

Discourse Network Analyzer Manual

Philip Leifeld, Johannes Gruber and Felix Rolf Bossner

Last update: DNA 2.0 beta 21 with rDNA 2.1.0 on May 21, 2018.

Contents

1	Introduction	1
	PHILIP LEIFELD AND JOHANNES GRUBER	
2	Methods for Network Construction	3
	PHILIP LEIFELD	
2.1	Graphical Intuition	3
2.2	Notation	5
2.3	Construction of One-Mode Networks	6
2.4	Normalisation for One-Mode Networks	8
2.5	Affiliation Networks	9
2.6	Normalisation of Affiliation Networks	9
2.7	Temporal Aggregation: Time Windows and Attenuation	10
3	Installation of DNA and rDNA	11
	JOHANNES GRUBER AND PHILIP LEIFELD	
3.1	Windows	12
3.2	macOS	14
3.3	Linux	16
3.4	Installing DNA and rDNA	18
4	Preparation of your DNA Workspace	22
	FELIX ROLF BOSSNER AND JOHANNES GRUBER	
4.1	Creating a new DNA Database	22
4.2	User Management: Multiple Coders and Permissions	26
4.3	Statement Types and Variables	29
4.4	Final Step: Approving your Workspace and Creating the DNA File	33
5	Importing and Organizing your Raw Data	35

5.1	Opening an Existing DNA Database	35
5.2	Importing Documents (Raw Data)	35
5.3	Organizing Documents (Raw Data)	43
6	Coding the Data	49
	JOHANNES GRUBER	
6.1	Creating a DNA Statement	49
6.2	Navigating Through Statements	54
6.3	Editing a Statements	54
6.4	Using the Regex Highlighter and Search Function	57
7	Exporting the Coded Data	59
	JOHANNES GRUBER	
7.1	Type of Network	59
7.2	Statement Type	63
7.3	File Format	63
7.4	Variable 1 and 2	64
7.5	Qualifier and Qualifier Aggregation	66
7.6	Normalization	72
7.7	Duplicates	73
7.8	Time Series Options	74
7.9	Exclude from Variable	76
7.10	Isolates	76
8	rDNA: Using DNA from R	78
	PHILIP LEIFELD AND JOHANNES GRUBER	
8.1	Getting Started with rDNA	78
8.2	Retrieving Networks and Attributes	81
8.3	Cluster analysis	87
8.4	Heatmaps	95
8.5	Multi-dimensional scaling	98
8.6	Network plots	104
8.7	Adding and manipulating documents in DNA from R	108

Chapter 1

Introduction

PHILIP LEIFELD AND JOHANNES GRUBER

This manual demonstrates how to install, set up, and use the open-source standalone software **Discourse Network Analyzer** (DNA) and its companion R package **rDNA** (Leifeld and Gruber 2018), which are designed for researchers using the method *discourse network analysis*.¹ By combining content analysis and dynamic network analysis, this method can reveal the structure and dynamics of policy debates. The method comprises three basic steps:

1. annotating statements of actors in unstructured (text) sources,
2. creating networks from the resulting structured data,
3. analysing and interpreting the results by employing the toolbox of network analysis.

The results can take a number of different forms, such as so-called congruence or conflict networks of actors or of concepts, affiliation networks of actors and concepts, and longitudinal versions of these networks (see Chapter 2 and Leifeld 2017 for a comprehensive overview of the method).

The benefit of using the **Java** software DNA is that it is specifically designed to aid the user in the first two of these basic steps of discourse network analysis. It is mainly designed for qualitative annotation of actors' statements in order to structure the text. The program can also create different kinds of network matrices based on these structured data and export them to other programs for further analysis and plotting. Additionally, while the software is primarily designed for discourse network analysis of actors and concepts, it is also flexible with regard to the definition of new statement types, for example using user-defined variables like "location" or "addressee" (see Section 4.3). While there are numerous alternative software packages for qualitative content analysis, there are very few which were specifically developed with discourse network analysis in mind, and therefore they lack the functionality necessary for exporting network data.

The companion package **rDNA** for the statistical computing environment **R** additionally helps with the third step mentioned above: analysis of the annotated statements. **rDNA** takes the structured data from DNA and permits further in-depth analysis using network analysis. While data can also be exported to other software such as **Ucinet**, **visone**, **NetMiner**, and **Gephi**, **R** is the preferred choice as it facilitates reproducible research, is free and open source, and has a large community of users and developers who are engaged in all kinds of data analysis tasks. **R** has several packages developed specifically for

¹This manual is a work in progress and will be continuously updated during the year 2018. See <https://github.com/leifeld/dna/blob/master/manual/> for the most recent version.

network analysis, such as `statnet` (Handcock et al. 2008), `igraph` (Csardi and Nepusz 2006), `xergm` (Leifeld et al. 2018, 2017), `sna` (Butts 2016), `network` (Butts 2008b), `ggraph` (Lin Pedersen 2017a), and `tidygraph` (Lin Pedersen 2017b). Most of these packages work seamlessly with data processed by `rDNA` and therefore add a myriad of possibilities to the native functions of our own R package.

In recent years, discourse network analysis has been employed by a growing number of scholars in a wide field of policy sectors, such as pension politics (Leifeld 2013, 2016), climate politics (Fisher et al. 2013a,b; Broadbent and Vaughter 2014; Gkiouzevas and Botetzagias 2015; Manfredo et al. 2014; Schneider and Ollmann 2014; Stoddart and Tindall 2015; Wagner and Payne 2017; Yun et al. 2014), software patents and property rights (Leifeld and Haunss 2012), internet policy (Breindl 2013; Haunss and Kohlmorgen 2009), infrastructure projects (Nagel 2016), energy policy (Brutschin 2013; Haunss et al. 2017; Imbert 2017; Rinscheid 2015), shooting rampages (Hurka and Nebel 2013), abortion (Muller 2014a,b, 2015), outdoor sports (Stoddart et al. 2015), deforestation (Rantala and Di Gregorio 2014), higher education (Nägler 2015), international financial politics (Haunss 2017), and online deception (Wu and Zhou 2015).

While a default toolbox of methods is available for discourse network analysis, new methods are being developed presently. For example, one promising approach is the application of inferential network models to the temporal network structure produced by `DNA` in order to model policy debates at the micro level. This will soon allow us to develop and test theories on how actors contribute statements to policy debates, and to forecast debates based on these theories (for an outlook, see Leifeld 2017).

The outline of this manual is as follows. Chapter 2 describes the types of networks `DNA` can export. Chapter 3 explains how to install `DNA` and `rDNA`, which both rely on a correctly configured Java runtime environment. Only consult this chapter if you experience problems with the installation on your own. The following four sections describe the usage of `DNA` in detail: Chapter 4 describes how to set up a project in `DNA`, including the creation of a database, adding and managing users, and how to set up or edit statement types and variables. Chapter 5 explains how you can import and organise your raw data (i. e., documents). Chapter 6 provides an overview of how you annotate statements in `DNA`. Even though this process is very straightforward, the section also reveals some functions that can help you to annotate material faster and more reliably. Chapter 7 explains how data can be exported to other programs for further analysis. What may have seemed abstract in Chapter 2 quickly becomes clear at this point—once you have exported a few example networks yourself. Chapter 8 is an introductory tutorial on using the `rDNA` package to perform additional analysis and plotting tasks using the infrastructure provided by R.

Both `DNA` and `rDNA` can be downloaded from [GitHub](#) (see Chapter 3). Please feel free to post questions and bug reports to the issue tracker on [GitHub](#).

Chapter 2

Methods for Network Construction

PHILIP LEIFELD

This chapter summarises the main network algorithms implemented in DNA graphically and using mathematical notation.

2.1 Graphical Intuition

Figure 2.1 illustrates how actors (as yellow nodes on the left) and concepts (as blue nodes on the right) are connected by dashed lines. These dashed lines represent the edges of a bipartite graph, also called an affiliation network. Substantively, these edges represent statements that were annotated in a policy debate. For example, actor 5 refers to concepts 3 and 4 in the debate.

Based on this bipartite graph, an actor congruence network and a concept congruence network can be inferred. For example, actors 1 and 2 jointly refer to the same concept 1, hence they are directly connected by an edge in the actor congruence network illustrated on the left. If actors 1 and 2 shared more than one concept, their edge weight would be proportional to the number of concepts they shared. Substantively, the strength of connection between two actors can be interpreted as their similarity in terms of the concepts they employ in the policy debate.

Conversely, concepts 1 and 3 are jointly referred to by the same actor 2, hence they are directly connected by an edge in the concept congruence network illustrated on the right. If concepts 1 and 3 were jointly referred to by more than one actor, their similarity would be greater than one. Substantively, the edge weights between concepts can be interpreted as their similarity in terms of the actors that employ them in the policy debate.

Actors and concepts are merely a substantive application where other variables 1 and 2 could have been encoded instead, such as persons and locations or speakers and addressees.

Simply modelling referral of a concept by an actor, however, is insufficient to capture agreement and opposition in policy debates. For example, actor 1 and actor 2 may either both support concept 1, they may both reject concept 1, or one of them may refer to concept 1 in a positive way while the other one may refer to concept 1 in a negative way. Depending on these configurations, one would infer a congruent or a conflictual relationship between actors 1 and 2.

Figure 2.2 illustrates how two types of networks can be generated, congruence and conflict networks. In a congruence network, edges are counted when both actors co-support or co-reject a concept. In a conflict network, edges are counted when the two actors' agreement patterns differ.

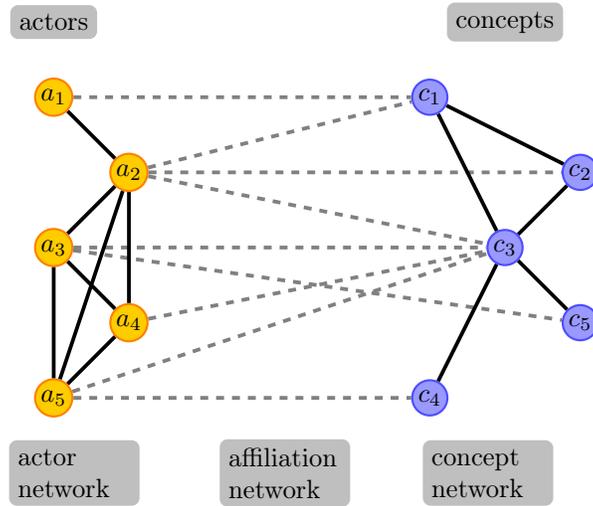


Figure 2.1: Illustration: Affiliation network (dashed lines) between actors (yellow nodes, variable 1) and concepts (blue nodes, variable 2) and their induced actor congruence network (solid lines on the left) and concept congruence network (solid lines on the right).

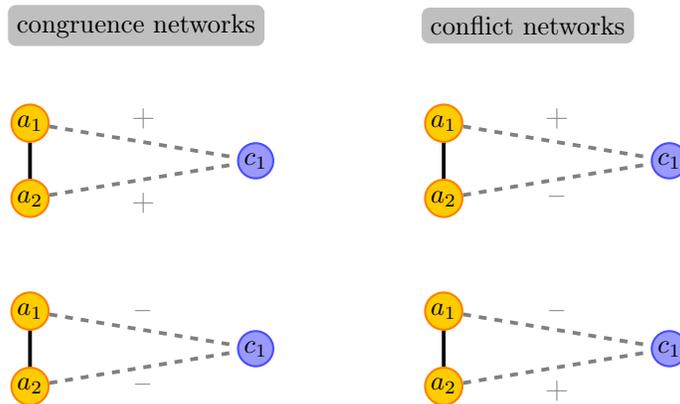


Figure 2.2: Illustration: congruence and conflict networks with a binary qualifier variable.

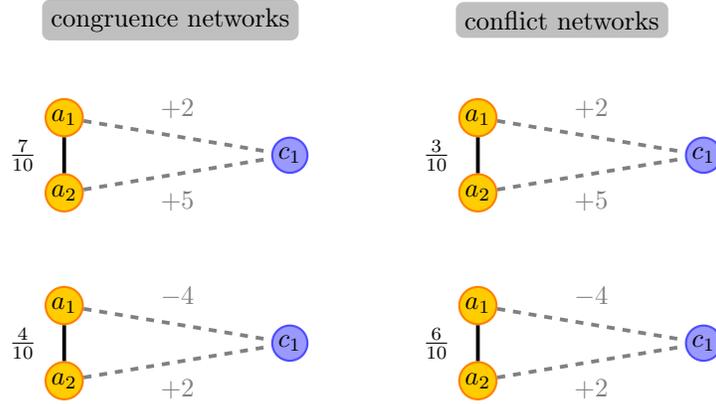


Figure 2.3: Illustration: congruence and conflict networks with an integer qualifier variable.

Qualifiers like “agreement” need not be binary (e. g., positive or negative). They could be represented by integer weights, such as intensity of agreement on a scale from -5 to $+5$. In this case, one would need to define the edge weights between nodes in a congruence network as the absolute difference between the two weights in the affiliation network subtracted from the maximum possible difference and divided by the maximum possible difference. Conversely, in a conflict network, one would need to define the edge weights between nodes as the absolute difference between the two weights in the affiliation network divided by the maximum possible difference, such that a value of 0 represents no conflict and 1 represents maximal conflict. In either case, these fractions would need to be counted over all concepts (or, more generally, over all nodes of the second variable). This calculation is illustrated in Figure 2.3.

Finally, it may be necessary to normalise the resulting affiliation, congruence, or conflict network. Normalisation may be necessary to avoid a core–periphery structure where those actors end up at the center of the network who refer to most concepts. Normalisation corrects for the verbosity of actors (or, more generally, for the centrality of a node in the affiliation network). Several normalisation methods are available, and they will be described below. Graph clustering can then be applied to the normalised networks to identify coalitions in policy debates. More details, especially on the topic of normalisation, can be found in Leifeld (2017).

The next section will introduce some formal notation to represent the data structures and transformations introduced above; then these transformations will be re-introduced more formally using mathematical notation, and finally normalisation methods will be proposed.

2.2 Notation

X is a three-dimensional array representing statement counts. x_{ijk} is a specific count value in this array, with the first index i denoting an instance of the first variable (e. g., organization or actor i), the second index j denoting an instance of the second variable (e. g., concept j), and the third index k denoting a level on the qualifier variable (e. g., agreement = 1). For example, $x_{ijk} = 5$ could mean that actor i mentions concept j with intensity k five times. X can be represented as a cuboid, as illustrated in Figure 2.4.

Where the qualifier variable is binary, *false* values are represented as 0 and *true* values as 1 on the k index, i. e., $K^{\text{binary}} = \{0; 1\}$. Where the qualifier variable is integer, the respective integer value is

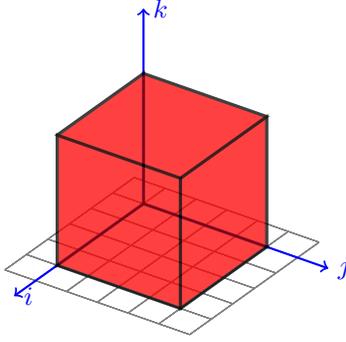


Figure 2.4: Statements in the X array can be represented in a cuboid data structure.

used as the level. This implies that k can take positive or negative values or 0, i.e., $K^{\text{integer}} \subseteq \mathbb{Z}$. Note that all k levels of the scale are included in K , not just those values that are empirically observed.

Indices with a prime denote a second instance of an element, e.g., i' may denote another organization. Y denotes the output matrix to be obtained by applying a transformation to X . Several transformations are possible and will be described below.

2.3 Construction of One-Mode Networks

2.3.1 Congruence Networks

In a congruence network, the edge weight between nodes i and i' represents the number of times they co-support or co-reject second-variable nodes (if a binary qualifier is used) or the cumulative similarity between i and i' over their assessments of second-variable nodes (in the case of an integer qualifier variable).

In the integer case, the similarity between nodes i and i' is defined as the cumulative similarity over levels k of the qualifier variable:

$$y_{ii'}^{\text{congruence}} = \Phi_{ii'} \left(\sum_{j=1}^n \sum_k \sum_{k'} x_{ijk} x_{i'jk'} \left(1 - \frac{|k - k'|}{|K| - 1} \right) \right) \quad (2.1)$$

where $\Phi_{ii'}(\cdot)$ denotes a normalization function (to be specified below).

Here, $|k - k'|$ is the difference in assessment of second-mode node k (e.g., concept) by two first-mode nodes i and i' . $|K| - 1$ is the maximum difference there can be, with $|K|$ indicating the number of levels in qualifier variable K . For example, if the qualifier scale is $[-5; 5]$, $|K| - 1 = 10$. The subtraction in the parentheses serves to convert distances (as in a conflict network) to similarities. The distances are counted over all statements, meaning that nodes i and i' count these similarities over all combinations of the levels k (for node i) and k' (for node i') for each second-variable node j (e.g., each concept) and weight them by how often these combinations occur. This weighting occurs in the $x_{ijk} x_{i'jk'}$ part. For example, if i mentions j at intensity -4 twice and i' mentions j at intensity $+2$ three times on an intensity scale $[-5; +5]$, this contributes $2 \cdot 3 \cdot \left(1 - \frac{|-4-2|}{11-1} \right) = 4.2$ to the edge weight between i and i' in the congruence network.

The binary case with $|K| = 2$ is a special case of the integer congruence network with a negative or positive agreement pattern, for example reflecting rejection or support of a concept by an actor. In

the binary case, congruent opinions always reduce to $1 - \frac{|k-k'|}{|K|-1} = 1$, and differences in opinion always reduce to $1 - \frac{|k-k'|}{|K|-1} = 0$. Hence the binary case can be more easily expressed by counting the matches on the k qualifier for all j items without computing any distances:

$$y_{ii'}^{\text{congruence binary}} = \Phi_{ii'} \left(\sum_{j=1}^n \sum_k x_{ijk} x_{i'jk} \right). \quad (2.2)$$

2.3.2 Conflict Networks

The same logic as for the congruence network can be applied to produce conflict networks. In the integer case, Equation 2.1 must be modified such that the relative distances are not subtracted from one, while everything else stays the same:

$$y_{ii'}^{\text{conflict}} = \Phi_{ii'} \left(\sum_{j=1}^n \sum_k \sum_{k'} x_{ijk} x_{i'jk'} \left(\frac{|k-k'|}{|K|-1} \right) \right) \quad (2.3)$$

In the binary case, Equation 2.2 must be modified such that contradictions instead of matches are counted. In other words, instead of counting $x_{ijk} x_{i'jk}$, $x_{ijk} x_{i',j,(1-k)}$ must be counted:

$$y_{ii'}^{\text{conflict binary}} = \Phi_{ii'} \left(\sum_{j=1}^n \sum_k x_{ijk} x_{i',j,(1-k)} \right). \quad (2.4)$$

2.3.3 The Subtract Method

In many empirical applications, it might make sense to combine the notions of congruence and conflict in a single signed and weighted network. If only congruence is considered, for example, one misses out on the possible fact that two actors may contradict each other on more concepts than they agree on. For this reason, it might make sense to subtract conflict edge weights from congruence edge weights and thereby construct a signed, weighted graph using the *subtract* method as follows:

$$y_{ii'}^{\text{subtract}} = y_{ii'}^{\text{congruence}} - y_{ii'}^{\text{conflict}} \quad (2.5)$$

Here, positive $y_{ii'}^{\text{subtract}}$ values indicate congruence in excess of conflict while negative values indicate conflict in excess of congruence. In some practical applications—for example, for visualisations of the congruence network—it may make sense to discard all negative values or introduce some other threshold value c for recoding all $y_{ii'}^{\text{subtract}} < c$ values as 0.

2.3.4 The Ignore Method

In some applications, qualifiers do not matter substantively, or there is only one level on the qualifier variable. In such applications, it is possible to just count all referrals of j by i across levels of k to get the number of times i mentions j in any way, then do the same for i' , and multiply both to yield the similarity between i and i' in terms of overlap in j , disregarding the levels of k :

$$y_{ii'}^{\text{ignore}} = \Phi_{ii'} \left(\sum_{j=1}^n \left(\left(\sum_k x_{ijk} \right) \left(\sum_k x_{i'jk} \right) \right) \right) \quad (2.6)$$

2.4 Normalisation for One-Mode Networks

Leifeld (2017) discusses the normalisation of congruence networks. Normalisation, however, is also possible for affiliation networks, as will be demonstrated below.

Normalisation can be necessary to correct networks for the activity or popularity of nodes. For example, if some first-variable nodes refer to a substantial number of second-variable nodes while others refer to few, the former will be more likely to be connected to many other nodes and especially those with similar levels of activity, which leads to a core–periphery structure of the discourse network. Normalisation corrects for this pattern by cancelling out the effect of activity or popularity of nodes. This will often lead to a clear cluster structure based on the similarity of node profiles, instead of a core–periphery structure.

In the simplest case, normalization can be switched off, in which case

$$\Phi_{ii'}^{\text{no}}(\omega) = \omega. \quad (2.7)$$

2.4.1 Average Activity Normalisation of One-Mode Networks

Edge weights can be divided by the *average activity* of nodes i and i' :

$$\Phi_{ii'}^{\text{avg}}(\omega) = \frac{\omega}{\frac{1}{2} \left(\sum_{j=1}^n \sum_k x_{ijk} + \sum_{j=1}^n \sum_k x_{i'jk} \right)}. \quad (2.8)$$

Average activity normalisation is the most commonly applied form of normalisation and works both with binary and weighted X arrays, i. e., with or without duplicate statements. It divides each weight by the mean of the number of second-variable referrals of nodes i and i' .

2.4.2 Jaccard Normalisation for One-Mode Networks

With *Jaccard normalisation*, we do not just count i 's and i' 's activity and sum them up independently, but we add up both their independent activities and their joint activity, i. e., both matches and non-matches:

$$\Phi_{ii'}^{\text{Jaccard}}(\omega) = \frac{\omega}{\sum_{j=1}^n \sum_k x_{ijk} [x_{i'jk} = 0] + \sum_{j=1}^n \sum_k x_{i'jk} [x_{ijk} = 0] + \sum_{j=1}^n \sum_k x_{ijk} x_{i'jk}}. \quad (2.9)$$

Jaccard normalisation works best with binary X arrays, i. e., if duplicate statements are not possible in the data structure.

2.4.3 Cosine Normalisation for One-Mode Networks

With *cosine normalization*, we modify Equation 2.8 to take the product in the denominator instead of the mean:

$$\Phi_{ii'}^{\text{cosine}}(\omega) = \frac{\omega}{\sqrt{\left(\sum_{j=1}^n \sum_k x_{ijk}\right)^2} \sqrt{\left(\sum_{j=1}^n \sum_k x_{i'jk}\right)^2}}. \quad (2.10)$$

This works best when duplicates are admitted but can also be applied to binary X arrays.

2.5 Affiliation Networks

While one-mode networks as portrayed in Section 2.3 are most useful for analysing coalition structure in policy debates, affiliation networks convey more complexity. This makes them harder to interpret with increasing complexity of the data but can be more informative for less complex discourse networks.

The simplest case is to ignore the qualifier variable:

$$y_{ij}^{\text{affiliation ignore}} = \Phi_{ij} \left(\sum_k x_{ijk} \right) \quad (2.11)$$

This only makes sense if there is only one level in K or if the qualifier variable does not matter substantively.

More interestingly, negative edges (e. g., rejection of concepts by actors) can be subtracted from positive edges (e. g., support of concepts by actors). This yields a signed, weighted affiliation network.

In the integer case, the respective cells in X can just be weighted by the respective level k . If the weight is negative, this will subtract x_{ijk} from the count:

$$y_{ij}^{\text{affiliation subtract integer}} = \Phi_{ij} \left(\sum_k kx_{ijk} \right) \quad (2.12)$$

In the binary case (assuming $K = \{0; 1\}$), 0 values need to be transformed into -1 before they can be subtracted:

$$y_{ij}^{\text{affiliation subtract binary}} = \Phi_{ij} \left(\sum_k (kx_{ijk} - (1-k)x_{ijk}) \right) \quad (2.13)$$

Alternatively, in the binary case (assuming $K = \{0; 1\}$), it is possible to map all combinations of k for each (ij) dyad into a multiplex network with three distinct types of edges, where 0 represents neither agreement nor disagreement, 1 represents agreement, 2 represents disagreement, and 3 represents a mix of both agreement and disagreement. This can be useful, for example, for visualising agreement, disagreement, and ambiguity/ambivalence in the same affiliation network using different colours. More formally:

$$y_{ij}^{\text{affiliation combine binary}} = \begin{cases} 0 & \text{if } \sum_k x_{ijk} = 0 \\ 1 & \text{if } x_{i,j,k=0} = 0 \wedge x_{i,j,k=1} > 0 \\ 2 & \text{if } x_{i,j,k=1} = 0 \wedge x_{i,j,k=0} > 0 \\ 3 & \text{if } x_{i,j,k=0} > 0 \wedge x_{i,j,k=1} > 0 \end{cases} \quad (2.14)$$

2.6 Normalisation of Affiliation Networks

Like one-mode networks, affiliation networks can be normalised. With *activity normalisation*, ties from more active nodes receive lower weights:

$$\Phi_{ij}^{\text{activity}}(\omega) = \frac{\omega}{\sum_{j=1}^n \sum_k x_{ijk}} \quad (2.15)$$

With *prominence normalisation*, ties to more prominent nodes receive lower weights:

$$\Phi_{ij}^{\text{prominence}}(\omega) = \frac{\omega}{\sum_{i=1}^m \sum_k x_{ijk}} \quad (2.16)$$

2.7 Temporal Aggregation: Time Windows and Attenuation

Networks can be temporally smoothed. For example, it is possible to create a series of temporally overlapping time slices and aggregate these slices into a single network to limit the temporal scope of congruence edges (*time window algorithm*). Using the same algorithm, it is possible to visualise change over time using animations. Or it is possible to make the edge weight proportional to the time that has passed between the relevant statements of i and i' (*attenuation algorithm*). These methods are more advanced and are introduced in [Leifeld \(2016\)](#).

Chapter 3

Installation of DNA and rDNA

JOHANNES GRUBER AND PHILIP LEIFELD

This section explains how DNA and rDNA can be installed on common desktop operating systems.

As DNA is written in Java, both DNA and rDNA rely on Java to work on your computer properly. Installing and configuring a valid Java Runtime Environment on your machine will thus be the first and only complicated step of the installation. Following the simple steps below, one should not run into problems while setting up Java. The advantage of the Java programming language for academic software is that it both runs on different operating systems without altering the source code, once the Runtime Environment is set up, and that it is, for the most part, open source. Besides setting up the Java Runtime Environment, the installation of DNA and rDNA is identical on different operating systems.

If you feel confident that Java is already correctly set up on your computer, you can therefore skip to Section 3.4 if you like. Otherwise please continue to the section for the operating system you wish to install DNA and rDNA on: [Windows](#), [macOS](#) or [Linux](#).

For more experienced users, here is a short version of the steps described below:

1. (On Mac: install [Apple's legacy version of Java](#)—even though we will never use it.)
2. Install Java Runtime Environment (JDK) (Version 8) on your computer.
3. (On Windows and Mac: set up the `JAVA_HOME` to the installation path of your JDK.)
4. Download the newest executable JAR from <https://github.com/leifeld/dna/releases>.
5. (On Linux: make the JAR file executable.)
(On Mac: allow executing apps from an unidentified developer.)
6. You can now run the standalone DNA or continue to install rDNA as well.
7. Download and install R (and RStudio).
8. In R: install the necessary R packages `rJava` and `devtools`.
9. In R: install rDNA via

```
devtools::install_github("leifeld/dna/rDNA", args = "--no-multiarch")
```

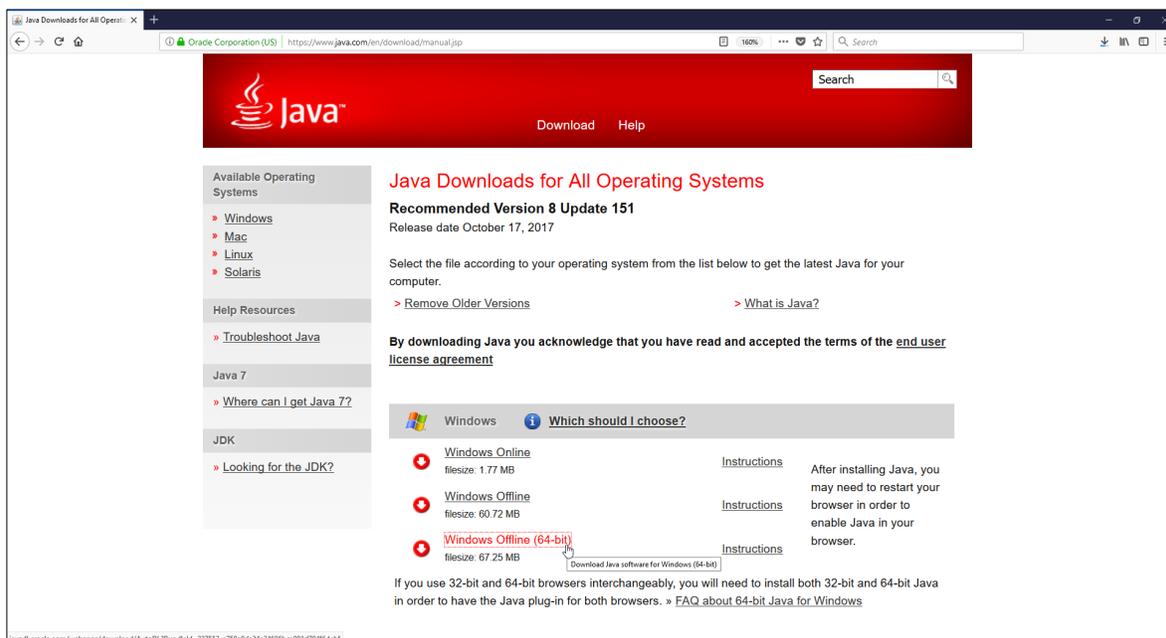


Figure 3.1: Downloading JDK from Oracle

3.1 Windows

3.1.1 Installing Java on Windows

To install the necessary Java Runtime Environment on your Windows computer, simply go to <https://www.java.com/en/download/manual.jsp>, scroll down to and download Windows Offline (64-bit) (see Figure 3.1; download Windows Offline instead if you are using a 32-bit version of Windows). During the installation, you can accept all the default options, including the installation path.

Next, you should set `JAVA_HOME` in your environmental variables to tell your Windows PC where your Java installation lives. This step is optional, but can prevent many issues with Java users had in the past. To set `JAVA_HOME`, you need to navigate to the menu edit the system environment variables. The easiest way to get there is to hit the  button on your keyboard and enter `environment`. Windows will then search for programs and settings menus that include this title and should usually display the menu we are looking for on top.¹ In this menu, you have to find the button `Environment variables...`. Clicking this button should open the window shown in Figure 3.2.

Under User Variables, click `New`.² Enter the variable name `JAVA_HOME` and the path to your Java installation in the field Variable value. If you have not altered the default installation location, you should find Java in `"C:\Program Files\Java\jre1.8.0_151"` or, if you chose to install a 32-bit version of Java, in `"C:\Program Files (x86)\Java\jre1.8.0_151"` (which will cause problems, though, if you try to use it with a 64-bit version of R).³

¹On older versions of Windows, this might not work. On Windows 7 you can alternatively right-click on My Computer and select Properties → Advanced. On Windows 8 Control Panel → System → Advanced System Settings.

²This sets `JAVA_HOME` just for the current user. If you want to make Java available for all users on the computer you are working on, you can create a System Variable instead.

³Note that you have to repeat this procedure whenever the installation path of Java changes, for example whenever Java is updated.

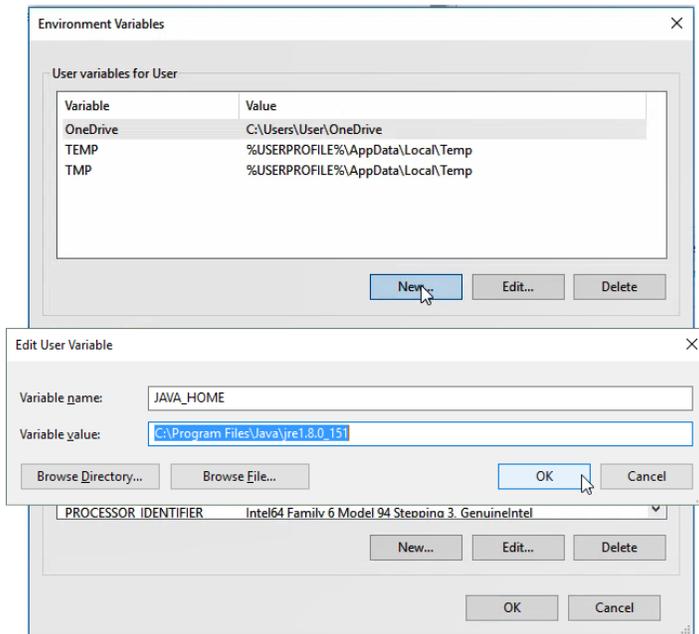


Figure 3.2: Edit JAVA_HOME to tell Windows where your Java lives.

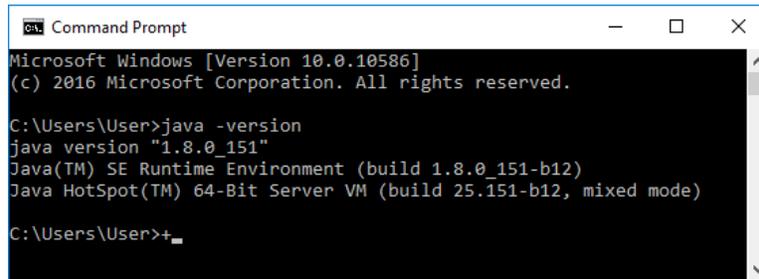


Figure 3.3: Testing Java installation in Windows command prompt

Windows should now recognise Java and be able to run Java commands. To test this, we can open the command prompt (press the  button on your keyboard and simply enter `cmd` and then hit `Enter`) and type a Java command, e.g., `java -version`. If the installation was successful, the output should display information about the Java-version and build as depicted in Figure 3.3.

After installing Java, you are ready to use DNA and could skip to Section 3.4 if you are not interested in installing rDNA as well. In order to use rDNA, the rest of this section will explain how to install R and the recommended **integrated development environment (IDE)** RStudio, which makes working with R a lot easier and also looks a lot better than the default interface.

3.1.2 Installing R on Windows

1. First, you need to download R from <https://cran.r-project.org/bin/windows/base/>.
2. At the top of the page, click on `Download R 3.5.0 for Windows` (or a newer version if available).

3. Install the downloaded file, e. g., `R-3.5.0-win.exe`. Usually, it is fine to leave all default settings in the installation options.
4. Go to <https://www.rstudio.com/products/rstudio/download/>.
5. At the bottom of the page, under `Installers for Supported Platforms`, click on the link `RStudio 1.1.453 - Windows Vista/7/8/10` (or a newer version if available). Again, the default installation options are fine in most cases and can be accepted without changes.
6. After installation, you can use R by opening RStudio.

3.1.3 Testing the Installation of RStudio

Traditionally, the first test you perform in a new programming language is to write a “Hello, World!” program. To do this in R, you simply type `print(“Hello World!”)` in the console (in the lower left corner of RStudio). Alternatively, you can make R perform a simple mathematical operation. If everything is set up correctly, the output should look like this:

```
print("Hello World!")

## [1] "Hello World!"

# You can also use R as a calculator
2 * 3

## [1] 6
```

The chunk of code above marks the first time we are using R commands in this manual. It might be worth explaining what this means for users who are not familiar with documents containing R code. Whenever code is shown in this manual it is decorated with a light grey background. Comments in R code (i. e., text targeted at the user to explain what is happening in a specific line) are marked with a # and are formatted in italic font and in dark grey. The output, which is generated by running a command, is marked by two # and formatted in black. This means that any line that does not start with ## contains R code you can copy and paste to the console in RStudio and run. Alternatively, you can also copy the code into an R script and execute it by either clicking on the  `Run` button in the upper right corner of the console in RStudio, or you can use the shortcut `Ctrl+Enter`. Either way, the highlighted code or the line in which the caret is currently flashing are sent to the console and executed. If this works fine, you should be able to continue to Section 3.4.

3.2 macOS

3.2.1 Installing Java on macOS

On macOS, you have to install two versions of Java in order for rDNA to work properly. The reasons behind this are too complicated to cover here. Basically, Apple built its own version of Java, which needs to be on your machine, even though it is outdated. Therefore we need to first install the legacy Java 6—which we will never use—before installing the correct Java Development Kit version 8.⁴

⁴If you do not wish to ever use rDNA or any other R package that relies on Java, you might not need both versions and can just download the newest Java Runtime Environment. However, installing Java version 8 before the legacy Java will cause problems if you ever change your mind.

Java SE Development Kit 8u162

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Solaris SPARC 64-bit (SVR4 package)	139.99 MB	jdk-8u161-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u161-solaris-sparcv9.tar.gz
Solaris x64	140.57 MB	jdk-8u161-solaris-x64.tar.Z
Solaris x64	97.02 MB	jdk-8u161-solaris-x64.tar.gz
Windows x86	198.54 MB	jdk-8u161-windows-i586.exe
Windows x64	206.51 MB	jdk-8u161-windows-x64.exe

Java SE Development Kit 8u161 Demos and Samples Downloads

You must accept the [Oracle BSD License](#) to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.93 MB	jdk-8u162-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u162-linux-arm64-vfp-hflt.tar.gz
Linux x86	169.01 MB	jdk-8u162-linux-i586.rpm
Linux x86	183.81 MB	jdk-8u162-linux-i586.tar.gz
Linux x64	166.13 MB	jdk-8u162-linux-x64.rpm
Linux x64	181.02 MB	jdk-8u162-linux-x64.tar.gz
macOS	247.12 MB	jdk-8u162-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.98 MB	jdk-8u162-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.3 MB	jdk-8u162-solaris-sparcv9.tar.gz
Solaris x64	140.68 MB	jdk-8u162-solaris-x64.tar.Z
Solaris x64	97.03 MB	jdk-8u162-solaris-x64.tar.gz
Windows x86	198.57 MB	jdk-8u162-windows-i586.exe
Windows x64	206.76 MB	jdk-8u162-windows-x64.exe

Java SE Development Kit 8u161 Demos and Samples Downloads

You must accept the [Oracle BSD License](#) to download this software.

Accept License Agreement Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	9.93 MB	jdk-8u161-linux-arm32-vfp-hflt-demos.tar.gz
Linux ARM 64 Hard Float ABI	9.96 MB	jdk-8u161-linux-arm64-vfp-hflt-demos.tar.gz

Figure 3.4: Downloading JDK from Oracle.

First, please download the file https://support.apple.com/downloads/DL1572/en_US/javaforox.dmg and install it, accepting all defaults. After this has finished, we can proceed to get the new version of the Java Development Kit. Go to <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html> and scroll down to Java SE Development Kit 8u162, accept the License Agreement and then click on [jdk-8u161-macosx-x64.dmg](#) to download the file (see Figure 3.4). Again, install the program accepting all defaults.

After installing Java, you are ready to use rDNA and could skip to Section 3.4 if you are not interested in installing rDNA as well. In order to use rDNA, the rest of this section will explain how to install R and the recommended **integrated development environment (IDE) RStudio**, which makes working with R a lot easier and also looks a lot better than R's default interface.

3.2.2 Installing R on macOS

1. First, you need to download R from <https://cran.r-project.org/bin/macosx/>.
2. At the top of the page, click on `R-3.5.0.pkg` (or a newer version if available).
3. Install the downloaded file. Usually, it is fine to leave all default settings in the installation options.
4. Go to <https://www.rstudio.com/products/rstudio/download/>.
5. At the bottom of the page, under Installers for Supported Platforms, click on the link `RStudio 1.1.453 - Mac OS X 10.6+ (64-bit)` (or a newer version if available). Install RStudio by simply dragging the application icon in the downloaded `.dmg` file to your Applications folder.
6. Then you need to install the program Xcode from the app store. The program is very large and will take a while to install.

7. After installation, you can use R by opening RStudio.

To test your installation of R, follow the instructions in Section 3.1.3.

Working with Java from within R on a Mac is a bit messy. Apple's own version of Java, although important to have installed, does not run in combination with R. That is why we have to tell your system which version of Java to use by default. To do this, we have to enter a few system commands, which you can either do in the Terminal app or directly from within R using the `system` function:

```
# list files in java_home
system("/usr/libexec/java_home -V")
##Matching Java Virtual Machines (3):
## 1.8.0_162, x86_64: "Java SE 8" /Library/Java/JavaVirtualMachines/jdk1.8.0...
## 1.6.0_65-b14-468, x86_64: "Java SE 6" /Library/Java/JavaVirtualMachines/...
## 1.6.0_65-b14-468, i386: "Java SE 6" /Library/Java/JavaVirtualMachines/1...

# see default version of Java
system("java -version")
##java version "1.8.0_162"
##Java(TM) SE Runtime Environment (build 1.8.0_162-b12)
##Java HotSpot(TM) 64-Bit Server VM (build 25.162-b12, mixed mode)
```

If your output looks like the output above, you are almost ready to install `rJava`. The only thing left to do is to associate Java with R. To do this, you can either use the terminal app, or you can invoke a system command directly from within R using the `system` function:

```
$sudo R CMD javareconf
```

Or in R:

```
system("sudo R CMD javareconf")
```

If `/usr/libexec/java_home -V` does not show `1.8.0_162` (or any other version starting with `1.8.`), you need to install Java version 8 again (see above) and possibly reboot your computer. If `java -version` shows `java version "1.6.0_65"`, but version 1.8 is listed in the output from the first command, you can set the default by executing the following command:

```
# Set JAVA_HOME
system("export JAVA_HOME=~/.usr/libexec/java_home -v 1.8")
```

After this, you should be able to continue to Section 3.4. However, depending on prior installations and the configuration of your machine, there can be other problems. You can find one nice tutorial and trouble-shooting guide [here](#).

3.3 Linux

3.3.1 Installing Java on Linux

Since you are using Linux, we assume that you are sufficiently comfortable with using the terminal.

First, check if Java might already be installed:

```
$java -version
```

If not, install it, e. g., via APT:

```
$sudo apt-get install default-jdk
```

After installing Java, you are ready to use rDNA and could skip to Section 3.4 if you are not interested in installing rDNA as well. In order to use rDNA, the rest of this section will explain how to install R and the recommended **integrated development environment (IDE) RStudio**, which makes working with R a lot easier and also looks a better than the default user interface.

3.3.2 Installing R on Linux

1. Since the version of R in the default repositories tends to be fairly outdated, we add the repository of the Comprehensive R Archive Network (CRAN) to our `sources.list`:

```
$sudo add-apt-repository \  
"deb [arch=amd64,i386] https://cran.rstudio.com/bin/linux/ubuntu \  
$(lsb_release -cs)/"
```

Note, that `lsb_release -a` automatically selects your flavour and version of Linux from the CRAN server. Visit **CRAN** to see for which Linux distributions R is available. `cran.rstudio.com` is also just one of several **CRAN mirrors**, so you could replace it with a different one if you prefer.

2. Next, you need to add R to your keyring. Here is how you would accomplish this in Ubuntu:

```
$sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E084DAB9
```

3. Update apt and install R (or `r-base-dev` if you wish to compile packages from source):

```
$sudo apt-get update  
$sudo apt-get install r-base
```

4. Now install RStudio via gdebi (and install gdebi first if you do not already have it):⁵,

```
$sudo apt-get install gdebi-core  
$wget https://download1.rstudio.org/rstudio-1.1.453-amd64.deb  
$sudo gdebi -n rstudio-RS_vers-amd64.deb  
$rm rstudio-RS_vers-amd64.deb
```

Note, that as of version 1.1.453, RStudio depends on an outdated version of `libgstreamer`. This version has already been deprecated in some linux distributions, which can lead to an error during installation of RStudio. If you run into trouble while installing RStudio, you should try installing the old version of `libgstreamer` side-by-side the newer library:

⁵Alternatively, you can download an installation file from <https://www.rstudio.com/products/rstudio/download/>.

```
# Download files with wget
$wget http://ftp.ca.debian.org/debian/pool/main/g/gstreamer0.10/libgstreamer\
0.10-0_0.10.36-1.5_amd64.deb
$wget http://ftp.ca.debian.org/debian/pool/main/g/gst-plugins-base0.10/libgs\
treamer-plugins-base0.10-0_0.10.36-2_amd64.deb

# Now install with gdebi
$sudo gdebi libgstreamer0.10-0_0.10.36-1.5_amd64.deb
$sudo gdebi libgstreamer-plugins-base0.10-0_0.10.36-2_amd64.deb

# And then clean up
$sudo rm libgstreamer0.10-0_0.10.36-1.5_amd64.deb libgstreamer-plugins-base0.10-
0_0.10.36-2_amd64.deb
```

5. For Linux, there are a few other system dependencies for `rDNA`. You should install these using:

```
$sudo apt-get install libudunits2-dev
$sudo apt-get build-dep libcurl4-gnutls-dev
$sudo apt-get install libcurl4-gnutls-dev
```

6. After the installation has finished, you can use R by opening RStudio.

To test your installation of R, follow the instructions in Section 3.1.3.

Before we can actually run `rDNA`, we need to associate Java with R. To do this, you should go back to the terminal:

```
$sudo apt-get install r-cran-rjava
$sudo R CMD javareconf
```

If this finishes without errors, you are ready to start installing `DNA` and the `rDNA` package as described in Section 3.4.

3.4 Installing `DNA` and `rDNA`

Once `Java` is set up correctly, you can simply download the latest version of `DNA` as a JAR file from <https://github.com/leifeld/dna/releases> (see Figure 3.5). JAR or `.jar` files are technically self-contained and executable archive files, which usually contain a computer program written in `Java`, along with all the files necessary to run the program. Once the download is finished, you can start the program by double-clicking on the downloaded file. However, on Linux, it is sometimes necessary to make the file executable first (e.g., via `$chmod +x /path/to/your/dna.jar` or using a GUI method). On newer version of macOS, a security exception needs to be made before you can run a program from an “unidentified developer” (i.e., if the program has not been registered with Apple). To do so for `DNA`, control-click the program’s icon, then choose `Open` from the shortcut menu. If clicking on the file does not open the program on a Windows machine, right-click on the `.jar` file → `Open with` → `Use another app` and then navigate to the file `"C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe"`.

If you are not interested in using `rDNA`, you can now skip to Chapter 4.

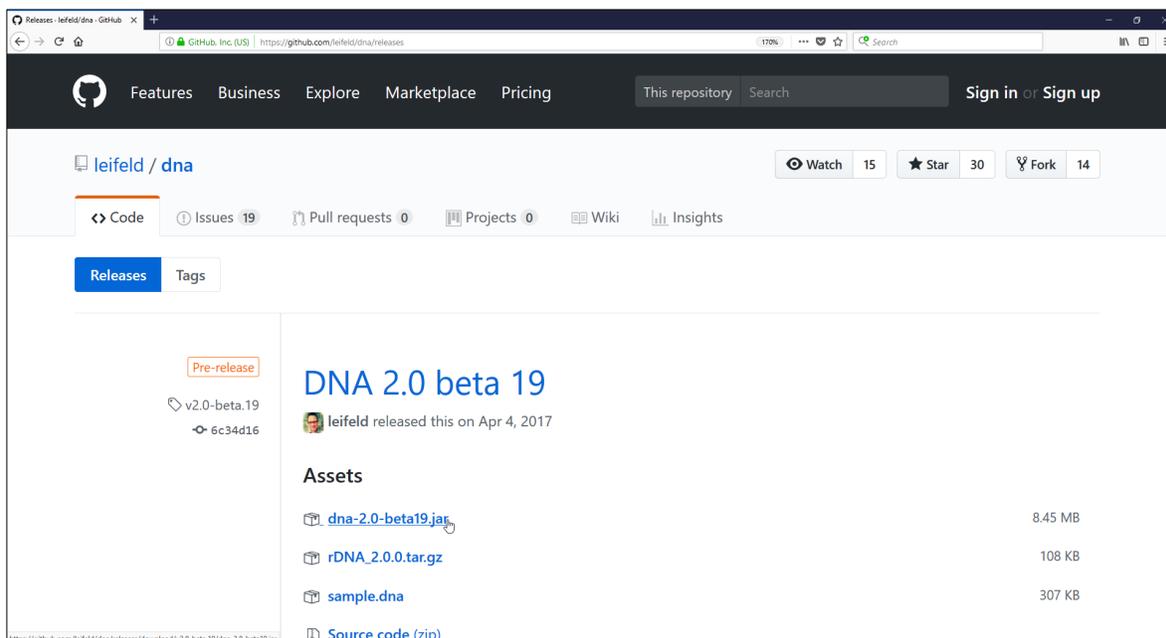


Figure 3.5: Download DNA jar file from GitHub releases page.

At this point, it is assumed that you have installed R and have at least a minimal understanding of how the program works (see Section 3.1.3). If that is the case, we can go ahead and install `rDNA` from within R.

First, we need to install the package `rJava` (Urbanek 2017), which is the most important dependency of `rDNA`:^{6,7}

```
install.packages("rJava")
```

To see if this worked, or to troubleshoot potential problems, we can run a few Java commands from within R:⁸

```
library("rJava")
# 1. initialize JVM
.jinit()

# 2. retrieve the Java-version
.jcall("java/lang/System", "S", "getProperty", "java.version")

## [1] "10.0.1"

# 3. retrieve JAVA_HOME location
.jcall("java/lang/System", "S", "getProperty", "java.home")
```

⁶Again this sometimes doesn't work that easily on macOS. If the installation fails, you could try to install the package from source using `install.packages("rJava", type="source")`.

⁷Alternatively, it can make sense on Linux systems to install `rJava` via apt: `sudo apt-get install r-cran-rjava`.

⁸Loading `rJava` for the first time regularly fails on macOS with the warning `...` . If this is the case, try the command `sudo ln -s $(/usr/libexec/java_home)/jre/lib/server/libjvm.dylib /usr/local/lib` in your terminal app.

```
## [1] "/usr/lib/jvm/java-11-openjdk-amd64"

# 4. retrieve Java architecture
.jcall("java/lang/System", "S", "getProperty", "sun.arch.data.model")

## [1] "64"

# 5. retrieve architecture of OS (This should have 64 in it if step 4 displays
#    "64")
.jcall("java/lang/System", "S", "getProperty", "os.arch")

## [1] "amd64"

# 6. retrieve architecture of R as well (This should again have 64 in it if
#    step 4 and 5 display 64)
R.Version()$arch

## [1] "x86_64"
```

For `rDNA` to work properly, you need to ensure that `rJava` works correctly. In particular, it is essential that the architectures of `Java`, your operating system, and your version of `R` match (see comments 4, 5, and 6 in the code chunk above).

Once this is done, you should install the package `devtools` (Wickham et al. 2018), which permits installing `R` packages from `GitHub`.

```
install.packages("devtools")
```

Since we only need one function from the package `devtools` at this point, it is not necessary to invoke the `library` command to load the whole package. Instead, you can write `devtools::` and then type the function you want to use.⁹

```
devtools::install_github("leifeld/dna/rDNA", args = "--no-multiarch")
```

After this is done as well, the final step of the installation is to test if `rDNA` can be loaded into `R` correctly and to perform a basic operation with it—opening `DNA` from within `R`. In order to do so, you first need to download `DNA`, which can also be done in `R` with the `dna_downloadJar` command (see Chapter 8 for more details on what these commands mean).

```
# download rDNA JAR
dna_downloadJar() # download DNA jar

# load library
library("rDNA")

# initialise the file you just downloaded
```

⁹The option `args = "--no-multiarch"` should normally not be necessary, but prevents errors on some operating systems. Since `devtools` tries to test both the 32-bit and 64-bit version of a package during installation, the process inevitably fails as only one architecture of `Java` is available.

```
dna_init()
```

```
# start up DNA from R with the sample file to see if everything worked  
dna_gui(infile = dna_sample())
```

If these commands can be executed correctly, you are ready to use both DNA and rDNA.

Chapter 4

Preparation of your DNA Workspace

FELIX ROLF BOSSNER AND JOHANNES GRUBER

After installing the program (see Chapter 3), you can now create your first DNA database for your own research project. How you set up a DNA database will mainly depend on the needs of your personal research design—which should usually be clear before you start analysing data. Therefore, DNA can be customised during the creation of a new database in accordance with how you are planning to use the tool.

4.1 Creating a new DNA Database

In order to create a new DNA database file, you have to click on the index tab `File` (in the upper left corner of your DNA program window) and select the option `New DNA database` (see Figure 4.1). As a result, a new window will open (see Figure 4.2), in which you find a menu that provides you with a step-by-step guidance for specifying the configuration of your personal DNA database

Clicking on the first tab in the sidebar of this menu—`Database` (see Figure 4.2)—opens a menu, which allows you to choose the file name and storage location of your database. For this first step of your set-up, DNA provides you with two options in respect to the type of database, in which your data is stored. Which of these options best fits your research project is dependent on the circumstances of your coding process:

Local .dna file The preset option `Local .dna file` means that the dataset is stored in a local file (technically an SQLite file) on your PC or device. This file, with the file extension `.dna`, can be moved on your machine, sent via email, uploaded and shared via a cloud file hosting service—such as Dropbox—and can generally be treated in the same way as any other file PC users are familiar with. A local `.dna` file will be sufficient in most user scenarios, for example, if you employ a single coder working on a single computer, if multiple coders work on a single dataset at non-overlapping intervals or when multiple coders work at the same time on different datasets, which you merge after the coding process (see Section 5.2.3). For most users, this simpler option will be adequate in order to use DNA. It is not necessary to be familiar with setting up and managing an SQLite or MySQL database. If you think the scenarios described above cover your intended use of DNA, you can now jump to the next section and start [Creating a Local DNA File](#).

Remote database on a server However, for more experienced user or research projects in which several coders want to work on the same database at the same time, a second option was included

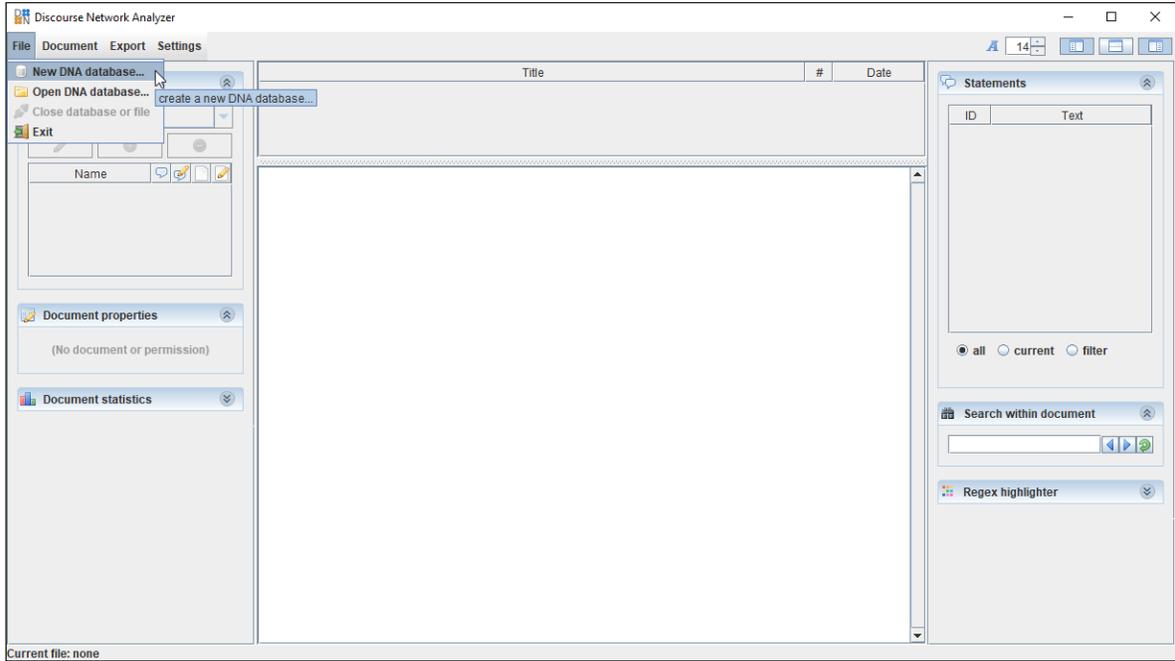


Figure 4.1: Starting a new Database

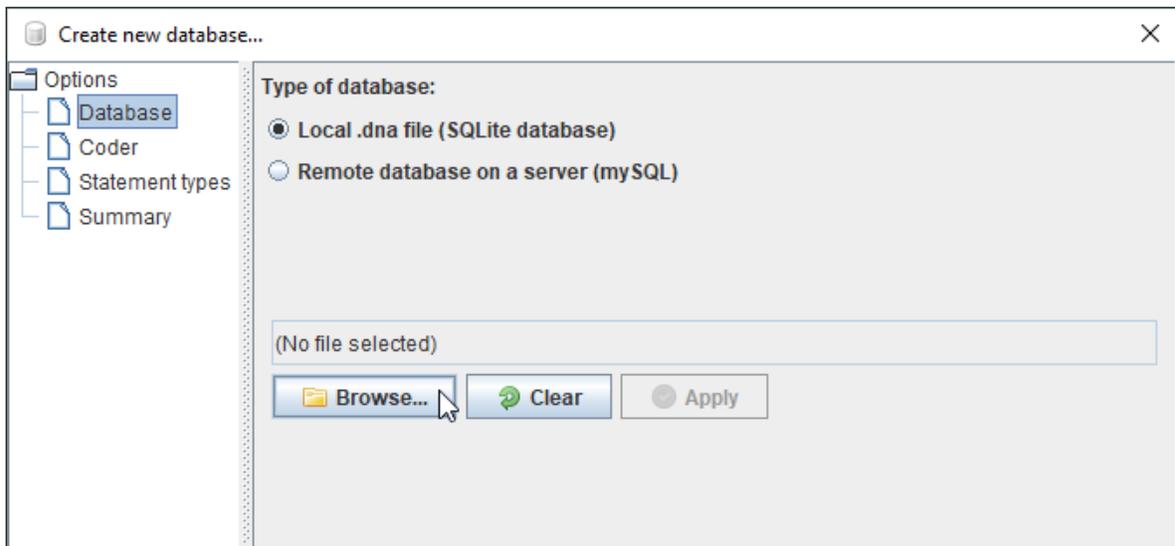


Figure 4.2: Choose if database will be stored locally or remotely

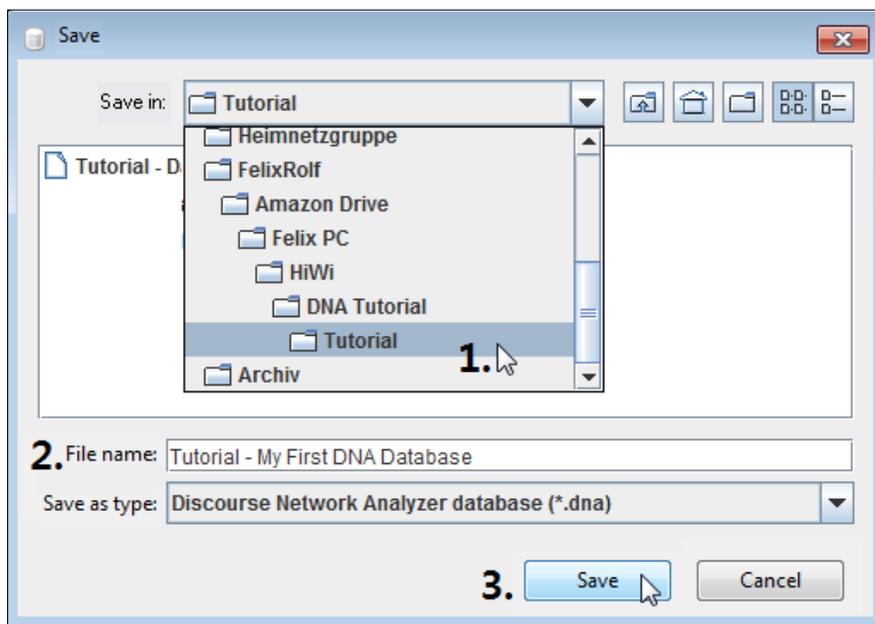


Figure 4.3: Choose location of database window

into `DNA Remote database on a server`. This stores your data in a MySQL database which could be stored locally on your machine—which would defy the purpose though—, on a private server—such as a Network-attached storage (NAS)—, or on an online cloud server. You should select this option if you employ a single coder working on multiple devices or multiple coders working on a single dataset at the same time. The preconditions for using this type of storage are that all coders have a stable connection to the database during the coding process—e. g., via the internet—and that you [set up an online MySQL database](#) in advance. If this is how you want to proceed, you can now jump directly to the section which describes the necessary steps for [Creating and Using a Remote Database \(MySQL\)](#).

4.1.1 Creating a Local DNA File

1. Click on the button `Browse` (see Figure 4.2). Now a pop-up menu—similar to the one shown in Figure 4.3—should be open.
2. In this pop-up menu, you can choose the storage location of your database on your local device from the `Save in` slide down menu. Enter the name of your database in the field `File Name` and confirm your choices by pressing the `Save` button (see Figure 4.3). Now the pop-up menu will close.
3. *Next, it is important, that you confirm your choices again by pressing the `Apply` button* (see Figure 4.4). If you forget to press this button, you cannot create the database in the final step, because the program will report “No database selected” (see Figure 4.14).

If you just employ a single coder and don’t want to change or supplement the preset standard research variables (`person`, `organization`, `concept`, `agreement`) or types of codeable statements (`Statement`, `Annotation`), you can now proceed directly to the [final step](#). If you use this manual as a beginner’s tutorial for working with DNA, however, it would be helpful to follow the steps outlined in sections 4.2 and 4.3 in order to gain a better understanding of the DNA’s potential uses and its functions.

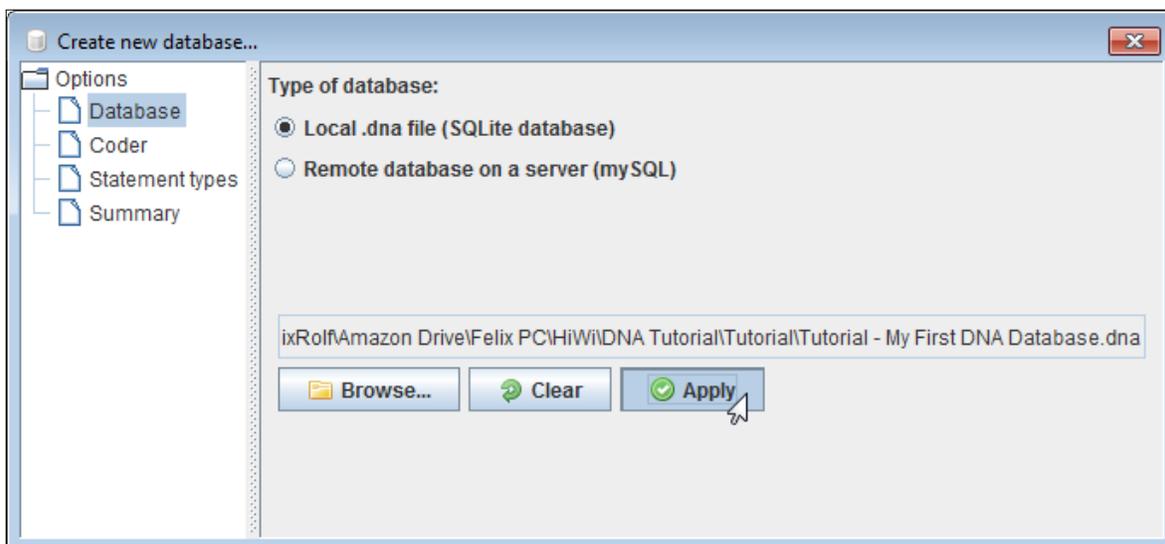


Figure 4.4: Apply database choice

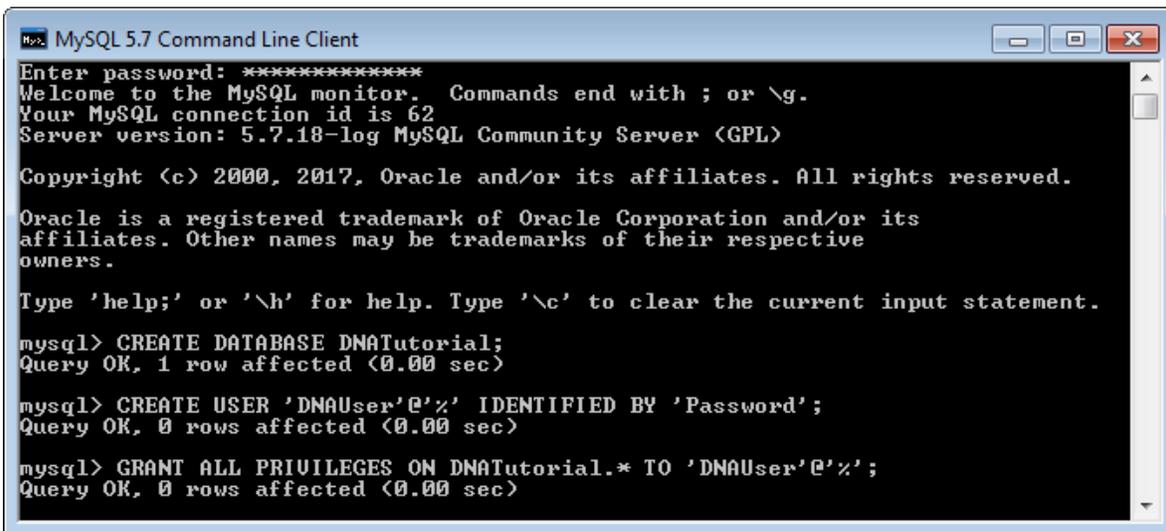
4.1.2 Creating and Using a Remote Database (MySQL)

Before you can configure DNA for working with a remote MySQL database, it is necessary to execute at least three basic operations in MySQL (see Figure 4.5).¹

1. You have to create a database on your MySQL server (*usually by the command* `CREATE DATABASE 'DatabaseName'`).
2. As you probably don't want to allow all coders access to all other databases stored on your MySQL server, you should create distinct user profile(s) for the coding process of your DNA project. Even if DNA itself allows for **managing multiple different coder roles**, we recommend to create separate user profiles for each of the individual coders—especially if they simultaneously edit the content of your database. It is also advisable to create passwords for the access to your database, not only for safety reasons, but also because DNA sometimes has problems with signing in users without a password. Consequently you would use the `CREATE USER 'Username'@'%' IDENTIFIED BY 'Password'` command. Note, that in this step you could also restrict the respective users access to your database to a specific device by replacing '%' through a particular server address if this is necessary.
3. Finally, you have to equip the users with the necessary rights to edit your database. In MySQL simply use `GRANT ALL PRIVILEGES ON Databasename.* TO 'Username'@'%'`, as it makes more sense to specify distinct user roles and rights directly in DNA (see 4.2), where options were tailored to fit discourse network-analytical coding purposes.

Once the MySQL database is set up, you only have to select the option `Remote database on a server` in the first tab of the sidebar menu `Database` in DNA (see **Creating a new DNA Database**) and enter the respective username and password created in the previous step in the respective fields `User` and `Password` as well as to specify the server address of the database, with which you want to connect, in the field `mysql://`. If you want to access the database remotely from another device, you have to indicate

¹For a detailed introduction to database management with MySQL see <https://dev.mysql.com/doc/mysql-getting-started/en/>.

A screenshot of a Windows-style window titled "MySQL 5.7 Command Line Client". The window has a black background with white text. The text shows the MySQL prompt and the execution of three SQL commands: "CREATE DATABASE DNATutorial;", "CREATE USER 'DNAUser'@'%' IDENTIFIED BY 'Password';", and "GRANT ALL PRIVILEGES ON DNATutorial.* TO 'DNAUser'@'%'". Each command is followed by a confirmation message: "Query OK, 1 row affected (0.00 sec)", "Query OK, 0 rows affected (0.00 sec)", and "Query OK, 0 rows affected (0.00 sec)". The window also displays the MySQL welcome message and server version information.

```
MySQL 5.7 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 62
Server version: 5.7.18-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE DNATutorial;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE USER 'DNAUser'@'%' IDENTIFIED BY 'Password';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON DNATutorial.* TO 'DNAUser'@'%' ;
Query OK, 0 rows affected (0.00 sec)
```

Figure 4.5: Create MySQL database

the URL or IP-address of your host server, the port (which is 3306 in default, but can be **configured manually**) and the name of your database in the format `Hostserveraddress:Port/Databasename`. If you use DNA on the device hosting the database you can instead use the configuration shown in Figure 4.6 (`localhost/Databasename`). By clicking the button **Check** you can now check if DNA is able to connect to your database. If this is successful, you will receive the message `Ok`. Tables will be created (see Figure 4.6); if not, DNA will report `Error: Connection could not be established`. In case of the latter, you should check the validity of your server address, username and password—and if necessary—repeat the steps outlined above. It should be noted that—for security reasons—MySQL doesn't allow remote access with the "root" superuser-profile in most cases. Similar to the generation of a local `.dna` file, it is finally important, that you confirm your choices again by pressing the **Apply** button (see Figure 4.6). If you forget to press this button, you cannot create the database in the **final step**, because the program will report "No database selected" (see Figure 4.14).

4.2 User Management: Multiple Coders and Permissions

This second step of preparing your DNA workspace allows you to generate multiple user identities with different sets of rights for different coders. Thus, you can specify for each coder, which parts of the dataset each user can see or edit and thereby pre-structure your coding and research process. In order to do so, click on second tab **Coder** in the sidebar of the **Create new database** menu (see Figure 4.7).

In the main window (see Figure 4.7) you can now see a list with all coders and how many of the 12 possible actions they are permitted to perform. Now you can either add a new user profile by clicking the **Add** button (see Figure 4.7) or select an existing coder and adjust her/his users rights by clicking on the user and then on the **Edit** button (see Figure 4.10). Both options will open the pop-up menu shown in (see Figure 4.8).

This pop-up menu allows you to configure an individual profile for each coder in three simple steps:

1. You can choose the *colour* for the coder (see Figure 4.8, step 1). It is recommended to choose different—if possible—divergent colours for each coder, because this permits you to detect at

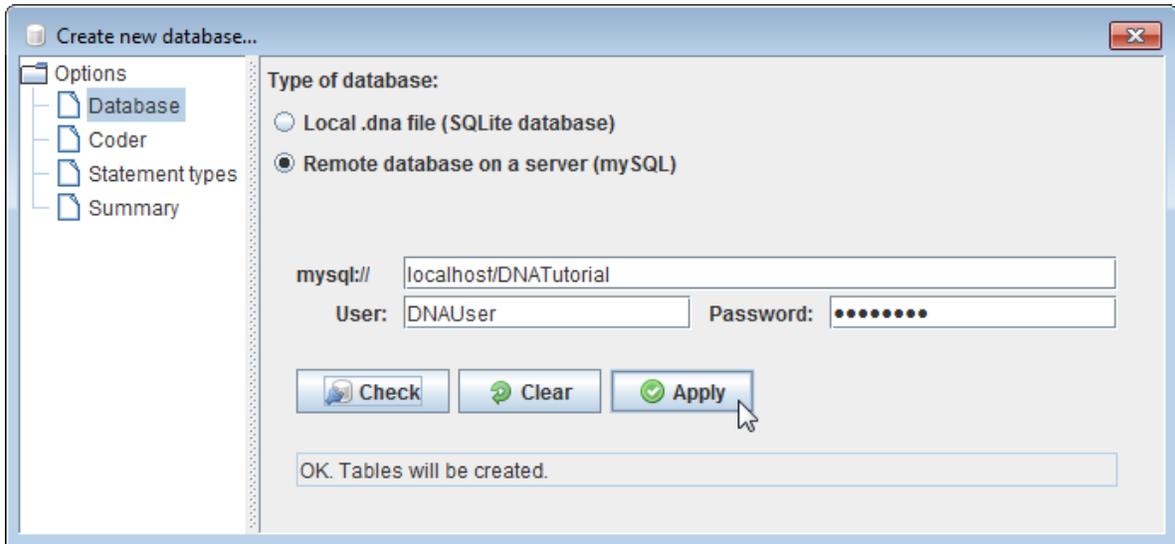


Figure 4.6: Connecting to local MySQL database

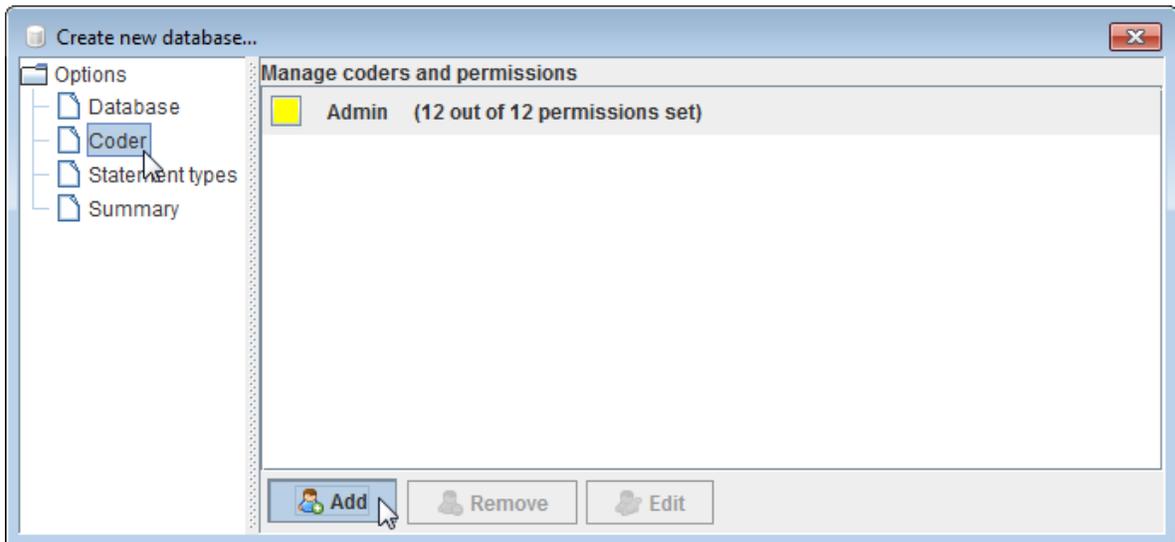


Figure 4.7: Adding a second coder to the database

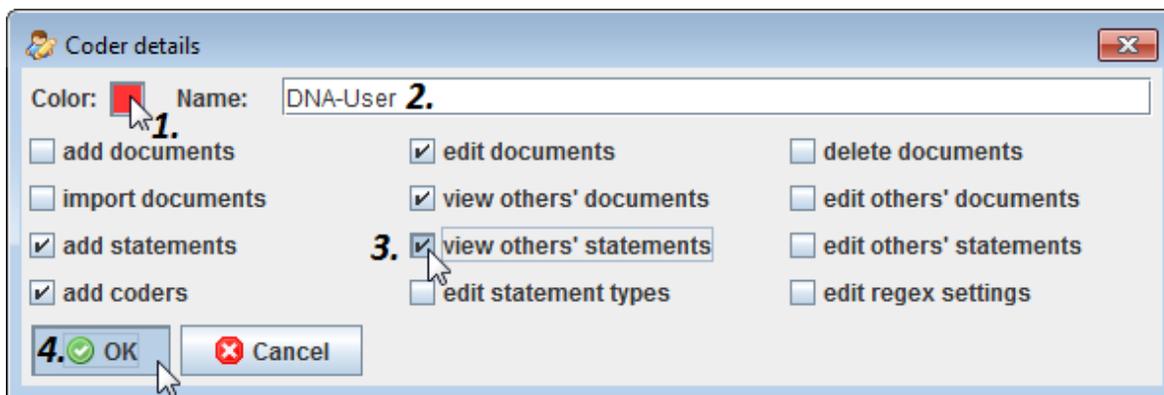


Figure 4.8: Configuring coder permissions

the first glance, which user coded which statement, as every coded statement is marked in the individual colour of its respective coder (see middle column of Figure 4.9).

2. You can enter the preferred name of each coder in the field `Name`. If possible with respect to data protection rules, it is recommended to use the real names of the coders. This makes it easier for them to select their profile (in the upper left of the main program window) the first time they start the program (see Figure 4.9).
3. The final step allows you to configure the *permissions* of each coder individually by (de)selecting the respective rights via a click (see Figure 4.8, step 3). Each new user has all of the 12 configurable permissions in the preset mode. Which parts of the dataset an individual coder should be able to see or edit, should depend on your coding process. For better orientation, a few practical implications of the 12 configurable permissions are listed below:

add documents The user can add new documents (i. e., raw data) manually (via copy and paste or retyping) to the database \Rightarrow user has (also) a research function.

import documents The user can import new documents from other sources like `.txt` or other `.dna` files to the database or recode the metadata of multiple documents \Rightarrow user has (also) a research function.

delete documents The user can delete documents from the database or dataset. This option requires at least the other permission `view others' documents` if the user has an organizing or editing function (structuring database for coding by other users) or the permission `add documents` and `add statements` if the coder determines own codes and organizes her/his own set of data.

edit documents The user can edit her/his own documents (i. e., raw data), but not necessarily the codings in these documents that were made by other users—which would require the permission `edit others' statements`—or the documents uploaded by other users—which requires the permission `edit others' documents`. This option requires at least the other permission `add documents` or `import documents` and should be selected if the user determines own codes and organizes her/his own set of data or acts as a researcher for the other coders.

view others' documents The user can view the documents uploaded by other users. This option is necessary for a collaborative coding process in which only a part of the users selects and uploads the raw data (i. e., documents) for all other users. The option should not be selected if each coder comes up with own codes and organizes her/his own set of data.

edit others' documents The user can edit the documents uploaded by other users. This option requires at least the other permission `view others' documents` and should be selected if a user organizes or edits the raw data provided by other users.

add statements The coder actually codes the data by creating and editing statements. If only a part of the users select and upload the raw data this option requires the additional permission `view others' documents`. If the coder suggests own codes and organizes her/his own set of data this option requires either the additional permission `add documents` or `import documents`.

view others' statements The coder can view the statements coded by other users. For example the Coder “DNA User” would not see the yellow statement of the Coder “Admin” in Figure 4.9 if this option was deselected for her/his user role. This option should be de-selected if you want to establish a blind coding process.

edit others' statements The coder can edit or correct the statements coded by other users. This option requires at least the other permission `view others' statements` and should only be selected for few users with an *organizing*, *controlling* or *editing function*.

add coders The user can add new coders (see Section 4.2). This option should only be selected for few users with an *organizing* function.

edit statement types The user can change or complement the variables of interest (see Section 4.3). This option should only be selected for very few users or the researchers themselves because possible adjustment of these variables is usually only necessary in cases when the research design and/or research questions change fundamentally.

edit regex settings The user can specify keywords which are highlighted in the text, along with a text color (see Section 6.4). For example, in Figure 4.9 the word `colors` is highlighted in the raw data text (middle column), because it was specified as a keyword in the *regex highlighter sidebar* in the bottom left of the DNA window. If a user does not have the right to edit the regex setting, the buttons `Add` and `Remove` in this highlighter would be hidden, but the keyword would nevertheless be visibly highlighted in the text and listed in the regex highlighter sidebar. Thus, if you specify a distinct set of theory based keywords in advance in order to render the coding procedure semi-automatic, you should not enable this option or select it only for *few users*, as the respective coder could change the keywords. However, if you don't have a theoretically relevant set of keywords in advance or just specify them as a assistance for your coders, you can allow them to formulate such keywords by themselves.

Please keep in mind, that every user can see and change to other user identities either accidentally or because of non-compliance, as s/he has to select her/his role the first time s/he starts the program and can change her/his role anytime (see above and Figure 4.9)

Finally you approve your choices by clicking the `OK` button (see Figure 4.8, step 4). It is possible to change the settings either in the “new database” menu by selecting the respective user and clicking the `Edit` button (see Figure 4.9) or changing the coder settings in the main menu. To do so, simply select a coder from the drop-down menu in the window at the top left of the main menu and then push the pencil icon underneath. The same menu as depicted in Figure 4.9 will open up again.

4.3 Statement Types and Variables

Clicking on the third tab in the sidebar of the “Create new database” menu—`Statement Types` (see Figure 4.11)—opens a menu, which allows you to adjust or supplement either the variables or the types of statements, which your coders derive from the raw data.

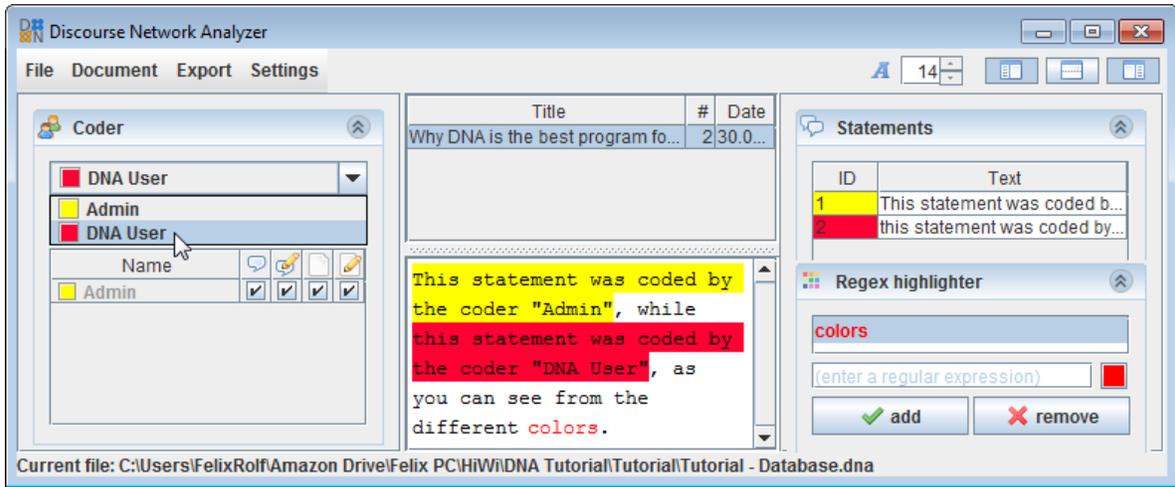


Figure 4.9: Change coder identity

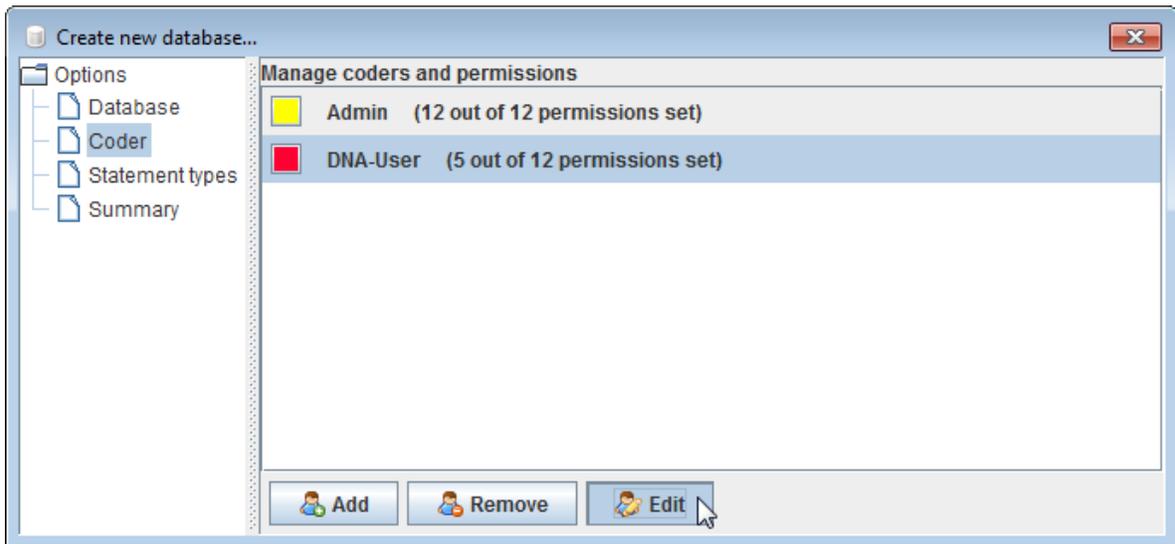


Figure 4.10: Edit coder details

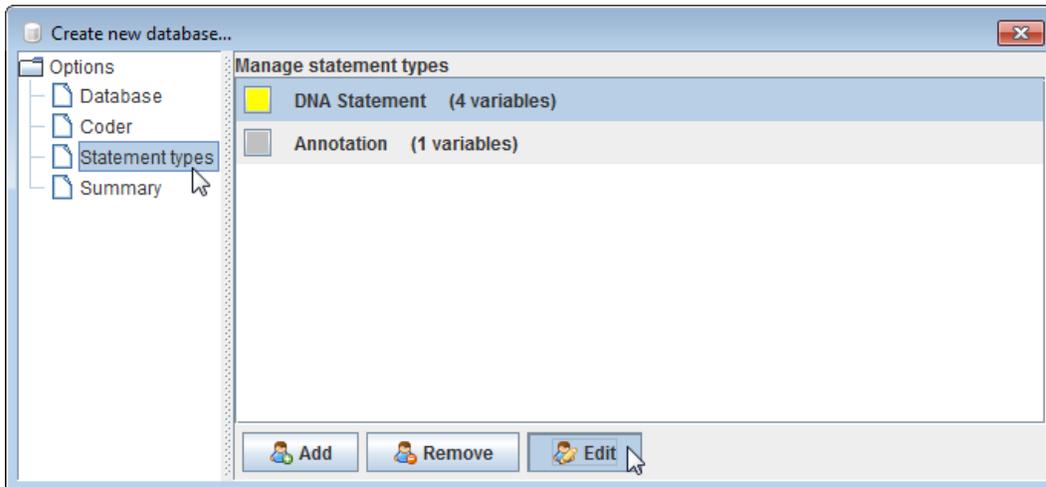


Figure 4.11: Edit Statement Types

4.3.1 Adjusting the Variables of Interest

The statement type DNA Statement represents a text portion of your raw data, where an actor reveals her/his opinion/belief/etc. about an issue. Thus, the main task of your coder(s) is to identify such text portions and gain the relevant data about the actor or his opinion/belief/etc. Your research question or theory should not only dictate what kind of information should be coded as statements, but also which relevant variables of this information should be captured by the coder. As you can see in the “Statement Types” menu, DNAs default configuration allows capturing four variables. Selecting DNA Statement and clicking on the button Edit (see Figure 4.11) opens a pop-up window (see Figure 4.12), which reveals the nature of this four preconfigured variables, along whose lines the coders can collect information:

- the *person* who makes the statement.
- the *organization* the speaker is affiliated with.
- the *concept* (opinion/belief/etc.) which is raised by the actor.
- a dummy variable indicating whether the actor *agrees* with the concept or not.

Furthermore the pop-up window depicted in Figure 4.12 shows, that each variable is assigned to a specific data type: While *person*, *organization* and *concept*—according to their nature as nominal variables—will be coded by a short text, *agreement* as a dichotomous variable will be coded as a **boolean data type**, which accordingly only allows for two forms (either agreement or non-agreement). Neither the data type nor the name of the variables can be changed directly. However by selecting a variable and clicking on the *trash symbol* (on the right side of the Add Variable button, Figure 4.12, step 4) you can delete a variable and subsequently replace it by a new one. Generating a new variable—either to replace one of the preconfigured variables or because you are interested in an additional or a different set of variables—is possible in five simple steps:

1. You have to *select an existing variable* in order to activate the variable menu (see 1, Figure 4.12).
2. Now you can enter the *name* of the new variable in the *text field* at the bottom of the pop-up window (see 2, Figure 4.12). For example, in Figure 4.12 we are interested in collecting the age

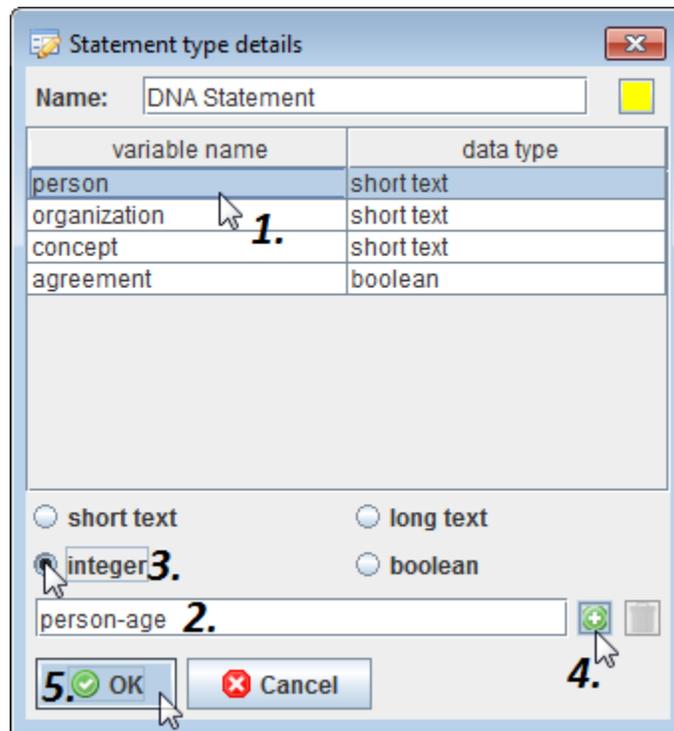


Figure 4.12: Edit Statement Type details

of the person who makes the statement. Please note, that DNA does not allow spaces in variable names. Putting a space in the variable name will disable the `Add Variable` button necessary for step 4.

3. Now you can choose the *data type* of your variable by clicking on one of the four options. In our example, we choose the option *integer*, as the age of a person is neither a nominal nor a dichotomous variable, but an *integer number*) (see Figure 4.12, step 3).
4. You have to click on the `Add-Variable` button, which has the form of a *green plus symbol* (see 4, Figure 4.12). If this button is disabled, you probably did not select a existing variable (step 1) or have a space in your variable name (see step 2).
5. Click the `OK` button to confirm your choices (see Figure 4.12, step 5).

Please note, that—for the statement type “DNA Statement”—you should only specify variables, in which you have an actual research interest in and that accordingly have to be coded for all statements by all coders. If you are interested in additional and optional information about some statements, you can specify them as variables of the other preconfigured statement type—“*Annotation*”.

4.3.2 Adjusting the Statement Types

There are very few research scenarios, in which it is necessary to complement the two existing types of statements with further ones or with an adjustment of type “DNA statement”. One of them would be, if you study two parallel yet different research questions, which employ the same dataset *and* the same coders at the same time. In this case, you could first rename the statement type “DNA Statement”

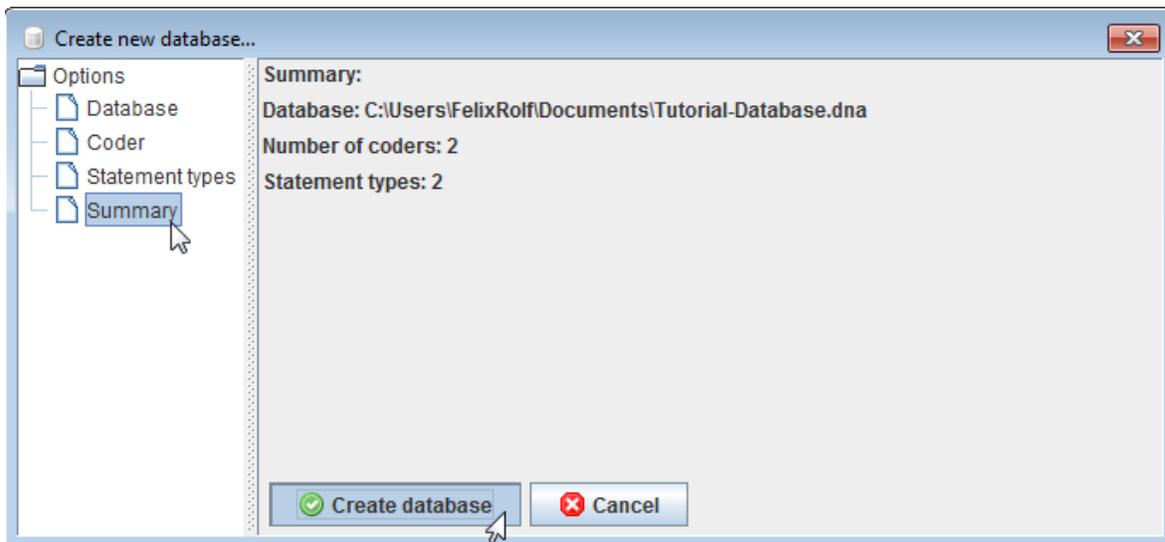


Figure 4.13: Summary of your about to be created DNA database

by selecting it from the statement type menu, clicking the `Edit` button (see Figure 4.11), entering the new name (in this case: “Statement for Research Project 1”) in the text field on top of the pop-up window (see Figure 4.12) and pressing the `OK` button (see 5, Figure 4.12). Subsequently you would open a new pop-up window by clicking on the `Add` button in the statement type menu (left button in Figure 4.11). Then name the new statement type (in this case: “Statement for Research Project 2”) in the text field on top of the pop-up window and choose a color (different from the other type) by clicking on the colored button next to this text field. Then you also need to specify the relevant variables synchronous to the procedure depicted in Section 4.3.1. However, please evaluate carefully, if it is really necessary for your second research interest that you specify a second statement type or if it would be possible to either conceptualize it as a variable of the existing statement type or study it sequentially or with a different set of coders (and therefore in a different DNA dataset). *More than two statement types (besides “Statement” and “Annotation”) can cause a confusion of the coders and therefore compromise the validity of the coding procedure.*

4.4 Final Step: Approving your Workspace and Creating the DNA File

Finally, clicking on the `Summary` tab in the sidebar of the “Create new database” menu provides you with a summary of your choices in respect to the configuration of your coding process (see Figure 4.13). After controlling each of the three information you can now create your database by clicking on the `Create database` button. If this button is disabled and you get the error “No database selected” (see Figure 4.14), you probably forgot to click the `Apply` button after specifying your database (see Section 4.1.1, step 3). After creating the database, the new database will open in the main DNA window (see Figure 4.1) and you can proceed towards loading up and organizing the raw data.

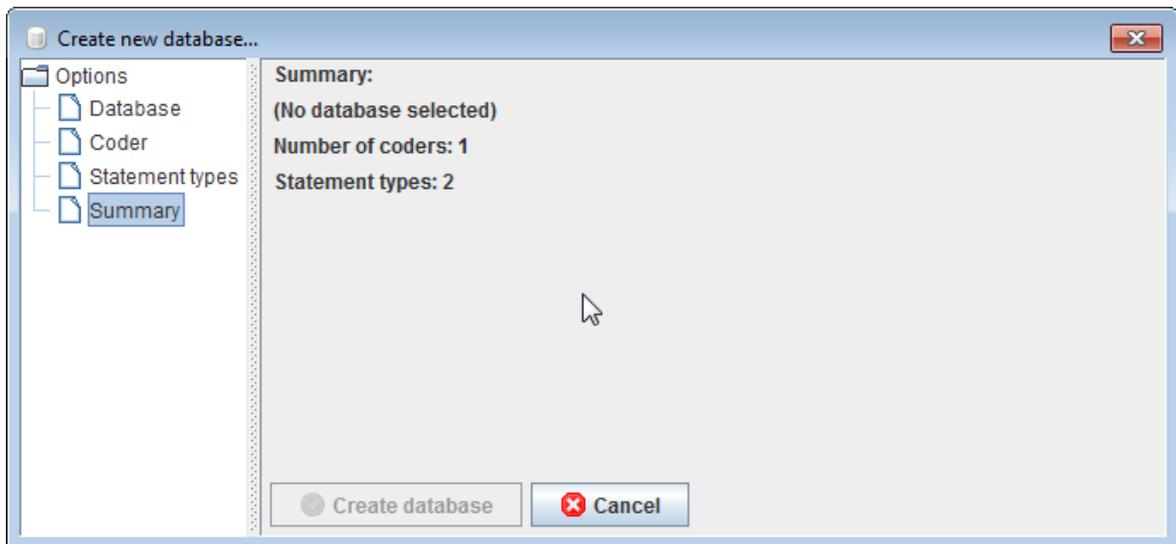


Figure 4.14: No database selected (e. g., if choice was not applied)

Chapter 5

Importing and Organizing your Raw Data

FELIX ROLF BOSSNER AND JOHANNES GRUBER

This section describes how to upload and organize your research project’s raw data—i. e., the text files (newspaper articles, press releases etc.) containing the uncoded statements—in DNA. First it will be layed out how you open an existing database—either locally or from a remote location. Then you will learn how to import new document into DNA—either by importing one document at a time or by selecting mutliple documents for import. Finally, we tell you how you can organise the documents in your database and how you can change your docuemtns’ metadata.

5.1 Opening an Existing DNA Database

First of all, you have to choose, in which DNA Database you want to upload and process your data. To open a DNA database, simply follow the steps depicted in Figure 1: First, click on the index tab **File** and select the option **Open DNA database** (see Figure 5.1, step 1). As a result, a pop-up window will appear, which allows you to choose between opening a **Local .dna file** or a **remote database on a server**. If your database is stored on a remote server, you should choose the second option and repeat the procedure outlined in [Creating and Using a Remote Database \(MySQL\)](#). If your dataset is stored in a folder on your local PC or device, you can proceed with the preset option and click on the button **Browse** (see Figure 5.1, step 2), which will open a further pop-up window, in which you can find your database by choosing its storage location from the **Save in** slide down menu (see step 3), selecting the respective database (see step 4) and clicking on the button **Open** both in the pop-up and the “Open existing database...” window (see steps 5 and 6).

5.2 Importing Documents (Raw Data)

There are three different—partly semi-automatic—ways to upload your raw data and related descriptive information (title, date, author, source, section and type of document) into DNA: Importing single Documents manually via copy and paste, Importing multiple Documents semi-automatically from text files and importing Documents from other DNA databases. . All three will be explained in detail in this section.

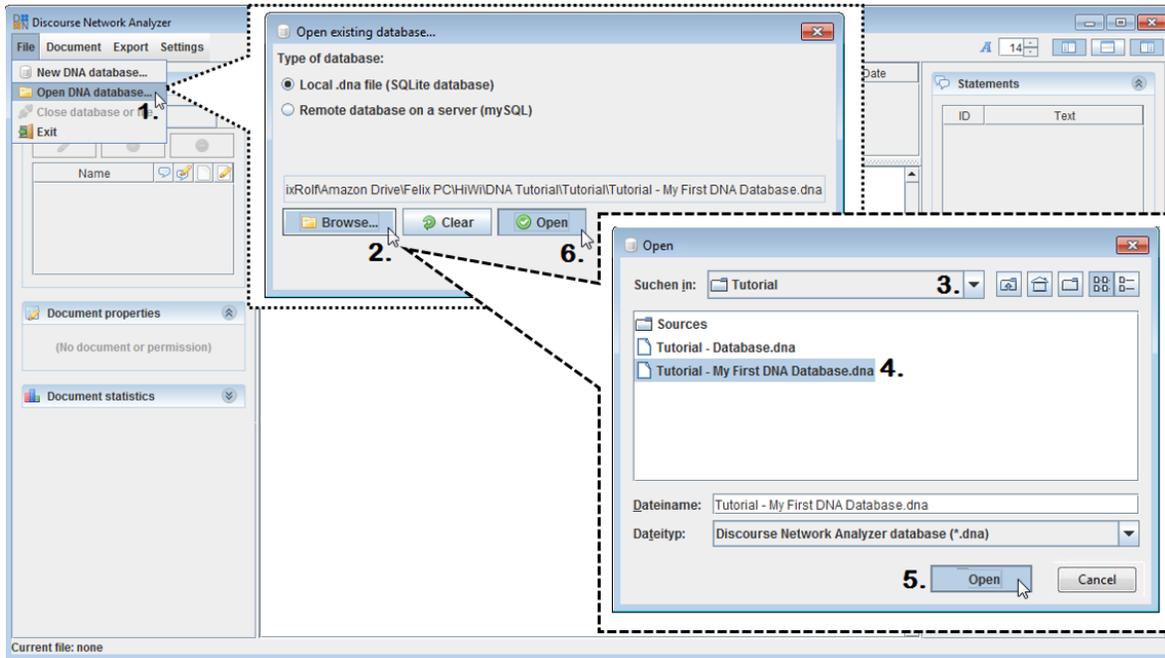


Figure 5.1: Open existing DNAdatabase

5.2.1 Importing Single Documents Manually via copy and paste

The most basic way to import data to DNA requires you to manually copy and paste the content and the descriptive information for each of your documents into the text fields of a pop-up window, which you open by clicking on the index tab `Documents` and selecting the option `Add new document` (see Figure 5.2). This window has eight text boxes, in which you can enter information from and about your source data (see Figure 5.2):

- The field `title` is mandatory and may include any kind of information, for instance a unique ID if you plan to collect additional information about the articles in a separate database. Duplicate article titles are not allowed.
- The field `date` is also mandatory and preset on the current time and day. You can change it by either clicking on the year, month, day or time and adjusting the respective value via the arrows on the right or by manually entering the date in the format `YYYY-MM-DD hh:mm:ss`. Please make sure you enter the date correctly because otherwise the algorithms for longitudinal data (see Section 2.7) will not work properly.
- The fields `author`, `source`, `section` and `type` are optional, but this additional information can help you to efficiently organize your data and ensure the reproducibility, transparency and future usage of your research project. You can enter these information either manually or select an author, source, section or type you specified for a previously added document from the drop-down menu, which appears when you click on the downward arrow button on the left of the respective field.
- To insert the content of your document, copy your article from a website or any other text source and paste it in the `text` field (*largest field at the bottom of the pop-up window*). Single line breaks are automatically removed, while double line breaks (paragraph breaks) are preserved. Some escape sequences and special characters are automatically removed when text is inserted.

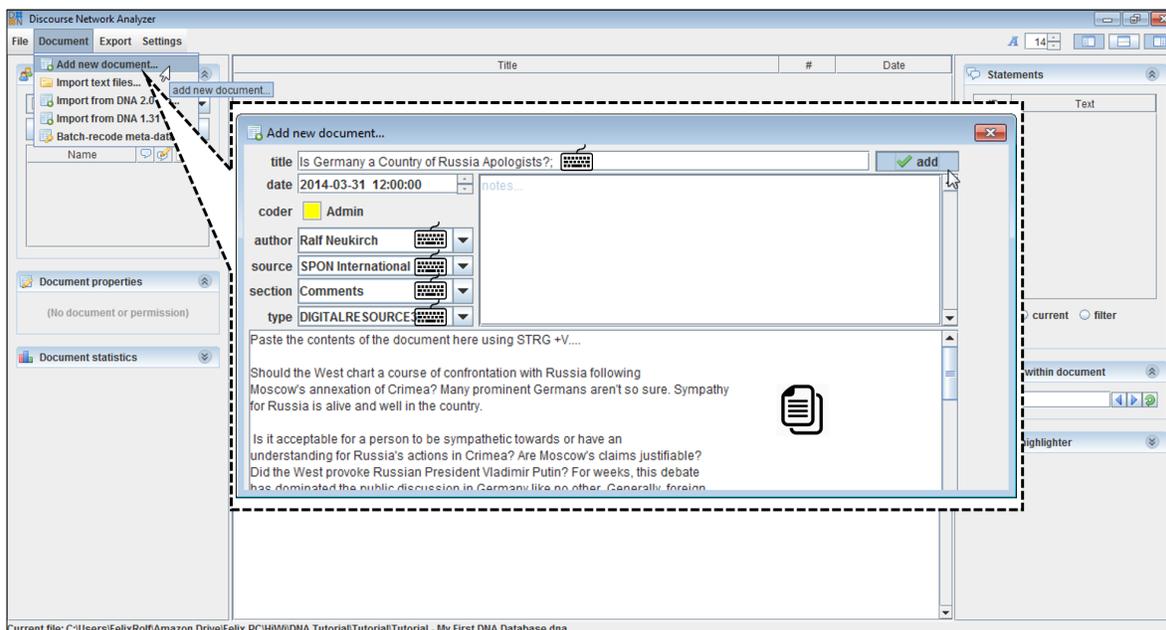


Figure 5.2: Open DNA-database

- If you want to add further meta information to your document, which does not fit the preset categories, you can use the field *notes*.

Finally—after checking your specifications—you can import the document to DNA by clicking the **Add** button.

5.2.2 Importing Multiple Documents Semi-automatically from Text Files

If you want to analyze a greater number of articles, it quickly becomes tedious to manually copy and paste each document and its meta data. This is why DNA also offers a semi-automatic way to upload multiple documents and their relevant meta data (author, date, source, type) at the same time.

Downloading and Preparing your Raw Data

This way of importing raw data to DNA requires that you save all documents as *separate .txt files* (one file for each article) *in a common folder*. Please note, that you have to use the *.txt* format for saving your data, as DNA can not import *.doc* or *.pdf* files.¹ In case you use the newspaper database of LexisNexis—which is available through many university libraries—for finding and retrieving your raw data, please make sure that you download all documents separately (by selecting the individual document before clicking the download button, see Figure 5.3, step 1-2) and choose the document format *Text* (under *Format Options* in the Download pop-up menu, (see Figure 5.3, step 3-4) before downloading the data (see Figure 5.3, step 5).²

¹You can, however, save Word-documents as *.txt* files or use an online converter to transform PDFs into *txt* files. Note, that you need to make sure (both cases) that the *.txt* file is saved with UTF 8 encoding.

²If you use *rDNA* it will soon also be possible to import LexisNexis data into DNA via using *rDNA* and a new *Rpackage* called *LexisNexisTools* (Gruber 2018).

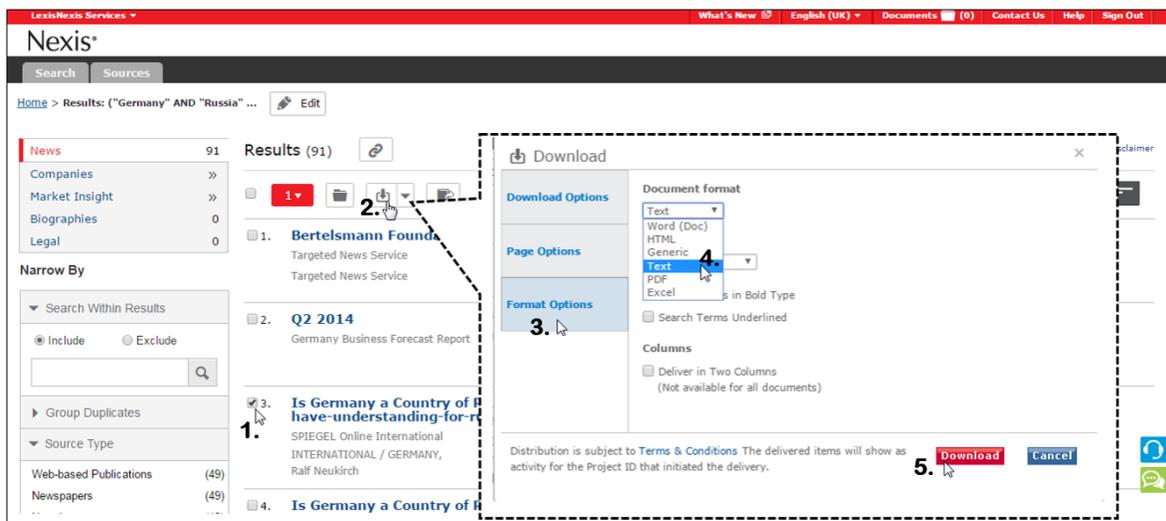


Figure 5.3: Downloading files from the LexisNexis newspaper archive

If you want to use the preset regex configurations (*in contrast to adjusting them*) for automatically detecting and uploading the meta data of your documents, you should use a *file name* in the format `DD.MM.YYYY - Author - Source - TYPE.txt` with blanks before and after the minuses, where `DD.MM.YYYY` is the date, on which the article was published. While `Author` and `Source` do not require a special format or length (e.g., you can use the first and/or last name of the author), the type of the document must always be indicated by capital letters. For example, the file name of the article <http://spon.de/aec1D>, which is used as an example here, would have the format `31.03.2014 - Ralf Neukirch - SPON International - DIGITALRESOURCE.txt`. Please note, that plain text files are sometimes saved as `.TXT` instead of `.txt` files. While this is technically the same, it can cause problems while importing multiple text files. If this is the case, you have to either change the preset Regex configuration or correct the `.txt` suffix manually in the file name(s). Otherwise the automatic detection of your documents' meta data will not work.

Importing your Raw Data into DNA

If you prepared your data adequately, you can retrieve the documents and the relevant additional information in four simple steps (see Figure 5.4):

1. Click on the index tab `Documents` and select the option `Import text files` (see Figure 5.4, step 1). As a result, a new window will open, in which you press the button `Select folder` (see step 2). This will open a further pop-up menu. Here, you have to select the *folder*, in which you saved the text files of your raw data, from the *Look in slide down menu* (see step 3) and click the button `Open` (see step 4).
2. Now all documents, which are stored in the respective folder, should be listed in the main window of the `Import text files...` pop-up (see Figure 5.5). If this isn't the case, please check if your documents are saved in the right file format (`.txt`). In order to check, whether DNA is able to automatically identify your documents' meta data, select one of the documents and click on the `Refresh` button (see Figure 5.5). If you specified the file names correctly, you can now see the respective meta data of the selected document in the fields `Title`, `Author`, `Source`, `Type` and `Date` of the `Preview` Section at the bottom right of the "Import text files" window (see Figure 5.5).

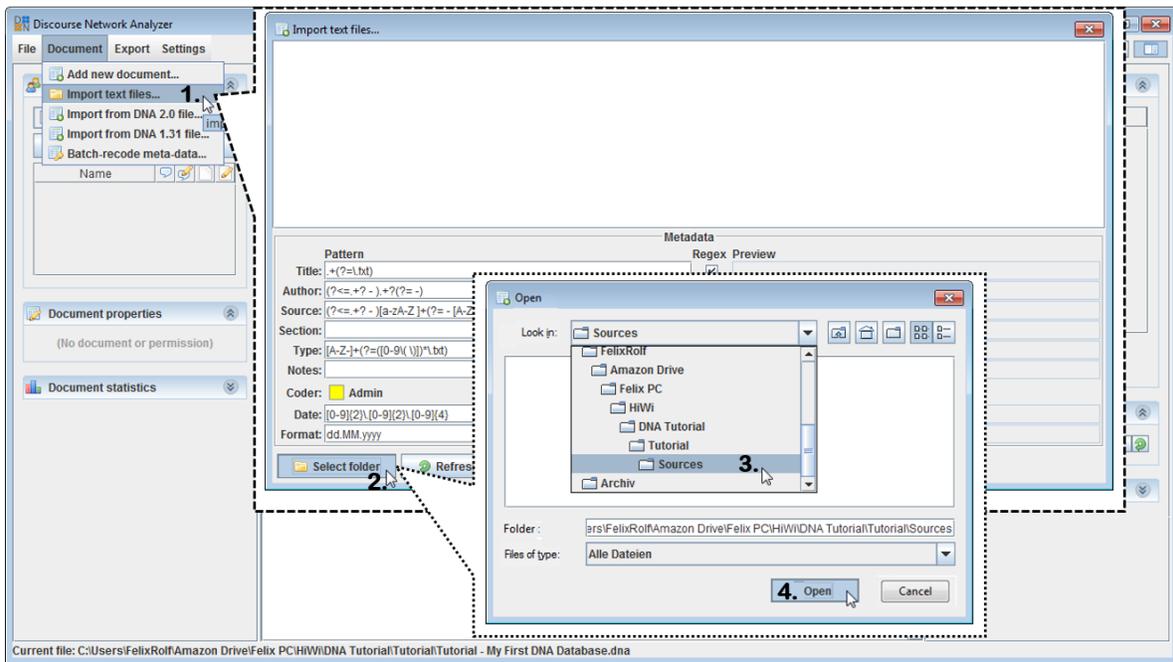


Figure 5.4: Import text files

3. If you want to adjust or amend the meta data manually, just select the document, *uncheck* the box `Regex` of the field you want to edit and enter the new or additional information in the field on the left. Then click again on the `Refresh` button to check, whether your changes were accepted.
4. Finally, click on the button `Import files` to import all documents of the respective folder into your DNA database (you do not need to select each document for import).

Adjusting the Regex Configuration for automatic identification of meta data

The previous steps assumed that you use the preset configuration of DNA to detect and upload the meta data (Title, Author, Source, Type, Date) of your documents automatically into your database. However, if you are interested in automatically importing additional information about your source data (in the fields `Section` or `Notes`) or if your file names depart from the naming system layed out here (but nevertheless contain all relevant information in a systematic order), DNA allows you to change, adjust or amend the pattern, through which the meta data about your documents is derived from the file names. The commands/rules, on which the “translation” of file names into meta data is based, are formulated in the **Regular expressions (in short: `Regex`) syntax** and can be edited for each kind of information (Title, Author, Source, Section, Tyoe, Notes, Date) in the field `Pattern` on the bottom left of the “Import text files...” window (see Figure 5.5). If you want to amend or adjust this settings it is recommended to use a **Regex Cheatsheet** (see e.g., cheatography.com or [this regex “translator”](#)). As further support, Figure 5.6 translates the preset regular expressions of the DNA `Import text files...` option.

5.2.3 Importing Documents from Other DNA Databases

You can also import documents from other DNA databases. This function is particularly relevant in two scenarios: First, if you not only want to use the *raw data*, *but also the coded statements* of an

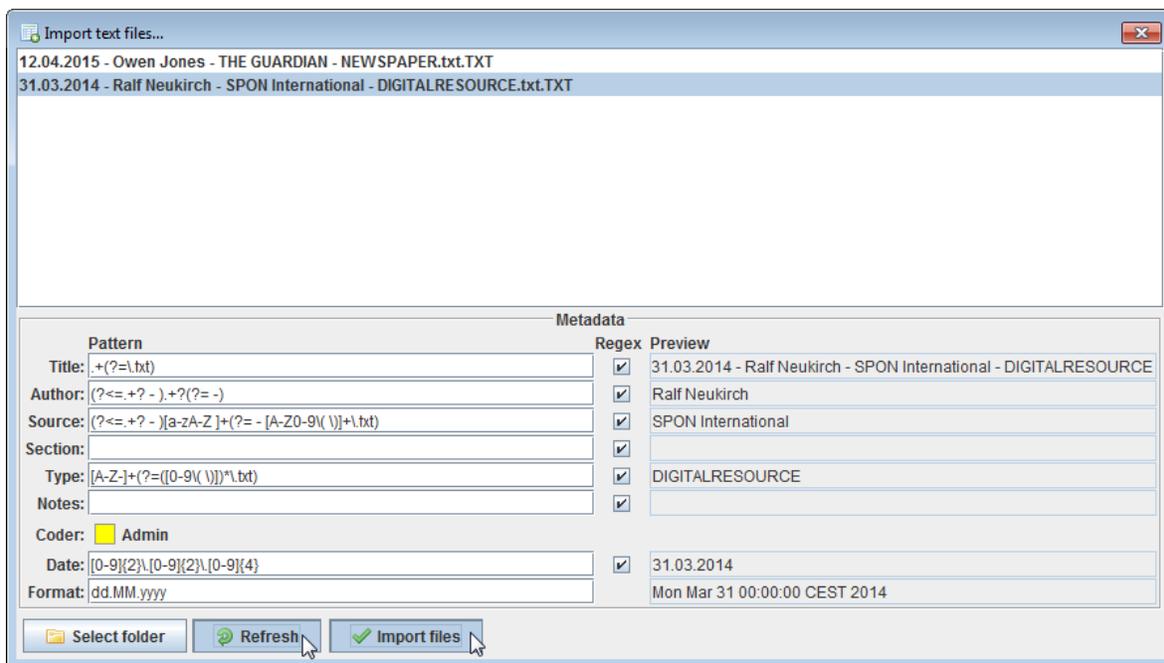


Figure 5.5: Import text files

already finished research project, this function allows you to import both. Secondly, if there is *more than one person working on the same project at the same time* and you did not use multiple user roles (see Section 4.2) to enable your coders to work on the same remote database. In the second scenario, you should use this function to prepare your datasets or merge the codings, as it is usually difficult to merge the files manually later on. In the latter scenario, the function helps you to avoid trouble with *duplicate statement IDs* and article names, as DNA will take care of e. g., duplicates automatically.

Make sure, that you *know which version of DNA* (DNA 2.0 or older) was used to create and edit the database, from which you want to import data, *before* using the “Import from DNA” function. If you use this manual as a beginner’s tutorial for working with DNA please download the file `sample.dna` from the DNA <https://github.com/leifeld/dna/releases>. This file contains a small selection of documents and statements from a larger project about congressional hearings on climate change, employed in the project described in Fisher et al. (2013a,b).

To import documents (and the included code statements), click on the index tab `Documents` and select the option `Import from DNA 2.0 file`, if DNA 2.0 was used to create and edit the database. As the internal structure of `.dna` files has significantly changed since version 1.31, databases created with an older version of DNA need to be imported using the separate method `Import from DNA 1.31 file` (see Figure 5.7, step 1). As a result of either step, a further pop-up menu will open (see Figure 5.8). In this window, you have to select the folder, in which you saved the text files of your raw data, from the `Look in` slide down menu (see step 2) and *select the respective .dna file* (see step 3). Click the button `Open` (see step 4) to then open the menu depicted in Figure 5.8.

In this menu, you can select, which documents (and respective which coded statements) from the original DNA database you want to import in your database by either manually checking or unchecking the boxes on the left of the document title or by using the function “Keyword filter”. This function is particularly helpful if you want to only import few documents with a specific common characteristic (author, topic) from a very large dataset. Clicking on the button `Keyword filter...` (see left button in Figure 5.8) opens a new pop-up window, in which you can enter a specific search term. For example,

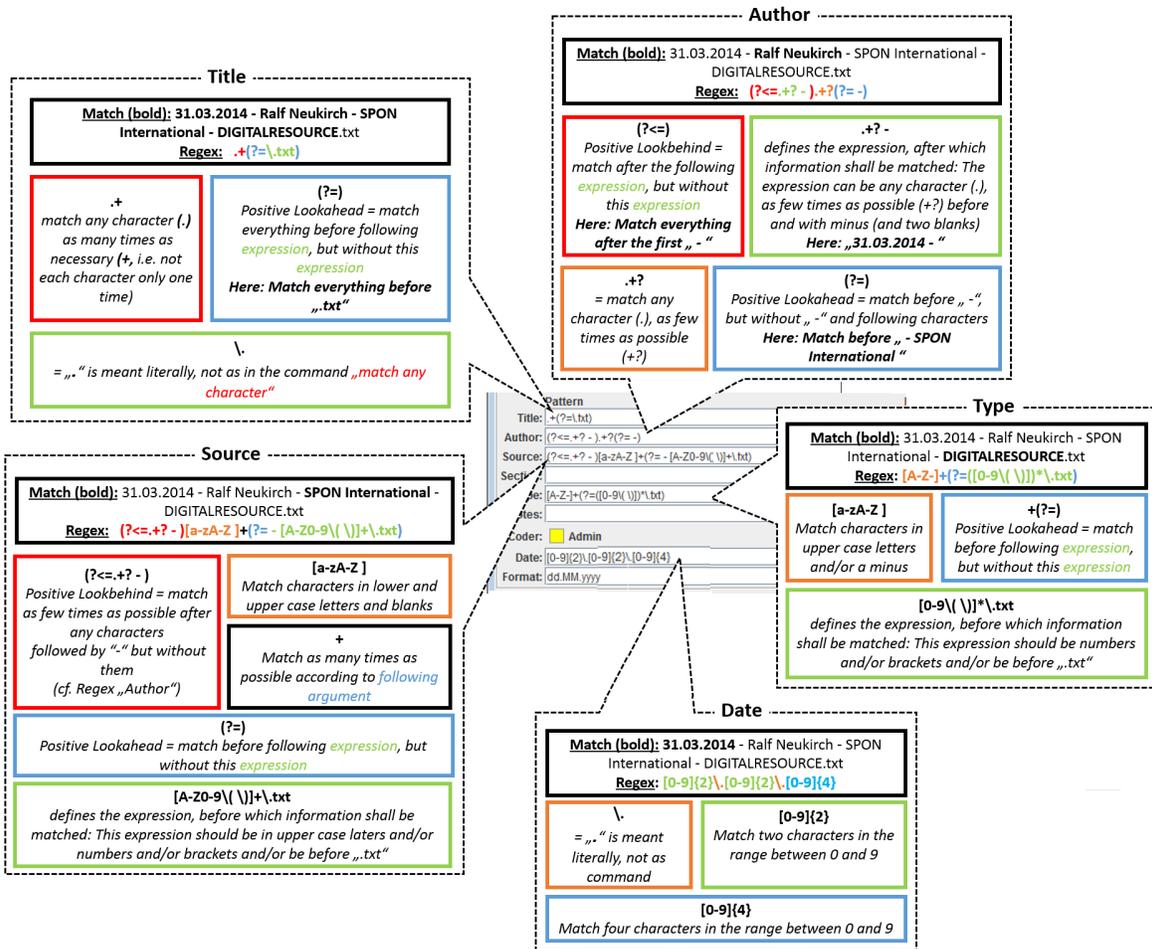


Figure 5.6: Import text files

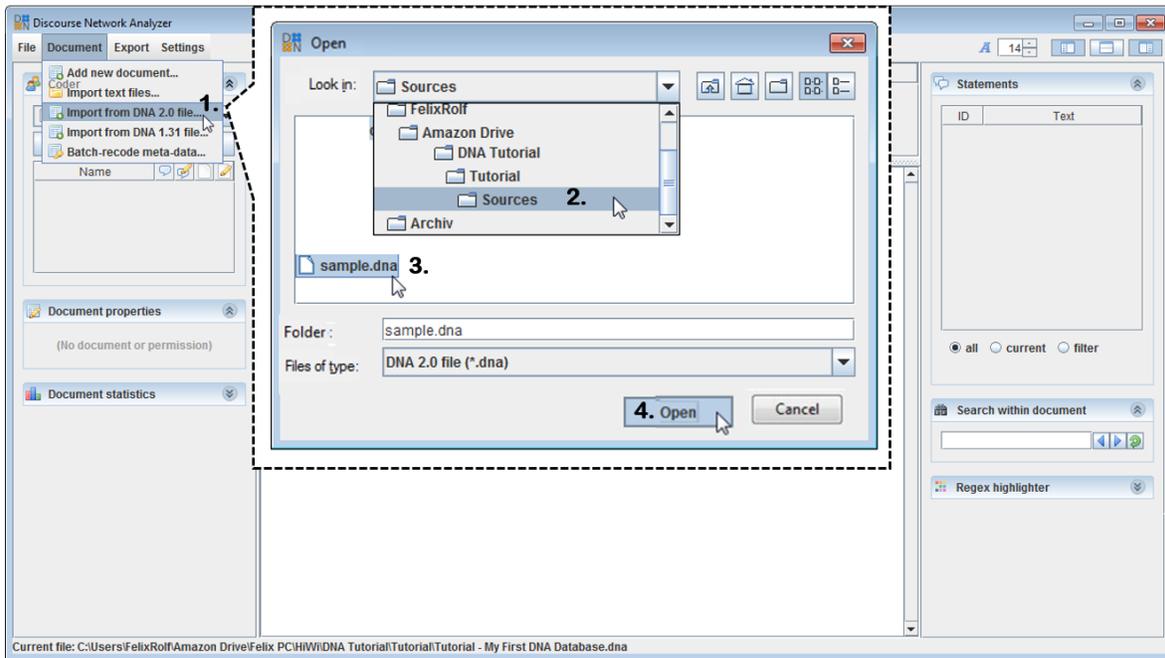


Figure 5.7: Import a DNA 2.0-database

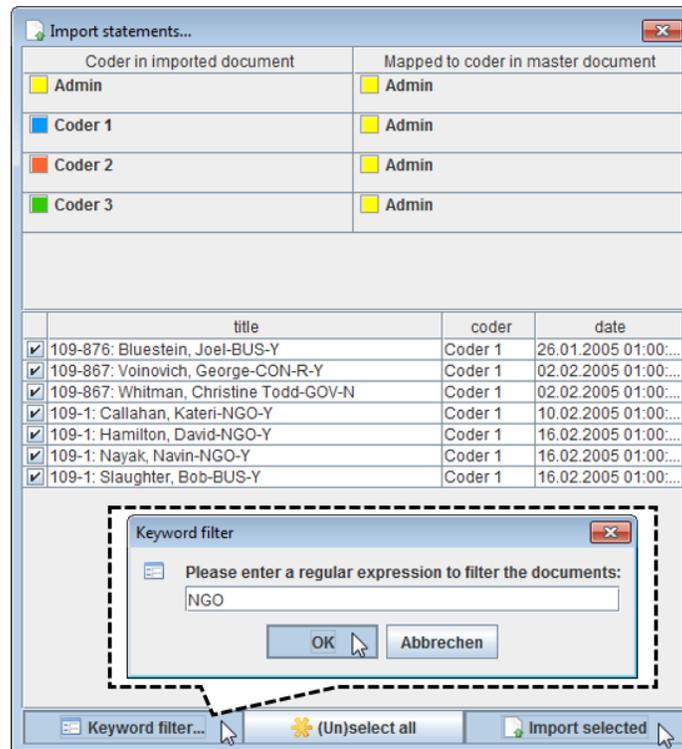


Figure 5.8: Import Statements menu

if you downloaded and opened the `sample.dna` file, you can select all congressional hearings of NGO representatives by *entering the keyword “NGO”* in the text field and pressing the button `OK` in the “Keyword filter” pop-up window (see Figure 5.8). Now only the boxes of the three documents, which contain the hearings of NGO representatives Kateri Callahan, David Hamilton and Nayak Navin, should be checked, while the other boxes are unchecked. The “Keyword filter” function is based on the same regex syntax described in [Adjusting the Regex Configuration for automatic identification of meta data](#). This means, you can also use more specified regular expressions (see Figure 5.6 or [regex cheatsheet](#)) to select certain articles. For example, if you enter a `N` in the “Keyword filter” DNA will select all articles starting with a capital N. If you want to undo your selections, you can also automatically select or unselect all articles by pressing the button `(Un)select all` in the middle of the “Import statements” window (see Figure 5.8). Pressing the right button `Import selected` in the same window imports all documents with a checked box (and the respective coded statements) in your DNA database (see Figure 5.8). If you use this manual as a beginner’s tutorial for working with DNA, you should try importing all documents and the respective statements from the file `sample.dna` into your database.

5.3 Organizing Documents (Raw Data)

5.3.1 Deleting and Navigating Through Documents

All your imported documents are listed in the upper middle table of the DNA main window. If you click on an article, its corresponding text (i. e., the speech) will be displayed in the text area below the document table. By clicking on, for example, the entry `109-1: Callahan, Kateri-NGO-Y` you open the speech of Kateri Callahan, a representative of the Alliance to Save Energy. You can adjust the size of the document table (by clicking on the bar above the text area and moving it vertically with your cursor) or its columns (by clicking on the edge of the column and moving it horizontally with your cursor). You can also customize the meta information, which are displayed in the document table: Just right click on any document and use the appearing context menu to (un-)check the boxes of the information you (don’t) want to be displayed (see Figure 5.9, step 1). A structured (and customised) overview of your raw data is essential for detecting missing information and thus efficiently controlling, organizing and coding your data. For example, if you display the meta information “Type” (by checking the respective box in the context menu), you can see that the type of all documents from the `sample.dna` file is not listed.

The same context menu can be used to delete documents from your database by *selecting the documents* you want to delete (pressing and holding the `Ctrl` key for selecting multiple documents), opening the context menu with a *right click* and choosing the option `Delete selected documents`.

5.3.2 Editing the Documents’ Metadata (Author, Time etc...)

DNA allows you to edit, delete or complement the descriptive information related to your raw data (title, date, author, source, section and type of document). Similiar to the procedures outlined in Section 5.2 there is a manual as well as a semi-automatic way to adjust the meta data of your documents.

Editing the documents’ meta data manually

The most basic way to edit your documents’ metadata is to *select the document*, of which you want to edit the information (by left-clicking on it) and adjusting the values in the `Document properties` submenu on the middle left of the DNA main window (see Figure 5.9, step 2) by either manually typing in the relevant information or by selecting an already specified author, a source, a section or a type from the drop-down menu on the right of the respective meta field. For example, in Figure 5.9 (step 2)

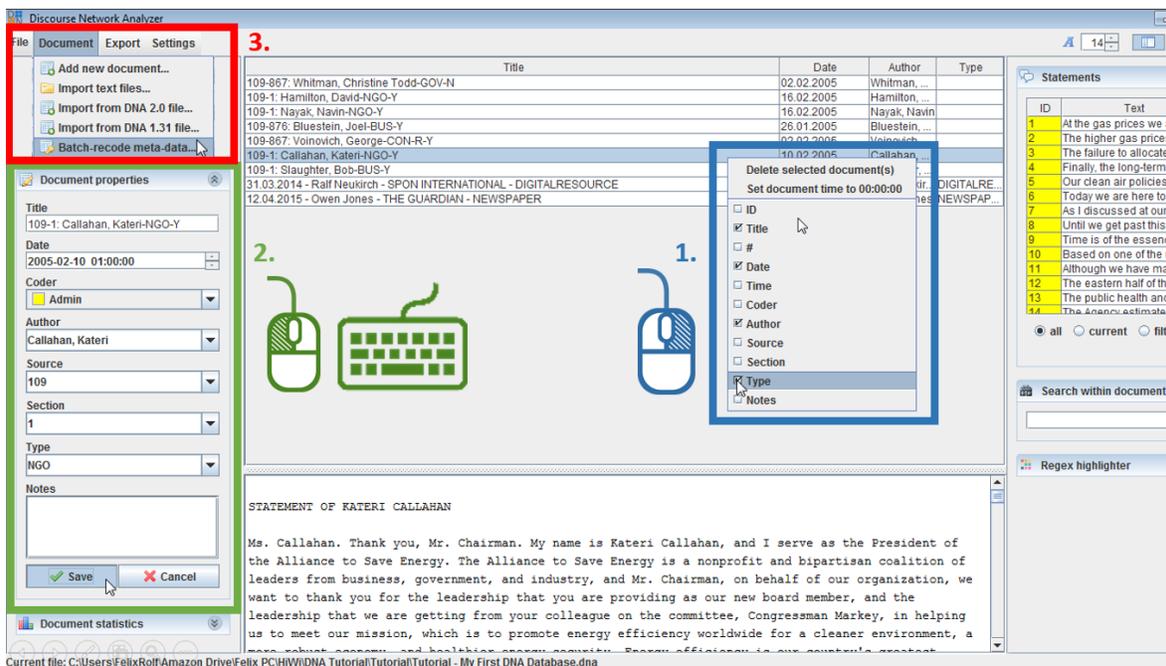


Figure 5.9: Import Statements menu

Kateri Callahans speech was selected, and the value “NGO” (for Non-Governmental Organisation) was manually specified as “Type of document” by entering it in the field Type of the “Document properties” submenu. Do not forget to press the button Save in the submenu (see Figure 5.9, step 2) to confirm your edits.

Please note, that you can manually only edit the meta data of *one document at one time*. If you try to select multiple documents for editing, the “Document properties” submenu will disappear, returning “(No document or permission)”.

Editing the documents’ meta data semi-automatically

However if you want to adjust the meta data of a greater number of articles, it quickly becomes tedious to manually edit information about each document. This is why DNA also offers a semi-automatic way to edit, delete or complement the descriptive information related to your documents. In order to edit your documents’ meta data semi-automatically, click on the index tab Documents and select the option Batch-recode meta-data (see Figure 5.9, step 3). As a result, a pop-up window similiar to Figure 5.10 will open. In the upper half of this pop-up window you find nine fields, which can be configured in order to adjust the meta data for *multiple documents at once*:

- The field Target field: specifies, which kind of meta information (i. e., title, author, source, section, type, notes) should be adjusted by choosing the respective meta data category from the slide-down menu (which you open by clicking the arrow on the right of the target field).
- The field Source field: specifies, where the data you want to use for adjusting the target field is stored. For example, if you simply want to delete or correct (e. g., misspelt) title-, author-, source-, section-, type- or notes-metadata, you usually choose the same field as source field as you have chosen as target field, since you want to adjust the data already stored in this field. However, if you want to add new data to a (maybe empty or incomplete) target field, you have

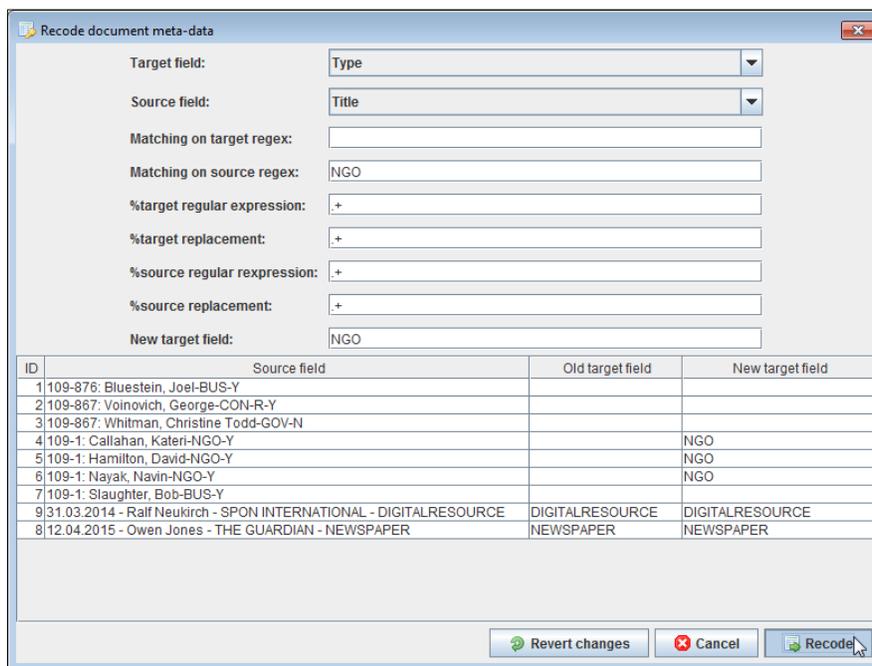


Figure 5.10: Meta information recode window

to choose the part of the meta information as source field, which contains the information, from which you want to derive the new data. As the document title should contain all relevant meta information, `Title` is usually used as source field for the latter case.

- The field `Matching on target regex` allows you to automatically delimit the documents which you want to adjust, based on the information stored in the document's target field. Similar to all regex implementations in DNA you can either use search terms or regular expressions to filter the documents. If you, for instance, misspelt the author "Ralf Neukirch" sometimes as "Ralf Neunkirch", you can correct all your misspellings by simply selecting "Author" as `Target field`, entering "Ralf Neunkirch" in the field `Matching on target regex`: and the correct version ("Ralf Neukirch") in the field `New target field`. As `Matching on target regex` automatically deselects all non-matching cases (here: All documents, who do not have "Ralf Neunkirch" specified as their author), the meta information (here: "Author") remains the same for all other documents.
- The field `Matching on source regex` similarly allows you to automatically filter the documents of which you want to alter the meta data, based on the information stored in the document's source field. For example, if you realise that Ralf Neukirch does not write for "SPON International" (as you erroneously specified), but for "THE GUARDIAN", you can simply correct all your misspecifications by first selecting `Source` as the `Target field` and `Author` as the `Source field`, secondly entering "Ralf Neukirch" in the field `Matching on source regex` and then specifying "THE GUARDIAN" as `New target field`.
- The field `%target regular expression` allows you to specify/match a part of the target field, which you want to use as new information in the same field. For example, if the field `Author` somehow contains the full document titles you can reduce the information in the field `Author` to just the name of the respective author by entering the regular expression `(?<=.+?--).+?(?=-)` (see Figure 5.6 or [regex cheatsheet](#)) in the field `%target regular expression` and entering `%target` in the field `New target field`. *Please note, that if you do not use this function, you*

should not change the preset value `.+` in this field—because if you do, your recoding might not obtain the expected results.

- The field `%target replacement` defines a new value for the information in the target field—similarly to the fields `New target field` and `%source replacement`. If you use `%target` as `New target field`, you have to specify the new, additional, corrected or reduced information in this field.
- The field `%source regular expression` allows you to specify/match a part of the source field, which you want to use as new information in the target field. For example, if your source field is `Title` and the titles of your documents have the recommended format (i.e., `DD.MM.YYYY - Author - Source - TYPE.txt` with blanks before and after the minuses; see Section 5.2.2) you can automatically specify the meta information for the field `Author` by (1.) choosing `Author` as the `Target field` and `Title` as the `Source field`, (2.) entering the regular expression `(?<=.+?--).+?(?= -)` (see Figure 5.6 or [regex cheatsheet](#)) in the field `%source regular expression` and (3.) entering `%source` in the field `New target field`. *Please note, that if you do not use this function, you should not change the preset value `.+` in this field—because if you do, your recoding might not obtain the expected results.*
- The field `%source replacement`—similarly to the fields `New target field` and `%target replacement`—defines a new value for the information in the target field. If you use `%source` as `New target field`, you have to specify the new, additional, corrected or reduced information in this field.
- The field `New target field` defines the new, corrected, reduced or additional data, which is entered in your target field (see examples above). Please note, that this field has to be set on `%source` (preset value) if you use the functions `%source regular expression` or `%source replacement` and has to be set on `%target` if you use the functions `%target regular expression` or `%target replacement`. Otherwise, the respective functions will not work.

The lower half of the “Recode document meta-data” pop-up window (see Figure 5.10) displays a table with four columns and a row for each of your documents, which help you to preview, control and trace back your changes to the meta data:

- The column `ID` contains the individual ID of each of your documents. This column can be particularly helpful if you specify a recoding procedures for a certain set of documents. If you know the ID of a few exemplary documents from this set, you can quickly trace back and understand the consequences of your recoding specifications by scrolling down to the respective IDs and taking a look at the other columns of these documents.
- The column `Source field` displays the field, from which you get the meta data for recoding the target field. It is particularly helpful to understand the sequence of information in the source field, if you want to specify a `%source regular expression` or use `Matching on source regex` (for example, if only some source fields contain the relevant information).
- The column `Old target field` shows the meta data in the target fields prior to your adjustments. It is particularly helpful if you want to use `%target regular expression` or use `Matching on target regex` (for example, if you only want to change the value of a certain set of target fields).
- The column `New target field` displays the consequences of your adjustment. It is particular helpful to check if your recoding will be successful or if some recoding outcomes are actually undesired (for example, if the target field already contained the relevant information, but is recoded nevertheless).

Your recodings are only applied, if you press the button `Recode` (on the lower right of the `Recode document meta data` window, see Figure 5.10). *Once this is applied, it cannot be undone!* So please control the consequences of your recodings by using the table at the lower half of the window. However, before pressing the `Recode` button, you *can* revert all adjustments by pressing the button `Revert changes` and therefore are able to experiment with the meta data (regex) specifications.

As noted previously, all documents from the file `sample.dna` do not specify any meta data concerning the type of the respective document. Both Figure 5.10 and Figure 5.11 illustrate an exemplary semi-automatic procedure for complementing this information based on the information stored in the document title (here: The organisation, to which the respective speaker belongs to). Thus in both examples, `Type` is selected as `Target field`, while `Title` is selected as `Source field`.

The example in Figure 5.10 uses manual search terms to specify the meta information for the document type. By entering “NGO” in the field `Matching on source regex` the adjustments are limited to the documents, which contain “NGO” in the document title. By entering “NGO” in the field `New target field`, the new value for `Type` is specified for the selected documents. As you can see in the table on the lower half of the `Recode meta-data` window, this very simple procedure is insofar successful, as only the target fields of documents containing hearings of NGO-representatives are changed and the target fields of all other documents (including those with already correct `Type` information) remain unchanged. However, this procedure would have to be repeated for each kind of organisation from the sample (NGO, GOV, BUS).

The more elegant way of semi-automatically specifying meta information is depicted in Figure 5.11, which uses the *Regex-syntax*. Here, by entering `?` in the field `Matching on target regex`, only those documents are selected for amendment, which do not already contain any information about the document type (therefore excluding those documents with already correct `Type` information). By specifying `(?<=.\+?-)[A-Z]+` as `%source regular expression` (and accordingly `%source` as `New target field`), DNA is instructed to filter any string of upper-case characters before a minus in the document title and set it as a new value for `Type`. Thus you can recode the document type for all documents at once, ensuring that already specified values are not overwritten—as evident from the table in the lower half of the window.

Recode document meta-data

Target field: Type

Source field: Title

Matching on target regex: ^\$

Matching on source regex:

%target regular expression: .+

%target replacement: .+

%source regular expression: (?<=,+\?)[A-Z]+

%source replacement: .+

New target field: %source

ID	Source field	Old target field	New target field
1	109-876: Bluestein, Joel-BUS-Y		BUS
2	109-867: Voinovich, George-CON-R-Y		CON
3	109-867: Whitman, Christine Todd-GOV-N		GOV
4	109-1: Callahan, Kateri-NGO-Y		NGO
5	109-1: Hamilton, David-NGO-Y		NGO
6	109-1: Nayak, Navin-NGO-Y		NGO
7	109-1: Slaughter, Bob-BUS-Y		BUS
9	31.03.2014 - Ralf Neukirch - SPON INTERNATIONAL - DIGITALRESOURCE	DIGITALRESOURCE	DIGITALRESOURCE
8	12.04.2015 - Owen Jones - THE GUARDIAN - NEWSPAPER	NEWSPAPER	NEWSPAPER

Revert changes Cancel Recode

Regex: ^\$
Match only empty fields (here: fields, which do not have any value for „Type“)

Regex: (?<=,+\?)[A-Z]+

(?<=)
Positive Lookbehind = match after the following expression, but without this expression

.+?-
defines the expression, after which information shall be matched: The expression can be any character (.), as few times as possible (+?) before and with minus

[A-Z]
Match only characters in upper case letters

+
match the characters as many times as necessary (+, i.e. not only one character)

Figure 5.11: Meta information recode window (regex explained)

Chapter 6

Coding the Data

JOHANNES GRUBER

Now that you know how to create a database and organise the documents in it, it's time to start with the actual coding. This section describes how to create, edit and navigate through statements as well as how to employ the regex highlighter and search function to make coding of statements easier and faster. If you want to recreate the steps outlined in this section for practice, you should download the file `sample.dna` from the DNA <https://github.com/leifeld/dna/releases> and open it with the newest version of DNA from the same page—if you haven't already done that. This sample is a small excerpt from a larger empirical research project that tries to map the ideological debates around American climate politics in the U.S. Congress over time. Details about the dataset from which this excerpt is taken are provided by Fisher et al. (2013a,b). Here, it suffices to say that the `sample.dna` file contains speeches from hearings in the U.S. Congress in which interest groups and legislators make statements about their views on climate politics.

6.1 Creating a DNA Statement

For the sake of this tutorial, we can create a new coder. You could also select one of the existing coders from the drop-down menu to create statements but it's just nicer if our new statement is associated with our names. Simply click on the plus sign in the coder menu in the upper left corner of the main window (see Figure 6.1, step 1). In the new menu that opens, enter your name and just leave all permissions selected for now (see Figure 6.1, step 2). Then you should choose a personal colour. Either you select a predefined one from the swatches tab or you define your own colour using one of the menus in the other tabs (see Figure 6.1, step 3). Later on, the statements you have created will be highlighted in this colour. After accepting the edit, you can select yourself from the drop-down coder menu. This should always be the first step before you or one of your coders start to work on the database.

To code a new DNA Statement, simply select a chunk of text by pressing and holding your left mouse button while sliding over it. When all the text you want to include in the statement is selected, push the right mouse button and select `Format as DNA Statement` from the appearing drop-down menu (see Figure 6.1). In the new menu that opens, you need to provide the details for the statement. Every DNA Statement consists of four pieces of information which you should ideally all provide (Figure 6.3):

Person The person or actor who speaks or makes the statement. In discourse network analysis it is most often assumed that the organisation, not the person, are the important actors in a policy

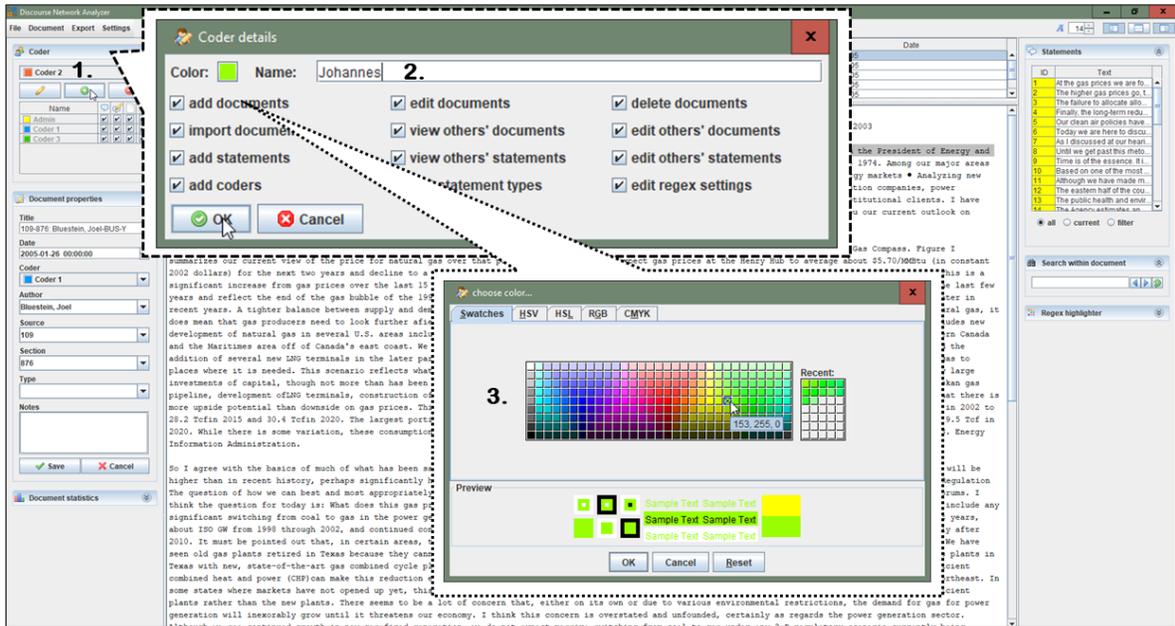


Figure 6.1: Create your own personal user for this exercise

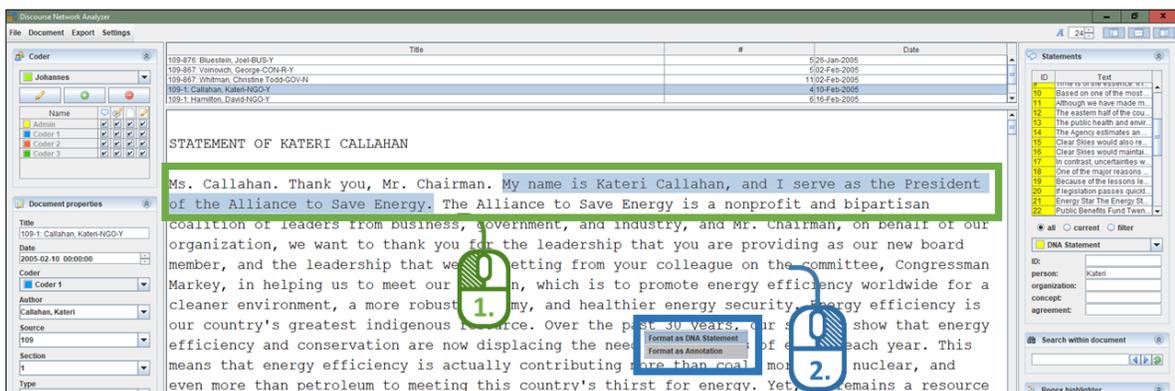


Figure 6.2: Create new DNA Statement

process. So if you have decided from the start of a project that this is the case, you could leave the person field empty. However, it might nevertheless be interesting in a later step if there are differences between persons from the same organisation, so it is advised that you complete all fields.

Organization The organisation the person who makes a statement is affiliated with.

Concept The concept to which the statement refers. Concepts are usually abstract representations of the topics which are discussed. In advocacy coalition research, for example, concepts are claims for policy instruments such as “CO₂ legislation will not hurt the economy”.

Agreement A dummy variable (i. e., a variable with only two possible outcomes) indicating whether the actor agrees with the category or not. Often this is a question of sentiment: if the speaker talks about the concept in a positive way we assume s/he agrees with it. And likewise that s/he disagrees when making a statement in a negative tone. As opposed to the other three fields, it is not possible to *not* provide this information. If you do not tick the box `agreement` you indicate disagreement with the selected concept.

There are two ways to provide the information: you can either click inside one of the boxes to write in a new category or you can choose a category from the drop-down menu (see Figure 6.3). In the first case, DNA will try to auto-complete your code by using all existing categories, so you can leave a field incomplete and thereby choose an existing category. In the latter case, the drop-down menu shows all previously coded information, so that it is not necessary to reenter previously coded categories. This is not only convenient but also serves a reliability purpose: multiple similar but not identical categories, created by misspelling or incorrect abbreviation, would lead to spurious results which could jeopardize analysis later on.

As you can see in Figure 6.3, each statement also has a unique identification number. The ID can't be changed by the user, but it can be used as a primary key if you want to record additional information about your statements in a separate database—e. g., to automatically merge the information again once you move to analysis in R or other programs.

There are also two more symbols in the upper right corner of the Statement window: a plus sign, which creates a copy of the current statement. This can be helpful if the same text passage can be coded as multiple statements, for example, when it mentions multiple persons or organisations which refer to the same concept. And a trash bin symbol, which completely removes your DNA Statement, but leaves the document intact (see Figure 6.3).

After you are done providing the detail information, you can simply click anywhere outside the menu to return to the main window. The moment you leave the DNA Statement window, all your edits are saved in your database—so there is no need to apply the changes or save your progress periodically (see Figure 6.3). If you click outside the window by accident, you can return to edit the statement by simply clicking on it again (more in the next section). After you return to the main page, the statement is now highlighted either in yellow or in the personal colour of the coder you selected (see Figure 6.4). All DNA Statements have the colour which was selected when the DNA database was created. The only way to change the highlight colour would be to create a new database. However, by opening the Settings and selecting `Color statements by coder` the highlight colour changes to the colour of the coder who created each statement.

Besides DNA Statements, you can also create annotations. In the sample database, annotations are highlighted in grey. You can use this feature, for example, if you are not quite sure if a text passage is a statement. Then you or another coder could review all annotations later on and decide if it should be coded as a statement or not. In projects with multiple coders who work on a remote database, the feature can also serve as a mean of communication between coders, for example, to provide instructions for a specific document or to make each other aware of a certain sentence or paragraph which might

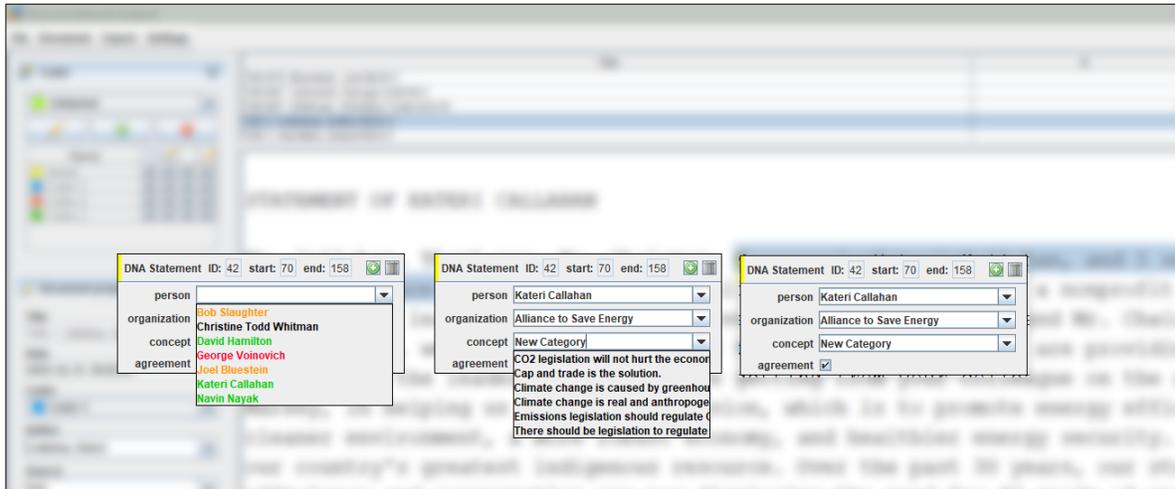


Figure 6.3: Create new DNA Statement

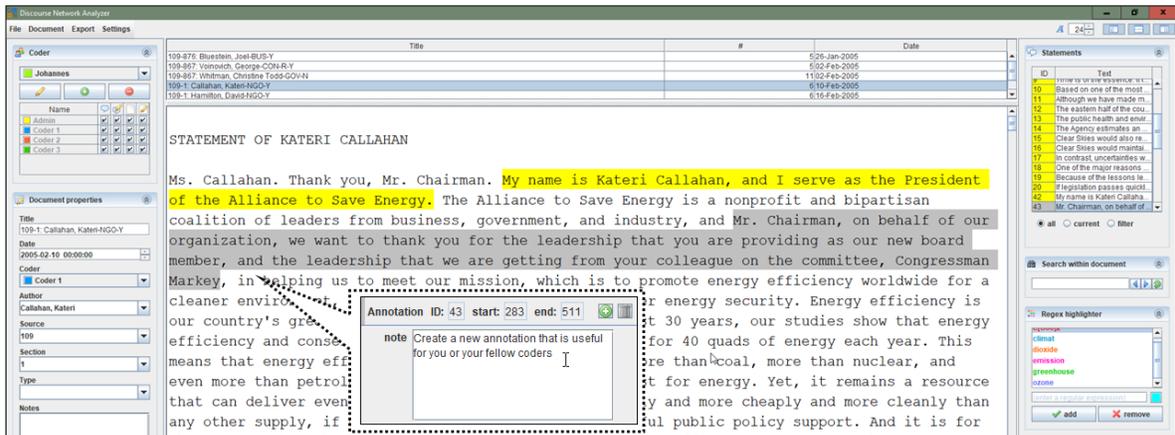


Figure 6.4: DNA Statements are highlighted in yellow, annotations in grey by default

not be a statement but nevertheless important in other ways. Basically, annotations in DNA work in the same way as in MS Word, Google Docs and other applications.

6.2 Navigating Through Statements

For this subsection, we focus our attention on the `Statements` window on the upper right corner of the main menu. This window is presented as a table with an “ID” column and a “Text” column in which the text underlying a DNA Statement is displayed (see Figure 6.5).

There are three ways to navigate through statements. The first is displayed if you select the option `all` below the list of statements. As the name suggests, the window shows all statements, ordered by appearance in your database. The statements in the first document are on top, the statements in the last document you added are at the bottom of the table. By clicking on a statement you jump to the document and position it appears.

The second option is called `current`. It provides a filter, so only the statements in the currently selected document are displayed in the `Statements` table. If you navigate to a different document, this selection will change. On top of the table, you should see the new statement you just created. If you haven’t added or removed a statement before, the ID of the statement should be “42”.

The third option is called `filter`. When you choose this option, the “Statements” box increases in size and shows five new fields in which you can provide keywords to filter by for different fields in the DNA Statement: ID, person, organization, concept, agreement and a drop-down menu where you can switch between statement types. By entering, for instance, “There should be legislation to regulate emissions.” in the concept field, only statements regarding that concept will be displayed. Again, it is not necessary to provide the full text but it is sufficient to write in enough characters to differentiate the code clearly from other ones in the database. For example, if there would be a different concept called “There should be legislation for cap and trade.”, then you would have to enter at least “There should be legislation *t*” before DNA can differentiate the two concepts. Or even easier, you could simply enter “emissions” to display only statements for which the concept contains the word “emissions”. Additionally, you can again use regular expressions in the filter fields (see Section 5.2.2). For instance, if you type `1[2-5]` into the ID field, DNA will display all statements between 12 and 15.

Instead of searching statements by just one variable, you can also combine filters. By entering “There should be legislation to regulate emissions.” in the concept field and “Bob” in the person field, only the DNA Statements regarding this specific concept and were made by Bob Slaughter—who is the only Bob in the database—are displayed.

The filter fields all work in this way except `agreement` which only takes the values 0 for disagreement and 1 for agreement.

6.3 Editing a Statements

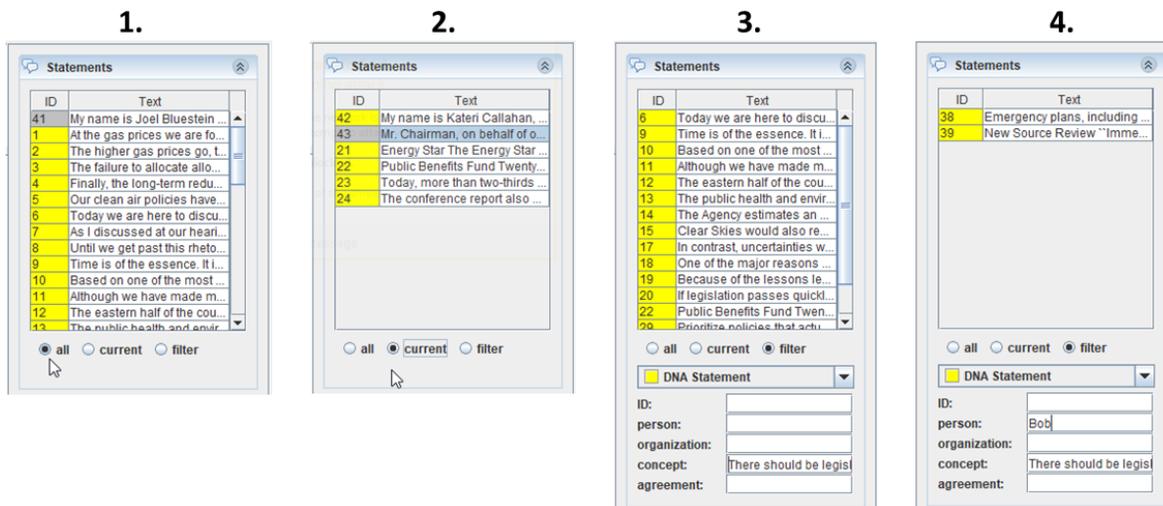


Figure 6.5: Statements window in detail

To edit a statement’s details, you first need to select it, either by choosing it from the Statements table (see Figure 6.5) or directly in the text. This will open the same window as in Figure 6.3 which we discussed in detail in Section 6.1.

Besides that, you can also provide additional information for the variables in the DNA Statements. To find this menu you first have to toggle the option to display a new window beneath the text window. To do that, find the view options in the top right corner of the main window (see Figure 6.6, step 1). The three options you see here control how DNA looks or, more precisely, which menus are displayed. The left option displays/hides the “Coder” and “Document properties” menus, while the right one displays/hides the “Statements”, “Search within document” and “Regex highlighter” menus. However, now we need the middle option which displays the mentioned field beneath the document text.

In this window, you can make edits to persons, organizations or concepts in all DNA Statements at once, rather than to single statements. The advantage is that you do not have to edit each individual statement if you notice that you misspelt a name, that two concepts which you have coded actually represent the same idea or if you simply find a better concept name which adds some clarity. When you edit a concept (or person or organisation) in this menu, it is changed in all DNA Statements at once. When this menu is first opened, it contains no information before you select either DNA Statement or Annotation from the left drop-down menu (see Figure 6.7).

To provide additional information for persons, organizations or concepts you have to click on the label symbol in the top right corner of the same window (see Figure 6.6, step 2). As in the other menu, the window contains no entries before you select if you want to edit statements or annotations from the left drop-down list (see Figure 6.6, step 3). In the table presented now, you can add colour, type, alias and notes for each person, organization or concept. This information is not directly used in DNA but, as we will see in Chapter 8, can be used during analysis with rDNA.

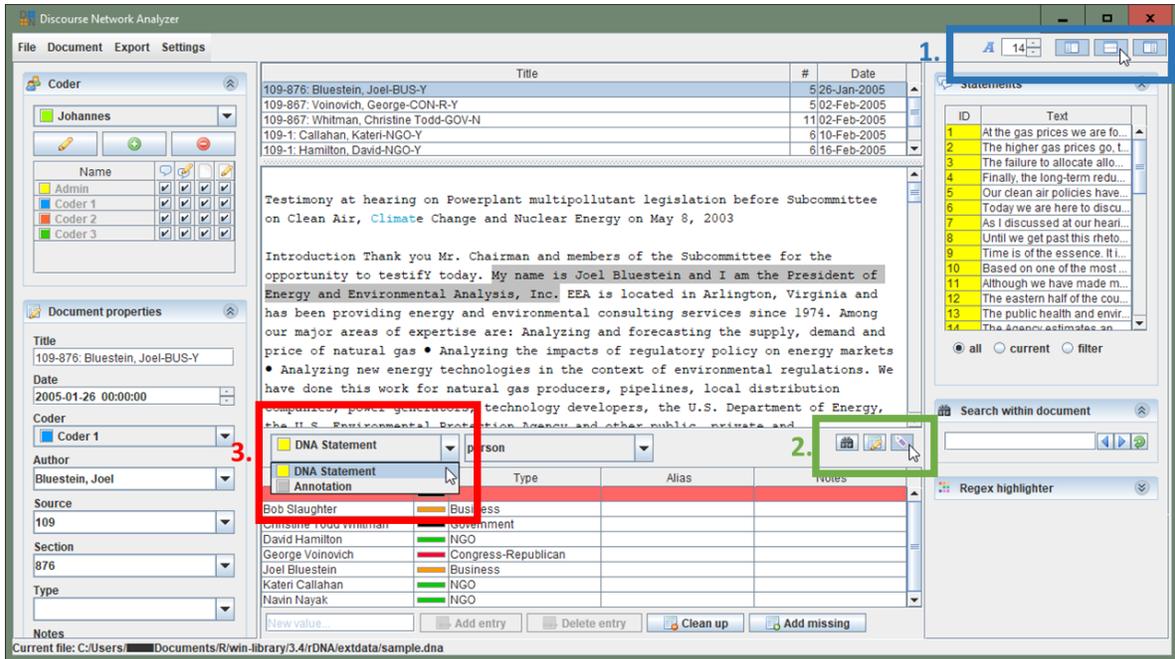


Figure 6.6: Statements window in detail

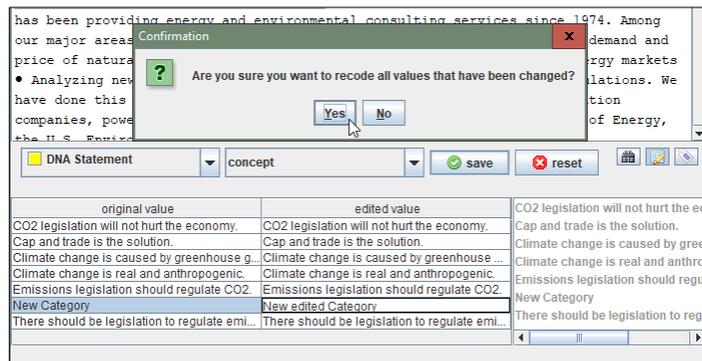


Figure 6.7: Recode a whole concept label instead of individual statement

6.4 Using the Regex Highlighter and Search Function

The last part of this section about coding statements in DNA describes how you can use the “Regex highlighter” and “Search within document” functions to help you quickly identify relevant parts of the documents you will be coding. Both functions are located on the right panel of the main window and can be folded or unfolded like every window in DNA (see Figure 6.8).

We begin with the `Search within document` menu as it is what most people are already familiar with. By entering a search term in the text field, you can jump between all occurrences of that keyword in the current document in the same way as it works in popular text processing programs (such as MS Word) or in your web browser. You can test this by entering the term “greenhouse effect” and browse through the occurrences in Navin Nayak’s hearing in the U.S. Congress (that is the second to last document in the database) with the left and right arrow buttons. As you might notice, all mentions of the greenhouse effect are fairly close together and Coder 1 has formatted a large part the section they are mentioned as DNA Statement. This is what will often happen during coding: as authors of a text or speakers will order their document or speech around sub-topics, the statements you wish to code can be close together. If you already know the most important keywords, it might therefore not be necessary to closely read the whole article or speech but can be more fruitful and efficient to jump between potential statements and only skim the remaining lines.

One feature which sets the search function in DNA apart from other programs with which you are already familiar is again that it can take Regex expressions. One useful feature of Regex commands in this scenario is that you can use multiple keywords at the same time by putting the vertical bar character `|`, which represents the *OR* operator between them. You find this character next to the Shift button on your keyboard. So for instance, you could write “global warming|climate change” to browse through all mentions of either of those words in the current document. Combining multiple keywords can serve as a quick way to browse through documents which are yet to code or which have already been coded to identify the most relevant parts.

The “*Regex highlighter*” works in much the same way. But instead of browsing through the occurrences of a key term, you can change its font colour to a colour of your choice. Looking again at Figure 6.8, you can see that multiple words have a different font colour: “dioxide” is orange, “greenhouse” is green, “clima” is blue and so on. This can help coders to notice when theory induced keywords appear in a paragraph. Again, it can be of help if you know a few basics about regular expressions, but as you can see, most of the keywords are written in plain English and you can use that as well if you prefer. However, consider that instead of the regular expression `C[Oo0]2` you would have to enter both “CO2” and “C02”. The Regex highlighter, just as the search function, is by default case insensitive. If you wish to highlight only specific cases of a word, you can put the option `(?-i)` at the beginning of your search string.

This concludes the section about how to code statements in DNA. Even though this chapter turned out to be shorter than the two previous ones, this is where you will spend most of your time while using DNA.

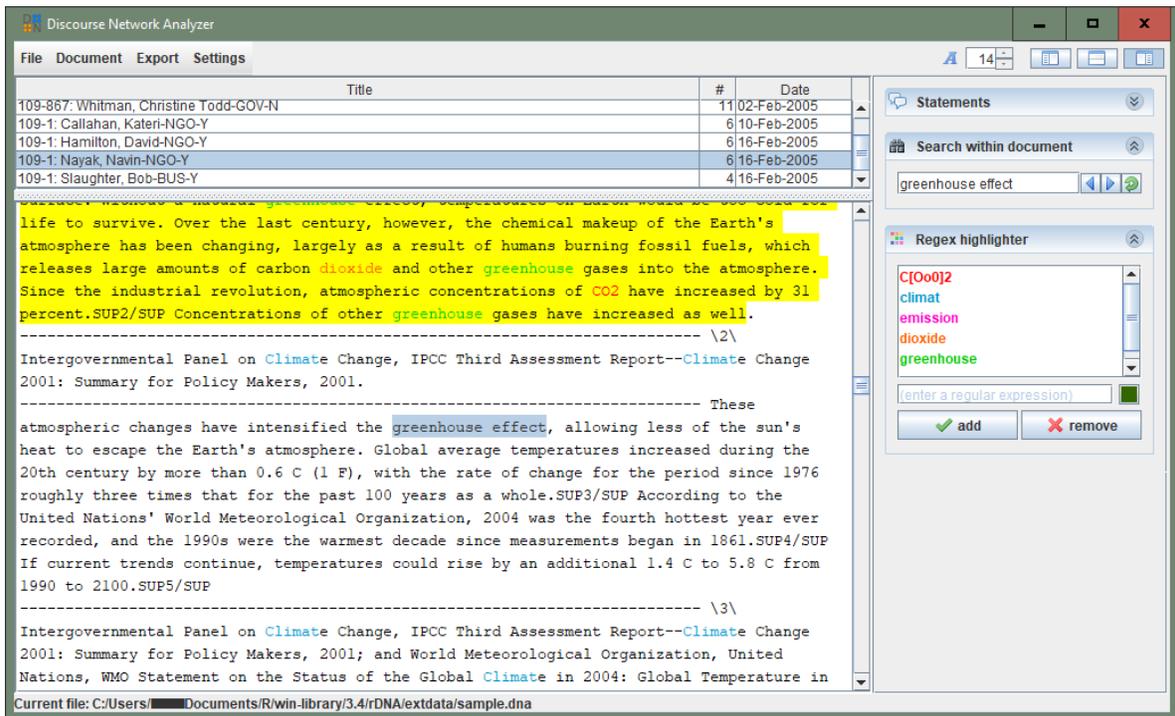


Figure 6.8: The Search and Regex highlighter functions

Chapter 7

Exporting the Coded Data

JOHANNES GRUBER

This section will walk you through the process of exporting data from DNA to continue analysis in different programs and provide detailed information about each of the 15 options you can change during export. If you intend to use `rDNA` to analyse your data—which is recommended—you will not have to export the data from DNA first as you can directly use the `.dna` database. However, `rDNA`'s function `dna_network` mirrors the options in the export window and hence it makes sense to familiarise yourself with the different ways in which networks can be exported from DNA.

To open the window, simply open the drop-down menu “Export” and click on the only entry `Export network...`. This will open the export window as depicted in Figure 7.1. The first thing you should do when you are not familiar with the different options is to tick the box `Display tooltips with instructions`. This will put some further information about each option on your screen when you rest your mouse cursor over one of them.

7.1 Type of Network

The first thing to consider is which type of network you want to export. There are three options: one-mode networks, two-mode networks and event lists.

If you select to export a “*one-mode network*”, the resulting matrix will have the same nodes in the rows and columns (e. g., organizations \times organizations) (see Table 7.1). The values in the cells in Table 7.1 are a function of how often Variable 1 and Variable 2 are referenced together in DNA Statements. These values represent the edge weights of a network. All other options left on default, in the one-mode network case, that is the sum of all products of co-references by e. e., organisation A with organisation B (see Section 2.3.4). Note, that the edge weight of an organisation with itself is always zero. One could also expect that these values would be aggregations of all statements an organisation made. However, this would not make theoretical sense as there is no edge between an organisation and itself. The the choice of the value zero in cells where row and column mention the same organisation are all zero is deliberate.

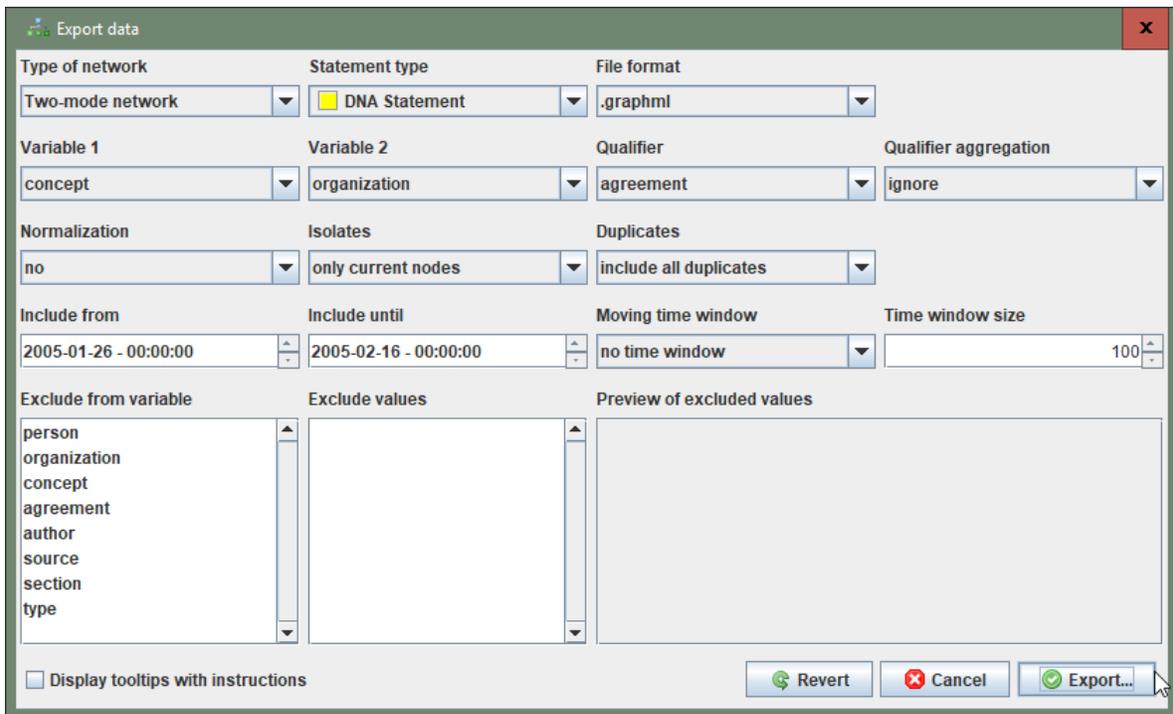


Figure 7.1: The export window

Table 7.1: One-mode network (organizations \times organizations over concept)

	Alliance to...	Energy and E...	Environmenta...	Senate
Alliance to Save Energy	0	4	10	4
Energy and Environmental Analysis, Inc.	4	0	2	6
Environmental Protection Agency	10	2	0	22
Senate	4	6	22	0

A “Two-mode network”, on the other hand, has different sets of nodes in the rows and columns of the resulting matrix (e. g., concepts \times organizations) (see Table 7.2). In this case, what you select in Variable 1 will become the rows of the matrix while Variable 2 will be the columns. In the two-

mode network case, the edge weights will be, all other options left on default, simple counts of all co-references of Variable 1 and 2.

Table 7.2: Two-mode network (concept \times organizations)

	Alliance to...	Energy and E...	Environmenta...	Senate
CO2 legislation will not hurt the economy.	0	2	1	2
Climate change is caused by greenhouse gases (CO2).	1	0	0	0
Emissions legislation should regulate CO2.	2	2	0	1
There should be legislation to regulate emissions.	1	0	10	2

The “*Event list*” is a little different from the other two options. Instead of summarising the statements over Variable 1 and 2, it simply lists all DNA Statements with each row containing all variables of a statement including the time which was set for the document a statement occurred in. The event list is thus a more detailed version of the statements window in the DNA user interface. As you can see in Table 7.3, this table is a lot larger than the matrices produced by the summarising algorithms used to produce one-mode and two node networks.

Table 7.3: Event list

id	time	docId	docTitle	docAuthor	docSource	docSection	docType	person	organization	concept	agreement
1	2005-01-26 01:00:00	1	109-876: Bluestein, Joel-BUS-Y	Bluestein, Joel	109	876		Joel Bluestein	Energy and Environmental Analysis, Inc.	CO2 legislation will not hurt the economy.	0
2	2005-01-26 01:00:00	1	109-876: Bluestein, Joel-BUS-Y	Bluestein, Joel	109	876		Joel Bluestein	Energy and Environmental Analysis, Inc.	CO2 legislation will not hurt the economy.	0
3	2005-01-26 01:00:00	1	109-876: Bluestein, Joel-BUS-Y	Bluestein, Joel	109	876		Joel Bluestein	Energy and Environmental Analysis, Inc.	Emissions legislation should regulate CO2.	0
4	2005-01-26 01:00:00	1	109-876: Bluestein, Joel-BUS-Y	Bluestein, Joel	109	876		Joel Bluestein	Energy and Environmental Analysis, Inc.	Emissions legislation should regulate CO2.	1
5	2005-02-02 01:00:00	2	109-867: Voinovich, George-CON-R-Y	Voinovich, George	109	867		George Voinovich	Senate	CO2 legislation will not hurt the economy.	0
6	2005-02-02 01:00:00	2	109-867: Voinovich, George-CON-R-Y	Voinovich, George	109	867		George Voinovich	Senate	There should be legislation to regulate emissions.	1
7	2005-02-02 01:00:00	2	109-867: Voinovich, George-CON-R-Y	Voinovich, George	109	867		George Voinovich	Senate	CO2 legislation will not hurt the economy.	0
8	2005-02-02 01:00:00	2	109-867: Voinovich, George-CON-R-Y	Voinovich, George	109	867		George Voinovich	Senate	Emissions legislation should regulate CO2.	0
9	2005-02-02 01:00:00	2	109-867: Voinovich, George-CON-R-Y	Voinovich, George	109	867		George Voinovich	Senate	There should be legislation to regulate emissions.	1
10	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
11	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
12	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
13	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
14	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
15	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
16	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	CO2 legislation will not hurt the economy.	0
17	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
18	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
19	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
20	2005-02-02 01:00:00	3	109-867: Whitman, Christine Todd-GOV-N	Whitman, Christine Todd	109	867		Christine Todd Whitman	Environmental Protection Agency	There should be legislation to regulate emissions.	1
21	2005-02-10 01:00:00	4	109-1: Callahan, Kateri-NGO-Y	Callahan, Kateri	109	1		Kateri Callahan	Alliance to Save Energy	Emissions legislation should regulate CO2.	1
22	2005-02-10 01:00:00	4	109-1: Callahan, Kateri-NGO-Y	Callahan, Kateri	109	1		Kateri Callahan	Alliance to Save Energy	There should be legislation to regulate emissions.	1
23	2005-02-10 01:00:00	4	109-1: Callahan, Kateri-NGO-Y	Callahan, Kateri	109	1		Kateri Callahan	Alliance to Save Energy	Climate change is caused by greenhouse gases (CO2).	1
24	2005-02-10 01:00:00	4	109-1: Callahan, Kateri-NGO-Y	Callahan, Kateri	109	1		Kateri Callahan	Alliance to Save Energy	Emissions legislation should regulate CO2.	1

7.2 Statement Type

There are usually two options for “*Statement type*” and you will probably never want to choose the second one—Annotations. The default is to export DNA Statements instead, which are the type that actually contains information worth exporting as a network. If you do wish to export your annotations, you will notice that the choices in Variable 1 and 2 are reduced to just the document variables: author, source, section and type. This is because the other variables, like concept or organization, do not exist in the type “Annotation”. Thus the only option that really makes sense is to export annotations as event lists.

Choosing the statement type makes the most sense if you have created your own types while setting up the database (see Section 4.3). As outlined there, very few scenarios exist in which you have more than one relevant statement type.

7.3 File Format

There are three different file formats which can be used for export at the moment: .csv, .dl and .graphml. Alternatively, the data can be brought to R using the `rDNA` package to import data directly from a .dna file (see Chapter 8).

The .csv format (short for comma-separated values) is basically a spreadsheet format in a text file. If you open it with a text editor, it will look something like this:

```
"";"Alliance to...";"Energy and E...";"Environmenta..."; "National Pet...";...
"CO2 legislation will not hurt the economy.";0;2;1;2;2;2;1
"Cap and trade is the solution.";0;0;0;0;0;0;1
"Climate change is caused by greenhouse gases (CO2).";1;0;0;0;0;1;1
"Climate change is real and anthropogenic.";0;0;0;0;0;1;2
"Emissions legislation should regulate CO2.";2;2;0;0;1;1;1
"There should be legislation to regulate emissions.";1;0;10;2;2;1;0
```

This is the exact same table as Table 7.2¹ but instead of rows and columns, the values are stored in plain text, with each line containing one row of the table and the values separated by semicolons. You can usually open these files in MS Excel, LibreOffice, or Numbers (Apple’s spreadsheet application) or other spreadsheet applications without any further steps. Note, however, that your spreadsheet application sometimes needs to be set to use semicolons to separate values instead of commas.²

.dl files are for use with the network analysis software `Ucinet`. Again, information in .dl files is stored in plain text and you can open the files using the text editor of your choice if you want to inspect the format in detail. Taking again Table 7.2 as an example, the respective .dl file looks like this:

```
dl nr = 6, nc = 7, format = fullmatrix
row labels:
"CO2 legislation will not hurt the economy."
"Cap and trade is the solution."
"Climate change is caused by greenhouse gases (CO2)."
"Climate change is real and anthropogenic."
"Emissions legislation should regulate CO2."
```

¹Except that column names needed to be truncated to fit on this page.

²This is usually only a problem in MS Excel running on a Windows computer where the CSV separator is set system-wide in the language options. If you experience problems, you need to use the import function in Excel rather than opening the .csv file directly.

```

"There should be legislation to regulate emissions."
col labels:
"Alliance to Save Energy"
"Energy and Environmental Analysis, Inc."
"Environmental Protection Agency"
"National Petrochemical & Refiners Association"
"Senate"
"Sierra Club"
"U.S. Public Interest Research Group"
data:
 0 2 1 2 2 2 1
 0 0 0 0 0 0 1
 1 0 0 0 0 1 1
 0 0 0 0 0 1 2
 2 2 0 0 1 1 1
 1 0 10 2 2 1 0

```

Ucinet is a Windows application which also runs on Mac and Linux if you use it with and emulators such as Wine or Bootcamp. A 60-day free trial version with all features included is available for free on their website: <https://sites.google.com/site/ucinetsoftware/downloads>. However, we advise to use either **visone** or R for visualising your networks as both are free, under active development and—in the case of R—open source.

The `.graphml` format is based on the open standard XML and while still readable by humans, the output file for Table 7.2 would be too long to display it here. `.graphml` files can be opened using **visone**, which is “a software tool intended for research and teaching in social network analysis” (see <http://visone.info/html/about.html>). Since it is a Java program like DNA itself, it can be run on any operating system capable of installing Java—which means basically all of them. You can take a glimpse at its visualisation capabilities in Figure 7.2, which shows the plot of Table 7.2.

7.4 Variable 1 and 2

As mentioned earlier, Variable 1 and 2 do different things depending on whether you select one-mode or two-mode under **Type of network**. In a one-mode network, Variable 1 will contain the node class used both for the rows and columns of the matrix. For example, select the variable for organizations in order to export an organization \times organization network such as shown in Table 7.1. Variable 2 in a one-node network will denote the variable through which the edges are aggregated (i. e., the values in the cells). For example, if you export a one-mode network of organizations, what makes the most sense is to aggregate how often they co-reference the same concept. That means that the second variable should be set to `concept`.

In a two-mode network, the first variable denotes the node class for the rows, while the second variable denotes the node class used for the columns of the resulting network matrix. For Table 7.2, for example, we set Variable 1 to `concept` and Variable 1 to `organization`. Instead of seeing which concepts were referenced by which organisation, we could also create a table that displays person \times concept. To do that we simply set Variable 1 to `concept` and Variable 2 to `person`. To make the logic underlying the use of the two variables even more clear we can also choose to create a somewhat redundant two-mode matrix. By setting Variable 1 to `organization` and Variable 2 to `person` we can count how often statements reference organisations and persons together. Since every person belongs to just one organisation, the resulting matrix will count all statements by a person in one cell of her/his column while the result of all other cells in her/his column is zero:

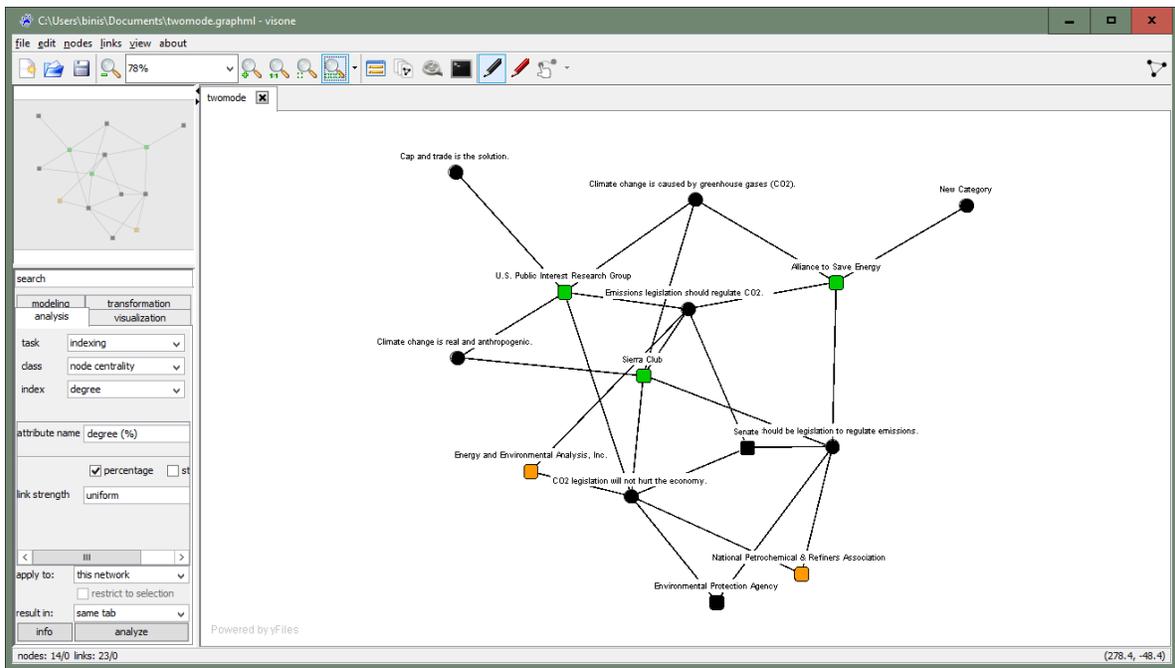


Figure 7.2: Plotting the network portrayed in Table 7.2 in Visone

Table 7.4: Two-mode network (variable1 = organization \times variable2 = person)

	Christine To...	George Voino...	Joel Blueste...	Kateri Calla...
Alliance to Save Energy	0	0	0	4
Energy and Environmental Analysis, Inc.	0	0	4	0
Environmental Protection Agency	11	0	0	0
Senate	0	5	0	0

When you look at the column with Christine Todd Whitman in above's table, you can see that all of her statements co-reference the Environmental Protection Agency. This is, therefore, the organisation she belongs to. A table like that would be rarely useful in a research project though, except maybe for checking if there had been inconsistencies during the coding process.

7.5 Qualifier and Qualifier Aggregation

The qualifier is a binary or integer variable which indicates different qualities or levels of association between variable 1 and variable 2. In many cases, the qualifier is binary, with the only possible outcomes being agreement and disagreement. However, the case could also be made to code different increments of agreement, for instance, by using a scale with -5 = strongly disagree to 5 = strongly agree. You already saw the qualifier in Section 6.1 when we created a new statement and had to tick a box in order to indicate support or rejection of the chosen concept.

The combination of qualifier and qualifier aggregation determines which algorithm will be used to calculate the edge weights (see Chapter 2). Again, this also depends on the chosen network type. In case of the event list, for instance, there is just one option: ignore. Choices for the other two network types are explained in the next two sub-sections.

7.5.1 One-mode Networks

Ignore This does the same as in two-mode networks: it ignores the qualifier and simply aggregates all co-references of Variable 1 and 2. As the ignore algorithm in DNA can be a little confusing for new users, this might be a good opportunity to complement Section 2.3.4 with an example. As you can see in Table 7.1, the edge weight between the Senate and the Sierra Club is seven. We can use Table 7.5 to calculate this number by hand.

Table 7.5: Agreement and disagreement to concepts by Senate and Sierra Club

	Cap and trad...		Climate chan...		Climate chan...		CO2 legislat...		Emissions le...		There should...	
	agree	disagree										
Alliance to Save Energy	0	0	1	0	0	0	0	0	2	0	1	0
Energy and Environmental Analysis, Inc.	0	0	0	0	0	0	0	2	1	1	0	0
Environmental Protection Agency	0	0	0	0	0	0	0	1	0	0	10	0
National Petrochemical & Refiners Association	0	0	0	0	0	0	0	0	0	0	0	0
Senate	0	0	0	0	0	0	0	2	0	1	2	0
Sierra Club	0	0										
U.S. Public Interest Research Group	0	0	0	0	0	0	0	0	0	0	0	0

Each value in Table 7.5 represents a specific value in a three-dimensional array which was described in Chapter 2. One of the x_{ijk} values is, for example, the count for $i = \text{“CO2 legislation will not hurt the economy.”}$, $j = \text{“Senate”}$ and $k = \text{disagree}$. Look in Table 7.5 to see that this specific value is 2.

You can also see that the Senate and Sierra Club co-reference three concepts: ‘‘CO2 legislation will not hurt the economy.’’, ‘‘Emissions legislation should regulate CO2.’’ and ‘‘There should be legislation to regulate emissions.’’. The other three are not co-referenced and are therefore not counted at all. For the co-referenced concepts, the edge weight is number of statements from Senate multiplied with number of statements from Sierra Club (see Equation 2.6; we are ignoring the normalisation for now):

First, we calculate $(\sum_k x_{ijk})(\sum_k x_{i'jk})$ with $i = \text{Senate}$ and $i' = \text{Sierra Club}$ for each concept j :

Concept 1:	$(0 + 0) \cdot (0 + 0) = 0$
Concept 2:	$(0 + 0) \cdot (1 + 0) = 0$
Concept 3:	$(0 + 0) \cdot (1 + 0) = 0$
Concept 4:	$(0 + 2) \cdot (2 + 0) = 4$
Concept 5:	$(0 + 1) \cdot (1 + 0) = 1$
Concept 6:	$(2 + 0) \cdot (1 + 0) = 2$

Then we solve $\sum_{j=1}^n$:

$$0 + 0 + 0 + 4 + 1 + 2 = 7$$

Congruence This option is only available for one-mode networks. It means that only similarity or matches on the qualifier variable are counted in order to construct an edge. In case of a binary second variable (e.g., (dis-)agreement) this means that the only statements counted are those where, for example, two organisations co-support or both co-reject a concept. You can see how this changes the output when you compare Table 7.1 and Table 7.6. Focusing again on the edge between the Senate and the Sierra Club, you can see now that the value dropped from seven to two as they do not support or reject all of the same concepts. We can repeat the same calculation, this time using Equation 2.2 from Section 2.3.1:

First, we calculate $\sum_k x_{ijk}x_{i'jk}$ for each concept j and $k = \text{agreement}$ as well as $k = \text{disagreement}$:

Concept 1:	$0 \cdot 0 + 0 \cdot 0 = 0$
Concept 2:	$0 \cdot 1 + 0 \cdot 0 = 0$
Concept 3:	$0 \cdot 0 + 0 \cdot 1 = 0$
Concept 4:	$0 \cdot 2 + 2 \cdot 0 = 0$
Concept 5:	$0 \cdot 1 + 1 \cdot 0 = 0$
Concept 6:	$2 \cdot 1 + 0 \cdot 0 = 2$

Then we solve $\sum_{j=1}^6$:

$$0 + 0 + 0 + 0 + 0 + 2 = 2$$

Table 7.6: One-mode congruence network (organizations \times organizations over concept)

	Alliance to...	Energy and E...	Environmenta...	Senate
Alliance to Save Energy	0	2	10	2
Energy and Environmental Analysis, Inc.	2	0	2	5
Environmental Protection Agency	10	2	0	22
Senate	2	5	22	0

With an integer qualifier variable, the inverse of the absolute distance plus one (i. e., the proximity) is used instead of a match.

Conflict Conflict is basically the opposite of congruence: while in the congruence network the matches (i. e., co-supported or co-rejected concepts) are counted, conflict counts only non-matches. With a binary qualifier, this means that organizations A and B are connected if one of them supports a concept and the other one rejects the concept. We can see how this plays out in Table 7.7. Taking the same example as above we can again calculate the edge weight of Senate and Sierra Club using Equation 2.4 from Section 2.3.2:

First, we calculate $\sum_k x_{ijk}x_{i'jk'}$ for each concept j . To make that easier to follow, we calculate $x_{ijk}x_{i'jk'}$ individually for the two possible cases: $k = \text{agreement}$ and $k' = \text{disagreement}$ as well as $k = \text{disagreement}$ and $k' = \text{agreement}$:

Concept 1 ($k = \text{agreement}; k' = \text{disagreement}$):	$0 \cdot 0 = 0$
Concept 1 ($k = \text{disagreement}; k' = \text{agreement}$):	$0 \cdot 0 = 0$
Concept 2 ($k = \text{agreement}; k' = \text{disagreement}$):	$0 \cdot 0 = 0$
Concept 2 ($k = \text{disagreement}; k' = \text{agreement}$):	$0 \cdot 1 = 0$
Concept 3 ($k = \text{agreement}; k' = \text{disagreement}$):	$0 \cdot 0 = 0$
Concept 3 ($k = \text{disagreement}; k' = \text{agreement}$):	$0 \cdot 1 = 0$
Concept 4 ($k = \text{agreement}; k' = \text{disagreement}$):	$0 \cdot 0 = 0$
Concept 4 ($k = \text{disagreement}; k' = \text{agreement}$):	$2 \cdot 2 = 4$
Concept 5 ($k = \text{agreement}; k' = \text{disagreement}$):	$0 \cdot 0 = 0$
Concept 5 ($k = \text{disagreement}; k' = \text{agreement}$):	$1 \cdot 1 = 1$
Concept 6 ($k = \text{agreement}; k' = \text{disagreement}$):	$2 \cdot 0 = 0$
Concept 6 ($k = \text{disagreement}; k' = \text{agreement}$):	$0 \cdot 1 = 0$

Then we solve $\sum_{j=1}^6 \sum_k$:

$$(0 + 0) + (0 + 0) + (0 + 0) + (0 + 4) + (0 + 1) + (0 + 0) = 5$$

Table 7.7: One-mode conflict network (organizations × organizations over concept)

	Alliance to...	Energy and E...	Environmenta...	Senate
Alliance to Save Energy	0	2	0	2
Energy and Environmental Analysis, Inc.	2	0	0	1
Environmental Protection Agency	0	0	0	0
Senate	2	1	0	0

Note, that the result of the congruence network plus the result of the conflict network equals the outcome of the ignore algorithm. This is because the difference between congruence and conflict is the factor which is ignored in the ignore network and all cases are added up.

With an integer qualifier, the absolute distance is used instead (see Section 2.3.2).

Subtract If the ignore network can be defined as adding up the cases on congruence and conflict, subtract is the exact opposite: to calculate it, a congruence network and a conflicting network are created separately and then the conflict network ties are subtracted from the congruence network ties. Taking the example from above, the edge between Senate and Sierra Club would be calculated using above results:

<p>Congruence: $y_{ii'}^{\text{congruence binary}} = 2$ Conflict: $y_{ii'}^{\text{conflict binary}} = 5$ Therefore:</p> $2 - 5 = -3$
--

See Table 7.8 to check that this is correct. The meaning of this outcome is that Senate and Sierra Club do have in fact more differences than they have in common. You can also see that this is true for several of the other organisation pairs, suggesting a quite controversial discourse.

Table 7.8: One-mode subtract network (organizations \times organizations over concept)

	Alliance to...	Energy and E...	Environmenta...	Senate
Alliance to Save Energy	0	0	10	0
Energy and Environmental Analysis, Inc.	0	0	2	4
Environmental Protection Agency	10	2	0	22
Senate	0	4	22	0

7.5.2 Two-mode (Affiliation) Networks

Ignore The agreement qualifier variable is ignored, i.e., the network is constructed as if all values on the qualifier variable were the same. In the most common case, the qualifier being (dis-)agreement, this means that co-references of a concept are aggregated, no matter if persons or organisations agree on them or not (see Table 7.1). The way in which the edge weights are constructed was already mentioned in Section 2.5. In the simplest case, a two-mode network with a binary qualifier, the edge weights for ignore are simple counts of co-references of variable 1 and 2.

Subtract If the qualifier is a regular binary (dis-)agreement value, all disagreeing statements will be subtracted from all agreeing statements. For example, if an organisation mentions a concept two times in a positive way and three times in a negative way, there will be an edge weight of -1 between the organization and the concept. You can see how this plays out for the sample file in Table 7.9. The *National Petrochemical & Refiners Association*, for example, disagreed twice to both “*CO2 legislation will not hurt the economy*” and “*There should be legislation to regulate emissions.*” resulting in edge weights of -2 for both concepts. If the qualifier is an integer value, subtract basically does the same: the absolute values of the negative qualifier codes are subtracted from the positive ones. In the example from above, if an organisation agrees somewhat (+1) to a concept twice and then disagrees strongly (-2) to the same concept twice, the edge weight would be -2.

Table 7.9: Two-mode network (concept \times organizations) with disagreements subtracted from agreements

	CO2 legislat...	Climate chan...	Emissions le...	There should...
Alliance to Save Energy	0	1	2	1
Energy and Environmental Analysis, Inc.	-2	0	0	0
Environmental Protection Agency	-1	0	0	10
Senate	-2	0	-1	2

Combine This option is only available for two-mode networks. Instead of doing mathematical operations on the qualifier values, combine creates a set of qualitative categories. In the case of a organization \times concept matrix, there would be four possible outcomes: If an organisation neither disagrees nor agrees (i.e., never references the concept at all) the value would be 0; if the organisation always agreed on the concept whenever it mentioned it, the value is 1; if the organisation always disagreed on the concept whenever it mentioned it, the value is 2; and when they reference a statement both in a positive and negative (i.e., when their stance towards it is mixed or ambiguous), the edge value is 3. You can see this in Table 7.10. With an integer variable, this may become more complex. As more combinations are possible and hence more qualitative categories need to be created (see Section 2.5).

Table 7.10: Two-mode network (concept \times organizations) with disagreements and agreements combined

	Alliance to...	Energy and E...	Environmenta...	Senate
CO2 legislation will not hurt the economy.	0	2	2	2
Climate change is caused by greenhouse gases (CO2).	1	0	0	0
Emissions legislation should regulate CO2.	1	3	0	2
There should be legislation to regulate emissions.	1	0	1	1

7.6 Normalization

In most real-life networks, normalisation is important. The reason is that some actors are more salient than others in public discussion. Government officials in charge of a certain problem will usually be covered more intensely than public interest groups who have an opposite stance. Other actors, such as political parties, follow a membership logic according to which they are by definition more diverse than others with regard to opinions. That means that due to their activity and diversity, these actors have agreement or disagreement ties to most other actors in the network at some point. This can obscure the real structure of a network and make it much harder to identify coalitions (Leifeld 2017).

To cancel out this activity effect, one can divide the edge weight between two organisations by a function of the two organizations' activity or frequency of making statements (see Section 2.4). This normalization procedure leaves us with a network of edge weights that reflect similarity in opinions without taking into account centrality or activity. Once again, which options are available depends on the network type. Only the option `no` is available in both types—which switches off normalization and is the default value.

- **Two-mode network**

Activity This divides the edge weights through the activity of the node from the first variable. For example, in concept \times organisation network in which organisation A has made four statements in total, the edge with concept B, which A mentioned once, has the value 0.25.

Prominence This divides the edge weights through the prominence of the node from the second variable. For example, in concept \times organisation network in which concept B was mentioned eight times by all organisations and once by organisation A, the edge of A and B has the value 0.125.

- **One-mode network**

Average activity Average activity divides edge weights between first-variable nodes by the average number of different second-variable nodes they are adjacent with in a two-mode network. For example, if organization A makes statements about 20 different concepts and B makes statements about 60 different concepts, the edge weight between A and B in the congruence network is divided by 40. To achieve a better scaling, all edge weights in the resulting normalized one-mode network matrix are scaled between 0 and 1. The respective algorithm can be found in Equation 2.8.

Jaccard similarity Jaccard similarity is a similarity measure with known normalizing properties. In contrast to average activity, it divides the co-occurrence frequency by the activity count of both separate actors plus their joint activity. The algorithm used for this normalisation can be found in Equation 2.9.

Cosine similarity Cosine similarity is another similarity measure with normalizing properties. It divides edge weights by the product of the nodes' activity. Find this algorithm in Equation 2.10.

Normalisation can be used in the different networks described in Section 7.5 to correct potential biases introduced by very active nodes. By using a threshold value on the edge weights before visualising the network, normalisation can make it easier to remove low-intensity ties without discriminating against organisations with a low media profile. Furthermore, the two normalisation algorithms which are based on vector similarities (Cosine and Jaccard) prepare networks to be fed into hierarchical cluster analyses, nonmetric multidimensional scaling, or other clustering techniques that are based on distance or similarity measures in order to identify coalitions in a policy debate as an alternative to community detection. Details about normalisation can be found in Leifeld (2017). In Table 7.11 you can see how the two-mode network from Table 7.2 looks like after applying the activity normalisation algorithm.

Table 7.11: Two-mode network (organizations × concept) activity normalised

	Alliance to...	Energy and E...	Environmenta...	Senate
CO2 legislation will not hurt the economy.	0.0000000	0.4	0.2000000	0.4000000
Climate change is caused by greenhouse gases (CO2).	1.0000000	0.0	0.0000000	0.0000000
Emissions legislation should regulate CO2.	0.4000000	0.4	0.0000000	0.2000000
There should be legislation to regulate emissions.	0.0769231	0.0	0.7692308	0.1538462

7.7 Duplicates

The next option in the export window would be *Isolates*, but as you will see, it makes more sense to talk about that option last. The background of the Duplicates feature is that multiple statements from the same person or organisation referring to the same statement do not always indicate stronger agreement or disagreement. For removing duplicates, there are again several options. This time, these are independent of your other choices in the export window:

Include all duplicates By default, all statements are included in a network export.

Ignore per document This removes duplicated statements which occur in the same document. In a newspaper article, for example, the number of times an actor is quoted with a statement may be a function of the journalist’s agenda or the reporting style of the news media outlet, rather than the actor’s deliberate attempt to speak multiple times about a specific topic. In these cases it makes sense to remove all duplicated statements in this article.

Ignore per calendar week/month/year In cases in which one or several newspapers reprint interviews, quotes or reports multiple times in different documents, it might make sense to remove these artefacts over time rather than on a document level.

Ignore across date range This removes all duplicated statements in all documents in the database. Consequently, edge weights of the different networks are converted to 1 if there has been co-reference or co-rejection or 0 if there has not. Usually, `Ignore across date range` therefore converts the network into a boolean matrix. However, as you can see in Table 7.12, there is one 2 value at the edge of “Energy and Environmental Analysis, Inc.” and “Emissions legislation should regulate CO2.”. This happened because this organisation contradicted itself by expressing both agreement and disagreement during the hearings.

Again we apply this transformation to Table 7.2 to illustrate how the resulting matrix is affected:

Table 7.12: Two-mode network (organization × concept) duplicates excluded across date range

	CO2 legislat...	Climate chan...	Emissions le...	There should...
Alliance to Save Energy	0	1	1	1
Energy and Environmental Analysis, Inc.	1	0	2	0
Environmental Protection Agency	1	0	0	1
Senate	1	0	1	1

7.8 Time Series Options

The options for time series are controlled by the four fields in the fourth row of the export window. `Include from` and `Include until` can be used to narrow down the number of documents which are included. The date and time which matter here are not when the DNA Statement was coded, but when the statement was made, which means the date set in the metadata of the document it occurred in. The standard values are the time and date of the oldest and newest document in the database, respectively. If you set, for example, `Include from` to “2005-01-27 - 00:00:00” the document titled “109-876: Bluestein, Joel-BUS-Y” will be excluded as this hearing was held one day earlier.

Moving time window, when set to anything other than the default `no time window`, DNA will create multiple overlapping time slices that are moved forward along the time axis. For each time slice, the network will be constructed and exported as an individual file. The statements in the sample database were made over a short period of time which leaves us only with the short slice units—seconds, minutes, hours and days—and windows for experimenting. However, this is sufficient to emphasize how times series are exported in DNA.

You could, for example, be interested in a fast moving debate with actors changing opinions and/or sides daily. In this case, you could set `Moving time window` to `using days` and `time window size` to one. Consequently, DNA would export one network per day. These networks do not necessarily feature all organisations or concepts but will only show active nodes, as you can see in Table 7.13. Since there were only hearings on five of the selected 20 days (we excluded January 26th in the last step), the remaining 15 matrices are even completely empty.

Table 7.13: Time series two-mode networks (organizations × concept) (only three of five active days are included)

2005-02-02						
	CO2 legislat...	Emissions le...	There should...			
Environmental Protec...	1	0	1			
Senate	1	1	1			
2005-02-10						
	Climate chan...	Emissions le...	There should...			
Alliance to Save Ene...	1	1	1			
2005-02-16						
	CO2 legislat...	Cap and trad...	Climate chan...	Climate chan...	Emissions.le...	There.should...
National Petrochemic...	1	0	0	0	0	1
Sierra Club	1	0	1	1	1	1
U.S. Public Interest...	1	1	1	1	1	0

In this special case, time slices are non-overlapping since it is moved one day at a time. However, if we would select two or a higher number in `time window size`, slices would overlap by the chosen time unit. For example, if there had been hearings on the first, second and third of February, a network would be created for the first and second February and a second network would feature statements

Table 7.14: Two-mode network (organizations \times concept) with all non-NGOs excluded

	Alliance to...	Sierra Club	U.S. Public...
CO2 legislation will not hurt the economy.	0	1	1
Cap and trade is the solution.	0	0	1
Climate change is caused by greenhouse gases (CO2).	1	1	1
Climate change is real and anthropogenic.	0	1	1
Emissions legislation should regulate CO2.	1	1	1
There should be legislation to regulate emissions.	1	1	0

from the second and third. If there were more dates included in the time range, this process would continue until the end of the time period is reached. Other time units work in the same way. To get mutually exclusive (i.e., non-overlapping) time slices, the user should select them manually from the output. For example, if the window size is 10 days, you could only select file number one, eleven, 21 and so on.

Instead of time units, it is also possible to use `event time`. This will create time slices of exactly 100 statement events, for example. However, it is possible that multiple events have identical timestamps. In this case, the resulting network time slice is more inclusive and also contains those statements that happened at the same time.

7.9 Exclude from Variable

The last row in the export window includes options which can be used to exclude statements which reference certain values. This can make sense if you are, for example, only interested in a certain kind of organisation. To include only NGOs in your network, you could select organization from `Exclude from variable` and then while holding the Ctrl key select “Energy and Environmental Analysis, Inc.”, “Environmental Protection Agency”, “National Petrochemical & Refiners Association” and “Senate” from the list on the right. The preview will now show which nodes are excluded. The exported network will miss all statements which reference one of these organisations and only contains statements made by NGOs:

Another common application for this feature is to exclude concepts which are very general and do not add to the structure of the discourse network. This will be shown in Section 8.2 where one concept makes the difference between a fairly convoluted and a much more telling network visualisation.

7.10 Isolates

We saved isolates for last because now that all non-NGOs are removed from the export, it is easier to explain what this option does. If you look again at Figure 7.14, you will notice that not only are the edge weights smaller than before the exclusion, the non-NGO organisation nodes are gone as well. If one of the concepts would have never been referenced by any of the NGOs, this concept would be gone as well. This is because by default only nodes that show at least minimal activity are included in the exported networks. As you can also see in Table 7.13, exclusion of statements can lead to very

differently sized matrices. However, if you have ever manually merged matrices of different sizes, you will know that it can be tedious to get everything into the right form. In these situations, it is easier to merge multiple networks if they have the same matrix dimensions. To achieve compatibility of the matrix dimensions anyway, it is possible to include all nodes of the selected variable(s) in the whole database, irrespective of time, qualifiers, and excluded values (but without any edge weights larger than 0, i.e., as isolates). You can see this in Table 7.15 which mirrors the information of Table 7.14 but has the same dimensions as, for example, Table 7.2.

Table 7.15: Two-mode network showing columns and rows with all values (even excluded or empty ones)

	Alliance to...	Energy and E...	Environmenta...	National Pet...	Senate	Sierra Club	U.S. Public...
CO2 legislation will not hurt the economy.	0	0	0	0	0	1	1
Cap and trade is the solution.	0	0	0	0	0	0	1
Climate change is caused by greenhouse gases (CO2).	1	0	0	0	0	1	1
Climate change is real and anthropogenic.	0	0	0	0	0	1	1
Emissions legislation should regulate CO2.	1	0	0	0	0	1	1
There should be legislation to regulate emissions.	1	0	0	0	0	1	0

As became clear in this section, the many options in DNA during export allow for an effective preparation of your data for further analysis. From the created network tables it is often already possible to draw some cautious inferences. From Table 7.8, for instance, it already became clear, that there is more conflict than congruence between several of the actors, such as between Senate and Sierra Club.

Chapter 8

rDNA: Using DNA from R

PHILIP LEIFELD AND JOHANNES GRUBER

DNA can be connected to the statistical computing environment R (Team 2017) through the `rDNA` package (Leifeld and Gruber 2018). There are two advantages to working with R on DNA data.

The first advantage is replicability. The network export function of `DNA` has many options. Remembering what options were used in an analysis can be difficult. If the analysis is executed in R, commands—rather than mouse clicks—are used to extract networks or attributes from DNA. These commands are saved in an R script file. This increases replicability because the script can be re-used many times. For example, after discovering a wrong code somewhere in the `DNA` database, it is sufficient to fix this problem in the `DNA` file and then re-run the R script instead of manually setting all the options again. This reduces the probability of making errors and increases replicability.

The second advantage is the immense flexibility of R in terms of statistical modelling. Analysing DNA data in R permits many forms of data analysis beyond simple visualization of the resulting networks. Examples include cluster analysis or community detection, scaling and application of data reduction techniques, centrality analysis, and even statistical modelling of network data. R is also flexible in terms of combining and matching the data from `DNA` with other data sources.

8.1 Getting Started with rDNA

The first step is to install R—which was explained in Section 3). Installing additional R packages for network analysis and clustering, such as `statnet` (Handcock et al. 2008; Goodreau et al. 2008; Handcock et al. 2016), `xergm` (Leifeld et al. 2018, 2017), `igraph` (Csardi and Nepusz 2006), and `cluster` (Maechler et al. 2017), is recommended. Moreover, it is necessary to install and correctly set up the `rJava` package (Urbanek 2017), on which the `rDNA` package depends, and the `devtools` package (Wickham et al. 2018), which permits installing R packages from GitHub (see Section 3.4). To install the packages necessary for this section, simply execute the following commands:

```
install.packages("statnet")
install.packages("xergm")
install.packages("igraph")
install.packages("cluster")
install.packages("rJava")
install.packages("devtools")
```

Before going on, the `rDNA` package must be attached to the workspace. If you have already set up `rDNA` (see Section 3.4) you can do this with:

```
library("rDNA")
```

To ensure that the following results can be reproduced exactly, we should set the random seed in R:

```
set.seed(12345)
```

Now we are able to use the package. The first step is to initialize DNA. Out of the box, `rDNA` does not know where the `DNA.jar` file is located, but will look for the newest version of it in your working directory. We also need to register DNA with `rDNA` to use them together. To do that, you need to save the `DNA.jar` file to the working directory of the current R session. If you haven't already downloaded DNA, you can do this from R by running `dna_downloadJar()`. This will place the newest version of DNA in your working directory. Then you can initialize DNA as follows (with `dna-2.0-beta21.jar` in this example):

```
dna_init("dna-2.0-beta21.jar")  
  
# If you use the current beta version (e.g., you have just downloaded it via  
# dna_downloadJar) you can omit the file name  
dna_init()
```

After initializing DNA, we can open the DNA graphical user interface from the R command line:

```
dna_gui()
```

Alternatively, we can provide the file name of a local DNA database as an argument, and the database will be opened in DNA. For example, we could open the `sample.dna` database which comes with the package. To that, we included a convenience function called `dna_sample`. You can use this function in one of two ways: either, you use it to copy the sample database into your current working directory, from where R can easily find it, or you use it directly in the package folder or `rDNA`:

```
# copy sample.dna to your working directory and then open it via dna_gui  
dna_sample()  
dna_gui("sample.dna")  
  
# or directly open the sample database  
dna_gui(dna_sample())
```

In addition to opening the GUI, we will want to retrieve networks and attributes from DNA—which is the main purpose of `rDNA`! For this to happen, a connection with a DNA database must first be established using the `dna_connection` function, which works similar to `dna_gui`:

```
conn <- dna_connection(dna_sample())
```

The `dna_connection` function accepts a file name of the database including full or relative path (or, alternatively, a connection string to a remote MySQL database) and optionally the login and password for the database (in case a remote MySQL database is used). Details about the connection can be printed by calling the resulting object named `conn`.

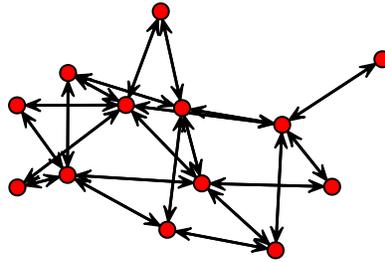
After initializing DNA and establishing a connection to a database, we can now retrieve data from DNA. We will start with a simple example of a two-mode network from the sample database. To compute the network matrix, the connection that we just established must be supplied to the `dna_network` function:

```
nw <- dna_network(conn)
```

The resulting matrix is the same that you have seen in Table 7.2. Another useful fact about the R environment is that there are functions for nearly every task you might want to perform in network analysis and it is also very well suited for plotting. The object we just created for example can easily be plotted with a single line of code in the `statnet` suite of packages:

```
# First attach the statnet package
library("statnet")

# Then simply use gplot
gplot(nw)
```



It is also easily possible to retrieve the attributes of a variable, for example the colours and types of organisations, using the `dna_attributes` function:

```
at <- dna_getAttributes(conn)
```

The result is a data frame with organizations in the rows and one column per organizational attribute:

Table 8.1: Attributes of "organization" from the sample database

id	value	color	type	alias	notes	frequency
16	Alliance to Save Energy	#00CC00	NGO			4
7	Energy and Environmental Analysis, Inc.	#FF9900	Business			4
14	Environmental Protection Agency	#000000	Government			11
25	National Petrochemical & Refiners Association	#FF9900	Business			4
11	Senate	#000000	Government			5
19	Sierra Club	#00CC00	NGO			6
22	U.S. Public Interest Research Group	#00CC00	NGO			6

The next section will provide usage examples of both the `dna_network` and the `dna_attributes` functions.

8.2 Retrieving Networks and Attributes

This section will explore the `dna_network` function and facilities for retrieving attributes in more detail. The `dna_network` function has a number of arguments, which resemble the export options in the DNA export window (see Chapter 7). The help page for the `dna_network` function provides details on these arguments. It can be opened using the command

```
help("dna_network")
```

If you are using RStudio, this will open the help window, which is by default in the lower right corner of the user interface. Instead of typing `help("Name_of_Function")` into the console you can also use `?Name_of_Function` or use the search bar in the help window in RStudio.

We will start with a simple example: a one-mode congruence network of organizations in a policy debate. We will use the same `sample.dna` database as in Chapter 7) As mentioned above, it is a small excerpt from a larger empirical research project that tries to map the ideological debates around American climate politics in the U.S. Congress over time. Accordingly, one should expect to find a polarized debate with environmental groups on one side and industrial interest groups on the other side. To compute a one-mode congruence network, the following code can be used:

```
congruence <- dna_network(conn,
  networkType = "onemode",
  statementType = "DNA Statement",
  variable1 = "organization",
  variable2 = "concept",
  qualifier = "agreement",
  qualifierAggregation = "congruence",
  duplicates = "document")
```

The result is an organization \times organization matrix, where the cells represent on how many concepts any two actors (i.e., the row organization and the column organization) had the same issue stance (by values of the qualifier variable `agreement`).

If you have read Chapter 7, you can clearly see now that the arguments of the `dna_network` function resemble the options in the DNA export window. Details on the various arguments of the function can be obtained by displaying the help page (`?dna_network`). In the code chunk above, `statementType = "DNA Statement"` indicates which statement type should be used for the network export. In this case, the statement type `DNA Statement` contains the variables `organization`, `concept`, and `agreement`. The argument `qualifierAggregation = "congruence"` causes `rDNA` to count how often the unique elements of `variable1` have an identical value on the `qualifier` variable (here: `agreement`) when they refer to a concept (`variable2`; more details can be found in Section 7.1).

If the algorithm finds duplicate statements within documents—i.e., statements containing the same organization, concept, and agreement pattern—, only one of them is retained for the analysis (`duplicates = "document"`).

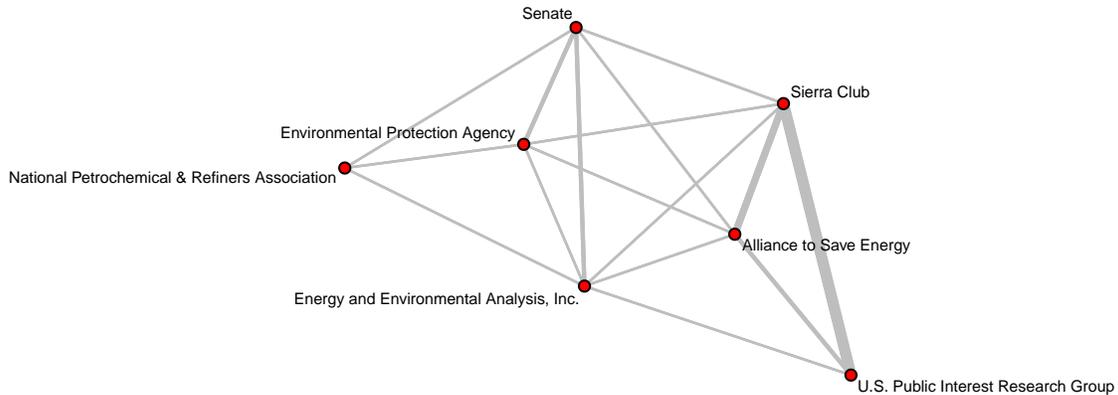
The resulting matrix can be converted to a network object and plotted as follows:

```
nw <- network(congruence)
plot(nw,
  edge.lwd = congruence^2,
  displaylabels = TRUE,
  label.cex = 0.5,
```

```

usearrows = FALSE,
edge.col = "gray"
)

```



A little background in R is needed to understand what is happening here. The plot function in R is somewhat special as it can handle a lot of different objects. Instead of trying to plot every object in the same way, R will automatically detect its class and use a specific plotting method for it. You can detect a class of the object we are plotting yourself with the function `class(nw)`. Since the class of this object is `network`, R will automatically use `plot.network`. That means that if you want to learn more about the plotting method we used above, you can simply type `help(plot.network)` into the R Console.

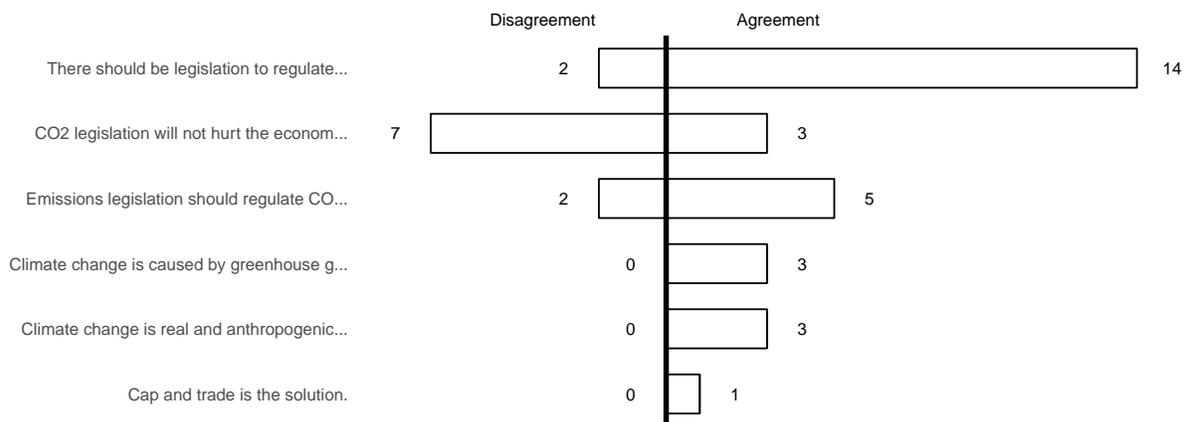
Here, we additionally used the `edge.lwd` argument of the `plot.network` function to make the line width proportional to the strength of congruence between actors. We used squared edge weights to emphasize the difference between low and high edge weights. And we also displayed the labels of the nodes at half the normal size, suppressed arrow heads, and changed the colour of the edges to grey. More information about the visualization capabilities of the `network` and `sna` packages are provided by Butts (2008a,b, 2015).

As you can see now, the network is not particularly polarized. We can suspect that some of the concepts are not very contested. If they are supported by all actors, this may mask the extent of polarization with regard to the other concepts. We can see if this is the case by plotting the agreement and disagreement towards concepts with `dna_barplot`:

```

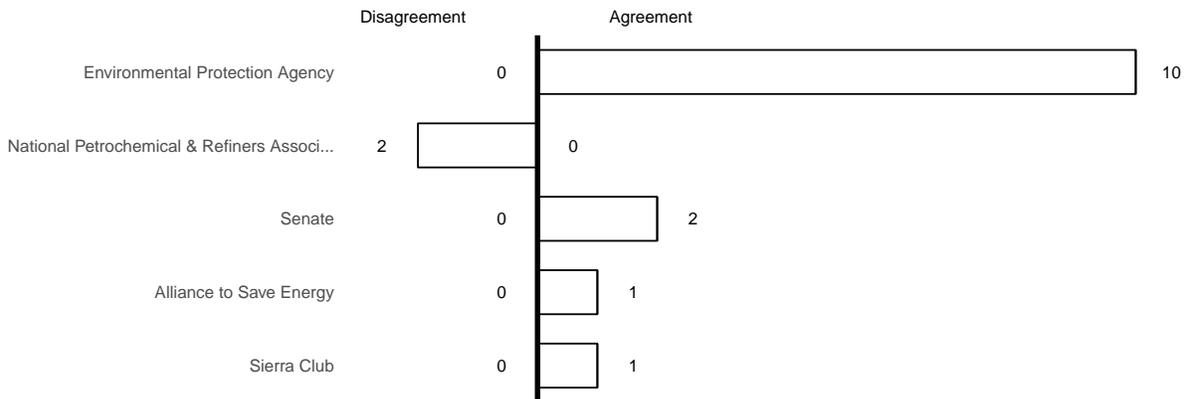
dna_barplot(conn, of = "concept", fontSize = 10)

```



It looks like the the concept “There should be legislation to regulate emissions.” is in fact very consensual. By repeating the same command with a few different options we can check if the organizations do in fact all agree on this matter:

```
dna_barplot(conn,
  of = "organization",
  fontSize = 10,
  excludeValues = list("concept" =
    "There should be legislation to regulate emissions."),
  invertValues = TRUE)
```

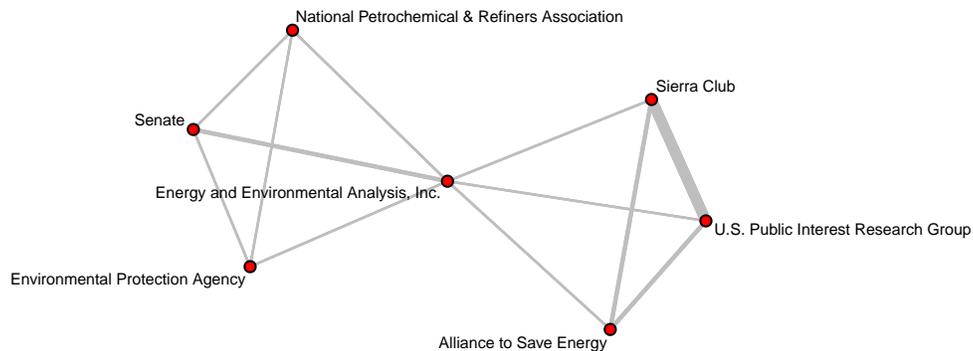


Instead of plotting agreement towards the concept, `of = "organization"` shows how often organizations agreed or disagreed with any statements. Yet, by combining `excludeValues` and `invertValues = TRUE`, we tell R to *only* regard the statement we want to look at. Again, you can get help with this function with the command `help(dna_barplot)`.

The plot shows that everyone but the “National Petrochemical & Refiners Association” agrees to the concept we chose, which indicates that consensus on the statement “There should be legislation to regulate emissions.” obfuscates the real structure of the network. Therefore we should exclude it from the congruence network. To do that, we need to export and plot the congruence network again and use the `excludeValues` argument this time:

```
congruence <-
  dna_network(conn,
    networkType = "onemode",
    statementType = "DNA Statement",
    variable1 = "organization",
    variable2 = "concept",
    qualifier = "agreement",
    qualifierAggregation = "congruence",
    duplicates = "document",
    excludeValues = list("concept" =
      "There should be legislation to regulate emissions.))
nw <- network(congruence)
plot(nw,
  edge.lwd = congruence^2,
  displaylabels = TRUE,
  label.cex = 0.5,
  usearrows = FALSE,
```

```
edge.col = "gray"
)
```



This reveals the structure of the actor congruence network much better. There are two camps revolving around environmental groups on the right and industrial interest groups and state actors on the left, with **Energy and Environmental Analysis, Inc.** taking a bridging position. The strongest belief congruence ties can be found within, rather than between, the coalitions.

Next, we should tweak the congruence network further by changing the appearance of the nodes. We can use the colours for the organization types saved in the database and apply them to the nodes in the network. We can also make the size of each node proportional to its activity. The `dna_attributes` function serves to retrieve these additional data from DNA. The result is a data frame with the relevant data for each organization in the `colour` and `frequency` columns:¹

```
at <- dna_getAttributes(conn,
  statementType = "DNA Statement",
  variable = "organization")

at

##   id          value      color      type alias notes
##  1 16      Alliance to Save Energy #00CC00      NGO
##  2  7  Energy and Environmental Analysis, Inc. #FF9900  Business
##  3 14      Environmental Protection Agency #000000 Government
##  4 25 National Petrochemical & Refiners Association #FF9900  Business
##  5 11          Senate #000000 Government
##  6 19          Sierra Club #00CC00      NGO
##  7 22  U.S. Public Interest Research Group #00CC00      NGO
##  frequency
##  1         4
##  2         4
##  3        11
##  4         4
##  5         5
##  6         6
##  7         6
```

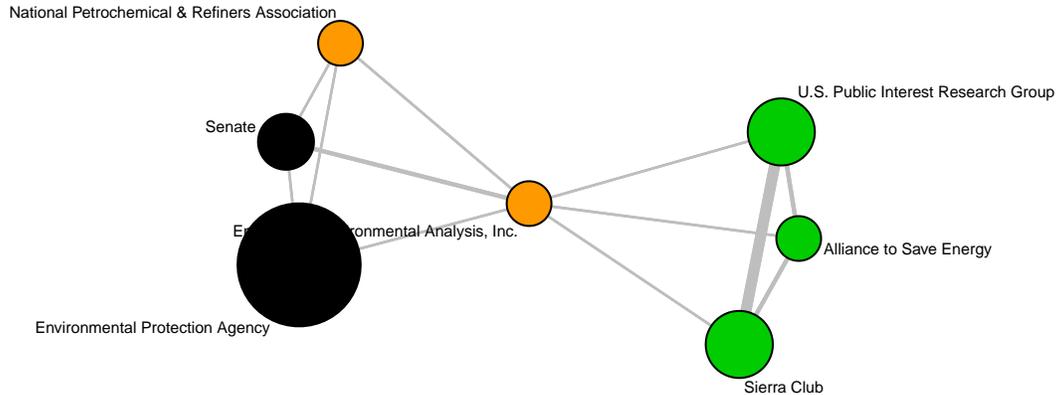
To use these data in the congruence network visualization, we can use the plotting facilities of the `plot.network` function:

¹This prints the data.frame directly to the console. If you wish to examine the data.frame in a spreadsheet-style viewer instead, you can use `View(at)`.

```

plot(nw,
     edge.lwd = congruence^2,
     displaylabels = TRUE,
     label.cex = 0.5,
     usearrows = FALSE,
     edge.col = "gray",
     vertex.col = at$color,
     vertex.cex = at$frequency
)

```



This yields a clear visualization of the actor congruence network, with simultaneous display of the network structure including its coalitions, the actors' activity in the debate, and actor types.

Another way to visualize a discourse network is a two-mode network visualization. To compute a two-mode network of organizations and concepts, the following code can be used:

```

affil <- dna_network(conn,
                    networkType = "twomode",
                    statementType = "DNA Statement",
                    variable1 = "organization",
                    variable2 = "concept",
                    qualifier = "agreement",
                    qualifierAggregation = "combine",
                    duplicates = "document",
                    verbose = FALSE)

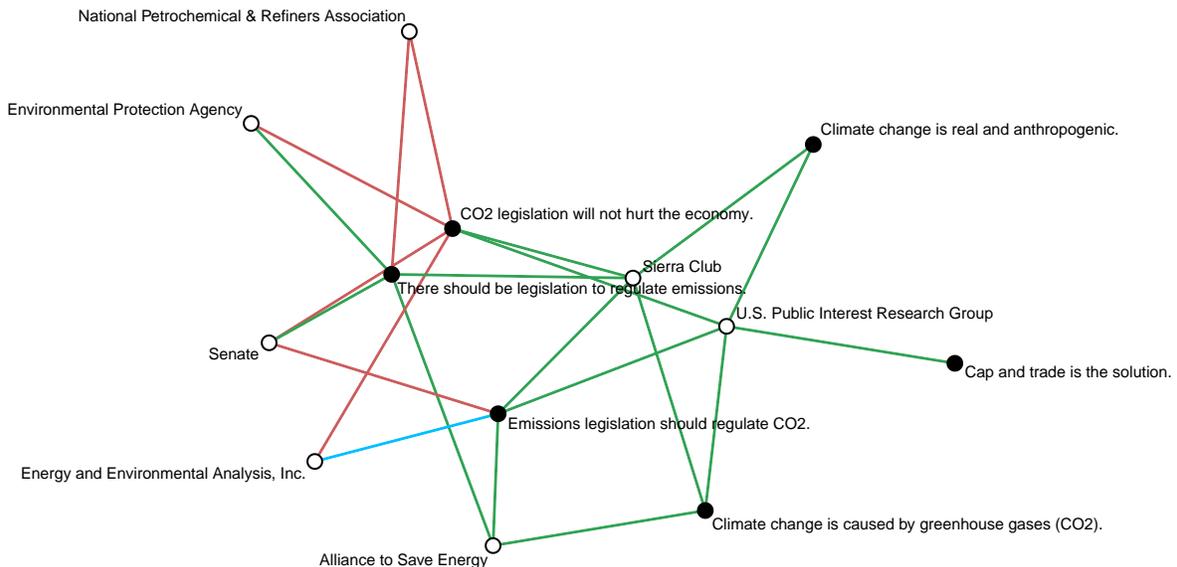
```

This creates a 7×6 matrix of organizations and their relations to concepts. The argument `networkType = "twomode"` is necessary because the rows and columns of the `affil` matrix should contain different variables. The arguments `variable1 = "organization"` and `variable2 = "concept"` define which variables should be used for the rows and columns, respectively. The arguments `qualifier = "agreement"` and `qualifierAggregation = "combine"` define how the cells of the matrix should be populated: `agreement` is a binary variable, and the `combine` option causes a cell to have a value of 0 if the organization never refers to the concept, 1 if the organization refers to the respective concept exclusively in a positive way, 2 if the organization refers to the concept exclusively in a negative way, and 3 if there are both positive and negative statements by the organization about the concept. We have covered this before in detail in Section 7.5. `rDNA` reports on the R console what each combination means.

As in the previous example, the resulting network matrix can be converted to a `network` object (as defined in the `network` package). The colours of the edges can be stored as an edge attribute, and the

resulting object can be plotted with different colours representing positive, negative, and ambivalent mentions.

```
nw <- network(affil, bipartite = TRUE)
colors <- as.character(t(affil))
colors[colors == "3"] <- "deepskyblue"
colors[colors == "2"] <- "indianred"
colors[colors == "1"] <- "#31a354"
colors <- colors[colors != "0"]
set.edge.attribute(nw, "color", colors)
plot(nw,
     edge.col = get.edge.attribute(nw, "color"),
     vertex.col = c(rep("white", nrow(affil)),
                    rep("black", ncol(affil))),
     displaylabels = TRUE,
     label.cex = 0.5
    )
```



In this example, we first converted the two-mode matrix to a bipartite `network` object, then created a vector of colours for the edges (excluding zeros), and inserted this vector into the `network` object as an edge attribute. It was necessary to work with the transposed `affil` matrix (using the `t` function) because the `set.edge.attribute` function expects edge attributes in a row-wise order while the `as.character` function returns them in a column-wise order based on the `affil` matrix. Finally, we plotted the network object with edge colours and labels. In the visualization, we used white nodes for organizations and black nodes for concepts.

A side note on colours in R: As you can see in this code chunk, there are three kinds of colours which R can process. Simple general colour names like *black* and *white*, very specific colour names like *deepskyblue* and *indianred*, and hexadecimal numbers that represent a specific mixture of red, green and blue. For finding the specific names, there is an extensive [R colour cheatsheet](#), which is also a good resource to get a quick overview of colours in R as well. Using hexadecimal numbers might seem overly complicated for the task of picking a colour for a plot at first. However, there are two reasons why you should consider it anyway. First, it is nowadays very easy to install a colour picker addon in your web browser, which gives you the opportunity to pick any colour you like on a website and use it for your

own plots in R. And second, because there is a fantastic website called <http://colorbrewer2.org/> that offers a number of very pleasant looking colour palettes which can make your plots appear more modern. The easiest way to use colours from both sources is by entering the hexadecimal numbers, preceded by #.

We can now see the opinions of all actors on the various concepts. The blue edge indicates that **Energy and Environmental Analysis, Inc.** has both positive and negative things to say about the concept “Emissions legislation should regulate CO2”. This is why this organization acts as a bridge between the two camps in the congruence network. Furthermore, we can now see more clearly that the concept we omitted in the congruence network, “There should be legislation to regulate emissions”, is viewed positively by four organizations, but still receives a negative mention by one actor. The green edges span both camps, and this caused additional connections between the two groups. The negative tie is ignored in the construction of the congruence network because conflicts are not counted and there is no second red tie to that concept.

8.3 Cluster analysis

In the last section we have seen how you can identify communities in a discourse by visually inspecting network plots. However, this becomes difficult very quickly as soon as discourse network analysis is performed on a larger database with more than just a few actors. Additionally, different levels of access to the media or other public forums can lead one organisation to be able to make many more statements than others and therefore mask the strength of ties between actors. As briefly mentioned in Section 7.6, it is therefore often fertile to perform certain clustering techniques on DNA networks to get a better grip of the characteristics of a discourse.

rDNAusers can perform this very easily, since our function `dna_cluster` already comprises options for several clustering algorithms, along with the appropriate normalisation techniques. Again you can get help on the function with `help(dna_cluster)`.

We can perform clustering directly on the connection to the sample database we have been using above and do not need to call `dna_network` first:

```
# Use the command on the connection to the sample
clust <- dna_cluster(conn)

# And simply type the name of the object to print information about it
clust

##
## Call:
## dna_cluster(connection = conn)
##
## Cluster method : ward.D2
## Distance : jaccard
## Number of objects: 7
## Used for colours :
## attribute1: "color"
## attribute2: "value"
```

As you can see from the information printed to the screen when `clust` was called, the cluster method used by default is `ward.D2`. The distance measure is set automatically, based on the properties of the network. Normally, this defaults to Euclidean distance, except if the network matrix which is

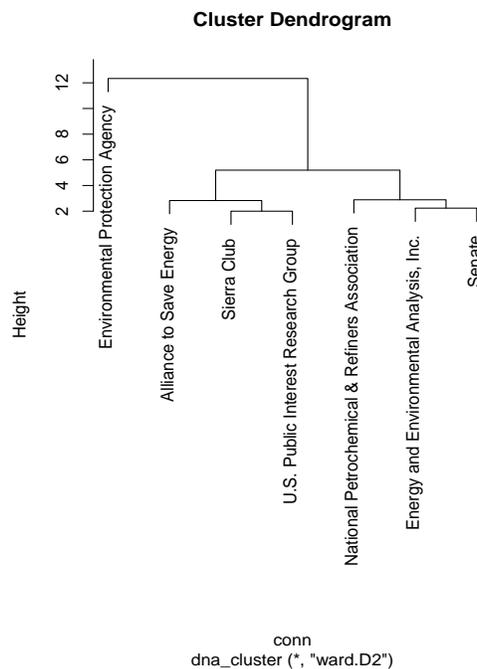
constructed in the background is binary. Since the default of `dna_cluster` is to exclude duplicated statements on the document level (`duplicates = "document"`), the resulting network matrix is binary in this case, meaning that 1 represents a connection between actors and 0 means there is no connection. This, in effect, normalises the network as it does not matter anymore how often an actor makes the same statement in the same document and since the actors in the sample database do not appear in more than one document each. In other scenarios, however, actors might make the same statement many times over different documents. In this case, you might want to use a different setting for `duplicates` argument. `duplicates = "acrosstime"`, for example, will make sure a binary network is created in all cases. If you turn the removal of duplicates off, the Euclidean distance will be used for clustering in this case and the number of statements each organisation has made will influence the clustering results:

```
clust <- dna_cluster(conn, duplicates = "include")
clust

##
## Call:
## dna_cluster(connection = conn, duplicates = "include")
##
## Cluster method : ward.D2
## Distance : euclidean
## Number of objects: 7
## Used for colours :
## attribute1: "color"
## attribute2: "value"
```

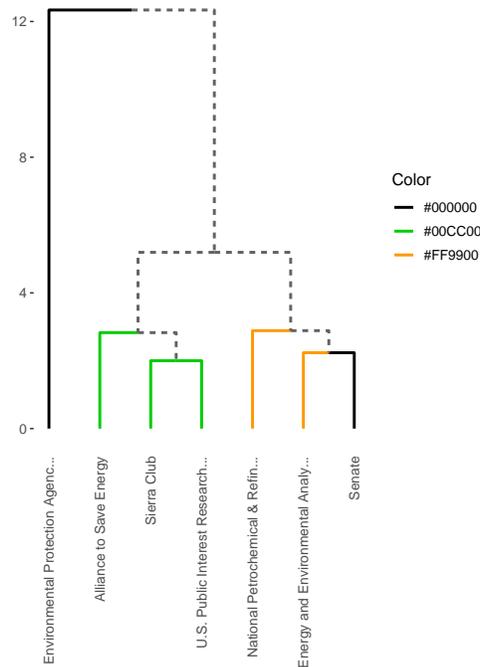
Objects created via `dna_cluster` can be directly plotted in R using the same plot command as above:

```
plot(clust)
```



`plot` produces a simple dendrogram in this case, which shows the different clusters as branches and the organisations as leaves of a tree. This plot might be informative enough in some cases, but it is neither very appealing nor is it particularly easy to customise its shape, colours or labels. This is why we have integrated a special plot function into `rDNA` to make the life of users easier and to produce arguably nicer plots which can often be directly used for publication:

```
dna_plotDendro(clust)
```



As you can see, this is essentially the same plot but the overly long labels on the x-axis were truncated (you can set the maximum number of characters using the argument `truncate =`), the leaves of the dendrogram have the colours chosen for each organisation in `DNA`, and the branches now consist of dashed lines to highlight where branches split. When you look into the help for `dna_plotDendro` you will see that the function is highly customisable and offers a range of extra features.

But first of all, what you can see from the plot is that it appears like the discussion consist of three broad clusters. “National Petrochemical & Refiners Association”, “Energy and Environmental Analysis, Inc.” and the “Senate” appear to form one cluster; the “Alliance to Save Energy”, “Sierra Club” and “U.S. Public Interest Research Group” make up the second one; while the “Environmental Protection Agency” seems to be a single community on its own. But remember, that data for this plot still includes all duplicated statements and the concept which we have identified in the last section to be so consensual, that is masks the conflict in this discussion. So we need to remove this now.

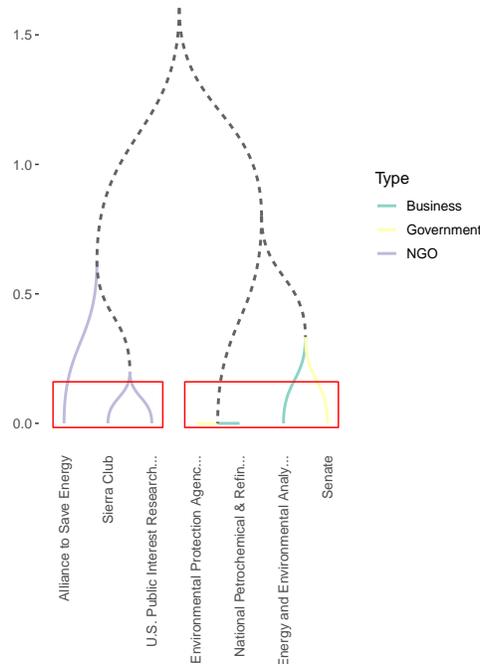
Excluding values from the clustering is not a dedicated option in `dna_cluster`, but it is very easy to do anyway. Note, the description of the `...` in `help(dna_cluster)`: “Additional arguments passed to `dna_network`.” This means that you can use nearly all arguments from `dna_network` by simply writing them within the brackets after `dna_cluster`. When we completely remove duplicates and the problematic concept as described, the resulting dendrogram looks a bit different:

```
clust <- dna_cluster(conn,
  duplicates = "acrossrange",
  attribute1 = "type",
```

```

cutree.k = 2,
excludeValues = list("concept" =
  "There should be legislation to regulate emissions.")
dna_plotDendro(clust, shape = "diagonal", colours = "brewer", rectangles = "red")

```



Now we can see that the two main branches of the dendrogram show the same communities we have identified in the network plot. As you have noticed, we have also changed some other things about the plot such as its shape, the selection of colours and we have plotted red rectangles around the two main clusters. As mentioned before, the function is highly customisable and can visualise up to four sets of information about the variables:

leaf_colours Takes the values "attribute1" or "attribute2" and colours the leaves accordingly. What values are in attribute1 and attribute2 of the object you are plotting can be set in "dna_cluster" and can be checked using the command `attributes(clust)$colours`. In this example, attribute1 was set to type, which means that the type of organisation was used to colour leaves.

leaf_ends This works in the same way as leaf_colours but instead of the leaf colours, it assigns different shapes to the line ends of the leaves.

activity This can be either turned on or off via TRUE and FALSE. If turned on, the size of the line ends will be determined by the activity of the leaf in the network—i.e., how many statements an actor made (minus the duplicated statements, if you chose to exclude them).

rectangles You can either provide a colour value to draw rectangles around groups or leave this empty to make the boxes disappear (which is the default). The group membership of each organisation in above's example is determined by providing `cutree.k = 2` in "dna_cluster". If you neither provide a value for `cutree.k` nor `cutree.h`, all leaves will belong to the same group.

Therefore it is possible to visualise a lot of different information in just one plot, which is great for publication, or if you want to use the plot to discover patterns for further analysis. The following plot shows this by employing the full potential of the function.

```

clust <- dna_cluster(conn,
  variable1 = "person",
  attribute1 = "value",
  attribute2 = "type",
  cutree.k = 2,
  excludeValues = list("concept" =
    "There should be legislation to regulate emissions.))

# Now the first attribute contains just the names of the persons
clust$attribute1

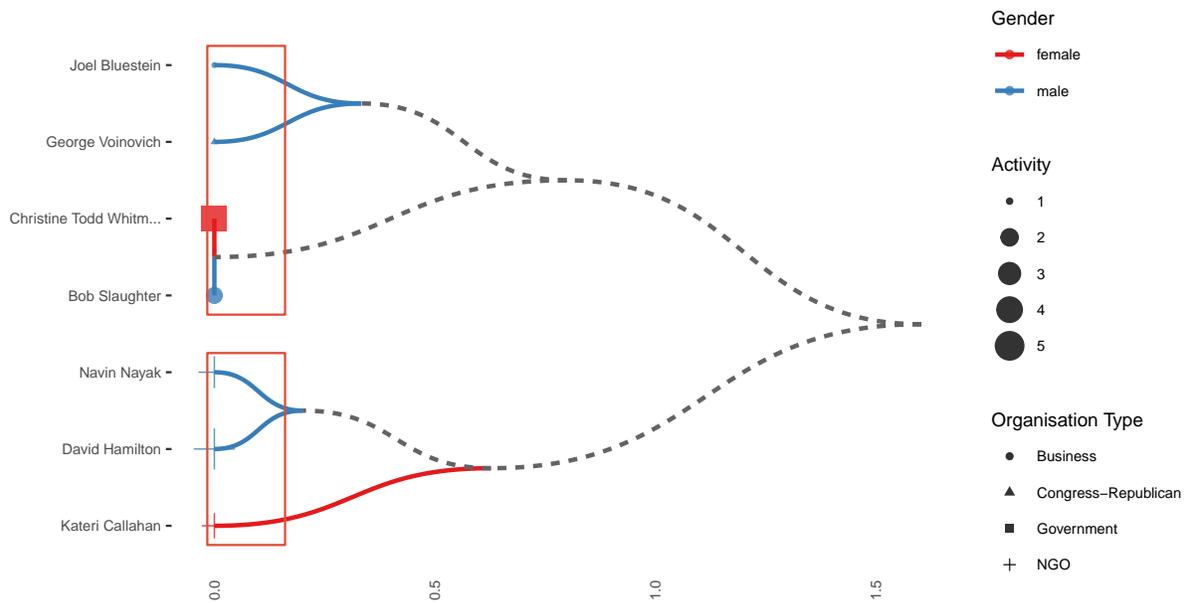
## [1] "Bob Slaughter"          "Christine Todd Whitman" "David Hamilton"
## [4] "George Voinovich"       "Joel Bluestein"        "Kateri Callahan"
## [7] "Navin Nayak"

# You can replace these values by something more informative like this
clust$attribute1 <- c("male", "female", "male", "male", "male",
  "female", "male")

# You can change the legend by changing the colours attribute
attr(clust, "colours") <- c("Gender", "Organisation Type")

# Then you are ready to plot
library("ggplot2")
dna_plotDendro(clust,
  activity = TRUE,
  shape = "diagonal",
  truncate = 20,
  leaf_colours = "attribute1",
  colours = "brewer",
  custom_colours = "Set1",
  rectangles = "#e34a33",
  leaf_ends = "attribute2",
  ends_alpha = 0.8,
  font_size = 9,
  leaf_labels = "ticks") +
  coord_flip()

```

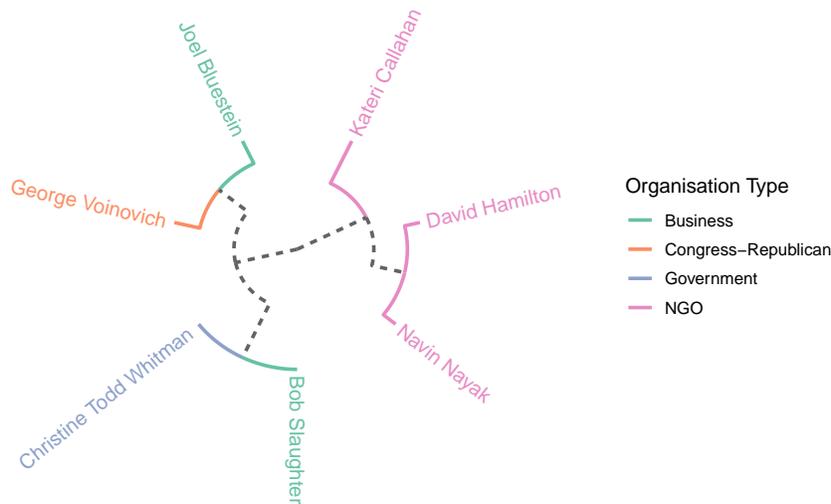


As becomes clear from this plot, the three NGOs are clustered together, while other organisations make up the second cluster. You can also see that while the NGOs are roughly the same in terms of activity, the number of statements from persons who belong to a business or government organisations or are in Congress, differ widely. Unsurprisingly, the gender of the speakers does not seem to influence in which cluster they end up, but at least now you know, how you can add this information.

You can also see in this example, how you can make further changes to the plot using commands from the `ggplot2` package (Wickham 2009): simply by adding them via `+` at the end of the plot function. `ggplot2` is very popular in the R community and you can find many tutorials online on how to produce plots with it. As `dna_plotDendro` essentially uses `ggplot2` under the hood, you can use functions from `ggplot2` and its extensions to manipulate the appearance of the dendrogram further. In this case, we added `+ coord_flip()` after the end of the call to `dna_plotDendro` to rotate the plot by 90° .

Another option might not be particularly useful with a small dataset but can work well with larger samples. When setting `circular = TRUE`, the dendrogram will be plotted as a circle, making room for more leaves without needing a large amount of horizontal space.

```
dna_plotDendro(clust,
  circular = TRUE,
  leaf_colours = "attribute2",
  leaf_labels = "nodes",
  colours = "brewer",
  custom_colours = "Set2",
  theme = "void")
```



What we did not explore so far, is that `dna_cluster` is also capable of using a range of different clustering methods besides the default `ward.D2`. First, we implemented all clustering methods from the `hclust` function in R (See `help(hclust)` for details). And second, we included the “`cluster_edge_betweenness`”, “`cluster_leading_eigen`” and “`cluster_walktrap`” algorithms from `igraph` package (Csardi and Nepusz 2006).

We can compare a few of these algorithms by running `dna_cluster` and `dna_plotDendro` multiple times and then arranging the individual plots in a grid:

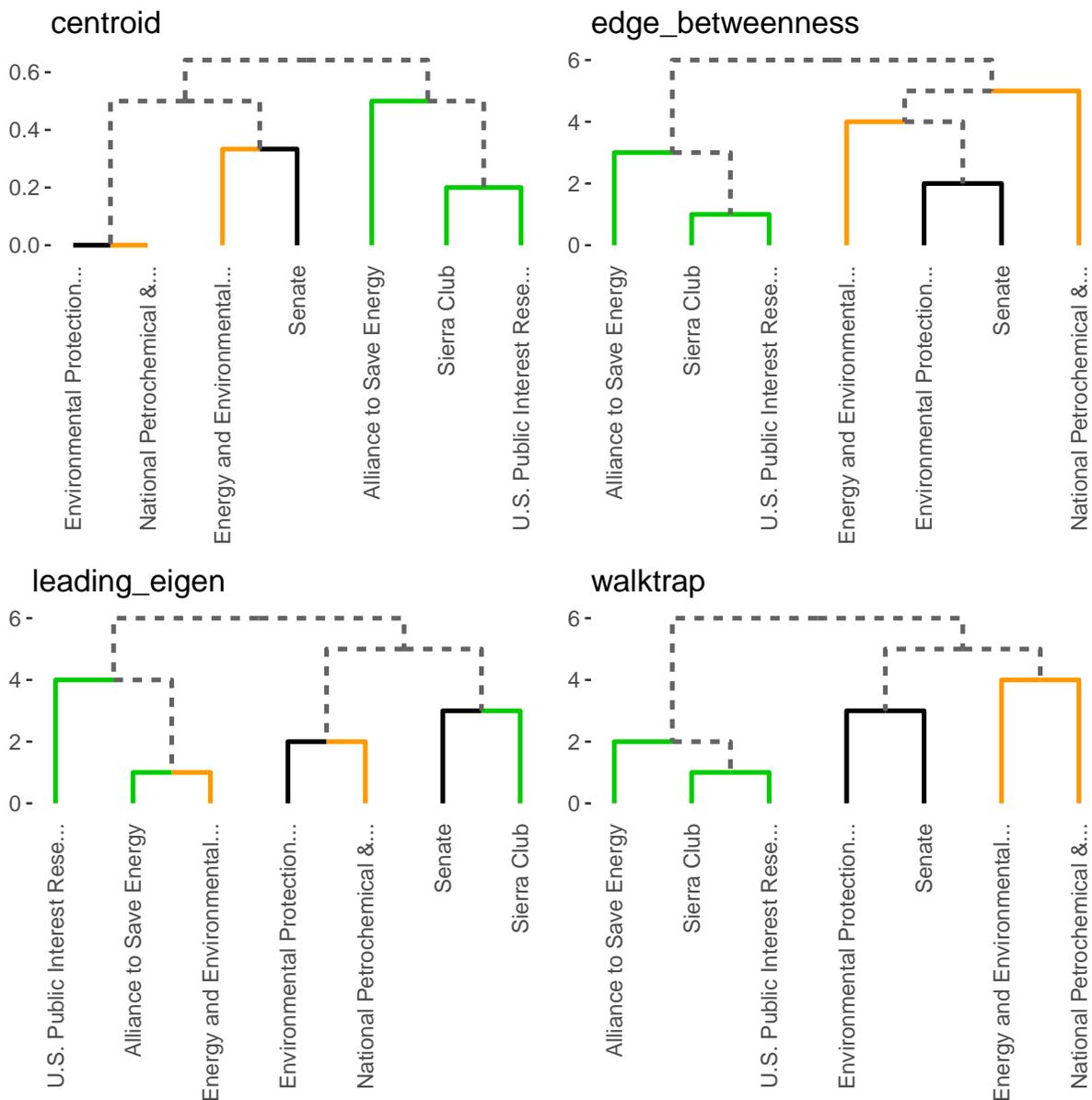
```
# We save the concept we want to exclude in a new object, so we do not have to
# repeat this line multiple times
excludeConcept <- list("concept" =
  "There should be legislation to regulate emissions.")

# Then we call dna_cluster four times with four different clustering methods
clust_centroid <- dna_cluster(conn,
  clust.method = "centroid",
  cutree.k = 2,
  excludeValues = excludeConcept)
clust_edbe <- dna_cluster(conn,
  clust.method = "edge_betweenness",
  cutree.k = 2,
  excludeValues = excludeConcept)
clust_leei <- dna_cluster(conn,
  clust.method = "leading_eigen",
  cutree.k = 2,
  excludeValues = excludeConcept)
clust_walktrap <- dna_cluster(conn,
  clust.method = "walktrap",
  cutree.k = 2,
  excludeValues = excludeConcept)

# Now we plot the four clustered objects but save the plots in objects instead
# of printing them to the screen directly
dend_centroid <- dna_plotDendro(clust_centroid, show_legend = FALSE, truncate = 25) +
  ggtitle("centroid")
```

```
dend_edbe <- dna_plotDendro(clust_edbe, show_legend = FALSE, truncate = 25)+
  ggtitle("edge_betweenness")
dend_leei <- dna_plotDendro(clust_leei, show_legend = FALSE, truncate = 25)+
  ggtitle("leading_eigen")
dend_walktrap <- dna_plotDendro(clust_walktrap, show_legend = FALSE, truncate = 25)+
  ggtitle("walktrap")

# Now we arrange the plots in a grid and save to a new object 'grid'
library("gridExtra")
grid <- grid.arrange(dend_centroid,
  dend_edbe,
  dend_leei,
  dend_walktrap)
```



As you can see, all but one of the cluster methods arrive at basically the same result: NGOs belong to one main cluster, while the other organisations form the second one. Only clustering the sample data based on the leading eigenvector algorithm from the `igraph` package arrives at a different result.

Once your plot is displayed in the plot pane of RStudio, you can use the export button to save the image as a PNG or PDF file for later use. Alternatively, you can also use another command from `ggplot2` to save the plot, which can often lead to better results than using RStudio's built-in feature:

```
ggsave(plot = grid, filename = "Cluster methods.pdf", device = "pdf",
       scale = 2, width = 10, height = 15, units = "cm")
```

You can experiment with your own data now and see how the different algorithms group your actors. However, which clustering algorithm works best with your data is a question we cannot offer advice here.

8.4 Heatmaps

Once you know which actors are clustered together, the next step of an analysis would be to determine what stances bind them together. One way to do so would be to use the `dna_network()` function to retrieve (dis-)agreement of the actors towards all the different concepts. This information is already present in objects created with `dna_cluster()` and can be retrieved from it using the `$` symbol. Instead of printing it to your console, you can use the `View` command to open it in a spreadsheet-style data viewer:

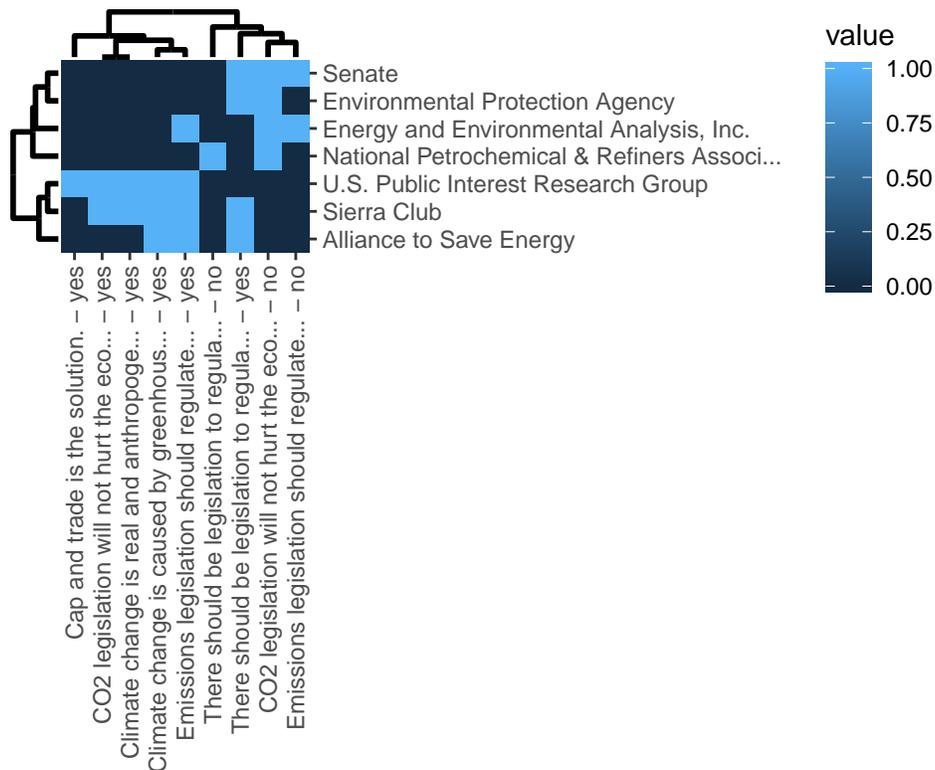
```
clust <- dna_cluster(conn)
View(clust$network)
```

	CO2 legislation will not hurt the economy. - 0	Emissions legislation should regulate CO2. - 0	There should be legislation to regulate emissions. - 0	CO2 legislation will not hurt the economy. - 1	Cap and trade is the solution. - 1	Climate change is caused by greenhouse gases (CO2). - 1	Climate change is real and anthropogenic. - 1	Emissions legislation should regulate CO2. - 1	There should be legislation to regulate emissions. - 1
Alliance to Save Energy	0	0	0	0	0	1	0	1	1
Energy and Environmental Analysis, Inc.	1	1	0	0	0	0	0	1	0
Environmental Protection Agency	1	0	0	0	0	0	0	0	1
National Petrochemical & Refiners Association	1	0	1	0	0	0	0	0	0
Senate	1	1	0	0	0	0	0	0	1
Sierra Club	0	0	0	1	0	1	1	1	1
U.S. Public Interest Research Group	0	0	0	1	1	1	1	1	0

The “- 0” or “- 1” behind each concept label stand for disagreement or agreement respectively. However, this table is hard to read, especially when there are more than just a few organisations or concepts present in the database.

One approach to make it easier to inspect the distribution of (dis-)agreement on concepts among members of each cluster are heatmap plots. Heatmaps are especially helpful to get a quick overview of a matrix and see where values are high or low. Combined with the dendrograms introduced in the last section, they can thus be helpful to get an insight into which concepts determine the clustering. The function to do this in `rDNA` is called `dna_plotHeatmap` and takes objects that were created with `dna_cluster`—just as `dna_plotDendro`:

```
clust <- dna_cluster(conn)
dna_plotHeatmap(clust)
```

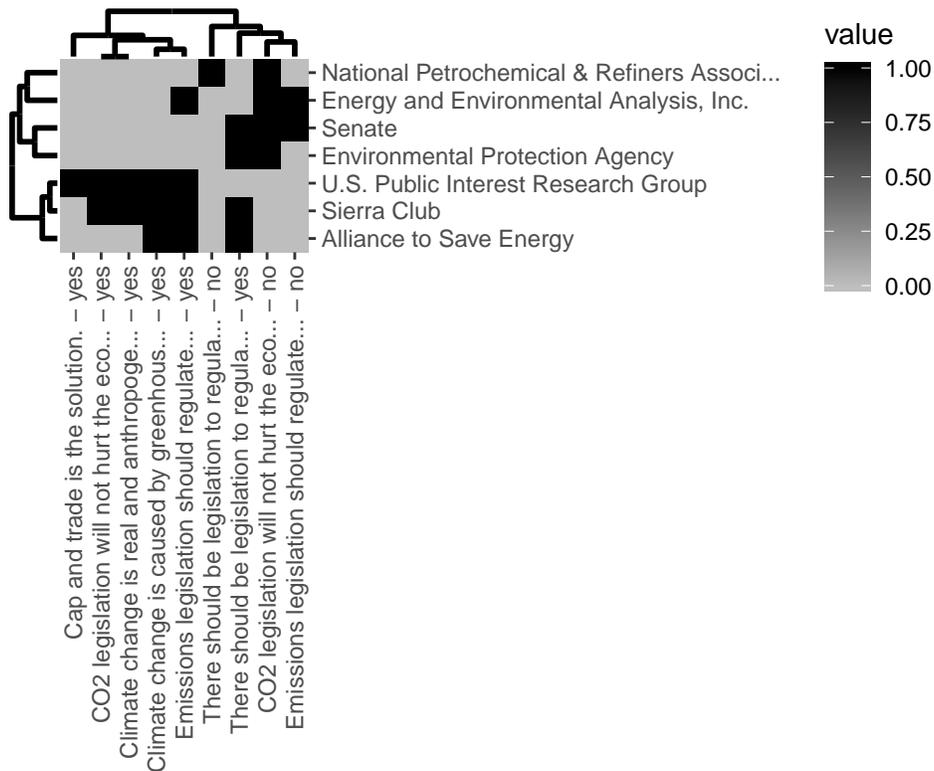


The dendrograms on the x- and y-axis of the heatmap show how organisations and concepts are grouped together by the clustering algorithm used to construct the object `clust`. As you can see, the main feature the NGOs have in common is their agreement to many of the concepts, while the other organisations seem to be more cautious in stating agreement towards any of the statements.

Since the heatmap is ordered by the dendrograms on its axis, changing the clustering algorithm will also change the heatmap. This makes it easy to compare how concepts and actors are clustered together. As many users—and some publishers—may prefer grayscale plots, we also use the following plot to demonstrate how to accomplish that with `dna_plotHeatmap`:

```
clust <- dna_cluster(conn,
                     clust.method = "walktrap")
dna_plotHeatmap(clust,
                colours = "gradient",
```

```
custom_colours = c('gray', 'black')
```



Using the walktrap algorithm changed the order of the four upper organisations and now grouped business and government organisations together. Note, however, that the dendrogram on the x-axis of the plot hasn't changed. This is because the concepts cannot be clustered using the "walktrap" algorithm but are instead ordered based on the default value `ward.D2`. If you prefer not doing that, you have the option to turn off the drawing of the dendrogram on the x-axis completely by using the parameter `dendro_x = FALSE`.

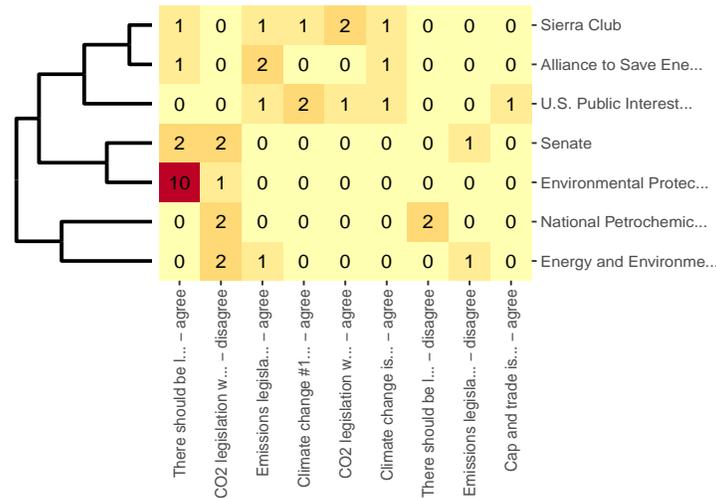
As you can see, by setting `colours` as `gradient` and providing two colour values, you can change the colour for low (first colour) and high (second colour) values in the underlying matrix. Providing more than two values within `c()`, will still lead to the first value being used for low values and the last value being used for high values. All additional colour values in between will also be used to construct a gradient of colours.

Alternatively, `colours` takes the setting `brewer` which will automatically choose a pleasant set of colours for the heatmap. In this case, `custom_colours`—if used—needs to be a name of one of the available colour palettes from the `RColorBrewer` package (Neuwirth 2014). You can display the available options using the R command `RColorBrewer::display.brewer.all()`.

As you will have noted, the labels of the concept are separated by (dis-)agreement and the words “yes” and “no” are added at the end. You can control which suffixes to use for the different qualifier levels by providing a list with “translations” of each level as you can see below. The level "0" will be replaced by “disagree” and "1" by “agree”. More levels can be named in the same fashion by adding more entries to this list. If your qualifier levels have a different meaning, the default values should always be changed as `rDNA` does not detect the meaning automatically.

As was mentioned before, `dna_clust` removes duplicates on the document level by default, leaving the dataset—in case of the sample—as a binary. If the heatmap is not binary, it might make sense to display the exact values inside the plot by setting `values = TRUE`.

```
clust <- dna_cluster(conn,
  duplicates = "include",
  clust.method = "walktrap")
dna_plotHeatmap(clust,
  values = TRUE,
  truncate = 20,
  colours = "brewer",
  custom_colours = "YlOrRd",
  dendro_x = FALSE,
  dendro_y_size = 0.4,
  qualifierLevels = list("0" = "disagree",
    "1" = "agree"),
  show_legend = FALSE)
```



Consult `help(dna_plotHeatmap)` to see what other options you can set. For example, read up on the `dendro_y_size` option we used above, to see what it means. Also note, that you can use the `...` argument to pass arguments to `dna_plotDendro`. Try using a different shape argument, for example, to see how this would work.

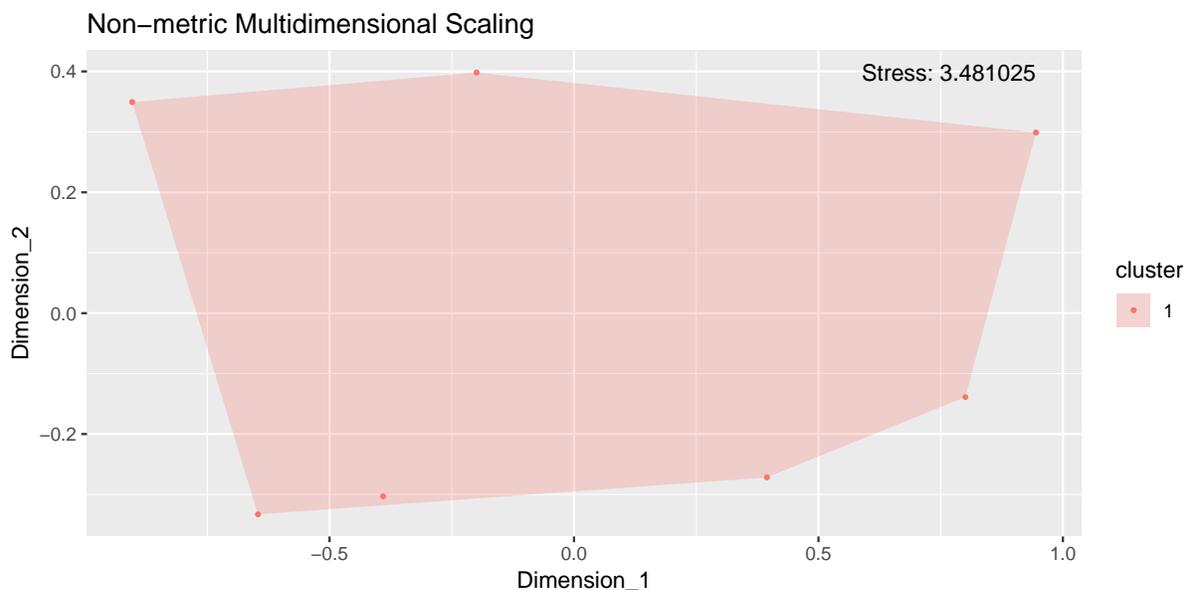
8.5 Multi-dimensional scaling

From the heatmap plot you got an idea now why the cluster analysis assigned certain group memberships to actors. Yet, what you cannot assess so far is how different or similar the clusters are to each other. The network matrix usually simply has too many columns to compare the different actors to each other all at once. One common tool to get around this problem is to reduce the number of dimensions—in this case the dimensions would be agreement and disagreement to each concept—until they can be plotted in a two-dimensional space. That is exactly what (non-metric) multidimensional

scaling (MDS) does.² Taking agreement and disagreement information towards all concepts, MDS can reduce the differences and similarities between actors to plot them in a two-dimensional space.

In `rDNA` we can perform this with the now well-known `dna_cluster` function and the `dna_plotCoordinates` command.

```
clust <- dna_cluster(conn)
dna_plotCoordinates(clust)
```

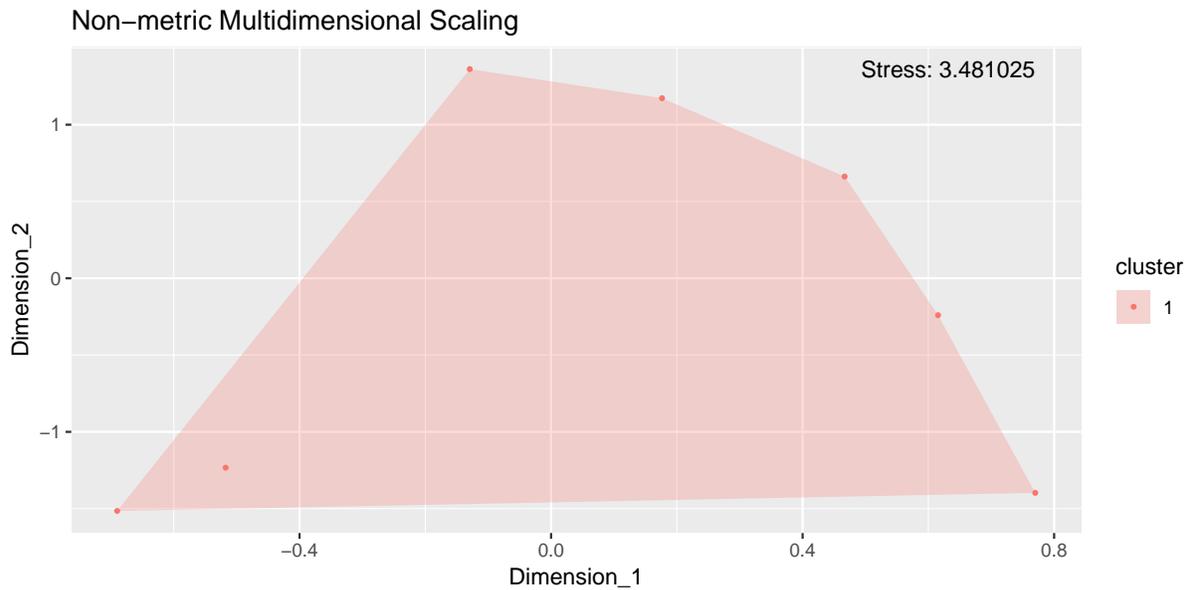


Each point in this dotplot represents an organisation, in this case, and the points are highlighted with colours and shapes which represent their membership in one of the clusters. Clusters are derived in this case with the `pam` function from the `cluster` package (Maechler et al. 2017) and the silhouette width is used to assess the best number of clusters automatically by default, which in this case happens to be one.

However, looking at the plot, you probably wonder why there only appear to be three dots. In fact, there is one point for every one of the seven organisations, but since they are so similar, they are plotted in almost exactly the same place. There are, however, tools to make them visible nevertheless. The first of these tools is called jittering: by adding random noise to the data, it is possible to prevent overplotting. In the `dna_plotCoordinates` function, this can be done by providing one or two numeric values to the `jitter` argument:

```
dna_plotCoordinates(clust, jitter = c(0.5, 1.7))
```

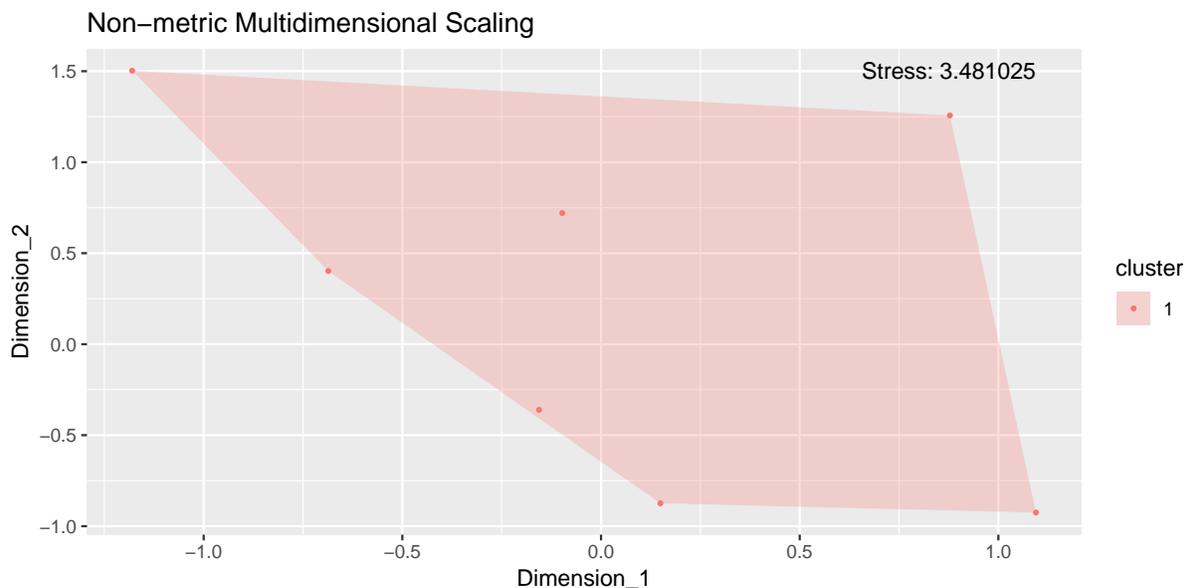
²Specifically we use Kruskal's non-metric multidimensional scaling which makes most sense for our kind of network data.



Now you can see points for all of the seven organisations. Additionally, polygons now encompass all points of the same cluster to visualise the area of a specific cluster. The first jitter value in the code chunk above determines the limits of how much a point can be displaced left or right on the x-axis. If a second value is provided, the points are additionally jittered on the y-axis.

Since the jittering happens randomly, the plot would usually look different every time. However, to ensure reproducibility of plots, we added a seed argument. As long as you do not change the seed, plots will look the same every time you plot them. Yet, if you change the seed, this will alter the position of the jittered points:

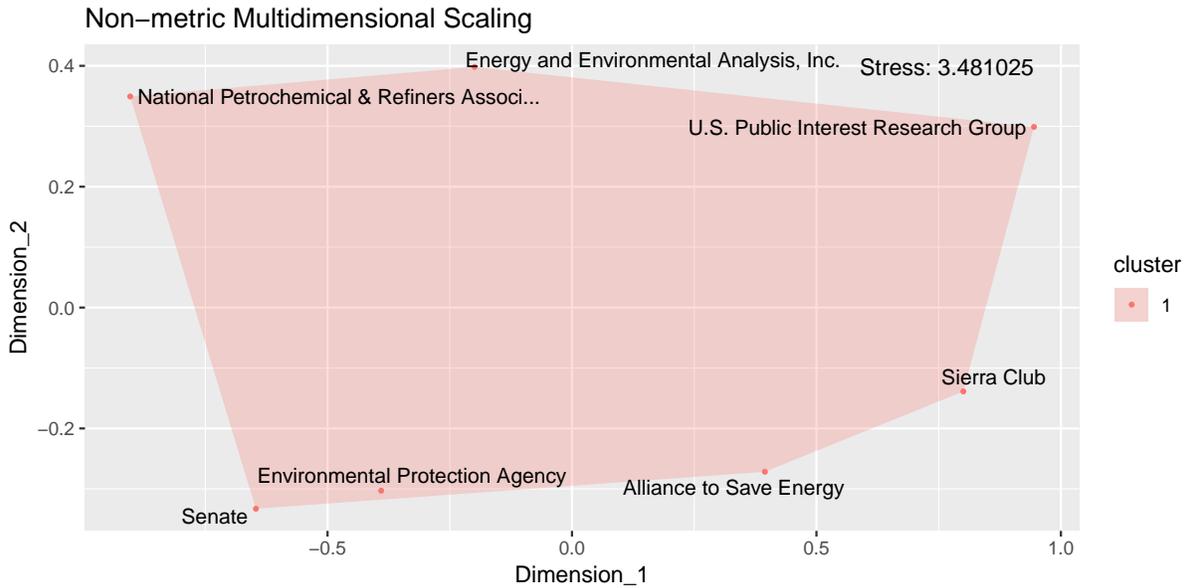
```
dna_plotCoordinates(clust, jitter = c(0.5, 1.7), seed = 23455)
```



Use the jittering option cautiously though, since it distorts the appearance of the plot heavily if you choose appearance high jitter values as in the examples above.

To evade the problem of distortion, you can use the other tool to show overplotted points in `rDNA`. Use option `label = TRUE` to plot the actor labels closely to their respective dots. In this case, the labels will be moved instead of the points if actors appear very closely together.

```
dna_plotCoordinates(clust, label = TRUE)
```

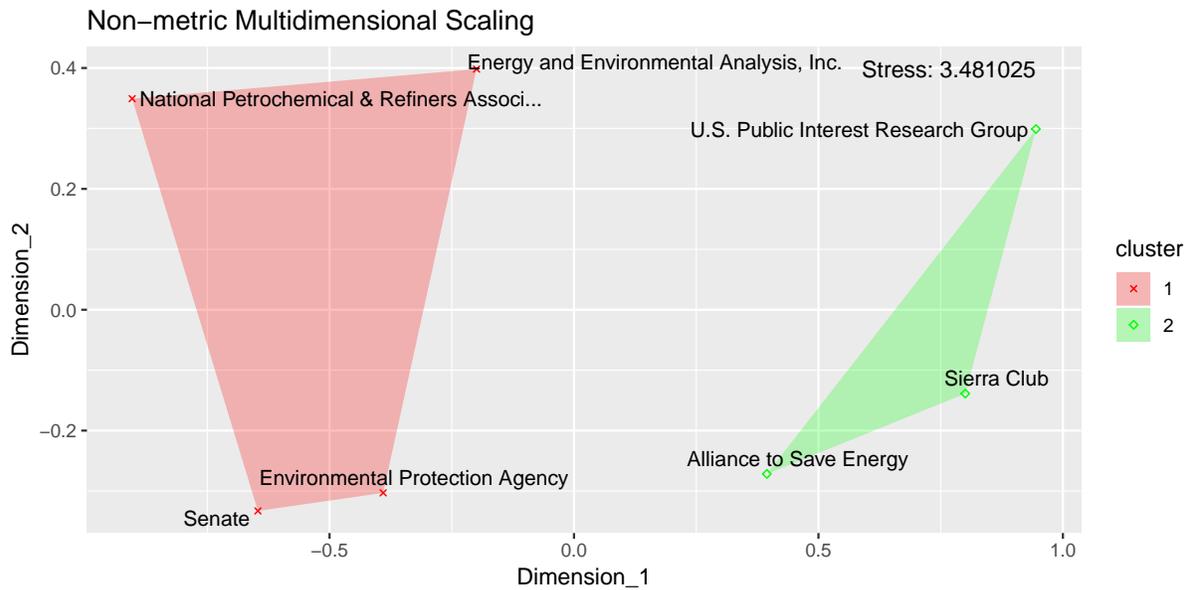


The downside in this case is that the polygons disappeared again, since you need at least three points in one cluster, before they become visible.

Like in the plot functions we showed before, `dna_plotCoordinates` again comes with several arguments to style the plots. Specifically, you can provide colours to `custom_colours` and numeric values to `custom_shape`.

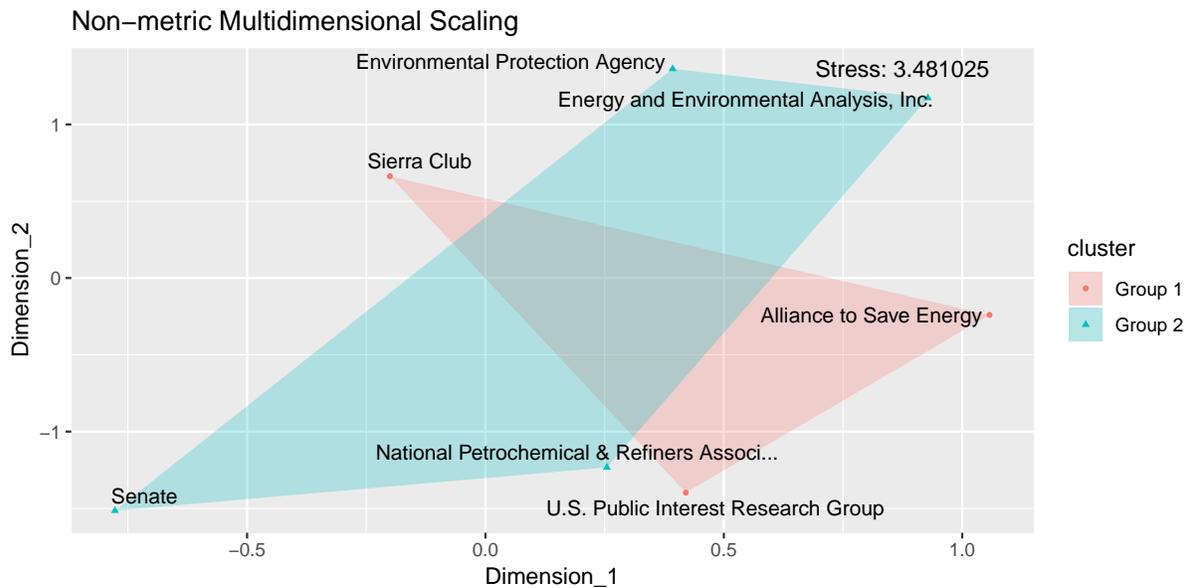
Another important point to highlight about `dna_plotCoordinates` is that there are two additional cluster algorithms to determine the groups. The first one, `pam`, was already mentioned above. The second one, `cluster_louvain` is from the `igraph` package (Csardi and Nepusz 2006) and can be chosen with `clust_method = "louvain"`:

```
dna_plotCoordinates(clust,
  label = TRUE,
  custom_colours = c("red", "green", "blue"),
  custom_shape = c(4,5),
  clust_method = "louvain")
```



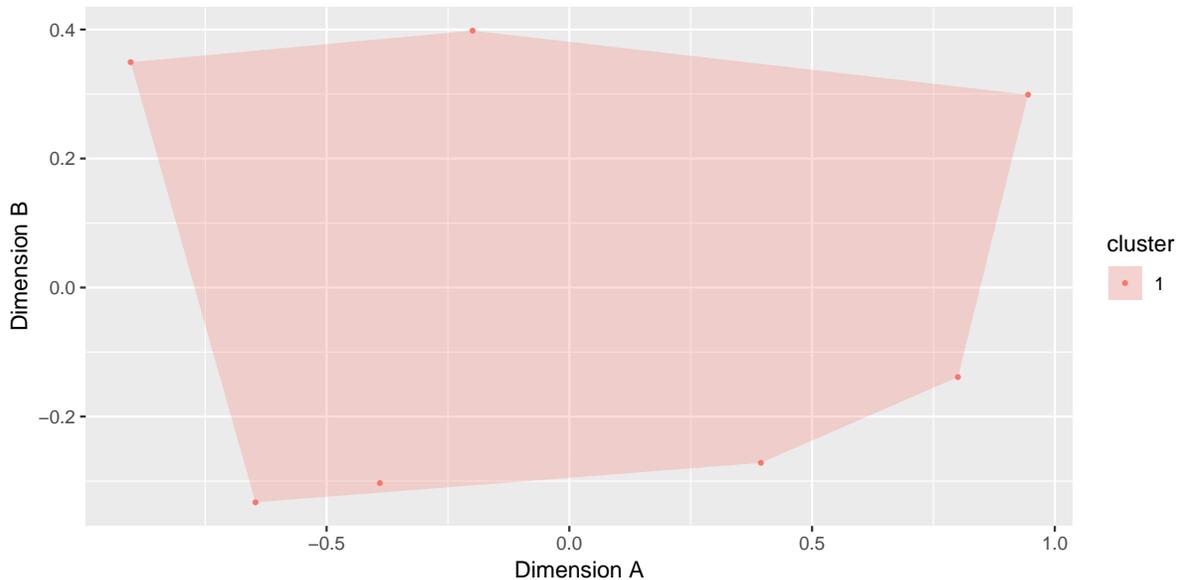
If you would rather use one of those explained above, you can simply set the option in `dna_cluster` and then choose `clust_method = "inherit"`:

```
clust <- dna_cluster(conn, clust.method = "edge_betweenness", cutree.k = 2)
dna_plotCoordinates(clust,
  draw_polygons = TRUE,
  label = TRUE,
  jitter = c(1.5, 1.7),
  clust_method = "inherit",
  seed = 12345)
```



Other ways to style the plot include setting the labels of the axis:

```
dna_plotCoordinates(clust,
  axis_labels = c("Dimension A", "Dimension B"),
  stress = FALSE,
  title = character())
```



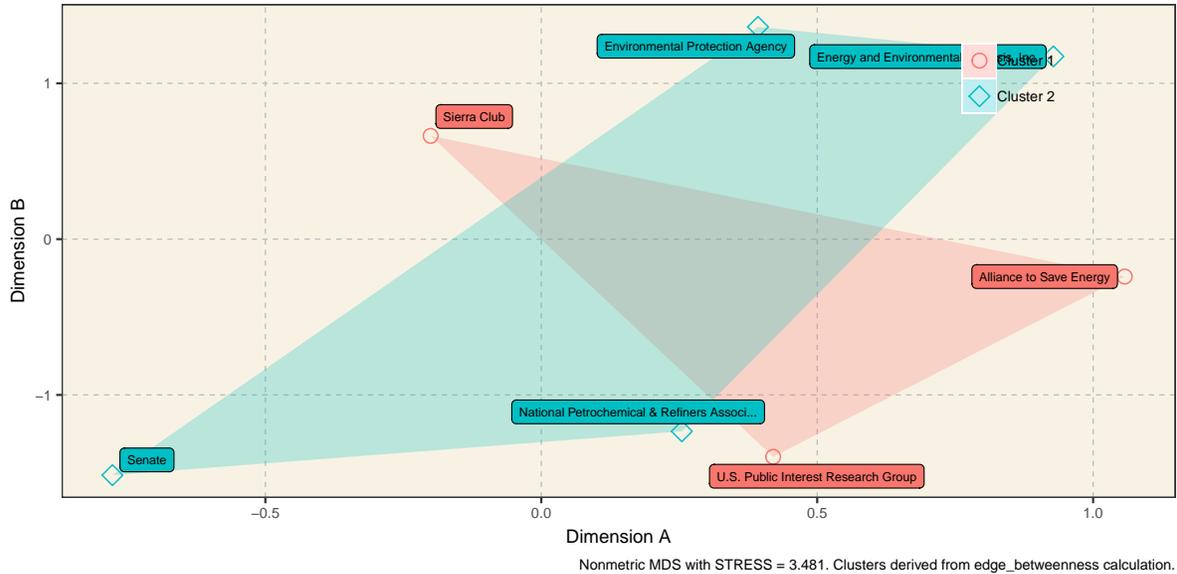
Or using again `ggplot2` commands to modify the plot much further:

```
# ggplot
library("ggplot2")
clust$group <- gsub("Group", "Cluster", clust$group)
dna_plotCoordinates(clust,
  jitter = c(1.5, 1.7),
  clust_method = "inherit",
  axis_labels = c("Dimension A", "Dimension B"),
  custom_shape = c(1, 5),
  stress = FALSE,
  title = character(),
  label = TRUE,
  label_size = 2,
  point_size = 3,
  label_background = TRUE) +
  theme_bw() +
  labs(caption = paste0("Nonmetric MDS with STRESS = ",
    round(attributes(clust$mds)$stress, 3),
    ". Clusters derived from ",
    clust$method,
    " calculation.")) +
  theme(panel.grid.minor = element_blank(),
  text = element_text(size = 9),
  panel.grid.major = element_line(colour = "grey",
    size = 0.3,
    linetype = 2),
  panel.background = element_rect(fill = "#F8F2E4"),
```

```

legend.position = c(0.85, 0.85),
legend.background = element_rect(fill = alpha(0.4)),
legend.title = element_blank()

```



8.6 Network plots

We have shown above already how you can use the R infrastructure to produce some basic network plots with the `statnet` suite of packages. Besides that, there are a number of packages in the R universe that are quite capable to cater to your network plotting needs. The already mentioned `igraph` package (Csardi and Nepusz 2006), for example, comes with very powerful functions for plotting networks. Other packages, such as `networkD3` (Allaire et al. 2017) are capable of creating interactive network plots, which are great to share, for example, on a website. The package `ggraph` (Lin Pedersen 2017a) was specifically developed to employ the same logic as `ggplot2`, the so-called grammar of graphics, to implement a consistent logic for plotting networks. This makes it possible to produce highly customised plots by combining a set of relatively few individual functions which each serve to control one specific aspect of a plot.

Whatever package you might prefer though, all of them have one thing in common: unlike graphical solutions such as `Gephi` or `visone`, using R for network analysis and plotting enables you to do reproducible research. This often outweighs the alleged convenience of GUI applications, as reproducibility is not just of academic value, but also enables you to copy the same plotting code again and again in your own research, once you are happy with the appearance of the network plot.

However, we do acknowledge that bringing the data into the right form and tweaking your network plots to look good can be daunting for beginners or even advanced R users who are new to network analysis. This is why `rDNA` comes with two dedicated functions to create network plots: `dna_plotNetwork` and `dna_plotHive`. Both are capable of handling DNA network objects:³

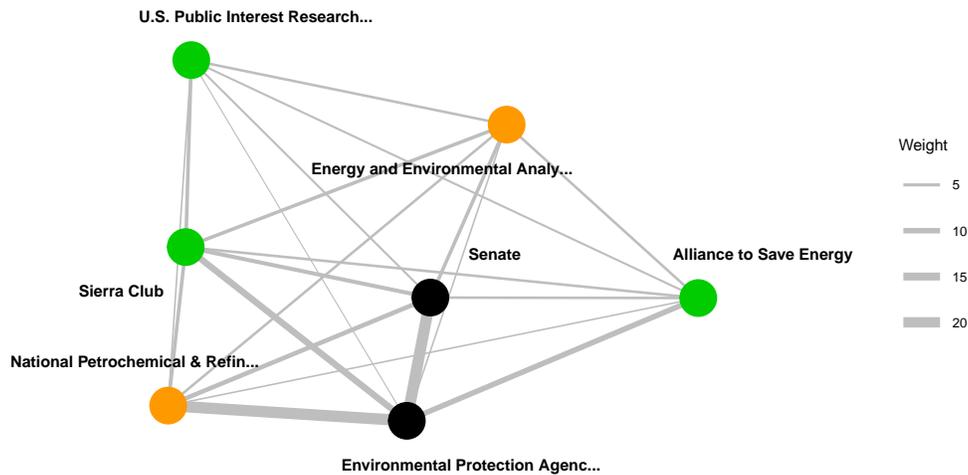
```

# Plot onemode network
nw <- dna_network(conn, networkType = "onemode")

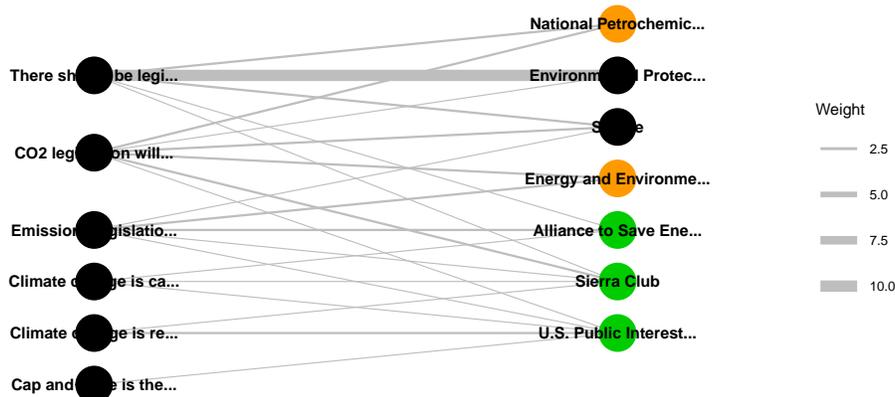
```

³Both functions technically first create an `igraph` object from a DNA network which is then plotted using `ggraph`.

```
dna_plotNetwork(nw)
```



```
# Or twomode  
nw <- dna_network(conn, networkType = "twomode")  
dna_plotNetwork(nw, truncate = 20, label_repel = 0) +  
  coord_flip()
```



As you might be aware though, network visualisation poses one problem: the choice of the right layout algorithm can be quite difficult and there is unfortunately not one that would solve the problems for all datasets. In `dna_plotNetwork`, more than a dozen options from the `igraph` package are available—not all of which are useful for plotting networks from DNA data. Here is the selection of algorithms which we think makes the most sense:

nicely This is probably the most useful option. Not technically a network algorithm, it employs `igraph` package to pick an appropriate layout. See `?igraph::nicely` for the details.

bipartite This algorithm is only really useful for two-mode networks. It minimises edge-crossings by plotting the nodes for each of the two variables in a separate row. See `?igraph::as_bipartite`.

circle Arranges the nodes in a circle.

dh Uses Davidson and Harel's simulated annealing algorithm to place nodes. See `?igraph::with_dh`.

- drl** Uses the force directed algorithm from the DrL toolbox to place nodes. See `?igraph::with_drl`.
- fr** Spreads the nodes based on the force-directed algorithm of Fruchterman and Reingold. See `?igraph::with_fr`
- gem** Places nodes on the plane using the GEM force-directed layout algorithm. See `?igraph::with_gem`.
- graphopt** Employs the Graphopt algorithm based on alternating attraction and repulsion to place nodes.
- kk** Uses the spring-based algorithm by Kamada and Kawai to place nodes. See `?igraph::with_kk`
- lgl** Uses the algorithm from Large Graph Layout to place nodes. See `?igraph::with_lgl`.
- mds** Performs metric multidimensional scaling for generating the coordinates of the nodes. This algorithm is what comes closest to the "stress minimization" layout from `visone`. See `?igraph::with_mds`.
- randomly** Places nodes randomly into the plot. See `?igraph::randomly`.
- star** This places one node in the centre and spreads the rest of the nodes in equal distances around it. See `?igraph::as_star`.

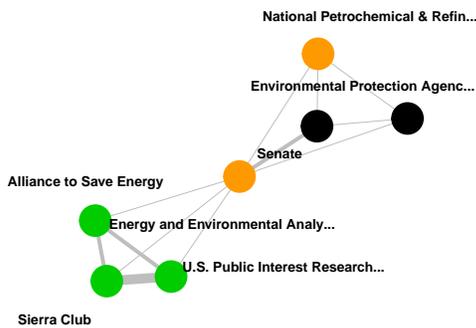
To get a better idea of how this works, look at the code chunk and plot below to see a small selection of the algorithms in action.

```
# First, let's construct a onemode network
nw <- dna_network(conn,
  networkType = "onemode",
  qualifierAggregation = "congruence",
  duplicates = "document",
  excludeValues = list("concept" =
    "There should be legislation to regulate emissions.))

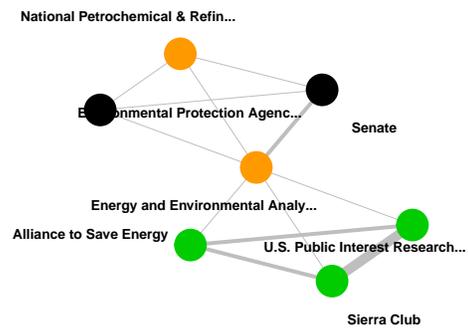
# As with the options for dna_cluster we produce four different plot objects
nw_fr <- dna_plotNetwork(nw, layout = "fr", show_legend = FALSE) +
  ggtitle("Fruchterman Reingold")
nw_graphopt <- dna_plotNetwork(nw, layout = "graphopt", show_legend = FALSE) +
  ggtitle("Graphopt")
nw_mds <- dna_plotNetwork(nw, layout = "mds", show_legend = FALSE) +
  ggtitle("MDS")
nw_randomly <- dna_plotNetwork(nw, layout = "kk") +
  ggtitle("Kamada and Kawai")

# The plots are again arranged in a grid
grid <- grid.arrange(nw_fr, nw_graphopt, nw_mds, nw_randomly)
```

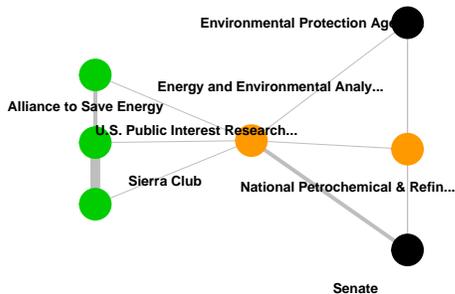
Fruchterman Reingold



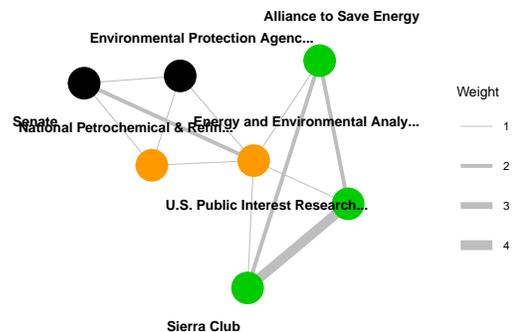
Graphopt



MDS



Kamada and Kawai

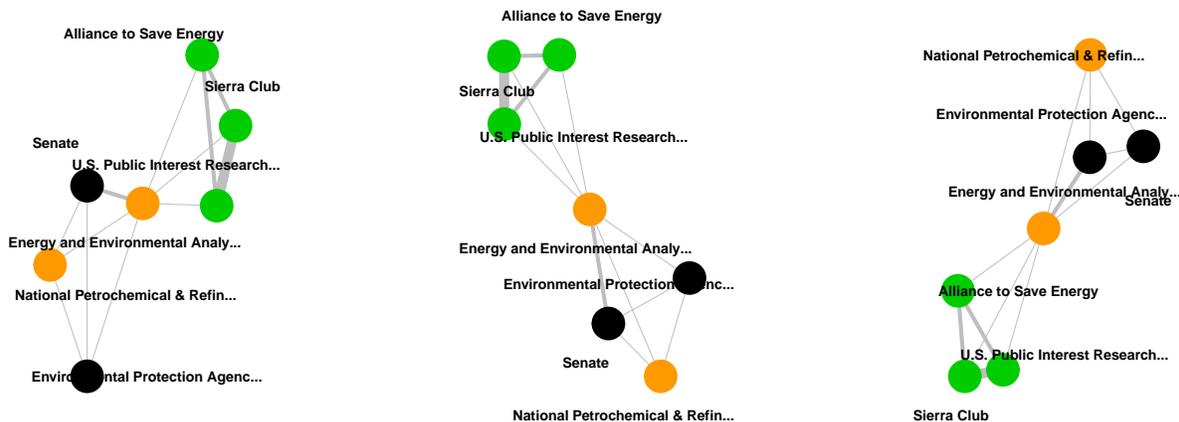


There are again several options to style the plot.

Some critics, however, doubt the usefulness of the common network plots—often referred to as hairballs—you see above. [Krzywinski et al. \(2012\)](#), for example, have argued that these network plots “lack reproducibility and perceptual uniformity because they do not use a node coordinate system”. In the function `dna_plotNetwork` a seed is automatically set to ensure at least reproducibility when you run the same code. You can change the argument `seed` to get an idea of how much chance is involved in calculating the layout.

```
nw_1 <- dna_plotNetwork(nw, show_legend = FALSE, seed = 1)
nw_2 <- dna_plotNetwork(nw, show_legend = FALSE, seed = 2)
nw_12345 <- dna_plotNetwork(nw, show_legend = FALSE, seed = 12345)

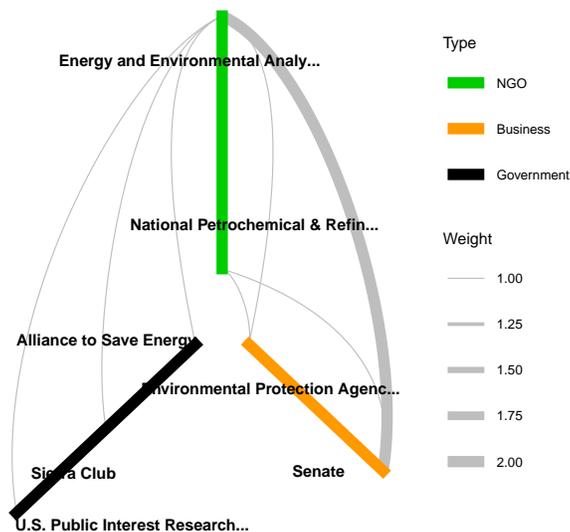
# The plots are again arranged in a grid
grid <- grid.arrange(nw_1, nw_2, nw_12345, nrow = 1)
```



To get around this problem, [Krzywinski et al. \(2012\)](#) suggest so-called hive plots. These plots feature radially distributed linear axes on which the nodes are positioned. You can choose yourself, what the axes mean, so it is easier to assess whether and how a certain attribute of a network influences clustering among actors.

In `rDNA`, you can produce hive plots with the `dna_plotHive` function, which is very similar to `dna_plotNetwork`, except that you additionally provide an argument to produce the axes.

```
dna_plotHive(nw, axis = "type")
```



8.7 Adding and manipulating documents in DNA from R

In `rDNA`, there are also several functions to manipulate the documents within a DNA database from R: `dna_getDocuments`, `dna_addDocument`, `dna_removeDocument` and `dna_setDocuments`.

`dna_getDocuments` retrieves a dataframe with all documents from a DNA connection into the R environment. This can be useful in multiple scenarios: If you have lost or never had original source files for the documents in the database, you can export the dataframe from R into a common format, such as a CSV file, which can be opened using, for example, MS Excel, LibreOffice, or Numbers (see `?write.csv`). This can also be useful if you have used DNA to add metadata to your documents.

A second scenario in which this could be useful is, if you want to calculate some statistics about your text—such as length of texts in words or characters, readability or lexical diversity—or if you want to support your manual coding by determining, for example, the most important words via keyness or tfidf algorithms (see [Welbers et al. \(2017\)](#) for an overview of working with text in R).

Another scenario would be if you want to use R instead of DNA to recode some of your documents' metadata. In the example below, we first retrieve the documents from DNA then alter two of the columns: "type" and "author".

```
docs <- dna_getDocuments(conn)
docs
```

id	title	text	coder	author	source	section	notes	type	date
1	109-876: Blues...	Testimony at...	2	Bluestein, Joel	109	876			2005-01-26 01:00:00
2	109-867: Voino...	OPENING STATE...	2	Voinovich, George	109	867			2005-02-02 01:00:00
3	109-867: Whitm...	Statement of...	2	Whitman, Christine Todd	109	867			2005-02-02 01:00:00
4	109-1: Callaha...	STATEMENT OF...	2	Callahan, Kateri	109	1			2005-02-10 01:00:00
5	109-1: Hamilto...	STATEMENT OF...	2	Hamilton, David	109	1			2005-02-16 01:00:00
6	109-1: Nayak,...	STATEMENT OF...	2	Nayak, Navin	109	1			2005-02-16 01:00:00
7	109-1: Slaught...	STATEMENT OF...	2	Slaughter, Bob	109	1			2005-02-16 01:00:00

```
# Fill empty column with the same word in every row
docs$type <- "hearing"
# Use regular expression to change the order of name and surname of the authors
docs$author <- sub('^(.*)', '(.*)', '\\2 \\1', docs$author)
docs
```

id	title	text	coder	author	source	section	notes	type	date
1	109-876: Blues...	Testimony at.....	2	Joel Bluestein	109	876		hearing	2005-01-26 01:00:00
2	109-867: Voino...	OPENING STATE.....	2	George Voinovich	109	867		hearing	2005-02-02 01:00:00
3	109-867: Whitm...	Statement of.....	2	Christine Todd Whitman	109	867		hearing	2005-02-02 01:00:00
4	109-1: Callaha...	STATEMENT OF.....	2	Kateri Callahan	109	1		hearing	2005-02-10 01:00:00
5	109-1: Hamilto...	STATEMENT OF.....	2	David Hamilton	109	1		hearing	2005-02-16 01:00:00
6	109-1: Nayak,...	STATEMENT OF.....	2	Navin Nayak	109	1		hearing	2005-02-16 01:00:00
7	109-1: Slaught...	STATEMENT OF.....	2	Bob Slaughter	109	1		hearing	2005-02-16 01:00:00

So how do we get this new dataframe back into DNA? The answer is the next function we will discuss: `dna_setDocuments`. This function has just a few arguments but is very capable and can also be dangerous if not used correctly. We, therefore, recommend that you make a backup copy of your database—which you should do regularly anyway—before you start to alter anything. After you have done this, you can provide a dataframe with ten columns:

ID must be integer and must contain the document IDs.

title must contain the document titles as character objects.

text must contain the document texts as character objects.

coder must contain the coder IDs as integer values.

author must contain the document authors as character objects.

source must contain the document sources as character objects.

section must contain the document sections as character objects.

notes must contain the document notes as character objects.

type must contain the document types as character objects.

date must contain the document dates as POSIXct objects or as numeric objects indicating milliseconds since 1970-01-01.

Since we retrieved the objects docs from the database, it already meets all these conditions. In other cases, however, you will have to rearrange the data a bit until you can push it to DNA. Before you actually change anything, it is recommended to do a test run, or in other words `simulate` what would be done to the database:

```
dna_setDocuments(conn, documents = docs, removeStatements = FALSE,
                 simulate = TRUE)
```

Note, that this is the default and that you need to set `simulate = FALSE` before anything really happens. You should also be cautious with the option `removeStatements` since when set to `TRUE`, it will destroy part of your work if a document in your new dataframe has the same ID as a document already present in the database.

8.7.1 Adding newspaper articles using LexisNexisTools

One common way of retrieving texts for analysis in DNA is by downloading articles from the commercial newspaper archive LexisNexis. Many university libraries have access to this database which maintains a collection of newspaper articles from many major outlets across Europe and North America. Its powerful search engine also allows for a finely grained search string to limit the number of articles for a specific topic.

To convert the raw files from LexisNexis though, we need another R package: `LexisNexisTools` (Gruber 2018). You can install this package via `devtools::install_github("JBGruber/LexisNexisTools")`. The workflow looks as follows:

```
library("LexisNexisTools")
# This places a sample TXT file in your working directory in
lnt_sample()

# Look for TXT files
my_files <- list.files(pattern = "TXT", ignore.case = TRUE)

# If this contains other files not from LexisNexis you can subset the vector
my_files <- my_files[grepl("^sample.TXT$", my_files)]

# Now the files can be converted into an LNToutput object
LNToutput <- lnt_read(my_files)

# And this object can be converted to work in rDNA
docs <- lnt_convert(LNToutput, to = "rDNA")
```

This object is almost ready to add to the DNA database. We only need to adjust the ID column, so that IDs are not duplicated and set a valid coder ID (i.e., one that is already present in the database).

```
docs_orig <- dna_getDocuments(conn)
```

```

# We create new unique IDs by IDs with the highest ID from the original data
docs$id <- docs$id + max(docs_orig$id)

# We use coder 1 to import the new documents
unique(docs_orig$coder)

## [1] 2

docs$coder <- 2

# Now we combine the original with the new documents and push it to DNA
docs <- rbind(docs_orig, docs)
dna_setDocuments(conn, documents = docs, simulate = FALSE)

```

Now the documents from LexisNexis are in the DNA database:

```

docs <- dna_getDocuments(conn)
docs

```

id	title	text	coder	author	source	section	notes	type	date
1	109-876: Blues...	Testimony at...	2	Bluestein,...	109	876			2005-01-26
2	109-867: Voino...	OPENING STATE...	2	Voinovich,...	109	867			2005-02-02
3	109-867: Whitm...	Statement of...	2	Whitman, C...	109	867			2005-02-02
4	109-1: Callaha...	STATEMENT OF...	2	Callahan,...	109	1			2005-02-10
5	109-1: Hamilto...	STATEMENT OF...	2	Hamilton,...	109	1			2005-02-16
6	109-1: Nayak,...	STATEMENT OF...	2	Nayak, Nav...	109	1			2005-02-16
7	109-1: Slaught...	STATEMENT OF...	2	Slaughter,...	109	1			2005-02-16
8	Lorem ipsum do...	Lorem ipsum do...	2		Guardi...			newspa...	2010-01-11
9	Lorem ipsum do...	Lorem ipsum do...	2		Guardi...			newspa...	2010-01-11
10	Lorem ipsum do...	Lorem ipsum do...	2	TREVOR Kav...	The Su...	FEATUR...		newspa...	2010-01-11
11	Lorem ipsum do...	Lorem ipsum do...	2	Tom Coghla...	The Ti...	NEWS;...		newspa...	2010-01-11
12	Lorem ipsum do...	Lorem ipsum do...	2	William Re...	The Ti...	EDITOR...		newspa...	2010-01-11
13	Lorem ipsum do...	Lorem ipsum do...	2	Tom Coghla...	The Ti...	NEWS;...		newspa...	2010-01-11
14	ranch noble as...	Afford ranch n...	2		Guardi...			newspa...	2010-01-08
15	PRISONER OF HL...	In publishing...	2		MAIL O...			newspa...	2010-01-10
16	R (programming...	R is a program...	2	Ross Ihaka...	Sunday...	NEWS;...		newspa...	2010-01-10
17	Wikipedia	Wikipedia is a...	2		DAILY...			newspa...	2010-01-09

8.7.2 Adding other documents

Besides LexisNexis, there is a myriad of sources from where you can get documents for your discourse network analysis. These can come in a number of different shapes and (file) formats. The task is then, how to get the documents into DNA with as little manual work as possible. Luckily, many people who use R have had similar problems and developed useful packages to make it as easy as possible to read in (text) data from, for example, MS Word, Excel and PDF files or directly from web pages via web scraping. There are many tutorials online on how to get your specific format in which you retrieved raw text data into R which is why we will not cover this part. However, before you can hand the documents over to DNA, you have to bring them into the correct format first.

We use the dataset “Irish budget speeches from 2010” from [Lowe and Benoit \(2013\)](#) which is integrated in the `quanteda` package ([Benoit 2018](#)) to exemplify the process of how this is done.⁴

⁴Install `quanteda` first with `install.packages("quanteda")` if you do not have it on your system already.

```

library("quanteda")
# First load the data
corpus <- data_corpus_irishbudget2010

# To get to the text data we have to subset the corpus object which transforms
# it into a data.frame
df <- corpus$documents

```

	texts	year	debate	number	foren	name	party
2010_BUDGET_01...	When I present...	2010	BUDGET	01	Brian	Lenihan	FF
2010_BUDGET_02...	This draconian...	2010	BUDGET	02	Richard	Bruton	FG
2010_BUDGET_03...	The other day,...	2010	BUDGET	03	Joan	Burton	LAB
2010_BUDGET_04...	We have heard...	2010	BUDGET	04	Arthur	Morgan	SF
2010_BUDGET_05...	This country h...	2010	BUDGET	05	Brian	Cowen	FF
2010_BUDGET_06...	At the end of...	2010	BUDGET	06	Enda	Kenny	FG
2010_BUDGET_07...	I put on recor...	2010	BUDGET	07	Kieran	ODonnell	FG
2010_BUDGET_08...	I agree with o...	2010	BUDGET	08	Eamon	Gilmore	LAB
2010_BUDGET_09...	The Labour Par...	2010	BUDGET	09	Michael	Higgins	LAB
2010_BUDGET_10...	I have been in...	2010	BUDGET	10	Ruairi	Quinn	LAB
2010_BUDGET_11...	I welcome the...	2010	BUDGET	11	John	Gormley	Green
2010_BUDGET_12...	I have a sense...	2010	BUDGET	12	Eamon	Ryan	Green
2010_BUDGET_13...	Deputy Gilmore...	2010	BUDGET	13	Ciaran	Cuffe	Green
2010_BUDGET_14...	Last night, th...	2010	BUDGET	14	Caoimhghin	O'Caolain	SF

Now the easiest way to arrive at the correct format is to create a new data.frame and fill it with the columns from the budget data. Note, that not all of the columns in DNA makes sense for all data sources. Simply leave a column empty if you think it does not make sense in your case, as is done for “section” in the following:

```

docs_new <- data.frame(id = df$number,
  title = row.names(df),
  text = df$texts,
  coder = 1,
  author = paste(df$foren, df$name),
  source = "Budget Statement 2010",
  section = "",
  notes = df$party,
  type = df$debate,
  date = df$year,
  stringsAsFactors = FALSE)

```

id	title	text	coder	author	source	section	notes	type	date
01	2010_BUDGET_01...	When I present...	1	Brian Lenihan	Budget Sta...		FF	BUDGET	2010
02	2010_BUDGET_02...	This draconian...	1	Richard Bruton	Budget Sta...		FG	BUDGET	2010
03	2010_BUDGET_03...	The other day,...	1	Joan Burton	Budget Sta...		LAB	BUDGET	2010
04	2010_BUDGET_04...	We have heard...	1	Arthur Morgan	Budget Sta...		SF	BUDGET	2010
05	2010_BUDGET_05...	This country h...	1	Brian Cowen	Budget Sta...		FF	BUDGET	2010
06	2010_BUDGET_06...	At the end of...	1	Enda Kenny	Budget Sta...		FG	BUDGET	2010
07	2010_BUDGET_07...	I put on recor...	1	Kieran O'Donnell	Budget Sta...		FG	BUDGET	2010
08	2010_BUDGET_08...	I agree with o...	1	Eamon Gilmore	Budget Sta...		LAB	BUDGET	2010
09	2010_BUDGET_09...	The Labour Par...	1	Michael Higgins	Budget Sta...		LAB	BUDGET	2010
10	2010_BUDGET_10...	I have been in...	1	Ruairi Quinn	Budget Sta...		LAB	BUDGET	2010
11	2010_BUDGET_11...	I welcome the...	1	John Gormley	Budget Sta...		Green	BUDGET	2010
12	2010_BUDGET_12...	I have a sense...	1	Eamon Ryan	Budget Sta...		Green	BUDGET	2010
13	2010_BUDGET_13...	Deputy Gilmore...	1	Ciaran Cuffe	Budget Sta...		Green	BUDGET	2010
14	2010_BUDGET_14...	Last night, th...	1	Caoimhghin O'Caolain	Budget Sta...		SF	BUDGET	2010

Before you can proceed, you should check if the columns already have the right classes to hand the data.frame over to DNA:

```
lapply(docs_new, class)

## $id
## [1] "character"
##
## $title
## [1] "character"
##
## $text
## [1] "character"
##
## $coder
## [1] "numeric"
##
## $author
## [1] "character"
##
## $source
## [1] "character"
##
## $section
## [1] "character"
##
## $notes
## [1] "character"
##
## $type
## [1] "character"
##
## $date
## [1] "character"
```

Compare this output with the requirements for the data.frame (see `?dna_setDocuments`) and you can see that only the ID column does not have the correct class. In this case, it is easy to correct that:

```
docs$id <- as.integer(docs$id)
```

The remaining steps are the same as before:

1. Retrieve the documents already in DNA with `dna_getDocuments` (can be skipped if no documents are present).
2. Make sure the new documents have unique IDs.
3. Use `rbind` to append the old data.frame with the new documents.
4. Use `dna_setDocuments` to push the data.frame into the DNA database.

```
# 1.
docs_orig <- dna_getDocuments(conn)

# 2.
docs$id <- docs$id + max(docs_orig$id)

# 3.
docs <- rbind(docs_orig, docs)

# 4.
dna_setDocuments(conn, documents = docs, simulate = FALSE)
```

This should be all it takes to get documents from pretty much any source into DNA.

Bibliography

- Allaire, J. J., Gandrud, C., Russell, K., and Yetman, C. J. (2017). *networkD3: D3 JavaScript Network Graphs from R*. R package version 0.4.
- Benoit, K. (2018). *quanteda: Quantitative Analysis of Textual Data*. R package version 1.2.0.
- Breindl, Y. (2013). Discourse networks on state-mandated access blocking in Germany and France. *info*, 15(6):42–62.
- Broadbent, J. and Vaughter, P. (2014). Inter-disciplinary analysis of climate change and society: A network approach. In Manfredo, M. J., Vaske, J. J., Rechkemmer, A., and Duke, E. A., editors, *Understanding Society and Natural Resources*, pages 203–228. Springer Netherlands, Dordrecht.
- Brutschin, E. (2013). *Dynamics in EU Policy-Making: The Liberalization of the European Gas Market*. PhD thesis.
- Butts, C. T. (2008a). Social network analysis with `sna`. *Journal of Statistical Software*, 24(6):1–51.
- Butts, C. T. (2008b). `network`: A package for managing relational data in R. *Journal of Statistical Software*, 24(2):1–36.
- Butts, C. T. (2015). *network: Classes for Relational Data*. The Statnet Project (<http://statnet.org>). R package version 1.13.0.
- Butts, C. T. (2016). *sna: Tools for Social Network Analysis*. R package version 2.4.
- Csardi, G. and Nepusz, T. (2006). The `igraph` software package for complex network research. *Inter-Journal*, Complex Systems:1695.
- Fisher, D. R., Leifeld, P., and Iwaki, Y. (2013a). Mapping the ideological networks of American climate politics. *Climatic Change*, 116(3):523–545.
- Fisher, D. R., Waggle, J., and Leifeld, P. (2013b). Where does political polarization come from? Locating polarization within the U.S. climate change debate. *American Behavioral Scientist*, 57(1):70–92.
- Gkiouzepas, G. and Botetzagias, I. (2015). Climate change coverage in Greek newspapers: 2001–2008. *Environmental Communication*, 11(4):490–514.
- Goodreau, S. M., Handcock, M. S., Hunter, D. R., Butts, C. T., and Morris, M. (2008). A `statnet` tutorial. *Journal of Statistical Software*, 24(9):1–26.
- Gruber, J. (2018). *LexisNexisTools. An R Package for Working with Files from LexisNexis*. R package version 0.1.0.
- Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., Krivitsky, P. N., Bender-deMoll, S., and Morris, M. (2016). *statnet: Software Tools for the Statistical Analysis of Network Data*. The Statnet Project (<http://www.statnet.org>). R package version 2016.9.

- Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., and Morris, M. (2008). **statnet**: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, 24(1):1–11.
- Haunss, S. (2017). (De-)legitimizing discourse networks: Smoke without fire? In Schneider, S., Schmidtke, H., Haunss, S., and Gronau, J., editors, *Capitalism and Its Legitimacy in Times of Crisis*, pages 191–220. Springer International Publishing, Cham.
- Haunss, S., Dietz, M., and Nullmeier, F. (2017). Der Ausstieg aus der Atomenergie: Diskursnetzwerk-analyse als Beitrag zur Erklärung einer radikalen Politikwende. *Zeitschrift für Diskursforschung*, (3):288–315.
- Haunss, S. and Kohlmorgen, L. (2009). Lobbying or politics? political claims making in IP conflicts. In Haunss, S. and Shadlen, K. C., editors, *The Politics of Intellectual Property*, pages 107–128. Edward Elgar Publishing.
- Hurka, S. and Nebel, K. (2013). Framing and policy change after shooting rampages: A comparative analysis of discourse networks. *Journal of European Public Policy*, 20(3):390–406.
- Imbert, I. (2017). *An Inquiry into the Material and Ideational Dimensions of Policymaking: A Case Study of Fuel Poverty in Germany*. PhD thesis.
- Krzywinski, M., Birol, I., Jones, S. J. M., and Marra, M. A. (2012). Hive plots – rational approach to visualizing networks. *Briefings in Bioinformatics*, 13(5):627–644.
- Leifeld, P. (2013). Reconceptualizing major policy change in the advocacy coalition framework: A discourse network analysis of German pension politics. *Policy Studies Journal*, 41(1):169–198.
- Leifeld, P. (2016). *Policy Debates as Dynamic Networks: German Pension Politics and Privatization Discourse*, volume 29 of *Schriften des Zentrums für Sozialpolitik Bremen*. Campus Verlag/University of Chicago Press, Frankfurt/New York.
- Leifeld, P. (2017). Discourse network analysis: Policy debates as dynamic networks. In Victor, J. N., Montgomery, A. H., and Lubell, M., editors, *The Oxford Handbook of Political Networks*, volume 1. Oxford University Press.
- Leifeld, P., Cranmer, S. J., and Desmarais, B. A. (2017). **xergm**: *Extensions of Exponential Random Graph Models*. R package version 1.8.2.
- Leifeld, P., Cranmer, S. J., and Desmarais, B. A. (2018). *Temporal Exponential Random Graph Models with btergm: Estimation and Bootstrap Confidence Intervals*.
- Leifeld, P. and Gruber, J. (2018). **rdNA**: *A Package to Control Discourse Network Analyzer from R*. University of Glasgow, School of Social and Political Sciences, Glasgow. R package version 2.1.0.
- Leifeld, P. and Haunss, S. (2012). Political discourse networks and the conflict over software patents in Europe. *European Journal of Political Research*, 51(3):382–409.
- Lin Pedersen, T. (2017a). **ggraph**: *An Implementation of Grammar of Graphics for Graphs and Networks*. R package version 1.0.1.
- Lin Pedersen, T. (2017b). **tidygraph**: *A Tidy API for Graph Manipulation*. R package version 1.0.0.
- Lowe, W. and Benoit, K. (2013). Validating estimates of latent traits from textual data using human judgment as a benchmark. *Political Analysis*, 21(3):298–313.
- Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2017). **cluster**: *Cluster Analysis Basics and Extensions*. R package version 2.0.6.

- Manfredo, M. J., Vaske, J. J., Reckemmer, A., and Duke, E. A., editors (2014). *Understanding Society and Natural Resources*. Springer Netherlands, Dordrecht.
- Muller, A. (2014a). Het meten van beleidscontroverse en polarisatie met discoursnetwerkanalyse: De case van het abortusdebat in de Belgische Kamer. *Sociologos. Tijdschrift voor Sociologie*, 35(3):159–184.
- Muller, A. (2014b). Het meten van discourscoalities met discoursnetwerkanalyse: Naar een formele analyse van het politieke vertoog. *Res Publica*, 56(3):337–364.
- Muller, A. (2015). Using discourse network analysis to measure discourse coalitions: Towards a formal analysis of political discourse. *World Political Science*, 11(2):17.
- Nagel, M. (2016). *Polarisierung im politischen Diskurs: Eine Netzwerkanalyse zum Konflikt um Stuttgart 21*. Springer VS, Wiesbaden.
- Nägler, R. (2015). With|out a partner. the idea of cooperation in higher education discourses. *SSRN Electronic Journal*.
- Neuwirth, E. (2014). *RColorBrewer: ColorBrewer Palettes*. R package version 1.1-2.
- Rantala, S. and Di Gregorio, M. (2014). Multistakeholder environmental governance in action: REDD+ discourse coalitions in Tanzania. *Ecology and Society*, 19(2).
- Rinscheid, A. (2015). Crisis, policy discourse, and major policy change: Exploring the role of subsystem polarization in nuclear energy policymaking. *European Policy Analysis*, 1(2).
- Schneider, V. and Ollmann, J. K. (2014). Punctuations and displacements in policy discourse: The climate change issue in Germany 2007–2010. In Silvern, S. and Young, S., editors, *Environmental Change and Sustainability*. InTech, Rijeka, Croatia.
- Stoddart, M. C. J., Ramos, H., and Tindall, D. B. (2015). Environmentalists’ mediawork for jumbo pass and the tobatic wilderness, Canada: Combining text-centred and activist-centred approaches to news media and social movements. *Social Movement Studies*, 14(1):75–91.
- Stoddart, M. C. J. and Tindall, D. B. (2015). Canadian news media and the cultural dynamics of multilevel climate governance. *Environmental Politics*, 24(3):401–422.
- Team, R. C. (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Urbanek, S. (2017). *rJava: Low-Level R to Java Interface*. R package version 0.9-9.
- Wagner, P. and Payne, D. (2017). Trends, frames and discourse networks: Analysing the coverage of climate change in Irish newspapers. *Irish Journal of Sociology*, 25(1):5–28.
- Welbers, K., van Atteveldt, W., and Benoit, K. (2017). Text analysis in R. *Communication Methods and Measures*, 11(4):245–265.
- Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.
- Wickham, H., Hester, J., and Chang, W. (2018). *devtools: Tools to Make Developing R Packages Easier*. R package version 1.13.5.
- Wu, J. and Zhou, L. (2015). DOBNet: Exploiting the discourse of deception behaviour to uncover online deception strategies. *Behaviour & Information Technology*, 34(9):936–948.
- Yun, S.-J., Ku, D., Park, N.-B., and Han, J. (2014). Framing climate change as an economic opportunity in South Korean newspapers. *Development and Society*, 43(2):219–238.