Data 550: Data Visualization I

Lecture 5: Designing plots for Communication

Dr. Irene Vrbik

University of British Columbia Okanagan

Introduction

- Today we will focus on some best practices for effective figure design for communication
- We will implement these principles using Altair.
- Some suggested supplementary readings from *Fundamentals of Data Visualization by Claus O. Wilke*
  - 22 Titles, captions, and tables
  - 24 Use larger axis labels

Lecture Objectives

- Discuss guidelines and best practices in visualization design.
- Directly label markings instead of using legends.
- Adjusting Axis Limits and formatting
- Modify titles of figure elements.

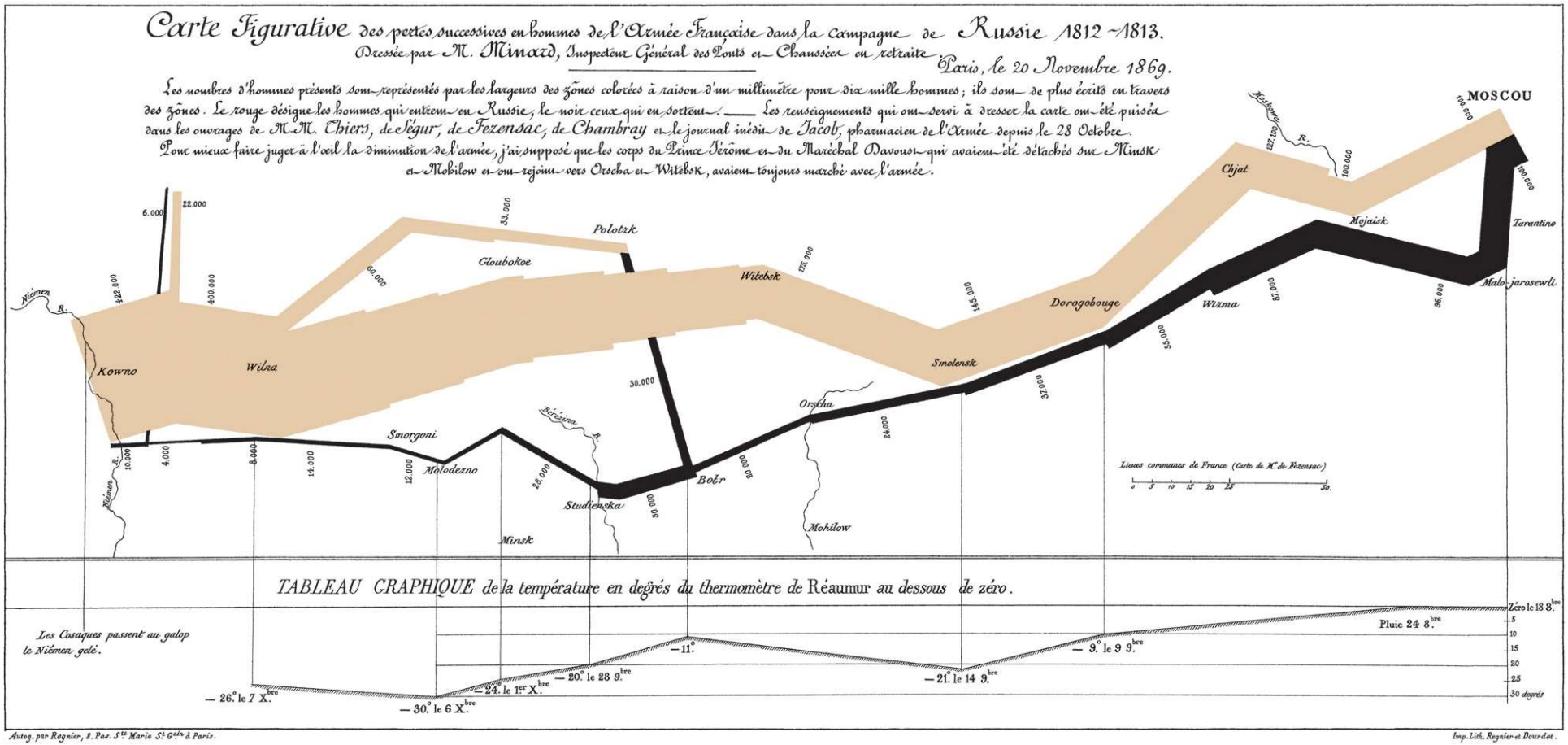
Quote

> Graphical excellence is the well-designed presentation of interesting data —a matter of substance, of statistics, and of design … [It] consists of complex ideas communicated with clarity, precision, and efficiency. … [It] is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space … [It] is nearly always multivariate… And graphical excellence requires telling the truth about the data.

Edward R. Tufte

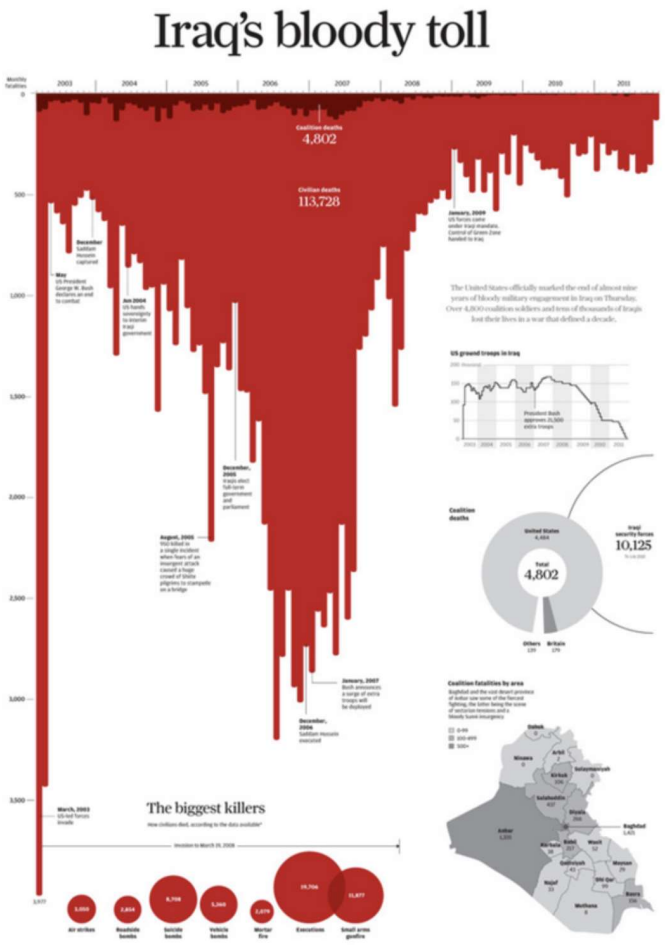
Page 51 of *The Visual Display of Quantitative Information (1983)*

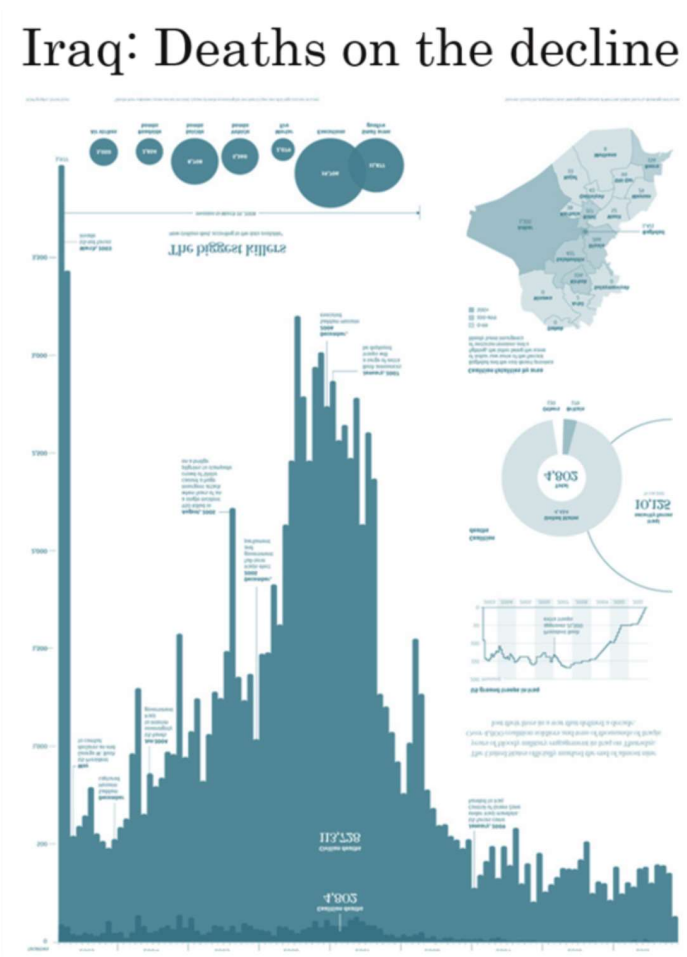
Napoleon’s retreat from Moscow



Joseph Minard’s visualization of Napoleon’s retreat from Moscow. Justifiably cited as a classic, it is also atypical and hard to emulate in its specifics.

Data can tell multiple stories

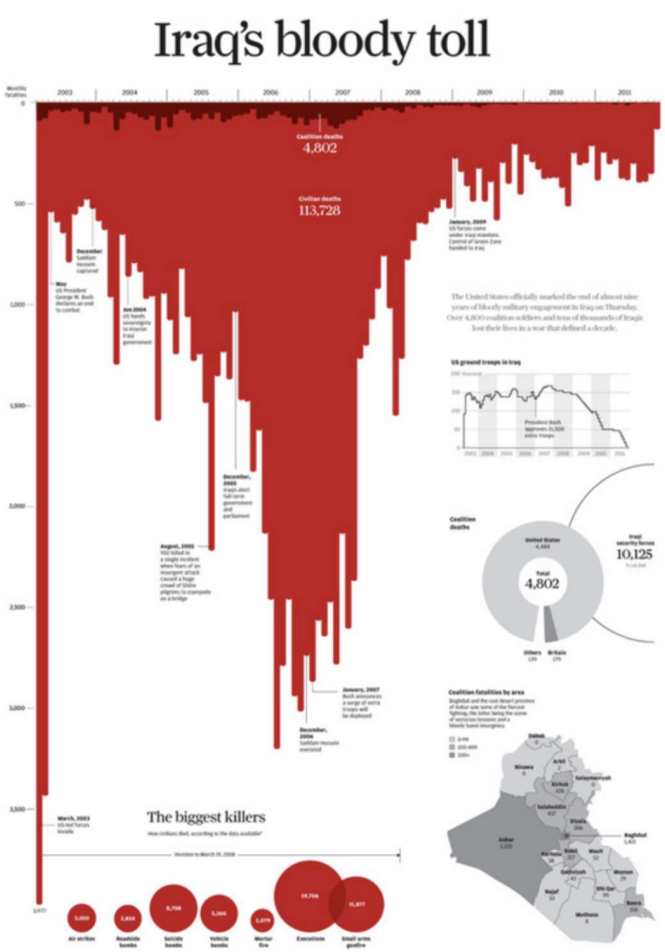




Source: South China Morning Post in late 2011, designed by Simon Scarr (click to see more detail)

Comparisons from Infoworld

Iraq's blood toll



- Title: establishes author's agenda
- Color: red for blood
- Orientation: impression of dripping blood

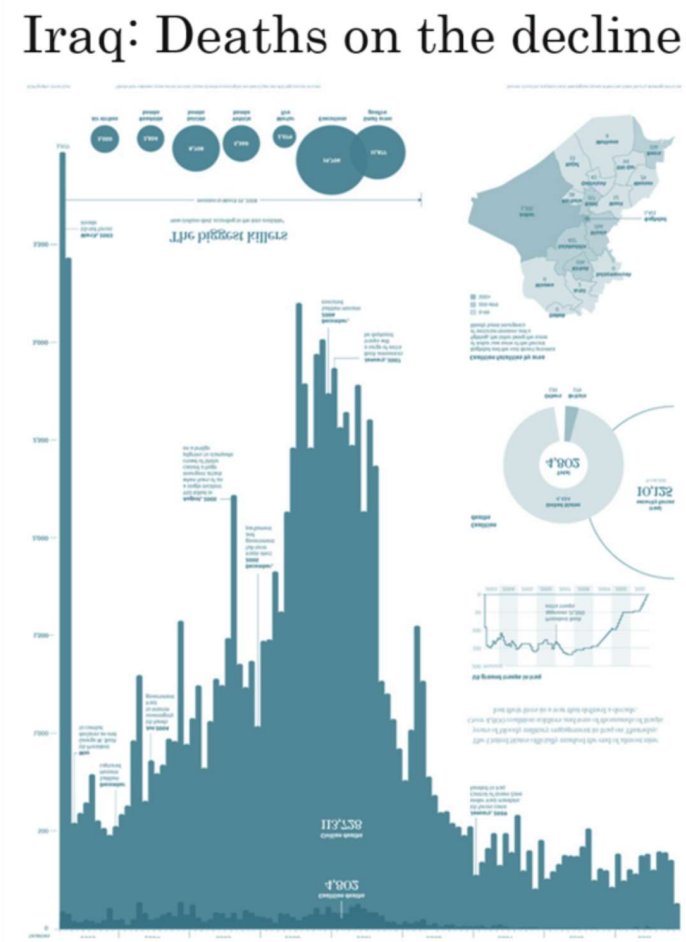
This piece of work won silver at Malofiej, an annual infographics awards conference, in 2012.

Iraq: Deaths on the decline

- With bars pointing upward it look a little more optimistic.

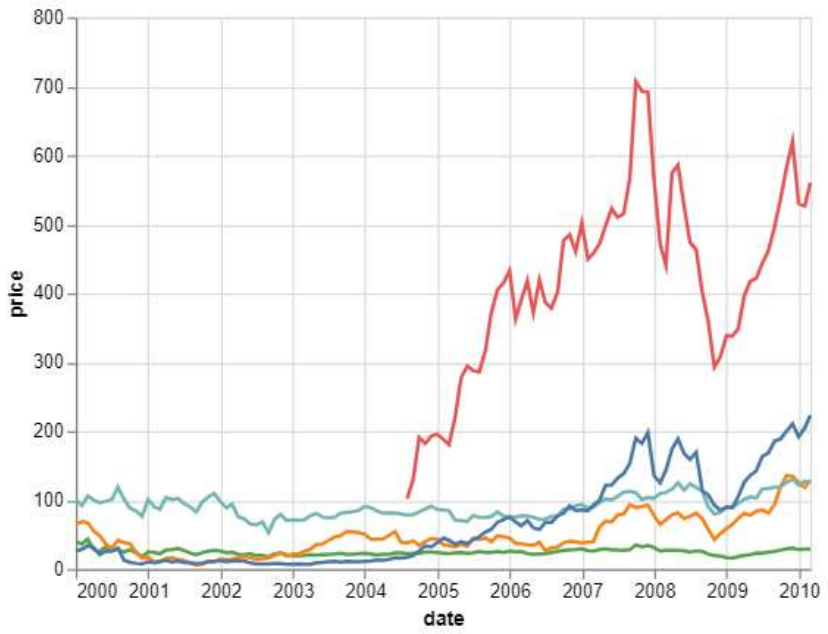
These chart present very different narratives.

Neither of these opinionated charts are inaccurate



Descriptive titles and labels

Untitled Charts



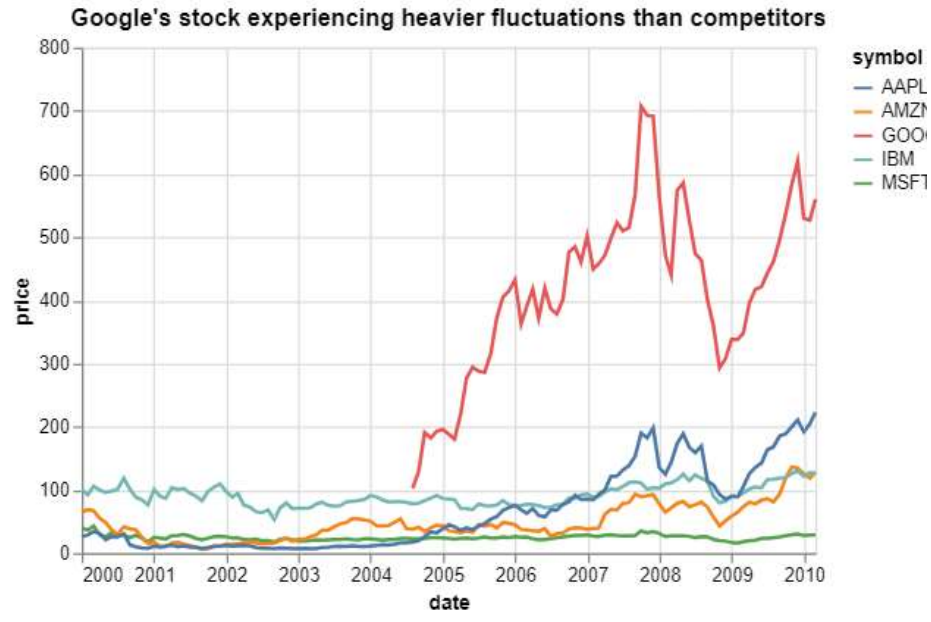
Other problems:

- The y-axis label is not very descriptive
- With no legend it is impossible to tell what this figure is showing!

Charts without titles are hard to interpret

Chart Titles

▶ Click to show the code



The **title** should answer the question you posed before making the visualization.

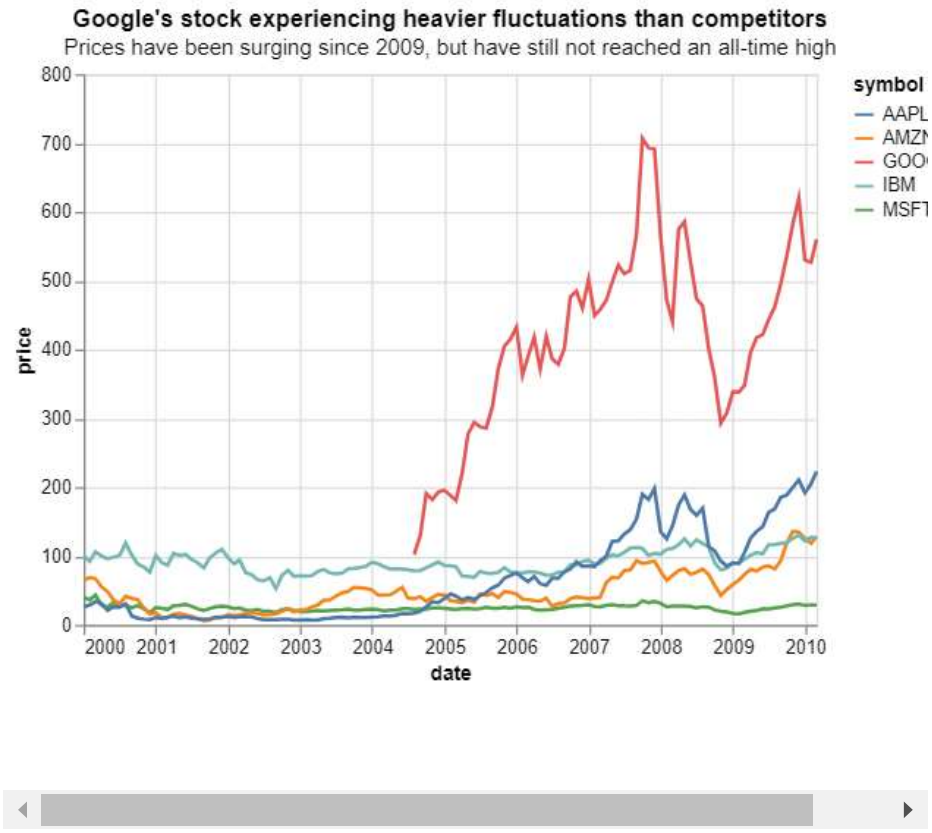
A less effective title: *Stock prices over time*

Alternative narrative: *Google's stock outperforms competitors in 2009*

Subtitle

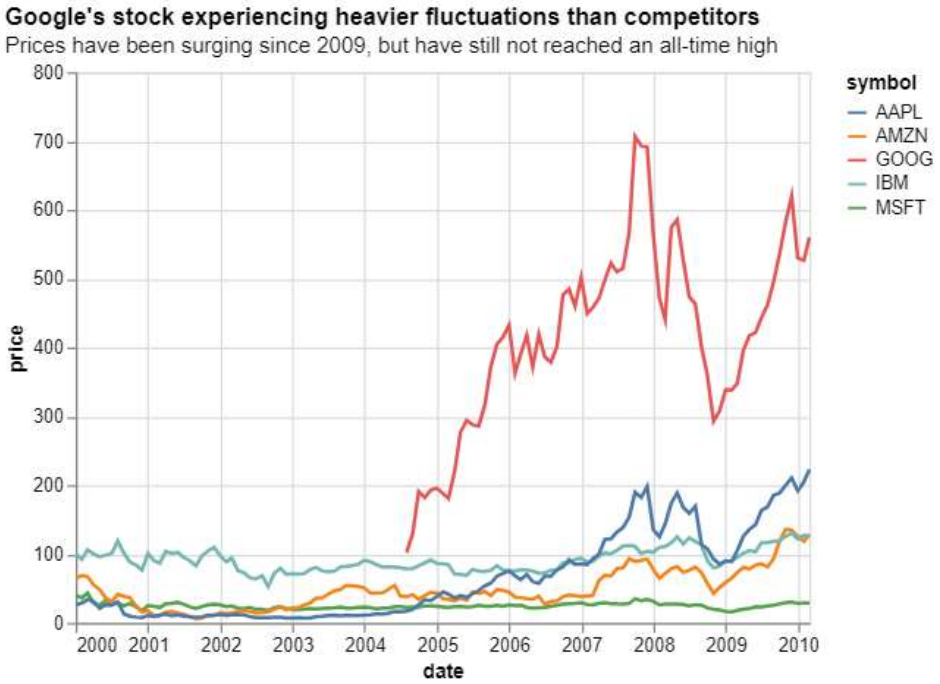
To add additional details we can use `TitleParams` to create the main/sub title which we pass to the title Chart parameter.

```
main_title = "Google's stock exper
sub_title = 'Prices have been surg

alt.Chart(stocks,
title=alt.TitleParams(
    text = main_title,
    subtitle = sub_title)
).mark_line(
).encode(
  alt.X('date'),
  alt.Y('price'),
  color='symbol'
)
```



Subtitle Aligned

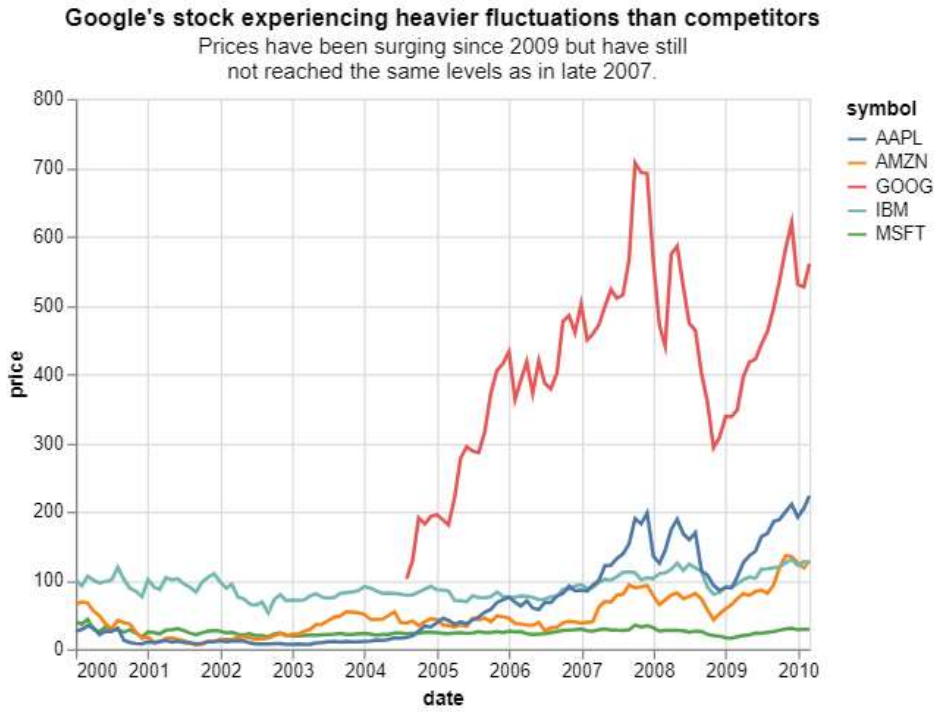
```
alt.Chart(stocks, title=alt.TitleParams(
    text = main_title,
    subtitle = sub_title,
    anchor='start') # to aligning sub and main title text
).mark_line().encode(alt.X('date'), alt.Y('price'), color='symbol')
```



If it makes sense to align your titles, you can set an anchor

Multiline titles

- Show the code

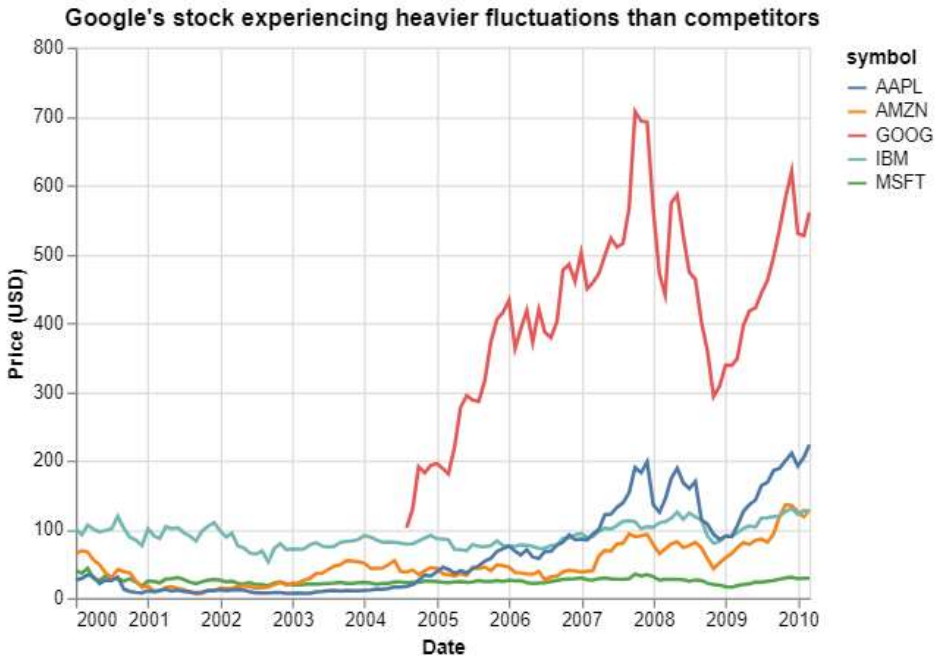


Altair converts lists of strings into multiline titles.

If we have a really long title, we can improve readability by breaking it into multiple lines.

Axis titles

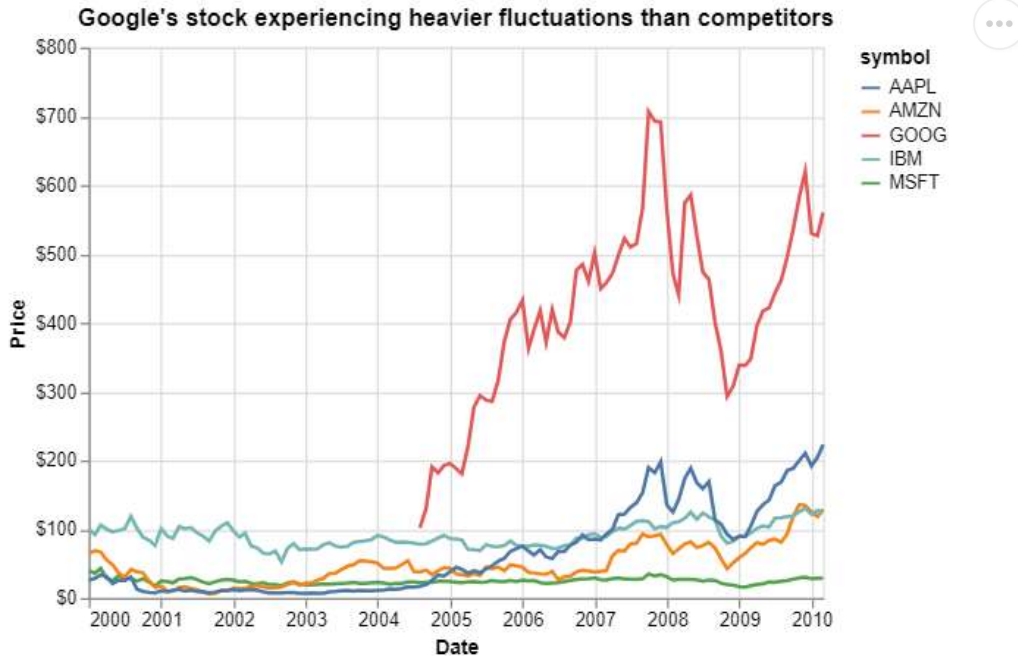
- Show the code



Axis titles should be capitalized regular words rather than dataframe column names

Units on axis

▶ Show the code



Units can be incorporated directly in the axis labels instead of the axis title

s is short for the International System of Units, known SI for short

Axis formating

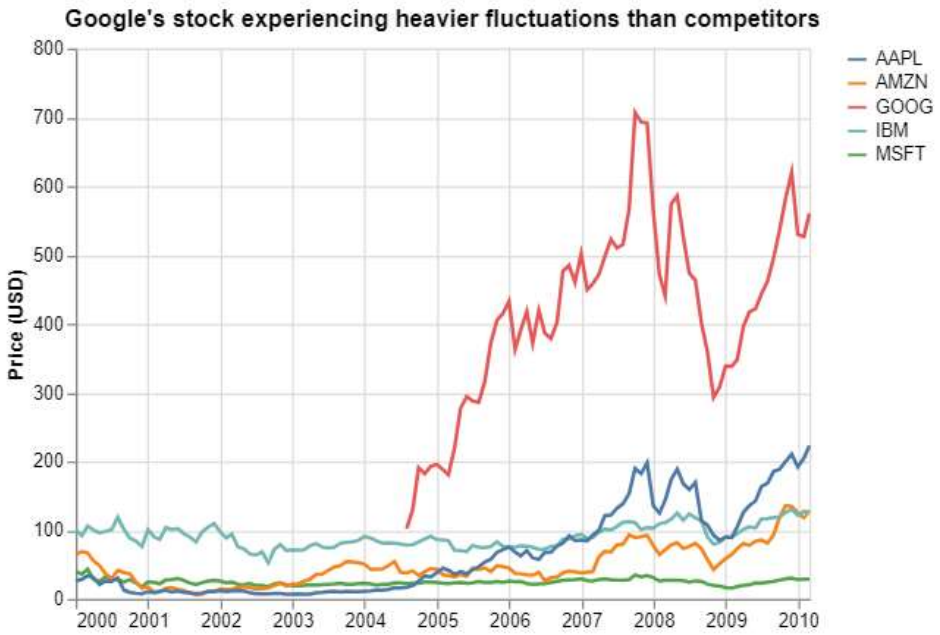
Other useful formatting string include:

- % for including a percentage sign,
- e to force a scientific format,
- d to force integer format,
- ~ which removes trailing zeros (e.g. 1.0 becomes 1), and
- , which adds a comma as the thousands separator
- s convert values to the units of the appropriate SI Prefix

All label format strings can be found here.

Remove labels for Temporal and Categorical Variables

- Show the code



Legend and axis titles are often redundant for categorical and temporal variable and can be removed

Direct labelling of categories

Direct labelling is often preferred over legend when applicable.

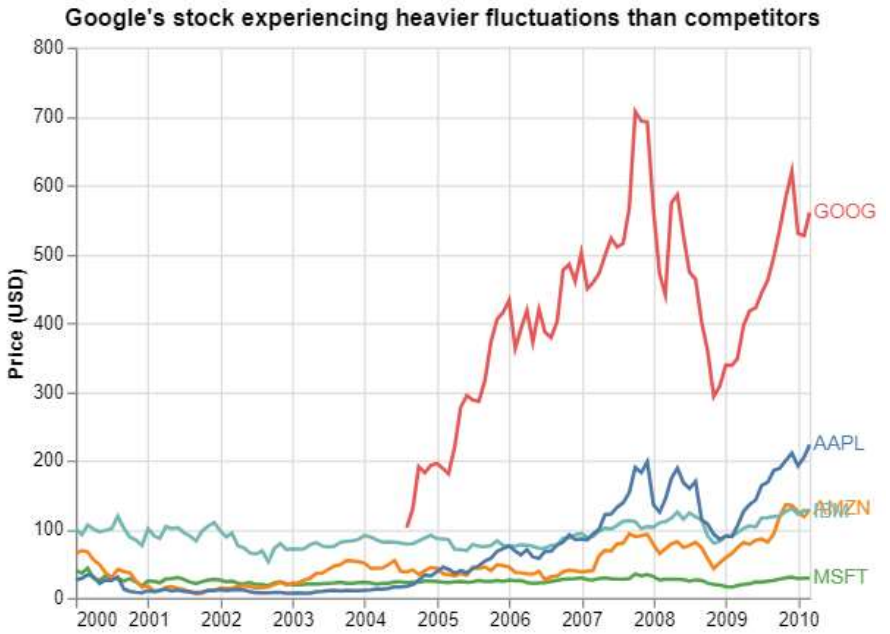
We create a data frame of points where our labels should go:

```
stock_max_date = stocks[stocks['date'] == stocks['date'].max()]
stock_max_date.head()
```

	symbol	date	price
122	MSFT	2010-03-01	28.80
245	AMZN	2010-03-01	128.82
368	IBM	2010-03-01	125.55
436	GOOG	2010-03-01	560.19
559	AAPL	2010-03-01	223.02

This is similar to our text labelling example

▶ Show the code



However, the `symbol` of the turquoise and yellow (IBM and Amazon) lines are not readable due to the overlap.

Text Mark Properties

As documented in the mark properties, we can set …

Property	Type	Description
align	anyOf(**`Align`**	The horizontal alignment of the text or ranged marks (area, bar, image, rect, rule). One of `"left"`, `"right"`, `"center"`.

dx	anyOf(`number`, [**`ExprRef`**](https://altair-viz.github.io/user_guide/generated/core/altair.ExprRef.html#altair.ExprRef "altair.ExprRef"))	The amount by which to offset the placement of text on the x-axis
dy	anyOf(`number`, [**`ExprRef`**](https://altair-viz.github.io/user_guide/generated/core/altair.ExprRef.html#altair.ExprRef "altair.ExprRef"))	The amount by which to offset the placement of text on the y-axis

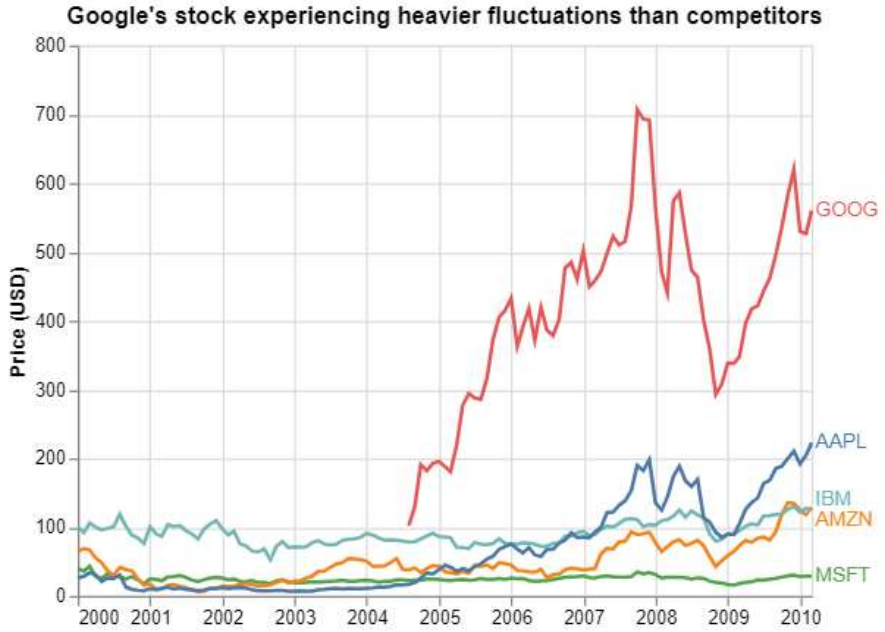
Overlapping text

There is no way to automatically separate the labels in Altair, so we have to manually specify new y-axis values for the text of the overlapping labels.

```
stock_max_date.loc[stock_max_date['symbol'] == "IBM", 'price'] = 140
stock_max_date.loc[stock_max_date['symbol'] == "AMZN", 'price'] = 110
```

	symbol	date	price
122	MSFT	2010-03-01	28.80
245	AMZN	2010-03-01	110.00
368	IBM	2010-03-01	140.00
436	GOOG	2010-03-01	560.19
559	AAPL	2010-03-01	223.02

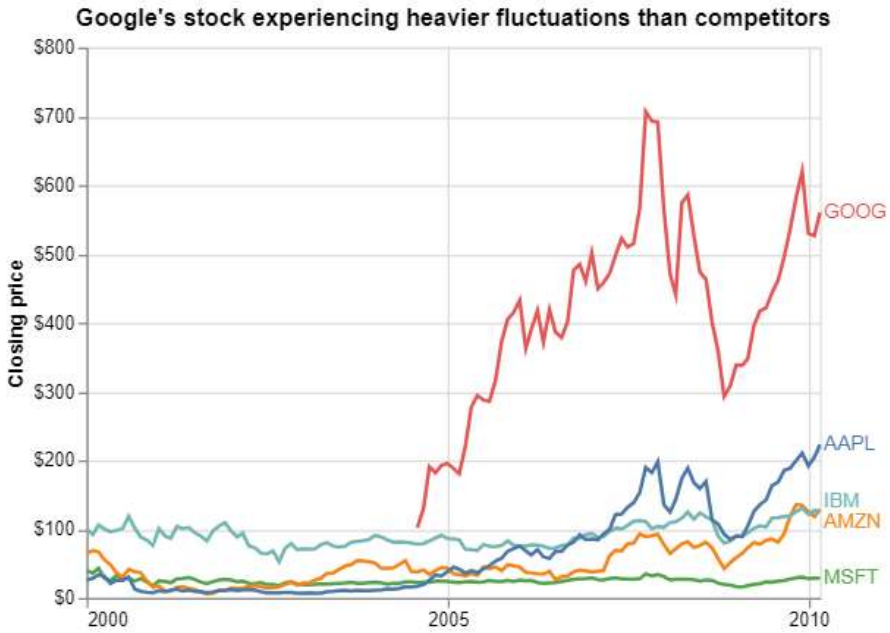
```
texts = alt.Chart(stock_max_date).mark_text(
  align='left',
  dx=2
  ).encode(
    x='date',
    y='price',
    text='symbol',
  color=alt.Color('symbol', legend=None))

lines + texts
```



23

Removing some x-axis ticks

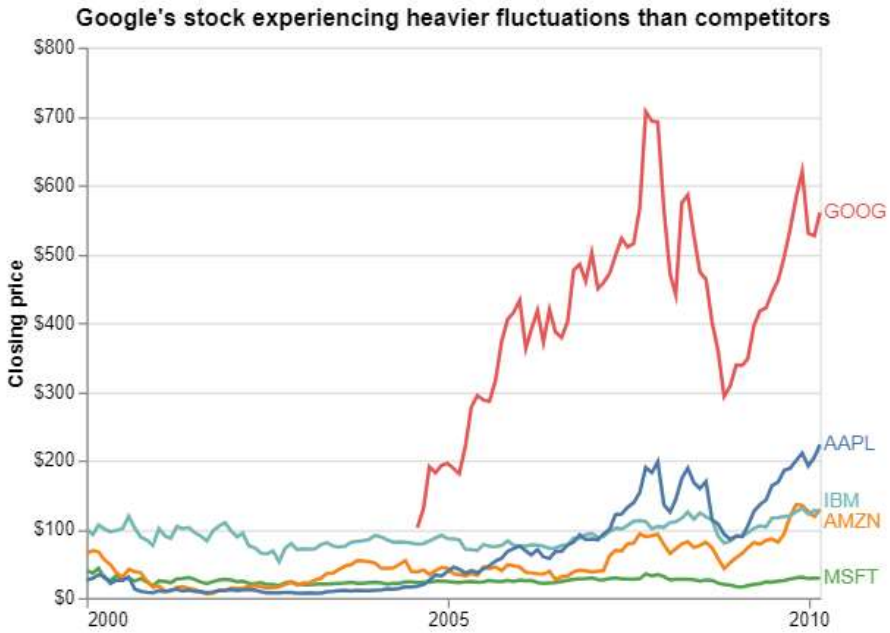
- Show the code



In order to achieve a specific effect, we might reduce the number of ticks along the x-axis

Removing gridlines

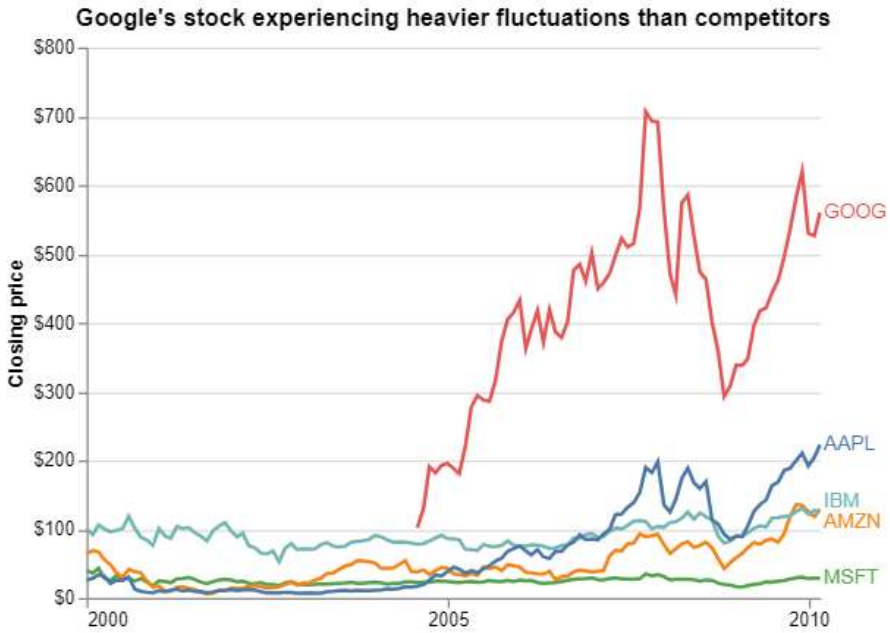
- Show the code



To turn off the vertical (resp. horizontal) gridlines, set `grid` parameter to `False` in alt.X (resp. alt.Y).

Removing Box

- Show the code



To remove the gray box outline set the `strokeWidth` of the layered chart to 0 with `configure_view`

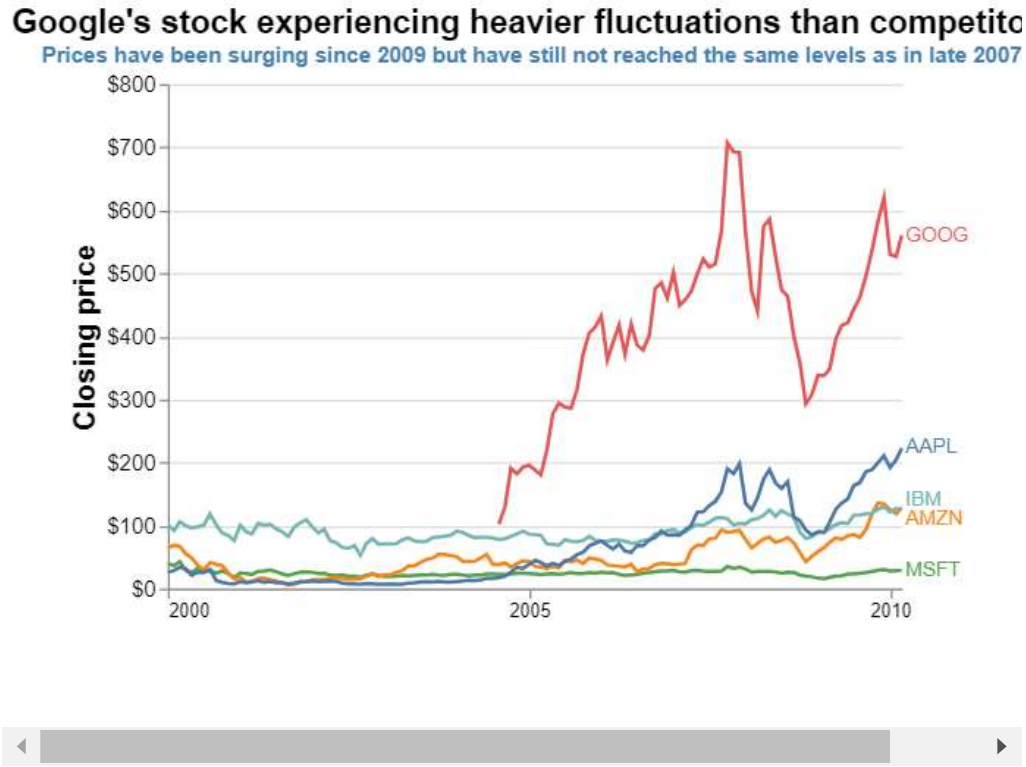
Comments

- The `configure_*` chart methods are useful when we want to make a modification to an already existing chart without copying and pasting all the code
- In general, it is preferred to change these parameters directly in the main chart code instead you can read this section of the docs to find out more.

We will cover an example with `configure_title` later in the slide deck.

Font Size and Colour

- Show the code



Font sizes and colours can be adjusted for different communication purposes (err on the side of larger)

To learn more about good guidelines for titles and labels, you can read section 22 - 22.2 in

Defining and transforming axis ranges

Wikipedia Donations

This data set contains the daily sum of all donations received by Wikipedia in the year 2020.

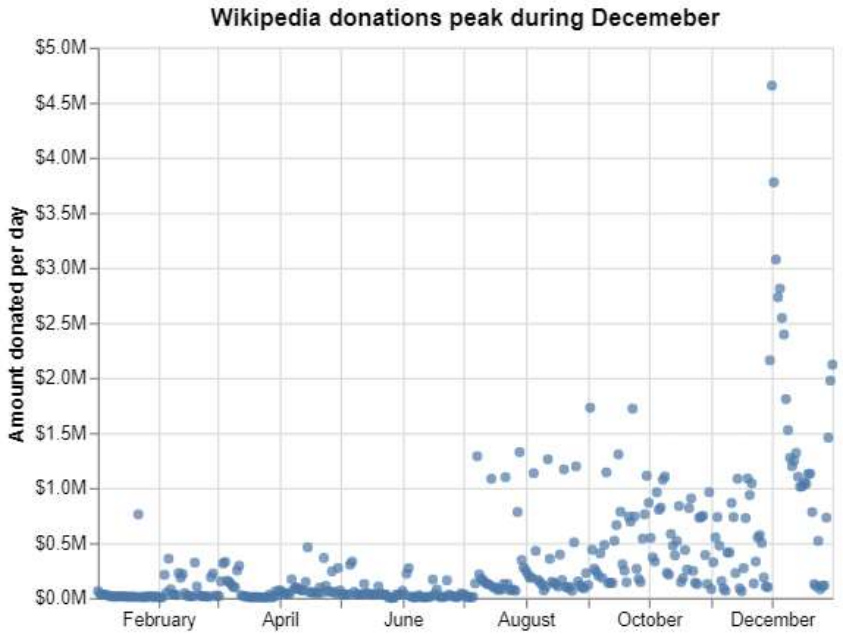
```
import pandas as pd

donations = pd.read_csv('data/donations.csv', parse_dates=['date'])
```

```
donations.head()
```

	date	sum	refund_sum	donations	refunds	avg	max	ytdsum	ytdloss	week_day
0	2020-01-02	62419.79	0.0	3342	0	18.677376	1040.00	879029.00	0.0	Thu
1	2020-01-03	37983.67	0.0	1949	0	19.488799	1000.00	917012.67	0.0	Fri
2	2020-01-04	26219.10	0.0	1337	0	19.610396	208.00	943231.77	0.0	Sat
3	2020-01-05	33856.07	0.0	1289	0	26.265376	5596.86	977087.84	0.0	Sun
4	2020-01-06	26447.11	0.0	1347	0	19.634083	699.67	1003534.95	0.0	Mon

Scatterplot

▶ Show the code



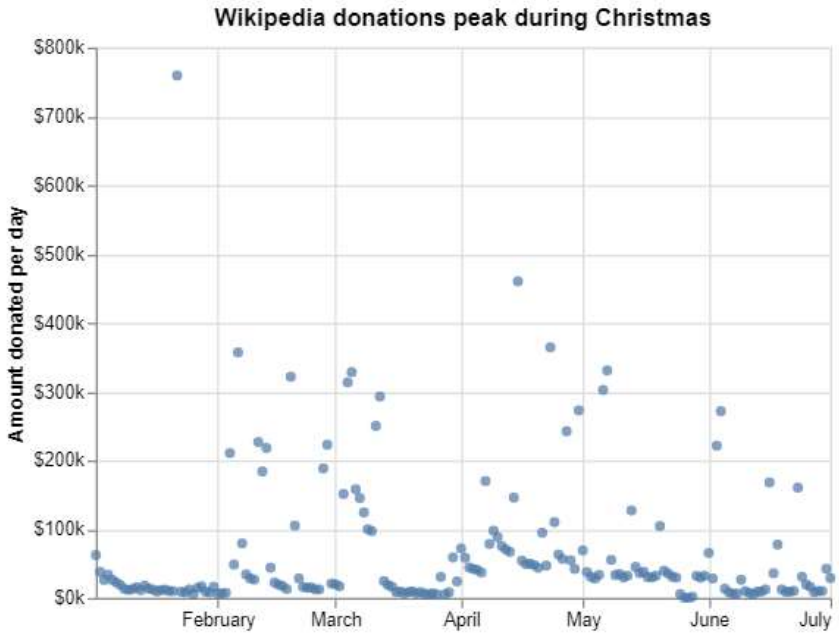
The most money is donated around Christmas

There also seems to be a higher amount donated during the fall then during spring and winter.

This plot has a lot of wasted space and is oversaturated

Filter

- Show the code



Filtering is often the easiest way to zoom in on an axis

To zoom in further we could use another filter *or* we could limit the y-axis domain directly in Altair …

we could also limit the x-axis direct in altair; see ex

Adjusting Axis Limits

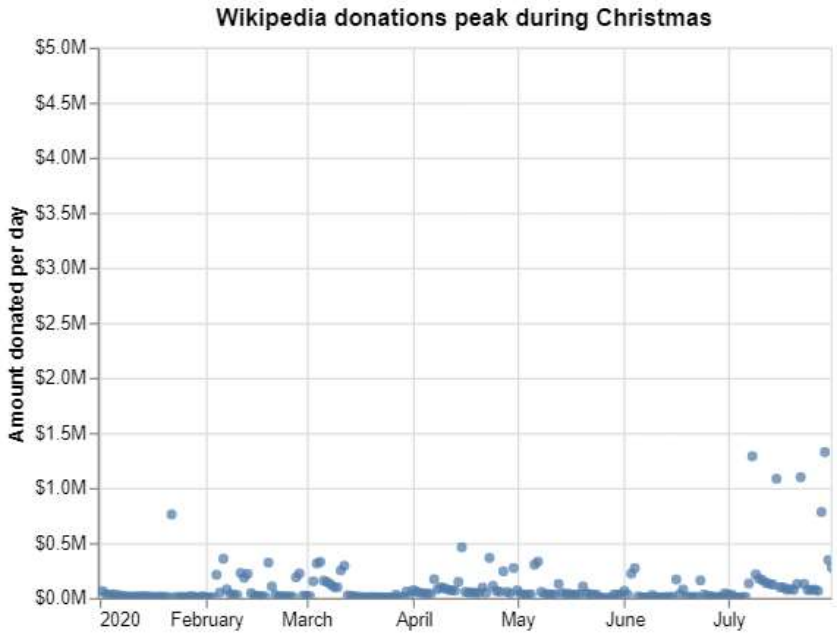
1. set `clip=True` inside the mark.
    - Without this, we would still see the circles extending beyond the range of the chart.
2. set `scale = alt.Scale(domain=[min, max])` in the appropriate encoding channel
    - For the x-axis we will look at the first 7 months of 2020
    - For the y-axis we'll go from 0 to 10K

N.B. `alt.Scale(zero=False)` removes the origin from plotting

33

X-axis scale

▶ Show the code



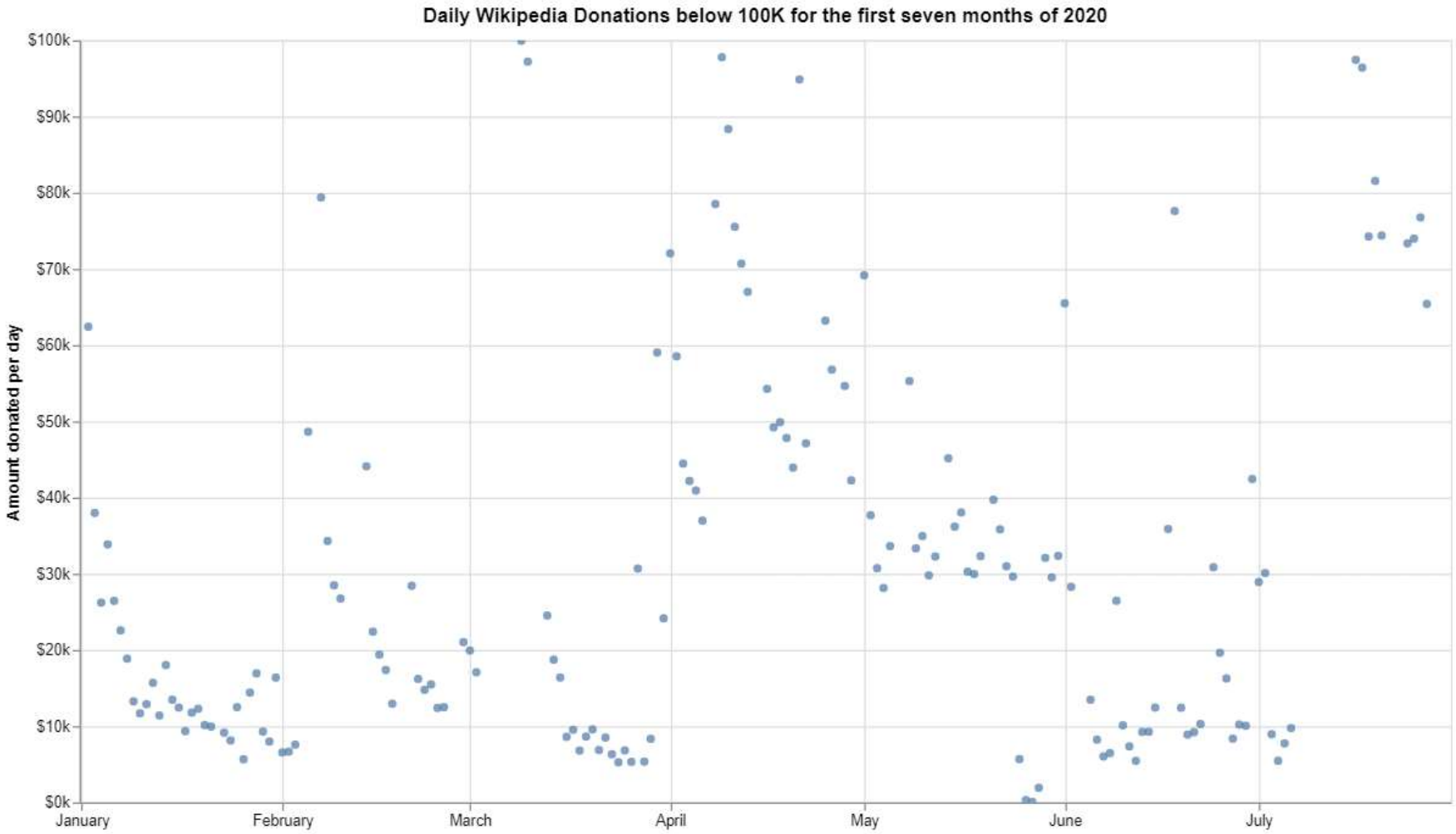
Limiting the x-axis domain with alt.scale is different than with filtering.

The y-axis still extends to $5M since the y-axis range is still based on all the data.

Altair scale for both axes

```
eda_title = "Daily Wikipedia Donations below 100K for the first seven month
alt.Chart(donations, title=eda_title
).mark_circle(
  clip=True
).encode(
    alt.X('date',
      axis=alt.Axis(format="%B", tickCount=7),
      title=None,
      scale=alt.Scale(domain=['2020-01-01', '2020-07-31'])),
    alt.Y(
      'sum',
      axis=alt.Axis(format='$s'),
      title='Amount donated per day',
      scale=alt.Scale(domain=[0, 100000]))
).properties(height=500, width=900)
```

Altair scale for both axes

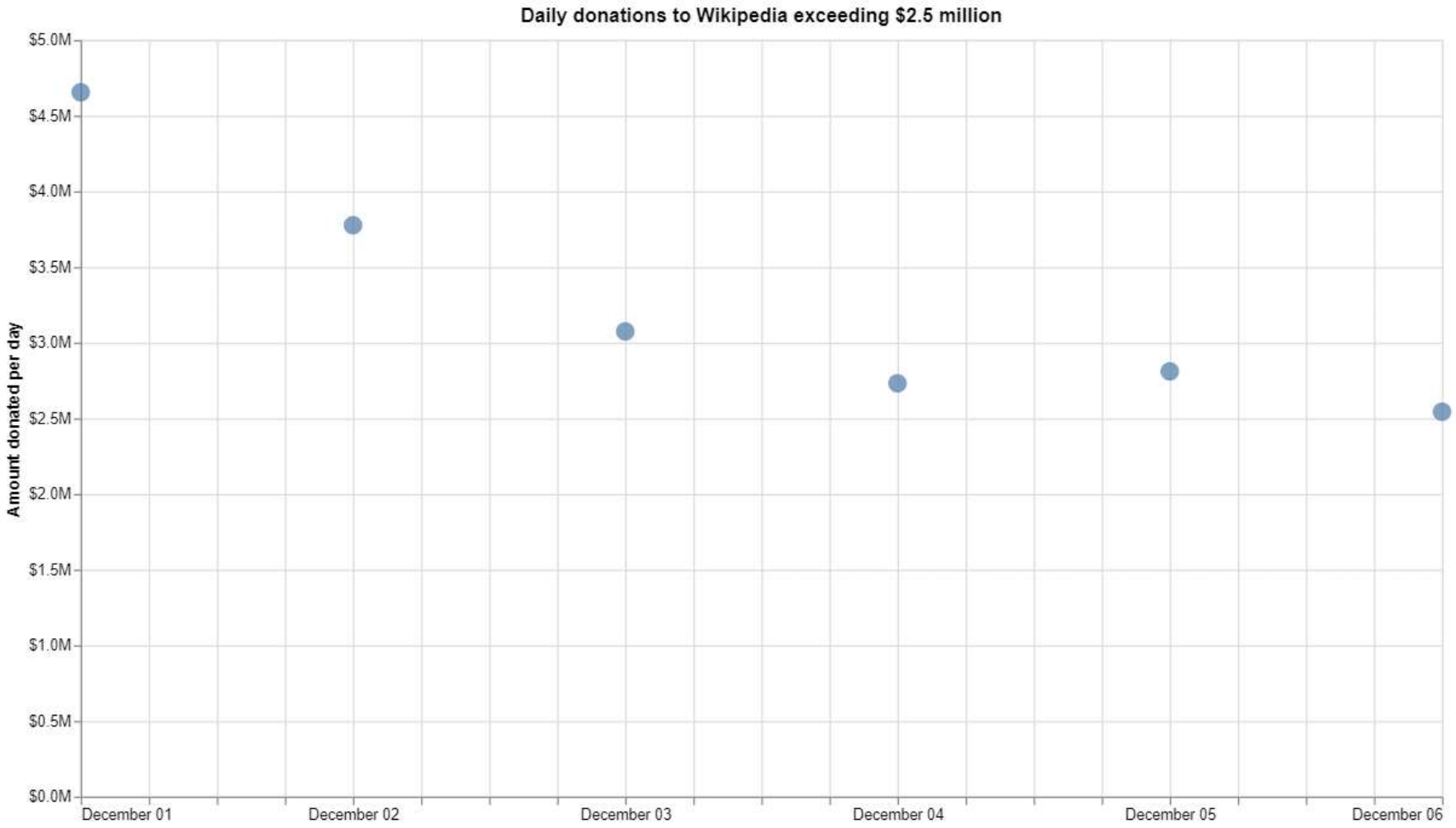


Large donations

- Suppose now that we wanted to investigate when the large donation day, say greater than $2.5M, occurred.
- As we have seen before, we can do that filtering directly within Altair

```
alt.Chart(donations[donations['sum'] > 2.5e6]
).mark_circle(size=150).encode(
    alt.X('date', title=None, axis=alt.Axis(format="%B %d")),
    alt.Y('sum',
      axis=alt.Axis(format='$s'),
      title='Amount donated per day')
).properties(
  title = 'Daily donations to Wikipedia exceeding $2.5 million',
  height=500, width=900
)
```

Large donations

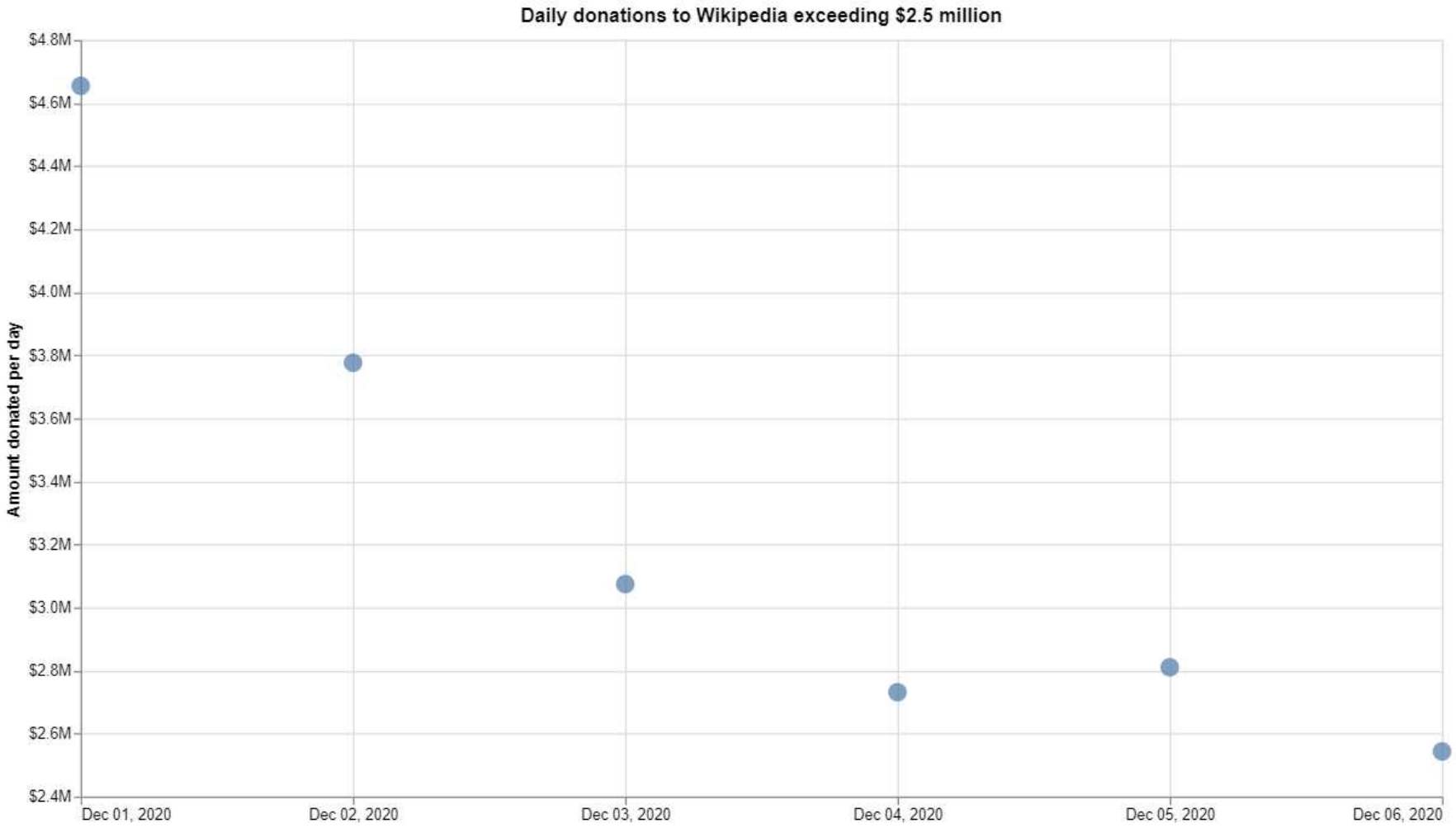


Excluding the origin

- Altair will often plot the origin.
- To exclude zero in scale domain use:

```
alt.Chart(donations[donations['sum'] > 2.5e6]).mark_circle(size=150).encode
    alt.X('date', title=None,
      axis=alt.Axis(format="%b %d, %Y", tickCount=6)), # or grid=False
    alt.Y('sum',
      axis=alt.Axis(format='$s'),
      title='Amount donated per day',
      scale = alt.Scale(zero=False))
).properties(
  title = 'Daily donations to Wikipedia exceeding $2.5 million',
  height=500, width=900
)
```

Excluding the origin



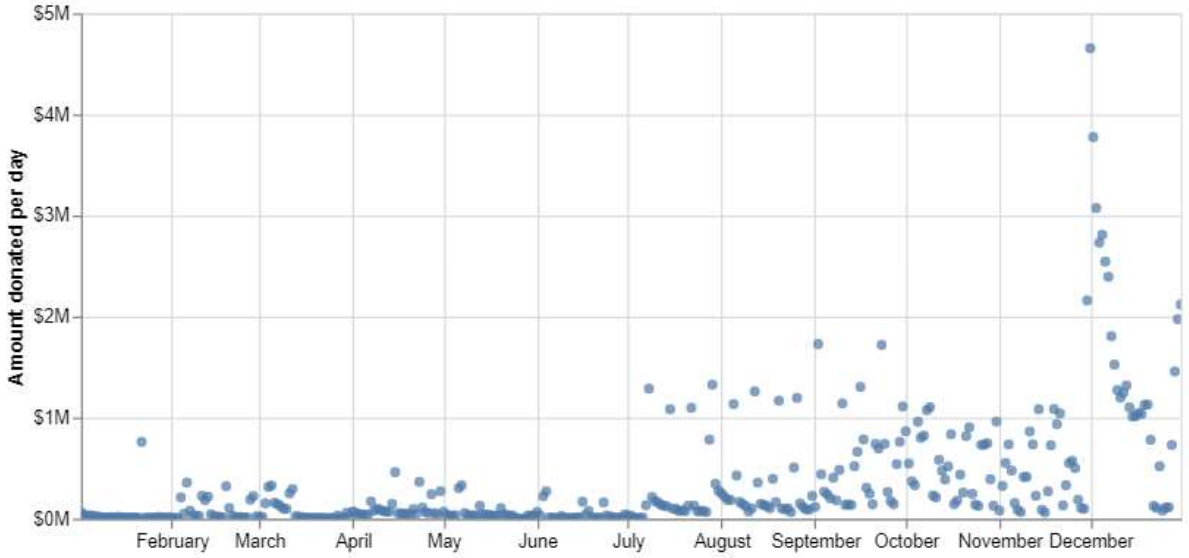
Comments

- While zooming can reveal detail of a specific region of the chart, it hides the variation among the rest of the data
- As discussed multiple times before, this is not generally a good idea
- Is there a way to see details for the days with lower total donations but still include all the observations in a single chart?

Interactive zoom

Try dragging and scrolling on the graphic below

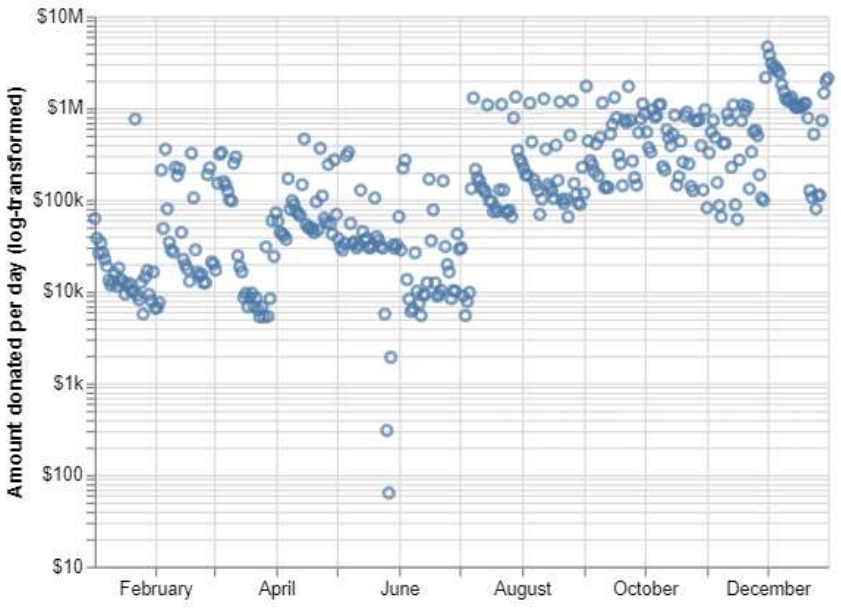
```
alt.Chart(donations).mark_circle().encode(
    alt.X('date', title=None),
    alt.Y('sum', axis=alt.Axis(format='$s'),
    title='Amount donated per day')
).properties(height=275, width = 600).interactive()
```



log-transformations

log[1] transformations increases the range used for the smaller values[2] and compresses the space used for larger values

```
log_title = 'Amount donated per da
alt.Chart(donations).mark_point(
).encode(
  alt.X('date', title=None),
  alt.Y(
    'sum',
    title=log_title,
    axis=alt.Axis(format='$s'),
    scale=alt.Scale(type='log')),
  alt.Tooltip('week_day')
)
```



1. Altair uses the logarithic base 10 (which is the most common choice).
2. If you data contain zeros use `symlog` instead