5 Grid display and analysis

The buttons at the bottom of any pane of the RepGrid window provide access to various grid display and analysis functions (Figure 77) which will be described in the following sections.

Figure 77: RepGrid analysis buttons

The buttons may have a small icon on the left or right. Clicking in the body of the button generally performs the analysis immediately. When the mouse cursor is over the button icon on the right of a button it changes to a button shape, and clicking brings up a dialog enabling the analysis parameters to be changed. When the mouse is over the menu icon on the left of a button it changes to show a menu symbol, and clicking on it evokes a popup menu enabling the function of the button to be changed.

5.1 Display: Plotting the grid as a matrix of ratings of elements on constructs

Clicking on button icon on the right of the ***Display*** button brings up a dialog which controls the way in which the content of the grid is displayed as a matrix (Figure 78).

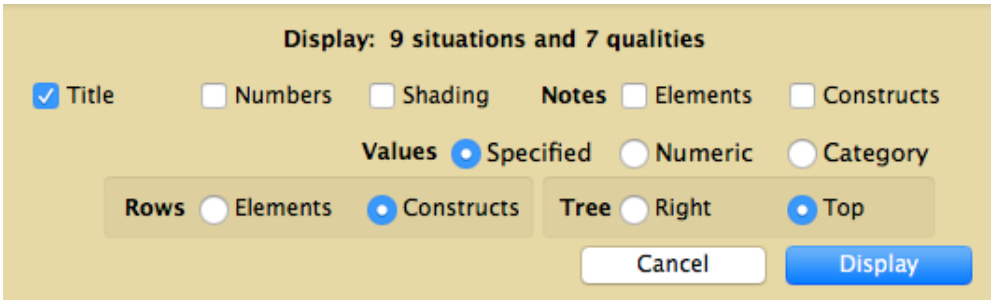


Figure 78: RepGrid *Display* dialog

The row of check boxes at the top determine whether the plot is titled, whether the elements and constructs are numbered, whether the ratings are shaded (to indicate the top third of high values and bottom third of low values), and whether the notes attached to elements and/or constructs are shown.

The ***Values*** panel determines whether numeric or categorical values should be displayed: as specified by the ***Use in plots*** checkbox in the construct dialog, or temporarily overriding that setting for all the constructs.

The ***Rows*** panel determines whether the matrix of grid data is displayed with elements or constructs as rows. For conceptual grids with only rating scale constructs it is conventional that constructs are displayed as rows, but where categorical values are displayed the plot has a more condensed format is the constructs are displayed as columns.

5.1.1 *Display plot output*

Figure 79 shows the plot produced when one clicks the ***Display*** button with the settings above. The title, constructs, elements and ratings are shown.

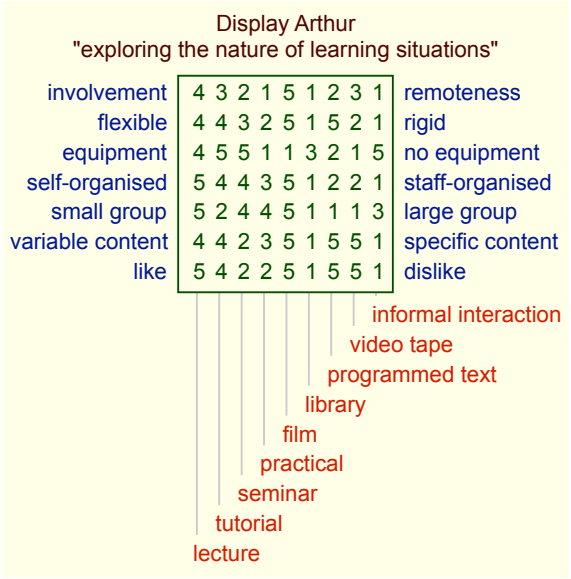


Figure 79: *Display* of the data in a grid as numbers

Figure 80 shows the plot produced when one specifies the values should be shown as categories and that the rows should be elements. Note that when no category applies to the middle value of the scale then none is shown.

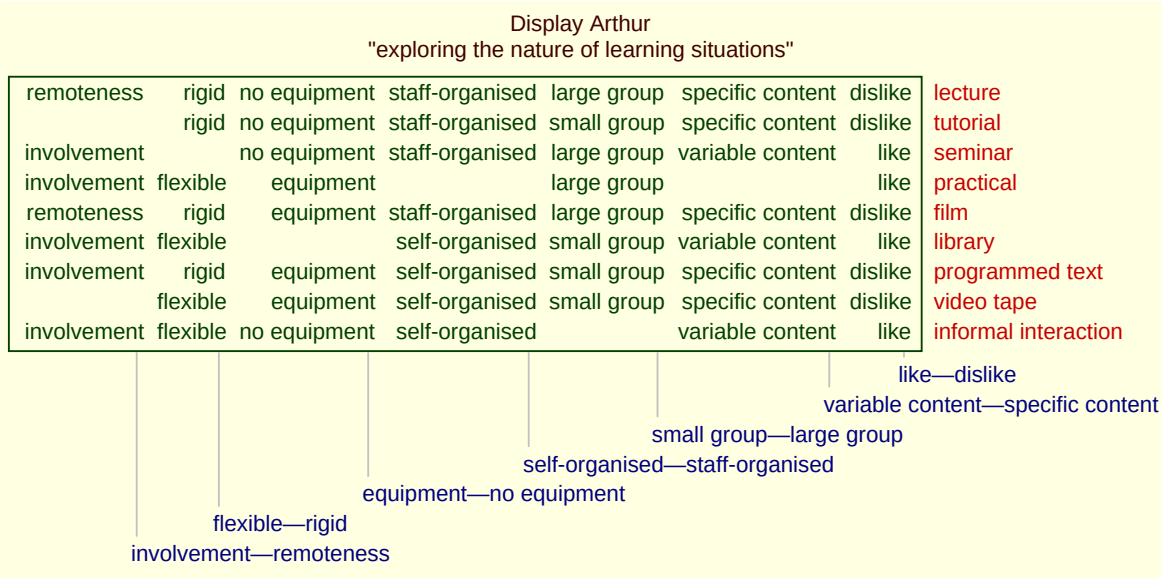


Figure 80: *Display* of the data in a grid as categories

Figure 81 shows a display of the house choice grid used in earlier examples which uses four types of construct.

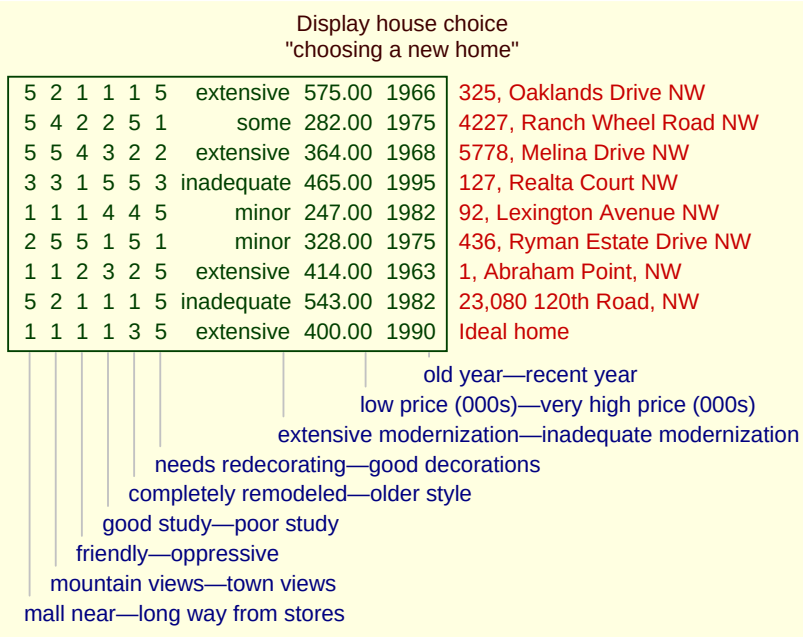


Figure 81: *Display* of the data in a mixed-type grid—constructs as columns

This mixed-type grid is displayed with constructs as columns as this is more compact than a display with constructs as rows shown in Figure 82.

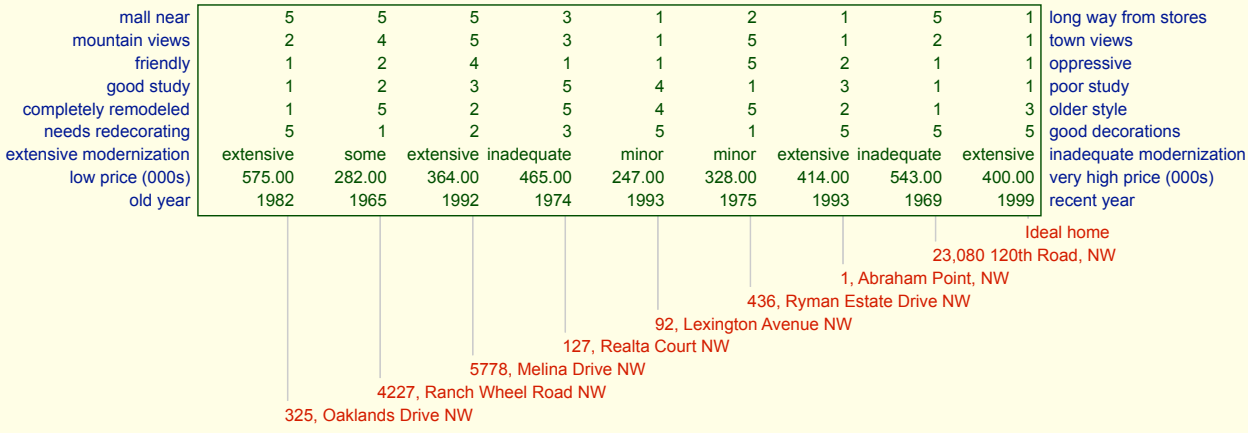


Figure 82: *Display* of the data in a mixed-type grid—constructs as rows

5.1.2 *Including classes as elements or constructs in the analyses*

As discussed in §3.4.7, classes may be converted to ***ideal elements*** or ***compound constructs*** as part of the grid being analyzed, participate in the analysis process and be shown in the plots. Figure 47 in

that section shows how the contact lens grid used to illustrate classes in §3.4.1 may be displayed with the classes represented as ideal elements. Figure 83 shows the same classes displayed as compound constructs. The pole name are labelled with the mathematical symbols ∈ for ***is an element of*** and ∉ for ***is not an element of*** to indicate whether an element is, or is not, classified under the class specified.

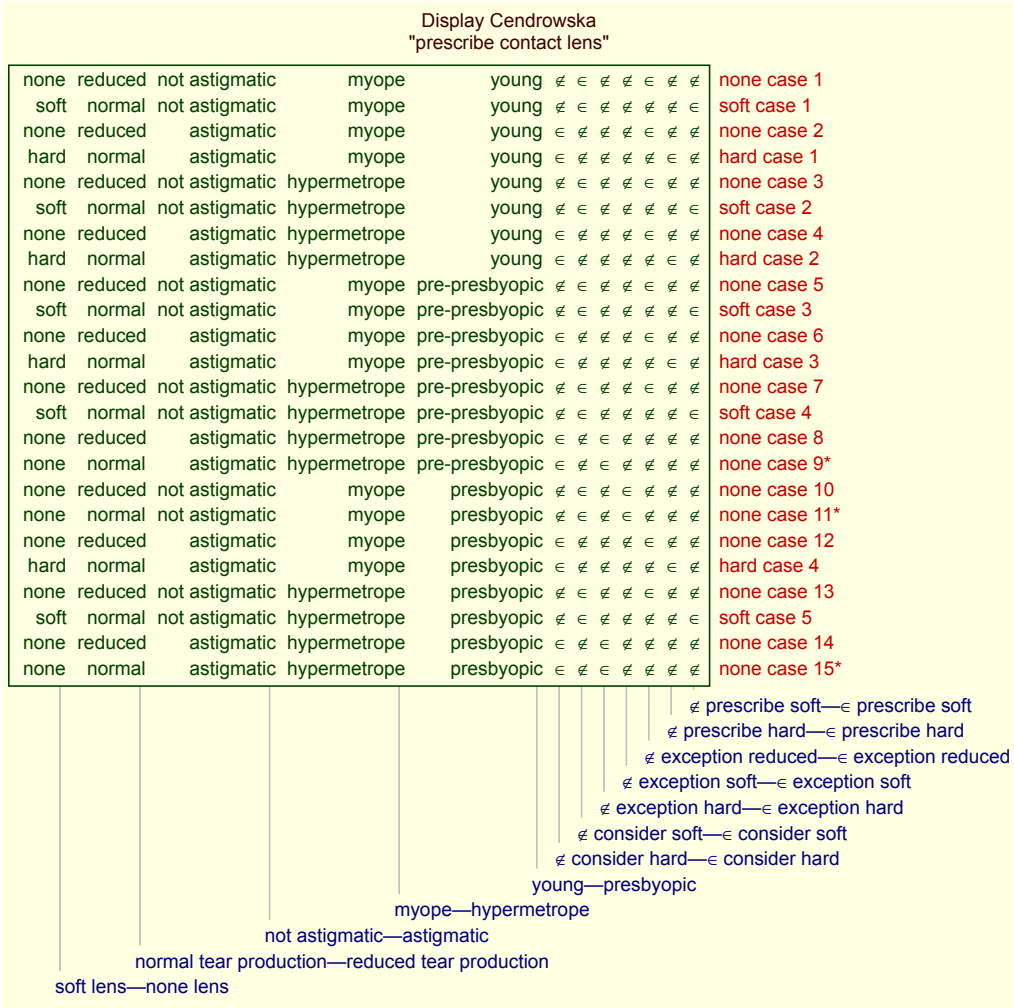


Figure 83: *Display* with classes included as compound constructs

The other analysis programs treat the elements or constructs representing classes as if they were normal parts of the grid and the inclusion of classes for each will not be illustrated except for the ***Matches*** analysis where it is insightful to examine the relation between ideal elements that could be entered without the availability of classes and those deriving from classes (§5.7.1).

5.2 Synopsis: Histograms and scree plot

Display presents the grid as a matrix of ratings of elements on constructs and is useful for examining the grid data, for example, in checking data entry. There are other assessments of the data in a grid

where an alternative form of display is useful, for example, how the element ratings are distributed across the possible values; are they skewed, have both poles been used, and so on. A histogram of the distribution of the element ratings on a construct supports rapid appraisal of such considerations. One may also be interested in the ***complexity*** of cognition exhibited; are all the constructs similar in the distinctions they make, how many distinctions are being instantiated, and so on.

The ***Synopsis*** presentation provides a different presentation of the grid data that addresses these issues by displaying histograms of the distribution of the element ratings on the constructs and a scree plot of the variance in the data accounted for by its principal components.

Clicking on button icon on the right of the ***Synopsis*** button brings up a dialog which controls the way in which the output is displayed (Figure 84). There are two panels, one managing the output of the histograms, and the other the scree plot of the principal components.

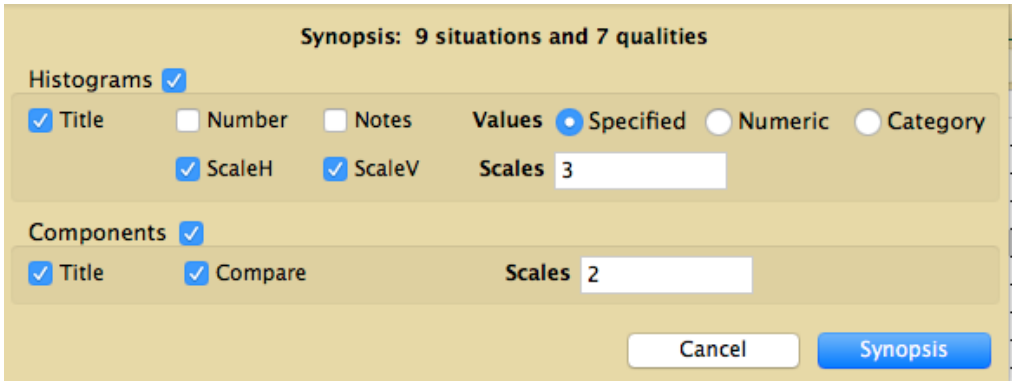


Figure 84: RepGrid *Synopsis* dialog

The check boxes on the histograms panel determine whether: the plot is titled; the constructs are numbered; the notes attached to the constructs are shown; a horizontal rating value scale and/or a vertical count scale are shown. The radio buttons determine whether the element labels should be as specified by ***Use in plots*** or all numeric or categorical. The text field on the right enables the geometry of the histograms to be adjusted. It can contain up to 5 numbers separated by commas: the number of pixels for each vertical increment in the histogram bars; their widths; horizontal space between bars; vertical space between histograms, and horizontal space at the end of each plot. If fewer than 5 values are specified the remainder are filled from the appropriate position in the default values *3, 3, 8, 4, 4*.

The check boxes on the components panel determine whether: the plot is titled; Frontier's (1976) comparative plot of the distribution if the principal components were randomly generated should also be plotted to provide an estimate of the number of significant components. The text field on the right enables the geometry of the scree plot to be adjusted. It can contain up to 4 numbers separated by commas: whether the vertical numeric increments are 1 or 2; vertical separation between scale points; horizontal separation between scale points; horizontal space at each end of plot. If fewer than 4 values are specified the remainder are filled from the appropriate position in the default values *2, 3, 20, 10*.

In both cases there is rarely a need to change the detailed geometry but the options are available to fine-tune the output for presentation or publication. Mousing over either the text field brings up help text giving information about available parameters.

5.2.1 *Synopsis histogram and scree plots output*

Figure 85 shows the plot produced when one clicks the ***Synopsis*** button with the settings above for the grid displayed in Figure 79.

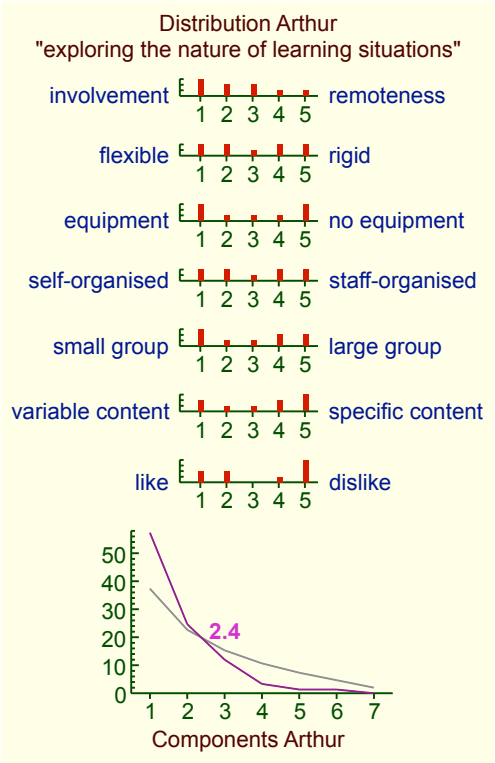


Figure 85: *Synopsis* of grid data

Figure 86 shows the plot produced when one clicks the ***Synopsis*** button with the settings above for the grids displayed in Figure 81. The histograms on at the bottom illustrate how construct categories are shown in histograms.

Both grids were elicited using PEGASUS-style elicitation with feedback of matches and the histograms show that the ratings on all constructs are fairly evenly distributed with poles used. Grids elicited in other ways may not have these characteristics, such as those with given constructs that those filling in the ratings do not normally use within their own constructions.

The scree plots suggest that the first grid exhibits some 2 significant underlying dimensions, and the second right 3. This provides a rapid assessment of the cognitive complexity of the grids—a full PrinGrid analysis (§5.5) would enable the nature of the underlying distinctions to be investigated. Note that the estimate of significant dimensions does not necessarily indicate that some constructs

Figure 86: *Synopsis* of grid data with mixed types

are redundant but more likely that there are insufficient elements to discriminate between all the constructs.

5.3 Focus: Sorting by similarity and hierarchical clustering

Shaw's (1980) Focus algorithm sorts the rows and columns of the grid display to bring similar elements and similar constructs closer together, and also shows the hierarchical structure of similarities that results from sorting the grid in this way.

Clicking on button icon on the right of the ***Focus*** button brings up a dialog that controls the Focus analysis which can be presented as a graphic plot or as textual data (Figure 87).

The upper pane controls the presentation of the graphic plot. The row of check boxes at the top determine whether: a plot is produced; it is titled; the elements and constructs are numbered; the ratings are shaded (to indicate the top third of high values and bottom third of low values); the notes attached to elements and/or constructs are shown.

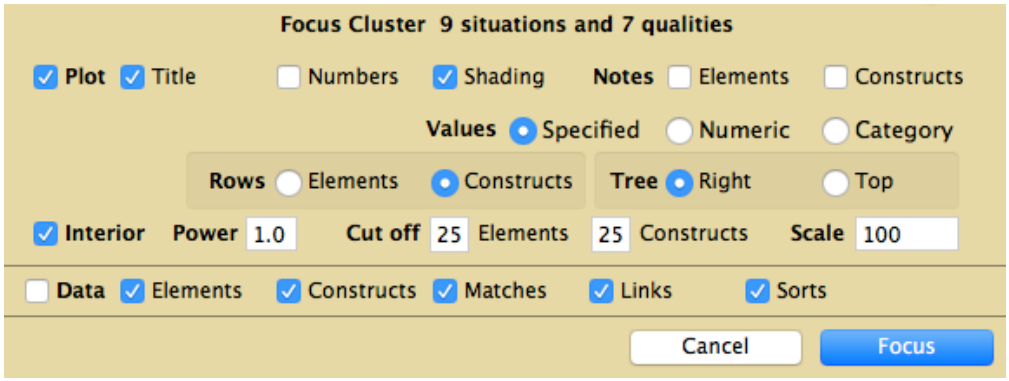


Figure 87: RepGrid *Focus Cluster* dialog

The ***Values*** panel determines whether numeric or categorical values should be displayed: as specified by the ***Use in plots*** checkbox in the construct dialog, or temporarily overriding that setting for all the constructs.

The ***Rows*** panel determines whether the matrix of grid data is displayed with elements or constructs as rows. The ***Tree*** panel determines whether Focus cluster tree for the columns is shown at the top of the grid or at the lower right.

The ***Interior*** check box controls the Focus matching strategy. Leaving it unchecked specifies the standard Focus algorithm in which items are matched only against the items at the edges of existing clusters. This sometimes leads to items with a high match to interior items being shown as having a lower match to an edge item. Checking the ***Interior*** check box allows Focus to match against interior items in an existing cluster; it then displays the interior match and places the item at the edge of that cluster that has highest match to the item.

The ***Power*** value determines the exponent used in the Minkowski metric used to compute matching scores (Shaw, 1980, p.160). The default (and generally recommended) power of 1.0 defines the standard city block metric normally used in the Focus algorithm. A power of 2.0 defines a Euclidean metric. Fractional powers in the range 0.1 to 10.0 may be used—a higher power weights larger differences more than smaller ones, and ***vice versa***.

The ***Cut off*** values determine the level of match below which an element or construct cluster will not be shown. The ***Scale*** value determines how much space will be allocated to the trees showing the cluster hierarchies.

The row of check boxes near the bottom determine whether: textual data from the analysis is displayed; element and construct data are output; match matrices, cluster links, and sorts are output.

5.3.1 Focus cluster plot output

Figures 88 and 89 shows the Focus cluster plots produced for the grids displayed in Figures 79 and 81 when ***Focus*** button is clicked with the settings above. The grids have been sorted to bring closely matching elements together, and closely matching constructs together.

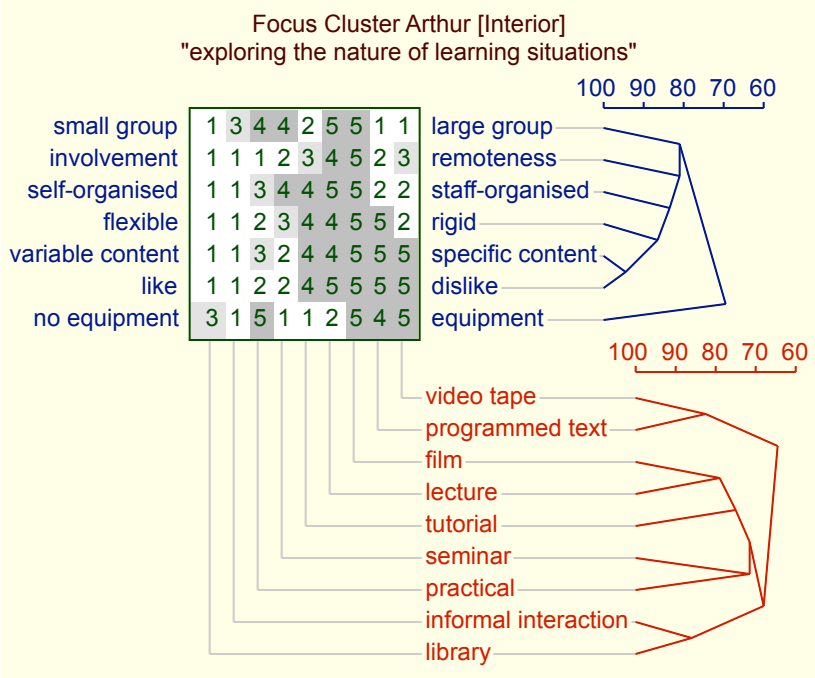


Figure 88: *Focus Cluster* analysis

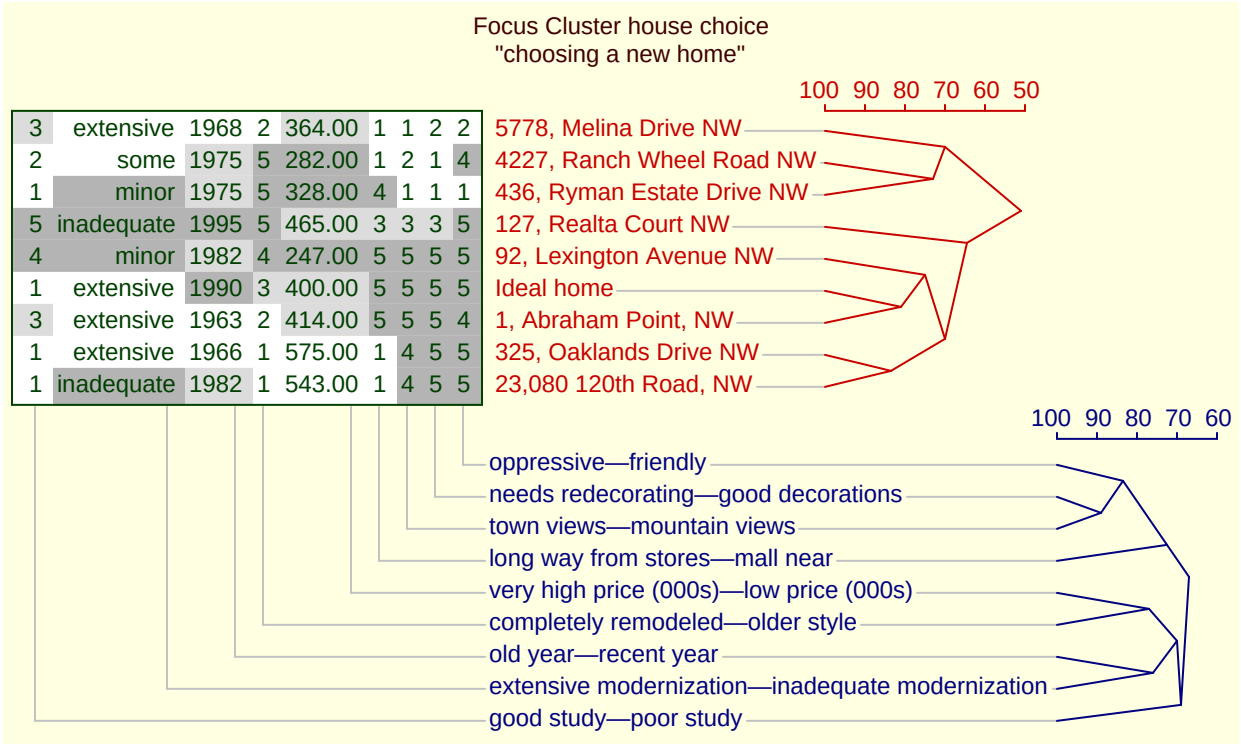
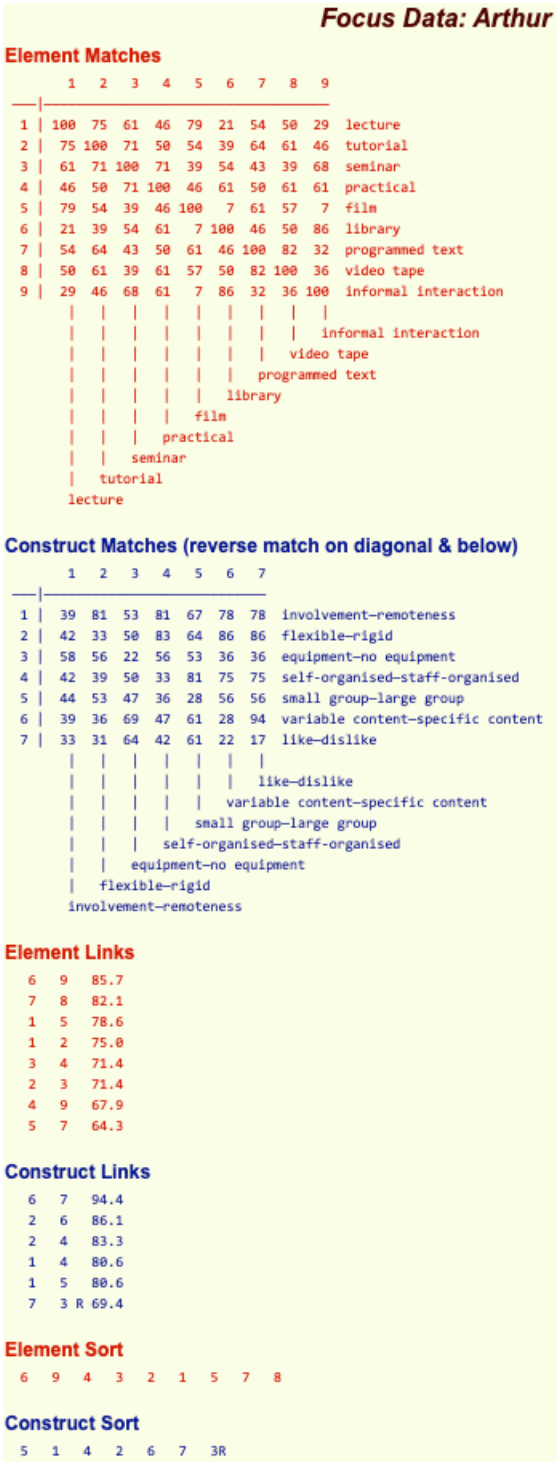


Figure 89: *Focus Cluster* of a mixed-type grid—constructs as columns

5.3.2 *Focus data output*

Figure 90 shows the data underlying the Focus analysis of Figure 88 when the ***Data*** checkbox is set in the Focus dialog.



Focus Data: Arthur

Element Matches

```
      1   2   3   4   5   6   7   8   9
 ---|-------------------------------------
 1 | 100  75  61  46  79  21  54  50  29  lecture
 2 |  75 100  71  50  54  39  64  61  46  tutorial
 3 |  61  71 100  71  39  54  43  39  68  seminar
 4 |  46  50  71 100  46  61  50  61  61  practical
 5 |  79  54  39  46 100   7  61  57   7  film
 6 |  21  39  54  61   7 100  46  50  86  library
 7 |  54  64  43  50  61  46 100  82  32  programmed text
 8 |  50  61  39  61  57  50  82 100  36  video tape
 9 |  29  46  68  61   7  86  32  36 100  informal interaction
       |   |   |   |   |   |   |   |
       |   |   |   |   |   |   |   informal interaction
       |   |   |   |   |   |   video tape
       |   |   |   |   |   programmed text
       |   |   |   |   library
       |   |   |   film
       |   |   practical
       |   seminar
       tutorial
      lecture
```

Construct Matches (reverse match on diagonal & below)

```
      1   2   3   4   5   6   7
 ---|-----------------------------
 1 |  39  81  53  81  67  78  78  involvement-remoteness
 2 |  42  33  50  83  64  86  86  flexible-rigid
 3 |  58  56  22  56  53  36  36  equipment-no equipment
 4 |  42  39  50  33  81  75  75  self-organised-staff-organised
 5 |  44  53  47  36  28  56  56  small group-large group
 6 |  39  36  69  47  61  28  94  variable content-specific content
 7 |  33  31  64  42  61  22  17  like-dislike
       |   |   |   |   |   |   |
       |   |   |   |   |   |   like-dislike
       |   |   |   |   |   variable content-specific content
       |   |   |   |   small group-large group
       |   |   |   self-organised-staff-organised
       |   |   equipment-no equipment
       |   flexible-rigid
      involvement-remoteness
```

Element Links

```
 6   9   85.7
 7   8   82.1
 1   5   78.6
 1   2   75.0
 3   4   71.4
 2   3   71.4
 4   9   67.9
 5   7   64.3
```

Construct Links

```
 6   7   94.4
 2   6   86.1
 2   4   83.3
 1   4   80.6
 1   5   80.6
 7   3 R 69.4
```

Element Sort

```
 6   9   4   3   2   1   5   7   8
```

Construct Sort

```
 5   1   4   2   6   7   3R
```

Figure 90: Data underlying a *Focus Cluster* analysis

The element and construct ***Matches*** are shown as a percentage of the maximum possible match, and it should be noted that the values on the diagonal and below it in the construct match matrix correspond to the match when one of the constructs is reversed in its values. The ***Links*** data corresponds to the clusters in the plot. The ***R*** indicates that the Focus algorithm has reversed the construct in determining the highest matches. The ***Sort*** data indicates how the algorithm has sorted the grid data to produce the Focus plot.

5.3.3 Status of the Focus hierarchical clusters

This manual does not cover the detailed interpretation of grid analyses but there several books that do so (Shaw, 1980, 1981; Shaw and McKnight, 1981; Pope and Denicolo, 2001; Jankowicz, 2004).

One issue that may arise is the meaningfulness of the cluster structure. Most grid datasets are too small for meaningful statistical analysis, but Heckmann and Bell (2016) have shown how ***bootstrap*** techniques (Efron and Tibshirani, 1993; Davison and Hinkley, 1997) may be used to investigate the robustness of the clusters against perturbation of the grid data. Bootstrap techniques are non-parametric in making no assumptions about the distribution from which the data is drawn, but treat the actual dataset as a sample fully representing that distribution and run the analysis many times against samples of that dataset to estimate the sensitivity of the full analysis to partial datasets.

Users may make a similar analysis in RepGrid by selecting different subsets of elements and constructs and comparing the resulting plots to provide some insight into the dependency of the results on the data provided, perhaps leading to further elicitation if there are concerns about the robustness of what seem to be interesting structures.

Note also that the elicitation techniques used may have a major impact on the representativeness of the grid data and the results of analysis. Shaw's (1980) computer-based elicitation techniques continually analyze the data, feed back high matches between elements and between constructs, and suggest the type of constructs or elements that the elicitee might add to reduce them (§4.2). This is designed to ensure that the clusters shown in analysis are not artefacts of inadequate coverage of the domain.

It is informative to test the cluster analysis on grids with a known hierarchical structure to determine whether it reproduces that structure. Shaw and Gaines (1998) did this for artificial and natural datasets where hierarchical structures were represented in grids. Figure 91 shows the simple hierarchy they used as an initial test.

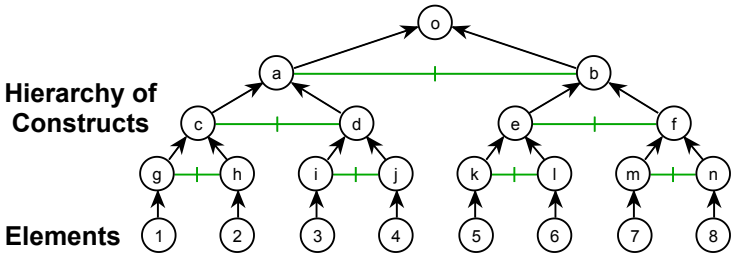


Figure 91: Simple hierarchy

Figure 92 shows this simple hierarchy represented as grid and then clustered using the Focus algorithm. It can be seen that Focus has reconstructed the original hierarchy from the grid representation. Indeed it can be shown from a mathematical analysis that it will necessarily do so for such hierarchical structures.

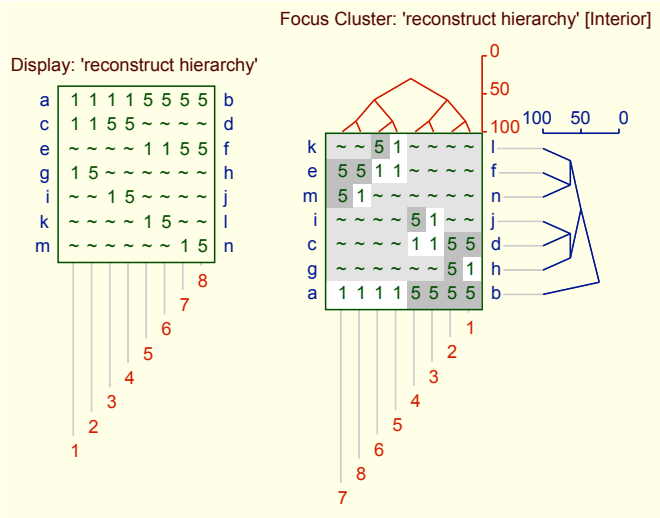


Figure 92: Simple hierarchy represented in a grid, *Display* and *Focus Cluster*

The metavalue ~ ***Inapplicable*** has been used in representing the ordinal relationships in the hierarchy as discussed in §3.3.6 on ordinal relations in grids. Yorke (1978; 1983) has noted that the middle value of a rating scale is ambiguous in being used for multiple purposes such as inapplicable, and it is interesting to see the impact of doing so with this dataset.

Figure 93 shows the grid with the midpoint value 3 replacing the metavalue ~ together with the resulting Focus analysis. Again it can be seen that Focus has reconstructed the original hierarchy suggesting that the overloading of the midpoint of a rating scale may not be a significant problem in practice.

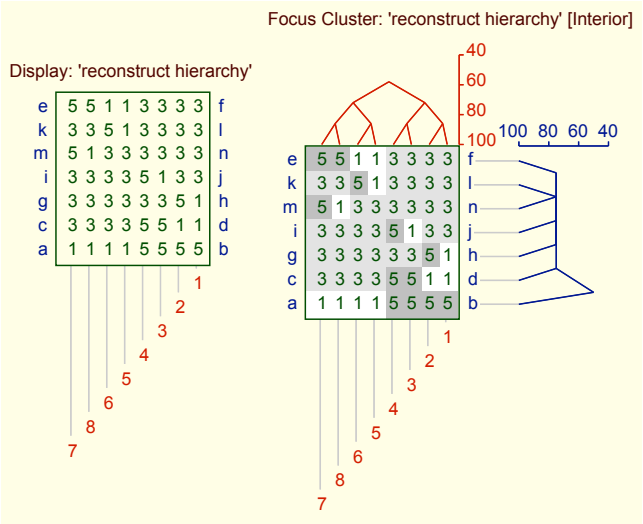


Figure 93: Simple hierarchy represented in a grid with no metavalues

The simple hierarchy is artificial and it is interesting to take a less well-structured example from the literature and test it in the same way. Figure 94 is extracted from a paper analyzing the relations between the early Internet services shortly after the Internet was commercialized in 1995 (Gaines et al., 1997). It may be seen as an example of Plato's ***collection and division*** (***diaresis***) technique (Kaldis, 2008; Gill, 2010), applied to modern technologies, but also exhibits some of the complications that Aristotle (Balme, 1987) notes in the application of the technique, such as the use of the same ***differentia*** on different branches, and exemplars that cut across the categorization.

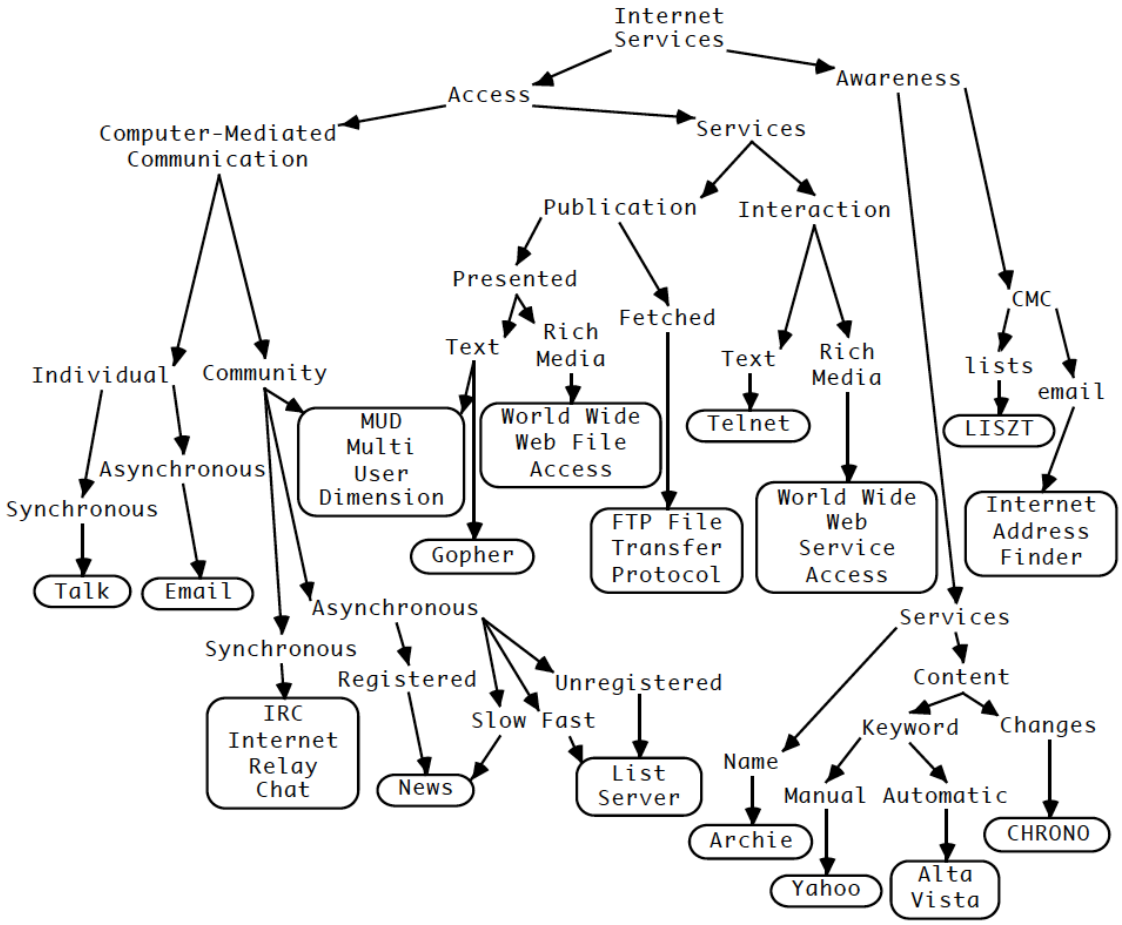


Figure 94: An analysis of relations between Internet services (Gaines et al., 1997)

Figure 95 shows the Internet services represented as a grid using the midpoint of the scale to represent inapplicable, and then clustered in Focus. Again it can be seen that the Focus algorithm has reconstructed the hierarchy, this time developed for an exposition of the services based on the collection and division method. That is, grid elicitation represents an alternative method to Plato's ***diaresis*** that develops the same structural model of the domain.

The construct matches in Figure 95 illustrate another feature of the Focus analysis. High matches are of obvious interest because they indicate similar constructs, but the low matches of other other constructs are also significant because they indicate major dimensions of construing that structure the domain. For example, the lowest matched constructs in Figure 95 are ***access—awareness*** and

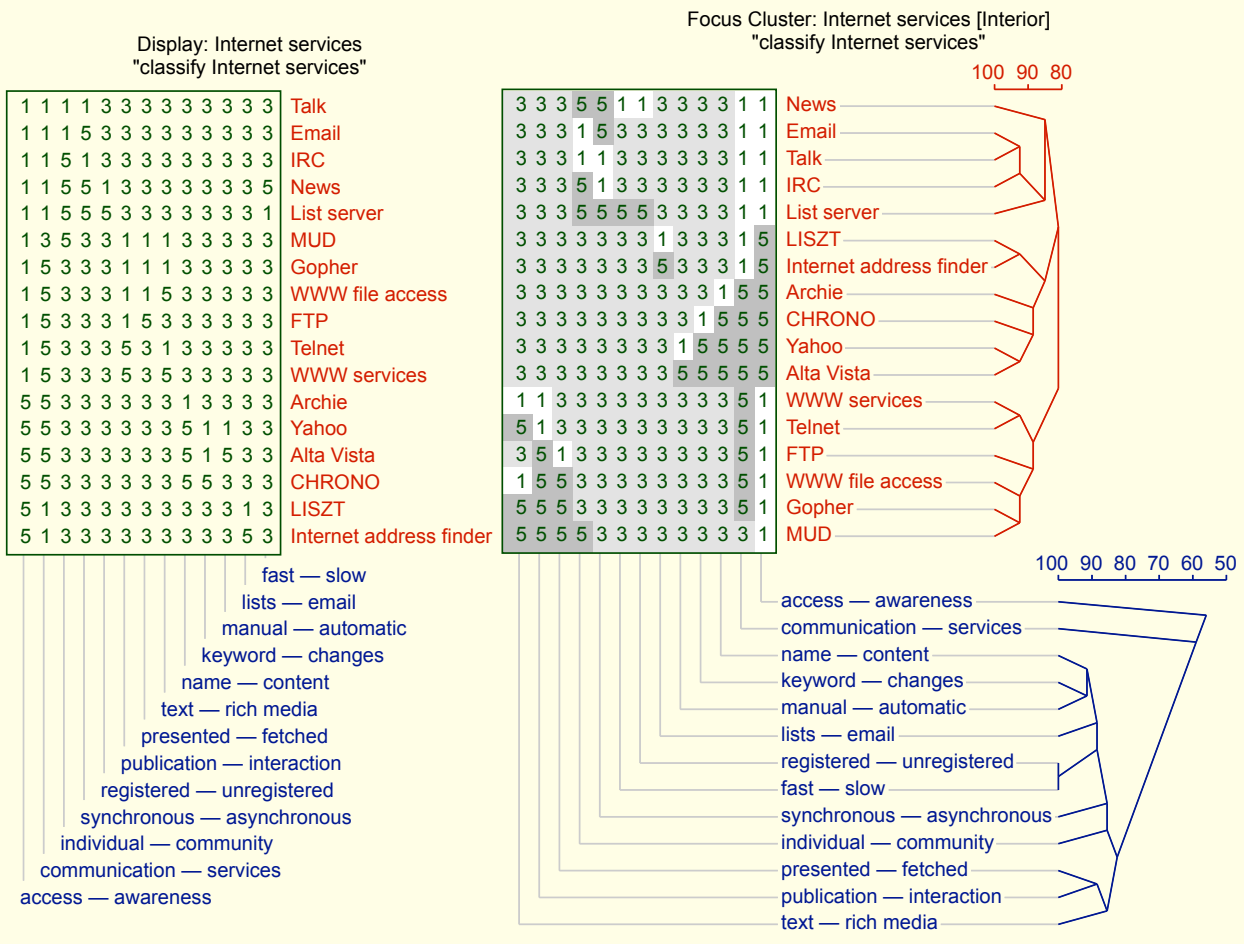


Figure 95: Internet services represented in a grid, *Display* and *Focus Cluster*

communication—services that are the top two distinctions made in the preliminary division of the domain in Figure 94. In a principal components analysis, as discussed in the following section, these constructs will appear as those dominating the structure of the domain and having the highest loading on the first two components. Thus, it is important to study the entire Focus cluster, not just the part representing the high matches.

5.4 PrinGrid Map: Spatial rotation and scree plot

Kelly (1955, ch.6) presents constructs as defining the dimensions of a ***psychological space*** (Kelly, 1969; Shaw and Gaines, 1992) in which the elements are located. The PrinGrid analysis is consistent with this and treats a grid as defining a geometric configuration in which the constructs form the axes of an n-dimensional space and the elements are represented by points located in that space determined by their ratings on the constructs. It rotates the configuration to lower its dimensionality as much as possible so that it may be plotted with principal components as axes in 2 or 3 dimensions.

Slater (1976; 1977) first applied this method of ***principal components analysis*** to conceptual grid data using a non-statistical, distance-based, geometric model and algorithms developed by Gower (1966) and his presentation of the results as ***biplots*** (Gower and Hand, 1995; Gower et al., 2011).

Clicking on button icon on the right of the ***PrinGrid*** button brings up a dialog which allows a principal component analysis of the grid to be displayed (Figure 96).

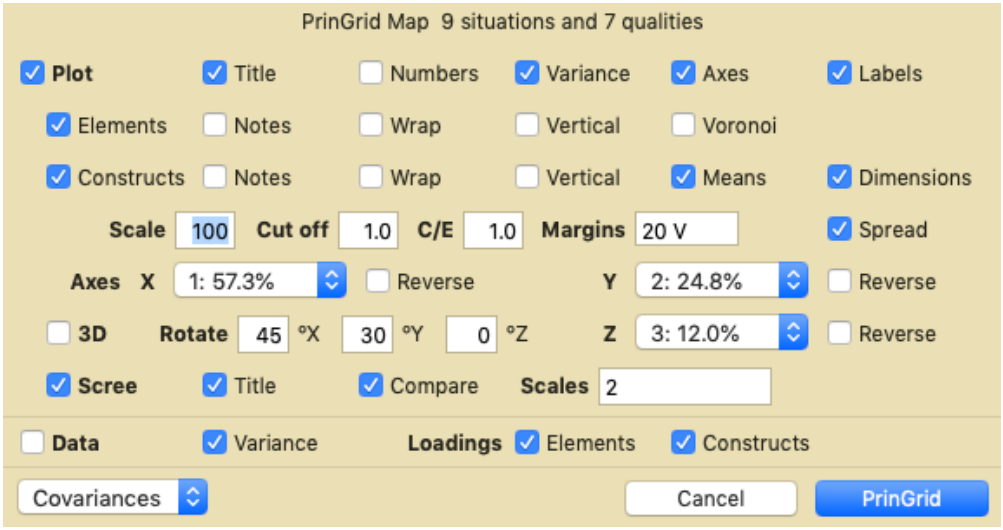


Figure 96: RepGrid *PrinGrid Map* dialog

The check boxes in the first row determine whether: a graphic plot is produced; it is titled; the elements and constructs are numbered; the percentage variance accounted for by each component is shown at the bottom of the plot; the component axes are shown; the axes, if shown, are labelled by component number and percentage variance.

The check boxes in the second row determine whether: the constructs and the notes attached to them are shown; the construct names are allowed to wrap if there are multiple words; they are placed vertically below or above their location in the plot, or horizontally to its right or left; a Voronoi diagram is plotted to indicate the points closest to the element location.

The check boxes in the third row determine whether: the elements and the notes attached to them are shown; the element names are allowed to wrap if there are multiple words; they are placed vertically below or above their location in the plot, or horizontally to its right or left; the constructs are plotted with the mean at the origin (so that asymmetric distributions of elements on constructs are visible); the construct dimensions are shown as a line between the poles.

Note that the ***Means*** check box determines whether the construct pole positions are symmetrical about the origin or whether they are placed such that the mean of the element values on the construct is at the origin. This option is useful to allow the plot to convey information about the asymmetrical use of a construct. It maximizes the information conveyed in that the angles between dimensions convey the correlations between them, the length of each conveys the combined loadings on the components plotted, and the position of the centre conveys the mean element value.

The check boxes in the fourth row determine: the overall scale of the plot (a useful option for larger grids to make more space for the labels and prevent then being spread too far from their orig-

inal positions); the ***cut off*** variance percentage below which components will not be shown; the ratio between the construct and element plot position multipliers (adjusting the length of the construct axes); the margins, if any, that will be added to the rectangle around the plotted points (specified as two numbers with a space between for horizontal or vertical components, e.g. 5 15, or one specifying both, followed by an optional "V" if the margins only apply when a Voronoi diagram is plotted); whether the element and construct pole names are spread out automatically to avoid overlap (one can also drag them to different positions in the plot itself).

Note that the C/E adjustment is also a useful option for grids with large numbers of constructs as the geometry of the analysis tends to project the element outside the hypercube formed by the constructs. Since only the relative direction of the construct dimensions is meaningful, not the absolute position of the poles, this scaling has no effect on the interpretation of the analysis.

The ***X*** and ***Y*** axis menus in the fourth row show the principal components with the percentage variance for which each accounts, and are used to select which components, if any, are plotted on the horizontal and vertical axes. The relative placement of the four quadrants is arbitrary, and the ***Reverse*** check boxes allow the plot to be reversed horizontally and vertically for greater perspicuity. These are useful options if multiple PrinGrid analyses are to be compared as they enable one to show the elements in similar quadrants in successive plots to the extent that this is possible.

The Z axis check box in the fifth row specifies that a three-dimensional plot should be produced, with the menu and ***Reverse*** checkbox specifying what component should be plotted on the Z axis and whether it should be reversed. The three values on the right specify the rotations in degrees of the X, Y and Z axes, respectively. The rotations are computed in the order Y, X, Z to keep the Y plots vertical if only X and Y rotations are used.

The check boxes in the sixth row determine whether: a scree plot should be produced; is titled; Frontier's (1976) comparative plot of the distribution if the principal components were randomly generated is also plotted to provide an estimate of the number of significant components.

The text field on the right enables the geometry of the scree plot to be adjusted. It can contain up to 4 numbers separated by commas: whether the vertical numeric increments are 1 or 2; vertical separation between scale points; horizontal separation between scale points; horizontal space at each end of plot. If fewer than 4 values are specified the remainder are filled from the appropriate position in the default values *2, 3, 20, 10*. In practice there is rarely a need to change the detailed geometry of the scree plot but the options are available to fine-tune the output for presentation or publication. Mousing over either the text field brings up help text giving information about available parameters.

The row of check boxes near the bottom determine whether a textual analysis is produced, and, if so, whether the percentage variances for the components are output, and whether the element and construct loadings on the components are output.

The menu at the bottom left specifies the metric to be used in calculating the data matrix for principal components analysis, either construct covariances or element distances (Minkowski—any power).

5.4.1 *PrinGrid Map plot output*

Figure 97 shows the plot produced for the grid of Figure 79 when one clicks the ***PrinGrid*** button with the settings above.

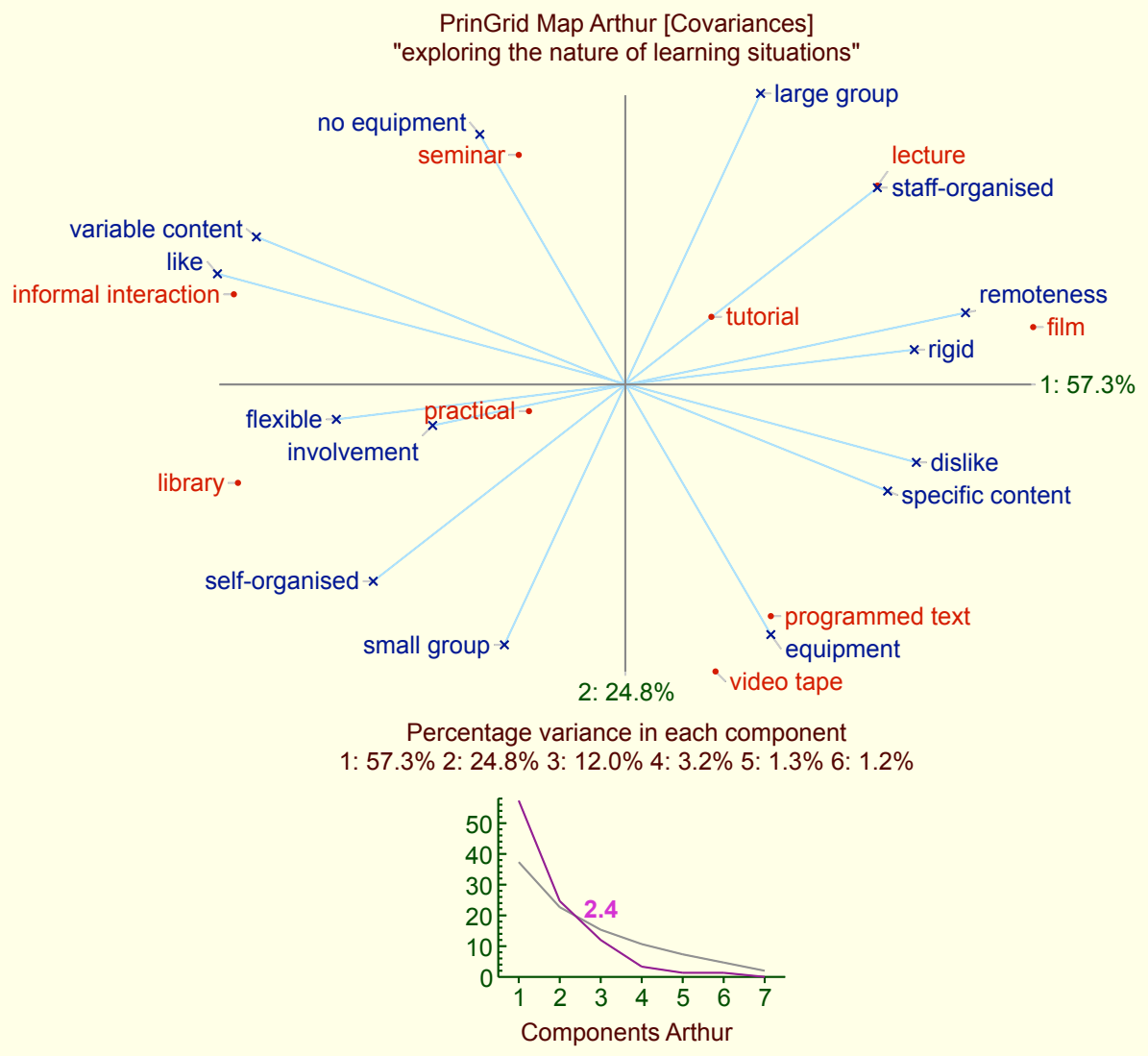


Figure 97: *PrinGrid Map* analysis

The plot opens in RepNet, and may be edited, annotated, and saved as a net and in graphic interchange formats. In particular, while the ***spread*** algorithm makes a reasonable attempt to place the element and pole names in a way that avoids overlap, the human eye may see more perspicuous locations. There are connecting lines between the labels and the points they label so that the linkage is apparent regardless of the label location.

The ***scree*** graph at the bottom plots the percentage variance accounted for by each component. The superimposed graph plots Frontier's (1976) estimated distribution if the principal components

were randomly generated which has been found to provide a good estimate of the number of significant components (Jackson, 1993). The number indicates this in terms of where the plots cross—when there is fractional value of significant components indicated the recommendation in the literature is to round down. However, if the estimate is used as a complexity indication the continuous scale of the fractional values may be useful.

Note that the comparative plot may cross the scree plot more than once indicating that there is more than one possible estimate of the number of significant components. There are continuing studies of such estimation techniques (Peres-Neto et al., 2005)—the literature is growing and the articles noted here provide useful search terms for such later literature as they are generally cited.

5.4.2 PrinGrid Map with Voronoi diagram

The interpretation of the *PrinGrid Map* plot generally involves noting: construct dimensions that intersect at a small angle and hence are similar in the context of the set of elements that have been construed (Kelly, 1969, p.105); elements that cluster together and hence are similar in the context of the constructs used to construe them; and the positioning of those clusters on the construct dimensions to gain understanding of the basis of the similarity.

This information is apparent visually in the conventional principal components analysis ***biplot*** (Gower et al., 2011) but there is no graphical plot supporting the visualization of element clusters as there is for *Focus*. Cluster analysis has long been a major research topic and there are a large number of approaches and algorithms to support it (Hennig et al., 2016), but most automated techniques are not simply explicable to those reflecting on their conceptual structures through grid analysis.

However, one readily-explained technique that supports visual understanding of a principal components biplot is to superimpose a ***Voronoi diagram*** (Okabe et al., 1992) that partitions the space to show locations that are nearer to one element location than to any other element location. This is a simple notion that dates back to Descartes and Dirichlet, and is easy to explain. However, computer algorithms to generate the diagram for any set of points proved difficult to design and prone to round-off errors for a variety of ill-conditioned configurations, and automatic generation of the diagrams was not implemented in the early grid analysis programs. Later research has resulted in a number of sound algorithms (Fortune, 1987) and computer implementations (Okabe et al., 1992).

Figure 98 shows the Voronoi partitions that PrinGrid superimposes on the plot when the ***Voronoi*** option is selected at the top right of the dialog (Figure 96). The options to wrap and centre the labels vertically above or below the locations were also selected to create a more compact appearance making it easier to place the element labels within their associated partitions. The Focus element tree from Figure 88 is shown at the right for comparison.

The way in which the elements cluster is now more readily perceived through the adjacency of their Voronoi partitions. For example, the highly related pairs on the left of the Focus tree are in adjacent partitions, and the linear sort of the tree appears as a two-dimensional path through adjacent Voronoi partitions. The relation between the partitions and the construct poles that characterize them is also apparent.

Gärdenfors (2000) has developed a theory of ***conceptual spaces*** that parallels Kelly's of ***psychological space*** in its psychological, philosophical and empirical foundations and techniques, and has

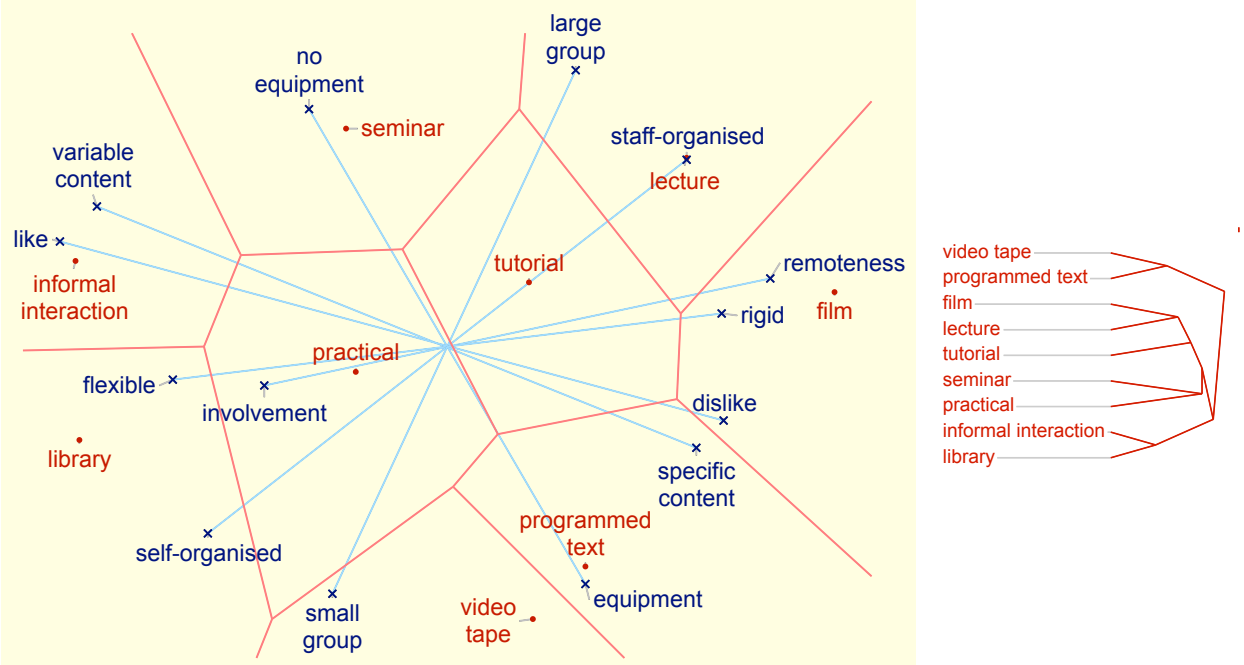


Figure 98: *PrinGrid Map* with Voronoi diagram (left) compared with Focus cluster (right)

argued that the Voronoi partition associated with an entity models its role as a conceptual ***prototype*** in Rosch's (1978; 1978) theory of categorization. There is significant ongoing research on conceptual spaces and their applications (Zenker and Gärdenfors, 2015; Kaipainen et al., 2019), much of which is relevant to personal construct psychology studies, even though the two areas have, so far, had little interaction.

5.4.3 PrinGrid Map with alternative metrics

The analysis of conceptual grids is largely based on distance measures calculated from the differences in ratings, but the major algorithms in common use are based on different measures, for example: construct covariances or correlations in Slater's (1976; 1977) ***INGRID***; sum of absolute differences in Shaw's (1980) ***FOCUS***; and chi-squared distances deriving from ***correspondence analysis*** (Greenacre, 2007, pp.177-181) in Feixas' ***RECORD/GRIDCOR*** (Feixas and Cornejo, 1996). Gower (1966) shows that distances matrices based on any of these, or other metrics complying the axioms for distance measures, may be be embedded in Euclidean space and analyzed using principal components analysis.

There are interesting relationships between these measures: covariances are equivalent to a Minkowski distance of power 2.0; sum of absolute differences is equivalent to a Minkowski distance of power 1.0; the two may be seen as related by regarding the power of 2.0 as weighting a difference by itself making larger differences more significant; correlations may be regarded as covariances normalized by standard deviations making the spread of usage of a rating scale less relevant; and so on.

The different distance measures underlying the mainstream techniques for developing conceptual models from grid data may raise questions about why one measure is used rather than another,

what difference does it make, which is best, and so on. Part of the answer is in the design objectives of the developers: Slater and Shaw were concerned to develop presentations of the grid data that would be understandable to those from whom the grids were elicited and could be simply explained to them. Slater took Kelly's notion of elements in the space defined by constructs and rotated that space to create a map of the elements in the plan that captured as much as possible of the spatial relationships between them. Shaw sorted the grid to bring similar items together, and presented this as hierarchical trees around the grid itself. From Slater's and his clients' perspective the Euclidean metric was natural to the space being rotated. From Shaw's and her clients' perspective the simple sum of absolute differences was easy to understand in the sorted grid that is part of the Focus presentation.

In RepGrid alternative metrics are made available in the *Focus Cluster* (and associated analyses) and *PrinGrid Map*, with the default for Focus being Minkowski distances power 1.0, and that for PrinGrid being covariances (equivalent to Minkowski distances power 2.0 (Gower, 1966)). However, both analyses may be run for any Minkowski power, not because any nonstandard value is recommended, but so that researchers may experiment with different metrics if they wish, for example, to run a sensitivity analysis to find what features of a conceptual model, if any, are subject to significant variation as the metric is changed.

For example, Figure 99 shows *Pringrid Map* analyses for Minkowski powers of 1.0, 2.0, 0.5 and 4.0. The first two are commonly used values and it can be seen that the component variances for 2.0 are the same as for covariances as in Figure 97 (as they must necessarily be). The unusual values of 0.5 and 4,0 show what happens as the power moves towards 0.0 where the elements will be equidistant, and towards infinity where the distance will be the maximum absolute difference over all constructs.

What is apparent, for this particular grid, is that the topology of the elements and their relations to the constructs changes little with this wide variation of the metric—the interpretation of the conceptual map is insensitive to the choices of metric shown. The same construct dimensions intersect at a small angle; the connectivity of the element Voronoi partitions is the same; and the positioning of those partitions on the construct dimensions is much the same.

It is possible to construct artificial grids where more substantial changes occur, and there are changes in the metric, for example those resulting from weighting constructs that can result in meaningful changes (§5.9). However, we have found for natural grids elicited to investigate anticipatory relations in a significant domain well-known to the elicitee, the analyses of Focus and Pringrid closely track one another, and that this seems to continue to apply to an wider spread of metrics.

There is scope for research on the impact of metrics on the analysis and interpretation of conceptual grids, and Rep Plus provides technical support for this. In theoretical terms, one might precisify the interpretations guidelines noted above and analyze the impact of varying the metric on the angle of intersection of the constructs, connectivity of the Voronoi partitions, and their relation to the construct dimensions, or similar interpretive guidelines. Empirically, one might analyze the impact of different metrics on the actually interpretation of a large corpus of grid data.

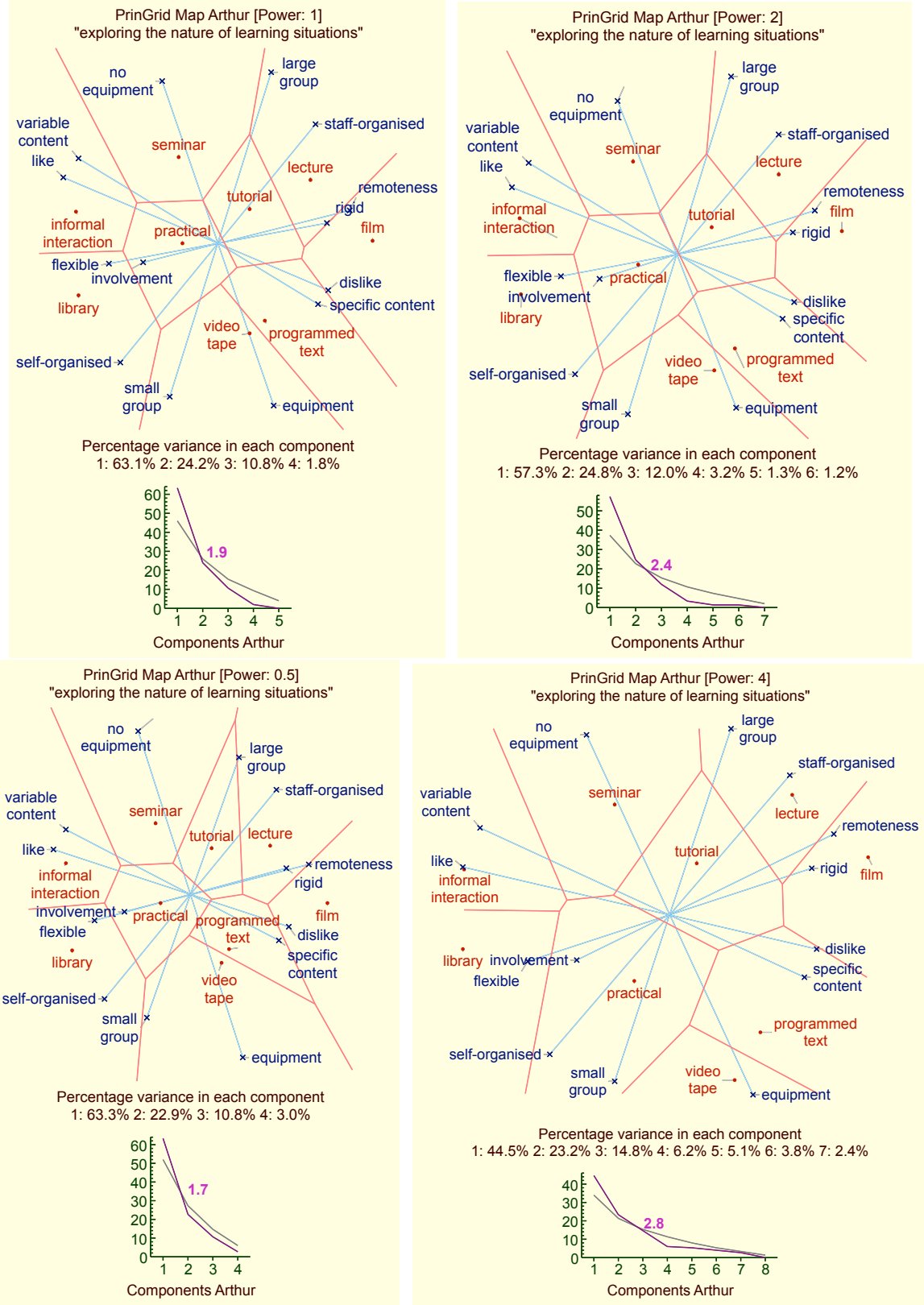


Figure 99: *PrinGrid Map* analyses with different Minkowski distance powers

Any such research needs to take into account both the ethos of personal construct psychology and the massive body of literature on the metrics of psychological space, and some of the interactions between them in Kelly's early studies preceding his 1955 book and his later commentaries on the nature of psychological space. Good starting points are Kelly's (1938) paper that analyzes Spencer's (1862, p.216) definition of *evolution* as "change from an indefinite, incoherent homogeneity to a definite, coherent heterogeneity" in mathematical and psychological terms, linking it to his discussion with Cyril Burt of *selection effects* in correlations and to Thurstone's (1935) use of factor analysis to study the "vectors of mind".

Kelly's student, Emmons (1939), extended his analysis with a critical study of factor analysis in psychology, and Kelly links it to his later research in an address to the *Moscow Psychological Society* in 1961 where he emphasizes that constructs are independent dimensions of psychological space that are brought into relationships as a selection effect of the elements chosen to populate that space in a particular context (Kelly, 1969, p.105)—construct systems create a *coherent heterogeneity* in Spencer's *incoherent homogeneity* or James' (1890, p.488) *buzzing confusion* of experience. Similarly, he emphasizes that there are no intrinsic *distances* in psychological space (Kelly, 1969, p.105), but these may be introduced to measure *similarities* between elements as counts of their *incidences* and *voids* with construct poles (Kelly, 1955, p.) (which can be extended to rating scales as structures constituted by multiple constructs (Gaines and Shaw, 2012, §3.4)).

The most substantial literature on the appropriate distance measures for psychological data is in studies of ***psychophysics*** from the 1930s to our era. The road map commences with Attneave's (1950) survey of *dimensions of similarity*, and proceeds through Torgerson's (1958) *theory and methods of scaling*, Shepard's (1964) *metric structure of stimulus space*, Tversky's (1977) *features of similarity*, Nosofsky's (1985) *identification of separable-dimension stimuli*, to Algom and Fitousi's (2016) *half a century of research on Garner interference and the separability–integrality distinction*.

The psychophysics literature is useful in presenting the issues in the operationalization of the notion of *psychological distance* but is largely based on perceptual similarities of well-defined physical stimuli and it not clear to what extent any conclusions apply to the complex experiences represented as *elements* in personal construct psychology. That in itself is a major research topic.

There are also issues with the notion of similarity in the psychological literature that have interesting constructivist interpretations. James (1890, p.579) noted that an entity could be similar to a second entity in some respect and similar to a third in another respect, but the second and third entities may have no similarity—the constructs on which a psychological metric is based may tacitly change. Nosofsky (1985, p.427-430) explains finding a different metric for similarities than that which Shepard (1964) derived as Shepard's subjects using different perceptual information from that which he varied in designing the experiment—the experimenter-as-scientist and the subject-as-scientist may employ different construct systems.

5.4.4 *PrinGrid Map with mixed construct types*

The *PrinGrid Map* analysis may be applied to grids with any mixture of the rating scale types available in *Rep Plus*. For example, Figure 100 shows a plot produced for the grid of Figure 81. The multiple types of the constructs are not apparent, as they are in the Focus plot, because the actual ratings are not shown in a PrinGrid map as they are in a Focus clustering.

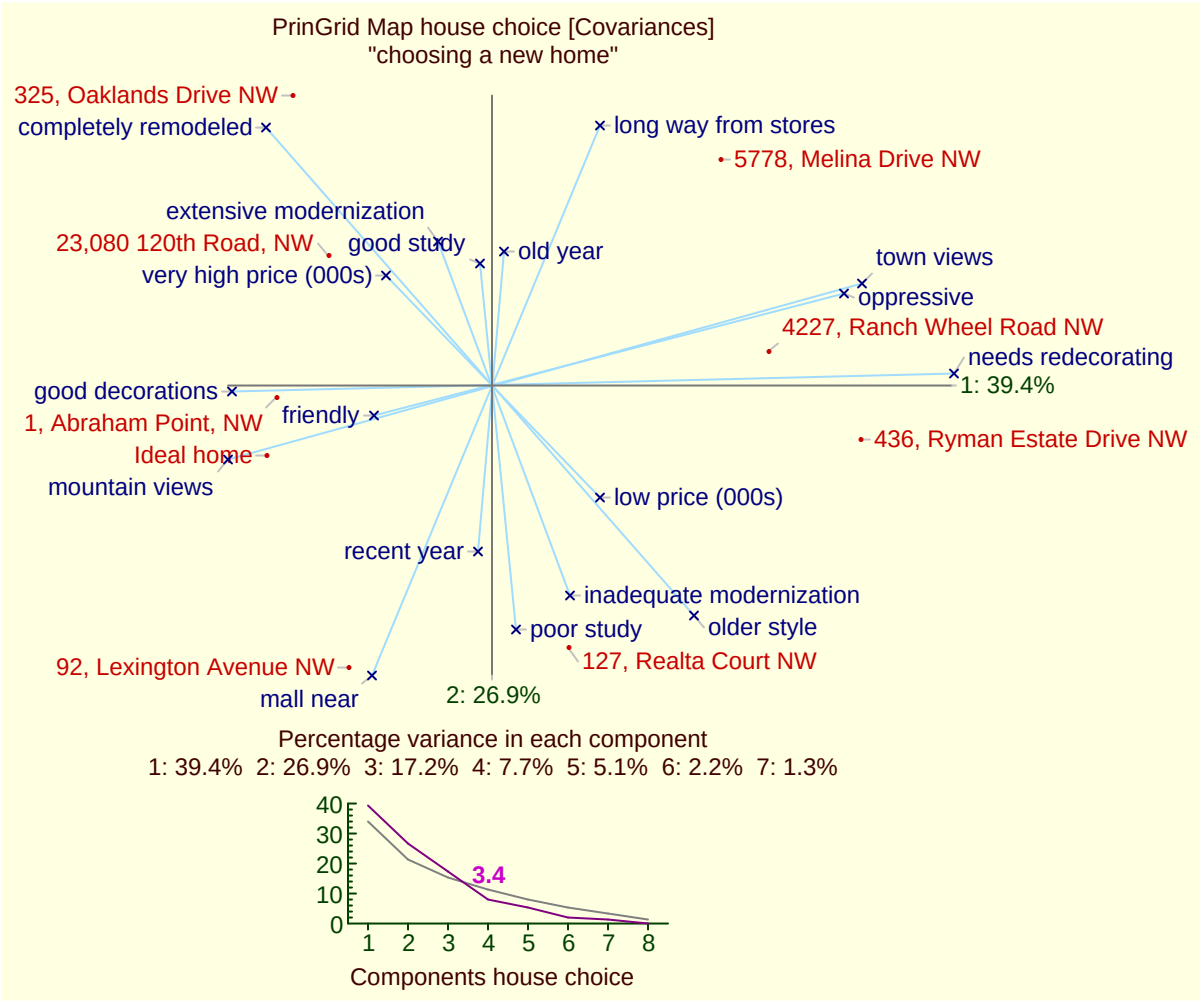


Figure 100: *PrinGrid Map* analysis of a mixed-type grid

5.4.5 *PrinGrid 3D plot*

It is sometimes of interest to show three components in what is essentially a two-dimensional section of a three-dimensional plot. Figure 101 shows the three-dimensional output generated when the check box to the left of the Z axis specification in Figure 96 is clicked. The X-Z plane is shown by the orange rectangle, and lines have been dropped from the element and construct pole positions to their coordinates in this plane.

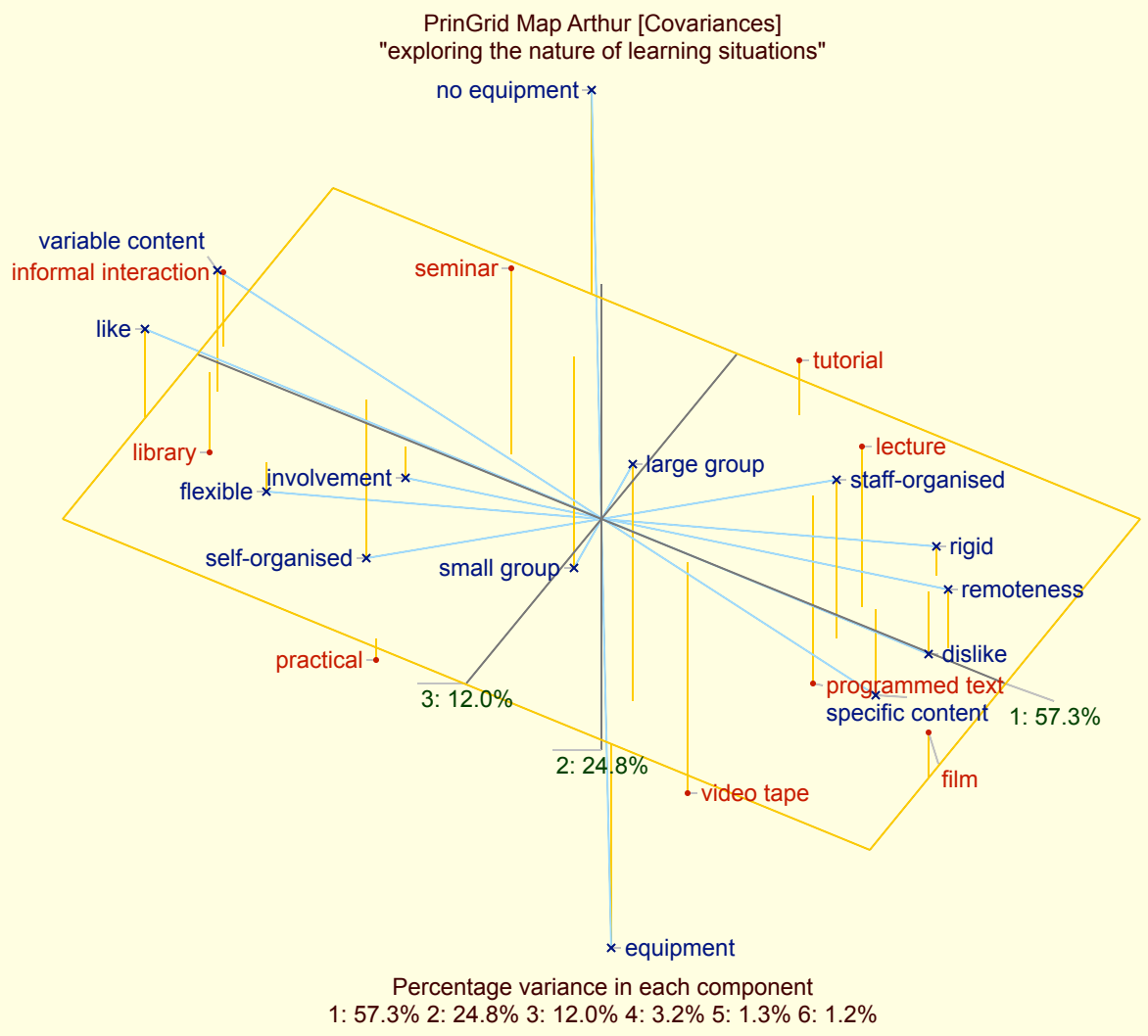


Figure 101: Three dimensional *PrinGrid Map* display

What component is shown on what is axis is arbitrary and may be varied for perspicuity, as may the angles of rotation of the three-dimensional plot before it is projected into two dimensions for display.

5.4.6 PrinGrid text output

Figure 102 shows the textual output of the data underlying the plot of Figures 97 produced when the ***Data*** checkbox at the bottom left of Figure 96 is checked, and below it the data underlying the plot of Figure 100. Note that the variance and Frontier estimate are shown for all the components, not just those above the specified cut-off, and also that, if comparing the data with other principal components analyses, the absolute values of the loadings produced are arbitrary and depend on how the grid data has been scaled—only the relative values are meaningful.

Percentage Variance in Each Component

1	2	3	4	5	6	7	
57.28	24.83	12.03	3.22	1.28	1.18	0.18	Variance %
37.04	22.76	15.61	10.85	7.28	4.42	2.04	Frontier estimate %
57.28	82.11	94.14	97.36	98.64	99.82	100.00	Cumulative variance %

Element Loadings on Each Component

	1	2	3	4	5	6	
1	1.537	1.216	-0.115	0.304	0.227	-0.056	lecture
2	0.525	0.410	-1.201	0.004	-0.108	0.297	tutorial
3	-0.655	1.401	-0.178	-0.206	-0.058	0.164	seminar
4	-0.587	-0.159	1.385	-0.374	0.218	0.280	practical
5	2.485	0.349	0.815	0.019	-0.331	-0.244	film
6	-2.359	-0.603	0.087	0.083	-0.426	-0.013	library
7	0.883	-1.418	-0.716	-0.685	0.121	-0.198	programmed text
8	0.556	-1.750	0.046	0.670	0.123	0.137	video tape
9	-2.385	0.554	-0.121	0.185	0.234	-0.368	informal interaction

Construct Loadings on Each Component

	1	2	3	4	5	6	
1	1.809	0.380	-0.068	0.650	-0.394	-0.230	involvement – remoteness
2	1.956	0.242	-0.578	-0.835	-0.185	-0.223	flexible – rigid
3	-0.991	1.700	-1.605	0.147	0.159	-0.010	equipment – no equipment
4	1.704	1.331	0.177	-0.035	-0.168	0.510	self-organised – staff-organised
5	0.867	1.866	1.159	-0.034	0.303	-0.228	small group – large group
6	2.141	-0.863	-0.182	-0.007	0.275	0.169	variable content – specific content
7	2.368	-0.640	-0.531	0.299	0.282	-0.081	like – dislike

Percentage Variance in Each Component

1	2	3	4	5	6	7	8	9	
40.86	25.04	18.63	7.71	5.08	1.83	0.77	0.08	0.00	Variance %
31.43	20.32	14.77	11.06	8.28	6.06	4.21	2.62	1.23	Frontier estimate %
40.86	65.90	84.53	92.24	97.33	99.16	99.92	100.00	100.00	Cumulative variance %

Element Loadings on Each Component

	1	2	3	4	5	6	
1	-1.159	1.794	-0.122	-0.105	-0.275	-0.259	325, Oaklands Drive NW
2	1.907	0.252	0.194	-0.161	-1.116	0.187	4227, Ranch Wheel Road NW
3	1.230	0.935	-1.208	-0.990	0.574	0.200	5778, Melina Drive NW
4	0.642	-1.033	1.591	-0.650	0.141	-0.506	127, Realta Court NW
5	-1.046	-1.740	0.391	-0.232	0.056	0.543	92, Lexington Avenue NW
6	2.272	-0.624	-0.622	1.142	0.422	-0.135	436, Ryman Estate Drive NW
7	-1.543	-0.558	-0.836	-0.177	0.247	-0.193	1, Abraham Point, NW
8	-0.667	1.489	1.539	0.647	0.427	0.272	23,080 120th Road, NW
9	-1.635	-0.515	-0.926	0.526	-0.476	-0.110	Ideal home

Construct Loadings on Each Component

	1	2	3	4	5	6	
1	1.018	2.246	0.714	-0.691	-0.280	0.249	mall near – long way from stores
2	2.221	0.476	-0.367	-0.169	0.351	-0.097	mountain views – town views
3	1.521	0.000	-1.158	0.335	0.834	0.029	friendly – oppressive
4	0.082	-1.234	0.624	-1.542	0.412	-0.105	good study – poor study
5	1.454	-1.699	0.330	0.137	-0.714	-0.217	completely remodeled – older style
6	-2.510	0.111	0.370	0.092	0.253	0.111	needs redecorating – good decorations
7	0.661	-0.612	2.085	0.586	0.676	0.222	extensive modernization – inadequate modernization
8	-0.638	1.035	0.508	0.100	0.321	-0.792	low price (000s) – very high price (000s)
9	-1.054	-0.542	-1.132	-0.381	0.347	0.126	old year – recent year

Figure 102: Data from the *PrinGrid Map* principal components analyses of the grids

5.4.7 *Pringrid analysis of hierarchical data*

It is interesting to compare Focus and PrinGrid analyses of the hierarchical data structures analyzed in §5.3.3 because, in both cases, the number of independent constructs used is known in advance. Figure 103 shows PrinGrid plots in 2 and 3D for the simple artificial hierarchy with seven constructs of Figure 92. The second and third components are equal, and can be seen on the left that a plot of the first two components does not adequately discriminate between the elements, but on the right that a plot of all 3 components clusters all the elements appropriately but not as clearly as in the Focus plot.

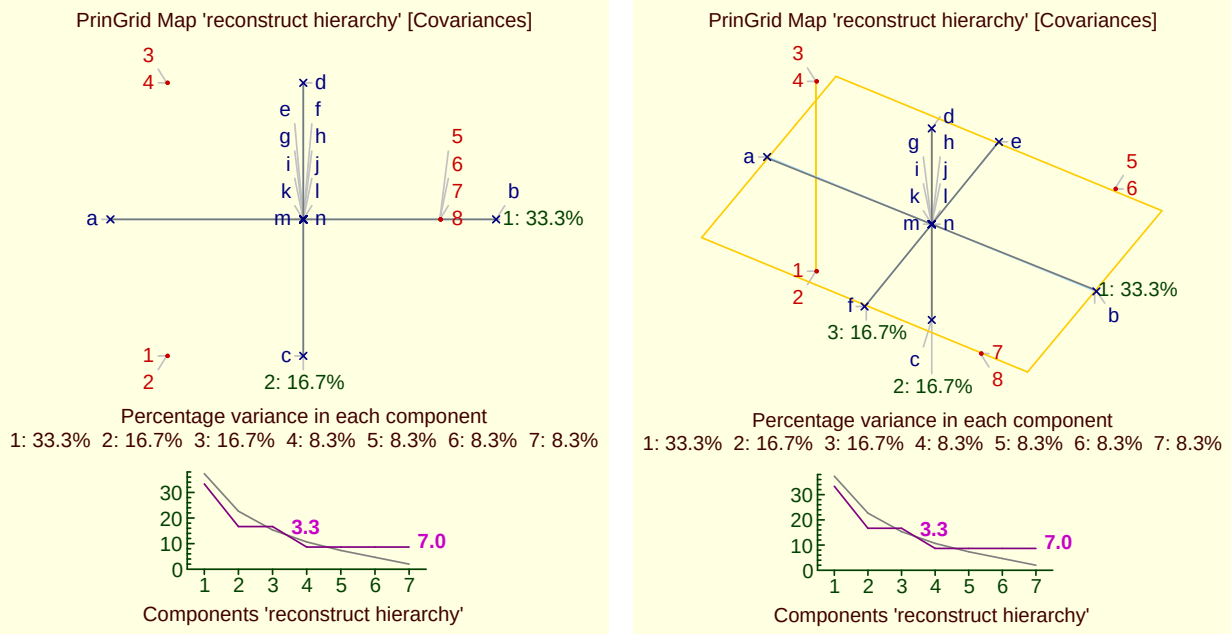


Figure 103: *PrinGrid* analysis of an artificial hierarchy in 2 and 3 dimensions

Frontier's (1976) comparative plot of the distribution two intersections indication that 3 components may reasonably account for the data but 7 are better. In the 3D plot, the correspondence of the axes to the ***a—b***, ***c—d*** and ***e—f*** constructs shows that the plot accurately represents the higher levels of the hierarchy, but the lack of representation of the four lowest level constructs indicates that a 3D plot cannot capture all of them, as would be expected with the artificial 7-dimensional data.

Figure 104 shows a comparison of PrinGrid and Focus analysis of the natural hierarchy of Internet services in the 1990s analyzed in §5.3.3. The Frontier plot indicates that 2D and 12D analyses are significant, picking up the 2 major constructs differentiating the Internet services, ***access—awareness*** and ***computer-mediation communication—services***, as well as the 12 constructs fully differentiating them. The element clusters of the Voronoi diagram correspond to those in the Focus element tree. Both forms of presentation make apparent the relations between the elements but through different visualizations—they complement one another.

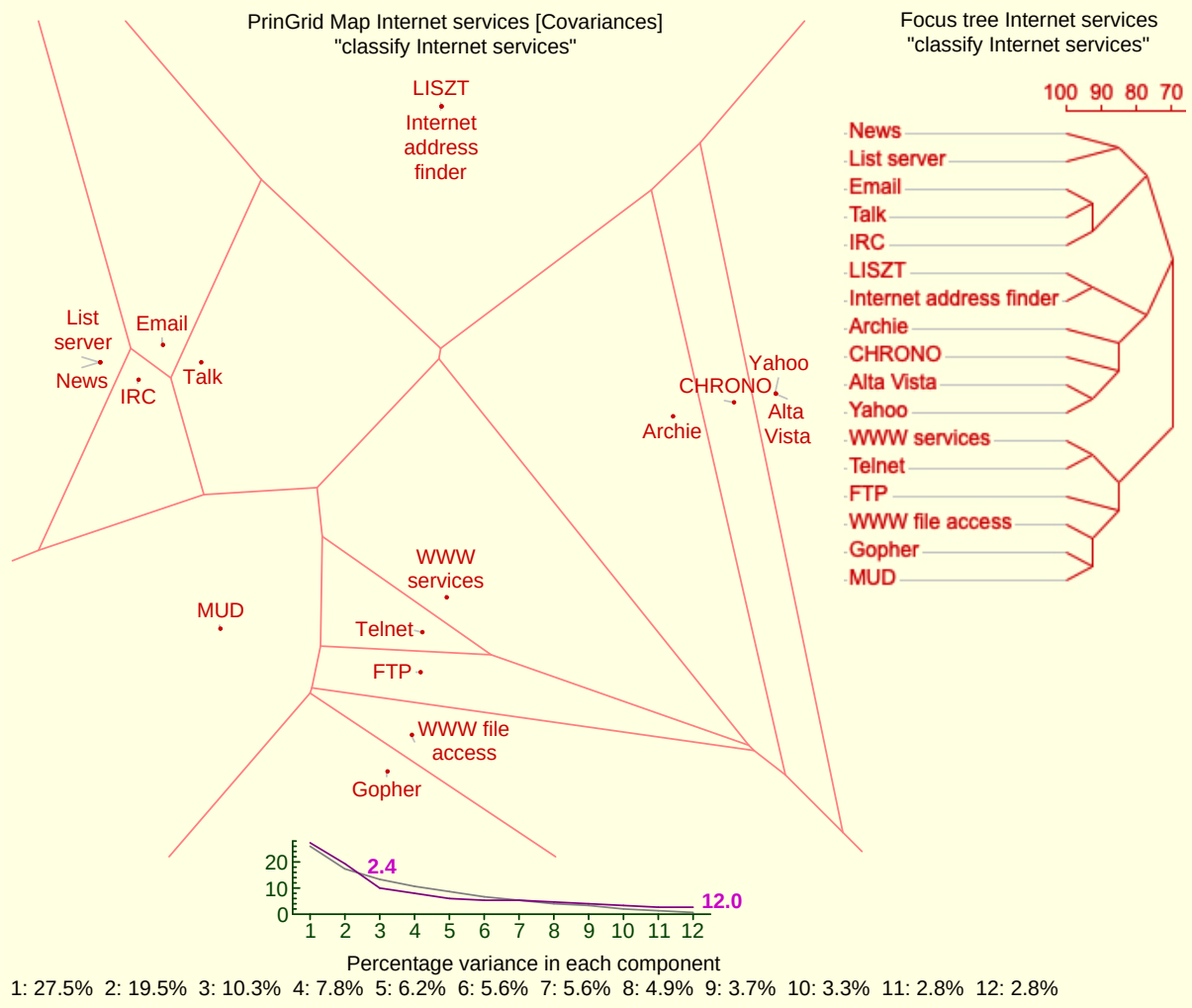


Figure 104: *PrinGrid* analysis of a natural hierarchy compared with *Focus* analysis

5.5 Crossplot: Plotting elements on constructs as orthogonal axes

Quadrant diagrams that show two orthogonal constructs as axes together with a set of elements plotted in the four quadrants thus created are commonly used in many literatures. The ***Crossplot*** tool in RepGrid allows two or three constructs to be selected as axes and the elements to be plotted in 2D or 3D respectively according to their ratings on the selected constructs,

Clicking on button icon on the right of the ***Crossplot*** button brings up a dialog which allows the constructs for a crossplot to be selected (Figure 105). The options are similar to those for PrinGrid except that the axes are constructs not components.

The row of check boxes at the top determine whether: the plot is titled; whether the elements and constructs are numbered; whether the element labels are spread to prevent overlap; what margins will be added for a Voronoi diagram; the scale of the plot.

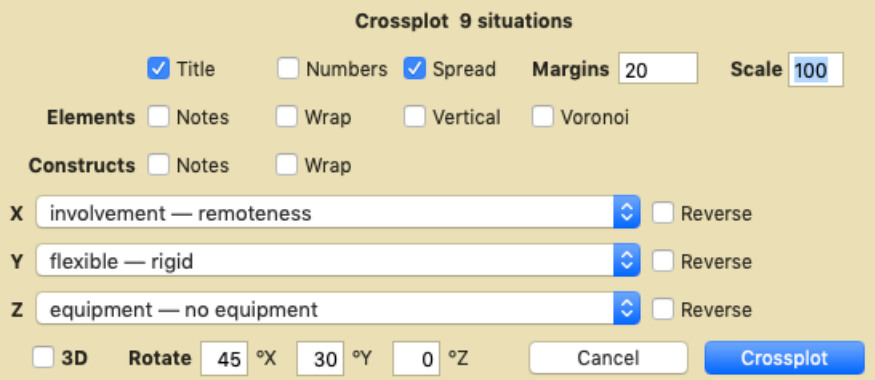


Figure 105: RepGrid *Crossplot* dialog

The row of check boxes on the second row determine whether: element notes should be shown; element labels should wrap; labels should be vertically above or below their location; whether a Voronoi diagram for the element locations should be plot.

The row of check boxes on the third row determine whether: construct notes should be shown; construct labels should wrap.

The fourth row allows the construct for the X axis to be selected, followed by a check box determining whether it is reversed.

The fifth row allows the construct for the Y axis to be selected, followed by a check box determining whether it is reversed.

The sixth row allows the construct for the Z axis to be selected, followed by a check box determining whether it is reversed.

On the bottom row, the *3D* check box specifies whether a three-dimensional plot will be produced and numbers in the three text boxes following determine the rotations of the 3D plot.

Figure 106 shows the plot produced when one clicks the ***Crossplot*** button with the settings above.

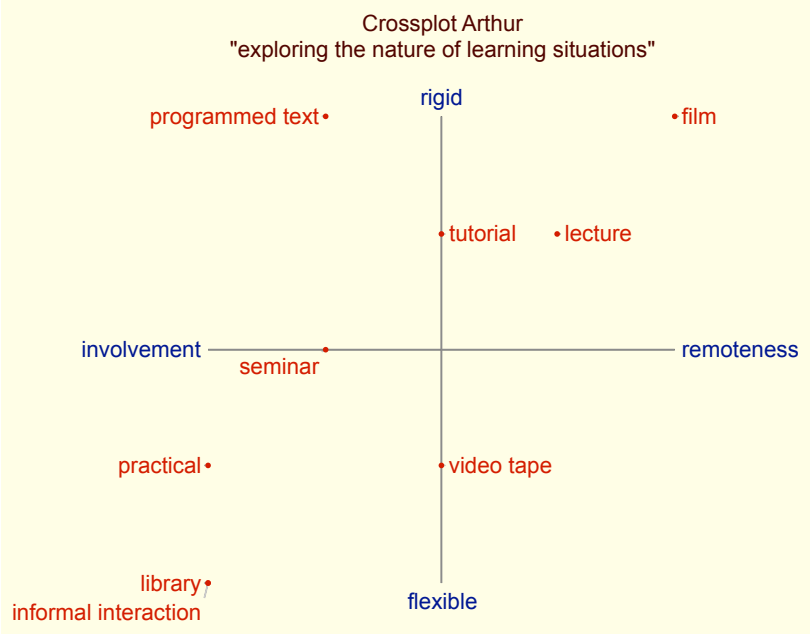


Figure 106: *Crossplot* of the elements on two selected constructs

Figure 107 shows the three-dimensional output generated when the *3D* check box is also set.

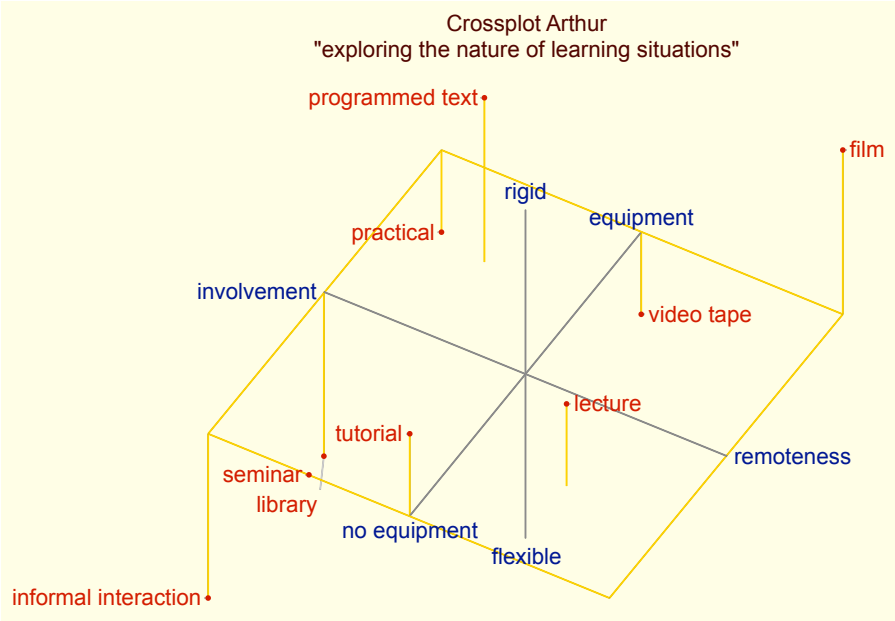


Figure 107: *Crossplot* of the elements on three selected constructs

Crossplots are useful visual presentations of grid data in their own right, and also useful in having users come to understand the PrinGrid plots as rotations of multi-dimensional crossplots.

5.6 Compare: Comparison of grids with some common elements and/or constructs

When two grids were elicited from the same person or from two members of the same community who are expected to use similar terminology to identify events and shared concepts, if they have have elements and/or constructs in common it is possible to be able to compare them for similarities and differences in the use of constructs and the construing of elements. The *Compare* tool in RepGrid provides a graphic comparison of such grids based on the *MINUS* algorithm of Shaw (1980) extended to grids having a diversity of rating scales including multiple types.

Note that determining common elements across grids is based on lexical equality of the element names, and common constructs on the lexical equality of the pole names and construct names (if any). Hence a grid being compared should not have two or more elements with the same name or two or more constructs that are equal on the above criterion. **The supposition that lexically equivalent elements and/or constructs are intended by the elicitee(s) to have *the same meaning* needs careful consideration and justification if the analysis is to be meaningful.** This can be addressed by discussions with the elicitees or a focus group representing them or managing the study.

5.6.1 *Methodology of grid comparison*

The methodological basis of grid comparison is illustrated in Figure 108 where two grids overlap in a central region of common elements/constructs and three pairs of peripheral regions having: common elements but different constructs; common constructs but different elements; and different elements and constructs.

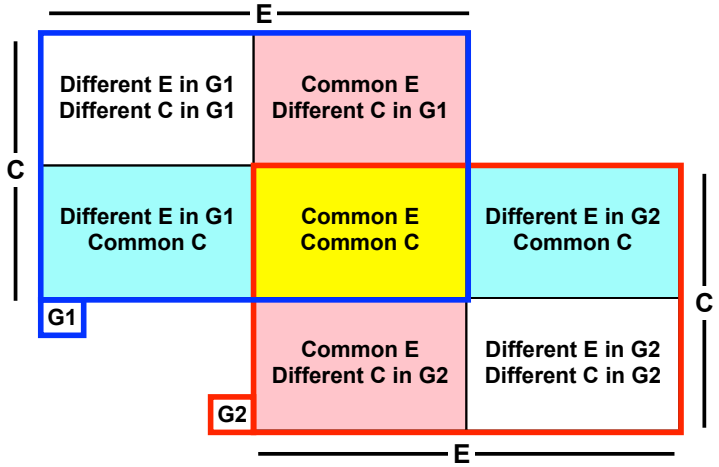


Figure 108: Pairwise comparison of grids with some common elements and/or constructs

The differences in ratings between the two grids in the common subgrid in the centre may be used to determine the degree of ***consensus*** and ***conflict*** (Shaw and Gaines, 1989) of the elicitees for each element and each construct represented in the subgrid. In essence, the common constructs are used to match the common elements, and the common elements are used to match the common constructs.

In the regions having some commonality this can be used to enable the items that are lexically different in one grid to be matched against all the items of that type in the other grids and the best

matches can be displayed. This is achieved by analyzing the subgrids comprising the central subgrid combined with each of the four peripheral grids to its left and right and above and below it. In essence: the common constructs in the central grid are used determine the best match for any element in G1 of an element in G2, and ***vice versa***; the common elements in the central grid are used determine the best match for any construct in G1 of an construct in G2, and ***vice versa***.

Figure 109 shows three common instances of the general schema shown in 108. On the left, grids having both elements and constructs in common arise in many different ways, as a temporal sequence from one individual, developing a grid at one time and then rerating after a learning experience, or through an exchange of grids where two or more people develop grids supplying their own elements and constructs and then exchange them with one another so that each person rates the other's elements on the other's constructs, and so on. In the centre, grids having common elements but different constructs arise when two or more individuals develop grids using a commonly agreed set of elements and wish to see the similarities and differences in their individual constructs. On the right, grids having common constructs but different elements arise when two or more individuals develop grids using a commonly understood set of public constructs and wish to see the similarities and differences in their individual experiences in terms of these constructs.

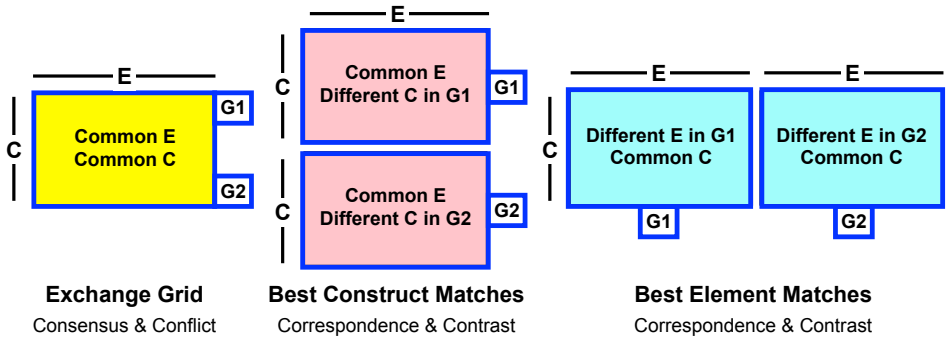


Figure 109: Some special cases of pairwise comparison of grids

The following subsections illustrate how the Compare tool analyses these three types of data, including grids instantiating the more situation shown in Figure 108 where all three forms of analysis may be possible. There are additional analyses possible for research designs where large numbers of grids have been elicited with the various forms of overlap discussed above, or even when there is no overlap but only a common domain of interest, and these are covered in the ***RepGrids*** manual.

5.6.2 The compare dialogue

Clicking on button icon on the right of the ***Compare*** button brings up a dialog which allows the current grid to be compared with a another selected one Figure 110). The ***Open Grid*** button at the bottom left is highlighted, and the ***Compare*** button on the left is disabled, because one needs to open a second grid for comparison before proceeding with the analysis. Clicking on the ***Open Grid*** button brings up a standard file open dialog where one can open a secondary grid for comparison. One may also drag a grid file to the dialogue to open it for comparison.

Note that the ***Select*** check boxes in the ***Element*** and ***Construct*** panes may be used to restrict the items in the primary grid that are used in the comparisons. This enables a subgrid to be used in a

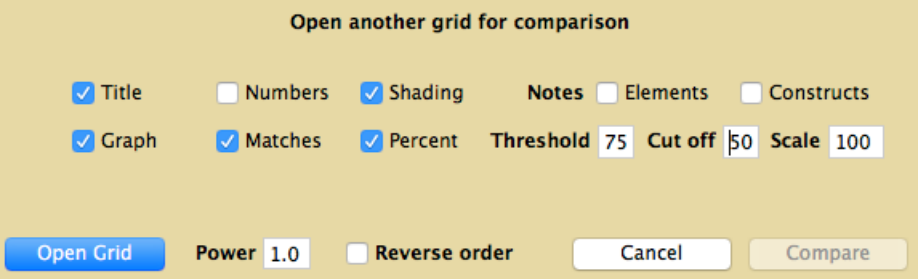
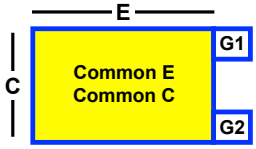


Figure 110: RepGrid initial *Compare* dialog

comparison and dynamically adjusted as a result of the comparison. This capability may be used, for example, to implement Shaw's (Shaw, 1980, p.77) ***CORE*** algorithm to identify core constructs in a set of grids. In particular, the WebGrid port of the Compare functionality provides an interactive user interface that makes this a very simple activity.

5.6.3 Comparing grids with substantial numbers of elements and constructs in common

Figure 111 shows the dialog when a grid having both elements and constructs in common is opened. The text at the top identifies the grids being compared and the number of elements and constructs they have in common.



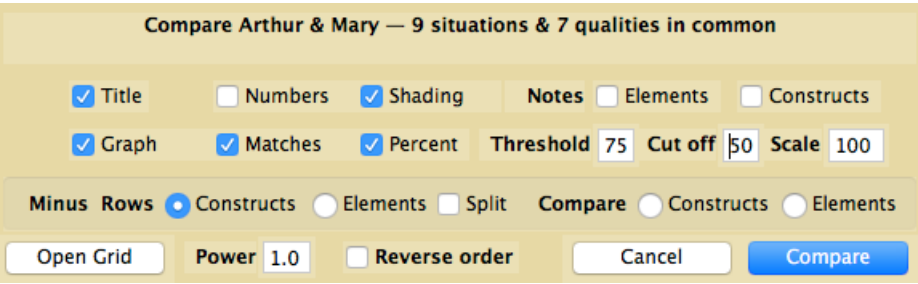


Figure 111: *Compare* dialog—grid with same elements and constructs opened for comparison

The first row of check boxes determine whether: the plot is titled; elements and constructs are numbered; high and low differences are shaded; notes attached to elements and constructs are shown.

The second row determines whether matches will be graphed; match values will be shown; the cumulative percentage of matches above or equal to the match value will be shown; and specifies the threshold, cut off, and scale. The ***Threshold*** value determines where in the plot matches will be shown as below threshold. The ***Cut off*** value specifies the lowest match that will be plotted. The ***Scale*** value determines how much space will be allocated to the graph of matches.

The panel below is only visible if the grids being compared have both common elements and common constructs. The four radio buttons select whether a ***Minus*** plot of the difference grid is required and, if so, whether the rows should be constructs or elements, or whether a ***Compare*** plot is required where the best matching constructs or elements are shown. The ***Split*** check box determines whether a ***Minus*** plot shows the difference in ratings or the actual ratings on separate lines (***Compare*** plots when only elements or only constructs are in common always use separate lines).

In the bottom row, the *Power* value determines the exponent used in the Minkowski metric used to compute matching scores as discussed in §5.3. The normal mode of comparison is that the primary grid is compared with the secondary grid, and the *Reverse order* check box reverses this.

Figure 112 shows the plot produced when one clicks the *Compare* button with the settings above.

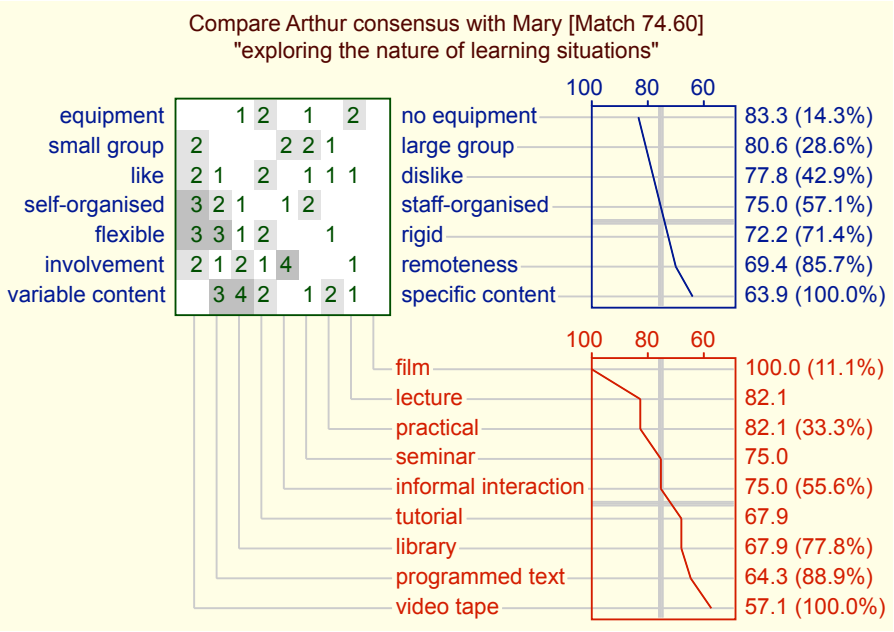


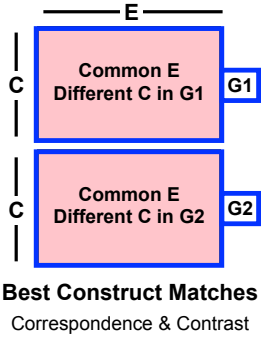
Figure 112: Comparing two grids with the same elements and constructs

On the left is shown the absolute difference between the ratings in the two grids with the constructs and elements sorted so that those that are most similar in the two grids are the top and on the left, respectively. The number on the right of the the title is the overall match between the two grids. If the power is set to 1.0 it corresponds to both the mean construct match and the mean element match.

The graphs on the right provide a plot of the individual construct and element matches between the grids being compared. The numbers on the right show the numeric match value and, in parentheses, the cumulative percentage of items matching at that value or greater.

5.6.4 *Comparing grids with a substantial number of elements in common*

Figure 113 shows the *Compare* dialog when a grid having the same elements but different constructs is opened for comparison. Figure 114 shows the plot produced when clicks on the *Compare* button with the settings above. Each of the constructs in the primary grid is shown with the best matching construct in the secondary grid on the line below it. Element differences have also been computed based on the two grids of matching constructs, and constructs and elements have been sorted so that those that are most similar in the two grids are the top and on the right, respectively.



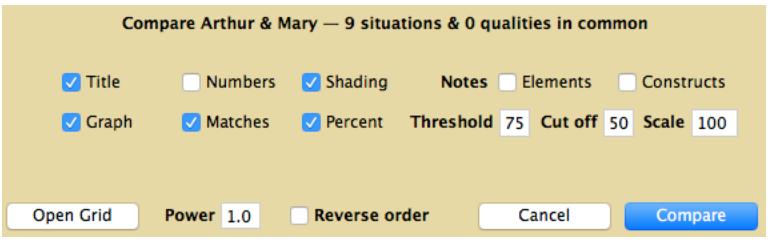


Figure 113: *Compare* dialog—grids with same elements but different constructs

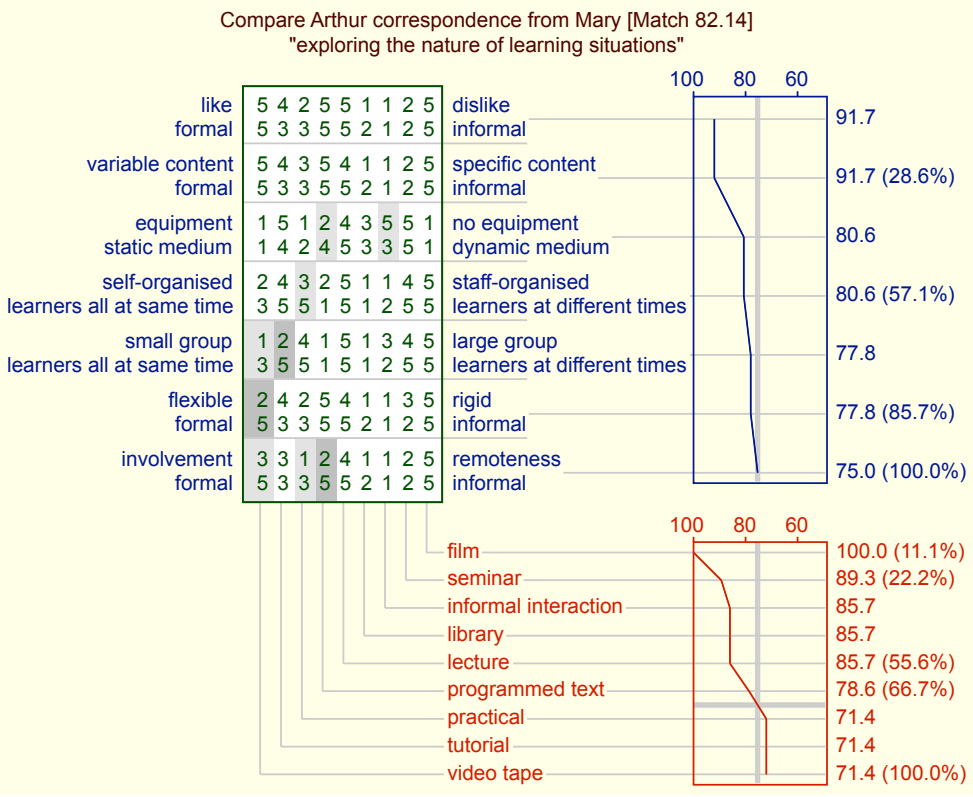


Figure 114: Comparing two grids with the same elements

The plot shows the best match in Mary's grid for each construct in Arthur's grid. If the ***Reverse order*** check box is set then a similar plot showing the best match in Arthur's grid for each construct in Mary's grid will be produced.

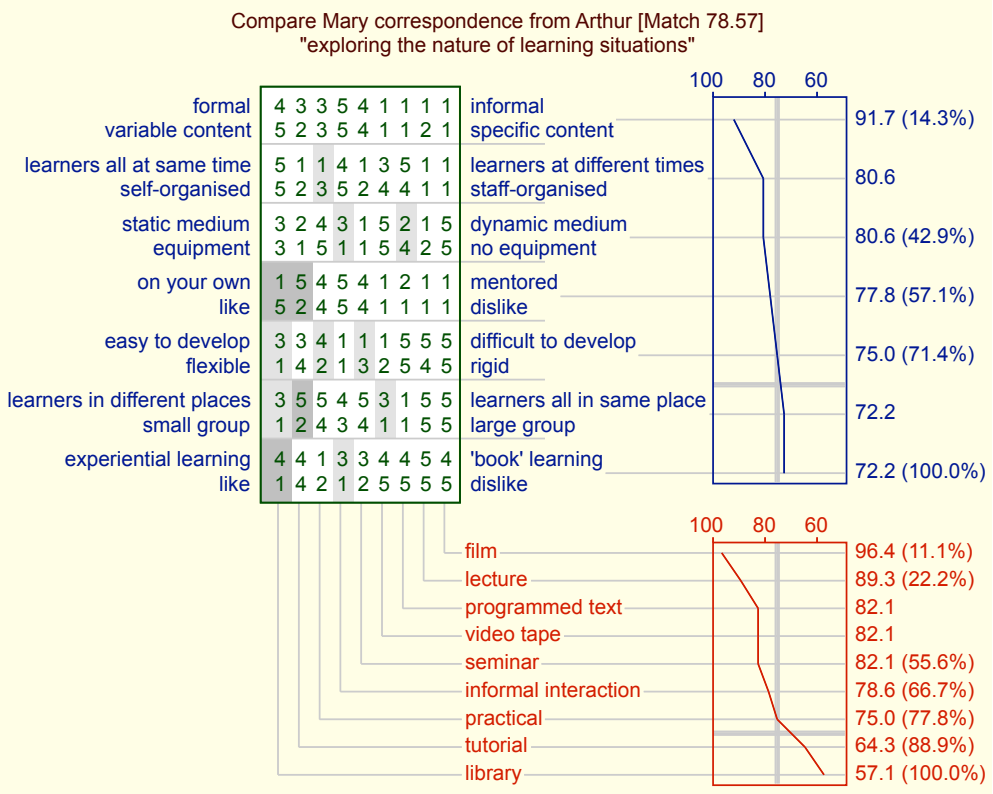


Figure 115: Comparing two grids with the same elements in reverse order

The reduced overall match value indicates that Mary's constructs enable her to better understand Arthur's than *vice versa*, and the effect of this is apparent in the construct and element match plots.

5.6.5 *Comparing grids with a substantial number of constructs in common*

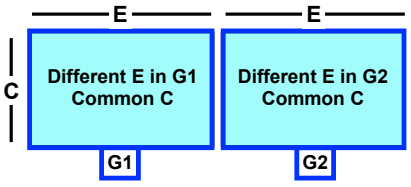


Figure 116 shows the ***Compare*** dialog when a grid having the same constructs but different elements is opened for comparison. Figure 117 shows the plot produced when one clicks on ***Compare***. Each of the elements in the primary grid is shown with the best matching element in the secondary grid on the line below it. Construct differences have also been computed based on the two grids of matching elements, and elements and constructs have been sorted so that those that are most similar in the two grids are the top and on the right, respectively.

The ***Reverse order*** check box may be used to determine the best matches for Paul's elements in Arthur's grid.

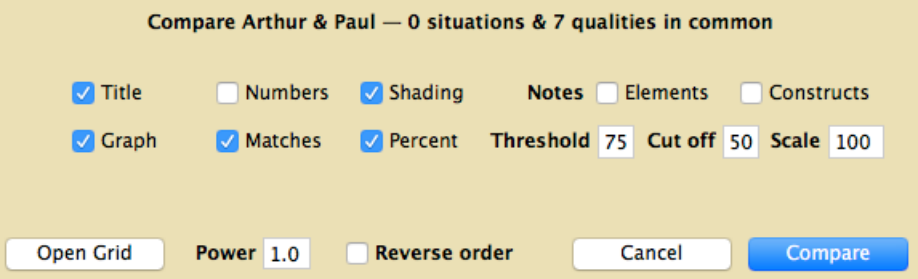


Figure 116: *Compare* dialog—grids with same constructs but different elements

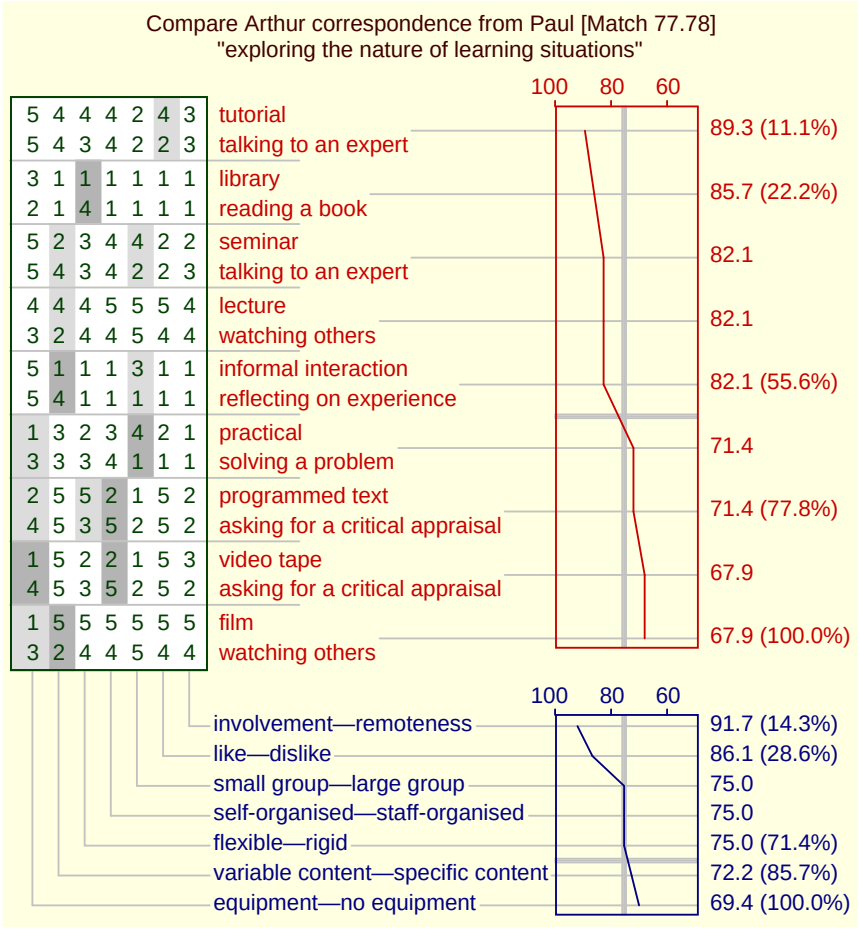


Figure 117: Comparing two grids with the same constructs

When a grid has a substantial number of elements and constructs the same, as discussed in §5.6.3, all three forms of match discussed above are possible. As shown in Figure 111, a panel appears in the *Compare* dialog box that allows the grid to be compared in terms of its common elements only (as in §5.6.4), its common constructs only (as in §5.6.5), or its common elements and common constructs (as the default in §5.6.3).

5.7 Match analysis: Display matches between elements and between constructs

It is often useful to be able to view the matches between selected elements or constructs in reverse order, for example, to see all the matches with an ***ideal element*** or a particularly significant construct. Clicking on button icon on the right of the ***Matches*** button brings up a dialog which allows matching elements and matching constructs to be displayed (Figure 118).

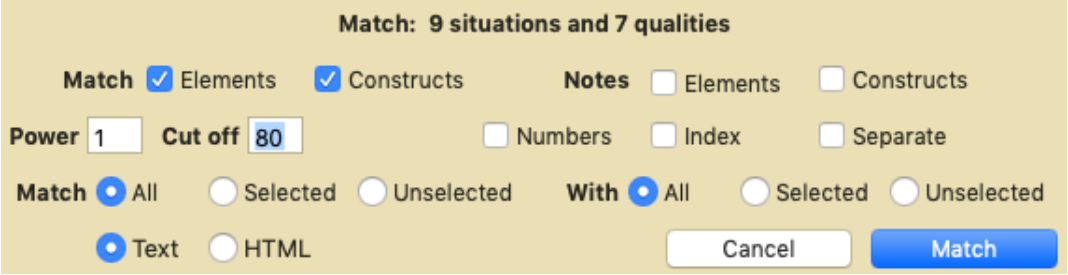


Figure 118: RepGrid *Match* dialog

The first row of check boxes determine whether: matches are output for elements, constructs, or both; element and construct notes are shown.

The second row specifies: the ***Power*** of the exponent used in the Minkowski metric used to compute matching scores (§5.3); the ***Cut off*** match value below which matches will not be shown; whether element and construct numbers will shown; whether the Honey index will be shown (Honey, 1979; Jankowicz, 2004); and whether the matches are separated by element or construct, or all sorted together.

The two sets of three radio buttons in the third row select whether: all, selected or unselected items are matched with all, selected or unselected items. The selection of elements and constructs is set up in the ***Elements*** and ***Constructs*** panes, respectively (and is independent of the state of the ***TextAnalyze Select*** check boxes).

The two radio buttons at the beginning of the bottom row select whether the matches are shown in a text window or as an HTML table in your browser window.

Figure 119 shows the output produced when one clicks on the ***Matches*** button with ***Text*** or ***HTML*** selected.

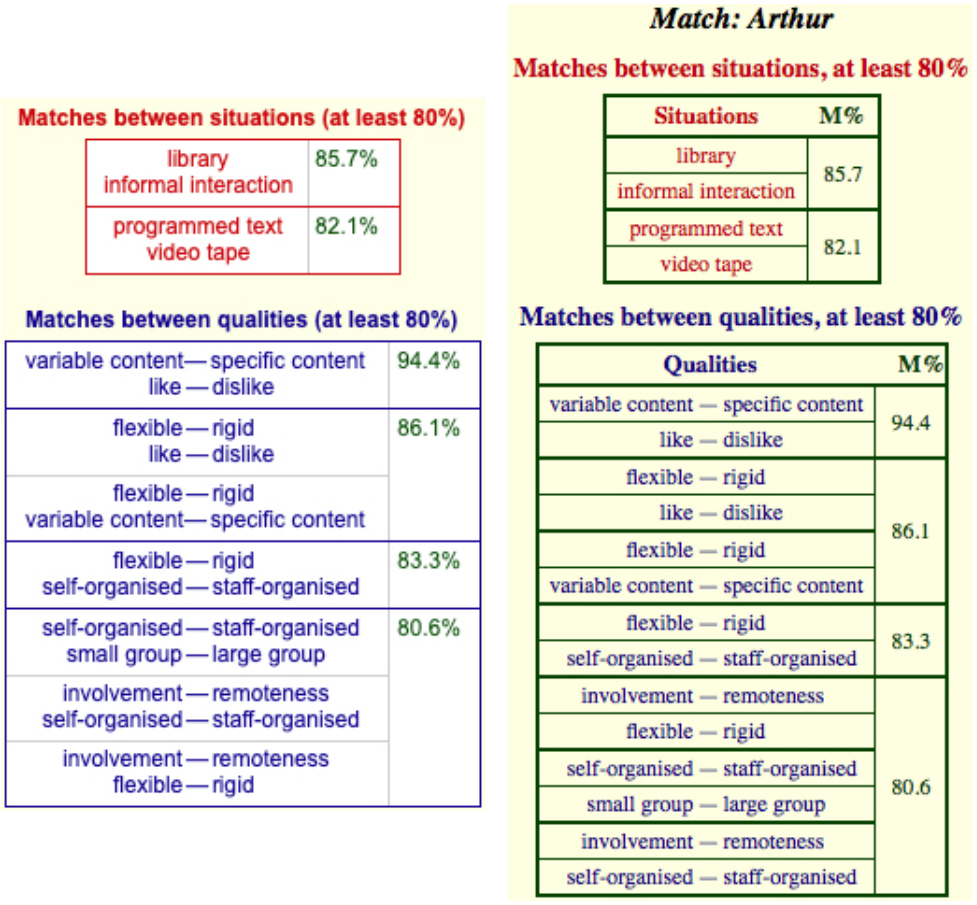


Figure 119: Element and construct matches—left text, right HTML

Figure 120 shows the *Match* dialog set up to show the matches Honey index between the construct *like—dislike* which has been selected and all the other constructs.

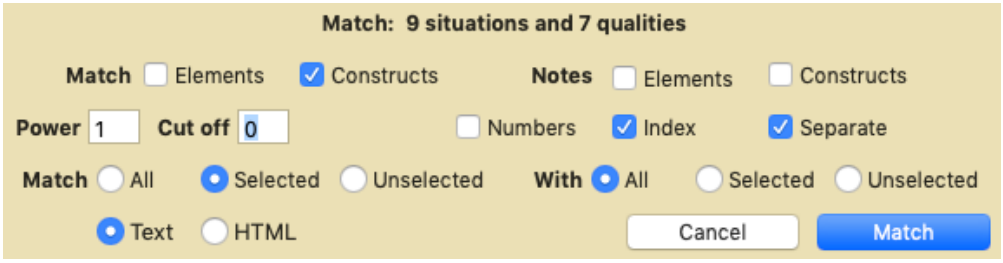


Figure 120: Selecting construct matches with Honey indices

Figure 121 shows the output produced when one clicks on the *Matches* button.

Match Arthur

Matches between qualities (at least 0%)

like — dislike	
variable content — specific content	94.4% H 1
flexible — rigid	86.1% H 2
involvement — remoteness	77.8% I 3
self-organised — staff-organised	75.0% I 4
no equipment — equipment	63.9% L 5
large group — small group	61.1% L 6

Match: Arthur

Matches between qualities

Quality	M%	T	R
like — dislike			
variable content — specific content	94.4	H	1
flexible — rigid	86.1	H	2
involvement — remoteness	77.8	I	3
self-organised — staff-organised	75.0	I	4
equipment — no equipment	63.9	L	5
small group — large group	61.1	L	6

Figure 121: Construct matches with Honey indices—left text, right HTML

Honey (1979) defined his index for purposes of content analysis and Jankovicz (2004) proves a detailed exposition of his method. The ***RepGrids*** tool for analyzing multiple grids in Rep Plus provides computational support for carrying out such content analyses including the use of indicators such as matches, mode scores, and Honey indices.

5.7.1 Using ideal elements derived from classes in a Match analysis

The ***Classes*** pane provides the option to include ideal elements derived from the classes in the grid supplied to any analysis program. Doing so with the *Matches* analysis provides useful insights into how the use of ideal elements entered directly may be used to solve a decision problem, and what further contribution is made through the extended properties available to specify intersects as classes.

If the ***Include*** checkbox in the ***Classes*** pane is checked for the contact lens prescription grid as discussed in §5.1.2 and the *Match* dialog is brought up it shows 32 elements, the 24 elements in the grid plus the 7 ideal elements generated from the classes. The ideal elements are automatically selected when they are added so that requesting the 100% matches of all the separated unselected elements with the selected ones will show what classes match each of the 24 cases. In addition the ***Constructs*** pane has been set to include all the constructs other than the lens prescription in the matches.

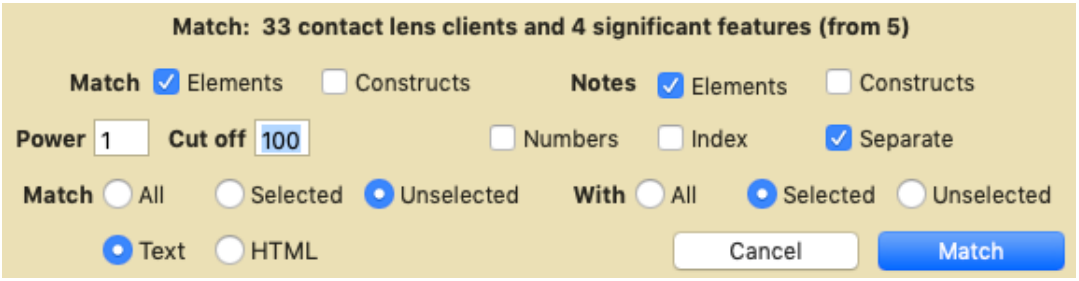


Figure 122: *Match* dialog to match each contact lens case against ideal elements from classes

Figure 123 shows the matches in two columns. The classes ***Consider soft*** and ***Prescribe soft*** generate the same ideal element with the only difference being in the comment field of the latter, ***prefer***

exception soft or exception reduced, and similarly for consider and prescribe hard. Consequently every case is matched against a consider/prescribe pair but, as the comments indicate, the prescribe match should not be taken into account if there is an exception match.

Matches between contact lens clients (at least 100%, based on selected significant features)

no lens 1	
lens infeasible	100.0%
soft lens 1	
lens feasible soft feasible prescribe soft (prefer exception soft)	100.0%
no lens 2	
lens infeasible	100.0%
hard lens 1	
lens feasible hard feasible prescribe hard (prefer exception hard)	100.0%
no lens 3	
lens infeasible	100.0%
soft lens 2	
lens feasible soft feasible prescribe soft (prefer exception soft)	100.0%
no lens 4	
lens infeasible	100.0%
hard lens 2	
lens feasible hard feasible prescribe hard (prefer exception hard)	100.0%
no lens 5	
lens infeasible	100.0%
soft lens 3	
lens feasible soft feasible prescribe soft (prefer exception soft)	100.0%
no lens 6	
lens infeasible	100.0%
hard lens 3	
lens feasible hard feasible prescribe hard (prefer exception hard)	100.0%

no lens 7	
lens infeasible	100.0%
soft lens 4	
lens feasible soft feasible prescribe soft (prefer exception soft)	100.0%
no lens 8	
lens infeasible	100.0%
no lens 9*	
lens feasible hard feasible exception hard [1] prescribe hard (prefer exception hard)	100.0%
no lens 10	
lens infeasible	100.0%
no lens 11*	
lens feasible soft feasible exception soft prescribe soft (prefer exception soft)	100.0%
no lens 12	
lens infeasible	100.0%
hard lens 4	
lens feasible hard feasible prescribe hard (prefer exception hard)	100.0%
no lens 13	
lens infeasible	100.0%
soft lens 5	
lens feasible soft feasible prescribe soft (prefer exception soft)	100.0%
no lens 14	
lens infeasible	100.0%
no lens 15*	
lens feasible hard feasible exception hard [2] prescribe hard (prefer exception hard)	100.0%

Figure 123: Matches of contact lens cases against ideal elements from classes

Thus the match analysis based on the ideal elements provides a solution to the lens prescription problem but leaves the inference of preferring, or taking into account, exceptions to the user.

It might be regarded as a ***decision support system*** designed to aid people in making better decision rather than an ***expert system*** that carries out the full inferential process. For some types of problem, the long-standing technique of matching ideal elements in conceptual grids may be seen as as a viable alternative to the logical inference techniques of artificial intelligence, or more generally, these techniques may be seen as similar approaches to supporting and emulating aspects of human intelligence.

If one is interested in modelling human decision-making then both the grid and logical inference models might be over-structured. For example, it might be appropriate to factor the grid of ideal elements in a way that makes each stage of the decision process dependent on only one construct. The initial step might be based on the construct of astigmatism and suggest prescribing a hard lens if the client is astigmatic and a soft lens otherwise, subject no exceptions applying. The next step might be based on the constructs of myopia and presbyopia and the possible exceptions based on intersects of them. The final step on either branch might be based on the construct of tear production and the exception if it is reduced. This interpretation satisfies Cendrowska's requirements and is a simple and natural representation of a readily learnt anticipatory process that can be derived from the classes or the ideal elements based on them.

Such considerations suggest that that the personal construct framework represented in conceptual grids and associated classes or intersects might be useful in the study of human rationality. Cendrowska's contact lens problem is rather more complex than the simple logical tasks that have been used in empirical studies of human rationality but analyzing those tasks in terms of matching ideal elements might provide insights into the difficulties that people have in solving what are apparently very simple logical tasks.

5.7.2 Match analysis of Wason's card selection task

For example, consider Wason's original card selection task: *"The subjects were presented with the following sentence, 'if there is a vowel on one side of the card, then there is an even number on the other side,' together with four cards each of which had a letter on one side and a number on the other side. The task was to select all those cards, but only those cards, which would have to be turned over in order to discover whether the experimenter was lying in making the conditional sentence."* (Wason, 1968, p.273). There are four possible card types and two possible visible sides, so there are eight possible cases that may be represented in a grid (Figure 124).

The first two constructs in the grid represent the factual situation, the next two the subject's knowledge of it, and the last whether the situation is consistent with the experimenter's statement or shows it to be a lie. A situation where both sides of the card are visible and provide evidence of a lie can be defined as a class ***visible is vowel and visible is odd*** (Figure 125).

If this card is included as an ideal element and the matches are computed based on the two ***visible*** constructs representing the subject's knowledge then, because only one side of the card is visible, the maximum match is 50% (Figure 126). However, the four situations that matched are the ones where a vowel or an odd number is visible which are the correct selections. What requires explanation with Wason's task is that the majority of subjects did not arrive at this solution and made incorrect selections.

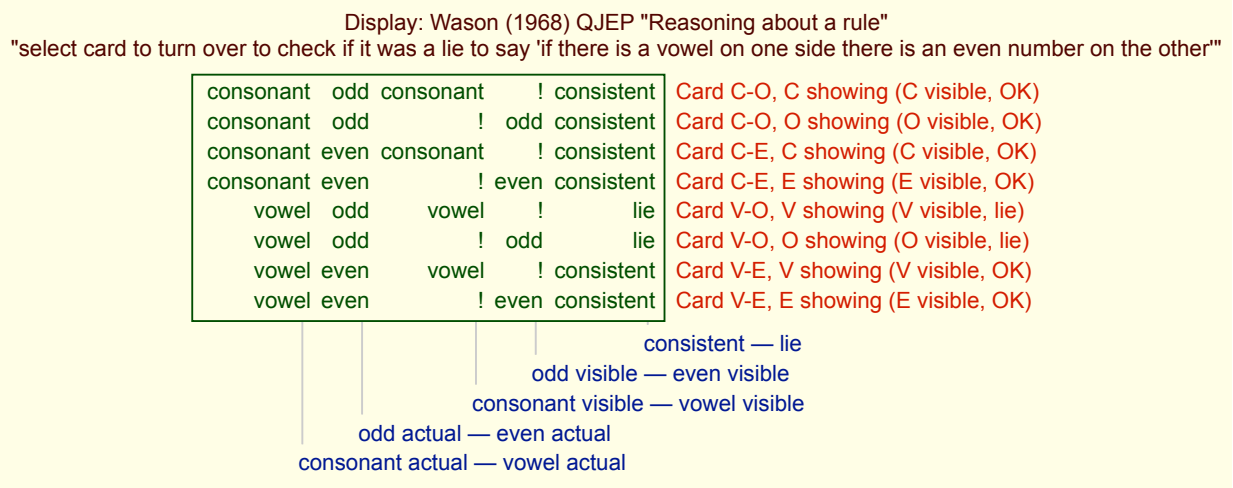
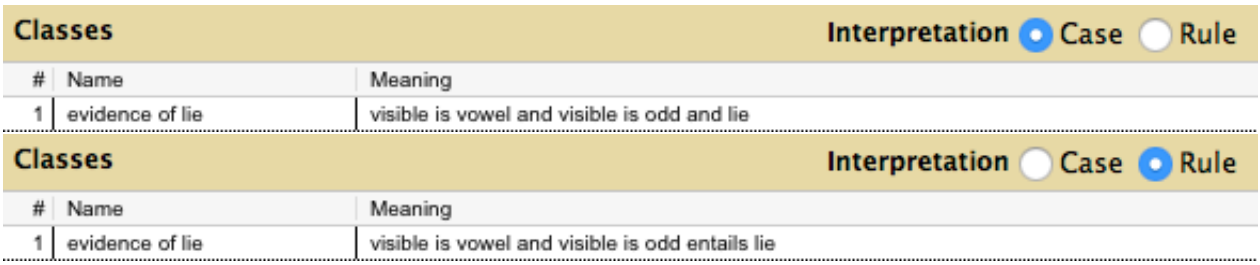


Figure 124: Grid representing the possible situations of Wason's card selection task



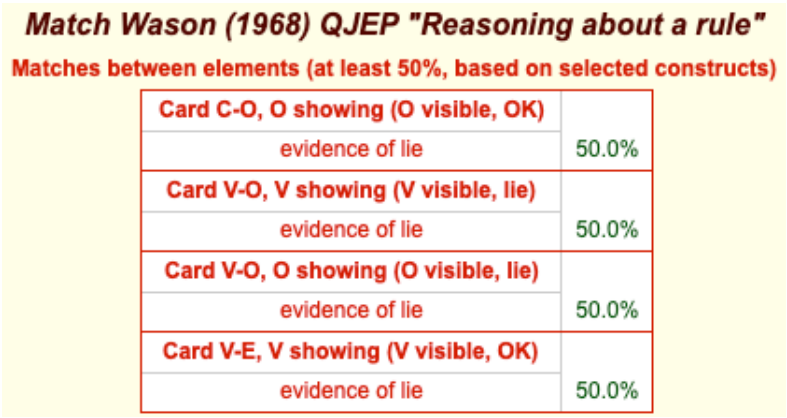
Classes Interpretation ● Case ○ Rule

#	Name	Meaning
1	evidence of lie	visible is vowel and visible is odd and lie

Classes Interpretation ○ Case ● Rule

#	Name	Meaning
1	evidence of lie	visible is vowel and visible is odd entails lie

Figure 125: Class specifying the state of card that provides evidence of a lie—as case or rule



Match Wason (1968) QJEP "Reasoning about a rule"

Matches between elements (at least 50%, based on selected constructs)

Card C-O, O showing (O visible, OK)	
evidence of lie	50.0%
Card V-O, V showing (V visible, lie)	
evidence of lie	50.0%
Card V-O, O showing (O visible, lie)	
evidence of lie	50.0%
Card V-E, V showing (V visible, OK)	
evidence of lie	50.0%

Figure 126: Matches of the class as an ideal element to the situations of Wason's card selection task

One explanation might be that the inference is difficult for people that the statement ***if there is a vowel on one side of the card, then there is an even number on the other side*** is a lie if a card has a vowel on one side and an odd number on the other. If the problem was stated in terms of a violation, that ***a card with a vowel on one side and an odd number on the other is not allowed*** then the two conditions

where a card should be checked are both contained in the problem statement and the selection task might be easier.

Such considerations have triggered wide-ranging research into variants of Wason's task and other reasoning problems, and there is now a massive literature of empirical data requiring explanation together with many theories suggesting further investigations (Evans et al., 1993; Hardman and Macchi, 2003; Stenning and Lambalgen, 2008; Adler and Rips, 2008). The conceptual modelling tools in Rep Plus provide a personal construct psychology framework within which to represent and compare such results and theories.

5.8 Analysis of selected elements and constructs

RepGrid makes it possible to display or analyze only part of a grid. At the bottom left of the ***Elements*** (§3.2) and ***Constructs*** (§3.3) panes are check boxes specifying that only selected items should be analyzed. In essence the grid to be analyzed is reduced to contain only the selected items, and hence all the analyses may be used on a partial grid without actually deleting elements or constructs.

In ***Think Again*** Shaw and McKnight (1981) present an example of the use of conceptual grids in decision support that illustrates the application of selected and weighted constructs in grid applications. As elements the user enters seven cars as potential choices together with an ***ideal car*** that will serve to elicit his preferences, and has rated the cars on relevant constructs. Figure 127 shows the grid developed to help in the choice of a car, and Figure 128 shows a Focus analysis where the cars clustered with ***ideal car*** suggest a basis for making an appropriate choice.

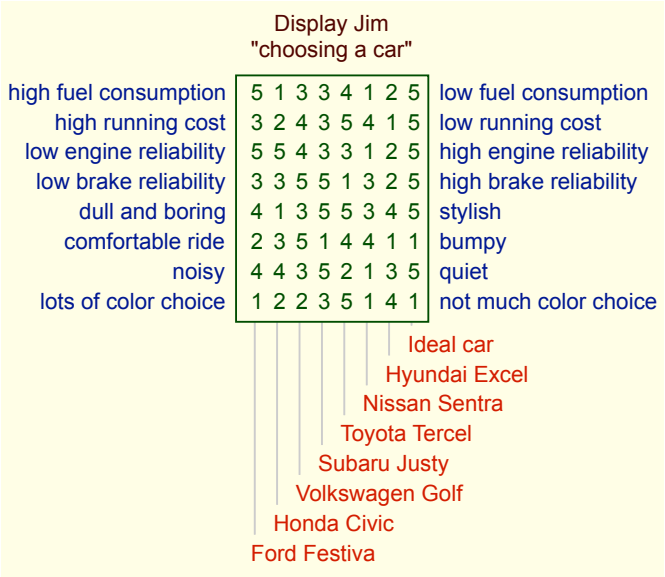


Figure 127: Car choice decision-support grid

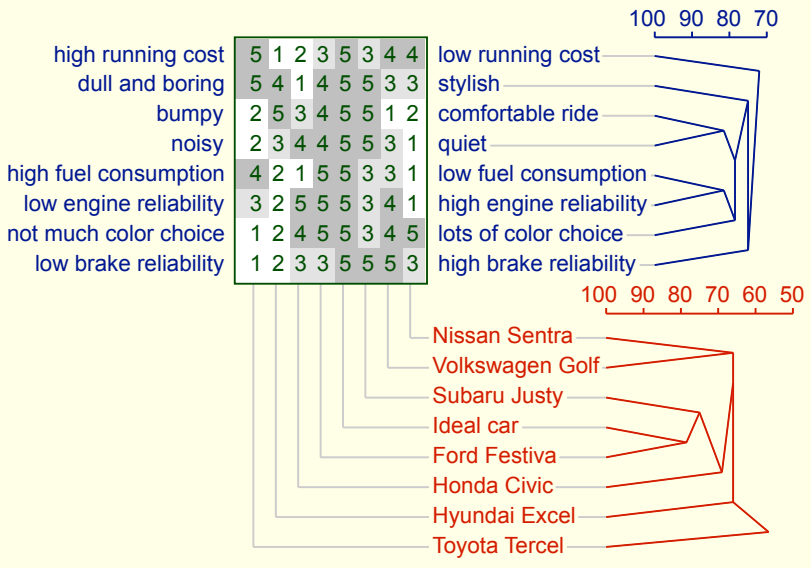


Figure 128: *Focus Cluster* analysis of car choice decision-support grid

After seeing this analysis he may be interested in clustering based only on what he sees as the most significant constructs, and can restrict the analysis by selecting these in the ***Constructs*** pane and clicking on ***Analyze Select*** as shown in Figure 129.

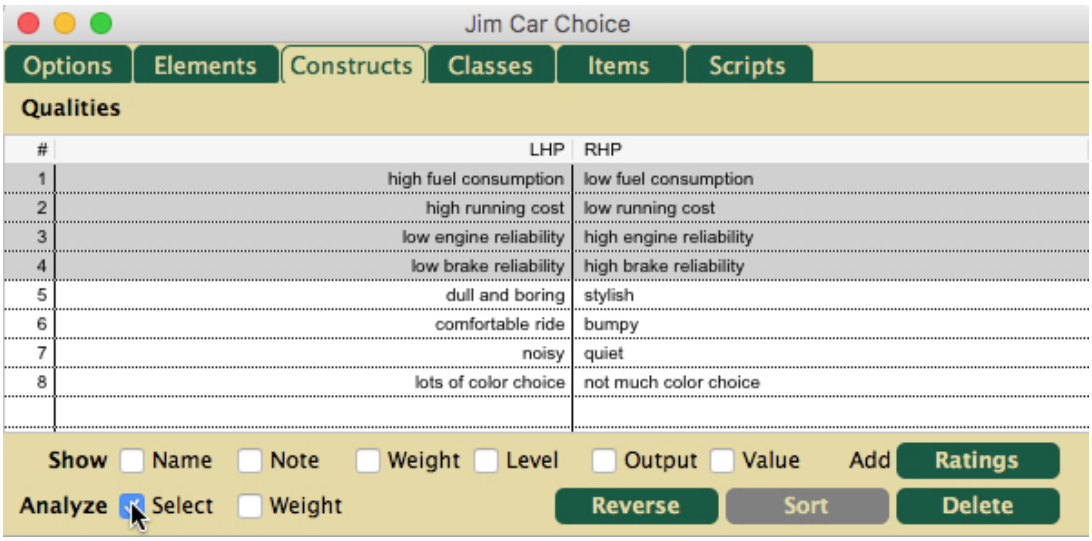


Figure 129: Car choice decision-support grid with analysis based on selected constructs

Clicking on the ***Focus*** button now results in an analysis based only on the selected constructs (Figure 130). The elements to be used can be selected on the ***Elements*** pane in the same way. Using ***Analyze Select*** for both elements and constructs provides the facility to display or analyze partial grids without editing the grid data.

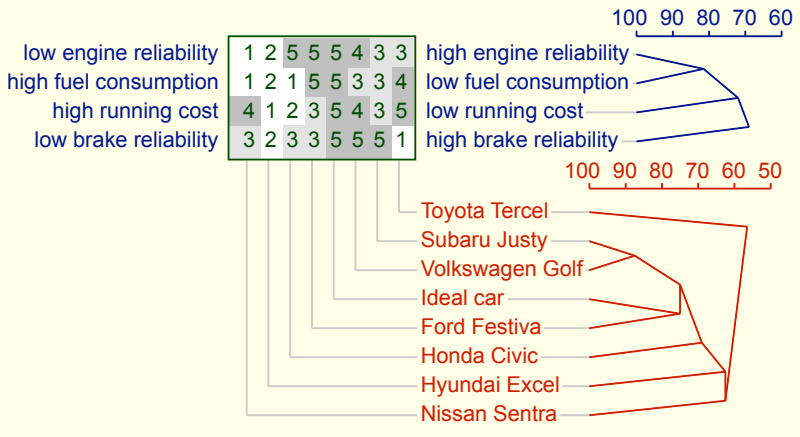


Figure 130: *Focus Cluster* analysis of car choice decision-support grid based on selected constructs

5.9 Analysis of weighted elements and constructs

Grid analysis can be refined in a more nuanced fashion by ***weighting*** elements and constructs. At the bottom left of the ***Elements*** and ***Constructs*** panes are check boxes specifying that the weight values should be used in analyses, notably *Display, Synopsis, Focus Cluster, PrinGrid Map,* and *Match*. The use of weights is indicated in the title of the output and the weight is shown in square parentheses by the elements and constructs in the plot.

Figure 131 shows the ***Constructs*** pane of the car choice grid where the user has weighted the constructs to indicate their relative importance to him in deciding which car best satisfies his needs. The ***Show Weight*** and ***Analyze Weight*** check boxes have been selected to show the weights and use them in analysis.

Qualities

#	LHP	RHP	Wt
1	high fuel consumption	low fuel consumption	8
2	high running cost	low running cost	9
3	low engine reliability	high engine reliability	6
4	low brake reliability	high brake reliability	10
5	dull and boring	stylish	5
6	comfortable ride	bumpy	6
7	noisy	quiet	4
8	lots of color choice	not much color choice	2

Show ☐ Name ☐ Note ☑ Weight ☐ Level ☐ Output ☐ Value Add Ratings

Analyze ☐ Select ☑ Weight Reverse Sort Delete

Figure 131: Car choice decision-support grid with weighted constructs

Clicking on the ***Display*** button displays the grid with the fact that weights are in use noted in the title, and the weight values displayed on the right of each construct (Figure 132).

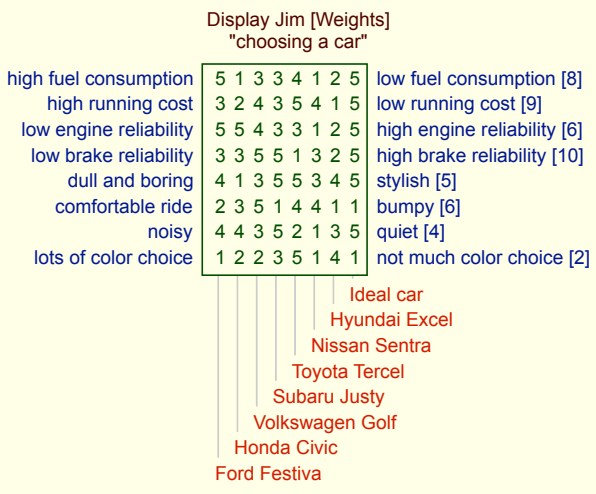


Figure 132: *Display* of car choice decision-support grid with construct weights

Only the relative values of weights are significant, and they may take any value from zero upwards. It is common to use 0 through 10 or 0 through 100. In the calculation of match scores for the ***Focus*** and ***Matches*** analyses, construct weights are used to weight differences in values on constructs in computing element matches, and element weights are used in computing construct matches. The effect on the analysis is the same as if the item with weight ***n*** had been entered in the grid ***n*** times.

Figure 133 shows the weighted ***Focus*** analysis for the car choice grid.

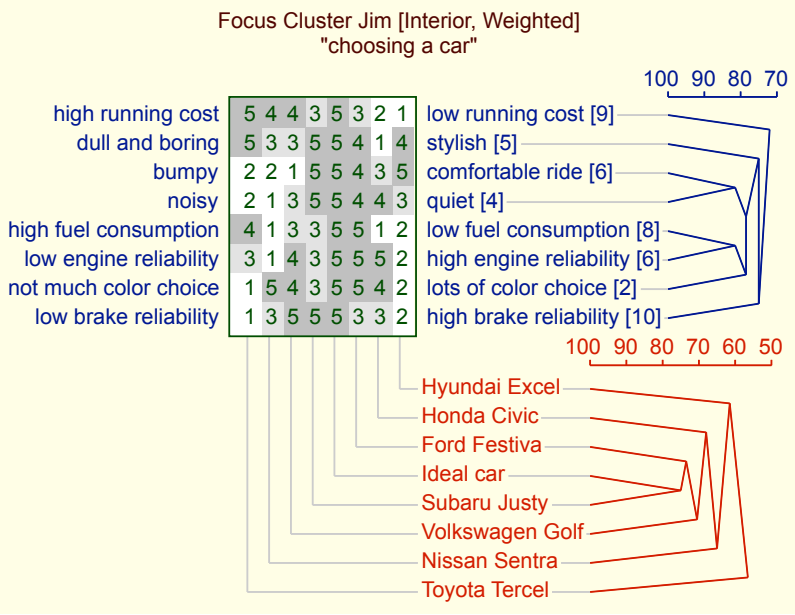


Figure 133: *Focus Cluster* analysis of car choice decision-support grid based on weighted constructs

In the unweighted analysis of Figure 128 ***Ideal car*** clusters most closely with ***Ford Festiva.*** However, in the weighted analysis of Figure 133, ***Ideal car*** clusters most closely with ***Subaru Justy.***

If the user wishes to see the rank ordering of the matches of the ***Ideal car*** against all the others then ***Ideal car*** may be selected in the ***Elements*** pane, and ***Matches*** selected in the ***Analysis*** menu with the parameters selected as shown below to show the element matches of the selected element with all the others (Figure 134). This results in the list of matches shown in Figure 135.

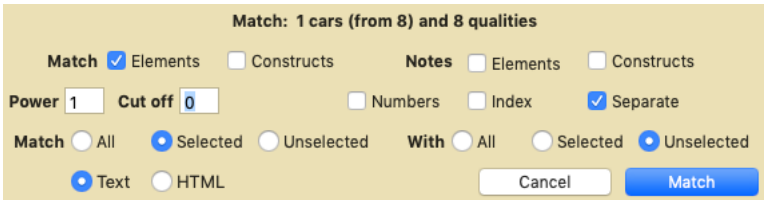


Figure 134: *Match* analysis to show how *Ideal car* compares the actual cars

Matches between cars (at least 0%, based on weighted qualities)

Ideal car	
Subaru Justy	75.0%
Ford Festiva	73.5%
Volkswagen Golf	62.5%
Toyota Tercel	51.0%
Honda Civic	41.5%
Hyundai Excel	36.5%
Nissan Sentra	35.5%

Figure 135: Weighted *Match* analysis of decision-support grid

Weights may also be used in other analyses. Figure 136 shows the *PrinGrid* output with the construct weights specified. ***Ideal car*** appears to be discriminated by the first component plotted on the horizontal axis, with ***Ford Fiesta*** being the nearest to it.

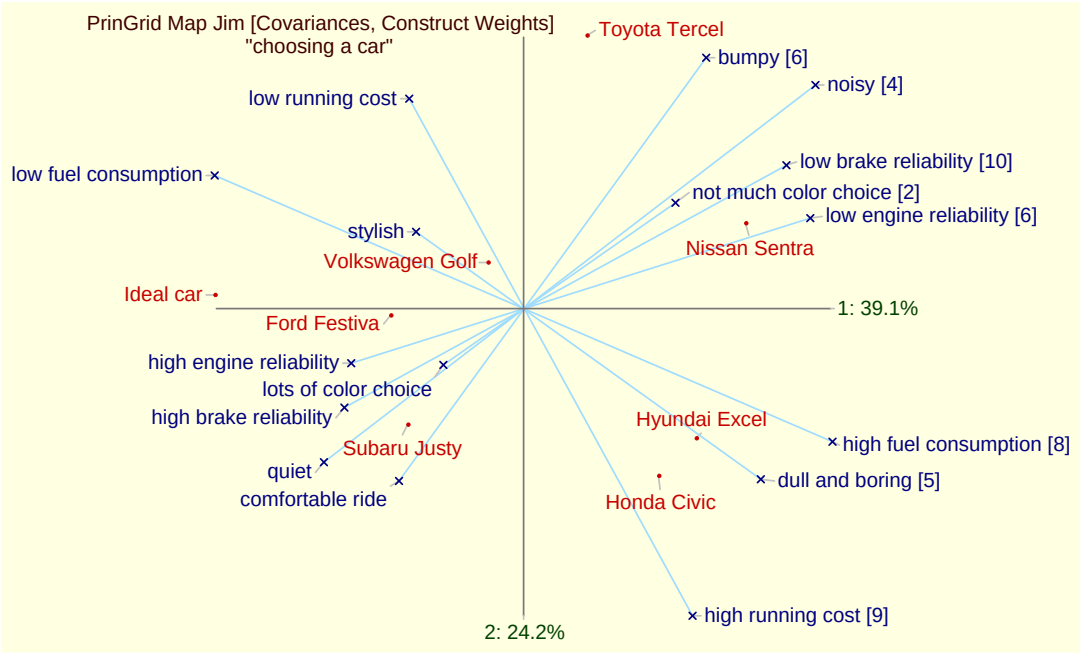


Figure 136: Weighted *PrinGrid* analysis of decision-support grid

The element and construct weights, if selected, are used multiplicatively in the calculation of the distance matrix for a principal components analysis. The effect of the weighting of the principal components analysis is equivalent to that of putting each element and each construct in the grid several times according to the weight allocated to it.

The results are similar to, but not the same as, those of the *Focus* and *Match* analyses, and this is because those analyses are based on a *boxcar* metric which sums absolute distances whereas the principal components analysis uses a *Euclidean* metric which sum the square of the distances and hence gives a greater weight to larger differences.

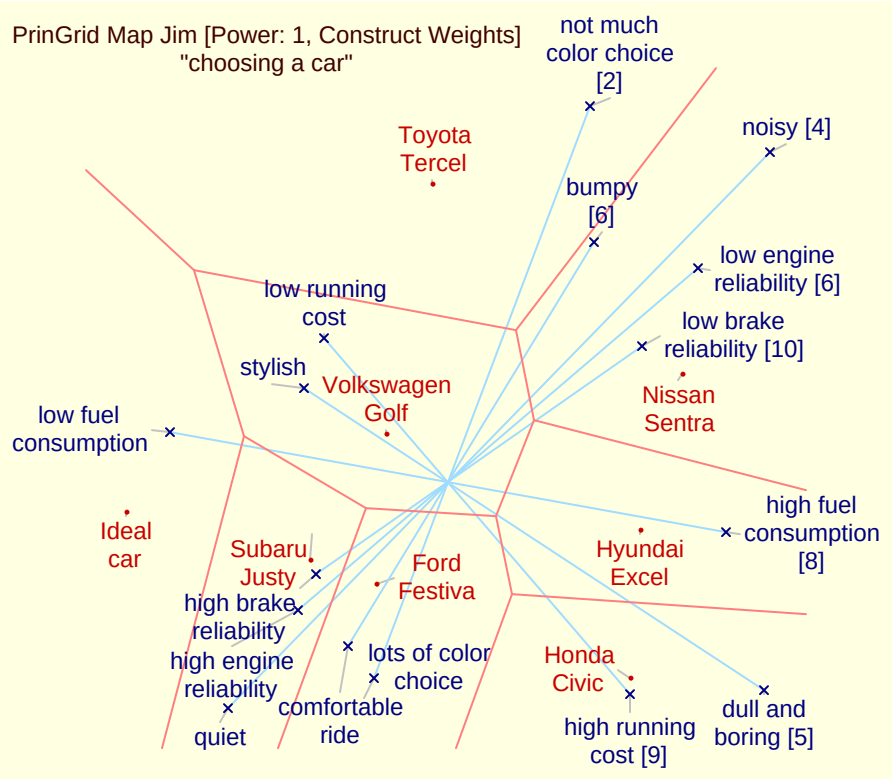


Figure 137: Weighted *PrinGrid* analysis of decision-support grid—Power 1.0, Voronoi diagram

Issues of alternative metrics and related research in the literature have been discussed in §5.4.3. The psychometrics literature suggests the *boxcar* metric corresponding to a Minkowski power of 1.0 is appropriate to human decision-making, which indicates that Figure 137 is more appropriate than Figure 136. However, the role of the grid analysis is to aid the client(s) to better understand the basis of the decision, not make it for them, and the grid analyses are primarily a basis for discussion, a ***conversation with self*** or a group discussion with those who will be effected by the decision. Study of the effect of different weightings and metrics can be used to ensure that the analyses reflect the intrinsic indeterminacy of multivariable decision-making. Some possibilities may be clearly unattractive, but others involve trade-offs between different desiderata that may be clarified by the analyses.

6 Style—managing the font and colour scheme of analyses

Clicking on the ***Style*** button brings up the dialog controls the colour schemes and grid identifier used in graphic output from all the analyses (Figure 138).

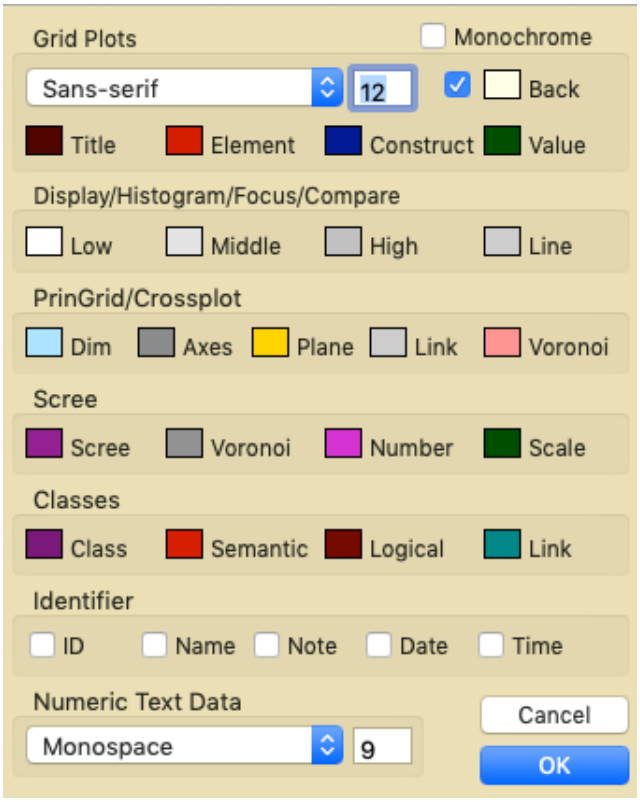


Figure 138: RepGrid ***Style*** dialog

All graphic output in Rep Plus is in the form of ***nets***, graphic data structures that are detailed in the the RepNet Manual, can be edited and annotated in RepNet, and can be converted to a variety of graphic formats suitable for high-quality publication in documents, presentations and the web (§7).

Grid Plots The ***Grid Plots*** panel is common to all analyses. If the ***Monochrome*** checkbox is checked then all the analyses are in shades of grey determined only by the intensity of the colour selected, not its hue. The popup menu and associated text box specify the font and size of text output. The ***Back*** checkbox and associated swatch specifies whether the analyses have a coloured background and, if so, what colour is to be displayed. Colours may be specified for: the title; element names and trees; construct names and trees; and values of ratings and component percentages.

Display/Histogram/Focus/Compare The ***Display/Histogram/Focus/Compare*** panel is specific to those analyses. Colours may be specified for: the background tints to low, medium and high rating values; and lines joining element and construct names to the grid.

PrinGrid/Crossplot The ***PrinGrid/Crossplot*** panel is specific to those analyses. Colours may be specified for: the construct dimension lines; plot axes; X-Z plane and lines dropping to it in 3D output; link lines connecting element and construct names to their coordinates; and Voronoi diagrams.

Scree The ***Scree*** panel is specific to that analysis. Colours may be specified for: the scree plot; comparison plot; estimated number of underlying dimensions; and scales.

Clases The ***Clases*** panel is specific to the listing of the class meanings. Colours may be specified for: the class name; semantic logical terms; logical connectives; and values.

Identifier The ***Identifier*** panel is common to all analyses. It specifies what fields will be used to construct a phrase that identifies the grid. The ***ID*** is an item named ***ID*** which the user can enter as a customized identifier for the grid. If its use is specified and no ***ID*** item has been entered then the ***UID*** item that RepGrid automatically creates to provide the grid with a unique identifier is used. The ***Name*** and ***Note*** are those entered in the fields of the ***Options*** pane. If both are specified then the ***Note*** is placed in parentheses after the ***Name.*** The ***Date*** and ***Time*** items are those item that RepGrid automatically creates when a grid is created. If no identifier fields are specified then the default of ***Name*** and ***Note*** is used.

Numeric Text Data The ***Numeric Text Data*** specifies the font to be used for data such as arrays. The font should be a monospaced to ensure the proper alignment of the data.

6.0.1 Colour selection

The colour swatches allow colours to be specified using either standard named colours or the colour selection widgets native to the platform on which Rep Plus is running. Clicking in the left half of a swatch brings up a menu of the 140 standard colours specified in the World Wide Web CSS documentation from which a colour may be selected (Figure 139).

Rep Plus tracks the last colour selected and this is presented as the first option at the top of the menu so that it is easy to replicate the previous colour selected. Clicking in the right half of a swatch brings up a colour selection widget native to the platform on which Rep Plus is running allowing for custom colours to be specified. Examples are provided in the RepNet Manual which also details how the grid colours are translated into colour styles in RepNet.

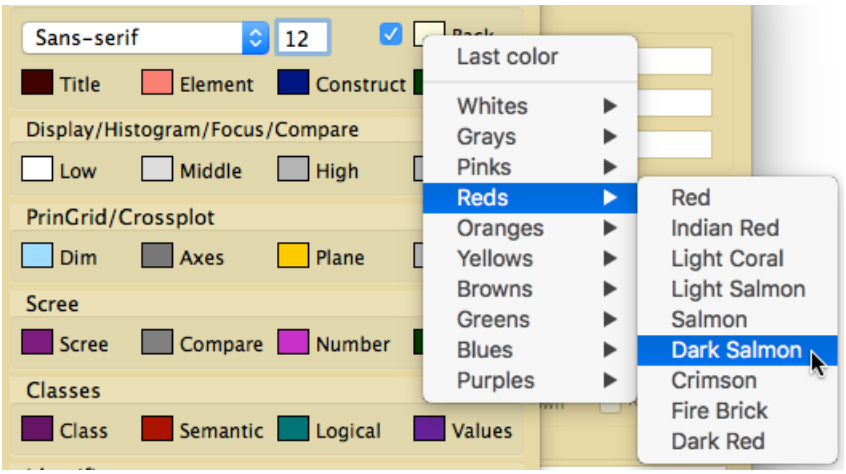


Figure 139: Colour selection when left half of swatch is clicked

7 Editing and exporting RepGrid output

The graphic plots and textual output from the RepGrid analyses may be used in reports, papers, theses or web pages, and Rep Plus supports their conversion to forms appropriate for publication. The plots produced by RepGrid analyses are vector graphics nets in ***RepNet*** format. The text outputs are fully styled documents in ***RepDoc*** format. Both can can be saved, exported, dragged and copied/-pasted in a variety of formats to document processors to produce a publication-quality documents.

RepNet organizes graphics as a set of nodes and links between them. Nodes can be selected by clicking on them and selected nodes are indicated with a light blue surround. Mousing down or CTL-clicking or right clicking in the graphic outside the nodes brings up a popup menu allowing the selected items to be exported as SVG, PNG or JPEG files (Figure 140). All three are standard graphic formats for the web. PNG and JPEG are generally accepted by word processors. SVG is a graphic interchange format accepted by graphics editors such as such as Illustrator, EazyDraw and Inkscape.

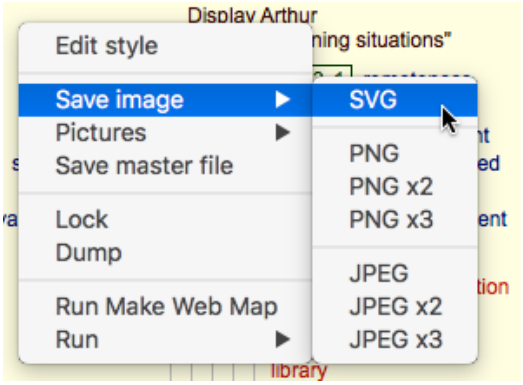


Figure 140: RepGrid graphic display of the data in a grid

RepNet also provides extensive graphic editing facilities such that the RepGrid analyses can be annotated for publication. More details are available in the RepNet manual.

Copy and paste of vector graphics is problematic with some word processors, particular for images containing Unicode text. When you select ***Copy*** from the ***Edit*** menu a bitmap image at screen resolution is made available in the clipboard for applications that cannot decode a RepNet file. You can increase the image resolution by using the ***Scale*** setting in each analysis dialog, and the font size setting in the ***Style*** dialog. Increase each by the same factor of 2 or 3 and generate a plot that is two or three times as large as usual. Copy this as a bitmap, paste it into your word processor, and rescale it to be 2 or 3 times smaller. The image quality when printed will then be substantially higher.

Exporting the graphic to a file, SVG or a 2x or 3x PNG or JPEG, provides an alternative process for exporting high-quality images to publications and presentations.

Text in RepDoc can be dragged or copied/pasted to other RepDoc documents or other document processes whilst in RTF retaining all styling. Graphic output from RepGrid can also dragged or copied/pasted to RepDoc documents to provide composite documents that can be saved or exported as RTF.

8 RepGrid/WebGrid integration

WebGrid (Gaines, 1995) was developed in 1994 as a way of making RepGrid functionality available on the World Wide Web and public WebGrid servers have long been used as an alternative to the stand-alone RepGrid program (Gaines and Shaw, 1996, 1997, 1998; Shaw and Gaines, 1996b, 1998). Whilst WebGrid proves the same functionality as RepGrid, the user interfaces for grid entry, elicitation and analysis are different, and Rep Plus is designed to make it simple to move grid data back and forth between the two programs so that users may move freely between them.

Rep Plus includes WebGrid and can act as a web server over the Internet, a local intranet, or on the same machine that is running the Rep Plus suite of programs. If RepGrid and WebGrid are running on the same machine the user interfaces of each provide capabilities to move grid data back and forth between them so that the two program can be used in an integrated fashion for different aspects of a task as the user prefers.

8.1 Transferring grid data from RepGrid to WebGrid

At the bottom left of the RepGrid window is a button labelled ***WebGrid*** (Figure 141). If clicked it starts up the WebGrid server for local use (if it is not already running) and copies the data from the grid being edited in RepGrid to WebGrid.

Figure 141: RepGrid to WebGrid transfer button

In addition, licking on the menu symbol on the left of the WebGrid button brings up a menu that allows one to select what WebGrid page will open with the grid data (Figure 142), for example one may transfer to *Display* and see the grid data displayed. The default choice *WebGrid* takes one to the WebGrid main page.

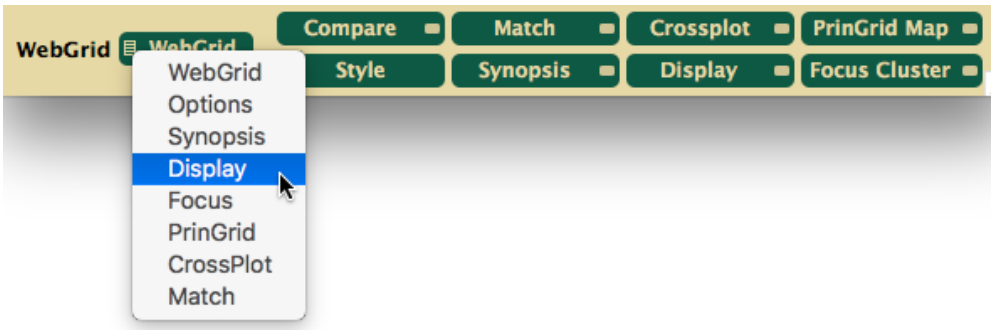


Figure 142: RepGrid to WebGrid transfer menu

Figure 143 shows the house choice grid opening in WebGrid after ***WebGrid*** has been clicked in its RepGrid window (Figure 20).

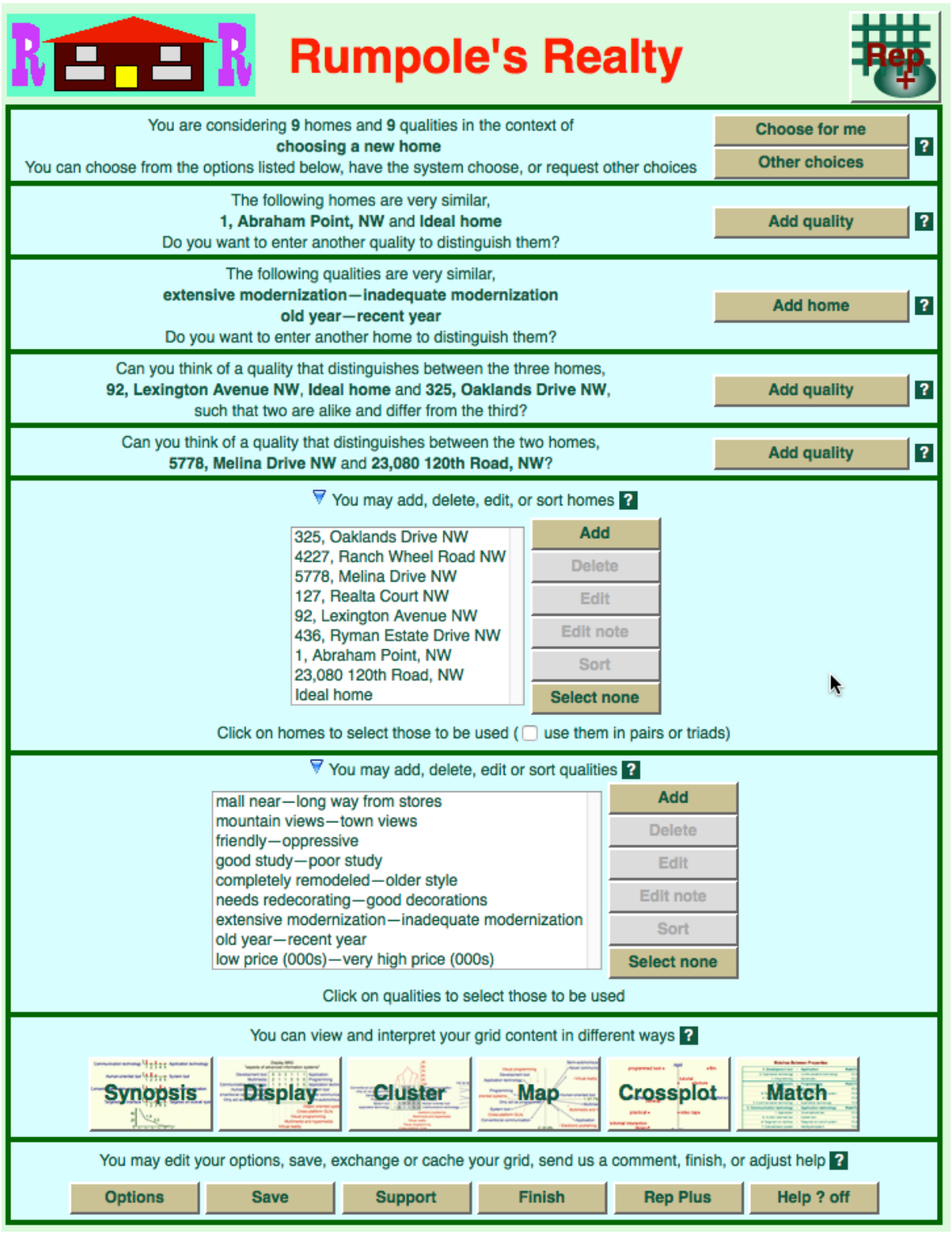


Figure 143: RepGrid grid data opened in WebGrid

The house choice grid was developed in WebGrid and uses its customization capabilities to restyle its pages. The CSS and HTML code to do so is automatically included as an item of data in the grid file, and shows up in the ***Items*** pane (§3.5). The item has no effect on the operation of RepGrid but is recognized as styling data in WebGrid.

The notes attached to the elements in the grid are in HTML with links to images and these are displayed when the elements themselves occur in appropriate contexts in WebGrid. Figure 144 shows the WebGrid page when the user clicks on ***Add quality*** in the fourth row to elicit a new construct from the triad of elements suggested. The element annotation contains links to relevant images that are displayed to help the use recollect the nature of the element.

Figure 144: Hypertext annotation of elements in WebGrid elicitation

Figure 143 shows how the annotation appears in the table of top matches for ***Ideal home.***.

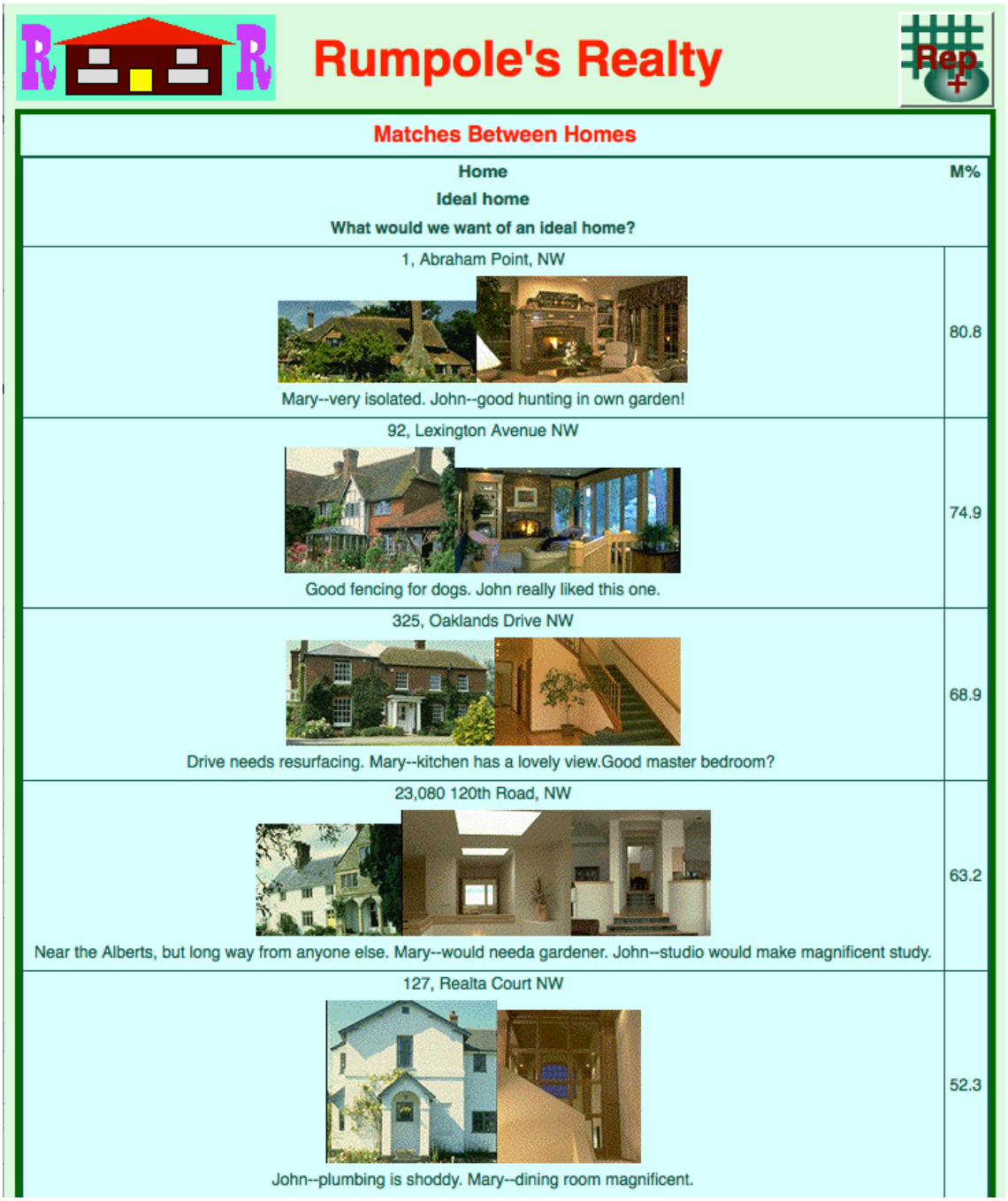


Figure 145: Hypertext annotation of elements in WebGrid list of matches

The element annotation for the top element is:

```
<p align="center"><img src="/RealEstate/H563.gif">
<img src="/RealEstate/R432.gif"><br>
Mary--very isolated. John--good hunting in own garden!<br>
```

The hypertext links are to images on the local server but could be to images or sounds anywhere on the web. Thus, the multimedia capabilities of web browsers enhance elicitation processes in WebGrid beyond those available in RepGrid. See the WebGrid manual for further details.

If WebGrid is not running on the same machine then grid data from a RepGrid file may may be uploaded using the ***Upload*** link on WebGrid's startup page.

8.2 Transferring grid data from WebGrid to RepGrid

When the WebGrid server recognizes it is running on the same machine as its client it modifies the user interface in the web browser to enable grid data being processed in the browser to be copied to Rep Plus to great a new grid window in RepGrid. A ***Rep Plus*** button is created at the bottom right of the WebGrid main page (Figure 143). Clicking on this copies the grid file to Rep Plus.

In addition to the ***Rep Plus*** button. the grid icon normally at the top right of every WebGrid page becomes a button with ***Rep*** + superimposed on the icon, and clicking on this also copies the grid data to Rep Plus and opens it in RepGrid. Thus, transferring data between WebGrid and RepGrid is a simple process that can be performed at any time.

Note that the grid that opens on either side of the transfer is a ***copy*** of that in the originating application. When transferring back to RepGrid the copy may be saved with the same name as the original grid or as a new version. If the old version is still open the user needs to close it without saving or otherwise manage the existence of multiple versions of the grid.

If WebGrid is running on a different machine then the grid data may be downloaded as a file either by clicking on the ***Save*** button at the main page and following the instructions there, or by simply saving the page as HTML. Rep Plus will open a WebGrid HTML file in RepGrid and extract the grid data from it.

9 Data Formats

Rep Plus supports a number of data formats for information transfer, including the basic file format of our earlier grid programs which is useful as a transfer format for grid data that needs to be digitally encoded for use in RepGrid. There is also an alternative spreadsheet format which can also be useful to encode data for fir RepGrid.

9.1 Basic grid format

The basic grid format used in Shaw's (1980) original repertory grid elicitation and analysis programs has been adopted by others and also provides a simple format for transfer of any grid data. Grids in this format have the following structure:-

Line 1	4 numbers separated by commas:- Number of elements—E Number of constructs—C Lower limit of rating scale Upper limit of rating scale
The next 3 lines are optional	
Line 2	Purpose
Line 3	Name
Line 4	Note
The ratings may be entered in one of two formats, packed or separated	
Line 5	Rating of Construct 1 on each element with no space ***e.g. 12345—ratings may include meta-values***
Line 5	Rating of Construct 1 on each element separated by commas ***e.g. 1,2,3,4,5—ratings may include meta-values***
The remaining items may be truncated and missing information will be filled appropriately	
Line 5+C	Left Pole of Construct 1
Line 6+C	Right Pole of Construct 1
Lines 7+C to 4+3*C	Remaining Constructs
Line 5+3*C	Element 1
Lines 5+3*C to 4+3*C+E	Remaining Elements
Line 5+3*C+E	Singular for Construct
Line 6+3*C+E	Plural for Construct
Line 7+3*E+C	Singular for Element
Line 8+3*E+C	Plural for Element

The grid data on "exploring the nature of learning situations" used in previous sections is shown below in this format.

```
9,7,1,5
exploring the nature of learning situations
Arthur
after class discussion
432151231
443251521
```

```
455113215
544351221
524451113
442351551
542251551
involvement
remoteness
flexible
rigid
equipment
no equipment
self-organised
staff-organised
small group
large group
variable content
specific content
like
dislike
lecture
tutorial
seminar
practical
film
library
programmed text
video tape
informal interaction
situation
situations
quality
qualities
```

A file in this format may be opened as a grid in RepGrid, or the file or the text may be dragged to the ***Rep 5 Manager*** window to create a new grid as (Figure 146).

As documented above, some of the lines in the grid data may be omitted and will be filled appropriately. This enables existing grid ratings to be transferred rapidly to RepGrid with other fields being added in the RepGrid editor as required. Below is a minimal version of the grid above.

```
9,7,1,5
432151231
443251521
455113215
544351221
524451113
442351551
542251551
```

If a file with this data is opened, or the data is dragged to a grid window and dropped on it, and the ***Display*** button is clicked then the output will be as shown in Figure 147. The elements and constructs have been given identifiers sufficient to support meaningful display and analysis, and the name, purpose, and full construct and element names may be entered in the appropriate RepGrid fields as required.

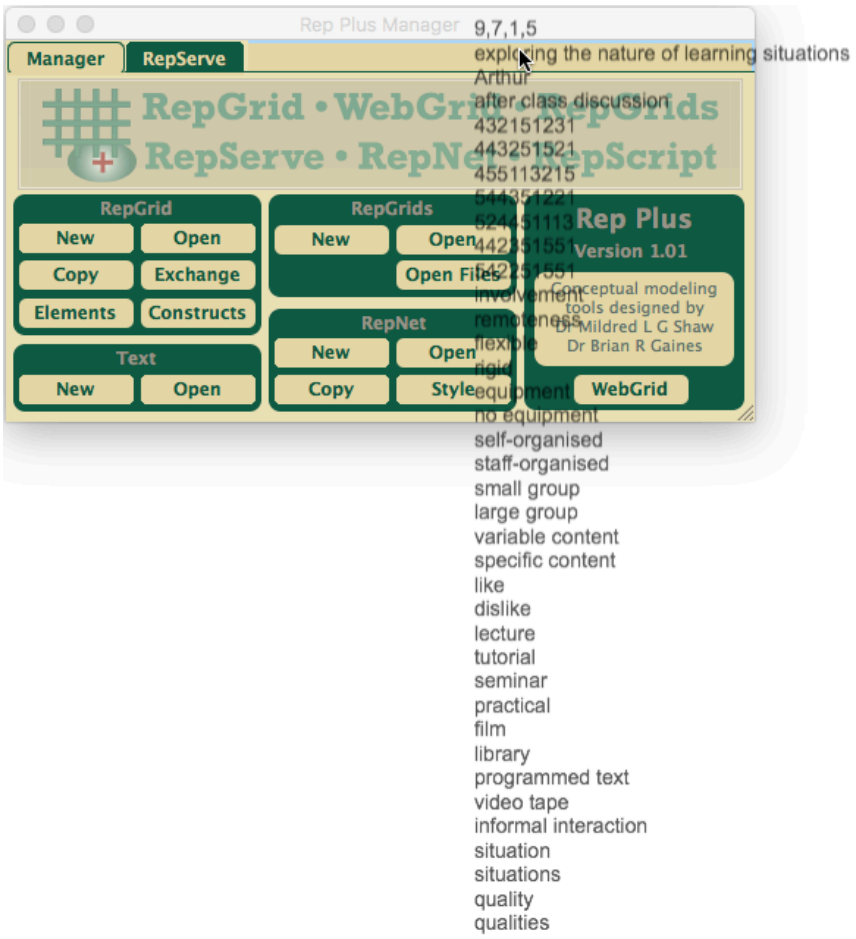


Figure 146: Grid data being dragged and dropped on the Rep Plus Manager window

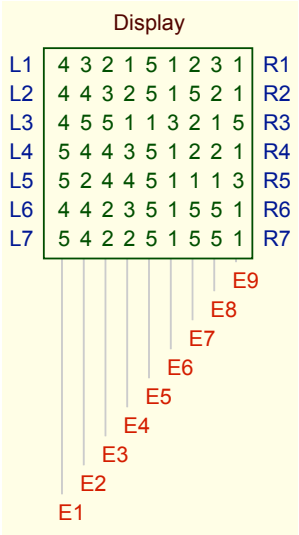


Figure 147: Minimal grid data displayed after being dragged and dropped

9.2 Spreadsheet grid entry format

A simple format for single or multiple grid data entry using spreadsheet programs was defined for RepGrid 2 and is supported by Rep Plus. It is based on the ***tab-delimited*** file format available in most spreadsheet programs.

The format is very simple, consisting of text separated by tab characters, and can also be generated in text editors and word processors, so that it provides another easy way to get data into RepGrid. Multiple grids may be stored in the same spreadsheet, and the format is designed to avoid the need to enter redundant data. For example, if several grids have the same scale this need be entered once only, similarly if several grids use the same elements, or the same constructs, these need be specified once only. This is particularly useful in entering grid data where multiple grids have common elements and/or common constructs.

Data is read in row by row commencing with the first row, and the various value settings persist unless overwritten by later settings. The first cell in a row normally contains a keyword such as ***Name***, ***Note***, ***Purpose***, ***Range*** or ***Grid***. The rest of the cells in that row contain data relating to the keyword. The keywords can be any order except that ***Grid*** must be the final one, and signifies that the grid data follows on several rows. The grid data must be terminated by a blank line (or by being the final data in the spreadsheet). Outside grid data, blank lines, and those not commencing with a keyword, are skipped and may be used to improve appearance and for comments.

Figure 148 shows some grid data in a spreadsheet:

	A	B	C	D	E
1	Name	pets			
2	Note	test data			
3	Purpose	illustrate data transfer			
4	Range	1	9		
5	Grid	dog	cat	rabbit	
6	remote	7	9	3	friendly
7	clean	3	1	7	messy
8					
9	Range	-3	3		
10	Grid				
11		2	-2	2	
12	quiet	2	-2	-2	noisy
13					

Figure 148: Two grids entered in a spreadsheet

Row 1 is optional and specifies the name for the first grid to be ***pets***. If this row is omitted the name is set to be ***Grid-1***.

Row 2 is optional and specifies the note for the first grid to be ***test data***.

Row 3 is optional and specifies the purpose for the first grid to be ***illustrate data transfer***.

Row 4 is optional and sets the scale range to be 1 to 9. If this row is omitted the range is set by default to be 1 to 5.

Row 5 is required and specifies that a grid dataset follows, with 3 elements, ***dog***, ***cat*** and ***rabbit***. The count of the number of elements in the grid following is obtained from the number of entries on this line.

Rows 6 and 7 specify two constructs and the ratings for each of the elements on these constructs.

Row 8 is blank and terminates the building of the first grid. Further data will belong to further grids.

Row 9 changes the scale to be from -3 to +3. If this line was not present the scale would remain as 1 to 9.

Because the name, note and purpose keywords have been omitted, the name will be generated as *Grid-2* and the note and purpose fields will be the same as those already specified.

Row 10 specifies that a grid dataset follows. Since no new elements are specified, the three elements already specified continue to apply.

Row 11 specifies the first construct of a new grid. Since no pole names are specified, the corresponding construct in the previous grid applies, that is, ***remote—friendly***.

Row 12 specifies another construct.

Row 13 is blank and end of the spreadsheet. It terminates building the second grid.

If this spreadsheet data is dragged to the ***Rep 5 Manager*** window, or is saved using the ***Text (tab delimited)*** option and opened in Rep Plus, two RepGrid windows are created, one for each of the grids in the spreadsheet file. These are shown below together with a Display of each grid.

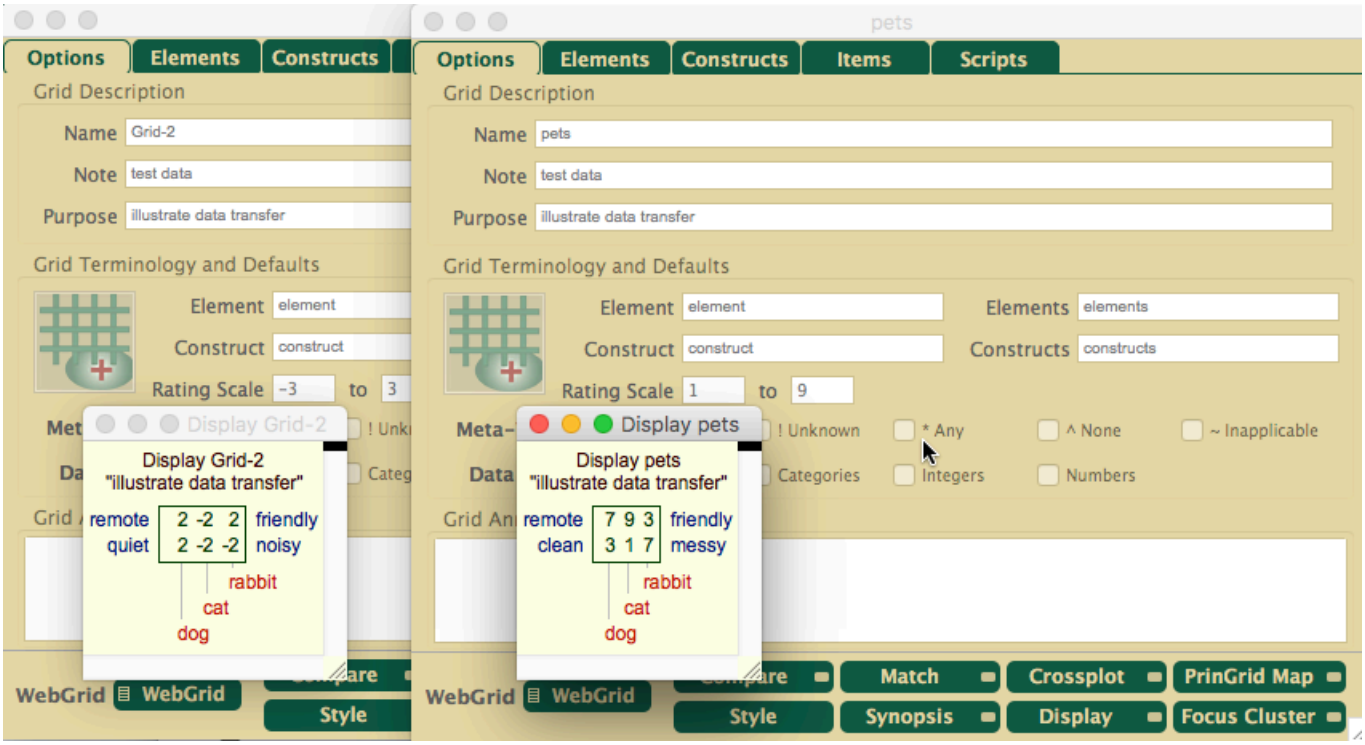


Figure 149: RepGrid windows created from spreadsheet tab-delimited data

10 Appendix: Elicit Scripts

The sample scripts for elicitation are in English and emulate Shaw's (1980) PEGASUS program. Facilitators may wish to edit them to make them more appropriate to particular communities and to translate them into other languages. The scripts are listed here to illustrate what is involved.

For many purposes there will be no need to change the programming and modifying the quoted text which constitutes the dialog will be sufficient. The programming language is fully documented in the ***RepScript Manual*** and editable in the Rep Plus ***Script Editor*** which automatically lays it out and colours it to make the syntactic form more apparent.

10.1 Script: Elicit Grid.repscript

The **Elicit Grid** script sets up the text styles, shows the welcome message, and passes control to the *Main* script at the *Initial* entry point.

```
// PEGASUS-style (Shaw 1980) conversational elicitation with feedback from matches

// set up standard text styles
StyleAdd("Text",1,0,12,RGB(0,100,0),"") // 12 pt green
StyleAdd("Center",2,0,12,RGB(0,100,0),"") // 12 pt green centred
StyleAdd("CenterBold",2,1,12,RGB(0,100,0),"") // 12 pt green centred bold
StyleAdd("HeadBig",2,1,18,RGB(255,0,0),"") // 18 pt red centred bold
StyleAdd("Head",2,1,14,RGB(255,0,0),"") // 14 pt red centred bold
StyleAdd("CenterBoldBig",2,1,14,RGB(0,100,0),"") // 14 pt green centred bold

// initialize appearance
SetBackColor(RGB(240,240,255))
StyleSet("Text")

// Commence flow of interactive dialog
Select Case vGet
Case ""
  UndoSave("Elicit Grid")
  SetMessage("")
  hSet("true","Initial") // set a flag to tell other code we are in initial phase
  if gGet("Context")="" or gGet("Name")="" or gGetI("ne")<6 then
    TextClear
    Output("PEGASUS Plus"+EOL+EOL,"HeadBig")
    Output("Program Elicits Grids and Sorts Using Similarities"+EOL+EOL,"Head")
    Output("This is a program to elicit a Repertory Grid. ")
    Output("A repertory grid is a technique devized by George Kelly to help you explore the
        dimensions of your thinking. ")
    Output("Please read carefully everything that is shown, and make sure you understand what you
         have to do. ")
    Output(EOL+EOL)
  end if
  ScriptFlow("/Elicit/Main/Initial") // normal entry
Case "*" // re-entry to script window when grid edited elsewhere
  TextClear
  Output("Continue Repertory Grid Elicitation"+EOL+EOL,"Head")
  ScriptFlow("/Elicit/Main/Initial")
End Select
```

10.2 Script: Elicit/Main.repscript

The **Main** script controls the flow of elicitation, passing control to other scripts which take action and then return to the Main script.

```
dim ne,nc,status As integer, te,tes,ten,tc,tcs,tcn As string
ne=gGetI("ne")
nc=gGetI("nc")
status=gGetI("Status")
te=gGet("E")
tes=gGet("Es")
if ne=1 then ten=te else ten=tes
tc=gGet("C")
tcs=gGet("Cs")
if nc=1 then tcn=tc else tcn=tcs

sub ClearMatches()
  dim i,n As integer
  n=gGetI("MatchC","80","0","M","MV") //get rid of construct matches in existing grid
  for i=0 to n-1
    hSet(vGetI(i,"MV"),vGet(i,"M"),"MatchC")
  next
  n=gGetI("MatchE","80","0","M","MV") //get rid of element matches in existing grid
  for i=0 to n-1
    hSet(vGetI(i,"MV"),vGet(i,"M"),"MatchE")
  next
end sub

Select case ScriptState

Case "Initial" // initial entry point
// main ScriptFlow of control
 hEmpty("MatchE")
 hEmpty("MatchC")
 ClearMatches
 ScriptFlow("CheckName","Context")

Case "Context"
  ScriptFlow("CheckContext","InitialElements")

Case "InitialElements"
  SetMessage(gGet("Name")+" is considering """+gGet("Context")+""""")
  ScriptFlow("GetInitialElements","CheckExchange")

Case "CheckExchange"
  if status=3 or status=5 then // exchange or constructs
    ScriptFlow("Constructs/RateAnyOpen","Options") // don't test for matches
  else
    ScriptFlow("Triads","CheckMatches") // new grid or elements
  end if

Case "CheckMatches"
  ScriptFlow("Match/DoMatch","Options")

Case "Options"
  hRemove("Initial") // unset flag
  TextClear
  Output("Select an option"+EOL+EOL,"Head")
  Output("Your grid has "+str(ne)+" "+ten+" and "+str(nc)+" "+tcn+", and you may chose what to do
       next. ")
  Output("Click on one of the options below to select it."+EOL+EOL)
  if nc<gGetI("LimitC") and ne>2 then Output("T\Elicit another "+tc+" from a triad"+EOL,"
      CenterBold")
  Output("E\List and edit your "+tes+EOL,"CenterBold")
```

```
  Output("C\List and edit your "+tcs+EOL,"CenterBold")
  Output("X\Finish now"+EOL,"CenterBold")
  ScriptWait("OptionsClick",kClick)
Case "OptionsClick"
  select case InCode
  case "T"
    ScriptFlow("AddConstruct/DoTriadChosen","CheckMatches")
  case "E"
    ScriptFlow("Elements/List","CheckMatches")
  case "C"
    ScriptFlow("Constructs/List","CheckMatches")
  case "X"
    ScriptFlow("Constructs/RateAnyOpen","Halt") // Check for unrated given constructs
  case "$"
    ScriptFlow("Options")
  else
    Alert("Not editable","The item in which you clicked cannot be edited")
    ScriptWait("OptionsClick",kClick)
  end select

Case "Halt"
  TextClear
  Output(EOL+"User has finished"+EOL,"Head")
  Halt("")

Case "CheckName"
  if gGet("Name")="" then
    hSet("false","Initial")
    Output("    What is your name or identification: ")
    ScriptWait("InputName",kText)
  else
    ScriptFlow
  end if
Case "InputName"
  gSet("Name",Input)
  Output(EOL+EOL)
  ScriptFlow

Case "CheckContext"
  if gGet("Context")="" then
    hSet("false","Initial")
    Output("State your purpose"+EOL+EOL,"Head")
    Output("You must decide on a purpose for doing the grid and keep this in mind when you chose
        the "+tes+"--")
    Output("the things you are going to think about during the program. These "+tes+" will then
        be used to elicit "+tcs+".")
    Output(EOL+EOL)
    Output("If you make a typing error press the delete key as many times as you want to eraze a
        character, then carry on.")
    Output(EOL+EOL)
    Output("    What is your purpose? ")
    ScriptWait("InputContext",kText)
  else
    ScriptFlow
  end if
Case "InputContext"
  gSet("Context",Input)
  Output(EOL+EOL)
  ScriptFlow

Case "GetInitialElements"
  if ne<6 then
    if hGet("Initial")="false" then TextClear // if we collected name or context
    hSet("false","Initial")
    Output("Name six or more "+tes+EOL+EOL,"Head")
```

```
    Output("You must choose a set of "+tes+" keeping in mind why you want to do this grid. ")
    if tes="Elements" and status<>5 then // only output following text if type of elements has
        not been specified and not a "Constructs" copy
      Output("They could be people, events, pieces of music, pictures, books or what you want, ")
      Output("but whatever you chose they must be of the same type and each must be well known to
          you. ")
      Output("Try to choose specific things. ")
    end if
    Output("Now type each one after each colon. ")
    Output("Do not forget to press return after each. ")
    Output("When you have entered as many "+tes+" as you wish, just press return alone to
        continue")
    Output(EOL+EOL)
    ScriptFlow("GetElements")
  else
    ScriptFlow
  end if

Case "GetElements"
  Output("  What is "+te+" "+str(ne+1)+": ")
  ScriptWait("InputElement",kText)
Case "InputElement"
  Output(EOL)
  if Input="" then
    if ne>=6 or status=5 then // "Constructs" copy allowed to avoid triadic elicitation
      Output(EOL)
      ScriptFlow
    else
      Alert("Not enough "+tes,"You have entered only "+str(ne)+" "+tes+". You need at least 6 to
          elicit a grid.")
      ScriptFlow("GetElements")
    end if
  else
    hEmpty("X")
    hSet(Input,"Name","X")
    gSet("NewE","X")
    ScriptFlow("GetElements")
  end if

Case "Triads"
  hSet("","E") // select random triads
  select case nc-sGetI(gGet("OpenC")) // don't include any given constructs that are totally
      unrated
  case 0
    ScriptFlow("AddConstruct/DoTriad","Triads")
  case 1
    TextClear
    Output("How to think about "+tcs+EOL+EOL,"Head")
    Output("Now you have one "+tc+" you know what to do. ")
    Output("You may think of "+tcs+" as lines along which each of your "+tes+" has a place in
        relation to all the other "+tes+". ")
    Output("Please do not use "+tcs+" which do not apply to all your "+tes+". ")
    Output("An example of this is redhead--blond, as it is impossible to rate a person with black
         hair on this "+tc+". ")
    Output("One pole must be in some sense what the other is not, and they must divide your "+tes
        +" into two approximately equal groups, ")
    Output("so please try to avoid "+tcs+" where nearly all the "+tes+" are at one end. ")
    Output("An example might be ""extremely tall--not extremely tall"""+EOL+EOL)
    ScriptFlow("AddConstruct/DoTriadNoClear","Triads")
  case 2
    ScriptFlow("AddConstruct/DoTriad","Triads")
  case 3
    ScriptFlow("AddConstruct/DoTriad","Triads")
  else
    ScriptFlow
```

```
  end select

Case "Compile"
  ScriptFlow

else
  Halt("Flow error--no label "+ScriptState+EOL)

end select
```

10.3 Script: Elicit/Elements.repscript

The **Elements** script lists, adds and rates elements, and allows members of a triad to be chosen.

```
dim ne,nc,status As integer, te,tes,ten,tc,tcs,tcn As string
ne=gGetI("ne")
nc=gGetI("nc")
status=gGetI("Status")
te=gGet("E")
tes=gGet("Es")
if ne=1 then ten=te else ten=tes
tc=gGet("C")
tcs=gGet("Cs")
if nc=1 then tcn=tc else tcn=tcs

dim i,j,jj,n As integer, lhp,rhp,v,e,e1,e2,s,t As string, b As Boolean

Select case ScriptState

Case "List"
  TextClear
  Output("List of "+tes+EOL+EOL,"Head")
  for j=0 to ne-1
    n=gGetI("OpenE",str(j))
    if n=0 then s="" else s=" ("+str(n)+" "+tcs+" not rated)"
    Output("E"+str(j)+"\"+"    "+gGet("E",str(j))+s+EOL)
  next
  Output(EOL)
  if ne<gGetI("LimitE") then Output("A\Click here to add another "+te+"."+EOL,"CenterBold")
  Output("$\Click in an item to edit it, or here if you have finished editing."+EOL,"CenterBold")
  ScriptWait("ListClick",kClick)
Case "ListClick"
  select case Left(incode,1)
  case "$"
    ScriptFlow
  case "A"
    ScriptFlow("AddElement","List")
  case "E"
    hSet(Right(incode,len(incode)-1),"E")
    ScriptFlow("EditElement/DoEditElement","List")
  else
    Alert("Not editable","The item in which you clicked cannot be edited")
    ScriptWait("ListClick",kClick)
  end select

Case "AddElement"
  TextClear
  Output("Add another "+te+EOL+EOL,"Head")
  Output("  What is "+te+" "+str(ne+1)+": ")
  ScriptWait("AddElementIn",kText)
Case "AddElementIn"
  if Input="" then
```

```
    Output(EOL)
    ScriptFlow
  else
    hSet(ne,"E")
    hEmpty("X")
    hSet(Input,"Name","X")
    gSet("NewE","X")
    Output(EOL+EOL+"Give your "+te+", "+Input+", a rating on each "+tc)
    Output(" by entering a number or clicking to use a popup menu."+EOL+EOL)
    ScriptFlow("CRatingLoop")
  end if

Case "CRating" // Entry after an element match
  TextClear
  Output("Rate your "+te+" on the "+tcs+EOL+EOL,"Head")
  Output("Give your "+te+", "+gGet("E",hGet("E"))+", a rating on each "+tc)
  Output(" by entering a number or clicking to use a popup menu."+EOL+EOL)
  ScriptFlow("CRatingLoop")
Case "CRatingLoop"
  j=hGetI("E")
  vCountSet(0,"Sort") // set up a vector of unrated constructs
  for i=0 to nc-1
    v=gGet("V",str(i),str(j))
    if v<>"?" then Output("    "+gGet("C",str(i))+": "+v+EOL) else vPush(i,"Sort")
  next
  ScriptFlow("CRatingLoopOpen")
Case "CRatingLoopOpen"
  if vOK("Sort") then
    ScriptFlow("CRatingLoopOut")
  else
    Output(EOL)
    ScriptFlow("EditElement/DoEditElement")
  end if
Case "CRatingLoopOut"
  i=vPopI("Sort")
  hSet(i,"C")
  Output("    "+gGet("C",str(i))+": ")
  ScriptWait("CRatingLoopIn",kCMenu)
Case "CRatingLoopIn"
  Output(EOL)
  i=hGetI("C")
  j=hGetI("E")
  s=NthField(Input," ",1)
  gSet("V",str(i),str(j),s)
  if gGet("V",str(i),str(j))<>s then
    Call gGet("C",str(i),"X")
    s=hGet("Range","X")
    Alert ("Value not appropriate","Enter a value in the range "+sGet(s,1)+" to "+sGet(s,2))
    ScriptFlow("CRatingLoopOut")
  else
    ScriptFlow("CRatingLoopOpen")
  end if

Case "ChooseTriad"
  hRemove("E")
  ScriptFlow("ReListTriad")
Case "ReListTriad"
  TextClear
  Output("Choose a triad of "+tes+EOL+EOL,"Head")
  Output("Press the return key to have the program select a triad. ")
  Output("Or, you may select up to three "+tes+" yourself. If you select less than three the
      others will be selected at random."+EOL+EOL)
  s="Triad: "
  e=hGet("E")
  jj=sGetI(e)
```

```
  for j=1 to jj
    t=gGet("E",sGet(e,j))
    if j=1 then s=s+t else s=s+", "+t
  next
  Output(s+EOL+EOL)
  e=hGet("E")
  for j=0 to ne-1
    if sFind(e,str(j))=0 then
      Output("E"+str(j)+"\"+"    "+gGet("E",str(j))+EOL)
    end if
  next
  Output(EOL)
  Output("$\Click in your chosen "+te+" to select it, or here if you have finished selecting."+
      EOL,"CenterBold")
  ScriptWait("TriadClick",kClick)
Case "TriadClick"
  select case Left(incode,1)
  case "$"
    ScriptFlow
  case "E"
    e1=hGet("E")
    e2=Right(incode,len(incode)-1)
    j=sGetI(e1)
    if j=0 then hSet(e2,"E") else hSet(sMake(e1,e2),"E")
    if j>1 then
      ScriptFlow
    else
      ScriptFlow("ReListTriad")
    end if
  else
    Alert("Not editable","The item in which you clicked cannot be edited")
    ScriptWait("TriadClick",kClick)
  end select

Case "Compile"
  ScriptFlow

else
  Halt("Flow error--no label "+ScriptState+EOL)

end select
```

10.4 Script: Elicit/EditElement.repscript

The **EditElement** script supports editing an element name and ratings.

```
dim ne,nc,status As integer, te,tes,ten,tc,tcs,tcn As string
ne=gGetI("ne")
nc=gGetI("nc")
status=gGetI("Status")
te=gGet("E")
tes=gGet("Es")
if ne=1 then ten=te else ten=tes
tc=gGet("C")
tcs=gGet("Cs")
if nc=1 then tcn=tc else tcn=tcs

dim n,i,j As integer, s,v As string, ok As Boolean

Select case ScriptState

Case "DoEditElement"
```

```
  TextClear
  j=hGetI("E")
  s=gGet("E",str(j))
  Output("Edit "+te+" """+s+"""+EOL+EOL,"Head")
  OutPut("E\    Name of "+te+": "+s+EOL+EOL)
  for i=0 to nc-1
    Output("C"+str(i)+"\    "+gGet("C",str(i))+": "+gGet("V",str(i),hGet("E"))+EOL)
  next
  Output(EOL)
  if hGet("Initial")="" then
  Output("D\Click here to delete the "+te+EOL,"CenterBold")
  end if
  Output("$\Click on the "+te+" name or the "+tc+" to edit it, or here when you have finished
      editing"+EOL,"CenterBold")
  ScriptWait("EditClick",kClick)
Case "EditClick"
  s=Incode
  Select Case Left(s,1)
  case "D"
    ScriptFlow("EDelete")
  case "E"
    ScriptFlow("EOut")
  case "C"
    hSet(Right(Incode,len(Incode)-1),"C")
    ScriptFlow("CRatingOut")
  case "$"
    ScriptFlow // return as specified by caller
  else
    Alert("Not editable","The item in which you clicked cannot be edited")
    ScriptWait("EditClick",kClick)
  end select

Case "EDelete"
   j=hGetI("E")
   s=gGet("E",str(j))
   ok=Confirm("OK to delete "+s+"?","Deleting the "+te+", "+s+", is irreversible. Click on OK if
       you really want to delete it.")
   if ok then
     gSet("RemoveE",str(j))
     ScriptFlow
   else
     ScriptFlow("DoEditElement")
   end if

Case "EOut"
  TextClear
  s=gGet("E",hGet("E"))
  Output("Edit name of "+te+" """+s+"""+EOL+EOL,"Head")
  Output("    Name of "+te+": ")
  ScriptWait("EIn",kText)
  Output(s)
Case "EIn"
  if Input="" then
    ScriptFlow("DoEditElement")
  else
    Output(EOL)
    hSet(Input,"Name","X")
    gSet("E",hGet("E"),"X")
    ScriptFlow("DoEditElement")
end if

Case "CRatingOut"
  TextClear
  i=hGetI("C")
  j=hGetI("E")
```

```
  Output("Edit rating for "+te+" """+gGet("E",str(j))+"""+EOL+EOL,"Head")
  Output("    "+gGet("C",str(i))+": ")
  ScriptWait("CRatingIn",kCMenu)
  OutputSelect(gGet("V",str(i),str(j)))
Case "CRatingIn"
  Output(EOL)
  if Input="" then
    ScriptFlow("DoEditElement")
  else
    i=hGetI("C")
    j=hGetI("E")
    s=NthField(Input," ",1)
    gSet("V",str(i),str(j),s)
    Call gGet("C",str(i),"X")
    if gGet("V",str(i),str(j))<>s then
      s=hGet("Range","X")
      Alert ("Value not appropriate","Enter a value in the range "+sGet(s,1)+" to "+sGet(s,2))
      ScriptFlow("CRatingOut")
    else
      ScriptFlow("DoEditElement")
    end if
  end if

Case "Compile"
  ScriptFlow

else
  Halt("Flow error--no label "+ScriptState+EOL)

end select
```

10.5 Script: Elicit/Constructs.repscript

The **Constructs** script lists and edits constructs.

```
dim ne,nc,status As integer, te,tes,ten,tc,tcs,tcn As string
ne=gGetI("ne")
nc=gGetI("nc")
status=gGetI("Status")
te=gGet("E")
tes=gGet("Es")
if ne=1 then ten=te else ten=tes
tc=gGet("C")
tcs=gGet("Cs")
if nc=1 then tcn=tc else tcn=tcs

dim i,k,n As integer, s,c As string

Select case ScriptState

Case "List"
  TextClear
  Output("List of "+tcs+EOL+EOL,"Head")
  for i=0 to nc-1
    n=gGetI("OpenC",str(i))
    if n=0 then s="" else s=" ("+str(n)+" "+tes+" not rated)"
    Output("C"+str(i)+"\"+"    "+gGet("C",str(i))+s+EOL)
  next
  Output(EOL)
  if nc<gGetI("LimitC") then Output("A\Click here to add another "+tc+"."+EOL,"CenterBold")
  Output("$\Click in an item to edit it, or here if you have finished editing."+EOL,"CenterBold")
  ScriptWait("ListClick",kClick)
```

```
Case "ListClick"
  select case Left(incode,1)
  case "$"
    ScriptFlow
  case "A"
    hRemove("E") // not from a triad or pair so make sure no elements
    ScriptFlow("AddConstruct/AddConstruct","List")
  case "C"
    hSet(Right(incode,len(incode)-1),"C")
    ScriptFlow("EditConstruct/DoEditConstruct","List")
  else
    Alert("Not editable","The item in which you clicked cannot be edited")
    ScriptWait("ListClick",kClick)
  end select

Case "RateAnyOpen"
  if gGetI("Open")>0 then
    hSet("true","RateOpen") // stop option to delete
    ScriptFlow("RateOpen","RateAnyOpen")
  else
    hRemove("RateOpen")
    ScriptFlow
  end if

Case "RateOpen"
  for i=0 to nc-1
    k=gGetI("OpenC",str(i))
    if k>n then
      hSet(i,"C")
      n=k
    end if
  next
  if n>0 then
    ScriptFlow("AddConstruct/RateConstruct")
  else
    ScriptFlow
  end if

Case "Compile"
  ScriptFlow

else
  Halt("Flow error--no label "+ScriptState+EOL)

end select
```

10.6 Script: Elicit/AddConstruct.repscript

The **AddConstruct** script supports adding constructs, directly, from triads and matches.

```
dim ne,nc,status As integer, te,tes,ten,tc,tcs,tcn As string
ne=gGetI("ne")
nc=gGetI("nc")
status=gGetI("Status")
te=gGet("E")
tes=gGet("Es")
if ne=1 then ten=te else ten=tes
tc=gGet("C")
tcs=gGet("Cs")
if nc=1 then tcn=tc else tcn=tcs

function MakeTriad() As Boolean
```

```
  // enter with 0 or more prescribed elements in E
  // select other elements at random to make the number up to 3 and return in E
  // return false if not enough elements to make a triad
  dim e,s As string
  if ne<3 then return false
  do
    e=hGet("E")
    if sGetI(e)>=3 then return true
    s=str(GetRandom(0,ne-1))
    if sFind(e,s)=0 then
      if e="" then e=s else e=e+TAB+s
      hSet(e,"E")
    end if
  loop
end function

// main program
dim i,j,n As integer, lhp,rhp,v,s,e,b As string

Select case ScriptState

Case "RateConstruct"
  TextClear
  i=hGetI("C")
  Output("Rate "+tes+" on """+gGet("C")+"""+EOL,"Head")
  ScriptFlow("ERating")

Case "AddConstruct"
  TextClear
  Output("Add another "+tC+EOL+EOL,"Head")
  ScriptFlow("LHP")

Case "DoTriadChosen"
  ScriptFlow("Elements/ChooseTriad","DoTriad")

Case "DoTriad"
  TextClear
  Output("Elicit "+tC+" from a triad"+EOL+EOL,"Head")
  ScriptFlow("Triad")

Case "DoTriadNoClear"
  ScriptFlow("Triad")

Case "Triad"
  if MakeTriad then
    e=hGet("E")
    Output("Can you choose two of this triad of "+tes+" which are in some way alike and different
         from the other one?"+EOL+EOL)
    for i=1 to 3
      s=sGet(e,i)
      Output(s+"\"+gGet("E",s)+EOL,"CenterBoldBig")
    next
    Output(EOL)
    Output("$\Click in the "+te+" which is different, or here if you cannot do this."+EOL,"
        CenterBold")
    ScriptWait("TriadClicked",kClick)
  else
    Halt("Not enough elements for triad"+EOL)
  end if
Case "TriadClicked"
  If InCode="$" then
    Output(EOL)
    ScriptFlow
  else
    e=hGet("E")
```

```
    if sGet(e,2)=Incode then // make sure the one clicked is first
      hSet(sMake(sGet(e,2),sGet(e,1),sGet(e,3)),"E")
    elseif sGet(e,3)=Incode then
      hSet(sMake(sGet(e,3),sGet(e,1),sGet(e,2)),"E")
    end if
    TextClear
    Output("Name the poles of your "+tC+EOL+EOL,"Head")
    Output("Now I want you to think what you have in mind when you separate the pair from the
        other one. ")
    Output("Just type one or two words for each pole to remind you what you are thinking or
        feeling when you use this "+tC+".")
    Output(EOL+EOL)
    Output("X\"+"Or click here if you cannot do this"+EOL+EOL,"CenterBold")
    ScriptFlow("LHP")
  end if

Case "LHP"
  // enter with 0 to 3 element numbers stored in "E"
  e=hGet("E")
  s=sGet(e,2)
  if s<>"" then
    s=gGet("E",s)
    e=sGet(e,3)
    if e<>"" then s=s+", "+gGet("E",e)
    s=" {"+s+"}"
  end if
  Output("    Left pole rated "+gGet("MinR")+s+": ")
  ScriptWait("LHPIn",kText)
Case "LHPIn"
  Output(EOL)
  if InCode="X" or Input="" then
    Output(EOL)
    ScriptFlow // return, no construct added
  else
    hSet(Input,"L") // hold LHP in "L"
    ScriptFlow("RHP")
  end if
Case "RHP"
  s=sGet(hGet("E"),1)
  if s<>"" then s=" {"+gGet("E",s)+"}"
  Output("    Right pole rated "+gGet("MaxR")+s+": ")
  ScriptWait("RHPIn",kText)
Case "RHPIn"
  Output(EOL)
  if InCode="X" or Input="" then
    Output(EOL)
    ScriptFlow // return, no construct added
  else
    i=nc
    hSet(i,"C") // hold construct number in "C"
    hEmpty("X")
    hSet("R","Type","X")
    hSet(hGet("L"),"LHP","X")
    hSet(Input,"RHP","X")
    gSet("NewC","X")
    e=hGet("E")
    b="true"
    for n=1 to 3
      s=sGet(e,n)
      if s<>"" then gSet("EndV",str(i),s,b)
       b="false"
    next
    ScriptFlow("ERating")
  end if
```

```
Case "ERating"
  TextClear
  i=hGetI("C")
  Output("Rate the "+tes+" on your "+tC+" """+gGet("C",str(i))+"""+EOL+EOL,"Head")
  Output("According to how you feel about them, please assign to each of the following "+tes+" a
      rating from ")
  Call gGet("C",str(i),"X")
  s=hGet("Range","X")
  Output(sGet(s,1)+" ("+hGet("LHP","X")+") to "+sGet(s,2)+" ("+hGet("RHP","X")+")")
  Output(" by entering a number or clicking to use a popup menu.")
  Output(EOL+EOL)
  Call gGet("SortV",str(i),"Sort","false")
  ScriptFlow("ERatingLoop")
Case "ERatingLoop"
  if vOK("Sort") then
    j=vExtractI("Sort")
    hSet(j,"E")
    v=gGet("V",str(hGetI("C")),str(j))
    if v<>"?" then
      Output("    "+gGet("E",str(j))+" "+v+EOL) // show already set values
      ScriptFlow("ERatingLoop")
    else
      ScriptFlow("ERatingLoopOut")
    end if
  else
    Output(EOL)
    ScriptFlow("EditConstruct/DoEditConstruct") // offer option to edit
  end if
Case "ERatingLoopOut"
  Output("    "+gGet("E",hGet("E"))+": ")
  ScriptWait("ERatingLoopIn",kCMenu)
Case "ERatingLoopIn"
  Output(EOL)
  i=hGetI("C")
  j=hGetI("E")
  s=NthField(Input," ",1)
  gSet("V",str(i),str(j),s)
  if gGet("V",str(i),str(j))<>s then
    s=hGet("Range","X")
    Alert ("Value not appropriate","Enter a value in the range "+sGet(s,1)+" to "+sGet(s,2))
    ScriptFlow("ERatingLoopOut")
  else
    ScriptFlow("ERatingLoop")
  end if

Case "Compile"
  ScriptFlow

else
  Halt("Flow error--no label "+ScriptState+EOL)

end select
```

10.7 Script: Elicit/Match.repscript

The **Match** script finds and displays element and construct matches, and gives the option to add constructs and elements to reduce the match.

```
dim ne,nc,status As integer, te,tes,ten,tc,tcs,tcn As string
ne=gGetI("ne")
nc=gGetI("nc")
status=gGetI("Status")
```

```
te=gGet("E")
tes=gGet("Es")
if ne=1 then ten=te else ten=tes
tc=gGet("C")
tcs=gGet("Cs")
if nc=1 then tcn=tc else tcn=tcs

dim i,j,n,i1,i2,j1,j2,v As integer, match As double, e,e1,e2,s As string, b As Boolean

Select case ScriptState

Case "DoMatch" // find any matches not already checked
  b=false
  if ne>3 and ne<gGetI("LimitE") then // test for C matches
    n=gGetI("MatchC","80","0","M","MV")
  //print str(n)+EOL
  //print vDump("M")+EOL
  //print vDump("MV")+EOL
  //print hDump("MatchC")+EOL
    for i=0 to n-1
      s=vGet(i,"M") // get match pair
      if not hCheck(s,"MatchC") then // new match
        i1=sGetI(s,1) // get first construct
        hSet(i1,"C1") // hold number in C1
        Call gGet("C",str(i1),"X1") // hold specification in X1
        i2=sGetI(s,2) // get second construct
        hSet(i2,"C2") // hold number in C2
        Call gGet("C",str(i2),"X2") // hold specification in X2
        v=vGetI(i,"MV") // get match value
        hSet(v,"M") // hold match value in M
        hSet(v,s,"MatchC") // record match pair in MatchC
        b=true // found a new construct match
        ScriptFlow("ShowCMatch","DoMatch")
       exit
      end if
    next
  end if
  if not b and nc>3 and nc<gGetI("LimitC") then // test for E matches
    n=gGetI("MatchE","80","0","M","MV")
    for i=0 to n-1
      s=vGet(i,"M") // get match pair
      if not hCheck(s,"MatchE") then // new match
        hSet(s,"E") // hold matching E numbers
        v=vGetI(i,"MV") // get match value
        hSet(v,"M") // hold match value in M
        hSet(v,s,"MatchE") // record match pair in MatchE
        b=true // found a new element match
        ScriptFlow("ShowEMatch","DoMatch")
       exit
      end if
    next
  end if
  if not b then ScriptFlow // no matches

Case "ShowCMatch"
  TextClear
  Output("Break "+tc+" match"+EOL+EOL,"Head")
  Output("The two "+tcs+" you called"+EOL)
  Output(hGet("Identifier","X1")+EOL+hGet("Identifier","X2")+EOL,"Center")
  Output("are matched at the "+hGet("M")+"% level.")
  Output(" This means that most of the time you are saying "+hGet("LHP","X1")+" you are also
      saying "+hGet("RHP","X2"))
  Output(", and most of the time you are saying "+hGet("RHP","X1")+" you are also saying "+hGet("
      LHP","X2")+"."+EOL+EOL)
  Output("Think of another "+te+" which is either:-"+EOL+EOL)
```

```
  Output("1\    "+hGet("LHP","X1")+" and "+hGet("RHP","X2")+EOL,"CenterBold")
  Output("or"+EOL,"Center")
  Output("2\    "+hGet("LHP","X2")+" and "+hGet("RHP","X1")+EOL+EOL,"CenterBold")
  Output("$\Click on the appropriate combination, or here if you cannot do this"+EOL,"CenterBold
      ")
  ScriptWait("BreakCMatch",kClick)
Case "BreakCMatch"
  Select Case Incode
  case "1"
    TextClear
    Output("Add a "+te+" to reduce a match between "+tcs+EOL+EOL,"Head")
    Output("X\"+"Or click here if you cannot do this"+EOL+EOL,"CenterBold")
    Output("  What is the "+te+" that is """+hGet("LHP","X1")+""" and """+hGet("RHP","X2")+""":
        ")
    ScriptWait("InElement",kText)
  case "2"
    TextClear
    Output("Add a "+te+" to reduce a match between "+tcs+EOL+EOL,"Head")
    Output("X\"+"Or click here if you cannot do this"+EOL+EOL,"CenterBold")
    Output("  What is the "+te+" that is """+hGet("RHP","X1")+""" and """+hGet("LHP","X2")+""":
        ")
    ScriptWait("InElement",kText)
  case "$"
    ScriptFlow // return as specified by caller
  else
    Alert("Not understood","The item in which you clicked is not an option")
    ScriptWait("BreakCMatch",kClick)
  end select
Case "InElement"
  if InCode="X" or Input="" then
    Output(EOL)
    ScriptFlow
  else
    j=ne
    hSet(j,"E")
    hEmpty("X")
    hSet(Input,"Name","X")
    gSet("NewE","X")
    b=InCode="1"
    gSet("EndV",hGet("C1"),str(j),BooStr(not b))
    gSet("EndV",hGet("C2"),str(j),BooStr(b))
    ScriptFlow("Elements/CRating")
  end if

Case "ShowEMatch"
  s=hGet("E")
  e2=gGet("E",sGet(s,1))
  e1=gGet("E",sGet(s,2))
  TextClear
  Output("Break "+te+" match"+EOL+EOL,"Head")
  Output("The two "+tes+" "+e1+" and "+e2+" are matched at the "+hGet("M")+"% level. ")
  Output("This means that so far you have not distinguished between them."+EOL+EOL)
  Output("Think of another "+tc+" which separates "+e1+" from "+e2+"."+EOL+EOL)
  Output("X\"+"Or click here if you cannot do this"+EOL+EOL,"CenterBold")
  ScriptFlow("AddConstruct/LHP")

Case "Compile"
  ScriptFlow

else
  Halt("Flow error--no label "+ScriptState+EOL)

end select
```

11 Bibliography

Adler, J. E. and Rips, L. J. (2008). *Reasoning: Studies of Human Inference and Its Foundations*. Cambridge University Press, Cambridge.

Algom, D. and Fitousi, D. (2016). Half a century of research on garner interference and the separability–integrality distinction. *Psychological Bulletin*, 142(12):1352–1383.

Attneave, F. (1950). Dimensions of similarity. *American Journal of Psychology*, 43(4):516–556.

Balme, D. M. (1987). Aristotle's use of division and differentiae. In Gotthelf, A. and Lennox, J. G., editors, *Philosophical issues in Aristotle's biology*. Cambridge University Press, Cambridge.

Bellman, L. (2012). *Auktoriserade fastighetsvärderares syn på värdering: tankemönster om kommersiella fastigheter*. Licentiate thesis.

Boose, J. H. and Gaines, B. R. (1988). *Knowledge Acquisition Tools for Expert Systems*. Academic Press, London.

Buchanan, B. G. and Shortliffe, E. H. (1984). *Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA.

Caputi, P. (2011). *Personal Construct Methodology*. Wiley, Chichester, UK.

Cendrowska, J. (1987). An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370.

Corbridge, C., Rugg, G., Major, N. P., Shadbolt, N. R., and Burton, A. M. (1994). Laddering: technique and tool use in knowledge acquisition. *Knowledge Acquisition*, 6(3):315–341.

Davies, S. (1991). *Definitions of Art*. Cornell University Press, Ithaca, NY.

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge University Press, Cambridge.

Dey, S. and Lee, S.-W. (2017). Reassure: Requirements elicitation for adaptive socio-technical systems using repertory grid. *Information and Software Technology*, 87:160–179.

Efron, B. and Tibshirani, R. (1993). *An introduction to the bootstrap*. Chapman & Hall, New York.

Emmons, O. (1939). *Linearity In Factor Analysis*. PhD thesis.

Evans, J. S. B. T., Newstead, S. E., and Byrne, R. M. J. (1993). *Human Reasoning: The Psychology of Deduction*. Lawrence Erlbaum, Hove.

Feixas, G. and Cornejo, J. M. (1996). *Manual de la técnica de rejilla mediante el programa RECORD ver. 2.0*. Paidós, Barcelona.

Fortune, S. (1987). A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174.

Fransella, F., Bell, R. C., and Bannister, D. (2004). *A Manual for Repertory Grid Technique*. Wiley, Chichester, UK.

Fromm, M. (2004). *The Repertory Grid Interview*. Waxman, Munster.

Frontier, S. (1976). Étude de la décroissance des valeurs propres dans une analyse en composantes principales: Comparaison avec le moddèle du bâton brisé. *Journal of Experimental Marine Biology and Ecology*, 25(1):67–75.

Gaines, B. R. (1995). Porting interactive applications to the web. In *4th International World Wide Web Conference Tutorial Notes*, pages 199–217. O'Reilly, Sebastopol, CA.

Gaines, B. R. (2015). Universal logic as a science of patterns. In Koslow, A. and Buchsbaum, A., editors, *The Road to Universal Logic: Festschrift for 50th Birthday of Jean-Yves Béziau*, pages 145–189. Birkhäuser, Basel.

Gaines, B. R., Chen, L. L.-J., and Shaw, M. L. G. (1997). Modeling the human factors of scholarly communities supported through the Internet and World Wide Web. *Journal American Society Information Science*, 48(11):987–1003.

Gaines, B. R. and Shaw, M. L. G. (1993a). Basing knowledge acquisition tools in personal construct psychology. *Knowledge Engineering Review*, 8(1):49–85.

Gaines, B. R. and Shaw, M. L. G. (1993b). Eliciting knowledge and transferring it effectively to a knowledge-based systems. *IEEE Transactions on Knowledge and Data Engineering*, 5(1):4–14.

Gaines, B. R. and Shaw, M. L. G. (1996). Webgrid: Knowledge modeling and inference through the World Wide Web. In Gaines, B. and Musen, M., editors, *Proceedings of Tenth Knowledge Acquisition Workshop*, pages 65–1–65–14.

Gaines, B. R. and Shaw, M. L. G. (1997). Knowledge acquisition, modeling and inference through the World Wide Web. *International Journal of Human-Computer Studies*, 46(6):729–759.

Gaines, B. R. and Shaw, M. L. G. (1998). Developing for web integration in Sisyphus-IV: WebGrid-II experience. In Gaines, B. and Musen, M., editors, *Proceedings of Eleventh Knowledge Acquisition Workshop*.

Gaines, B. R. and Shaw, M. L. G. (2012). Computer aided constructivism. In Caputi, P., Viney, L. L., Walker, B. M., and Crittenden, N., editors, *Constructivist Methods*, pages 183–222. Wiley, New York.

Gärdenfors, P. (2000). *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge, MA.

Gaut, B. (2000). "Art" as a cluster concept. In Carroll, N., editor, *Theories of Art Today*, pages 25–44. University of Wisconsin Press, Madison, WI.

Gill, M. L. (2010). Division and definition in plato's sophist and statesman. In Charles, D., editor, *Definition in Greek philosophy*, pages 172–199. Oxford University Press, Oxford.

Gower, J., Lubbe, S., and Le Roux, N. (2011). *Understanding Biplots*. John Wiley, Chichester, UK.

Gower, J. C. (1966). Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–338.

Gower, J. C. and Hand, D. J. (1995). *Biplots*. Chapman & Hall, London.

Greenacre, M. J. (2007). *Correspondence Analysis in Practice*. Chapman & Hall, London.

Hardman, D. and Macchi, L. (2003). *Thinking: Psychological Perspectives on Reasoning, Judgment, and Decision Making*. Wiley, Hoboken, NJ.

Hayes-Roth, F., Waterman, D. A., and Lenat, D. B. (1983). *Building Expert Systems*. Addison-Wesley, Reading, Massachusetts.

Heckmann, M. and Bell, R. C. (2016). A new development to aid interpretation of hierarchical cluster analysis of repertory grid data,. *Journal of Constructivist Psychology*, 29(4):368–381.

Hennig, C. M., Meilă, M., Murtagh, F., and Rocci, R. (2016). *Handbook of cluster analysis*. Chapman & Hall/CRC, Boca Raton.

Honey, P. (1979). The repertory grid in action: how to use it to conduct an attitude survey. *Industrial and Commercial Training*, 11(11):452–459.

Jackson, D. A. (1993). Stopping rules in principal components analysis: A comparison of heuristical and statistical approaches. *Ecology*, 74(8):2204–2214.

James, W. (1890). *The Principles of Psychology*. Macmillan, London.

Jankowicz, D. (2004). *The Easy Guide to Repertory Grids*. Wiley, Chichester, UK.

Kaipainen, M., Zenker, F., Hautamäki, A., and Gärdenfors, P. (2019). *Conceptual Spaces: Elaborations and Applications*. Synthese Library 405. Springer,, Cham.

Kaldis, B. (2008). The question of platonic division and modern epistemology. In Dillon, J. M., Zovko, M.-E., and Doner, J. F., editors, *Platonism and Forms of Intelligence*. Akademie, Berlin.

Kelly, G. A. (1938). The assumption of an originally homogeneous universe and some of its statistical implications. *Journal of Psychology*, 5:201–208.

Kelly, G. A. (1955). *The Psychology of Personal Constructs*. Norton, New York.

Kelly, G. A. (1969). A mathematical approach to psychology. In Maher, B., editor, *Clinical Psychology and Personality: The Selected Papers of George Kelly*, pages 94–113. Wiley, New York.

Kirkcaldy, B., Pope, M., and Siefen, G. (1993). Sociogrid analysis of a child and adolescent psychiatric clinic. *Social Psychiatry and Psychiatric Epidemiology*, 28(6):296–303.

Kolodner, J. L. (1993). *Case-based Reasoning*. Morgan Kaufmann, San Mateo, CA.

Korenini, B. (2014). Consistent laddering: A new approach to laddering technique. *Journal of Constructivist Psychology*, 27(4):317–328.

Korzybski, A. (1951). The role of language in the perceptual processes. In Blake, R. R. and Ramsey, G. V., editors, *Perception, an Approach to Personality*, pages 170–205. Ronald Press, New York.

Leach, C., Freshwater, K., Aldridge, J., and Sunderland, J. (2001). Analysis of repertory grids in clinical practice. *British Journal of Clinical Psychology*, 40:225–248.

Mireaux, M., Cox, D. N., Cotton, A., and Evans, G. (2007). An adaptation of repertory grid methodology to evaluate australian consumers' perceptions of food products produced by novel technologies. *Food Quality and Preference*, 18(6):834–848.

Nosofsky, R. (1985). Overall similarity and the identification of separable-dimension stimuli: A choice model analysis. *Perception & Psychophysics*, 38(5):415–432.

Okabe, A., Boots, B. N., and Sugihara, K. k. (1992). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, NY.

Peres-Neto, P. R., Jackson, D. A., and Somers, K. M. (2005). How many principal components? stopping rules for determining the number of non-trivial axes revisited. *Computational Statistics & Data Analysis*, 49(4):974–997.

Pope, M. L. and Denicolo, P. M. (2001). *Transformative Professional Practice: Personal Construct Approaches to Education and Research*. Whurr, London.

Rad, A., Wahlberg, O., and Öhman, P. (2013). How lending officers construe assessments of small and medium-sized enterprise loan applications: a repertory grid study. *Journal of Constructivist Psychology*, 26(4):262–279.

Récanati, F. (2013). *Mental Files*. Oxford University Press, Oxford.

Reynolds, T. J. and Gutman, J. (1988). Laddering theory, method, analysis, and interpretation. *Journal of Advertising Research*, 28(1):11–31.

Rosch, E. (1978). Principles of categorization. In Rosch, E. and Lloyd, B. B., editors, *Cognition and Categorization*, pages 27–48. Lawrence Erlbaum, Hillsdale, NY.

Rosch, E. and Lloyd, B. B. (1978). *Cognition and Categorization*. Lawrence Erlbaum, Hillsdale, NY.

Shaw, M. L. G. (1980). *On Becoming a Personal Scientist: Interactive Computer Elicitation of Personal Models of the World*. Academic Press, London.

Shaw, M. L. G. (1981). *Recent Advances in Personal Construct Technology*. Academic Press, London.

Shaw, M. L. G. and Gaines, B. (1992). Kelly's 'Geometry of psychological space' and its significance for psychological modeling. *New Psychologist*, pages 23–31.

Shaw, M. L. G. and Gaines, B. R. (1989). Comparing conceptual structures: consensus, conflict, correspondence and contrast. *Knowledge Acquisition*, 1(4):341–363.

Shaw, M. L. G. and Gaines, B. R. (1996a). Requirements acquisition. *Software Engineering Journal*, 11(3):149–165.

Shaw, M. L. G. and Gaines, B. R. (1996b). Webgrid: Knowledge elicitation and modeling on the web. In Maurer, H., editor, *Proceedings of WebNet96*, pages 425–432. Association for the Advancement of Computing in Education, Charlottesville, VA.

Shaw, M. L. G. and Gaines, B. R. (1998). Webgrid II: Developing hierarchical knowledge structures from flat grids. In Gaines, B. and Musen, M., editors, *Proceedings of Eleventh Knowledge Acquisition Workshop*.

Shaw, M. L. G. and Gaines, B. R. (2005). Expertise and expert systems: emulating psychological processes. In Fransella, F., editor, *The Essential Practitioner's Handbook of Personal Construct Psychology*, pages 87–94. Wiley, Chichester, UK.

Shaw, M. L. G. and McKnight, C. (1981). *Think Again: Personal Problem-solving and Decision-making*. Prentice-Hall, Englewood Cliffs, NJ.

Shepard, R. N. (1964). Attention and the metric structure of the stimulus space. *Journal of Mathematical Psychology*, 1(1):54–87.

Slater, P. (1976). *Dimensions of Intrapersonal Space: Volume 1*. John Wiley, London.

Slater, P. (1977). *Dimensions of Intrapersonal Space: Volume 2*. John Wiley, London.

Spencer, H. (1862). *First Principles*. Williams and Norgate, London.

Stenning, K. and Lambalgen, M. v. (2008). *Human Reasoning and Cognitive Science*. MIT Press, Cambridge, MA.

Stephenson, W. (1953). *The Study of Behavior: Q-technique and its Methodology*. University of Chicago Press, IL.

Tan, F. B., Tung, L.-L., and Xu, Y. (2009). A study of web-designers' criteria for effective business-to-consumer (b2c) websites using the repertory grid technique. *Journal of Electronic Commerce Research*, 10(3):155–177.

Thurstone, L. L. (1935). *The Vectors of Mind: Multiple-factor Analysis for the Isolation of Primary Traits*. University of Chicago science series. University of Chicago Press, Chicago.

Tofan, D., Avgeriou, P., and Galster, M. (2014). Validating and improving a knowledge acquisition approach for architectural decisions. *International Journal of Software Engineering and Knowledge Engineering*, 24(4):553–589.

Torgerson, W. S. (1958). *Theory and Methods of Scaling*. Wiley, New York.

Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

Wason, P. C. (1968). Reasoning about a rule. *Quarterly Journal of Experimental Psychology*, 20(3):273–281.

Whitehead, A. N. (1929). *Process and Reality: An Essay in Cosmology*. Free Press, New York.

Yorke, D. M. (1978). Repertory grids in educational research: Some methodological considerations. *British Educational Research Journal*, 4(2):63–74.

Yorke, D. M. (1983). Straight or bent? an inquiry into rating scales in repertory grids. *British Educational Research Journal*, 9(2):141–151.

Young, S. M., Edwards, H. M., McDonald, S., and Thompson, J. B. (2005). Personality characteristics in an xp team: a repertory grid study. *SIGSOFT Software Engineering Notes*, 30(4):1–7.

Zenker, F. and Gärdenfors, P. (2015). *Applications of Conceptual Spaces: The Case for Geometric Knowledge Representation*. Springer, Cham.

Most of our publications cited in the references are freely available on the web