Getting Started

At the Digitas data team, we have developed an R package to make the process of creating a standard way for graphics in our in-house style using R's ggplot2 library a more reproducible process, as well as making it easier for people new to R to create graphics. This vignette contains the functions of the digitasthemeV2 package, which once installed locally, provides helpful functions for creating and exporting graphics made in ggplot in the style used by the Digitas data team.

Load all the libraries you need

```
library(sysfonts)
library(extrafont)
#> Registering fonts with R
library(ggplot2)
```

Install the digitasthemeV2 package

Package digitasthemeV2 is not on CRAN, so you will have to install it locally and then load it.

```
library(digitasthemeV2)
```

When you have downloaded the package and successfully installed it, you are good to go and create charts.

How does the digitasthemeV2 package work?

The package has five functions for plots:
1.**theme_digitas()**
2.**scale_fill_digitas()**
3.**scale_color_digitas()**
4.**add_font()**
5.**save_plot()**

A basic explanation and summary here:
1.**theme_digitas()**: Has no arguments and is added to the ggplot chain after you have created a plot. What it does is generally makes text size, font and colour, axis lines, axis text and many other standard chart components into Digitas style.
2.**add_font()**:Add gotham rounded font to the theme.
3.**scale_fill_digitas()** and **scale_color_digitas()** :These functions work on the aesthetics specified in the scale.
4.**save_plot()**:Saves the graphics on local.

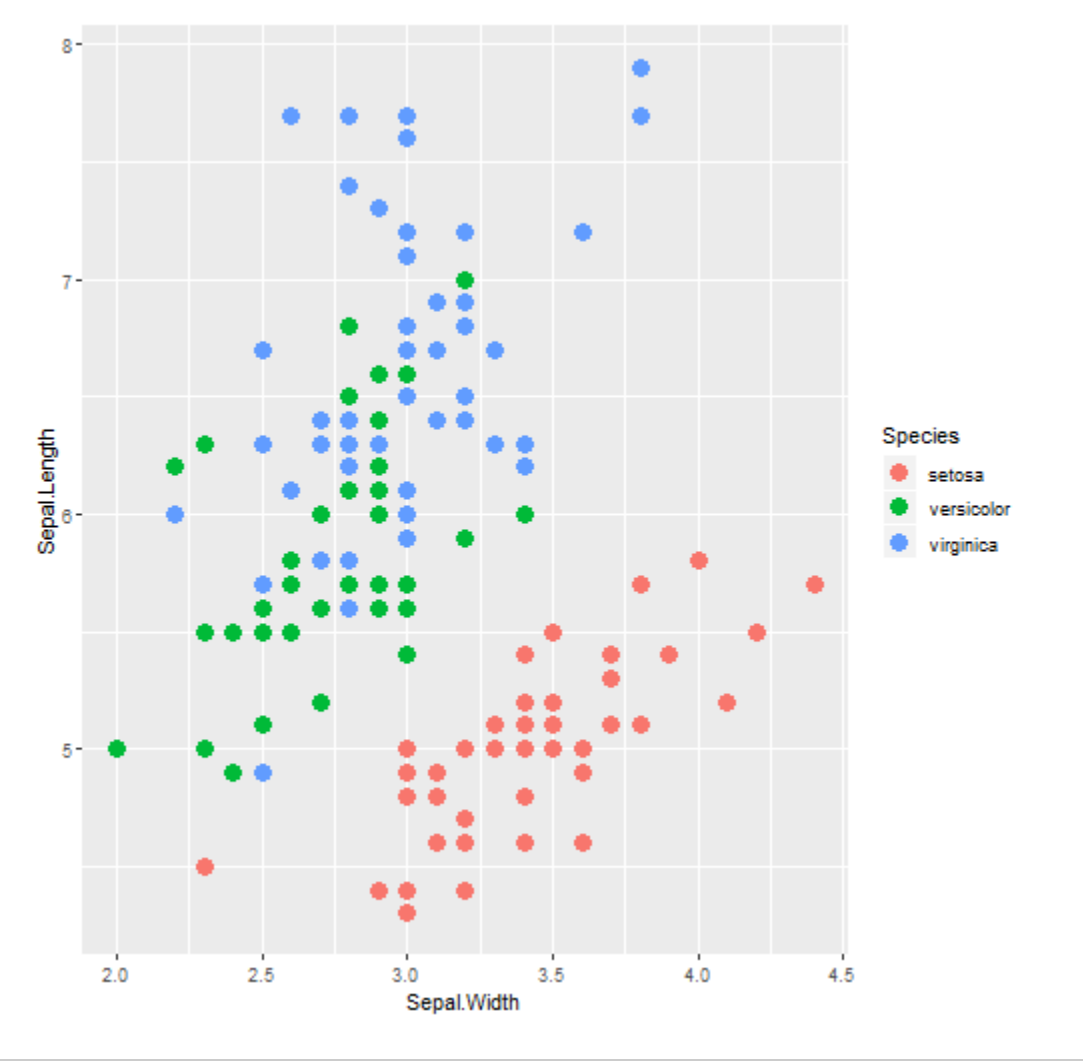
Note: that colours for lines in the case of a line chart or bars for a bar chart, do not come out of the box from the theme_digitas() function, but need to be explicitly set in your other standard ggplot chart functions.

1.Add font

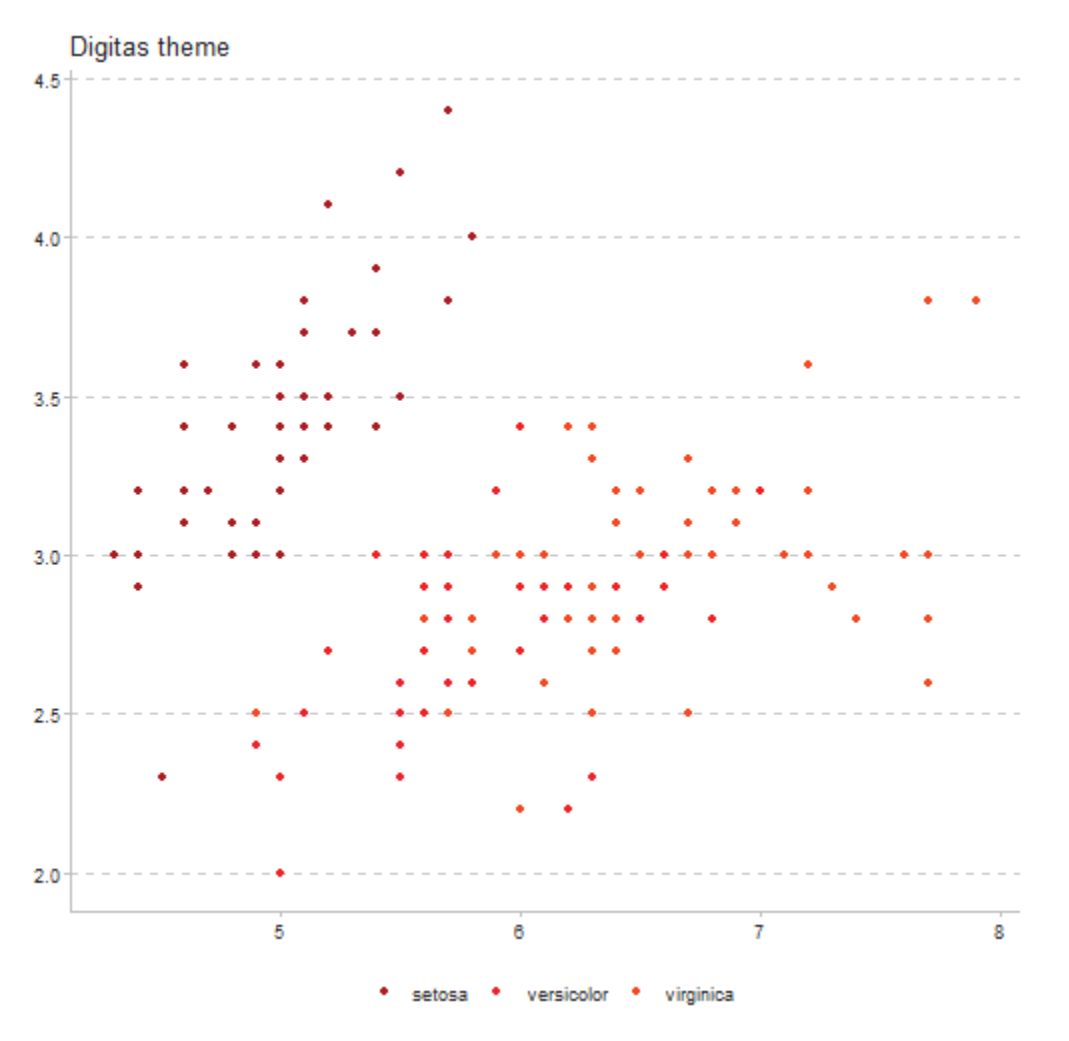
run add_font() function.

2.Plot a ggplot graph and add digitas theme and colors from color palette

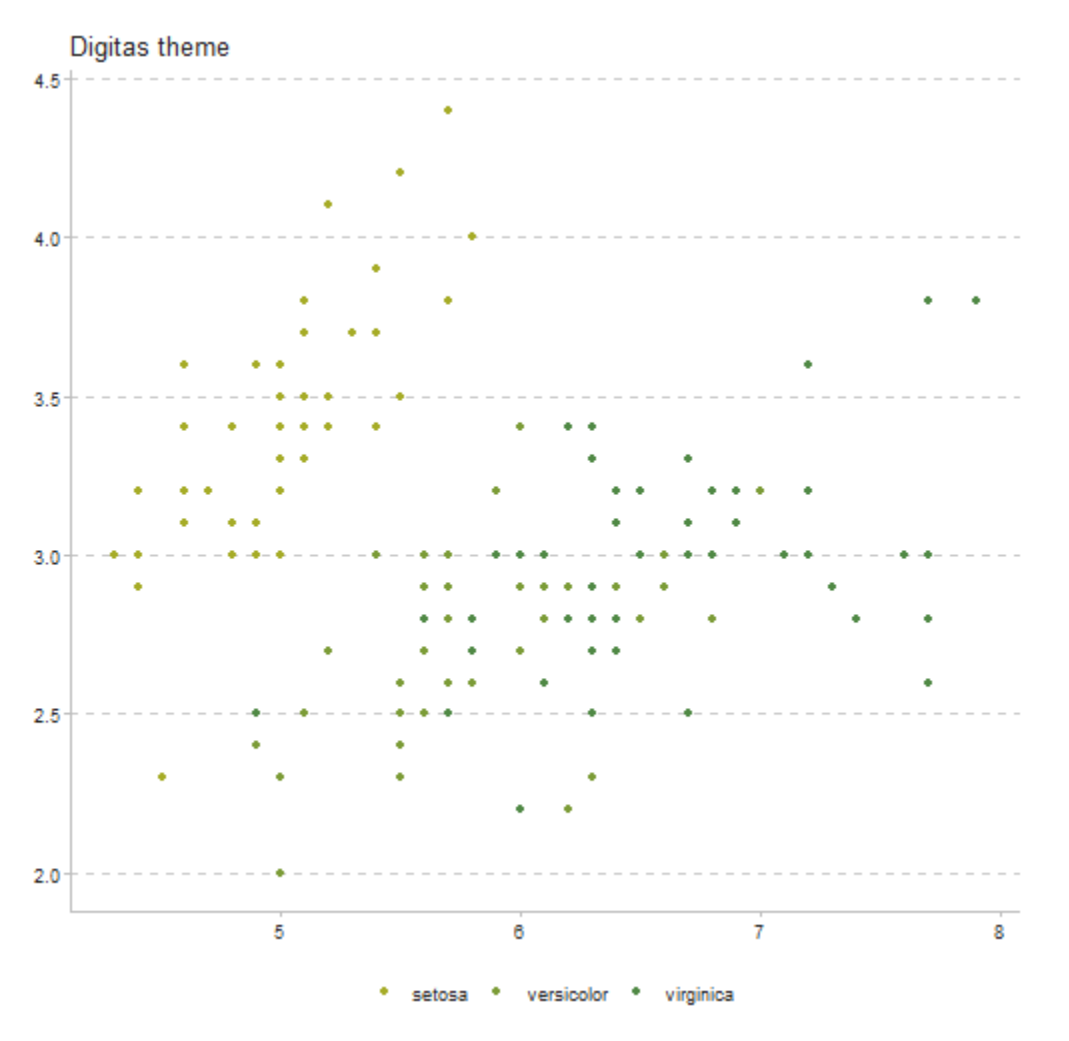
```
options(warn=-1)
library(digitasthemeV2)
#Without theme
ggplot(iris, aes(Sepal.Width, Sepal.Length, color = Species)) +  geom_point(size = 4)
```




```
#With theme
data("iris")
d1 <- qplot(x  = Sepal.Length, y =Sepal.Width,colour = Species,data = iris,geom = "point")
d1 + theme_digitas() +
  scale_color_digitas(theme="red") +
  labs(title="Digitas theme")
```




```
d1 + theme_digitas() +
  scale_color_digitas(theme="green") +
  labs(title="Digitas theme")
```




```
d1 + theme_digitas() +
  scale_color_digitas(theme="blue") +
  labs(title="Digitas theme")
```

Here is what the theme_digitas() function actually does under the hood. It essentially modifies certain arguments in the theme function of ggplot2.

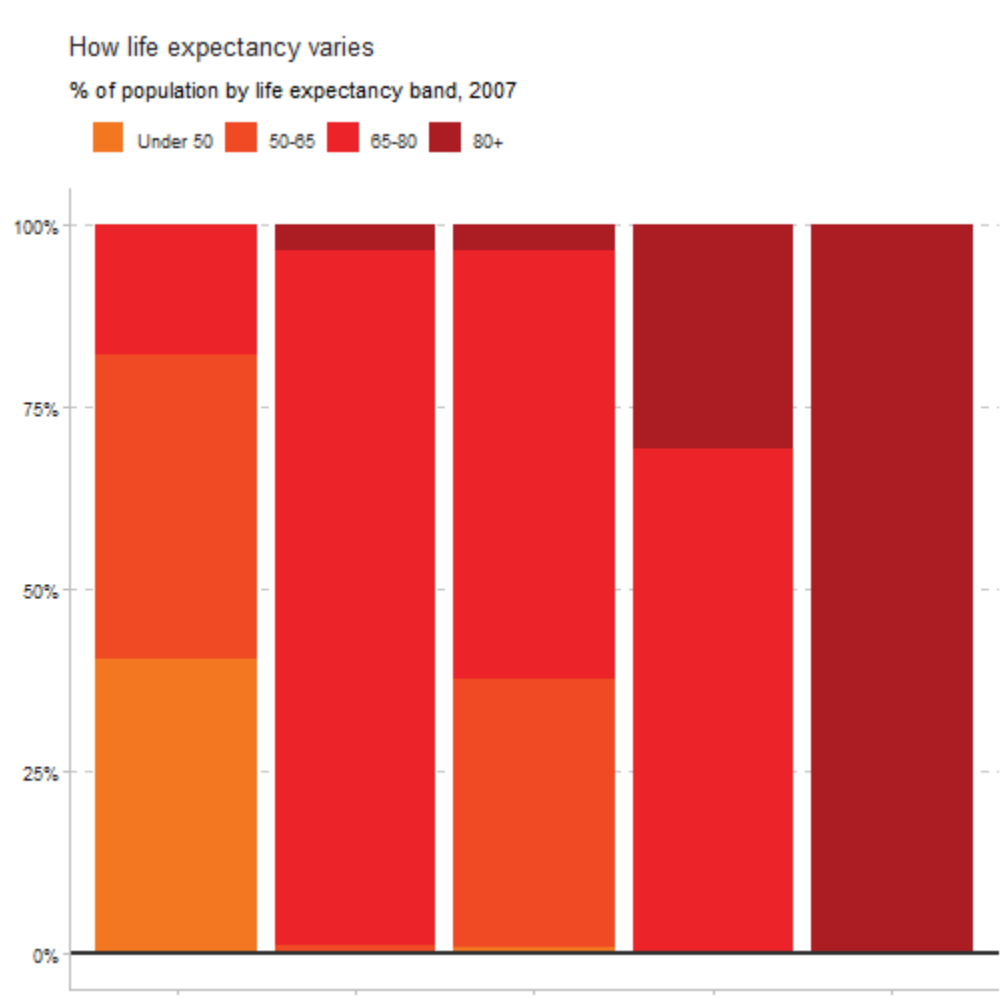
You can modify these settings for your chart, or add additional theme arguments, by calling the theme function with the arguments you want - but please note that for it to work you must call it after you have called the theme_digitas() function. Otherwise theme_digitas() will override it.

Both scale_fill_digitas() and scale_color_digitas() functions have few color palettes:
scale_color_digitas():This function has four themes - red,green,blue and black.(Work as a scale_color_manual function)
scale_fill_digitas():This function has eight themes - red,green,blue, black,red_green,red_blue,red_black and red_green_blue_black(Work as a scale_fill_manual function)

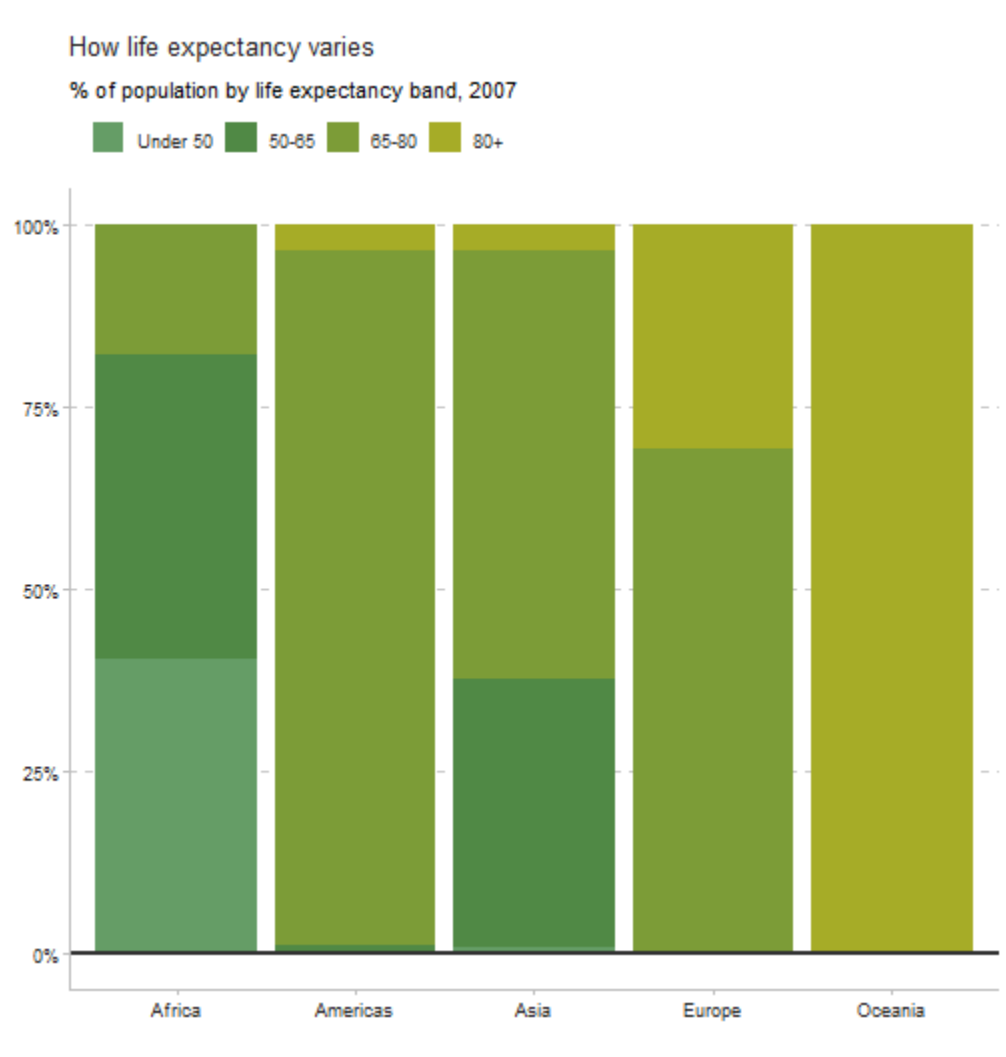
Note:If you have any confusion when to use either of them ,please explore scale_color_manual() and scale_fill_manual()from ggplot2 package.
Few more examples are:

```
library("gapminder")
library("tidyr")
library("dplyr")
#>
#> 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

stacked_df <- gapminder %>%
  filter(year == 2007) %>%
  mutate(lifeExpGrouped = cut(lifeExp,
                              breaks = c(0, 50, 65, 80, 90),
                              labels = c("Under 50", "50-65", "65-80", "80+"))) %>%
  group_by(continent, lifeExpGrouped) %>%
  summarise(continentPop = sum(as.numeric(pop)))

#set order of stacks by changing factor levels
stacked_df$lifeExpGrouped = factor(stacked_df$lifeExpGrouped, levels = rev(levels(stacked_df$lifeExpGrouped)))
```

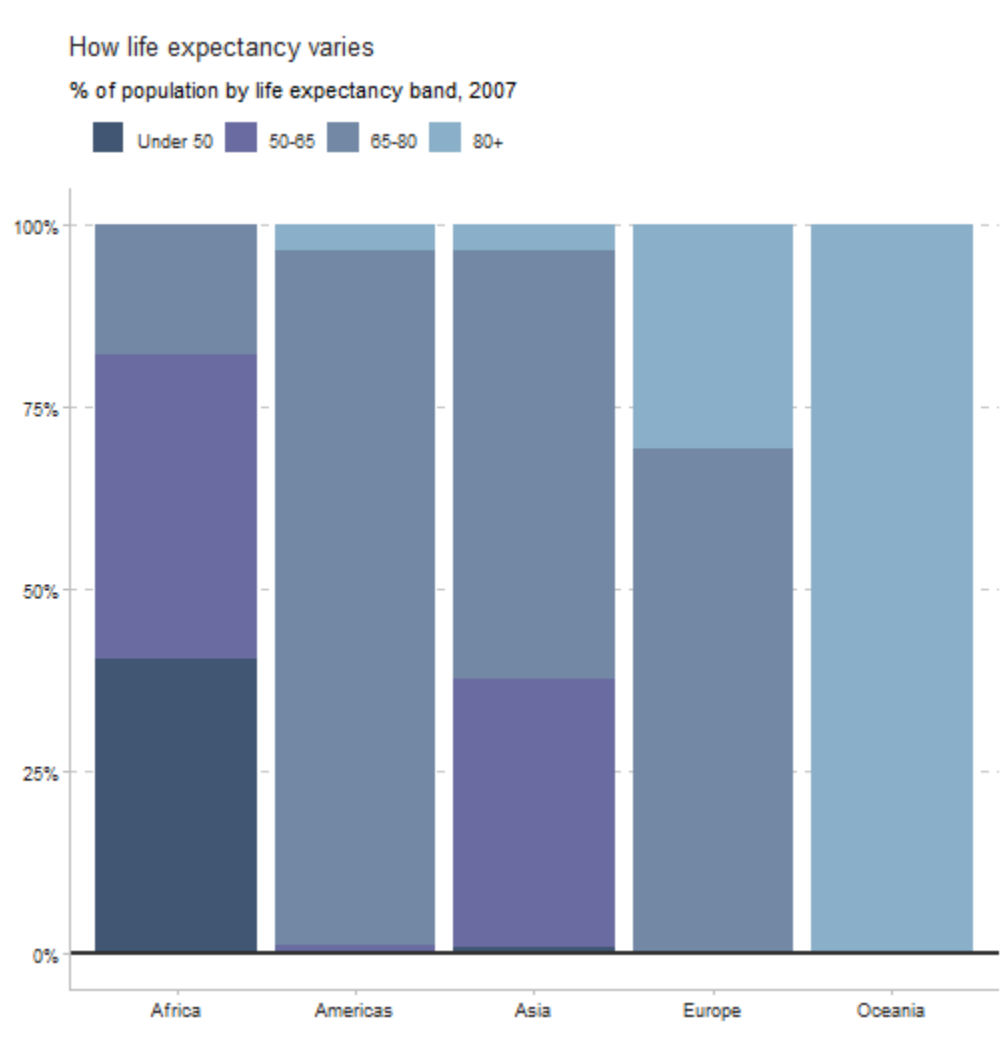
```
#create plot
ggplot(data = stacked_df,
                       aes(x = continent,
                           y = continentPop,
                           fill = lifeExpGrouped)) +
  geom_bar(stat = "identity",
           position = "fill") +
  theme_digitas() +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_digitas(theme="red") +
  geom_hline(yintercept = 0, size = 1, colour = "#333333") +
  labs(title = "How life expectancy varies",
       subtitle = "% of population by life expectancy band, 2007") +
  theme(legend.position = "top",
        legend.justification = "left") +
  guides(fill = guide_legend(reverse = TRUE))
```



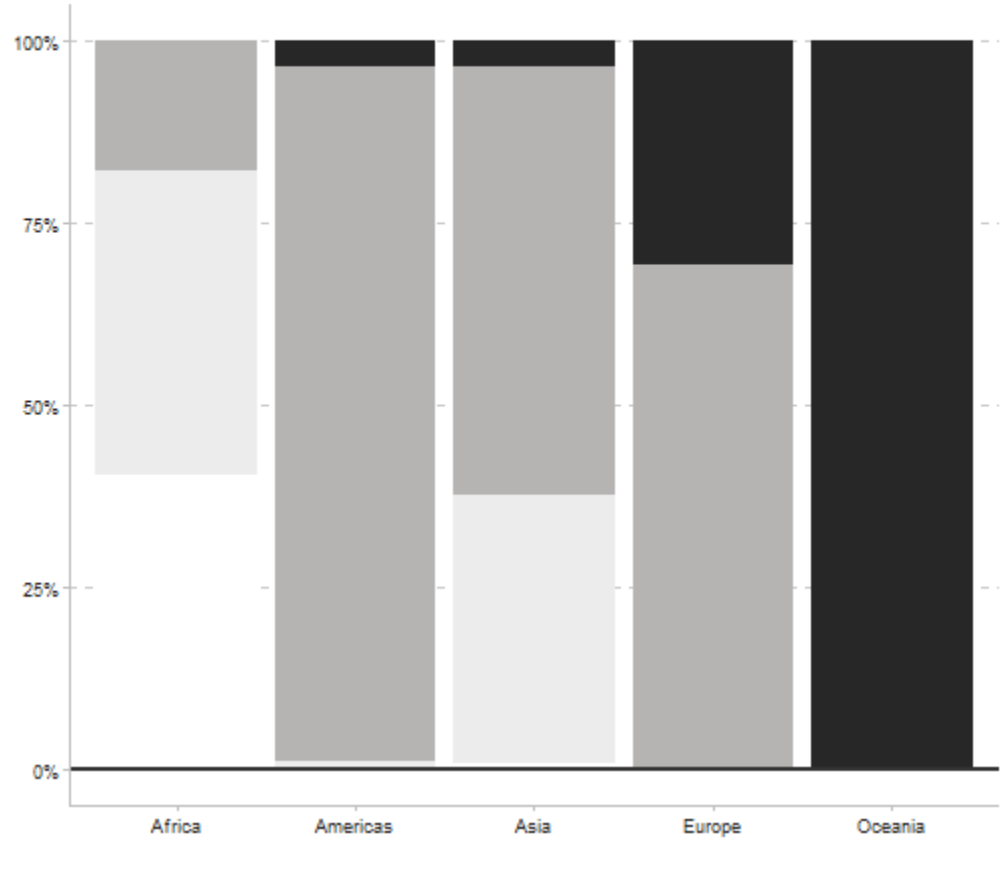

```
ggplot(data = stacked_df,
                       aes(x = continent,
                           y = continentPop,
                           fill = lifeExpGrouped)) +
  geom_bar(stat = "identity",
           position = "fill") +
  theme_digitas() +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_digitas(theme="green") +
  geom_hline(yintercept = 0, size = 1, colour = "#333333") +
  labs(title = "How life expectancy varies",
       subtitle = "% of population by life expectancy band, 2007") +
  theme(legend.position = "top",
        legend.justification = "left") +
  guides(fill = guide_legend(reverse = TRUE))
```




```
stacked_bars<-ggplot(data = stacked_df,
                       aes(x = continent,
                           y = continentPop,
                           fill = lifeExpGrouped)) +
  geom_bar(stat = "identity",
           position = "fill") +
  theme_digitas() +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_digitas(theme="blue") +
  geom_hline(yintercept = 0, size = 1, colour = "#333333") +
  labs(title = "How life expectancy varies",
       subtitle = "% of population by life expectancy band, 2007") +
  theme(legend.position = "top",
        legend.justification = "left") +
  guides(fill = guide_legend(reverse = TRUE))
stacked_bars
```




```
ggplot(data = stacked_df,
                       aes(x = continent,
                           y = continentPop,
                           fill = lifeExpGrouped)) +
  geom_bar(stat = "identity",
           position = "fill") +
  theme_digitas() +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_digitas(theme="black") +
  geom_hline(yintercept = 0, size = 1, colour = "#333333") +
  labs(title = "How life expectancy varies",
       subtitle = "% of population by life expectancy band, 2007") +
  theme(legend.position = "top",
        legend.justification = "left") +
  guides(fill = guide_legend(reverse = TRUE))
```



3.Save out your finished chart

After adding the theme_digitas() to your chart there is one more step to get your plot ready for sharing. finalise_plot() will left-align the title, subtitle and add the footer with a source and an image in the bottom right corner of your plot. It will also save it to your specified location. The function has five arguments:

Here are the function arguments:

finalise_plot(plot_name, source, save_filepath, width_pixels = 640, height_pixels = 450)

plot_name: The variable name that you have called your plot, for example for the chart example above plot_name would be "stacked_bars"
source:The source text that you want to appear at the bottom left corner of your plot. You will need to type the word "Source:" before it, so for example source = "Source: Digitas" would be the right way to do that.
save_filepath:The precise filepath that you want your graphic to save to, including the .png extension at the end. This does depend on your working directory and if you are in a specific R project.
width_pixels:This is set to 640px by default, so only call this argument if you want the chart to have a different width, and specify what you want it to be.
height_pixels:This is set to 450px by default, so only call this argument if you want the chart to have a different height, and specify what you want it to be.
logo_image_path:This argument specifies the path for the image/logo in the bottom right corner of the plot. The default is for a placeholder PNG file with a background that matches the background colour of the plot, so do not specify the argument if you want it to appear without a logo. If you want to add your own logo, just specify the path to your PNG file. The package has been prepared with a wide and thin image in mind.

Example of how the finalise_plot() is used in a standard workflow. This function is called once you have created and finalised your chart data, titles and added the theme_digitas() to it:

```
finalise_plot(plot_name = stacked_bars,
              source = "Source: Digitas",
              save_filepath = "plot.png",
              width_pixels = 640,
              height_pixels = 550

              )
```