04 Basic Graphs

R Graphics

- R is capable of creating high quality graphics.
- Graphs are typically created using a series of high-level and low-level plotting commands.
    - High-level functions create new plots
    - Low-level functions add information to an existing plot.
- Customize graphs (line style, symbols, color, etc) by specifying graphical parameters
    - Specify graphic options using the `par()` function.
    - Can also include graphic options as additional arguments to plotting functions

The plot Function

- The most basic plotting function is called `plot`.
- It takes a set of x and y coordinates and produces a plot based on those coordinates.
- The simplest plot is a simple scatterplot, but optional arguments make it possible to produce a variety of different plot styles.

```
set.seed(1010)
xval <- 1:100
yval <- rnorm(100)
plot(xval, yval)
plot(yval, yval)
```

The `plot` Function

- Other optional arguments control features such as axis labelling and annotation.
- An optional argument called `pch` controls the plotting symbol(s) used by `plot`.
  - The argument can be either a single character, or a numerical index into a table of symbols.
  - Specifying an index of 0 produces an "invisible" symbol.
  - The `pch` argument is recycled to as many values as their are points to be plotted.
- Plotting symbol colors can be customised with the `col` argument.
  - Colors can be specified by name with character strings such as "`red`" and "`blue`".

```
plot(xval, yval, main="An Overall Title",
     xlab="A Label for the x-axis",
     ylab = "A Label for the y-axis")

plot(xval, yval, pch=19, col=2)
plot(xval, yval, pch=2, col=3)
plot(xval, yval, pch=6, col=4)

plot(xval, yval, pch="*", col="orange")
plot(xval, yval, pch="$", col="purple")
plot(xval, yval, pch="A", col="darkgreen")
plot(xval, yval, pch="B", col="darkred", cex=0.7)

cols <- sample(c("red", "green4", "blue"), length(xval),
               replace=TRUE)
plot(xval, yval, pch=16, col=cols)
```

Multiple Graphs

- To create a $n \times m$ grid of figures use `par()` with either the `mfcol` or `mfrow` settings.
  - `mfcol = c(nr, nc)` adds figures by column
  - `mfrow = c(nr, nc)` adds figures by row

```
par(mfrow=c(2,2))
plot(xval, yval, pch=15, col=cols, main="pch = 15")
plot(xval, yval, pch=17, col=cols, main="pch = 17")
plot(xval, yval, pch=18, col=cols, main="pch = 18")
plot(xval, yval, pch=21, col=cols, main="pch = 21")

par(mfrow=c(1,4))
plot(xval, yval, pch=19, col=4, main="col = 4")
plot(xval, yval, pch=19, col=5, main="col = 5")
plot(xval, yval, pch=19, col=6, main="col = 6")
plot(xval, yval, pch=19, col=7, main="col = 7")
```

Different Types of Plot

- By default, the `plot` function produces a scatterplot by drawing plotting symbols at the given $(x,\, y)$ coordinates.
- The optional `type` argument makes it possible to produce other types of plot by describing their content.
- The plot types are:

`"p"`	points (i.e. a scatterplot)
`"l"`	lines (i.e. a line plot
`"b"`	both (points and lines)
`"c"`	just the lines from `type="b"`
`"o"`	points and lines overplotted
`"h"`	high-density needles
`"s"`	step function, horizontal step first
`"S"`	step function, vertical step first
`"n"`	nothing (i.e. no plot contents)

Example

```
set.seed(12345)
x <- 1:20
y <- runif(20)
par(mfrow=c(3,3))
plot(x, y, type="p", main="type = \"p\"")
plot(x, y, type="l", main="type = \"l\"")
plot(x, y, type="b", main="type = \"b\"")
plot(x, y, type="c", main="type = \"c\"")
plot(x, y, type="o", main="type = \"o\"")
plot(x, y, type="h", main="type = \"h\"")
plot(x, y, type="s", main="type = \"s\"")
plot(x, y, type="S", main="type = \"S\"")
plot(x, y, type="n", main="type = \"n\"")
```

Line Types

- The line type can be specified with an argument of the form `lty=type`. The line type can be specified by either name or number:

1	"solid"	2	"dashed"	3	"dotted"
4	"dotdash"	5	"longdash"	6	"twodash"

```
set.seed(12345)
x <- 1:20; y <- runif(20)
par(mfrow=c(2,3))
plot(x, y, type="l", lty=1, main="lty = 1")
plot(x, y, type="l", lty=2, main="lty = 2")
plot(x, y, type="l", lty=3, main="lty = 3")
plot(x, y, type="l", lty=4, main="lty = 4")
plot(x, y, type="l", lty=5, main="lty = 5")
plot(x, y, type="l", lty=6, main="lty = 6")
```

Line Type and Thickness Control

- Line thickness can be set with lwd=1 where the greater the number, the thicker the line.
- The type, lwd, lty and col arguments can be used together to generate a variety of line effects.
- Note that in the following plot, the size of dot and the gap between them have both been scaled by the value of lwd.

```
set.seed(123)
y <- rnorm(10)
par(mfrow=c(3,1))
plot(y, type="l", lty=3, lwd=1, col="red",
     main="lwd = 1")
plot(y, type="l", lty=3, lwd=2, col="blue",
     main="lwd = 2")
plot(y, type="l", lty=3, lwd=3, col="orange",
     main="lwd = 3")
```

Controlling Axis Limits

- The plot arguments xlim and ylim supply two values which are use to determine the limits on the x and y axes.
- By default the range between the limits is expanded by 8% so that points near the edge of the plot do not overlap the edges.

```
set.seed(111)
y <- rnorm(50)
par(mfrow=c(3,1))
plot(y, pch=16, col="darkgreen", ylim=c(-3, 3),
     main="ylim = c(-3, 3)")
plot(y, pch=16, col="darkgreen", ylim=c(-5, 5),
     main="ylim = c(-5, 5)")
plot(y, pch=16, col="darkgreen", ylim=c(-10, 10),
     main="ylim = c(-10, 10)")
```

Scatterplot and Line Graphs

- Common arguments for `plot()`, see `par()` for a complete list

`type`	1-character string denoting the plot type
`xlim`	x limits, `c(x1, x2)`
`ylim`	y limits, `c(y1, y2)`
`main`	Main title for the plot
`sub`	Sub title for the plot
`xlab`	x-axis label
`ylab`	y-axis label
`col`	Color for lines and points
`pch`	Plotting symbol or a character string
`cex`	The character expansion of the plot symbols
`lty`	Line type
`lwd`	Line width

Adding to Plots

- Once a plot has been created it can be added to with a variety of low-level functions.

`points`	draw points
`lines`	draw connected line segments
`abline`	add a straight line to a plot
`segments`	draw disconnected line segments
`arrows`	add arrows to a plot
`rect`	add rectangles to a plot
`polygon`	add polygons to a plot
`text`	add text to a plot

Example: Two different groups

- If data has two or more than two groups, we can separate groups in a plot, using different `pch` and `col`.
  - Just use `plot` function with different values of `pch` and `col`.
  - Generate an empty `plot` and then use `points` function.
- The `points` argument can be
  - two separate vectors where one vector is the x-coordinates and the other is the y-coordinates,
  - a two-column matrix or
  - a two-element list with x and y components
- A plot `legend` can be added to explain different values of `pch` and `col`.

```r
set.seed(123)
x <- rnorm(50)
y <- rnorm(50)
group <- sample(0:1, 50, replace=TRUE)

par(mfrow=c(1,3))
plot(x, y, xlab="X", ylab="Y", main="Y vs X",
     pch=ifelse(group==1, 17, 19),
     col=ifelse(group==1, "green", "purple"))

plot(x,y, xlab="X", ylab="Y", main="Y vs X", type="n")
points(x[group==1], y[group==1], pch=17, col="green")
points(x[group==0], y[group==0], pch=19, col="purple")

plot(x,y, xlab="X", ylab="Y", main="Y vs X", type="n")
points(cbind(x,y)[group==1,], pch=17, col="green")
points(cbind(x,y)[group==0,], pch=19, col="purple")
legend("bottomright", c("group 1","group 0"), pch=c(17,19),
        col=c("green", "purple"), cex=1.5)
```

Example: Line Graphs

- Like points, the lines argument can be
  - two separate vectors where one vector is the x-coordinates and the other is the y-coordinates,
  - a two-column matrix or
  - a two-element list with x and y components
- If there is only one component then the argument is plotted against its index (same with `plot` and `points`)

```
set.seed(1010)
x <- runif(100, -3, 3)
y <- dnorm(x)
plot(x, y, type="l", lty=2, col=4)

oo <- order(x)
plot(x[oo], y[oo], type="l", lty=2, col=2)

plot(x, y, type="n")
lines(sort(x), y[oo], type="p", pch=19, col="orange")
lines(sort(x), y[oo], type="l", lty=2, col="purple")

plot(y, type="n")
lines(sort(y), type="b", pch=20, col="red")

plot(x, type="n")
lines(sort(x), type="l", lty=3, lwd=2, col="blue")
```

Example: Histograms

```
set.seed(1010)
x <- rnorm(1000)

# Basic Histogram
hist(x, main="Histogram of X", col="orange")

# Plot histogram along with a normal density
# Set freq=FALSE, so that the density histogram
# is plotted (area sums to 1)
hist(x, freq=FALSE, col="orange",
     main="Histogram with Normal Curve")
xpts <- seq(min(x), max(x), length.out=50)
ypts <- dnorm(xpts, mean=mean(x), sd=sd(x))
lines(xpts, ypts, lwd=2, col="blue")
```

Example: Boxplot

```
# Basic boxplot
par(mfrow=c(1,3))
boxplot(x, main="Boxplot of X", col="orange")
boxplot(x, main="Boxplot of X", border="orange", col=0)
boxplot(x, main="Boxplot of X", col="purple", boxwex=0.3)

# Side-by-Side Boxplots
set.seed(123)
wp <- sample(which(x > 0), 400)
wn <- sample(which(x < 0), 100)
group <- rep(0, 1000)
group[c(wp, wn)] <- 1
boxplot(x˜group, main="Boxplot of X by Group",
        names=c("Group 0", "Group 1"), xlab="",
        col=c("cyan", "pink"), boxwex=0.3)
```

Example: Density Curve

```
# Plot a 5th order polynomial
curve(3*x^5-5*x^3+2*x, from=-1.25, to=1.25, col="blue")

# Plot the gamma density
curve(dgamma(x, shape=2, scale=1), from=0, to=7, lwd=2,
      col="red")

# Plot multiple curves, notice that the first curve
# determines the range of the x-axis
curve(dnorm, from=-3, to=5, lwd=2, col="red")
curve(dnorm(x, mean=2), lwd=2, col="blue", add=TRUE)

# Add vertical lines at the means
lines(c(0, 0), c(0, dnorm(0)), lty=2, col="red")
lines(c(2, 2), c(0, dnorm(2, mean=2)), lty=2, col="blue")
```