Workshop 7 - Data Quality (full answers)

JPS

Introduction

In today's workshop we are going to practice applying and designing various adjustments to improve the validity of our estimations in the presence of missing data and other sampling issues leading to selection bias. Specifically, we will use probability weights and imputation methods, to adjust the findings from datasets derived from two surveys: the Crown Court Sentencing Survey and the Leeds Parks Survey.

Exercise 1: The Crown Court Sentencing Survey is based on questionnaires self-completed by judges after imposing a sentence. When analysing this data we see lots of questions capturing the characteristics of the case that were left incomplete (item-missingness), which could be biasing our findings. Here we will practice different imputation methods (mean, regression and multiple imputation) to adjust for selection bias and make sure that we can use the information available in those cases affected by item-missingness. As we did in Week 2 we will try to estimate the relative importance of different sentencing guidelines' factors in determining custodial sentences, but this time we will be able to use the degree of offence seriousness. This is the most important factor according to the guidelines, but one we could not use before because it is missing in a considerable number of cases.

Exercise 2: In 2018 Dr Anna Barker and myself designed a survey to estimate - amongst other things - how often do people in Leeds use their public parks. This was a non-probability based survey, so the sample is likely non-representative (affected by selection bias). Just like for probability samples affected by non-response we can adjust this problem using weights. Here we will use information from the 2011 Census, specifically the age and gender distribution of Leeds residents, to produce poststratification weights for the Leeds Parks Survey. Once the weights are generated you are required to provide an adjusted estimate of the proportion of Leeds residents who seldom or never visit a park.

Exercise 1. Imputation

Imputation methods can be used to adjust for different types of problems of missing data but they are especially appealing to adjust for item-missingness since they allow us to *fill-in* some of the missing values and in so doing avoid *listwise deletion* (dropping entire cases just because the value in one or a few variables is missing). That is, they allow us to make the most of the data available.

Let's practice different imputation methods using the Crown Court Sentencing Survey (CCSS). Some key variables in the CCSS, such as the level of offence seriousness, or the number of previous convictions, are heavily affected by item-missingness, which have pushed researchers to a methodological dilemma: either to exclude these key variables from their analyses (probably leading to confounding bias) or to include them and lose all those cases affected by item-missingness (i.e. listwise deletion, probably leading to selection bias and definitely leading to a loss of statistical power). We are going to see how imputation methods can be used to overcome this trade-off, i.e. include all relevant variables without having to lose any cases. We will do this employing three imputation methods, in order of sophistication: mean imputation, regression imputation, and multiple imputation (this last one will be covered in the 'full answers' version of the tutorial.

We can start by accessing a couple of years worth of cases of assault sentenced in the Crown Court. To do so we can go to the Sentencing Council website (https://www.sentencingcouncil.org.uk/research-and-resources/data-collections/crowncourt-sentencing-survey/) and download the data directly from there. Click on ‘Datasets’ at the bottom of the page and select the assault files for 2013 and 2014. Save them in a folder of your choice and use the direction to that folder to open the datasets from R.

```r
assa13 = read.csv("ASSAULT_2013_NEW.csv", stringsAsFactors = TRUE)
assa14 = read.csv("ASSAULT_2014_NEW.csv", stringsAsFactors = TRUE)
```

Since we are going to analyse these two datasets together we first need to combine them into a single dataset. In Week 2 we used the *merge* command to combine two datasets horizontally (we added variables), here we are going to combine two datasets vertically (to add more cases on the same variables), this is often referred to as appending - rather than merging - data. This procedure only works if we have the exact same variables in each of the datasets to be combined. We can use *table* and *names* to check that we have each variable twice, once in each dataset.

```r
#In this case all variables are present in the two datasets, so we can proceed with
#the appending.
table(c(names(assa13), names(assa14)))
#This is to combine the cases from the two datasets.
assa = rbind(assa13, assa14)  #rbind stands for row (cases) bind.
#The following is to make sure that the original names are maintained in the dataset
#that results from the merging.
names(assa) = names(assa13)
```

We are going to trim down the variables in the dataset to simplify the analysis. We will do that by keeping only the offence type, sentence outcome, number of previous convictions, level of seriousness and Step One factors (i.e. the list of harm and culpability factors that judges should use to determine the level of seriousness of the offence).

```r
names(assa)
vars = c("OFFENCE","OUTCOME","SERIOUSNESS","APO_GREATER_HARM_1","APO_GREATER_HARM_2",
         "APO_GREATER_HARM_3","APO_LESSER_HARM_1","APO_HIGHER_CULP_STAT_1",
         "APO_LOWER_CULP_1","APO_LOWER_CULP_2","APO_LOWER_CULP_3","APO_LOWER_CULP_4",
         "APO_LOWER_CULP_5","PREV_CONVICTIONS")
#APO_HIGHER_CULP_STAT_2, APO_HIGHER_CULP_STAT_3, and APO_HIGHER_CULP_STAT_4 could
#also be included but I dropped them because they are only present in less than 20 cases
#in the sample that we are using.
assa = assa[vars]
```

To simplify the analysis, but also since different harm and culpability factors are applied differently for different offence types, we will focus on cases of ABH S.47 (assault with bodily harm), the most common assault offence sentenced in the Crown Court.

Question: Can you drop all offence types other than ‘S.47’? There are different ways you can subset datasets according to specific criteria; hint: in previous workshops we have used ‘dataset[which(variable==’category’),]’.

```r
table(assa$OFFENCE, useNA="ifany")  #This is to check the exact value label for ABH.
assa = assa[which(assa$OFFENCE=="S.47"),]  #Keeping only cases of OFFENCE equal to 'S.47'.
assa$OFFENCE = NULL   #Keeping the dataset tidy.
```

To prepare the analysis exploring the influence that different factors have on the sentence outcome we can also simplify that variable by turning it into a binary.

Question: Can you recode OUTCOME into a different variable capturing whether sentenced to custody or not? Hint: there are multiple ways you can follow to recode variables, in previous workshops we have used *ifelse*.

```r
table(assa$OUTCOME, useNA="ifany")  #Checking the exact value label for custody.
assa$CUSTODY = ifelse(assa$OUTCOME=="Immediate custody", 1, 0)  #The recoding.
assa$OUTCOME = NULL  #Keeping the dataset tidy.
```

In addition, a quick exploratory analysis shows us that the formatting used across variables is not entirely consistent. We need to fix that. We also need to make sure that R will identify the missing cases in 'PREV_CONVICTIONS' and 'SERIOUSNESS'.

```
summary(assa) #This is to detect the different value labels used in the two datasets
#that we have combined.
#Changing labels for APO_LESSER_HARM so all Step-One factors follow the same formatting.
#I use as.character() to ensure that the new variable keeps the label of the categories
#used in the original format.
assa$APO_LESSER_HARM_1 = ifelse(assa$APO_LESSER_HARM_1=="None stated", "-",
                                as.character(assa$APO_LESSER_HARM_1))
#Setting missing cases in PREV_CONVICTIONS as NA.
assa$PREV_CONVICTIONS = ifelse(assa$PREV_CONVICTIONS=="Not answered",NA,
                               as.character(assa$PREV_CONVICTIONS))
#factor() is used so R can understand that PREV_CONVICTIONS is an ordinal variable.
assa$PREV_CONVICTIONS = factor(assa$PREV_CONVICTIONS,
                               levels=c("None", "1 to 3", "4 to 9", "10 or more"))
#Same with SERIOUSNESS.
table(assa$SERIOUSNESS, useNA="ifany")  #Listing all the values used in SERIOUSNESS.
assa$SERIOUSNESS = ifelse(assa$SERIOUSNESS=="Not answered"|
          assa$SERIOUSNESS=="No existing guideline"|assa$SERIOUSNESS=="Not Answered"|
          assa$SERIOUSNESS=="Not asked", NA, assa$SERIOUSNESS)
#The above code can be read as, if 'SERIOUSNESS' equals 'Not answered',
#'No existing guideline', 'Not Answered', or 'Not asked', then coded it as NA,
#otherwise keep its original value
assa$SERIOUSNESS = factor(assa$SERIOUSNESS) #No need to identify any levels like for
#'PREV_CONVICTIONS' since these can be inferred from the numerical labels used,
#'which go from 1 to 3.
```

If we want to explore the effect of previous convictions, offence seriousness, or any of the Step-One factors on the probability of receiving an immediate custody we can just specify a logistic model. Notice however that if we do not undertake any adjustments this model will proceed to eliminate any cases where either 'SERIOUSNESS' or 'PREV_CONVICTIONS' is missing (listwise deletion).

Question: Can you estimate a logit model regressing 'CUSTODY' on all the case characteristics that we have kept in our dataset and find out how many cases we lose as a result of listwise deletion (call this model 'logit1')? Hint1: To estimate a logit model you need the *glm* command with the option *family='binomial'*. Hint2: The number of cases dropped is indicated in the *summary* of the model, third line from the bottom.

```
logit1 = glm(CUSTODY~SERIOUSNESS + APO_GREATER_HARM_1 + APO_GREATER_HARM_2 +
         APO_GREATER_HARM_3 + APO_LESSER_HARM_1 + APO_HIGHER_CULP_STAT_1 +
         APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 + APO_LOWER_CULP_4 +
         APO_LOWER_CULP_5 + PREV_CONVICTIONS, family="binomial", data=assa)
summary(logit1)
```

As we can see from the model output, there are 1037 cases dropped from our sample. Let's impute those missing values so we do not have to drop entire cases from our model. The simplest method is mean imputation, a form of single imputation where we replace the missing values in a given variable by the mean of the values observed in that same variable.

Notice how this method assumes missing completely at random. If the missing data mechanism (i.e. the reason why values are missing) is related to any of the variables used in the model, it will not adjust for selection bias, but at least it will allow us to use all the cases in the sample, offsetting the loss of statistical power. This, together with the simplicity of the method, can explain its popularity. I have used it multiple times, however, we should be critical when relying on this method, since not only does not adjust for selection bias, it also distorts the distribution of the variables imputed by reducing their variability, artificially placing a spike in the distribution where the mean is located. We can visualise these problems by undertaking the mean imputation of 'SERIOUSNESS'. 'SERIOUSNESS' is an ordinal variable so technically we will be using median imputation.

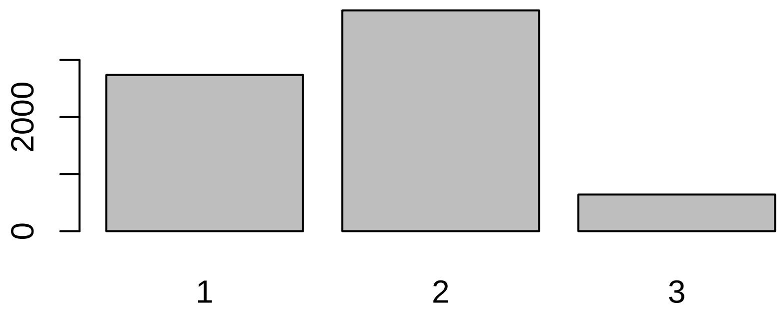
```
table(assa$SERIOUSNESS, useNA="ifany")
```

```
##
##    1    2    3 <NA>
## 2738 3869  643  911
```

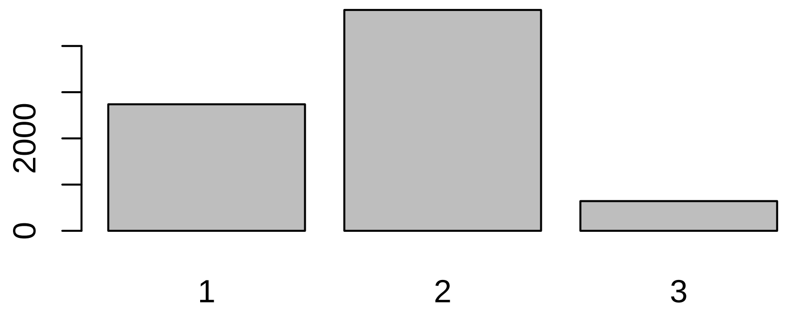
```
#Below we are saying, if SERIOUSNESS is missing replace it by a 2 (its median),
#otherwise keep its original value.
assa$SERIOUS_MEANIMP = factor(ifelse(is.na(assa$SERIOUSNESS)==TRUE, 2, assa$SERIOUSNESS))
table(assa$SERIOUS_MEANIMP)
```

```
##
##    1    2    3
## 2738 4780  643
```

```
#The above is to check that the transformation went ok. You can see that the 911 NA
#cases were added to the category 2, as expected.
#The following is to compare the distribution of the unadjusted and imputed variables.
plot(assa$SERIOUSNESS)
```



```
plot(assa$SERIOUS_MEANIMP)
```



```r
sd(as.numeric(assa$SERIOUSNESS), na.rm=TRUE)
```

```
## [1] 0.61879
```

```r
sd(as.numeric(assa$SERIOUS_MEANIMP))
```

```
## [1] 0.59028
```

```r
#You can see how the distribution of the imputed variable has been artificially modified,
#which is not great.
assa$SERIOUS_MEANIMP = NULL  #We get rid of this variable as we won't use it anymore.
```

A better imputation method is regression imputation. We can use all the cases with complete information to regress the variable affected by item-missingness on the other variables in our sample. Once we have estimated that model we can use it to predict the missing values in a way that it incorporates what we know about the missing mechanism from the other variables available in the dataset. Below we can see how to do this for the case of ‘SERIOUSNESS’. Since this is an ordinal variable I suggest using an ordered logit model. This is a new regression model, not too different from the logit/logistic models we have used so far.

The logit models we have used until now are designed for binary data, whereas the ordered logit model allows for > 2 ranked categories. In binary logit models the intercept indicates the threshold after which cases are predicted as 1s. In ordered logit models we have $c - 1$ thresholds, with c being the number of ranked categories. For example, for the case of ‘SERIOUSNESS’ we have $c = 3$ and therefore 2 intercepts (thresholds), the first one indicating whether a case makes a transition from seriousness 1 to seriousness 2, and the second one indicating transitions from seriousness 2 to seriousness 3. The rest of the regression coefficients are to be interpreted as for a binary logistic model. That is, they are expressed in the same unit of measurement, log-odds, and can be transformed into odds ratios using *exp*. To specify the ordered logit model we will use the *polr* command from the *MASS* package.

```r
#Regression imputation
library(MASS) #To be able to use polr.
ologit = polr(SERIOUSNESS ~ APO_GREATER_HARM_1 + APO_GREATER_HARM_2 + APO_GREATER_HARM_3 +
         APO_LESSER_HARM_1 + APO_HIGHER_CULP_STAT_1 + APO_LOWER_CULP_1 + APO_LOWER_CULP_2 +
         APO_LOWER_CULP_3 + APO_LOWER_CULP_4 + APO_LOWER_CULP_5 + PREV_CONVICTIONS +
         CUSTODY, data=assa, Hess=TRUE)
summary(ologit)
#We can drop previous convictions and APO_HIGHER_CULP_STAT_1 since they are not significant.
ologit = polr(SERIOUSNESS ~ APO_GREATER_HARM_1 + APO_GREATER_HARM_2 + APO_GREATER_HARM_3 +
         APO_LESSER_HARM_1 + APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 +
         APO_LOWER_CULP_4 + APO_LOWER_CULP_5 + CUSTODY, data=assa, Hess=TRUE)
summary(ologit)
```

Once the model is estimated we can undertake the predictions of the missing values using *predict*. We will embed this command within an *ifelse* command to tell R that we want a new variable for seriousness taking the values of 'SERIOUSNESS' that are not missing, and predictions from our model to replace the missing values.

```
#We also use the command factor so the imputed variable is still taken as an ordinal.
assa$SERIOUS_REGIMP = factor(ifelse(is.na(assa$SERIOUSNESS)==TRUE, predict(ologit),
                                    assa$SERIOUSNESS))
```

We can look at the distribution of the variable obtained using regression imputation, and check how regression imputation reflects more accurately the distribution of the observed values of the original variable 'SERIOUSNESS' than mean imputation.

```
table(assa$SERIOUSNESS, useNA="ifany")
```

```
##
##    1    2    3 <NA>
## 2738 3869  643  911
```

```
table(assa$SERIOUS_REGIMP)
```

```
##
##    1    2    3
## 3051 4460  650
```

```
#The distribution of the imputed variable is now more similar to the original one.
#The strong peak at 2 is not there anymore.
sd(as.numeric(assa$SERIOUSNESS), na.rm=TRUE)
```

```
## [1] 0.61879
```

```
sd(as.numeric(assa$SERIOUS_REGIMP))
```

```
## [1] 0.60579
```

In addition to reflecting more accurately the distribution of the original variable, the regression-imputed values will be able to adjust (even if partially) for selection bias if the missing data mechanism behind the item missingness in 'SERIOUSNESS' is associated with any of the variables used in our last ordered logit model. This could be the case for example if responses to the question on 'SERIOUSNESS' in the CCSS were recorded less commonly in simpler cases, e.g. those that are not defined by a large number of Step One factors. If that was the case we could learn about the data generating mechanism using Step One factors and the observed cases. We can test this hypothesis using a binary logit model with outcome whether the value of 'SERIOUSNESS' is missing (or not), and explanatory variables the different Step One factors used in our ordered logit model.

```
#This is to create a variable capturing whether 'SERIOUSNESS' is missing
assa$missing = ifelse(is.na(assa$SERIOUSNESS)==TRUE, 1, 0)
logit_miss = glm(missing~APO_GREATER_HARM_1 + APO_GREATER_HARM_2 + APO_GREATER_HARM_3 +
             APO_LESSER_HARM_1 + APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 +
             APO_LOWER_CULP_4 + APO_LOWER_CULP_5, family="binomial", data=assa)
summary(logit_miss)
assa$missing = NULL
```

Our hypothesis is corroborated. The probability that values of 'SERIOUSNESS' are missing is negatively associated with the presence of Step One factors, whatever their sign (increasing or reducing harm and culpability). Thus, it seems that judges who presided over complex cases, where multiple Step One factors were present, did not overlook the 'SERIOUSNESS' category from the CCSS questionnaire. Furthermore, since most of the factors we use for the prediction of missing values are strongly associated with the missing data mechanism (the probability of missingness), our adjustment based on regression imputation should help to correct the selection bias present in the CCSS, which was left unadjusted in our previous model. We can assess whether that is the case by comparing the results from our previous model with those from a new model where seriousness has been regression imputed. To do so we will use a new function **tab_model**, which allows us to present results from multiple regression models more neatly.

```
#The new model including seriousness adjusted through regression imputation.
logit2 = glm(CUSTODY~SERIOUS_REGIMP + APO_GREATER_HARM_1 + APO_GREATER_HARM_2 +
             APO_GREATER_HARM_3 + APO_LESSER_HARM_1 + APO_HIGHER_CULP_STAT_1 +
             APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 + APO_LOWER_CULP_4 +
             APO_LOWER_CULP_5 + PREV_CONVICTIONS, family="binomial", data=assa)
library(sjPlot)  #This is to be able to use tab_model.
```

```
## Warning: package 'sjPlot' was built under R version 4.4.2
```

```
tab_model(logit1, logit2)  #To present the two models in a way that is easy to compare.
```

Question: Can you see any relevant differences between the unadjusted and the adjusted model?

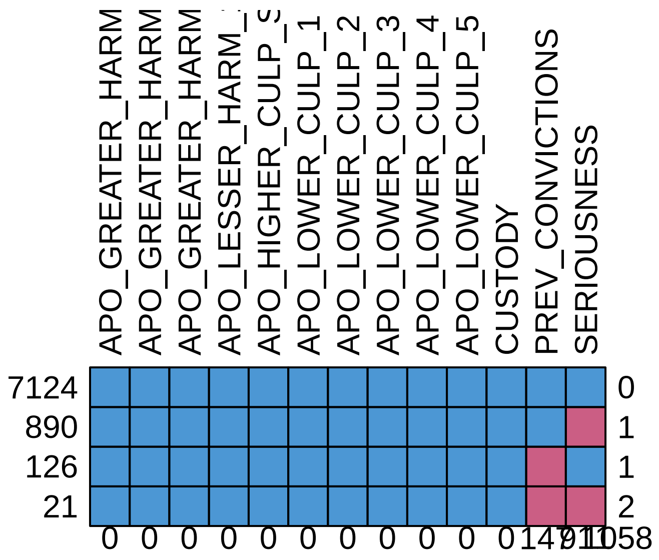
We can see certain differences between the unadjusted and the adjusted model. The effect of 'SERIOUSNESS' on the probability of custody is weaker in the unadjusted model. We can also see how 'SERIOUSNESS' is not the only biased coefficient. Furthermore, we can see how the number of cases dropped through listwise deletion has gone down from 1037 to 147, which increases the model's degrees of freedom (or statistical power), which means higher precision in our estimates, as shown by the smaller p-values and narrower confidence intervals. The remaining 147 cases still dropped represent those with missing values still present in 'PREV_CONVICTIONS'. To deal with them we could replicate the same approach and impute those missing values in 'PREV_CONVICTIONS' using regression imputation, but we can do even better.

We have seen how regression imputation is superior to mean imputation, however it is still a *single imputation* method. Single imputed values are predictions (from a mean or a more complex multivariate model). These predictions are not perfect, they are estimated with uncertainty (i.e. standard errors). If we use them in subsequent analyses as if they were real data as opposed to estimates, we will be providing a misleading account of their precision. To account for the uncertainty associated with the imputation process we can use multiple imputation.

Under imputation we estimate a range of possible values to be imputed, rather than *the* predicted value. This way we can create multiple datasets, each one including slightly different imputed values reflecting the uncertainty of the imputation process. We then proceed to undertake our analysis of interest (in our case to model the probability of receiving a custodial sentence) for each of the datasets that we have created. For each one of them we will obtain slightly different results, reflecting the uncertainty of the imputation process. As a final step we can pool the estimates obtained in each dataset into a single average estimate that will incorporate the variability (uncertainty) observed across the analyses carried out using different datasets.

In the code below we will undertake multiple imputations for both 'SERIOUSNESS' and 'PREV_CONVICTIONS' simultaneously using the *mice* package and the *mice* command. We specify that we want 5 different datasets to be imputed using predictive mean matching (*pmm*). In essence predictive mean matching is the combination of a regression model with an added matching procedure to make sure that values to be estimated are expressed in the same units of measurement of the variable where they are going to be imputed.

```
assa$SERIOUS_REGIMP = NULL  #We drop our previous single imputation for seriousness.
library(mice)
md.pattern(assa, rotate.names=TRUE) #To visualise the missing data patterns.
```



md.pattern is a useful command to identify which variables are affected by item non-response and by how much. We can see three main missing data patterns, ordered by relevance: 890 cases where only 'SERIOUSNESS' is missing, 126 cases where 'PREV_CONVICTIONS' is missing, and 21 cases where values for both of those variables are missing.

To undertake the imputation we use *mice*, which creates an object specifying the multiple imputation procedure that we want to run.

```
#The multiple imputation process.
miassa = mice(assa,m=5,meth='pmm',seed=7)
#I then proceed to check the imputed values for 'SERIOUSNESS' and 'PREV_CONVICTIONS'.
#Each row represents a missing value imputed, each column represents 1 of the 5
#imputations requested.
miassa$imp$SERIOUSNESS
miassa$imp$PREV_CONVICTIONS
#The following is to check the first 20 cases for the complete first dataset imputed.
complete(miassa,1)[1:20,]
```

Notice how the imputed values are not always the same, which reflects the uncertainty associated with the imputation process. The better the imputation model is at predicting missing values the more closely aligned the imputed values will be across the multiple datasets created. This can be observed by comparing the similarity across imputed values of 'SERIOUSNESS' and across values of 'PREV_CONVICTIONS', the former are much more accurately predicted by the auxiliary information that we have, which is mainly formed of Step One factors and therefore should contain all the information we need to determine the offence seriousness (according to the sentencing guidelines in England and Wales).

Ok, at this point we can proceed to undertake our substantive analysis (exploring which factors are associated with the probability of receiving a custodial sentence) using our new adjusted data based on multiple imputation. To do so we need to specify the model using the *with* command, and summarise our results using *pool*, as shown below.

```
logit3 = with(miassa, glm(CUSTODY~SERIOUSNESS + APO_GREATER_HARM_1 + APO_GREATER_HARM_2 +
              APO_GREATER_HARM_3 + APO_LESSER_HARM_1 + APO_HIGHER_CULP_STAT_1 +
              APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 + APO_LOWER_CULP_4 +
              APO_LOWER_CULP_5 + PREV_CONVICTIONS, family="binomial"))
summary(pool(logit3))
```

We can determine that this is the best of the three adjustments we have undertaken since it can - even if only partially - adjust for the problem of selection bias stemming from item missingness in both 'SERIOUSNESS' and 'PREV_CONVICTIONS', without biasing the measures of uncertainty in our final model, i.e. propagating the uncertainty of the imputation process accurately. If you want to see how this is the case you could compare the standard errors from model 'logit3' against those from 'logit2', ideally after imputing the missing values for 'PREV_CONVICTIONS' using regression imputation too, so 'logit2' can be estimated without losing any cases.

This procedure will be shown in the version of the workshop including *full answers*.

```
#Regression imputation for missing cases in 'SERIOUSNESS'
ologit = polr(SERIOUSNESS ~ APO_GREATER_HARM_1 + APO_GREATER_HARM_2 + APO_GREATER_HARM_3 +
         APO_LESSER_HARM_1 + APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 +
         APO_LOWER_CULP_4 + APO_LOWER_CULP_5 + CUSTODY, data=assa, Hess=TRUE)
assa$SERIOUS_REGIMP = factor(ifelse(is.na(assa$SERIOUSNESS)==TRUE,
                                    predict(ologit), assa$SERIOUSNESS))
#Regression imputation for missing cases in 'PREV_CONVICTIONS'
ologit = polr(PREV_CONVICTIONS ~ APO_GREATER_HARM_1 + APO_GREATER_HARM_2 +
         APO_GREATER_HARM_3 + APO_LESSER_HARM_1 + APO_LOWER_CULP_1 + APO_LOWER_CULP_2 +
         APO_LOWER_CULP_3 + APO_LOWER_CULP_4 + APO_LOWER_CULP_5 + CUSTODY,
         data=assa, Hess=TRUE)
assa$PREVCONV_REGIMP = factor(ifelse(is.na(assa$PREV_CONVICTIONS)==TRUE, predict(ologit),
                                     assa$PREV_CONVICTIONS))
#The model including regression imputations for 'SERIOUSNESS' and 'PREV_CONVICTIONS'
logit2 = glm(CUSTODY~SERIOUS_REGIMP + APO_GREATER_HARM_1 + APO_GREATER_HARM_2 +
         APO_GREATER_HARM_3 + APO_LESSER_HARM_1 + APO_HIGHER_CULP_STAT_1 + APO_LOWER_CULP_1 +
         APO_LOWER_CULP_2 + APO_LOWER_CULP_3 + APO_LOWER_CULP_4 + APO_LOWER_CULP_5 +
         PREVCONV_REGIMP, family="binomial", data=assa)
```

`{rresults="hide",} #Standard errors from the model imputing 'SERIOUSNESS' using regression imputation. summary(logit2)$coefficients[,2]   #Standard errors from the model imputing 'SERIOUSNESS' using multiple imputation. summary(pool(logit3))[,3]` The standard errors in 'logit2' are underestimated as a result of taking imputed values as real data points, rather than estimated data points. This is potentially quite problematic, as it could lead to false positives (type I errors).

Exercise 2. Poststratification

You are required to calculate weights using poststratification to adjust for selection bias in the Charitable giving to parks and green spaces survey. You should then apply these weights to estimate the proportion of Leeds residents who 'seldom or never' visit a park.

To calculate the weights you can rely on the gender and age distribution of Leeds residents available at the Census. The age distribution is 0.491 'male' and 0.509 'female'. The age distribution is 0.494 '20 to 44', 0.302 '45 to 64', and 0.203 '65+'. You can compare the population distribution for these two variables to the gender and age distribution captured in the survey (available on Minerva), as we did in slide 8 of the lecture.

To open the dataset we need to use the *read.spss* command (from the library *foreign*), including the option *to.data.frame=TRUE.*

```
library(foreign)
parks = read.spss("LeedsParksSurvey.sav", to.data.frame=TRUE)
#You can ignore the warning message, it refers to specific symbols used in SPSS to code
#some of the values in the dataset, and it has no effect in how the data is read in R.
```

The dataset is quite simple, following a little exploratory analysis we can see that we have four variables, Q4 capturing frequency of park visits, Q2 capturing duration of those visits, and Q31 and Q32 representing gender and age respectively.

```r
summary(parks)
```

We are going to use age and gender to caculate the weights, but we can see that both variables have missing cases that are not identified as NAs (so R does not recognise them as missing), and age does not follow the exact categories used in the Census.

Question: Can you set categories capturing missing cases in age and gender as NAs (to simplify things, you can also take 'Other' in gender as missing)? And can you recode the age categories in the survey so they reflect those used in the Census?

```r
#Gender
table(parks$Q31, useNA="ifany") #This is to list all the categories used for gender.
#Then the recoding procedure using ifelse and '|' to indicate 'or'.
parks$Q31 = ifelse(parks$Q31=="Other"|parks$Q31=="Prefer not to say", NA,
                   as.character(parks$Q31))
table(parks$Q31, useNA="ifany") #This is to check that the recoding went well.
prop.table(table(parks$Q31)) #And these are the proportions of each gender in our sample,
#which we will need to use to calculate the weights.
#Now age, where we can set the NAs and undertake the necessary recoding of categories
#through one same procedure.
table(parks$Q32, useNA="ifany")
#For the recoding we could use ifelse as previously, but to keep the code a bit
#tidier and more intuitive I will use recode().
library(car)  #This is to be able to use the command recode.
parks$Q32 = recode(parks$Q32, "c('19 or younger', 'Prefer not to say')=NA;
                               c('20 - 24', '25 - 34', '35 - 44')='20 - 44';
                               c('45 - 54', '55 - 59', '60 - 64')='45 - 64'")
#Again, always important to check that the recoding process worked as expected.
table(parks$Q32, useNA="ifany")
#And the table capturing the proportions that we need to calculate the weights.
prop.table(table(parks$Q32))
```

You now have all the necessary information.

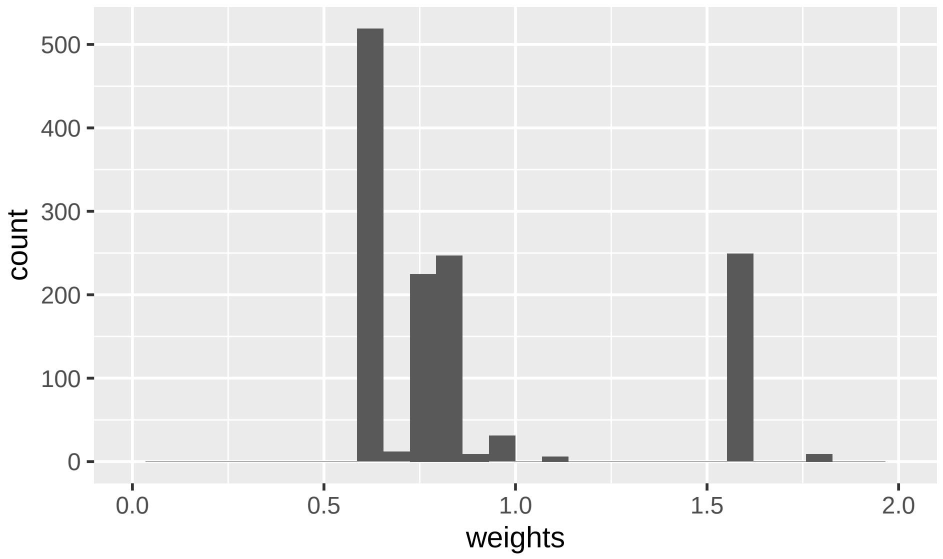
Question: Can you calculate poststratification weights for gender and age? Hint1: You should see these weights as new variables to be added to your dataset. So the first thing you can do is to create two new variables, one for the weights for gender, and another for age. At this point do not worry about the content of those variables (so for example we could start with this 'parks$wgender=c(1:1439)'), you can replace that content with the actual value that you calculate for your weights. Hint2: I recommend that you calculate the weights for gender ('Q31') first, and then do the same for age ('Q32') in a different variable, then to combine the two weights simply multiply them. Hint3: Remember from the lecture that poststratification weights are nothing more than the ratio of a given characteristic in the population divided by the ratio of that same characteristic in your sample; so for the case of gender the weights attributed to women would be 0.509/0.560, then we need to calculate those for men too, and then do the same thing for age. Hint4: Notice as well that there will be missing cases for age and gender in the sample, I recommend that you give them a weight equal to 1 since we do not the gender of these cases.

```r
#Calculating weights for gender.
#I start by creating a new variable.
parks$wgender = c(1:1439)
#Then I calculate the weights by recoding that variable.
#I do so in three stages, but we could have also used a single procedure with ifelse.
parks$wgender[parks$Q31=="Female"] = 0.509 / prop.table(table(parks$Q31))[1]
parks$wgender[parks$Q31=="Male"] = 0.491 / prop.table(table(parks$Q31))[2]
parks$wgender[is.na(parks$Q31)==TRUE] = 1
#Then I check that the transformation makes sense.
```

```r
head(parks)
table(parks$wgender) #It does men are weighted more heavily than women, which
#should help us adjust for the fact that they are underrepresented in our sample.
```

```r
#I do the same thing to calculate weights for age.
parks$wage = c(1:1439)
parks$wage[parks$Q32=="20 - 44"] = 0.494 / prop.table(table(parks$Q32))[1]
parks$wage[parks$Q32=="45 - 64"] = 0.302 / prop.table(table(parks$Q32))[2]
parks$wage[parks$Q32=="65+"] = 0.203 / prop.table(table(parks$Q32))[3]
parks$wage[is.na(parks$Q32)==TRUE] = 1
table(parks$wage)
```

```r
#Combining age and gender weights.
parks$weights = parks$wgender * parks$wage
#Visualising the final weights variable.
summary(parks$weights)
library(ggplot2)
ggplot(parks, aes(x=weights)) + geom_histogram() + xlim(0, 2)
```



One way to assess how representative our original sample is by plotting the variable capturing the weights we have calculated. In our case we can see a good share of weights that lie close to 0.6 and 1.6, that is, relatively far away from 1, so we can expect that these weights will play a meaningful role in adjusting our sample to make it more representative.

Once you have calculated your weights you need to tell R so it can understand that the variable created represents that, a set of weights. To do so you can use *svydesign*, from the *survey* package. First we need to specify whether cluster sampling was used in the sampling strategy using the *id* option. We did not use cluster sampling in this survey, so we will specify *id=~1*, but we will get back to this in Workshop 8 (Hierarchical Data), where we will use the European Social Survey, where individual respondents are clustered within regions and countries. We also need to identify the dataset to be used and the variable capturing the weights.

```r
library(survey)
parkweights = svydesign(id=~1, data=parks, weights=parks$weights)
```

Ok, now we have everything we need to estimate proportion of Leeds resident who ‘seldom or never’ visit parks, while adjusting for the fact that the original sample was not representative. Specifically, we need to use the command *svytable*.

```r
prop.table(table(parks$Q4))              #The unadjusted distribution for park use.
prop.table(svytable(~parks$Q4, parkweights)) #The adjusted distribution for park use.
```

Question: What is the 'true' proportion of residents who 'seldom or never'? And how does that differ from the estimate that we would have obtained from our original (unadjusted) sample?

We can see that the original sample gives us a slightly biased answer. In particular, that sample was more likely to capture older people, which results in underestimating how frequently Leeds residents visit their local parks. Specifically, the proportion of residents who 'seldom or never' use parks is 2.2% if we naively assume that the sample used is representative, and 2.9% after we apply the adjustment based on poststratification weights.

Preparation for the next workshop

Next week you will see another important subject, hierarchical data, where we will learn how to analyse using different strategies, such as fixed effects or multilevel modelling. That workshop is composed of only one exercise, which will be fully guided. The reason for that is the difficulty of the workshop, which is why it will be essential that you reproduce it slowly on your own in advance, and bring your questions to that session, so we can make the most of it.