Modelos Computacionais em Economia

Marcleiton Morais

Universidade Federal do Tocantins (UFT)

1 de fevereiro de 2023

Noções de programação

O que é um programa?

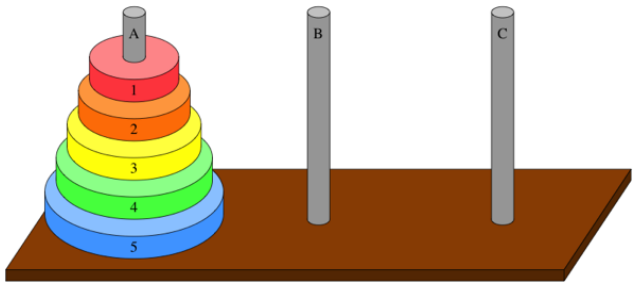
- Um programa é uma sequência de instruções que diz o que o computador deve fazer.
- Basicamente, essas instruções incluem:
  - Entrada e saída de dados;
  - Repetição de tarefas (*loop*, *for* ou *while*);
  - Execução condicional.

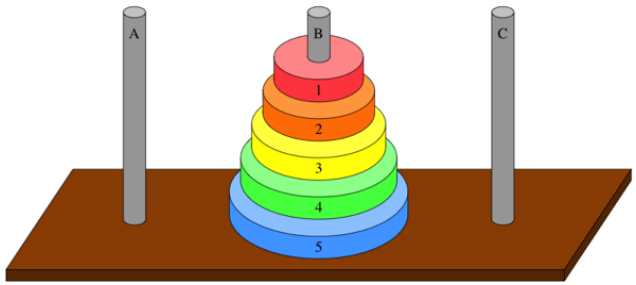
Hello World!?

Existe uma certa tradição do primeiro programa em uma nova linguagem ser o "**Hello World**".

O que é um algoritmo?

- É uma receita que especifica uma sequência exata de passos para resolver uma categoria de problemas.





O que é um algoritmo?

O que é um algoritmo interessante?

- Resolve um problema importante;
- É útil em muitas situações;
- É escalável;
- É belo.

O que é um algoritmo?

Exemplo de algoritmos:

- Indexação de páginas de internet (page rank);
- Criptografia;
- Algoritmos para correção de erros;
- Reconhecimento de padrões;
- Consistência de bases de dados;
- Computação humana.

Noções de programação

Valores e tipos

- Valores são sequências de letras ou números;
- Valores podem ser de vários tipos diferentes (numéricos, caracteres etc).

Variáveis

- Uma variável é um nome que se refere a um valor (*d<-5*);
- Operações comuns nesse contexto são definir uma variável e atribuir um valor a uma variável.

Operadores e expressões

- Operadores são símbolos que representam uma operação (*+,- etc*);
- Expressões são combinações de valores, variáveis e operadores (*count=count+1*).

Noções de programação

Funções

São sequências de operações que implementam computações.
Você deve abusar de funções, pois torna o seu programa mais claro, menor (pois remove código repetido), modular e bem desenhado.
Ex. *log10(x)*

Fluxo de Execução

É a ordem em que as operações são executadas em um programa.
Isso é importante, pois você só pode chamar uma variável ou função depois que ela for definida.

Escopo de variáveis

Variáveis podem ser locais ou globais. [*De uma forma geral, evitamos definir variáveis globais*]

Noções de programação

Condicionais

Permitem o controle do fluxo de execução dependendo dos resultados intermediários do seu programa.

Iteração

Uma das tarefas que um computador faz muito bem sem ficar chateado é fazer atividades repetidas. As duas estruturas mais comuns são:

- Estruturas do tipo **For**;
- Estruturas do tipo **While**.

Introduction to R (RStudio)

Agenda

- Comparison of R to its alternatives;
- Ressources for learning R;
- Installing R;
- An introductory R session.

What is R?

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.

Why R?

- I Most popular environment in statistics and machine learning communities.
- Open source, fast growing ecosystem.
- Packages for almost everything:
  - Data processing and cleaning
  - Data visualization
  - Interactive web-apps
  - Typesetting, writing articles and slides
  - The newest machine learning routines, etc
- Accomplishes the things you might be used to do doing in Stata (data processing, fitting standard models) and those you might be used to doing in Matlab (numerical programming).

Alternatives to R

- **Stata** (proprietary): Most popular statistical software in economics, easy to use for standard methods, not a good programming language.
- **Matlab** (proprietary): Numerical programming environment, matrix based. Programming in (base) R is quite similar to Matlab.
- **Python** (open): General purpose programming language, standard in industry, not targeted toward data analysis and statistics, but lots of development for machine learning. More overhead to write relative to R.
- **Julia** (open): New language for numerical programming, fast, increasingly popular in macro / for solving complicated structural models, not geared toward data analysis.

Resources for learning R

- **An Introduction to R**
  Complete introduction to base R. My recommended place to get started.
  ▸ Link ▸ Link ▸ Link
- **R for Data Science**
  Introduction to data analysis using R, focused on the tidyverse packages. If your goal is to find a substitute for Stata, start here.
  ▸ Link
- **Econometrics with r:** ▸ Link
- **Forecasting: Principles and Practice** ▸ Link
- **Advanced R**
  In-depth discussion of programming in R. Read later, if you want to become a good R programmer. ▸ Link

Resources for learning extensions to R

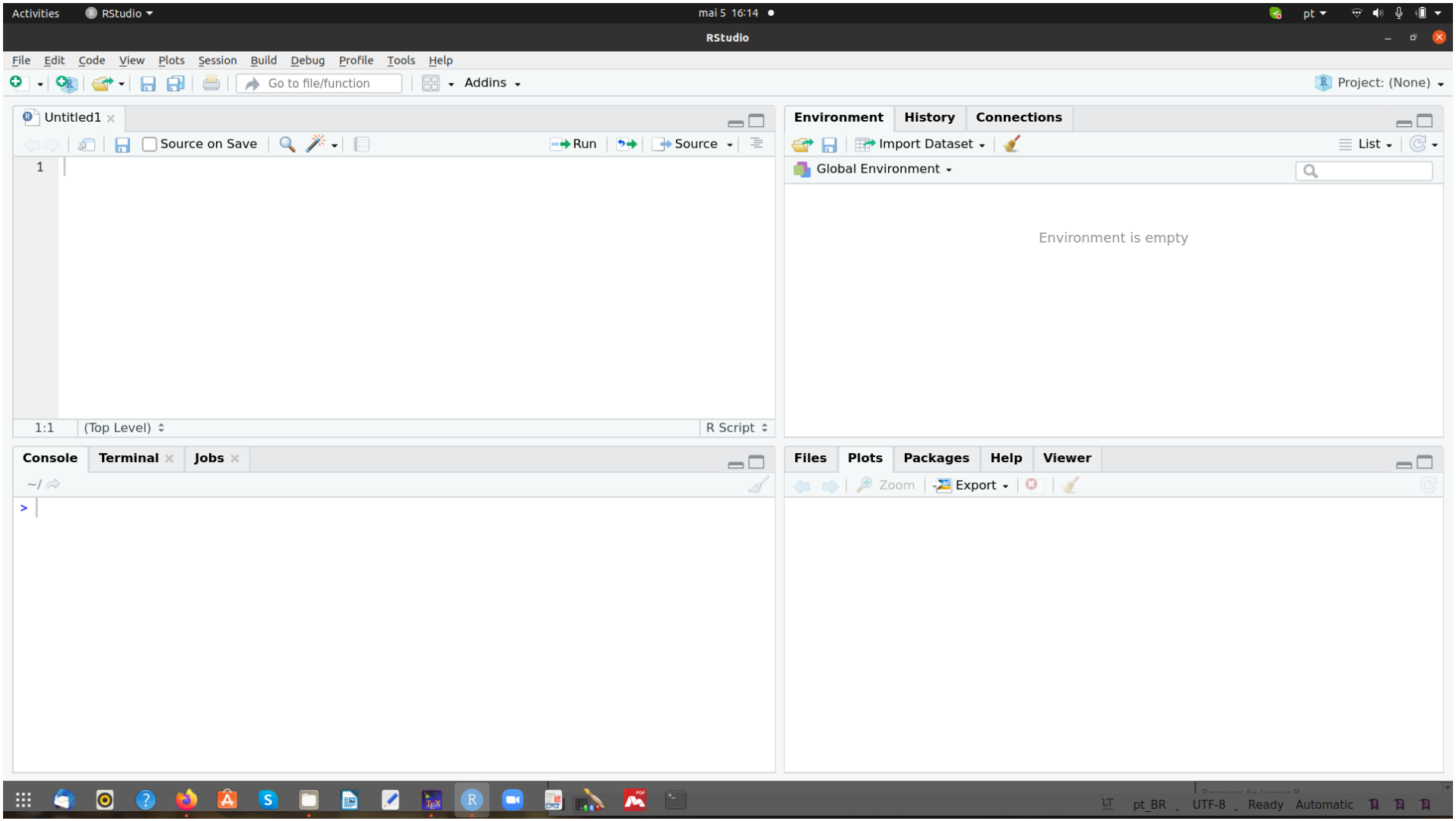
- **Data Visualization - A Practical Introduction**
  Textbook on data visualization, using ggplot2.
  ▸ Link
- **ggplot2 - Elegant Graphics for Data Analysis**
  In depth discussion of R-package for data vizualization.
  ▸ Link
- **An Economist's Guide to Visualizing Data**
  Guidelines for good visualizations (not R-specific).
  ▸ Link
- **A Layered Grammar of Graphics**
  The theory behind ggplot2.
  ▸ Link
- **QuantEcon: Open source code for economic modeling**
  ▸ Link

Installing R

- Link

- Link

RStudio


Activities
RStudio
mai 5 16:14
pt
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Console
Terminal
Jobs
~/
>
Files
Plots
Packages
Help
Viewer
Zoom
Export
pt_BR
UTF-8
Ready
Automatic

R

Files

- **.R** *scripts*
- **.RData** *workspace*
- **.Rhistory** *R history*
- **.Rproj** *R projects*

Basics

- Workspace and directory:

```
getwd() # Returns current working directory
# Sets working directory
setwd("/home/mrm/Documents/Teaching/metodoscomputacionais")
dir() # Lists the content of the working directory
ls() # workspace objetcs
rm(list=ls())  #Clears memory
graphics.off() #Clears graphs
save.image("my_workspace.RData") # Saves the workspace
load("my_workspace.RData") # Loads the workspace
rm(file1,file2) #Cleanup
q() # Closes session and save workspace image
```

We can define a working directory. Note for Windows users : R uses slash ("/") in the directory instead of backslash ("\").

Basics

- Packages:

```
install.packages("pachagename") # Installs a pachage
library(pachagename) # Loads pachages
require(pachagename) # Loads pachages
```

- Especial symbols:

```
NA # Missing values
NaN # Not a Number
TRUE # Logic value true (T)
FALSE # Logic value false (F)
NULL # Null value
Inf: Infinity
-Inf: Minus Infinity.
```

```
0/0
```

```
## [1] NaN
```

```
1/0
```

```
## [1] Inf
```

Basics

- **Caution**: integer versus modulo division.

```
5 %/% 3
# 5 divided by 3 without decimal positions --> 1
5 %% 3
# rest of division of 5 by 3 --> 2
```

- **Caution**: decimal notation with . and not ,

```
1,2
--> Error: unexpected ',' in "1,"
1.2
# correct notation
```

A sample session in R

- Please type the commands on the following slides in your RStudio terminal.
- This session is based on

  ▸ Link
- R can be used as a simple calculator and we can perform any simple computation.

```
# Sample Session
# This is a comment
2 # print a number
2+3 # perform a simple calculation
log(2) # natural log
```

A sample session in R

- R can be used as a simple calculator and we can perform any simple computation.

```
# Sample Session
# This is a comment
2 # print a number
```

```
## [1] 2
```

```
2+3 # perform a simple calculation
```

```
## [1] 5
```

```
log(2) # natural log
```

```
## [1] 0.6931472
```

Numeric and string objects.

```
x <- 2 # store an object
x # print this object
```

```
## [1] 2
```

```
(x <- 3) # store and print an object
x <- "Hello" # store a string object
x
```

```
## [1] 2
```

```
x <- "Hello" # store a string object
x
```

```
## [1] "Hello"
```

Vectors

```
#store a vector
Height <- c(168, 177, 177, 177, 178, 172, 165, 171, 178, 170)
Height[2] # Print the second component
```

```
## [1] 177
```

```
# Print the second, the 3rd, the 4th and 5th component
Height[2:5]
```

```
## [1] 177 177 177 178
```

```
(obs <- 1:10) # Define a vector as a sequence (1 to 10)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

Vectors

```
Weight <- c(88, 72, 85, 52, 71, 69, 61, 61, 51, 75)
# Performs a simple calculation using vectors
BMI <- Weight/((Height/100)^2)
BMI
```

```
## [1] 31.17914 22.98190 27.13141 16.59804 22.40879 23.32342
22.40588
## [8] 20.86112 16.09645 25.95156
```

Vectors

- We can also describe the vector with **length()**, **mean()** and **var()**.

```
length(Height)
```

```
## [1] 10
```

```
mean(Height) # Compute the sample mean
```

```
## [1] 173.3
```

```
var(Height)
```

```
## [1] 22.23333
```

Matrices

```
M <- cbind(obs,Height,Weight,BMI) # Create a matrix
typeof(M) # Give the type of the matrix
```

```
## [1] "double"
```

```
class(M) # Give the class of an object
```

```
## [1] "matrix"
```

```
is.matrix(M) # Check if M is a matrix
```

```
## [1] TRUE
```

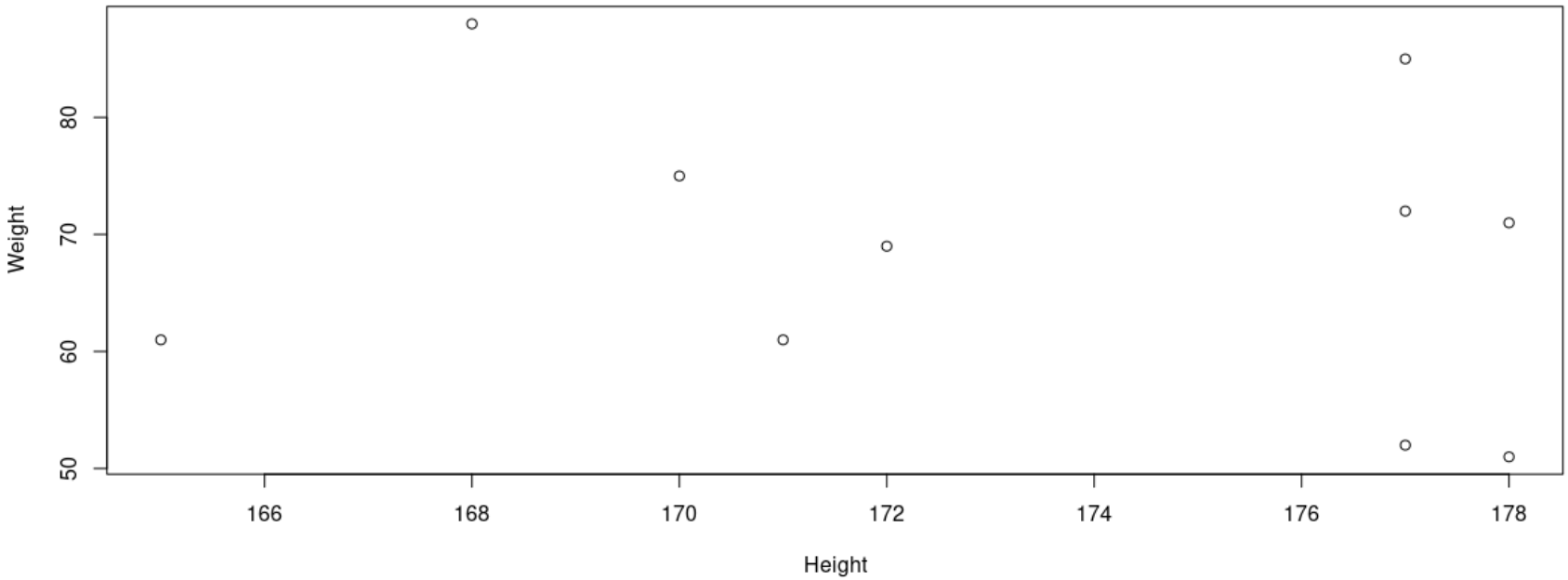
```
dim(M) # Dimensions of a matrix
```

```
## [1] 10 4
```

Simple plotting

- For "quick and dirty" plots, use **plot**.
- For more advanced and attractive data visualizations, use **ggplot**.

```
plot(Height,Weight,ylab="Weight",xlab="Height")
```



Dataframes (tibbles)

- **tibbles** are modernized versions of **dataframes**.
- Technically: Lists of vectors (with names).
- Can have different datatypes in different vectors.

```
library(tibble) # Load the tidyverse tibble package
mydat <- as_tibble(M) # Creates a tibble
names(mydat) # Give the names of each variable
```

```
## [1] "obs"    "Height" "Weight" "BMI"
```

```
summary(mydat) # Descriptive Statistics
```

```
##       obs             Height          Weight           BMI
## Min.   : 1.00   Min.   :165.0   Min.   :51.00   Min.   :16.10
## 1st Qu.: 3.25   1st Qu.:170.2   1st Qu.:61.00   1st Qu.:21.25
## Median : 5.50   Median :174.5   Median :70.00   Median :22.70
## Mean   : 5.50   Mean   :173.3   Mean   :68.50   Mean   :22.89
## 3rd Qu.: 7.75   3rd Qu.:177.0   3rd Qu.:74.25   3rd Qu.:25.29
## Max.   :10.00   Max.   :178.0   Max.   :88.00   Max.   :31.18
```

Reading and writing data

- There are many routines for reading and writing files.
- Tidyverse versions are in the readr package.

```
library(readr) #load the tidyverse readr package
write_csv(mydat, "my_data.csv")
mydat2=read_csv("my_data.csv")
```

```
## Parsed with column specification:
## cols(
## obs <- col_double(),
## Height <- col_double(),
## Weight <- col_double(),
## BMI <- col_double()
## )
```

Reading and writing data

```
mydat2
```

```
# A tibble: 10 x 4
obs Height Weight   BMI
<dbl>  <dbl>  <dbl> <dbl>
1      1    168      88  31.2
2      2    177      72  23.0
3      3    177      85  27.1
4      4    177      52  16.6
5      5    178      71  22.4
6      6    172      69  23.3
7      7    165      61  22.4
8      8    171      61  20.9
9      9    178      51  16.1
10     10    170      75  26.0
```

Important mathematical functions

```
exp(1)
exp(log(5))
sin(pi/2)
cos(pi/2)
max(4,2,5,1)
min(4,2,5,1)
sum(4,2,5,1)
prod(4,2,5,1)
sqrt(16)
factorial(4) # "4 factorial", 4!
choose(5,2) # "5 choose 2"
```

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

Important mathematical functions

Function	Meaning
log(x)	log to base e of x
exp(x)	antilog of x (e^x)
log(x,n)	log to base n of x
log10(x)	log to base 10 of x
sqrt(x)	square root of x
factorial(x)	$x!$
choose(n,x)	binomial coefficients $n!/(x!\ (n-x)!)$
gamma(x)	$\Gamma(x)$, for real x $(x-1)!$, for integer x
lgamma(x)	natural log of $\Gamma(x)$
floor(x)	greatest integer $< x$
ceiling(x)	smallest integer $> x$
trunc(x)	closest integer to x between x and 0 trunc(1.5) $= 1$, trunc(-1.5) $= -1$ trunc is like floor for positive values and like ceiling for negative values
round(x, digits=0)	round the value of x to an integer
signif(x, digits=6)	give x to 6 digits in scientific notation
runif(n)	generates n random numbers between 0 and 1 from a uniform distribution
cos(x)	cosine of x in radians
sin(x)	sine of x in radians
tan(x)	tangent of x in radians
acos(x), asin(x), atan(x)	inverse trigonometric transformations of real or complex numbers
acosh(x), asinh(x), atanh(x)	inverse hyperbolic trigonometric transformations of real or complex numbers
abs(x)	the absolute value of x, ignoring the minus sign if there is one

Write your own functions - Next week

Defining functions

```
example_function = function(a, b=2) {
r=a/b
return(r)
}
example_function(3)
```

```
## [1] 1.5
```

```
example_function(3,4)
```

```
## [1] 0.75
```

```
example_function(b=4,a=3)
```

```
## [1] 0.75
```

Data visualisation

Prerequisite

- Load the tidyverse by running:

```
install.packages("tidyverse")
library(tidyverse)
```

```
#>  Attaching packages                 tidyverse 1.3.0
#>  ggplot2 3.3.2      purrr   0.3.4
#>  tibble  3.0.3      dplyr   1.0.2
#>  tidyr   1.1.2      stringr 1.4.0
#>  readr   1.4.0      forcats 0.5.0
#>    Conflicts                    tidyverse_conflicts()
#>  dplyr::filter() masks stats::filter()
#>  dplyr::lag()    masks stats::lag()
```

Data visualisation

Do cars with big engines use more fuel than cars with small engines?

US Environmental Protection Agency data: ggplot2::mpg

mrm

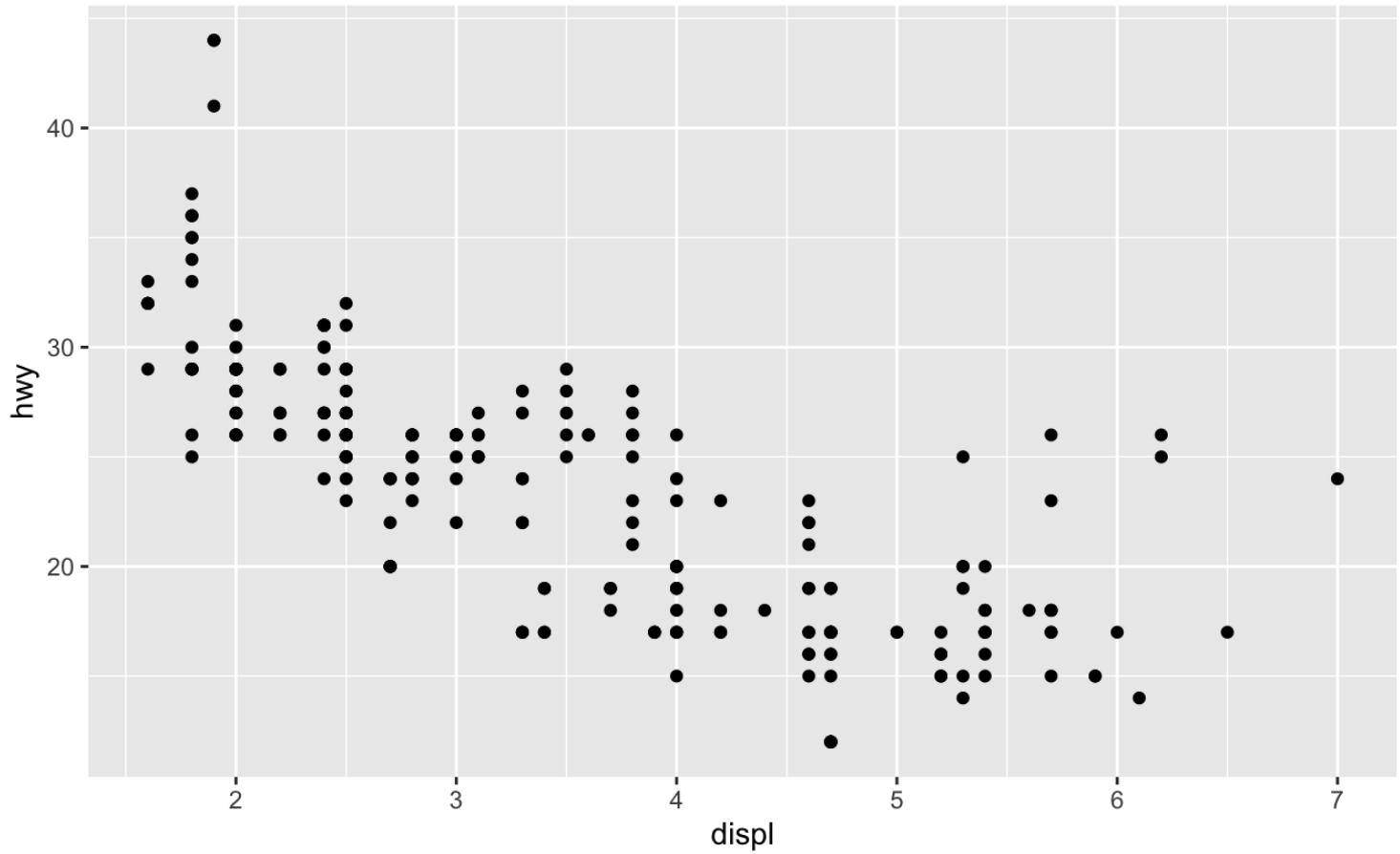
```
#> # A tibble: 234 x 11
#>   manufacturer model displ  year   cyl trans      drv      cty   hwy fl
#>   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr>
#> 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p
#> 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p
#> 3 audi         a4      2    2008     4 manual(m6) f        20    31 p
#> 4 audi         a4      2    2008     4 auto(av)   f        21    30 p
#> 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p
#> 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p
#> # ... with 228 more rows
```

1) displ, a car's engine size, in litres. 2) hwy, a car's fuel efficiency on the highway, in miles per gallon (mpg).

Data visualisation

Creating a ggplot

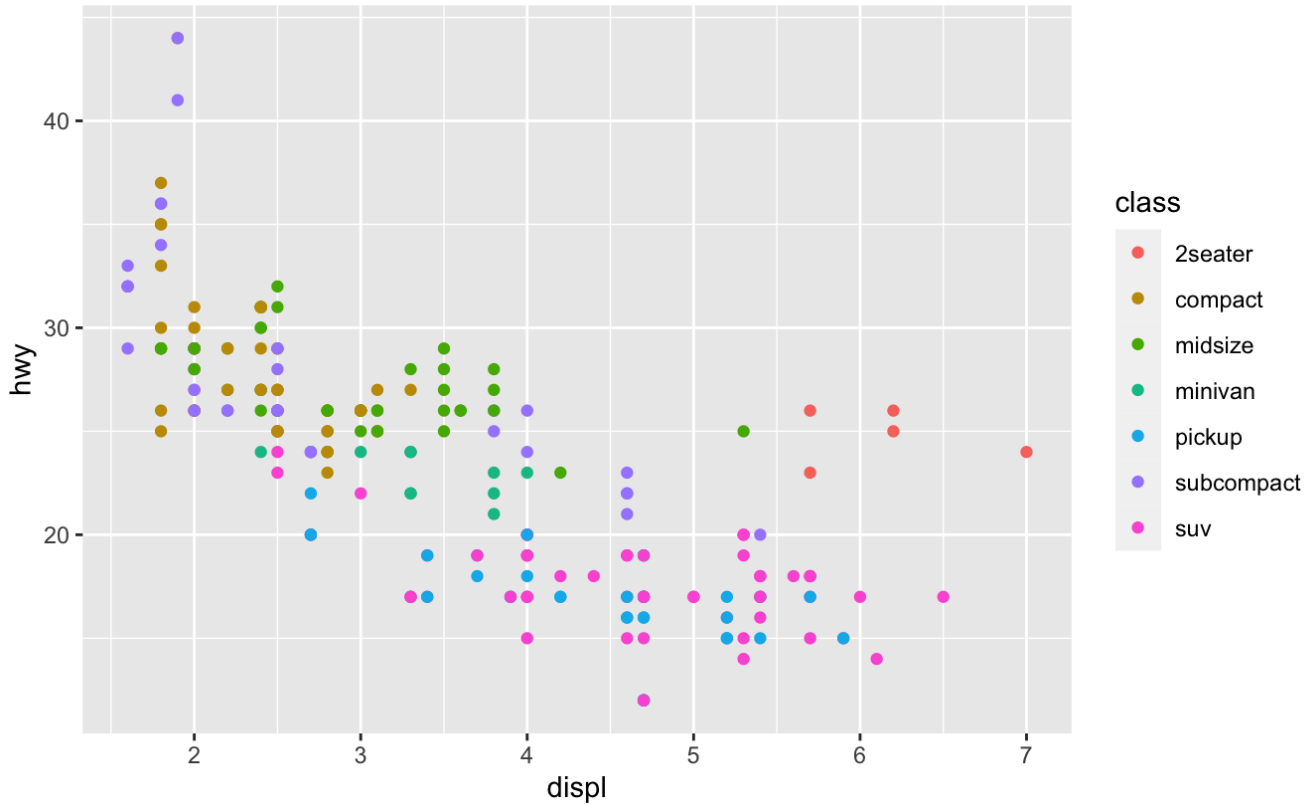
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy))
```



Data visualisation

Aesthetic mappings

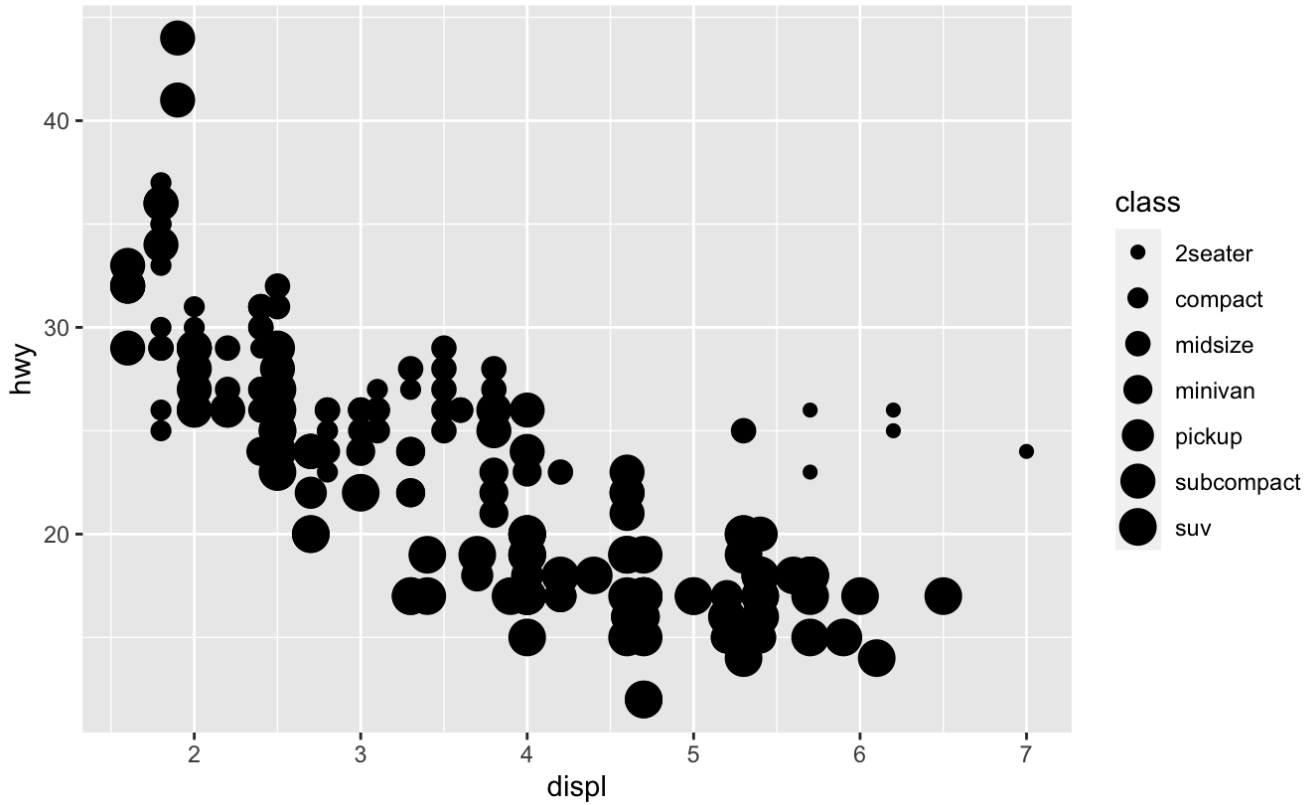
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy,
color = class))
```



Data visualisation

Size aesthetic

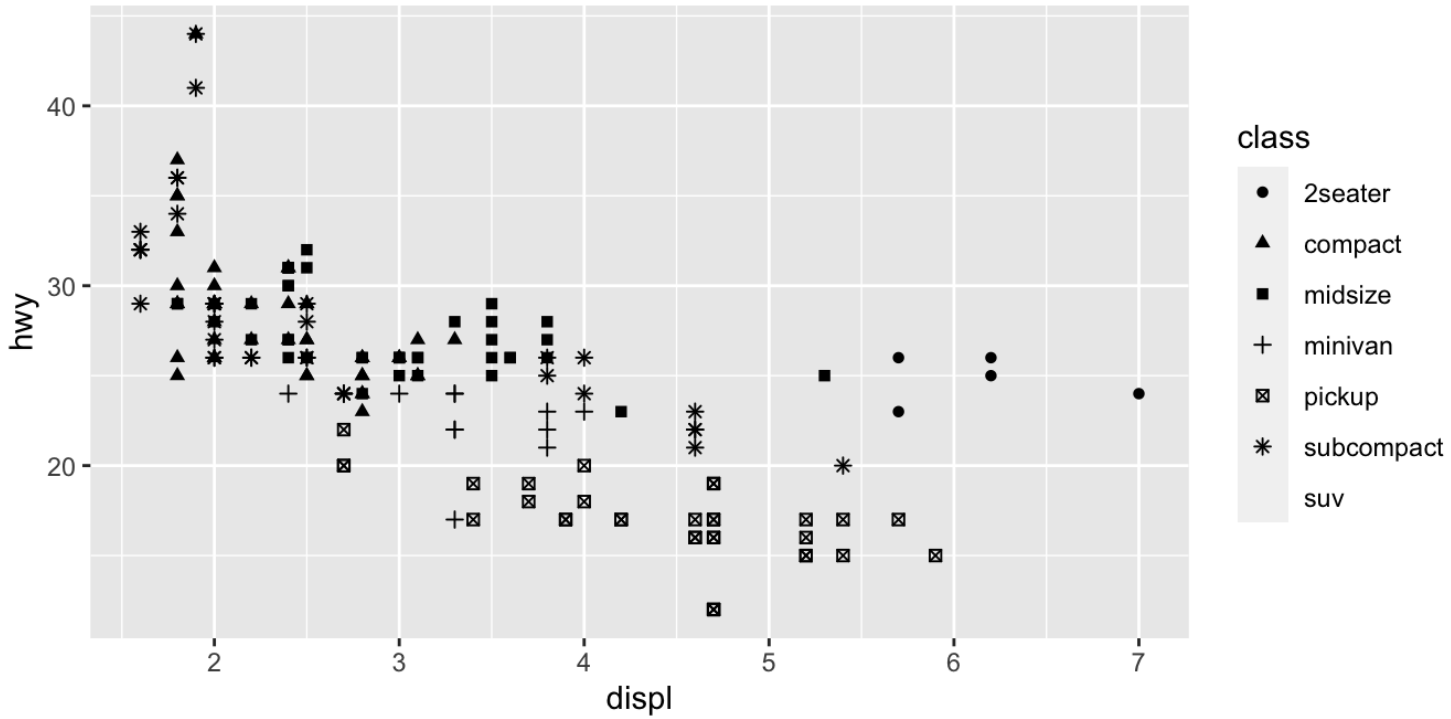
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy,
size = class))
#> Warning: Using size for a discrete variable is not
advised.
```



Data visualisation

Alpha aesthetic

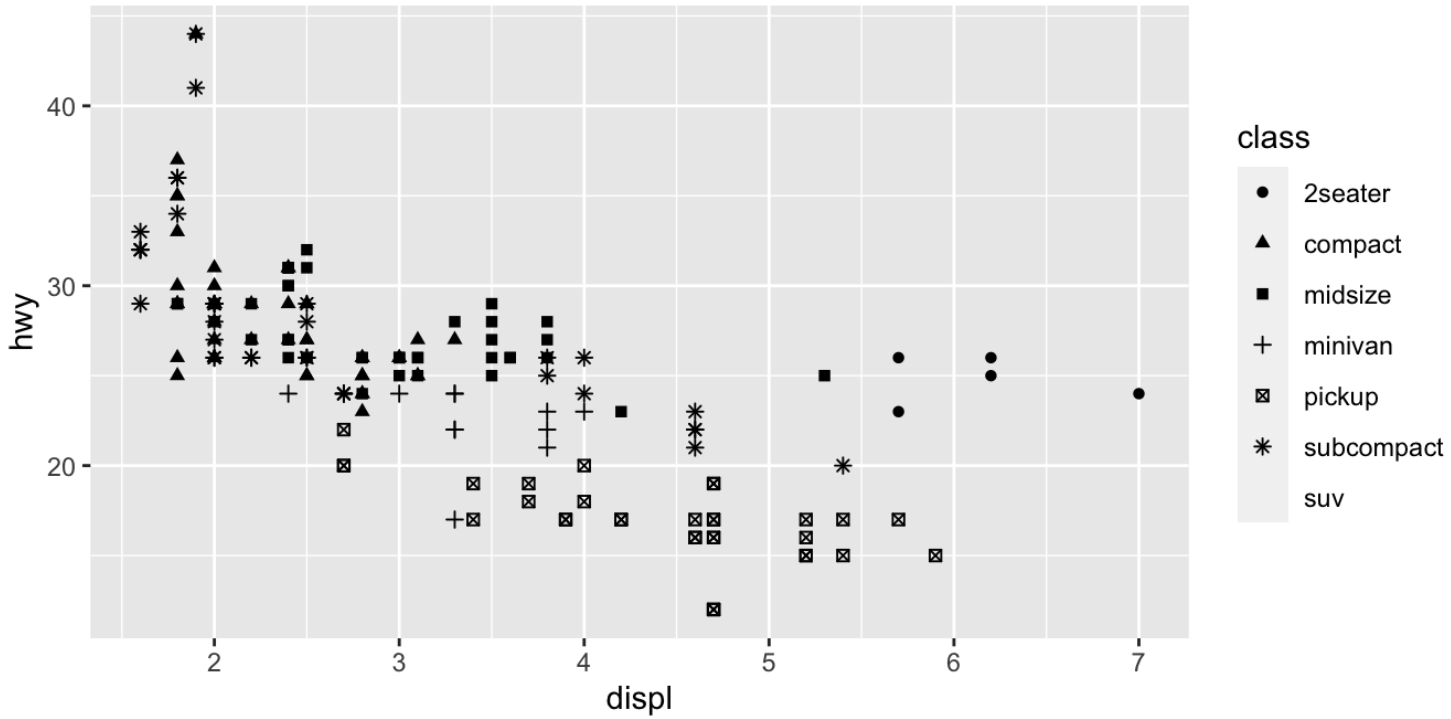
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy,
alpha = class))
```



Data visualisation

Shape aesthetic

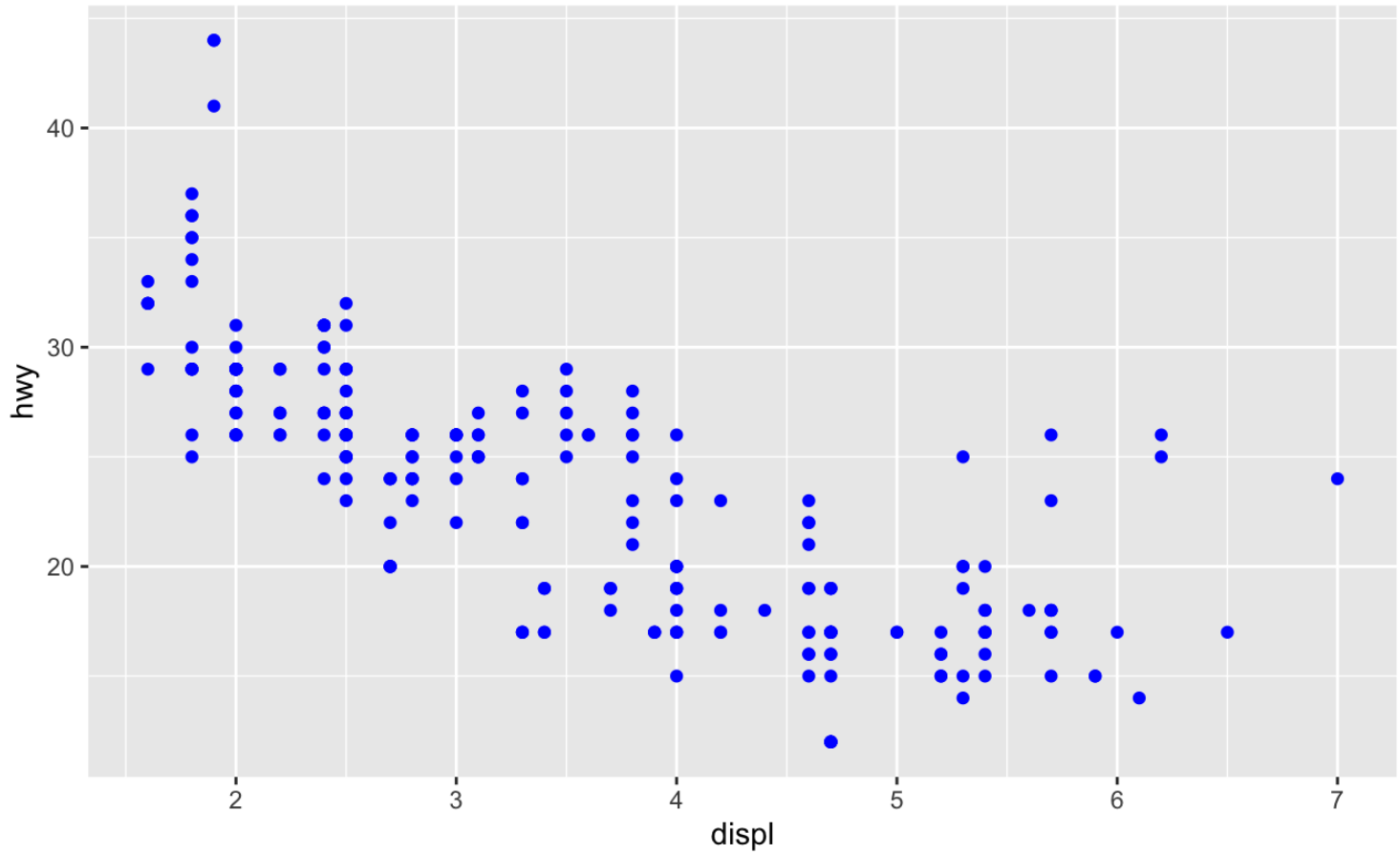
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy,
shape = class))
```



Data visualisation

Set the aesthetic properties manually

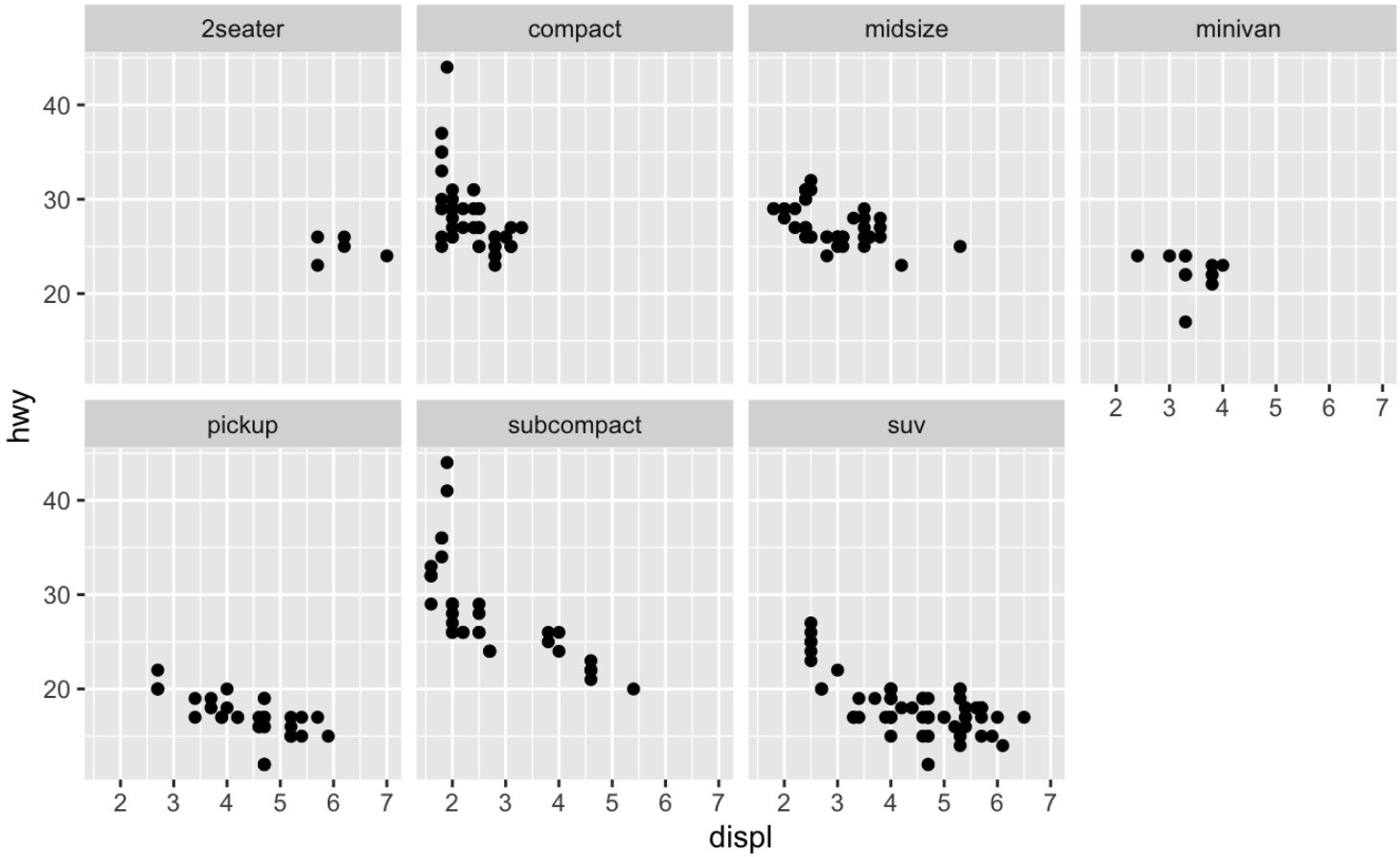
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy), color =
```



Data visualisation

Facets

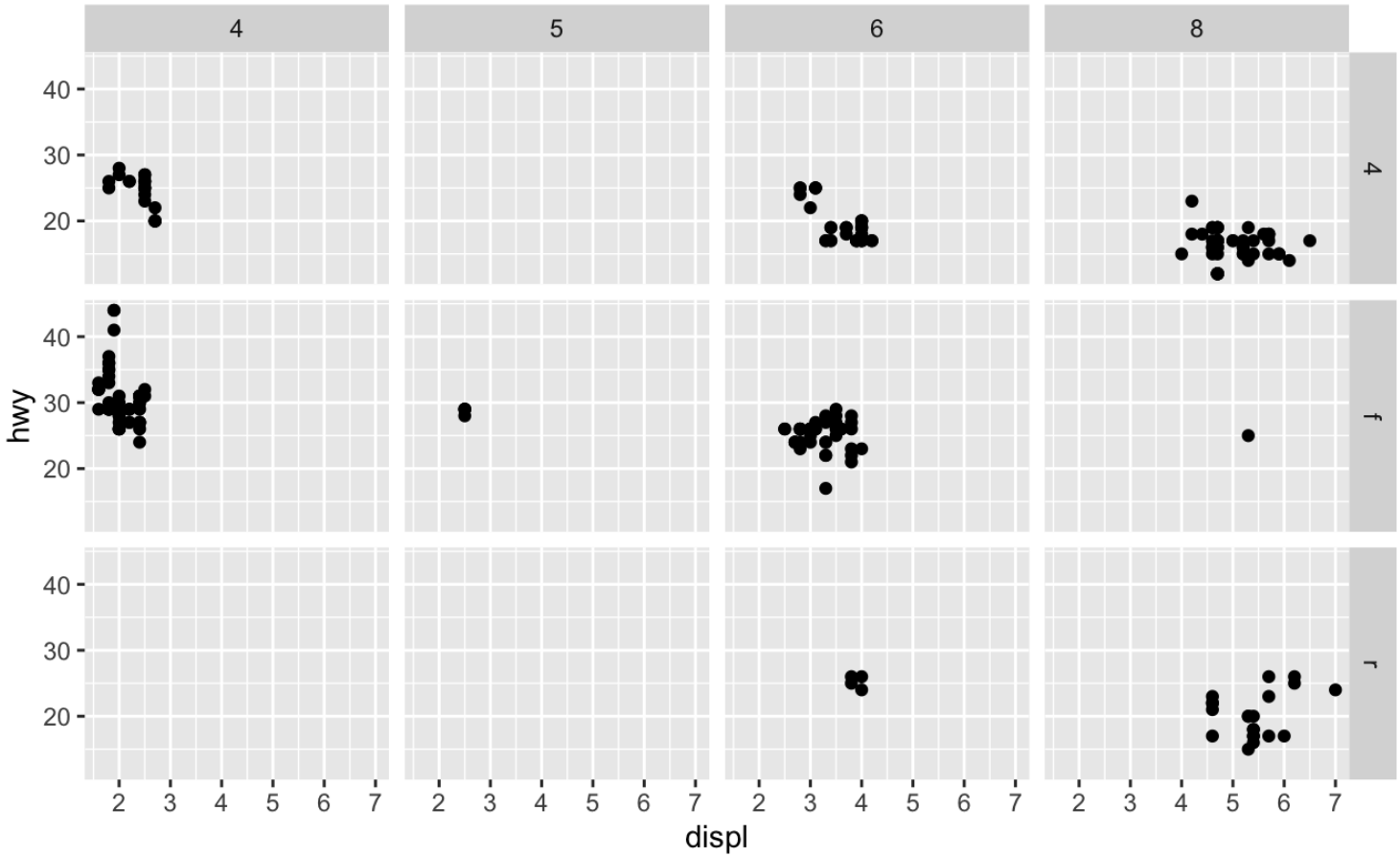
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) +
facet_wrap(~ class, nrow = 2)
```



Data visualisation

Combination of two variables

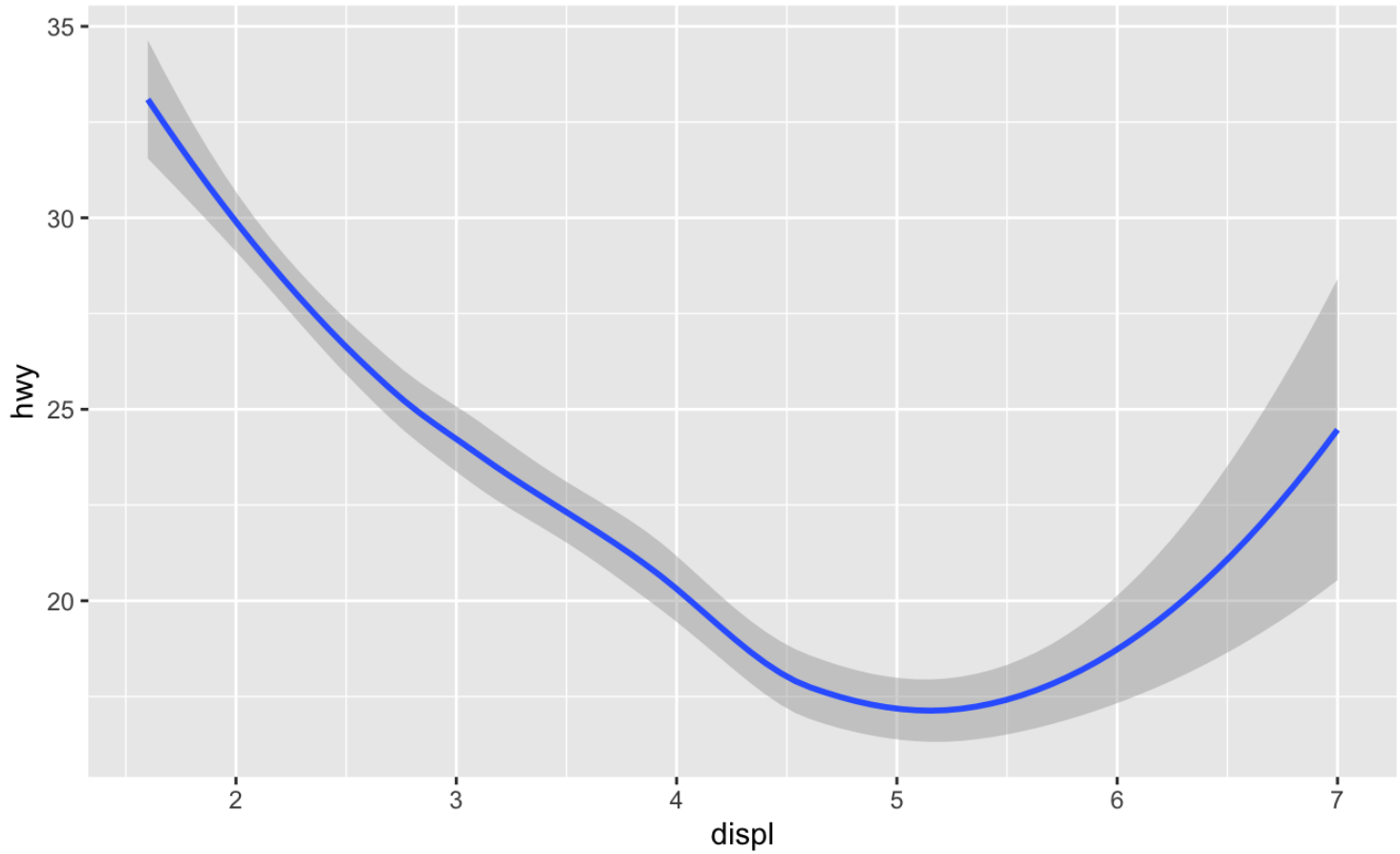
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy)) +
facet_grid(drv ~ cyl)
```



Data visualisation

Geometric objects

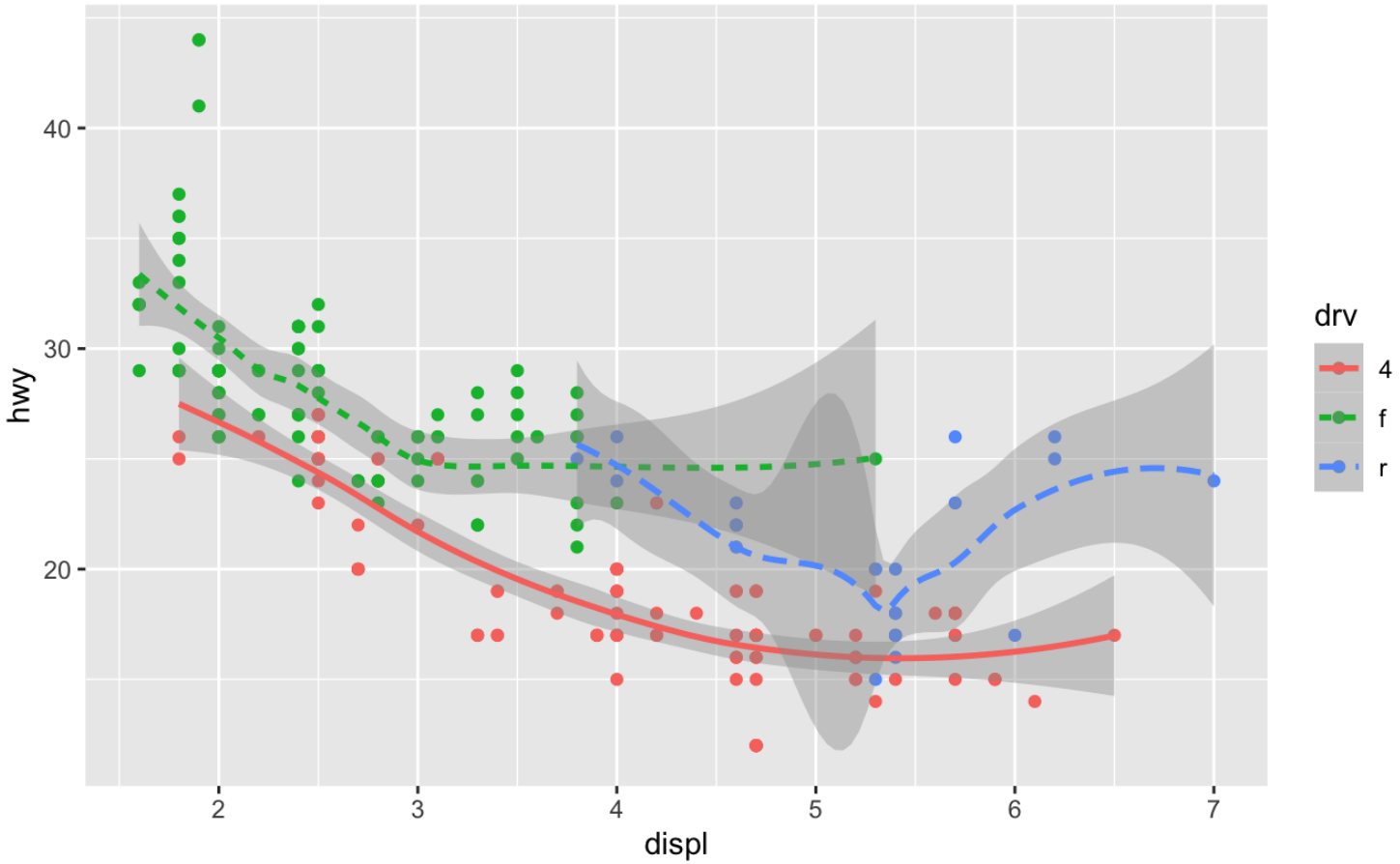
```
ggplot(data = mpg) +
geom_smooth(mapping = aes(x = displ, y = hwy))
```



Data visualisation

Linetype of a line and

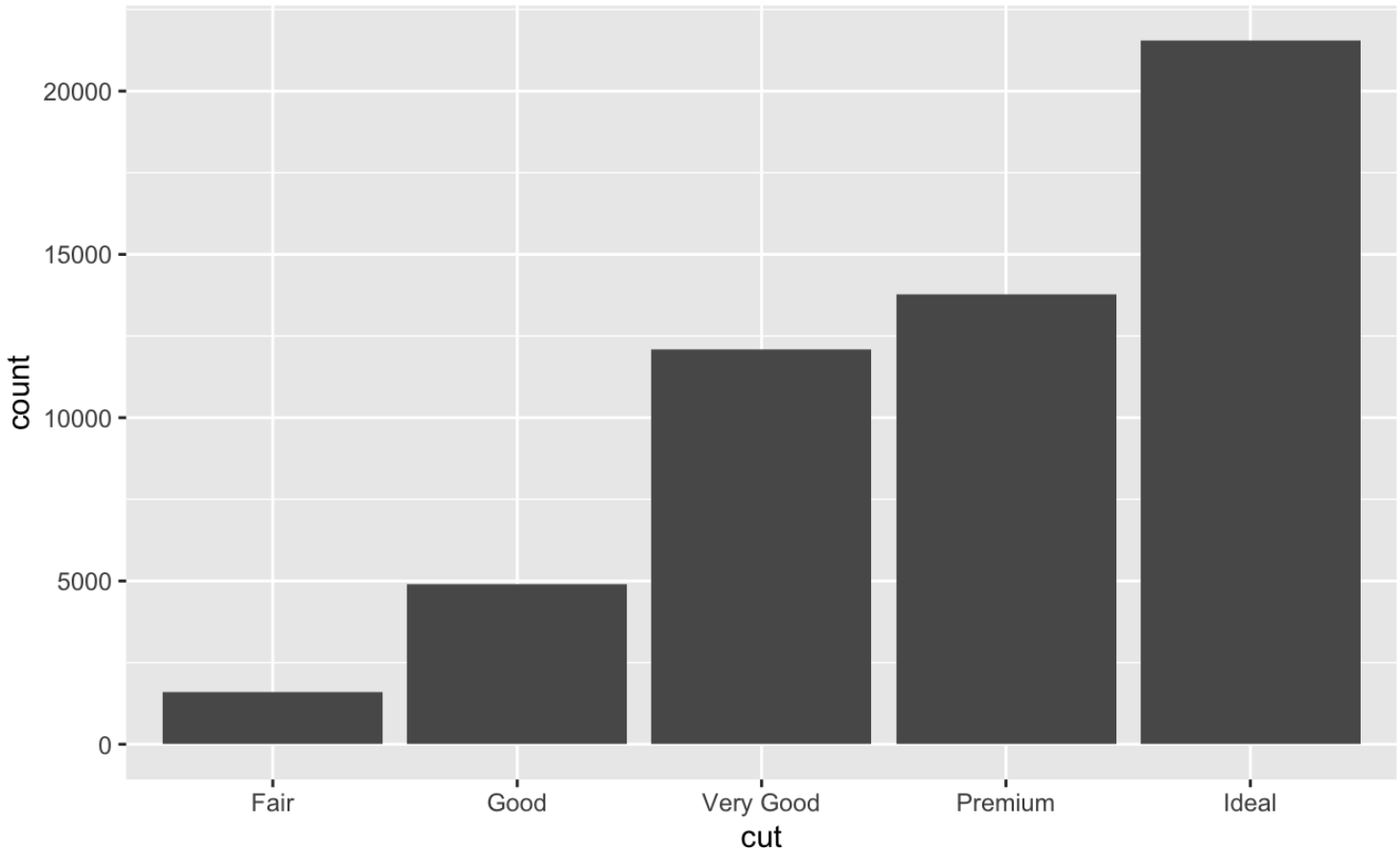
```
ggplot(data = mpg) +
geom_point(mapping = aes(x = displ, y = hwy, color = d
geom_smooth(mapping = aes(x = displ, y = hwy, linetype
```



Data visualisation

Bar chart

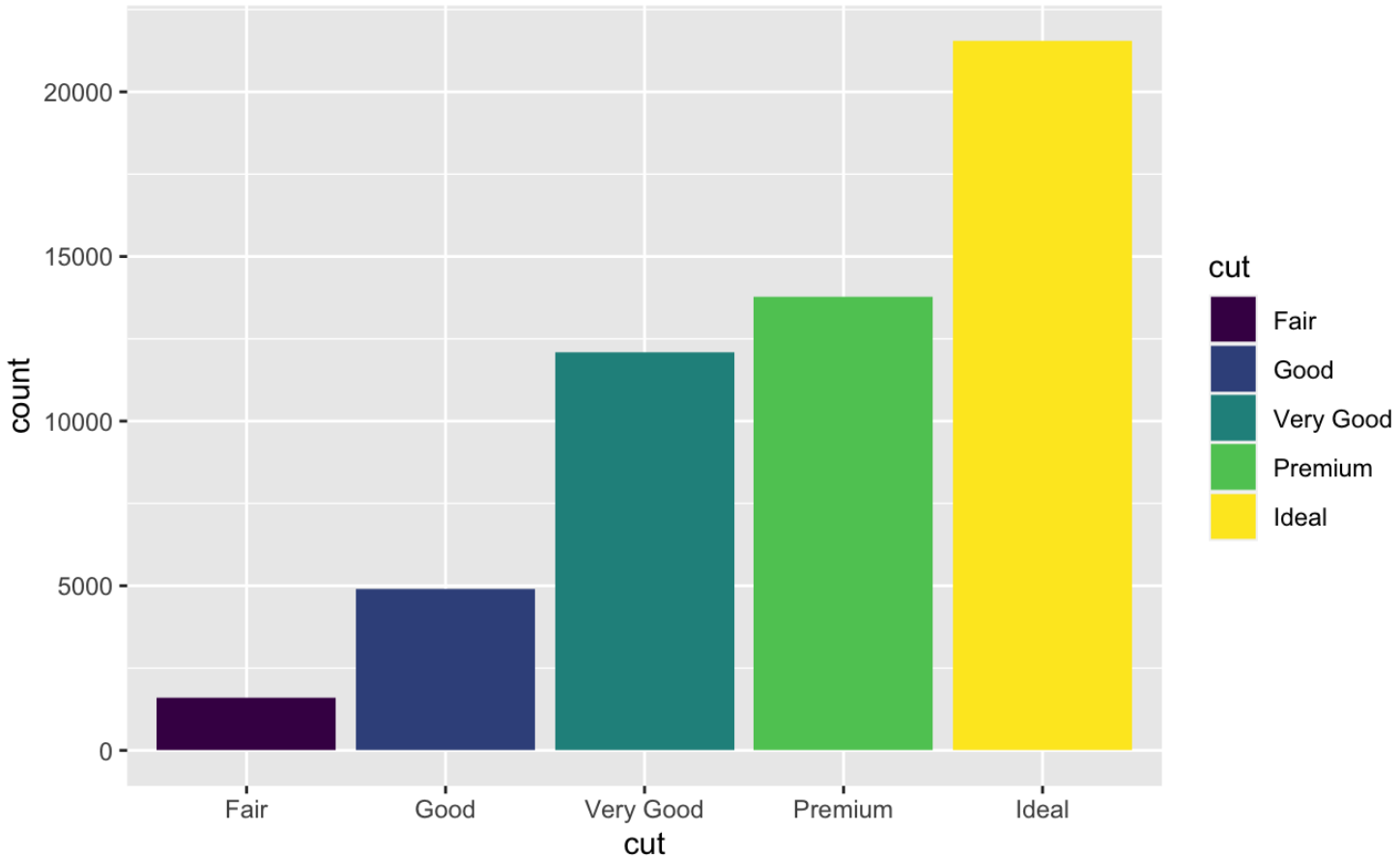
```
ggplot(data = diamonds) +
geom_bar(mapping = aes(x = cut))
```



Data visualisation

Bar chart

```
ggplot(data = diamonds) +
geom_bar(mapping = aes(x = cut, fill = cut))
```



Help!

```
help(solve) #help page for command solve
?solve #same as help(solve)
help("exp")
help.start()
help.search("solve") #list of commands which could be related
to string solve

??solve # same as help.search(solve)
example(exp) #examples for the usage of 'exp'
example("*") #special characters in quotation marks
```

Book: [Crawley, 2012].

References

Crawley, M. J. (2012).
The R book.
John Wiley & Sons.