OLLSCOIL NA hEIREANN, CORCAIGH
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLAISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

ST4060 - ST6015 - ST6040
Statistical Learning and Machine Learning I

Tutorials and exercises - 2021-22
Focused set version 1.1

Eric Wolsztynski
eric.w@ucc.ie

Contents

Note: some of the questions in this document are treated in *Statistical Computing with R* by Maria L. Rizzo (2008), Chapman & Hall.

0 Practicing with R

Question 0.1

Our objective here is to manipulate a dataset and run summary statistics on the data.

(a) Set the R working directory as a desired location folder (named for instance `ST4060_practicals`). Then open an R script in your editor and save it in the working directory (named e.g. `ST4060_practical1-1.R`).

(b) Start by looking into the built-in dataset `airquality`.

- Load the dataset `airquality` in the R environment
- Have a quick peak at it using function `head(airquality)`
- Study the dataset further with `?airquality`, `dim()`, `names()` and `summary()`

(c) A detailed summary of the `Month` and `Day` variables is not necessary. Run a more detailed analysis of the dataset as follows:

- Apply `summary()` to the first column of data
- Display contents of the first 4 columns at once
- Display contents of columns 1, 2 and 4 only, at once
- Attach the dataset to the environment (for ease of use) using `attach()`
- View summary statistics for variable `Ozone`
- Compute its median, and identify a potential problem in the data
- Compare its mean and median (use argument `na.rm=TRUE`)
- Count the number of `NA`'s contained in vector `Ozone`
- Create a new vector called `Oz` that does not contain these `NA`'s
- Compute the following summary statistics for this new variable: `mean()`, `median()`, `sd()`, `var()`, 25th quantile

(d) Run analyses on a specific subset...

- Display a subset of the whole dataset that includes entries with `Wind`>10 and `Temp`<70 only
- Store this subset in a new variable (named e.g. `cool_n_windy`)
- Compute the correlation coefficient between `Wind` and `Temperature` information for this subset
- Now remove `NA`'s from the two vectors and recomputed the correlation

(e) It's time to visualize the data.

- Simply instruct to plot the whole dataset with `plot(airquality)`
- Compare this plot with `pairs(airquality[c(1:4)])`
- Plot a histogram of the `Ozone` variable using `hist()`

- Plot a scatterplot of `Ozone` vs `Temperature` using formula `Ozone~Temp`
- Plot a boxplot of `Ozone` as a function of `Month` (`Ozone~Month`)

(f) Now save a plot you'd like to present in a talk/report as a `pdf`. We plan on having two panels side-by-side containing the scatterplot and the boxplot obtained earlier. You may first like to create a directory called e.g. `output` in your ST4060 practicals directory, for tidiness.

- Open a pdf file connection using `pdf(...)` – this will create the file on disk and make it available for R to write into
- Prepare your plotting window using
  `par(mfrow=c(1,2), font=2, font.lab=2, font.axis=2, pty='s')`
- Plot the scatterplot of `Ozone` against `Temperature`, with points marked by the symbol `'*'` and of `size 2`, add a main title `"Ozone vs Temperature"`, an x-label `"Temperature"`, and a y-label `"Ozone"`
- Plot the boxplot, with line width 2, filled with gray, and titled `"Nice boxplot"`
- Close the pdf file connection using `dev.off()`. Without this step the file will not be readable, i.e. it could not be used. Leaving the R session with an open connection will likely result in an I/O problem or error of some sort.

(g) Develop your own R function that computes the Inter-Quartile Range (IQR) for a given vector of values `x`, by filling up the blanks within the following structure:

```
iqr <- function(x){
      ...
}
```

Then test it! (Hint: `iqr(Oz)`)

(h) Clean up! This means detaching the dataset from the R environment using `detach()`; and/or using `rm()`.

Question 0.2

Our objective here is to create a short "video clip" from a few image frames, by implementing dedicated functions and using an array data object. Use the following six images: `cinq.pgm`, `quatre.pgm`, `trois.pgm`, `deux.pgm`, `un.pgm`, `zero.pgm`. Note that a `.pgm` file is editable; in these, the third row indicates image dimensions (e.g. in the form 662 310 for a 662 x 310 picture).

(a) Set the R working directory as a desired location folder. Ideally this folder will include another folder called `data`, which itself contains the image files. Otherwise adapt the file paths in your R instructions accordingly. Then open an R script in your editor and save it in the working directory (named e.g. `ST4060_practical1-2.R`).

(b) Start by looking into the data a bit:

- Scan the dimensions of `cinq.pgm` using `scan()` and save values in variable `dim5`
- Load image `cinq.pgm` into a variable named `mat5`, using functions `scan()` and `matrix()`

- View the matrix as an image using `image()`
- Analyse what's wrong with this plot (at least 3 things are!)

(c) A quick fix using `t()` inside the call to `image()` is not sufficient to fix the orientation. We need to define a process that we will apply to all such images in order to view them the way we want. Since we will repeat the same process, we implement it as an R function.

- Write a function called `flipv()` that flips a matrix (or image) vertically
- Write another function, called `mimage()`, that runs a "personalised" call to `image()`, using grayscale and removing axes
- Try these out on mat5 using instruction `mimage(flipv(mat5))`

(d) Repeat the process successively on `quatre.pgm`, `trois.pgm`, `deux.pgm`, `un.pgm` and `zero.pgm`. We may as well create a wrapping function wrap() that runs all required instructions in a nice short call, i.e. for example `wrap("cinq.pgm")`. Then run all calls at once:

```
wrap("cinq.pgm")
wrap("quatre.pgm")
wrap("trois.pgm")
wrap("deux.pgm")
wrap("un.pgm")
wrap("zero.pgm")
```

(e) Store all images into an array, using `flipv()`. This will allow to carry all frames in the one object. Then display each frame of the array successively using a `for()` loop. (E.g. from 5 down to 0.)

(f)
- Install package animation
- Load the package using `library(animation)`
- Create a function, say `gif.gen()`, that will be used in the final call below
- Generate a gif file using `saveGIF(gif.gen(),"mygif.gif",outdir="...")`
- Try out `saveHTML()` also!

(g) Want more? Look up package `rimage`...

1 Stochastic modelling and KDE's

1.1 Distribution models and simulation

Question 1.1 (Simulation)

Using a sample size of $N = 1000$, generate a sample of realizations of an $Exp(1/2)$ random variable, using an appropriate R command. Plot its histogram and overlay the curve of the theoretic probability distribution function, as well as the curve of a nonparametric density estimate for the sampling distribution.

Question 1.2 (Simulation)

(a) Implement pseudo-random generation of Huber's contamination model

$$f_\varepsilon(u) = \varepsilon\phi(u) + (1-\varepsilon)h(u)$$

where $\varepsilon \in (0,1)$, $\phi(u)$ denotes the Standard Normal distribution, and using the t-distribution with 3 degrees of freedom for $h()$. Generate 3 different samples of size $N = 100$ from Huber's contamination model, setting ε to be successively 0.95, 0.40 and 0.20. Provide the sample means and standard deviations, rounding off to 3 decimal places, for each of the three samples.

Note: you may write a function, e.g. `rhuber <- function(N,epsilon=0,dof=3){...}` out of convenience, but this is not a requirement.

(b) Create a 3-frame plot showing the histograms of each generated sample; the frames should be organised in 3 rows and 1 column and the ranges of the x-axes should be set equally for all 3 plots to allow for direct comparison.

(c) Create a dataframe that contains the 3 samples organised in columns, and specify names for each column so as to keep track of the value of epsilon used to generate same (e.g. `e095`, `e040` and `e020` could be used as names). Write this dataframe to a `.csv` file so that the file, once open (e.g. in Microsoft Excel), only shows 3 columns (i.e. it should not contain a first column with row numbers).

1.2 Nonparametric density estimation

Question 1.3 (kernel density estimation)

Generate samples of $n = 1000$ variates from the following distributions, and for each sample,

- compute its kernel density estimator (KDE),
- plot the KDE over the rug of sample points,
- overlay a line indicating the true underlying pdf used to generate the sample:

(a) a Normal $\mathcal{N}(2,1)$;

(b) a Student t-distribution with $t = 3$;

(c) an $Exp(2)$.

Any comments?

Question 1.4 (Multivariate density estimation)

(a) Compute and plot a 2D KDE for the Old Faithful dataset in R, using the appropriate function(s) from R's `KernSmooth` package.

(b) Compute and plot a nonparametric regression function estimate for the Old Faithful data using local polynomials from the same package.

Question 1.5 (Multivariate density estimation)

(a) Generate a random sample of size $n = 1000$ from the $\mathcal{N}_2(\mu, \Sigma)$ distribution.

(b) Fit a 2D (bivariate Normal) kernel density estimator to the sample (use R's `MASS::kde2d` function).

(c) Fit a 2D kernel density estimator to the sample using a product of univariate kernels, i.e. implement the KDE defined by

$$\hat{f}(u) = \frac{1}{n} \sum_{i=1}^{n} \prod_{j=1}^{d} \frac{1}{h_j} K\left(\frac{u^{(j)} - X_i^{(j)}}{h_j} \right)$$

with $d = 2$, $h_1 = h_2 = 0.5$, and u ranging over the (x, y) grid computed by `kde2d` in the previous question.

(d) For each KDE,

- plot the cloud of points and add a contour of the associated KDE;
- generate a perspective plot with angle $\theta = 30^o$;
- generate a perspective plot with angle $\theta = -60^o$.

2 Resampling methods

2.1 Monte Carlo

Question 2.1 (Monte-Carlo integration)

Compute a Monte Carlo estimate of

$$\theta = \int_2^4 e^{-x} dx$$

and compare your result with the exact value of the integral.

Question 2.2 (Monte-Carlo integration)

Use the Monte Carlo approach to evaluate the standard Normal cdf (assume $x \geq 0$ for simplicity):

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

Evaluate also the variance and 95% confidence interval associated with your estimate.

Question 2.3 (Monte-Carlo estimation)

(a) Implement a function that given an integer $k < n$ and a sample X, computes the trimmed mean of X, which is defined for an ordered sample $X_{(1)}, \ldots, X_{(n)}$ by

$$\bar{X}_{[-k]} = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} X_{(i)}$$

(b) Implement a Monte Carlo simulation using the standard Normal as sampling distribution, with $n = 20$ and $M = 1000$, to evaluate the distribution and MSE of this estimator under this model.

(c) Run a similar simulation using a Student t-distribution with 1 d.o.f. as samplig distribution, and compare output estimator performances.

Question 2.4 (Step-by-step question - do it yourself :)) (Monte-Carlo estimation)

With Monte Carlo repetitions, the objective is to set up M repetitions of a statistical experiment, where for each experiment:

- we generate a new sample of observations randomly,
- we perform a statistical analysis (including model fitting) on this sample,
- we store the results.

Once the M experiments are finished, we analyse the sample distribution of the parameter estimates, generate some plots and store some information. This approach is used in particular to approximate the asymptotic characteristics of some statistical procedure and benchmark several techniques in terms of their distribution. One example could be comparing two robust estimators for linear regression with heavy-tailed noise (e.g. log-Normal, Laplace)... For example we could use least squares (LS) and robust M-estimation (RM).

(a) Initialize the simulation parameters and storing variables.

- Let $N = 50$ (sample size) and $M = 100$ (number of Monte Carlo repetitions)
- Set $a = 7$ and $b = 3$ (resp. intercept and slope parameters in a linear model)
- Create the vector of regressors (design) `x = rep(c(1:5), N/5)`
- Set noise parameters to be $m = 0.5$ and $s = 1.2$ (mean and standard deviation)
- Set a random seed (e.g. `rseed=0`) for pseudo-random generation
- Allocate storage vectors `LSvec = RMvec = matrix(0,2,M)`
- Finally, run `set.seed(rseed)` and import libraries `MASS` and `VGAM`

(b) Implement the Monte Carlo repetitions. Create a for loop from 1 to M, within which you will:

- Generate a new sample of realizations of noise e from a $log\mathcal{N}(m, s)$
- Generate a new sample of observations `y = a + b*x + e`
- Estimate (a, b) via Least Squares for this new sample (use `lm` or `mylm`)
- Estimate (a, b) via robust M-estimation using `rlm`
- Store these estimates in the adequate vectors (use e.g. `rbind`)
- Note: you can also decide to store the samples of noise and observations for each loop – in case this may be useful later on, for instance

(c) Analyse the two sets of estimates.

- Create a plotting window with 2×3 panels with `par(mfrow=c(2,3))`
- Plot histograms for each set of estimates
- Plot nonparametric density estimates for each vector with `plot(density())`
- Compare the biases, variances and MSE's for all estimators
- Check that the tradeoff between bias and variance is found in the MSE
- Which approach seems more appropriate?
- Note: A similar analysis should be carried out with Normal noise to assess the potential loss incurred by the use of an M-estimator in place of the optimal Least Squares.

(d) Write outputs to file.

- Create a dataframe containing all outputs of interest, adding names for each column
- Write this dataframe to disk as an output `.csv` file, using `write.csv()`
- Test this file: view it in Excel

- Test again: load it up using `read.csv()` and recompute the biases as a check

Question 2.5 (Monte-Carlo estimation)

We aim to demonstrate the statistical properties of two estimators of the standard deviation via Monte Carlo simulations. Given a sample of observations $\{X_1, \ldots, X_N\}$, and using the sample mean

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i$$

the two estimators considered are denoted s and $\hat{\sigma}$ and are defined by

$$s^2 = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \bar{X})^2$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^{N} (X_i - \bar{X})^2$$

(a) Implement a Monte Carlo experiment to generate M samples of size N from a Normal $\mathcal{N}(0, 2^2)$ distribution, and compute sample standard deviations estimates for s and $\hat{\sigma}$, for each of these samples using the formulas above. Include also a computation of the sample standard deviation using R's function `sd` for each Monte Carlo sample generated. Generate $M = 1000$ Monte Carlo estimates for four different sample sizes: $N \in (10, 20, 50, 100)$. Finally, prepare also a 4-panel plot window (using `par(mfrow=c(2,2))`) for the plots requested below.

(b) Check whether R's function `sd` corresponds to one of the estimators s or $\hat{\sigma}$.

(c) Plot the histograms for both Monte Carlo samples for s and $\hat{\sigma}$ for the case $N = 10$, in separate plot panels. Also provide values, rounded to two decimal places, for the Monte Carlo biases and variances for both estimators and for $N = 10$.

(d) Based on your analysis in this question, give your conclusions on the comparison of bias and variance of these two estimators, s and $\hat{\sigma}$.

(e) Plot boxplots corresponding to all four sample sizes, in order to show the progression of the distribution of each estimator with respect to sample size N. Display the plot corresponding to s in one plot panel, and that corresponding to $\hat{\sigma}$ in the other of the last two remaining plot panel.

Question 2.6 (Monte-Carlo estimation)

In this question, we aim to assess whether the rate of convergence of the sample mean of a χ^2-distributed sample may depend upon the number of degrees of freedom associated with the distribution.

(a) Implement `M=100` Monte Carlo repetitions of an experiment such that:

- All values in $\{2, 4, 10\}$ are successively used as number of degrees of freedom `ndf`;

- A sample size of `n=100` is used;
- For each value of `ndf`, `M=100` samples `x` of a χ^2-distribution with `ndf` degrees of freedom are generated and their mean stored in an array `ms` of dimensions `M`×`3`.

(b) After running the implementation in part (a), generate a figure showing the boxplots for the distributions of means corresponding to each number of degrees of freedom. Comment (briefly) on this figure. Can you observe a particular feature of the χ^2-distribution?

Question 2.7

Note: parts of this quesiton are also covered in Section 2.

(a) Implement pseudo-random generation of Huber's contamination model

$$f_\varepsilon(u) = \varepsilon\phi(u) + (1-\varepsilon)h(u)$$

where $\varepsilon \in (0,1)$, $\phi(u)$ denotes the Standard Normal distribution, and using the t-distribution with 3 degrees of freedom for $h()$. Generate 3 different samples of size $N = 100$ from Huber's contamination model, setting ε to be successively 0.95, 0.40 and 0.20. Provide the sample means and standard deviations, rounding off to 3 decimal places, for each of the three samples.

Note: you may write a function, e.g. `rhuber <- function(N,epsilon=0,dof=3){...}` out of convenience, but this is not a requirement.

(b) Create a 3-frame plot showing the histograms of each generated sample; the frames should be organised in 3 rows and 1 column and the ranges of the x-axes should be set equally for all 3 plots to allow for direct comparison.

(c) Create a dataframe that contains the 3 samples organised in columns, and specify names for each column so as to keep track of the value of epsilon used to generate same (e.g. `e095`, `e040` and `e020` could be used as names). Write this dataframe to a `.csv` file so that the file, once open (e.g. in Microsoft Excel), only shows 3 columns (i.e. it should not contain a first column with row numbers).

(d) Implement a Monte Carlo (MC) simulation in which you generate $M = 500$ samples of size $N = 100$ from the Huber $f_{0.40}(u)$ distribution (i.e. using $\varepsilon = 0.40$), and another M samples from the Normal $N(0,1)$ distribution. For each Monte Carlo repetition, compute and store the sample means and standard deviations of both the Normal and the Huber samples. Provide the averages of the MC samples of means and standard deviations for both distributions, rounding off all averaged values to 3 decimal places.

Note: If your implementation of Huber's model $f_\varepsilon(u)$ did not work out in (a), you may generate samples from Student's t-distribution with 3 degrees of freedom instead.

2.2 Bootstrapping

Question 2.8 (Bootstrap estimation of standard error)

Load the law school dataset in the `bootstrap` package, estimate the correlation between the two variables in this dataset, and evaluate the bootstrap estimate of the standard error associated with this estimation.

Question 2.9 (Bootstrap linear regression estimates)

Consider R's `cars` dataset.

(a) Obtain relevant regression parameter estimates for this dataset.

(b) Generate $M = 10000$ bootstrap estimates for these coefficients.

(c) Inspect the one-dimensional (i.e. marginal) distributions for all relevant bootstrap parameter estimates, and state your conclusions.

(d) Inspect the joint distribution for these sample parameter estimates, and state your conclusions.

Question 2.10 (Nonlinear estimation and bootstrapping)

Load the following data from R's `mtcars` dataset:

```
x = mtcars$disp
y = mtcars$mpg
```

(a) Task: fit an exponential model

$$Y_i = exp(\theta_1 + \theta_2 X_i) + \varepsilon_i, \qquad i = 1, \ldots, n, \qquad \varepsilon_i \text{ i.i.d.}$$

to the sample `y` using R's function `nls()`. Use initial values $\theta^{(0)} = (3, -0.01)$.
Required:

(i) quote the coefficient estimates for this model fit;

(ii) explain, based on numerical assessment but without performing any further computations, whether this model appropriately describes the relationship between `x` and `y`;

(iii) provide a plot showing the model fit *as a line* going through the data points. Plot the data points in black and the model fit in red.

(b) Name an alternative regression technique that would provide a nonlinear representation of the relationship between `x` and `y`, without assuming a particular model.

(c) Task: set the pseudo-random seed to 1 (R instruction `set.seed(1)`) and compute $B = 100$ bootstrap estimates for the model fit of (a).
Required:

(i) quote the bootstrap means and standard deviations for estimators $\hat{\theta}_1$ and $\hat{\theta}_2$;

(ii) quote the bootstrap estimate of the standard error for estimator $\hat{\theta}_1$;

(iii) quote a nonparametric 95% confidence intervals for model parameter θ_1;

(iv) comment on the confidence interval found in (c.iii).

Round off your numerical answers to 4 decimal places where applicable.

2.3 Cross-validation

Question 2.11 (Cross-validation frameworks)

Load the following data from dataset `Boston` in library `MASS`:

```
x = Boston[, c("crim", "indus", "rm", "tax")]
y = Boston$medv
```

You will probably need to coerce `x` into a matrix before passing into `lm`:

```
x = as.matrix(x)
lmo = lm(y~x)
summary(lmo)
```

This linear model seems alright, but what about its predictive performance?

(a) Perform a 50%-50% train-test split of the dataset. Fit the linear model on the training data, generate predictions from this model for the test data, and calculate the corresponding prediction Root Mean Square Error (RMSE).

(b) Implement Leave-One-Out CV on the data (`x`, `y`) and calculate the LOO-CV test set prediction RMSE estimate.

(c) Implement K-fold CV on the data (`x`, `y`) and calculate the K-fold CV test set prediction RMSE estimate, using K=5.

(d) Implement K-fold CV on the data (`x`, `y`) and calculate the K-fold CV test set prediction RMSE estimate, using K=10.

(e) Compare the prediction error estimates obtained from (a), (b), (c), and (d).

3 Estimation theory

3.1 Linear regression

Question 3.1 (Step-by-step question - do it yourself :))

The objective here is to implement linear regression analyses and explore basic aspects of stochastic modelling.

(a) Set the R working directory as a desired location folder. Then open an R script in your editor, and save it in the working directory, using a sensible name.

(b) We consider the in-built dataset faithful, which contains... well, look at `?faithful`. We want to fit the data as follows:

$$\text{eruptions} = \alpha + \beta \times \text{waiting} + \text{noise}$$

Let's look at the data first. Note: when plotting, think about presentation too.

- Plot eruptions against waiting times
- Fit a linear model to `eruptions~waiting` using `lm()`, and store its output in a variable called e.g. `lm.out`
- Display the coefficients α and β obtained from the fit
- Add a line $\alpha + \beta x$ to the plot with `abline()`
- Display a summary of the output of `lm()`
- Compare the components of `lm.out` and of `summary(lm.out)` with `names()`
- Store the value of the adjusted coefficient of determination (i.e. the adjusted R^2) associated with the linear fit

(c) Now we look at forecasting based on this linear fit.

- Develop a 95% confidence interval of the mean eruption duration for the waiting time of 80 minutes using

  ```
  predict( lm(...), newdata=data.frame(...), interval="confidence" )
  ```

- Generate a 5-step-ahead prediction, including corresponding confidence interval, for waiting times `max(waiting)+c(1:5)`
- In a plot window with extended x- and y-axes so as to include predicted values (use arguments `xlim=c(...,...)` and `ylim=c(...,...)` inside the call to `plot()`):
  - plot the actual data (eruptions against times)
  - add a line that indicates the linear fit
  - finally, add the 5 forecast points in red using `points()` and note whether they fit on the line or not

(d) We now carry out some significance tests, to assess whether there is a significant relationship between waiting times and eruptions. This significance is tested under the null hypothesis $H_0 : \beta = 0$.

- Display the coefficients of the output `lm.out`
- Identify what `summary(lm.out)$coefficients[,4]` corresponds to
- Learn more about it from the help page `?summary.lm`

We further run some diagnostic checks on the residuals:

- Prepare a 1×2 plotting window with `par(mfrow=c(1,2), ...)`
- Plot the residuals obtained from the linear fit `lm.out`
- Plot a QQ-plot of these residuals with `qqnorm()`, storing its output in `qqlm`
- Add the line corresponding to the linear fit `qqlm$y~qqlm$x` (e.g. in red)

(e) Write your own `lm()` function! This is good practice to learn to manipulate vector products. Create the structure for a function called `mylm()`, which takes two arguments (`x` and `y`), as follows:

```
mylm <- function(x, y){
# Returns my own linear fit ?
     ...
}
```

Inside the body of this function:

- compute `xm` and `ym`, centered versions of `x` and `y`
- Compute `b`, the linear estimate of β, defined by $(x^T x)^{-1}(x^T y)$, using `%*%`
- Compute the linear estimate of α, defined as the sample mean of `y-bx`
- Compute the residuals `y - a - bx`
- Create a list that contains 3 components, namely estimates for the intercept and slope, and the residuals
- Return this list as the last instruction in the body of your function `mylm()`

Now try it! Compare

```
lm(eruptions~waiting, data=faithful)
```

with

```
mylm(faithful$waiting , faithful$eruptions)
```

(f) Go further by trying out the following:

```
# Improve display...
summary.mylm <- function(out){
     print("")
     print("Coefficients:")
     print(paste("(Intercept)          x",sep=""))
     print(c(out$myintercept,out$mycoef))
```

```
}

# Test it!
summary.mylm( mylm(faithful$waiting,faithful$eruptions) )

# Improve (i.e. rewrite) this function to allow for plotting!
mylm <- function(x, y, DOPLOT=FALSE, ...){
     ...
     out = list(myintercept=a, mycoef=b, residuals=res)
     if(DOPLOT){
          plot(x, y, cex=1.2, ...)
          abline(a=a, b=b, col='red', lwd=1.5)
     }
     return(out)
}
# Test again!
mylm(faithful$waiting,faithful$eruptions)
mylm(faithful$waiting,faithful$eruptions,DOPLOT=T)
```

NB: the definition of function `mylm` that R will use is whichever one of the two definitions above that gets "run" last...

Question 3.2

Simulate a model with observations given for $i = 1, \ldots, N$ by

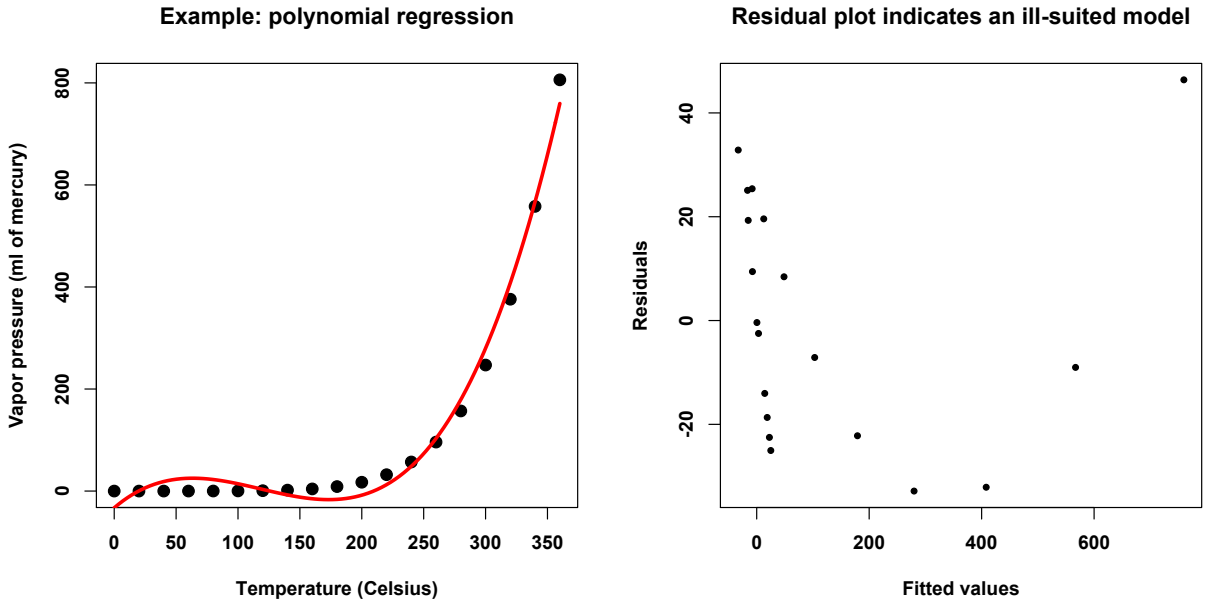
$$Y_i = \theta_0 + \theta_1 X_i + \varepsilon_i$$

where $X = \{1, 2, 5, 5.5, 9\}$, $\theta_0 = 3$, $\theta_1 = 1.5$, and $\varepsilon \sim N(0, 1.2)$, and using $N = 100$. Apply ordinary least squares estimation to your set of simulated observations. Then apply weighted least squares with weights $w = (.1, .1, .35, ..35, .1)$ and compare outputs. Comment on your results and how the comparison was carried out.

3.2 Polynomial and nonlinear regression

Question 3.3

`attach(pressure); x=temperature; y=pressure`
Implement the linear regression to obtain the following plot:



Question 3.4

Why does the following not "work"? Hint: complete the code to plot the objective function.

```
par(mfrow=c(1,2),font.lab= 2, font.axis=2)
model <- function(x, theta){
     return(theta*x)
}
crit <- function(theta,x,y){
# must have theta as 1st parameter and return a single value...
     return( sum( y-model(x,theta) ) )
}

thbar = 1.8
x = rep(c(1,3,7,8), len=100)
y = model(x,thbar) + rnorm(x)

plot(x, y, pch=20, cex=1.5, main="optim example 1: data")
points(x,model(x,thbar),col='green',pch=20,cex=2)

(optim.out <- optim(par=c(1), fn=crit, x=x, y=y,
     method="L-BFGS-B", bluelower=c(0.01), upper=c(3.05)))
```

Question 3.5

Implement a simulation of the nonlinear model

$$Y = \exp(-\theta X) + \varepsilon$$

and minimize the sum of least squares for this model using `optim`.

Question 3.6

Implement the same simulation and minimize the sum of least squares for this model using `optimx`, this time.

3.3 Regularisation

Question 3.7

The `Blood Pressure.txt` dataset contains measurements of systolic blood pressure, age, waist circumference, cholesterol and BMI index, for 75 subjects (variable `PatientID` is dropped from the dataset). Four linear regression models have been fit to the dataset in order to describe the variable of interest systolic blood pressure in terms of the other 4 variables. Regularisation parameters have been calibrated so that their corresponding models yield optimal regularisation. Table 3.7 below presents output from these 4 model fits (where "e-net" stands for "elastic net"). Analyse and comment on these results, for example in terms of:

- The effect of each of the regularisation schemes;
- Increase in model fit error;
- Potential "issues" or "challenges" within the dataset;
- Overall impact of, or necessity for regularisation for this data;
- Limitations of the output presented.

	GLM	ridge	e.net	LASSO
(Intercept)	56.710	64.703	65.854	64.707
Age	0.200	0.220	0.146	0.118
Waist	0.557	0.356	0.482	0.520
Cholesterol	0.003	0.004	0.000	0.000
BMI	0.030	0.378	0.048	0.000
Errors	139.76	141.96	143.26	143.26

Table 1: Table for Question 1.

4 Smoothing

4.1 Splines

Question 4.1 (NB: For this one we need to re-generate the dataset)

Read in the dataset `rough_Makeham_rates.csv`, which contains a sample of simulated Makeham mortality rates with additive noise. For information, the procedure used to generate this sample is as follows:

```
LT = read.table("irl_lifetable_2005.txt", sep=",", header=TRUE)
# keep only the first 106 rows from LT:
SLT = LT[c(1:106),]
mx = SLT[,8]
x = SLT$Age # age grid
# roughly fit a Makeham model to this data:
onls = nls(mx~A+B*c^x,start=list(A=.0003, B=.00002, c=1.108))
ABc = summary(onls)$coef[,1]
# now add noise to the fitted f.o.m. curve:
set.seed(1)
x = seq(0, 110, by=1)
mx = ABc[1]+ABc[2]*ABc[3]^x
mxn = mx
s1 = which(x<86)
s2 = which(x>85)
mxn[s1] = pmax(0.005,mx[s1]+rnorm(length(s1),0,.03))
mxn[s2] = mx[s2]+rnorm(length(s2),0,.06)
dat = cbind(x,mx,mxn)
```

This means that the data comes from a Makeham model of mortality rates μ_x (as a function of age x)

$$\mu_x = A + Bc^x, \quad x = 0, 1, \ldots$$

with true parameters $A = .0003$, $B = .00002$, $c = 1.108$, which was perturbed by some mostly-symmetric additive noise.

(a) Fit a Makeham model to the sample, using `nls()` with initial values `(A=.0003, B=.00002, c=1.108)`.

(b) Repeat the previous step, but on the smoothed curve obtained from a generic P-spline. Compare the mean squared errors of the model parameter estimates obtained from both nonlinear fits.

Question 4.2

Import the simulated dataset of female mortality rates from a hypothetical cohort of life insurance policyholders into R from `insdata.csv`:

```
dat = read.csv("insdata.csv")
age = dat$Age
mF = dat$mF
```

(a) Compute a first P-spline where the smoothing control parameter is set by cross-validation. Create a plot of the dataset (black dots) along with the P-spline (red solid curve).

(b) Compute a second P-spline for the same dataset, where the smoothing control parameter is half that of the P-spline obtained in (a).

(c) Show that the two P-spline outputs are evaluated over the same points on the x-axis.

(d) Compute and compare the MSE's for the P-splines obtained in (a) and (b). Comment on their difference, and propose a reason as to why they differ.

(e) Compute a B-spline basis using the first, second and third quartiles of the age data as knots. Create a plot of this B-spline basis.

(f) Quote the coordinates of a policyholder aged 60 on the B-spline basis computed in (e), up to four decimal places. Indicate these coordinates with a line on the plot obtained in (e).

(g) Compute the corresponding B-spline for the (age, mF) data. Find the value of the output coefficients for the B-spline expression.

(h) Compare and comment on the MSE obtained for that B-spline with the MSE's obtained from the two P-splines obtained in (d).

(i) Compute interpolations for all ages within the range of age data, using respectively P-spline smoothing and local polynomial regression. Plot the interpolated points over the observations, using red for P-spline values and blue for local polynomial regression values. Note the standard deviations of each of the interpolated samples.

Question 4.3

Fit a B-spline and a P-spline to the `Prestige` dataset (`library(cars)`). You'll need to run

```
library(splines)
library(car)
```

Plot the corresponding curves and compare the MSEs.