chapter 6

# Scales, Axes and Legends

library(ggplot2)  
library(gridExtra)  
library(dplyr)

## 6.1 Introduction

* - Scales control the mapping from data to aesthetics.
* - Scales also provide the tools that let you read the plot: the axes and legends.

Formally, each scale is a function from a region in data space (the domain of the scale) to a region in aesthetic space (the range of the scale). The axis or legend is the inverse function: it allows you to convert visual properties back to data.

## 6.2 Modifying Scales

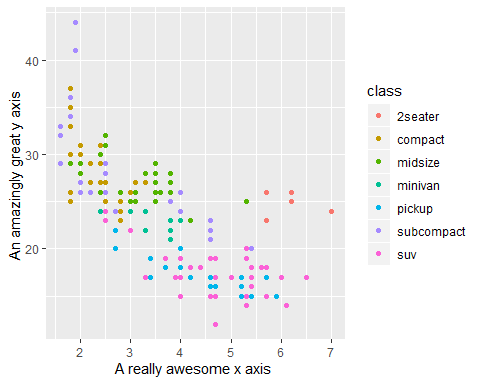
# when u write:   
ggplot(mpg, aes(displ, hwy)) +  
geom\_point(aes(colour = class))

-> actually happens (default) :

ggplot(mpg, aes(displ, hwy)) +  
geom\_point(aes(colour = class)) +  
scale\_x\_continuous() +  
scale\_y\_continuous() +  
scale\_colour\_discrete()

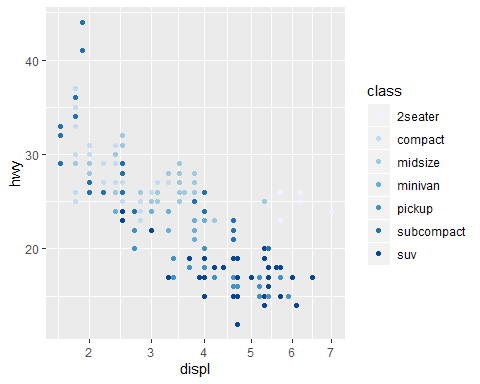
---------------------------------------------------------------------------------------------------------------------

ggplot(mpg, aes(displ, hwy)) +  
geom\_point(aes(colour = class)) +  
scale\_x\_continuous("A really awesome x axis ") +  
scale\_y\_continuous("An amazingly great y axis ")



But !! Use of + to “add” scales is not adding. It is overriding!!!

ggplot(mpg, aes(displ, hwy)) +  
geom\_point(aes(colour = class)) +  
scale\_x\_sqrt() +  
scale\_colour\_brewer()



\* naming scheme for scales : made up of three pieces separated by “\_"

1. scale
2. name of the aesthetic (color, shape, x …)
3. name of the scale (continuous, discrete, brewer…)

## 6.3 Guides : Legends and Axes

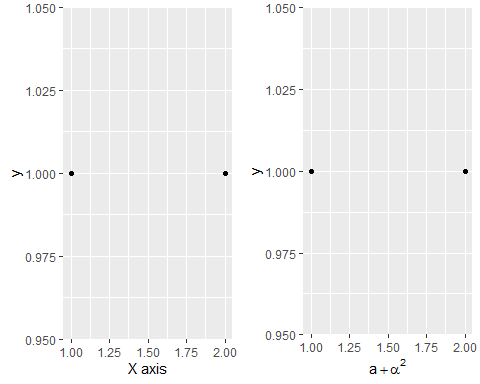
- axes & legends are the same type of thing!

|  |  |  |
| --- | --- | --- |
| Axis | Legend | Argument name |
| Label | title | *name* |
| Ticks & grid line | key | *breaks* |
| Tick label | Key label | *labels* |

### 6.3.1 Scale Title

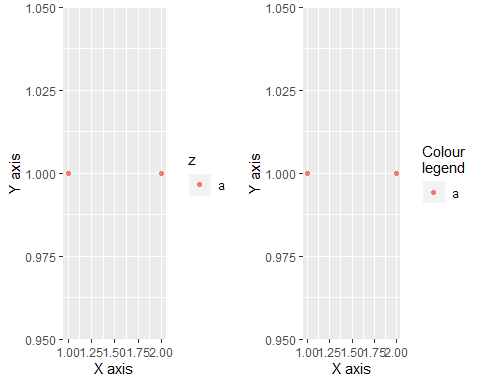
- the first argument to the scale function, **name**, is the axes/legend title.

df = data.frame(x = 1:2, y =1, z = "a")  
p = ggplot(df, aes(x,y)) + geom\_point()  
p1 = p + scale\_x\_continuous("X axis")  
p2 = p + scale\_x\_continuous(quote(a + alpha ^ 2)) # methematical expressions by quote  
grid.arrange(p1,p2, ncol=2)



-> save you some typing : xlab(), ylab(), labs()

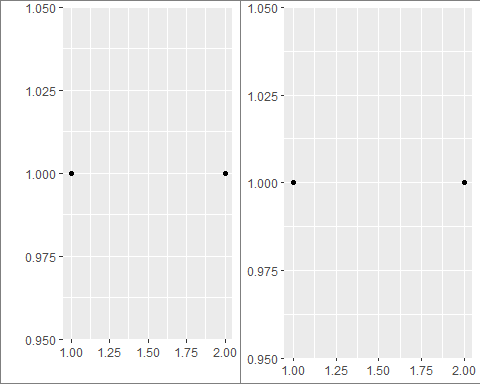
p <- ggplot(df, aes(x, y)) + geom\_point(aes(colour = z))  
  
p1 = p +  
 xlab("X axis") +  
 ylab("Y axis")  
  
p2 = p + labs(x = "X axis", y = "Y axis", colour = "Colour\nlegend")  
  
grid.arrange(p1,p2, ncol=2)



\* two ways to remove the axia label :

1. “” omits the label, but still allocates space
2. NULL removes the label and its space

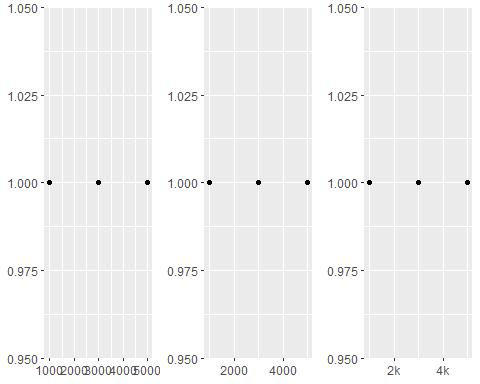
p <- ggplot(df, aes(x, y)) +  
geom\_point() +  
theme(plot.background = element\_rect(colour = "grey50"))  
  
p1 = p + labs(x = "", y = "")  
p2 = p + labs(x = NULL, y = NULL)  
  
grid.arrange(p1,p2, ncol=2)



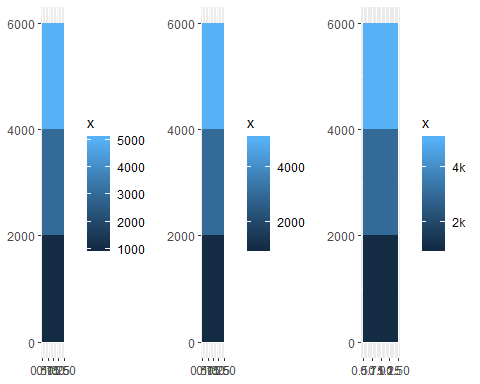
### 6.3.2 Breaks and Labels

**breaks** argument controls which values appear as tick marks on axes and keys on legends.

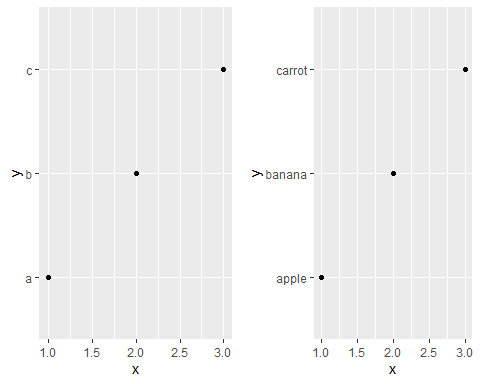
df = data.frame(x = c(1, 3, 5)\*1000, y = 1)  
axs = ggplot(df, aes(x, y)) +   
 geom\_point() +  
 labs(x = NULL, y = NULL)  
  
a1 = axs  
a2 = axs + scale\_x\_continuous(breaks = c(2000,4000))  
a3 = axs + scale\_x\_continuous(breaks = c(2000,4000), labels = c("2k", "4k"))  
  
grid.arrange(a1,a2,a3, ncol = 3)



leg <- ggplot(df, aes(y, x, fill = x)) +  
 geom\_tile() +  
 labs(x = NULL, y = NULL)  
  
l1 = leg  
l2 = leg + scale\_fill\_continuous(breaks = c(2000, 4000))  
l3 = leg + scale\_fill\_continuous(breaks = c(2000, 4000), labels = c("2k", "4k"))  
  
grid.arrange(l1,l2,l3,ncol = 3)

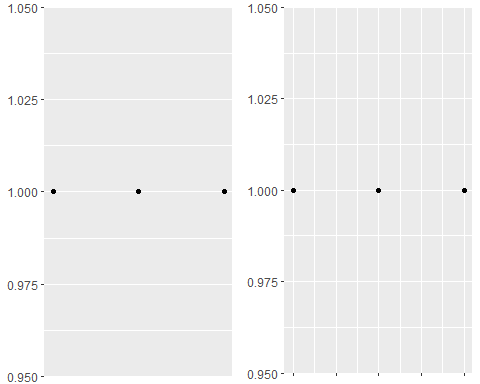


df2 <- data.frame(x = 1:3, y = c("a", "b", "c"))  
p1 = ggplot(df2, aes(x, y)) +  
 geom\_point()  
p2 = ggplot(df2, aes(x, y)) +  
 geom\_point() +  
 scale\_y\_discrete(labels = c(a = "apple", b = "banana", c = "carrot"))  
  
grid.arrange(p1, p2, ncol = 2)



To suppress breaks or labels : set them to NULL

a1 = axs + scale\_x\_continuous(breaks = NULL)  
a2 = axs + scale\_x\_continuous(labels = NULL)  
  
grid.arrange(a1, a2, ncol = 2)



u can supply a function to breaks or labels

- breaks function : should have one argument, the limits, and should return a numeric vector of breaks

- labels function : should accept a numeric vector of breaks and return a character vector of labels

usefull labelling functions : •

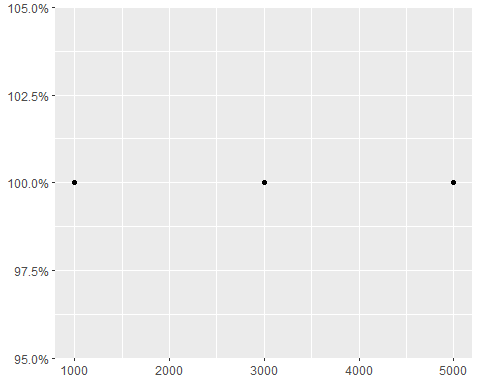
- scales::comma format() - adds commas to make it easier to read large numbers. •

- scales::unit format(unit, scale) - adds a unit suffix, optionally scaling.

- scales::dollar format(prefix, suffix) - displays currency values, rounding to two decimal places and adding a prefix or suffix. •

- scales::wrap format() - wraps long labels into multiple lines.

axs + scale\_y\_continuous(labels = scales::percent\_format())

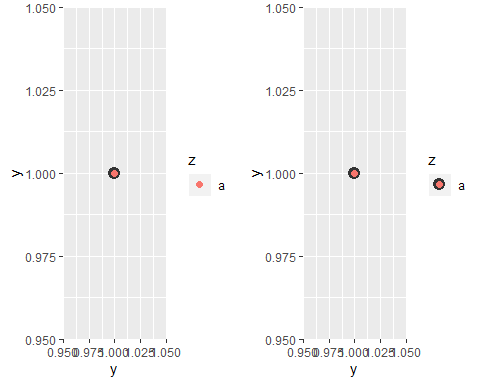


## 6.4 Legends

### 6.4.1 Layers and Legends

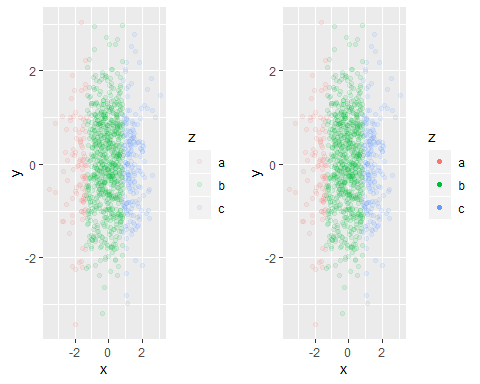
You can override whether or not a layer appears in the legend with : show.legend

df = data.frame(x = 1:2, y =1, z = "a")  
  
p1 = ggplot(df, aes(y,y)) +   
 geom\_point(size = 4, color = "grey20") +  
 geom\_point(aes(color = z), size = 2)  
  
p2 = ggplot(df, aes(y,y)) +   
 geom\_point(size = 4, color = "grey20", show.legend = TRUE) +  
 geom\_point(aes(color = z), size = 2)  
  
grid.arrange(p1, p2, ncol = 2)



You can display differently to the geoms in the plot : override.aes

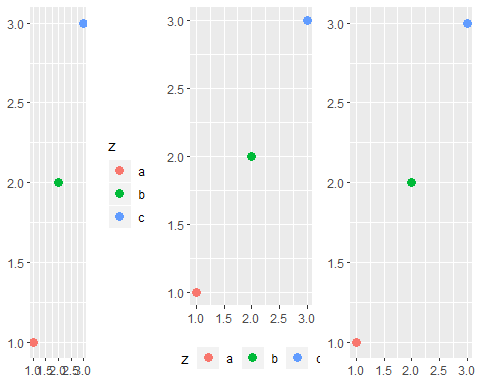
norm <- data.frame(x = rnorm(1000), y = rnorm(1000))  
norm$z <- cut(norm$x, 3, labels = c("a", "b", "c"))  
  
p1 = ggplot(norm, aes(x, y)) +  
geom\_point(aes(colour = z), alpha = 0.1)  
  
p2 = ggplot(norm, aes(x, y)) +  
geom\_point(aes(colour = z), alpha = 0.1) +  
guides(colour = guide\_legend(override.aes = list(alpha = 1)))  
  
grid.arrange(p1, p2, ncol = 2)



### 6.4.2 Legend Layout

overall display of the legends are controlled through the : theme()

df <- data.frame(x = 1:3, y = 1:3, z = c("a", "b", "c"))  
  
base <- ggplot(df, aes(x, y)) +  
geom\_point(aes(colour = z), size = 3) +  
xlab(NULL) +  
ylab(NULL)  
  
theme1 = base + theme(legend.position = "right") # the default  
theme2 = base + theme(legend.position = "bottom")  
theme3 = base + theme(legend.position = "none")  
  
grid.arrange(theme1, theme2, theme3, ncol = 3)

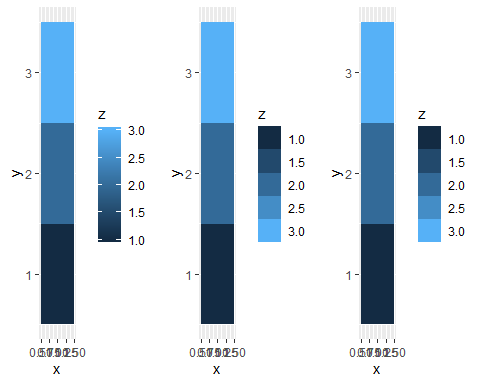


### 6.4.3 Guide Functions

\* guide\_colourbar()

* \* guide\_legend()

df <- data.frame(x = 1, y = 1:3, z = 1:3)  
base <- ggplot(df, aes(x, y)) + geom\_raster(aes(fill = z))  
  
g1 = base  
g2 = base + scale\_fill\_continuous(guide = guide\_legend())  
g3 = base + guides(fill = guide\_legend()) # guides() works like labs()  
  
grid.arrange(g1,g2,g3, ncol = 3)



guide\_legend() : usefull options!!

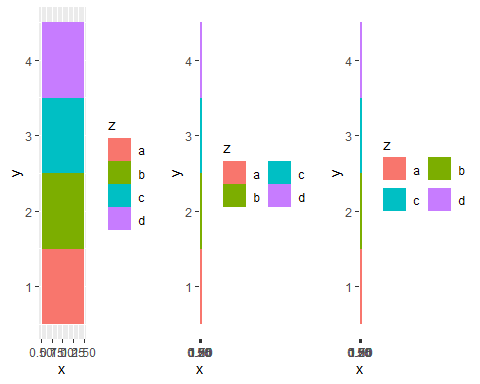
- nrow, ncol

- reverse

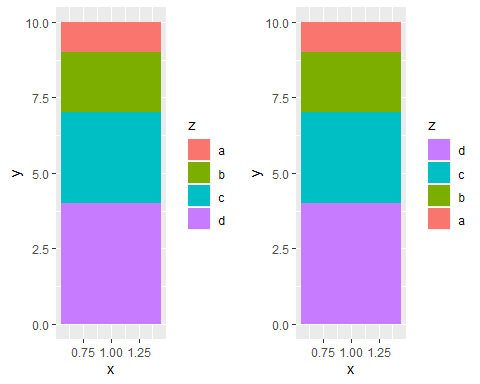
- override.aes

- keywidth, keyheight : specify the size of the keys

# 1. nrow, ncol  
df <- data.frame(x = 1, y = 1:4, z = letters[1:4])  
p <- ggplot(df, aes(x, y)) + geom\_raster(aes(fill = z))  
p1 = p  
p2 = p + guides(fill = guide\_legend(ncol = 2))  
p3 = p + guides(fill = guide\_legend(ncol = 2, byrow = TRUE))  
  
grid.arrange(p1,p2,p3, ncol = 3)



# 2. reverse  
p <- ggplot(df, aes(1, y)) + geom\_bar(stat = "identity", aes(fill = z))  
p1 = p  
p2= p + guides(fill = guide\_legend(reverse = TRUE))  
  
grid.arrange(p1,p2, ncol = 2)

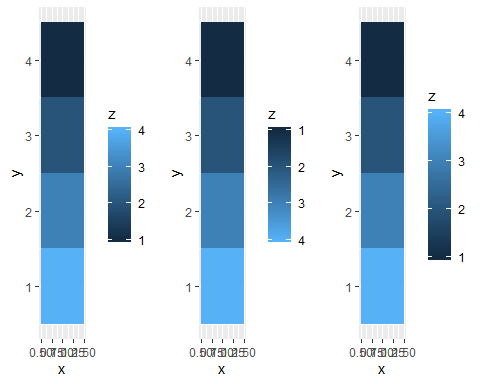


guide\_colourbar

- barwidth and barheight (along with default.unit) : allow you to specify the size of the bar. These are grid units, e.g. unit(1, “cm”).

- nbin : controls the number of slices. - reverse : flips the colour bar to put the lowest values at the top.

df <- data.frame(x = 1, y = 1:4, z = 4:1)  
p <- ggplot(df, aes(x, y)) + geom\_tile(aes(fill = z))  
p1 = p  
p2 = p + guides(fill = guide\_colorbar(reverse = TRUE))  
p3 = p + guides(fill = guide\_colorbar(barheight = unit(4, "cm")))  
  
grid.arrange(p1,p2,p3, ncol = 3)

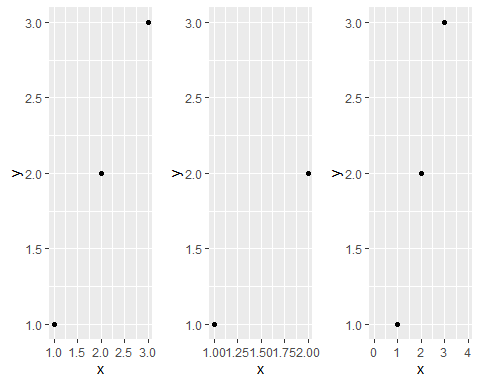


## 6.5 Limits

* - For continuous scales : numeric vector of length two
* - For discrete scales : charatoer vector which enumerates all possible values
* - you only want to set the upper or lower limit, set the other value to NA

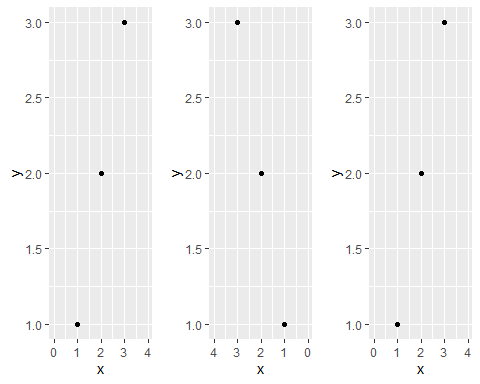
df = data.frame(x = 1:3, y = 1:3)  
base = ggplot(df, aes(x, y)) + geom\_point()  
  
p1 = base  
p2 = base + scale\_x\_continuous(limits = c(1,2))  
p3 = base + scale\_x\_continuous(limits = c(0,4))  
  
grid.arrange(p1,p2,p3, ncol = 3)

## Warning: Removed 1 rows containing missing values (geom\_point).



easier : xlim(), ylim(), lims()

p1 = base + xlim(0, 4)  
p2 = base + xlim(4, 0)  
p3 = base + lims(x = c(0, 4))  
  
grid.arrange(p1,p2,p3, ncol = 3)



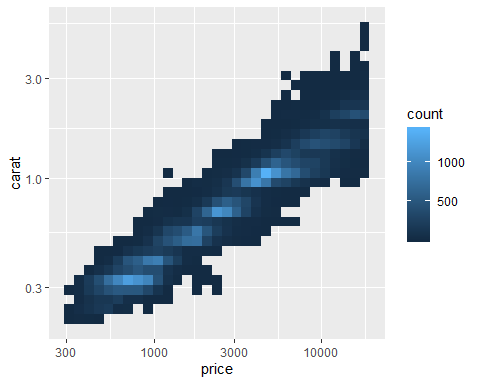
## 6.6 Scales Toolbox

### 6.6.1 Continuous Position Scales

The most common continuous position scales are

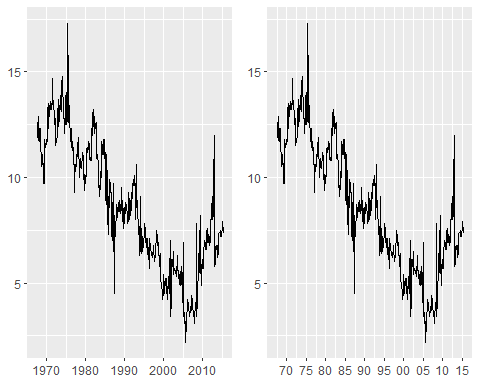
- scale\_x\_continuous scale\_y\_continuous

# trans argument  
# there are many "transfomer"  
  
ggplot(diamonds, aes(price, carat)) +  
 geom\_bin2d() +  
 scale\_x\_continuous(trans = "log10") +  
 scale\_y\_continuous(trans = "log10")



If you use a transformed scale, the axes will be labelled in the original data space; if you transform the data, the axes will be labelled in the transformed space.

# Data & date/time data  
# -> date\_breaks : position breaks by date units  
# -> date\_labels : controls the display of the labels  
base <- ggplot(economics, aes(date, psavert)) +  
geom\_line(na.rm = TRUE) +  
 labs(x = NULL, y = NULL)  
  
p1 = base # Default breaks and labels  
p2 = base + scale\_x\_date(date\_labels = "%y", date\_breaks = "5 years")  
grid.arrange(p1,p2,ncol = 2)



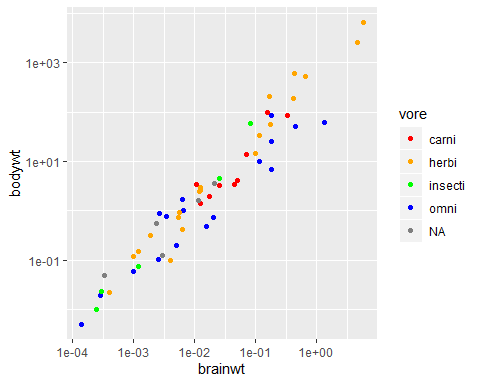
### 6.6.2 Colour

### 6.6.3 The Manual Discrete Scale

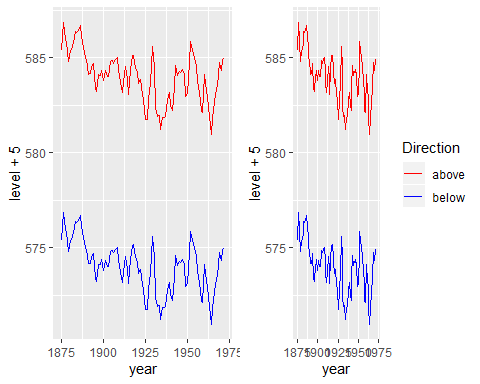
To customise these scales, create your own new scale with the manual scale!

plot <- ggplot(msleep, aes(brainwt, bodywt)) +  
 scale\_x\_log10() +  
 scale\_y\_log10()  
  
colours <- c(  
carni = "red",  
insecti = "orange",  
herbi = "green",  
omni = "blue"  
)  
  
plot +  
 geom\_point(aes(colour = vore)) +  
 scale\_colour\_manual(  
 values = c("red", "orange", "green", "blue"), # you can : values = colours   
 na.value = "grey50"  
 )

## Warning: Removed 27 rows containing missing values (geom\_point).



huron <- data.frame(year = 1875:1972, level = as.numeric(LakeHuron))  
  
p1 = ggplot(huron, aes(year)) +   
 geom\_line(aes(y = level + 5), colour = "red") +  
 geom\_line(aes(y = level - 5), colour = "blue")   
  
p2 = ggplot(huron, aes(year)) +  
 geom\_line(aes(y = level + 5, colour = "above")) +  
 geom\_line(aes(y = level - 5, colour = "below")) +  
 scale\_colour\_manual("Direction",  
 values = c("above" = "red", "below" = "blue")  
 )  
  
grid.arrange(p1,p2,ncol=2)



### 6.6.4 The identity Scale

Used when data is already scaled!

head(luv\_colours)

## L u v col  
## 1 9341.570 -3.370649e-12 0.0000 white  
## 2 9100.962 -4.749170e+02 -635.3502 aliceblue  
## 3 8809.518 1.008865e+03 1668.0042 antiquewhite  
## 4 8935.225 1.065698e+03 1674.5948 antiquewhite1  
## 5 8452.499 1.014911e+03 1609.5923 antiquewhite2  
## 6 7498.378 9.029892e+02 1401.7026 antiquewhite3

ggplot(luv\_colours, aes(u, v)) +  
 geom\_point(aes(colour = col), size = 3) +  
 scale\_color\_identity() +  
 coord\_equal()

