An Instroduction to R Statistics

W. John Braun, UBC

Workshop - Quantitative Sciences Course Union

November 24, 2020

Box plots

The `rivers` data set contains the lengths of 141 important or major North American rivers. A quick numeric summary of these data is obtained through

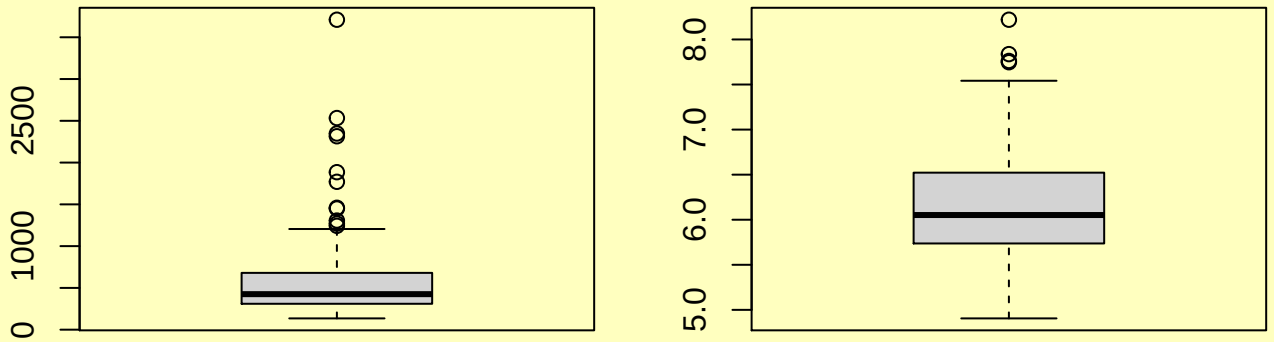
```
summary(rivers)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   135.0   310.0   425.0   591.2   680.0  3710.0
```

Box plots

A box plot, as shown in the left panel below, can be constructed using

```
boxplot(rivers)
```



Taking logs and then computing the box plot gives the graph in the right panel:

```
boxplot(log(rivers))
```

The result is much more symmetric; the histogram would be hard to distinguish from a normal distribution.

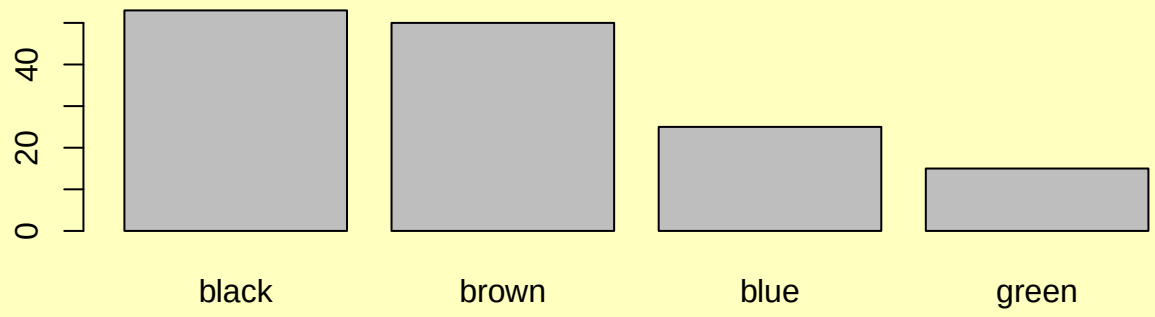
Eye Colour - Categorical Data

A sample of brown-haired males revealed the following eye colour counts:

black	brown	blue	green
53	50	25	15

A bar chart for these data is constructed using

```
eyecolourcounts <- c("black" = 53, "brown" = 50, "blue" = 25,
    "green" = 15) # assign data to eyecolourcounts with <-
barplot(eyecolourcounts)
```

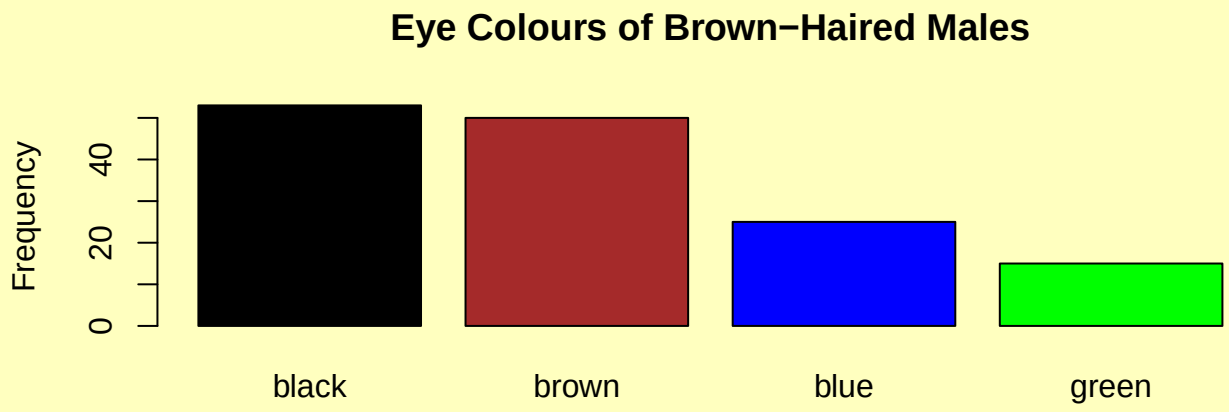


Eye Colour - Categorical Data

We can improve the graph in many ways.

We can add a title, a vertical axis label, and we might want to colour the bars:

```
barplot(eyecolourcounts,  col=names(eyecolourcounts),
    ylab="Frequency")
title("Eye Colours of Brown-Haired Males")
```



Consider the `cuckoos` data frame in the *DAAG* package.

```
library(DAAG)
```

The columns are

```
names(cuckoos)
```

```
## [1] "length"  "breadth" "species" "id"
```

```
summary(cuckoos)
```

```
##      length         breadth                species          id        
##  Min.   :19.60   Min.   :15.00   hedge.sparrow:14   Min.   : 21.00  
##  1st Qu.:21.90   1st Qu.:16.20   meadow.pipit :45   1st Qu.: 50.75  
##  Median :22.35   Median :16.60   pied.wagtail :15   Median : 81.00  
##  Mean   :22.45   Mean   :16.55   robin        :16   Mean   :103.88  
##  3rd Qu.:23.23   3rd Qu.:17.00   tree.pipit   :15   3rd Qu.:132.75  
##  Max.   :25.00   Max.   :17.50   wren         :15   Max.   :238.00
```

Tabulating and Graphing Categorical Data

We can count the number of observations for each species with the `table()` function:

```
table(cuckoos$species)
```

```
##
## hedge.sparrow  meadow.pipit  pied.wagtail          robin    tree.pipit
##            14            45            15            16            15
##          wren
##            15
```

Assigning this result to an object call `speciescounts` results in a named vector that can be used in a call to `barplot()`:

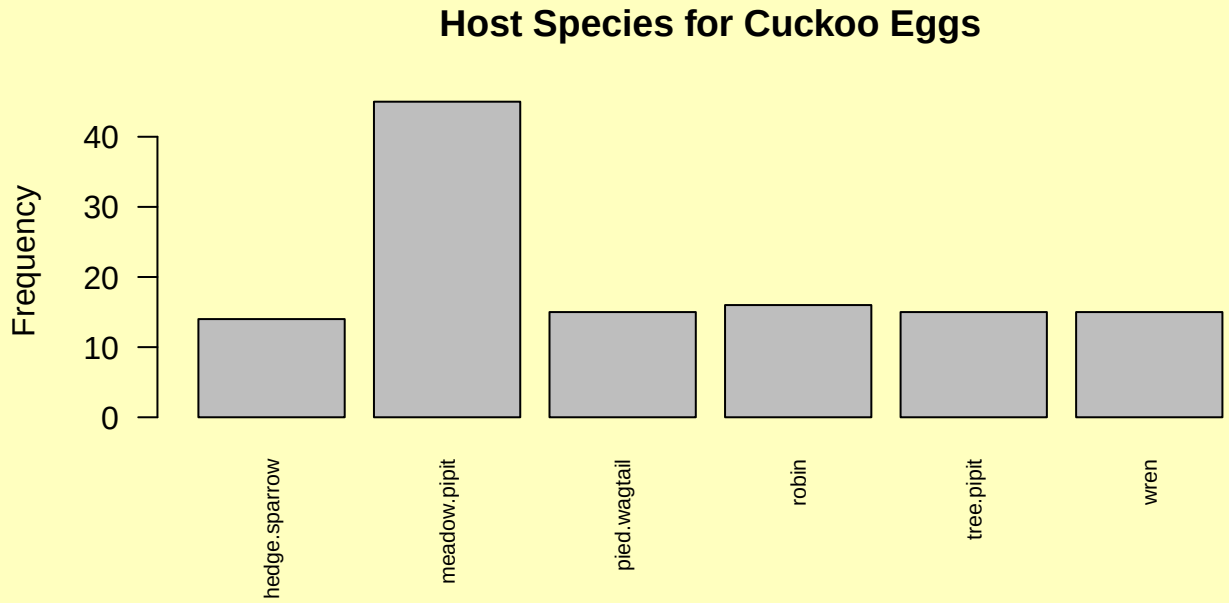
```
speciescounts <- table(cuckoos$species)
```

```
speciescounts
```

```
##
## hedge.sparrow  meadow.pipit  pied.wagtail          robin    tree.pipit
##            14            45            15            16            15
##          wren
##            15
```

Tabulating and Graphing Categorical Data

```
barplot(speciescounts, las=2, cex.names=.7, ylab="Frequency")
title("Host Species for Cuckoo Eggs")
```



The vertical tick labels were constructed for readability using the `las` argument and their size was controlled using `cex.names`.

T-tests

The goal of these tests, and the related confidence intervals, is to provide information about the mean of a single population, or about the difference in means of two population.

The critical assumption underlying the t-test is that the measurements are independent of each other.

We will use simulation to demonstrate the techniques.

One sample

We suppose that we have a random sample of measurements from a population with unknown mean μ and variance σ^2.

Without telling you how, I will simulate such 8 such measurements, storing them in an object called `X`, and we will use a test to determine if the true mean is 0 or not:

```
## [1]  2.83 -0.46  2.02  3.79  1.03  2.48  0.44  0.45
```

We can calculate the mean and standard deviation for this sample using the `mean()` and `sd` functions:

```
mean(X);  sd(X)
```

```
## [1] 1.572745
## [1] 1.43926
```

One sample test

Clearly, the sample mean is not 0, but the true mean could still be 0, and this result could just be the result of random sampling error. The t-test helps us answer this question:

```
t.test(X, conf.level = .995)


##
##  One Sample t-test
##
## data:  X
## t = 3.0908, df = 7, p-value = 0.01755
## alternative hypothesis: true mean is not equal to 0
## 99.5 percent confidence interval:
##  -0.4776044  3.6230947
## sample estimates:
## mean of x
##  1.572745
```

One sample test

The small p-value indicates strong evidence against the hypothesis that the true mean is 0. In fact, this assertion is correct, since the code used to generate the random sample was as follows:

```
X <- rnorm(8, mean = 1.5) # true mean is 1.5
```

Note that we have used a 99.5% confidence interval to estimate the mean. This differs from the usual 95% that you might have been told to use.

Two samples - matched pairs

If there is a one-to-one correspondence between measurements in one of the samples with measurements in the other sample, then the appropriate way to compare the means is by taking the differences, and running a one-sample test on the differences.

This can be done with the `paired` option in the `t.test()` function.

Let's suppose L is a set of left foot lengths (in cm) for a sample of 15 adult males, and R contains the corresponding right foot lengths.

We would be interested in any systematic difference in the lengths of the feet.

Two samples - matched pairs

A simulation model for the case where there is no difference could be the following:

```
L <- rnorm(15, mean =28, sd = 1)
R <- L + rnorm(15, mean = 0, sd = .03)
```

Here we have assumed that the left feet are normally distributed with a mean of 28 cm and a standard deviation of 1 cm.

The right feet lengths are not exactly equal to the left feet lengths, but on average there is no difference.

The standard deviation of the difference is small.

Let's see what the t-test says:

```
t.test(L, R, paired=TRUE)

##
##  Paired t-test
##
## data:  L and R
## t = -0.4055, df = 14, p-value = 0.6912
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.01798248  0.01226399
## sample estimates:
## mean of the differences
##            -0.002859246
```

The p-value is large indicating that there is no evidence of a difference, in line with the truth.

The yield (y, in kg/plot) was measured for various salinity concentrations (x, measured in units of electrical conductivity).

18 measurements were recorded in a file called *tomato.txt* whose contents summarized below:

```
##     salinity          yield
##  Min.   : 1.600   Min.   :41.00
##  1st Qu.: 2.150   1st Qu.:48.58
##  Median : 4.900   Median :53.05
##  Mean   : 5.456   Mean   :52.44
##  3rd Qu.: 9.150   3rd Qu.:56.40
##  Max.   :10.200   Max.   :63.10
```

The first column contains the salinity concentration levels, and the second column contains the yield measurements.

We read these data into R using the `read.table()` function (or using a menu option in RStudio):

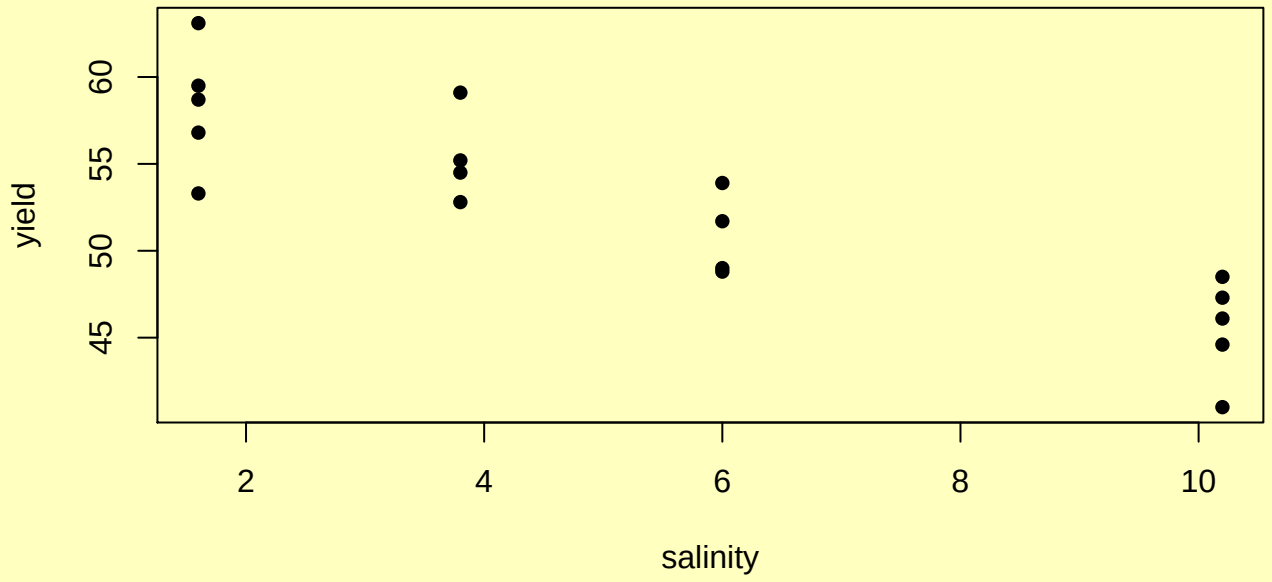
```
tom <- read.table("tomato.txt", header=FALSE)
```

Since there is no header, we should apply some sensible names to the data frame:

```
names(tom) <- c("salinity", "yield")
```

We next construct a *scatterplot* of the data to look for patterns and outliers.

```
plot(yield ~ salinity, data = tom, pch=16)
```



The scatterplot gives an indication of a clear downward trend as salinity increases.

The trend is also vaguely linear.

We can investigate this with the `lm()` function:

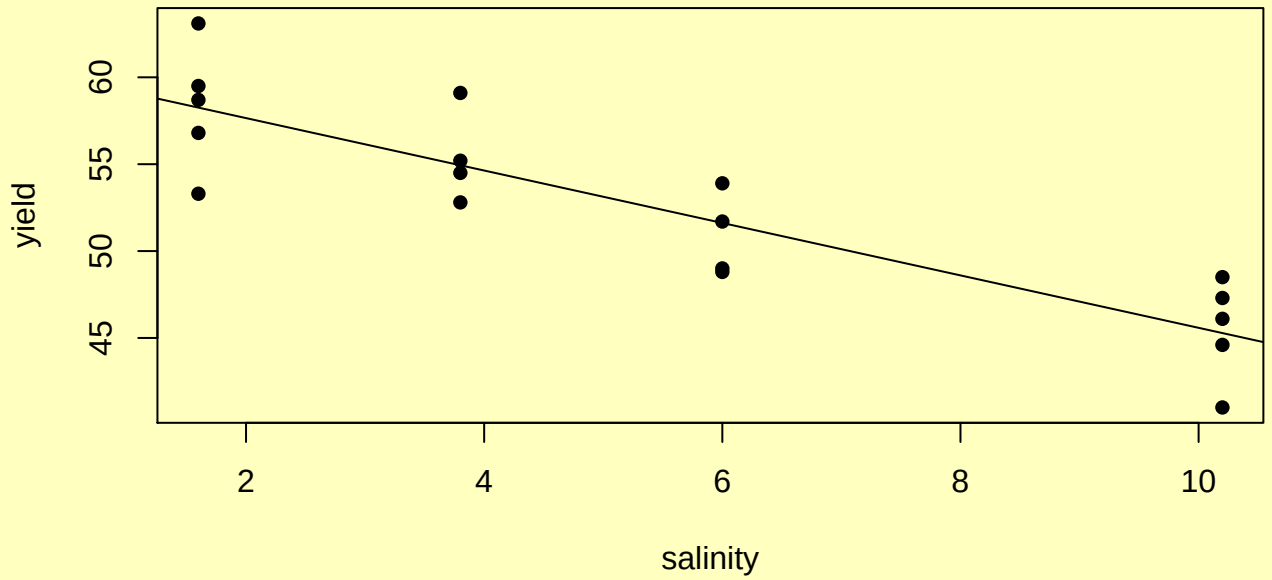
```
tom.lm <- lm(yield ~ salinity, data = tom)
```

We can explore the output from the fitted model using the `summary()` function:

```
summary(tom.lm)

##
## Call:
## lm(formula = yield ~ salinity, data = tom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9560 -1.9665  0.1729  1.8255  4.8440
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  60.6700     1.2807  47.372  < 2e-16
## salinity     -1.5088     0.2005  -7.527 1.21e-06
##
## Residual standard error: 2.828 on 16 degrees of freedom
## Multiple R-squared:  0.7798,Adjusted R-squared:  0.766
## F-statistic: 56.65 on 1 and 16 DF,  p-value: 1.212e-06
```

We can overlay the scatterplot of the data with the fitted line using the `abline()` function and the output from the `lm()` function:

```
abline(tom.lm)
```



One factor ANOVA

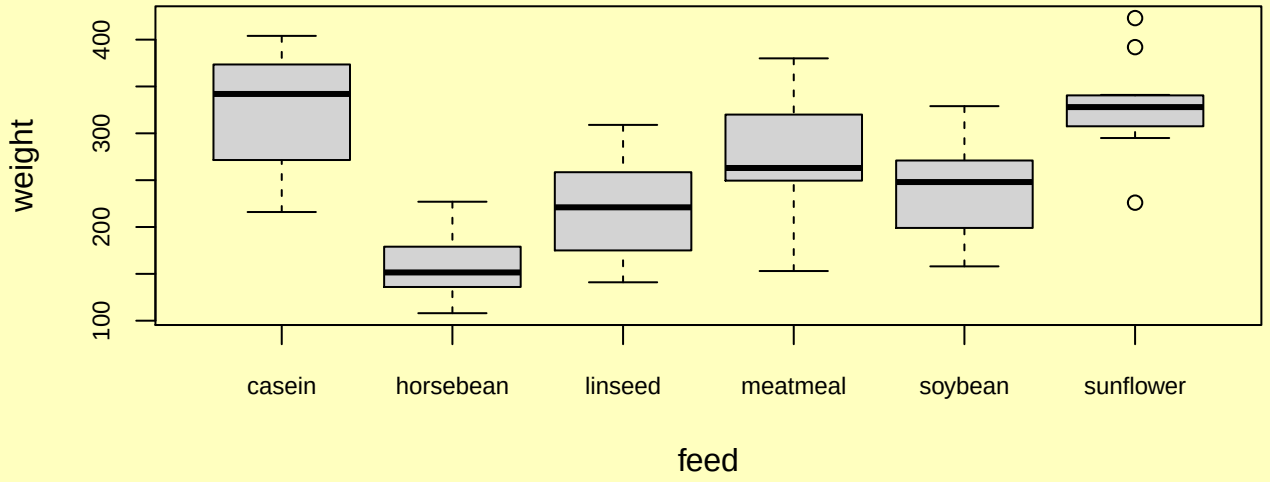
The `chickwts` data frame contains measurements of the weights of chicks who have been randomly assigned to groups, each of which has been given a different type of feed.

It is of interest to know whether the different feed types lead to systematic differences in weight.

We refer to feed type as a *factor* having different *levels* representing the particular kinds of feed, e.g. linseed, horsebean, and so on.

Side-by-side *boxplots*, as displayed below, are a useful way to visualize these data.

```
plot(weight ~ feed, data = chickwts, cex.axis=.75)
```



Because `feed` is a factor, `plot()` yields box plots.

From the graph, it seems that horsebean leads to lower weights than some of the other feed types.

It is hard to tell for sure if there is variability between treatments because of the variability within treatments, that is noise due to unmeasured factors.

We can test whether there is a difference in the mean weights statistically with the analysis of variance (ANOVA).

A general purpose procedure is as follows:

```
chick.lm <- lm(weight ~ feed, data = chickwts)
anova(chick.lm)

## Analysis of Variance Table
##
## Response: weight
##           Df Sum Sq Mean Sq F value    Pr(>F)
## feed       5 231129   46226  15.365 5.936e-10
## Residuals 65 195556    3009
```

One factor ANOVA

The test statistic compares the variability in the averages with the variability in the noise through an F-statistic.

A p-value is computed which gives the strength of evidence against the *null hypothesis*, that is the hypothesis that there is no difference in the means.

A small p-value – and in this case, it is very small – indicates strong evidence against the null hypothesis, in favour of the alternative that there is a difference.

The data frame `table.b4` in the *MPV* library contains the following columns:

```
y sale price of the house (in thousands of dollars)
x1 taxes (in thousands of dollars)
x2 number of baths
x3 lot size (in thousands of square feet)
x4 living space (in thousands of square feet)
x5 number of garage stalls
x6 number of rooms
x7 number of bedrooms
x8 age of the home (in years)
x9 number of fireplaces
```

There are 24 observations on these *variables* in the data frame.

A natural question to ask is whether any or all of the given variables or *covariates* could be used to predict the sale price of a house.

We consider a linear model of the form

$$y = \beta_0 + \sum_{j=1}^{9} \beta_j x_j + \varepsilon.$$

The elements of ε are assumed to be uncorrelated random variables with mean 0 and common variance σ^2.

Fitting the model

The *lm()* function will take care of the coefficient estimation, variance estimation, t and F and p-value calculations in one function call.

For example, if we want to relate house price, y to x_1 and x_4,

```
house.lm <- lm(y ~ x1 + x4, data=table.b4)
```

We can view the output from this, using the `summary` function as in

```
summary(house.lm)
```

Estimating and predicting

The model can now be used to estimate the expected house price for houses with x_1 taxes and x_4 amount of living space using the formula

$$\widehat{y} = 11.5 + 2.92x_1 + 3.15x_4.$$

This can be accomplished in R using the `predict()` function.

For instance, suppose we want to estimate the mean sale price for homes with $2000 taxes and 3000 square feet of living area. Use

```
predict(house.lm, newdata = data.frame(x1 = 2, x4 = 3))
```

```
##        1
## 26.85605
```

Estimating and predicting

For interval estimation, use `interval = "confidence"` for estimation and `interval = "predict"` for prediction. For interval estimation of a prediction:

```
predict(house.lm, newdata = data.frame(x1 = 2, x4 = 3),
     interval = "predict")

##        fit      lwr      upr
## 1 26.85605 10.23286 43.47925
```

Interval estimation for the mean:

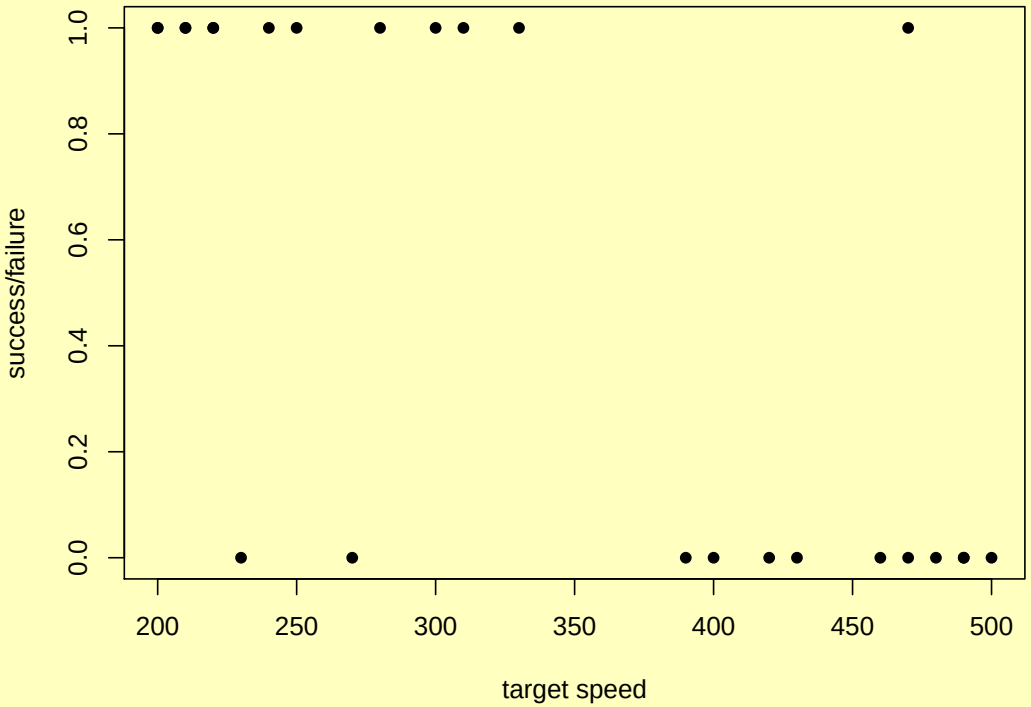
```
predict(house.lm, newdata = data.frame(x1 = 2, x4 = 3),
     interval = "confidence")

##        fit      lwr      upr
## 1 26.85605 11.42015 42.29196
```

The data in `p13.1` in the *MPV* package describes successes and failures of surface-to-air missiles as they relate to target speed.

Such *binary data* are not nearly normally distributed, so it the efficacy of least-squares becomes very questionable here.

We now suggest some of the R functions that can be used to begin to analyze such data.

The data are plotted to the right, with successes on the vertical axis being represented by a '1' and failures being represented by a '0'.



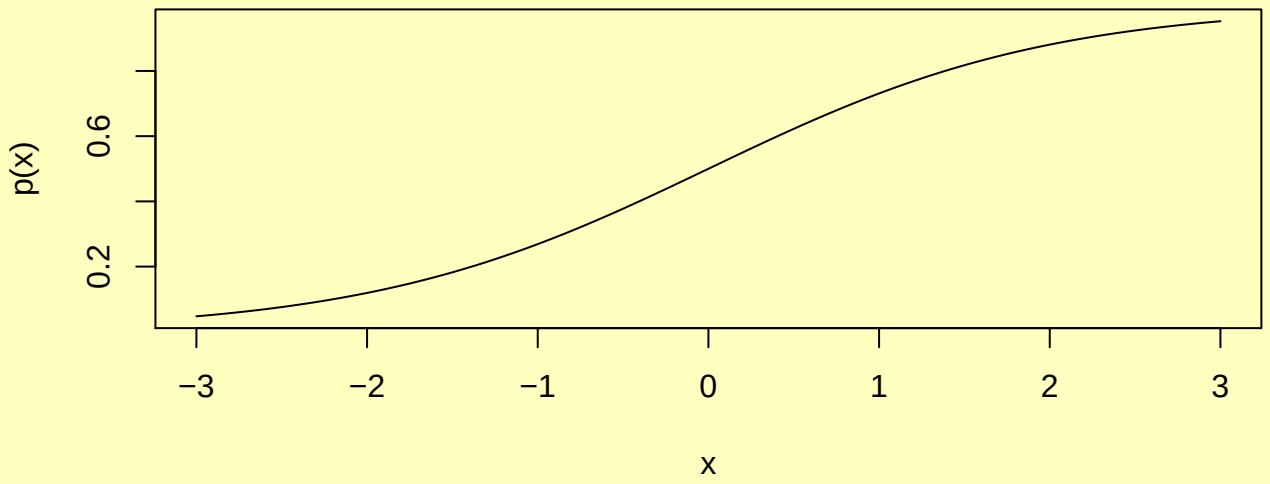

```
library(MPV) #  contains the p13.1 data frame
plot(p13.1, xlab = "target speed", ylab = "success/failure",
    pch=16) # produces a different plotting character
```

Perhaps the most popular function for this purpose is the *logistic* function

$$p(x) = \frac{e^x}{e^x + 1}.$$

The function is sketched below.

```
curve(exp(x)/(1 + exp(x)), from = -3, to = 3, ylab="p(x)")
```



The inverse of the logistic function is logit function:

$$\ell(p) = \log\left(\frac{p}{1-p}\right).$$

While p is restricted to take values between 0 and 1, the logit function can take any possible value, so relating the logit function to a straight line or other linear combination is a possibility. That is,

$$\ell(p(x)) = \beta_0 + \beta_1 x$$

which means that we can express the probability of an event in terms of a covariate x.

$\ell()$ is an example of a *link* function.

To fit the logistic regression model to the missile success data, try

```
p13.glm <- glm(y ~ x, data = p13.1, family = binomial)
```

To see the results, try

```
summary(p13.glm)
```

Note that we did not specify the link function; the default choice with the binomial family is the logit.

If we only want to see the coefficients and their standard errors, try

```
summary(p13.glm)$coefficients

##                Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept)  6.0708839 2.108996265  2.878566 0.003994883
## x           -0.0177047 0.006075513 -2.914107 0.003567073
```

This output tells us that the logit of the probability of success as a linear function of target speed has intercept 6.07 and slope -.0177.

Standard error estimates for these parameter estimates are supplied and indicate, in particular, that the slope is clearly negative.

The logistic curve can now be used to calculate probabilities of success at the various speeds.

We will use the `predict()` function with `type = "response"`.

If `type` is not specified, the default is to use the predictions on the linear scale.

Modelling binary responses with logistic regression

```
plot(p13.1, pch=16, xlab = "target speed",
     ylab = "success/failure")
newspeeds <- 200:500 # speeds at which we can predict
lines(newspeeds, predict(p13.glm, newdata=
    data.frame(x = newspeeds),  type = "response"))
```