Data 550: Data Visualization I

Lecture 2: Effective Use of Visual Channels

Dr. Irene Vrbik
University of British Columbia Okanagan

https://github.com/ubco-mds-2023/Data-550

Introduction

- So far we have seen how to use points and lines to represent data visually.
- Today, we will see how we can also use areas and bars
- Before coding, let's discuss when it is appropriate to use a certain geometric mark to graphically represent data.
- Through a series of examples, we will get a feel for visual encodings are the most effective in different scenarios.

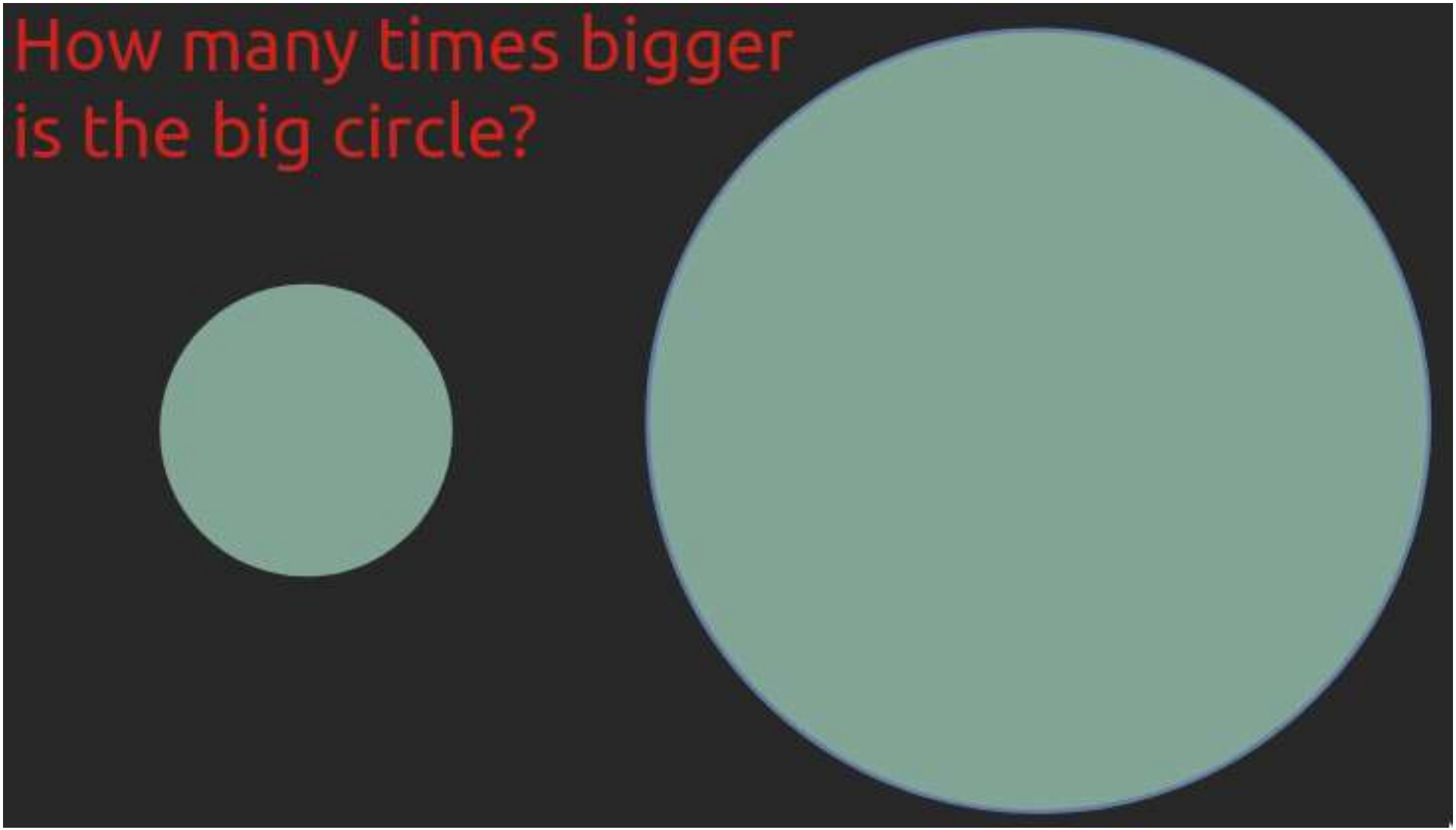
https://github.com/ubco-mds-2023/Data-550

Lecture outcomes:

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

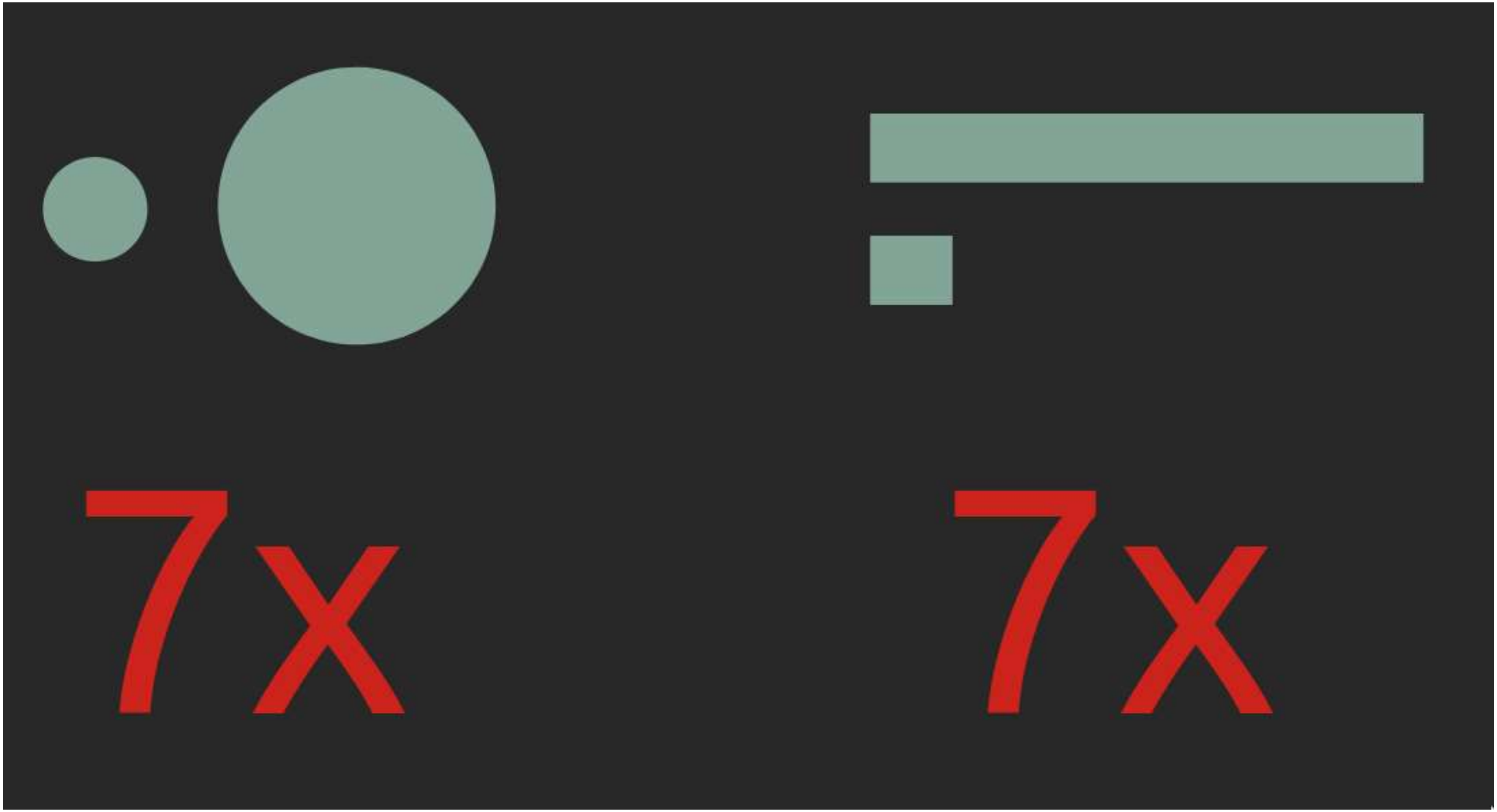
1. Create and interpret Line chart and Area Charts
2. Create and interpret Bar charts and Histograms
3. Choose effective visual encodings
4. Customize axes labels and main titles

https://github.com/ubco-mds-2023/Data-550



https://github.com/ubco-mds-2023/Data-550

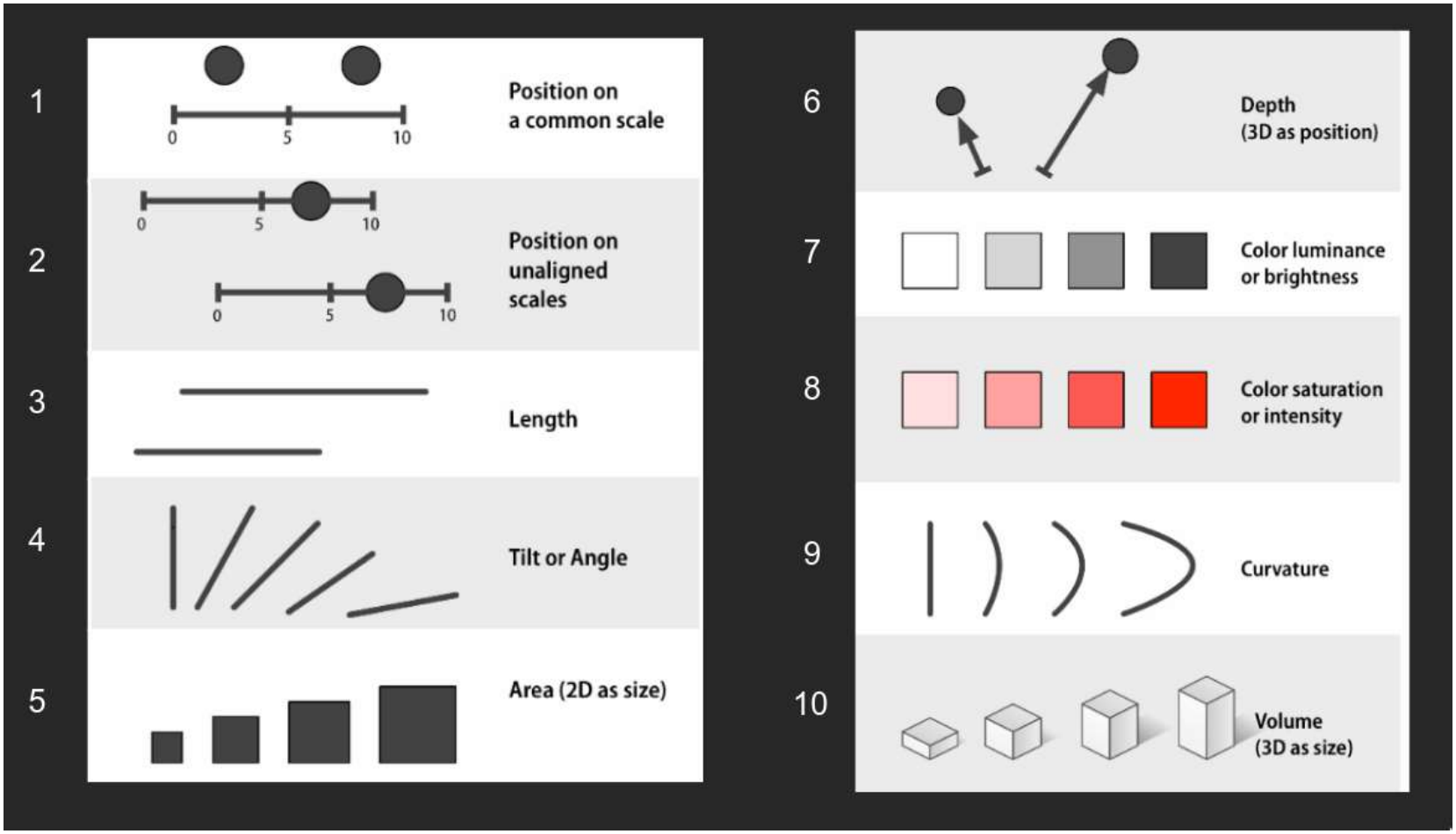
https://github.com/ubco-mds-2023/Data-550



Examples taken from Jeffrey Heer's PyData talk

https://github.com/ubco-mds-2023/Data-550

Rank of Visual Channels by Efficiency

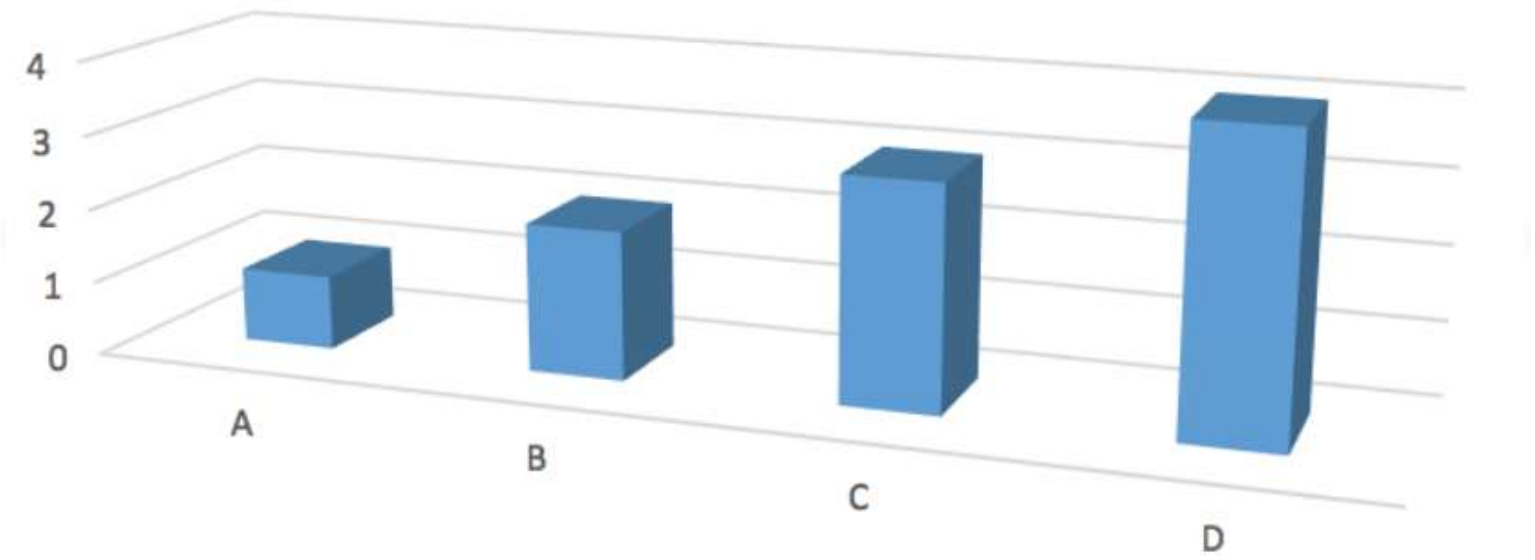


Channels for mapping ordered data (continuous or other quantitative measures), arranged top-to-bottom from more to less effective Munzer (2014, 102).

https://github.com/ubco-mds-2023/Data-550

Don’t use 3D unneccessarily

As we saw in lecture 1, this makes the plot harder to interpret

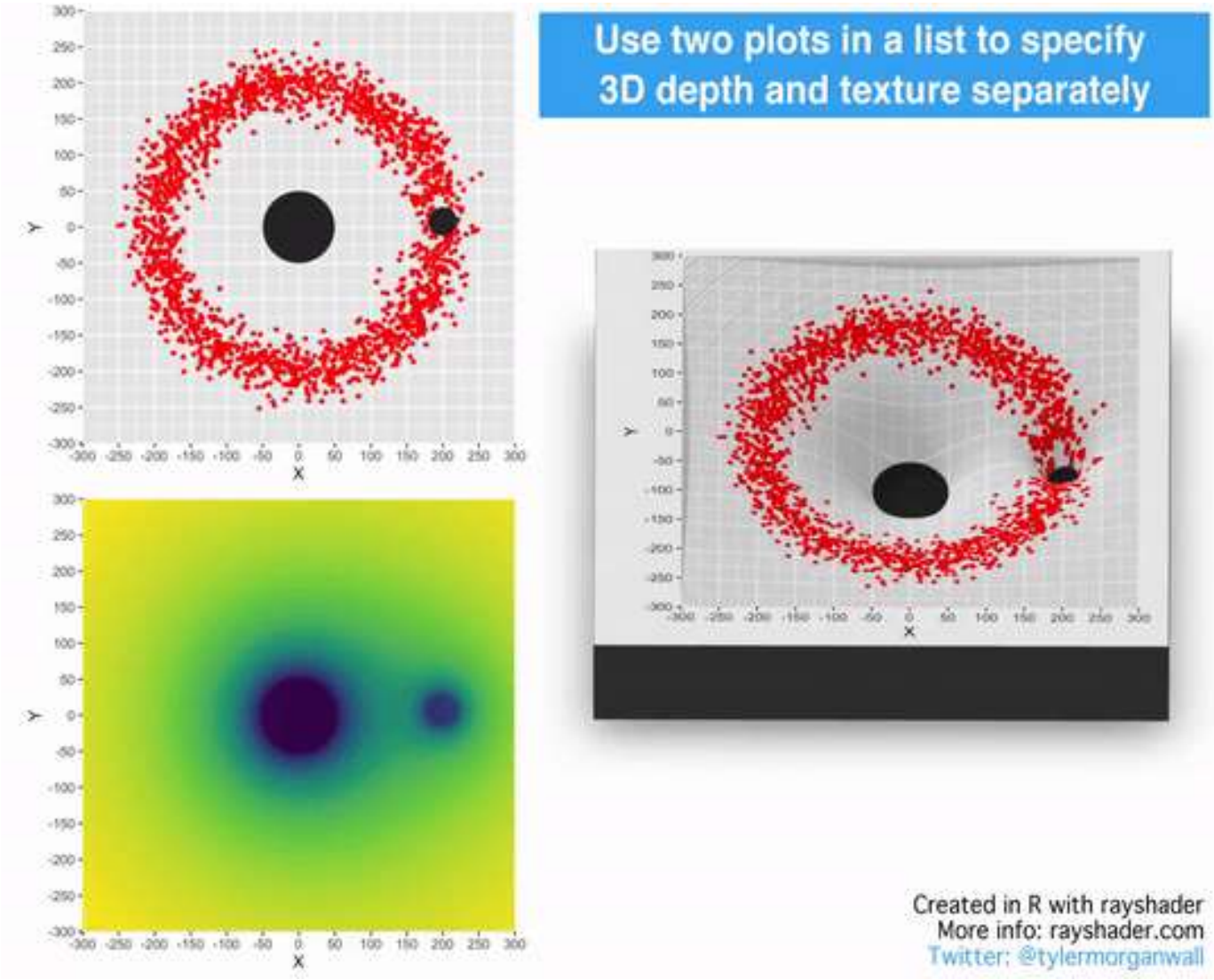


https://github.com/ubco-mds-2023/Data-550

Meaningful 3D

3D plots can facilitate interpretation when used appropriately

https://github.com/ubco-mds-2023/Data-550



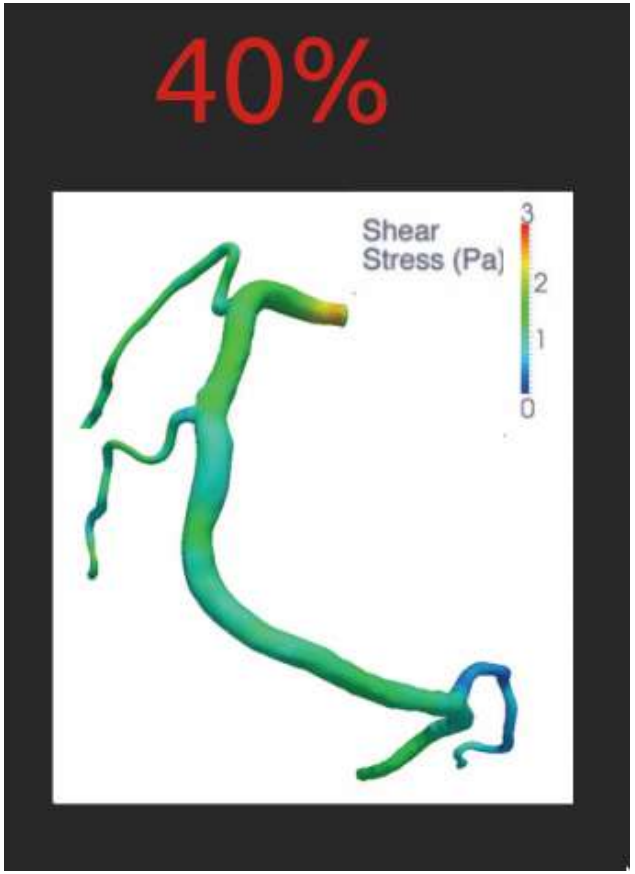
The Researcher Claus Wilke's has authored a good chapter on this topic if you are interested to learn more. Visit the Rayshader site for more great 3D plots

https://github.com/ubco-mds-2023/Data-550

Case study: Heart Disease

- Heart disease is the most common cause of death, yearly killing almost 9 million people
- By detecting regions of low shear stress in the arteries around the heart, doctors can identify patients that are on their way to develop heart disease and take action early to improve the patient's survival chances.
- To evaluate the shear stress in the arteries, the regular practice is to use a digital 3D representation of the artery coloured according to the amount of shear stress.

https://github.com/ubco-mds-2023/Data-550



- Here the colormap changes from blue for the areas of interest (low stress) to cyan, green, yellow, and red for higher stress.
- When using the visualization you see in this slide, about 40% of the areas of low shear stress were correctly identified by doctors.

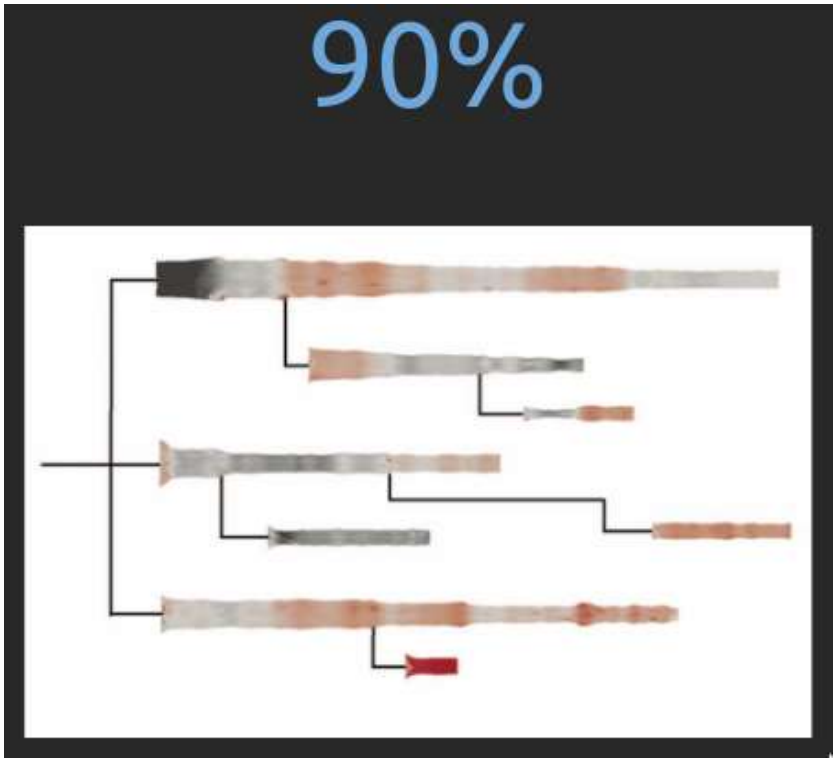
https://github.com/ubco-mds-2023/Data-550

Different Colourscale

- This alternative colourmap highlights low shear stress in bright red (the rest are in black and white.)
- When using the visualization you see in this slide, about 70% of the areas of low shear stress were correctly identified by doctors.

https://github.com/ubco-mds-2023/Data-550

From 3D to 2D



- When a 2d representation of the blood vessel was considered about 90% of the areas of low shear stress were correctly identified by doctors.

https://github.com/ubco-mds-2023/Data-550

Area Charts

- As an alternative to alternative to the line charts we saw last lecture, it may be useful to create *area charts*.
- An area chart or area graph displays how one or more groups' numeric values change over the progression of a second variable
- It is based on the line chart.
- The area between axis and line are commonly emphasized with colors, and textures.

https://github.com/ubco-mds-2023/Data-550

Data Introduction

- Here we learn:
  - how to create area charts
  - rules of thumb for when they may be preferred over line charts
- We will be visualizing global health and population data for a number of countries.

https://github.com/ubco-mds-2023/Data-550

Global Development Data

Column	Description
country	Country name
year	Year of observation
population	Population in the country at each year
region	Continent the country belongs to
sub_region	Sub-region the country belongs to
income_group	Income group as specified by the world bank in 2018
life_expectancy	The mean number of years a newborn would live if mortality patterns remained constant
income	GDP per capita (in USD) adjusted for differences in purchasing power
children_per_woman	Average number of children born per woman
child_mortality	Deaths of children under 5 years of age per 1000 live births
pop_density	Average number of people per km2
co2_per_capita	CO2 emissions from fossil fuels (tonnes per capita)
years_in_school_men	Mean number of years in primary, secondary, and tertiary school for 25-36 years old men
years_in_school_women	Mean number of years in primary, secondary, and tertiary school for 25-36 years old women

jump ahead

https://github.com/ubco-mds-2023/Data-550

Importing the data

```
1 import altair as alt
2 import pandas as pd
3
4 gm_url = 'https://raw.githubusercontent.com/vrbiki/data/main/world-data-gap
5 gm = pd.read_csv(gm_url)
6 gm
```

	country	year	population	region	sub_regi
0	Afghanistan	1800-01-01	3280000	Asia	Southern Asia
1	Afghanistan	1801-01-01	3280000	Asia	Southern Asia
2	Afghanistan	1802-01-01	3280000	Asia	Southern Asia

https://github.com/ubco-mds-2023/Data-550

	country	year	population	region	sub_regi
3	Afghanistan	1803-01-01	3280000	Asia	Southern Asia
4	Afghanistan	1804-01-01	3280000	Asia	Southern Asia
...	...	...	...	...	...
38977	Zimbabwe	2014-01-01	15400000	Africa	Sub-Saharan Africa
38978	Zimbabwe	2015-01-01	15800000	Africa	Sub-Saharan Africa

https://github.com/ubco-mds-2023/Data-550

	country	year	population	region	sub_regi
38979	Zimbabwe	2016-01-01	16200000	Africa	Sub-Saharan Africa
38980	Zimbabwe	2017-01-01	16500000	Africa	Sub-Saharan Africa
38981	Zimbabwe	2018-01-01	16900000	Africa	Sub-Saharan Africa

38982 rows × 14 columns

https://github.com/ubco-mds-2023/Data-550

Data Information

```
1 gm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38982 entries, 0 to 38981
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   country               38982 non-null  object
 1   year                  38982 non-null  object
 2   population            38982 non-null  int64
 3   region                38982 non-null  object
 4   sub_region            38982 non-null  object
 5   income_group          38982 non-null  object
 6   life_expectancy       38982 non-null  float64
 7   income                38982 non-null  int64
 8   children_per_woman    38982 non-null  float64
 9   child_mortality       38980 non-null  float64
```

https://github.com/ubco-mds-2023/Data-550

Parsing Dates

```
1 gm = pd.read_csv(gm_url, parse_dates=['year'])
2 gm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38982 entries, 0 to 38981
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   country               38982 non-null  object
 1   year                  38982 non-null  datetime64[ns]
 2   population            38982 non-null  int64
 3   region                38982 non-null  object
 4   sub_region            38982 non-null  object
 5   income_group          38982 non-null  object
 6   life_expectancy       38982 non-null  float64
 7   income                38982 non-null  int64
 8   children_per_woman    38982 non-null  float64
 9   child_mortality       38980 non-null  float64
```

https://github.com/ubco-mds-2023/Data-550

19

Working with large data sets

One strategies for working with large datasets point to the URL where the data lives which the chart creation.

```
alt.Chart('https://website.com/data.csv').mark_line().encode(
    x='column1:T',
    y='column2:Q')
```

The drawback of not using a dataframe, is that Altair cannot rely on pandas to infer what type of data there is in each column, so we need to help it by indicating the datatype with `:*` where `*` indicates the data type …

https://github.com/ubco-mds-2023/Data-550

Data Types

Data Type	Code	Description
Ordinal	**O**	a discrete ordered quantity, eg. “dislike”, “neutral”, “like”
Nominal	**N**	a discrete unordered quantity, eg. eye color, postal code, flower-type
Quantitative	**Q**	a continuous quantity, eg. 5, 5.0, 5.011

Temporal : T

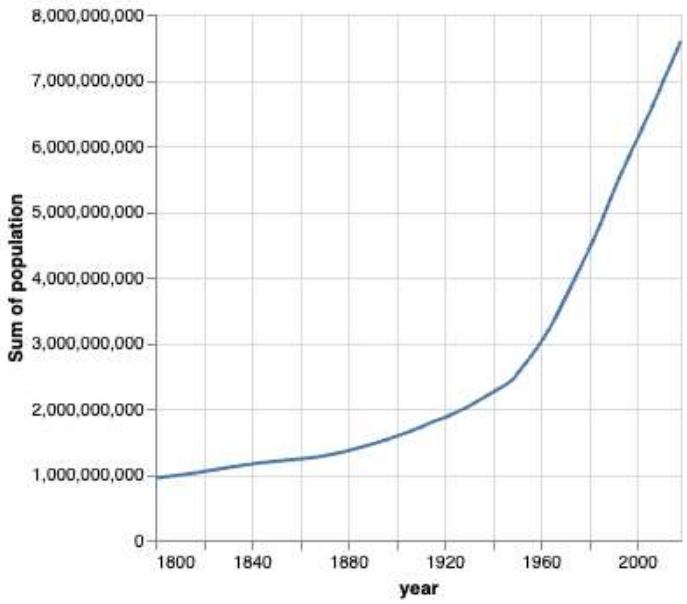
https://github.com/ubco-mds-2023/Data-550

Line chart

Task: Plot the total population over time as a line chart.

```
alt.Chart(_____).mark_<_____>().encode(
    x='year__',
    y='________')
```

▶ Code



https://github.com/ubco-mds-2023/Data-550

Customize axes labels

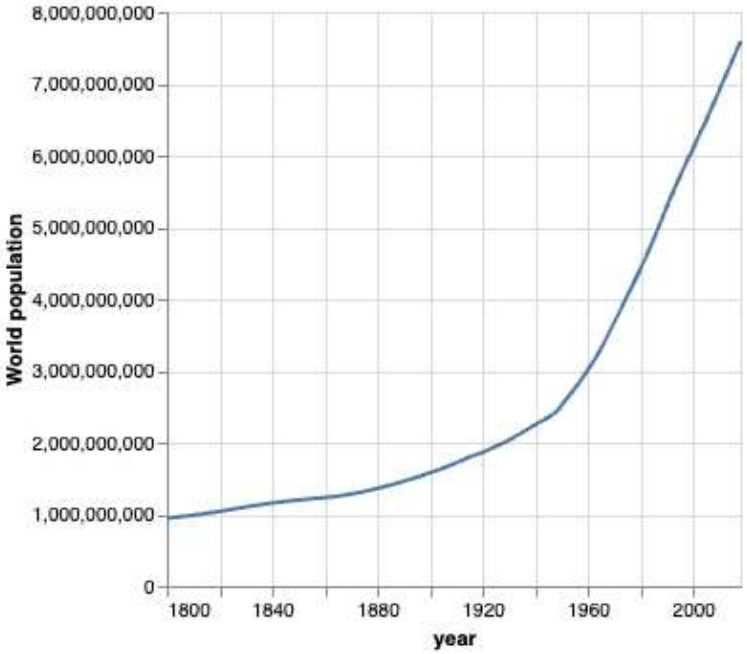
- The Y-axis label is changed to reflect the operation that we have performed; here taking the “sum” of the countries.
- To change the default label, we will need to move from the encoding shorthands to the long-form channel encoding

Shorthand	Equivalent long-form
`x='name'`	`alt.X('name')`
`x='name:Q'`	`alt.X('name', type='quantitative')`
`x='sum(name)'`	`alt.X('name', aggregate='sum')`
`x='sum(name):Q'`	`alt.X('name', aggregate='sum', type='quantitative')`
`x='count():Q'`	`alt.X(aggregate='count', type='quantitative')`

https://github.com/ubco-mds-2023/Data-550

Line Chart with changed Axis label

```
alt.Chart(gm_url).mark_line().encode(x='year:T',
    y=alt.Y('sum(population):Q', title='World population'))
```

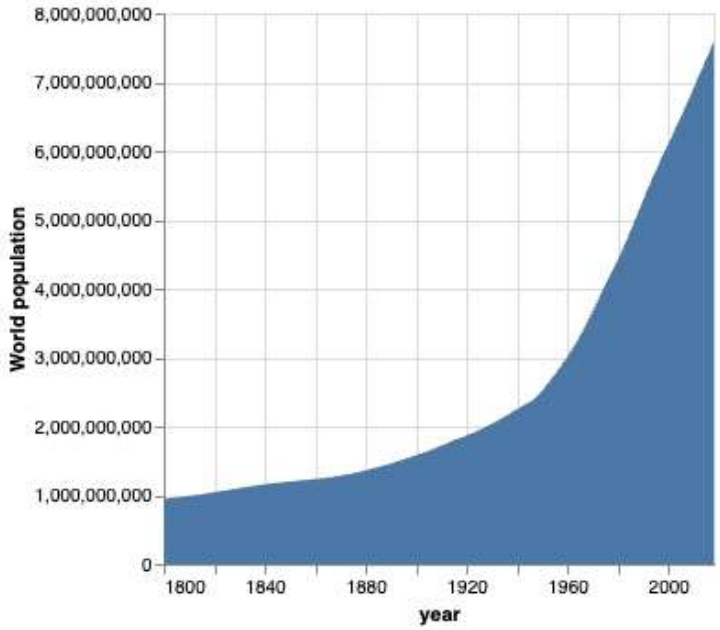


*Question: Can you guess how this code would changed to make an **area** chart?*

https://github.com/ubco-mds-2023/Data-550

Area Chart with changed Axis label

```
alt.Chart(gm_url).mark_area().encode(
    x='year:T',
    y=alt.Y('sum(population):Q', title='World population'))
```

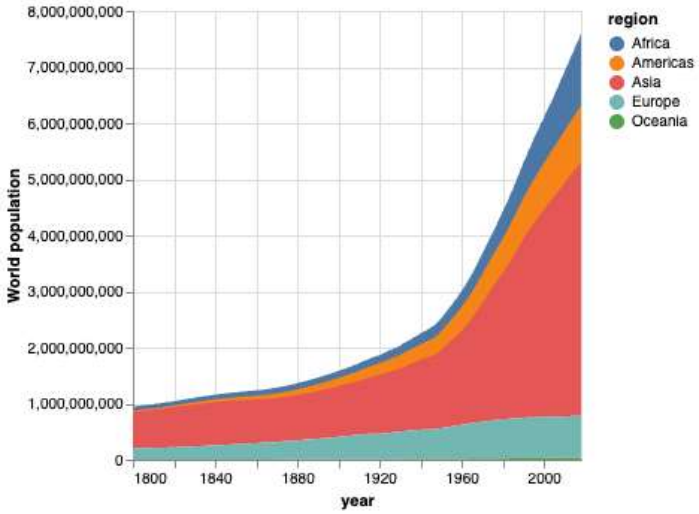


A video on how the world population changed

https://github.com/ubco-mds-2023/Data-550

Stacked area chart

```
alt.Chart(gm_url).mark_area().encode(alt.X('year:T'),
    alt.Y('sum(population):Q', title='World population'),
    color='region:N')
```

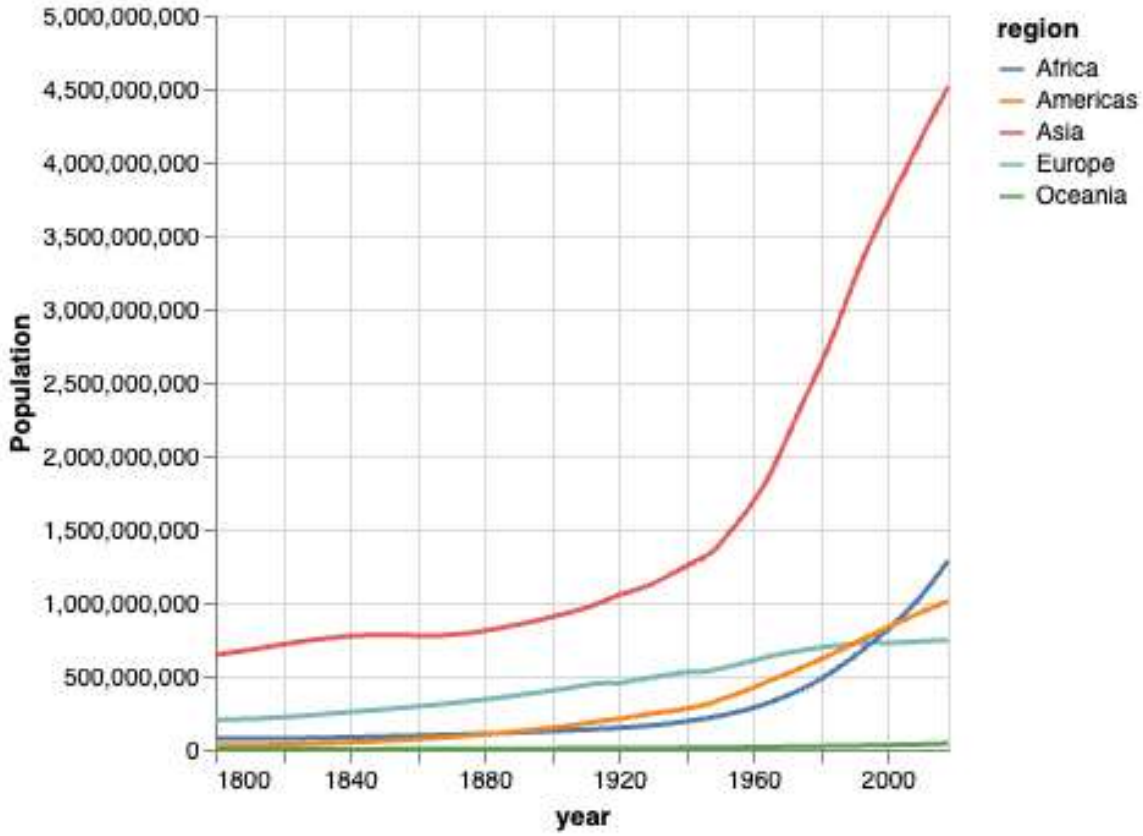


It is helpful to clarify that the areas are stacked on top of each other rather than layered behind each other.

https://github.com/ubco-mds-2023/Data-550

Line Chart by Region

```
alt.Chart(gm_url).mark_line().encode(alt.X('year:T'),
    alt.Y('sum(population):Q', title='Population'),
    color='region:N')
```



https://github.com/ubco-mds-2023/Data-550

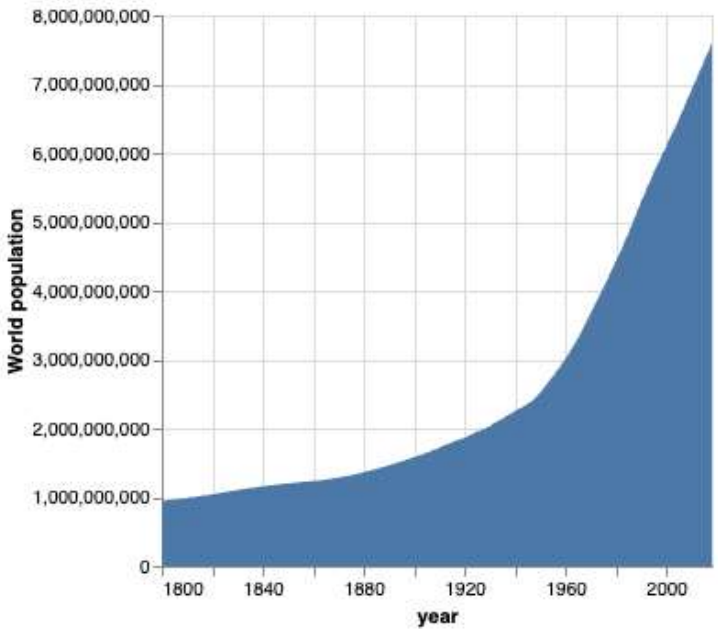
Summary

- Both line and area charts are effective choices for showing a single trend over time.
- However, when visualizing the trends over time for multiple groups, lines and areas have different advantages
- As a general guideline, area charts are preferred when the total of the groups is more important than comparing the differences between groups
- Line charts are preferred when the individual groups are the more important than the total of the groups

Stopped here Jan, 10th (lec 2)

https://github.com/ubco-mds-2023/Data-550

Will this trend continue



Fortunately, the world population is predicted to stabilize around 11 billion people at the end of the century.

- *Why?*

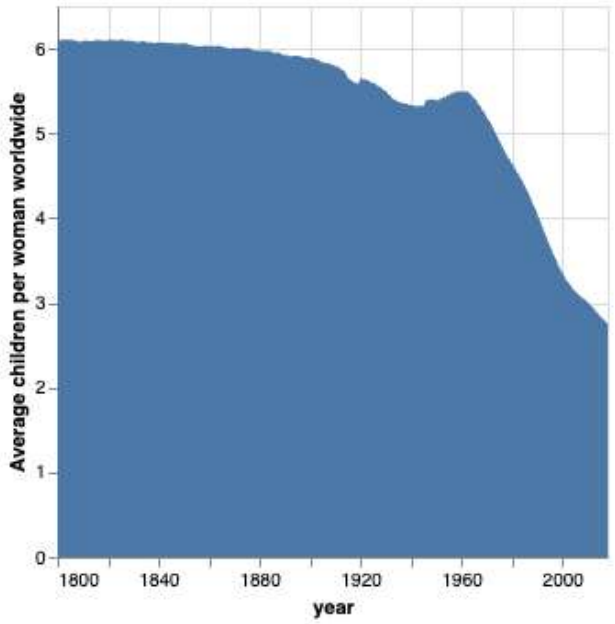
https://github.com/ubco-mds-2023/Data-550

Any features in the Global Development Data that might helps us answer this?

Children per woman

Because across the world, women are having fewer and fewer babies (explanations for this decrease at ourworldindata)

```
alt.Chart(gm_url).mark_area().encode(
    alt.X('year:T'), alt.Y('mean(children_per_woman):Q',
    title='Average children per woman worldwide'))
```

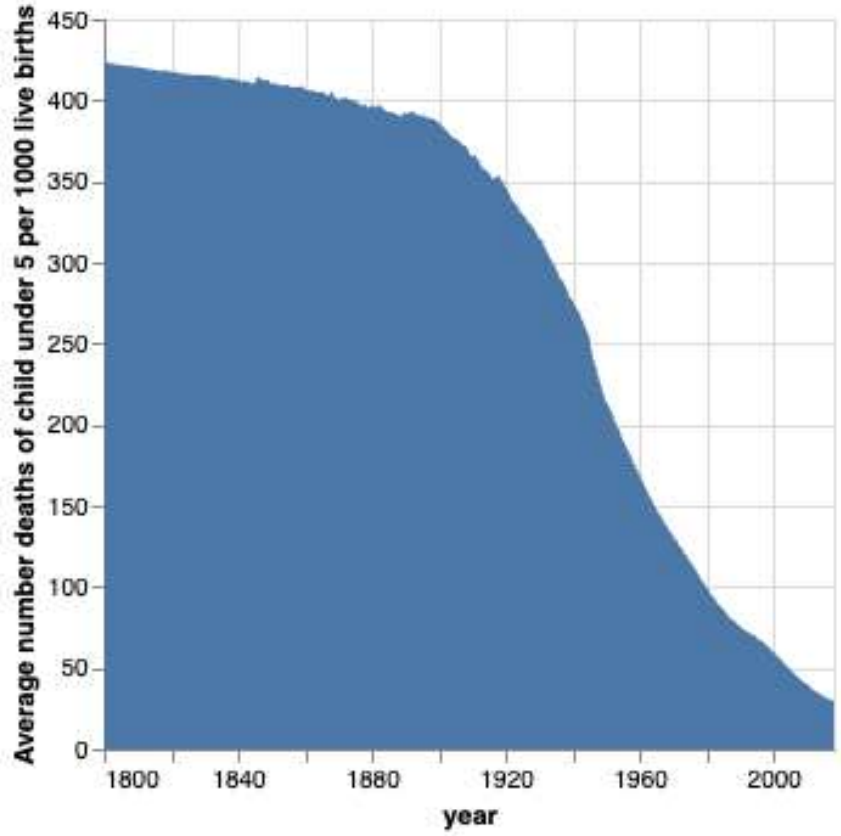


https://github.com/ubco-mds-2023/Data-550

See this predicted stabilization described using blocks and age distribution here

Child Mortality

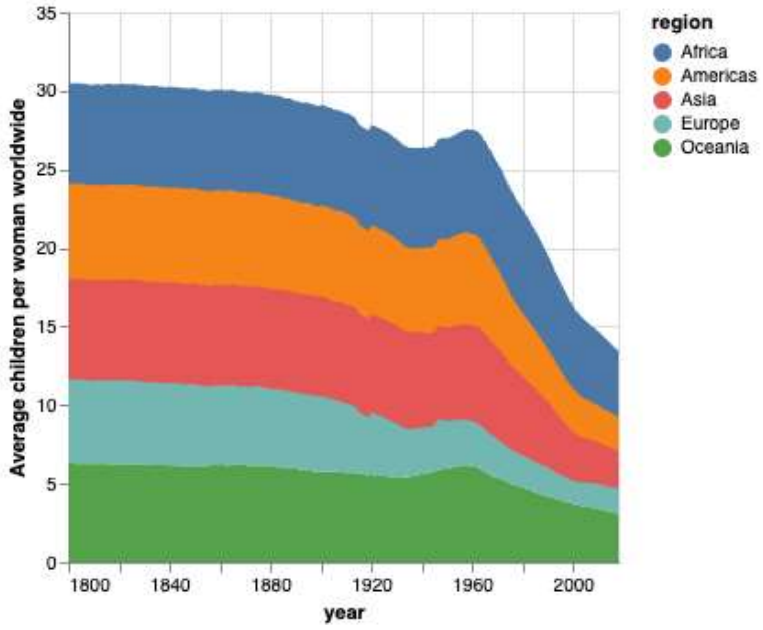
```
alt.Chart(gm_url).mark_area().encode(
    alt.X('year:T'), alt.Y('mean(child_mortality):Q',
    title='Average number deaths of child under 5 per 1000 live births'))
```

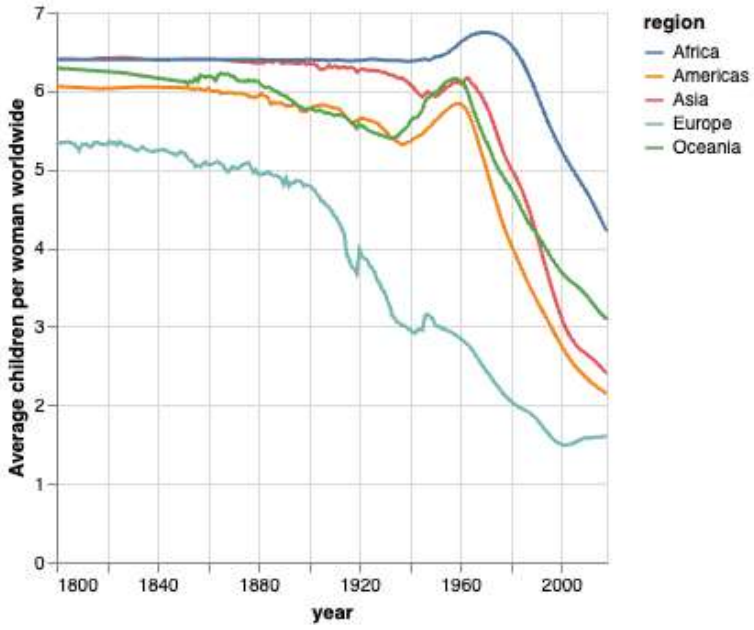


https://github.com/ubco-mds-2023/Data-550

Question

Which of the graphs does a better job in answering the question *Which regions average fertility rate decreased the most over the years*?





https://github.com/ubco-mds-2023/Data-550

Bar charts

- A bar chart is a graph that presents *categorical* (i.e nominal) data with rectangular bars with heights (or lengths) proportional to the values that they represent.
- The heights commonly correspond to the frequency (i.e. *count*) of observations falling in that category.
- The bars can be plotted vertically or horizontally.

How many people lived in each region in 2018?

https://github.com/ubco-mds-2023/Data-550

Subsetting the data

```
1 gm2018 = gm[gm['year'] == '2018'] # filter on year 2018
2 gm2018.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 178 entries, 218 to 38981
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   country               178 non-null    object
 1   year                  178 non-null    datetime64[ns]
 2   population            178 non-null    int64
 3   region                178 non-null    object
 4   sub_region            178 non-null    object
 5   income_group          178 non-null    object
 6   life_expectancy       178 non-null    float64
 7   income                178 non-null    int64
 8   children_per_woman    178 non-null    float64
 9   child_mortality       177 non-null    float64
```

https://github.com/ubco-mds-2023/Data-550

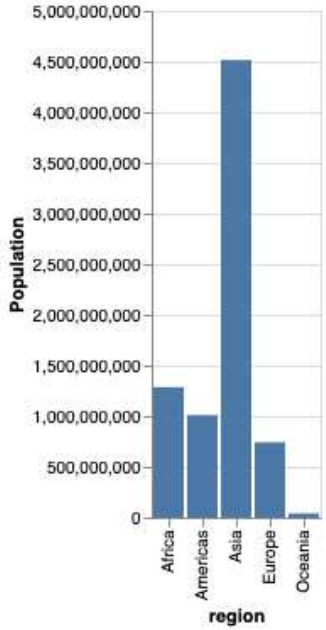
Total population per region

Can you guess what marking is needed for a bar plot?

```
alt.Chart(___).mark_???().encode(
    alt.X(_____),
    alt.Y(______,
    title=________))
```

```
alt.Chart(gm2018).mark_bar().encode(
    alt.X('region'),
    alt.Y('sum(population)',
    title='Population'))

# same as:
# alt.Chart(gm2018).mark_bar().encode(
#     x = alt.X('region'),
#     y = alt.Y('sum(population)',
#       title='Population'))
```



https://github.com/ubco-mds-2023/Data-550

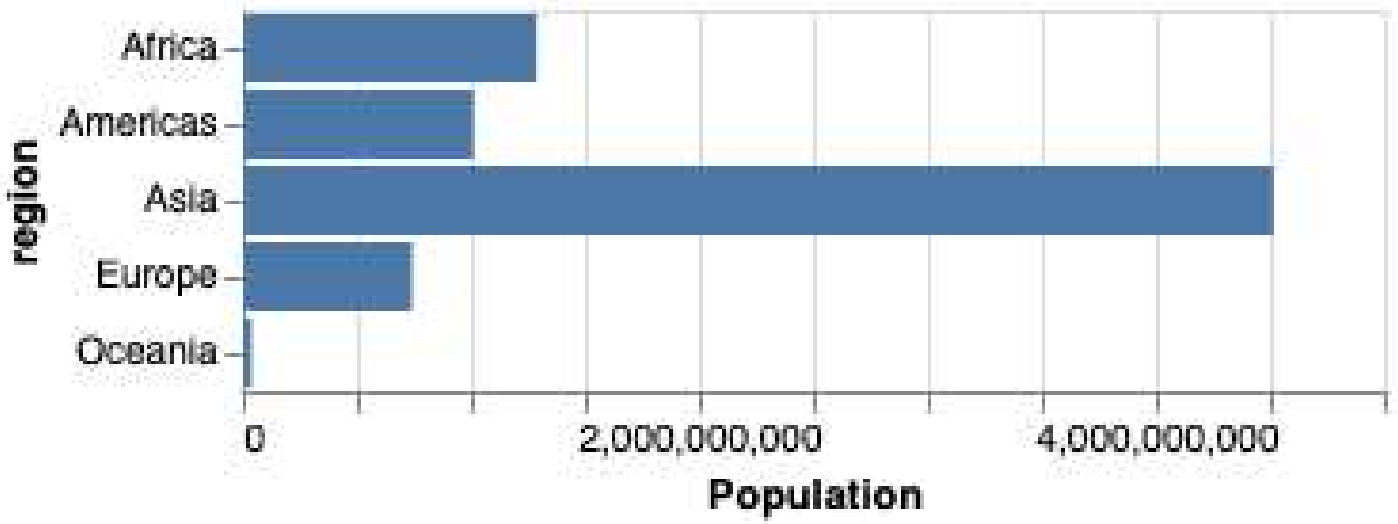
35

Horizontal bar charts

Horizontal bar charts are preferred when the labels on the categorical axis become so long that they need to be rotated to be readable in a vertical bar chart. . . .

```
1 alt.Chart(gm2018).mark_bar().encode(
2     alt.X('sum(population)', title='Population'),
3     alt.Y('region'))
```

flipping x and y changes bar graph from vertical to horizontal



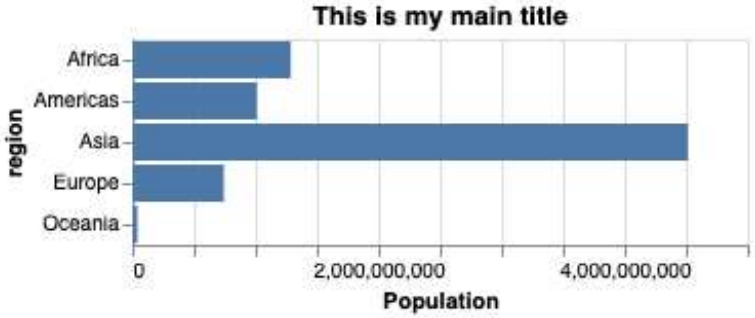
https://github.com/ubco-mds-2023/Data-550

Adding Main Titles

Horizontal bar charts are preferred when the labels on the categorical axis become so long that they need to be rotated to be readable in a vertical bar chart. . . .

We can add a main title by applying the `properties()` method to our chart …

```
1  alt.Chart(gm2018).mark_bar().encode(
2     alt.X('sum(population)', title='Population'),
3     alt.Y('region')).properties(title = "This is my main title")
```

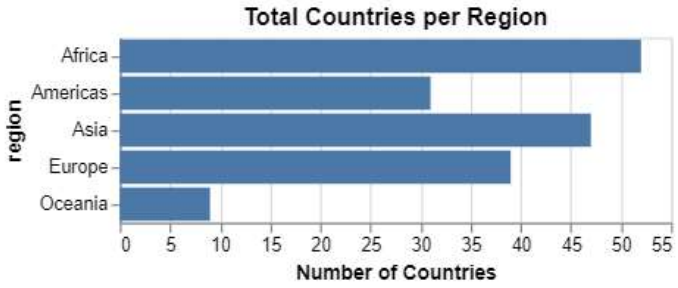


https://github.com/ubco-mds-2023/Data-550

Bar charts for counts

How many countries there were within each region?

```
alt.Chart(gm2018).mark_bar().encode(
    alt.X('count()', title='Number of Countries'),
    alt.Y('region')).properties(title = 'Total Countries per Region')
```



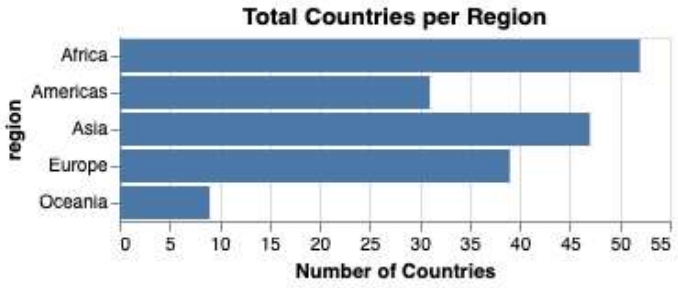
Since we already have filtered the data to year 2018, we know that each country will only occur once in the dataset.So to answer this question, we simply *count()* the number of observations/rows in the dataframe for each region.

https://github.com/ubco-mds-2023/Data-550

Sorting (ascending)

- By default, categorical values are sorted by alphabetical order.
- For most nominal data, it is easier to interpret if the bars are sorted from high to low values since the longest bar is closet to the axis.

```
1  alt.Chart(gm2018).mark_bar().encode(
2      alt.X('count()', title='Number of countries'),
3      alt.Y('region', sort='x'))
```



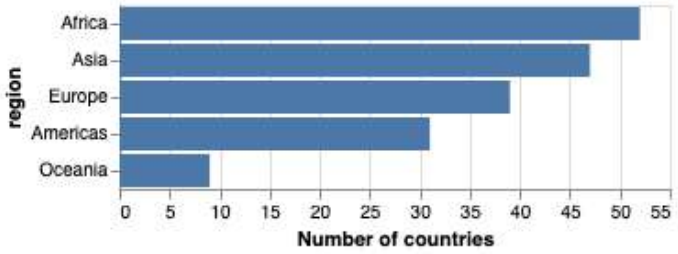
https://github.com/ubco-mds-2023/Data-550

Sorting (descending)

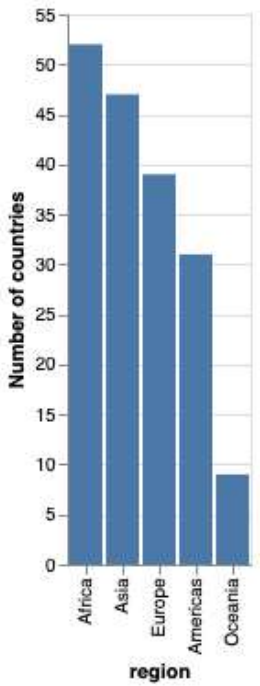
- If you want to instead sort the bars in descending order, use the minus sign before the axis reference
- This can be helpful when making a vertical bar chart since it would then align the tallest bar next to the y-axis.

https://github.com/ubco-mds-2023/Data-550

```
1 alt.Chart(gm2018).mark_bar().encod
2     alt.X('count()', title='Number
3     alt.Y('region', sort='-x'))
```




```
1 alt.Chart(gm2018).mark_bar().encod
2     alt.X('region', sort='-y'),
3     alt.Y('count()', title='Number
```

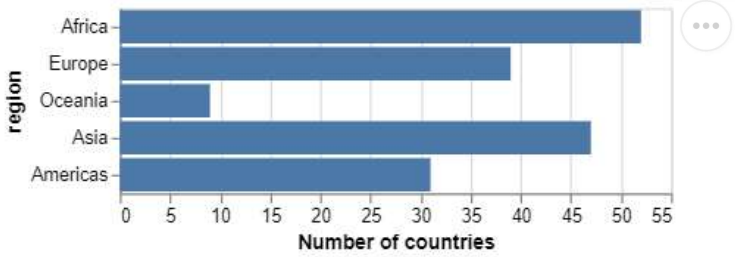


https://github.com/ubco-mds-2023/Data-550

Sort (custom)

A custom order may also be used for the sort.

```
my_order = ['Africa', 'Europe', 'Oceania', 'Asia', 'Americas']
alt.Chart(gm2018).mark_bar().encode(
    alt.X('count()', title='Number of countries'),
    alt.Y('region', sort=my_order))
```



This is useful when we have ordinal data, eg. days of the week.

To learn more about important considerations when plotting counts of categorical

https://github.com/ubco-mds-2023/Data-550

Histograms

- Another popular chart type is a *histogram*

What are the steps for drawing a histogram for a single quantitative feature?

1. divide the range of possible values into groups called *bins*
2. For each bin, a rectangle is constructed with a base length equal to the range of values in that specific bin and height equal to the number of observations falling into that bin.

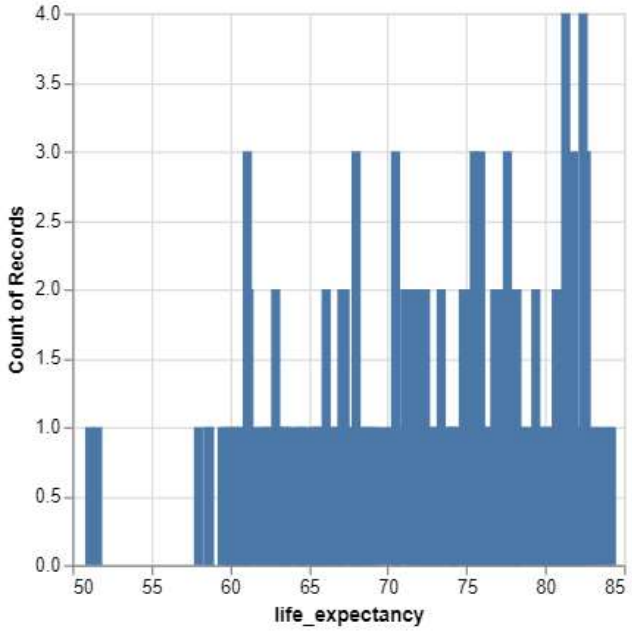
- Referred to as estimating the "distribution" of the data

https://github.com/ubco-mds-2023/Data-550

Estimating Distribution

What is the distribution of life expectancy within our data set?

```
alt.Chart(gm2018).mark_bar().encode(
    alt.X('life_expectancy'),
    alt.Y('count()'))
```

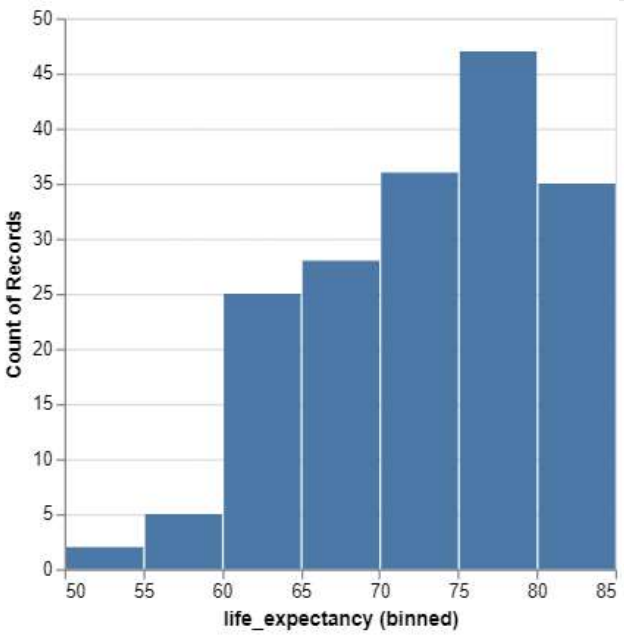


https://github.com/ubco-mds-2023/Data-550

Estimating Distribution (with binning)

We'll want to create "bins" that represent ranges of numerical values and then count all the observations within each of these ranges.

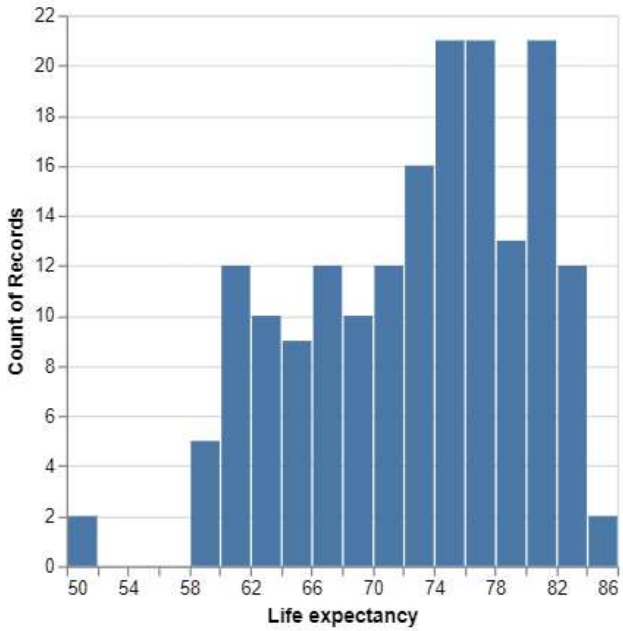
```
alt.Chart(gm2018).mark_bar().encode(
    alt.X('life_expectancy', bin=True), # enable binning
    alt.Y('count()'))
```



https://github.com/ubco-mds-2023/Data-550

Narrowing bins

```
alt.Chart(gm2018).mark_bar().encode(
    alt.X('life_expectancy', bin=alt.Bin(maxbins=30),
    title='Life expectancy'), # renamed x-axis
    alt.Y('count()'))
```



https://github.com/ubco-mds-2023/Data-550

https://github.com/ubco-mds-2023/Data-550