Multivariate Data and the Grammar of Graphics

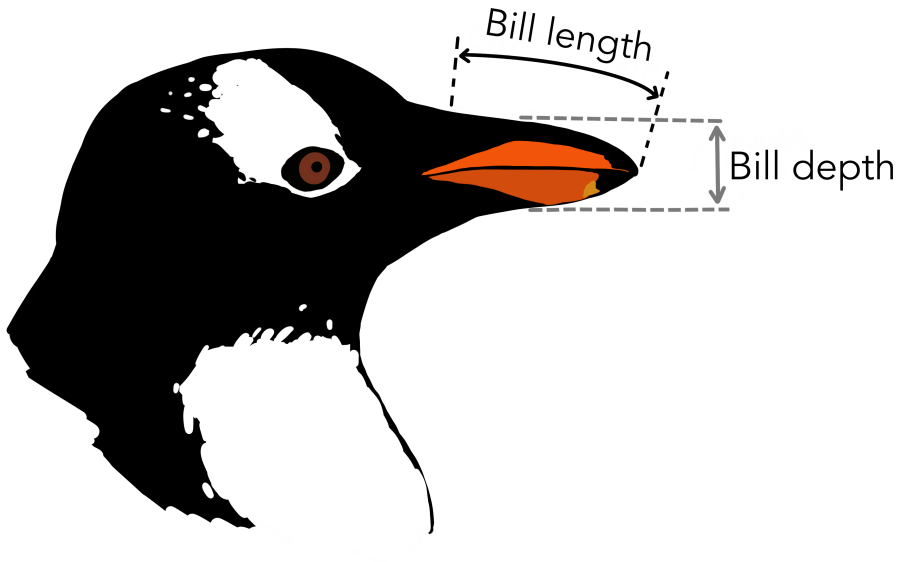
Dennis Sun
Stanford University
DATASCI / STATS 112

January 20, 2023

1 Penguins!

2 Visualizing Multivariate Data

Palmer Penguins



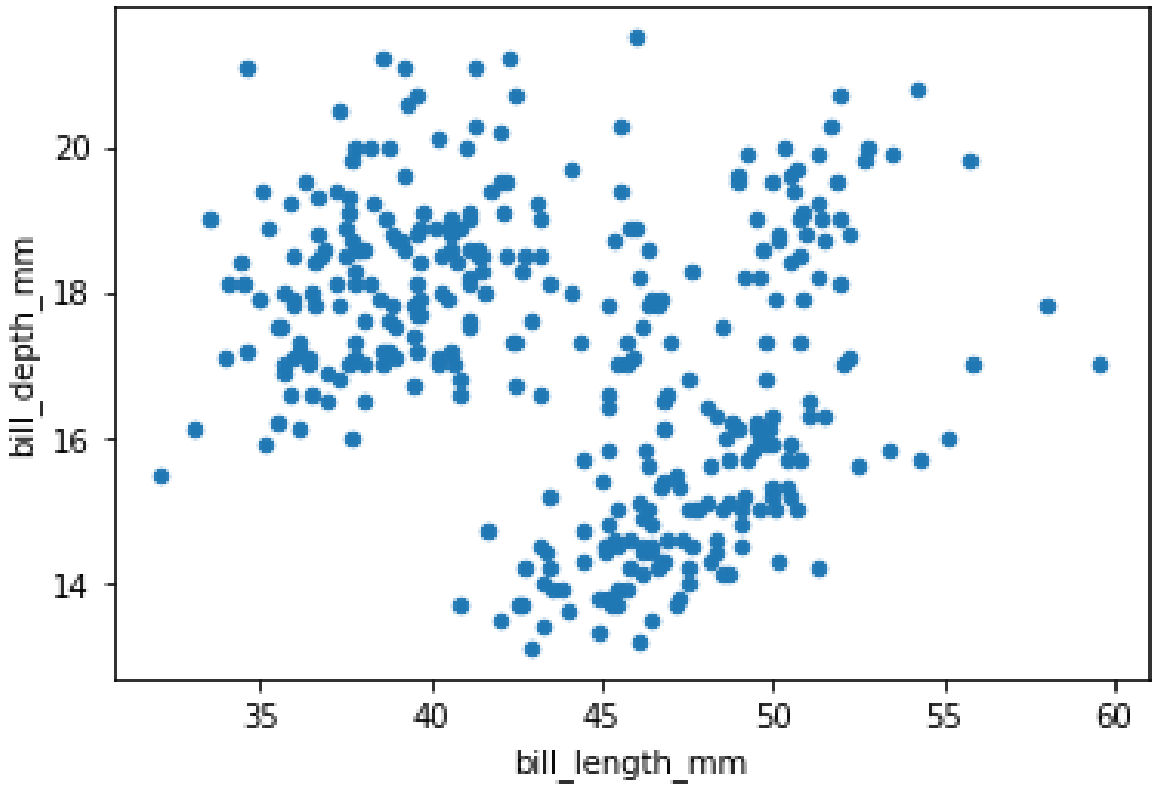
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007
...	...	...	...	...	...	...	...	...
339	Chinstrap	Dream	55.8	19.8	207.0	4000.0	male	2009
340	Chinstrap	Dream	43.5	18.1	202.0	3400.0	female	2009
341	Chinstrap	Dream	49.6	18.2	193.0	3775.0	male	2009
342	Chinstrap	Dream	50.8	19.0	210.0	4100.0	male	2009
343	Chinstrap	Dream	50.2	18.7	198.0	3775.0	female	2009

344 rows × 8 columns

Bill Length vs. Bill Depth

To visualize the relationship between two quantitative variables, we make a scatterplot.

```
df_penguins.plot.scatter(x="bill_length_mm", y="bill_depth_mm")
```



Bill Length vs. Bill Depth

To summarize the relationship, we calculate the correlation coefficient R.

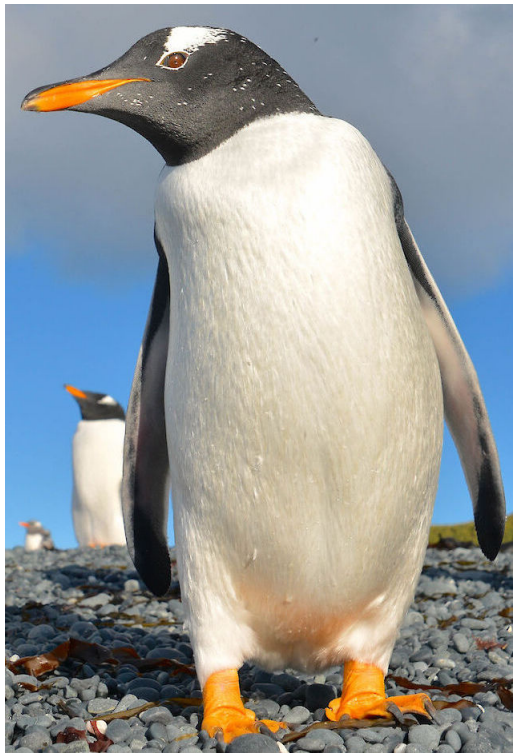
```
df_penguins[["bill_length_mm", "bill_depth_mm"]].corr()
```

	bill_depth_mm	bill_length_mm
bill_depth_mm	1.000000	-0.235053
bill_length_mm	-0.235053	1.000000

Adding Species

But wait! There are 3 penguin species.

Adelie



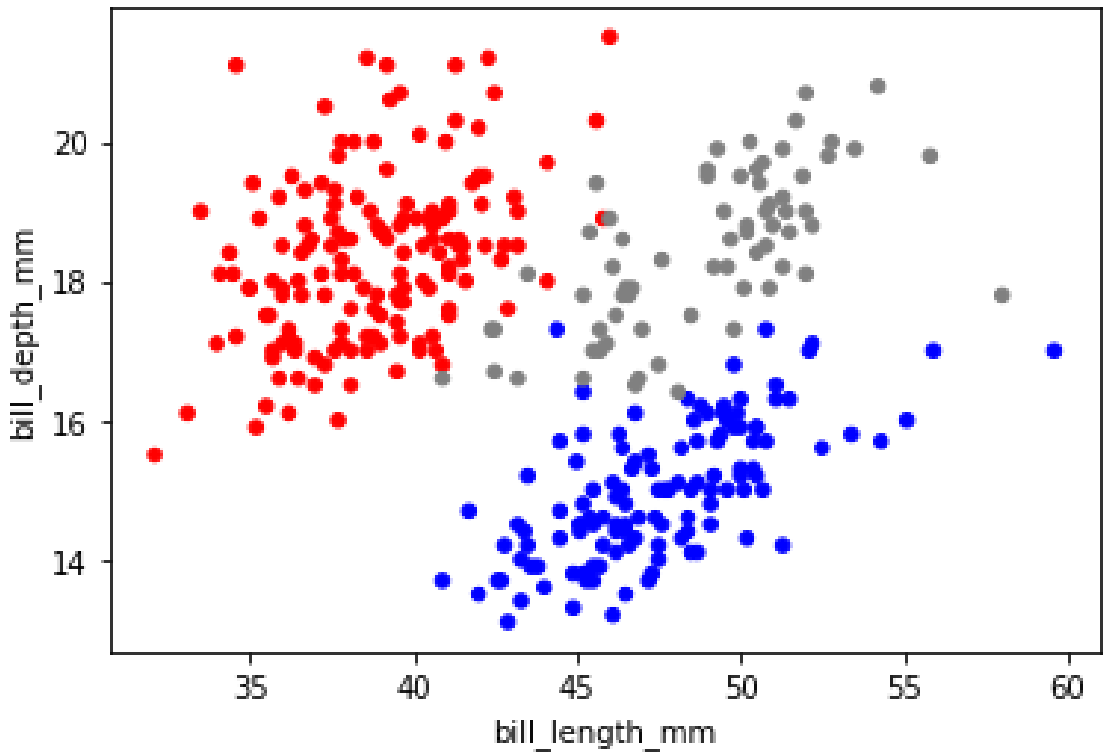
Gentoo

Chinstrap

How do we incorporate this third variable into our visualization?

We can represent the third dimension using color.

```
df_penguins.plot.scatter(
    x="bill_length_mm", y="bill_depth_mm",
    c=df_penguins["species"].map({
        "Adelie": "red",
        "Gentoo": "blue",
        "Chinstrap": "gray"
        })
)
```



This code is ugly! Pandas is good for visualizing 1 or 2 variables. It's not ideal for highly multivariate data.

Wait! Has the relationship between Bill length and depth changed?

Let's check the correlation between length and depth for each species.

```
df_penguins.groupby("species")[["bill_depth_mm",
                                "bill_length_mm"]].corr()
```

species		bill_depth_mm	bill_length_mm
Adelie	bill_depth_mm	1.000000	0.391492
	bill_length_mm	0.391492	1.000000
Chinstrap	bill_depth_mm	1.000000	0.653536
	bill_length_mm	0.653536	1.000000
Gentoo	bill_depth_mm	1.000000	0.643384
	bill_length_mm	0.643384	1.000000

But remember, the overall correlation between length and depth was negative!

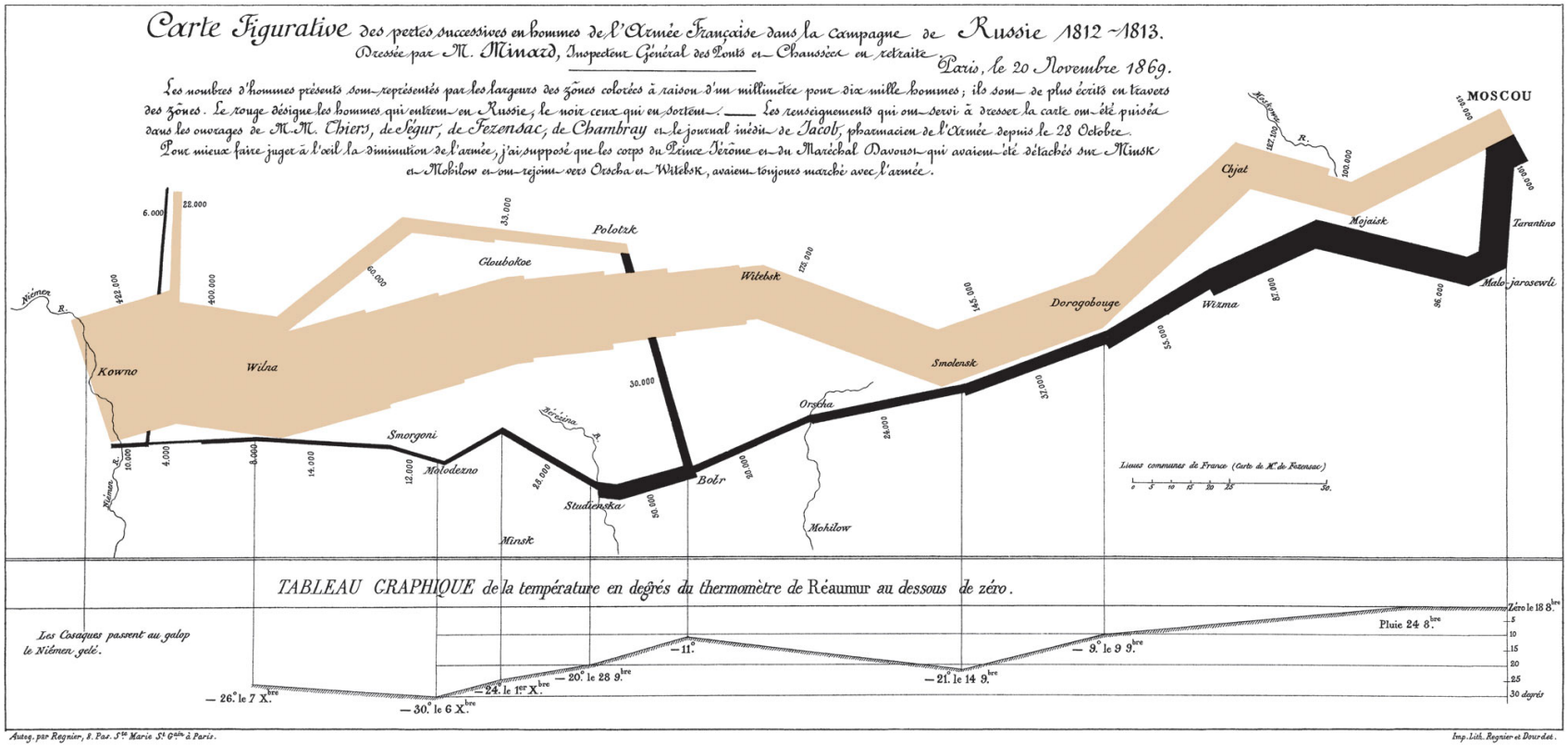
	bill_depth_mm	bill_length_mm
bill_depth_mm	1.000000	-0.235053
bill_length_mm	-0.235053	1.000000

Does this remind you of something we've seen before?

1 Penguins!

2 Visualizing Multivariate Data

Minard's Map of Napoleon's Russia Campaign

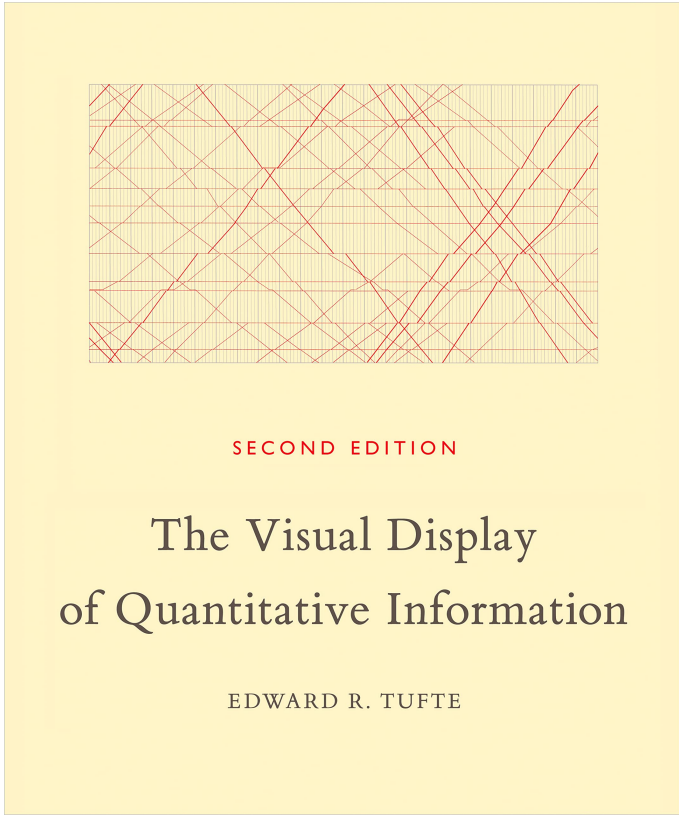


According to Edward Tufte, "it may well be the best statistical graphic ever drawn."

Tufte on Graphical Excellence

Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.

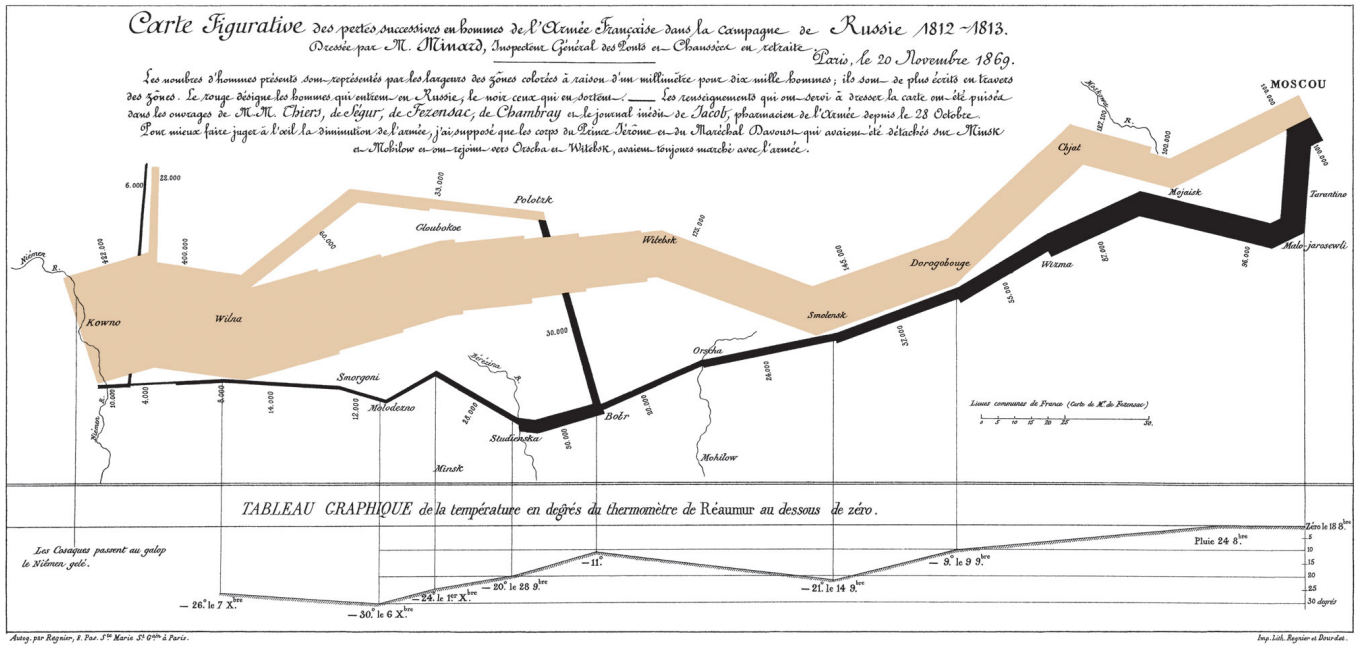
Graphical excellence is nearly always multivariate.



Aesthetic Mappings

How do we create data-dense graphics?

By mapping every dimension in the graphic (an “aesthetic”) to a dimension in the data.



- x ⟷ longitude
- y ⟷ latitude
- width ⟷ size of army

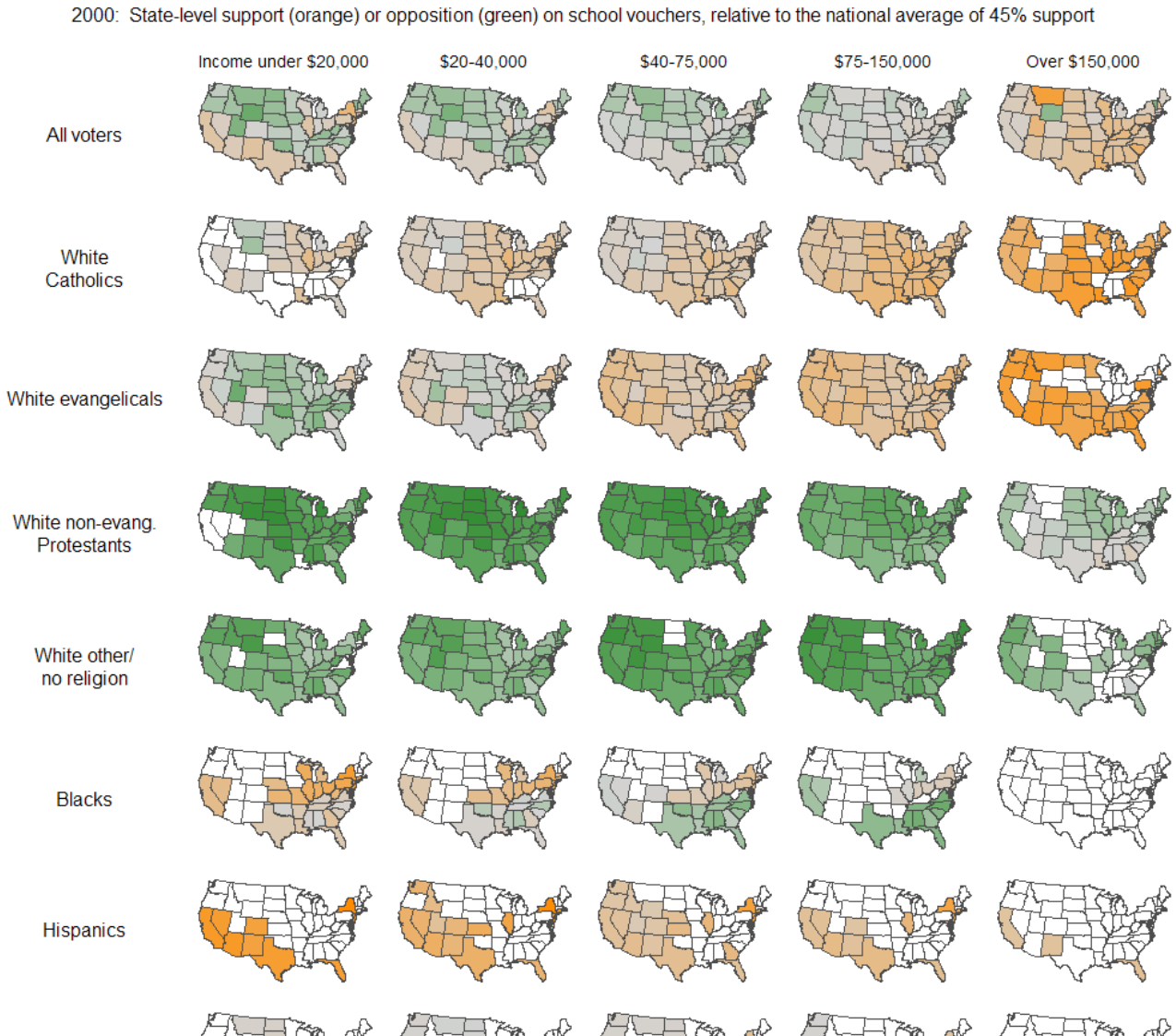
- color ⟷ direction of army
- y (line graph) ⟷ temperature
- x / text (line graph) ⟷ date

Aesthetics

Pay attention to which aesthetics are associated with quantitative variables and which are associated with categorical variables.

Facets

One way to pack more variables without overplotting is to show many small plots. (Tufte calls this “small multiples.”)



Source: Gelman

Grammar of Graphics

The **grammar of graphics** says that every plot can be described by just a few components:

- aesthetic mappings
- geometric objects (e.g., points, lines, bars)
- ...and a few other things.

We want plotting libraries that allow us to specify the plot we want by just specifying the aesthetic mappings and a geometric object.

Example: I want a line plot, where

- x $\longleftrightarrow$ longitude
- y $\longleftrightarrow$ latitude
- width $\longleftrightarrow$ size of army
- color $\longleftrightarrow$ direction of army

Libraries that do this include `ggplot2` in R and `plotly` in Python.