Chapter7 : Positioning

```
library(ggplot2)
library(gridExtra)
mpg2 <- subset(mpg, cyl != 5 & drv %in% c("4", "f") & class != "2seater")
```

7.1 Introduction

- how facets are laid out on a page
- how coordinate systems within a panel work

* 4 components that control position

- Position adjustments : adjust overlapping
- Position scales : control how the values in the data mapped to positions on the plot
- Facetting : mechanism for automatically laying out multiple plots on a page
- Coordinate systems : control how the two independent position scales are combined to create a 2d coordinate system

7.2 Facetting

* 3 types of facetting

- facet_null() : single plot, default
- facet_wrap() : "wraps" a 1d ribbon of panels into 2d
- facet_grid() : produces a 2d grid of panels defined by variables which form the row & columns

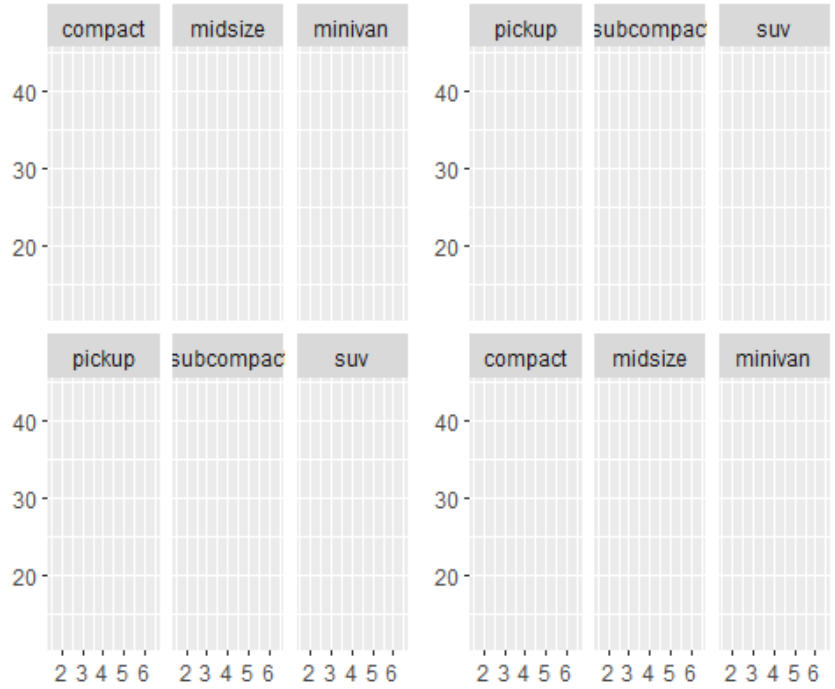
7.2.1 Facet Wrap

Useful if you have a single variable with many levels and want to arrange the plots in a more space efficient manner.

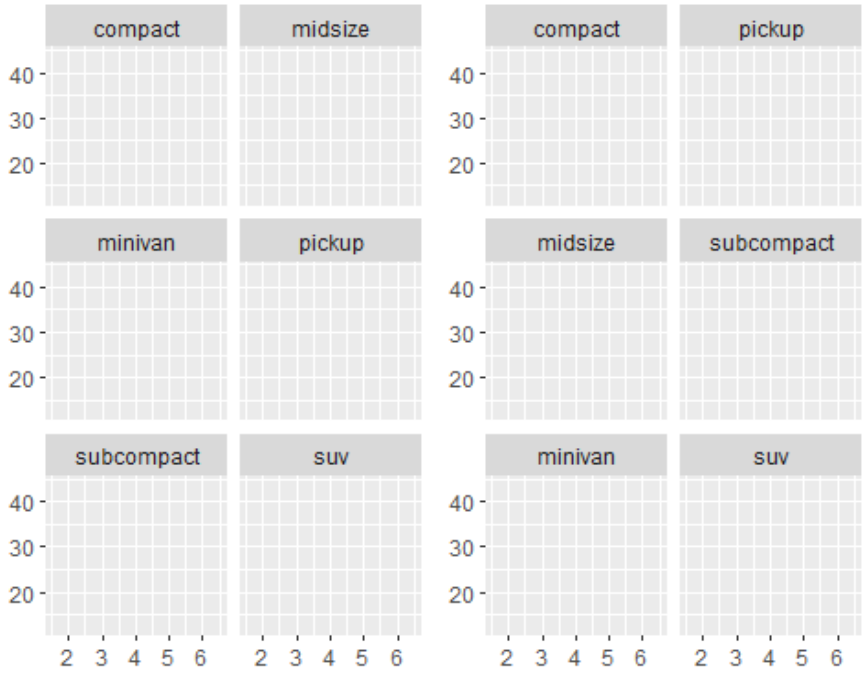
- *ncol, nrow* : control how many columns and rows
- *as.table* : TRUE -> facets are laid out like a table
- *dir* : controls the direction of wrap(h or v)

```
base <- ggplot(mpg2, aes(displ, hwy)) +
  geom_blank() +
  xlab(NULL) +
  ylab(NULL)

b1 = base + facet_wrap(~class, ncol = 3)
```

```
b2 = base + facet_wrap(~class, ncol = 3, as.table = FALSE)
b3 = base + facet_wrap(~class, nrow = 3)
b4 = base + facet_wrap(~class, nrow = 3, dir = "v")

grid.arrange(b1, b2, ncol = 2)
```



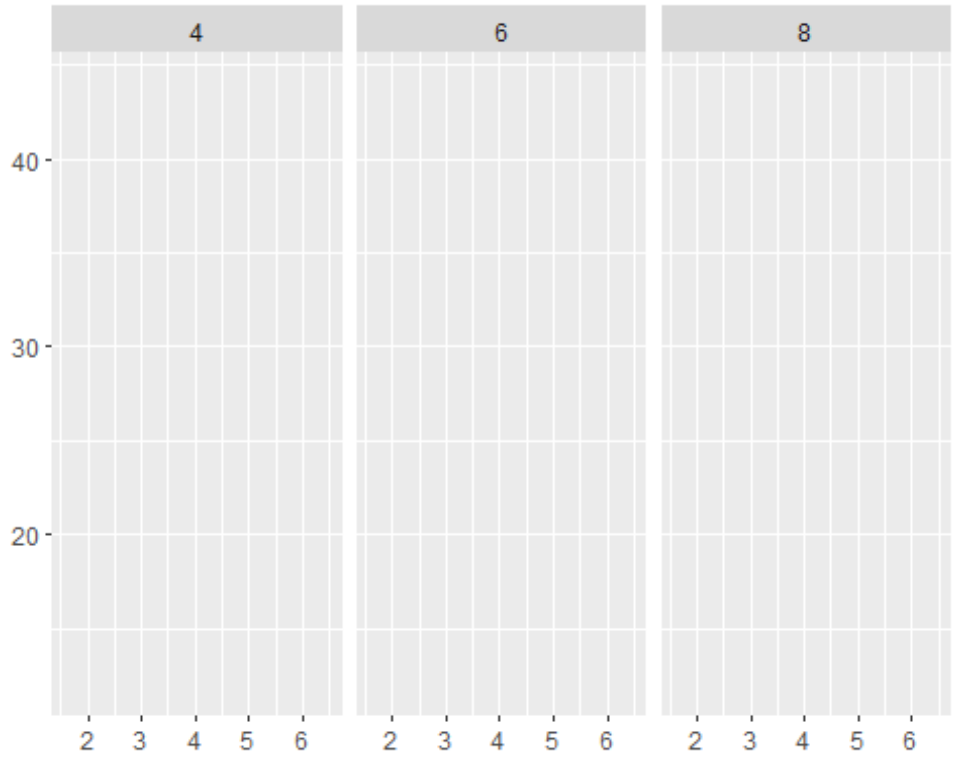

```
grid.arrange(b3, b4, ncol = 2)
```



7.2.2 Facet Grid

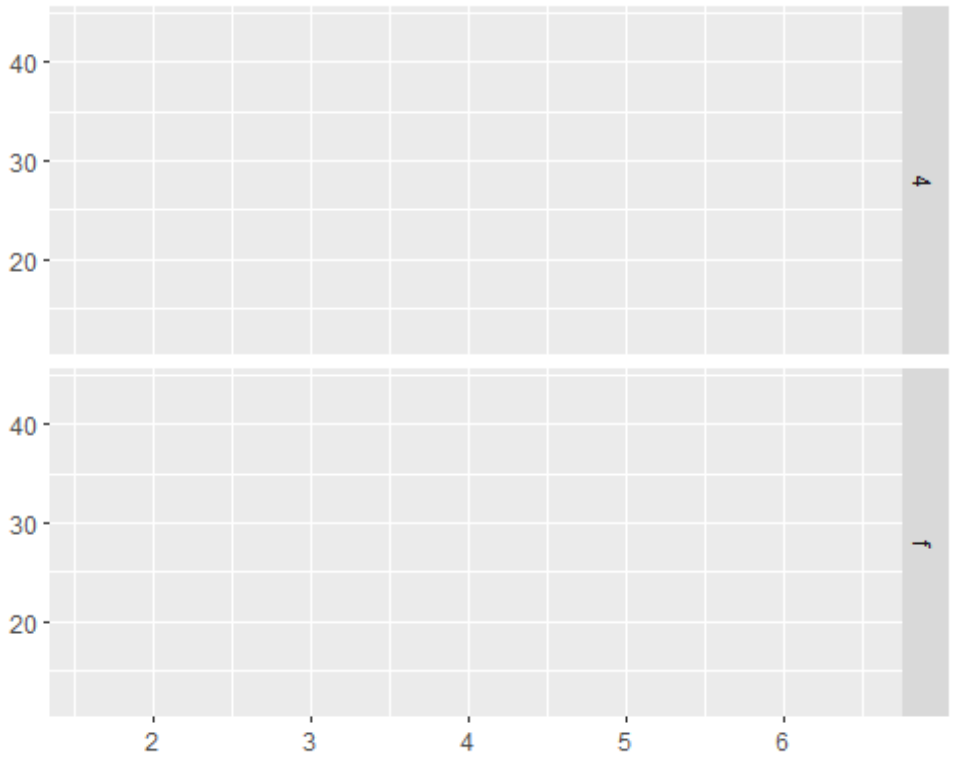
- . ~ : spreads the values across the columns

```
base + facet_grid(. ~ cyl)
```



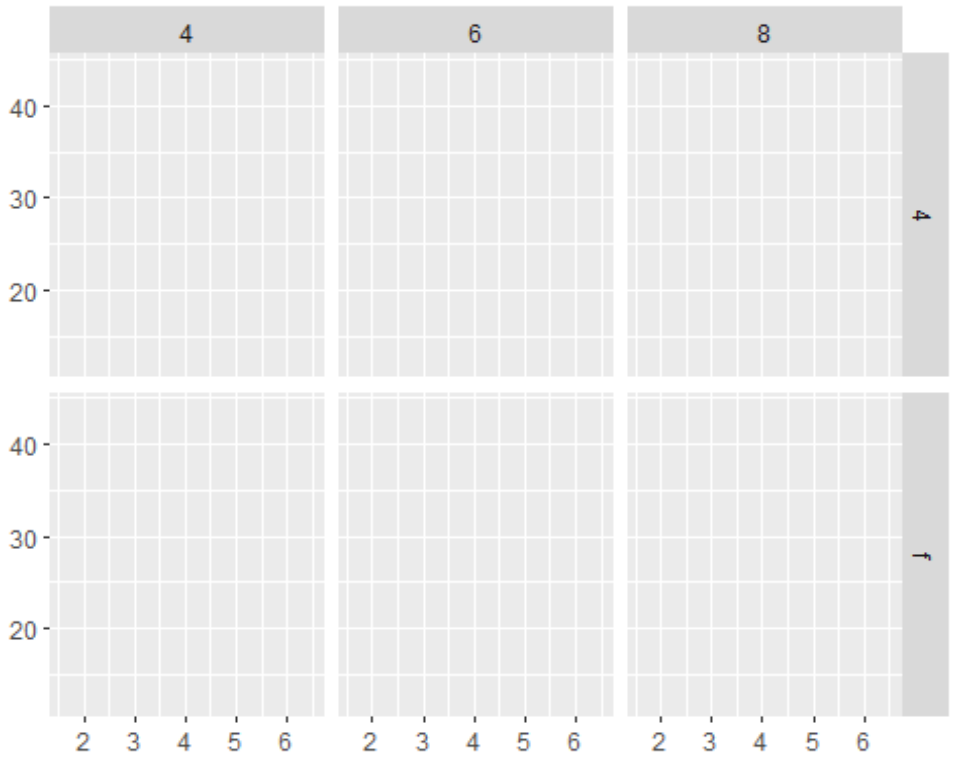
- ~ . : spreads the values across the rows

```
base + facet_grid(drv ~ .)
```



- ~ : across col and row

```
base + facet_grid(drv ~ cyl)
```

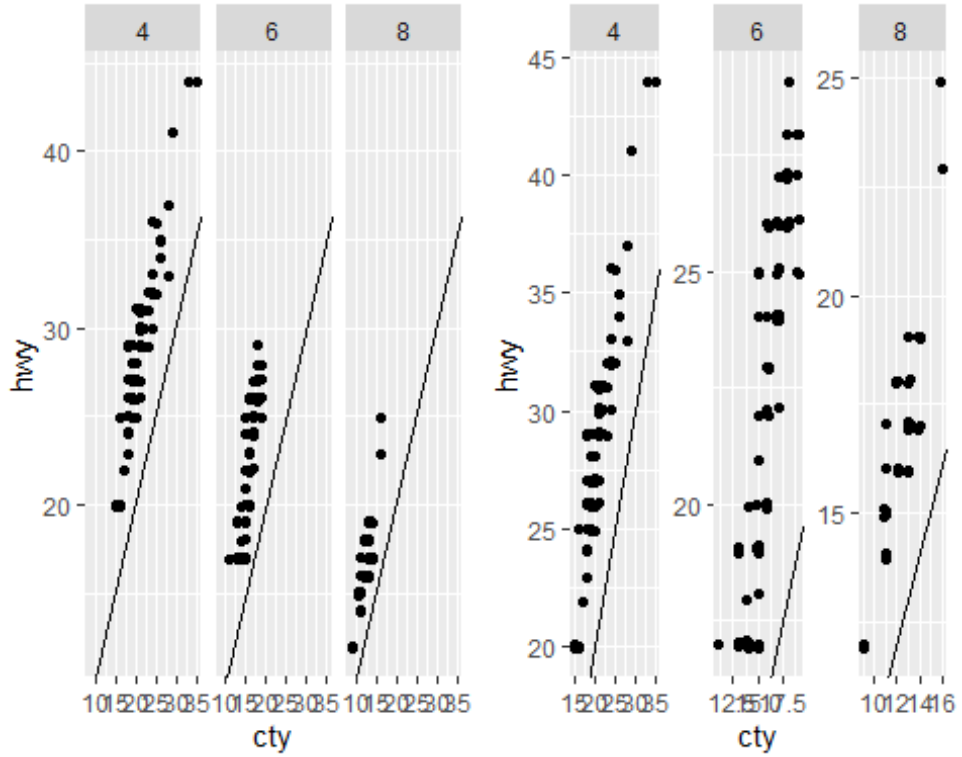


7.2.3 Controlling Scales

facet_warp(), facet_grid() both can control whether the position scales are the same or vary : scales

- scales = "fixed": x and y scales are fixed across all panels.
- scales = "free x": the x scale is free, and the y scale is fixed.
- scales = "free y": the y scale is free, and the x scale is fixed.
- scales = "free": x and y scales vary across panels.

```
p = ggplot(mpg2, aes(cty, hwy)) +
  geom_abline() +
  geom_jitter(width = 0.1, height = 0.1)

p1 = p + facet_wrap(~cyl)
p2 = p + facet_wrap(~cyl, scales = "free")

grid.arrange(p1,p2, ncol = 2)
```

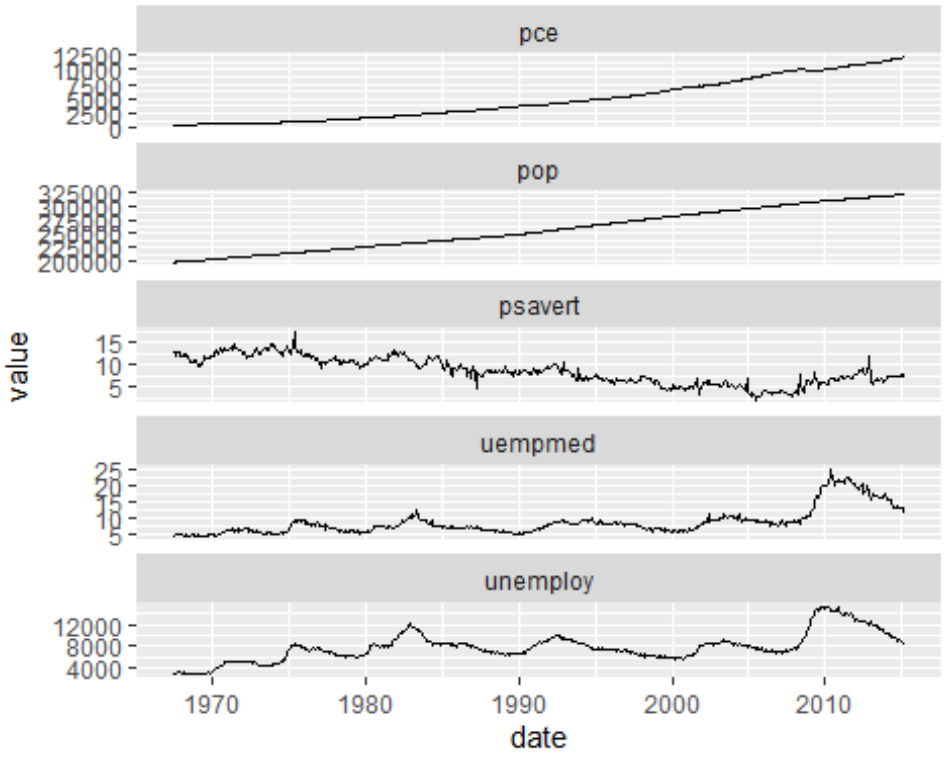


Free scales are useful when display time series!

```
head(economics_long)
```

```
## # A tibble: 6 x 4
##   date       variable value  value01
##   <date>     <chr>    <dbl>    <dbl>
## 1 1967-07-01 pce       507. 0
## 2 1967-08-01 pce       510. 0.000265
## 3 1967-09-01 pce       516. 0.000762
## 4 1967-10-01 pce       512. 0.000471
## 5 1967-11-01 pce       517. 0.000916
## 6 1967-12-01 pce       525. 0.00157
```

```
ggplot(economics_long, aes(date, value)) +
  geom_line() +
  facet_wrap(~variable, scales = "free_y", ncol = 1)
```

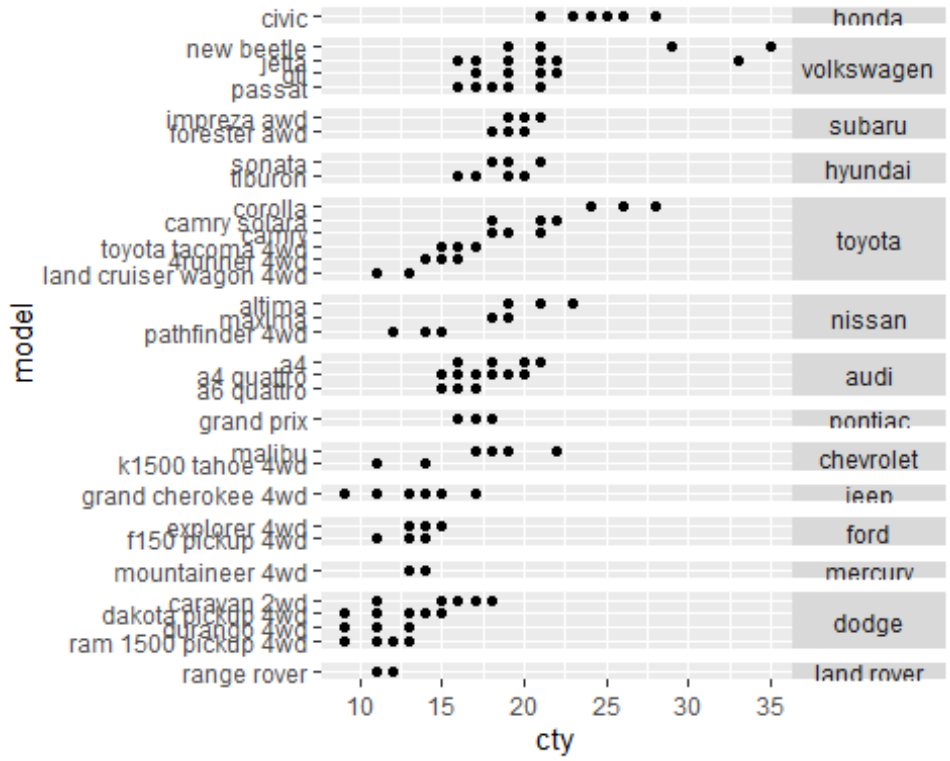


facet_grid() has additional parameter : space

- space = "free" => each column (or row) will have width (or height) proportional to the range of the scale

```
mpg2$model <- reorder(mpg2$model, mpg2$cty)
mpg2$manufacturer <- reorder(mpg2$manufacturer, -mpg2$cty)

ggplot(mpg2, aes(cty, model)) +
  geom_point() +
```

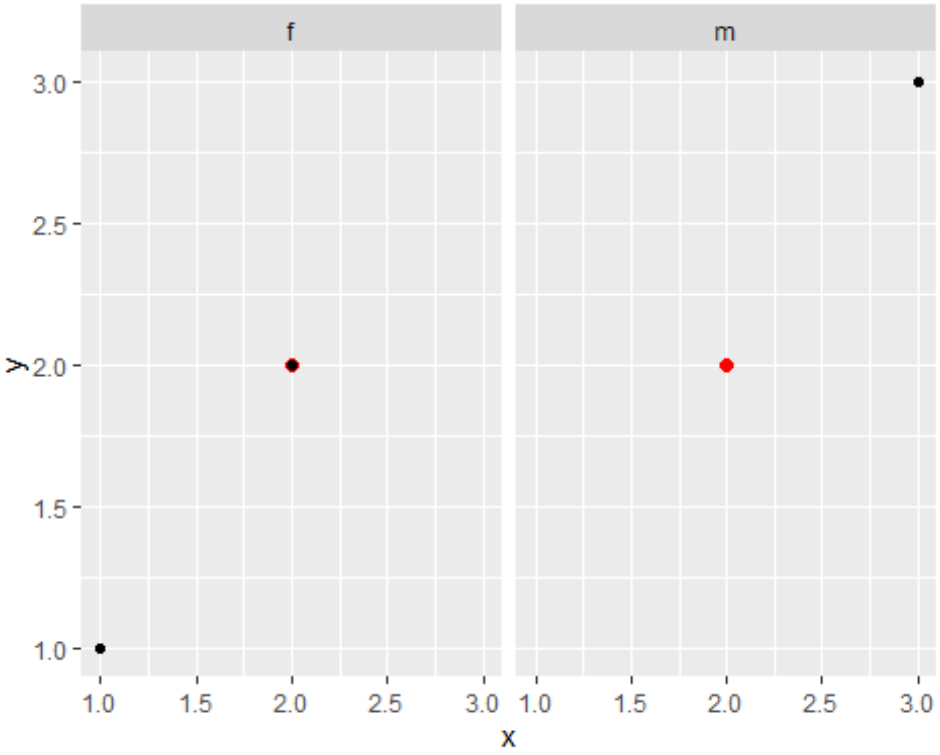
```
  facet_grid(manufacturer ~ ., scales = "free", space = "free") +
  theme(strip.text.y = element_text(angle = 0))
```



7.2.4 Missing Facetting Variables

it will display the map in every facet: missing facetting variables are treated like they have all values.

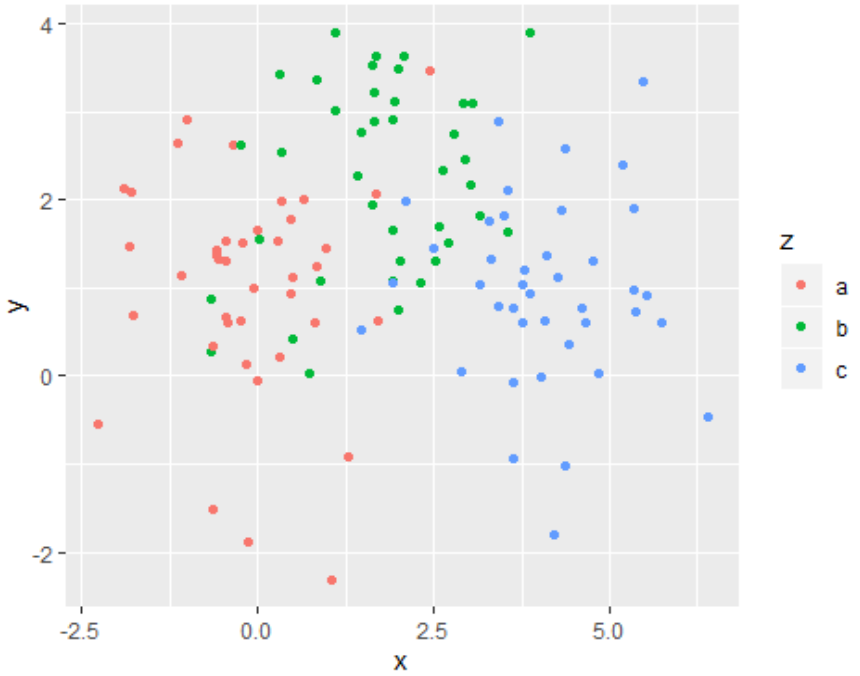
```
df1 <- data.frame(x = 1:3, y = 1:3, gender = c("f", "f", "m"))
df2 <- data.frame(x = 2, y = 2)

ggplot(df1, aes(x, y)) +
  geom_point(data = df2, colour = "red", size = 2) + # df2 does not have gend
er : but it's okay!!
  geom_point() +
  facet_wrap(~gender)
```



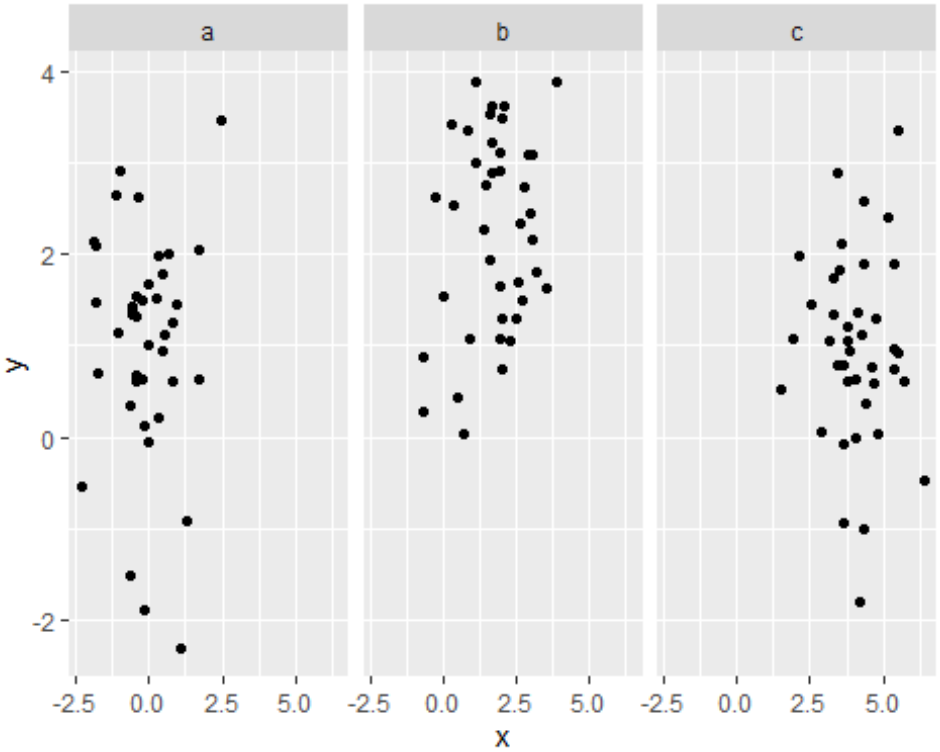
7.2.5 Grouping vs Facetting

When using aesthetics to differentiate groups, the groups are close together and may overlap, but small differences are easier to see.

```
df <- data.frame(
x = rnorm(120, c(0, 2, 4)),
y = rnorm(120, c(1, 2, 1)),
z = letters[1:3]
)

ggplot(df, aes(x, y)) +
  geom_point(aes(colour = z))
```

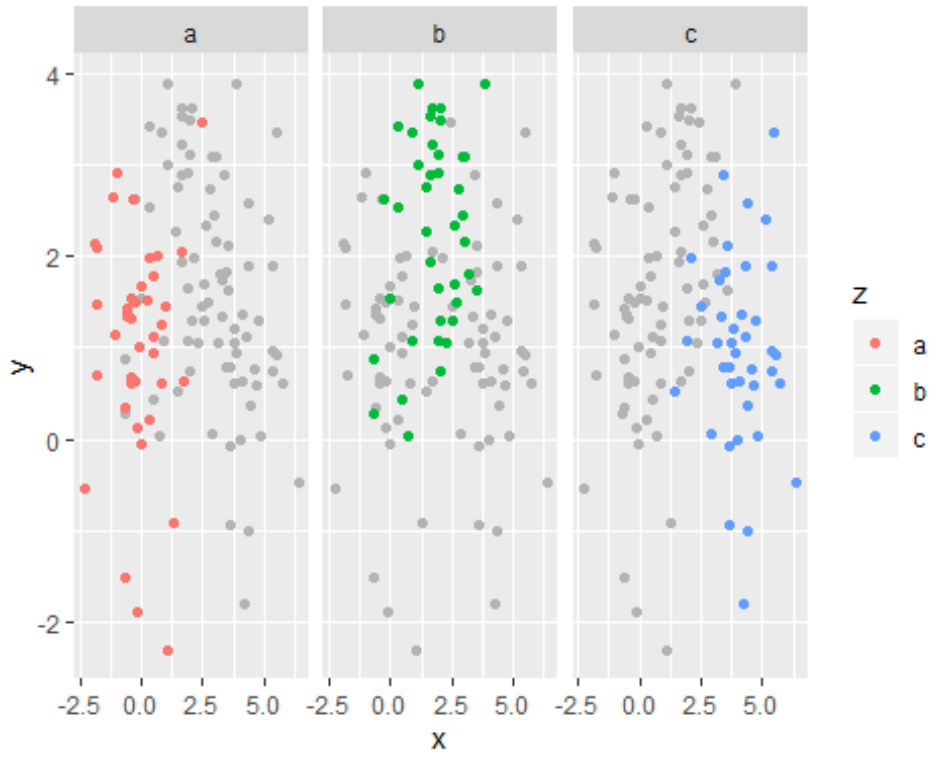



```
ggplot(df, aes(x, y)) +
  geom_point() +
  facet_wrap(~z)
```



useful technique ~ !

```
df2 <- dplyr::select(df, -z)

ggplot(df, aes(x, y)) +
  geom_point(data = df2, colour = "grey70") +
  geom_point(aes(colour = z)) +
  facet_wrap(~z)
```

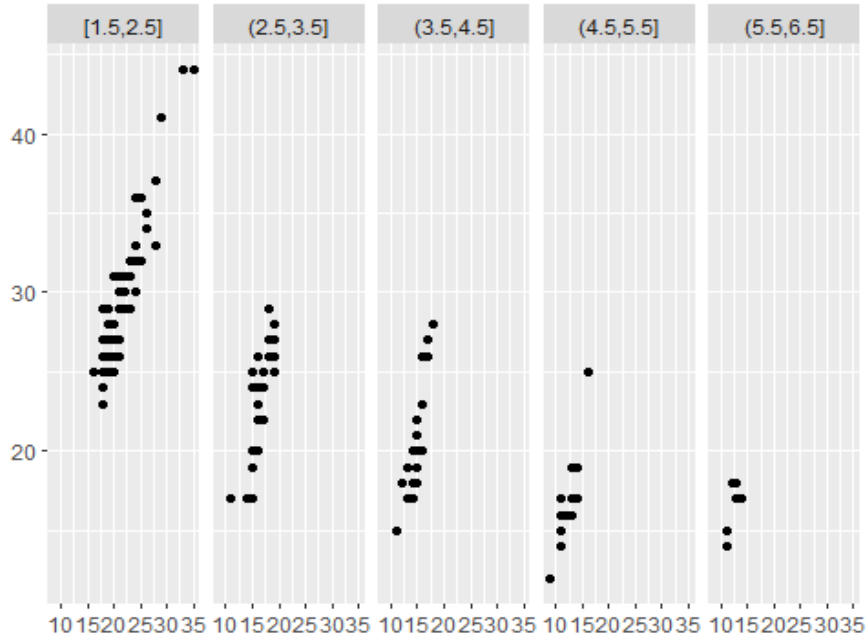


7.2.6 Continuous Variables

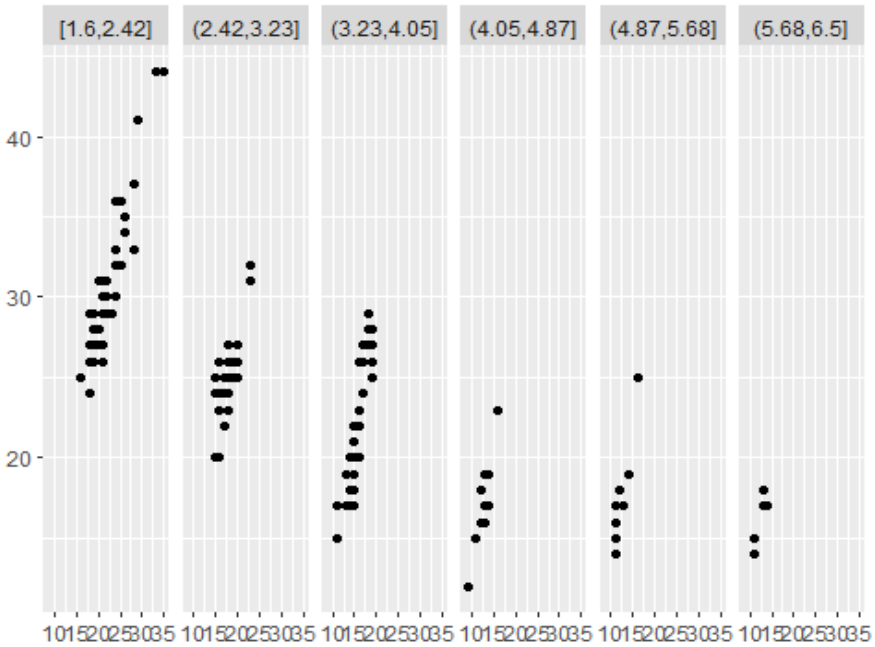
- Divide the data into n bins each of the same length: cut_interval(x, n)
- Divide the data into bins of width: cut_width(x, width)
- Divide the data into n bins each containing (approximately) the same number of points: cut_number(x, n = 10)

```
# Bins of width 1
mpg2$disp_w <- cut_width(mpg2$displ, 1)
# Six bins of equal length
mpg2$disp_i <- cut_interval(mpg2$displ, 6)
# Six bins containing equal numbers of points
mpg2$disp_n <- cut_number(mpg2$displ, 6)
```

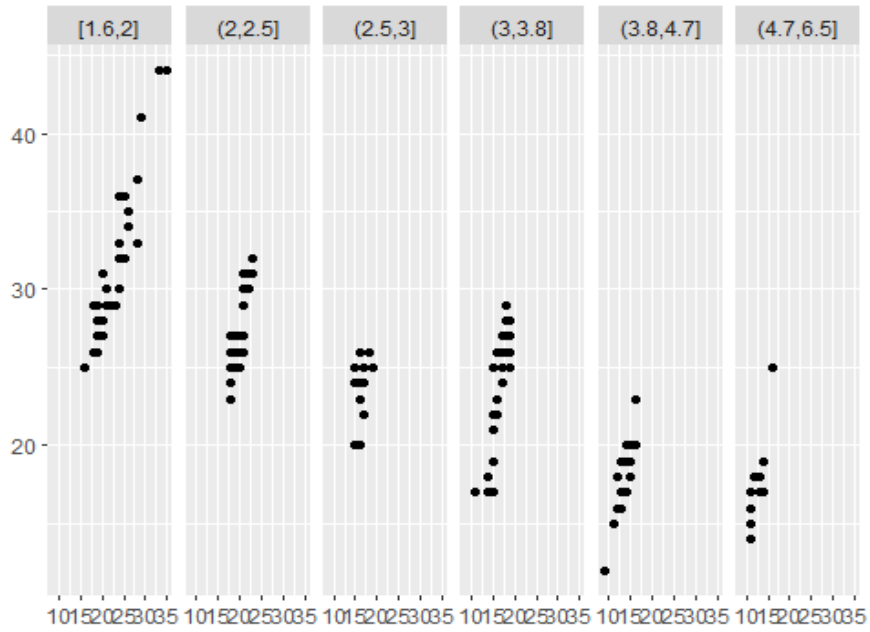
```
plot = ggplot(mpg2, aes(cty, hwy)) +
  geom_point() +
  labs(x = NULL, y = NULL)

plot + facet_wrap(~disp_w, nrow = 1)
```




```
plot + facet_wrap(~disp_i, nrow = 1)
```



```
plot + facet_wrap(~disp_n, nrow = 1)
```



7.3 Coordinate Systems

Main jobs :

1. Combine the two position aesthetics to produce a 2d position on the plot.

2. In coordination with the faceter, coordinate systems draw axes and panel backgrounds. While the scales control the values that appear on the axes, and how they map from data to position, it is the coordinate system which actually draws them.

There are two types of coordinate system!

1. linear

- coord cartesian() : the default Cartesian coordinate system, where the 2d position of an element is given by the combination of the x and y positions.
- coord flip() : Cartesian coordinate system with x and y axes flipped.
- coord fixed() : Cartesian coordinate system with a fixed aspect ratio.

2. non-linear

- coord map() / coord quickmap() : Map projections.
- coord polar(): Polar coordinates.
- coord trans() : Apply arbitrary transformations to x and y positions, after the data has been processed by the stat.

7.4 Linear Coordinate Systems

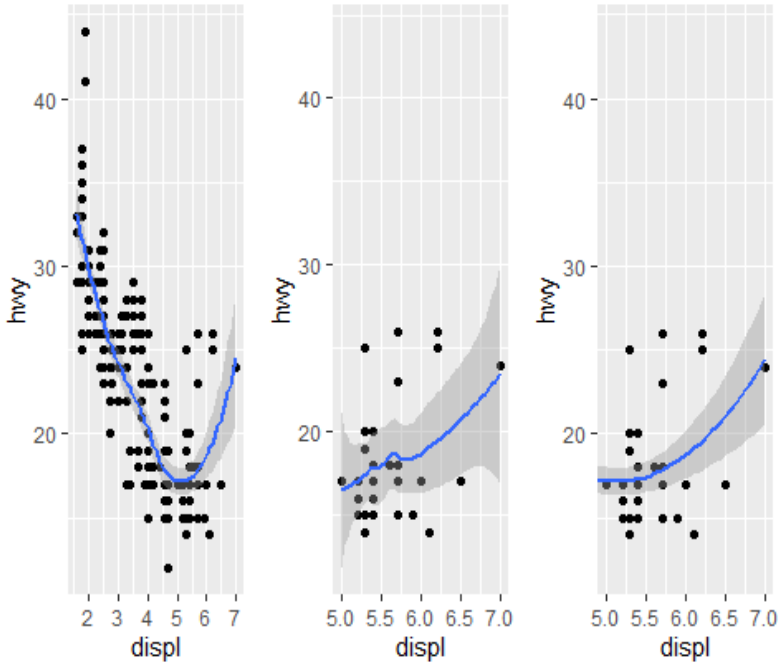
7.4.1 coord_cartesian()

- arguments : xlim(), ylim()

The key difference is how the limits work: when setting scale limits, any data outside the limits is thrown away; but when setting coordinate system limits we still use all the data, but we only display a small region of the plot.

```
base <- ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth()

b1 = base
b2 = base + scale_x_continuous(limits = c(5,7))
b3 = base + coord_cartesian(xlim = c(5, 7))
grid.arrange(b1,b2,b3, ncol = 3)
```

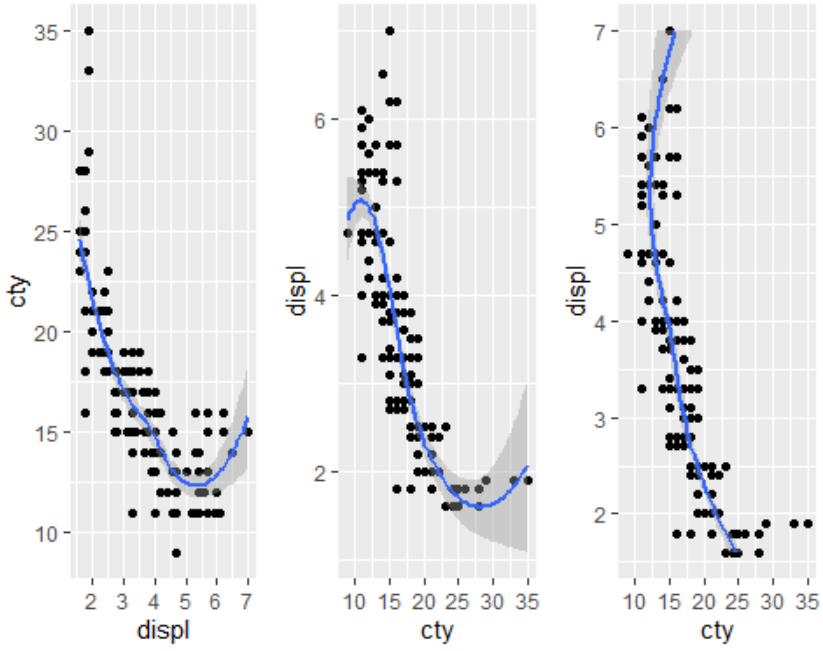
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning: Removed 196 rows containing non-finite values (stat_smooth).

## Warning: Removed 196 rows containing missing values (geom_point).

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



7.4.2 coord_flip()

- If you are interested in x conditional on y (or you just want to rotate the plot 90 degrees)

```
g1 = ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  geom_smooth()
# Exchanging cty and displ rotates the plot 90 degrees, but the smooth
# is fit to the rotated data.
g2 = ggplot(mpg, aes(cty, displ)) +
  geom_point() +
  geom_smooth()
# coord_flip() fits the smooth to the original data, and then rotates
# the output
g3 = ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  geom_smooth() +
  coord_flip()

grid.arrange(g1,g2,g3,ncol = 3)
```



7.4.3 coord_fixed()

- fixes the ratio of length on the x and y axes

7.5 Non-linear Coordinate Systems

- skip