STAT 2005 – Programming Languages for Statistics

Tutorial 3 Statistical Graphics

2020

LIU Ran

Department of Statistics, The Chinese University of Hong Kong

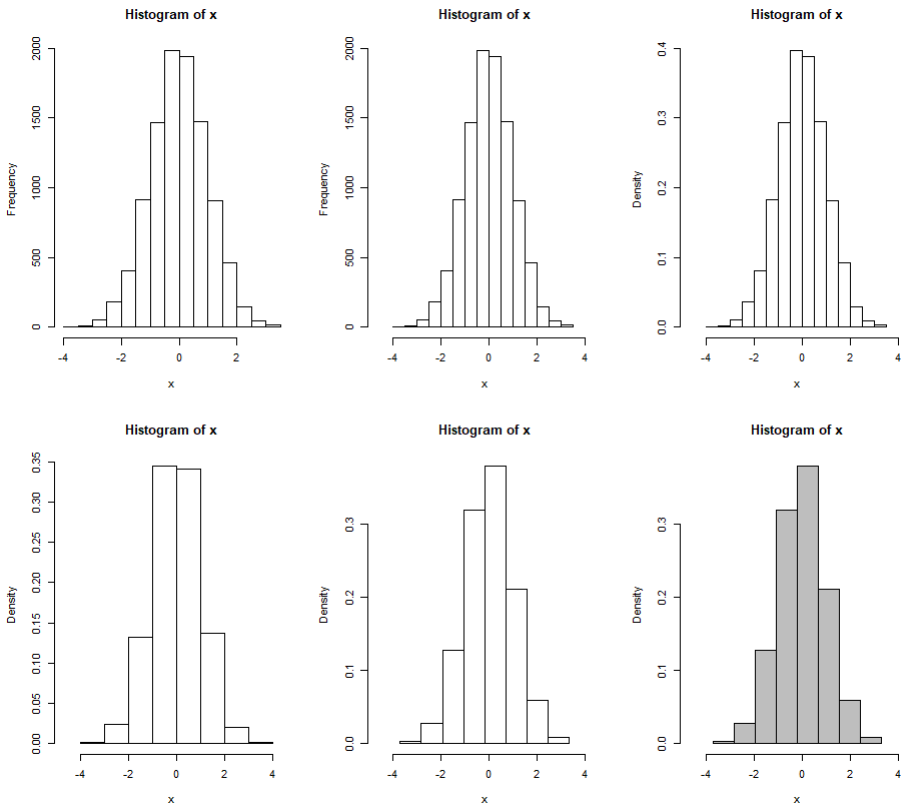
1 Statistical Graphics

We are going to review the concepts about the histogram, qqnorm and boxplot commands. And after that **matplot** command will also be mentioned.

1.1 histogram

A histogram is an approximate representation of the distribution of numerical data. When we want to check the density function of a dataset, it is helpful to draw a histogram.

```
set.seed(2005)
x = rnorm(10000)
par(mfrow=c(2,3))
hist(x)
hist(x,xlim = c(min(x)-1,max(x)+1))
# xlim means the value range of xaxis
hist(x,xlim = c(min(x)-1,max(x)+1), freq = F)
# freq represents the type for yaxis(density or frequency)
hist(x,xlim = c(min(x)-1,max(x)+1), freq = F, nclass = 10)
# nclass means the number of bins
hist(x,xlim = c(min(x)-1,max(x)+1), freq = F, breaks = seq(min(x),max(x),length.out = 9))
# You can also use 'breaks' argument to divide the dataset
hist(x,xlim = c(min(x)-1,max(x)+1), freq = F, breaks = seq(min(x),max(x),length.out = 9),
col='grey')
```



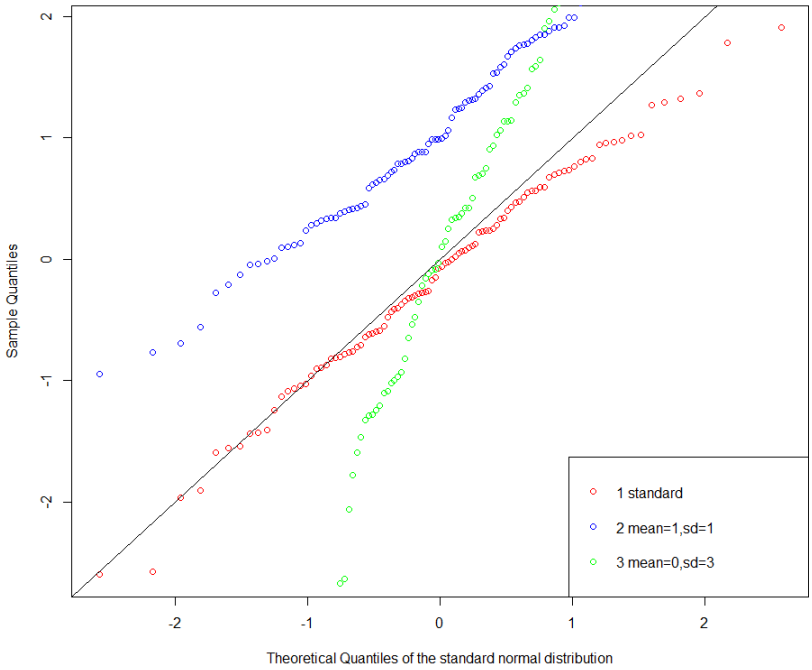
Remark 1.1. **freq** if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, density values, are plotted.

Remark 1.2. The density value 0.4 may not be the proportion of this bin, which means $2000/10000 \neq 0.4$. Because $\int_a^b f(x)dx = P(X \in [a,b])$, the density value multiplied by the width of this bin roughly equals to the proportion $0.4 * width \approx \frac{2000}{10000}$.

1.2 qqnorm

A Q–Q (quantile-quantile) plot is a probability plot, which is a graphical method for comparing two probability distributions by plotting their quantiles against each other. **qqnorm** compares the data with the standard normal distribution.

```
set.seed(2005)
y1 = rnorm(100,mean = 0,sd = 1)
y2 = rnorm(100,mean = 1,sd = 1)
y3 = rnorm(100,mean = 0,sd = 3)
x1 = qqnorm(y1,plot.it = F) # not plot it individually
x2 = qqnorm(y2,plot.it = F)
x3 = qqnorm(y3,plot.it = F)
plot(x1,col = 'red',
     xlab = 'Theoretical Quantiles of the standard normal distribution',
     ylab = 'Sample Quantiles')
points(x2,col = 'blue')
points(x3,col = 'green')
abline(a=0,b=1) # y=x
# If your data satisfy the defined distribution, the fitted line will be y = x
legend('bottomright',legend = paste(1:3,c('standard','mean=1,sd=1','mean=0,sd=3')),
       pch = 1,col = c('red','blue','green'))
```



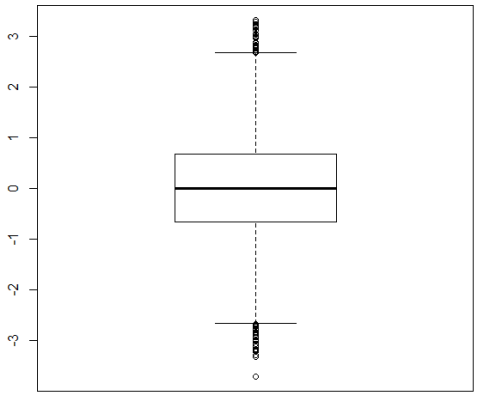
Remark 1.3. The qqnorm function will give the standard normal distribution quantiles. If the data satisfy the standard normal distribution, points generated by **qqnorm** will be around the line $y = x$. If data satisfy a normal distribution but it has not standard mean or variance, the slope and the intercept will be influenced. And from the figure we can see the variance affects the slope and the mean affects the intercept. But no matter what mean and variance are, as long as your data is generated by a normal distribution, it will still be a straight line. Therefore, it is appropriate to use **qqnorm** to see if the data satisfy a normal distribution.

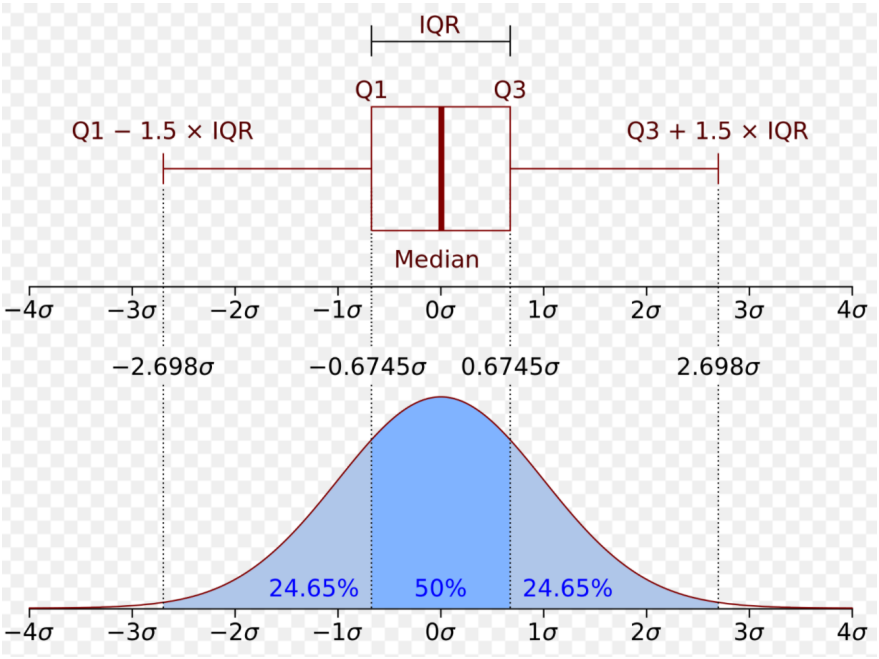
1.3 boxplot

A box plot is a method for graphically depicting groups of numerical data through their quartile . A quartile is a type of quantile which divides the number of data points into four parts.

1. Median (Q2 / 50th percentile)
2. First quartile (Q1 / 25th percentile)
3. Third quartile (Q3 / 75th percentile)
4. Interquartile range (IQR) = Q3 - Q1

```r
boxplot(x) # Box Plot
boxplot(x)$out # the outliers
index = which(x %in% boxplot(x)$out) # find the index of outliers
```





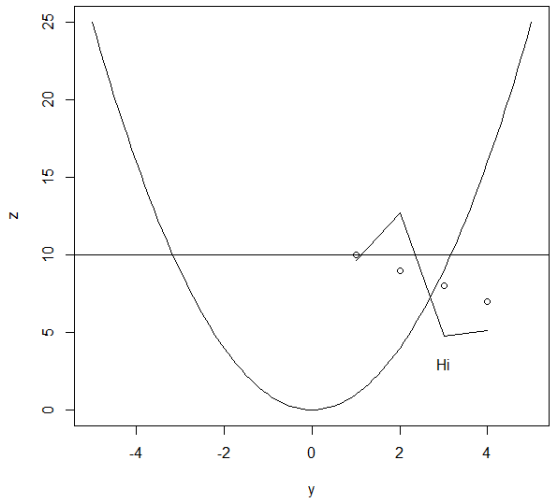
Remark 1.4. Although the histogram seems to be more intuitive, the box plot needs much less space and you can draw different groups' box plots in one figure. Meanwhile, it does not need to choose the number of bins or the width between breaks, which heavily affect the appearance of a histogram.

1.4 Add Elements To Existing Graphs

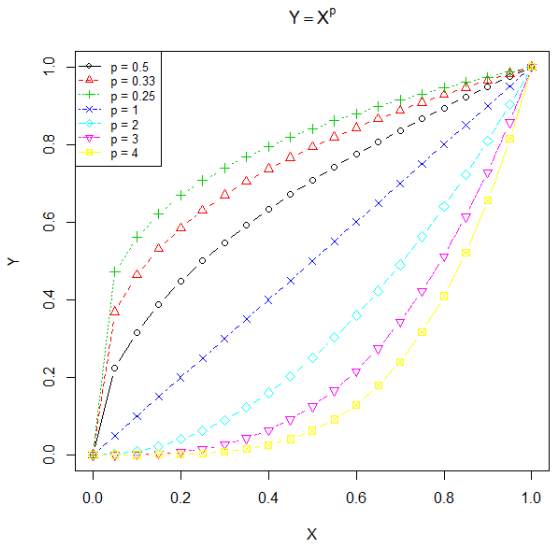
Function	Description
`abline(a,b)`	A line with intercept `a` and slope `b`
`points(x,y)`	Points `(x,y)`
`lines(x,y)`	Several line segments pass through points `(x,y)`
`text(x,y,text)`	Text at `(x,y)`

```
set.seed(2005)
y = seq(-5,5,length.out=10)
z = y**2
plot(y, z, type="l")
abline(h=10) # add a horizontal line
# use v = 0 to draw a vertical line
points(1:4, 10:7) # Add points
# x coordinates c(1,2,3,4)
# y coordinates c(10,9,8,7)
# point coordinates (1,10), (2,9), (3,8), (4,7)
lines(1:4, 10*abs(rnorm(4))) # Add three line segments
text(3, 3, "Hi") # Text
```

Add Elements



matplot



2 matplot

- **matplot**: Plot several lines in one graph at the same time.

```
# plot the figure for f(x) = x^p
x = seq(0, 1, by=0.05)
n = 4
p = c(1/(2:n), 1:n)
N = length(p)
X = matrix(rep(x, N), ncol=N) # each column of X is the vector x
Y = t(t(X)^p) # Y[,j] = X[,j]^p[j] calculate the function values by columns
col = pch = lty = 1:N
matplot(X, Y, main=expression(Y==X^p), pch=pch, col=col, lty=lty, type="b")
# We can use the expression command to show the mathematical formula
legend("topleft", legend = paste0("p = ", round(p, 2)),
       pch=pch, col=col, lty=lty, cex=0.8)
# round(x,4): rounds the values in its first argument to the specified number of decimal places
```

```
> p
[1] 0.5000000 0.3333333 0.2500000 1.0000000 2.0000000 3.0000000 4.0000000
```

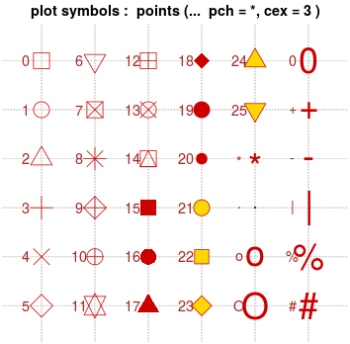
```
> X
      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
 [1,] 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 [2,] 0.05 0.05 0.05 0.05 0.05 0.05 0.05
 [3,] 0.10 0.10 0.10 0.10 0.10 0.10 0.10
 [4,] 0.15 0.15 0.15 0.15 0.15 0.15 0.15
 [5,] 0.20 0.20 0.20 0.20 0.20 0.20 0.20
 [6,] 0.25 0.25 0.25 0.25 0.25 0.25 0.25
 [7,] 0.30 0.30 0.30 0.30 0.30 0.30 0.30
 [8,] 0.35 0.35 0.35 0.35 0.35 0.35 0.35
 [9,] 0.40 0.40 0.40 0.40 0.40 0.40 0.40
[10,] 0.45 0.45 0.45 0.45 0.45 0.45 0.45
[11,] 0.50 0.50 0.50 0.50 0.50 0.50 0.50
[12,] 0.55 0.55 0.55 0.55 0.55 0.55 0.55
[13,] 0.60 0.60 0.60 0.60 0.60 0.60 0.60
[14,] 0.65 0.65 0.65 0.65 0.65 0.65 0.65
[15,] 0.70 0.70 0.70 0.70 0.70 0.70 0.70
[16,] 0.75 0.75 0.75 0.75 0.75 0.75 0.75
[17,] 0.80 0.80 0.80 0.80 0.80 0.80 0.80
[18,] 0.85 0.85 0.85 0.85 0.85 0.85 0.85
[19,] 0.90 0.90 0.90 0.90 0.90 0.90 0.90
[20,] 0.95 0.95 0.95 0.95 0.95 0.95 0.95
[21,] 1.00 1.00 1.00 1.00 1.00 1.00 1.00
> Y
            [,1]       [,2]       [,3] [,4]   [,5]     [,6]       [,7]
 [1,] 0.0000000 0.0000000 0.0000000 0.00 0.0000 0.000000 0.00000000
 [2,] 0.2236068 0.3684031 0.4728708 0.05 0.0025 0.000125 0.00000625
 [3,] 0.3162278 0.4641589 0.5623413 0.10 0.0100 0.001000 0.00010000
 [4,] 0.3872983 0.5313293 0.6223330 0.15 0.0225 0.003375 0.00050625
 [5,] 0.4472136 0.5848035 0.6687403 0.20 0.0400 0.008000 0.00160000
 [6,] 0.5000000 0.6299605 0.7071068 0.25 0.0625 0.015625 0.00390625
 [7,] 0.5477226 0.6694330 0.7400828 0.30 0.0900 0.027000 0.00810000
 [8,] 0.5916080 0.7047299 0.7691606 0.35 0.1225 0.042875 0.01500625
 [9,] 0.6324555 0.7368063 0.7952707 0.40 0.1600 0.064000 0.02560000
[10,] 0.6708204 0.7663094 0.8190363 0.45 0.2025 0.091125 0.04100625
[11,] 0.7071068 0.7937005 0.8408964 0.50 0.2500 0.125000 0.06250000
[12,] 0.7416198 0.8193213 0.8611735 0.55 0.3025 0.166375 0.09150625
[13,] 0.7745967 0.8434327 0.8801117 0.60 0.3600 0.216000 0.12960000
[14,] 0.8062258 0.8662391 0.8979008 0.65 0.4225 0.274625 0.17850625
[15,] 0.8366600 0.8879040 0.9146912 0.70 0.4900 0.343000 0.24010000
[16,] 0.8660254 0.9085603 0.9306049 0.75 0.5625 0.421875 0.31640625
[17,] 0.8944272 0.9283178 0.9457416 0.80 0.6400 0.512000 0.40960000
[18,] 0.9219544 0.9472682 0.9601846 0.85 0.7225 0.614125 0.52200625
[19,] 0.9486833 0.9654894 0.9740037 0.90 0.8100 0.729000 0.65610000
[20,] 0.9746794 0.9830476 0.9872585 0.95 0.9025 0.857375 0.81450625
[21,] 1.0000000 1.0000000 1.0000000 1.00 1.0000 1.000000 1.00000000
```

Remark 2.1. Suitable legend should be added for understanding and interpretation.

3 Useful things

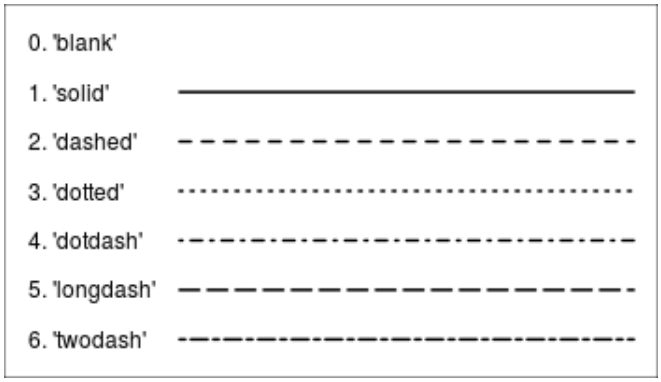
3.1 pch

Use the pch= option to specify symbols to use when plotting points. For symbols 21 through 25, specify border color (**col=**) and fill color (**bg=**).



3.2 lty

You can use **lty=** to specify the type of lines. You can choose the type you want by the integer index or the character.



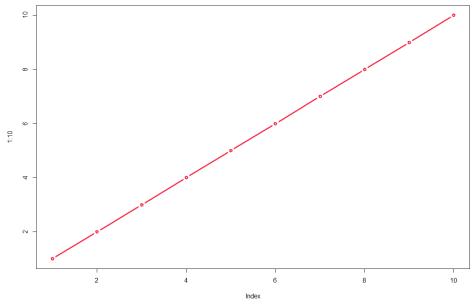
3.3 Color

You can specify colors in R by index, name, hexadecimal, or RGB. For example col=1, col="white", and col="#FFFFFF" are equivalent. Because the colors in R are too many to choose, sometimes we select the optimal color from the Crayola Crayon colors(optional). The following are the Crayola Crayon colors:

Because the names in the table do not have the same names in R, for example, you cannot use **col='Scarlet'** directly. You should load the package 'broman' to get the corresponding hexadecimal of the color you want.

```
library(broman) # contains the crayons color
plot_crayons() # display the above figure
colors = brocolors("crayons") # return a list which contains all the colors by their hexadecimal
sel_color = colors['Scarlet'] # choose the color you want to plot
plot(1:10,type='b',lwd = 3,col = sel_color) # use the color
```

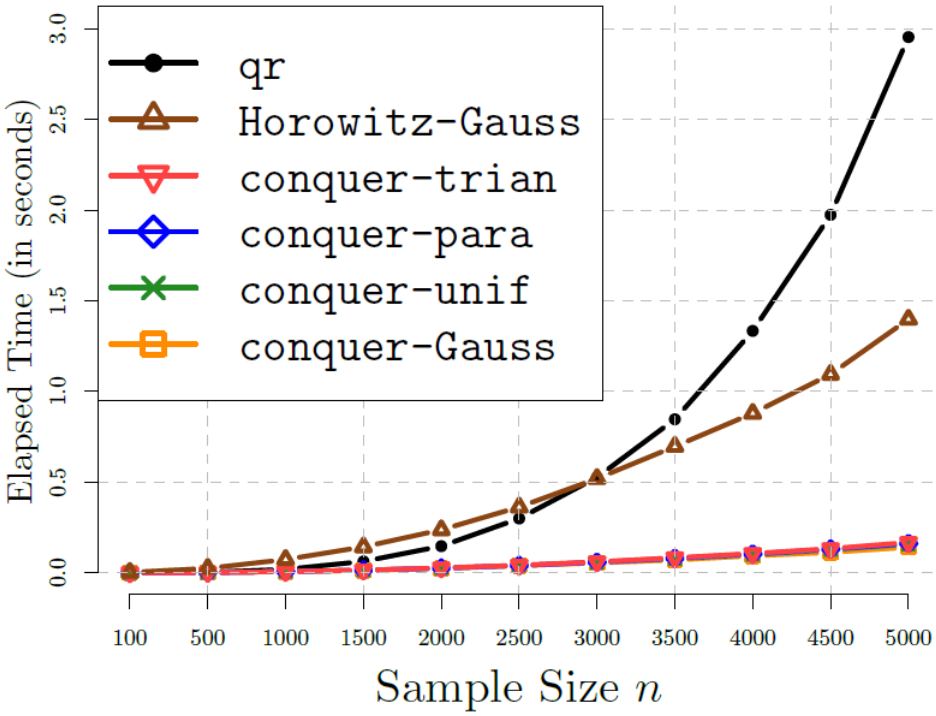
Remark 3.1. Go to wiki to check more detail about Crayola Crayon colors.



4 Example(Optional)

How to draw a nice figure:

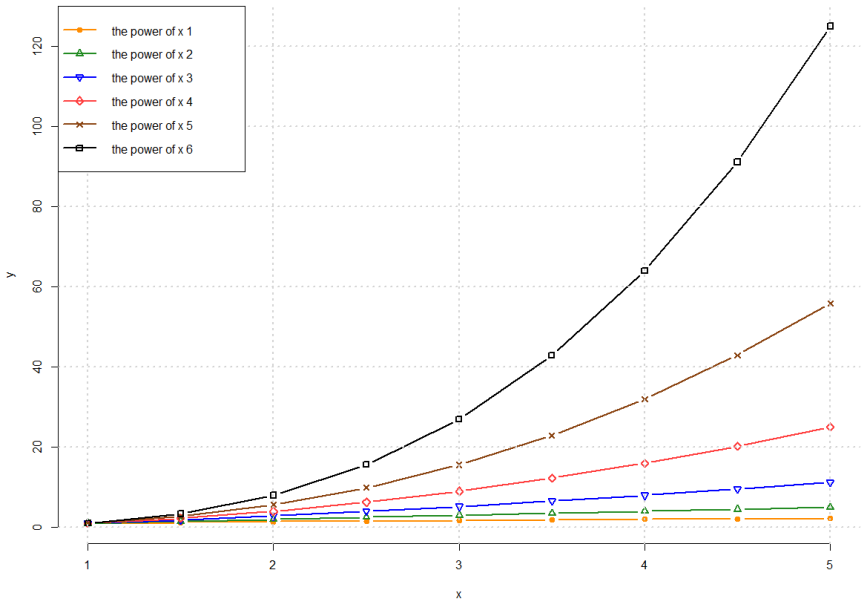
1. Be familiar with commands for drawing, such as the parameter commands for plot(base). Try to know the effect of each command.
2. Watch many wonderful figures in others' paper(Nature, Science...), and try to find the advantages of their setting.
3. Try to reproduce the figures by yourself.
4. Create your own setting for next time use.



1. Generate similar data, here we use $f(x) = x^{p/2}$ power function to generate similar data.
2. Notice the **par()** setting, the isolated axis means we should use **bty = 'n'**.
3. Use the same **pch** and **color** parameters.
4. Use the same type of lines and proper line width, here we use the solid type and **lwd = 2**.
5. Notice the grid setting, here we use **grid(lwd=2)**.
6. Make a nice legend.
7. Detail adjustment.(lab title, axis range, main title)

Remark 4.1. Go to related websites to pick the colors you want from images: imagecolorpic.

```r
par(bty = 'n')
index = 1:6
x_range = seq(1,5,by=0.5)
sines <- outer(x_range, index, function(x, p) x^(p/2))
# generate a matrix: sines[i,j] = func(x_range[i],index[j])
matplot(x_range,sines)
my_shape = as.integer(c('19','2','6','5','4','0'))
# black, brown, red, blue, green, yellow
# many similar colors, col = 'green' is not the one we want
my_color = c('#000000','#8B4513','#FF4040','#0000FF','#228B22','#FF8C00')
my_color = my_color[seq(length(my_color),1)]
matplot(x_range,sines, type = "b",
        pch = my_shape[index],
        lty = 1,
        col = my_color[index],
        lwd = 2,
        xlab = 'x',
        ylab = 'y',
        xlim = c(min(x_range),max(x_range)),
        ylim = c(min(sines),max(sines)),
        )
grid(lwd=2)
legend('topleft',legend = paste('the power of x',index),
       col = my_color[index],
       pch = my_shape[index],
       lty = 1,
       lwd = 2)
```



After reproducing others' figures, you can store the setting, such as **pch** and **color**, and next time, use these settings to plot your own figures.