hci lab
HUMAN × COMPUTER INTERACTION

삼성전자 AI전문가 교육과정 사전교육 - 데이터 분석 및 시각화

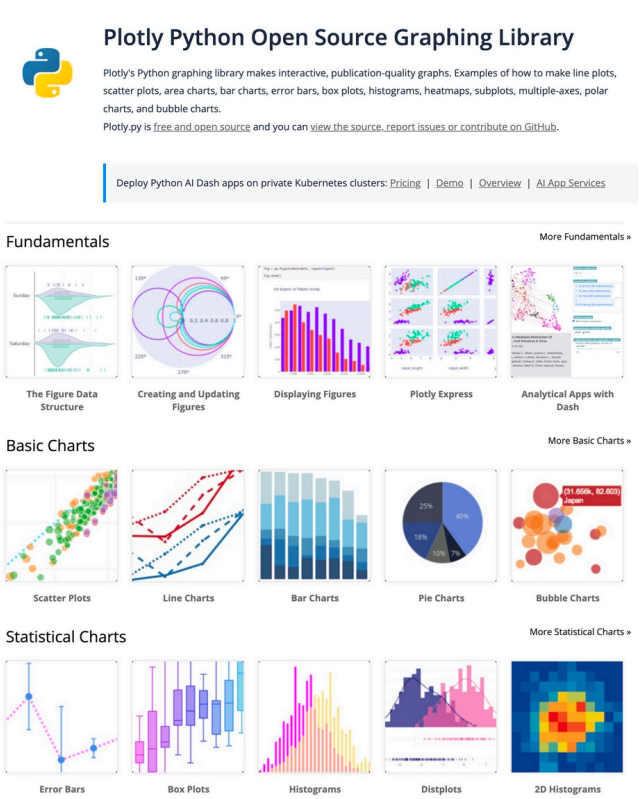
Altair 실습 1 - Intro

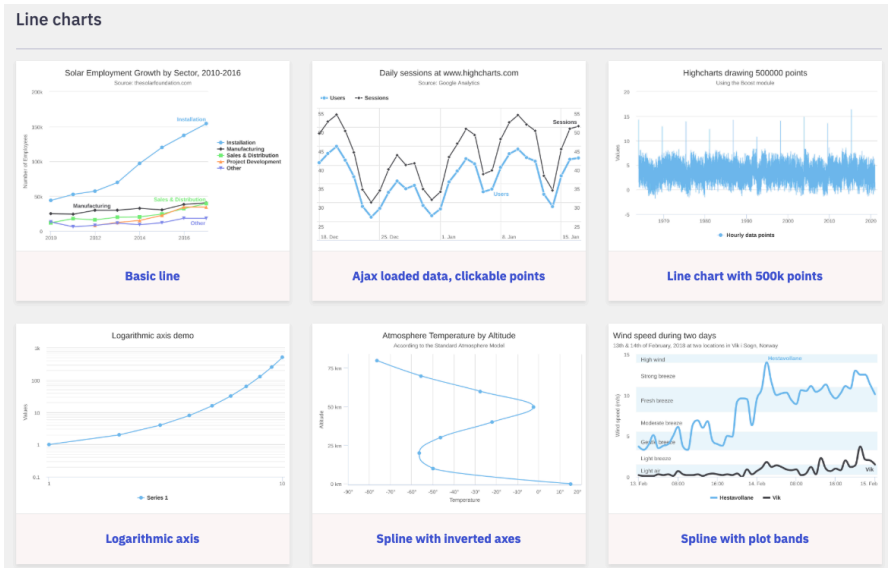
Altair

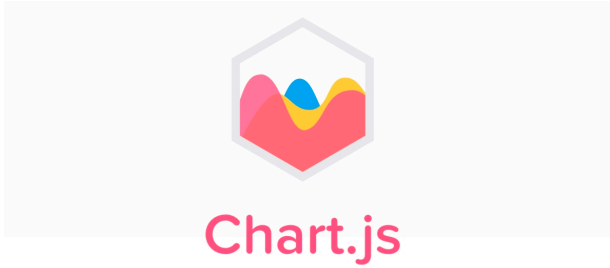
- Altair라는 시각화 라이브러리를 다룹니다
- Google Colab에서 실습을 진행하며 altair, pandas 라이브러리를 사용합니다.

다양한 시각화 라이브러리

- 여전히 많이 쓰이는 차트 타입 기반 시각화 생성 도구
- Plotly, Highcharts, chartjs 등등등…

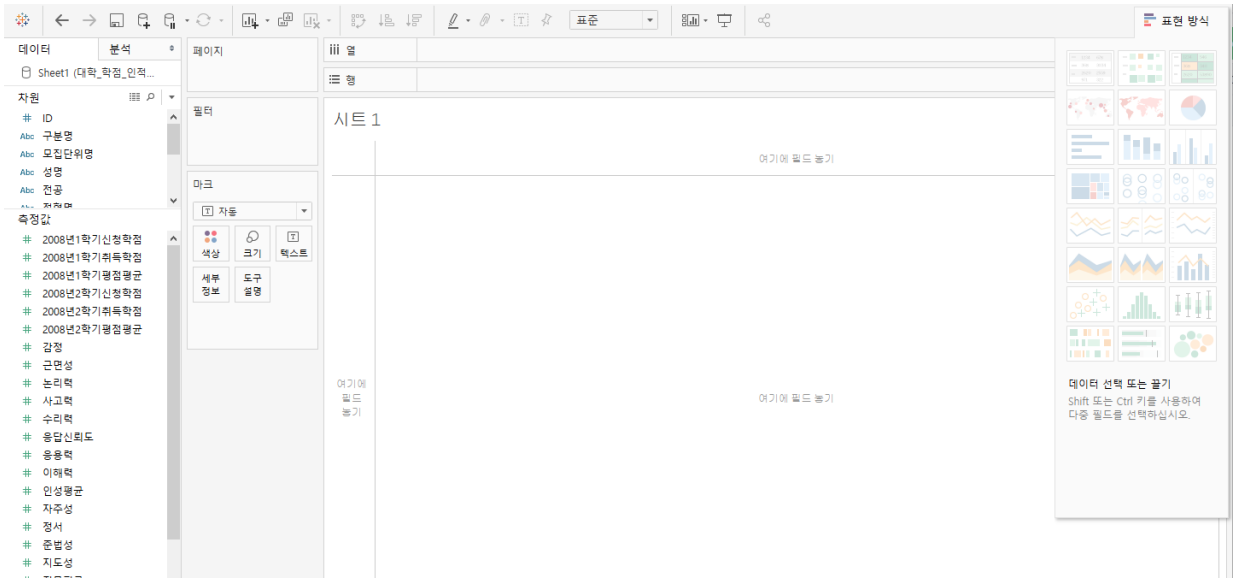


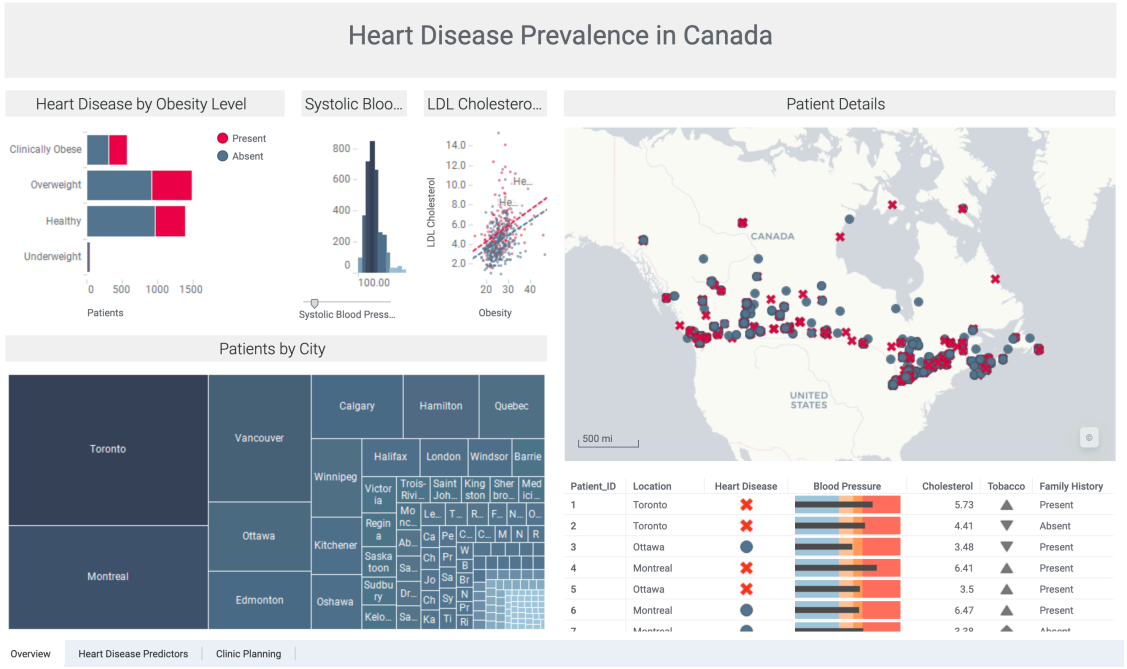




다양한 시각화 라이브러리

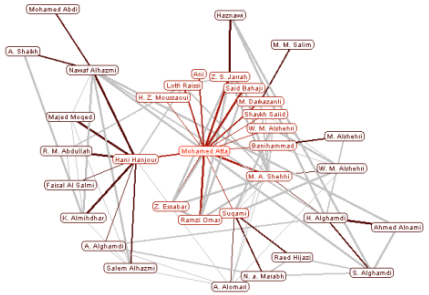
- 상업적인 성공 사례
- Spotfire, Tableau



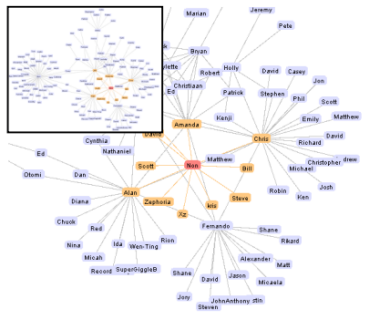


다양한 시각화 라이브러리

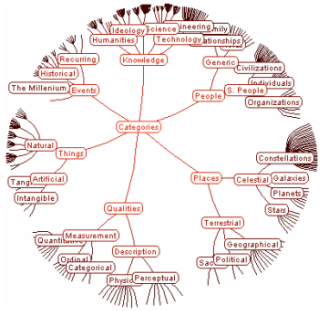
- 한참 전…C# (~2000년대 초반)
- 이후 JAVA 기반 시각화 라이브러리 Prefuse의 등장



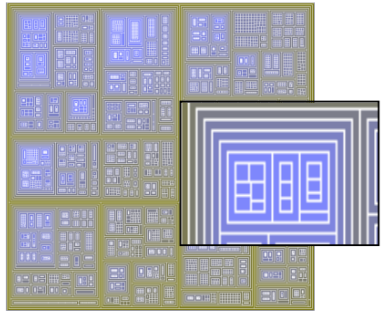
(a) Animated radial layout.



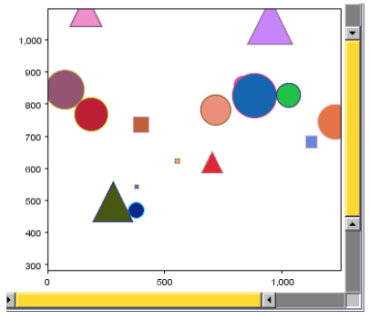
(b) Force-directed layout with overview.



(c) Hyperbolic tree.



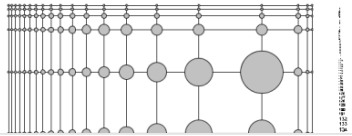
(d) TreeMap.



(e) SpotPlot scatterplot.

Figure 1. Sample prefuse visualizations.

422



Prefuse: a toolkit for interactive information visualization

J Heer, SK Card, JA Landay - Proceedings of the SIGCHI conference on …, 2005 - dl.acm.org

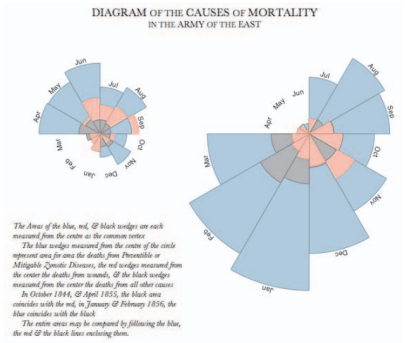
Although information visualization (infovis) technologies have proven indispensable tools for making sense of complex data, wide-spread deployment has yet to take hold, as successful infovis applications are often difficult to author and require domain-specific customization. To …

☆ Cited by 1017 Related articles All 39 versions »

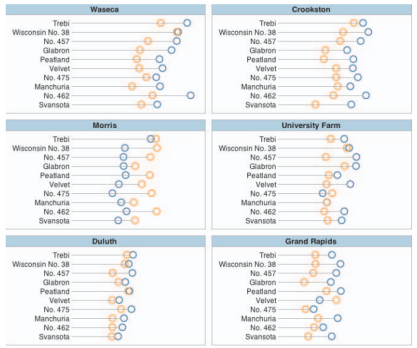
다양한 시각화 라이브러리

- Protovis : 최초의 웹 기반 시각화 라이브러리
- 당시 인기를 끌던 ActionScript, Flash 기반으로 작성됨

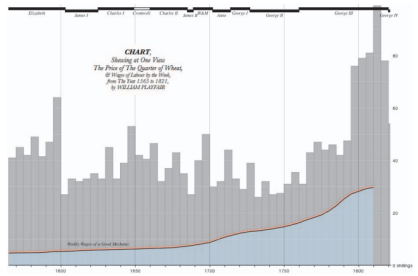
(a)



(b)



(c)



(d)

(e)

Protovis: A graphical toolkit for visualization

M **Bostock**, J **Heer** - IEEE transactions on visualization and …, 2009 - ieeexplore.ieee.org

Despite myriad tools for visualizing data, there remains a gap between the notational efficiency of high-level visualization systems and the expressiveness and accessibility of low-level graphical systems. Powerful visualization systems may be inflexible or impose …

☆ Cited by 462 Related articles All 17 versions »

다양한 시각화 라이브러리

- 대표적인 open source library : D3.js
- 웹 브라우저 상의 시각적 요소 (=마크)의 속성 (=채널)에 데이터를 매핑 가능한 선언적 시각화 라이브러리

Data-Driven Documents

Jeffrey Heer

mike bostock

다양한 시각화 라이브러리

- 대표적인 open source library : D3.js
- 웹 브라우저 상의 시각적 요소 (=마크)의 속성 (=채널)에 데이터를 매핑 가능한 선언적 시각화 라이브러리
- 2011년 IEEE InfoVis 발표 => 이후 필수 기술로 자리매김

다양한 시각화 라이브러리

$\mathbb{D}^3$: Data-Driven Documents

Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer

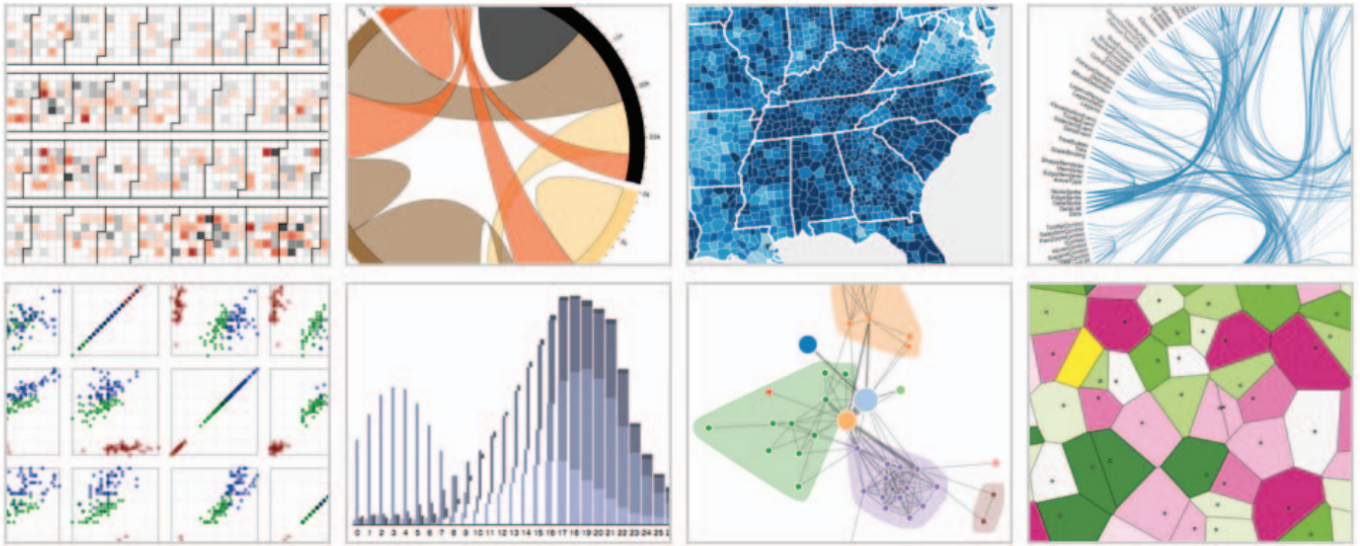


Fig. 1. Interactive visualizations built with D3, running inside Google Chrome. From left to right: calendar view, chord diagram, choro-
pleth map, hierarchical edge bundling, scatterplot matrix, grouped & stacked

Abstract—Data-Driven Documents (D3) is a novel representation-transparer
the underlying scenegraph within a toolkit-specific abstraction, D3 enables
tation: the standard *document object model* (DOM). With D3, designers sel
applying dynamic transforms to both generate and modify content. We show
ness and better integrates with developer tools than prior approaches, whil
powerful declarative components. Immediate evaluation of operators furthe
Additionally, we demonstrate how D3 transforms naturally enable animation
over intermediate representations.

Index Terms—Information visualization, user interfaces, toolkits, 2D graphic

D³ data-driven documents

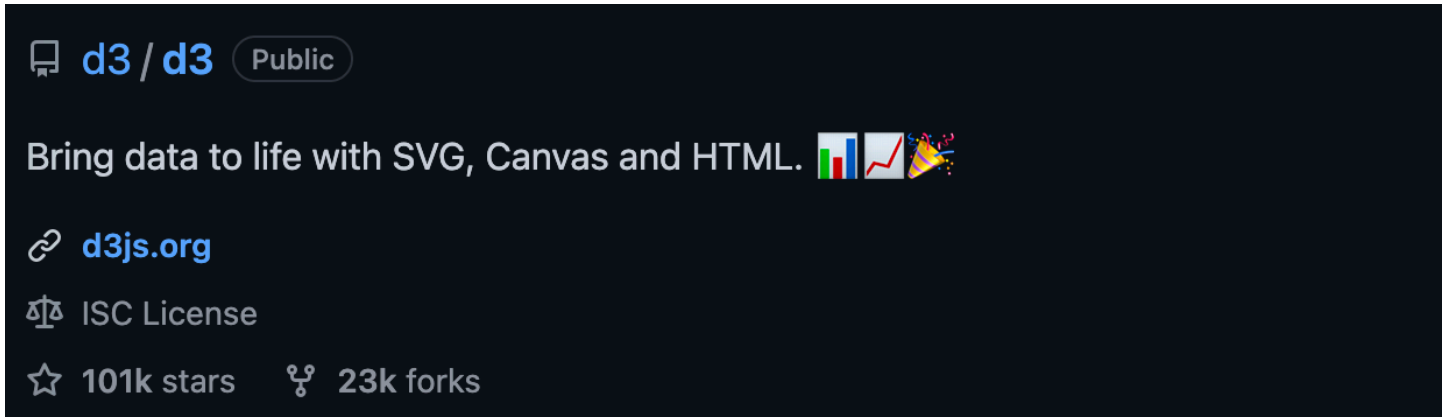
M Bostock, V Ogievetsky, J Heer - IEEE transactions on …, 2011 - ieeexplore.ieee.org

… abstraction, **D3** enables direct inspection and manipulation of a native representation: the standard document object model (DOM). With **D3**, designers selectively bind input **data** to …

☆ 저장 99 인용 3675회 인용 관련 학술자료 전체 20개의 버전

다양한 시각화 라이브러리

- 대표적인 open source library: D3
- Data-Driven Documents
    - Javascript library
    - Famous
    - Citation / star 많음
    - 꾸준한 업데이트 및 기능 추가
    - 높은 자유도
    - 원하는 시각화는 뭐든 그릴 수 있음
    - dom에 직접 접근 및 조작
- d3가 할 수 있는 일들 (gallery)
- Github



다양한 시각화 라이브러리

- D3
- 그러나…
    - D3의 몇 가지 단점
        - 자유도가 높은 만큼 어려움
        - Learning Curve가 가파름
        - 어렵고, 복잡하고, 새로운 시각화에는 유리하지만
        - 간단하고 Conventional한 시각화를 그리기에는 overhead 존재

다양한 시각화 라이브러리

- D3
- 그러나...
  - D3의 몇 가지 단점
    - 자유도가 높은 만큼 어려움
    - Learning Curve가 가파름
    - 어렵고, 복잡하고, 새로운 시각화에는 유리하지만
    - 간단하고 Conventional한 시각화를 그리기에는 overh

```
// Copyright 2021 Observable, Inc.
// Released under the ISC license.
// https://observablehq.com/@d3/bar-chart
function BarChart(data, {
  x = (d, i) => i, // given d in data, returns the (ordinal) x-value
  y = d => d, // given d in data, returns the (quantitative) y-value
  title, // given d in data, returns the title text
  marginTop = 20, // the top margin, in pixels
  marginRight = 0, // the right margin, in pixels
  marginBottom = 30, // the bottom margin, in pixels
  marginLeft = 40, // the left margin, in pixels
  width = 640, // the outer width of the chart, in pixels
  height = 400, // the outer height of the chart, in pixels
  xDomain, // an array of (ordinal) x-values
  xRange = [marginLeft, width - marginRight], // [left, right]
  yType = d3.scaleLinear, // y-scale type
  yDomain, // [ymin, ymax]
  yRange = [height - marginBottom, marginTop], // [bottom, top]
  xPadding = 0.1, // amount of x-range to reserve to separate bars
  yFormat, // a format specifier string for the y-axis
  yLabel, // a label for the y-axis
  color = "currentColor" // bar fill color
} = {}) {
  // Compute values.
  const X = d3.map(data, x);
  const Y = d3.map(data, y);

  // Compute default domains, and unique the x-domain.
  if (xDomain === undefined) xDomain = X;
  if (yDomain === undefined) yDomain = [0, d3.max(Y)];
  xDomain = new d3.InternSet(xDomain);

  // Omit any data not present in the x-domain.
  const I = d3.range(X.length).filter(i => xDomain.has(X[i]));

  // Construct scales, axes, and formats.
  const xScale = d3.scaleBand(xDomain, xRange).padding(xPadding);
  const yScale = yType(yDomain, yRange);
  const xAxis = d3.axisBottom(xScale).tickSizeOuter(0);
  const yAxis = d3.axisLeft(yScale).ticks(height / 40, yFormat);

  // Compute titles.
  if (title === undefined) {
    const formatValue = yScale.tickFormat(100, yFormat);
    title = i => `${X[i]}\n${formatValue(Y[i])}`;
  } else {
    const O = d3.map(data, d => d);
    const T = title;
    title = i => T(O[i], i, data);
  }

  const svg = d3.create("svg")
      .attr("width", width)
      .attr("height", height)
      .attr("viewBox", [0, 0, width, height])
      .attr("style", "max-width: 100%; height: auto; height: intrinsic;");

  svg.append("g")
      .attr("transform", `translate(${marginLeft},0)`)
      .call(yAxis)
      .call(g => g.select(".domain").remove())
      .call(g => g.selectAll(".tick line").clone()
          .attr("x2", width - marginLeft - marginRight)
          .attr("stroke-opacity", 0.1))
      .call(g => g.append("text")
```

다양한 시각화 라이브러리

- D3
- 그러나…
  - D3의 몇 가지 단점
    - 자유도가 높은 만큼 어려움
    - Learning Curve가 가파름
    - 어렵고, 복잡하고, 새로운 시각화에는 유리하지만
    - 간단하고 Conventional한 시각화를 그리기에는 overhead 존재
  - ex) Tooltip
    - 정말 간단한 기능이지만, 굉장히 많은 코드를 작성해야 함

다양한 시각화 라이브러리

- D3
- 그러나…
    - D3의 좀 심각한 단점?
        - js와 html, dom에 대해 알아야 함
        - 아니면 다루기조차 힘듦…

다양한 시각화 라이브러리

- D3
- 그래도...
    - D3의 이로 인한 장점
        - react, vue와 같은 Web Framework와 궁합이 좋다
        - 시각화 툴이 아니더라도
        - 웹개발에 소소한 도움이 됨
- 어쨌든!!

다양한 시각화 라이브러리

- VEGA
  - D3의 단점을 상쇄하기 위해 D3를 wrapping한 라이브러리
  - D3를 기반으로 시각화를 기술하는 시각화 문법 을 제시함
  - 자주 사용하는 거의 “모든 " 시각화를 기술할 수 있는 문법 제시
  - Vega renderer는 JSON으로 된 명세에 맞는 시각화 생성 및 제공

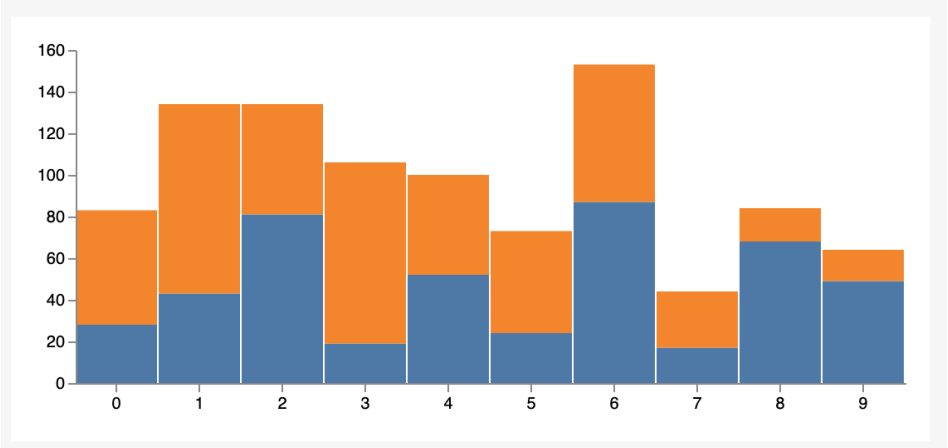
다양한 시각화 라이브러리

- VEGA
  - D3의 단점을 상쇄하기 위해 D3를 wrapping한 라이브러리
  - D3를 기반으로 시각화를 기술하는 시각화 문법 을 제시함
  - 자주 사용하는 거의 “모든 " 시각화를 기술할 수 있는 문법 제시
  - Vega renderer는 JSON으로 된 명세에 맞는 시각화 생성 및 제공
  - 그러나…
    - JSON 형태로 데이터 전달 => 직관적이지 않음
    - 거추장스러운 부분이 많음
    - 또한 많은 사람들에게 익숙한 형식이 아님

다양한 시각화 라이브러리

- VEGA : D3의 단점을 상쇄하기 위해 D3를 wrapping한 라이브러리

```
"scales": [
  {
    "name": "x",
    "type": "band",
    "range": "width",
    "domain": {"data": "table", "field": "x"}
  },
  {
    "name": "y",
    "type": "linear",
    "range": "height",
    "nice": true, "zero": true,
    "domain": {"data": "table", "field": "y1"}
  },
  {
    "name": "color",
    "type": "ordinal",
    "range": "category",
    "domain": {"data": "table", "field": "c"}
  }
],

"axes": [
  {"orient": "bottom", "scale": "x", "zindex": 1},
  {"orient": "left", "scale": "y", "zindex": 1}
],

"marks": [
  {
    "type": "rect",
    "from": {"data": "table"},
    "encode": {
      "enter": {
        "x": {"scale": "x", "field": "x"},
        "width": {"scale": "x", "band": 1, "offset": -1},
        "y": {"scale": "y", "field": "y0"},
        "y2": {"scale": "y", "field": "y1"},
        "fill": {"scale": "color", "field": "c"}
      },
      "update": {
        "fillOpacity": {"value": 1}
      },
      "hover": {
        "fillOpacity": {"value": 0.5}
```



다양한 시각화 라이브러리

Vega-Lite: A Grammar of Interactive Graphics

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer

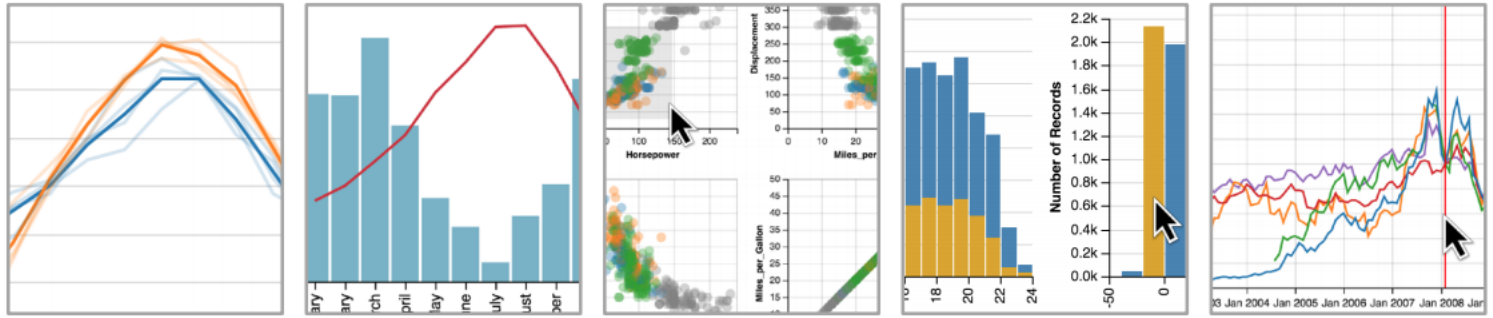


Fig. 1. Example visualizations authored with Vega-Lite. From left-to-right: layered line chart combining raw and average values, dual-axis layered bar and line chart, brushing and linking in a scatterplot matrix, layered cross-filtering, and an interactive index chart.

Abstract—We present Vega-Lite, a high-level grammar that enables rapid specification of *interactive* data visualizations. Vega-Lite combines a traditional grammar of graphics, providing visual encoding rules and a composition algebra for layered and multi-view displays, with a novel grammar of interaction. Users specify interactive semantics by composing *selections*. In Vega-Lite, a selection is an abstraction that defines input event processing, points of interest, and a predicate function for inclusion testing. Selections parameterize visual encodings by serving as input data, defining scale extents, or by driving conditional logic. The Vega-Lite compiler automatically synthesizes requisite data flow and event handling logic, which users can override for further customization. In contrast to existing reactive specifications, Vega-Lite selections decompose an interaction design into concise, enumerable semantic units. We evaluate Vega-Lite through a range of examples, demonstrating succinct specification of both customized interaction methods and common techniques such as panning, zooming, and linked selection.

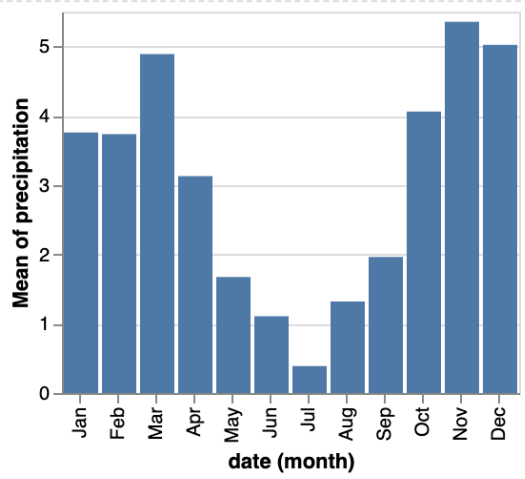
Index Terms—Information visualization, interaction, systems, toolkits, declarative specification

다양한 시각화 라이브러리

- VEGA-Lite
  - Vega의 거추장스러운 문법을 간결하게 만듦
  - Vega와 호환 가능
  - Interaction 가능
  - 역시 Vega 인터프리터에 의해 해석되어 렌더링 됨

다양한 시각화 라이브러리

- VEGA-Lite
  - Bar chart
  - Vega 80줄 => Vega-Lite 10줄



Open in Vega Editor

```
{
  "data": {"url": "data/seattle-weather.csv"},
  "mark": "bar",
  "encoding": {
    "x": {
      "timeUnit": "month",
      "field": "date",
      "type": "ordinal"
    },
    "y": {
      "aggregate": "mean",
      "field": "precipitation",
      "type": "quantitative"
    }
  }
}
```

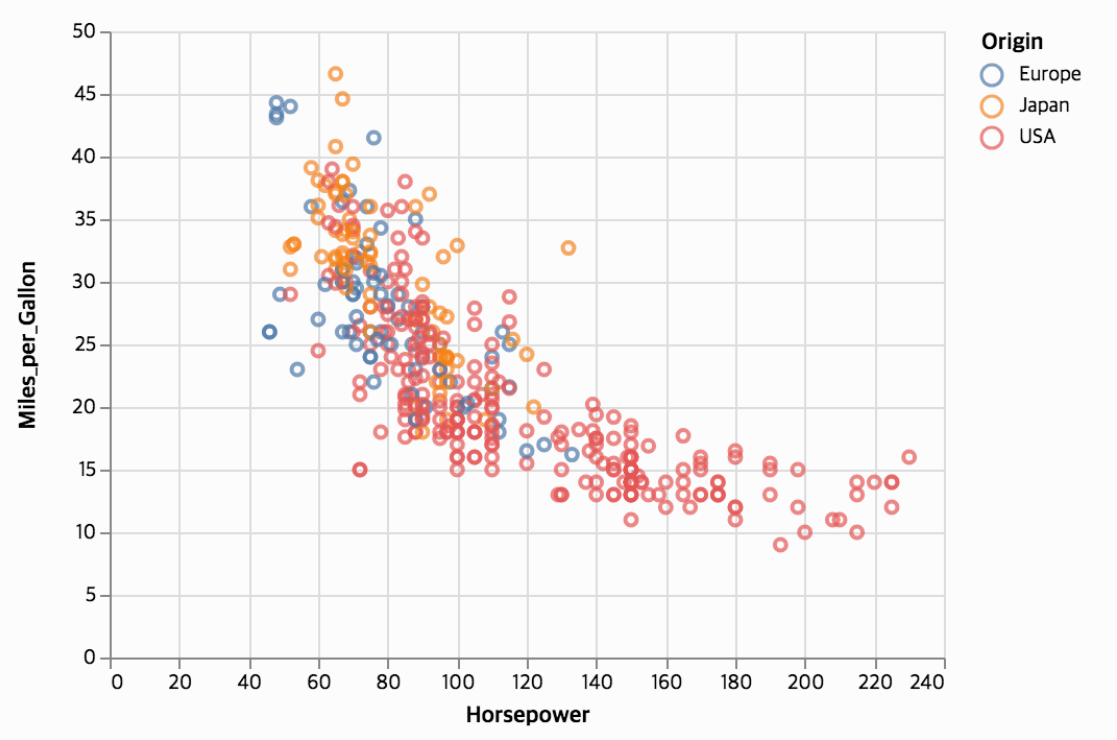
다양한 시각화 라이브러리

- VEGA-Lite
  - 여전히 JSON기반의 명세 사용
  - 익숙하지 않은 형태
  - 한번 더 wrapping한 altair 등장!

Altair 소개

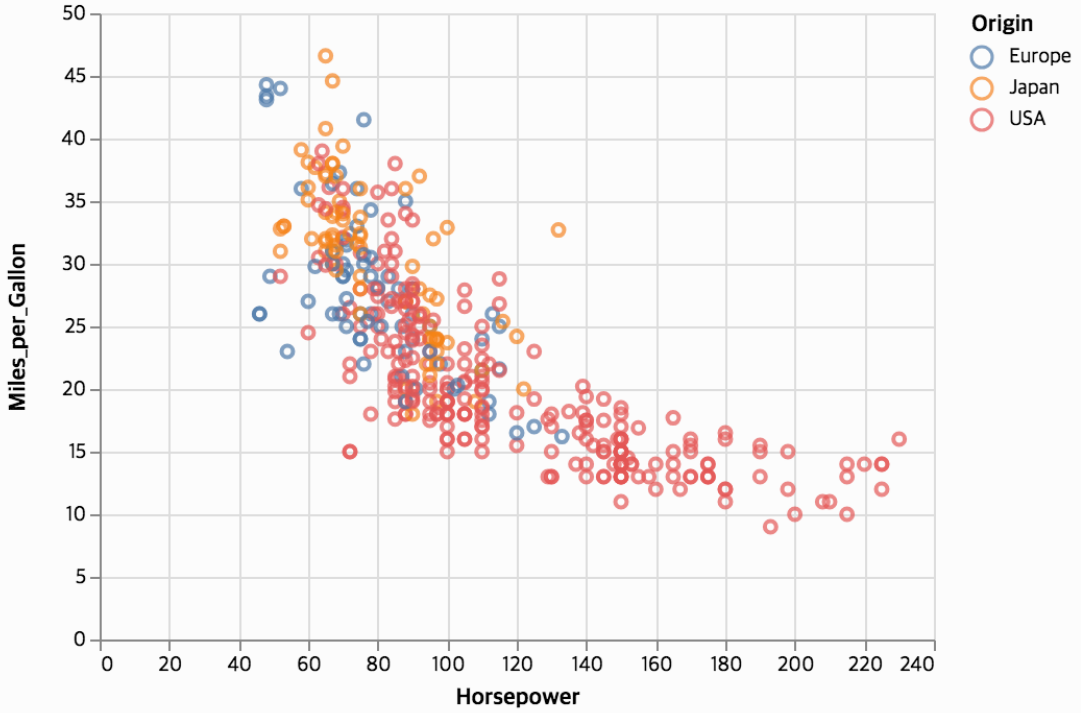
- Vega-Lite에 기반한 Python declarative statistical Visualization library
- Python code를 입력하면…
- Vega-Lite 명세를 가지는 JSON 형태로 번역해서 시각화

```
import altair as alt
from vega_datasets import data

source = data.cars()

alt.Chart(source).mark_circle(size=60).encode(
    x='Horsepower',
    y='Miles_per_Gallon',
    color='Origin',
    tooltip=['Name', 'Origin', 'Horsepower', 'Miles_per_Gallon']
).interactive()
```



Altair 소개

- Vega-Lite에 기반한 Python declarative statistical Visualization library
- 시각화 파이프라인
  - 지금까지 살펴본 것들의 총집합
  - Altair 라이브러리를 Python에서 작성
  - 인터프리터가 Vega-life 문법으로 번역 (JSON)
  - Vega 인터프리터가 이를 해석해 D3로 번역 (JS)
  - 웹 브라우저가 그림



Altair 소개

- Vega-Lite에 기반한 Python declarative statistical Visualization library
- 연표
  - Tableau (2003)
  - D3 (2011)
  - Vega (2013)
  - Vega-Lite (2016)
  - Altair (2016)

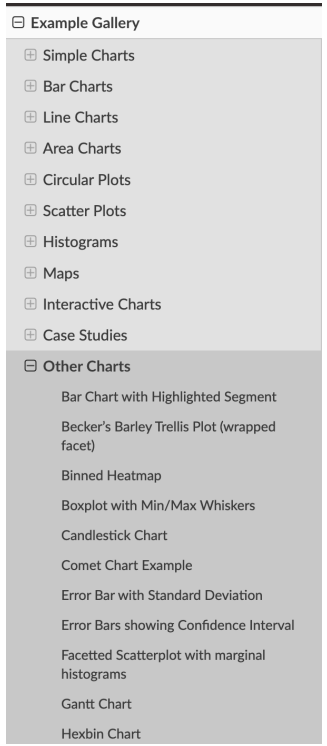
Altair: Interactive Statistical Visualizations for Python

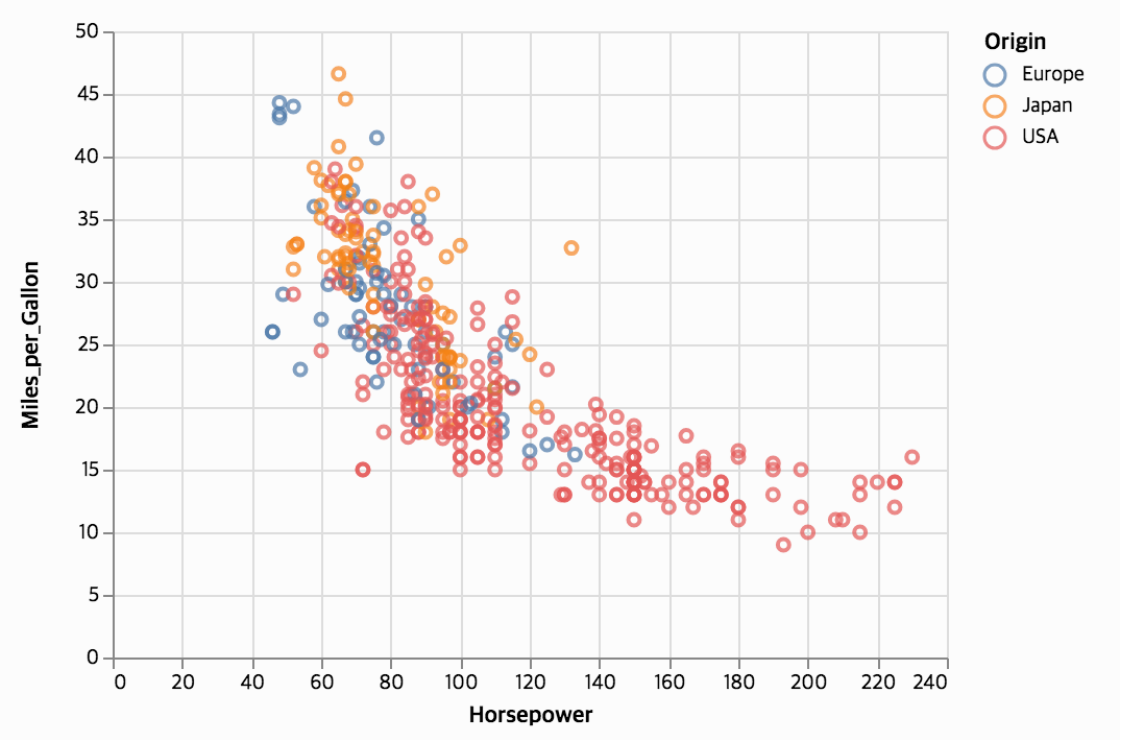
Jacob VanderPlas[1], Brian E. Granger[2], Jeffrey Heer[1], Dominik Moritz[1], Kanit Wongsuphasawat[1], Arvind Satyanarayan[3], Eitan Lees[4], Ilia Timofeev[5], Ben Welsh[6], and Scott Sievert[7]

1 University of Washington **2** California Polytechnic State University, San Luis Obispo **3** MIT CSAIL **4** Florida State University **5** TTS Consulting **6** Los Angeles Times Data Desk **7** University of Wisconsin–Madison

Altair 소개

- Vega-Lite에 기반한 Python declarative statistical Visualization library
- 간결하고 빠른 Visualization grammar 제공
- 다양한 Visualization을 만들수 있음
  - Example gallery





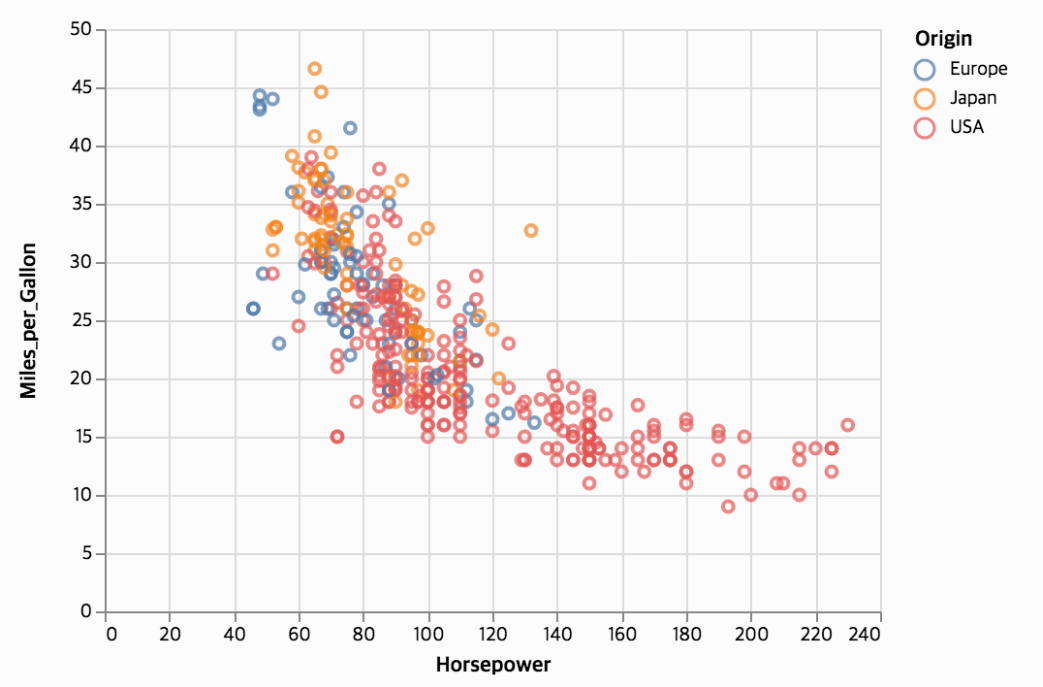
Altair 소개

- Declarative?
  - Visualization에 무엇을 포함하고 싶은지 (Data, Graphical marks, and Encoding channels)
  - High-level의 명세로 제공해주기만 하면 된다.
  - 다른 라이브러리들에서는 어떻게 시각화를 구현해야할지 low-level drawing commands와 loop를 사용하여 일일이 정의해줘야 한다. (D3)

- 정리
  - Altair는 Data column과 Visual encoding channel들 간에 link만 정의
  - 시각화의 다른 부분들은 Altair가 알아서 처리해준다.

- Altair로 Scatterplot (산포도) 그리기

```
import altair as alt

# load a simple dataset as a pandas DataFrame
from vega_datasets import data
cars = data.cars()

alt.Chart(cars).mark_point().encode(
    x='Horsepower',
    y='Miles_per_Gallon',
    color='Origin',
).interactive()
```



D3의 경우는?

- 훨씬 복잡하고, 알아야할 개념들도 많음
- 구현에 걸리는 시간도 오래걸림
- 새로운 시각화를 밑바닥부터 구현할 것이 아니라면 굳이 d3을 사용할 필요는 없음

```
d3.csv("icu_data.csv", function(data) {
    data.forEach(function(d) {
        d.Prob_Mortality = +d.Prob_Mortality;
        d.icustay_id = +d.icustay_id;
        d.age = +d.age;
        d.los_icu = +d.los_icu;
        d.hospital_expire_flag = +d.hospital_expire_flag;
        d.first_careunit = d.first_careunit;
    });

    var xMax = d3.max(data, function(d) {
            return d[xCat];
        }) * 1.05,
        xMin = d3.min(data, function(d) {
            return d[xCat];
        }),
        xMin = xMin > 0 ? 0 : xMin,
        yMax = d3.max(data, function(d) {
            return d[yCat];
        }) * 1.05,
        yMin = d3.min(data, function(d) {
            return d[yCat];
        }),
        yMin = yMin > 0 ? 0 : yMin;
    x.domain([xMin, xMax]);
    y.domain([yMin, yMax]);
```

시작하기 전에

- 이론 슬라이드를 통해
- Mark / Channel을 복습합시다!!

hci lab
HUMAN × COMPUTER INTERACTION

Altair 실습 1

Basic Statistical Visualization

Data

- Altair의 대부분의 경우 Data는 Pandas data frame을 사용
  - Named data column 또는 data field로 구성
- 필요하다면 URL로부터 Data를 로드하는 것도 가능!
- 실습에서는 vega-datasets repo의 dataset을 사용할 예정

```
from vega_datasets import data
data.list_datasets()
```

- 따라해 봅시다

```
from vega_datasets import data  # import vega_datasets
cars = data.cars()              # load cars data as a Pandas data frame
cars.head()                     # display the first five rows
```

Data

- 아래와 같이 Data가 정상적으로 출력되셨나요?

	Acceleration	Cylinders	Displacement	Horsepower	Miles_per_Gallon	Name	Origin	Weight_in_lbs	Year
0	12.0	8	307.0	130.0	18.0	chevrolet chevelle malibu	USA	3504	1970-01-01
1	11.5	8	350.0	165.0	15.0	buick skylark 320	USA	3693	1970-01-01
2	11.0	8	318.0	150.0	18.0	plymouth satellite	USA	3436	1970-01-01
3	12.0	8	304.0	150.0	16.0	amc rebel sst	USA	3433	1970-01-01
4	10.5	8	302.0	140.0	17.0	ford torino	USA	3449	1970-01-01

- 이번에는 url로 Data를 로드해 봅시다.

```
from vega_datasets import data  # import vega_datasets
data.cars.url
pd.read_json(data.cars.url).head() # load JSON data into a data frame
```

tidy dataframe

- Tidy dataframe?
  - Data tidying을 거쳐서 얻어진 data
  - 구조를 가지고 있으며 manipulation, model, and visualization등 여러 작업을 수행하기가 쉽다.
  - 각각의 variable은 개별 column
  - 각각의 observation (or case)는 row로 구성된다.
  - Reference (tidy data)
  - Reference (tidy data 변환)
- 따라해 봅시다

```
df = pd.DataFrame({
    'city': ['Seattle', 'Seattle', 'Seattle', 'New York', 'New York', 'New York', 'Chicago', 'Chicago', 'Chicago'],
    'month': ['Apr', 'Aug', 'Dec', 'Apr', 'Aug', 'Dec', 'Apr', 'Aug', 'Dec'],
    'precip': [2.68, 0.87, 5.31, 3.94, 4.13, 3.58, 3.62, 3.98, 2.56]
})

df
```

Chart object

- Chart object는 Altair에서 가장 기본이 되는 object입니다.

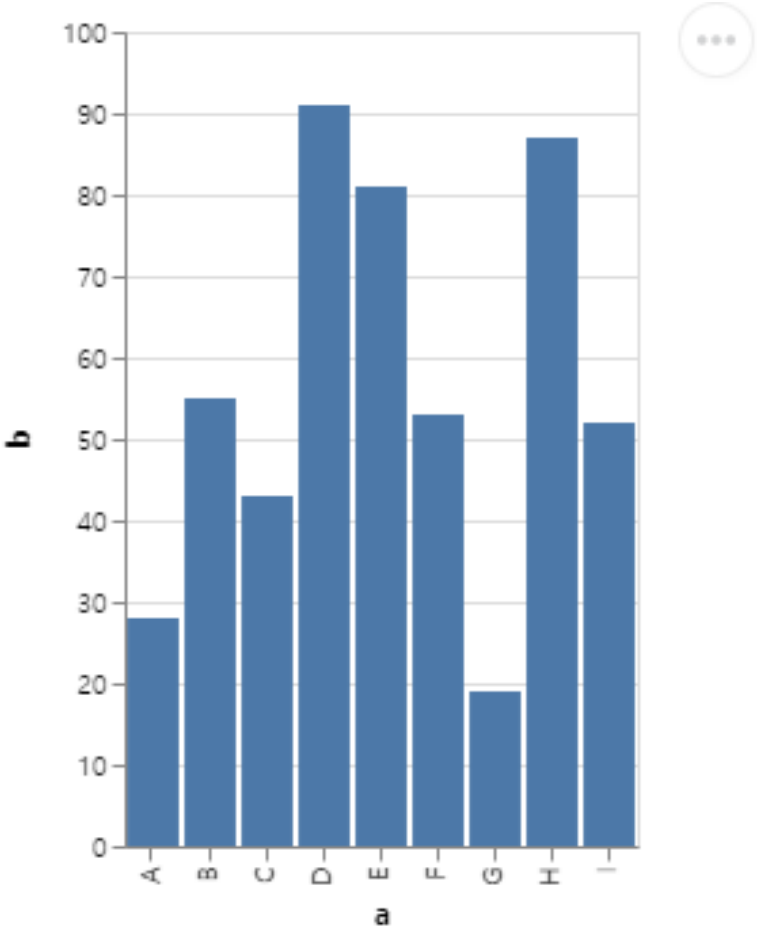
- Data frame을 single argument로 받을 수 있게 해줍니다.

```
chart = alt.Chart(df)
```

- Chart 말고도 다른 object들도 있지만, 그건 나중에...

- 이제 chart object를 사용하여 간단한 시각화를 해봅시다.

Simple bar chart




```
source = pd.DataFrame({
    'a': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I'],
    'b': [28, 55, 43, 91, 81, 53, 19, 87, 52]
})

alt.Chart(source).mark_bar().encode(
    x='a',
    y='b'
)
```

Simple bar chart

```
source = pd.DataFrame({
    'a': ['A', 'B', 'C', 'D',
          'E', 'F', 'G', 'H', 'I'],
    'b': [28, 55, 43, 91,
          81, 53, 19, 87, 52]
})

alt.Chart(source).mark_bar().encode(
    x='a',
    y='b'
)
```

```
source2 = alt.Data(
    values=[{'a': 'A', 'b': 28},
            {'a': 'B', 'b': 55},
            {'a': 'C', 'b': 43},
            {'a': 'D', 'b': 91},
            {'a': 'E', 'b': 81},
            {'a': 'F', 'b': 53},
            {'a': 'G', 'b': 19},
            {'a': 'H', 'b': 87},
            {'a': 'I', 'b': 52}])

alt.Chart(source2).mark_bar().encode(
    x='a:O',
    y='b:Q'
)
```

Data type

- Data를 dataframe으로 주는 경우
  - Encoding이 간단하다면 Altair는 자동으로 필요한 Data type을 결정해줍니다.
  - Pandas에서 data type 정보를 제공해주기 때문에 가능

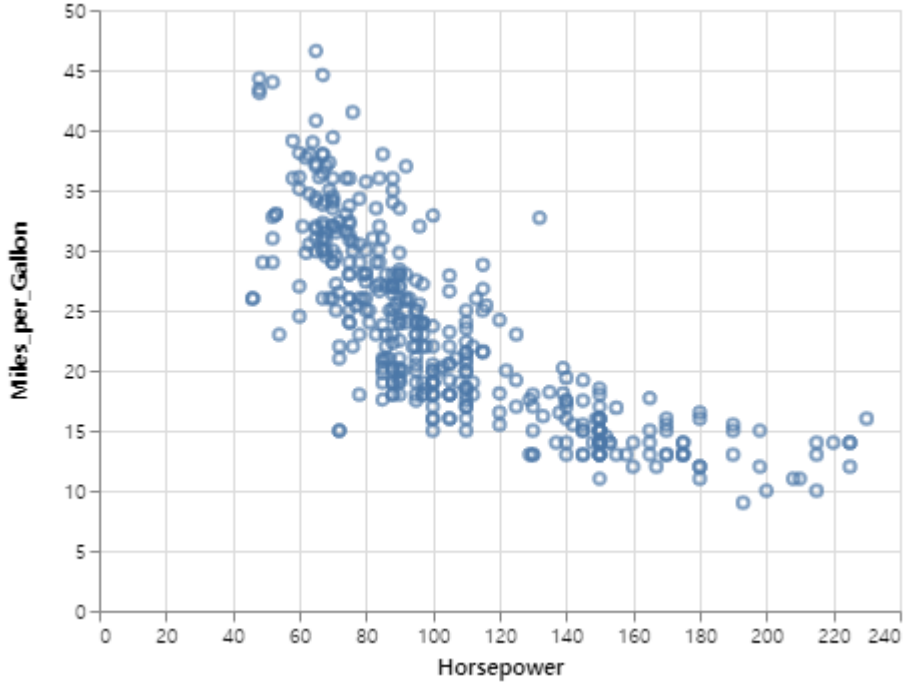
- Data를 dataframe을 사용하지 않고 주는 경우
  - Data type 정보가 없기 때문에 사용자가 직접 data type을 제공해줘야 합니다.
  - Encoding 파트에서 보다 자세히 다룰 예정

Data Type	Shorthand Code	Description
quantitative	Q	a continuous real-valued quantity
ordinal	O	a discrete ordered quantity
nominal	N	a discrete unordered category
temporal	T	a time or date value

Scatterplot

```
from vega_datasets import data

url = data.cars.url
alt.Chart(url).mark_point().encode(
    x='Horsepower:Q',
    y='Miles_per_Gallon:Q'
)
```

```
from vega_datasets import data

cars = data.cars()
alt.Chart(cars).mark_point().encode(
    x='Horsepower:Q',
    y='Miles_per_Gallon:Q'
)
```



Long-form vs Wide-form data

- Wide-form data
  - One row per independent variable
  - Metadata는 row에 기록
  - Label은 column에 기록된다
- Long-form data
  - One row per observation
  - Metadata는 테이블에 value로 기록된다.
- Altair는 long-form data에서 잘 동작한다.

Long-form vs Wide-form data

- Wide-form data

```
wide_form = pd.DataFrame({'Date': ['2007-10-01', '2007-11-01', '2007-12-01'],
                          'AAPL': [189.95, 182.22, 198.08],
                          'AMZN': [89.15, 90.56, 92.64],
                          'GOOG': [707.00, 693.00, 691.48]})
print(wide_form)
```

- Long-form data

```
long_form = pd.DataFrame({'Date': ['2007-10-01', '2007-11-01', '2007-12-01',
                                   '2007-10-01', '2007-11-01', '2007-12-01',
                                   '2007-10-01', '2007-11-01', '2007-12-01'],
                          'company': ['AAPL', 'AAPL', 'AAPL',
                                      'AMZN', 'AMZN', 'AMZN',
                                      'GOOG', 'GOOG', 'GOOG'],
                          'price': [189.95, 182.22, 198.08,
                                    89.15,  90.56,  92.64,
                                   707.00, 693.00, 691.48]})
print(long_form)
```

Long-form vs Wide-form data

- Wide-form data – One row per independent variable

	Date	AAPL	AMZN	GOOG
0	2007-10-01	189.95	89.15	707.00
1	2007-11-01	182.22	90.56	693.00
2	2007-12-01	198.08	92.64	691.48

- Long-form data – One row per observation

	Date	company	price
0	2007-10-01	AAPL	189.95
1	2007-11-01	AAPL	182.22
2	2007-12-01	AAPL	198.08
3	2007-10-01	AMZN	89.15
4	2007-11-01	AMZN	90.56
5	2007-12-01	AMZN	92.64
6	2007-10-01	GOOG	707.00
7	2007-11-01	GOOG	693.00
8	2007-12-01	GOOG	691.48

Name	Height	Weight
John	160	67
Christopher	182	78

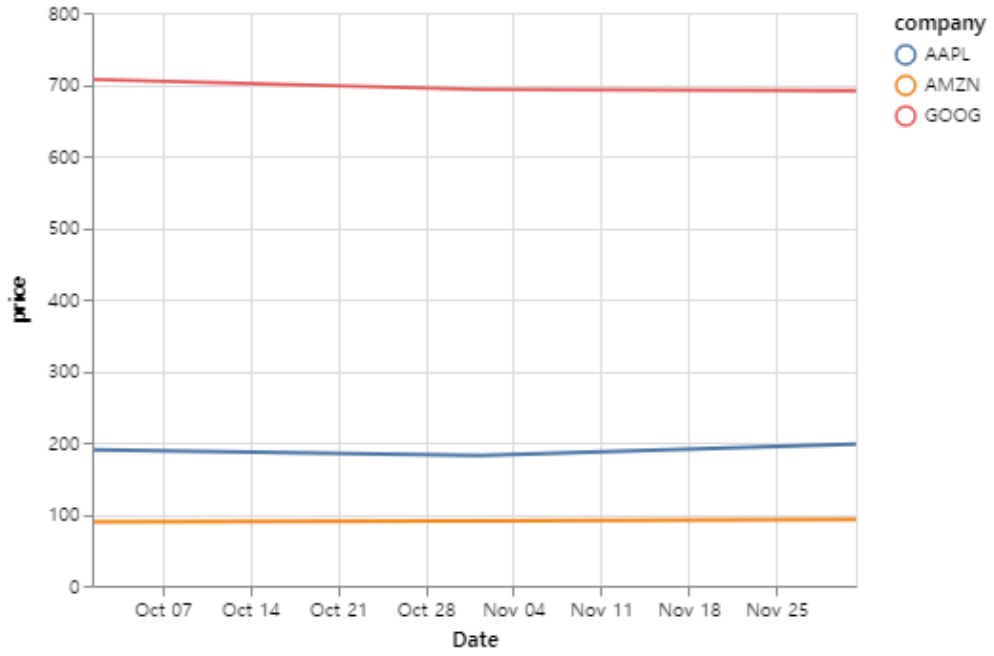
Name	Attribute	Value
John	Height	160
John	Weight	67
Christopher	Height	182
Christopher	Weight	78

Long-form vs Wide-form data

- Altair는 long-form data에서 잘 동작한다.
  - 관련된 data와 metadata들이 table 안에 있기 때문에 시각화를 하기에 용이함
  - Wide-form data에서는 이러한 정보들이 row와 column의 label에 있기 때문에 불편

- Converting between long-form and wide-form
  - Altair에서는 변환기능을 제공하고 있지 않다.
  - 따라서 반드시 별도의 preprocessing step을 거쳐야 한다.
  - Pandas를 사용할 경우, wide-form to long-form은 melt를 사용하여
  - Long-form to wide-form은 pivot을 사용하여 변환할 수 있다.
  - 자세한 사항은 Pandas의 Reshaping and Pivot Table을 참고하자

Line chart




```
long_form = pd.DataFrame({'Date': ['2007-10-01', '2007-11-01', '2007-12-01',
                                   '2007-10-01', '2007-11-01', '2007-12-01',
                                   '2007-10-01', '2007-11-01', '2007-12-01'],
                          'company': ['AAPL', 'AAPL', 'AAPL',
                                      'AMZN', 'AMZN', 'AMZN',
                                      'GOOG', 'GOOG', 'GOOG'],
                          'price': [189.95, 182.22, 198.08,
                                    89.15,  90.56,  92.64,
                                    707.00, 693.00, 691.48]})
print(long_form)
```

```
alt.Chart(long_form).mark_line().encode(
  x='Date:T',
  y='price:Q',
  color='company:N'
)
```

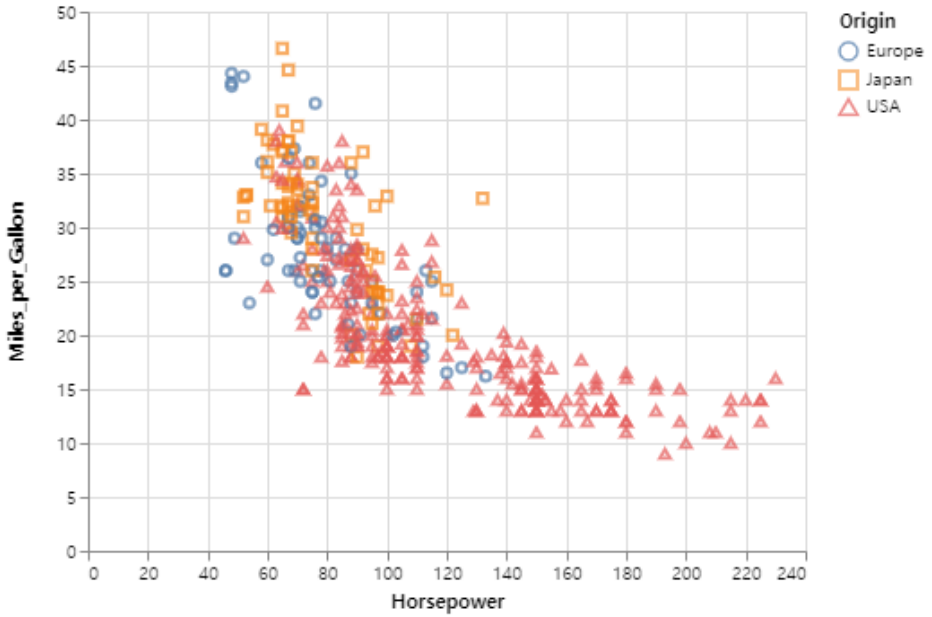
hci lab
HUMAN × COMPUTER INTERACTION

Altair 실습 1

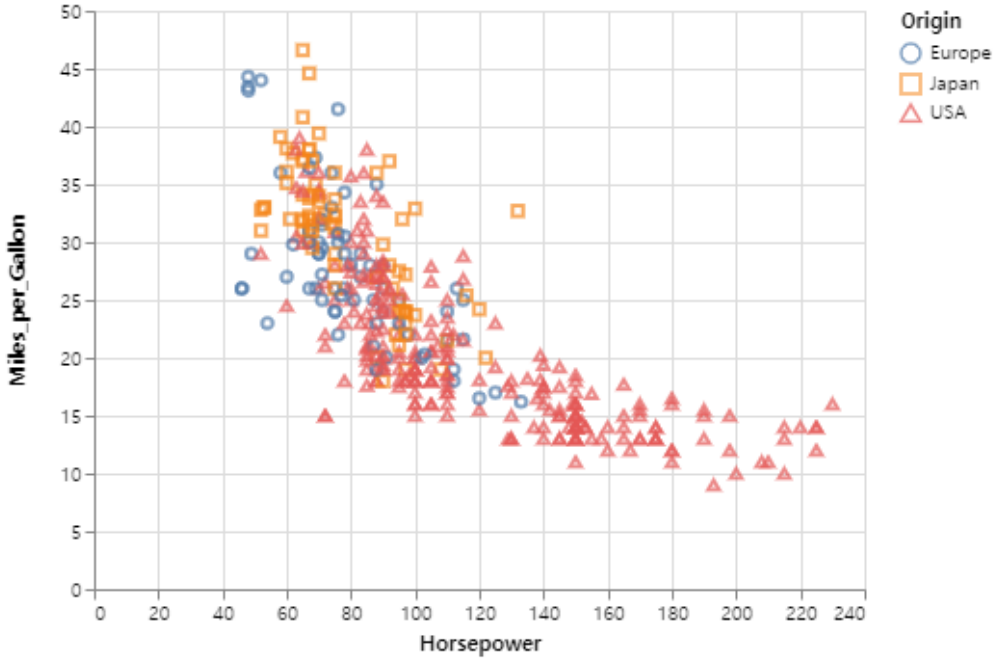
Encodings

Encodings

- 시각화를 통해 효과적으로 정보를 전달하기 위해서는 데이터 속성을 올바르게 시각적 속성에 매핑하는 것이 중요
- Altair에서는 시각적 속성들이 데이터 column에 매핑된 것을 encoding이라 한다.
- 아래의 시각화에서는 몇개의 Encoding을 사용하고 있을까?



Encodings




```
from vega_datasets import data
cars = data.cars()

alt.Chart(cars).mark_point().encode(
    x='Horsepower',
    y='Miles_per_Gallon',
    color='Origin',
    shape='Origin'
)
```

Encoding Data Types

Data Type	Shorthand Code	Description
quantitative	Q	a continuous real-valued quantity
ordinal	O	a discrete ordered quantity
nominal	N	a discrete unordered category
temporal	T	a time or date value

- 만약 Dataframe에 의해 data의 type이 정해지지 않는다면
- Numeric data는 quantitative
- Date/time data는 temporal
- String data는 nomial로 인식한다.

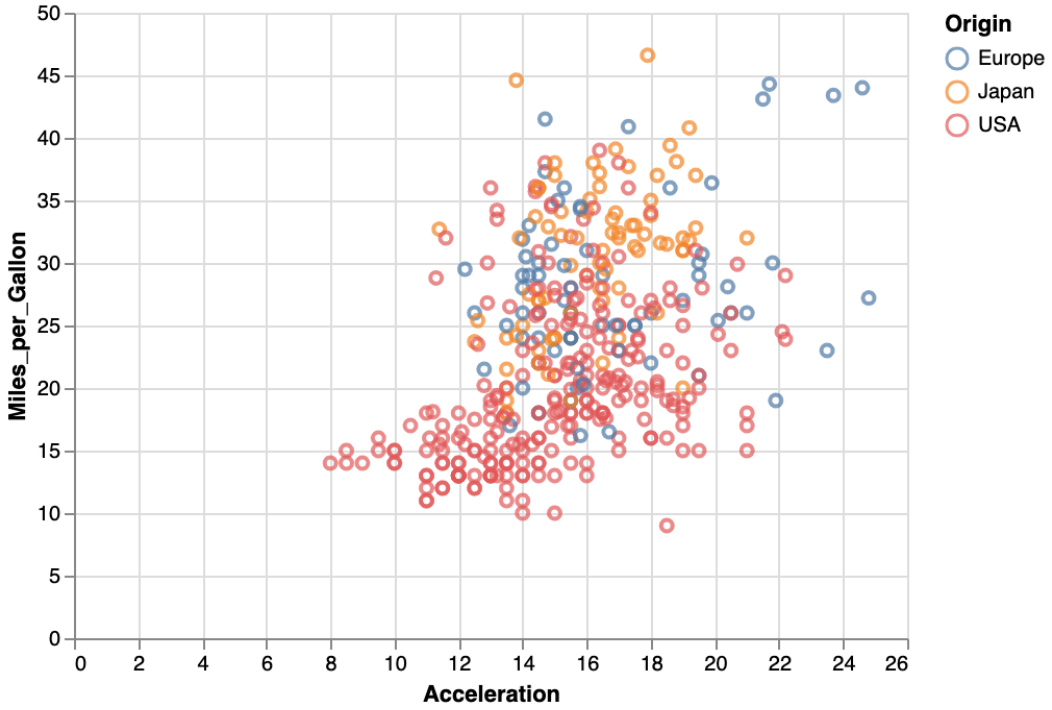
Encoding Data Types

- Short-form
  - 빠른 데이터 탐색에 용이

```
alt.Chart(cars).mark_point().encode(
    x='Acceleration:Q',
    y='Miles_per_Gallon:Q',
    color='Origin:N'
)
```

- Long-form
  - 상세한 encoding 수정을 할때 용이

```
alt.Chart(cars).mark_point().encode(
    alt.X('Acceleration', type='quantitative'),
    alt.Y('Miles_per_Gallon', type='quantitative'),
    alt.Color('Origin', type='nominal')
)
```



Encoding Channels

- Altair에서는 다양한 Encoding channel들을 제공하고 있다.
- 실습을 진행하기에 앞서 데이터를 로드해보자 : 국가별 경제 / 의료 수준 데이터

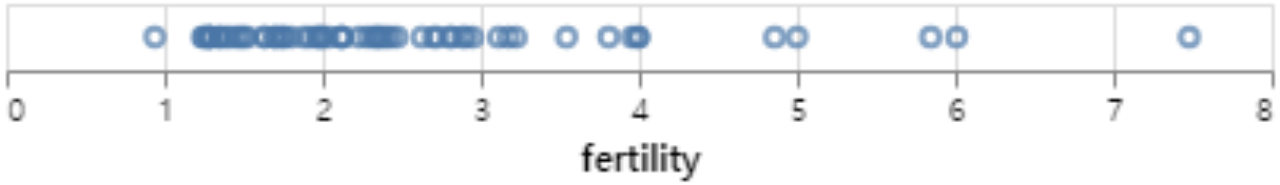
```
gapminder = data.gapminder()
data2000 = gapminder.loc[gapminder['year'] == 2000]
data2000.head(5)
```

	cluster	country	fertility	life_expect	pop	year
9	0	Afghanistan	7.4792	42.129	23898198	2000
20	3	Argentina	2.3500	74.340	37497728	2000
31	3	Aruba	2.1240	73.451	69539	2000
42	4	Australia	1.7560	80.370	19164620	2000
53	1	Austria	1.3820	78.980	8113413	2000

Encoding Channels – X, Y

- X encoding channel
  - 기본적으로 축을 나누는 것과 title은 자동으로 결정됩니다.

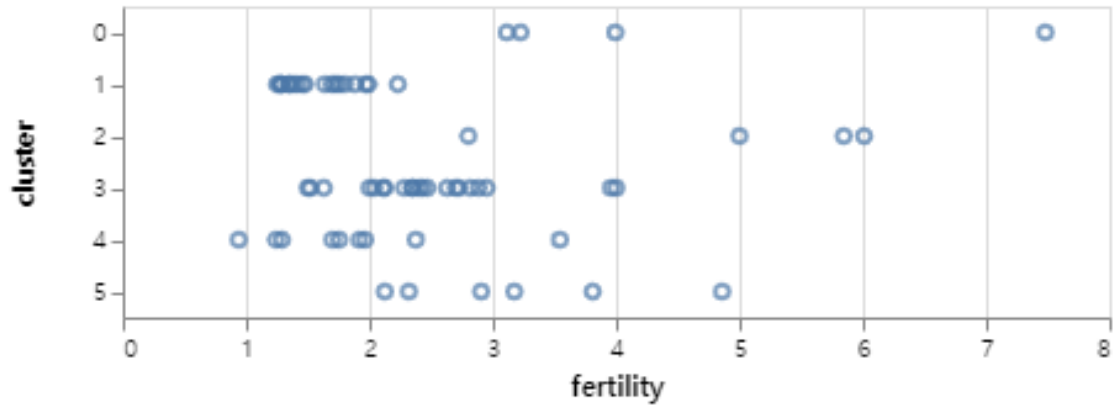
```
alt.Chart(data2000).mark_point().encode(
    alt.X('fertility:Q')
)
```



Encoding Channels – X, Y

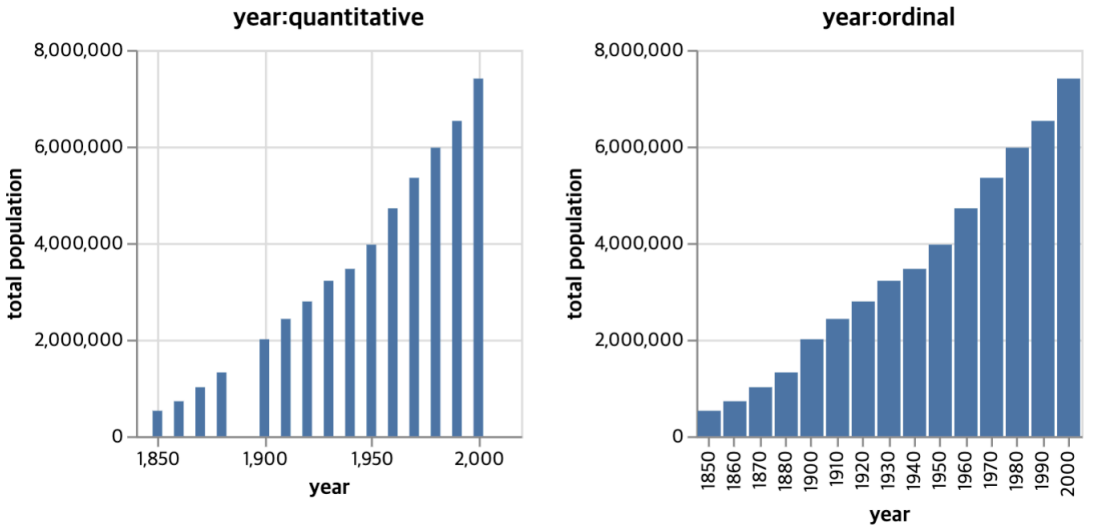
- Y encoding channel
  - Y축을 Ordinal data (cluster)에, X축을 Quantitative data (fertility)에 매핑

```
alt.Chart(data2000).mark_point().encode(
    alt.X('fertility:Q'),
    alt.Y('cluster:O')
)
```



Encoding Channels – X, Y

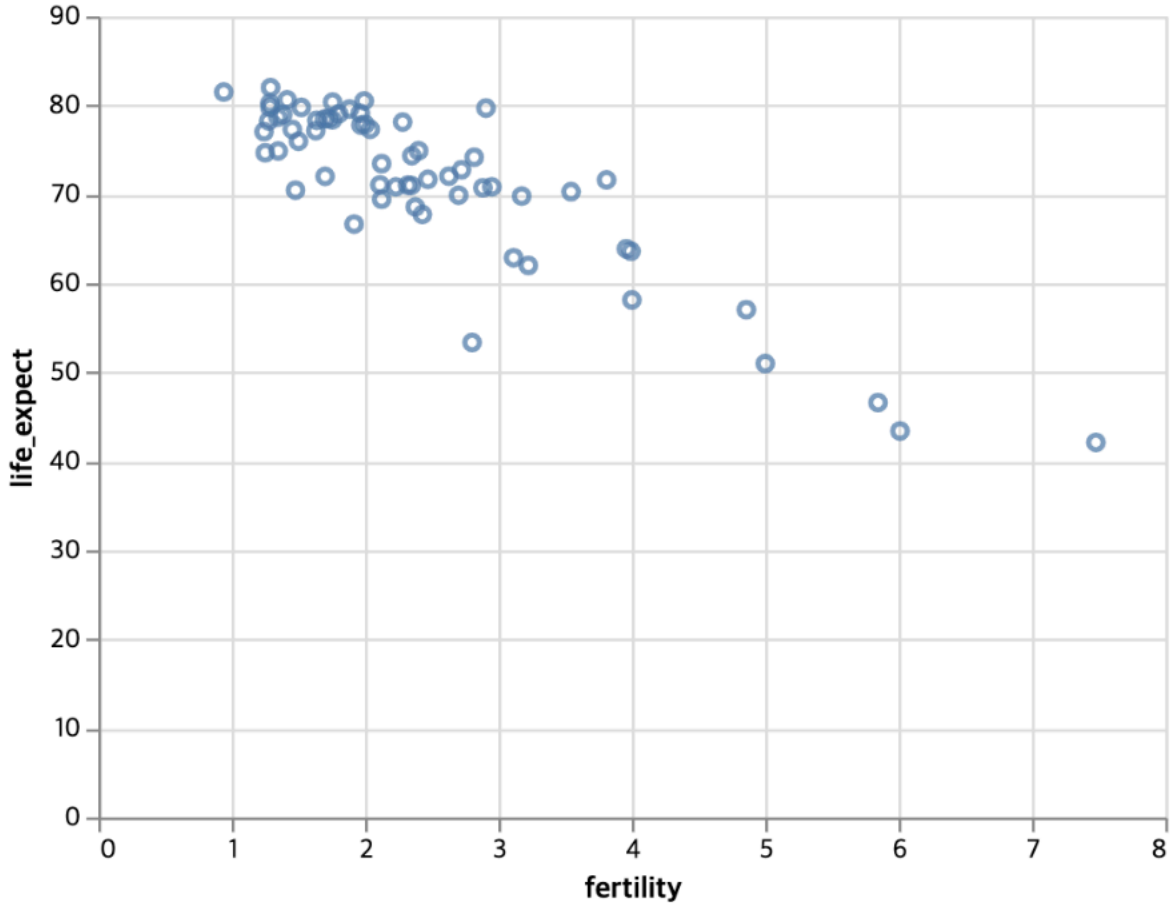
```
from vega_datasets import data
pop = data.population() # data.population.url

base = alt.Chart(pop).mark_bar().encode(
    alt.Y("mean(people):Q", title="total population")
).properties(
    width=200,
    height=200
)

alt.hconcat(
  base.encode(x='year:Q').properties(title='year:quantitative'),
  base.encode(x='year:O').properties(title='year:ordinal')
)
```



Encoding Channels – X, Y

```
alt.Chart(data2000).mark_point().encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q')
)
```

출산율 / 기대수명과의 관계?

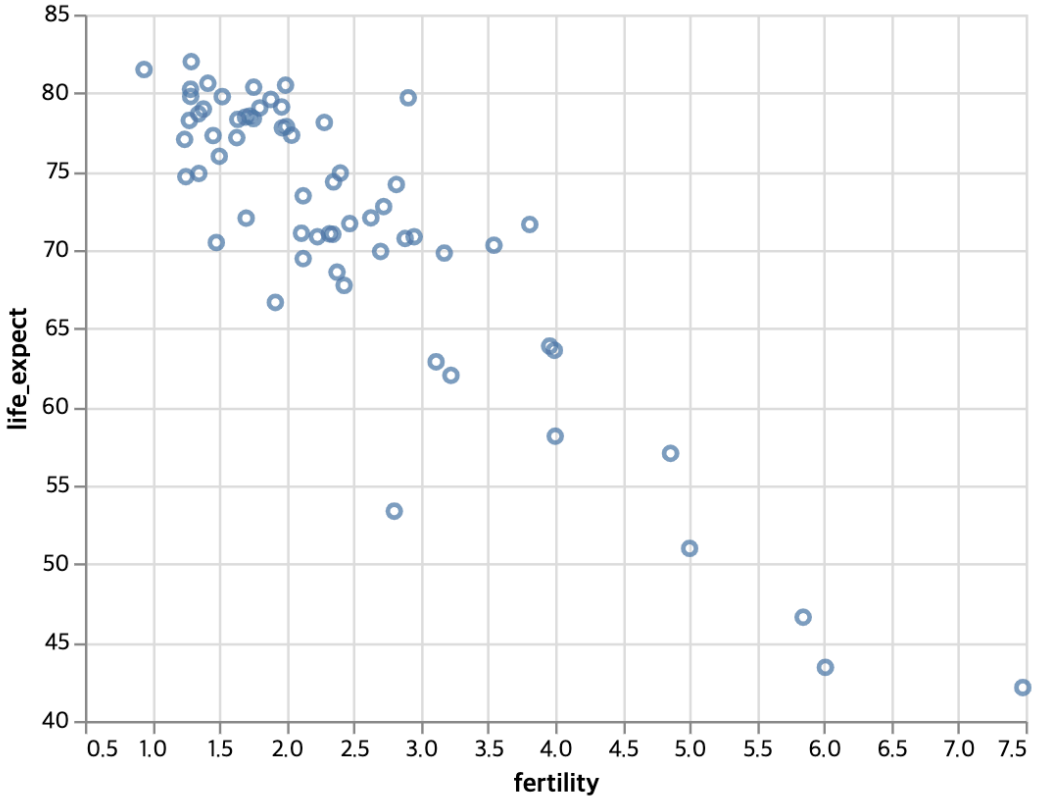


Encoding Channels – X, Y

- Zero baseline이 필요없을 경우
  - Axis의 end point는 자동으로 5와 10의 배수인 nice number로 맞춰진다.

```
alt.Chart(data2000).mark_point().encode(
    alt.X("fertility:Q", scale=alt.Scale(zero=False)),
    alt.Y('life_expect:Q', scale=alt.Scale(zero=False)),
)
```

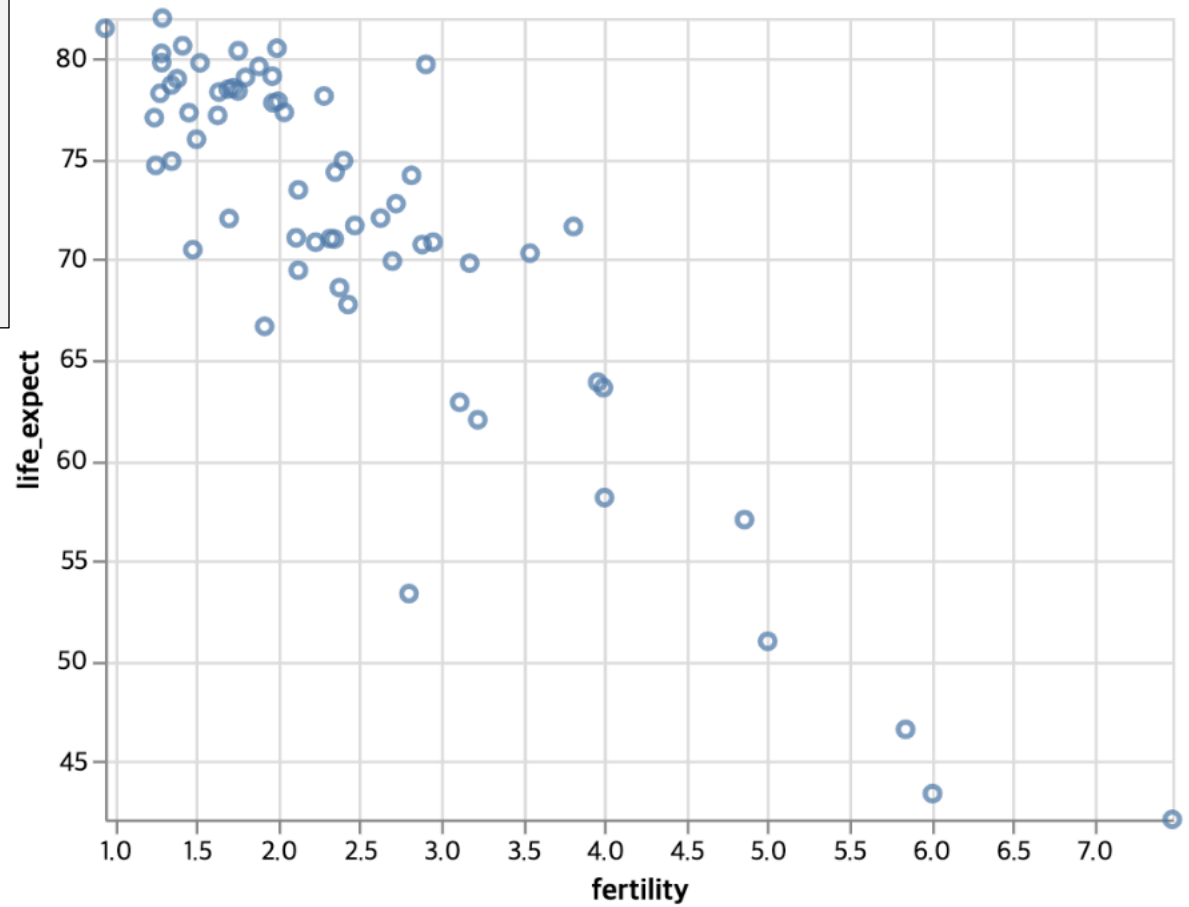
출산율 / 기대수명과의
관계가 더 잘 드러남



Encoding Channels – X, Y

```
alt.Chart(data2000).mark_point().encode(
    alt.X("fertility:Q", scale=alt.Scale(
        zero=False, nice=False)),
    alt.Y('life_expect:Q', scale=alt.Scale(
        zero=False, nice=False))
)
```

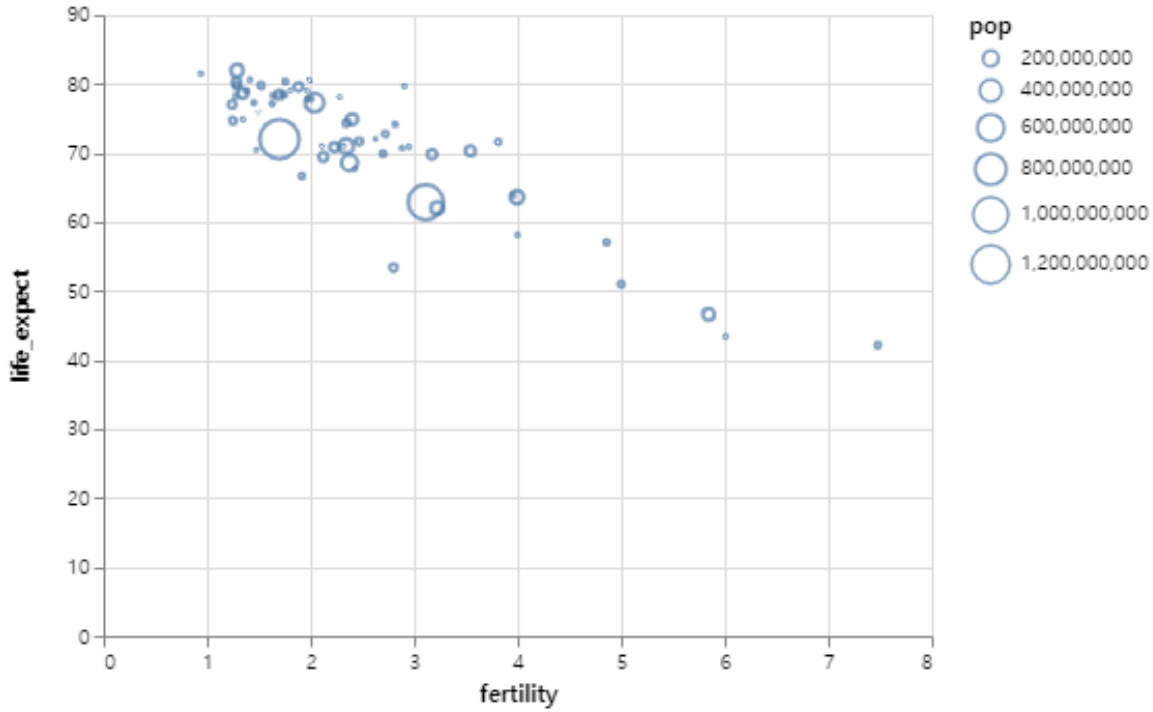
end point를 nice하지 않은 number로 맞춤



Encoding Channels - Size

- Size encoding channel
  - Size는 일반적으로 mark의 size를 의미한다.
  - Mark의 종류에 따라 size의 정의가 다를 수 있다.

```
alt.Chart(data2000).mark_point().encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q')
)
```

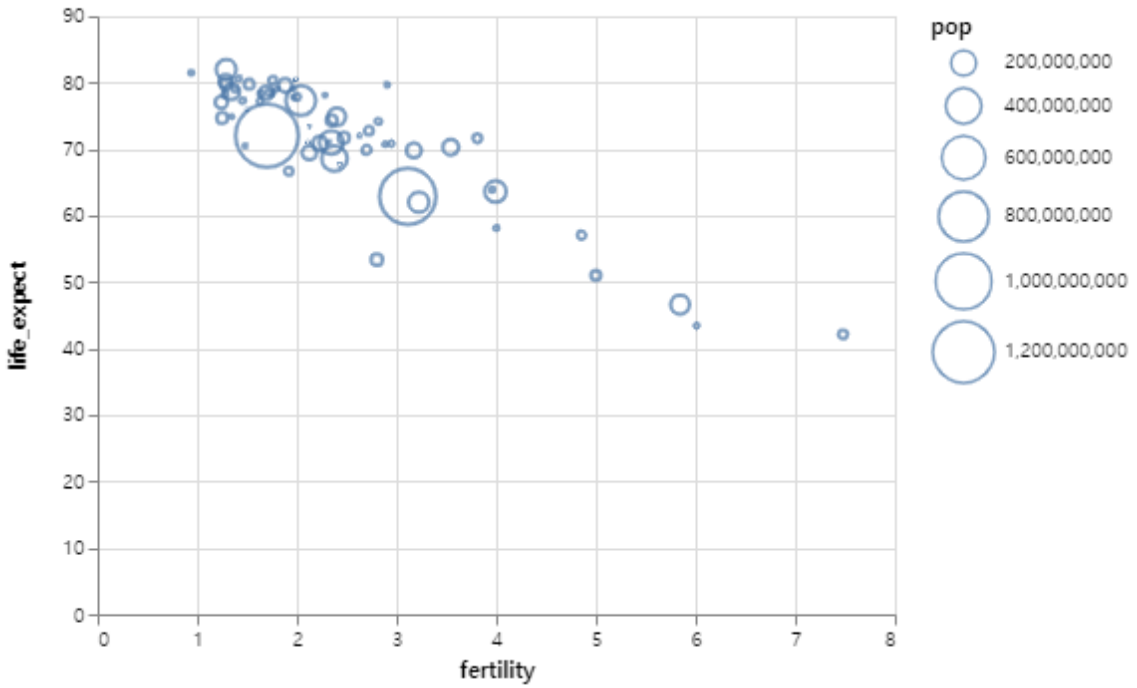


- Size range가 마음에 들지 않는다면?

Encoding Channels - Size

- Scale attribute의 range parameter를 사용하여 수정 가능!
  - 0는 zero value에, 1000은 가장 data값에 매핑된다.

```
alt.Chart(data2000).mark_point().encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q',
             scale=alt.Scale(range=[0, 1000]))
)
```



Encoding Channels – Color and Opacity

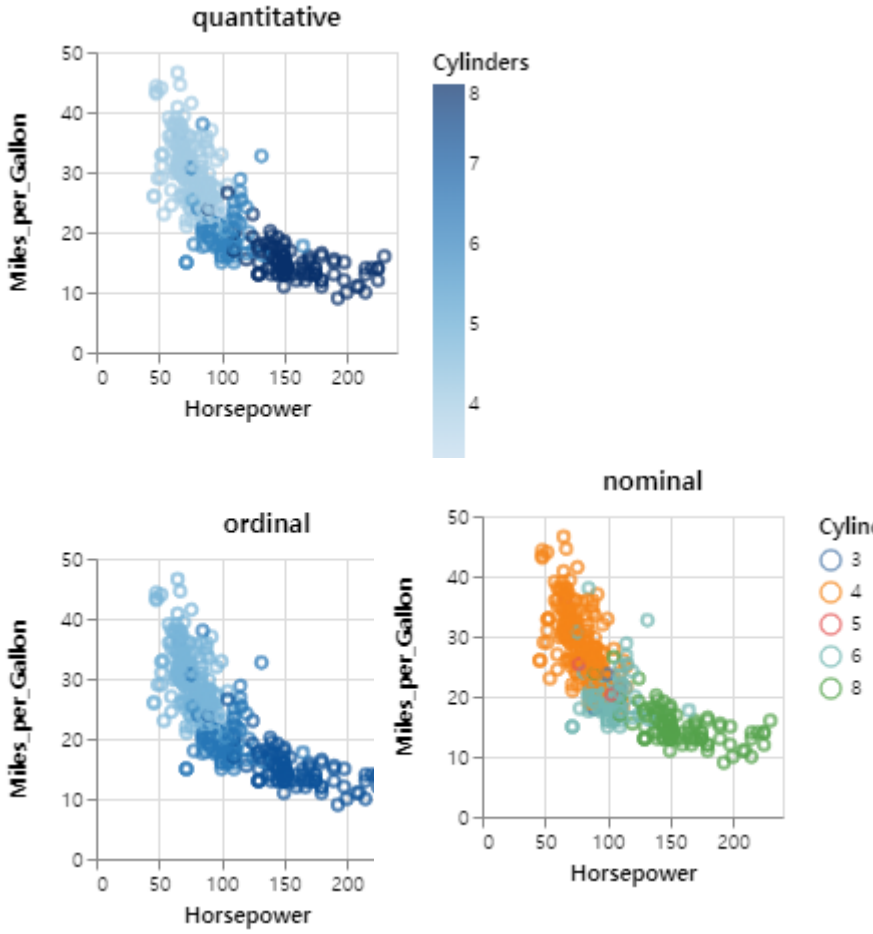
- Color encoding channel
  - Color는 mark의 color를 결정한다.
  - Color를 설정하는 법은 data type에 따라 달라진다.

- Nominal data의 경우
  - 시각적으로 잘 구분되기만 하면 됨

- Ordinal and quantitative data의 경우
  - Order가 드러나야 함
  - Ordered color gradients 사용

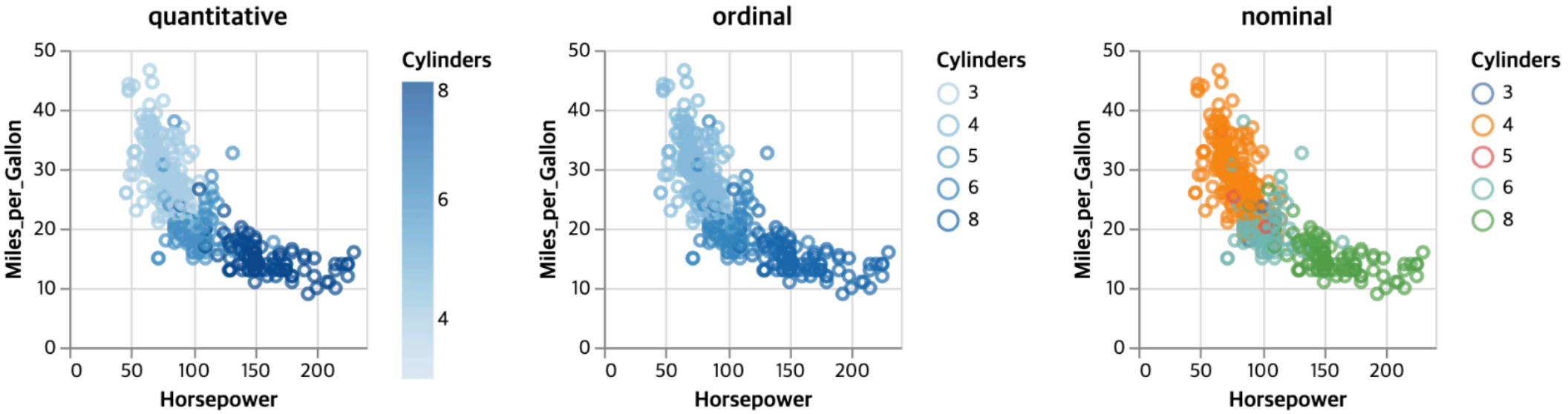
Encoding Channels – Color and Opacity

```
base=alt.Chart(cars).mark_point().encode(
    x='Horsepower:Q',
    y='Miles_per_Gallon:Q'
).properties(
    width=150,
    height=150
)

alt.vconcat(
    base.encode(color='Cylinders:Q').properties(title='quantitative'),
    base.encode(color='Cylinders:O').properties(title='ordinal'),
    base.encode(color='Cylinders:N').properties(title='nominal')

)
```



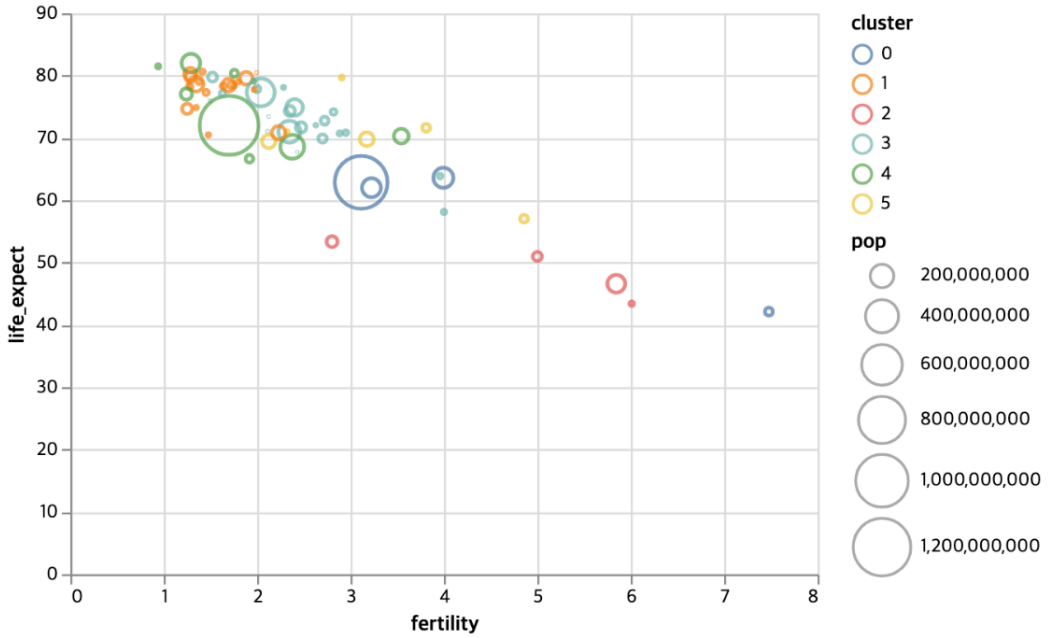
Encoding Channels – Color and Opacity

```
alt.hconcat(
    base.encode(color='Cylinders:Q').properties(title='quantitative'),
    base.encode(color='Cylinders:O').properties(title='ordinal'),
    base.encode(color='Cylinders:N').properties(title='nominal')

)
```



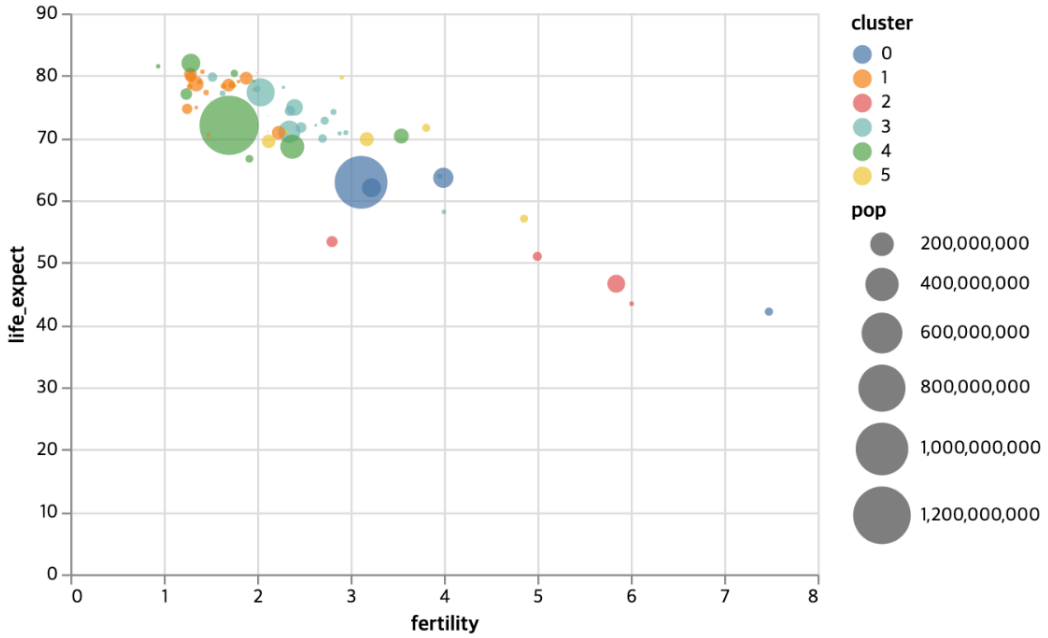
Encoding Channels – Color and Opacity

```
alt.Chart(data2000).mark_point().encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q', scale=alt.Scale(range=[0, 1000])),
    alt.Color('cluster:N'),
)
```

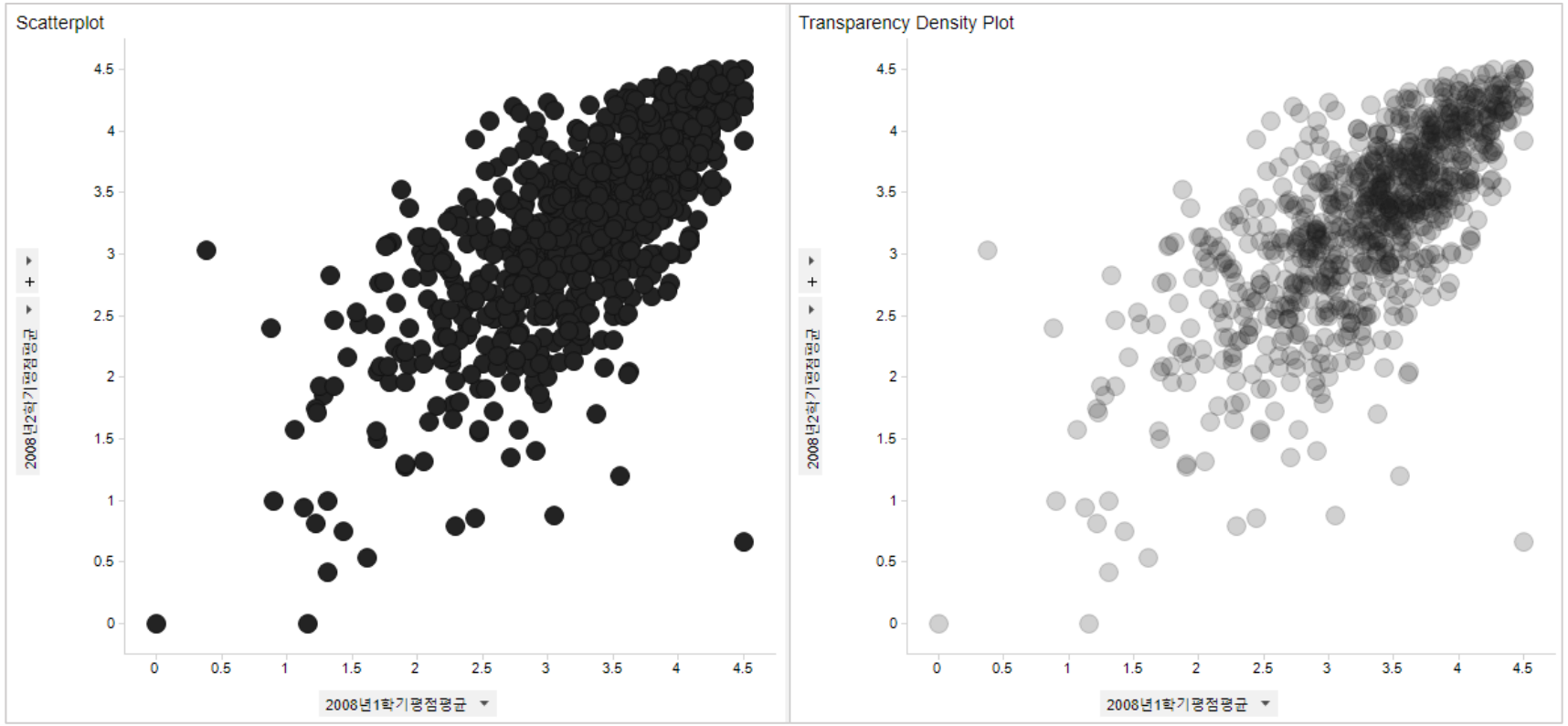


Encoding Channels – Color and Opacity

```
alt.Chart(data2000).mark_point(filled=True).encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q', scale=alt.Scale(range=[0, 1000])),
    alt.Color('cluster:N'),
)
```



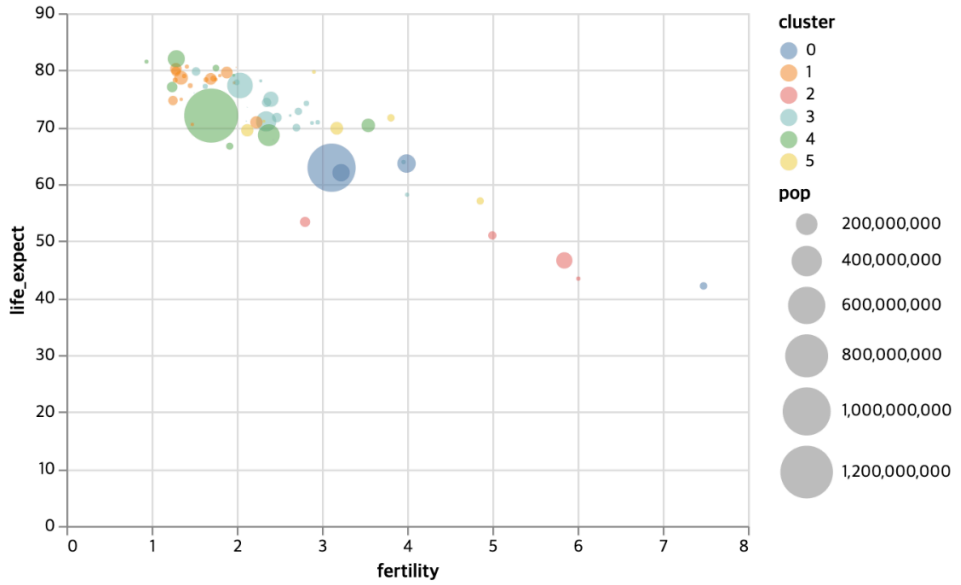
Encoding Channels – Color and Opacity



Encoding Channels – Color and Opacity

- Solve over-plotting problem with opacity

```
alt.Chart(data2000).mark_point(filled=True).encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q', scale=alt.Scale(range=[0, 1000])),
    alt.Color('cluster:N'),
    alt.OpacityValue(0.5)
)
```

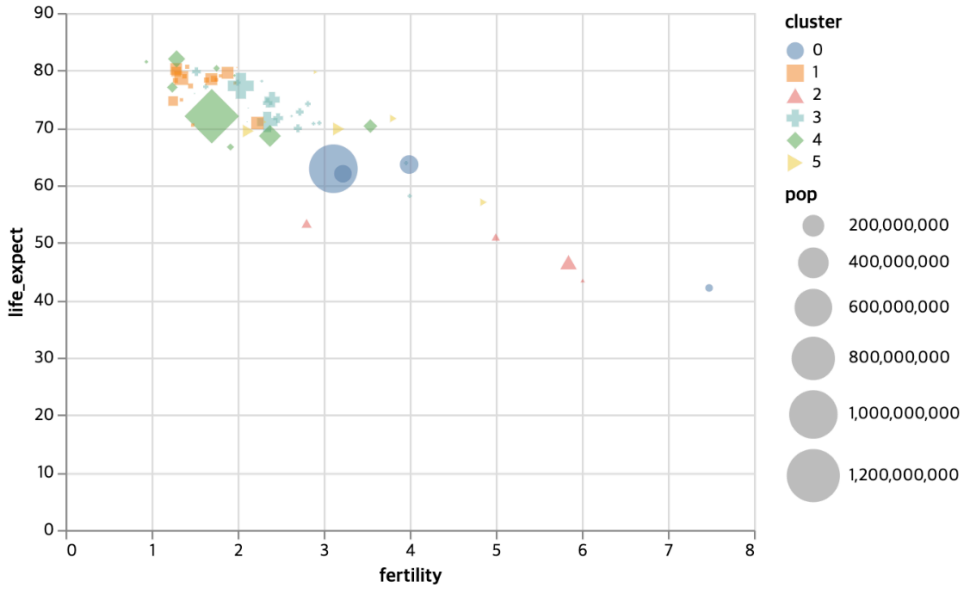


Encoding Channels – Shape

- Shape encoding channel
  - Point mark의 모양을 결정
  - Shape encoding channel은 nominal data에서만 사용될 수 있다!

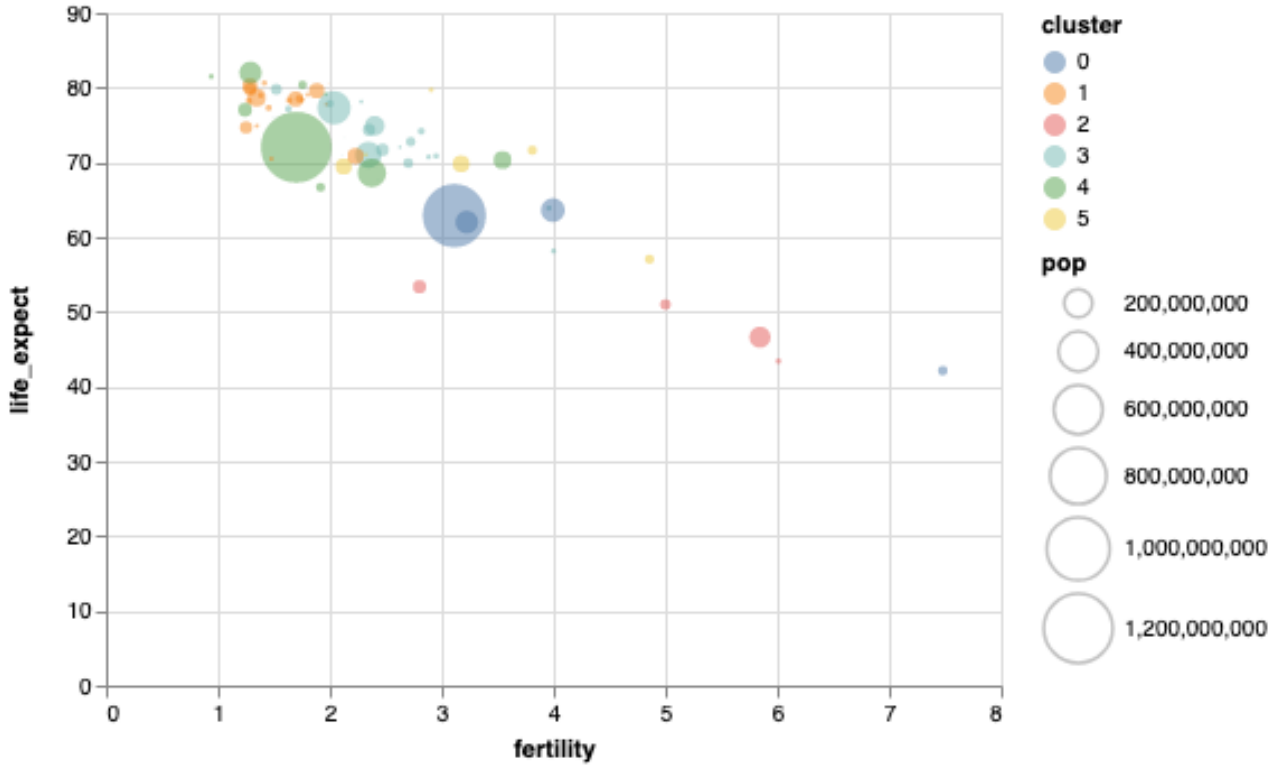
Encoding Channels – Shape

```
alt.Chart(data2000).mark_point(filled=True).encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q', scale=alt.Scale(range=[0, 1000])),
    alt.Color('cluster:N'),
    alt.OpacityValue(0.5),
    alt.Shape('cluster:N')
)
```



Encoding Channels – Tooltips

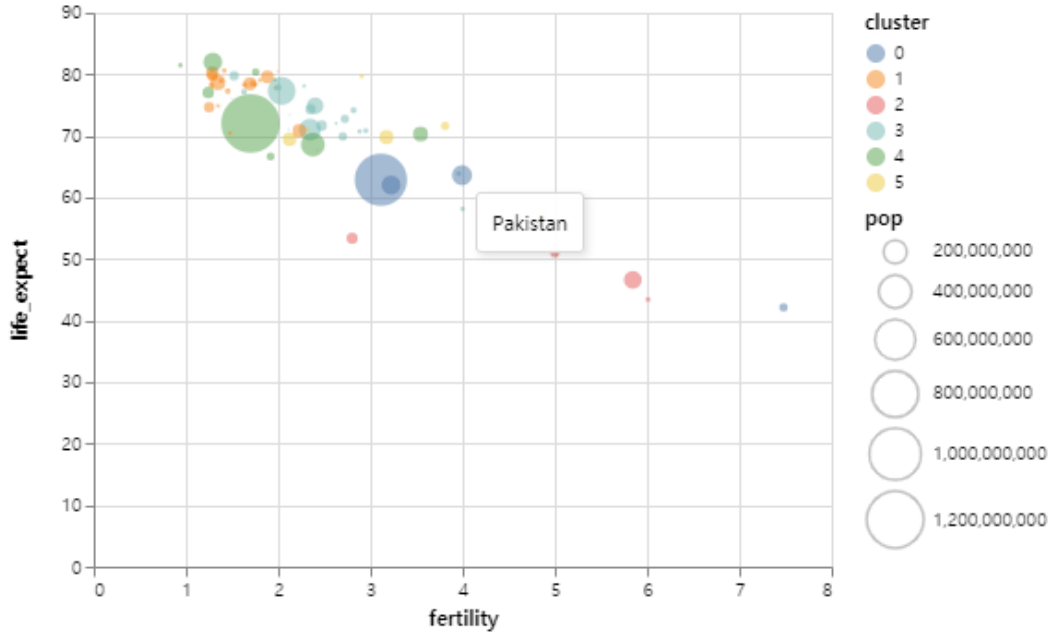
- 많은 Encoding들을 시각화에 적용했지만 아직 각각의 포인트들이 어느 나라에 대응하는지 알 수가 없다!



Encoding Channels – Tooltips

- Tooltip

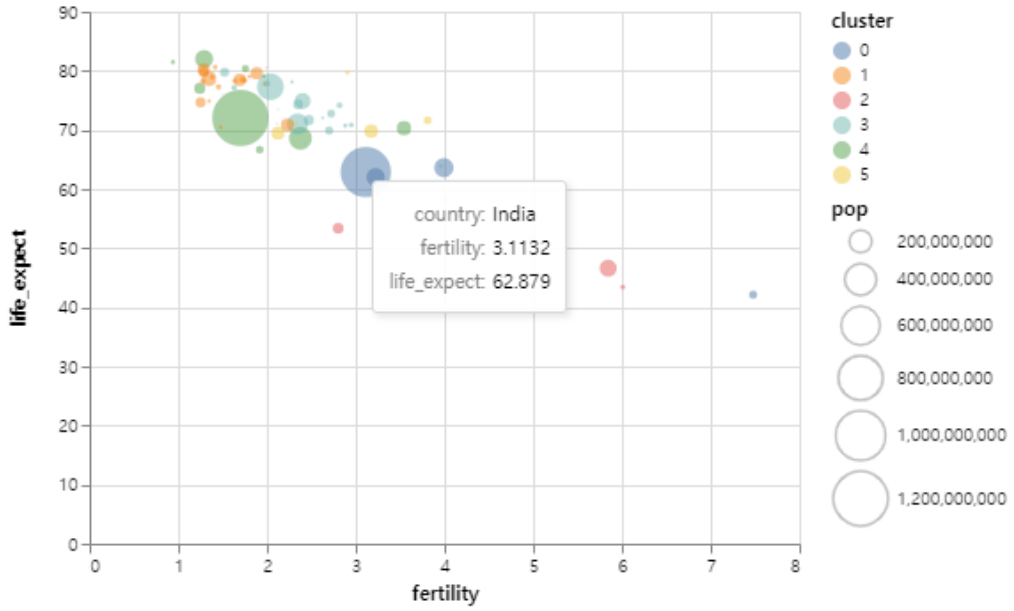
```
alt.Chart(data2000).mark_point(filled=True).encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q', scale=alt.Scale(range=[0, 1000])),
    alt.Color('cluster:N'),
    alt.OpacityValue(0.5),
    alt.Tooltip('country'),
)
```



Encoding Channels – Tooltips

- 여러개의 툴팁을 추가하는 것도 가능!

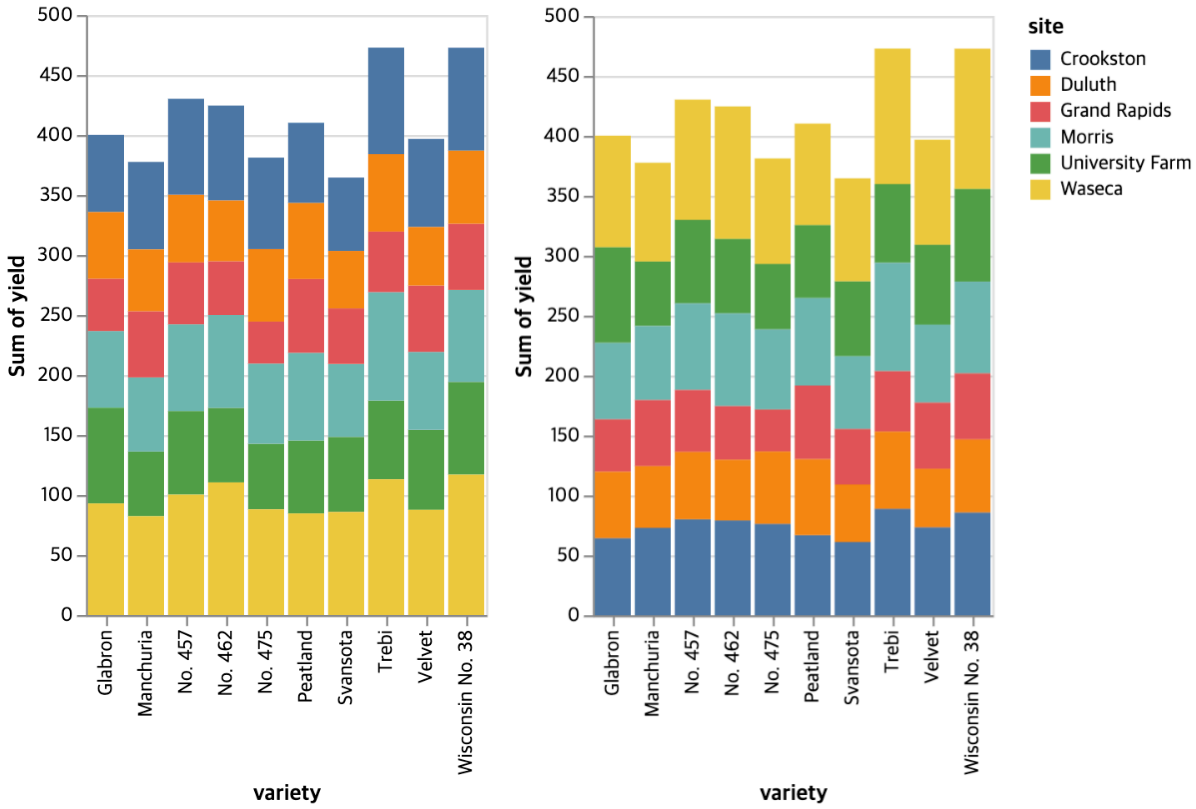
```
alt.Chart(data2000).mark_point(filled=True).encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q', scale=alt.Scale(range=[0, 1000])),
    alt.Color('cluster:N'),
    alt.OpacityValue(0.5),
    alt.Shape('cluster:N'),
    tooltip = [
        alt.Tooltip('country:N'),
        alt.Tooltip('fertility:Q'),
        alt.Tooltip('life_expect:Q')
    ]
)
```



Encoding Channels – Ordering

```
barley = data.barley()

alt.Chart(barley).mark_bar().encode(
    x='variety:N',
    y='sum(yield):Q',
    color='site:N',
    order=alt.Order('site', sort='descending')
)
```

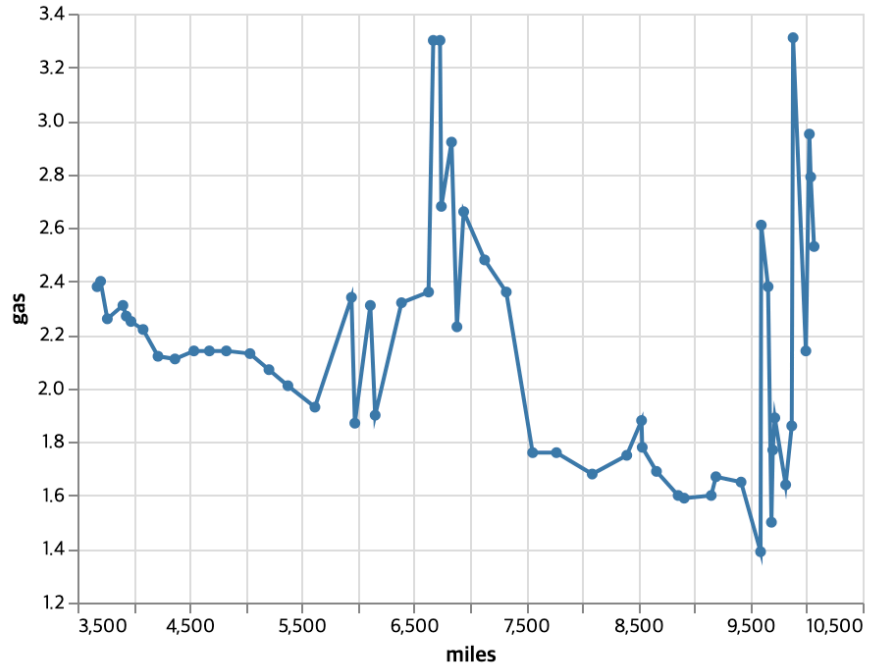
```
barley = data.barley()

alt.Chart(barley).mark_bar().encode(
    x='variety:N',
    y='sum(yield):Q',
    color='site:N',
    order=alt.Order('site', sort='ascending')
)
```



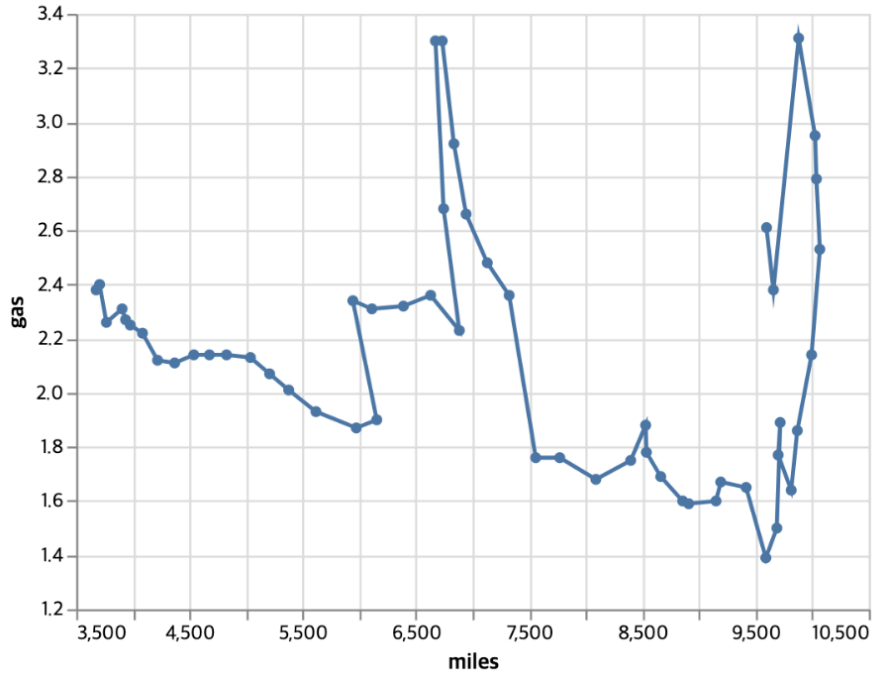
Encoding Channels – ordering

- x, y 축과 다른 field를 이용해 scatterplot을 그릴 때 유용

```
driving = data.driving()

alt.Chart(driving).mark_line(point=True).encode(
    alt.X('miles', scale=alt.Scale(zero=False)),
    alt.Y('gas', scale=alt.Scale(zero=False)),
)
```




```
alt.Chart(driving).mark_line(point=True).encode(
    alt.X('miles', scale=alt.Scale(zero=False)),
    alt.Y('gas', scale=alt.Scale(zero=False)),
    order='year'
)
```

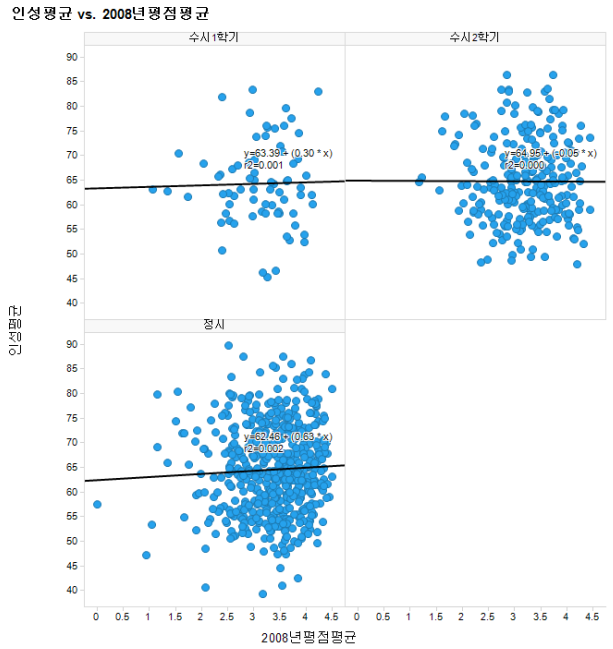


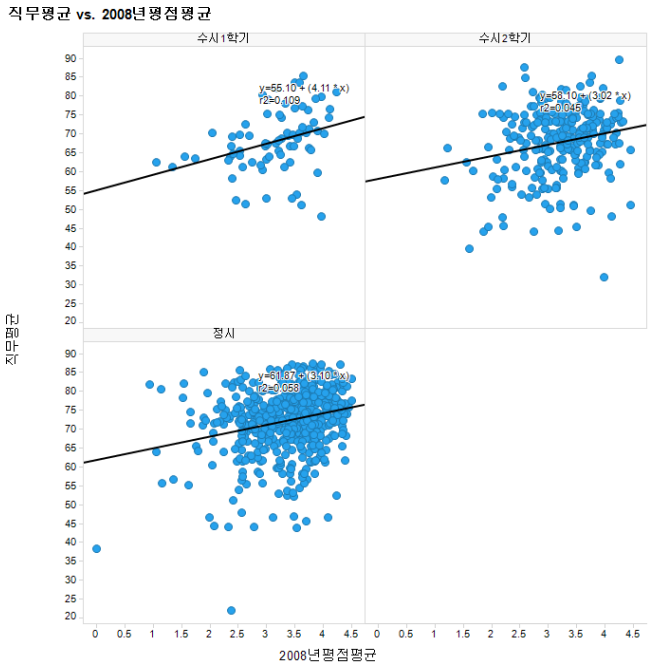
Encoding Channels – Column & Row Facets

- 공간적 위치는 강력한 visual encoding이다.
- 하지만, 이미 X, Y channel에 매핑해버렸는데….

Encoding Channels – Column & Row Facets

- 이러한 경우 trellis plot를 사용하는 것이 하나의 대안이 될 수 있다.
  - Row 또는 Column을 기준으로 여러개의 Sub plot을 생성

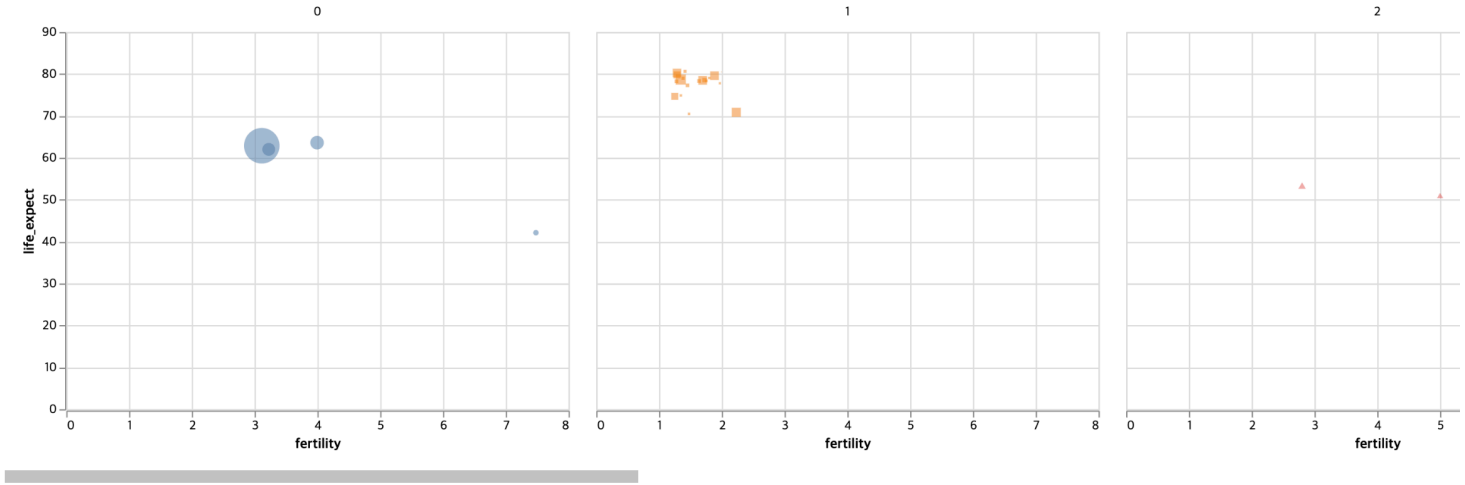




Encoding Channels – Column & Row Facets

- Plot이 화면에 맞지 않아서 subplot간의 비교가 어렵다!

```
alt.Chart(data2000).mark_point(filled=True).encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q', scale=alt.Scale(range=[0, 1000])),
    alt.Color('cluster:N'),
    alt.OpacityValue(0.5),
    alt.Shape('cluster:N'),
    alt.Order('pop:Q', sort='descending'),
    alt.Column('cluster:N'),
)
```

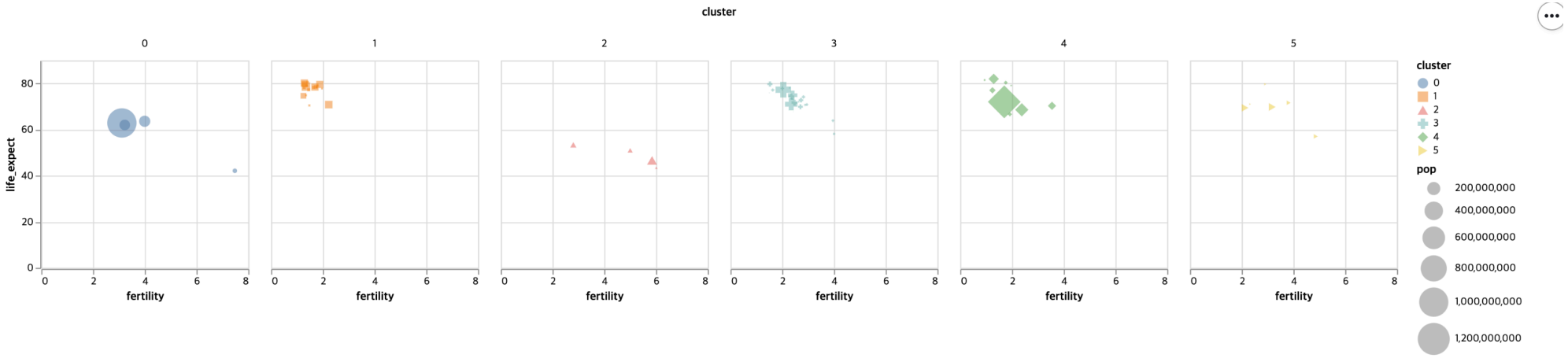


Encoding Channels – Column & Row Facets

- Width와 Height를 조절해서 Subplot을 더 보기 쉽게 만들자
- Cluster 별로 Subplot이 생성되므로 legend도 없애서 공간확보를 하자!

Encoding Channels – Column & Row Facets

```
alt.Chart(data2000).mark_point(filled=True).encode(
    alt.X("fertility:Q"),
    alt.Y('life_expect:Q'),
    alt.Size('pop:Q', scale=alt.Scale(range=[0, 1000])),
    alt.Color('cluster:N'),
    alt.OpacityValue(0.5),
    alt.Shape('cluster:N'),
    alt.Order('pop:Q', sort='descending'),
    alt.Column('cluster:N'),
).properties(width=200,height=200)
```



- Column 대신 Row를 사용하려면 어떻게 해야 할까?

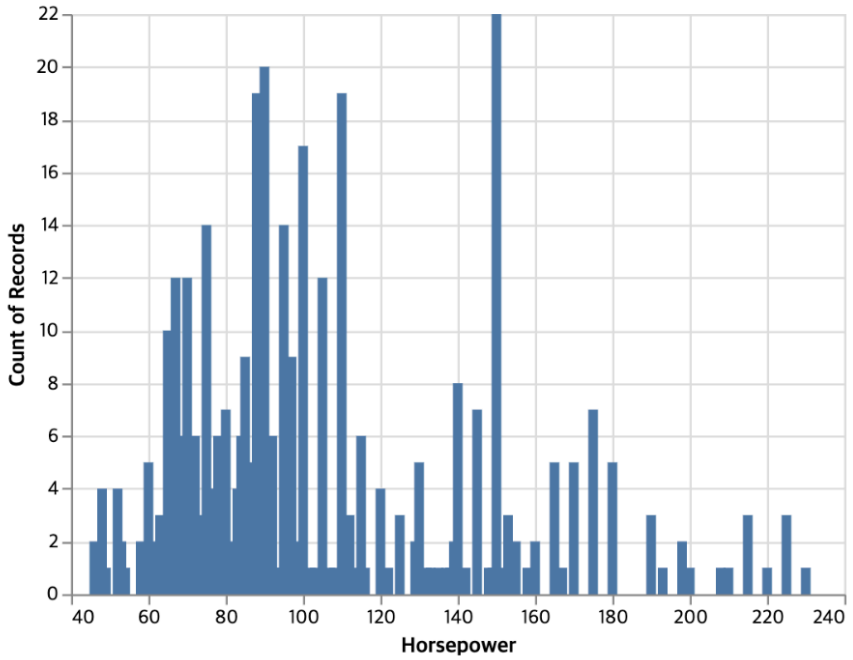
hci lab
HUMAN × COMPUTER INTERACTION

Altair 실습 1

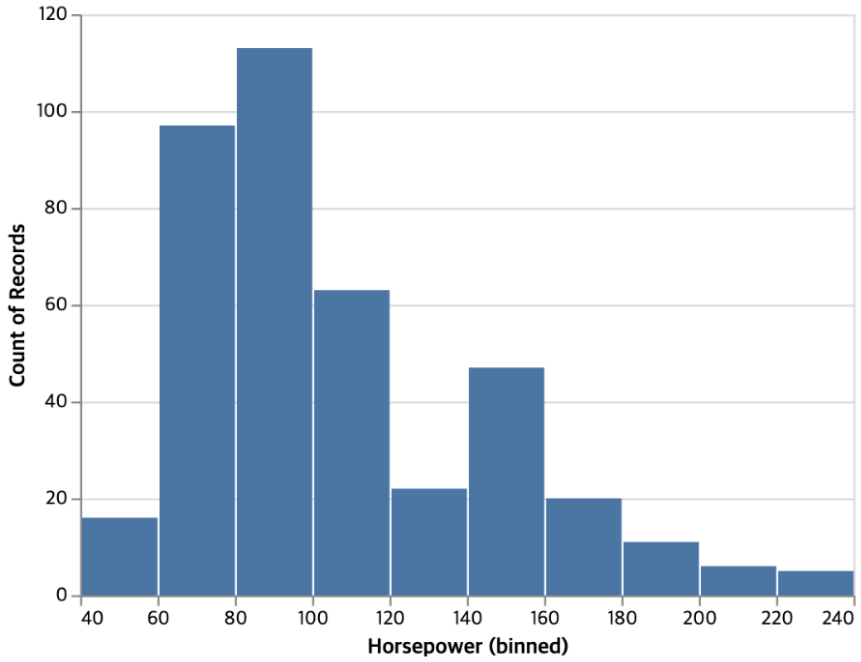
Binning, Aggregation, and Sorting

Encoding Channels – Binning & Aggregation

```
alt.Chart(cars).mark_bar().encode(
    alt.X('Horsepower'),
    y='count()'
)
```



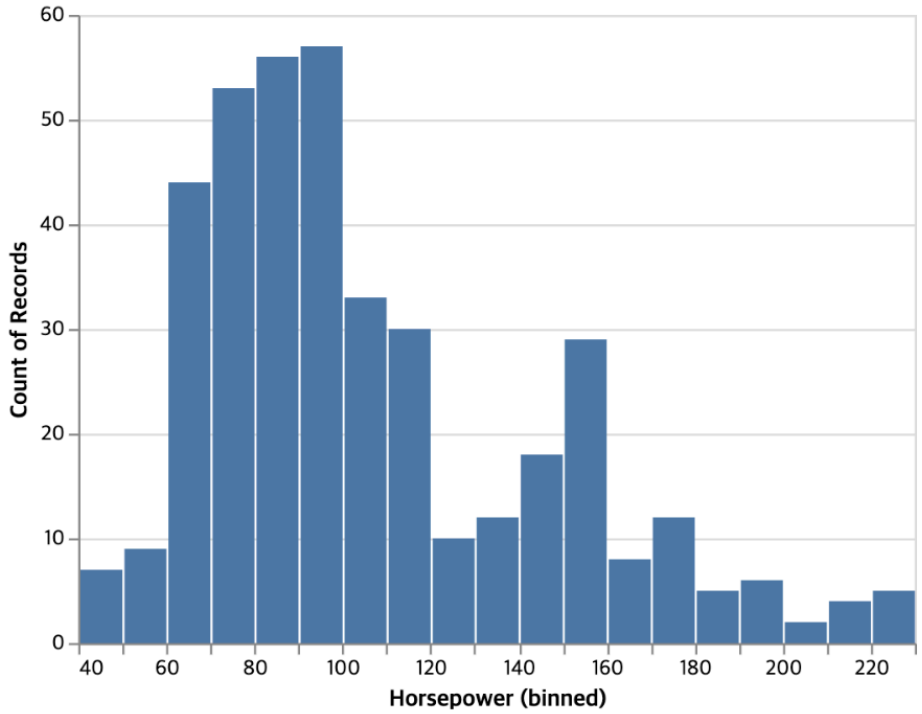

```
alt.Chart(cars).mark_bar().encode(
    alt.X('Horsepower', bin=True),
    y='count()'
)
```



Encoding Channels – Binning & Aggregation

- Bin의 수를 조절하고 싶다면?

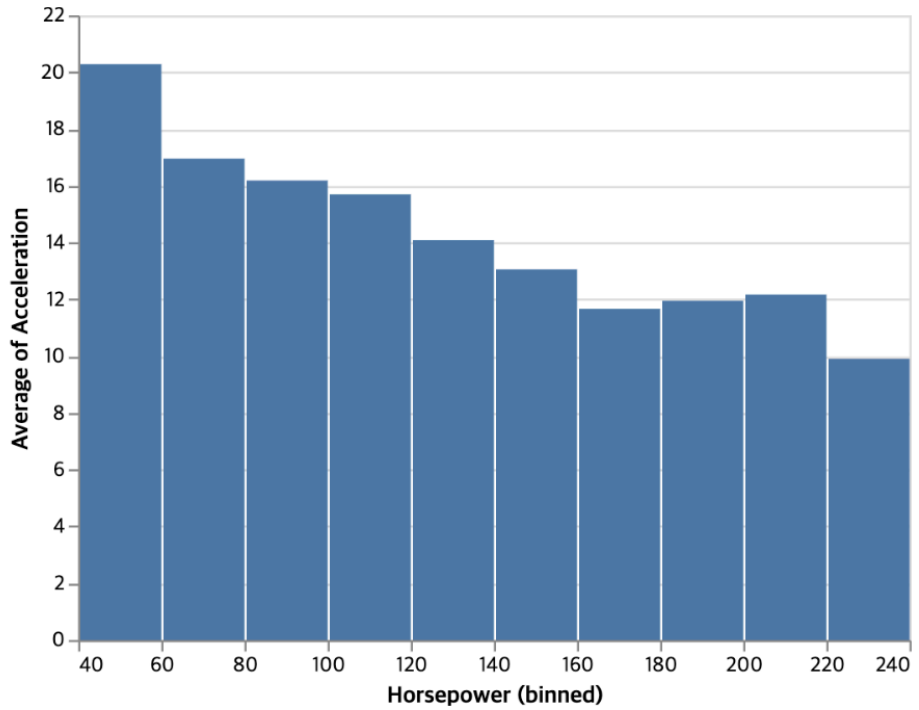
```
alt.Chart(cars).mark_bar().encode(
    alt.X('Horsepower', bin=alt.Bin(maxbins=20)),
    y='count()'
)
```



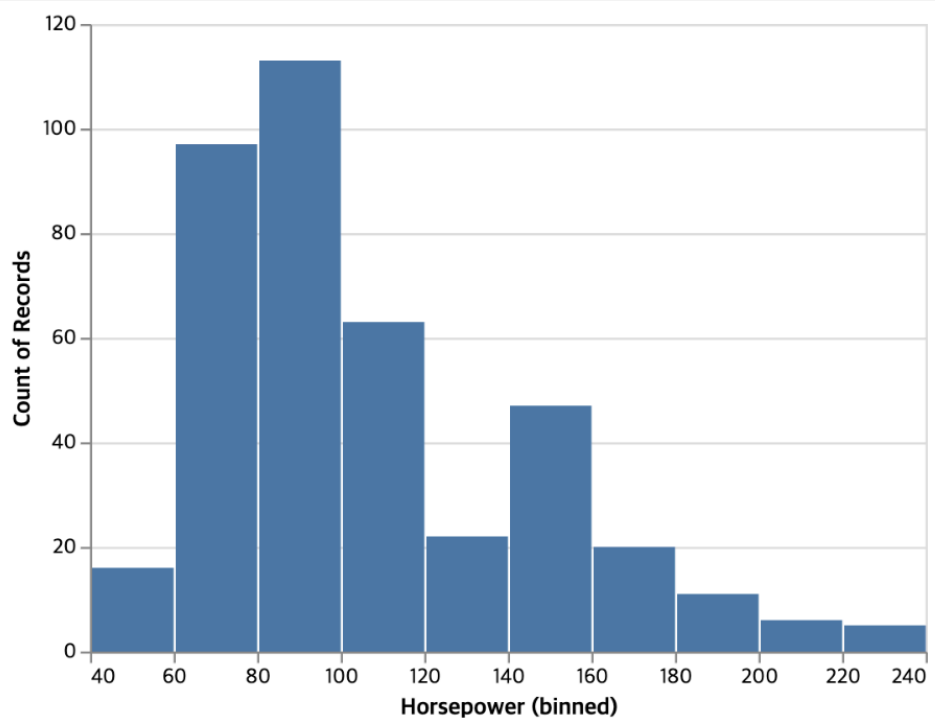
Encoding Channels – Binning & Aggregation

- Aggregation 방법을 바꾸고 싶다면?

```
alt.Chart(cars).mark_bar().encode(
    alt.X('Horsepower', bin=alt.Bin(maxbins=10)),
    y='average(Acceleration):Q'
)
```




```
alt.Chart(cars).mark_bar().encode(
    alt.X('Horsepower', bin=alt.Bin(maxbins=10)),
    y='count(Acceleration):Q'
)
```



Encoding Channels – Sorting

- Sorting Axis

```
barley = data.barley()

base = alt.Chart(barley).mark_bar().encode(
    y='mean(yield):Q',
    color=alt.Color('mean(yield):Q', legend=None)
).properties(width=100, height=100)
```

```
ascending = base.encode(
    alt.X(field='site', type='nominal', sort='ascending')
).properties(
    title='Ascending'
)
```

```
explicit = base.encode(
    alt.X(field='site', type='nominal',
          sort=['Duluth', 'Grand Rapids', 'Morris',
                'University Farm', 'Waseca', 'Crookston'])
).properties(
    title='Explicit'
)
```

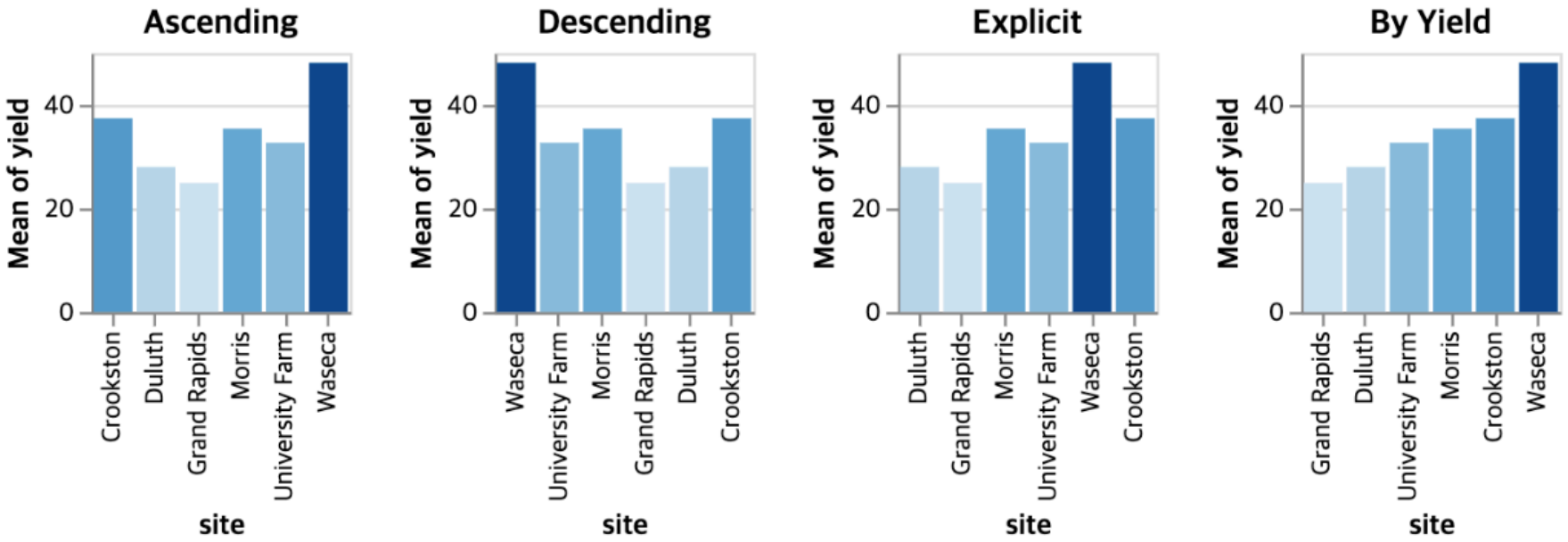
```
descending = base.encode(
    alt.X(field='site', type='nominal', sort='descending')
).properties(
    title='Descending'
)
```

```
sortfield = base.encode(
    alt.X(field='site', type='nominal',
          sort=alt.EncodingSortField(field='yield', op='mean'))
).properties(
    title='By Yield'
)
```

Encoding Channels – Sorting

- Sorting Axis

```
ascending | descending | explicit | sortfield
```



Encoding Channels – Sorting

- Sorting Legend

```
alt.Chart(barley).mark_rect().encode(
    alt.X('mean(yield):Q', sort='ascending'),
    alt.Y('site:N', sort='descending'),
    alt.Color('site:N',
              sort=['Morris', 'Duluth', 'Grand Rapids',
                    'University Farm', 'Waseca', 'Crookston'])
)
```