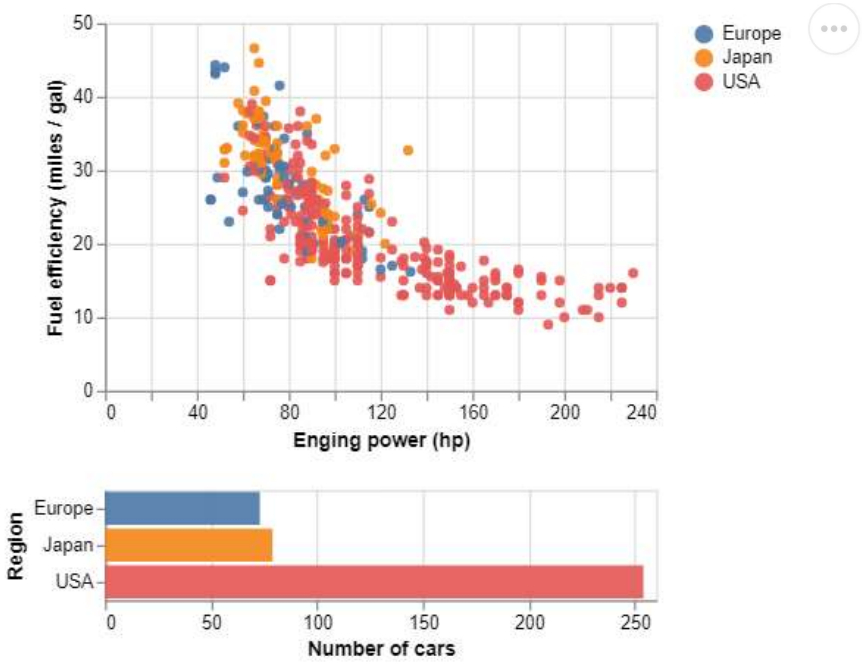
Binding to legend 2

- Code



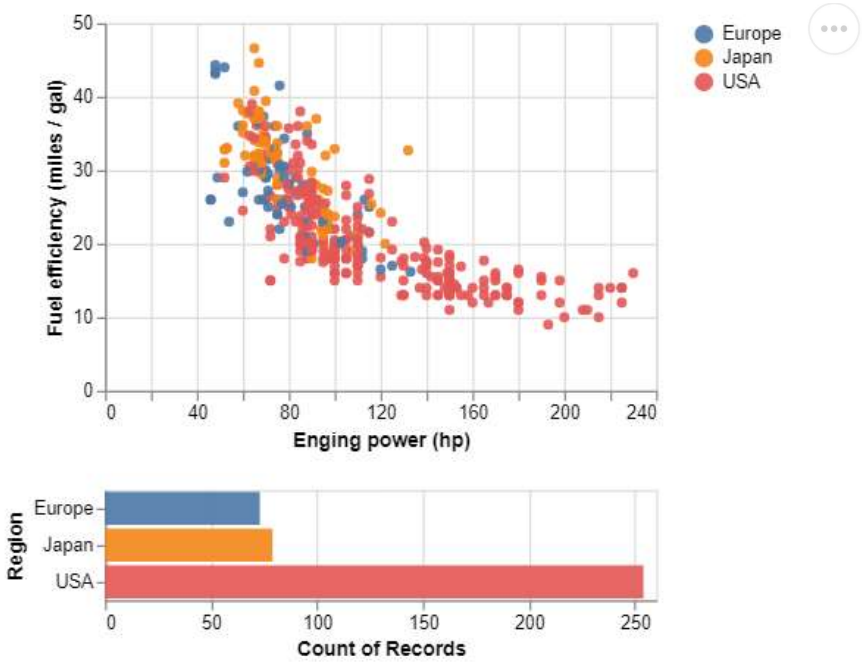
Conditional Filtering

▶ Code



Conditional Filtering (Fixed axis)

▶ Code



Sharing Altair Visualizations

Saving as HTML

```
import altair as alt
from vega_datasets import data

cars = data.cars.url
chart = alt.Chart(cars
).mark_circle().encode(
    alt.X('Horsepower:Q'),
    alt.Y('Miles_per_Gallon:Q')
)

chart.save('cars-scatter.html')
```

- The easiest way to save Altair charts is as an HTML document via the `save` method.
- An HTML file then can either be incorporated in a web page or sent to anyone via email.
- When someone opens an HTML file containing an Altair chart in a web browser they will see the visualization, including any interactivity you have used!