ggplot2templating_readme

Kristen Hansen

2024-02-26

Setting a Theme

This is the section where you would set the font, size, and all text in the plot, legend, and background of the plotting.

Pre-Fabricated Themes

There are many themes for ggplot2.

Eight themes are built in to ggplot2 1.1.0:

1. theme_grey(), the signature ggplot2 theme with a light grey background and white gridlines
2. theme_bw(): a variation on theme_grey() that uses a white background and thin grey grid lines.
3. theme_linedraw(): a theme with only black lines of various widths on white backgrounds, reminiscent of a line drawing.
4. theme_light(): similar to theme_linedraw() but with light grey lines and axes, to direct more attention towards the data.
5. theme_dark(): the dark cousin of theme_light(), with similar line sizes but a dark background. Useful to make thin coloured lines pop out.
6. theme_minimal(): a minimalistic theme with no background annotations.
7. theme_classic(): a classic-looking theme, with x and y axis lines and no gridlines.
8. theme_void(): a completely empty theme.

Additional Pre-built themes are available from packages like *ggthemes*, which has 15 themes. ## Legends

The legend elements control the appearance of all legends. You can also modify the appearance of individual legends by modifying the same elements in guide_legend() or guide_colourbar().

Element Setter Description

legend.background element_rect() legend background legend.key element_rect() background of legend keys legend.key.size unit() legend key size legend.key.height unit() legend key height legend.key.width unit() legend key width legend.margin unit() legend margin legend.text element_text() legend labels legend.text.align 0–1 legend label alignment (0 = right, 1 = left) legend.title element_text() legend name legend.title.align 0–1 legend name alignment (0 = right, 1 = left)

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```r
library(grDevices)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

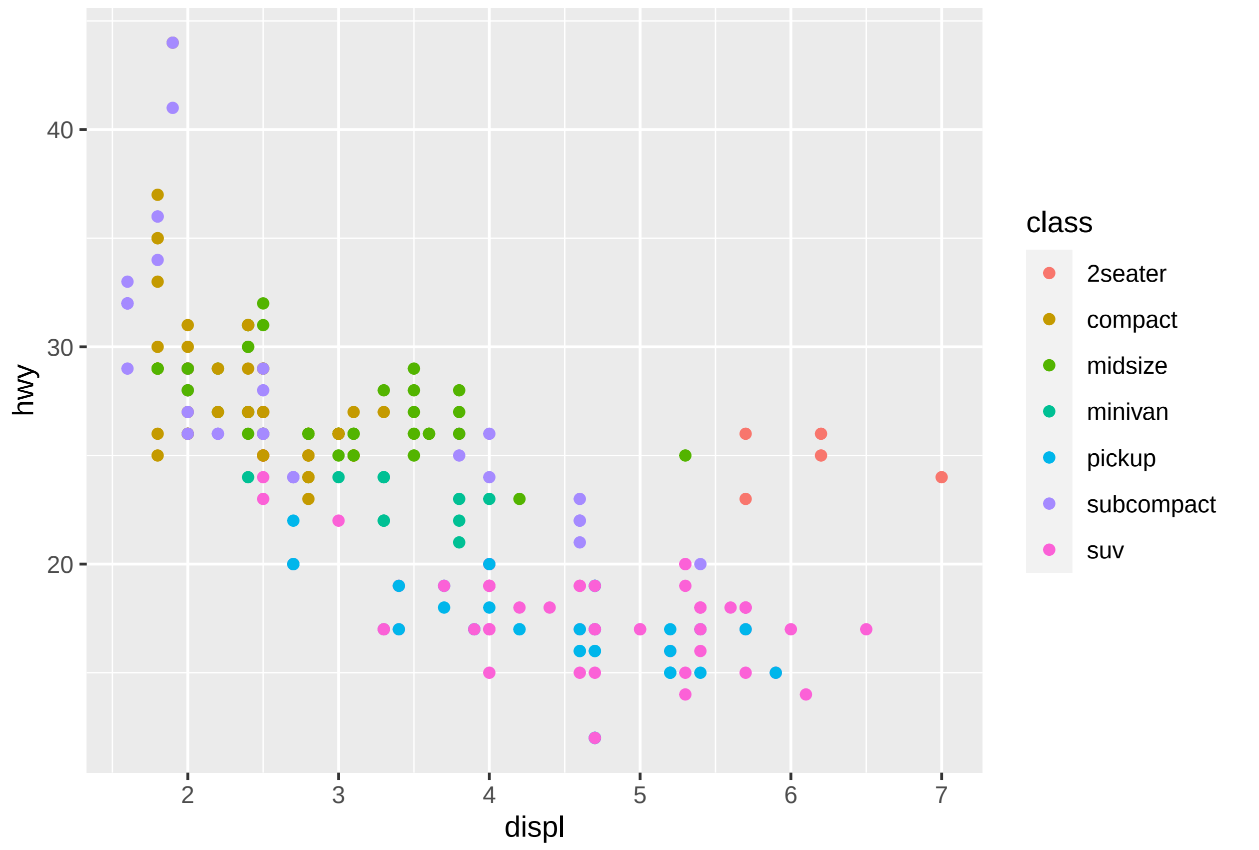
Theme Example

```r
p = ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
## Clean theme example:
  saved_theme <- theme_grey() +
  theme(
    # Text
    text = element_text(family = "sans"),
    # Axes
    axis.title = element_text(face = "bold", size = 8),
    axis.line = element_line(color="black", linewidth = .25),
    axis.text.y = element_text(color="black", size = 6),
    axis.text.x = element_text(color="black", size = 6),
    axis.ticks = element_line(color="black", linewidth = .25),
    # Title
    plot.title = element_text(face = "italic", size = 15, hjust = 0.5),
    # Legends
    legend.title =  element_text(color = "black", size = 6),
    legend.position = "bottom",
    legend.text = element_text(size = 5),
    legend.key.size = unit(0.4, "lines"),
    legend.box.spacing = unit(0, "cm"),
    legend.background = element_rect(fill = "gray95"),
    # Panel
    panel.border = element_blank(),
    panel.background = element_rect(fill="black"),
    panel.grid.major = element_line(color="grey", linewidth = .10),
    panel.grid.minor = element_line(color="grey", linewidth = .10)
)
  ## save it as RDS
saved_theme %>% saveRDS('saved_theme.rds')
rm(saved_theme)

## Margin sizes according to Science formatting requirements
# Widths

# 1 column = 5.5 cm / 2.25 in
cw1 <- 2.25
# 2 column = 12 cm / 4.75 in
cw2 <- 4.75
# 3 column = 18.3 cm / 7.25 in
cw3 <- 7.25
```

Example Plot

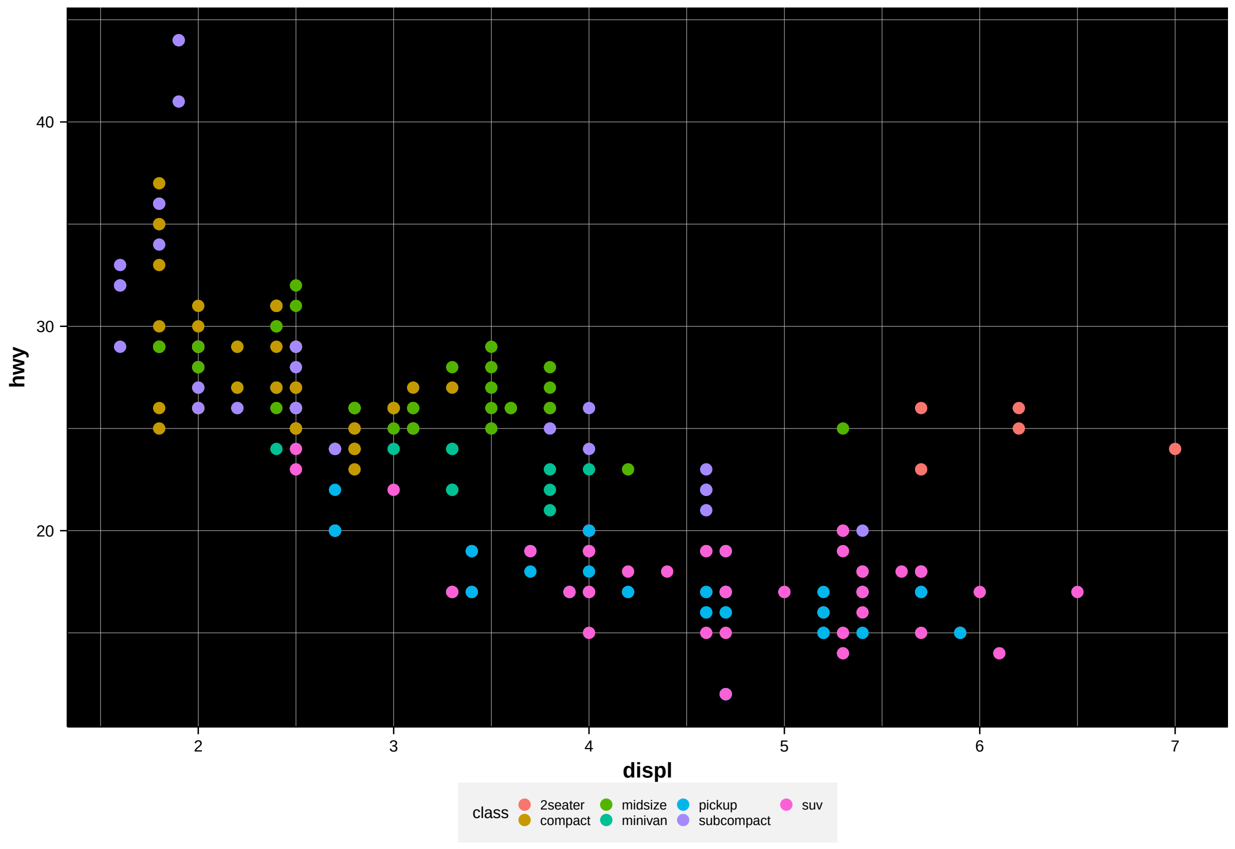
Below is a dummy plot: First we can see what the plot looks like with the default ggplot2 theming. A grey background, white axis lines and bright colors.

```
#Default GGPLOT2 Theming
p
```



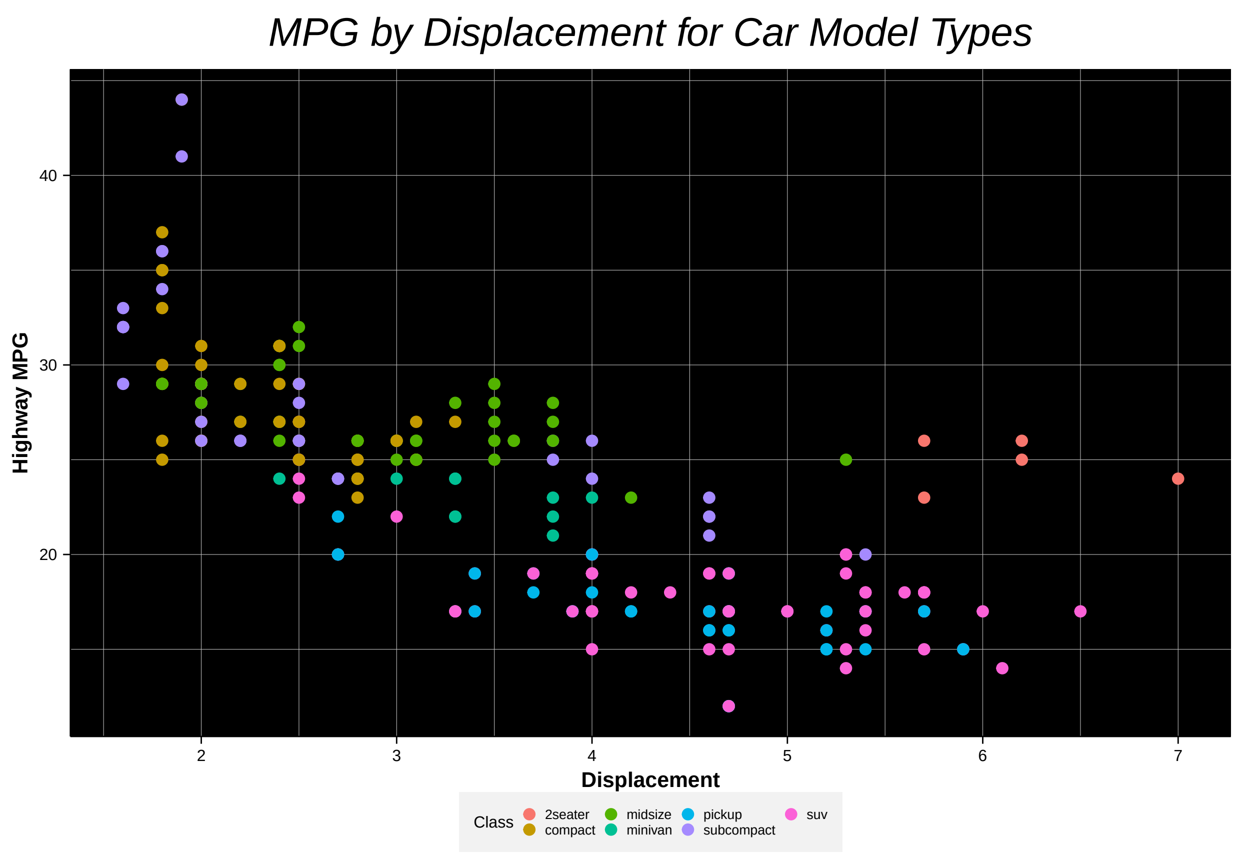
If we were to add our saved theme, for instance a dark theme. This can be added to the same plot with the following. As you can see the plot theming has changed. For any particular plot you can also always change parts of theme custom to any particular plot as well however this is not suggested as this theme is built for us at Axle on the RWE team. - NOTE TO OTHERS THAT I AM STILL DECIDING ON FONTS AND GRID LINE PRESENCE

```r
#Our Theme
themex <- readRDS("saved_theme.rds")
p + themex
```



In the following we change the axis labels those must be built custom for all plots. Note that colour here is the title for the legend, if the variable being displayed were numeric it would be labeled 'fill' in ggplot2.

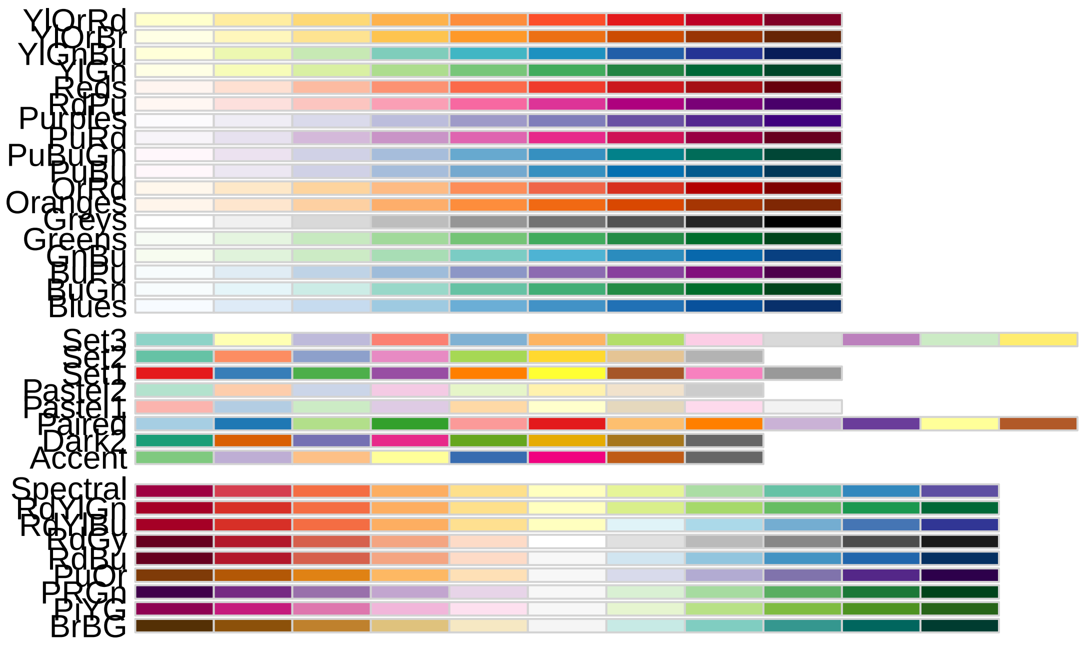
```r
p + themex + labs(x = "Displacement", y = "Highway MPG", title = "MPG by Displacement for Car Model Typ
```



Colors

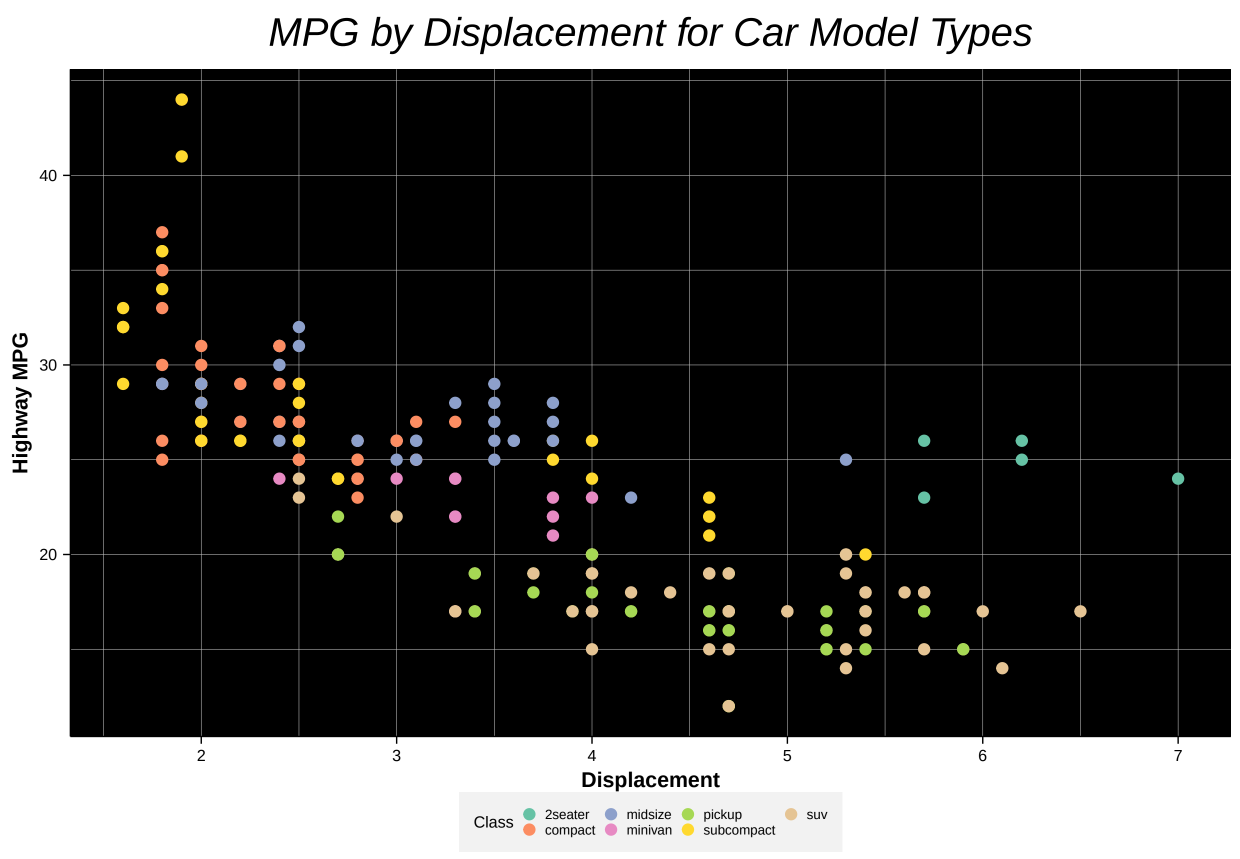
For figures that are going to used to display study results it is important to choose colors that are good for the display medium in which they will be used.

```
#install.packages("RColorBrewer")
library(RColorBrewer)
par(mar=c(3,4,2,2))
display.brewer.all()
```



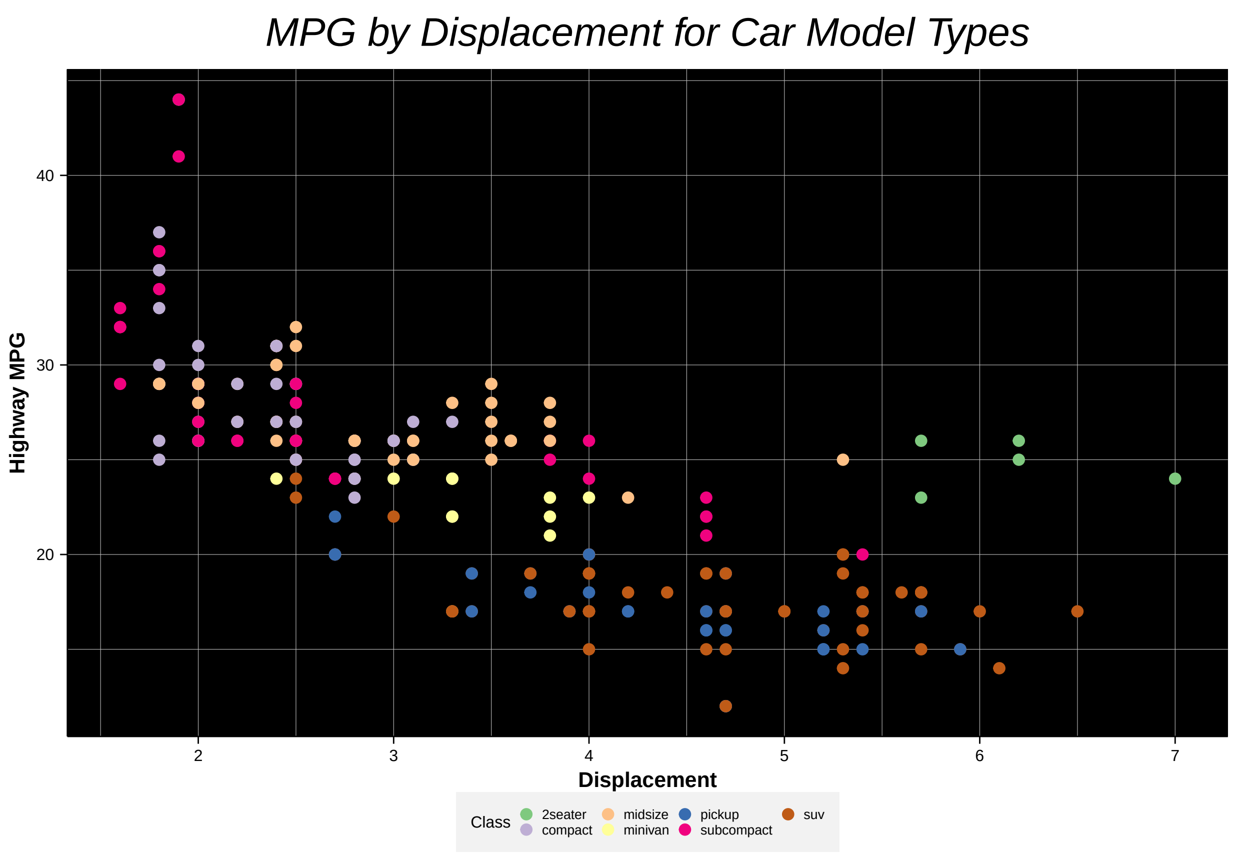
In addition to palette that are already built we can build our own palettes as well. It is important to consider how many colors you need for a particular plot. The MPG plot above has a total of 7 colors needed. So we need a palette that has over 7 colors.

```
t = p + themex + labs(x = "Displacement", y = "Highway MPG", title = "MPG by Displacement for Car Model
t + scale_color_brewer(palette = "Set2")
```



Also should be noted whether your theming is dark or light, as the colors can all be on the light or dark side of the spectrum depending on the theme background. Here we made the theme dark so all the colors should be below medium in shade but highly different in color hue. Thus we will change the colors accordingly.

```r
t + scale_color_brewer(palette = "Accent")
```



Finally, we note here that color is different from fill in ggplot. For color this is not a sequential numeric variable but a categorical variable thus colors like the first palettes with a single color growing darker, do not work here or convey the correct thought.

If your variable is numeric or has an order. Then we use fill instead of color and we used "scale_fill_brewer" instead.

Figure Saving

When saving a figure always save an SVG copy. But journals often want png or pdf just make sure if you need a high resolution image to set the dpi to 300.

```
# ggsave(p,
# filename = "filename.svg",
# device = c("svg", "png", "pdf"),
# width = cw2,
# height = cw2,
# #dpi = 300
# )
```