E03

R資料視覺化
ggplot2
統計繪圖套件

吳漢銘

國立臺北大學 統計學系

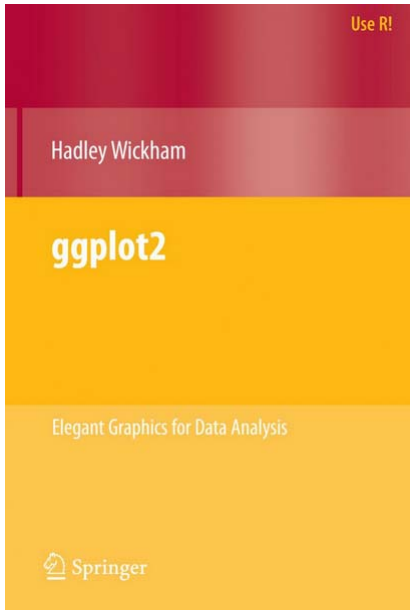
http://www.hmwu.idv.tw

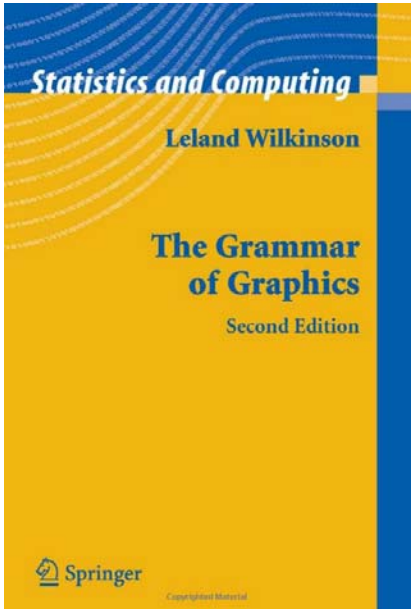
What is ggplot2

ggplot2 [編輯]

維基百科，自由的百科全書

ggplot2是統計程式語言R的一個數據可視化繪圖包。ggplot2由Hadley Wickham在2005年創造。它具現了利蘭·威爾金森所著《圖標的語法 — 一個數據可視化通用框架》（*Grammar of Graphics* — a general scheme for data visualization）中將圖形分解為語素（如尺度、圖層）的思想。ggplot2可以作為R語言基礎繪圖包的替代，同時ggplot2預設有多種印刷及網頁尺寸。自2005年以來，ggplot2已經發展成為最受歡迎的R包之一。[2][3]





- High-level graphics system developed by Hadley Wickham.
- Implements grammar of graphics from Leland Wilkinson.
- Streamlines many graphics workflows for complex plots.
- Syntax centered around main **`ggplot`** function.
- Simpler **`qplot`** function provides many shortcuts.

The principle of a ggplot

- The principle that a plot:

  Plot = data + aesthetics + geometry

  - **`data`**: a data frame (dataset).
  - **`aesthetics`**:
    - indicates **`x`** and **`y`** variables,
    - tells R how data are displayed in a plot.
      (e.g. color, size and shape of points, the height of bars etc.)
  - **`geometry`**: to the type of graphics
    (bar chart, histogram, box plot, line plot, density plot, dot plot etc.)

General ggplot syntax

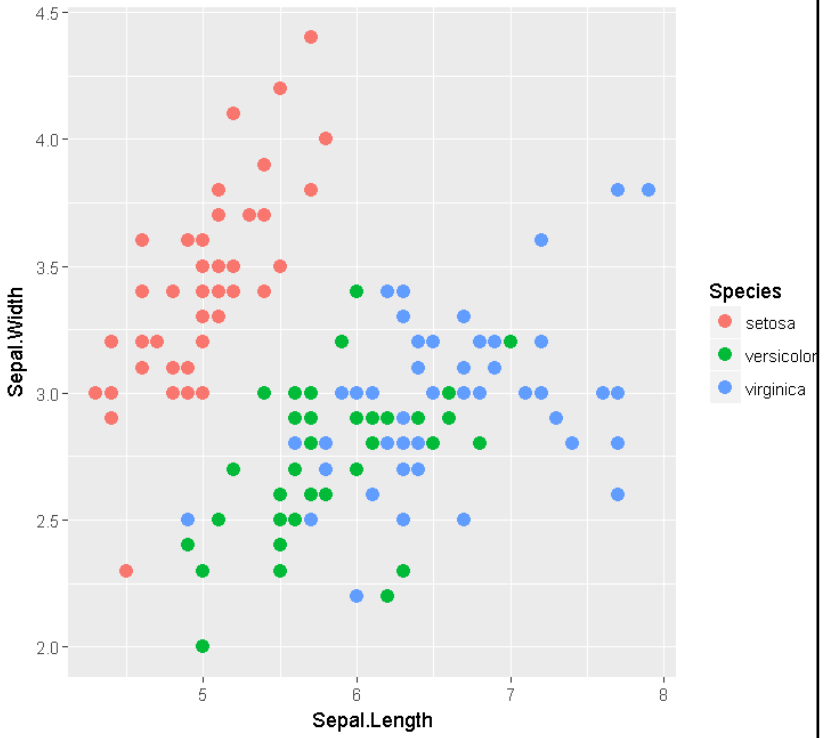
General ggplot syntax

```
ggplot(data, aes(...)) + geom() + ... + stat() + ...
```

```
> install.packages("ggplot2")
> library(ggplot2)
> ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
        geom_point(size=3)
```

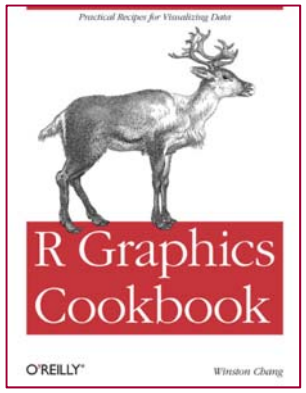
Other important parts of plot:

- **Faceting** implies the same type of graph can be applied to each subset of the data. (e.g, for variable gender, creating 2 graphs for male and female.)
- **Annotation** lets you to add text to the plot.
- **Summary Statistics** allows you to add descriptive statistics on a plot.
- **Scales** are used to control x and y axis limits.



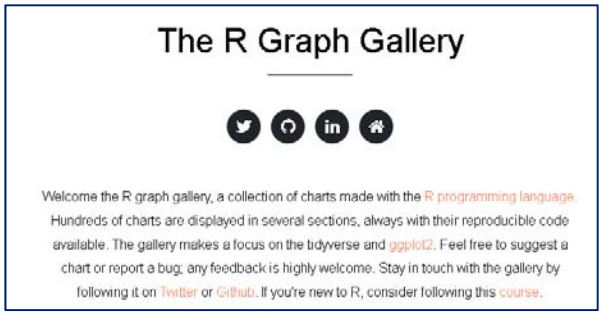
Why ggplot2?

- More elegant & compact code than base graphics.
- More aesthetically pleasing than base graphics.
- Very powerful for exploratory analysis.
- Supports a continuum of expertise.
- Easy to get started, plenty of power for complex figures.
- Publication-quality figures.
- Excellent themes can be created with a single command.
- Its colors are nicer and more pretty than the usual graphics.
- Easy to visualize data with multiple variables.
- Provides a platform to create simple graphs providing plethora of information.

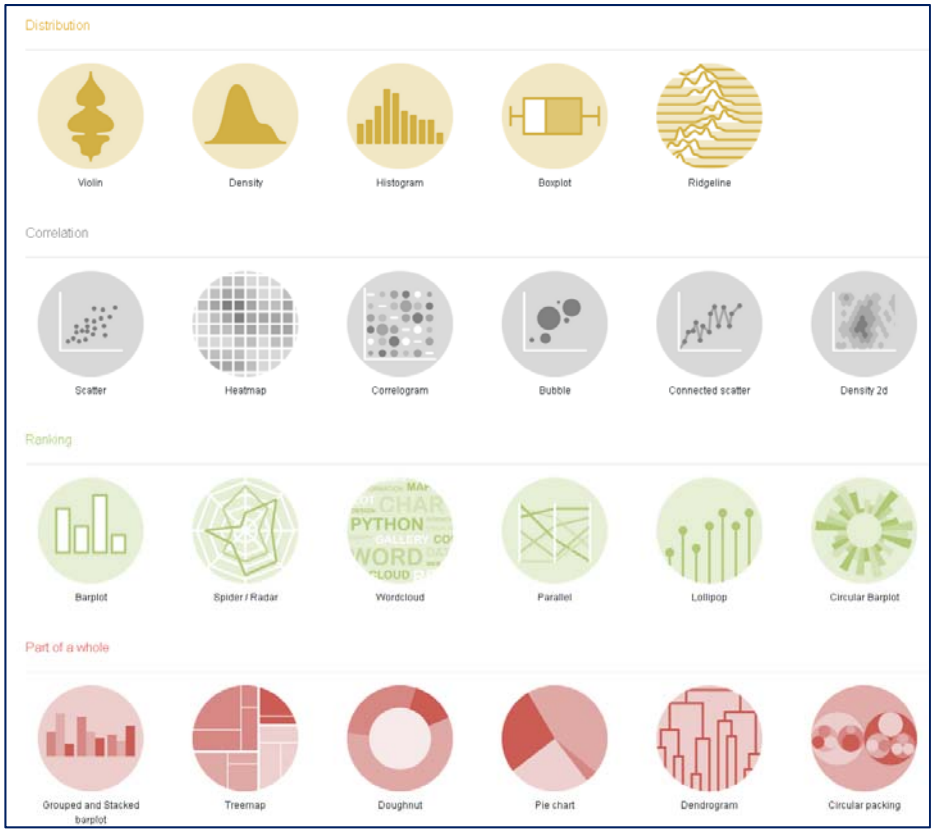


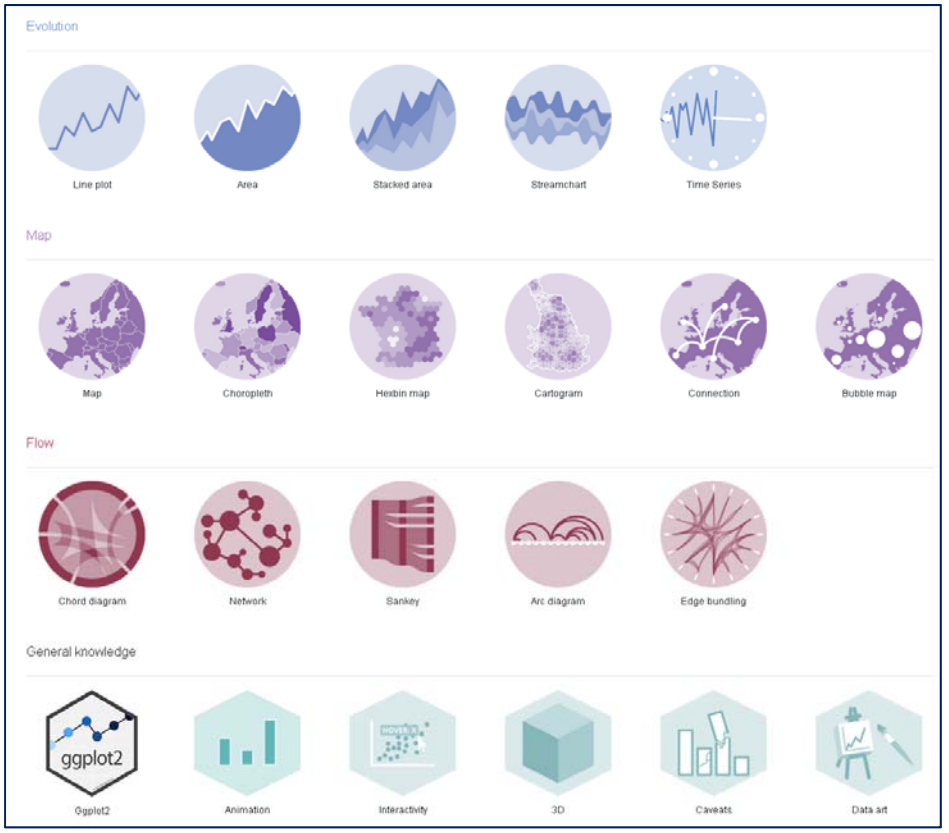
- Manual: http://had.co.nz/ggplot2/
- Introduction: http://www.ling.upenn.edu/~joseff/rstudy/summer2010_ggplot2_intro.html
- Book: http://had.co.nz/ggplot2/book/
- R Graphics Cookbook: http://www.cookbook-r.com/Graphs/

The R Graph Gallery

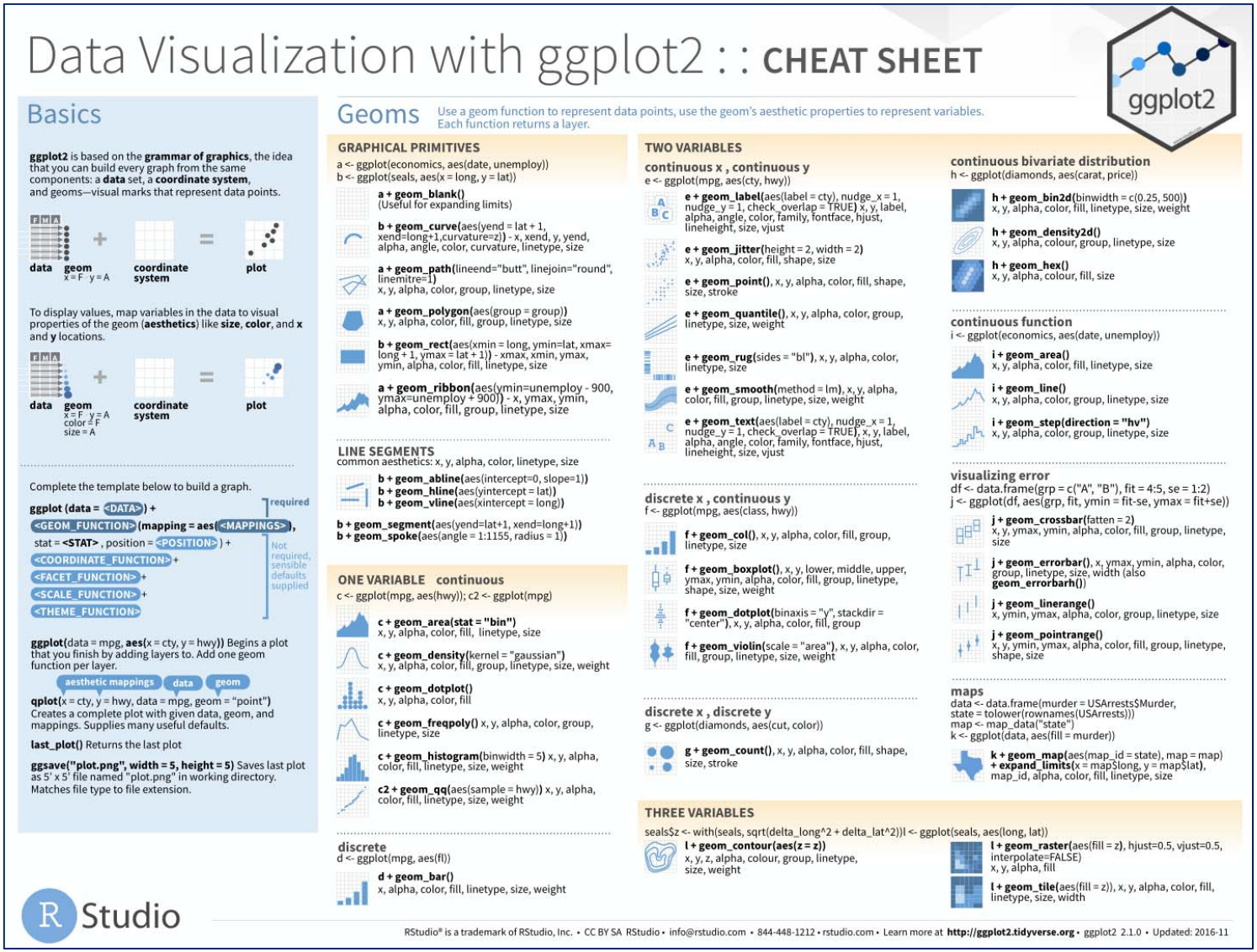


https://www.r-graph-gallery.com/index.html





Data Visualization with ggplot2: Cheat Sheet



Data Visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

data · geom (x = F · y = A) · coordinate system · plot

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data · geom (x = F · y = A · color = F · size = A) · coordinate system · plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION> ) +                 Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank() (Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1) x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group)) x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom_area(stat = "bin") x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian") x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot() x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

```
d <- ggplot(mpg, aes(fl))
```

d + geom_bar() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2) x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

```
g <- ggplot(diamonds, aes(cut, color))
```

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom_bin2d(binwidth = c(0.25, 500)) x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d() x, y, alpha, colour, group, linetype, size

h + geom_hex() x, y, alpha, colour, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom_area() x, y, alpha, color, fill, linetype, size

i + geom_line() x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv") x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

j + geom_crossbar(fatten = 2) x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

k + geom_map(aes(map_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

THREE VARIABLES

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))
```

l + geom_contour(aes(z = z)) x, y, z, alpha, colour, group, linetype, size, weight

l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE) x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

RStudio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **http://ggplot2.tidyverse.org** • ggplot2 2.1.0 • Updated: 2016-11

https://www.rstudio.com/resources/cheatsheets/

Data Visualization with ggplot2: Cheat Sheet

Stats An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).

Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function). Use **..name..** syntax to map stat variables to aesthetics.

geom to use | stat function | geommappings

i + stat_density2d(aes(fill = ..level..), geom = "polygon")

variable created by stat

c + stat_bin(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..

c + stat_count(width = 1) **x, y,** | ..count.., ..prop..

c + stat_density(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

e + stat_bin_2d(bins = 30, drop = T)
x, y, fill | ..count.., ..density..

e + stat_bin_hex(bins=30) **x, y, fill** | ..count.., ..density..

e + stat_density_2d(contour = TRUE, n = 100)
x, y, color, size | ..level..

e + stat_ellipse(level = 0.95, segments = 51, type = "t")

l + stat_contour(aes(z = z)) **x, y, z, order** | ..level..

l + stat_summary_hex(aes(z = z), bins = 30, fun = max)
x, y, z, fill | ..value..

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5) **x, y** | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area") **x, y** | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40) **x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") **x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se=T, level=0.95) **x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd=0.5)) **x** | ..x.., ..y..

e + stat_identity(na.rm = TRUE)

ggplot() + stat_qq(aes(sample=1:100), dist = qt, dparam=list(df=5)) **sample, x, y** | ..sample.., ..theoretical..

e + stat_sum() x, y, size | ..n.., ..prop..

e + stat_summary(fun.data = "mean_cl_boot")

h + stat_summary_bin(fun.y = "mean", geom = "bar")

e + stat_unique()

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

(n <- d + geom_bar(aes(fill = fl)))

scale_ | aesthetic to adjust | prepackaged scale to use | scale-specific arguments

n + scale_fill_manual(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

range of values to include in mapping | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

GENERAL PURPOSE SCALES

Use with most aesthetics

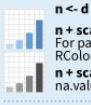
scale_*_continuous() - map cont' values to visual ones
scale_*_discrete() - map discrete values to visual ones
scale_*_identity() - use data values **as** visual ones
scale_*_manual(values = c()) - map discrete values to manually chosen visual ones
scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks") - treat data values as dates.
scale_*_datetime() - treat data x values as date times. Use same arguments as scale_x_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)

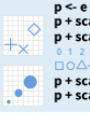


n <- d + geom_bar(aes(fill = fl))
n + scale_fill_brewer(palette = "Blues")
For palette choices:
RColorBrewer::display.brewer.all()
n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")

COLOR AND FILL SCALES (CONTINUOUS)

o <- c + geom_dotplot(aes(fill = ..x..))
o + scale_fill_distiller(palette = "Blues")
o + scale_fill_gradient(low="red", high="yellow")
o + scale_fill_gradient2(low="red", high="blue", mid = "white", midpoint = 25)
o + scale_fill_gradientn(colours=topo.colors(6))
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES



p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7))
p + scale_radius(range = c(1,6))
p + scale_size_area(max_size = 6)

Coordinate Systems

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5))
xlim, ylim
The default cartesian coordinate system

r + coord_fixed(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates

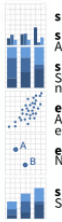
r + coord_polar(theta = "x", direction=1)
theta, start, direction
Polar coordinates

r + coord_trans(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation=c(41, -74, 0))projection, orienztation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.



s <- ggplot(mpg, aes(fl, fill = drv))
s + geom_bar(position = "dodge")
Arrange elements side by side
s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height
e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting
e + geom_label(position = "nudge")
Nudge labels away from points
s + geom_bar(position = "stack")
Stack elements on top of one another

Each position adjustment can be recast as a function with manual **width** and **height** arguments
s + geom_bar(position = position_dodge(width = 1))

Themes

r + theme_bw()
White background with grid lines
r + theme_gray()
Grey background (default theme)
r + theme_dark()
dark for contrast

r + theme_classic()
r + theme_light()
r + theme_linedraw()
r + theme_minimal()
Minimal themes
r + theme_void()
Empty theme

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(. ~ fl)
facet into columns based on fl

t + facet_grid(year ~ .)
facet into rows based on year

t + facet_grid(year ~ fl)
facet into both rows and columns

t + facet_wrap(~ fl)
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(drv ~ fl, scales = "free")
x and y axis limits adjust to individual facets
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(. ~ fl, labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(fl ~ ., labeller = label_bquote(alpha ^ .(fl)))

α^{c}	α^{d}	α^{e}	α^{p}	α^{r}

t + facet_grid(. ~ fl, labeller = label_parsed)

c	d	e	p	r

Labels

t + labs(**x** = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
<AES> = "New <AES> legend title")

Use scale functions to update legend labels

t + annotate(geom = "text", x = 8, y = 9, label = "A")

geom to place | manual values for geom's aesthetics

Legends

n + theme(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"

n + guides(fill = "none")
Set legend type for each aesthetic: colorbar, legend, or none (no legend)

n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E"))
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred)
t + coord_cartesian(
xlim = c(0, 100), ylim = c(10, 20))

With clipping (removes unseen data points)
t + xlim(0, 100) **+ ylim**(10, 20)
t + scale_x_continuous(limits = c(0, 100)) **+ scale_y_continuous**(limits = c(0, 100))

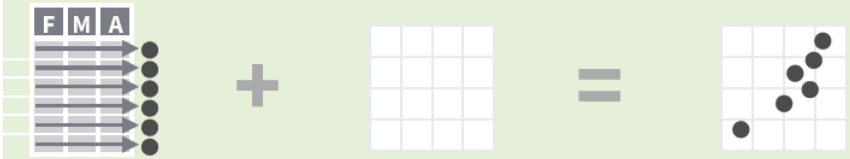
RStudio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **http://ggplot2.tidyverse.org** • ggplot2 2.1.0 • Updated: 2016-11

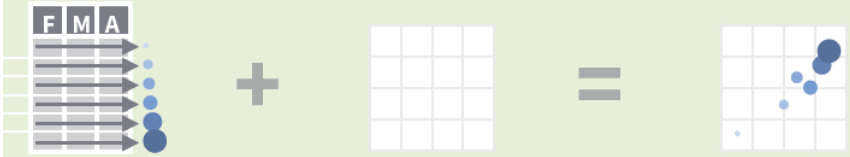
ggplot2: Basics

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

ggplot (data = <DATA>**) +**
<GEOM_FUNCTION>**(mapping = aes(**<MAPPINGS>**),**
stat = <STAT>, position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>

required

Not required, sensible defaults supplied

ggplot(data = mpg, **aes(**x = cty, y = hwy**))** Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings | data | geom

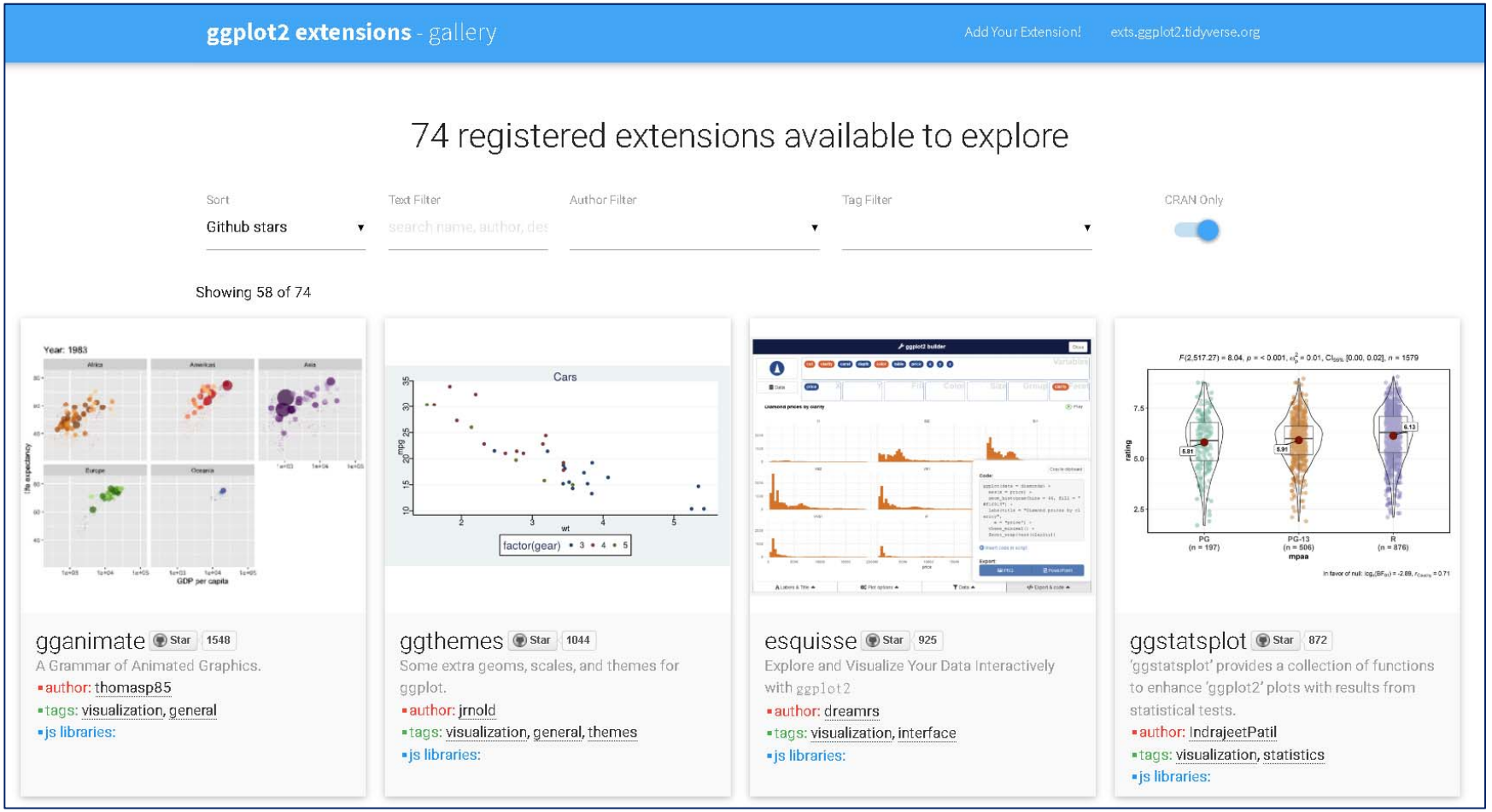
qplot(x = cty, y = hwy, data = mpg, geom = "point"**)** Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

ggplot2 extensions

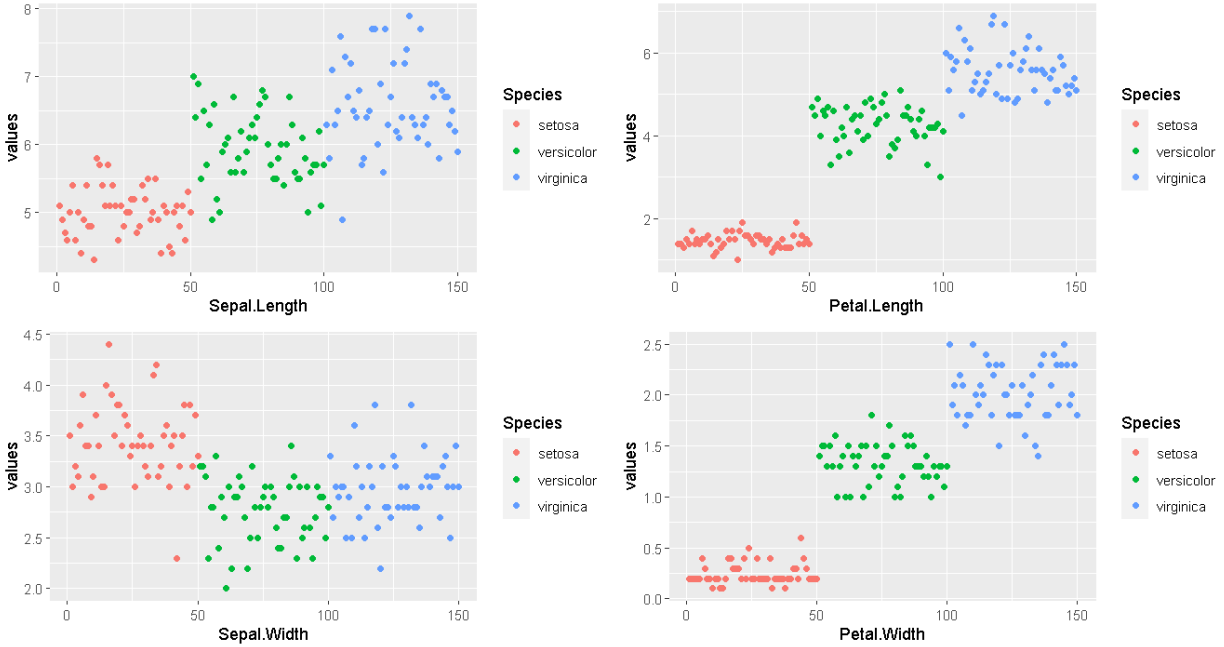
https://exts.ggplot2.tidyverse.org/gallery/



索引圖 (Index plot)

```
ggplot(iris, aes(x=1:150, y=Sepal.Length, color=Species)) + geom_point()
```

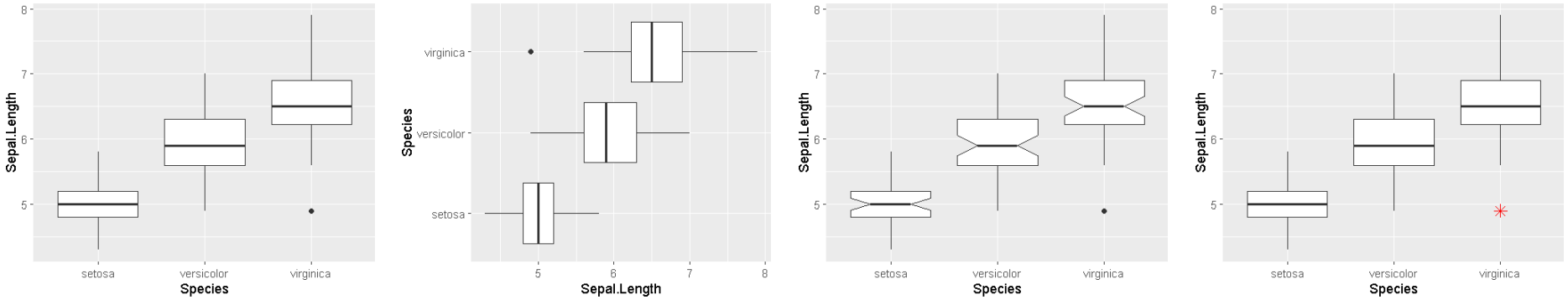
```
iris.index.plot <- function(x){
  ggplot(iris, aes(x=1:nrow(iris), y=iris[,x], color=Species)) +
    geom_point() +
    labs(x=names(iris)[x], y="values")
}
index.list <- lapply(1:4, iris.index.plot)
library(grid)
library(gridExtra)
marrangeGrob(index.list, nrow=2, ncol=2, top="")
```




```
> # select numerical variables
> mydata <- infert

> dim(mydata)
> head(mydata)
> str(mydata)
> id <- sapply(mydata, is.numeric)
> id
> mydata.numeric <- mydata[, id]
> dim(mydata.numeric)
> head(mydata.numeric)

# is.integer, is.double,
# is.logical, is.character,
# is.POSIXt, is.POSIXlt,
# is.POSIXct
```

盒型圖 (Box plots)

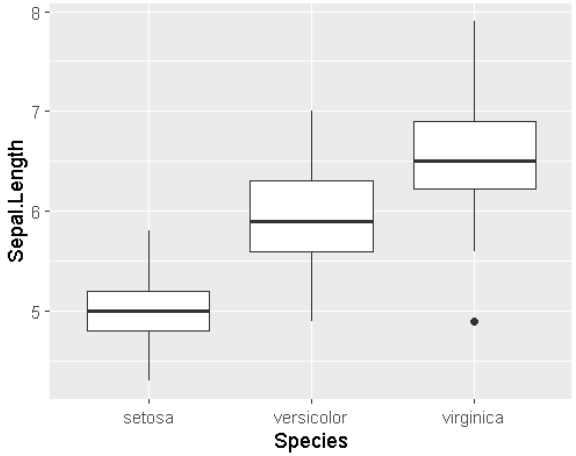
```
library(ggplot2)
p <- ggplot(iris, aes(x=Species, y=Sepal.Length)) +
  geom_boxplot()
p

p + coord_flip() # Rotate the box plot

# Notched box plot
ggplot(iris, aes(x=Species, y=Sepal.Length)) +
  geom_boxplot(notch=TRUE)

ggplot(iris, aes(x=Species, y=Sepal.Length)) +
  geom_boxplot(outlier.colour="red", outlier.shape=8,
               outlier.size=3)
```

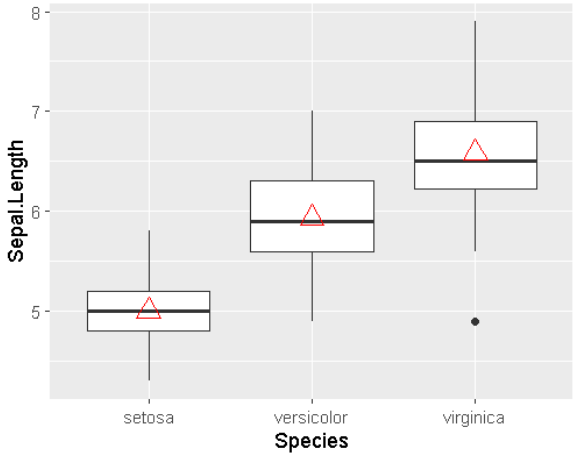


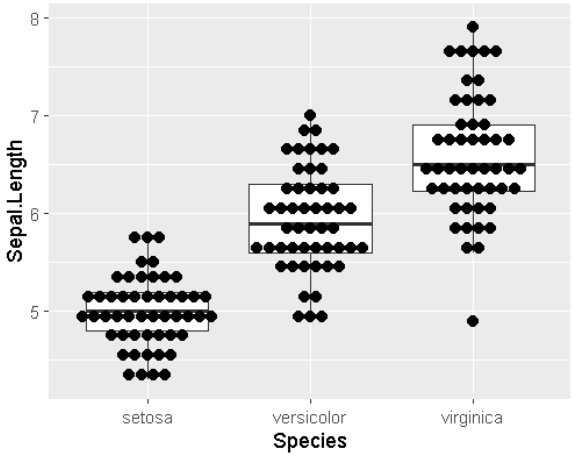
geom_boxplot understands the following aesthetics (required aesthetics are in bold):
x, lower, upper, middle, ymin, ymax, alpha, colour, fill, group, linetype, shape, size, weight

Box plot with dots

```
# stat_summary(): add mean points to a box plot
p <- ggplot(iris, aes(x=Species, y=Sepal.Length)) +
  geom_boxplot()

p

p + stat_summary(fun=mean, geom="point", shape=2, size=4, col="red")

p + geom_dotplot(binaxis="y", stackdir="center")
```

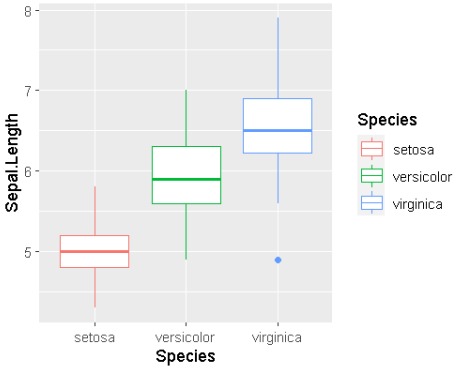


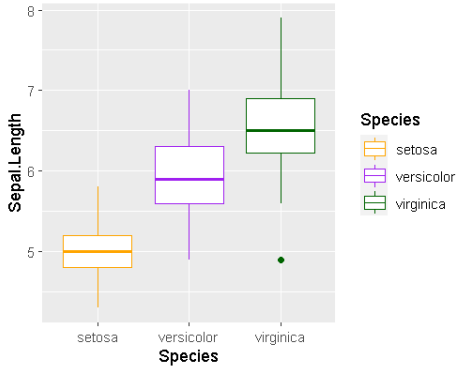


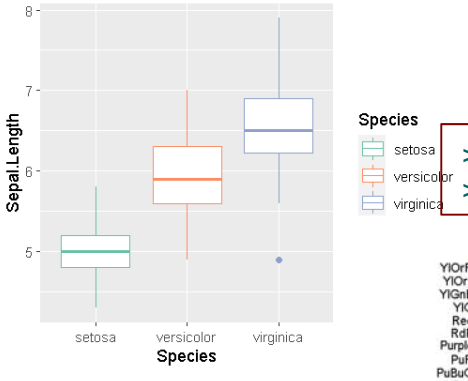


Change box plot line colors

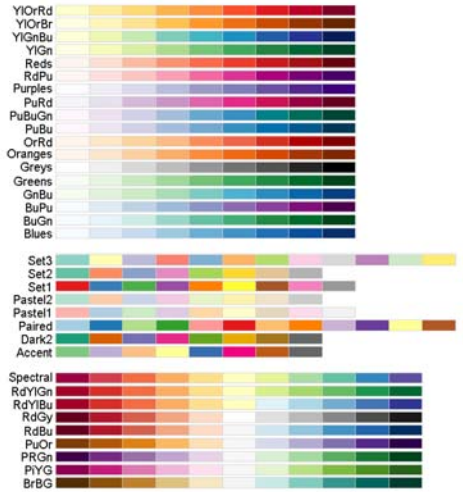
```
p <- ggplot(iris, aes(x=Species, y=Sepal.Length, color=Species)) +
  geom_boxplot()
p

# Use custom color palettes
p + scale_color_manual(values=c("orange", "purple", "darkgreen"))

# Use brewer color palettes
p + scale_color_brewer(palette="Set2")
```







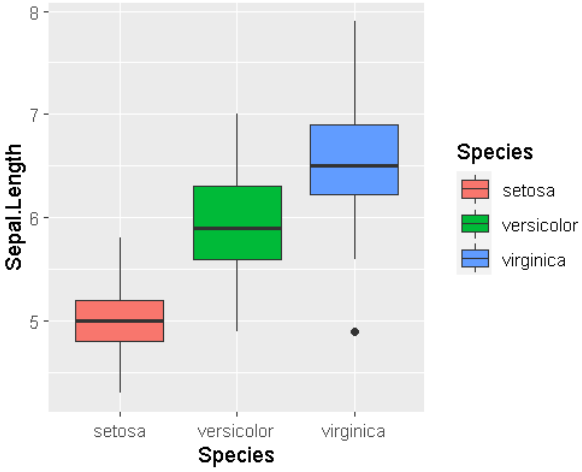

```
> library(RColorBrewer)
> display.brewer.all()
```

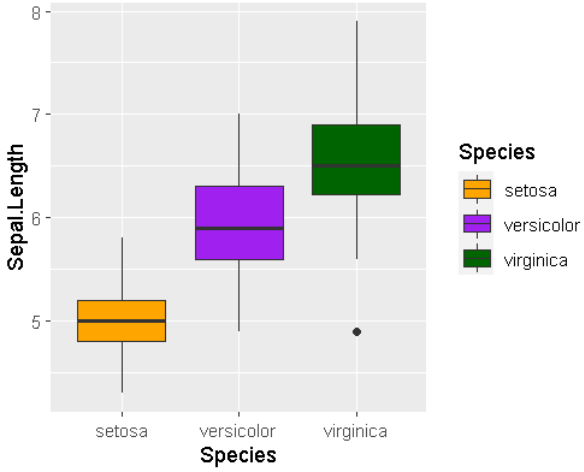


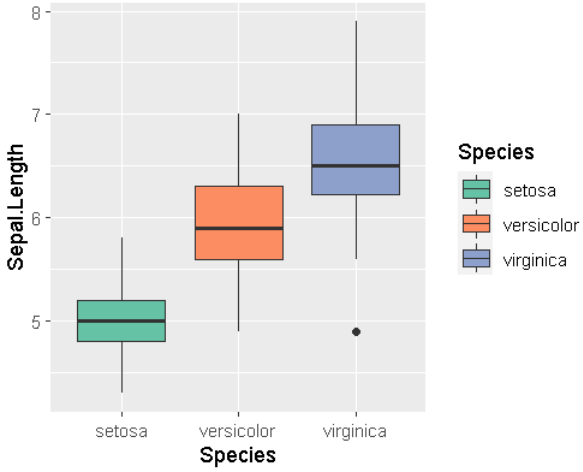

```
# turn off legends
ggplot(iris, aes(x=Species, y=Sepal.Length, color=Species)) +
  geom_boxplot(show.legend = FALSE)

# remove the legend after the plot is created
p + theme(legend.position = "none")
```

Change box plot fill colors

```
p <- ggplot(iris, aes(x=Species, y=Sepal.Length, fill=Species)) +
  geom_boxplot()
p

p + scale_fill_manual(values=c("orange", "purple", "darkgreen"))

p + scale_fill_brewer(palette="Set2")
```



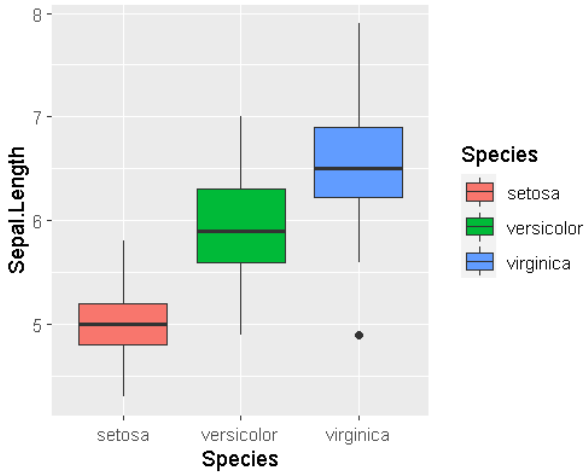


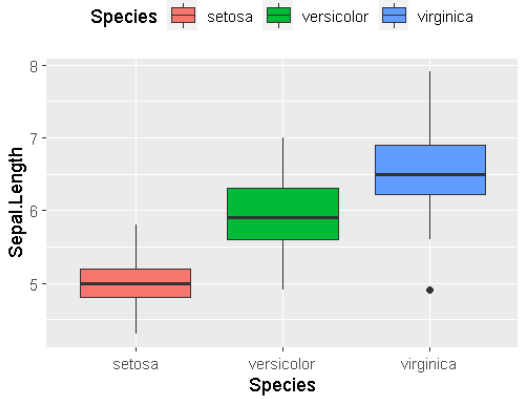


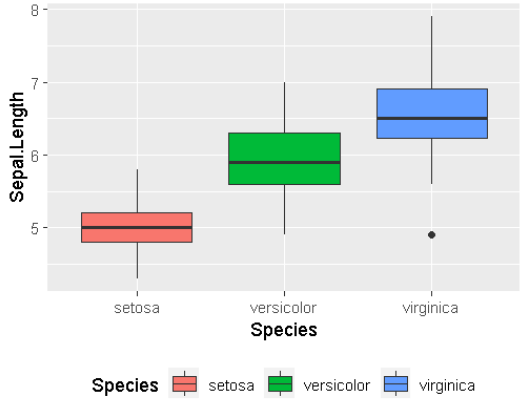
Change the legend position

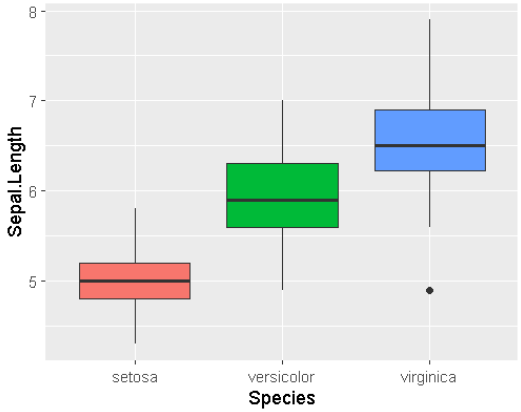
```
p <- ggplot(iris, aes(x=Species, y=Sepal.Length, fill=Species)) +
  geom_boxplot()
p
p + theme(legend.position="top")
p + theme(legend.position="bottom")
p + theme(legend.position="none")
```

legend.position are:
"left", "top", "right", "bottom".





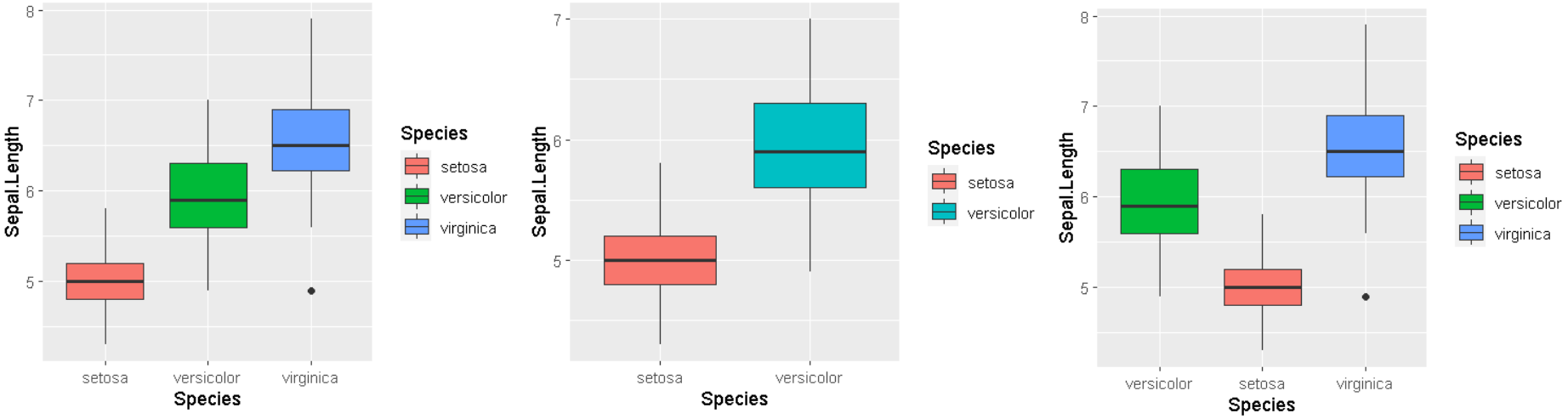




Change the order of items in the legend

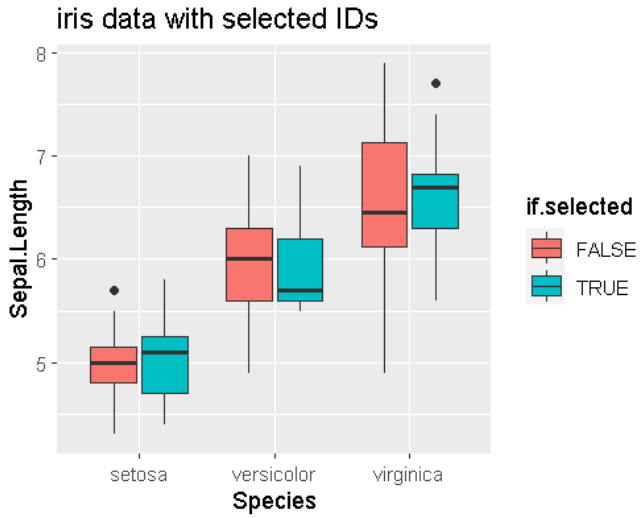
```
> levels(iris$Species)
[1] "setosa"     "versicolor" "virginica"
```

```
levels(iris$Species)

p <- ggplot(iris, aes(x=Species, y=Sepal.Length, fill=Species)) +
  geom_boxplot()
p
p + scale_x_discrete(limits=c("setosa", "versicolor"))
p + scale_x_discrete(limits=c("versicolor", "setosa", "virginica"))
```



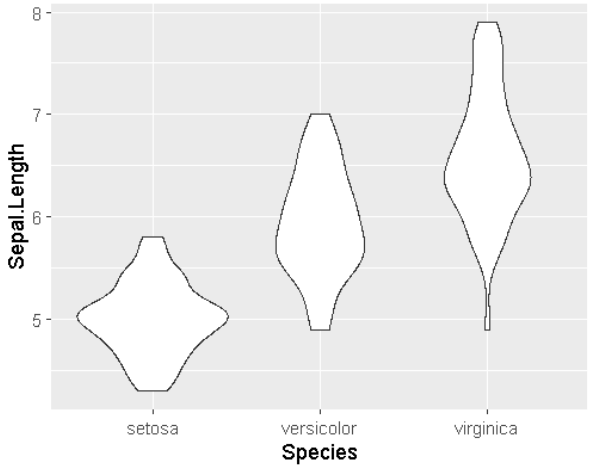
Box plot with multiple groups

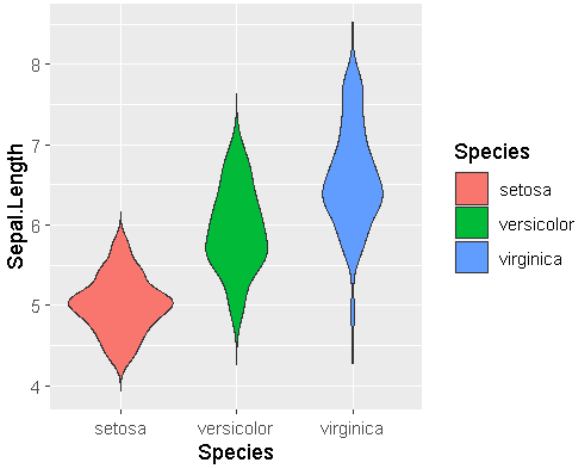
```
> select.id <- sample(1:150, 50)
> if.selected <- 1:150 %in% select.id
> iris.sel <- cbind(iris, if.selected)
> head(iris.sel)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species if.selected
1          5.1         3.5          1.4         0.2  setosa        TRUE
2          4.9         3.0          1.4         0.2  setosa       FALSE
3          4.7         3.2          1.3         0.2  setosa       FALSE
4          4.6         3.1          1.5         0.2  setosa       FALSE
5          5.0         3.6          1.4         0.2  setosa        TRUE
6          5.4         3.9          1.7         0.4  setosa        TRUE
>
> p <- ggplot(iris, aes(x=Species, y=Sepal.Length, fill=if.selected)) +
+   geom_boxplot() +
+   labs(title="iris data with selected IDs")
> p
```

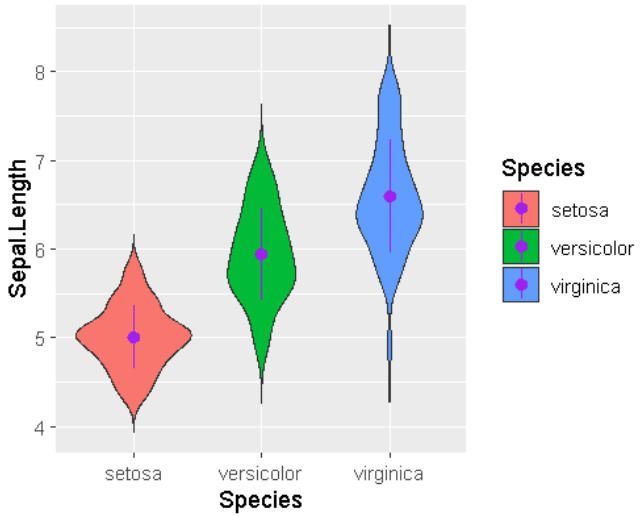


小提琴圖 (Violin plots)

```
ggplot(iris, aes(x=Species, y=Sepal.Length)) +
  geom_violin()

p <- ggplot(iris, aes(x=Species, y=Sepal.Length, fill=Species)) +
  geom_violin(trim=FALSE)
p

p + stat_summary(fun.data=mean_sdl, fun.args = list(mult = 1),
                 geom="pointrange", color="purple")
```

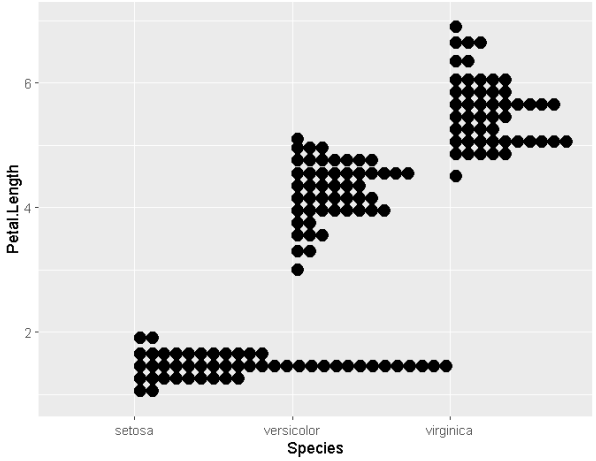


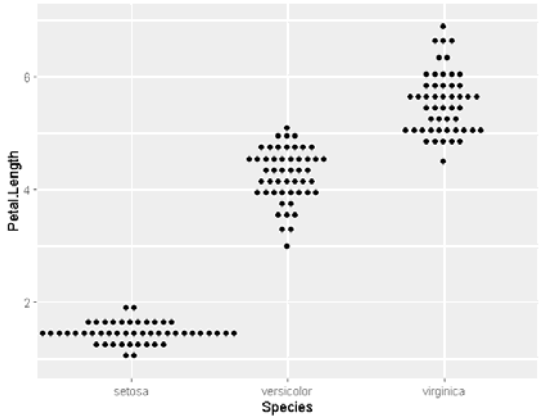


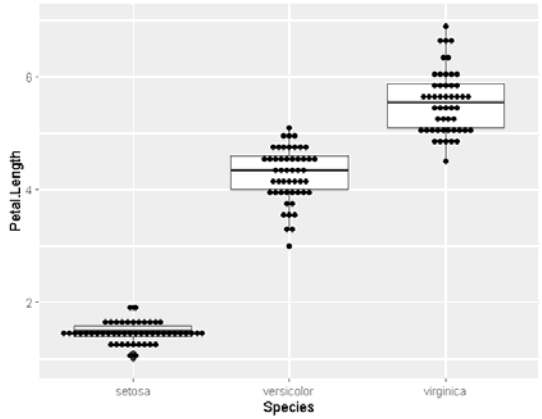


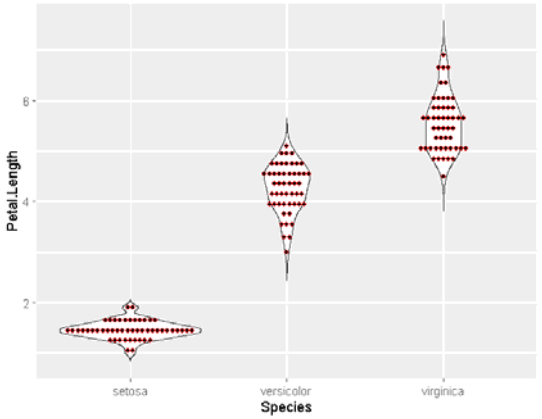
點圖 (Dot plots)

```
ggplot(iris, aes(x=Species, y=Petal.Length)) +
  geom_dotplot(binaxis="y")

ggplot(iris, aes(x=Species, y=Petal.Length)) +
  geom_dotplot(binaxis="y", stackdir="center",
               stackratio=1.5, dotsize=0.5)

ggplot(iris, aes(x=Species, y=Petal.Length)) +
  geom_boxplot()+
  geom_dotplot(binaxis="y", stackdir="center", dotsize=0.5)

ggplot(iris, aes(x=Species, y=Petal.Length)) +
  geom_violin(trim = FALSE)+
  geom_dotplot(binaxis="y", stackdir="center", dotsize=0.5, colour="red")
```



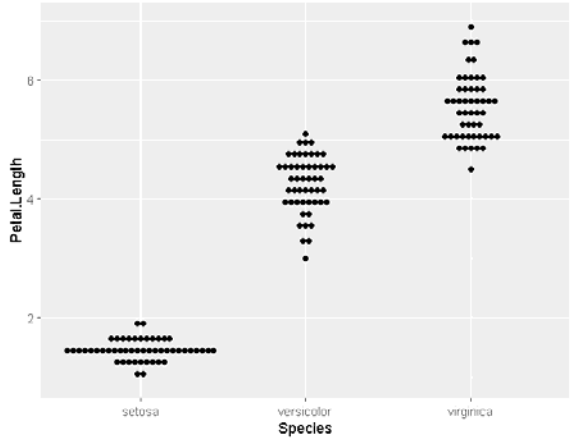


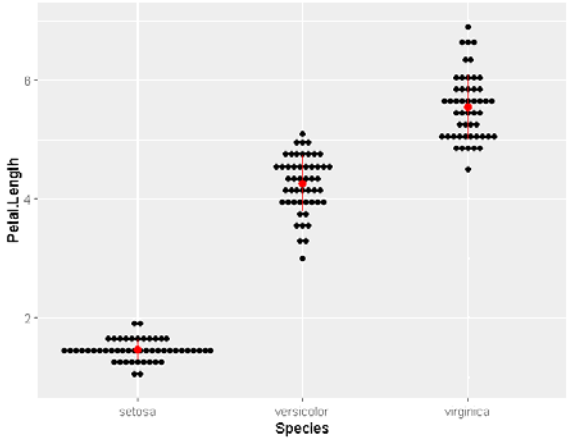


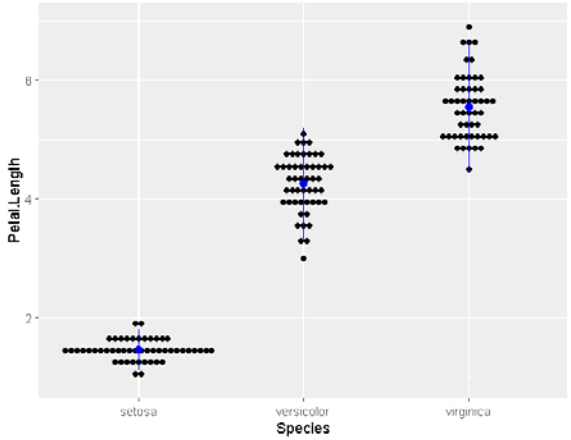


Dot plots + 統計量

```
p <- ggplot(iris, aes(x=Species, y=Petal.Length)) +
       geom_dotplot(binaxis="y", stackdir="center", dotsize=0.5)
p
p + stat_summary(fun=mean, geom="point", shape=17,
                 size=3, color="red")

my_summary <- function(x, a=1) {
   m <- mean(x)
   ymin <- m - a * sd(x)
   ymax <- m + a * sd(x)
   c(y=m, ymin=ymin, ymax=ymax)
}
p + stat_summary(fun.data=my_summary, color="red")

p + stat_summary(fun.data=my_summary, color="blue", fun.args=list(a=2))
```

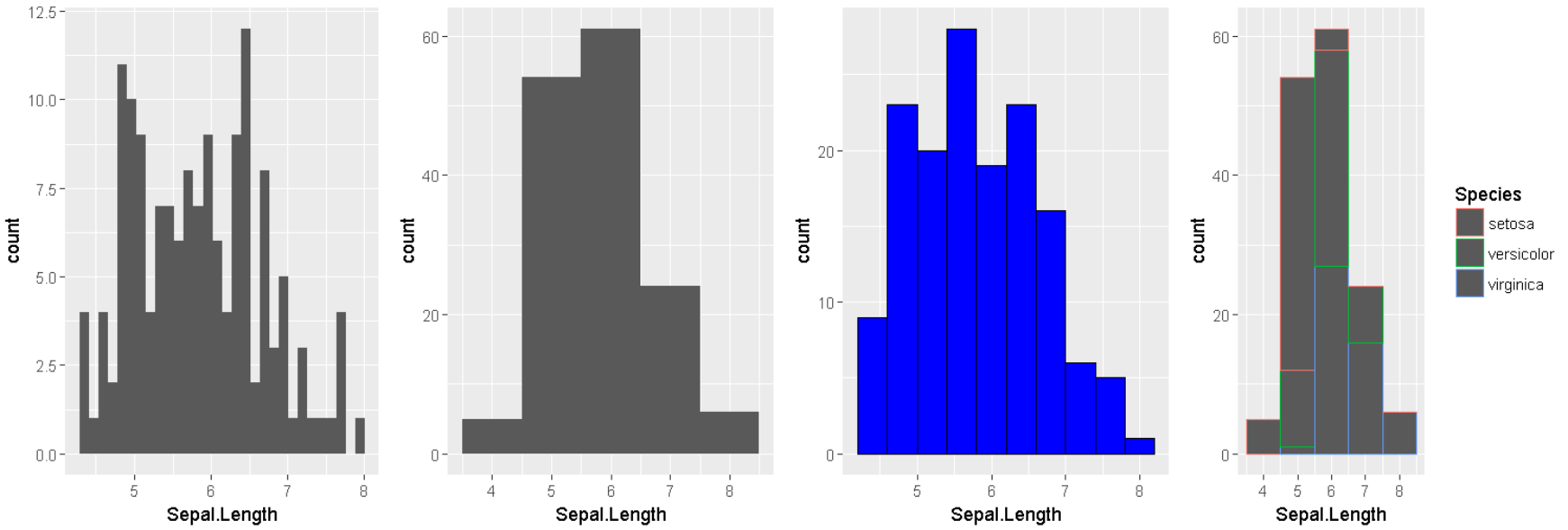






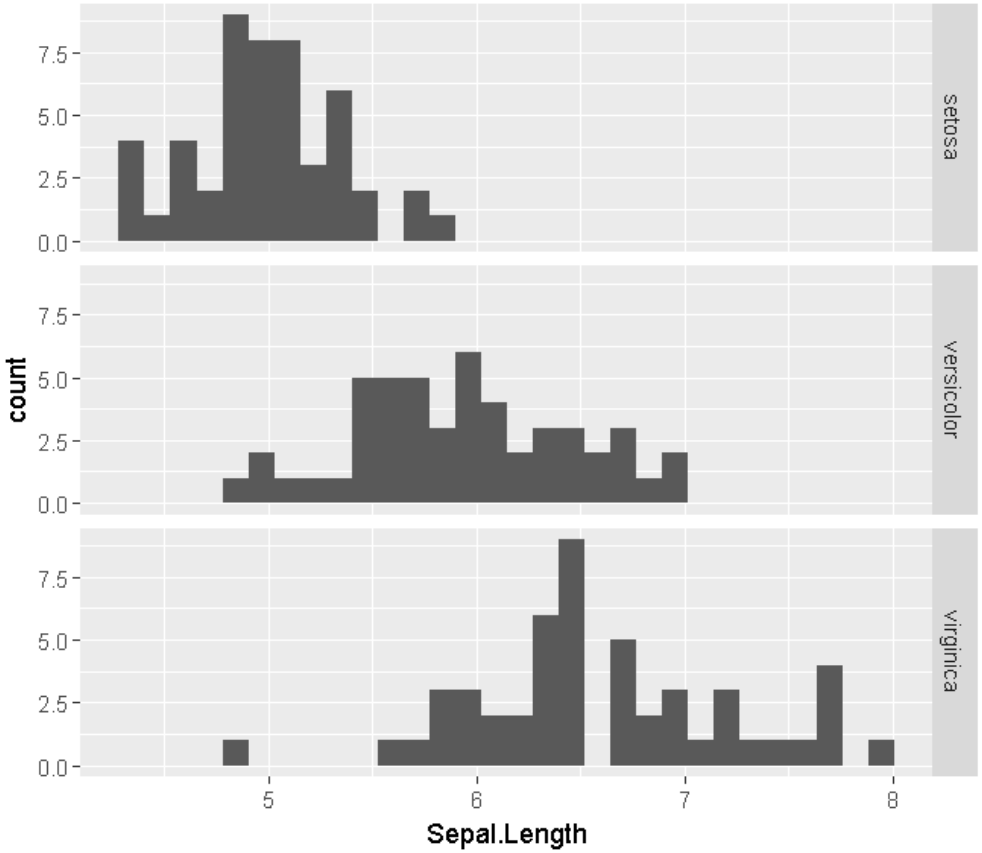
直方圖 (Histogram)

```
> # install.packages("gridExtra")
> library(gridExtra)
> h1 <- ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram()
> h2 <- ggplot(data=iris, aes(x=Sepal.Length)) + geom_histogram(binwidth=1)
> h3 <- ggplot(data=iris, aes(x=Sepal.Length)) +
        geom_histogram(color="black", fill="blue", bins = 10)
> h4 <- ggplot(data=iris, aes(x=Sepal.Length, color=Species)) + geom_histogram(binwidth = 1)
> grid.arrange(h1, h2, h3, h4, nrow=1, ncol=4)
```



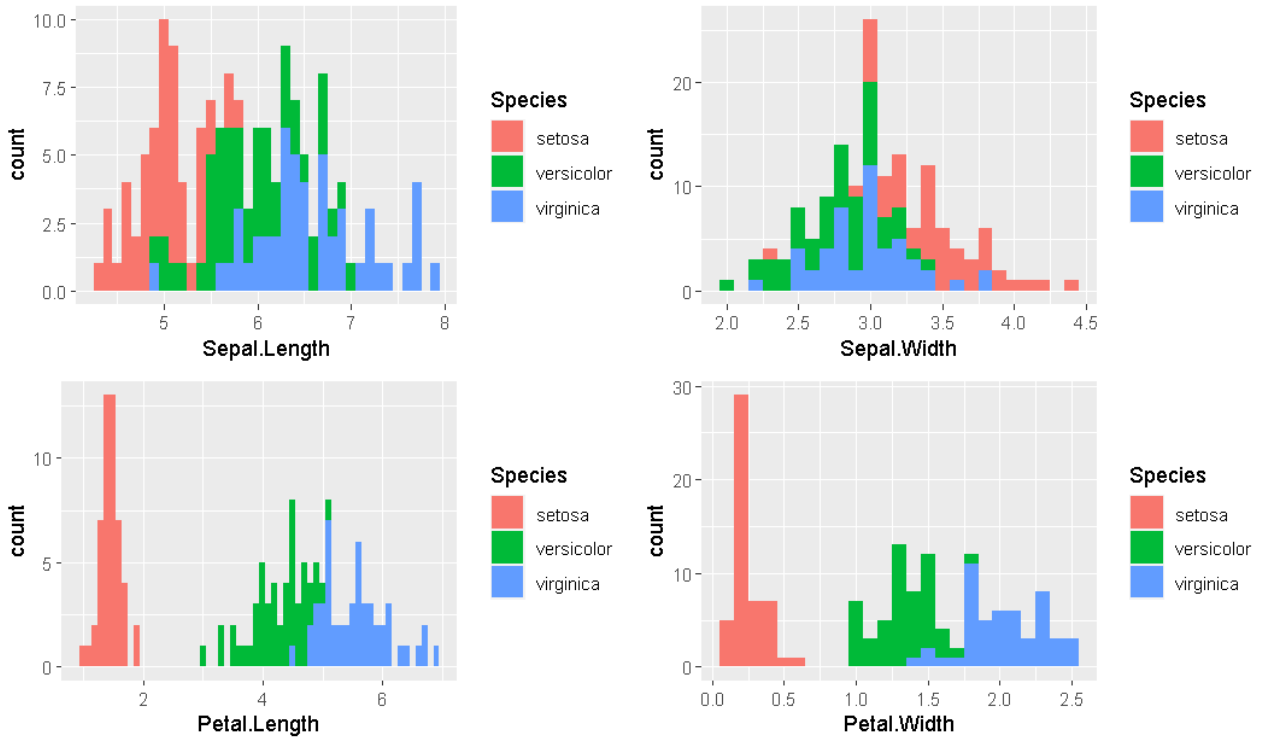
Histogram

```
> p <- ggplot(data=iris, aes(x=Sepal.Length))
> p <- p + geom_histogram()
> p + facet_grid(Species~.)   #row
```



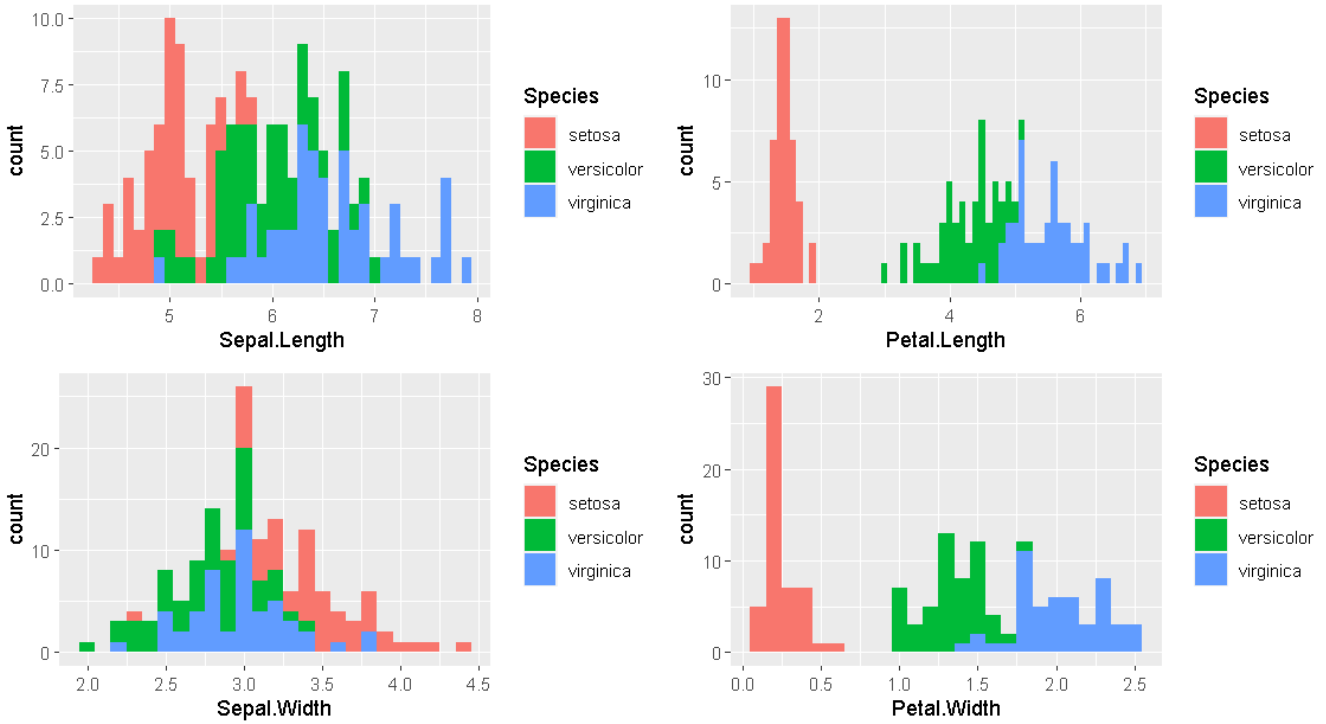
Histogram

```
> library(gridExtra)
> sl <- ggplot(iris, aes(x=Sepal.Length, fill=Species)) +
    geom_histogram(binwidth = 0.1)
> sw <- ggplot(iris, aes(x=Sepal.Width, fill=Species)) +
    geom_histogram(binwidth = 0.1)
> pl <- ggplot(iris, aes(x=Petal.Length, fill=Species)) +
    geom_histogram(binwidth = 0.1)
> pw <- ggplot(iris, aes(x=Petal.Width, fill=Species)) +
    geom_histogram(binwidth = 0.1)
> grid.arrange(sl, sw, pl, pw, nrow = 2)
```



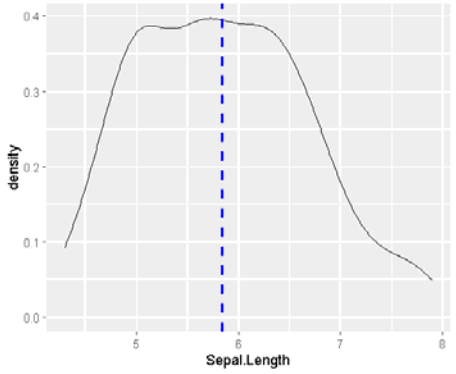
Histogram

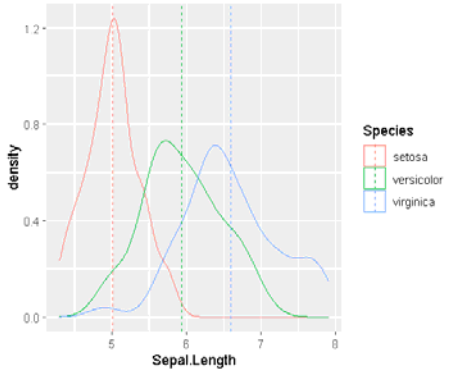
```
iris.hist <- function(x){
  ggplot(iris, aes(x=iris[,x], fill=Species)) +
    geom_histogram(binwidth = 0.1) +
    xlab(names(iris)[x])
}

hist.list <- lapply(1:4, iris.hist)
library(grid)
marrangeGrob(hist.list, nrow=2, ncol=2, top="")
```

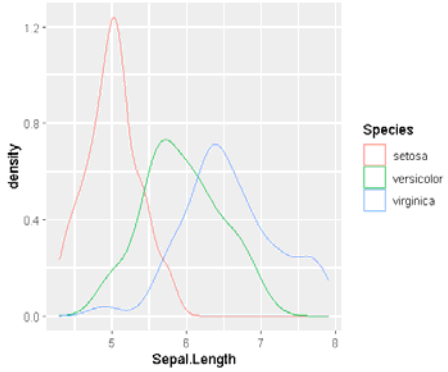


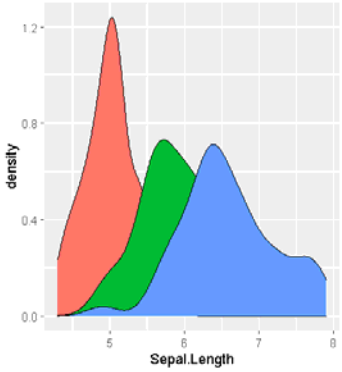
機率密度圖 (Density plots)

```
p1 <- ggplot(iris, aes(x=Sepal.Length)) +
        geom_density()

p2 <- ggplot(iris, aes(x=Sepal.Length, color=Species)) +
        geom_density()

p3 <- ggplot(iris, aes(x=Sepal.Length, fill=Species)) +
        geom_density()

# Add mean line
p1 + geom_vline(aes(xintercept=mean(Sepal.Length)),
                color="blue", linetype="dashed", size=1)

mu <- tapply(iris$Sepal.Length, iris$Species, mean)
mu.df <- data.frame(Sp=names(mu), grp.mean=mu)
head(mu.df)
p2 + geom_vline(data=mu.df, aes(xintercept=grp.mean, color=Sp),
             linetype="dashed")
```



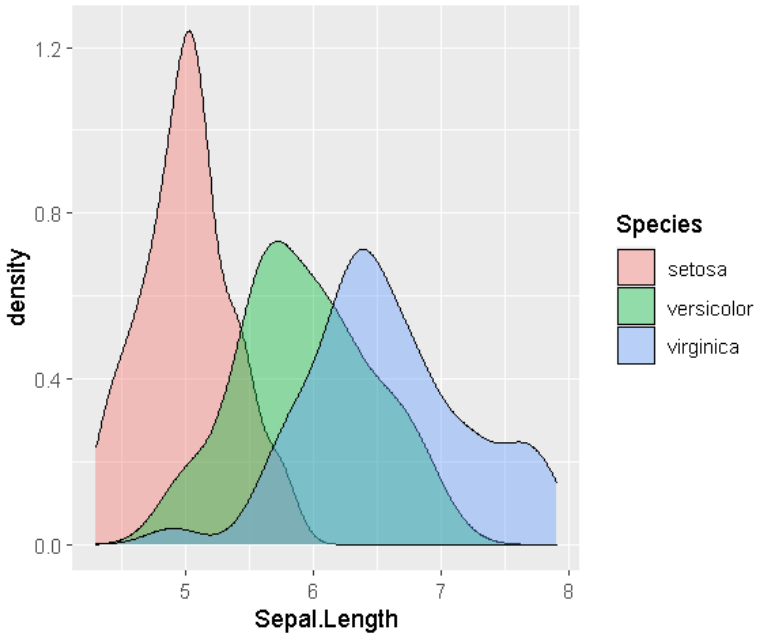






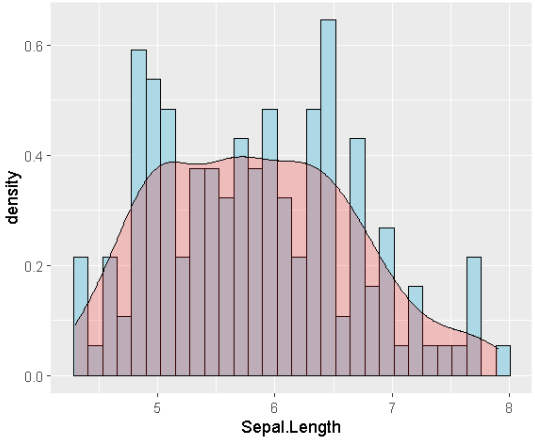
Change line color, line type and fill color

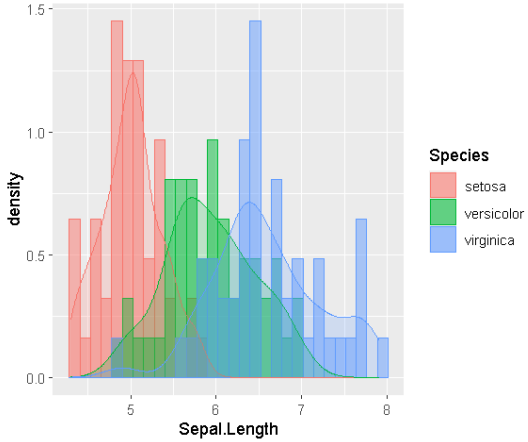
```
# Change line color, line type and fill color
ggplot(iris, aes(x=Sepal.Length))+
  geom_density(color="darkblue", fill="lightblue", linetype="dashed")

# Use semi-transparent fill
ggplot(iris, aes(x=Sepal.Length, fill=Species)) +
  geom_density(alpha=0.4)
```

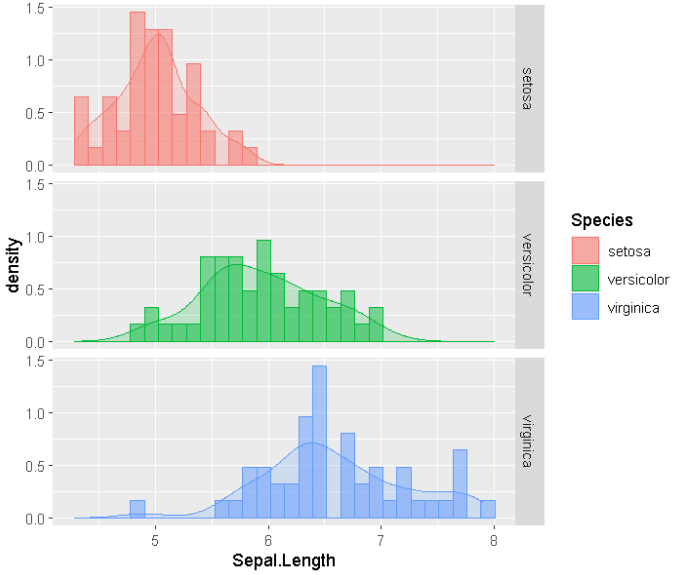


Combine histogram and density plots

```
# Histogram with density plot
ggplot(iris, aes(x=Sepal.Length)) +
 geom_histogram(aes(y=..density..), colour="black", fill="lightblue") +
 geom_density(alpha=0.2, fill="red")

# Color by groups
phd <- ggplot(iris, aes(x=Sepal.Length, color=Species, fill=Species)) +
         geom_histogram(aes(y=..density..), alpha=0.5, position="identity") +
         geom_density(alpha=.2)
phd

# facets: Split the plot in multiple panels
phd + facet_grid(Species ~ .)
```







Create the histogram with a density scale using the computed varlable `..density..`

mtcars {datasets}

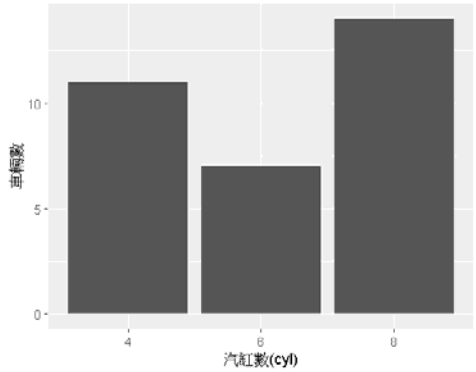
mtcars {datasets}: Motor Trend Car Road Tests ((1974 Motor Trend US) data frame包含32台車與11個車的屬性

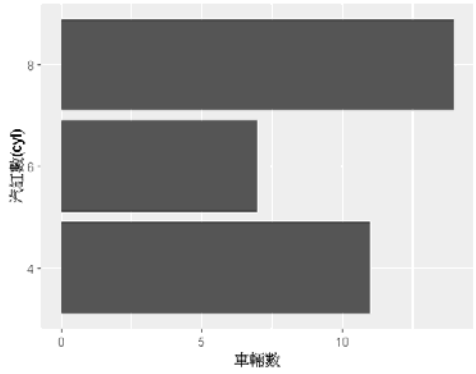
- mpg：Miles/(US) gallon 耗油量
- cyl：Number of cylinders 汽缸數
- disp：Displacement (cu.in., cubic inch) 單汽缸排氣量
- hp：Gross horsepower 馬力
- drat：Rear axle ratio 後輪軸減速比
- wt：Weight (1000 lbs) 車體重量
- qsec：1/4 mile time 1/4英里加速秒數
- vs：V/S，0代表V型引擎，1代表直立式引擎
- am：Transmission (0 = automatic自排, 1 = manual手排)
- gear：Number of forward gears 變速箱數
- carb：Number of carburetors 化油器數

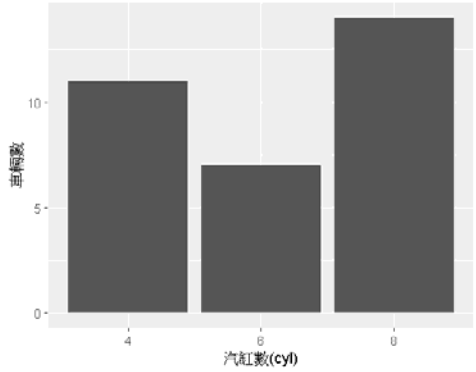
```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 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
```

長條圖 (Bar plot)(Bar chart)

```
> # mtcars$cyl <- as.factor(mtcars$cyl)
> p <- ggplot(mtcars, aes(x=cyl)) +
+       geom_bar() +
+       labs(x="汽缸數(cyl)", y="車輛數")
>
> p + coord_flip()
>
> cyl.df <- data.frame(table(mtcars$cyl))
> cyl.df
  Var1 Freq
1    4   11
2    6    7
3    8   14
> names(cyl.df) <- c("cyl", "count")
> cyl.df
  cyl count
1   4    11
2   6     7
3   8    14
>
> ggplot(cyl.df, aes(x=cyl, y=count)) +
+   geom_bar(stat="identity") +
+   labs(x="汽缸數(cyl)", y="車輛數")
```



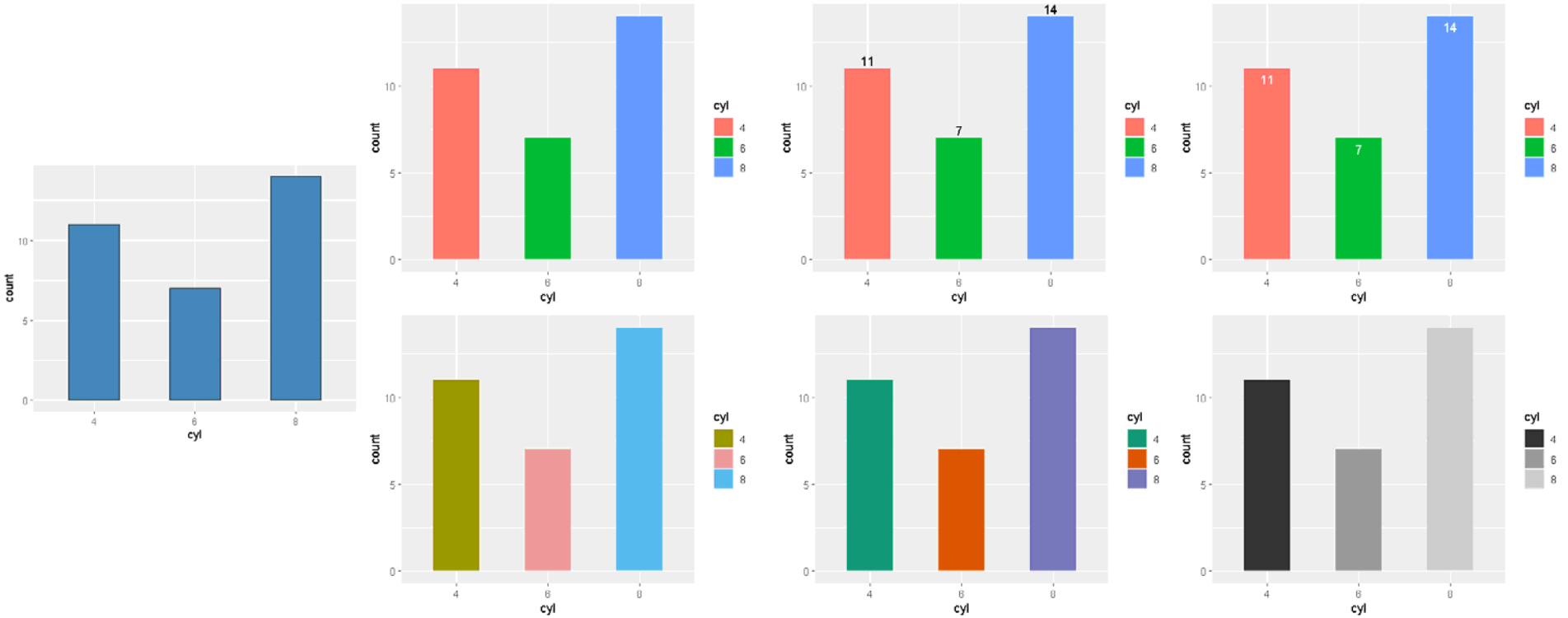




`ggtitle("Main title")`: Adds a main title above the plot
`xlab("X axis label")`: Changes the X axis label
`ylab("Y axis label")`: Changes the Y axis label
`labs(title="Main title", x="X axis label", y="Y axis label")`: Changes main title and axis labels

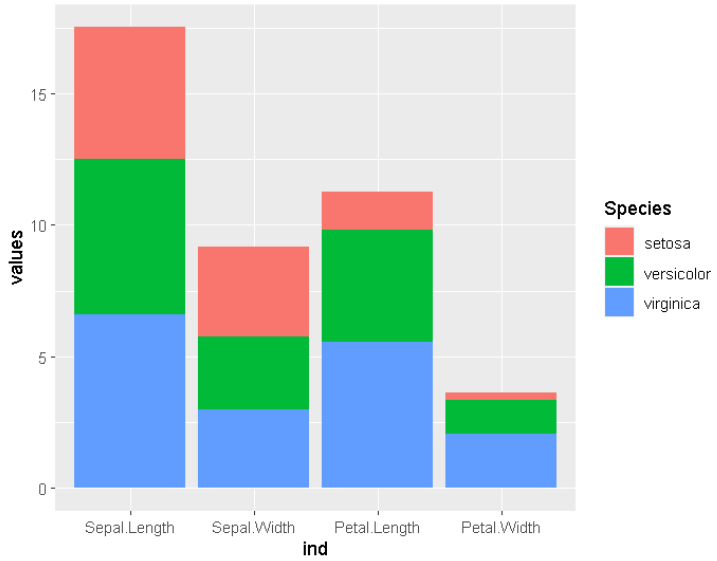
Bar plot

```
ggplot(cyl.df, aes(x=cyl, y=count)) +
       geom_bar(stat="identity", width=0.5, color="black", fill="steelblue")
p <- ggplot(cyl.df, aes(x=cyl, y=count, fill=cyl)) +
       geom_bar(stat="identity", width=0.5)
p
p + geom_text(aes(label=count), vjust=-0.3, size=4)
p + geom_text(aes(label=count), vjust=1.6, color="white", size=4)
p + scale_fill_manual(values=c("#999000", "#E69F99", "#56B4E9"))
p + scale_fill_brewer(palette="Dark2")
p + scale_fill_grey()
```



Bar plot

```
> iris.mean <- aggregate(iris[,1:4], by=list(Species=iris$Species), FUN=mean)
> iris.mean
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        5.006       3.428        1.462       0.246
2 versicolor        5.936       2.770        4.260       1.326
3  virginica        6.588       2.974        5.552       2.026
> mydata <- cbind(stack(iris.mean[,-1]), Species = iris.mean$Species)
> mydata
   values          ind    Species
1   5.006 Sepal.Length     setosa
2   5.936 Sepal.Length versicolor
3   6.588 Sepal.Length  virginica
4   3.428  Sepal.Width     setosa
5   2.770  Sepal.Width versicolor
6   2.974  Sepal.Width  virginica
7   1.462 Petal.Length     setosa
8   4.260 Petal.Length versicolor
9   5.552 Petal.Length  virginica
10  0.246  Petal.Width     setosa
11  1.326  Petal.Width versicolor
12  2.026  Petal.Width  virginica
> ggplot(mydata, aes(x=ind, y=values, fill = Species)) +
+   geom_bar(stat="identity")
```

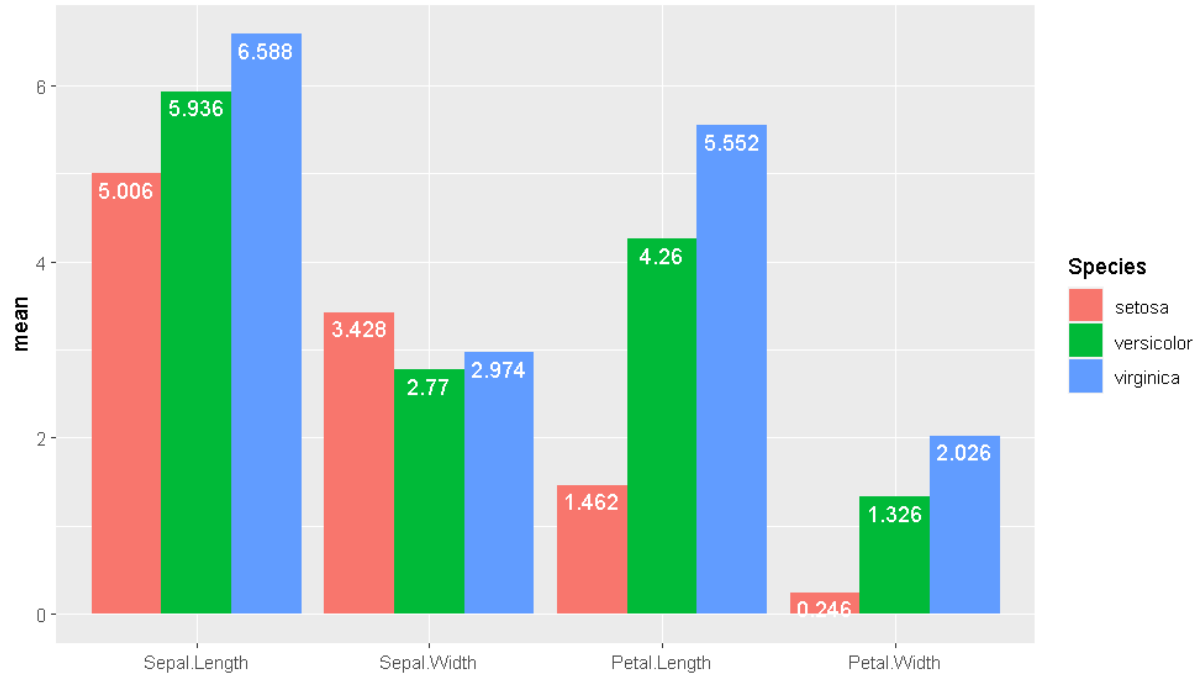


`position_dodge` for creating side-by-side barcharts.

Other position adjustments: **`position_identity`**, **`position_jitterdodge`**, **`position_jitter`**, **`position_nudge`**, **`position_stack`**

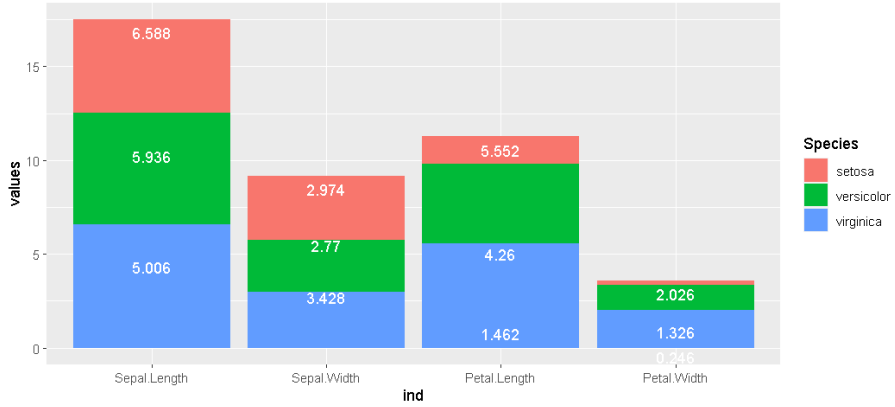
Bar plot

```
ggplot(mydata, aes(x=ind, y=values, fill=Species)) +
  geom_bar(stat="identity", position="dodge") +
  geom_text(aes(label=values), vjust=1.4, color="white",
            position = position_dodge(0.9)) +
  labs(x="", y="mean")
```

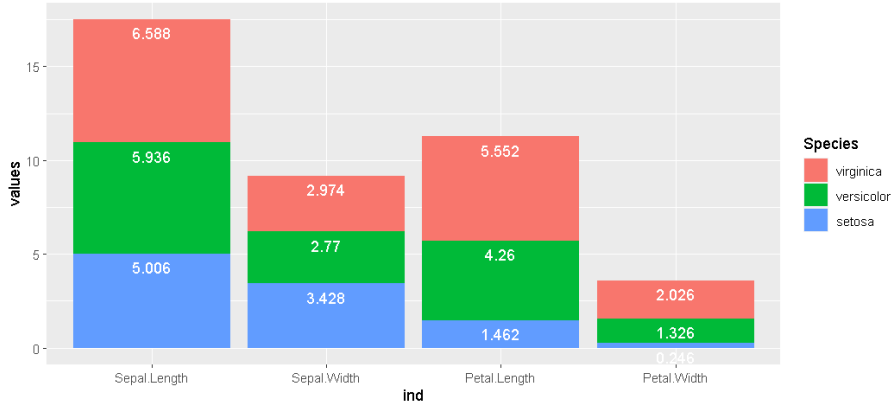


Bar plot

```
> library(plyr)
> mydata.sorted <- arrange(mydata, ind, Species)
> mydata.sorted
   values          ind    Species
1   5.006 Sepal.Length     setosa
2   5.936 Sepal.Length versicolor
3   6.588 Sepal.Length  virginica
...
10  0.246  Petal.Width     setosa
11  1.326  Petal.Width versicolor
12  2.026  Petal.Width  virginica
```

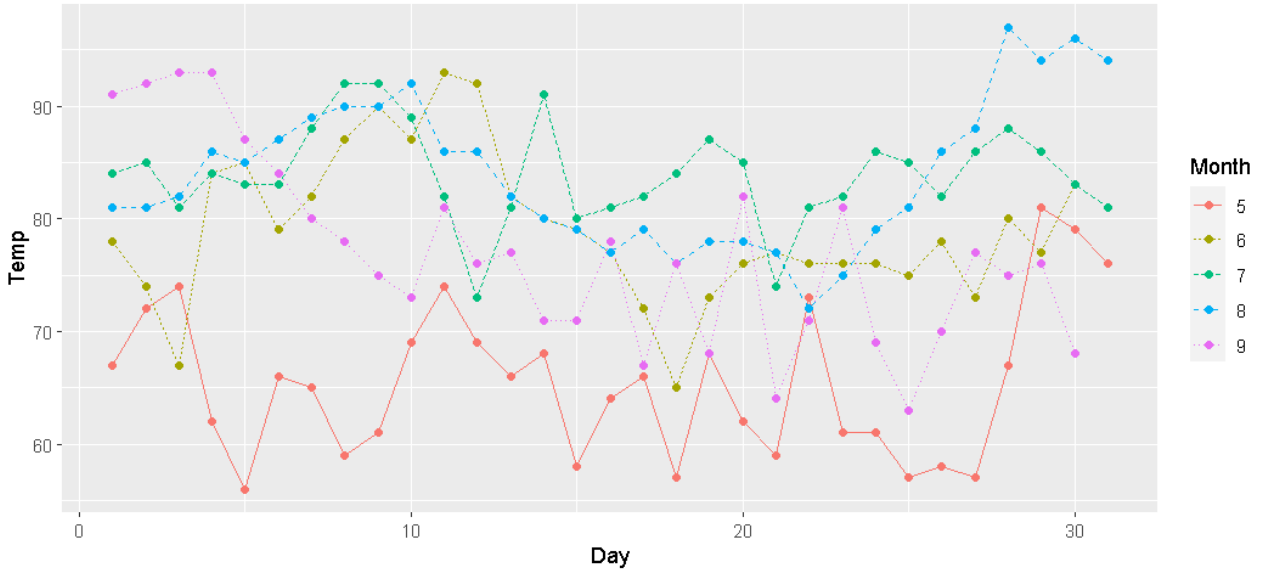



```
> mydata.sorted$Species <- factor(mydata.sorted$Species, levels=rev(levels(mydata$Species)))
> mydata.cumsum <- ddply(mydata.sorted, "ind",
+                        transform, label.ypos=cumsum(values))
> mydata.cumsum
   values          ind    Species label.ypos
1   5.006 Sepal.Length     setosa      5.006
2   5.936 Sepal.Length versicolor     10.942
3   6.588 Sepal.Length  virginica     17.530
...
10  0.246  Petal.Width     setosa      0.246
11  1.326  Petal.Width versicolor      1.572
12  2.026  Petal.Width  virginica      3.598
>
>
>
> ggplot(mydata.cumsum, aes(x=ind, y=values, fill=Species)) +
+   geom_bar(stat="identity") +
+   geom_text(aes(y=label.ypos, label=values), vjust=1.6, color="white")
```



線圖 (Line plot)

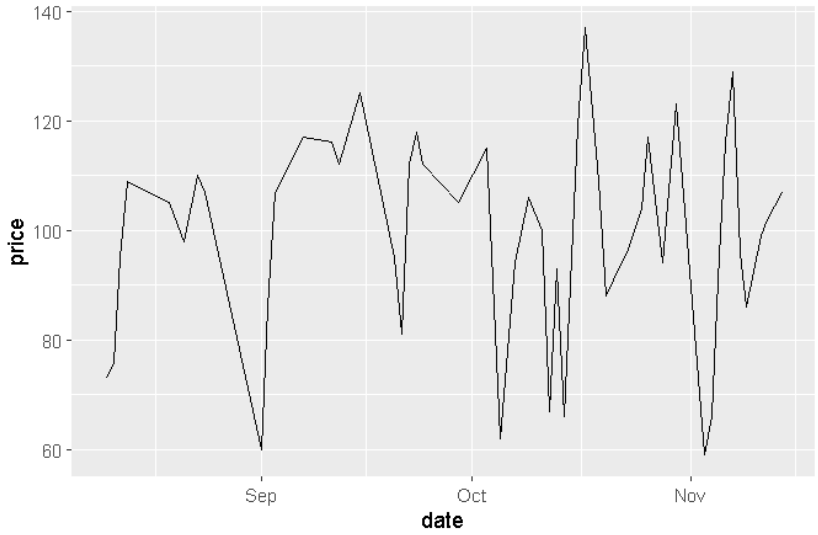
```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> airquality$Month <- factor(airquality$Month)
> ggplot(airquality, aes(x=Day, y=Temp, group=Month, color=Month)) +
+   geom_line(aes(linetype=Month)) +
+   geom_point()
```

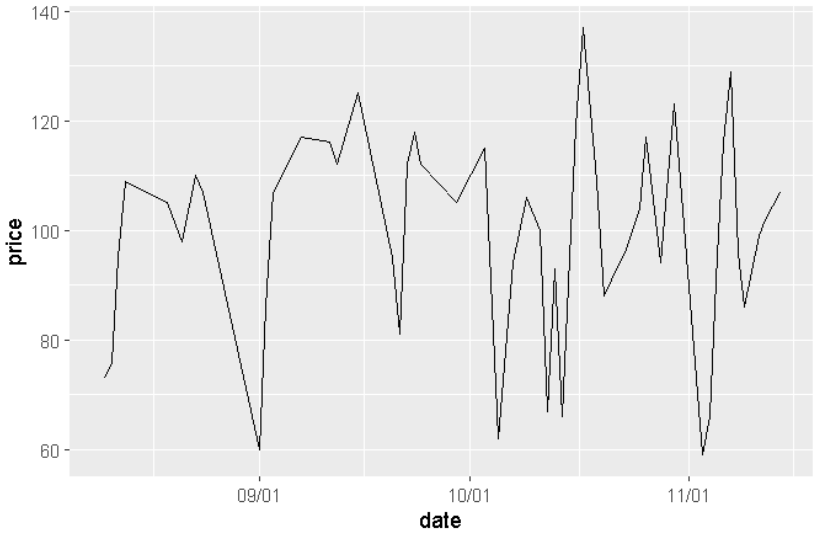


Line plot

```
> sales <- data.frame(
+   date = seq(Sys.Date(), length.out=100, by="1 day")[sample(100, 50)],
+   price = floor(rnorm(50, mean=100, sd=20))
+ )
> sales <- sales[order(sales$date), ]
> head(sales)
         date price
28 2020-08-10    73
27 2020-08-11    76
1  2020-08-12    95
22 2020-08-13   109
8  2020-08-16   107
18 2020-08-19   105
```

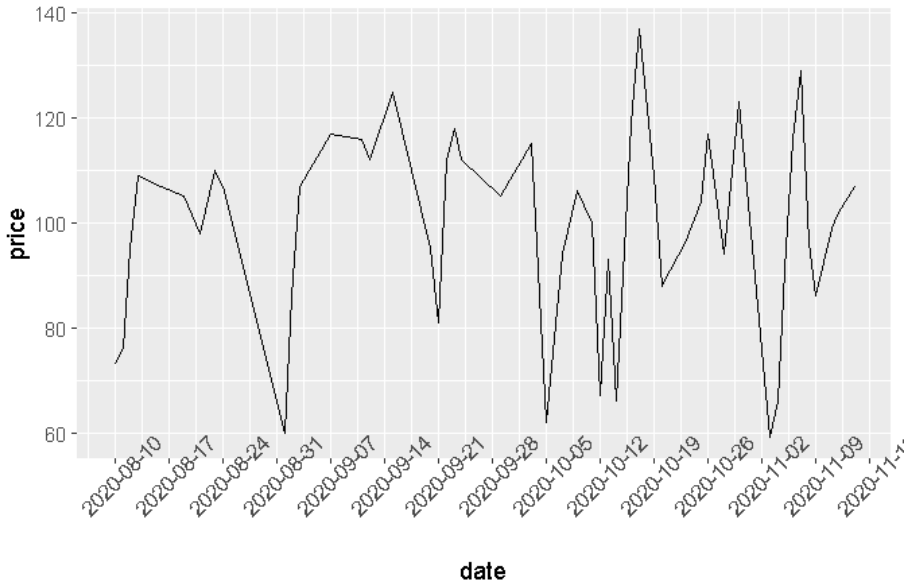
```
lp <- ggplot(data=sales, aes(x=date, y=price)) + geom_line()
lp

lp + scale_x_date(date_labels=("%m/%d"))
```

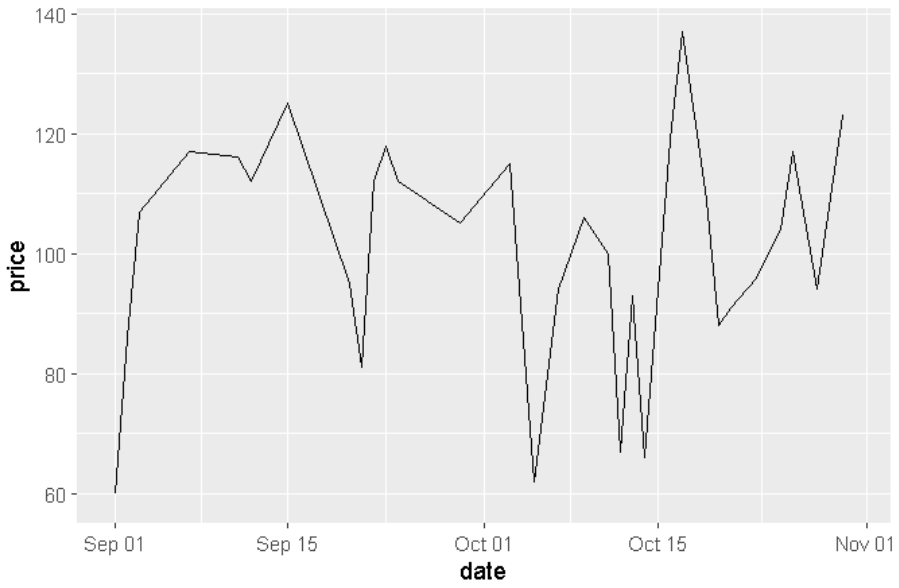




Line plot

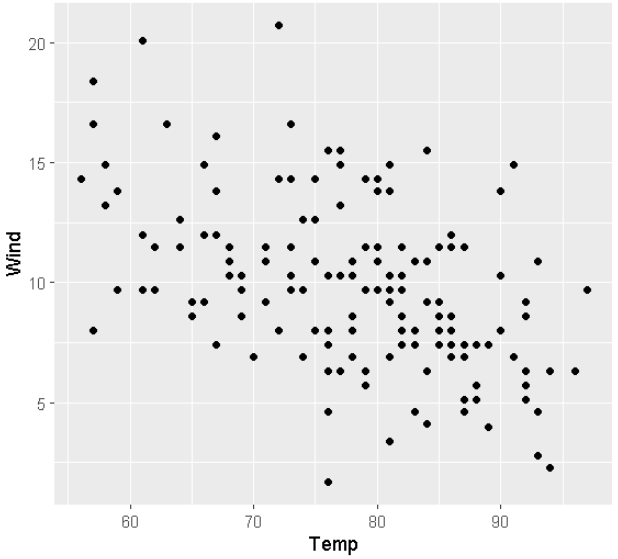
```
lp + scale_x_date(breaks = date_breaks("1 week")) +
     theme(axis.text.x = element_text(angle=45))

range(sales$date)
# "2020-08-10" "2020-11-07"
amin <- as.Date("2020-09-01")
amax <- as.Date("2020-10-31")
lp + scale_x_date(limits = c(amin, amax))
```

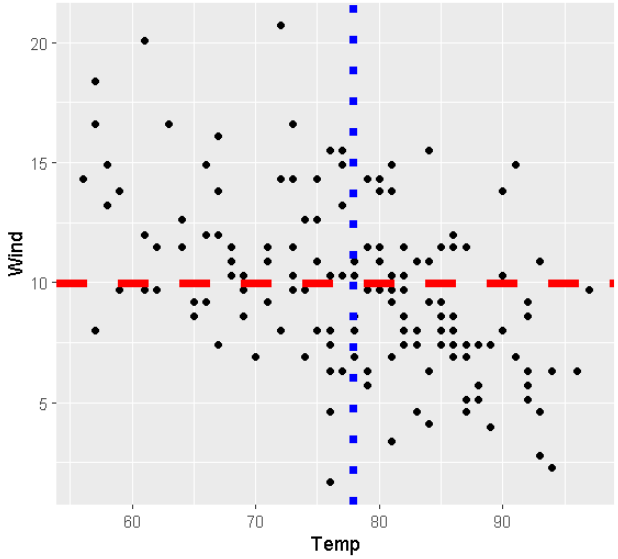


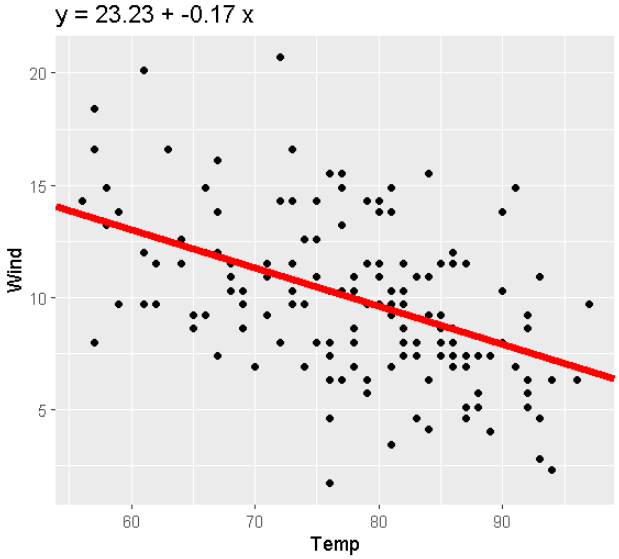


於圖上加直線、迴歸線

```
pp <- ggplot(airquality, aes(x=Temp, y=Wind)) +
        geom_point()
pp
pp + geom_hline(yintercept=mean(airquality$Wind), linetype="dashed", color="red", size=2) +
     geom_vline(xintercept=mean(airquality$Temp), linetype="dotted", color="blue", size=2)

beta <- lm(Wind ~ Temp, data=airquality)$coefficients
pp + geom_abline(intercept=beta[1], slope=beta[2], color="red", size=2) +
     ggtitle(paste0("y = ", round(beta[1], 2), " + ",  round(beta[2], 2), " x"))
```

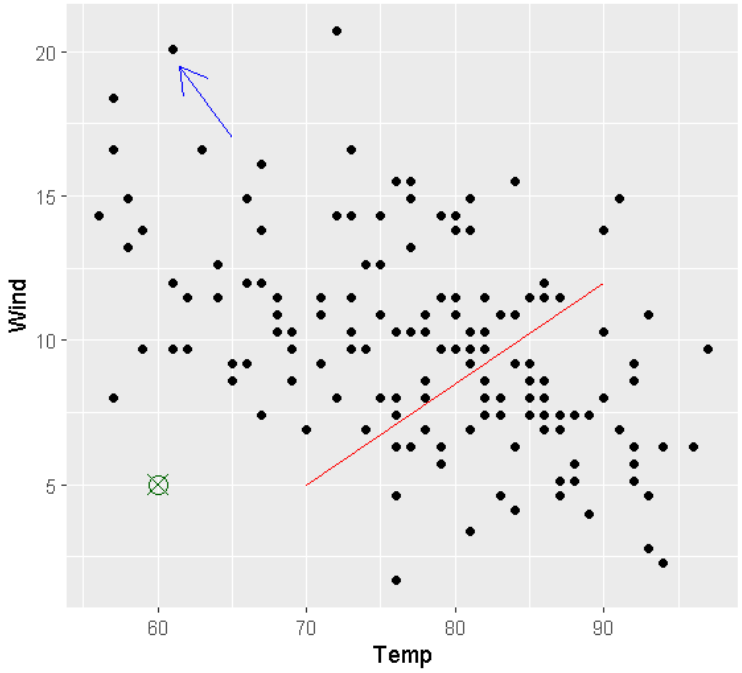


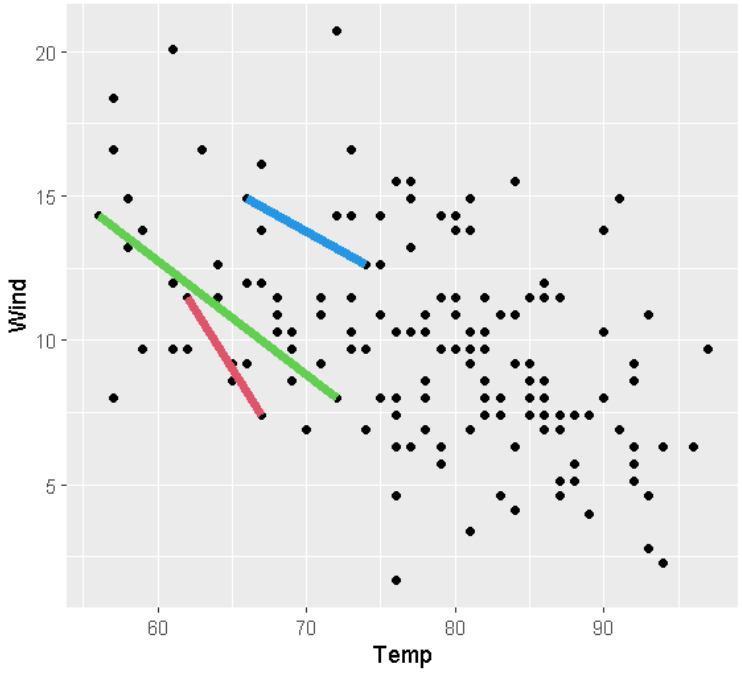




於圖上加線段、箭頭

```
pp + geom_segment(aes(x=70, y=5, xend=90, yend=12), color="red") +
     geom_segment(aes(x=65, y=17, xend=61.5, yend=19.5), color="blue",
                  arrow = arrow(length = unit(0.5, "cm"))) +
     geom_point(aes(x=60, y=5), color="darkgreen", shape=13, size=4)

xy.df <- data.frame(x1=airquality$Temp[1:3], y1=airquality$Wind[1:3],
                    x2=airquality$Temp[4:6], y2=airquality$Wind[4:6])
pp + geom_segment(data=xy.df, aes(x=x1, y=y1, xend=x2, yend=y2), color=2:4, size=2)

?geom_curve
```

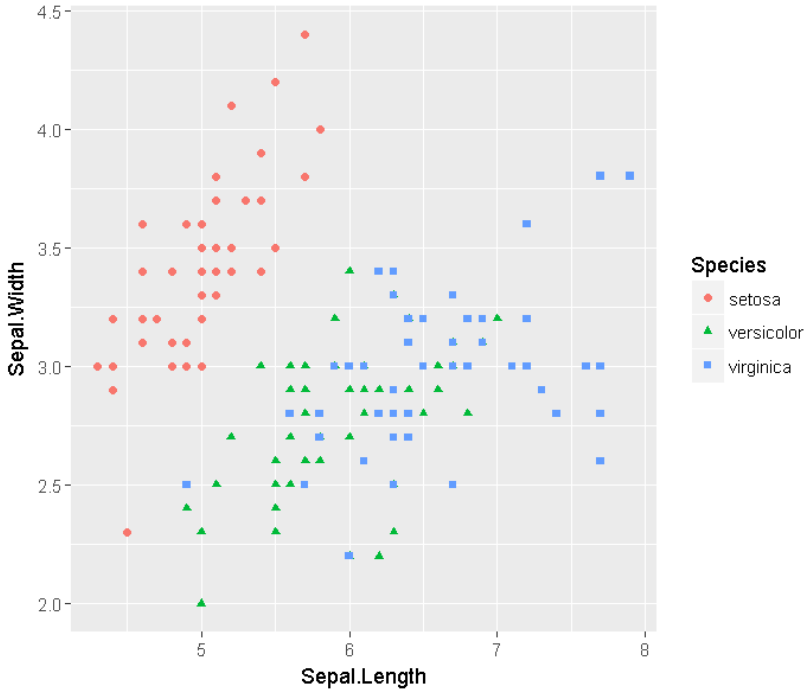


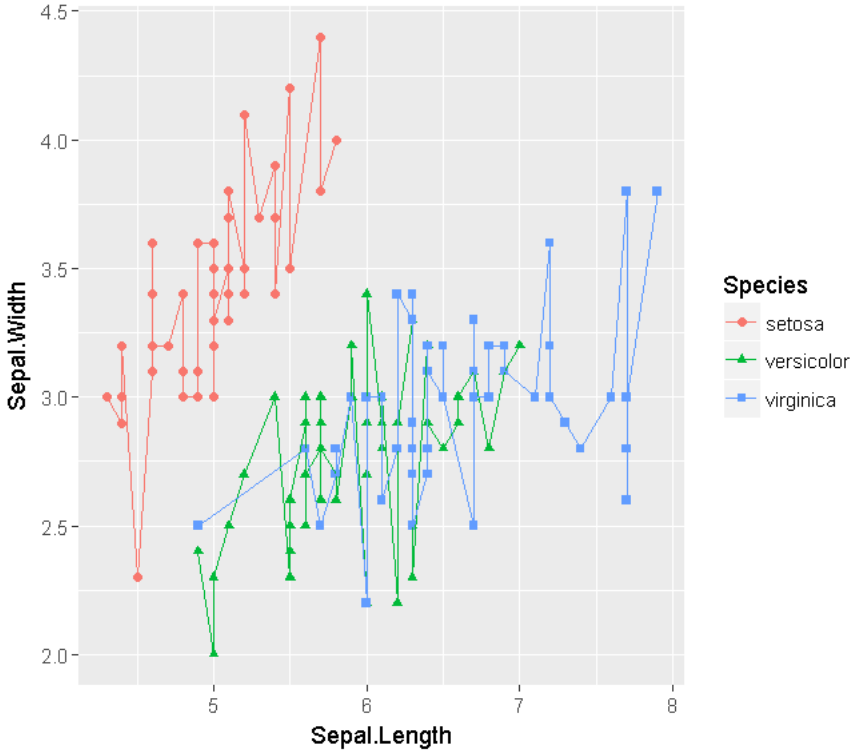


散佈圖 (Scatterplot)

```
> ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width,
      shape=Species, color=Species)) +
> geom_point()
```

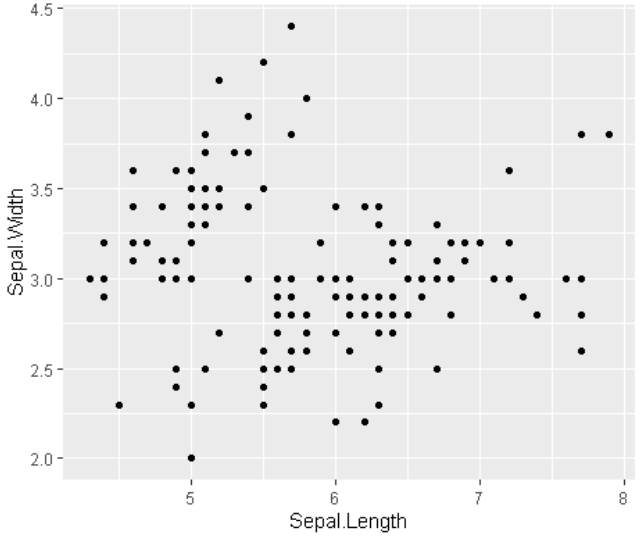
```
> p <- ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, shape=Species, color=Species))
> p <- p + geom_point()
> p
> p + geom_line(aes(y=Sepal.Width))
```

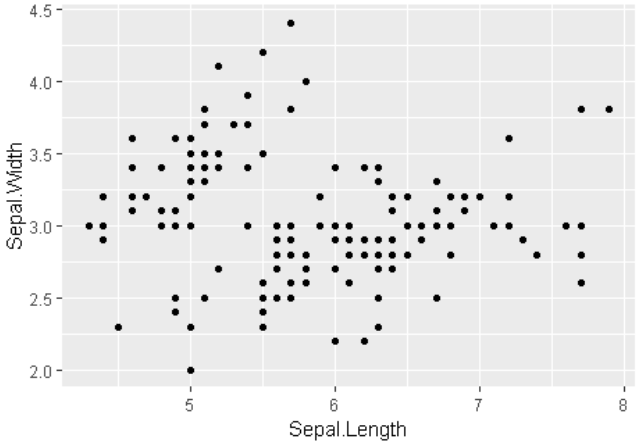


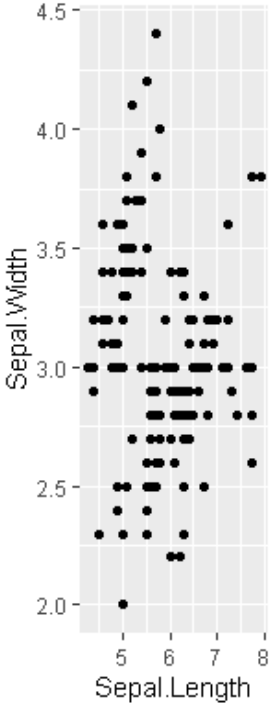


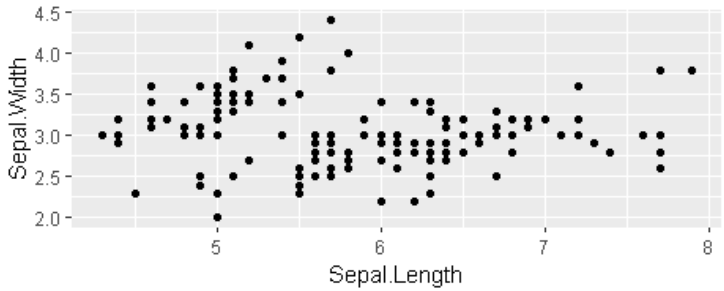
Scatterplot, 座標軸

```
> p <- ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width)) +
+     geom_point()
> p
> p + coord_fixed(ratio=1)
> p + coord_fixed(ratio=0.5)
> p + coord_fixed(ratio=5)
```



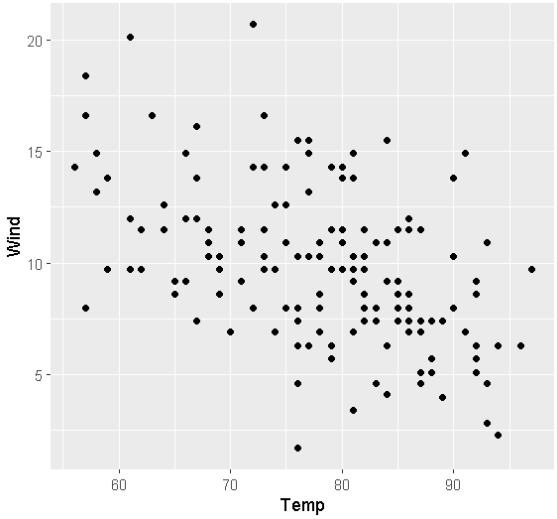


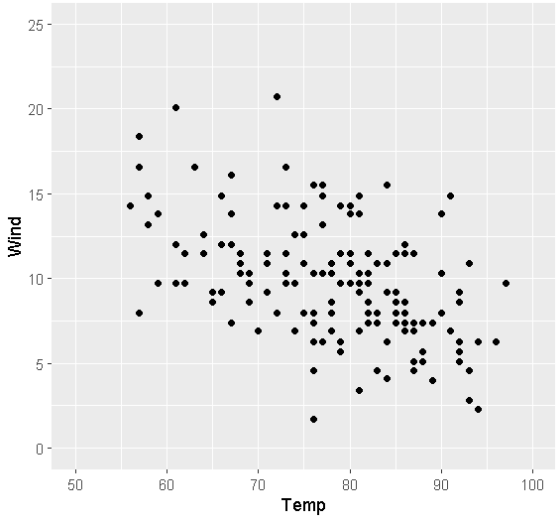


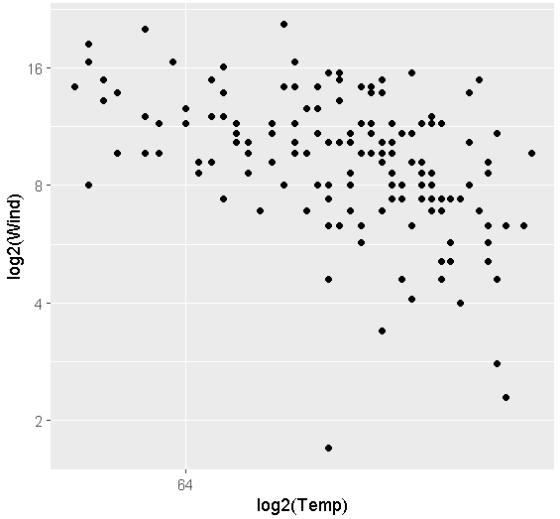


Scatterplot, 座標軸

```
# Change x and y axis limits
gp <- ggplot(airquality, aes(x=Temp, y=Wind)) +
        geom_point()
gp
gp + xlim(50, 100) + ylim(0, 25)
gp + expand_limits(x=c(50, 100), y=c(0, 25))


# Axis transformations
# trans: "log2", "log10", "sqrt"
gp + scale_x_continuous(trans="log2") + scale_y_continuous(trans="log2") +
     labs(x="log2(Temp)", y="log2(Wind)")
gp + coord_trans(x="log2", y="log2")
gp + scale_y_sqrt() # square root
gp + scale_y_reverse() # Reverse coordinates
```

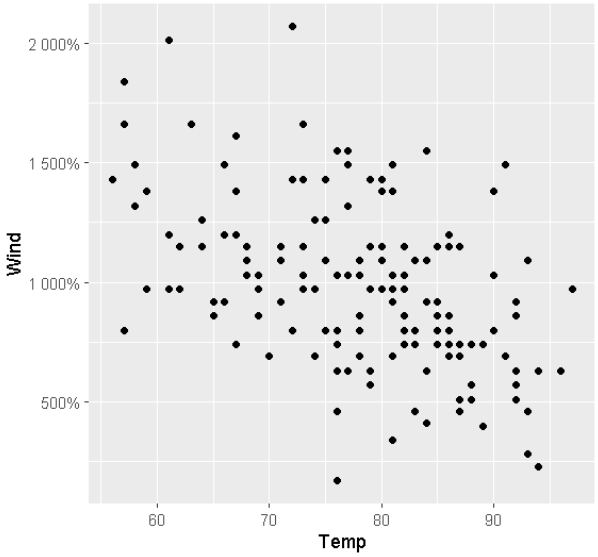


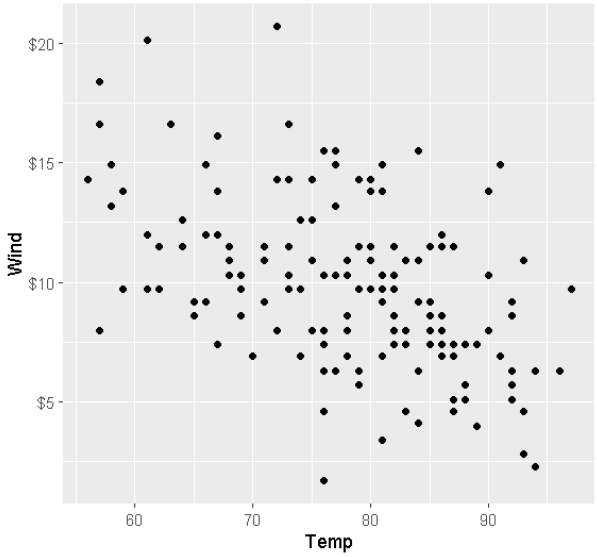


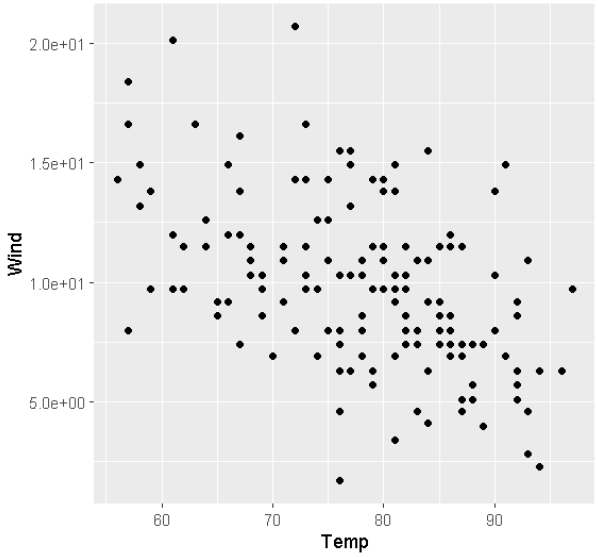


Scatterplot, 座標軸

```
gp + scale_y_continuous(labels = percent)
gp + scale_y_continuous(labels = dollar)
gp + scale_y_continuous(labels = scientific)
```

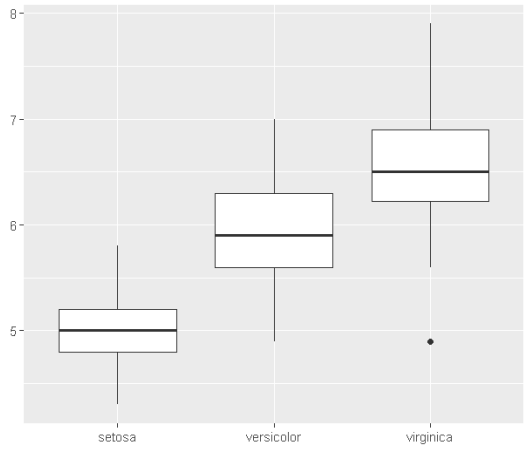


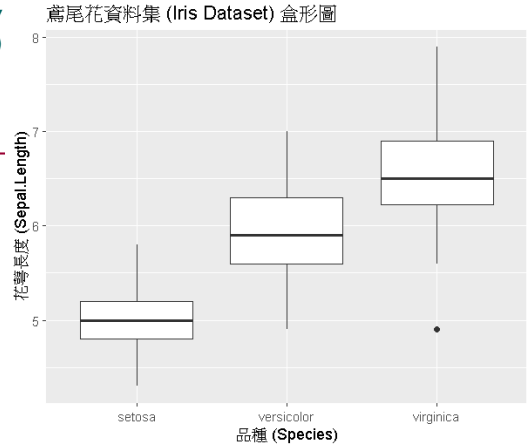


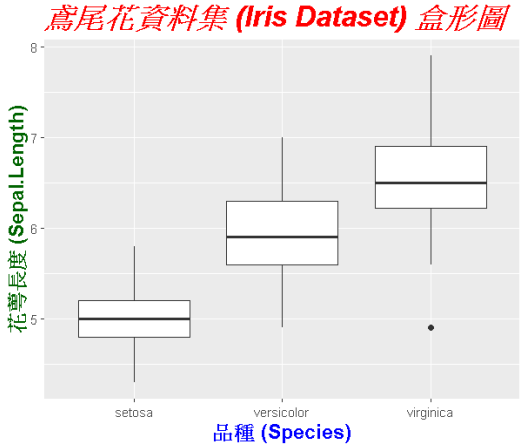


Customize the appearance of the main title and axis labels

```
p <- ggplot(iris, aes(x=Species, y=Sepal.Length)) +
       geom_boxplot() +
       ggtitle("鳶尾花資料集 (Iris Dataset) 盒形圖") +
       xlab("品種 (Species)") +
       ylab("花萼長度 (Sepal.Length)")
p
p + theme(plot.title=element_text(color="red", size=20, face="bold.italic"),
          axis.title.x=element_text(color="blue", size=14, face="bold"),
          axis.title.y=element_text(color="darkgreen", size=14, face="bold"))

# Hide the main title and axis titles
p + theme(plot.title=element_blank(),
          axis.title.x=element_blank(),
          axis.title.y=element_blank())
```





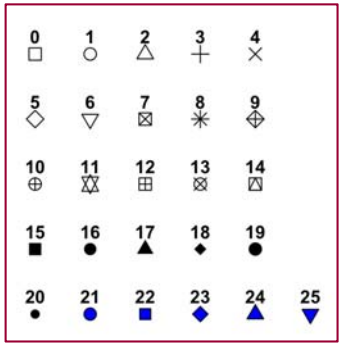



```
theme(plot.title = element_text(family, face, colour, size),
      axis.title.x = element_text(family, face, colour, size),
      axis.title.y = element_text(family, face, colour, size))
```

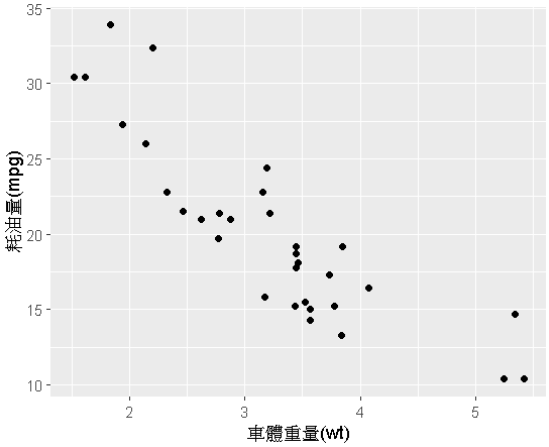
- family : font family
- face : font face. Possible values are “plain”, “italic”, “bold” and “bold.italic”

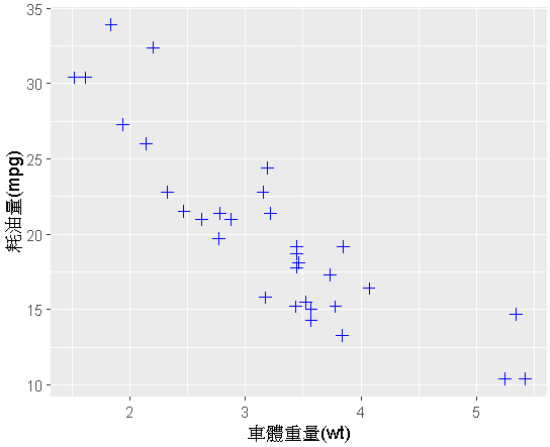
Scatterplot, 標題、資料點外形

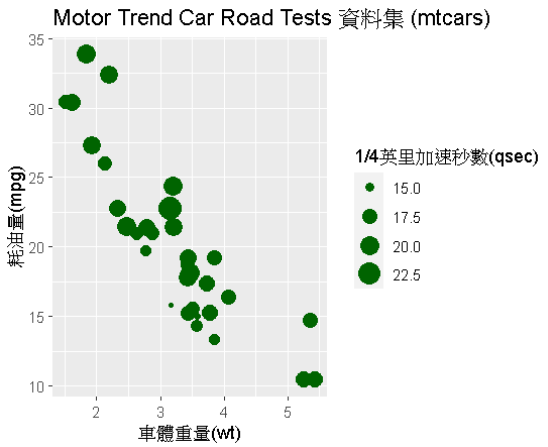
```
mtcars$cyl <- as.factor(mtcars$cyl)
head(mtcars)

ggplot(mtcars, aes(x=wt, y=mpg)) +
  geom_point() +
  labs(x="車體重量(wt)", y="耗油量(mpg)")

ggplot(mtcars, aes(x=wt, y=mpg)) +
  geom_point(size=2, color="blue", shape=3) +
  labs(x="車體重量(wt)", y="耗油量(mpg)")


ggplot(mtcars, aes(x=wt, y=mpg)) +
  geom_point(aes(size=qsec), color="darkgreen") +
  labs(x="車體重量(wt)", y="耗油量(mpg)", size="1/4英里加速秒數(qsec)",
       title="Motor Trend Car Road Tests 資料集 (mtcars)")
```




```
ggtitle("the main title")
xlab("the x axis label")
ylab("the y axis label")
labs(x="x label", y="y label",
     title="main title",
     fill="legend title")
```

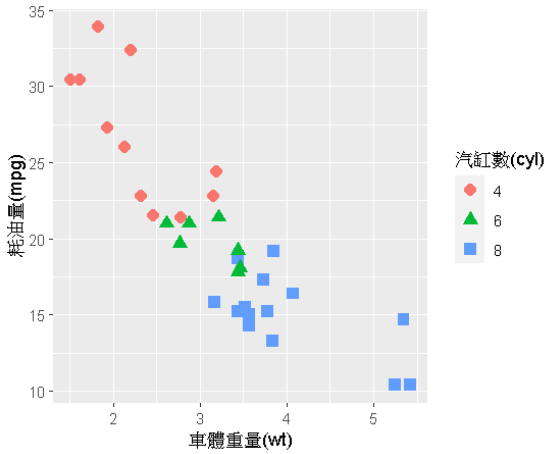


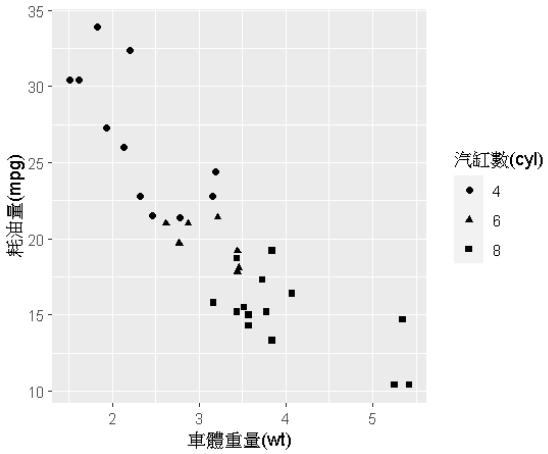


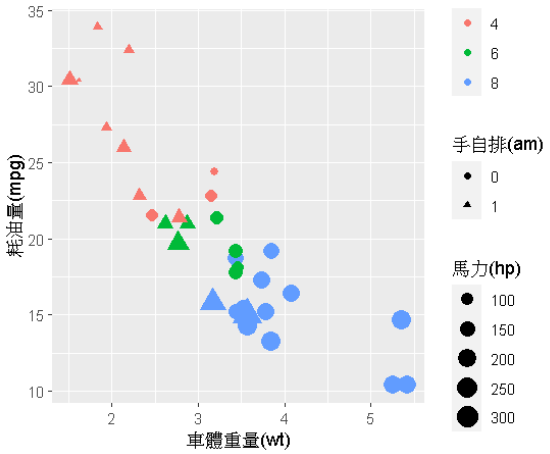


Scatterplot，資料點外形、顏色、大小

```
# mtcars$cyl <- as.factor(mtcars$cyl)
ggplot(mtcars, aes(x=wt, y=mpg, shape=cyl)) +
  geom_point() +
  labs(x="車體重量(wt)", y="耗油量(mpg)", shape="汽缸數(cyl)")

ggplot(mtcars, aes(x=wt, y=mpg, shape=cyl, color=cyl)) +
  geom_point(size=3) +
  labs(x="車體重量(wt)", y="耗油量(mpg)", shape="汽缸數(cyl)", color="汽缸數(cyl)")

mtcars$am <- as.factor(mtcars$am)
ggplot(mtcars, aes(x=wt, y=mpg, shape=am, color=cyl, size=hp)) +
  geom_point() +
  labs(x="車體重量(wt)", y="耗油量(mpg)", shape="手自排(am)", color="汽缸數(cyl)", size="馬力(hp)")
```

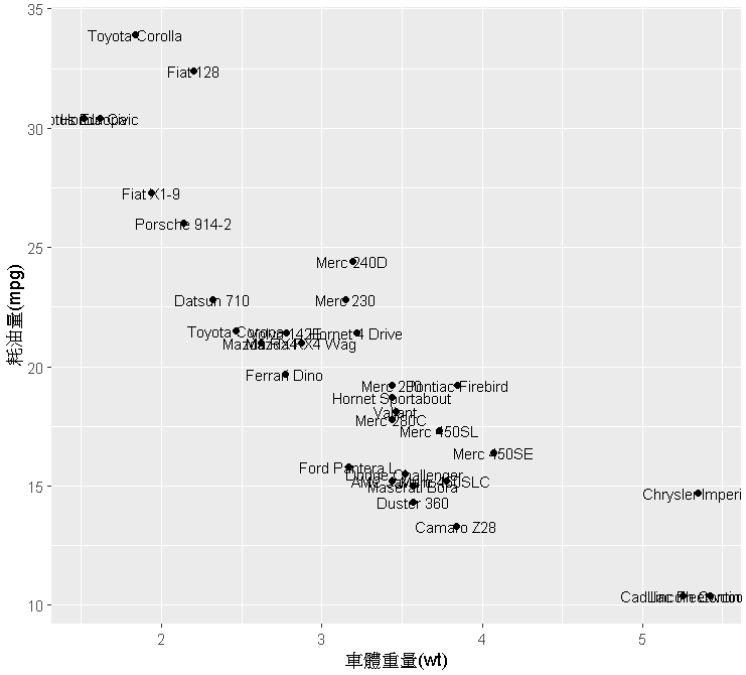


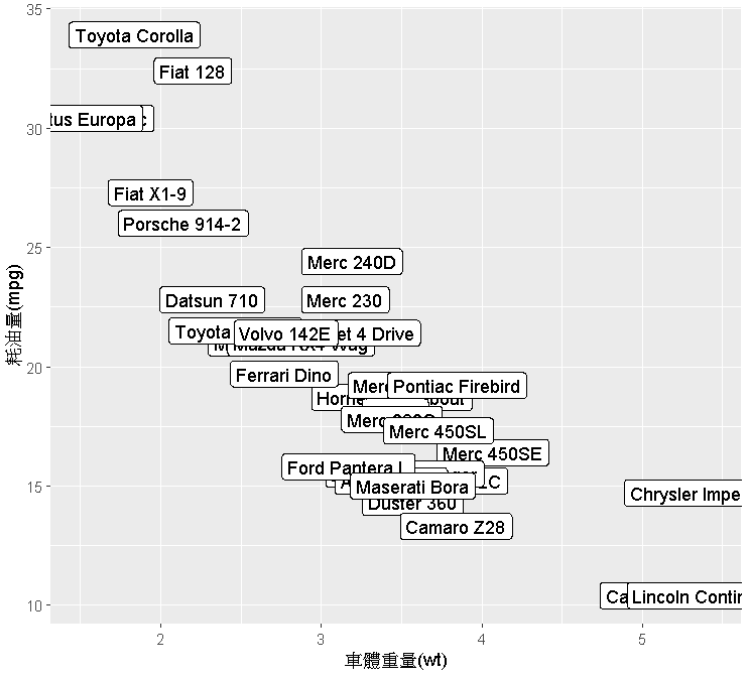




Scatterplot，文字標註

```
p <- ggplot(data=mtcars, aes(x=wt, y=mpg, label=rownames(mtcars))) +
       geom_point() +
       geom_text(size=3) +
       labs(x="車體重量(wt)", y="耗油量(mpg)")
p
p + geom_label()
```

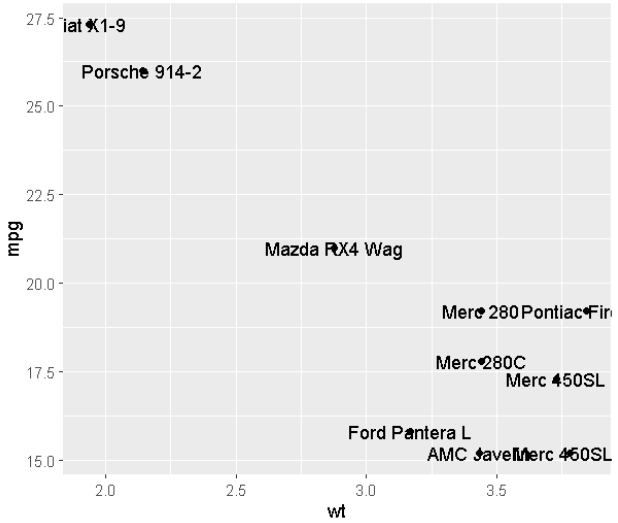


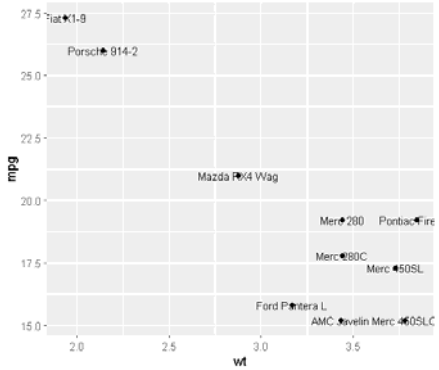


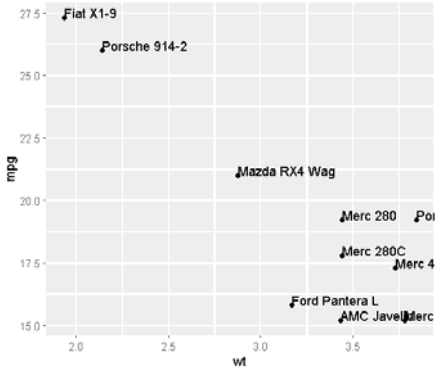
geom_text understands the following aesthetics (required aesthetics are in bold):
x, y, label, alpha, angle, colour, family, fontface, group, hjust, lineheight, size, vjust

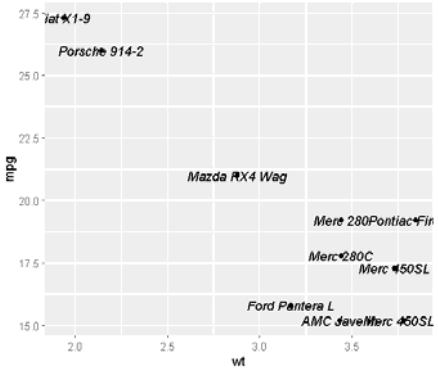
Add text annotations to a graph

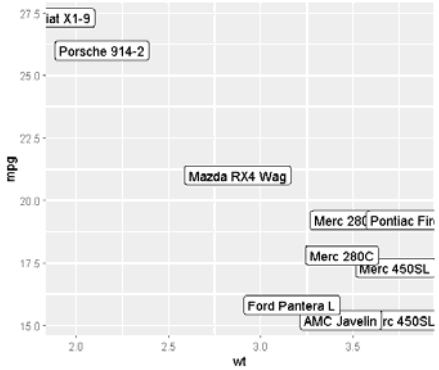
```
set.seed(123)
id <- sample(1:nrow(mtcars), 10)
mtcars.subset <- mtcars[id, ]

sp <- ggplot(mtcars.subset, aes(x=wt, y=mpg, label=rownames(mtcars.subset))) +
        geom_point()
sp + geom_text()
sp + geom_text(size=3)
sp + geom_text(hjust=0, vjust=0)

# 1(normal), 2(bold), 3(italic), 4(bold.italic)
sp + geom_text(aes(fontface=3))
sp + geom_label()
```





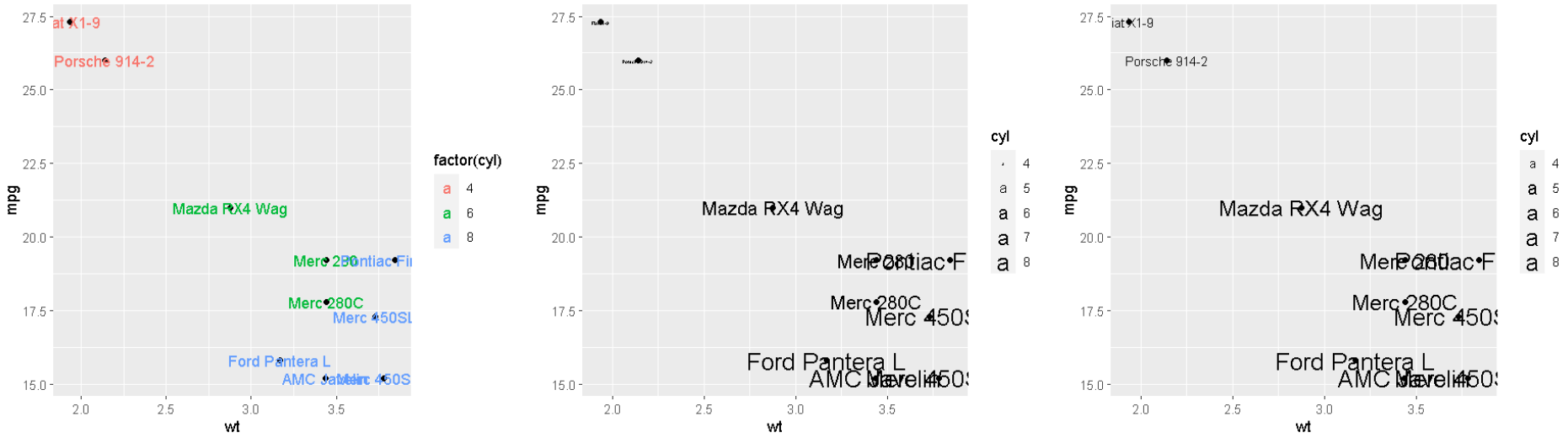






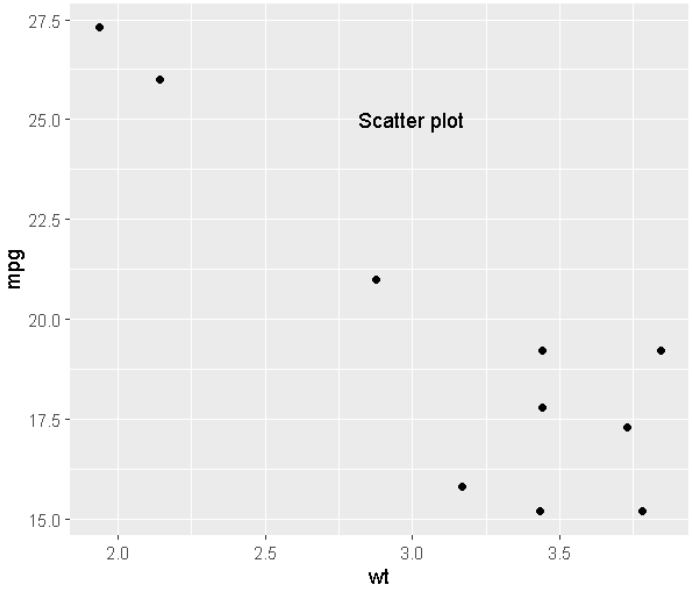
標註文字的大小

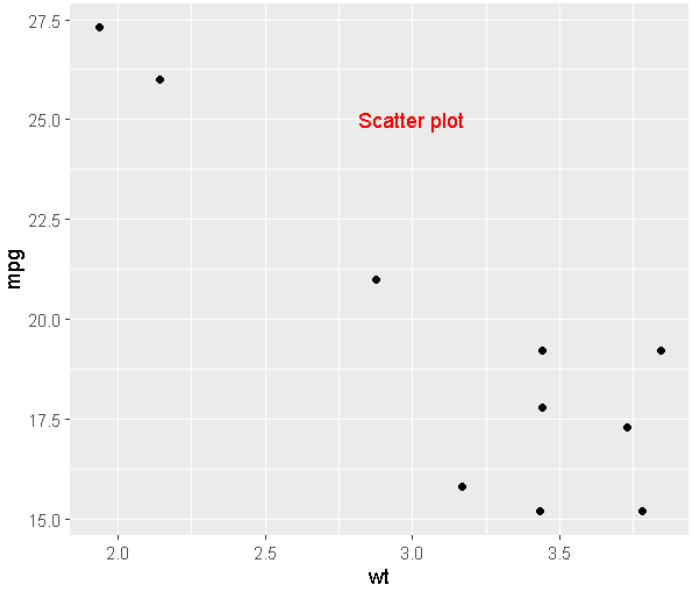
```
sp + geom_text(aes(color=factor(cyl)))
sp + geom_text(aes(size=cyl))
sp + geom_text(aes(size=cyl)) + scale_size(range=c(3, 6))
```



於圖上標註文字

```
sp + geom_text(x=3, y=25, label="Scatter plot")
sp + annotate(geom="text", x=3, y=25, label="Scatter plot", color="red")
```

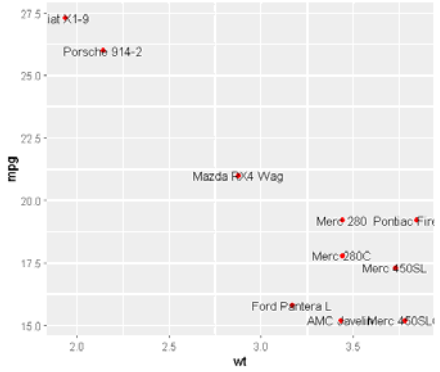


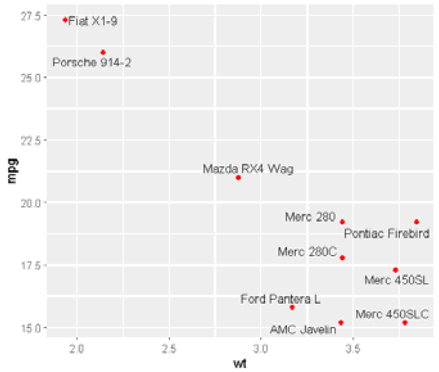


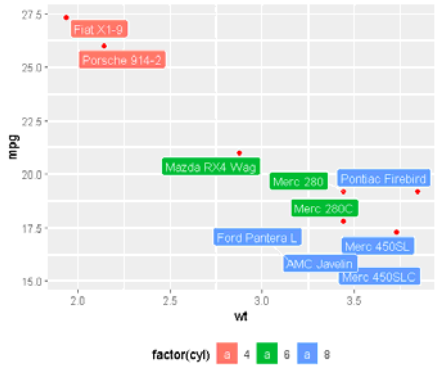
- `geom_text()`: adds text directly to the plot
- `geom_label()`: draws a rectangle underneath the text, making it easier to read.
- `annotate()`: adding small text annotations at a particular location on the plot
- `annotation_custom()`: Adds static annotations that are the same in every panel

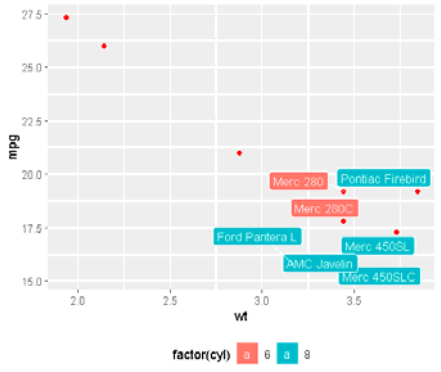
標註文字不重疊

```
# ggrepel: Avoid overlapping of text labels
# install.packages("ggrepel")
require("ggrepel")

sp2 <- ggplot(mtcars.subset, aes(x=wt, y=mpg, label=rownames(mtcars.subset))) +
         geom_point(color="red")

sp2 + geom_text(size = 3.5)

sp2 + geom_text_repel(size = 3.5)

sp2 + geom_label_repel(aes(fill=factor(cyl)), color="white", size=3.5) +
   theme(legend.position="bottom")

label.sub <- subset(mtcars.subset, wt > 3 & mpg < 20)
sp2 + geom_label_repel(data=label.sub,
                       aes(label=rownames(label.sub), fill=factor(cyl)),
                       color="white", size=3.5) +
      theme(legend.position="bottom")
```



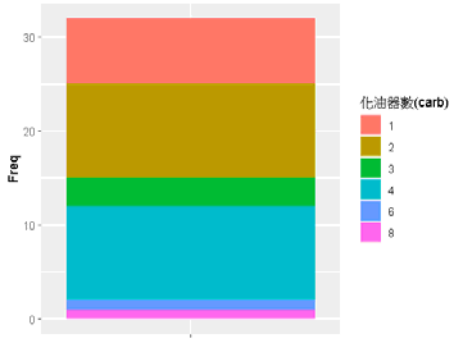


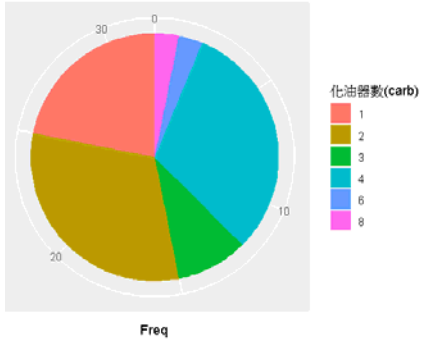


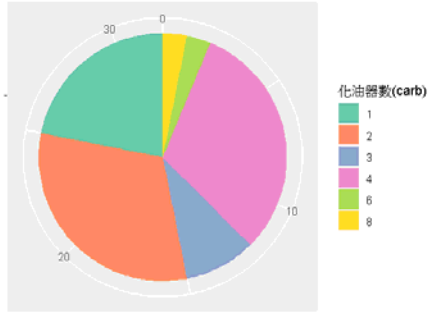


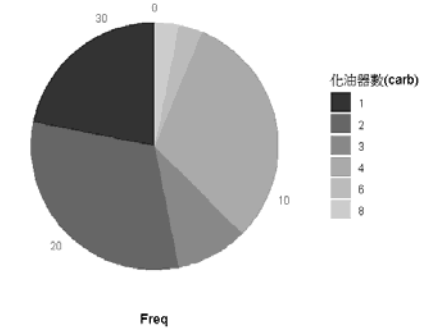
圓餅圖 (Pie chart)

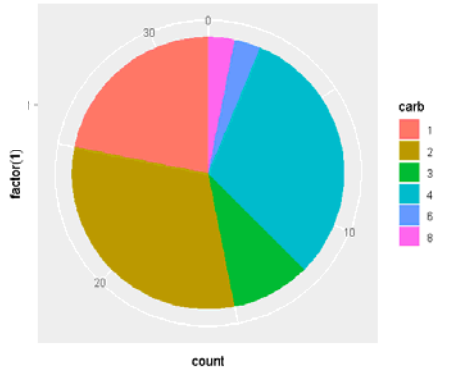
```
> carb.df <- data.frame(table(mtcars$carb))
> names(carb.df) <- c("carb", "Freq")
> carb.df
  carb Freq
1    1    7
2    2   10
3    3    3
4    4   10
5    6    1
6    8    1
>
> bar.pt <- ggplot(carb.df, aes(x="", y=Freq, fill=carb)) +
+            geom_bar(width=1, stat="identity") +
+            labs(x="", fill="化油器數(carb)")
> bar.pt
>
> pie <- bar.pt + coord_polar("y", start=0)
> pie
> pie + scale_fill_brewer(palette="Set2")
> pie + scale_fill_grey() + theme_minimal()
>
> mtcars$carb <- factor(mtcars$carb)
> ggplot(mtcars, aes(x=factor(1), fill=carb)) +
+   geom_bar(width = 1) +
+   coord_polar("y")
```





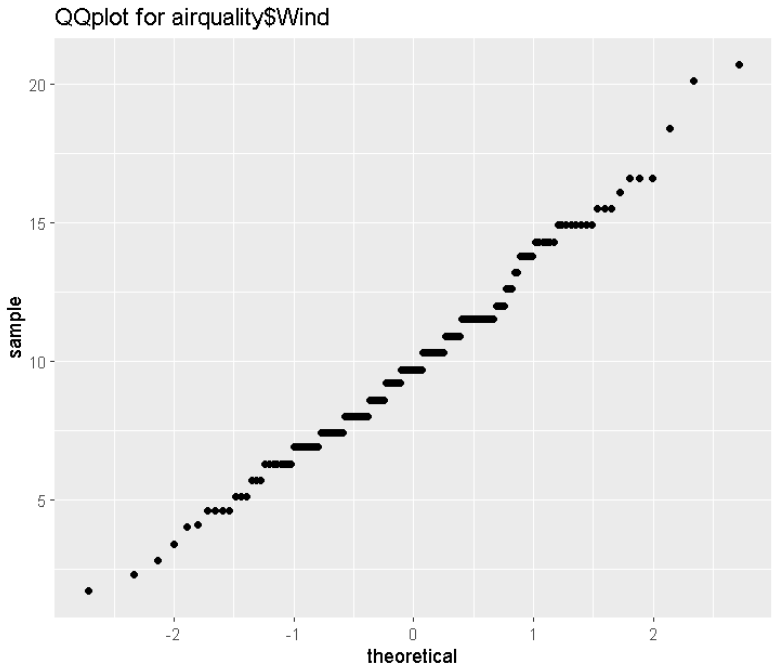


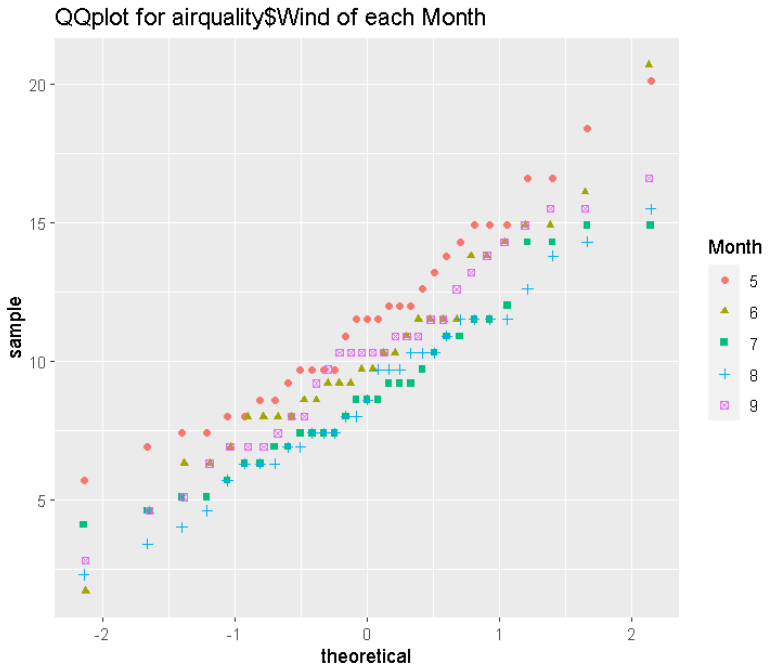




QQplot (quantile-quantile plot)

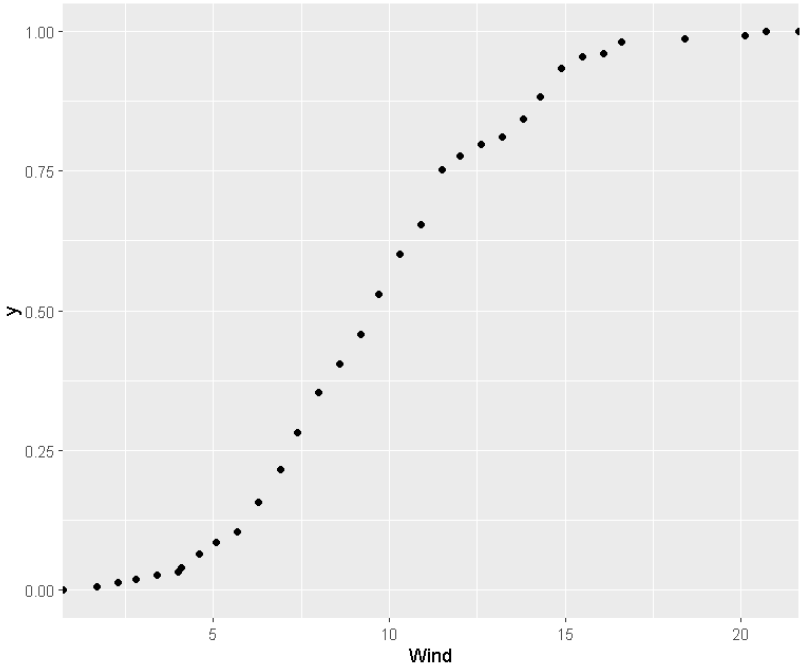
```
ggplot(airquality, aes(sample=Wind)) +
  stat_qq() +
  labs(title="QQplot for airquality$Wind")

ggplot(airquality, aes(sample=Wind, shape=Month, color=Month)) +
  stat_qq() +
  labs(title="QQplot for airquality$Wind of each Month")
```

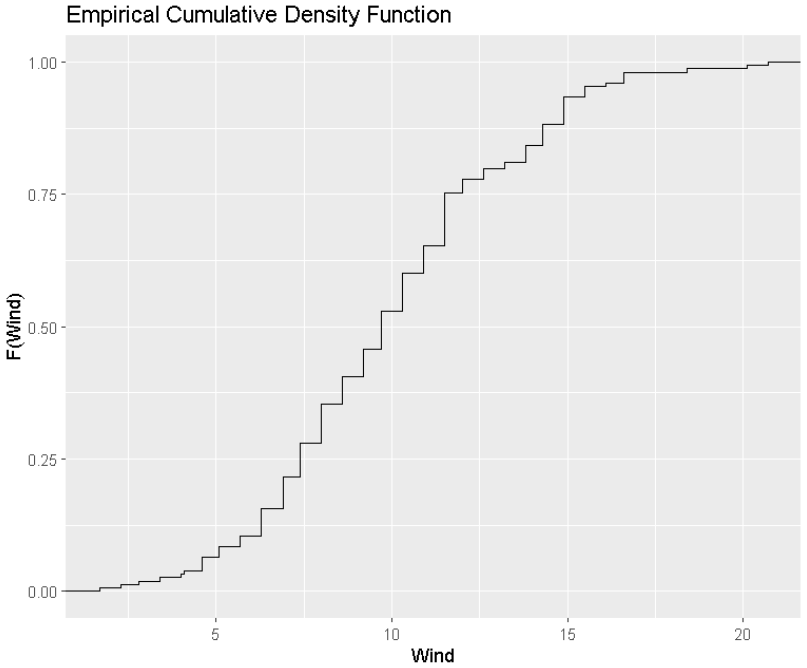




Empirical Cumulative Density Function

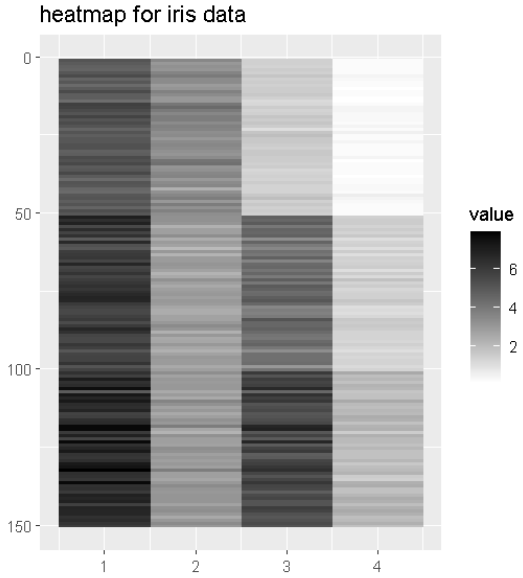
```
ggplot(airquality, aes(x=Wind)) +
  stat_ecdf(geom = "point")

ggplot(airquality, aes(x=Wind)) +
  stat_ecdf(geom = "step") +
  labs(title="Empirical Cumulative Density Function",
       y = "F(Wind)", x="Wind")
```

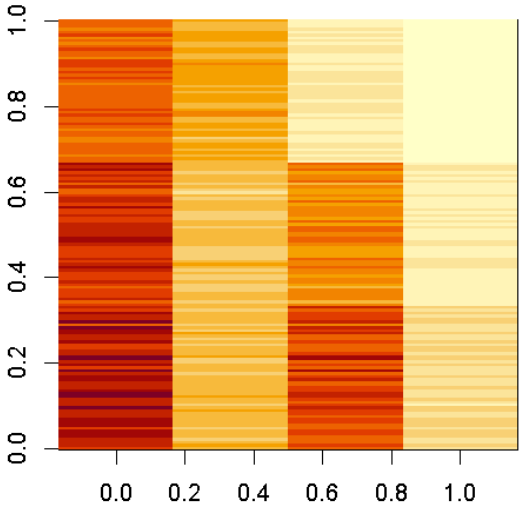


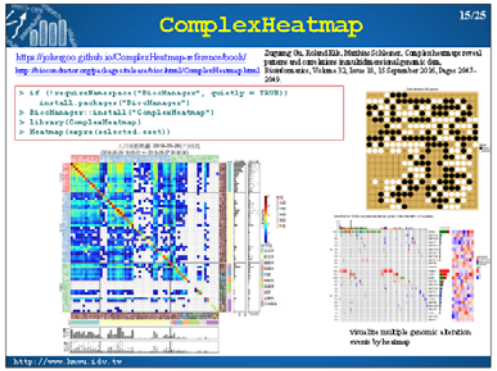


Basic heatmap (熱圖) with ggplot2

```
library(tidyr)
xdata <- iris[, 1:4]
n <- nrow(xdata)
p <- ncol(xdata)
iris.df <- data.frame(x=rep(1:p, each=n), y=rep(1:n, p),
                      value=gather(xdata)$value)
str(iris.df)
ggplot(iris.df, aes(x=x, y=y, fill=value)) +
  geom_raster() +
  scale_fill_gradient(low="white", high="black", na.value=NA) +
  scale_y_reverse() +
  labs(x="", y="", title="heatmap for iris data")

image(t(iris[, 1:4])[, nrow(iris[, 1:4]):1])
```







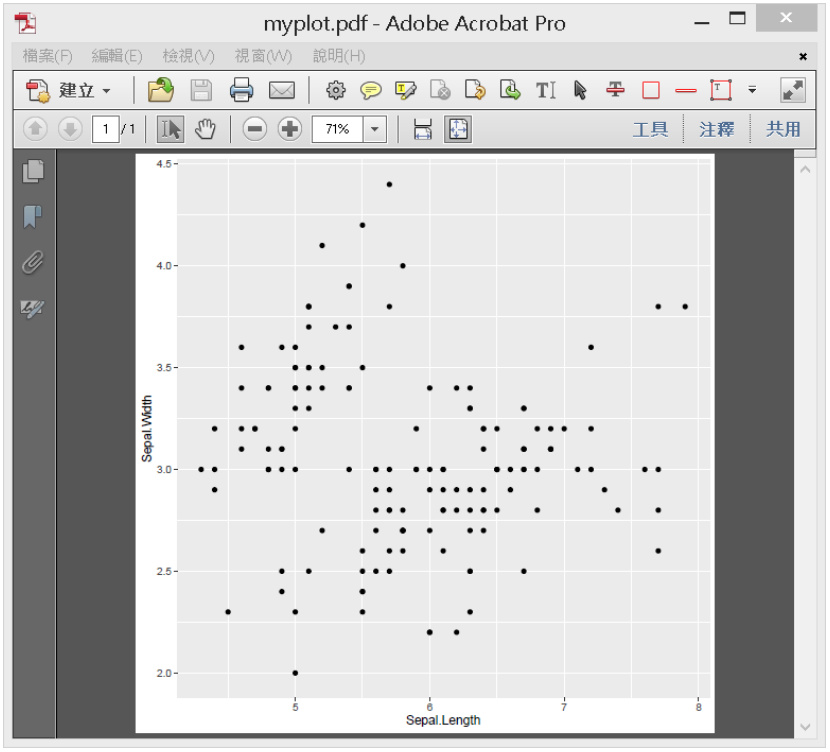
http://www.hmwu.idv.tw/web/R/E06-hmwu_R-heatmap.pdf

Save a ggplot object

```
> # print(): print a ggplot to a file
> myplot <- ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width)) +
+             geom_point()
> pdf("myplot.pdf") # or png("myplot.png")
> print(myplot)
> dev.off()
windows
      2
```

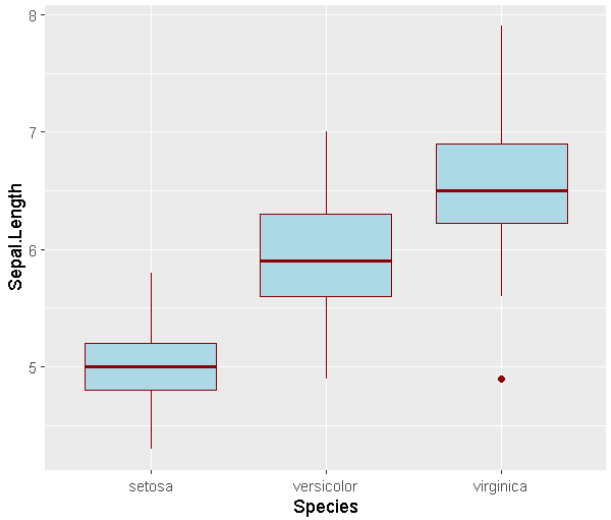
```
> getwd()
> list.files()
```

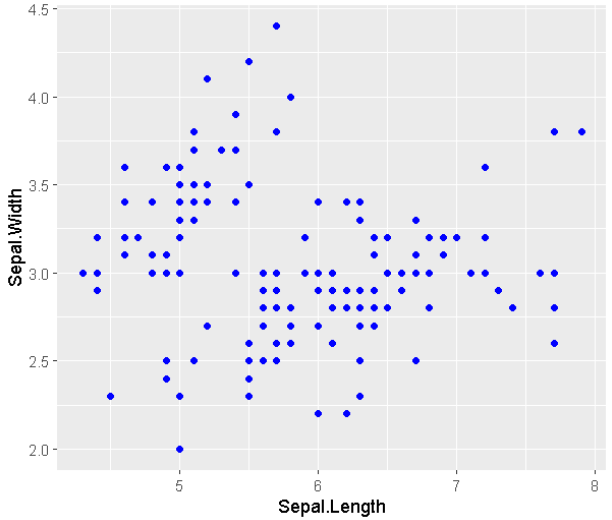
```
> # ggsave: save the last ggplot
> ggplot(mtcars, aes(wt, mpg)) + geom_point()
> ggsave("myplot.png")
Saving 6.06 x 5.24 in image
>
> # ggsave: save a ggplot object
> ggsave(file="myplot2.pdf",
         plot=myplot,
         device="pdf",
         scale=1.5)
Saving 9.09 x 7.86 in image
```



顏色 (1)

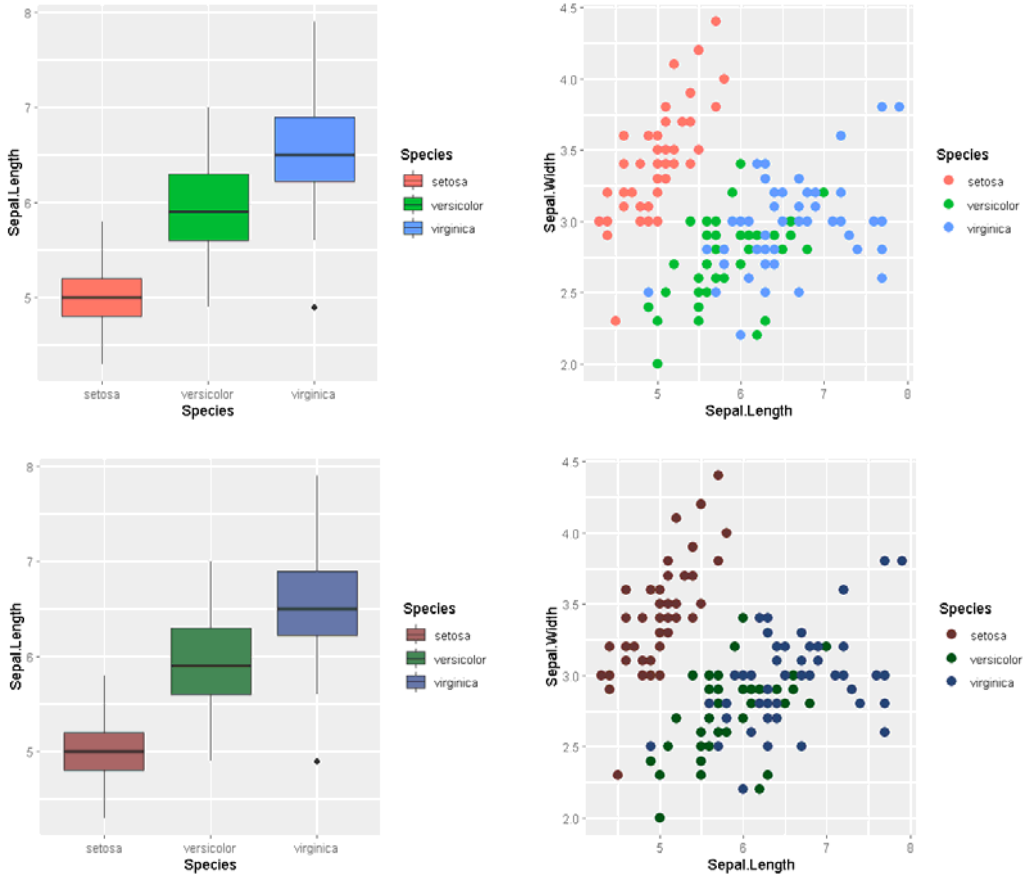
```
ggplot(iris, aes(x=Species, y=Sepal.Length)) +
  geom_boxplot(fill="lightblue", color="darkred")

ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width)) +
  geom_point(color="blue")
```





顏色 (2)

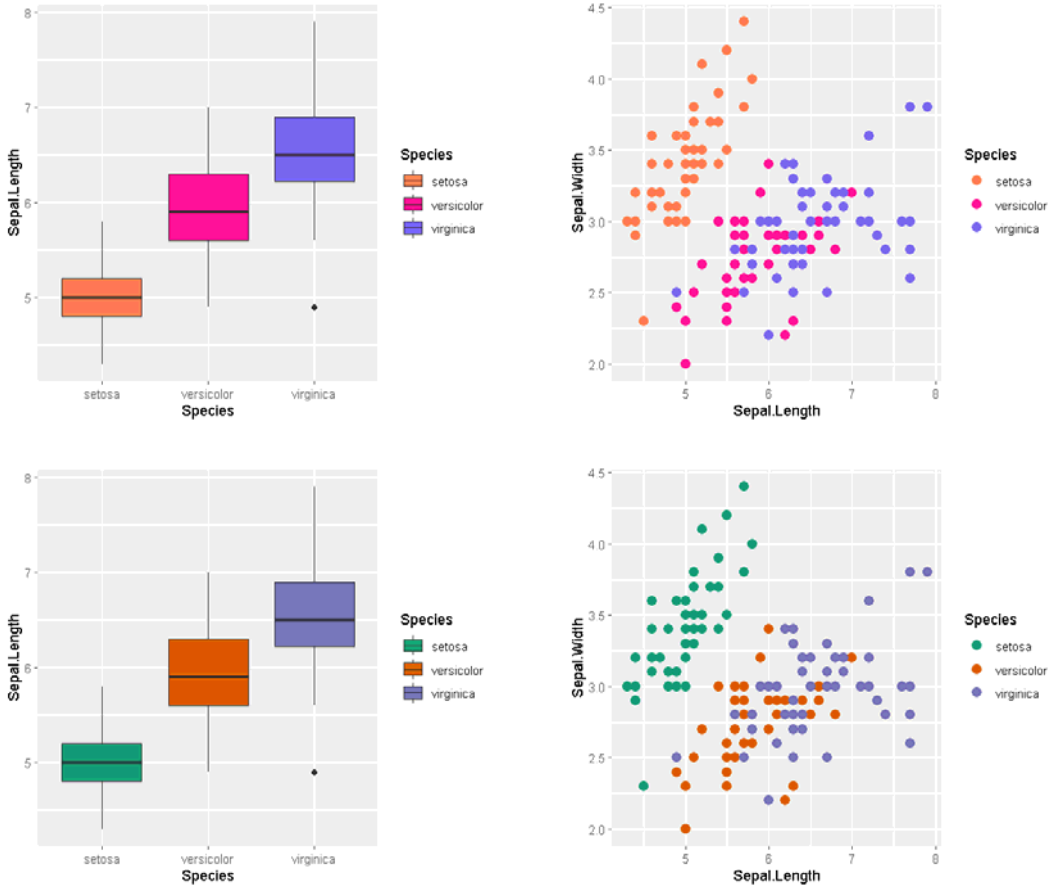
```
bp <- ggplot(iris, aes(x=Species, y=Sepal.Length, fill=Species)) + geom_boxplot()
bp
sp <- ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) + geom_point(size=3)
sp

bp + scale_fill_hue(l=50, c=40)
sp + scale_color_hue(l=30, c=40)
```

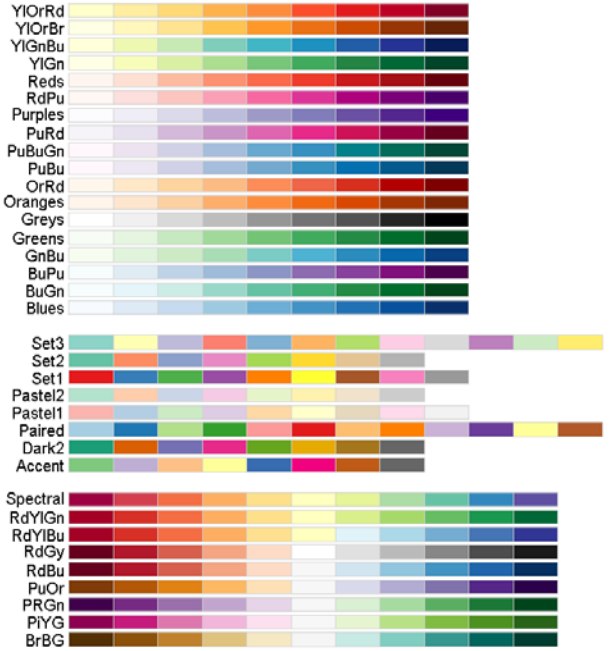


`h`: range of hues to use: [0, 360]
`c`: chroma (intensity of colour), maximum value varies depending on combination of hue and luminance.
`l`: luminance (lightness): [0, 100]

顏色 (3)

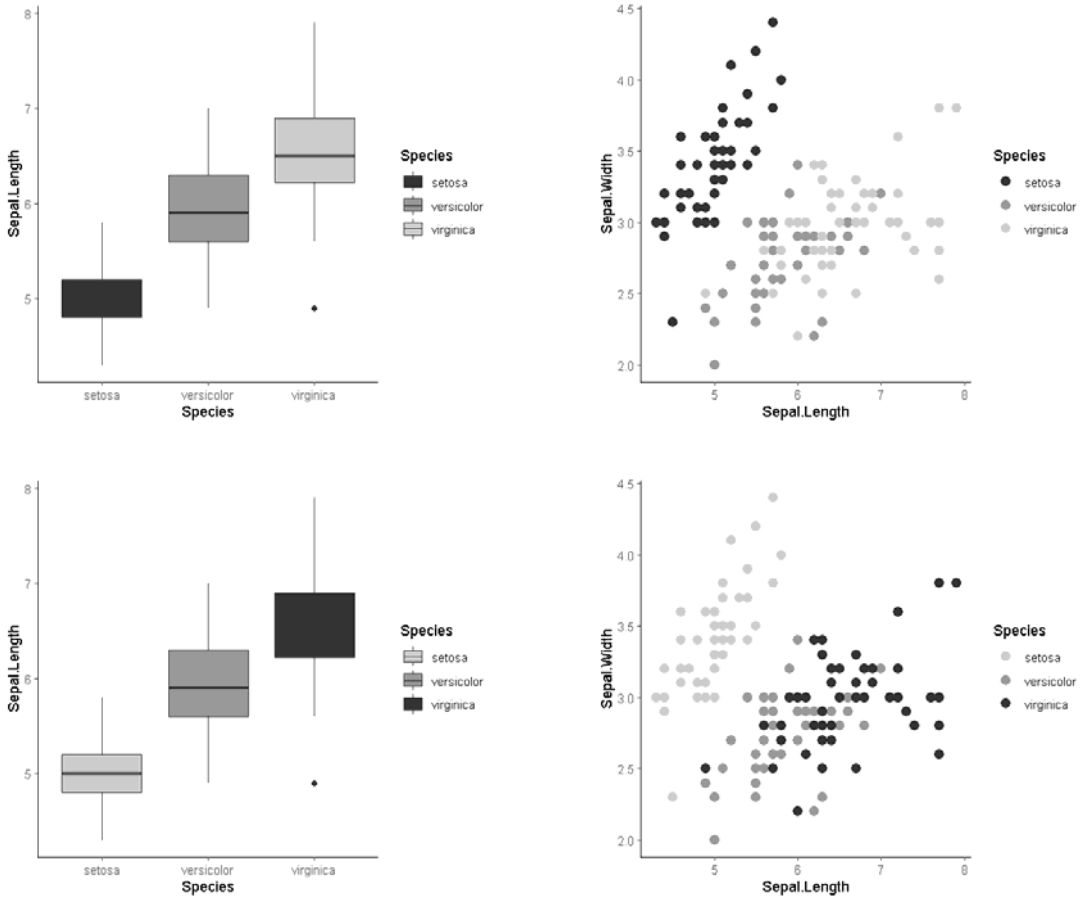
```
bp + scale_fill_manual(values=c("coral", "deeppink", "slateblue2"))
sp + scale_color_manual(values=c("coral", "deeppink", "slateblue2"))

# Use RColorBrewer palettes
bp + scale_fill_brewer(palette="Dark2")
sp + scale_color_brewer(palette="Dark2")
```





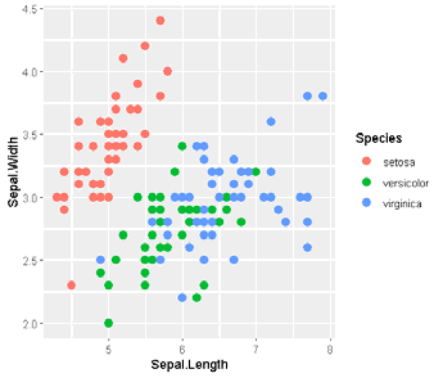
顏色 (4)

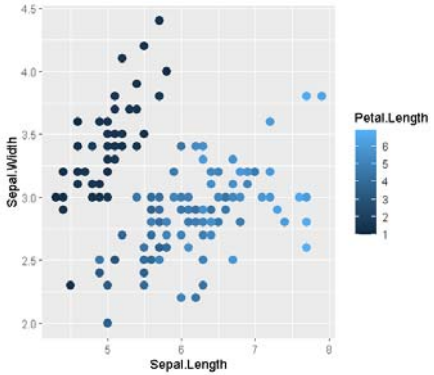
```
# Use gray colors, theme_classic(): turn bg white
bp + scale_fill_grey() + theme_classic()
sp + scale_color_grey() + theme_classic()

bp + scale_fill_grey(start=0.8, end=0.2) + theme_classic()
sp + scale_color_grey(start=0.8, end=0.2) + theme_classic()
```

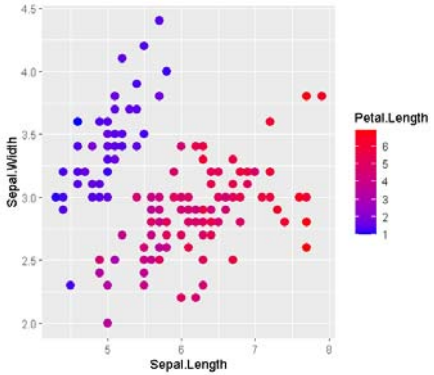


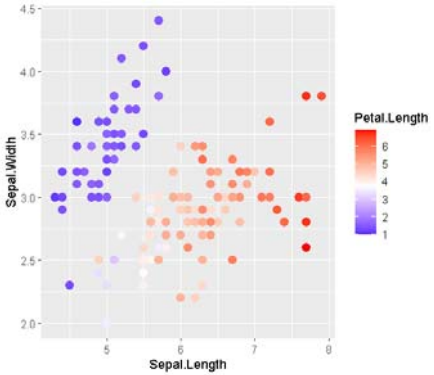
顏色 (5)

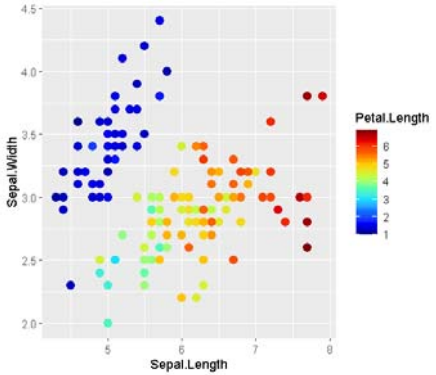
```
# Continuous colors
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point(size=3)
sc <- ggplot(iris, aes(x=Sepal.Length,
                       y=Sepal.Width,
                       color=Petal.Length)) +
        geom_point(size=3)
sc

# Sequential color scheme
sc + scale_color_gradient(low="blue", high="red")

# Diverging color scheme
mid.value <- mean(iris$Petal.Length)
sc + scale_color_gradient2(midpoint=mid.value, low="blue", mid="white", high="red")

# Gradient between n colors
library(fields)
sc + scale_color_gradientn(colours=tim.colors(10))
```





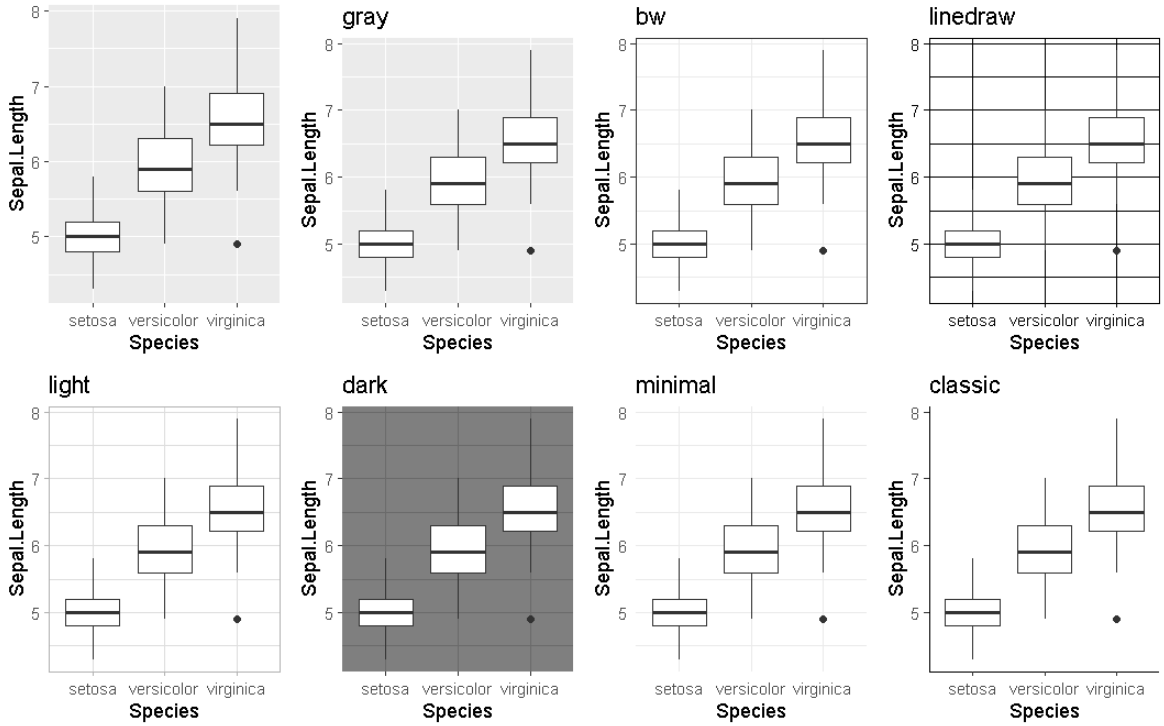






佈景主題 (ggplot2 themes)

```
library(gridExtra)
p <- ggplot(iris, aes(x=Species, y=Sepal.Length)) + geom_boxplot()
p1 <- p + theme_gray() + labs(title="gray")
p2 <- p + theme_bw() + labs(title="bw")
p3 <- p + theme_linedraw() + labs(title="linedraw")
p4 <- p + theme_light() + labs(title="light")
p5 <- p + theme_dark() + labs(title="dark")
p6 <- p + theme_minimal() + labs(title="minimal")
p7 <- p + theme_classic() + labs(title="classic")
grid.arrange(p, p1, p2, p3, p4, p5, p6, p7, nrow=2, ncol=4)
```



Themes

r + theme_bw()
White background with grid lines

r + theme_gray()
Grey background (default theme)

r + theme_dark()
dark for contrast

r + theme_classic()
r + theme_light()
r + theme_linedraw()

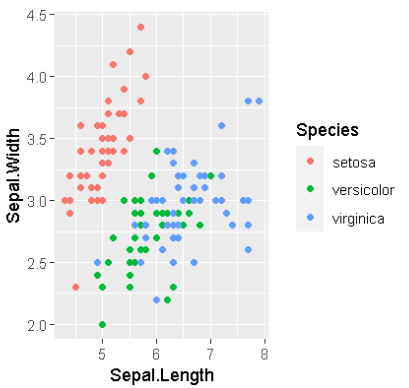
r + theme_minimal()
Minimal themes

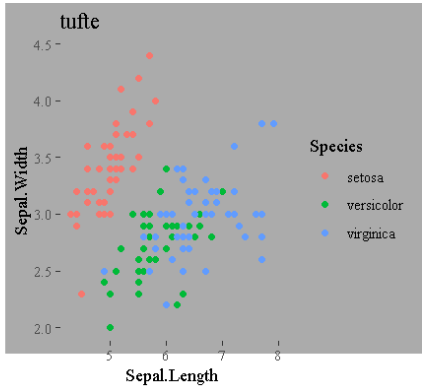
r + theme_void()
Empty theme

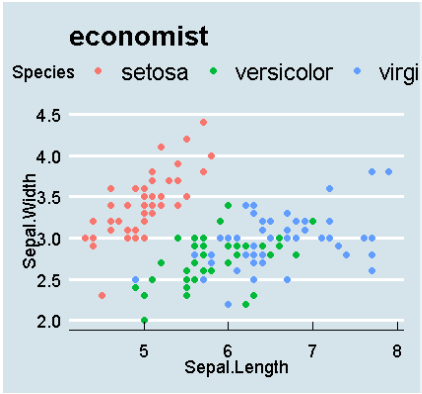
ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'

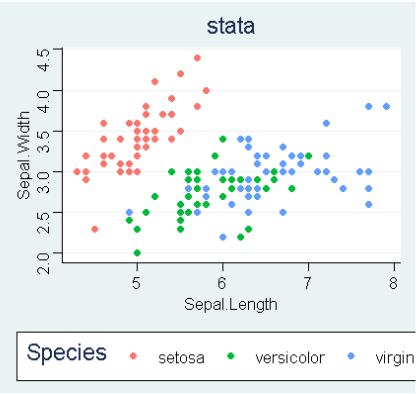
```
# install.packages("ggthemes")
library(ggthemes)
sp <- ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) + geom_point()
sp1 <- sp + theme_tufte() + labs(title="tufte") # a minimalist theme
sp2 <- sp + theme_economist() + labs(title="economist")
sp3 <- sp + theme_stata() + labs(title="stata")
sp4 <- sp + theme_hc() + labs(title="hc") # Highcharts JS
sp5 <- sp + theme_wsj() + labs(title="wsj") # Wall Street Journal
grid.arrange(sp, sp1, sp2, sp3, sp4, sp5, nrow=2, ncol=3)
```

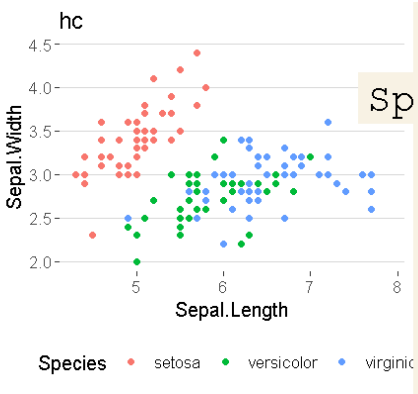
```
theme_base, theme_calc, theme_clean,
theme_economist, theme_excel,
theme_excel_new, theme_few,
theme_fivethirtyeight, theme_foundation,
theme_gdocs, theme_hc, theme_igray,
theme_map, theme_pander, theme_par
theme_solarized, theme_solid, theme_stata,
theme_tufte, theme_wsj.
```

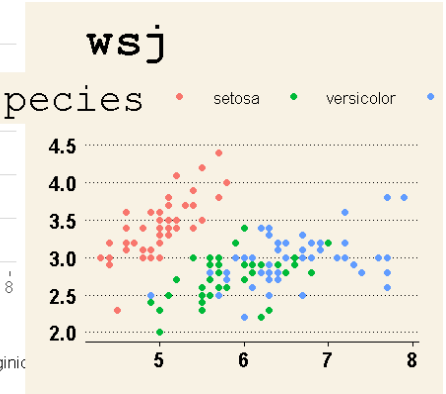












ggplot2: Geoms

Geoms: Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

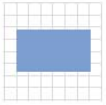
```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank()
(Useful for expanding limits)

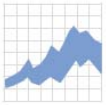
b + geom_curve(aes(yend = lat + 1, xend=long+1),curvature=1**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1**)**
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group)**)**
x, y, alpha, color, fill, group, linetype, size



b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size



a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)**)** - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1)**)**
b + geom_hline(aes(yintercept = lat)**)**
b + geom_vline(aes(xintercept = long)**)**

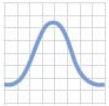
b + geom_segment(aes(yend=lat+1, xend=long+1)**)**
b + geom_spoke(aes(angle = 1:1155, radius = 1)**)**

ggplot2: One variable

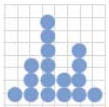
ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

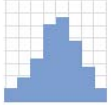


c + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight

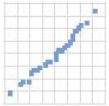


c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size



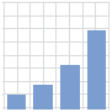
c + geom_histogram(binwidth = 5**)** x, y, alpha, color, fill, linetype, size, weight



c2 + geom_qq(aes(sample = hwy)**)** x, y, alpha, color, fill, linetype, size, weight

discrete

d <- ggplot(mpg, aes(fl))



d + geom_bar()
x, alpha, color, fill, linetype, size, weight

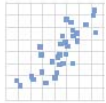
ggplot2: Two variables

TWO VARIABLES

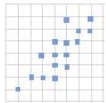
continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

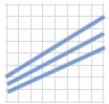
e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)** x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust



e + geom_jitter(height = 2, width = 2**)** x, y, alpha, color, fill, shape, size

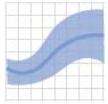


e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

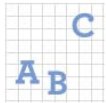


e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"**)**, x, y, alpha, color, linetype, size



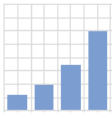
e + geom_smooth(method = lm**)**, x, y, alpha, color, fill, group, linetype, size, weight



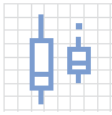
e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**, x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

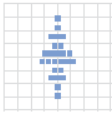
f <- ggplot(mpg, aes(class, hwy))



f + geom_col(), x, y, alpha, color, fill, group, linetype, size



f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

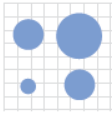


f + geom_dotplot(binaxis = "y", stackdir = "center"**)**, x, y, alpha, color, fill, group

f + geom_violin(scale = "area"**)**, x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))



g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

ggplot2: Two variables

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

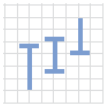
i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

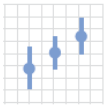
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

j + geom_crossbar(fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, group, linetype, size



j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size



j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map**)**
+ expand_limits(x = map$long, y = map$lat**)**,
map_id, alpha, color, fill, linetype, size

ggplot2: Three variables

THREE VARIABLES

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype,
size, weight

l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5,
interpolate=FALSE**)**
x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill,
linetype, size, width

ggplot2: Stats

Stats An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).

Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function). Use **..name..** syntax to map stat variables to aesthetics.

geom to use | stat function | geommappings

i + stat_density2d(aes(fill = ..level..), geom = "polygon")

variable created by stat

c + stat_bin(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..

c + stat_count(width = 1) **x, y,** | ..count.., ..prop..

c + stat_density(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd=0.5)) **x** | ..x.., ..y..

e + stat_identity(na.rm = TRUE)

ggplot() + stat_qq(aes(sample=1:100), dist = qt, dparam=list(df=5)) **sample, x, y** | ..sample.., ..theoretical..

e + stat_sum() x, y, size | ..n.., ..prop..

e + stat_summary(fun.data = "mean_cl_boot")

h + stat_summary_bin(fun.y = "mean", geom = "bar")

e + stat_unique()

e + stat_bin_2d(bins = 30, drop = T)
x, y, fill | ..count.., ..density..

e + stat_bin_hex(bins=30) **x, y, fill** | ..count.., ..density..

e + stat_density_2d(contour = TRUE, n = 100)
x, y, color, size | ..level..

e + stat_ellipse(level = 0.95, segments = 51, type = "t")

l + stat_contour(aes(z = z)) **x, y, z, order** | ..level..

l + stat_summary_hex(aes(z = z), bins = 30, fun = max)
x, y, z, fill | ..value..

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5) **x, y** | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area") **x, y** | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40) **x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") **x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se=T, level=0.95) **x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot2: Scales

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

```
(n <- d + geom_bar(aes(fill = fl)))
```

```
n + scale_fill_manual(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))
```

scale_ | aesthetic to adjust | prepackaged scale to use | scale-specific arguments

range of values to include in mapping | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

GENERAL PURPOSE SCALES

Use with most aesthetics

scale_*_continuous() - map cont' values to visual ones
scale_*_discrete() - map discrete values to visual ones
scale_*_identity() - use data values **as** visual ones
scale_*_manual(values = c()**)** - map discrete values to manually chosen visual ones
scale_*_date(date_labels = "%m/%d", date_breaks = "2 weeks"**)** - treat data values as dates.
scale_*_datetime() - treat data x values as date times. Use same arguments as scale_x_date(). See ?strptime for label formats.

SHAPE AND SIZE SCALES

```
p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7))
```

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

```
p + scale_radius(range = c(1,6))
p + scale_size_area(max_size = 6)
```

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)

```
n <- d + geom_bar(aes(fill = fl))
```

```
n + scale_fill_brewer(palette = "Blues")
```

For palette choices:
RColorBrewer::display.brewer.all()

```
n + scale_fill_grey(start = 0.2, end = 0.8,
na.value = "red")
```

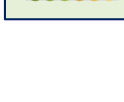
COLOR AND FILL SCALES (CONTINUOUS)

```
o <- c + geom_dotplot(aes(fill = ..x..))
```

```
o + scale_fill_distiller(palette = "Blues")
```

```
o + scale_fill_gradient(low="red", high="yellow")
```

```
o + scale_fill_gradient2(low="red", high="blue",
mid = "white", midpoint = 25)
```



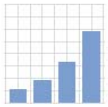
```
o + scale_fill_gradientn(colours=topo.colors(6))
```

Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

ggplot2: Coordinate Systems, Position Adjustments

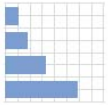
Coordinate Systems

r <- d + geom_bar()



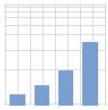
r + coord_cartesian(xlim = c(0, 5)**)**
xlim, ylim
The default cartesian coordinate system

r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units



r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates

r + coord_polar(theta = "x", direction=1 **)**
theta, start, direction
Polar coordinates

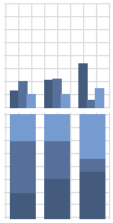


r + coord_trans(ytrans = "sqrt"**)**
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation=c(41, -74, 0)**)**projection, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

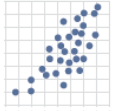
Position adjustments determine how to arrange geoms that would otherwise occupy the same space.



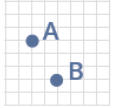
s <- ggplot(mpg, aes(fl, fill = drv))

s + geom_bar(position = "dodge")
Arrange elements side by side

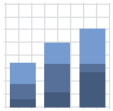
s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height



e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting



e + geom_label(position = "nudge")
Nudge labels away from points



s + geom_bar(position = "stack")
Stack elements on top of one another

Each position adjustment can be recast as a function with manual **width** and **height** arguments
s + geom_bar(position = position_dodge(width = 1))

ggplot2: Labels, Legends, Faceting, Zooming

Labels

t + labs(**x** = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
<AES> = "New <AES> legend title")

Use scale functions to update legend labels

t + annotate(geom = "text", x = 8, y = 9, label = "A")

geom to place

manual values for geom's aesthetics

Legends

n + theme(legend.position = "bottom"**)**
Place legend at "bottom", "top", "left", or "right"

n + guides(fill = "none"**)**
Set legend type for each aesthetic: colorbar, legend, or none (no legend)

n + scale_fill_discrete(name = "Title",
labels = c("A", "B", "C", "D", "E"**)**
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred)

t + coord_cartesian(
xlim = c(0, 100), ylim = c(10, 20)**)**

With clipping (removes unseen data points)

t + xlim(0, 100**) + ylim(**10, 20**)**

t + scale_x_continuous(limits = c(0, 100)**) + scale_y_continuous(**limits = c(0, 100)**)**

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(cols = vars(fl))
facet into columns based on fl

t + facet_grid(rows = vars(year))
facet into rows based on year

t + facet_grid(rows = vars(year), cols = vars(fl))
facet into both rows and columns

t + facet_wrap(vars(fl))
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(rows = vars(drv), cols = vars(fl), scales = "free")
x and y axis limits adjust to individual facets
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(cols = vars(fl), labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(rows = vars(fl), labeller = label_bquote(alpha ^ .(fl)**))**

α^c	α^d	α^e	α^p	α^r