Lecture 4: Date and Time, Plots

Cheng Lu

Overview

- Date and Time
    - Date Object
    - Date and Time Objects
- Plot
    - X-Y Plot
    - Scatter Plot
    - Bar Plot
    - Box Plot
- Homework

Date Object

as.Date() function convert character to objects of class "Date" representing calendar dates.

Example

```
> myDate <- as.Date("2020-09-21")
> mode(myDate)
[1] "numeric"
> class(myDate)
[1] "Date"
> # internally it is number of days since Jan 01, 1970
> as.numeric(myDate)
[1] 18526
> myDates <- as.Date(c("1970-01-01", "1970-01-02"))
> as.numeric(myDates)
[1] 0 1
```

Date Object

as.Date() can only recognize some standard time type.

Example

```
> as.Date("1970/01/01")
[1] "1970-01-01"
> as.Date("01/01/1970") #non-standard input
[1] "0001-01-19"
> # set up date format when we generate dates
> as.Date("02/24/2015", format = "%m/%d/%Y")
[1] "2015-02-24"
> as.Date("15,Feb-24", format = "%y,%b-%d")
[1] "2015-02-24"
> as.Date("February24 2015", format = "%B%d %Y")
[1] "2015-02-24"
```

Date Object

Below are the formats for dates

Formats for dates

Code	Value	Example
%d	Day of the month	23
%m	Month	01
%b	Month(abbreviated)	Jan
%B	Month(full)	January
%y	Year(2 digits)	90
%Y	Year(4 digits)	1990

Table: Formats for dates

The abbreviated month and the full month will be different for different locale. use *Sys.setlocale("LC_TIME", "English")* to change locale time into English.

Date Object

format() function convert "Date" object to character.

Example

```
> format(Sys.Date()) # system date
[1] "2020-01-06"
> myDate2 <- format(Sys.Date(), "%Y %b %d")
> myDate2
[1] "2020 Jan 06"
> class(myDate2)
[1] "character"
> mode(myDate2)
[1] "character"
```

Date Object

Other useful functions for "Date" objects

Example

```
> myDate <- as.Date("1970-01-01")
> months(myDate)
[1] "January"
> quarters(myDate)
[1] "Q1"
> weekdays(myDate)
[1] "Thursday"
> seq(from = myDate, to = myDate + 2, by = "day")
[1] "1970-01-01" "1970-01-02" "1970-01-03"
> myDate + 31
[1] "1970-02-01"
> seq(myDate, myDate + 31, by = "week")
[1] "1970-01-01" "1970-01-08" "1970-01-15" "1970-01-22" "1970-01-29"
```

Date and Time Objects

as.POSIXct() and **as.POSIXlt()** convert character to object of class "POSIXct" and "POSIXlt" respectively, representing date and time.

Example

```
> myDateTime1 <- as.POSIXct("01,01,2014 10:20:20",
+                           format = "%d,%m,%Y %H:%M:%S")
> myDateTime2 <- as.POSIXlt("01,01,2014 10:20:20",
+                           format = "%d,%m,%Y %H:%M:%S")
> mode(myDateTime1)
[1] "numeric"
> mode(myDateTime2)
[1] "list"
> class(myDateTime1)
[1] "POSIXct" "POSIXt"
> class(myDateTime2)
[1] "POSIXlt" "POSIXt"
```

Date and Time Objects

strptime() and **difftime()**

Example

```
> dt1 <- strptime("2013.12/23 01:00:34", "%Y.%m/%d %H:%M:%S")
> dt2 <- strptime("2013.12/23 01:02:37", "%Y.%m/%d %H:%M:%S")
> dt1
[1] "2013-12-23 01:00:34 EST"
> class(dt1)
[1] "POSIXlt" "POSIXt"
> dt1$sec # other elements: min, hour, mday, wday, yday, mon, year
[1] 34
> dt2 - dt1 # take difference
Time difference of 2.05 mins
> difftime(dt2, dt1, units = "secs")
Time difference of 123 secs
```

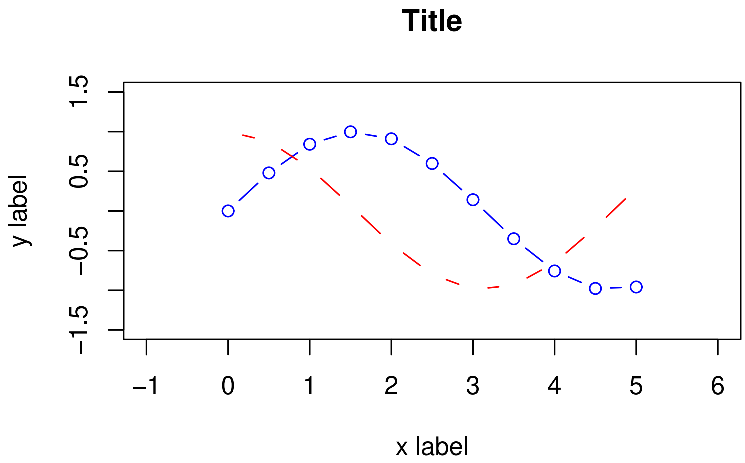
X-Y Plot

plot() function is the most basic graphic function in R, it doesn't need any package to start. And **lines()** function add line segments to the plot, **points()** function add points to the plot.

Example

```
> x <- seq(0, 5, by = 0.5)
> y <- sin(x)
> plot(x, y, type = "b", xlim = c(-1,6), ylim = c(-1.5, 1.5),
+      main = "Title", xlab = "x label", ylab = "y label", col = "blue")
> y2 <- cos(x)
> lines(x, y2, type = "c", col = "red")
```

X-Y Plot



X-Y Plot

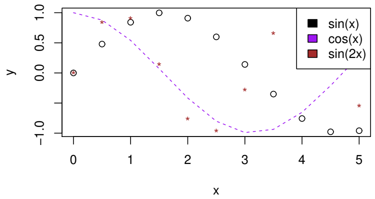
- main: title of the plot
- xlab,ylab: lable for x or y axis
- xlim,ylim: value range on x or y axis
- lwd: line width
- lty: type of line
- pch: type of point
- col: color for line or point

Type **?par** for different lty, and **?points** for different pch.

X-Y Plot

Example

```
y3 <- sin(2*x)
plot(x, y)
lines(x, y2, lty = "dashed", col = "purple")
points(x, y3, pch = "*", col = "brown")
legend("topright", c("sin(x)","cos(x)","sin(2x)"),
       fill = c("black","purple","brown"))
```

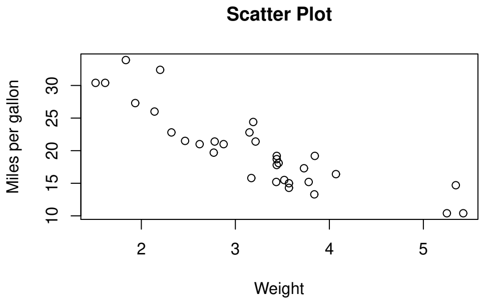


Scatter Plot

Scatter Plot of Miles per gallon against weight (weight in x-axis and Miles per gallon in y-axis)

Example

```
plot(mtcars$wt, mtcars$mpg, main = "Scatter Plot",
     xlab = "Weight", ylab = "Miles per gallon")
```

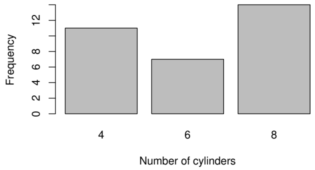


Bar Plot

Barplor of frequency against number of cylinders

Example

```
> table(mtcars$cyl)
 4  6  8
11  7 14
> barplot(table(mtcars$cyl),
+         xlab = "Number of cylinders", ylab = "Frequency")
```



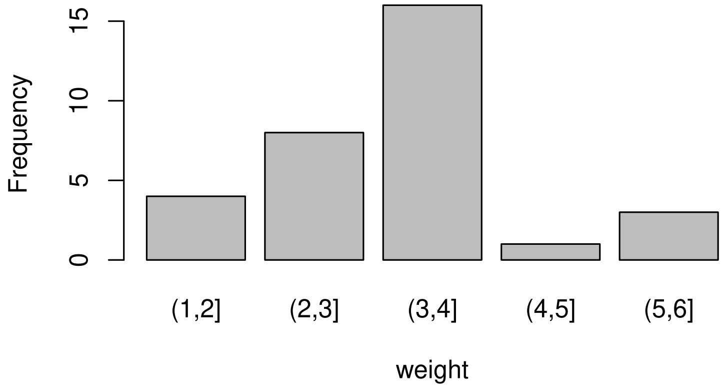
Bar Plot

Use **cut()** function divide data into intervals, then use **table()** function to generate tables, then use **barplot()** to plot.

Example

```
> cut(mtcars$wt, breaks = 1:6) # also use "breaks = 6" to get 6 intervals
 [1] (2,3] (2,3] (2,3] (3,4] (3,4] (3,4] (3,4] (3,4] (3,4] (3,4] (3,4] (4,5]
[14] (3,4] (5,6] (5,6] (5,6] (2,3] (1,2] (1,2] (2,3] (3,4] (3,4] (3,4] (3,4]
[27] (2,3] (1,2] (3,4] (2,3] (3,4] (2,3]
Levels: (1,2] (2,3] (3,4] (4,5] (5,6]
> table(cut(mtcars$wt, breaks = 1:6))
(1,2] (2,3] (3,4] (4,5] (5,6]
    4     8    16     1     3
> barplot(table(cut(mtcars$wt, breaks = 1:6)),
+         xlab = "weight", ylab = "Frequency")
```

Bar Plot



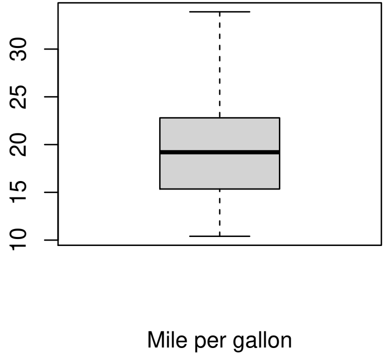
Box Plot

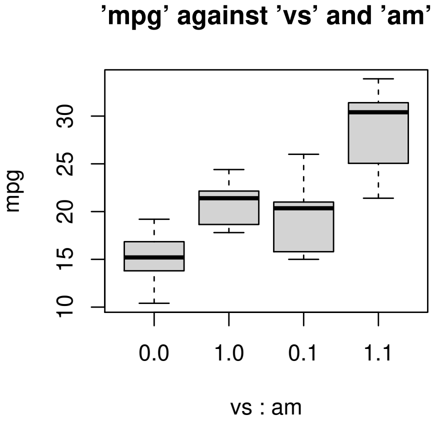
fivenum() finds the minimum, first quartile, median, third quartile, maximum. And **boxplot()** summarize these numbers in a plot.

Example

```
> fivenum(mtcars$mpg)
[1] 10.40 15.35 19.20 22.80 33.90
> # setup 2 figures in 1 row and 2 column which filled by rows
> par(mfrow = c(1, 2)) # how about mfcol?
> boxplot(mtcars$mpg, xlab = "Mile per gallon")
> # boxplot(mtcars$mpg ˜ mtcars$vs + mtcars$am,
> #         main = "'mpg' against 'vs' and 'am'")
> boxplot(mpg ˜ vs + am, data = mtcars,
+         main = "'mpg' against 'vs' and 'am'")
```

Box Plot





dev.off() can be used to clear all figures and reset all plot settings.