STATS 3001 Statistical Modelling III
Practical 2
Week 3, Semester 1, 2018

The following exercises are intended to provide an indication of some of the considerations a statistician makes in the process of analysing data (cleaning/exploration/model fitting/interpretation) and also to explore model specification in R.

The dataset `att.csv` is available on MyUni and contains information on 102 software projects undertaken at AT&T between 1986 and 1991.[1]

The dataset contains the total work hours spent on the software project, `hours` the outcome variable, and potential predictor variables. The purpose of the analysis is to see whether the cost of a software project can be reliably predicted by variables known before the project is undertaken. The variables in `att.csv` are summarised in the table below.

Variable	Description
`hours`	The total work hours spent on the software project.
`fpoints`†	The count of 'functional points' identified in the software: a measure of the software complexity.
`os`	The operating system used: `"Unix"` or `"MVS"`.
`db`	The database management system used: `"IDMS"`, `"IMS"`, `"INFORMIX"`, `"INGRESS"` or `"Other"`.
`lang`	The programming language used: `"C"`, `"COBOL"`, `"PLI"` or `"Other"`.
`noother`	A boolean value that is `TRUE` when no `"Other"` for database or the language was specified. Is `FALSE` if either the database or the language was `"Other"`.

†A measure of the functionality the software provides to a user: https://en.wikipedia.org/wiki/Function_point

Perform the following steps:

(1) Read the data into R. Call the imported data frame `att`. Look at the first six rows of the dataset by using the `head()` function.

```
att<-read.csv("att.csv")
head(att)

##   hours fpoints  os    db  lang noother
## 1 15000    1059 MVS Other COBOL   FALSE
## 2  1850     234 MVS Other COBOL   FALSE
## 3 13033    1533 MVS Other COBOL   FALSE
## 4 11742     339 MVS   IMS COBOL    TRUE
## 5   283     205 MVS Other     C   FALSE
## 6 17992     420 MVS  IDMS COBOL    TRUE
```

[1] https://en.wikipedia.org/wiki/AT%26T_Corporation

(2) Have a look at the data. Is there any cleaning of the data that should be done (data entry errors)? Hint: yes, there are three, I altered the dataset from its original form. You may want to use the function `summary()` to help you (this is especially the case when you have thousands if not more rows).

(a) If you find any `NA` values, is there redundant information in the dataset that will allow you to deduce what the value(s) should be or can you simply 'impute' the most likely value based on the structure of the data?

Note: to change individual values in a `data.frame` you can write:

```
dataf[i,j]<-...
```

where

- `dataf` is the `data.frame`;
- `i` is the row;
- `j` is the column; and
- `...` is the new value you want to assign to the ij^{th} element of `dataf`.

If you are inserting characters (i.e., a 'string') into the `data.frame`, don't forget the quotes around the characters (e.g., `"New value"`). Also note you can print rows and columns of a `data.frame` using:

```
dataf[5,]        # print row 5
dataf[,2]        # print column 2
dataf[,"hours"]  # print column named "hours"
dataf$hours      # another way to print the hours column
```

(b) If there are any data entry errors that have an obvious and sensible replacement, replace the old incorrect value.

(3) Now let's look at the data graphically so we can get a feel for the relationships between the variables.

(a) You can get pairwise plots of the variables in `att` by using the function `pairs()` and including `att` as the input argument to the function.

(b) The `pairs()` function can be hard to interpret if we have a combination of continuous and categorical variables, so we will manually decide on the appropriate plots between variables at this stage. **Note**: using `pairs()` is fine for your Assignment 1.

We are most interested in the relationships between the predictor variables and the outcome. There is an excellent package `ggplot2` we will use to produce the following plots.[2] It makes plots easier to specify and, on occasion, can produce plots worthy of sharing with friends and family. Given the package is installed on your machine, you can load this package to your R-session by specifying:

```
library(ggplot2)
```

(c) Produce a plot between the two continuous variables of `hours` and `fpoints`. Using `ggplot2` this can be done like so:

```
ggplot(att,aes(fpoints,hours)) + geom_point()
```

Describe the relationship between `hours` and `fpoints`.

Note the way plots are specified here are of the form:

```
ggplot(dataset,aes(x_variable,y_variable)) + geom_xxx()
```

The first argument to `ggplot()` is always the dataset to be used. The next argument is the 'aesthetics' which relate to the features/variables from the dataset we want to include in the plot; these are things like x and y variables and variables to control the colouring etc. Aesthetics are wrapped in the `aes()` function to specify they are aesthetics of the plot. Just running

```
ggplot(dataset,aes(x_variable,y_variable))
```

will only give you a blank plot. This is why we will 'add' geometries like the point 'geometry', that is:

```
 + geom_point()
```

Plotting 'points' is not the only geometry available of course, for example you will want to use the 'boxplot' geometry (`geom_boxplot`) when plotting a continuous variable against a categorical variable.

[2] http://docs.ggplot2.org/current/

(d) Let's make our plot more informative and prettier by specifying that the points themselves are to be coloured based on `lang` used and the points are shaped based on the `db` used:

```
# pass data (first arg) and x,y variables in aes()
ggplot(att,aes(fpoints,hours)) +
   geom_point(aes(col=lang,shape=db))

# aes(col=lang,shape=db) is specified in geom_point but
# col=lang,shape=db could be included in the first aes() statement
```

(e) Now you have been given enough information to create boxplots to depict the relationships between the outcome `hours` and a categorical variable. Create four separate boxplot plots relating to each of the categorical predictors, with `hours` on the y-axis and a categorical variable on the x-axis. Which predictors seem to have a relationship with the outcome variable?

(f) To look at the relationship between the categorical variables, you may want to use the `table()` function. For example, by running

```
with(att,table(os,db))
```

you are creating a cross-tabulation between `os` and `db`. The `with(att,...)` specifies that vectors `os` and `db` can be found in the `att` dataset.

Note that `table()` isn't restricted to only two variables. Try three or four variables to see what is returned.

Produce cross-tabs between the categorical predictor variables to get a feel for their relationships. Are there any combinations of levels in the categorical variables that do not exist in the data? What is the minimum number of observations required to have all possible combinations of the categorical variable values present in the data?

(4) We will now look at design matrices for different model specifications using the `att` data.

(a) We want to create a design matrix X for the usual $\boldsymbol{Y} = X\boldsymbol{\beta} + \boldsymbol{\mathcal{E}}$ model using `db`, `lang` and `noother` as the predictors. We can create the design matrix in R called `X` using the command:

```
X<-model.matrix(~ db + lang + noother, data=att)
head(X) # have a look at the object
```

Note that the intercept term is included in the design matrix by default.

(b) How have the levels of the categorical variables been represented in the design matrix?

(c) How many columns are there in the design matrix? How many rows?

(d) Can one of these columns be generated from the information contained in the others? That is, is there linear dependence in the columns of X?

(e) If the columns of X are denoted $\boldsymbol{x}_j$ for $j = 1, 2, \ldots, p$, write out an equation specifying one column as a linear combination of other columns. e.g.,

$$\boldsymbol{x}_7 = \boldsymbol{x}_1 - \boldsymbol{x}_6 + \boldsymbol{x}_8.$$

(f) To check whether any design matrix has linearly dependent columns, you could solve $X\boldsymbol{\alpha} = \mathbf{0}$ algebraically to see whether a non-zero $\boldsymbol{\alpha}$ is a solution (i.e. $\boldsymbol{\alpha} \neq \mathbf{0}$).

However, this would be hard/tedious/prone to error. How can you check whether that design matrix has linearly dependent columns in R easily? Would the function to produce a matrix's determinant (`det()`) be useful?

i. Try removing the `noother` column from your design matrix (you may want to create another model matrix called `X0`, say, that doesn't include `noother`). Are the columns linearly independent now?

ii. Check the output when running the following:

```
summary(lm(hours ~ db + lang + noother, data=att))
summary(lm(hours ~ db + lang , data=att))
```

How has R dealt with the linear dependence in the design matrix?

(5) We will now fit some linear models and assess their fit.

(a) Create a model (called `lm0`) in R using the command:

```
lm0 <- lm(hours ~ fpoints + os + db + lang ,data=att)
```

You can observe the model fit using `summary(lm0)`. Look at the diagnostic plots using the commands:

```
par(mfrow=c(2,2)) # create a lattice in the plot window to contain 2x2 plots
plot(lm0)
par(mfrow=c(1,1)) # reset plotting device to be 1x1
```

For this exercise, we will consider the diagnostic plots produced by `plot(lm)` to compare the model fits.

(b) Now create a model called `lm1` that is the same as `lm0` except the outcome `hours` is (natural) log transformed. Also create the model called `lm2` that is the same as `lm0` except the outcome `hours` and the predictor `fpoints` are (natural) log transformed.

Look at the diagnostic plots for the models `lm1` and `lm2` and decide whether `lm0`, `lm1` or `lm2` seem to have the most appropriate fit.

(c) We can use the function `anova()` to perform an F-test to test whether the inclusion of variables in the model are warranted. For example, we can determine whether `os` should be included in the `lm2` model by testing the null hypothesis

$$H_0\colon\ \beta_{\texttt{osUnix}} = 0.$$

This can be done in R via the commands:

```
lm2a<-lm(log(hours) ~ log(fpoints)  + db + lang      ,data=att)
lm2 <-lm(log(hours) ~ log(fpoints)  + db + lang + os ,data=att)
anova(lm2a,lm2)
```

What is your conclusion?

(d) Is the p-value for the F-test the same as the p-value for the t-test corresponding to the $\hat{\beta}_{\texttt{osUnix}}$ when you run `summary(lm2)`? Are there other relationships you can see?

(e) Now test whether `lang` should be included in the model by testing the null hypothesis H_0: $\beta_{\texttt{langCOBOL}} = \beta_{\texttt{langOther}} = \beta_{\texttt{langPLI}} = 0$ in R using the two model specifications provided below. What is your conclusion?

```
lm3a<-lm(log(hours) ~ log(fpoints)  + db        ,data=att)
lm3 <-lm(log(hours) ~ log(fpoints)  + db + lang ,data=att)
anova(lm3a,lm3)
```