Design Considerations in Conjoint Analysis

William Hughes

May 29, 2013

Abstract

The package was inspired by a conjoint analysis survey that I was asked to help analyze. While the results were available, the original design justification was not. In particular we did not know which cases were meant to be ”holdouts”.

Note that I am considering the problem of using conjoint analysis to assess preferences. If the goal is different (e.g. finding optimal machine settings) the interpretation of things like “noise” is different, and there may be different design criteria.

1 Design

1.1 intro

It is usually noted that the design chosen for a conjoint analysis is very important. However, aside from some remarks on the minimum number of cards that need to be chosen from the full factorial design, and the unhelpful advice that this number should be made larger to account for noise, most summaries written for the practitioner tell you very little (And most stuff written for the Statistician either tells you nothing, or assumes that you already know the basics.)

In practice, the design is chosen by deciding on a number of cards, then asking some computer program for a design with about that number of cards. Indeed, if the number of cards chosen is reasonable, the default settings of most packages will result in a good model. However, this does not allow for trade offs (e.g. exact orthogonality vs size), nor is “The Computer told me” a very satisfying answer if someone asks “Why did you choose this model?”

I will try to give some indication of why certain design criteria are important by considering a simple example.

Suppose we are studying factors that managers use to hire people. We look at four factors, university attended, Prestige, Excellent, Good; sex, male or female; Dress, smart or messy; and hair, long or short.

The managers are all the same. They will pick first by university, Prestige, Excellent, Good in that order, within a university they will first pick a male over a female, and finally they will pick a smartly dressed candidate over a messily dressed candidate. The managers do not care about hair length

We can get R to produce the full factorial set of cards.

```
> experiment = expand.grid(
+ University=c("Prestige","Excellent","Good"),
+ Sex=c("Male","Female"),
+ Dress =c("smart","messy"),
+ Hair=c("long","short"))
> print(experiment)
```

```
   University    Sex Dress  Hair
1    Prestige   Male smart  long
2   Excellent   Male smart  long
3        Good   Male smart  long
4    Prestige Female smart  long
5   Excellent Female smart  long
6        Good Female smart  long
7    Prestige   Male messy  long
8   Excellent   Male messy  long
9        Good   Male messy  long
10   Prestige Female messy  long
11  Excellent Female messy  long
12       Good Female messy  long
13   Prestige   Male smart short
14  Excellent   Male smart short
15       Good   Male smart short
16   Prestige Female smart short
17  Excellent Female smart short
18       Good Female smart short
19   Prestige   Male messy short
20  Excellent   Male messy short
21       Good   Male messy short
22   Prestige Female messy short
23  Excellent Female messy short
24       Good Female messy short
```

In this simple case we have 24 cards. It might be practical to use all possible cards (usually it is not) but we will consider a subset of the cards anyway.

We will use a simple model.

$$r = \beta_{11}X_{11} + \beta_{12}X_{12} + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + d \tag{1}$$

Where r is the rank, β_i are the coefficients to be estimated, the X_i are the variables, University (coded as two variables), Sex, Well Dressed and Hair, and d is the intercept.

Note that this model does not directly reflect how the managers make decisions. We hope that it can come close enough.

In matrix form we have

$$R = \mathbf{X}\beta + \mathbf{d} \tag{2}$$

This is a bit ugly, so we absorb the intercept by prepending a column of ones to $\mathbf{X}$ and adding a β_0 obtaining

$$R = \mathbf{X}\beta \tag{3}$$

We are assuming that there are no significant cross terms (e.g. that there are no managers that like short hair on males and long hair on females).

We note that there are 6 parameters to be estimated, so we will need at least 6 cards (this leaves nothing for noise, but we can reasonably assume that noise is low. More significant is the fact that the model does not fit perfectly with real life, and we do not have any extra stuff to take care of this "noise".)

The solution is

$$\beta = (\mathbf{X^TX})^{-1}\mathbf{XR} \tag{4}$$

So the minimum requirement is that the "information matrix" $\mathbf{X^TX}$ be invertible.

1.2 Orthogonality

We start by taking the first six lines.

```
> first.six = experiment[c(1,2,3,4,5,6),]
> print(first.six)
```

```
  University    Sex Dress Hair
1   Prestige   Male smart long
2  Excellent   Male smart long
3       Good   Male smart long
4   Prestige Female smart long
5  Excellent Female smart long
6       Good Female smart long
```

A glance at this tells us that it is not very good. For one thing we only have cases for long hair. How are we supposed to tell whether managers are interested in hair length if we never ask about it?

Indeed, the information matrix (when augmented) will be singular. Lets set up an analysis anyway. Let us take ranks from worst to best, so positive utility is good. We can get the ranks with a bit of R magic. Note that we take the ranks of all cases, not just the first six lines

```
> ranks <- function(mat) ((NROW(mat)+1)-order(do.call(order, lapply(1:(NCOL(mat)-1), functio
> rnks = ranks(data.matrix(experiment))
> print(rnks)
```

```
 [1] 24 16  8 20 12  4 22 14  6 18 10  2 23 15  7 19 11  3 21 13  5 17  9  1
```

(It is a fun R exercise to figure out why the ranks function works, but it is easy to check by hand.)

(The package can work out the ranks that would be produced if we only have the first six lines)

```
> despack=NULL
> despack$designs[[1]]=first.six
> M.Conjoint(despack,rnks)
```

and the output is

```
Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]) :
  contrasts can be applied only to factors with 2 or more levels
```

Not the most user friendly error message but the package thinks there is a problem. We pick a new set of 11 cases.

```
> design.two=experiment[c(1,3,18,7,16,17,8,22,23,10,24),]
> print(design.two)
```

```
   University    Sex Dress  Hair
1    Prestige   Male smart  long
3        Good   Male smart  long
18       Good Female smart short
7    Prestige   Male messy  long
16   Prestige Female smart short
17  Excellent Female smart short
8   Excellent   Male messy  long
22   Prestige Female messy short
23  Excellent Female messy short
10   Prestige Female messy  long
24       Good Female messy short
```

The ranks are

```
> rns = ranks(data.matrix(design.two))
> print(rns)
```

```
 [1] 11  3  2 10  9  5  6  8  4  7  1
```

Note that to get the ranks for cases 8 and 10 the manager had to flip a coin.

```
> rnks1=rnks
> rnks1[10]=17
> rnks1[22]=18
> despack=NULL
> despack$designs=NULL
> despack$designs[[1]]=design.two
> M.Conjoint(despack,rnks1)
```

```
Average values are output
They may or may not be meaningful

Utilities

intercept  5.690
Prestige   3.810
Excellent -0.262
Good      -3.548
Male       1.464
Female    -1.464
smart      0.429
messy     -0.429
long      -0.607
short      0.607

Importances

University 0.59537572
Sex        0.23699422
Dress      0.06936416
Hair       0.09826590
```

The program has incorrectly assigned an importance to hair length, more than the importance of being well dressed. What went wrong?

Well, lets check the correlation of the variables

```
> cor(data.matrix(design.two))

            University        Sex       Dress       Hair
University  1.00000000 0.06185896 -0.19920477  0.2390457
Sex         0.06185896 1.00000000  0.06900656  0.8280787
Dress      -0.19920477 0.06900656  1.00000000 -0.1000000
Hair        0.23904572 0.82807867 -0.10000000  1.0000000
```

The chosen design is not orthogonal, that is the variables are not pairwise uncorrelated Indeed we note that the correlation between Sex and Hair is high (>.8).

Now there is no perfectly orthogonal design that uses 11 cards, but we can do much better

```
> design.three=experiment[c(1,3,4,8,11,14,17,18,19,21,22),]
> print(design.three)

   University    Sex Dress  Hair
1    Prestige   Male smart  long
3        Good   Male smart  long
```

```
4    Prestige Female smart  long
8   Excellent   Male messy  long
11  Excellent Female messy  long
14  Excellent   Male smart short
17  Excellent Female smart short
18       Good Female smart short
19   Prestige   Male messy short
21       Good   Male messy short
22   Prestige Female messy short
```

```
> print(cor(data.matrix(design.three)))
```

```
           University        Sex      Dress      Hair
University  1.0000000 -0.1256562 -0.1256562 0.1256562
Sex        -0.1256562  1.0000000 -0.1000000 0.1000000
Dress      -0.1256562 -0.1000000  1.0000000 0.1000000
Hair        0.1256562  0.1000000  0.1000000 1.0000000
```

(I will explain below how I got this model) We note that the cross correlations are all about .1

The ranks for the new design are

```
> rnk = ranks(data.matrix(design.three))
> print(rnk)
```

```
 [1] 11  3  9  6  4  7  5  1 10  2  8
```

```
> despack=NULL
> despack$designs=NULL
> despack$designs[[1]]=design.three
> M.Conjoint(despack,rnks)
```

```
Average values are output
They may or may not be meaningful

Utilities

intercept  5.5
Prestige   4.0
Excellent  0.0
Good      -4.0
Male       1.0
Female    -1.0
smart      0.5
messy     -0.5
long       0.0
short      0.0
```

```
Importances

University 7.272727e-01
Sex        1.818182e-01
Dress      9.090909e-02
Hair       1.376224e-17
```

And we see that for this design we (correctly) have that hair has no effect.

Thus when we chose a design we have to make sure that no unimportant effects are correlated with an important effect. Since we do not know what effects are important (that is what we are trying to find out) we need to choose a design that has no large correlations. This leads immediately, to an orthogonal design in which all the correlations are zero. Unfortunately, orthogonal designs do not always exist. Practically, however, an almost orthogonal design is fine.

1.3 Balance

Now consider the design

```
> design.four=experiment[c(1,4,7,10,13,16,18,20),]
> print(design.four)

   University    Sex Dress  Hair
1    Prestige   Male smart  long
4    Prestige Female smart  long
7    Prestige   Male messy  long
10   Prestige Female messy  long
13   Prestige   Male smart short
16   Prestige Female smart short
18       Good Female smart short
20  Excellent   Male messy short
```

We note the university is almost always Prestige. Do a conjoint analysis.

```
> despack=NULL
> despack$designs=NULL
> despack$designs[[1]]=design.four
> M.Conjoint(despack,rnks)

Average values are output
They may or may not be meaningful

Utilities

intercept  3.0
Prestige   2.0
```

```
Excellent -1.0
Good      -1.0
Male       1.5
Female    -1.5
smart      1.0
messy     -1.0
long       0.5
short     -0.5

Importances

University 0.3333333
Sex        0.3333333
Dress      0.2222222
Hair       0.1111111
```

Not a very good result. The importance of university is badly underestimated. And according to the utilities, there is no difference between an excellent university and a good university.

Let us simulate some "noise" by assuming one pair of cards (the last pair corresponding to lines 18 and 20) was misranked

```
> rnks1=rnks
> rnks1[c(18,20)]= c(13,3)
> despack=NULL
> despack$designs=NULL
> despack$designs[[1]]=design.four
> M.Conjoint(despack,rnks1)

Average values are output
They may or may not be meaningful

Utilities

intercept  3.0
Prestige   2.0
Excellent -2.0
Good       0.0
Male       1.5
Female    -1.5
smart      1.0
messy     -1.0
long       0.5
short     -0.5

Importances
```

```
University 0.4
Sex        0.3
Dress      0.2
Hair       0.1
```

Now an excellent university is a lot worse than a good university.
So pick a new model

```
>              design.five=experiment[c(2,5,7,12,15,16,20,23),]
>              print(design.five)
```

```
   University    Sex Dress  Hair
2   Excellent   Male smart  long
5   Excellent Female smart  long
7    Prestige   Male messy  long
12       Good Female messy  long
15       Good   Male smart short
16   Prestige Female smart short
20  Excellent   Male messy short
23  Excellent Female messy short
```

We note that the universities are much better distributed. Do the analysis

```
> despack=NULL
> despack$designs=NULL
> despack$designs[[1]]=design.five
> M.Conjoint(despack,rnks)
```

```
Average values are output
They may or may not be meaningful

Utilities

intercept  4.50
Prestige   3.00
Excellent  0.00
Good      -3.00
Male       0.75
Female    -0.75
smart      0.25
messy     -0.25
long       0.25
short     -0.25

Importances

University 0.70588235
```

```
Sex        0.17647059
Dress      0.05882353
Hair       0.05882353

>
```

Note that the relative importance of university is much better and that an excellent university is much better than a good university. Simulate some "noise"

```
> rnks1=rnks
> rnks1[c(15,23)]= c(9,7)
> despack=NULL
> despack$designs=NULL
> despack$designs[[1]]=design.five
> M.Conjoint(despack,rnks1)

Average values are output
They may or may not be meaningful

Utilities

intercept  4.583
Prestige   2.917
Excellent -0.333
Good      -2.583
Male       1.000
Female    -1.000
smart      0.500
messy     -0.500
long       0.250
short     -0.250

Importances

University 0.61111111
Sex        0.22222222
Dress      0.11111111
Hair       0.05555556
```

Things have not changed very much. In particular an excellent university still has a much higher utility than a good university.

Thus we see that having a good "balance" of universities is useful. In general, "balance", having close to equal numbers of each level for a variable, is a good thing. The resulting design tends to give better results and is more resistant to noise.

1.4 Finding a Design

1.4.1 Classical Designs

So how can we get such designs. One way is to take a so called “classical” design. These go under a number of names, e.g. “fractional factorial”. Classical designs are both orthogonal and well balanced. Usually a classical design will only work for a given number of factors and levels, and for a given number of cards. In practice, you find a classical design which fits your problem (number of factors and levels) and look for a number of cards close to what you want. This does not always work. There may be no known classical design of the type you want for the problem you have. Even if there is, the number of cards may be wrong. One set of classical designs, the “fractional factorial” designs exist for a wide range of problems. So one trick is to specify a fractional factorial design, and give a minimum number of cards. A design will be produced, but may have a lot more cards than the specified minimum.

1.4.2 Computer Search

Another method is too keep searching through designs till you find a good one. This would be ridiculously tedious by hand, but is quite doable by computer.

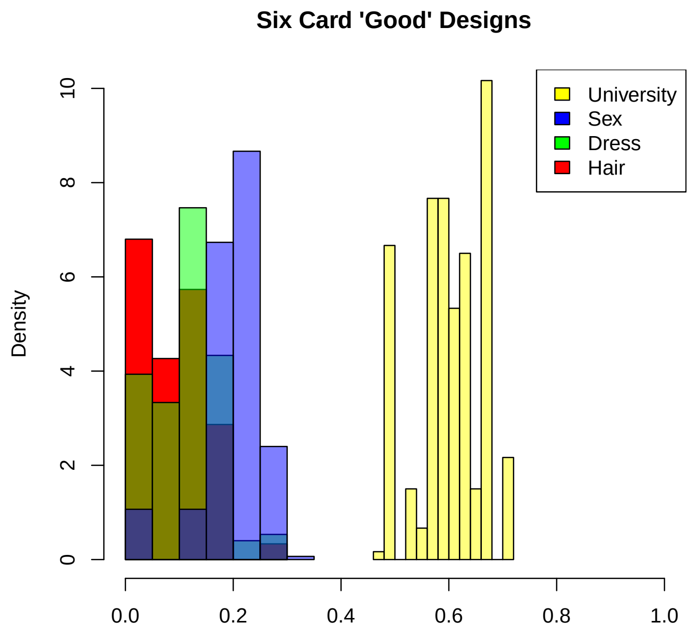
Of course we will need a way to tell the computer how to recognize a good design (it is impractical to check all designs by hand). We can simply ask that the design be non singular, approximately balanced and has no “large” correlations. A number of definitions of large could be used, one obvious one is the maximum value of the cross correlations. Smaller numbers of cards have fewer designs with very low correlations. With eight cards correlations up to .3 have to be accepted to get a reasonable number of designs. For larger numbers of cards this can be reduced to .2 or even .1.

A simple R program can check 1,000,000 designs in a couple of hours. Beware that there are often many fewer designs available. For example, there are only 10,626 designs available if we choose 20 cards from the 24 cards in our full design.

A simple strategy is to take the average values of importance from a number of “good” designs. This will guard against having a single design that gives outlying values.

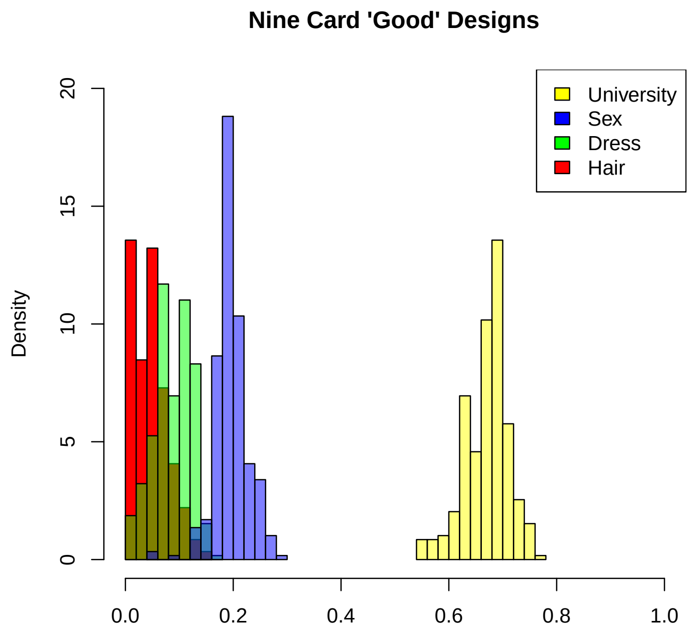
To get some idea of how things might work, I did some simple calculations using the sample experiment First I used `mc.good.design` which looks for “good” designs. I then calculated the importances of the factors for each design.

The first experiment was done with the minimum number of cards possible, 6. For this small a number of cards, no valid designs were found for cross correlations below 0.15. This was changed to 0.28. Even so, 100000 designs had to be checked to get a reasonable number, 300, of valid designs. A histogram of the importances follows.



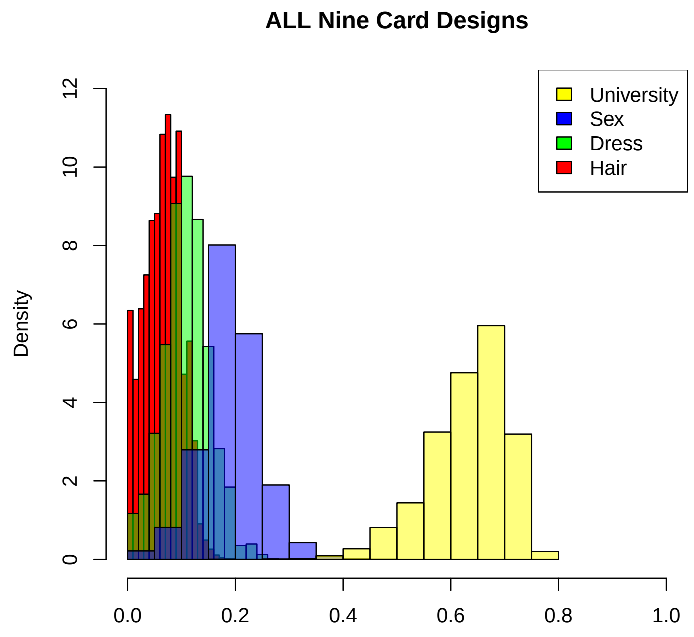
As can be noted, the most important factor “university” is well separated. The other factors are not well distinguished. Nor is it always clear that “hair” is less important than “dress”.

If we increase the number of cards to 9 things start to look a bit better. We can now reduce the maximum cross correlation to 0.15. We need to look at 170000 designs to get 300 valid designs.



The other factors are now separated (especially “sex” which only has a small overlap with the other two factors).

The advantage to using a “good” design can be seen in a simulation where any design that could be used (non singular, at least one of every level) was



used.

While the most important factor "university" is well separated, there is still a little overlap and the other factors are not very well separated. The factor "sex" is seen as the third or fourth most important factor almost 14% of the time, while with the "good" designs "sex" was the third or fourth most important factor about 4% of the time.

1.4.3 Optimization

Instead of looking for a number of "good" designs we can instead look for the "best" design. The standard way to do this is to check the information matrix. Indeed we can show that the variances/co-variances of the β coefficients are proportional to the inverse of the information matrix, $\mathbf{Y} = (\mathbf{X^T X})^{-1}$. One popular criterion, the "D" criterion, is to minimize the determinant of $\mathbf{Y}$ (that is maximize the determinant of the information matrix). You can also use the A criterion, minimize the trace of $\mathbf{Y}$ or the In practice the different criteria tend to produce similar models.

You might not that by optimizing the information matrix you are not insuring a low correlation, well balanced design. True, but most optimal designs are low correlation and well balanced.

Many software packages are capable of producing "optimal" designs according to one of the above criteria (note: the designs chosen may not be truly optimal as the search routines normally used tend to land at local extrema, but in practice this is not a problem) In R, you can use the `AlgDesign` package

You need to choose the number of cases/cards (see below), n. Then ask for this many cards from the full design. Exact syntax ($n = 11$) is

```
> set.seed(123)  #only added for consistency reasons
> des.11 = optFederov(~., experiment, nTrials = 11)$design
> print(des.11)

   University    Sex Dress  Hair
1    Prestige   Male smart  long
3        Good   Male smart  long
4    Prestige Female smart  long
8   Excellent   Male messy  long
11  Excellent Female messy  long
14  Excellent   Male smart short
17  Excellent Female smart short
18       Good Female smart short
19   Prestige   Male messy short
21       Good   Male messy short
22   Prestige Female messy short
```

(Note this will use the "D" criterion) This was how `design.three` was obtained.

1.5 Holdouts

It is traditional with Conjoint Analysis to add a number of extra cards to a design. These cards will be ranked during the survey, but will no be used in the initial analysis. They are used to check the model produced by the Conjoint analysis, either on globally or case by case.

One problem is that there is no agreed on way of using the holdouts. One commonly used method is to use the Conjoint Model to predict the ranking of the holdout cards and then compare this ranking to the ranking obtained from the survey. In the simplest case of two holdouts the question is simply whether the model is right or wrong. If a larger number of holdouts is used the total rankings can be compared (e.g. Kendall's tau) or partial comparisons (e.g. the highest ranked card).

Another problem is that there is no agreed way to use the information. One possible approach is to use only those subjects for which the Conjoint model predicts correctly. Another is to scrap the analysis if the fit is too bad. The question is what to do then. Do you try a different type of analysis? If you suggest to your boss that the (very expensive) data set is useless, then it is you, not the data set, that will be discarded.

I suggest a simple alternate approach. Rather than use the holdout cards to check the analysis, one can create extra "good" designs. So given n cards and m holdouts, we can look for "good" designs from the $(n + m)$ choose n n card designs that can be extracted from the set of $(n + m)$ cards. The importances and utilities can be calculated as averages over all the "good" designs.

Since, under this scheme there is no real distinction between holdout and non holdout cards, one method is to select a set of m cards that maximizes the number of n card good design subsets. A simpler scheme is to obtain one "very good" n card design, either by searching or using an optimization scheme, then adding $m - n$ cards which maximize the number of "good" designs.

1.6 Sample Size

The question of sample size in conjoint analysis is a bit complicated as there are two questions

- How many questions to ask each subject
- How many subjects

These questions are not unrelated, but it is perhaps best to initially treat them as if they were.

Note the information we want from each subject is "How important are each of these factors in making your decision". If we could ask this question directly we would. However, we know that if we ask the question in this form we will not get a very good answer. So, instead, we ask the subject to rank a number of "cards" and attempt to determine the answer from these ranks. One way of doing this (there are others) is to perform a linear fit. If the linear model fit perfectly, then we would only need enough cards for the information matrix to be non-singular; for each variable one less than the number of levels and one for the constant. This is the minimum number of cards we need. Under the perfect fit assumption, there is no noise, and no p-values. We are merely performing an algebraic manipulation.

However, we know that a linear model is not a perfect way of describing how people make decisions. So strictly speaking, using the linear model to answer "How important are each of these factors in making your decision?" does not work! The question is "How well does it work?". In practice, the answer is "well enough". Does increasing the number of cards make the method work better? The answer would seem to be yes. A rule of thumb is to take three more cards than the minimum required.

If we treat the answers to the first question as noiseless, then our second question is straightforward sampling theory. The rule of thumb is that your error is about $\sqrt{\texttt{number of samples}}$. So to get answers correct to 10% use 100 subjects, to get answers correct to 3% use 1000 subjects. (Here we are contemplating such questions as "what percentage of the population prefers factor 1?")

However, neither of these really addresses the question you will have. You are given the average utilities, and the average importances. To what extent can you trust these? The problem is that the uncertainties in these numbers are influenced by a combination of the number of cards used and the number of subjects asked. Furthermore, the way things vary may depend on factors you do not know. For example, if everyone is the same, using more subjects will not reduce the error. On the other hand, if there are many differences, using lots of subjects is necessary, and using more subjects will give better answers (although the model error may be correlated and will not decrease), but in this case the averages (especially average utilities) may be meaningless.

So now you are saying, enough already, how large a sample do I need. The problem is that there is no good single answer, nor any good rule of thumb. Something that is worth doing is to histogram the importances and utilities by subject (Note that `despack$utils` and `despack$imps` are what is needed here.) In this simple example there was only one subject (all subjects were assumed to be identical), however, usually the subjects will not be identical. You can look for evidence that there is more than one importance peak (e.g that some subjects give a factor a different importance.) Similarly for utilities. In particular look for some subjects giving a factor a positive utility, while other subject give the same factor a negative utility.