

Lecture 19 - R Software

Ashutosh Rajput
Department of Mathematics

Rajdhani College
University of Delhi

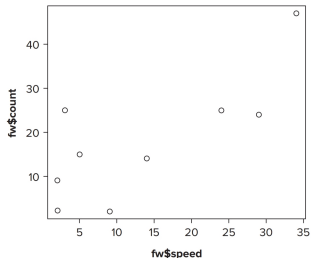
Basic Scatter plots

```
> fw
```

	count	speed
Taw	9	2
Torrige	25	3
Ouse	15	5
Exe	2	9
Lyn	14	14
Brook	25	24
Ditch	24	29
Fal	47	34

- We use \$ syntax to get at the variables, in case of data frame.

```
> plot(fw$speed, fw$count)
```



Adding Axis Labels

Axes can be labelled easily using the `xlab` and `ylab` instructions.

The command will be given as:

```
> plot(function$column1 name, function$column2 name, xlab = 'x-axis name',  
       ylab = 'y-axis name')
```

For example, to create labels for the above data you might use something like the following:

```
> plot(fw$speed, fw$count, xlab = 'Speed m/s', ylab = 'Count of Mayfly')
```

Plotting Symbols

The `pch` = instruction refers to the plotting character, by giving an integer value as:

```
> plot(function$column1 name, function$column2 name, pch = n)
```

where $n \in 1, 2, \dots, 25$

Note: `cex` = can be used to change the size of points.

Setting Axis Limits

We can set the limits of each axis by using `xlim` = and `ylim` = instructions as:

```
xlim = c(start, end)
```

```
ylim = c(start, end)
```

Adding Lines of Best-Fit to Scatter plots

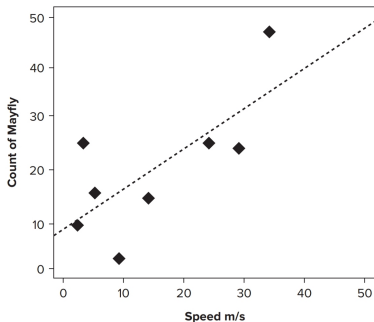
We use `abline()` command to add a straight line matching the slope and the intercept of a series of points when you produced a QQ plot.

```
> abline(lm(function$column1 name, function$column2 name))
```

Therefore, the above commands for the given example can be used as:

```
> plot(fw$speed, fw$count, xlab = 'Speed m/s', ylab = 'Count of Mayfly', pch = 18, cex = 2, xlim = c(0, 50), ylim = c(0, 50))
```

```
> abline(lm(fw$speed, fw$count), lty = 'dotted', lwd = 2, col = 'gray50')
```



Line Charts

If the data is a sequence but not numerical values, we have a trickier situation. For example, time interval might be recorded as month of the year.

```
> rain
```

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
3	5	7	5	3	2	6	8	5	6	9	8

```
> plot(rain, type = 'b')
```

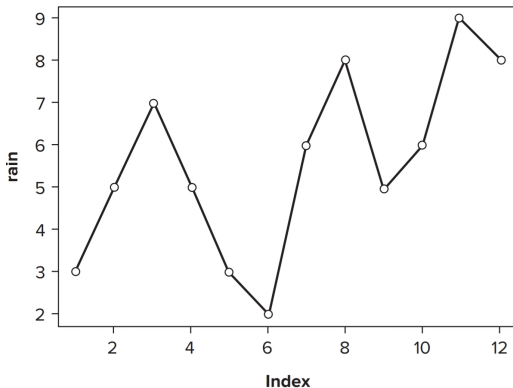


Table: The `type` = Instruction Can Alter the Way Data is Drawn on the Plot Area

Instruction	Explanation
<code>type = 'p'</code>	Points only .
<code>type = 'b'</code>	Points with line segments between .
<code>type = 'l'</code>	Lines segments alone with no points .
<code>type = 'o'</code>	Lines overplotted with points, that is, no gap between the line segments .
<code>type = 'c'</code>	Line segments only with small gaps where the points would be .
<code>type = 'n'</code>	Nothing is plotted! The graph is produced, setting axis scales but the data are not actually drawn in .

Pie Charts

If data that represents how something is divided up between various categories, the pie chart is a common graphic choice to illustrate the data. The pie chart is commonly used to display proportional data. You can create pie charts using the `pie()` command.

```
> data11
```

```
[1] 3 5 7 5 3 2 6 8 5 6 9 8
```

- `pie()` converts these values to the proportions of the total and then the angle of the pie slices is determined.

```
> data8
```

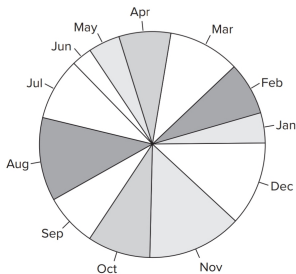
```
[1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct"
[11] "Nov" "Dec"
```

Remark: The slices are labeled with the names of the data.

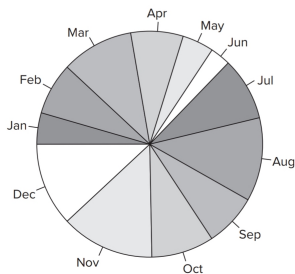
- The direction and starting point of the slices can be altered by using the `clockwise =` and `init.angle =` instructions.
- By default the slices are drawn counter-clockwise, so `clockwise = TRUE`, can set this to produce clockwise slices.

```
> pc = c('gray40', 'gray50', 'gray60', 'gray70', 'gray80', 'gray90')
```

```
> pie(data11, labels = data8, col = pc, clockwise = TRUE, init.angle = 180)
```



Counter-clockwise



Clockwise

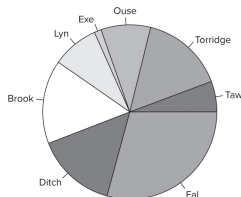
When the data are part of a data frame, we must use the `$` syntax to access the column we require.

```
> fw
```

	count	speed
Taw	9	2
Torrige	25	3
Ouse	15	5
Exe	2	9
Lyn	14	14
Brook	25	24
Ditch	24	29
Fal	47	34

```
> pc = c('gray65', 'gray70', 'gray75', 'gray80', 'gray85', 'gray90')
```

```
> pie(fw$count, labels = row.names(fw), col = pc, cex = 1.2)
```



The following data example shows a matrix of bird observation data; the rows and the columns are named:

```
> bird
```

	Garden	Hedgerow	Parkland	Pasture	Woodland
Blackbird	47	10	40	2	2
Chaffinch	19	3	5	0	2
Great Tit	50	0	10	7	0
House Sparrow	46	16	8	4	0
Robin	9	3	0	0	2
Song Thrush	4	0	6	0	0

```
> pie(bird[,1], col = pc)
```

