Part VI
Populations & Samples – Theoretical & Empirical Distributions

R functions that will be used in this laboratory include:

(a) `dnorm()`: Obtain the density values for the theoretical normal distribution;

(b) `pnorm()`: Given a normal deviate or deviates, obtain the cumulative probability;

(c) `qnorm()`: Given the cumulative probabilty. calculate the normal deviate;

(d) `sample()`: take a sample from a vector of values. Values may be taken without replacement (once taken from the vector, the value is not available for subsequent draws), or with replacement (values may be repeated);

(e) `density()`: fit an empirical density curve to a set of values;

(f) `rnorm()`: Take a random sample from a theoretical normal distribution;

(g) `runif()`: similar to `rnorm()`, but sampling is from a uniform distribution;

(h) `rt()`: similar to `rnorm()`, but sampling is from a t-distribution (the degrees of freedom must be given as the second parameter);

(i) `rexp()`: similar to `rnorm()`, but sampling is from an exponential distribution;

(j) `qqnorm()`: Compare the empirical distribution of a set of values with the empirical normal distribution.

1 Populations and Theoretical Distributions

Exercise 1

(a) Plot the density and the cumulative probability curve for a normal distribution with a mean of 2.5 and SD = 1.5.
Code that will plot the curve is

```
> curve(dnorm(x, mean = 2.5, sd = 1.5), from = 2.5 - 3 * 1.5, to = 2.5 +
+     3 * 1.5)
> curve(pnorm(x, mean = 2.5, sd = 1.5), from = 2.5 - 3 * 1.5, to = 2.5 +
+     3 * 1.5)
```

(b) From the cumulative probability curve in (a), read off the area under the density curve between `x=0.5` and `x=4`. Check your answer by doing the calculation

```
> pnorm(4, mean = 2.5, sd = 1.5) - pnorm(0.5, mean = 2.5, sd = 1.5)

[1] 0.7501335
```

Exercise 1, continued

(a) The density for the distribution in items (i) and (ii), given by `dnorm(x, 2.5, 1.5)`, gives the relative number of observations per unit interval that can be expected at the value `x`. For example `dnorm(x=2, 2.5, 1.5)` $\simeq 0.2516$. Hence

(i) In a sample of 100 the expected number of observations per unit interval, in the immediate vicinity of `x = 2`, is 25.16

(ii) In a sample of 1000 the expected number of observations per unit interval, in the immediate vicinity of `x = 2`, is 251.6

(iii) The expected number of values from a sample of 100, between 1.9 amd 2.1, is approximately $0.2 \times 251.6 = 50.32$

[The number can be calculated more exactly as
`(pnorm(2.1, 2.5, 1.5) - pnorm(1.9, 2.5, 1.5)) * 1000`]

Repeat the calculation to get approximate and more exact values for the expected number

(i) between 0.9 and 1.1

(ii) between 2.9 and 3.1

(iii) between 3.9 and 4.1

By way of example, here is the code for (a):

```
> curve(dnorm(x, mean = 2.5, sd = 1.5), from = 2.5 - 3 * 1.5, to = 2.5 +
+     3 * 1.5)
> curve(pnorm(x, mean = 2.5, sd = 1.5), from = 2.5 - 3 * 1.5, to = 2.5 +
+     3 * 1.5)
```

Exercise 2

(a) Plot the density and the cumulative probability curve for a t-distribution with 3 degrees of freedom. Overlay, in each case, a normal distribution with a mean of 0 and SD=1.
[Replace `dnorm` by `dt`, and specify `df=10`]

(b) Plot the density and the cumulative probability curve for an exponential distribution with a rate parameter equal to 1 (the default). Repeat, with a rate parameter equal to 2. (When used as a failure time distribution; the rate parameter is the expected number of failures per unit time.)

2 Samples and Estimated Density Curves

Exercise 3
Use the function `rnorm()` to draw a random sample of 25 values from a normal distribution with a mean of 0 and a standard deviation equal to 1.0. Use a histogram, with `probability=TRUE` to display the values. Overlay the histogram with: (a) an estimated density curve; (b) the theoretical density curve for a normal distribution with mean 0 and standard deviation equal to 1.0. Repeat with samples of 100 and 500 values, showing the different displays in different panels on the same graphics page.

```
> par(mfrow = c(1, 3), pty = "s")
> x <- rnorm(50)
> hist(x, probability = TRUE)
```

```
> lines(density(x))
> xval <- pretty(c(-3, 3), 50)
> lines(xval, dnorm(xval), col = "red")
```

Exercise 4
Data whose distribution is close to lognormal are common. Size measurements of biological organisms often have this character. As an example, consider the measurements of body weight (`body`), in the data frame `Animals` (*MASS*). Begin by drawing a histogram of the untransformed values, and overlaying a density curve. Then

(a) Draw an estimated density curve for the logarithms of the values. Code is given immediately below.

(b) Determine the mean and standard deviation of `log(Animals$body)`. Overlay the estimated density with the theoretical density for a normal distribution with the mean and standard deviation just obtained.

Does the distribution seem normal, after transformation to a logarithmic scale?

```
> library(MASS)
> plot(density(Animals$body))
> logbody <- log(Animals$body)
> plot(density(logbody))
> av <- mean(logbody)
> sdev <- sd(logbody)
> xval <- pretty(c(av - 3 * sdev, av + 3 * sdev), 50)
> lines(xval, dnorm(xval, mean = av, sd = sdev))
```

Exercise 5
The following plots an estimated density curve for a random sample of 50 values from a normal distribution:

```
> plot(density(rnorm(50)), type = "l")
```

(a) Plot estimated density curves, for random samples of 50 values, from (a) the normal distribution; (b) the uniform distribution (`runif(50)`); (c) the t-distribution with 3 degrees of freedom. Overlay the three plots (use `lines()` in place of `plot()` for densities after the first).

(b) Repeat the previous exercise, but taking random samples of 500 values.

Exercise 6
There are two ways to make an estimated density smoother:

(a) One is to increase the number of samples, For example:

```
> plot(density(rnorm(500)), type = "l")
```

Exercise 6, continued

(b) The other is to increase the bandwidth. For example

```
> plot(density(rnorm(50), bw = 0.2), type = "l")
> plot(density(rnorm(50), bw = 0.6), type = "l")
```

Repeat each of these with bandwidths (`bw`) of 0.15, with the default choice of bandwidth, and with the bandwidth set to 0.75.

Exercise 7

Here we experiment with the use of `sample()` to take a sample from an empirical distribution, i.e., from a vector of values that is given as argument. Here, the sample size will be the number of values in the argument. Any size of sample is however permissible.

```
> sample(1:5, replace = TRUE)
> for (i in 1:10) print(sample(1:5, replace = TRUE))
> plot(density(log10(Animals$body)))
> lines(density(sample(log10(Animals$body), replace = TRUE)), col = "red")
```

Repeat the final density plot several times, perhaps using different colours for the curve on each occasion. This gives an indication of the stability of the estimated density curve with respect to sample variation.

3 *Normal Probability Plots

Exercise 8

Partly because of the issues with bandwidth and choice of kernel, and partly because it is hard to density estimates are not a very effective means for judging normality. A much better tool is the normal probability plot, which works with cumulative probability distributions. Try

```
> qqnorm(rnorm(10))
> qqnorm(rnorm(50))
> qqnorm(rnorm(200))
```

For samples of modest to large sizes, the points lie close to a line.
The function `qreference()` (*DAAG*) takes one sample as a reference (by default it uses a random sample) and by default provides 5 other random normal samples for comparison. For example:

```
> library(DAAG)
> qreference(m = 10)
> qreference(m = 50)
> qreference(m = 200)
```

Exercise 9

The intended use of `qreference()` is to draw a normal probability for a set of data, and place alongside it some number of normal probability plots for random normal data. For example

```
> qreference(possum$totlngth)
```

Obtain similar plots for each of the variables `taill`, `footlgth` and `earconch` in the possum data. Repeat the exercise for males and females separately

Exercise 10
Use normal probability plots to assess whether the following sets of values, all from data sets in the `DAAG` package, have distributions that seem consistent with the assumption that they have been sampled from a normal distribution?

(a) the difference `heated - ambient`, in the data frame `pair65` (*DAAG*)?

(b) the values of `earconch`, in the `possum` data frame (*DAAG*)?

(c) the values of `body`, in the data frame `Animals` (*MASS*)?

(d) the values of `log(body)`, in the data frame `Animals` (*MASS*)?

4 Boxplots – Simple Summary Information on a Distribution

In the data frame `cfseal` (*DAAG*), several of the columns have a number of missing values. A relevant question is: "Do missing and non-missing rows have similar values, for columns that are complete?"

Exercise 11
Use the following to find, for each column of the data frame `cfseal`, the number of missing values:

```
sapply(cfseal, function(x)sum(is.na(x)))
```

Observe that for `lung`, `leftkid`, `rightkid`, and `intestines` values are missing in the same six rows. For each of the remaining columns compare, do boxplots that compare the distribution of values for the 24 rows that had no missing values with the distribution of values for the 6 rows that had missing values.

Here is code that can be used to get started:

```
present <- complete.cases(cfseal)
boxplot(age ~ present, data=cfseal)
```

Or you might use the lattice function and do the following:

```
present <- complete.cases(cfseal)
library(lattice)
present <- complete.cases(cfseal)
bwplot(present ~ age, data=cfseal)
```

Exercise 12
Tabulate, for the same set of columns for which boxplots were obtained in Exercise 2, differences in medians, starting with:

```
median(age[present]) - median(age[!present]))
```

Calculate also the ratios of the two interquartile ranges, i.e.

```
IQR(age[present]) - IQR(age[!present]))
```

Part VII
Informal Uses of Resampling Methods

1 Bootstrap Assessments of Sampling Variability

Exercise 1
The following takes a with replacement sample of the rows of `Pima.tr2`.

```
> rows <- sample(1:dim(Pima.tr2)[1], replace=TRUE)
> densityplot(~ bmi, groups=type, data=Pima.tr2[rows, ],
+           scales=list(relation="free"), xlab="Measure")
```

Repeat, but using `anymiss` as the grouping factor, and with different panels for the two levels of `type`. Repeat for several different bootstrap samples. Are there differences between levels of `anymiss` that seem consistent over repeated bootstrap samples?

Exercise 2
The following compares density plots, for several of the variables in the data frame `Pima.tr2`, between rows that had one or more missing values and those that had no missing values.

```
> missIND <- complete.cases(Pima.tr2)
> Pima.tr2$anymiss <- c("miss","nomiss")[missIND+1]
> library(lattice)
> stripplot(anymiss ~ npreg + glu | type, data=Pima.tr2, outer=TRUE,
+           scales=list(relation="free"), xlab="Measure")
```

The distribution for `bmi` gives the impression that it has a different shape, between rows where one or more values was missing and rows where no values were missing, at least for `type=="Yes"`. The bootstrap methodology can be used to give a rough check of the consistency of apparent differences under sampling variation. The idea is to treat the sample as representative of the population, and takes repeated with replacement ("bootstrap") samples from it. The following compares the qq-plots between rows that had missing data (`anymiss=="miss"`) and rows that were complete (`anymiss=="nomiss"`), for a single bootstrap sample, separately for the non-diabetics (`type=="No"`) and the diabetics (`type=="Yes"`).

```
> rownum <- 1:dim(Pima.tr2)[1]  # generate row numbers
> chooserows <- sample(rownum, replace=TRUE)
> qq(anymiss ~ bmi | type, data=Pima.tr2[chooserows, ],
+    auto.key=list(columns=2))
```

Wrap these lines of code in a function. Repeat the formation of the bootstrap samples and the plots several times. Does the shift in the distribution seem consistent under repeating sampling?

Judgements based on examination of graphs are inevitably subjective. They do however make it possible to compare differences in the shapes of distributions. Here, the shape difference is of more note than any difference in mean or median.

Exercise 3
In the data frame `nswdemo` (*DAAGxtras* package), compare the distribution of `re78` for those who received work training (`trt==1`) with controls (`trt==0`) who did not.

```
> library(DAAGxtras)
> densityplot(~ re78, groups=trt, data=nswdemo, from=0,
+             auto.key=list(columns=2))
```

Exercise 3, continued
The distributions are highly skew. A few very large values may unduly affect the comparison. A reasonable alternative is to compare values of `log(re78+23)`. The value 23 is chosen because it is half the minimum non-zero value of `re78`. Here is the density plot.

```
> unique(sort(nswdemo$re78))[1:3]  # Examine the 3 smallest values
> densityplot(~ log(re78+23), groups=trt, data=nswdemo,
+             auto.key=list(columns=2))
```

Do the distribution for control and treated have similar shapes?

Exercise 4
Now examine the displacement, under repeated bootstrap sampling, of one mean relative to the other. Here is code for the calculation:

```
> twoBoot <- function(n=999, df=nswdemo, ynam="re78", gp="trt"){
+   fac <- df[, gp]; if(!is.factor(fac))fac <- factor(fac)
+   if(length(levels(fac)) != 2) stop(paste(gp, "must have 2 levels"))
+   y <- df[, ynam]
+   d2 <- c(diff(tapply(y, fac, mean)), rep(0, n))
+   for(i in 1:n){
+     chooserows <- sample(1:length(y), replace=TRUE)
+     faci <- fac[chooserows]; yi <- y[chooserows]
+     d2[i+1] <- diff(tapply(yi, faci, mean))
+ }
+   d2
+ }
> ##
> d2 <- twoBoot()
> quantile(d2, c(.025,.975))   # 95% confidence interval
```

Note that a confidence interval should not be interpreted as a probability statement. It takes no account of prior probability. Rather, 95% of intervals that are calculated in this way can be expected to contain the true probability.

2 Use of the Permutation Distribution as a Standard

Exercise 5
If the difference is entirely due to sampling variation, then permuting the treatment labels will give another sample from the same null distribution. The permutation distribution is the distribution of differences of means from repeated samples, obtained by permuting the labels.
This offers a standard against which to compare the difference between treated and controls. Does the observed difference between treated and controls seem "extreme", relative to this permutation distribution? Note that the difference between `treat==1` and `treat==1` might go in either direction. Hence the multiplication of the tail probability by 2. Here is code:

```
> dnsw <- numeric(1000);
> y <- nswdemo$re78; treat <- nswdemo$trt
> dnsw[1] <- mean(y[treat==1]) - mean(y[treat==0])
> for(i in 2:1000){
+   trti <- sample(treat)
+   dnsw[i] <- mean(y[trti==1]) - mean(y[trti==0])
+ }
> 2*min(sum(d2<0)/length(d2), sum(d2>0)/length(d2))  # 2-sided comparison
```

Replace `re78` with `log(re78+23)` and repeat the calculations.

Part VIII
Sampling Distributions, & the Central Limit Theorem

Package: `DAAGxtras`

1 Sampling Distributions

The exercises that follow demonstrate the sampling distribution of the mean, for various different population distributions. More generally, sampling distributions of other statistics may be important.

Inference with respect to means is commonly based on the sampling distribution of the mean, or of a difference of means, perhaps scaled suitably. The ideas extend to the statistics (coefficients, etc) that arise in regression or discriminant or other such calculations. These ideas are important in themselves, and will be useful background for later laboratories and lectures.

Here, it will be assumed that sample values are independent. There are several ways to proceed.

- The distribution from which the sample is taken, although not normal, is assumed to follow a common standard form. For example, in the life testing of industrial components, an exponential or Weibull distribution might be assumed. The relevant sampling distribution can be estimated by taking repeated random samples from this distribution, and calculating the statistic for each such sample.
- If the distribution is normal, then the sample distribution of the mean will also be normal. Thus, taking repeated random samples is unnecessary; theory tells us the shape of the distribution.
- Even if the distribution is not normal, the Central Limit Theorem states that, by taking a large enough sample, the sampling distribution can be made arbitrarily close to normal. Often, given a population distribution that is symmetric, a sample of 4 or 5 is enough, to give a sampling distribution that is for all practical purposes normal.
- The final method [the "bootstrap"] that will be described is empirical. The distribution of sample values is treated as if it were the population distribution. The form of the sampling distribution is then determined by taking repeated random with replacement samples (bootstrap samples), of the same size as the one available sample, from that sample. The value of the statistic is calculated for each such bootstrap sample. The repeated bootstrap values of the statistic are used to build a picture of the sampling distribution.

  With replacement samples are taken because this is equivalent to sampling from a population in which each of the avaialble sample values is repeated an infinite number of times.

The bootstrap method obviously works best if the one available sample is large, thus providing an accurate estimate of the population distribution. Likewise, the assumption that the sampling distribution is normal is in general most reasonable if the one available sample is of modest size, or large. Inference is inevitably hazardous for small samples, unless there is prior information on the likely form of the distribution. As a rough summary:

- Simulation (repeated resampling from a theoretical distribution or distributions) is useful
  - as a check on theory (the theory may be approximate, or of doubtful relevance)
  - where there is no adequate theory
  - to provide insight, especially in a learning context.
- The bootstrap (repeated resampling from an empirical distribution or distributions) can be useful

- when the sample size is modest and uncertainty about the distributional form may materially affect the assessment of the shape of the sampling distribution;
- when standard theoretical models for the population distribution seem unsatisfactory.

The idea of a sampling distribution is wider than that of a sampling distribution of a statistic. It can be useful to examine the sampling distribution of a graph, i.e., to examine how the shape of a graph changes under repeated bootstrap sampling.

Exercise 1
First, take a random sample from the normal distribution, and plot the estimated density function:

```
> y <- rnorm(100)
> plot(density(y), type = "l")
```

Now take repeated samples of size 4, calculate the mean for each such sample, and plot the density function for the distribution of means:

```
> av <- numeric(100)
> for (i in 1:100) {
+     av[i] <- mean(rnorm(4))
+ }
> lines(density(av), col = "red")
```

Repeat the above: taking samples of size 9, and of size 25.

Exercise 2
It is also possible to take random samples, usually with replacement, from a vector of values, i.e., from an empirical distribution. This is the bootstrap idea. Again, it may of interest to study the sampling distributions of means of different sizes. Consider the distribution of heights of female Adelaide University students, in the data frame `survey` (*MASS* package). The following takes 100 bootstrap samples of size 4, calculating the mean for each such sample:

```
> library(MASS)
> y <- na.omit(survey[survey$Sex == "Female", "Height"])
> av <- numeric(100)
> for (i in 1:100) av[i] <- mean(sample(y, 4, replace = TRUE))
```

Repeat, taking samples of sizes 9 and 16. In each case, use a density plot to display the (empirical) sampling distribution.

Exercise 3
Repeat exercise 1 above: (a) taking values from a uniform distribution (replace `rnorm(4)` by `runif(4)`); (b) from an exponential distribution with rate 1 (replace `rnorm(4)` by `rexp(4, rate=1)`).
[As noted above, density plots are not a good tool for assessing distributional form. They are however quite effective, as here, for showing the reduction in the standard deviation of the sampling distribution of the mean as the sample size increases. The next exercise but one will repeat the comparisons, using normal probability plots in place of density curves.]

Exercise 4
Laboratory 3 examined the distribution of `bmi` in the data frame `Pima2` (*MASS* package). The distribution looked as though it might have shifted, for data where one or more rows was missing, relative to other rows. We can check whether this apparent shift is consistent under repeated sampling. Here again is code for the graph for `bmi`

```
> library(MASS)
> library(lattice)
> complete <- complete.cases(Pima.tr2)
> completeF <- factor(c("oneORmore", "none")[as.numeric(complete) +
+     1])
> Pima.tr2$completeF <- completeF
> densityplot(~bmi, groups = completeF, data = Pima.tr2, auto.key = list(columns = 2))
```

Now take one bootstrap sample from each of the two categories of row, then repeating the density plot.

```
> rownum <- seq(along = complete)
> allpresSample <- sample(rownum[complete], replace = TRUE)
> NApresSample <- sample(rownum[!complete], replace = TRUE)
> densityplot(~bmi, groups = completeF, data = Pima.tr2, auto.key = list(columns = 2),
+     subset = c(allpresSample, NApresSample))
```

Wrap these lines of code in a function. Repeat the formation of the bootstrap samples and the plots several times. Does the shift in the distribution seem consistent under repeating sampling?

Exercise 5
More commonly, one compares examines the displacement, under repeated sampling, of one mean relative to the other. Here is code for the calculation:

```
> twot <- function(n = 99) {
+     complete <- complete.cases(Pima.tr2)
+     rownum <- seq(along = complete)
+     d2 <- numeric(n + 1)
+     d2[1] <- with(Pima.tr2, mean(bmi[complete], na.rm = TRUE) -
+         mean(bmi[!complete], na.rm = TRUE))
+     for (i in 1:n) {
+         allpresSample <- sample(rownum[complete], replace = TRUE)
+         NApresSample <- sample(rownum[!complete], replace = TRUE)
+         d2[i + 1] <- with(Pima.tr2, mean(bmi[allpresSample],
+             na.rm = TRUE) - mean(bmi[NApresSample], na.rm = TRUE))
+     }
+     d2
+ }
> d2 <- twot(n = 999)
> dens <- density(d2)
> plot(dens)
> sum(d2 < 0)/length(d2)

[1] 0.185
```

Those who are familiar with t-tests may recognize the final calculation as a bootstrap equivalent of the t-test.

Exercise 6
The range that contains the central 95% of values of `d2` gives a 95% confidence (or coverage) interval for the mean difference. Given that there are 1000 values in total, the interval is the range from the 26th to the 975th value, when values are sorted in order of magnitude, thus:

```
> round(sort(d2)[c(26, 975)], 2)

[1] -1.06  2.43
```

Repeat the calculation of `d2` and the calculation of the resulting 95% confidence interval, several times.

2 The Central Limit Theorem

Theoretically based t-statistic and related calculations rely on the assumption that the sampling distribution of the mean is normal. The Central Limit Theorem assures that the distribution will for a large enough sample be arbitrarily close to normal, providing only that the population distribution has a finite variance. Simulation of the sampling distribution is especially useful if the population distribution is not normal, providing an indication of the size of sample needed for the sampling distribution to be acceptably close to normal.

Exercise 7
The function `simulateSampDist()` (*DAAGxtras*) allows investigation of the sampling distribution of the mean or other stastistic, for an arbitrary population distribution and sample size. Figure 1 shows sampling distributions for samples of sizes 4 and 9, from a normal population. The function call is

```
> library(DAAGxtras)
> sampvalues <- simulateSampDist(numINsamp = c(4, 9))
> plotSampDist(sampvalues = sampvalues, graph = "density", titletext = NULL)
```

Experiment with sampling from normal, uniform, exponential and t_2-distributions. What is the effect of varying the value of `numsamp`?
[To vary the kernel and/or the bandwidth used by `density()`, just add the relevant arguments in the call to to `simulateSampDist()`, e.g. `sampdist(numINsamp=4, bw=0.5)`. Any such additional arguments (here, `bw`) are passed via the `...` part of the parameter list.]

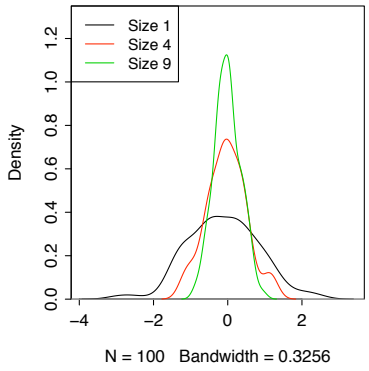

Figure 1: Empirical density curve, for a normal population and for the sampling distributions of means of samples of sizes 4 and 9 from that population.

Exercise 8
The function `simulateSampDist()` has an option (`graph="qq"`) that allows the plotting of a normal probability plot. Alternatively, by using the argument `graph=c("density","qq")`, the two types of plot appear side by side, as in Figure 2. Figure 2 is an example of its use.

```
> sampvalues <- simulateSampDist()
> plotSampDist(sampvalues = sampvalues, graph = c("density", "qq"))
```

In the right panel, the slope is proportional to the standard deviation of the distribution. For means of a sample size equal to 4, the slope is reduced by a factor of 2, while for a sample size equal to 9, the slope is reduced by a factor of 3.
Comment in each case on how the spread of the density curve changes with increasing sample size. How does the qq-plot change with increasing sample size? Comment both on the slope of a line that might be passed through the points, and on variability about that line.

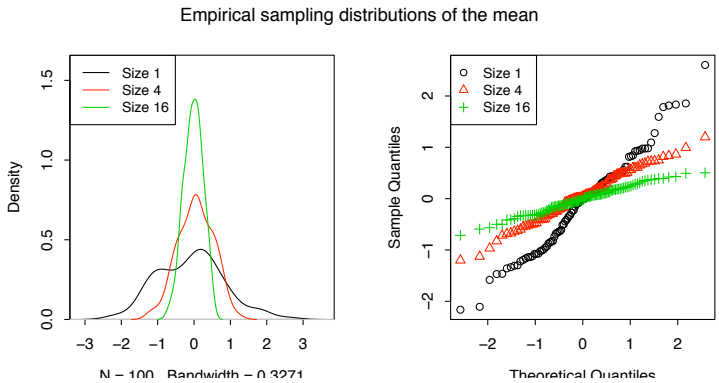

Figure 2: Empirical density curves, for a normal population and for the sampling distributions of means of samples of sizes 4 and 9, are in the left panel. The corresponding normal probability plots are shown in the right panel.

Exercise 9
How large is "large enough", so that the sampling distribution of the mean is close to normal? This will depend on the population distribution. Obtain the equivalent for Figure 2, for the following populations:

(a) A t-distribution with 2 degrees of freedom
[`rpop = function(n)rt(n, df=2)`]

(b) A log-normal distribution, i.e., the logarithms of values have a normal distribution
[`rpop = function(n, c=4)exp(rnorm(n)+c)`]

(c) The empirical distribution of heights of female Adelaide University students, in the data frame `survey` (*MASS* package). In the call to `simulateSampDist()`, the parameter `rpop` can specify a vector of numeric values. Samples are then obtained by sampling with replacement from these numbers. For example:

```
> library(MASS)
> y <- na.omit(survey[survey$Sex == "Female", "Height"])
> sampvalues <- simulateSampDist(y)
> plotSampDist(sampvalues = sampvalues)
```

How large a sample seems needed, in each instance, so that the sampling distribution is approximately normal – around 4, around 9, or greater than 9?

Part IX
Simple Linear Regression Models

The primary function for fitting linear models is `lm()`, where the `lm` stands for linear model.[3]

R's implementation of linear models uses a symbolic notation[4] that gives a straightforward powerful means for describing models, including quite complex models. Models are encapsulated in a *model formula*. Model formulae that extend and/or adapt this notation are used in R's modeling functions more generally.

1 Fitting Straight Lines to Data

Exercise 1
In each of the data frames `elastic1` and `elastic2`, fit straight lines that show the dependence of `distance` on `stretch`. Plot the two sets of data, using different colours, on the same graph. Add the two separate fitted lines. Also, fit one line for all the data, and add this to the graph.

Exercise 2
In the data set pressure (*datasets*), the relevant theory is that associated with the Claudius-Clapeyron equation, by which the logarithm of the vapor pressure is approximately inversely proportional to the absolute temperature. Transform the data in the manner suggested by this theoretical relationship, plot the data, fit a regression line, and add the line to the graph. Does the fit seem adequate?
[For further details of the Claudius-Clapeyron equation, search on the internet, or look in a suitable reference text.]

Exercise 3
Run the function `plotIntersalt()`, which plots data from the data frame `intersalt` (*DAAGxtras* package). Data are population average values of blood pressure and of salt in the body as measured from urine samples, from 52 different studies. Is the fitted line reasonable? Or is it a misinterpretation of the data? Suggest alternatives to regression analysis, for getting a sense of how these results should be interpreted? What are the populations where levels are very low? What is special about these countries?
[The function `plotIntersalt()` is available from
`http://www.maths.anu.edu.au/~johnm/r/functions/`]
Enter

```
> webfile <- "http://www.maths.anu.edu.au/~johnm/r/functions/plotIntersalt.RData"
> load(con <- url(webfile))
> close(con)
```

[3]The methodology that `lm()` implements takes a very expansive view of linear models. While models must be linear in the parameters, responses can be highly non-linear in the explanatory variables. For the present attention will be limited to examples where the explanatory variables ("covariates") enter linearly.

[4]The notation is a version of that described in "Wilkinson G.N. and Rogers, C. E. (1973) Symbolic description of factorial models for analysis of variance. Appl. Statist., 22, 392-9."

Exercise 4
A plot of heart weight (`heart`) versus body weight (`weight`), for Cape Fur Seal data in the data set `cfseal` (*DAAG*) shows a relationship that is approximately linear. Check this. However variability about the line increases with increasaing weight. It is better to work with `log(heart)` and `log(weight)`, where the relationship is again close to linear, but variability about the line is more homogeneous. Such a linear relationship is consistent with biological allometry, here across different individuals. Allometric relationships are pairwise linear on a logarithmic scale.
Plot `log(heart)` against `log(weight)`, and fit the least squares regression line for `log(heart)` on `log(weight)`.

```
> library(DAAG)
> cflog <- log(cfseal[, c("heart", "weight")])
> names(cflog) <- c("logheart", "logweight")
> plot(logheart ~ logweight, data=cflog)
> cfseal.lm <- lm(logheart ~ logweight, data=cflog)
> abline(cfseal.lm)
```

Use `model.matrix(cfseal.lm)` to examine the model matrix, and explain the role of its columns in the regression calculations.

2 Multiple Explanatory Variables

Exercise 5
For the data frame `oddbooks` (*DAAG*),

(a) Add a further column that gives the density.

(b) Use the function `pairs()`, or the *lattice* function `splom()`, to display the scatterplot matrix. Which pairs of variables show evidence of a strong relationship?

(c) In each panel of the scatterplot matrix, record the correlation for that panel. (Use `cor()`) to calculate correlations).

(d) Fit the following regression relationships:

  (i) `log(weight)` on `log(thick)`, `log(height)` and `log(breadth)`.

  (ii) `log(weight)` on `log(thick)` and `0.5*(log(height) + log(breadth))`. What feature of the scatterplot matrix suggests that this might make sense to use this form of equation?

(e) Take whichever of the two forms of equation seems preferable and rewrite it in a form that as far as possible separates effects that arise from changes in the linear dimensions from effects that arise from changes in page density.

[NB: To regress `log(weight)` on `log(thick)` and `0.5*(log(height)+log(breadth))`, the model formula needed is `log(weight) ~ log(thick) + I(0.5*(log(height)+log(breadth)))`
The reason for the use of the wrapper function `I()` is to prevent the parser from giving `*` the special meaning that it would otherwise have in a model formula.]

Part X
Extending the Linear Model

Package: `DAAG`,

Ideas that will be important in the expansive view of linear models that will now be illustrated include: *basis function*, *factor*, and *interaction*. The reach of R's *model formulae* is wide.

1 A One-way Classification – Eggs in the Cuckoo's Nest

This demonstrates the use of linear models to fit qualitative effects.

Like many of nature's creatures, cuckoos have a nasty habit. They lay their eggs in the nests of other birds. First, let's see how egg length changes with the host species. This will use the graphics function `stripplot()` from the *lattice* package. The data frame `cuckoos` is in the *DAAG* package.

```
> par(mfrow=c(1,2))
> library(DAAG)
> library(lattice)
> names(cuckoos)[1] <- "length"
> table(cuckoos$species)
```

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

```
> stripplot(species ~ length, data=cuckoos)
> ## Look also at the relationship between length and breadth;
> ## is it the same for all species?
> cuckoo.strip <- stripplot(breadth ~ length, groups=species,
+                           data=cuckoos, auto.key=list(columns=3))
> print(cuckoo.strip)
> par(mfrow=c(1,1))
```

Exercise 1
Now estimate the means and standard deviations for each of the groups, using direct calculation:

```
> with(cuckoos, sapply(split(length, species), mean))
```

```
hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
        23.11         22.29         22.89         22.56         23.08
         wren
        21.12
```

```
> with(cuckoos, sapply(split(length, species), sd))
```

```
hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
       1.0494        0.9196        1.0723        0.6821        0.8801
         wren
       0.7542
```

[The function `split()` splits the lengths into six sublists, one for each species. The function `sapply()` applies the function calculation to the vectors of lengths in each separate sublist.]
Check that the SD seems smallest for those species where the SD seems, visually, to be smallest.

Exercise 2
Obtain, for each species, the standard error of the mean. This is obtained by dividing the standard deviation by the square root of the number of values:

```
> sdev <- with(cuckoos, sapply(split(length, species), sd))
> n <- with(cuckoos, sapply(split(length, species), length))
> sdev/sqrt(n)

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
       0.2805        0.1371        0.2769        0.1705        0.2272
         wren
       0.1947
```

Exercise 3
Now estimate obtain the means for each of the groups froam a model that fits a separate constant term for each species. The model can be specified using several different, but equivalent, formulations, or parameterizations.

- The following the species means directly, as parameters for the model. It forces all parameters to be species means

  ```
  > lm(length ~ -1 + species, data=cuckoos)
  ```

- In the following alternative parameterization, the first parameter estimate is the mean for the first species, while later estimates are differences from the first species. In other words, the first species becomes the baseline.

  ```
  > lm(length ~ species, data=cuckoos)
  ```

Now answer the following

(a) Use the function `fitted()` to calculate the fitted values in each case, and check that they are the same. Reconcile the two sets of results.

(b) Examine and interpret the model matrices for the two different parameterizations.

(c) Use the `termplot()` function to show the effects of the different factor levels. Be sure to call the function with `partial.resid=TRUE` and with `se=TRUE`. Does the variability seem similar for all host species?

Exercise 4
The following creates (in the first line) and uses (second line) a function that calculates the standard error of the mean.

```
> se <- function(x)sd(x)/sqrt(length(x))
> with(cuckoos, sapply(split(length, species), se))

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
       0.2805        0.1371        0.2769        0.1705        0.2272
         wren
       0.1947
```

Exercise 4, continued
The standard error calculation can be done in a single line of code, without formally creating the function `se()`. Instead, the function definition can be inserted as an anonymous function, so that it does not need a name. The function definition is inserted where the function name would otherwise appear:

```
> with(cuckoos, sapply(split(length, species),
+                      function(x)sd(x)/sqrt(length(x))))

hedge.sparrow  meadow.pipit  pied.wagtail         robin    tree.pipit
       0.2805        0.1371        0.2769        0.1705        0.2272
         wren
       0.1947
```

2 Regression Splines – one explanatory variable

This pursues the use of smoothing methods, here formally within a regression context.

Exercise 5
The following is based on the `fruitohms` data frame (in *DAAG*)

(a) Plot `ohms` against `juice`.

(b) Try the following:

```
> plot(ohms ~ juice, data=fruitohms)
> with(fruitohms, lines(lowess(juice, ohms)))
> with(fruitohms, lines(lowess(juice, ohms, f=0.2), col="red"))
```

Which of the two fitted curves best captures the pattern of change?

(c) Now fit a natural regression spline. Try for example:

```
> library(splines)
> plot(ohms ~ juice, data=fruitohms)
> hat <- with(fruitohms, fitted(lm(ohms ~ ns(juice, 4))))
> with(fruitohms, lines(juice,hat, col=3))
> ## Check the locations of the internaal knots.
> attributes(ns(fruitohms$juice, 4))$knots

  25%   50%   75%
18.62 41.00 48.00
```

Experiment with different choices for the number of degrees of freedom. How many degrees of freedom seem needed to adequately capture the pattern of change? Plot the spline basis functions. Add vertical lines to the plot that show the knot locations.

Exercise 6
In the `geophones` data frame (*DAAG*), plot `thickness` against distance. Use regression splines, as in Exercise 6, to fit a suitable curve through the data. How many degrees of freedom seem needed? Add vertical lines to the plot that show the locatons of the knots.

3 Regression Splines – Two or More Explanatory Variables

We begin by using the `hills2000` data (*DAAG* version 0.84 or later) to fit a model that is linear in `log(dist)` and `log(climb)`, thus

```
> lhills2k <- log(hills2000[, 2:4])
> names(lhills2k) <- c("ldist",  "lclimb", "ltime")
> lhills2k.lm <- lm(ltime ~ ldist + lclimb, data = lhills2k)
```

Use `termplot()` to check departures from linearity in `lhills2k.lm`. Note whether there seem to be any evident outliers.

Exercise 7
Now use regression splines to take out the curvature that was evident when `ltime` was modeled as a linear function of `ldist` and `lclimb`. Use `termplot()` to guide your choice of degrees of freedom for the spline bases. For example, you might try

```
> lhills2k.ns <- lm(ltime ~ ns(ldist,2) + lclimb, data = lhills2k)
```

Again, examine the diagnostic plots? Are there any points that should perhaps be regarded as outliers?
Does a normal spline of degree 2 in `ldist` seem to give any benefit above a polynomial of degree 2.

Note: The coefficients of the spline basis terms do not give useful information on what degree of spline curve is required. See exercise 9 below. If one fits a spline of degree 3 to a relationship that is esentially linear, all three coefficients are likely to be highly significant. Rather, check how the residual sum of squares changes as the number of degrees of freedom for the spline curve increases. [F-tests are not strictly valid, as successive models are not nested (the basis functions change), but they may be a helpful guide.]

Exercise 8
The *MASS* package has the function `lqs()` that can be used for a *resistant* regression fit, i.e., the effect of outlying points is attenuated. Try the following

```
> library(MASS)
> lhills2k.lqs <- lqs(ltime ~ ns(ldist,2) + lclimb, data = lhills2k)
> plot(resid(lhills2k.lqs) ~ fitted(lhills2k.lqs))
> big4 <- order(abs(resid(lhills2k.lqs)), decreasing=TRUE)[1:4]
> text(resid(lhills2k.lqs)[big4] ~ fitted(lhills2k.lqs)[big4],
+      labels=rownames(lhills2k)[big4], pos=4)
```

Try the plot without the two largest outliers. Does this make any difference of consequence to the fitted values?

Exercise 10
Try the following:

```
x <- 11:20
y <-  5 + 1.25*x+rnorm(10)
summary(lm(y ~ ns(x,2)))$coef
summary(lm(y ~ ns(x,3)))$coef
summary(lm(y ~ ns(x,4)))$coef
```

Note that the coefficents are in all cases very mcuh larger than their standard errors. It takes both degree 2 spline basis terms, additional to the constant, to fit a line. All three degree 3 terms are required, and so on! Splines do not give a parsimonious representation, if the form of the model is known.

4 Errors in Variables

Exercise 10
Run the accompanying function `errorsINx()` several times. Comment on the results. The underlying relationship between y and x is the same in all cases. The error in x is varied, from values of x that are exact to values of x that have random errors added with a variance that is twice that of the variance of x.

4.1 Function used

This is available from `http://www.maths.anu.edu.au/~johnm/r/functions/`

```
"errorsINx" <-
  function(mu = 8.25, n = 100, a = 5, b = 2.5, SDx=1, sigma = 2,
           timesSDx=(1:5)/2.5){
    mat <- matrix(0, nrow=n, ncol=length(timesSDx)+2)

    x0 <- mu*exp(rnorm(n,0,SDx/mu))/exp(0)

    y <- a + b*x0+rnorm(n,0,sigma)
    mat[, length(timesSDx)+2] <- y
    mat[,2] <- x0
    mat[,1] <- y
    sx <- sd(x0)
    k <- 2
    for(i in timesSDx){
      k <- k+1
      xWITHerror <- x0+rnorm(n, 0, sx*i)
      mat[, k] <- xWITHerror
    }
    df <- as.data.frame(mat)
    names(df) <- c("y", "x", paste("x",timesSDx,sep=""))
    df
  }

## Now use function to simulate y vs x relationships, with several
## different values of timesSDx, which specifies the ratio of the
## errors in x variance to SD[x]
    oldpar <- par(mar=c(3.6,3.1,1.6,0.6), mgp=c(2.5,0.75,0),
                  oma=c(1,1,0.6,1),
                  mfrow=c(2,3), pty="s")
    mu <- 20; n <- 100; a <- 15; b <- 2.5; sigma <- 12.5; timesSigma<-(1:5)/2.5
    mat <- errorsINx(mu = 20, n = 100, a = 15, b = 2.5, sigma = 5,
                     timesSDx=(1:5)/2.5)
    beta <- numeric(dim(mat)[2]-1)
    sx <- sd(mat[,2])
    y <- mat[, 1]
    for(j in 1:length(beta)){
      xj <- mat[,j+1]
      plot(y ~ xj, xlab="", ylab="", col="gray30")
      if(j==1)
        mtext(side=3, line=0.5, "No error in x") else{
          xm <- timesSigma[j-1]
          mtext(side=3, line=0.5, substitute(tau == xm*s[z], list(xm=xm)))
        }
```

```
      if(j>=4)mtext(side=1, line=2, "x")
      if(j%%3 == 1)mtext(side=2, line=2, "y")
      errors.lm <- lm(y ~ xj)
      abline(errors.lm)
      beta[j] <- coef(errors.lm)[2]
      bigsigma <- summary(errors.lm)$sigma
      print(bigsigma/sigma)
      abline(a, b, lty=2)
    }
    print(round(beta, 3))

plotIntersalt <-
function (dset = intersalt1, figno = 2)
{
    oldpar <- par(oma = c(6.5, 0, 0, 0), mar = par()$mar - c(0,
        0, 3.5, 0))
    on.exit(par(oldpar))
    lowna <- c(4, 5, 24, 28)
    plot(dset$na, dset$bp, pch = 15, ylim = c(50, 85),
         xlab = "Median sodium excretion (mmol/24hr)",
         ylab = "Median diastolic BP (mm Hg)", type = "n")
    points(dset$na[-lowna], dset$bp[-lowna], pch = 16)
    points(dset$na[lowna], dset$bp[lowna], pch = 1, lwd = 3)
    u <- lm(bp ~ na, data = dset)
    abline(u$coef[1], u$coef[2])

    figtxt <-
       paste("Fig. ", figno, ": Plot of median blood pressure versus salt",
             "\\n(measured by sodium excretion) for 52 human",
             "\\npopulations. Four results (open circles) are for",
             "\\nnon-industrialised societies with very low salt intake,",
             "\\nwhile other results are for industrialised societies.",
       sep = "")
    mtext(side = 1, line = 6, figtxt, cex = 1.1, adj = 0, at = -20)
}
```

Part XI
Multi-level Models

1 Description and Display of the Data

1.1 Description

This laboratory will work with data on corn yields from the Caribbean islands of Antigua and St Vincent. Data are yields from packages on eight sites on the Caribbean island of Antigua. The data frames `ant111b` and `vince111b` hold yields for the standard treatment, here identified as `111`, for sites on Antigua and St Vincent respectively. Additionally, there will be some use of the more extensive data in the data frame `antigua`. All three data frames are in recent versions (≥ 0.84) of the *DAAG* package. See `help(ant11b)` for details of the source of these data.

The data frame `ant111b` has data for n=4 packages of land at each of eight sites, while `vince111b` data for four packages at each of nine sites. As will be described below, two possible predictions are:

(a) Predictions for new packages of land in one of the existing sites.

(b) Predictions for new packages in a new site.

The accuracies for the second type of prediction may be much less accurate than for the first type. A major purpose of this laboratory is to show how such differences in accuracy can be modeled.

1.2 Display

We begin by examining plots, for the treatment `111`, from the combined data for the two islands. This information for the separate islands is summarized in the datasets `ant111b` and `vince111b` in the *DAAG* package.

A first step is to combine common columns of `ant111b` and `vince111b` into the single data frame `corn111b`.

```
> library(lattice)
> library(DAAG)
> corn111b <- rbind(ant111b[, -8], vince111b)
> corn111b$island <- c("Antigua", "StVincent")[corn111b$island]
```

- The following plot uses different panels for the two islands:

```
> corn.strip1 <- stripplot(site ~ harvwt | island, data = corn111b,
+     xlab = "Harvest weight")
```

- The following plot uses different panels for the two islands, but allows separate ("free" = no relation) vertical scales for the two plots.

```
> corn.strip2 <- stripplot(site ~ harvwt | island, data = corn111b,
+     xlab = "Harvest weight", scale = list(y = list(relation = "free")))
```

- The following uses a single panel, but uses different colours (or, on a black and white device, different symbols) to distinguish the two islands. Notice the use of `auto.key` to generate an automatic key:

```
> corn.strip3 <- stripplot(site ~ harvwt, data = corn111b, groups = island,
+     xlab = "Harvest weight", auto.key = list(columns = 2))
```

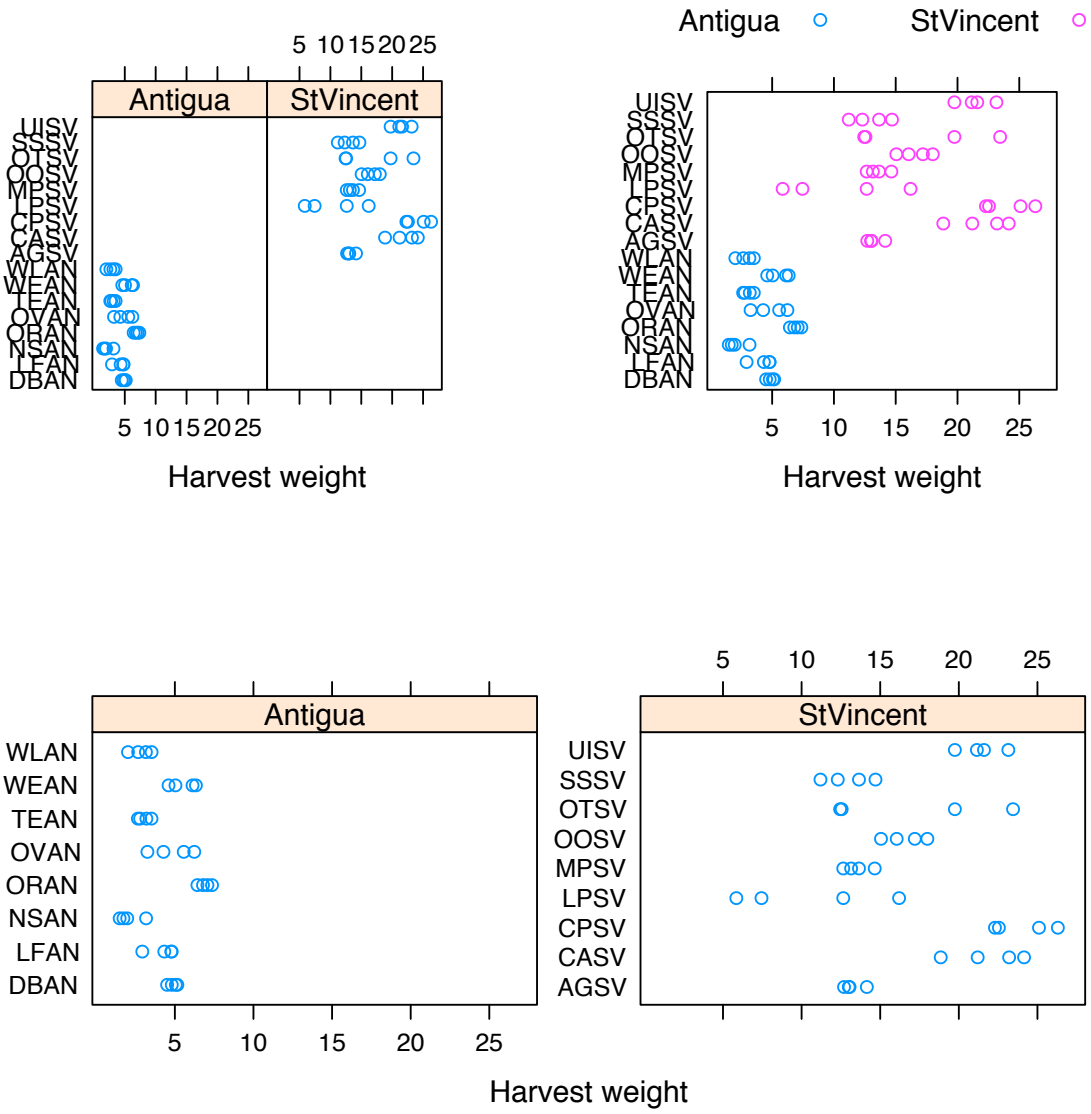

Figure 3: Yields for the four packages of corn on sites on the islands of Antigua and St Vincent.

Next, we will obtain package means for the Antiguan data, for all treatments.

```
> with(antigua, antp <<- aggregate(harvwt, by = list(site = site,
+     package = block, trt = trt), FUN = mean))
> names(antp)[4] <- "harvwt"
```

Notice the use of the version `<<-` of the assignment symbol to ensure that assignment takes place in the workspace.

Now plot mean harvest weights for each treatment, by site:

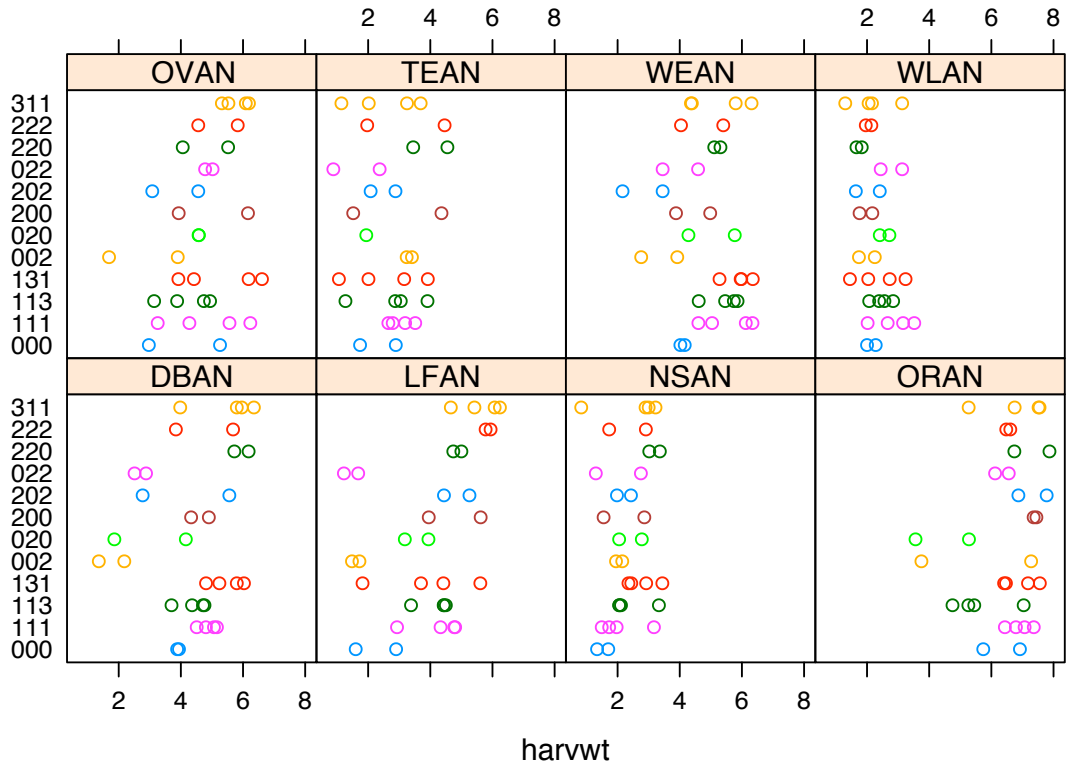

Figure 4: Yields for the four packages of corn on each of eight sites on the island of Antigua.

Questions and Exercises

(a) Which set of sites (Antigua or St Vincent) shows the largest yields?

(b) Create a plot that compares the logarithms of the yields, within and between sites on the two islands. From this plot, what, if anything, can you say about the differet variabilities in yield, within and between sites on each island?

2 Multi-level Modeling

***Analysis using *lme*:** The modeling command takes the form:

```
> library(nlme)
> ant111b.lme <- lme(fixed = harvwt ~ 1, random = ~1 | site, data = ant111b)
```

The only fixed effect is the overall mean. The argument `random = ~1|site` fits random variation between sites. Variation between the individual units that are nested within sites, i.e., between packages, are by default treated as random. Here is the default output:

```
> options(digits = 4)
> ant111b.lme

Linear mixed-effects model fit by REML
  Data: ant111b
  Log-restricted-likelihood: -47.21
```

```
  Fixed: harvwt ~ 1
(Intercept)
      4.292

Random effects:
 Formula: ~1 | site
        (Intercept) Residual
StdDev:       1.539     0.76

Number of Observations: 32
Number of Groups: 8
```

Notice that *lme* gives, not the components of variance, but the standard deviations (`StdDev`) which are their square roots. Observe that, according to *lme*, $\widehat{\sigma_B^2} = 0.76^2 = 0.578$, and $\widehat{\sigma_L^2} = 1.539^2 = 2.369$. The variance for an individual package is $\widehat{\sigma_B^2} + \widehat{\sigma_L^2}$.
Those who are familiar with an analysis of variance table for such data should note that *lme* does not give the mean square at any level higher than level 0, not even in this balanced case.

Note that the yields are not independent between different packages on the same site, in the population that has packages from multiple sites. (Conditional on coming from one particular site, package yields are however independent.)

The take-home message from this analysis is:

- For prediction for a new package at one of the existing sites, the standard error is 0.76
- For prediction for a new package at a new site, the standard error is $\sqrt{1.539^2 + .76^2} = 1.72$
- For prediction of the mean of n packages at a new site, the standard error is $\sqrt{1.539^2 + 0.76^2/n}$. This is NOT inversely proportional to n, as would happen if the yields were independent within sites.

Where there are multiple levels of variation, the predictive accuracy can be dramatically different, depending on what is to be predicted. Similar issues are arise in repeated measures contexts, and in time series. Repeated measures data has multiple profiles, i.e., many small time series.

2.1 Simulation

The following function simulates results from a multilevel model for the case where there are `npackages` packages at each of `nplots` plots.

```
> "simMlevel" <- function(nsites = 8, npackages = 4, mu = 4, sigmaL = 1.54,
+     sigmaB = 0.76) {
+     facSites <- factor(1:nsites)
+     facPackages <- factor(1:npackages)
+     dframe <- expand.grid(facPackages = facPackages, facSites = facSites)
+     nall <- nsites * npackages
+     siteEffects <- rnorm(nsites, 0, sigmaL)
+     err <- rnorm(nall, 0, sigmaB) + siteEffects[unclass(dframe$facSites)]
+     dframe$yield <- mu + err
+     dframe
+ }
```

The default arguments are `sigmaB` = 0.76 and `sigma` = 1.54, as for the Antiguan data.

2.2 Questions and Exercises

(a) Repeat the analysis

  (a) for the Antiguan data, now using a logarithmic scale.

  (b) for the St Vincent data, using a logarithmic scale.

(b) Overlay plots, for each of the two islands, that show how the variance of the mean can be expected to change with the number of packages n.

(c) Are there evident differences between islands in the contributions of the two components of variance? What are the practical implications that flow from such differences as you may observe?

(d) Use the function `simMlevel()` to simulate a new set of data, using the default arguments. Analyse the simulated data. Repeat this exercise 25 or more times. How closely do you reproduce the values of `sigmaL`=1.54 and `sigmaB`=0.76 that were used for the simulation?

3 Multi-level Modeling – Attitudes to Science Data

These data are from in the *DAAG* package for R. The data are measurements of attitudes to science, from a survey where there were results from 20 classes in 12 private schools and 46 classes in 29 public (i.e. state) schools, all in and around Canberra, Australia. Results are from a total of 1385 year 7 students. The variable `like` is a summary score based on two of the questions. It is on a scale from 1 (dislike) to 12 (like). The number in each class from whom scores were available ranged from 3 to 50, with a median of 21.5.

There are three variance components:

```
Between schools 0.00105
Between classes 0.318
Between students 3.05
```

The between schools component can be neglected. The variance for a class mean is $0.318 + 3.05/n$, where n is the size of the class. The two contributions are about equal when n =10.

4 *Additional Calculations

We return again to the corn yield data.

Is variability between packages similar at all sites?:

```
> if (dev.cur() == 2) invisible(dev.set(3))
> vars <- sapply(split(ant111b$harvwt, ant111b$site), var)
> vars <- vars/mean(vars)
> qqplot(qchisq(ppoints(vars), 3), 3 * vars)
```

Does variation within sites follow a normal distribution?:

```
> qqnorm(residuals(ant111b.lme))
```

What is the pattern of variation between sites?

```
> locmean <- sapply(split(log(ant111b$harvwt), ant111b$site), mean)
> qqnorm(locmean)
```

The distribution seems remarkably close to normal.

Fitted values and residuals in *lme*: By default fitted values account for all random effects, except those at level 0. In the example under discussion `fitted(ant111b.lme)` calculates fitted values at level 1, which can be regarded as estimates of the site means. They are not however the site means, as the graph given by the following calculation demonstrates:

```
> hat.lm <- fitted(lm(harvwt ~ site, data = ant111b))
> hat.lme <- fitted(ant111b.lme)
> plot(hat.lme ~ hat.lm, xlab = "Site means", ylab = "Fitted values (BLUPS) from lme")
> abline(0, 1, col = "red")
```

The fitted values are known as BLUPs (Best Linear Unbiased Predictors). Relative to the site means, they are pulled in toward the overall mean. The most extreme site means will on average, because of random variation, be more extreme than the corresponding "true" means for those sites. There is a theoretical result that gives the factor by which they should be shrunk in towards the true mean.

Residuals are by default the residuals from the package means, i.e., they are residuals from the fitted values at the highest level available. To get fitted values and residuals at level 0, enter:

```
> hat0.lme <- fitted(ant111b.lme, level = 0)
> res0.lme <- resid(ant111b.lme, level = 0)
> plot(res0.lme, ant111b$harvwt - hat0.lme)
```

5 Notes – Other Forms of Complex Error Structure

Time series are another important special case. A first step is, often, to subtract off any trend, and base further analysis on residuals about this trend. Observations that are close together in time are typically more closely correlated than observations that are widely separated in time.

The variances of the mean of n observations with variance σ^2 will, assuming that positive correlation between neighbouring observations makes the major contribution to the correlation structure, be greater than $\frac{\sigma^2}{n}$.

Here is a simple way to generate data that are sequentially correlated. The autocorrelation plot shows how the estimated correlation changes as observations move further apart.

```
> y <- rnorm(200)
> y1 <- y[-1] + 0.5 * y[-length(y)]
> acf(y1)
```

Of course the multiplier in `y1 <- y[-1] + 0.5*y[-1000]` can be any number at all, and more complex correlation structures can be generated by incorporating further lags of `y`.