

The best way to learn Inference is to use it. We invite readers to work through the following familiarization session and see what happens. First-time users may not yet understand every detail, but the best plan is to type what you see and observe what happens as a result.

Notice: In order for this example to run properly, Inference requires that most recent versions of the following R packages be properly installed: lattice, MASS

Notice that we initialized the data parts container (the R environment) by calling two reference libraries (MASS for custom functions and lattice for Trellis graphics):

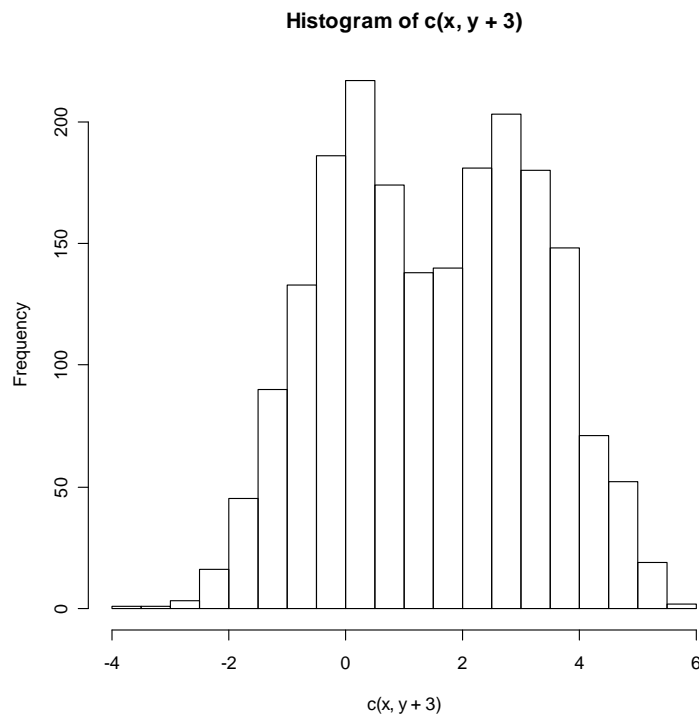
```
lattice.options(default.theme="col.whitebg")
```

Session A

Generate 1,000 pairs of normal variates. Create a histogram of a mixture of normal distributions. Experiment with the number of bins (25) and the shift (3) of the second component.

```
x <- rnorm(1000)
y <- rnorm(1000)

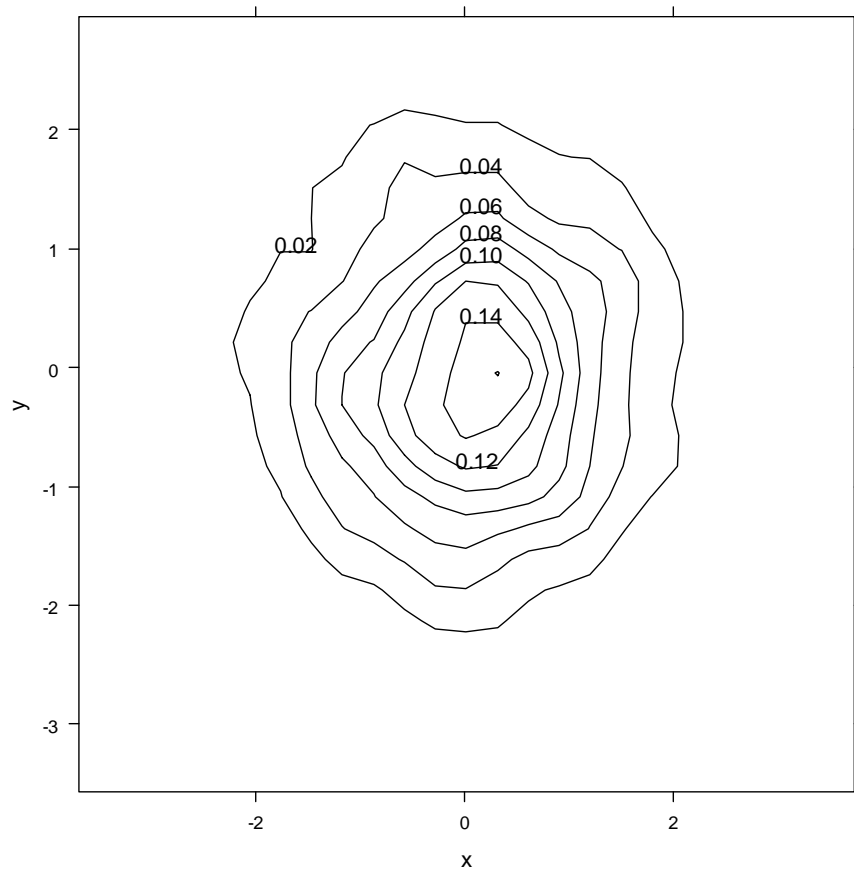
hist(
  c(x,y+3),
  breaks=25
)
```



Load the library needed for a custom function. Create a 2-D density plot. We convert its output into a form suitable for the Trellis visualization routines

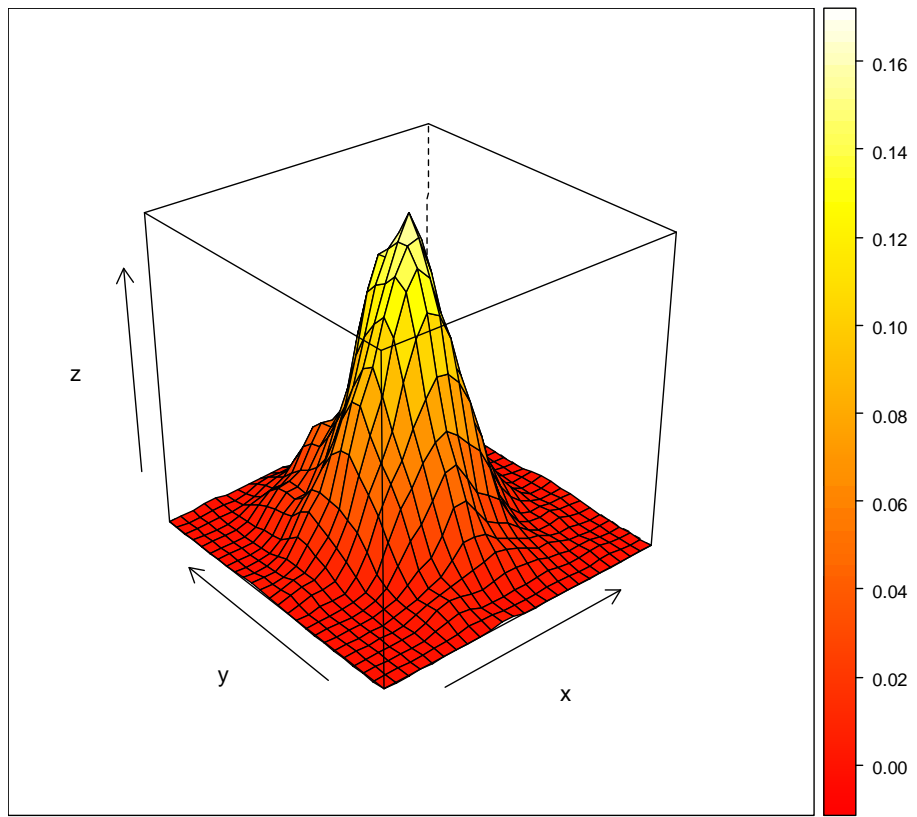
```
dd <- con2tr(kde2d(x,y))
```

```
contourplot(  
  z~x+y,  
  data=dd,  
  aspect=1  
)
```



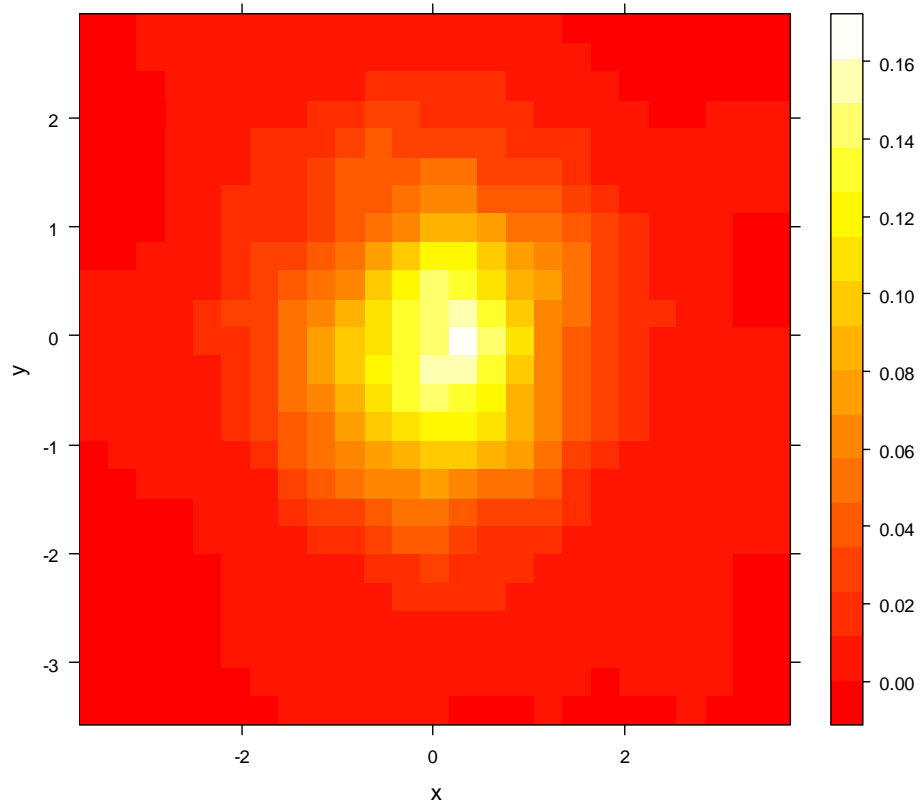
Create a perspective plot with superimposed color levels

```
wireframe(  
  z~x+y,  
  data=dd,  
  drape=TRUE  
)
```

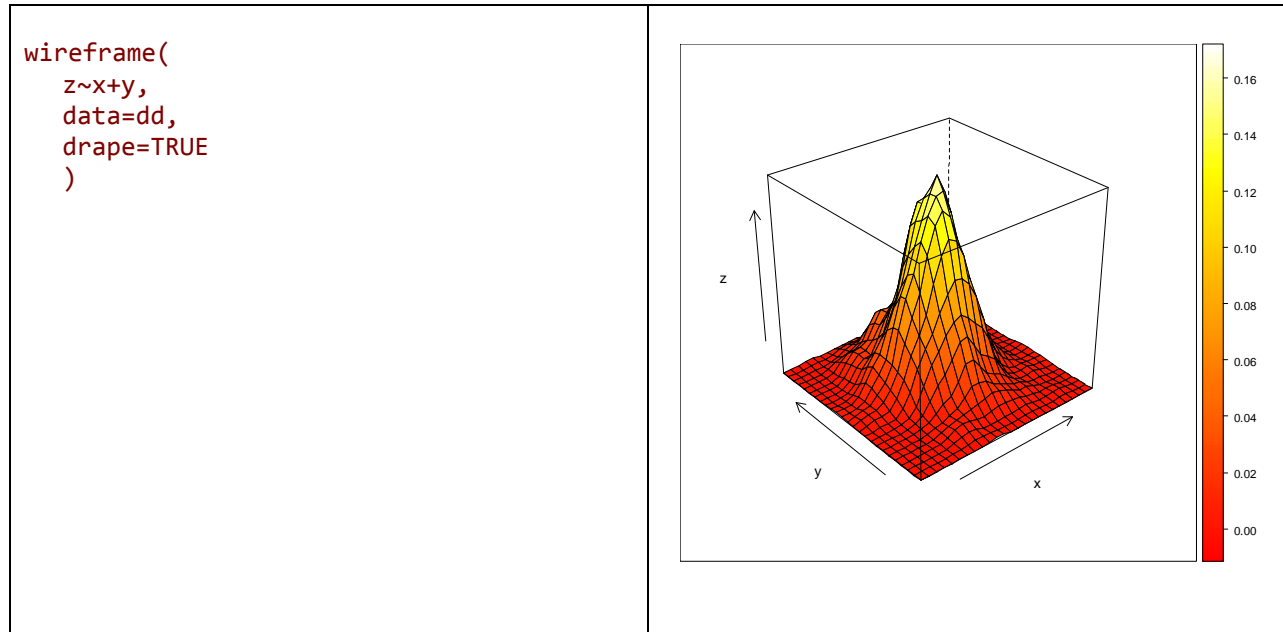


Finally, create a grayscale or pseudo-color plot

```
levelplot(  
  z~x+y,  
  data=dd,  
  aspect=1  
)
```



Code can be placed in a table. A code block can be identified through its "Label" reference. And, code can be output (displayed) but not executed (left cell) or not outputted but executed (right cell)



Session B

Do the following:

- make x a vector with the sequence $x = (1, 1.5, 2, \dots, 19.5, 20)$.
- Use w as a weight vector and to give the standard deviation of the errors
- make a data frame of three columns named x, y, and w. Remove the original x, y and w.
- Fit a simple linear regression of y on x and look at the analysis

```
x <- seq(from=1, to=20, by=0.5)
```

```
w <- 1 + x/2
```

```
y <- x + w*rnorm(x)
```

```
dum <- data.frame(x, y, w)
```

```
rm(x, y, w)
```

```
fm <- lm(
  y~x,
  data=dum
)
```

```
summary(fm)
```

```
Call:
lm(formula = y ~ x, data = dum)

Residuals:
    Min       1Q   Median       3Q      Max
-16.2228  -4.5819   0.5514   4.4452   8.6095

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.5412     2.0158  -1.261   0.215
x              1.3302     0.1692   7.861 2.09e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.947 on 37 degrees of freedom
Multiple R-Squared:  0.6255, Adjusted R-squared:  0.6154
F-statistic:  61.8 on 1 and 37 DF,  p-value: 2.086e-09
```

Since we know the standard deviations, we can do the weighted regression.

```
fm1 <- lm(
  y~x,
  data=dum,
  weight=1/w^2
)

summary(fm1)

Call:
lm(formula = y ~ x, data = dum, weights = 1/w^2)

Residuals:
    Min       1Q   Median       3Q      Max
-1.994263 -0.613331  0.005398  0.647614  1.657357

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.0115     0.9111  -1.110   0.274
x              1.1601     0.1458   7.959 1.56e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9894 on 37 degrees of freedom
Multiple R-Squared:  0.6313, Adjusted R-squared:  0.6213
F-statistic:  63.34 on 1 and 37 DF,  p-value: 1.558e-09
```

Do the following operations:

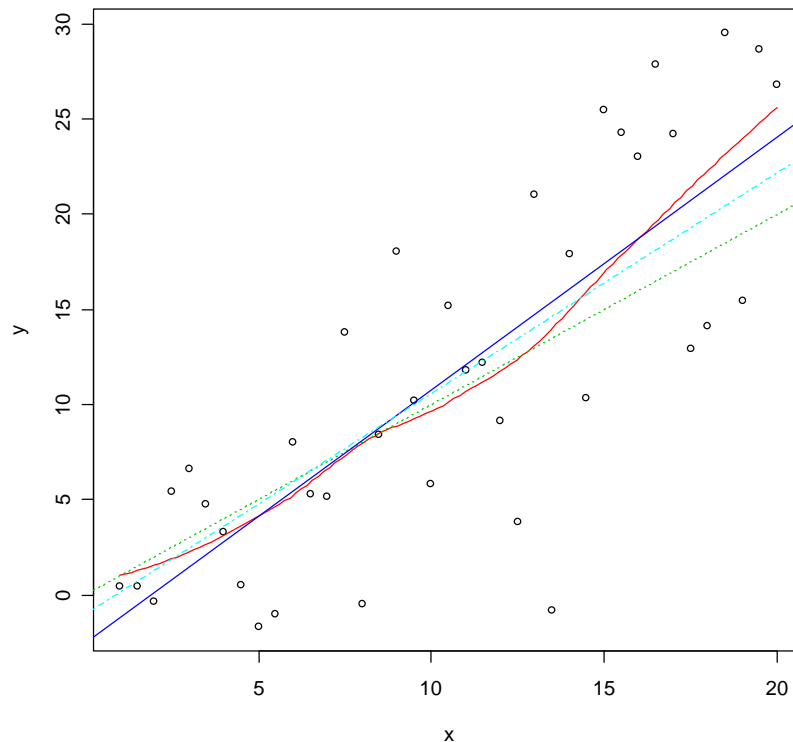
- Fit the smooth regression curve using a modern regression function (loess).
- Make the columns in the data frame visible as variables
- Make a standard scatterplot. To this plot we will add the three regression lines (or curves) as well as the known true line

- First add in the local regression curve using the spline interpolation between the calculated points
- Add in the true regression line (intercept 0, slope 1) with a different line type and color
- Add in the unweighted regression line. `abline()` is able to extract the information it needs from the fitted regression object
- Finally, add in the weighted regression line type 4. This one should be the most accurate estimate, but may not be, of course. The outcome is shown below.
- **IMPORTANT NOTE:** Since `core graphics` employs the layering of graphic objects onto the graphics device, script-based visualization requires that the sequence of graphic calls be combined into a single function, as illustrated below. This is, of course, what happens in the lattice graphics.

```
lrf <- loess(y~x, dum)
attach(dum)

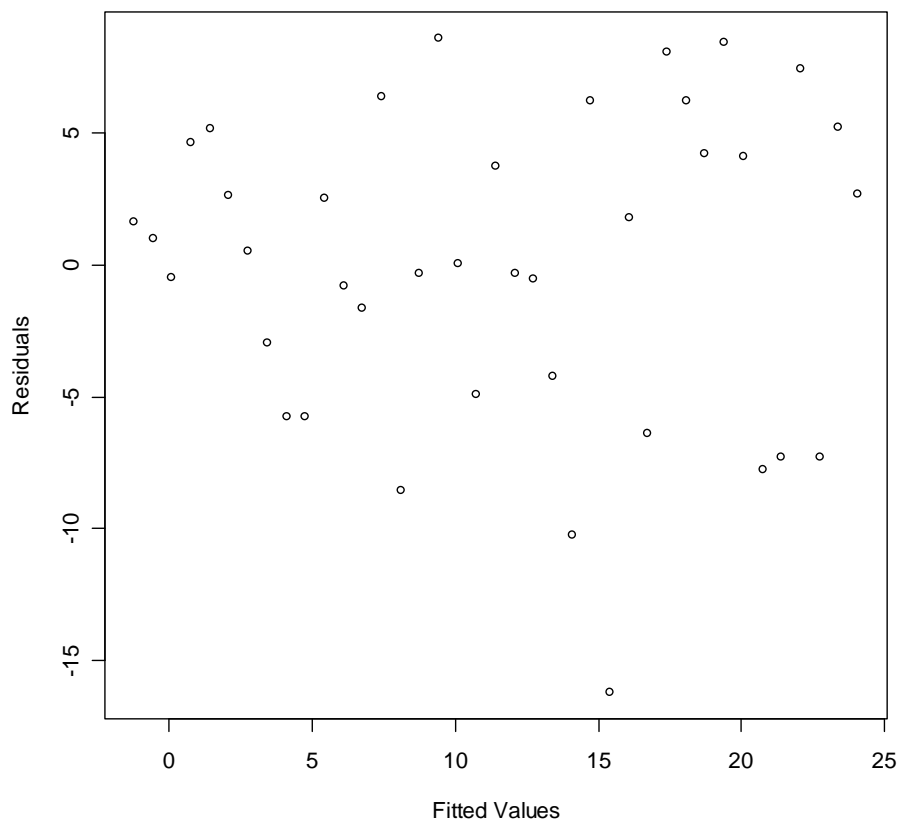
show <- function(x,y) {
  plot(x,y)
  lines(spline(x, fitted(lrf)), col=2)
  abline(0, 1, lty=3, col=3)
  abline(fm, col=4)
  abline(fm1, lty=4, col=5)
}

show(x,y)
```



A standard regression diagnostic plot to check for heteroscedasticity, that is, for unequal variances. The data are generated from a heteroscedastic process, so can you see this from the plot?

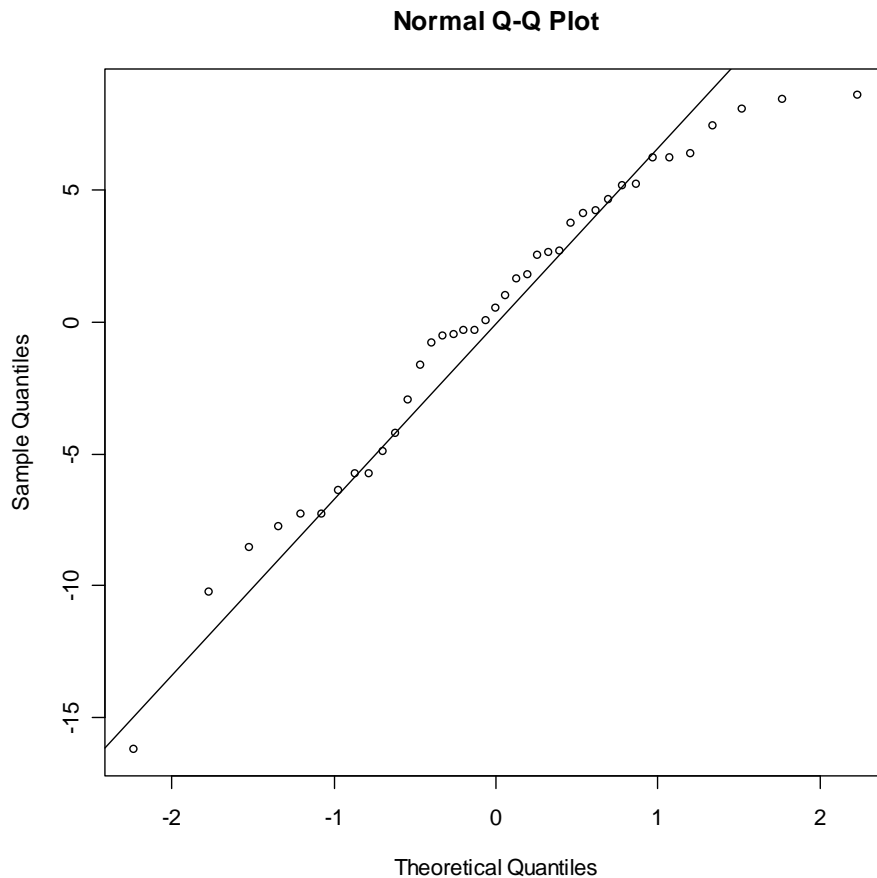
```
plot(
  fitted(fm),
  resid(fm),
  xlab="Fitted Values",
  ylab="Residuals"
)
```



A normal scores plot to check for skewness, kurtosis and outliers. (Note that the heteroscedasticity may show up as apparent non-normality.)

```
show <- function(x) {
  qqnorm(resid(fm))
  qqline(resid(fm))
}
```

```
show(x)
```



Session C

We now look at the data from the 1879 experiment of Michelson to measure the speed of light. There are five experiments (column Expt); each has 20 runs (column Run) and Speed is the recorded speed of light in km/sec, less 299,000. (The currently accepted value on this scale is 734.5).

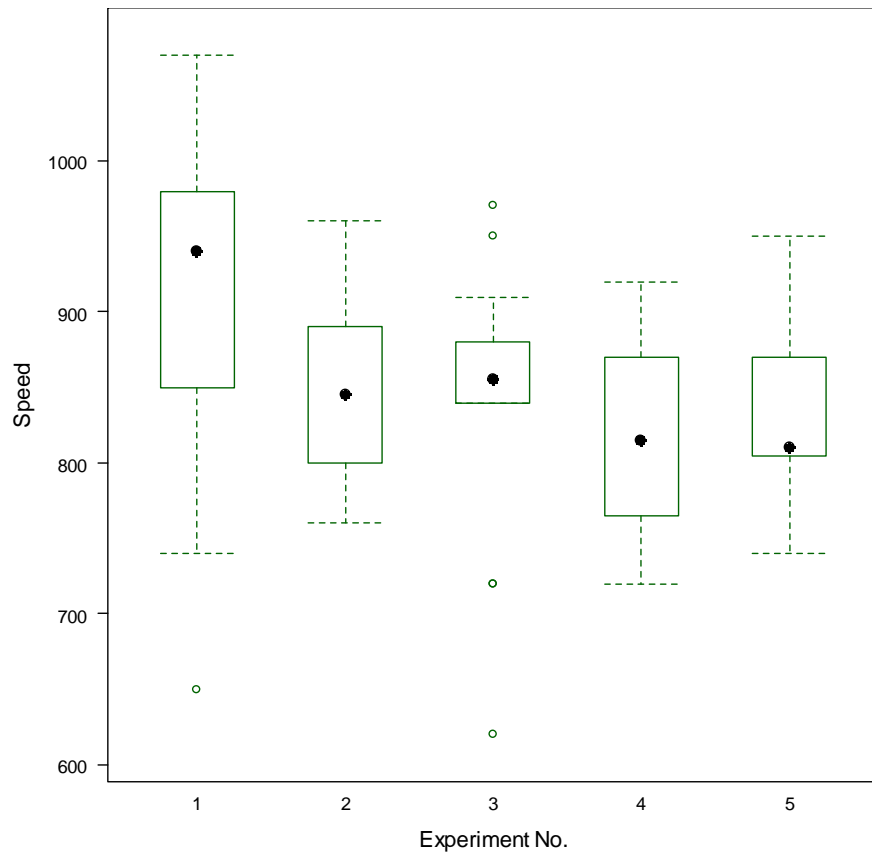
Initially, we will do the following:

- obtain the library that contains the experimental data (MASS)
- make the columns visible by name
- compare the five experiments with simple boxplots.

```
attach(michelson)
```

```
bwplot(
  Speed ~ Expt,
  main="Speed of Light Data",
  xlab="Experiment No."
)
```

Speed of Light Data



Analyze as a randomized block design, with runs and experiments as factors
Code Item:

```
fm <- aov(Speed ~ Run + Expt)
summary(fm)
```

```
Df Sum Sq Mean Sq F value Pr(>F)
Run 19 113344 5965 1.1053 0.363209
Expt 4 94514 23629 4.3781 0.003071 **
Residuals 76 410166 5397
```

```
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fit the sub-model omitting the nonsense factor, runs, and compare using a formal analysis of variance

```
fm0 <- update(fm, . ~ . - Run)
anova(fm0, fm)
```

Analysis of Variance Table

Model 1: Speed ~ Expt

Model 2: Speed ~ Run + Expt

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	95	523510				
2	76	410166	19	113344	1.1053	0.3632