Workshop 2 - Selecting Explanatory Variables (full answers)

JPS

Introduction

Let's practice the content of today's lecture with a couple of exercises using the Crown Court Sentencing Survey, a dataset published by the Sentencing Council for England and Wales, capturing the sentence outcome and main case characteristics for offences sentenced in the Crown Court. First, we are going to see how useful stepwise selection can be at building models when the research goal is to *predict*. We will then contrast such unsupervised model selection process with the more careful approach that we should follow when modelling to *explain*. Lastly, we will see how we can use the VIF (Variance Inflation Factor) to detect problems of multicollinearity, and learn how to merge different datasets when you have a unique case identifier. After completing this workshop you will have gained three important new techniques for your expanding toolbox, stepwise regression, VIF, and data merging.

Each of the exercises presented is linked to relevant research questions in the field of sentencing. The first one replicates an article published in the British Journal of Criminology and takes it forward employing stepwise regression. The second exercise suggests an important piece of empirical research that could be easily explored with the techniques that you will acquire today. It deals with an ongoing debate in the sentencing literature, which rather surprisingly, remains unexplored using quantitative methods.

Exercise 1: Imagine we are a consultant for a criminal law firm that wants to use data analytics to inform what type of clients they should take in. We suggest focusing on cases where the most severe penalty (immediate custody) can be avoided. To do so we decide to build a predictive model based on the case characteristics, but also on the characteristics of the court where the sentence is going to be imposed.

Exercise 2: According to the 2009 Criminal Justice Act, judges and magistrates must follow sentencing guidelines in deciding the appropriate sentence. Critics have argued that forcing sentencers to comply with the guidelines has made sentencing in England and Wales more punitive. Specifically Hutton (2013) and Reitz (2013) argued that the guidelines have relegated the relevance of mitigating factors. We will explore this claim by testing whether the effect of mitigating factors on the sentence outcome is weaker than the effect derived from the use of aggravating factors.

Accessing the Data

We are going to use a simplified version of the Crown Court Sentencing Survey (CCSS) including the court where each of the sentences was imposed.[1] We are going to match that with data from the Census capturing the demographic characteristics of the Lower Super Output Area where the court is located, and with a range of other court characteristics obtained from various official reports. This combined dataset was used in: Pina-Sanchez, J. & Grech, D. (2017) Location and Sentencing: To What Extent Do Contextual Factors Explain between Court Disparities? *British Journal of Criminology*.

Let's now access each of those three datasets and proceed to merge them ourselves:

```r
ccss = read.csv("CCSS2011_trimmed.csv", stringsAsFactors = TRUE)
#We use 'stringsAsFactors = TRUE' so the nominal variables are read as factors rather
#than as characters.
```

[1]The full version of the CCSS is available here: https://www.sentencingcouncil.org.uk/research-and-resources/data-collections/crowncourt-sentencing-survey/.

```
#Also, if you have not created a project or have the dataset in the same folder as your
#R script is, make sure you provide the direction to the folder where you saved the
#dataset.
```

This is a clean(er) version of the 2011 CCSS, where I dropped cases where the sentence outcome was missing, variables that are not relevant, others that were measured inconsistently, and created dummy variables for the principal offence so they are ready to be used in a model. I have also tried to provide meaningful variable and value labels, if something is not clear you can always check the metadata file 'CCSS_metadata.xls', available on Minerva.

By checking the main features of this dataset at the top-right menu in Rstudio you can see that it is composed of 49 variables, which is probably bigger than what we are used to. To have a quick look at the content of the dataset we can use…

```
names(ccss)
```

```
##  [1] "Sentencing.court" "Step1HigherCulpA" "Step1HigherCulpC" "Step1HigherCulpD"
##  [5] "Step1HigherCulpF" "Step1HigherCulpG" "Step1HigherCulpI" "Step1HigherCulpJ"
##  [9] "Step1HigherHarmA" "Step1HigherHarmB" "Step1HigherHarmC" "Step1LowerCulpA"
## [13] "Step1LowerCulpB"  "Step1LowerCulpC"  "Step1LowerCulpD"  "Step1LowerCulpE"
## [17] "Step1LowerHarmA"  "Aggravating.3"    "Aggravating.5"    "Aggravating.9"
## [21] "Aggravating.19"   "Aggravating.25"   "Aggravating.26"   "Aggravating.28"
## [25] "Aggravating.37"   "Aggravating.44"   "Aggravating.45"   "Aggravating.54"
## [29] "Aggravating.55"   "Aggravating.70"   "Aggravating.72"   "Mitigating.3"
## [33] "Mitigating.6"     "Mitigating.16"    "Mitigating.20"    "Mitigating.22"
## [37] "Mitigating.24"    "Mitigating.25"    "Mitigating.29"    "Mitigating.33"
## [41] "Mitigating.36"    "Mitigating.39"    "pleafirst"        "disorder"
## [45] "affray"           "GBH"              "intent"           "common"
## [49] "custody"
```

Ok, so we have a variable capturing the court where the sentence was imposed (this has been anonymised so it is just an id rather than the original name), 16 harm and culpability factors, 14 aggravating factors, 11 mitigating factors, 'pleafirst' indicating whether a guilty plea was entered at first opportunity, five dummy variables capturing the specific case of assault being sentenced (with assault with bodily harm, ABH, used as the reference category), and 'custody' indicating whether the offender received an immediate custodial sentence. With the exception of 'Sentencing.court', all the variables included are binary. To assess their relative prevalence we can simply use…

```
summary(ccss)
```

Does it all look ok? I would say so. Our outcome variable seems evenly distributed, no missing cases except 291 cases in 'Sentencing.court' and big enough frequencies so no case characteristics are super rare.

Let's now import the census data and the court characteristics datasets.

```
census = read.csv("Census.csv")
names(census)
summary(census)
```

We can see again a court id (we will use this to carry out the data merge) and 15 variables capturing the proportion of households featuring different sociodemographic characteristics (such as Muslim residents, whites, teenagers, unemployed, etc.) within the same LSOA where the court is located.

```
courtchars = read.csv("CourtChars.csv")
names(courtchars)
summary(courtchars)
```

Here we find the same court id, six binary variables indicating whether certain facilities are present in the court, such as interview rooms, wifi (important to facilitate judges to use the guidelines in their tablets), etc., and a continuous variable 'Volume' capturing the average number of cases processed in each court per month.

Now, notice how most variables are either binary, or proportions (confined to the 0-1 range), but then we have 'Volume', which takes a 217-4226 range. This could create different types problems and make the computational process more demanding, especially when estimating large models (like the ones we are going to estimate, including over 60 explanatory variables) and computationally intensive models (like stepwise regression). We can eliminate those problems by transforming 'Volume' so it takes a narrower range, e.g. we can simply divide it by 1000. There are no downsides to this and it is something you should always consider when using variables with vast ranges. All we have to remember is, when interpreting the regression coefficient for this variable, to bring it back to its original scale.

```
courtchars$Vol_thous = courtchars$Volume / 1000
summary(courtchars$Vol_thous)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2170  0.9875  1.3510  1.7407  2.3260  4.2260
```

```
courtchars$Volume = NULL
```

Good, so we have imported all the three datasets and checked they are in good shape. Let's merge them now. When putting together different datasets it is important to consider what we want to do. Are we trying to add new cases or new variables? In our case it will be adding new variables (i.e. extend the dataset horizontally). Notice however that the three datasets do not have the same number of observations. This is what is known as *one to many* merge, as opposed to a *one to one*. In this case we will need an id variable that can be used to tell R where to match each case. Fortunately for us we can do that using the courts id.[2]

Before running the *merge* command we need to make sure that the variables used to identify unique ids are given the same name in each dataset.

```
names(ccss)[names(ccss) == 'Sentencing.court'] = 'court'
names(census)[names(census) == 'court_name'] = 'court'
names(courtchars)[names(courtchars) == 'Court'] = 'court'
```

We can now proceed with the merging, which we will undertake in two stages since the command *merge* only allow us to merge two datasets at a time. You might notice that after the merging we lose 291 observations from the ccss dataset. This is because the three datasets did not have exactly the same courts (courts have been closing down systematically over the last decade so depending when the data was captured the list of Crown Court locations varies). Given that the lost cases represent less than 10% of the original sample we can simply report this as a caveat and move on with the analysis. In Workshop 7 we will see how to deal with missing data more proficiently.

```
merged1 = merge(ccss, courtchars, by = 'court')
merged2 = merge(merged1, census, by = 'court')
```

Once the merging process is successfully undertaken we can drop the variable 'court' from the final dataset since we are not going to use this in our models, instead we will use the variables describing their characteristics. If we were to include both court characteristics and dummy variables for each court location, we would have perfect collinearity. This means that with one of the explanatory variables you can predict perfectly the values of another explanatory variable, and as a result the model becomes unidentifiable.

```
merged2$court = NULL
```

Exercise 1. Modelling to Predict

Our research goal, as explained in the Introduction, is to predict custodial sentences as accurately as possible. To do so we will try to build the best possible predictive model. But how do we do that? First, we want a

[2]If you face a different *merging* scenario you can assess what procedure to follow here Quick-R.

logistic model since the outcome to be predicted is binary (custody or not). Then we need to consider the next challenge, how to select the set of explanatory variables. We have seen in the lecture that throwing all of the variables we have in a model is a poor strategy since some of them might only be adding noise and we will end up overfitting the model. You might decide to throw them all in anyway, and then in a second stage remove those that are not significant. That would not be ideal either, since some variables are significant when others are not present in the model, so the moment you take one out, others that were not significant might become so. Hence, what you coudl do is to try all different combinations. However, given the large number of variables in our dataset we cannot possibly do that ourselves in a supervised fashion. We have 70 variables, if we were to choose just 20 of them, we would have to compare 115,631,859,759,041,340 models. Instead we will use an unsupervised process, stepwise regression.

To illustrate the benefits of using stepwise selection when building a predictive model we will take the full model as a benchmark. This is the model where all the explanatory variables are included, which can be specified in R easily using '.' on the right hand side of the equation.

```
full.model = glm(custody ~., data=merged2, family=binomial)
summary(full.model)
```

We can see there are several non-significant variables in the model. Most of the court and area specific characteristics are not significant, but neither are many of the harm, culpability, aggravating and mitigating factors, which by definition are supposed to be *substantively* significant. This is likely a consequence of having overfitted our model. Let's see now whether we can improve that model by trimming it down using stepwise regression. This, however, can take a while (about 4 minutes in my laptop).

We are going to use *stepAIC* from the package *MASS*. This command will employ a stepwise selection algorithm to find the set of explanatory variables that minimises the model's AIC (Akaike Information Criteria, a useful measure of goodness of fit).

```
library(MASS)
step.model = stepAIC(full.model)
#The above example follows backward selection only, if you want to combine both
#backward and forward selection you should add, `direction = "both"'
summary(step.model)
```

You can see how R is running multiple iterations with different sets of explanatory variables following backward selection. In each iteration we can see a list of AIC associated with the removal of each variable. You can also see how the models' AIC go down following each iteration, until a model is achieved for which the AIC would not improve after eliminating any more variables.

Ok, let's now assess which model seems to perform better, our initial full model, or this new model obtained through stepwise selection. To compare model performance we will use different metrics, the AIC, the models' size (number of coefficients included, considering both the total number of coefficients included and those found significant), and their predictive accuracy.

```
full.model$aic
```

```
## [1] 3809.711
```

```
step.model$aic
```

```
## [1] 3782.476
```

The AIC is considerably lower in the 'step.model'. Also, expected, the 'step.model' employs far less explanatory variables. We use the command *coef* to list the regression coefficients in each model, and *length* to count them.

```
length(coef(full.model))
```

```
## [1] 70
```

```
length(coef(step.model))
```

```
## [1] 50
```

Even though the stepwise model relies on a smaller set of explanatory variables, it finds more significant variables than if we were to use all the variables available to us. This is calculated by looking at the models' p-values, which are derived by taking the model's results in a matrix form using *coef(summary("name of model")*, and indicating that we want the fourth column, *[,4]*, where the p-values are stored. Then we use *ifelse* transform p-values from a numeric variable into a binary, capturing whether the variable is significant or not. Lastly, we use *sum* to count the number of significant variables in each model.

```
pvalues_full = coef(summary(full.model))[,4]
significant_full = ifelse(pvalues_full>0.05, 0, 1)
sum(significant_full)
```

```
## [1] 34
```

```
pvalues_step = coef(summary(step.model))[,4]
significant_step = ifelse(pvalues_step>0.05, 0, 1)
sum(significant_step)
```

```
## [1] 37
```

Lastly, and most importantly, the trimmed model still manages to provide higher predictive accuracy. We can check that using the *predict* function, with the option *type = response*, which gives us the predicted probability of custody for each case estimated from the model (using the explanatory variables included). We can then use *ifelse* again to turn that predicted probability into a binary, custody or not. Lastly, we can compare our predicted sentences of custody with the actual sentences imposed recorded in the original data. All of the above (lower AIC, less variables included, more variables found significant, and higher predictive accuracy) combined, illustrates quite well some of the advantages of using stepwise procedures for exploratory and predictive purposes.

```
probs_full = predict(full.model, type = "response") #The predicted sentence
#according to the full model.
predicted_cust_full = ifelse(probs_full > 0.5, "immediate custody", "")
observed_cust_full = merged2$custody  #This is the actual sentence
mean(predicted_cust_full == observed_cust_full)
```

```
## [1] 0.7983577
```

```
#This is to count the number of matches between the predicted and the observed
#(true) sentence.
#Then we can repeat the same procedure for the step model and compare.
probs_step = predict(step.model, type = "response")
predicted_step_full = ifelse(probs_step > 0.5, "immediate custody", "")
observed_step_full = merged2$custody
mean(predicted_step_full == observed_step_full)
```

```
## [1] 0.7994982
```

Sidenote: to be able to assess the predictive accuracy of our models we should not really be using the same dataset, but rather we should use one dataset to estimate the model, and a different one to test its predictive accuracy. The full instructions to be uploaded after the workshop show how to do so.

We could do so here by splitting our datatset into a training and test sample. We can undertake such a dataset split using the command *createDataPartition* (from the machine learning package *caret*), and the *%>%* operator.

```
set.seed(10)  #we are going to partition our sample at random, to make sure that
#you always conduct the same random sample you can use set.seed
library(caret) #this is to use createDataPartition
library(dplyr) #this is to be able to use %>%
training.sample = merged2$custody %>% createDataPartition(p = 0.8, list = FALSE)
train.data = merged2[training.sample, ]
test.data = merged2[-training.sample, ]
```

First we use *set.seed*. This is because we are going to ask R to draw pseudo-random numbers. By providing a figure to *set.seed*, we will ensure that we get the same random numbers every time we replicate our code. Next, we have used *createDataPartition* to create a vector representing the 80% of cases that have been picked from our original dataset ('merged2') to conform the training.sample. The choice of 80% is rather arbitrary, researchers normally use 50%, however since the size of our sample is not massive, and because we have a good number of explanatory variables, I did not want to risk ending with a sample too small to undertake the model selection effectively. The *%>%* operator facilitates using loops in R. Specifically, our command can be translated as: for every case in a given variable in the dataset ('custody' in our case), give it an 80% chance of being picked up, and report those that were selected. At this point we can use the vector we have created ('training.sample') to create our training and test samples. So we can run the stepwise selection model on the 'training.data', and use the *predict* command using the 'test.data'.

Exercise 2. Modelling to Explain

Let's now compare the above approach with the model selection process that we should follow when exploring a clearly identified confirmatory research question. In the introduction we suggested testing the hypothesis that aggravating factors are more relevant than mitigating factors in deciding the sentence outcome (Hutton 2013; and Reitz 2013).

To opt for a full model (i.e. throwing all available explanatory variables in the model) would not be an adequate strategy. It would result in an overfitted model, including variables that we do not want to test or that are not useful controls. A stepwise procedure would not offer an adequate strategy either. The hypothesis to be tested is very clearly defined, it requires a comparison of the effect of aggravating and mitigating factors, but our previous 'step.model' only kept 9 out of the 14 aggravating factors available in the dataset, and 10 out of the 11 mitigating factors.

Since we want to test a specific hypothesis the choice of the variables to be included in the model should be theoretically driven. In this case, we should retain all aggravating and mitigating factors, even if they are not significant. In fact, whether or not these variables are significant is a relevant consideration in deciding whether to refute or corroborate the hypothesis that mitigating factors have been relegated in the sentencing guidelines.

Further, we should select variables that, based on the - sentencing - theory, can be considered to have an effect on the - sentence - outcome. This is a subjective choice and should always involve careful judgement. For the particular case we are considering, this choice is perhaps easier than for most other questions explored by social scientists, since the sentencing guidelines provide a list of factors deemed to be relevant in deciding the sentence outcome. So we can simply include all of them.[3]

1. Try to do this yourself. Take the 'ccss' data (where all the sentencing guidelines factors are recorded) to estimate a logistic model regressing custodial sentences on all the sentencing guideline factors. Hint1: you can use *glm* with the option *family=binomial* to estimate logistic models. Hint2: To include all remaining variables other than 'custody' as explanatory variables you can use '~.'. Hint3: you might need to decide what to do with 'Sentencing.court', either remove it from the dataset (*ccss$court=NULL*) or include it (*ccss$court=as.character(ccss$Sentencing.court)*). Question: Can you see why including 'Sentencing.court' as it is recorded in 'ccss' is problematic?

[3]This statement, however, is a pretty broad oversimplification. If you want to know more about additional considerations that we ought to take into account when modelling sentencing data, I have just written about that here (Pina-Sánchez, 2024).

```r
#ccss$court = as.character(ccss$court)
ccss$court = NULL
theo.model = glm(custody ~., data=ccss, family=binomial)
summary(theo.model)
```

'Sentencing.court' is recorded as a factor, but in reality it is a character (categorical) variable. Hence, its class should be modified in order to be able to interpret it correctly, since there is no underlying ranking of courts (court 1 is not higher, bigger, etc. than court 2 or court 76). Alternatively, you could simply drop 'Sentencing.court' from the model since most courts are not significant and there are 76 of them, which could overfit the model.

2. One good way to see if the model is overfitted is to test the presence of multicollinearity. Use the command *VIF* as shown below, substituting 'theo.model' for the name you gave your last model. Question: Do you find multicollinearity in any of your explanatory variables? Hint1: You can use the cut-off point suggested in today's lecture (slide 10).

```r
library(regclass) #this is to use VIF
VIF(theo.model)
```

We do not find problems of multicollinearity (defined as *VIF>5*).

3. Ok, so we can proceed to interpret the model's coefficients and test our hypothesis. Remember, we want to compare the importance of aggravating and mitigating factors to test whether the latter are less important than the former. Question: How would you do that? Hint1: Since they are all measured in the same scale (binary) you can simply add up the coefficients for all the aggravating factors found significant (if they are not significant we can think of them as not having an influence on the outcome, or at least not having an influence that we can detect) and compare that against the sum of the coefficients of all significant mitigating factors. Question: What is your conclusion?

```r
#Overall effect associated to the 10 aggravating factors found significant
sum(coef(theo.model)[19], coef(theo.model)[20], coef(theo.model)[23], coef(theo.model)[25],
    coef(theo.model)[26], coef(theo.model)[27], coef(theo.model)[28], coef(theo.model)[29],
    coef(theo.model)[30], coef(theo.model)[31])
```

```
## [1] 7.477075
```

```r
#Overall effect associated to the 10 mitigating factors found significant
sum(coef(theo.model)[32], coef(theo.model)[33], coef(theo.model)[34], coef(theo.model)[35],
    coef(theo.model)[36], coef(theo.model)[37], coef(theo.model)[38], coef(theo.model)[39],
    coef(theo.model)[40], coef(theo.model)[41])
```

```
## [1] -9.428598
```

Out of the 14 aggravating factors listed in the sentencing guidelines only 10 are significant. When all of these 10 significant aggravating factors are present in a case they can push the the log-odds of custody up by 7.48. Out of the 11 mitigating factors listed in the guidelines 10 are significant, when all of these are present in a case they can push the log-odds of custody down by -9.43. In conclusion, it seems that Hutton (2013) and Reitz (2013) are wrong, even though there are less mitigating factors listed in the guidelines, they are more consequential than aggravating factors, at least for the case of offences of assault processed in the Crown Court.

Just so you can appreciate how slow quantitative sentencing research moves and the multiple research opportunities for properly trained researchers. It took sentencing scholars a decade to actually test this question empirically (Chen et al., 2013), even though the data with which to do so has always been publicly available.

Preparation for next week's workshop

Next week we will practice path analysis, a type of analysis that can be used direct from indirect effects when we seek causal reasoning (i.e. when we model to explain). The first exercise seeks to test Tyler's (1990) procedural justice model. This is a guided exercise so you are requested to replicate the procedures list there slowly to try to follow the reasoning behind them. The second exercise is unguided. You are requested to use the Labour Force Survey to explore the gender gap. Specifically you are asked to estimate the direct effect of gender on salaries, and a potential indirect effect through the number of hours worked. As with this week's unguided exercise, you are strongly encouraged to try to resolve this exercise following the hints left in the instructions.