

DIY MaxDiff

2nd edition

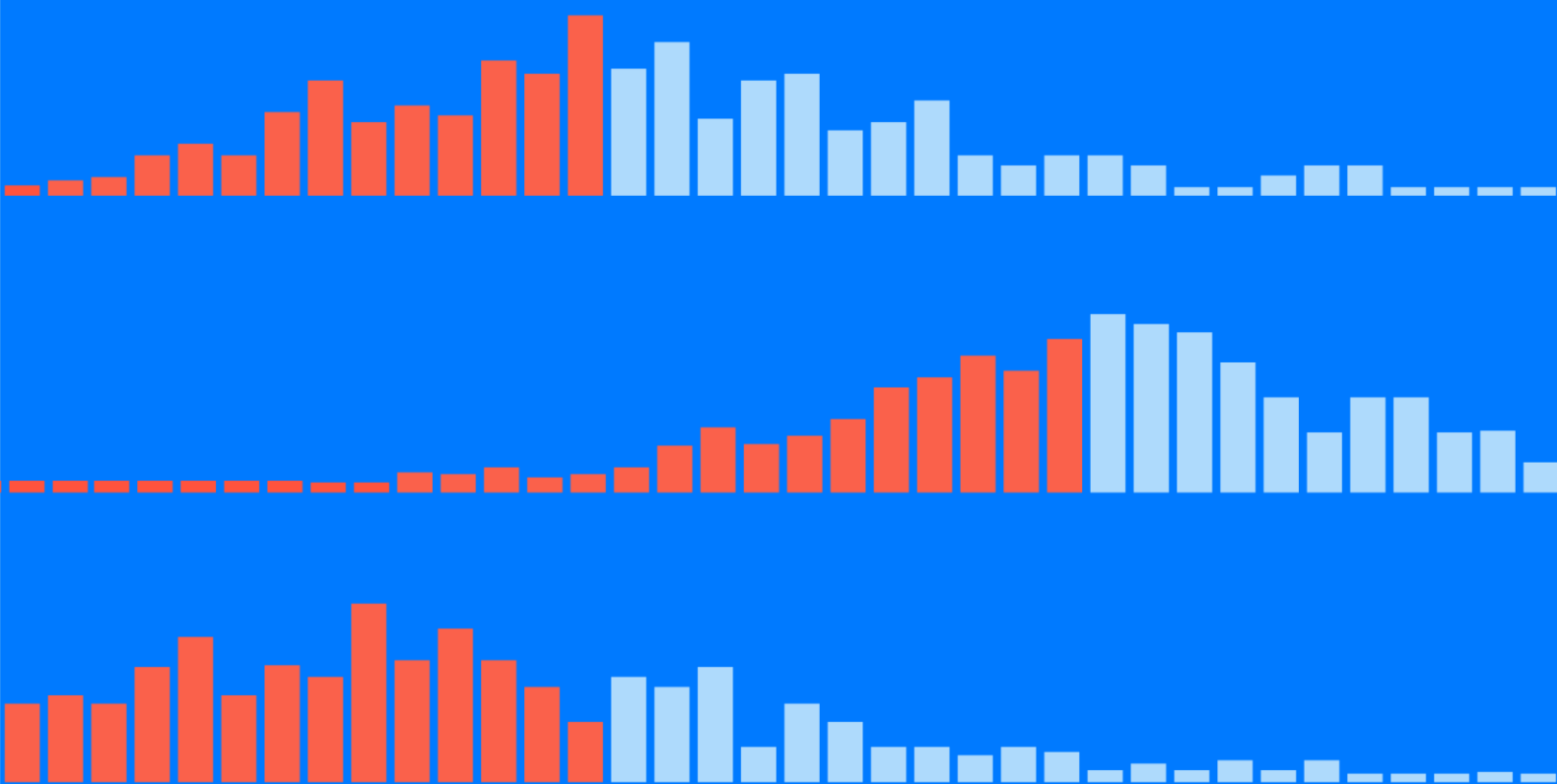


Table of Contents

Who is this eBook for?	2
When to use MaxDiff	3
Creating a list of alternatives to be evaluated	6
Standard experimental designs	10
Designs with many alternatives	22
Prohibitions.....	27
Data collection.....	31
Data files	33
Statistical analysis.....	36
Latent class analysis	48
Hierarchical Bayes	61
Multivariate analyses of coefficients	72
Anchored MaxDiff.....	82

Who is this eBook for?

This eBook is for anybody that wants to do their own MaxDiff studies. It provides a complete guide for conducting your own MaxDiff studies, guiding you through the stages of:

- Creating a list of alternatives
- Creating an experimental design
- Collecting the data
- Statistical analysis
- Reporting

When to use MaxDiff

MaxDiff is a survey research technique for working out relative preferences. What do people like most? Second-most? Etc.

It is useful in situations when simpler techniques – such as asking people to rate things or provide rankings – are considered likely to give poor data.

A MaxDiff study involves presenting a sample of respondents with a series of questions, in which each question contains a list of alternatives and the respondent is asked which alternative they like the most (*best*) and which the least (*worst*). An example is shown below. The list of alternatives changes from question to question.

e-Rewards

69%

Thinking about the type of person you would like to have as the President of the USA, which of these characteristics is **most appealing** to you, and which is **least appealing**?

Screen 8 of 10

Most appealing		Least appealing
<input type="radio"/>	Entertaining	<input type="radio"/>
<input type="radio"/>	Concerned about poverty	<input type="radio"/>
<input type="radio"/>	From a traditional American background	<input checked="" type="radio"/>
<input checked="" type="radio"/>	Good in a crisis	<input type="radio"/>
<input type="radio"/>	Experienced in government	<input type="radio"/>

[Privacy Policy](#)

MaxDiff is used to resolve two practical problems with traditional rating scales:

- Poor discrimination between alternatives, with respondents in surveys often rating multiple alternatives as very important, or 10, on a 10-point scale
- *Yeah-saying biases*, which are a type of *response bias*, whereby some respondents typically give much higher ratings than others

Consider the problem of working out what capabilities people would most like in the President of the United States. Asking people to rate the importance of each of the following characteristics would likely not be so useful. We all want a decent and ethical president. But we also want a president who is healthy. And the President needs to be good in a crisis. We would end up with a whole lot of the capabilities shown in the table on the next page being rated as 10 out of 10 important. Some people may give an average rating of 9, whereas others may give an average rating of 5, just because they differ in terms of how strongly they like to state things. It is for such problems that MaxDiff is ideal.

Decent/ethical	Good in a crisis	Concerned about global warming	Entertaining
Plain-speaking	Experienced in government	Concerned about poverty	Male
Healthy	Focuses on minorities	Has served in the military	From a traditional American background
Successful in business	Understands economics	Multilingual	Christian

The end-point of a MaxDiff study is usually one or both of the following:

- A ranking of alternatives in order of preference. For example, if the study is being used for product-concept testing, the goal is to work out the relative appeal of the concepts.
- An understanding differences between people in terms of their preferences for the alternatives. For example, a study examining preferences for product attributes may be designed as an input to a segmentation exercise, looking to find segments of people with different preferences.

Creating a list of alternatives to be evaluated

The first stage in a MaxDiff study is to identify the list of alternatives to be studied.

Typically, MaxDiff is used to compare either *attributes*, *advertising messages*, *product claims*, *promotional offers*, and *brands*, although in principle it can be used to compare any objects.

It is usually straightforward to come up with a list of brands to be compared. In a recent study where we were interested in the relative preference for Google and Apple, we used the following list of brands:

- Apple
- Google
- Samsung
- Sony
- Microsoft
- Intel
- Dell
- Nokia
- IBM
- Yahoo

The logic was to have a good cross-section of consumer software and hardware brands. Is it a good list? We return to that in the discussion of *context effects*, below.

When conducting studies involving product concepts, the key is to ensure that short descriptions are used, as a MaxDiff question becomes problematic if users cannot easily read and compare the alternatives.

An example of *attributes* is the list of capabilities of a president listed in the previous chapter. More commonly, MaxDiff studies involving attributes focus on preferences for the attributes of a product.

There are some mistakes to avoid when choosing alternatives for a MaxDiff study: having too many alternatives, vaguely-worded alternatives, and context effects.

Too many alternatives

The more alternatives, the worse the quality of the resulting data. With more alternatives you have only two choices: You can ask more questions, which increases fatigue and reduces the quality of the data, or you can collect less data on each alternative, which reduces the reliability of the data. The damage of adding alternatives grows the more alternatives that you have (e.g., the degradation of quality from having 14 over 13 alternatives is greater than that of having 11 over 10). Techniques for dealing with large numbers of alternatives are described in [Designs with many alternatives](#).

Vague wording

Vaguely-worded alternatives are impossible to interpret. For example, if you are evaluating gender as an attribute, it is better to use male or female exclusively. Otherwise, if the results show that ‘Gender’ is important, you will not know which gender was most appealing.

If you ask about attributes without specifying the level of the attribute, such as asking people about the importance of price, the resulting data will be of limited meaning. For some people "price" will just mean “not too expensive”, while others will interpret it as a deep discount. This can partially be improved by nominating a specific price point, such as *Price of \$100*, but even then, the attribute has some ambiguity when it comes to interpretation time, as one respondent may have a reference price of \$110 and another a reference price of \$90. A better way to address price may be in terms of discounts (e.g., \$10 discount). However, the issue of “*Compared to what?*” remains, which is why choice modeling (rather than MaxDiff) is the methodology better suited to understanding trade-offs between attributes with different levels.

Context effects

A *context effect* in MaxDiff occurs when we think that level of preference that a person feels towards an alternative is conditional upon the alternatives it is compared against. In the two studies described in this eBook, there is a reasonable likelihood of context effects. In the case of the technology study, if a user sees a question comparing *Samsung*, *Nokia*, *Google*, *Sony*, and *Apple*, there is a good chance they will be thinking about *Google* in terms of its hardware. By contrast, if *Google* is shown against *Yahoo*, it will more likely be thought of in terms of its search engine. Similarly, in the study of presidential capabilities, perhaps *Successful in business* and *Understands economics* are not in the same question, each will become a bit more important, as people will treat them as surrogates for each other.

In practice, context effects are something to be thought about and lived with. It is often impractical to avoid alternatives simply because of the possibility of context effects. Moreover, experimental designs typically present each alternative with each other alternative, so context effects can be averaged out. This is discussed further in the next chapter.

Standard experimental designs

This chapter describes the essentials for creating an experimental design for a MaxDiff study.

Sometimes more advanced designs are required; these are described in the next chapter.

As discussed, MaxDiff involves a series of questions – typically, six or more. Each of the questions has the same basic structure, as shown below, and each question shows the respondent a subset of the list of alternatives (e.g., five). People are asked to indicate which option they prefer the most, and which they prefer the least. Each question is identical in structure but shows a different list of alternatives. The *experimental design* is the term for the instructions that dictate which alternatives to show in each question.



Which of these companies would you be most likely to recommend to a friend or a colleague and which would you be least likely to recommend to a friend or colleague?

Most likely to		Least likely to
<input type="radio"/>	Intel	<input type="radio"/>
<input type="radio"/>	IBM	<input checked="" type="radio"/>
<input checked="" type="radio"/>	Google	<input type="radio"/>
<input type="radio"/>	Sony	<input type="radio"/>
<input type="radio"/>	Dell	<input type="radio"/>

What an experimental design looks like

Designs are either *single version designs* or *multiple version designs*.

Single version designs

The most straightforward designs involve showing each person the same questions. The table below shows an experimental design for the technology study. This study had ten alternatives. This design involved asking people six questions, where each question had five alternatives (options). The design dictated that the first question should show brands 1, 3, 5, 6, and 10. The second shows brands 1, 5, 7, 8, and 9. And so on.

	↕ Option 1 ↴	↕ Option 2 ↴	↕ Option 3 ↴	↕ Option 4 ↴	↕ Option 5 ↴
Question 1	1.0	3.0	5.0	6.0	10.0
Question 2	1.0	5.0	7.0	8.0	9.0
Question 3	1.0	2.0	4.0	9.0	10.0
Question 4	2.0	3.0	6.0	7.0	9.0
Question 5	2.0	4.0	5.0	6.0	8.0
Question 6	3.0	4.0	7.0	8.0	10.0

Multiple version designs

Multiple version designs are designs where some or all respondents are asked different questions. Two additional columns are added at the beginning of the design. The first shows the version and the second shows the question number within the version. The remaining columns are the same as in a standard design. The table below shows a design with two versions (ten or more is the norm if using multiple versions). Note that the first six rows contain the single-version design shown above, and the next six show a new design, created by randomly swapping around aspects of the original design.

	Version	Question	Option 1	Option 2	Option 3	Option 4	Option 5
1	1.0	1.0	1.0	3.0	5.0	6.0	10.0
2	1.0	2.0	1.0	5.0	7.0	8.0	9.0
3	1.0	3.0	1.0	2.0	4.0	9.0	10.0
4	1.0	4.0	2.0	3.0	6.0	7.0	9.0
5	1.0	5.0	2.0	4.0	5.0	6.0	8.0
6	1.0	6.0	3.0	4.0	7.0	8.0	10.0
7	2.0	1.0	3.0	5.0	6.0	7.0	9.0
8	2.0	2.0	1.0	4.0	6.0	7.0	8.0
9	2.0	3.0	2.0	4.0	5.0	7.0	10.0
10	2.0	4.0	1.0	2.0	3.0	4.0	9.0
11	2.0	5.0	2.0	6.0	8.0	9.0	10.0
12	2.0	6.0	1.0	3.0	5.0	8.0	10.0

Multiple-version designs come about for two different reasons. Sometimes they are used because researchers wish to have different designs to deal with context effects, which are discussed later in this chapter. Other times they are used to deal with large numbers of alternatives, which are discussed in the chapter [Designs with many alternatives](#).

Creating an experimental design for the technology case study

The previous chapter introduced a MaxDiff study looking at ten technology brands. Here we show how to create an experimental design for it. This is done using Displayr/Q, but the same logic can be applied using any other software.

In this example, there are ten alternatives (brands). As the brands are relatively easy to evaluate, five alternatives were shown per question. It is always nice to have a small number of questions, so we start with a small number of questions – four – and a single version. It is possible to create a design with these inputs, but not a good design: Displayr returned a list of warnings, which are reproduced below.

You have specified 4 questions. It is sometimes recommended that $\text{number.questions} \geq 3 * \text{number.alternatives} / \text{alternatives.per.question}$ (i.e., that you should have at least 6 questions).	×
One or more of the alternatives appears only 2 time(s). A common recommendation is that each alternative should appear 3 times. You can review the frequencies by viewing the detailed outputs.	×
The largest binary absolute correlation is 1. You should consider having more questions. You can review the binary correlations by viewing the detailed outputs.	×
The absolute value of the correlations varies from 0 to 1. This may not be a problem, but ideally the absolute value of the correlations should be constant (this is not always possible). Consider increasing the number of questions.	×
Some alternatives never appear together. You can review the pairwise frequencies by viewing the detailed outputs.	×
Some alternatives only ever appear together. You can review the pairwise frequencies by viewing the detailed outputs.	×

The first of the warnings tells us that we have too few questions and suggests we should have six. The other warnings are specific problems with the design that are consequences of having too few questions (or too many alternatives – these are two sides of the same coin). If we increase the number of questions to five, we still get lots of warnings. At six, the warnings go away.

More detail about how to create the designs – as well as the characteristics of a good design – are discussed throughout the rest of this chapter.

Inputs in creating a MaxDiff experimental design

The number of alternatives

Typically, this is just the number of alternatives that you want to research (see the previous chapter). For example, in the technology study described in the previous chapter, this was ten, and in the study of presidential capabilities there were 16 alternatives.

The more alternatives, the worse the analysis in terms of the reliability of the results.

Alternatives per question

This is the number of alternatives that a respondent is asked to compare in a question. Nearly all studies use four to six alternatives. It is difficult to envisage a situation where more than seven would be appropriate. Considerations when choosing the number of alternatives include:

- If you have fewer than four alternatives it is not really a MaxDiff study, as three alternatives is a complete ranking, and two alternatives prevents you from asking the “worst” question.

- Cognitive difficulty: In the example question shown above this is five. With studies where the alternatives are lengthy or difficult to understand it is often better to have four alternatives per question. Where the alternatives are very easy to understand, six may be appropriate.
- The number of alternatives in each question should be no more than half the number of alternatives in the entire study. Otherwise, it becomes difficult to create a good experimental design, and a straightforward ranking exercise is likely to be sufficient.

Number of questions

A rule of thumb provided by Sawtooth Software states that the ideal number of questions is at least:

$$3 * \text{Alternatives} / \text{Alternatives per question}$$

which leads to each alternative being shown to each respondent at least three times. This would suggest that in the technology study with its ten alternatives, we should have used at least $3 * 10 / 5 =$ six questions.

There are two conflicting factors to trade off when setting the number of questions. The more questions, the more respondent fatigue, and the worse your data becomes. The fewer questions, the less data, and the harder it is to work out the relative appeal of alternatives that have a similar level of overall appeal. We return to this topic in the discussion of checking designs, below.

Number of versions

Should a MaxDiff study have multiple versions of the experimental design with different respondents seeing different versions, or a single experimental design seen by all respondents? The answer to this depends on several factors:

- If you use modern analysis techniques, such as latent class analysis and hierarchical Bayes (both described in later chapters), then a single version is usually okay. If you are using *counting analysis*, also described in a later chapter, it is generally advisable to have lots of versions (e.g., one for every respondent), although even then *counting analysis* is not an appropriate way to analyze MaxDiff data.
- Whether *context effects* are likely to be problematic: With a single-version MaxDiff study, the design ensures that context effects are averaged to an extent. If we ask multiple versions, this averaging will be more comprehensive. While some people interpret this as a strong argument for context effects, it is not quite as strong as it appears. Context effects do not cancel out. If you have context effects, having multiple versions just means that whatever bias they add is estimated consistently. A simpler approach than using multiple versions is to randomize, where this randomization is automatically performed by the data collection software. In particular:

- Randomization of question ordering: If each respondent sees the questions in a different order, then the context effects are likely to be averaged in much the same way as will be achieved by using multiple versions.
- Randomizing the order of alternatives: This can be done either on a question-by-question basis, or between respondents.
- Whether the goal of the study is to compare people, or rank the alternatives: If we believe that *context effects* exist, using different versions will cause them to be averaged across respondents. If our goal is segmentation, then this is undesirable, as it will cause results to differ by people due to differences in the versions rather than between people.

Sawtooth Software suggests that when utilizing multiple versions, as few as ten are sufficient to minimize order and context effects. However, if administering the study online, it might be better to have one per respondent or, if that is impractical due to how the study is administered, 100 different versions. While there is clearly a diminishing marginal utility to adding extra designs, the cost is zero.

Repeats

When software creates an experimental design it usually starts by generating a design using randomization and then seeks to improve this design using various algorithms. Occasionally the initial randomly-generated design has some quirk that makes it hard for the algorithms to improve it. Most experimental design packages have an option to repeat the process multiple times to see if it can be improved. Generally, the “number of repeats” setting in the software should be left at its default level, unless you have identified a problem with your design.

Increasing the number of repeats helps only occasionally. Problems with experimental designs are usually best addressed by increasing the number of questions.

Software

MaxDiff experimental design in Q

Experimental designs in Q are generated by selecting **Create > Marketing > MaxDiff > Experimental Design**. The options correspond to the categories in the previous section.

MaxDiff experimental design in Displayr

Experimental designs in Displayr are generated by selecting **Anything > Advanced Analysis > MaxDiff > Experimental Design**. The options correspond to the categories in the previous section.

Checking the design

In an ideal world, a MaxDiff experimental design has the following characteristics, where each alternative appears:

1. At least three times.
2. The same number of times.
3. With each other alternative the same number of times (e.g., each alternative appears with each other alternative twice).

Due to a combination of math and a desire to avoid respondent fatigue, few MaxDiff experimental designs satisfy all three requirements (the last one is particularly tough). Earlier in the chapter we showed a single version design for ten alternatives, five alternatives per question, and six questions. The output shown to the right is for a modification of this design, where the number of alternatives per question is reduced from five to four, which causes the design to become poor. How can we see it is poor?

```
$binary.design
      Alternative
Question Alternative 1 2 3 4 5 6 7 8 9 10
Question 1          1 0 0 1 1 0 0 0 0 1
Question 2          0 1 0 0 1 0 0 1 1 0
Question 3          0 0 1 0 0 0 1 1 0 1
Question 4          1 1 0 0 0 1 0 1 0 0
Question 5          0 0 0 1 0 1 1 0 1 0
Question 6          0 1 1 1 0 0 0 0 1 0
```

```
$design
      Alternative
Question 1 2 3 4
      1 1 4 5 10
      2 2 5 8 9
      3 3 7 8 10
      4 1 2 6 8
      5 4 6 7 9
      6 2 3 4 9
```

First, you will see various warnings if you create the design in Displayr, Q, or R, as shown at the beginning of the chapter. The warnings will be pointing out the following problems.

```
$frequencies
 1  2  3  4  5  6  7  8  9 10
2  3  2  3  2  2  2  3  3  2
```

The first problem is that some of the alternatives are shown two times (`frequencies`). As discussed earlier, we generally want each alternative to be seen three times.

```
$pairwise.frequencies
      1 2 3 4 5 6 7 8 9 10
1  2 1 0 1 1 1 0 1 0 1
2  1 3 1 1 1 1 0 2 2 0
3  0 1 2 1 0 0 1 1 1 1
4  1 1 1 3 1 1 1 0 2 1
5  1 1 0 1 2 0 0 1 1 1
6  1 1 0 1 0 2 1 1 1 0
7  0 0 1 1 0 1 2 1 1 1
8  1 2 1 0 1 1 1 3 1 1
9  0 2 1 2 1 1 1 1 3 0
10 1 0 1 1 1 0 1 1 0 2
```

A second problem is that the design is not *balanced*, as some alternatives have been seen

three times and others two times. In the early days of MaxDiff it was common to analyze the data using counting analysis, and this technique assumes that the data is balanced. When using more modern techniques it is desirable but not essential to have balance.

The `pairwise.frequencies` table shows us how often each alternative appears in questions with each other alternative (the main diagonal contains the `frequencies`). We can see that many pairs of alternatives never appear together (e.g., 1 and 3, 1 and 7). Ideally, each alternative will appear the same number of times with each other alternative. Such designs are sometimes also referred to as being *balanced* (see the section on jargon at the end of the chapter).

As the whole purpose of a MaxDiff study is to understand the relative appeal of different options, it is not ideal that we have pairs of alternatives that the respondent never explicitly encounters. It should be emphasized that while it is not ideal to have two alternatives never appearing together, neither is it a catastrophe. If using latent class analysis or hierarchical Bayes, the analysis itself is often able to compensate for weaknesses in the experimental design, although it would be foolhardy to rely on this without very carefully checking the design using a small sample (discussed below).

	1	2	3	4	5	6	7	8	9	10
1	1.00	0.00	-0.50	0.00	0.25	0.25	-0.50	0.00	-0.71	0.25
2	0.00	1.00	0.00	-0.33	0.00	0.00	-0.71	0.33	0.33	-0.71
3	-0.50	0.00	1.00	0.00	-0.50	-0.50	0.25	0.00	0.00	0.25
4	0.00	-0.33	0.00	1.00	0.00	0.00	0.00	-1.00	0.33	0.00
5	0.25	0.00	-0.50	0.00	1.00	-0.50	-0.50	0.00	0.00	0.25
6	0.25	0.00	-0.50	0.00	-0.50	1.00	0.25	0.00	0.00	-0.50
7	-0.50	-0.71	0.25	0.00	-0.50	0.25	1.00	0.00	0.00	0.25
8	0.00	0.33	0.00	-1.00	0.00	0.00	0.00	1.00	-0.33	0.00
9	-0.71	0.33	0.00	0.33	0.00	0.00	0.00	-0.33	1.00	-0.71
10	0.25	-0.71	0.25	0.00	0.25	-0.50	0.25	0.00	-0.71	1.00

The table above shows the *binary correlations*. This *correlation matrix* shows the correlations between each of the columns of the experimental design (i.e., of the `binary.design` shown on the previous page). Looking at row 4 and column 8 we see a problem. Alternative 4 and 8 are perfectly negatively correlated. That is, whenever alternative 4 appears in the design alternative 8 does not appear, and whenever 8 appears, 4 does not appear. One of the useful things about MaxDiff is that it can sometimes still work even with such a flaw in the experimental design (although, again, it is a dangerous design that needs to be carefully checked).

Another concerning aspect of this design is the large range in the correlations. In most other areas where experimental designs are used, the ideal design results in a correlation of 0 between all the variables. MaxDiff designs differ from this, as, on average, there will always be a negative correlation between the variables. However, the basic idea is the same: We strive for designs where the correlations are as close to 0 as possible. Correlations in the range of -0.5 and 0.5 are usually no cause for concern in MaxDiff studies.

Checking using a small sample

A few things can ruin a MaxDiff study. One is a poor design which cannot be used to estimate relative preference. Another is errors in how the data is captured. A third is a poorly constructed set of alternatives that contain one or more alternatives that are preferred or hated by everybody due just to lazy wording. Each of these problems can be detected by checking the design using a small sample.

Good practice is to get a data file after about 10% of the data has been collected. This will be either 10% of the final sample, or just a pilot study. When working with a team that has limited experience with MaxDiff, it is also a good idea to complete a few questions yourself, remember your responses, and check that they are recorded correctly in the data file. The data file is checked as follows:

1. Create tables to check the balance of the design (i.e., how many times each alternative has been shown). If the design has been programmed incorrectly this will usually be clear from these tables.
2. Look at the raw data and check that the design in the raw data is as expected.

3. Estimate a latent class analysis model (discussed in a later chapter). This is the model most likely to detect a fundamental problem of some kind. If a subset of the respondents received experimental designs that were flawed in some way, this can show up as an error in the latent class analysis, whereas hierarchical Bayes models are a little less likely to detect such problems due to the way they pool data between respondents (i.e., if one respondent has an experimental design that is very poor, that respondent's coefficients are assigned based on what other respondents have answered).
4. Form preliminary conclusions. That is, check that the model is telling you what you need to know for the project to be a success. Yes, the sampling error will be relatively high, but key conclusions should still be making sense at this time.

This is the gold standard for checking a design. You can conduct this process along with all the other approaches. If you are brave, you can do just this step and skip the earlier approaches; but skipping testing on a small sample is foolhardy, as reviewing the results from a small sample of real respondents checks things much more thoroughly than the other approaches.

One last comment on checking designs: Experienced researchers check more than novice researchers – they have learned from pain. If this is your first MaxDiff experiment, make sure you check everything very carefully.

Checking designs with multiple versions

When you set the number of versions to more than one, this will *not* change any of the warnings described in the previous section. All these warnings relate to the quality of the design for an individual person. Increasing the number of versions improves the design for estimating results for the total sample, but this does not mean the designs change in any way for individual respondents. So, if you are doing any analysis at the respondent level, changing the number of versions does not help in any way.

Additional detailed outputs are provided when using multiple versions, which show the properties of the whole design. These show the binary correlations across the entire design, and the pairwise frequencies. Their interpretation is as described above, except that it relates to the entire design, rather than to the design of each version.

Software instructions

In Q and Displayr, check the **Detailed outputs** option to see all the outputs.

Fixing a poor design

The first thing to do when you have a poor design is to increase the setting for the number of *repeats*. Start by setting it to ten. Then, if you have patience, try 100, and then bigger numbers. This works only occasionally, and when it does, it is a good outcome.

If changing the number of repeats does not work, you need to change something else. Reducing the number of alternatives and/or increasing the number of questions is usually effective.

Jargon

The standard experimental designs are created using *incomplete block designs*, where *block* refers to the questions, *incomplete* refers to an incomplete set of alternatives appearing in each question, *alternatives* are referred to as *treatments*, and the *binary design* is referred to as an *incidence matrix*.

A *balanced incomplete block design* is one where each alternative appears the same number of times with each other alternative.

Designs with many alternatives

There are several strategies for dealing with large numbers of alternatives:

- Sparse MaxDiff / Random question allocation
- Bridging designs
- Express MaxDiff / Nested designs
- Hybrid MaxDiff
- Constructed MaxDiff / Relevant Items MaxDiff
- Bandit MaxDiff

With a standard design, as you increase the number of alternatives you need to increase the number of questions. This can lead to situations where there are too many questions for it to be practical to get respondents to answer all the questions. This section lists strategies that can be used in such situations.

The issue of how many alternatives is too many has no correct answer. The more alternatives, the lower the quality of the data. There is no magical tipping point. This is really the same issue as the one of how big grid questions can be in surveys, and how long a questionnaire should be. Some experienced MaxDiff researchers use more than 100 alternatives in studies. Others never use more than 20.

Before explaining the design strategies, a note of caution: If each respondent does not see each alternative three or more times, we may not get reliable data at the respondent level (e.g., segmentation may become unreliable). A design with many alternatives is typically appropriate only when the focus is on ranking the appeal of the alternatives.

Sparse MaxDiff / Random question allocation

The simplest and often best strategy is to create a large design and then randomly allocate respondents to different questions. For example, if a design has 20 questions, each respondent may be randomly allocated ten.

While this is a simple strategy, it is also a good strategy in that it is both simple and tends to produce good quality designs.

Bridging designs

This strategy splits the alternatives into overlapping sets, creating a separate experimental design for each. For example, one design may be for alternatives 1 through 10, and another for alternatives 8 through 17. Or you can have one design testing, say, {1...10}, {11...20}, {21...30}, {31...40}, {1,5,9,13,17,21,25,29,33, 37}, etc.

This strategy makes most sense when there are natural groupings of alternatives. In general, it is better to allocate questions randomly instead, as bridging works well only if the alternatives that are common cover a good range from low to high appeal.

Express MaxDiff / Nested designs

This approach involves deciding to show each respondent a subset of the alternatives, where an experimental design is used to work out which respondents see which alternatives and another design is used to show which alternatives appear in which question.

Either randomly assign subsets of alternatives to respondents or use an incomplete block design to allocate alternatives to respondents. Then use an incomplete block design for each set of alternatives. For example, if there are 100 alternatives, you could:

- Create a design with 100 alternatives, 20 blocks (questions) and ten alternatives per block (Design A).
- Create a second design with ten alternatives, six questions and five alternatives per block (Design B).
- Randomly allocate each respondent to one of the 20 blocks in Design A (i.e., so that the respondent only sees the ten alternatives in this block).
- Use Design B to create the questions to be shown to the respondent (where the alternatives used are dictated by the respondent's block in Design A).

Hybrid MaxDiff

Respondents are asked to provide ratings of a larger number of alternatives (e.g., ratings of how much they like them on a scale of 0 to 10), and then MaxDiff is used for a subset of the alternatives.

There are three options for selecting the subset of alternatives

- Randomly allocate different brands to the MaxDiff study.
- Have a fixed list of brands that are used in the MaxDiff study (e.g., the brands that are most interesting to the users of the research), or
- Select the alternatives for each respondent that they have the highest answer to.

If using random allocation, this can be analysed in the same way as any MaxDiff experiment and the ratings ignored. However, a better solution, which is also the required solution for the other two approaches¹ is to set it up as if an anchored MaxDiff (see [Setting up the analysis of anchored MaxDiff in Q and Displayr](#)):

- Set up the MaxDiff as a **Ranking Variable Set** in Displayr or **Question** in Q (see the instructions in https://docs.displayr.com/wiki/Setting_Up_a_MaxDiff_Experiment_as_a_Ranking).
- Treat the importance data as ranking data.

It is usually a bad idea to choose which brands to show which respondents based on their ratings. The reason that this is a bad idea is that the resulting data contains something called *endogeneity*, which makes valid statistical analysis very difficult (as the residuals of the models cease to be independent). The analysis of this data is described in the chapter on anchored MaxDiff.

¹ If the other two approaches are analyzed with a standard model, such as the traditional Hierarchical Bayes model, the model will suffer from an endogeneity bias.

Constructed MaxDiff / Relevant Items MaxDiff

This is the same idea as hybrid MaxDiff, except that options that are irrelevant to each respondent are also excluded.²

Bandit MaxDiff

For the initial respondents this works like Sparse MaxDiff / Random question allocation, but once it becomes clear which alternatives are most popular, only these are shown to subsequent respondents.

This approach is useful where the goal is to identify the most preferred alternatives and preferences between them, but it does not collect data that permits conclusions about:

- Segmentation
- Preferences among the less popular alternatives

Our products do not support Bandit MaxDiff.

² Bahna, Eric and Christopher Chapman (2018), "Constructed, Augmented MaxDiff," 2018 Sawtooth Software Conference, Provo, UT. Accessed at:
<https://www.sawtoothsoftware.com/download/techpap/2018Proceedings.pdf>

Prohibitions

In a MaxDiff study, prohibitions are rules regarding which alternatives cannot be shown with which other alternatives.

A *prohibition* in a MaxDiff study is a rule regarding specific sets of alternatives that should not appear in the same question. For example, in a product-testing study there may be two very similar versions of a concept. A goal of the study may be to see which is preferred, but it may be believed that the study will be undermined if both alternatives appear in the same question. Or, alternatives may represent attributes, where some relate to difference levels (e.g., Price \$4, Price \$5, or Price \$6).

Below we list some strategies for addressing prohibitions, but prior to doing so we emphasize that prohibitions in MaxDiff are often a poor idea. One problem is that they tend to lead to bloating of the number of alternatives, as often prohibitions are wanted to address the use of very similar alternatives. As mentioned earlier, the fewer the alternatives the better, but if you end up having multiple levels for each attribute, respondent fatigue will grow. A second problem with prohibitions is that they always either result in less reliable estimates of preference or increase context effects. This is discussed in more detail with the design strategies below. Often prohibitions are wanted when the real solution is to use choice modeling instead.

Randomizing across respondents.

For the situation where there are two alternatives that should not be seen together, the rigorous approach is to randomize across respondents, so that no respondent sees both alternatives. The process for generating the design is:

- Generate a standard experimental design, where you include only one of the alternatives to be substituted.
- Randomly swap one of the alternatives. For example, if you have generated a design for ten alternatives, for a random selection of respondents, replace all the alternative 10s in the design with 11s. Alternatively, the replacement can be done sequentially (e.g., changing every second respondent's design).

This strategy has the disadvantage that you end up with a smaller sample size for each of the alternatives that you do not want to appear together. Another method is to allow people to see both alternatives. However, this runs the risk of causing context effects. For example, if you test \$4 and \$5, whether somebody sees \$4 first may change how they feel when they see \$5, due to the *anchoring heuristic*.

Random swapping of alternatives

Wherever a prohibited combination appears, manipulate the design in such a way as to remove the prohibition by replacing one of the alternatives that conflicts with the prohibition with another (and, if possible, doing a reverse substitution elsewhere, so as to limit the reduction of balance).

Creating designs using randomization

Create sets by randomly selecting alternatives (e.g., using R's `sample` function) and discard any sets that contain the prohibited features. Note that this approach is sensible only if different respondents see different sets of alternatives.

Sawtooth Software

Sawtooth Software's MaxDiff experimental design software has options for generating experimental designs with prohibitions regarding alternatives that should not appear together. These designs will, in a statistical sense, outperform the other strategies in this section, as they simultaneously trade off the considerations in the earlier chapter about good designs (e.g., balance) while also addressing the prohibitions.

While this may sound appealing, a disadvantage is that if, say, you have tested three price points, such as \$3, \$4, and \$5, a price alternative appears in the design up to three times as often as other attributes. In the related field of choice modeling there is a lot of evidence that such things can

increase the importance of attributes. Furthermore, as mentioned earlier, context effects are likely a bigger issue.

If you do find yourself in a situation where you feel you need software that has this functionality, please contact us and we will consider adding the feature.

Data collection

This is the easy step. You need:

- Include the MaxDiff questions as a part of a longer questionnaire.
- Use software the supports MaxDiff properly.
- Check.

Include MaxDiff in a longer questionnaire

Typically, MaxDiff questions are asked as a part of longer questionnaires. It's can be helpful to make sure you have other questions in the study which can be used to cross-check the results of the MaxDiff. For example, if doing a MaxDiff of the cell phone market, you would collect data on current cell phone plans and carriers, and would expect that the MaxDiff results would be correlated with this other data.

Survey software that supports MaxDiff

Not all survey software supports MaxDiff. There are two things that are key:

- Support for MaxDiff style questions. Typically, this will either be
 - Easy solution to versioning.
 - A survey platform that supports MaxDiff style questions.

Most likely to		Least likely to
<input type="radio"/>	Intel	<input type="radio"/>
<input type="radio"/>	IBM	<input checked="" type="radio"/>
<input checked="" type="radio"/>	Google	<input type="radio"/>
<input type="radio"/>	Sony	<input type="radio"/>
<input type="radio"/>	Dell	<input type="radio"/>

- We also recommend that you use software that will allow you to:
 - Have a code list of all the alternatives and then use the design to filter these by respondent (rather than using piping, which can be error prone and hard to analyze).
 - Collect some other profiling data (e.g., age, gender, etc.)
 - Check, before you program the questionnaire, that the data is going to come out in a format that is analyzable (see the next chapter)

Data files

Two files are needed for MaxDiff analysis:

1. The data file
2. The experimental design file

Perhaps the most painful part of a MaxDiff study is dealing with data files. The requirements are simple, but the difficulty is that the data files provided by some data collection software platforms have not been created with much thought about how the data needs to be used.

Qualtrics

If you are using data from Qualtrics, both Q and Displayr have special-purpose tools for reading their MaxDiff data, and these are automatically applied by selecting the **Qualtrics** option when importing the data.

This results in MaxDiff data automatically being set up as an **Experiment** question/variable set.

Alchemer (formerly Survey Gizmo)

If using data from Survey Gizmo, MaxDiff can be set up in Q using **Automate > Browse Online Library > Marketing > MaxDiff > Convert Alchemer (Survey Gizmo) MaxDiff Data for Analysis**, and in Displayr using **Anything > Advanced Analysis > MaxDiff > Convert Alchemer (Survey Gizmo) MaxDiff Data for Analysis**.

All other data formats

Experimental design files

The experimental design file is typically a CSV file or an Excel file containing a single sheet, where the design is as described in [What an experimental design looks like](#).

Data files

There are two aspects to getting an appropriate data file. The simple bit is getting a data file. The most useful file format is normally one specifically developed for survey data, such as an SPSS data file (file extension of `*.sav`). Simple file formats like CSV files and Excel files can be used but will typically add a lot of pain (why is discussed below).

A good data file for a MaxDiff study will contain:

- One row of data for each respondent
- All the other data for the study in the same file (e.g., demographics)
- One variable indicating the version seen by each respondent (if multiple versions were used)
- Two variables for each question, where one indicates the alternative chosen as *best* and the other the *worst* choice
- Labels in the variables which make it easy to work out what was chosen. That is, a poor variable is one that says that option 4 was chosen, as it is not clear if this means that the 4th option in the question was chosen, or the 4th alternative. CSV and Excel files often are poor for this, as they do not store *metadata*.
- Randomization removed. That is, if randomization was used to change the order of options, the data file should look as if no randomization was used.

Statistical analysis

- The analysis of MaxDiff data consists of two stages: *statistical analysis* and *reporting*.
- This chapter provides an overview of what statistical analysis needs to achieve with MaxDiff. The next two chapters describe the two key techniques: latent class analysis and hierarchical Bayes.

“Keep it simple, stupid!” is good advice in many areas of life, but not when it comes to analyzing MaxDiff data, where simple = wrong – and wrong in not merely a pedantic way. The case study we will use in this chapter shows how simple methods can lead to massively wrong conclusions.

Case study

In this chapter we will work our way through a MaxDiff data set based on 302 respondents asking about the ten technology brands listed in earlier chapters. The data file is here: https://wiki.q-researchsoftware.com/images/f/f1/Technology_2017.sav. If you are trying to reproduce all the outputs in this eBook, please note that this study was also conducted in 2012, and some of the outputs are from that study (in particular, **Anchored MaxDiff**). The design is https://wiki.q-researchsoftware.com/images/7/78/Technology_MaxDiff_Design.csv. This design is different from the one described in the earlier chapter on experimental design.

Counting the best scores

The simplest way to analyze MaxDiff data is to count how many people selected each alternative as being most preferred. The first column of the table to the right shows the number of people who choose each brand as best. This analysis shows that Apple is, by a long way, clearly the best brand.

This analysis ignores the data on which alternative was least preferred in each question. We should look at that. It shows us something interesting. While Apple is clearly the

	Best	Worst
Apple	464	155
Google	348	40
Samsung	333	103
Sony	227	69
Microsoft	187	87
Dell	82	255
Nokia	64	282
Intel	48	176
IBM	32	314
Yahoo	27	331

most popular, it has its fair share of detractors. So, focusing just on its best scores does not tell the true story.

The next table shows the differences. It now shows that Apple and Google are almost tied in preference. But we know from looking at the best scores that this is not correct.

What is going on here? First, Apple is the most popular brand. This last table is just misleading. Second, and less obviously, the reason that the last table tells us a different story is that Apple is a divisive brand, with lots of adherents and a fair number of detractors. This means that we need to be focused on measuring preferences at the respondent level and grouping similar respondents (i.e., segmentation). As we will soon see, there is a third problem lurking in this simplistic analysis.

	Best - Worst
Apple	309
Google	308
Samsung	230
Sony	158
Microsoft	100
Dell	-173
Nokia	-218
Intel	-128
IBM	-282
Yahoo	-304

Looking at best and worst scores by respondent

To understand data at the respondent level we will start by looking at the experimental design and responses for a single person. The table below shows the MaxDiff experimental design used when collecting the data. The choices of the first respondent in the data set are shown by color. Blue shows which alternative was chosen as best, red as worst. The question that we are trying to answer is, what is the respondent's rank ordering of preference between the ten tech brands?

Questions	Alternatives					Choices
	1	2	3	4	5	
1	Apple	Microsoft	IBM	Google	Nokia	Best
2	Apple	Sony	Dell	Yahoo	Nokia	Worst
3	Microsoft	Intel	Samsung	Sony	Nokia	
4	IBM	Google	Intel	Sony	Dell	
5	Microsoft	Google	Samsung	Dell	Yahoo	
6	Apple	IBM	Intel	Samsung	Yahoo	

The simplest solution is to count the number of times each option is chosen, giving a score of 1 for each time it is chosen as best and -1 for each time it is chosen as worst. This leads to the following scores, and rank ordering, of the brands:

Microsoft 3 > Google 1 ≈ Samsung 1 ≈ Dell 1 > Apple ≈ Intel ≈ Sony > Yahoo -1 > Nokia -2 > IBM -3

This approach is very simple, and far from scientific. Look at Yahoo. Yes, it was chosen as worst once, and our *counting analysis* suggests it is the third worst brand, less appealing to the respondent than each of Apple, Intel, and Sony. However, look more carefully at Question 5. Yahoo has been compared with Microsoft, Google, Samsung, and Dell. These are the brands that the respondent chose as most preferred in the experiment, and thus the data suggests that they are all better than Apple, Intel, and Sony. That is, there is no evidence that Yahoo is worse than Apple, Intel, and Sony. The counting analysis is simple – but wrong.

A more rigorous analysis

We make the analysis more rigorous by considering which alternative was compared with which others. This makes a difference because not all combinations of alternatives can be tested, as it would lead to enormous fatigue. We have already concluded that *Yahoo* is no different from Apple, Intel, and Sony, which leads to:

Microsoft > Google ≈ Samsung ≈ Dell > Apple ≈ Intel ≈ Sony ≈ Yahoo > Nokia > IBM

Consider the key implication of this. Counting analysis fails because it ignores the experimental design. If we use a single version of the design, employing counting analysis can lead to misleading conclusions.

Returning to the first respondent's data, which brand is the second most preferred? Each of *Samsung*, *Google*, and *Dell* have been chosen as best once. Does this mean they are all in equal second? No, it does not. In Question 4, Dell was against *Google*, and *Google* was preferred. Thus, we know that:

Microsoft > Google > Dell > Apple ≈ Intel ≈ Sony ≈ Yahoo > Nokia > IBM

But note that Samsung has been removed. Samsung is a problem. It may be between Microsoft and Google. It may be between Google and Dell. Or it may be less than Dell. There is no way to tell. We

can guess that it has the same appeal as Dell. Samsung is in blue: while the guess is not silly, it is nevertheless not a super-educated guess:

Microsoft > Google > Samsung ≈ Dell > Apple, Intel, Sony, Yahoo > Nokia > IBM

A more difficult problem is posed by respondent 13's data, shown below. She chose Apple twice as best, Samsung twice, and Google and IBM once each. Which is her favorite? Here it gets ugly.

ID 13

Questions	Alternatives					Choices
	1	2	3	4	5	
1	Apple	Microsoft	IBM	Google	Nokia	Best
2	Apple	Sony	Dell	Yahoo	Nokia	Worst
3	Microsoft	Intel	Samsung	Sony	Nokia	
4	IBM	Google	Intel	Sony	Dell	
5	Microsoft	Google	Samsung	Dell	Yahoo	
6	Apple	IBM	Intel	Samsung	Yahoo	

The data shows that:

Apple > Google (Question 1)	Apple > IBM (Question 1)	IBM > Google (Question 4)
Google > Samsung (Question 5)	Samsung > Apple (Question 6)	Samsung > IBM (Question 6)

This data is contradictory. Look at the first row of conclusions. They tell us that:

Apple > IBM > Google

But the last three tell us that

Google > Samsung > Apple ≈ IBM

Most people's instinct, when confronted by data like this, is to say that the data is bad and to chuck it. Unfortunately, it is not so simple. It turns out most of us give inconsistent data in surveys. We get distracted and bored, taking less care than we perhaps should. We change our minds as we think. The interesting thing about MaxDiff is not that it leads to inconsistent data; rather, it is that it allows us to see that the data is contradictory. This is a good thing: had we instead asked the respondent to rank the data, it would still have contained errors, but we would never have seen them, as we would have no opportunity to see the inconsistencies.

To summarize, computing scores for each respondent by summing up the best scores and subtracting the worst scores is not valid, because:

1. Analysis that ignores the experimental design aspects of which alternatives are shown with which other alternatives will be misleading.
2. Respondents provide inconsistent data and this needs to be modeled in some way.
3. We do not have enough data to get a complete ordering of the alternatives from a single person.
4. People differ in their preferences (compare respondents 1 and 13), so we should not pool all the data when performing analyses.

Fortunately, a bit of statistical wizardry can help us with these problems, each of which is well understood.

The problem of respondents providing inconsistent data is not new. It has been an area of active academic research since the 1920s.³ The area of research that deals with this is known as *random utility models*, and if you are reading this post you may already be familiar with this class of models (e.g., *multinomial logit*, *latent class logit*, *random parameters logit* are all models that solve this problem). These models also take into account which alternatives were shown in which questions, so they solve the second issue as well.

The problems of needing to pool data among respondents while still considering differences between them has been one of the most active areas in statistics for the past 20 or so years. Consequently, good solutions exist for those problems as well. The next two chapters introduce two techniques that address all four issues above: latent class analysis, and hierarchical Bayes.

Coefficients and preference share

Earlier we saw that the data for respondent 1 was consistent with the following preferences:

Microsoft > Google > Samsung ≈ Dell > Apple ≈ Intel ≈ Sony ≈ Yahoo > Nokia > IBM

³ Thurstone, L. L. (1927a) A Law of comparative judgment. *Psychological Review*, 34, 273-286.

Such an analysis is useful for a single respondent, but there is no straightforward way to summarize such findings across a sample of respondents. The solution is to assign a score to each of the alternatives such that the score is consistent with the preferences. These scores are referred to as *coefficients*. One convention when assigning coefficients is to assign a score of 0 to the first brand and then assign the other coefficients relative to that. The first column of numbers below is consistent with the data above. An even better way of doing this is to compute the average and subtract that from all the scores, so that the average coefficient becomes 0. This is shown in the second column. Both conventions are in widespread use.

	Apple is 0	Average is 0
Apple	0	-0.4
Google	2	1.6
Samsung	1	0.6
Sony	0	-0.4
Microsoft	3	2.6
Intel	0	-0.4
Dell	1	0.6
Nokia	-1	-1.4
IBM	-2	-2.4
Yahoo	0	-0.4

In the coefficients above, a value of 1 has been used as the difference between all ranks. We can do better than that. If a respondent is consistent in their preferences, we would want to give a bigger difference, whereas if they are inconsistent, a smaller difference. This is achieved by *logit scaling*. Logit scaling involves estimating coefficients that can be used to predict the probability that a person will choose an alternative in a question, such that these probabilities best align with their actual choices.

The basic idea of logit scaling is best appreciated by a simple example. The first column of the table below shows the proportion of people to choose each option as best in the first question of the tech study. The **Coefficient** column contains *coefficients*. The third column contains $e^{\text{Coefficient}}$. The

preference share is computed by dividing the values in this column by the total.⁴ For example, $3.6702 / 7.7511 = .4735 = 47.35\%$. Some key aspects of this are:

- We assign coefficients to each respondent.
- The coefficients tell us about the respondent's preference between the alternatives: If one alternative has a higher coefficient than another, it implies that the respondent prefers the alternative with the higher coefficient.
- The coefficients also allow us to compute preference shares, such that these shares are consistent with the choices of best and worst predictions in the data.

	Preference share	Coefficient	exp(Coefficient)
Apple	47.35%	1.3002	3.6702
Microsoft	16.89%	.2692	1.3090
IBM	2.65%	-1.5832	.2053
Google	26.82%	.7319	2.0789
Nokia	6.29%	-0.7182	.4876
Total	100.00%	0.00	7.7511

In this example, the coefficients exactly match the preference shares observed for the first question. However, in practice, we have a few more challenges that we need to address:

- There are inconsistencies in answers across questions. We can resolve this by computing the coefficients that most closely match the preference shares across all of the questions, which means that they do not perfectly match with the observed preferences for each question.
- Respondents differ in their preferences. We saw this with respondents 1 and 13 above. To address this, we need to compute a separate set of coefficients for each respondent.

⁴ Sawtooth Software has popularized an alternative formula for computing preference shares. For the formulas, see [https://wiki.q-researchsoftware.com/wiki/Marketing - MaxDiff - Analyze as a Ranking Question - Compute Sawtooth-Style Preference Shares from Individual-Level Parameter Means \(K Alternatives\)](https://wiki.q-researchsoftware.com/wiki/Marketing_-_MaxDiff_-_Analyze_as_a_Ranking_Question_-_Compute_Sawtooth-Style_Preference_Shares_from_Individual-Level_Parameter_Means_(K_Alternatives))

- We do not have enough data to estimate accurately the preferences of a single respondent. For example, for the first respondent, we had insufficient data to work out preferences for Google versus Samsung versus Dell. The solution to this problem is to *pool* the data, so that we can estimate one respondent's preferences by borrowing information from other respondents. For example, if we found that most respondents preferred Samsung to Dell, we could assume that this is true for Respondent 1, even though we have no direct data about this from respondent 1. While this idea may sound a bit strange if you are new to it, it has long been established that models that pool data in such a way outperform models that are estimated separately for each respondent (and, as mentioned, we do not have enough data to estimate models for each respondent anyway).

Techniques for estimating coefficients

Two main techniques in widespread use today for estimating coefficients from MaxDiff are hierarchical Bayes and latent class analysis. These are the focus of the next two chapters.

Utilities

Typically, the word *utility* refers to transformed coefficients from a *choice model*. When discussed in the context of a MaxDiff model, it typically refers to any of the following:

- Coefficients at the respondent level
- Average coefficients
- Preference shares at the respondent level
- Average preference shares
- Some transformation of the coefficients or shares, at the average or respondent level (e.g., scaled so that a 0 is given for the least preferred alternative and 100 for the most preferred alternative)

Using preference shares, and referring to them as preference shares, is often best for non-technical audiences. Alternatively, use utilities or coefficients, but report them using a *bumps chart (ranking plot)*, which focuses on relativities rather than the actual values.

You can extract the utilities (zero-centered coefficients) for each person by selecting a MaxDiff model and:

- In Q: **Create > Marketing > MaxDiff > Save variable(s) > Zero-Centered Utilities**
- In Displayr: by scrolling to the bottom of the object inspector and choosing **SAVE VARIABLE(S) > RLH (Root Likelihood) > Zero-Centered Utilities**.

You can then transform these into other formats using the various tools for transforming variables. For example, if you want to modify them to have a minimum of 0 and a maximum of 100:

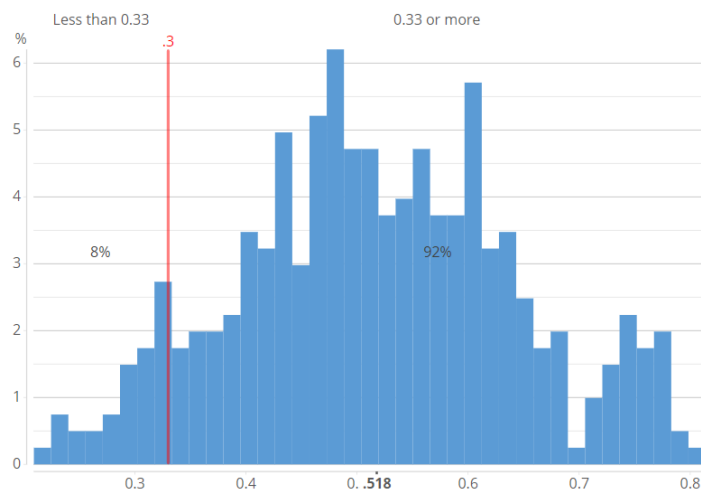
- In Displayr:
 - Select the variables
 - Click the + to insert new variables and select **Ready-Made New Variable(s) > Scale Variables > Unit Scale Within Case**.
 - Select the new variables, click the + to insert new variables and select **Ready-Made New Variable(s) > Multiply** and set **The single numeric value to Multiply by** to 10.
- In Q:
 - Select the variables
 - **Automate > Browse Online Library > New Variables > Scale Variable(s) > Unit Scale Within Case**.
 - Right click on the first of the newly created variables and select **Edit R Variable**
 - Enter at the beginning of the first line of code `100*`
 - Press the play button (the blue triangle).
 - Press **Update R Variable**.

Data cleaning – removing poor quality respondents

MaxDiff questions can be a bit boring. When bored, some respondents will select options without taking the time to evaluate them with care. Fortunately, this is easy to detect with MaxDiff: we look at how well the model predicts a person's actual choices. The simplest way to do this is to count up the

number of correctly predicted best choices. However, an even better approach is to use the RLH statistic. The approach works as follows:

- For each respondent, calculate the RLH (this is described below).
- Plot the RLH statistics for each person and determine a cut-off point. For example, if we've given people five options in each question, the cut-off needs to be at least $1/5 = 0.2$. However, hierarchical Bayes models tend to overfit data, so a higher cut-off is prudent. The histogram to the right suggests that there is a clump of people at around 0.33, which perhaps represents random choosers, but there is no easy way to be sure (if your data contains information on time taken to complete questions, this can also be taken into account).
- Re-estimate the model using only people with RLH statistics above the cut-off value.



How to calculate RLH

You can extract the RLH for each person by selecting a MaxDiff model and:

- In Q: **Create > Marketing > MaxDiff > Save variable(s) > RLH (Root Likelihood)**
- In Displayr: by scrolling to the bottom of the object inspector and choosing **SAVE VARIABLE(S) > RLH (Root Likelihood)**.

How RLH is calculated

The RLH (root likelihood) is computed as follows:

For each question, compute the estimated probability that the person chooses the option that they choose. (This is a computation performed by the model you use, such as Hierarchical Bayes).

- Multiply the probabilities together. For example, in a study involving four MaxDiff questions, if a person chooses an option as best that the model predicted they had a 0.4 probability of choosing, and their choices in the remaining three questions had probabilities of 0.2, 0.4, and

0.3 respectively, then the overall probability is 0.0096. This is technically known as the person's *likelihood*.

- Compute $\text{likelihood}^{(1/k)}$, where k is the number of questions. In this example, the result is 0.31. The value is known as the *root likelihood* (RLH). It is better than just looking at the percentage of the choices that the model predicts correctly, as it rewards situations where the model was close and penalizes situations where the model was massively wrong. Note that the RLH value of 0.31 is close to the mean of the values (technically, it is the *geometric mean*).

Two main techniques in widespread use today for estimating coefficients from MaxDiff are hierarchical Bayes and latent class analysis. These are the focus of the next two chapters.

Latent class analysis

Latent class analysis is used to find groups of people that revealed similar preferences when forming segments.

Latent class analysis is like *cluster analysis*. You put in a whole lot of data and tell it how many classes (i.e., clusters) you want. Latent class analysis is something of a catch-all expression covering many different techniques for forming groups in data. There are two specific variants which can be used for analyzing MaxDiff data: one is the *tricked-logit model* and another is the *rank-ordered logit model*.⁵ The tricked-logit model is the one popularized by Sawtooth. Both models give pretty much the same results, and both are available in Q and Displayr.

When to use latent class analysis

Latent class analysis is used with MaxDiff studies for two quite different reasons:

1. To create segments. The output of latent class analysis is a small number of groups of respondents with different preferences. These groups can be treated as segments.
2. As an alternative to hierarchical Bayes. As is discussed in the next chapter, if one is not focused on segmentation, hierarchical Bayes is the go-to advanced analytic technique for analyzing MaxDiff data. However, latent class analysis makes very different assumptions and can occasionally outperform hierarchical Bayes, particularly if data shows evidence of a small number of discrete segments.

⁵ See <https://www.displayr.com/tricked-vs-rank-ordered-logit/> for a discussion about the differences between the two.

Case study

The output below shows the results from the 5-class model, which is to say a latent class analysis model that assumes there are five types of people in terms of their preferences as revealed by the MaxDiff study. (Why five? We'll return to this later.)

The **Mean** column shows the average *coefficient* estimated for each respondent. These coefficients have some special properties we'll explore later, but for the moment, the way to read them is that the average coefficient across all the respondents and all the alternatives is 0. Reading down the **Mean** column, we can see that Google is, on average, the most preferred brand, followed by Apple and Samsung.

MaxDiff: 5-class Latent Class Analysis

Leave-2-out cross-validation accuracy (in-sample): 56.3% (65.1%); mean RLH: 0.31 (0.32)

● Class 1 (28%) ● Class 2 (25%) ● Class 3 (21%) ● Class 4 (15%) ● Class 5 (10%)

	Respondent Coefficients	Mean	Standard Deviation
Apple		1.1	2.0
Microsoft		0.7	0.6
IBM		-1.2	0.4
Google		1.3	0.8
Intel		-0.5	0.3
Samsung		0.9	1.4
Sony		0.7	0.6
Dell		-0.7	0.9
Yahoo		-1.4	0.5
Nokia		-0.8	0.8

n = 302 cases used in estimation; number of questions: 6; questions used in estimation: 4; questions left out: 2;
number of alternatives: 10; alternatives per question: 5; number of parameters: 49;
accuracy (in-sample): 56.3% (65.1%); mean RLH: 0.31 (0.32); g.mean RLH: 0.29 (0.30); certainty: 22% (26%);
log-likelihood: -1,516 (-2,890); BIC: 3,312 (6,060);
time taken to run analysis: 10 seconds; column width: 0.12;

The histograms show the variation among the respondents. The actual values are not shown because they are not particularly meaningful. The key things to appreciate are the patterns and relativities. We can see, for example, that preferences for Apple are very diverse relative to most of the other brands, whereas there is limited disagreement regarding Google. This is communicated quantitatively via the **Standard Deviation** column.

It may have occurred to you that there is something a bit odd about this output. We have told you we are estimating a latent class model with five classes, but the histograms reveal that there are more than five unique values for each coefficient. This is because some respondents have data that is best described as being an average of multiple classes.

The colors show the values by segment. Looking at Apple, we can see that rightmost column is orange, which corresponds to Class 2. That is, the second class have a strong preference for Apple. We can also see they like Google, are in the middle of the pack for IBM, and, dislike Sony and Nokia relative to the other groups.

The table below shows the coefficients estimated in each of the five classes and the size of the classes. Looking at Class 2, we can see that it tells the same story in terms of preferences for Apple, Microsoft, Sony, and Nokia that we could see in the histograms.

	↕ Class 1 ↴ (28.3%)	↕ Class 2 ↴ (24.8%)	↕ Class 3 ↴ (21.1%)	↕ Class 4 ↴ (15.5%)	↕ Class 5 ↴ (10.4%)
Apple	2.9	3.0	-2.7	1.0	-.0
Microsoft	-.0	1.7	.8	.4	.1
IBM	-1.7	-.9	-1.2	-1.4	-.2
Google	1.5	1.6	2.3	-.5	.7
Intel	-1.1	-.6	-.2	-.2	-.0
Samsung	1.7	-.7	2.3	2.1	-2.0
Sony	1.1	-.5	.6	1.6	1.2
Dell	-2.3	-.1	-.6	.8	-.1
Yahoo	-1.1	-1.6	-1.4	-2.4	-.5
Nokia	-.9	-1.9	.0	-1.3	.8

The next table shows the *preference share* by class. This is often the most useful output when the goal is to describe the segments. Reading the first row, for example, we can see that the analysis has indeed identified that a fundamental difference between segments relates to their preference for Apple, with Google, by contrast, having some popularity in all segments and Samsung being strong in one segment (Class 2).

%	Class 1 (31.9%)	Class 2 (22.8%)	Class 3 (21.7%)	Class 4 (16.4%)	Class 5 (7.16%)	Total
Apple	65.2	.5	41.2	36.4	1.1	35.9
Microsoft	10.0	6.6	14.9	.8	19.0	9.4
IBM	.3	.8	2.2	.7	7.5	1.4
Google	11.4	17.6	12.4	8.4	16.0	12.9
Intel	.5	2.3	5.2	1.7	5.5	2.5
Samsung	9.9	58.4	1.6	24.3	2.2	21.0
Sony	1.8	9.2	9.1	18.1	21.4	9.1
Dell	.3	2.4	10.7	.5	3.9	3.3
Yahoo	.3	.7	1.3	.7	9.5	1.3
Nokia	.3	1.5	1.4	8.4	14.0	3.1

It is also possible to compute *coefficients* and *preference shares* for each respondent, but this will be the focus of the next chapter (the process and interpretation are the same).

Choosing the number of classes

As with cluster analysis, a key input to latent class analysis is the required number of *classes*, where class means the same thing as cluster in cluster analysis and segment in segmentation.

Our eBook *How to do Market Segmentation* contains a more detailed discussion of issues involved in how to select the number of classes from a segmentation, so this chapter will focus purely on the technical side of selecting the number of classes.

If the goal of the analysis is to create segments, the basic approach to choosing the number of segments is to trade off statistical and managerial considerations. If our focus is on creating an alternative to a hierarchical Bayes model, then we just take statistical considerations into account.

In the header and footer to the latent class analysis outputs you will see the *Prediction accuracy* and *BIC*. The output below shows these measures for models from one to ten classes, and for a hierarchical Bayes model, shown in the bottom row.

	⇄ BIC ↑	⇄ In-sample accuracy ↑	⇄ Out-of-sample accuracy ↑	⇄ In-sample RLH ↑	⇄ Out-of-sample RLH ↑
MNL	6738.723	.434	.432	.262	.267
2 Latent classes	6325.556	.585	.560	.291	.297
3 Latent classes	6174.173	.608	.561	.302	.301
4 Latent classes	6110.547	.632	.566	.310	.305
5 Latent classes	6059.779	.651	.563	.317	.306
6 Latent classes	6050.465	.665	.588	.322	.310
7 Latent classes	6039.637	.666	.555	.325	.308
8 Latent classes	6056.244	.668	.565	.328	.310
9 Latent classes	6067.898	.692	.598	.332	.310
10 Latent classes	6066.790	.687	.570	.336	.311
HB	2883.014	.955	.680	.617	.347

This study asked each respondent six questions. We have used four of their questions, randomly selected, to estimate the models. We then predict the best choice for the remaining two questions, which is shown as *Out-of-sample accuracy* in the table above. We can see that the best of the latent class analysis models is the one with 9 segments. The hierarchical Bayes model is clearly the best of all the models.

The root likelihood (RLH), is a measure of accuracy which takes into account the probability of the prediction (e.g., if the model predicts that an alternative has a 90% chance of being selected, but it's not selected, the statistic is worse than if a 50% probability and not selected). It suggests that the 10 class solution is best, which implies that we might want to investigate even larger number of classes than 10 if our focus is purely on predictive accuracy.

An alternative statistic for working out the number of classes is the BIC statistic. The smaller the better. This statistic suggests that the best model is the one with seven classes.

Which statistic should we believe? The *out-of-sample RLH* is the best of these in that it relies on few assumptions and is relatively sensitive. But, it's still affected by the randomness of selecting the questions to use in the analysis. The BIC statistic, by contrast, is more consistent, so it can be useful. The lack of consistency across the different statistics means that there is no clear way of making a choice, so we should instead focus on non-statistical considerations, or, use the HB model.

Other considerations that can be taken into account when selecting the number of classes are:

- The stability of the **Mean** column: You should keep increasing the number of classes until the means do not change much between different number of classes.
- The usefulness of the solution in inspiring segmentation strategy: With a study about brand, this reduces to whether the brands of interest to the stakeholders have differing preference shares by segment. The above segmentation may be useful for *Apple*,

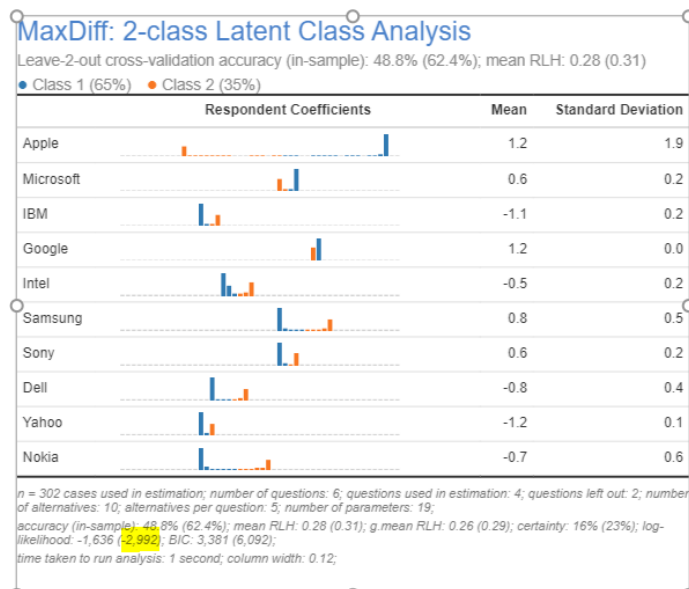
Samsung, Sony, and Google but has limited value for the other brands. Admittedly this segmentation is not so interesting anyway, as brand is rarely a very useful segmentation variable.

- The complexity of the solution for stakeholders: The fewer classes, the more intelligible.

Start points

Latent class models start by randomly generating a solution, and then improving on this. But, the initial random configuration may not be the best, meaning that the final solution is what's known as a *local optima*. To check if this is a problem it is possible to re-run the solution from a different random start point. To do this:

- Duplicate the model that you've created.
- Change **Inputs > MODEL > Seed**. Choose any integer you like (e.g., 143, 23,34, 34324). Whatever number you choose will lead to a different random start.
- Examine the in-sample log-likelihood (see below in yellow). The closer to 0 the better.



```

1 library(FlipMaxDiff)
2 df <- if (exists("formCovariates"))
3   names(formCovariates) <- sample(
4     data.frame(formCovariates)
5   ) else
6     NULL
7
8 LCA2.2 <- FitMaxDiff(design = if(
9   version = if (is.null(formVer
10  best = data.frame(formBest),
11  worst = data.frame(formWorst),
12  seed = 1232,
13  alternative.names = if (formMe
14  n.classes = if(exists("formCla
15  subset = if (all(QFilter)) NUL
16  weights = QPopulationWeight,
17  characteristics = df,
18  lc = if (exists("formLC")) for
19  output = if (formType == "Vary
20  tasks.left.out = formCV,
21  is.tricked = if(exists("formMc
22  algorithm = if (formType == "
23  hb.iterations = if(exists("for
24  hb.chains = if(exists("formCha
25  hb.max.tree.depth = if(exists(
  
```

OUTPUT

Re-estimating the model

All the models estimated in this chapter have used two questions for cross-validation. Once we have selected the desired number of classes we should re-estimate the model without cross-validation, so that we make full use of all the available data. This final model should be used for reporting purposes.

Profiling latent classes

Once we have created our segmentation we can allocate each person to a class and then profile the classes by creating tables. The table below, for example, shows the 5-class solution by product ownership. If you compare this table with the latent class solution itself, you will see that the product ownership lines up with the preferences exhibited in the MaxDiff questions.

Column %	iPad	iPod	iPhone	Samsung mobile phone	Other mobile phone (not Samsung and not iPhone)	Mac computer - desktop	Mac computer - laptop
1	46% ↑	41%	52% ↑	18% ↓	22% ↓	53% ↑	59% ↑
2	9% ↓	13% ↓	7% ↓	43% ↑	28%	3% ↓	1% ↓
3	24%	22%	23%	17%	20%	18%	19%
4	17%	19%	15%	15%	17%	25%	18%
5	4%	5%	4% ↓	7%	13% ↑	3%	3%
NET	100%	100%	100%	100%	100%	100%	100%

Software – Q

Q has two entirely different systems for estimating latent class models. If the data has been set up as a *Ranking Question*,⁶ the general *latent class analysis* approach can be used (**Create > Segments > Latent Class Analysis**). This approach is particularly useful if your focus is market segmentation, as it contains automated tools for selecting the number of segments and for creating segments with multiple different types of data. See our eBook *How to do Market Segmentation* for more information about this.

The rest of this section (and eBook) instead assumes you are using the special-purpose latent class analysis routines designed for MaxDiff.

Importing the experimental design

Go to **File > Data Sets > Add to Project** and import the experimental design however you want. If you wish to replicate the examples in this book, you can download the design from here:

https://wiki.q-researchsoftware.com/images/7/78/Technology_MaxDiff_Design.csv

Importing the survey data

Import the data file using **File > Data Sets > Add to Project**. If you wish to replicate the examples in this book, you can download the data from here: https://wiki.q-researchsoftware.com/images/f/f1/Technology_2017.sav

The screenshot shows the 'Inputs Properties' dialog box for 'MaxDiff - Latent Class Analysis'. The dialog is divided into several sections: 'EXPERIMENTAL DESIGN', 'RESPONDENT DATA', and 'MODEL'. Under 'EXPERIMENTAL DESIGN', there are fields for 'Design location' (set to 'Variables'), 'Alternatives' (a list of five alternatives: 'Alt: 1 [Alt.1] [Technolo...', 'Alt: 2 [Alt.2] [Technolo...', 'Alt: 3 [Alt.3] [Technolo...', 'Alt: 4 [Alt.4] [Technolo...', and 'Alt: 5 [Alt.5] [Technolo...'), 'Version' (set to 'Version [Technology_...'), and 'Best selections' (a list of three selections: 'Q5d: Most [Q5d_left] [...', 'Q5e: Most [Q5e_left] [...', and 'Q5f: Most [Q5f_left] [T...'). Under 'RESPONDENT DATA', there are fields for 'Version' (empty), 'Worst selections' (a list of three selections: 'Q5d: Least [Q5d_right] [...', 'Q5e: Least [Q5e_right] [...', and 'Q5f: Least [Q5f_right] [...'), and 'Alternative names' (set to 'For example: Coke, Diet Coke, ...'). Under 'MODEL', there are fields for 'Type' (set to 'Latent class analysis'), 'Number of classes' (set to 5), 'MaxDiff model' (set to 'Tricked logit'), and 'Questions left out for cross-validation' (set to 2).

⁶ https://docs.displayr.com/wiki/Setting_Up_a_MaxDiff_Experiment_as_a_Ranking

Estimating the model

Select **Create > Marketing > MaxDiff > Latent Class Analysis**, changing **Inputs > EXPERIMENTAL DESIGN > Design source** to **Variables** and:

- Click into the **Alternatives** box and type “alt”. Then select all the “Alt” variables from the experimental design file. Be sure to select them in the right order (1 to 5). As this design is a single-version design, there is no need to specify the version.
- For the **RESPONDENT DATA**:
 - Click in the **Best selections** field in the Object Inspector (far right), and type `Most`, which filters the variables that contain the word “most” in the name. Select the six variables, being careful to choose them in order.
 - Select all the **Worst selections** by searching for `Least`.
- Under **MODEL**:
 - Set the **Number of Classes** to 5.
 - Set **Questions left out for cross-validation** to 2.

Choosing the number of classes

Copy and paste the latent class analysis output in the **Report**, changing the number of classes. Do this as many times as you like to create different models. Then:

- Select **Create > Marketing > MaxDiff > Compare Models**
- Select your **MaxDiff models** in **Inputs > Input Models**

Class parameters

Click on the model in the **Report** and select **Inputs > DIAGNOSTICS > Create > Marketing > MaxDiff > Diagnostic > Class Parameters Table** on the right.

Class preference shares

Click on the model in the **Report** and select **Inputs > SAVE VARIABLE(S) > Class Preference Shares** on the right.

Class membership

Click on the model output and select **Inputs > SAVE VARIABLE(S) > Class Membership** on the right. This adds a new variable to the project.

Profiling segments

Press **Create > Tables > Table**, and select a class membership question (e.g. *Class memberships from max.diff*) in the **Blue Dropdown menu** and another question in the **Brown Dropdown menu** (e.g. Q6 *Device ownership*)

Software – Displayr

Importing the experimental design

Select **+ Add a data set** in the bottom left and import the experimental design whichever way you wish. If you want to replicate the examples in this book, you can download the design from here:
https://wiki.q-researchsoftware.com/images/7/78/Technology_MaxDiff_Design.csv

Importing the survey data

Click the **+** button under **Data Sets** in the bottom left. If you wish to replicate the examples in this book, you can download the data from here:

https://wiki.q-researchsoftware.com/images/f/f1/Technology_2017.sav

Estimating the model

Anything > Advanced Analysis > MaxDiff > Latent Class Analysis, changing **Inputs** > **EXPERIMENTAL DESIGN** > **Design source** to **Variables** and:

- For the experimental design in **Data** (Technology_MaxDiff_Design.csv), drag **Alt** from the experimental design file in the **Data** tree (bottom left) to **Alternatives** (on the right). As this design is a single-version design, there is no need to specify the version.
- For the data file in **Data** (Technology_2017.sav):
 - Click in the **Best selections** field in the Object Inspector (far right), and type `Most`, which filters the variables that contain the word “most” in the name. Select the six, being careful to choose them in order.
 - Select all the **Worst selections** by searching for `Least`.
- Under **MODEL**:
 - Set the **Number of Classes** to 5.
 - Set **Questions left out for cross-validation** to 2.

MaxDiff - Latent Class Analysis

EXPERIMENTAL DESIGN

Design location: Variables

Alternatives: Alt: 1 [Alt.1] [Technolo..., Alt: 2 [Alt.2] [Technolo..., Alt: 3 [Alt.3] [Technolo..., Alt: 4 [Alt.4] [Technolo..., Alt: 5 [Alt.5] [Technolo...

Version: Version [Technology...]

RESPONDENT DATA

Version:

Best selections: Q5d: Most [Q5d_left] [... , Q5e: Most [Q5e_left] [... , Q5f: Most [Q5f_left] [... , ...

Worst selections: Q5d: Least [Q5d_right] [... , Q5e: Least [Q5e_right] [... , Q5f: Least [Q5f_right] [... , ...

Alternative names: For example: Coke, Diet Coke, ...

MODEL

Type: Latent class analysis

Number of classes: 5

MaxDiff model: Tricked logit

Questions left out for cross-validation: 2

Choosing the number of classes

Copy and paste the page with a latent class analysis output on it, changing the number of classes. Do this as many times as you wish to produce new models. Then:

- Select **Anything > Advanced Analysis > MaxDiff > Compare Models**
- Select your **MaxDiff** models in **Inputs > Input Models**

Class parameters

Click on the model output, and, in the object inspector (right-side of the screen), select **Inputs > DIAGNOSTICS > Class Parameters Table** (you may have to scroll down to see this option).

Class preference shares

Click on the model output, and, in the object inspector (right-side of the screen), select **Inputs > DIAGNOSTIC > Class Preference Shares Table** (you may have to scroll down to see this option).

Class membership

Click on the model output, and, in the object inspector (right-side of the screen), select **Inputs > SAVE VARIABLE(S) > Class Membership** (you may have to scroll down to see this option).

Profiling segments

On a new page, press the **Table** button, and drag across a class membership variable (e.g. class memberships from max.diff) from **Data** to the **Rows** box and Q6 Device ownership to **Columns**.

Hierarchical Bayes

Hierarchical Bayes is the state-of-the-art technique for estimating coefficients for respondents in a MaxDiff study.

Hierarchical Bayes for MaxDiff is a similar idea to latent class analysis. There is no accurate way of describing how it works without getting into the math in some detail. A simple way of describing how it works is to contrast it with latent class analysis:











- Latent class analysis essentially pools the data of respondents together by assuming that there are a small number of different types of respondents.
- Hierarchical Bayes pools together the data by assuming that there are an infinite number of segments, by assuming that the diversity between the respondents can be described, *a priori*, as being multivariate normal.

When computing coefficients for each respondent, both models can be understood as assigning respondents coefficients as a weighted average of other similar respondents. As the hierarchical Bayes model assumes an infinite number of segments, in practice it is able to approximate the data of respondents more closely, even though the multivariate normal assumption is not really a good description of the true variation in the population.

As you can see from the output below, the resulting histograms of coefficients for the respondents are clearly not overly influenced by the normality assumption (as none of these distributions is normal).

MaxDiff: Hierarchical Bayes

Leave-2-out cross-validation accuracy (in-sample): 68.0% (95.5%); mean RLH: 0.35 (0.62)

	Respondent Coefficients	Mean	Standard Deviation
Apple		2.8	5.5
Microsoft		1.2	2.0
IBM		-2.3	1.6
Google		2.3	2.0
Intel		-1.1	1.3
Samsung		1.8	3.4
Sony		1.0	2.1
Dell		-1.4	2.4
Yahoo		-2.6	1.5
Nokia		-1.6	2.5

n = 302 cases used in estimation; number of questions: 6; questions used in estimation: 4; questions left out: 2; number of alternatives: 10; alternatives per question: 5; number of parameters: 54;
 accuracy (in-sample): 68.0% (95.5%); mean RLH: 0.35 (0.62); g.mean RLH: 0.21 (0.59); certainty: 4% (67%); log-likelihood: -1,871 (-1,287); BIC: 4,050 (2,883);
 lowest effective sample size (Mean): 89.0 at Samsung; highest Rhat (Mean): 1.1 at Yahoo; lowest effective sample size (St. Dev.): 29.0 at Samsung; highest Rhat (St. Dev.): 1.1 at Microsoft; time taken to run analysis: 40 seconds; column width: 0.44;

Respondent-level coefficients

Coefficients for each respondent, also known as *respondent-level parameters*, can be extracted from both hierarchical Bayes and latent class analysis models. You may recall that two chapters earlier we analyzed the data for respondent 1 and concluded that the data showed that:

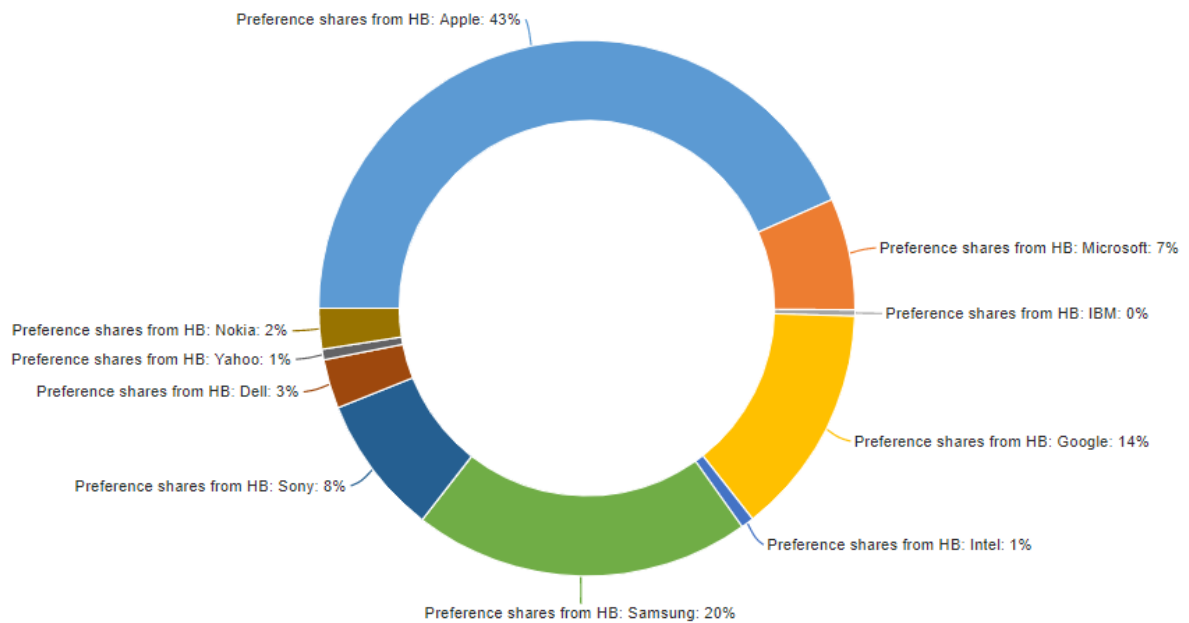
Microsoft > Google > **Samsung** = Dell > Apple, Intel, Sony, Yahoo > Nokia > IBM

Samsung is in blue because it was more of a guesstimate. The estimated coefficients for the respondent are broadly consistent with this ordering, although we see Samsung being a little more preferred than Google, and Yahoo being more negatively positioned than Nokia. This is because where there is limited information, the analysis borrows preferences from other respondents.

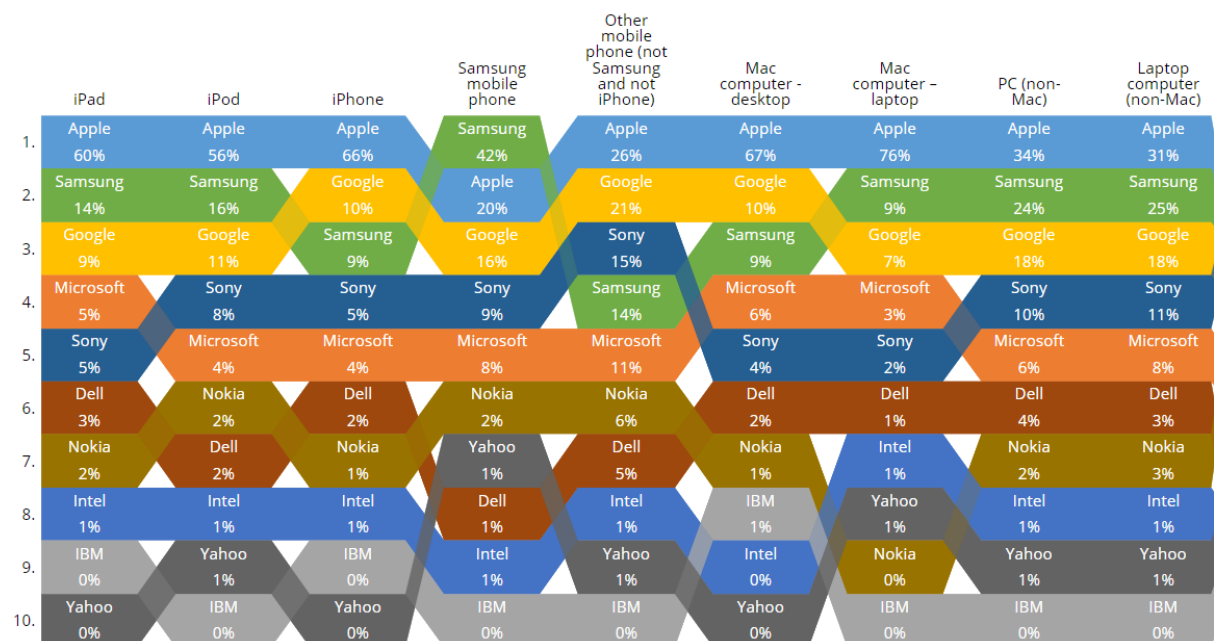
Values	Apple	Microsoft	IBM	Google	Intel	Samsung	Sony	Dell	Yahoo	Nokia
1	-.34	5.33	-3.93	3.31	-.95	2.08	-.51	1.24	-2.93	-3.28
2	.62	1.30	-4.93	5.92	-2.30	3.85	.66	-.82	-4.10	-.20
3	8.76	1.02	-3.34	4.04	-2.38	.30	.21	-6.03	-1.88	-.70
4	-.52	1.15	-4.13	4.29	-.77	5.40	-3.71	1.48	-2.13	-1.05
5	8.14	-.20	-2.17	.21	-2.12	2.17	3.33	-1.00	-5.01	-3.36
6	2.57	.31	-4.95	3.78	-1.22	.18	4.19	-.79	-3.59	-.48
7	3.99	1.90	-2.59	5.35	-2.79	.21	.27	-.73	-4.92	-.69
8	8.59	1.60	-.55	4.57	-.56	-4.87	-2.41	.21	-1.41	-5.18
9	8.42	-.02	-.52	.50	-.52	1.51	-.37	-2.93	-.73	-5.33
10	-2.18	.59	1.16	1.28	-1.16	-2.14	2.92	-1.84	.96	.42
11	4.28	3.63	-2.62	2.52	-1.02	3.95	-.23	-2.80	-2.97	-4.75
12	-6.27	1.47	-1.28	4.99	-2.81	.02	1.10	-2.09	.90	3.97
13	8.22	.86	.98	2.14	-.45	-.20	-2.37	-.17	-3.02	-6.00

Preference shares

Preference shares can also be estimated for each respondent. The following donut plot shows the average preference shares.



These are useful for producing analyses with other data, by comparing average preference shares by the other data. For example, the plot below compares the preference shares by product ownership.



Technical parameters

The software used to estimate hierarchical Bayes has a few technical settings, which are described in this section. Typically, unless there is a warning to do otherwise (e.g., the maximum tree depth has been exceeded), there is no need to modify these settings.

Number of classes

As with latent class analysis, it is possible to estimate multiple classes for hierarchical Bayes, where a multivariate normal distribution is assumed within each class. However, this tends not to make a big difference. The basic process for selecting the number of classes is to focus only on the *cross-validation accuracy* and out-of-sample *RLH*, as the *BIC* statistic is problematic when comparing numbers of classes in this situation and the various managerial issues are irrelevant (as the resulting classes do not correspond to segments, because the classes overlap).

Iterations

By default, the hierarchical Bayes software described in this book runs for 100 *iterations*, where each *iteration* is an attempt to improve the model. Sometimes this will be too few (i.e., the model has yet to *converge* after 100 iterations), and it is appropriate to increase the number of iterations. The software will automatically give a warning if it thinks that more iterations are required. You can find out more about this here: <https://www.displayr.com/convergence-hb-maxdiff/>

Chains

This option specifies how many separate *chains* (independent analyses) to run, where the chains run in parallel, given the availability of multiple cores. Increasing the number of chains increases the quality of the results. It does, however, result in a longer running time if chains are queued up. If using Q or Displayr, it is recommended to leave this option at its default value of 8, which is optimal for the *R servers* that we use.

Maximum tree depth

This is a very technical option. The practical implication is that this option should need to be changed only if a warning appears indicating that the maximum tree depth has been exceeded. The default value is 10 and a warning should not appear under most circumstances. If such a warning does appear, try increasing the maximum tree depth to 12 rather than a larger number, which could increase computation time significantly.

Software – Q

If you have not already created a latent class analysis, follow the steps in the previous chapter (see [Software – Q](#)), but choose **Hierarchical Bayes** instead of **Latent Class Analysis**.

If you have already created a latent class analysis, instead click on it and any associated outputs in the **Report** and copy-and-paste them. Then, on the latent class analysis output, change **Type** (under **Inputs > MODEL** on the right) to **Hierarchical Bayes** and set the **Number of classes** to 1.

The remaining instructions describe how to create the Hierarchical Bayes model if you have first created a latent class analysis model.

Respondent-level coefficients

Click on the model in the **Report** and select **Inputs > SAVE VARIABLE(S) > Individual-level Coefficients** or **Zero-centered Utilities**. These options are identical (coefficients is the term that people from an academic background will use; utilities the term that former Sawtooth users tend to use).

Respondent-level preference shares

Click on the model in the **Report** and select **Inputs > SAVE VARIABLE(S) > Preference Shares**.

Donut chart

Press **Create > Charts > Visualization > Donut Chart**, and click into the **Variables** box in **DATA SOURCE**. Then, select the preference share variables and choose your preferred formatting options (under **Chart**).

Ranking plot

Create > Tables > Table and select the preference share variables in the **blue dropdown** menu and **Q6** in the **brown dropdown** menu. Then:

- Right-click on the table and select **Statistics – Cells**, change the statistic to **% Column Share** (turning off Averages).
- From **Show data as** in the toolbar, select **Ranking Plot**.
- In the **Object Inspector** (right-side of the screen), select **Chart > FORMATTING > Show Values > Yes - Below**.

Preference simulation

The most straightforward way to simulate preference share is to change the preference shares for brands you wish to exclude to 0, and use **% Share**, **% Column Share**, or **% Row Share** as **Statistics – Cells**, as this automatically rebases any values and converts them to a percentage. Then, if desired, the alternatives can also be deleted from the table underlying the plot (by first converting it to a table).

To remove an item from the computation of the shares in a table (for example, to remove *Apple*), the R code of the preference share variables is edited (do this via the **Variables and Questions** tab by right-clicking on the question > **Edit R Variable**. In the **R CODE**, replace:

```
prop.table(exp(as.matrix((flipMaxDiff::RespondentParameters(max.diff))))),  
1)
```

with:

```
z =  
prop.table(exp(as.matrix((flipMaxDiff::RespondentParameters(max.diff))))),  
1)  
z[,1] = 0  
z
```

Noting that `max.diff` can be replaced with whatever your model is named, and that we have set column 1 to be zero as this is the column for *Apple*.

Software - Displayr

If you have not already created a latent class analysis, follow the steps in the previous chapter (see [Software – Displayr](#)), but choose **Hierarchical Bayes** instead of **Latent Class Analysis**.

If you have already created a latent class analysis, instead click on the page in the **Pages** tree that contains latent class analysis output and select **Duplicate**. Then, change **Type** to **Hierarchical Bayes** and set the **Number of classes** to 1.

The remaining instructions describe how to create the hierarchical Bayes model if you have first created a latent class analysis model.

Respondent-level coefficients

Click on the model output, and, in the object inspector (right-side of the screen), select **Inputs > SAVE VARIABLE(S) > Individual-Level Coefficients** or **Zero-Centered Utilities**. You may have to scroll down to see these option. These options are identical (coefficients is the term that people from an academic background will use; utilities the term that former Sawtooth users tend to us).

Respondent-level preference shares

Click on the model output, and, in the object inspector (right-side of the screen), select **Inputs > SAVE VARIABLE(S) > Preference Shares** (you may have to scroll down to see this option).

Donut chart

On a new page, select **Visualization > Donut Chart**. For **DATA SOURCE** (in the Object Inspector) use the box **Variables in 'Data'**. Drag across the preference share variables into the **Variables** box, and choose your preferred formatting options (under **Chart**).

Ranking plot

On a new page, select **Table** and drag across the preference share variables as **Rows** and Q6 as **Columns**. Then:

- In the object inspector, using **Statistics – Cells**, change the statistic to **% Column Share** (turning off Average).
- With the table selected, press **Visualization > Ranking Plot**.
- In the **Object Inspector** (right side of the screen), select **Chart > FORMATTING > Show Values > Yes - Below**.

Preference simulation

The most straightforward way to simulate preference share is to change the preference shares for brands you wish to exclude to 0, and use **% Share, % Column Share**, or, **% Row Share** as **Statistics – Cells**, as this automatically rebases any values and converts them to a percentage. Then, if desired, the alternatives can also be deleted from the table underlying the plot (by first converting it to a table).

To remove an item from the computation of the shares in a table (for example, to remove *Apple*), the R code of the preference share variables is edited. Select a variable from the variable `Preference shares` from `max.diff` and the **R CODE** will be exposed in the right-hand panel.

Replace:

```
prop.table(exp(as.matrix((flipMaxDiff::RespondentParameters(max.diff)))),
1)
```

with:

```
z =
prop.table(exp(as.matrix((flipMaxDiff::RespondentParameters(max.diff)))),
1)
z[,1] = 0
z
```

where `max.diff` is the name of the model. Drag the preference shares question on to the page to see the effect.

Software – R

```
my.design <- read.csv('https://wiki.q-researchsoftware.com/images/7/78/Technology_MaxDiff_Design.csv')
library(foreign)
my.data <- read.spss('https://wiki.q-researchsoftware.com/images/f/f1/Technology_2017.sav',
  use.value.labels = TRUE, to.data.frame = TRUE)

library(flipMaxDiff)
HB <- FitMaxDiff(design = my.design,
  best = my.data[, grep("left", names(my.data))],
  worst = my.data[, grep("right", names(my.data))],
  tasks.left.out = 2,
  is.tricked = TRUE,
  algorithm = "HB-Stan")

# Viewing the main output
print(HB)
```

Multivariate analyses of coefficients

It is common to use respondent-level coefficients as inputs into other analyses.

When doing this a bit more care is required than when using these techniques with other data.

It is commonplace for people to analyze coefficients using other multivariate techniques, such as factor analysis, cluster analysis, regression, and TURF. There are a number of considerations to keep in mind when doing this:

- The individual-level coefficients are themselves the result of models with assumptions. These assumptions can have a significant impact on the resulting analyses, as discussed below.
- The coefficients are estimates with noise attached to them. When you conduct another statistical analysis, using these coefficients as inputs, the level of noise compounds.
- The coefficients are not independent. The precise value of one person's coefficient for one alternative is related to the precise value of the others. For example, if the coefficients have been mean-centered, if the first alternative has a coefficient of -1 then the remaining coefficients will sum to a value of 1. This can lead to weird and obscure errors when applying traditional statistical techniques (e.g., PCA).

Cluster analysis and latent class analysis

A two-step approach is widespread in market research, whereby first the coefficients are estimated for each respondent using hierarchical Bayes, and then either cluster analysis or latent class analysis is used to create segments. The problem with this two-step approach is that it leads to a compounding of errors. The solution is instead to use latent class analysis to estimate the coefficients and create the segments simultaneously.

Hierarchical Bayes (HB) generally achieves a higher predictive accuracy than latent class analysis. This is because it does not assume a small number of archetypal respondents. If you are doing segmentation the whole point is to assign people to a small number of groups, so all the predictive accuracy gains of hierarchical Bayes relative to latent class analysis are likely lost. Furthermore, as hierarchical Bayes makes a different set of assumptions, and these assumptions are both wrong to some unknown extent and inconsistent with latent class analysis, it is highly likely that the two-step approach will be substantially inferior in terms of the true predictive accuracy (i.e., if computed after forming the segments).

As discussed in our eBook *How to do Segmentation*, it is not always the case that the solution that is technically the best will also be the best from a managerial perspective, so it is often worthwhile to use both latent class analysis and also hierarchical Bayes.

Statistical tests

The assumptions of all standard statistical tests are violated by respondent-level coefficients. As each person's coefficient is a weighted average of those of the others, this means that the assumption of observations being independent (aka random sampling) is never met. Further, the structure of the dependency between the coefficients is not consistent with any of the common complex sampling models (e.g., stratification, clustering).

A solution to this is to conduct tests using the coefficients of the archetypal respondents rather than the respondent-level coefficients. The easiest way to do this is to use Q and set up the MaxDiff as **Ranking** questions⁷ and then use Q's automated statistical tests.

The automated significance tests from Q are shown below. Color-coding shows relative performance. Thus, if we wish to test whether the difference between *Apple* by gender is significant, we need to remove all the other brands from the analysis (in Q, right-click on the cells and select **Remove**; in Displayr, click on them and select **Delete**. **Reset** or **Undo** is used for adding all the categories back into the analysis). This is shown on the table below. The table on the right below only contains the data for Intel; it shows that Intel is significantly more preferred, in an absolute sense, among men than women.

Coefficient	Male	Female
Passive (7)	.00	.00
Apple	-.76	-.42

Coefficient	Male	Female
Passive (7)	.00 +	.00 +
Intel	-.78 +	-1.76 +

⁷ https://wiki.q-researchsoftware.com/wiki/MaxDiff_Specifications

In each of these tables the automatic significance testing focuses on relativities. To see how the brands perform in an absolute sense, we need to use gender as a filter rather than as the columns (when using gender in the columns, Q interprets this as meaning you are wanting to compare the genders).

The table output below shows the table with a filter for women and with a Planned Test of Statistical Significance⁸ explicitly comparing *Apple* with the benchmark value of 0, which reveals that the absolute performance of *Apple* among women is significantly below the benchmark rating of 7 out of 10 (from an Anchored MaxDiff study, discussed in the next chapter).

Anchored Max-Diff

SUMMARY

Coefficient	Coefficient
Passive (7)	.00 ↑
Apple	-.50
Microsoft	-.50
IBM	-1.21 ↓
Google	.26 ↑
Intel	-1.17 ↓
Hewlett-Packard	-1.09 ↓
Sony	-.07 ↑
Dell	-1.16 ↓
Yahoo	-1.11 ↓
Nokia	-.19 ↑

Hypothesis testing

Filter: Female
Unweighted
base n = 160; 49% filtered out

Independent Samples t-Test - Comparing Two Coefficients
t = -4.318
Degrees of Freedom = 208.37
n = 320
Effective Sample Size = 320
p = 0.00002

Significant

Null hypothesis: There is no difference in the population between the Probability % for 'Passive (7)', 'Apple' of people in 'Passive (7)' than Apple'.
At the 0.05 level of significance, the null hypothesis is rejected.

Tables Variables and Questions Data No
Filter: Female AND

⁸ https://wiki.q-researchsoftware.com/wiki/Planned_Tests_Of_Statistical_Significance

Dimension reduction

Dimension reduction approaches like *Principal Components Analysis* and *Factor Analysis* are, at a mathematical level, usually computed from correlations.⁹ When we assign coefficients to respondents we do so with error, and the correlations between the coefficients become biased. If we are using hierarchical Bayes, it estimates correlations between variables, and these can be used as inputs to the dimension reduction techniques. To extract these correlations, run the following R code:

```
library(rstan)
cho.mat <- extract(max.diff$stan.fit, pars = "L_omega")[[1]]
cor.mat <- matrix(0, dim(cho.mat)[2], dim(cho.mat)[2])
for (i in 1:dim(cho.mat)[1])
  cor.mat <- cor.mat + cho.mat[i, , ] %*% t(cho.mat[i, , ])
cor.mat <- cor.mat / dim(cho.mat)[1]
```

where `max.diff` on line 2 above is the name of the output of a hierarchical Bayes calculation. To run R code in Q, select **Create > R Output**, paste the code into the editor on the right-hand side and click on the **Calculate** button. Running R code in Displayr is the same except that you begin by clicking the **Calculation** button and drawing a box for the calculation on your page.

If using latent class analysis, high correlations are inevitable due the math of the model,¹⁰ so using respondent-level coefficients as an input to dimension reduction will inevitably lead to false conclusions.

Note that even if you resolve these issues, a practical problem with using correlations is that the variables are not independent, so you will often get an error message.

⁹ Although they can often be equivalently defined in other ways. For example, PCA can be defined relative to the SVD of raw data.

¹⁰ For example, if we have two classes and everybody has a 100% probability of being assigned to either of the classes, then there will be a correlation of 1 or -1 between every variable that contains a coefficient.

Regression

Regression has all the problems described in the previous two sections. There are two solutions to this: regression can be estimated via a *sweep operator* using the estimated correlations,¹¹ or the regression relationships can be built into the MaxDiff estimation process. It seems unlikely that either of these approaches is ever worth the hassle.

TURF

Below we describe how to do TURF with MaxDiff, and then provide a caution about its use.

How to conduct TURF with MaxDiff

TURF assumes that the data is binary (i.e., 0s and 1s). People like an alternative (1) or not (0). MaxDiff instead estimates numeric values for each alternative. Consequently, in order to use TURF we need to first discretize the coefficients (i.e., the utilities).

There are two standard approaches: using rankings or thresholds. With rankings, we first assign a 1 to, say, the two alternatives with the highest coefficients for each respondent. Or, the three highest, or the four highest. It's an arbitrary decision.

With thresholds, we assign a 1 to all coefficients above some threshold (e.g., above the average). The advantage that this approach has over the rankings is that it allows respondents to vary in terms of how many 1s they have in the data, and in the real world we would expect such variation to also exist.

¹¹ James H. Goodnight (1979), "A Tutorial on the SWEEP Operator," *American Statistician* 33: 149-58.

For example, if one respondent loves three of 10 alternatives but hates the rest, she will have three 1s, whereas an average respondent will have five 1s. But, the flipside of this is that where the threshold is set is completely arbitrary and even more difficult to explain than choices regarding rankings.

Software - Q

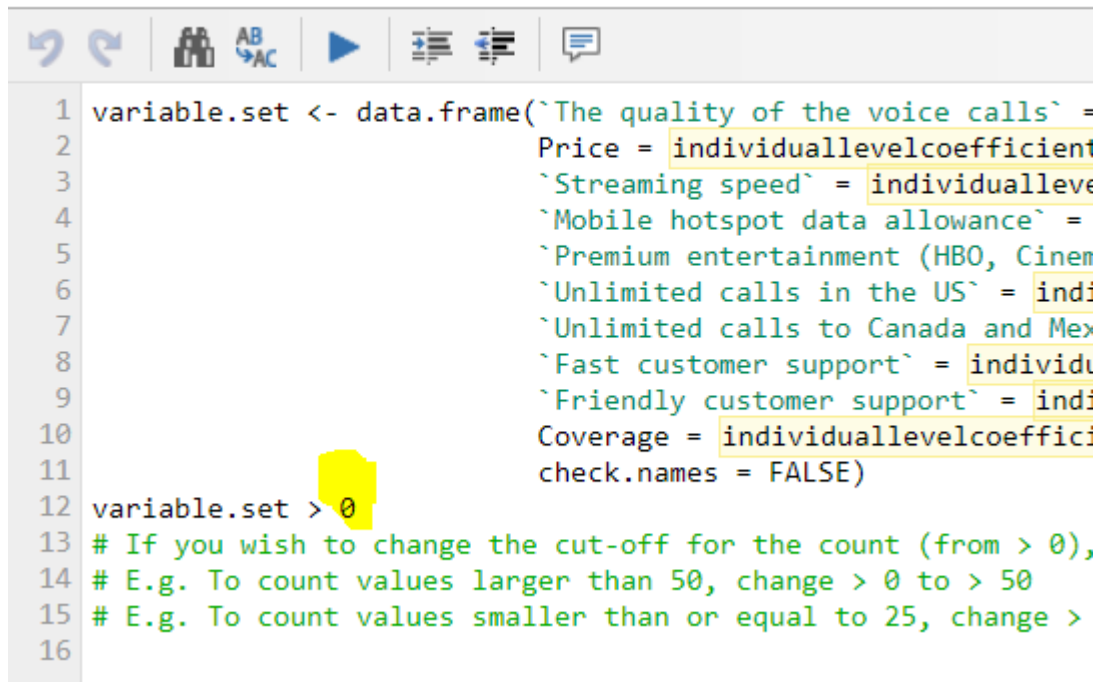
To create the 1s and 0s from rankings:

- Select the coefficients/utilities in the **Variables and Questions** tab.
- **Automate > Browse Online Library > Create new Variables > Scale Variables > Ranks Within Case.** You may have a few prompts to go through.
- Change the **Question Type** of the newly-created question to **Pick Any**.
- In the **Count This Value** column, choose the ranks you wish to have treated as 1s. For example, if there are 13 alternatives, and you want to use the two with the highest rank in the TURF, you would check on **Count This Value** for 12 and 13.

To implement the threshold approach,

- Select the coefficients/utilities in the **Variables and Questions** tab.
- **Automate > Browse Online Library > Create new Variables > Binary Variables.** This will automatically set values greater than 0 to 1, where with the coefficients/utilities, a 0 is the average.
- If you wish to use a threshold other than 0, right-click on one of the newly-created variables, select **Edit R Variable** and change the threshold (see the yellow in the screen shot below).

R CODE



```
1 variable.set <- data.frame(`The quality of the voice calls` =
2                             Price = individuallevelcoefficient
3                             `Streaming speed` = individualleve
4                             `Mobile hotspot data allowance` =
5                             `Premium entertainment (HBO, Cinem
6                             `Unlimited calls in the US` = indi
7                             `Unlimited calls to Canada and Mex
8                             `Fast customer support` = individu
9                             `Friendly customer support` = indi
10                            Coverage = individuallevelcoefficient
11                            check.names = FALSE)
12 variable.set > 0
13 # If you wish to change the cut-off for the count (from > 0),
14 # E.g. To count values larger than 50, change > 0 to > 50
15 # E.g. To count values smaller than or equal to 25, change >
16
```

To run the TURF, use **Automated > Browse Online Library > TURF > Total Unduplicated Reach and Frequency**. Please see our TURF eBook for more information.

Software - Displayr

To create the 1s and 0s from rankings:

- Select the coefficients/utilities in **Data Sets** tree.
- On the right of the screen, **Properties > Transformations > Scale Variables > Ranks Within Case**.
- Select **GENERAL > Structure: Binary – Multi**
- Press the **Select categories** button and choose the ranks you wish to have treated as 1s. For example, if there are 13 alternatives, and you want to use the two with the highest rank in the TURF, you would check on **Count This Value** for 12 and 13.

To implement the threshold approach,

- Select the coefficients/utilities in **Data Sets** tree.

- On the right of the screen, select **Properties > Transformations > Binary Variables**. This will automatically set values greater than 0 to 1, where with the coefficients/utilities, a 0 is the average.
- If you wish to use a threshold other than 0, expand out the variable set and click on the first variable, and then modify the threshold in the **R CODE** (see the yellow in the screen shot below).

The screenshot shows the TURF analysis software interface. On the left, the 'Data Sets' panel lists various attributes. The central panel displays a list of 10 variables with their corresponding utility scores and counts. The 'R CODE' panel on the right shows a code snippet where 'variable.set > 0' is highlighted in yellow, indicating the threshold for binary transformation.

To run the TURF, use **Anything > Advanced Analysis > TURF > TURF Analysis**. Please see our TURF eBook for more information.

The problems with TURF and MaxDiff

There are numerous ways of using MaxDiff outputs in a TURF analysis. Regardless of which approach you use, there is always a fundamental incompatibility between TURF and MaxDiff. The point of MaxDiff in market research is typically to find the smallest number of products or offers that appeals to the largest proportion of people. However, MaxDiff does not tell you anything about the appeal of products. It only tells you about relative appeal.

Consider a TURF study of preferences for Major League Baseball teams with the goal of working out which are the key teams for a fast food chain to sponsor. Further, assume that you do an international study without any screeners and use MaxDiff. There is a good chance that the Yankees will end up having the highest preference share from the MaxDiff, as the most well-known team. The TURF will then likely have the Yankees as the number one team to sponsor. However, it turns out that among American baseball teams the Yankees are the most hated,¹² so performing a TURF on the MaxDiff results will give you precisely the wrong answer.

¹² <https://fivethirtyeight.com/features/america-has-spoken-the-yankees-are-the-worst/>

Of course, if you were really trying to do a TURF analysis of baseball teams, you would be smart enough to filter the sample to exclude people who are not fans of the sport; but in a normal MaxDiff study you do not have this luxury. If you are evaluating alternatives that you know little about – which is usually why people do MaxDiff – you do not know which people dislike all of them and so run the risk of making precisely this type of mistake.

The solution to these problems is just to use simpler data. If you know that you need to perform a TURF, you will get a much better TURF if the input data is much simpler (e.g., a multiple-response question asking people which of the alternatives they like).

Anchored MaxDiff

Anchored MaxDiff experiments supplement standard MaxDiff questions with additional questions designed to work out the absolute importance of the attributes.

The table below shows the preference shares for a MaxDiff experiment in which the attributes being compared are technology brands.¹³

Probability %	Male	Female
Apple	8.87 +	11.33 +
Microsoft	8.78	10.20
IBM	4.64	5.35
Google	19.82	19.95
Intel	9.92 ↑	5.88 ↓
Hewlett-Packard	7.42 +	5.78 +
Sony	17.43	16.44
Dell	5.64	5.49
Yahoo	5.81	5.57
Nokia	11.67	14.03

Looking at the analysis we can see that:

- *Google* and *Sony* come in first and second place in terms of preference.
- *Apple* has done better among the women than the men.
- *Intel* and *Hewlett-Packard* have done relatively better among the men than the women.

Preference shares necessarily add up to 100%. Preference shares show relativities. Thus, while a naïve reading of the data would lead one to conclude that women like *Apple* more than men do, the data does not actually tell us this (i.e., it is possible that the men like every single brand more than the women do, but because the analysis is expressed as a percentage, such a conclusion cannot be obtained).

¹³ This has been set up as a **Ranking** question in Q. The preference shares are computed under the assumption of a single-class latent class model and are labeled as **Probability %** in Q and Displayr.

Coefficient	Male	Female
Apple	.00 ⁺	.00 ⁺
Microsoft	-.01	-.10
IBM	-.65	-.75
Google	.80	.57
Intel	.11 [↑]	-.66 [↓]
Hewlett-Packard	-.18 ⁺	-.67 ⁺
Sony	.68	.37
Dell	-.45	-.72
Yahoo	-.42	-.71
Nokia	.27	.21

The table on the left shows the same analysis but in terms of the coefficients. This is also uninformative, as these are indexed relative to the first brand in this case, which is *Apple*. Thus, that men and women both have a score of 0 is an assumption of the analysis rather than an insight (the color-coding is because the significance test is comparing the Preference Shares).

Anchored MaxDiff resolves this conundrum by using additional data as a benchmark. In the table on the right, a question asking likelihood to recommend each of the brands has been used to anchor the MaxDiff experiment. A rating of “Passive” 7 out of 10 by respondents has been used as a benchmark and assigned a coefficient of 0.¹⁴ All of the other coefficients are thus interpreted relative to this benchmark value. Thus, we can see that Apple has received a score of less than seven amongst both men and women and so, in some absolute sense, the brand is performing poorly (as a score of less than seven in a question asking about likelihood to recommend is typically regarded as a poor score). The analysis also shows that men have a marginally lower absolute score than women in terms of *Apple* (-0.68 versus -0.50), whereas *Google* has equal performance among the men and the women.

Coefficient	Male	Female
Passive (7)	.00	.00
Apple	-.68 ⁺	-.50 ⁺
Microsoft	-.59	-.50
IBM	-1.20	-1.21
Google	.26	.26
Intel	-.53 [↑]	-1.17 [↓]
Hewlett-Packard	-.81 ⁺	-1.09 ⁺
Sony	.09	-.07
Dell	-1.04	-1.16
Yahoo	-1.06	-1.11
Nokia	-.33 ⁺	-.19 ⁺

¹⁴ Specifying which coefficient is 0 is done in Q and Displayr by dragging the category (on the table) to be the first category.

Types of anchored MaxDiff experiments

There are two common types of *anchored* MaxDiff experiments: dual-response format, and MaxDiff combined with rating scales.

Dual-response format

The dual-response format involves following each MaxDiff question with another question asking something like:

Considering the four features shown above, would you say that...

- All are important
- Some are important, some are not
- None of these are important.

MaxDiff combined with rating scales

Before or after the MaxDiff experiment, the respondent provides traditional ratings (e.g., rates all the alternatives on a scale from 0 to 10 according to their likelihood of recommending each of the alternatives).

When to use anchored MaxDiff

When should you use anchored MaxDiff? It is not obvious to us that it is a useful technique (we discuss why below). It is included in this eBook for two reasons:

- Some people like it.
- One can use **Hybrid MaxDiff**, which is analyzed in the same way.

What is the problem with anchored MaxDiff? We use MaxDiff in situations where we feel that traditional questions will give poor data, either due to poor discrimination between respondents or due to yea-saying biases. However, anchored MaxDiff makes use of the same styles of question that MaxDiff was invented to avoid, so we end up with the worst of both worlds: we have all the problems of simple rating scales and all the complexity of MaxDiff.

Setting up the analysis of anchored MaxDiff in Q and Displayr

Anchored MaxDiff is most easily analyzed in Q and Displayr by setting up the data as a **Ranking** question in Q or structure in Displayr¹⁵. If doing this, hierarchical Bayes is not available, but similar models can be used instead.¹⁶

¹⁵ See https://wiki.q-researchsoftware.com/wiki/MaxDiff_Specifications

¹⁶ See https://wiki.q-researchsoftware.com/wiki/MaxDiff_Case_Study

The easiest way to think about anchored MaxDiff is as a *ranking with ties*. If a respondent was given a question showing options A, B, C, and F, and chose B as most preferred and F as least preferred, this means that: $B > A \approx C > F$.¹⁷

¹⁷ Alternatively, we could assume that the difference between the appeal of B versus either A and C is equal to the difference between either A and C versus F, as is implicit in the *tricked logit* model.

Setting up anchored MaxDiff data in Q and Displayr

Note that this is a different model from that used by Sawtooth Software.

When each of the choice questions is set up in Q¹⁸, a 1 is used for the most preferred item, a -1 for the least preferred item, 0 for the other items that were shown but not chosen and NaN for the items not shown. Thus, $B > A \approx C > F$ is encoded as:

A	B	C	D	E	F
0	1	0	NaN	NaN	-1

where the alternatives not shown are coded as NaN.

Note that when analyzing this data, Q only looks at the relative ordering, and any other values could be used instead, if they imply the same ordering.

Setting up the dual-response format anchored MaxDiff studies in Q

Anchoring is accommodated in Q by introducing a new alternative. In the case of the dual response, we will call this new alternative `zero`. Consider again the situation where the respondent has been faced with a choice of A, B, C, and F, and has chosen B as best and F as worst, which leads to $B > A \approx C > F$.

The `zero` alternative is always assigned a value of 0. The value assigned to the other alternatives is then relative to these zero values.

¹⁸ https://docs.displayr.com/wiki/Setting_Up_a_MaxDiff_Experiment_as_a_Ranking

All are important

Where all of the items are important this implies that all are ranked higher than the `Zero` option:

A	B	C	D	E	F	Zero
2	3	2	NaN	NaN	1	0

Some are important

Where some of the items are important this implies that the most preferred item must be more important than `Zero`, the least preferred item must be less preferred than `Zero`, but we do not know the relative preference of the remaining items relative to `Zero`, and thus this is coded as:

A	B	C	D	E	F	Zero
0	1	0	NaN	NaN	-1	0

Note that although this coding implies that $A = C = \text{Zero}$, the underlying algorithm does not explicitly assume these things are equal. Rather, it simply treats this set of data as not providing any evidence about the relative ordering of these alternatives.

None are important

A	B	C	D	E	F	Zero
-2	-1	-2	NaN	NaN	-3	0

Setting up the combined MaxDiff with ratings in Q

Prior to explaining how to use ratings to anchor the MaxDiff it is useful first to understand how ratings data can be combined with the MaxDiff experiment without anchoring. Again, keep in mind the situation where a MaxDiff task reveals that $B > A \approx C > F$. Consider a rating question where the six alternatives are rated, respectively, 9, 10, 7, 7, 7 and 3. Thus, the ratings imply that: $B > A > C \approx D \approx E > F$ and this information can be incorporated into Q as just another question in the MaxDiff experiment:

A	B	C	D	E	F
9	10	7	7	7	3

Anchoring is achieved by using the scale points. We can use some or all of the scale points as anchors. From an interpretation perspective it is usually most straightforward to choose a specific point as the anchor value. For example, consider the case where we decide to use a rating of 7 as the anchor point. We create a new alternative for the analysis which we will call *Seven*.

In the case of the MaxDiff tasks, as they only focus on relativities, they are set up in the standard way. Thus, where a MaxDiff question reveals that $B > A > C \approx D \approx E > F$, we include this new anchoring alternative, but it is assigned a value of NaN as nothing is learned about its relative appeal from this task.

A	B	C	D	E	F	Seven
-2	-1	-2	NaN	NaN	-3	NaN

The setup of the ratings data is then straightforward. It is just the actual ratings provided by respondents, but with an additional item containing the benchmark value:

A	B	C	D	E	F	Seven
9	10	7	7	7	3	7

Want to cut your analysis and reporting time in half?

See Displayr in action →

Analysis and reporting software built to save you time

