Inter-Coder Agreement Analysis

ATLAS.ti 8 Windows
Contents

Inter-coder Agreement (ICA) ................................................................................................................................................................... 4

Why It Matters .................................................................................................................................................................................................. 4

At What Phase in Your Project Should ICA Analysis Be Performed? .................................................................................................................. 4

Reliability and Validity .................................................................................................................................................................................................. 5

Requirements for Coding .................................................................................................................................................................................................. 5

Semantic Domains .................................................................................................................................................................................................. 5

Rules for Applying Codes .................................................................................................................................................................................. 6

Instructions for Coders (Summary Form) .................................................................................................................................................................. 7

How Agreement/Disagreement is Measured .................................................................................................................................................. 7

Methods for Testing ICA .................................................................................................................................................................................................. 8

Common Mistakes .................................................................................................................................................................................................. 12

Sample Size ........................................................................................................................................................................................................ 12

Acceptable Levels of Reliability ........................................................................................................................................................................ 13

How to Set Up a Project for ICA Analysis .................................................................................................................................................. 13

Project Management for the Principal Investigator (PI) ............................................................................................................................................... 14

Project Management for Coders ........................................................................................................................................................................ 18

Merging Projects for ICA Analysis .................................................................................................................................................................. 20

Running the ICA Analysis .................................................................................................................................................................................................. 21

Calculating an ICA Coefficient ........................................................................................................................................................................ 23

Exporting Results .................................................................................................................................................................................................. 24

References .............................................................................................................................................................................................................. 25
Inter-coder Agreement (ICA)

Why It Matters

The purpose of collecting and analyzing data is that researchers find answers to the research questions that motivated the study in the first place. Thus, the data are the trusted ground for any reasoning and discussion of the results. Therefore, the researchers should be confident that their data has been generated taking precaution against distortions and biases, intentional or accidental, and that the mean the same thing to anyone who uses them. Reliability grounds this confidence empirically (Krippendorff, 2004).

Richards (2009) wrote: “But being reliable (to use the adjective) beats being unreliable. If a category is used in different ways, you will be unable to rely on it to bring you all the relevant data. Hence, you may wish to ensure that you yourself are reliably interpreting a code the same way across time, or that you can rely on your colleagues to use it in the same way” (p. 108).

There are two ways to rationalize reliability, one routed in measurement theory, which is less relevant for the type of data that ATLAS.ti users have. The second one is an interpretivist conception of reliability. When collecting any type of interview data or observations, the phenomena of interest usually disappears right after it has been recorded or observed. Therefore, the analyst’s ability to examine the phenomena relies heavily on a consensual reading and use of the data that represent the phenomena of interest. Researchers need to presume that their data can be trusted to mean the same to all of their users.

This means “that the reading of textual data as well as of the research results is replicable elsewhere, that researchers demonstrably agree on what they are talking about. Here, then, reliability is the degree in which members of a designated community agree on the readings, interpretations, responses to, or uses of given texts or data. […] Researchers need to demonstrate the trustworthiness of their data by measuring their reliability” (Krippendorff, 2004, p. 212).

Testing the reliability of the data is a first step. Only after establishing that the reliability is sufficiently high, it makes sense to proceed with the analysis of the data. If there is considerable doubt what the data mean, it will be difficult to justify the further analysis and also the results of this analysis.

ATLAS.ti’s inter-coder agreement tool lets you assess the agreement of how multiple coders code a given body of data. In developing the tool we worked closely together with Prof. Klaus Krippendorff one of the leading experts in this field, author of the book Content Analysis: An Introduction of Its Methodology, and the originator of the Krippendorff’s alpha coefficient for measuring inter-coder agreement.

The need for such a tool as an integrated element in ATLAS.ti has long been evident and has been frequently requested by users. By its nature, however, it could not and cannot be a magic “just click a button and hope for the best” solution kind of tool. If you randomly click on any of the choices that ATLAS.ti offers to calculate an inter-coder agreement coefficient, ATLAS.ti will calculate something. Whether the number you receive will be meaningful and useful depends on how you have set up your project and your coding.

This means if you want to test for inter-coder agreement, it requires at least a minimal willingness to delve into some of the basic theoretical foundations of what inter-coder agreement is, what it does and can do, and also what it cannot do. In this manual, we provide some of the basics, but this cannot be a replacement for reading the literature and coming to understand the underlying assumptions and requirements for running an inter-coder agreement analysis.

Please keep in mind that the inter-coder agreement tool crosses the qualitative-quantitative divide. Establishing inter-coder agreement has its origin in quantitative content analysis (see for instance Krippendorff, 2019; Schreier, 2012). If you want to apply it and want to adhere to scientific standards, you must follow some rules that are much stricter than those for qualitative coding.

If you want to develop a code system as a team, yes, you can start coding independently and then see what you get. But this approach can only be an initial brainstorming at best. It cannot be used for testing inter-coder agreement.

At What Phase in Your Project Should ICA Analysis Be Performed?

A good time to have your coding checked by other coders is when you have built a stable code system and all codes are defined. This means, this is somewhere in the middle of the coding process. Once a satisfactory ICA coefficient is achieved, the principal investigator has the assurance that his or her codes can be understood and applied by others and can continue to work with the code system.

Reliability and Validity

Whereas reliability offers the certainty that research findings can be reproduced and that no or only limited external “noise” has contaminated the data or the results, validity assures that the assertions made by the research reflect the reality it claims to represent. Validity concerns truth(s).

Reliability relates to validity in the following ways:

The more unreliable the data, the less likely it is that researchers can draw valid conclusions from the data. In terms of coding data, this means that researchers need to identify valid accounts in the data to a degree better than by chance. If the agreement of two or more coders is not better than the agreement by chance, then reliability is low and you cannot infer that a common understanding of the data exists. Thus: Unreliability limits the chance of validity.
On the other hand, **reliability does not necessarily guarantee validity**. Two coders may share the same world view and have the same prejudices may well agree on what they see, but could objectively be wrong. Also, if two researchers may have a unique perspective based on their academic discipline but their reading is not shared by many people outside their own scholarly community, the reliability might be high but the outcome of the research has little chance of being substantiated by evidence of the reality that is inferred. As Krippendorff (2004) states: “Even perfectly dependable mechanical instruments, such as computers, can be wrong – reliably.” (p. 213).

A third aspect is the **dilemma between high reliability and validity**. Interesting interpretations might not be reproducible, or interesting data may not occur often enough to establish reliability. Highly reliable data might be boring and oversimplified in order to establish a high reliability in the first place.

### Requirements for Coding

Sources for unreliable data are intra-coder inconsistencies and inter-coder disagreements. To detect these, the coding process needs to be replicated. Replicability can be assured when several independently working coders (at least two) agree:

- on the use of the written coding instruction.
- by highlighting the same textual segments to which the coding instructions apply.
- by coding them using the same codes, or by identifying the same semantic domains that describe them and code them using the same codes for each semantic domain.

### Semantic Domains

A semantic domain is defined as a set of distinct concepts that share common meanings. You can also think about them as a category with sub codes. Examples of semantic domains are:

**EMOTIONS**
- emotions: joy
- emotions: excitement
- emotions: surprise
- emotions: sadness
- emotions: anger
- emotions: fear

**STRATEGIES**
- strategies: realize they are natural
- strategies: have a Plan B
- strategies: adjust expectations
- strategies: laugh it off
- strategies: get help

**ACTOR**
- actor: partner
- actor: mother
- actor: father
- actor: child
- actor: sibling
- actor: neighbour
- actor: colleague

Each semantic domain embraces **mutually exclusive concepts** indicated by a code. If you code a data segment with ‘emotions: surprise’ you cannot code it also ‘emotions: fear’. If both are mentioned in close proximity, you need to create two quotations and code them separately. You can however apply codes from different semantic domains to one quotation. You find more information on **multi-valued coding** in the section “Rules for Applying Codes”.
Semantic Domains are Context Dependent

At times it might be obvious from the domain name what the context is. The above sub code ‘supporting each other’ belongs to the context ‘benefits of friendship’ and it is not about work colleagues supporting each other. If the context is still unclear and could be interpreted in different ways, you need to make it unambiguous in the code definition.

Semantic Domains need to be Conceptually Independent

Conceptual independence means:
- a sub code from one domain is only specific for this domain. It does not occur in any other domain.
- thus, each code only occurs once in the code system

Therefore, as semantic domains are logically or conceptually independent from each other, it is possible to apply codes from different semantic domains to the same or overlapping quotations. For instance, in a section of your data, you may find something on the benefits of friendship and some emotional aspects that are mentioned. As these codes come from different semantic domains (BENEFITS and EMOTIONS), they can both be applied.

Developing a Code System with Semantic Domains

Semantic domains can be developed deductively or inductively. In most studies applying a qualitative data analysis approach, development is likely to be inductive. This means, you develop the codes while you read the data step by step. For example, in an interview study about friendship, you may have coded some data segments ‘caring for each other’, ‘supporting each other’ and ‘improve health and longevity’. Then you realize that these codes can be summarized on a higher level as ‘BENEFITS OF FRIENDSHIP’. Thus, you set up a semantic domain BENEFITS with the sub codes:
- benefits: caring for each other
- benefits: supporting each other
- benefits: improve health and longevity

You continue coding and come across other segments that you code ‘learning from each other’. As this also fits the domain BENEFITS, you add it to the domain by naming it:
- benefits: learning from each other

And so on. This way you can build semantic domains step by step inductively while developing the code system.

Once the code system is ready and the code definitions are written in the code comment fields, you can prepare the project for inter-coder agreement testing. At this stage, you can no longer expand a semantic domain. See “How to Set Up a Project for ICA Analysis”.

When developing a code system, the aim is to cover the variability in the data so that no aspect that is relevant for the research question is left out. This is referred to as exhaustiveness. On the domain level this means that all main topics are covered. On the sub code level, this means that the codes of a semantic domain cover all aspects of the domain and the data can be sorted in the available codes without forcing them. An easy way out is to include a catch all ‘miscellaneous’ code for each domain into which coders can add all data that they think does not fit anywhere else. However, keep in mind that such catch all codes will contribute little to answering the research questions.

Rules for Applying Codes

- codes from one domain need to be applied in a mutually exclusive manner.
- codes from multiple semantic domains can be applied to the same or overlapping data segments.

Mutual exclusiveness: You can only apply one of the sub codes of a semantic domain to a quotation or to overlapping quotations. Using the same code colour for all codes of a semantic domain will help you to detect possible errors.

If you find that you have coded a quotation with two codes from the same semantic domain, you can fix it by splitting the quotation. This means, you change the length of the original quotation and create one new quotation, so you can apply the two codes to two distinct quotations. See Figure 1: How to correct a violation of mutual exclusiveness.
Requirements for Coding

If codes within a semantic domain are not applied in a mutually exclusive manner, the cu-alpha coefficient cannot be calculated.

Multi-Valued Coding: This means that you can apply multiple codes of different semantic domains to the same quotation. For instance, a respondent talks about anger in dealing with her mother and mentions that she had to adjust expectations, this can be coded with codes from the three semantic domains EMOTIONS; ACTOR and STRATEGIES.

Coding with codes from multiple semantic domains will allow you to see how the various semantic domains are related for other analyses than inter-coder agreement, the example between opinions on recent events, people involved and type of consequences. For this you can use the code co-occurrence tools. You find information on the Code Co-Occurrence Tools in the full manual.

Instructions for Coders (Summary Form)

An important requirement for inter-coder agreement analysis is the independence of the coders. Thus, the person who has developed the code system cannot be one of the coders whose coding goes into the ICA analysis. In addition to the principal investigator who develops the code system, two or more persons are needed who apply the codes.

The coders will receive a project bundle file from the principle investigator (see “How to Set Up a Project for ICA Analysis”). It is recommended to provide the coders with the following instruction:
- When importing the project that you receive from the principal investigator, add your name or initials to the project name.
- After opening the project, double check your user name (see “User Accounts”).
- Apply all codes of a semantic domain in a mutual exclusive manner.
- If fitting you can apply codes from multiple semantic domains to the same or overlapping quotations.
- Do not make any changes to the codes - do not alter the code definition, do not change the code name or the code color.
- If you do not understand a code definition, or find a code label not fitting, create a memo and name it 'Comments from coder name'. Write all of your comments, issues you find and ideas you have into this memo.
- Do not consult with other coders. It is important that your coding remains unbiased and the data is coded by all coders independently.
- Once you are done coding, create a project bundle file and send it back to the principal investigator.

How Agreement/Disagreement is Measured

The coefficients that are available measure the extent of agreement or disagreement of different coders. Based on this measure one can infer reliability, but the coefficients do not measure reliability directly. Therefore: what is measured is inter-coder agreement and not inter-coder reliability.
Agreement is what we measure; reliability is what we infer from it.

Suppose two coders are given a text and code it following the provided instructions. After they are done, they give you the following record of their coding:

- There is agreement on the first pair of segments in terms of length and location (agreement: 10 units).
- On the second pair they agree in length but not in location. There is agreement on an intersection of the data (agreement: 12 units, disagreement: 2 + 2 units).
- In the third pair, coder A finds a segment relevant that is not recognized by coder B (disagreement: 4 units).
- The largest disagreement is observed in the last pair of segments. Coder A takes a narrower view than coder B (agreement: 4 units, disagreement: 6 units).

In terms of your ATLAS.ti coding, you need to think of your document as a textual continuum. Each character of your text is a unit of analysis for ICA, starting at character 1 and ending for instance at character 17500. For audio and video documents, the unit of analysis is a second. Images can currently not be used in an ICA analysis.

The quotation itself does not go into the analysis, but the characters or seconds that have been coded. In other words, if each coder creates quotations, it is not a total disagreement if they have not highlighted the exact same segment. The part of the quotations that overlap go into the analysis as agreement; the other parts as disagreement.

Another option is to work with pre-defined quotations. So that coders only need to apply the codes. For more information see “How to Set Up a Project for ICA Analysis”.

You can use the inter-coder agreement tool for text, audio and video data.

Methods for Testing ICA

ATLAS.ti currently offers three methods to test inter-coder agreement:
- Simple percent agreement
- Holsti Index
- Two of the Krippendorff's family of alpha coefficients.

All methods can be used for two or more coders. For scientific reporting, we recommend the Krippendorff alpha coefficients.

Percent Agreement

Percentage Agreement is the simplest measure of inter-coder agreement. It is calculated as the number of times a set of ratings are the same, divided by the total number of units of observation that are rated, multiplied by 100.

The benefits of percentage agreement are that it is simple to calculate and it can be used with any type of measurement scale. Let’s take a look at the following example: There are ten segments of text and two coders only needed to decide whether a code applies or does not apply:
HOW AGREEMENT/DISAGREEMENT IS MEASURED

<table>
<thead>
<tr>
<th>Segments</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Coder 2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Percent Agreement (PA) = number of agreements / total number of segments

PA = 6 / 10
PA = 0.6 = 60%

Coder 1 and 2 agree 6 out of 10 times, so percent agreement is 60%. One could argue this is quite good. This calculation, however, does not account for chance agreement between ratings. If the two coders were not to read the data and would just randomly code the 10 segments, we would expect them to agree a certain percentage of the time by chance alone. The question is: How much higher is the 60% agreement over the agreement that would occur by chance? Below only the results are presented if chance agreement is taken into account. If you are interested in the calculation, take a look at Krippendorff (2004, p. 224-226). The agreement that is expected by mere chance is (9.6 + 1.6) / 20 = 56%. The 60% agreement thus is not impressive at all. Statistically speaking, the performance of the two coders is equivalent to having reliably coded only 1 of the 10 segments, and have arbitrarily assigned 0s and 1s to the other 9 segments.

Holsti Index

Holsti’s method (1969) is a variation of percentage agreement, as percent agreement cannot be used if the coders have not all coded the same data segments. When coders were allowed to create their own quotations and did not code pre-defined quotations, the Holsti index needs to be user. Please note, that also the Holsti index does not take into account chance agreement.

The formula for the Holsti Index is:

\[ PA\ (\text{Holsti}) = \frac{2A}{N_1+N_2} \]

PA (Holsti) represents percentage of agreement between two coders.

A is the number of the two coders’ consensus decisions, and \( N_1 \) and \( N_2 \) are numbers of decisions the coders have made respectively.

Percentage agreement and the Holsti Index are equal when all coders code the same units of sample.

Cohens Kappa

ATLAS.ti does not offer a calculation for Cohen’s Kappa, because of severe limitations. Cohen’s kappa is a modification of Scott’s pi and according to Krippendorff (2019) and Zwick (1988) a rather unfortunate one because of a conceptual flaw in its calculation. Unlike the more familiar contingency matrices, which tabulate N pairs of values and maintain reference to the two coders, coincidence matrices tabulate then pairable values used in coding, regardless of who contributed them. In effect treating coders as interchangeable, Cohen’s kappa, by contrast, defines expected agreement in terms of contingencies, as the agreement that would be expected if coders were statistically independent of each other. Cohen’s conception of chance fails to include disagreements between coders’ individual predilections for particular categories, punishes coders who agree on their use of categories, and rewards those who do not agree with higher kappa-values. This is the cause of other noted oddities of kappa. The statistical independence of coders is only marginally related to the statistical independence of the units coded and the values assigned to them. Cohen’s kappa, by ignoring crucial disagreements, can become deceptively large when the reliability of coding data is to be assessed.

In addition, both Cohen’s kappa and Scott’s P assume an infinite sample size. Krippendorff’s alpha coefficient in comparison is sensitive to different sample sizes and can also be used on small samples.
Krippendorff’s Family Of Alpha Coefficients

The family of alpha coefficients offers various measurement that allow you to carry out calculations at different levels:

- **C-alpha (binary)**
  - At the most general level, you can measure whether different coders identify the same sections in the data to be relevant for the topics of interest represented by codes.
  - You can, but do not need to use semantic domains at this level. It is also possible to enter single codes per domain. You get a value for alpha binary for each code or semantic domain in the analysis, and a summary value for all items in the analysis.

  - All text units are taken into account for this analysis, coded and uncoded matter.

  - If you work with pre-defined quotations, the binary coefficient will be 1 for a semantic domain if only codes of the same semantic domain have been applied, regardless of which codes within the domain.

  - The summary alpha binary is always 1 when working with pre-defined quotations, as all coded segments are the same.

- **CU-alpha (binary)**
  - Another option is to test whether different coders were able to distinguish between the codes of a semantic domain. For example, if you have a semantic domain called 'type of emotions' with the sub codes:
    - emotions::contentment
    - emotions::excitement
    - emotions::embarrassment
    - emotions::relief
  - The coefficient gives you an indication whether the coders were able to reliably distinguish between for instance 'contentment' and 'excitement', or between 'embarrassment' and 'relief'. The cu-alpha will give you a value for the overall performance of the semantic domain. It will however not tell you which of the sub codes might be problematic. You need to look at the quotations and check where the confusion is.

  - The cu-alpha coefficient can only be calculated if the codes of a semantic domain have been applied in a mutually exclusive manner. This means only one of the sub codes per domain is applied to a given quotation. See "Requirements for Coding".

Figure 4: Krippendorff’s alpha family – from the general to the specific
HOW AGREEMENT/DISAGREEMENT IS MEASURED

**CU-ALPHA**

Cu-alpha is the summary coefficient for all cu-alphas. It takes into account that you can apply codes from multiple semantic domains to the same or overlapping quotations. See the information on multivalued coding in “Requirements for Coding”. Thus Cu-alpha is not just the average of all cu-alphas.

If codes of a semantic domain A have been applied to data segments that are coded with codes of a semantic domain B, this does not affect the cu-alpha coefficient for either domain A or B, but it affects the overall Cu-alpha coefficient. You can interpret the Cu-alpha coefficient as indicating the extent to which coders agree on the presence or absence of the semantic domains in the analysis. Formulated as a question: Could coders reliably identify that data segments belong to a specific semantic domain, or did the various coders applied codes from various other semantic domains?

In the calculation for both the cu- and Cu-alpha coefficient, only coded data segments are included in the analysis.

**CSU-ALPHA**

This coefficient also belongs to the family of alpha coefficients, but it is not yet implemented. Once implemented, it will allow you to drill down a level deeper and you can check for each semantic domain which code within the domain performs well or not so well. It indicates the agreement on coding within a semantic domain.

For example, if you have a domain ‘type of emotions’ with the sub codes:
- emotions::contentment
- emotions::excitement
- emotions::embarrassment
- emotions::relief

Coders may not code ‘contentment’ and ‘excitement’ consistently. You may have a cu-alpha coefficient for the entire domain of 0.76, which is satisfactory (see “Acceptable Levels of Reliability”). But if you look inside the semantic domain, the coefficients for ‘contentment’ and ‘excitement’ might not be so high. Thus, the Cs(u)u-alpha coefficient allows you to look inside a semantic domain.


**Calculation of Krippendorff’s Alpha**

According to Krippendorff (2019), the inter-coder agreement can be calculated as follows:

\[
\alpha = 1 - \frac{D_o}{D_e}
\]

whereby De is the expected disagreement by chance.

- \(\alpha = 1\) indicates perfect reliability.
- \(\alpha = 0\) indicates the absence of reliability. The units and the values assigned to them are statistically unrelated.
- \(\alpha < 0\) when disagreements are systematic and exceed what can be expected by chance.

When coders pay no attention to the text, which means they just apply the codes in an arbitrary manner, their coding has no relationship to what the text is about, then the observed disagreement is 1. In other words, the observed disagreement (Do) equals the maximum expected disagreement (De), which is 1. If we enter this in the formula, we get:

\[
\alpha = 1 - 1/1 = 0.000.
\]

If on the other hand, agreement is perfect, which means observed disagreement (Do) = 0, then we get:

\[
\alpha = 1 - 0/expected\ disagreement = 1.000.
\]

To better understand the relationship between actual, observed and expected agreement, let’s look a the following contingency table:
HOW AGREEMENT/DISAGREEMENT IS MEASURED

The tables represent a semantic domain with five codes. The matrices with perfect and expected agreement/disagreement serve as a benchmark what could have been if agreement either would have been perfect or if coders had applied the codes randomly. There are two matrices for expected agreement/disagreement indicating that statistically speaking there are many possibilities for a random attribution of the codes.

The black numbers show agreement, the red numbers disagreement.

- In the contingency table for observed agreement, we can see that the coders agree in applying code 4. There are no red numbers, thus no disagreement.
- There is some disagreement about the application of code 2. Agreement is 38% percent. In 2 % of the cases, code 1 was applied instead of code 2.
- There is a bit more confusion regarding the application of code 5. The coders have also applied code 1 and code 3. If this were a real life example, you would take a look at the code definitions for code 5 and code 2 and ask yourself why coders had difficulties to apply these codes in a reliable manner. The coefficient, however, is acceptable (see “Acceptable Levels of Reliability”). There is no need for re-testing.

For a full mathematical explanation on how the alpha coefficients are calculated see the Appendix written by Prof. Krippendorff and Wikipedia.

Common Mistakes

A common belief is that consensus is better than individual judgment. Rather than to work independently, coders are encouraged to discuss what they code and to reach their decision by compromise. However, data generated in this way does not ensure reproducibility, nor does it reveal the extent of it. The results often reflect the social structure of the group, the most prestigious member of the group dominating the outcome. Reproducibility requires at least two independent coders.

Another common procedure is to allow coders to consult each other if problems arise, e.g. they do not understand a coding instruction, or they have problems applying some of the codes to the data given to them. This also compromises reliability. Ideally, the coding instructions should be clear and easy to understand. If this is not the case, the coders involved in the discussion create their own interpretation of what the codes mean. This is difficult to communicate to others and therefore jeopardizes reproducibility. In addition, the process loses stability as the data coded in the early phases were coded based on a different understanding of the codes. If coders do not work independently and discuss problems during the process of coding, the reliability coefficient might be higher in the end, but this is partly illusory. If the data were to be given to different coders not having the same insights, reliability is likely to be lower again. As it is common that code systems evolve, and code definitions and coding rules need to be refined, you can ask coders to write down all the issues they see with the current instructions and definitions. Based on their notes, the coding system can be refined and tested again, but with different coders.

Another common mistake is to assume that it is best to ask other experts, colleagues with a long history of involvement in the subject of the research, or close friends and associates to serve as coders. Those coders are likely to agree, but not because they carefully follow the coding instructions, but because they know each other and the purpose of the research. This also results in higher measures of reliability, but also does not serve reproducibility.

Sample Size

The data you use for the ICA analysis needs to be representative of the total amount of data you have collected, thus of those data whose reliability is in question. Furthermore, the number of codings per code needs to be sufficiently high. As a
The phenomenon of agreement/disagreement is measured by the proportion of codings per code that yield at least five agreements by chance. Krippendorff (2019) uses Bloch and Kraemer’s formula 3.7 (1989:276) to obtain the required sample size. The table below lists the sample size for the three smallest acceptable reliabilities \( \alpha_{\text{min}}: 0.667 / 0.800 / 0.900; \) for four levels of statistical significance: 0.100 / 0.050 / 0.010 / 0.005, and for semantic domains up to 10 codes (probability \( p_c \)).

<table>
<thead>
<tr>
<th>Smallest acceptable ( \alpha )</th>
<th>.667</th>
<th>.800</th>
<th>.900</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Significance level</strong></td>
<td>.100</td>
<td>.050</td>
<td>.010</td>
</tr>
<tr>
<td>10 codes or ( p_c = 1.00 )</td>
<td>104*</td>
<td>172</td>
<td>344</td>
</tr>
<tr>
<td>9 codes or ( p_c = 1.11 )</td>
<td>95</td>
<td>136</td>
<td>312</td>
</tr>
<tr>
<td>8 codes or ( p_c = 1.25 )</td>
<td>85</td>
<td>141</td>
<td>281</td>
</tr>
<tr>
<td>7 codes or ( p_c = 1.43 )</td>
<td>76</td>
<td>125</td>
<td>251</td>
</tr>
<tr>
<td>6 codes or ( p_c = 1.67 )</td>
<td>67</td>
<td>110</td>
<td>220</td>
</tr>
<tr>
<td>5 codes or ( p_c = 0.20 )</td>
<td>58</td>
<td>95</td>
<td>190</td>
</tr>
<tr>
<td>4 codes or ( p_c = 0.25 )</td>
<td>49</td>
<td>81</td>
<td>161</td>
</tr>
<tr>
<td>3 codes or ( p_c = 0.33 )</td>
<td>41</td>
<td>67</td>
<td>135</td>
</tr>
<tr>
<td>2 codes or ( p_c = 0.50 )</td>
<td>36</td>
<td>60</td>
<td>119</td>
</tr>
</tbody>
</table>

\*number of codings for the semantic domain after merging

Table 1: Required sample size (adapted from Krippendorff, 2018)

### Example:
If you have a semantic domain with 4 codes and each of the codes are equally distributed, you need a minimum of 139 codings for this semantic domain if the minimum alpha should be 0.800 at a 0.05 level of statistical significance. For a lower alpha of 0.667, you need a minimum of 81 codings at the same level of statistical significance.

The formula for calculating the sample size is as follows:

\[
N_c = 2 \times \frac{Z^2 \left( (1 + \alpha_{\text{min}})(3 - \alpha_{\text{min}}) \right)}{4(1 - \alpha_{\text{min}})P_c(1 - P_c) - \alpha_{\text{min}}}
\]

Figure 6: Calculating sample size if your semantic domains have more than 10 codes and you work with 2 coders

It applies to 2 coders and for semantic domains up to 10 sub codes.

- By 4 codes in a semantic domain, the estimated proportion \( p_c \) of all values \( c \) in the population is 0.250 (1/4), by 5 codes, it is 0.20 (1/5), by 6 codes 0.1667 (1/6) and so on. If the distribution of your codes in a semantic domain is unequal, you need to make a new estimate for the sample size by using a \( p_c \) in the formula shown in Figure 6 that is correspondingly less than 1/4, 1/5, 1/6 and so on.
- If your semantic domain has more than 10 codes, you can calculate the needed sample size by adjusting the value for \( p_c \) as well.

You can see the corresponding \( z \) value from a standard normal distribution table. For a \( p \)-value of 0.05, \( z = 1.65 \).

The equation to estimate the needed sample size applies when working with 2 coders. If you work with more coders, you need to increase the sample size accordingly.

### Acceptable Levels of Reliability

The last question to consider is when to accept or reject coded data based on the ICA coefficient. Krippendorff’s (2019) recommendations are:

- Strive to reach \( \alpha = 1.000 \), even if this is just an ideal.
- Do not accept data with reliability values below \( \alpha = 0.667 \).
- In most cases, you can consider a semantic domain to be reliable if \( \alpha \geq 0.800 \).
- Select at least 0.05 as the statistical level of significance to minimize the risk of a Type 1 error, which is to accept data as reliable if this is not the case.

Which cut-off point you choose always depends on the validity requirements that are placed on the research results. If the result of your analysis affects or even jeopardizes someone’s life, you should use more stringent criteria. A cut-off point of \( \alpha = 0.800 \) means that 80% of the data is coded to a degree better than chance. If you are satisfied with \( \alpha = 0.500 \), this means 50% of your data is coded to a degree better than chance. The sound a bit like tossing a coin and playing heads or tails.

---

ATLAS.ti 8 Windows – Inter-Coder Agreement Analysis
How to Set Up a Project for ICA Analysis

When you prepare sub projects for the various coders, do not put them into ‘Inter-coder Mode’. This first needs to be done, when you merge the sub projects of the coders.

Summary of the steps

• The principal investigator (PI) develops the code system providing clear definitions for each code.
• The PI creates a snapshot project from his or her project (see “Creating Snapshots”) that either contains only documents and the code system, or documents, pre-defined quotations, and the code system. See “Removing Codings”.
• The PI creates a project bundle file of the snapshot project for the coders. See “Exporting the Project for Distribution”.
• The coders import the project bundle file, rename the project file during the process of import. See “Importing the Master Project and Renaming It”.
• The coders double-check under which user name they are logged in. See “User Accounts”.
• The coders code the data independently. If they have questions or difficulties applying the codes, they write notes into a memo.
• Once the coders are done, they prepare a project bundle file for the PI. See “Exporting Sub-Projects for The PI”.
• The PI imports the project bundle files from each coder and briefly checks the work of each coder and reads the memo they have written (if any).
• The principal investigator starts the inter-coder agreement mode and merges the projects of the coders. See “Merging Projects”.
• The principal investigator runs the ICA analysis. See “Running the ICA Analysis”.

Below you find technical instructions for both the principal investigator and the coders for what they need to know in terms of project management.

Project Management for the Principal Investigator (PI)

Creating Snapshots

For an ICA analysis it is important that you create snapshot projects for the coders in order to make sure that all coders work on projects that contain documents and codes with identical IDs.

To create a snapshot, select **File/ Snapshot**.
HOW TO SET UP A PROJECT FOR ICA ANALYSIS

Change the default name to indicate that this is a sub project for one of the coders and click CREATE SNAPSHOT.

Just like ATLAS.ti project files, snapshots are saved internally in the ATLAS.ti environment. A snapshot projects also has the same ID as the project from which it is created.

Open the snapshot project and prepare it for the coders by removing documents, and quotations or only the codings.

Removing Codings

If you want to create a sub project for coders that contains pre-defined quotations:

- Open the Document Manager and highlight all documents.
- Click on the TOOLS tab and from there select REMOVE CODINGS.

Your project now contains documents, pre-defined quotations and codes. The codes show a frequency of 0. All codings has been removed from all quotations. Thus, all quotations are uncoded.

Exporting the Project for Distribution

To export the project for distribution to all team member, select FILE / EXPORT.

Click on the PROJECT BUNDLE button. This opens the Windows File Manager. Select a location for storing the project bundle. The default project bundle name consists of the project name plus the author and date in parentheses behind it, e.g., 'Project X_sub project for ICA (PI name 2019-09-27).'

Distribute the project bundle file to all coders, e.g. via email, a folder on a server that everyone can access, or also via a cloud link or cloud folder.

The project bundle file has the file extension ".atlproj" and can be read by both ATLAS.ti 8.x Windows and Mac.
HOW TO SET UP A PROJECT FOR ICA ANALYSIS

Project bundle files can be shared via a cloud service like Dropbox, OneDrive, Google Drive, etc. Another option is to upload the project bundle file to a server of your choice and send a link to all team members so that they can download the file.

The default name of the bundle is the project name + author and date. Renaming the bundle does not change the name of the project contained within the bundle. Think of the project bundle file as a box that contains your ATLAS.ti project plus all documents that you want to analyze. Putting a different label on the outside of the box will not change the name of the project file, which is contained inside the box.

**Merging Projects**

The Merge Tool reunites projects that were originally divided for analytical or economical reasons. It brings together the contributions of the coders. In the following a few general principles about merging are explained:

**Master And Imported Projects:** The Master project is the project into which another project, called the "Import project" is merged. The Master project has to be loaded first before invoking the **Merge Project** option.

**Merge Strategy**

The default option is to unify all objects that are **identical** and to add all objects that do not yet exist in the Master project. This is why it is so important to start from a common Master project, create sub projects from it and distribute those to the coders.

**Identical Entities**

When an entity is created in ATLAS.ti—regardless if it is a document, a code, a quotation, a memo, a network, a group or a comment--this entity receives a unique ID, comparable to a fingerprint. When merging projects, ATLAS.ti compares the IDs of the various entities. If they have the same ID (fingerprint), they are unified. If the fingerprint is different, they are added.

**As coders do not add codes when serving as coders for an ICA analysis, there should never be duplicated codes!**

Identical entities are unified, all others are added.

**Entities with and without comments:** If there is a code C in the Master project that has no comment, and a code C in the Import project that has a comment, the merged procedure will add this comment to the merged project.

In the case of an unsolvable conflict (e.g., code C in the Master project has a comment, and code C in the Import project also has a comment) the user can define which of the two conflicting entities will win. If the comment of the Master project should be kept, you need to select the option **"Keep."** If the comment of the Import project should be kept, you need to select the option **"Override,"** as shown in Figure 12.
Coders should be instructed not to change code comments. If they want to write notes, they should create a memo and use it as writing space. If you merge a project for ICA analysis, you should never encounter a merge conflict related to codes!

**How to Merge**

Before you merge, the recommendation is that you import all project bundle files first and take a look at the projects. This way you can see whether the coders have done their work as instructed.

- Import the project bundle files of code 1 and 2.
- Open the project of coder 1 and create a snapshot. Do not accept the default name, enter a name that indicates that this is the project for the ICA test for coder 1 and 2.
- Close the project of coder 1 and continue to work with the renamed snapshot.

Select **Analyze / Enable Intercoder Mode**.

Select **File / Merge**.

Select the **Merge Project** option and select the project of coder 2.
Deactivate the option to create a snapshot as you are already working with a snapshot.

Click on the **Merge** button.

ATLAS.ti checks the two projects for identical and different items. All identical items will be merged, all different items will be added. See "Merge Strategy". In ICA mode, the codings of the different coders are added. This way, you can see which coder has applied which code to a quotation and ATLAS.ti can compare the codings in the further analysis.

After this process is completed, you see a **Pre-Merge Summary**:

If the project has been set-up as described above and the coders have followed the instructions, there should be no merge conflict.

Click on **Merge**.

After merging, check the final merge report and the merged project for plausibility. If you are satisfied with the results, save the project. If not, you can always select **Undo**.

If you work with more than two coders, continue with merging the next sub-project.
Project Management for Coders

User Accounts

ATLAS.ti automatically creates a user account based on the name that you use on your computer.

- If you want to check under which name you are working, select the **Tools & Support** tab and then **User Manager**.
- To create a new user account, select **User Manager**. Next, click on the button **New User** at the left-hand side in the ribbon.
- After creating a new account, select **Switch User** to log in using a different user name.

![User Management tab in ATLAS.ti 8 Windows](image)

A reason for creating a new user account is if two persons on a team have the same name. When merging projects that come from different computers and that use the same user name, the user name will be duplicated (e.g. Tom and Tom (2)). This means you can still distinguish the two users. However, it may be a bit cumbersome to always remember who is who.

Importing the Master Project and Renaming It

- Open ATLAS.ti and select the **Import Project Bundle** button. If another project is currently open, select **File/New** and then the Import Project Bundle option.

During the process of importing the project, the name of the project can be changed:

![Renaming a project file during the process of importing a project bundle](image)

- Rename the project by adding your name or initials to the project name. This is important for the project administrator later when he or she merges the projects of all team members (see Figure 10).
- Click **Import**.

Renaming Projects On the Opening Screen

If you want to rename a project after it is imported, you can either do it on the opening screen when starting ATLAS.ti. If the project is already open, you need to close it to return to the opening screen.
On the opening screen, right click on the project that you want to rename and select **Rename Project**.

![Renaming a project on the opening screen](image)

**Exporting Sub-Projects for The PI**

After the coders are done coding the data, they create their own project bundle file and send it to the PI.

To export the project, select **File / Export**. See Figure 9. The name of the exported bundle file will include the name of the currently logged in user and the date, e.g. *Project X_sub project for ICA_coder 2 (Tom 2017-09-30)*.

It is recommended that all team members save a copy of the project bundle file on their computer, or their personal space on a server.

**Merging Projects for ICA Analysis**

It is recommended to import the project bundle files of the coders first and to check what the coders have done and whether they followed the instructions.

- Import the project bundle files of all coders.
- Open the project of coder 1 and create a snapshot. Do not accept the default name, enter a name that indicates that this is the project for the ICA test for coder 1 and 2.
- Close the project of coder 1 and continue to work with the renamed snapshot.

![Enable Inter-coder Mode](image)

Select **Analyze / Enable Intercoder Mode**.

Next, merge the project of coder 2 and save the merged project. Deactivate the option to create a snapshot as you already merged into a snapshot of the project.
In ICA mode, the ‘codings’ of all coders will become visible. A coding is the link between a code and a quotation. In ICA mode, you can see which coder has applied which code to a quotation.

To see the authors of the various codings, change the view settings from icons to user in the View tab of the contextual Document ribbon:

Each coder is indicated by a different color. Hover over the user icon to see the name of the user. In Figure 21, all quotations where created by the blue user Susanne. The brown user was asked to apply the codes of the code system to these predefined quotations.
Running the ICA Analysis

- Click on the **InterCoder Agreement** button in the **Analyze** tab.
- Click on the **Add Coder** button and select two or more coders.
- Click on the **Add Documents** button and select the documents that should be included in the analysis. The default option is for all selected documents to be regarded as one continuum and all coded quotations of these documents will go into the calculation. You can, however, also view agreements / disagreements per document and get a coefficient for each document.

Only the documents are shown that have been coded by the selected coders.

- Click on the **Add Semantic Domain** button and add one or more codes to the domain. Alternatively, you can drag and drop codes from the Project Explorer or the code list into the semantic domain field:

![Figure 22: Adding codes to a semantic domain](image)

Repeat this process for each semantic domain.

As there currently is no function to define a semantic domain formerly, all codes that you are adding from the list will be considered to belong to a semantic domain. As you can see in Figure 22, semantic domains were already built indirectly in the process of developing the code system by adding the same prefix to all codes that belong to the same semantic domain like ACTIVITIES or BENEFITS. Please remember that all sub codes of a semantic domain (i.e., all codes with prefixes) must be applied to quotations in a mutual exclusive manner. See “Requirements for Coding”.

You can also add single codes to semantic domains. You can then however not calculate the cu-alpha coefficient as it is specific to semantic domains. See also Methods for Testing ICA.

After you have added coders, documents and codes, your screen looks similar to what is shown in Figure 23.
The quotations of the semantic domain are displayed on the right-hand side. When you select a quotation, the full content is shown in the preview and you can write comments. You also see where the quotation is located in the document continuum. When you scroll to the right in the quotation view window, you see the codes that have been applied by the selected coders.

If you click on one of the lines, only the quotations of the selected code and coder are shown in the quotation window (Figure 24).

You can remove coders, codes or documents from the analysis by clicking on the red x.

To the right of each coder line you see some descriptive statistics:

The blue coder (Susanne) has applied the code “activities clubbing / going for a drink” 11 times. The number of characters that are coded with this code are 1331 out of a total of 172,414 characters in all three selected documents, which is 0.8%.

The brown coder (Sabine) has applied the code “activities clubbing / going for a drink” 8 times. The number of characters that are coded with this code are 1057 out of a total of 172,414 characters in all three selected documents, which is 0.6%.

Calculating an ICA Coefficient

To calculate ICA, select one of the four methods from the ribbon.
See “Methods for Testing ICA” for more detail.

In Figure 27, the c-alpha binary for the semantic domain 'activities' is 0.983 and for the semantic domain 'benefits' is 0.937. For both domains, it is 0.949. This means that the reliability for identifying relevant from irrelevant matter related to the codes of these domain is high.

The results for cu-alpha and Cu-alpha for the two selected domains are shown in Figure 28.
RUNNING THE ICA ANALYSIS

The difference between the two measures is that the lower case cu-alpha applies to one semantic domain only, the capital Cu-alpha applies to all selected semantic domains. It is not simply an average of all lower case cu-alphas, as sub codes from different domains can potentially be applied to one quotations. This is referred to as multi-valued coding (see Krippendorff, et al., 2016) and the section on “Rules for Applying Codes” in this manual.

Exporting Results

You can either print the document continuums that you see on your screen or save them in PDF format. Another option is to export the quantitative results including the coefficients to Excel.

- To print the document continuums, click on the Print button in the ribbon. To save it as PDF file, select a PDF printer.
- To export the results of an ICA analysis to Excel, click on the Excel Export button in the ribbon.

The Excel report contains the following:

- **First line**: project name
- **Second line**: Type of coefficient, if one was selected.
- **Legend**: Explains the column headers of the table
- Followed by **coders** and **documents** used in the analysis
- Above each table the **semantic domain** and its codes are listed that go into the analysis.
- The table contains the **descriptive statistics** for each code and coder, and if a **coefficient** was selected, the resulting number.

If you have chosen the detailed view for each document in the ICA tool, the table contains the analysis for all documents plus the analysis for each individual document.

![ICA Excel report](image)

**Figure 29**: ICA Excel report
References


MacPhail, Catherine, Khoza, Nomhle, Abler, Laurie and Ranganathan, Meghna (2015). *Process guidelines for establishing Intercoder Reliability in qualitative studies*. Qualitative Research, 16 (2), 198-212.


