a publication-ready scatterplot with ggplot2

We recently had a discussion in the departement about visualisation for science. The main concern about ggplot2 seemed to be the steep learning curve. Other programs might produce the required graphs quicker, without coding, even interactive. However, I think with a little bit of training and keeping up to date with the new developments in ggplot, it will be the last visualization software you have to learn. Moreover, it will give you freedom of the mind by making it possible to create whatever graphs you want and even get inspired by the newest developments in visualising data.

Here, I'd like to give a quick overview and an idea about how to produce a publication ready graph.

Data format

When learning ggplot2, I found that one of the hardest parts was to actually have the data in the right format. If the format is correct, ggplot2 does everything surprisingly well. Like all of Hadley Wickham's packages, it works with tidy data. This means, that every observation is a row, and every variable is a column. This sounds straightforward, but is surprisingly often messed up among datasets. We are using the iris dataset which contains measurements for three different plant species.

```
data(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Packages

Before we start plotting, we load some package which I find make life quite easy to start with ggplot. Most people have a hard time customizing the `theme` of a ggplot. The `theme` essentially configures the layout, i.e. things like axes and labels. An excellent package to make this easy and to have a common theme across all your plot is `ggthemr`. Check out the ggthemr GitHub repo for a detailed manual.

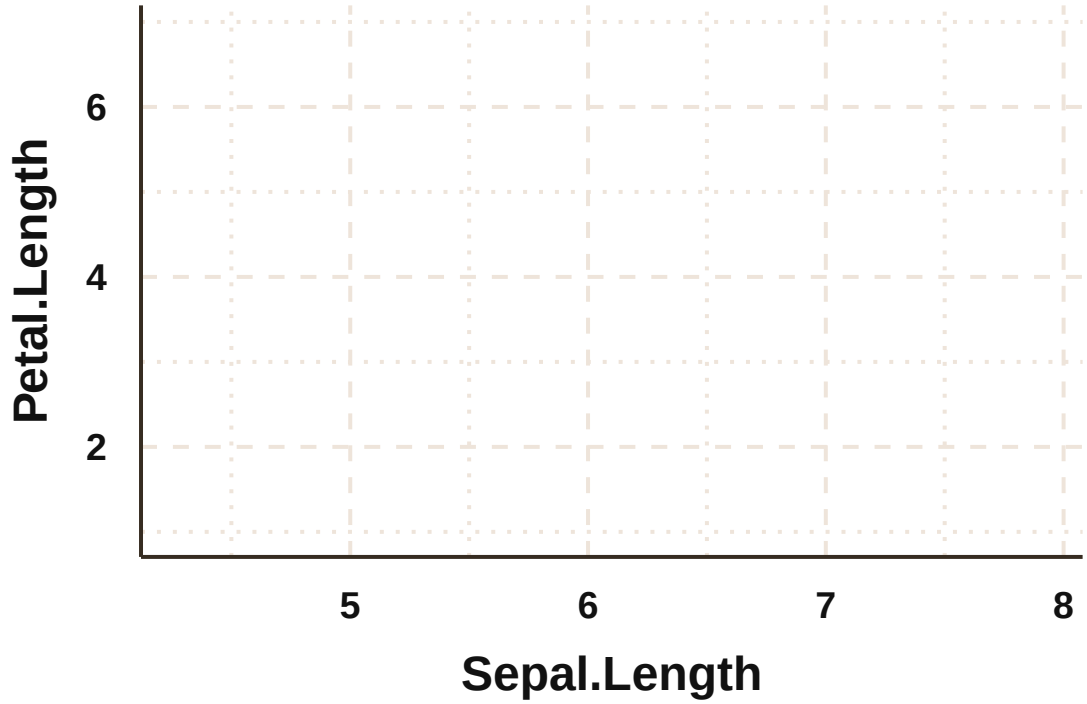
```
# devtools::install_github('cttobin/ggthemr')
library(ggplot2)
library(ggthemr)
```

Now we are setting all our layouts once at the beginning. The palette specifies the colors, the layout things like axis and lines and the spacing gives the plot some space between axes and labels. One thing I found really practical is text_size, which automatically adjust all text in the plot according to a reference size.

```
ggthemr(palette = "fresh", layout = "scientific", spacing = 3,
    line_weight = 0.7, text_size = 18, type = "outer")
```

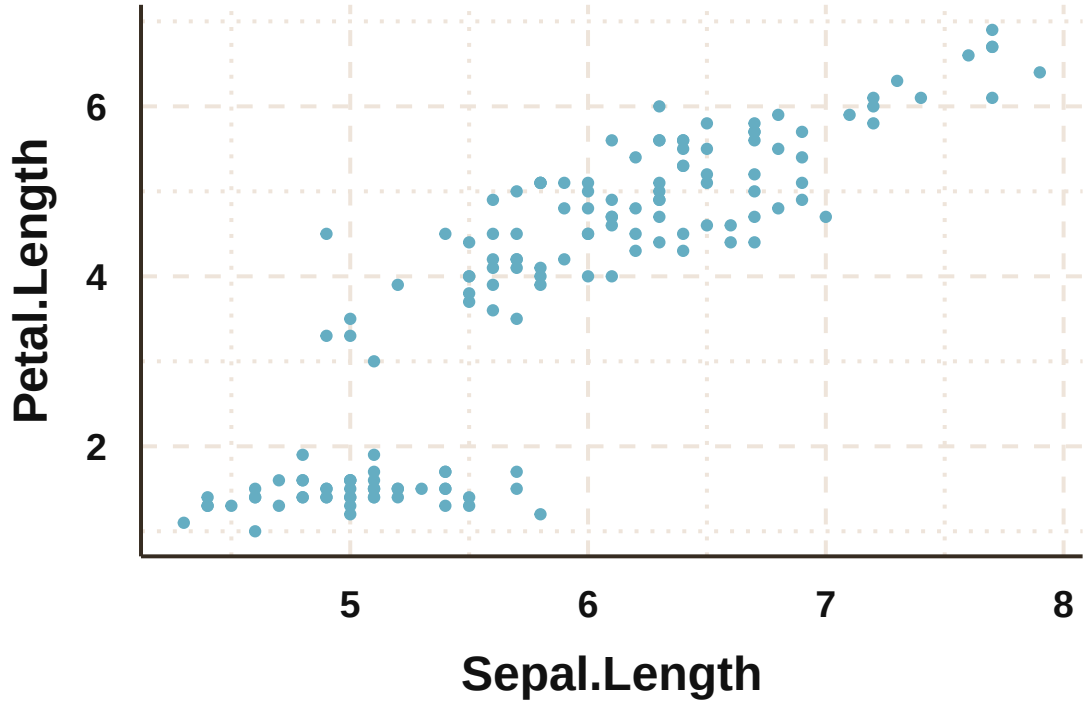
So how does ggplot work? Well, it is not entirely straightforward, which is why some people are not yet using it I guess. We first produce the basis of the plot, i.e. we define the `data.frame` where all the data is stored, and we define the so-called aesthetics (`aes()`). This defines which variables will be plotted. Here we want to have the Sepal.Length on the x-axis and the Petal.Length on the y axis. Let's do it.

```
p1 <- ggplot(data = iris, aes(x = Sepal.Length, y = Petal.Length))
p1
```



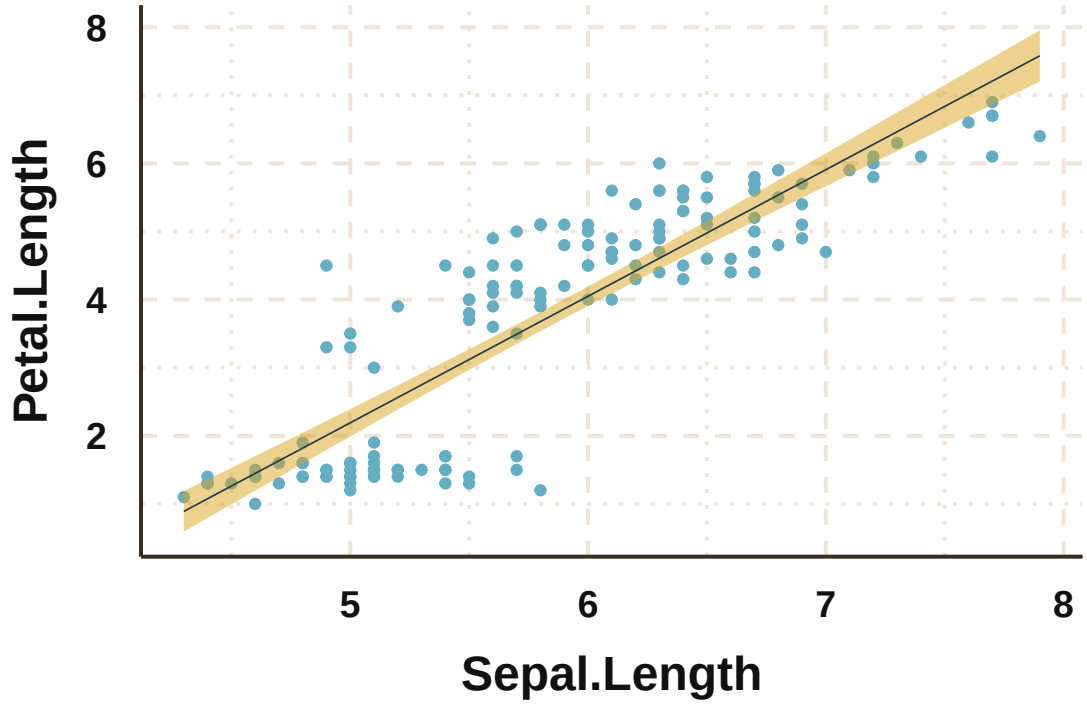
No data was plotted! Why is this? It's because we didn't define yet **how the data should be represented**. This is the whole point about the layered grammar of graphics! So do we want points, bars, boxplots or what? We have two continous variables, so let's try it with a scatterplot.

```
p1 + geom_point()
```



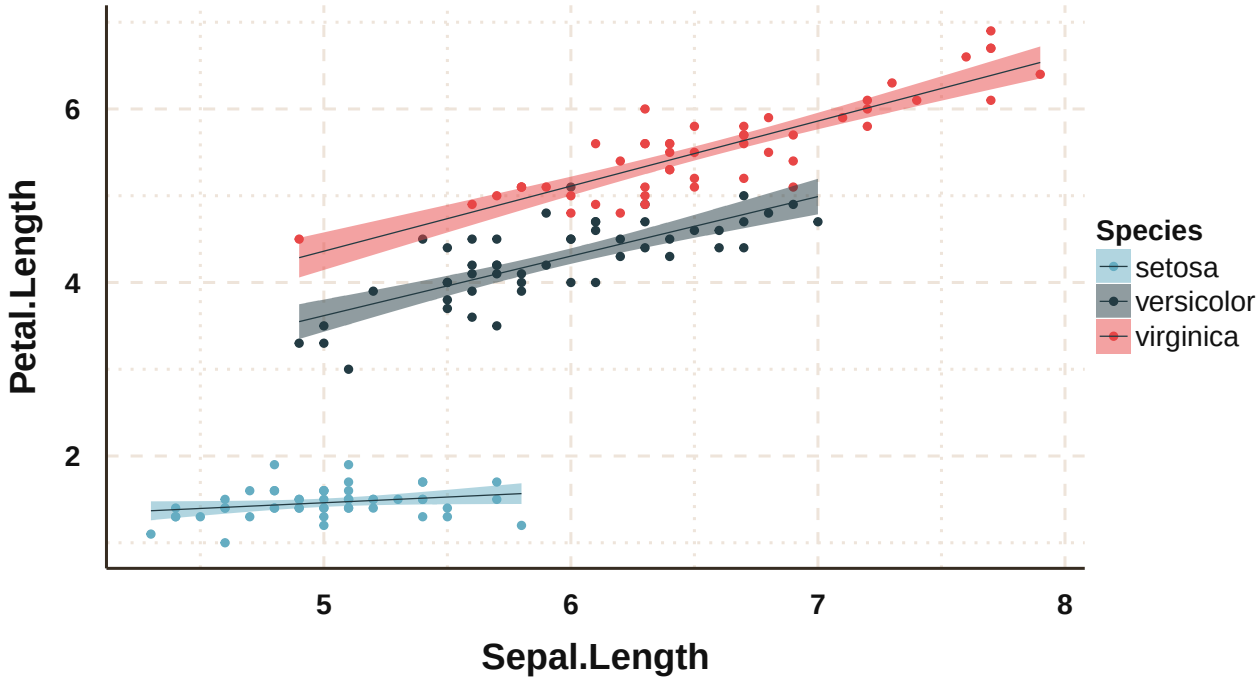
This looks a bit more like a proper plot. Now we are adding a trend line to the plot. This can be just a smoother or the regression line of a linear model. Here, we choose the linear model (lm) method. This works equivalent to plotting the points, we essentially add another layer, 'geom_smooth', to the graph.

```
p1 +
  geom_point() +
  geom_smooth(method = "lm", level = 0.95, fill = "goldenrod", size = 0.3, alpha = 0.5)
```



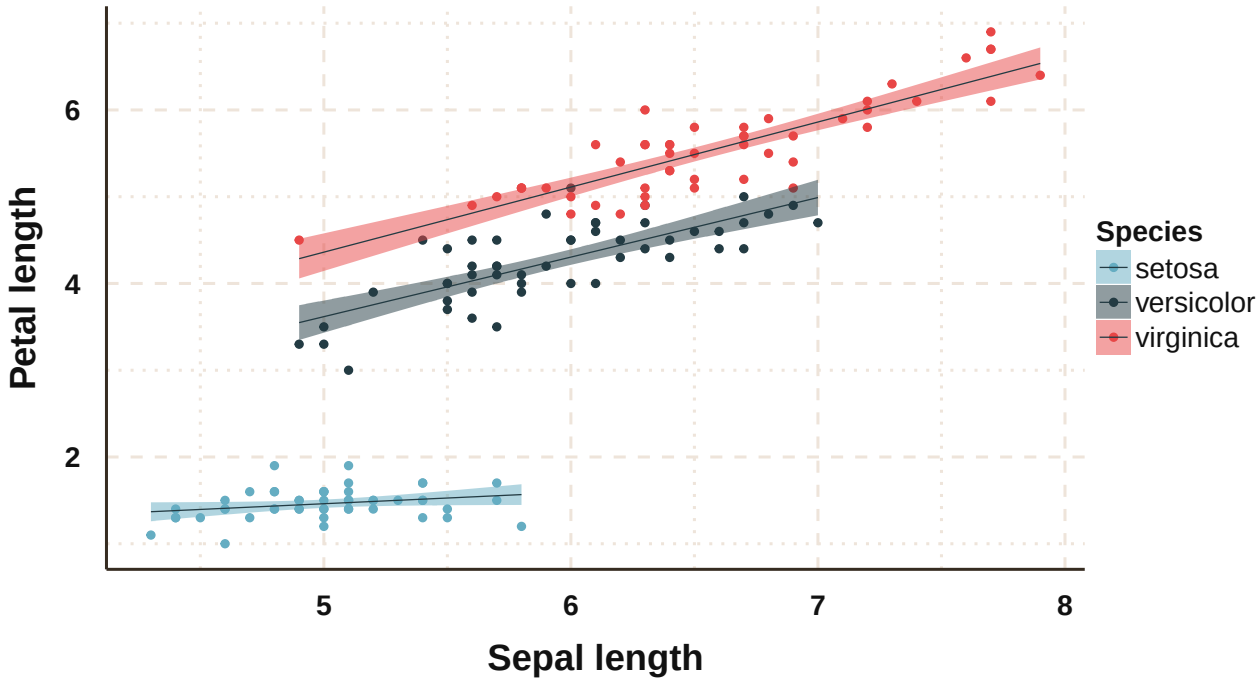
Often you want to highlight group-specific trends in your data. Here, we are looking at three different plant species, which are specified in the factor `species` in the iris data.frame. `species` is a variable, which is why we need to specify it as an aesthetic again. We add the `aes(color = Species))` argument to the `geom_point` function to map the point color according to the species. We do the same for the `geom_smooth` to get regression lines per Species. However, here we use `aes(fill= Species)` to fill the confidence bands around the regression line in the same color that we used for the points.

```
p1 +
    geom_point(aes(color = Species)) +
    geom_smooth(method = "lm", level = 0.95, size = 0.3,
                alpha = 0.5, aes(fill= Species))
```



Now we can do some finetuning. First, we change the axis labels and give a title to the plot

```
p1 +
    geom_point(aes(color = Species)) +
    geom_smooth(method = "lm", level = 0.95, size = 0.3,
                alpha = 0.5, aes(fill= Species)) +
    xlab("Sepal length") +
    ylab("Petal length")
```



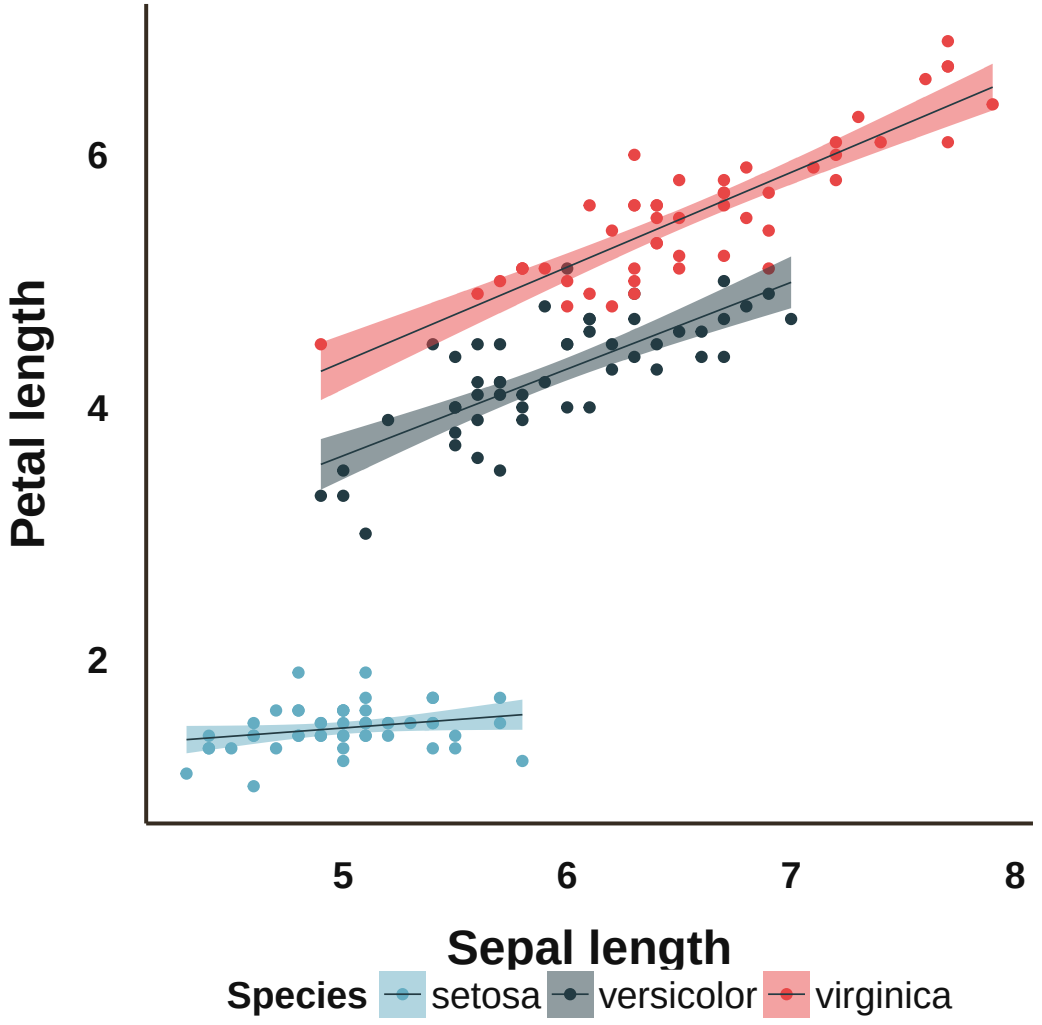
We also might want to give all the labels a bit more breathing space. And I'm not sure whether I like the gridlines. This means we change the `ggthemr` template form the beginning. Instead, we use the clean layout

and a larger spacing.

```
ggthemr(palette = "fresh", layout = "clean", spacing = 3.5,
    line_weight = 0.7, text_size = 18, type = "outer")
```

And this is how the graph looks now.

```
p1 +
    geom_point(aes(color = Species)) +
    geom_smooth(method = "lm", level = 0.95, size = 0.3,
                alpha = 0.5, aes(fill= Species)) +
    xlab("Sepal length") +
    ylab("Petal length") +
    legend_bottom()
```



That's it!

{% include disqus.html %}