VISUALISING DATA IN R

OU24 Graduate Skills Class

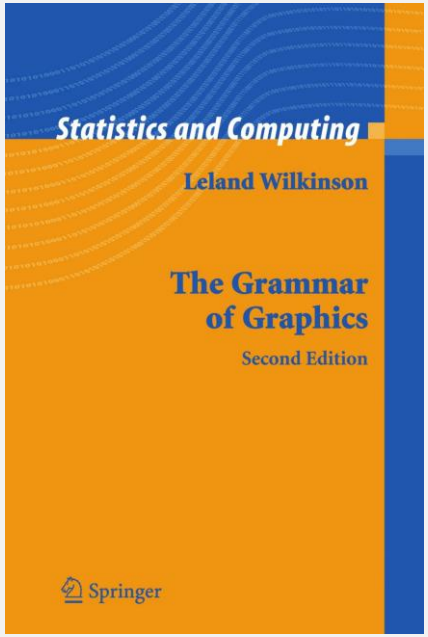
Damon Wischik

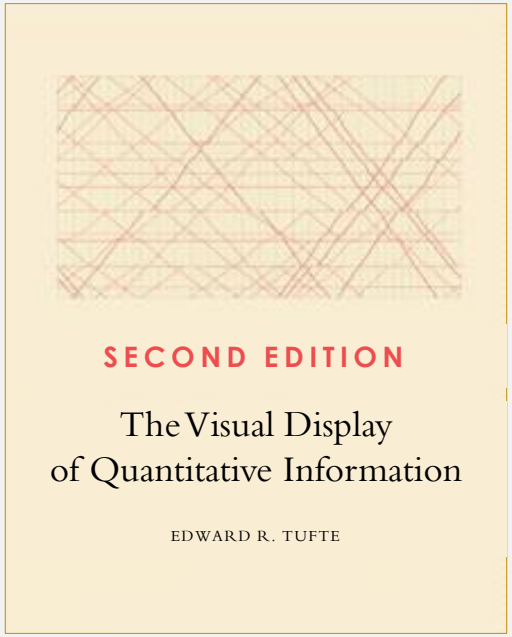
R's Grammar of Graphics codifies some standard patterns in plotting data. It will simplify your life — if you learn the way it thinks, and if you don't step outside its scope.

Lecture: high-level concepts in ggplot

Practical: how to actually use it

rhetoric = grammar + style + reason / arrangement

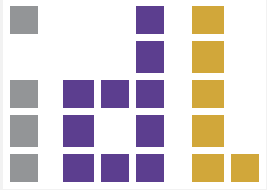




R + ggplot2

Javascript + D3

Vega Lite



and many many badly conceived libraries ...

matplotlib

First get Jupyter+Python+R up and running

Department of Computer Scienc

https://www.cl.cam.ac.uk/~djw1005/Teach/ggvis

UNIVERSITY OF CAMBRIDGE

Department of Computer Science a

Computer Laboratory › Teaching › Courses 2018–19 › OU24 Visualizing

Course pages 2018–19

Visualizing data with R

Course materials

Lecturer: Dr Damon Wischik
Taken by: Part III, ACS, PhD
Course admin: Research Skills unit OU24 [Moodle]

Lecture and practical class: Wednesday 30 January, 9–11am, Inte

Before the class: please set up Python+ggplot (or R+ggplot if yo
below. Any problems with setting up, please contact me before t

Aims

The R language is widely used by data scientists. It is concise and
powerful graphics library called ggplot. This course will show you
use ggplot for flexible and versatile data visualization. See examp

Resources

- Setting up Python+ggplot [notebook]
- Setting up R+ggplot [notebook]
- Slides (to come)
- Practical exercises (to come)

Home

ggplot-python

localhost:8888/notebooks/ggplot-python.ipynb

jupyter ggplot-python (autosaved)

Logout

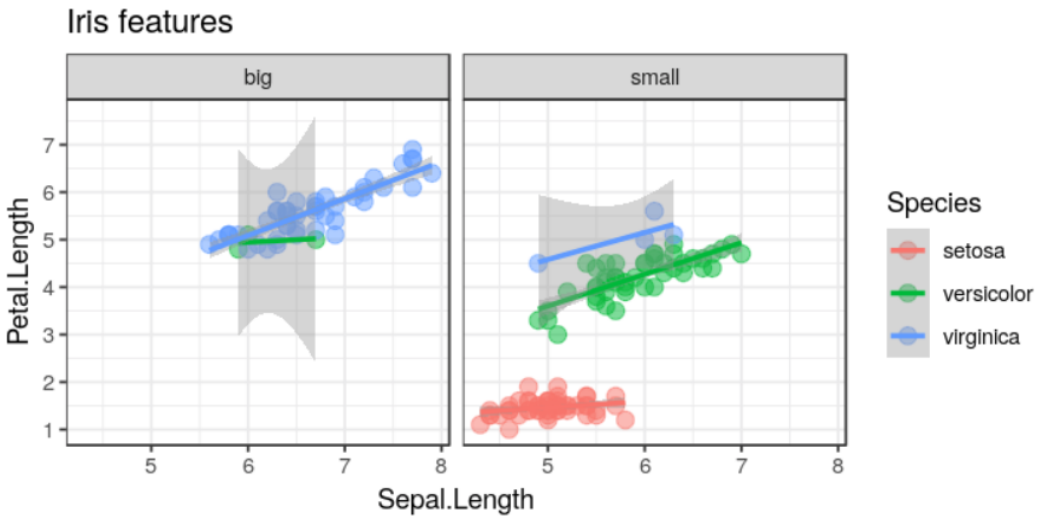
File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

You can run arbitrary R cells in the Jupyter notebook. If a line starts with `%R`, that line is run in R. If a cell starts with `%%R`, the entire cell is run in R. The `%%R` magic has some optional arguments, e.g. to set the size of the graphic that R might be asked to generate.

In [35]:

```
%%R -w6 -h3 -uin -r120

ggplot(data=iris) +
    geom_point(aes(x=Sepal.Length, y=Petal.Length, col=Species), size=3, alpha=
    stat_smooth(aes(x=Sepal.Length, y=Petal.Length, col=Species), method='lm')
    scale_y_continuous(breaks=seq(0,10)) +
    facet_wrap(~petal.size) +
    ggtitle('Iris features') +
    theme_bw()
```



data

stat

geom

aes

facet

position

coord

guides

Home

ggplot-python

localhost:8888/notebooks/ggplot-python.ipynb

jupyter ggplot-python (autosaved)

Logout

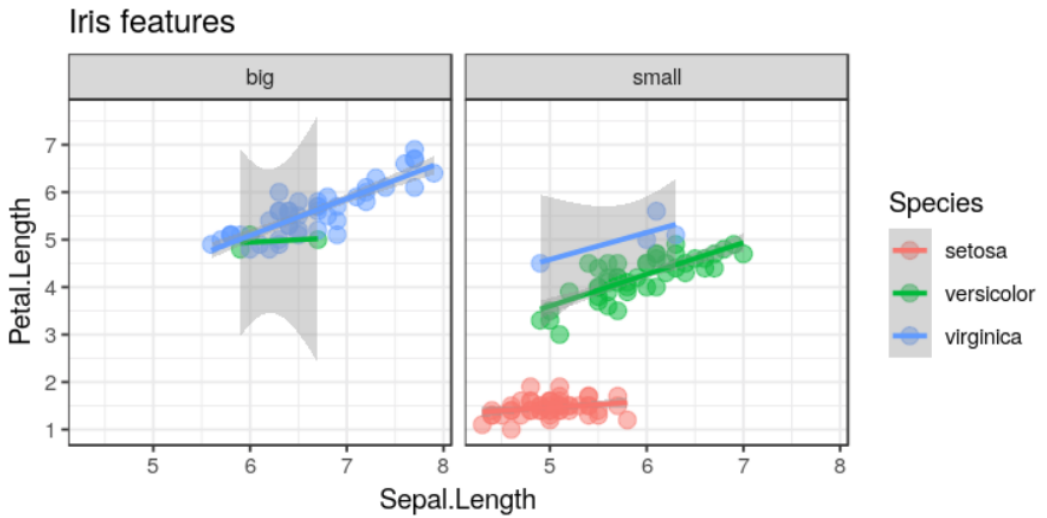
File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

You can run arbitrary R cells in the Jupyter notebook. If a line starts with `%R`, that line is run in R. If a cell starts with `%%R`, the entire cell is run in R. The `%%R` magic has some optional arguments, e.g. to set the size of the graphic that R might be asked to generate.

In [35]:

```
%%R -w6 -h3 -uin -r120

ggplot(data=iris) +
    geom_point(aes(x=Sepal.Length, y=Petal.Length, col=Species), size=3, alpha=
    stat_smooth(aes(x=Sepal.Length, y=Petal.Length, col=Species), method='lm')
    scale_y_continuous(breaks=seq(0,10)) +
    facet_wrap(~petal.size) +
    ggtitle('Iris features') +
    theme_bw()
```

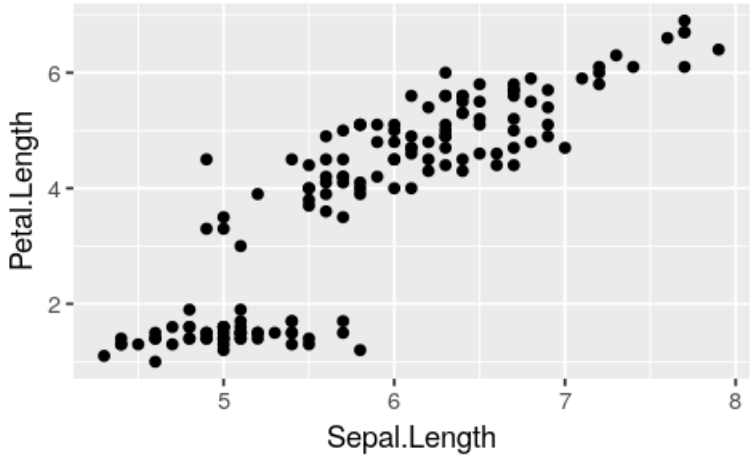


Data comes in data frames.

ggplot2 is only for this sort of data.

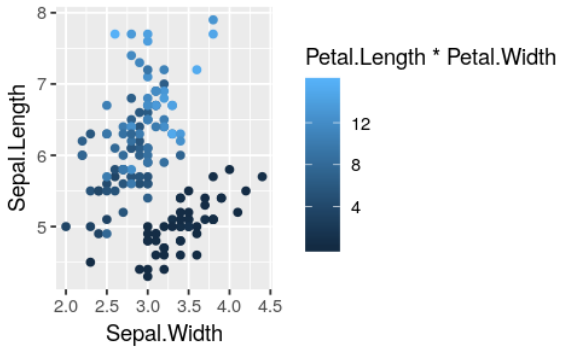
Sepal. Length	Sepal. Width	Petal. Length	Petal. Width	Species
5.0	3.4	1.6	0.4	setosa
6.5	3.0	5.5	1.8	virginica
5.0	3.5	1.3	0.3	setosa
6.7	2.5	5.8	1.8	virginica

```
ggplot(data=iris) +
    geom_point(aes(x=Sepal.Length, y=Petal.Length))
```

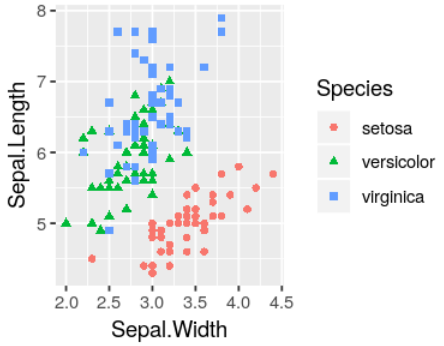


- The *aesthetic mapping* specifies which data columns should be mapped to which visual dimensions

```
ggplot(data=iris) +
    geom_point(aes(x=Sepal.Width,
                   y=Sepal.Length,
                   col=Petal.Length*Petal.Width))
```

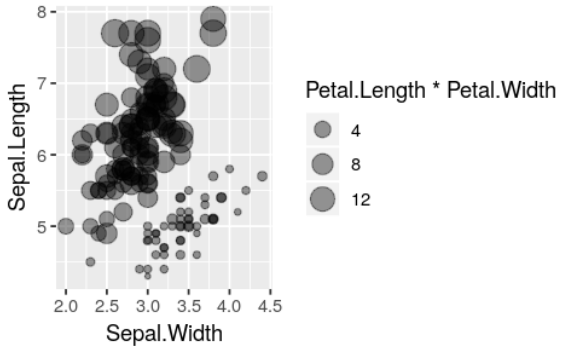


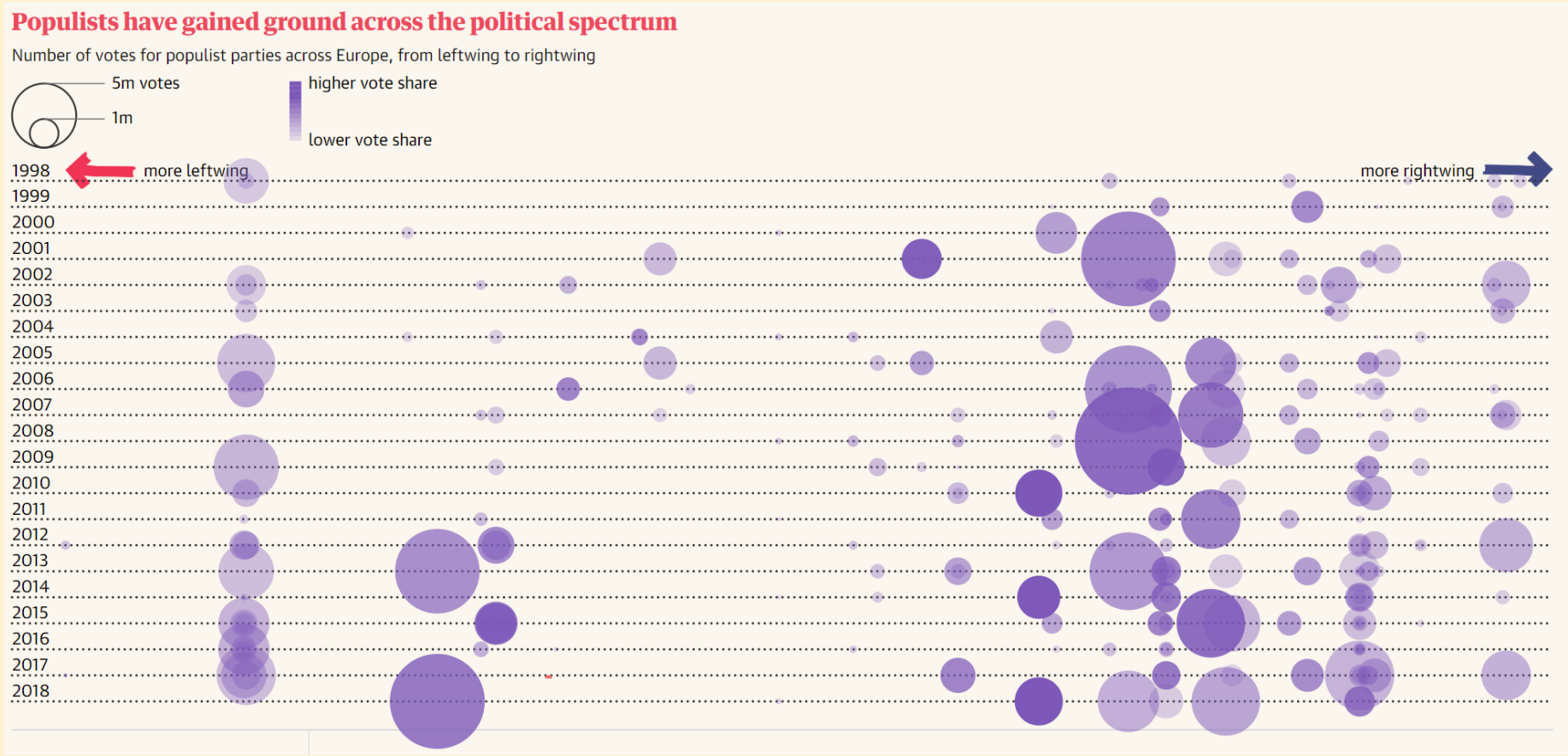

```
ggplot(data=iris) +
    geom_point(aes(x=Sepal.Width,
                   y=Sepal.Length,
                   col=Species,
                   shape=Species))
```



Species is a discrete (string) value, so the default scale is discrete.

```
ggplot(data=iris) +
    geom_point(aes(x=Sepal.Width,
                   y=Sepal.Length,
                   size=Petal.Length*Petal.Width),
               alpha=.4)
```



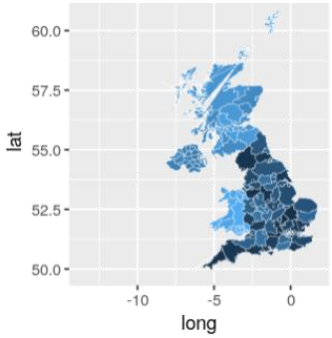


https://www.theguardian.com/world/ng-interactive/2018/nov/20/revealed-one-in-four-europeans-vote-populist

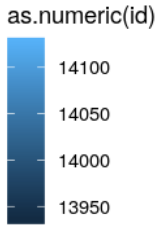
Exercise. What is the aesthetic mapping?

- The *aesthetic mapping* specifies which data columns should be mapped to which visual dimensions
- The entire range of data values is mapped onto the visual range, which can be configured with `scale_*`

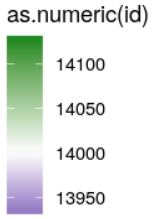
id	long	lat	order	hole	piece	group	id1	name1	name	type
14116	-4.624721	53.32681	412744	FALSE	2	14116.2	1033	Wales	Gwynedd	Unitary Authority (wales)
14116	-4.661944	53.31958	413897	FALSE	2	14116.2	1033	Wales	Gwynedd	Unitary Authority (wales)
13953	-3.113055	54.92708	27837	FALSE	1	13953.1	1030	England	Cumbria	Administrative County

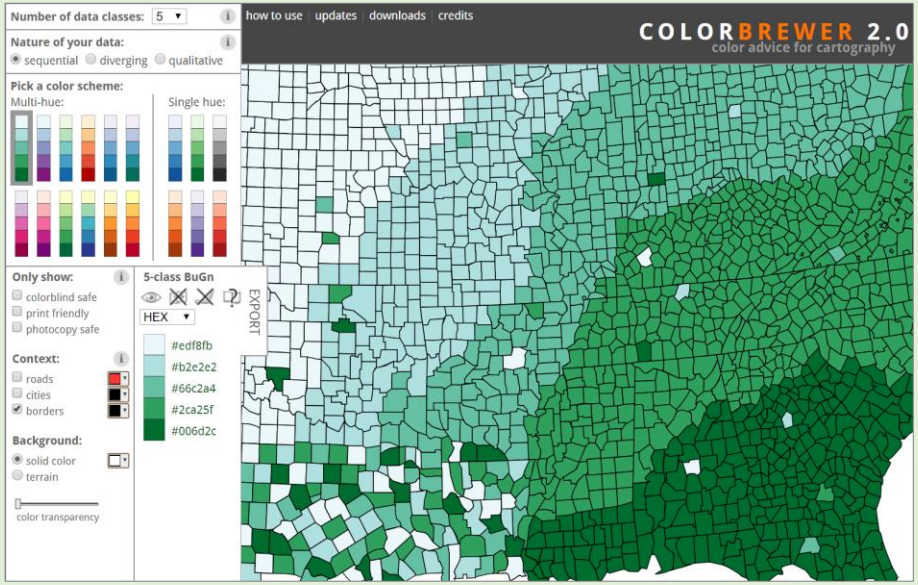


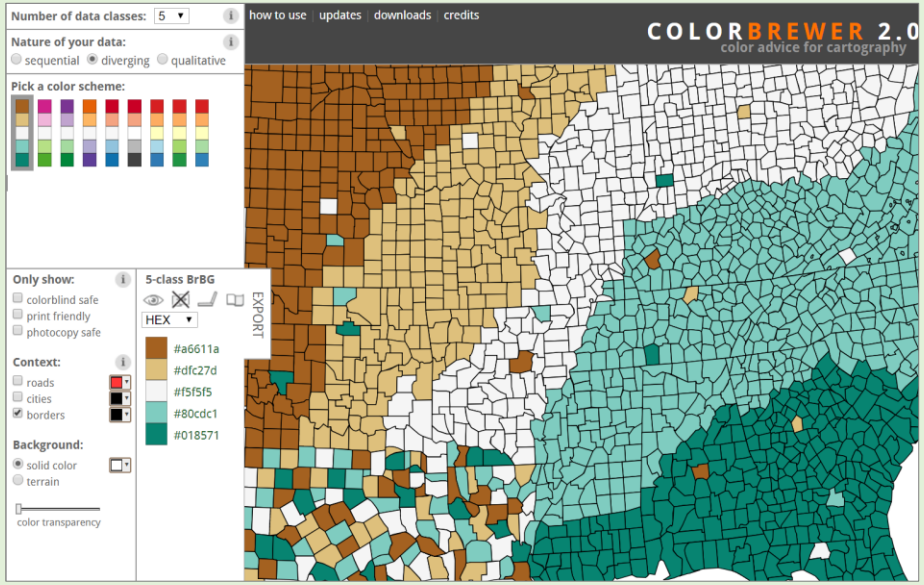

```
ukmap <- fread('https://teachingfiles.blob.core.windows.net/datasets/uk_poly.csv')

ggplot(data=ukmap) +
    geom_polygon(aes(x=long, y=lat, group=group, fill=as.numeric(id)), col='white', size=.1) +
    coord_fixed(ratio=1/cos(50*2*pi/360))
```

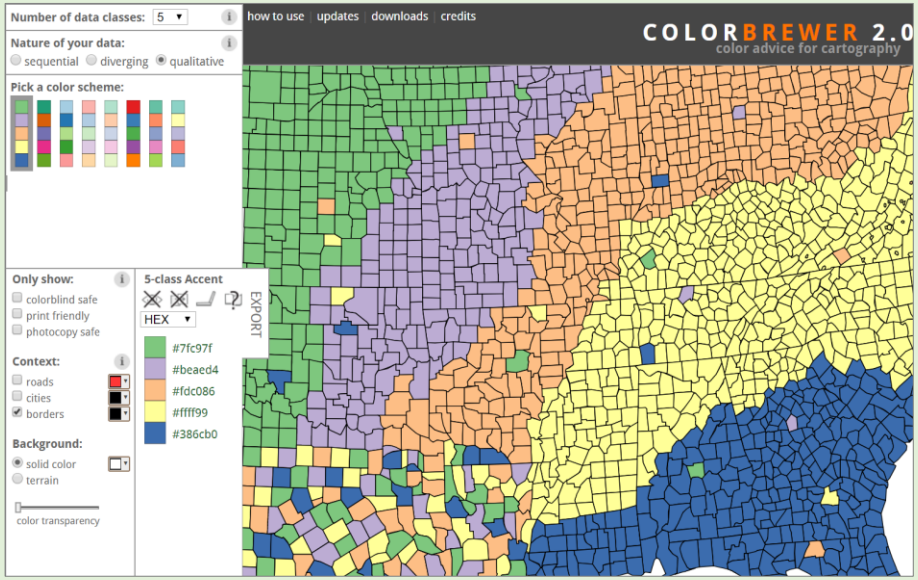



```
ggplot(data=ukmap) +
    geom_polygon(aes(x=long, y=lat, group=group, fill=as.numeric(id)), col='white', size=.1) +
    coord_fixed(ratio=1/cos(50*2*pi/360)) +
    scale_fill_gradient2(midpoint=14000, high='forestgreen', low='darkblue')
```









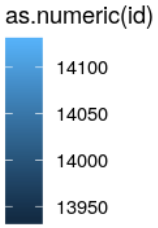
Source code and feedback

Color Brewer: sequential / diverging / qualitative scales, for discrete data

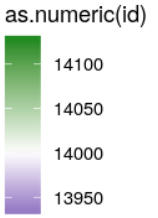
- The *aesthetic mapping* specifies which data columns should be mapped to which visual dimensions
- The entire range of data values is mapped onto the visual range, which can be configured with `scale_*`

id	long	lat	order	hole	piece	group	id1	name1	name	type
14116	-4.624721	53.32681	412744	FALSE	2	14116.2	1033	Wales	Gwynedd	Unitary Authority (wales)
14116	-4.661944	53.31958	413897	FALSE	2	14116.2	1033	Wales	Gwynedd	Unitary Authority (wales)
13953	-3.113055	54.92708	27837	FALSE	1	13953.1	1030	England	Cumbria	Administrative County


```
ukmap <- fread('https://teachingfiles.blob.core.windows.net/datasets/uk_poly.csv')

ggplot(data=ukmap) +
    geom_polygon(aes(x=long, y=lat, group=group, fill=as.numeric(id)), col='white', size=.1) +
    coord_fixed(ratio=1/cos(50*2*pi/360))
```



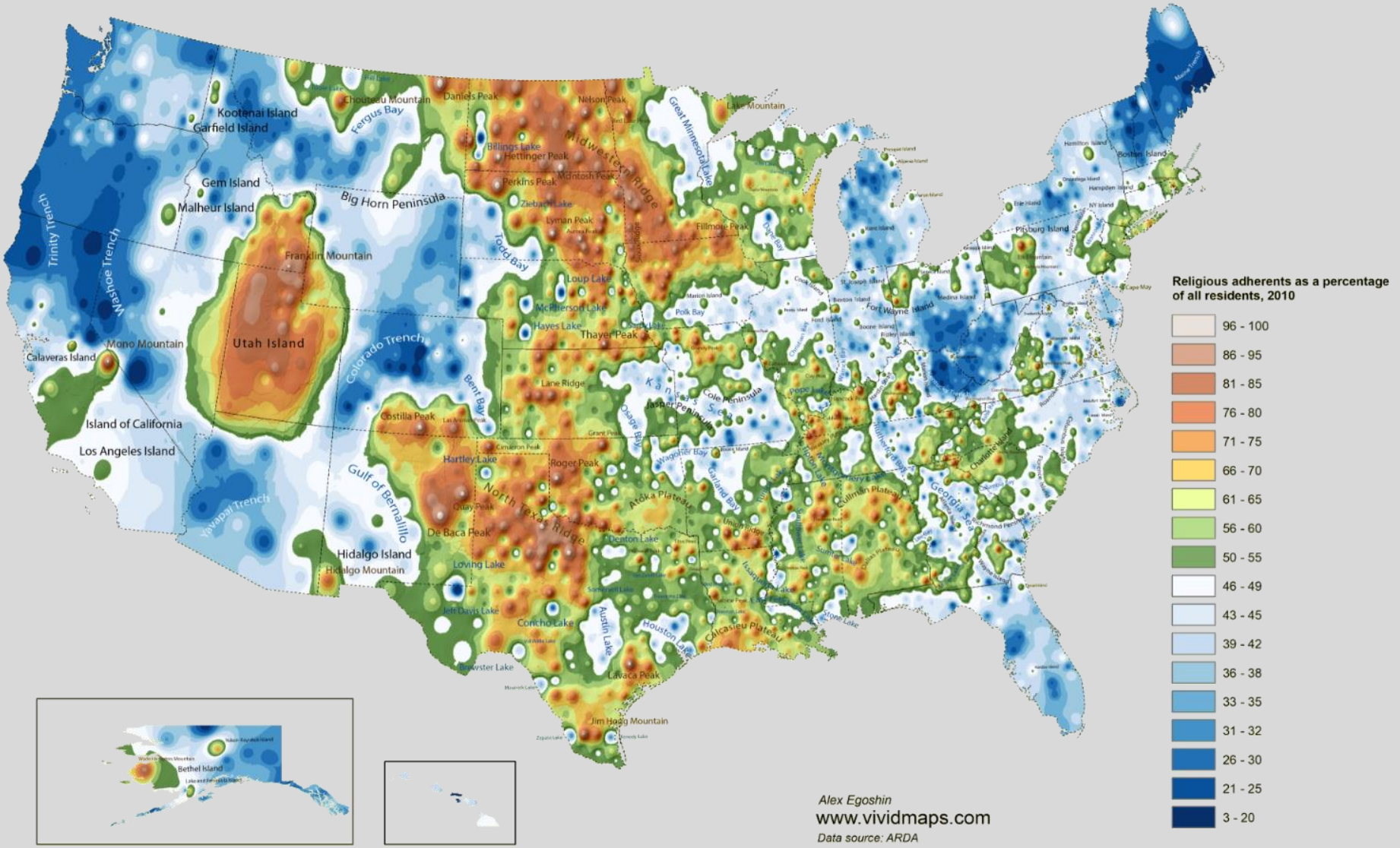

```
ggplot(data=ukmap) +
    geom_polygon(aes(x=long, y=lat, group=group, fill=as.numeric(id)), col='white', size=.1) +
    scale_fill_gradient2(midpoint=14000, high='forestgreen', low='darkblue') +
    coord_fixed(ratio=1/cos(50*2*pi/360))
```



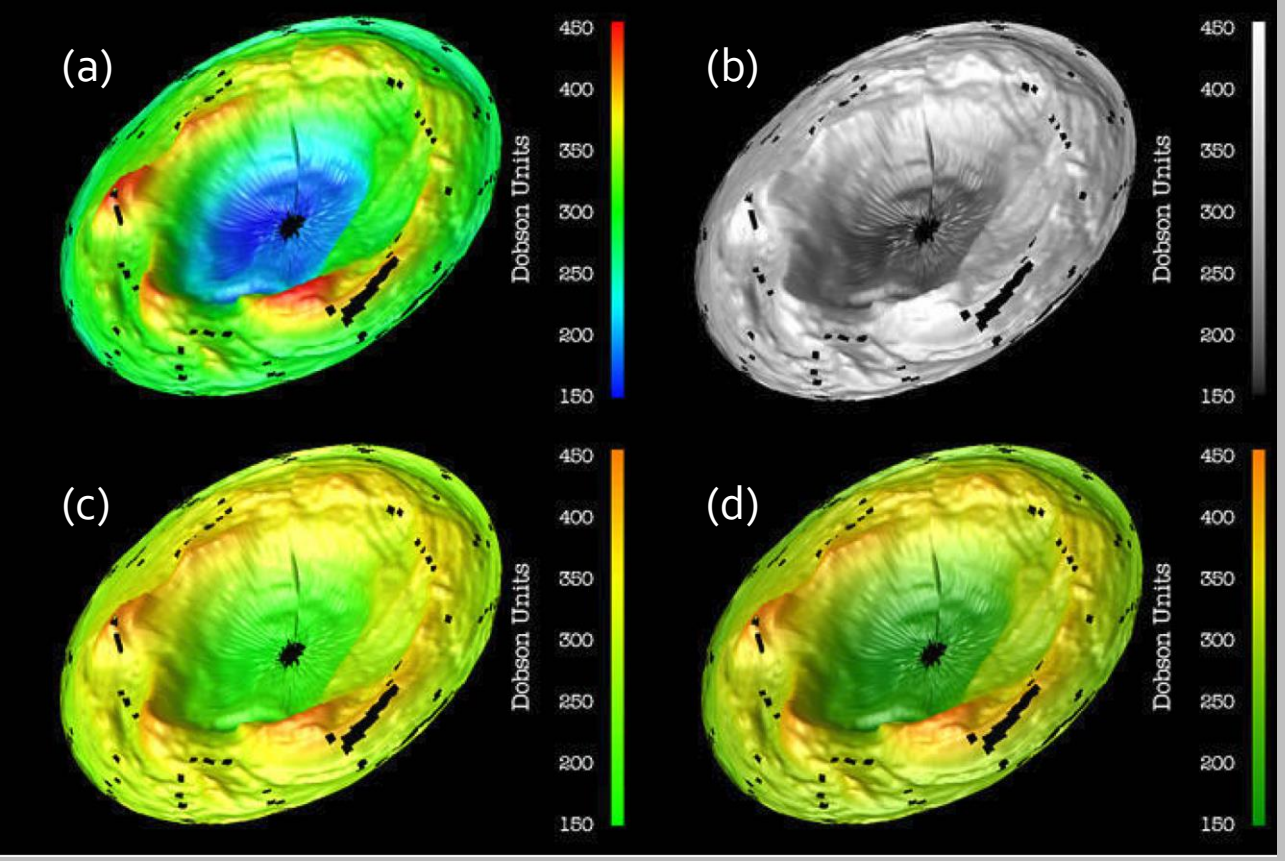
Error: continuous values supplied to a discrete scale

```
ggplot(data=ukmap) +
    geom_polygon(aes(x=long, y=lat, group=group, fill=as.numeric(id)), col='white', size=.1) +
    scale_fill_brewer(type='qual') +
    coord_fixed(ratio=1/cos(50*2*pi/360))
```

Examples of colour scales



Examples of colour scales



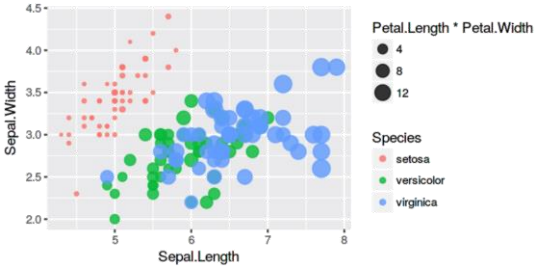
DATASET: total column density of ozone above the southern hemisphere (*Why Should Engineers and Scientists Be Worried About Color?* Rogowitz and Trienish, 1998)

(a) rainbow palette
(b) brightness palette
(c) divergent hue palette
(d) combines (b) and (c)

- The *aesthetic mapping* specifies which data columns should be mapped to which visual dimensions
- The entire range of data values is mapped onto the visual range, which can be configured with `scale_*`

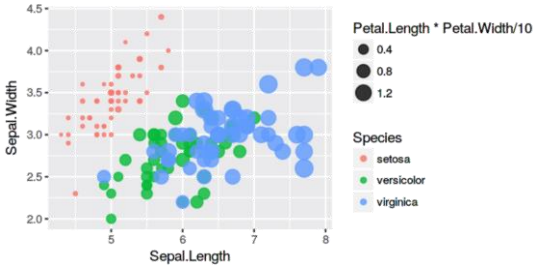
which is a common source of confusion.

```
ggplot(data=iris) +
    geom_point(aes(x=Sepal.Length, y=Sepal.Width, size=Petal.Length * Petal.Width, col=Species)) +
    scale_size_area()
```



Hey, the points are occluding each other. I need to make them smaller

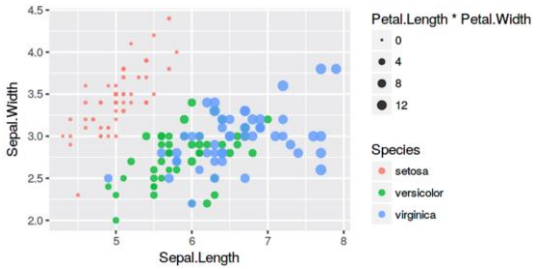
```
ggplot(data=iris) +
    geom_point(aes(x=Sepal.Length, y=Sepal.Width, size=Petal.Length * Petal.Width / 10, col=Species)) +
    scale_size_area()
```



Hold on, the points are exactly the same size as before. WTF?

The data range is different, but scale training has mapped it to precisely the same visual range as before!

```
ggplot(data=iris) +
    geom_point(aes(x=Sepal.Length, y=Sepal.Width, size=Petal.Length * Petal.Width, col=Species)) +
    scale_size_area(max_size=3, limits=c(0,NA))
```



I really want to map data range [0, max_in_data] to the output range [0, 3pt]

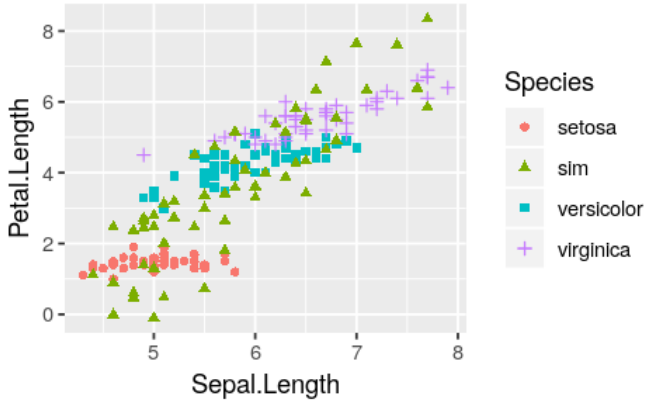
- The *aesthetic mapping* specifies which data columns should be mapped to which visual dimensions
- The entire range of data values is mapped onto the visual range, which can be configured with `scale_*`

which gives us a nifty blending trick

```
# Generate a synthetic dataset
fit <- lm(Petal.Length ~ Sepal.Length, data=iris)
df <- copy(iris)
df[, Petal.Length := simulate(fit)]
df <- df[sample(nrow(iris),60,replace=FALSE)]

# Plot both iris and the synthetic dataset
ggplot() +
    geom_point(data=iris, aes(x=Sepal.Length, y=Petal.Length, col=Species, shape=Species)) +
    geom_point(data=df, aes(x=Sepal.Length, y=Petal.Length, col='sim', shape='sim'))
```

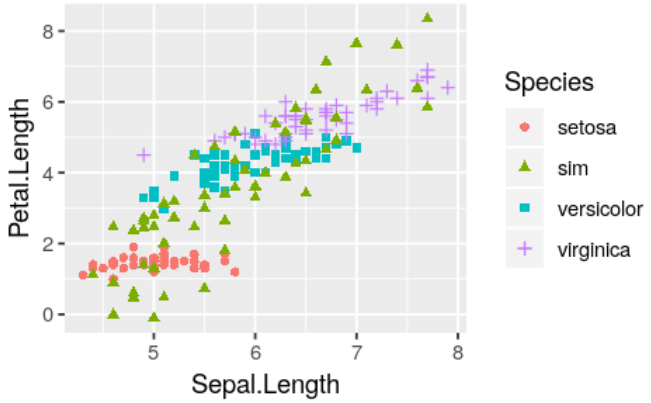
Training phase:
what data values do we see for col, across the entire plot?
"setosa" "versicolor" "virginica" "sim"

Scale phase:
the data has 4 distinct string values, so we'll use a discrete colour scale by default



- The *aesthetic mapping* specifies which data columns should be mapped to which visual dimensions
- The entire range of data values is mapped onto the visual range, which can be configured with `scale_*`

```
# Generate a synthetic dataset
fit <- lm(Petal.Length ~ Sepal.Length, data=iris)
df <- copy(iris)
df[, Petal.Length := simulate(fit)]
df <- df[sample(nrow(iris),60,replace=FALSE)]

# Plot both iris and the synthetic dataset
ggplot() +
    geom_point(data=iris, aes(x=Sepal.Length, y=Petal.Length, col=Species, shape=Species)) +
    geom_point(data=df, aes(x=Sepal.Length, y=Petal.Length, col='sim', shape='sim'))
```



- Syntactic sugar:
  plot specs can be set in ggplot(), and they become defaults for the plot layers

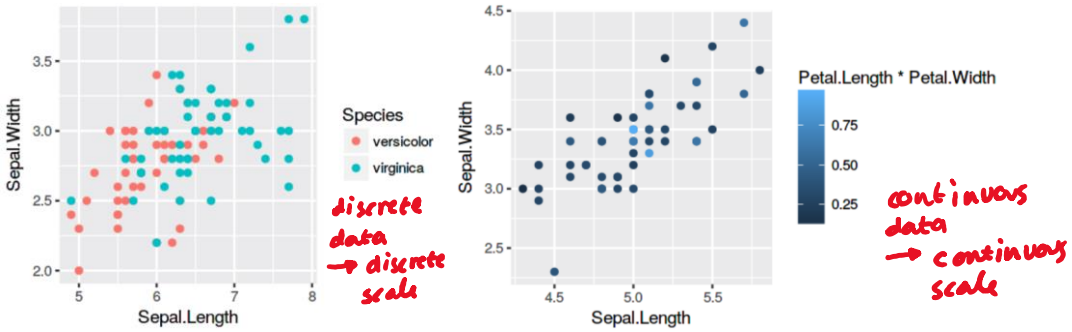
```
ggplot(data=iris, aes(x=Sepal.Length, y=Petal.Length)) +  # set default data, x, y
    geom_point(aes(col=Species, shape=Species)) +         # use default data, x, y
    geom_point(data=df, aes(col='sim', shape='sim'))      # override data, use default x,y
```

- The *aesthetic mapping* specifies which data columns should be mapped to which visual dimensions
- The entire range of data values is mapped onto the visual range, which can be configured with `scale_*`

I have two nice plots. What if I combine them?

```
ggplot() +
    geom_point(data=iris[Species != 'setosa'], aes(x=Sepal.Length, y=Sepal.Width, col=Species))
```

```
ggplot() +
    geom_point(data=iris[Species == 'setosa'], aes(x=Sepal.Length, y=Sepal.Width, col=Petal.Length*Petal.Width))
```



discrete data → discrete scale

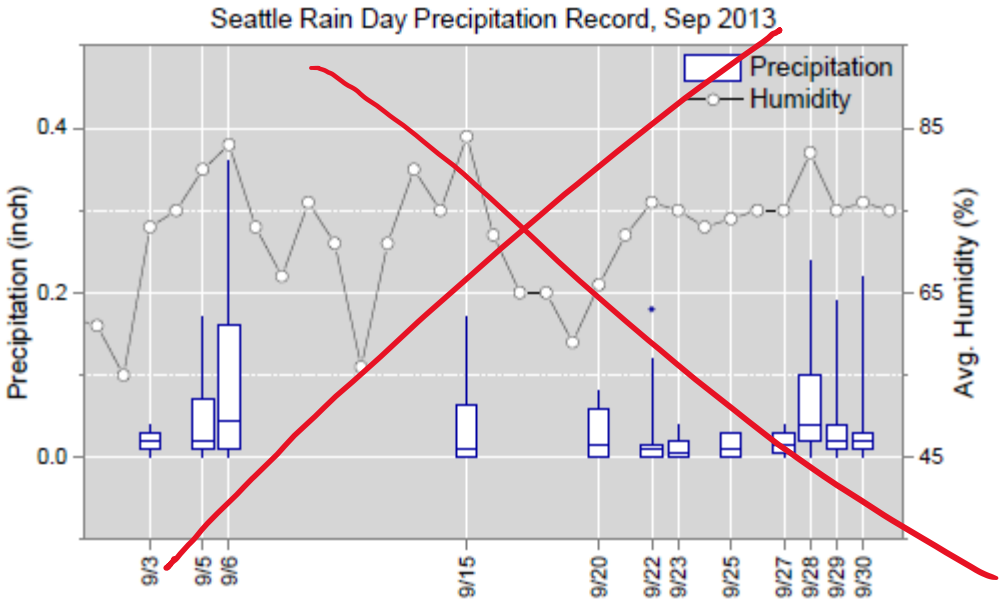
continuous data → continuous scale

```
ggplot() +
    geom_point(data=iris[Species == 'setosa'], aes(x=Sepal.Length, y=Sepal.Width, col=Petal.Length*Petal.Width)) +
    geom_point(data=iris[Species != 'setosa'], aes(x=Sepal.Length, y=Sepal.Width, col=Species))
```

This produces an error when it's training the scale. (In other situations it might coerce num → string and show you a discrete scale with thousands of levels.)

- The *aesthetic mapping* specifies which data columns should be mapped to which visual dimensions
- The entire range of data values is mapped onto the visual range, which can be configured with `scale_*`

How would you set up the y axis for this plot?



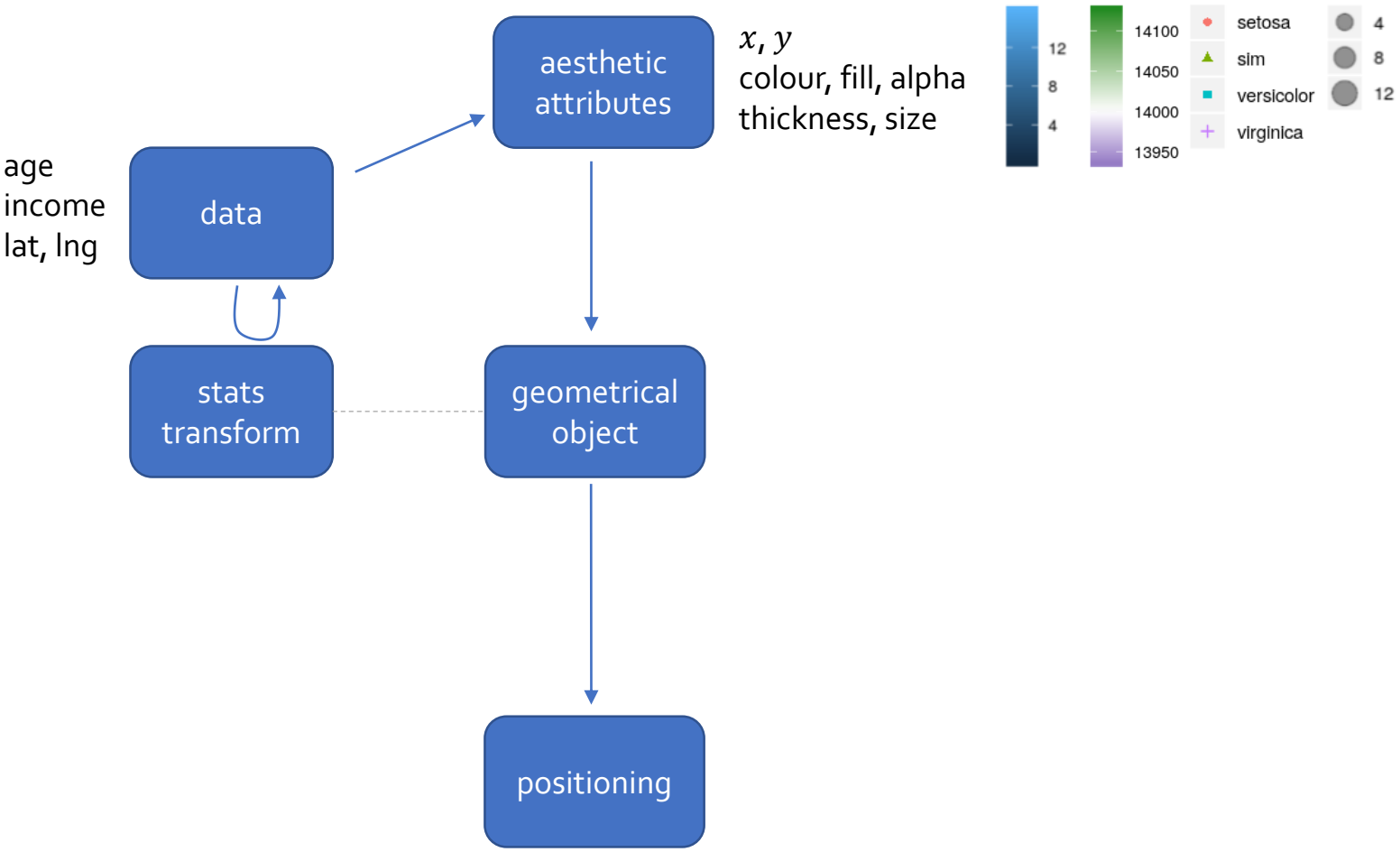
You can't! If you use

aes(y = precipitation)

& aes(y = Humidity)

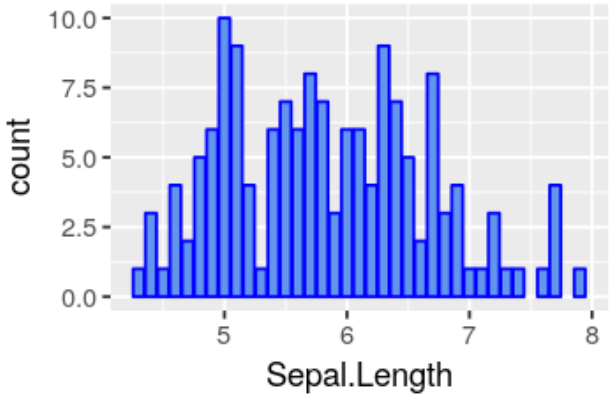
ggplot will simply construct a single y scale for all the values.

Components of a chart



- A geom is an object that is plotted, occupying part of the coordinate space
- A stat is a transformation of the data
- Each geom comes with a default stat (sometimes just stat='identity')
  Some stats come with a default aes

```
ggplot(data=iris) +
    geom_bar(aes(x=Sepal.Length, y=..count..), col='blue', fill='cornflowerblue', stat='bin', bins=37)

ggplot(data=iris) +
    geom_bar(aes(x=Sepal.Length), col='blue', fill='cornflowerblue')
```



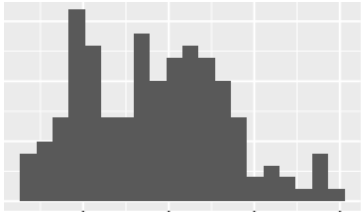
stat_bin transforms the x data, adding a new column called ..count..

Apply the default stat for geom_bar
- this adds a ..count.. column
- and applies aes(y = ..count..)

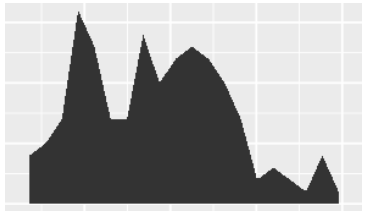
- A geom is an object that is plotted, occupying part of the coordinate space
- A stat is a transformation of the data
- Each geom comes with a default stat (sometimes just stat='identity')
  Some stats come with a default aes

So you can easily change how the data is depicted

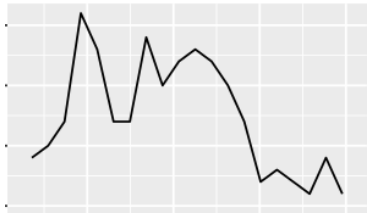
```
ggplot(data=iris) +
    geom_bar(aes(x=Sepal.Length), stat='bin', bins=20)

ggplot(data=iris) +
    geom_area(aes(x=Sepal.Length, y=..count..), stat='bin', bins=20)

ggplot(data=iris) +
    geom_line(aes(x=Sepal.Length, y=..count..), stat='bin', bins=20) +
    scale_y_continuous(limits=c(0,NA))

ggplot(data=iris) +
    geom_point(aes(x=Sepal.Length, y=..count..), stat='bin', bins=20) +
    scale_y_continuous(limits=c(0,NA))
```



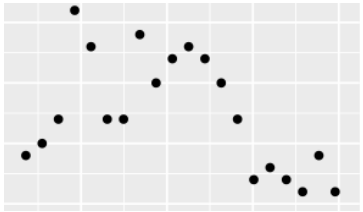
geom_bar



geom_area



geom_line



geom_point

- A geom is an object that is plotted, occupying part of the coordinate space
- A stat is a transformation of the data
- Each geom comes with a default stat (sometimes just stat='identity')
  Some stats come with a default aes

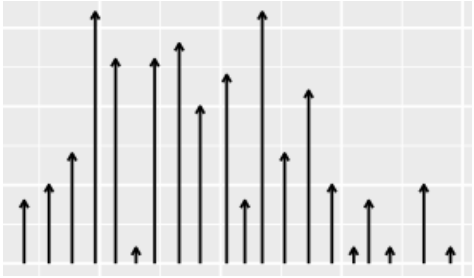
which is usually helpful, but sometimes bothersome —

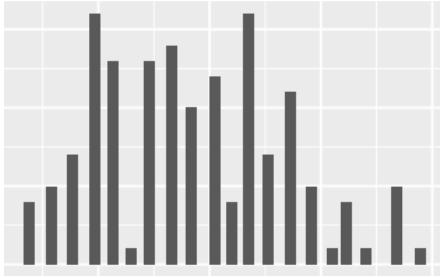
If the stat stuff gets confusing, just give up and preprocess the data yourself.

```
ggplot(data=iris) +
    geom_segment(aes(x=Sepal.Length, xend=Sepal.Length, y=0, yend=..count..), stat='bin', bins=20)
```

Error: stat_bin() must not be used with a y aesthetic.

```
# Do my own version of stat_bin (group the data by Sepal.Length, and get counts)
df = as.data.table(iris)[, list(Sepal.Length=mean(Sepal.Length), count=.N), by=cut(Sepal.Length, breaks=20)]

ggplot(data=df) +
    geom_segment(aes(x=Sepal.Length, xend=Sepal.Length, y=0, yend=count), arrow=arrow(length=unit(0.03, 'npc')))

ggplot(data=df) +
    geom_rect(aes(xmin=Sepal.Length-0.05, xmax=Sepal.Length+0.05, ymin=0, ymax=count))
```





We often call graphics charts (from *χάρτης* or Latin *charta*, a leaf of paper or papyrus). There are pie charts, bar charts, line charts, and so on. This book shuns chart typologies. For one thing, charts are usually instances of much more general objects. Once we understand that a pie is a divided bar in polar coordinates, we can construct other polar graphics that are less well known. We will also come to realize why a histogram is not a bar chart and why many other graphics that look similar nevertheless have different grammars.

There is also a practical reason for shunning chart typology. If we endeavor to develop a charting instead of a graphing program, we will accomplish two things. First, we inevitably will offer fewer charts than people want. Second, our package will have no deep structure. Our computer program will be unnecessarily complex, because we will fail to reuse objects or routines that function similarly in different charts. And we will have no way to add new charts to our system without generating complex new code. Elegant design requires us to think about a theory of graphics, not charts.

A chart metaphor is especially popular in user interfaces. The typical interface for a charting program is a catalog of little icons of charts. This is easy to construct from information gathered in focus groups, surveys, competitive analysis, and user testing. Much more difficult is to understand what users intend to do with their data when making a graphic. Instead of taking this risk, most charting packages channel user requests into a rigid array of chart types. To atone for this lack of flexibility, they offer a kit of post-creation editing tools to return the image to what the user originally envisioned. They give the user an impression of having explored data rather than the experience.

Leland Wilkinson. The Grammar of Graphics, section 1.1.

"A histogram is just a geom_bar with a stat_bin"

Think in terms of combining simple geoms and stats, and you'll be able to create an endless variety of charts, without having to learn a taxonomy.

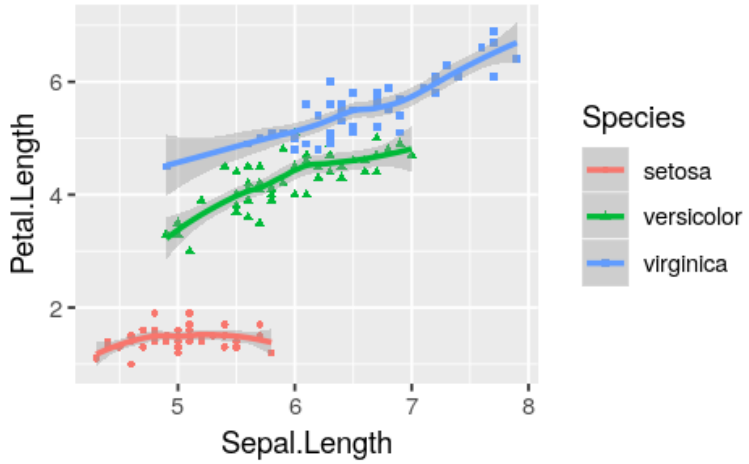
But... ggplot2 provides a confusing taxonomy of geoms and stats! Happily you don't need to remember them, because they are mostly just groupings of simpler geoms, with sensible defaults for stat.

geom_histogram =
    geom_bar with stat=count

stat_smooth =
    geom_ribbon + geom_line with stat=smooth

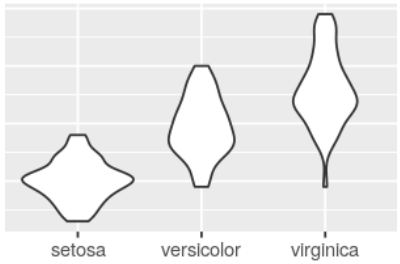
Some useful stats + geoms:

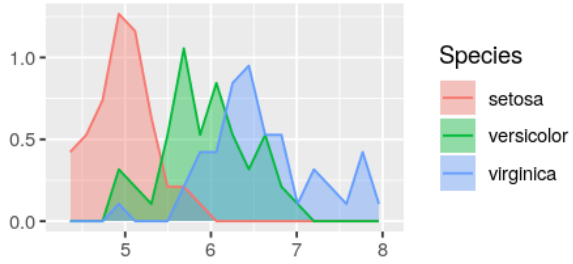
```
ggplot(data=iris, aes(x=Sepal.Length, y=Petal.Length, col=Species)) +
    geom_point(aes(shape=Species)) +
    stat_smooth(method='loess')

ggplot(data=iris, aes(x=Sepal.Length, y=Petal.Length, col=Species)) +
    geom_ribbon(aes(group=Species), stat='smooth', method='loess', size=.2, fill='grey75', col=NA) +
    geom_line(stat='smooth', method='loess') +
    geom_point(aes(shape=Species), size=1)
```

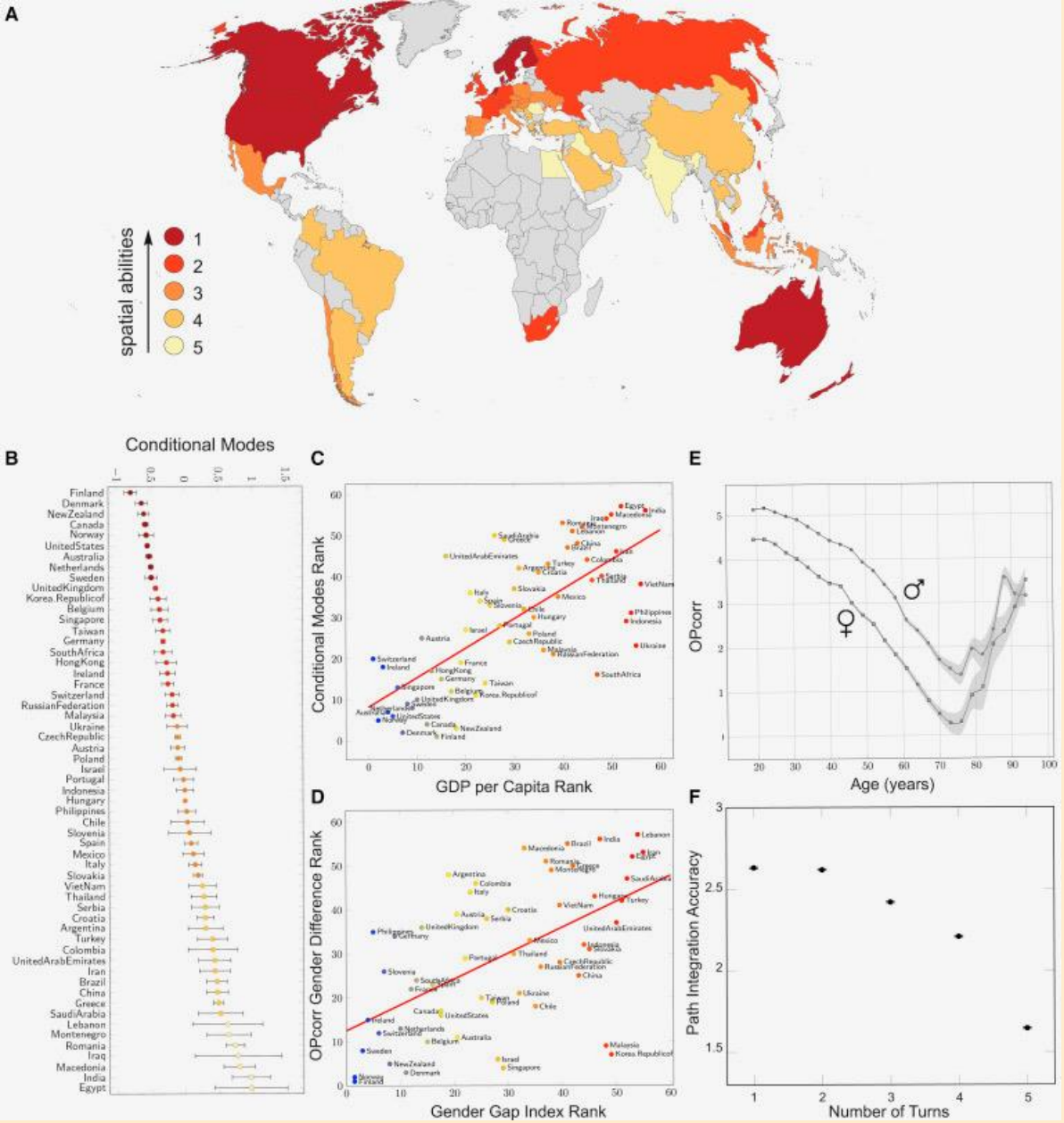


Some useful stats + geoms:

```
ggplot(data=iris) +
    geom_boxplot(aes(x=Species, y=Sepal.Length))

ggplot(data=iris) +
    geom_violin(aes(x=Species, y=Sepal.Length))

ggplot(data=iris) +
    geom_area(aes(x=Sepal.Length, y=..density.., fill=Species), position='identity', stat='bin', alpha=.4, bins=20) +
    geom_line(aes(x=Sepal.Length, y=..density.., col=Species), stat='bin', bins=20)
```







DATASET: Spatial navigation ability, measured in a computer game (*Global determinants of navigation ability*, Coutrot et al. 2017)

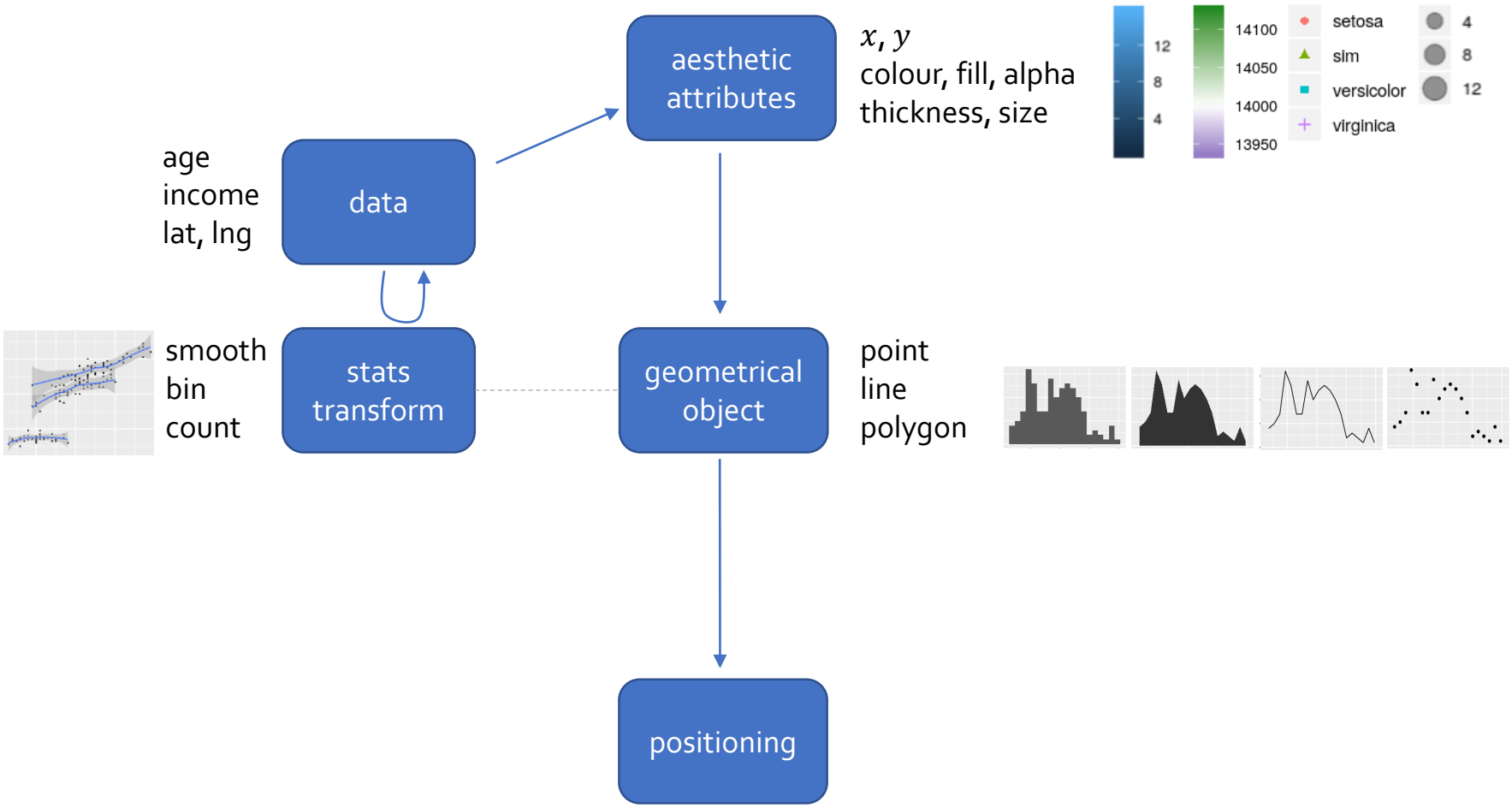
2.5 million subjects were shown a map with waypoints, then asked to visit the waypoints, then shoot a flare at their start point.

OP = Overall Performance (path duration, path length, shooting accuracy, combined using PCA)

CM = Conditional Modes (overall performance compared to global average)

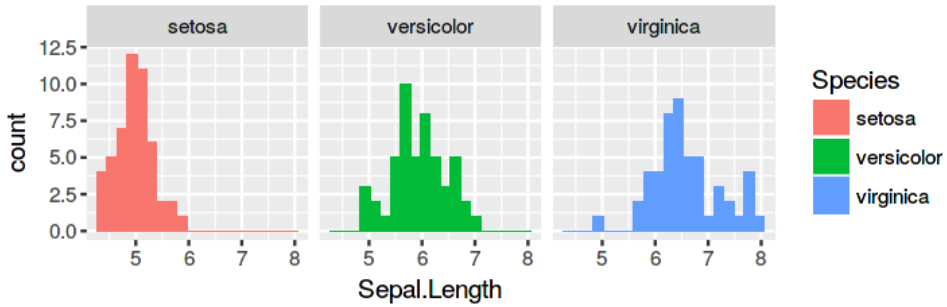
Exercise: what are the geoms in this plot?

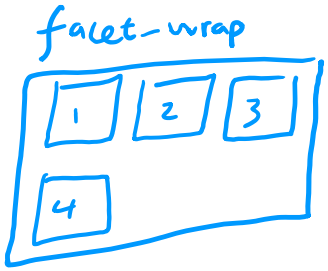
Components of a chart



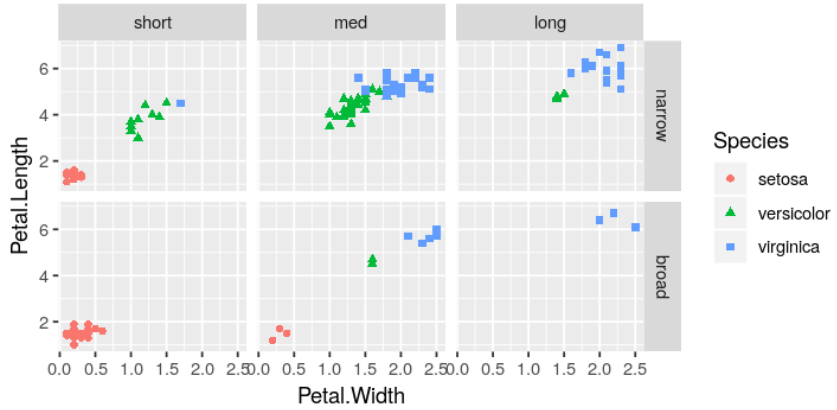
- A facetted plot shows several panels, each containing a subset of the data

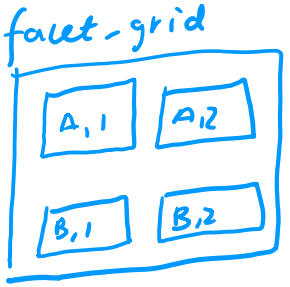
```
ggplot(data=iris) +
    geom_bar(aes(x=Sepal.Length, fill=Species), stat='bin', bins=20) +
    facet_wrap(~Species)
```

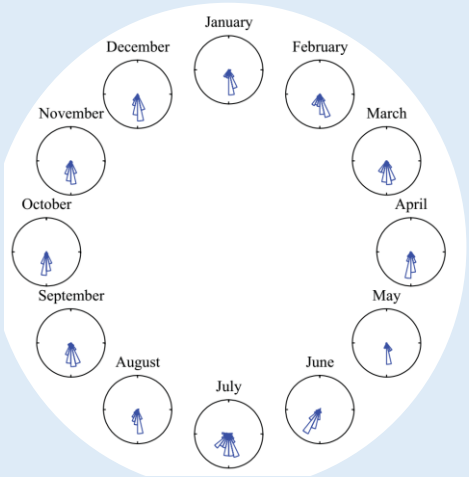




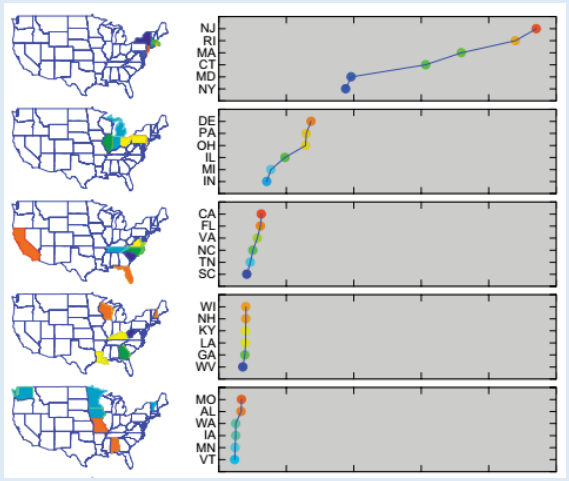

```
# Create two categorical (i.e. string) columns, by binning Sepal.Width and Sepal.Length into buckets
iris[, Sepal.Width.f := cut(Sepal.Width, 2, labels=c('narrow', 'broad'))]
iris[, Sepal.Length.f := cut(Sepal.Length, 3, labels=c('short', 'med', 'long'))]

ggplot(data=iris) +
    geom_point(aes(x=Petal.Width, y=Petal.Length, col=Species, shape=Species)) +
    facet_grid(Sepal.Width.f ~ Sepal.Length.f)
```



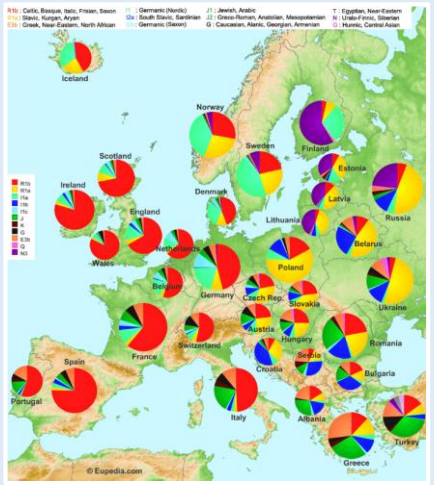




This is a type of faceting —
but ggplot2 has only implemented simple rectangular arrangements.

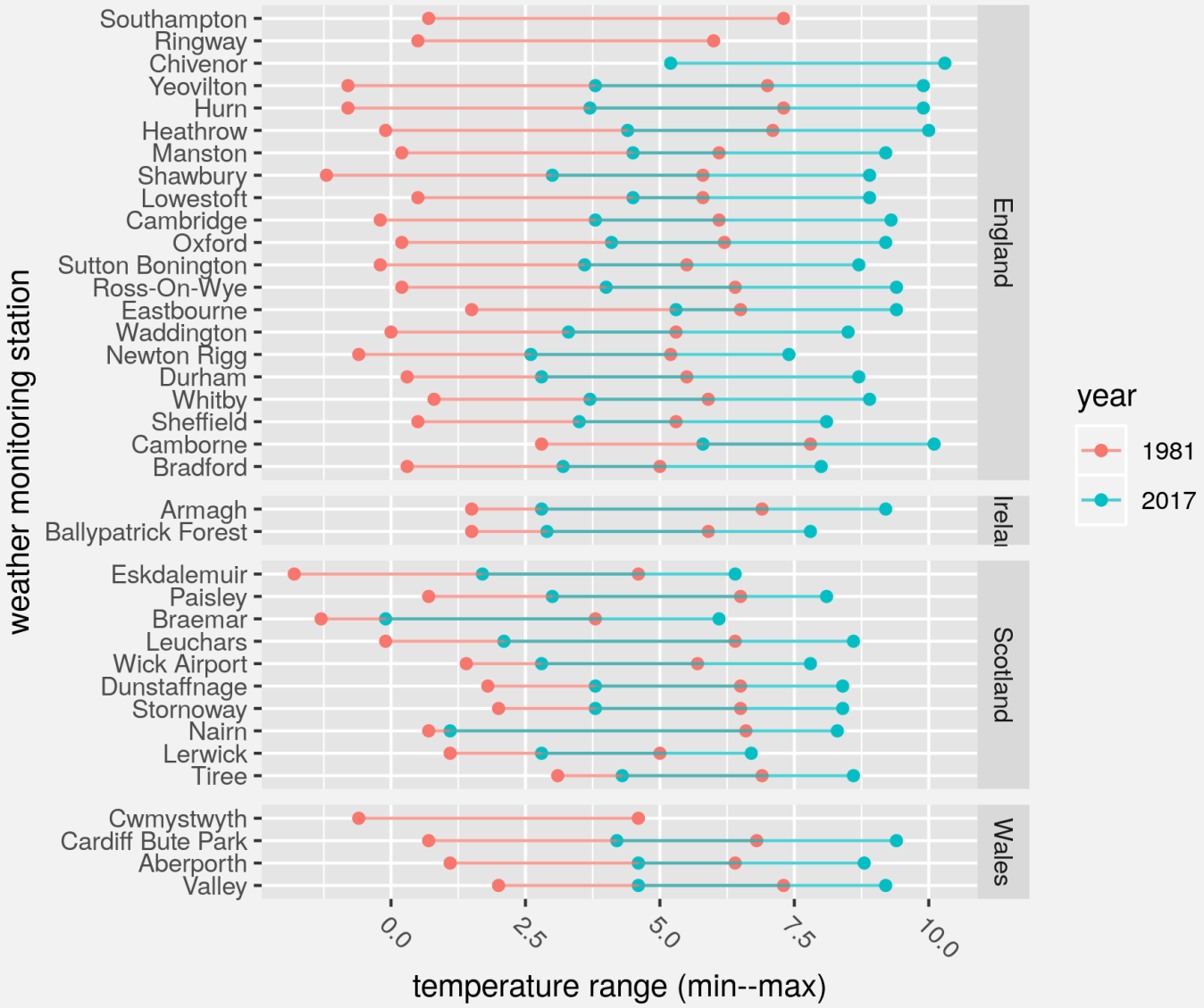


This is a visual arrangement, not a data arrangement. It's outside the scope of the Grammar of Graphics.
(Hack around with gridExtra instead.)



The Grammar of Graphics doesn't go this far (but it should).

Some of the things you *can't* do with facets

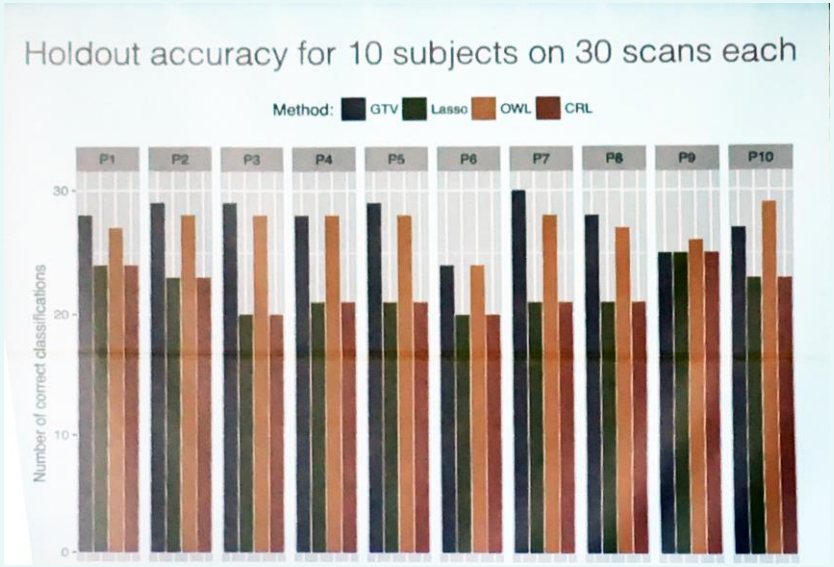


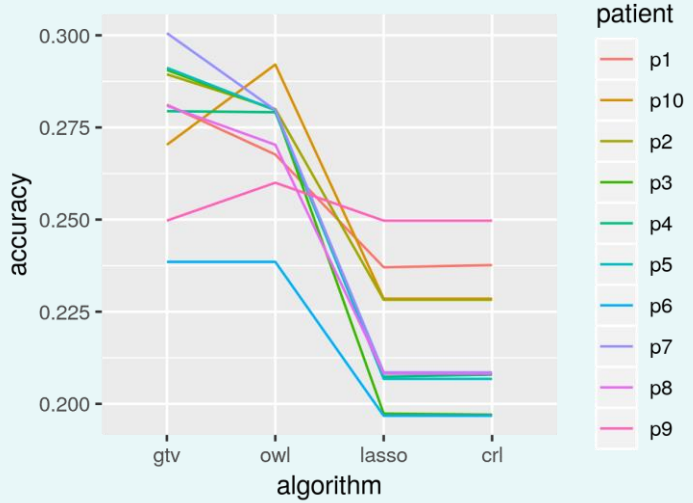
Exercise: control which axes are shared between facets,
and adjust the size of each facet according to how much y-range it spans.

DATASET: medical data for 10 patients was processed by 4 machine learning algorithms, and each algorithm was scored for prediction accuracy.

patient ID	machine learning algorithm	accuracy score
p2	lasso	0.2282353
p3	owl	0.2797059
p3	crl	0.1970588
⋮	⋮	⋮

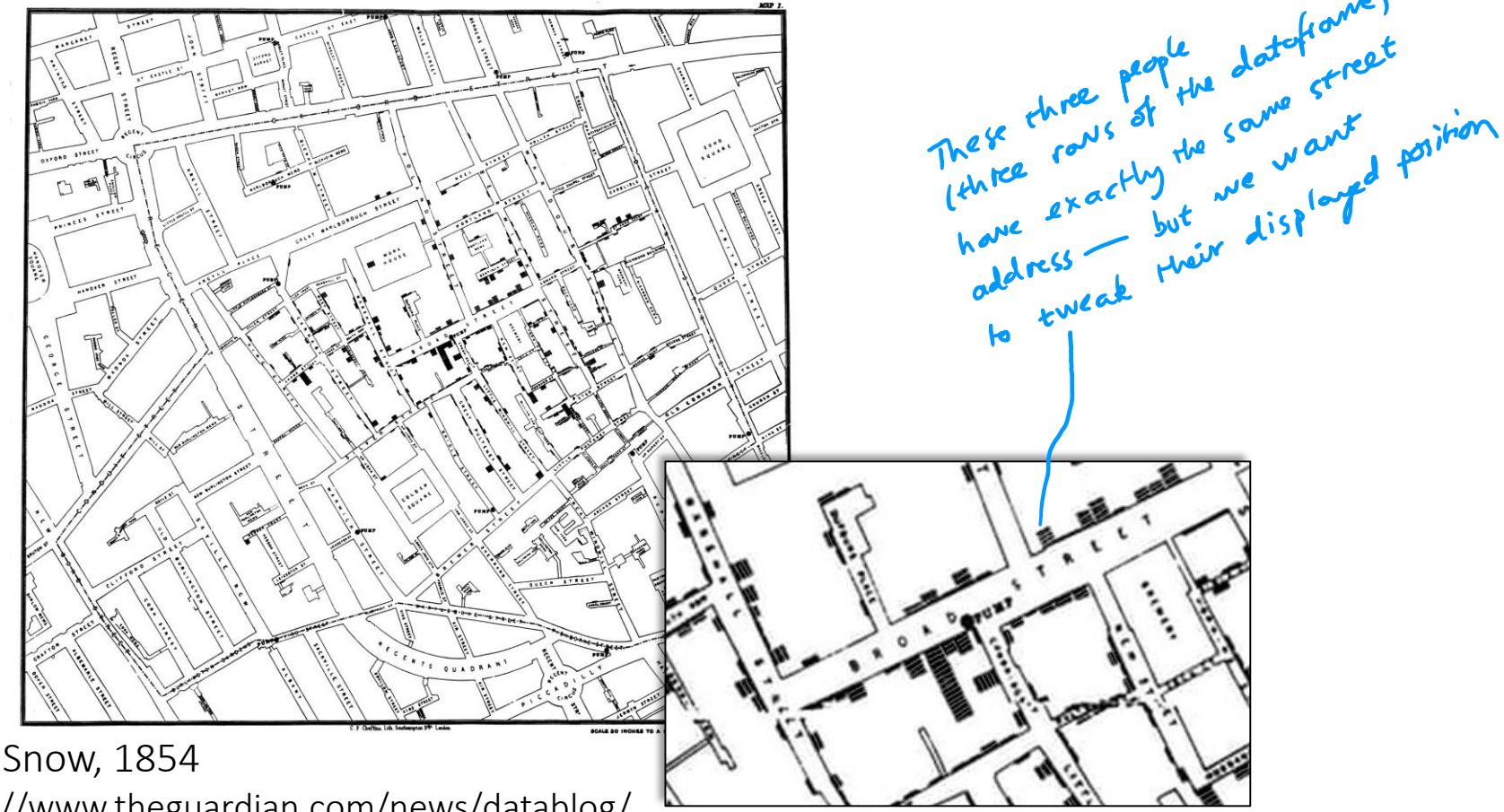
```
ggplot(data=scans) +
    geom_bar(aes(x=algorithm, fill=algorithm, y=accuracy), stat='identity') +
    facet_grid(~patient)

ggplot(data=scans) +
    geom_line(aes(x=algorithm, y=accuracy, group=patient, col=patient))
```





Exercise: what comparisons are you inviting the viewer to make?

- Each object in a geom has values for its x and y scales
  The coordinate system says how x and y should be located on the display
- The displayed position of a geom can be tweaked, to deal with overlap



John Snow, 1854
https://www.theguardian.com/news/datablog/2013/mar/15/john-snow-cholera-map

- Each object in a geom has values for its x and y scales
  The coordinate system says how x and y should be located on the display
- The displayed position of a geom can be tweaked, to deal with overlap

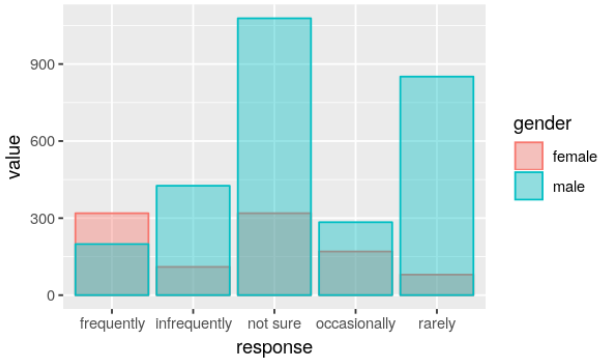
DATASET: A survey of U.S. scholars (Morton and Price, 1989).
Surveyed were 5,385. Respondents numbered 3,835. Respondents answered the question "How often, if at all, do you think the peer review refereeing system for scholarly journals in your field is biased in favour of males?"

	rarely	infrequently	occasionally	frequently	not sure
male respondents	851	426	284	199	1078
female respondents	80	110	170	319	319

gender	response	value
male	rarely	851
female	rarely	80
male	infrequently	426
⋮		

override the default, stat_count

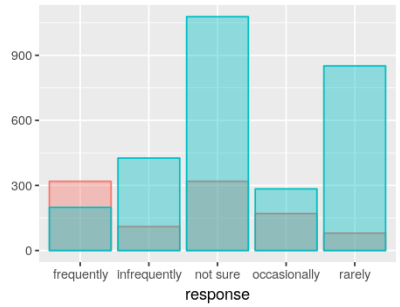
```
ggplot(data=survey) +
    geom_bar(aes(x=response, y=value, col=gender, fill=gender), stat='identity', position='identity', alpha=.4)
```



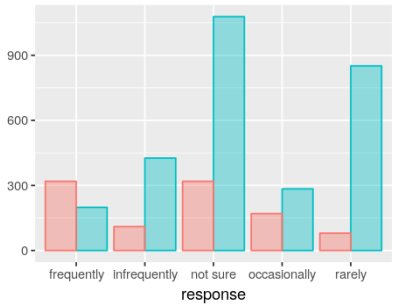
At each x there are two rows in the dataset
→ two bars that overlap

- Each object in a geom has values for its x and y scales
  The coordinate system says how x and y should be located on the display
- The displayed position of a geom can be tweaked, to deal with overlap

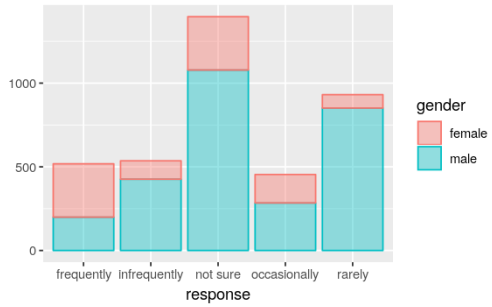
```
ggplot(data=survey) +
    geom_bar(aes(x=response, y=value, col=gender, fill=gender), stat='identity', position='identity', alpha=.4)

ggplot(data=survey) +
    geom_bar(aes(x=response, y=value, col=gender, fill=gender), stat='identity', position='dodge', alpha=.4)

ggplot(data=survey) +
    geom_bar(aes(x=response, y=value, col=gender, fill=gender), stat='identity', position='stack', alpha=.4)
```



position_identity



position_dodge

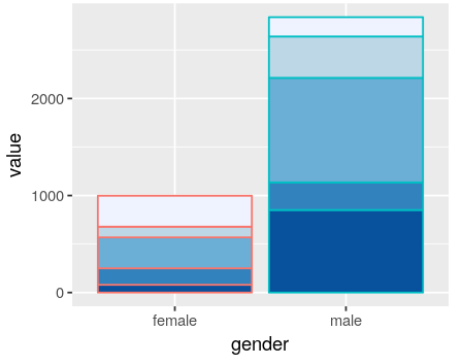


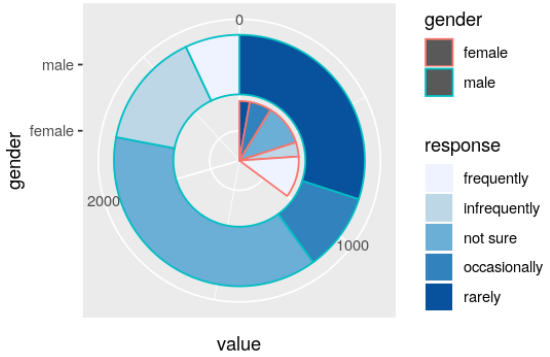
position_stack

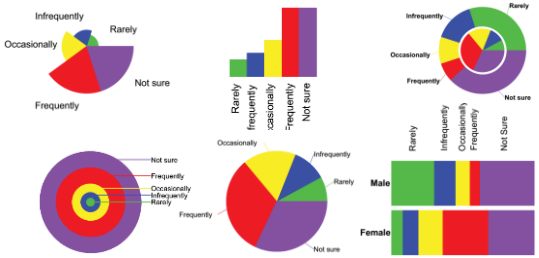
- Each object in a geom has values for its x and y scales
  The coordinate system says how x and y should be located on the display
- The displayed position of a geom can be tweaked, to deal with overlap

```
ggplot(data=survey) +
    geom_bar(aes(x=gender, y=value, col=gender, fill=response), stat='identity', position='stack') +
    scale_fill_brewer()

ggplot(data=survey) +
    geom_bar(aes(x=gender, y=value, col=gender, fill=response), stat='identity', position='stack') +
    scale_fill_brewer() +
    coord_polar(theta='y')
```

Use y for the θ-coordinate, x for the r-coordinate



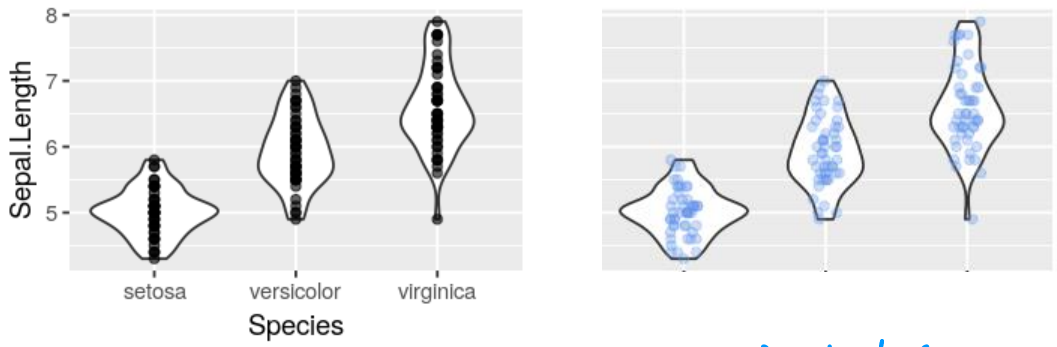




These are all stacked barcharts — just in different coordinate systems!

- Each object in a geom has values for its x and y scales
  The coordinate system says how x and y should be located on the display
- The displayed position of a geom can be tweaked, to deal with overlap

```
ggplot(data=iris, aes(x=Species, y=Sepal.Length)) +
    geom_violin() +
    geom_point(alpha=.6)

ggplot(data=iris, aes(x=Species, y=Sepal.Length)) +
    geom_violin() +
    geom_point(position=position_jitter(width=0.1, height=0), col='cornflowerblue', alpha=.3)
```



position_jitter

by class

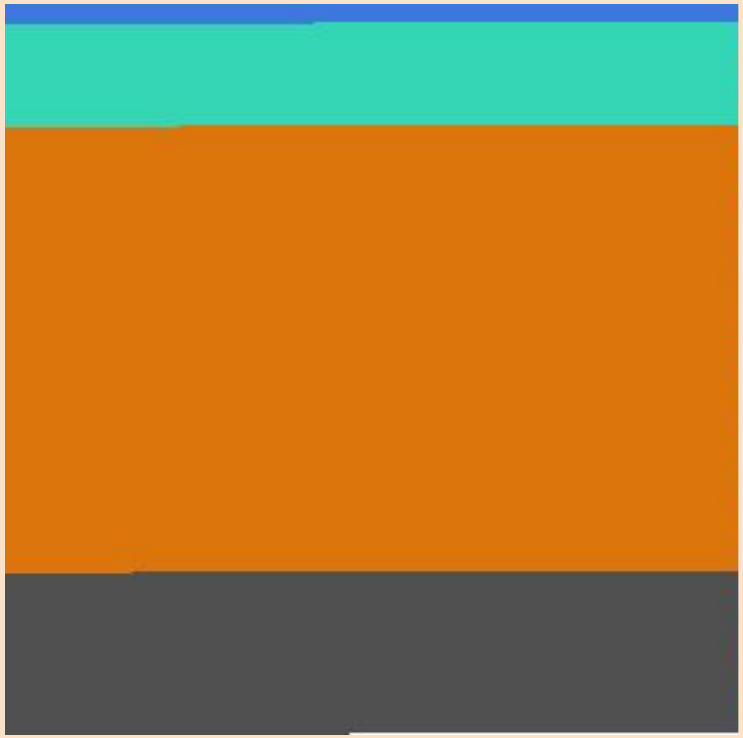
by time

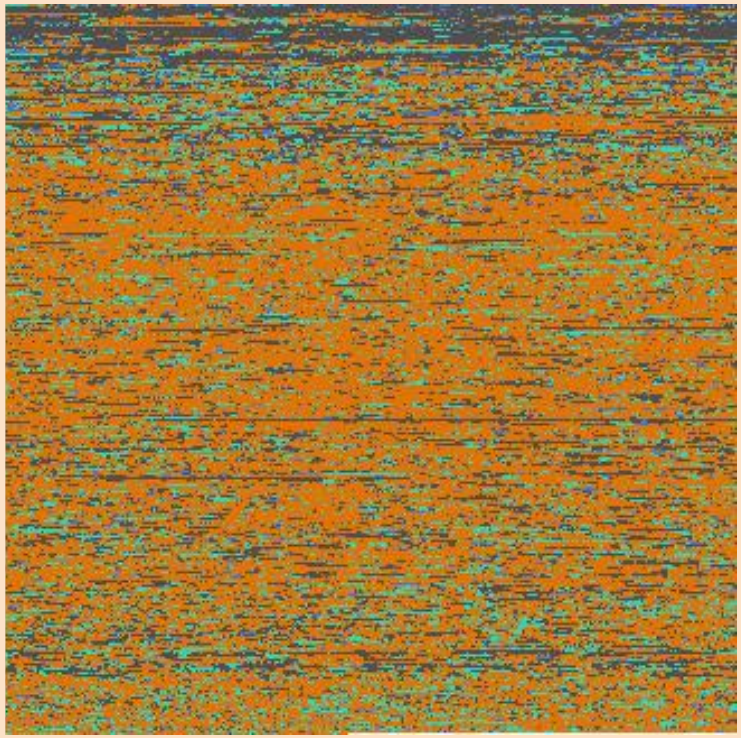
friendly

host nation

civilian

enemy

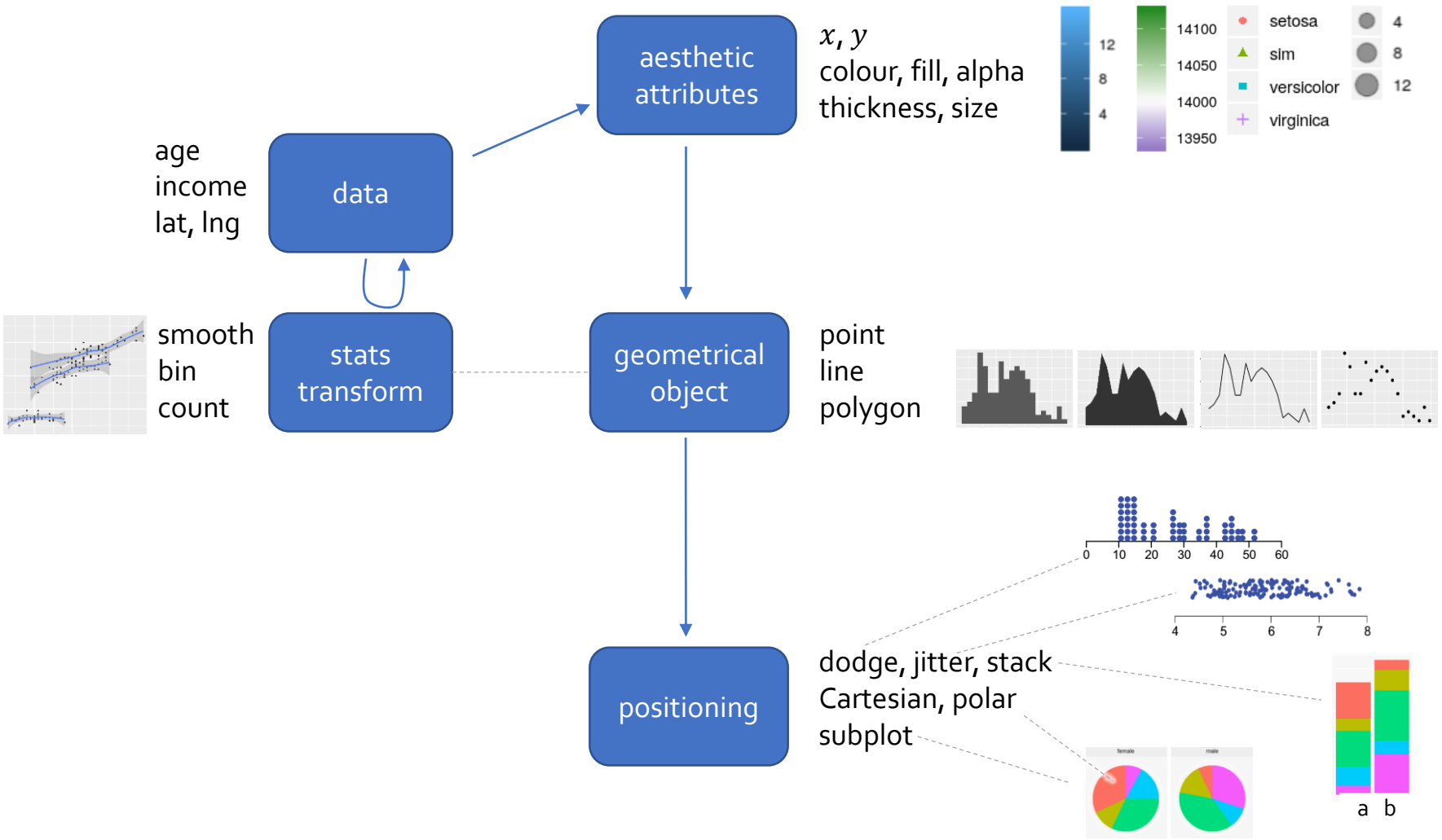




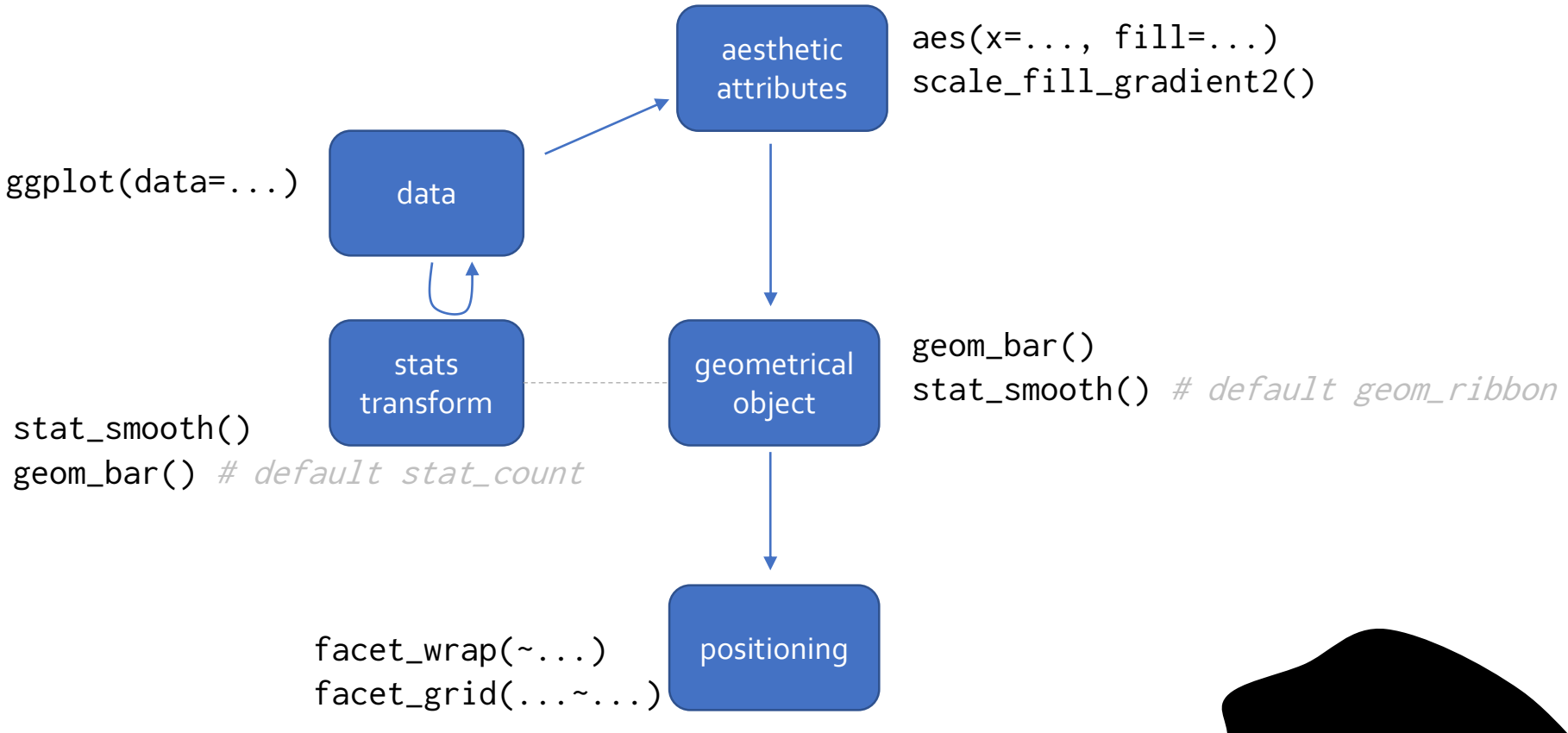
Canadian designer Kamel Makhloufi's pair of stark graphs visualize the human toll of the Iraq war. Each pixel represents a death.

https://www.flickr.com/photos/melkaone/5121285002/

Components of a chart



Components of a chart



data. aes. stat. geom. facet. position. coord. guides.

- Apply styling with theme(). Beautiful code = ugly plot, ugly code = beautiful plot.
- Modify the guide (i.e. ticks or legend) for a scale

  scale_y_continuous(breaks=..., labels=...) scale_colour_discrete(..., guide=FALSE)

  scale_x_datetime(...) guides(colour = guide_legend(...), size=FALSE)

- Name the guides

  labs(x="", title=...) scale_colour_discrete(..., name=...)

```
ggplot(data=survey) +
    geom_bar(aes(x=response, y=value, col=gender, fill=gender), stat='identity', position='dodge', alpha=.4)

g <- ggplot(data=survey) +
        geom_bar(aes(x=response, y=value, col=gender, fill=gender), stat='identity', position='dodge', alpha=.4) +
        scale_y_continuous(breaks=c(0,250,500,750,1000)) +
        guides(colour = guide_legend(override.aes=list(alpha=1))) +
        labs(x="", y="", title="Number of responses") +
        theme_economist() +
        theme(plot.background = element_rect(color=NA, fill="transparent"),
              panel.background = element_rect(color=NA, fill="grey90"),
              legend.background = element_rect(color=NA, fill="transparent"),
              axis.text.x = element_text(angle=-45, hjust=0))
ggsave(g, file='~/winhome/Downloads/myplot.png', bg='transparent', width=3, height=3)
```