Ornis Hungarica 2015. 23(2): 163–188.
DOI: 10.1515/orhu-2015-0018

Hitchhikers' guide to analysing bird ringing data

Part 1: data cleaning, preparation and exploratory analyses

Andrea Harnos[1*], Péter Fehérvári[2] & Tibor Csörgő[3]

Andrea Harnos, Péter Fehérvári & Tibor Csörgő 2015. Hitchhikers' guide to analysing bird ringing data – Part 1. – Ornis Hungarica 23(2): 163–188.

Abstract Bird ringing datasets constitute possibly the largest source of temporal and spatial information on vertebrate taxa available on the globe. Initially, the method was invented to understand avian migration patterns. However, data deriving from bird ringing has been used in an array of other disciplines including population monitoring, changes in demography, conservation management and to study the effects of climate change to name a few. Despite the widespread usage and importance, there are no guidelines available specifically describing the practice of data management, preparation and analyses of ringing datasets. Here, we present the first of a series of comprehensive tutorials that may help fill this gap. We describe in detail and through a real-life example the intricacies of data cleaning and how to create a data table ready for analyses from raw ringing data in the R software environment. Moreover, we created and present here the R package; `ringR`, designed to carry out various specific tasks and plots related to bird ringing data. Most methods described here can also be applied to a wide range of capture-recapture type data based on individual marking, regardless to taxa or research question.

Keywords: data cleaning, R statistical software, banding data, statistical analysis, mark-recapture, data management

Összefoglalás Feltehetően a madárgyűrűzésből származó adatok szolgáltatják a leghosszabb időtávot felölelő és legtöbb adatot tartalmazó gerinces adatbázist a Földön. A gyűrűzés eredeti célja a madarak vonulási útvonalainak feltérképezése volt, azonban más kutatási területeken is felhasználásra kerültek ezen adatok, például állomány monitoringra, demográfiai változók becslésére, illetve a klímaváltozás hatásainak feltárására, hogy csak néhányat említsünk. Annak ellenére, hogy ilyen fontos és széles körben használt adatforrás, meglepő módon nincs egységes útmutató arra vonatkozóan, hogy hogyan kell az adatokat előkészíteni és megfelelően elemezni. A jelen munka az első része egy olyan sorozatnak, amely részletesen tárgyalja a gyűrűzési adatok kezelését, elemzését az R statisztikai program használatával. Ebben az első részben egy valós példán keresztül bemutatjuk az adattisztítás lépéseit, vagyis hogy a nyers gyűrűzési adatokból hogyan készítsünk elemezhető adattáblázatot. Bemutatjuk továbbá az általunk készített `ringR` nevű R könyvtárat, amely olyan függvényeket tartalmaz, amelyek segítik speciális, a gyűrűzési adatokat jól jellemző ábrák és leválogatások elkészítését. Bár gyűrűzési példán mutatjuk be az elemzéseket, feltehetően olyanok számára is hasznos lehet ez az útmutató, akiknek a gyűrűzési adatokhoz hasonló struktúrájú, egyedek jelölésén alapuló adatsoraik vannak.

Kulcsszavak: adattisztítás, R statisztikai program, gyűrűzési adatok, statisztikai elemzés, jelölés-visszafogás, adatkezelés

[1]*Department of Biomathematics and Informatics, Szent István University, Faculty of Veterinary Science, 1078 Budapest, István utca 2., Hungary, e-mail: harnos.andrea@gmail.com*
[2]*Department of Zoology, Hungarian Natural History Museum, 1088 Budapest, Baross utca 13., Hungary*
[3]*Department of Anatomy, Cell- and Developmental Biology, Eötvös Loránd University, 1117 Budapest, Pázmány Péter sétány 1/C, Hungary*
**corresponding author*

Introduction

Ever since Mortensen first fitted starlings *(Sturnus vulgaris)* with his home-made engraved plates in 1890 (Preuss 2001) hundreds of millions of birds were ringed to-date (e.g. EURING databank: www.euring.org, North American Bird Banding Laboratory (BBL) www.pwrc.usgs.gov/bbl, South African Bird Banding Unit, http://safring.adu.org.za). The method of deploying rings (or various other unique tags that remain on the individual throughout its lifespan) spread all over the world, consequently creating a global network of ringing schemes (e.g. Greenwood 2009), ringing stations (e.g. Pilastro *et al.* 1998, Nowakovski 2003, Crewe *et al.* 2008) and individual ringers. Therefore, bird ringing datasets constitute possibly the largest source of temporal and spatial information on any vertebrate taxa (except human) available on the globe. Initially, bird ringing was designed to understand the spatial patterns of migration and is still one of the benchmark methods to assess movements of bird species despite its limitations (Thorup *et al*. 2014). Fitting individual identifiers to birds was soon revolutionizing other disciplines including ecology, behaviour, demography, morphology and effects of climate change on vertebrate systems to name a few (Spina 1999, Dingemanse *et al.* 2003, Robinson *et al.* 2007, Brown & Oschadleus 2008, Osenkowski *et al.* 2012). Indeed, over 120,000 papers have been published to date that used or mentioned bird ringing data to some extent (www.scholar.google.com, search terms: *"avian"* OR *"bird"* AND *"ringing*" OR *"banding"-"chromosome banding"*, accessed 2015-11-25).

Working with ornithologists and field biologists however, showed us that theoretical misconceptions and bad practice in data analyses of bird ringing datasets are surprisingly common despite the generality, and interdisciplinary usage of ringing. Thousands of tutorials, textbooks, online courses are readily available to help the analysis, however it is easy to get lost amidst the abundance of possibilities. Here we will give yet another tutorial, or rather a series of tutorials. We specifically deal with general avian ringing data and present guidelines of good practice on data management and analyses. The methods and plots described can easily be generalized to various capture-recapture type data regardless to taxa or the research question, making it possibly useful for researchers working in different fields.

The first part of the series introduces the reader to the basic concepts of statistics (see *Box 1*) and serves as a guide for exploring and cleaning raw data. We address the most frequent problems, their solutions with example computations. Our philosophy is to give the reader a perspective on what we believe is good practice in bird ringing data management and producing guidelines and a selection of techniques we found the most useful when analysing ringing datasets.

Data sources

Bird ringing is more than just a research methodology, it is also a recreational activity and an effective tool for environmental education (Dearborn & Kark 2010, Şekercioğlu 2012). Thus, data may derive from sources ranging from individual ringers, demonstrative temporal trapping sessions, species-specific research projects to large scale continuous ringing stations. These sources can be categorized to two groups: (1) presence only data and (2) data from standardized sampling. The first group constitutes data deriving from sources when bird ring-

Population and sample

Although both statisticians and biologists use the term *population,* the meaning is different in the two disciplines. This overlap in terminology may often lead to confusion, typically when obtained results are inferred.

Statistical population is the set of objects, individuals etc. that we wish to (or can) make inference on. For instance, consider the working example of this tutorial, Pied Flycatchers migrating through the Ócsa Bird Ringing Station. Defining the sensu stricto biological population in this case is difficult as we hardly have information on where each individual breeds, although intuitively the biological population might be all birds migrating through Hungary, or the Carpathian Basin. Defining the statistical population is more straightforward: it is the set of birds migrating through the surroundings of the station that we could have or did actually captured. The individuals captured constitute our **sample**, and, based on our sample, we make inference on all the individuals we did not catch but could have captured. In this case the **observational or sampling units** (elements of the sample) are the individuals that we caught and measured, the **observations** are the characters of the sampling units that we measured (e.g. wing length) or observed (e.g. age).

Box 1. Basic concepts
1. doboz Alapfogalmak

ing was carried out as a recreational, educational activity or when the aim was to maximize the number of individuals/species ringed. Quite often only the data of capture is available in these cases without any further information on habitat, trapping method and effort (e.g. number of nets used and the duration of trapping). Presence only data is suitable for qualitative analyses of species or population specific movements (Thorup *et al*. 2014), or for using it to model distribution patterns on a large spatial scale (e.g. Walther *et al.* 2007). In the second group, bird ringing was carried out with a more or less constant trapping effort and with similar methodology over time. Typically, these data are from ringing stations operating at constant locations, citizen science projects with standard methodology (e.g. CES, Peach *et al.* 1998, 2004). Species specific research and conservation programs also constitute this latter group. These data sources – albeit with careful consideration – may be used to answer questions related to demography, phenology or other topics when trapping probabilities are explicitly or inherently the subject of analyses.

Sampling issues

A research process can be described in several consecutive steps: (1) identify the topic and research question, (2) define the population *(Box 1)* to be inferred on and formulate hypotheses, (3) design sampling process, (4) collect data (5) analyse data and (6) interpret and report results. However, when analysing retrospective data, this structure cannot be followed as data has already been collected. Therefore, steps (4) and (5) could be rephrased as: ‘evaluate whether the available data is collected in a fashion suitable for answering the research question addressed’. Ornithologists often mistakenly refer to, and hence analyse long-term bird ringing data as unbiased samples of a given population. However, standard field methodology does not necessarily produce representative samples. A sample may be representative of a specific aspect while non-representative from another. For instance, we have ringing data with biometric measurements of Red-backed Shrikes *(Lanius collurio)* caught with live lure bownets at a given location in autumn. The birds were captured during the same time of the year, in similar habitats and at a well-defined location (e.g. surroundings of a ringing station). All birds were handled and measured by the same ringer using the same equipment, making the sampling procedure standard. The wing length measurements of these birds may be considered a representative sample of the wing length of all shrikes at the location. However, this may not be true for body mass measurements as we used traps that require the birds to be hungry. There are no tests available that may help us evaluate how well our sample represents the statistical target population, and as the example above shows, it depends on the question addressed. Therefore, we highly recommend a thorough consultation process with peers on whether data can be considered as a representative sample suitable for addressing specific research questions, prior to initiating any further analyses.

Data cleaning and management

Once we have confirmed that our data is suitable, the next step is data cleaning. In the following sections we will lead you through the intricate steps necessary to transform raw ringing data to technically correct data suitable for statistical analyses (de Jong & van der Loo 2013). Data cleaning is a three-stage recursive process consisting of screening, diagnosing and editing abnormalities. Screening typically entails (1) graphical representation of distributions (boxplots, histograms and scatterplots) for numerical variables, (2) creating frequency distributions and contingency tables in case of factors, and (3) calculating summary statistics. All these predominantly serve the purpose of pin-pointing anomalies. Once identified, these anomalies have to be carefully investigated and if possible their potential cause diagnosed. Quite often this diagnosis (or the lack of diagnosis) will then help deciding how to treat these data anomalies. These steps are then repeated as many times as necessary prior to starting advanced statistical analyses. Note that screening is in essence exploratory data analysis, therefore it also allows an analyst to familiarise with patterns in the data, facilitating the implementation of future, more complex statistical analyses.

Data cleaning is often neglected or carried out crudely, yet it has crucial repercussions on the obtained results. Bird ringing datasets are often heterogeneous in coding and are prone to systematic measurement bias changing over time, due to the fact that they were often collected over a long-time period and recorded by various people. Therefore, a rigorous data cleaning procedure is essential and cannot be over-emphasized. Statistical societies recommend that description of data cleaning be a standard part of reporting statistical methodology (American Statistical Association 1999, Broeck *et al.* 2005). We believe that it should also be a fundamental part of reporting the analyses on ringing datasets.

The software used

For data cleaning and analyses, we use R (www.r-project.org, R Core Team 2015a), a free software environment that has become the benchmark of statistical data analyses

Variable types

Variables are in essence the characteristics we measured or observed. We can differentiate variables based on their measurement scales:

Nominal variables consist of two or more categories and these categories lack an intrinsic order. For example, sex of a bird, colour, habitat type etc.

Ordinal variables have two or more categories just like nominal variables, only the categories can also be ordered or ranked. Fat or muscle scores are ordinal variables in our case. Nominal and ordinal variables are also called as qualitative or categorical variables. These are the so called factors in R. The only meaningful summary of these variables are counting the observations in the categories. We can code the categories with characters, or character series (e.g. M or male) or even numbers (e.g. 1, 2, …), but note that although mathematical operations may be carried out, these are totally meaningless.

Interval variables can be measured along a continuum and have meaningful distances between measurements defined, but the zero value is arbitrary (for example, temperature measured in degrees Celsius or Fahrenheit). So the difference between 20 °C and 30 °C is the same as between 30 °C and 40 °C.

Ratio variables have both a meaningful zero value and the distances between different measurements are defined. For example, temperature measured in Kelvin is a ratio variable as 0 Kelvin (often called absolute zero) indicates that there is no temperature and 4 °K is the double of 2 °K. In our case, length and mass variables are measured on the ratio scale (and also on the interval scale). Interval and ratio scale variables are called as quantitative or numeric variables.

Box 2. Variable types
2. doboz Változó típusok

in the past decade. The major advantage is its disadvantage: it is an environment where you have to write commands in a specific language. Learning this language takes time and patience which are commodities not all biologists have. This tutorial is not to teach users the R environment, but to help people start data analyses with limited knowledge of the R language. In case you want to start learning from the very beginning, we refer you to tutorials like Venables *et al.* (2015) and our own beginner's guide to basic syntax and coding (see the online appendix). Furthermore, this tutorial does not substitute a statistical course. We suppose the readers have a basic understanding of statistics. However if you feel the need to refresh your knowledge we suggest reading Fowler and Cohen (1996), Freedman *et al.* (2007) or Shahbaba (2012) and for Hungarian readers, Reiczigel *et al.* (2014).

Those, who are still intimidated by typing codes and would rather open menus and fill dialogue panes, we recommend using the R Commander interface (Fox 2005). In R Commander the codes can be generated and run by menus and dialogue panes. More detailed information about R Commander can be found on the www.rcommander.com website, or see Shahbaba's (2012) book for an introductory textbook using R Commander.

We prefer using R because the scripts (series of commands) can be saved as a simple text file (with the extension name ".r") and rerun. If we have to make similar analysis, we have the codes ready, and the analysis can be reconstructed at any time. It is very useful to meticulously comment the codes. This additional work will enormously help us later when we want to repeat our analysis and have to remember the process, as well as when we have to write about the applied statistical procedures when preparing a manuscript for publication. The R codes of this paper were written in RStudio (www.rstudio.com), an integrated development environment for R, free to use for non-commercial purposes. It runs on Linux, MS-Windows and OS X. It includes a console, a syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management. The tutorial was prepared using the R Markdown (Allaire 2015) authoring format that enables creating dynamic documents, presentations, and reports from R easily. It combines the core syntax of markdown (an easy-to-write plain text format) with embedded R code chunks to be executed, so their output can be included in the final document. R Markdown documents are fully reproducible (they can be automatically regenerated whenever the underlying R code or the data changes). RStudio has many features that make it easy to write and run R Markdown documents.

We strongly recommend trying to run the illustrating R codes. The online appendix (OH_2015_23(2)_163-188_appendix.zip) of this tutorial includes the scripts along with the entire code we used. We also recommend trying to write similar codes to work with your own data, because this is the only way to learn using R.

The `ringR` *package*

We prepared a specific R package, `ringR` including useful functions to handle ringing data (determine first capture events, calculate elapsed time between two capture events etc.) and to make specific plots (e.g. to plot capture frequencies). These functions can be used in other fields where similar capture-recapture data are collected. The `ringR` package can be downloaded from ringR.gryllosoft.hu along with the installation instructions (ringRinstall.pdf).

Technical comments for the codes and the data

R-codes, variable names, keywords or parameters – if these are in the text or in the examples – are differentiated by the font type used e.g. `t.test()`, `WING`, `AGE`, `alternative = 'two sided'`. Note that we only describe function arguments or parameters that we specifically used in the analyses. The functions generally have more parameters to set, please refer to their help page for a comprehensive description.

We used R 3.2.2 for the analysis and data preparation on the Ubuntu 14.04 platform. The codes were written with RStudio 0.98.1103.

How should a data table look like?

For data analysis, data should be organised into a table, including variables as columns and observations as rows. The first row should consist of the column headings (variable names, see *Box 3* for naming conventions). Each subsequent row has to have the same amount of data (columns). Make sure that your observations (records) and the individuals (typically ring number) both have unique identifiers. There should be nothing in the tables apart from your raw data.

We strongly discourage using the space (rather use the underscore "_" character instead) and unconventional characters like letters with accents, all mathematical ope-

Recommendations to name variables

Select simple but explanatory variable names using naming conventions compatible with the program that will be used for analysis. Each name in the data set must be unique. Begin variable names with a letter (A–Z). Variable names may include numbers, although not as the first character, but not blanks or other special characters (an underscore or point are valid). You may use upper and lowercase letters to suggest word separation or use decimal points or the underscore character. For example, BodyMass, Body.Mass, BODY_MASS or Body_mass are perfect names. The key is to remain consistent throughout the analysis. A good idea may be to differentiate original variables and newly created variables in the naming, for example we use capital letters for original and lower case lettering in newly derived variables.

Box 3. Recommendations to name variables
3. doboz Javaslatok a változók elnevezéséhez

rators like = or / in any part of the table, including the heading. Also avoid using thousand separators. These characters may be misinterpreted by the statistical software, causing an array of problems. Note that these rules hold also for file and folder names.

The simplest way to store data is saving it to a delimited text files (i.e. data in the text file are separated by a predefined field separator, such as a semicolon, comma, tab or space; the separator defines the columns seen in a spreadsheet editor).

Data used to illustrate the text

Through most of the text we are going to illustrate the procedures to process data originating from the bird ringing database of the Ócsa Bird Ringing Station found in central Hungary (Harnos *et al.* 2015a, Kovács *et al.* 2011, 2012). The area is a post-glacial relic bog, with an extensive central Reed-bed, surrounded by woodlands and shrubs. We demonstrate the concepts and practice using the data of the Pied Flycatchers *(Ficedula hypoleuca)* collected between 1984 and 2014 at Ócsa, a randomly selected subset of the dataset used in the paper 'Sex and age dependent migration phenology of the Pied Flycatcher in a stopover site in the Carpathian Basin' (Harnos *et al.* 2015b) published in the same volume of Ornis Hungarica (see description of the data there). In the accompanying online appendix of this paper you can find the dataset *(Box 6)* as it was exported from the Bird Ringing database of the Ócsa Bird Ringing Station (except that the variable names were translated to English). While the variable names of the original dataset *(Table 1)* are in uppercase, the names of the newly created variables are in lowercase.

Data manipulation in a Spreadsheet program

Cleaning data and preparing it for analysis is a tiresome, yet fundamental part of data analyses. Although it may seem that doing it in R is less effective yet we recommend that spreadsheet editors like Ms Excel or OpenOffice Calc (Elliott *et al.* 2006) only be used for tasks that (1) involve data entry, (2) do not need to be traced or reconstructed and (3) do not modify data values. For instance quite often values are entered and later deleted from cells during the data

Column	Variable name	Description (units)	Format	Codes and ranges	Missing values
A	ID	unique identification number of observation (record)	Numeric (6.0)	1–999999	
B	DATE	date of capture	Date (MM/DD/YYYY)	01/01/1989–12/31/2014	
C	RECAP	capture or recapture?	Numeric	0: first capture, 1: recapture	
D	RING	ring number	Text		
E	AGE	age of bird	Text (2)	*Table 2.*	null
F	SEX	sex of bird	Text (1)	F: female, M: male	null, N
G	FAT	fat score	Numeric (1.0)	0–8	null
H	MUSCLE	muscle score	Numeric (1.0)	0–3	null
I	MASS	body mass (g)	Numeric (3.1)	0–99	null, 0
J	WING	wing length (mm)	Numeric (2.0)	0–99	null, 0
K	THIRD	length of third primary (mm)	Numeric (2.0)	0–99	null, 0
L	TAIL	tail length (mm)	Numeric (2.0)	0–99	null, 0

Table 1. Description of the variables of Pied Flycatcher dataset
1. táblázat A kormos légykapó adattáblázat változói

entry process. MS Excel may store information in these cells, despite they remain visually empty. These nuisance values will either hinder data import to R or will cause empty rows or columns that are unnecessary. Getting rid of these is relatively easy, simply select the range of values you wish to keep (the dataset with valid information) and copy it to a new spreadsheet and save it as a text file (preferably .csv extension).

These editors can be also used safely to (1) merge data tables to one (providing that the number of rows is in a reasonable range), (2) changing the number of decimals and (3) changing unconventional characters described above.

Data cleaning and manipulating in R

Getting data into R

Setting the working directory

The working directory is a directory (folder) on a computer or on an external data storage device, including clouds where R searches, saves and loads files (data files, figures etc.). If the working directory is not specified, R works from the default working directory. Its location depends on the operating system and the function `getwd()` serves to find this location. The first task is to set the working directory to where all the data and script files are stored. For example, if data are in the `D:\mydirectory` directory under MS-Windows, then type:

```
>setwd("D:/mydirectory")
```

Note that the directory separator can either be "/" or "\\" (but not "\").

Importing data from a text file

Most statistical software can import and export text files. In R these can be imported with the generic `read.table()` function. However, we recommend checking the character encoding of your dataset, for instance by opening it with a simple text editor, such as Geany (open source text editor available for the major operating systems, www.geany.org). The encoding can be set by the `encoding` argument of the `read.table()` function (see `?read.table`). Also inspect (1) if there are variable names in the first row (`header = TRUE`); (2) the type of the field separator (in case of semicolon: `sep=';'`); (3) the decimal character (default: `dec='.'`); (4) the missing value character (default: `NA`, in our case: `na.string = 'null'`); (5) if there are row names in the first column of the file (default: `FALSE, row.names = TRUE`, if yes.); (6) if the character strings are quoted (default: `" or '`). For further settings refer to the R help (type `?read.table` and press Enter). The `read.table()` function returns a data frame, the most commonly used data structure in R (for details please refer to `introR.r` attached to this tutorial as an online appendix or see e.g. Venables 2015).

```
>mydata = read.table("FICHYP.csv",
 sep = ";", dec = ".", header = TRUE,
 na.string = c("null"))
```

Reading the data from Excel

Alternatively, use the `xlsx` package (Dragulescu 2014) to access Excel files. The first (and only the first!) row should include variable/column names. The sheet's index (`sheetIndex`) or the sheet's name (`sheetName`) has to be specified.

```
>library(xlsx)
>mydata = read.xlsx("FICHYP.xlsx",
  sheetIndex = 1)
```

Reading data from a database

One can also acquire data directly from a relational database if the necessary permissions are available. Basically, there are two ways to connect to databases in R. The first uses the ODBC (Open DataBase Connectivity) facility (`RODBC` package), the second uses the DBI package along with the specialized package needed to access that particular database (Spector 2008). Data from just about any relational database is accessible from R by sending an SQL query to the standard interfaces (R Core Team 2015b).

Overview of the data

Viewing the structure of the dataset

The internal structure of an R object can be compactly displayed with the `str()` function *(Box 4)*. For data frames, it shows the number of observations and variables and gives the type of the variables, and the first few values of these. The basic variable types are numeric (`num`), integer (`int`), factor (`Factor`), character (`char`) and logical (`logi`) etc. Note that these are somewhat different types as we previously defined *(Box 2, Table 1)*. For instance wing length (`WING` in our dataset) is typically measured to the nearest millimetre, hence it becomes an integer, while body mass (`MASS`) is often measured to the nearest 0.1 g, thus stored as a numeric variable. In case of quantitative variables, these deviations have little effect on further analyses. On the other hand, factor variables are used to store both nominal and ordinal categorical data. In case of numeric or character variables, `str()` lists the first few values. In case of factors, `str()` function gives the number of levels, the first few levels and the numeric codes of the first few observations. The numeric codes are integers from 1 to the number of levels and are given in the alphabetical order of the levels.

NOTE: If the imported data frame includes factors instead of numeric values, check (1) the settings used when reading the table; (2) if the correct character was used for specifying the decimal point; (3) if the correct character strings were used to specify missing values; (4) values in a numeric variable includes any characters other than figures (note that the letter `"o"` is NOT equivalent to the number `"0"`!). All different values of a factor variable can be listed using the `levels()` function with the name of the variable (e.g. `levels(mydata$MASS)`). If a factor appears as a numeric variable, than probably your categories were coded by numbers. For conversion to a factor, read the next chapter.

Conversions between numeric and factor variable types

The `as.numeric()` function converts factors (and other types) to numeric variables. Note that this function can NOT be applied to correct incorrectly entered or imported variables! We strongly recommend correcting the clearly mistyped values in the original data file itself and then reimporting the data with R.

The `as.factor()` (or simply `factor()`) function converts all types of variables (including numerics or integers) to factors. By default, values that cannot be converted to the specified type will be converted

```
# Structure of the table
>str(mydata)

'data.frame':1966 obs. of  12 variables:
 $ ID : int  115849 115577 115288 115650 ...
 $ DATE   : Factor w/ 929 levels "10/1/1993","10/9/1999",..: 169 607 636
100 254 ...
 $ RECAP : int  0 0 0 0 0 0 0 0 0 0 ...
 $ RING  : Factor w/ 1703 levels "2E5711","2E9089",..: 222 223 224 225 259
260 ...
 $ AGE   : Factor w/ 7 levels "1","1+","1Y",..: 3 3 3 6 3 3 3 3 6 6 ...
 $ SEX   : Factor w/ 3 levels "F","M","N": NA NA NA 2 NA NA NA NA 1 2 ...
 $ FAT   : int  0 2 2 0 3 1 2 4 1 0 ...
 $ MUSCLE: int  NA NA NA NA NA NA NA NA NA NA ...
 $ MASS  : num  11.4 13.3 13.2 12.6 14.4 12.3 13.2 12.2 12.3 13.1 ...
 $ WING  : int  NA NA NA 81 82 82 81 79 80 NA ...
 $ THIRD : int  NA NA NA NA NA NA NA NA NA NA ...
 $ TAIL  : int  NA NA NA 52 55 54 54 55 55 NA ...
```

Box 4. Structure of `mydata`
4. doboz A `mydata` struktúrája

```
>mydata$FAT = factor(mydata$FAT)
>str(mydata$FAT)

Factor w/ 7 levels "0","1","2","3",..: 1 3 3 1 4 2 3 5 2 1 ...

>mydata$MUSCLE = factor(mydata$MUSCLE)
>str(mydata$MUSCLE)

Factor w/ 4 levels "0","1","2","3": NA NA NA NA NA NA NA NA NA NA ...

>mydata$ID = factor(mydata$ID)
>str(mydata$ID)

Factor w/ 1966 levels "1120","1570",..: 636 634 633 635 591 594 586 589
588 593 ...
```

Box 5. Conversion of `FAT`, `MUSCLE` and `ID` to factors
5. doboz A `FAT`, `MUSCLE` és `ID` változók faktorrá konvertálása

to `NA` values with a warning. In our case, the `FAT` and `MUSCLE` variables are measured on ordinal scale and coded with numbers, thus looking like numeric variables. These should be converted to factors by using the `factor()` function. Often the ring number or other individual identifiers are also coded with numbers. These variables should be used as categorical and should be converted to factors. After conversion, DO check once again the structure of the variables with the `str()` function *(Box 5)*.

Viewing the table

Data frames can be opened, viewed and edited with the `fix()` function. If we want to inspect the first or last rows of the data frame, we can use the `head()` or `tail()` functions. The first parameter is the name of the dataset, the second (optional, with default value 6) is the number of rows displayed *(Box 6)*.

If we want to look at specific rows of the table, we just have to make a subset of it and submit the code. For example, if we want to look at the rows from 10 to 15, type: `mydata[10:15,]`. (For subsetting methods, check the `introR.r` script in the online appendix.)

Numerical summaries of the variables

Exploratory analyses are the first and among the most important steps in data analyses. These can help pin-point hidden anomalies such as incorrect data entry and can help the analyst familiarize to data patterns. We shall start with numerical summaries of the variables, by calling the `summary()` function with the name of the data frame. In case of factors, the `summary()` provides the frequencies for the first seven categories. In case of numeric or integer variables, it provides basic descriptive statistics: minimum (`Min.`), maximum (`Max.`), mean, the first quartile (`1st Qu.`), median and third quartile (`3rd Qu.`) values. The first quartile splits off the lowest 25% of data from the highest 75%, the median cuts the dataset in half and the third quartile splits off the highest 25% of data from the lowest 75%. The median gives a measure of the centre of the data, the minimum and maximum give the range of the data and the first and third quartiles give a

```
>head(mydata)

      ID      DATE RECAP   RING AGE  SEX FAT MUSCLE MASS WING THIRD TAIL
1 115849 7/19/1983     0 E03042  1Y <NA>   0   <NA> 11.4   NA    NA   NA
2 115577  8/7/1983     0 E03192  1Y <NA>   2   <NA> 13.3   NA    NA   NA
3 115288  8/9/1983     0 E03204  1Y <NA>   2   <NA> 13.2   NA    NA   NA
4 115650 4/29/1984     0 E24327  2Y    M   0   <NA> 12.6   81    NA   52
5 112284 8/14/1984     0 E47390  1Y <NA>   3   <NA> 14.4   82    NA   55
6 112623 8/21/1984     0 E48262  1Y <NA>   1   <NA> 12.3   82    NA   54
```

Box 6. Head of the data table
6. doboz Az adattáblázat eleje

sense of the spread of the data, especially when compared to the other values. The output also shows the number of missing values. In *Box 7*, we show the output of the `summary()` only for certain variables.

This output gives a quick overview of the data. Notice that we have invalid values in `MASS` like 0 (probably the missing values were coded by 0, which is a common mistake in data entry).

Extracting year and day of the year (`yearday`) from the `DATE` variable

Year and the number of days elapsed from a specified day of the year (often 1st of January) are usually needed for most analyses. Various original data files include a vast number of date formats and these probably have to be transformed. In our case, these variables can be extracted from the `DATE` variable e.g. by using the `ring.date()` function of the `ringR` package (see other R-packages such as `chron` to use other means of handling date and time data). The date variable (`DATE` in this example) can be set with `datevar` parameter.

The format (`format` parameter) of the date has to be specified. The `ring.date()` function tries to extract valid dates. A % code tells R to look for a range of substrings. For example, the `%d` indicator makes R look for numbers 1–31 where leading zeros are allowed, so 01 , 02 ,… 31 are recognized as well. The character that separates day, month and year (here '/') has to be specified: `%m` means the number of month (01–12), `%d` is the day of the month as decimal number (01–31), `%y` is the year without century (00–99) and – alternatively – `%Y` is the year including century. The `ring.date()` function extracts the year and day of the year from

```
>summary(mydata[,c('AGE', 'SEX','FAT', 'MASS','WING')])

  AGE         SEX          FAT             MASS             WING
 1   :   7   F   :622   Min.   :0.00   Min.   : 0.00   Min.   : 0.0
 1+  :  40   M   :943   1st Qu.:0.00   1st Qu.:11.90   1st Qu.:78.0
 1Y  :1614   N   : 39   Median :1.00   Median :12.50   Median :80.0
 2   :  31   NA's:362   Mean   :1.47   Mean   :12.61   Mean   :79.2
 2+  :   3              3rd Qu.:2.00   3rd Qu.:13.20   3rd Qu.:81.0
 2Y  : 251              Max.   :6.00   Max.   :81.00   Max.   :91.0
 FEJ:  20               NA's   :55     NA's   :74      NA's   :274
```

Box 7. Basic summary statistics of the dataset
7. doboz Az adattáblázat alapvető leíró statisztikái

the `datevar` (here `DATE`) variable and appends it to the original data frame.

```
>library(ringR)
>mydata = ring.date(
  data    = mydata,
  datevar = 'DATE',
  format  = "%m/%d/%Y")
```

Useful plots to explore the characteristics of migration

Determining captures and recaptures

To analyse the timing of migration, we need to determine the first captures of an individual of the specified year. The `RECAP` variable distinguishes only the capture of an unmarked individual from the capture of a marked one, thus annual first captures can not be determined with this variable. Therefore, we need to create new variable that distinguishes annual first captures and recaptures. This can be done with the `ring.firstcapture()` function of the `ringR` package. The name of the data frame, the `periodvar` (here: `year`) and `perioddayvar` (here: `yearday`) variables and the `IDvar` (here: `RING`) have to be set. The new variable is called `period.recap` and it is appended to the data frame.

```
>mydata = ring.firstcapture(
  data          =  mydata,
  periodvar     = 'year',
  perioddayvar  = 'yearday',
  IDvar         = 'RING')
```

Yearly capture and recapture frequencies

It is often useful to visualize the number of captured birds per year. The `ringR` package provides the function `plot.periodic.captures()` that can create a graph depicting both annual capture and recapture frequencies *(Figure 1)*. Note that the `ydivider` parameter sets the interval lengths between tick marks on the y-axis.

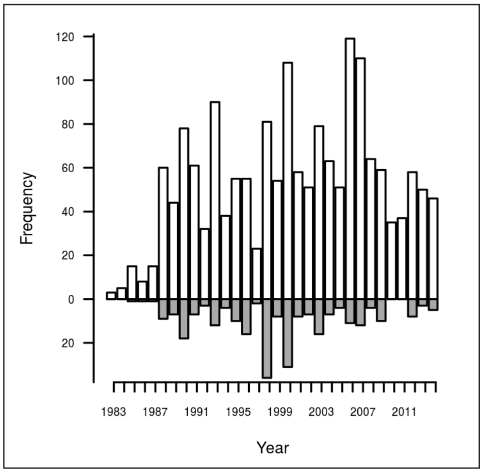

Figure 1. Yearly capture (white bars) and recapture (grey bars) frequencies
1. ábra Éves fogási (fehér oszlopok) és visszafogási gyakoriságok (szürke oszlopok)

```
>par(las=1, omi=c(0,0,0,0),
  mar=c(4, 4, 2, 2) + 0.1, bty='l')
>plot.periodic.captures(
  data          = mydata,
  periodvar     = 'year',
  perioddayvar  = 'yearday',
  recapvar      = 'period.recap',
  ydivider      = 20  )
```

We used the `par()` function to change the default layout and graphical settings prior to calling the plotting function. Please refer to the help of the `par()` function for further details. This layout setting is used in case of all further plots in the tutorial.

Daily capture and recapture frequencies

Pied Flycatchers are thought to migrate in two distinct migration waves at the study site. A potential way to corroborate this assumption is to plot the distribution (frequen-

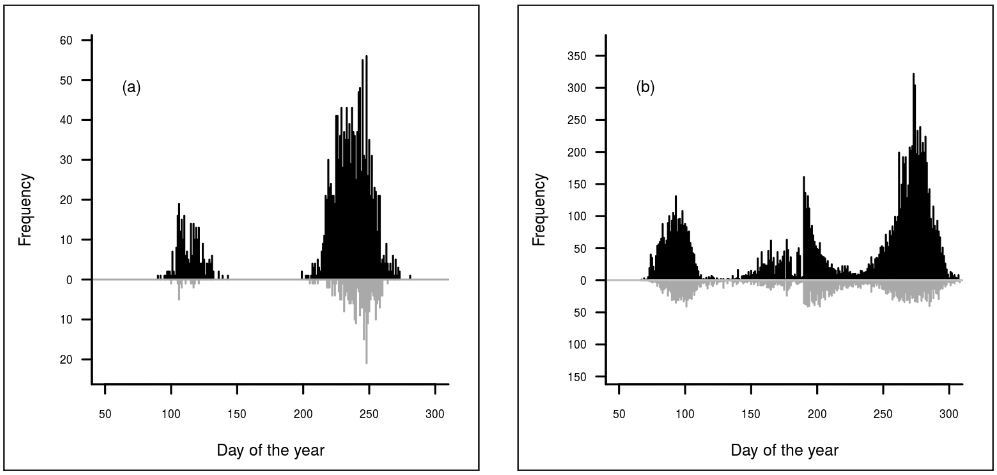

Figure 2. Daily first capture (black bars above 0) and recapture (gray bars below 0) frequencies in case of Pied Flycatchers (a) and Chiffchaffs (b)

2. ábra Napi fogási (fekete vonalak a 0 felett) és visszafogási (szürke vonalak a 0 alatt) gyakoriságok a kormos légykapó (a) és a csilpcsalpfüzike (b) esetén

cies) of dates (`yearday`) separately for captures (`period.recap == 0`) and recaptures (`period.recap == 1`). This can be done by the `plot.daily.captures()` function.

```
>plot.daily.captures(
  data          = mydata,
  perioddayvar = 'yearday',
  recapvar     = 'period.recap',
  ylim = c(0,0), xlim = c(50,300),
  xlab = "Day of the year",
  ylab = "Frequency",
  cap.color = "black",
  recap.color = "darkgray",
  ydivider = 10, title = NULL)
```

Figure 2a shows first capture and the recapture frequencies of a given day of the year. Indeed, there are no trapped individuals between the 150th and the 180th day of the year, indicating that the data can be grouped into two distinct seasons. To illustrate a species with a different behaviour at the study site, we included the Chiffchaffs *(Phylloscopus collybita)* (*Figure 2b*, data and code are not included). Chiffchaff capture frequencies show three overlapping waves: spring migration (days 70–110), breeding/dispersion (days 111–240) and autumn migration (days 241–310). Notice that seemingly there is an abrupt change in capture frequencies during the breeding season. This is caused by the sampling design, as CES protocol (a fraction of the nets is open each day) is applied in the first half of this period, while from the 190th day all nets were open until the end of the migratory period.

Cumulative capture dates

Another useful graph is the cumulative distribution of the date of first and last captures *(Figure 3a, b)*. For each individual, a horizontal black bar shows the time elapsed between the first and last captures at the study site. The individuals are ordered according to first capture dates. This graph may allow to simultaneously depict migration waves and stopover durations. The lack of data between the 150th and 180th days of the year is just as apparent as in the case of the previous

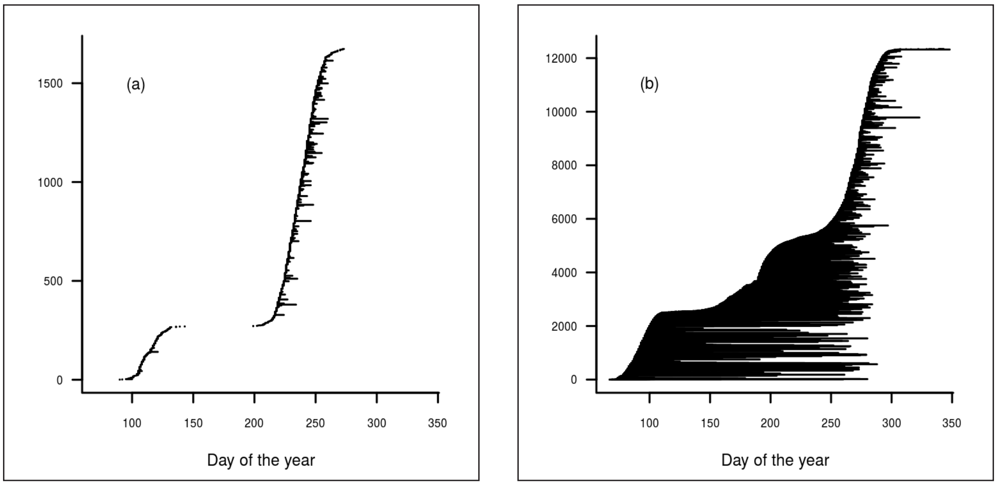

Figure 3. Cumulative distribution of the date of first captures depicted together with the last capture dates in case of Pied Flycatchers (a) and Chiffchaffs (b)
3. ábra Az első és utolsó megfogások (vízszintes vonalak) kumulatív eloszlása a kormos légykapó (a) és a csilpcsalpfüzike (b) esetén

graph showing capture frequencies, while we can also see that there is probably a considerable difference in stopover duration in spring and autumn. For Chiffchaffs *(Figure 3b)* there is no gap in the pattern indicating a continuous presence of the species, and a considerable proportion of the individuals arriving in spring remains at the study site through the trapping period. The changes in the rate of increase in the number of captured birds (the "steepness" of the cumulative curve) show the three previously identified waves.

These two figures allow the analyst to get a detailed overview of the basic species-specific annual behaviour pattern at a given study site. We recommend using both graphs simultaneously.

```
>plot.cum.captures(
  data          = mydata,
  IDvar         = 'RING',
  periodvar     = 'year',
  perioddayvar = 'yearday')
```

Defining migratory seasons

It may be important to separate observations belonging to the spring and autumn migratory phases, or any other predefined periods. Based on *Figures 2a* and *3a* the 150th day seems to be a reasonable cut-off point. Define a new variable called `season` using the `ifelse()` function. This function has the form `ifelse(test, yes, no)` `test` is the logical condition, `yes` is what will be returned if the logical condition `test` is true, and `no` is what will be returned when false. Note that the arguments of the `ifelse()`, as well as many other R functions can also be specified by just writing the values of the arguments following one another according to the help of the given function, without actually referring to the argument's name:

```
>ifelse(mydata$yearday < 150,
  "spring", "autumn")
```

It is not necessary to follow this order when referring to the arguments' names:

```
>ifelse(yes = "spring",
  no = "autumn",
  test = mydata$yearday < 150)
```

We need to convert the type of the resulting variable to a factor.

```
>mydata$season = ifelse(
  test = mydata$yearday < 150,
  yes  = "spring",
  no   = "autumn")
>mydata$season = factor(mydata$season)
```

Checking and exploring categorical variables

We can check the levels and the number of levels of a factor and the counts in the categories with the following codes:

```
>levels(mydata$AGE)
[1] "1" "1+" "1Y" "2" "2+" "2Y" "FEJ"

>nlevels(mydata$AGE)
[1] 7

>table(mydata$AGE)
1   1+  1Y     2   2+   2Y  FEJ
7   40 1614   31    3  251   20
```

The levels are overlapping and ambiguous as the same age classes were coded with different characters *(Table 2)*. Correct these mistakes and recode the variable so that corresponding age classes are represented with the same character string. For records where the age was 1st calendar year we change the values `1` to `1Y` using the `ifelse()` function. Note, that without the `as.character()` function inside the `ifelse()`, the `AGE` variable gets the numeric codes of the original

Code	Definition
1Y	first calendar year birds, hatched in the year of ringing
1+	older than a calendar year, exact age unknown
2Y	second calendar year birds, hatched in the year preceding ringing
2+	older than two calendar years, exact age unknown
FEJ	fully grown, otherwise undetermined
1	first calendar year birds, hatched in the year of ringing
2	second calendar year birds, hatched in the year preceding ringing

Table 2. Age class codes with definitions
2. táblázat Kor kategóriák és definícióik

factor. After such recoding, the variable has to be converted to factor again, using the `factor()` function. (2nd calendar year birds can be recoded in a similar fashion, see the online appendix.)

Warning! Recoding the original variable of a dataset is best avoided. In order to avoid damaging your original variable, we strongly recommend creating a new variable instead.

```
>mydata$AGE = ifelse(
  test = mydata$AGE == "1",
  yes  = "1Y",
  no   = as.character(mydata$AGE))
>mydata$AGE = factor(mydata$AGE)
#cheking the levels of the variable
>levels(mydata$AGE)

[1] "1+" "1Y" "2" "2+" "2Y" "FEJ"
```

Quite often, we only need to differentiate two age categories; juveniles (birds hatched in the year of ringing) and adults (all other birds where age is determined). Again, using the `ifelse()` function we can create a new variable where age will be recoded to the juv and the adult levels. The first call of the command function creates a new variable (age) in which records that are coded with 1Y in the AGE variable become juv all other records will be NA. The second ifelse() call takes the new age variable and recodes it so as 1+, 2+ and 2Y records become adult, all others remain either NA or juv. (The | symbol is used for the OR logical operator in R.) Notice that the age values for the birds with undetermined age categories (2 and FEJ) now are NA-s in the new variable.

```
>mydata$age = ifelse(
  test = mydata$AGE == "1Y",
  yes  = "juv",
  no   = NA)
>mydata$age = ifelse(
  test = mydata$AGE == "1+" |
         mydata$AGE == "2Y" |
         mydata$AGE == "2+",
  yes  = "adult",
  no   = mydata$age)
```

Repeat this process with the SEX factor. N denotes NA in this case, so we recode N to NA:

```
>levels(mydata$SEX)

[1] "F" "M" "N"

>mydata$SEX = ifelse(
  test = mydata$SEX == "N",
  yes  = NA,
  no   = as.character(mydata$SEX))
>mydata$SEX = factor(mydata$SEX)
```

Contigency tables

Contingency tables provide counts of records in category combinations of two variables. The table() function can create contingency tables by giving two (or more) categorical variables in the arguments. Applying the prop.table() function to a contingency table yields relative frequencies of cell counts. These ratios can be calculated against row totals (margin = 1), column totals (margin = 2) and the grand total (default).

```
#contingency table
>table(mydata$age, mydata$SEX)
           F   M
   adult  95 179
   juv   522 761

#relative frequencies according to
#row totals
>prop.table(
  table(mydata$age, mydata$SEX),
  margin = 1)
                 F         M
   adult 0.3467153 0.6532847
   juv   0.4068589 0.5931411
#relative frequencies according to
#coloumn totals
>prop.table(
  table(mydata$age, mydata$SEX),
  margin = 2)
                 F         M
   adult 0.1539708 0.1904255
   juv   0.8460292 0.8095745
#relative frequencies
>prop.table(
  table(mydata$age, mydata$SEX))

                  F          M
   adult 0.06101477 0.11496468
   juv   0.33526012 0.48876044
```

Visual overview of categorical variables

Visual inspection of the number of records by factor levels (frequency distribution) is useful to discover interesting patterns. For a single factor the barplot() function creates a bar graph, while the mosaicplot() can be used to visualize the relationship of two categorical variables. barplot() should be used with the frequency, while

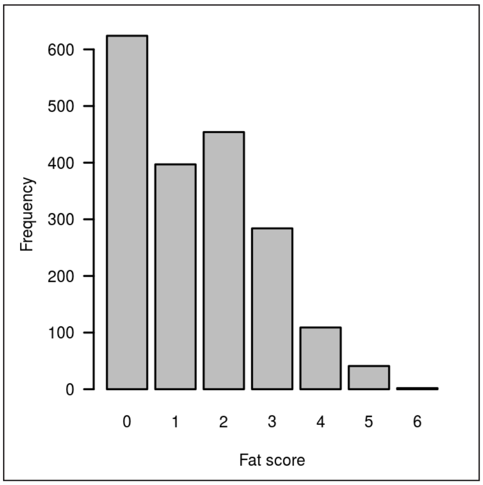

Figure 4. Barplot to visualize the number of records by factor levels
4. ábra Oszlopdiagram a faktorszintenkénti rekordszám bemutatására

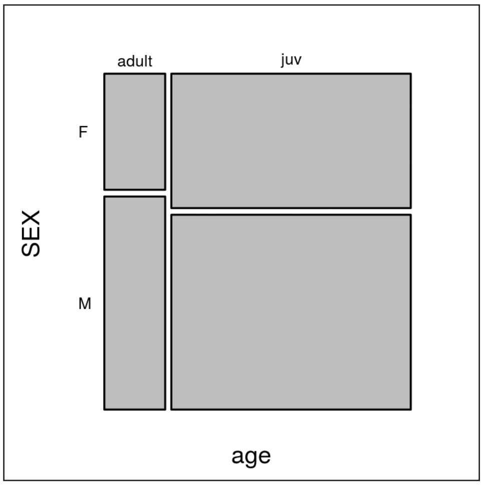

Figure 5. Mosaicplot to visualize the relationship of two categorical variables
5. ábra Mozaikábra két kategóriás változó kapcsolatának bemutatására

`mosaicplot()` with the contingency table. A mosaic plot is an area proportional visualization of a contingency table. It is composed of tiles (corresponding to the cells) created by recursive horizontal and vertical splits of a rectangle. The area of each tile is proportional to the corresponding cell entry (number of observations), given the dimensions of previous splits.

```
>barplot(
  table(mydata$FAT),
  xlab = "Fat score",
  ylab = "Frequency")
>with(mydata[mydata$RECAP == 0,],
  mosaicplot(table(age, SEX),
    main = ""))
```

We show the fat scores (`FAT`) of all observations in the dataset *(Figure 4)*. The `FAT` score is 0 for most observations, and the number of observations decreases with higher scores. In contrast, the mosaicplot *(Figure 5)* shows the proportions of the newly created age categories (horizontal split). Note that we used the `with()` function to create the plot only for a subset of the original dataset. The first argument of the `with()` is the row-wise selection of first captures, while the second argument is the `mosaicplot()` function call. Within the `mosaicplot()` call, only the variable names (without the `$` reference) are given. The rectangles of the age classes are further split vertically in proportion to the sex ratio: the majority of the observations are juveniles and sex ratios between age classes are similar, although slightly biased to males, indicating that sex and age are presumably independent of each other.

Checking and exploring numeric variables

Previously we have seen how the `summary()` call returns the basic descriptive statistics of the variables. It is often necessary to work with individual variables and specific statistics (`min()`, `max()`, `mean()`, `median()`,

```
>which(mydata$MASS > 20)

[1] 1240 1410

>xx = which(mydata$MASS > 20)
#List the cases
#First column: row numbers in the output
>mydata[xx,]

          ID      DATE RECAP    RING AGE SEX FAT MUSCLE MASS WING THIRD
 1240 244759  8/7/2004     0 T373846  1Y   M   2      2 81.0   81    61
 1367 355719 4/29/2006     0 T468033   2   F   0      2 23.2   78    60
      TAIL year yearday period.recap season  age
 1240   NA 2004     219            0 autumn  juv
 1367   54 2006     118            0 spring <NA>
```

Box 8. Listing the row numbers of the cases and the whole records where the `MASS` is above 20 grams
8. doboz Az esetek száma és a teljes rekordok kilistázása `MASS` > 20 gramm esetén

`sd()` and `var()`) can be used similarly. For instance we wish to check the range (minimum and maximum values) of the `MASS` variable.

```
>range(mydata$MASS, na.rm = TRUE)
[1] 0 81
```

Surprisingly, there are records of birds with 0 mass in the dataset. We can also count the number of records where mass is 0 with the `nrow()` function. Quite often missing values are coded with 0 instead of a specific character string. The problem is that the 0 will be recognized as a valid measurement and hence influence all derived descriptive statistics, models etc. Changing these values to `NA`-s is essential and easy with the `ifelse()` function. We highly recommend avoiding the use of 0-s instead of `NA`-s (or `null`-s in case of databases) in your own data tables.

```
>nrow(mydata[mydata$MASS == 0,])
[1] 80
>mydata$MASS =
  ifelse(
    test = mydata$MASS == 0,
    yes  = NA,
    no   = mydata$MASS)
```

Note that in case of a numeric variable, we do not use the `as.character()` in `ifelse()`. The upper limit of the range of the `MASS` variable is also troubling. Body mass of Pied Flycatchers seldom exceeds 20 g, thus 81 g is practically impossible. List the row numbers of the cases which are above 20 grams with the `which()` function. We save these numbers to a vector called `xx` *(Box 8)*.

We recommend listing the whole records with suspicious values to be able to check all other measurements and characteristics belonging to the same capture event. In this case we can immediately recognise that the `WING` length value was recorded as `MASS`.

Suspicious and obviously mistyped data should be checked and corrected, or – if we cannot correct it – should be excluded from further analyses. However, we strongly recommend recording and documenting all exclusions. In this case, checking the ringing notes revealed that the suspicious values were in fact data entry errors. The correct values are 15.5 and 13.2, respectively, and the data frame should be corrected accordingly. Change these simply by assigning the new values individually (if a value

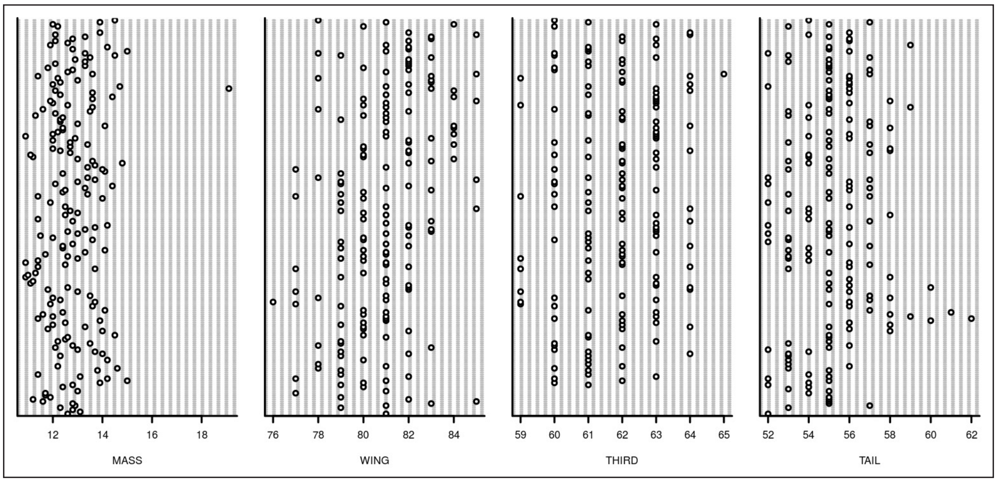

Figure 6. Cleveland dotplots of the `MASS`, `WING`, `THIRD` and `TAIL` variables for the adult male birds in spring

6. ábra Cleveland pontábrák a `MASS`, `WING`, `THIRD` és `TAIL` változókra tavasszal, az öreg hím madarak esetén

should be deleted, type `NA` instead). We can read the IDs of the observations from the previous output.

```
>mydata[mydata$ID ==
         244759,]$MASS = 15.5
>mydata[mydata$ID ==
         355719,]$MASS = 13.2
```

Note, that the safest method to change a certain value in a specific record is using the record identifier to select it. Consider also the rest of the numeric variables: `WING`, `THIRD` and `TAIL`. In all cases zeros should be changed to `NA`-s, and should be checked similarly (see online appendix).

Plotting the values

Although we have already cleaned our numerical values of obvious problems, there may still be errors or suspicious values remaining. Using graphical tools can be rewarding in identifying such records.

Cleveland dotplots

One of the most useful visualization tools are Cleveland dotplots (Cleveland 1993). Here, the row number of an observation (vertical axis) is plotted versus its value (horizontal axis). This graph should be prepared for relatively small subsets of the data and it is useful to make this graph for several variables simultaneously. We plot the `MASS`, `WING`, `THIRD` and `TAIL` values of the spring observations *(Figure 6)*. For shortening the code, we first create a subset of the spring male records. Note that we use the `par(mfrow = c(1,4))` function to get four graphs in a row.

```
>mydata.spring.M = mydata[
  mydata$season == "spring" &
    mydata$SEX == "M",]
#Creating multiple Cleveland dotplots
>par(mfrow = c(1,4))
>with(mydata.spring.M,
  dotchart(MASS, xlab = "MASS"))
>with(mydata.spring.M,
  dotchart(WING, xlab = "WING"))
>with(mydata.spring.M,
  dotchart(THIRD, xlab = "THIRD"))
>with(mydata.spring.M,
  dotchart(TAIL, xlab = "TAIL"))
>par(mfrow = c(1,1))
```

A suspicious value can be identified on the `MASS` plot. Since it is the only value above 18 grams, and the rest of the measurements of this bird are not out of range, the suspicious value have to be found and checked *(Box 9)*. Checking the ringing notes showed that this value is valid, so we leave it in the dataset. Note that there are suspicious values on the other plots. You can find the codes of the detailed inspection and correction of these in the online appendix.

The distribution of the values

In the next step, we check the distribution of the `MASS` variable. The distribution of a variable is essentially how the values are spread across its range. Visually exploring distributions allows the analyst to observe the shape (e.g. skewness, modality), the range and the most frequently observed values. There are several graphical tools to visualize distributions, all grasping different aspects of the data. Here we describe the most commonly used types.

Histogram

The range of values is partitioned into several adjacent and nonoverlapping intervals (bins) and the number of observed values in each interval (frequency) or its ratio to the total number of observations (relative frequency) are recorded. If we draw a rectangle above each interval with an area proportional to the frequency or relative frequency (set `freq = F` in the `hist()` function) of the interval, the resulting graph is called histogram. The `main = "Males"` argument creates the title of the histogram. Distributions should be investigated in each major group separately. We show the graphs for the first captures (`RECAP == 0`) of the juvenile male and female groups collected during the autumn migration. In this case, the conditions are connected with `&` (AND) operator which means that all the conditions must be true simultaneously. When visualizing more than one group of observations, it is essential to scale the corresponding axes and the number of bins identically in the two graphs. Here we used the generic arguments `xlim`, `ylim` and `breaks` to set the limits of the axes' range and the number of bins. The number of bins influences the appearance of the plot considerably and often the default settings have to be adjusted. We advise to try several settings.

```
>xx = which(mydata[
  mydata$season == "spring",]$MASS > 18)

>mydata[mydata$season == "spring",][xx,]

          ID      DATE RECAP   RING AGE SEX FAT MUSCLE MASS WING THIRD TAIL
 1469 500595 4/19/2007    NA 8E1095  2Y   M   2      2 19.1   83    63   56
      year yearday period.recap season   age
 1469 2007     108            0 spring adult
```

Box 9. Listing row number and record where `MASS` is above 18 grams
9. doboz Sorszám és teljes rekord `MASS`>18 gramm esetén

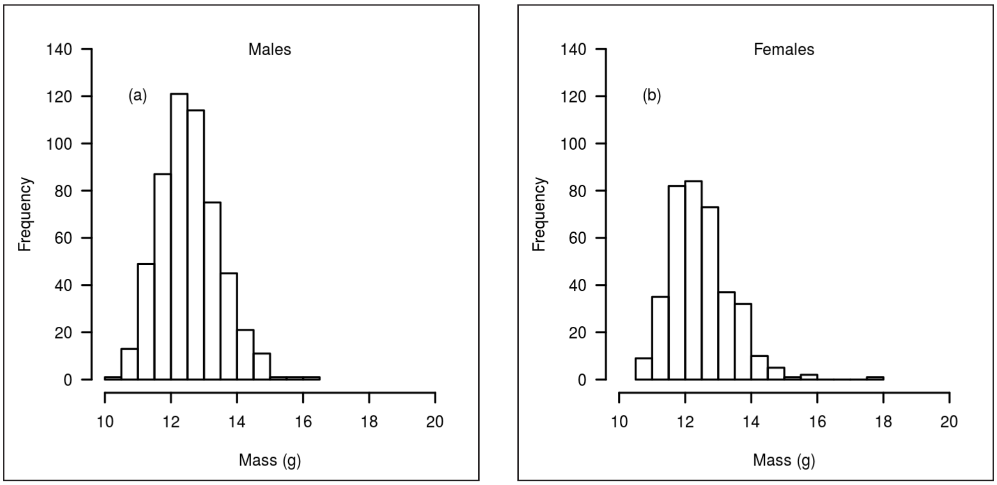

Figure 7. Histogram of `MASS` in case of juvenile male (a) and female (b) birds captured in autumn
7. ábra A `MASS` változó hisztogramja az ősszel megfogott fiatal hímek (a) és tojók (b) esetén

```
>with(mydata[
  mydata$RECAP  == 0 &
  mydata$age    == "juv" &
  mydata$SEX    == "M" &
  mydata$season == "autumn",],
  hist(MASS, main = "Males",
    xlim   = c(10,20),
    ylim   = c(0,140),
    breaks = 20,
    xlab   = "Mass (g)"))
>with(mydata[
  mydata$RECAP  == 0 &
  mydata$age    == "juv" &
  mydata$SEX    == "F" &
  mydata$season == "autumn",],
  hist(MASS, main = "Females",
    xlim   = c(10,20),
    ylim   = c(0,140),
    breaks = 20,
    xlab   = "Mass (g)"))
```

The two histograms *(Figure 7a, b)* show that slightly more males were observed than females as we have previously seen on the mosaicplot *(Figure 5)*. Also notice that the shape of both distributions are similar, both are unimodal (one peak) and more or less symmetrical. There are also a couple of odd, relatively large values on the right hand side of both histograms. These should be handled with caution but bear in mind that when measuring body mass, gut and crop load can have a substantial effect on individual values. These values are within the range of values acceptable for the species. There is no indication that other factors are causing this pattern.

Smoothed histogram

Another useful plot is the smoothed histogram. This is frequently more informative than the traditional version. With this technique small curve segments are calculated for each observation and these segments are then added up resulting in a smooth function (Ieno & Zuur 2015). We can plot individual values on the x axis with the `rug = TRUE` (default) parameter. We use the `densityPlot()` function of the `car` package (Fox & Weisberg 2011) and plot the density of `MASS` in the two sexes separately with the `MASS ~ SEX` formula for the captures of juveniles in autumn. A useful argument

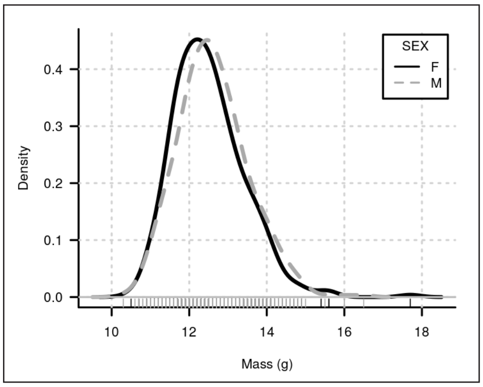

Figure 8. Smoothed histograms of `MASS` of juvenile birds captured in autumn
8. ábra A `MASS` változó simított hisztogramja az ősszel megfogott fiatal madarak esetén

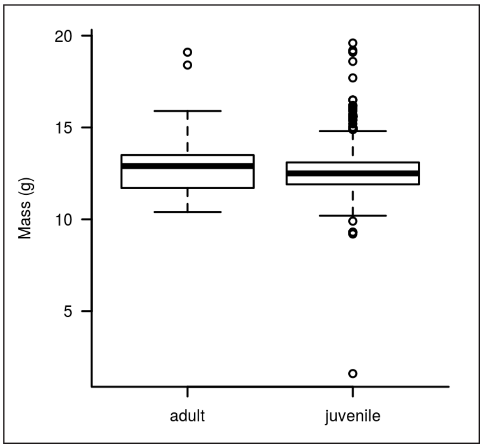

Figure 9. Boxplots of `MASS` by age groups
9. ábra A `MASS` változó korcsoportonkénti boxplotjai

that may be adjusted is the bandwidth (`bw`) parameter which controls the amount of smoothing, please refer to the help page of `densityPlot()`.

```
>library(car)
>with(mydata[mydata$RECAP == 0 &
  mydata$age == "juv" &
  mydata$season == "autumn",],
  densityPlot(MASS ~ SEX,
    xlab = "Mass (g)"))
```

Creating the smoothed histograms of males and females on the same plot reveals *(Figure 8)* that there may be a slight difference in body mass in the two groups. This would have been more difficult to observe using the traditional histograms.

Boxplot

The next useful graph is the boxplot (box and whiskers plot): the bottom and top of the box are the first and third quartiles and the band inside the box is the median. The ends of the whiskers may represent different values. The default in R: the lowest value is still within 1.5 IQR (Interquartile range: `3rd Qu.-1st Qu.`) of the lower quartile, and the highest value is still within 1.5 IQR of the upper quartile. The values outside the

```
>which(mydata$MASS < 5)

[1] 1704

>mydata[which(mydata$MASS < 5),]

         ID     DATE RECAP   RING AGE SEX FAT MUSCLE MASS WING THIRD TAIL
 1708 583178 9/7/2009     1 W59225  1Y   M   2      2  1.6   NA    NA   NA
      year yearday period.recap season age
 1708 2009     249            1 autumn juv
```

Box 10. Listing row number and record where `MASS` is less than 5 grams
10. doboz Sorszám és teljes rekord `MASS<5` gramm esetén

whiskers are suspicious points. Here we depict `MASS` by age groups *(Figure 9)*.

```
>with(mydata[
  mydata$season == "autumn",],
  boxplot(MASS ~ age,
    ylab  = "Mass (g)",
    names = c("adult", "juvenile")))
```

Check the obviously too small value in the juvenile male group *(Box 10)*.

Checking the ringing notes revealed that the decimal point was misplaced. Correct and recreate the figure but this time group the data by the `AGE` and `SEX` variables *(Figure 10)*. Note that with the `varwidth = TRUE` setting, the width of the boxes are proportional to the square-roots of the subgroup sample sizes.

```
>mydata[mydata$ID ==
        "583178",]$MASS = 16
>with(mydata[mydata$season ==
        "autumn",],
  boxplot(MASS ~ age:SEX,
    ylab     = "Mass (g)",
    varwidth = TRUE))
```

Final steps

Checking, screening and cleaning your data is a process of repeated cycles. Each time you make a decision on changing a value, it is strongly advised to rerun your codes to see how it altered the entire data structure. Saving your changes is essential, however we advise to create multiple versions of your dataset (e.g. by using version numbers in the file name), instead of overwriting the same file over and over again. Saving a text file from R can be done with the `write.table()` function. The first parameter is the data frame to be saved, then we have to specify the name of the file (`file`), the field separator (`sep`) and the decimal

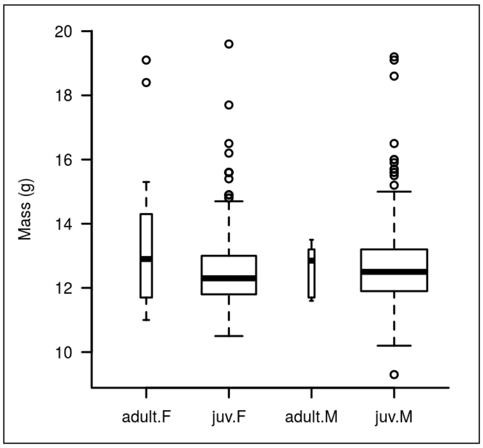

Figure 10. Boxplots of `MASS` by age and `SEX` groups in autumn
10. ábra A `MASS` változó boxplotjai az őszi kor- és ivarcsoportonként

symbol (`dec`). Use the `quote = FALSE` setting to avoid surrounding the character or factor values by double quotes, and the `row.names = FALSE` setting to avoid writing the row names into the output file.

```
>write.table(mydata,
  file      = 'FICHYP_corr_1.csv',
  sep        = ';',
  dec        = '.',
  quote      = FALSE,
  row.names = FALSE)
```

Concluding remarks

The quality of your results substantially depends on having error free and consistent data suitable to analyse. Every data set is different, therefore the methods described in this tutorial will not fit perfectly to all of them. However, the logic of data cleaning presented here may serve as a guideline in how to carry out this task. We plan to maintain a continuous development of the `ringR` package, therefore we welcome bug reports, comments and ideas on development to the corresponding author's address.

Acknowledgments

The authors express their gratitude for the work of all the volunteers who collected data at the Ócsa Bird Ringing Station throughout the years. We are grateful for Márton Demeter for helping in compiling the scripts for the `ringR` package. We also thank our colleagues – especially János Kis and Imre Sándor Piross – who helped us improve earlier versions of this manuscript. This work was supported by OTKA under Grant No. 108571.

References

Allaire, J. J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., Wickham, H. & Hyndman, R. 2015. rmarkdown: Dynamic documents for R. R package version 0.5.1. – http://CRAN.R-project.org/package=rmarkdown

American Statistical Association 1999. Ethical guidelines for statistical practice. – Alexandria (Virginia): American Statistical Association. Available: http://www.amstat.org/profession/index.cfm?fuseaction=ethicalstatistics

Brown, M. & Oschadleus, D. 2008. The ongoing role of bird ringing in science – a review. – In: Harebottle, D. M., Craig, A. J. F. K., Anderson, M. D., Rakotomanana, H. & Muchai, M. (eds.) Proceedings of the 12th Pan-African Ornithological Congress, pp. 6–8.

Cleveland, W. S. 1993. Visualizing data. – Hobart Press, Summit, NJ., pp. 346

Crewe, T. L., McCracken, J. D., Taylor, P. D., Lepage, D. & Heagy, A. E. 2008. The Canadian Migration Monitoring Network-Réseau canadien de surveillance des migrations: ten-year report on monitoring landbird population change. – CMMN-RCSM Scientific Technical Report No. 1. Bird Studies Canada, Port Rowan, Ontario, pp. 69

Dearborn, D. C. & Kark, S. 2010. Motivations for conserving urban biodiversity. – Conservation Biology 24(2): 432–440. DOI: 10.1111/j.1523-1739.2009.01328.x

de Jonge, E. & van der Loo, M. 2013. An introduction to data cleaning with R. – Statistics Netherlands, The Hague, pp. 53.

Dingemanse, N. J., Both, C., Van Noordwijk, A. J., Rutten, A. L. & Drent, P. J. 2003. Natal dispersal and personalities in Great Tits *(Parus major)*. – Proceedings of the Royal Society of London B: Biological Sciences 270(1516): 741–747. DOI: 10.1098/rspb.2002.2300

Dragulescu, A. A. 2014. xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files. R package version 0.5.7. – http://CRAN.R-project.org/package=xlsx

Elliott, A. C., Hynan, L. S., Reisch, J. S. & Smith, J. 2006. Preparing data for snalysis using microsoft excel. – Journal of Investigative Medicine 54(6): 334–341. DOI: 10.2310/6650.2006.05038

Fox, J. 2005. The R Commander: A basic-statistics graphical user interface to R. – Journal of Statistical Software 19(9): 1–42. DOI: 10.18637/jss.v014.i09

Fox, J. & Weisberg, S. 2011. An {R} Companion to applied regression. 2nd ed. – Thousand Oaks CA. Sage, pp. 472

Fowler, J. & Cohen, L. 1996. Statistics for ornithologists. – British Trust for Ornithology, pp. 155

Freedman, D., Pisani, R. & Purves, R. 2007. Statistics. 4th ed. – W. W. Norton and Company, inc. New York, pp. 697

Greenwood, J. J. D. 2009. 100 years of ringing in Britain and Ireland. – Ringing & Migration 24(3): 147–153. DOI: 10.1080/03078698.2009.9674385

Harnos, A., Ágh, N., Kovács, Sz., Lang, Zs. & Csörgő, T. 2015a Increasing protandry in the spring migration of the Pied Flycatcher *(Ficedula hypoleuca)* in Central Europe. – Journal of Ornithology 156(2): 543–546. DOI 10.1007/s10336-014-1148-3

Harnos, A., Lang, Zs., Fehérvári, P. & Csörgő, T. 2015b Sex and age dependent migratory phenology of the Pied Flycatcher in a stopover site in the Carpathian Basin. – Ornis Hungarica 23(2): 10–19. DOI: 10.1515/orhu-2015-0010

Ieno, E. N. & Zuur, A. F. 2015. A beginner's guide to data exploration and visualisation with R. – Highland Statistics Ltd. Newburgh, pp. 160

Kovács, Sz., Csörgő, T., Harnos, A., Fehérvári, P. & Nagy, K. 2011. Change in migration phenology and biometrics of two conspecific *Sylvia* species in Hungary. – Journal of Ornithology 152(2): 365–373. DOI: 10.1007/s10336-010-0596-7.

Kovács, Sz., Csörgő, T., Harnos, A., Fehérvári, P. & Nagy, K. 2012. Changes in migration phenology and biometrical traits of Reed, Marsh and Sedge Warblers. – Open Life Sciences 7(1): 115–125. DOI: 10.2478/s11535-011-0101-1

Nowakowski, J. 2003. Catch numbers at ringing stations is a reflection of bird migration intensity, as exemplified by autumn movements of the Great Tit *(Parus major)*. – Ring 25(1–2): 3–15. DOI: 10.2478/v10050-008-0070-6

Osenkowski, J. E., Paton, P. W. C. & Kraus, D. 2012. Using long-term constant-effort banding data to monitor population trends of migratory birds: a 33-year assessment of adjacent coastal stations. – The Condor 114(3): 470–481. DOI: 10.1525/cond.2012.110169

Peach, W. J., Baillie, S. R. & Balmer, D. E. 1998. Long-term changes in the abundance of passerines in Britain and Ireland as measured by constant effort mist-netting. – Bird Study 45(3): 257–275. DOI: 10.1080/00063659809461098

Peach, W. J., Baille, S. R., & Buckland, S. T. 2004. Current practices in the British Trust for Ornithology Constant Effort Sites Scheme and comparisons of temporal changes in mist-net captures with changes in spot-mapping counts at the extensive scale. – Studies in Avian Biology. 29: 46–56.

Pilastro, A., Macchio, S., Massi, A., Montemaggiori, A. & Spina, F. 1998. Spring migratory routes of eight trans-Saharan passerines through the central and western Mediterranean; results from a network of insular and coastal ringing sites. – Ibis 140(4): 591–598. DOI: 10.1111/j.1474-919X.1998.tb04704.x

Preuss, N. O. 2001. Hans Christian Cornelius Mortensen: aspects of life and of the history of bird ringing. – Ardea 89(special issue): 1–6.

R Core Team 2015a R: A language and environment for statistical computing. – R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/.

R Core Team 2015b R Data Import/Export. – R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/.

Reiczigel, J., Harnos, A. & Solymosi, N. 2014. Biostatisztika nem statisztikusoknak [Biostatistics for non statisticians]. – Pars Kft., Budapest, pp. 462. (in Hungarian)

Robinson, R. A., Baillie, S. R. & Crick, H. Q. 2007. Weather dependent survival: implications of climate change for passerine population processes. – Ibis 149(2): 357–364. DOI: 0.1111/j.1474-919X.2006.00648.x

Robinson, R. A., Julliard, R. & Saracco, J. F. 2009. Constant effort: studying avian population processes using standardised ringing. – Ringing and Migration 24: 199–204. DOI: 10.1080/03078698.2009.9674392

Shahbaba, B. 2012. Biostatistics with R. – Springer-Verlag, New York, pp. 352, DOI:10.1007/978-1-4614-1302-8

Şekercioğlu, Ç. H. 2012. Promoting community-based bird monitoring in the tropics: conservation, research, environmental education, capacity-building, and local incomes. – Biological Conservation 151(1): 69–73. DOI: 10.1016/j.biocon.2011.10.024

Spector, P. 2008. Data manipulation with R. – Springer-Verlag, New York, pp. 154, DOI: 10.1007/978-0-387-74731-6

Spina, F. 1999. Value of ringing information for bird conservation in Europe. – Ringing & Migration 19(1): 29–40. DOI: 10.1080/03078698.1999.9674209

Thorup, K., Korner-Nievergelt, F., Cohen, E. B. & Baillie, S. R. 2014. Large-scale spatial analysis of ringing and re-encounter data to infer movement patterns: A review including methodological perspectives. – Methods in Ecology and Evolution 5(12): 1337–1350. DOI: 10.1111/2041-210X.12258

Van den Broeck, J., Argeseanu Cunningham S., Eeckels, R. & Herbst, K. 2005. Data cleaning: detecting, diagnosing, and editing data abnormalities. – PLoS Medicine 2(10): e267. p. 0966–0970. DOI: 10.1371/journal.pmed.0020267

Venables, W. N., Smith, D. M. & the R Core Team 2015. An introduction to R. pp. 99 – https://cran.r-project.org/

Walther, B. A., Schäffer, N., Van Niekerk, A., Thuiller, W., Rahbek, C. & Chown, S. L. 2007. Modelling the winter distribution of a rare and endangered migrant, the Aquatic Warbler *Acrocephalus paludicola*. – Ibis 149(4): 701–714. DOI: 10.1111/j.1474-919X.2007.00690.x