Analysis of the mtcars dataset

Marco Totolo

Summary

With reference to the mtcars dataset, we are interested to answer to the following points:

1. Is an automatic or manual transmission better for MPG?
2. Quantify the MPG difference between automatic and manual transmissions.

We have tried to fit a model taking into account possible confounding variables (ignoring interaction terms).

We have found that weight, number of cylinders and horsepower are the most influential factors for mpg. A manual transmission also seems to have a positive influence on mpg but the associated regression coefficient isn't statistically significant and therefore we can't answer question 2.

Exploratory Analysis

```r
data(mtcars)
dim(mtcars)
```

```
## [1] 32 11
```

We can see that the dataset is composed of 32 observations of 11 variables. We'll first type cast the categorical variables and then plot a preliminary graph of mpg vs. am (Figure 1). We can see that there seems to be a dependence of mpg from am, but possibly there are further confounding variables.

```r
mtcars$am<-as.factor(mtcars$am)
levels(mtcars$am)<-list(automatic="0",manual="1")
mtcars$vs<-as.factor(mtcars$vs)
mtcars$cyl<-as.factor(mtcars$cyl)
mtcars$gear<-as.factor(mtcars$gear)
mtcars$carb<-as.factor(mtcars$carb)
```

Modelling

Let's try first to fit directly mpg to am:

```r
fit1<-lm(mpg~am,mtcars)
summary(fit1)$adj.r.squared
```

```
## [1] 0.3384589
```

We can see that the R squared is quite low, most of the variance is left unexplained. This suggests that there could be one or more confounding variables. There are different strategies to find which variables to add to our model. We could update our model one variable at a time and see if our R squared improves. Or we could use an automated process to do it for us, for example using the step() function that minimizes the AIC factor.

We preferred defining a function that, starting from a fit that includes all of the possible variables, drops one variable at a time based on their relevance until we are left with the most relevant ones. You can find the definition of this function in the Appendix. Note that in this case the resulting fit is the same as the one we would obtain with the step() function.

```
fit2<-mystep(mtcars,"mpg",keep="am")
summary(fit2)$coeff
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 33.70832390 2.60488618 12.940421 7.733392e-13
## cyl6        -3.03134449 1.40728351 -2.154040 4.068272e-02
## cyl8        -2.16367532 2.28425172 -0.947214 3.522509e-01
## hp          -0.03210943 0.01369257 -2.345025 2.693461e-02
## wt          -2.49682942 0.88558779 -2.819404 9.081408e-03
## ammanual     1.80921138 1.39630450  1.295714 2.064597e-01
```

```
summary(fit2)$adj.r.squared
```

```
## [1] 0.8400875
```

We can see now that the R squared is much better. In this model the influence of having a manual transmission on mpg consumption is 1.8 mile/gallon. However, note that the p-value (>0.2) tells us that the coefficient for the associated predictor is not statistically significant, so in the end we can't satisfactory quantify the influence of the type of transmission on mpg.

Let's turn now to the diagnostics for the model. The residuals distribution (Figure 2) and qqplot (Figure 3) show that the residual distribution is quite normal. The Shapiro-Wilk test for normality also has a satisfactory value (>0.1)

```
shapiro.test(fit2$residuals)$p.value
```

```
## [1] 0.4478562
```

The hat-values are all quite similar, with tolerable exceptions:

```
as.vector(round(hatvalues(fit2), 3))
```

```
##  [1] 0.234 0.250 0.112 0.184 0.137 0.176 0.197 0.200 0.231 0.184 0.184
## [12] 0.093 0.101 0.098 0.250 0.294 0.261 0.133 0.161 0.128 0.278 0.174
## [23] 0.183 0.153 0.102 0.125 0.105 0.176 0.230 0.232 0.471 0.164
```

Conclusions

We have found a satisfactory linear model for the mtcars dataset, predicting mpg from weight, number of cylinders, horse power and automatic transmission type. Unfortunately the coefficient for this last parameter is not statistically significant, therefore we can't quantify its influence on mpg. This is most probably because cars with manual transmission tend to be smaller and with fewer cylinders. See Figure 4 for a comparison of all the variables of the model.

Appendix

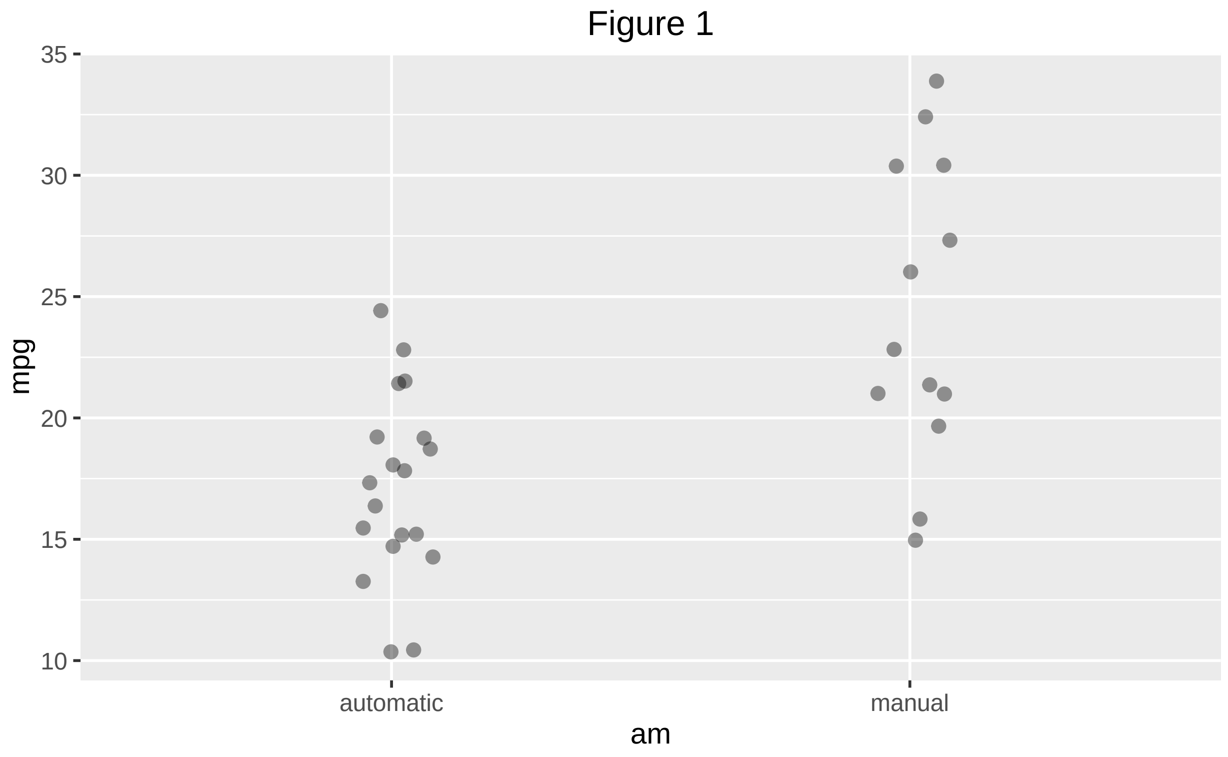


Figure 1
35
30
25
20
15
10
mpg
automatic
manual
am

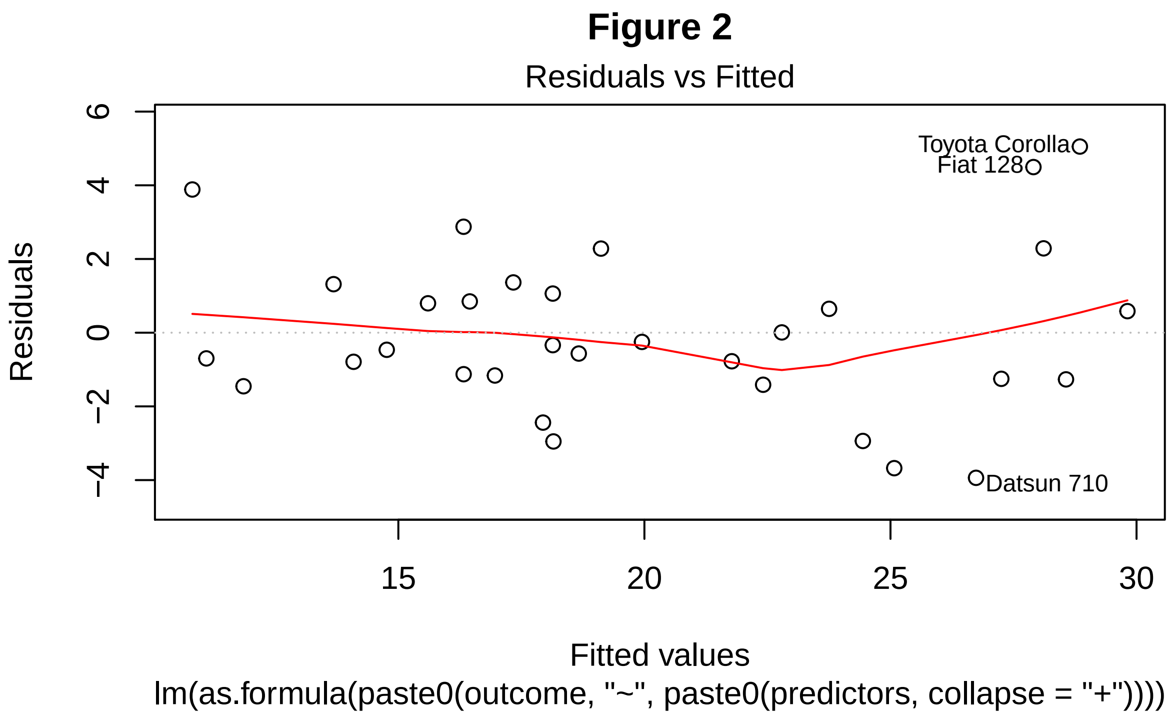


Figure 2
Residuals vs Fitted
6
4
2
0
−2
−4
Residuals
Toyota Corolla
Fiat 128
Datsun 710
15
20
25
30
Fitted values
lm(as.formula(paste0(outcome, "~", paste0(predictors, collapse = "+"))))

Figure 3

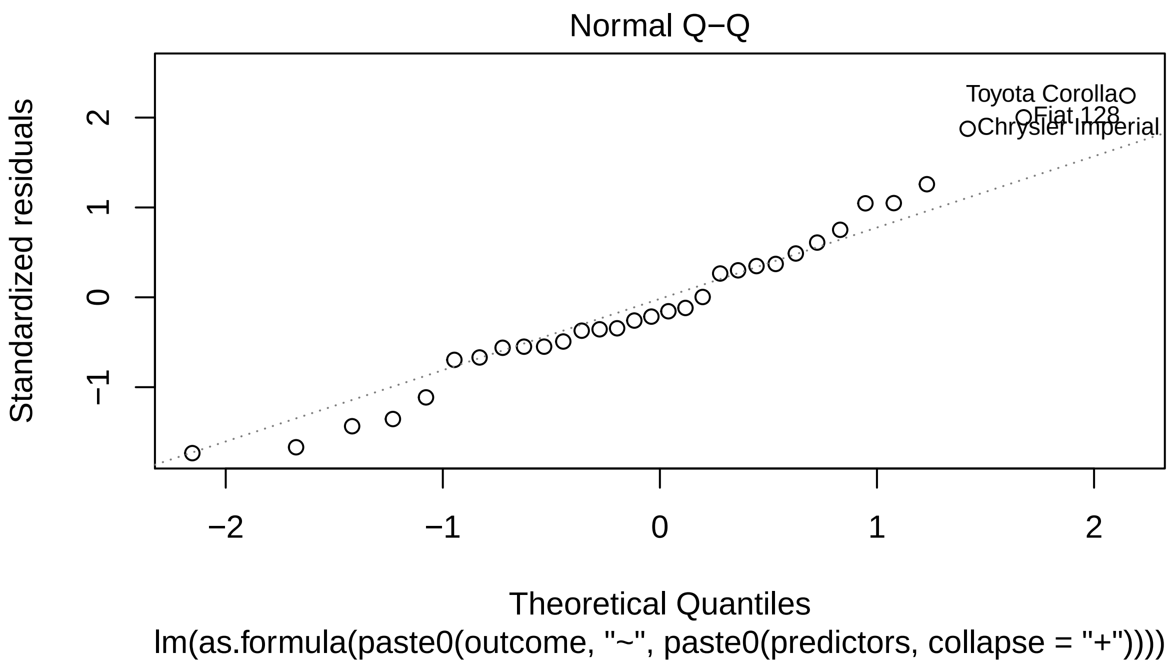
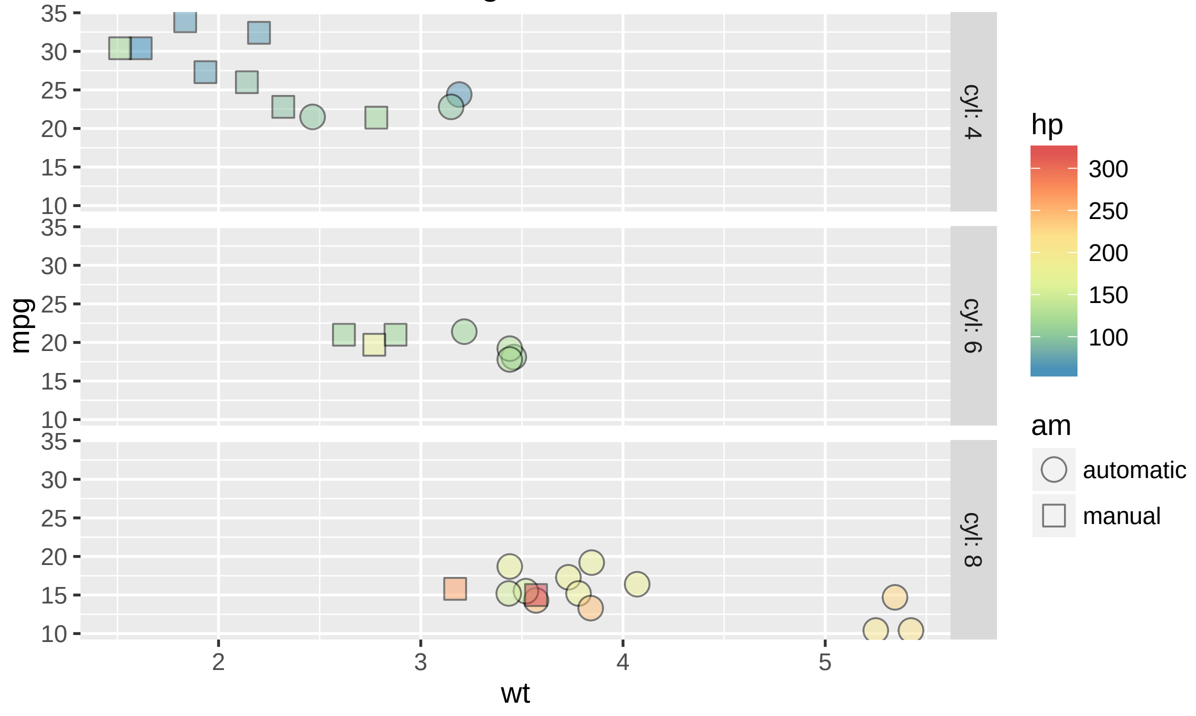


Figure 4



Definition of mystep function

```r
# parameters:
# -df: dataframe for the linerar model
# -outcome: string name of the outcome variable
# -threshold: pvalue level under which the regressors are considered significant
# -keep: string or string vector of variable names to keep in the model, independently of threshold
# -returns a variable of class "lm"
mystep <- function(df,outcome,threshold=0.05,keep=NULL) {
      # get the predictor list
      predictors<-names(df)[names(df)!=outcome]
      # dummy variables gets a special treatment later
      f.predictors<-names(df)[sapply(df,is.factor)&names(df)!=outcome]
      # create first fit with all variables
      for (i in 1:length(predictors)-1) {
            fit<-lm(as.formula(paste0(outcome,"~",paste0(predictors,collapse="+"))),df)
            #get the pvalues
            pvalues<-summary(fit)$coefficients[,4]
            #drop the intercept value
            pvalues<-pvalues[2:length(pvalues)]
            #for factor variables, consider the lowest pvalue
            for (j in f.predictors) {
                  # get the min
                  val<-min(pvalues[grepl(paste0("^",j),names(pvalues))],na.rm=T)
                   # drop all values
                  pvalues<-pvalues[!grepl(paste0("^",j),names(pvalues))]
                  # add the minimum
                  pvalues<-append(pvalues,val)
                  # updte names
                  names(pvalues)[length(names(pvalues))]<-j
            }
            #exclude the variables to keep from the vector
            for (k in keep) {
                  pvalues<-subset(pvalues,!grepl(paste0("^",k),names(pvalues)))
            }
            #check if there's no non significant predictor
            if (max(pvalues)<threshold) {
                  return(fit)
            }
            #find the less significant predictor
            todrop<-names(pvalues)[which.max(pvalues)]
            #drop it from the list
            predictors<-predictors[predictors!=todrop]
            f.predictors<-f.predictors[f.predictors!=todrop]
      }
      return(fit)
}
```