library(simplevis)

library(dplyr)

library(palmerpenguins)

**Overview**

simplevis provides gglot2 (and leaflet) wrapper functions with an objective to help users make beautiful visualisation with less brainpower and typing.

we discussed how simplevis provides families of functions for when the user is colouring or facetting by a variable or both or neither.

If you haven’t read that post, please read that one before this one.

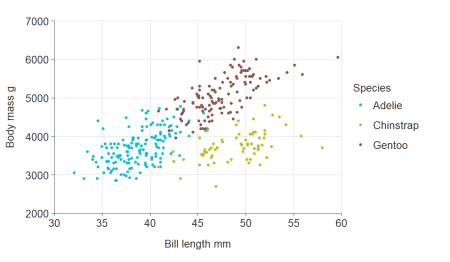
In the current post, we discus how to adjust titles and scales within simplevis.

**Titles**

Default titles in simplevis try to provide a polished quick graph with minimal code by:

* having no title, subtitle or caption
* converting x, y and colour titles to sentence case using the snakecase::to\_sentence\_case function.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species)



You can add or adjust titles using the title, subtitle, x\_title, y\_title, col\_title and caption arguments.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species,

title = "Adult penguin mass by bill length and species",

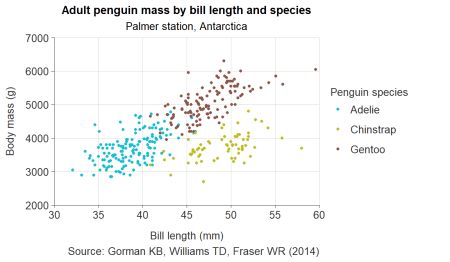
subtitle = "Palmer station, Antarctica",

x\_title = "Bill length (mm)",

y\_title = "Body mass (g)",

col\_title = "Penguin species",

caption = "Source: Gorman KB, Williams TD, Fraser WR (2014)")



If you want no x, y or colour title, you need to use x\_title = "", y\_title = "" or col\_title = "" as applicable.

**Scales: consistent prefixes and the autocomplete**

simplevis uses consistent prefixes in arguments to help users narrow down what they are looking for and then enable the Rstudio auto-complete to provide options.

In general, arguments that relate to:

* the x scale start with x\_
* the y scale start with y\_
* the colour scale start with col\_
* facetting start with facet\_

Therefore, if you know want to adjust the x scale but can’t think how, you can start typing x\_ within the simplevis function, press tab, and then you will be presented with a lot of options. You can use the arrow keys to scroll through these, and the tab to select.

**Numeric scales**

simplevis graphs numeric scales default to:

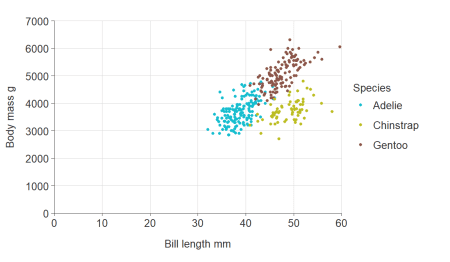
* starting from zero for numeric scales on bar graphs.
* *not* starting from zero for numeric scales on all other graphs.

You can use the x\_zero and y\_zero arguments to change the defaults.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species,

x\_zero = TRUE,

y\_zero = TRUE)

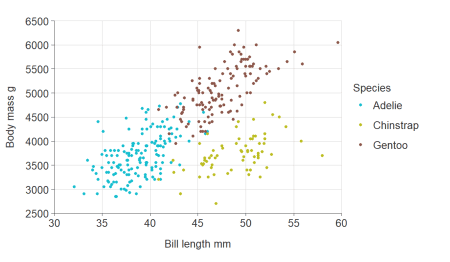


Adjust the number of breaks for numeric x and/or y scales.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species,

y\_pretty\_n = 10,

x\_pretty\_n = 6)



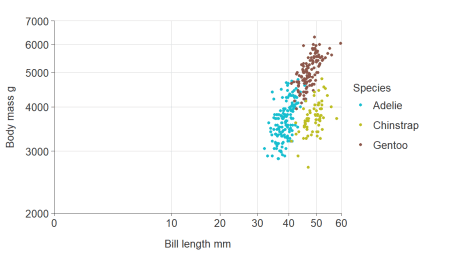
Transform numeric x and y scales.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species,

x\_trans = "sqrt",

x\_zero = T,

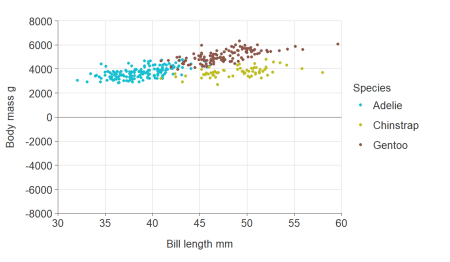
y\_trans = "log10")



Balance a numeric scale so that it has equivalence between positive and negative values.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species,

y\_balance = T)

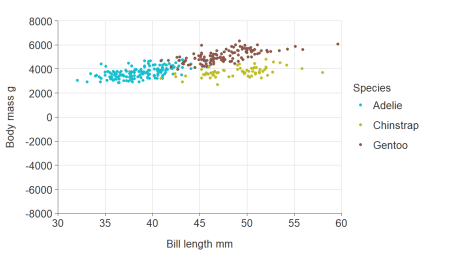


Zero lines default on if a numeric scale includes positive and negative values, but can be turned off if desired.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species,

y\_balance = T,

y\_zero\_line = F)



**Discrete scales**

simplevis automatically orders hbar graphs of character variables alphabetically.

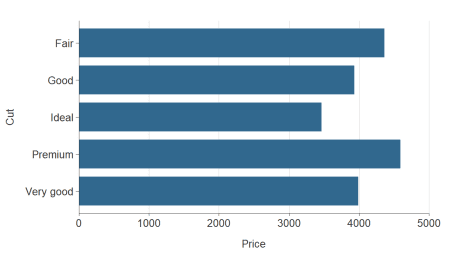
plot\_data <- ggplot2::diamonds %>%

mutate(cut = as.character(cut)) %>%

group\_by(cut) %>%

summarise(price = mean(price))

gg\_hbar(plot\_data, price, cut)



If there is an inherent order to the character variable that you want it to plot in, then you should convert the variable to a factor, and give it the appropriate levels.

cut\_levels <- c("Ideal", "Premium", "Very Good", "Good", "Fair")

plot\_data <- ggplot2::diamonds %>%

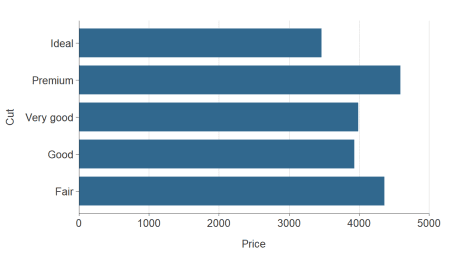
mutate(cut = as.character(cut)) %>%

mutate(cut = factor(cut, levels = cut\_levels)) %>%

group\_by(cut) %>%

summarise(price = mean(price))

gg\_hbar(plot\_data, price, cut)



Discrete scales can be reversed easily using the relevant y\_rev or x\_rev argument.

plot\_data <- ggplot2::diamonds %>%

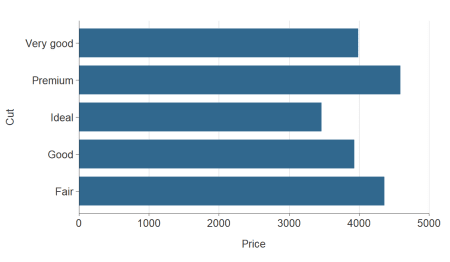
mutate(cut = as.character(cut)) %>%

group\_by(cut) %>%

summarise(price = mean(price))

gg\_hbar(plot\_data, price, cut,

y\_rev = TRUE)



Simple hbar and vbar plots made with gg\_bar() or gg\_hbar can be ordered by size using y\_reorder or x\_reorder. For other functions, you will need to reorder variables in the data as you wish them to be ordered.

plot\_data <- ggplot2::diamonds %>%

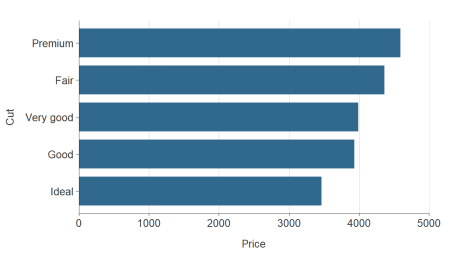
mutate(cut = as.character(cut)) %>%

group\_by(cut) %>%

summarise(price = mean(price))

gg\_hbar(plot\_data, price, cut,

y\_reorder = T)



**Colour scales**

Customise the colour title. Note that because colour labels will be converted to sentence case by default in simplevis, but we can turn this off when we do not want this to occur using ggplot2::waiver()

plot\_data <- ggplot2::diamonds %>%

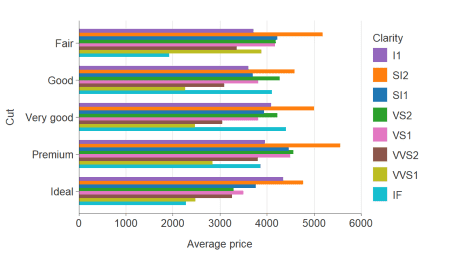
group\_by(cut, clarity) %>%

summarise(average\_price = mean(price))

gg\_hbar\_col(plot\_data, average\_price, cut, clarity,

col\_labels = ggplot2::waiver(),

pal\_rev = TRUE)



Reverse the palette.

plot\_data <- ggplot2::diamonds %>%

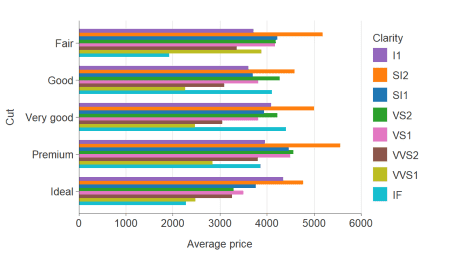
group\_by(cut, clarity) %>%

summarise(average\_price = mean(price))

gg\_hbar\_col(plot\_data, average\_price, cut, clarity,

col\_labels = ggplot2::waiver(),

pal\_rev = TRUE)



Reverse the order of coloured bars.

plot\_data <- ggplot2::diamonds %>%

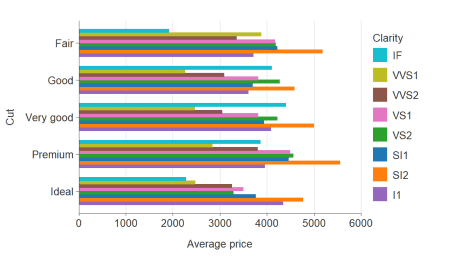
group\_by(cut, clarity) %>%

summarise(average\_price = mean(price))

gg\_hbar\_col(plot\_data, average\_price, cut, clarity,

col\_labels = ggplot2::waiver(),

col\_rev = TRUE)



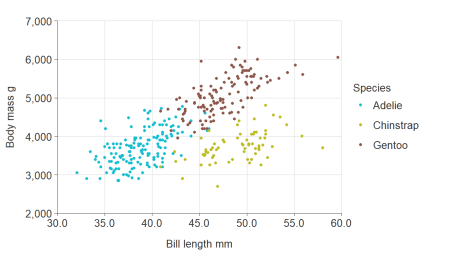
**Labels**

You can adjust x, y or colour scale labels using x\_labels, y\_labels or col\_labels arguments, and functions from the scales package.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species,

y\_labels = scales::comma\_format(),

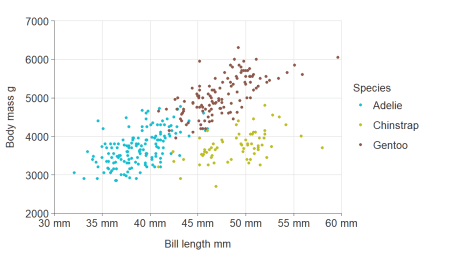
x\_labels = scales::number\_format(accuracy = 0.1))



Or via a function.

gg\_point\_col(penguins, bill\_length\_mm, body\_mass\_g, species,

x\_labels = function(x) glue::glue("{x} mm"))



Note there is a default sentence case transformation for categorical x, y or col variables. But you can use ggplot2::waiver() to turn this off.

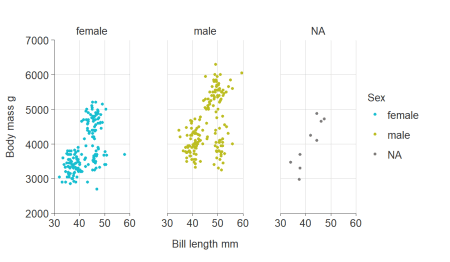
The facet\_labels argument works slightly differently in that it provides access to the ggplot2 labeller argument within the ggplot facet\_wrap function. Therefore you need to use ggplot labeller functions to modify them.

Notice with the default transformation turned off, the sex variable returns to being lower case as it is in the penguins dataset.

gg\_point\_col\_facet(penguins, bill\_length\_mm, body\_mass\_g, sex, sex,

col\_labels = ggplot2::waiver(),

facet\_labels = ggplot2::label\_value)

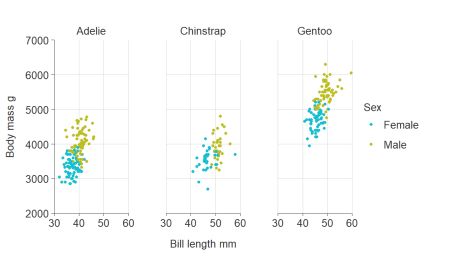


**NA values**

You can quickly remove NA values by setting x\_na, y\_na, col\_na or facet\_na arguments to FALSE.

gg\_point\_col\_facet(penguins, bill\_length\_mm, body\_mass\_g, sex, species,

col\_na = F)



**Expanding the scale**

To expand the scale use x\_expand and y\_expand arguments with the ggplot2::expansion function, which allows to expand in either or both directions of both x and y in an additive or multiplative way.

plot\_data <- storms %>%

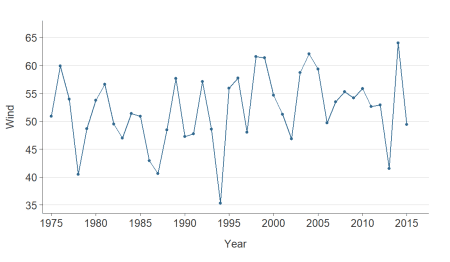
group\_by(year) %>%

summarise(wind = mean(wind))

gg\_line(plot\_data, year, wind,

x\_expand = ggplot2::expansion(add = c(1, 2.5)),

y\_expand = ggplot2::expansion(mult = c(0.05, 0.1)))



**Further information**

More blogs to come on simplevis.