Car features *(data manipulation)*

`data(Cars93)` in the MASS package contains data on 93 makes of car sold in the USA.

1. The `Type` variable classifies the type of market the car is aimed at. Find the cheapest car in each type, and the car with the greatest fuel efficiency

2. Compute the mean horsepower for each type, and the difference between each cars horsepower and the mean for its type

3. Create two new data frames for US and non-US cars.

4. Use `write.table()` to save the US car data to a file. Read it in and check to see that all the factors are correctly set as factors.

5. Use `save()` to save the non-US car data to a file. Remove the non-US data frame and `load()` the file. Note that the factors are still set.

Foreign data formats *(importing, internet)*

Use the `download.file` command to download the files

```
http://www.biostat.washington.edu/~thomas/statadta/pc5.dta
http://www.biostat.washington.edu/~thomas/statadta/pbc.sav
```

These are in Stata and SPSS format respectively. Load the foreign pacackage and use `read.dta` and `read.spss` to read the files. Use `summary` to examine these data and compare them to `data(pbc)` in the survival package.

Now use

```
drugs<-read.table(url("http://lib.stat.cmu.edu/datasets/csb/ch21a.dat"))
names(drugs)<-c("obs","morphine","THC","rep","activity","temp","temp60")
```

and use

```
url.show("http://lib.stat.cmu.edu/datasets/csb/ch21a.txt")
```

to read the documentation.

New York air quality *(graphics)*

`data(airquality)` contains measurements of ozone concentration in New York from May to September 1983, together with other relevant variables.

1. Plot the ozone concentration over time. Look at how different plotting types (`"l"`, `"h"`) affect the appearance. The EPA standard for ozone until recently was 120ppb, 140ppb was moderate nonattainment and 160ppb serious nonattainment. Indicate these with horizontal lines in appropriate colors. Use the `text()` function to annotate the severe ozone days with their dates.

2. Ozone is produced by chemical reactions in the air that require sunlight. Draw a scatterplot of ozone and solar radiation. Is the relationship monotone?

3. Perhaps wind or temperature are responsible for the shape of the plot. Use the `coplot` function to draw scatterplots of ozone and solar radiation for different levels of wind speed, of temperature, and of both at once. Do the same thing with the trellis command `xyplot`.

Anderson's Irises *(graphics)*

`data(iris)` is a famous dataset analysed by Fisher. *Iris setosa*, *I. versicolor* and *I. virginica* are related 'flag' irises of North America. The flowers are typically different shades of blue or purple and the plants are different sizes, but as with many plants there is enough colour and size variation that these are not sufficient to classify the species. *I. setosa* is notable in that the 'standards', the vertical sepals in an iris flower, are almost non-existent, while *I. virginica* has the classic iris form.

1. Draw a histogram of sepal length. Are there visible subgroups?

2. Plot sepal length and petal length for the irises, with different point colors for each species. This helps distinguish small flowers from large flowers with small sepals

3. Do the same for petal length and width, and for sepal length and width.

4. Look at `help(matplot)` for more ways to do plots of this sort.

Meta-analysis: *(Graphics, arrays, packages, objects)*

The `rmeta` package does simple meta-analysis of randomised trials.

1. Install the `rmeta` package using the `install.packages` command or the packages menu.

2. Read the documentation for the `catheter` data set and load it into R.

3. Following the example in `help(catheter)` run a Mantel-Haenszel (fixed effects) meta-analysis. The `meta.MH` function does the computations and returns an object. This object has useful `print`, `summary` and `plot` methods.

4. The defaults for the `plot` method are good for printing, but some color might be attractive for on-screen viewing.

```
plot(a,col=meta.colors(summary="DarkGreen"))
plot(a,col=meta.colors(summary="DarkGreen",lines="LightGreen"))
plot(a,col=meta.colors(summary="DarkGreen",
      lines="LightGreen",box="ForestGreen"))
plot(a,col=meta.colors(summary="DarkGreen",
      lines="LightGreen",box="goldenrod",text="magenta"))
```

   (with freedom comes responsibility)

5. The `summary()` function gives odds ratios for each trial; we might also want other graphical summaries. One such is given by `fourfoldplot()`. Look at `examplefourfoldplot`. This function needs to be given a $2 \times 2 \times k$ array from tabulating colonisation vs treatment in each study. The `catheter` data frame has 4 vectors containing the total and total colonised, by treatment, for each study. We need to rearrange the data.

   (a) Compute the numbers not colonised

```
catheter$uncol.trt<-catheter$n.trt-catheter$col.trt
catheter$uncol.ctrl<-catheter$n.ctrl-catheter$col.ctrl
```

   (b) Create a matrix of the relevant columns. This is really a vector with the columns stacked on top of each other.

```
col.table<-as.matrix(catheter[1:13,c('uncol.ctrl',
         'uncol.trt','col.ctrl','col.trt')])
col.table
as.vector(col.table) ## look at the storage order
```

(c) Turn this into a $13 \times 2 \times 2$ array and then transpose it to $2 \times 2 \times 13$

```
col.table<-array(col.table,c(13,2,2))
col.table<-aperm(col.table,c(2,3,1))
```

(d) The default standardisation makes each plot a circle if there is no association. An alternative is to show the variation in sample size between studies.

```
fourfoldplot(col.table)
fourfoldplot(col.table,std='all')
```

6. With the data in array form we can practice using `apply` to get row or column percentages. Row percentages involve dividing by the row totals, which are sums over dimension 2 of the array (ie preserving dimensions 1 and 3).

```
row.totals<-apply(col.table,c(1,3),sum)
row.frac<-sweep(col.table,c(1,3),row.totals,'/')
round(100*row.frac,1)
```

Of course it would be easier to divide by `n.trt` or `n.ctrl` before doing any of the other computations

Anscombe's quartet *(graphics, linear regression)*
`data(anscombe)` has variables `x1,...,x4,y1,...,y4`.

1. Fit linear regression models to each pair using eg

```
model1<-lm(y1~x1,data=anscombe)
```

2. Use `summary()` to look at the model summaries. The models appear identical.

3. Draw the four x-y scatterplots, and use `abline()` to add the regression line to each (perhaps in a contrasting color)

4. Plot each linear model to see how the diagnostics show up different departures from linearity

5. Look at the example code in `help(anscombe)`. Try to work out what it is doing.

Linear Regression *(linear models, graphics, transformations)*

Load the `trees` dataset, which contains measurements of the height, diameter, and volume of black cherry trees.

1. Use `pairs()` to look at relationships between these variables. Are they reasonably linear? Try `coplot` or `xyplot` to see relationships between Height and Volume controlling for Girth.

2. Fit a linear model to predict Volume from Height and Girth. Plot this model to see how well it fits.

3. A multiplicative model makes more physical sense for these data. Try a linear model for predicting the logarithm of Volume from the logarithms of Height and Girth

4. If all trees were the same shape the volume would be proportional to Height $\times$ Girth^2, ie, the coefficients of log(Height) and log(Girth) would be 1 and 2 respectively. Add `offset(log(Height))` and `offset(2*log(Girth))` to the model and use `anova()` to test this hypothesis.

5. The constant of proportionality is $\pi/(3 \times 24^2)$ for a right circular cone and $\pi/24^2$ for a cylinder. Plot actual volume against Height $\times$ Girth^2 and draw lines corresponding to the relationship for a cone and a cylinder.

6. Look at `help(plotmath)`. Label the two lines with the appropriate mathematical formulas.

Stratified binary data *(logistic and conditional logistic regression – assumes you know these methods).*

The `infert` data set is from a stratified case–control study that preceded the general availabilty of conditional logistic regression. We can compare the original analyses with modern analyses.

The study (Trichopoulos et al. (1976) Br. J. of Obst. and Gynaec. 83, 645-650) was examining the association between spontaneous and induced terminations of pregnancy and subsequent infertility. Cases were ascertained at two Greek hospitals and two controls were matched to each case on the number of previous children (`parity`), education and age.

1. Tabulate the number of cases and controls by the number of previous terminations.

   ```
   ftable(case~spontaneous+induced,data=infert)
   ```

   The first analysis of Trichopoulos et al. was to compute odds ratios comparing each cell of this table to the cell with no previous terminations. We can do this by fitting a saturated logistic regression model, or by direct calculation.

   (a) The `interaction()` function creates a factor variable that cross-classifies all the levels of a set of given factors. The function `interaction(spontaneous,induced)` will give a factor that indicates each separate cell in the table. Fit a logistic regression model, extract the coefficients and convert them to odds ratios.

   (b) `xtabs( spontaneous + induced + case, data = infert)` gives a $3 \times 3 \times 2$ table. Dividing the 2nd 'layer' by the first gives the odds, and dividing the result by the upper left cell gives the odds ratios. You might want to `round()` the result.

   The first approach makes it easier to calculate confidence intervals. Trichopoulos et al. reported lower 95% confidence limits for these odds ratios. The `summary()` method reports standard errors; these can be extracted as the `$coef` component of the summary object. The `qnorm()` function can be used in the same way as a set of Normal tables to find the critical values.

2. Compare models with `spontaneous` and `induced` to those with `factor(spontaneous)` and `factor(induced)`, and compare models with or without interactions.

3. Next try adjusting for the matching variables one at a time. Adjusting for all of them together is likely to give biased results: logistic regression

estimates are biased away from zero when the number of variables is too large.

4. The modern analysis would be conditional logistic regression, stratifying by the 83 matched sets of case and controls. How does it compare to the logistic regression analyses?

5. What does R do if you try to adjust for age explicitly in the conditional logistic regression?

Simulation for power calculations *(loops, objects)*

An easy way to perform power calculations is simulation. Suppose you are testing a drug that reduces blood pressure. You have 50 people in each of treatment and control groups, and expect the systolic blood pressure to have a mean of 150mmHg and standard devation of 15mmHg in the control group, and to be 10mmHg lower in the treatment group.

1. Assuming the distributions to be approximately Normal, simulate one set of data and perform a t-test using the `t.test` function.

2. Using the `names` function, look at the components of the object returned by `t.test`. The p-value is `t.test(x,y)$p.value`

3. Write a loop to generate data and perform a t-test 1000 times, storing the values in a vector `a`. What is the power of the study? Compare the results with those given by `power.t.test`.

4. Suppose in the treated group the standard deviation were increased to 20mmHg. The `power.t.test` function can't handle this, so rewrite your simulation to compute the power.

Receiver Operating Characteristic curves *(graphics, indexing, efficiency).*

Given a continuous test variable T and a binary status variable D the receiver operating characteristic (ROC) curve summarises how well T predicts D. They first arose in radio engineering, but now are most used in medical diagnostics research. The ROC curve plots the true positive rate $P(T > c|D = 1)$ against the false positive rate $P(T > c|D = 0)$ for every possible threshold c. A perfect test has true positive rate 1 and false positive rate 0; a perfectly useless test has equal true and false positive rates.

1. For any given cutpoint the true and false positive rates can be computed

   ```
   ptrue<-mean(T[D==1]>c)
   pfalse<-mean(T[D==0]>c])
   ```

2. It is only necessary to compute this for observed values of c (and `-Inf`). Write a `for()` loop to do it.

3. Rewrite the `for()` loop to use `sapply()`. Is it faster? Easier to understand?

4. Write a function to draw the ROC curve from vectors D and T.

5. A way to speed up the calculation is to find a different algorithm. You can rewrite $P(T > c|D == 1)$ as $P(T > c\&D == 1)/P(D == 1)$. The denominator doesn't depend on c. The numerator can be computed by ordering the data appropriately and using the `cumsum()` command, which produces cumulative sums of a vector.

6. The area under the ROC curve is a useful summary of the discrimatory power of T. How would you compute it?

7. What if you only wanted the area under the portion of the curve with $P(D = 0|T > c)$ less than, say, 0.05, because the test would never be operated at a higher false positive rate. Update your function to compute this partial area under the curve.

8. Make your function return a ROC object that has sensible `plot` and `print` methods and a `summary` method that computes partial area under the curve.

9. Use `package.skeleton()` to start producing an R package with these functions.

Data to test your code can be found in the "survival" package, `data(pbc)`. Use bilirubin levels (`T<-pbc$bili`) as the test value, and define the status as two-year survival: `D<-pbc$status==1 & pbc$time<730`.

Clustered data regression *(model frames/formula, language)* In linear regression with clustered data the usual estimate for $\hat{\beta}$ works but the standard errors are wrong. A valid estimate of var$[\hat{\beta}]$ is

$$\left(X^T X\right)^{-1} \left(U^T U\right) \left(X^T X\right)^{-1}$$

where $U_i = \sum_t x_{it}(y_{it} - \mu_{it})$.

1. Suppose we have a function `mylm(formula,data)` The idiom for creating model matrices is

   ```
   m<-match.call()
   m[[1]]<-as.name(''model.frame'')
   m<-eval(m,parent.frame())   ## the model frame
   X<-model.,matrix(terms(formula),m)
   Y<-model.response(m)
   ```

   Write a function to compute $\hat{\beta}$ and $\left(X^T X\right)^{-1}$.

2. Now we can add a `cluster=` argument to the function. When constructing the model frame the cluster argument will automatically be added. We can extract it with

   ```
   group<-model.extract(m,''cluster'')
   ```

   and use the `rowsum()` function to compute the collapsed sums U. It is then easy to produce the correct model-robust variance matrix

3. *(tricky)* Suppose we wanted to put the cluster specification in the model formula, as, say, `y~x+id(group)`.

   It would be necessary to break this into two formulas `y~x` and `~id(group)`. Look at what `terms(y x+id(group),specials=''id'')` does. The "specials" attribute identifies which part of the "variables" attribute is `id(group)`. So we can identify the real variables and the clustering variable. One approach to constructing the formulas is seen in the code for `aov` in handling the `Error()` term: use `paste` to produce character strings and then `as.formula` to convert them back to formulas. Try doing this.