Using grattantheme

This vignette explains how to use `grattantheme` to quickly and consistently apply Grattan chart formatting to charts made in `R` using `ggplot`.

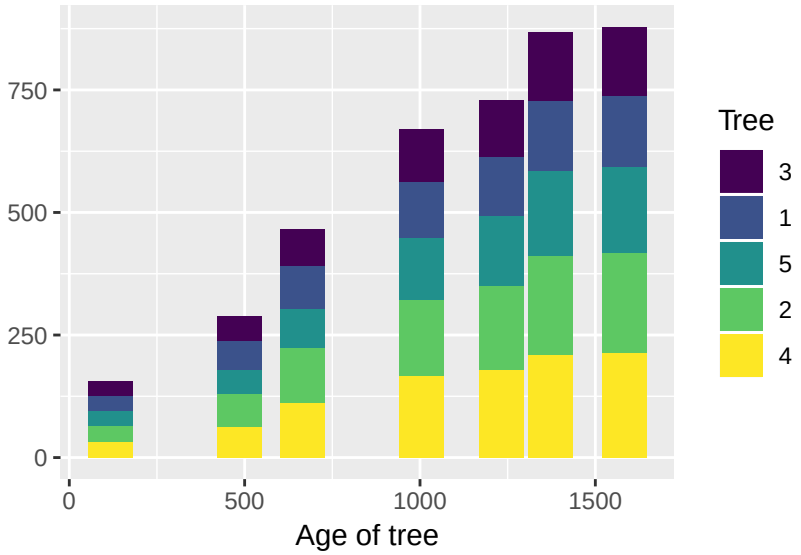
When creating a chart using `ggplot` we have to:

- Choose a dataset;
- Map variables to chart aesthetics `aes()`;
- Choose a `geom_`.

For example, using the `Orange` dataset tracking the growth of five orange trees by age:

```
plot <- ggplot(Orange,
               aes(x = age,
                   y = circumference,
                   fill = Tree)) +
        geom_bar(stat = "identity") +
        labs(x = "Age of tree",
             y = "",
             colour = "Tree")
```

This successfully plots the data we want to plot:

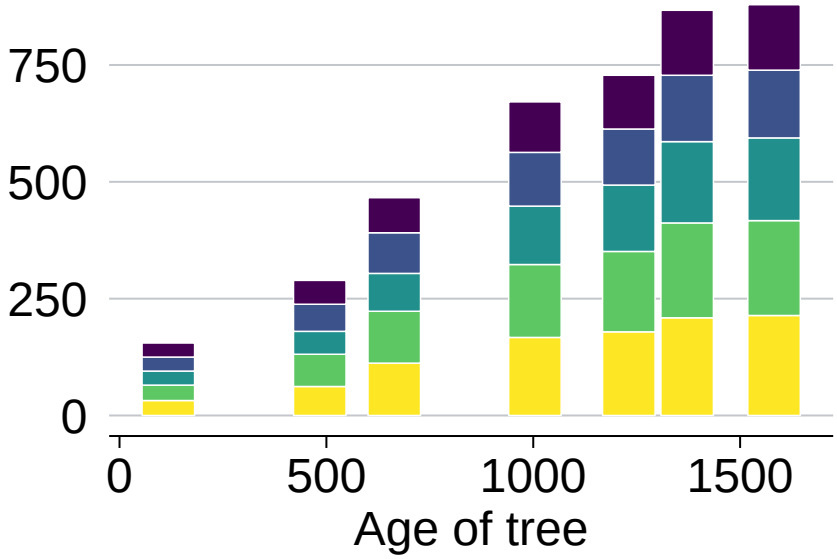


But it doesn't yet *look* like a Grattan chart. To adjust the *look* we adjust '`theme`' elements, like `axis.ticks.x = element_line(colour = "black")` to adjust the axis tickmarks on the x axis; `panel.grid.major.x = element_blank()` to turn off vertical gridlines; and so on; and on; and on. We also need to adjust aesthetic colours to the Grattan palette; setting, for example, `fill = "#F68B33"`. The `grattantheme` package contains tools and shortcuts to simplify this process.

Formatting theme elements with `theme_grattan()`

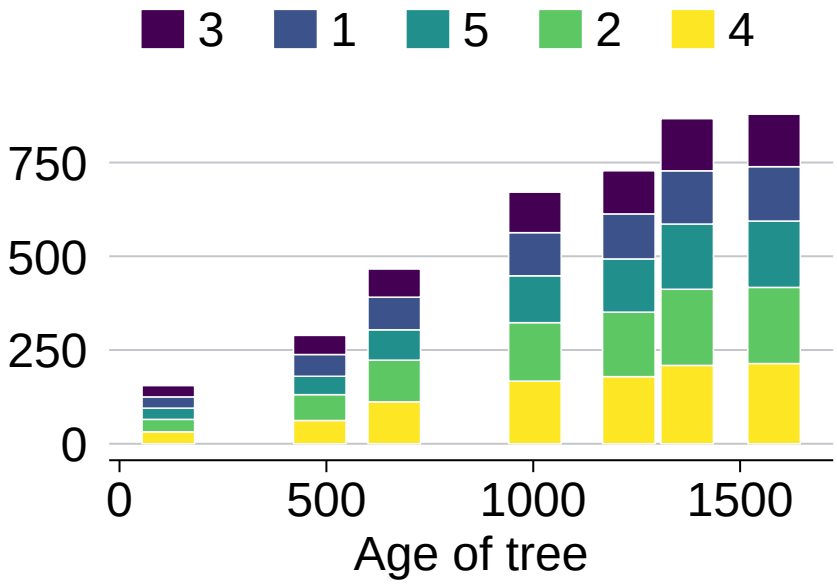
The function `theme_grattan()` contains all of the Grattan `theme` adjustments in one handy command. Combined with `grattan_colour_manual`, which easily changes colours of aesthetics, your `R` chart will be ready for a report or a slide in no time.

```
plot +
  theme_grattan()
```



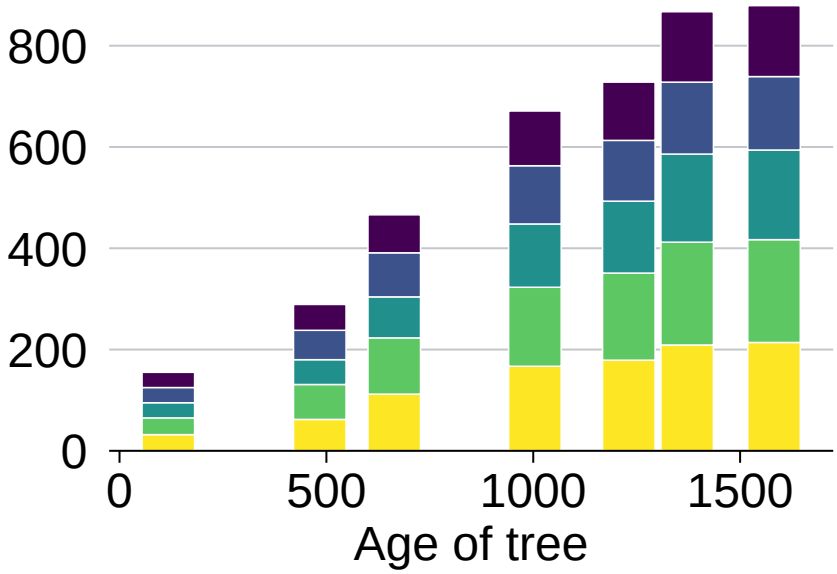
By default, `theme_grattan()` supresses the legend to allow for clearer on-chart labelling. We can include the legend with the `legend` argument, which takes `"off"`, `"top"`, `"bottom"`, `"left"` or `"right"`:

```
plot +
  theme_grattan(legend = "top")
```



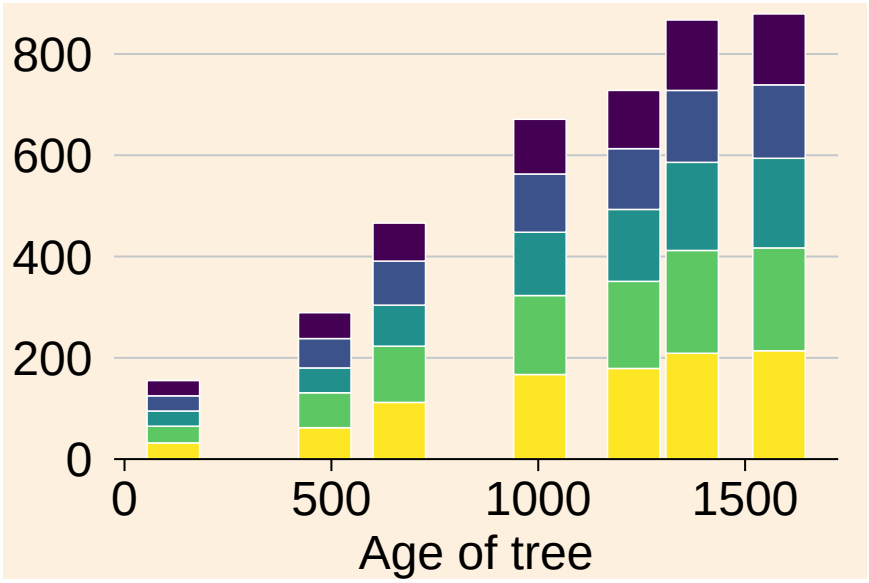
To align the y-axis with zero, change the y scale with `grattan_y_continuous()`:

```
plot +
  theme_grattan() +
  grattan_y_continuous()
```



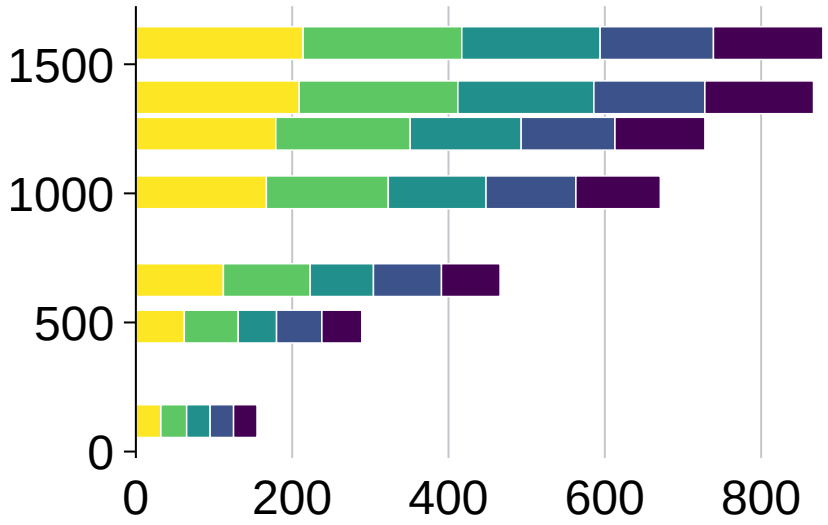
Sometimes we'll want a chart for a box in a report. We can change the background colour with the `background` argument:

```
plot +
  theme_grattan(background = "orange") +
  grattan_y_continuous()
```



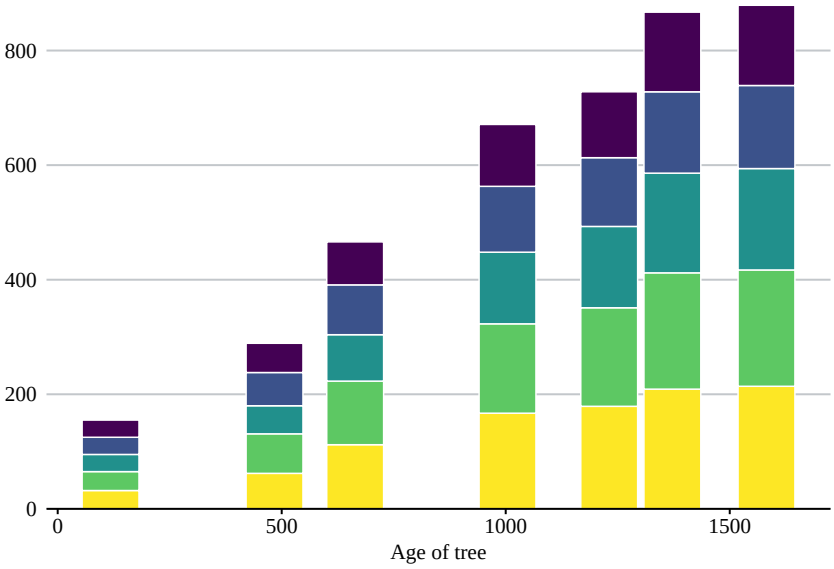
The standard Grattan rules for `x` and `y` axes flip if the chart is a horizontal bar chart. The `x` axis then follows the rules of the `y` axis, and vice-versa. If we are using a 'flipped' chart (imlemented with `coord_flipped()`), we can tell `theme_grattan` this is the case using the argument `fillped` set to `TRUE`.

```
plot +
  coord_flip() +
  theme_grattan(flipped = TRUE) +
  grattan_y_continuous()
```



The final adjustments we can specify with `theme_grattan` are the font size and font family. The defaults meet Grattan formatting requirements, but if we do need to change them we can:

```
plot +
  theme_grattan(base_size = 8, base_family = "serif") +
  grattan_y_continuous()
```

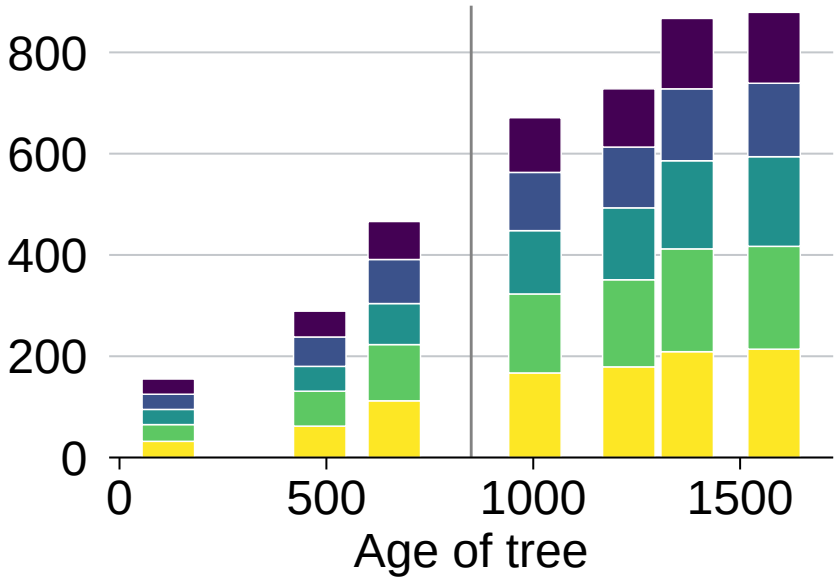


Using Grattan colours

Grattan's colours are loaded with `grattantheme`. The HEX codes for individual Grattan colours can be called using `grattan_[colourname]`, eg `grattan_lightorange`. Colours names are taken from the chart-guide and are:

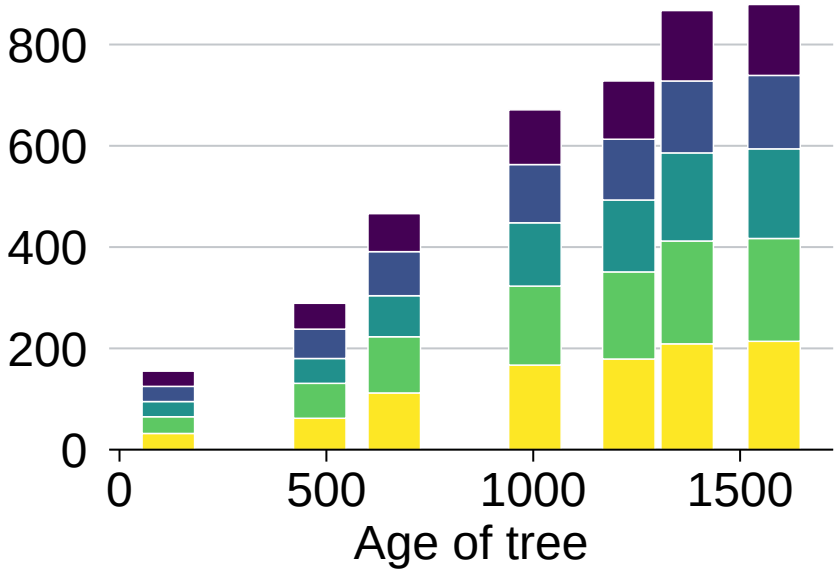
We can call a single colour:

```
plot +
  geom_vline(xintercept = 850, colour = grattan_grey3) +
  theme_grattan() +
  grattan_y_continuous()
```



We can also use the `grattan_fill_manual()` or `grattan_colour_manual()` functions to change the colours of our fill or colour *aesthetics*. In our example, we have five different trees each represented by a colour, so we set the number of colours to five: `grattan_fill_manual(n = 5)`:

```
plot +
  theme_grattan() +
  grattan_y_continuous()
```

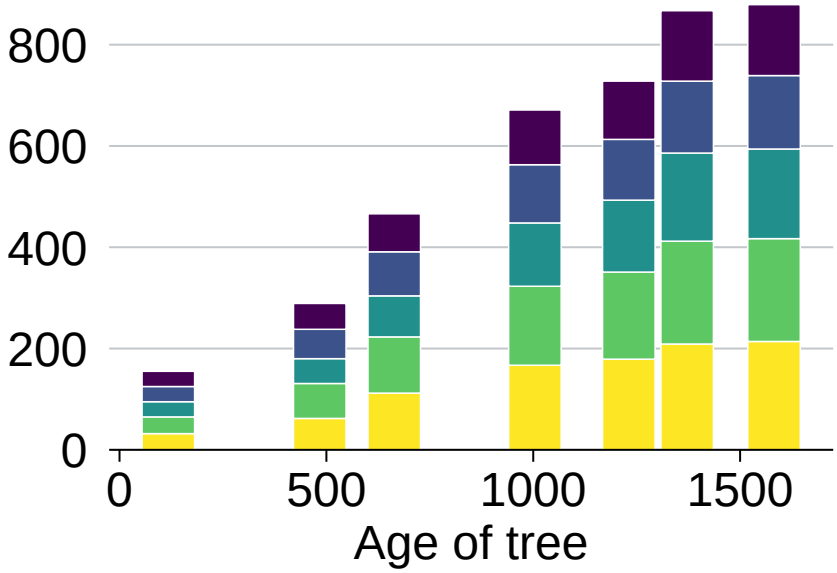



```
  grattan_fill_manual(n = 5)
#> <ggproto object: Class ScaleDiscrete, Scale, gg>
#>     aesthetics: fill
#>     axis_order: function
#>     break_info: function
#>     break_positions: function
#>     breaks: waiver
#>     call: call
#>     clone: function
#>     dimension: function
#>     drop: TRUE
#>     expand: waiver
#>     get_breaks: function
#>     get_breaks_minor: function
#>     get_labels: function
#>     get_limits: function
#>     guide: legend
#>     is_discrete: function
#>     is_empty: function
#>     labels: waiver
#>     limits: NULL
#>     make_sec_title: function
#>     make_title: function
#>     map: function
#>     map_df: function
#>     n.breaks.cache: NULL
#>     na.translate: TRUE
#>     na.value: NA
#>     name: waiver
#>     palette: function
#>     palette.cache: NULL
#>     position: left
#>     range: <ggproto object: Class RangeDiscrete, Range, gg>
#>         range: NULL
#>         reset: function
```

```
#>         train: function
#>         super:  <ggproto object: Class RangeDiscrete, Range, gg>
#>     reset: function
#>     scale_name: manual
#>     train: function
#>     train_df: function
#>     transform: function
#>     transform_df: function
#>     super:  <ggproto object: Class ScaleDiscrete, Scale, gg>
```

We can reverse the order of the fill colours using the `reverse` argument:

```
plot +
  theme_grattan() +
  grattan_y_continuous()
```

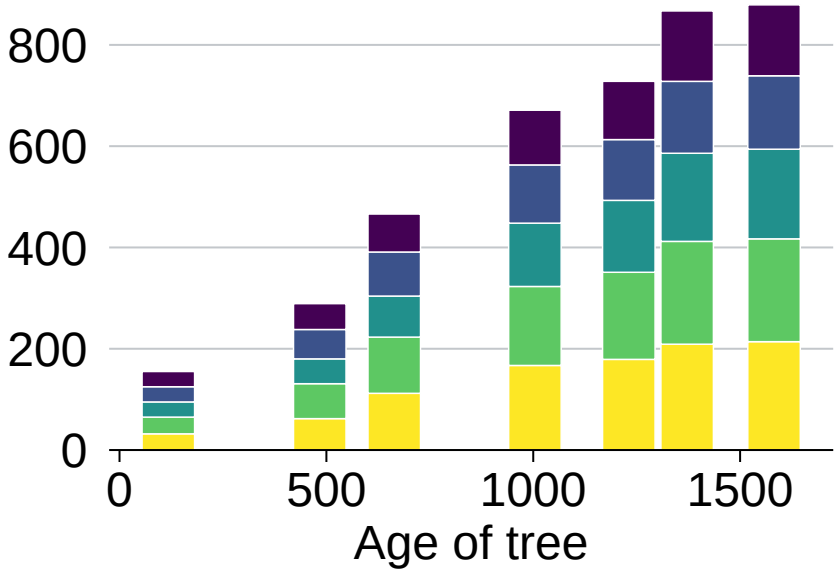



```
  grattan_fill_manual(n = 5, reverse = TRUE)
#> <ggproto object: Class ScaleDiscrete, Scale, gg>
#>     aesthetics: fill
#>     axis_order: function
#>     break_info: function
#>     break_positions: function
#>     breaks: waiver
#>     call: call
#>     clone: function
#>     dimension: function
#>     drop: TRUE
#>     expand: waiver
#>     get_breaks: function
#>     get_breaks_minor: function
#>     get_labels: function
#>     get_limits: function
#>     guide: legend
#>     is_discrete: function
#>     is_empty: function
#>     labels: waiver
```

```
#>     limits: NULL
#>     make_sec_title: function
#>     make_title: function
#>     map: function
#>     map_df: function
#>     n.breaks.cache: NULL
#>     na.translate: TRUE
#>     na.value: NA
#>     name: waiver
#>     palette: function
#>     palette.cache: NULL
#>     position: left
#>     range: <ggproto object: Class RangeDiscrete, Range, gg>
#>         range: NULL
#>         reset: function
#>         train: function
#>         super:  <ggproto object: Class RangeDiscrete, Range, gg>
#>     reset: function
#>     scale_name: manual
#>     train: function
#>     train_df: function
#>     transform: function
#>     transform_df: function
#>     super:  <ggproto object: Class ScaleDiscrete, Scale, gg>
```

Note that if you do not specify *enough* colours, will receive an error:

```
plot +
  theme_grattan() +
  grattan_y_continuous()
```




```
  grattan_fill_manual(n = 3)
#> <ggproto object: Class ScaleDiscrete, Scale, gg>
#>     aesthetics: fill
#>     axis_order: function
#>     break_info: function
```

```
#>     break_positions: function
#>     breaks: waiver
#>     call: call
#>     clone: function
#>     dimension: function
#>     drop: TRUE
#>     expand: waiver
#>     get_breaks: function
#>     get_breaks_minor: function
#>     get_labels: function
#>     get_limits: function
#>     guide: legend
#>     is_discrete: function
#>     is_empty: function
#>     labels: waiver
#>     limits: NULL
#>     make_sec_title: function
#>     make_title: function
#>     map: function
#>     map_df: function
#>     n.breaks.cache: NULL
#>     na.translate: TRUE
#>     na.value: NA
#>     name: waiver
#>     palette: function
#>     palette.cache: NULL
#>     position: left
#>     range: <ggproto object: Class RangeDiscrete, Range, gg>
#>         range: NULL
#>         reset: function
#>         train: function
#>         super:  <ggproto object: Class RangeDiscrete, Range, gg>
#>     reset: function
#>     scale_name: manual
#>     train: function
#>     train_df: function
#>     transform: function
#>     transform_df: function
#>     super:  <ggproto object: Class ScaleDiscrete, Scale, gg>
```