# Graphics in R

A. Ardalan

17/10/2022

# Types of more common use Statistical Graphs

`Types of Data (Variables)`

## Numeical Variables

- Estimation and shape of Distribution
  - Histogram and Density plots
  - Box-plot
- Comparisons
  - Box-plots & Violin plots
- Associations and finding the structure between variables
  - Scatter plots
  - Time series plots
- Other Advanced Graphic Tools

## Categorical Variables

- Count and Percentages
  - Bar charts and Pie charts

## Explore Graphics

# Overview of R graphics

1. Standard `graphics` in R

- The `graphics` package, as the base graphics system, provides a complete set of functions for creating a wide variety of plots.

Grid system:

2. Trellis & `lattice` graphics

- Trellis Graphics is a family of techniques for viewing complex, multi-variable data sets
- The techniques were given the name Trellis because: usually results are in a rectangular array of plots.

3. Grammar of Graphics `ggplot2`

- A powerful approach, based on the "Grammar of Graphics"
- A graphics language, composed of layers, "geoms" (points, lines, regions), each with graphical "aesthetics" (color, size, shape).

# Data Sets Which I am using here

## Titanic Data set

RMS Titanic was a British passenger liner which sank in the North Atlantic Ocean on 15 April 1912 after striking an iceberg. Voyage: from UK to US.

| head | pclass | survived | Residence | name | age | sibsp | parch | ticket | fare | cabin | embarked | boat | body | home.dest | Gender |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3rd | Died | American | Abbing, Mr. Anthony | 42 | 0 | 0 | C.A. 5547 | 7.55 | | S | | | | Male |
| 2 | 3rd | Died | American | Abbott, Master. Eugene Joseph | 13 | 0 | 2 | C.A. 2673 | 20.25 | | S | | | East Providence, RI | Male |
| 3 | 3rd | Died | American | Abbott, Mr. Rossmore Edward | 16 | 1 | 1 | C.A. 2673 | 20.25 | | S | | 190 | East Providence, RI | Male |
| 4 | 3rd | Survived | American | Abbott, Mrs. Stanton (Rosa Hunt) | 35 | 1 | 1 | C.A. 2673 | 20.25 | | S | A | | East Providence, RI | Female |
| 5 | 3rd | Survived | Other | Abelseth, Miss. Karen Marie | 16 | 0 | 0 | 348125 | 7.65 | | S | 16 | | Norway Los Angeles, CA | Female |

## `USArrests` Data set

Violent Crime Rates by US State

| head | Murder | Assault | UrbanPop | Rape |
|---|---|---|---|---|
| Alabama | 13.2 | 236 | 58 | 21.2 |
| Alaska | 10.0 | 263 | 48 | 44.5 |
| Arizona | 8.1 | 294 | 80 | 31.0 |
| Arkansas | 8.8 | 190 | 50 | 19.5 |
| California | 9.0 | 276 | 91 | 40.6 |

## 'mtcars' Data set

Motor Trend Car Road Tests

| head | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |

# Types of Graphs in R

- High Level
  Some examples of high level graphics functions are:
  - Pie Charts, Bar Charts and Histograms
  - Box-and-Whisker Plots
  - Scatter plots
  - Time Series Plots
  - Surface Plots
  - Other Advanced Plots
- Low Level
  The high-level graphics facilities in R are built on a set of flexible low-level ones. The low level facilities include:
  - Page Layout
  - Setup of Plotting Coordinates
  - Drawing Points and Lines
  - Drawing Polygons and Rectangles
  - Color Management

# graphics package: base graphics system

1. **plot**
   The **'plot'** function produces an entire graph with a single function call.
   It can be useful to access graphics functionality at a lower level so that it is possible to create graphs in a much more flexible way.
2. **plot.new**
3. **plot.window**
   - The functions **plot.new** and **plot.window** are the functions which make it possible to work in this low-level way.
   - **plot.new** is used to begin a new plot,
   - **plot.window** is used to set up coordinate systems.
   - Neither function does any actual drawing.

# Graphical commands

High level graphical commands create the plot

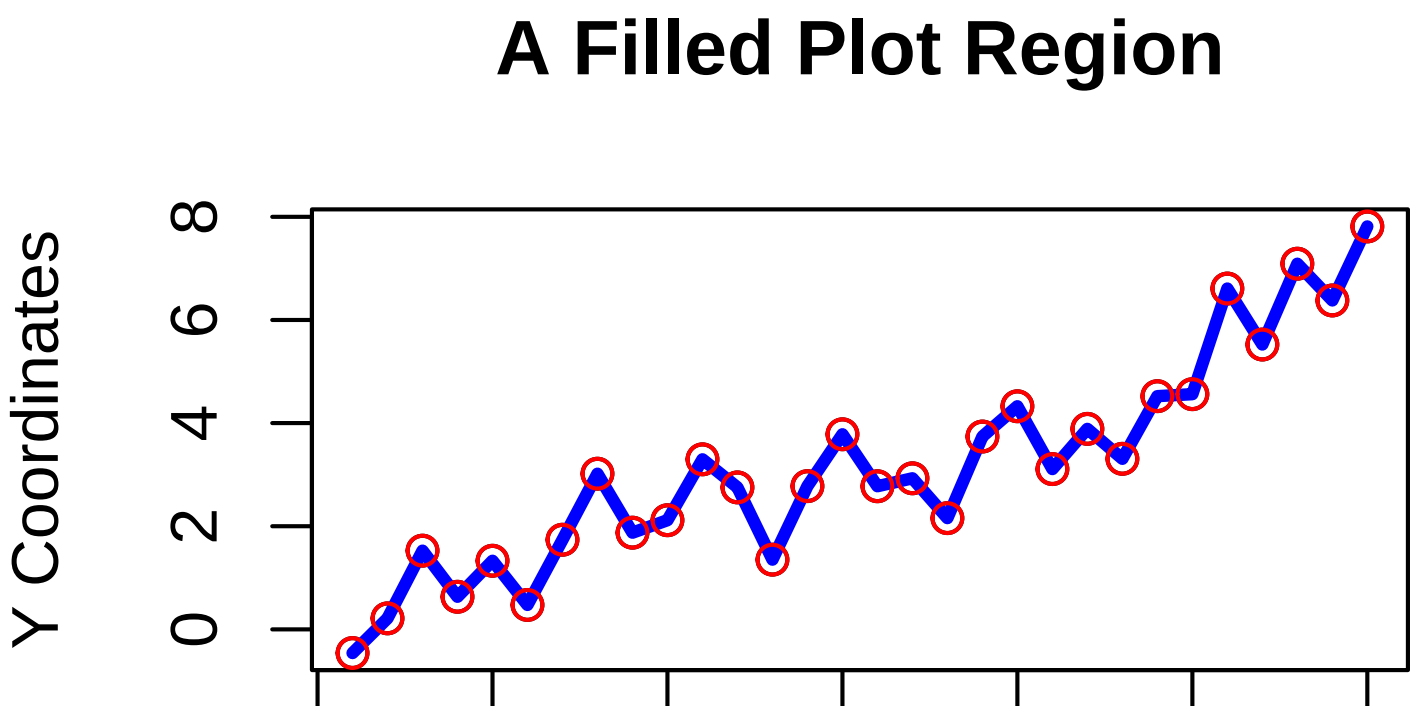
1. plot() # Scatter plot, and general plotting
2. hist() # Histogram
3. barplot() # Barplot
4. boxplot() # Boxplot
5. pairs() # Plots for multivariate data (Matrix scatter plots)
6. qqnorm() # Normal probability plot

Low level graphical commands add to the plot

1. points() # Add points
2. lines() # Add lines
3. text() # Add text
4. abline() # Add lines
5. arrows() # Add arrows
6. segments() # add line segment
7. legend() # Add legend

# Plot

```
x = 1:30 ; y = rnorm(30) + x/5
plot(x, y, xlim = range(x), ylim = range(y),
     xlab = "X Coordinates", ylab = "Y Coordinates",
     main = "A Filled Plot Region")
lines(x, y, lwd = 3, col = "blue")
points(x, y, pch = 21, col = "red")
```



# Plot 2

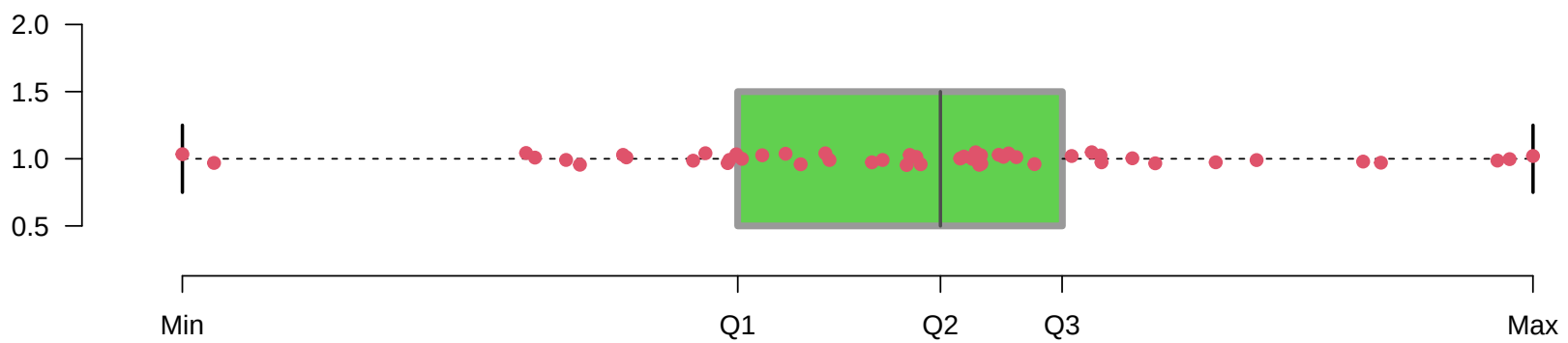
1. Type of the plot
   - plot() # point
   - plot(..., type = "l") # lines
   - plot(..., type = "b") # points connected by lines
   - plot(..., type = "o") # lines are over the points
   - plot(..., type = "h") # vertical lines
   - plot(..., type = "s") # steps
2. Type of the line
   - plot(..., type = "l", lty=1) # solid
   - plot(..., type = "l", lty=2) # dashed
   - plot(..., type = "l", lty=3) # dotted
   - plot(..., type = "l", lty=4) # dotdash
   - plot(..., type = "l", lty=5) # longdash
   - plot(..., type = "l", lty=6) # twodash

# Drawing a Boxplot : Step by Step, using Low-level graphics

```
par(mar = c(2, 2.5, 0, 0)) # Setting the Margins
mydata = rnorm(n = 50, mean = 1, sd = 4); x = mydata; n = length(mydata) # Generating 50 data from Normal

Minx = min(x); Maxx = max(x); Quarts = quantile(x, c(0.25, 0.5, 0.75)) # Finding the min, max and Quartile
SM = c(Minx, Quarts, Maxx); names(SM) = c('Min', paste('Q', 1:3, sep = ''), 'Max') # Summary of data in 5
SM
```

```
##         Min          Q1          Q2          Q3         Max
## -12.1489628  -2.5618246   0.9389338   3.0448150  11.1709671
```

```
plot.new()
plot.window(xlim = c(Minx - 0.15 * sd(x), Maxx + 0.15 * sd(x)) , ylim = c(0.2, 2) )

rect(xleft = SM[2], ybottom = 0.5, xright = SM[4], ytop = 1.5, lwd = 4 , col = 3, border = "gray60")

segments(x0 = SM[1], y0 = 1, x1 = SM[2], y1 = 1, lty = 'dashed')
segments(x0 = SM[4], y0 = 1, x1 = SM[5], y1 = 1, lty = 'dashed')

segments(SM[1], 0.75, SM[1], 1.25, lwd = 2)
segments(SM[5], 0.75, SM[5], 1.25, lwd = 2)
segments(SM[3], 0.5, SM[3],1.5, lwd = 2, col = "gray30" )
axis(1, at = round(SM,2), labels = names(SM)); axis(2, las = 2)

points(x = mydata, y = runif(n, 0.95, 1.05), col = 2, pch = 19)
```

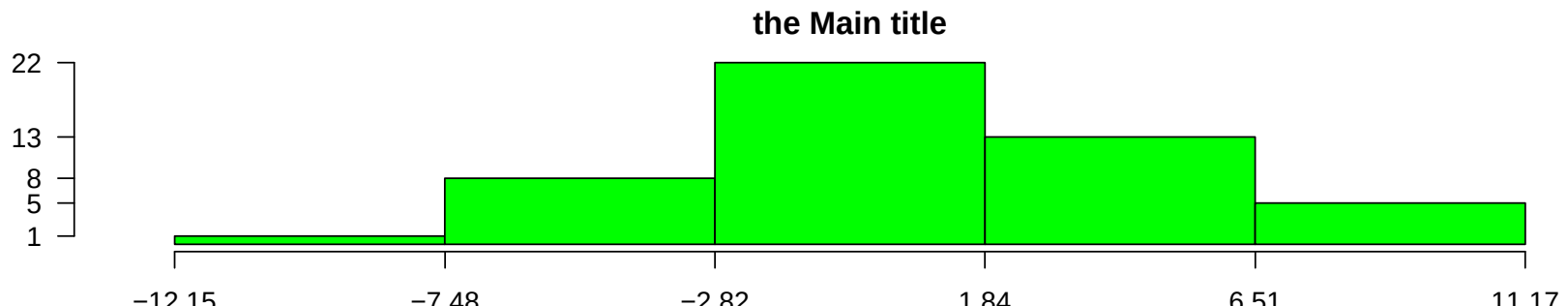


# Drawing a Histogram : Step by Step, using Low-level graphics

```
par(mar = c(2, 2.5, 2, 0))

k = 5                        # number of classes or bars
widthx = (Maxx - Minx)/k     # Setting the bin

cut_ps = seq(Minx, Maxx, by = widthx) # Setting the cut-points
grx    = cut(x, cut_ps)               # classified the data in groups
(TF    = table(grx))                   # Tabulate the grouped
```

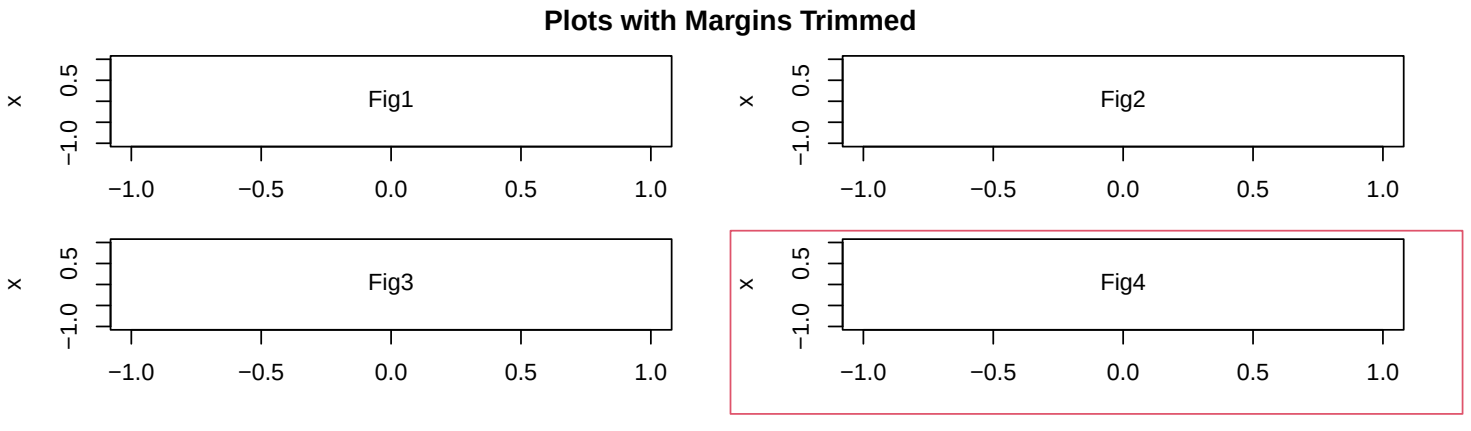
```
## grx
## (-12.1,-7.48] (-7.48,-2.82]  (-2.82,1.84]   (1.84,6.51]   (6.51,11.2]
##             1             8            22            13             5
```

```
plot.new()
plot.window(xlim = c(Minx-.15*sd(x), Maxx+.15*sd(x)) , ylim = c(0, max(TF)) )
Hist_col = 'green'
for(i in 1:length(TF))
polygon(x = rep(cut_ps[i:(i+1)], each = 2), y = c(0, TF[i], TF[i], 0), col = Hist_col)

axis(1, at = round(cut_ps, 2)); axis(2, at = TF, las = 2)

title(main ='the Main title', sub ='the subtitle', xlab = 'xlab', ylab = 'ylab')
```



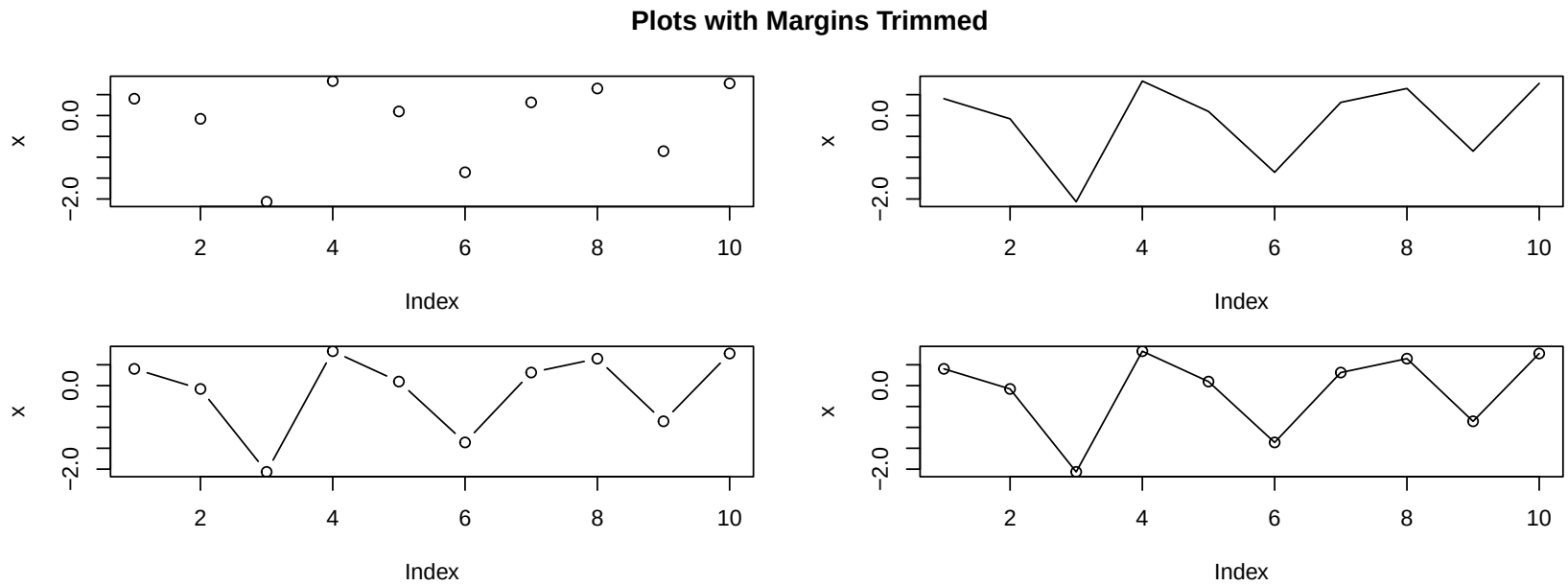
# Control graphical parameters by par function 1

```
par(mfrow = c(2, 2)) # a 2*2 array of figures and combine plots
par(mar = c(3, 4, 0.3, 2.1)) # The size of margins
par(oma = c(3, 4, 2, 4))  # The size of outer margins

x = c(-1, 1)
plot(x, x, type = "n"); text(0, 0, 'Fig1')
plot(x, x, type = "n"); text(0, 0, 'Fig2')
plot(x, x, type = "n"); text(0, 0, 'Fig3')
plot(x,x, type = "n");  text(0, 0, 'Fig4'); box("figure", col =
title(main = "Plots with Margins Trimmed", outer = TRUE)
box(which = 'outer', col = 'blue', lty = "dashed")
```



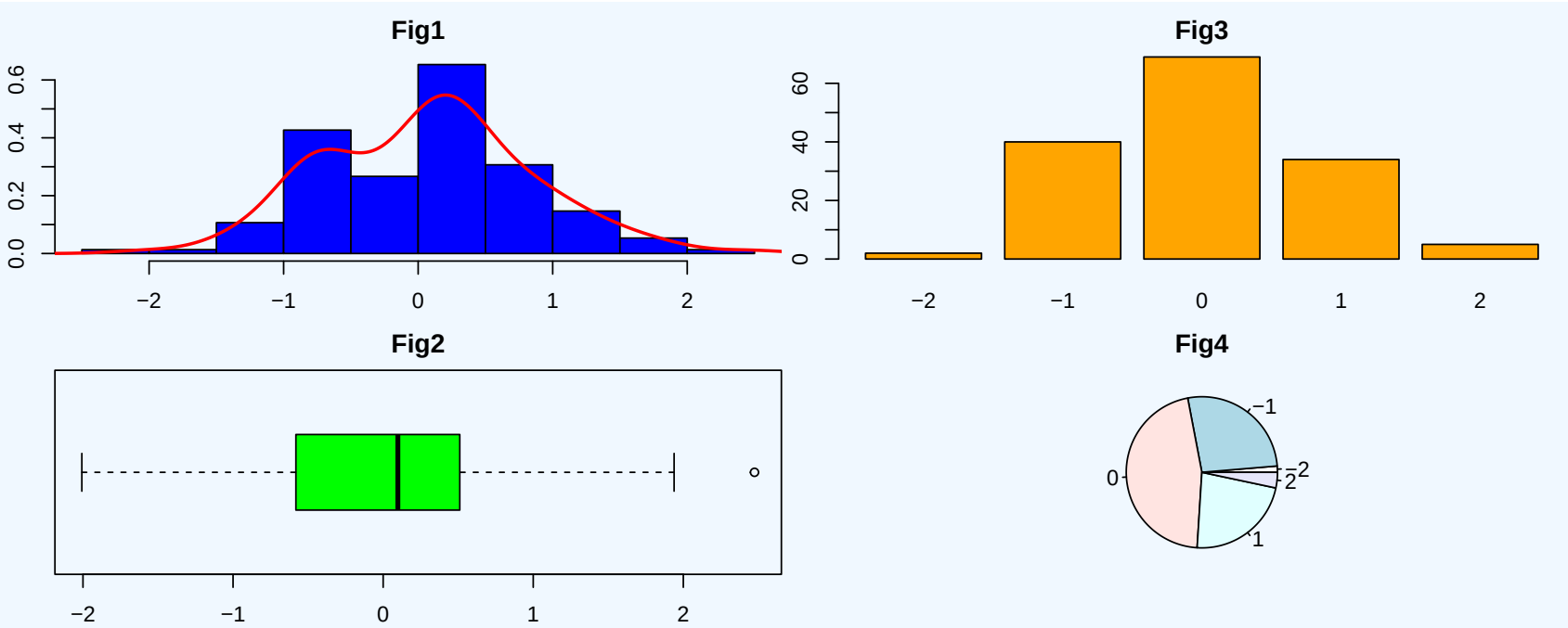
# Control graphical parameters by par function 2

```
par(mfrow = c(2, 2)) # a 2*2 array of figures
par(mar = c(5.1, 4.1, 0.1, 2.1)) # The size of margins
par(oma = c(0, 0, 4, 0))  # The size of outer margins

x = rnorm(10)
plot(x, type = "p"); plot(x, type = "l")
plot(x, type = "b"); plot(x, type = "o")
title(main = "Plots with Margins Trimmed", outer = TRUE)
```



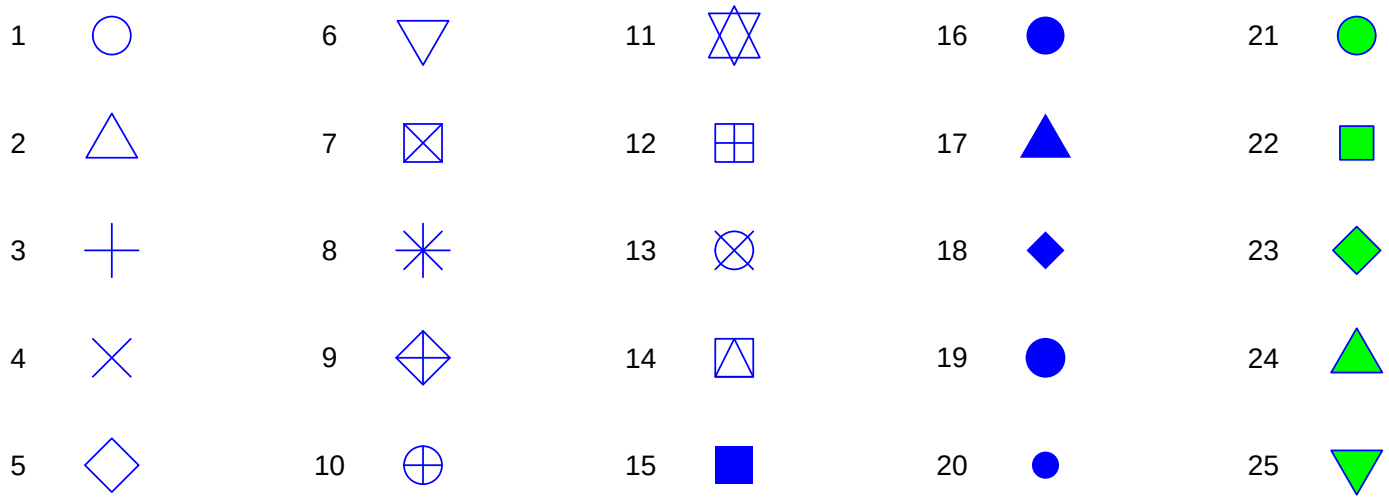
# Control graphical parameters by par function 3

```r
x = rnorm(n = 150, mean = 0, sd = 0.8)

par(mfcol = c(2, 2))  # Two rows, two columns
par(mar = c(2.1, 2.1, 2.1, 0.1))
par(bg = "aliceblue") # Aliceblue background color

hist(x, probability = TRUE, col = 'blue', main = "Fig1")     # Top left
lines(density(x), col = "red", lwd = 2)                      # Add a line graph
boxplot(x, col = 'green', main = "Fig2", horizontal = TRUE) # Bottom left
barplot(table(round(x)), col = 'orange', main = "Fig3")      # Top right
pie(table(round(x)), main = "Fig4")                          # Bottom right
```



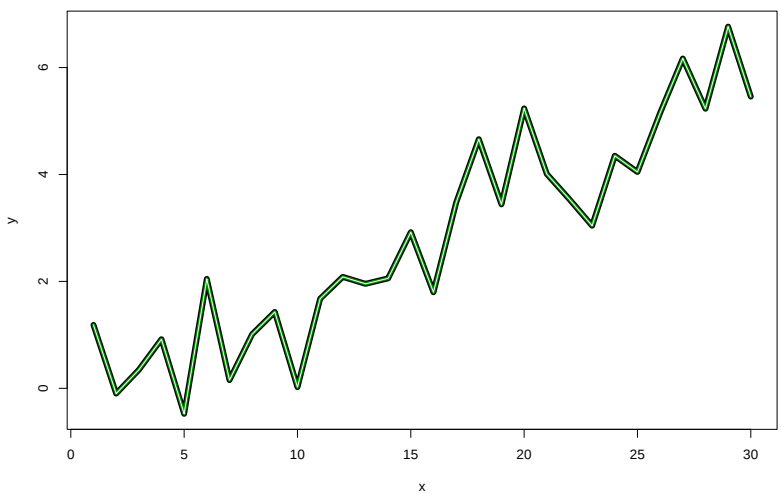
# Point Characters

```
plot.new(); plot.window(xlim = c(4, 26), ylim = c(3, 26) )

 r = rep(1:5 * 5, each = 5); t = rep(5:1 * 5, 5)
 points(r, t,
        pch = 1:25,      # Symbol
        cex = 3,         # Size of the symbol
        col = "blue",    # Border color of the symbol
        bg  = "green")   # Background color of the symbol

 text(r - 1.5, t, 1:25)
```

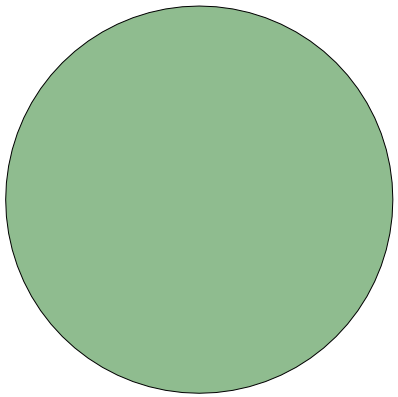


# More on plots and coordinates

```
# Cartesian coordinates
x = 1:30
y = rnorm(30) + x/5
plot(x, y, type = "n")
# add 3 lines
lines(x, y, lwd = 7)
lines(x, y, lwd = 4,
      col = "green4")
lines(x, y, lwd = 1,
      col = "white")
```




```
# polar coordinate
th = seq(0, 2*pi, length = 73)[1:72]
x = cos(th); y = sin(th)
plot.new()
plot.window(xlim = c(-1,1),
            ylim = c(-1,1),
            asp = 1)
polygon(x, y,
        col = "darkseagreen")
```



# More on par function

R graphics are controlled by use of the par function. par makes it possible to control low-level graphics by querying and setting a large set of graphical parameters. Graphical parameters control many features such as:

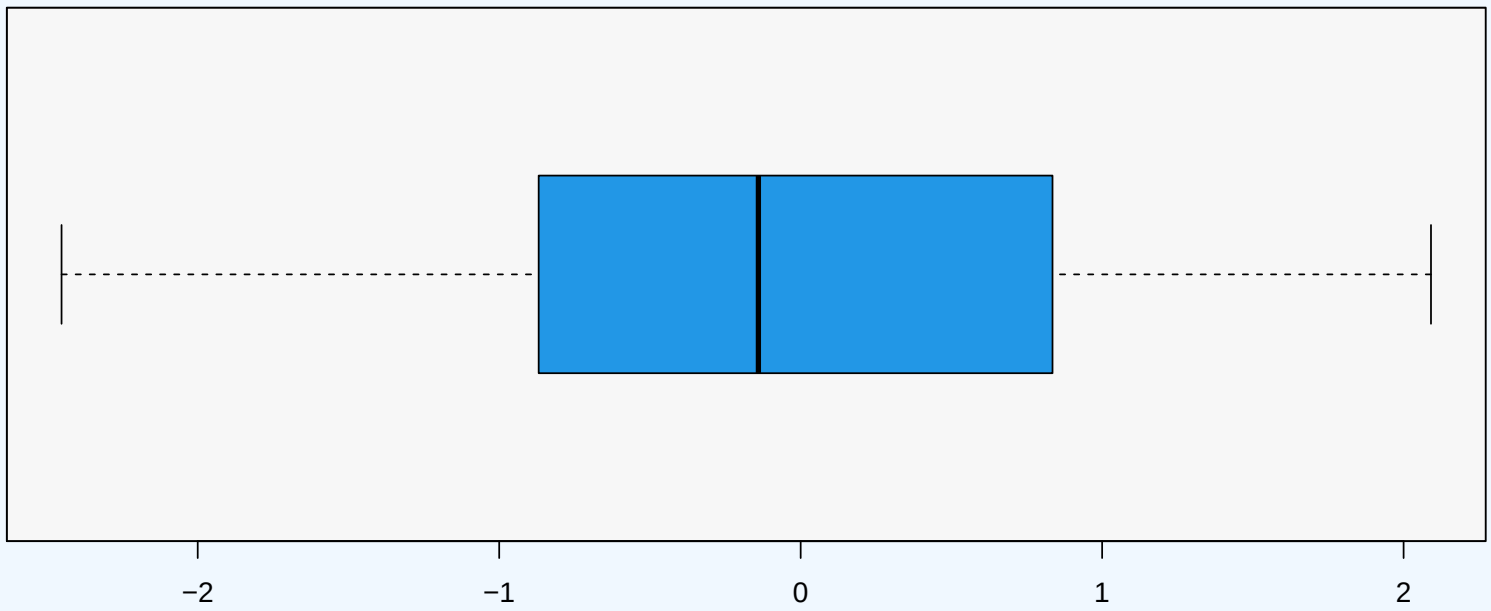
- the layout of figures on the device
- the size of the margins around plots
- the colours, sizes and typefaces of text
- the colour and texture of lines
- the style of axis to be used
- the orientation of axis labels

```
names(par())
```

```
##  [1] "xlog"      "ylog"      "adj"       "ann"       "ask"       "bg"
##  [7] "bty"       "cex"       "cex.axis"  "cex.lab"   "cex.main"  "cex.sub"
## [13] "cin"       "col"       "col.axis"  "col.lab"   "col.main"  "col.sub"
## [19] "cra"       "crt"       "csi"       "cxy"       "din"       "err"
## [25] "family"    "fg"        "fig"       "fin"       "font"      "font.axis"
## [31] "font.lab"  "font.main" "font.sub"  "lab"       "las"       "lend"
## [37] "lheight"   "ljoin"     "lmitre"    "lty"       "lwd"       "mai"
## [43] "mar"       "mex"       "mfcol"     "mfg"       "mfrow"     "mgp"
## [49] "mkh"       "new"       "oma"       "omd"       "omi"       "page"
## [55] "pch"       "pin"       "plt"       "ps"        "pty"       "smo"
## [61] "srt"       "tck"       "tcl"       "usr"       "xaxp"      "xaxs"
## [67] "xaxt"      "xpd"       "yaxp"      "yaxs"      "yaxt"      "ylbias"
```

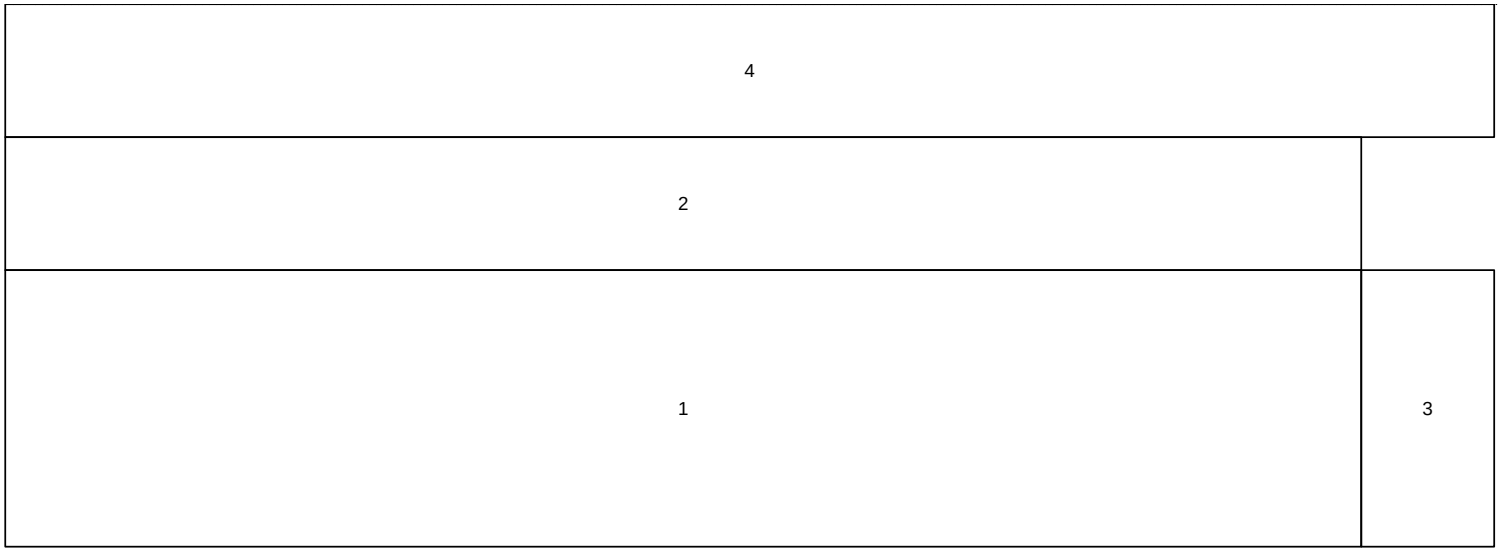
# Background color

```r
set.seed(2); x <- rnorm(100)

par(bg = "aliceblue") # Aliceblue background color

plot.new(); plot.window(xlim = range(x), ylim = c(0.5, 1.5))

rect(par("usr")[1], par("usr")[3],
     par("usr")[2], par("usr")[4],
     col = "#f7f7f7") # Change the plot region color



boxplot(x, col = 4, horizontal = T ,add = T) # Create your plot
```

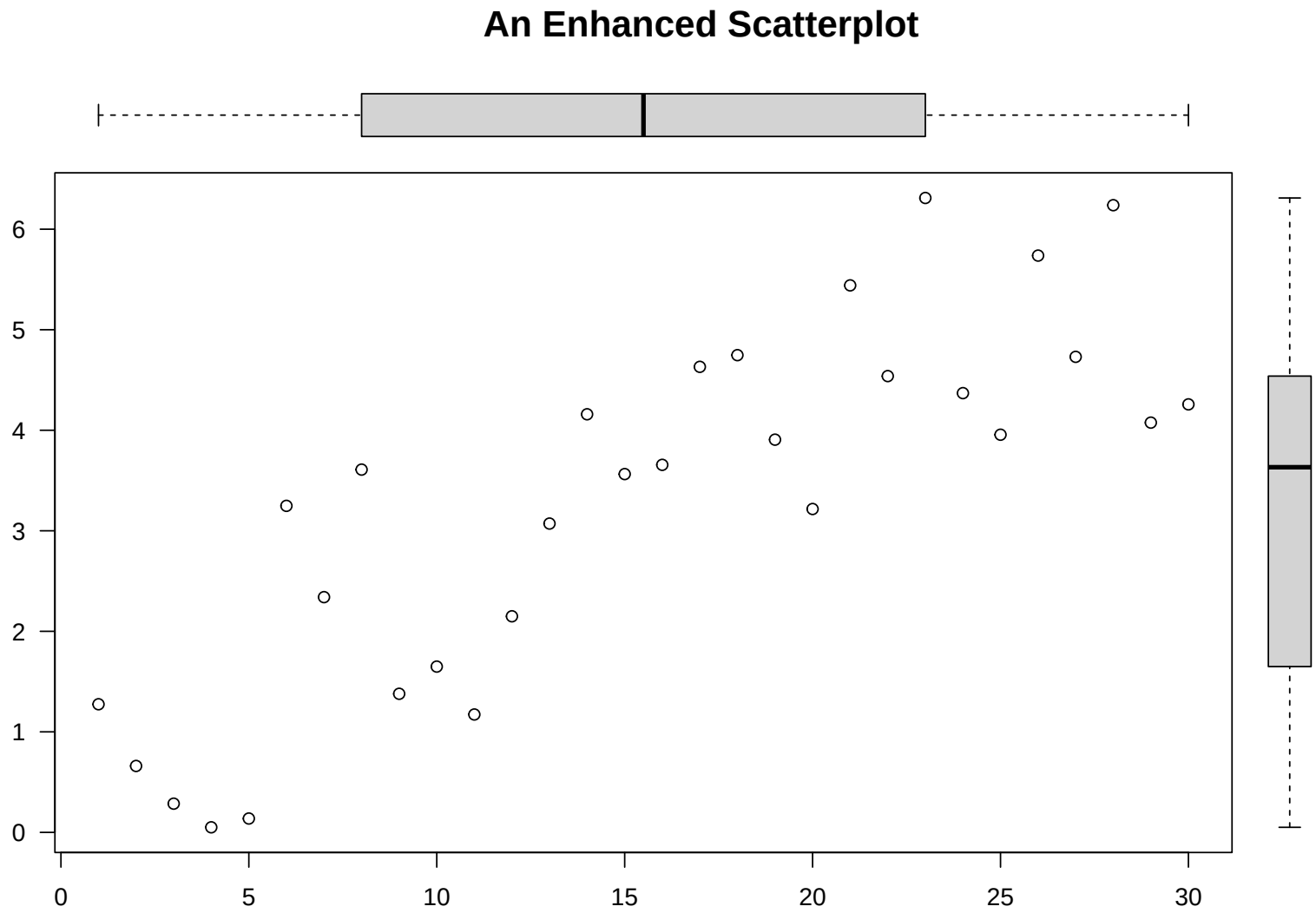


# layout 1

```
layout(rbind(c(0,4,4,0),
             c(0,2,0,0),
             c(0,1,3,0),
             c(0,0,0,0)),
       height = c(lcm(2), lcm(2), 1, lcm(2)),
       width  = c(lcm(2), 1, lcm(2), lcm(1)))
layout.show(4)
box("outer", lty = "dotted")
```



# layout 2




```
##   [1] -5.0000000 -4.7959184 -4.5918367 -4.3877551 -4.1836
##   [7] -3.7755102 -3.5714286 -3.3673469 -3.1632653 -2.9591
```

# Trellis and Lattice

- R also provides an implementation of Trellis plots via the package `lattice` by Deepayan Sarkar, on the basis of the "Grid" graphics system written by Paul Murrell of Auckland.
- Trellis plots embody a number of design principles and these principles are evident in a number of new plot types in Trellis and in the default choice of colors, symbol shapes, and line styles provided by Trellis plots.
- Trellis plots provide a feature known as multipanel conditioning, which creates multiple plots by splitting the data being plotted according to the levels of other variables.

# Why `lattice` system?

- The default appearance of the `lattice` plots is superior in some areas. The default colors and the default data symbols have been deliberately chosen to make it easy to distinguish.
- The arrangement of plot components is more automated in `lattice`. It is usually not necessary to set figure margins manually.
- Legends can be automatically generated by the `lattice` system.
- The output from lattice functions is `grid` output, so many powerful `grid` features are available for annotating, editing, and saving the graphics output.

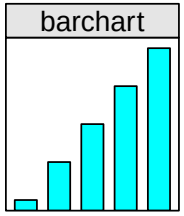
# The lattice system structure in R

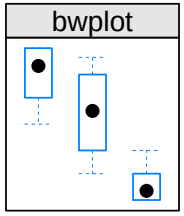
`graph_type(formula, data = )`

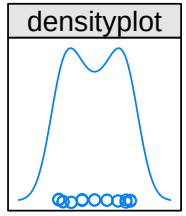
| formula | description |
|---|---|
| `~ y` | Some univariate plot (boxplot, histogram, boxplot, ...) |
| `~ y \| A` | Univariate, separate panels for levels of factor A |
| `~ y \| z` | Univariate, cutting z into discrete ranges |
| `y ~ x` | Bivariate |
| `y ~ x \| A` | Bivariate, separate panels for levels of A |
| `y ~ x \| A + B` | Multiple conditioning variables |
| `y1 + y2 ~ x1 + x2` | Multiple Y and X variables |

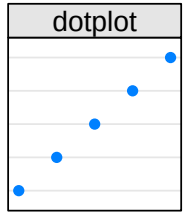
| graph_type | description | graph_type | description |
|---|---|---|---|
| `barchart` | bar chart | `bwplot` | boxplot |
| `cloud` | 3D scatterplot | `contourplot` | 3D contour plot |
| `densityplot` | kernal density plot | `dotplot` | dotplot |
| `histogram` | histogram | `levelplot` | 3D level plot |
| `splom` | scatterplot matrix | `stripplot` | strip plots |
| `xyplot` | scatterplot | `wireframe` | 3D wireframe |

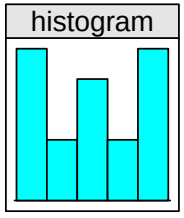
# Types of the the `lattice` system

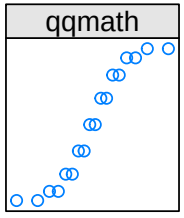


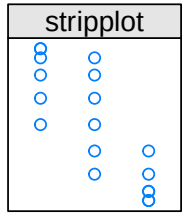



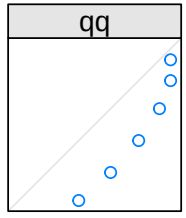



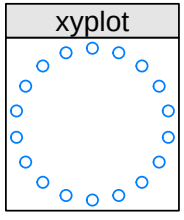



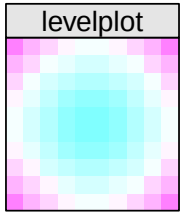



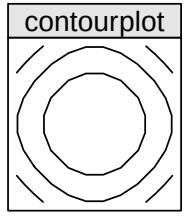



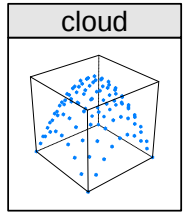



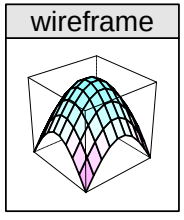



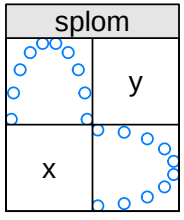



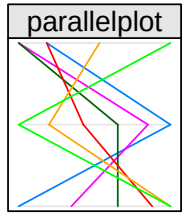



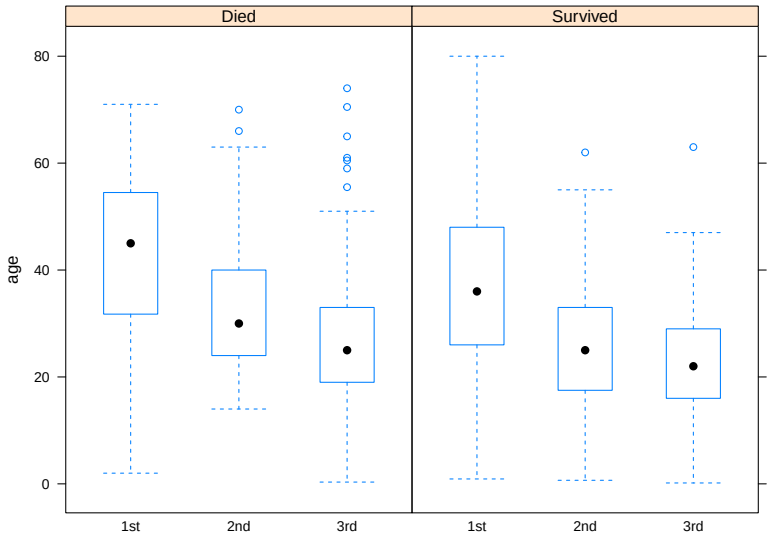



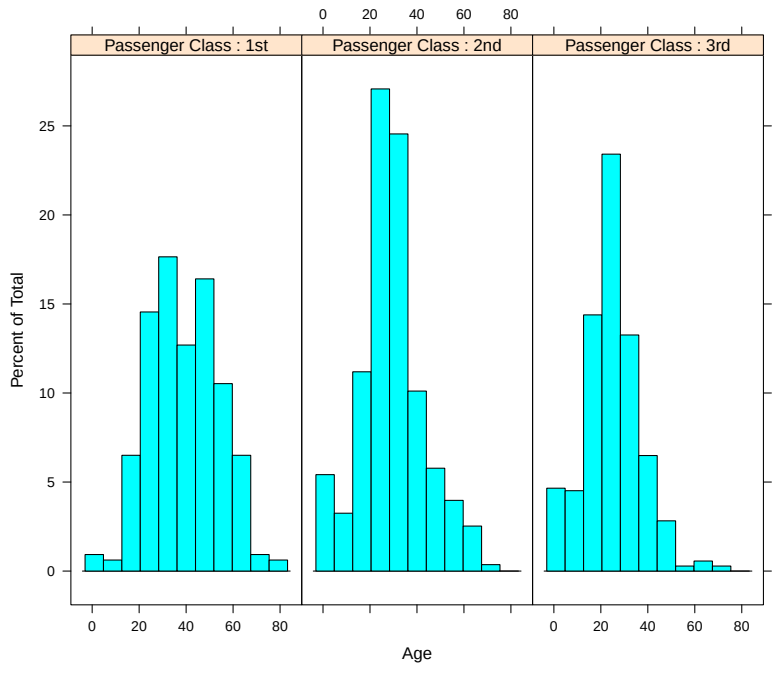



# Histogram and Box-Wishker plot in `lattice`

```
bwplot(age~pclass | survived,
       data = Titanic )
```

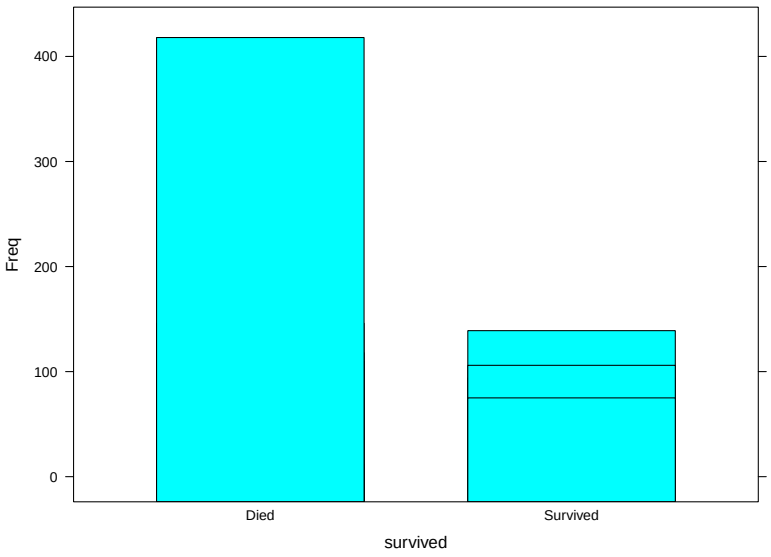



```
# Condition on Passenger Class & Gender
histogram(~ age | pclass,
  data   = Titanic,
  layout = c(3,1),
   strip = strip.custom(
           strip.names = TRUE,
            var.name   =
             "Passenger Class"),
    xlab = "Age")
```

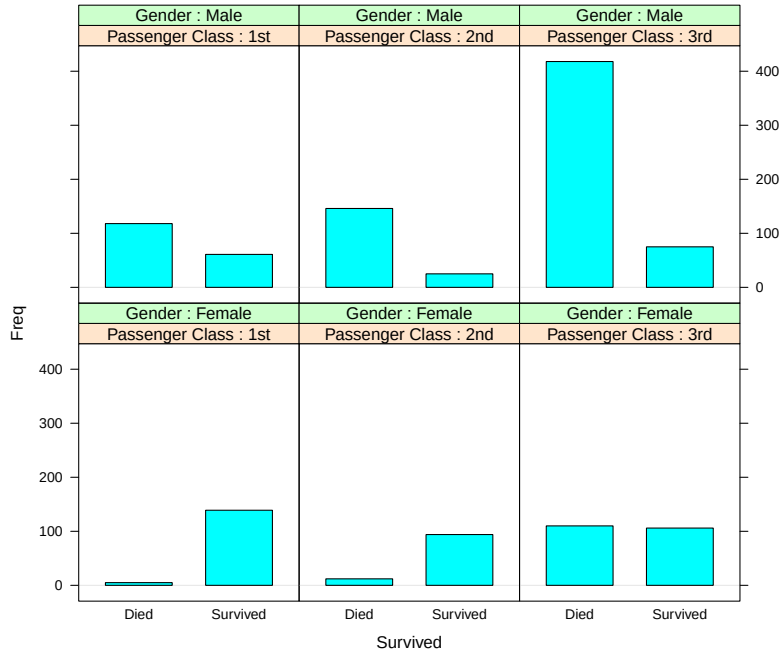


# Barchart

```
surv.tab = xtabs(~survived +
                   pclass +
                   Gender,
                   data = Titanic)
surv.df = as.data.frame(surv.tab)

# In barchart you need Freq ~ X
barchart(Freq ~ survived,
         data = surv.df,
        xlab = "survived")
```

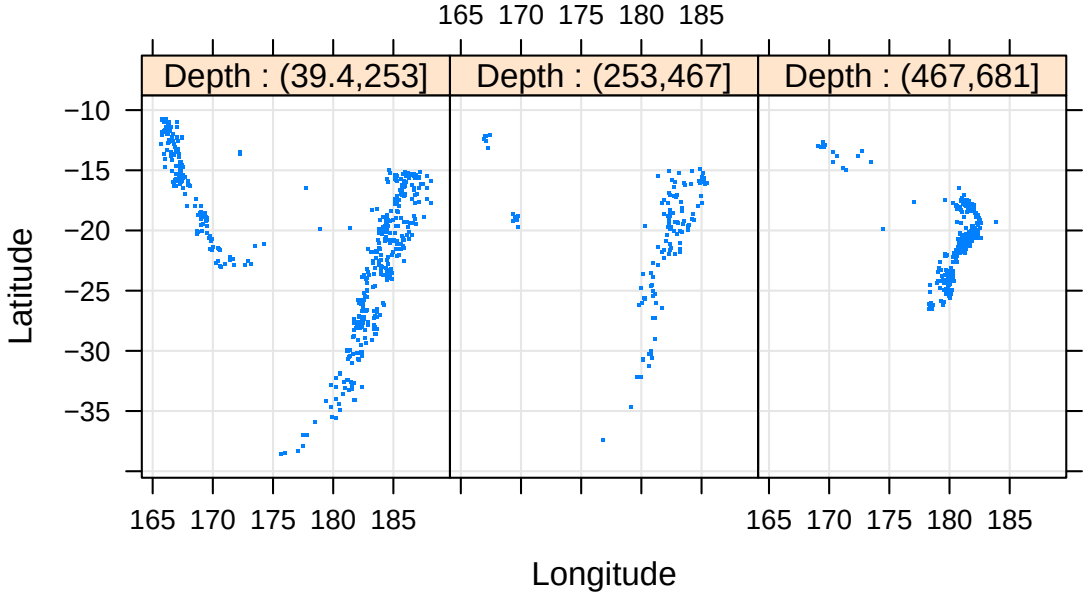



```
# Condition on Passenger Class & Gender
barchart(
  Freq ~ survived | pclass * Gender,
  data   = surv.df, origin = 0,
   strip = strip.custom(
           strip.names = TRUE,
            var.name   =
      c("Passenger Class", "Gender")),
    xlab = "Survived")
```



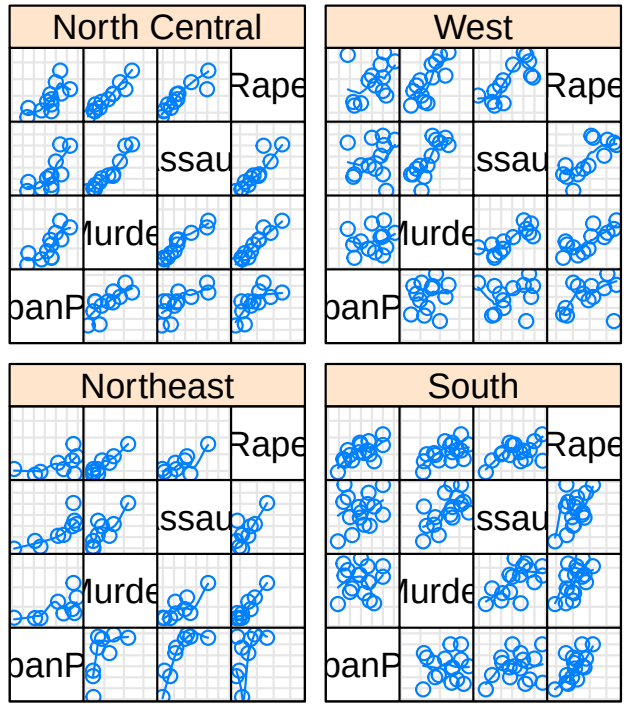
# xyplot: Scatter plot in lattice

```
library(lattice)
xyplot(lat ~ long | cut(depth, 3),
       data = quakes, aspect = "iso", pch = ".", cex = 2,
       type = c("p", "g"), xlab = "Longitude", ylab="Latitude",
       strip = strip.custom(strip.names = TRUE,
                            var.name = "Depth"))
```



# splom: Scatter Plot Matrix

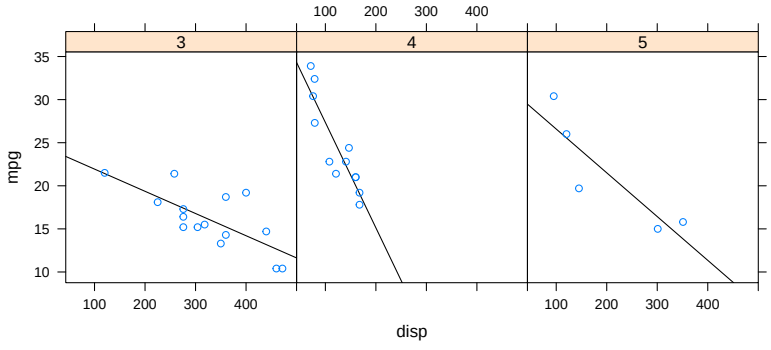
```
splom(~USArrests[c(3, 1, 2, 4)] | state.region,
      pscales = 0, type = c("g", "p", "smooth"),
      layout = c(2,2))
```



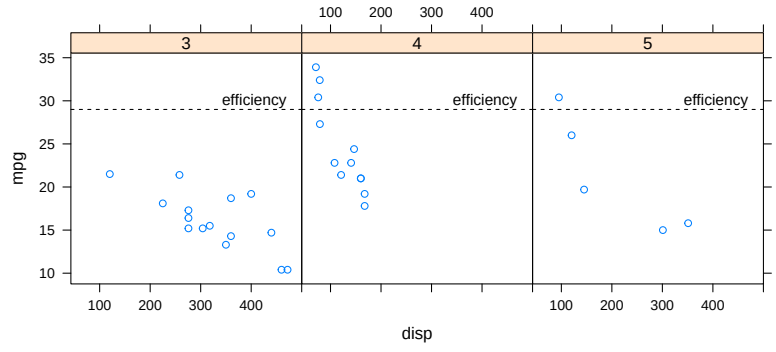

Scatter Plot Matrix

# panel function in lattice

```
xyplot(mpg ~ disp | factor(gear),
                    data = mtcars,
  layout=c(3, 1), aspect=1,

  panel = function(x, y, ...) {
  panel.lmline(x, y)
  panel.xyplot(x, y, ...)
})
```




```
xyplot(mpg ~ disp | factor(gear),
       data=mtcars,
layout=c(3, 1), aspect = 1,
panel = function(...) {
panel.xyplot(...)
panel.abline(h = 29, lty = "dashed")
panel.text(470,29.5, "efficiency",
           adj = c(1, 0), cex= 0.9)
})
```



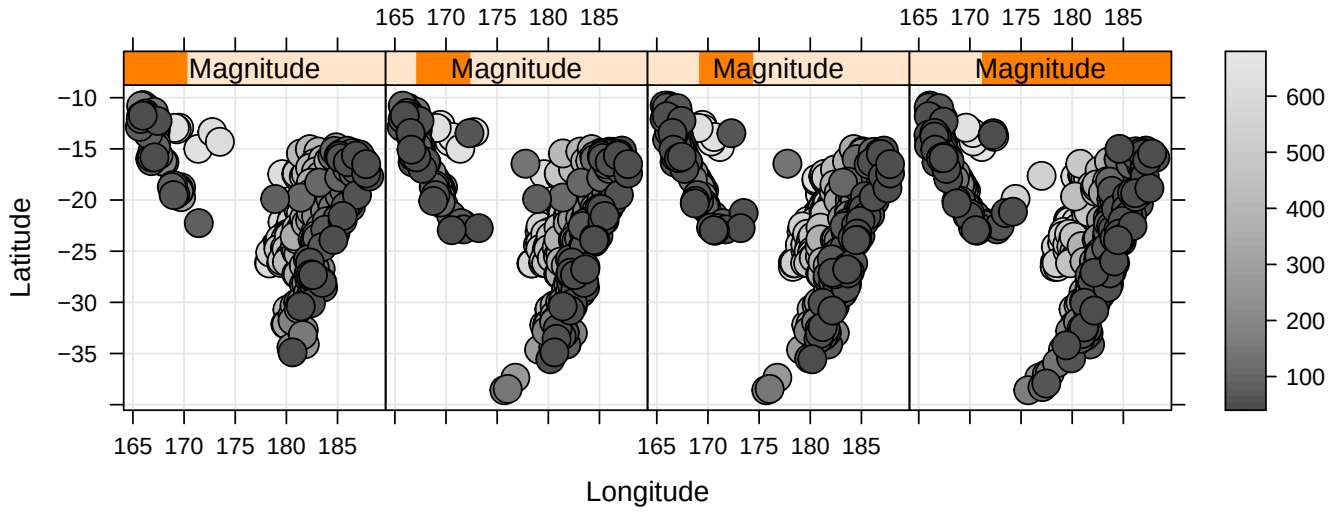
# Most common used `panels`

A selection of predefined panel functions for adding graphical output to the panels of lattice plots.

| Function | Description |
|---|---|
| `panel.points()` | Draw data symbols at locations (x, y) |
| `panel.lines()` | Draw lines between locations (x, y) |
| `panel.segments()` | Draw line segments between (x0, y0) and (x1, y1) |
| `panel.arrows()` | Draw line segments and arrowheads to the end(s) |
| `panel.polygon()` | Draw one or more polygons with vertices (x, y) |
| `panel.text()` | Draw text at locations (x, y) |
| `panel.abline()` | Draw a line with intercept a and slope b |
| `panel.curve()` | Draw a function given by expr |
| `panel.rug()` | Draw axis ticks at x- or y-locations |
| `panel.grid()` | Draw a (gray) reference grid |
| `panel.loess()` | Draw a loess smooth through (x, y) |
| `panel.violin()` | Draw one or more violin plots |
| `panel.smoothScatter()` | Draw a smoothed 2D density of (x, y) |

# panel

```
depth.col = gray.colors(100)[cut(quakes$depth, 100, label = FALSE)]
depth.ord = rev(order(quakes$depth))
quakes$Magnitude=equal.count(quakes$mag,4); quakes$color=depth.col; quakes.ordered=quakes[depth.ord,]
depth.breaks = do.breaks(range(quakes.ordered$depth), 50)

xyplot(lat ~ long | Magnitude, data = quakes.ordered,
       aspect = "iso", groups = color, cex = 2, col = "black",
       panel = function(x, y, groups, ..., subscripts) {
           fill <- groups[subscripts]
           panel.grid(h = -1, v = -1)
           panel.xyplot(x, y, pch = 21, fill = fill, ...)},

     legend = list(right = list(fun = draw.colorkey, args =
                                list(key = list(col = gray.colors, at = depth.breaks)))),
       xlab = "Longitude", ylab = "Latitude")
```



# Grammar of Graphics: ggplot2 1

Leland Wilkinson's Grammar of Graphics provides another completely different paradigm for producing statistical plots and this approach to plotting has been implemented for R by Hadley Wickham's ggplot2 package.

- One advantage of this package is that it makes it possible to create a very wide variety of plots from a relatively small set of fundamental components.
- The ggplot2 package also has a feature called facetting, which is similar to lattice's multipanel plots.

# Grammar of Graphics: ggplot2 2

## Every graph can be described as a combination of

- data: a data frame: quantitative, categorical;
- aesthetic: mapping of variables into visual properties: x, y, size, color, ...
- geometric objects ("geom"): points, lines, areas, arrows, ...
- layers: graph elements combined with "+"
  - coordinate system ("coord"): Cartesian, polar, log, map,

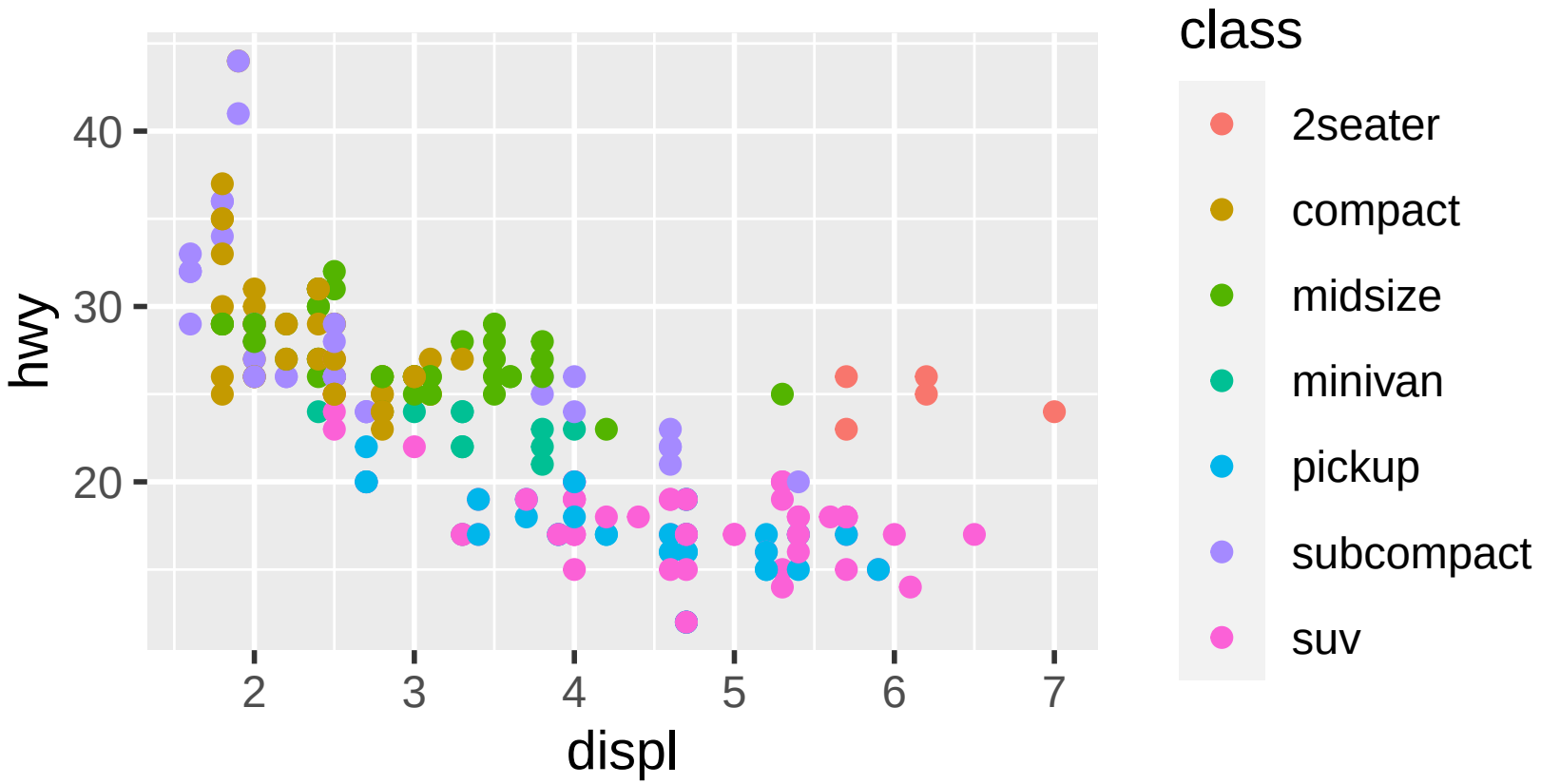
## And some more

- statistical transformations ("stat") – data summaries: mean, sd, binning & counting, ...
- scales: legends, axes
- position adjustments: jitter, dodge, stack, ...
- faceting: small multiples or conditioning to break a plot

# Two Column Layout

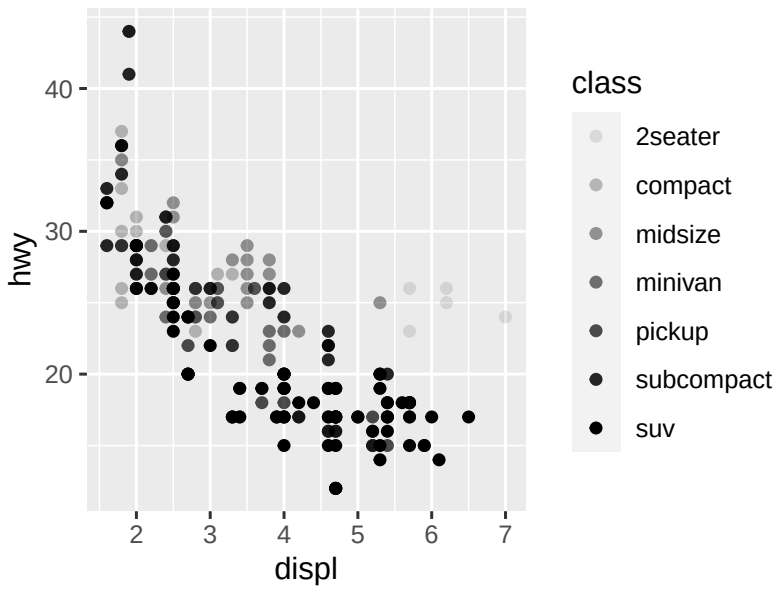
```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

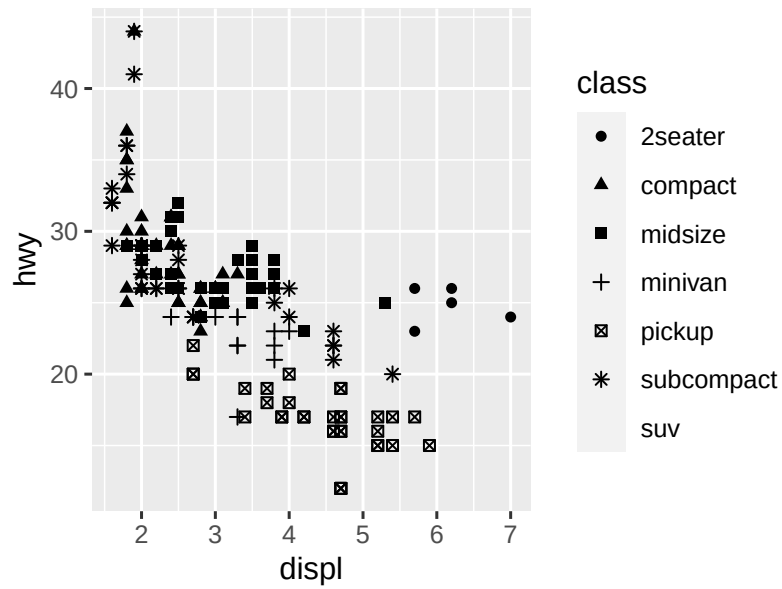
```
ggplot(data = mpg) +
  geom_point(mapping =
               aes(x = displ, y = hwy, color = class))
```



# Two Column

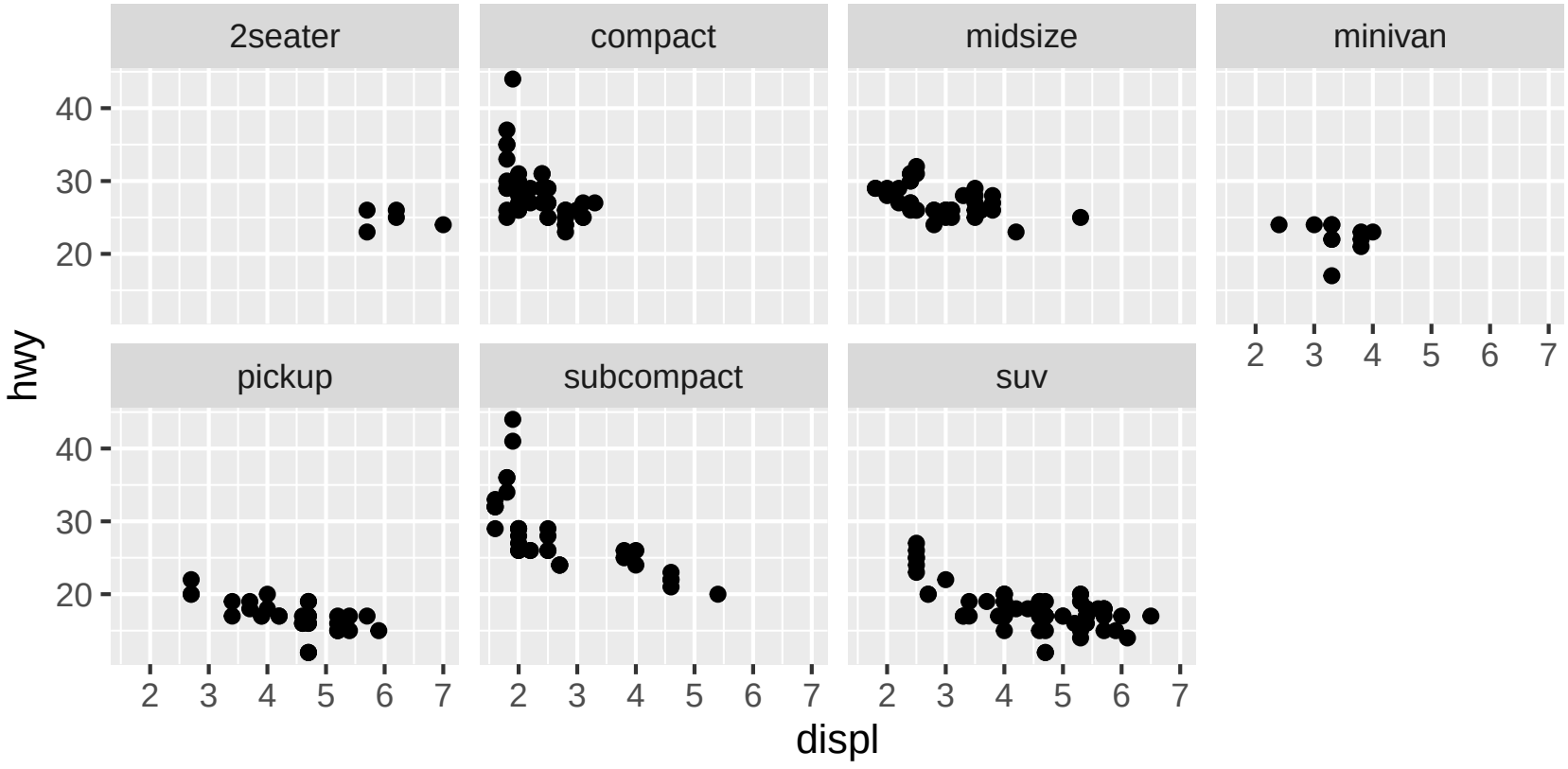
```
# Left
ggplot(data = mpg) +
  geom_point(mapping =
                aes(x = displ, y = hwy, alpha = class))
# Right
ggplot(data = mpg) +
  geom_point(mapping =
                aes(x = displ, y = hwy, shape = class))
```



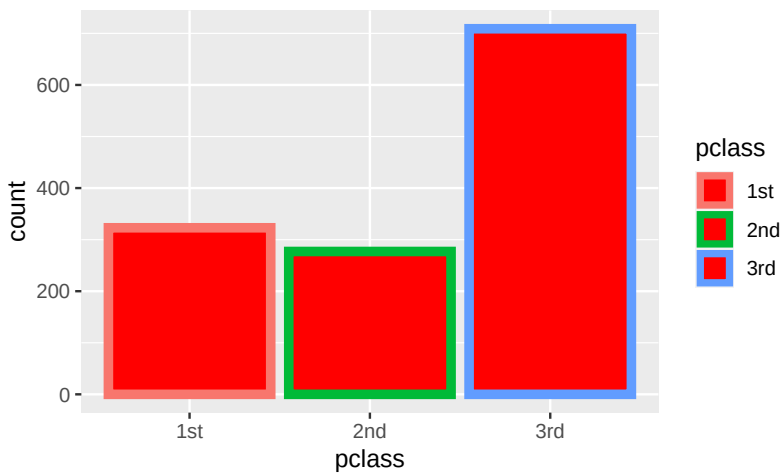
# facet

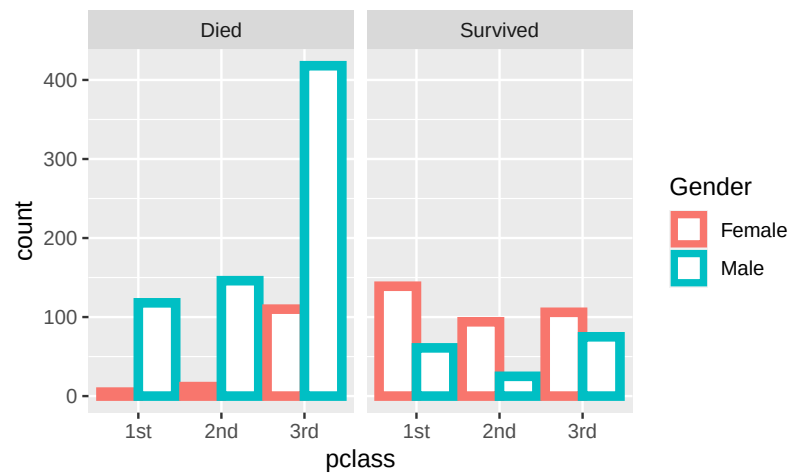
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



# geom_bar 1

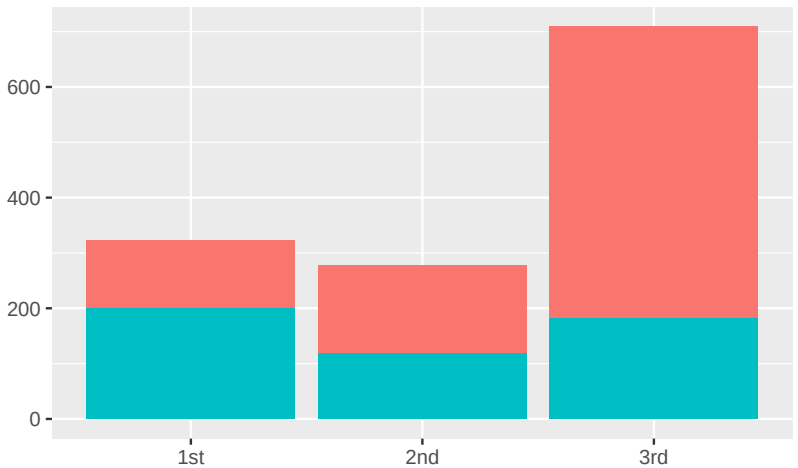
```r
#Left
ggplot(data = Titanic) +
  geom_bar(mapping = aes(x = pclass, colour = pclass),
           fill    = 'red',
           lwd     = 2)

# Right
ggplot(data    = Titanic,
       mapping = aes(x= pclass, colour= Gender, fill= pclass))+
  geom_bar(lwd = 2, fill = "white", position = "dodge") +
  facet_wrap(~ survived)
# position can be 'dodge' and fill, 'stack' is the default for b
```

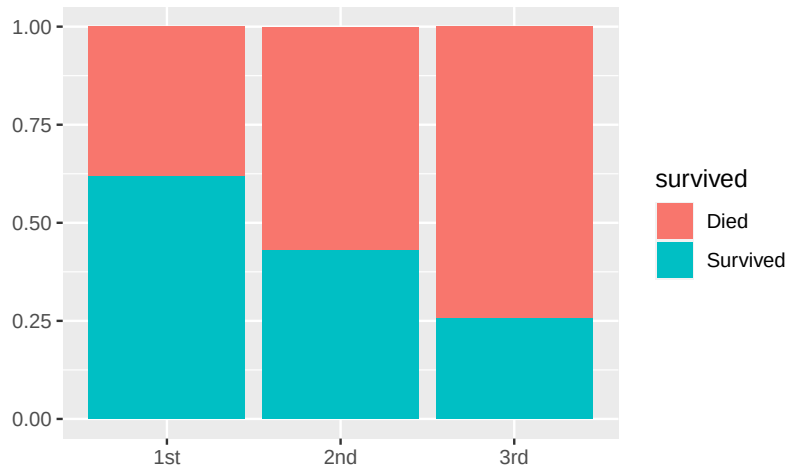


# geom_bar 2

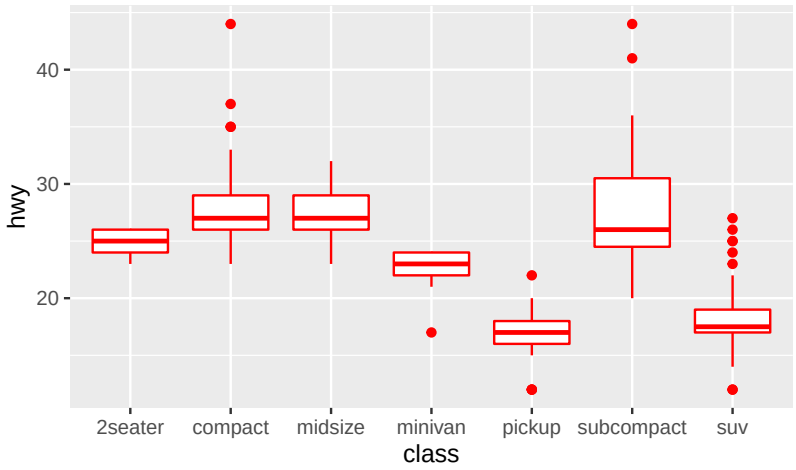
```
dplot = ggplot(Titanic, aes(pclass, fill = survived)) +
  xlab(NULL) + ylab(NULL) + theme(legend.position = "none")

# position stack is the default for bars, so `geom_bar()`
# is equivalent to `geom_bar(position = "stack")`.
#Left
dplot + geom_bar() +
  theme(plot.background = element_rect(fill = "lightblue"))
# Right
dplot + geom_bar(position = "fill") +
        theme(legend.position = "right") # the default
```

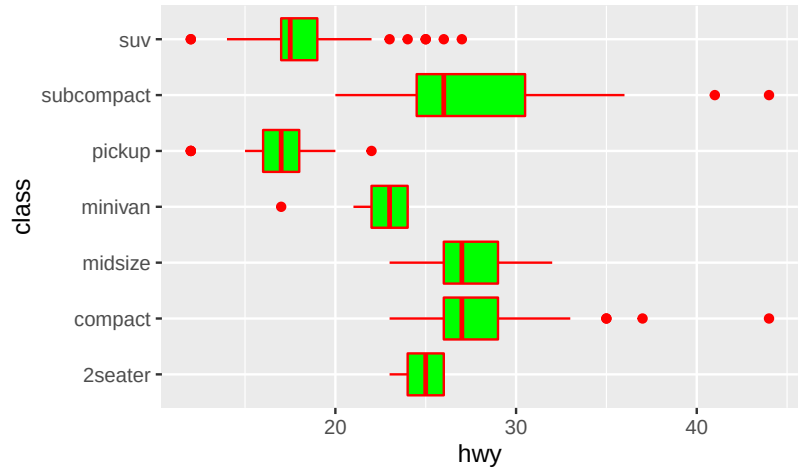


# geom_box

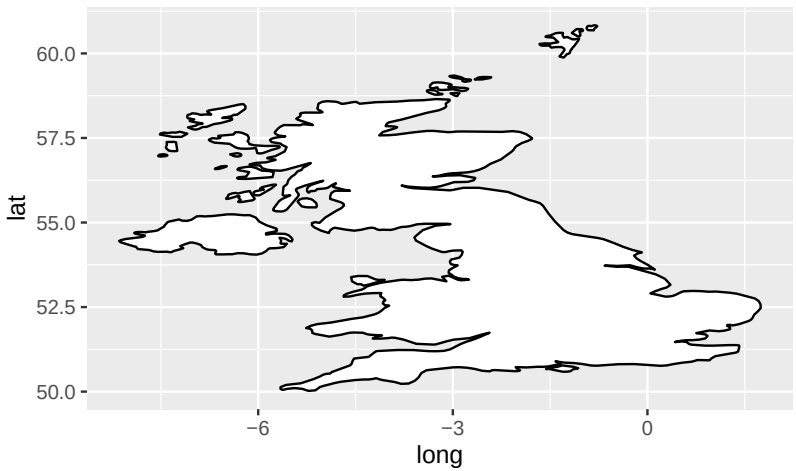
```
#Left
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot(col = 'red')
#Right
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot(col = 'red', fill = 'green') +
  coord_flip()
```

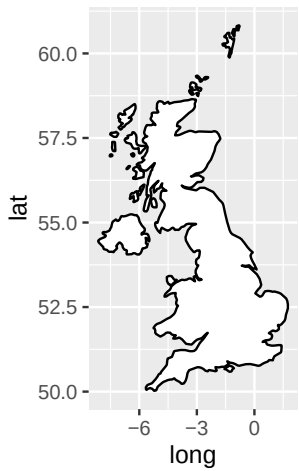


# map

```
library(maps); library(tidyverse)

UK <- map_data("world") %>% filter(region=="UK")
#Left
ggplot(UK, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black")
#Right
ggplot(UK, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black") +
  coord_quickmap()
```

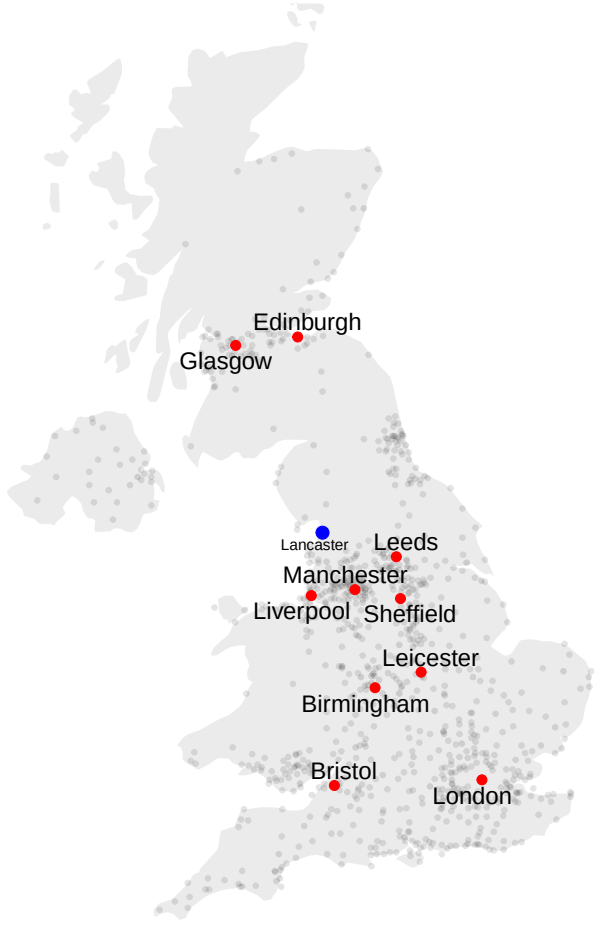


# map 2

```
# Get a data frame with longitude, latitude, and size of bubbles
library(ggrepel)
UK_city <- world.cities %>% filter(country.etc == "UK")
ggplot(data = UK) +
  geom_polygon(aes(x=long, y = lat, group = group),
                fill="grey", alpha=0.3) +
  geom_point( data = UK_city, aes(x=long, y=lat)) +
  theme_void() + ylim(50,59) + coord_map()

# Second graphic with names of the 10 biggest cities
ggplot(data = UK) +
  geom_polygon( aes(x=long, y = lat, group = group),
                fill="grey", alpha=0.3) +
  geom_point(UK_city, aes(x=long, y=lat, alpha=pop)) +
  geom_text_repel(UK_city %>% arrange(pop) %>% tail(10),
                  aes(x=long, y=lat, label=name), size=5) +
  geom_point(UK_city %>% arrange(pop) %>% tail(10),
             aes(x=long, y=lat), color="red", size=3) +
  theme_void() + ylim(50,59) + coord_map() +
  theme(legend.position="none")
```

# map 3


Edinburgh
Glasgow
Lancaster
Leeds
Manchester
Liverpool
Sheffield
Leicester
Birmingham
Bristol
London

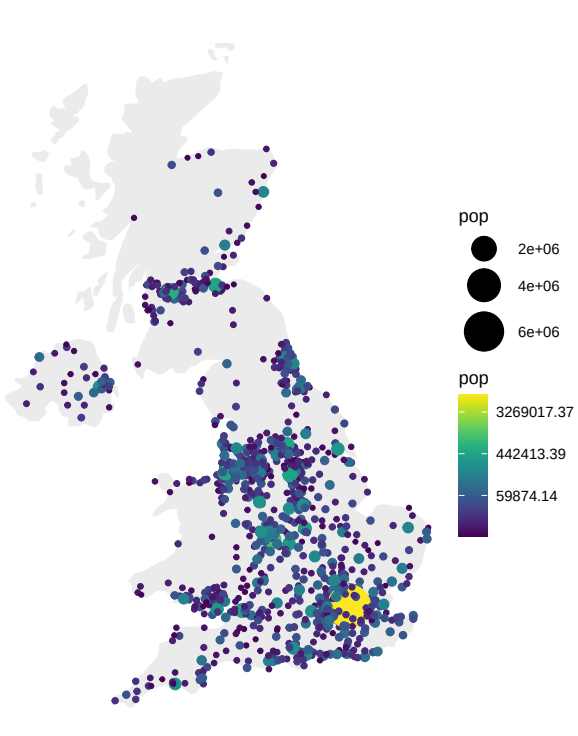
# map 4

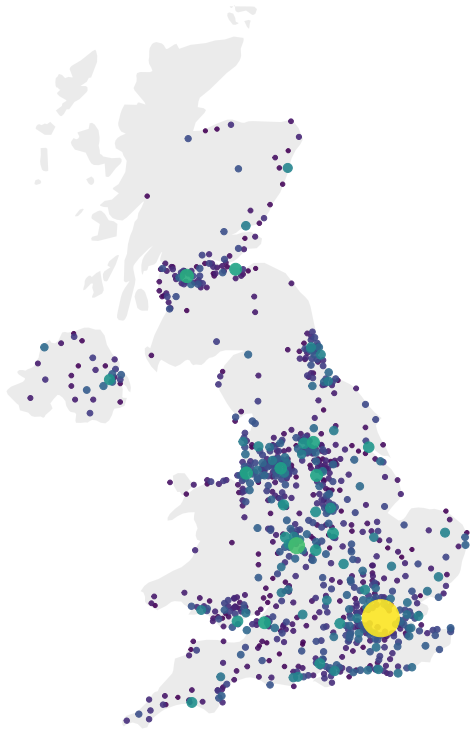
```
# virids package for the color palette
library(viridis)

# Left: use size and color
ggplot() +
  geom_polygon(data = UK, aes(x=long, y = lat, group = group), fill="grey", alpha=0.3) +
  geom_point( data=UK_city, aes(x=long, y=lat, size=pop, color=pop)) +
  scale_size_continuous(range=c(1,12)) +
  scale_color_viridis(trans="log") +
  theme_void() + ylim(50,59) + coord_map()
```

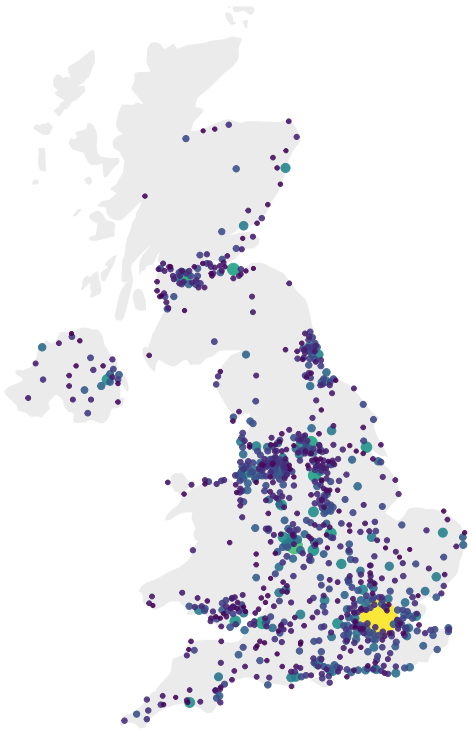
```
# Center: reorder your dataset first! Big cities appear later = on top
UK_city %>%
 arrange(pop) %>%
 mutate( name=factor(name, unique(name))) %>%
 ggplot() +
    geom_polygon(data = UK, aes(x=long, y = lat, group = group), fill="grey", alpha=0.3) +
    geom_point( aes(x=long, y=lat, size=pop, color=pop), alpha=0.9) +
    scale_size_continuous(range=c(1,12)) +
    scale_color_viridis(trans="log") +
    theme_void() + ylim(50,59) + coord_map() + theme(legend.position="none")
```

```
# Right: just use arrange(desc(pop)) instead
UK_city %>%
 arrange(desc(pop)) %>%
 mutate( name=factor(name, unique(name))) %>%
 ggplot() +
    geom_polygon(data = UK, aes(x=long, y = lat, group = group), fill="grey", alpha=0.3) +
    geom_point( aes(x=long, y=lat, size=pop, color=pop), alpha=0.9) +
    scale_size_continuous(range=c(1,12)) +
    scale_color_viridis(trans="log") +
    theme_void() + ylim(50,59) + coord_map() + theme(legend.position="none")
```

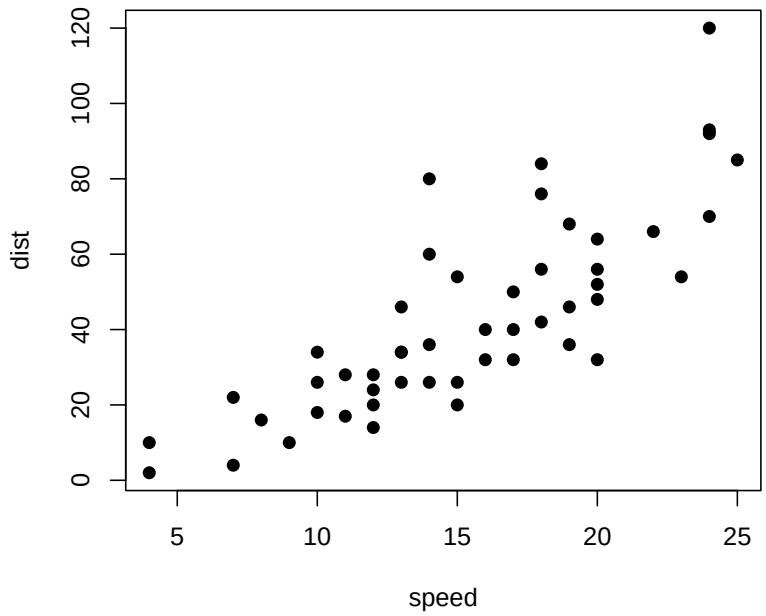
# map 4_2





# 3 columns




The figure on the left-hand side shows the `cars` data. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

# A psychological case study

Topic: The effects of childhood sexual abuse on adult females.

- Measurements:
  - Post-traumatic stress disorder on a standard scale (ptsd)
  - Childhood physical abuse on a standard scale (cpa)
  - Childhood sexual abuse in two groups: Abused or Not-Abused (csa)
- Research question:

1. Is there any relationship between ptsd and cpa? If so, what model do you suggest?
2. Does the csa have impact on ptsd?
3. Do the csa and cpa together any affect on ptsd?

- Your task in each case :
  1. First, Summaries the data by using descriptive statistic
  2. In each case, Specify a model and interpret it.

# childhood sexual abuse data

```
library(faraway)
```

```
##
## Attaching package: 'faraway'

## The following object is masked from 'package:maps':
##
##     ozone

## The following objects are masked from 'package:survival':
##
##     rats, solder

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```
data("sexab")
library(GGally)
```

# Slide with ggpairs3

```
ggpairs(sexab, aes(colour = csa))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `
## `stat_bin()` using `bins = 30`. Pick better value with `
```

# My topic for this slide

| head | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |