STATISTICAL MODELLING Part IIC, Michaelmas 2021
Practical 4: More on linear regression By courtesy of Dr. S. A. Bacallado and Dr. R. Shah

Model selection

Reload the house prices data from Practical 3.

```
> file_path <- "http://www.statslab.cam.ac.uk/~rds37/teaching/statistical_modelling/"
> HousePrices <- read.csv(paste0(file_path, "HousePrices.csv"))
> HousePricesLM1 <- lm(Sale.price ~ ., data = HousePrices)
> summary(HousePricesLM)
```

The results of this analysis on the houses data is not totally satisfactory. The p-values for the null hypotheses that exclude each of the variables `Bedrooms`, `Lot.size` and `Property.tax` are rather large. A simpler model may also explain the data adequately. The coefficient estimates in such a simpler model will have greater precision and predictions will be more accurate. Let us fit a model without them.

```
> attach(HousePrices)
> HousePricesLM2 <- lm(Sale.price ~ Bathrooms + Living.area + Year.built)
> summary(HousePricesLM2)
```

Now we have a smaller model, call this S_0, where each of the variables appears to be indispensable to the model fit (in one-by-one t-tests; see next paragraph). For example, if we consider an even smaller model without `Year.built`, the chance of observing data as extreme as ours (in terms of carrying evidence against the validity of the even simpler model and in favour of the model S_0) is a rather remote 3.7%. Note however, that this is different from the p-value corresponding to `Year.built` observed in the full model (`HousePricesLM1`). Of course, these different p-values correspond to *different* null hypotheses, so this is certainly to be expected.
Lecturer's note: The reasoning above is common in the practice, but as I mentioned in the lectures, the purpose of such *ad hoc* model selection procedure is unclear. If the goal is to test the significance of some coefficients, post model-selection inference is needed to control statistical errors. If the goal is making better predictions by achieving a better bias-variance tradeoof, criteria like Mallows' C_p or cross-validation should be used. If the goal is to pick the "correct" model, BIC could be used.
Is our smaller model better than the original full model? Although we have only discarded variables that seemed insignificant, the t-tests performed only consider the individual contribution of each variable to the model fit. It may well be the case that a group of individually insignificant variables are very significant as a group. A rather extreme case of this arises in the following artificial scenario.

```
> set.seed(1)
> X1 <- rnorm(50)
> X2 <- X1 + 0.05 * rnorm(50)
> y <- 1 + X1 + X2 + rnorm(50)
> summary(lm(y ~ X1 + X2))
```

Neither of the variables above appear significant, though the result of the F-test in the final line of the summary output suggests that the model that omits both variables is not consistent with the data. Note `X1` and `X2` are very highly correlated (try `cor(X1, X2)`). We have seen in lectures and the example sheets that this high correlation causes the coefficient estimates for `X1` and `X2` to have very high variance, which explains why individually none is seen to be significant when the other is in the null model.
To check whether in simplifying our model for house prices we have not omitted any important variables, we can perform an F-test to test the null hypothesis that the simpler model S_0 is correct, against the alternative of the full model. We do this by supplying both `lm` outputs to the `anova` function, with the smaller model first; we also supply the intercept-only model for pedagogical reasons.

```
> anova(lm(Sale.price ~ 1), HousePricesLM2, HousePricesLM1)
Analysis of Variance Table

Model 1: Sale.price ~ 1
```

```
Model 2: Sale.price ~ Bathrooms + Living.area + Year.built
Model 3: Sale.price ~ Bedrooms + Bathrooms + Living.area + Lot.size +
    Year.built + Property.tax
  Res.Df        RSS Df  Sum of Sq      F Pr(>F)
1     84 3.5481e+11
2     81 1.7963e+11  3 1.7518e+11 26.0111 9.187e-12 ***
3     78 1.7511e+11  3 4.5203e+09  0.6712    0.5723
```

The p-value of the test is 0.5723 so this gives no reason to reject the null hypothesis that our simpler model is correct. Note that we supplied the models in order of nestedness, and so the `anova` function performs F-tests between adjacent models. The first test is nearly the same as that shown in the last line of `summary(HousePricesLM2)` with the difference that the estimate of σ^2 does not come from model `HousePricesLM2` but from `HousePricesLM1` (convince yourself that this is still a valid F-test).
The function `model.matrix` applied to output from `lm` gives the design matrix used in the regression fit (try it out, and note the first column of 1's representing an intercept term). Let X and X_0 be the outputs from `model.matrix` applied to `HousePricesLM1` and `HousePricesLM2` respectively, $X_I := 1_n$, and let P, P_0 and P_I be the orthogonal projections on to the column spaces of X, X_0 and X_I respectively (so e.g. $P = X(X^TX)^{-1}X^T$ or $P_I = n^{-1}1_{n\times n}$, where $1_{n\times n}$ is the n-by-n matrix with all ones). Further let y be the response `Sale.price`, let n be the number of observations and let p and p_0 be the number of columns of X and X_0 respectively. Lastly, let $Z_1 \sim F_{p_0-1,n-p}$ and $Z_2 \sim F_{p-p_0,n-p}$. The following table gives the formulae for the numbers in the output of the `anova` function.

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	$n-1$	$\|(I-P_I)y\|^2$				
2	$n-p_0$	$\|(I-P_0)y\|^2$	p_0-1	$\|(P_0-P_I)y\|^2$	$\frac{\frac{1}{p_0-1}\|(P_0-P_I)y\|^2}{\frac{1}{n-p}\|(I-P)y\|^2}$	$\mathbb{P}(Z_1 \geq F)$
3	$n-p$	$\|(I-P)y\|^2$	$p-p_0$	$\|(P-P_0)y\|^2$	$\frac{\frac{1}{p-p_0}\|(P-P_0)y\|^2}{\frac{1}{n-p}\|(I-P)y\|^2}$	$\mathbb{P}(Z_2 \geq F)$

Variable transformations

Now let us look at predicting the earnings of films based on their opening weekend takings, the number of screens they opened to, their production budgets and their rating from the review aggregator Rotten Tomatoes (`rottentomatoes.com`). Note that this prediction task is rather important for film studios, who would want to get an idea of how well their film will do at the box office soon after it has opened. It is also relevant for cinemas, who would want to know how often they should be showing the films in order to maximise their profits. Although the Rotten Tomatoes rating would not be available at that time, there would be some initial reviews, so our version of the prediction task is not too unrealistic. The dataset we will study looks at the US takings (in millions of dollars) of films released in 2009 that opened on more than 500 screens in the US. We only look at those films for which the production budget is available.

```
> detach(HousePrices)
> Movies <- read.csv(paste(file_path, "Movies.csv", sep =""))
> Movies
> attach(Movies)
```

Plot the response `Total.Gross` against the explanatory variables. You can use `par(mfrow = c(2, 2))` to get them all in one view; `par(mfrow = c(1, 1))` will reset the plotting parameters to just show one plot per view. We see that the plots of the response against the opening weekend takings and production budget show the points are very bunched near the origin, with the points spreading out as we move away to the top right-hand corner.
Let us log transform the response and the variables for the opening weekend takings and production budget, and repeat the same plots. Now a linear model looks more appropriate for the data. We can check that, indeed, $\lambda = 0$ in the Box–Cox family of transformations is a reasonable choice giving an interpretable model: recall that we fit the parameters $(\lambda, \beta, \sigma^2)$ by MLE; we can start by maximising the

log-likelihood with respect to (β, σ^2) so that we obtain a function of λ (Lecturer's note: This is called the profile likelihood function); then, it is common to plot this function to find its or to check that our proposed choice of λ is not far from it (our case). The `R` function `boxcox` from the `MASS` package returns this plot. We load the package first.

```
> library(MASS)
> boxcox(lm(Total.Gross ~ log(Opening) + Screens + RT + log(Budget)))
```

Let us fit a linear model to the transformed data.

```
> MoviesLM <- lm(log(Total.Gross) ~ log(Opening) + Screens + RT + log(Budget))
> summary(MoviesLM)
```

The high R^2 value shows we have a good fit, and the large F-statistic in the bottom row of the summary output shows that the simple intercept-only model is not at all adequate. The number of opening screens, however, does not appear to be significant, and indeed its coefficient estimate is quite close to 0.
We can try to improve the model by omitting the `Screens` variable.

```
> MoviesLM2 <- lm(log(Total.Gross) ~ log(Opening) + RT + log(Budget))
> summary(MoviesLM2)
```

Why is there no point in doing `anova(MoviesLM2, MoviesLM)`? Examining the diagnostic plots for `MoviesLM2` shows there is little heteroscedasticity. By contrast, if we hadn't log-transformed the predictors and the response, the variance of the errors would increase with the mean response. In that case, we shouldn't trust our p-values too much. The exercises below continue the analysis of this data.

Exercises

1. There is one high leverage observation in the movies dataset. Fit a new linear model omitting this observation and also omitting the `Screens` variable. (Recall the functions `hatvalues` and `which`, and the `subset` option of `lm`).

2. Download the data for film earnings in 2010.

   ```
   > Movies2010 <- read.csv(paste(file_path, "Movies2010.csv", sep =""))
   ```

   Compute 95% *prediction* intervals (see `?predict.lm`) for each of the earnings of these films. Remember that you will need to transform prediction intervals you get (though you will not need to transform the data). What proportion of the actual film earnings fall within the prediction intervals you have calculated? Repeat the procedure with 50% prediction intervals.

Forward and backward selection

We can also employ forward and backward selection methods to guide our model choice in the house prices example. Implementations of these algorithms are not part of the standard `R` functions but are contained in the package `MASS`.

```
> library(MASS)
> stepAIC(lm(Sale.price ~ 1, data = HousePrices), scope =
+ Sale.price ~ Bedrooms + Bathrooms + Living.area + Lot.size + Year.built + Property.tax,
+ direction = "forward")  # forward selection
> stepAIC(HousePricesLM1, direction = "backward")  # backward selection
```

Note that the plus signs on the left-hand side simply indicate that the command spans more than one line: they are not part of the command itself. In both of the algorithms, variables are added or deleted until no addition or deletion of a variable decreases the AIC. Reassuringly, both automatic model selection methods deliver our model S_0.

Post model selection inference

As discussed in lectures, confidence intervals formed after model selection has been performed can have less coverage than their nominal value. Let us explore this with the house prices data. We start by detaching the `Movies` dataset and attaching `HousePrices` again.

```
detach(Movies)
attach(HousePrices)
```

First we create a matrix of artificial responses based on the fitted values from `HousePricesLM2` with Gaussian noise whose standard deviation is set to the estimate of σ from the same model. Recall that this model has predictors `Bathrooms`, `Living.area`, and `Year.built`.

```
set.seed(2)
n <- nrow(HousePrices)
n_reps <- 1000
Sale.price_mat <- fitted.values(HousePricesLM2) +
  summary(HousePricesLM2)$sigma * matrix(rnorm(n*n_reps), n, n_reps)
```

Here is a function that takes as input a vector of responses `y` and returns the confidence interval for `Bedrooms`.

```
confint_Bdrm <- function(y) {
  LinMod <- lm(y ~ Bedrooms + Bathrooms + Living.area + Lot.size + Year.built + Property.tax)
  return(confint(LinMod)["Bedrooms", ])
}
```

You can verify that this confidence interval has the correct coverage. For every artificial response simulated from the linear model, compute the confidence interval and evaluate how often the true coefficient for `Bedrooms`, which is equal to 0 as this variable is not in the model, is inside the interval.

```
ConfInts <- apply(Sale.price_mat, 2, confint_Bdrm)
mean((ConfInts[1, ] < 0) * (0 < ConfInts[2, ]))
```

Here, the function `apply` passes every column of `Sale.price_mat` to `confint_Bdrm`, and returns a matrix containing the confidence interval for every realisation of the response. Let's examine what happens when we derive confidence intervals after doing backward selection. The following function takes as input a vector of responses and returns a confidence interval for `Bedrooms` after model selection has been performed using backward selection. Note that for each response `y`, there is a chance that `Bedrooms` will not be in the selected model, and so a confidence interval cannot be computed. In this case we return a vector of `NA` values: these represent missing values in R.

```
confint_bkwd_Bdrm <- function(y) {
  LinMod1 <- lm(y ~ Bedrooms + Bathrooms + Living.area + Lot.size + Year.built + Property.tax)
  LinMod2 <- stepAIC(LinMod1, direction = "backward", trace = 0)
  ConfInt2 <- confint(LinMod2)
  if ("Bedrooms" %in% row.names(ConfInt2)) {
    return(ConfInt2["Bedrooms", ])
  } else {
    return(c(NA, NA))
  }
}
```

Next we compute the coverage probabilities of the confidence intervals returned. The `na.rm` option of `mean` allows us to compute the mean ignoring `NA` values.

```
ConfInts_bkwd <- apply(Sale.price_mat, 2, confint_bkwd_Bdrm)
mean((ConfInts_bkwd[1, ] < 0) * (0 < ConfInts_bkwd[2, ]), na.rm = TRUE)
```

We can query the proportion of times `Bedrooms` is not selected by backward selection using

```
mean(is.na(ConfInts_bkwd[1, ]))
```