Computational Skills for Biostatistics I: Lecture 3

Amy Willis, Biostatistics, UW

17 April, 2019

Graphical communication in practice

Graphical communication is critical for both *exploring* and *explaining* data

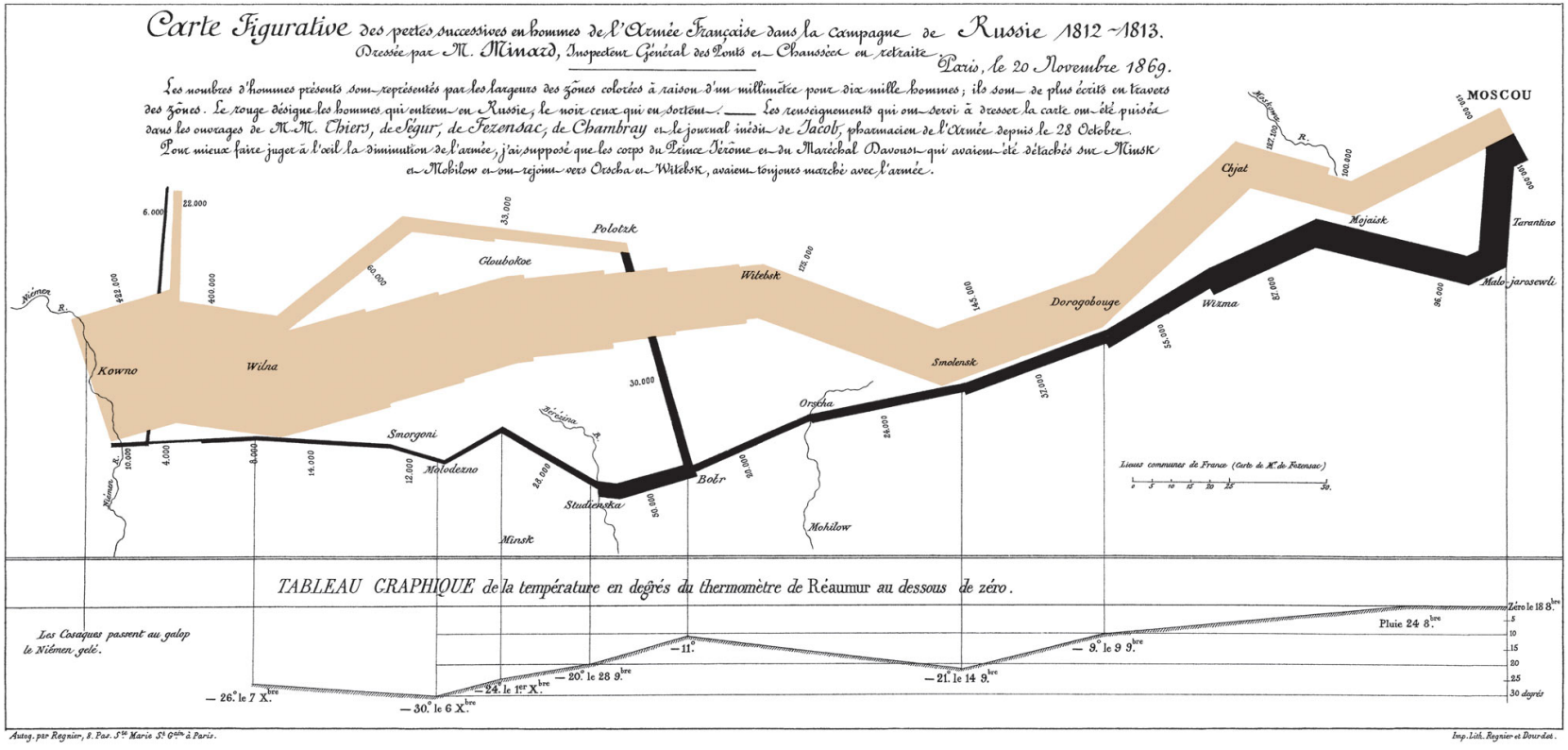
- base graphics in `R`: static
- `ggplot`: static
- `shiny`: interactive
- Non-data-based figures

Name to know: Edward Tufte

The Rules:

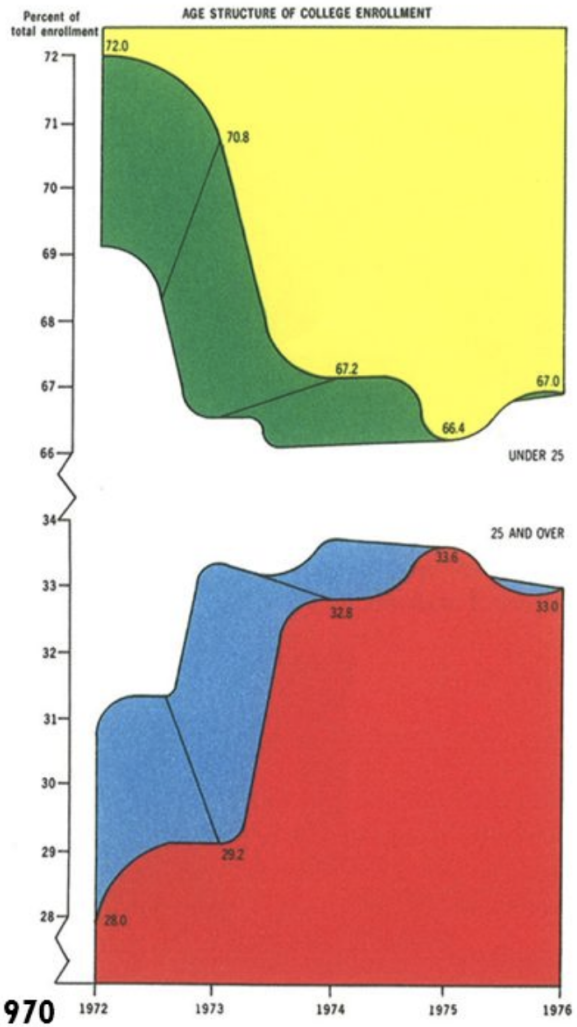
1. Show Your Data
2. Use Graphics
3. Avoid Chartjunk
4. Utilize Data-ink
5. Use Labels
6. Utilize Micro/Macro
7. Separate Layers
8. Use Multiples
9. Utilize Color
10. Understand Narrative

The greatest statistical graphic ever drawn



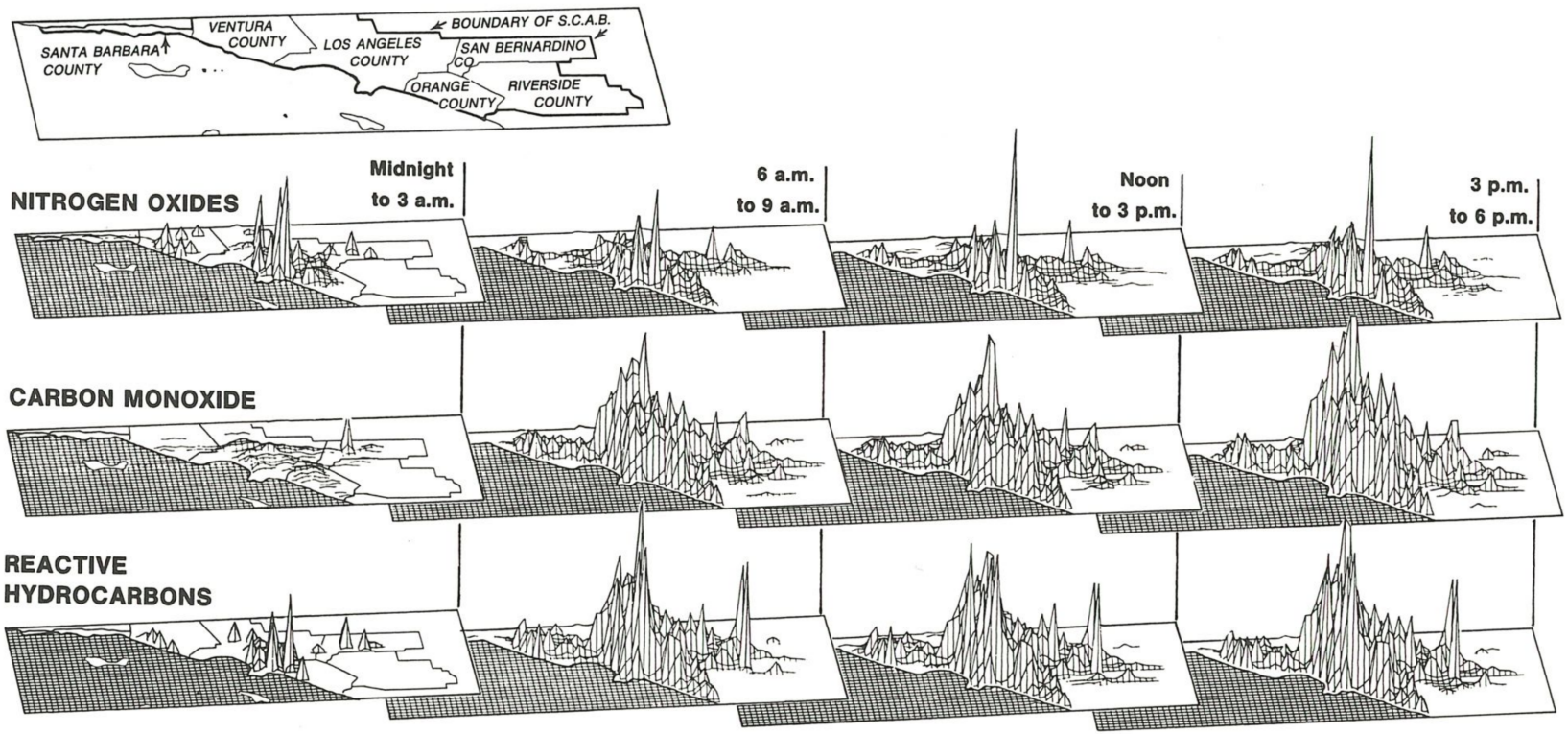
The worst statistical graphic ever drawn

"This may well be the worst graphic ever to find its way into print"

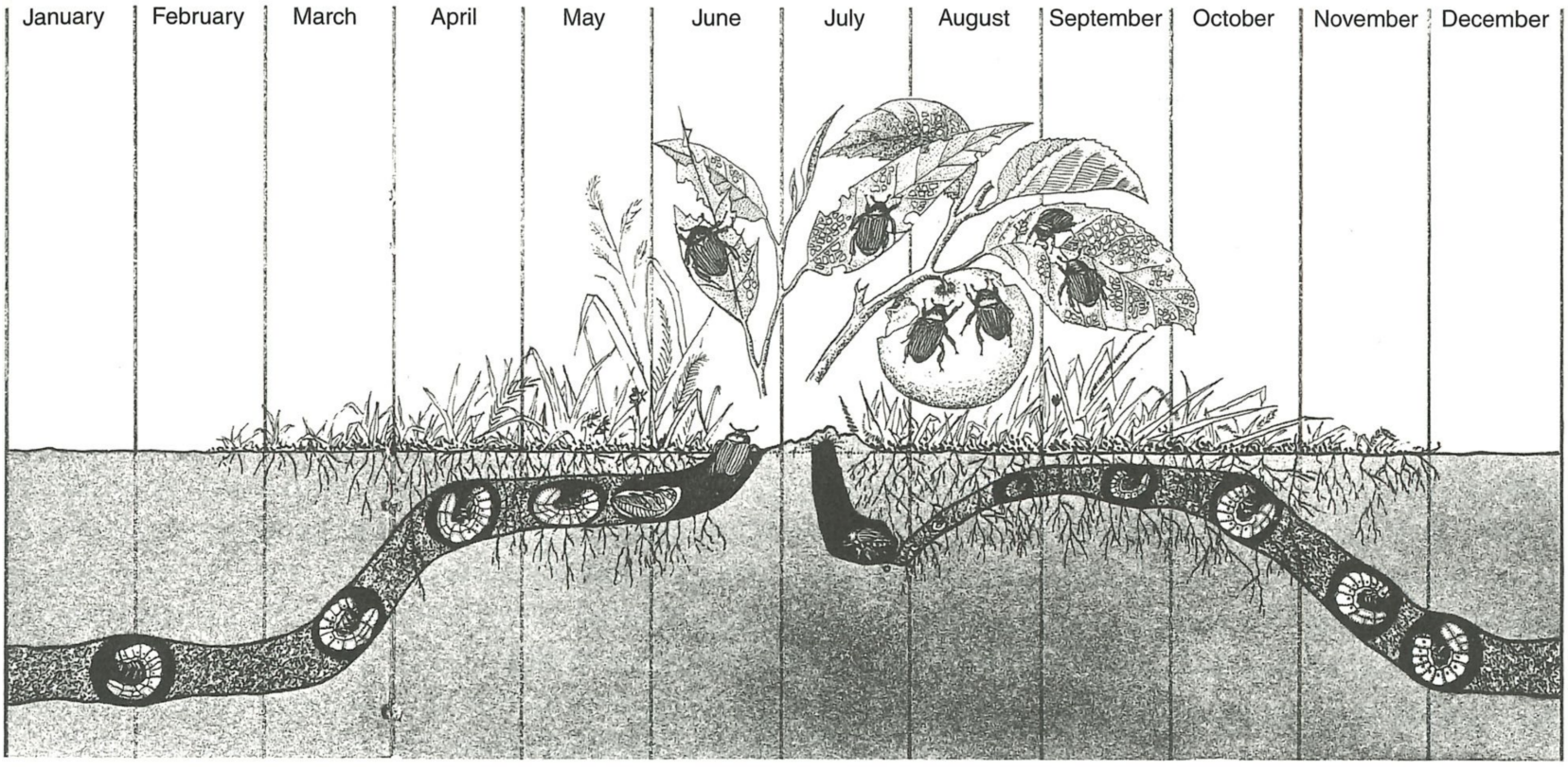


American Education, 1970

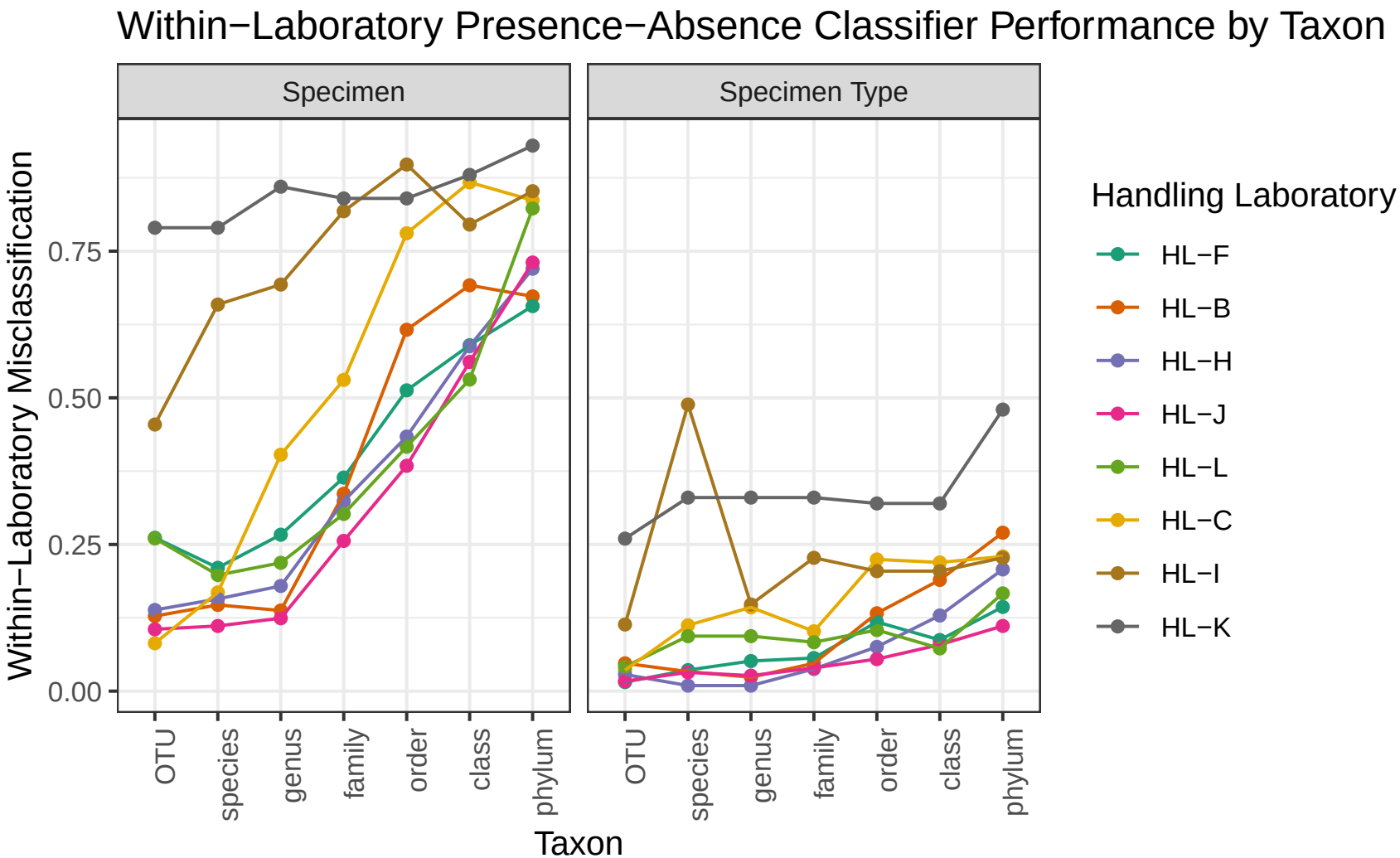
Small multiples



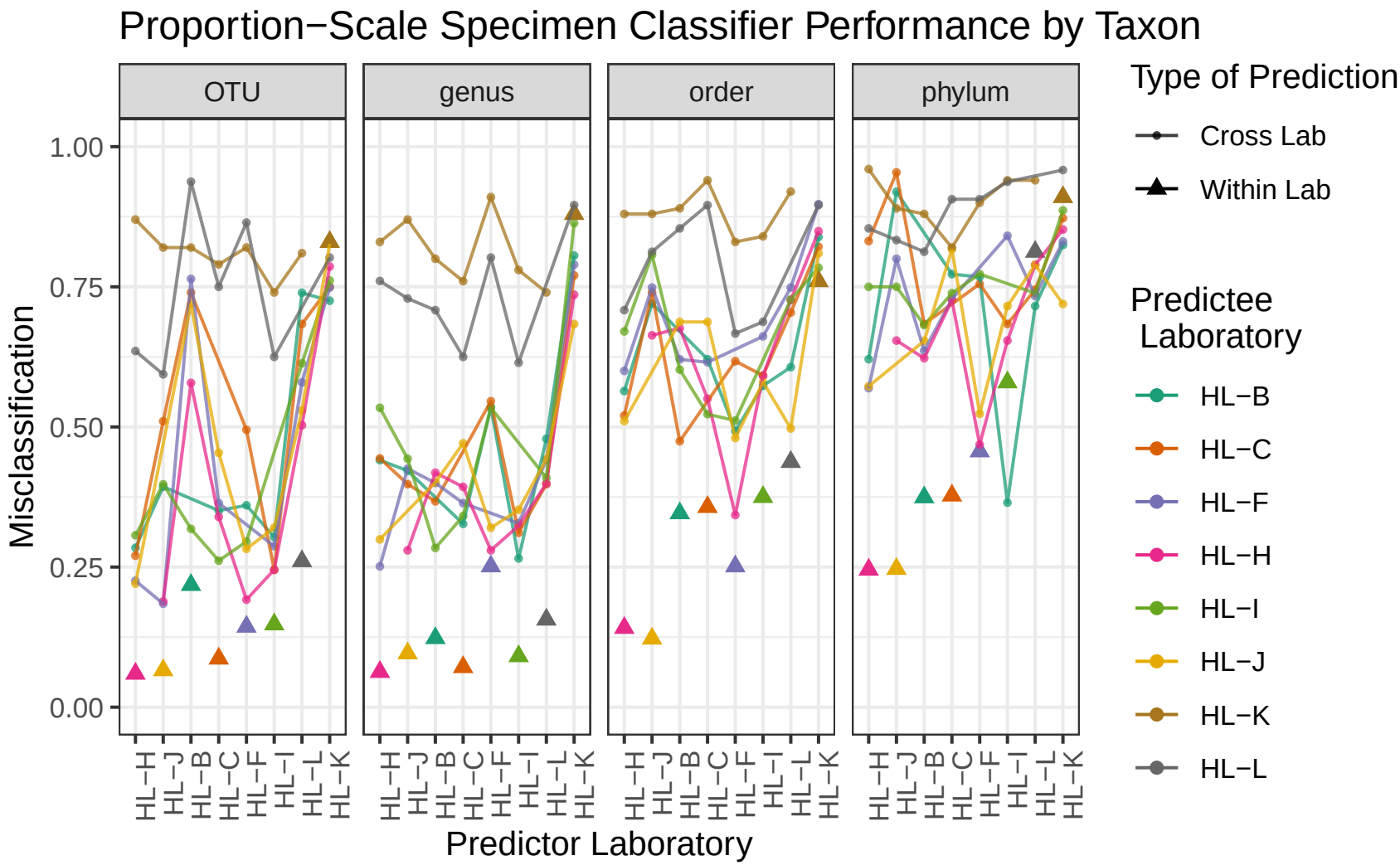
Graphics can be beautiful



Let's critique!



Context is important



ggplot

- `ggplot` is a function available in the `ggplot2` package.
- Graphics are built in layers: a plot is initialised, *then* data is then drawn, *then* annotations are added.
- Annotations include
  - scales
  - labels
  - legends
  - coordinate systems

ggplot

ggplot {ggplot2} R Documentation

Create a new ggplot

Description

`ggplot()` initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage

```
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
```

Arguments

`data`	Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify`. If not specified, must be suppled in each layer added to the plot.
`mapping`	Default list of aesthetic mappings to use for plot. If not specified, must be suppled in each layer added to the plot.
`...`	Other arguments passed on to methods. Not currently used.
`environment`	If an variable defined in the aesthetic mapping is not found in the data, ggplot will look for it in this environment. It defaults to using the environment in which `ggplot()` is called.

Details

`ggplot()` is used to construct the initial plot object, and is almost always followed by + to add component to the plot. There are three common ways to invoke `ggplot`:

1. `ggplot(df, aes(x, y, <other aesthetics>))`
2. `ggplot(df)`
3. `ggplot()`

The first method is recommended if all layers use the same data and the same set of aesthetics, although this method can also be used to add a layer using data from another data frame. See the first example below. The second method specifies the default data frame to use for the plot, but no aesthetics are defined up front. This is useful when one data frame is used predominantly as layers are added, but the aesthetics may vary from one layer to another. The third method initializes a skeleton `ggplot` object which is fleshed out as layers are added. This method is useful when multiple data frames are used to produce different layers, as is often the case in complex graphics.

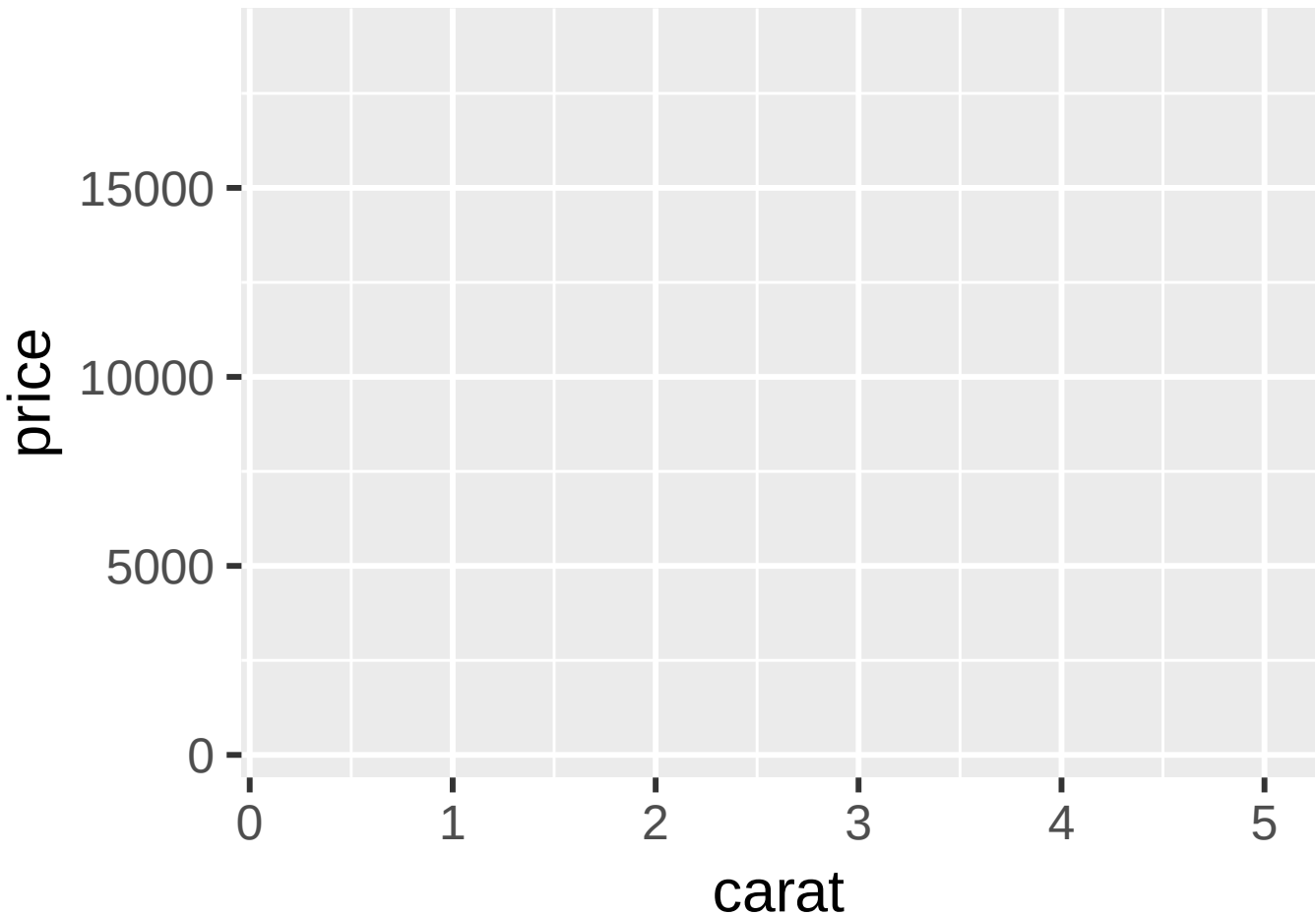
ggplot

```
diamonds %>% as_tibble
```

```
## # A tibble: 53,940 x 10
##    carat cut       color clarity depth table price     x
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl>
##  1 0.23  Ideal     E     SI2      61.5    55   326  3.95
##  2 0.21  Premium   E     SI1      59.8    61   326  3.89
##  3 0.23  Good      E     VS1      56.9    65   327  4.05
##  4 0.290 Premium   I     VS2      62.4    58   334  4.2
##  5 0.31  Good      J     SI2      63.3    58   335  4.34
##  6 0.24  Very Good J     VVS2     62.8    57   336  3.94
##  7 0.24  Very Good I     VVS1     62.3    57   336  3.95
##  8 0.26  Very Good H     SI1      61.9    55   337  4.07
##  9 0.22  Fair      E     VS2      65.1    61   337  3.87
## 10 0.23  Very Good H     VS1      59.4    61   338  4
## # ... with 53,930 more rows
```

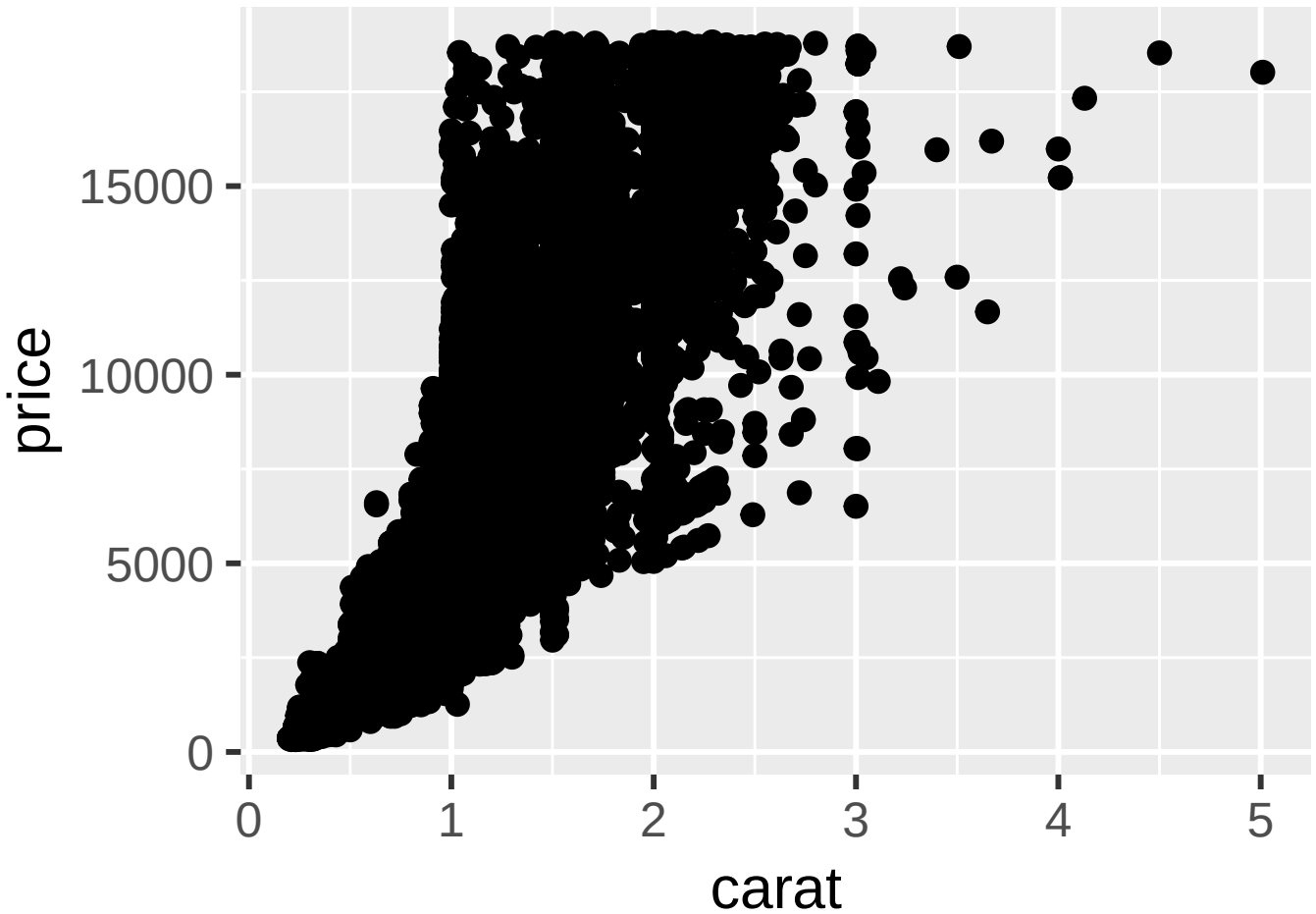
ggplot

```
ggplot(diamonds, aes(x = carat, y = price))
```



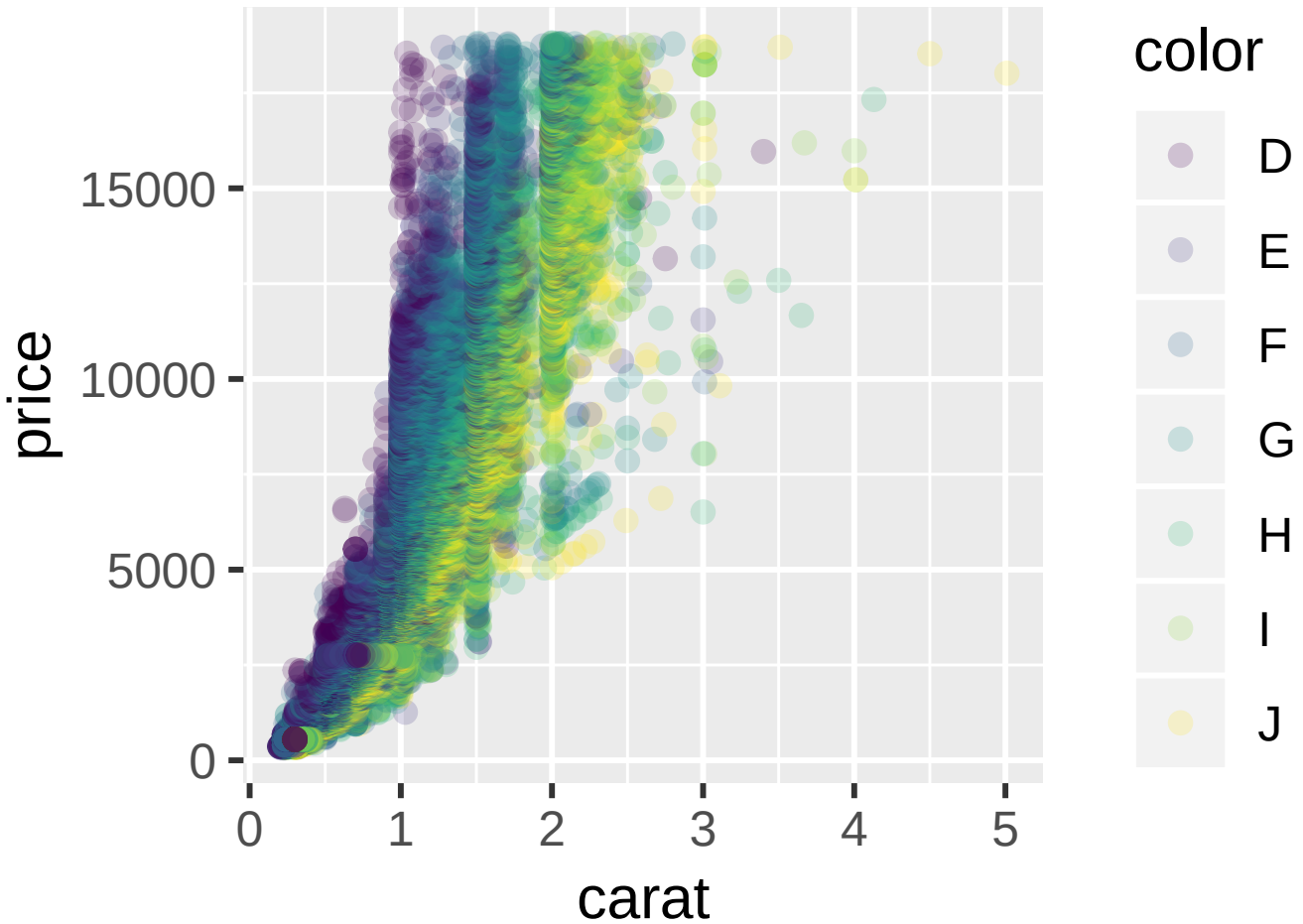
Initialize, *then* add plotting elements

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point()
```



Customise features about the layer

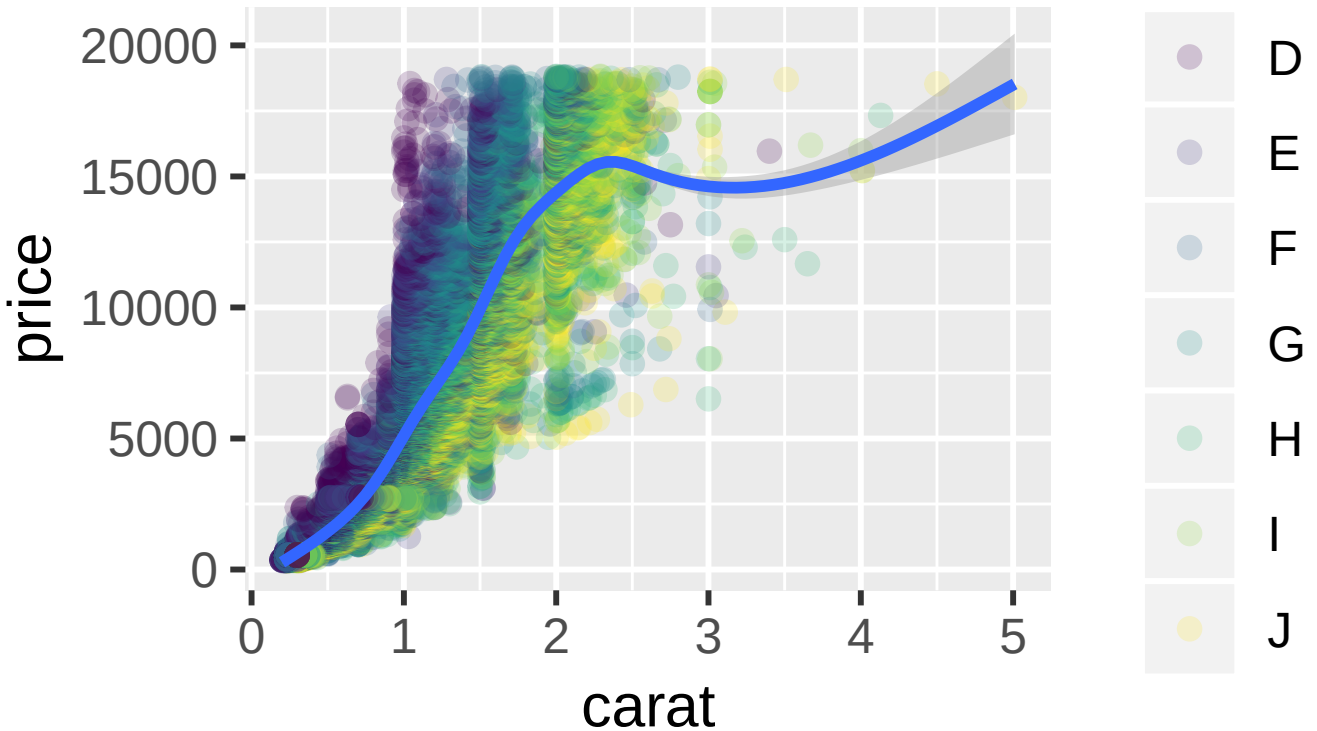
```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(aes(col = color), alpha = 0.2)
```



Add another layer

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(aes(col = color), alpha = 0.2) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs"
```



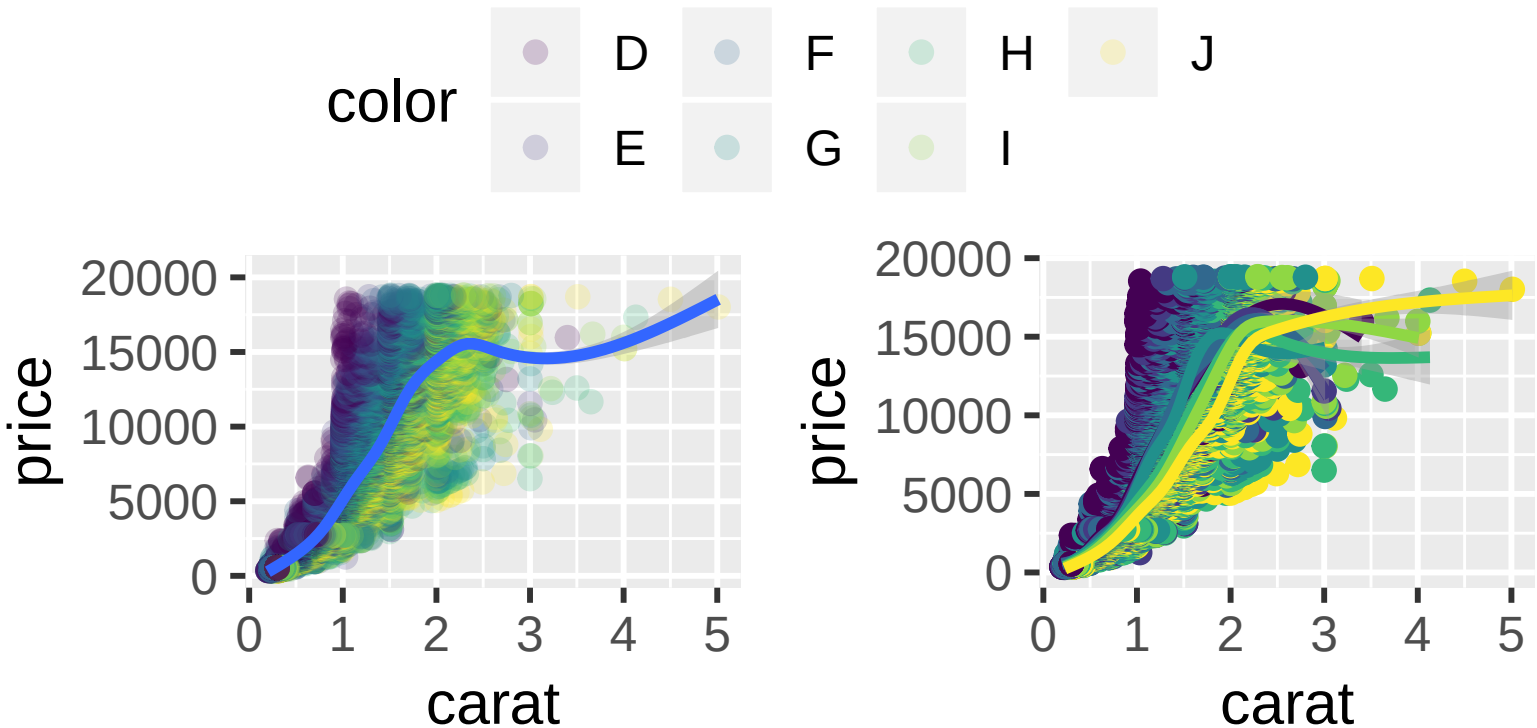
ggplot usually gives sensible results, but, e.g., the authors are not going to be experts in smoothing algorithms. Exercise caution and use your judgement!

What do we think is going to differ?

```
g1 <- ggplot(diamonds,
             aes(x = carat, y = price)) +
  geom_point(aes(col = color), alpha = 0.2) +
  geom_smooth()
g2 <- ggplot(diamonds,
             aes(x = carat, y = price, col = color),
             alpha = 0.2) +
  geom_point() +
  geom_smooth()
```

What do we think is going to differ?

```
ggpubr::ggarrange(g1, g2, common.legend=TRUE)
```



Equivalent calls

```
ggplot(diamonds, aes(x = carat, y = price)) + geom_point()
ggplot(diamonds) + geom_point(aes(x = carat, y = price))
ggplot(diamonds, aes(x = carat)) + geom_point(aes(y = price))
```

Which is best?

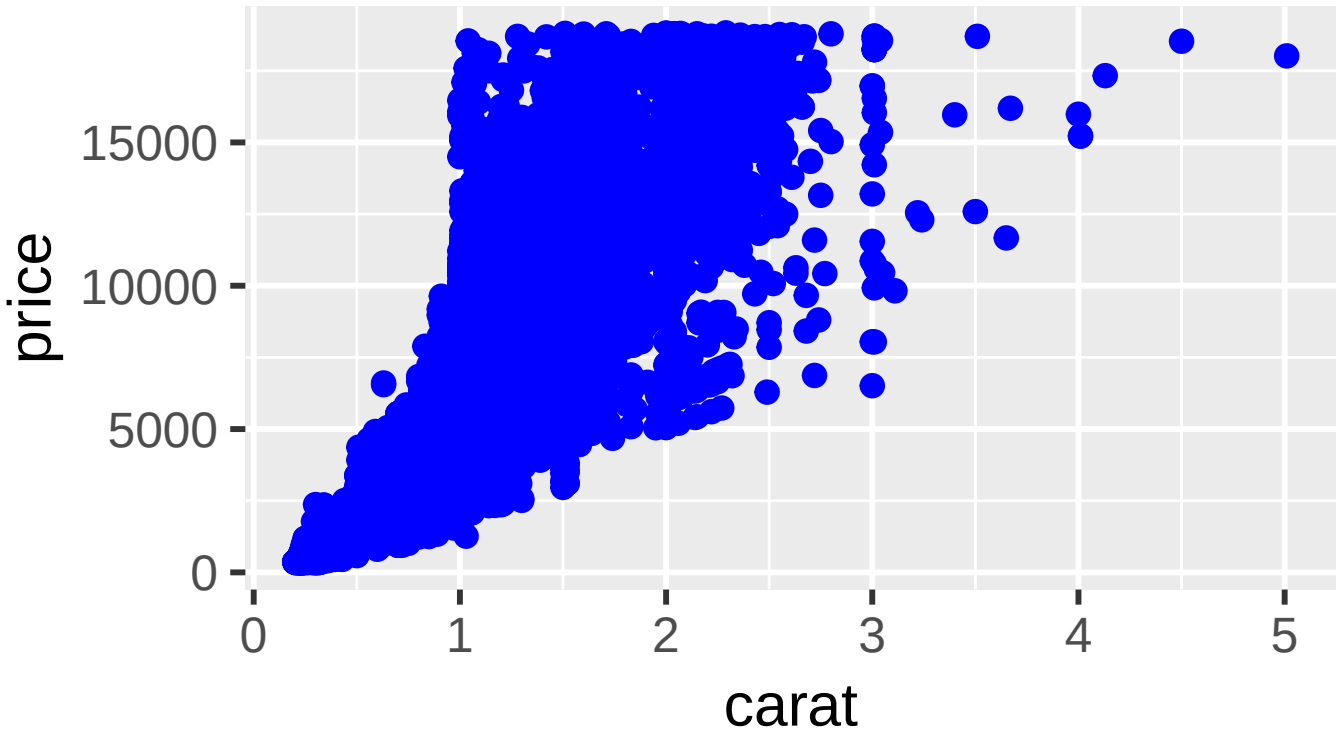
Layering objects

Arguments to a ggplot call:

- The first argument to a ggplot is the data frame (or tibble)
- You can fix the aesthetics with `aes()` OR you can add aesthetics in layers
- Any aspect of a plot that you want to vary based on a variable needs to be wrapped in a `aes()` call

aes() calls

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(col = "blue")
```

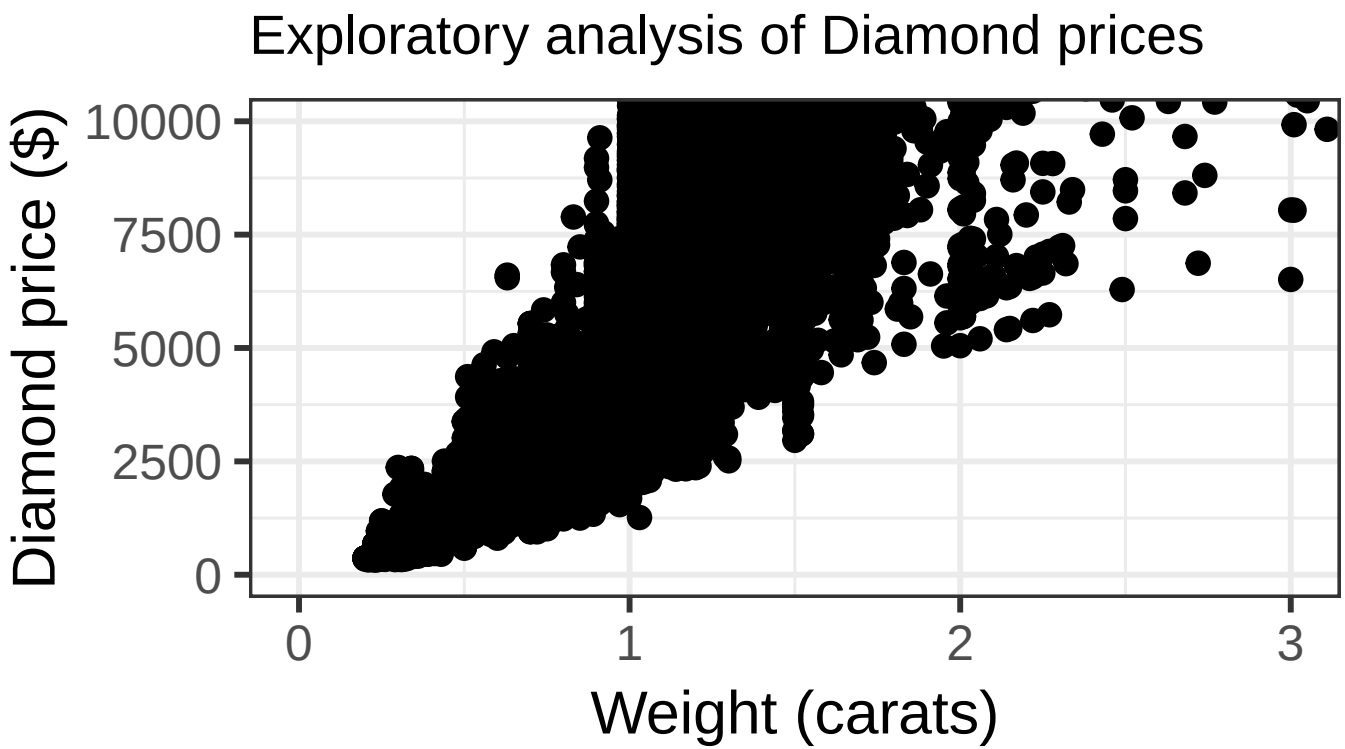


The following results in an error. Why?

```
ggplot(diamonds, aes(x=carat, y=price))+geom_point(col=color)
```

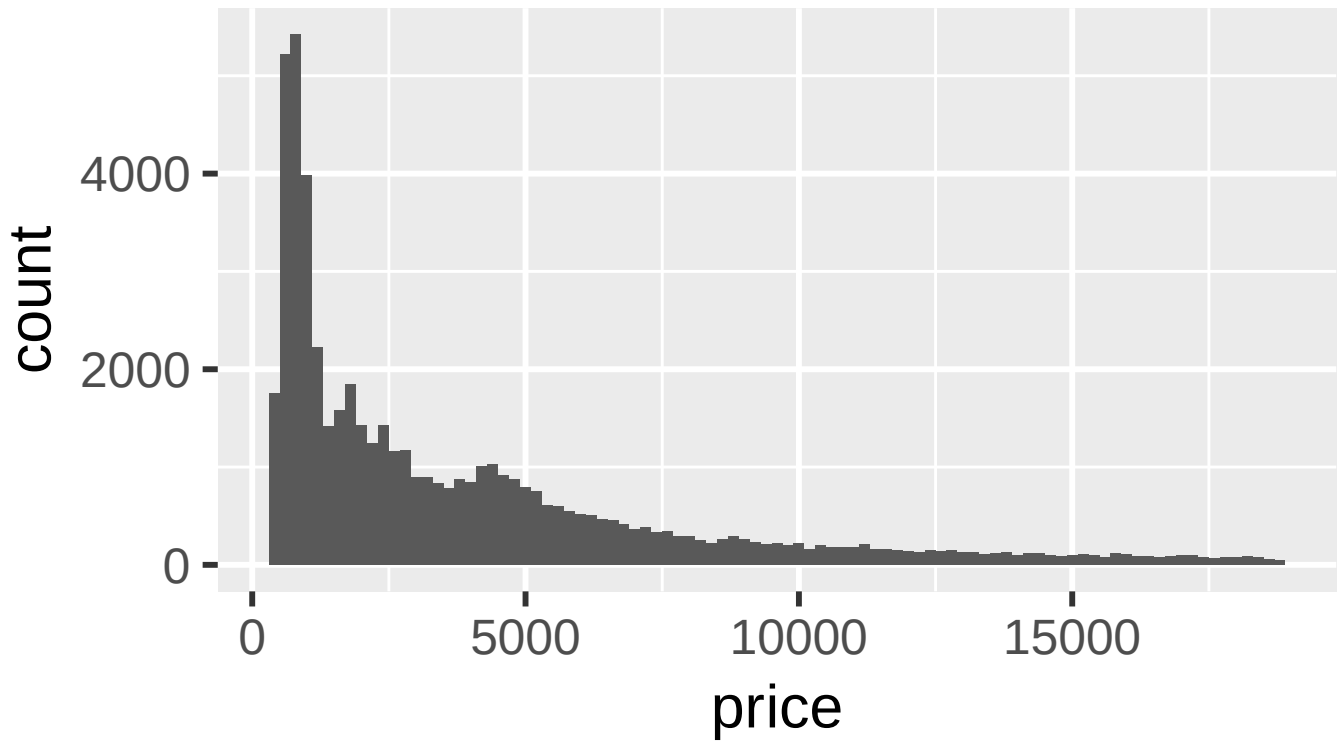
More layers

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point() + theme_bw() +
  labs(x="Weight (carats)", y="Diamond price ($)") +
  ggtitle("Exploratory analysis of Diamond prices") +
  coord_cartesian(xlim = c(0, 3), ylim=c(0, 10000)) +
  theme(plot.title = element_text(size = 10))
```



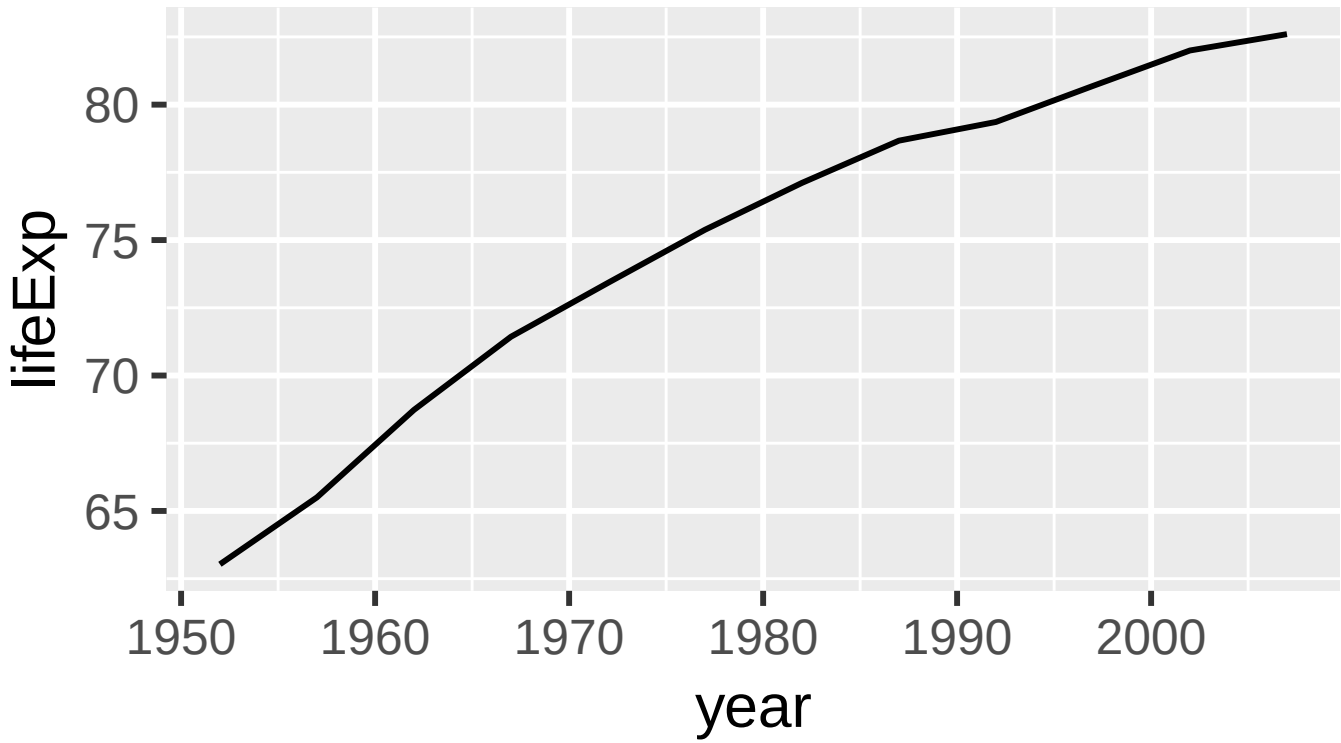
Histograms

```
ggplot(diamonds, aes(x = price)) +
  geom_histogram(binwidth = 200)
```



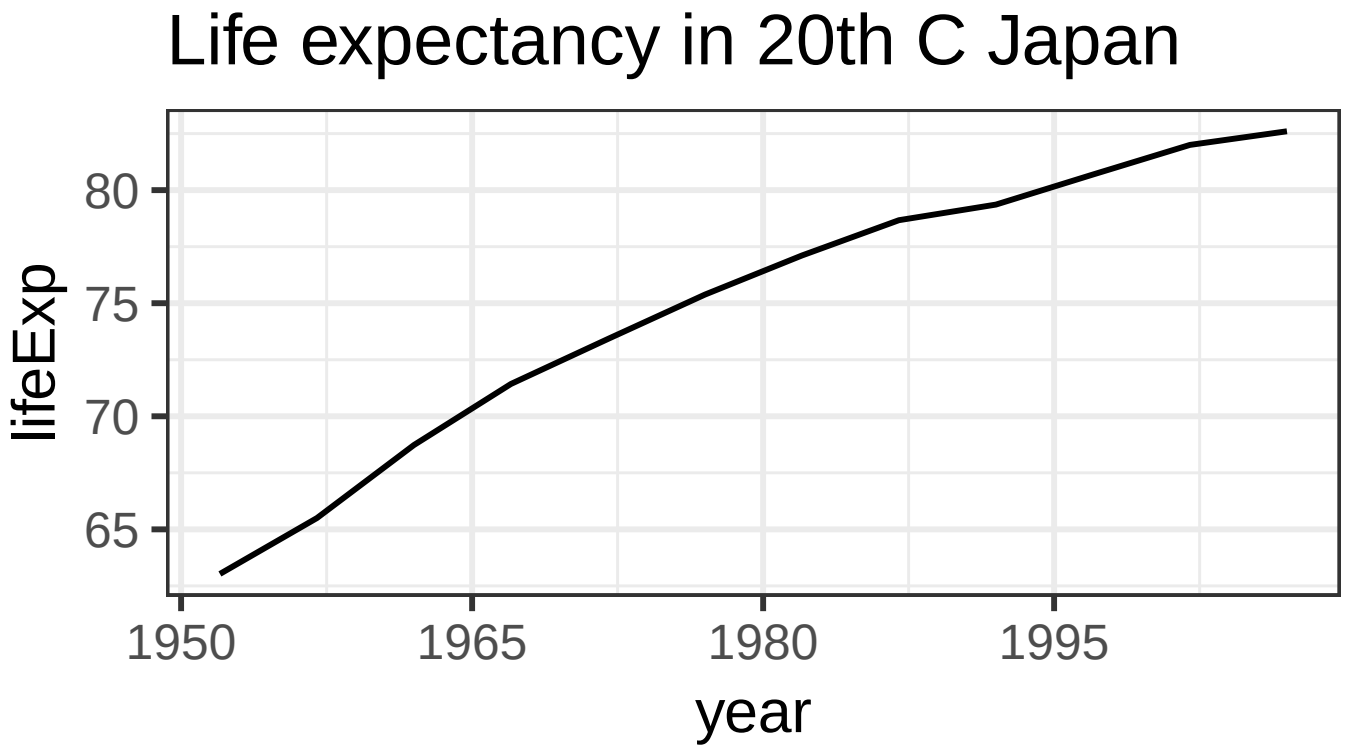
Timeseries

```
g <- ggplot(gapminder %>% filter(country == "Japan"),
            aes(x = year, y = lifeExp)) + geom_line()
g
```



Storing and modifying

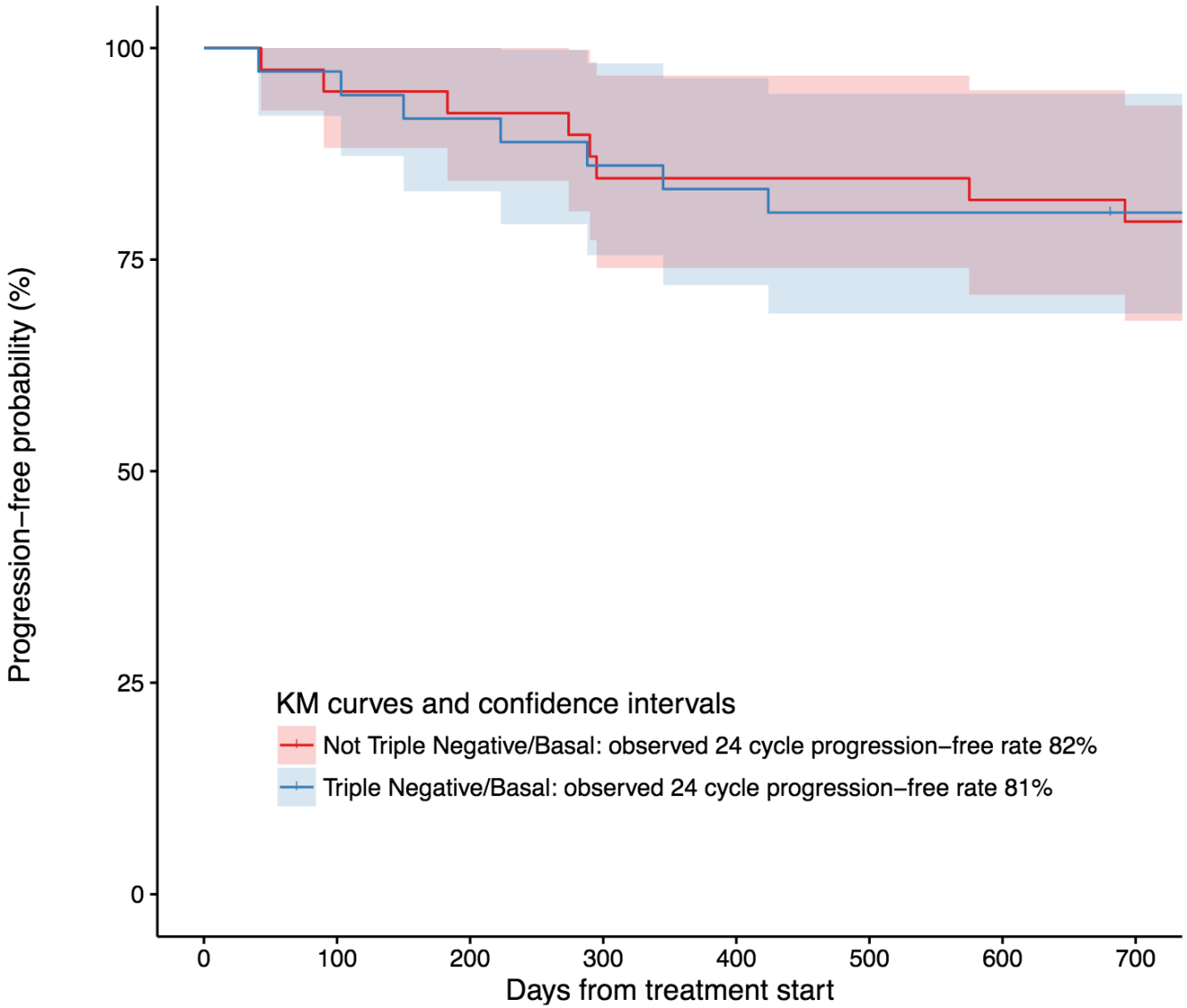
```
g + scale_x_continuous(breaks = seq(1950, 2011, 15)) +
  theme_bw() +
  ggtitle("Life expectancy in 20th C Japan")
```



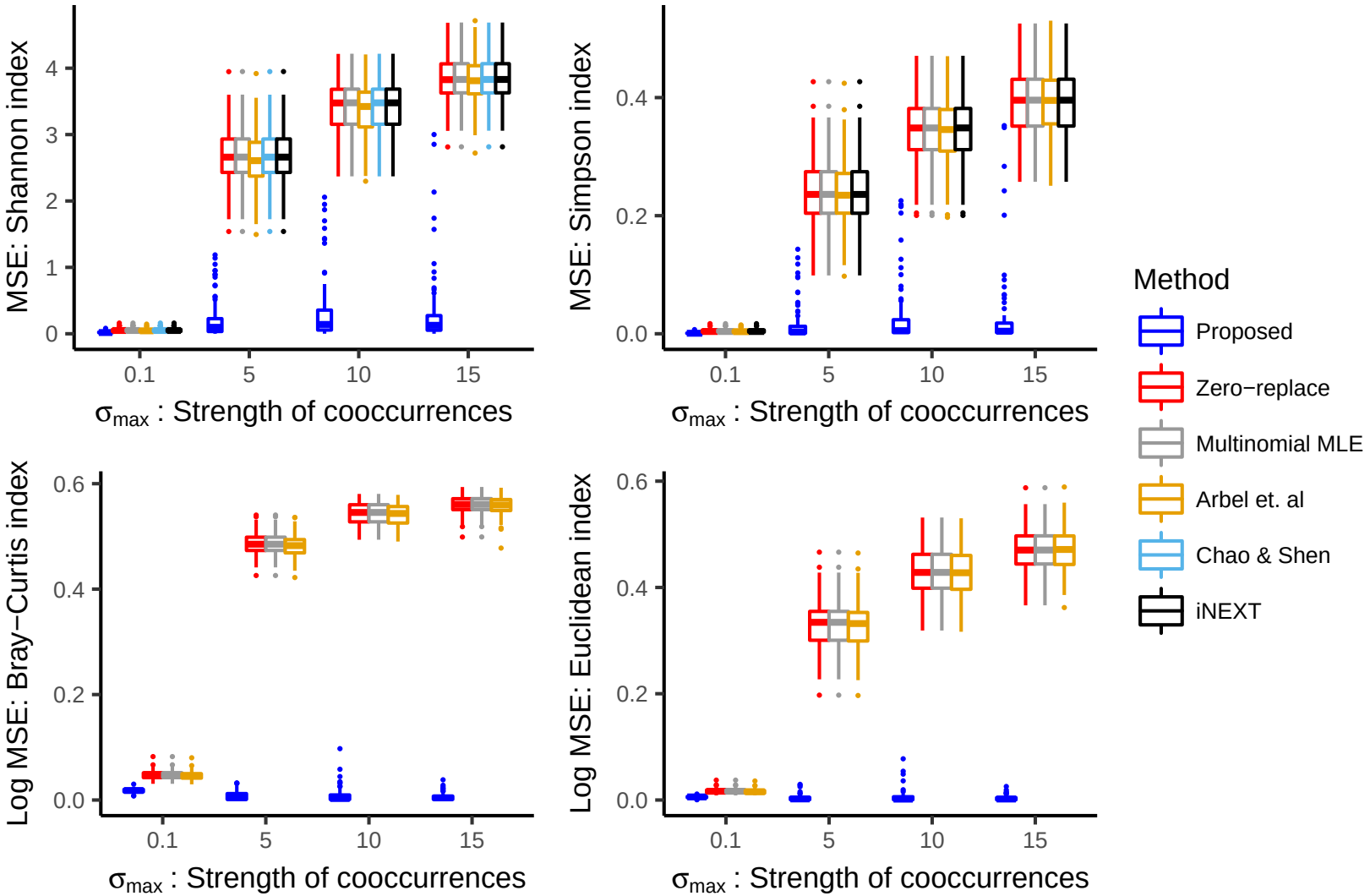
Creating and plotting ggplot objects is not always fast; this may help

Showing intervals using transparency

The most important information should be the most clear, the next most important information should be the next most clear, etc..



Infinitely customisable



All the layers

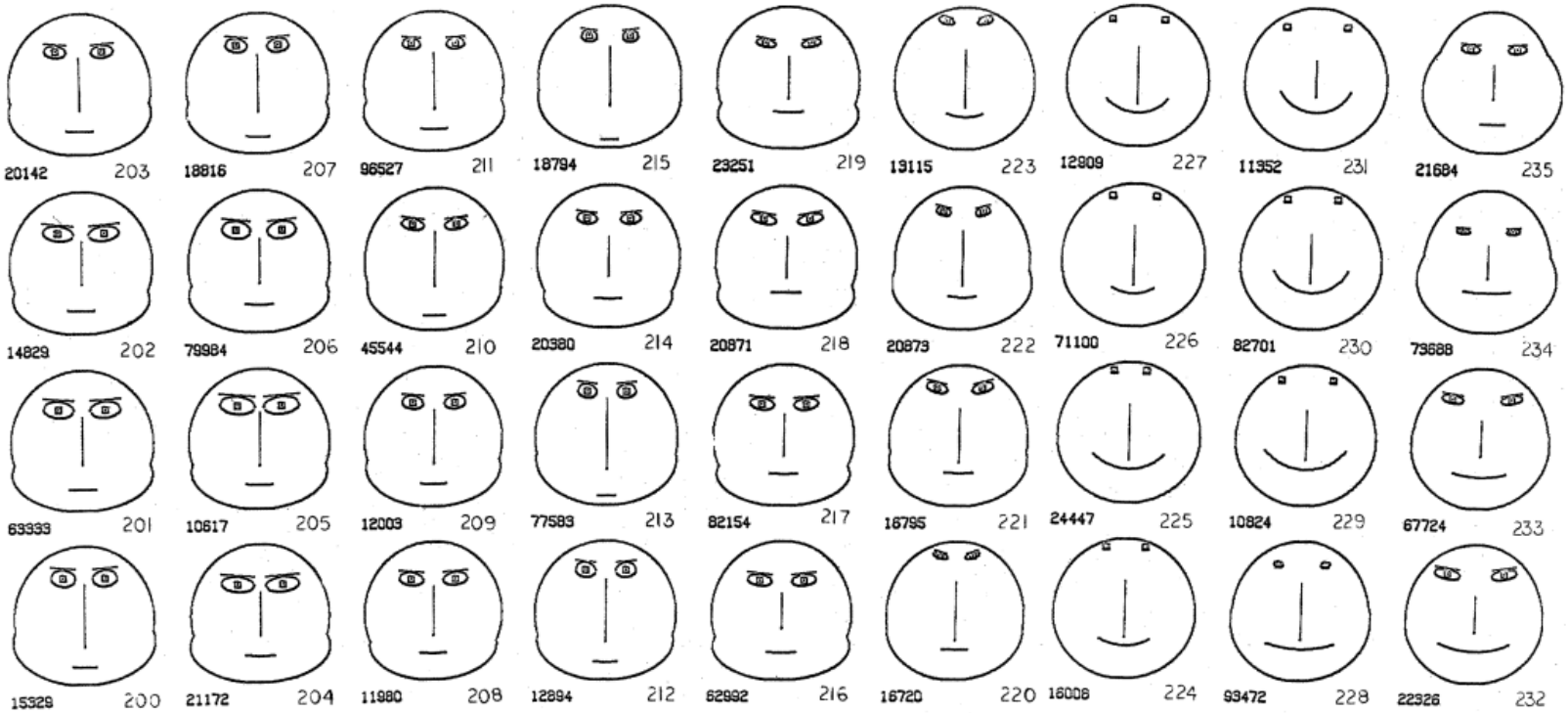
```
euc_plot <- ggplot(full_data_frame[order(full_data_frame$sigma_max), ],
                   aes(x = sigma_max_char, y = mse_loss_euclidean_subse
  geom_boxplot(outlier.size=0.1) +
  xlab(label=expression(sigma[max]~": Strength of cooccurrences")) +
  ylab("Log MSE: Euclidean index") +
  theme_bw() +
  theme(text = element_text(size = 9),
        panel.border = element_blank(), panel.grid.major = element_blan
        panel.grid.minor = element_blank(), axis.line = element_line(co
  scale_color_manual(values=c("blue", "red", "#999999", "#E69F00", "#56

ggpubr::ggarrange(shannon_plot, simpson_plot, bc_plot, euc_plot, ncol=2
                  common.legend = T, legend="right")

ggsave("vary_sigma_max_beta_sd_1_n_20_p_2_q_20_sigma_min_0_01_all_share
```

Catering to our senses

2. FACES FOR 53 GEOLOGICAL SPECIMENS OF EXAMPLE 2



Final tips

- Never, ever show a pie chart to a statistical audience... or the internet!
  - Exploding or otherwise
- Align plots with similar axes
- Defer less important points to appendices
  - Use figures **where appropriate**. Would a table suffice?
- Critique your own figures
  - Especially before public talks/conferences

Highly recommended reading:

"Graphical Display of Quantitative Information" by Edward Tufte

Limitations of `ggplot`

- *Forces you to use good coding practice* (?)
  - Data should all come from the same data frame
  - Painful at first but better than a retraction!
- Constrained to two-dimensional (see `plot3d`, `scatterplot3d`)
- No graphs/trees (`igraph`)
- Static (`shiny`)
- Limited use for non-statistical graphics (Illustrator, Photoshop, Inkscape)

Other things

- `1L` in `R`

Coming up

- Reminder: `git pull` – any time you work on any file in a directory with version control!
  - You won't get your comments on your homework if you don't
- Homework 3 due next Wednesday at the usual time in the usual way