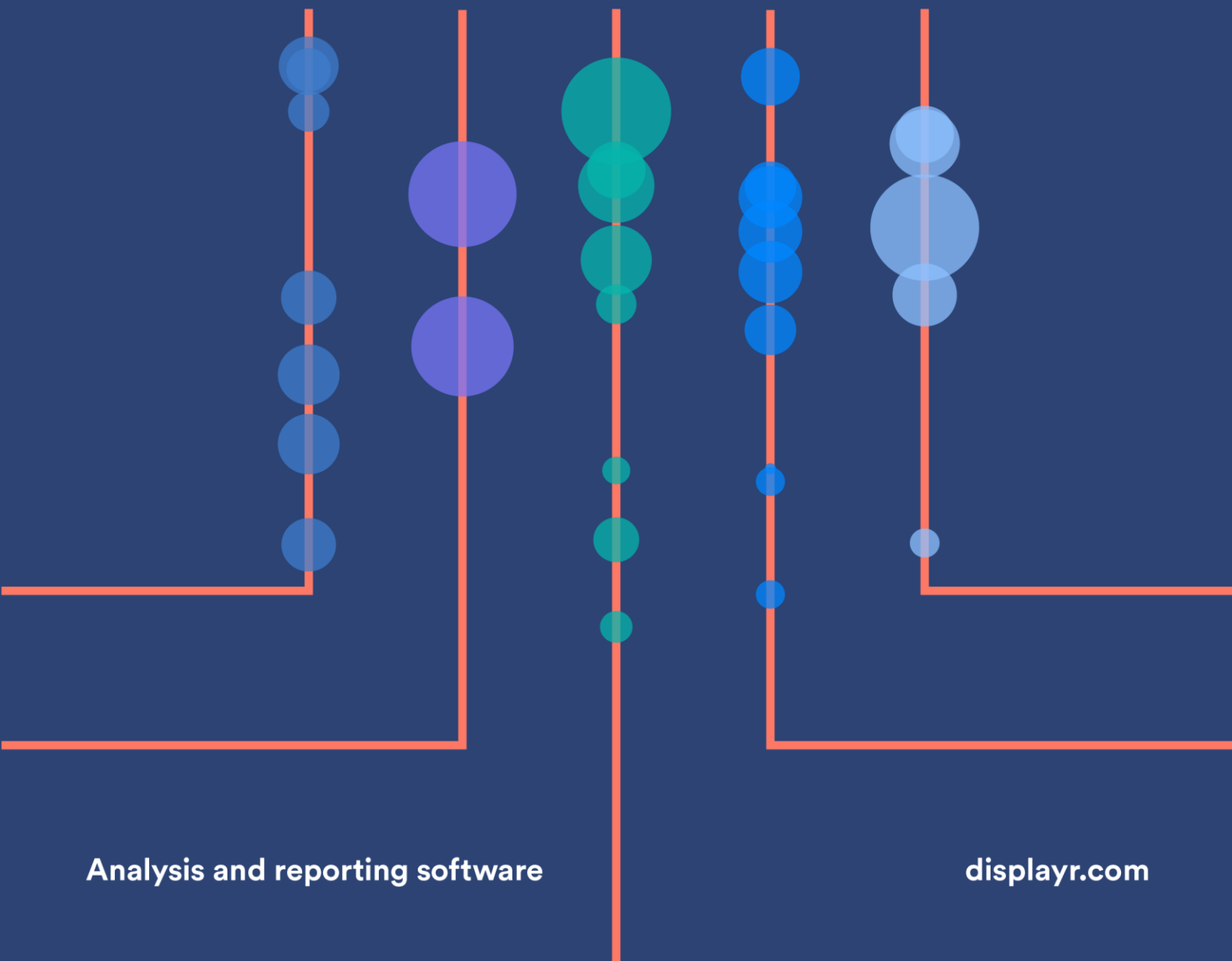
DISPLAYR

RESEARCHSOFTWARE
A DIVISION OF DISPLAYR

Market Segmentation

2nd edition



Analysis and reporting software

displayr.com

Table of Contents

WHO SHOULD READ THIS GUIDE 4

TECHNIQUES AND SOFTWARE 13

DATA PREPARATION 32

1. Don't knows and non-responses 33
2. Missing data due to not being asked (skips and filters) 38
3. Ordinal Variables 41
4. Nominal Variables 45
5. Nominal variables that have more than 3 categories 48
6. Questions with different ranges 50
7. Text data 53
8. No or few observations have complete data 57
9. Missing data can be predicted based on non-missing data 60
10. Data is missing for a really good reason (nonignorable) 65
11. Weights 68
12. MaxDiff, Conjoint, Driver Analysis, and Choice Data 70

FORMING SEGMENTS 72

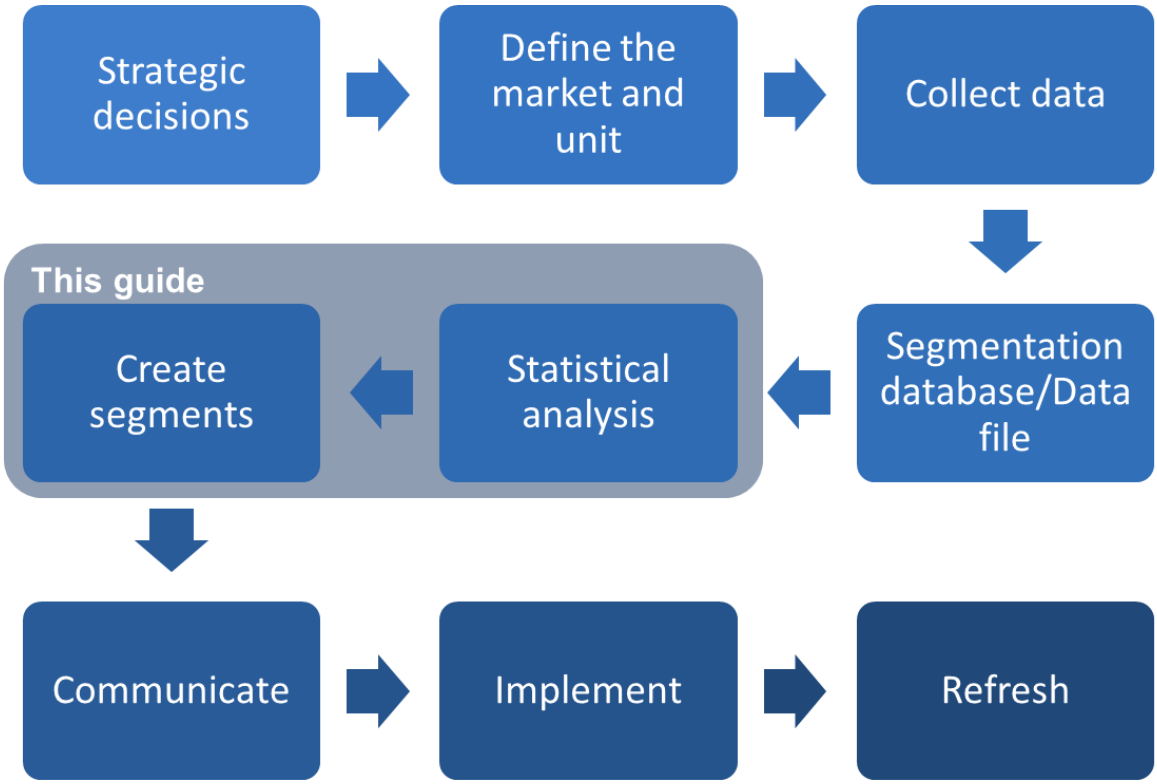
13. Segments that mainly reflect response biases 73
14. Weird results from latent class regression 77
15. Visualizing differences between segments 81
16. Segments differ on only a few variables 83
17. Identifying "golden questions" 87
18. Working out the best number of segments 90
19. Algorithm has not converged 95
20. Local optima 97
21. Recreating segments in a new data set 99

22. Segment typing tools 102
23. Measuring replicability 105
24. Increasing replicability 109
25. Quickly profiling segments 112
26. Increasing the predictability of the segments 117
27. Finding the most managerially useful segments 122

WHO SHOULD READ THIS GUIDE

> This guide provides a detailed set of instructions explaining how to create segments using survey data.
>
> A basic familiarity with surveys and market segmentation is assumed.

Segmenting a market involves nine phases. This guide is concerned with only two of them: statistical analysis and creating segments, although the other phases are discussed where relevant.



Inspirational versus operational segmentation

A key strategic decision when segmenting a market is whether to create an *inspirational segmentation* or an *operational segmentation.*

Inspirational segmentation is designed to inspire and educate stakeholders. Typical applications of inspirational segmentation include:

- Creating market segments that can be used as a framework for planning and strategy purposes.

- Creating customer archetypes to help service staff have empathy towards their customers.
- Creating personas to help technical staff understand how products and services are used.

The alternative to inspirational segmentation is *operational segmentation.* As with *inspirational segmentation,* operational segmentation involves dividing a market up into parts. However, it has an additional requirement that all customers are allocated into segments in real-time and treated differently based on their segment membership. Typically, this involves allocating customers in databases to segments. For example:

- Segmenting a customer list to send different campaigns to each customer.
- Building aircraft lounges for different customers.
- Prioritizing people in queues.

This guide focuses exclusively on inspirational segmentation (i.e., not on operational segmentation), although many of the principles discussed in this guide are applicable to both.

The starting point is data

This guide assumes you have already collected your data and are wanting to use it to create segments. To do this, your data will need to be in an appropriate form. Typically, a *raw data file* is required, which contains the distinct data points for each observation to be included in the analysis. Furthermore, all the techniques used for forming segments require that the data file is in a *flat*[1] format, where each row represents a person (or a person at a point in time; this is discussed in the next section) and each column represents some property of that person, which will typically be how they have

StartDate_date	Q002	Q003	Q004	Q005	Q005_2	Q006	Q007__1
28/09/2012 09:47:34	1		1	2	2		1
28/09/2012 09:45:35	6		1	2	2		0
28/09/2012 09:56:17	1		2	2	2		1
28/09/2012 09:56:37	6		2	2	2		1
28/09/2012 09:55:19	11		3	2	2		1
28/09/2012 10:01:04	6		1	2	2		1
28/09/2012 10:16:46	3		1	1	1	5	1
28/09/2012 09:55:15	3		1	1	1	1	1
28/09/2012 10:59:20	1		1	2	2		1
28/09/2012 10:53:39	1		2	2	2		1
28/09/2012 11:01:35	3		1	2	2		0
28/09/2012 11:11:37	3		1	1	1	2	1
28/09/2012 11:00:27	6		1	2	2		0
28/09/2012 11:15:41	1		1	2	2		1
28/09/2012 11:35:34	1		2	2	2		0
28/09/2012 11:18:11	3		1	1	1	2	1
28/09/2012 11:28:24	6		1	2	2		1
28/09/2012 11:35:35	1		1	2	2		1
28/09/2012 11:40:34	1		2	2	2		1
28/09/2012 11:36:16	6		1	2	2		1
28/09/2012 11:53:12	15		2	2	2		1
28/09/2012 11:38:11	1		1	2	2		1
28/09/2012 11:38:10	1		1	2	2		1
28/09/2012 11:26:03	15		2	2	2		1

[1] The main alternative to flat formats is that the data exists in a *relational database.* Such data needs to be converted to a flat file. Sometimes software used for forming segments will perform this process automatically (e.g., Q and Displayr automatically flatten the data in MDD databases).

answered a question, or part of a question, in a survey. The raw data file will typically look something like the table shown to the right.

A data file format with metadata

The raw data on its own is not enough. We also need *metadata.* This is best appreciated by looking at the raw data shown above. It is easy to work out what the first column shows. But what about the remaining columns? What does Q002 mean? And what does it mean if the person has a value of 6? The *metadata* is the information that explains what all the data means.

When performing segmentation, it is generally desirable to obtain your survey data in a rich file format, such as SPSS `.sav` and Triple S files, as these files contain both the raw data and the metadata and have also them hooked up together. 'Dumb' file formats like CSV files and Excel workbooks can also be used - but they add a lot more pain - as the metadata is not matched to up to the raw data.

Clustering versus judgment

Throughout this guide the focus is on using statistical methods that find *clusters,* also known as *classes,* in data. An alternative approach to segmentation is to use judgment.[2]

[2] Growing segmentation trees for strategic advantage. Timothy Bock and Chris Styles. Business Horizons, 2002, vol. 45, issue 2, 17-22. Date: 2002.

People versus people-in-situations

Segmentations of people or companies

Most segmentations are of people or companies. Data is collected from a person about their needs, wants, behaviors, or preferences, and this data is used to group people, or the companies that they represent, into segments.

Segmentations of people-in-situations

Sometimes, it makes sense for a person to be in multiple segments at different points of time. For example, needs when purchasing a beer at a bar may be different to those when purchasing beer to drink while watching TV. Such segmentations are performed by:

- Obtaining data from people at multiple points in time. Typically, this is done in a single survey, where they are asked to provide data about one or more recent occasions, transactions, or point-in-times.
- *Stacking* the data, so that each row in the data file represents a person at a specific point in time. For example, if we have asked people about what they ate for breakfast, lunch, and dinner, then we would have three rows of data per respondent. Our *DIY Driver Analysis* guide contains more detail about how to do this.
- Proceeding with the analysis as if conducting a segmentation of people. That is, the only difference in process relates to how the data is collected and organized. The actual process of forming the segments is the same regardless of whether the data is of people or people-in-situations.

What success looks like

The table to the right below is an example of the end-point of a basic inspirational segmentation. It describes four segments of toothpaste buyers. Typically, there will be a lot more information than shown here, both in terms of richer qualitative information and detailed tables of differences between segments.

		Segment Names (Toothpaste market)			
		Worrier	**Sociable**	**Sensory**	**Independent**
Who	*Demographics*	Large families 25-40	Teens Young smokers	Children	Males 35-50
	Psychographics	Conservative Hypochondriacs	Highly social Active	Self-involved Pleasure-seeking	Self-sufficient
What	*Main brand*	Crest	MacLeans Ultrabrite	Colgate	Supermarket brand
	Pack size preference	Large dispensers	Large	Medium	Small
	Price paid	Low	High	Medium	Low
Where	*Channel*	Supermarket	Supermarket	Supermarket	Neighborhood shops
Why	*Benefits sought*	Stop decay	Attract attention	Flavour	Functionality
Segment size		50%	30%	15%	5%
Potential for growth		Low	High	Medium	Low

Market segmentation is a strategic problem, not a statistical method

In our guides on driver analysis and MaxDiff we highlight how some techniques give the wrong results. The situation is different with segmentation. You can ignore all the advice in this guide and still produce a great segmentation (although you will be more likely to produce a great segmentation if you follow the instructions in this guide).

The 4 secrets of a successful segmentation

Ultimately, the secret to having successful segmentation studies tends to come down to the following:

1. Having good data. If your data is uninteresting, your segmentation will be uninteresting.
2. Not doing dumb things. This is largely about avoiding making technical mistakes. There are lots of easy mistakes if you do not know what to do, such as dealing with missing data, nominal variables, weights, and local optima, to name but a few.
3. Creating lots of alternative segmentations. There is no magical best segmentation. Success comes with looking at lots of segmentations. The more the better.
4. Systematically evaluating the alternative segmentations, with a focus on choosing the one that will be most useful to the stakeholders.

All but the first of these things is discussed in detail in this guide.

Software

This guide provides instructions for segmentation using Q and Displayr. Q is specifically designed for market segmentation, so it is used in most of these examples.

The structure of this guide

This guide breaks down the process of creating a segmentation into 25 issues. Eleven of the issues relate to data preparation and ten are issues that need to be addressed when forming segments.

Data preparation issues
1. Don't knows and non-responses
2. Skipped and filtered questions
3. Ordinal variables
4. Nominal variables
5. Nominal variables with >3 categories
6. Questions with different ranges
7. No observations have complete data
8. Missing data can be predicted using non-missing data
9. Data is missing for a really good reason
10. Weights
11. Using MaxDiff/conjoint/choice data

Issues addressed while forming segments
12. Yeah-saying bias in ratings
13. Scale effect in MaxDiff/conjoint/choice data
14. Visualizing differences between segments
15. Segments differ on only a few variables
16. Identifying "golden questions"
17. How many segments?
18. Algorithm has not converged
19. Local optima
20. Recreate segments in a new data set
21. Measuring replicability
22. Increasing replicability
23. Quickly profiling segments
24. Weak relationship between segments and other data
25. Choosing the final segmentation

Case study

This guide is mainly structured around a case study, looking at Americans' confidence in different institutions and the importance of different civic behaviors. The case study uses US data from 2014 collected by NORC at the University of Chicago. The data set that we will be using contains 3,842 cases and 350 variables. Our goal is to form a segmentation using two sets or ratings.

The data set used in this case study can be downloaded from here: http://wiki.q-researchsoftware.com/wiki/DIY_Advanced_Analysis. The original data set is here: http://gss.norc.org/Get-The-Data.

Confidence in institutions

The table below summarizes the data from the first rating scale (called *Confidence In* in the data file). Respondents were asked to rate how much confidence they had in 13 different institutions.

Row %	IAP	A GREAT DEAL	ONLY SOME	HARDLY ANY	DK	NA	NET
Banks & fin. institutions	33%	9% ↓	37% ↑	21% ↑	0% ↓	0% ↓	100%
Major companies	33%	11% ↓	42% ↑	12% ↓	1%	0%	100%
Organized religion	33%	12% ↓	37% ↑	16% ↓	1%	1% ↑	100%
Education	33%	16% ↑	39% ↑	12% ↓	0% ↓	0% ↓	100%
Federal govt	33%	8% ↓	29% ↓	29% ↑	1%	0%	100%
Organized labor	33%	7% ↓	38% ↑	19%	3% ↑	0%	100%
The Press	33%	5% ↓	31% ↓	30% ↑	1%	0%	100%
Medicine	33%	25% ↑	34%	7% ↓	0% ↓	0%	100%
Television	33%	6% ↓	33%	28% ↑	1% ↓	0%	100%
US supreme court	33%	14%	37% ↑	14% ↓	1% ↑	0%	100%
The scientific community	33%	27% ↑	33%	5% ↓	2% ↑	0%	100%
Congress	33%	3% ↓	25% ↓	37% ↑	1%	0%	100%
Military	33%	33% ↑	27% ↓	6% ↓	1% ↓	0%	100%

Importance of civic behaviors

The table below summarizes the importance ratings that the sample provided for nine civic behaviors (called *Importance* in the data file).

Row %	IAP	NOT AT ALL IMPORTANT	2	3	4	5	6	VERY IMPORTANT	CANT CHOOSE	NO ANSWER	NET
Always to vote in elections	67%	1%	1%	1%	2% ↓	3% ↓	4% ↓	20% ↑	1%	0%	100%
Never to try to evade taxes	67%	1% ↓	0% ↓	0% ↓	1% ↓	2% ↓	3% ↓	25% ↑	1%	0%	100%
Always to obey laws	67%	0% ↓	0% ↓	0% ↓	1% ↓	2% ↓	6%	22% ↑	0% ↓	0%	100%
Keep watch on action of govt	67%	1% ↓	0% ↓	1% ↓	2% ↓	3% ↓	6%	19% ↑	1%	0%	100%
Active in social/political associations	67%	2% ↑	1% ↑	3% ↑	7% ↑	8% ↑	5% ↓	5% ↓	1% ↑	0%	100%
Understand others' points of view	67%	1%	1%	1% ↓	3% ↓	5%	7% ↑	15%	0% ↓	0%	100%
Choose products for politics/ethics/envir.	67%	2% ↑	1% ↑	2% ↑	6% ↑	7% ↑	7% ↑	7% ↓	2% ↑	0%	100%
Help worse off people in America	67%	0% ↓	0% ↓	1% ↓	3%	6% ↑	8% ↑	14%	0% ↓	0%	100%
Help worse off people in rest of World	67%	2% ↑	2% ↑	3% ↑	6% ↑	7% ↑	4% ↓	7% ↓	1%	0%	100%

TECHNIQUES AND SOFTWARE

> This chapter provides an overview of the main techniques (algorithms) used to form segments.

This chapter reviews the main techniques available for finding segments in data. As explained below, and then in more detail throughout this guide, *latent class analysis* is the best technique. However, this does not mean the other techniques are without any merit. Ultimately, an inspirational segmentation should be judged by its usefulness in inspiring its users, and there is no guarantee that the segmentation created using the state-of-the-art techniques will be best in this regard.

Hierarchical clustering

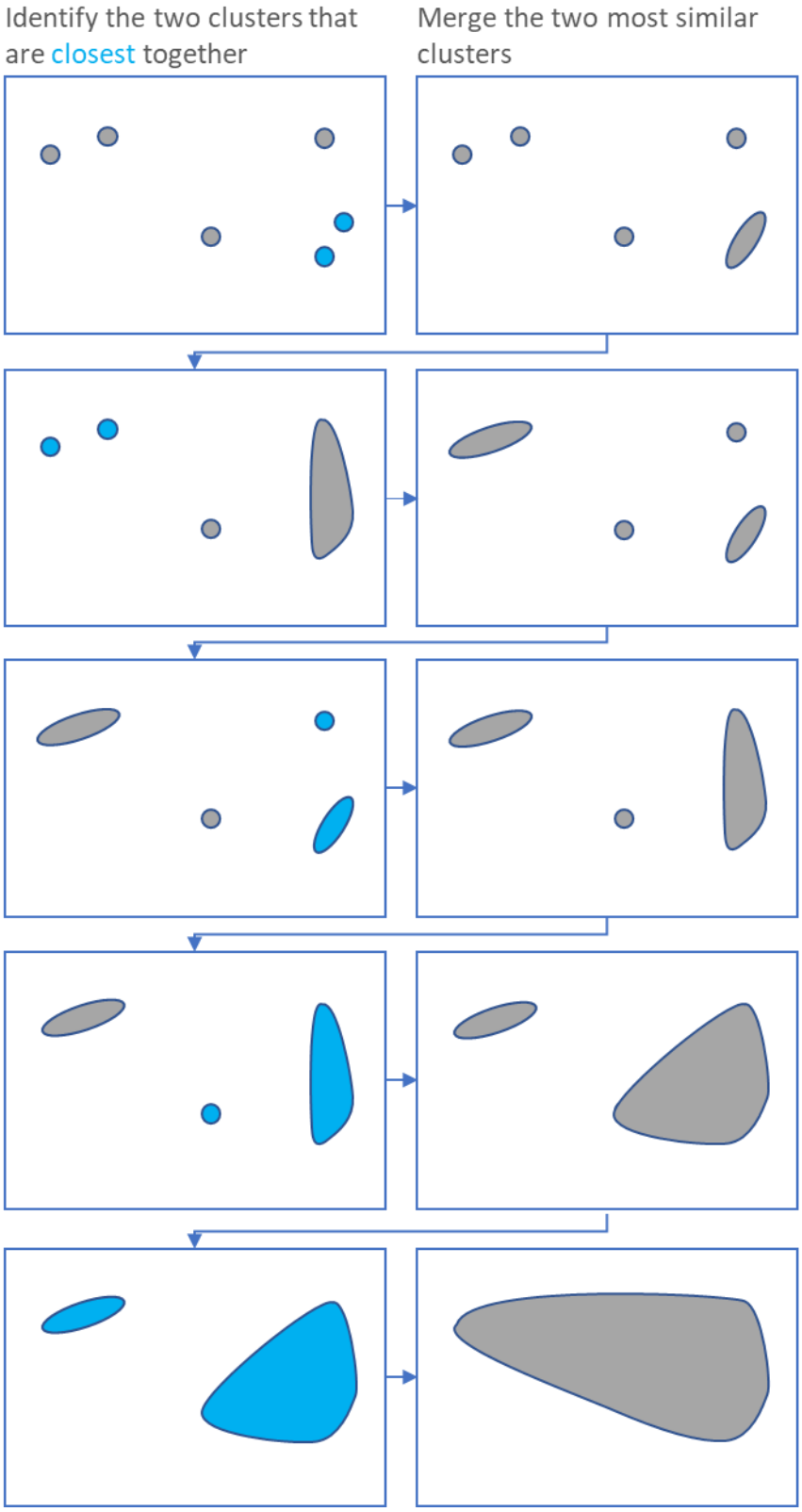
Hierarchical clustering, also known as *hierarchical cluster analysis,* is an algorithm that groups similar objects into groups called *clusters*. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

Hierarchical clustering can be performed with either a *distance matrix* or *raw data.* When raw data is provided, the software will automatically compute a distance matrix in the background. The distance matrix below shows the distance between six objects.

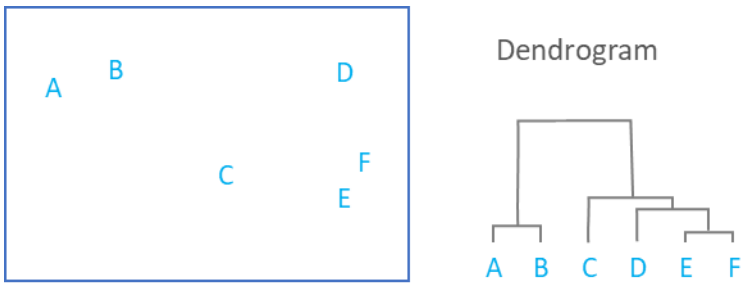
	A	B	C	D	E
B	16				
C	47	37			
D	72	57	40		
E	77	65	30	31	
F	79	66	35	23	10

How hierarchical clustering works

Hierarchical clustering starts by treating each observation as a separate cluster. Then, it repeatedly executes the following two steps: (1) identify the two clusters that are closest together, and (2) merge the two most similar clusters. This continues until all the clusters are merged together. This is illustrated in the diagrams on the next page.


Identify the two clusters that are closest together
Merge the two most similar clusters

The main output of Hierarchical Clustering is a *dendrogram,* which shows the hierarchical relationship between the clusters:



Measures of distance (similarity)

In the example above, the *distance* between two clusters has been computed based on length of the straight line drawn from one cluster to another. This is commonly referred to as the *Euclidean distance.* Many other *distance metrics* have been developed.

The choice of distance metric should be made based on theoretical concerns from the domain of study. That is, a distance metric needs to define similarity in a way that is sensible for the field of study. For example, if clustering crime sites in a city, city block distance may be appropriate (or, better yet, the time taken to travel between each location). Where there is no theoretical justification for an alternative, the Euclidean is generally preferred, as it is usually the most appropriate measure of distance in the physical world.

Linkage Criteria

After selecting a distance metric, it is necessary to determine from where distance is computed. For example, it can be computed between the two most similar parts of a cluster (*single-linkage*), the two least similar bits of a cluster (*complete-linkage*), the center of the clusters (*mean* or *average-linkage*), or some other criterion. Many linkage criteria have been developed.

As with *distance metrics,* the choice of linkage criteria should be made based on theoretical considerations from the domain of application. A key theoretical issue is what causes variation. For example, in archeology, we expect variation to occur through innovation and natural resources, so working out if two groups of artifacts are similar may make sense based on identifying the most similar members of the cluster.

Where there are no clear theoretical justifications for choice of linkage criteria, *Ward's method* is the sensible default. This method works out which observations to group based on reducing the sum of squared distances of each observation from the average observation in a cluster. This is often

appropriate as this concept of distance matches the standard assumptions of how to compute differences between groups in statistics (e.g., *ANOVA*, *MANOVA*).

The problems with hierarchical cluster analysis

The strengths of hierarchical clustering are that it is easy to understand and easy to do. The weaknesses are that it rarely provides the best solution, it involves lots of arbitrary decisions, it does not work with missing data, it works poorly with mixed data types, it does not work well on very large data sets, it is not applicable with weighted data, and its main output, the dendrogram, is commonly misinterpreted. There are better alternatives, such as *latent class analysis.*

Q

Hierarchical clustering is available in Q using **Create > Segments > Hierarchical Cluster Analysis**. This automatically treats the variables as the cases, so does not produce segments in the traditional sense (e.g., it is used for creating segments of brands, rather than segments of people). If you want to cluster cases together, use **Create > R Output** and the `hclust` function in base R.

For example, to create a four-cluster hierarchical clustering using Ward's method:

- Go to the **Variables and Questions** tab
- Right-click on a variable
- **Insert Variable(s) > R Variable > Numeric**
- Paste in the following code, swapping `Q050__1, Q050__2, Q050__3, Q050__4, Q050__5` with your variable names:

```
DF = data.frame(Q050__1, Q050__2, Q050__3, Q050__4, Q050__5)
distance.matrix <- dist(data.matrix(DF), method = "euclidean")
hc <- hclust(distance.matrix, "ward.D2")
cutree(hc, 4)
```

Displayr

Hierarchical clustering is available in Displayr using **Anything > Advanced Analysis > Cluster > Hierarchical Cluster Analysis**. This automatically treats the variables as cases, so does not produce segments in the traditional sense (e.g., it is used for creating segments of brands, rather than segments of people). If you want to cluster cases together, use **Insert > R Output** and the `hclust` function in base R.

For example, to create a four-cluster hierarchical clustering using Ward's method:

- Hover over the **Data sets** tree and click the + to insert a variable
- **Custom Code > R – Numeric**

- Paste in the following code, swapping `Q050__1, Q050__2, Q050__3, Q050__4, Q050__5` with your variable names:

```
DF = data.frame(Q050__1, Q050__2, Q050__3, Q050__4, Q050__5)
distance.matrix <- dist(data.matrix(DF), method = "euclidean")
hc <- hclust(distance.matrix, "ward.D2")
cutree(hc, 4)
```

k-means cluster analysis

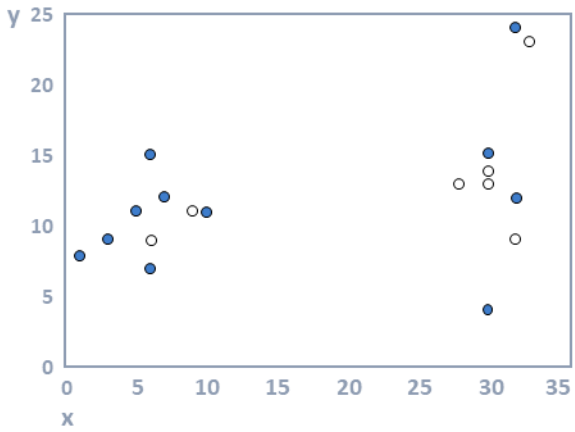
In the 1960 and 1970s newer techniques which took advantage of improvements in computer speed were developed for finding clusters, the most well-known of which is *k*-means. *k*-means works as follows:

Step 1: Specify the number of clusters (*k*)

The first step in *k*-means is to specify the number of clusters, which is referred to as *k*. Traditionally researchers will conduct *k*-means multiple times, exploring different numbers of clusters (e.g., from 2 through 10).

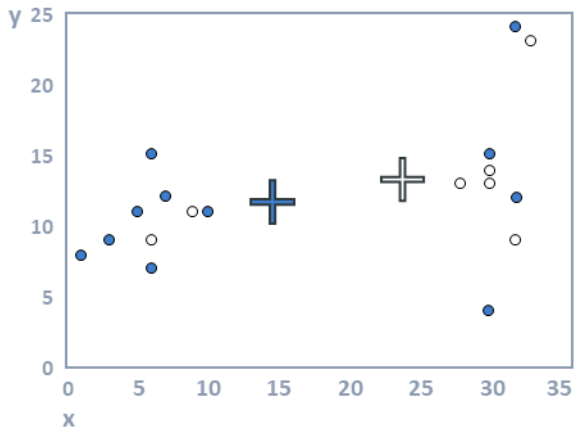
Step 2: Allocate observations to clusters

The most straightforward approach is to randomly assign observations to clusters, but there are many other approaches (e.g., using *hierarchical clustering*). In the diagram below, the 18 observations have been represented by dots on a *scatterplot,* where **x** is shown by the horizontal position of each **object** and **y** by the vertical. The objects have been randomly assigned to the two clusters (*k* = 2), where one cluster is shown with filled dots and the other with unfilled dots.



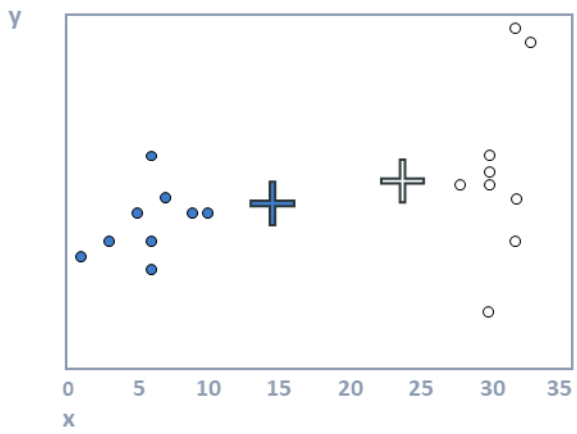
Step 3: Compute cluster means

For each cluster, the average value is computed for each of the variables. In the plot below, the average value of the filled dots for the variable represented by the horizontal position (x) of the dots is around 15; for the variable on the vertical dimension it is around twelve. These two means are represented by the filled cross. Or, stated slightly differently: the filled cross is in the middle of the black dots. Similarly, the white cross is in the middle of the white dots. These crosses are variously referred to as the *cluster centers, cluster means,* and *cluster centroids*.



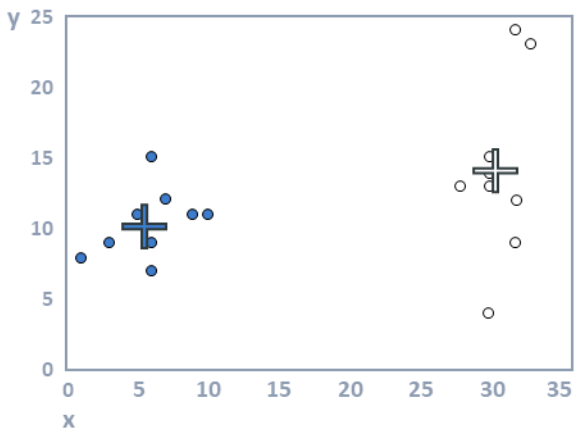
Step 4: Allocate each observation to the closest cluster center

In the plot above, some of the filled dots are closer to the white cross and some of the white dots are closer to the black cross. When we reallocate the observations to the closest clusters we get the plot below.



Step 5: Repeat steps 3 and 4 until the solution converges

Looking at the plot above, we can see that the crosses (the cluster means) are no longer accurate. In the following plot they have been recomputed using step 3. In this example the cluster analysis has *converged* (i.e., reallocating observations and updating means cannot improve the solution). In examples with more data a few more iterations are typically required (i.e., steps 3 and 4 are repeated until no respondents change clusters).



Variations of *k*-means

There are several different *k*-means algorithms. The one described above is known as the *batch algorithm* and is the default in SPSS (**Analyze > Classify > K-Means Cluster Analysis**) and in Q's **K-means Cluster Analysis**.

A common variant is to instead allocate the first observation to the most similar cluster, then update the means, then allocate the second respondent to the most similar cluster, then update the means, etc.

Outliers

If there are truly extreme observations, *k-means* will allocate them to their own clusters. Thus, if you have no very small clusters when you use *k-means*, it means you do not have any serious outliers/extreme observations. If you do end up with small clusters you can investigate the data and decide whether to ignore this data from further analyses. This approach is both easier than using special-purpose robust algorithms and more rigorous, as human judgment and insight is used to address outliers (rather than them being automatically addressed in the background).

The next two techniques are *robust* variants of k-means, designed to prevent the cluster analysis from having problems due to outliers/extreme observations. Although the ideas sound intuitive appealing, there is little evidence that they are in practice better.

Q

Q has two alternative ways of performing *k*-means cluster analysis.

Create > Segments > K-Means Cluster Analysis uses batch *k*-means that works with incomplete data and weights. Once you have created the analysis, you need to then click on it and press **Inputs > SAVE VARIABLE(S) > Cluster Membership** to create a new variable that indicates which respondent is in which segment.

Create > Segments > Legacy Cluster Analysis implements a traditional *k*-means algorithm that accommodates neither weights nor missing data, but has automated tools for selecting the number of clusters and computing replicability (see **Working out the best number of segments** and **Measuring replicability**). It automatically creates a new variable indicating segment membership.

Displayr

Anything > Advanced Analysis > Cluster > K-Means Cluster Analysis creates a *batch k-means* that works with incomplete data and weights. Once you have created the analysis, you need to then click on it and press **Inputs > SAVE VARIABLE(S) > Cluster Membership** to create a new variable that indicates which respondent is in which segment.

k-medoids

An alternative to *k*-means is to use algorithms that are less sensitive to extreme values. Whereas *k*-means minimizes the sum of squared differences between observations and the closest cluster, *k-medoids* minimizes the sum of differences. This means that observations further from the center of a cluster become less influential. Survey data tends not to contain many extreme observations due to the use of question types with limited choices, so the benefit of this approach tends not to compensate for its disadvantage (using an obscure technique that is less well understood than *k*-means).

Software

In Q and Displayr, *k-medoids* is available via the `pam` function in the `cluster` R package.

Trimmed k-means

Another *robust* alternative to *k*-means is to use *trimmed k-means,* which essentially works in the same way as *k-means,* except that it excludes a percentage of the most extreme values when computing the means (e.g., the 10% of values that are furthest from the mean).

In addition to suffering from the same problems as *k-medoids* it can lead to serious problems in reporting. This is because once the clusters have been found, it is natural to want to assign all the observations in the data to a cluster, but *trimmed k-means* assumes you are happy to ignore, say, 10% of the respondents from all future reporting. This can be difficult to justify to more diligent stakeholders who will question the usefulness of a segmentation that is not exhaustive (i.e., where some people are not allocated to segments).

Software

In Q and Displayr, trimmed *k*-means is available via the `tkmeans` function in the `tclust` R package.

Mixed-mode cluster analysis

Mixed-mode cluster analysis is an improvement over *k*-means in terms of its ability to include non-numeric data. The simplest implementations allow a mixture of numeric and nominal (categorical) variables.

Mixed-mode cluster analysis is a simplification of *latent class analysis* (described below). The simplification occurs in two ways. First, mixed-mode cluster analysis makes a technical assumption of constant priors for class sizes. That is, it groups people to the cluster that they are closest to, whereas latent class analysis groups people based both on their similarity to the clusters and the size of the clusters (e.g., if one person is equally close to two clusters, they are assigned to the larger cluster). Second, mixed-mode cluster analysis assumes people are in exactly one cluster, whereas latent class analysis allows for people to be partially in multiple clusters. On face value these assumptions do not sound particularly troubling and may even appear desirable. However, for reasons well beyond the

scope of this guide the consequence of these two assumptions is that mixed-mode cluster analysis is likely more susceptible to *local optima.*[3]

Q

Create > Segments > Mixed-Mode Cluster Analysis. A variable indicating segment membership is automatically created when the analysis completes.

Displayr

Anything > Advanced Analysis > Cluster > Mixed-Mode Cluster Analysis. A variable indicating segment membership is automatically created when the analysis completes.

Neural networks

The main neural network algorithm that can be used for market segmentation is the *self-organizing map (SOM).* The principle benefit of the self-organizing map is that its outputs can be useful from a visualization purpose. Although it can be used for market segmentation, the underlying math can easily be shown to be the same as *k*-means when used for segmentation if you use the technique properly, and worse than *k*-means otherwise.[4]

---

[3] Latent Class Analysis is estimated using the *EM Algorithm* which is known, in certain situations, to be guaranteed to achieve a global maximum (i.e., not fail to find the best solution). No such proofs exist for Mixed-Mode cluster analysis nor are they likely to exist, as the consequences of these assumptions is that the objective function is not smooth and inconsistent with a mathematical law (Bayes' Theorem).

[4] The self-organizing map algorithm is like *k-means,* but where the clusters are permitted to overlap, the degree of overlap reduces during the algorithm, and the overlap is controlled by a grid. When the overlap is reduced to 0 by the end of the algorithm it just becomes a variant of *k-means* known as of *Batch k-means*. If the overlap is not left at 0, it becomes automatically 0 anyway when observations are assigned to the most similar cluster, the consequence of which is that people are assigned to clusters that are inevitably local optima. See A new approach for exploring multivariate data: self-organizing maps. Timothy Bock. International Journal of Market Research. 1 April 2004. Vol. 46 No. 2, 2004 p.189–203.

Latent class analysis

Latent class analysis works in essentially the same way as *k*-means cluster analysis,[5] except that:

- It is *mixed-mode* so it can be used with non-numeric data. Segments can be created with a combination of numeric data and categorical data. Furthermore, it can be used with MaxDiff, Conjoint, Driver Analysis, and Choice Modeling data sets without needing to first compute respondent-level coefficients (Q's mixed-mode cluster analysis can deal with these exotic forms of data).
- Most implementations of latent class analysis can deal with weights, whereas most implementations of *k*-means cannot (both SPSS's *k*-means cluster analysis and Q's *k*-means cluster analysis do deal with weights).
- Most implementations of latent class analysis are smarter in how they deal with missing values than are most implementations of *k*-means (this is discussed below in Error! Reference source not found. and **Missing data can be predicted** based on non-missing data).
- The underlying math is a bit smarter, so it tends to get better results all else being equal (see the discussion above in **Mixed-mode cluster analysis)**.

These benefits have led to latent class analysis becoming the gold-standard for forming segments. However, it does have a few limitations:

- It is a lot slower than *k*-means cluster analysis. This makes it impractical for huge databases and this can lead to problems with convergence and local optima (see **Algorithm has not converged** and **Local optima**).
- Experienced users of *k*-means have various tricks for tweaking the algorithm by transforming variables. With numeric data the same tweaks will work, but they do not work for more exotic data such as MaxDiff, *choice modeling, conjoint*, and *driver analysis* (see **Weird results from latent class regression**).
- The latent class analysis algorithm does not assign each respondent to a class. Instead, it computes a probability that a respondent will be in a class. Then, at the end of the analysis, observations are assigned to the segment for which they have the highest probability. This leads to two different ways of computing the sizes of the segments and the mean values of

[5] That is, batch *k-means* is a simplification of the *EM-Algorithm* used to fit *Latent Class Analysis* models.

each class. For example, the segment sizes can be computed as the average of the probabilities, or, by counting the number of people assigned to each class. These computations often lead to slightly different answers (and can lead to big difference occasionally, but this is usually a sign that the segmentation is problematic anyway). This can cause confusion. The best approach is usually to only show stakeholders computations based on the variable that has assigned each person to the most similar class.[6]

Latent class analysis in Q

The most general approach is to use **Create > Segments > Latent Class Analysis**. A variable indicating segment membership is automatically created when the analysis completes.

This algorithm can also perform **Latent class regression**, and jointly analyze data using both traditional latent class analysis and latent class regression at the same time (e.g., if jointly analyzing traditional rating scales and a conjoint study).

Latent class analysis in Displayr

The most general approach is to use **Anything > Advanced Analysis > Cluster > Latent Class Analysis**. This automatically creates a variable indicating segment membership for each respondent at the bottom of the data set.

This algorithm can also perform **Latent class regression**, and jointly analyze data using both traditional latent class analysis and latent class regression at the same time (e.g., if jointly analyzing traditional rating scales and a conjoint study).

[6] Here "best" means most pragmatic.

Latent class regression

Most latent class models are conceptually very similar to cluster analysis. They identify segments of people that differ in either their average ratings of numeric variables, or, their preferences for particular choices in nominal variables.

Special-purpose latent class algorithms have been developed for analyzing MaxDiff, conjoint/choice, and regression-type data in general (e.g., key driver analysis). The algorithms are sometimes referred to as *latent class analysis* and other times as *latent class regression,* as they simultaneously estimate segments and estimate regression models within these segments.

These algorithms take the raw data and experimental designs (where appropriate) as inputs, and provide as outputs:

- Descriptions of differences between the segments (e.g., in terms of utilities).
- Variables that allocate people to the most appropriate segment.

The next few sections describe the basics of how to access these techniques. More detailed information is available in our eBooks (see https://www.qresearchsoftware.com/ebooks).

Latent class analysis for MaxDiff in Q

- **Create > Marketing > MaxDiff > Latent Class Analysis**. To obtain a variable that contains segment membership, click on the output and select **Inputs > SAVE VARIABLE(S) > Class Membership**.
- **Create > Segments > Latent Class Analysis** will perform the analysis properly provided that the data is set as **Question Type** of **Ranking** or **Experiment**. This will automatically create a variable in the data file that stores segment membership.

Latent class analysis for choice-based conjoint in Q

- **Automate > Browse Online Library > Choice Modeling > Latent Class Analysis**. To obtain a variable that contains segment membership, click on the output and select **Inputs > SAVE VARIABLE(S) > Class Membership**.
- **Create > Segments > Latent Class Analysis** will perform the analysis properly provided that the data is set as **Question Type** of **Ranking** or **Experiment**. This will automatically create a variable in the data file that stores segment membership.

Latent class analysis for driver analysis in Q

Create > Segments > Latent Class Analysis will perform the analysis properly provided that the data is set as **Question Type** of **Experiment.** Refer to the discussion of **Metric Conjoint Analysis** in https://wiki.q-researchsoftware.com/wiki/Experiments_Specifications for the specific details of how the data needs to be set up.

Latent class analysis for MaxDiff in Displayr

- **Anything > Advanced Analysis > MaxDiff > Latent Class Analysis**. To obtain a variable that contains segment membership, click on the output and select **Inputs > SAVE VARIABLE(S) > Class Membership**.
- **Anything > Advanced Analysis > Cluster > Latent Class Analysis** will perform the analysis properly provided that the data is set with the **Structure** of **Ranking** or **Experiment**. This will automatically create a variable in the data file that stores segment membership.

Latent class analysis for choice-based conjoint in Displayr

- **Anything > Advanced Analysis > Choice Modeling > Latent Class Analysis**. To obtain a variable that contains segment membership, click on the output and select **Inputs > SAVE VARIABLE(S) > Class Membership**.
- **Anything > Advanced Analysis > Cluster > Latent Class Analysis** will perform the analysis properly provided that the data is set with the **Structure** of **Ranking** or **Experiment**. This will automatically create a variable in the data file that stores segment membership.

Latent class analysis for driver analysis in Displayr

Anything > Advanced Analysis > Cluster > Latent Class Analysis will perform the analysis properly provided that the data is set with the **Structure** of **Experiment.** Refer to the discussion of **Metric Conjoint Analysis** in https://wiki.q-researchsoftware.com/wiki/Experiments_Specifications for the specific details of how the data needs to be set up.

Hierarchical Bayes

Hierarchical Bayes (HB) is a technique for analyzing MaxDiff, choice-based conjoint, and driver analysis data. Its main output is a set of *utilities* for each respondent in the data file, which can then be analyzed using traditional cluster analysis and latent class software.

More information about how to run this technique is in our eBook on MaxDiff (see https://www.qresearchsoftware.com/ebooks).

Dimension reduction techniques

Dimension reduction techniques are often used in the data preparation stage of segmentation. The two most widely used of these techniques prior to market segmentation are probably principal components analysis (PCA) and multiple correspondence analysis (MCA), but there are many other approaches.

PCA is applicable to numeric data. If being analyzed with categorical data, you whould typically create *dummy variables* first. MCA is applicable to categorical data.

This approach, which is also known as *tandem clustering,* is popular but has some serious issues:

1. A lot of information is lost. For example, if components that explain 50% of the variance are used in the segmentation, then 50% of the data has been ignored.
2. It transforms the dimensional space, giving emphasis to variables that are less correlated with other variables (e.g., if 10 variables are highly correlated, they are all replaced with a single component). This can cause very strong patterns in the data to be overlooked. The flip side is that the strong patterns can be obvious and non-interesting, and the automatic reweighting of the variables that occurs can sometimes be a virtue, as it can lead to the identification of very different segments. As discussed later in this guide, it is useful to generate lots of different segmentations, and this approach is useful when done in conjunction with other approaches.

Principal components analysis in Q

1. **Create > Dimension Reduction > Principal Components Analysis**
2. Select the **Variables** to be analyzed
3. Check **Categorical as binary** (if you have ordinal variables, make sure you first change their **Question Type** to **Number** or **Number – Multi**).
4. Select **Inputs > SAVE VARIABLE(S) > Components/Dimensions**

Principal components analysis in Displayr

1. **Anything > Advanced Analysis > Dimension Reduction > Principal Components Analysis**
2. Select the **Variables** to be analyzed
3. Check **Categorical as binary** (if you have ordinal variables, make sure you first change their **Structure** to **Numeric** or **Numeric – Multi**).
4. Select **Inputs > SAVE VARIABLE(S) > Components/Dimensions**

Multiple correspondence analysis in Q

1. **Create > Dimension Reduction > Principal Components Analysis**
2. Select the **Input variables** to be analyzed
3. **Create > Dimension Reduction > Principal Components Analysis > Save variable(s)**

Multiple correspondence analysis in Displayr

1. **Anything > Advanced Analysis > Dimension Reduction > Multiple Correspondence Analysis**
2. Select the **Input variables** to be analyzed
3. **Anything > Advanced Analysis > Dimension Reduction > Principal Components Analysis > Save variable(s)**

Predictive models

Predictive modeling techniques are useful at various stages of the segmentation process (e.g., when **Increasing the predictability of the segments** and **Recreating segments in a new data set**).

Three predictive techniques that quite often work well on segmentation-related problems are:

- *Support vector machine.*
- *Random forest*.
- *Linear discriminant analysis (LDA)*.

The first two of these are relatively modern machine learning algorithms. LDA, by contrast, is a very old multivariate statistical technique, which tends to work well because its underlying assumptions are very similar to those of cluster analysis and latent class analysis.

Predictive models in Q

Press **Automate > Browse Online Library > Machine Learning > Classifier** and then choose the desired model (e.g., **Random Forest**). You can change between different models using the **Algorithm** dropdown.

Predictive models in Displayr

Press **Anything > Advanced Analysis > Machine Learning** and then choose the desired model (e.g., **Random Forest**). You can change between different models using the **Algorithm** dropdown.

DATA PREPARATION

1. Don’t knows and non-responses

Most of the techniques in use today for forming segments are not able to deal with *don’t know, refused, not applicable,* and other forms of item non-response.

It is usually best to treat them as numeric variables.

Most of the techniques used for creating segments are designed only to deal with numeric data. If there are *don't know* or *not answered* responses, these techniques cannot be used without modifying the data.

Best approach: Recode as missing values

The simplest approach to dealing with don't know and not answered categories is to recode them as missing values.

The table below shows data on the confidence that people have in various American institutions. In this case there are three categories that need to be set to missing values: *IAP* (inapplicable), *DK,* and *NA.*

Row %	IAP	A GREAT DEAL	ONLY SOME	HARDLY ANY	DK	NA	NET
Banks & fin. institutions	33%	9% ↓	37% ↑	21% ↑	0% ↓	0% ↓	100%
Major companies	33%	11% ↓	42% ↑	12% ↓	1%	0%	100%
Organized religion	33%	12% ↓	37% ↑	16% ↓	1%	1% ↑	100%
Education	33%	16% ↑	39% ↑	12% ↓	0% ↓	0% ↓	100%
Federal govt	33%	8% ↓	29% ↓	29% ↑	1%	0%	100%
Organized labor	33%	7% ↓	38% ↑	19%	3% ↑	0%	100%
The Press	33%	5% ↓	31% ↓	30% ↑	1%	0%	100%
Medicine	33%	25% ↑	34%	7% ↓	0% ↓	0%	100%
Television	33%	6% ↓	33%	28% ↑	1% ↓	0%	100%
US supreme court	33%	14%	37% ↑	14% ↓	1% ↑	0%	100%
The scientific community	33%	27% ↑	33%	5% ↓	2% ↑	0%	100%
Congress	33%	3% ↓	25% ↓	37% ↑	1%	0%	100%
Military	33%	33% ↑	27% ↓	6% ↓	1% ↓	0%	100%

The importance data is shown below. The categories to set to missing here are *IAP, CAN'T CHOOSE,* and *NO ANSWER.*

Row %	IAP	NOT AT ALL IMPORTANT	2	3	4	5	6	VERY IMPORTANT	CANT CHOOSE	NO ANSWER	NET
Always to vote in elections	67%	1%	1%	1%	2%↓	3%↓	4%↓	20%↑	1%	0%	100%
Never to try to evade taxes	67%	1%↓	0%↓	0%↓	1%↓	2%↓	3%↓	25%↑	1%	0%	100%
Always to obey laws	67%	0%↓	0%↓	0%↓	1%↓	2%↓	6%	22%↑	0%↓	0%	100%
Keep watch on action of govt	67%	1%↓	0%↓	1%↓	2%↓	3%↓	6%	19%↑	1%	0%	100%
Active in social/political associations	67%	2%↑	1%↑	3%↑	7%↑	8%↑	5%↓	5%↓	1%↑	0%	100%
Understand others' points of view	67%	1%	1%	1%↓	3%↓	5%	7%↑	15%	0%↓	0%	100%
Choose products for politics/ethics/envir.	67%	2%↑	1%↑	2%↑	6%↑	7%↑	7%↑	7%↓	2%↑	0%	100%
Help worse off people in America	67%	0%↓	0%↓	1%↓	3%	6%↑	8%↑	14%	0%↓	0%	100%
Help worse off people in rest of World	67%	2%↑	2%↑	3%↑	6%↑	7%↑	4%↓	7%↓	1%	0%	100%

Once you have recoded your categories as missing values, you then need to make sure that you address them appropriately. This is discussed in Error! Reference source not found., **Missing data can be predicted** based on non-missing data, and **Data is missing** for a really good reason.

Recode as missing values in Q

Create a table of data, right-click on the row or column heading that corresponds to the category you wish to set as missing and then click **Remove**.

Recode as missing values in Displayr

Click on the *variable* or *variable set* in the *Data tree* (bottom-left of the screen) and click the **DATA VALUES > Missing values** button in the **Properties** tab on the right-side of the screen. For each category that you wish to remove, set **Missing Values** to **Exclude from analyses**.

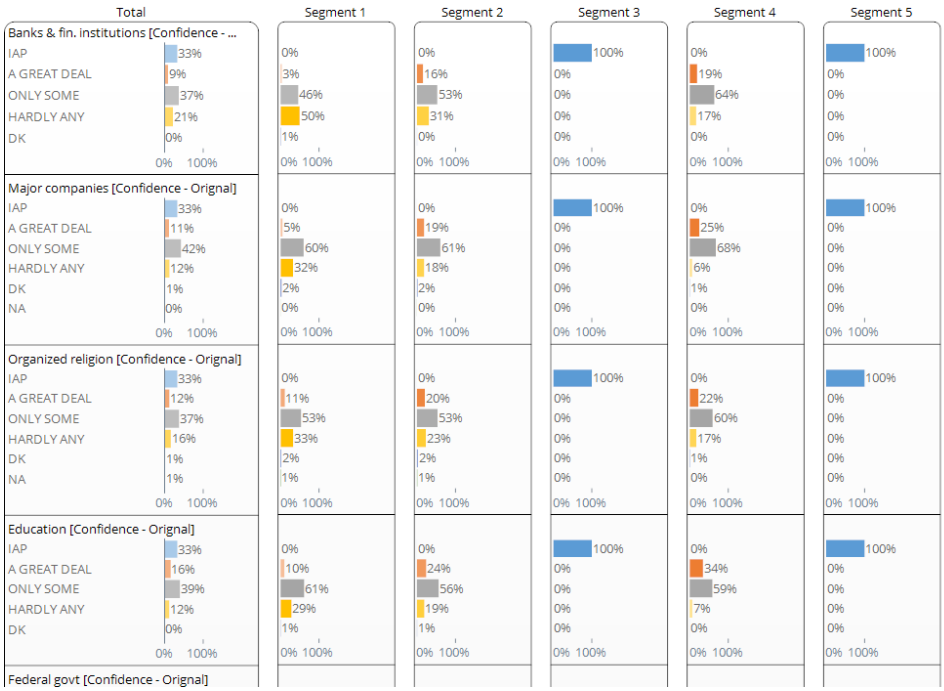
2nd best approach: Use techniques designed for nominal data

Latent class analysis and cluster analysis techniques have been developed to deal with *nominal* data (i.e., unordered categorical variables). If the segmentation variables are treated as nominal then the *Don't Know* and *Not Applicable* responses are treated just like any other categories.

While this technically solves the problem, in practice the resulting segmentations tend to be worse than those that treat the categories as missing values, as:

- If the other categories are numeric or ordered, this approach throws away a lot of information about how people differ. As the purpose of segmentation is to differences between people, this is a problem.
- If the other categories are numeric or ordered, this information is ignored by the techniques that find the segments, you can end up with illogical groupings (e.g., people that either *Strongly Agreed* or *Strongly Disagreed*). This problem can be reduced by merging smaller adjacent categories together.
- The segmentations can be difficult to interpret as the results typically become so large that they cannot all be viewed on the screen at the same time.
- With more than 3 categories, or any small categories, the segmentations can be unstable (see **Nominal variables that have more than 3 categories**).
- The segmentation can end up reflecting uninteresting differences (e.g., one segment who said *don't know* a lot).

We can readily appreciate these problems by looking at the output generated using our case study data. Below is the output for just the first four variables, which is all that can readily fit on a screen. Note how for segment 3 and segment 5, we have segments of people that were not asked the question! So, we have dealt with not applicable data in a way that is statistical meaningful but managerially dumb.



The output for all the data is shown below. Clearly this is impractical to view. If you cannot view a segmentation you cannot describe it and thus cannot use it.

2. Missing data due to not being asked (skips and filters)

Many surveys use skips and filters to avoid asking people questions where the answer is already known. For example, if from Q1 we know that somebody does not drink Coke, there is no need to ask them how frequently they consume Coke in Q2.

Where data is missing due to skips and filters, it needs to be recoded so that the missing values are changed to categories that reflect what is known about a person.

Consider the following two questions:

Q1. Which of these brands do you drink?
Coke
Pepsi
Fanta

Q2. How many times per week do you drink…
Only show brands selected in Q1

	<1	1	2	3	4 or more
Coke	❍	❍	❍	❍	❍
Pepsi	❍	❍	❍	❍	❍
Fanta	❍	❍	❍	❍	❍

The data from Q2 will contain missing values for people that indicate they don't drink the brands in Q1. But, these missing values are very different from the types of missing values that we considered in the previous section. If somebody has missing values in Q2, it tells us that they don't drink the brand.

The solution is to recode the data as if only a single question was asked:

Q1&2. How many times per week do you drink…

	Never	<1	1	2	3	4 or more
Coke	❍	❍	❍	❍	❍	❍
Pepsi	❍	❍	❍	❍	❍	❍
Fanta	❍	❍	❍	❍	❍	❍

Usually, the easiest way to do this is just to rebase the variables (i.e., recode the missing data as a specific value, such as 0).

Re-basing a variable in Q

- Go to the **Variables and Questions** tab
- Press the … button in the middle of the screen for the variable(s) of interest
- Uncheck **Missing Data** next to the missing data categories
- Fill in appropriate values for **Value** and **Label** and press **OK**

Re-basing a variable in Displayr

- Select the variable in the *Data Sets Tree* (bottom-left of the screen)
- Press the **DATA VALUES > Missing values** button in the *Object inspector (*right-side of the screen)
- Change the **Missing Values** selection to **Include in analysis**
- Fill in appropriate values for **Value** and **Label** and press **OK**

3. Ordinal Variables

> Ordinal variables have values that contain a natural ordering, such as: Unimportant, Somewhat Important, Important, and Extremely Important.

Curiously, most market researchers collect ordinal data for segmentation (e.g., importance ratings, agree-disagree scales), but most software that can be used for segmentation is not designed to take ordinal data into account.

Best approach: treat the ordinal data as if it is numeric

Most researchers that have ordinal data tend to treat it as numeric when they perform segmentation. This seems to be the best approach. It is usually a good idea when doing this to also recode the data so that the numbers have an easy-to-understand meaning. For example, for the confidence data, the data file represents *A GREAT DEAL* as a 1, *ONLY SOME* as 2, and *HARDLY ANY* as 3. It is better to *reverse code* these values, as that means that when it comes time to interpret averages of this data, higher averages translate to more confidence.

Row %	A GREAT DEAL	ONLY SOME	HARDLY ANY	NET
Banks & fin. institutions	13% ↓	55% ↑	32% ↑	100%
Major companies	17% ↓	64% ↑	19% ↓	100%
Organized religion	19% ↓	57% ↑	24% ↓	100%
Education	23% ↑	59% ↑	18% ↓	100%
Federal govt	11% ↓	44% ↓	45% ↑	100%
Organized labor	12% ↓	59% ↑	30% ↑	100%
The Press	8% ↓	47% ↓	45% ↑	100%
Medicine	38% ↑	52%	11% ↓	100%
Television	9% ↓	49% ↓	42% ↑	100%
US supreme court	22%	56% ↑	22% ↓	100%
The scientific community	42% ↑	51%	7% ↓	100%
Congress	5% ↓	38% ↓	57% ↑	100%
Military	50% ↑	41% ↓	9% ↓	100%

In addition to *reverse coding,* it is common to try and make sure that categories are represented with appropriate numeric values. For example, if a question asks for people to give ratings from 0 to 10, it is a good idea to recode the values so that a 0 response by a person is recorded as a 0 in the data file (typically, the first category will have a value of 1, which means that scales that start at 0 actually are assigned values starting at 1 by most data collection software).

Where the categories of a question represent ranges, these can be represented by midpoints or approximations thereof. For example, the values in brackets represent values for the categories corresponding to the number of times per week: *Less than once every 4 weeks* (.1), *Every 2 to 4 weeks* (0.4), *Once a week* (1), *2 to 6 times a week* (4), *Once a day* (7), *Multiple times a day* (15).

You could rightly look at these values and conclude that the numbers are not obviously correct. *Multiple times a day* could easily have been given a value of 14, 20, or even 50. Who knows what is correct? The good news is that it turns out not to make a big difference unless you do something silly (e.g., values of 0.1, 0.2, 20, and 1,000).

Treating data as numeric in Q

On the **Variables and Questions tab**, change the **Question Type** to either **Number** or **Number – Multi**. Then, press the **…** button in the middle of the screen and change the **Value** column entries.

Treating data as numeric in Displayr

Select the data in the **Data tree** (bottom-left of the category) and change the **Structure** (in the **Properties** tab on the right of the screen) to either **Numeric** or **Numeric – Multi**. Click the **DATA VALUES > Values** button and change the **Values** column entries if necessary.

2nd best approach: use algorithms designed for ordinal data

There are a variety of special purpose algorithms that can be used for ordinal data. Our own internal studies have found that usually, but not always, these methods are inferior to the simpler approach of treating the variables as numeric. Furthermore, the algorithms for ordinal data tend to be a lot slower, and it is hard to justify using a technique that is both slower and not as good.

Treating data as rankings in Q

On the **Variables and Questions** tab set **Question Type** to **Ranking**. This will mean that the segmentation will treat the data as a ranking, with variables with the highest values for each respondent regarded as being ranked equal first, the second highest values equal second, and so on.

Treating data as rankings in Displayr

Select the data in the **Data tree** (bottom-left of the category) and change the **Structure** (in the **Properties** tab on the right of the screen) to **Ranking**. Note that if you choose the **Ordinal** structures Displayr will just compute averages (because it is usually the best approach).

3rd best approach: merge all the categories to binary

If you are uncomfortable with the idea of treating ordinal data as numeric, an alternative approach is to merge together the ordinal categories until you only have two categories. The resulting binary variables are also technically numeric, so all traditional cluster analysis methods can be used.

This approach throws away a lot of information about difference between people. As the point of segmentation is to understand differences between people, this is a serious limitation.

4th best approach: Dimension reduction techniques

Dimension reduction techniques, such as *PCA* and *multiple correspondence analysis*, can be used to convert the ordinal data into numeric data, which can then be used with any of the standard segmentation techniques. Please refer to the discussions of the limitations of this approach in **Dimension reduction techniques**.

4. Nominal Variables

> *Nominal* variables have categories that are unordered. For example: Grocery Store, Bar, Restaurant, Vending Machine, and Other.

The most popular cluster analysis algorithms, such as *k*-means and the various hierarchical algorithms are not designed for nominal variables. The previous chapter made the point that there is no need to use special-purpose algorithms for ordinal data. The case of nominal variables is very different. It is generally the case that, while there are alternatives, special-purpose algorithms are superior.

Best approach: use techniques specifically designed for nominal variables

The benefits of using a specialist technique designed for nominal variables are most readily appreciated by reading through the alternatives (listed after this section), all of which are problematic.

Nominal variables in Q

On the **Variables and Questions tab,** change the **Question Type** of the data to **Pick One** or **Pick One – Multi**. Use **Latent class analysis** or **Mixed-mode cluster analysis**.

Nominal variables in Displayr

Change the **Structure** of the variable sets to either **Nominal** or **Nominal – Multi**. Use **Latent class analysis** or **Mixed-mode cluster analysis**.

2nd best approach: Merge the categories of each variable until there are only two categories

The first step is to combine the categories. For example, if your variable contained the categories of *Coke*, *Diet Coke*, *Pepsi*, and *Diet Pepsi*, you might combine them to be *Colas* and *Diet Colas*. Once the data is binary, any of the standard techniques can be used.

This approach suffers from the same problems as the previous approach but, additionally, throws out a lot of information.

3rd best approach: convert each categorical variable to binary variables

This approach replaces each of the nominal variables with a set of *indicator variables*. These are binary variables where there is one variable for each of the categories of the nominal variable, and each variable has value of 1 if the corresponding category was selected and 0 otherwise (this is sometimes referred to as *dummy variables* and as *one hot coding*). Cluster analysis or latent class analysis can then be run as if the data was numeric.

This approach can be as good as using the specialist techniques. However, there are some downsides:

- Politics. Sometimes it is suggested that it is inappropriate to use this method. The explanations for why tend not to make much sense. However, rather than getting bogged down in the detail of the argument, the key thing to appreciate is that if you use specialist techniques for nominal data - as described in the previous section, nobody complains.
- If you have *mixed-mode* data, such as some numeric variables and some nominal variables, then this approach does not work, as the resulting segmentation ends up being dominated by the numeric data (because the numeric data contains more information).
- The approach is a lot more time consuming, as you must recode the new variables and, if you decide to merge categories, it takes a lot more time.

4th best approach: Dimension reduction techniques

This approach is identical, and has the same problems, as when used with ordinal variables. See **Dimension reduction techniques** and **Ordinal Variables**.

5. Nominal variables that have more than 3 categories

The smaller the size of nominal categories, the more difficult it typically is to interpret the resulting segments. Generally, when there are more than three categories this tends to cause a problem.

The further a category is from being selected by 50% of the sample, the less influential it is in the segmentation. The reason for this is best appreciated by the extreme case where there is a category containing only a single person. As nobody else has the same data for that variable, it is impossible for anybody else to be similar to that respondent on that category.

Small categories cause two other types of pain. First, the more small categories, the more categories in general, and thus the harder it is to interpret the data. Second, where there are lots of small categories, as none are particularly influential, the resulting segmentation can be unreliable, with nonsensical results.

Best approach: Merge similar categories

Merging similar categories so that small segments disappear is usually the most useful solution.

2nd best approach: Ignore the problem

A challenge with merging is that it is not clear what the right level of merging is. Should we merge so that we have three categories? Should we merge so that no category has less than 20% of the sample? There are no good answers to these questions. Consequently, a viable alternative is to ignore the problem and perform a segmentation and then, if the results are problematic, start merging categories until the results stabilize.

6. Questions with different ranges

The scale of a variable is a determinant of the influence that a variable has in the segmentation. It is usually a good idea to adjust for this when performing the segmentation.

Cluster analysis and most latent class analysis methods implicitly take differences in scale into account. Scale refers to both the *range* of a variable and the number of unique values that a variable can take (e.g., a 5-point scale has 5 unique values and a range of 4). There are several different facets to this:

- Variables with a greater range tend to have a bigger influence. E.g., if you have a 10-point scale and a 3-point scale, the likelihood is that the segments will differ primarily in terms of the data with the 10-point scale.
- Even when variables are adjusted to take into account differences in range, they can differ substantially in their variance (and, for mathematical reasons, once we adjust variables to have the same range, variables that originally had fewer scale points end up having a higher variance than variables with more, and this higher variance makes them more influential, all else being equal).
- The number of scale points is a determinant of the variance. For example, all else being equal, when a 2-point scale is scaled to go from 0 to 1, it will have a higher variance than a 10-point scale that has been scaled to be in the range of 0 to 1. Variables with higher ranges are more influential.
- If using **Mixed-mode cluster analysis** or **Latent Class Analysis** to simultaneously analyze numeric and variables with other data types (e.g., nominal), it is usually the case that some of the different types of data tend to dominate the solution. In particular, MaxDiff data tends to swamp other types of data, and numeric variables tend to swamp nominal variables.

Option 1: Use algorithms that permit weighting of sets of variables

Q has an option to automatically allow for different sets of variables (i.e., questions) having different ranges. See **Segments differ on only a few variables** for a discussion of how to do this.

This does not work if you want to scale variables within a set of variables, and does not exist in Displayr - in such situations the methods described in the rest of this chapter can be used.

Option 2: Scale variables to have a range of 1

This has long been the standard recommendation in the cluster analysis literature. For example, if you had three-point and five-point scales, they could be recoded as:

Option	Numeric value
Unimportant	.00
Important	.50
Very important	1.00

Option	Numeric value
Strongly disagree	.00
Disagree	.25
Neither	.50
Agree	.75
Strongly agree	1.00

Option 3: Scale variables to have a standard deviation of 1

This has long been a standard practice in market research.

Option 4: Use dimension reduction to pre-process the data

Dimension reduction techniques, such as *PCA* and *multiple correspondence analysis*, which are based on correlations and as such implicitly standardize the range, can be used to create a new set of numeric variables with a common standard deviation. Please refer to the discussions of the limitations of this approach in **Dimension reduction techniques**.

7. Text data

Text data needs to be converted to numeric data prior to being used in a segmentation. The best approach for doing this is usually to use principal components analysis of text data.

All the widely used algorithms developed for segmenting require data to be numeric or categorical. Consequently, if we have text data, we need to turn it into numeric data prior to using it in the segmentation process.

The number of techniques that have developed for analyzing text data is too numerous to mention. One of the newest of the techniques, *Principal Component Analysis of Text Data*[7] is, in our experience, the most reliable approach to including text data in segmentations.

There are two key outputs of principal component analysis of text data:

- Variables that can be used in the segmentation.
- Loadings.

An example data set and loadings is shown below. The way that we read this is that two components have been identified, and they explain 21.2% of the variance in the text data, where:

- The first component primarily identifies the extent to which people:
    - Said "nothing"
    - Had negative sentiment (i.e., there is a -ve sign on `Positive sentiment`)
    - Wrote short response (`nchars`)
- The second component identifies the extent to which people didn't mention service.
- Due to the richness of text data, the "correct" number of components to summarize a data set is usually in the dozens or hundreds, and, consequently, we instead choose a relatively small number of components to keep things practical (e.g., 10 or less), recognizing that whatever variance is explained in the smaller components is unlikely to be important in a segmentation contact.

[7] https://www.displayr.com/principal-component-analysis-of-text-data/

Text	Likes - Phone provider
1	great offers
2	Discount
3	No contracts and ability to set up payment arrangements
4	I like that they are reliable.
5	it is well
6	everything
7	Nothing
8	nothing
9	the rates are reasonable. i like tmobile tuesdays. i feel appreciated by the company.
10	no contract and low rates
11	Nothing
12	they have good customer service
13	I like my plan, which is unlimited talk and data, and I like that the network is very stable.
14	the price
15	That I have good reception and never have billing problems
16	They have terrific customer service and I get a good deal with a bundle.
17	Financing
18	Speed
19	very likely
20	Everything

Principal Component Loadings

2 components explaining 21.2% of the variance

	Component 1 (14.3%) *Eigenvalue: 73.28*	Component 2 (6.9%) *Eigenvalue: 35.30*
Any: nothing	0.787	0.080
Any: not	0.756	0.095
Any: no	0.657	0.138
Positive sentiment	-0.511	-0.401
nchars	-0.466	0.026
sentiment.cleaned	-0.453	-0.300
sentiment.normalized	-0.427	-0.352
Any: everything	0.310	0.099
Any: plan	-0.304	0.265
Exclusive: everything	0.296	0.080
Any: data	-0.233	0.173
Any: like	-0.220	0.106
Any: able	-0.219	-0.084
Any: price	-0.218	0.144
Any: phone	-0.211	0.197
Any: unlimited	-0.210	0.196

The image above shows the top of the loadings. The bottom of the loadings is:

Exclusive: always	-0.072	-0.024
Any: always	-0.065	-0.013
Exclusive: pay	-0.064	0.039
Exclusive: never	-0.059	0.048
Any: service	-0.227	-0.744
Any: good service	-0.112	-0.537
Exclusive: good service	-0.110	-0.513
Any: good	-0.220	-0.479
Exclusive: service	-0.091	-0.317
Any: customer service	-0.134	-0.292
Exclusive: customer service	-0.110	-0.255
Exclusive: reliable	-0.113	-0.199
Any: great	-0.146	-0.180
Stem: great	-0.146	-0.176
Any: reliable	-0.142	-0.167
Any: reception	-0.073	-0.150
Negative sentiment	-0.104	0.125

Displayr instructions

1. **Anything > Advanced Analysis > Text Analysis > Advanced > Principal Components Analysis (Text)**
2. Select the **Text Variable**. If you have multiple variables, either perform the analysis multiple times, or, create a new text variable that combines the multiple variables
3. (Optional) Choose your preferred solution by modifying the **Number of components** and **Rotation method**
4. **SAVE VARIABLE(S)** > **Components/Dimensions**
5. Expand the new variable set, right-click on each variable and click **Rename** and give it an appropriate name

Q instructions

1. **Create > Text Analysis > Advanced > Principal Components Analysis (Text)**
2. Select the **Text Variable**. If you have multiple variables, either perform the analysis multiple times, or, create a new text variable that combines the multiple variables
3. (Optional) Choose your preferred solution by modifying the **Number of components** and **Rotation method**
4. **SAVE VARIABLE(S)** > **Components/Dimensions**
5. Right-click on the component names and rename them

8. No or few observations have complete data

> Most of the widely used cluster analysis algorithms can be highly misleading or simply fail when most or all the observations are missing.

If all the observations have one or more missing values then most traditional cluster analysis methods will not work at all. This is true of all the main cluster analysis methods in R and SPSS (even when you select the options for running the analysis using partial data).

Best approach: Use cluster analysis and latent class algorithms that work with partial data

Most latent class analysis software will work with partial data, while most software written for the other techniques will not.

Q and Displayr's **Latent Class Analysis, Mixed Mode Cluster Analysis,** and **K-Means (batch)** do this. This comes with some assumptions, as discussed in the next two chapters.

2nd best approach: Impute missing values

Imputation refers to tools for predicting the values that would have existed if the data was not missing. If you use a sensible approach to imputing missing values (and replacing missing values by the average of their other values is not a sensible approach), running cluster or latent class analysis on the imputed data set means that the missing data is treated as being the desirable *missing at random* (defined in the next chapter).

This can be done via code in Q and Displayr (all the main imputation R packages are available).

3rd best approach: Forming clusters based on complete cases and allocate partial cases to segments

A popular approach is to form segments using only observations with complete cases, and then assign the observations with incomplete data to the most similar segment based on the data available. This approach is used by SPSS's cluster analysis algorithms and Q's old **Legacy Cluster Analysis** algorithm.

If you decide to use this approach, please make sure you read the next chapter, as it described how to check how risky this approach can be.

Worst approach: Form clusters based only on complete cases

This is the most common approach. If you decide to use this approach, please make sure you read the next chapter, as it described how to check the dangers of this approach.

9. Missing data can be predicted based on non-missing data

Consider the situation where people with missing data on one variable have higher values on another variable than do people without missing data.

When missing data is predictable in such a way, it has important implications for how segments need to be created.

Missing data assumptions

When conducting an analysis of data that involves missing values it is necessary to make assumptions about what has caused the data to be missing. Making an incorrect assumption can profoundly damage the validity of a segmentation.

There are four qualitatively distinct types of missing data. Missing data is either: *structurally missing, missing completely at random (MCAR), missing at random (MAR)*, or *nonignorable* (also known as *missing not at random*). The type of missing data dictates how missing data needs to be treated for any analysis to be meaningful.

Structurally missing data

Structurally missing data is data that is not missing due to a logical reason -- it is data that is missing because it should not exist. For example, in the table below, the first and third observation have missing values for **Age of youngest child**, and this is because these people have no children. Such missing data is typically best addressed by excluding people with such missing data from any analysis of the variables with these structurally missing values.

In the **How many colas did you drink in the past 24 hours** column, there are also three structurally missing values. In this case, we can logically deduce that the correct value is 0, so this value should be used in place of the missing values in any analysis of this variable (as discussed in **Missing data due to not being asked (skips and filters)**).

ID	Children	Age of youngest child	Did you drink Coca-Cola in the last 24 hours?	How many colas did you drink in the past 24 hours?
1	No		No	
2	Yes	18	Yes	2
3	No		No	
4	Yes	13	No	
5	Yes	8	Yes	1

Missing completely at random (MCAR)

Looking at the table to the below, we need to ask ourselves: what is the likely income of the fourth observation? The simplest approach is to note that 50% of the other people have high incomes and 50% have low incomes, and to assume therefore that there is a 50% chance she has a high income

and a 50% chance she has a low income. This is known as assuming that the missing value is ***m**issing **c**ompletely **a**t **r**andom* (MCAR). When we make this assumption, we are assuming that whether the person has missing data or not is completely unrelated to the other information in the data.

ID	Gender	Age	Income
1	Male	Under 30	Low
2	Female	Under 30	Low
3	Female	30 or more	High
4	Female	30 or more	
5	Female	30 or more	High

It is relatively easy to check the assumption that data is missing completely at random. If you can predict which units have missing data (e.g., using common sense, regression or some other method), then the data is not MCAR. A more formal way of testing is to use *Little's MCAR test*.[8]This is available in Q using **Automate > Browse Online Library > Missing Data > Little's MCAR Test**, and in Displayr via **Anything > Data > Missing Data > Little's MCAR Test**.

When data is missing completely at random, it means that we can safely undertake analyses using only observations that have complete data (provided we have enough of such observations).

The MCAR assumption is rarely a good assumption. It is only likely to be true in situations where the data is missing due to some truly random phenomena (e.g., if people were randomly asked 10 of 15 questions in a questionnaire).

Missing at random (MAR)

In the case of *missing completely at random*, the assumption was that there was no pattern. An alternative assumption, known somewhat confusingly as *missing at random* (MAR), instead assumes that we can predict the value that is missing based on the other data.

[8] Little, R. J. A. (1988). A test of missing completely at random for multivariate data with missing values. Journal of the American Statistical Association, 83(404), 1198-1202.

Looking again at the table above, we return to the problem of trying to work out the value of the fourth observation on income. A simple predictive model is that income can be predicted based on gender and age. Looking at the table below, which is the same as the one above, we note that our missing value is for a Female aged 30 or more, and the other female aged 30 or more has a High income, and so we can predict that the missing value should be High. Note that the idea of prediction does not mean we can perfectly predict a relationship. All that is required is a *probabilistic relationship* (i.e., that [9]we have a better than chance probability of predicting the true value of the missing data).

When data is missing at random, it means that we need to either use an advanced imputation method or an analysis method specifically designed for *missing at random* data.

Missing at random is always a safer assumption than missing completely at random. This is because any analysis that is valid with the assumption that the data is missing completely at random will also be valid under the assumption that the data is missing at random, but the opposite is not the case.

Nonignorable

It may be the case that we do not believe we can confidently make any conclusions about the likely value of missing data. For example, it is possible that people with very low incomes and very high incomes are shy and tend to refuse to answer. Or there could be some other reason we just do not know. This is known variously as *missing not at random data* and as *nonignorable missing data*.

It is common to include *structural missing data* as a special case of data that is *nonignorable,* but this misses an important distinction, which is that *structurally missing data* is easy to analyze, whereas other forms of *nonignorable missing data* are highly problematic.

When missing data is nonignorable, it means that we cannot use any of the standard methods for dealing with missing data (e.g., imputation, or algorithms specifically designed for missing values).

If the missing data is nonignorable, it means that any standard calculations give the wrong answer. Consider a study looking at homelessness.[10] Data was obtained from 31 women, of whom 14 were located six months later. Of these, three had exited from homelessness, so the estimated proportion to have exited homelessness is 3/14 = 21%. As there is no data for the 17 women who could not be contacted (i.e., 31 – 14), it is possible that none, some, or all of these 17 may have exited from homelessness. This means that potentially the proportion to have exited from homelessness in the sample is between 3/31 = 10% and 20/31 = 65%, and thus reporting the 21% as being the correct

[9] Little, R. J. A. and D. B. Rubin (1987). Statistical Analysis with Missing Data. Brisbane, John Wiley & Sons

[10] Manski, Charles F. (1995). Identification Problems in the Social Sciences. Harvard University Press.

result is misleading. Note that in this example the missing data is nonignorable and treating it as missing at random would also be inappropriate, since the inability to contact the women is likely to be causally related with whether or not they have exited from homelessness. Thus, strategies designed for data which is missing at random, such as imputation, will not work.

Best approach if data can be predicted based on non-missing data: Use cluster analysis and Latent class analysis algorithms designed for partial data

In Q and Displayr, all latent class and two cluster analysis algorithms have been designed to work with data that is missing at random (MAR): **Mixed-Mode Cluster Analysis** and **K-Means (batch)**. The **Legacy Cluster Analysis** should not be used in this situation, as it is based on complete cases.

2nd best approach: Impute missing values

See the previous chapter

10. Data is missing for a really good reason (nonignorable)

Sometimes missing data is deeply significant. For example, if a question asks “how important are APIs to you” it is likely that the people who say “don’t know” do so because they do not understand the question. In such situations, even partial-data algorithms designed for *missing at random* (missing data that can be predicted based on non-missing data) data are not appropriate.

There are a number of ways of working out whether missing data is nonignorable:

- Common sense.
- Follow-up interviews where people are asked why they said “don’t know”
- View missing data visualizations and look for problems:
    - Q: **Automate > Browse Online Library > Missing Data > Plot by Case**
    - Displayr: **Anything > Data > Missing Data > Plot by Case**.

Best approach if missing data is nonignorable: Treat the data as nominal

If missing data is missing not at random, the purest solution is to treat the data as nominal (see **Nominal Variables** and **Nominal variables that have more than 3** categories).

However, if the data is otherwise numeric or ordinal, the resulting analyses can be very poor, as by treating the data as being nominal we are ignoring all the relativities.

2nd best approach if missing data is nonignorable: Exclude variables

If the problem only exists in a subset of the variables, it can be practical to exclude those variables.

3rd best approach if missing data is nonignorable: Use partial-data techniques

This is, unfortunately, one of those areas of data analysis where the theoretically superior options tend to be impractical. Consequently, in practice, the most expedient course of action is sometimes to pretend the problem does not exist and treat the nonignorable data as if it is *missing at random by* using techniques that form segments using partial data (see the previous chapter).

Worst approach if missing data is nonignorable: Use imputation

Imputation can also be used to replace the missing values, but this makes the MAR assumption and tends to involve a doubling-up of errors, as described in the earlier chapters.

11. Weights

> Most cluster and latent class analysis algorithms ignore survey weights.

Most techniques that are used for forming segments have not been designed for use with survey weights, because in the fields where they were invented (biometrics, computer science, academic statistics), survey weights are relatively rare.

Best approach: Use algorithms that take weights into account

The best approach to this problem is to use software that can accommodate weights. In practice, that will generally limit you to using software designed for survey researchers, which include most of the methods in Q, SPSS, and Displayr.

2nd best approach: Create bootstrapped samples

Most cluster analysis techniques can be applied to weighted data. The trick is to create a new data set by randomly sampling with replacement from the original data set, where the probability of selection of each observation is proportional to its weight. The new sample should be the same size as the existing sample.

This can be done using the R function called `sample`.

While using a bootstrapped sample is a statistically valid thing to do, it is inferior to using algorithms that take the weights into account, as the sampling process adds noise to the data. Furthermore, the approach is difficult to explain to end-users.

12. MaxDiff, Conjoint, Driver Analysis, and Choice Data

MaxDiff, conjoint, driver analysis, and choice data require special-purpose techniques when forming segments.

The basic structure of the data used for conjoint, MaxDiff, and driver analysis is wholly incompatible with the standard cluster and latent class analysis algorithms (i.e., you will get nonsensical outputs if you use it).

Option 1: Use latent class analysis algorithms designed for MaxDiff/conjoint/driver analysis

See **Latent class regression**.

Option 2: Use Hierarchical Bayes followed by cluster analysis or latent class analysis

This approach first uses **Hierarchical Bayes** to estimate *utilities* for every respondent, and then these utilities are analyzed using standard segmentation algorithms.

This approach is often inferior to using a purpose-built latent class analysis algorithm. This is because the HB makes one set of assumptions, and these assumptions are different to the assumptions made when forming the segments. The net effect of these conflicting assumptions is that the amount of error in the process compounds, and the segmentation is not as strong.

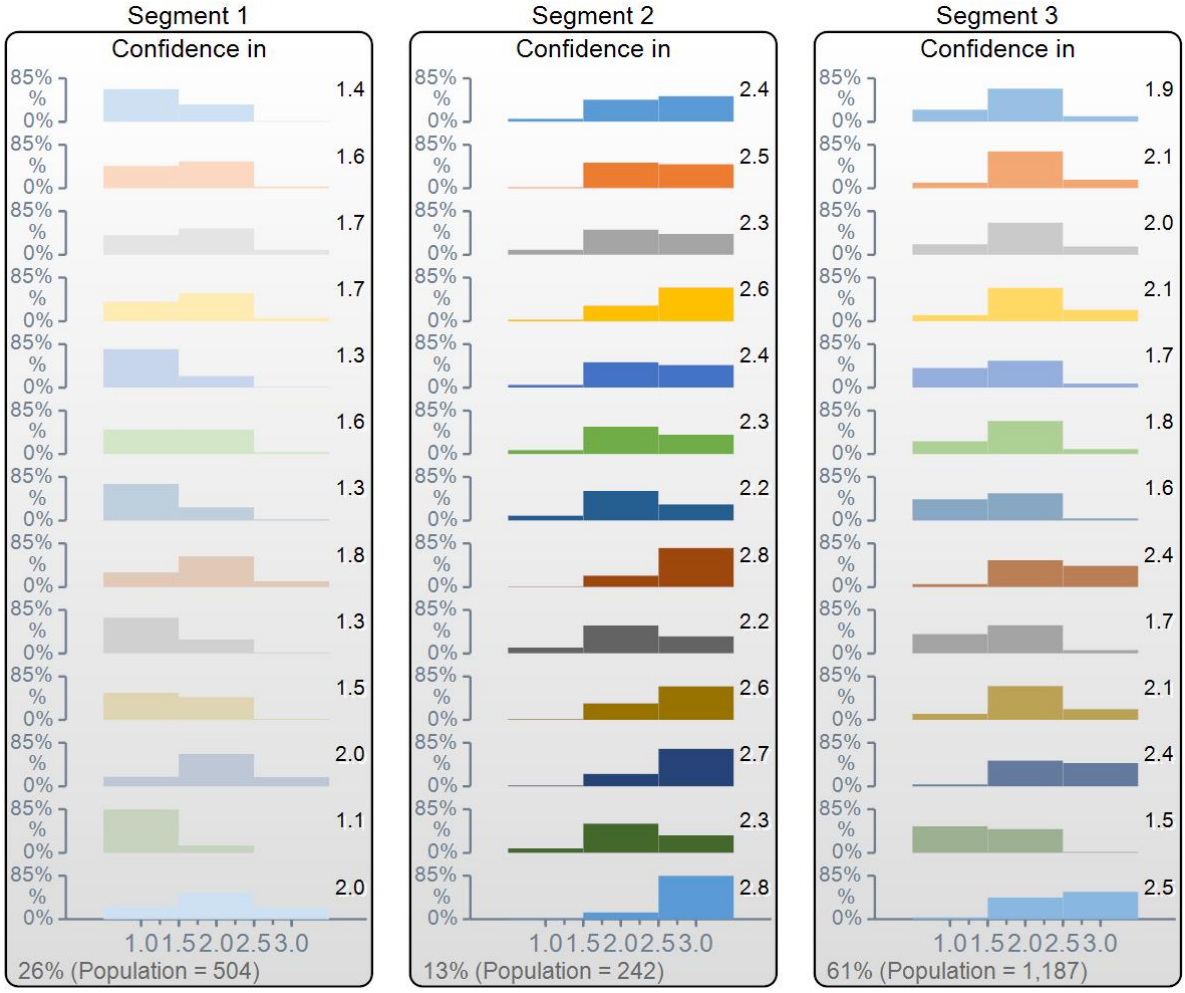
For more information about this approach, please refer to The Accuracy of Hierarchical Bayes When the Data Contains Segments.

FORMING SEGMENTS

13. Segments that mainly reflect response biases

When segmentation is performed using rating scale data, it is not uncommon to find segments that have either high means for all the variables, or, low means for all the variables. Such results can be symptomatic of problems with the data rather than being insightful.

The output below shows three segments from a Latent class analysis of the Confidence variables. Have a go at interpreting the segments. The height of the columns in each histogram indicate the proportion of people to have given a rating of 1, 2, or 3. The numbers to the right show the averages. What do you see?



Segment 1 have low levels of confidence in everything. Segment 2 have high levels of confidence in everything. Segment 3 have moderate levels of confidence in everything.

There are two obvious explanations for this result. One is that in fact the key difference between people is in their average level of confidence in institutions. An alternative explanation is that the segmentation has just identified response biases (i.e., differences in how people answer questions in general).

Many researchers believe that results like these are due to a response bias called *yeah-saying bias,* where some people differ in terms of their tendency to give low versus high ratings.

There are strategies for removing yeah-saying bias from data. However, it is important to keep in mind that we often do not really know if we have yeah-saying bias. It could instead be that if people have a particular level of confidence in one institution, they tend to have a similar confidence in all the institutions. Where it is suspect that there is yea-saying bias, a prudent course of action is to create one or more segmentation using the original data, and one or more segmentations using the corrected (standardized data), and compare the resulting segmentations in terms of their overall usefulness (see **Finding the most managerially useful segments**).

Best approach: Modify each observation's data to have a mean of 0 and a standard deviation of 1

The first 13 columns of the table below show some of the raw data. The first respondent (row) contains a mix of 2s and 3s, with an average value of 2.46. By contrast, the third respondent's only given ratings of 1 and 2, with an average of 1.54.

The **Standardized** columns are computed using the formula:

Standardized = (Un-standardized – Mean) / Standard Deviation

As a result of the standardization, each respondent's data has a mean of 0 and a standard deviation of 1. This makes it much easier to compare the relativities between the respondents. For example, when we looked at Un-Standardized data it shows that all the respondents have a rating of 2. However, when we look at the Standardized scores, we can more readily see that such a rating was a high rating for the third respondent (row), an average rating for the second respondent, and a relatively low rating for the first respondent.

Un-standardized															Standardized														
Banks & fin. institutions	Major companies	Organized religion	Education	Federal govt	Organized labor	The Press	Medicine	Television	US supreme court	The scientific community	Congress	Military	**Mean**	**Standard Deviation**	Banks & fin. institutions	Major companies	Organized religion	Education	Federal govt	Organized labor	The Press	Medicine	Television	US supreme court	The scientific community	Congress	Military	**Mean**	**Standard Deviation**
2	3	3	3	2	2	2	2	2	3	3	2	3	**2.46**	**0.52**	-0.89	1.04	1.04	1.04	-0.89	-0.89	-0.89	-0.89	-0.89	1.04	1.04	-0.89	1.04	**0.00**	**1.00**
2			1	1	2	3	2	2				3	**2.00**	**0.76**	0.00			-1.32	-1.32	0.00	1.32	0.00	0.00				1.32	**0.00**	**1.00**
1	1	1	2	1	2	2	2	1	2	2	1	2	**1.54**	**0.52**	-1.04	-1.04	-1.04	0.89	-1.04	0.89	0.89	0.89	-1.04	0.89	0.89	-1.04	0.89	**0.00**	**1.00**

This can be dangerous if there are missing data (as standardization implicitly assumes that each person has seen the same options).

Software

In Q: **Automate > Browse Online Library > Create New Variables > Scale Variables > Standardize Within Case.**

In Displayr: Select the variables and click **Properties > TRANSFORMATIONS > Standardize Within Case** on the right.

Second best approach: Use latent class models that focus on relativities

In both Q and Displayr, an alternative to standardizing the raw data is to use latent class algorithms specifically designed for ranking data. This is achieved in Q by changing the **Question Type** to **Ranking**, and in Displayr by changing the **Structure** to **Ranking**.

While the underlying math of this approach is rather cool, in practice it is weaker than the traditional approach. First, our experience is that it produces worse segments in terms of discrimination with other variables. Second, it is very slow. It can easily take 100 times as long to compute as the traditional method. Third, sometimes it gets weird and unusable results. I discuss this in more depth in the next chapter.

14. Weird results from latent class regression

Latent class regression, which is a variant of latent class analysis used with MaxDiff, choice modeling and similar problem regression-based data sets, can sometimes provide non-useful segments.

This chapter describes the non-useful segments that can occur and strategies for avoiding them.

Weird types of results

Scale effects

Sometimes latent class regression can identify segments that look just like the segments we saw in the last chapter (i.e., appear to have a yeah-saying bias). That is, a key difference between the segments may just be that one segment has coefficients close to 0, whereas another may have the same relativities between the coefficients, but with higher overall magnitudes (i.e., it looks as if one segments' result is just the other's multiplied by a constant).

The root cause of this pattern is a difference in the *errors* of respondents. Respondents who provide a very high level of precision when answering the questions, providing completely consistent responses from question-to-question, have higher *utilities* than those who answer in a more inconsistent or haphazard way.

The segment which finds everything equally important

With MaxDiff and *ranking latent class regressions* sometimes a segment appears which has roughly the same coefficients for all the predictor variables. These uninteresting segments seem to be something of an 'other' segment, containing a whole lot of small segments squashed together.

Differences in intercepts

When conducting latent class analysis with ratings-based conjoint data or driver analysis data, sometimes the key difference between segments is only in the intercept. This may reflect yeah-saying bias or may be a meaningful difference between people in their attitudes to whatever is being measured.

Best approach: Individual-level parameters with traditional latent class analysis or cluster analysis

The best approach to weird results from latent class regression is often to;

- Estimate a **Hierarchical Bayes** model with multiple classes (this is a hybrid of HB and latent class regression).
- Extract the individual-level parameters from the model.
- Scale the individual-level parameters in some way that will bring out relativities. For example, setting them to have a minimum of 0 and a maximum of 100.
- Treat the scaled individual-level parameters as numeric variables and form segments using cluster analysis or latent class analysis.

This is discussed in more detail in the eBook on MaxDiff.

2nd best approach: Scale-adjusted latent class analysis

Variants of latent-class regression have been developed that automatically remove the scale-effect when forming segments. These are available in *Latent Gold.*

3rd best approach: Estimate lots of segments and group them together

An alternative approach is to use latent class regression and estimate many segments (e.g., 40), and then manually merge them together. However, that the approach described in the previous chapter is both faster and more likely to be successful.

15. Visualizing differences between segments

From a data visualization perspective, there's nothing particularly special about market segmentation. The key outputs from market segmentation are, in the main, crosstabs, and there is no shortage of ways to visualize crosstabs. However, in this chapter we share one specific visualization that we find particularly useful when evaluating segmentations.

The output below is from a Multivariate Analysis of Variance (MANOVA). The first two columns represent the two segments of a segmentation and the last two represent the variables. This visualization is particularly useful in two ways:

- It uses a heatmap shading to emphasize differences between the segments. So, we can easily see that, for example, **Confidence in Congress** is comparatively similar between the segments.
- The **R-Squared** column summarizes the extent to which the segmentation variable differs by the segmentation. Thus, we can quickly see that in the analysis below, the key distinction between the segments relates to **Importance: Help worse of people in rest of the World.** With market segmentation we are trying to find segments that differ in lots of ways. As this segmentation is largely about a single variable, this tells us that it is not a great segmentation. The next chapter describes how to improve it.

MANOVA: 2 Standardized

Significant - Smallest p-value (after applying False Discovery Rate correction): < 0.000000000001

	Flag and Christian Values n: 565	Scientific Citizenry n: 664	R-Squared	*p*
Confidence in: Banks & fin. institutions (n: 2566)	1.99	1.66	0.07	< .001
Confidence in: The Press (n: 2548)	1.52	1.73	0.03	< .001
Confidence in: US supreme court (n: 2520)	1.87	2.12	0.04	< .001
Confidence in: The scientific community (n: 2487)	2.15	2.51	0.09	< .001
Confidence in: Congress (n: 2528)	1.46	1.51	0.00	.042
Confidence in: Military (n: 2554)	2.55	2.29	0.04	< .001
Importance: Choose products for politics/ethics/envir. (n: 1197)	4.59	5.27	0.04	< .001
Importance: Help worse off people in America (n: 1247)	6.38	5.46	0.13	< .001
Importance: Help worse off people in rest of World (n: 1229)	5.77	3.74	0.31	< .001

n = from 1197 to 2566 cases used in estimation of a total sample size of 3,842; null hypotheses: two-sided; multiple comparisons correction: False Discovery Rate correction applied simultaneously to the entire table

MANOVA in Q

Create > Analysis of Variance > One-Way ANOVA

MANOVA in Displayr

Anything > Advanced Analysis > Analysis of Variance > One-Way ANOVA

16. Segments differ on only a few variables

As discussed in the previous chapter, it's typically a problem if segments only differ on a small number of variables.

When segments differ on only one or two variables it is commonly a sign that there is a technical problem of some kind. For example, perhaps missing data have been given a value of 99 or -99 rather than being set as being missing values. If the possibility of a technical problem has been ruled out, the problem can be rectified by changing the scale (i.e., weight) of variables in the analysis.

Best approach: Reweight the variables/questions

It is possible to tweak a segmentation by changing the range of variables used in the segmentation. Where a variable dominates a segmentation, its effect can usually be reduced by either:

1. Eliminating it from the analysis.
2. Recoding it so that it has a smaller range.
3. Including other variables multiple times.

Similarly, if a variable has the same mean in all the segments, recoding it so it has a higher range will often cause the segments to differ on that variable.

A few things to keep in mind when employing this strategy:

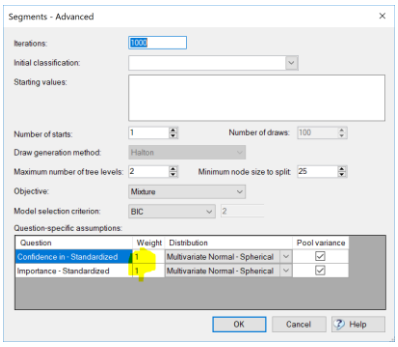
- Where the variable is highly correlated with lots of other variables, you may need to make the change very large and perhaps also recode the other variables.
- There is often a tipping point. For example, reducing the range by 50% may do little, where reducing it by 51% may have a huge effect.
- Modern software will often automatically rescale the range of the data in the background,[11] so recoding the data may have no effect. This is true of both Q and Displayr if using **Latent Class Analysis** and **Mixed-Mode Cluster Analysis**.

[11] Cluster analysis software will do this by recoding the data prior to performing the analysis. Latent class analysis software can instead achieve this end by making assumptions regarding the variance of the variables. The effect is the same, but the math is different.

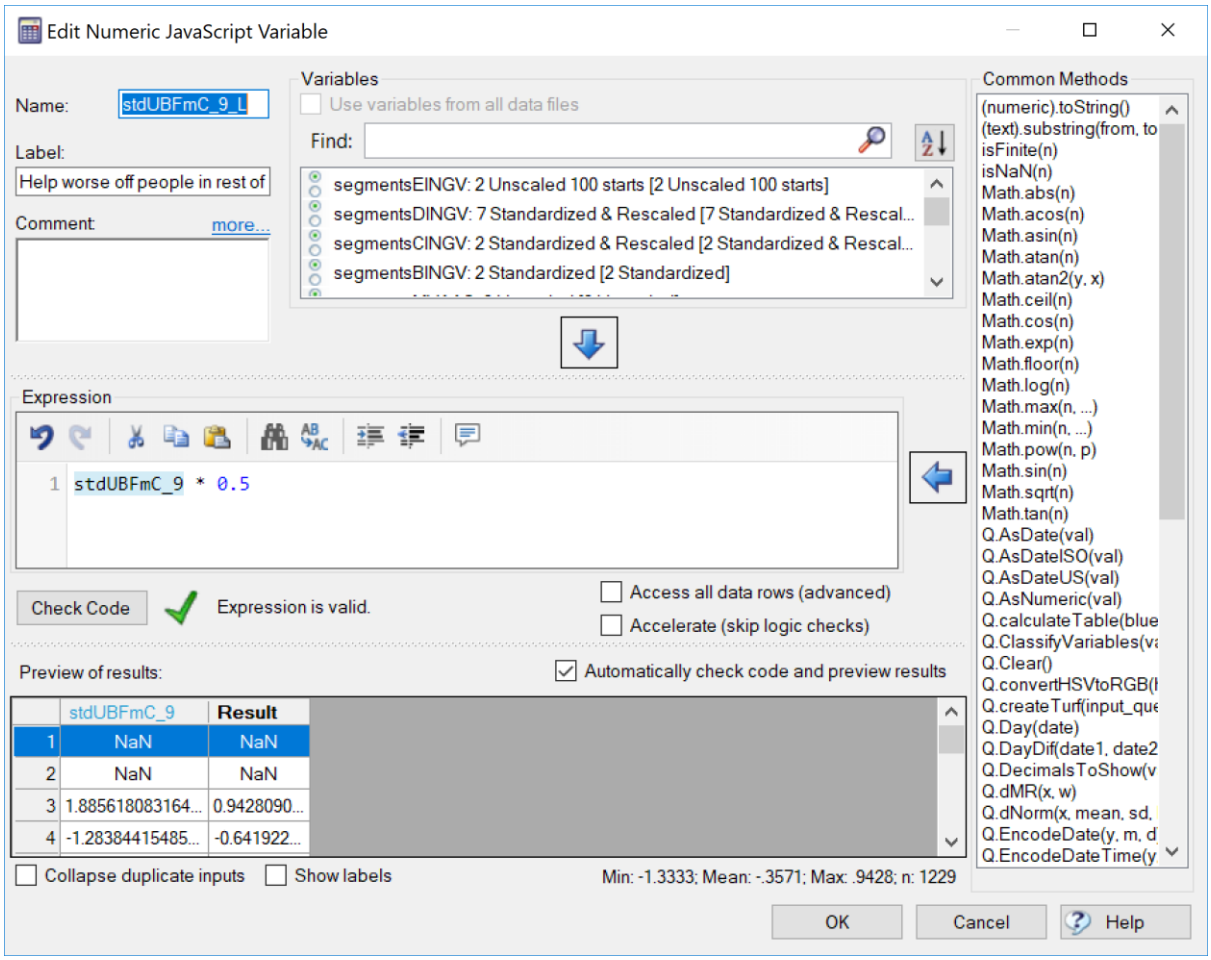
- You can do it multiple times, “rotating” the segmentation however you wish. (The metaphor of rotation is appropriate; this is essentially the same idea as rotating factors in factor analysis.)

Software

There are many, many, different ways to recode data. In Q, there are two efficient ways of recoding data when the goal is to change the weighting of variables in the segmentation:



1. *Changing the weight of an entire question.* Click **Create > Segments > Latent Class Analysis** or **Mixed-Mode Cluster Analysis**, press **Advanced**, and then change the value of the **Weight** next to each question. The bigger the value, the greater the weight applied to all the variables in the question.
2. *Changing the weight of variables within a question.*
    a. On the **Variables and Questions** tab, select the variables in a question, right-click, and select **Copy and Paste Question(s) > Linked**.
    b. Right-click on any of the individual variables you wish to modify and select **Edit variable**.
    c. Modify the code by multiplying it by a number. In the example below the range of the variable is halved by multiplying it by 0.5.



Second best approach: Use dimension reduction techniques

Dimension reduction techniques, such as *PCA* and *multiple correspondence analysis,* can be used to pre-process the data, with the resulting components used in the segmentation instead of the original variables. This can be beneficial because these techniques both remove the range of the original variables and correlations between the variables.

17. Identifying “golden questions”

Sometimes it is useful to have a small number of questions that can be used to allocate people to segments. Such questions are known as *golden questions.*

Applications of golden questions include, allocating people to segments in follow-up studies; tagging customer databases; and training salespeople to ask them so that customers can be targeted with the right offers.

Typically, “golden questions” are identified after a segmentation has been created. The basic process is:

- The segments are created.
- A set of potential golden variables are identified. These may be the original data variables used to form the segments, or some other, more readily-collectable data (e.g., age, or some shortened wordings of the original questions).
- **Predictive models** are used to predict the segments based on the potential golden variables. Random Forest is often a good algorithm for this.
- The prediction accuracy of the predictive analysis is computed. It is important to both compute the overall prediction accuracy and the prediction accuracy for each segment (it’s common to have situations where the overall prediction accuracy is really high, but one of the smaller segments cannot be predicted at all).
- The least important variables are removed from the predictive model, one by one, until the prediction accuracy drops to the lowest level acceptable. There is no science to make such a decision. It is entirely dependent on the context.

While the approach described above is the most common approach, it suffers from a serious limitation. The segmentation that is defined by the golden questions is typically qualitatively quite different to the original segmentation. The effect is typically a lot bigger than implied by the prediction accuracy, as the measurement error of golden questions is usually different to that of the original data.

One solution to this problem is to identify the golden variables (i.e., golden questions) at the same time as creating the initial segmentation. There are two strategies for doing this:

1. Where the golden questions are a subset of the variables being used in the segmentation, you create an initial segmentation, and then, one-by-one,

remove the variables from the segmentation that have the smallest R-Squared (see the previous chapter for a discussion of how to compute these).

2. Where you are using different variables (e.g., demographics to predict segment membership), use the strategies described in **Increasing the predictability of the segments**.

18. Working out the best number of segments

While there are several rules of thumb that can be used to work out the number of segments, in practice the best approach is to think through strategy rather than statistical considerations.

How many segments should you have? How should you work out how many segments? Is this a job for a statistician? Where does the client fit in? What role should judgment play?

Working out the correct number of segments is very much an art rather than a science. The best method is to try lots of different approaches and use your judgment to trade-off their recommendations.

Approach 1: Compare segments using metrics designed for choosing the number of segments

For the past 70 or so years statisticians and marketing scientists have been regularly inventing new ways to choose the number of segments. The most widely used in survey research is the Bayesian Information Criterion (BIC)[12], but there is no shortage of other techniques with wonderful names like Calinski-Haribasz, Bayes Factors, Entropy, the Corrected Akaike's Information Criterion, and so on.

Automatically choosing the number of segments using Q

In Q, for example, when you use **Create > Segments > Latent Class Analysis,** and tell it to automatically choose the number of segments (**Number of segments per split** set to **Automatic** with **Maximum** set to, say 20), it produces a table that looks like this:

	Log-likelihood	Parameters	BIC	Entropy	Iterations
Aggregate	-38,360.438	24.000	76,905.784	NaN	2.000
2 classes	-37,217.891	47.000	74,797.892	.658	23.000
3 classes	-36,733.245	70.000	74,005.804	.706	27.000
4 classes	-36,446.475	93.000	73,609.466	.654	129.000
5 classes	-36,213.107	116.000	73,319.934	.635	329.000
6 classes	-36,108.311	139.000	73,287.546	.626	295.000
7 classes	-35,917.083	162.000	73,082.292	.674	158.000

[12] Schwarz, Gideon E. (1978), "Estimating the dimension of a model", Annals of Statistics, 6 (2): 461–464 https://doi.org/10.1214%2Faos%2F1176344136

8 classes	-35,811.320	185.000	73,047.969	.653	224.000
9 classes	-35,775.089	208.000	73,152.709	.636	119.000

8 class solution has the best (lowest) BIC

As is true in most areas of statistics, as models get more complex, they usually fit the model better. That is, the more segments (`classes`), the higher the `log-likelihood` will usually be.

The BIC statistics is a heuristic that trades off two concerns: how well the model fits the data and the complexity of the model. BIC = - 2 `Log-likelihood + ln(n) Parameters`, where the number of parameters is a function of the number of classes. The 8-class solution is chosen because it has the lowest BIC.

It must be stressed that "heuristic" is just a fancy way of saying that somebody proposed the statistic and it seems to do an OK job sometimes; it is not science.[13]

The entropy can also be used for determining the number of segments. Entropy is a measure of cohesiveness, taking values in the range of 0 to 1. In Q, a 1 indicates that there is no variation within segments and values closer to 0 indicate a lot of variation within each segment. (It is common to see entropy defined the other way around, with 0 indicating no variation.)

Automatically choosing the number of segments using Displayr

When using latent class analysis in Displayr (**Anything > Advanced Analysis > Cluster**) with **Choose how many groups to find** set to **Work out the number of groups automatically** the BIC is used. To view the BIC values, select the output and then select **Anything > Advanced Analysis > Cluster > Diagnostic > Analysis Report.**

If you have instead created segments using the specialist MaxDiff or Choice Modeling tools (**Anything > Advanced Analysis > MaxDiff > Latent Class Analysis** and **Anything > Advanced Analysis > Choice Modeling > Latent Class Analysis**), you need to manually create models for each number of segments that is of interest to you, and collate the BIC statistics, which appear at the bottom of each table. This can be done automatically using **Anything > Advanced Analysis > MaxDiff > Ensemble of Models** and **Anything > Advanced Analysis > Choice Modeling > Ensemble of Models**.

[13] In fairness to Schwarz (1978), he was not proposing the BIC for segmentation, and in other contexts it has a more rigorous footing.

Approach 2: How strongly do the segments relate to other data?

If you form segments and the resulting segments are associated with other data, such as demographics and behavior, this suggests that the segments are in some sense valid. This same logic can be used to select the number of segments: if, say, the four-segment solution is more strongly related to other data than the five-segment solution, this implies that it is likely a better number of segments.

See **Quickly profiling segments** for a discussion of how to quickly work out the relationship between the segments and other data.

Approach 3: Take managerial considerations into account

Typically, the key managerial considerations (other than those addressed by the two previous approaches) are:

1. The fewer the segments the better. Every extra segment adds to the complexity of implementing a segmentation.
2. Are the segments easy to name? If you cannot find a good name to explain what's special about a segment, it tends to be near-impossible to effectively use the segment in decision-making.
3. Are the segments inspirational? That is, when the segment is described to the end-users of the segmentation (e.g., the marketers), does it give them lots of great ideas?

Approach 4: Replicability

The higher the replicability of the segmentation, the better. How to compute replicability is described in **Measuring replicability**.

19. Algorithm has not converged

> Most older software has a default setting for the *number of iterations* that need to be modified.

Most cluster analysis and latent class analysis algorithms (excluding hierarchical clustering) work by choosing an initial sub-optimal partition of the data, and then seeking to improve upon this initial partition. Each attempt to improve on the partition is called an *iteration*.

The software stops trying to improve when either:

- The model has converged. This typically means either that:
  - The result cannot be improved. This is what is done with k-means cluster analysis.
  - The result cannot be improved more than a negligible amount. This is what is done with latent class analysis.
- A pre-specified maximum number of iterations has occurred.

It is not desirable to stop when the maximum number of iterations has occurred, as this suggests that the final result is not the best result that can be achieved. Nevertheless, this maximum number of iterations exists as a failsafe, to deal with situations where for some reason convergence is not practical. This is usually because either the amount of data is huge or the calculations are slow to perform.

Where software stops due to the number of iterations reaching the pre-specified maximum, a warning of some kind will typically be provided, saying something like “maximum number of iterations exceeded” or “model has not converged”. If this occurs, the fix is to increase the number of iterations. However, often this warning can be hard to spot.

With older software, such as SPSS and R, the pre-specified maximum number of iterations was determined in the 1970s, when compute resources were much more limited than today. As a result, it is advisable to always set the maximum number of iterations in such software to a large number, thereby avoiding the risk of missing a problem. For example, 1000. This is the default in Q and Displayr.

20. Local optima

> It is often beneficial to perform cluster analysis and latent class analysis multiple times, each time with a different random start point.

The first stage in most modern cluster analysis and latent class analysis algorithms is to initially partition the data in some way, and then seek to improve upon this partition. Most commonly this initial partitioning involves randomly assigning observations to a pre-specified number of clusters or classes.[14]

Often, the initial partitioning of the data determines the final solution. That is, one initial partitioning may lead to one result, and an alternative partitioning may end up with a different result. If the result is not the best of all possible results,[15] it is referred to as being a *local optimum.*

One way to decrease the chance of a result being a local optimum is to perform the analysis multiple times, each time using a different initial partitioning.

A useful analogy is to imagine being parachuted into a mountain range and then climbing ever upwards until you are at the highest point you can get to without ever heading down. The popular algorithms used are called *hill climbing algorithms* because all they can typically do is climb upwards. There is a good chance that such a strategy doesn't find the highest point, as unless you happen to land on the highest mountain, going upwards will not get you to the highest peak.

Although it is always good practice to run the analysis multiple times, it should be remembered that the goal of segmentation is to find a useful set of segments. There is no good reason to believe that the best solution from a segmentation perspective will be the one that best fits the data however, the better a solution fits the data the more likely it will be useful.

Software

Q's main latent class analysis algorithm (**Create > Segments > Latent Class Analysis**) allows the user to automatically increase the **Number of starts** in the **Advanced** section. You can also set the number of start points in **Create > Segments > K-Means Cluster Analysis** and **Create > Segments > Legacy Cluster Analysis**.

If using conjoint and MaxDiff latent class analysis software, code is required to change the number of starts (by running the algorithms in a loop).

In Displayr, **Anything > Advanced Analysis > Cluster > K-Means Cluster Analysis** allows the user to control the number of start points.

---

[14] In the case of Latent Class Analysis, this occurs by randomly generating parameters that then determine the class membership.

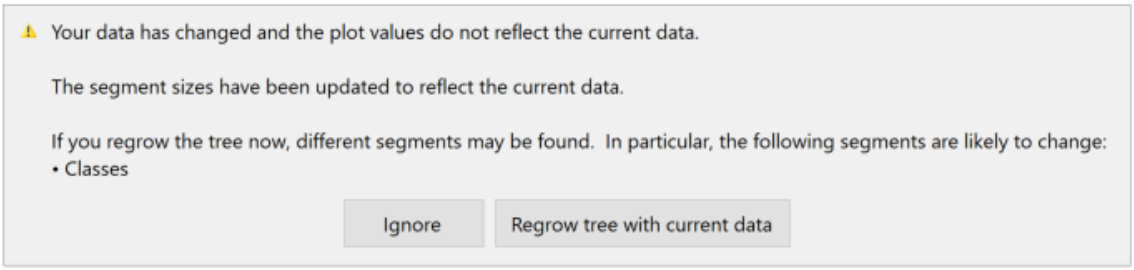
[15] Where *best* typically means the highest log-likelihood or lowest residual sum of squares.

21. Recreating segments in a new data set

It is often necessary to recreate segments in a new data set, such as a new wave of a study or a customer database.

Approach 1: Use the in-built tools in Q and Displayr

Q and Displayr have been designed to make it easy to automatically apply segments to new data sets. In Q, import the new data set (**File > Data Sets > Update**) and then click the **Ignore** button, and the new data will be allocated into segments.



This approach only works if the variable names and coding of the variables is identical in both data sets.

Approach 2: Write/extract code to automatically assign people to segments

Please refer to How to Allocate Observations to Segments in Excel.

Approach 3: Use a predictive model

For example, use a *random forest* or *linear discriminant analysis* to create a prediction function, where the outcome variable is the segment membership and the segmentation variables are the predictors. See **Predictive models** and **Segment typing tools.**

22. Segment typing tools

A *segment typing tool* is a tool designed to allow people to work out how to allocate people into segments.

What segment typing tools are used for

The main applications of segment typing tools are:

- When recruiting people to participate in qualitative research. The recruiter may email people asking them a series of questions. They then use the segment typing tool to work out which people are in which segment.
- To assign people into segments. For example, staff in a call center or shop may use a segment typing tool to assign prospects into segments so they can work out how best to sell to them.

Different statistical approaches for creating segment typing tools

The most straightforward approach to creating a segment typing tool is to create a predictive model and then use its predictions for segment typing. For example, use a *random forest* or *linear discriminant analysis* to create a prediction function, where the outcome variable is the segment membership and the segmentation variables are the predictors. See **Predictive models.**

A more complicated approach is to reproduce the algorithm used to form the segments (see How to Allocate Observations to Segments in Excel). Where practical – and it is often not – this approach is guaranteed to be most accurate, although the difference between this approach and an appropriate statistical model will be marginal.[16]

[16] For example, linear discriminant analysis is closely related to latent class analysis and can be identical, and can multinomial logistic regression and support vector machines are also closely related and usually similarly accurate.

Segment typing tool formats

Most segment typing tools are either:

- Tree diagrams, where the user answers questions and follows the relevant branch.
- Spreadsheets.
- Online tools. Where a predictive model has been used (see the previous section), and automatic segment typing tool is able to be created by pressing **DIAGNOSTICS > Create Similator.**

23. Measuring replicability

> Replicability is the extent to which a segmentation is likely to be re-discovered by another researcher.

Ideally, a segmentation can be reproduced by another person with a similar data set. This is particularly relevant in academic environments where it is evidence that you have discovered something of scientific interest. But, in commercial segmentations this is more of a nice to have, as in reality you can have multiple good segmentations of most data sets (e.g., gender, age, life stage). Furthermore, the most replicable segmentations tend to be uninteresting (e.g., yeah-saying biases replicate well) and can be local optima. So, while this chapter describes how to compute replication, it should not be assumed that this is necessarily a good idea.

Best approach: Compute bootstrap replication many times

Create multiple *bootstrap samples* by randomly sampling with replacement from the original data set, performing the analysis in each bootstrap sample, and comparing the result with the original result. The more consistent the solutions, the better.

This can be done automatically in Q using **Create > Segments > Legacy Cluster Analysis.** However, this algorithm assumes MCAR and numeric data, so use with care!

A computational trick for computing bootstrap replicability for other algorithms is to create a new variable which, for each case in the data file, shows the number of times it was randomly selected (i.e., 0, 1, 2, etc.). This trick both makes everything quite simple to do, and, allows incorporation of sampling weights.

In Q or Displayr:

- Insert a numeric JavaScript variable
- Set the **Access all data rows (advanced)**
- Paste in the code below as the **Expression**:

```
var wgt = new Array(N); // Creating an array to
                        // store the variable.
for (var i = 0; i < N; i++) // Setting the
                            //initial values to 0.
    wgt[i] = 0;
// Using random sampling with replacement to
// count up the number of times to replicate
// the data.
var seed = 1;
function random() { // Slightly dodgy
```

```
// see http://stackoverflow.com/a/19303725/1547926
    var x = Math.sin(seed++) * 10000;
    return Math.floor((x - Math.floor(x)) * (N - 1));
}
var counter = 0;
while (counter++ < N)
    wgt[random()]++; // Incrementing the number
              //of times the value has been selected

// Replace 'WEIGHT' with the name of your weight
// variable. If there is none, delete the next 3 lines – be sure to keep
'wgt' at the end.
for (var i = 0; i < N; i++) {
    wgt[i] *= WEIGHT[i];
}
wgt
```

- Tag the variable as a weight (in Q, toggle on a **W** tag for the variable in the Variables and Questions tab. In Displayr, select the variable in the Data tree and toggle **usable as a weight**).
- Re-run the clustering or latent class analysis with this weight on.

Second-best approach: Compute the bootstrap replication a small number of times (e.g., once)

This is the same as the previous but only done a few times.

Third-best approach: Split-sample replication

This is like the bootstrapping approach, except that the initial sample is split in half, the analysis is run on the two halves, and compared. Although this is the traditional approach in market research for assessing replicability, it is decidedly inferior, as:

- It checks if the segmentations performed in the half-samples are replicable, which is no guarantee that the actual result of interest is replicable.
- As the half-samples are smaller they are more susceptible to sampling error, so replication may incorrectly appear not to exist.

24. Increasing replicability

> There are techniques that can increase the replicability of a segmentation.

If it is desired, the replicability of a segmentation can be improved. However, there is always a cost to doing this. One can either seek to find the segmentation that best fits the data, or the one that is most replicable, but not both. That is, a focus on replicability is a choice to search for a local optimum.

Best approach: Use algorithms that search for replicable solutions

Various algorithms have been developed that seek to find solutions that are replicable. For example, *bagged clustering*[17] performs *k*-means on multiple bootstrapped samples, and then uses hierarchical cluster analysis to group the cluster centers together.

These algorithms are also known as *consensus clustering algorithms* (e.g., Sawtooth Software's CCEA) and *ensemble clustering algorithms*.

Bagged clustering is performed by the `bclust` function in the `e1071` R package, which is available in both Q and Displayr.

Second-best approach: Create lots of segmentations and compare them in terms of replicability

Lots of segmentations can be run and compared based on their replicability. This can be done automatically in Q using **Create > Segments > Legacy Cluster Analysis.** However, as mentioned in the previous chapter, this algorithm assumes MCAR and numeric data, so use with care!

[17] Friedrich Leisch. Bagged clustering. Working Paper 51, SFB "Adaptive Information Systems and Modeling in Economics and Management Science", August 1999. http://www.ci.tuwien.ac.at/~leisch

25. Quickly profiling segments

One of the key pieces of information necessary to evaluate the usefulness of a segmentation is the extent to which the segment is correlated with other data. For example, if the profile of each segment is different in terms of age, gender, and behaviors, this often makes execution of different marketing strategies to the different segments more straightforward.

This chapter describes how to quickly profile large numbers of segments.

There is a three-step process for quickly profiling lots of segments:

1. Create lots of crosstabs.
2. Summarize these crosstabs as a table.
3. Visualize the summary as a heatmap.

Step 1: Create lots of crosstabs

Creating lots of crosstabs in Q

There are several ways of creating lots of crosstabs in Q. For example:

1. **Insert > Tables > Basic Tables**
2. Select the segment membership variables as **Key questions**
3. Select the variables that you wish to profile the segments by (e.g., age, gender, media usage, product usage) as the **Profiling questions**
4. Choose **Nowhere (don't generate SUMMARY tables)**
5. Press **OK**.

This will create a series of folders containing crosstabs.

Creating lots of crosstabs in Displayr

1. **Anything > Report > Summary Report**
2. Select **Detailed report breaking down every variable set by a list of key variable sets** and press **Next >**.
3. Select the segmentation variables and press **Next >**.
4. Choose your desired output. The table (bottom left) is likely the best choice and press **Create Report**.

This will create a series of pages, each of which will contain a crosstab of each question in the study by the alternative segmentations.

Step 2: Summarize the crosstabs as a table

The table below shows a crosstab of work status by a two-segment latent class analysis solution. In a typical segmentation study there can easily be 50 or more such tables. An alternative to reading through all the studies and reading each is just to compute the number of tables that contains statistically significant results.

Column %	WORKING FULLTIME	WORKING PARTTIME	TEMP NOT WORKING	UNEMPL, LAID OFF	RETIRED	SCHOOL	KEEPING HOUSE	OTHER	NET
Segment 1	55%	54%	55%	59%	53%	40%	48%	55%	53%
Segment 2	45%	46%	45%	41%	47%	60%	52%	45%	47%
NET	100%	100%	100%	100%	100%	100%	100%	100%	100%

In Q and Displayr it is possible to select all the tables of interest (i.e., those created in the previous step), and create a summary table which shows the statistical significance of all of the tables. The first six columns of such a table are shown below. The number in each of the cells represents a *z-score,* computed form the *p*-value. That is:

- The higher the *z*-score, the more statistically significant the relationship between the segmentation and the profiling variable (i.e., the smaller the *p*-value).
- Values of 1.96 indicate a significant relationship at the 0.05 level.
- *z*-scores are shown instead of *p*-values as *z*-scores are easier for comparison purposes (i.e., it is intuitively obvious to most people that a *z* of 4 is much bigger than one of 2, than to compare 0.00006 with 0.0455)
- The *z*-scores are capped at 5.

	Wealth	Own Stoc-ks	Owns Business	Has Depts	Labor force stat-us	Hours	G
2 Unscaled 100 starts	1.6	.8	.2	1.5	2.2	.4	
7 Standardized & Rescaled	2.9	3.0	2.1	1.6	4.1	2.0	
2 Standardized & Rescaled	1.8	.6	.6	1.3	2.8	2.3	
2 Standardized	3.9	1.7	1.3	.8	3.1	1.5	
2 Unscaled	1.7	.7	.0	1.3	2.1	1.2	

Creating a summary table of crosstabs in Q

- Select all the tables
- **Automate > Browse Online Library > Significance Testing in Tables > Identify Interesting Tables**

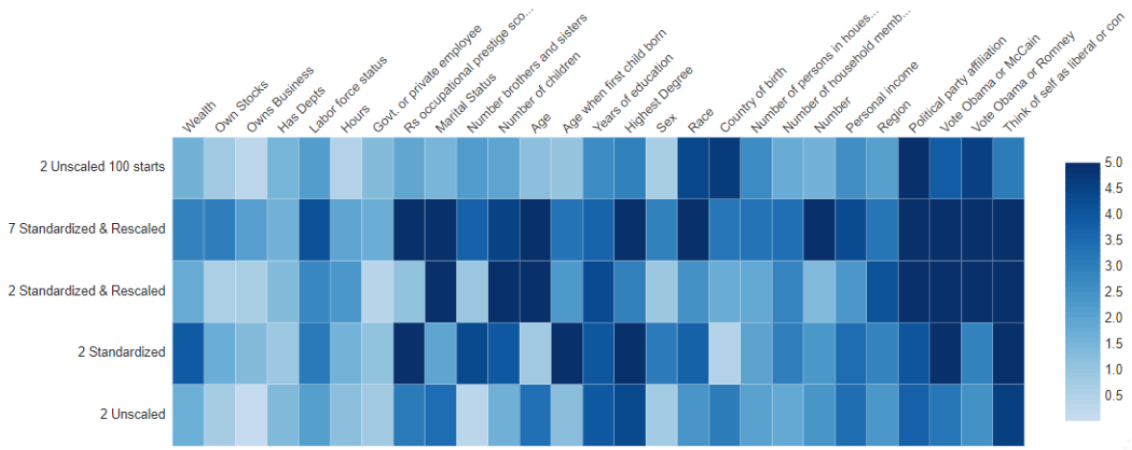
Please note that this table will not update if the inputs to it change. It needs to be re-created (this is just to reduce computational time).

Creating a summary table of crosstabs in Displayr

- Select all the tables
- **Anything > Report > Identify Interesting Tables**

Please note that this table will not update if the inputs to it change. It needs to be re-created (this is just to reduce computational time).

Step 3: Visualize the summary tables as a heatmap



From the heatmap we can readily see that the segmentation in row 2 (7 Standardized & Rescaled) has much higher *z*-scores on average than the other segmentations, suggesting it is, all else being equal, the best of the segmentations. By more carefully examining the heatmap we can also see qualitative differences between the segmentations in terms of how they relate to the profiling data.

Creating the heatmap in Q

- **Create > Chart > Visualization > Heatmap**
- Set **Output** to `most.significant.results` (this is the name of the table created above; if you have created multiple-such tables, it will have different names)

Creating the heatmap in Displayr

- **Visualization > Heatmap**
- Set **Output** to `most.significant.results` (this is the name of the table created above; if you have created multiple-such tables, it will have different names)

26. Increasing the predictability of the segments

This chapter describes what to do when segments are not correlated with other variables of interest (e.g., demographics).

Sometimes segments do not correlate sufficiently strongly with any profiling data. This can make implementation of the segmentation difficult. This chapter describes various strategies for improving the predictability of a segmentation.

The term *segmentation variables* is used to refer to the variables that we initially wish to use when forming the segments (e.g., attitudes, preferences, utilities). *Profiling variables* are the variables that we want to be correlated with the segments (e.g., age, gender, media usage).

Of all the issues addressed in this eBook, this is one that in practice researchers are most likely to get hideously wrong. The last of the strategies – including profiling variables as segmentation variables – has the unfortunate characteristic that it is almost always a terrible strategy, but it is often looks successful even when it is not.

Best strategy: Upweight variables that are correlated with other variables

Inevitably some of the segmentation variables (i.e., the variables that we are using in the cluster or latent class analysis) are more closely related to the profiling variables (i.e., the variables that we want to be correlated with the segments). We can therefore increase the predictability of a segmentation by ensuring that these variables are the main variables used when forming the segments.

There are a few variants of this strategy. Provided that some of the segmentation variables are correlated with the profiling variables, they all work. The easiest of the strategies is 1a. It is not clear which of the strategies is best.

Strategies for doing this include:

1. Use the technique described in the previous chapter (**Quickly profiling segments**) to work out which of the segmentation variables are related to the profiling variables and either:
    a. Use only these variables when forming the segments.
    b. Give a higher weight (i.e., range) to the variables with stronger relationships a larger range (see **Segments differ on only a few variables**).

    c. Use judgment to form segments using the profiling variables that are closely related to lots of segmentation variables (e.g., age, frequency of usage, household structure).
2. Build a series of **Predictive models** where the segmentation variables are the outcome variables, and the they are the *outcome variable* and the *profiling variables* (i.e., the variables that we want to be correlated with the segments) are the *predictors* and either:
    a. Give a higher weight (range) to variables with better predictive accuracy.
    b. Save the predicted values from the predictive models and use these as the segmentation variables.

2nd best strategy: Use dimension reduction techniques

This strategy has several steps:

1. Use a dimension reduction technique such as PCA or multiple correspondence analysis to jointly analyze the segmentation variables and the profiling variables.
2. Save the components/dimensions from the analysis.
3. Use cluster analysis or latent class analysis to form groups using the components. There are a number of different ways to explore when doing this:
    a. Rescale the components so that its standard deviation is proportional to the variance explained in the dimension reduction.
    b. Do not use components that are not correlated with the segmentation variables.
    c. Do not use components that are correlated with the segmentation variables but not with the profiling variables.

3rd best strategy: Multivariate predictive trees

Algorithms have been developed which create a “tree” like structure, predicting one or more outcome variables, based on one or more predictor variables. The most well-known of these are CHAID and

CART, which predict a single numeric or categorical variable. However, these have also been developed for analyzing multiple dependent variables at the same time.

Multivariate predictive trees in Q

Create > Classifier > Mixed-Mode Tree

Multivariate predictive trees in Displayr

Anything > Advanced Analysis > Machine Learning > Mixed-Mode Tree

4th best strategy: Use concomitant/covariate variable latent class algorithms

Latent class analysis software has been developed which explicitly includes predictors. However, our experience has been that such models tend not to have any real impact on the relationship between the profiling variables and the segments, and as such we have not included them in either Q or Displayr.

Worst strategy: Add demographics or other predictor variables to the segmentation

It is common for market researchers to include profiling variables as segmentation variables. While this can work, it is a high-risk approach. It can also lead to a lot of implementation problems, whereby the segments that are described in the research end up being very different to those that are experienced in the implementation.

The problem with this approach is essentially that while it ensures that the segments that are identified are strongly related to the profiling variables, several problems can occur:

- No real differences are found on the segmentation variables. That is, the segmentation may be in practice be driven entirely by the profiling variables, and researchers inadvertently end up exaggerating trivial differences on the segmentation variables when presenting the results.
- The resulting segments focus on the predictor variables that are most closely correlated with each other, rather than on the predictor variables that are most closely correlated with the segmentation variables.
- There are much cleaner/neater segmentations that get missed. For example, if the segmentation is driven by age and age-correlated variables, it is usually better to then create segments purely based on age, than to have segments that are highly correlated with age (e.g., 85% of young people being in a segment).

27. Finding the most managerially useful segments

If you follow all the steps above, you can still end up with an entirely uninteresting segmentation. The steps above are all designed to address statistical issues, but segmentation is ultimately about strategy and implementation.

The previous 24 issues have focused on various technical aspects of segmentation. Sadly, segmentation is not so simple as to be able to be addressed by a checklist. Ultimately, a good segmentation is one that is managerial useful, and there is no guarantee that addressing various technical issues will achieve this end.

The secret to successful segmentations is to:

- **Generate lots of candidate segmentations.** That is, do not just do the best approaches described above. Also generates lots of alternative solutions. For example:
    - Investigate different numbers of segments
    - Change the data used
    - Increase the number of starts
    - Use **Dimension reduction techniques**
    - Standardizing the data in a different way
    - If using latent class analysis with numeric data, change the various distributional assumptions for latent class analysis (Q only)
    - Using a different algorithm
- Systematically compare the solutions. As discussed in **Working out the best number of segments**, we can compare based on:
    - How strongly do the segments relate to other data?
    - The fewer segments the better
    - Are the segments easy to name?
    - Are the segments inspirational?
    - Are we sure the segment differences aren't due to response biases?
    - Perhaps: replicability

This process should take days, and a systematic process should be used. It is often useful to create a table that lists the strengths and weaknesses of all the alternative segmentations. For example, for the GSS study, I created the table below. The goal is to systematically evaluate all the candidate segmentations, identifying the one that is, overall, best.

	Significant relationships with other data (z-Scores > 3)	Easy to name	How inspirational
2 Unscaled	11	Moderate	No obvious implications
2 Standardized	14	Moderate	No obvious implications
2 Standardized & Scaled	10	Poor	No obvious implications
7 Standardized & Scaled	22	Easy	Clear ideas for strategy
3 Standardized & Scaled	15		
3 Standardized & Scaled + 100 start points			
Etc.			

DISPLAYR

RESEARCHSOFTWARE
A DIVISION OF DISPLAYR

Want to cut your analysis and reporting time in half?

See Displayr in action ⟶

Analysis and reporting software built to save you time