Visualization (Exploring variation)

Peter Ganong and Maggie Shi

January 25, 2026

Introduction

Roadmap of lecture

- Introduce three key ideas in data visualization
  - exploration vs. production
  - headline vs. sub-messages
  - iteration
- introduce datasets for this lecture
- Categorical variables (in lecture 2 we called these `N`/`O`)
- Continuous variables (in lecture 2 we called these `Q`)
  - Exploring typical values
  - Exploring and dealing with unusual values

What is exploratory data analysis?

Data visualization has two distinct goals

1. **exploration** for you to *learn* as much as possible
2. **production** for you to *teach* someone else what you think the key lessons are

Exploration vs. production

- When you are in exploration mode, you will look at lots of patterns and your brain filters out the noise
- Production mode is like putting a cone on your dog.
- You are deliberately limiting the reader's field of vision such that they see the key messages from the plot *and avoid too many distractions*

“A Sunday on La Grande Jatte” by Seurat

“A Sunday on La Grande Jatte” by Seurat

Headline vs submessages

- **Headline** is what you see first when you look at the painting or you look at the plot.
- **Submessages** are what you see later, on closer inspection
- Often, the most interesting patterns in the data are ones that you don’t see right away when you make the very first plot.

Iterating on plot design

> "Make dozens of plots"

Quoctrung Bui, former 30538 guest lecturer and former Harris data viz instructor

Iterating on plot design

What does he mean?

- The first plot you make will never be the one you should show
- As you are generating graphs for yourself in exploration mode, you will produce many candidates that could end up being used in production mode
- As a rule of thumb, you should try out at least three different plotting concepts (marks)
- Within each concept, you will need to try out several different encodings

Summary:

1. Decide if you are trying to explore the data or produce a plot for someone else
2. For any given plot, look closely (like Seurat) beyond just the headline
3. Iterate

Intro to data

Introduction to data

- Most of our visualization lectures are based on the University of Washington textbook, but the textbook doesn't have enough material on exploratory data analysis. So we are supplementing with
  - Data Visualization
  - Exploratory Data Analysis material in the **R for Data Science** textbook (with the code translated to Altair)
- `diamonds`, `mpg` are from "Exploratory Data Analysis"
- `movies` (also used last lecture) is from the UW textbook
- `penguins` is from "Data Visualization"

Categorical variables

Categorical variables: roadmap

- introduce `diamonds`
- show table
- show bar graph

introduce dataset `diamonds`

```python
import pandas as pd
from plotnine.data import diamonds
diamonds.shape
```

```
(53940, 13)
```

```python
diamonds.head()
```

	carat	cut	color	clarity	depth	table	price	x	y
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35

`diamonds` data dictionary

Variable	Definition	Values
`price`	price in USD	\$326–\$18,823
`carat`	weight of diamond	0.2-5.01
`cut`	quality of the cut	Fair, Good, Very Good, Premium, Ideal
`color`	diamond color	D (best) to J (worst)
`clarity`	measure of how clear diamond is	I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best)
`x`	length in mm	0-10.74
`y`	width in mm	0-58.9
`z`	depth in mm	0-31.8
`depth`	$\frac{z}{mean(x,y)}$	43-79
`table`	width of top of diamond relative to widest point	43-95

Summarizing cut in a table

```python
diamonds_cut = diamonds.groupby('cut').size()
diamonds_cut
```

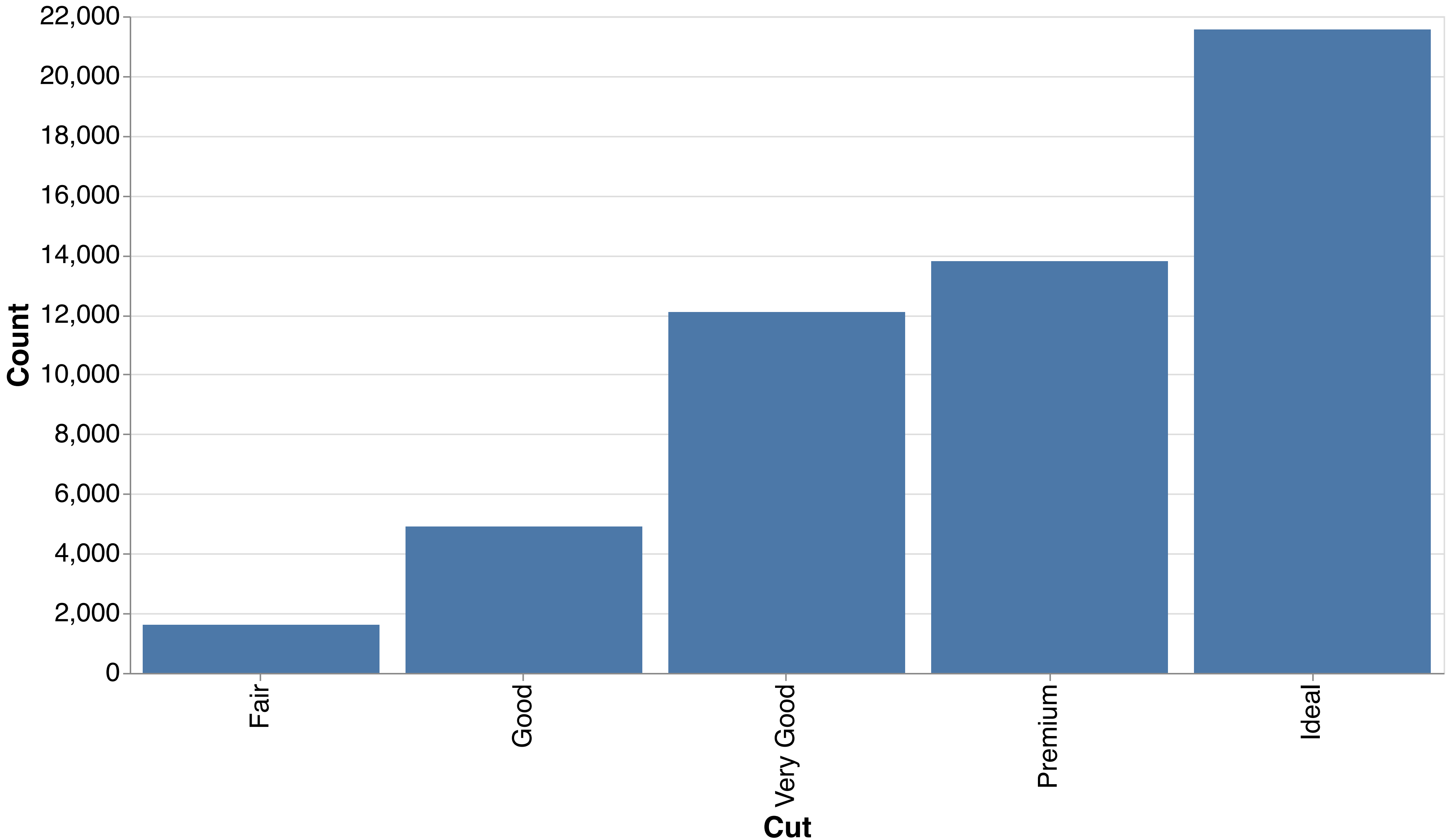
```
cut
Fair          1610
Good          4906
Very Good    12082
Premium      13791
Ideal        21551
dtype: int64
```

Summarizing with a bar graph (code)

```
#moves the index (`cut`) into a normal column so Altair can read it.
diamonds_cut = diamonds_cut.reset_index().rename(columns={0:'N diamonds'})

# extract ordering of `cut` as a Python list to use for sorting
cut_order = diamonds['cut'].cat.categories.tolist()

alt.Chart(diamonds_cut).mark_bar().encode(
    alt.X('cut:O', title = "Cut", sort=cut_order),
    alt.Y('N diamonds:Q', title = "Count")
).properties(width=640, height=360).configure_axis(
    labelFontSize=18,
    titleFontSize=20
)
```

Note: we have included syntax to modify graph properties here. Going forward our `.qmd` source code uses these throughout, but we will omit in the slides for the sake of space.

Summarizing with a bar graph (plot)



Categorical variables: summary

- This section is very brief because there's basically only one good way to plot categorical variables with a small number of categories and this is it.
  - You can use `mark_point()` instead of `mark_bar()`, but overall, there's a clear right answer about how to do this.
- We include this material mainly to foreshadow the fact that we will do a lot on categorical variables in the next lecture when we get to "Exploring Co-variation"

Continuous Variables

Continuous variables: roadmap

- Binning + histograms using `movies`
- Histograms and density plots using `penguins`
- Exploring carat size using `diamonds`

Remark: The skills are absolutely fundamental and so we will intentionally be a bit repetitive.

Recall: `movies` dataset

```
movies_url = 'https://cdn.jsdelivr.net/npm/vega-datasets@1/data/movies.json'
movies = pd.read_json(movies_url)
movies.head()
```

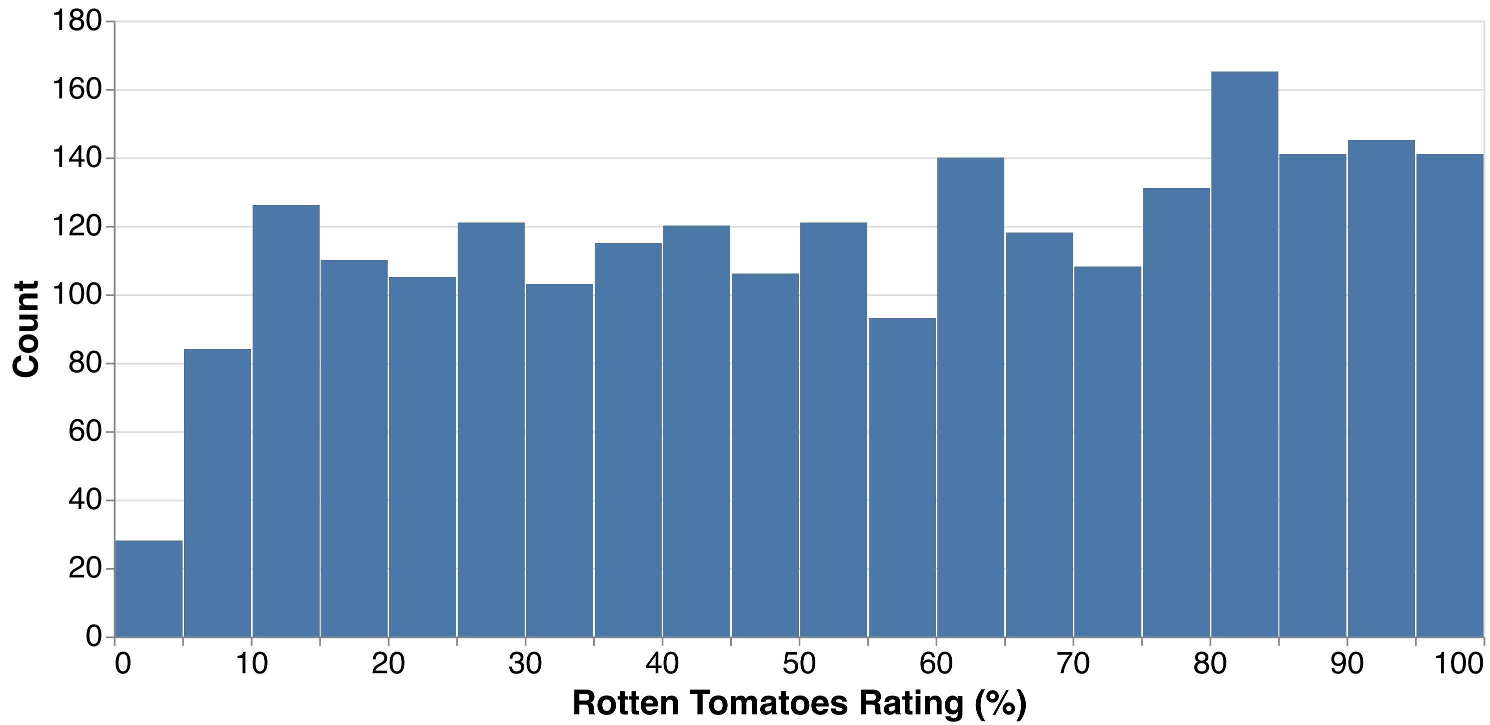
	Title	US_Gross	Worldwide_Gross	US_DVD_Sales	Production_Budget	Release_Date	MPAA_Rating	Runni
0	The Land Girls	146083.0	146083.0	NaN	8000000.0	Jun 12 1998	R	NaN
1	First Love, Last Rites	10876.0	10876.0	NaN	300000.0	Aug 07 1998	R	NaN
2	I Married a Strange Person	203134.0	203134.0	NaN	250000.0	Aug 28 1998	None	NaN
3	Let's Talk About Sex	373615.0	373615.0	NaN	300000.0	Sep 11 1998	None	NaN
4	Slam	1009819.0	1087521.0	NaN	1000000.0	Oct 09 1998	R	NaN

Histogram using `mark_bar()`

- **Rotten Tomatoes** ratings are determined by taking “thumbs up” and “thumbs down” judgments from film critics and calculating the percentage of positive reviews.
- This is a continuous measure, but we can bin it to create a **histogram** of frequencies

Histogram using `mark_bar()`

```
hist_rt = alt.Chart(movies_url).mark_bar().encode(
    alt.X('Rotten_Tomatoes_Rating:Q', bin=alt.BinParams(maxbins=20), title
    alt.Y('count():Q', title = "Count")
)
hist_rt
```

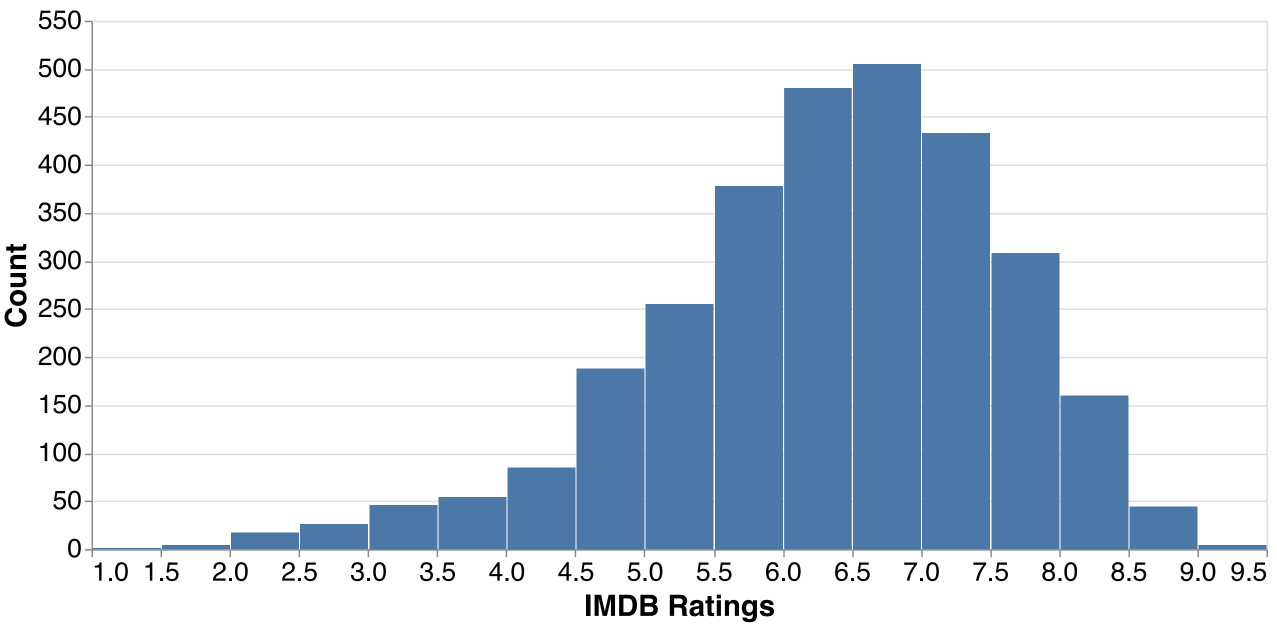


Discussion question: what are the headline and sub-messages?

Histogram of IMDB ratings

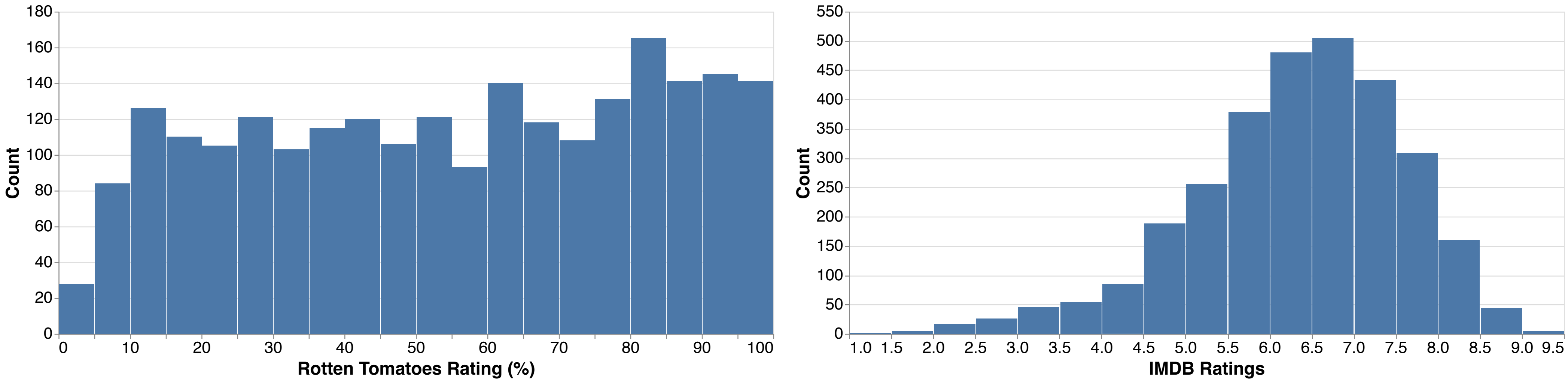
IMDB ratings are formed by averaging scores (ranging from 1 to 10) provided by the site's users.

```
hist_imdb = alt.Chart(movies_url).mark_bar().encode(
    alt.X('IMDB_Rating:Q', bin=alt.BinParams(maxbins=20), title = "IMDB Rat
    alt.Y('count():Q', title = "Count")
)
hist_imdb
```



Side-by-side

```
hist_rt | hist_imdb
```



Discussion question: compare the two ratings distributions. If *your goal for the headline of the graph is about differentiating between good and bad movies*, which rating is more informative?

Introducing the `penguins` dataset

```
url = ("https://raw.githubusercontent.com/mcnakhaee/palmerpenguins/master/p
penguins = pd.read_csv(url)
penguins.head()
```

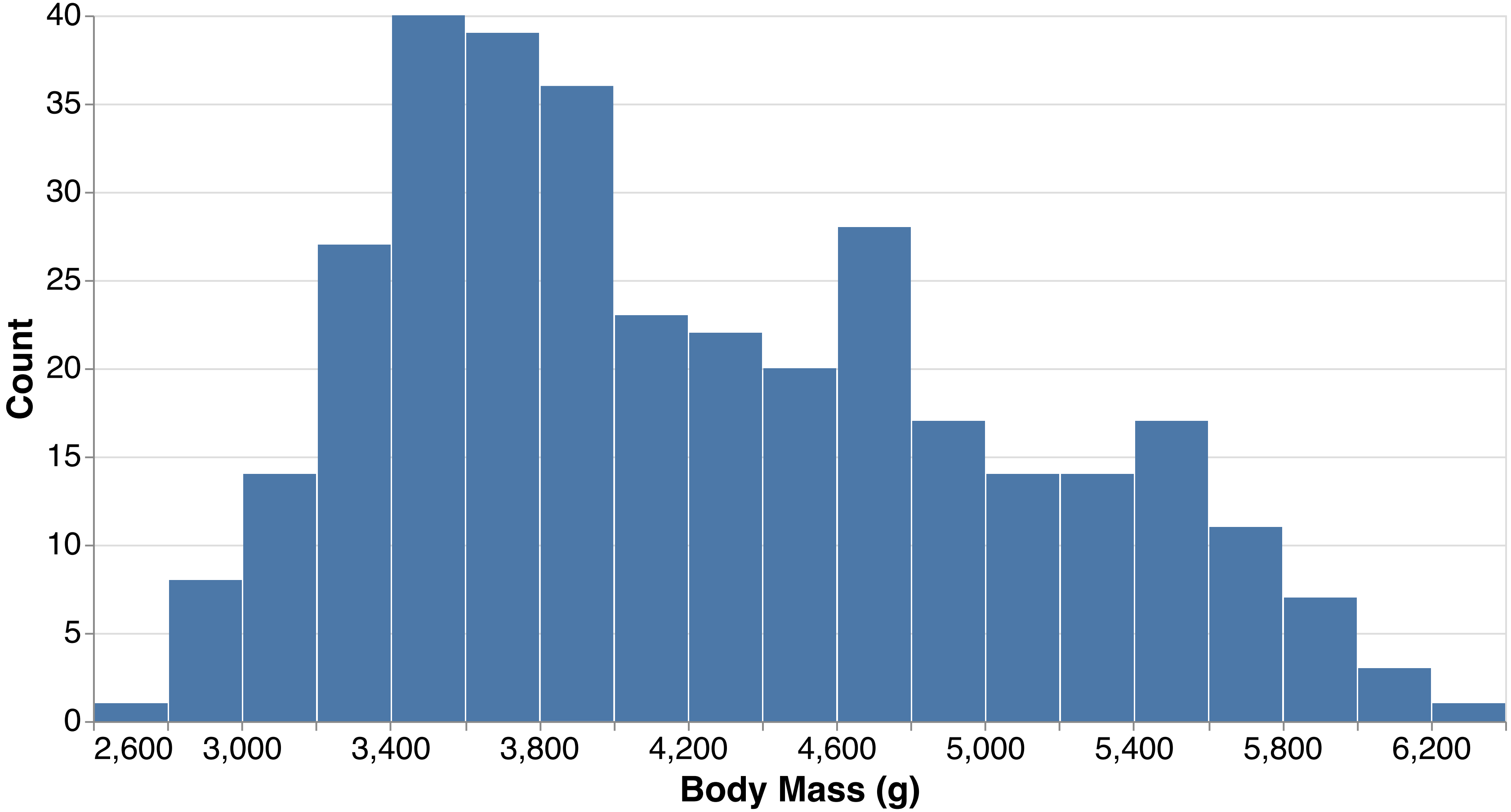
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	boc
0	Adelie	Torgersen	39.1	18.7	181.0	375
1	Adelie	Torgersen	39.5	17.4	186.0	380
2	Adelie	Torgersen	40.3	18.0	195.0	325
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	345

Histogram with steps of 200

- We previously picked the maximum number of equally-spaced bins (`BinParams(maxbins=20)`) and let `altair` choose “nice”-looking bin widths for the histogram
- Alternatively, we can manually control the bin width using `step`

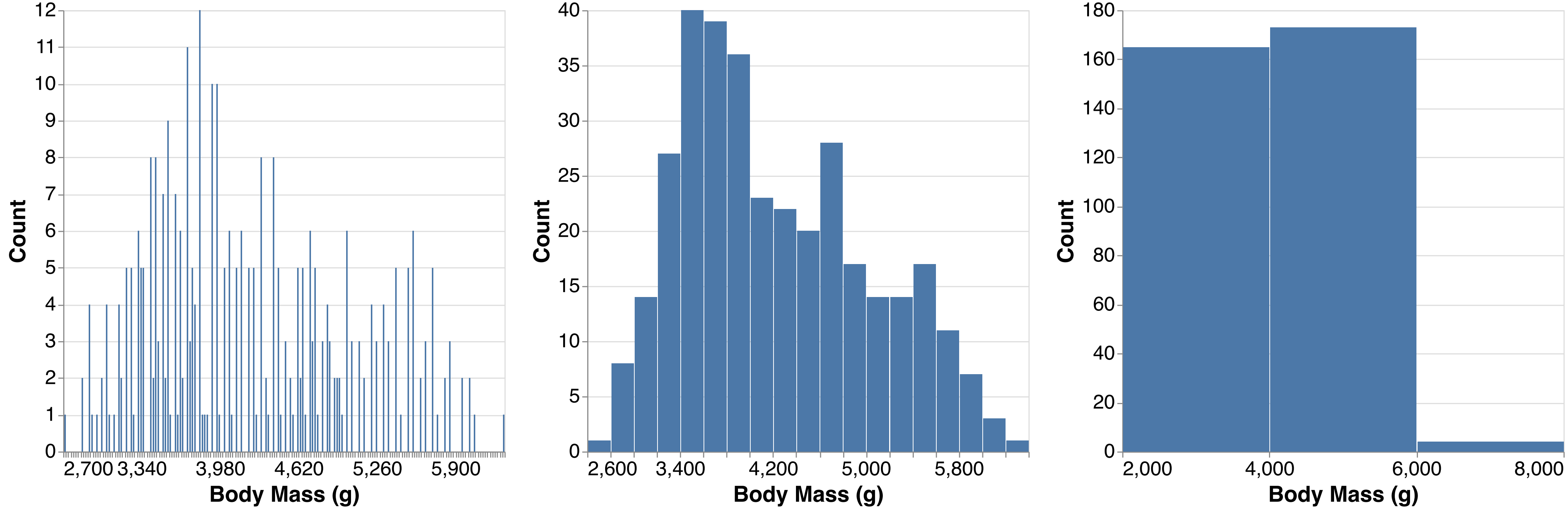
Histogram with steps of 200

```
alt.Chart(penguins).mark_bar().encode(
    alt.X('body_mass_g:Q', bin=alt.BinParams(step=200), title = "Body Mass
    alt.Y('count():Q', title = "Count")
)
```



Histogram `step` parameter

`step=2000` vs. `step=200` vs. `step=20`

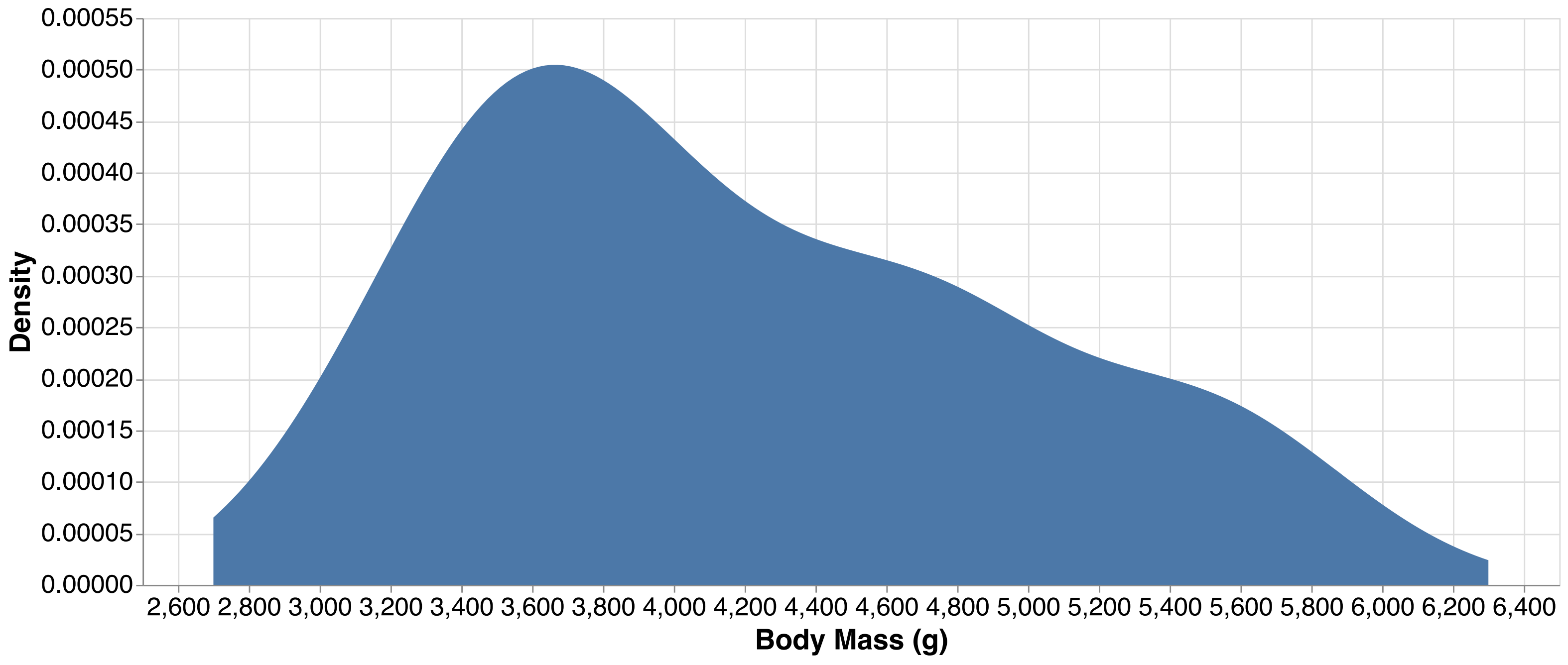


Discussion question: what headline message(s) come from each `binwidth` choice? Which do you prefer?

Density plot

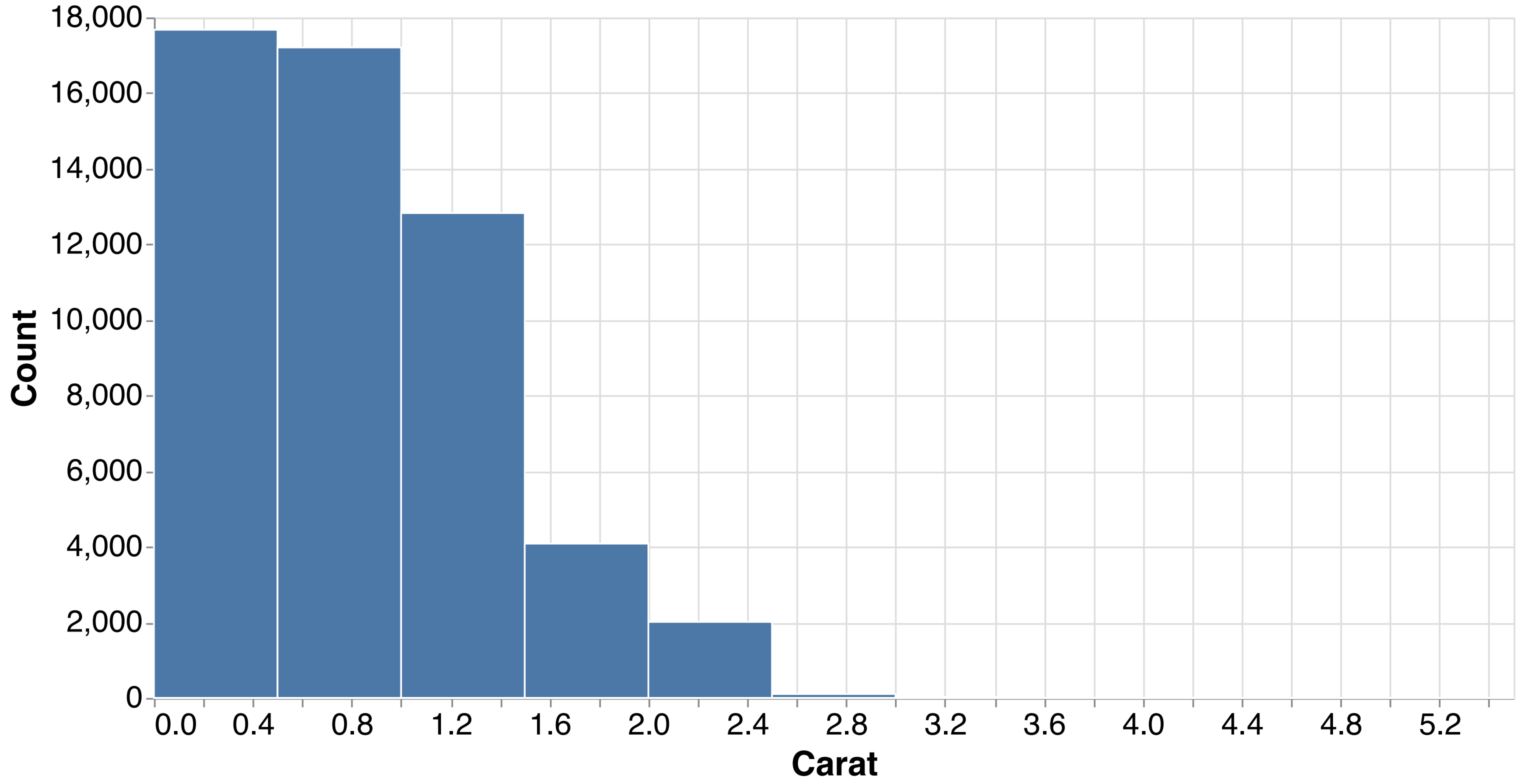
An alternative to a histogram for exploring frequency in continuous variable: density plot using `transform_density`

```
alt.Chart(penguins).transform_density(
    'body_mass_g',
    as_=['body_mass_g2', 'density']
).mark_area().encode(
    alt.X('body_mass_g2:Q', title = "Body Mass (g)"),
    alt.Y('density:Q', title = "Density")
)
```



Back to diamonds, focus on `carat`

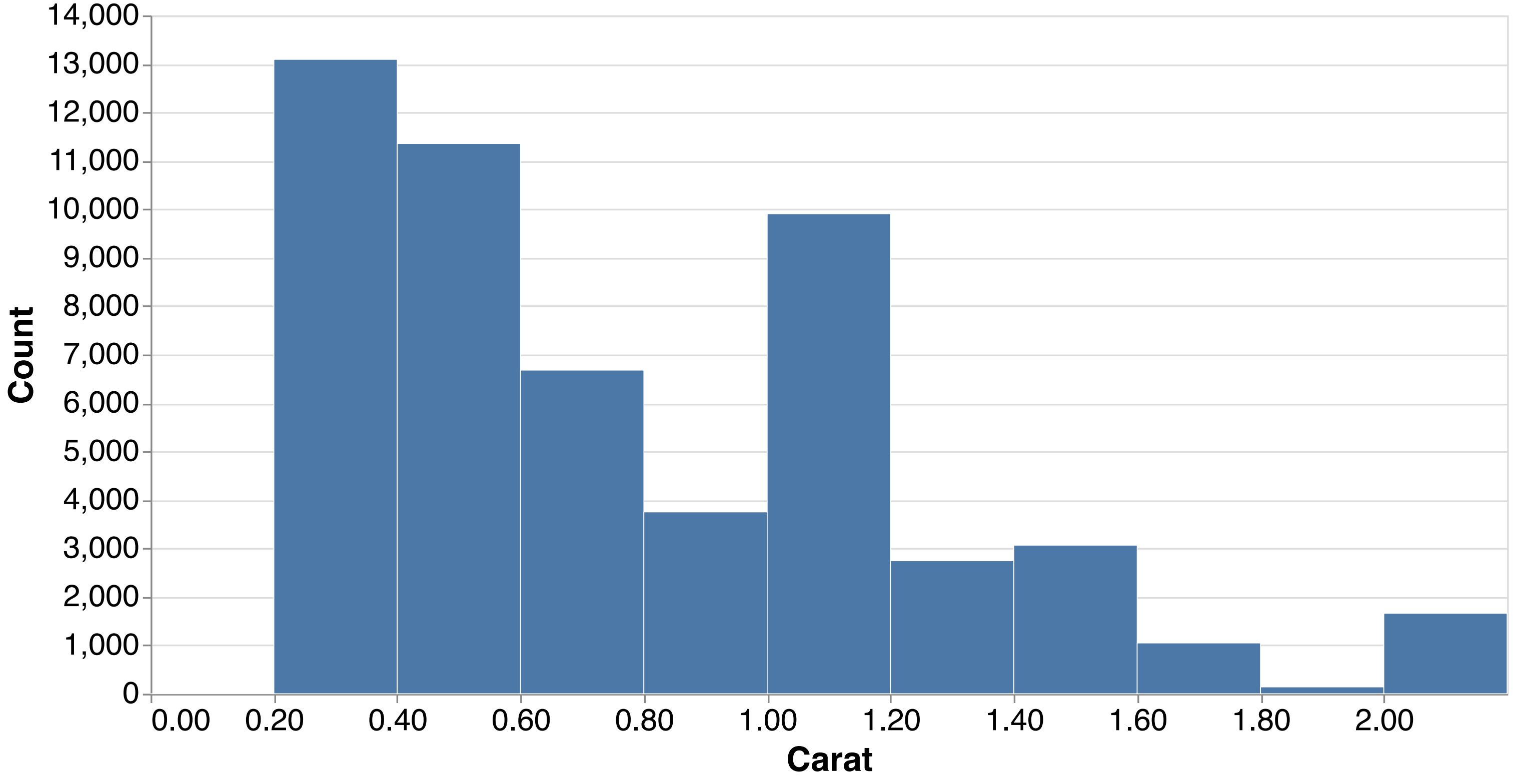
```
alt.data_transformers.disable_max_rows()  #disable 5k max rows

alt.Chart(diamonds).mark_bar().encode(
    alt.X('carat', bin=alt.Bin(maxbins=10), title = "Carat"),
    alt.Y('count()', title = "Count")
)
```



First plot iteration reveals most of sample is < 2

Histogram of carat

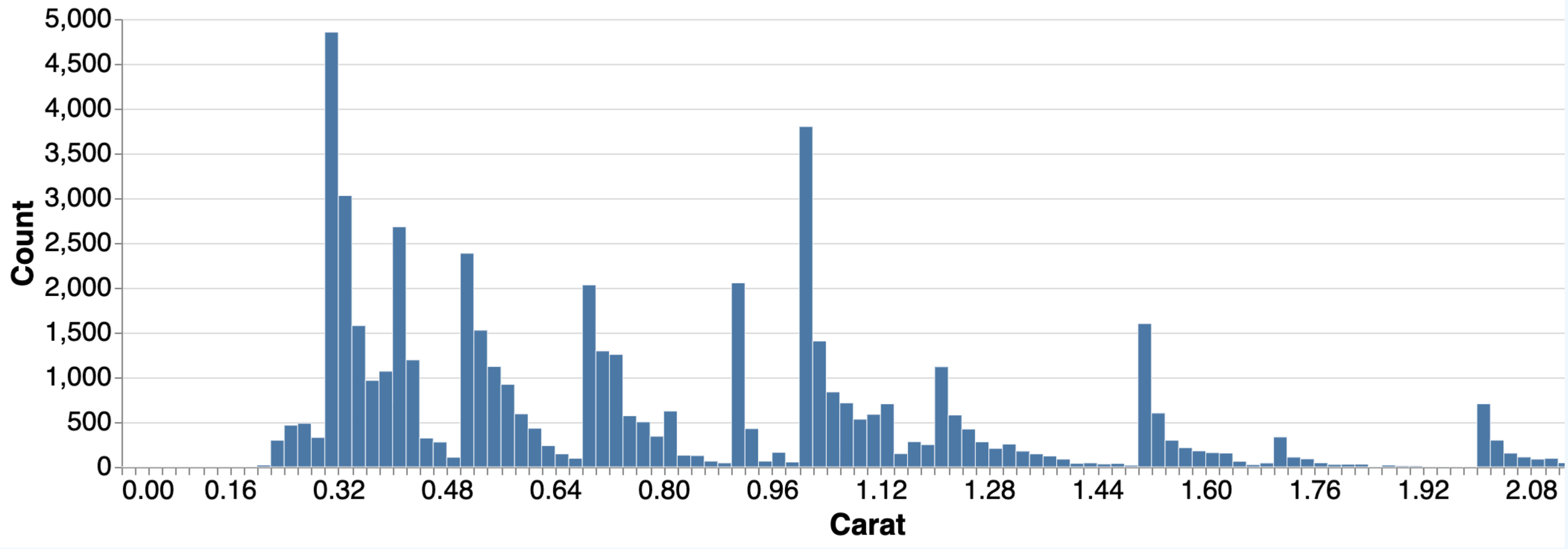
```
diamonds_small = diamonds.loc[diamonds['carat'] < 2.1]

alt.Chart(diamonds_small).mark_bar().encode(
    alt.X('carat', bin=alt.BinParams(step=0.2), title = "Carat"),
    alt.Y('count()', title = "Count")
)
```



Second plot iteration reveals count is not *entirely* decreasing in carat

In-class exercise: histogram of `carat`

```
alt.Chart(diamonds_small).mark_bar().encode(
    alt.X('carat', bin=alt.BinParams(step=0.02), title = "Carat"),
    alt.Y('count()', title = "Count"))
```



Discussion questions

1. What is the headline of the 3rd plot iteration? Submessages?
2. What questions does it raise?

Typical continuous variables: summary

- Main tool to explore uni-dimensional continuous variables: histograms
- Varying the bin widths can reveal different patterns

Continuous variables: unusual values

Unusual continuous variables: roadmap

- case study: y dimension in diamonds
  - explore some unusual values
  - three options for handling unusual values

diamonds: identify unusual y values

First pass to examine for unusual values: summary statistics

```python
diamonds['y'].describe()
```

```
count    53940.000000
mean         5.734526
std          1.142135
min          0.000000
25%          4.720000
50%          5.710000
75%          6.540000
max         58.900000
Name: y, dtype: float64
```

diamonds: examine unusual y values

```
diamonds.loc[(diamonds['y'] < 3) ]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
11963	1.00	Very Good	H	VS2	63.3	53.0	5139	0.0	0.0	0.0
15951	1.14	Fair	G	VS1	57.5	67.0	6381	0.0	0.0	0.0
24520	1.56	Ideal	G	VS2	62.2	54.0	12800	0.0	0.0	0.0
26243	1.20	Premium	D	VVS1	62.1	59.0	15686	0.0	0.0	0.0
27429	2.25	Premium	H	SI2	62.8	59.0	18034	0.0	0.0	0.0
49556	0.71	Good	F	SI2	64.1	60.0	2130	0.0	0.0	0.0
49557	0.71	Good	F	SI2	64.1	60.0	2130	0.0	0.0	0.0

diamonds: examine unusual y values

```
diamonds.loc[(diamonds['y'] > 20)]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
24067	2.00	Premium	H	SI2	58.9	57.0	12210	8.09	58.9	8.06
49189	0.51	Ideal	E	VS1	61.8	55.0	2075	5.15	31.8	5.12

diamonds: compare to 10 random obs

```
diamonds.sample(n=10)
```

	carat	cut	color	clarity	depth	table	price	x	y	z
26918	2.04	Premium	I	SI1	62.2	57.0	16942	8.14	8.07	5.04
14235	1.02	Ideal	I	VVS2	62.1	54.0	5767	6.47	6.51	4.03
41123	0.60	Premium	D	SI2	62.0	57.0	1196	5.43	5.35	3.34
19728	1.24	Ideal	F	VS2	62.2	56.0	8298	6.94	6.88	4.30
24862	2.50	Fair	H	SI2	64.9	58.0	13278	8.46	8.43	5.48
36766	0.30	Ideal	E	VVS1	60.1	57.0	956	4.41	4.38	2.64
10416	1.00	Very Good	E	SI2	62.9	60.0	4781	6.30	6.35	3.98
5120	0.91	Very Good	H	SI1	62.7	56.0	3762	6.14	6.18	3.86
47558	0.70	Ideal	I	SI2	62.3	55.0	1874	5.67	5.72	3.55
46565	0.53	Ideal	D	VS2	60.9	57.0	1783	5.17	5.24	3.17

What to do with unusual values?

1. Drop row
2. Code value to `NA`
3. Winsorize value

Option 1: drop rows

```
diamonds_clean = diamonds.loc[(diamonds['y'] >= 3) | (diamonds['y'] <= 20)]
diamonds_clean
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

53940 rows × 10 columns

Option 2: recode to missing

```
diamonds_missing = diamonds.copy()
diamonds_missing['y'] = np.where((diamonds_missing['y'] < 3) |
     (diamonds_missing['y'] > 20),
     np.nan, diamonds_missing['y'])
diamonds_missing[diamonds_missing['y'].isna()]
```

	carat	cut	color	clarity	depth	table	price	x	y	z
11963	1.00	Very Good	H	VS2	63.3	53.0	5139	0.00	NaN	0.00
15951	1.14	Fair	G	VS1	57.5	67.0	6381	0.00	NaN	0.00
24067	2.00	Premium	H	SI2	58.9	57.0	12210	8.09	NaN	8.06
24520	1.56	Ideal	G	VS2	62.2	54.0	12800	0.00	NaN	0.00
26243	1.20	Premium	D	VVS1	62.1	59.0	15686	0.00	NaN	0.00
27429	2.25	Premium	H	SI2	62.8	59.0	18034	0.00	NaN	0.00
49189	0.51	Ideal	E	VS1	61.8	55.0	2075	5.15	NaN	5.12
49556	0.71	Good	F	SI2	64.1	60.0	2130	0.00	NaN	0.00
49557	0.71	Good	F	SI2	64.1	60.0	2130	0.00	NaN	0.00

Option 3: winsorize

Winsorizing re-codes outliers to a numeric value, keeping them in the data.

To winsorize at 1 percent:

- Replace anything less than the 1st percentile with the 1st percentile
- Replace anything more than the 99th percentile with the 99th percentile

Option 3: winsorize

```python
diamonds_winsor = diamonds.copy()
pctile01 = diamonds_winsor['y'].quantile(0.01)
pctile99 = diamonds_winsor['y'].quantile(0.99)

print(f"1st Percentile: {pctile01}")
print(f"99th Percentile: {pctile99}")
```

```
1st Percentile: 4.04
99th Percentile: 8.34
```

Option 3: winsorize

```
diamonds_winsor['y_winsor'] = np.where(diamonds_winsor['y'] < pctile01, pct
                                np.where(diamonds_winsor['y'] > pctile99, p
                                diamonds_winsor['y']))
diamonds_winsor
```

	carat	cut	color	clarity	depth	table	price	x	y	z	y_winsor
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	4.04
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	4.04
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31	4.07
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	4.23
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	4.35
...	...	...	...	...	...	...	...	...	...	...	...
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50	5.76
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61	5.75
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56	5.68
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74	6.12
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64	5.87

53940 rows × 11 columns

When might you winsorize?

An example from Earnings Instability paper by Ganong and coauthors.

The paper is trying to quantify how much earnings change from month to month for the typical US worker.

Consider the following fake data (next slide)

Toy winsorization example

Suppose we have observations for earnings changes. 99% of the data follows a normal distribution with std. dev. 0.2 and 1% of the data is extremely large changes

last month ($)	this month ($)	% change	|% change |
600	600	0%	0%
600	570	-5%	5%
600	540	-10%	10%
600	630	5%	5%
…			
(99% of sample)			
…			
600	300	-50%	50%
6000	300	-95%	95%
300	600	100%	100%
300	6000	1900%	1900%

What is the standard deviation of the % change in earnings?

assumption	SD
do not winsorize	97.2%
winsorize at 50%	20.5%

Illustrative calculation here

When else is this useful? Income data, test scores, stock returns.

Real-world winsorization example

Table A-2: Earnings Volatility Under Different Winsorization Choices

Specification	Lower bound	Upper bound	Std. dev.
Winsorize top/bottom 1% of all changes	-0.66	1.83	31%
Winsorize top/bottom 0.1% of nonzero changes	-0.93	13	60%
Winsorize top/bottom 0.5% of nonzero changes	-0.81	4	41%
Winsorize top/bottom 1% of nonzero changes	-0.71	2.39	34%
Winsorize top/bottom 5% of nonzero changes	-0.42	0.71	22%
Winsorize changes larger than 50%	Bottom 2% of data	Top 5% of data	20%

Notes: The variable is the percent change in pay.

Source: Table A-2 from Earnings Instability paper

Pros + cons of each option

- **Dropping the observation**
  - Does not manipulate the data values
  - But can’t use that observation *at all* in your analysis
- **Recoding to missing**
  - Manipulates the data values
  - Allows you to use that observation – just not *that variable* – in your analysis
- **Winsorizing**
  - Manipulates the data values
  - Allows you to use that observation + that variable in your analysis

`diamonds`: what would you do?

- What would you do where `x`, `y`, and `z` are all 0?
- What would you do where `y > 20`?

`diamonds`: what would we do?

There is often not a “right” answer or you won’t know the answer without talking to a data provider.

Our best guesses:

- Rows where `x`, `y`, and `z` are all zero: set to `NA`
- Rows where `y > 20`: winsorize? (hard to know for sure…)

Unusual continuous values: summary

Problem	Action
Erroneous row	drop row
Erroneous cell	set to NA or winsorize

How do I decide which problem I have? Examine unusual values in context of other columns (same row) and other rows (same columns).

How do I decide whether to set to `NA` or winsorize? Ideally, ask your data provider what's going on with these values.

Unusual values case study

Unusual values case study: roadmap

- Introduce `mpg` dataset
- Research question 1

> What is the relationship between engine size and gas mileage?”

- Research question 2

> Why do some cars have better than typical mileage?

- Ad hoc identification of outliers
- Inspect fields describing outliers
- Uncover pattern

Introducing the `mpg` dataset

- `manufacturer` — car maker (e.g., toyota, ford)
- `model` — specific model name
- `displ` — engine size (liters)
- `hwy` — gas mileage highway miles per gallon
- `class` — vehicle class (compact, suv, pickup, etc.)

Introducing the `mpg` dataset

	manufacturer	model	displ	hwy	class
0	audi	a4	1.8	29	compact
1	audi	a4	1.8	29	compact
2	audi	a4	2.0	31	compact
3	audi	a4	2.0	30	compact
4	audi	a4	2.8	26	compact
...	...	...	...	...	...
229	volkswagen	passat	2.0	28	midsize
230	volkswagen	passat	2.0	29	midsize

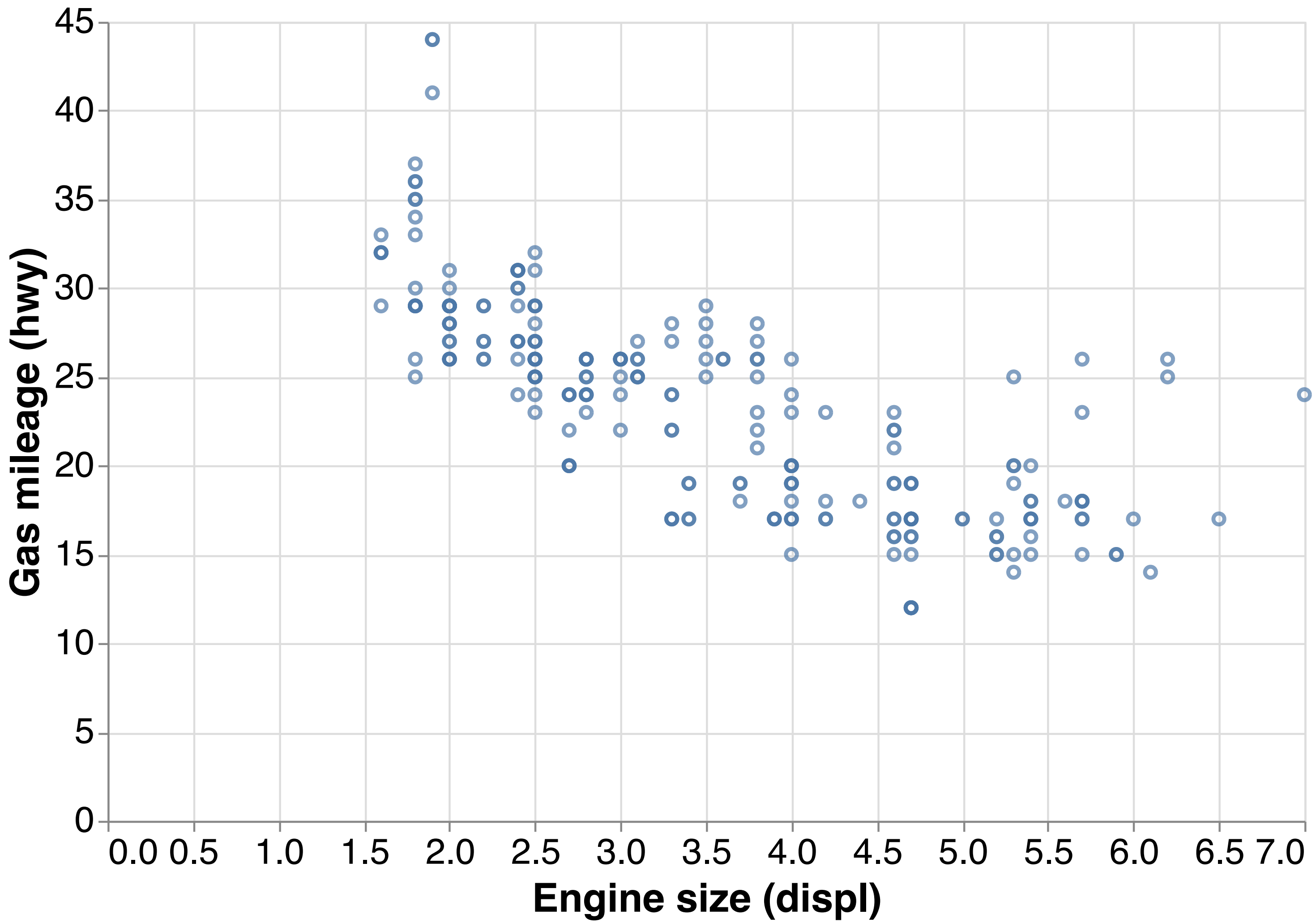
	manufacturer	model	displ	hwy	class
231	volkswagen	passat	2.8	26	midsize
232	volkswagen	passat	2.8	26	midsize
233	volkswagen	passat	3.6	26	midsize

234 rows × 5 columns

Q1: What is the relationship between engine size and gas mileage?

```
base = alt.Chart(mpg).mark_point().encode(
    alt.X('displ:Q', title = "Engine size (displ)"),
    alt.Y('hwy:Q', title = "Gas mileage (hwy)")
)
base
```

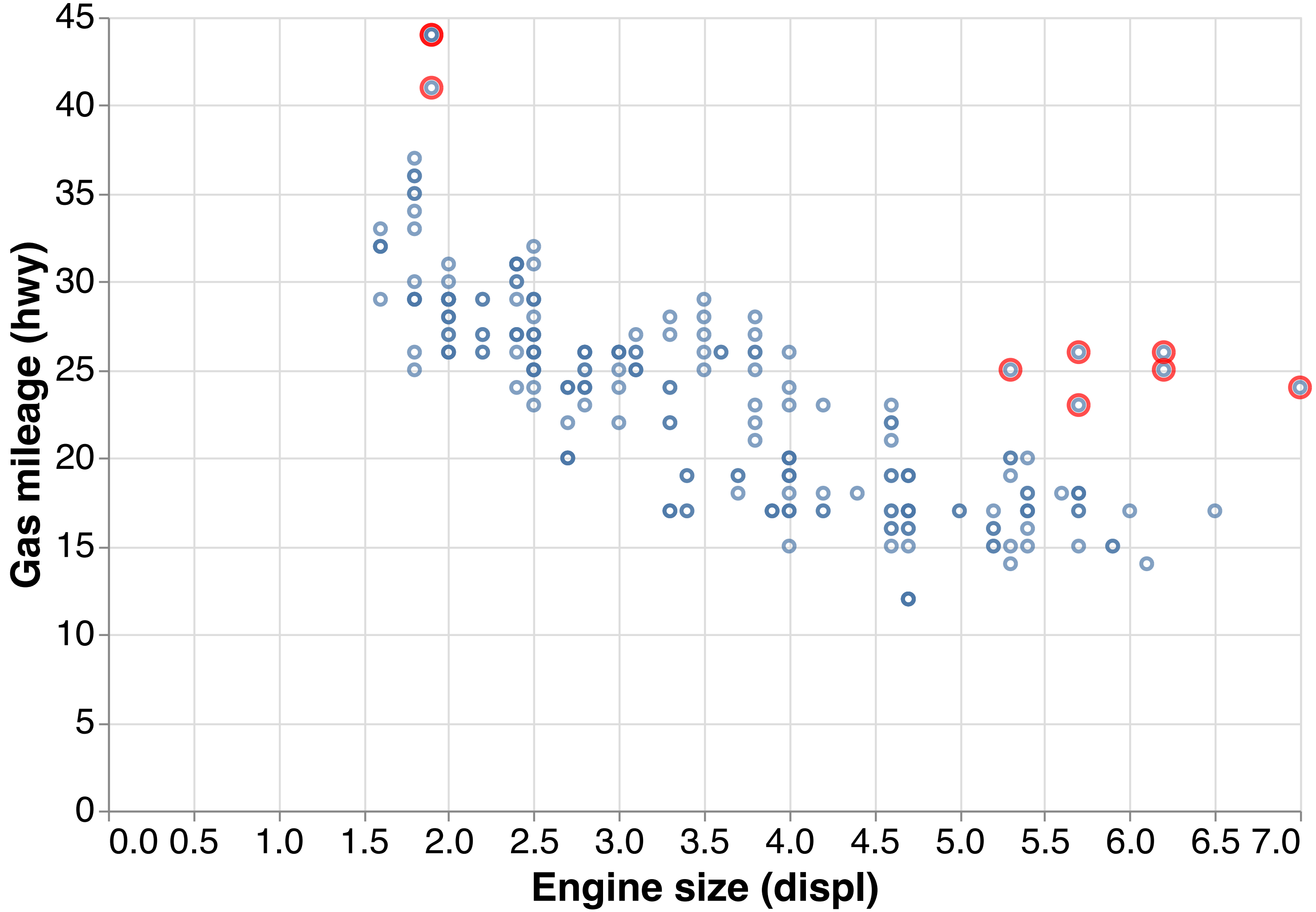
Q1: What is the relationship between engine size and gas mileage?



Q2: Why do some cars have better than typical mileage?

```
potential_outliers = mpg.loc[(mpg["hwy"] > 40)
    | ((mpg["hwy"] > 20)
    & (mpg["displ"] > 5))]
```

Q2: Why do some cars have better than typical mileage? (plot)



Q2: In-class exercise (table)

Discussion q – which fields do you want to study further on the plot (and why?)

```
['manufacturer', 'model', 'displ', 'year', 'cyl', 'trans', 'drv', 'cty',
'hwy', 'fl', 'class']
```

	manufacturer	model	displ	year	cyl	tr
23	chevrolet	corvette	5.7	1999	8	manual(
24	chevrolet	corvette	5.7	1999	8	auto(l4)
25	chevrolet	corvette	6.2	2008	8	manual(
26	chevrolet	corvette	6.2	2008	8	auto(s6)
27	chevrolet	corvette	7.0	2008	8	manual(

	manufacturer	model	displ	year	cyl	tr
158	pontiac	grand prix	5.3	2008	8	auto(s4)
212	volkswagen	jetta	1.9	1999	4	manual(
221	volkswagen	new beetle	1.9	1999	4	manual(
222	volkswagen	new beetle	1.9	1999	4	auto(l4)

Let's focus on two fields

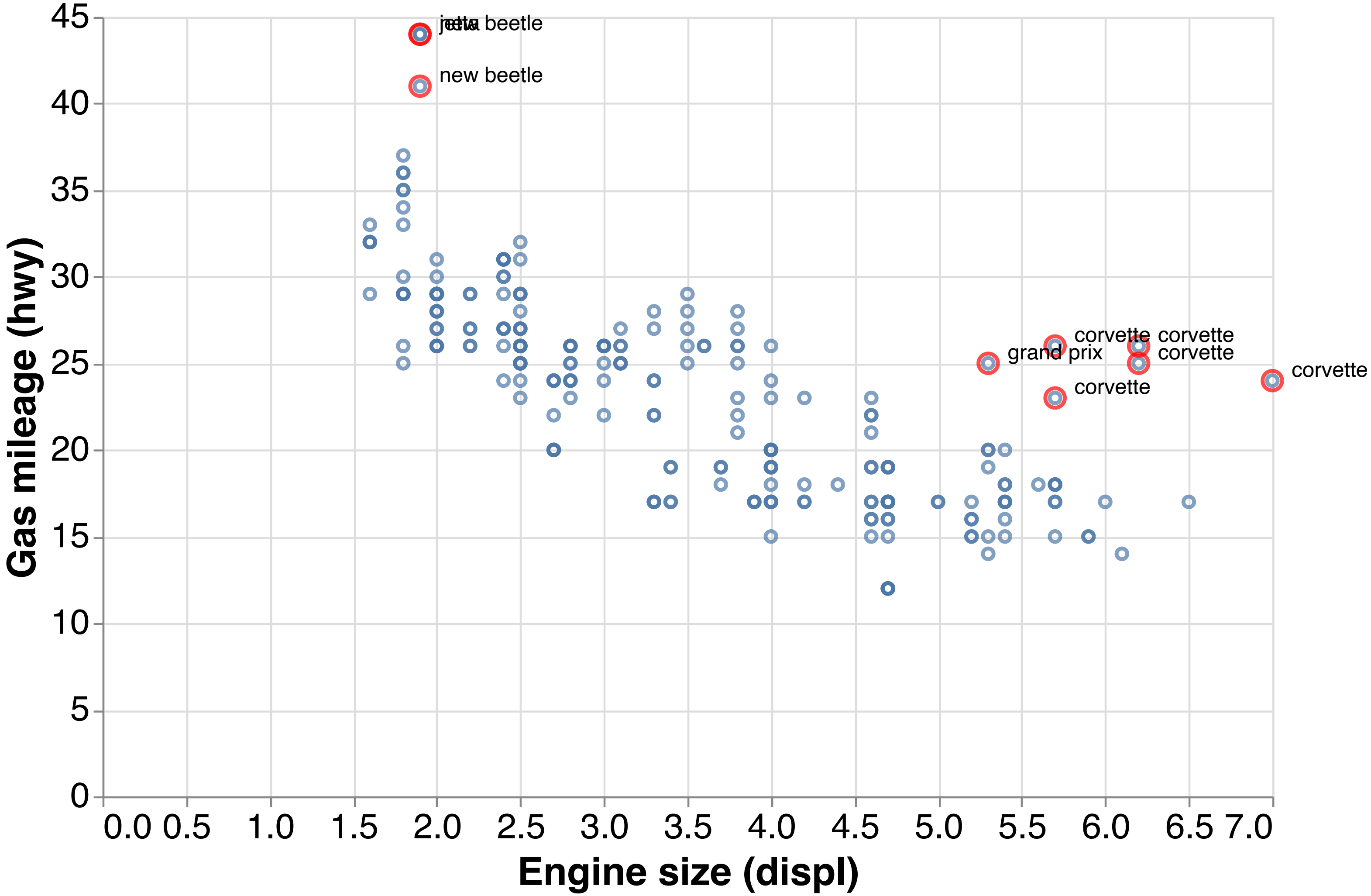
Fields

1. `model`
2. `class`

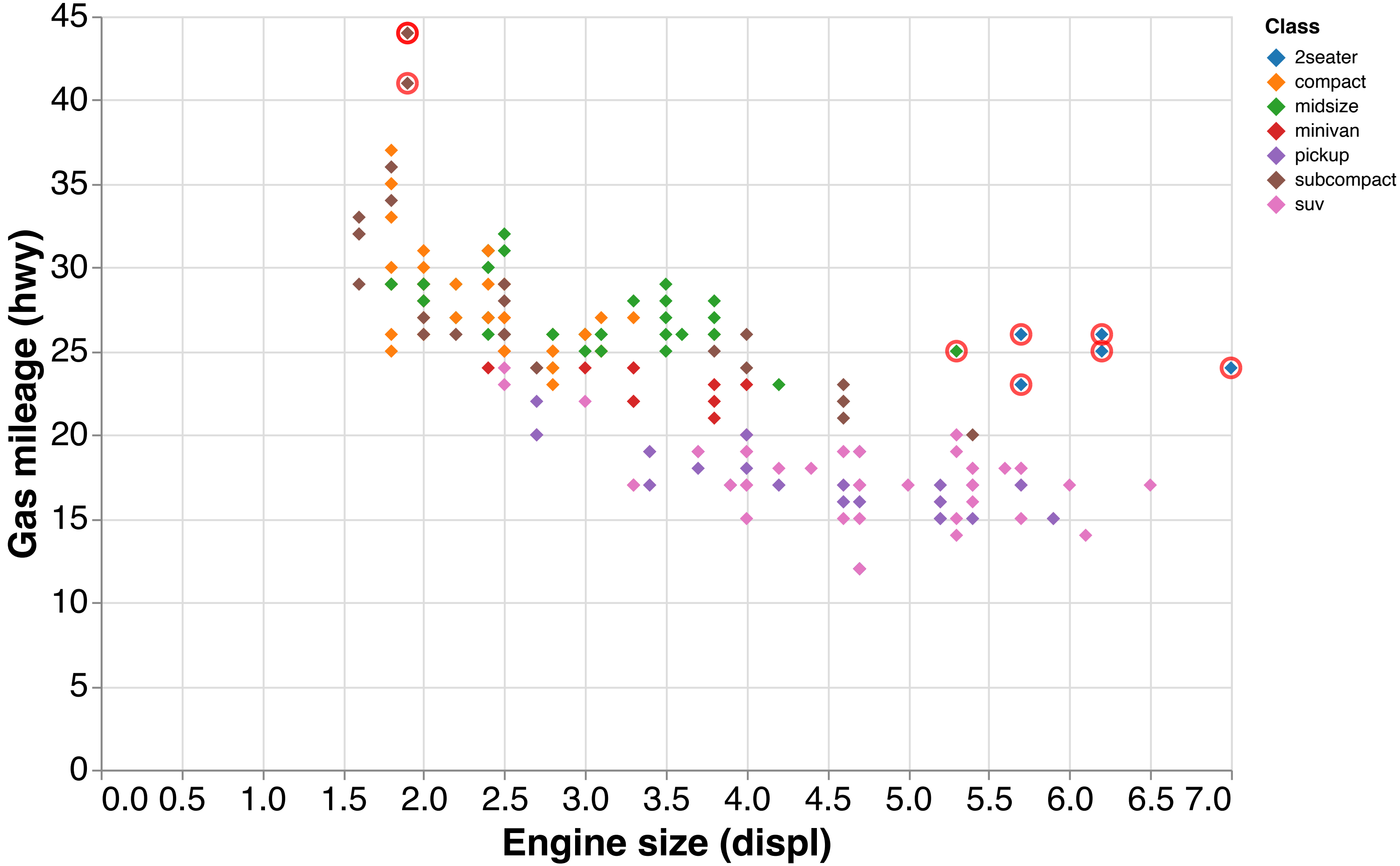
How did I know to use these? Context knowledge about different types of cars.

Don't have context knowledge about your dataset? Use LLM/Google/human subject matter expert to help you identify patterns

Q2: Why do some cars have better than typical mileage?



Q: How are there big engines and good mileage? `color`



Discussion

- We applied labels to each outlier car using the `model` field
- We hypothesized that the field `class` would capture what these models would have in common
- This is the elegant corner case where one variable (`class`) explains many of the outlier patterns in terms of fuel-efficient cars
  - All the large engines on the top-right are `class` == "2seater"
  - Both of the small engines on the top-left are `class` == "subcompact" (but many subcompacts are less fuel efficient)
- Caveat: most datasets are not as clean as this example is, but we've chosen this example for instructional purposes

Unusual values case study: summary

Research question

> Why do some cars have better than typical mileage? (What's going on with these outliers?)

What did we do?

1. We identified outliers by hand
2. We looked at variables for those outliers
3. We went back to the plot with a variable which we thought could provide a unified explanation for the outliers.
4. It mostly did (subcompact cars and 2 seater cars are both very fuel efficient)