Best Practices in Research Data Management

Hauke Sonnenberg & Michael Rustler

2018-09-21 11:46:09

Contents

Preface **5**

1 Introduction **7**
1.1 Why RDM ? . . . 7
1.2 For Whom ? . . . 7

2 Best Practices **9**
2.1 Plan and Fund a New Project . . . 9
2.2 Data Storage . . . 10
2.3 Metadata . . . 19
2.4 Data Collection . . . 22
2.5 Data Publishing and Sharing . . . 23

3 Case Studies **29**
3.1 Geogenic Salination . . . 29
3.2 LCA Modelling . . . 29
3.3 Pilot Plants . . . 30

4 Other Projects **33**
4.1 Spree2011 (2007) . . . 33
4.2 MIACSO (2009) . . . 33
4.3 KURAS . . . 34
4.4 OGRE . . . 35
4.5 Flusshygiene . . . 35
4.6 DEMEAU . . . 35
4.7 DEMOWARE . . . 35
4.8 OPTIWELLS . . . 35
4.9 RWE . . . 36

5 Literature Review **37**
5.1 Data Storage . . . 37

6 FAQs **39**
6.1 Naming . . . 39
6.2 Tools for Researchers . . . 39
6.3 Writing More Robust R Code . . . 40
6.4 Learning R on DataCamp . . . 40
6.5 Using Subversion at KWB . . . 41
6.6 How to Build Your Own Kwb Styled R Package? . . . 41
6.7 How to Install KWB R Packages? . . . 41
6.8 Working with Excel . . . 41
6.9 Heterogenous Software Versions on KWB Computers . . . 43
6.10 R Package/Version Dependency of R Scripts . . . 43

6.11 Complex R Script Dependencies . . . 44
6.12 Heterogenous Coding Styles . . . 44
6.13 Collaborative Version Control . . . 45
6.14 Workflow Automation . . . 45
6.15 Encoding . . . 46
6.16 Collaborative Writing . . . 46
6.17 Exchanging Data with Colleagues/Project Partners . . . 47

7 Glossary **49**
7.1 Reproducibility . . . 49
7.2 Provenance . . . 49
7.3 Techniques . . . 49
7.4 Tools . . . 50

Preface

> "These days, data trails are often a morass of separate data and results and code files in which no one knows which results were derived from which raw data using which code files."
>
> — Professor Charles Randy Gallistel, Rutgers University

This document is the outcome of the KWB project FAKIN (Forschungsdatenmanagement an kleinen Instituten = research data management at small institutes).

It defines best practices for research data management. It is mainly based on the personal experiences of

the authors having worked in many different research projects at KWB.

The document is outlined as follows:

- Chapter 1 explains why and for whom research data management is important
- Chapter 2 defines Best Practices for different topics.
- Chapter 3 gives an overview on three case studies with different data related tasks that have been solved within the FAKIN project, while
- Chapter 4 what data challenges occured in the past in other KWB projects and how they were solved in order to increase our awareness at KWB,
- Chapter 5 provides FAQs and finally
- Chapter 6 is a glossary for explaining commonly used terms within this document.

This document is assumed to be a "living" document. We highly appreciate any comments and suggestions for improvements. What are your experiences with research data managment tasks? Can you provide solutions for specific tasks?

Chapter 1

Introduction

A very good book on how research data management (RDM) improves reproducibility is provided by Kitzes et al. (2018) which also is available online at http://practicereproducibleresearch.org/.

1.1 Why RDM ?

Doing research data management (RDM) is useful according to Labfolder.com (2018) as it:

- Saves time for research
- Avoids risk of data loss
- Ensures transparency and reproducibility
- Increases data visibility and number of citations
- Fulfill funders' requirements and receive more grants
- Produce new knowledge and make more discoveries just by re-using data
- Archive, retrieve and re-use own data

Recommended literature:

- Introduction to research data management in the geosciences (in German!)
- Handbook research data management (in German!) (Buettner et al., 2011)

1.2 For Whom ?

Info on the document – create links between projects and thematic groups

Chapter 2

Best Practices

2.1 Plan and Fund a New Project

2.1.1 Data Management Plan

Action 1: Look through a data management planning checklist DMP checklist?

Action 2: Create a data management plan. You can use DMP Online as it has a template or if required, use a funder's format.

RDM checklist before starting with a new project Brauchen wir eine Data Management Plan? See e.g. https://dmponline.dcc.ac.uk/

2.1.2 Electronic Lab Notebooks

Electronic Lab Notebooks (ELNs) https://www.labfolder.com/research-data-management/ ELNs are an essential asset for researchers to fulfil any requirements for data management, and they create direct bridges between scientists and stakeholders. By adopting an ELN, the data lifecycle can proceed smoothly and easily: from creating and collecting data digitally in one place to one-click data archiving, ELNs empower researchers by allowing them to implement their RDM plan without effort and time investment.

An ELN to make your data FAIR? Welches ist das?

2.1.3 Define Acronyms

Acronyms are unique, clear names for objects. They should

- be short but meaningful and easy to remember,
- be all lowercase,
- consist of only alphanumeric letters (`a-z`, `0-9`) or the hyphen (`-`).

Acronyms for Projects

Do we already have rules for naming and spelling of projects and project folders or files?

If not, we should, at the start of a project, define official project acronyms. These are uniform spellings of project names, intended to be used in folder or file names.

Whenever we want to indicate the relation to a certain project in a file or folder name, we use the project acronym in exactly the typing that was defined. This is important as we want to distinguish between raw data, processed data and project results in our data workflow.

The project acronyms are defined in a simple text file `PROJECTS.txt` in the `//server/projects$` folder, see Project Folder Structure.

Should we also define three letter codes for projects? That could be beneficial in tables with each column representing a project or in overview plots.

Acronyms for Organisations

It is very important to know the origin or owner of data. This is an important piece of metadata Therefore we define unique acronyms for the owners of data that we use. The acronyms are defined in a special file `ORGANISATIONS.txt`

At the start of a research project

- Choose a project acronym and store it in `PROJECTS.txt`.
- Check if the organisations that you expect to get data from are listed in `ORGANISATIONS.txt` and extend this file if necessary.
- Create a subfolder for your project and subfolders for the organisations in the rawdata folder structure.

At the start of a project or if an employee or trainee enters the project

- Give an introduction to our research data management as described in this document.

Regularly during the project

- Check if the folder structure within your project's rawdata subfolder still complies with the rawdata folder structure and clean the structure, if not.

2.2 Data Storage

This chapter describes

- where to store data files, see Where to Store Data,
- what names to choose for files and folders, see Naming of Files and Folders,
- how to group files in folders, see Folder Structures.

With the following recommendations we intend to comply with the Ten Simple Rules for Digital Data Storage.

2.2.1 Where to Store Data

We want to store project relevant files on a KWB network drive that is accessible to all the persons working in the project.

We want to avoid that project relevant files are stored

- locally on the hard drive of an employee or trainee or
- on the personal network drive assigned to an employee or trainee.

2.2.2 Naming of Files and Folders

In this section we define rules for good names for files and folders. We want to apply these rules whenever we have the freedom to define our own names, i.e. when creating a new file or a new folder.

We pursue different objectives:

- *Objective 1:* Names should not cause problems during automated data processing.
- *Objective 2:* Names should be meaningful to you and your colleagues. You and they should be able to guess what a folder or file contains so that required files are found easily.
- *Objective 3:* Names should not cause problems when copying files within your (computer) system or between different systems, especially between different operating systems or systems in different countries.

We process files automatically. Listing or reading files with non-standard characters in their names often cause problems. These problems are easy to avoid by sticking to some simple restrictive rules when naming files and folders.

Allowed Characters

In order to meet *Objectives 1 and 3* we define the following **set of characters** that are allowed in file or folder names:

- upper case letters `A-Z`,
- lower case letters `a-z`,
- numbers `0-9`,
- underscore `_`,
- hyphen `-`,
- dot `.`.

See the FAQ chapter for answers to the following questions:

- Why are special characters not allowed?
- Why is the space character not allowed?

Instead of German special characters (ä, ö, ü, Ä, Ö, Ü, ß) use the following substitutions: `ae`, `oe`, `ue`, `Ae`, `Oe`, `Ue`, `ss`.

Instead of space, use underscore `_` or hyphen `-` (see next).

Separation of Words or Parts of Words

Use underscore to separate different words in the file or folder name that contain different types of information:

- `results_today`, `results_tomorrow` instead of `results-today`, `results-tomorrow`,
- `protocol_hauke`, `protocol_michael` instead of `protocol-hauke`, `protocol-michael`

Use hyphen `-` instead of underscore `_` to visually separate the parts of compound words or names:

- `site-1` instead of `site_1`,
- `dissolved-oxygen` instead of `dissolved_oxygen`,
- `clean-data` instead of `clean_data`.

Use hyphen (or no separation at all) in dates (see below).

Using hyphen instead of underscore in composed words will not split the composed words into their parts when splitting a file or folder name at underscore.

For example, splitting the name `project-report_example-project-1_v1.0_2018-07-02` at underscore results in the following words (giving different types of information on the file or folder)

- `project-report` (type of document),
- `example-project-1` (name of related project),
- `v1.0` (version number),
- `2018-07-02` (version date).

Capitalisation

Should we decide on if and when to use capitals, i.e. should we allow only one of the following spellings:

`dissolved-oxygen` (all lower case), `dissolved-Oxygen` (attributes lower case, nouns upper case), `Dissolved-oxygen` (first letter upper case), `Dissolved-Oxygen` (all parts of compund words upper case)?

Formatting of Dates and Numbers

We want to write **dates** in file or folder names in one of these forms:

- `yyyy-mm-dd` (e.g. `2018-06-28`)
- `yyyymmdd` (e.g. `20180628`)

By doing so, file names only differing in the date will display chronologically in file listings. This is because, by default, files are sorted alphanumerically by their name. Using the first form improves the visual distinction of the year, month and day part of the date. Using hyphen instead of underscore will keep these parts together when splitting the name at underscore (see above).

When using numbers in file or folder names to bring them into a certain order we should use leading zeroes and specify the amount of digits that will be used, e.g. `01`, `02`, `03` or `001`, `002`, `003`, etc.

Allowed Words

In order to meet *Objective 2*, we should define **sets of allowed words** in so called vocabularies. Only words from the vocabularies are then expected to appear as words in file or folder names. Getting accustomed to the words from the vocabularies and their meanings allows for more precise file searching.

This is most important to clearly indicate that a file or folder relates to "special objects", such as projects or organisations or monitoring sites. At least for projects and organisations we want to define vocabularies in which "official" acronyms are defined for all projects and all organisations from which we expect to receive data (see the chapter on acronyms). Always using the acronyms defined in the vocabularies allows to search for files or folders belonging to one specific project or being provided by one specific organisation.

We could also define vocabularies of words describing other properties of a file or folder. We could e.g. decide to always use `clean-data` instead of `data-clean`, `cleaned-data`, `data-cleaning`, `Datenbereinigung`, `bereinigte-daten`, etc.

Order of Words

We could go one step further and define the order in which we expect the words to appear in a file or folder name. Which types of information should go first in the filename? The order of words determines in which way files are grouped visually when being listed by their name. If the acronym of the organisation goes first, files are grouped by organisation. If the acronym of the monitoring site goes first, files are grouped by monitoring site.

Allowed Languages

Do not mix words from different languages within one and the same file or folder name. For example, use `regen-ereignis` or `rain-event` instead of `regen-event` or `rain-ereignis`.

Within one project, use either only English words or only German words in file or folder names. This restriction may be too strict. However, I think that we should follow this rule at least for the top level folder structures. It is not nice that we see folders `AUFTRAEGE` (German) and `GROUNDWATER` (English) as folder names within the same parent folder.

Avoid Very Long Names

In order to meet *Objective 3*, very long file and folder names should be avoided. File and folder names should be

- as long as necessary to be meaningful but at the same time
- as short as possible.

In the Windows operating system, long file paths can cause problems when copying files. If the full path to the target location of a file exceeds a certain length (between 255 and 260 characters) you get the following error message (here in German): `Die Dateinamen wären zu lang für den Zielordner. Kürzen Sie die Dateinamen und wiederholen Sie den Vorgang, oder verwenden Sie einen anderen Ort, der einen kürzeren Pfad hat.`

This problem can be avoided by

- avoiding very long file or folder names,
- avoiding very deep folder structures.

If folder or file names are generated by software (e.g. logger software, modelling software, reference manager) we should check if the software allows to modify the naming scheme.

If we nevertheless have to deal with deeply nested folder structures and/or very long file or folder names we should store them in a flat folder hierarchy (i.e. not in `\\server\projekte$\department-name\projects\project-name\dat`

Examples

Examples for good file names:

- `AHRC_TechnicalApp_Response_20120925.docx` or
- `AHRC-TechnicalApp-Response20120925.docx`

Example for bad file names:

- `what we got back from funders about the data stuff.docx`

2.2.3 Raw Data

Especially in environmental sciences, raw data often cannot be reproduced (e.g. rainfall, river discharge measurements) and are therefore of high value. Thus the following rules apply in case of raw data:

- can be renamed (in case this is documented in a metadata file)
- the content must not be changed
- will be stored as read-only in an secure area (i.e. one folder contains all raw data)

added from: 04_rawdata.Rmd

As raw data we define data that we receive from a device or from a project partner.

Most of our research results are based on data. We acknowledge the importance of raw data by

- storing then in a special place where it is specially secured
- describing them with metadata

Rawdata are stored in the rawdata folder structure

2.2.4 Data Workflow

For each project separation of:

- raw data (i.e. `\\server\\rawdata`)
- data processing (i.e. `\\server\\processing`)
- results (i.e. `\\server\\projekte$`)

Raw data are highly valuable because they if necessary cannot or only with high costs be reproduced. Thus raw data will be compiled on the highest level (i.e. `\\server\\rawdata`) for each `project` and `origin` (e.g. KWB, BWB) as shown below:

Raw data folder structure:

```
TestProjekt
  BWB
    Regen
      METADATEN
      regen.xls
    Labor
      METADATEN
      labor.xls

  KWB
    Durchfluss
      METADATEN
      q01.csv
      q02.csv
      q03.csv
```

The data processing is a "playground", where different approaches or scenarios can be tested and where possibly different versions for a specific approach or scenario are available. The files will be stored here according to the corresponding `project` and `topic` and the data processing step, respectively.

Data processing folder structure:

```
TestProjekt
  01_Bereinigung
    METADATEN
    regen_roh.csv
    regen.csv
    qualitaet.csv
    durchfluss.csv
  02_Modellierung
    sommer
    winter
    VERSIONEN
      v0.1
      v1.0
        sommer
        winter
  Software
```

In the results folder only the "reporting relevant" results should be saved. Relevant results files (e.g. plots) should be added according to `project` and `project structure` (i.e. per `working package`) in form of links from the `data processing` folder structure (see above).

Results folder structure:

```
TestProjekt
  Data-Work Packages
    WP1_Monitoring
    WP2_Modellierung
      sommer.lnk
      winter.lnk
```

2.2.5 Folder Structures

Restrictions/Conventions:

- Each top-level folder should represent a project, i.e. should be defined in the top level file `PROJECTS.txt`.
- Each possible owner should be defined in the top level file `ORGANISATIONS.txt`.
- The naming convention for the organisations is the same as for projects.

s)

2.2.5.1 Project Folder Structure

We said that we want to concentrate on the folder structures within the project folders. Nevertheless, we would like to give a recommendation on how the project folders could be organised within its top level folder. In this structure, there are no subfolders for the different departments any more.

```
//server/projects$
- PROJECTS.txt
- project-1/
- project-2/
- project-3/
```

In the `projects$` folder

- each subfolder name should appear in the file `PROJECTS.txt`
- there should not be any folder on the top level that does not represent a project.

- there should be no other files on the top level as the files that are described in this documentation.
- there are no subfolders representing departments any more. The mapping of projects to departments is done in the file `PROJECTS.txt`

2.2.5.2 Rawdata Folder Structure

We will create a network folder `//server/rawdata$` in which all files have set the read-only property. We suggest to store raw data by project first and by the organisation that owns (i.e. generated, provided) the data second. This could look like this:

```
//server/rawdata$
- ORGANISATIONS.txt
- PROJECTS.lnk [Symbolic Link to PROJECTS.txt in //server/projects$]
- flusshygiene
  - bwb
  - kwb
  - uba
  - ...
- ogre
  - kwb
  - bwb
  - uba
  - ...
- ...
```

2.2.6 Versioning

Versioning or version control is the way by which different versions and drafts of a document (or file or record or dataset or software code) are managed. Versioning involves the process of naming and distinguishing between a series of draft documents that leads to a final or approved version in the end. Versioning allows you to disclose an audit trail for the revision and update of drafts and final versions.

2.2.6.1 Manual

We propose the following workflow:

- It is only allowed to modify the current file (e.g. `filename.pptx`), which contains no version name as postfix
- A version is created by copying the current file. This copy gets a not already defined version name as file name ending (e.g. `filename_v0.1.pptx`)
- The versioned file is moved to a folder ‘VERSIONS” and set as read-only.
- In the folder `VERSIONS` is a file `VERSIONS.txt`, which contrains addtional information on the available version within that folder

Drawbacks:

- possibly more time demanding,
- needs time getting used to it and
- requires high level of discipline

Advantages:

- Simple and safe method, which requires not version control software
- Cleaner workspace as old versions are stored in a separate folder
- Sorting by name equals the chronology

Example:

```
BestPractices_Workshop.ppt
VERSIONS/
  + VERSIONS.txt
  + BestPractices_Workshop_v0.1.ppt
  + BestPractices_Workshop_v0.2.ppt
  + BestPractices_Workshop_v1.0.ppt
```

Content of file `VERSIONS.txt`:

```
BestPractices_Workshop.ppt
- v0.1: first draft version, written by Michael Rustler
- v0.2: after additions by Hauke Sonnenberg
- v1.0: first final version, after review by Pascale Rouault
```

2.2.6.2 Automatically

The versioning is done automatically in case a version control software, like Git or Subversion are used.

At KWB we currently use the following version control software:

- Subversion: for internally storing programm code (e.g. R-scripts/packages) we have an Subversion server, which is accessible from the KWB intranet. However, this requires:
  - the installation of the client software TortoiseSVN and a
  - valid user account (for accessing the server) which is currently provided by the IT department on request
- Git: for publishing programm code external on our KWB organisation group on Github. Currently all repositories are public (i.e. are visible for everyone), but also use of private repositories is possible for free as KWB is recognised as non-for-profit company by Github, offering additional benefits for free

Use of version control software is required in case of programming (e.g. in R, Python, and so on) and can be useful in case of tracking changes in small text files (e.g. configuration files that run a specific R script with different parameters for scenario analysis).

Drawbacks:

- Special software (TortoiseSVN), login data for each user on KWB-Server and some basic training are required
- In case of collaborate coding: sticking to 'best-practices' for using version control is mandatory, e.g.:
  - timely check in of code changes to the central server,
  - Speaking to each other: so that not two people work at the same time at the same program code in one script as this leads to conflicts that need to be resolved manually, which can be quite time demanding. You are much better off if you avoid this in the upfront by talking to each other

Advantages:

- Only one filename per script (file history and code changes are managed either internally on a KWB server in case of using TortoiseSVN or externally for code hosted on Github)
- Old versions of scripts can be restored easily

- Additional comments during `commit` (i.e. at the time of transfering the code from the local computer to the central version control system about `why` code changes were made and build-in diff-tools for tracking changes improve the reproducibility

Attention: version control software is not designed for versioning of raw data and thus should not be used for it. General thoughts on the topic of 'data versioning' are available here: https://github.com/leeper/data-versioning

A presentation with different tools for version control is available here: https://www.fosteropenscience.eu/node/597

2.2.7 Organising E-Mails

> "Most people now routinely send and receive lots of messages every day and as a Yesult, their inbox can get very quickly overloaded with hundreds of personal and work-related email. Setting aside some time to organise your emails will ensure information can be found quickly and easily, and is stored securely. Why should I organise my email? Apart from the obvious frustration and time wasted looking for that email you remember sending to someone last month, email is increasingly used to store important documents and data, often with information related to the attachments within the email itself."
>
> — University of Cambridge (2018)

How can I ensure my emails remain organised?

Here are some general tips to ensure your email remains organised in the long term University of Cambridge, 2018:

- Delete emails you do not need. Remove any trivial or old messages from your inbox and sent items on a regular (ideally daily) basis.
- Use folders to store messages. Establish a structured file directory by subject, activity or project.
- Separate personal emails. Set up a separate folder for these. Ideally, you should not receive any personal emails to your work email account.
- Limit the use of attachments. Use alternative and more secure methods to exchange data where possible (see 'data sharing' for options). If attachments are used, exercise version control and save important attachments to other places, such as a network drive.

2.2.8 Managing References

For managing reference there are plenty of tools available. A detailed overview is provided by (Fenner et al., 2014) in the Chapter Reference Management At KWB we use Endnote, for which an internal guideline document (KWB-EndNote-Guideline-v02.pdf,) was developed.

2.2.9 Data Preservation

Data must be retained to support your research findings

Standards: ?5 years?

2.3 Metadata

Metadata are data about data. It is up to us to define

- what metadata (about raw data, processed data, produced plots, documents, reports, etc.) we want to store,
- where to store metadata and
- in what format to store metadata.

We plan to specify the requirements in more detail when dealing with the test projects (case studies). Then, will also check metadata standards.

Metadata about raw data should always be stored.

Metadata about processed data are also important. However, in case of automated processing with a script, it may be possible to deduce the content of a generated file from the content of the script.

2.3.1 Why Metadata?

Why should we store metadata about data? Metadata are required to

- interpret raw data, e.g. "Why are the oxygen values so high? Ah, I see, someone was there to clean the sensor!",
- gain an overview about available data,
- know what we are allowed to do with the data.

2.3.2 What Metadata to Store?

General

What information would someone need to find/re-use your data? E.g.

- Location,
- Title,
- Creator name,
- Description,
- Date collected?

Metadata about Raw Data

What are the most important information about raw data that we receive?

- Obtained from whom, when, via whom and which medium, e.g.
  - `E-Mail from A to B on 2018-01-25` or
  - `USB-Stick given personally from C to D on 2018-01-26`
- Restriction of usage, e.g.
  - `only for project x` or
  - `only within KWB` or

  - `must not be published!` or
  - `should be published!`!
- Description of content and format
  - Where measurements were taken?
  - What methods were used (to take samples, to analyse parameters in the laboratory)?
  - What devices where used?
  - What do the columns of the table (in the database, XLS/CSV file) mean?
  - In what units are the values given?

Metadata about Processed Data

What are the most important information about the data that we produce?

- Who created the file? If the file was created by a script, what script created the file and who ran the script?
- When was the file created?
- What was the input data (e.g. raw data or preprocessed data)?
- Which methods were applied to generate the output from the input?
- What was the environment, what were boundary conditions, e.g.
  - versions of software,
  - versions of R packages?

Metadata about Programming Scripts

- What does the script do?
- Who wrote the script?
- How to use the script? Give a short tutorial.

Regarding R programming, we should consider providing scripts in the form of R packages. The R packaging system defines a framework of how to answer all of the above questions. See how we did this (not yet always with with a tutorial) with our packages on GitHub.

2.3.3 Where to Store Metadata?

The two main options of storing metadata are:

1. together with the data or near to the data i.e. in the same folder in which the data file resides,
2. in a central file or database.

Unless we have a professional solution (i.e. software on metadata management) we should prefer the first approach which is simpler and more flexible than the second one.

2.3.4 In What Format to Store Metadata?

keep metadata in plain text file “readme.txt”

2.3.5 Metadata Management Tools

Tools for metadata tracking and data standards are:

- metadata editor, e.g. online editor of GFZ Potsdam

2.3.6 Metadata Standards

We want to use a metadata standard.

Examples for metadata standards are (in brackets are listed the institutions who publish their data by using this standard):

- DataCite metadata scheme
  - Leibniz-Zentrum für Agrarlandschaftsforschung e. V. (ZALF)
  - Deutsches GeoForschungsZentrum Potsdam (GFZ) (+ ISO + DIF + Dublin Core)
- ISO 19115-2
  - Bundesanstalt für Gewässerkunde (BfG)
  - Alfred Wegener Institute, Helmholtz Centre for Polar and Marine Research (AWI)

Best-practices roadmap:

1. Check meta data standards, e. g. DataCite (see also: ZALF, GFZ Potsdam)
2. Define minimum metadata requirements at KWB for raw and processed data. projects.

The 'best-practices for metadata' will be developed for the case studies which are assessed within FAKIN project.

We propose to define some special files that contain metadata related to files and folders. To indicate that these files have a special meaning, the file names are all uppercase.

2.3.7 File PROJECTS.txt

This file contains the project acronyms as we want to use them e.g. as top-level folder names in our project folder structure. The projects are grouped by department.

Possible content of `PROJECTS.txt`:

```
# Department SUW (Surface Water)
dswt: DSWT
flusshygiene: Flusshygiene
kuras: KURAS
mia-cso: MIACSO
monitor: MONITOR
ogre: OGRE
reliable-sewer: RELIABLE_SEWER
sema: SEMA
sema-berlin: SEMA Berlin
sema-berlin-2: SEMA Berlin 2
spree-2011: SPREE2011
spree-2011-2: SPREE2011 "reloaded"
```

```
# Department GRW (Groundwater)
optiwells: OPTIWELLS
optiwells-2: OPTIWELLS 2
wellma: WELLMA

# Department WWT (Wastewater Treatment)
...
```

In the file `PROJECTS.txt` the project acronyms appear in alphabetical order. They map the acronym to a project name or a project title and the year of the start of the project.

Question: Do we already have a place where "official" metadata about projects are stored? If yes, the acronym could be included there. But then, everybody should know about it!

2.3.8 File ORGANISATIONS.txt

Possible content of `ORGANISATIONS.txt`

```
bwb: Berliner Wasserbetriebe
kwb: Kompetenzzentrum Wasser Berlin
uba: Umweltbundesamt
```

2.4 Data Collection

Data are often inconsistent, incomplete, incorrect, or misspelled. Data cleaning is essential.

For data cleaning you may use a GUI (Graphical User Interface) based tool like OpenRefine http://openrefine.org/ or choose a programmatic approach.

In the following we describe how data can be imported into the R-Programming Environment, which can be used for data cleaning, aggregation and visualisation. (Grolemund and Wickham, 2017)

2.4.1 Logger Devices

The R package kwb.logger (Sonnenberg, 2018) helps to import raw data from loggers used in different KWB projects into the software R (R Core Team, 2017), which is used for data processing (e.g. data cleaning, aggregation and visualisation).

For details, which loggers currently are supported by the R packages kwb.logger please check the documentation website.

2.4.2 Spreadsheets

General recommendations for working with EXCEL spreadsheets is given in the FAQs section.

2.4.2.1 Import Data From One Excel File

- Save the original file in the rawdata zone.

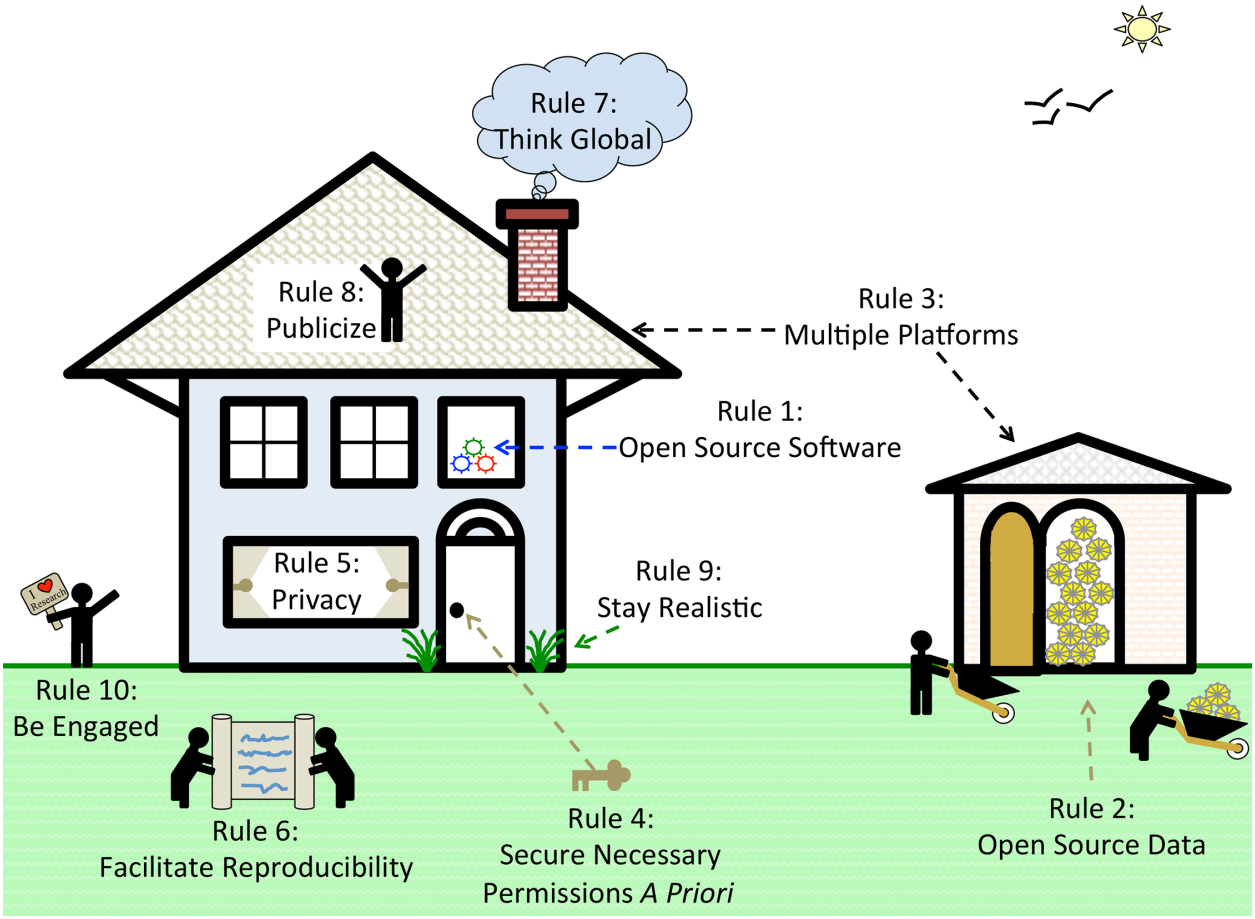


Figure 2.1: Modern life context for the ten simple rules (Boland et al., 2017)

2.4.2.2 Import Data From Many Excel Files

2.4.2.2.1 Files Are In the Same Format

Import Excel files of the same format by

- defining a function that is able to read the data from that file
- calling this function in a loop for each file to import.

2.4.2.2.2 Files Are In Different Formats

We developed a general approach of importing data from many Excel files in which the formats (e.g. more than one table area within one sheet, differing numbers of header rows) differ from file to file.

2.5 Data Publishing and Sharing

> "This figure provides a framework for understanding how the "Ten Simple Rules to Enable Multi-site Collaborations through Data Sharing" (Boland et al., 2017) can be translated into easily understood modern life concepts.

Rule 1 is Open-Source Software. The openness is signified by a window to a room filled with algorithms that are represented by gears.

Rule 2 involves making the source data available whenever possible. Source data can be very useful for researchers. However, data are often housed in institutions and are not publicly accessible. These files are often stored externally; therefore, we depict this as a shed or storehouse of data, which, if possible, should be provided to research collaborators.

Rule 3 is to "use multiple platforms to share research products." This increases thechances that other researchers will find and be able to utilize your research product—this is represented by multiple locations (i.e., shed and house).

Rule 4 involves the need to secure all necessary permissions a priori. Many datasets have data use agreements that restrict usage. These restrictions can sometimes prevent researchers from performing certain types of analyses or publishing in certain journals (e.g., journals that require all data to be openly accessible); therefore, we represent this rule as a key that can lock or unlock the door of your research.

Rule 5 discusses the privacy issues that surround source data. Researchers need to understand what they can and cannot do (i.e., the privacy rules) with their data. Privacy often requires allowing certain users to have access to sections of data while restricting access to other sections of data. Researchers need to understand what can and cannot be revealed about their data (i.e., when to open and close the curtains).

Rule 6 is to facilitate reproducibility whenever possible. Since communication is the forte of reproducibility, we depicted it as two researchers sharing a giant scroll, because data documentation is required and is often substantial.

Rule 7 is to "think global." We conceptualize this as a cloud. This cloud allows the research property (i.e., the house and shed) to be accessed across large distances.

Rule 8 is to publicize your work. Think of it as "shouting from the rooftops." Publicizing is critical for enabling other researchers to access your research product.

Rule 9 is to "stay realistic." It is important for researchers to "stay grounded" and resist the urge to overstate the claims made by their research.

Rule 10 is to be engaged, and this is depicted as a person waving an "I heart research" sign. It is vitally important to stay engaged and enthusiastic about one's research. This enables you to draw others to care about your research."

—- (Boland et al., 2017)

Recommended literature:

- Ten simple rules to enable multi-site collaborations through data sharing (Boland et al., 2017)
- Guidelines for publishing (PhD) research data (Kaden and Kleineberg, 2018)

2.5.1 Repositories

Repositories for permanently deposing data are for example:

- General
  - Figshare,
  - Zenodo (a joint project between OpenAIRE and CERN),
  - Mendeley data,
  - Dataverse,

  - Dryad
- Focus on environmental and earth sciences
  - Pangea
  - GFZ Potsdam data services

In addition repositories for publishing program code are for example Github or Gitlab, however these do not offer build-in long term preservation by default.

We are currently using the following repositories and can recommend them for similar usage to others:

- Github: for developing/publishing program code, because it offers the opportunity to automatically get a
- Zenodo DOI for each software released publicly on Github (for details see: https://guides.github.com/activities/citable-code/)

A blog post
Bosman and Kramer (2016) provide results of a large survey carried out in 2015 among more than 15000 researchers. Insights can be gained on:

- Which scholary communications tools are used and
- Are there disciplinary differences in usage?

They finally summarise: "Another surprising finding is the overall low use of Zenodo – a CERN-hosted repository that is the recommended archiving and sharing solution for data from EU-projects and -institutions. The fact that Zenodo is a data-sharing platform that is available to anyone (thus not just for EU project data) might not be widely known yet."

2.5.2 ORCID

Problem:

> "Two large challenges that researchers face today are discovery and evaluation. We are overwhelmed by the volume of new research works, and traditional discovery tools are no longer sufficient. We are spending considerable amounts of time optimizing the impact—and discoverability—of our research work so as to support grant applications and promotions, and the traditional measures for this are not enough. — (Fenner and Haak, 2014)

Solution:

> "Open Researcher & Contributor ID (ORCID) is an international, interdisciplinary, open and not-for-profit organization created to solve the researcher name ambiguity problem for the benefit of all stakeholders. ORCIDwas built with the goal of becoming the universally accepted unique identifier for researchers:
>
> 1. ORCID is a community-driven organization
> 2. ORCID is not limited by discipline, institution, or geography
> 3. ORCID is an inclusive and transparently governed not-for profit organization
> 4. ORCID data and source code are available under recognized open licenses
> 5. the ORCID iD is part of institutional, publisher, and funding agency infrastructures.
>
> Furthermore, ORCID recognizes that existing researcher and identifier schemes serve specific communities, and is working to link with, rather than replace, existing infrastructures."

— (Fenner and Haak, 2014)

2.5.3 Licenses

> "In most countries in the world, creative work is protected by copyright laws. International conventions, and primarily the Berne Convention of 1886, protect the copyright of creators even across international borders for 50 years after the death of the creator. This means that copying and using the creative work is limited by conditions set by the creator, or another copyright holder. For example, in many cases musical recordings may not be copied and further distributed without the permission of the musician, or of the production company that has acquired the copyright from the musician. Facts about the universe that are discovered through research are not subject to copyright, but the collection, aggregation, analysis and interpretation of research data may be considered creative work, and could be protected by copyright laws. Thus, the consumption of research publications is governed by copyright law. Furthermore, even data sharing is often governed by copyright laws, because the compilation of data to be shared often requires a creative effort. Another case of resarch-relevant copyrighted products is software that is developed in the course of research. In all of these cases, if license terms are not explicitly specified, the work is considered to be protected as"all rights reserved". This means that no one but the creator of the work can use the work unencumbered. For software this means that copying and further distribution of the software is prohibited. Even running the software may be restricted. The exact selection of a license is beyond the scope of this section, but depends on your intentions and goals with regard to the software"
>
> — (Rokem and Chirigati, 2018)

Recommended literature:

- Intellectual Property and Computational Science (Stodden, 2014)
- forschungslizenzen.de (For, 2018)
- Creative Commons Licences (Friesike, 2014)
- choosealicense.com/

2.5.4 File Formats

> "Scientific data is saved in a myriad of file formats. A typical file format might include a file header, describing the layout of the data on disk, metadata associated with the data, and the data itself, often stored in binary format. In some cases (e.g., CSV (or comma-separated value) files), data will be stored as text. The danger of proliferation of file formats in scientific data lies in the need to build and maintain separate software tools to read, write and process all these data formats. This makes interoperability between different practitioners more difficult, and limits the value of data sharing, because access to the data in the files remains limited."
>
> — (Rokem and Chirigati, 2018)

2.5.5 Data Exchange Standards

WaterML2:

> "...is a new data exchange standard in Hydrology which can basically be used to exchange many kinds of hydro-meteorological observations and measurements. WaterML2 has been initiated and designed over a period of several years by a group of major national and international organizations from public and private sector, such as CSIRO, CUAHSI, USGS, BOM, NOAA, KISTERS

Table 2.1: Suitability of file formats for long-term preservation (Kaden and Kleineberg, 2018)

	More than ten years	Up to ten years	Not suitable
Text	PDF/A, TXT, ASC, XML	PDF, RTF, HTML, DOCX, PPTX, ODT, LATEX	DOC, PPT
Data	CSV	XLSX, ODS	XLS
Pictures	TIFF, PNG, JPG 2000, SVG	GIF, BMP, JPEG	INDD, EPS
Audio	WAV	MP3, MP4	
Video	Motion JPG 2000, MOV	MP4	WMV

and others. WaterML2 has been developed within the OGC Hydrology Domain Working group which has a mandate by the WMO, too."

— WaterML2

ODM2: is an information model and supporting software ecosystem for feature-based earth observations

Chapter 3

Case Studies

The three following case-studies are tested in detail within FAKIN (i.e. proposed best-practices will be applied for this case studies and cross-checked whether their application is useful.

3.1 Geogenic Salination

Adapt and test with new folder drive workflow as proposed

- `rawdata`
- `processing`
- `processing`

3.2 LCA Modelling

Challenge:

The LCA modelling software Umberto can produce large raw data output files (> 300 MB csv files) that sometimes are even to big for EXCEL 2010 (> 1 millions) but need to be aggregated (e.g. grouped by specific criteria). This was usually performed manually within EXCEL in case that model output data was below EXCEL‘s 1 million row limit.

Workflow improvement developed within FAKIN:

An open source R package kwb.umberto was programmed for automating:

- data import the Umberto model results,
- performing data aggregating to the user needs and finally
- exporting the aggregated results in an `results.xlsx` EXCEL spreadsheet.

This `results.xlsx` EXCEL spreadsheet is referenced by another EXCEL spreatsheet `figues.xlsx` (which contains the figure templates and just links to the `results.xlsx` in order to update the predefined figures).

This workflow now reduces the time consuming and error-prone formerly manually performed data aggregation in EXCEL, whilst still enabling the users to adapt the figures to their needs without coding knowledge.

3.3 Pilot Plants

Challenges:

The output of (on-line) monitoring technologies is often difficult to interpret and also inconvenient to handle as the output formats of different devices (in one water treatment scheme) can vary strongly. Furthermore, frequent reporting and documentation of the treatment performance via (on-line) monitoring can be time consuming for the personnel and requires advanced software solutions. An alternative to commercial (and often expensive) software solutions are tools which are based on the open software R (R Core Team, 2017). The free software approach allows any R programmer to produce customized tools for each individual end-user.

Thus an automated reporting tool is developed within the AQUANES project for enabling an integrative assessment of the different monitoring devices and integration with water quality data obtained from analysis in the laboratory for four different pilot plant sites in order to:

- Increase the reliability and reproducibility of handling large amounts of data by reducing the likeliness in human error in complex systems and by increasing the transparency of the data processing.
- Promote the use of customized R tools for different end-user such as utilities, consultants and other research teams.

Therefore the open source R package aquanes.report (Rustler, 2018) was programmed, which is able to:

- import operational and lab data for each pilot site,
- performs temporal aggregation (e.g. 5 min, 1 h, 24h median values),
- visualises raw or aggregated data either interactively in a web browser or by
- creating a standardised report (e.g. monthly) in html, pdf or docx

For the four different pilot plant sites the data (operational and lab data) for being imported into the R tool came from various sources at different temporal resolutions, which are detailed below:

- Haridwar: operational data stored by Autarcon in mySQL database (temporal resolution: ~ 2-3 minutes, i.e. ~ 0.7 million data points per month), which is accessible from the web and thus could be easily imported into R. Lab data was provided by Autarcon initially in a unstructured format, which was impossible to be automatically imported into R. However, after agreeing on a standardised EXCEL spreadsheet format (e.g. one spreadsheet per site, one sheet per parameter and additional sheets providing metadata for parameters and sites) it was possible to integrate the lab data into the R tool.
- Basel Lange-Erlen: operational data is provided by the water supplier in EXCEL spreadsheets on a weekly basis for each site (i.e. “Rein” and “Wiese”) with a temporal resolution of 5 minutes (i.e. ~ 0.5 million data points per month). Lab data are provided by the water supplier in a single comma separated csv file, which is exported from a database. Thus the structure of the lab data was standardised and could be easily imported into the R tool.
- Berlin-Schönerlinde: operational data from the WEDECO pilot plant are collected using a SCADA system (~ temporal resolution: seconds, i.e. ~ 10 million data points per month). Lab data are provided by BWB in a single EXCEL spreadsheet. However, its structure often changes in case it is updated by BWB, making an automated importing using R impossible without adapting the import functions. Thus lab data were not integrated in the R tool for this site.
- Berlin-Tiefwerder: operational data from the PENTAIR pilot plant are collected using a SCADA system (~ temporal resolution: ~ seconds, i.e. ~ 10 million data points per month). Lab data are provided by BWB in a single EXCEL spreadsheet. However, its structure often changes in case it is updated by BWB, making an automated importing using R impossible without adapting the import functions. Thus lab data were not integrated in the R tool for this site.

The high temporal high resolution (~ seconds) of the operational data for both Berlin pilot plants resulted in large data amounts (~ 10 million data points per month), which required an large effort to optimise the performance of the R reporting tool in oorder to enable the visualisation of the pilot plant‘s raw data for its test operation period (~ 18 months) on computers with limited RAM ressources (~ 8-12 GB).

The R tool is used by KWB (for the sites Berlin-Schönerlinde and Berlin-Tiefwerder) regulary for assessing the pilot plants‘ operational performance interactively. In addition for an advanced assessment only the data importing and aggregation routines and combined with R scripts developed by KWB students.

For the other two pilot plant sites Haridwar and Basel Lange-Erlen the AQUANES project partners use the automated R reporting tool in a similar way.

Chapter 4

Other Projects

Here we are just suming up the data workflows and created tools in (old) KWB projects, but – in contrast to the case studies – the workflows and tools were not tested explicitly within the FAKIN project.

The goal of this chapter is to enlarge the internal knowledge base at KWB about what and how data challenges have already been successfully solved in the past.

4.1 Spree2011 (2007)

Used data by source (data formats in parentheses)

- KWB:
  - water level and discharge at one monitoring site (Text/CSV)
  - rain (Text/CSV)
- BWB:
  - pumping rates in the pumping stations (Excel)
  - water levels in the pumping stations (Excel)
  - rain at some gauges near the monitoring site (Excel)

Tasks and methods by topic

- Dry-weather and wet-weather calibration of a sewer network model (Infoworks)
  - InfoWorks: Creating rain input files
  - InfoWorks: Creating RTC input files

Questions that arose:

- Where to store presentations (trainee vs. employee)?
- Where to store the raw data (personal drive of the trainee)?
- How does Infoworks interpret timestamps, how do BWB provide timestamps? -> metadata

4.2 MIACSO (2009)

Monitoring

- sites: one site in the sewer (monitoring container), more sites in the river
- variables: water quantity and quality
- devices: online sensors

Modelling

- Sewerage: Infoworks
- River hydraulics: Hydrax
- River quality: QSim

Data storage

- High amount of data -> extra server: `moby`
- We put some effort in planning good folder structures for the data. Nevertheless the structure at the end of the project is not as clean as it was planned.
- Data that we received from project partners was stored in `Daten/EXTERN`.
- Raw data was stored in a folder `Daten/RAW` which was write-protected and required a special user-account for storing new data.

```
Daten/
  ACCESS/ # MS Access databases, containing raw data
  EXTERN/ # External data (by organisation)
  META/   # MS Access databases, containing metadata
          # (about calibration, maintenane, sites, variables)
  RAW/    # Text files containing raw data, from KWB-own devices only, by site
```

Metadata

- many devices in the container -> meta data about device cleaning and maintenance important -> tool: `META_Maintenance.mdb`

Methods and Tools

- We imported most of the data from text files into MS Access databases in -> tool: `MiaCsoRawImport.mdb`
- We calibrated the sensors offline by using SQL queries to provide calibrated data from raw data -> tool: `MiaCsoMetaCalibControl.mdb`
- We used SQL queries to perform data processing -> tool: `MiaCsoStatAnalysis.mdb`
- Data validation (outlier detection) was done in a two step procedure:
  1. Automatic preselection using MS Access tool `MiaCsoStatAnalysis.mdb`
  2. Manual selection using self-developed graphical tool in Origin

Developed Tools:

- MS Access Applications
  - `MetaMaint.mdb`: Monitoring Metadata Management
  - `MiaCsoRawImport.mdb`: Text File Import to MS Access
  - `MiaCsoStatAnalysis.mdb` (project deliverable): Definition and automatic execution of sequences of SQL queries
- Origin extension to interactively select and store outliers graphically
- R packages
  - `kwb.mia.evalCritO2` (project deliverable): graphical evaluation of critical oxygen conditions in the river
  - `kwb.mia.iw`: Calculation of file sizes of InfoWorks result csv-files exported from InfoWorks.
  - `kwb.miacso`: functions used in MIA-CSO, for example for plotting data availabilities.

4.3 KURAS

Developed Tools:

- Frontend for KURAS Database of Rainwater Management Measures: `KURAS_DB_Acc2003_hs.mdb`
- R package `kwb.kuras`: Interface to KURAS database

4.4 OGRE

- Decision to use CUAHSI Community Observations Data Model (ODM)
- R script to import lab data from Excel to MS Access database implementing ODM

Developed Tools:

- R packages
  - `kwb.ogre`
  - `kwb.ogre.model`
  - `kwb.odm`
  - `kwb.odmx`

4.5 Flusshygiene

- Adaptation of free online monitoring data visualisation HydroServerLite
- Reusage of lab data import script developed in OGRE

4.6 DEMEAU

Developed R packages for groundwater modelling:

- kwb.hantush (Rustler, 2016a)
- kwb.vs2dh
- kwb.demeau

4.7 DEMOWARE

R package kwb.qmra (Rustler, 2016b) is a generic QMRA (Quantitative Microbiological Risk Assessment) calculation engine assessing the performance of water supply systems.

It was successfully applicated for the Old Ford wasterwater treatment plant (Kraus et al., 2016; Rustler, 2016c). A detailed documentation for the R package and its usage is available online.

4.8 OPTIWELLS

Created R packages:

- kwb.wtaq (Sonnenberg and Rustler, 2016): groundwater modelling, e.g. for assessing the impact of production well (e.g. well diameter, pumping rates) and aquifer characteristics (e.g. hydraulic conductivity) on the resulting drawdawn. A detailled tutorial is available online,
- kwb.epanet: wrapper for (pressurised)pipe network simulation model EPANET

4.9 RWE

(Semi)automated creation of a complex groundwater simulation model with MODFLOW-2005 for the project Maxflow. The developed model consisted of up to three model layers with up to 1000 abstraction wells with temporal-spatial varying pumping rates within the simulation period. For automating the model generation the Python package flopy v3.2.5 (Bakker et al., 2016b,a) was used with some minor modifications by KWB (see here). Github was used as version control software for tracking changes in the code and each model scenario was stored in its Github branch (for details, see here).

Chapter 5

Literature Review

5.1 Data Storage

5.1.1 Ten Simple Rules for Digital Data Storage

The following rules are defined by Hart et al. (2016):

> Rule 1: Anticipate How Your Data Will Be Used
>
> Rule 2: Know Your Use Case
>
> Rule 3: Keep Raw Data Raw
>
> Rule 4: Store Data in Open Formats
>
> Rule 5: Data Should Be Structured for Analysis
>
> Rule 6: Data Should Be Uniquely Identifiable
>
> Rule 7: Link Relevant Metadata
>
> Rule 8: Adopt the Proper Privacy Protocols
>
> Rule 9: Have a Systematic Backup Scheme
>
> Rule 10: The Location and Method of Data Storage Depend on How Much Data You Have
>
> — (Hart et al., 2016)

Chapter 6

FAQs

6.1 Naming

Why is the space character not allowed?

Why is the space character not allowed in file or folder names?

File paths containing spaces need to be put into quotes when being used in a command line, such as in:

```
"C:\Programs\model software xyz\run.exe" "\\server\path with space\input.txt"
```

With all paths lacking the space character you can omit the quotes:

```
C:\Programs\model-software-xyz\run.exe \\server\path-without-space\input.txt
```

Consequent usage of paths without spaces allows for simpler programming and leads to less errors during program execution.

Why are special characters not allowed in file or folder names?

The way in which special characters are encoded may differ from system to system, especially between systems with different language settings. File names containing German special characters are e.g. not shown correctly on a French computer and vice versa.

Excluding (e.g. language-specific) special characters from the set of allowed characters in file or folder names avoids problems when exchanging files between partners in different countries.

We may have a problem with a file from Sofia, called

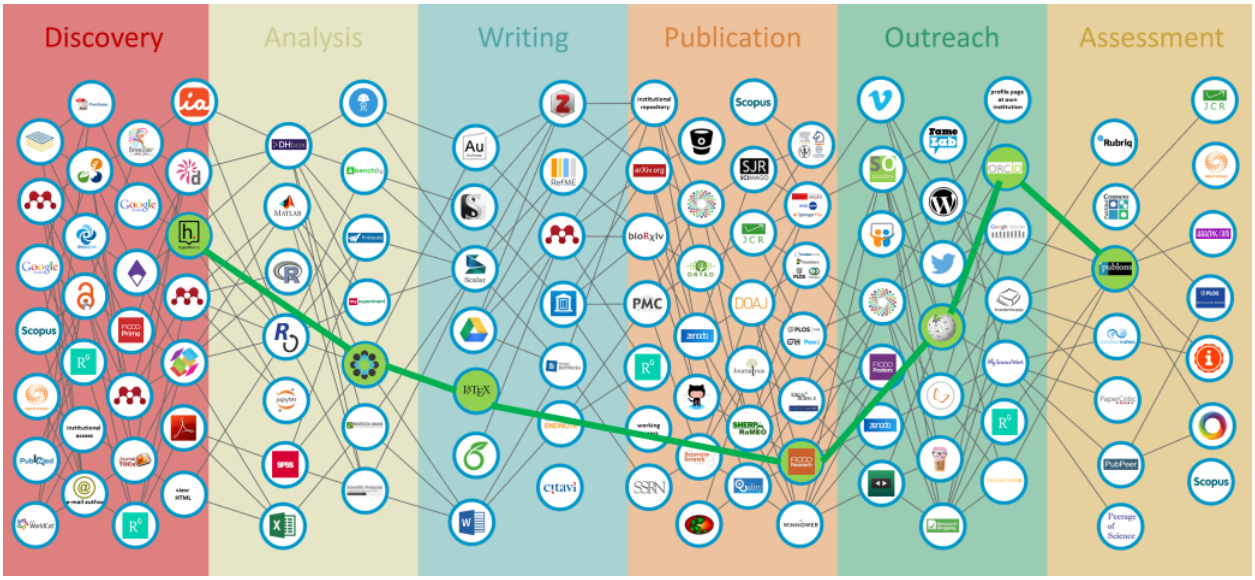
```
Example file from Canasoft  .        ,  .    _Troncons.xml
```

and they may have a problem with a file from us, called

```
abgebrochene schmuz- und regenwasserhaltungen schützenstraße nord 2. ba.xls
```

6.2 Tools for Researchers

A very exhaustive overview of tools used in the researcher‘s workflow are provided by Bosman and Kramer (2018), which are grouped according to different research phases as shown below.



6.3 Writing More Robust R Code

For details on making R code more robust to work on different computers please read the following tutorial.

Recommended literature:

- Good enough practices in scientific computing (Wilson et al., 2017)
- Ten simple rules for making research software more robust (Taschuk and Wilson, 2017)
- R programming books (freely available online!)
  - R for data science (Grolemund and Wickham, 2017)
  - Advanced R (Wickham, 2015a, 2018)
  - R packages (Wickham, 2015b)

6.4 Learning R on DataCamp

Since June 2017 we have a corporate premium group for the online R programming and learning platform DataCamp with five seats, which improvd the R le**R**rning workflow dramatically.

Before that we already had a free KWB group on DataCamp. However, in case you were interested in doing the paid courses, each user needed to subscribe him/herself and needed to pay the fee on a monthly basis to DataCamp individually, which is now obsolete.

For using DataCamp at KWB the following workflow applies:

1. **Become a member in our free R@KWB DataCamp group** by clicking on the following link. In case you do not have a DataCamp account yet, please register yourself first by using your KWB email address. The membership in this group serves two purposes: on the one hand you should start on DataCamp with the free courses like for example Introduction to R first. On the other hand this group serves us as a kind of long term memory, to assess who and how frequently the DataCamp courses are used as the amount of users in the free group is – in contrast to our paid premium DataCamp group – not limited.

2. If you want to do a paid DataCamp course, please contract our DataCamp administrator Nicolas Caradot either personally or via email. In case there is a free seat available in our corporate premium group, he will add you so that you can start any of the available premium DataCamp courses.
3. As soon as you either do not want to or you do not have the time for a longer period do one of the paid DataCamp courses, please contact our group administrator Nicolas Caradot either personally or via email, so that he can make your seat available for others at KWB who are interested in learning

R.

6.5 Using Subversion at KWB

For details on how to use Subversion at KWB please read the following tutorial

6.6 How to Build Your Own Kwb Styled R Package?

For building your own R package from scratch we developed a helper R package kwb.pkgbuild (available on Github), which builds a KWB styled skeleton for your future R package.

With the help of this tool Andreas Matzinger was able to convert this R scripts on resilience within a few hours into the R package kwb.resilience which is now available on Github.

For more details on turning your own code into a R package checkout the tutorial at the package documentation website.

Recommended literature:

- R packages (Wickham, 2015b)
- Advanced R (Wickham, 2015a)

6.7 How to Install KWB R Packages?

Most R packages developed at KWB are not only available in the intranet but are also available on Github. Please check the following website: http://kwb-r.github.io/status/ or https://github.com/KWB-R

Installation of these R packages can be performed with the following command in R(studio):

```
### Required to install an R package from Github
install.packages("devtools", repos = "https://cloud.r-project.org")

### Now install your desired R package (e.g. "kwb.resilience")
devtools::install_github("kwb-r/kwb.resilience")
```

However, in case your required R package that is not yet Github please read the following tutorial

6.8 Working with Excel

Excel often crashes in case:

- a formula is applied for a whole column (i.e. 1 million rows)

– a lot of data is processed

Solution: data in own spreadsheet file

Hauke's "best Practices" for EXCEL (unvalidated, to be discussed!)

- Trennung zwischen Eingabe, Verarbeitung und Ausgabe zumindest auf Tabellenblattebene, d.h. ein Tabellenblattt (oder mehrere) für Eingabe, eines (oder mehrere) für Verarbeitung, eines (oder mehrere) für formatierte Ausgabe und / oder Diagramme
- ggf. Aufteilen auf mehrere Dateien. Das hätte den Nachteil, dass nicht mehr alles in einer Datei ist und nicht so leicht übergeben werden kann. In jedem Fall müsste eine Namenskonvention getroffen werden, z.B.`<file_name>`_input.xlsx, `<file_name>`_calc.xlsx, `<file_name>`_output.xlsx
- Verwenden der relativ neuen EXCEL-Features `Als Tabelle` formatieren.

Vorteil: Formeln können auf ganze Tabellenspalten angewendet werden; Spaltennamen anstatt Zellbezüge mit (unsprechenden) Buchstaben und Zahlen. z.B. Formel für Spalte `Volumen_L`: `= Durchfluss[@[Q_L_s]] * 60 * 5`

- Ein Tabellenblatt pro Tabelle
- Genau eine Headerzeile pro Tabelle mit eindeutigen Spaltennamen
- Ein Tabellenblatt, das die Bedeutung der Spaltennamen erläutert mit Spalten `Tabellenblatt; Spalte; Bedeutung; Einheit; Formel` Vorteil: Dieses Tabellenblatt sollte ausreichen, um die wesentlichen Berechnungen zu verstehen.

Drawback: needs to be permanently kept up-to-date!

- Hilfsspalten mit (dadurch benannten) Zwischenberechnungen anstatt Wiederholung von langen Ausdrücken in Formeln

A general online workshop on the topic Data Organisation in Spreadsheets is provided for free by the DataCarpentry organisation.

copied from `10_old_german_chapters.Rmd` :

Best Practices zum Arbeiten mit Excel:

- z.B. Zellbezüge benennen, dadurch werden Formeln besser lesbar

Ihr werdet es kaum glauben, aber am Anfang meiner Zeit am KWB habe ich noch mit Excel gearbeitet. Ich habe auch komplexe Sachen gemacht und auch Excel-Makros programmiert. Ein Beweis findet sich in meiner persönlichen Logdatei:

> Fr, 14.09.07 08:15-19:30 benutzerdefinierte Excelfunktionen zur Modellgüte in Personl.xls, Modul1; TW-Kalibrierung Wochentag fertig

Ich würde heute nicht mehr empfehlen, Excel-Makros zu programmieren. Es ist umständlich, es gibt keine Bibliotheken und der Quellcode lässt sich nicht unabhängig von der Exceldatei verwalten, so dass keine ordentliche Versionsverwaltung möglich ist. Und wir wollen nicht mehr ohne Versionsverwaltung programmieren!

Wenn wir programmieren, dann sollten wir das einheitlich in R tun. R ist frei, es gibt eine großartige Community und wir haben mittlerweile eine große Expertise erlangt.

6.9 Heterogenous Software Versions on KWB Computers

Unterschiedliche Softwareversionen (z.B. R) können dazu führen, dass Skripte auf verschiedenen Rechnern nicht das gleiche Verhalten zeigen.

Die IT-Abteilung ist in der Lage an bestimmte Nutzergruppen die gleiche Software (z.B. RStudio / R / Miktex) auszurollen. Dies sollte in Zukunft konsequent genutzt werden, indem auf alle Computer an denen potentiell programmiert wird zu dieser Nutzergruppe hinzugefügt werden und somit alle die gleichen Softwareversionen installiert haben.

6.10 R Package/Version Dependency of R Scripts

Lösungsvorschlag:

Es ist eine **Mindestdokumentation** der verwendeten R Version und sämtlicher R-Pakete (inklusive ihrer Versionen) zu fordern. Dazu kann in R die Funktion `sessionInfo()` genutzt werden. Die Ausgabe dieser Funktion kann entweder in eine Metadaten-Textdatei `session_info.txt` geschrieben werden oder im Falle der Erzeugung von R-Markdown-Dokumenten direkt am Anfang der Analyse im `R-Markdown` Dokument ausgegeben werden.

Das Schreiben der Metadaten-Datei `session_info.txt` sollte standardisert über eine Funktion in einem KWB R-Paket (z.B. `kwb.utils`) implementiert werden.

Direktausgabe in R Console / RMarkddown:

```
sessionInfo()
```

```
## R version 3.5.0 (2017-01-27)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.5 LTS
##
## Matrix products: default
## BLAS: /home/travis/R-bin/lib/R/lib/libRblas.so
## LAPACK: /home/travis/R-bin/lib/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] kableExtra_0.9.0    tibble_1.4.2        knitcitations_1.0.8
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.18      compiler_3.5.0    pillar_1.3.0
##  [4] plyr_1.8.4        highr_0.7         tools_3.5.0
```

```
##  [7] digest_0.6.17     lubridate_1.7.4   jsonlite_1.5
## [10] evaluate_0.11     viridisLite_0.3.0 pkgconfig_2.0.2
## [13] rlang_0.2.2       bibtex_0.4.2      rstudioapi_0.7
## [16] curl_3.2          yaml_2.2.0        xfun_0.3
## [19] RefManageR_1.2.0  httr_1.3.1        stringr_1.3.1
## [22] xml2_1.2.0        knitr_1.20        hms_0.4.2
## [25] rprojroot_1.3-2   R6_2.2.2          rmarkdown_1.10
## [28] bookdown_0.7      readr_1.1.1       magrittr_1.5
## [31] backports_1.1.2   scales_1.0.0      htmltools_0.3.6
## [34] rvest_0.3.2       colorspace_1.3-2  stringi_1.2.4
## [37] munsell_0.5.0     crayon_1.3.4
```

Schreiben in standardisierte Metadatei:

```
sink("session_info.txt")
sessionInfo()
sink()
```

Hierzu ist auch noch ein Tutorial zu erstellen!

Komplexere, technische Möglichkeiten zum Paketmanagement werden im gerade vom DFG gefördertem Projekt O2R in der Entwicklung befindlichen CRAN task view for computational environments and reproducibility genannt.

Mit dem R Paket packrat lässt sich das Paketmanagement ggf. verbessern.

Als Beispielanwendung dient die Datenanalyse zur Feinstaubbelastung mit Sensebox-Daten, die komplett reproduzierbar sind (mittels der Plattform mybinder) und ohne Installation von Interessierten im Webbrowser ausgeführt werden kann (siehe hier).

6.11 Complex R Script Dependencies

Dieses Problem tritt insbesondere auf, wenn mehrere verschiedene Nutzer gemeinsam mit den gleichen Skripten arbeiten (wie z.B. im abgeschlossenen Projekt OGRE).

Proposed solution:

Bewusstmachen der Skriptabhängigkeiten

Identifizieren von Optimierungspotentialen -> möglicherweise Elimination von Abhängigkeiten

Workflow dokumentieren und Tutorial, am besten als R-Markdown Dokument, erstellen. Das ist insbesondere wichtig, wenn Skripte häufig verwendet werden.

6.12 Heterogenous Coding Styles

Currently there is no established coding style at KWB for in case of programming e.g. R scripts

(R)Programmers at KWB will use the tidyverse coding style http://style.tidyverse.org as default. This will help increasing both, the readability and reusability of the developed (R-)scripts at KWB (currently: 1000 R scripts).

Recommended literature:

Clean code (Martin, 2009)

6.13 Collaborative Version Control

Multiple people developing code together.

1. At KWB we use Subversion (SVN) for version control (see: How to use subversion?)
2. Speak with each other (especially if you are working on code is likely to be changed be others)
3. Regularly perform updates/commits with the version control software SVN

In case the code base (e.g. R scripts) developed in 'project A' should be reused/adapted in a new 'project B' (e.g. FLUSSHYGIENE) it is not allowed to
change the code base in the original 'project A'. Instead the code needs to be copied in a folder for the new project by using Subversion's SVN copy command. In case this procedure is not followed established processes in the souce project will be overwritten.

6.14 Workflow Automation

Ressources:

- Reproducible Research Automation: This lesson shows how to use automation in R to improve the reproducibility of research by automating tasks.
- useR!2017 Video: Data Carpentry: Open and Reproducible Research (Tutorial)
  - Part1
  - Part2
- Use of an R package to facilitate reproducible research, e.g.:
  - rprrkg by rOpenSci
  - rrtools by Ben Marwick

Recommended literature:

- Kitzes et al. (2018)

6.15 Encoding

Umlaute und Sonderzeichen werden falsch angezeigt, wenn R-Skripte in unterschiedlichen Encodings abgespeichert und eingelesen werden.

Vorgabe einer Default Encoding Einstellung in RStudio (z.B. UTF-8)

Alternativ könnten auch alle Umlaute in Unicode dokumentiert werden (siehe folgendes Beispiel). Allerdings ist dies wohl nicht praktikabel, da die Lesbarkeit der Texte erschwert wird und es sollte daher von uns der erste Ansatz (Vorgabe von UTF-8 als Default Encoding) angestrebt werden.

A great blogpost for the topic (How do I write UTF-8 encoded content to a file?) is provided by Ushey (2018), including background information on how encoding in R works.

In order to increase the portability of R script he recommends the following:

> "Portable R scripts should use unicode code points, to avoid accidental mis-encoding of string literals."
>
> — (Ushey, 2018)

Encoding example in R:

```
### Richtiges Skript Encoding WICHTIG (richtiges Einlesen des Skripts nur wenn
### mit gleichem Encoding eingelesen  wie es auch abgespeichert wurde)
print("Ü")
```

```
## [1] "Ü"
```

```
### Skript Encoding EGAL (da Umlaut in Unicode codiert wurde)
print("\u00dc")
```

```
## [1] "Ü"
```

See also Hauke's article Understanding Encoding being part of the documentation of our R package kwb.fakin that is available on GitHub.

6.16 Collaborative Writing

Project proposal with many partners. How can be guaranteed that only the most recently document is used and how can multiple people work at the same time in that document in the "hot phase" of proposal writing?

Ulf made good experiences with Office 365 (provided by KWR at that time). Also writing of up to
10 persons in parallel were not a problem, because the currently processed paragraph automatically gets blocked for others.

However, in case a whole chapter needs to be processed by one person this is not performaed automatically. Thus it is recommended to delete the chapter for a short term from the syncronised document and replace it with a placeholder like 'Chapter currently in revision by XXXX. Will be put back in this document until YYYY-MM-DD'.

6.17 Exchanging Data with Colleagues/Project Partners

copied from: "10_old_german_chapters.Rmd", to be translated:

Es gibt internen Datenaustausch und Austausch mit Projektpartnern. Oft werden Dateien intern per E-Mail verschickt. Das sollten wir nicht tun, da wir unnötig Kopien von Dateien anlegen. Dateien sollten im besten Fall nur an einem Ort abgelegt sein. Dort sollten sie allerdings durch ein Backup-System gesichert sein. Natürlich muss der Zugriff auf diesen Ort gewährleistet sein. Einem Studenten mit eingeschränkten Rechten eine Datei per mail zuzuschicken, weil sie an dem Ort, an dem sie gespeichert ist, für ihn nicht zugänglich ist, ist meines Erachtens nicht die richtige Lösung. Es muss dann ein Ort geschaffen werden, an dem der gemeinsame Zugriff möglich ist. Laut Bodo ist der Ordner `Exchange` im Projektordner ein solcher Platz. Aus dem Grund der allgemeinen Zugänglichkeit wir er auch in manchen Projekten als allgemeiner Datenablageplatz "misbraucht". Es kann aber niemandem ein Vorwurf gemacht werden, da ja nirgendwo dokumentiert ist, was in diesem Ort abgelegt werden soll.

Das können wir nun ändern:

Laut Bodo sind Dateien im Exchange-Ordner nur kurzfristig abzulegen, um z.B. Studenten Zugriff darauf zu geben. Die Dateien sollen nach dem Gebrauch wieder aus dem Ordner gelöscht werden. Demnach sollte der Ordner im Normallfall leer sein. Dies ist in einigen Projekten nicht der Fall.

Wir brauchen eine einheitliche Definition der Bedeutungen von Ordnern.

This should be documented in the metadata.

Chapter 7

Glossary

A detailed glossary covering the following topics is provided by Rokem and Chirigati (2018) is used throughout the glossary.

7.1 Reproducibility

> "… is a cornerstone of science. Definitions vary greatly across scientific disciplines, but the meaning that we find most prevalent is the 'calculation of quantitative scientific results by independent scientists using the original datasets and methods' (Stodden, Leisch, & Peng, 2014). The goals of reproducibility go beyond duplicating someone else's investigation: it also entails having reproducibility for yourself, defeating self-deception in scientific results (Ioannidis, 2005; Nuzzo, 2015), and extending another researcher's methods to build your own work. Reproducibility is a matter of degree, not of kind. We say that research is reproducible if reproducibility applies to the results to some extent. That is, some of the corresponding experiments and scientific methods are deemed to be reproducible.
>
> — (Rokem and Chirigati, 2018)

For more details see: Rokem and Chirigati (2018)

7.2 Provenance

> "As the volume of digital data increases and the complexity of computational processes that manipulate these data grows, it is becoming increasingly important to manage their provenance. The Oxford English Dictionary defines provenance as the source or origin of an object; its history and pedigree; a record of the ultimate derivation and passage of an item through its various owners. Provenance helps determine the value, accuracy, and authorship of an object."
>
> — (Rokem and Chirigati, 2018)

For more details see: Rokem and Chirigati (2018)

7.3 Techniques

- Version control

- Literate Programming
- Data Publication
- Data cleaning/munging
- Software Testing
- Continuous Integration
- Workflow Management
- File Format Standards
- Licensing
- Virtualization and Environment Isolation

For details see: Rokem and Chirigati (2018)

7.4 Tools

- Programming Language and Related Tools
- Documentation Generators
- Version Control
- Data Munging and Analysis
- Data Visualization
- Software Testing and Continuous Integration
- Virtualization and Environment Isolation
- Data Sharing and Repositories
- Document Authoring
- File Formats

For details see: Rokem and Chirigati (2018)

Bibliography

(2018). Forschungslizenzen. http://forschungslizenzen.de/. Accessed: 2018-06-29.

Bakker, M., Post, V., Langevin, C. D., Hughes, J. D., White, J., Starn, J., and Fienen, M. (2016a). Flopy: Python package for creating, running, and post-processing modflow-based models.

Bakker, M., Post, V., Langevin, C. D., Hughes, J. D., White, J. T., Starn, J. J., and Fienen, M. N. (2016b). Scripting MODFLOW model development using python and FloPy. *Groundwater*, 54(5):733–739.

Boland, M. R., Karczewski, K. J., and Tatonetti, N. P. (2017). Ten simple rules to enable multi-site collaborations through data sharing. *PLOS Computational Biology*, 13(1):e1005278.

Bosman, J. and Kramer, B. (2016). Github and more: sharing data & code. https://101innovations.wordpress.com/2016/10/09/github-and-more-sharing-data-code/.

Bosman, J. and Kramer, B. (2018). Workflows. https://101innovations.wordpress.com/workflows.

Buettner, S., Hobohm, H.-C., and Mueller, L. (2011). *Handbuch Forschungsdatenmanagement*. Bock u. Herchen, Bad Honnef.

Fenner, M. and Haak, L. (2014). *Unique Identifiers for Researchers*, pages 293–296. Springer International Publishing, Cham.

Fenner, M., Scheliga, K., and Bartling, S. (2014). *Reference Management*, pages 125–137. Springer International Publishing, Cham.

Friesike, S. (2014). *Creative Commons Licences*, pages 287–288. Springer International Publishing, Cham.

Grolemund, G. and Wickham, H., editors (2017). *R for Data Science*. O'Reilly Media, Sebastopol, CA, 1 edition.

Hart, E. M., Barmby, P., LeBauer, D., Michonneau, F., Mount, S., Mulrooney, P., Poisot, T., Woo, K. H., Zimmerman, N. B., and Hollister, J. W. (2016). Ten simple rules for digital data storage. *PLOS Computational Biology*, 12(10):e1005097.

Kaden, B. and Kleineberg, M. (2018). Guidelines zur veröffentlichung dissertationsbezogener forschungsdaten.

Kitzes, J., Turek, D., and Deniz, F., editors (2018). *The Practice of Reproducible Research: Case Studies and Lessons from the Data-Intensive Sciences*. University of California Press, Oakland, CA.

Kraus, F., Seis, W., Remy, C., Rustler, M., Jubany i Guell, I., Espi, J. J., and Clarens, F. (2016). Deliverable d3.2: Show case of the environmental benefits and risk assessment of reuse schemes. Report, public report in the DEMOWARE project (www.demoware.eu), KWB.

Martin, R. C. (2009). *Clean Code*. Pearson Education, Inc. ISBN-13 978-0-13-235088-4978-1498716963.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rokem, A. and Chirigati, F. (2018). *Glossary*. University of California Press, Oakland, CA.

Rustler, M. (2016a). kwb.hantush (v.0.2.1). Tutorial: http://kwb-r.github.io/kwb.hantush/.

Rustler, M. (2016b). kwb.qmra (v.0.1.1).

Rustler, M. (2016c). Quantitative microbiological risk assessment for different wastewater reuse options in Old Ford (v.1.0). For performing the quantitative microbial risk assessment the R package kwb.qmra, v.0.1.1 (https://doi.org/10.5281/zenodo.154111) is used.

Rustler, M. (2018). aquanes.report (v.0.5.0).

Sonnenberg, H. (2018). kwb.logger (v 0.2.0).

Sonnenberg, H. and Rustler, M. (2016). kwb.wtaq (v.0.2.1). Tutorial website: http://kwb-r.github.io/kwb.wtaq.

Stodden, V. (2014). *Intellectual Property and Computational Science*, pages 225–235. Springer International Publishing, Cham.

Taschuk, M. and Wilson, G. (2017). Ten simple rules for making research software more robust. *PLOS Computational Biology*, 13(4):e1005412.

Ushey, K. (2018). String encoding and r. https://kevinushey.github.io/blog/2018/02/21/string-encoding-and-r/.

Wickham, H., editor (2015a). *Advanced R*. The R Series. Chapman and Hall/CRC, Boca Raton, FL, 1 edition.

Wickham, H., editor (2015b). *R Packages: Organize, Test, Document, and Share Your Code*. O'Reilly Media, Sebastopol, CA, 1 edition.

Wickham, H., editor (2018). *Advanced R*. 2 edition.

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., and Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*, 13(6):e1005510.