MA4605 2015 Lab Class E

This week we will look at three key objectives

1. Creating and Inspecting Diagnostic Plots for Linear Models

2. Download and install an R package

3. Fitting a Robust Regression Model

Part 1 : Diagnostic Plots

- To extract the residuals from a fitted model, we use the `resid()` command, specifying the name of the model.
- Recall last week we fitted several models for the Cheeses data set . (To reload it, run the ***Lab D Setup script*** again).
- We will mostly look at the ***Multiple Linear Regression Models*** specifically.

```
Fit1 = lm(Taste ~ Acetic + H2S)
Fit2 = lm(Taste ~ Acetic + Lactic)
Fit3 = lm(Taste ~ H2S + Lactic)
FitAll = lm(Taste ~ Acetic + H2S + Lactic)
```

```
Fit2res = resid(Fit2)

# Type in "Fit2res" to get a sense of the data.
```

Exercise: Compute the variance of the Fit2 residuals (use the `var()` command)

To inspect the diagnostic plots, simply use the plot command, specify the name of the fitted model.

```
plot(Fit2)
# Hit Return after inspecting each screen
```

Four plots are automatically presented but in fact there are six altogether. These plots will indicate influential points. Look for the data points that are specifically numbers.

To specify an individual plot to be presented, type each of the following command. Run each command separately.

On your submission sheet make a sketch for diagnostic plots 1, 3 and 4.

In your sketches detail the location of the identified influential points. You don't not have to be precise about unnamed data points.

```
plot(Fit2, which=c(1))

plot(Fit2, which=c(2))

plot(Fit2, which=c(3))

plot(Fit2, which=c(4))

plot(Fit2, which=c(5))

plot(Fit2, which=c(6))
```

For the data points that were identified as influential by the diagnostic plots, we can determine the Cook's Distance measure, using the command `cooks.distance()`.

```
cooks.distance(Fit2)

# Make it easier to read the output
# specify 6 decimal places only.

round(  cooks.distance(Fit2)   ,6)
```

Part 2 : Packages

Packages are collections of R functions, data, and compiled code in a well-defined format. The directory where packages are stored is called the library. R comes with a standard set of packages (for example the MASS function). To activate a package, the function is library(), specifying the name of the package.

```
library()    # see all packages installed
search()     # see packages currently loaded

library(MASS) #  MASS is now activated.

# ( we can now create rlm() models )
```

Installing Packages

Others are available for download and installation.To install a package, we use the command `install.packages()`.You can expand the types of analyses you do be adding other packages. A complete list of contributed packages is available from CRAN.

Follow these steps:

1. Download and install a package (you only need to do this once).
2. To use the package, invoke the library(package) command to load it into the current session.

(You need to do this once in each session, unless you customize your environment to automatically load it each time.)

On MS Windows:

1. Choose Install Packages from the Packages menu.
2. Select a CRAN Mirror. (e.g. Ireland)
3. Select a package. (e.g. chemCal)
4. Then use the **`library(package)`** function to load it for use. (e.g. **`library(chemCal)`**)

```
# quotation marks

install.packages("MethComp")
install.packages("chemCal")

# no quotation marks

library(MethComp)
library(ChemCal)
```

If we have time, we will try out the demonstration code on the packages reference document.

chemCal provides simple functions for plotting linear calibration functions and estimating standard errors for measurements according to the ***Handbook of Chemometrics and Qualimetrics: Part A*** by Massart et al.

There are also functions estimating the limit of detection (LOD) and limit of quantification (LOQ).

The functions work on model objects from - optionally weighted - linear regression (lm) or robust linear regression (rlm from the MASS package).

Part 3: Fitting Robust Regression Models

We can create a ***Robust Linear Regression*** module using the `rlm()` command from the MASS package.

The construction is almost identical to that of an `lm()` model , such as the ones we have fitted last week.

```
library(MASS)
Fit2.rr = rlm(Taste ~ Acetic + Lactic)
summary(Fit2.rr)
```

Exercise: Write down the regression equation for this model.

To find out the weights given to each point, we use the following code. We can also sort these weights in ascending order (if necessary).

```
Weights( Fit2.rr )
```

Was robust regression necessary here?