AnthroTools: A Package in R

Benjamin Grant Purzycki[1], Alastair Jamieson-Lane[2]

[1]Centre for Human Evolution, Cognition, and Culture, University of British Columbia, Vancouver, BC, Canada

[2]Department of Mathematics, University of British Columbia, Vancouver, BC, Canada

Corresponding Authors:
Benjamin Grant Purzycki, Centre for Human Evolution, Cognition, and Culture, University of British Columbia, 1871 West Mall, Vancouver, BC, V6T 1Z2, Canada.
Email : bgpurzycki@alumni.ubc.ca

Alastair Jamieson-Lane, Department of Mathematics, The University of British Columbia, Room 121, 1984 Mathematics Road, Vancouver, BC, V6T 1Z2, Canada.
Email: aja107@math.ubc.ca

Abstract: We created AnthroTools, a basic free-list analysis tool that prepares free-list data for further analysis as well as basic, Bayesian cultural consensus analysis for use in the open source program R. Free-list data elicitation is a simple technique for ethnographic research. By listing items, participants interrogate their own mental models of any given domain, and researchers acquire rich, naturalistic, *emic*, qualitative data that is remarkably useful for further qualitative and quantitative research and development. However, back-end data preparation is considerable, and often requires specific software or extensive preparation in standard spreadsheet programs. Additionally, current cultural consensus analysis tools either require specialized software or have computationally taxing methods. AnthroTools expedites these techniques, rapidly examines diagnostics, and compares analyses to idea simulations. In this paper, we introduce the functions of the package, including walkthrough examples, and point to its novel features and manifold analytical options.

Keywords: cognitive anthropology, free-list data, cultural consensus analysis, software

Acknowledgements: The authors were financially supported by the Cultural Evolution of Religion Research Consortium (CERC), sponsored by SSHRC and the John Templeton Foundation during the preparation of this work. We thank W. Penn Handwerker, John Shaver, and Aiyana Willard for their input.

Last Modified: May 9, 2016

1. Introduction

Analytical tools in social research often suffer from a considerable amount of front-end work for data management and preparation for further analyses. While extant free software such as ANTHROPAC (Borgatti 1996) is ready-made for consensus and content domain analyses, such software often requires data to be organized in ways not readily amenable for integration with greater data sets and requires navigating software-specific interfaces that are not readily transferrable to other programs. Because of this, we created AnthroTools (Jamieson-Lane and Purzycki 2016), a package for use in R (R Core Team 2012) to enhance foundational data management for social scientists interested in content domains. This paper introduces the package including walk-through examples with the mathematical operations of its functions. Currently, there are two primary modes of analysis in AnthroTools: 1) multi-factor free-list data analysis and 2) cultural consensus analysis. In this paper, we first briefly discuss the benefits of AnthroTools, and then discuss these modes of analysis in turn, with examples that are pre-loaded into the package for ease of use.

2. AnthroTools

In our judgment, the best—if not only—available analytical software devoted to free-list analysis is ANTHROPAC/UCINET (Borgatti 1996; Borgatti, Everett, and Freeman 2002). With respect to free-list analyses, ANTHROPAC requires that one organize free-list data in a text file in the following format:

```
# informant1
item1
item2
item3
item4
item5
item6

# informant2
item1
item2
item3
item4
item5
item6

# informant3
item1
item2
item3
item4
item5
item6
```

Especially for multi-site projects with large sample sizes, this input method is extremely laborious and not readily compatible with otherwise standard data storage formats such as spreadsheets with pivot table functions. Another alternative is to organize free-list data in a spreadsheet with the following standardized format:

Participant ID#	Order Listed	Item Listed
1	1	item1
1	2	item2
1	3	item3
2	1	item1
2	2	item2
2	3	item3
3	1	item1
3	2	item2
4	1	item1
4	2	item2
4	3	item3
4	4	item4

This method is helpful for creating quick tables with spreadsheet software (e.g., "pivot" or "pilot" tables in Excel or Calc, respectively), but subsequent calculations (e.g., of item salience) or transformations (e.g., dichotomizing presence and absence) can be time consuming and prone to error, particularly if one has a large dataset with many participants. Moreover, if researchers are not inclined or able to write macros that often require updating with subsequent versions of software, these tasks can be especially tedious.

With respect to free-list data, AnthroTools allows researchers to get beyond these limitations by: 1) quickly analyzing free-list data, 2) converting it into various datasets that are more immediately useable for merging with other data and subsequent analyses, and 3) taking advantage of the versatility of the free, open-source program R (R Core Team 2012).

Here, we write with the assumption that readers have a very basic user-level understanding of R with no handy free-list data. For those who have little to no experience with R, we highly recommend starting with Field, Miles, and Field (Field, Miles, and Field 2012). In order to ease the transition, we have also written a ready-made companion AnthroTools R script as an online supplement to make running analyses quicker and easier with step-by-step instructions (http://xxx.xxx.xxx). In the event that readers already have free-list data in the ANTHROPAC format, we have also included a function that readily converts this data into a standardized spreadsheet format (see below).

3. Free-list Analyses

3.1. Getting Started

To install and load the package, open R and run the following:

```
install.packages("devtools")
library("devtools")
install_github('alastair-JL/AnthroTools')
library(AnthroTools)
```

Note that it may help to run this on occasion as we update the package.

For ease of transition, we created a function that also allows users who have compiled data in the Anthropac format to transform it into a more manageable format for further data integration. The "LoadFromAnthropac" function works simply by letting R know where your file is and works with both .txt and .csv files. First, be sure to set your working directory for where the file is (e.g., `setwd("C:/Users/Benjamin/Desktop")`, then run:

```
LoadFromAnthropac("C://Users//Benjamin//Desktop//Bands.txt")
```

This tells R to transform a text file called "Bands" located on the desktop (of Benjamin's computer) into a more readily analyzable data set.

Running the following opens the general help menu for the package:

```
help("AnthroTools")
```

This help file is replete with links to specific functions within the package, provides more examples, output definitions, full code, extra discussion, and sources.

3.2. Primary Salience Calculations

Free-list data elicitation techniques are invaluable for assessing the content and structure of human thought (Gravlee 1988; Quinlan 2005; Romney and D'Andrade 1964; Schrauf and Sanchez 2008; Smith 1993; Smith et al. 1995; Smith and Borgatti 1997, 1997; Thompson and Juan 2006). Free-list tasks require that participants list objects in a given domain. At its core, the task lends itself to accounting for the ubiquity of specific items, but also to concept salience (i.e., its cognitive accessibility and/or importance) of specific items across the minds of individuals. Generally, free-list tasks are thought of as a preliminary step toward more focused research efforts (Bernard 2011). However, the presence or absence or co-occurrence of listed items, item frequencies, and/or salience scores may also be used as dependent or independent variables in targeted studies (Purzycki n.d.; Schrauf and Sanchez 2008).

To begin with a common hypothetical example, let us say that you asked people to freely list the kinds of fruits they knew (Bernard and Ryan 2009). After installing and loading the package, simply running:

```
data(FruitList)
```

will call up the sample free-list data we created. To view the dataset, run:

```
View(FruitList)
```

Note that in this dataset, there are three variables: "Subj" is the participant ID number, "Order" is the order in which participants listed items, and "CODE" is the data point. In this dataset, there are 20 participants.

The CalculateSalience command calculates each item's salience score and creates a new column for this score. We will create a new object called "FL" that is this new dataset.

```
FL <- CalculateSalience(FruitList)
```

```
View(FL)
```

Item salience is calculated by taking the order in which an item is listed, inversely coding this order number (e.g., when an individual lists five items, first-listed items are given a “5” whereas the fifth item listed gets a “1”), and dividing this number by the total number of items listed by that individual. This way, items listed first get a salience score of “1” and each subsequently listed item gets a smaller score. The full syntax for CalculateSalience is:

```
FL <- CalculateSalience(dataset, Order = "Order", Subj = "Subj", CODE = "CODE
        ", GROUPING = NA, Rescale = FALSE, Salience = "SScore")
```

This allows you to call your variables whatever you want and associate them with the function’s operations (e.g., if your participant ID variable is called “ID” in your dataset, the argument would be `Subj = “ID”`).

We have included two novel features for additional analyses: the “GROUPING” and “Rescale” components. The GROUPING component breaks free-list data down into groups in case you wish to analyze differences between any categorical factors (e.g., sex of participant, cultural group, etc.; see Section 3.4). The “Rescale” component normalizes salience scores in the event that you wish to normalize salience, so as to prevent individuals with long lists from dominating your salience analysis. This function simply divides all individual salience scores by the sum of salience so that all salience scores add to 1 for each participant. The “Salience” argument tells R what to call a new column containing the salience calculation (in this case, we called it “SScore”).

In the event that there are inconsistencies or errors in the data set, this function will operate, but also return warnings. We recommend viewing the warnings as this can be useful for diagnostics. If, for instance, you have multiple participants with the same ID, or your order variables do not start at “1” or have multiples of the same order number, the error will point you to which individuals have which errors and what the error is. See Section 3.6 below for quick cleaning options.

The function “SalienceByCode” takes this new data set with the by-item salience scores, and creates another output specifically designed to handle overall salience by item type or category. Run the following:

```
FL.S <- SalienceByCode(FL, dealWithDoubles = "MAX")
View(FL.S)
```

The output should look like this:

	CODE	MeanSalience	SumSalience	SmithsS
1	pear	0.695833	5.566667	0.278333
2	orange	0.644444	3.866667	0.193333
3	apple	0.758889	11.383333	0.569167
4	strawberry	0.592593	5.333333	0.266667
5	banana	0.731250	5.850000	0.292500
6	plum	0.811905	5.683333	0.284167
7	lemon	0.508333	2.033333	0.101667
8	peach	0.875000	1.750000	0.087500

This argument calculates by-item mean salience and the sum of salience scores. It calculates Smith's S by dividing the sum of salience scores by number of participants in the sample (again, in this case *n* = 20) (Borgatti 1998; Quinlan 2005; Smith 1993; Smith et al. 1995). The standard equation for item categories' salience (Smith's *S*) is:

Equation 1: $S = \frac{\sum s}{N}$

where *s* = individual item salience and *N* = participant sample size (Borgatti 1998; Quinlan 2005; Smith 1993; Smith et al. 1995; Smith and Borgatti 1997).

As is often the case, data points get repeated. For instance, participants might list specific items more than once or subsequent recoding of data yields multiple instances for the same category. This can inflate Smith's S values. We therefore created the "dealWithDoubles" argument to allow flexibility in handling such instances. On the default setting, the function will assume that no such cases arise. If there are such cases, R will report an error and encourage you to use the "dealWithDoubles" command. So, if you run the same script above, but delete the "dealWithDoubles" argument, you'll see the error notifying you of repeated items.

Aside from DEFAULT, you also have the options MAX, MEAN, SUM and IGNORE. MAX indicates that you want the computer to attend *only* to the first time a respondent lists a particular CODE, and ignore subsequent mentions (e.g., if someone lists "apple" twice, it only keeps track of its earliest listing). For MEAN, the computer determines each respondent's mean salience for repeated items and uses this value to calculate Smith's S. For SUM, you are asking the computer to determine each respondent's TOTAL salience with respect to a given code. If this value is greater than "1", R will report an error identifying the source of the problem, and recommend normalization. IGNORE is merely a way of suppressing errors, and is thus not recommended. The full syntax for SalienceByCode is:

```
FL.S <- SalienceByCode(FL, Subj = "Subj", CODE = "CODE", GROUPING = "GROUP",
       Salience = "Salience", dealWithDoubles = "DEFAULT")
```

3.3. Tables for Further Analyses

We also created a variety of options for transforming free-list data into useful tables for further analyses (e.g., using free-list data as a dependent or independent variable, running factor analysis, etc.). The general syntax is:

```
FLT <- FreeListTable(dataset, CODE = "CODE", Salience = "Salience", Subj =
         "Subj", tableType = "DEFAULT")
View(FLT)
colSums(FLT)
```

Currently, there are four types of tables: "PRESENCE", "SUM_SALIENCE","MAX_SALIENCE" and "FREQUENCY". "PRESENCE" converts data into a "1" if participants mentioned the specified code and "0" if they did not. This might be helpful for logistic regressions where your dependent variable is whether or not someone listed a specific type of item. If you specify "FREQUENCY", then you will get a count of how often each code was mentioned by each person. This might be useful in general linear models, especially when your data consist of relatively frequently repeated items. If you use "SUM_SALIENCE" then you will get the total salience each person has associated with each code. If you use "MAX_SALIENCE" then you will get the maximum salience, i.e., the salience of the code the first time it was mentioned. Quick summations for each column can be made with the `colSums()` function.

3.4. The Grouping Argument

Grouping allows salience analysis to be carried out on individuals belonging to different groups (for example, different field sites). This option is optimal for cross-cultural or any comparative study where participants are grouped by any given factor. Again, simply identifying which factor you wish to consider by using the GROUPING argument will perform the full task. Subject code overlap between groups is permitted, and salience calculations can be done on a group-by-group basis for each item code. The following includes syntax for the toy example included in the package. You'll see that the final output is a table calculating salience by item, by group.

```
data(WorldList)
WL <- CalculateSalience(WorldList , GROUPING = "GROUPING")
WL.S <- SalienceByCode(WL, dealWithDoubles = "MAX", GROUPING = "GROUPING")
View(WL.S)
```

The output will look something like the table below. The GROUPING variable here divides the analysis by "culture"—the mainlanders, people from the island, and some people from the moon. Note that the set is sorted by CODE and the mean salience, sum of salience scores, and Smith's S are all calculated based on within group sample sizes (we truncated the table and the decimal places).

```
GROUPING     CODE         MeanSalience  SumSalience   SmithsS
MAINLAND     plum         0.8333        5.0000        0.2941
ISLAND       plum         0.7729        6.1833        0.4756
MOON         plum         0.5833        1.7500        0.1250
MAINLAND     strawberry   0.6875        2.7500        0.1618
ISLAND       strawberry   0.5000        2.0000        0.1538
MOON         strawberry   0.4722        1.4167        0.1012
MAINLAND     pear         0.5833        2.9167        0.1716
ISLAND       pear         0.5733        2.8667        0.2205
MOON         pear         0.3500        1.4000        0.1000
...          ...          ...           ...           ...
```

3.5. Cleaning Data

Now that we have covered the basic functions of the free-list analysis component of AnthroTools, we also wish to walk through our `CleanFreeList` function. Note that this function should be used with caution as it removes problematic data in specific ways. Sometimes it is simply easier to make corrections manually, but in the event that you wish to streamline various errors associated with free-lists, you can use this function to do so. In default state, its full form is:

```
CleanFreeList(mydata, Order = "Order", Subj = "Subj", CODE = "CODE",
        ejectBadSubj = TRUE, deleteDoubleCode = TRUE, ConsolidateOrder = TRUE,
        RemoveMissingData = TRUE)
```

The “ejectBadSubj” argument simply excises participants who have bad data ranging from duplicates to missing order values. If participants repeat items or your post-hoc codes happen to repeat, you can drop all subsequent instances with the “deleteDoubleCode” argument. “ConsolidateOrder” corrects for errors in order listing sequences. For example, if for some reason a participant’s order sequence is 1, 2, 5, 7, it will correct it to 1, 2, 3, 4. This matters for subsequent salience calculations. If you have rows where the “Code” variable is NA or blank, you can remove them with the “RemoveMissingData” argument. Further details for free-list analyses can be found by running the help function for AnthroTools and selecting the available links for further inquiry, or by running `help(CleanFreeList)` directly.

4. Cultural Consensus Analysis

4.1. Description of method

Cultural consensus analysis (Anders and Batchelder 2012, 2013; Anders, Oravecz, and Batchelder 2014; Caulkins 2004; Miller et al. 2004; Oravecz, Anders, and Batchelder 2013; Oravecz, Vandekerckhove, and Batchelder 2014; Romney, Weller, and Batchelder 1986) is a form of analysis designed to study respondents’ answers to (approximately) multiple choice questions, where the interviewer does not necessarily know the correct answer or where the notion of a “correct” answer may itself be misguided. The analysis searches for correlations between respondents’ answers, and uses these to infer the level of “expertise” of each respondent, before going on to determine which answer is considered correct by the majority of respondents (weighted such that more attention is paid to respondents with greater expertise). The answers found are then considered to be the “cultural consensus”.

Such analysis might be applicable to such questions as “What is the name of this plant?” or “Is this illness/injury contagious?” The analysis depends on several main assumptions (Hruschka and Maupin 2013; Romney et al. 1986), but the two most crucial are that questions have a single “correct” answer and that all questions are in the same domain.

4.2. Theory

Consensus analysis rests on the assumption that you are asking multiple choice questions[1], and that for any given question, participants know the answer to the question with some probability *p*, and guess the answer with some probability *1-p*. *P* is expected to vary between participants depending on their “competence”, but does not vary between questions: it is assumed that all questions are of similar difficulty and within the same domain of knowledge.

The first step in the analysis is to estimate the value of *p* associated with each individual. This is done by building the correlation between individuals’ answers **M** and then accounting for

[1] UCINET includes a “continuous” cultural consensus analysis that allows responses to be on scales rather than categories. The authors of the program caution its use, however, and it hasn’t yet been thoroughly interrogated.

the probability that two individuals might pick the same answer by chance, resulting in a new matrix **M***, which is designed to represent the rate at which each pair of individuals select the same answer because they both knew the correct answer. Entries of **M*** are found using the equation:

Equation 2: $M^*{}_{i,j} = (LM_{i,j} - 1)/(L - 1)$

Here, *L* represents the number of possible answers to a given question.

Assuming all questions and individuals are relatively independent of one another, this value (the rate of both knowing the answer) should be equal to the product of the *p* values for the two individuals[2]. Thus, for every pair of individuals we have the equation:

Equation 3: $M_{i,j} = p_i p_j$

Here, we refer to the *i,j*th entry of the matrix, and the values of *p* refer to the competence of subjects *i* and *j*.

In general, this system will result in more equations than unknown variables, and thus cannot be solved exactly. However, assuming that all the assumptions of consensus analysis have been met, the set of equations should be *approximately* soluble. The AnthroTools package does this using what we'll call the "Comrey Iteration" (Comrey 1962)[3].

Once this factor analysis has been used to determine the competence scores of each individual, than the "correct" answer to each question can be inferred from the survey by using Bayes' theorem:

Equation 4: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

In the particular case of consensus analysis, this translates to "The probability of answer A being correct for Question 1 given our current survey results is equal to the probability of seeing *this particular* series of answers *given* that A was correct, multiplied by the prior probability that A was correct, and divided by the probability of seeing this particular series of answers (without knowing anything about the answer)"[4].

4.3. Use of Consensus Analysis with the AnthroTools package

[2] If I know the correct answer 2/3 of the time, and you know the correct answer 2/3 of the time, then we both know the correct answer 4/9 of the time.

[3] There is nothing inherent in Comrey's method of factor analysis to prevent competence scores above one or below zero. Furthermore, in going from **M** to **M*** it is possible to introduce negative values, for example, if two individuals agree with one another *less* than would be expected by chance than the method may infer that the two are in active disagreement, leading to unusual results.

[4] In practice, once competence is known, it is easy to calculate P(survey results |answer is A), and all else being equal, it can be assumed that P(answer is A) is equal to P(answer is B), and so on, leading to P(answer is A) = 1/N (where N is the number of possible answers). The only remaining difficulty is determining P(survey results), but this we can avoid calculating simply by remembering that the probabilities of all possible answers must add up to one.

AnthroTools automates consensus analysis with the “ConsensusPipeline” function. The following is a walkthrough of our toy example. First, call up the dataset:

```
data(ConsensusTestData)
View(ConsensusTestData)
```

Here is the data set:

	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7	Question 8
Person 1	2	1	2	2	1	2	3	1
Person 2	3	2	1	3	1	1	2	1
Person 3	2	2	1	1	3	2	3	2
Person 4	1	2	3	1	2	1	3	1
Person 5	1	3	1	3	1	2	2	2
Person 6	2	2	1	2	3	1	2	2
Person 7	3	2	1	1	1	1	2	1
Person 8	3	2	3	1	2	1	2	3

To run the consensus analysis, simply invoke the function “ConsensusPipeline” and indicate the number of total possible answers to your multiple-choice key (in this case it’s 3):

```
Results <- ConsensusPipeline(ConsensusTestData, 3)
Results
```

These results contain the calculated answers, the competency assumed for each person, the actual test scores found for each person (assuming the answers calculated are correct) and the probability distribution for each of the answers. Note that in general we are primarily interested in the answers calculated, but looking at the other results can be useful for informing you if your input has led to strange results (for example, negative probabilities). Here is the output, which will have the following elements: $Answers; $Competence, $origCompetence; $TestScore; $Probs; and $reportback:

```
$Answers
Question 1 Question 2 Question 3 Question 4 Question 5 Question 6 Question 7 Question 8 
         3          2          1          1          1          1          2          1 

$Competence
   Person 1    Person 2    Person 3    Person 4    Person 5    Person 6    Person 7    Person 8 
0.000000000 0.730098933 0.007199317 0.269227910 0.026867347 0.245326389 0.973873955 0.513715567 

$origCompetence
[1] -0.284995564  0.730098933  0.007199317  0.269227910  0.026867347  0.245326389  0.973873955  0.513715567

$TestScore
Person 1 Person 2 Person 3 Person 4 Person 5 Person 6 Person 7 Person 8 
   0.250    0.875    0.375    0.500    0.375    0.500    1.000    0.625 

$Probs
    Question 1  Question 2  Question 3  Question 4  Question 5   Question 6   Question 7  Question 8
1 0.0010616681 0.001750085 0.963055951 0.955268538 0.962671730 0.9979612622 0.0034738974 0.982874743
2 0.0004699594 0.996354875 0.006855928 0.007459002 0.030349790 0.0002476973 0.9955920386 0.001984112
3 0.9984683724 0.001895040 0.030088121 0.037272460 0.006978479 0.0017910405 0.0009340641 0.015141145

$reportback
[1] "We encountered 1 individuals with ``negative'' competence. We found 0 individuals with competence over
one. The magnitude of the main factor was 1.4000155273283. The second factor's magnitude was 1.06401058316597,
giving a ratio of 1.31579097941164."

$reportNumbers
[1] 1.000000 0.000000 1.400016 1.064011 1.315791
```

The $Answers section provides the culturally “correct” answers provided by the function, while $Competence reports the estimated cultural competence for each individual. $origCompetence is the competence score before the transformation into the [0, 1] range. Assuming, of course, that the answer key is correct, $TestScore reports test scores for each individual while $Probs is the probability that each answer is correct for a given question.

We also provide various qualitative reports in the output. $reportback gives important feedback from the analyses. In the case of strange results, it may be worthwhile to examine whether or not the assumptions have been satisfied. The return value “reportback” has a written summary detailing such anomalies, including the competence scores which are out of bounds, along with the “Comrey Ratio”[5] used to determine if it is likely that a single domain is Present. “reportNumbers” yields the raw numbers from this summary, should you wish to analyze them.

4.4. Simulating data

If you would like to further test your data by comparing it statistically to “ideal” data with similar properties (that is, randomly generated data where all the analyses’ assumptions hold true), you can generate such data using the “GenerateConsensusData” function.

```
FakeData <- GenerateConsensusData(8,12,5)
```

Here, the numbers refer to the number of “participants” in our imagined survey (in this case “8”), the number of questions (“12”), and the number of possible answers to each question (“5). Your output will be divided into three chunks. For now, only the “$Survey” component is of concern. It is a person x question matrix of your data. This matrix is suitable for pushing through your consensus analysis pipeline as follows.

```
FakeResults <- ConsensusPipeline(FakeData$Survey,5,safetyOverride=TRUE)[6]
FakeResults
```

With these ideally generated results in hand, you can then compare to the “true” results, namely:

```
FakeResults$Answers
FakeData$Answers
```

[5] Here we refer to the “Comrey Ratio” as the ratio between the primary and secondary vectors found when determining the competence scores from the agreement matrix. Previous explanations of consensus analysis frequently refer to this as an “eigenvalue” ratio (Borgatti and Halgin 2011; Weller 2007). To the best of our understanding, the eigenvalues of the agreement matrix are not the appropriate numbers to use in this context. Based on the context, the lengths of the factors found with the Comrey iteration appears to be the closest appropriate analogue (cf. Borgatti and Halgin 2011). It is unclear whether previous authors were using the correct concept with the wrong label, or the correct label with the wrong concept. For further discussion of this difficulty, please view “help(ConsensusCaveats)”, although in either case, the “criterion” lacks strong justification.

[6] Under any normal circumstance we do not recommend setting the parameter “safetyOverride” to true. However, for the sake of analyzing randomly generated results, it can prove helpful, as randomly generated results can occasionally (by chance) lead to invalid results.

By generating large amounts of such data, one can easily statistically test the method's correctness against the certainty predicted by "FakeResults$Probs". The function "ConsensusStressTest" does this automatically (this might take a little time):

```
StressTestResults <- ConsensusStressTest(15, 25, 4, 5000,lockCompetence =
      0.6)
```

Here we have instructed the computer to create 5000 surveys, each with 15 participants and 25 questions. Each question has 4 possible answers. Further, we have demanded that ALL participants have a true competence score of 0.6. This allows us to investigate how much variance in competence is produced purely as a result of luck as opposed to actual variation in skill levels (Hruschka and Maupin 2013). If you want to compare to results where competence does in fact vary, simply leave "lockCompetence" blank, and participants will be assigned a variety of competence values between zero and one.

For each simulated survey, the stress test method calculates the "correct" answers, determines the probability it assigns to each of these answer, and finds the Comrey Ratio associated with its solution. It then determines the *expected* number of errors (according to the analysis), along with the *actual* number of errors (via comparison to "true" results generated along with the survey). We note that the standard analysis often over or under estimates the number of errors by a factor of two (depending on the details of the survey) and that it is thus useful to compare to simulate results to determine how well calibrated the methods estimations are.

The average competence values and variation in competence values are also calculated, allowing one to ask such questions as "Assuming that all individuals have a true competence of 0.6, what is the chance of variance greater than 0.1 *purely via statistical noise?*"

```
mean(StressTestResults[[3]]>0.1)
```

Another important question one might ask is: "Even if all assumptions of consensus analysis hold true, what is the probability that my results will satisfy the "Comrey Ratio'' > 3 criterion" (see note 6)? The following answers this question:

```
sum(StressTestResults[[4]]>3.0)/5000
```

Using this tool prior to conducting a survey may help inform you as to whether you have sufficient questions/participants to gain meaningful insight from your data. However, it is worth noting that even with 15 participants and 25 questions, only 22% of simulated surveys in our testing achieved this criterion, suggesting that the criterion as it currently stands is either a little too conservative, or simply inappropriate to the task, as it accepts only one fifth of survey results even when all assumptions of consensus analysis are met perfectly.

4.5. Stress Tests

Having implemented Cultural Consensus Analysis in this package, and compared its findings to a large number of simulated data sets that perfectly match all assumptions of the model, several results become apparent. Firstly, the method as it currently stands is a reasonable estimation tool, and given sufficiently large number of participants and questions, it will correctly determine the "culturally appropriate" answer in the vast majority of cases. However, the classical method does suffer from a number of questionable approximations and questionable rules of thumb, in particular the use of Comrey approximation, and the "rule of three" with respect to the Comrey ratio criterion.

The primary issue with the Comrey ratio criteria is that there appears to be no mathematical or experimental basis justifying the particular choice of the value three. Our experiments above indicate that in many cases survey results that fit all assumptions of consensus analysis perfectly will fail to achieve a Comrey ratio of 3 or higher, thus indicating that the criteria itself does not accurately distinguish between appropriate an inappropriate uses of Consensus analysis.

The use of Comrey's minimum residual method of factor analysis also proves problematic, for the simple reason that this method was never designed to be used on probabilities. It contains no restriction keeping its results bounded between zero and one, and hence will from time to time result in competence values with no discernable meaning. More disturbingly, this leads one to question the validity of the competence values that it *does* find.

To explore how much of an issue this was, we used the following lines of code (again, running this will take a little time):

```
StressTestResultsB <- ConsensusStressTest(15,25,4,5000,lockCompetence = 0.6)
StressTestResultsC <- ConsensusStressTest(15,25,4,5000)
```

In the case where the actual competence of all individuals was locked at 0.6 (fairly far away from both 0 and 1), the stress test encountered 10 individuals whose estimated competence was outside the given range (out of a total of 75000). However, in the second case where individual competencies are sampled uniformly between zero and one prior to each survey, the stress test encountered a total of 5685. That is to say, on average, more than one such individual per survey. Given that the ways of dealing with such unacceptable competence values are largely *ad hoc*, this presents a clear challenge to the usefulness of this method.

A final issue faced by consensus analysis that was not previously obvious is the fact that its calculations of uncertainty are in fact poorly calibrated. More specifically, when dealing with simulated data such as that generated above, the method will often predict close to 2,000 errors in its assessment of the "correct" answer, when in actual fact there are only 1,496. This discrepancy can become even worse in cases where all individuals have only a moderate level of competency: in the case of StressTestResultsC above, we found that the algorithm predicted 2163 wrong guesses, while only determining a total of 1057 wrong answers (out of a total of 125,000 questions). More troublesome is the fact that for surveys with fewer questions and participants, the method is actually liable to *underestimate* its fallibility; in the case of simulating 5000 surveys of 8 questions and 8 participants (4 possible answers, competence locked at 0.6), it was found that the method predicted 2199 errors, but ended up making 2769. Simply put, the fact that Comrey iteration is only used to *estimate* the competence values of those surveyed leads to all subsequent determinations of probability being estimates as well- and not, in all cases, unbiased ones.

Our original plan upon making this observation was to provide the current code (with suitable warnings) and then find a way of implementing a better method, namely, one that used Bayesian reasoning throughout, rather than merely in the final step. As it turned out, this has already been achieved (Oravecz et al. 2014), along with standalone program to go with it. The R package CCTpack (Anders 2014) provides an alternative for those who prefer to remain within the framework of R. Our observations thus far are that these more thorough methods spend significant amounts of time in their parameter estimation, particularly for large data sets, while the more approximate method provided in AnthroTools provides a significantly quicker result. In cases where simply determining the "correct" answers is required, and your data set is large enough to limit the issues caused by approximation, AnthroTools may thus prove useful as an exploratory measure, with the heavier tools used for validity testing.

5. Conclusion

Our primary goals for developing this tool were to make free-list and consensus analyses easier to manage, readily accessible, more efficient, and immediately useable for field researchers interested in content domains. It should be especially helpful for researchers doing cross-cultural work, or working with large samples. AnthroTools allows researchers to work within R, and rapidly calculates tables for further analyses. It is our hope to inspire cultural anthropologists to use these powerful and useful methods who would otherwise avoid free-list data and cultural consensus analyses because of their backend work or mathematical esoterica. For those who already do, we hope this makes the process easier. If researchers have a particular kind of add-on that they would find beneficial, we are more than happy to entertain all suggestions.

6. References

Anders, R. 2014. CCTpack: Cultural consensus theory applications to data. R package version 1.4.

Anders, R. and W. H. Batchelder. 2012. Cultural consensus theory for multiple consensus truths. *Journal of Mathematical Psychology* 56(6):452–69.

Anders, R., Z. Oravecz, and W. H. Batchelder. 2014. Cultural consensus theory for continuous responses: A latent appraisal model for information pooling. *Journal of Mathematical Psychology* 61:1–13.

Anders, R. and W. H. Batchelder. 2013. Cultural consensus theory for the ordinal data case. *Psychometrika* 80(1):151–81.

Bernard, H. R. 2011. *Research Methods in Anthropology*. 5th Edition. Lanham: AltaMira Press.

Bernard, H. R. and G. W. Ryan. 2009. *Analyzing Qualitative Data: Systematic Approaches*. Thousand Oaks: SAGE.

Borgatti, S. P. 1996. ANTHROPAC 4. Columbia: Analytic Technologies.

Borgatti, S. P. 1998. Elicitation techniques for cultural domain analysis. In *The Ethnographer's Toolkit, Vol. 3*. Walnut Creek: AltaMira Press.

Borgatti, S. P., M. G. Everett, and L. C. Freeman. 2002. Ucinet for Windows: Software for Social Network Analysis. Harvard: Analytic Technologies.

Borgatti, S. P. and D. S. Halgin. 2011. Consensus analysis. Pp. 171–90 in *A Companion to Cognitive Anthropology*, edited by D. B. Kronenfeld, G. Bennardo, V. C. de Munck, and M. D. Fischer. Wiley-Blackwell. Retrieved April 14, 2016. (http://onlinelibrary.wiley.com/doi/10.1002/9781444394931.ch10/summary).

Caulkins, D. D. 2004. Identifying culture as a threshold of shared knowledge: A consensus analysis method. *International Journal of Cross Cultural Management* 4(3):317–33.

Comrey, A. L. 1962. The minimum residual method of factor analysis. *Psychological Reports* 11(1):15–18.

Field, A., J. Miles, and Z. Field. 2012. *Discovering Statistics Using R*. 1 edition. Thousand Oaks, Calif: SAGE Publications Ltd.

Gravlee, L. 1988. The uses and limitations of free listing in ethnographic research. Retrieved (http://gravlee.org/ang6930/freelists.htm).

Hruschka, D. J. and J. N. Maupin. 2013. Competence, agreement, and luck testing whether some people agree more with a cultural truth than do others. *Field Methods* 25(2):107–23.

Jamieson-Lane, A. and B. G. Purzycki. 2016. AnthroTools: A Package in R.

Miller, M. L., J. Kaneko, P. Bartram, Joe Marks, and Devon D. Brewer. 2004. Cultural consensus analysis and environmental anthropology: Yellowfin Tuna Fishery Management in Hawaii. *Cross-Cultural Research* 38(3):289–314.

Oravecz, Z., R. Anders, and W. H. Batchelder. 2013. Hierarchical bayesian modeling for test theory without an answer key. *Psychometrika* 80(2):341–64.

Oravecz, Z., J. Vandekerckhove, and W. H. Batchelder. 2014. Bayesian cultural consensus theory. *Field Methods* 1525822X13520280.

Purzycki, B. G. in press. The evolution of gods' minds in the Tyva Republic. *Current Anthropology*.

Quinlan, M. 2005. Considerations for collecting freelists in the field: Examples from ethobotany. *Field Methods* 17(3):219–34.

R Core Team. 2012. R: A Language and Environment for Statistical Computing. Vienna: R Foundation for Statistical Computing.

Romney, A. K. and R. G. D'Andrade. 1964. Cognitive aspects of English kin terms. *American Anthropologist* 66(3):146–70.

Romney, A. K., S. C. Weller, and W. H. Batchelder. 1986. Culture as consensus: A theory of culture and informant accuracy. *American Anthropologist* 88(2):313–38.

Schrauf, R. W. and J. Sanchez. 2008. Using freelisting to identify, assess, and characterize age differences in shared cultural domains. *The Journals of Gerontology Series B: Psychological Sciences and Social Sciences* 63(6):S385–93.

Smith, J. J. 1993. Using ANTHOPAC 3.5 and a spreadsheet to compute a free-list salience index. *Field Methods* 5(3):1–3.

Smith, J. J. and S. P. Borgatti. 1997. Salience counts—and so does accuracy: Correcting and updating a measure for free-list-item salience. *Journal of Linguistic Anthropology* 7(2):208–9.

Smith, J. J., L. Furbee, Kelly Maynard, Sarah Quick, and Larry Ross. 1995. Salience counts: A domain analysis of English color terms. *Journal of Linguistic Anthropology* 5(2):203–216.

Thompson, E. C. and Z. Juan. 2006. Comparative cultural salience: Measures using free-list data. *Field Methods* 18(4):398–412.

Weller, S. C. 2007. Cultural consensus theory: Applications and frequently asked questions. *Field Methods* 19(4):339–68.