Ggplot comes with several inbuilt themes that can be easily applied to any plot. However, one can tweak these out-of-the-box styles using the theme() function. We did this last time. Furthermore, one also can create a complete customized theme, that's what we're going to do in this post.

**How the masters do it**

When we create a theme from scratch we have to define all arguments of the theme function and set the complete argument to TRUE, signaling that the generated object indeed is a complete theme. Setting complete = TRUE also causes all elements to inherit from blank elements. This means, that every object that we want to show up in our plots has to be defined in full detail.

If we are a little lazy, instead of defining each and every argument, we also can start with an existing theme and alter only some of its arguments. Actually, this is exactly what the creators of ggplot did. While theme\_gray() is "the mother of all themes" and fully defined, for example theme\_bw() builds upon theme\_gray() , while theme\_minimal in turn builds on theme\_bw() .

Further, it is obvious from the code of theme\_bw or theme\_minimal how much more convenient it is to create a new theme by building on an existing theme.

**Creating our very own theme**

What's good enough for Hadley and friends, is good enough for me. Therefore, I'm going to create my own theme based on my favourite theme, theme\_minimal().We can create a new theme as a function calling an existing theme, which is altered by %+replace% theme() with all alterations defined in theme().

Several arguments are passed along to the function constituting a new theme and the existing theme called within the function: Specified are the default sizes for text (base\_size), lines in general (base\_line\_size) as well as lines pertaining to rect-objects (base\_rect\_size), further defined is the font family. To ensure consitent look, all sizes aren't defined in absolute terms but relative to base sizes, using the rel() function. Therefore, for especially big or small plots the base sizes can be in- or decreased, with all other elements being adjusted automatically.

library(ggplot2)

library(gridExtra)

library(dplyr)

# generating new theme

theme\_new <- function(base\_size = 11,

base\_family = "",

base\_line\_size = base\_size / 170,

base\_rect\_size = base\_size / 170){

theme\_minimal(base\_size = base\_size,

base\_family = base\_family,

base\_line\_size = base\_line\_size) %+replace%

theme(

plot.title = element\_text(

color = rgb(25, 43, 65, maxColorValue = 255),

face = "bold",

hjust = 0),

axis.title = element\_text(

color = rgb(105, 105, 105, maxColorValue = 255),

size = rel(0.75)),

axis.text = element\_text(

color = rgb(105, 105, 105, maxColorValue = 255),

size = rel(0.5)),

panel.grid.major = element\_line(

rgb(105, 105, 105, maxColorValue = 255),

linetype = "dotted"),

panel.grid.minor = element\_line(

rgb(105, 105, 105, maxColorValue = 255),

linetype = "dotted",

size = rel(4)),

complete = TRUE

)

}

Other than in theme\_minimal() I'm decreasing the base size to 11 and set the base line size and base rect size to base size devided by 170. I don't change the font family. The plot title is changed to a dark blue, bold font in the set base size and is left alinged. Axis text and axis ticks are set to have 75% and 50% of the base size, while their colour is changed to a light grey. Finally, the lines of the grid are defined to be dotted and light grey, with the major grid lines having the base line size and the minor grid lines having four times this size.

The result looks like this:

# base plot

base\_plot <- data.frame(x = rnorm(n = 100, 1.5, 2),

y = rnorm(n = 100, 1, 2),

z = c(rnorm(n = 60, 0.5, 2), rnorm(n = 40, 5, 3))) %>%

ggplot(.) +

geom\_jitter(aes(x = x, y = y, color = z, size = z),

alpha = 0.5) +

geom\_jitter(aes(x = x, y = y, size = z),

alpha = 0.8,

shape = 21,

color = "white",

stroke = 0.4) +

scale\_size\_continuous(range = c(1, 18), breaks = c(1, 4, 5, 13, 18)) +

guides(size = FALSE, color = FALSE) +

labs(y = "Flight Hight", x = "Flight Distance")

# plot with customized theme

p1 <- base\_plot +

ggtitle("Bubbels - theme\_new()") +

theme\_new()

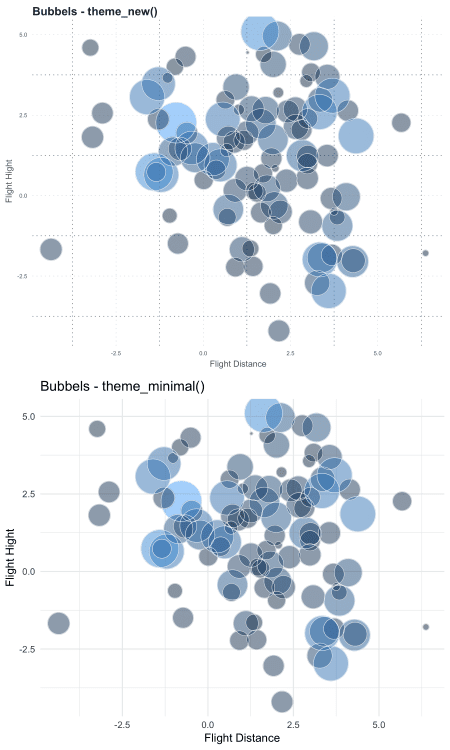
# plot with theme minimal

p2 <- base\_plot +

ggtitle("Bubbels - theme\_minimal()") +

theme\_minimal()

grid.arrange(p1, p2, nrow = 2)



For later use, we can save our theme gerenrating script and source our customized theme function whenever we want to make use of our created theme.

So go ahead, be creative and build your signature theme!