Bab 5 Membuat grafik dengan R

Dasapta Erwin Irawan dan Prana Ugi

August 23, 2015

Contents

1 Pendahuluan

R memiliki kemampuan visualisasi yang sangat baik. Bila anda masih menggunakan *spreadsheet* untuk membuat grafik, cobalah R. Tidak ada salahnya. Bila merasa terlalu sulit karena berbasis *command line* tinggalkan saja dan kembali ke Ms. Excel. Tapi kalau merasa cocok, tuliskan pengalaman anda dalam bentuk *blogpost* atau artikel pendek. Kirimkan kepada kami.

Seperti halnya fungsi-fungsi yang lain, R telah menyediakan fungsi dasar untuk membuat grafik atau plot. Namun demikian karena sifatnya yang *open source*, maka banyak pihak yang berkontribusi dengan membuat package. Beberapa package yang terkenal adalah:

- `lattice`
- `ggplot2`

Untuk menggunakannya jangan lupa mengetik perintah:

```
install.packages("lattice")
install.packages("ggplot2")
library(lattice)
library(ggplot2)
```

2 Membuat grafik dengan fungsi dasar R

Terinspirasi oleh blog Lee Mendelowitz "plotting using R's base graph functions"

2.1 Parameter dan fungsi terkait

Untuk membuat grafik dengan fungsi dasar R, maka anda harus mengetahui beberapa parameter sebagai berikut:

- `pch`: tipe simbol
- `lty`: tipe garis
- `lwd`: lebar garis
- `col`: warna plot
- `las`: orientasi nama sumbu
- `bg`: warna latar
- `mar`: ukuran margin
- `oma`: ukuran margin luar
- `mfrow`: jumlah plot per baris dan kolom. Ini untuk menampilkan lebih dari satu plot dalam satu halaman.

Beberapa fungsi terkait: * `plot`: fungsi dasar membuat grafik. Secara *default* akan membuat *scatterplot*. * `axis`: menambah label atau *tick mark* pada sumbu * `lines`: menambah garis pada plot * `points`: menambah titik pada plot * `text`: menambah teks pada plot * `title`: menambah judul sumbu pada plot * `mtext`: mengatur margin teks

2.2 Plot pertamaku

Dalam contoh kali ini, kita akan menggunakan data yang dibuat secara acak oleh R dengan perintah `rnorm()`. Jadi anda belum perlu mengunduh dan mengimpor data dari luar.

```
x <- rnorm(100)
y <- rnorm(100)
plot(x, y, pch=21,
     mar=c(4,4,2,2),
     col='red',bg='black',
     xlim=c(-3,3),
     ylim=c(-3,3))
fit <- lm(y ~ x)
abline(fit, lwd = 3, col = "blue")
title('Plot pertamaku')
text(-2, -2, 'Label')
legend("topleft",
      legend="Data",
      pch=21,
      pt.bg='black',
      col='red')
```

Membuat plot dengan fungsi dasar R sangatlah sederhana. Misalnya anda menggunakan dataset `mtcars` yang telah ada dalam instalasi R. Anda dapat langsung melakukan korelasi dan menambahkan garis regresi. Ketik perintah sebagai berikut.

```
attach(mtcars) # untuk memberitahu R bahwa kita menggunakan dataset `mtcars`
mtcars # melihat isi data
str(mtcars) # melihat tipe data, seluruhnya numerik
dim(mtcars) # melihat dimensi data = 32 baris dan 11 kolom
plot(wt, mpg) # membuat scatterplot antara wt (weight) dan mpg (miles per galon)
abline(lm(mpg~wt)) # membuat garis regresi
title("Regresi antara berat mobil (wt)-konsumsi BBM (mpg)")
```

2.3 Menyimpan plot

Anda dapat menyimpan plot yang telah dibuat dengan perintah sebagai berikut sesuai dengan kebutuhan format penyimpanan yang diinginkan.

```
pdf("plotku.pdf") # untuk format pdf. Anda dapat melengkapinya dengan lokasi folder kerja.
png("plotku.png") # untuk format png
jpeg("plotku.jpg") # untuk format jpg
bmp("plotku.bmp") # untuk format bmp
```

2.4 Membuat histogram

Anda dapat membuat histogram dengan fungsi `hist(x)` dengan "x" bertipe numerik. Kita masih menggunakan dataset `mtcars`. Berikut contohnya.

```
## histogram sederhana untuk variabel "mpg"
hist(mtcars$mpg)
```

```
## histogram berwarna dengan pengaturan jumlah "Bins"
hist(mtcars$mpg, breaks=12, col="red")

## Menambahkan kurva distribusi normal
x <- mtcars$mpg
h<-hist(x,
        breaks=10,
        col="red",
        xlab="mil per galon",
        main="Histogram dengan garis kurva normal")
xfit <- seq(min(x),
            max(x),
            length=40)
yfit <- dnorm(xfit,
              mean=mean(x),
              sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)

## Kernel Density Plot
d <- density(mtcars$mpg) # returns the density data
plot(d) # plots the results

## Density Plot berwarna
d <- density(mtcars$mpg)
plot(d,
     main="Kernel Density mil per galon")
polygon(d,
        col="red",
        border="blue")
```

2.5 Dot plots

Bentuk plot lainnya adalah "dot plot". Gunakan fungsi `dotchart()` untuk membuatnya. Anda dapat menambah opsi pengelompokkan. Berikut contohnya.

```
dotchart(mtcars$mpg,
         labels=row.names(mtcars),
         cex=.7,
         main="Konsumsi BBM berbagai merk mobil",          xlab="Miles Per Gallon")

mtcars
# Dotplot: dengan pengelompokkan
## Sortasi berdasarkan mpg, pengelompokkan dan pewarnaan berdasarkan cylinder
x <- mtcars[order(mtcars$mpg),] # sortasi mpg
x$cyl <- factor(x$cyl) # harus dikonversi menjadi tipe factor sebagai dasar klasifikasi
x$color[x$cyl==4] <- "red" # pengelompokkan warna
x$color[x$cyl==6] <- "blue"
x$color[x$cyl==8] <- "darkgreen"
dotchart(x$mpg,
         labels=row.names(x),
         cex=.7,groups= x$cyl,
```

```
main="Konsumsi BBM berdasarkan merk mobil (berdasarkan jumlah silinder)",
xlab="mil per galon",
gcolor="black",
color=x$color)
```

2.6 Bar Plots

Membuat "barplots" dengan fungsi `barplot(height)`, dengan `height` berjenis vektor atau matriks. Bila `height` adalah vektor, maka nilainya akan menentukan tinggi balok (bar) dalam plot. Bila ia berjenis matriks dan bila digunakan opsi `beside=FALSE` maka tiap balok akan m .... then each bar of the plot corresponds to a column of height, with the values in the column giving the heights of stacked "sub-bars". If height is a matrix and beside=TRUE, then the values in each column are juxtaposed rather than stacked. Include option names.arg=(character vector) to label the bars. The option horiz=TRUE to createa a horizontal barplot.

```
# Simple Bar Plot
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution",
   xlab="Number of Gears")

# Simple Horizontal Bar Plot with Added Labels
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution", horiz=TRUE,
  names.arg=c("3 Gears", "4 Gears", "5 Gears"))

# Stacked Bar Plot with Colors and Legend
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by Gears and VS",
  xlab="Number of Gears", col=c("darkblue","red"),
  legend = rownames(counts))

# Grouped Bar Plot
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by Gears and VS",
  xlab="Number of Gears", col=c("darkblue","red"),
  legend = rownames(counts), beside=TRUE)
```

Bar plots need not be based on counts or frequencies. You can create bar plots that represent means, medians, standard deviations, etc. Use the aggregate( ) function and pass the results to the barplot( ) function.

By default, the categorical axis line is suppressed. Include the option axis.lty=1 to draw it.

With many bars, bar labels may start to overlap. You can decrease the font size using the cex.names = option. Values smaller than one will shrink the size of the label. Additionally, you can use graphical parameters such as the following to help text spacing:

```
# Fitting Labels
par(las=2) # make label text perpendicular to axis
par(mar=c(5,8,4,2)) # increase y-axis margin.

counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution", horiz=TRUE, names.arg=c("3 Gears", "4 Gears", "5   Gears"), ce
```

Yang akan kami tambahkan adalah tipe:

- *Line charts*
- *Pie charts*
- *Boxplots*
- *Violin Plots*
- *Bagplot*
- *Scatterplots*

2.7 Multiple plot dalam satu halaman

Untuk menampilkan lebih dari satu plot dalam satu halaman anda dapat menggunakan perintah `mfrow`. Perintah ini biasanya diperlukan bila anda ingin membandingkan dua grafik secara berdampingan atau atas-bawah. Berikut contohnya. Selain cara ini anda juga dapat menginstalasi package `gridExtra` yang lebih mudah.

```
plot.new() # untuk membuat plot baru
par(mfrow= c(2,2)) # mengatur jumlah baris dan kolom
par(mar = c(3, 3, 2, 2)) # mengatur margin antar plot
plot(x, y, pch = 20, main="plot 1")
plot(x, z, pch = 19, main="plot 2")
plot(y, z, pch = 1, main="plot 3")
plot(y, z, pch = 5, main="plot 4")
```

2.8 Plot dua grup data dalam satu grafik

Dengan menggunakan perintah `points()` anda dapat menambahkan titik data yang berbeda grup. Berikut contohnya.

```
plot.new()
x <- rnorm(100)
y <- x + rnorm(100)
g <- gl(2, 50, labels = c("Kelas A", "Kelas B"))
str(g)
plot(x,y, type='n') # Draws no points
points(x[g == "Kelas A"], y[g == "Kelas A"],
       col="blue",
       pch=1)
points(x[g == "Kelas B"], y[g == "Kelas B"],
       col="red",
       pch=19)
legend("topleft", c("Kelas A", "Kelas B"),
       col=c("blue", "red"),
       pch=c(1,19))
```

3 Membuat grafik dengan package `ggplot2`

This document was translated from: Build a plot layer by layer by Hadley Wickham

4 Pendahuluan

Salah satu ide dibalik ggplot2 adalah ia memungkinkan kita secara iteratif membuat plot yang sangat kompleks selangkah demi selangkah. Setiap langkah ekivalen dengan membuat sebuah lapisan di atas lapisan lainnya (*layer by layer*).

Plot dibuat dari:

- data yang sumbernya sama tapi diplot bersamaan pada sumbu yang sama,
- atau data yang sama sumbernya tapi diplot dengan setting yang berbeda, misal satu variabel pada sumbu X dengan dua atau lebih variabel pada sumbu Y,
- atau data yang berbeda sumbernya.

Anda telah dapat membuat plot dengan fungsi seperti `geom_point()` dan `geom_histogram()`. Dalam bab ini anda akan belajar lebih dalam mengenai *layers* (lapisan dalam plot) dan bagaimana anda mengendalikan lima komponen (dalam sebuah plot): data, estetis, geometri, statistik, dan penyesuaian posisi.

Tujuannya adalah memberikan perkakas (*tools*) bagi anda untuk membangun plot yang canggih sesuai dengan masalah yang akan dipecahkan.

Bab yang bersifat teoritis ini akan didampingi oleh bab berikutnya "toolbox" yang lebih praktikal, mengaplikasikan komponen-komponen dasar untuk menjawab tantangan visualisasi.

5 Membangun sebuah plot

Sejauh ini, bila kita membuat plot dengan fungsi `ggplot()`, kita menganggap telah membuat sebuah layer dengan fungsi `geom()`. Tapi sangat penting untuk memahami bahwa sebenarnya ada dua langkah terpisah di dalamnya. Pertama kita membuat sebuah plot dengan dataset baku (*default*) "mtcars" dan elemen estetis:

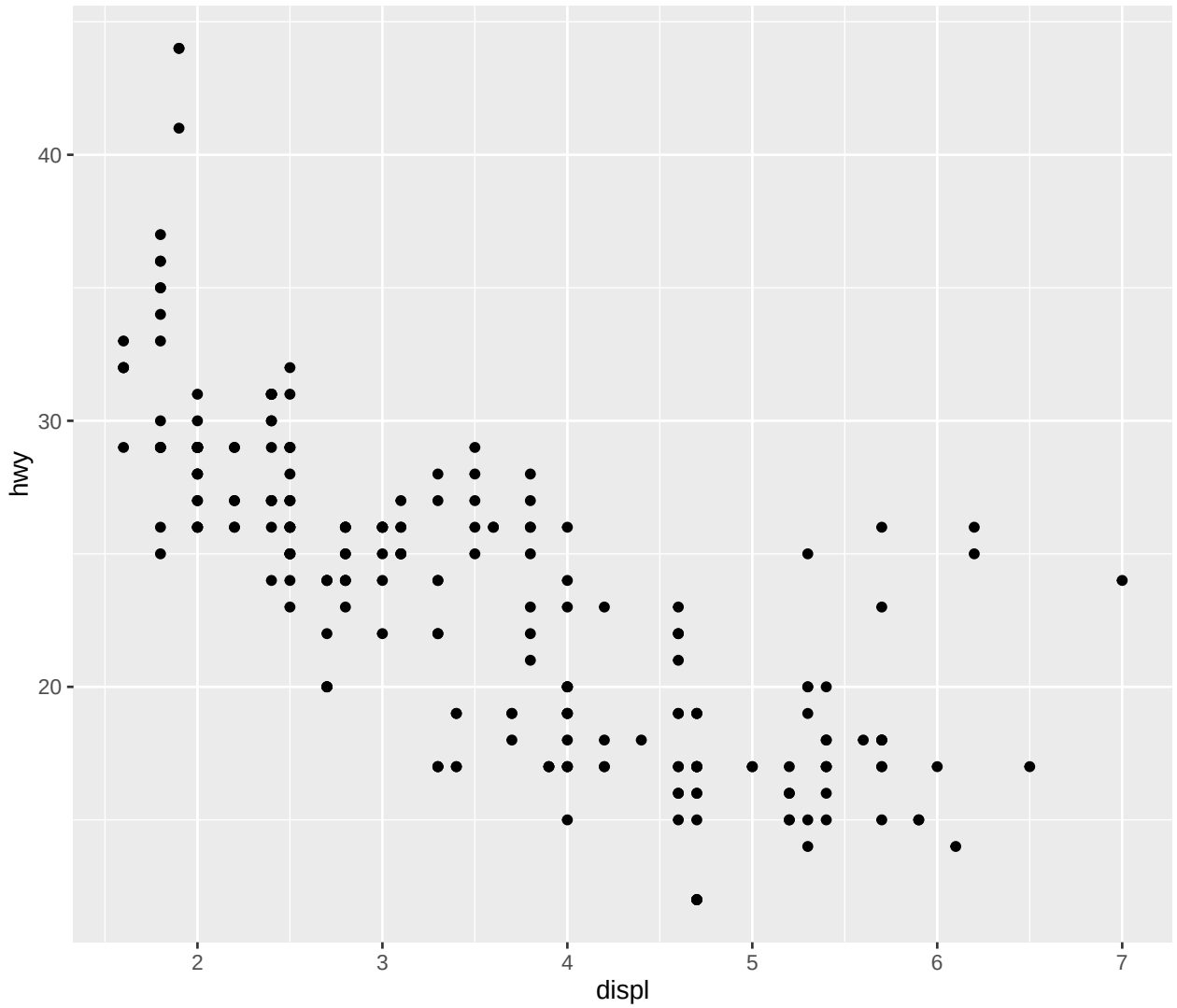
```
library(ggplot2)
p <- ggplot(mpg, aes(displ, hwy))
p
```

Plot tidak tampil (pesan kesalahan akan muncul) sebelum kita membuat sebuah layer: tidak ada yang bisa dilihat!

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

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



`geom_point()` adalah jalan pintas (*shortcut*). Di belakang layar, fungsi tersebut memanggil fungsi `layer()` untuk membuat layer baru:

```
p + layer(
  mapping = NULL,
  data = NULL,
  geom = "point", geom_params = list(),
  stat = "identity", stat_params = list(),
  position = "identity"
)
```

Fungsi tersebut di atas mengatur spesifikasi lima komponen pada layer:

- **mapping**: satu set setting estetik menggunakan fungsi `aes()`. Setting baku (default) ggplot2 akan digunakan bila setting khusus tidak disebutkan dalam fungsi atau NULL.
- **data**: layer data akan menggunakan dataframe yang telah disebut dalam fungsi `ggplot()`. Hal lebih detil akan dijelaskan lebih rinci dalam *bagian data*.

- **geom**: layer ini menyebutkan spesifikasi object geometri yang digunakan untuk memperlihatkan observasi dalam plot. Komponen ini serta penggunaannya sebagai salah satu *toolbox* akan dibahas lebih dalam di *bagian geom*.

Komponen geoms dapat menggunakan argumen tambahan. Seluruh perintah `geom` menggunakan aestetik sebagai parameternya. Kalau kita gunakan sebuah komponen aestetik, misal `colour` sebagai parameter, perintah itu tidak akan di-skalakan, sehingga membuat kita dapat mengendalikan tampilan plot, seperti dijelaskan dalam bagian `setting vs. mapping` di bawah ini. You can pass params in ... (in which case stat and geom parameters are automatically teased apart), or in a list passed to geom_params.

- **stat**: layer ini menset transformasi statistik yang akan digunakan. Transformasi statistik yang meringkas elemen statisik yang berguna sebagai kunci histogram plot dan *smoothing*. Untuk menjaga data seperti adanya, gunakan "identity" stat. Pelajari lebih banyak transformasi statistik.

Kita hanya perlu menset satu perintah `stat` dan `geom`: setiap `geom` memiliki sebuah stat default, dan setiap `stat` adalah adalah sebuah `geom` default.

- **position**: layer ini digunakan untuk mengatur object yang saling overlap, seperti *jittering*, *stacking* atau *dodging*. Lebih dalam akan dibahas di bagian *position*.

Sangatlah penting untuk memahami fungsi `layer()` untuk memahami konsep *layer*, tapi kita akan jarang menggunakannya. Kita akan menggunakan jalan pintas (*shortcut*) fungsi `geom_point(mapping, data, ...)` yang sama persis dengan peritnah `layer(mapping, data, geom = "point", ...)`.

6 Data

Setiap layer harus memiliki beberapa data yang berkaitan dengannya, dan data tersebut harus ada dalam sebuah dataframe. Ini adalah aturan baku, karena:

Data kita sangat penting, sehingga harus disebutkan secara eksplisit.

Sebuah dataframe akan lebih mudah disimpan dibanding bila dalam bentuk banyak vector, yang artinya lebih mudah untuk direproduksi atau dibagikan kepada pihak lain.

Diantara packages R yang ada, akan ada pemisahan peran yang jelas: ggplot2 memvisualkan data frame, sedangkan package lainnya dapat mentransformasi data frame ke dalam format yang tepat (pelajari hal ini dalam bagian visualisasi model/*model visualisation*).

Data pada tiap layer tidak harus sama, dan seringkali berguna untuk mengkombinasi multiple dataset dalam plot yang sama. Sebagai ilustrasi, saya akan membuat dua buah dataset berkaitan dengan `mpg` dataset. Pertama saya akan melakukan fitting loess model dan membuat prediksi darinya. (Inilah yang dilakukan oleh `geom_smooth()` di belakang layar).

```
mod <- loess(hwy ~ displ, data = mpg)
grid <- data.frame(displ = seq(min(mpg$displ), max(mpg$displ), length = 50))
grid$hwy <- predict(mod, newdata = grid)
head(grid)
```

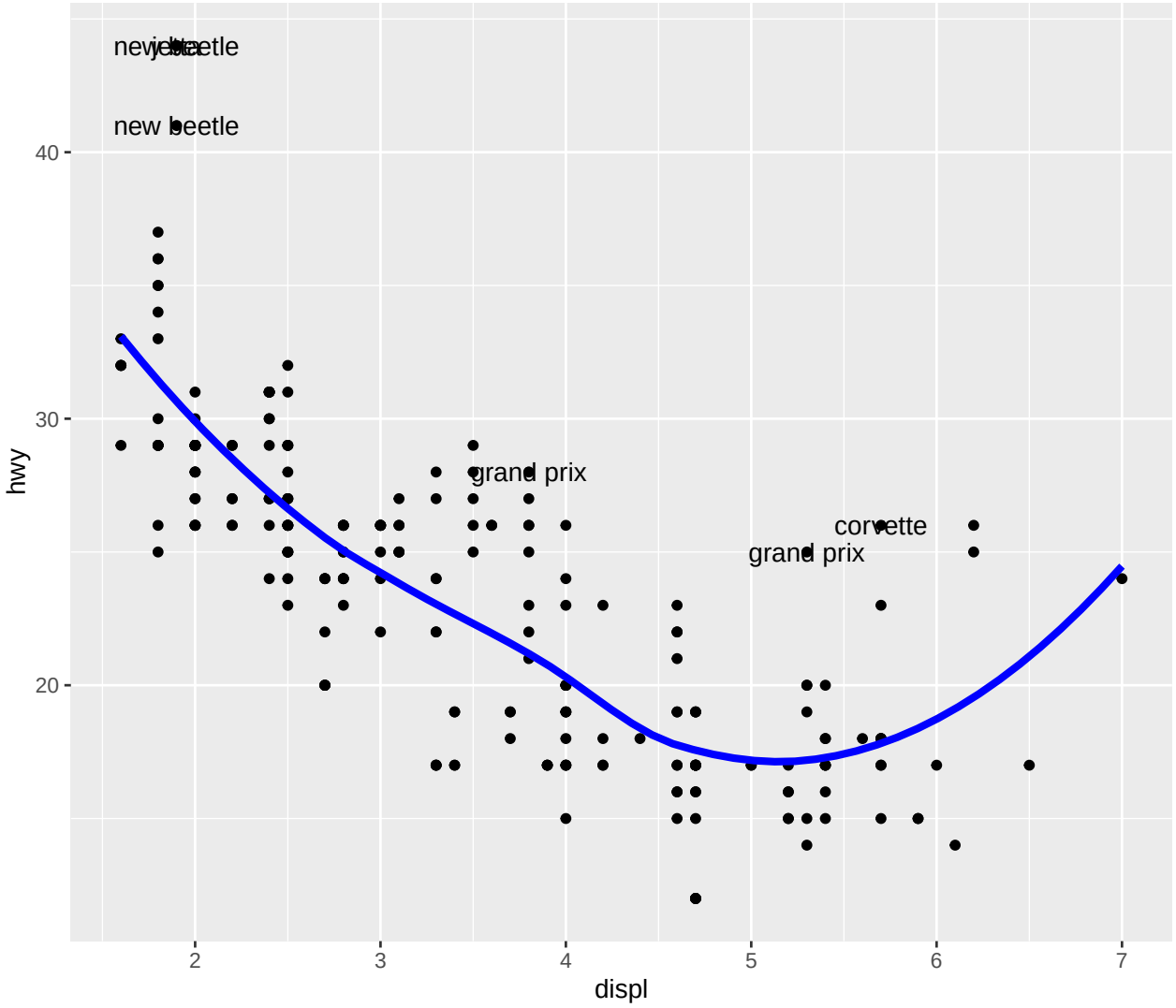
```
##      displ      hwy
## 1 1.600000 33.09286
## 2 1.710204 32.16100
## 3 1.820408 31.26635
## 4 1.930612 30.41403
## 5 2.040816 29.60168
## 6 2.151020 28.82979
```

Selanjutnya saya akan mengisolasi observasi yang sangat jauh dari nilai prediksinya.

```
std_resid <- resid(mod) / mod$s
outlier <- subset(mpg, abs(std_resid) > 2)
```

Saya membuat dataset berikut karena sangat umum untuk mempertajam data mentah dengan ringkasan statistik dan beberapa anotasi. Dengan dataset baru ini, saya mempu memperbaiki *scatterplot* yang pertama dengan mengnumpangkan garis *smoothing*, dan memberikan label untuk titik-titik *outlier*.

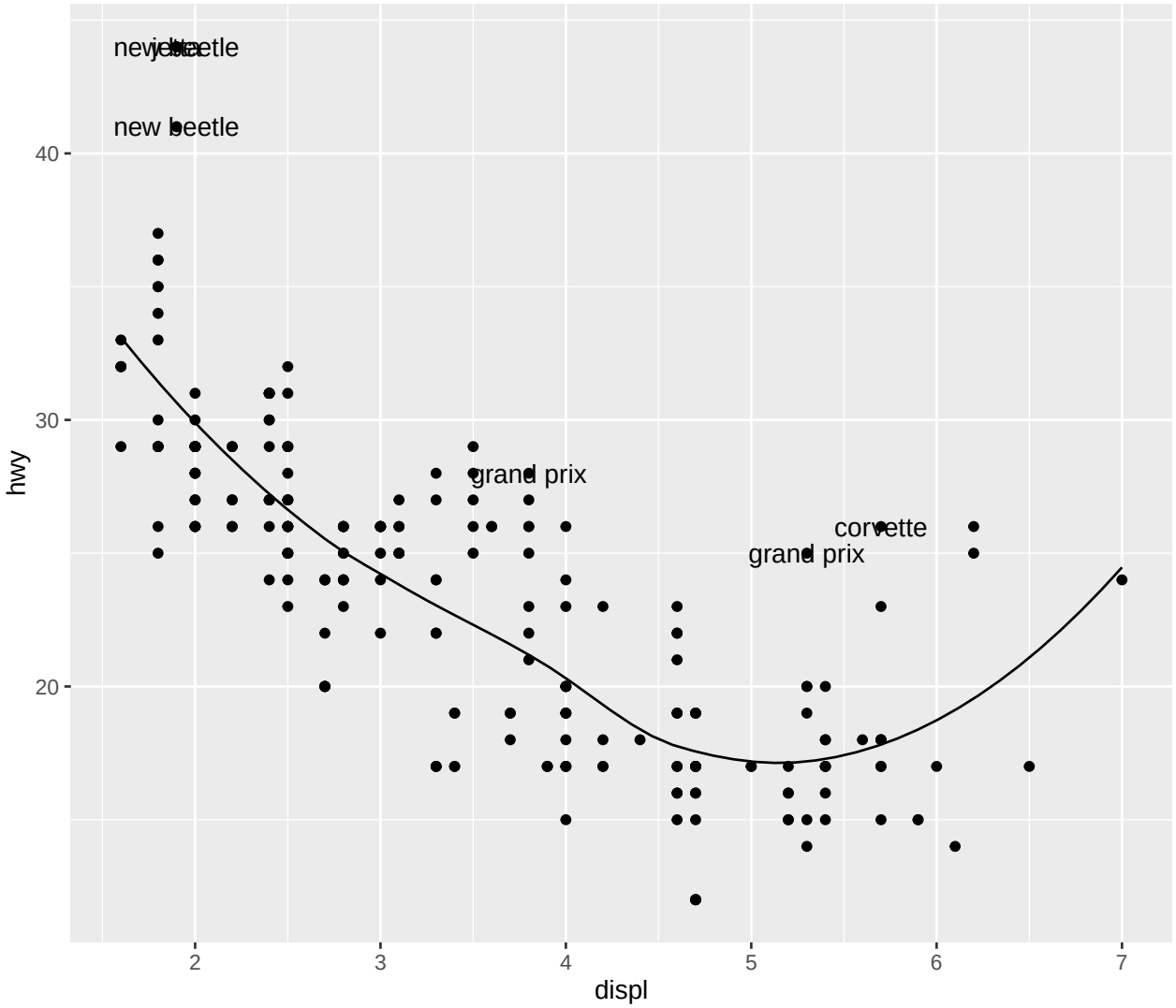
```
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_line(data = grid, colour = "blue", size = 1.5) +
  geom_text(data = outlier, aes(label = model))
```



(Label teks agak susah dibaca, tapi anda akan belajar bagaimana memperbaikinya, atau disebut *polishing*).

Dalam contoh ini, tiap layer menggunakan sebuah dataset yang berbeda. Kita dapat mendefinisi ulang plot yang sama, dengan menghilangkan dataset baku;

```
ggplot(mapping = aes(displ, hwy)) +
  geom_point(data = mpg) +
  geom_line(data = grid) +
  geom_text(data = outlier, aes(label = model))
```



Dalam kasus ini, saya tidak menyukai gaya ini karena data primer akan susah teridentifikasi (dan karena memerlukan pengetikan lebih banyak saat diatur sebagai argumen ke dalam `ggplot()`). Tapi kita mungkin akan lebih menyukainya bila memang tidak jelas dataset primernya, atau saat komponen estetisnya bervariasi antara satu layer dengan layer yang lain.

NB: bila kita menghilangkan dataset dalam `ggplot()`, maka kita harus secara eksplisit menyebutkan sebuah dataset untuk tiap layer. Harap dicatat bahwa *facetting* tidak akan bekerja tanpa dataset default: *facetting* akan mempengaruhi seluruh layer sehingga ia memerlukan dataset dasar yang mendefinisikan facet-facetnya. Lihat *missing facetting variable* untuk lebih jelasnya.

6.1 Latihan

1. Dua argumen pertama dalam ggplot adalah `data` dan `mapping`. Dua argumen pertama dalam fungsi layer adalah `mapping` dan `data`. Mengapa urutan argumennya berbeda? (petunjuk: pikirkan tentang apa yang anda set paling sering).

2. Kode di bawah ini menggunakan `dplyr` package untuk membuat beberapa ringkasan statistik tentang masing-masing golongan mobil (anda akan belajar lebih banyak di bab *data transformation*)

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.2.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
class <- mpg %>%
  group_by(class) %>%
  summarise(n = n(), hwy = mean(hwy))
```

Gunakan data untuk membuat ulang plot di bawah ini.

!(ex1.png)

7 Pemetaan estetik (aesthetic mappings)

Pemetaan estetik didefinisikan oleh fungsi `aes()`, menjelaskan berbagai variabel dipetakan untuk membuat visualisasi. `aes()` memerlukan urutan variable estetik berpasangan seperti berikut:

```
aes(x = displ, y = hwy, colour = class)
```

(kalau kita menggunakan ejaan Amerika, maka gunakan *color*, dan ggplot2 akan memperbaikinya di belakang layar)

Di sini kita memetakan sumbu-x untuk `displ`, sumbu-y untuk ‘`class`. Dua argumen awal dapat dihilangkan, yang mana secara berurutan akan berkaitan dengan kedua sumbu-x dan y), sehingga akan membuat perintah yang hasilnya sama dengan perintah di atas:

```
aes(displ, hwy, colour = class)
```

Saat kita dapat melakukan manipulasi dengan `aes()`, misal `aes(log(carat), log (price))`, `akan lebih baik kalau hanya memasukkan persamaan sederhana. Disarankan untuk memindahkan transformasi yang kompleks dalam perintah`aes()`ke perintah`mutate()`(dalam`dplyr“‘ package), yang akan anda pelajari secara terpisah di bagian *mutate*. Ini memudahkan untuk memeriksa hasil kerja kita dan lebih cepat (karena kita hanya akan perlu melakukan transformasi sekali, tidak tiap kali membuat plot)

7.1 Membuat spesifikasi estetik pada plot vs pada layer

Pemetaan estetik (*aesthetic mappings*) dapat dimasukkan ke dalam fungsi `ggplot()` awal, pada layer secara individual, atau dalam kombinasinya. Semua perintah di bawah ini membuat spesifikasi plot yang sama:

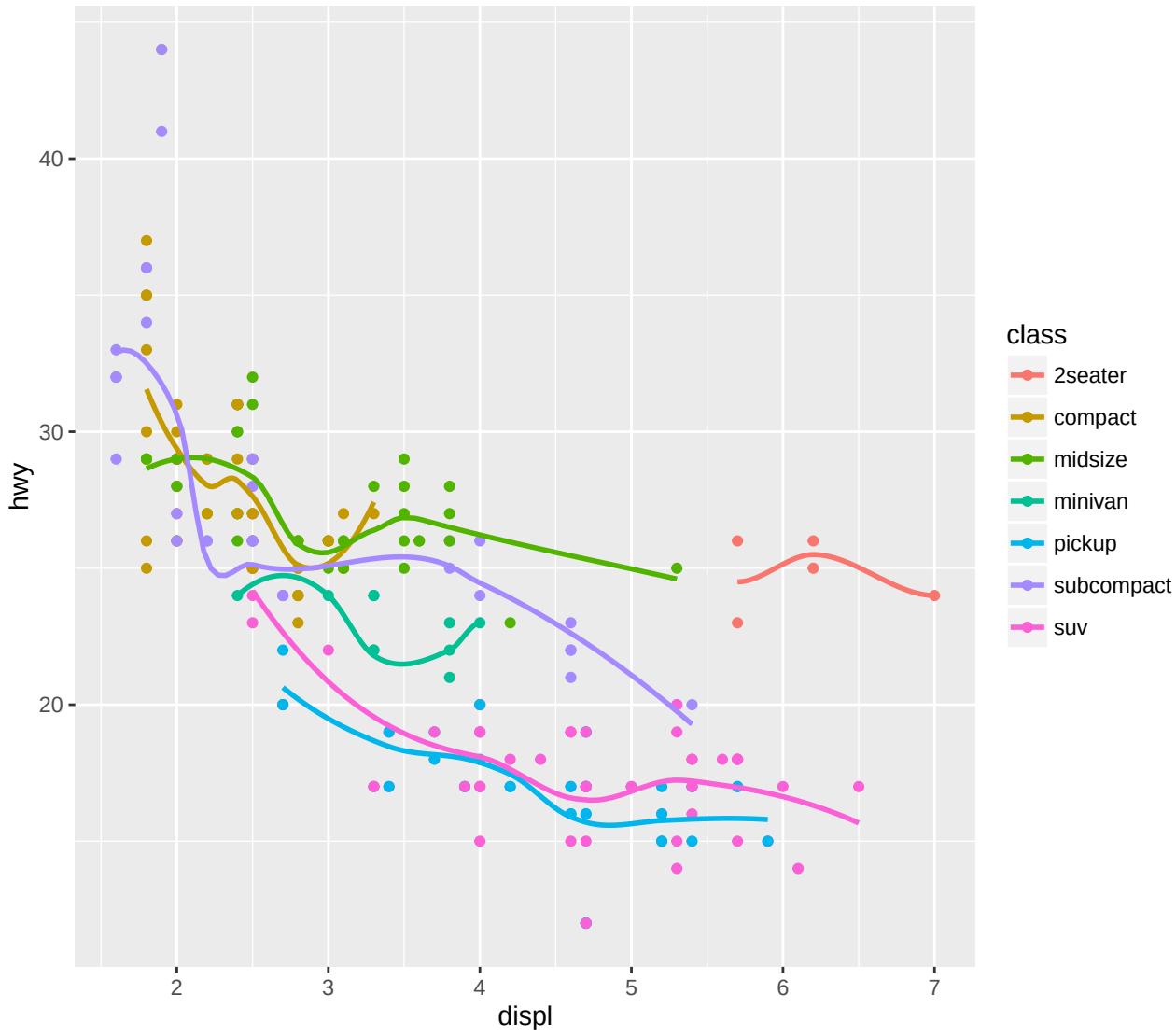
```
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(colour = class))
ggplot(mpg, aes(displ)) +
  geom_point(aes(y = hwy, colour = class))
ggplot(mpg) +
  geom_point(aes(displ, hwy, colour = class))
```

Kita dapat menambah, menindih (*override*), atau menghilangkan mapping:

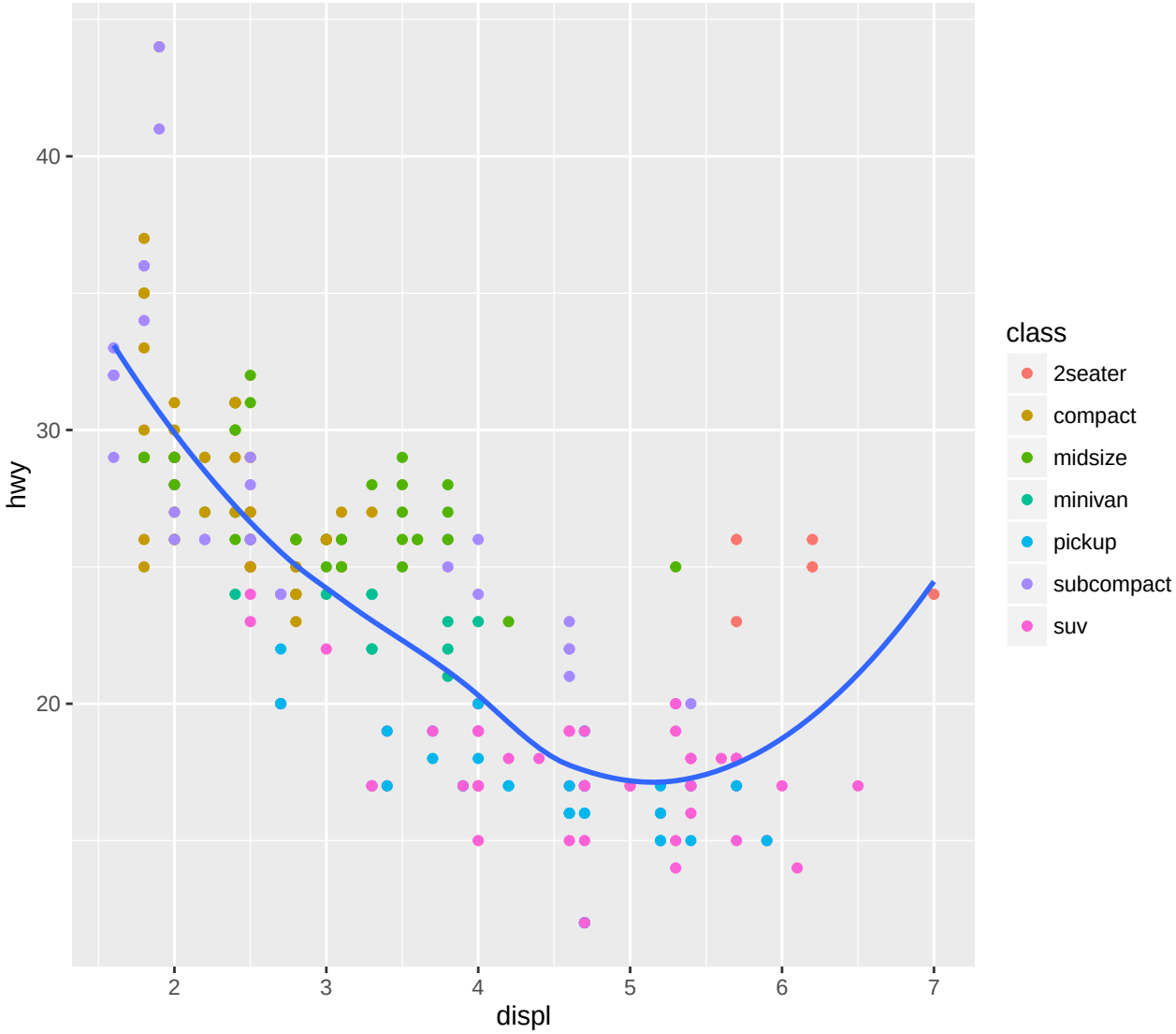
Operation	Estetik layer	Hasil
Add	aes(colour = cyl)	aes(mpg, wt, colour = cyl)
Override	aes(y = disp)	aes(mpg, disp)
Remove	aes(y = NULL)	aes(mpg)

Kalau kita hanya punya satu layer dalam plot, cara kita mengatur estetik tidak akan berpengaruh. Namun, pembedaan diperlukan saat kita menambahkan layer tambahan. Dua plot berikut fokus kepada dua aspek yang berbeda dari data:

```
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point() +
  geom_smooth(se = FALSE)
```




```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(colour = class)) +
  geom_smooth(se = FALSE)
```

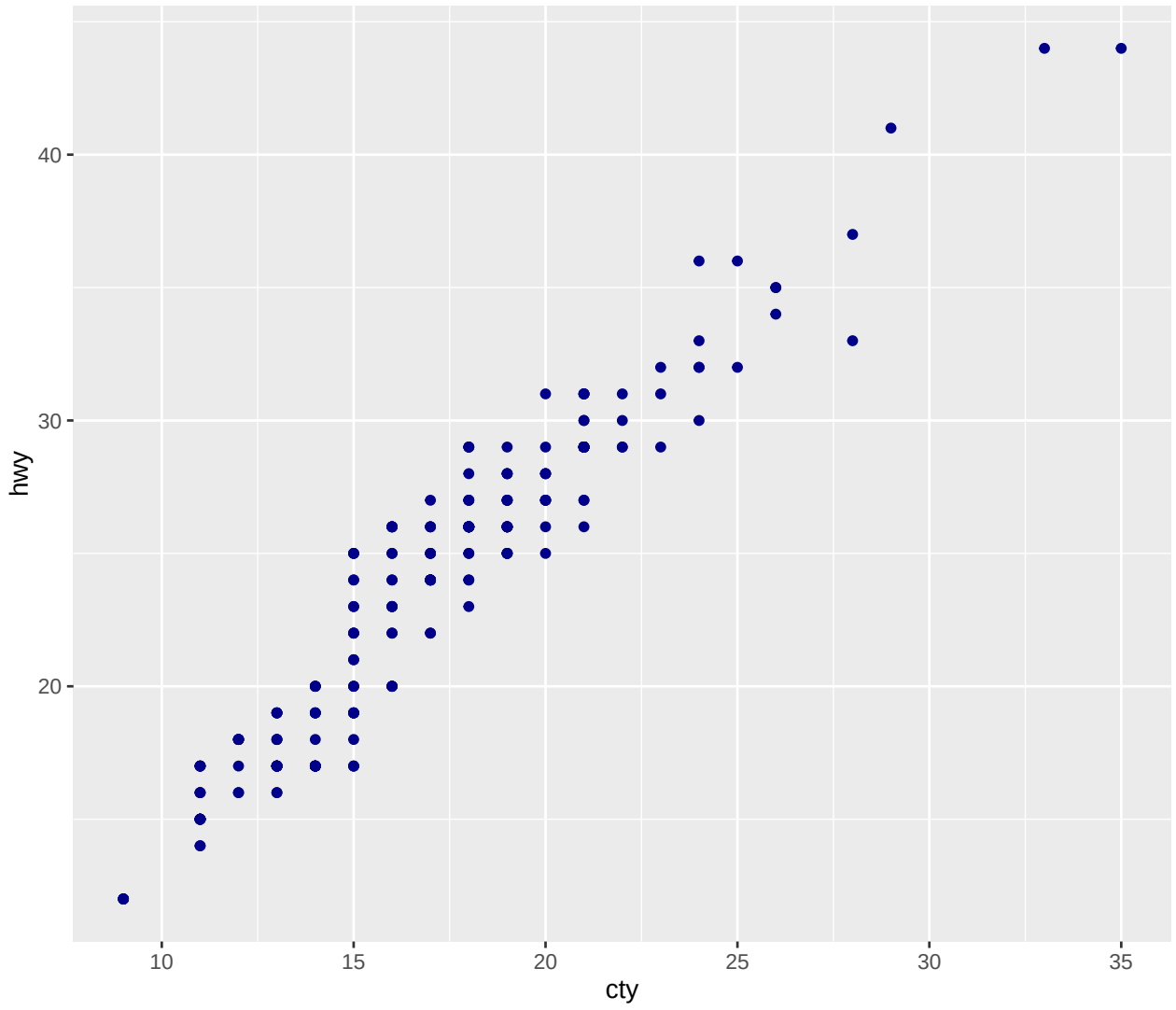


8 Setting vs mapping

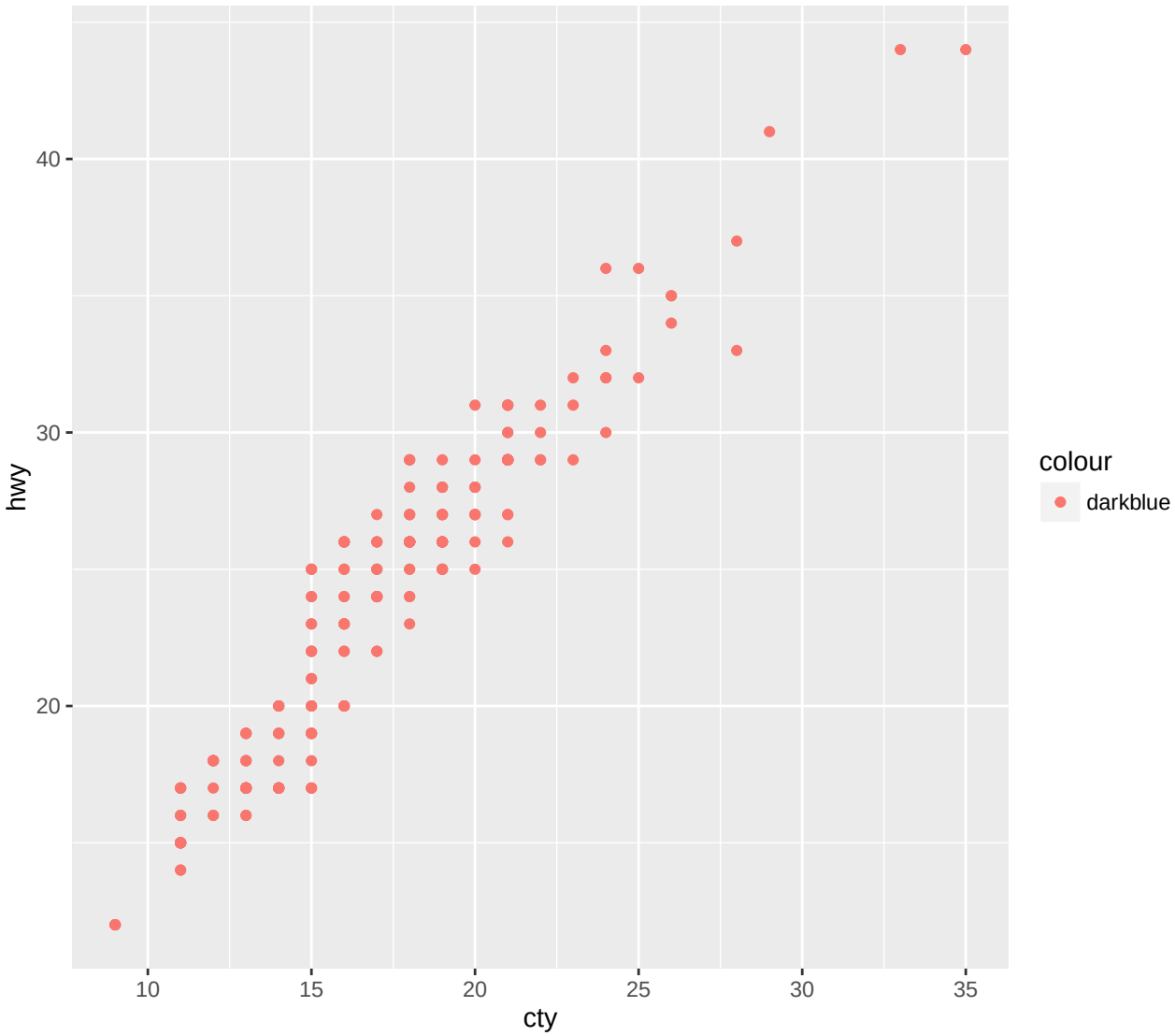
Selain mengatur properti estetik menjadi variabel, kita juga dapat mengaturnya sebagai nilai tunggal sebagai parameter layer. Kita memetakan estetik ke dalam variabel (misal: aes(colour = cut)) atau mengaturnya sebagai konstanta (misal: colour = "red").

Plot berikut ini dibuat dengan kode yang mirip, tapi memiliki output yang berbeda. Plot kedua memetakan warna menjadi 'darkblue'. Ini akan membuat variabel baru mengandung hanya nilai warna 'darkblue' dan kemudian membuat gradasi warna (colour scale). Karena nilai ini bersifat diskrit, maka warna akan diatur dengan gradasi yang seimbang, dan karena hanya ada nilai tunggal, maka awarna ini adalah merah muda 'pinkish'.

```
ggplot(mpg, aes(cty, hwy)) +
  geom_point(colour = "darkblue")
```

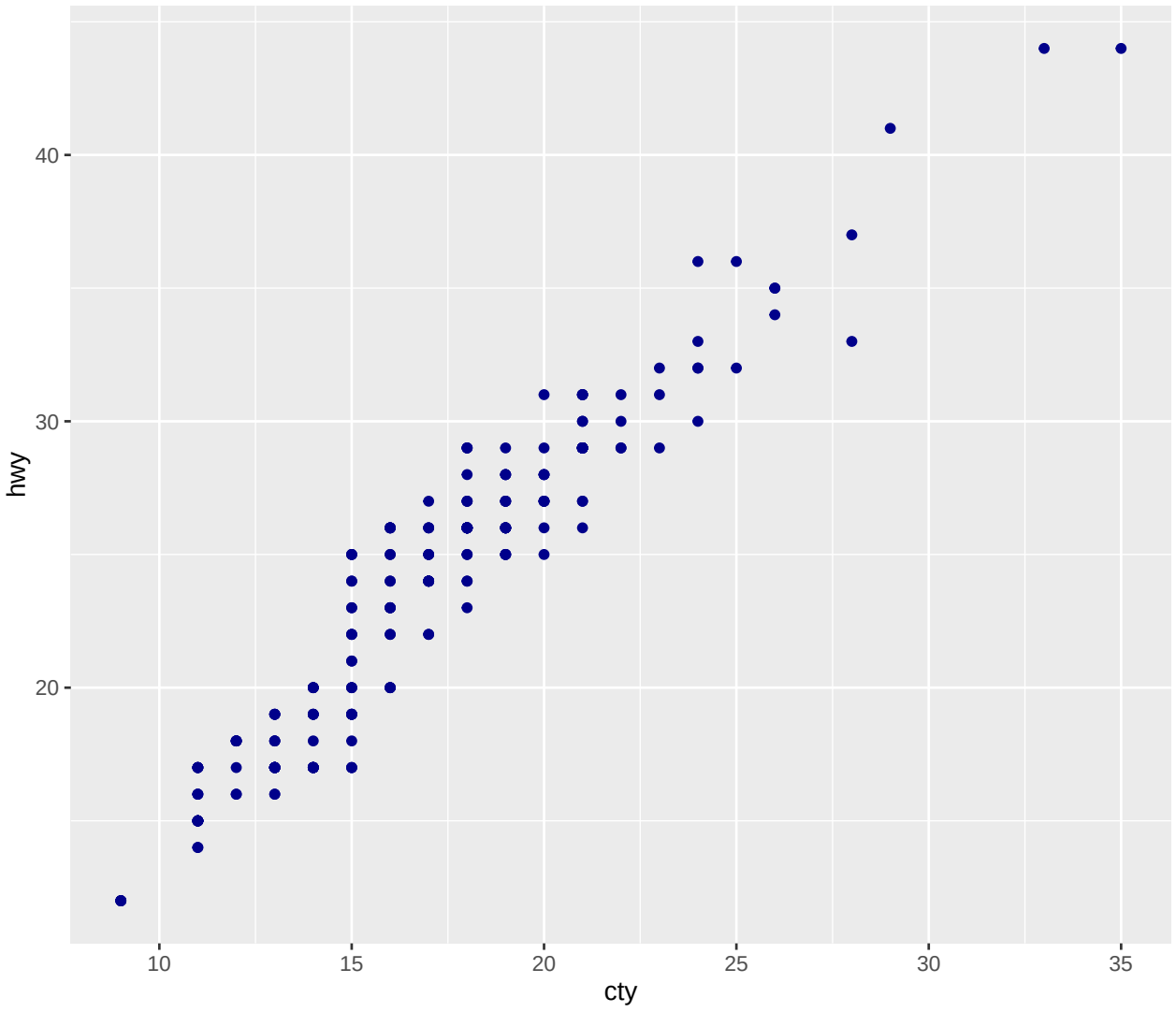



```
ggplot(mpg, aes(cty, hwy)) +
  geom_point(aes(colour = "darkblue"))
```



Pilihan yang ketiga adalah memetakan nilainya untuk mengubah skala warna yang baku:

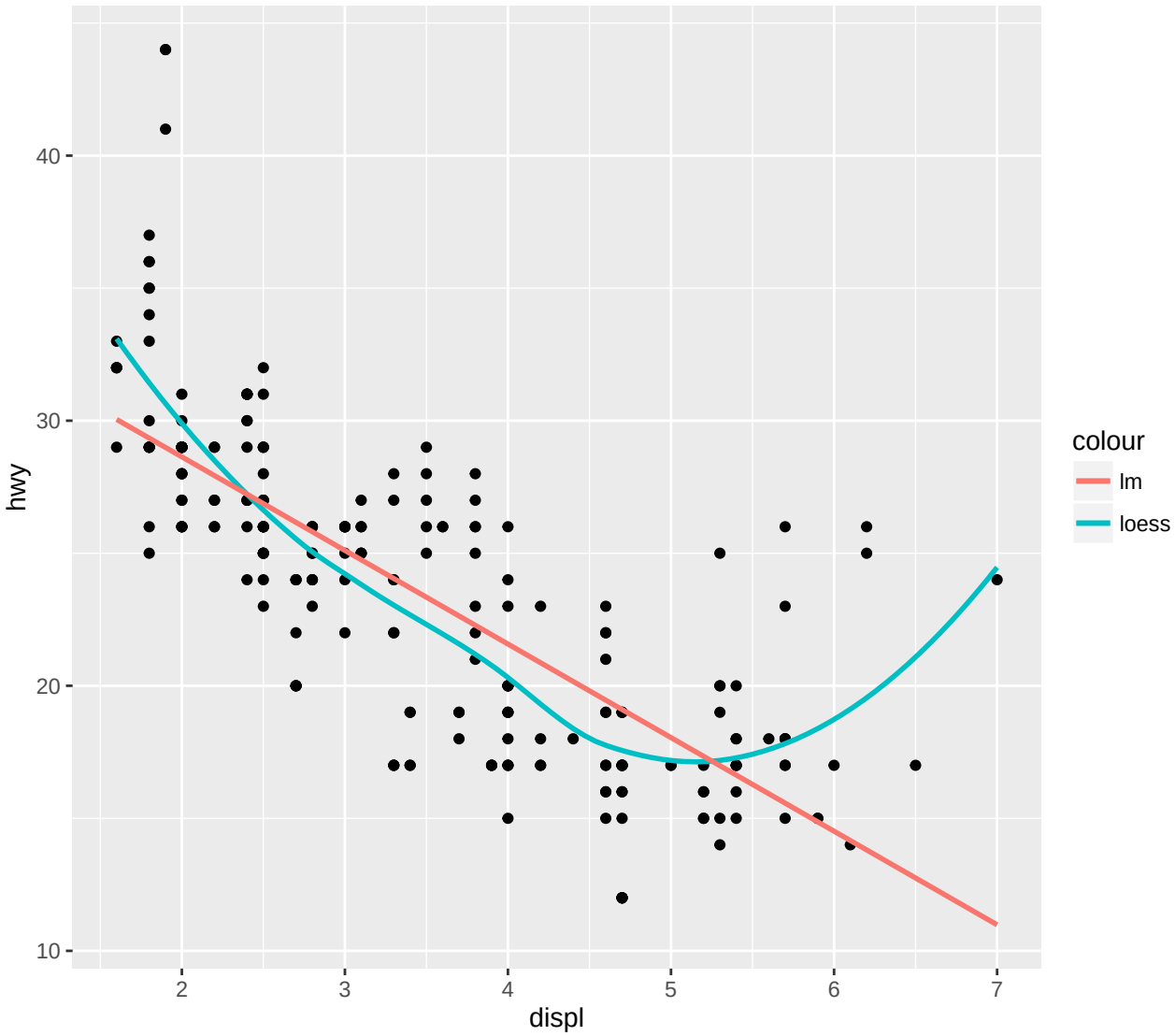
```
ggplot(mpg, aes(cty, hwy)) +
  geom_point(aes(colour = "darkblue")) +
  scale_colour_identity()
```



Cara ini akan tepat kalau kita punya sebuah kolom yang telah berisi warna. Kita akan mempelajarinya di bagian *identity scale*.

Terkadang akan cocok untuk memetakan estetik dengan nilai konstan. Sebagai contoh, bila kita ingin menampilkan beberapa dengan nilai parameter bervariasi, kita dapat memberi nama untuk masing-masing layer:

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth(aes(colour = "loess"), method = "loess", se = FALSE) +
  geom_smooth(aes(colour = "lm"), method = "lm", se = FALSE)
```

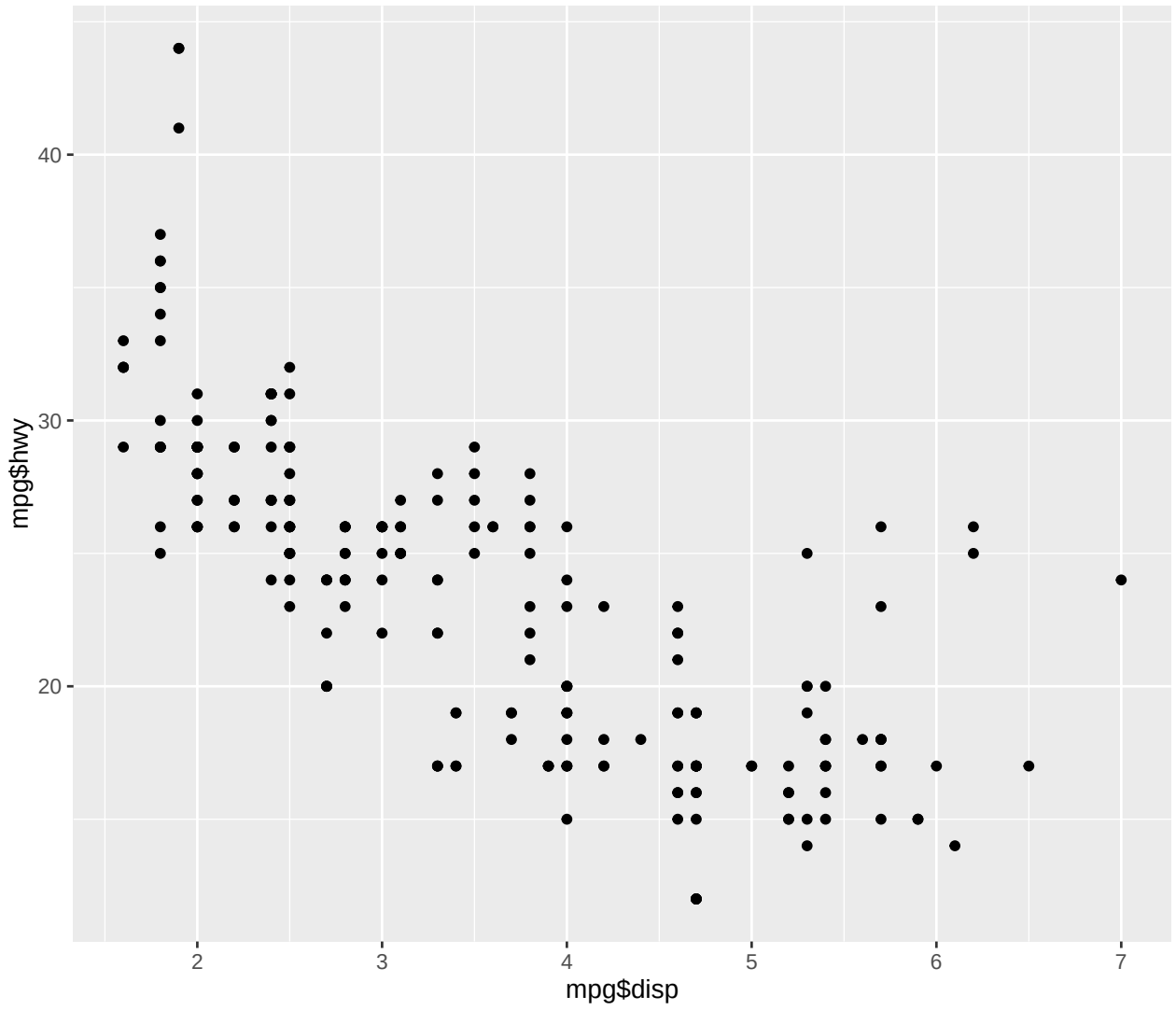


Legenda yang baku biasanya kurang informatif, tapi kita dapat dengan mudah mengaturnya bila telah mempelajari bagian *legends and axes*.

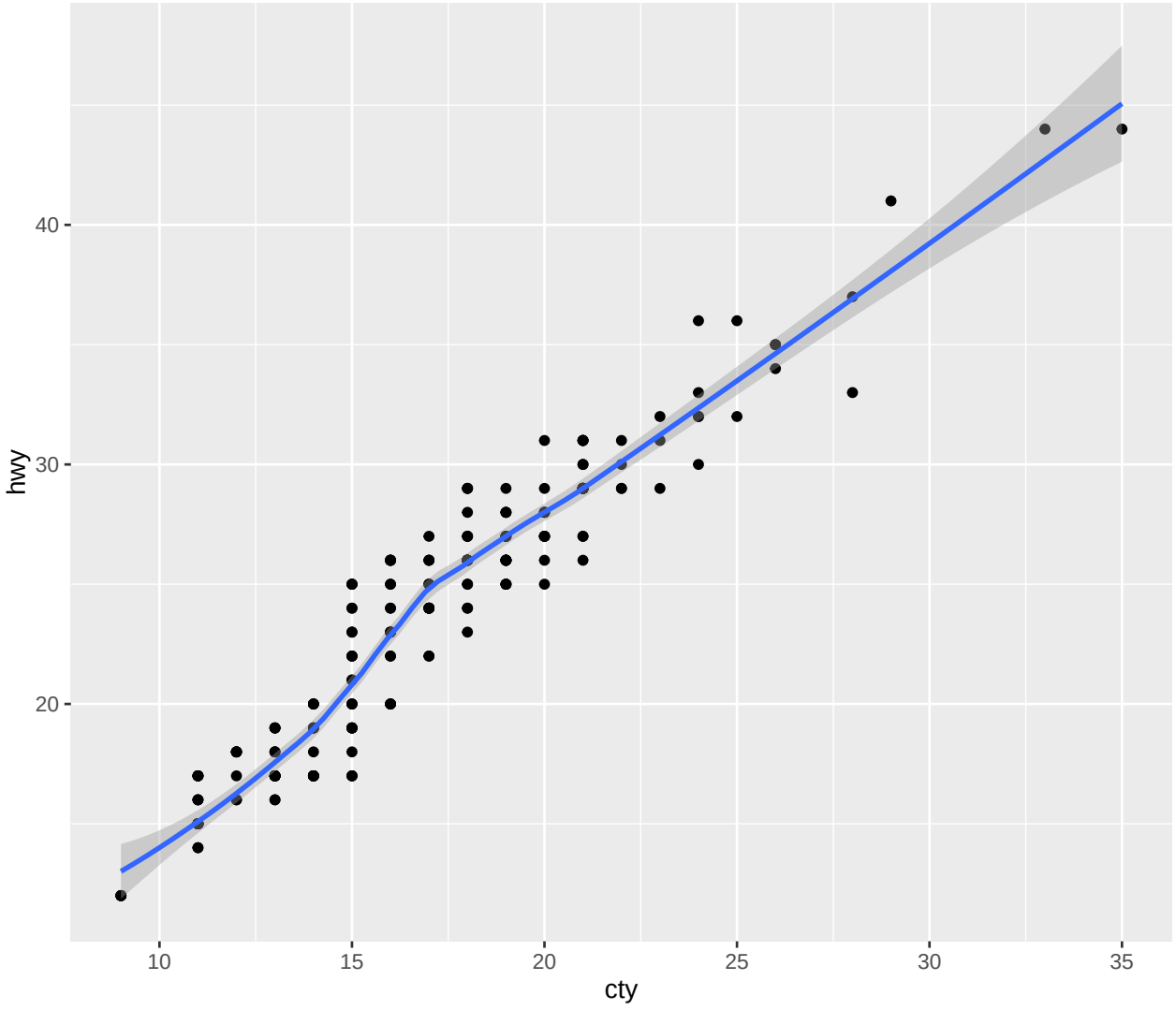
8.1 Latihan

1. Sederhanakan spesifikasi plot di bawah ini:

```
ggplot(mpg) +
  geom_point(aes(mpg$disp, mpg$hwy))
```

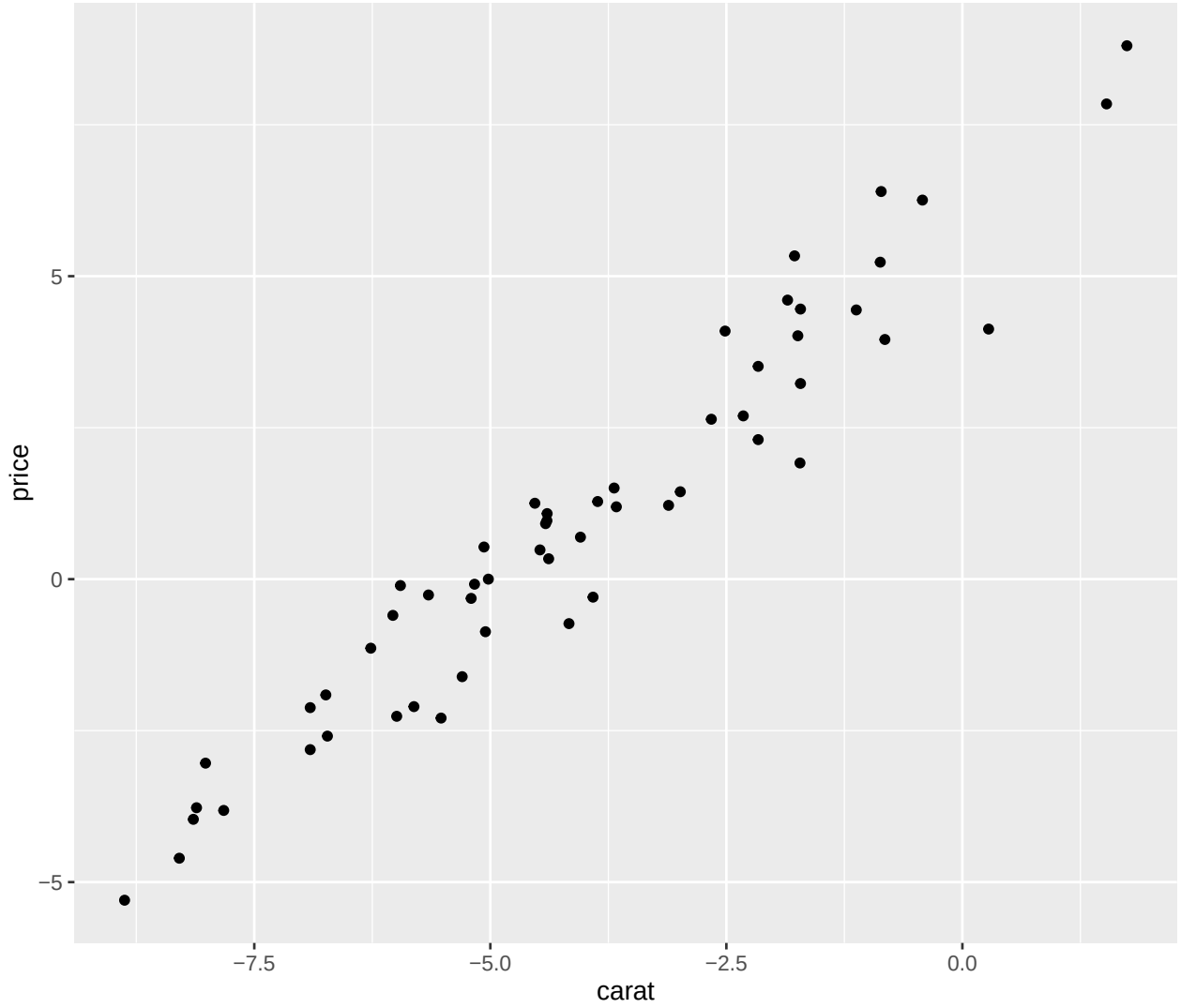



```
ggplot() +
  geom_point(mapping = aes(y = hwy, x = cty), data = mpg) +
  geom_smooth(data = mpg, mapping = aes(cty, hwy))
```



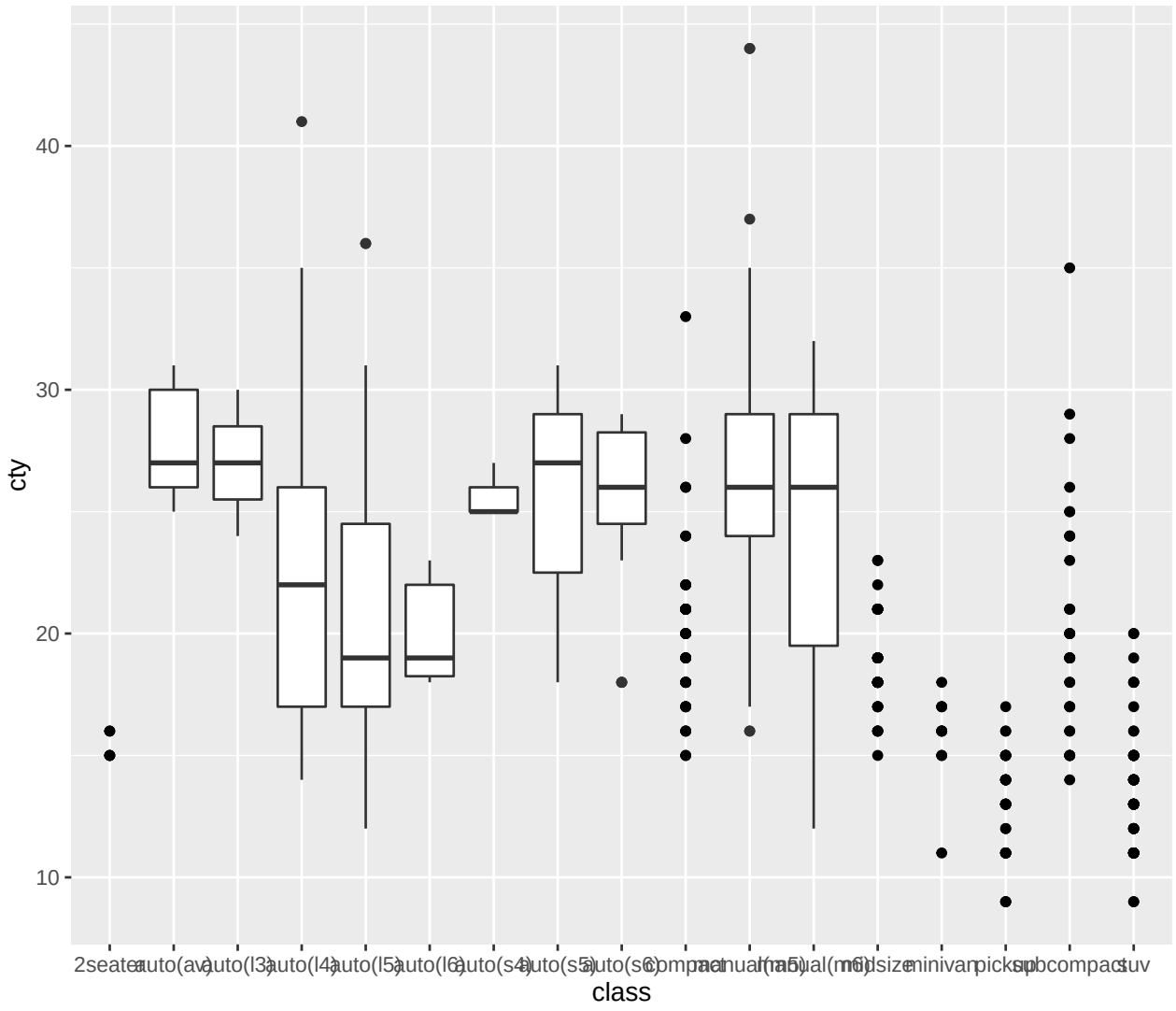

```
ggplot(diamonds, aes(carat, price)) +
  geom_point(aes(log(brainwt), log(bodywt)), data = msleep)
```

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



2. Apa yang dihasilkan oleh kode berikut ini? Apakah ia dapat bekerja? Apakah masuk akal? Mengapa/mengapa tidak?

```
ggplot(mpg) +
  geom_point(aes(class, cty)) +
  geom_boxplot(aes(trans, hwy))
```



9 Geoms

Obyek geometrik, atau `geoms` singkatnya, adalah yang melakukan rendering layer. Layer ini mengendalikan tipe plot yang dibuat. Sebagai contoh, menggunakan geom titik (*point geom*) akan membuat *scatterplot*, bila menggunakan line geom akan membuat plot garis (*line plot*).

Beberapa jenis geom adalah sbb:

- `geom_blank()`: display nothing. Most useful for adjusting axes limits using data.
- `geom_point()`: points.
- `geom_path()`: paths.
- `geom_ribbon()`: ribbons, a path with vertical thickness.
- `geom_segment()`: a line segment, specified by start and end position.
- `geom_rect()`: rectangles.

- `geom_polyon()`: filled polygons.
- `geom_text()`: text.

Plot untuk variabel tunggal:

- Diskrit (*Discrete*):
  - geom_bar(): display distribution of discrete variable.
- Kontinyu (*Continuous*):
  - geom_histogram(): bin and count continuous variable, display with bars.
  - geom_density(): smoothed density estimate geom_dotplot(): stack individual points into a dot plot.
  - geom_freqpoly(): bin and count continuous variable, display with lines.

Plot untuk variabel ganda (*Two variables*):

Kedua variabel kontinyu (*Both continuous*):

- geom_point(): scatterplot.
- geom_quantile(): smoothed quantile regression.
- geom_rug(): marginal rug plots.
- geom_smooth(): smoothed line of best fit.
- geom_text(): text labels.

Memperlihatkan distribusi (*Show distribution*):

- geom_bin2d(): bin into rectangles and count.
- geom_density2d(): smoothed 2d density estimate.
- geom_hex(): bin into hexagons and count.

Minimum satu variabel diskrit (*At least one discrete*):

- geom_count(): count number of point at distinct locations
- geom_jitter(): randomly jitter overlapping points.

Satu variabel kontinyu, dan satu variabel lainnya diskrit (*One continuous, one discrite*):

- “‘geom_bar(stat = “identity”): a bar chart of precomputed summaries
- “‘geom_boxplot(): boxplots.
- “‘geom_dotplot(): carefully adjust location of overlapping points.
- “‘geom_violin(): show density of values in each group.

Satu variabel waktu, satu variabel kontinyu (*One time, one continuous*):

- `geom_area()`: area plot.
- `geom_line()`: line plot.
- `geom_step()`: step plot.

Menampilkan error (*Display error*):

- `geom_crossbar()`: vertical bar with center. geom_errorbar(): error bars.
- `geom_linerange()`: vertical line.
- `geom_pointrange()`: vertical line with center.

Spasial (*Spatial*)

- `geom_map()`: fast version of geom_polygon() for map data.

Tiga variabel (*Three variables*):

- `geom_contour()`: contours.
- `geom_tile()`: tile the plane with rectangles.
- `geom_raster()`: fast version of geom_tile() for equal sized tiles.

Masing-masing geom memiliki satu set estetik, beberapa diantaranya harus dinyatakan. Sebagai contoh, point geoms membutuhkan posisi x dan y, dan mengerti perintah estetik untuk warna, ukuran, dan bentuk (*colour, size and shape aesthetics*). Sebut plot balok (*bar plot*) membutuhkan *height* (ymax), dan mengerti perintah untuk ketebalan balok, warna garis batas, dan warna balok (*width, border colour and fill colour*). Tiap geom punya daftar estetik di dokumen penjelasan (*documentation*).

Beberapa geoms berbeda parameternya. Misal, kita dapat menggambarkan bujursangkar dalam tiga cara:

- Dengan menggunakan perintah `geom_tile()` lokasi x dan y (*location* x and y) dan dimensi lebar dan tinggi (*dimensions* width and height).
- Dengan menggunakan perintah `giving geom_rect()` batas atas ymax (*top* ymax), batas bawah ymin (*bottom* ymin), batas kiri xmin (*left* xmin) dan batas kanan xmax (*right* xmax).
- Dengan menggunakan perintah `giving geom_polygon()`, sebagai sebuah data frame empat baris dengan nilai posisi x dan y di setiap sudutnya.

Geoms lainnya yang berkaitan adalah:

- `geom_segment()`, dan `geom_line()`
- `geom_area()` dan `geom_ribbon()`.

Memilih parameter plot yang paling tepat untuk data kita akan memudahkan menganalisis plot.

9.1 Latihan

1. Unduh dan cetak ggplot2 cheatsheet yang praktis sebagai referensi penggunaan geoms.
2. Lihat dokumentasi untuk geoms. Estetik mana yang mereka gunakan? Bagaimana meringkasnya?
3. Bagaimana cara terbaik untuk geom yang tidak dikenal? Buat daftar tiga sumber yang membantu anda memulai.
4. Identifikasi geom yang digunakan untuk tiap plot berikut ini.

!(Ex3.png)

!(Ex4.png)

!(Ex5.png)

5. Untuk masing-masing problem di bawah ini, usulkan geom yang tepat:

- Menampilkan variabel berubah terhadap waktu.
- Menampilkan distribusi rinci variabel tunggal.
- Memfokuskan trend umum suatu dataset yang besar.
- Membuat peta.
- Membuat label titik anomali (*outlier*).

10 Stats

Transformasi statistik, akan mentransformasi data, umumnya dengan membuat ringkasan dengan teknik tertentu. Sebagai contoh fungsi stat yang berguna adalah fungsi *smoother*, yang menghitung rata-rata smoothing dari y, dan kondisional untuk x. Tanpa sadar, kita sebenarnya telah menggunakan berbagai fungsi stat ggplot2, hanya saja fungsi berjalan di belakang layar untuk banyak fungsi geoms yang penting:

- `stat_bin()`: geom_bar(),geom_freqpoly(),geom_histogram()“‘
- `stat_bin2d()`: `geom_bin2d()`
- `stat_bindot()`: `geom_dotplot()`
- `stat_binhex()`: `geom_hex()`
- `stat_boxplot()`: `geom_boxplot()`
- `stat_contour()`: `geom_contour()`
- `stat_quantile()`: `geom_quantile()`
- `stat_smooth()`: `geom_smooth()`
- `stat_sum()`: `geom_count()`

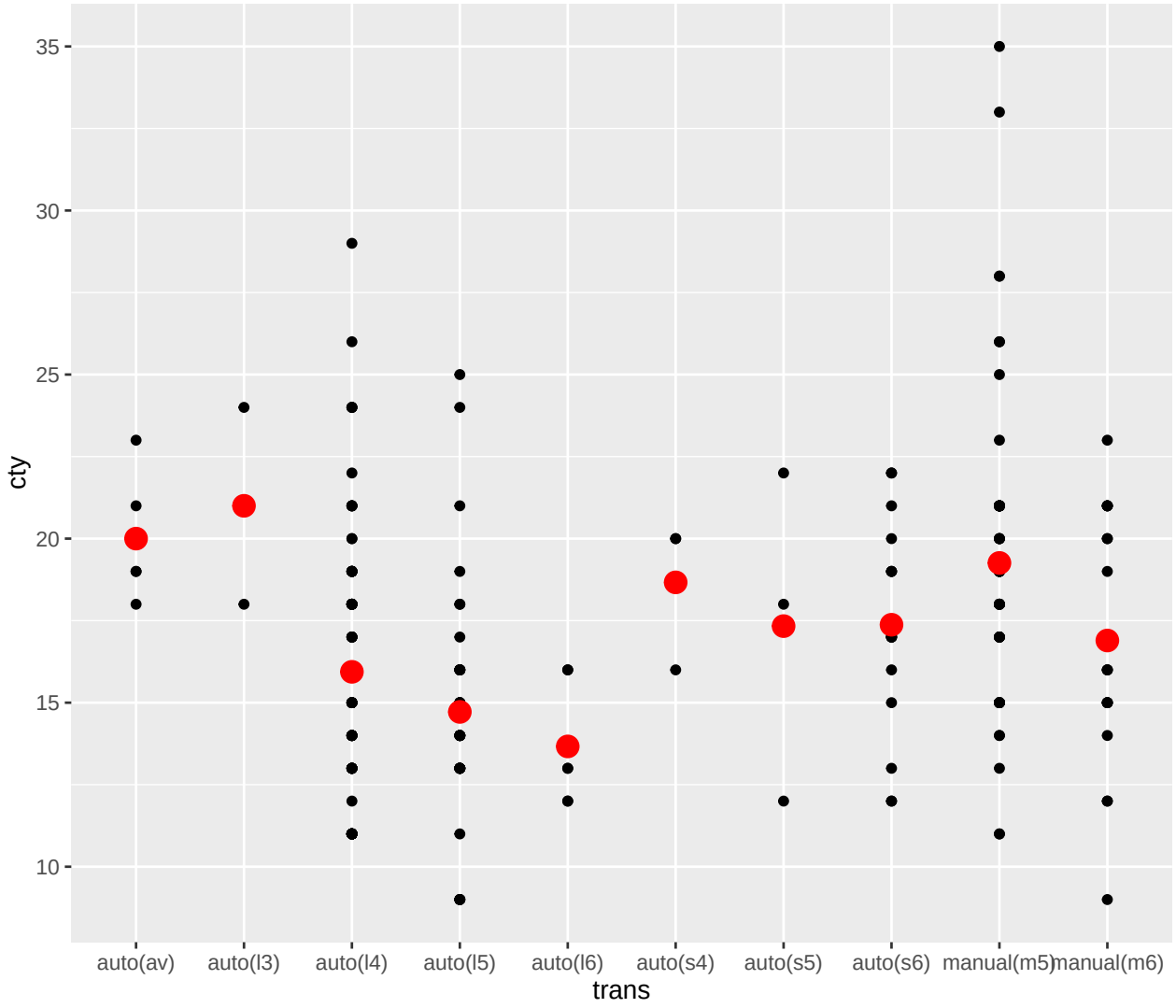
Kita jarang memanggil (*call*) fungsi-fungsi berikut secara langsung, tetapi mempelajarinya akan sangat bagus. Detil transformasi statistik yang dapat dihasilkan ada dalam dokumentasi.

Beberapa analisis stat yang dapat dibuat dengan fungsi `geom_` adalah:

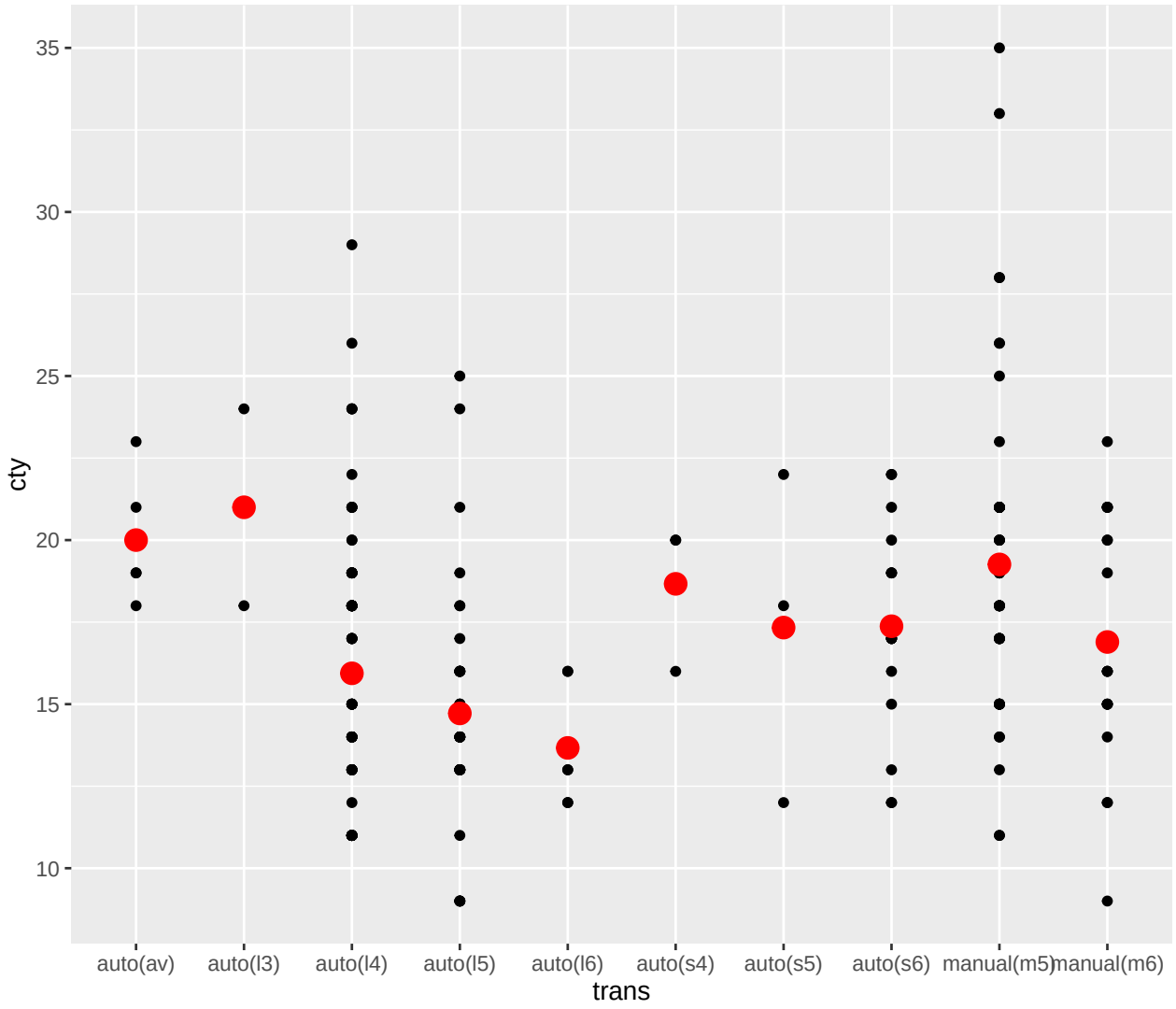
- `stat_ecdf()`: compute a empirical cumulative distribution plot.
- `stat_function()`: compute y values from a function of x values.
- `stat_summary()`: summarise y values at distinct x values.
- `stat_summary2d()`, `stat_summary_hex()`: summarised binned values.
- `stat_qq()`: perform calculations for a quantile-qunatile plot.
- `stat_spoke()`: convert angle and radius to position.
- `stat_unique()`: remove duplicated rows.

Ada dua cara untuk menggunakan fungsi-fungsi ini. Kita dapat menambahkan fungsi `stat_()` dan menindih parameter geom baku, atau menambahkan fungsi `geom_() function` dan menindih parameter stat baku:

```
ggplot(mpg, aes(trans, cty)) +
  geom_point() +
  stat_summary(geom = "point", fun.y = "mean", colour = "red", size = 4)
```



```
ggplot(mpg, aes(trans, cty)) +
  geom_point() +
  geom_point(stat = "summary", fun.y = "mean", colour = "red", size = 4)
```



Menurut saya, lebih baik menggunakan kode yang kedua karena kita aka ringkasan, bukan data mentah (*raw data*).

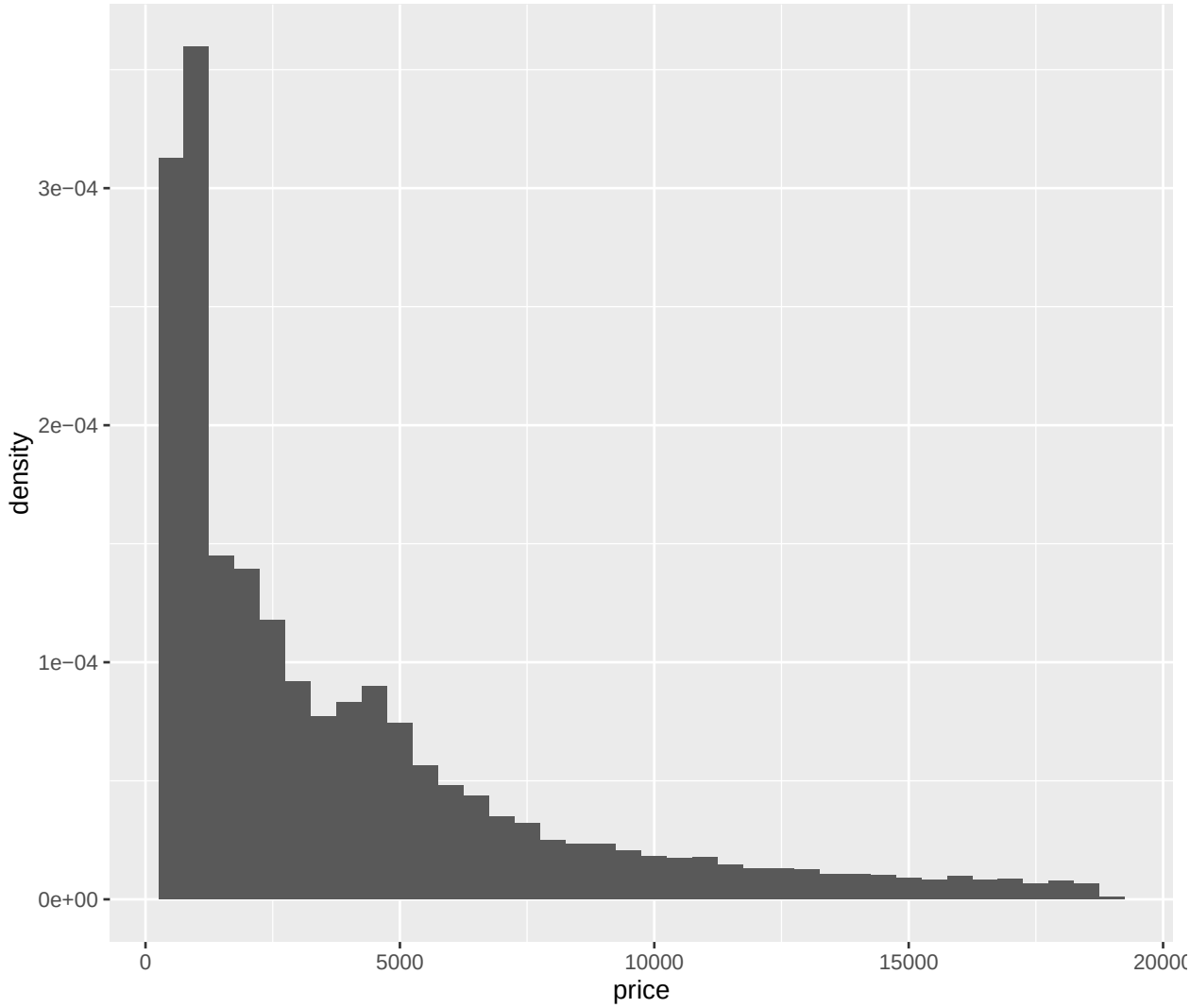
10.1 Variabel yang dibuat oleh fungsi (*Generated variables*)

Secara internal, sebuah fungsi stat menggunakan sebuah data frame sebagai input dan menghasilkan sebuah data frame sebagai output, dan sehingga sebuah stat menambahkan variabel baru ke dalam dataset aslinya. Dimungkinkan untuk memetakan estetik untuk variabel-variabel baru ini. Sebagai contoh, `stat_bin`, fungsi statistik untuk membuat histogram, yang akan membuat beberapa variable berikut:

- `count`, the number of observations in each bin
- `density`, the density of observations in each bin (percentage of total / bar width)
- `x`, the centre of the bin

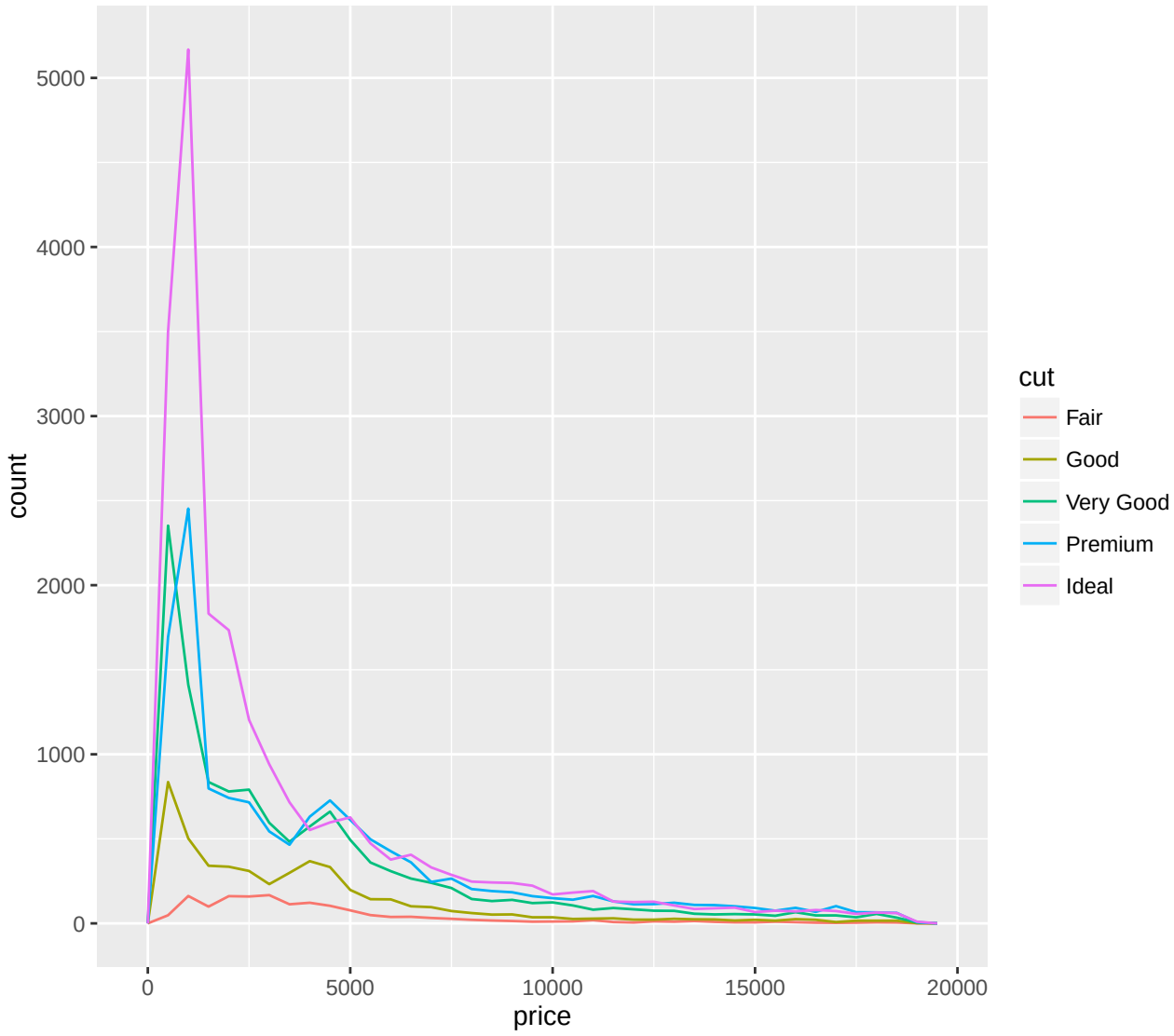
Variabel yang dibuat oleh suatu fungsi dan disimpan dalam dataset orisinal juga dapat digunakan. Misalnya, fungsi geom histogram baku menggunakan jumlah observasi sebagai ketinggian balok (variabel *count*), tapi jikan kita lebih suka histogram yang lebih tradisional, kita dapat menggunakan variabel *density*. Contoh berikut ini memperlihatkan density histogram dari nilai `carat` dataset intan (diamond dataset).

```
ggplot(diamonds, aes(price)) +
  geom_histogram(aes(y = ..density..), binwidth = 500)
```

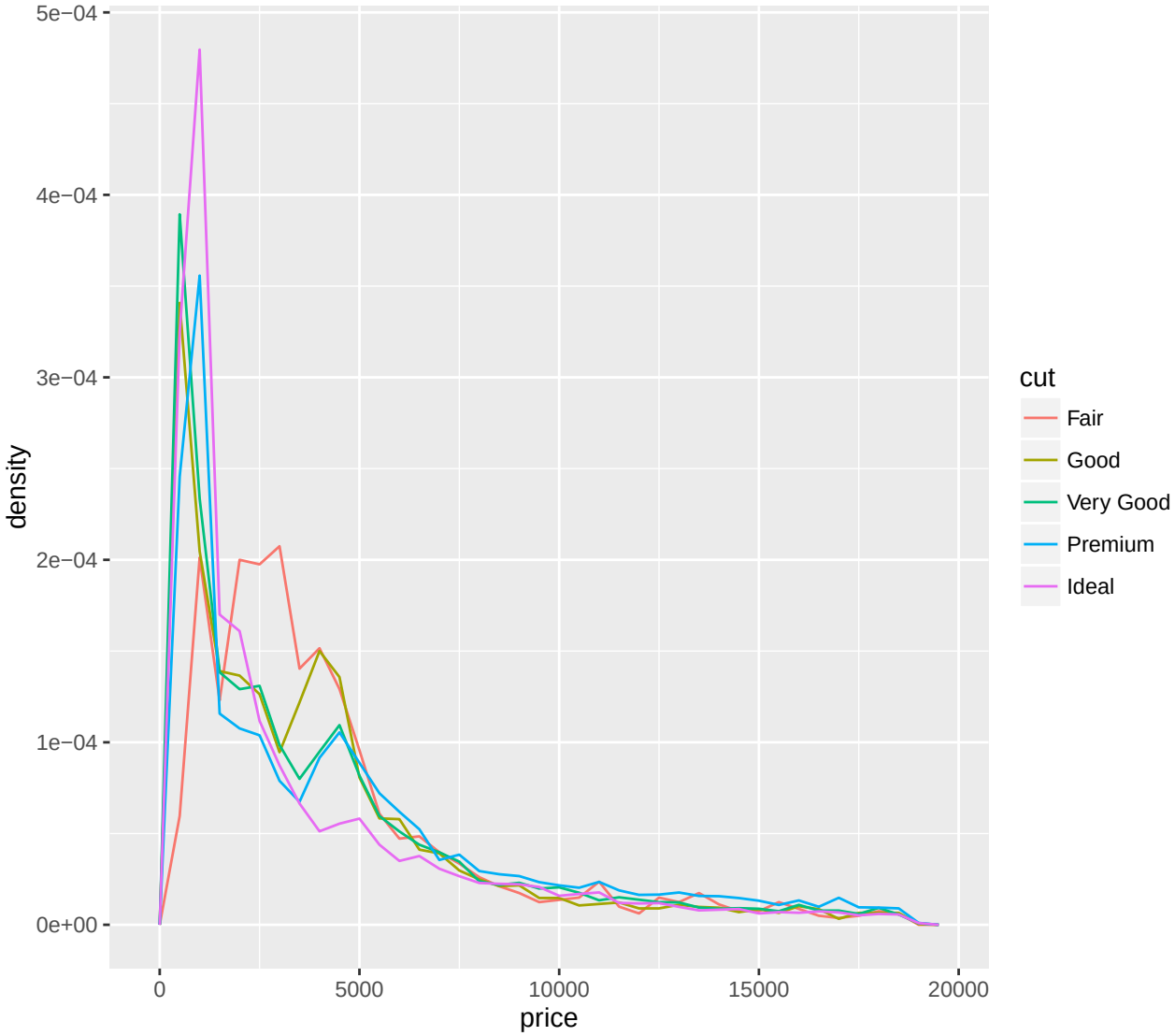


Teknik ini akan bermanfaat bila kita akan membandingkan distribusi beberapa grup yang memiliki ukuran sampel berbeda-beda. Contoh, akan sulit membandingkan distribusi harga (*price*) berdasarkan nilai *cut* karena beberapa grup memiliki perbedaan sangat kecil. Lebih mudah membandingkan bila kita menstandardisasi masing-masing grup untuk area yang sama:

```
ggplot(diamonds, aes(price, colour = cut)) +
  geom_freqpoly(binwidth = 500)
```




```
ggplot(diamonds, aes(price, colour = cut)) +
  geom_freqpoly(aes(y = ..density..), binwidth = 500)
```



Hasil plot ini mengejutkan: kualitas intan yang rendah terlihat lebih mahal secara rata-rata. Kita akan kembali ke masalah ini dalam bagian *removing trend*.

Nama-nama variabel yang terbentuk dari suatu fungsi harus diawali dan diakhiri dengan `.`, saat digunakan. Hal ini mencegah agar tidak tertukar dengan variabel orisinal, dan agar pembaca atau analis berikutnya jelas bahwa variabel tersebut adalah hasil fungsi proses statistik. MasingEach statistic lists the variables that it creates in its documentation.

10.2 Latihan

1. Kode berikut ini membuat dataset yang sama dengan fungsi `stat_smooth()`. GUnakan geoms yang sesuai untuk meniru fungsi geom_smooth() baku.

```
mod <- loess(hwy ~ displ, data = mpg)
smoothed <- data.frame(displ = seq(1.6, 7, length = 50))
pred <- predict(mod, newdata = smoothed, se = TRUE)
smoothed$hwy <- pred$fit
smoothed$hwy_lwr <- pred$fit - 1.96 * pred$se.fit
smoothed$hwy_upr <- pred$fit + 1.96 * pred$se.fit
```

2. Fungsi stats apakah yang digunakan dalam plot berikut ini?

!(Ex2.png)

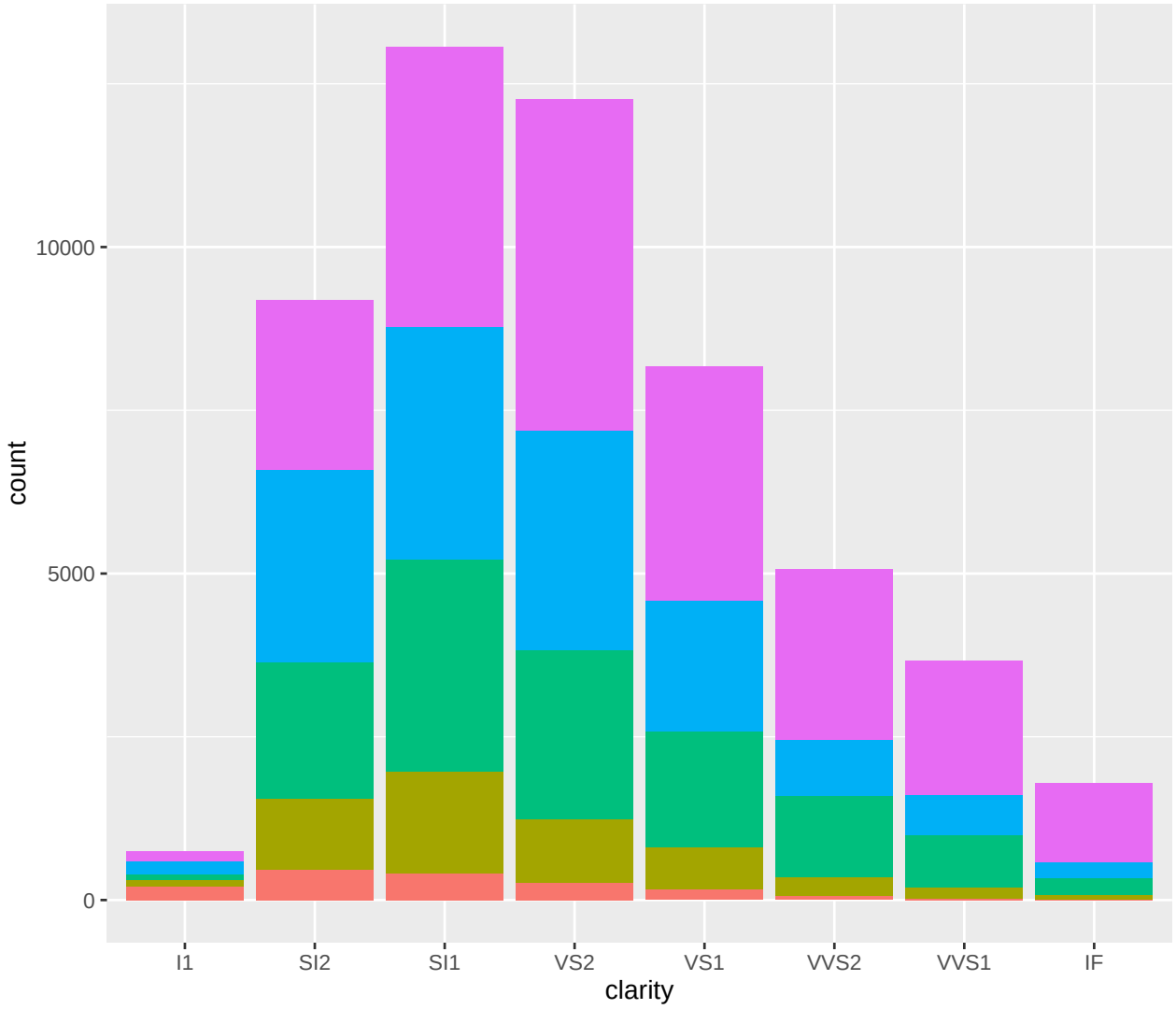
Baca dokumen pembantu untuk fungsi `stat_sum()` kemudian gunakan `geom_count()` untuk membuat plot yang memperlihatkan proporsi mobil yang memiliki kombinasi `drv` dan `trans` (dalam dataset `mtcars`).

11 Pengaturan posisi (*position adjustment*)

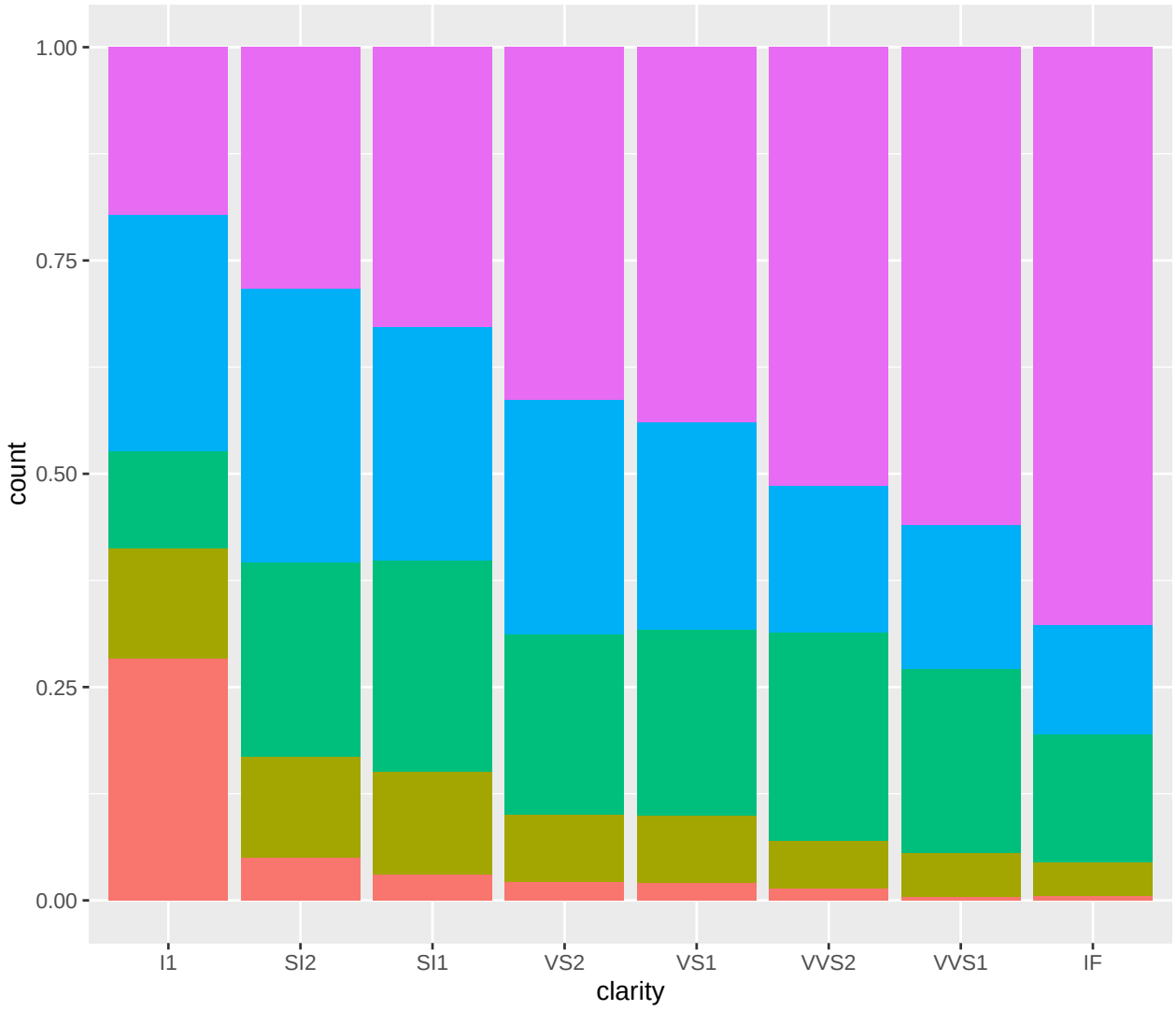
Pengaturan posisi mengatur posisi element dalam sebuah layer. Tiga jenis pengaturan dapat digunakan untuk plot balok (*bars plot*):

position_dodge(): place overlapping bars (or boxplots) side-by-side. position_stack(): stack overlapping bars (or areas) on top of each other. position_fill(): stack overlapping bars, scaling so the top is always at 1.

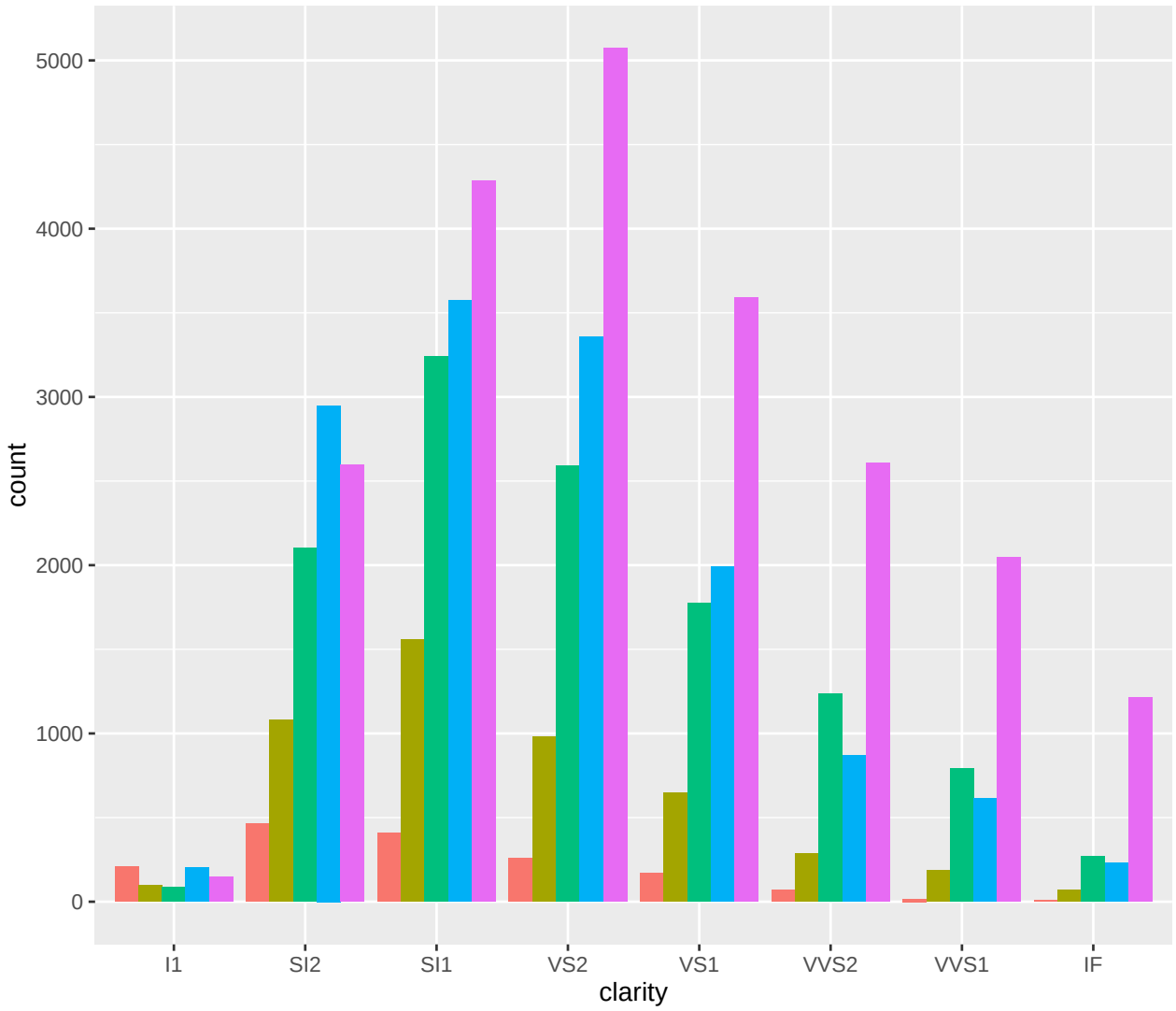
```
dplot <- ggplot(diamonds, aes(clarity, fill = cut)) +
  theme(legend.position = "none")
dplot + geom_bar()
```



```
dplot + geom_bar(position = "fill")
```



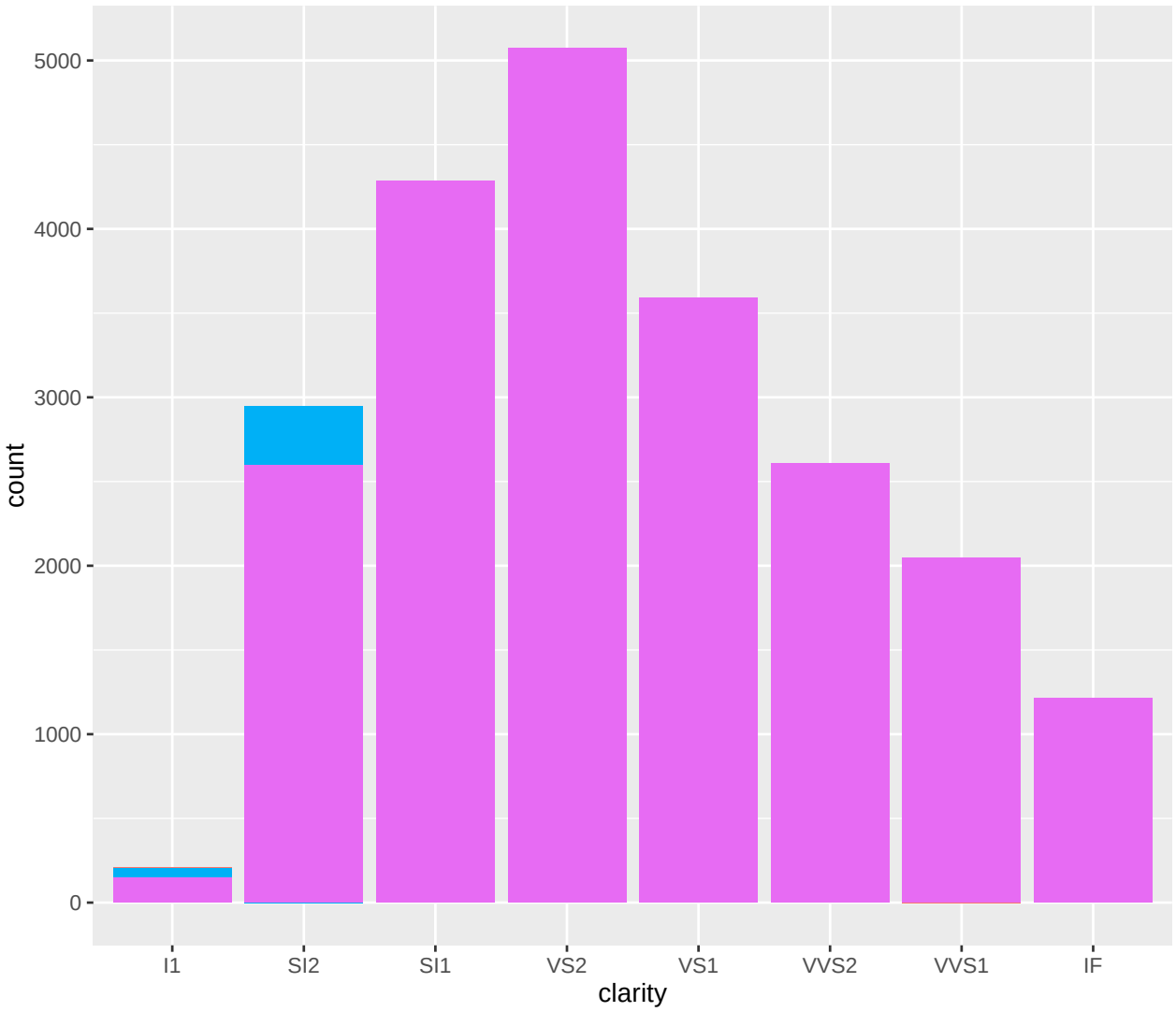

```
dplot + geom_bar(position = "dodge")
```



Stacking adalah pengaturan posisi yang baku untuk balok, dengan demikian `geom_bar()` akan ekivalen dengan `geom_bar(position = "stack")`. *Dodging* akan mirip dengan *faceting*, kelebihan dan kekurangannya dijelaskan dalam bagian *dodging vs. faceting*.

Juga ada pengaturan posisi yang tidak berpengaruh apa-apa: `position_identity()`. Fungsi tersebut tidak bekerja untuk balok, karena masing-masing balok menutupi balok yang ada di belakangnya. Poligon frekuensi adalah teknik yang lebih baik dalam kasus ini:

```
dplot + geom_bar(position = "identity")
```



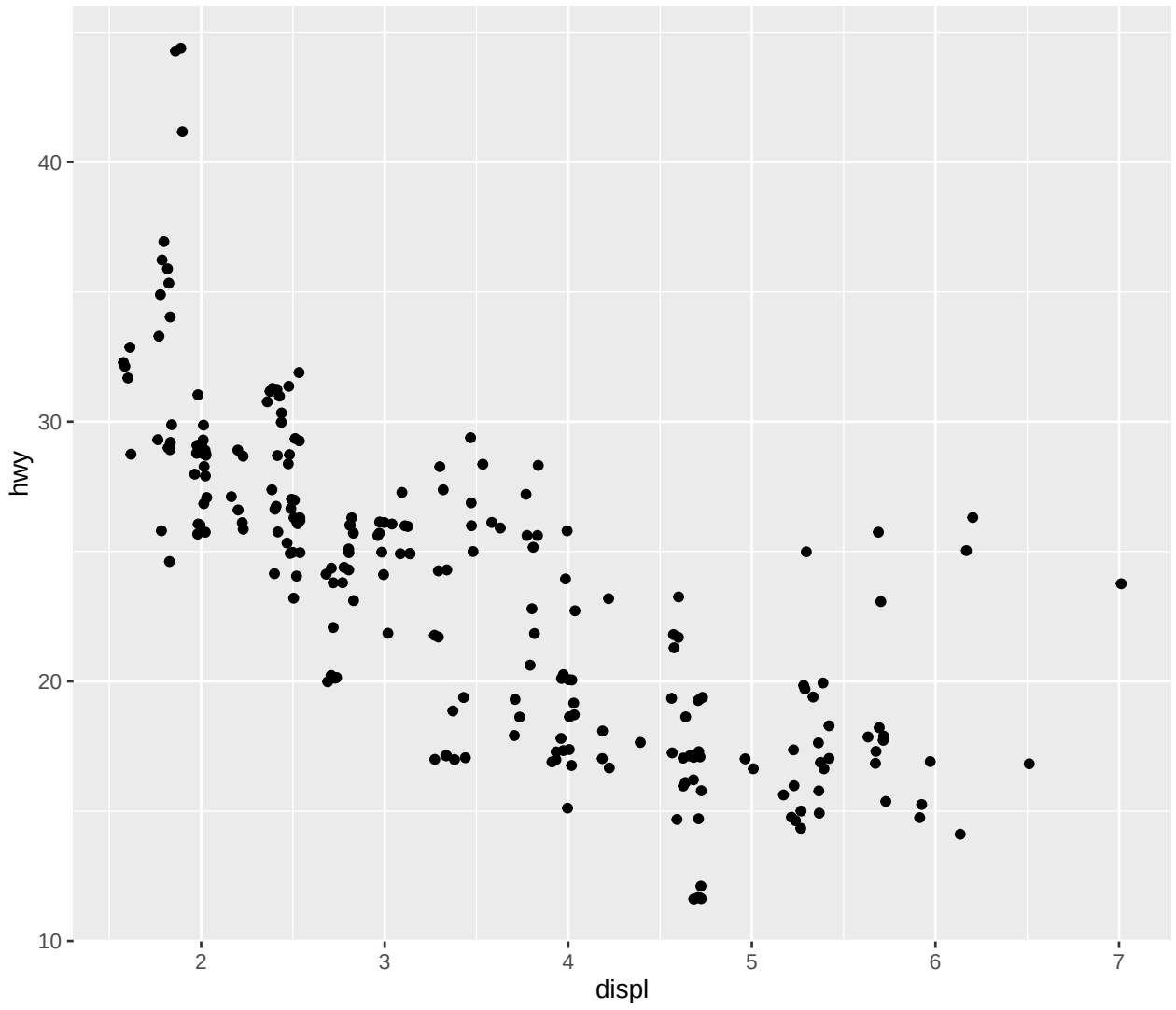

```
#ggplot(diamonds, aes(clarity, colour = cut)) +
#  geom_freqpoly(aes(group = cut)) +
#  theme(legend.position = "none")
```

Ada tiga pengaturan posisi yang sering digunakan untuk plot titik (*points plot*):

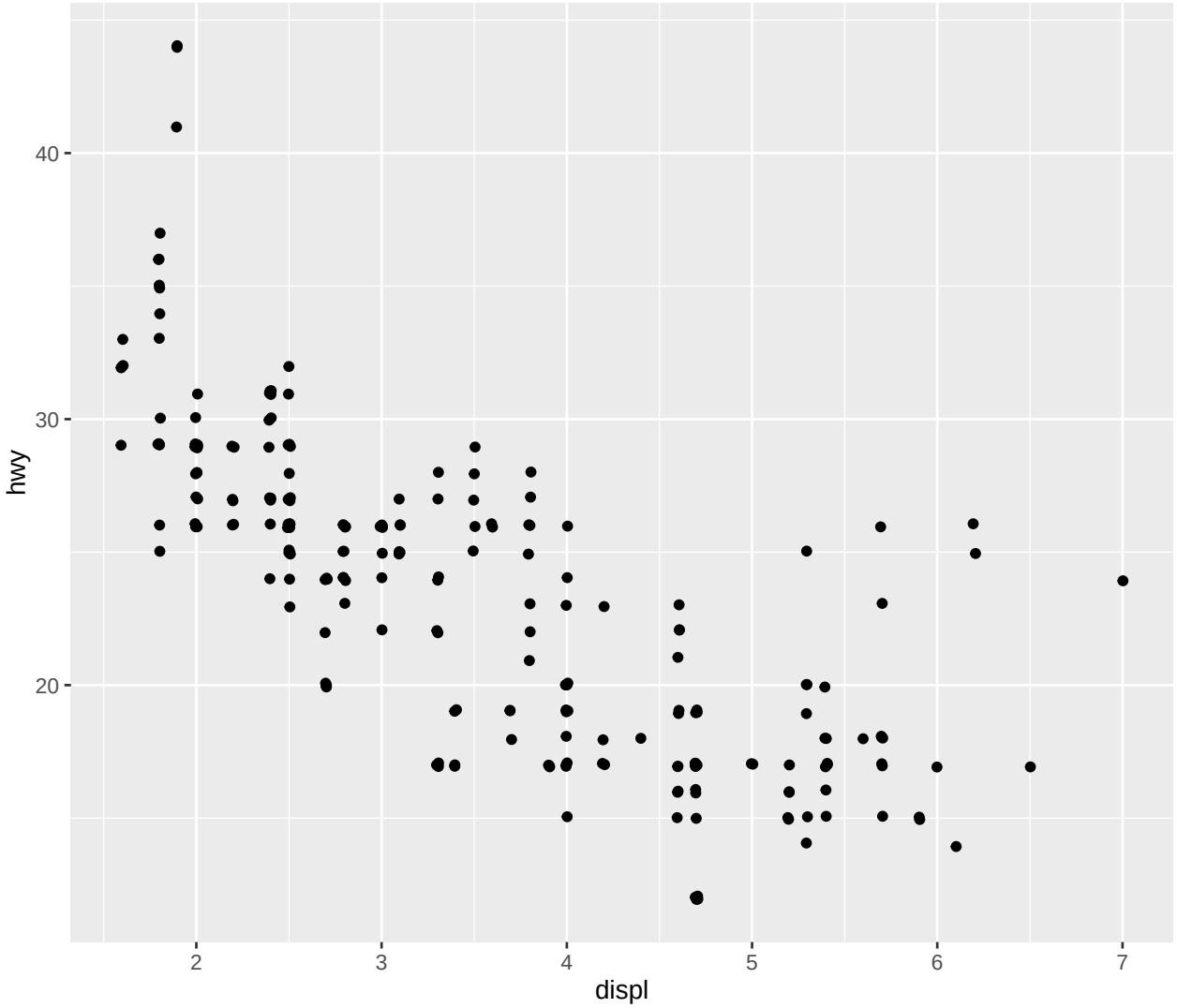
- `position_nudge()`: memindahkan titik dengan nilai perpindahan yang tetap.
- `position_jitter()`: memberikan *noise* acak untuk tiap titik.
- `position_jitterdodge()`: *dodge* titik dalam kelompok, kemudian menambahkan sedikit *noise* acak

Dapat dicatat bahwa cara kita memberikan parameter dalam pengaturan posisi berbeda untuk stats dan geoms. Kita dapat membuat obyek berisi pengaturan posisi, yang memberikan argumen tambahan dalam fungsi:

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(position = "jitter")
```




```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(position = position_jitter(width = 0.02, height = 0.2))
```



Ini lebih jelas, fungsi `geom_jitter()` memberikan jalan pintas yang lebih sesuai.

Pengaturan posisi biasanya digunakan untuk data diskrit. Data kontinyu umumnya tidak overlap dengan tepat, dan bila ini terjadi (karena rapatnya data), pengaturan kecil seperti *jittering* tidak mampu memperbaiki kondisi.

11.1 Latihan

1. Kapan kita perlu menggunakan `position_nudge()`? Baca dokumentasi.

2. Banyak pengaturan posisi yang hanya dapat digunakan bersama beberapa geoms. Contoh, kita tidak dapat menumpuk *boxplots* atau *errors bars*. Mengapa? Properti apa yang harus dinyatakan dalam geom agar dapat ditumpuk? Properti apa pula yang harus dinyatakan agar dapat di-*dodge* (*dodgeable*)?

3. Mengapa kita dapat menggunakan `geom_jitter()` bukan `geom_count()`? Apa kelebihan dan kekurangan masing-masing?

4. Kapan kita dapat menggunakan plot area yang bertumpuk (*stacked area plot*)? Apa saja kelebihan dan kekurangannya bila dibandingkan dengan plot garis (*line plot*)?

12 Tautan terkait

- Quick R
- Build a plot layer by layer
- Tutorial ggplot2 Harvard University
- ggplot2 presentation - Hadley Wickham
- Lee Mendelowitz blog on plotting using R's base graph functions