Visualization for Data Science
Exploratory Data Analysis III

Administrivia – Instructor Absences

October 20th through October 27th

- Dr. K is in Gaborone for CompEd Conference
- October 20th Class8A. No in-person class. Two tutorial files this week.
- October 22nd Class8B. Matt will lead the lecture.
- October 27th Class 9A. No in-person class. Zoom lecture. Flight lands at 1:30pm so we will zoom together at 3:30pm. The zoom lecture will also be recorded.

Administrivia – Quiz Update

Quiz 5

- Monday and Tuesday this week
- ~~All tests will be hidden~~

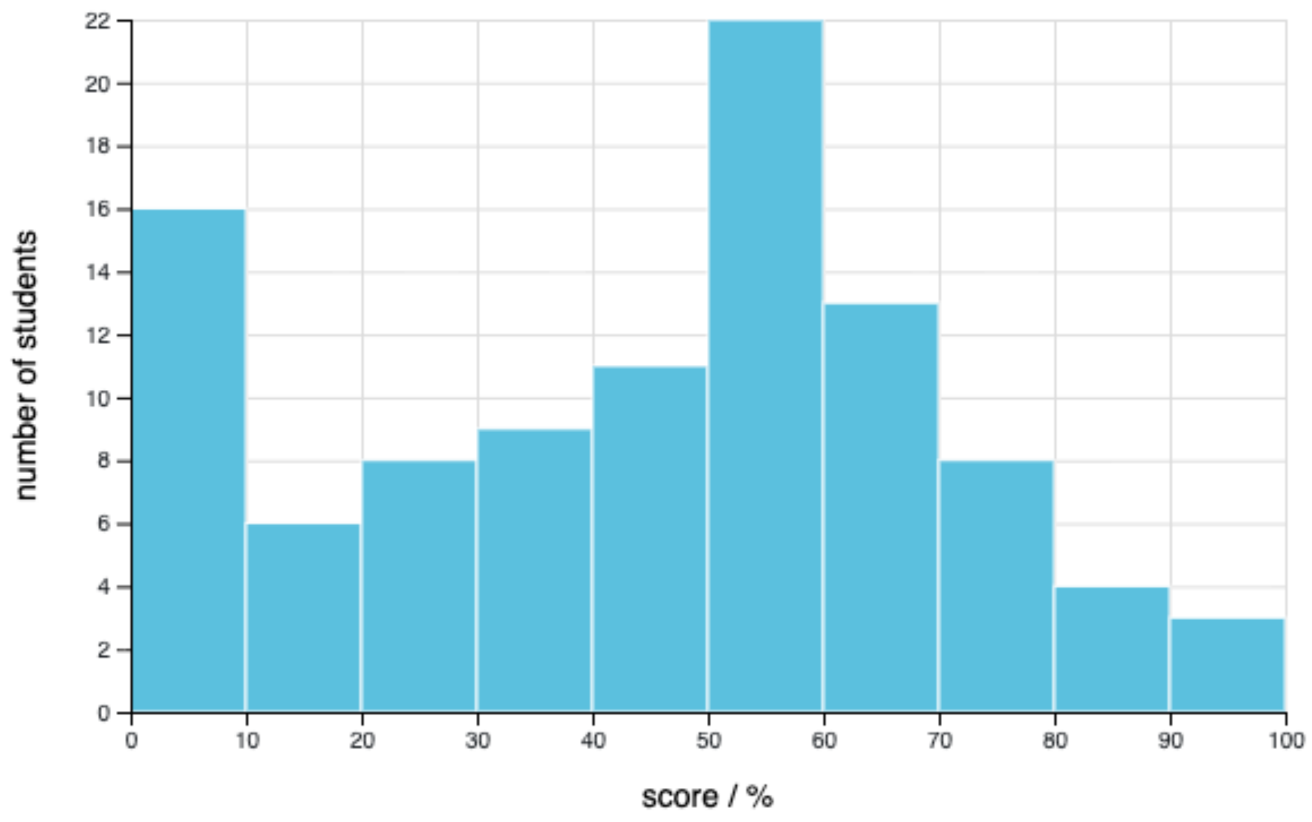
Quiz 6

- Tuesday and Wednesday next week
- All tests will be hidden

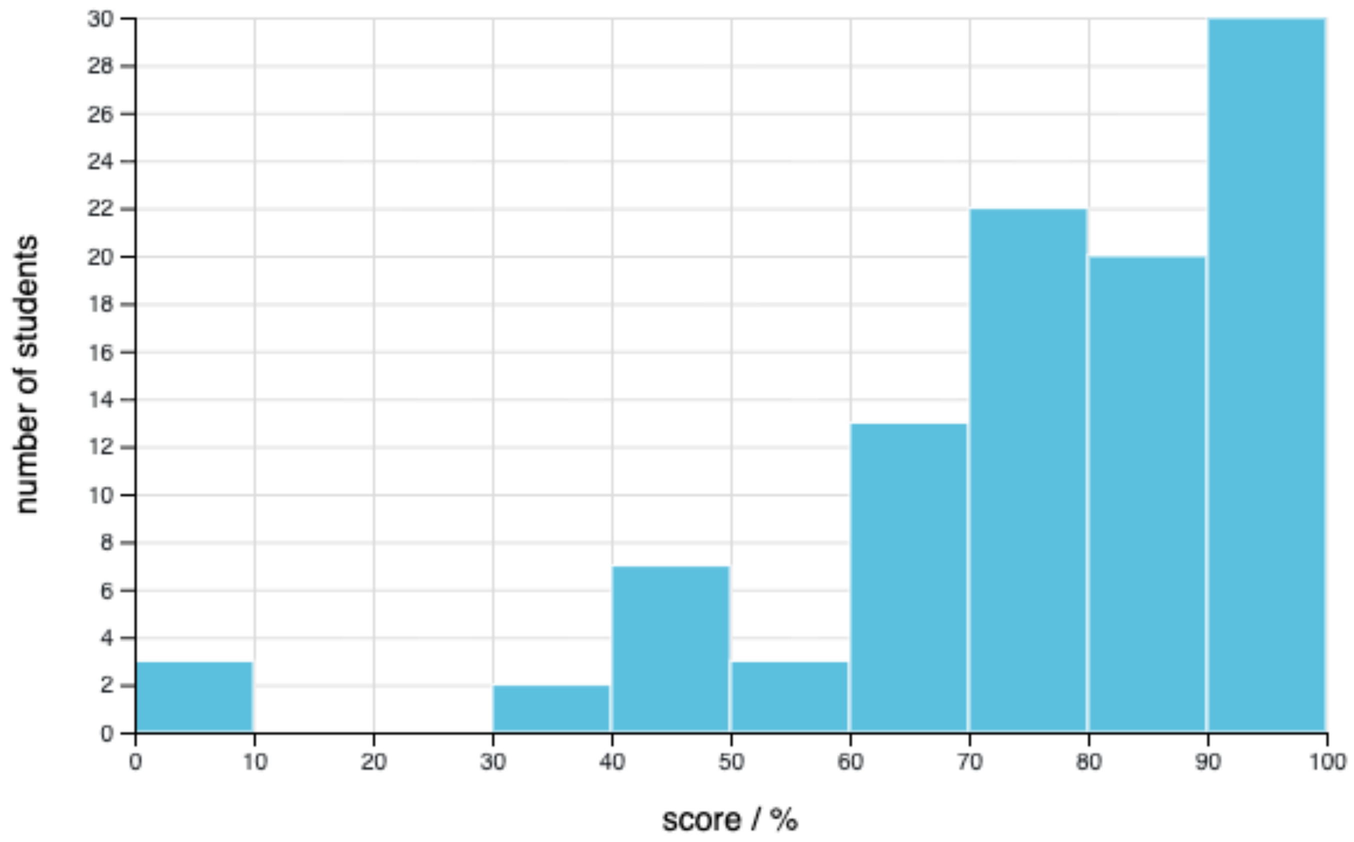
Quiz 4 Stats

quiz01	93	41%	47 min 17 s
quiz02	98	100%	24 min 18 s
quiz03	99	88%	9 h 37 min 27 s
quiz04	100	43%	45 min 53 s
quiz05	100	76%	45 min 51 s

Quizzes 4: Score statistics



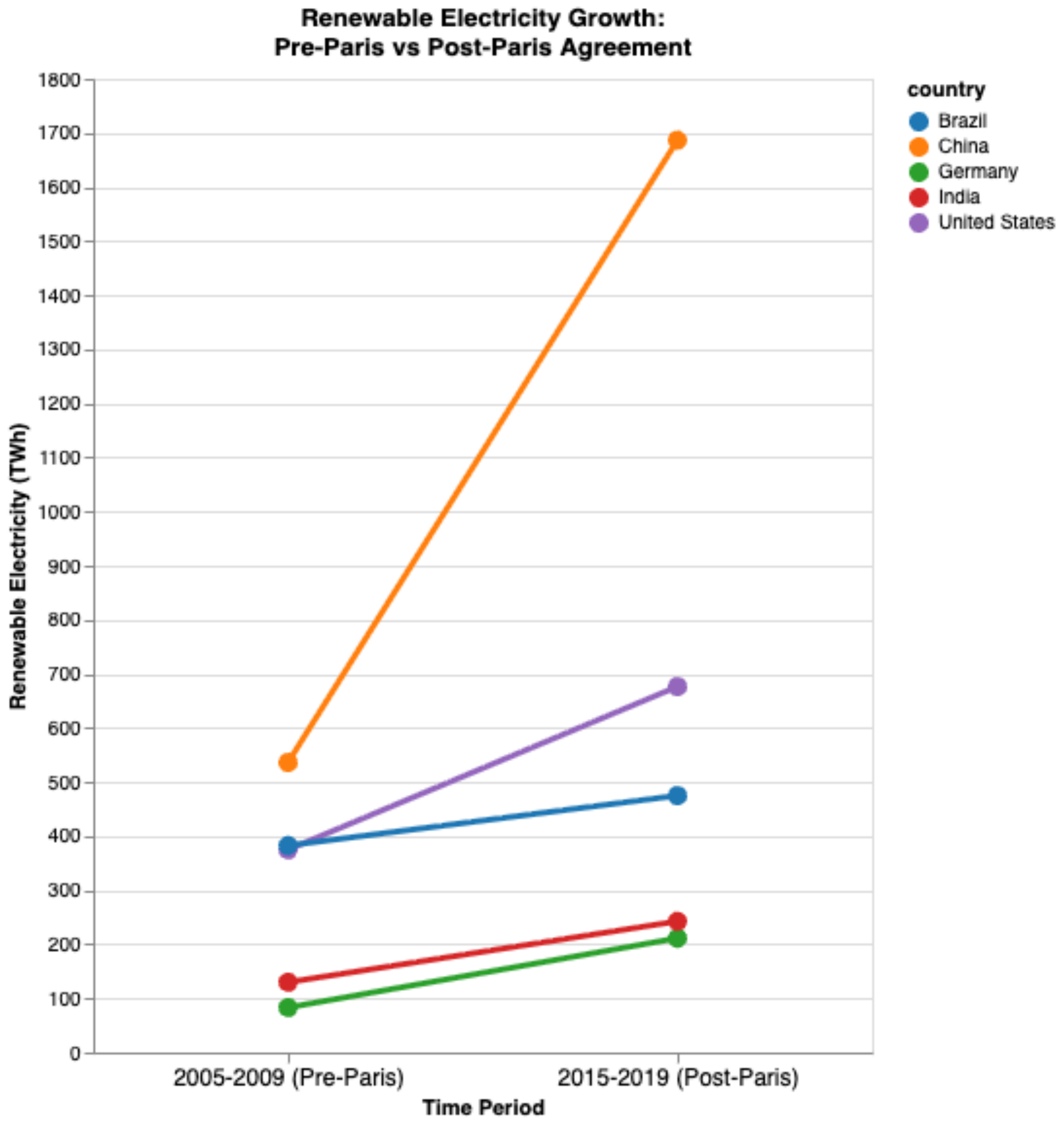
Quizzes 5: Score statistics



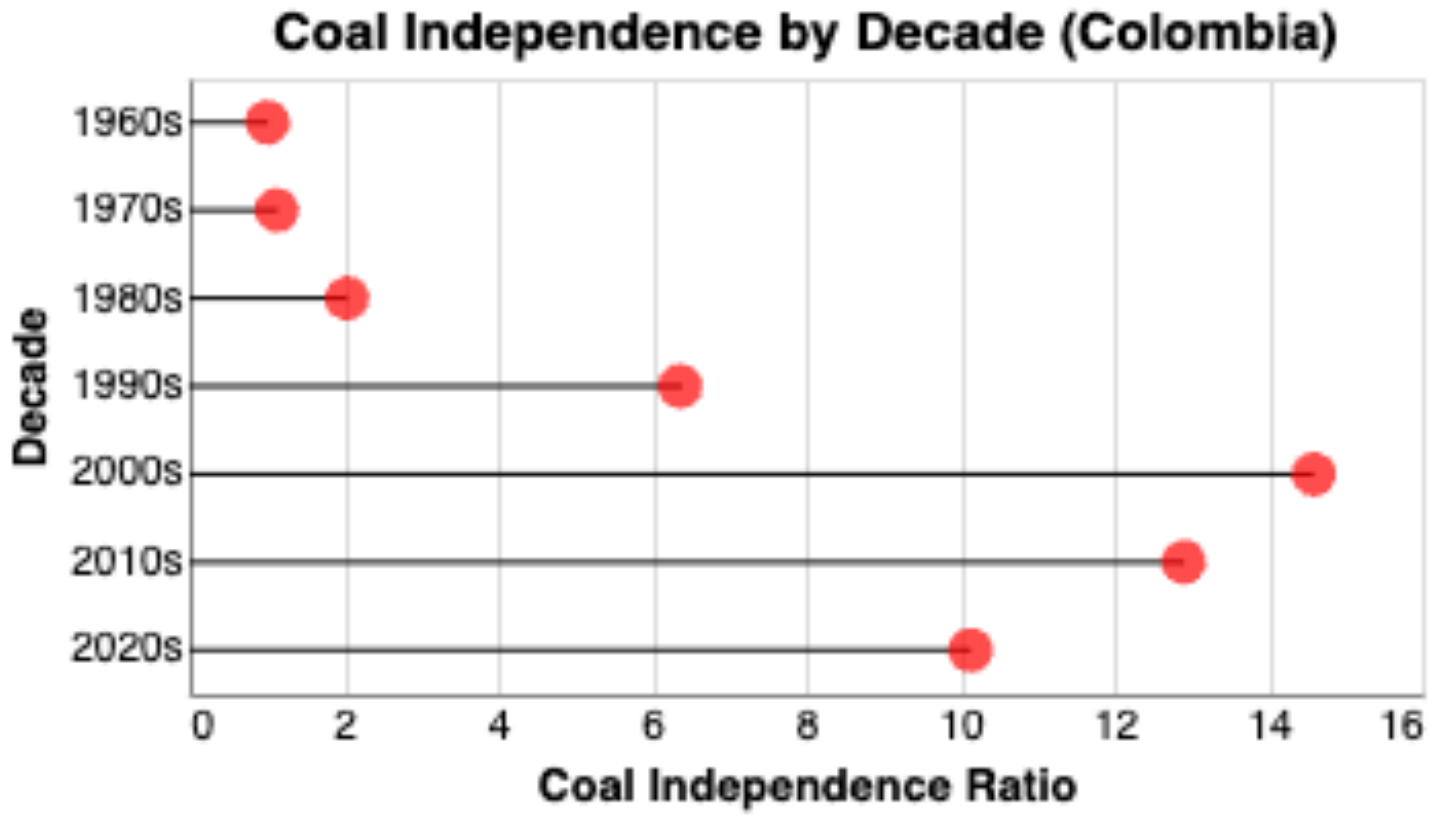
Quiz 4 – Where did you struggle (SELECT ALL)

A. Data Task – converting data from wide to long
B. Viz Task – Stacked Area Chart
C. Viz Task – Heatmap
D. Viz Task – Lollipop
E. Data Task – Pandas Wrangling

Lollipop



Renewable Electricity Growth:
Pre-Paris vs Post-Paris Agreement
Renewable Electricity (TWh)
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
2005-2009 (Pre-Paris)
2015-2019 (Post-Paris)
Time Period
country
Brazil
China
Germany
India
United States



Coal Independence by Decade (Colombia)
Decade
1960s
1970s
1980s
1990s
2000s
2010s
2020s
0
2
4
6
8
10
12
14
16
Coal Independence Ratio

```python
base = alt.Chart(slope_data)

# Lines showing change (core of slope graph)
lines = base.mark_line(strokeWidth=3).encode(
    x=alt.X('period_label:O', title='Time Period',
            axis=alt.Axis(labelAngle=0, labelFontSize=12)),
    y=alt.Y('renewables_electricity:Q', title='Renewable Electricity (TWh)',
            axis=alt.Axis(format='.0f')),
    color=alt.Color('country:N', title='Country', scale=alt.Scale(scheme='category10'), legend = None),
)

# Points at each period for precision
points = base.mark_circle(size=100, opacity = 1).encode(
    x='period_label:O',
    y='renewables_electricity:Q',
    color=alt.Color('country:N', scale=alt.Scale(scheme='category10'), legend= None),
)
# Combine all layers
slope_graph = alt.layer(lines, points, labels).resolve_scale(
    color='independent'
)
slope_graph
```

```python
base = alt.Chart(colombia_data).encode(
    y=alt.Y(
        'decade_label:O',
        title='Decade',
        axis=alt.Axis(labels=True, ticks=False)  # keep labels, remove ticks
    ),
    x=alt.X('coal_independence:Q')
)

# Lollipop "stick"
sticks = base.mark_rule().encode(
    x=alt.X('coal_independence:Q').title('Coal Independence Ratio').axis(
            labels=True, ticks=False),
    x2=alt.value(0)
)

# Lollipop "head"
circles = base.mark_circle(size=120, color='red').encode(# SOLUTION
    x='coal_independence:Q'
)

# Combine and set the global properties of width, height and title.
lollipop = (sticks + circles).properties(
    width=300,
    height=150,
    title="Coal Independence by Decade (Colombia)"
).configure_axis(
    ticks=False  # remove ticks globally
)
lollipop
```

Chart Specification:

STEP 1: Create a base chart using the `colombia_data` dataset. For this chart

- Encode `decade_label` on the **y channel** as ordinal.
- Encode `coal_independence` on the **x channel** as quantitative.
- Set y-axis title to `"Decade"`.
- For the y axis, remove the ticks but keep the labels (`ticks = False`, `labels = True`)
- NOTE: This chart has no mark but don't worry it will be addressed later

STEP 2: Use mark_rule to draw the "sticks" from 0 to the coal independence value for each decade.

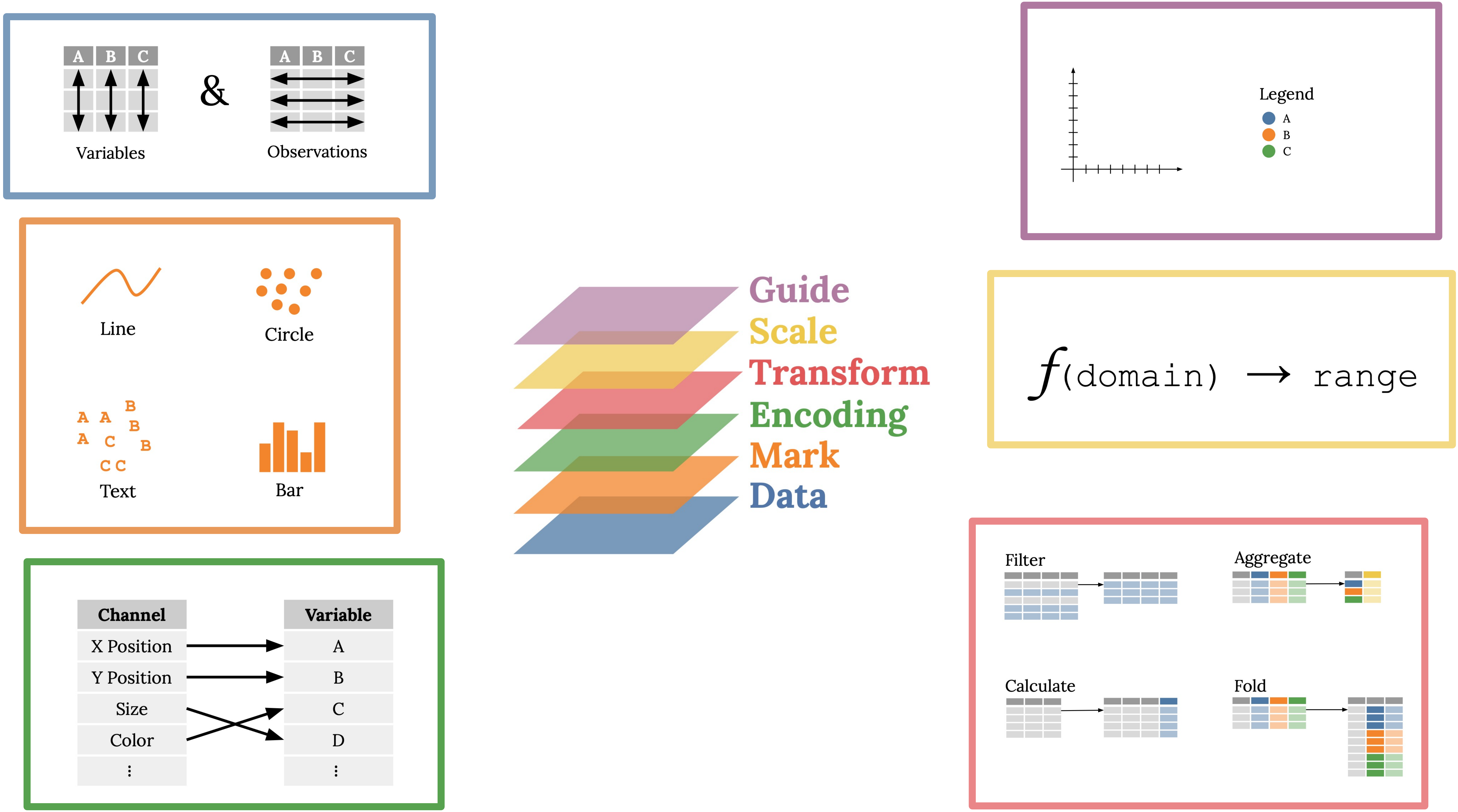
- Startwith the mark `mark_rule`
- Encode `coal_independence` on the **x channel**
- Set x-axis title to `"Coal Independence Ratio"`.
- For the x axis, remove the ticks but keep the labels (`ticks = False`, `labels = True`)
- Encode `alt.value(0)` on the **x2 channel** as quantitative.

STEP 3:Use mark_circle to draw the "heads" of the lollipops at the coal independence value.

- Encode `coal_independence` on the **x channel**
- Circle color: `red`.
- Circle size: 120 pixels.

Additional Styling Specifications:

- Remove axis ticks but keep labels readable.
- Width: 300px, Height: 150px.
- Chart title: *"Coal Independence by Decade (Colombia)"*.



A B C
Variables
&
A B C
Observations
Line
Circle
A A B A B A C B C C
Text
Bar
Channel
Variable
X Position
A
Y Position
B
Size
C
Color
D
⋮
⋮
Guide
Scale
Transform
Encoding
Mark
Data
Legend
A
B
C
f(domain) → range
Filter
Aggregate
Calculate
Fold

Hierarchical Grammar of Specification

Altair's structure is declarative — it describes *what* to draw, not *how*. Each visualization is built from **nested layers of specification objects**, which belong to one of these levels:

Level	Purpose	Example
Chart	Container for data, transformations, and top-level properties.	`alt.Chart(df)`
Mark	Defines *what kind of geometry* to draw.	`.mark_bar()` , `.mark_boxplot()` , `.mark_area()`
Encoding	Maps data fields to *visual channels*.	`.encode(x='var:Q', y='count()')`
Channel Definition	Customizes each encoding (axis, scale, title, etc.).	`alt.X('var:Q', axis=alt.Axis(...))`
Property / Config	Top-level stylistic and layout settings.	`.properties(width=400, title='My Plot')`
Transform	Data manipulation before rendering.	`.transform_density(...)` , `.transform_filter(...)`

Structure Overview

Every Altair chart follows this **hierarchical specification pattern**:

```
Chart
 ├── Data
 ├── Transform(s)
 ├── Mark (geometry + style)
 ├── Encoding(s)
 │     ├── Channel Definitions (x, y, color, size…)
 │     │     ├── Field & Type
 │     │     ├── Optional transformations (bin, aggregate)
 │     │     ├── Visual customizations (axis, scale, legend)
 ├── Facet/Repeat/Layer (layout logic)
 ├── Properties (title, width, height)
 └── Configurations (global style)
```

If you step back, every chart can be summarized as:

```
(
alt.Chart(data)
    .transform_*()
    .mark_*(
        # base mark styling
        property=value,
        submark={...},  # for complex marks
    )
    .encode(
        alt.X(field:type, **channel_options),
        alt.Color(...),
        ...
    )
    .facet(...) / .repeat(...)
    .properties(width=..., height=..., title=...)
    .configure_*()
)
```

2. Customizing Channels (Encodings)

Each encoding channel describes **what data** goes on **which visual dimension**. You can write them in **shorthand** or **expanded form**:

Basic shorthand:

```
x='weight:Q'
y='species:N'
color='gender:N'
```

Expanded form:

```
x=alt.X(
    field ='weight',     #this is the long form, we just typically omit the field and type parameters and just type
    type= 'quantitative',
    bin=alt.BinParams(maxbins=20),
    axis=alt.Axis(grid=False),
    scale=alt.Scale(domain=[0,1000]),
    title='Weight (g)'
)
```

Method Chaining Option

```
x=alt.X('weight:Q').bin(maxbins=20).scale(domain=[0,1000]).axis(grid=False).title('Weight (g)')
```

Channel customization objects

Object	Used For	Common Parameters
`alt.X()` , `alt.Y()`	Position channels	`title` , `scale` , `bin` , `axis` , `sort` , `aggregate` ,
`alt.Color()`	Color mapping	`scale` , `legend` , `title`
`alt.Size()`	Bubble area, line width	`scale` , `legend` , `title`
`alt.Shape()`	Marker shape	`legend` , `scale`
`alt.Column()` / `alt.Row()`	Faceting	`title` , `header` , `spacing`
`alt.Facet()`	Multi-view layout	`columns` , `title` , `resolve_scale`

Each has **optional helper constructors**:

Supporting Style Objects

Helper	Controls	Example
`alt.Axis()`	Axis appearance	`alt.Axis(title='Weight', grid=False, ticks=True)`
`alt.Scale()`	Scale behavior	`alt.Scale(type='log', domain=[0,1000])`
`alt.Legend()`	Legend layout	`alt.Legend(title='Species', orient='right')`
`alt.Header()`	Facet header	`alt.Header(titleOrient='bottom', labelPadding=0)`

EDA with Hawks Dataset

Cooper's hawk, photo by Mike Baird

Red-tailed hawk, photo by Don Sniegowski

Sharp-shinned hawk, photo by Tod Petit

How to choose bin sizes – Existing Methods

- Square Root Rule: $\sqrt{(number\ of\ data\ items)}$
  - Simple, most commont, not sensitive to distribution shape
- Sturges' Rule: $\lceil \log_2(n)+1 \rceil$
  - best used for normally distributed data, works best for smaller datasets, one limitation is that it can oversimplify for large datasets
- Doane's Formula: improvement to Sturges', best used for skewed distributions, it adjusts for skewness
- Rice Rule: $\lceil 2 \cdot n^{1/3} \rceil$
  - Similar to square-root rule but gives more bins
- Scott's Rule: based on the standard deviation, it minimizes bias
- Freedman-Diaconis Rule: based on the IQR, it is more robust to outliers and skewed distributions.

So what should I do?

By Sample Size

Sample Size	Recommended Method	Typical Bins
n < 30	5-7 bins fixed	5-7
30 ⩽ n < 100	Square Root	6-10
100 ⩽ n < 1000	Sturges' or Square Root	10-30
n ⩾ 1000	Freedman-Diaconis	20-50
n ⩾ 10,000	Freedman-Diaconis or Scott's	50-200

By Data Characteristics

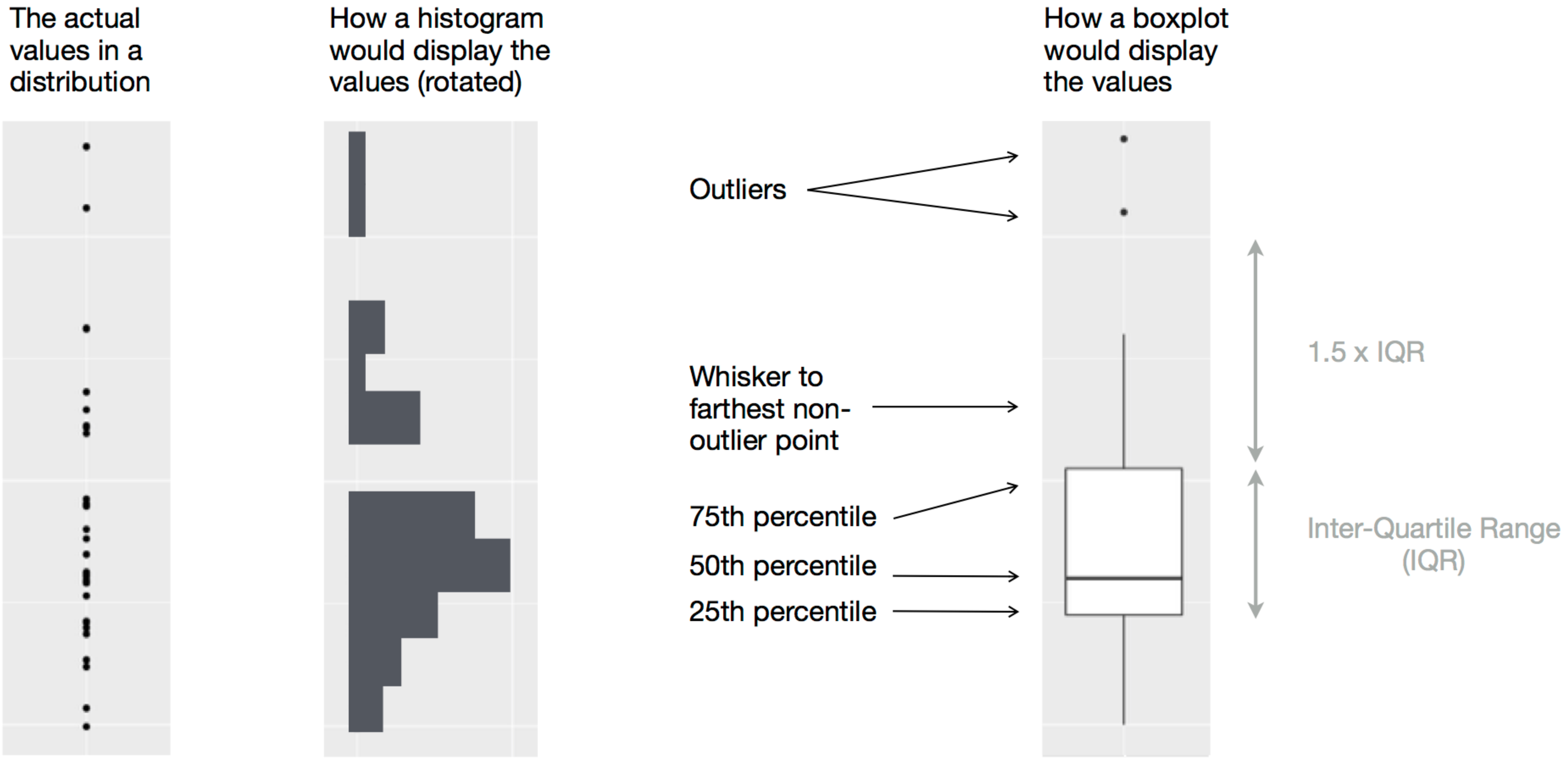
Data Type	Best Method	Why
Normal distribution	Sturges' or Scott's	Designed for normal data
Skewed distribution	Doane's or Freedman-Diaconis	Handles asymmetry
With outliers	Freedman-Diaconis	Uses robust IQR
Unknown distribution	Freedman-Diaconis	Most generally applicable
Multimodal	Start with more bins (30-50)	Need to see multiple peaks

By Purpose

Purpose	Recommendation	Bins
Quick exploration	Square Root	Medium (10-30)
Presentation	Manual tuning for clarity	Fewer (5-15)
Detailed analysis	Freedman-Diaconis	More (20-50)
Finding patterns	Try multiple, compare	Iterate
Publication	Freedman-Diaconis + adjust	Well-justified

By default, Altairs uses maxbins=10 as default.
It isn't data specific.
So make sure that you do some investigation before proceeding.

Multivariate Visual Idioms: Boxplots



https://r4ds.had.co.nz/exploratory-data-analysis.html

"In a nutshell: You should always perform appropriate EDA before further analysis of your data. Perform whatever steps are necessary to become more familiar with your data, check for obvious mistakes, learn about variable distributions, and learn about relationships between variables. EDA is not an exact science ***{ it is a very important art!}****"*

- Howard J. Seltman