Workshop 5 - Data Quality

JPS

27/12/2021

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 datasets derived from different surveys. These are: the European Social Survey, the Crown Court Sentencing Survey, and the Leeds Parks Survey.

Exercise 1: We will use the European Social Survey (arguably the most interesting and rigorous cross-country social survey worldwide) to learn how to apply probability weights to adjust for problems of selection bias in sampling. We will learn how to use this important tool to improve the validity of survey research by exploring the self-reported levels of happiness and the main factors explaining differences in happiness across individuals.

Exercise 2: Here we will practice different methods of imputation (mean, regression and multiple imputation) to adjust for problems of item-missingness in the Crown Court Sentencing Survey. 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 severely affected by non-response.

Exercise 3: Using information from the last Census regarding the age and gender distribution of Leeds residents you are requested to produce probability 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. Probability Weights

Let's start by accessing the European Social Survey (ESS), which you can get from Minerva. The ESS is a truly unique dataset, in its geographical coverage but also in the wide range of topics covered, which might be quite relevant to some of your substantive areas of interest, e.g. social trust, attitudes towards the Criminal Justice system, etc. Furthermore, the data is really easy to download, no need to submit an application, just provide some information through a short registration process and it can be downloaded directly from their website (http://www.europeansocialsurvey.org/). The version of the ESS available on Minerva has got a STATA format (.dta), so we will first need to install the package *foreign*. You will also need to modify the code below to provide the direction to the folder where you have saved the dataset.

```
library(foreign)
#setwd("C:/Users/...")
ess = read.dta("ESS9e01_1.dta")
```

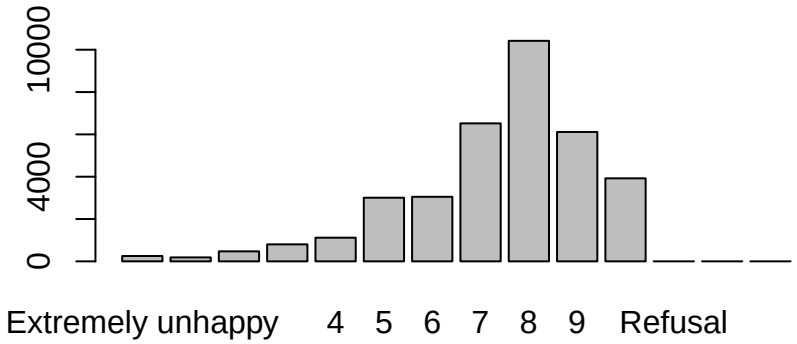
There are 491 variables in the dataset so we will start by keeping the variables that we will use in our analysis. These are: gender ('gndr'), age ('agea'), self-reported happiness ('happy'), how often meet with other people ('sclmeet'), feeling of safety walking alone in the dark ('aesfdrk'), number of people living in

the same household ('hhmmb'), and the probability weights calculated by the survey designers of the ESS ('dweight').

```
vars = c("gndr","agea","happy","sclmeet","aesfdrk","hhmmb","dweight")
ess = ess[vars]
```

We can provide a first approximation to the research question by exploring the distribution of self-reported levels of happiness.

```
plot(ess$happy)
```




```
table(ess$happy, useNA="ifany")
```

More people report to be happy than not, with 8 (out of 10) as the most common value (the mode) of happiness reported. If we want to calculate the average of that distribution we first need to transform 'happy' into a numeric variable. This is a bit tricky since 'happy' is a factor with both characters and numbers as levels. One approach we can take to transform this variable is by combining two *ifelse* commands, one for each of the character levels ('extremely happy' and 'extremely unhappy'), with the rest of the values being transformed directly into numeric using *as.numeric(as.character)*.

```
ess$happyrec = ifelse(ess$happy=="Extremely unhappy",0,
                      ifelse(ess$happy=="Extremely happy",10,as.numeric(as.character(ess$happy))))
class(ess$happyrec)   #This is to check that happyrec is now a numeric variable
table(ess$happyrec, useNA="ifany")  #This is to check that the transformation went ok
```

ok, we have now 'happyrec' as a numeric variable, so we can calculate the average using *mean*, but to do so we need to specify the option *na.rm=TRUE* to drop the 148 missing cases in 'happyrec'.

```
mean(ess$happyrec)
mean(ess$happyrec, na.rm=TRUE)
```

The mean is 7.38. Let's now apply the design weights available in the ESS to calculate the weighted mean and assess the extent to which the different selection probabilities for different demographic groups could be affecting the validity of our estimations. Quoting the ESS Weighting European Social Survey Data report, "*The main purpose of the design weights is to correct for the fact that in some countries respondents have different probabilities to be part of the sample due to the sampling design used. Applying the weights allows to correct for this and obtain estimates that are not affected by a possible sample selection bias.*''

To apply the ESS weights we will use the *survey* package. To tell R about the configuration of the weights we

need to use the *svydesign* command. We need to specify whether cluster sampling was used in the sampling strategy using the *id* option. Today we will assume no clustering, so we will specify *id=~1*, but we will get back to this in Workshop 7 (Hierarchical Data), since we can identify the different countries and even the regions from where the data has been collected as clusters. Other than *id* we need to identify the dataset to be used and the variable capturing the weights.

```
library(survey)
essweighted = svydesign(id=~1, data=ess, weights=ess$dweight)
```

Once we have set up the weights through *svydesign* we need to employ specific commands from the *survey* package to obtain weighted estimates. Let's see a few examples by calculating means for 'happyrec' and 'agea' and a frequency table for 'gndr', requesting both the unadjusted and adjusted estimates.

```
mean(ess$happyrec, na.rm=TRUE)              #Unadjsuted mean
```

```
## [1] 7.377394
```

```
svymean(~happyrec, essweighted, na.rm=TRUE) #Adjusted (weighted) mean
```

```
##             mean     SE
## happyrec 7.4199 0.0106
```

```
mean(ess$agea, na.rm=TRUE)
```

```
## [1] 50.73109
```

```
svymean(~agea, essweighted, na.rm=TRUE)
```

```
##        mean     SE
## agea 49.756 0.1035
```

```
prop.table(table(ess$gndr))                 #Unadjusted frequency table
```

```
##
##      Male    Female No answer
## 0.4715258 0.5284742 0.0000000
```

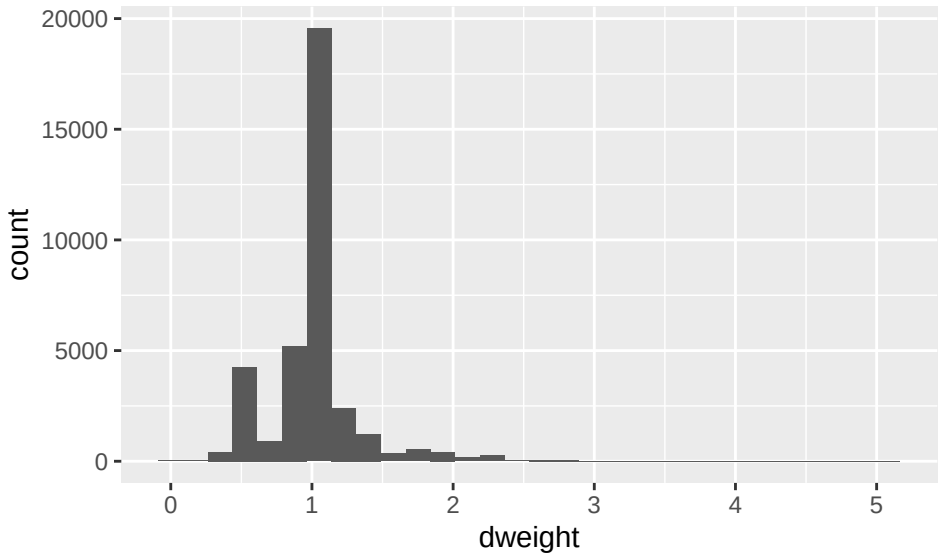
```
prop.table(svytable(~gndr, essweighted))    #Adjusted (weighted) freq. table
```

```
## gndr
##      Male    Female No answer
## 0.4767499 0.5232501 0.0000000
```

We can see some differences between the unadjusted and weighted estimates, for example, if we do not adjust for missing data we are over-representing older people, but the differences are almost negligible. This is a good testament to the quality of the ESS. We will see in Exercise 3 how weights can be very useful in correcting biased estimates from surveys that cannot achieve the same level of rigour.

One way you can anticipate the impact that adjustments based on weights will have is by plotting the actual variable capturing those weights.

```
library(ggplot2)
ggplot(ess, aes(x=dweight)) + geom_histogram()
```



For the case of 'dweight' most weights can be seen to lie close to 1, that is, the weight attributed to most cases in the adjusted estimates is not different from the unadjusted estimates.

We can also use these weights to carry out *weighted regression*, which we will need to explore our second research question: What are the factors explaining self-reported levels of happiness? Let's first run a standard (unweighted) linear model to explore this.

```
summary(ess)
```

```
lm1 = lm(happyrec~sclmeet+aesfdrk+hhmmb+gndr+agea, data=ess)
summary(lm1)
```

```
##
## Call:
## lm(formula = happyrec ~ sclmeet + aesfdrk + hhmmb + gndr + agea,
##     data = ess)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.5255 -0.8694  0.2410  1.1991  5.6604
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   5.7098456  0.0861606  66.270   <2e-16 ***
## sclmeetLess than once a month 0.7417776  0.0770670   9.625   <2e-16 ***
## sclmeetOnce a month           1.3837143  0.0768993  17.994   <2e-16 ***
## sclmeetSeveral times a month  1.6586188  0.0736056  22.534   <2e-16 ***
## sclmeetOnce a week            1.8106044  0.0741301  24.425   <2e-16 ***
## sclmeetSeveral times a week   1.9909348  0.0732101  27.195   <2e-16 ***
## sclmeetEvery day              2.0340973  0.0759210  26.792   <2e-16 ***
## aesfdrkSafe                  -0.4711031  0.0227905 -20.671   <2e-16 ***
## aesfdrkUnsafe                -1.0548523  0.0309023 -34.135   <2e-16 ***
## aesfdrkVery unsafe           -1.4728539  0.0521360 -28.250   <2e-16 ***
## hhmmb                         0.1487007  0.0078050  19.052   <2e-16 ***
## gndrFemale                    0.2263005  0.0198151  11.421   <2e-16 ***
## agea                         -0.0005991  0.0005811  -1.031    0.303
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.801 on 35195 degrees of freedom
##   (807 observations deleted due to missingness)
## Multiple R-squared:  0.1195, Adjusted R-squared:  0.1192
## F-statistic: 398.1 on 12 and 35195 DF,  p-value: < 2.2e-16
```

All factors are significant and pointing in the expected direction. The only exception being age, although as we saw last week this apparent non-significant effect might be due to having wrongly assumed a linear relationship. This is something worth exploring, if you remember how to specify quadratic effects do it for the continuous explanatory variables included in the model, 'agea' and 'hhmb'.

One more thing to notice that will be relevant in the next exercise is how the model has dropped 807 observations due to missingness (reported at the bottom of the *summary* output). The *lm* command has automatically undertaken a listwise deletion. That is, all cases affected by item-missingness in at least one of the variables included in the model were dropped. To examine those 807 observations you can use the *complete.cases* command. Notice that since item-missingness is not uniform across the dataset, depending on which variables you end up including in your model you will be using a different sample. In the interest of consistency, and robust model comparisons, it might be a good idea to undertake the listwise deletion ourselves so any models we want to compare will always be using the same sample size. Let's do that now, in the next exercise we will see how imputation methods can help us find a better solution.

```
ess[!complete.cases(ess),] #To identify the cases affected by item-missingness
ess_noNA = na.omit(ess)    #To undertake listwise deletion in our dataset
#Once we do this ourselves lm does not have to drop these cases
```

```
lm1 = lm(happyrec~sclmeet+aesfdrk+hhmmb+gndr+agea, data=ess_noNA)
summary(lm1)
```

```
##
## Call:
## lm(formula = happyrec ~ sclmeet + aesfdrk + hhmmb + gndr + agea,
##     data = ess_noNA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.5255 -0.8694  0.2410  1.1991  5.6604
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   5.7098456  0.0861606  66.270   <2e-16 ***
## sclmeetLess than once a month  0.7417776  0.0770670   9.625   <2e-16 ***
## sclmeetOnce a month            1.3837143  0.0768993  17.994   <2e-16 ***
## sclmeetSeveral times a month   1.6586188  0.0736056  22.534   <2e-16 ***
## sclmeetOnce a week             1.8106044  0.0741301  24.425   <2e-16 ***
## sclmeetSeveral times a week    1.9909348  0.0732101  27.195   <2e-16 ***
## sclmeetEvery day               2.0340973  0.0759210  26.792   <2e-16 ***
## aesfdrkSafe                   -0.4711031  0.0227905 -20.671   <2e-16 ***
## aesfdrkUnsafe                 -1.0548523  0.0309023 -34.135   <2e-16 ***
## aesfdrkVery unsafe            -1.4728539  0.0521360 -28.250   <2e-16 ***
## hhmmb                          0.1487007  0.0078050  19.052   <2e-16 ***
## gndrFemale                     0.2263005  0.0198151  11.421   <2e-16 ***
## agea                          -0.0005991  0.0005811  -1.031    0.303
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.801 on 35195 degrees of freedom
## Multiple R-squared:  0.1195, Adjusted R-squared:  0.1192
## F-statistic: 398.1 on 12 and 35195 DF,  p-value: < 2.2e-16
```

To undertake weighted regression we will use the command *svyglm*, which allows us to specify any kind of generalised linear models, including logistic and standard linear models. To tell R the type of model to be estimated we need to specify the type of distribution assumed for our residuals using the *family* option. In the case of a standard linear model we want our residuals to be Gaussian, i.e. normally distributed. We also need to identify the dataset to be used, and the *svy* object where we set up the configuration of the weights.

```
lmweighted = svyglm(happyrec~sclmeet+aesfdrk+hhmmb+gndr+agea,
                    family=gaussian(), data=ess_noNA, design=essweighted)
summary(lmweighted)
```

```
##
## Call:
## svyglm(formula = happyrec ~ sclmeet + aesfdrk + hhmmb + gndr +
##     agea, design = essweighted, family = gaussian(), data = ess_noNA)
##
## Survey design:
## svydesign(id = ~1, data = ess, weights = ess$dweight)
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   5.9190029  0.1272805  46.504  < 2e-16 ***
## sclmeetLess than once a month 0.6975647  0.1210866   5.761 8.44e-09 ***
## sclmeetOnce a month           1.3163393  0.1197027  10.997  < 2e-16 ***
## sclmeetSeveral times a month  1.5820916  0.1163234  13.601  < 2e-16 ***
## sclmeetOnce a week            1.7472351  0.1165707  14.989  < 2e-16 ***
## sclmeetSeveral times a week   1.9104318  0.1159083  16.482  < 2e-16 ***
## sclmeetEvery day              1.9559430  0.1186060  16.491  < 2e-16 ***
## aesfdrkSafe                  -0.4574523  0.0224539 -20.373  < 2e-16 ***
## aesfdrkUnsafe                -1.0411581  0.0356256 -29.225  < 2e-16 ***
## aesfdrkVery unsafe           -1.4427846  0.0730181 -19.759  < 2e-16 ***
## hhmmb                         0.1109469  0.0098721  11.238  < 2e-16 ***
## gndrFemale                    0.2282906  0.0207729  10.990  < 2e-16 ***
## agea                         -0.0014333  0.0006338  -2.261   0.0237 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 3.185389)
##
## Number of Fisher Scoring iterations: 2
```

As it was the case for the weighted means and frequency tables, we can observe some differences between the coefficients of the weighted and the unadjusted models. For example, the linear effect of age is now stronger, rendering the estimate statistically significant. This is a small but important difference since the unadjusted model would have lead us to misinterpret the effect of this key variable.

We can use the same command to specify any kind of generalised linear models, the logistic model is one of them. Let's do that by taking happiness as a binary outcome. Notice how we need to specify now a binomial distribution in the *family* option.

```
ess_noNA$happybin = ifelse(ess_noNA$happyrec>5,1,0)
#The following line is to make sure that happybin is considered in the survey design.
essweighted = svydesign(id=~1, data=ess_noNA, weights=ess_noNA$dweight)
logitweighted = svyglm(happybin~sclmeet+aesfdrk+hhmmb+gndr+agea,
```

```
                        family=binomial, data=ess_noNA, design=essweighted)
summary(logitweighted)
```

The same process can be undertaken with more complex models. And for different datasets. Official, national and international surveys - and basically any proper survey - will provide a set of probability weights. When using survey data in the future, make sure that you apply them in your analysis, which will help you adjust for problems of selection bias, unit non-response, within cluster correlation, and many other issues affecting sampling designs.

Exercise 2. 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. I am a bit embarrassed to confess that, as far as I am aware, only the first of those methods has been used in the sentencing literature to adjust for the problem of item-missingness affecting the CCSS. So, after finishing this exercise, you will be equipped to surpass the current methodological frontier in sentencing research.

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.

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

In Week 2 we used the *merge* command to merge two datasets horizontally (we added new variables), here we are going to merge two datasets vertically (to add more cases to the same variables), this is often referred to as appending - rather than merging - data. This procedure only works if we have the same variables in each of the datasets to be merged. The following procedure can be used to identify variables that are only present in one of the datasets. Since we find that the same variables are captured in the two datasets we also proceed to append them.

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

```
names(assa)
vars = c("OFFENCE","OUTCOME","SERIOUSNESS","APO_GREATER_HARM_1","APO_GREATER_HARM_2","APO_GREATER_HARM_3
#APO_HIGHER_CULP_STAT_2, APO_HIGHER_CULP_STAT_3, and APO_HIGHER_CULP_STAT_4 could also be included but
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 offences sentenced in the Crown Court.

```
table(assa$OFFENCE, useNA="ifany")
assa = assa[which(assa$OFFENCE=="S.47"),]
assa$OFFENCE = NULL
```

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

```
table(assa$OUTCOME, useNA="ifany")
assa$CUSTODY = ifelse(assa$OUTCOME=="Immediate custody", 1, 0)
assa$OUTCOME = NULL
```

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)
#Changing labels for APO_LESSER_HARM so all Step-One factors follow the same formatting
assa$APO_LESSER_HARM_1 = ifelse(assa$APO_LESSER_HARM_1=="None stated","-",as.character(assa$APO_LESSER_H
#Setting missing cases in PREV_CONVICTIONS as NA
assa$PREV_CONVICTIONS = ifelse(assa$PREV_CONVICTIONS=="Not answered",NA,as.character(assa$PREV_CONVICTI
assa$PREV_CONVICTIONS = factor(assa$PREV_CONV, levels = c("None", "1 to 3", "4 to 9", "10 or more"))
#Same with SERIOUSNESS
table(assa$SERIOUSNESS, useNA="ifany")
assa$SERIOUSNESS = ifelse(assa$SERIOUSNESS=="Not answered"|assa$SERIOUSNESS=="No existing guideline"|
                          assa$SERIOUSNESS=="Not Answered"|assa$SERIOUSNESS=="Not asked",NA,assa$SERIOUS
assa$SERIOUSNESS = factor(assa$SERIOUSNESS)
```

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). As we can see from the model output, there are 1037 cases dropped from our sample for that reason.

```
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)
```

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 visuallise these problems by undertaking the mean imputation of 'SERIOUSNESS'. 'SERIOUSNESS' is an ordinal variable so technically we will be using median imputation.

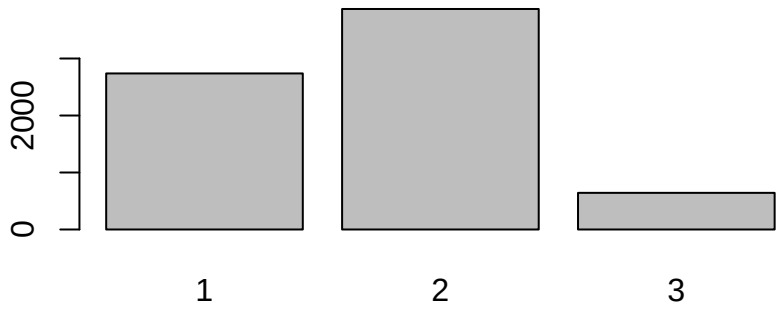
```
table(assa$SERIOUSNESS, useNA="ifany")
```

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

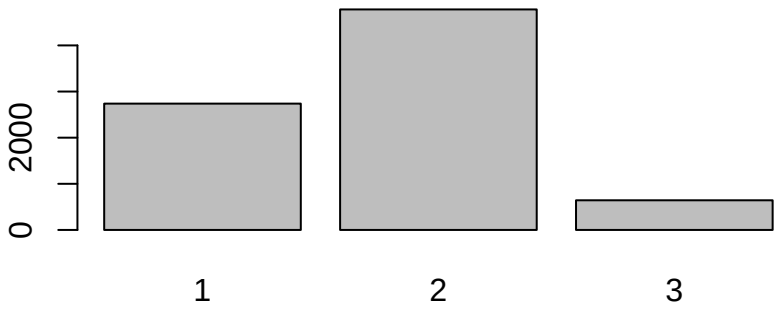
```
#Below we are saying, if SERIOUSNESS is missing replace it by a 2 (its median)
assa$SERIOUS_MEANIMP = factor(ifelse(is.na(assa$SERIOUSNESS)==TRUE, 2, assa$SERIOUSNESS))
table(assa$SERIOUS_MEANIMP)
```

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

```
#This is to compare the distribution of the unadjusted and imputed variables
plot(assa$SERIOUSNESS)
```




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




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

```
## [1] 0.6187864
```

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

```
## [1] 0.5902803
```

```
assa$SERIOUS_MEANIMP = NULL
```

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. 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 form of logistic regression.

The logistic model we have used so far is 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 in a binary logistic model. That is, they are expressed in the same unit of measurement, log-odds. To specify the ordered logit model we will use the *polr* command from the *MASS* package.

```
#Regression imputation
library(MASS)
ologit = polr(SERIOUSNESS ~ APO_GREATER_HARM_1 + APO_GREATER_HARM_2 + APO_GREATER_HARM_3 + APO_LESSER_HA
              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_HA
              APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 + APO_LOWER_CULP_4 + APO_LOWER_CULP
              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.

```
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
```

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

```
## [1] 0.6187864
```

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

```
## [1] 0.6057947
```

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
                 APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 + APO_LOWER_CULP_4 + APO_LOWER_C
                 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. In any case, the fact that most of the factors we use for the prediction of missing values are strongly associated with the missing data mechanism (the probability of missingness), means that 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.

```
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)
summary(logit1)
summary(logit2)
```

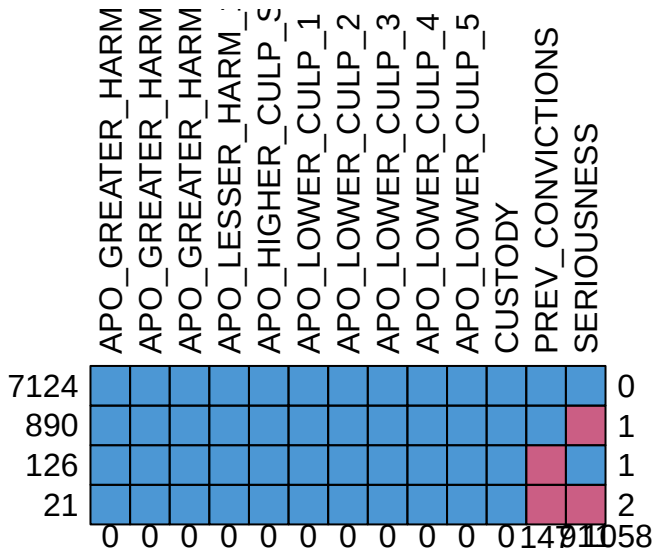
We can see certain differences between the unadjusted and the adjusted model. The effect of 'SERIOUSNESS' on the probability of custody is stronger in the former, which might lead to wrong interpretations of the importance of this factor. We can also see how 'SERIOUSNESS' is not the only biased coefficient. Furthermore, we can see how the number of cases dropped through listwise deletion has gone down from 1037 to 147, which represent those cases 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 of them with 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
library(mice)
md.pattern(assa, rotate.names=TRUE)
```



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, sort of similar to the *svydesign* command that we use to set up probability weights.

```
#The multiple imputation process.
miassa = mice(assa,m=5,meth='pmm',seed=7)
#To check the imputed values for 'SERIOUSNESS' and 'PREV_CONVICTIONS'.
#Each row represents a missing value imputed, each column represents each one of the 5 imputations requ
miassa$imp$SERIOUSNESS
miassa$imp$PREV_CONVICTIONS
#To observe 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).

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_HA
                          APO_LESSER_HARM_1 + APO_HIGHER_CULP_STAT_1 + APO_LOWER_CULP_1 + APO_LOWER_CULI
                          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 wanted 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.

```
#Regression imputation for missing cases in 'SERIOUSNESS'
ologit = polr(SERIOUSNESS ~ APO_GREATER_HARM_1 + APO_GREATER_HARM_2 + APO_GREATER_HARM_3 + APO_LESSER_HA
              APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 + APO_LOWER_CULP_4 + APO_LOWER_CULI
              CUSTODY, data=assa, Hess=TRUE)
assa$SERIOUS_REGIMP = factor(ifelse(is.na(assa$SERIOUSNESS)==TRUE, predict(ologit), assa$SERIOUSNESS))
#Regression imputation for missing cases in 'PREV_CONVICTIONS'
ologit = polr(PREV_CONVICTIONS ~ APO_GREATER_HARM_1 + APO_GREATER_HARM_2 + APO_GREATER_HARM_3 + APO_LESS
              APO_LOWER_CULP_1 + APO_LOWER_CULP_2 + APO_LOWER_CULP_3 + APO_LOWER_CULP_4 + APO_LOWER_CULI
              CUSTODY, data=assa, Hess=TRUE)
assa$PREVCONV_REGIMP = factor(ifelse(is.na(assa$PREV_CONVICTIONS)==TRUE, predict(ologit), assa$PREV_CONV
#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)
```

```
summary(logit2)$coefficients[,2]  #Standard errors from the model imputing 'SERIOUSNESS' using regressi
```

```
##                                                                    (Intercept)
##                                                                     0.06660479
##                                                                 SERIOUS_REGIMP2
##                                                                     0.06064566
##                                                                 SERIOUS_REGIMP3
##                                                                     0.13250108
## APO_GREATER_HARM_1Injury/fear of injury which is serious in context of the offence
##                                                                     0.06597029
##                                    APO_GREATER_HARM_2Victim particularly vulnerable
##                                                                     0.07019241
##                     APO_GREATER_HARM_3Sustained or repeated assault on same person
##                                                                     0.05847242
##      APO_LESSER_HARM_1Injury/ fear of injury which is less serious in context of t
##                                                                     0.08400456
##                                                 APO_HIGHER_CULP_STAT_1Race/religion
##                                                                     0.36843068
##                                  APO_LOWER_CULP_1Subordinate role in group or gang
##                                                                     0.19062747
##                                     APO_LOWER_CULP_2Greater degree of provocation
##                                                                     0.14892715
##                                               APO_LOWER_CULP_3Lack of premeditation
##                                                                     0.07842583
##            APO_LOWER_CULP_4Mental disorder/learning disability where linked to the
##                                                                     0.21026965
##                                             APO_LOWER_CULP_5Excessive self defence
##                                                                     0.17071252
##                                                                 PREVCONV_REGIMP2
##                                                                     0.05851514
##                                                                 PREVCONV_REGIMP3
##                                                                     0.08479119
##                                                                 PREVCONV_REGIMP4
##                                                                     0.14785815
```

```
summary(pool(logit3))[,3]         #Standard errors from the model imputing 'SERIOUSNESS' using multiple
```

```
##  [1] 0.08670989 0.09358025 0.21801033 0.07136947 0.07374538 0.06606634
##  [7] 0.08527735 0.36956990 0.19174872 0.14963088 0.08095336 0.21152991
## [13] 0.17280431 0.06200549 0.08570200 0.14985723
```

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 3. 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 you will needs to use the *read.spss* command (from the library *foreign*), including the option *to.data.frame=TRUE*.

-I recommend that you calculate the weights for gender ('Q31') first, and then do the same for age ('Q32') in a different variable.

-Notice that the survey doesn't use the same categories for age as those used in the Census, so a little recoding will be necessary.

-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.

-To combine the two weights simply multiply them.

-Once you have calculated your weights set them up using *svydesign*, as we did in Exercise 1.

-Then, provide an adjusted estimate of the proportion of Leeds resident who 'seldom or never' visit parks. To do so you can use *svytable*.

##########To be removed from the handout##########

```
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 val
```

Exploratory analysis. I recode gender ('Q31') and age ('Q32') to make sure missing cases are denoted as NAs and the different age categories in the survey reflect those used in the Census.

```
summary(parks)
#Gender
table(parks$Q31, useNA="ifany")
parks$Q31 = ifelse(parks$Q31=="Other"|parks$Q31=="Prefer not to say", NA, as.character(parks$Q31))
table(parks$Q31, useNA="ifany")
prop.table(table(parks$Q31))
#Age
table(parks$Q32, useNA="ifany")
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'")
table(parks$Q32, useNA="ifany")
prop.table(table(parks$Q32))
```

Calculating weights for gender.

```
parks$wgender = c(1:1439)
parks$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$Q31)==TRUE] = 1
table(parks$wgender)
```

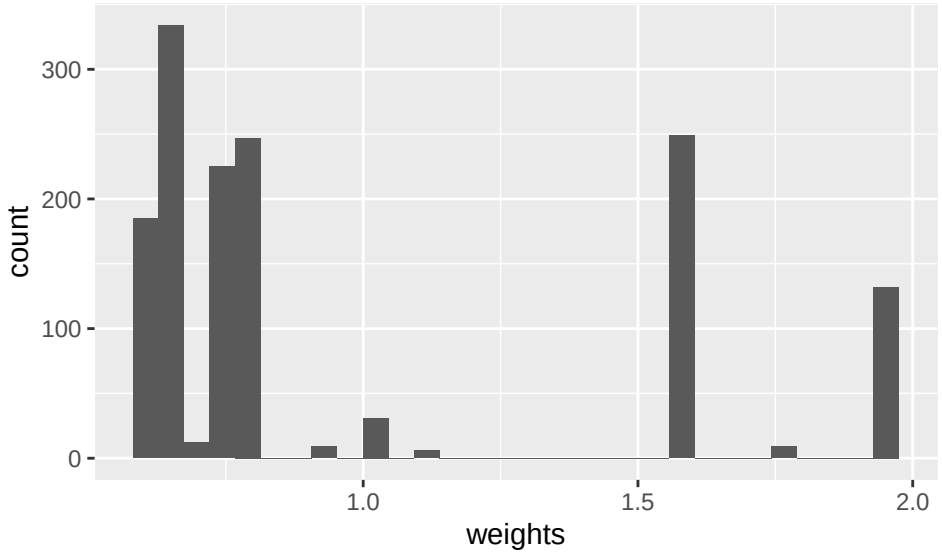
Calculating weights for age.

```
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)
```

Combining age and gender weights.

```
parks$weights = parks$wgender * parks$wage
library(ggplot2)
ggplot(parks, aes(x=weights)) + geom_histogram()
```



Applying the weights.

```
library(survey)
parkweights = svydesign(id=~1, data=parks, weights=parks$weights)
prop.table(table(parks$Q4))        |#The unadjusted distribution for park use.
prop.table(svytable(~parks$Q4, parkweights))  #The adjusted distribution for park use.
```

```
##
##       Almost every day   Once or twice a week   Once every two weeks
##                   TRUE                   TRUE                   TRUE
##           Once a month Less than once a month        Seldom or never
##                   TRUE                   TRUE                   TRUE
```

It seems that the sample was biased. In particular it was more likely to capture older people, which resulted 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.