Discourse Network Analyzer Manual

Philip Leifeld, Johannes Gruber and Felix Rolf Bossner

Last update: DNA 2.0 beta 20 with rDNA 2.0.4 on February 5, 2018.

Contents

Chapter 1

Introduction

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 2018), which are designed for researchers using the method *discourse network analysis*.[1] By combining content and dynamic network analysis, this method can reveal the structure and dynamics of debates, such as political discourses. For the user, the method comprises three basic steps: first, to annotate statements of actors in unstructured (text) sources; second, to create networks from the thereby structured data; and finally to analyse and interpret the results by employing the toolbox of network analysis. The results can take a number of different forms, such as congruence or conflict networks at the actor or concept level, affiliation networks of actors and concept stances, and longitudinal versions of these networks (see Section 2 or Leifeld (2016a) for a comprehensive overview of the method).

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, 2016b), climate politics (Fisher et al. 2013a,b; Broadbent and Vaughter 2014; Gkiouzepas and Botetzagias 2015; Manfredo et al. 2014; Schneider and K. 2014; Stoddart and Tindall 2015; Wagner and Payne 2017; Yun et al. 2014), software patents and property rights (Leifeld and Haunss 2012; Herweg 2013), Internet policy (Breindl 2013), infrastructure projects (Nagel 2016), energy policy (Brutschin 2013; Haunss et al. 2017; Imbert 2017; Rinscheid 2015; Rinscheid et al. 2015), shooting rampages (Hurka and Nebel 2013), abortion (Muller 2014a,b, 2015), outdoor sports (Stoddart et al. 2015), water politics (Brandenberger et al. 2015; Cisneros 2015), deforestation (Rantala and Di Gregorio 2014), genetically modified organisms (Tosun et al. 2015), higher education (Nägler 2015), international financial politics (Werner 2015), and online deception (Wu and Zhou 2015). Many of these studies were already employing `DNA` and as the academic interest in discourse network analysis has steadily grown in recent years, the use of both, the method and the software, will eventually become more widespread.

The benefit of using the `Java` software `DNA` is that it is specifically designed to aid the user in the first two of basic steps of discourse network analysis. It is mainly designed for qualitative content analysis which in this case means, to annotate the statements, or in other words, to

[1] *This manual is a work in progress and will be continuously updated during the year 2018. See* `github.com/leifeld/dna/blob/master/manual/` *for the most recent version.*

structure the text data. The program is also able, to create different kinds of network matrices and export them to other programs for further analysis and plotting. DNA is furthermore designed to help the user to develop a comprehensive workflow, from importing raw text, over managing multiple users as well as different types of documents, supporting the coding process with a regular expression highlighter to basic analysis options during export. Furthermore, while the software is primarily designed for discourse network analysis, it is also very flexible as the statement types can be adjusted (*see Section 4.3*) to accommodate different content analysis tasks, such as framing analysis. 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 lack the functionality necessary for exporting network data.

The companion R package to DNA—rDNA—additionally helps with the remaining basic step mentioned above: analysis of the coded statements. rDNA integrates the results from coding performed in DNA with the statistical computing environment R to perform more in-depth analysis of the coded material. While data can be exported to other software such as UCINET, visone, NETMINER, GEPHI and others, R is the preferred choice, as it enables reproducible research, is free and open source and has a huge community of users and developers who are engaged in all kinds of data analysis tasks. Consequently, R has several fantastic packages developed specifically for network analysis—such as **statnet** (Handcock et al. 2008a), **igraph** (Csardi and Nepusz 2006a), **sna** (Butts 2016), **network** (Butts 2008a), **tidygraph** (Pedersen 2017b) and **ggraph** (Pedersen 2017a)—and a lot more that can aid content and network analysis even though they were written with other tasks in mind. 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.

The outline of this manual is as follows. Section 2 is a concise and fairly technical description of the types of networks DNA can export. Section 3 explains how to install DNA and rDNA which both rely on a correctly set up Java runtime environment. While installing the programs themselves would probably not be worth its own section, installing Java on some operating systems can be surprisingly tough—at least without our explanations. Then four sections follow, which describe the usage of DNA in detail: Section 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. Section 5 explains how you can import and organise your raw data (i. e. documents). Section 6 and Section 7 will explain—once they are completed—how material is coded in DNA and how coded data can be exported to other programs for further analysis. Section 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 Section 3*). Questions and bug reports can be posted in the issue tracker on GitHub.

Chapter 2

DNA algorithms

Philip Leifeld

This section summarizes the main algorithms implemented in DNA in a technical way.

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 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 organization i mentions concept j with intensity k five times.

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 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. The following transformations are possible:

2.1 Congruence

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:

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

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

In the binary case, i. e., $|K| = 2$, this reduces to

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

2.2 Conflict

Binary case:

$$y_{ii'}^{\text{conflict binary}} = \Phi_{ii'} \left(\sum_{j=1}^{n} \sum_{k} (1 - x_{ijk}) x_{i'jk} + x_{ijk}(1 - x_{i'jk}) \right). \tag{2.3}$$

More generally, in the integer case:

$$y_{ii'}^{\text{conflict}} = \Phi_{ii'} \left(\sum_{j=1}^{n} \sum_{k} \sum_{k'} x_{ijk} x_{i'jk'} \left(\frac{|k - k'|}{|K| - 1} \right) \right) \tag{2.4}$$

2.3 Subtract

$$y_{ii'}^{\text{subtract}} = y_{ii'}^{\text{congruence}} - y_{ii'}^{\text{conflict}} \tag{2.5}$$

2.4 Ignore

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

2.5 Normalization

In the simplest case, normalization can be switched off, in which case $\Phi_{ii'}^{\text{no}}(\omega) = \omega$.

Alternatively, edge weights can be divided by the average activity of nodes i and i':

$$\Phi_{ii'}^{\text{avg}}(\omega) = \frac{\omega}{\frac{1}{2} \left(\sum_{j=1}^{n} \sum_{k} x_{ijk} + \sum_{j=1}^{n} \sum_{k} x_{i'jk} \right)}. \tag{2.7}$$

With Jaccard normalization, we don't 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:

$$\Phi_{ii'}^{\text{Jaccard}}(\omega) = \frac{\omega}{\sum_{j=1}^{n} \sum_{k} x_{ijk}[x_{i'jk} = 0] + \sum_{j=1}^{n} \sum_{k} x_{i'jk}[x_{ijk} = 0] + \sum_{j=1}^{n} \sum_{k} x_{ijk} x_{i'jk}}. \tag{2.8}$$

With cosine normalization, we take the product in the denominator:

$$\Phi_{ii'}^{\text{cosine}}(\omega) = \frac{\omega}{\sqrt{(\sum_{j=1}^{n} \sum_k x_{ijk})^2} \sqrt{(\sum_{j=1}^{n} \sum_k x_{i'jk})^2}}. \tag{2.9}$$

2.6 Affiliation networks

Ignoring the qualifier variable:

$$y_{ij}^{\text{affiliation ignore}} = \Phi_{ij} \left(\sum_k x_{ijk} \right) \tag{2.10}$$

Subtracting negative from positive ties (integer case):

$$y_{ij}^{\text{affiliation subtract binary}} = \Phi_{ij} \left(\sum_k k \cdot x_{ijk} \right) \tag{2.11}$$

Subtracting negative from positive ties (binary case):

$$y_{ij}^{\text{affiliation subtract binary}} = \Phi_{ij} \left(\sum_k (k \cdot x_{ijk} - (1-k) \cdot x_{ijk}) \right) \tag{2.12}$$

Note that the binary case is *not* merely a special case of the weighted affiliation network in this case.

2.7 Normalization for affiliation networks

With *activity* normalization, ties from active nodes receive lower weights:

$$\Phi_{ij}^{\text{activity}}(\omega) = \frac{\omega}{\sum_{j=1}^{n} \sum_k x_{ijk}} \tag{2.13}$$

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

$$\Phi_{ij}^{\text{prominence}}(\omega) = \frac{\omega}{\sum_{i=1}^{m} \sum_k x_{ijk}} \tag{2.14}$$

Chapter 3

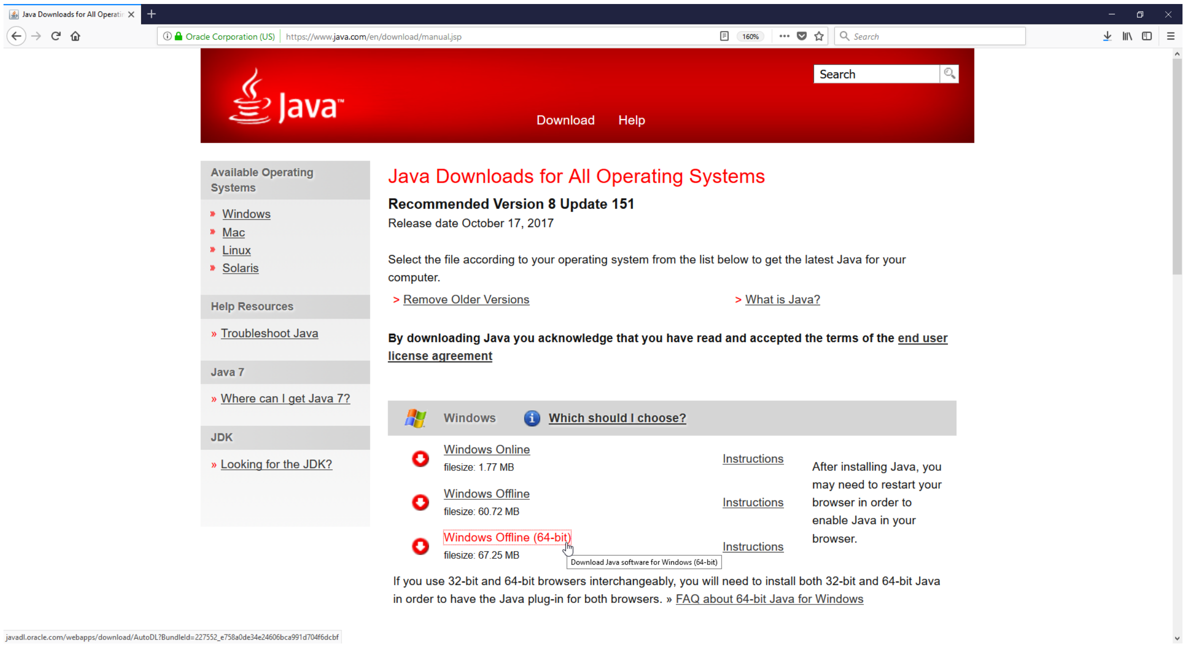
Installation of DNA and rDNA

Johannes Gruber

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. This section turned out to be longer than we initially expected as we noticed that installing Java on different operating systems can be a bit challenging—luckily for you though we figured it out anyway and provide you with all the steps. Installing and configuring a valid Java Runtime Environment on your machine will thus be the first and only complicated step of the installation. However, 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 (JRE) (Version 8) on your computer.
3. (On Windows and Mac: set up the "JAVA_HOME" to the installation path of your JRE.)
4. Download the newest executable JAR from github.com/leifeld/dna/releases.
5. (On Linux: make the JAR file executable.)
   (On Mac: allow excetuting apps from an unidentified developer.)
6. You can now run the standalone DNA or continue to install rDNA as well.


Java Downloads for All Operatir
Oracle Corporation (US)
https://www.java.com/en/download/manual.jsp
160%
Search
Java™
Search
Download
Help
Available Operating Systems
Windows
Mac
Linux
Solaris
Help Resources
Troubleshoot Java
Java 7
Where can I get Java 7?
JDK
Looking for the JDK?
Windows
Which should I choose?
Download Java software for Windows (64-bit)
javadl.oracle.com/webapps/download/AutoDL?BundleId=227552_e758a0de34e24606bca991d704f6dcbf

Java Downloads for All Operating Systems

Recommended Version 8 Update 151

Release date October 17, 2017

Select the file according to your operating system from the list below to get the latest Java for your computer.

> Remove Older Versions   > What is Java?

By downloading Java you acknowledge that you have read and accepted the terms of the end user license agreement

Windows Online
filesize: 1.77 MB   Instructions

Windows Offline
filesize: 60.72 MB   Instructions

Windows Offline (64-bit)
filesize: 67.25 MB   Instructions

After installing Java, you may need to restart your browser in order to enable Java in your browser.

If you use 32-bit and 64-bit browsers interchangeably, you will need to install both 32-bit and 64-bit Java in order to have the Java plug-in for both browsers. » FAQ about 64-bit Java for Windows

Figure 3.1: Downloading JRE from Oracle

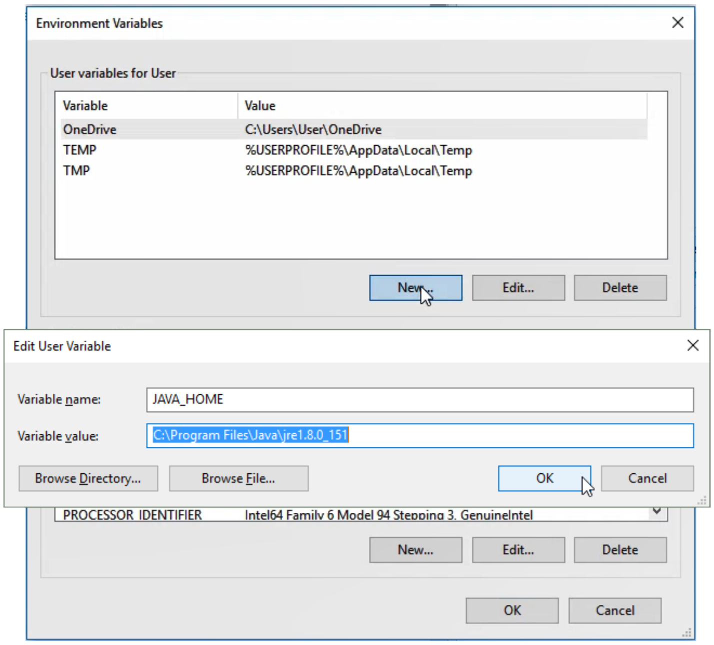
7. Download and install R (and RStudio).
8. In R: install the necessary R packages rJava and devtools.
9. In R: install rDNA via

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

3.1 Windows

To install the necessary Java Runtime Environment on your Windows computer, simply go to 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, people 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 which include this title and should usually display the menu we are looking



Environment Variables

User variables for User

Variable	Value
OneDrive	C:\Users\User\OneDrive
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Edit User Variable

Variable name: JAVA_HOME

Variable value: C:\Program Files\Java\jre1.8.0_151

Variable	Value
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 94 Stepping 3, GenuineIntel

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

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 haven't altered the default install 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]

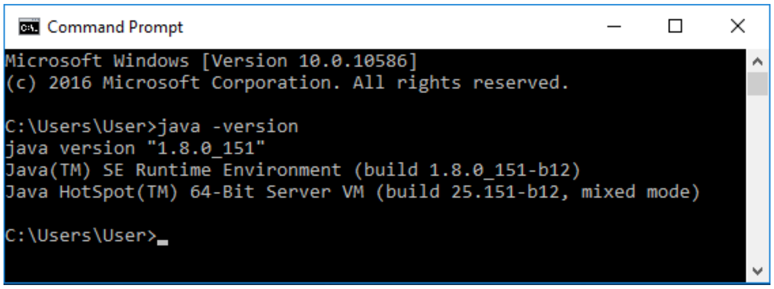
Windows should now recognise `Java` and be able to run `Java` commands. To test this, we can open the Command Prompt (press the ⊞ button on your keyboard and simply enter "cmd" and then hit "Enter") and type a `Java` command, e. g. `''java -version''`. If the installation was successful, the output should 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 a recommended integrated development environment (IDE) called `RStudio`, which makes working with `R` a lot easier and also looks a lot better than the default interface.

[1] On 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".

[2] This 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.

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



```
Microsoft Windows [Version 10.0.10586]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\User>java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)

C:\Users\User>_
```

Figure 3.3: Testing Java installation in Windows Command Prompt

Install `R` on Windows

1. First, you need to download R from cran.r-project.org/bin/windows/base/.
2. On the top of the page click on **Download R 3.4.3 for Windows** (or a newer version if available).
3. Install the downloaded file, e. g. "R-3.4.3-win.exe". Usually, it is fine to leave all default settings in the installation options.
4. Go to rstudio.com/products/rstudio/download/.
5. At the bottom of the page, under "Installers for Supported Platforms", click on the link **RStudio 1.1.383 - 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 unchanged.
6. After installation, you can use `R` by opening `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" (*the window which covers the left half 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, to explain what this means for users who are not familiar with documents which contain 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 #, 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 every line which does not start with ## contains R code which you can copy and paste to the Console in RStudio and run. Alternnativly, you can also copy the code to an R script and execute it by either clicking on this button Run on the upper right of RStudio, near the corner or you can use the shortcut "Ctrl+Enter". Both ways, 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 the next section which describes Installing the programs themselves.

3.2 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, but 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 Runtime Environment version 8.[4]

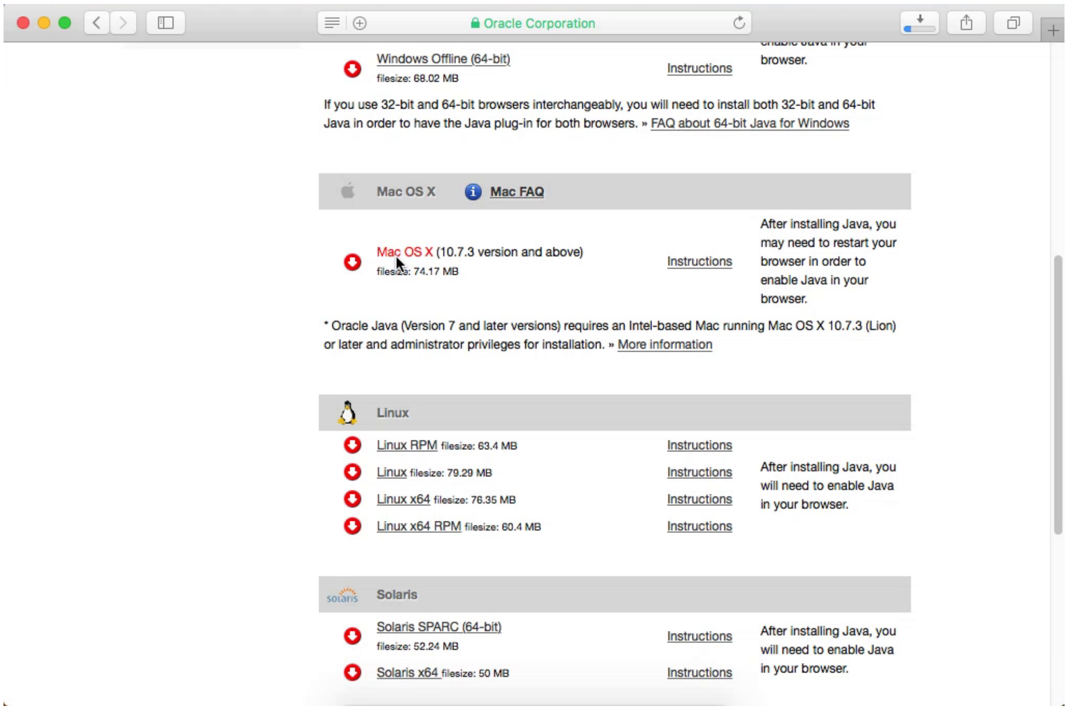
First, please download the file support.apple.com/downloads/DL1572/en_US/javaforosx.dmg and install it, accepting all defaults. After this has finished, we can proceed to get the new version of the Java Runtime Environment. Go to java.com/en/download/manual.jsp and scroll down to download **"Mac OS X (10.7.3 version and above)"** (see Figure 3.4). Again, install the program accepting all defaults.

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 a recommended integrated development environment (IDE) called RStudio, which makes working with R a lot easier and also looks a lot better than R's default interface.

Install R on Mac

1. First, you need to download R from cran.r-project.org/bin/macosx/.
2. On the top of the page click on **R-3.4.3.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 rstudio.com/products/rstudio/download/.
5. At the bottom of the page, under "Installers for Supported Platforms", click on the link **RStudio 1.1.383 - Mac OS X 10.6+ (64-bit)** (or a newer version if available). Again the default installation options are fine in most cases and can be accepted unchanged.

[4] If you do not wish to ever use rDNA or any other R package which 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'll ever change your mind.



Windows Offline (64-bit) filesize: 68.02 MB	Instructions	browser.

If you use 32-bit and 64-bit browsers interchangeably, you will need to install both 32-bit and 64-bit Java in order to have the Java plug-in for both browsers. » FAQ about 64-bit Java for Windows

Mac OS X   Mac FAQ

Mac OS X (10.7.3 version and above) filesize: 74.17 MB	Instructions	After installing Java, you may need to restart your browser in order to enable Java in your browser.

* Oracle Java (Version 7 and later versions) requires an Intel-based Mac running Mac OS X 10.7.3 (Lion) or later and administrator privileges for installation. » More information

Linux

Linux RPM filesize: 63.4 MB	Instructions	After installing Java, you will need to enable Java in your browser.
Linux filesize: 79.29 MB	Instructions	
Linux x64 filesize: 76.35 MB	Instructions	
Linux x64 RPM filesize: 60.4 MB	Instructions	

Solaris

Solaris SPARC (64-bit) filesize: 52.24 MB	Instructions	After installing Java, you will need to enable Java in your browser.
Solaris x64 filesize: 50 MB	Instructions	

Figure 3.4: Downloading JRE from Oracle

6. Then you need to install the program “Xcode” from the app store.

7. After installation, you can use `R` by opening `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” (*the window which covers the left half 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, to explain what this means for users who are not familiar with documents which contain `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 #, 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 every line which does not start with either # or ## contains R code which you can copy and paste to the Console in RStudio and run. Alternatively, you can also copy the code to an R script and execute it by either clicking on this button Run ▾ on the upper right of RStudio, near the corner or you can use the shortcut "Ctrl+Enter". Both ways, the highlighted code or the line in which the caret is currently flashing are sent to the console and executed.

Now unfortunatly, 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_60, 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_60"
##Java(TM) SE Runtime Environment (build 1.8.0_60-b27)
##Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)
```

If your output looks like the above, you are ready to install rJava. If the first command does not show `1.8.0_60, x86_64` (or any other version staring with 1.8.), you need to install Java version 8 again (*see above*) and possibly reboot your computer. If the second command 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 excecuting the following command:

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

After that, you should be able to continue to the next section which describes Installing the programs themselves.

3.3 Linux

Since you are using Linux, we assume that you are sufficiently comfortable with using the command line. Therefore, we only provide the necessary steps for installing Java as commands.

First check if Java might already be installed:

```
$java -version
```

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

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

Optional: You can also install the Java development kit at this point, which is sometimes recommended for working with R and Java.

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

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 a recommended integrated development environment (IDE) called RStudio, which makes working with R a lot easier and also looks a better than the default GUI.

Install R on Linux

1. Since the version of R on 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.rstu
    dio.com/bin/linux/ubuntu artful/"
   ```

   Note, that you need to replace /ubuntu artful/ with your flavour and version of Linux. Visit CRAN to see which ones are 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. Seen below is how you would accomplish that in Ubuntu:

   ```
   $gpg --keyserver keyserver.ubuntu.com --recv-key E084DAB9
   $gpg -a --export E084DAB9 | sudo apt-key add -
   ```

   Or

   ```
   $sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E2
     98A3A825C0D65DFD57CBB651716619E084DAB9
   ```

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

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

4. Now install `RStudio` via gdebi (and install gdebi first if you don't already have it)[5]:

```
$sudo apt-get install gdebi-core
$wget https://download1.rstudio.org/rstudio-1.1.419-amd64.deb
$sudo gdebi -n rstudio-1.1.419-amd64.deb
$rm rstudio-1.1.419-amd64.deb
```

5. (Up until version 1.1.423, `RStudio` depends on an outdated version of (libgstreamer). You thus either need to use the new version (which is currently just a preview) or install the old version of (libgstreamer) using the method explained in this blogpost.)
6. After the installation has finished, you can use `R` by opening `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" (*the window which covers the left half 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. Since it looks similar to the terminal commands we used above, you probably have no problem reading it. But just in case, it might be worth to explain what you see there: 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 #, 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 every line which does not start with ## contains `R` code which you can copy and paste to the Console in `RStudio` and run. Alternnativly, you can also copy the code to an `R` script and execute it by either clicking on this button ➡ Run ▾ on the upper right of `RStudio`, near the corner or you can use the shortcut "Ctrl+Enter". Both ways, the highlighted code or the line in which the caret is currently flashing are sent to the console and executed.

[5] Alternatively, you can download an installation file from rstudio.com/products/rstudio/download/.

Now before we can actually run `rDNA`, we need to associate `Java` with R. To do this, you can either go back to the terminal, or you can invoke a system command directly from within `R` using the `system` function:

```
$sudo R CMD javareconf
```

Or:

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

After this is finished, you are now set to start installing `DNA` and the `rDNA`-package themselves.

3.4 Installing the programs themselves

Once `Java` is set up correctly, you can simply download the latest version of `DNA` as a JAR file from github.com/leifeld/dna/releases (*see Figure 3.5*). JAR or `.jar` files are technically archive files which usually contain a computer program written in `Java`, along with all the pictures and libraries 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 program from an "unidentified developer" (i. e., if the program has not been registered with apple) needs to be made a security exception before you can run it. 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"`.

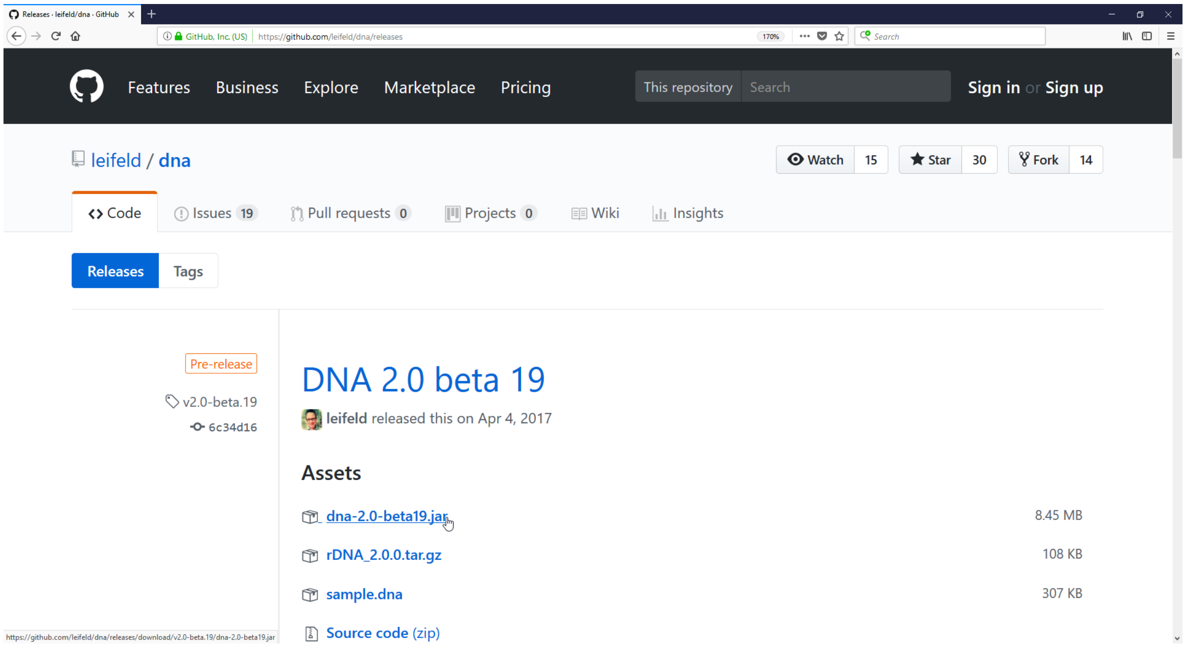
If you are not interested in using `rDNA`, you can now skip to the next section.

At this point, I assume that you have installed `R` and have at least a minimal understanding of how the program works. If that is not the case, you might want to jump back to where we explain how to install Install `R` on Windows, Install `R` on Mac or Install `R` on Linux. If you have already done this, we can go ahead and install `rDNA` from within `R`. First, we need to install the package `rJava` (Urbanek 2016), which is the most important dependency of `rDNA`:

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

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

```r
library("rJava")
# 1. initialize JVM
.jinit()
# 2. retrieve the Java-version
.jcall("java/lang/System", "S", "getProperty", "java.version")
```



Pre-release
v2.0-beta.19
6c34d16

DNA 2.0 beta 19

leifeld released this on Apr 4, 2017

Assets

dna-2.0-beta19.jar	8.45 MB
rDNA_2.0.0.tar.gz	108 KB
sample.dna	307 KB
Source code (zip)	

https://github.com/leifeld/dna/releases/download/v2.0-beta.19/dna-2.0-beta19.jar

Figure 3.5: Download DNA jar file from GitHub releases page

```
## [1] "1.8.0_151"

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

## [1] "/usr/lib/jvm/java-8-openjdk-amd64/jre"

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

## [1] "64"

# 5. retreive 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"
```

Now what you want to make sure, in case something is not working correctly with `rJava`, is if the architectures of `Java`, your operating system and your version of `R` match (*see comments 4., 5., and 6. in above's code chunk*).

Once this is done, you should install the package `devtools` (Wickham and Chang 2016), 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.[6]

```
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 `download.file` command (*see Section 8 for more information about this code chunk*).

```
# download two files necessary to test rDNA
download.file(
  "https://github.com/leifeld/dna/raw/master/manual/dna-2.0-beta20.jar",
  destfile = "dna-2.0-beta20.jar", mode = "wb")         # download DNA jar

# load library
library("rDNA")

# initialise the file you just downloaded
dna_init("dna-2.0-beta20.jar")

# 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 correclty, you are ready and set to use both `DNA` and `rDNA`. How you can do so will be described in the rest of this manual.

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

Chapter 4

Using DNA: Preparation of your DNA Workspace

Felix Rolf Bossner and Johannes Gruber

After installing the program (see Section 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.

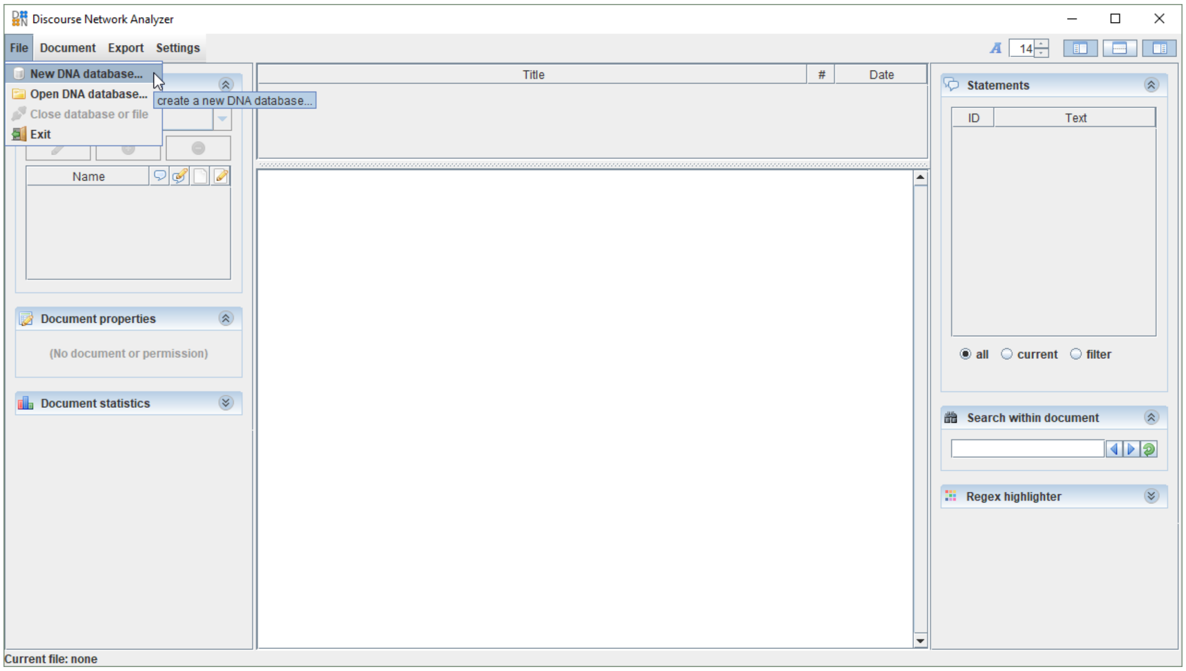
4.1 Creating a new DNA database

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:

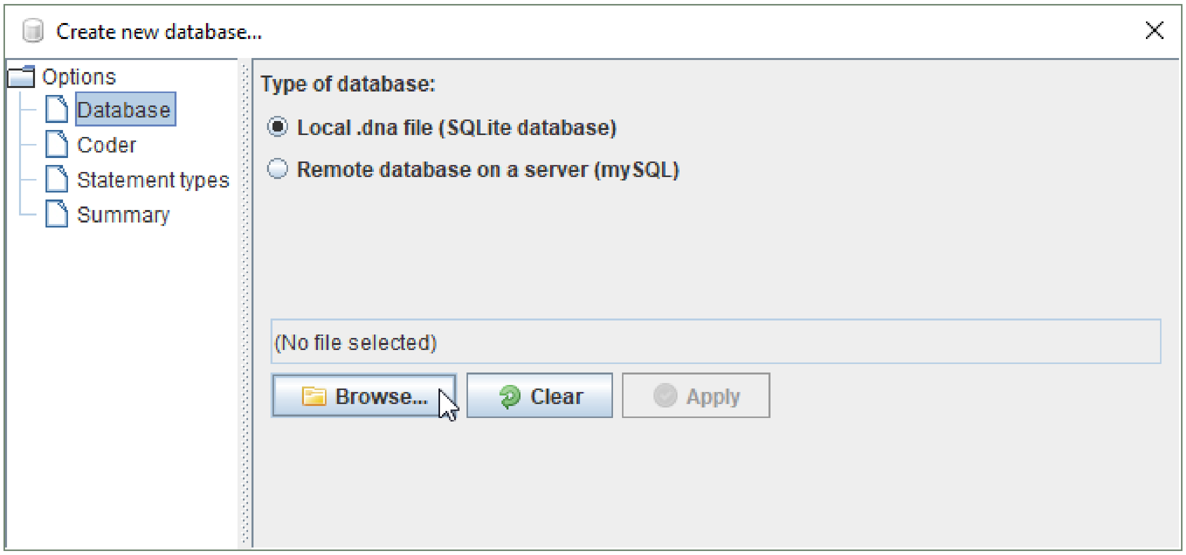
> The preset option **"Local .dna file"** means, that the dataset is stored in a local file [1] on your PC or device. This file, with the file extension .dna, can be moved on your machine, sent via email, uploaded and shared via a cloud file hosting service—such as Dropbox—and can generally be treated 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,

[1] Technically an SQLite file.



Current file: none

Figure 4.1: Starting a new Database



Type of database:

● Local .dna file (SQLite database)

○ Remote database on a server (mySQL)

(No file selected)

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

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 adequate in order to use DNAİt 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.

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 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 sever—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 descibes the necessary steps for Creating and using a remote database (MySQL).

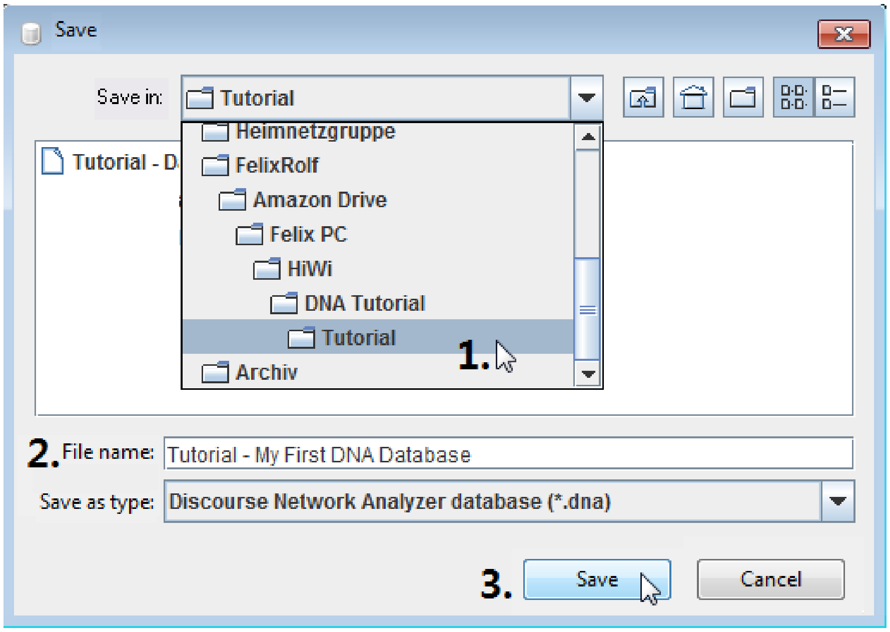
4.1.1 Creating a local DNA file

1. Click on the button **"Browse"** (*see Figure 4.2*). Now a pop-up menu—similiar 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.

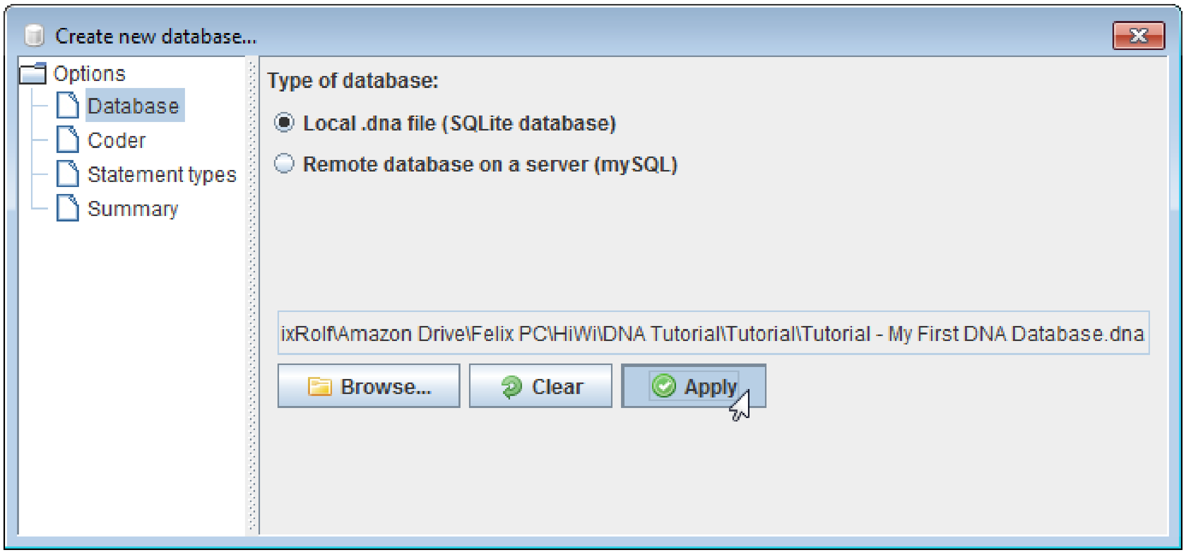


Save in: Tutorial

File name: Tutorial - My First DNA Database

Save as type: Discourse Network Analyzer database (*.dna)

Figure 4.3: Choose location of database window



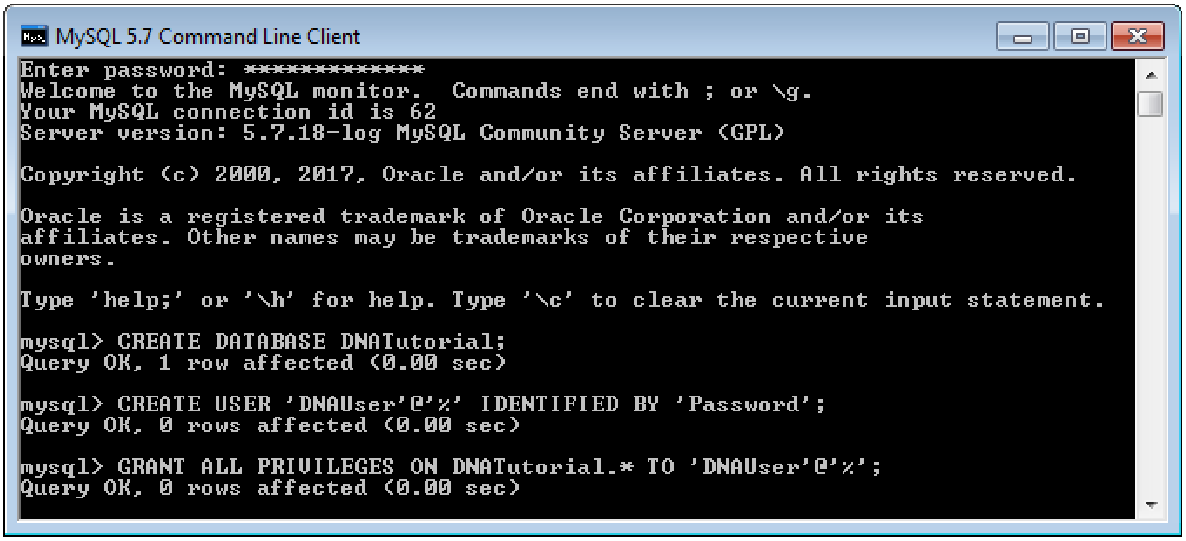
Type of database:

● Local .dna file (SQLite database)

○ Remote database on a server (mySQL)

ixRolf\Amazon Drive\Felix PC\HiWi\DNA Tutorial\Tutorial\Tutorial - My First DNA Database.dna

Figure 4.4: Apply database choice



```
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

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*).[2]

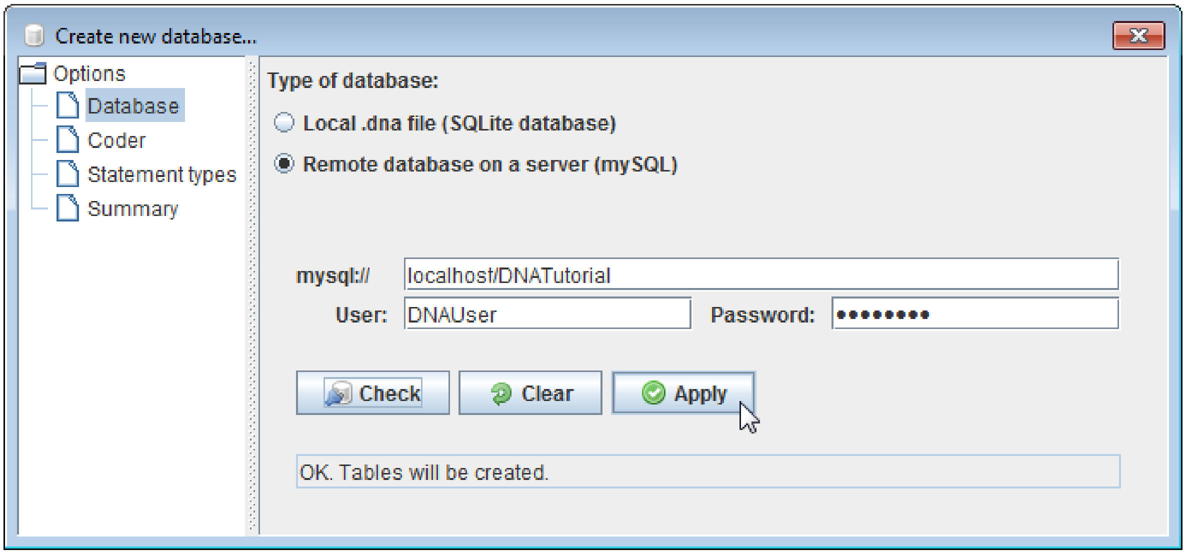
1. You have to create a database on your MySQL server (*usually by the command* `CREATE DATABASE 'DatabaseName'`).

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*

[2] For a detailed introduction to database management with MySQL see dev.mysql.com/doc/mysql-getting-started.



Type of database:

○ Local .dna file (SQLite database)

● Remote database on a server (mySQL)

mysql:// localhost/DNATutorial

User: DNAUser Password: ••••••••

OK. Tables will be created.

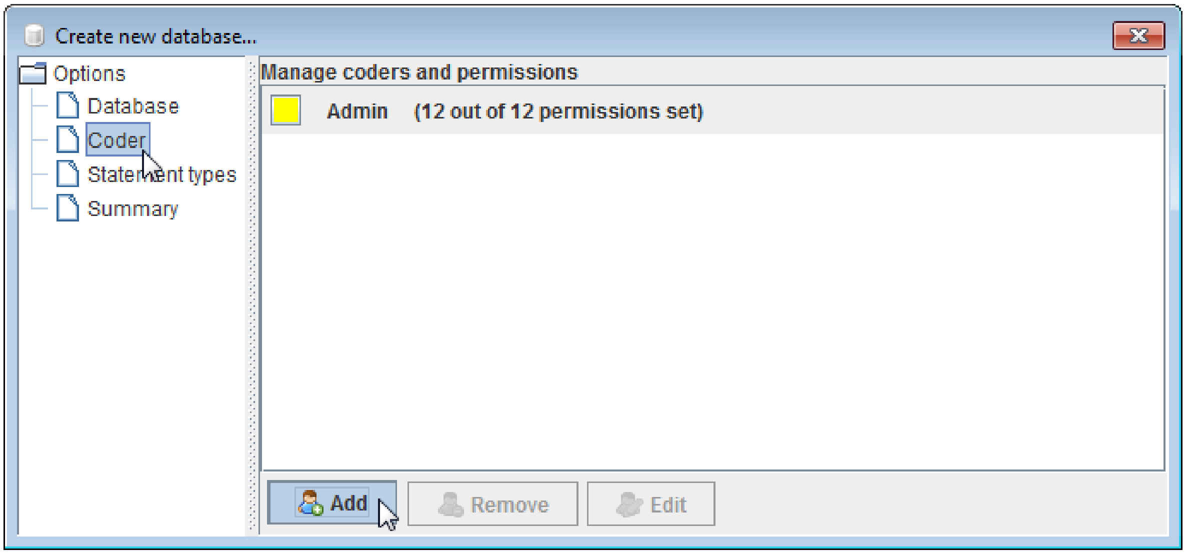
Figure 4.6: Connecting to local MySQL database

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 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

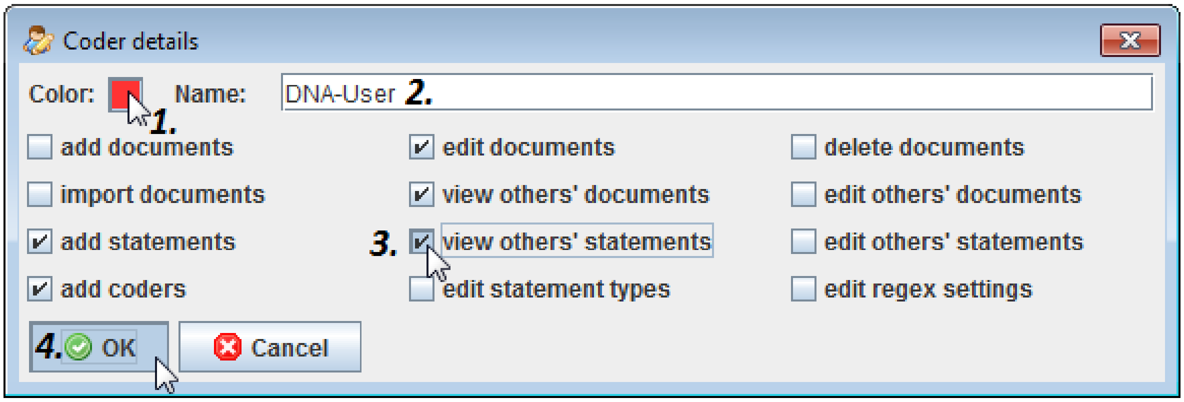
This second step of preparing your DNA workspace allows you to generate multiple user identities with different sets of rights for different coders. Thus, you can specify for each coder, which parts of the dataset each user can see or edit and thereby pre-structure your coding and research process. In 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



🟨 Admin (12 out of 12 permissions set)

Figure 4.7: Adding a second coder to the database



Color: 🟥 Name: DNA-User

☐ add documents	☑ edit documents	☐ delete documents
☐ import documents	☑ view others' documents	☐ edit others' documents
☑ add statements	☑ view others' statements	☐ edit others' statements
☑ add coders	☐ edit statement types	☐ edit regex settings

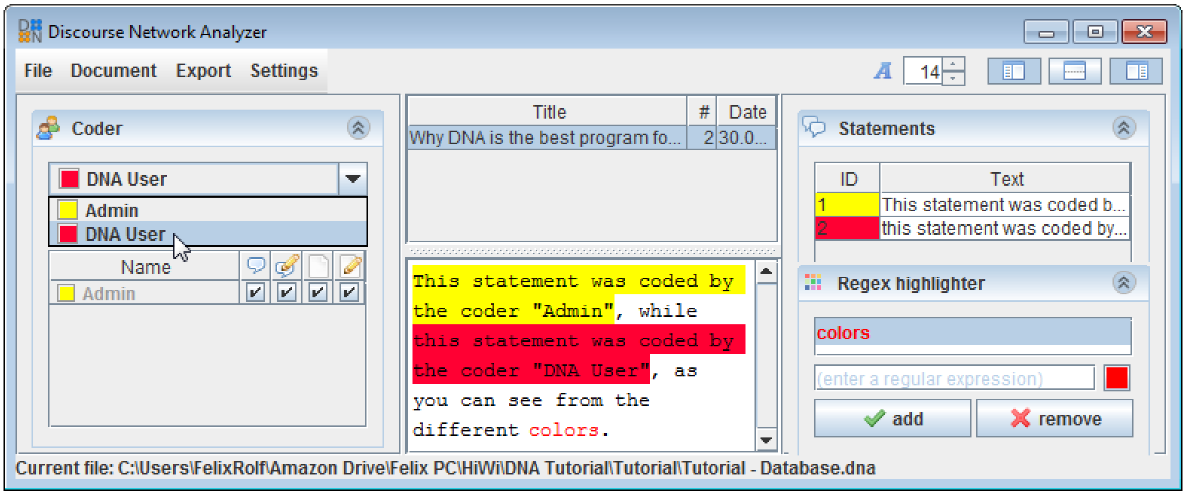
Figure 4.8: Configuring coder permissions

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 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



Title	#	Date
Why DNA is the best program fo...	2	30.0...

```
This statement was coded by
the coder "Admin", while
this statement was coded by
the coder "DNA User", as
you can see from the
different colors.
```

ID	Text
1	This statement was coded b...
2	this statement was coded by...

Current file: C:\Users\FelixRolf\Amazon Drive\Felix PC\HiWi\DNA Tutorial\Tutorial\Tutorial - Database.dna

Figure 4.9: Change coder identity

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 in *Table 4.1* . 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.

Table 4.1: User permissions explained

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

Table 4.1: User permissions explained

Permission	**Practical Implication**
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.

Table 4.1: User permissions explained

Permission	Practical Implication
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. 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.

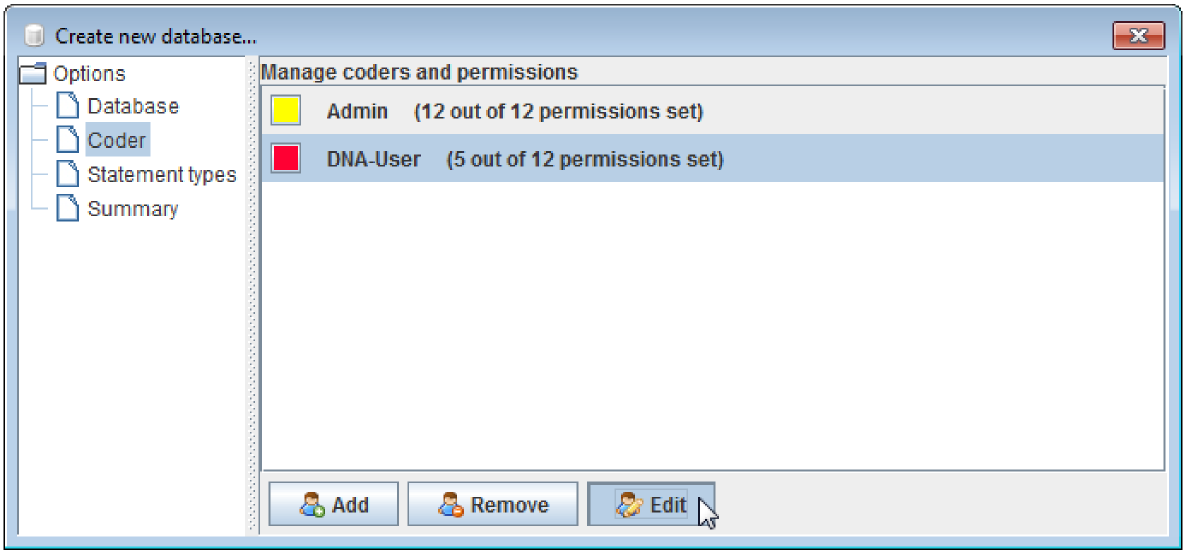
4.3 Statement Types and Variables

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.

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.

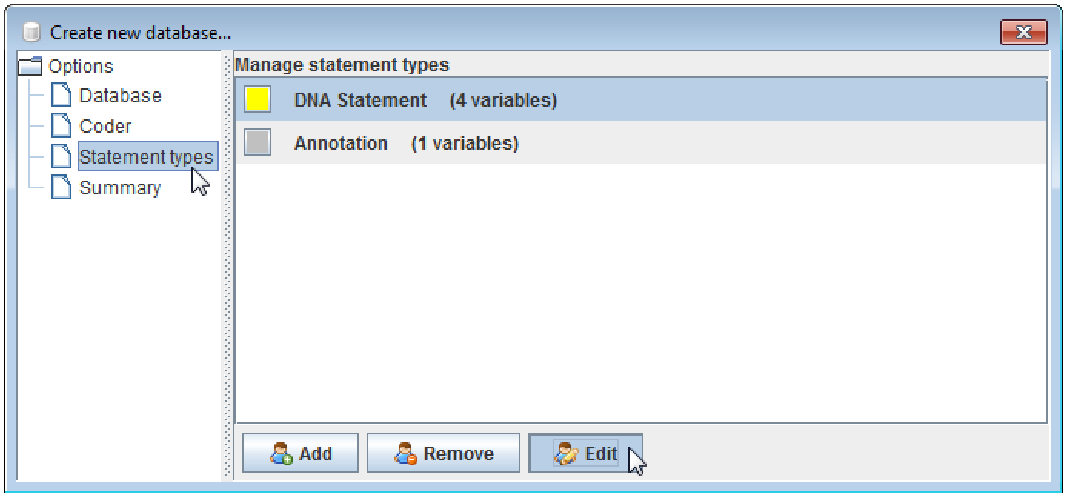


Manage coders and permissions

🟨 Admin (12 out of 12 permissions set)

🟥 DNA-User (5 out of 12 permissions set)

Figure 4.10: Edit coder details

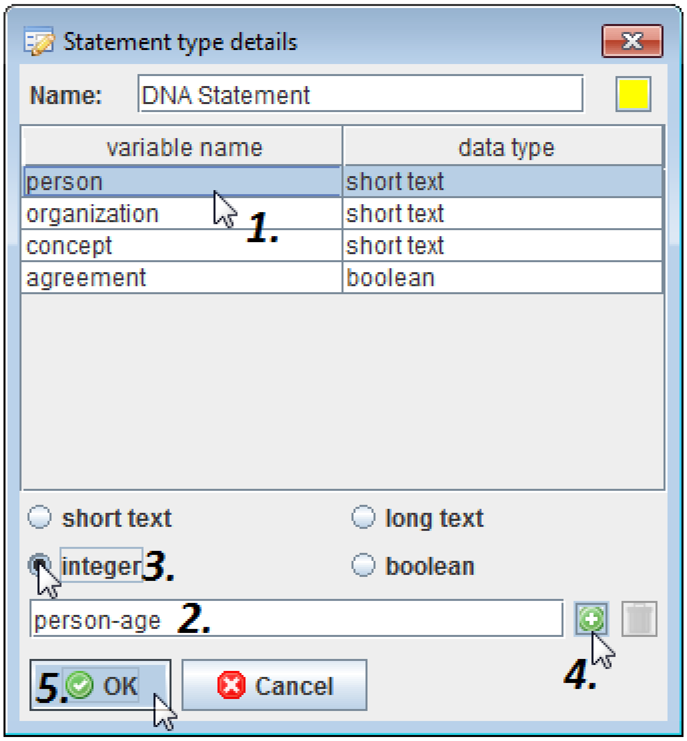


Manage statement types

🟨 DNA Statement (4 variables)

⬜ Annotation (1 variables)

Figure 4.11: Edit Statement Types



Name: DNA Statement

variable name	data type
person	short text
organization	short text
concept	short text
agreement	boolean

○ short text ○ long text

● integer ○ boolean

person-age

Figure 4.12: Edit Statement Type details

- 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 of the person who makes the statement. Please note, that DNA does

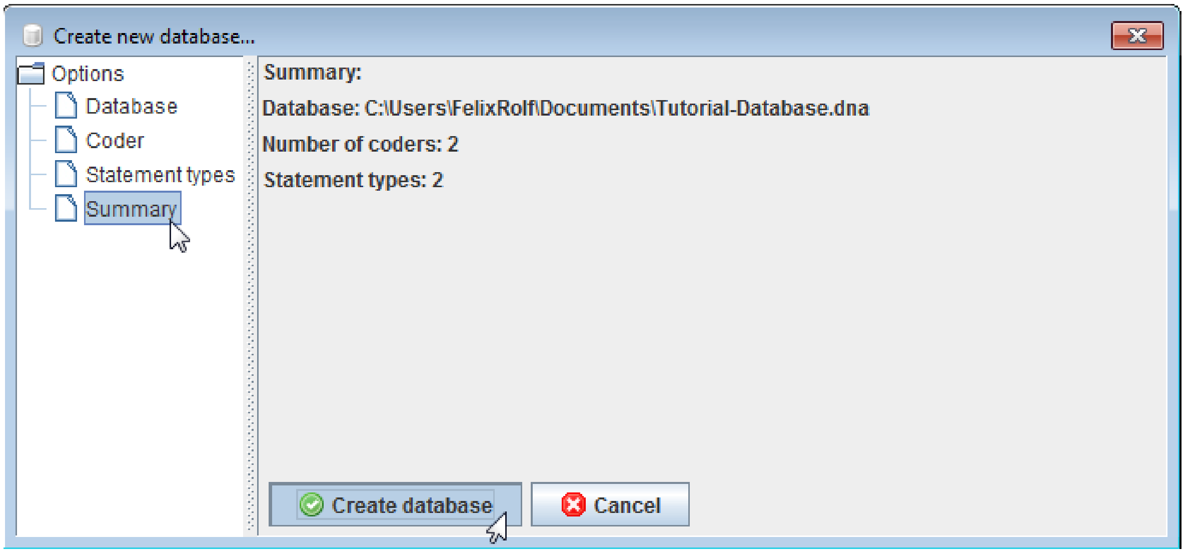
not allow spaces in variable names. Putting a space in the variable name will disable the “Add Variable” button necessary for step 4.

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” 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 neccesary 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**.



Summary:

Database: C:\Users\FelixRolf\Documents\Tutorial-Database.dna

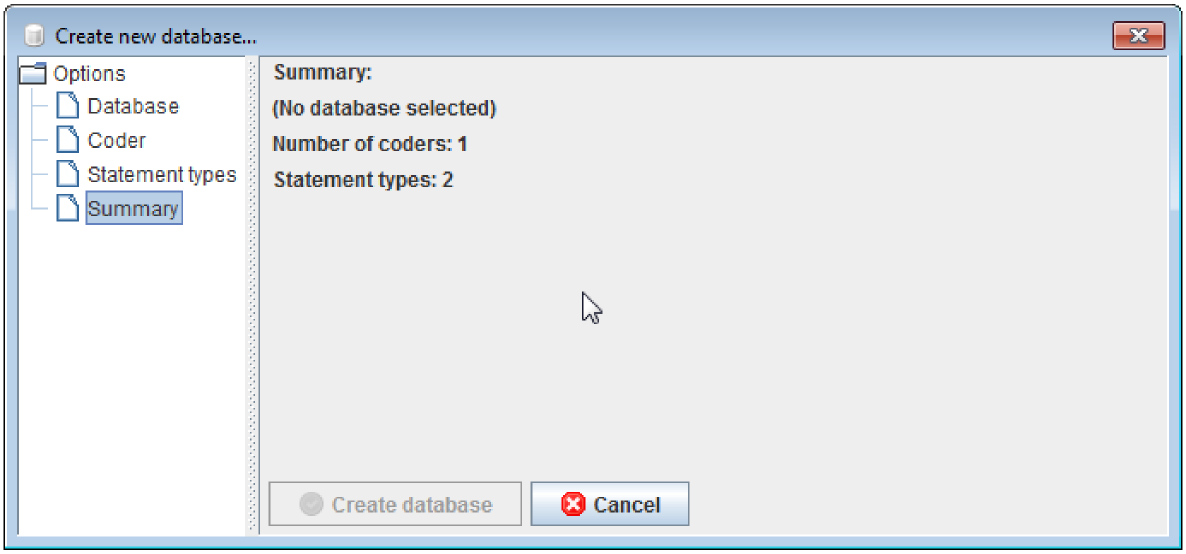
Number of coders: 2

Statement types: 2

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

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.



Summary:

(No database selected)

Number of coders: 1

Statement types: 2

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

Chapter 5

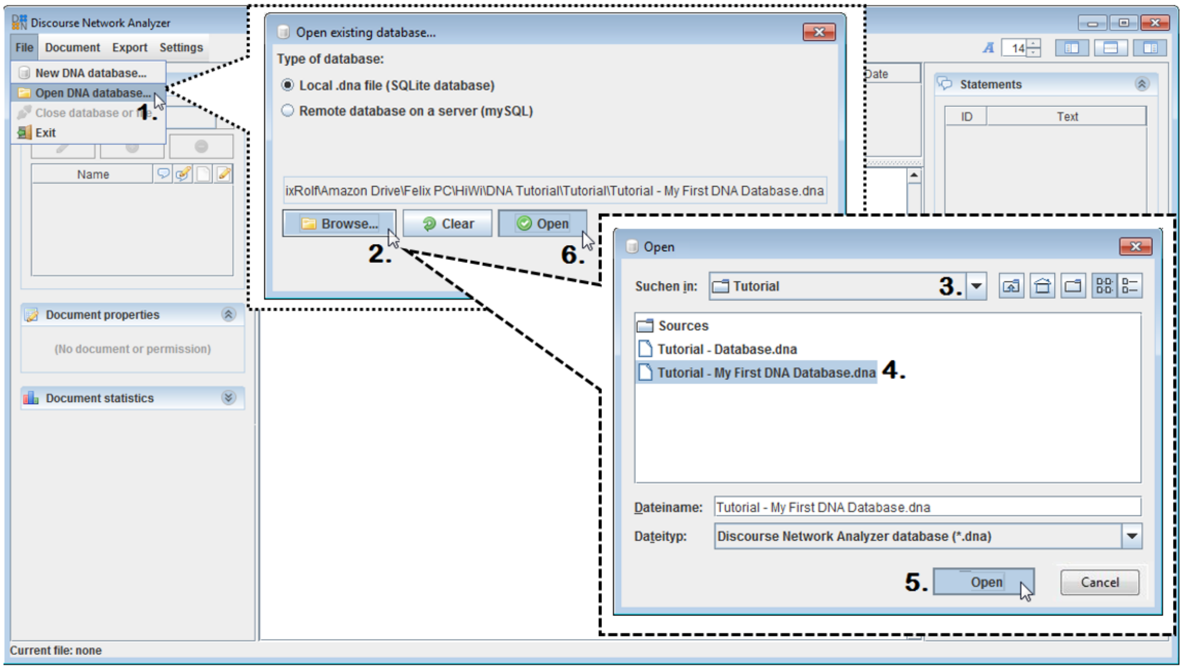
Using DNA: Importing and Organizing your Raw Data

Felix Rolf Bossner

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 documenst 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 DNAdatabase, 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*).



Type of database: ● Local .dna file (SQLite database) ○ Remote database on a server (mySQL)

ixRolf\Amazon Drive\Felix PC\HiWi\DNA Tutorial\Tutorial\Tutorial - My First DNA Database.dna

Dateiname: Tutorial - My First DNA Database.dna

Dateityp: Discourse Network Analyzer database (*.dna)

Figure 5.1: Opene DNAdatabase

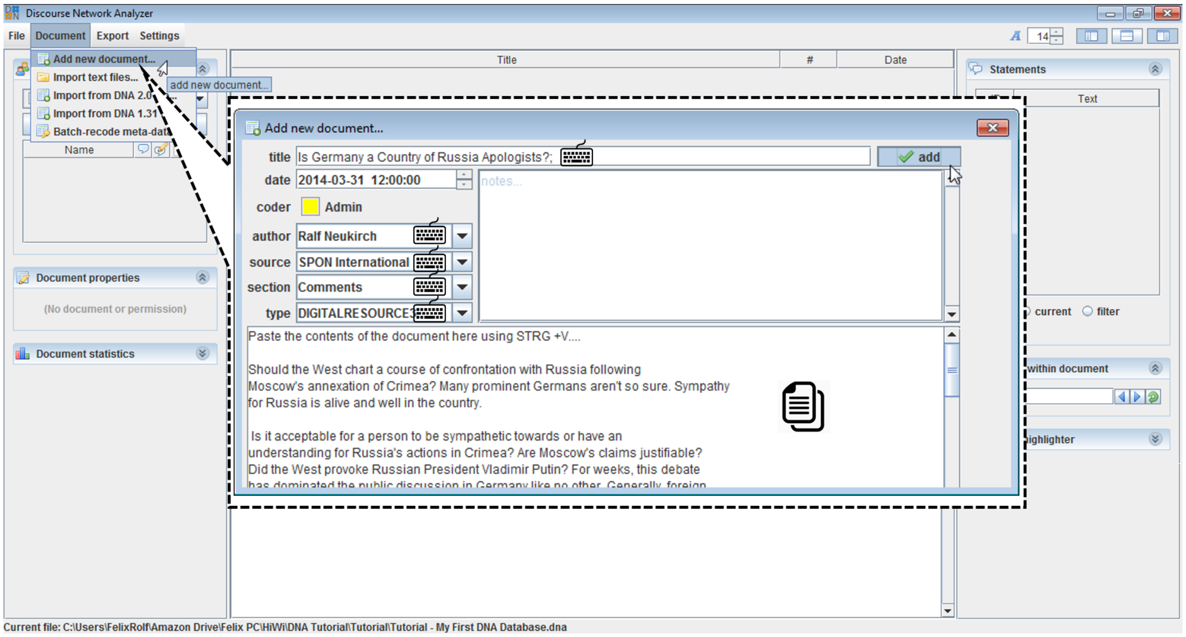
5.2 Importing Documents (Raw Data)

There are four 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, Importing Documents from other DNA databases and using rDNA to import data which is already available in R (WIP!). All four will be explained in detail in this section.

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



Add new document...

title: Is Germany a Country of Russia Apologists?;
date: 2014-03-31 12:00:00
coder: Admin
author: Ralf Neukirch
source: SPON International
section: Comments
type: DIGITALRESOURCE3
notes...

Paste the contents of the document here using STRG +V....

Should the West chart a course of confrontation with Russia following Moscow's annexation of Crimea? Many prominent Germans aren't so sure. Sympathy for Russia is alive and well in the country.

Is it acceptable for a person to be sympathetic towards or have an understanding for Russia's actions in Crimea? Are Moscow's claims justifiable? Did the West provoke Russian President Vladimir Putin? For weeks, this debate has dominated the public discussion in Germany like no other. Generally, foreign

Current file: C:\Users\FelixRolf\Amazon Drive\Felix PC\HiWi\DNA Tutorial\Tutorial\Tutorial - My First DNA Database.dna

Figure 5.2: Open DNA-database

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 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 reproducibilty, 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 buttonon 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.

- 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.

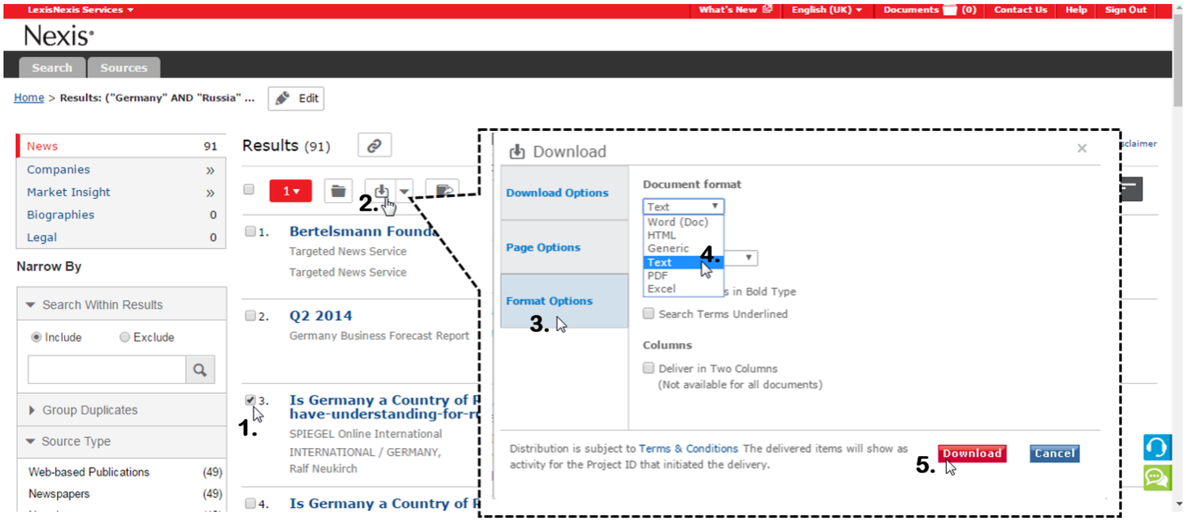


Figure 5.3: Downloading files from the LexisNexis newspaper archive

5.2.2 Importing multiple Documents semi-automatically from text files

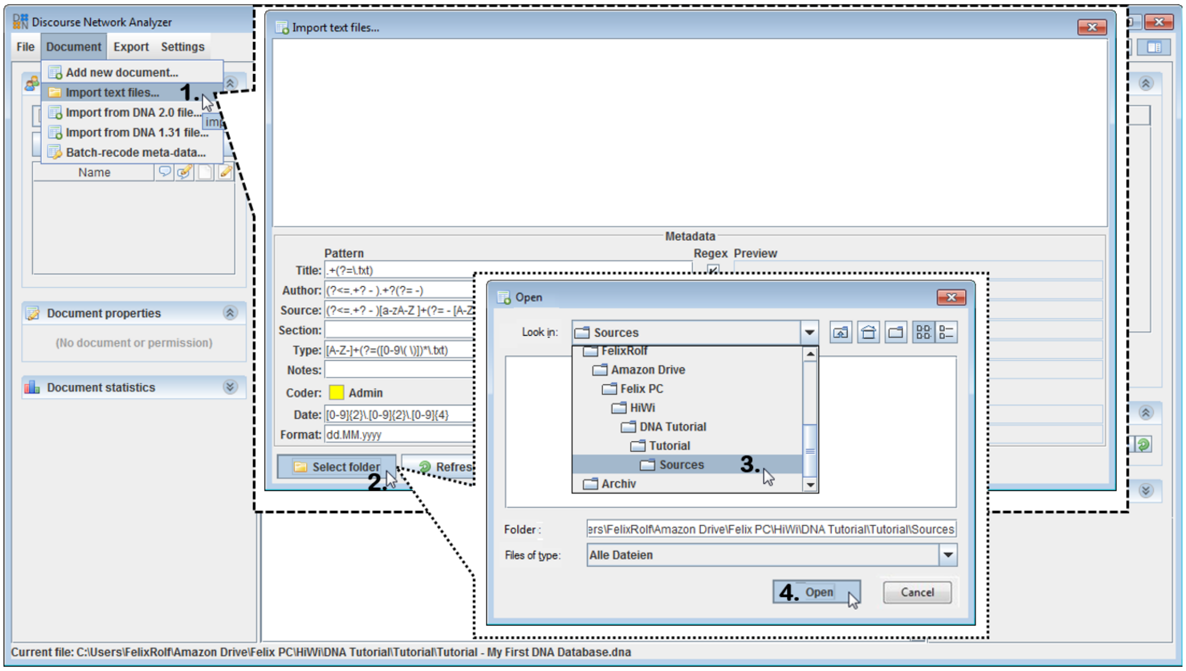
If you want to analyze a greater number of articles, it quickly becomes tedious to manually copy and paste each document and its meta data. This is why DNA also offers a semi-automatic way to upload multiple documents and their relevant meta data (author, date, source, type) at the same time.

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]

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 spon.de/aeclD, which

[1] You 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.

[2] If you use rDNA it will soon also be possible to import LexisNexis data into DNA via using rDNA and a new Rpackage called LexisNexisTools.



Pattern

Title: .+(?=\.txt)

Author: (?<=.+? -).+?(?= -)

Source: (?<=.+? -)[a-zA-Z]+(?= - [A-Z

Section:

Type: [A-Z-]+(?=([0-9\(\)])*\.txt)

Notes:

Coder: Admin

Date: [0-9]{2}\.[0-9]{2}\.[0-9]{4}

Format: dd.MM.yyyy

Regex ☑ Preview

Look in: Sources

Folder: ers\FelixRolf\Amazon Drive\Felix PC\HiWi\DNA Tutorial\Tutorial\Sources

Files of type: Alle Dateien

Current file: C:\Users\FelixRolf\Amazon Drive\Felix PC\HiWi\DNA Tutorial\Tutorial\Tutorial - My First DNA Database.dna

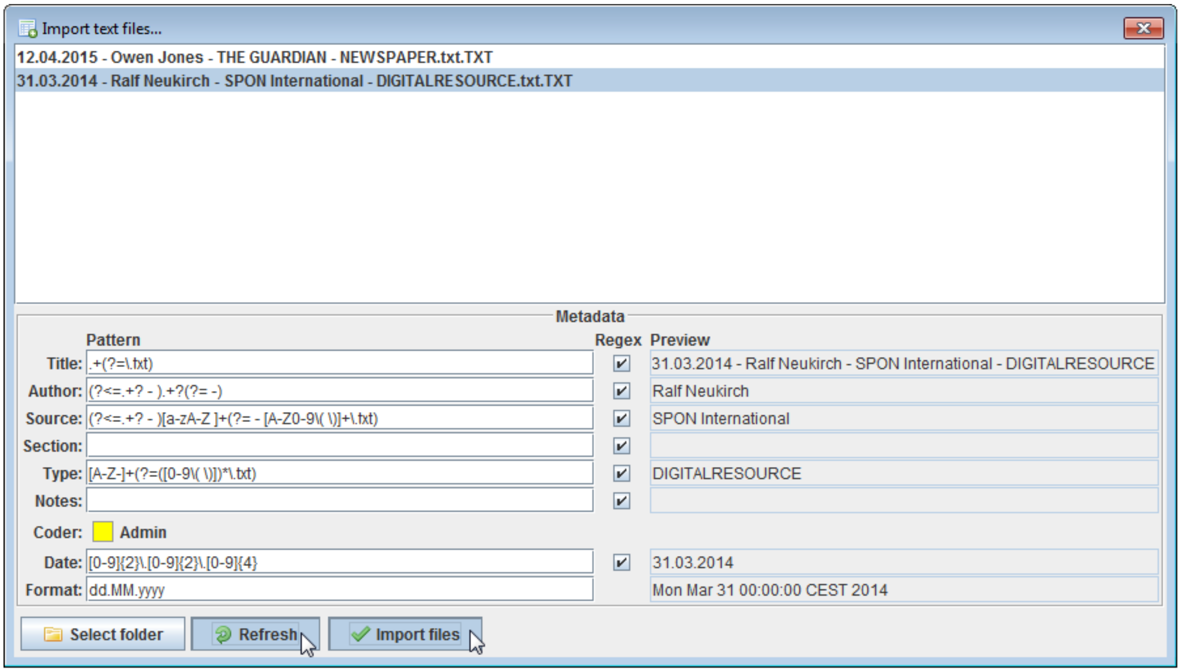
Figure 5.4: Import text files

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 somtimes 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



12.04.2015 - Owen Jones - THE GUARDIAN - NEWSPAPER.txt.TXT
31.03.2014 - Ralf Neukirch - SPON International - DIGITALRESOURCE.txt.TXT

Metadata

	Pattern	Regex	Preview
Title:	.+(?=\.txt)	☑	31.03.2014 - Ralf Neukirch - SPON International - DIGITALRESOURCE
Author:	(?<=.+? -).+?(?= -)	☑	Ralf Neukirch
Source:	(?<=.+? -)[a-zA-Z]+(?= - [A-Z0-9\(\)]+\.txt)	☑	SPON International
Section:		☑	
Type:	[A-Z-]+(?=([0-9\(\)])*\.txt)	☑	DIGITALRESOURCE
Notes:		☑	
Coder:	🟨 Admin		
Date:	[0-9]{2}\.[0-9]{2}\.[0-9]{4}	☑	31.03.2014
Format:	dd.MM.yyyy		Mon Mar 31 00:00:00 CEST 2014

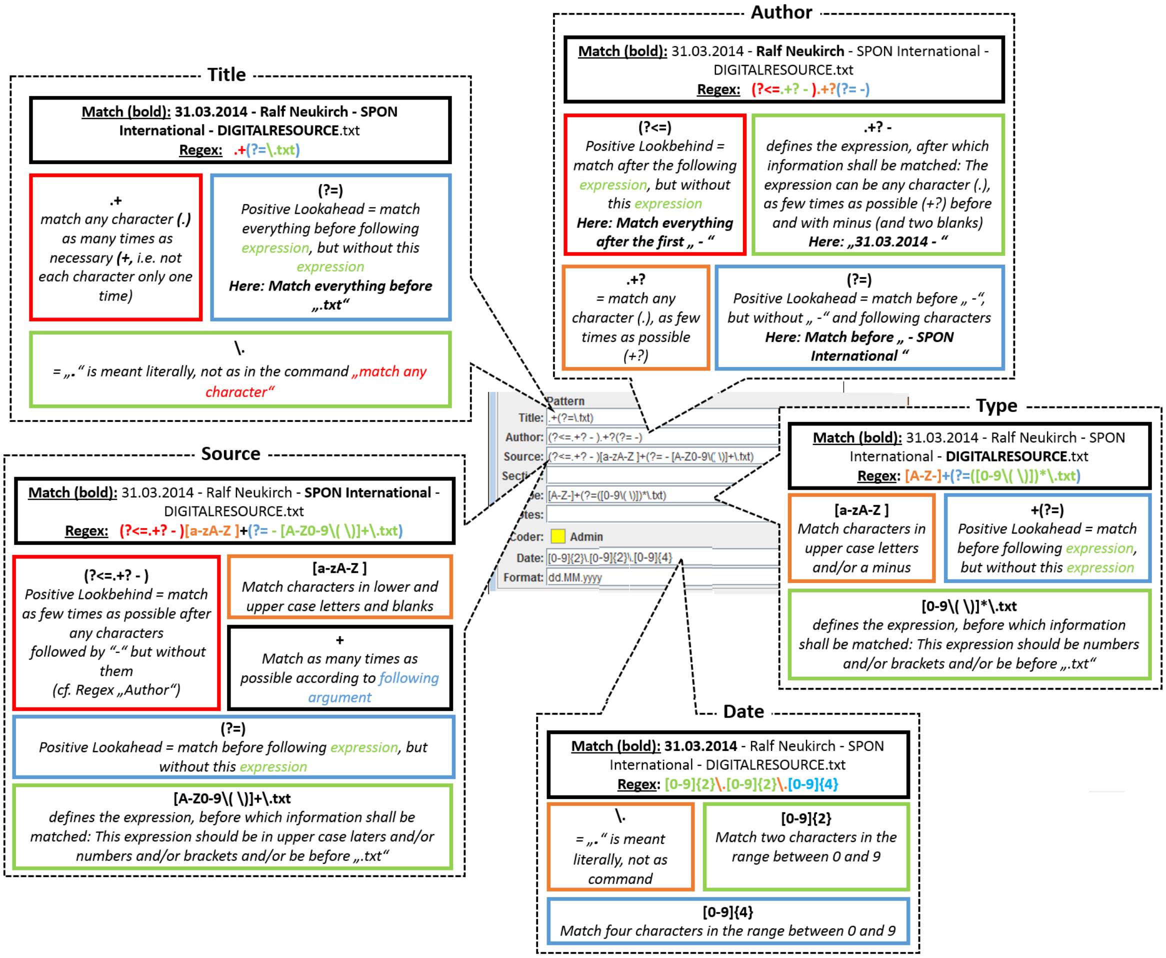
Figure 5.5: Import text files

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*).

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 expessions (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*



Title: .+(?=\.txt)
Author: (?<=.+? -).+?(?= -)
Source: (?<=.+? -)[a-zA-Z]+(?= - [A-Z0-9\(\)]+\.txt)
Section:
Type: [A-Z-]+(?=([0-9\(\)])*\.txt)
Notes:
Coder: 🟨 Admin
Date: [0-9]{2}\.[0-9]{2}\.[0-9]{4}
Format: dd.MM.yyyy

Title

Match (bold): 31.03.2014 - Ralf Neukirch - SPON International - DIGITALRESOURCE.txt
Regex: .+(?=\.txt)

.+
match any character (.) as many times as necessary (+, i.e. not each character only one time)

(?=)
Positive Lookahead = match everything before following expression, but without this expression
Here: Match everything before „.txt“

\.
= „.“ is meant literally, not as in the command „match any character“

Author

Match (bold): 31.03.2014 - **Ralf Neukirch** - SPON International - DIGITALRESOURCE.txt
Regex: (?<=.+? -).+?(?= -)

(?<=)
Positive Lookbehind = match after the following expression, but without this expression
Here: Match everything after the first „ - “

.+? -
defines the expression, after which information shall be matched: The expression can be any character (.), as few times as possible (+?) before and with minus (and two blanks)
Here: „31.03.2014 - “

.+?
= match any character (.), as few times as possible (+?)

(?=)
Positive Lookahead = match before „ -“, but without „ -“ and following characters
Here: Match before „ - SPON International “

Source

Match (bold): 31.03.2014 - Ralf Neukirch - **SPON International** - DIGITALRESOURCE.txt
Regex: (?<=.+? -)[a-zA-Z]+(?= - [A-Z0-9\(\)]+\.txt)

(?<=.+? -)
Positive Lookbehind = match as few times as possible after any characters followed by “-“ but without them (cf. Regex „Author“)

[a-zA-Z]
Match characters in lower and upper case letters and blanks

+
Match as many times as possible according to following argument

(?=)
Positive Lookahead = match before following expression, but without this expression

[A-Z0-9\(\)]+\.txt
defines the expression, before which information shall be matched: This expression should be in upper case laters and/or numbers and/or brackets and/or be before „.txt“

Type

Match (bold): 31.03.2014 - Ralf Neukirch - SPON International - **DIGITALRESOURCE**.txt
Regex: [A-Z-]+(?=([0-9\(\)])*\.txt)

[a-zA-Z]
Match characters in upper case letters and/or a minus

+(?=)
Positive Lookahead = match before following expression, but without this expression

[0-9\(\)]*\.txt
defines the expression, before which information shall be matched: This expression should be numbers and/or brackets and/or be before „.txt“

Date

Match (bold): **31.03.2014** - Ralf Neukirch - SPON International - DIGITALRESOURCE.txt
Regex: [0-9]{2}\.[0-9]{2}\.[0-9]{4}

\.
= „.“ is meant literally, not as command

[0-9]{2}
Match two characters in the range between 0 and 9

[0-9]{4}
Match four characters in the range between 0 and 9

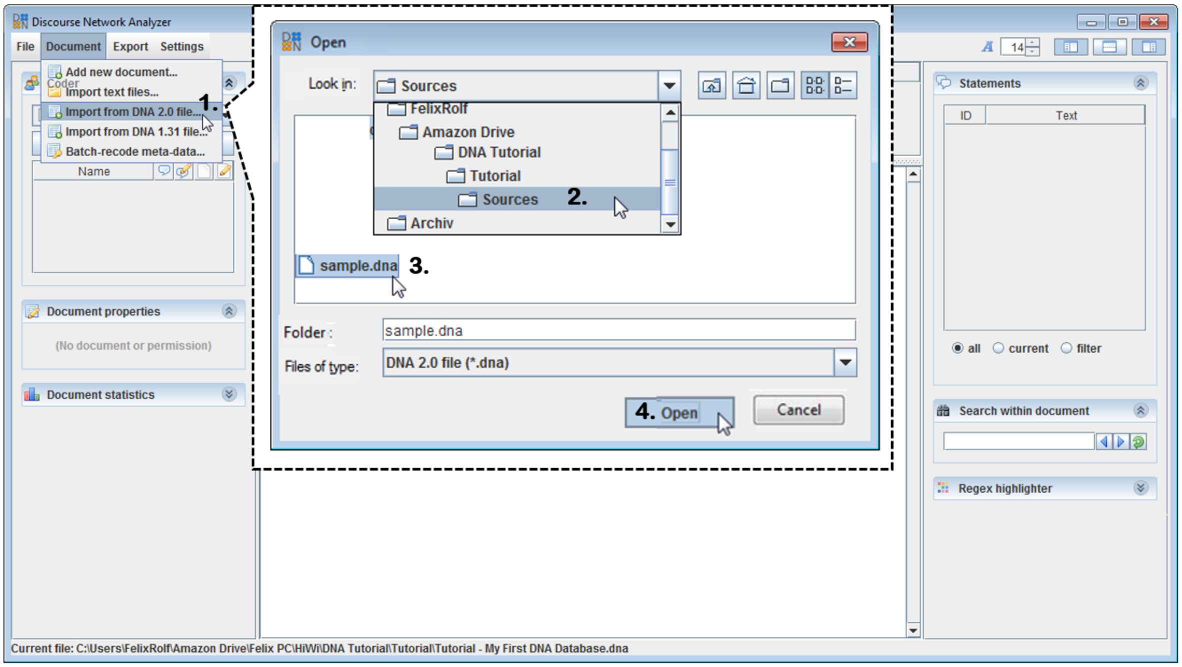
Figure 5.6: Import text files

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 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



Look in: Sources

Folder: sample.dna

Files of type: DNA 2.0 file (*.dna)

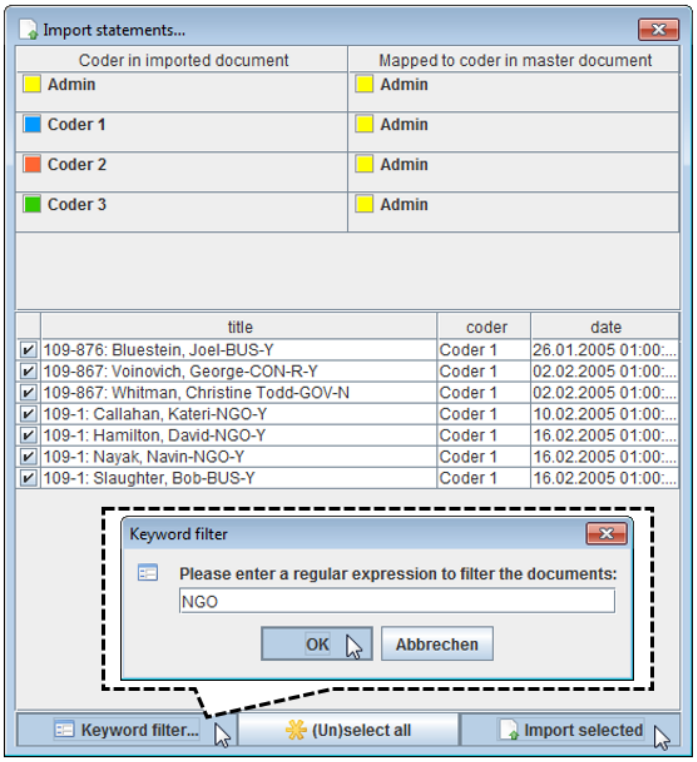
Current file: C:\Users\FelixRolf\Amazon Drive\Felix PC\HiWi\DNA Tutorial\Tutorial\Tutorial - My First DNA Database.dna

Figure 5.7: Import a DNA 2.0-database

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 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 impored using the seperate 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, 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 filer"



Coder in imported document	Mapped to coder in master document
🟨 Admin	🟨 Admin
🟦 Coder 1	🟨 Admin
🟥 Coder 2	🟨 Admin
🟩 Coder 3	🟨 Admin

	title	coder	date
☑	109-876: Bluestein, Joel-BUS-Y	Coder 1	26.01.2005 01:00:...
☑	109-867: Voinovich, George-CON-R-Y	Coder 1	02.02.2005 01:00:...
☑	109-867: Whitman, Christine Todd-GOV-N	Coder 1	02.02.2005 01:00:...
☑	109-1: Callahan, Kateri-NGO-Y	Coder 1	10.02.2005 01:00:...
☑	109-1: Hamilton, David-NGO-Y	Coder 1	16.02.2005 01:00:...
☑	109-1: Nayak, Navin-NGO-Y	Coder 1	16.02.2005 01:00:...
☑	109-1: Slaughter, Bob-BUS-Y	Coder 1	16.02.2005 01:00:...

Please enter a regular expression to filter the documents:

NGO

Figure 5.8: Import Statements menu

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 "ˆN" 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)

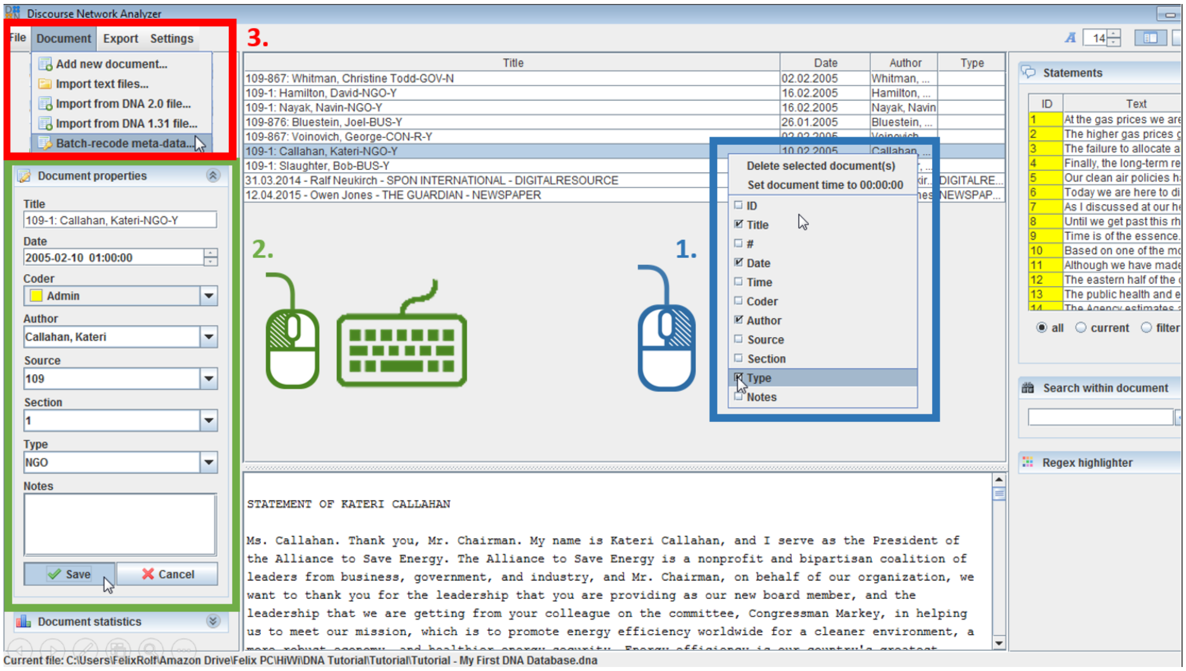
5.3.1 Deleting and navigating through documents

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 colums (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' meta data (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.



Title	Date	Author	Type
109-867: Whitman, Christine Todd-GOV-N	02.02.2005	Whitman, ...	
109-1: Hamilton, David-NGO-Y	16.02.2005	Hamilton, ...	
109-1: Nayak, Navin-NGO-Y	16.02.2005	Nayak, Navin	
109-876: Bluestein, Joel-BUS-Y	26.01.2005	Bluestein, ...	
109-867: Voinovich, George-CON-R-Y	02.02.2005	Voinovich, ...	
109-1: Callahan, Kateri-NGO-Y	10.02.2005	Callahan, ...	
109-1: Slaughter, Bob-BUS-Y			
31.03.2014 - Ralf Neukirch - SPON INTERNATIONAL - DIGITALRESOURCE		...kir...	DIGITALRE...
12.04.2015 - Owen Jones - THE GUARDIAN - NEWSPAPER		...nes	NEWSPAP...

Title: 109-1: Callahan, Kateri-NGO-Y
Date: 2005-02-10 01:00:00
Coder: 🟨 Admin
Author: Callahan, Kateri
Source: 109
Section: 1
Type: NGO
Notes:

ID	Text
1	At the gas prices we are
2	The higher gas prices g
3	The failure to allocate a
4	Finally, the long-term re
5	Our clean air policies h
6	Today we are here to di
7	As I discussed at our he
8	Until we get past this rh
9	Time is of the essence.
10	Based on one of the mo
11	Although we have made
12	The eastern half of the
13	The public health and e
14	The Agency estimates

● all ○ current ○ filter

STATEMENT OF KATERI CALLAHAN

Ms. Callahan. Thank you, Mr. Chairman. My name is Kateri Callahan, and I serve as the President of the Alliance to Save Energy. The Alliance to Save Energy is a nonprofit and bipartisan coalition of leaders from business, government, and industry, and Mr. Chairman, on behalf of our organization, we want to thank you for the leadership that you are providing as our new board member, and the leadership that we are getting from your colleague on the committee, Congressman Markey, in helping us to meet our mission, which is to promote energy efficiency worldwide for a cleaner environment, a more robust economy, and healthier energy security. Energy efficiency is our country's greatest

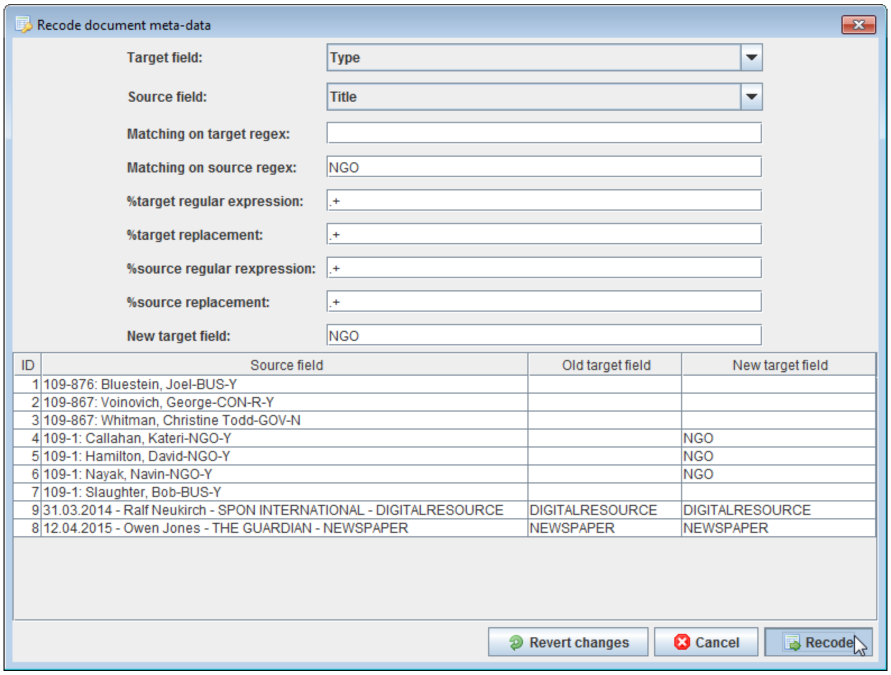
Current file: C:\Users\FelixRolf\Amazon Drive\Felix PC\HiWi\DNA Tutorial\Tutorial\Tutorial - My First DNA Database.dna

Figure 5.9: Import Statements menu

Editing the documents' meta data manually. The most basic way to edit your documents' meta data 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)* 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**:



Target field: Type

Source field: Title

Matching on target regex:

Matching on source regex: NGO

%target regular expression: .+

%target replacement: .+

%source regular rexpression: .+

%source replacement: .+

New target field: NGO

ID	Source field	Old target field	New target field
1	109-876: Bluestein, Joel-BUS-Y		
2	109-867: Voinovich, George-CON-R-Y		
3	109-867: Whitman, Christine Todd-GOV-N		
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		
9	31.03.2014 - Ralf Neukirch - SPON INTERNATIONAL - DIGITALRESOURCE	DIGITALRESOURCE	DIGITALRESOURCE
8	12.04.2015 - Owen Jones - THE GUARDIAN - NEWSPAPER	NEWSPAPER	NEWSPAPER

Figure 5.10: Meta information recode window

- 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. misspelled) 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 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. Similiar to all regex implementations in `DNA` you can either use search terms or regular expressions to filter the documents. If you, for instance, misspelled the author “Ralf Neukirch” sometimes as “Ralf Neu**n**kirch”, you can correct all your misspellings by simply selecting “Author” as “Target field”, entering “Ralf Neu**n**kirch” 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.

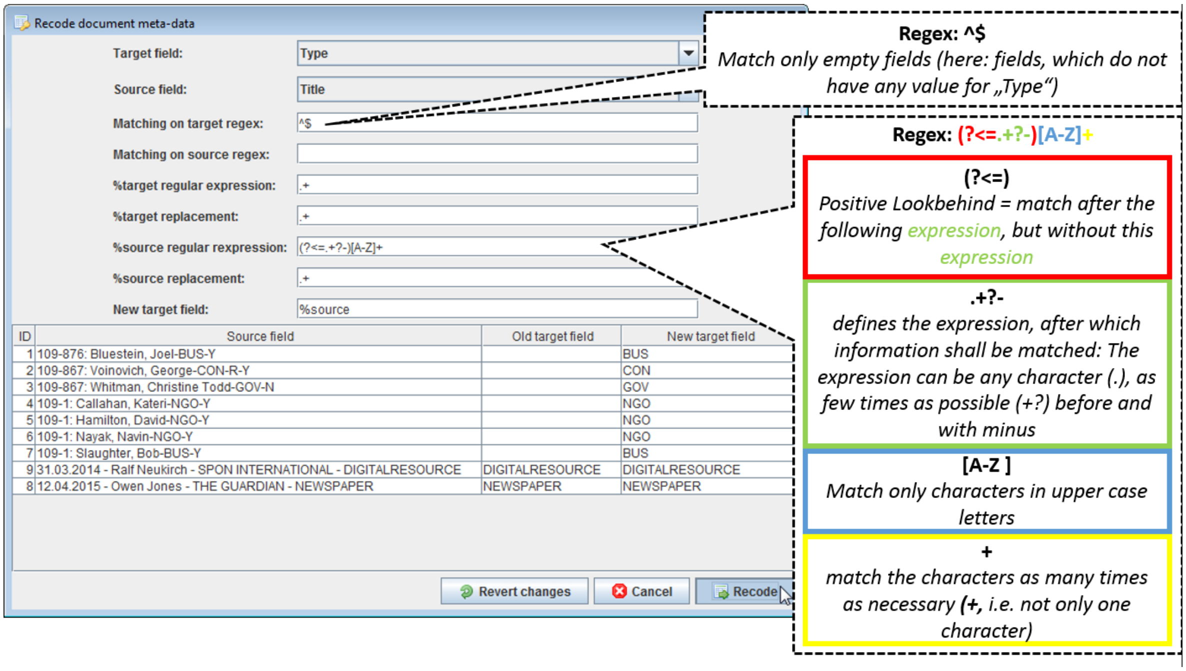
The lower half of the “Recode document meta-data” pop-up window (*see Figure 5.10*) displays a table with four columns and a row for each of your documents, which help you to preview, control and trace back your changes to the meta data:

- 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



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 rexpression:	(?<=.+?-)[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

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)

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.

Chapter 6

Using `DNA`: Coding the Data

Coming soon...

Chapter 7

Using `DNA`: Exporting the coded Data

Coming soon...

Chapter 8

rDNA: Using DNA from R

Philip Leifeld

DNA can be connected to the statistical computing environment R (R Core Team 2014) through the rDNA package (Leifeld 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. Installing additional R packages for network analysis and clustering, such as statnet (Goodreau et al. 2008; Handcock et al. 2008b, 2016), xergm (Leifeld et al. 2017a,b), igraph (Csardi and Nepusz 2006b), and cluster (Maechler et al. 2017), is recommended. Moreover, it is necessary to install the rJava package (Urbanek 2016), on which the rDNA package depends, and the devtools package (Wickham and Chang 2016), which permits installing R packages from GitHub (*see Section 3.4*).

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

After installing these supplementary packages, the rDNA package can be installed from GitHub. The devtools package contains a function that permits easy installation of R packages from GitHub and can be used as follows to install rDNA:

```
library("devtools")
install_github("leifeld/dna/rDNA")
```

Once installed, the rDNA package must be attached to the workspace:

```
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. We need to register DNA with rDNA to use them together. To do that, one needs to save the DNA .jar file to the working directory of the current R session and then initialize DNA as follows (with dna-2.0-beta20.jar in this example):

```
dna_init("dna-2.0-beta20.jar")
```

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 that is provided for download on GitHub under Releases:

```
dna_gui("sample.dna")
```

For this to work, the database file has to be saved in the working directory of the R session, or the path needs to be provided along with the file name.

In addition to opening the GUI, we will want to retrieve networks and attributes from DNA. For this to happen, a connection with a DNA database must first be established using the dna_connection function:

```
conn <- dna_connection("sample.dna")
```

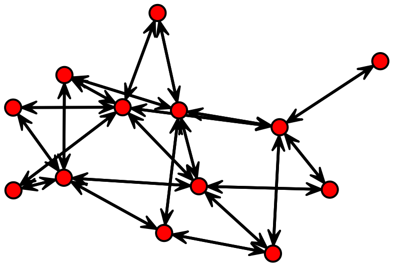
The dna_connection function accepts a file name of the database including full or relative path (or, alternatively, a connection string to a remote MySQL database) and optionally the login and password for the database (in case a remote MySQL database is used). Details about the connection can be printed by calling the resulting object called 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 can be plotted using visualization functions from the statnet suite of packages:

```
library("statnet")
gplot(nw)
```



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

```
at <- dna_attributes(conn)
```

The result is a data frame with organizations in the rows and one column per organizational attribute. 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. The help page for the `dna_network` function provides details on these arguments. It can be opened using the command

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

We will start with a simple example: a one-mode congruence network of organizations in a policy debate. The `sample.dna` database 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. 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")
```

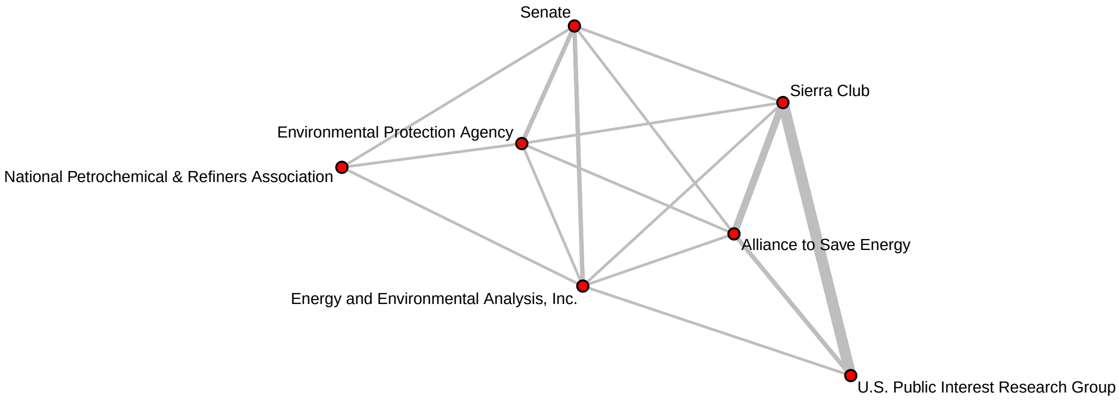
The result is an organization × organization matrix, where the cells represent on how many concepts any two actors (i e., the row organization and the column organization) had the same issue stance (by values of the qualifier variable `agreement`).

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 th 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`).

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:

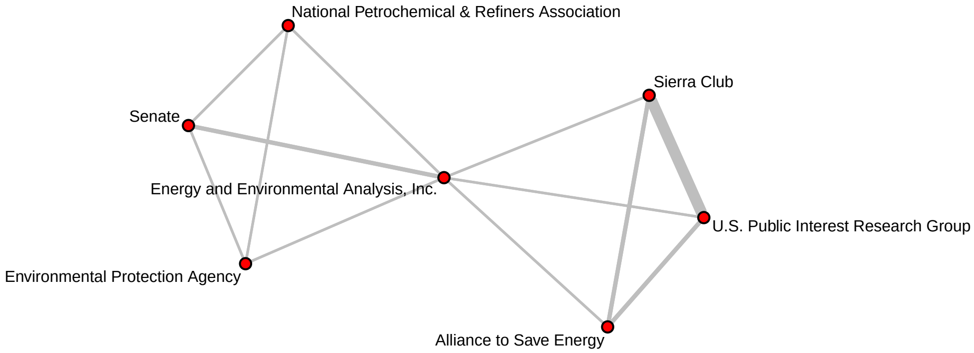
```
nw <- network(congruence)
plot(nw,
     edge.lwd = congruence^2,
     displaylabels = TRUE,
     label.cex = 0.5,
     usearrows = FALSE,
     edge.col = "gray"
     )
```


Senate
Sierra Club
Environmental Protection Agency
National Petrochemical & Refiners Association
Alliance to Save Energy
Energy and Environmental Analysis, Inc.
U.S. Public Interest Research Group

Here, we 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. 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 (2008b,c, 2015).

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. From our experience with the dataset, we can tell in this particular case that the concept “There should be legislation to regulate emissions.” is in fact very consensual. If everybody agrees to this concept, it 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"
     )
```


National Petrochemical & Refiners Association
Sierra Club
Senate
Energy and Environmental Analysis, Inc.
U.S. Public Interest Research Group
Environmental Protection Agency
Alliance to Save Energy

This reveals the structure of the actor congruence network. 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.

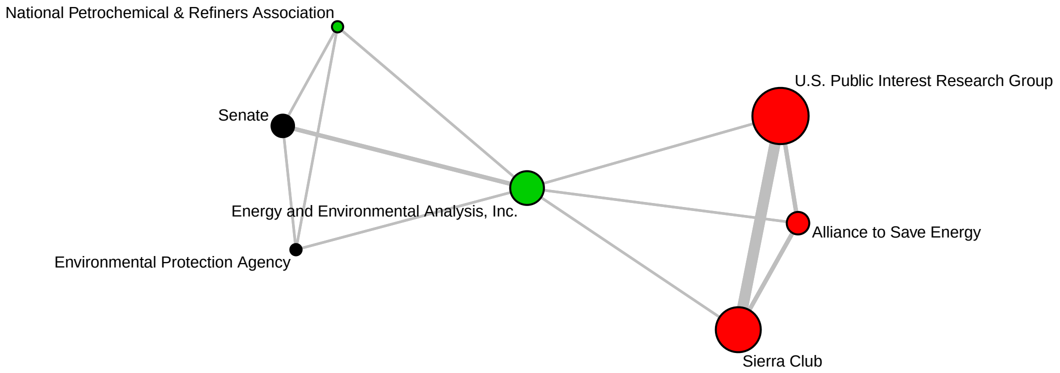
Next, we should tweak the congruence network further by changing the appearance of the nodes. We can use the colours for the organization types saved in the database and apply them to the nodes in the network. We can also make the size of each node proportional to its activity. The `dna_attributes` function 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:

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

##   id                                         value   color       type alias note
## 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 in dataset in network
## 1         2       TRUE       TRUE
## 2         3       TRUE       TRUE
## 3         1       TRUE       TRUE
## 4         1       TRUE       TRUE
## 5          2        TRUE        TRUE
## 6          4        TRUE        TRUE
## 7          5        TRUE        TRUE
```

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

```
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
     )
```


National Petrochemical & Refiners Association
U.S. Public Interest Research Group
Senate
Energy and Environmental Analysis, Inc.
Alliance to Save Energy
Environmental Protection Agency
Sierra Club

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.

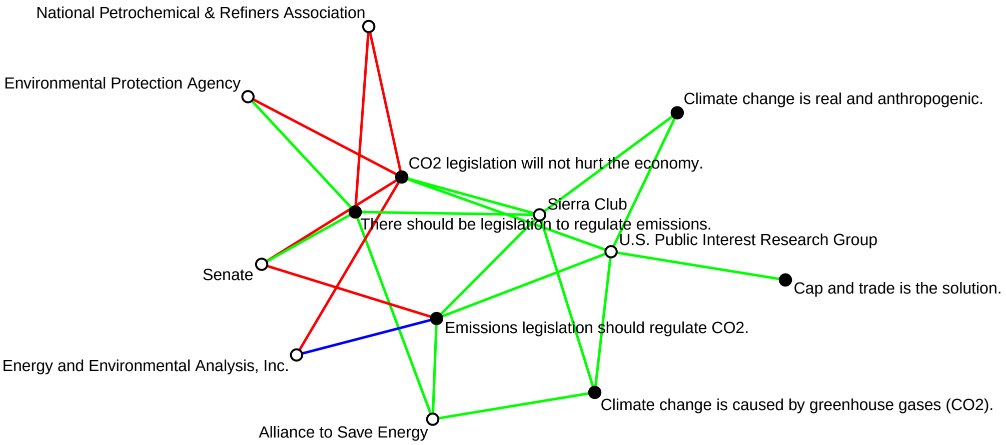
Another way to visualize a discourse network is a two-mode network visualization. To compute a two-mode network of organizations and concepts, the following code can be used:

```
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. `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"] <- "blue"
colors[colors == "2"] <- "red"
colors[colors == "1"] <- "green"
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
     )
```


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

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.

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.

Bibliography

Brandenberger, L. M., Schläpfer, I., Leifeld, P., and Fischer, M. (2015). Overlapping subsystems: Swiss water policy across media and parliament.

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: Market: The liberalization of the european gas market.

Butts, C. T. (2008a). network: a package for managing relational data in r. *Journal of Statistical Software*, 24(2).

Butts, C. T. (2008b). Social network analysis with sna. *Journal of Statistical Software*, 24(6):1–51.

Butts, C. T. (2008c). 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.

Cisneros, P. (2015). Subsystem interconnectedness as part of coalition strategies for policy change: Mining and water management in ecuador between 1991 and 2010.

Csardi, G. and Nepusz, T. (2006a). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Csardi, G. and Nepusz, T. (2006b). The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9.

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.

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. (2008a). statnet: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, 24(1):1–11.

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., and Morris, M. (2008b). `statnet`: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, 24(1):1–11.

Haunss, S., Dietz, M., and Nullmeier, F. (2017). Der ausstieg aus der atomenergie: Diskursnetzwerkanalyse als beitrag zur erklärung einer radikalen politikwende. *Zeitschrift für Diskursforschung*, (3):288–315.

Herweg, S. (2013). Politische diskursnetzwerke und der konflikt um das antipiraterieabkommen acta.

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.

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. (2016a). *Discourse Network Analysis: Policy Debates as Dynamic Networks.*

Leifeld, P. (2016b). *Policy Debates as Dynamic Networks: German Pension Politics and Privatization Discourse*, volume 29 of *Schriften des Zentrums für Sozialpolitik Bremen*. Campus, Frankfurt, 1. aufl., neue ausg edition.

Leifeld, P. (2018). *rDNA. A Package to Control Discourse Network Analyzer from R.* University of Glasgow, School of Social and Political Sciences, Glasgow. `R` package version 2.0.4.

Leifeld, P., Cranmer, S. J., and Desmarais, B. A. (2017a). *Temporal Exponential Random Graph Models with `btergm`: Estimation and Bootstrap Confidence Intervals*. Forthcoming.

Leifeld, P., Cranmer, S. J., and Desmarais, B. A. (2017b). *`xergm`: Extensions of Exponential Random Graph Models*. `R` package version 1.8.2.

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.

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., Rechkemmer, 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*.

Pedersen, T. L. (2017a). *ggraph: An Implementation of Grammar of Graphics for Graphs and Networks*. R package version 1.0.0.

Pedersen, T. L. (2017b). *tidygraph: A Tidy API for Graph Manipulation*. R package version 1.0.0.

Rantala, S. and Di Gregorio, M. (2014). Multistakeholder environmental governance in action: REDD+ discourse coalitions in tanzania. *Ecology and Society*, 19(2).

`R` Core Team (2014). *`R`: A Language and Environment for Statistical Computing*. `R` Foundation for Statistical Computing, Vienna, Austria.

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).

Rinscheid, A., Eberlein, B., and Schneider, V. (2015). Complex policy trajectories in risk domains: Nuclear power in Canada, Germany and Japan. In *International Conference on Public Policy (ICPP)*.

Schneider, V. and K., J. (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., Ramos, H., and Tindall, D. B. (2015). Environmentalists' mediawork for jumbo pass and the tobeatic 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. and Tindall, D. B. (2015). Canadian news media and the cultural dynamics of multilevel climate governance. *Environmental Politics*, 24(3):401–422.

Tosun, Jale, Schaub, and Simon (2015). To mobilize or not: political attention and the regulation of GMOs. In *Seventh International Conference on Coexistence between Genetically Modified (GM) and non-GM based Agricultural Supply Chains, Amsterdam.*

Urbanek, S. (2016). *rJava: Low-Level R to Java Interface.* R package version 0.9-8.

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.

Werner, C. (2015). The relevance of left and right in EU affairs: A case study of German parliament debates on the Greek crisis. In *Conference "Democracy: A Citizen Perspective"*.

Wickham, H. and Chang, W. (2016). *devtools: Tools to Make Developing R Packages Easier.* R package version 1.12.0.

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.