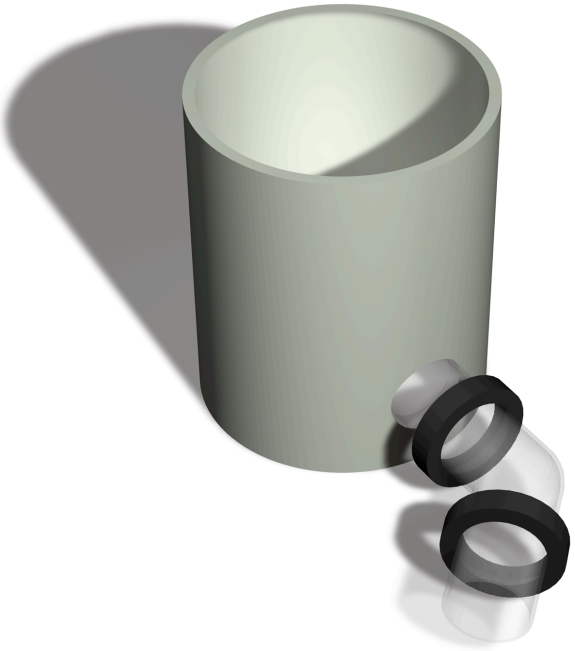
Informatic Tools to Explore Social Behavior in House Mice



author // **rico leuthold**

supervised by // **Prof. Barbara König**

Contents

List of Tables

List of Figures

1. Introduction

In May 2008, the research group around Prof. Barbara König at the Institute of Zoology of the University of Zurich decided to take their research in the field of Animal Behaviour to the next level. Especially interested in the complex social behaviour of house mice *(Mus domesticus)* (see section 2.2 on page 7 for details) they had the idea to design an automated system, to collect spatial position data of mice living in a shed.

1.1. Motivation

The motivation for such an effort is to obtain a statistically relevant set of data to develop, test and prove scientific thesis about the complex social behaviour of the house mice. The research interests are outlined in section 2.3 on page 8. An automated system has the advantage that it collects data 24 hours a day, and the mice are in no way disturbed in their natural behavior by the presence of men.

My personal motivation comes from the interest in informatics and my study in biology. The project gave me a lot of room to enhance my skills in informatics, while working on an interesting biological topic. Additionally I am interested in the *network approach* to analyze complex social behaviour by the use of *Graph theory*[13], which is outlined in section 7 on page 36.

The system collecting the data in the shed has been developed and constructed by *New Behaviour*. For further information about the design and operation of the system please see section 4.1 on page 12. Thus, the main task for my master thesis was to provide tools to store the data and make it accessible to the users, which are mainly the researchers involved in the house mice project.

1.2. Task

Handling the spatial position data collected by the automated system is challenging, as it includes several tasks which can be carried out only using informatics on a level that biologists normally can not cope with.

1.2.1. Storing and Accessing the Data

The sheer amount of data, expected to be collected by the automated system, has to be stored in a secure and stable way. The only reasonable way to fulfill these conditions, is to set up a Relational Database Management System[1] (RDBMS) (see section 4.3 on page 18).

To access the data, an intuitive user interface must be available, which facilitates the data exploration and export (see section 4.4 on page 26). The technology used to create the interface should be cross-platform compatible.

[1] A Relational Database Management System is a Database Management System in which data is stored in the form of tables and the relationship among the data is also stored in the form of tables[18].

1.2.2. Additional Tasks

Additionally, the user should be provided with an interface to administrate attribute data, such as the sex of the mice (see section 4.2 on page 17) and to carry out some simple data analysis methods.

Last but not least, some functionality should be present which allows the user to compile data for a possible analysis according to the network approach (see section 7 on page 36).

2. Biology of the house mice

2.1. Introduction

House mice (*Mus domesticus*) live under a multitude of different environmental conditions. Therefore they have to be very flexible in terms of their social, territorial an reproductive behavior[5,6,4]. Under natural conditions house mice have a very low expectancy of life (100 to 150 days). During this short life span, female mice are eager to have as many offspring as possible. Living under optimal food conditions, a female mouse has a litter of 5-7 pups every four weeks[27,35]. The juvenile mortality is estimated to be between 50% and 85%[27,28,34]. Having heavier pups is an advantage, as they can be weaned much faster, hence it has a positive influence on the future reproduction[25]. Litter size of wild house mice increases from the first to the second lactation, and decreases after the fifth lactation[35,33].

Figure 1: House mice *(Mus doemsticus)*

2.2. General social behavior

House mice typically live in a social collective which forms a reproduction group. These groups usually include a dominant male, one or several females with their litters and possibly some subordinate males[39,36,41,32].

The mating habit is generally polyandric (a female has many male reproduction partners), but observations of monogamic living pairs have been reported[31]. Juvenile females are allowed to stay in the territory of their parents to raise their own offspring[37]. There is evidence of stranger females immigrating into social groups as well[1,36,41,5,3].

2.3. Research topics

The research group in animal behavior around Professor König at the University of Zürich is generally interested in the complex social behavior of the house mice. Mainly the influence of the social partner choice, other then mating, onto the fitness of the female mice, is in the focus of the studies. Such a social selection only exists, if the fitness benefit for the individual mouse is measurable. Therefore, reproductive cooperation such as joint nesting, for example, should manifest in an increased number of offspring [23]. Cooperative interactions can extend to sharing of broad-rearing duties like nursing.

The following short summaries of articles should give some insight into the current projects.

2.3.1. Communal nursing

Over forty years ago, first observation of wild female house mice, belonging to the same reproduction group, which raise their pups in communal nests have been reported [42]. Ever since, this behavior has been noticed for house mice living under all kind of environmental conditions. [39,40,38,26,3].

This is an astonishing behavior, as the energetic cost of lactation is extremely high, hence influences the females future reproduction success. To wean a litter of about seven to eight pups, the mother has to produce 100 grams of milk within 22 days, which is equivalent to 1100 KJ[2] [2]. Depending on the size of the litter and the duration of the lactation phase, the length of the interbirth intervals change [24,25,30,33].

Several experiments have been carried out with house mice living under laboratory conditions. Changes in group size and the composition of the group in terms of relatedness of the individuals revealed, that non-offspring nursing is an integral part of the reproductive behavior of female house mice in egalitarian groups. However, the probability for such mutualistic cooperation was highest when a female shared a nest with a familiar sister to form a low-skew society [29].

According to Koenig [29], the direct benefits of allomaternal care for the pups could be diverse. As there are, improved survival, improved future reproduction, improved growth or immunological benefits.

There are physiological benefits for the mother as well, which lead to an increase of their reproductive output. Of notably interest is the hypothesis named *Metabolic peak load reduction*. Explained in short, a solitary nursing female tends to have a peak of energy consumption during the nursing phase of her litter. Nevertheless, female house mice are limited in their maximal sustainable metabolism [7]. This limit depends on the age and physique of the individual and is called *metabolic ceiling*. Partial synchronization in reproduction within females of a group, combined with the communal nursing, can balance the energy demand for the individual female, as the burden of nursing the litter is divided. Hence, the energy demand for

[2] The Joule is a derived unit of energy and defined as the work done by a force of one newton travelling through a distance of one metre. [15]. 1 Kilo Joule = 10^3 Joule.

each female stays always on a medium level and therefore a lower energy demand over the whole lifetime [29]. This can result in an increased reproduction success.

However, cause to the artificial setup of the experiments, no conclusion about the communal nursing under natural conditions could be made.

2.3.2. Home range

Olivia Dieser [8] carried out a long-term analysis to investigate the territorial social behavior of the house mice population living in the shed (see section 3 on page 10). Data used in this work has been retrieved via the *miceminer* application (see section 6.2.3 on page 35 and section 6.2.4 on page 35). Based on the data of 321 individual mice, collected over a period of 15 months, she analyzed the influence of the sex, the season and the time of day, on the usage of the artificial nestboxes. Furthermore, the seasonal influence on the composition of the mouse population has been determined. Refer to section 4.1 on page 12 for details about the artificial nestboxes and the system which is collecting the data.

She found out, that during summer, when the mating activity is high, more female than male mice lived in the barn, while in winter the ratio was about one. This result can be explained by the polyandric mating behavior of the house mice. Only dominant male mice get access to a female for mating. Hence, subdominant male mice are banished from the shed by the dominant mice. This finding is supported by the fact, that less overlaps of the territories for male mice have been detected in summer than in winter.

Females, on the other hand, used more boxes during the summer, possibly to increase their chances of mating with many different males.

In general, female mice visited more different artificial nestboxes than male mice. This could be explained by the limit of boxes male mice can defend to mark their territory.

3. Shed setup

The shed, which is a single room building located in a forest near Effretikon (Switzerland), covers an area of 64m^2 (12.8m x 5.75m). Stable plastic walls (height ~50cm) divide the area in to four sectors and an entry space. Small transit holes in the dividers ensure access to the different segments (see figure 3 on page 11).

The entry area provides workspace for the researchers and is used to store tools and material. Furthermore the central computer for the data collection system (see section 4.1 on page 12) as placed in this area.

The 40 artifical nestboxes (see figure 6 on page 14 for a schematic model) are distributed in the four sectors (see figure 3 on page 11 for the current positioning of the boxes) along with some plastic pipe structures, bricks and smaller plastic walls and shelters to structure the environement (see figure 2 on page 10). These structuring elements ensures the existence of several terretories, and provide hideouts for the subordinate mice.

Figure 2: Overview of the interior of the shed. Visibel are the artificial nestboxes with the box number on the white cover panel, the grey colored sector dividers and the accessory elements to structure the are.

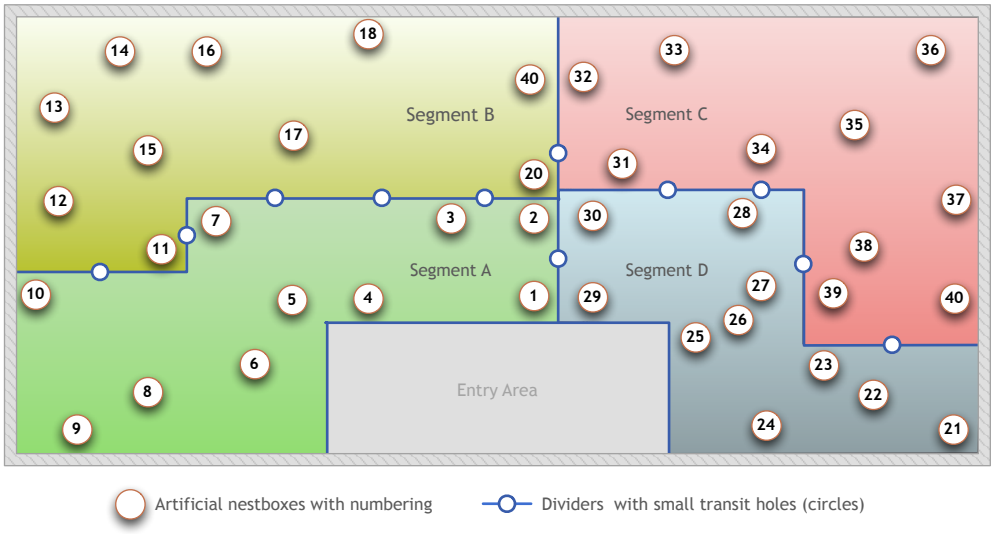


Figure 3: Schema of the shed including the box postions and the plastic walls dividing the area in the four segments.

In addition, there is one dedicated pipe which connects the interior to the outside world, allowing the mice to freely leave or enter the shed. This connection is not included in the data mentioned in section 5 on page 27.

4. Data

When gathering data for behavioural analysis of different species, typically the observer tracks the position of one or several individuals over the time. Additionally pheno- and genotypical data of the individuals is collected.

Usually the spatial postion data is collected by one ore a few people. In this project the data is collected by an antenna system (see section 4.1 on page 12). This approach has the advantage, that the data is recorded 24 hours a day, without presence of people needed.

To collect phenotypical data, so called *population checks* (see 4.2 on page 17 for details) are conducted every 6 to 8 weeks. During these checks, the mice get caught for measuring, weighing (see section 4.2 on page 17 for details) and - if not already present and the mouse has a weight of at least 18 grams - an Radio-frequency identification[3] (RFID) transponder is injected under the skin of the mouse (see figures 4 and 5). About 240 transponders are injected every Year.

Figure 4: Pictured an ID-100A Microtransponder from $Trovan^{©}$. The transponder weighs 0.1 g and the coat is made of biocompatible glass.
Picture courtesy of Trovan.

The genetical information is needed to get information about the relatedness between the mice within the population. Genotypical information is not included in the data used for this work.

4.1. Collecting spatial position data

To collect the spatial position data, a project specific system has been by built by *New Behaviour*, a company specialized in creating systems to study animal behaviour.

The basic idea is to use Radio-frequency identification[4] transponders, to identify the mice at specific locations in the shed. These Radio-frequency identification transponders are about the size of a rice corn and are injected under the skin of the mice (see figure 5 on page 13).

[3] Radio-frequency identification (RFID) is the use of an object (typically referred to as an RFID tag) applied to or incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves [19]

[4] Radio-frequency identification (RFID) is the use of an object (typically referred to as an RFID tag) applied to or incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves [19]

Figure 5: Subcutaneous injection of an RFID transponder.

Furthermore, the system includes 40 artificial nestboxes, made of PVC[5], which have a diameter and a height of 15 centimeters. The entry tube is about 20 to 25 centimeters long, made of *Plexiglass*, and is bent by an angle of 45 degrees to slow down the mice when they go trough the pipe (see fig. 6 on page 14). This is improving the accuracy of the two antennas which are wrapped around the tube and read the RFID transponders. The antennas have a coverage area with a radius of about 10 to 12 centimeters.

The Radio-frequency identification transponders used are passiv, meaning that they do not include a battery. The antenna acts as a scanner, presenting an inductive field which excites the Radio-frequency identification transponder when entering the coverage area of the antenna. This energy is used by the transponder to send its identification to the antenna.

So, whenever a mouse carrying a transponder passes by an antenna, it's identification is recorded and sent to a central computer along with the antenna adress. The computer then writes the received data to a text file. The structure of these data files is explained in detail in the next section.

4.1.1. Data File Format

The data files are simple text files, where each line denotes an *event* registered by an antenna in the system. Every day the data file is closed, saved, and a new data file is created automatically by the system.

Basically there are to types of events. The system output for first type is shown in figure 7 on page 15. It shows the line of text written to the data file when the Radio-frequency

[5] Polyvinyl chloride is a widely used thermoplastic polymer [17]

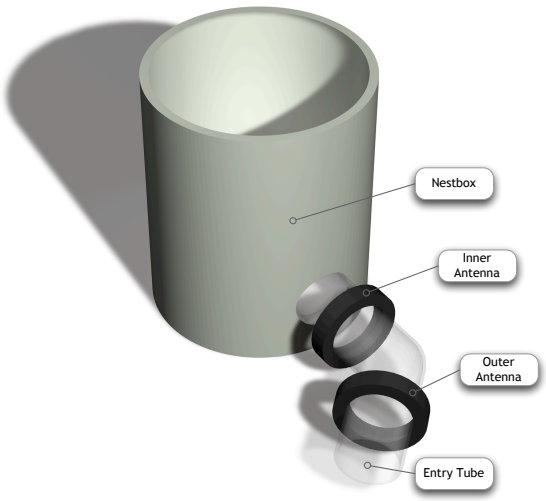


Figure 6: 3d-Model of an artificial nestbox with the two antennas wrapped around the entry tube.

identification transponder has been identified by an antenna. The id of the transponder is a unique, ten character wide, alphanumeric value.

If a transponder enters or leaves the coverage area of an antenna, the second event type occurs. The ouput is shown in figure 8 on page 15.

The following lines show the data written to the data file by the system, when a mouse passes the two antennas attached to a box. The events taking place when each of these lines is written to the data file is explained in the following list. The list numbers match the line numbers of the data file fragment.

```
1 2008-10-01 16:31:08:499;   111;   0;
2 2008-10-01 16:31:09:095;   113;   0;
3 2008-10-01 16:31:42:512;   111;   5;  00 06 B8 D4 5A
4 2008-10-01 16:31:42:807;   113;   5;  00 06 B8 D4 5A
5 2008-10-01 16:31:43:619;   111;   0;
6 2008-10-01 16:31:44:014;   113;   0;
```

1. The transponder **enters coverage area** of the antenna with address `111`.
2. The transponder **enters coverage area** of the antenna with address `113`.
3. The transponder is **identified** as `00 06 B8 D4 5A` at antenna with address `111`.
4. The transponder is **identified** as `00 06 B8 D4 5A` at antenna with address `113`.
5. The transponder **leaves coverage area** of the antenna with address `111`.

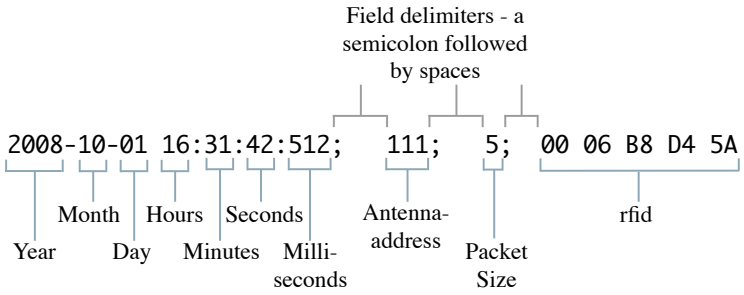


Figure 7: Typical dataset in a data file with and Radio-frequency identification transponder identification

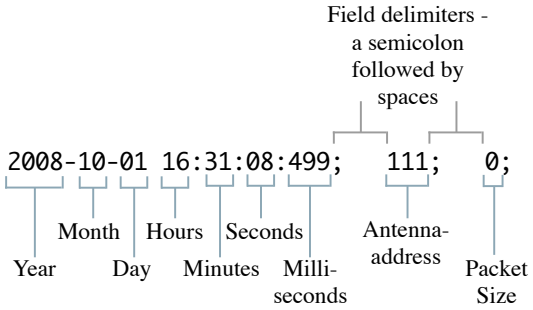


Figure 8: Typical dataset in a data file without an Radio-frequency identification transponder identification

6. The transponder **leaves coverage area** of the antenna with address `113`.

Depending on the event type, the value of the packet size is either a `0` for events without a transponder identification value, or a `5` if the transponder has been identified. For the data processing (see section 5 on page 27) only the datasets with a transponder identification are taken into account.

4.1.2. Antenna adressing

There are always two antennas attached to an entry tube of a box (see figure 6 on page 14). The antenna address is three digits long, composed by the box number it is attached to (first two digits), and the position at the entry tube of the box. Outer antennas have a `1` as the last digit of the address, innner antennas a `3` (e.g the antennas attached to box 11 are addressed 111 for the outer, 113 for the inner antenna, respectively). Needless to say, that box numbers have to be unique.

Unfortunately there are a few exceptions to that schema, as for a few antennas, the correct addressing failed due to technical problems (cp. 4.1.4 at page 16).

4.1.3. General system design and functionality

- Can-Bus
- cable loop
- rfid identification how
- What happens in the boxes (boards) next to the antennas
- How is can bus working/implemented
- Where are the different cable loops (which antennas connected to a loop)
- General system layout (cabeling, protocols)
- Programmed software
- etc.

4.1.4. Problems

Unfortunately the problems with the system are diverse. Antenna drop outs, broken laptop or cable connections chewed by the mice, to name just a few. Some of these problems could be solved, other still remain.

However, one of biggest problem is, that the readings at the antennas are not as reliable as planned. This is clearly visible in the following numbers:

From the 5 of April 2007 to the 27 of March 2009, xxx datasets (table `data`) have been recorded. Assuming that the system works perfect, we would get xxx *direction results*. Compared to the real number of *direction results*, which is xxx, we have a yield of xx%.

In the next step, two *direction results* should create a *stay result*. The script which searches for the *stay results* could find xxx results in the xxx *direction results*, which is xx%. Most likely this low numbers are an effect of missed antenna readings. Normally, an antenna reads out the transponder information 10 times, to ensure maximum

Another problem are the transponders which have never been injected in a mouse, but are recorded by the system.

```
2007-11-25 10:09:12:906;   381;   5;   00 06 B8 E2 70
2007-11-25 10:09:16:529;   381;   5;   00 05 B8 D2 70
2007-11-25 10:09:16:932;   381;   5;   00 06 B8 E2 70
2007-11-25 10:09:19:950;   381;   0;
2007-11-25 10:09:20:253;   381;   5;   00 06 B8 E2 70
```

This clipping of a data file shows an Radio-frequency identification transponder which appears as `00 05 B8 D2 70` on line two but as `0 06 B8 E2 70` on the other data lines. The `rfid` table contains xxx different Radio-frequency identification's whereof xxx could not be identified as an injected transponder.

Furthermore, for some antennas the address could not be set properly. In one case two antennas even sent the same adress, so that no distinction of the data could be made. The workaround for this problem is, two find an unused address which we are able to set and map it to the desired address during the data import.

4.2. Collecting Data Attributes

During the *population checks*, the whole shed, including all artificial nestboxes, plastic pipes and other structuring elements, is checked to screen the mice population. During these so called population and nest checks the data is collected using the following procedure.

1. Each box is opened and the following observations made:
   - **Mice in the box:** If adult mice are in the box, they get identified by their transponder, measured and weighed. Furthermore it will be checked, if the sex of the mouse has already been determined.
   - **Bedding:** Determine if the nestbox shows marks of usage, like trampled bedding.
   - **Pups:** Does the box contain pups, and if yes how many and whats their estimated age.
   - **Communal nest:** Check if two or more litters are reared in the box.
2. All the possible shelters, like planks an pipes, are checked for mice presence as well.

4.3. Data storage

All the data is stored in a MySQL[6] database, which is a Relational Database Management System[7] (RDMS). The following figure (fig. 22) shows all the tables including the data types[8] of the columns and their relations to columns in other tables.

A schema of the database including all the tables and the references between the table columns can be found in the Appendix on page 47.

The tables can be divided into different groups by looking at the type of data they hold. The following sections explain these groups and the associated tables in short.

4.3.1. Processed data

This group comprehends tables with data based on the spatial position data collected by the system in the shed (see section 4.1 on page 12 for details).

The tables belonging to that group are outlined in figure 9 on page 19.

data table A perl[9] script imports the data files written by the system in the shed into the `data` table.

Shown next is a row of the `data` table followed by short explanation of the columns.

```
(first part of table row)
+---------+--------------+---------------------+----------+
| id      | rfid         | time                | millisec |
+---------+--------------+---------------------+----------+
| 7321019 | 00069B4D4D | 2009-02-07 00:51:56 |      173 |
+---------+--------------+---------------------+----------+

(second part of table row)
+-----+-----------------+------+---------+---------+
| ant | import          | i    | dir_id  | res_id  |
+-----+-----------------+------+---------+---------+
| 121 | 09-0206155809 |    4 |   40102 |   10001 |
+-----+-----------------+------+---------+---------+
```

`id` is an auto increment integer and therefore a unique identifier of a dataset in this table.

`rfid` is a transponder value. This value is a reference to a value in the `id` column of the `rfid` table (see paragraph 4.3.2 on page 22).

`time` and `millsec` columns harbor the time the dataset has been recorded. Unfortunately the *MySQL* `DATETIME` data type does not include the milliseconds. Hence, these values have to be stored in a seperate column.

[6] MySQL database

[7] A Relational Database Management System is a Database Management System in which data is stored in the form of tables and the relationship among the data is also stored in the form of tables [18].

[8] An overview of the MySQL data types can be found on the following webpage: http://dev.mysql.com/doc/refman/5.0/en/data-types.html

[9] Perl is a high-level, general-purpose, interpreted, dynamic programming language. [16]

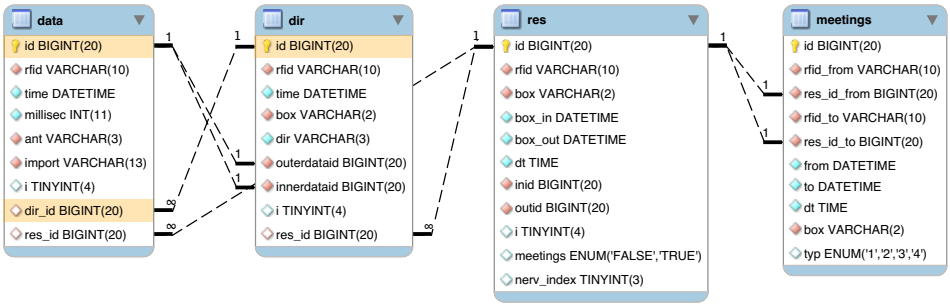


Figure 9: Schemata of the data tables with the processed spatial position data and the relations between them.

`ant` denotes the antenna the data was recorded at. This value is a reference to a value in the `id` column of the `ant` table (see paragraph 4.3.2 on page 24).

`import` is a reference to a dataset in the `logs` table, simply unveils from which data file this dataset is part of.

`i` is an indicator if the dataset could is used in a direction result (see 4.3.1) and stay result (see 4.3.1). Table 1 on page 48 gives an overview of the `i` values in the different tables.

`dir_id` is a reference to an `id` in the `dir` table, if that dataset could be used in a *direction result*. Else, this value is `NULL`.

`res_id` is a reference to an `id` in the `res` table, if that dataset could be used in *stay result*. Else this value is `NULL`.

dir table Another perl script searches for matching pairs of datasets in the `data` table which form a *direction result*.

When a mouse carrying an Radio-frequency identification transponder passes the two antennas attached to an artificial nestbox, within a certain time, it is possible to determine if the mouse went in or out of that box (see section 5.2 on page 28 for details about the script).

Shown next is a row of the `dir` table followed by short explanation of the columns.

```
(first part of table row)
+--------+-------------+---------------------+-----+-----+
| id     | rfid        | time                | box | dir |
+--------+-------------+---------------------+-----+-----+
| 40102  | 00069B4D4D  | 2009-02-07 00:51:56 | 12  | in  |
+--------+-------------+---------------------+-----+-----+

(second part of table row)
+--------------+--------------+------+----------+
| outerdataid  | innerdataid  | i    | res_id   |
+--------------+--------------+------+----------+
|      7321019 |      7321020 |    4 |    10001 |
+--------------+--------------+------+----------+
```

`id` and `rfid` have the identical function as in the `data` table.

`time` denotes the moment the the `rfid` entered or left the `box`.

`box` is a reference to a value in the `id` column of the `box` table (see paragraph 4.3.2 on page 23).

`dir` is either *in* or *out* and depicts the direction.

`outerdataid` and `innerdataid` values can be used to backtrack the datasets in the `data` table making up the *direction result*. This is explained in detail in section 5.2 on page 28.

`i` is an indicator if the direction result could is used in a (see 4.3.1). Table 1 on page 48 gives an overview of the `i` values in the different tables.

`res_id` is a reference to an `id` in the `res` table, if that dataset could be used in *stay result*. Else this value is `NULL` (= empty).

res table When two *direction results*, one with a `dir` value of `in` and the other with a `dir` value of `out`, as well as matching `rfid` and `box` values are found, they form a so called *stay result* (see section 5.3 on page 29 for details).

Shown next is a row of the `res` table followed by short explanation of the columns.

```
(first part of table row)
+-------+-------------+-----+---------------------+---------------------+----------+
| id    | rfid        | box | box_in              | box_out             | dt       |
+-------+-------------+-----+---------------------+---------------------+----------+
| 10001 | 00069B4D4D  | 12  | 2009-02-07 00:51:56 | 2009-02-07 00:52:01 | 00:00:05 |
+-------+-------------+-----+---------------------+---------------------+----------+

(second part of table row)
+----------+-------+---------+---+-----------+------------+
| dt       | inid  | outid   | i | meetings  | nerv_index |
+----------+-------+---------+---+-----------+------------+
| 00:00:05 | 40102 | 7321021 | 4 | TRUE      |          1 |
+----------+-------+---------+---+-----------+------------+
```

`id`, `rfid` and `box` were already explained for the previous tables, and have the same function in this table.

`box_in` and `box_out` denote the beginning and the end time of the sojourn of the `rfid` in a `box`.

`inid` and `outid` can be used to backtrace the datasets in the `dir` or `data` table making up the *stay result*. This is explained in detail in section 5.3 on page 29.

`i` is either `3` or `4` and indicates the type of the result. This distinction is explained in detail in section 5.3 on page 29.

`meetings` is set to `true` when the result set has been analyzed for meetings.

`nerv_index` gives an idea about how *nervous* the mouse was when she stayed in the box. The *nervousity* is explained in detail in section 5.3 on page 29.

meetings table In this work, an event is termed a *meeting*, if *stay results* of different mice in the same box show a temporal overlap. These meeting data are written to the `meeting` table (see section 5.4 on page 32 for details).

Shown next is a row of the `meetings` table followed by short explanation of the columns.

```
(first part of table row)
+--------+-------------+--------------+-------------+------------+
| id     | rfid_from   | res_id_from  | rfid_to     | res_id_to  |
+--------+-------------+--------------+-------------+------------+
| 302635 | 0006CD478F  |       596630 | 0006CC7962  |     596100 |
+--------+-------------+--------------+-------------+------------+

(second part of table row)
+---------------------+---------------------+----------+-----+-----+
| from                | to                  | dt       | box | typ |
+---------------------+---------------------+----------+-----+-----+
| 2009-03-27 16:58:36 | 2009-03-27 17:01:00 | 00:02:24 | 13  | 1   |
+---------------------+---------------------+----------+-----+-----+
```

`id` is the unique identifier of a dataset.

`rfid_from` and `rfid_to` denote the two participating mice (Radio-frequency identification transponders) of the *meeting result*.

`res_id_from` points to the `id` of the *stay result* from the `rfid_from`, and the `res_id_to` to the one of the `rfid_to`. This allows to look up the the *stay results* making up this `meeting result`.

`from`, `to` and `dt` declare the start, the end time and the duration of the meeting.

`box` contains the reference to an `id` in the `box` table (see 4.3.2).

The different values of the `typ` column are explained in detail in section 5.4 on page 32.

4.3.2. System members data

This group encloses the tables wich contain data about the different *members* of the data collection system in the shed.

The tables included in that group are outlined in figure 10 on page 22.

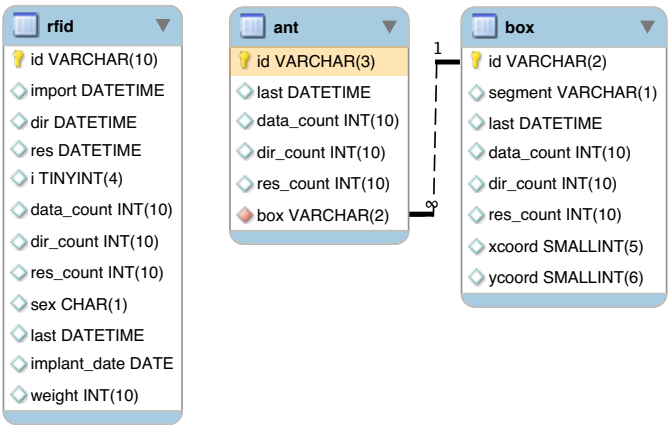


Figure 10: Schemata of the database tables containing the *system members* data.

rfid table The `rfid` table contains all the different transponder id's recorded by the system, along with the corresponding attribute data and some summarized information about each Radio-frequency identification.

Shown next is a row of the `rfid` table followed by short explanation of the columns.

```
(first part of table row)
+--------------+---+------------+-----------+-----------+
| id           | i | data_count | dir_count | res_count |
+--------------+---+------------+-----------+-----------+
| 0006955EED   | 3 |          1 |         0 |         0 |
+--------------+---+------------+-----------+-----------+

(second part of table row)
+-----+---------------------+--------------+--------+
| sex | last                | implant_date | weight |
+-----+---------------------+--------------+--------+
| m   | 2007-08-29 23:35:41 | 2007-08-28   |   18.0 |
+-----+---------------------+--------------+--------+
```

`id` is the unique, 10 character wide alphanumeric transpnder identification.

`i` is is a helper value used in the data import procedure. It is `0` if new data for the transponder has been imported, `2` if the data is searched for *direction results* and `3` if the data has been searched for results.

`data_count` contains the sum of datasets in the `data` table.

`dir_count` contains the sum of datasets in the `direction` table.

`res_count` contains the sum of datasets in the `results` table.

`sex` column holds the gender of the mouse.

`last` points to the maximum value in the `time` column of the `data` table for this mouse.

`implant_date` holds the date of the transponder injection.

`weight` contains a decimal number with the weight of the mouse in grams.

box table The box table contains all the information about the 40 artificial nestboxes. Shown next is a row of the `box` table followed by short explanation of the columns.

```
(first part of table row)
+----+---------+---------------------+------------+
| id | segment | last                | data_count |
+----+---------+---------------------+------------+
| 01 | A       | 2009-03-27 16:38:59 |     122439 |
+----+---------+---------------------+------------+

(second part of table row)
+-----------+-----------+--------+--------+
| dir_count | res_count | xcoord | ycoord |
+-----------+-----------+--------+--------+
|     44389 |      9824 |    246 |    708 |
+-----------+-----------+--------+--------+
```

`id` is the unique 2 character wide box identification.

`segment` point to the segment (A, B, C or D) the box is located in.

The meaning of the `last` `data_count`, `dir_count` and `res_count` columns is already explained in the `rfid` table.

`xcoord` and `ycoord` contain the x (entrance side wall) and y (left wall from the entrance wall) coordinates in centimeter of the box in the barn.

ant table The ant table contains all the information about the 80 antennas.
Shown next is a row of the `ant` table followed by short explanation of the columns.

```
+-----+---------------------+------------+-----------+-----------+-----+
| id  | last                | data_count | dir_count | res_count | box |
+-----+---------------------+------------+-----------+-----------+-----+
| 011 | 2009-03-27 16:38:59 |      54220 |     46099 |     19648 | 01  |
+-----+---------------------+------------+-----------+-----------+-----+
```

`id` is the unique 3 character wide antenna identification.
The meaning of the `last` `data_count`, `dir_count` and `res_count` columns is already explained in the `rfid` table.
`box` points to the `id` value in the `box` table, the antenna is attached to.

4.3.3. Auxiliary tables

This group includes tables that contain percalculated data for faster data browsing with the *miceminer* application and a table with the imported data files.

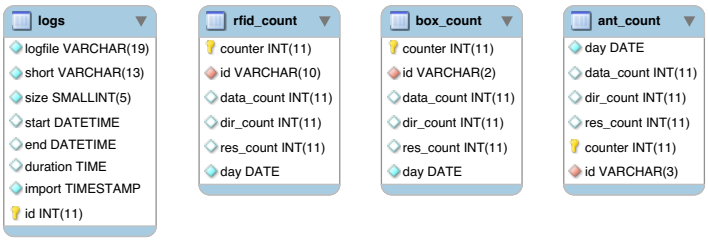


Figure 11: Schemata of the database tables with the auxiliary data. The relations are not shown.

logs table The `logs` table contains information about the imported data files.
Shown next is a row of the `logs` table followed by short explanation of the columns.

```
(first part of table row)
+-----+---------------------+---------------+------+---------------------+
| id  | logfile             | short         | size | start               |
+-----+---------------------+---------------+------+---------------------+
| 608 | 20090326_170750.txt | 09-0326170750 | 1660 | 2009-03-26 17:07:51 |
+-----+---------------------+---------------+------+---------------------+
```

```
(second part of table row)
+---------------------+----------+---------------------+
| end                 | duration | import              |
+---------------------+----------+---------------------+
| 2009-03-27 17:07:40 | 23:59:49 | 2009-04-12 01:03:01 |
+---------------------+----------+---------------------+
```

`id` is the unique identifier of dataset.

The original filename of the data file is stored in the `logfile` column.

`short` values are a kind of code made up from parts of the data filename. This column was important at the beginning of the project, when we had to import data files with a different structure.

`size` contains the filesize in bytes of the data file.

`start` and `end` indicate the first an and last point in time of an antenna reading, this data file contains.

`duration` is simply the period between the `start` and `end` values.

`import` refers to the time the import of the data in the file into the `data` table was finished.

rfid_count, box_count, ant_count These three tables contain per day data counts for transponders (`rfid_count`), boxes (`box_count`) and antennas (`ant_count`), and are exclusively needed by the *miceminer* application to allow faster browsing of the data.

Shown next is a row of the `rfid_count`, `box_count`, `ant_count` tables followed by short explanation of the columns.

```
(row of table rfid_count)
+---------+------------+------------+------------+-----------+-----------+
| counter | id         | day        | data_count | dir_count | res_count |
+---------+------------+------------+------------+-----------+-----------+
|     999 | 0006B9C5E8 | 2008-07-14 |         10 |         4 |         2 |
+---------+------------+------------+------------+-----------+-----------+


(row of table box_count)
+---------+----+------------+------------+-----------+-----------+
| counter | id | day        | data_count | dir_count | res_count |
+---------+----+------------+------------+-----------+-----------+
|     999 | 10 | 2008-07-31 |        259 |       114 |        39 |
+---------+----+------------+------------+-----------+-----------+


(row of table ant_count)
+---------+-----+------------+------------+-----------+-----------+
| counter | id  | day        | data_count | dir_count | res_count |
+---------+-----+------------+------------+-----------+-----------+
|     999 | 121 | 2008-07-19 |        196 |       174 |        47 |
+---------+-----+------------+------------+-----------+-----------+
```

`counter` is the unique identifier of dataset.

`id` contains the identification of the transponder, the box or the antenna.

`day` points to the day the table row holds the data counts for.

`data_count`, `dir_count` and `res_count` contain the per day countes for the datasets, the *direction results* and the *stay results*.

4.4. Data access

Most of the mainly used relational databases offer Structured Query Language[10] (SQL) to manage and retrive the data. However, SQL is not easy to learn and use, lacks built-in export functionality of the data to file formats such as $Microsoft Excel^{©}$, and does not offer any possibility to visualize the data. Therefore, a feature rich but still handy and intuitive, GUI[11] (GUI) is available, making it easy for the user to explore, visualize and export data (see section 6.1 on page 34).

[10] The Structured Query Language is a database computer language designed for the retrieval and management of data in relational database management systems. [21]

[11] A graphical user interface is a type of user interface which allows people to interact with electronic devices such as computers [...] with images rather than text commands [14]

5. Data Processing

Figure 12 shows an overview of the different parts involved in the data importing and analysis process. The perl scripts which are reading data from and writing data to the database build the main part of this framework[12].

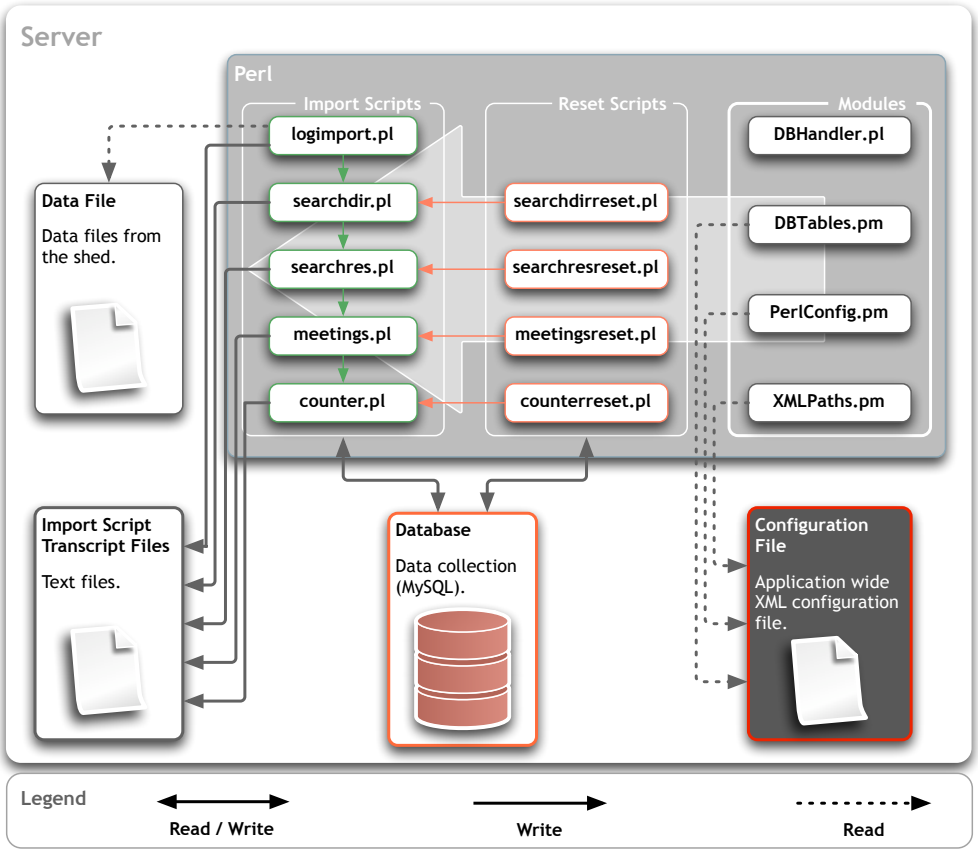


Figure 12: Overview of the parts involved in the data processing.

The *Import Scripts* run in a cascade, beginning with the `logimport.pl` script, which is reading the data files from the shed, and ending with the `counter.pl` script (indicated in figure 12 by the green arrows). Each of these scrips writes a transcript file to make their progress traceable. Upon completion of the cascade, the data files are imported and stored in

[12]The directories in which the scripts, modules and files are located, are listed in table 4 on page 55.

the respective tables as outline in section 4.3 on page 18.

A scheduled, daily task checks for new data files uploaded to the server and starts the cascade if needed[13].

The *Reset Scripts* set the database back to a state from where the import cascade can be rerun, starting with the corresponding import script (indicated in figure 12 by the orange arrows). If, for example, the `searchdirreset.pl` has been executed, the import cascade can be started with the `searchdir.pl` script[14].

The *Modules* read out the configuration and make it available for the import and reset scripts (indicated in figure 12 by the large grey arrow with the white border). The whole configuration for the application is stored in an XML file. The part of configuration file accountable for the perl part is shown in appendix B.3.2 on page 57. As the perl scripts are heavily interacting with the database, the database configuration part of the configuration files is read as well by some of the *Modules*[15].

5.1. Importing

The `logimport.pl` extracts the spatial position data from the data files and writes it to the `data` table [16].

Prior to the data extraction, the script creates a backup of the actual database[17]. Subsequently the data is extracted from the data files and written to the database.

As mentioned in section 4.1.4 on page 16, some of the antennas could not be addressed properly. Therefore, the script has to catch these exceptional values and convert them to the designated ones[18].

5.2. Direction Results

The `searchdir.pl` script searches for matching pairs of antenna readings in the `data` table, which build a *direction result*. A pair matches if,

- the transponder identification values of the two readings is the same,
- time difference in seconds between the two readings is not greater then the value set as the `<antennaInterval>` value [19] in the configuration,
- and the readings originate from the two different antennas attached to a box.

Figure 13 shows the composition of a direction *in* and a direction *out* result by means of an example at nestbox 16. The `<antennaInterval>` value is set to 5 seconds.

[13] For details about this topic see appendix B.2 on page 56.
[14] It is recommended to start the import cascade by using the import BASH scripts. This is explained in appendix B.2 on page 56.
[15] The database configuration is shown and explained in appendix B.3.1 on page 57.
[16] The `data` table structure is descriped section 4.3.1 on page 18.
[17] The configuration for the backup is explained in appendix B.3.2 on page 57.
[18] See listing A.1.1 on page 44 for the detailed conversions.
[19] See appendix B.3.2 on page 57.

Figure 13: Illustration of the possible composition of a *direction result*.

This is exactly the order this script searches for *direction results*. The findings are written to the `dir` table[20]. The value for the `time` field of the *direction result* is always taken from the antenna reading at the outer antenna. Therefore, a transpondered mouse is out of a box, when she passed the outer antenna.

If the `<antennaInterval>` has been changed, the `searchdirreset.pl` needs to be executed followed by the scripts in the import cascade starting with the `searchdir.pl` script [14].

5.3. Stay Results

The `searchres.pl` aims to find matching *direction results* in the `dir` table, which build a *stay results*.

A pair matches if,

- the transponder identification values of the two *direction results* are the same
- the `direction` values of the *direction results* are oppositional
- and the *direction results* are recorded at the same box.

or,

- the transponder identification values of a *direction result* with a `direction` value of `in` and an unused (=could not be used for a *direction result*) dataset from the `data` table recorded at an outer antenna, are the same
- and belong to the same box.

In the former case, the *stay result* is written to the `res` table as a **type 3** result in the latter case as a **type 4** result [21]. Figures 14 and 15 illustrate the differences between the types by means of an example of at box 16. The `<antennaInterval>` value is set to 5 seconds.

Additionaly, the script has to handle the temporal overlaps of stay results. Since, the antenna system is not working perfect (see section 4.1.4), if we let the script keep all possible *stay results*, we get situations as illustrated in figures 16 and 17. All the *stay results* (1, 2 and

[20] The `dir` table structure described in sction 4.3.1 on page 20
[21] See section 4.3.1 on page 21 for details about the `res` table.

Figure 14: Illustration of a Type 3 *stay result* at box 16, made up by a direction `in` and a direction `out` result.

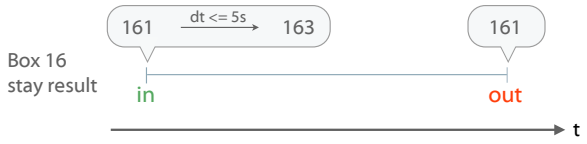


Figure 15: Illustration of a Type 4 *stay result* made up by a direction `in` result and a reading at the outer antenna.

3) originate from the same transponder recorded at the same or different boxes. Hence, the antenna system must have missed readings, but we have no clue about the exact moment.

In figure 16 we have the situation, that one *stay result* (1) overlaps two others (2),(3). A reasonable explanation would be, that the antenna system missed a recordings between the *in* moments of *stay results* (1) and between the *out* moments of *stay results* (3) and (1). Since we have two *stay results*, (2) and (3), which are looking valid, we discard *stay result* (1), where we don't know what really happened. Furthermore, this approach is reasonable, as the probability of some, maximum two, missed antenna readings is much higher than the probabilty that eight antenna readings building two perfect looking *stay results* are wrong.

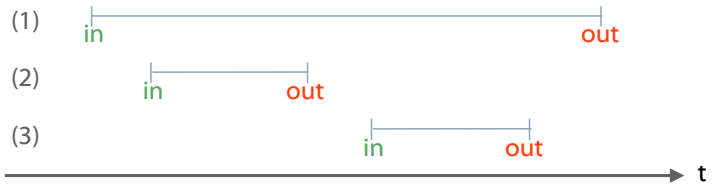


Figure 16: Illustration of a possible result overlap situation, where a *stay result* overlaps many others.

Figures 17 and 18 illustrate the situation where one *stay result* overlaps another. In this case, the *stay result* (1) is discarded as well.

The justification for this approach is pretty similar to the one just discussed. *Stay result* (2) looks valid, but there must be missed antenna readings between the time of *stay result* (1) and *stay result* (2) *in*, and between *stay result* (2) and *stay result* (1) respectively. In figure 18 the two *stay results* are only partly overlapping. *Stay result* (1) is discarded as well, but *stay result* (2) will be discared in the next step.

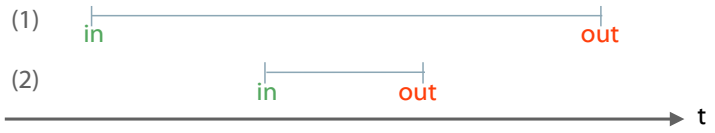


Figure 17: Illustration of a possible result overlap situation, where a *stay result* fully overlaps another.

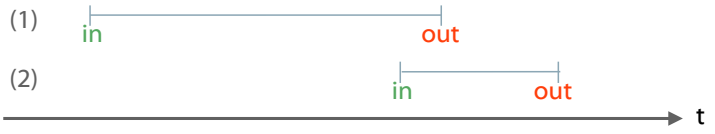


Figure 18: Illustration of a possible result overlap situation, where a *stay result* partly overlaps another.

Now that the *stay result* is checked for overlaps with other *stay results*, the script checks for other antenna readings that were recorded during the *stay result* as illustrated by an example in figure 19. Obviously, only antenna readings originating from the same transponder are taken into account.

In figure 19, four other antenna readings could be found, two of them not even from the same box. This *stay result* will be discarded, as we don't know what really happened.

As seen in figure 19, the reading at antenna `163`, which is the inner antenna attached to nest-box 16, is colored green. If we have a look at the situation illustrated in figure 20, we can see that several datasets which are exclusively recorded at the innner antenna are overlapped. In such a case, we keep the result, as the transpondered mouse has not left the box[22], and denote

[22] As stated in section 5.2, a transpondered mouse is out af the box if she is recorded at the outer antenna.

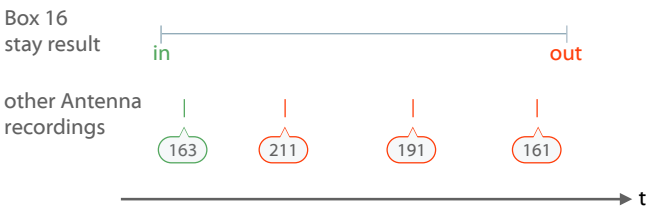


Figure 19: Illustration of a possible dataset overlap situation, where other antenna readings have been found during a*stay result*.

it a *nervous mouse*, as we think, that she is only checking the entry tube out of nervousness.

The *nervous index* is simply the number of readings at the inner antenna during the *stay result*, which is 4 for the situation shown in figure 20. This value is written to the database as explained in section 4.3.1.

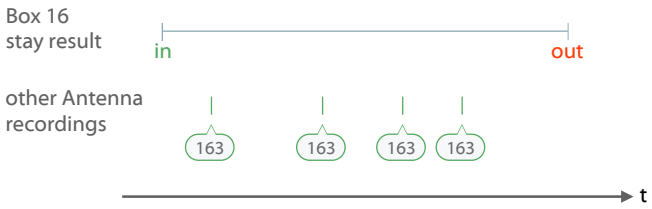


Figure 20: Illustration of a possible dataset overlap situation, where other antenna readings have been found during a *stay result*, but only at the inner antenna of the box. Such a situation is termed a *nervous mouse*.

The search for the *stay result* overlaps and the overlapped antenna recordings could be combined. Though, we are interested in the influence of these steps on the numbers.

5.4. Meeting Results

The `meetings.pl` script determines, when and how many time the transpondered mice spend together in the nestboxes. The *stay results* make the basis. A *meeting result* is found, whenever

- two *stay results*,
- of two different transpondered mice,
- which take place in the same nestbox,

- temporally overlap.

6. Accessing and exploring the Data

Data exploration to scan data for interesting aspects.

6.1. Introducing miceminer

- technology
- installation
- usability
- limitations

6.1.1. Data Filtering

Shortly explain the filtering possibilities of the program.

6.1.2. Data Visualization

Shortly explain the visualization possibilities of the program.

6.1.3. Exporting Data

Shortly explain the export function of the program.

6.2. Data analysis with miceminer

Miceminer provides some simple data analysis functionality.

6.2.1. Home Range Data

Functionality of the home range analysis.

Minimal polygon calculation for boxes area. Applying threshold which boxes to take into account calculation.

Exporting dbf files which can be imported into ARCGIS.

Applications Above method is outdated, nowadays Kernel Density Estimation is used -> Anna

6.2.2. Shared Preferences

Is there a preference between to female mice. Calculate if two females meet more often than expected, which lets us draw conclusions about a their relationship.

Applications Prof. Koenig ?

6.2.3. Monthly Box

A matrix showing which mice has been seen in which boxes over the selected time range.

Applications Prof. Koenig ?

6.2.4. Monthly Antennas

A matrix showing which mice has been seen at which antennas over the selected time range.

Applications Prof. Koenig ?

7. Exploring Social Network

7.1. Introduction to Social Networks

Short introduction to network theory.

- Graph (weighted/unweighted, directed/undirected)
- Node
- Vertex

7.2. Advantages of the Networks approach

Understanding the network structure gives us better understanding of the role of the individual in the population.

7.3. Relational Data For the Network

Preparation of the data.

- Period one month
- The monthly association duration sum must be more than one hour (3600 seconds)
- The count of monthly association must be higher than one

7.4. Visual Exploration

7.4.1. introduction

- Visual pattern detection
- Network topology

7.4.2. Visual exploration with miceminer

- Weighted Graph (Edge labeling)
- Attribute highlighting
- Filtering Edges
- Ego network
- Walking the graph
- Exporting data for Netdraw

7.4.3. Visual exploration with Netdraw

Will see how I use netdraw ...

7.5. Analysis of the Network Over the Time

7.5.1. introduction

At each discrete time step (month), the network is quantified by the following measurements, tests.

7.5.2. Node Based Measures

- Network edge density
- Path length
- Clustering coefficient
- Degree
- Node betweenness

7.5.3. Statistical Tests for Node Based Measures (p. 88 ff.)

Null Hypothesis is randomized network data. (Monte Carlo)

- mean degree
- mean clustering coefficient
- Category measures (Mann-Whitney test p.110)

7.5.4. Substructures

Evidence of segregation by discrete categories (p.117)

- Association pattern (Individuals are more likely to interact with individuals of similar type) (p .119)
- Degree correlation (Nodes with high degree connect to other nodes with high degree ?) (p.123)
- Community structure (p.124)

7.5.5. Network Analysis with miceminer

Only Node Based measures.
Carry out Simplified Longitudinal Network Analysis with Promising Individuals.

7.5.6. Network Analysis with UCINET/Netdraw

Need to see what is possible within reasonable time.

7.5.7. Results

How did the values (e.g. node based measures) evolve or change over the time.

Promising Individuals Following individuals over time and see how their node based measures or place within the network change.

8. Conclusions

... will see.

9. References

[1] P.K. Anderson. The role of breeding structure in evolutionary processes of Mus musculus populations. *Proc. Symp. Mutational Process*, pages 17–21, 1965.

[2] J. Riester & H. Markl B. König. Maternal care in house mice (Mus musculus): Ii. The energy cost of lactation as a function of litter size. *Journal of Zoology*, 216:195–210, 1988.

[3] A.E.M Baker. Gene flow in house mice: behavior in a population cage. *Behav Ecol Sociobiol*, 8:83–90, 1981.

[4] R.J. Berry. *Biology of the House Mouse*. Academic Press, 1981.

[5] F.H. Bronson. The reproductive ecology of the house mouse. *The Quarterly Review of Biology*, 54:265–299, 1979.

[6] F.H. Bronson. The adaptability of the house mouse. *Scientific American*, 250:90–97, 1984.

[7] K. A. Hammond & J. Diamond. An experimental test for a ceiling on sustained metabolic rate in lactating mice. *Physiological Zoology*, 65:952–977, 1992.

[8] Olivia Dieser. Terretoriales Verhalten einer frei lebenden Hausmauspopulation. *Unpublished*, 2008.

[9] Wikipedia Editors. AFP— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/AFP.

[10] Wikipedia Editors. API— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/API.

[11] Wikipedia Editors. CGI— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/Common_Gateway_Interface.

[12] Wikipedia Editors. Database management system — wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/Database_management_system.

[13] Wikipedia Editors. Graph theory — wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/Graph_theory.

[14] Wikipedia Editors. GUI— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/GUI.

[15] Wikipedia Editors. Joule— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/Joule.

[16] Wikipedia Editors. Perl— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/Perl.

[17] Wikipedia Editors. PVC— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/PVC.

[18] Wikipedia Editors. Relational database management system — wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/RDBMS.

[19] Wikipedia Editors. RFID — wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/RFID.

[20] Wikipedia Editors. Samba— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/Samba_(software).

[21] Wikipedia Editors. SQL — wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/SQL.

[22] Wikipedia Editors. XML— wikipedia, the free encyclopedia, . URL http://en.wikipedia.org/wiki/Xml.

[23] Andrea Weidt et al. Not only mate choice matters: fitness consequences of social partner choice in female house mice. *Animal Behaviour*, 73(3):801–808, 2007.

[24] S. Fuchs. Consequences of premature weaning on the reproduction of mothers and offspring in laboratory mice. *Zeitschrift für Tierpsychologie*, 55:19–32, 1981.

[25] S. Fuchs. Optimality of parental investment: the influence of nursing on reproductive success of mother and female young house mice. *Behavioral Ecology and Sociobiology*, 10:39–51, 1982.

[26] M. Steg & L. Barnes J. Werboff. Communal nursing in mice: strain specific effects of multiple mothers on growth and behavior. *Psychon Sci*, 19:269–271, 1970.

[27] R.J Berry & M.E. Jakobson. Life and death in an island population of the house mouse. *Experimental Gerontology*, 6:187, 1971.

[28] R.J Berry & M.E. Jakobson. Life and death in an island population of the house mouse. *Journal of zoology*, 173:341, 1975.

[29] B. König. *Cooperation in Primates and Humans. Mechanisms and Evolution*. Springer-Verlag Berlin Heidelberg, 2006.

[30] B. König. Der Konflikt zwischen Anzahl und Qualität von Nachkommen aus evolutionsbiologischer Sicht. *Konstanzer Blätter für Hochschulfragen*, 25:34–51, 1987.

[31] W.Z. Lidicker. Social behaviour and density regulation in house mice living in large enclosures. *Journal of Animal Ecology*, 45:677–697, 1976.

[32] J.H. Mackintosh. Behaviour of the house mouse. *Symposia of the Zoological Society of London*, 47:337–365, 1981.

[33] B. König & H. Markl. Maternal care in house mice. the weaning strategy as a means of parental manipulation of offspring quality. *Behavioral Ecology and Sociobiology*, 20: 1–9, 1987.

[34] N. H. Westwood & A. H. Reisner P. R. Pennycuik, P. G. Johnston. Variation in Numbers in a House Mouse Population Housed in a Large Outdoor Enclosure: Seasonal Fluctuations. *Journal of Animal Ecology*, 55(1):371–391, 1986.

[35] J. Pelikan. Patterns of reproduction in the house mouse. *Symposia of the Zoological Society of LondonWalls, G. 1984. Scenic and allied reserves of the Symposia of the Zoological Society of LondonWalls, G. 1984. Scenic and allied reserves of the Symposia of the Zoological Society of LondonWalls, G. 1984. Scenic and allied reserves of the Symposia of the Zoological Society of LondonWalls, G. 1984. Scenic and allied reserves of the Symposia of the Zoological Society of LondonWalls, G. 1984. Scenic and allied reserves of the Symposia of the Zoological Society of London*, 47:205–229, 1981.

[36] J.D. Reimer & M.L. Petras. Breeding structure of the house mouse in a population cage. *Journal of Mammalogy*, 48:88–99, 1967.

[37] M.L. Petras. Studies of natural populations of Mus. I. Biochemical polymorphisms and their bearing in breeding structure. *Evolution*, 21:259–274, 1967.

[38] M.X. Zarrow & V.H. Denenberg R. Gandelman, R.E. Paschke. Care of young under communal conditions in the house mouse (Mus musculus). *Developmental Psychobiology*, 3:245–250, 1970.

[39] P. Crowcroft & F.P. Rowe. Social organization and territorial behaviour in the wild house mouse. *Proc Zool Soc Lond*, 140:517–531, 1963.

[40] A. Sayler & M. Salmon. Communal nursing in mice: influence of multiple mothers on the growth of the young. *Science*, 164:1309–1310, 1969.

[41] RK Selander. Behavior and genetic variation in natural populations. *Integrative and Comparative Biology*, 10:53–66, 1970.

[42] C.H. Southwick. Regulatory mechanisms of house mouse populations: social behavior affecting litter survival. *Ecology*, 36:627–634, 1955.

A. Code

A.1. Data processing scripts

A.1.1. logimport.pl

For the full listing of the source code please visit http://gist.github.com/125712.

Listing 1: Antenna address conversions

```
405                 ##
406                 # Antennas beginning with 1A are mapped to box 16, 2A to box 17
407                 $ant =~ s/1A/16/;
408                 $ant =~ s/2A/17/;
409
410                 ##
411                 # Antennas 421 and 423 map to box 13 (Mail from B.Koenig 30.12.2008)
412                 $ant =~ s/421/131/;
413                 $ant =~ s/423/133/;
```

A.1.2. searchdir.pl

For the full listing of the source code please visit http://gist.github.com/125704.

```
0  #!/usr/bin/perl -w
1
2  ################################################################################
3  # Searches for dirction results in the raw data created by 'logimport.pl'
4  # - Step 2 of importing the data.
5  #
6  # Continues with 'searchres.pl' when finished.
7  #
8  # rleuthold@access.ch - 8.1.2009
9  ################################################################################
```

A.1.3. searchres.pl

For the full listing of the source code please visit http://gist.github.com/125705.

```
0  #!/usr/bin/perl -w
1
2  ################################################################################
3  # Searches for results - Step 3 of importing the data.
4  #
5  # Continues with 'counter.pl' when finished.
6  #
7  # rleuthold@access.ch - 8.1.2009
8  ################################################################################
```

A.1.4. meetings.pl

For the full listing of the source code please visit http://gist.github.com/125706.

```
0 #!/usr/bin/perl
1
2 ##############################################################################
3 # This scripts searches for meetings between two mice which is
4 # the time two mice spend in the boxes together.
5 #
6 # rleuthold@access.ch - 9.1.2009
7 ##############################################################################
8 use strict;
9 use warnings;
```

B. Tables & schemata

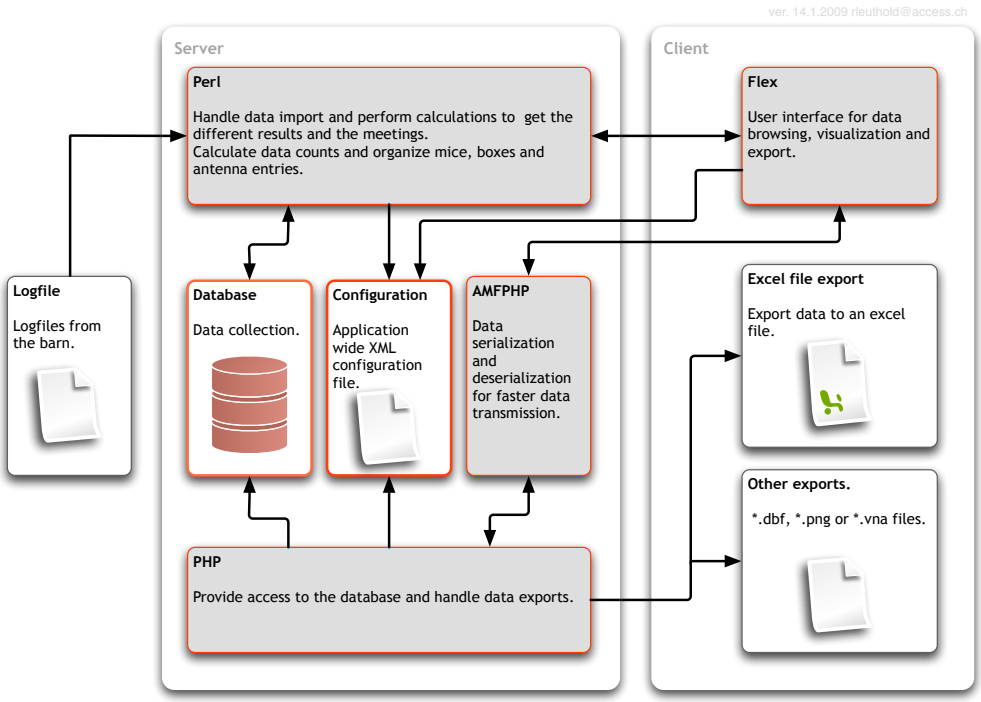


Figure 21: Overview of the application design.

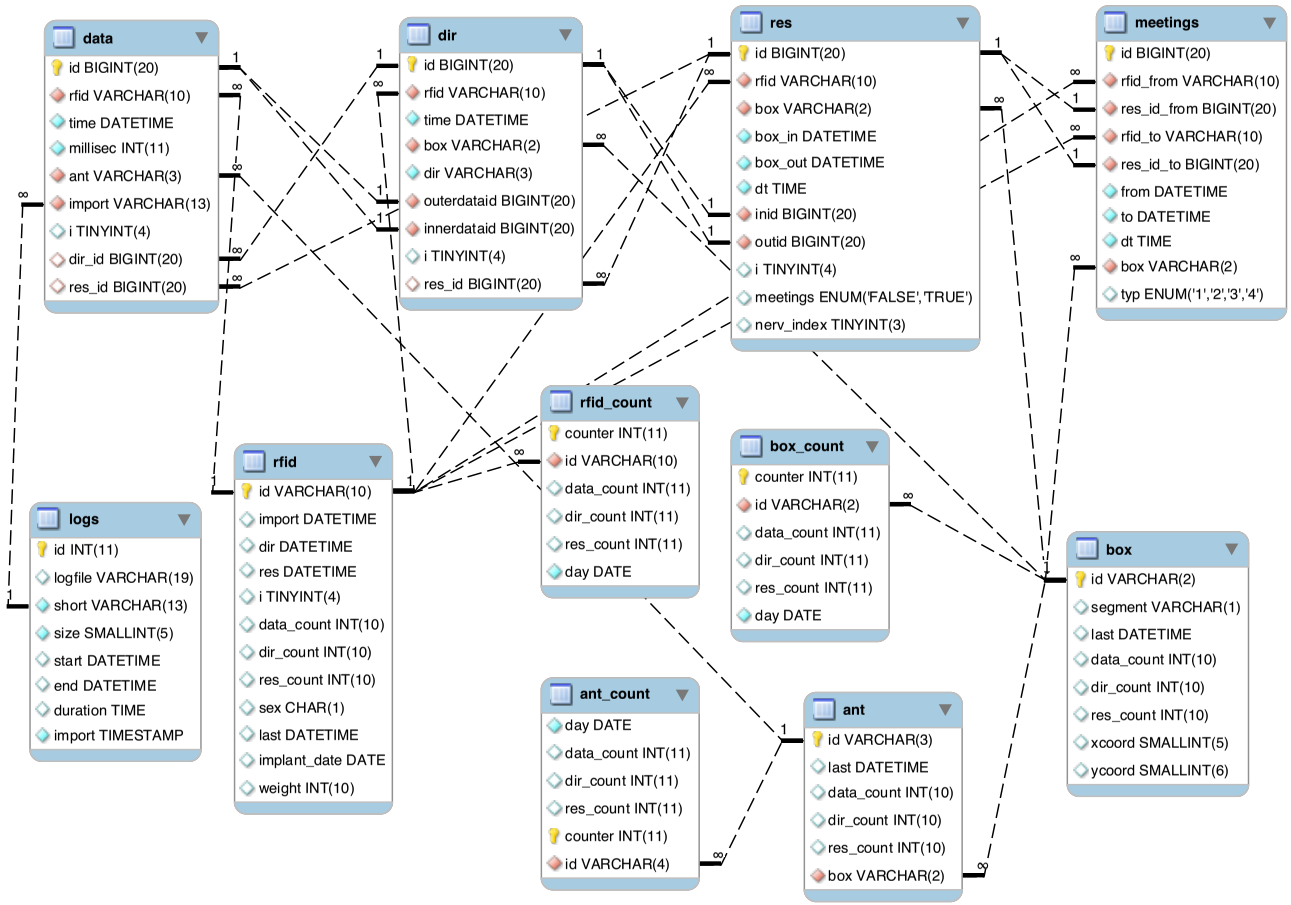


Figure 22: Database schema including all the tables and the relations between them.

Table 1: Meaning of the `i` values in the database tables.

	0	**1**	**2**	**3**	**4**
data	Dataset has not been analyzed yet.	Dataset has been analyzed but no matching dataset to form a *direction result* could be found.	Dataset is part of a *direction result* .	Dataset is part of a typ **3** *stay result*.	Dataset is part of a typ **4** *stay result*.
dir	*Direction result* has not been analyzed yet.	*direction result* has been analyzed but no matching *direction result* to form a *stay result* could be found.	-	*Direction result* is part of a typ **3** *stay result*.	*Direction result* is part of a typ **4** *stay result*.
res	-	-	-	*Stay result* is of typ **3**	*Stay result* is of typ **4**

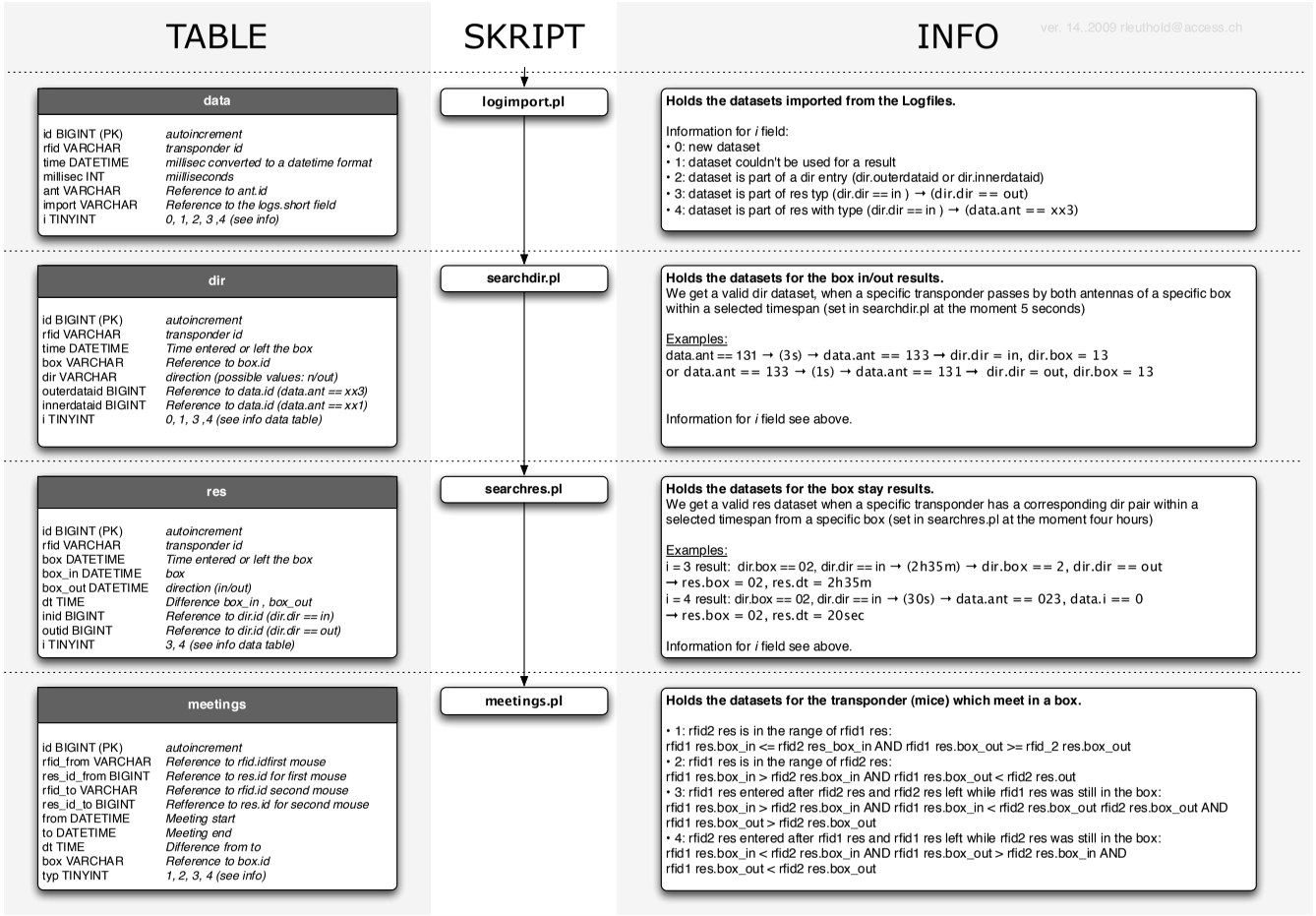


Figure 23: Cheatsheet for the data processing steps.

B.1. Server information

B.1.1. General information

Model	Dell Inspiron 530
Processor	Intel(R) Core(TM)2 Duo CPU E7200 @ 2.53GHz, 2525 MHz
Memory (RAM)	2 GB
Operating system	Linux version 2.6.27-7-server (gcc version 4.3.2 (Ubuntu 4.3.2-1ubuntu11))
IP-address	130.60.23.6
Webserver	Apache/2.2.9 (Ubuntu)
SSH	OpenSSH 5.1p1
PHP version	5.2.6-2ubuntu4
Perl version	5.10.0

Table 2: General information about the server

B.1.2. Web addresses

Please note that the server is only accessible from within the network of the *University of Zürich*.

Address	Description
http://zool-miceminer.uzh.ch	Main Site containing containing a lot of information about the project.
afp://zool-miceminer.uzh.ch	AFP[a] access to the server.
smb://zool-miceminer.uzh.ch	Samba[b] access to the server.
http://zool-miceminer.uzh.ch/phpmyadmin	Web-based MySQL database administration tool (phpMyAdmin).
http://zool-miceminer.uzh.ch/amfphp/browser/index.html	AMFPHP service browser.
http://zool-miceminer.uzh.ch/webalizer	Webserver traffic analyzer (Webalizer).
http://zool-miceminer.uzh.ch/miceminer/srcview/index.html	Source of the *miceminer* application.
http://zool-miceminer.uzh.ch/asdoc/index.html	*Adobe Flex* API[c] documentation of the *miceminer* application.
http://zool-miceminer.uzh.ch/conf/config.xml	XML-based application wide configuration file (special login required).
http://zool-miceminer.uzh.ch/conf/micedata_create.sql	The SQL which are used to create the database structure (special login required).
http://github.com/rico/miceminer-perl/tree/master	GIT[d] repository of the perl part of the application containing all the source code.
http://github.com/rico/miceminer-thesis/tree/master	GIT[e] repository of this document including all the LaTeX source files, schemata, etc.

Table 3: List of web addresses.

[a] The Apple Filing Protocol is a network protocol that offers file services for Mac OS X [9].
[b] Samba allows file and print sharing between computers running Windows and computers running Unix [20].
[c] An application programming interface is a set of routines, data structures, object classes and/or protocols provided by libraries and/or operating system services in order to support the building of applications [10].
[d] Git is a free distributed revision control, or software source code management project /citewiki:git.
[e] Git is a free distributed revision control, or software source code management project /citewiki:git.

B.1.3. Files & directories overview

Directory	Description
`/var/www/mouse`	**Base directory** of the whole project relevant files and folders.

All of the following files and directories are located within the base directory!

`conf/conf.xml`	The main XML[23] configuration file.
`import_scripts//`	Contains the BASH[24] scripts to start the import cascade at a specific script (see B.2 on page 56 for details).
`data/`	Uploaded but not yet imported data files.
`data/cronlogs/`	Transcripts created the perl scripts involved in the data import process.
`data/imported/`	After successful import of a data file into the database, the file is moved into this directory.
`miceminer/`	The *miceminer* application main files and folders.
`perl/cgi`	CGI[25] scripts written in perl which are used by the *miceminer* application.
`perl/extras/`	Perl scripts to control the data validity in the database and to compile data for some special analysis. The task of these scripts is usually explained in the header section of the file.
`perl/import/`	Contains the perl script involved in the data import and processing.
`perl/lib/`	The perl modules needed by all the import, reset and CGI scripts.
`perl/reset/`	Scripts to set back the data in the database, so that the import scripts can be rerun.
`amfphp/services/miceMiner/Mice.php`	AMFPHP services file.
`asdoc/`	Contains the *miceminer* API online documentation files.

[23] Extensible Markup Language is a general-purpose specification for creating custom markup languages [22].

Table 4: Overview of the important files and directories.

[24] Bourne-again shell is a free software Unix shell written for the GNU Project /citewiki:bash.

[25] The Common Gateway Interface is a standard protocol for interfacing external application software with an information server, commonly a web server [11].

B.2. Starting & scheduling import scripts

To start the import cascade, a set of BASH scripts is provided (see table 4 on page 55 for the directoy these scripts are located in). Each of these scripts starts the import cascade at a specific per script and sends a mail including the transcript of the process to miceminer@gmail.com upon completion.

BASH script	perl script
import_step_1.sh	logimport.pl
direction_step_2.sh	searchdir.pl
results_step_3.sh	searchres.pl
meetings_step_4.sh	meetings.pl
counter_step_5.sh	counter.pl

Table 5: Overview of the BASH scripts and the perl import script they start.

These scripts can be started manually from the command line or through a so called *cronjob* which schedules the execution of the scripts.

```
# m h  dom mon dow   command
0 23 * * *       /var/www/mouse/import_scripts/import_step_1.sh
```

The above listing shows the actual *cronjob* which starts the import cascade every day at 11 o'clock p.m. beginning with the `logimport.pl` script.

B.3. Configuration

The application wide configuration is stored in a single XML file. The different possibilities to access or view the file can be found in the 4 table and the 3 table.

Please be careful on what you change. Never move the file to another location or delete it. Some minor knowledge in XML is advantageous to make changes in the configuration. It is strongly recommended to make a backup copy of the original file before applying changes. The file contains some explanations of what can be changed easely and what should not be changed.

B.3.1. Database

The following listing shows the part of the XML-configuration file responsible for the database connection to the *MySQL* server.

The values for the connection are set in the four nodes, `<dbname>` contains the database name, `<dbuser>` the user name, `<dbpass>` the user password and `<dbsocket>` the MySQL socket. This allows to switch the database, but only whne the table structures match (see the entry for the SQL file to create the database in table 3 on page 52).

The mapping to the database tables can be changed by setting the node value to another table, but only when the table structures match.

B.3.2. Perl

The perl part of the XML configuration file.

Listing 2: Perl configuration

```
<perl>
        <!-- the folder with the perl import skripts -->
        <scriptsFolder>/var/www/mouse/perl/import/</scriptsFolder>
        <!-- The mysql user which has SELECT and LOCK TABLES privileges on
             the database specified
                in <dbname> and needs NO password -->
        <dbBackupUser>mice_backup</dbBackupUser>
        <!-- the location to save the backup -->
        <dbBackupDirectory>/var/www/mouse/dbbackups/</dbBackupDirectory>
        <!-- the maximal time in seconds that can elapse between the two
             antennas at a box
                to build a valid direction result -->
        <antennaInterval>5</antennaInterval>
</perl>
```

scriptsFolder The directory containing the perl *Import Scripts* (see section 5 on page 27 for details).

dbBackupUser A MySQL user with `SELECT` and `LOCK TABLES` privileges on the database set as the `<dbname>` value in the databse configuration (see appendix B.3.1 on page 57).

dbBackupDirectory The directory in which the database backups are saved which are created at the beinning of the data import process.

antennaInterval Number of seconds that are allowed to elapse between the transponder readings at the inner and the outer antenna of box, to form a valid *direction result*.

B.3.3. GUI & PHP

Listing 3: Maximum hours of stay results

```
<!-- the maximum duration of a stay result in hours -->
<maxStay>9</maxStay>
```

Listing 4: Components configuration

```
<!--
   Choose which components to create when the application is loaded by
       setting the onStart attribute to 'true' or 'false'.
   The name attribute values correspond to the button labels in the top menu
       of the application.
   Do not change the name attribute values!
-->
<components>
        <component name="Browse Data" onStart="true">
                <info><![CDATA[
                        <p>This component is used to browse and filter the
                            data.</p>
                        <p>Please check out the <a href="http://zool-
                            miceminer.uzh.ch/index.php#browse_data_sc" target
                            ="_blank">different screencasts</a> for the
                            Browse Data component.</p>
                ]]></info>
        </component>
        <component name="Data Overview" onStart="false">
                <info><![CDATA[
                        <p>This components lists the mice, boxes and antennas
                             with their attributes.</p>
                        <p>The difference between the <b>Browse Data</b>
                            component and this component, is that the counts
                            show the values for the whole period of data in
                            the database and not over a specific date range
                            .</p>
                        <p>The filter possibilities are basically the same as
                             in the <b>Browse Data</b> component.</p>
                        <p>Getting data for a mouse, a box or an antenna is
                            not possible in this component.</p>
```

```
102                 ]]></info>
103         </component>
104         <component name="Data Analysis" onStart="false"/>
105         <component name="Graph Data" onStart="false">
106                 <info><![CDATA[
107                 <p>Graph Data help coming soon ...</p>
108                 ]]></info>
109         </component>
110         <component name="Upload Files" onStart="false"/>
111 </components>
```

Shown in the next listing is the beginning of the configuration for the grids in the GUI. The different part of the configuration is explained on the spot in the configuration file.

Listing 5: sample Grid configuration

```
418 <grids>
419         <!--
420                 The grids are the ui components which look like tables.
421                 You can change the label and width attributes as you like.
422
423                 The label sets the string for the header part of the column.
424
425                 You will find some further documentation of the other
                        attributes below.
426
427         -->
428         <itemsGrids>
429
430                 <!--    item grid (grid child of <itemGrids>)
431
432                                 id: unique identifier of this grid
433                                 label: data grid label
434                                 method: the method in the remote object
                                        source (amfphp services file -> Mice.php)
435                                 filename: the prefix when exporting the file
                                        to excel
436                                 initsort: points to one of the col field id
                                        attribute values and sets this column as
                                        the initial sort column
437                 -->
438
439                 <grid id="miceItemAll" label="mice" header="mice" method="
                        getMice" filename="mice_list"
440                             initsort="data_count">
441
442
443                         <!--    columns (col) for the grid
444
445                                         field: the field in the data provider
                                                or database
446                                         label: column label
447                                         width: column width in pixel
```

```
448                                         sort: what kind of sort to offer for
                                                this column (shown in the header
                                                when choosed to filter this
                                                column) -
449                                                         alphanum -> textfield
450                                                         numeric ->
                                                                slider numeric
451                                                         date -> slider date
452                         -->
453
454                         <col field="id" label="rfid" width="80" sort="
                                alphanum"/>
455                         <col field="last" label="last dataset" width="100"
                                sort="date"/>
456                         <col field="implant_date" label="implant date" width=
                                "60" sort="date"/>
457                         <col field="sex" label="sex" width="40" sort="
                                alphanum" />
458                         <col field="data_count" label="dataset count" width="
                                100" sort="numeric" />
459                         <col field="dir_count" label="in/out count" width="70
                                " sort="numeric" />
460                         <col field="res_count" label="stay count" width="70"
                                sort="numeric" />
461
462                         <!--    chart
463
464                                 the chart to show for this grid
465                         -->
466
467                         <chart type="colChart" id="genItemChart" />
468                 </grid>
```

C. Glossary

Relational Database Management System A Relational Database Management System is a Database Management System in which data is stored in the form of tables and the relationship among the data is also stored in the form of tables[18]. 5

Database Management System A Database Management System is computer software that manages databases[12].

Radio-frequency identification Radio-frequency identification (RFID) is the use of an object (typically referred to as an RFID tag) applied to or incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves[19]12

Structured Query Language The Structured Query Language is a database computer language designed for the retrieval and management of data in relational database management systems.[21] 26

KJ The Joule is a derived unit of energy and defined as the work done by a force of one newton travelling through a distance of one metre.[15]. 1 Kilo Joule = 10^3 Joule. 8

perl Perl is a high-level, general-purpose, interpreted, dynamic programming language.[16] 18

PVC Polyvinyl chloride is a widely used thermoplastic polymer[17] 13

GUI A graphical user interface is a type of user interface which allows people to interact with electronic devices such as computers [. . .] with images rather than text commands[14] 26

API An application programming interface is a set of routines, data structures, object classes and/or protocols provided by libraries and/or operating system services in order to support the building of applications[10]. 52

AFP The Apple Filing Protocol is a network protocol that offers file services for Mac OS X[9]. 52

Samba Samba allows file and print sharing between computers running Windows and computers running Unix [20]. 52

XML Extensible Markup Language is a general-purpose specification for creating custom markup languages [22]. 54

CGI The Common Gateway Interface is a standard protocol for interfacing external application software with an information server, commonly a web server [11]. 55

BASH Bourne-again shell is a free software Unix shell written for the GNU Project /citewiki:bash. 55

GIT Git is a free distributed revision control, or software source code management project /citewiki:git. 52