Styling ggplot2 graphics

Rolf Bänziger, r.banziger@westminster.ac.uk

17/10/2022

Introduction

We know the three key components of a statistical graphic (data, aesthetics and geometries) and how they fit in the ggplot2 framework. But these three key components are not enough to produce graphics used in the real world, and ggplot2 does have additional components. In this lab, we explore how to change the appearance of graphics.

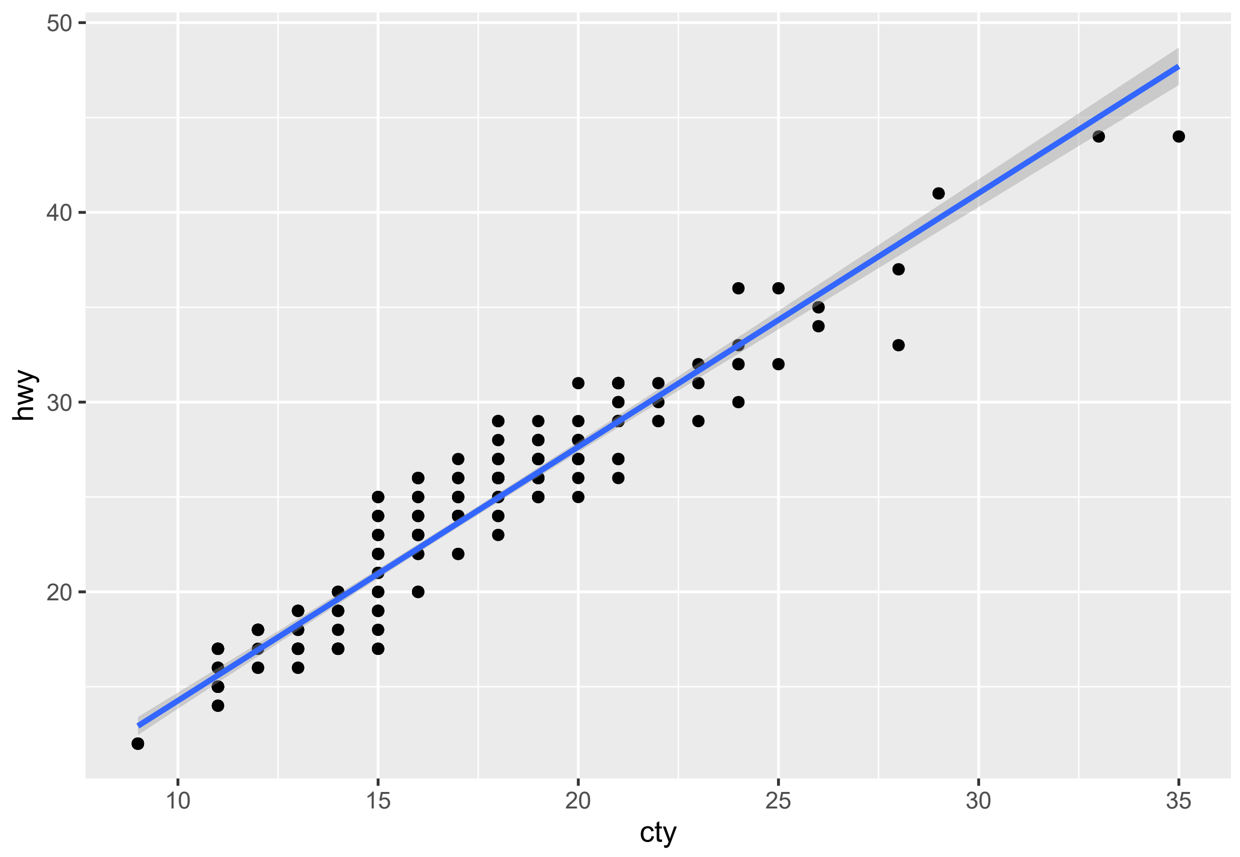
Style a graphic

Let's investigate if city MPG and highway MPG are correlated. We map highway mileage (hwy) to the x-axis and city mileage (cty) to the y-axis, and add a scatter plot layer using `geom_jitter`. The jitter geom is a shortcut for `geom_point(position = "jitter")` and adds a small amount of random variation to the location of each point to avoid overplotting (i.e. multiple points on the exact same location). Cf. [https://ggplot2.tidyverse.org/reference/geom_jitter.html]. We also fit a trend line using linear regressions(`method = "lm"`) with `geom_smooth()`.

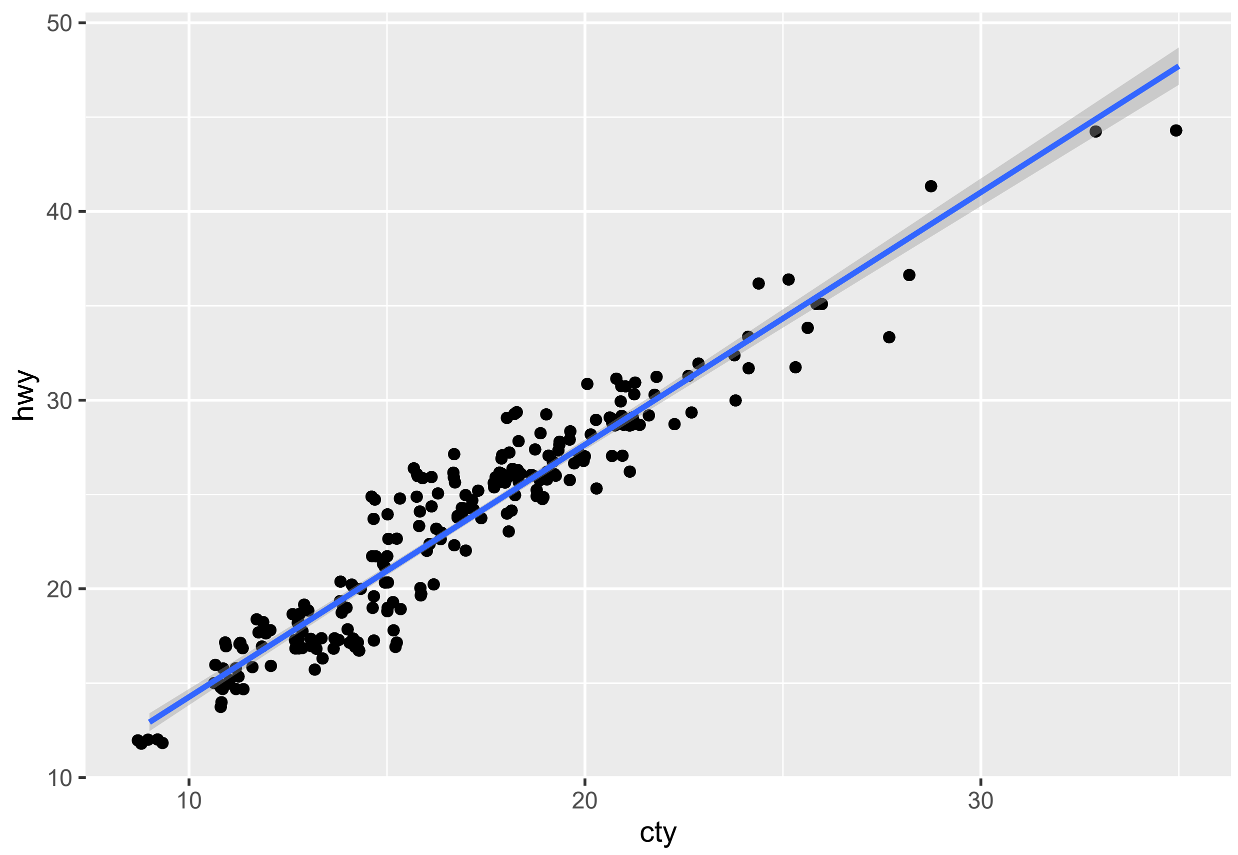
Don't forget to load the package `ggplot2` first.

```
library(ggplot2)
```

```
ggplot(mpg) + aes(cty, hwy) + geom_point() +
  geom_smooth(method = "lm")
```

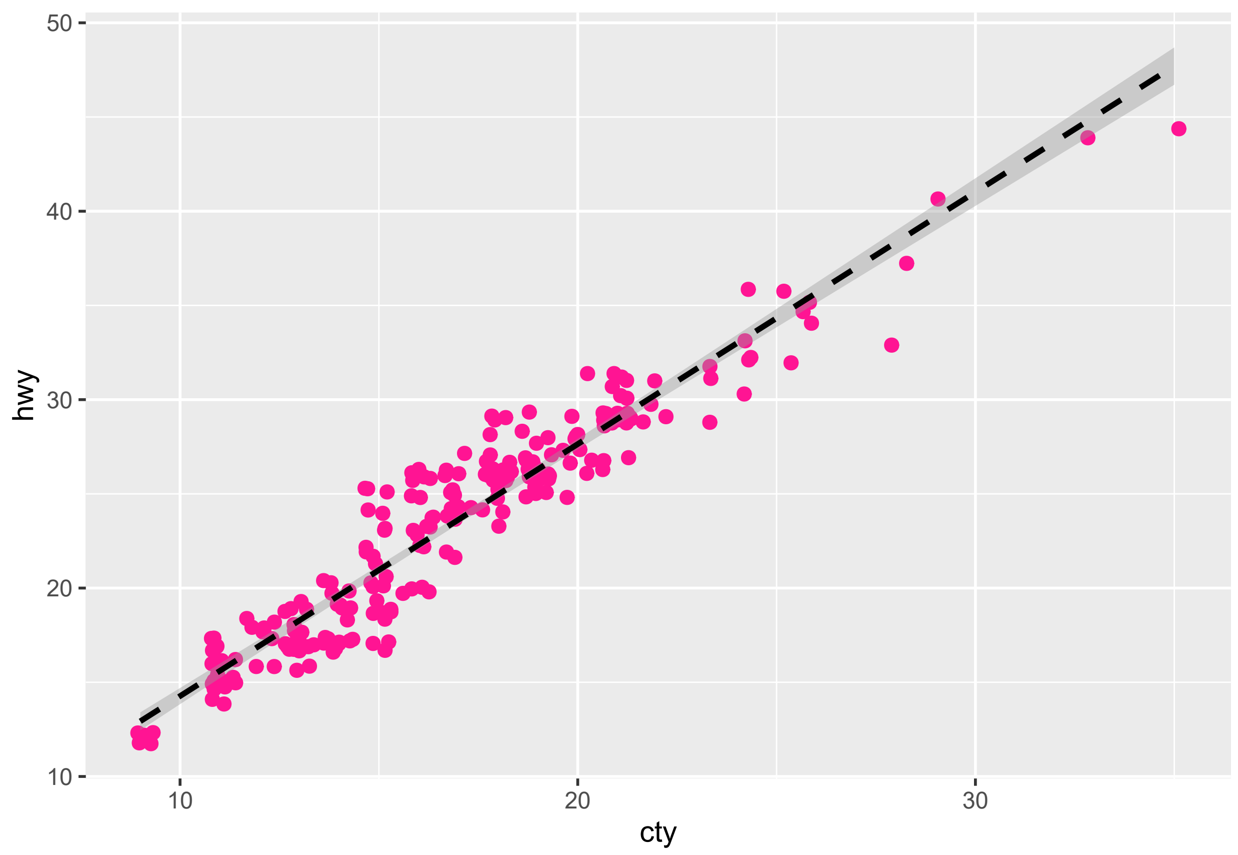


```r
ggplot(mpg) + aes(cty, hwy) + geom_jitter() +
  geom_smooth(method = "lm")
```

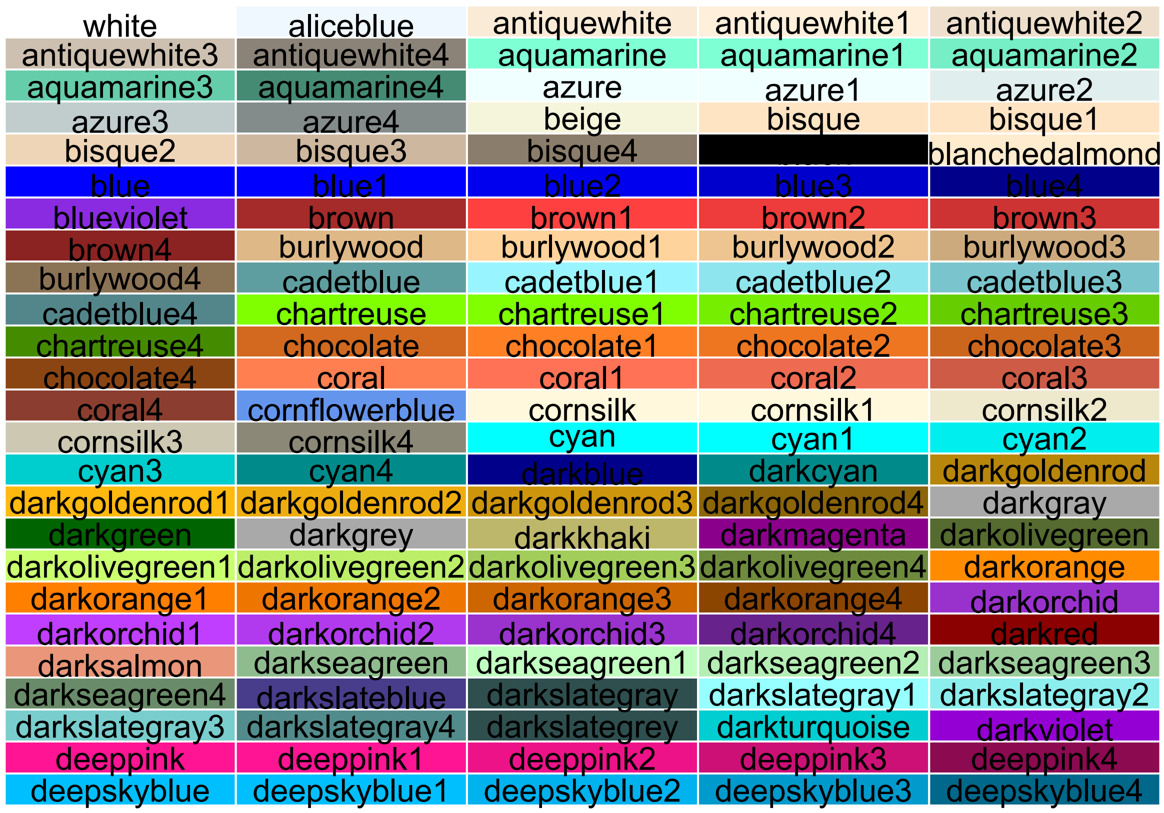


Let's freshen up this graphic a bit. We can set any aesthetic to a fixed setting, e.g. make the dots pink. To do this, we need to pass the aesthetic as a parameter in the `geom_*` function (instead of the `aes()` function). In the next example, we're colouring the dots a bit bigger and pink, and making the smoothed line a dashed black line. Visit Aesthetic specifications for a overview of the available values for each aesthetic.

```
ggplot(mpg) + aes(cty, hwy) +
  geom_jitter(colour = "deeppink", size = 2) +
  geom_smooth(method = "lm", colour = "black", linetype = "dashed")
```

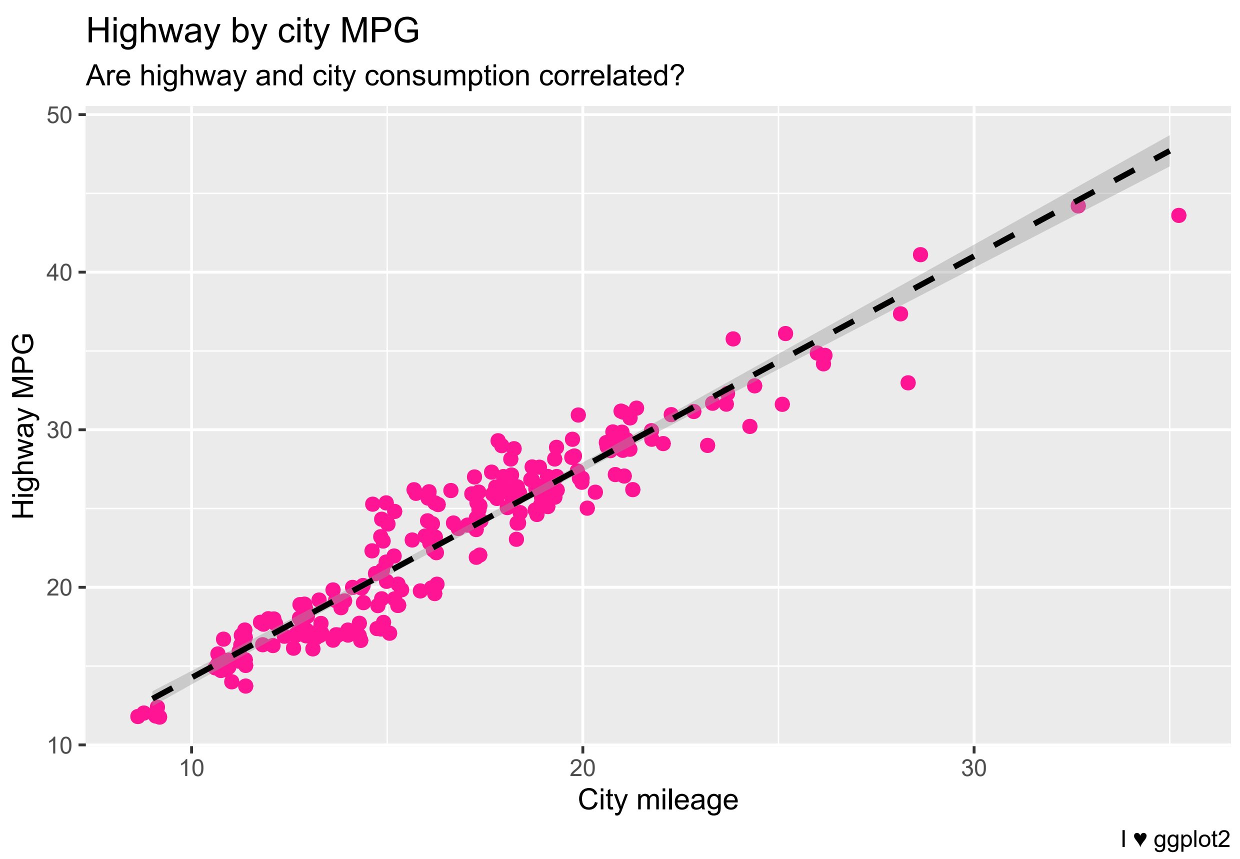


Refer to R graph gallery or the figure below for a list of all available colours in R.



The x- and y-axis labels aren't very useful in the graphic above. To specify labels, use the `labs()` function. Any parameters of the `label()` function are optional.

```
ggplot(mpg) + aes(cty, hwy) +
  geom_jitter(colour = "deeppink", size = 2) +
  geom_smooth(method = "lm", colour = "black", linetype = "dashed") +
  labs(x = "City mileage", y = "Highway MPG",
       title = "Highway by city MPG",
       subtitle = "Are highway and city consumption correlated?",
       caption = "I \U2665 ggplot2")
```

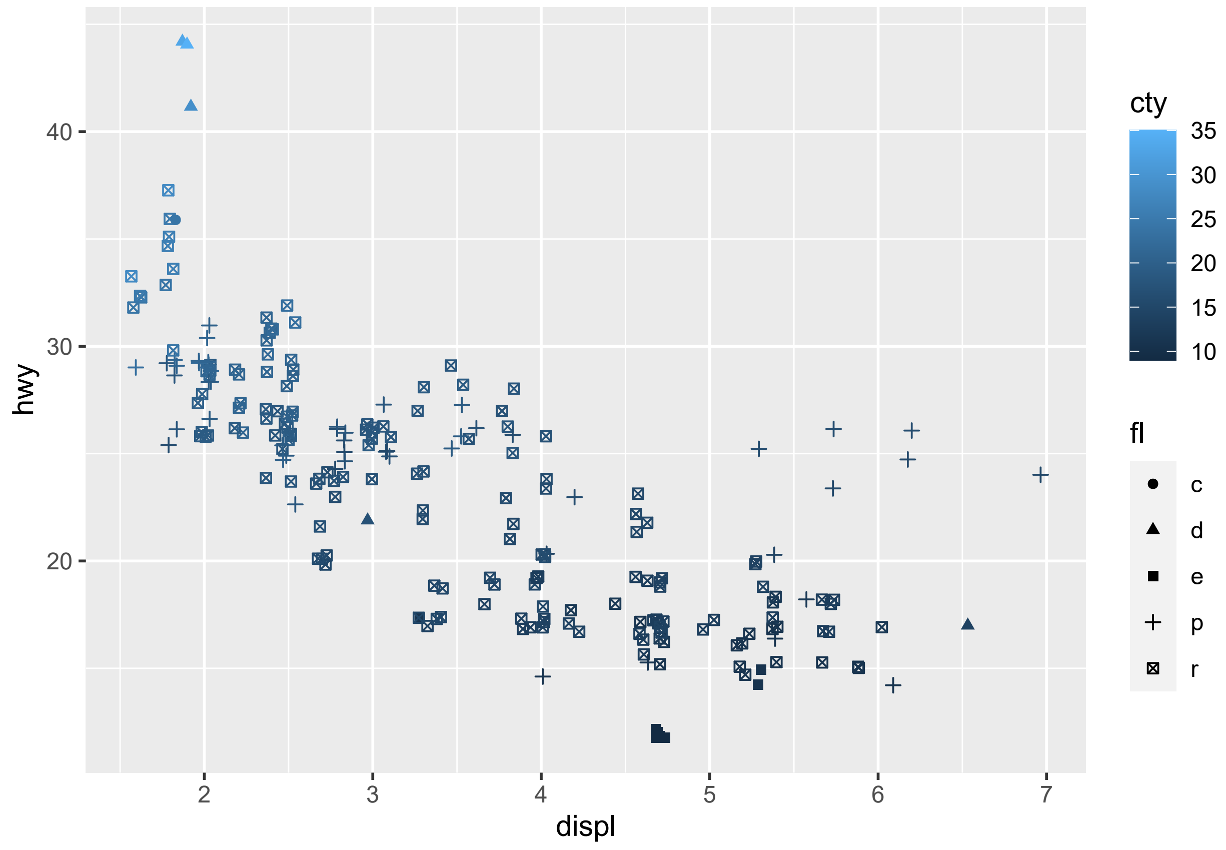

Highway by city MPG
Are highway and city consumption correlated?
Highway MPG
50
40
30
20
10
10
20
30
City mileage
I ♥ ggplot2

Legends

ggplot2 automatically creates legends when needed. In the next example, we create a graphic where we encode four variables: engine displacement and highway mileage in the position (x- and y-axes), city mileage as colour and fuel type as shape.

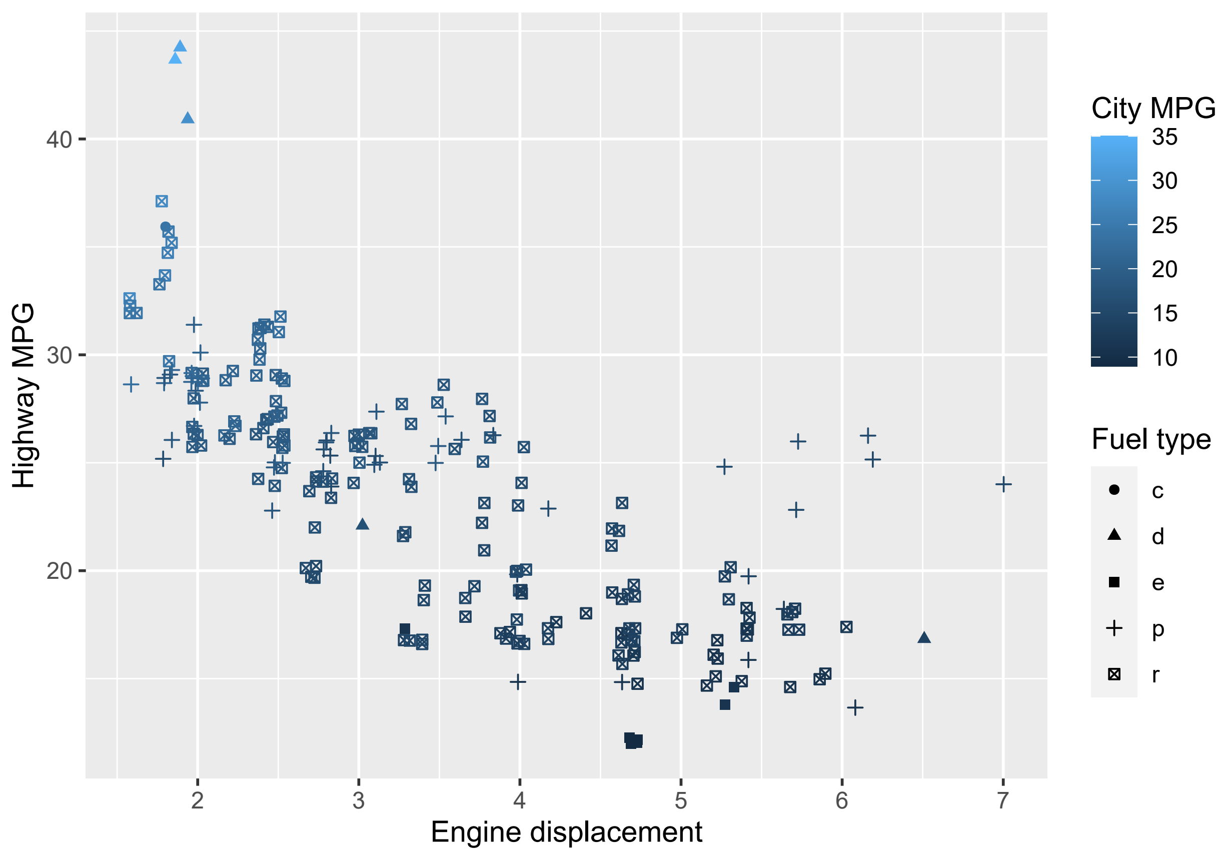
Because *city mileage* is a continuous scale, ggplot creates a colour bar legend. *Fuel type* is discrete, therefore *ggplot2* creates a standard legend.

```
ggplot(mpg) + aes(displ, hwy, colour = cty, shape = fl) + geom_jitter()
```



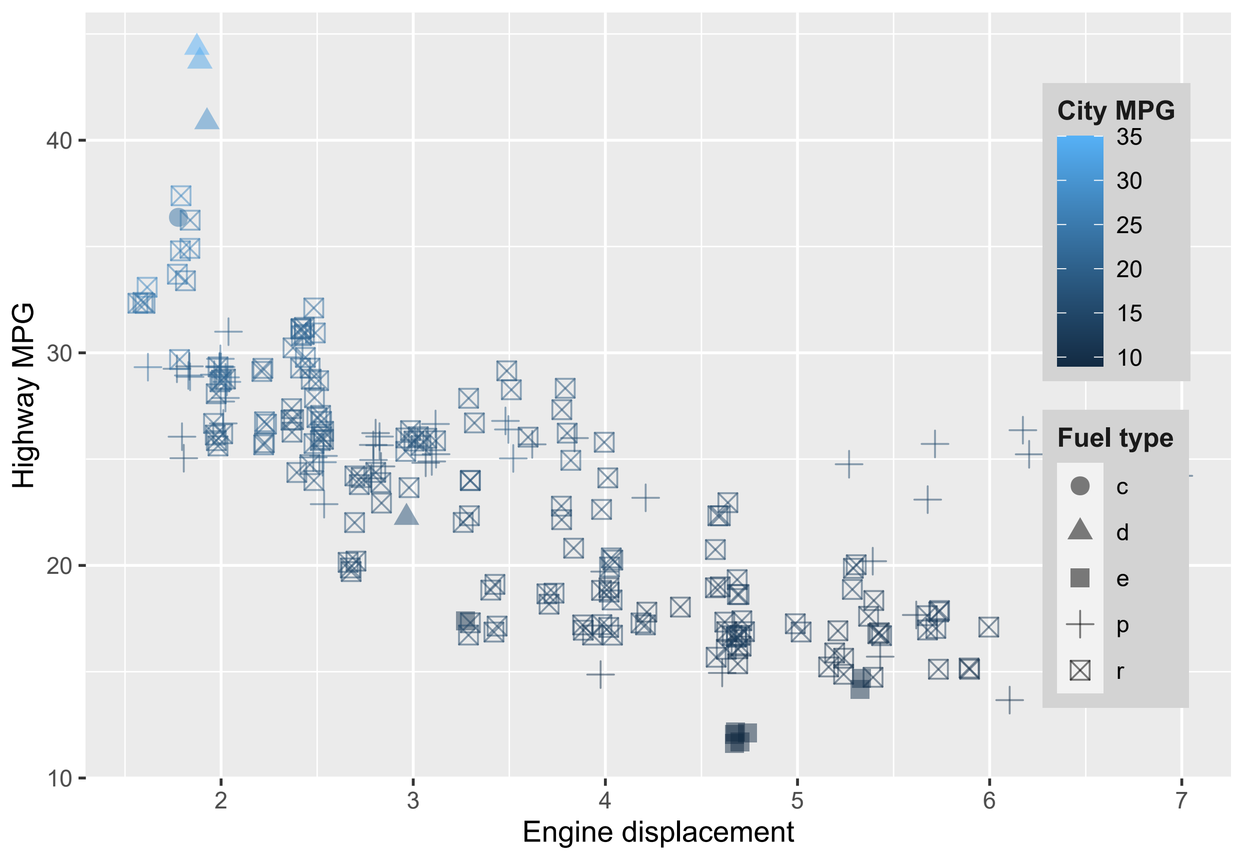
Let's use `labs()` to change the names of all four variables.

```
ggplot(mpg) + aes(displ, hwy, colour = cty, shape = fl) + geom_jitter(aes()) +
  labs(x = "Engine displacement", y = "Highway MPG",
       colour = "City MPG", shape = "Fuel type")
```



Finally, let's style the legend. The function `theme()` is used to control the appearance of any element of the graphic.

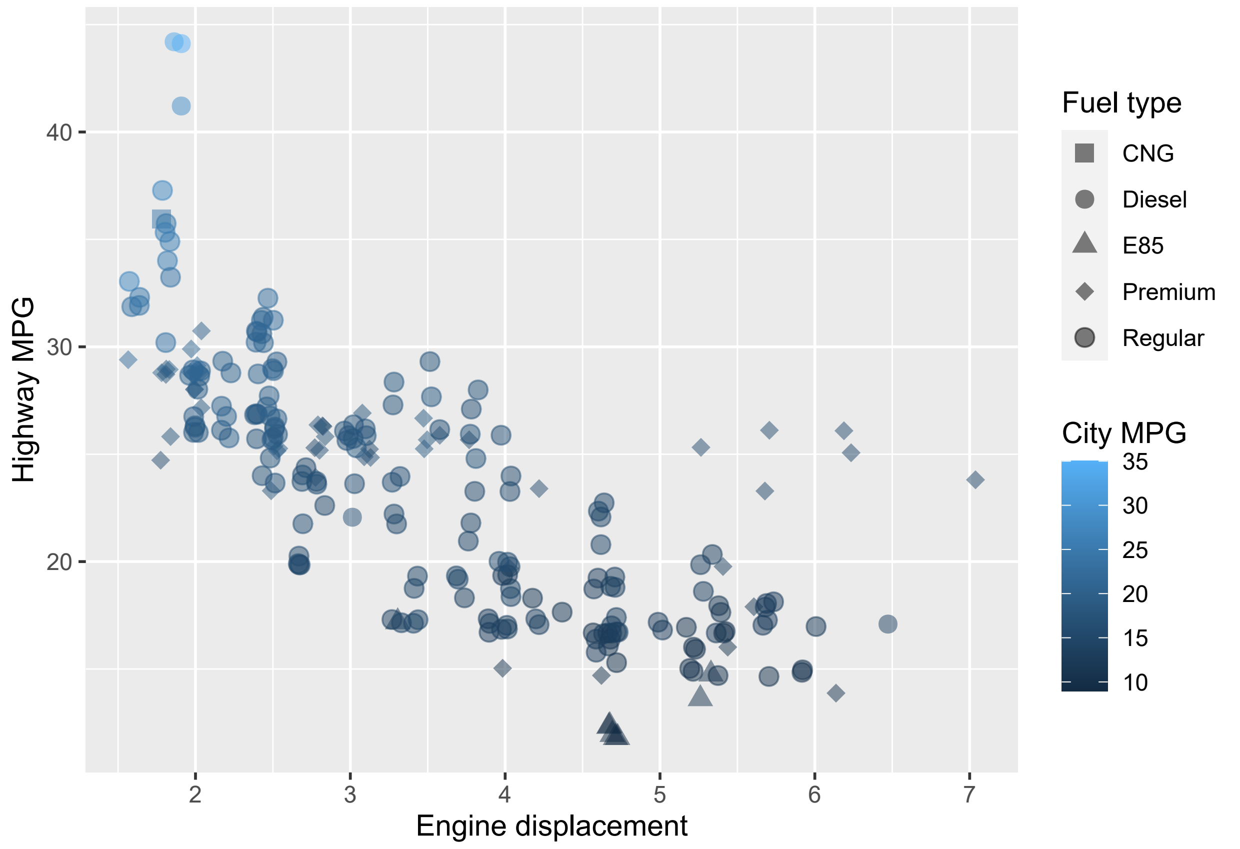
```
ggplot(mpg) + aes(displ, hwy, colour = cty, shape = fl) +
  geom_jitter(size = 3, alpha = 0.5) +
  labs(x = "Engine displacement", y = "Highway MPG",
       colour = "City MPG", shape = "Fuel type") +
  theme(legend.position = c(0.9, 0.5),
        legend.background = element_rect(fill="lightgrey",
                                         size=0.5),
        legend.title = element_text(colour="grey10", size=10,
                                     face="bold"))
```


City MPG
35
30
25
20
15
10
Fuel type
c
d
e
p
r
Highway MPG
40
30
20
10
Engine displacement
2
3
4
5
6
7

Scales

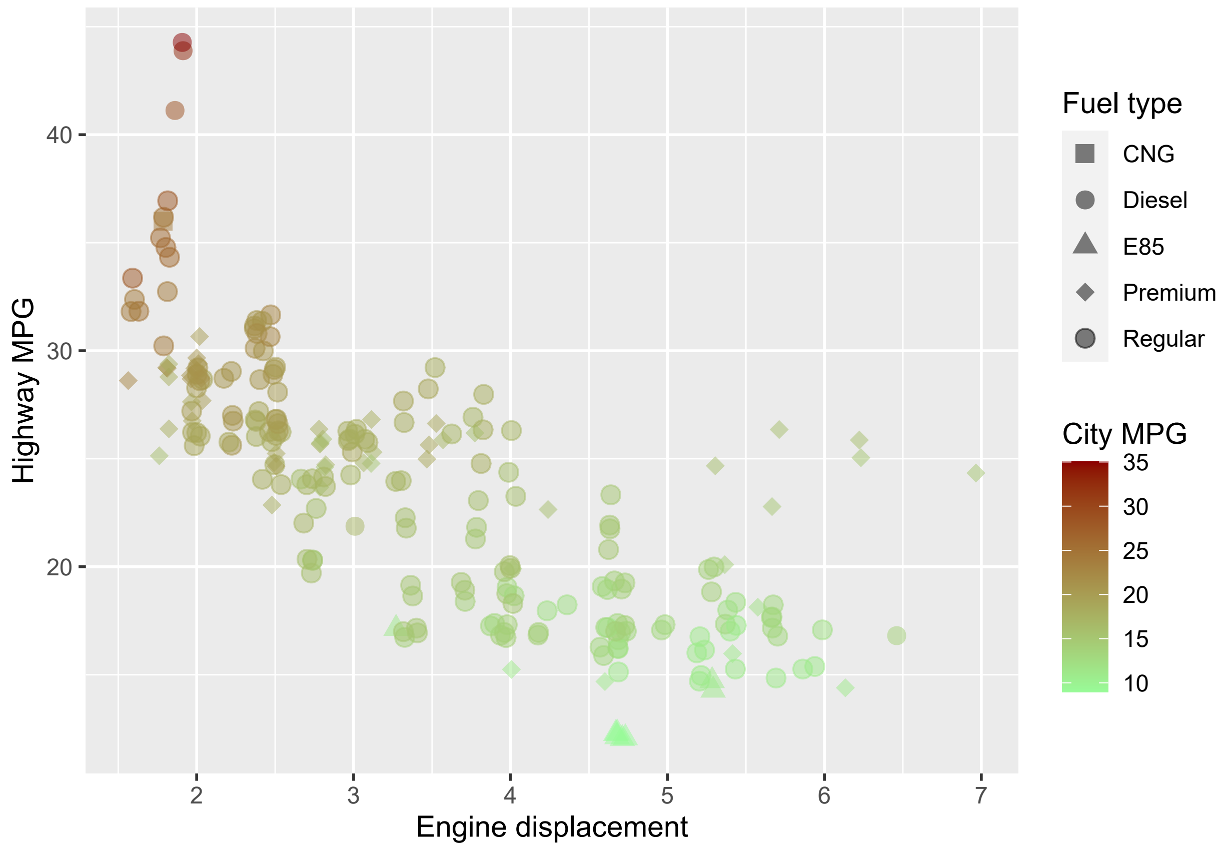
The values for fuel type are not descriptive, let's change them by adding a custom scale. We use the `scale_shape_manual()` function to specify specific shapes (shapes in R are numbered, we use shapes 15 to 19) and the labels for each category. We also make the shapes a bit larger and translucent (`size = 3, alpha = 0.5`).

```
ggplot(mpg) + aes(displ, hwy, colour = cty, shape = fl) +
  geom_jitter(size = 3, alpha = 0.5) +
  labs(x = "Engine displacement", y = "Highway MPG",
       colour = "City MPG", shape = "Fuel type") +
  scale_shape_manual(values = 15:19,
                     labels = c("CNG", "Diesel", "E85", "Premium", "Regular"))
```



Let's also specify a custom scale for the *colour* aesthetic. This time, we use a gradient scale, where we specify the "low" and "high" colour.

```
ggplot(mpg) + aes(displ, hwy, colour = cty, shape = fl) +
  geom_jitter(size = 3, alpha = 0.5) +
  labs(x = "Engine displacement", y = "Highway MPG",
       colour = "City MPG", shape = "Fuel type") +
  scale_shape_manual(values = 15:19,
                     labels = c("CNG", "Diesel", "E85", "Premium", "Regular"))+
  scale_colour_gradient(low = "palegreen", high = "darkred")
```

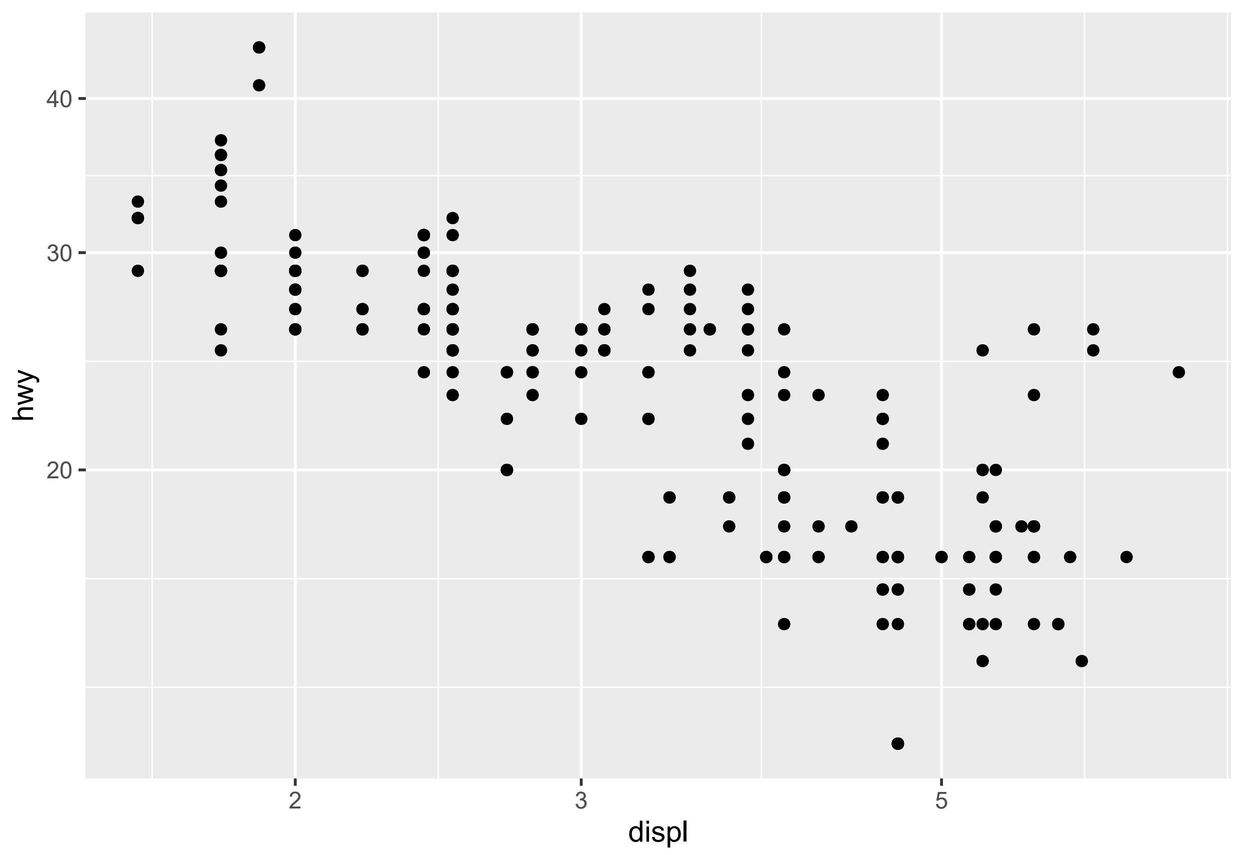


ggplot2 supports many other scales, see [https://ggplot2.tidyverse.org/reference/index.html#section-scales] for an list of available options.

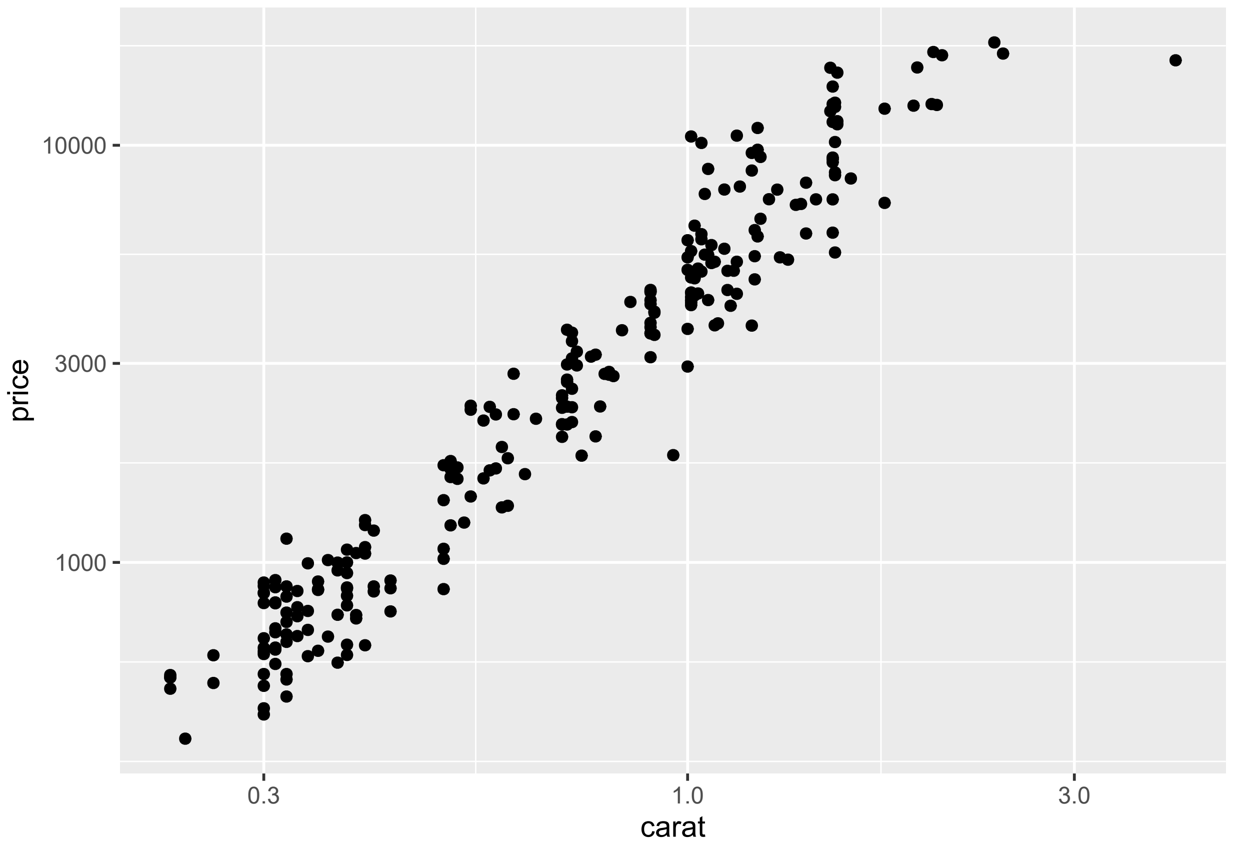
Logarithmic scales

One commonly used scale is the logarithmic scale. Note how we can specify different scales for the x and y-axes.

```
ggplot(mpg) + aes(displ, hwy) + geom_point() +
  scale_x_log10() + scale_y_log10()
```

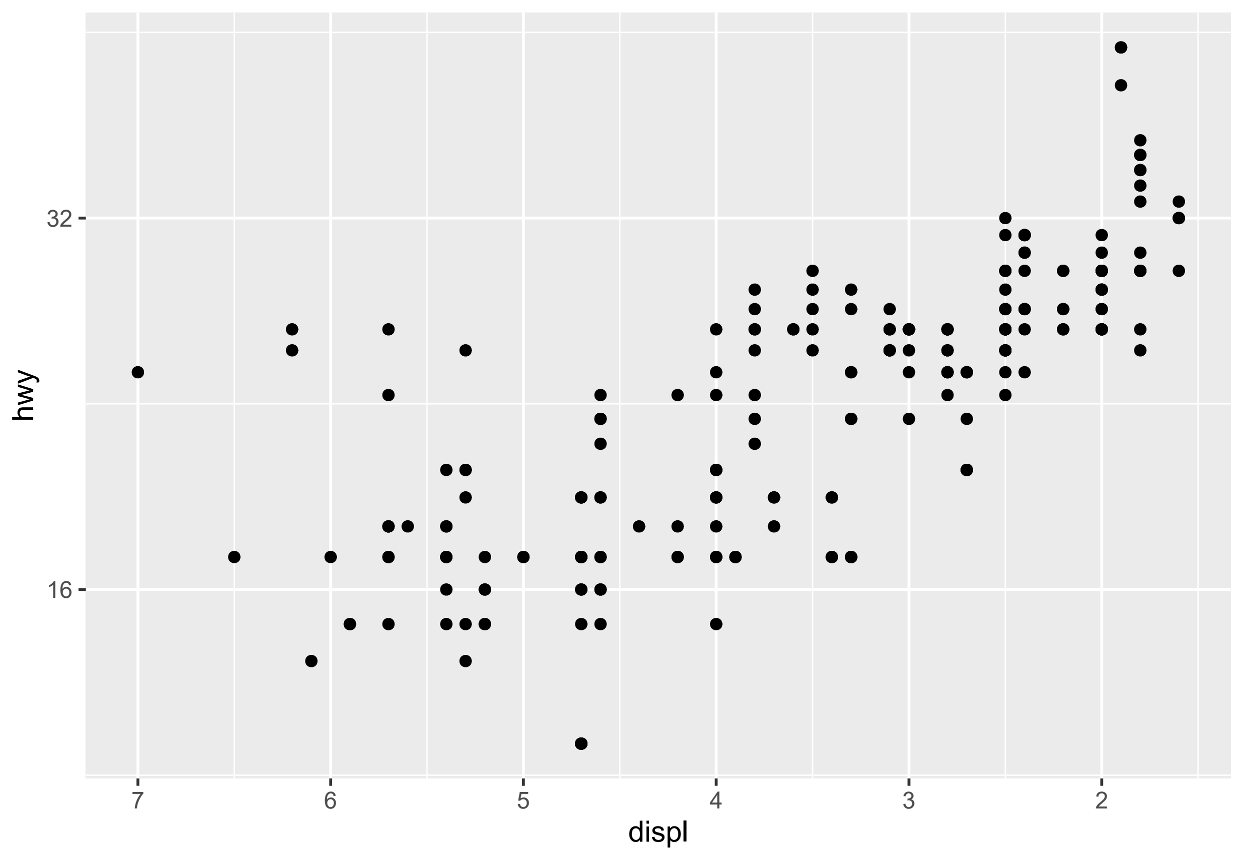


```
ggplot(diamonds[sample(nrow(diamonds), 250),]) + aes(carat, price) + geom_point() +
  scale_x_log10() + scale_y_log10()
```



If we want to use a different logarithm, e. g. base2, we need to use `scale_*_continuous` and specify the transformation:

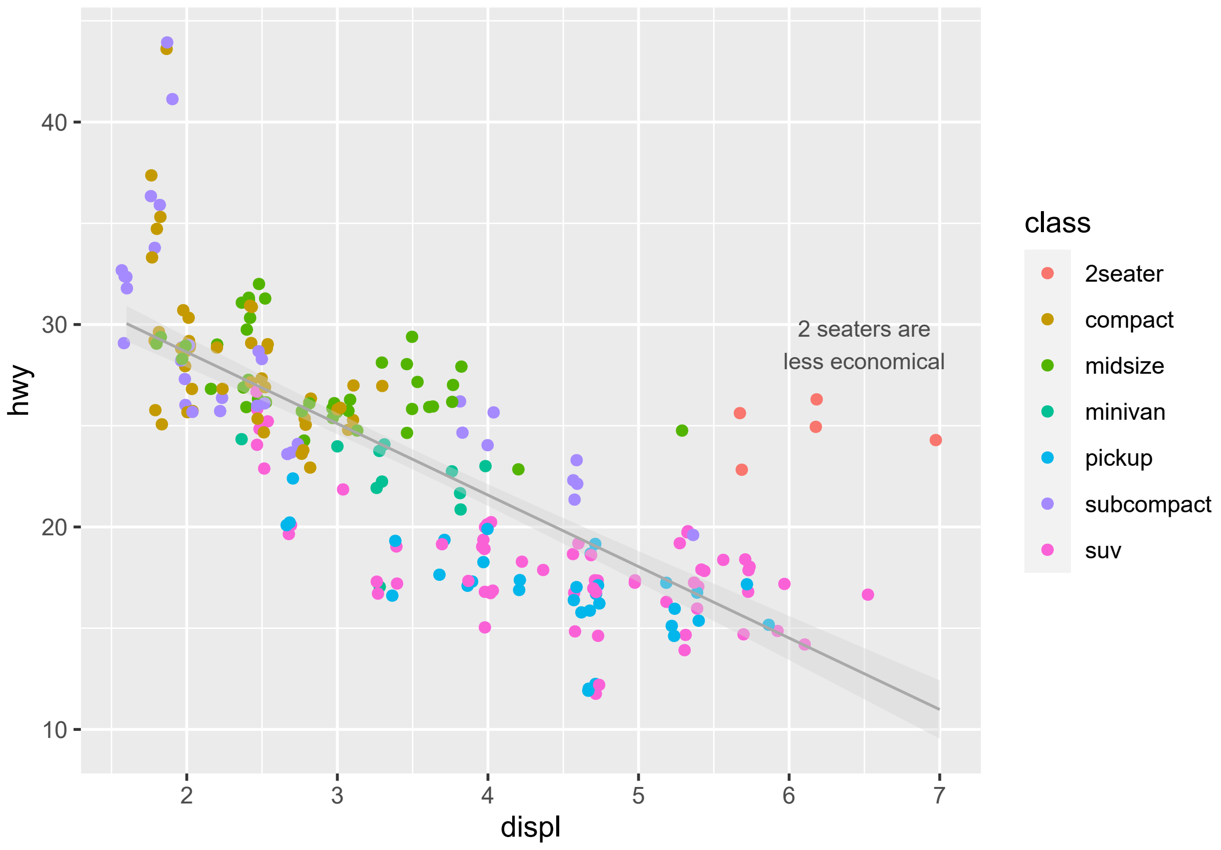
```
ggplot(mpg) + aes(displ, hwy) + geom_point() +
  scale_x_continuous(trans = "reverse") +
  scale_y_continuous(trans = "log2")
```



Annotations

Sometimes we need to add a label to a chart to explain something. *ggplot* provides the `annotate()` function to add an annotation layer.

```
ggplot(mpg) + aes(displ, hwy, colour = class) +
  geom_jitter() +
  geom_smooth(method = "lm", formula = y ~ x, colour = "darkgrey",
              size = 0.5, fill = "lightgrey") +
  annotate(geom = "text", x= 6.5, y = 29,
           label = "2 seaters are\nless economical",
           colour = "grey30", size = 3)
```

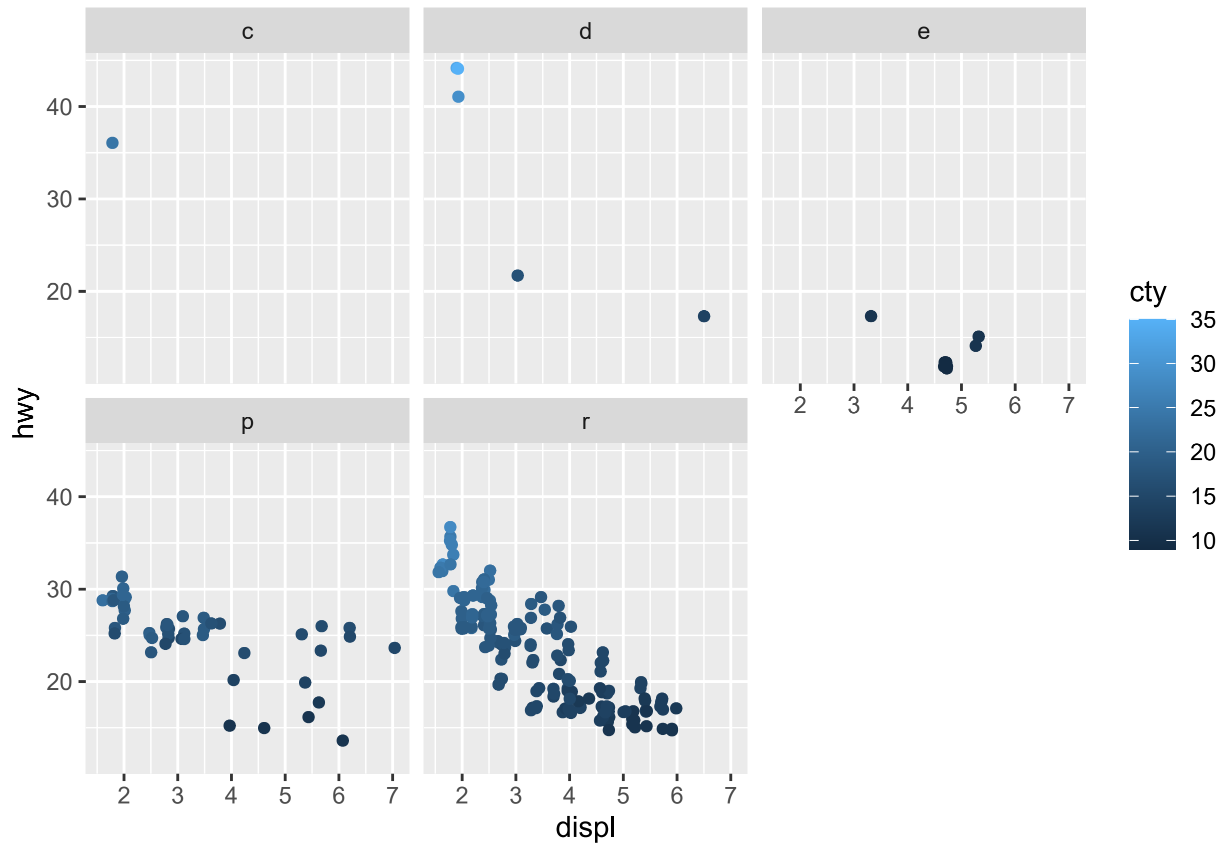


hwy
40
30
20
10
2
3
4
5
6
7
displ
2 seaters are
less economical
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

Facets

While *ggplot2* supports several aesthetics, encoding to many variables in one graphic will produce a confusing chart. Sometimes, it is easier to produce several graphics instead. *ggplot2* makes it easy to create a series of charts with faceting. We just need to add one of the facet functions (`facet_wrap()` or `facet_grid()`) and pass the respective variable wrapped in a `vars()` call.

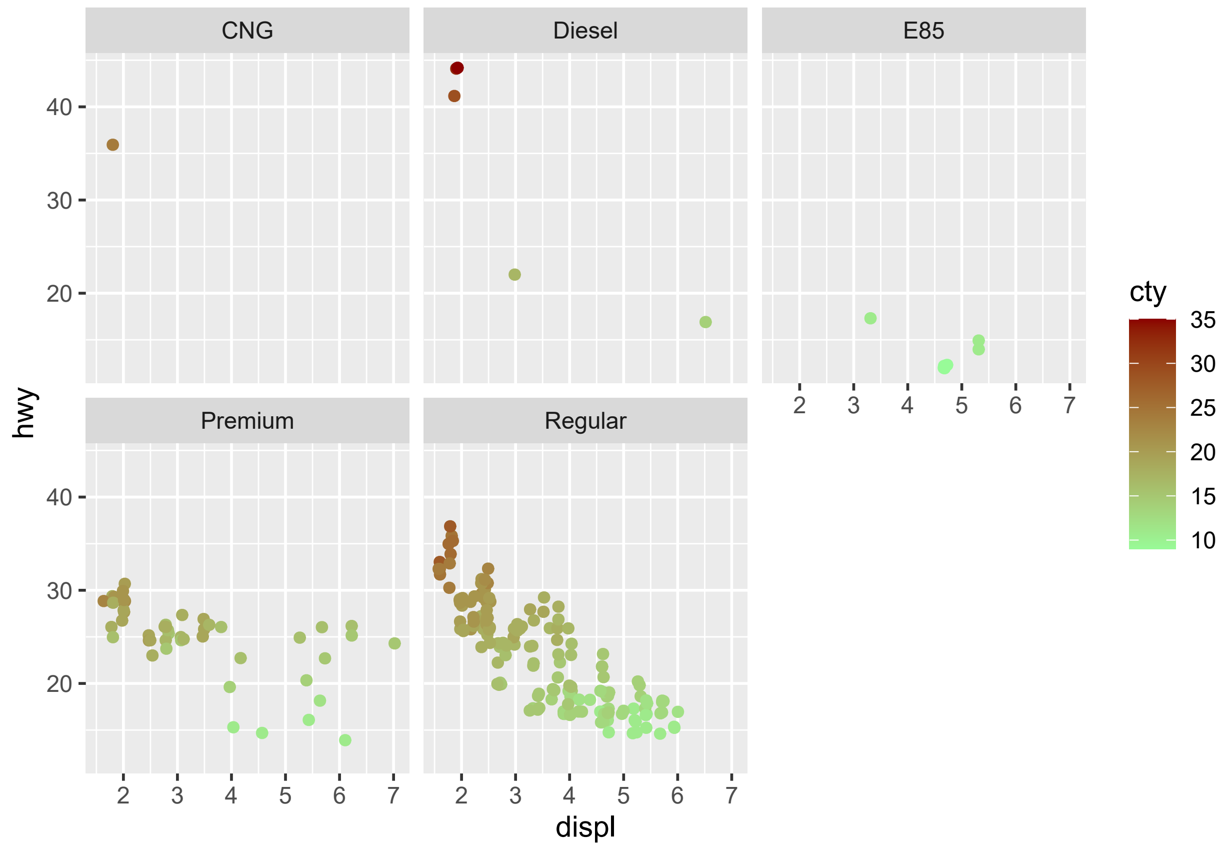
```
ggplot(mpg) + aes(displ, hwy, colour = cty) + geom_jitter() +
  facet_wrap(vars(fl))
```



But now, the labels for each chart are meaningless again. To define custom labels, we first need to define a vector with the labels. Then we set the *name* of each element to the value in the data.

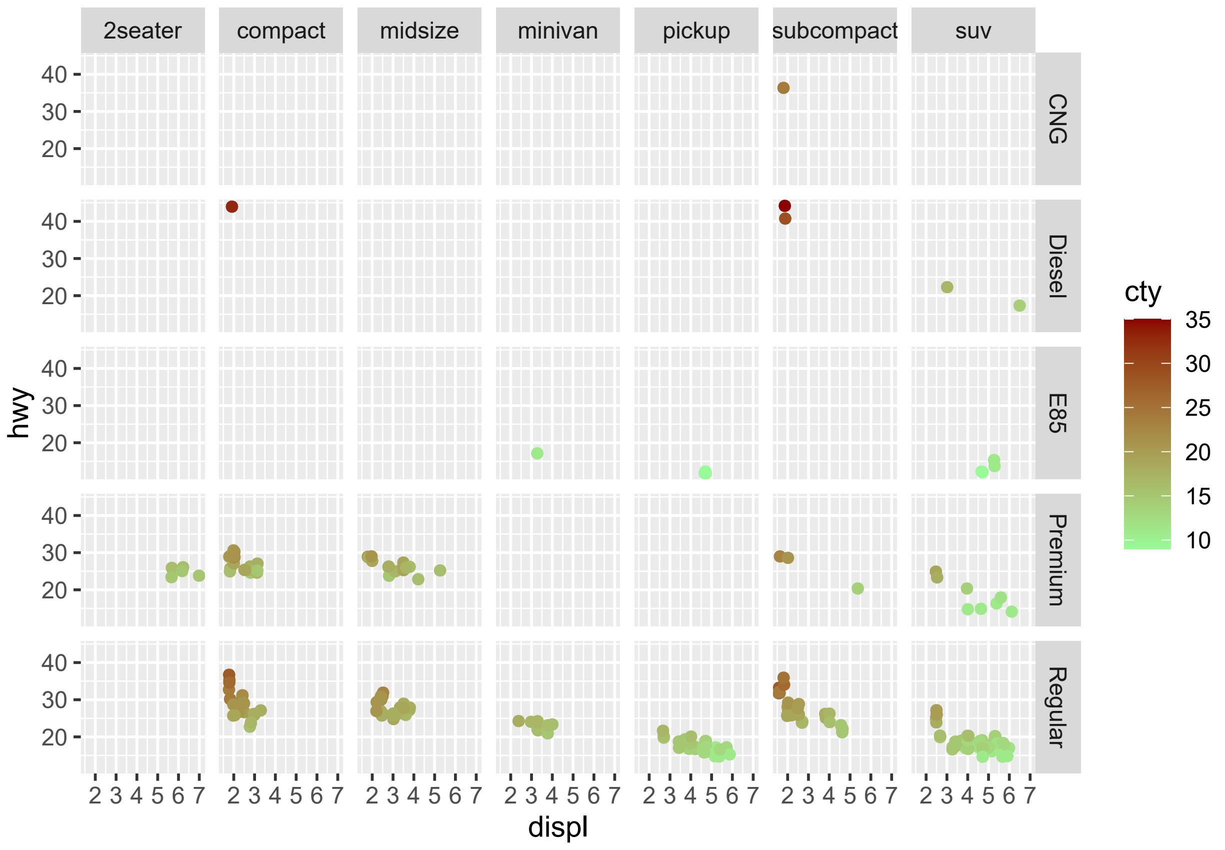
And of course, we can also still add custom scales.

```
fl.labels <- c("CNG", "Diesel", "E85", "Premium", "Regular", "x")
names(fl.labels) <- c("c", "d", "e", "p", "r", "x")

ggplot(mpg) + aes(displ, hwy, colour = cty) + geom_jitter() +
  scale_colour_gradient(low = "palegreen", high = "darkred") +
  facet_wrap(vars(fl), labeller = labeller(fl = fl.labels ))
```



We can also use two variables in a facet grid.

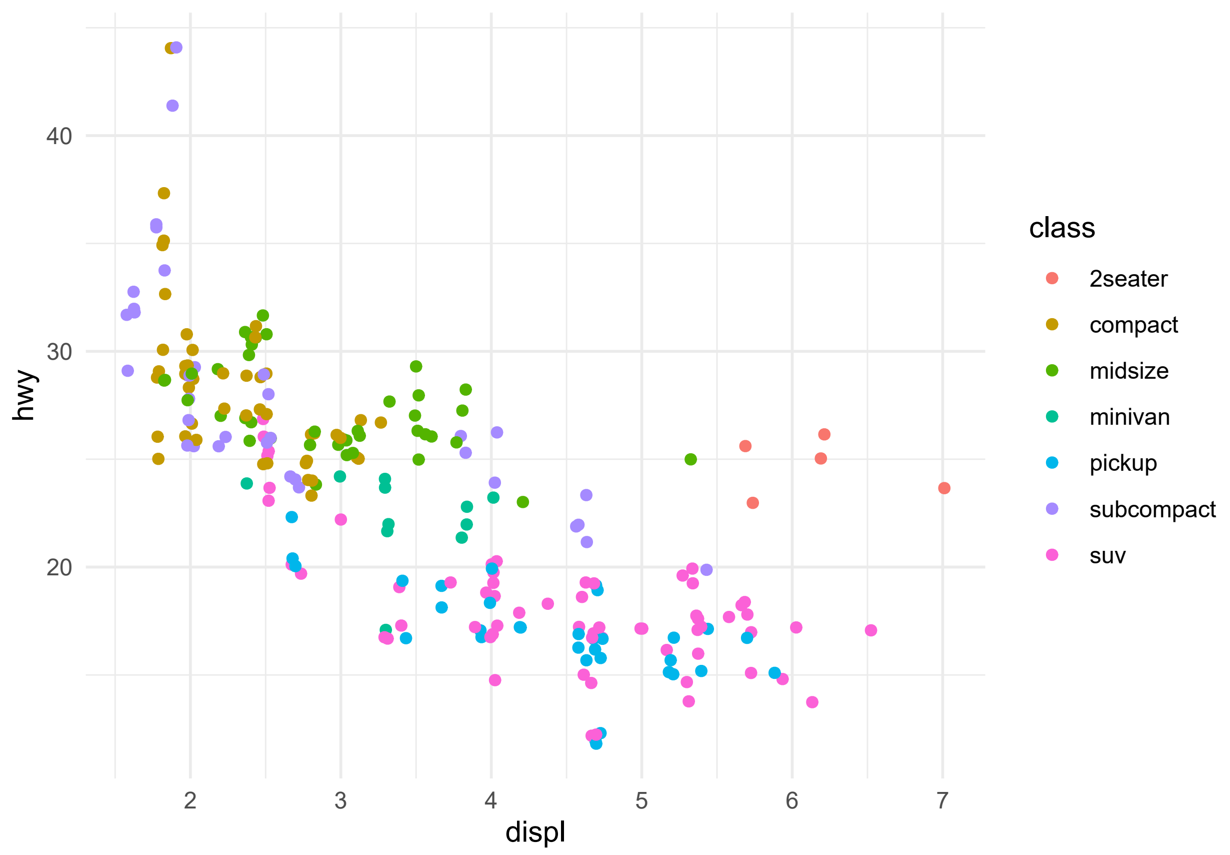
```r
fl.labels <- c("CNG", "Diesel", "E85", "Premium", "Regular")
names(fl.labels) <- c("c", "d", "e", "p", "r")

ggplot(mpg) + aes(displ, hwy, colour = cty) + geom_jitter() +
  scale_colour_gradient(low = "palegreen", high = "darkred") +
  facet_grid(fl ~ class, labeller = labeller(fl = fl.labels ))
```



Themes

Themes are a way to change the look of a graphics completely.

```
ggplot(mpg) + aes(displ, hwy, colour = class) + geom_jitter() +
  theme_minimal()
```



ggplot2 comes with a some predefined themes. More themes are defined in the package `ggthemes`. You'll need to install it first:

```
install.packages("ggthemes")
```

And then we need to load it:

```
library(ggthemes)
```

ggthemes provides over 100 themes, palettes and scales.

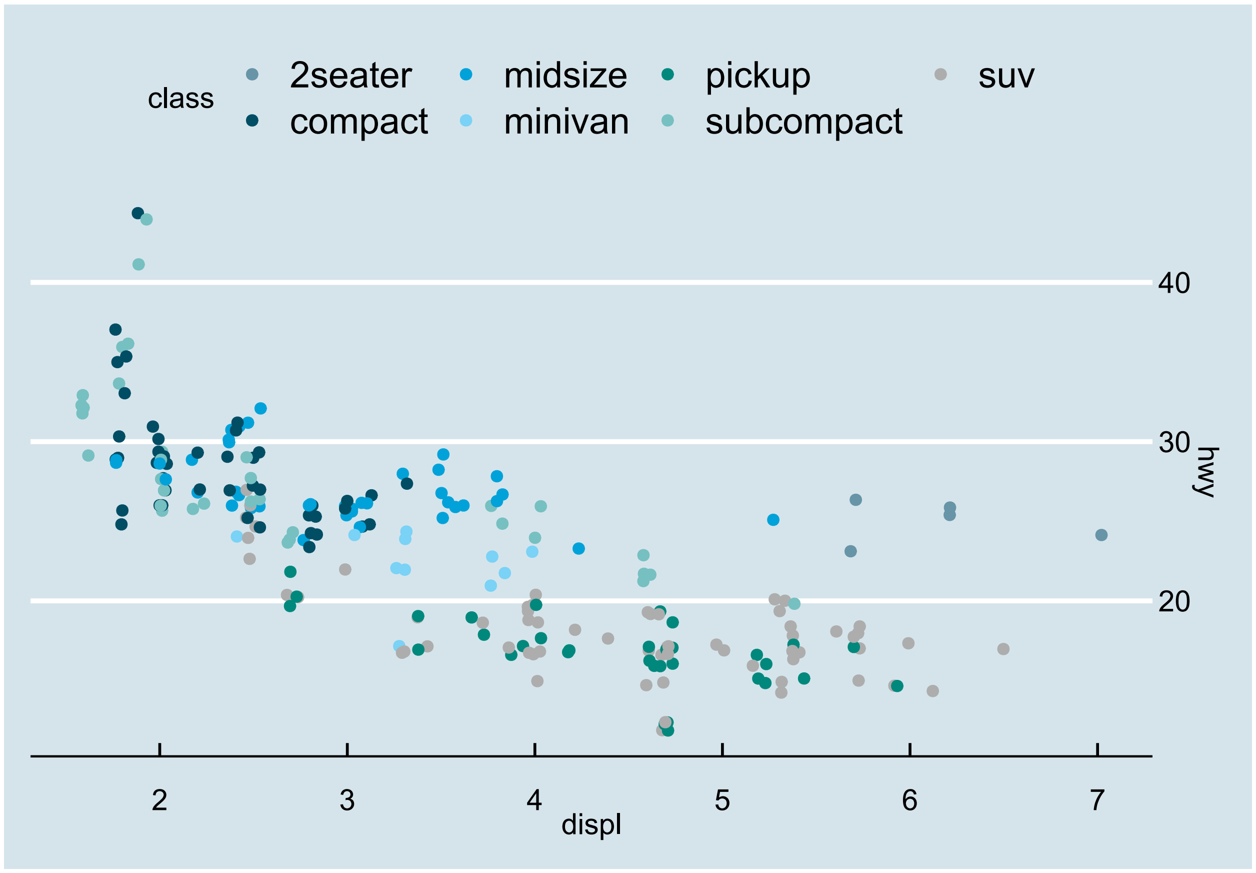
```
ls("package:ggthemes")
```

```
##   [1] "bank_slopes"                 "calc_pal"
##   [3] "calc_shape_pal"              "canva_pal"
##   [5] "canva_palettes"              "circlefill_shape_pal"
##   [7] "cleveland_shape_pal"         "colorblind_pal"
##   [9] "economist_pal"               "excel_new_pal"
##  [11] "excel_pal"                   "extended_range_breaks"
##  [13] "extended_range_breaks_"      "few_pal"
##  [15] "few_shape_pal"               "fivethirtyeight_pal"
##  [17] "gdocs_pal"                   "geom_rangeframe"
##  [19] "geom_tufteboxplot"           "GeomRangeFrame"
##  [21] "GeomTufteboxplot"            "ggthemes_data"
##  [23] "hc_pal"                      "palette_pander"
##  [25] "ptol_pal"                        "scale_color_calc"
##  [27] "scale_color_canva"               "scale_color_colorblind"
##  [29] "scale_color_continuous_tableau" "scale_color_economist"
##  [31] "scale_color_excel"               "scale_color_excel_new"
##  [33] "scale_color_few"                 "scale_color_fivethirtyeight"
##  [35] "scale_color_gdocs"               "scale_color_gradient_tableau"
##  [37] "scale_color_gradient2_tableau"  "scale_color_hc"
##  [39] "scale_color_pander"              "scale_color_ptol"
##  [41] "scale_color_solarized"           "scale_color_stata"
##  [43] "scale_color_tableau"             "scale_color_wsj"
##  [45] "scale_colour_calc"               "scale_colour_canva"
##  [47] "scale_colour_colorblind"         "scale_colour_economist"
##  [49] "scale_colour_excel"              "scale_colour_excel_new"
##  [51] "scale_colour_few"                "scale_colour_fivethirtyeight"
##  [53] "scale_colour_gdocs"              "scale_colour_gradient_tableau"
##  [55] "scale_colour_gradient2_tableau" "scale_colour_hc"
##  [57] "scale_colour_pander"             "scale_colour_ptol"
##  [59] "scale_colour_solarized"          "scale_colour_stata"
##  [61] "scale_colour_tableau"            "scale_colour_wsj"
##  [63] "scale_fill_calc"                 "scale_fill_canva"
##  [65] "scale_fill_colorblind"           "scale_fill_continuous_tableau"
##  [67] "scale_fill_economist"            "scale_fill_excel"
##  [69] "scale_fill_excel_new"            "scale_fill_few"
##  [71] "scale_fill_fivethirtyeight"      "scale_fill_gdocs"
##  [73] "scale_fill_gradient_tableau"     "scale_fill_gradient2_tableau"
##  [75] "scale_fill_hc"                   "scale_fill_pander"
##  [77] "scale_fill_ptol"                 "scale_fill_solarized"
##  [79] "scale_fill_stata"                "scale_fill_tableau"
##  [81] "scale_fill_wsj"                  "scale_linetype_stata"
##  [83] "scale_shape_calc"                "scale_shape_circlefill"
##  [85] "scale_shape_cleveland"           "scale_shape_few"
##  [87] "scale_shape_stata"               "scale_shape_tableau"
##  [89] "scale_shape_tremmel"             "show_linetypes"
##  [91] "show_shapes"                     "smart_digits"
##  [93] "smart_digits_format"             "solarized_pal"
##  [95] "stat_fivenumber"                 "stata_linetype_pal"
##  [97] "stata_pal"                       "stata_shape_pal"
##  [99] "StatFivenumber"                  "tableau_color_pal"
## [101] "tableau_div_gradient_pal"        "tableau_gradient_pal"
## [103] "tableau_seq_gradient_pal"        "tableau_shape_pal"
## [105] "theme_base"                      "theme_calc"
## [107] "theme_clean"                     "theme_economist"
## [109] "theme_economist_white"           "theme_excel"
## [111] "theme_excel_new"                 "theme_few"
## [113] "theme_fivethirtyeight"           "theme_foundation"
## [115] "theme_gdocs"                     "theme_hc"
## [117] "theme_igray"                     "theme_map"
## [119] "theme_pander"                    "theme_par"
## [121] "theme_solarized"                 "theme_solarized_2"
## [123] "theme_solid"                     "theme_stata"
## [125] "theme_tufte"                     "theme_wsj"
## [127] "tremmel_shape_pal"               "wsj_pal"
```

Let’s see a few examples.

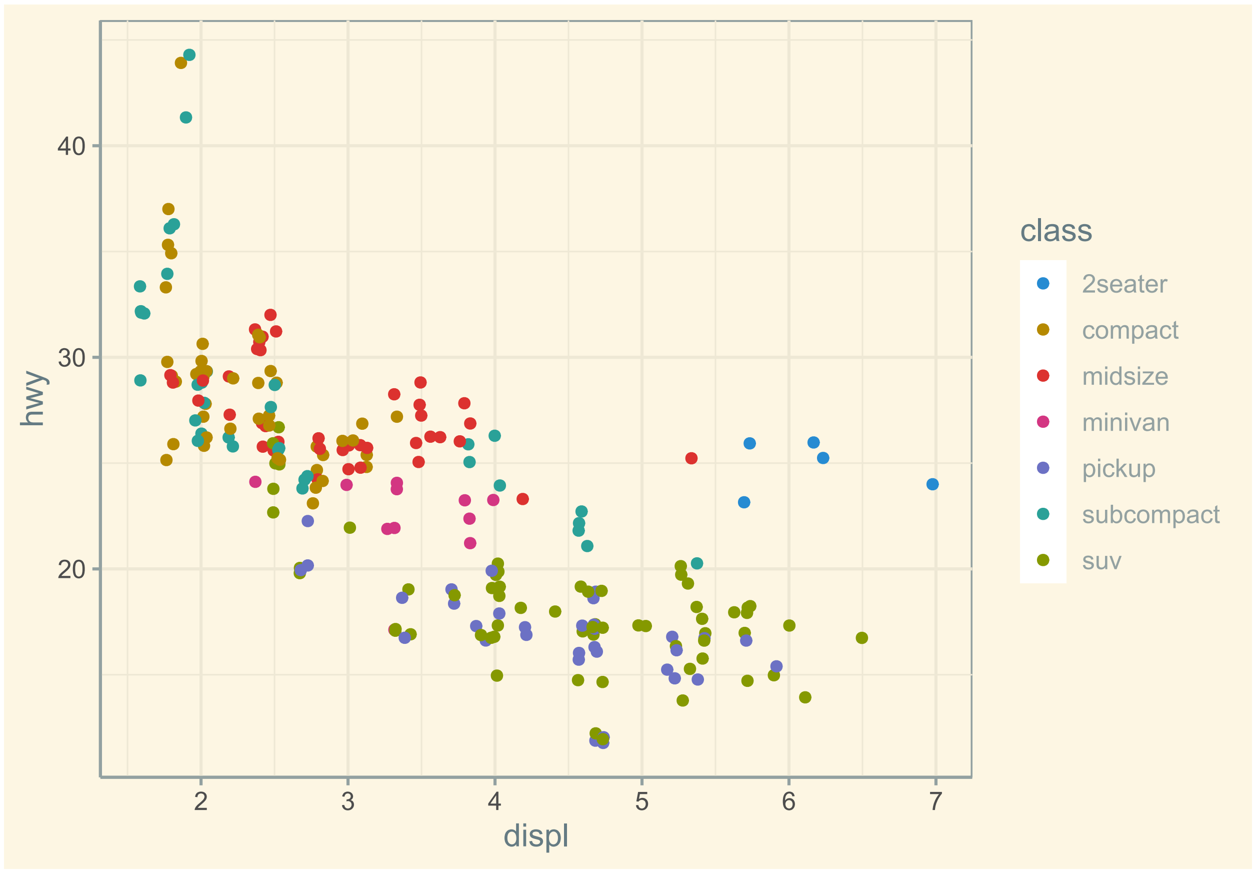
Economist

```
base.plot <- ggplot(mpg) + aes(displ, hwy, colour = class) + geom_jitter()

base.plot +
  theme_economist() + scale_colour_economist() +
  scale_y_continuous(position = "right")
```



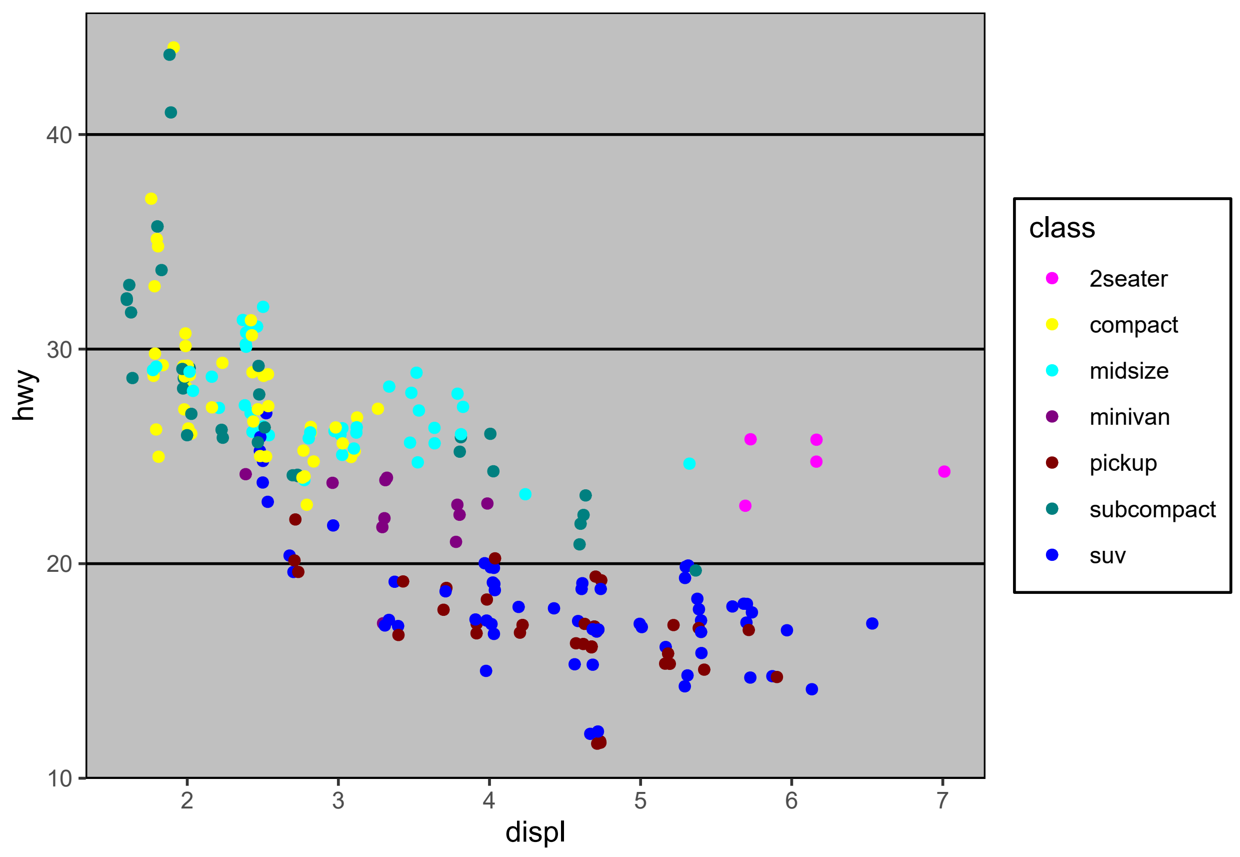
Solarized light

```
base.plot + theme_solarized() + scale_colour_solarized("blue")
```



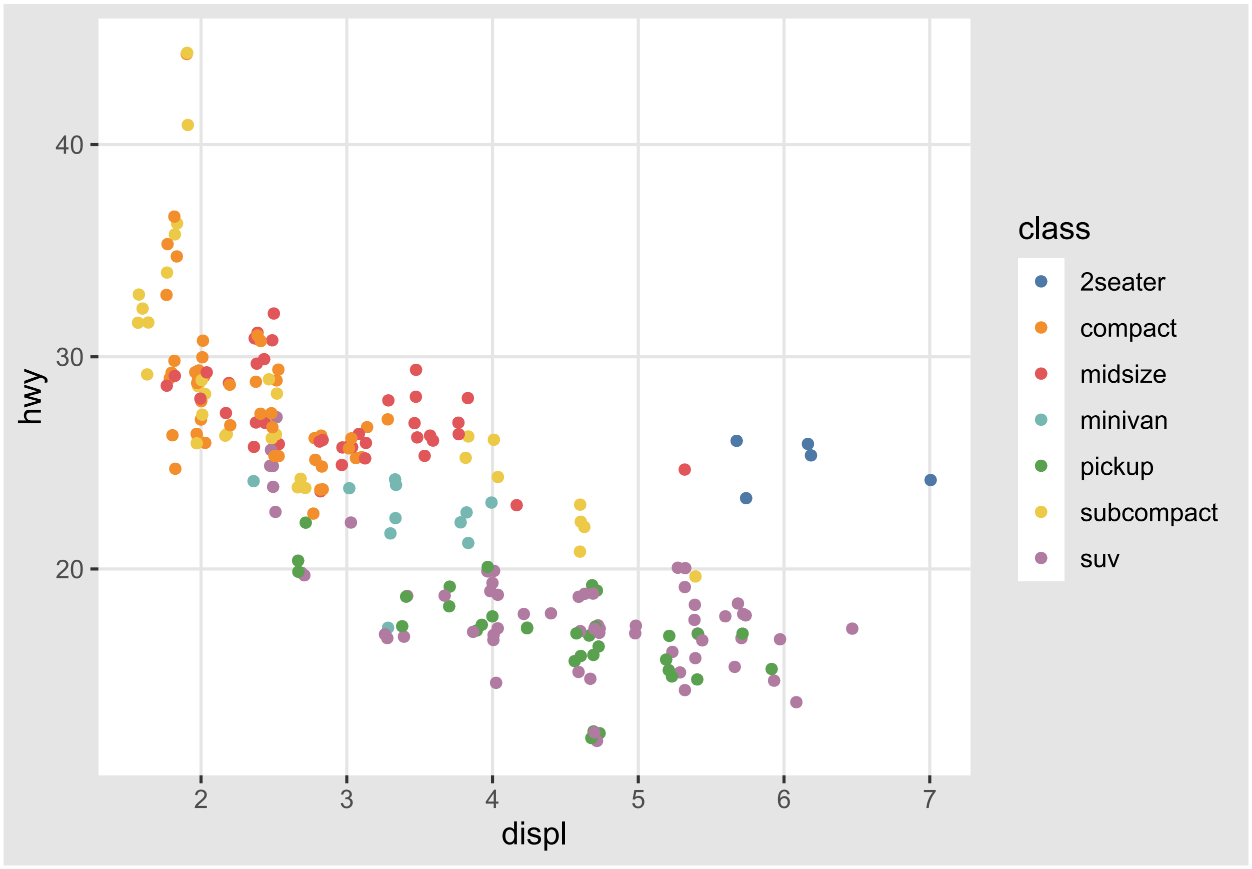
Excel 2003

```
base.plot + theme_excel() + scale_colour_excel()
```



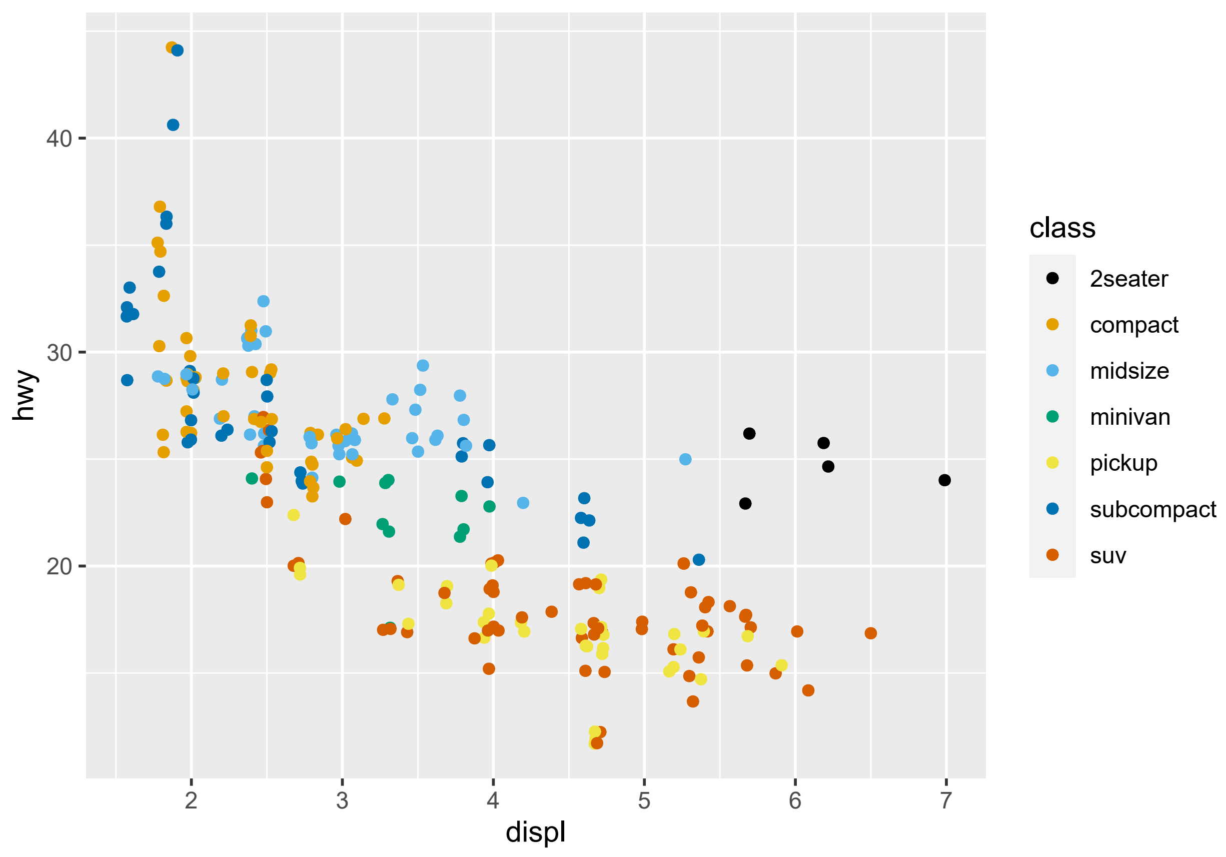
Tableau

```
base.plot + theme_igray() + scale_colour_tableau()
```



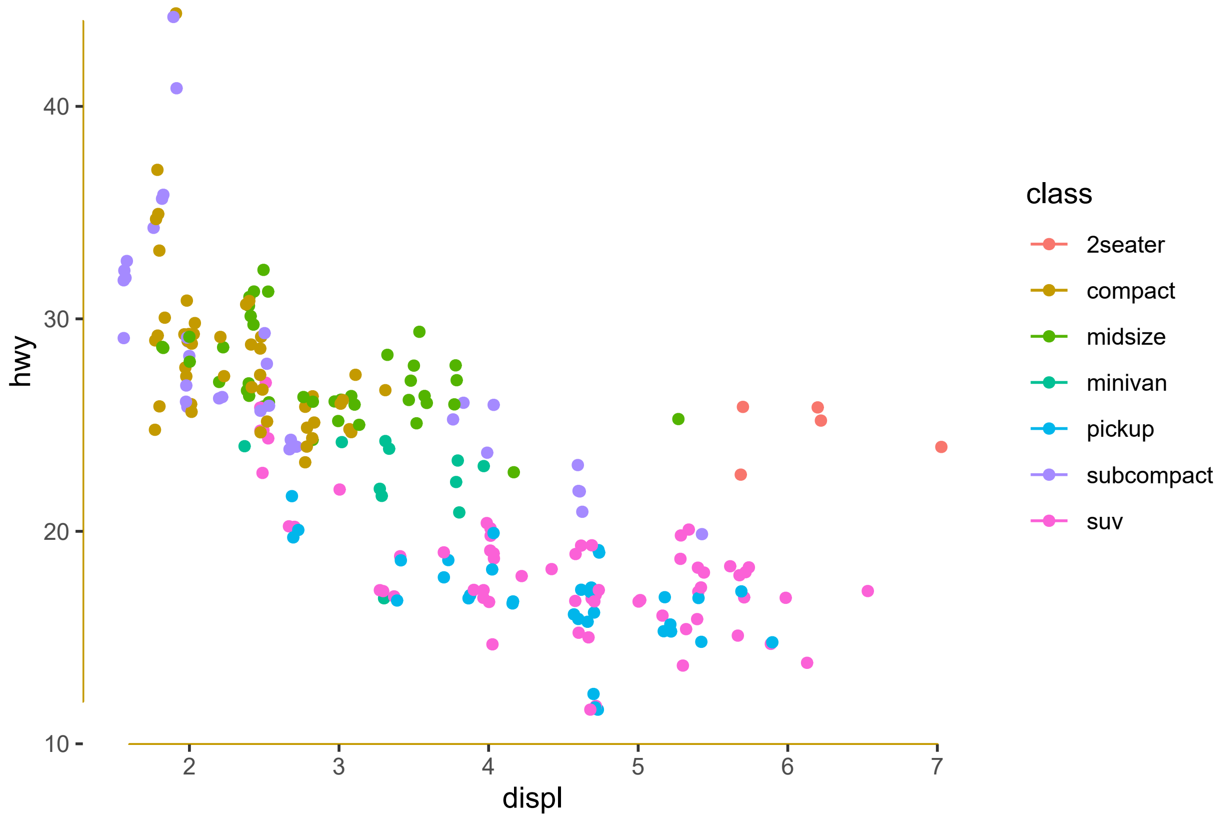
Colour blind scales

```
base.plot + scale_colour_colorblind()
```



Tufte

```
base.plot + geom_rangeframe() + theme_tufte(base_family = "sans")
```

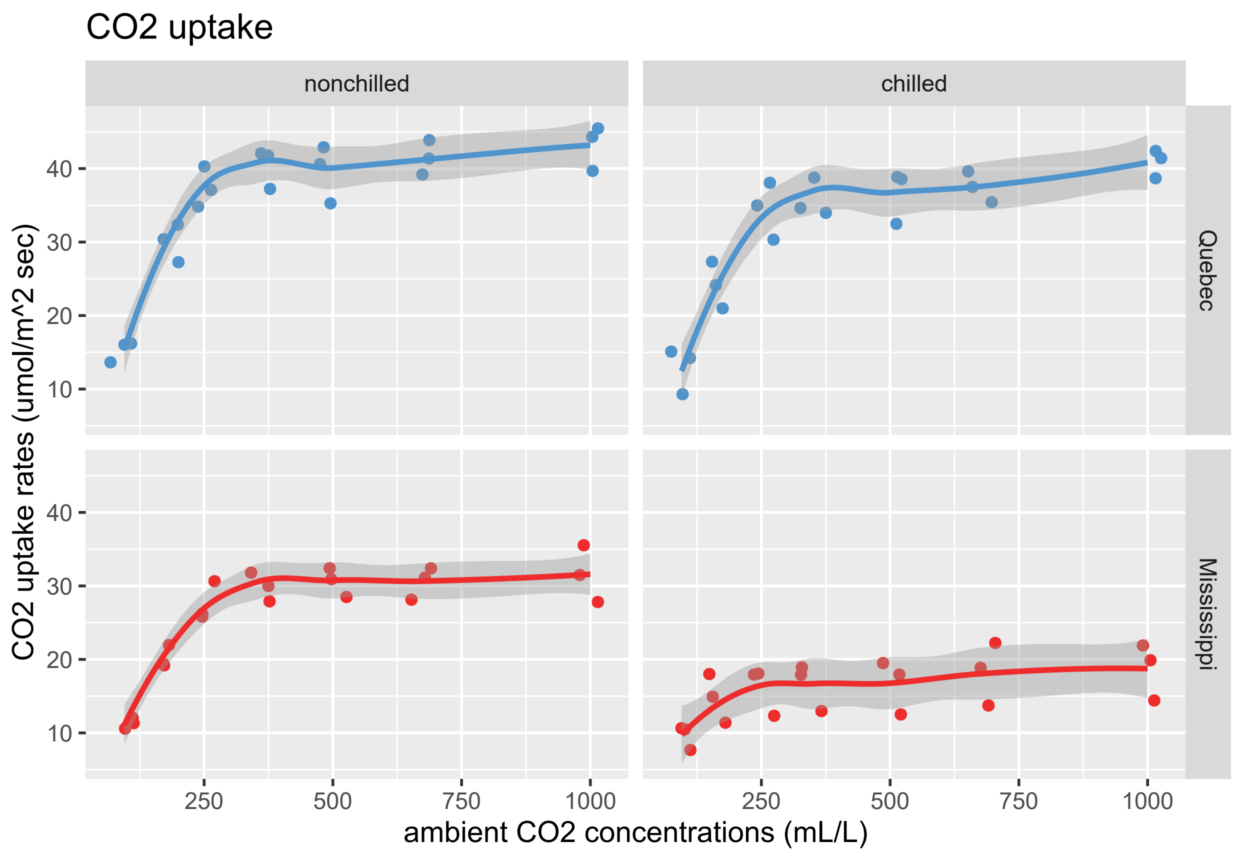

hwy
40
30
20
10
2
3
4
5
6
7
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

Exercises

CO2 uptake

Use the built-in data set `CO2` for the following exercises.

1. Create a scatter plot showing the CO2 uptake by ambient CO2 concentrations. Should you use jitter or not?
2. Add a dotted trend line using linear regression("lm") in your favourite colour. Change the dots to sky blue large triangles.
3. Encode the origin of the plant in the colour of the triangles. Make sure you still have only **one** trend line.
4. Label the chart, axes and legend meaningfully.
5. Use two trend lines (one for each origin). Colour plants from Quebec blue and plants from Mississippi red.
6. Create the chart above for each Plant using `facet_wrap()`.
7. Create the following chart. Tip: You can hide the legend using the `theme()` function and its `legend.position` parameter.



Mammals sleep

Use the `msleep` data set for the following exercises.

1. Investigate if there is a correlation between body weight and brain weight. You might need to use logarithmic scales.

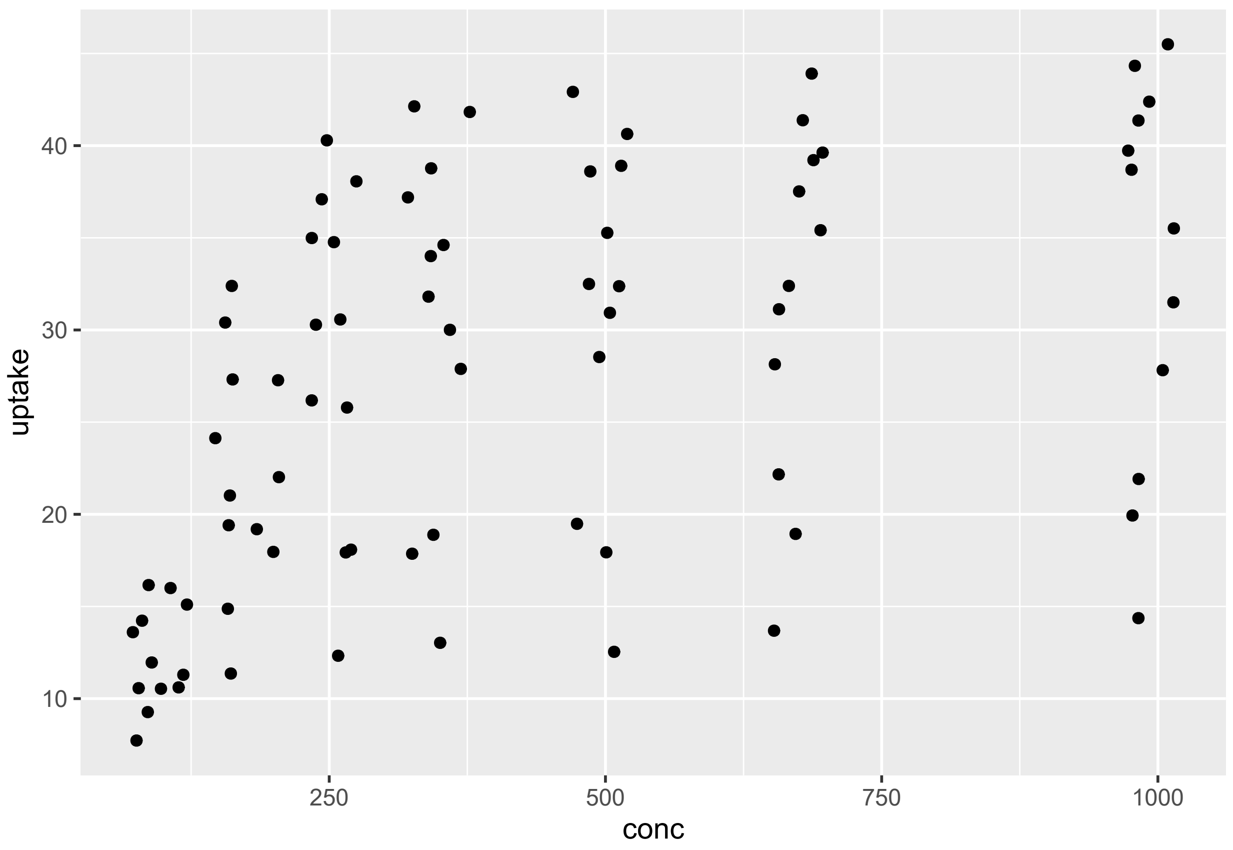
2. Investigate if the food has an influence on body/brain weight.
3. Does the body weight have an influence on the total sleep? What else has an influence on the sleep behaviour?

Solutions

CO2 uptake

1. Create a scatter plot showing the CO2 uptake by ambient CO2 concentrations. Should you use jitter or not?

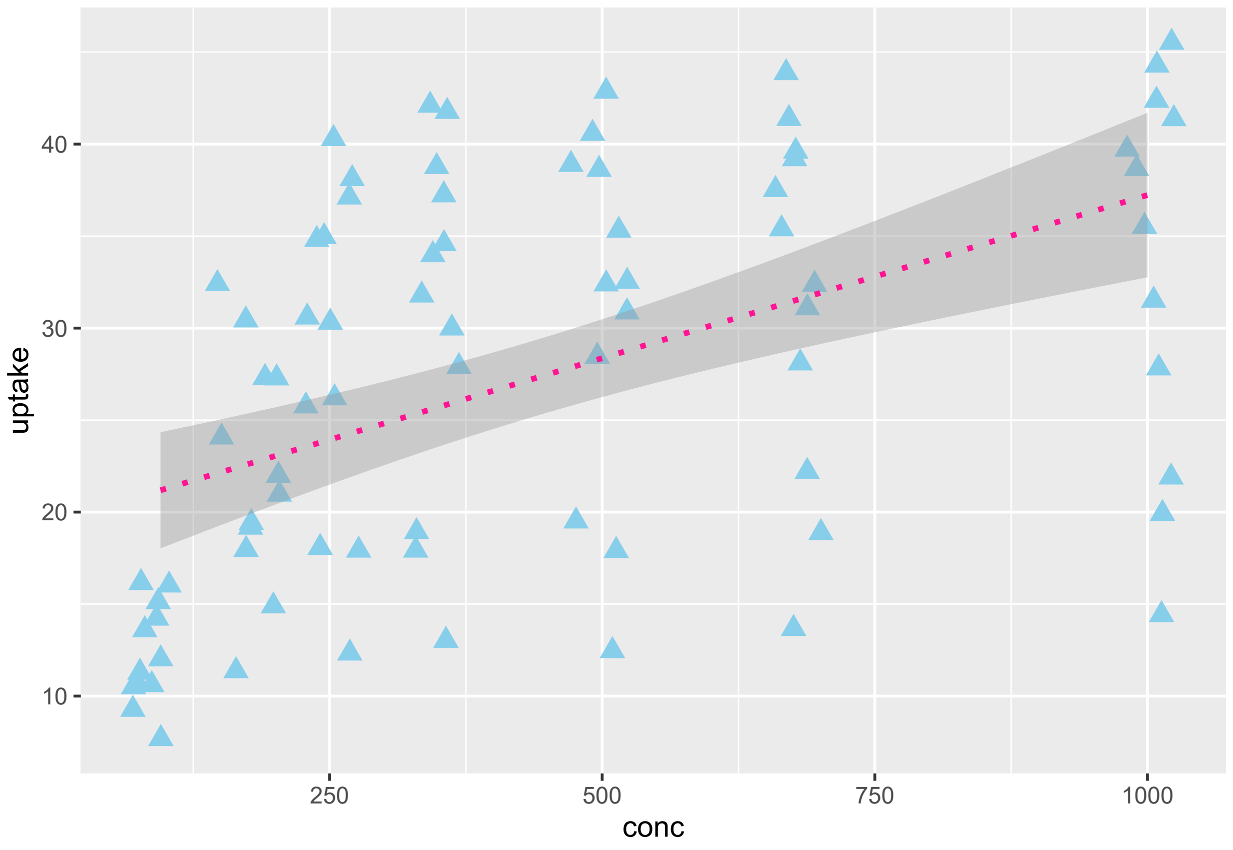
```
ggplot(CO2) + aes(conc, uptake) + geom_jitter()
```



2. Add a dotted trend line using linear regression("lm") in your favourite colour. Change the dots to sky blue large triangles.

```
ggplot(CO2) + aes(conc, uptake) +
  geom_jitter(shape = 17, size = 3, colour = "skyblue") +
  geom_smooth(method = "lm", colour = "deeppink1", linetype = "dotted")
```

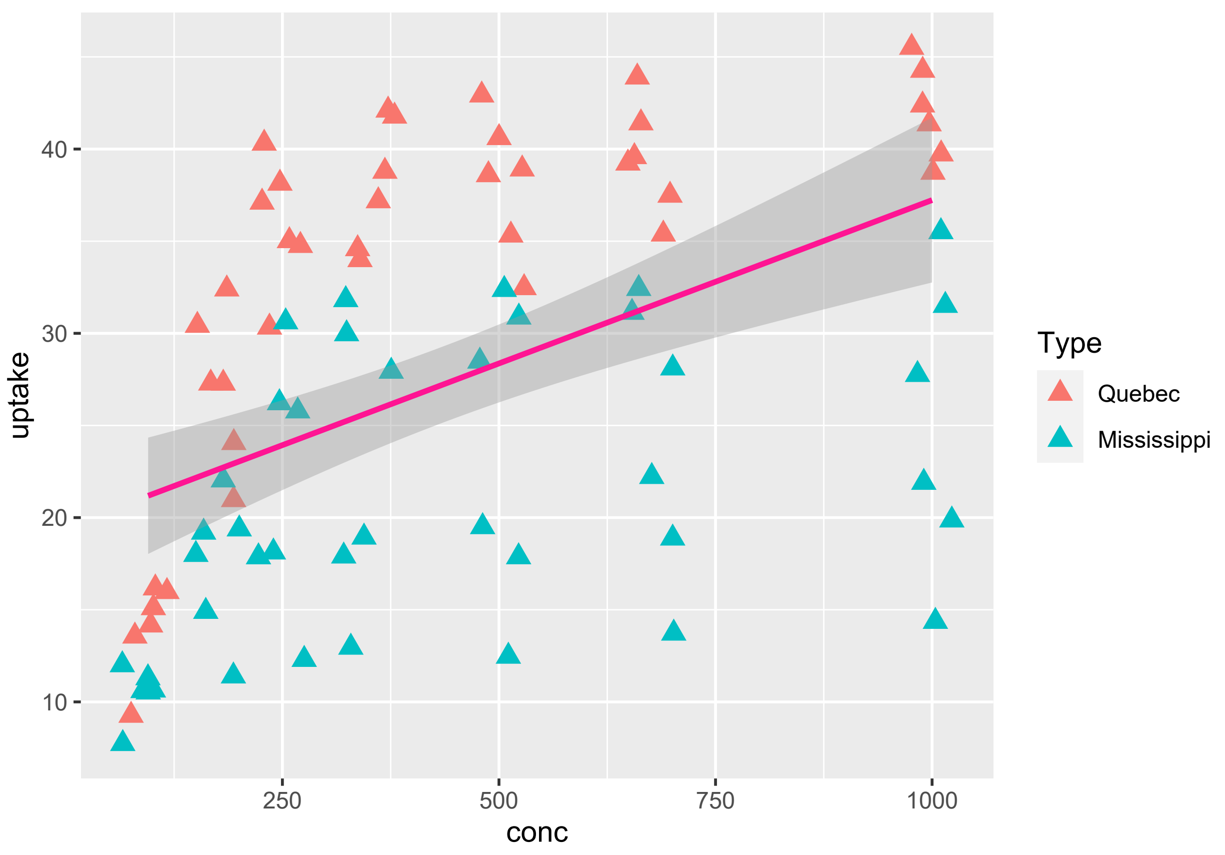
```
## `geom_smooth()` using formula 'y ~ x'
```



3. Encode the origin of the plant in the colour of the triangles. Make sure you still have only **one** trend line.

```
ggplot(CO2) + aes(conc, uptake) +
  geom_jitter(aes(colour = Type), shape = 17, size = 3) +
  geom_smooth(method = "lm", colour = "deeppink1")
```

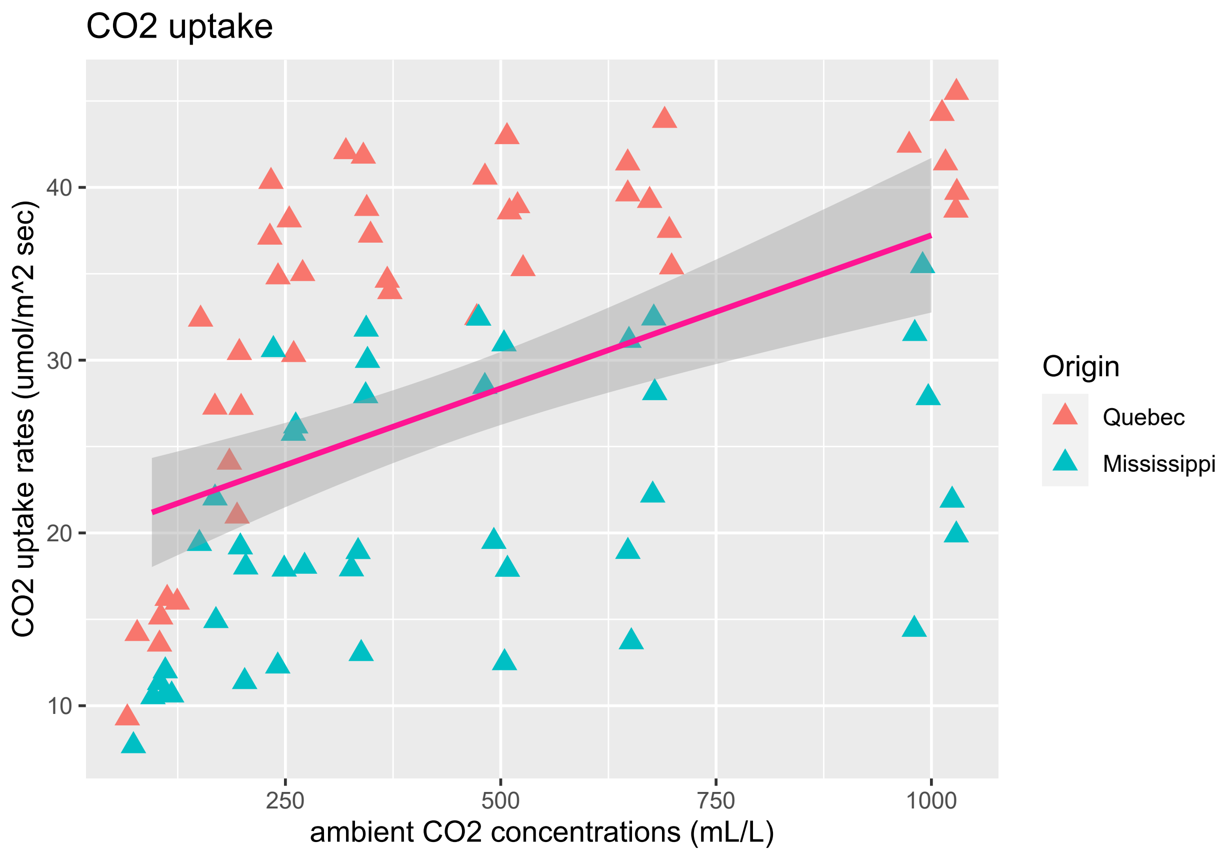
```
## `geom_smooth()` using formula 'y ~ x'
```



4. Label the chart, axes and legend meaningfully.

```r
ggplot(CO2) + aes(conc, uptake) +
  geom_jitter(aes(colour = Type), shape = 17, size = 3) +
  geom_smooth(method = "lm", colour = "deeppink1") +
  labs(title = "CO2 uptake",
       colour = "Origin",
       x = "ambient CO2 concentrations (mL/L)",
       y = "CO2 uptake rates (umol/m^2 sec)")
```

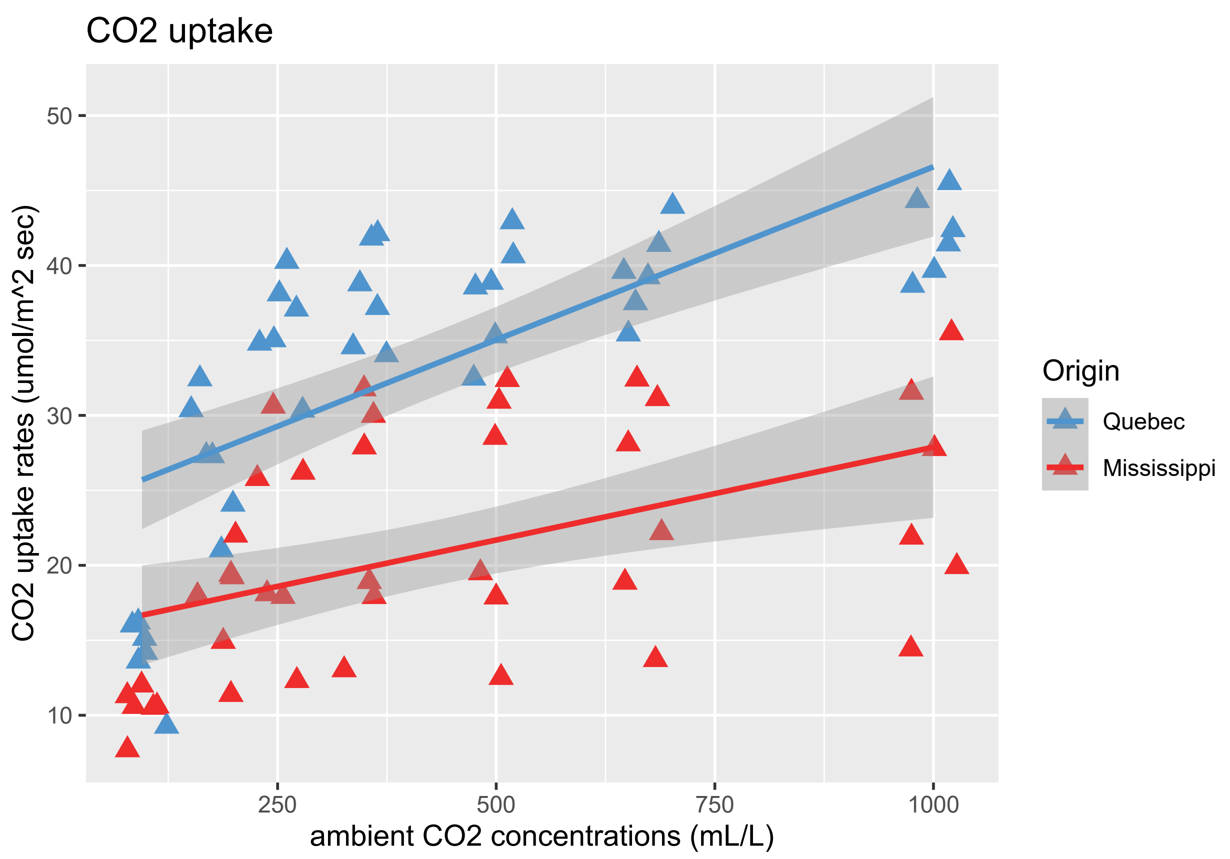
```
## `geom_smooth()` using formula 'y ~ x'
```



5. Use two trend lines (one for each origin). Colour plants from Quebec blue and plants from Mississippi red.

```
ggplot(CO2) + aes(conc, uptake, colour = Type) +
  geom_jitter(shape = 17, size = 3) +
  geom_smooth(method = "lm") +
  scale_colour_manual(values = c("steelblue3", "firebrick2")) +
  labs(title = "CO2 uptake",
       colour = "Origin",
       x = "ambient CO2 concentrations (mL/L)",
       y = "CO2 uptake rates (umol/m^2 sec)")
```

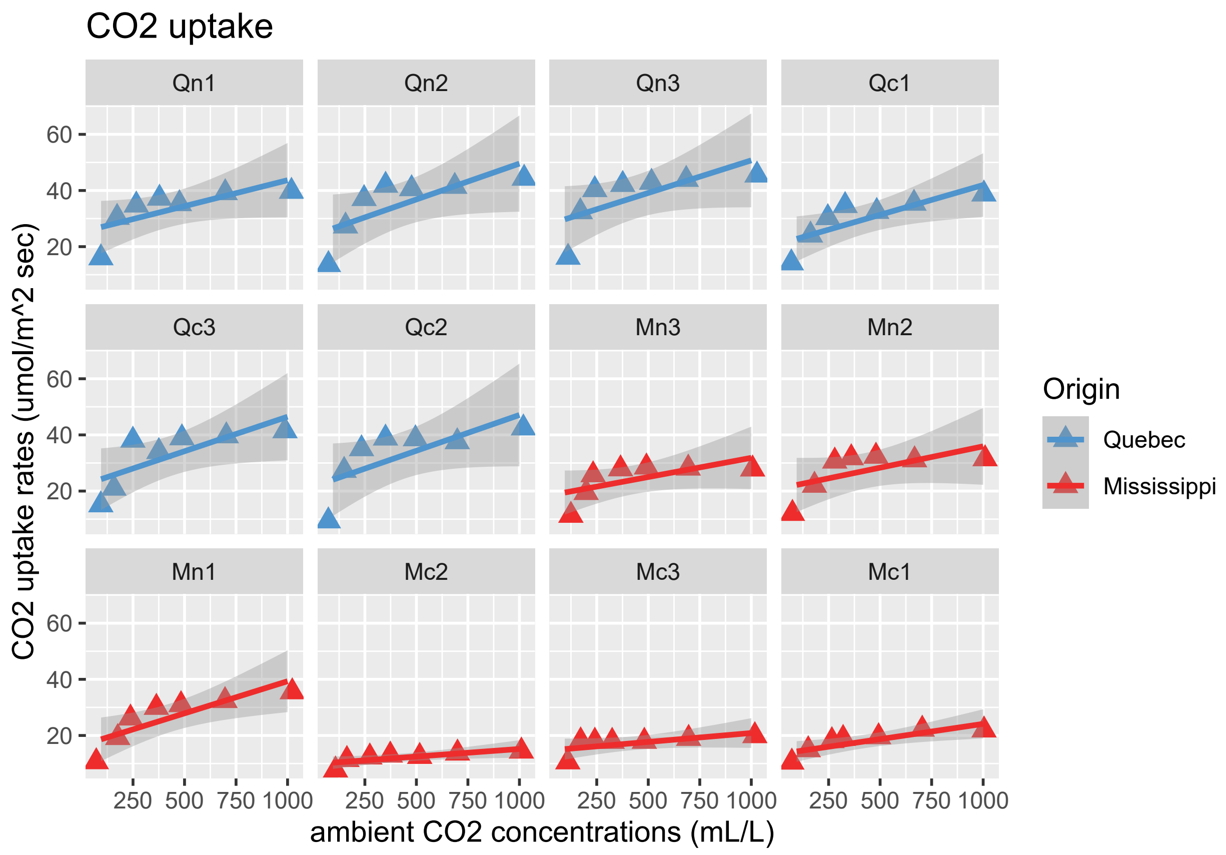
```
## `geom_smooth()` using formula 'y ~ x'
```



6. Create the chart above for each Plant using `facet_wrap()`.

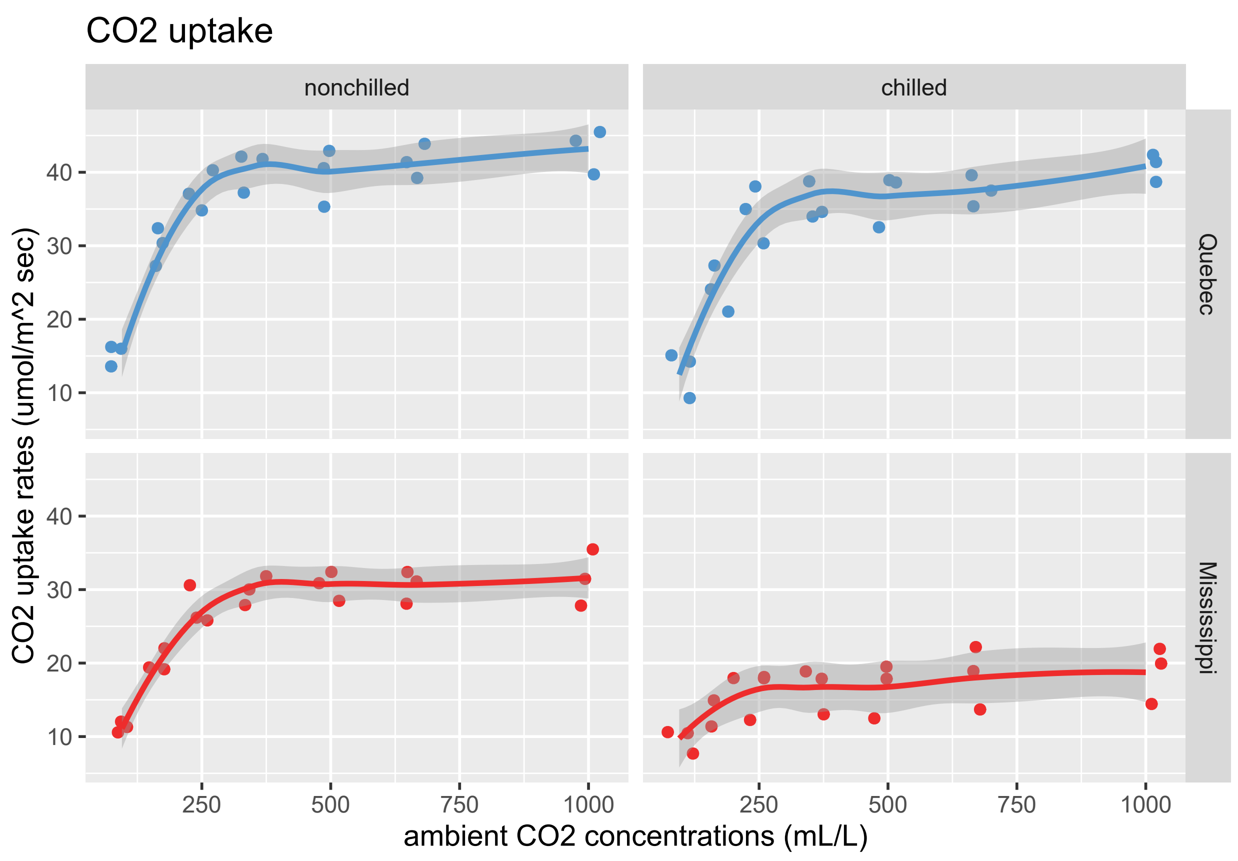
```
ggplot(CO2) + aes(conc, uptake, colour = Type) +
  geom_jitter(shape = 17, size = 3) +
  geom_smooth(method = "lm") +
  scale_colour_manual(values = c("steelblue3", "firebrick2")) +
  labs(title = "CO2 uptake",
       colour = "Origin",
       x = "ambient CO2 concentrations (mL/L)",
       y = "CO2 uptake rates (umol/m^2 sec)") +
  facet_wrap(vars(Plant))
```

```
## `geom_smooth()` using formula 'y ~ x'
```



7. Create the following chart. Tip: You can hide the legend using the `theme()` function and its `legend.position` parameter.

```
ggplot(CO2) + aes(conc, uptake, colour = Type) +
  geom_jitter() +
  geom_smooth(method = "loess", formula = y ~ x) +
  scale_colour_manual(values = c("steelblue3", "firebrick2")) +
  labs(title = "CO2 uptake",
       colour = "Origin",
       x = "ambient CO2 concentrations (mL/L)",
       y = "CO2 uptake rates (umol/m^2 sec)") +
  facet_grid(Type ~ Treatment) +
  theme(legend.position = "none")
```



Mammals sleep

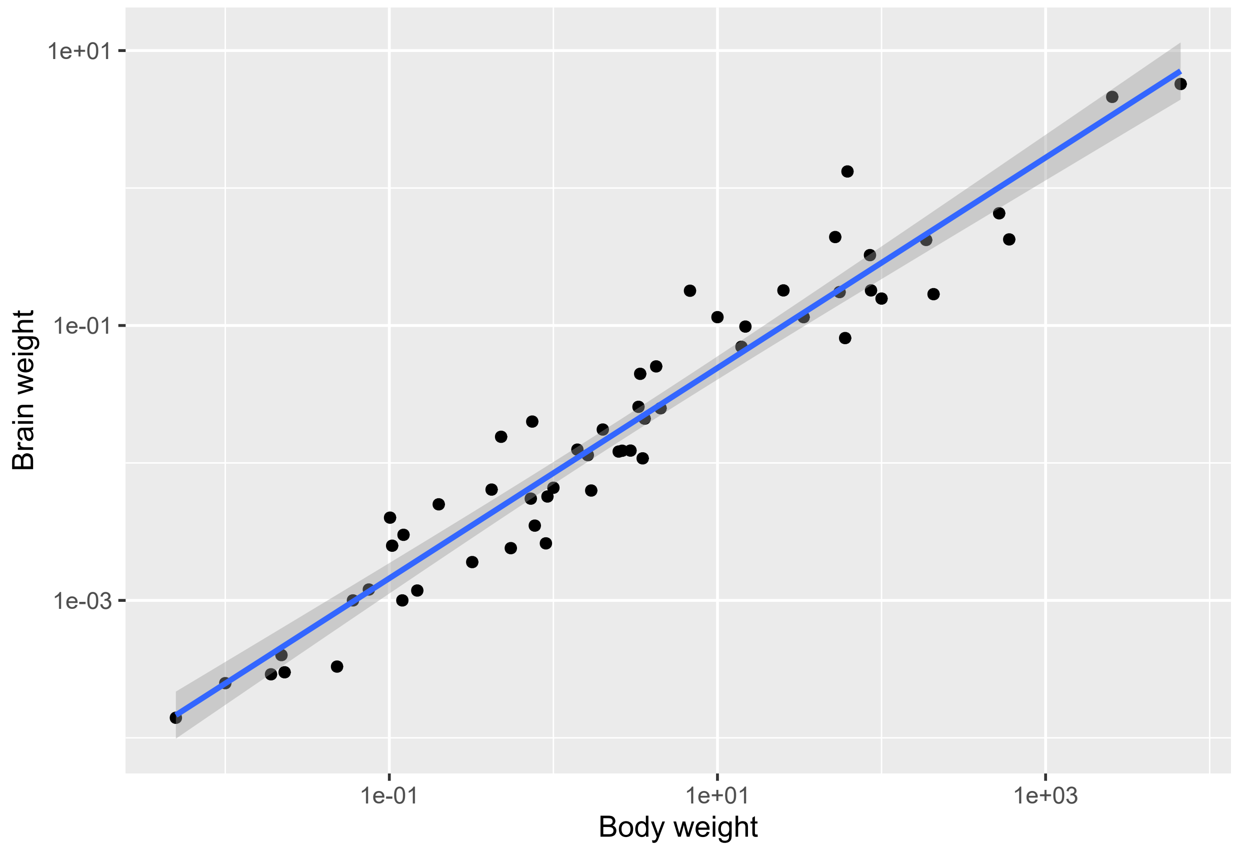
Use the `msleep` data set for the following exercises.

1. Investigate if there is a correlation between body weight and brain weight. You might need to use logarithmic scales.

```
ggplot(msleep) + aes(bodywt, brainwt) + geom_point() +
  scale_x_log10() + scale_y_log10() +
  geom_smooth(formula = y~x, method = "lm") +
  labs(x = "Body weight", y = "Brain weight")
```

```
## Warning: Removed 27 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 27 rows containing missing values (geom_point).
```

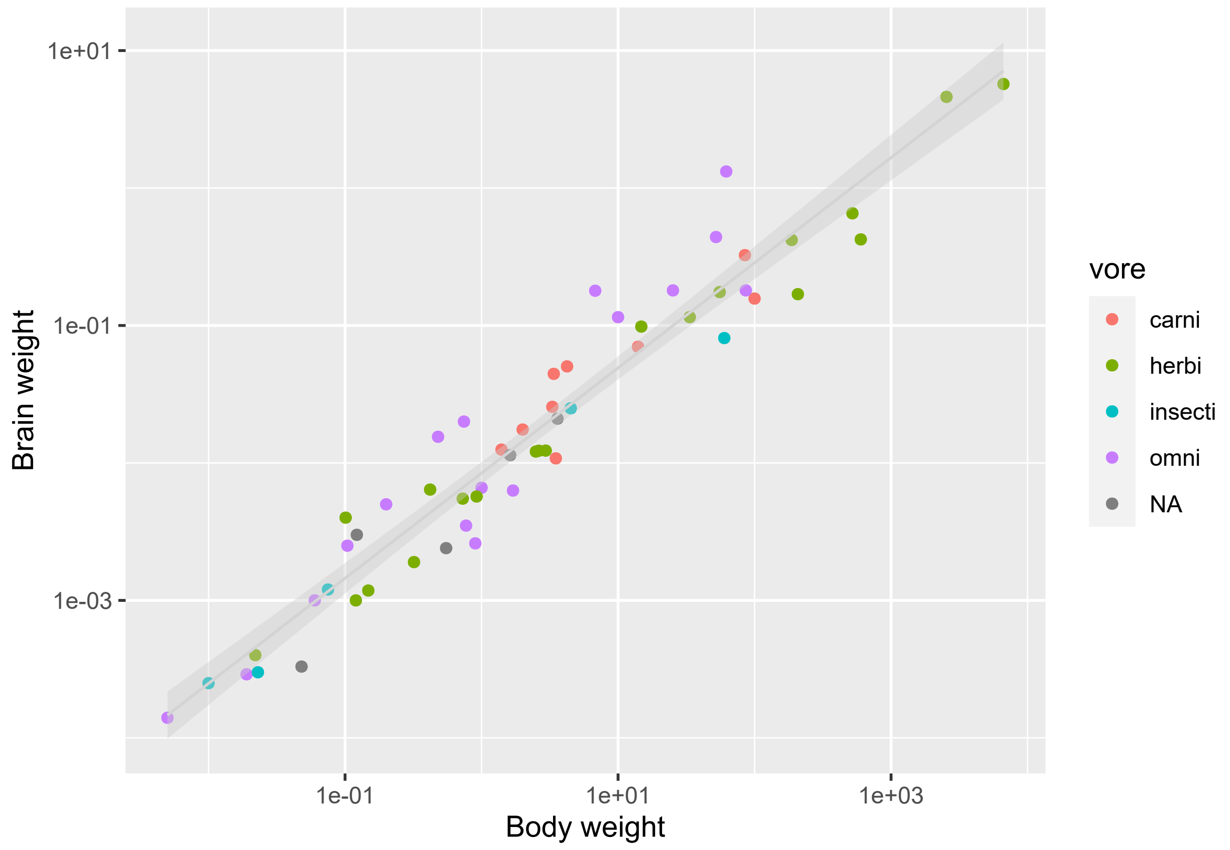


2. Investigate if the food has an influence on body/brain weight.

```
ggplot(msleep) + aes(bodywt, brainwt) +
  geom_point(aes(colour = vore)) +
  geom_smooth(formula = y~x, method = "lm",
              size = 0.5, colour= "lightgrey", fill = "grey80") +
  scale_x_log10() + scale_y_log10() +
  labs(x = "Body weight", y = "Brain weight")
```

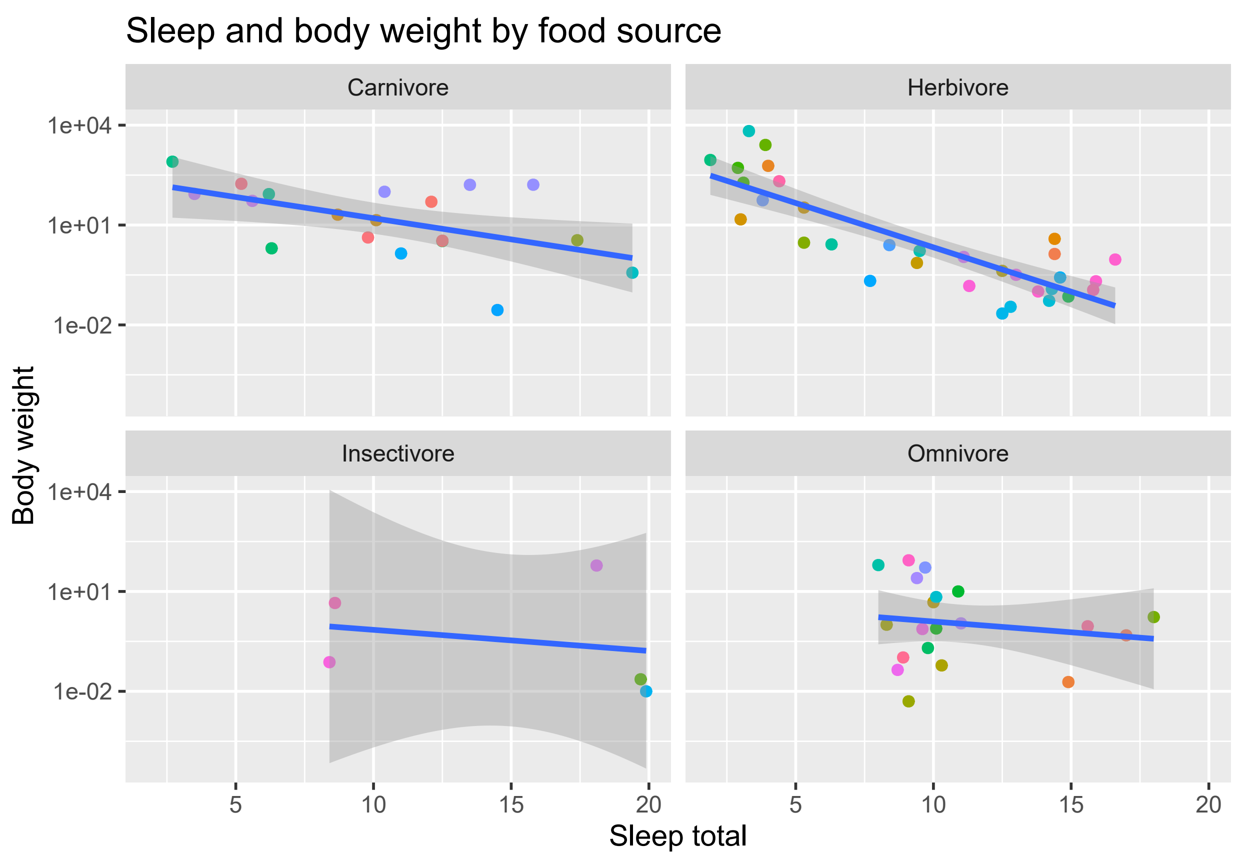
```
## Warning: Removed 27 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 27 rows containing missing values (geom_point).
```



3. Does the body weight have an influence on the total sleep? What else has an influence on the sleep behaviour?

```r
vore.labels <- c("Carnivore", "Herbivore", "Insectivore", "Omnivore")
names(vore.labels) <- c("carni", "herbi", "insecti", "omni")

msleep.noNA <- msleep[!is.na(msleep$vore),]

ggplot(msleep.noNA) + aes(sleep_total, bodywt) +
  geom_point(aes(colour = genus)) +
  geom_smooth(formula = y~x,
              method = "lm") +
  scale_y_log10() +
  labs(x = "Sleep total", y = "Body weight",
       title = "Sleep and body weight by food source") +
  facet_wrap(vars(vore), labeller = labeller(vore = vore.labels)) +
  theme(legend.position = "none")
```



Sleep and body weight by food source
Carnivore
Herbivore
Insectivore
Omnivore
1e+04
1e+01
1e-02
Body weight
5
10
15
20
Sleep total