Data Visualization

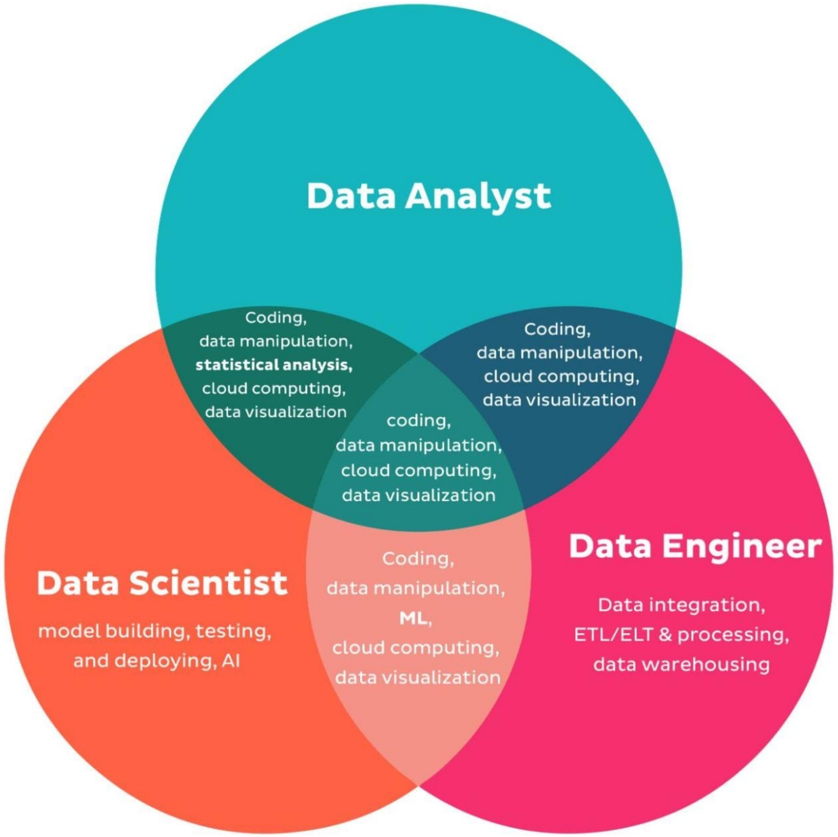
Use of human perception

No inherent spatial dimension

Data visualization is the visual presentation of abstract data using computational means, with the purpose of solving problems, communicating information, and amplifying human cognition.

Form connections and make decisions

Data Visualization Skills



https://www.kdnuggets.com/navigating-data-science-job-titles-data-analyst-vs-data-scientist-vs-data-engineer

Importance of Data Visualization

Dataset I		Dataset II		Dataset III		Dataset IV	
x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

https://en.wikipedia.org/wiki/Anscombe%27s_quartet

Activity: Explore a Dataset

Objective

Calculate and compare basic statistics (mean, standard deviation, variance) for the variables X and Y in the four subsets (Series I, II, III, and IV), and propose appropriate visualizations based on your findings.

Link:
https://colab.research.google.com/drive/1fQvEYa7Q4f0k90ru3nt3GqjZ96kYaamA?usp=sharing

Scatterplot

Data:
- two quantitative values

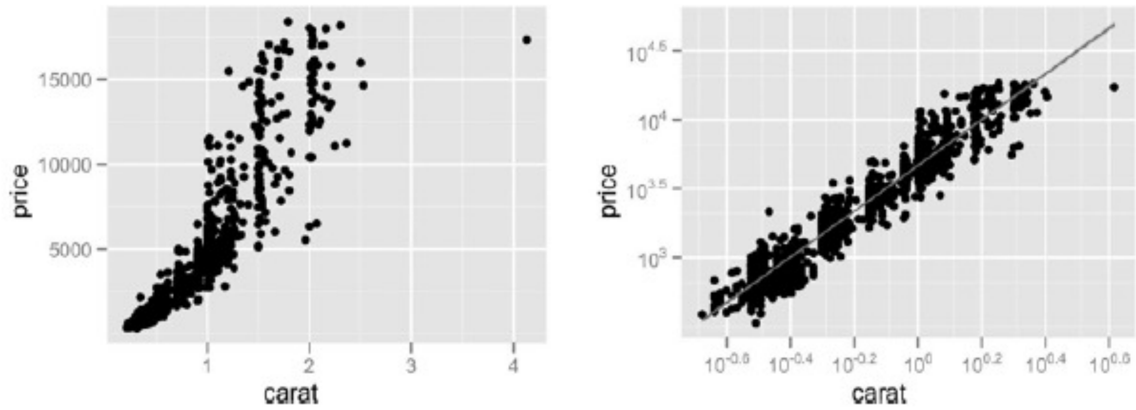
Geometry:
- points

Encoding:
- horizontal and vertical positions

Tasks:
- find trends, outliers, distribution, correlation, clusters

Scalability:
- hundreds of items

Diamond
Price by carat



Line Charts

Data:

- two quantitative values (one maybe time)

Geometry:

- Points (lines connect points)

Encoding:

- Horizontal ordered by a quantitative value
- Vertical length express quantitative attribute

Tasks:

- find trends, correlation

Scalability:

- hundreds of items

Time context

Bar charts

Data:

- One categorical value, one quantitative value

Geometry:

- lines

Encoding:

- Length express quantitative value
- Spatial region: one by categorial value
- Ordered?

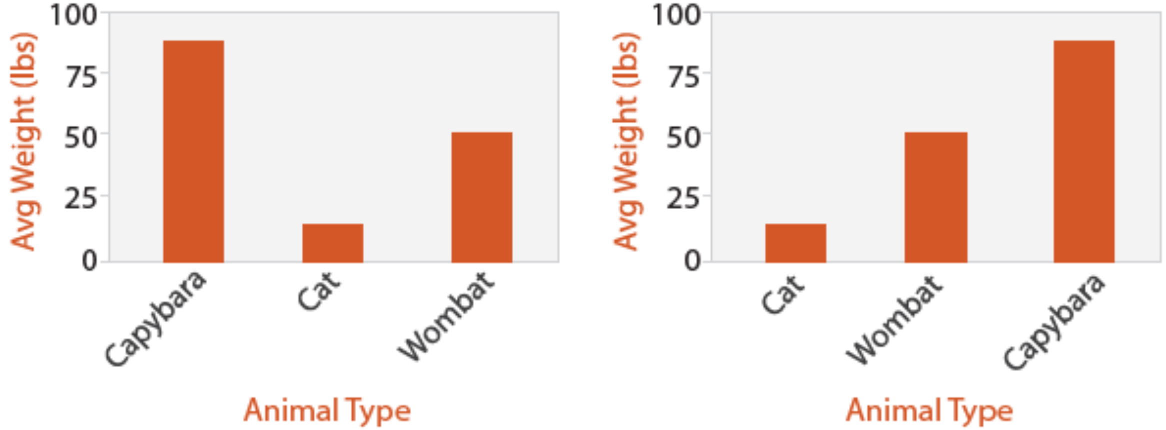
Tasks:

- Compare, lookup values

Scalability:

- Dozens of hundreds of categorial values

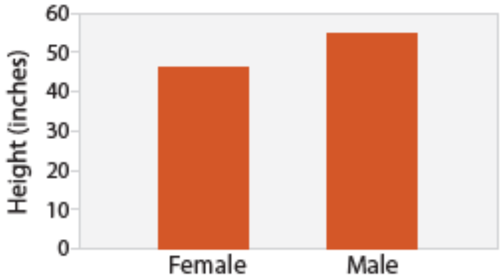
Animals weight

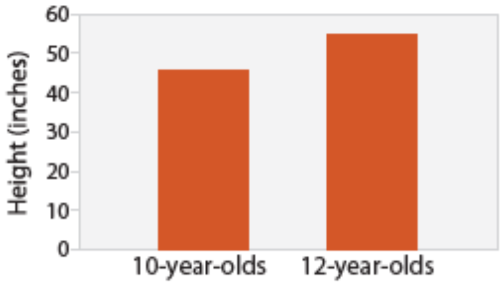


Bar charts vs Line charts

Depends on attributes:

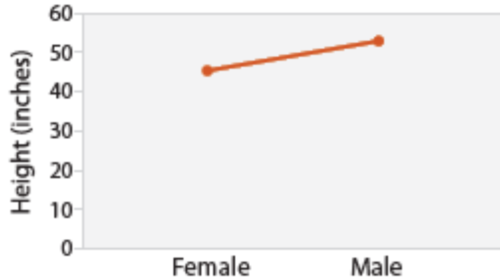
- Categorical: bar chart
- Quantitative ordered: line chart

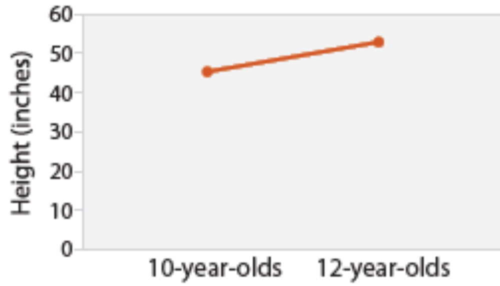




Do not use line chart for categorical attributes

- Implies trending of data
  - Ex: “The more male a person is, the taller he is”





Pie charts

Data: One categorical value, one quantitative value
Geometry:

- Angle: draw pie charts by angle
- Length of the arc of the slice
- Area of the slice

Encoding:

- Area marks with angle channel
- Angle/area more less accuracy than line length

Tasks:

- Compare, part-to-whole relationship

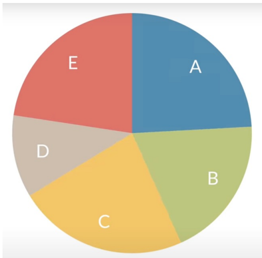
Scalability:

- Max 5 slices

Geometry: angle, length of the arc, area of the arc

Pie charts vs Bar charts

Pie chart:
do not have a common baseline



Bar chart:
common baseline (x-axis)

Downside in pie charts:

- We can not see small differences between slices compared to bar charts

Good in pie charts:

- Part-to-whole relationship: it is really difficult to see what is the percentage that a bar represents in a bar charts. It is more easy in a pie chart

Why We Don't Read them By Angle:

https://www.youtube.com/watch?v=NxmHDNNTFyk

Pie charts, donut charts studies and publications: https://kosara.net/publications.html

Stacked Bar chart

Data:

- two or more categorical values, one quantitative value

Geometry:

- Vertical stack lines

Encoding:

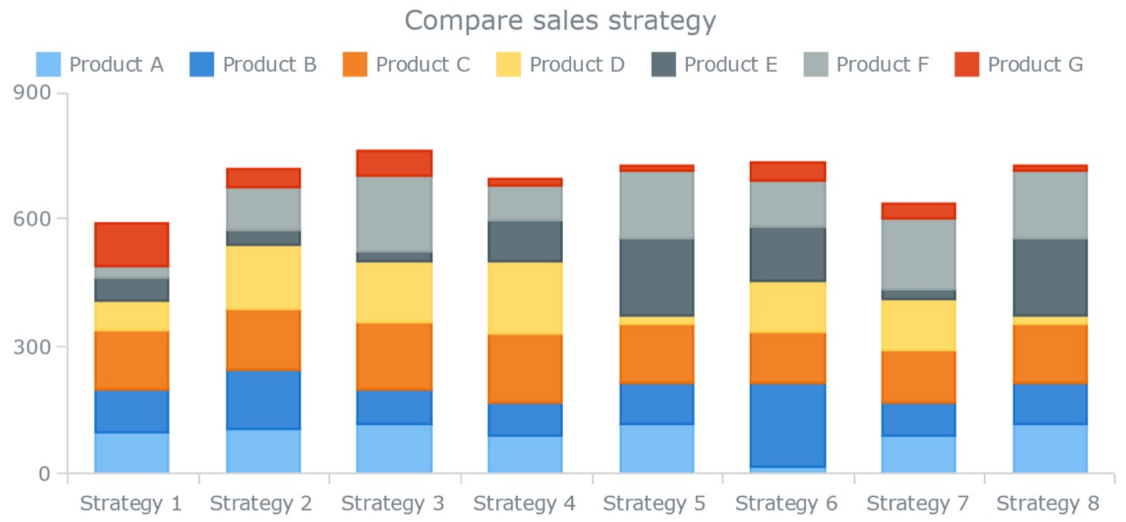
- Length express quantitative value
- Color hue express categories
- Ordered?

Tasks:

- Compare, part-to-whole relationship

Scalability:

- Dozens of categorial values



Heatmap

Data:

- Two categorical attributes (matrix)
- One quantitative attribute

Geometry:

- Area

Encoding:

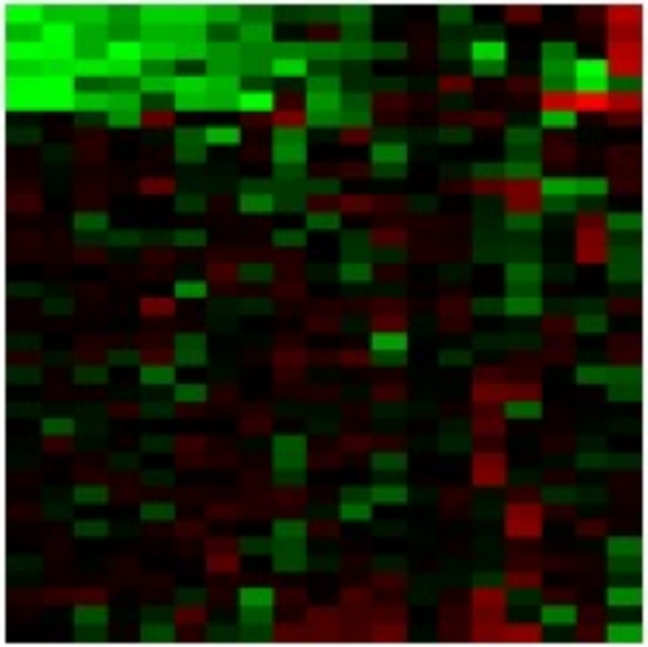
- Color hue express quantitative attribute
    - Diverging colormap

Tasks:

- Find clusters, find outliners

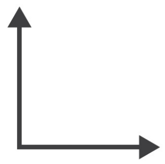
Scalability:

- Hundreds of categorical attributes (depends on the size of screen)

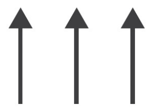


Axis orientation

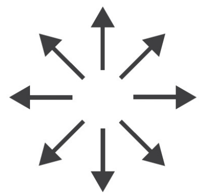
- ➔ Rectilinear



- ➔ Parallel



- ➔ Radial



Scatterplot matrix, parallel coordinates

Scatterplot matrix:

- Axes: rectilinear
- Figure: points
- Scalability: dozen attributes

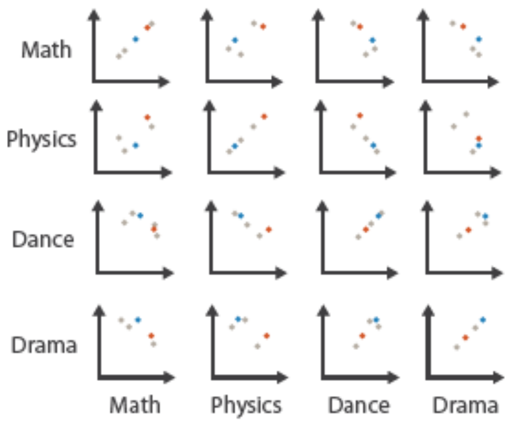
Parallel coordinates:

- Axes: parallel
- Figure: lines representing items
- Scalability: dozen attributes

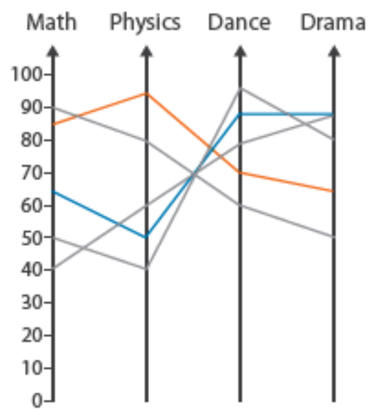
Table

Math	Physics	Dance	Drama
85	95	70	65
90	80	60	50
65	50	90	90
50	40	95	80
40	60	80	90

Scatterplot Matrix



Parallel Coordinates



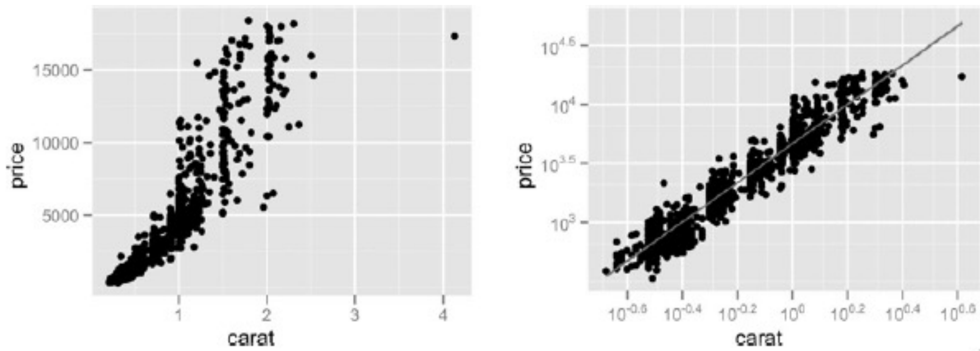
Task: correlation

Scatterplot matrix:

- Positive correlation:
  - Diagonal low-to-high
- Negative correlation:
  - Diagonal high-to-low
- Uncorrelated

Parallel coordinates:

- Positive correlation:
  - Parallel line segments
- Negative correlation:
  - All segments intersect at the midpoint.
- Uncorrelated:
  - Scattered crossing



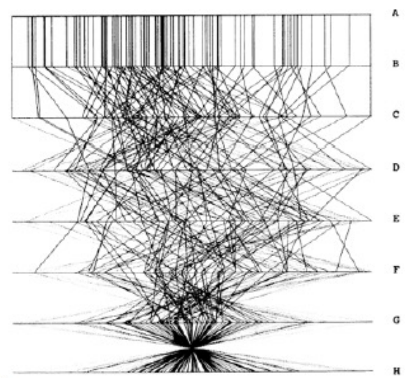
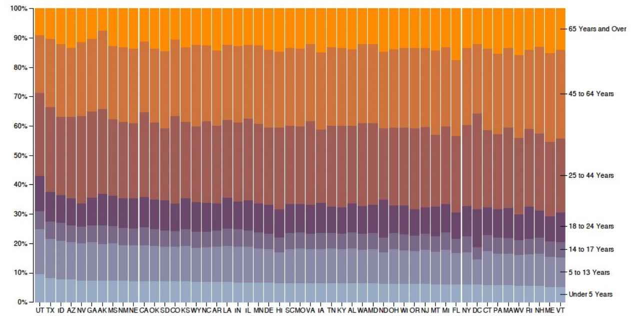
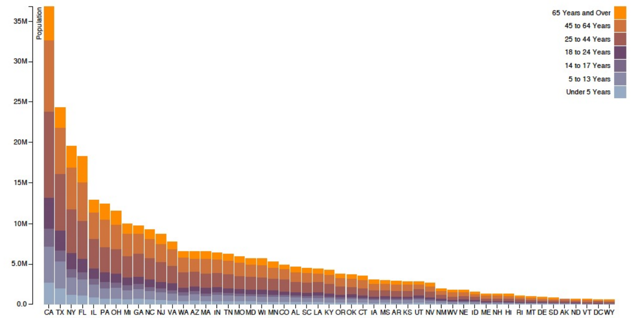
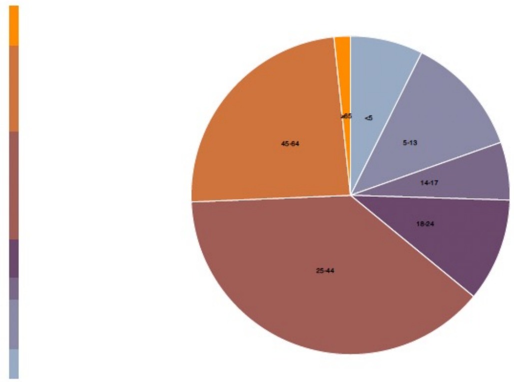


Figure 3. Parallel Coordinate Plot of Six-Dimensional Data Illustrating Correlations of ρ = 1, .8, .2, 0, −.2, −.8, and −1.

Normalized stacked bar chart







Data:

- two or more categorical values, one quantitative value

Geometry:

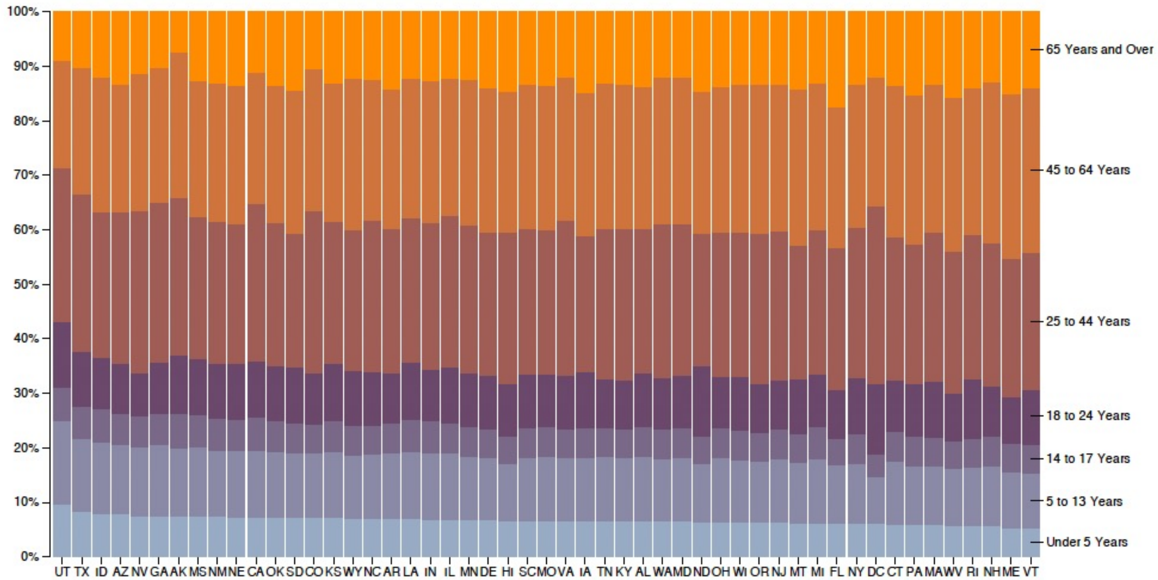
- Vertical stack lines

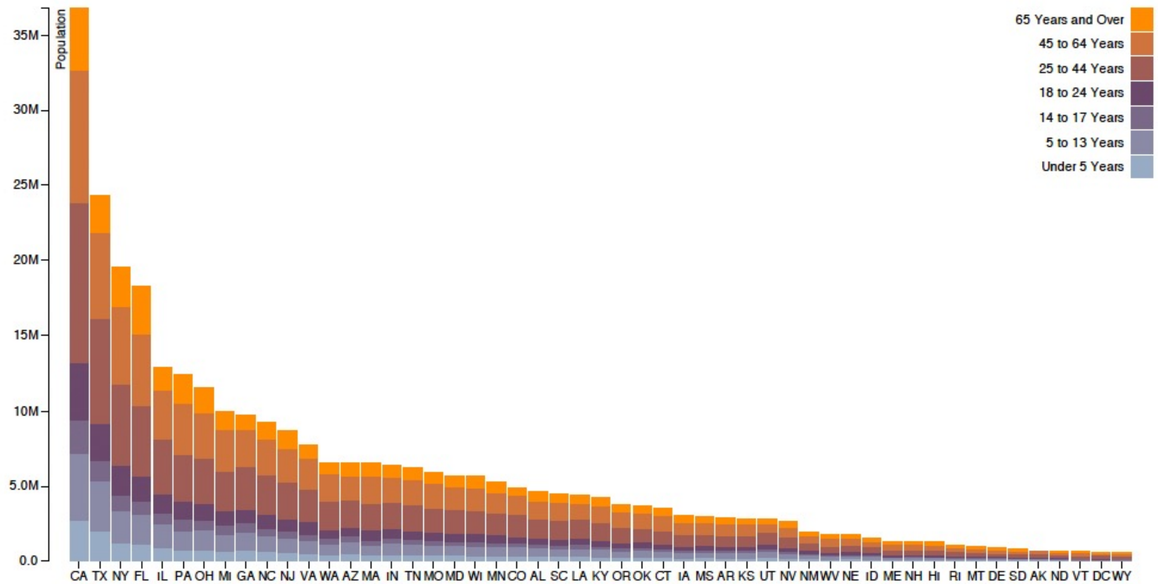
Encoding:

- Stacked bar chart, normalized to full vert height
- Single stacked bar equivalent to full pie
  - high information density: requires narrow rectangle

Tasks:

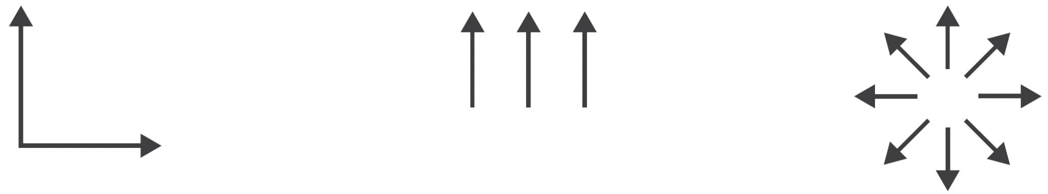
- Compare, part-to-whole relationship


100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
65 Years and Over
45 to 64 Years
25 to 44 Years
18 to 24 Years
14 to 17 Years
5 to 13 Years
Under 5 Years
UT TX ID AZ NV GA AK MS NM NE CA OK SD CO KS WY NC AR LA IN IL MN DE HI SC MO VA IA TN KY AL WA MD ND OH WI OR NJ MT MI FL NY DC CT PA MA WV RI NH ME VT


Population
35M
30M
25M
20M
15M
10M
5.0M
0.0
65 Years and Over
45 to 64 Years
25 to 44 Years
18 to 24 Years
14 to 17 Years
5 to 13 Years
Under 5 Years
CA TX NY FL IL PA OH MI GA NC NJ VA WA AZ MA IN TN MO MD WI MN CO AL SC LA KY OR OK CT IA MS AR KS UT NV NM WV NE ID ME NH HI RI MT DE SD AK ND VT DC WY

Axis orientation limitations

➔ Rectilinear ➔ Parallel ➔ Radial



- Rectilinear: scalability = #axes
  - 2 axes best
  - 3 problematic
  - 4+ impossible
- Parallel: unfamiliarity, training time
- Radial: perceptual limits
  - angles lower precision than lengths
  - asymmetry between angle and length

Links:

- Jon Schwabish, One chart at a time:
  - https://www.youtube.com/watch?v=ClMqlGT4V-M&list=PLfv89tPxlTiVIrwuSBCISiBaGSH1CJR5-&index=18

Approaches to visualize data

Imperative

Low-level

- Specify *how* should be done
- Focus on plot construction details
- Control over plotting details, but laborious for complex visualizations

Declarative

High-level

- Specify ***what*** should be done
- Focus on **data** and **relationships**
- Smart defaults give us what we want without complete control over minor details

The key idea is that you are declaring links between data columns and visual encoding channels, such as the x-axis, y-axis, and color

https://towardsdatascience.com/declarative-vs-imperative-plotting

Vega-Altair: Declarative Visualization in Python

Altair provides a declarative Python API for statistical visualization, built on top of Vega-Lite.

Tidy data

"Tidy data" is a consistent way to structure datasets to facilitate analysis and visualization.

- Each **variable** is a **column**; each column is a variable
- Each **observation** is a **row**; each row is an observation
- Each **value** is a **cell**; each cell is a single value

Wide format (**not** tidy)

Year	sales_ecuador	sales_peru
2020	100	80
2021	150	120

Tidy format

year	country	sales
2020	Ecuador	100
2020	Peru	80
2021	Ecuador	150
2021	Peru	120

https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

Sample data in Altair's companion package vega_datasets

```python
# load a sample dataset as a pandas DataFrame
from vega_datasets import data

cars = data.cars()
cars
```

	Name	Miles_per_Gallon	Cylinders	Displacement	Horsepower	Weight_in_lbs	Acceleration	Year	Origin
0	chevrolet chevelle malibu	18.0	8	307.0	130.0	3504	12.0	1970-01-01	USA
1	buick skylark 320	15.0	8	350.0	165.0	3693	11.5	1970-01-01	USA
2	plymouth satellite	18.0	8	318.0	150.0	3436	11.0	1970-01-01	USA
3	amc rebel sst	16.0	8	304.0	150.0	3433	12.0	1970-01-01	USA
4	ford torino	17.0	8	302.0	140.0	3449	10.5	1970-01-01	USA
...	...	...	...	...	...	...	...	...	...
401	ford mustang gl	27.0	4	140.0	86.0	2790	15.6	1982-01-01	USA
402	vw pickup	44.0	4	97.0	52.0	2130	24.6	1982-01-01	Europe
403	dodge rampage	32.0	4	135.0	84.0	2295	11.6	1982-01-01	USA
404	ford ranger	28.0	4	120.0	79.0	2625	18.6	1982-01-01	USA
405	chevy s-10	31.0	4	119.0	82.0	2720	19.4	1982-01-01	USA

406 rows × 9 columns

Tidy structure:

- rows with one **observation** each
- data **columns** (or fields, variables) with one feature each

Altair supports:

Pandas Dataframes, CSV, TSV, JSON, URL

Sample data in Altair's companion package vega_datasets

```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 406 entries, 0 to 405
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Name              406 non-null    object
 1   Miles_per_Gallon  398 non-null    float64
 2   Cylinders         406 non-null    int64
 3   Displacement      406 non-null    float64
 4   Horsepower        400 non-null    float64
 5   Weight_in_lbs     406 non-null    int64
 6   Acceleration      406 non-null    float64
 7   Year              406 non-null    datetime64[ns]
 8   Origin            406 non-null    object
dtypes: datetime64[ns](1), float64(4), int64(2), object(2)
memory usage: 28.7+ KB
```

Altair uses the pandas data types to infer the data it is working with

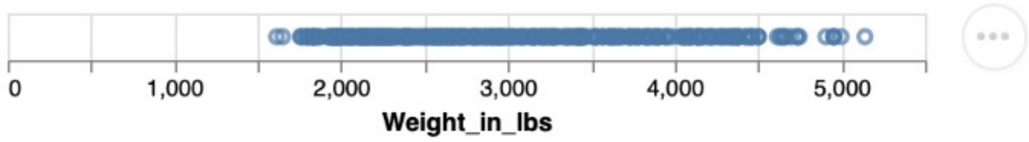
Adding graphical elements via marks

```python
# import altair with an abbreviated alias
import altair as alt

# load a sample dataset as a pandas DataFrame
from vega_datasets import data

cars = data.cars()

# make the chart
alt.Chart(cars).mark_point()
```

Encoding columns as visual channels

Mapping a dataframe *column* (Weight_in_lbs) to the *x-scale*

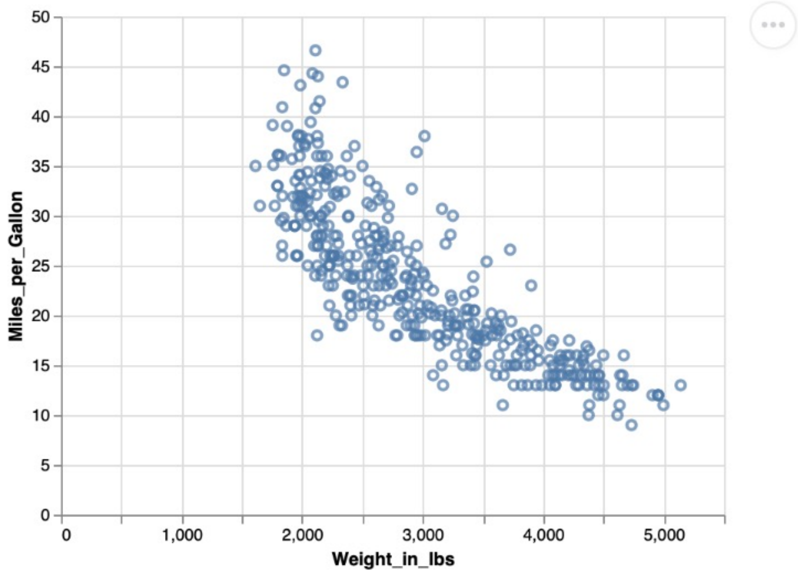
```
# make the chart
alt.Chart(cars).mark_point().encode(
    x='Weight_in_lbs',
)
```



Encoding columns as visual channels

Mapping a dataframe *column* (Miles_per_Gallon) to the *y-scale*

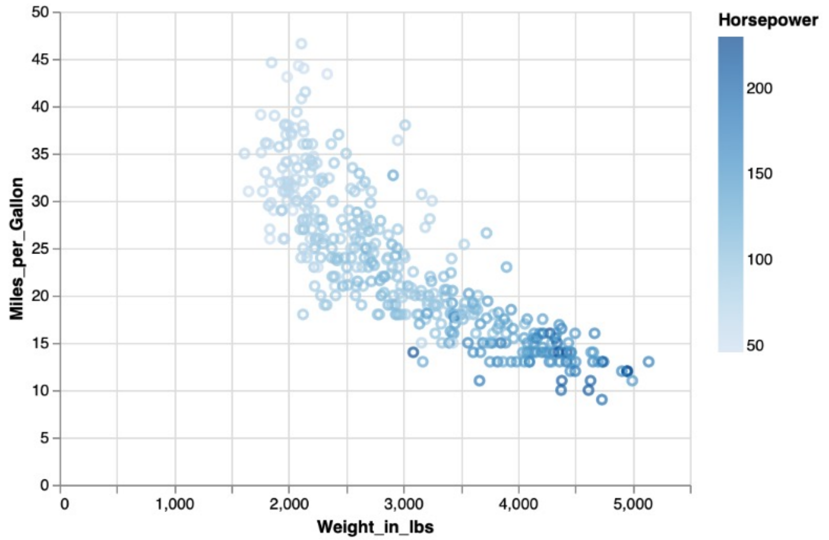
```
# make the chart
alt.Chart(cars).mark_point().encode(
    x='Weight_in_lbs',
    y='Miles_per_Gallon'
)
```



Encoding columns as visual channels

Mapping a *numerical* dataframe *column* (Horsepower) to the *color scale*

```
# make the chart
alt.Chart(cars).mark_point().encode(
    x='Weight_in_lbs',
    y='Miles_per_Gallon',
    color='Horsepower'
)
```

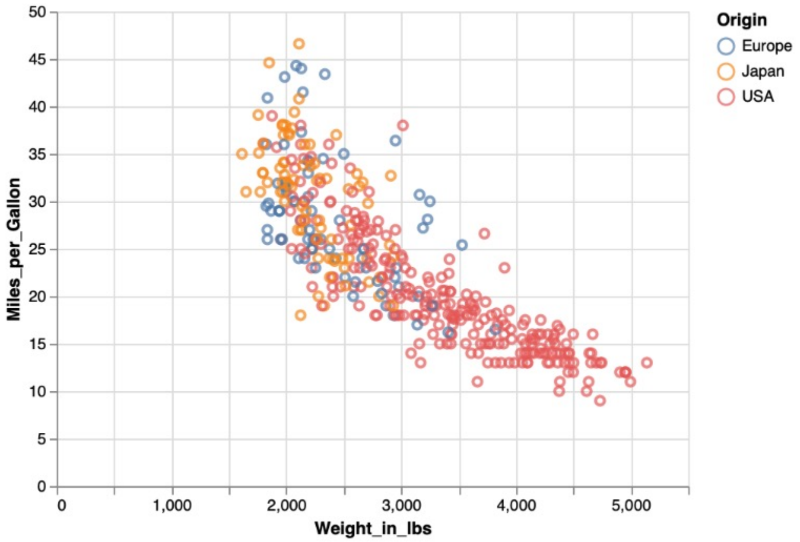


Is there a relationship between horsepower and car weight, or fuel-efficiency?

Encoding columns as visual channels

Mapping a *categorical* dataframe *column* (Origin) to the *color scale*

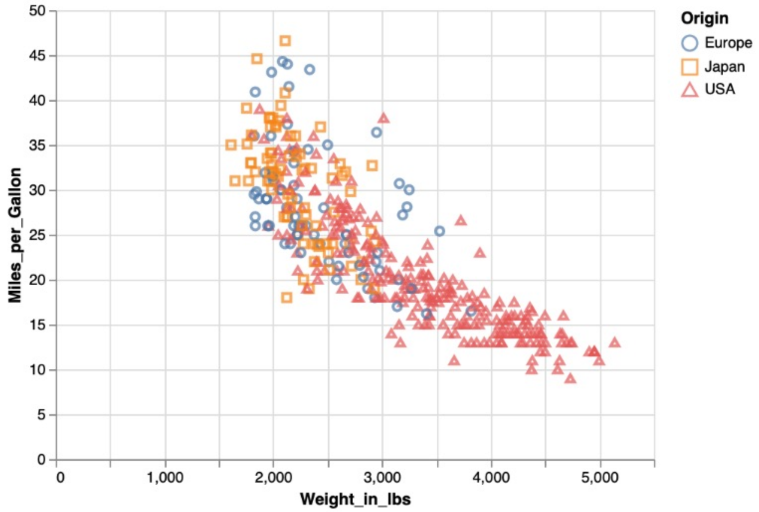
```
# make the chart
alt.Chart(cars).mark_point().encode(
    x='Weight_in_lbs',
    y='Miles_per_Gallon',
    color='Origin'
)
```



Encoding columns as visual channels

Mapping a dataframe *column* (Origin) to the *shape scale*

```
# make the chart
alt.Chart(cars).mark_point().encode(
    x='Weight_in_lbs',
    y='Miles_per_Gallon',
    color='Origin',
    shape='Origin'
)
```

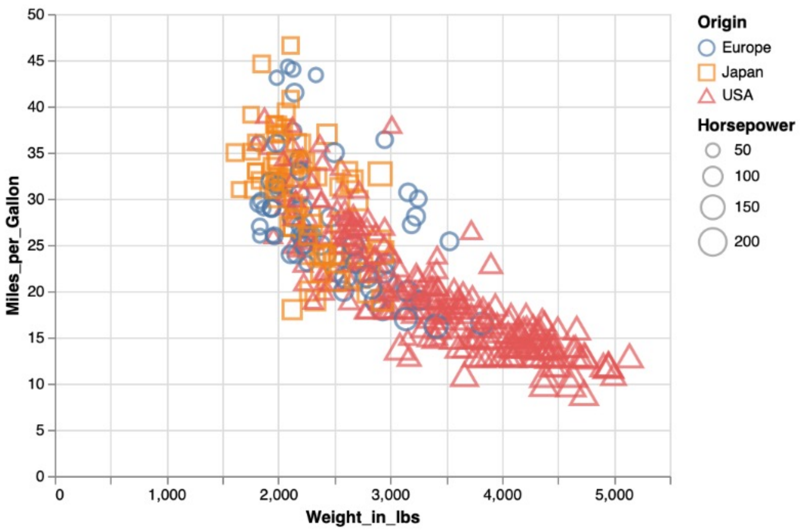


Encoding the same categorical column “**Origin**” as both *color* and *shape*.

Encoding columns as visual channels

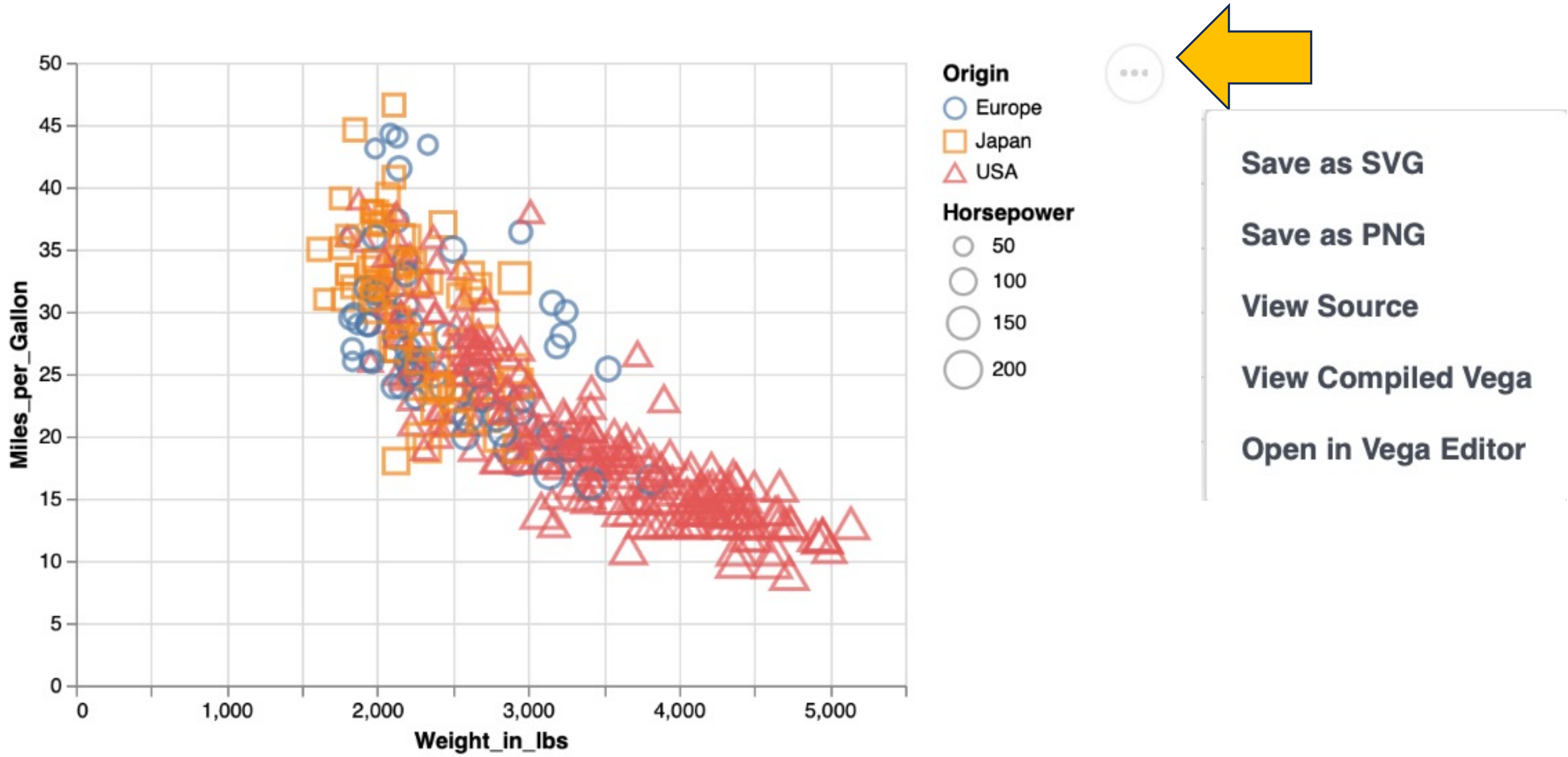
Mapping a dataframe *column* (Horsepower) to the *size scale*

```
# make the chart
alt.Chart(cars).mark_point().encode(
    x='Weight_in_lbs',
    y='Miles_per_Gallon',
    color='Origin',
    shape='Origin',
    size='Horsepower'
)
```



Too much?

Save the plot



Aggregations

Data aggregations are built into Altair

```
# make the chart
alt.Chart(cars).mark_point().encode(
    x='mean(Weight_in_lbs)',
    y='mean(Miles_per_Gallon)',
    color='Origin'
)
```