# Basic Data Visualization

STAT2005

Chapter 3

# Data Visualization

- R can produce professional statistical graphics easily. It has extensive graphical procedures with many options.
- Note that some options in R's graphics are very complicated and involved. Here we only discuss some simple and commonly used options.
- In this chapter, we will introduce how to produce
  - Pie chart
  - Histogram
  - Quantile-quantile plot
  - Box plot
  - Scatter plot
  - Matrix scatter plot
  - Time series plot
  - Mathematical function plot

# High level plots

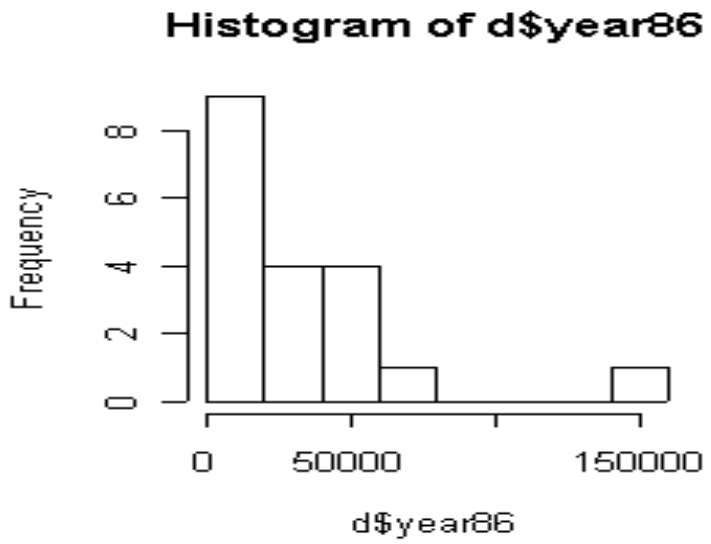
- The functions to draw these plots in R are called "high level" because you don’t need to worry about the details of where the ink goes; you just describe the plot you want, and R does the drawing.
- All the graphics in R are produced in separate graphic windows.
- These windows as well as the graphic can be resized, copy and paste in the usual way.
- By using copy and paste, we can easily include any of the R’s output to our report.

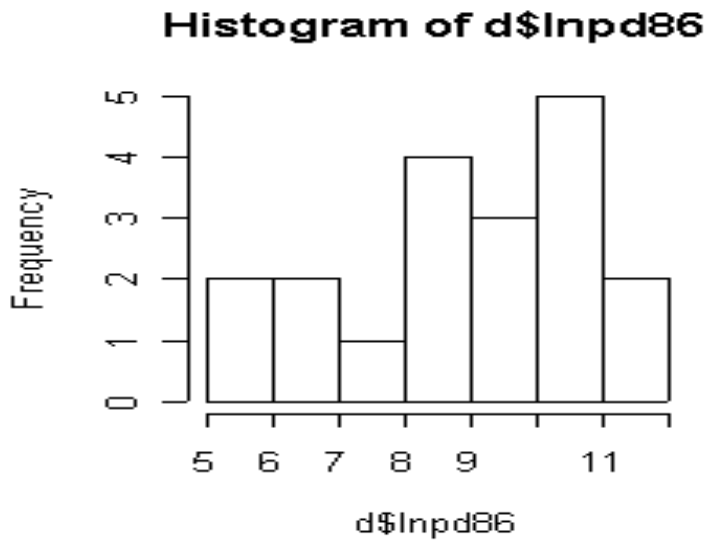
# Histogram

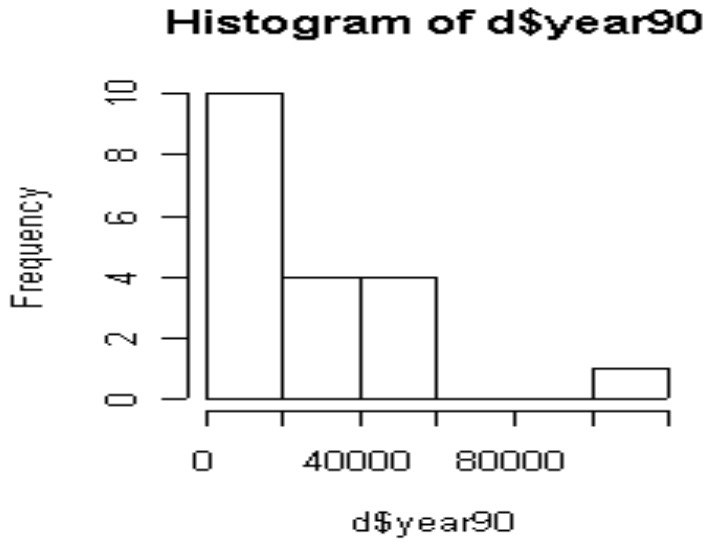
- Histogram is one of the most commonly used graphics for displaying the distribution of the data.
- In many textbooks and articles, we can see many graphics are produced within a single frame layout for aesthetic reason.
- R can produce several graphics inside a frame using the function `par()` with `mfrow` option. `par` stands for graphical parameters and `mfrow` stands for multi-frame filled in row-wise.
- There is also an option `mfcol` which stands for multi-frame filled in column-wise. We can type `help(par)` to look at all the options in `par()`.
- In the following example, we use the option `mfrow=c(2,2)`. This will set up a 2-by-2 multi-frame for graphics produced to be filled in row-wise.

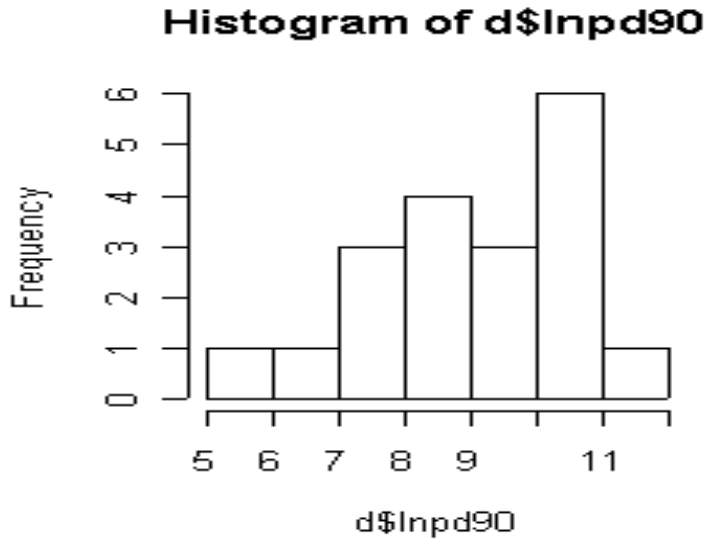
```
> d<-read.csv("popden1.csv", stringsAsFactors=T)
# read in data
> names(d)
# display names in d
[1] "district" "year86"   "year90"   "Region"
"lnpd86"   "lnpd90"
> par(mfrow=c(2,2))
# set up a 2x2 multi-frame for graphics
> hist(d$year86)
# produce 4 histograms filled in row by row
> hist(d$lnpd86)
> hist(d$year90)
> hist(d$lnpd90)
```

| district | year86 | year90 | Region | lnpd86 | lnpd90 |
|---|---|---|---|---|---|
| Islands | 290 | 293 | NT | 5.669881 | 5.680173 |
| Sai_Kung | 365 | 1026 | NT | 5.899897 | 6.933423 |
| Tai_Po | 1033 | 1496 | NT | 6.940222 | 7.31055 |
| North | 1074 | 1211 | NT | 6.979145 | 7.099202 |
| Yuen_Long | 1545 | 1664 | NT | 7.342779 | 7.41698 |
| Tuen_Mun | 3611 | 4711 | NT | 8.19174 | 8.457655 |
| Tsuen_Wan | 4159 | 4581 | NT | 8.33303 | 8.429673 |
| Sha_Tin | 5402 | 7378 | NT | 8.594525 | 8.906258 |
| Southern | 6380 | 6701 | HK | 8.760923 | 8.810012 |
| Wan_Chai | 20182 | 18209 | HK | 9.912546 | 9.809671 |
| Central/West | 20854 | 20479 | HK | 9.945301 | 9.927155 |
| Kwai_Tsing | 21464 | 21158 | KL | 9.974132 | 9.959773 |
| Eastern | 27387 | 30316 | KL | 10.21782 | 10.31943 |
| Yau_Tsim | 45355 | 33232 | KL | 10.72228 | 10.41127 |
| Wong_Tai_Sin | 46940 | 41331 | KL | 10.75663 | 10.62937 |
| Kowloon_City | 47156 | 41759 | KL | 10.76122 | 10.63967 |
| Sham_Shui_Po | 56875 | 48822 | KL | 10.94861 | 10.79594 |
| Kwun_Tong | 60826 | 52562 | KL | 11.01577 | 10.86975 |
| Mong_Kok | 142718 | 116531 | KL | 11.86863 | 11.66591 |


Histogram of d$year86
Frequency
0
2
4
6
8
0
50000
150000
d$year86

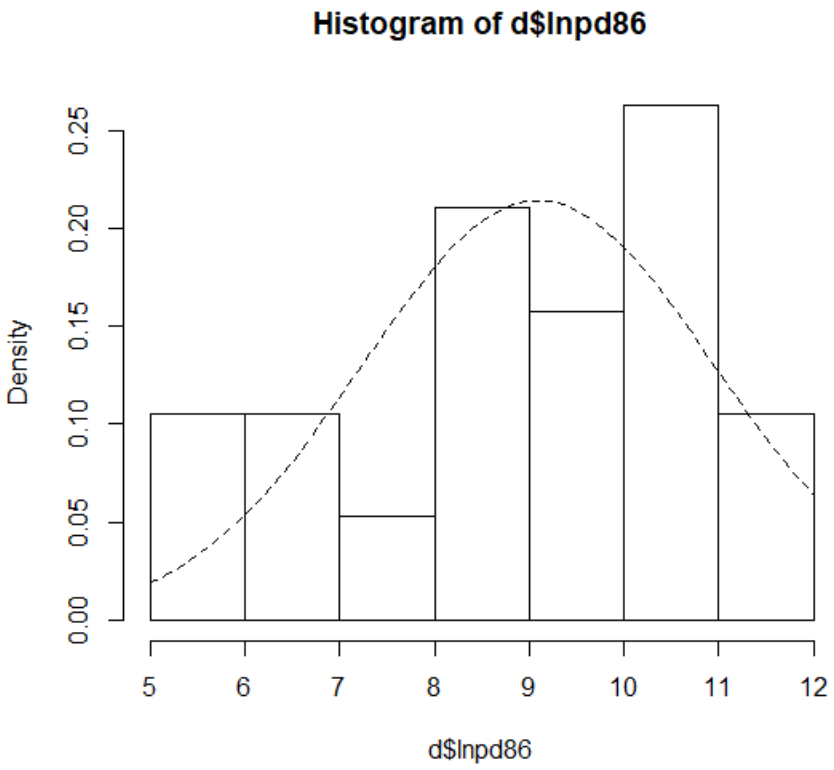

Histogram of d$lnpd86
Frequency
0
1
2
3
4
5
5
6
7
8
9
11
d$lnpd86

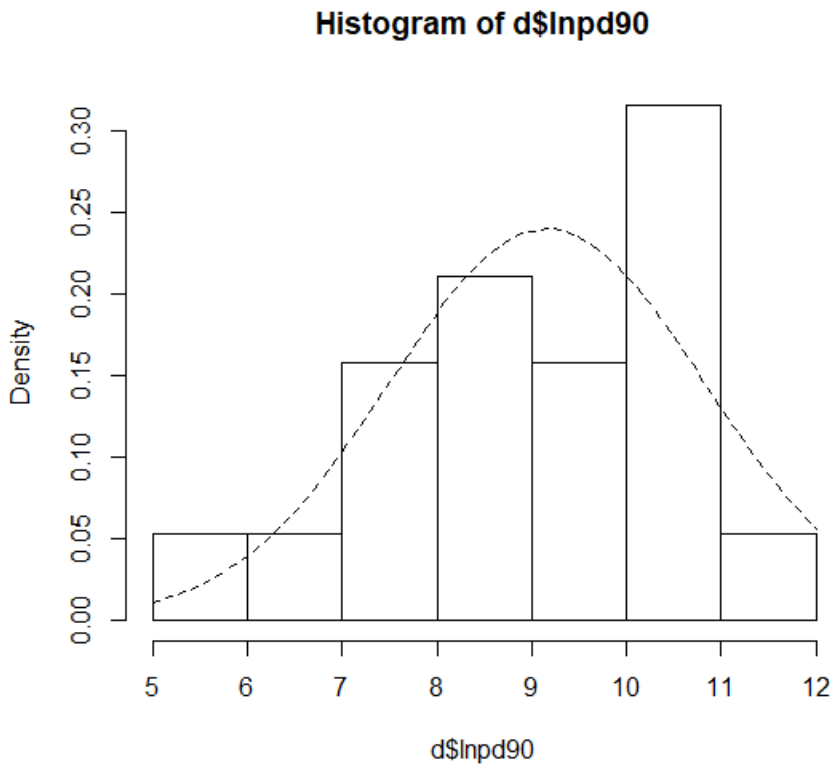

Histogram of d$year90
Frequency
0
2
4
6
8
10
0
40000
80000
d$year90


Histogram of d$lnpd90
Frequency
0
1
2
3
4
5
6
5
6
7
8
9
11
d$lnpd90

- From the histograms, the original variables are highly skewed to the right and the log-transformation reduces the skewness.
- In many statistical techniques, we usually assume that the data are normally distributed. The following commands add the normal density to the histograms of `lnpd86` and `lnpd90` for reference.

```
> par(mfrow=c(1,2))
# set up a 1 by 2 multi-frame
> hist(d$lnpd86,freq=F)
# histogram with density instead of frequency
> x<-seq(5,12,0.1)          # set up sequence for x
> lines(x,dnorm(x,mean(d$lnpd86),sd(d$lnpd86)),lty=2)
# add density lines
> hist(d$lnpd90,freq=F)  # repeat for lnpd90
> lines(x,dnorm(x,mean(d$lnpd90),sd(d$lnpd90)),lty=2)
```

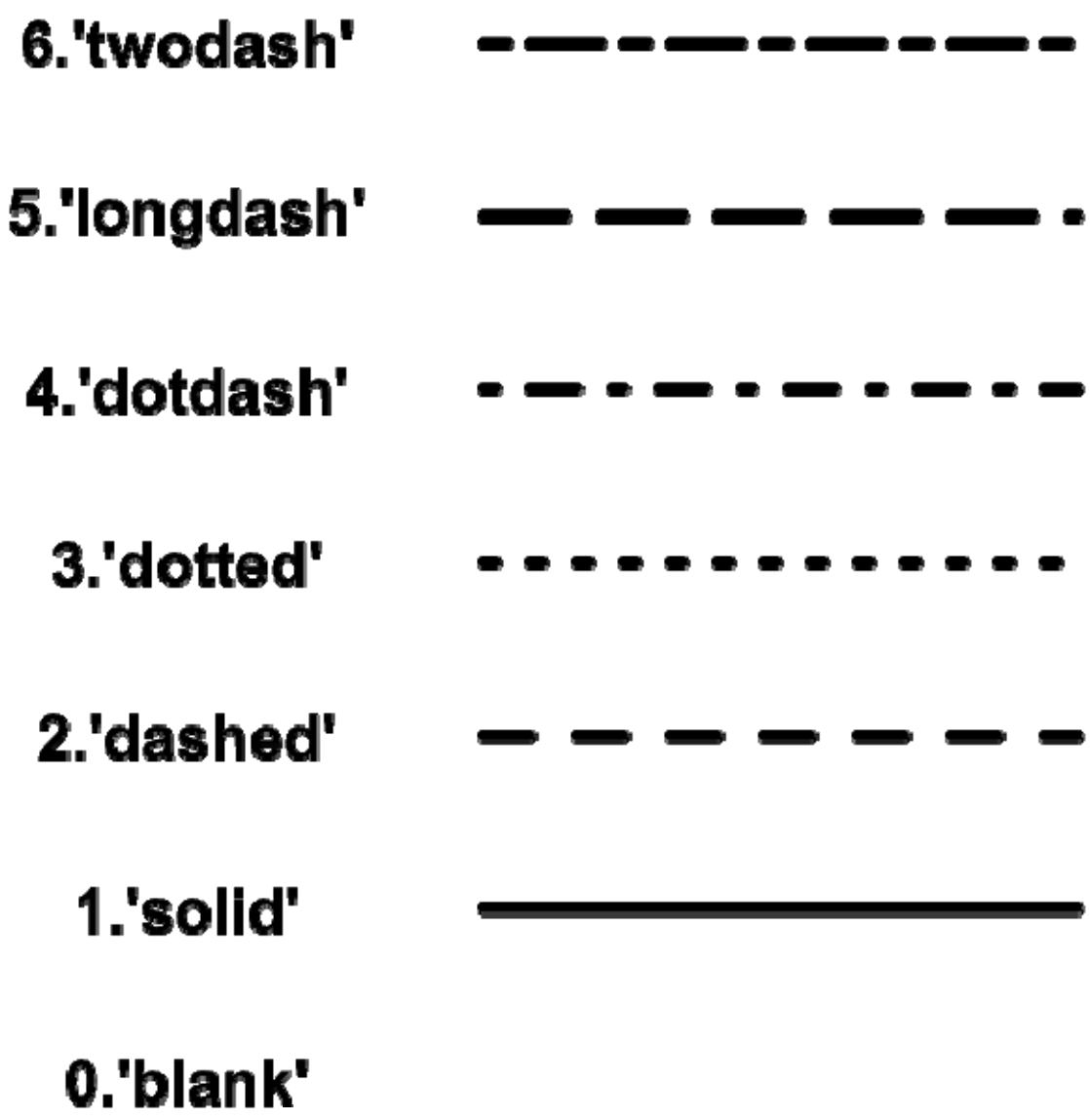

Histogram of d$lnpd86
Density
0.00
0.05
0.10
0.15
0.20
0.25
5
6
7
8
9
10
11
12
d$lnpd86


Histogram of d$lnpd90
Density
0.00
0.05
0.10
0.15
0.20
0.25
0.30
5
6
7
8
9
10
11
12
d$lnpd90

`x<-seq(5,12,0.1)` generates a sequence from 5 to 12 with an increment 0.1 and `dnorm(x,m,s)` gives the normal density with mean `m` and standard deviation `s`. Note that the option `freq=F` in `hist()` produces the histogram in density rather than frequency.

```
lines(x,dnorm(x,mean(d$lnpd90),sd(d$lnpd90)),lty=2)
```

adds the density curve to the histogram using dashed line (`lty=2`).


6.'twodash'
5.'longdash'
4.'dotdash'
3.'dotted'
2.'dashed'
1.'solid'
0.'blank'

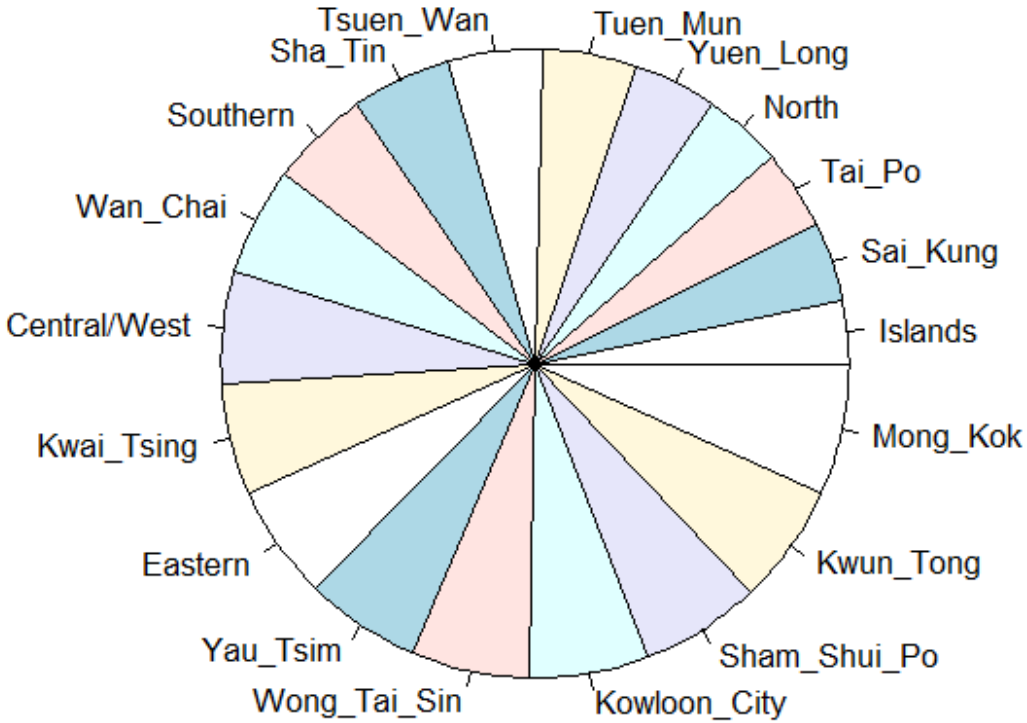
# Pie chart

Pie charts display a vector of numbers by breaking up a circular disk into pieces whose angle (and hence area) is proportional to each number.

The following produces a pie chart with `district` as labels.

```
pie(d$lnpd90,labels=d$district, cex=0.8,
main="log(density) HK1990")
```

`cex` is the character expansion factor. It indicates the amount by which the text and symbols should be scaled relative to the default size. 1 is default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.




It is rather difficult to compare the proportion of `log(density)` in various district. Pie chart is not an effective way to display this information.

# Bar chart

Bar chart is more effective to display the information when the proportion of each category is close to each other.

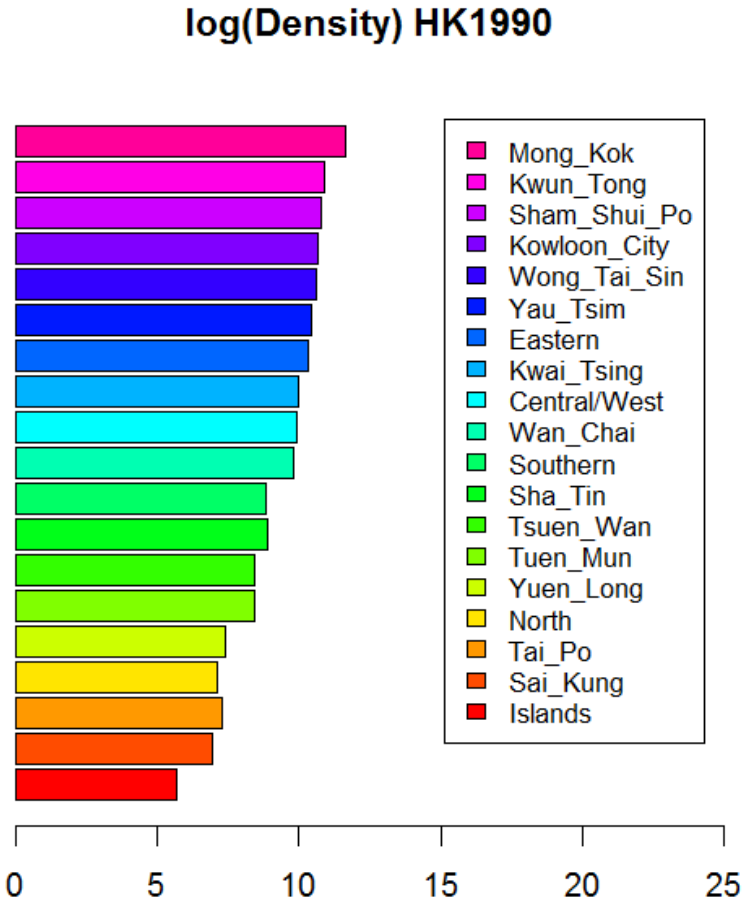
We produce two bar charts, one with colour and legend; the other is black and white, with the legend text lies inside the bar and the numbers appear at the end.

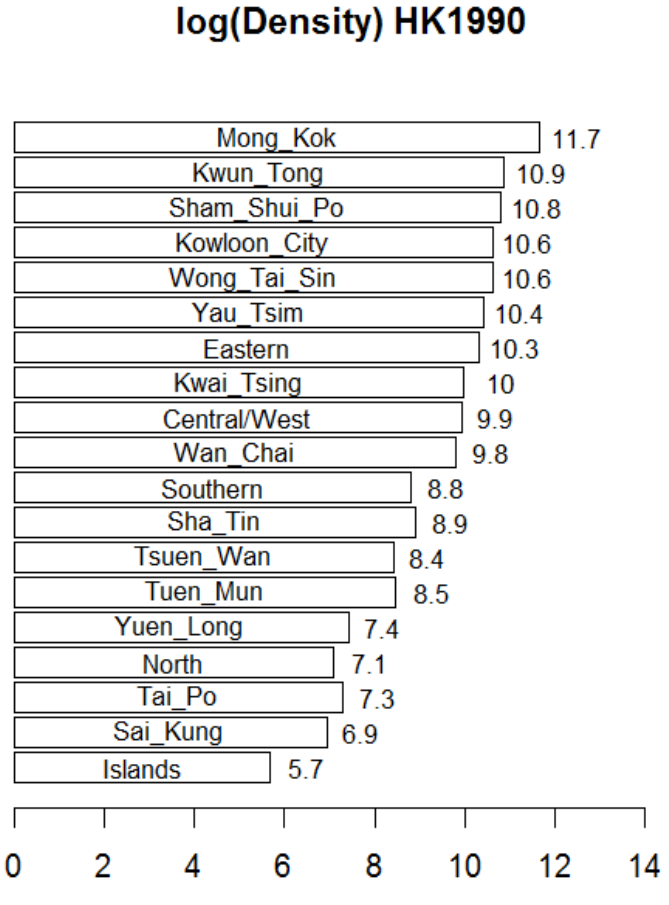
```
# bar chart with color
barplot(d$lnpd90, horiz=T, col=rainbow(20),
          xlim=c(0,25), legend.text=d$district,
          args.legend=list(x=25, y=23, cex=0.8),
          main="log(Density) HK1990"
)
```

The option `horiz=T` produces a horizontal bar. The option `col=rainbow(20)` use rainbow colour scheme up to 20 colours, `args.legend` is a list for the legend.

```
# bar chart with legend inside and number at the end
y<-barplot(d$lnpd90,horiz=T,col="white",xlim=c(0,15),
main="log(Density) HK1990")
x<-round(d$lnpd90,1)
# obtain the x-coord. and round to 1 decimal
text(0.5*x,y,d$district,cex=0.8)
# add legend inside the bar
text(x+0.8,y,labels=x,cex=0.8)
# add number at the end
```

The y-coordinate of bars are saved in `y`. The value `d$lnpd90` is rounded to one decimal place and saved in `x`. The legend `d$district` is printed at `(0.5*x, y)` with `cex=0.8`. Finally `x` is printed at `(x+0.8, y)`.


log(Density) HK1990
Mong_Kok
Kwun_Tong
Sham_Shui_Po
Kowloon_City
Wong_Tai_Sin
Yau_Tsim
Eastern
Kwai_Tsing
Central/West
Wan_Chai
Southern
Sha_Tin
Tsuen_Wan
Tuen_Mun
Yuen_Long
North
Tai_Po
Sai_Kung
Islands
0
5
10
15
20
25


log(Density) HK1990
Mong_Kok 11.7
Kwun_Tong 10.9
Sham_Shui_Po 10.8
Kowloon_City 10.6
Wong_Tai_Sin 10.6
Yau_Tsim 10.4
Eastern 10.3
Kwai_Tsing 10
Central/West 9.9
Wan_Chai 9.8
Southern 8.8
Sha_Tin 8.9
Tsuen_Wan 8.4
Tuen_Mun 8.5
Yuen_Long 7.4
North 7.1
Tai_Po 7.3
Sai_Kung 6.9
Islands 5.7
0
2
4
6
8
10
12
14

# Grouped bar chart

We would like to draw a grouped bar chart of `lnpd86` and `lnpd90` for each district: `HK`, `KL` and `NT`. The first step is to compute the sum for each group.

```
yr86<-by(d$lnpd86,d$Region,sum)
# sum lnpd86 by Region
yr90<-by(d$lnpd90,d$Region,sum)
# sum lnpd90 by Region
rs<-cbind(yr86,yr90)
# form 3x2 matrix rs
rs
       yr86     yr90
HK 28.61877 28.54684
KL 86.26508 85.29111
NT 57.95122 60.23391
```

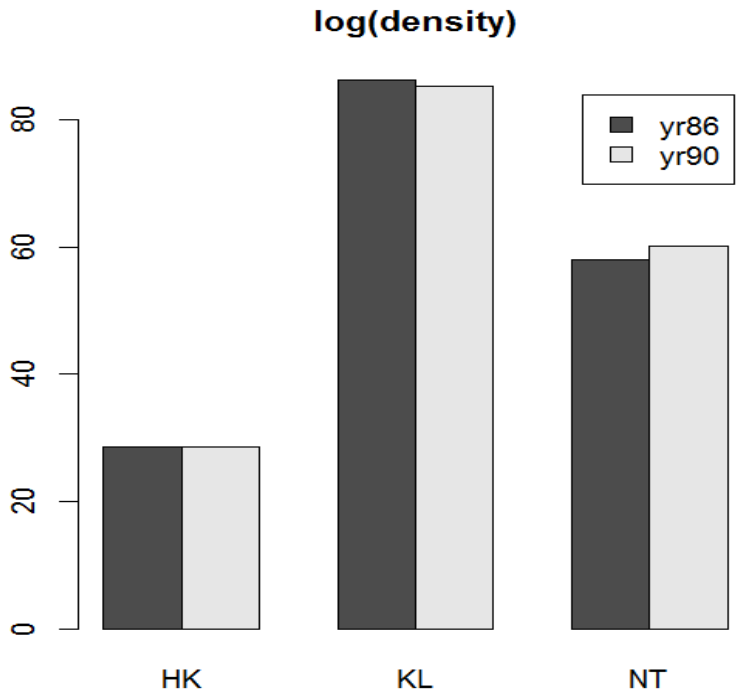
To produce a vertical grouped bar chart:

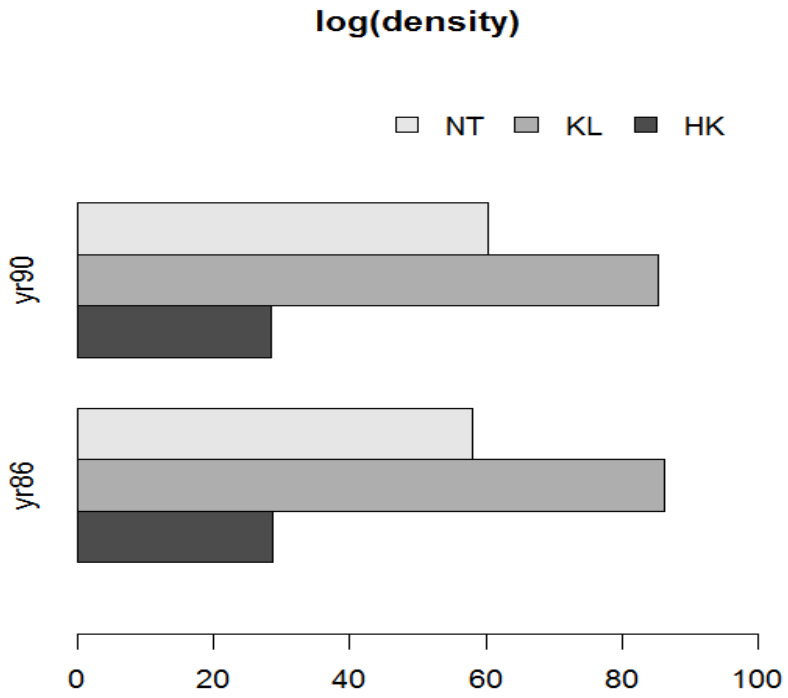
```
# vertical
barplot(t(rs),beside=T,horiz=F,
legend.text=c("yr86","yr90"),main="log(density)")
```

The option `beside=T` sets the grouped bar to be side-by-side instead of stacked.

The next one is a horizontal grouped bar chart with legend placed on the top horizontally. `args.legend` is a list, `horiz=T` means the legend is placed horizontally and `bty="n"` sets the legend box to have no border.

```
# horizontal with legend without border
barplot(rs,beside=T, horiz=T, xlim=c(0,100),
ylim=c(0,10), legend.text=c("HK","KL","NT"),
args.legend=list(horiz=T,bty="n"),main="log(density)")
```


log(density)
yr86
yr90
0
20
40
60
80
HK
KL
NT


log(density)
NT
KL
HK
yr90
yr86
0
20
40
60
80
100

We may produce a bar chart with proportion of each category instead. First the built-in function `prop.table()` returns the row proportion or column proportion of a matrix.

```
> prop.table(t(rs),margin=1)
# margin index=1 means row
             HK        KL        NT
yr86 0.1655843 0.4991179 0.3352978
yr90 0.1639946 0.4899764 0.3460290
```

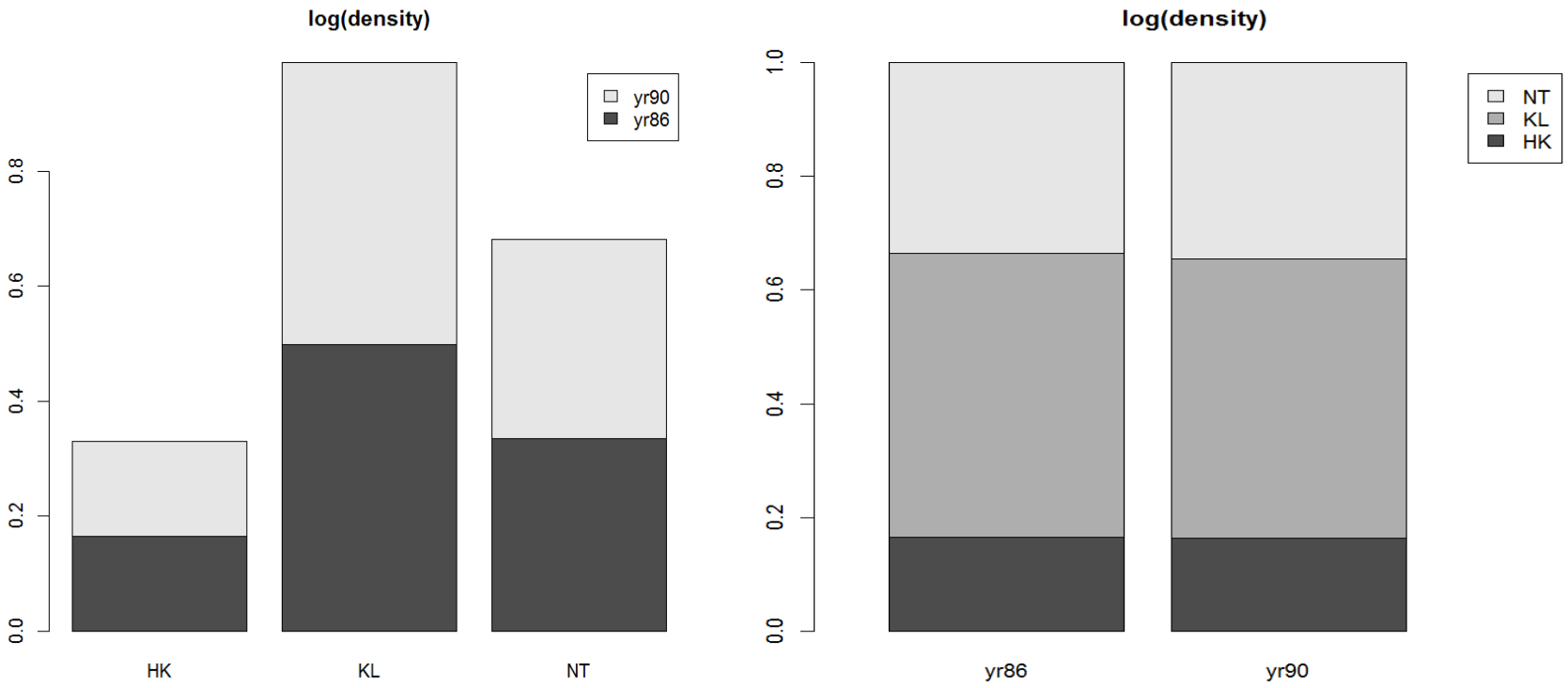
Then we can produce the `barplot` using:

```
> barplot(prop.table(t(rs),1),horiz=F,
legend.text=c("yr86","yr90"), main="log(density)")
```

Similarly,

```
> prop.table(rs,margin=2)
# margin index=2 means column
         yr86      yr90
HK 0.1655843 0.1639946
KL 0.4991179 0.4899764
NT 0.3352978 0.3460290
```

```
> barplot(prop.table(rs,2), horiz=F, xlim=c(0,3),
legend.text=c("HK","KL","NT"), main="log(density)")
```

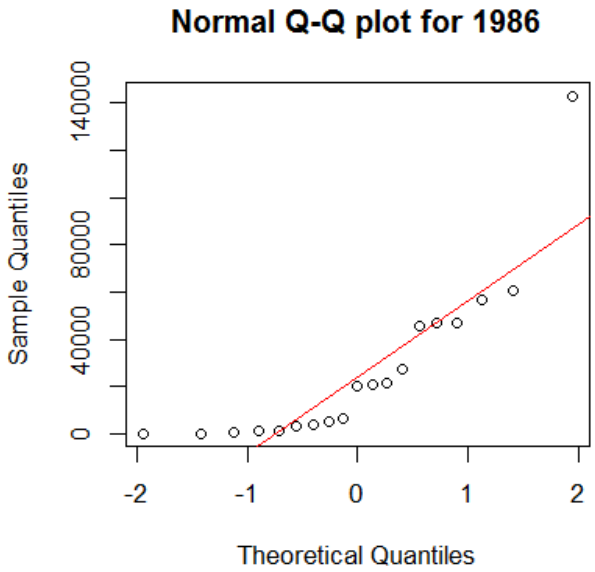

log(density)
yr90
yr86
0.0
0.2
0.4
0.6
0.8
HK
KL
NT
log(density)
NT
KL
HK
0.0
0.2
0.4
0.6
0.8
1.0
yr86
yr90

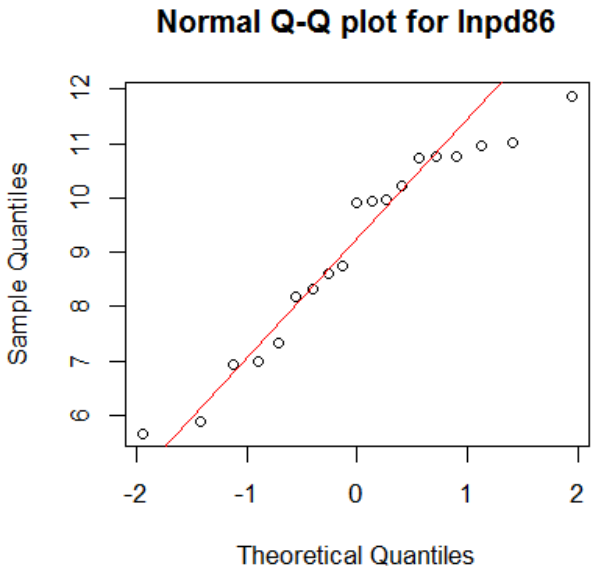
# Normal quantile-quantile plot (Normal QQ plot)

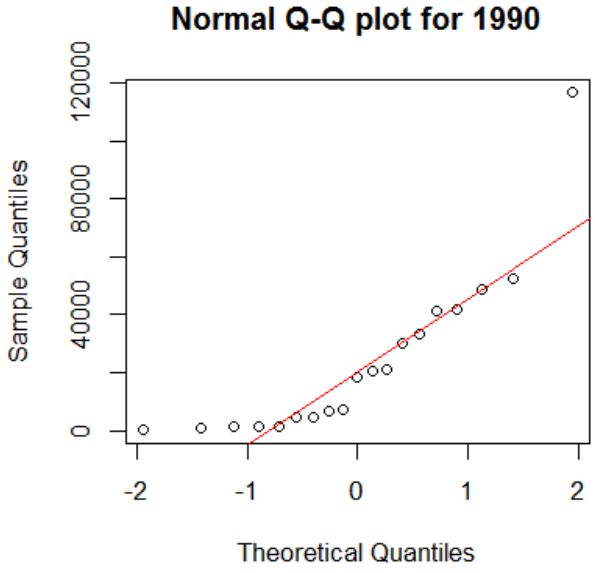
- Normal probability plot is a commonly used graphical method for checking the normality assumption.
- Many of our statistical tests require that the data are a random sample coming from a normal population. That is, the data are independently and identically distributed (i.i.d.) as normal.
- In practice, it is impossible to know the true distribution of our data. We can only hope that the distribution of data is not significantly different from a normal distribution.
- Normal probability plot is a plot of the sample quantiles versus the theoretical quantiles from a standard normal distribution. If the points are close to a 45 degree straight line, the sample quantiles are linearly related to the quantiles of the standard normal distribution and hence the normality assumption of the data is plausible.

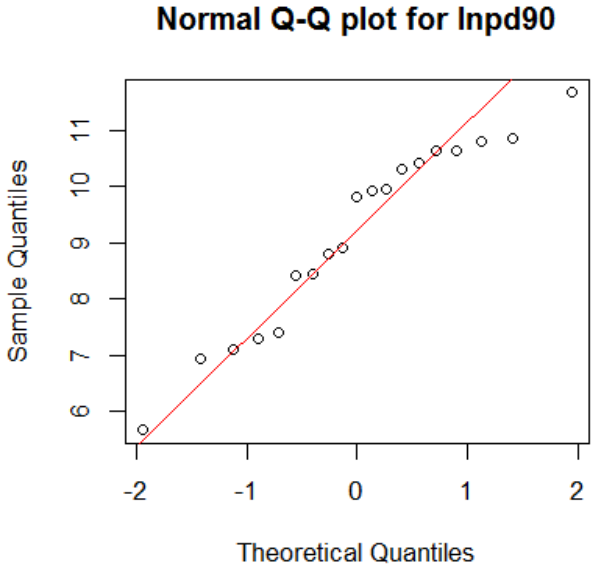
- R has a built-in function `qqnorm()` to provide this plot and `qqline()` to add a reference line to the plot.
- Let us illustrate this by producing the normal-QQ plot of the population density in year 1986 and year 1990 together with their log-transformed variables `lnpd86` and `lnpd90` with main title.
- What we expect in this plot is that the original variables are quite different from a normal distribution while the log-transformed variables should be closer to a normal distribution.

```r
par(mfrow=c(2,2))
qqnorm(d$year86,main="Normal Q-Q plot for 1986")
# qq-normal plot for year86
qqline(d$year86,col="red")
# add ref. line in red color
qqnorm(d$lnpd86,main="Normal Q-Q plot for lnpd86")
# qq-normal plot for lnpd86
qqline(d$lnpd86,col="red")
qqnorm(d$year90, main="Normal Q-Q plot for 1990")
# qq-normal plot for year90
qqline(d$year90,col="red")
qqnorm(d$lnpd90,main="Normal Q-Q plot for lnpd90")
# qq-normal plot for lnpd90
qqline(d$lnpd90,col="red")
```


Normal Q-Q plot for 1986
Sample Quantiles
0
40000
80000
140000
-2
-1
0
1
2
Theoretical Quantiles


Normal Q-Q plot for lnpd86
Sample Quantiles
6
7
8
9
10
11
12
-2
-1
0
1
2
Theoretical Quantiles


Normal Q-Q plot for 1990
Sample Quantiles
0
40000
80000
120000
-2
-1
0
1
2
Theoretical Quantiles


Normal Q-Q plot for lnpd90
Sample Quantiles
6
7
8
9
10
11
-2
-1
0
1
2
Theoretical Quantiles

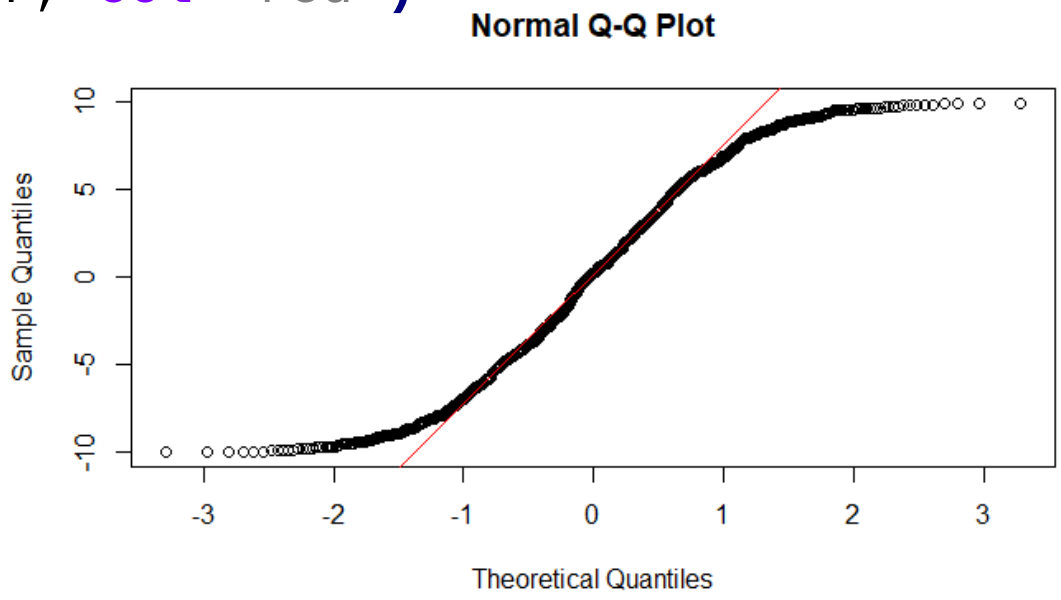
# General QQ plot

Consider a random sample form a uniform distribution.

```
r<-runif(1000,-10,10)
```

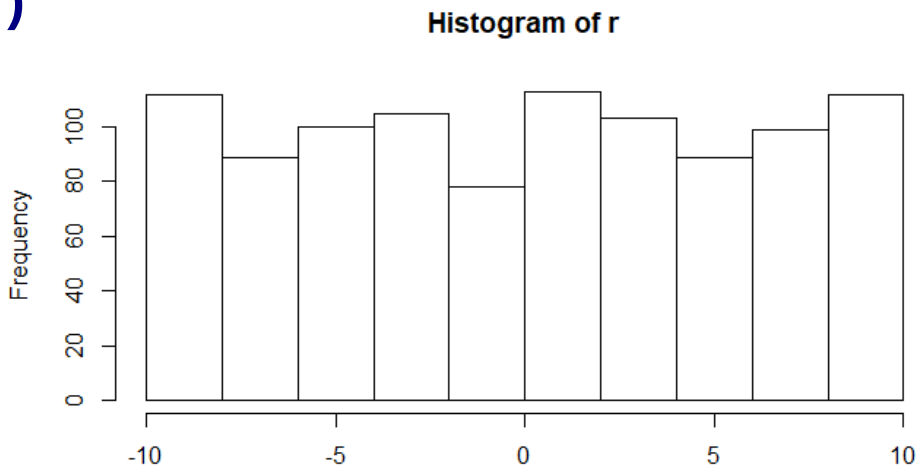
Suppose we don't know it is generated from uniform. We may check that it does not follow normal

```
qqnorm(r)
qqline(r, col="red")
```



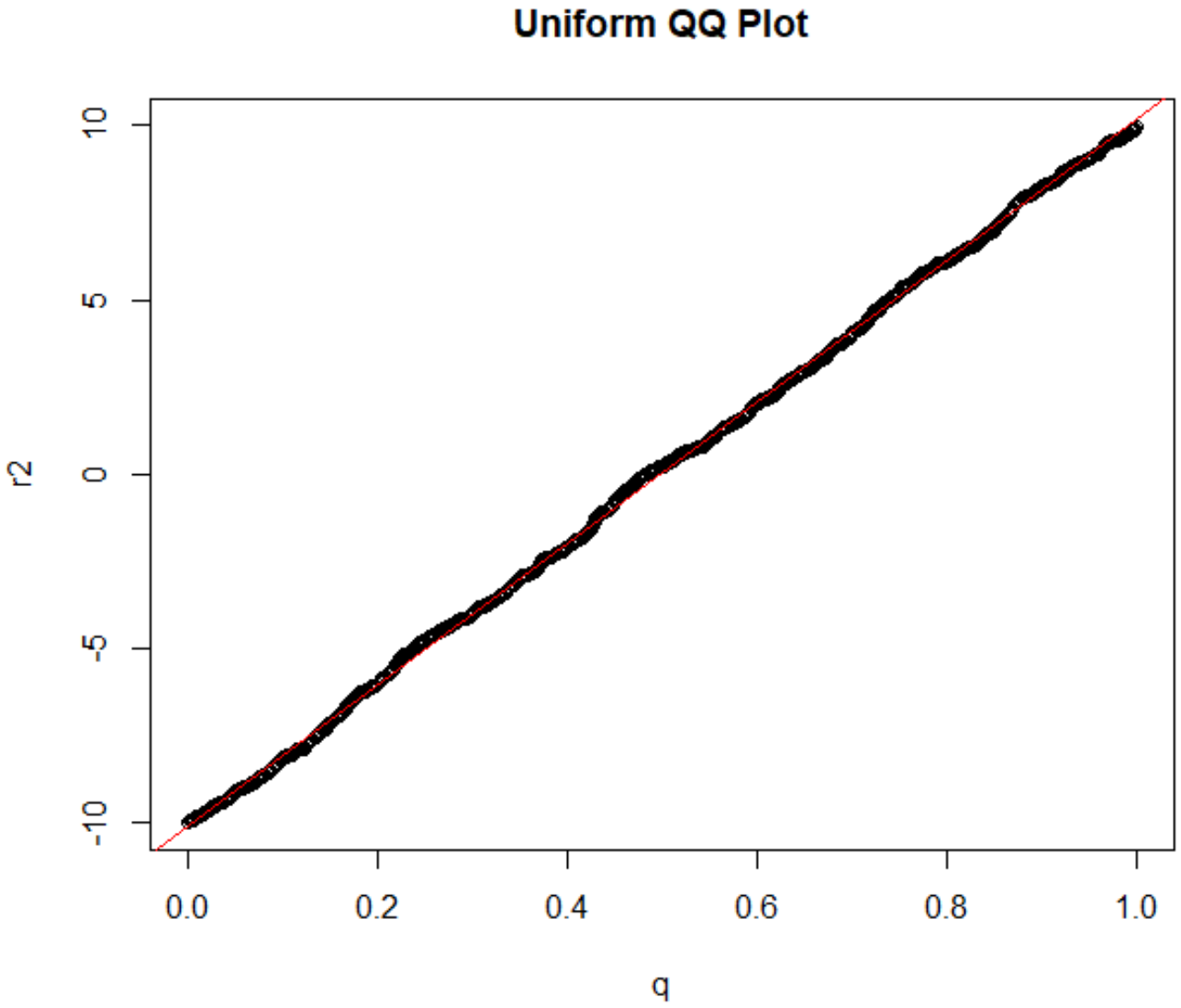
With a histogram, we suspect that it may have a uniform distribution.

```
hist(r)
```




We can verify this idea using a QQ plot

```
n<-length(r)
r2<-sort(r)
i<-((1:n)-0.5)/n
q<-qunif(i)
plot(q,r2,main="Uniform QQ Plot")
abline(lsfit(q,r2), col="red")
```


Uniform QQ Plot
r2
10
5
0
-5
-10
0.0
0.2
0.4
0.6
0.8
1.0
q

# Remarks

The QQ plot is a graphical tool to help us assess if a data set is normally distributed. The conclusion can be quite subjective.

There are several normality tests in statistics one could use to objectively determine if a given dataset is normally distributed. Here are some of them.

- **Shapiro-Wilk test**: This test is widely regarded as a reliable test of normality. It is best for smaller samples sizes ($n < 50$), as it has greater power to reject the null hypothesis of normality with smaller samples.
- **Kolmogorov-Smirnov test**: This is a more general test that compares the empirical distribution function of the sample with the cumulative distribution function of a reference distribution, in this case, the normal distribution. However, it has less power to reject the null hypothesis of normality with smaller samples compared to the Shapiro-Wilk test.

- **Anderson-Darling test**: This is a modification of the Kolmogorov-Smirnov test that gives more weight to the tails of the distribution. Like the Shapiro-Wilk test, it is more powerful with smaller samples.
- **Jarque-Bera test**: This test is based on the skewness and kurtosis of the distribution. It has greater power for larger samples ($n > 2000$).

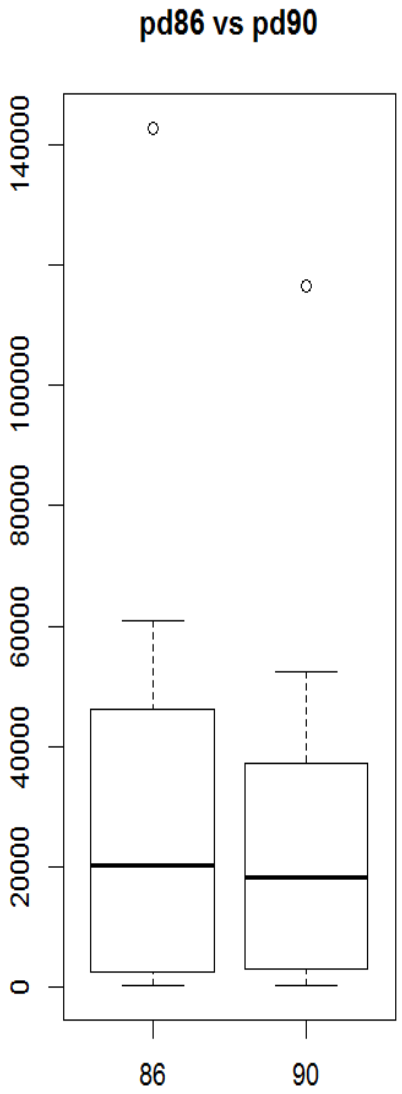
Each of these tests has its own strengths and weaknesses, and the choice of which to use depends on the specifics of the data set and the context in which it's being used. For instance, for small sample sizes, the Shapiro-Wilk test may be preferred, while for larger sample sizes, the Jarque-Bera test might be more appropriate.

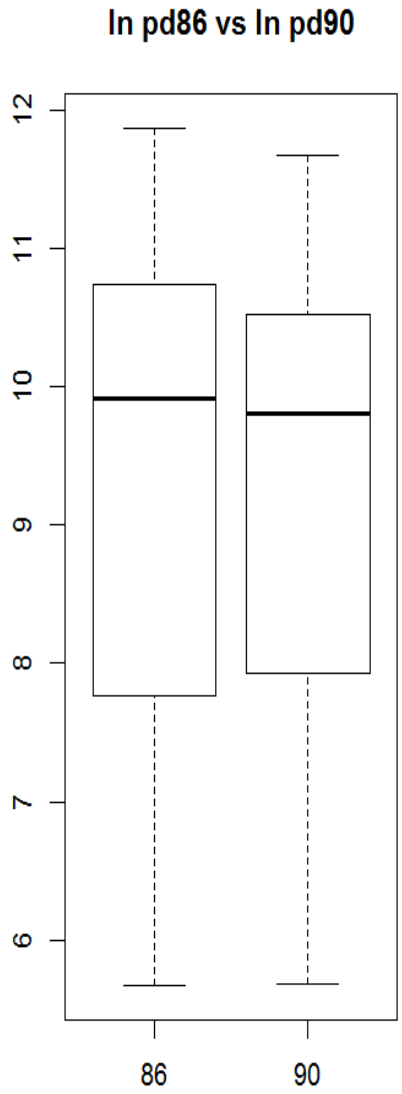
It is worth noting that no test is perfect, and the results should be used in conjunction with graphical methods, like QQ plots.

# Box plot

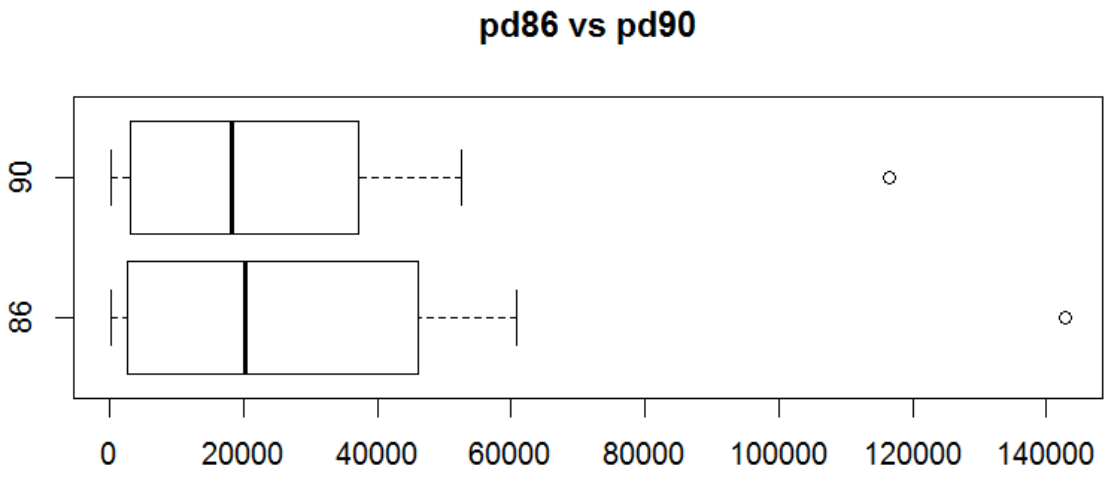
- Box plot is another commonly used graphical method for displaying distribution of data other than histogram.
- It can also be used to detect outliers and to compare the distribution of two samples.
- Again, R has a built-in function `boxplot()` to produce the boxplot.
- Let us illustrate the `boxplot()` function using our population density example. We want to produce a side-by-side boxplot for comparing the distributions of population density in 1986 with `lnpd86` and 1990 with `lnpd90`. We use the model formula (`y-variable ~ x-variable`) inside the `boxplot()` to produce these plots.

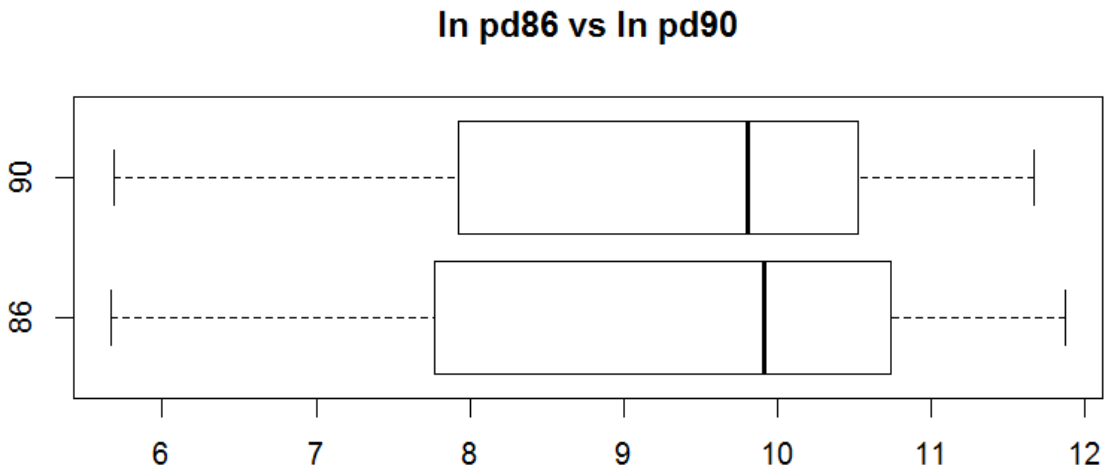
```r
par(mfrow=c(1,2))                    # set up a 1x2 multi-frame
yr<-c(rep(86,19),rep(90,19))
# create yr as year indictor
pd<-c(d$year86,d$year90)
# stack pd86 with pd90 as a vector
lnpd<-c(d$lnpd86,d$lnpd90)
# stack lnpd86 with lnpd90 as a vector
boxplot(pd~yr,main="pd86 vs pd90")
# boxplot for year86 and year90
boxplot(lnpd~yr,main="ln pd86 vs ln pd90")
# boxplot for lnpd86 and lnpd90
par(mfrow=c(2,1))                    # set up a 2x1 multi-frame
boxplot(pd~yr,horizontal=T,main="pd86 vs pd90")
# horizontal boxplot
boxplot(lnpd~yr,horizontal=T,main="ln pd86 vs ln pd90")
# horizontal boxplot
```




- The lower and upper edge of the box are the 25th and 75th percentile, the data points lie outside the lower and upper fence are considered as outliers.
- Clearly there is an outlier in each sample.
- This outlier disappears when the log-transformation is used.

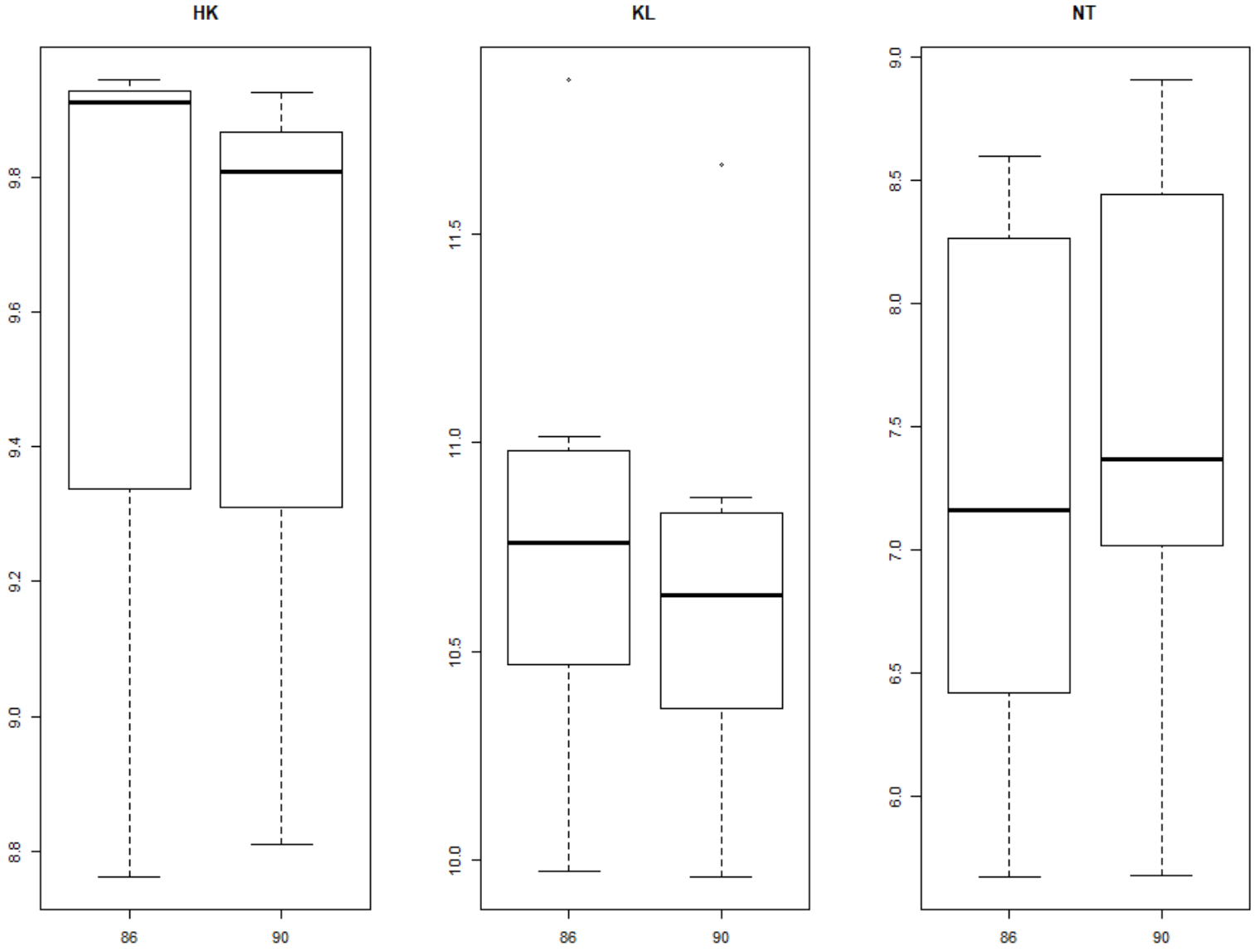

pd86 vs pd90
90
86
0
20000
40000
60000
80000
100000
120000
140000


ln pd86 vs ln pd90
90
86
6
7
8
9
10
11
12

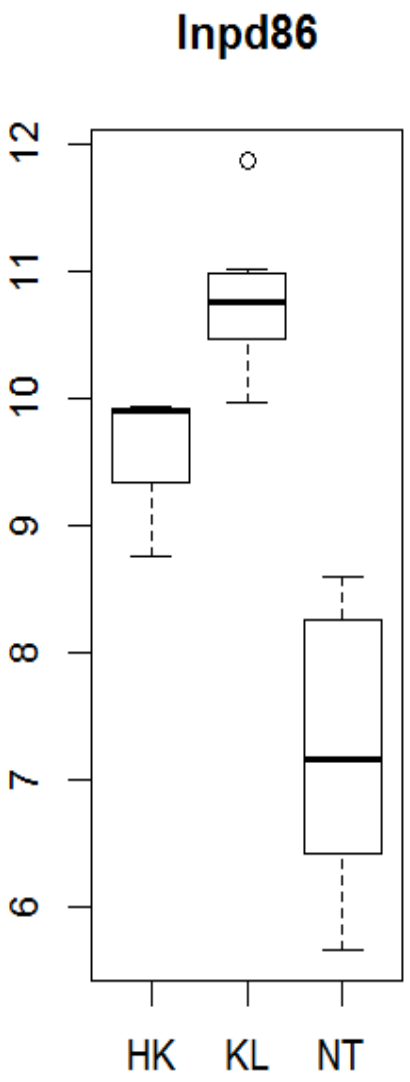
There are other ways to compare and present these data. For example, we can split the data by Region (HK, KL, NT) and produce the boxplots as below.

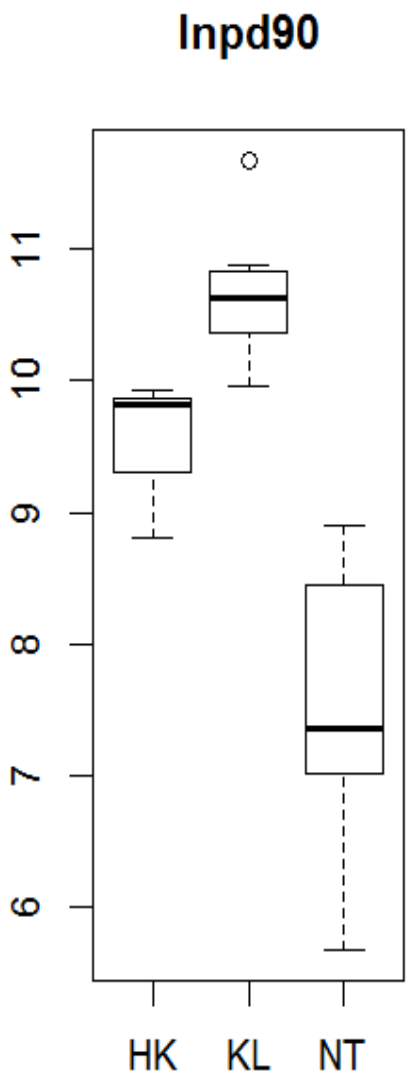
```
s86<-split(d$lnpd86,d$Region) # split year86 by Region
s90<-split(d$lnpd90,d$Region) # split year90 by Region
par(mfrow=c(1,3))             # set multiframe 1x3
boxplot(c(s86$HK,s90$HK)~rep(c(86,90),each=length(s86$HK)
),main="HK",xlab="",ylab="")  # boxplot for HK
boxplot(c(s86$KL,s90$KL)~rep(c(86,90),each=length(s86$KL)
),main="KL",xlab="",ylab="")  # boxplot for KL
boxplot(c(s86$NT,s90$NT)~rep(c(86,90),each=length(s86$NT)
),main="NT",xlab="",ylab="")  # boxplot for NT
par(mfrow=c(1,1))         # reset multiframe to 1x1
title(sub="ln pd86 vs ln pd90 by Region")    # sub-title


par(mfrow=c(1,2))         # set mfrow to (1,2)
boxplot(lnpd86~Region,data=d,main="lnpd86")
# boxplot using formula
boxplot(lnpd90~Region,data=d,main="lnpd90")
```




ln pd86 vs ln pd90 by Region


lnpd86
12
11
10
9
8
7
6
HK
KL
NT

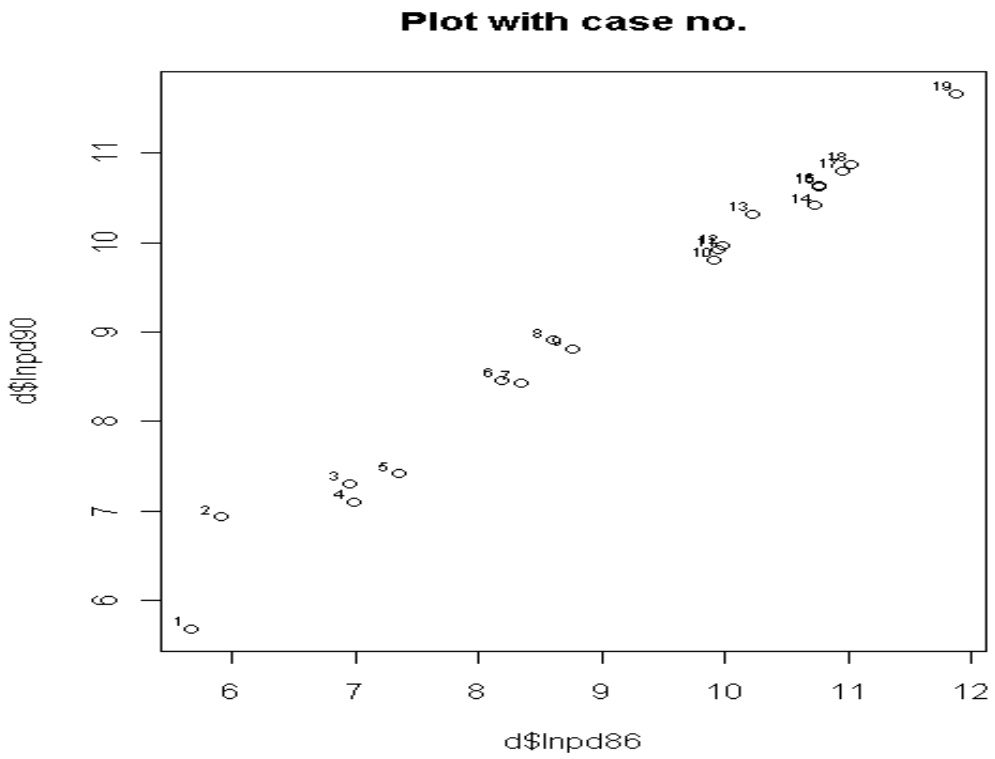

lnpd90
11
10
9
8
7
6
HK
KL
NT

- Clearly `HK` and `KL` has decreased population density while `NT` has increased population density as expected.
- Finally we can boxplot to compare different regions as the graph on the second plot.
- Note that `Region` is a factor object in `d`. We use the formula `lnpd86~Region` and `lnpd90~Region` to produce these boxplots.
- Clearly the population density in `KL` is the highest and `NT` is the lowest.

# Scatter plot

- Scatter plot is a very natural and useful graphical method to find out the relationship between two variables.
- From the plot, we can have a better idea whether the two variables are linearly or nonlinearly related or if these two variables have high or low, positive or negative correlation.

```
par(mfrow=c(1,1))
# reset multiframe to 1x1
plot(d$lnpd86,d$lnpd90,main="Plot with case no.")
# plot lnpd90 vs lnpd86
text(d$lnpd86-0.1,d$lnpd90+0.1,cex=0.6)
# add case no to the points
```

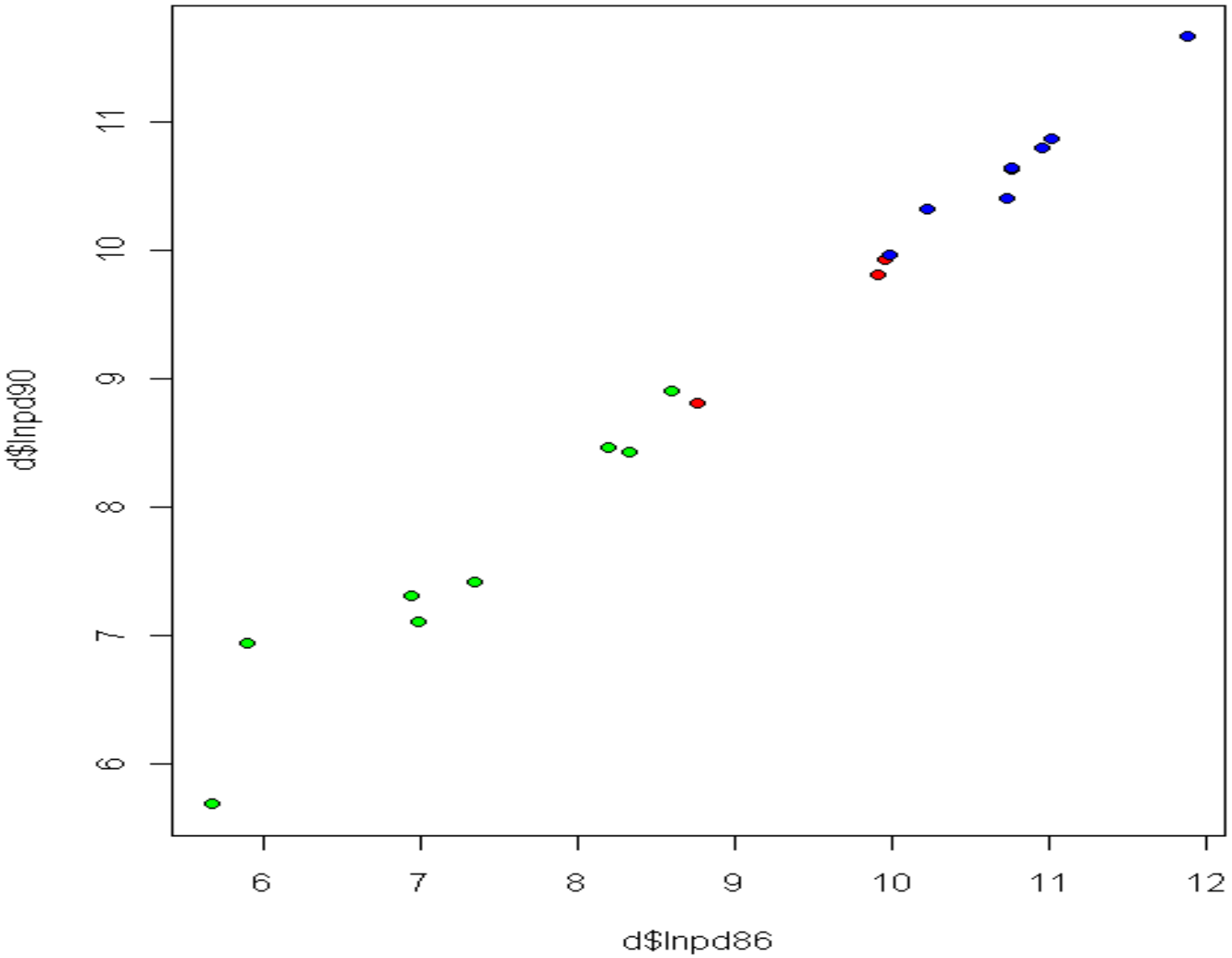


- The `main` option in `plot()` is to add the main title of the plot.
- `text()` will add case number in the plot with x- and y-coordinates.
- `-0.1` and `+0.1` are to offset and add to the points to avoid overlapping.

# Using symbol and colour in scatter plot

We can produce a scatter plot of `lnpd90` vs `lnpd86` with colour for each region. First note that `d$Region` is a factor object with 3 levels:

```
d$Region
 [1] NT NT NT NT NT NT NT NT HK HK HK KL KL KL KL KL KL KL KL KL
Levels: HK KL NT
plot(d$lnpd86,d$lnpd90,pch=21,bg=c("red","blue","green")[d$Region])
```

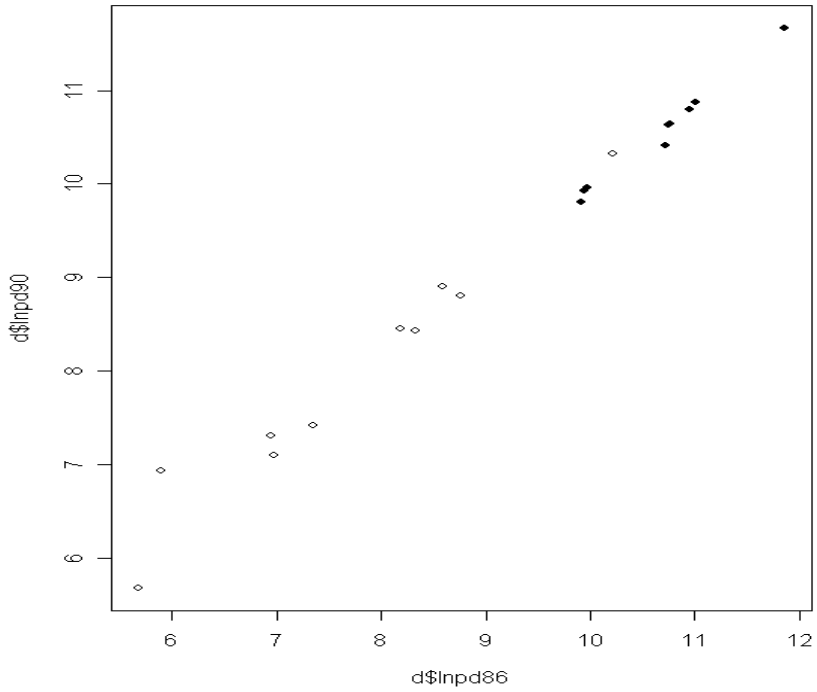

d$lnpd90
6
7
8
9
10
11
d$lnpd86
6
7
8
9
10
11
12

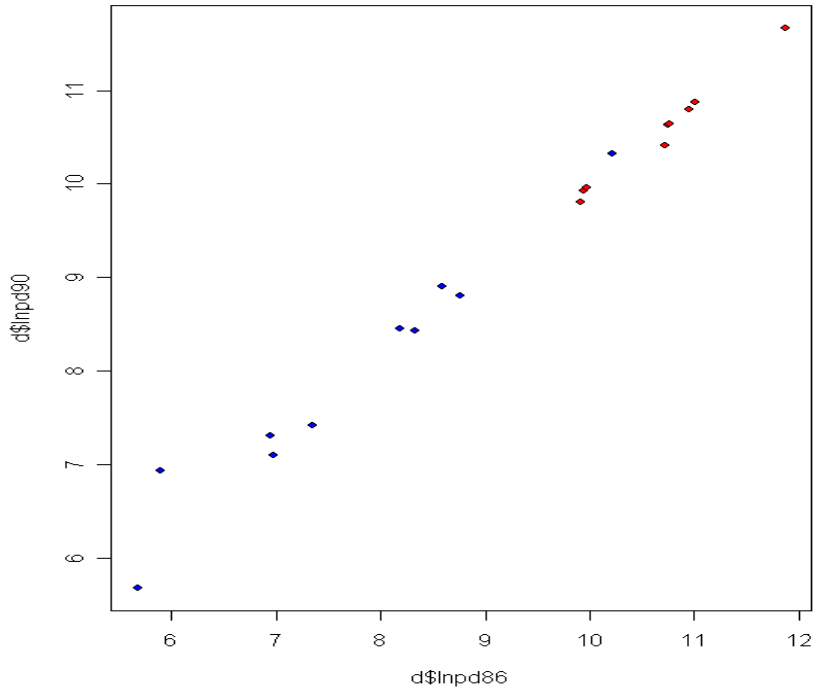
- In this example, one of the interesting questions is to find out which district have increased or decreased population.
- We can use different symbols and colour to indicate these two types of districts.
- First let us explain how to select districts that the population density had increased between 86 and 90.

```
d[d$year86<d$year90,]
# select from d according to the condition
    district year86 year90 Region    lnpd86    lnpd90
1    Islands    290    293     NT  5.669881  5.680173
2   Sai_Kung    365   1026     NT  5.899897  6.933423
3     Tai_Po   1033   1496     NT  6.940222  7.310550
4      North   1074   1211     NT  6.979145  7.099202
5  Yuen_Long   1545   1664     NT  7.342779  7.416980
6   Tuen_Mun   3611   4711     NT  8.191740  8.457655
7  Tsuen_Wan   4159   4581     NT  8.333030  8.429673
8    Sha_Tin   5402   7378     NT  8.594525  8.906258
9   Southern   6380   6701     HK  8.760923  8.810012
13   Eastern  27387  30316     KL 10.217824 10.319431
```

Note that `d$year86<d$year90` creates a logical vector **TRUE** or **FALSE** depending on the condition and `d[d$year86<d$year90,]` would select the observation according to this logical vector.

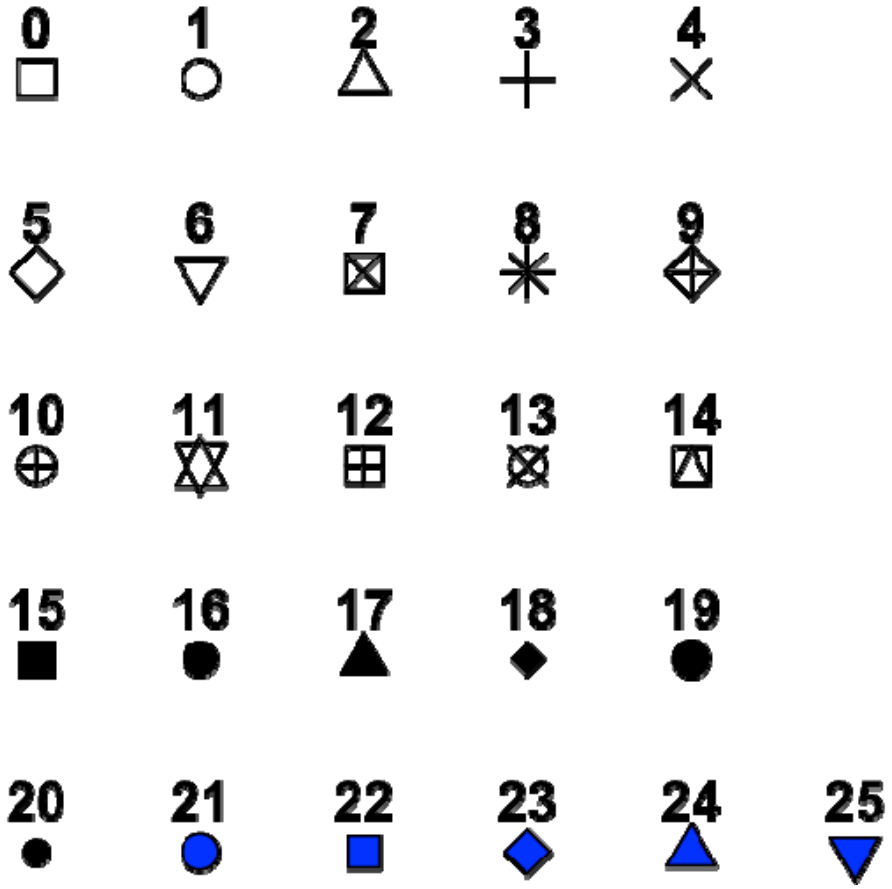
```
# create a numeric vector, id=2 if year86<year90; id=1 otherwise
id<-(d$year86<d$year90)+1
par(mfrow=c(1,2))
# create 1x2 multiframe graphics
# plot lnpd90 vs lnpd86 with different symbols according to id
plot(d$lnpd86,d$lnpd90,pch=c(19,21)[id])
# plot lnpd90 vs lnpd86 with color according to id: red=decrease, blue=increase
plot(d$lnpd86,d$lnpd90,pch=21,bg=c("red","blue")[id])
```




From the plot, it is clear that there is an increase in population density for the developing districts (low population density) while decrease in the developed districts (high population density).

The option pch stands for plotting character (19=solid circle, 20=bullet, 21=circle, 22=square, 23=diamond, 24=triangle point-up, 25=triangle point down).



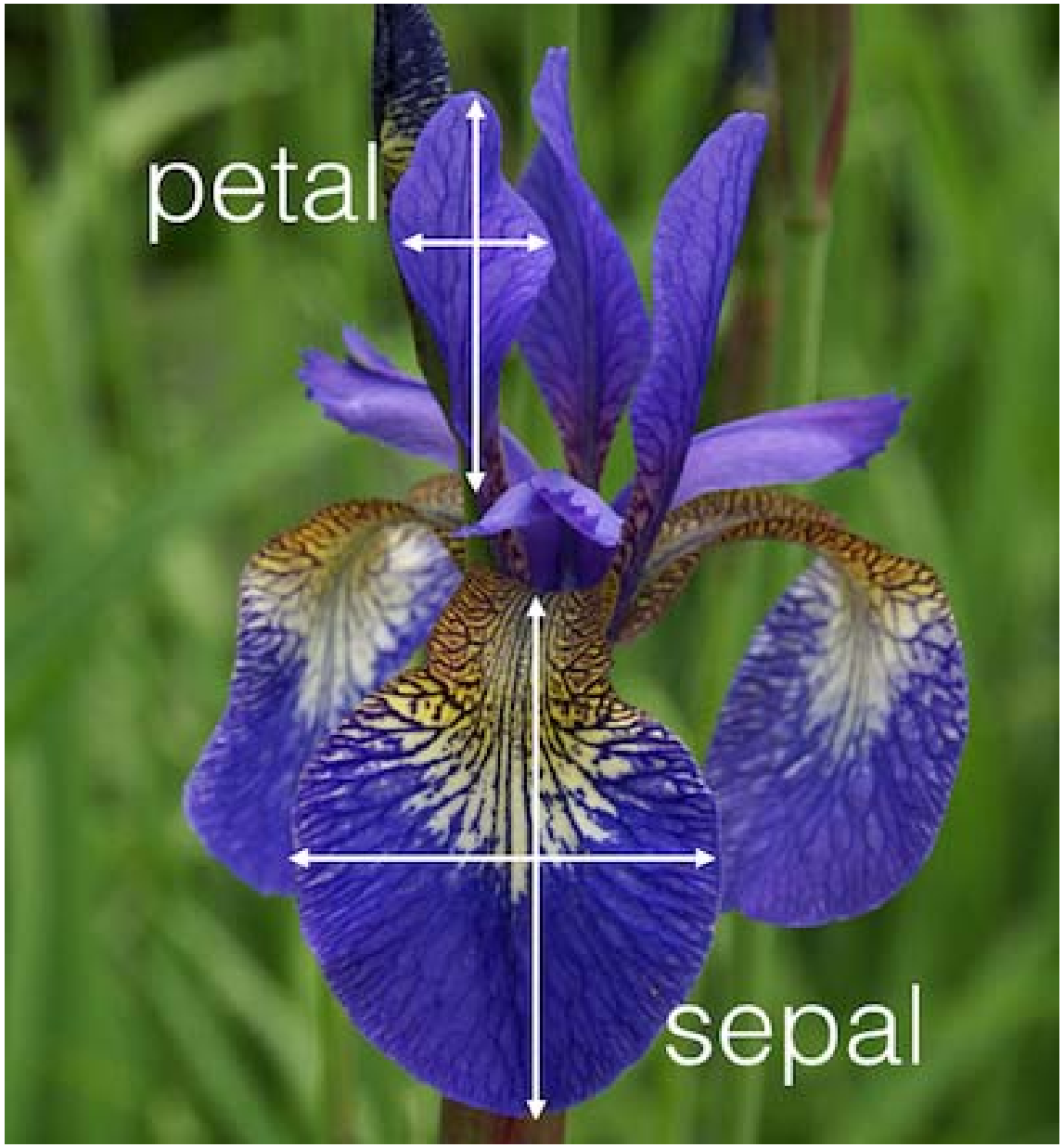
# Question

How can we add a line to separate the red points from the blue points?

[Hint: see `help(abline)`]

# Matrix Scatter Plot

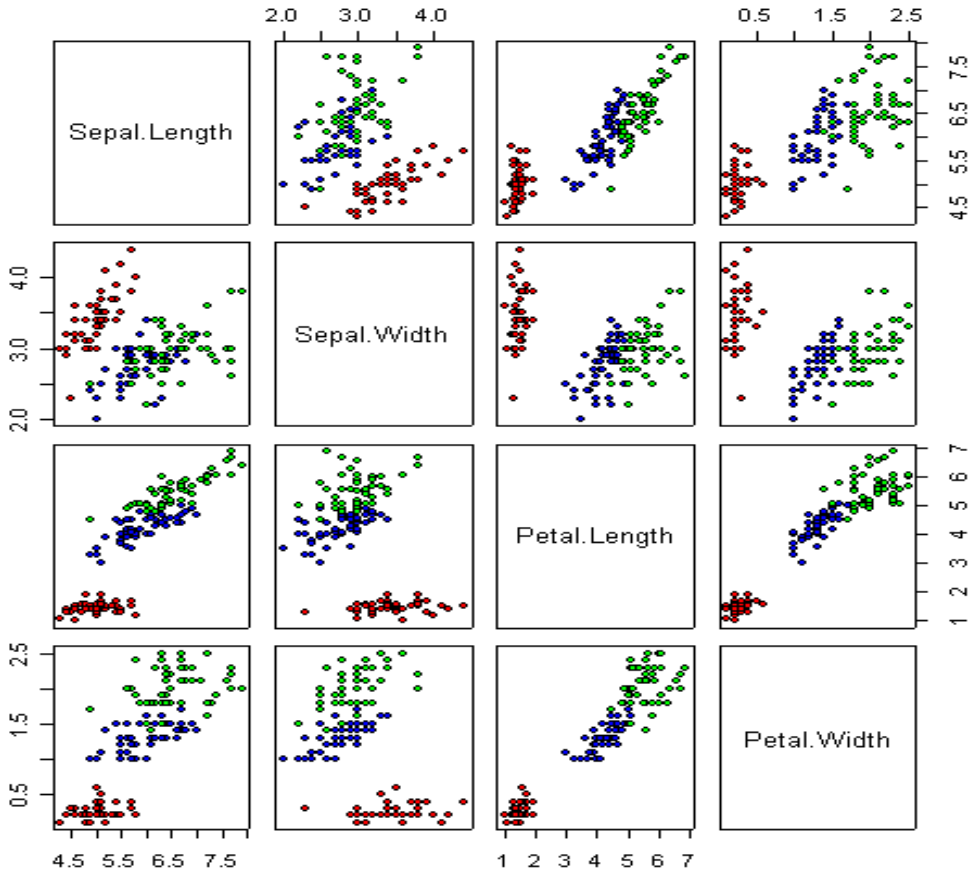
- Let us consider another famous dataset `iris.csv` by R.A. Fisher.
- This `iris.csv` dataset contains 150 observations and 5 variables (`sepal length`, `sepal width`, `petal length`, `petal width` and `species label`).
- The first four variables are measurement of iris flower.
- The last variable indicates the specimen of the flower.
  1: setosa
  2: versicolor
  3: virginica
- Now we produce a matrix scatter plot of these four variables with different colours for each specimen using the function `pairs()`.




Source: blog.kaggle.com


Source: rajritvikblog.wordpress.com

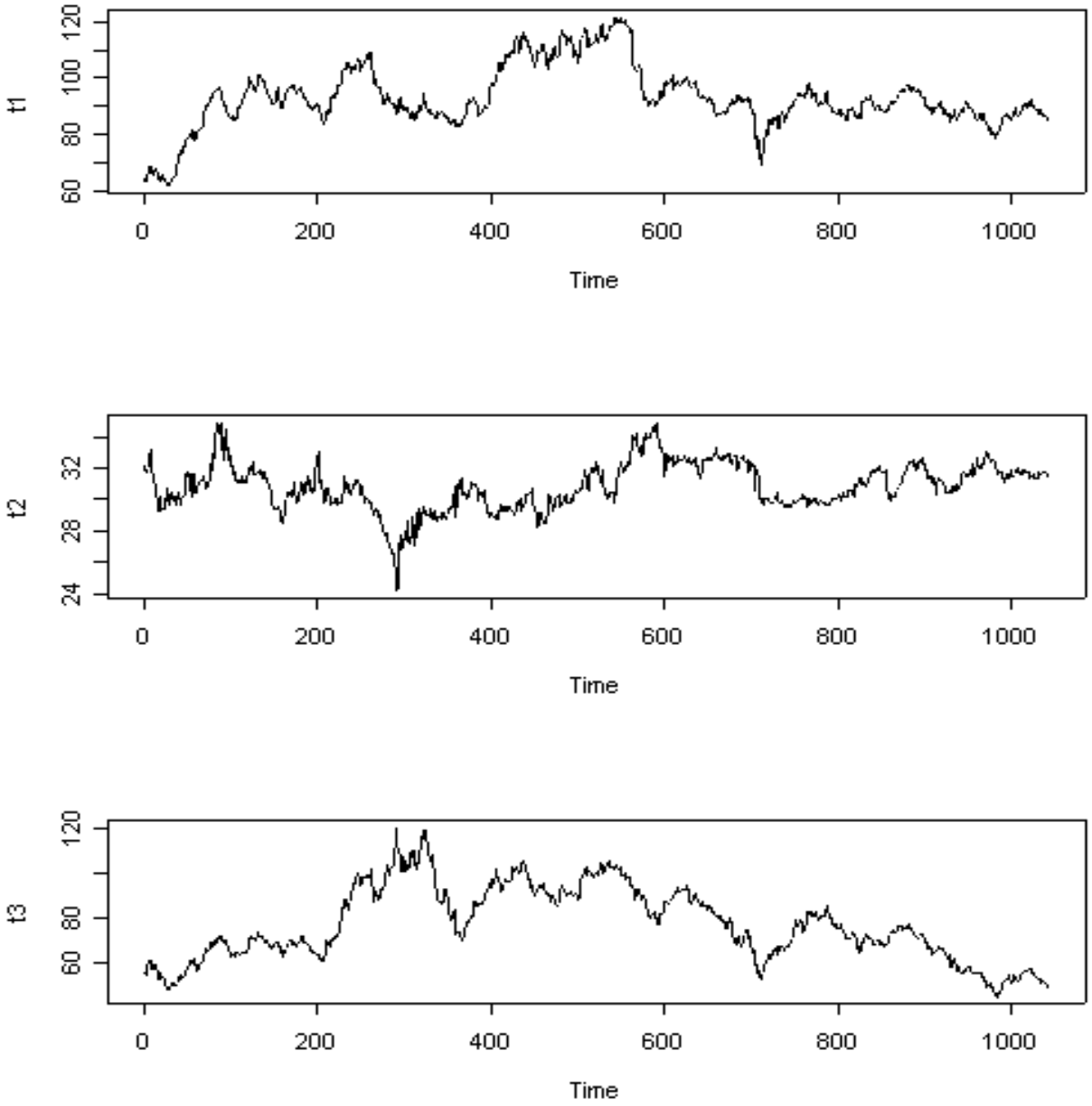
```
d<-read.csv("iris.csv")
# produce the scatter plot matrix with color according to species
pairs(d[,1:4],pch=21,bg=c("red","blue","green")[d$Species])
```



# Time series plot

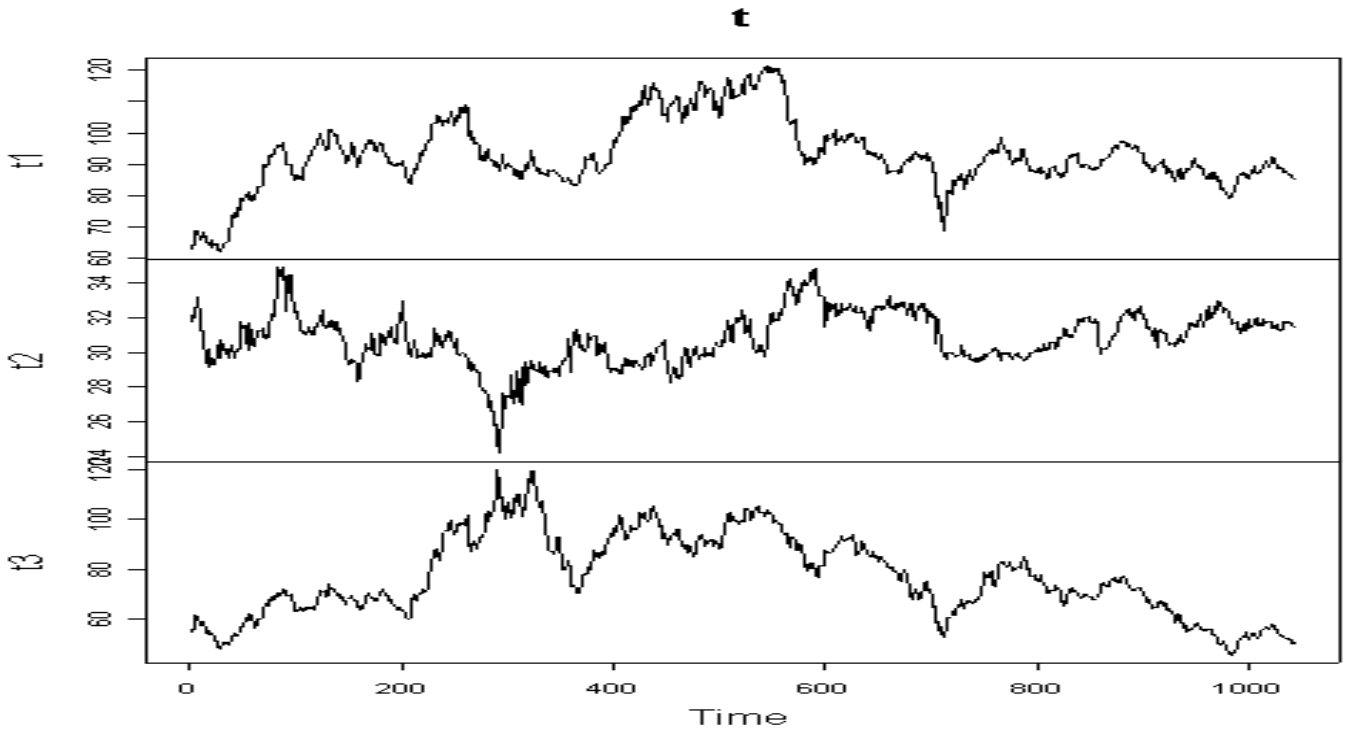
- Another commonly used statistical graphic is time series plot.
- It is simply a plot of values (index) against the time (x-axis).
- The file `stock.dat` contains adjusted daily closing price for the stock HSBC (0005), CLP (0002) and Cheung Kong (0001) from 1/1/1999 to 31/12/2002. Let us read in this data file and produce some time series plots.

```
d<-read.table("stock.dat")
# read in data
names(d)
# display names in d
[1] "HSBC" "CLP"  "CK"
t1<-as.ts(d$HSBC)
# change to time series object
t2<-as.ts(d$CLP)
t3<-as.ts(d$CK)
par(mfrow=c(3,1))
# set up multiframe graphics
plot(t1); plot(t2); plot(t3)
# time series plots
```


t1
60
80
100
120
0
200
400
600
800
1000
Time
t2
24
28
32
0
200
400
600
800
1000
Time
t3
60
80
120
0
200
400
600
800
1000
Time

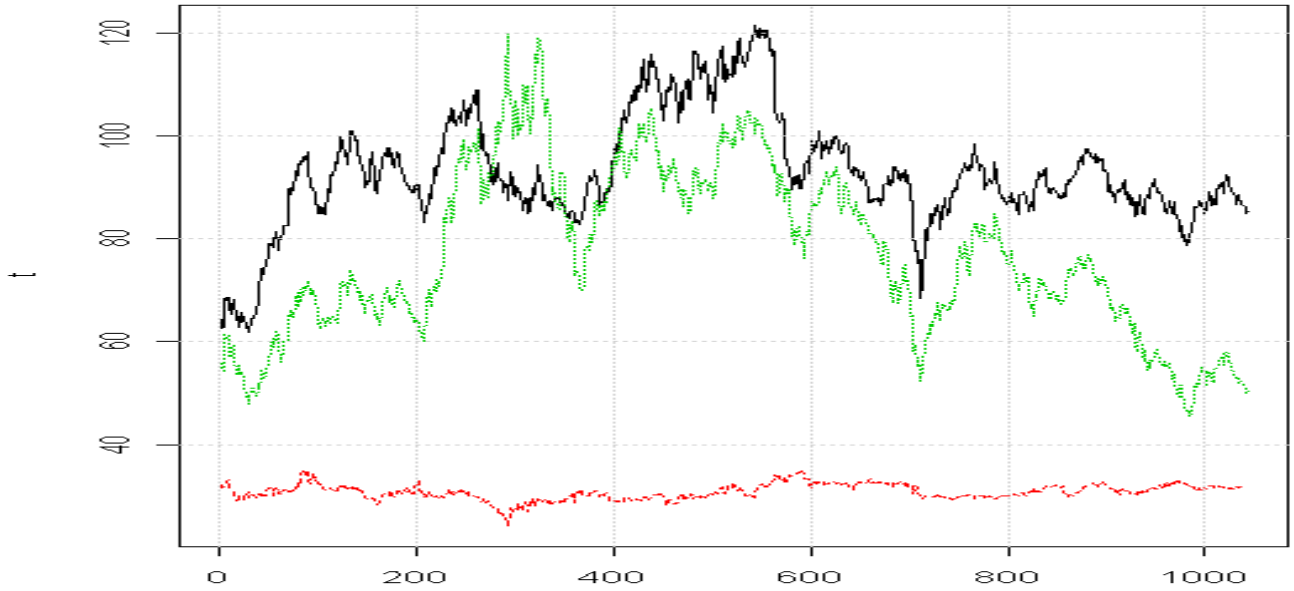
We can combine these time series into a matrix and plot them in a single plot.

```
t<-cbind(t1,t2,t3)
# combine these time series column-by-column
par(mfrow=c(1,1))
# reset the multiframe graphics into one plot
plot(t)              # plot the time series matrix
```



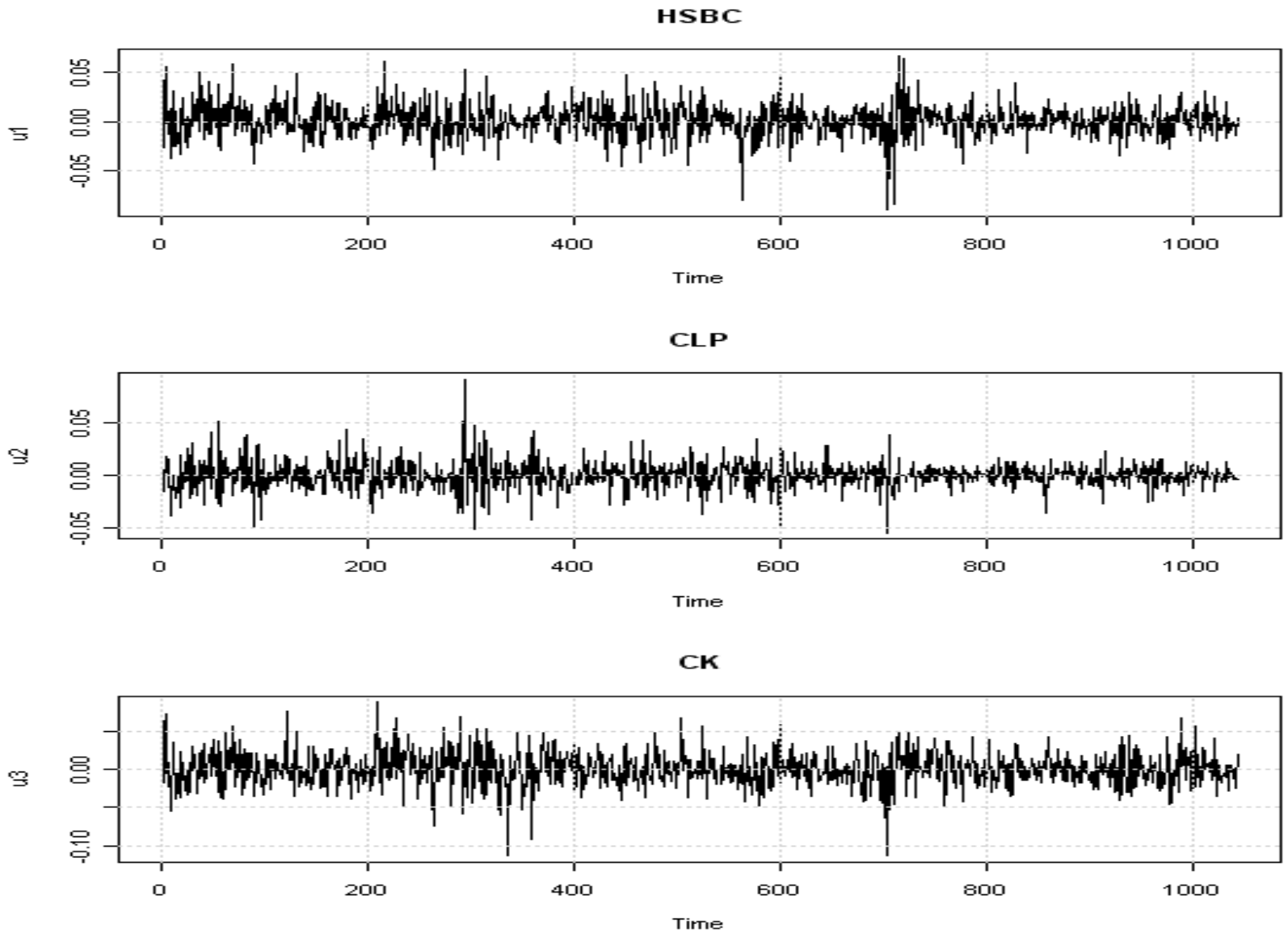
If we want to plot them on the same y-axis, we can use `matplot`. We can also add in grid lines to the plots or `matplot` by entering `grid()`.

```
matplot(t,type="l")
# use matplot to plot on the same y-axis
grid()                          # add in grid lines
```



- One commonly used transformation in financial time series is $u_t = \ln(S_t/S_{t-1})$, where $S_t$ is the stock price or index at time t.
- The built-in function `lag()` can only be applied to a time series object and will return the next value in the time series.

```
par(mfrow=c(3,1))
# define 3x1 multiframe graphic
u1<-log(lag(t1)/t1)          # transform
u2<-log(lag(t2)/t2)
u3<-log(lag(t3)/t3)
plot(u1,main="HSBC"); grid()
# plot with main title and add grid lines
plot(u2,main="CLP"); grid()
plot(u3,main="CK"); grid()
```


HSBC
u1
0.05
0.00
-0.05
0
200
400
600
800
1000
Time
CLP
u2
0.05
0.00
-0.05
0
200
400
600
800
1000
Time
CK
u3
0.00
-0.10
0
200
400
600
800
1000
Time

# Mathematical function plot

Consider the function

$$f(x) = \begin{cases} x \sin \frac{\pi}{x}, & x \neq 0, \\ 0, & x = 0. \end{cases}$$

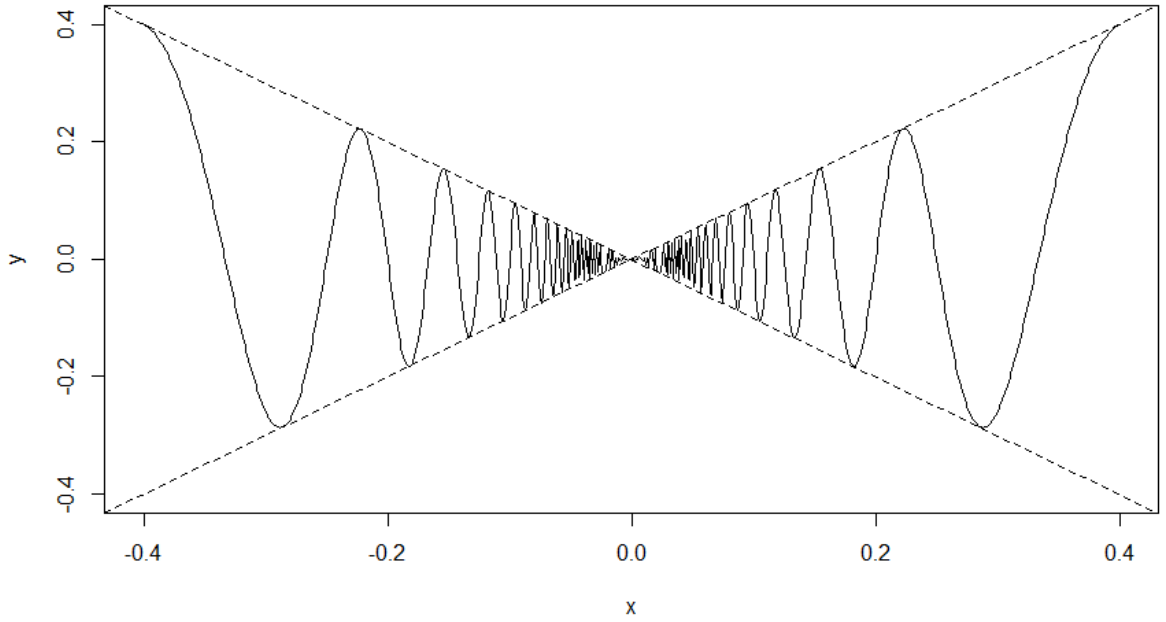
We would like to plot this function on [-0.4,0.4].

Furthermore, we would like to add two asymptotes

$$y = \pm x$$

to our plot

```r
# Mathematical function
x1 <- seq(-0.4,-0.001, by=0.001)
x2 <- seq(0.001,0.4, by=0.001)
x <- c(x1, 0, x2) # x = 0 is a special case
y <- c(x1*sin(pi/x1), 0, x2*sin(pi/x2))
plot(x,y, type="l", ylim=c(-0.4,0.4))
abline(0,1, lty=2)
abline(0,-1, lty=2)
```

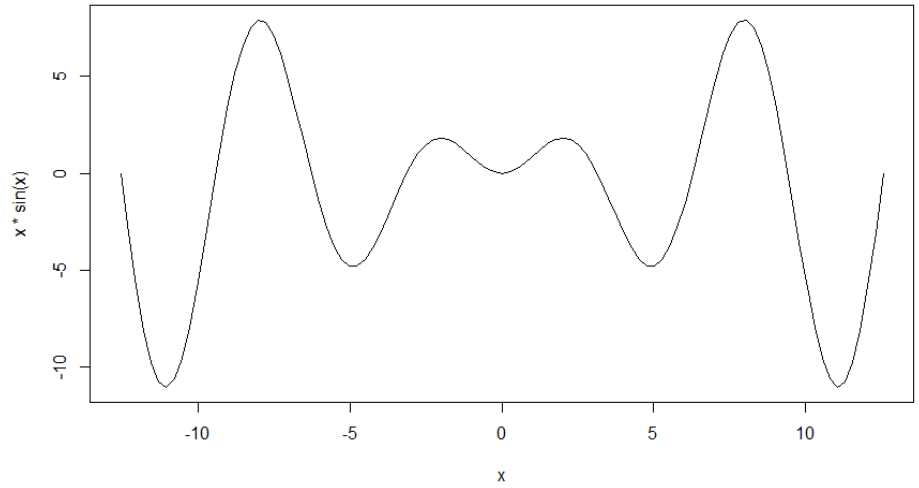


# `curve()` function

For simple mathematical functions, the `curve()` function is a handy tool for plotting.

For example,

```
curve(x*sin(x),-4*pi,4*pi)
```

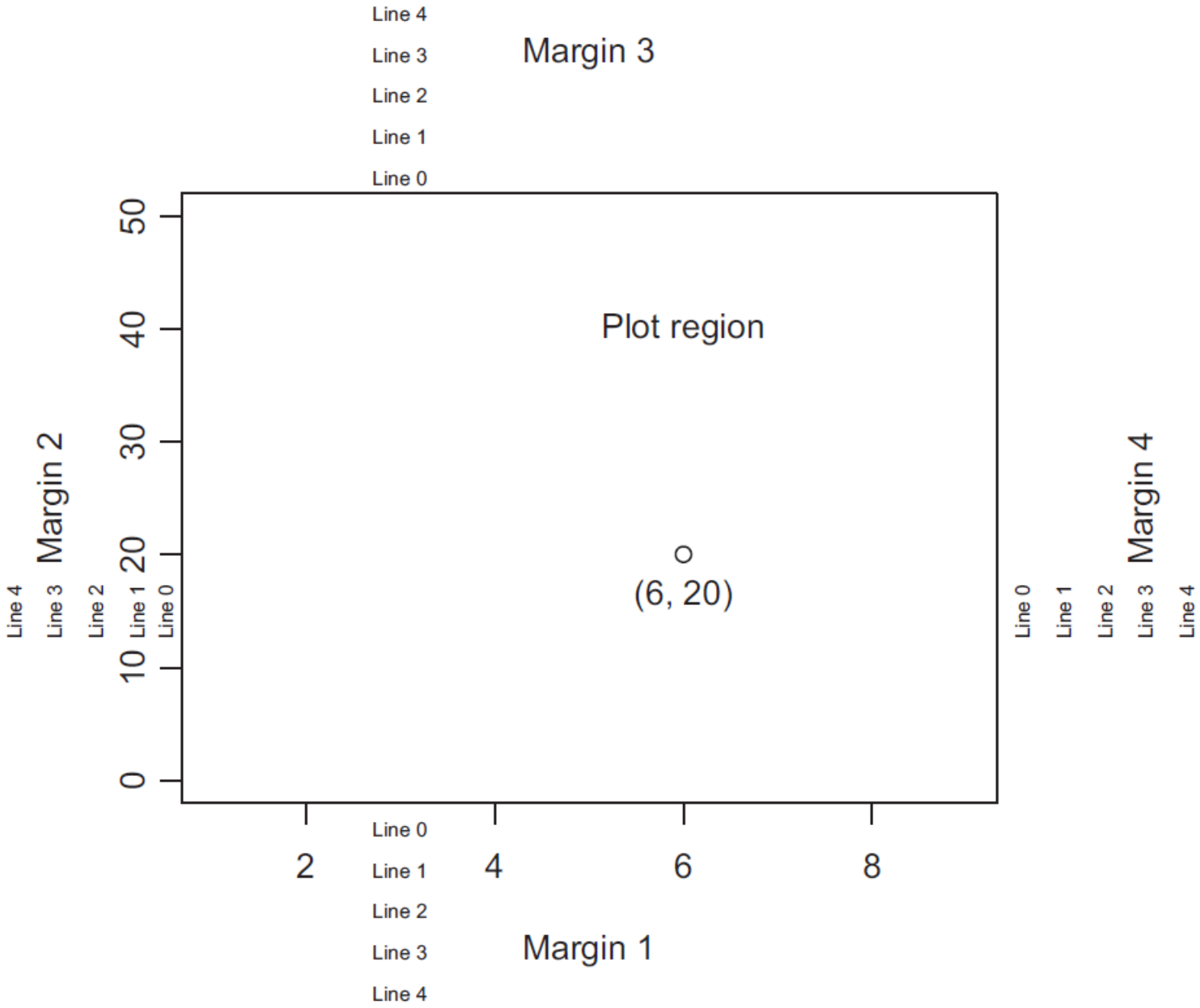


# Low level graphics functions

- We will describe some of the low level functions, which are available for users to customize their plots.
- We will start with a description of how R views the page it is drawing on, then show how to add points, lines, and text to existing plots, and finish by showing how some of the common graphics settings are changed.

# The plotting region and margins

- Base graphics in R divides up the display into several regions.
- The plot region is where data will be drawn. Within the plot region R maintains a coordinate system based on the data.
- Outside the plot region are the margins, numbered clockwise from 1 to 4, starting at the bottom.
- Normally text and labels are plotted in the margins, and R positions objects based on a count of lines out from the plot region.


Line 4
Line 3
Line 2
Line 1
Line 0
Margin 3
Plot region
(6, 20)
Margin 2
Margin 4
Margin 1
0
10
20
30
40
50
2
4
6
8

# Adding to plots

Several functions exist to add components to the plot region of existing graphs:

```
points(x, y, ...) # adds points
lines(x, y, ...) # adds line segments
text(x, y, labels, ...) # adds text into the graph
abline(a, b, ...) # adds the line $y = a + bx$
abline(h = y, ...) # adds a horizontal line
abline(v = x, ...) # adds a vertical line
polygon(x, y, ...)
# adds a closed and possibly filled polygon
segments(x0, y0, x1, y1, ...) # draws line segments
arrows(x0, y0, x1, y1, ...) # draws arrows
symbols(x, y, ...)
# draws circles, squares, thermometers, etc.
legend(x, y, legend, ...) # draws a legend
```

# Example: Orange tree

- The `Orange` data frame is in the built-in `datasets` package in R.
- It consists of 35 observations on the age (in days since December 31, 1968) and the corresponding circumference of five different orange trees, with identifiers

```
> unique(as.character(Orange$Tree))
[1] "1" "2" "3" "4" "5“
```

- Since `Orange$Tree` is a factor, we use `as.character()` to get the displayed form, and `unique()` to select the unique values.

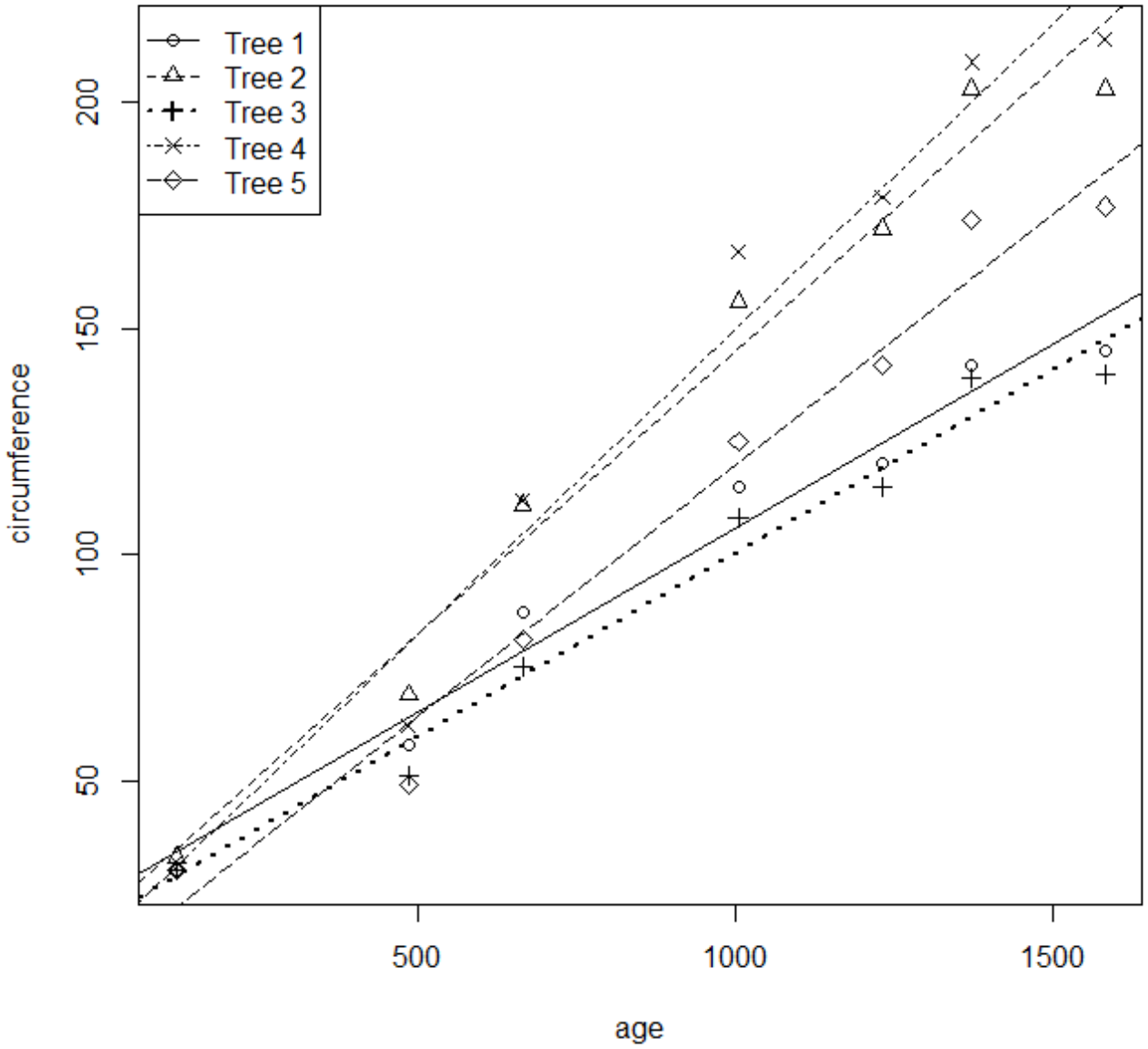
The basic scatterplot is obtained from

```
plot(circumference~age, pch =
as.numeric(as.character(Tree)),data = Orange)
```

The best-fit lines for the five trees can be obtained using the `lm()` function which relates `circumference` to age for each tree.

A legend has been added to identify which data points come from the different trees.

```
abline(lm(circumference~age, data = Orange, subset =
Tree == "1"), lty = 1)
abline(lm(circumference~age, data = Orange, subset =
Tree == "2"), lty = 2)
abline(lm(circumference~age, data = Orange, subset =
Tree == "3"), lty = 3, lwd = 2)
abline(lm(circumference~age, data = Orange, subset =
Tree == "4"), lty = 4)
abline(lm(circumference~age, data = Orange, subset =
Tree == "5"), lty = 5)
legend("topleft", legend = paste("Tree", 1:5), lty =
1:5, pch = 1:5, lwd = c(1, 1, 2, 1, 1))
```


Tree 1
Tree 2
Tree 3
Tree 4
Tree 5
circumference
50
100
150
200
500
1000
1500
age

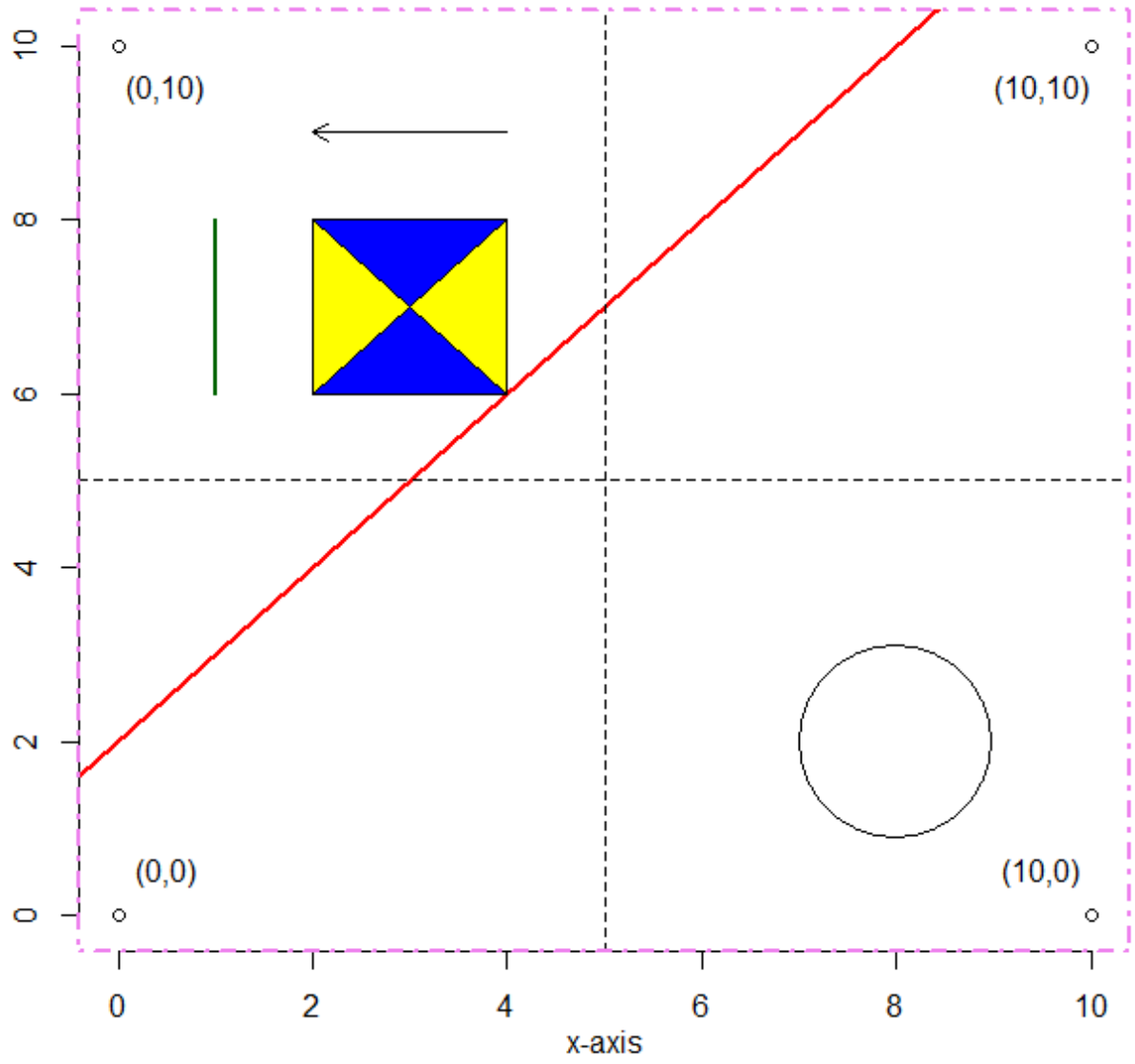
# Labelling

One may also wish to annotate graphs outside the plot region. Several functions exist to do this:

```
title(main, sub, xlab, ylab, ...)
# adds a main title, a subtitle,
# an x-axis label and/or a y-axis label
mtext(text, side, line, ...)
# draws text in the margins
axis(side, at, labels, ...)
# adds an axis to the plot
box(...) # adds a box around the plot region
```

# Example: A sketch

```
plot(0, 0, type="n", xlim=c(0,10), ylim=c(0,10),
      bty="n", xlab="", ylab="")
points(c(0,0,10,10), c(0,10,0,10))
text(
      c(0.5,0.5,9.5,9.5),
      c(0.5,9.5,0.5,9.5),
      labels = c("(0,0)","(0,10)","(10,0)","(10,10)")
)
abline(0,1,lwd=2,col="red")
abline(h=5,lty=2); abline(v=5,lty=2)
polygon(c(2,4,4,2), c(6,6,8,8), col="yellow")
polygon(c(2,4,2,4), c(6,6,8,8), col="blue")
segments(1,6,1,8, col="darkgreen", lwd=2)
arrows(4,9,2,9, length = 0.1)
symbols(8,2, circles=1, add=T, inch=F)
mtext("x-axis",1,2)
box(lty=4, lwd=2, col="violet")
```


(0,10)
(10,10)
(0,0)
(10,0)
0
2
4
6
8
10
x-axis