Best Practices in Research Data Management

Hauke Sonnenberg & Michael Rustler

2018-06-26 10:39:29

Contents

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

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?

2.1.2.1 Project acronyms

At the start of a project we define a project acronym. This acronym is intended to be used in file and folder 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 Folder Structures.

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

2.2 Data storage

2.2.1 Folder structure

@Michael: Regeln für die Verbesserung der Ordnerstruktur

2.2.1.1 Regeln dafür, was wohin gehört

Was gehört in den Projektordner, was nicht?

Software sollte nicht innerhalb der Projekte gespeichert werden

Vorteil:

- Software wird ggf. projektübergreifend verwendet
- Software besteht oft aus vielen und großen Binärdateien und bläht den Projektordner auf
- Software hat oft viele Unterverzeichnisse und erzeugt eine hohe Verschachtelungstiefe im Projektordner

Nachteil:

- Es ist ggf. nicht klar, in welchem Projekt eine Software verwendet wurde

@Michael: Vorschlag für verbesserte Ordnerstrukturen

2.2.2 Regeln zur Vermeidung langer Dateipfade

Lange Dateipfade können im Windows-Betriebssystem ein Problem darstellen. Das äußert sich beim Kopieren von Dateien, wenn der Pfad des Zielorts, der sich durch die Kopie ergeben würde, eine bestimmte Länge überschreitet. Die zugehörige Fehlermeldung lautet dann: *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.

2.2.3 Naming

@Michael: Konventionen für Ordner- und Dateinamen, Vorschlag einer Nomenklatur

Useful file names are:

- consistent
- meaningful to you and your colleagues
- allow you to find the file easily

It is useful if your department/project agrees on the following elements of a file name:

- Vocabulary – choose a standard vocabulary for file names, so that everyone uses a common language
- Punctuation – decide on conventions for if and when to use punctuation symbols, capitals, hyphens and spaces
- Dates – agree on a logical use of dates so that they display chronologically i.e. YYYY-MM-DD
- Order - confirm which element should go first, so that files on the same theme are listed together and can therefore be found easily

• Numbers – specify the amount of digits that will be used in numbering so that files are listed numerically e.g. 01, 002, etc.

!!!!! Good practice: Remove spaces from file names or use punctuation such as underscores and hyphens to separate words e.g. "AHRC_TechnicalApp_Response20120925.docx" or "AHRC-TechnicalApp-Response20120925.docx" rather than "what we got back from funders about the data stuff.docx !!!!

2.2.4 Raw data

Especially in environmental sciences, raw data often cannot be reproduced (e.g. rainfall, river discharge measurements) and are therefore of high value.

- dürfen umbenannt werden, dies muss in den Metadaten dokumentiert werden
- dürfen inhaltlich nicht verändert werden
- werden schreibgeschützt abgelegt
- werden in einem eigenen Bereich abgelegt (ein Ordner enthält alle Rohdaten)

2.2.5 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.

2.2.5.1 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
```

2.2.5.2 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

2.2.6 Organising your e-mail account

> “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.

2.2.7 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.

2.2.8 Metadata

What information would someone need to find/re-use your data? Location, title, creator name, description, date collected

- keep metadata in plain text file “readme.txt”

Tools for metadata tracking and data standards are:

- Metadatenstandards prüfen, z. B. DataCite (siehe u.a. ZALF, GFZ Potsdam)

2.2.9 Data preservation

Data must be retained to support your research findings

Standards: ?5 years?

2.3 Data collection

Data are often inconsistent, incomplete, incorrect, or misspelled Data cleaning is essential You may also use OpenRefine http://openrefine.org/ to clean your messy data Or use the following tools

2.3.1 Logger devices

2.3.2 Spreadsheets

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. The free software approach allows any R programmer to produce customized tools for each individual end-user `to be added in manual.bib: (Gibert et al., 2015; Sonnenberg et al., 2014).`

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 DEMOWARE

Entstandene R Pakete:

- Grundwassermodellierung
  - kwb.hantush
  - kwb.vs2dh
  - kwb.demoware
- Quantitatives mikrobiologisches Risikomanagement
  - kwb.qmra: wird im Rahmen von AQUANES(#aquanes) weiter genutzt

4.7 OPTIWELLS

Created R packages:

- kwb.wtaq: Groundwater Modelling
- kwb.epanet: (Pressure)Pipe Network Simulation (EPANET)

4.8 RWE

(Semi)automatisierte Erstellung eines komplexen MODFLOW Modells (mehrere Layer, mehrere hunderte Entnahmebrunnen mit zeitlich variierender Entnahmemenge sowie Hinzufügen/Entfernen von Brunnen innerhalb des Simulationszeitraums) in Phyton mittels flopy sowie Entwicklung der Modellszenarien auf Github (siehe hier: Maxflow).

Input Christian !?!

Chapter 5

FAQs

5.1 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

-> Abhilfe: Daten in eigene Datei

Haukes “Best Practices” für Excel (ungeprüft, zu diskutieren!)

- 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. _input.xls, _calc.xls, _output.xls
- 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.

-> Nachteil: Das muss immer aktuell gehalten werden!

- 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.

5.2 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.

5.3 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:

```r
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] knitcitations_1.0.8
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.17     bookdown_0.7.13  lubridate_1.7.4  digest_0.6.15
##  [5] rprojroot_1.3-2  plyr_1.8.4       R6_2.2.2         jsonlite_1.5
##  [9] backports_1.1.2 magrittr_1.5     evaluate_0.10.1 httr_1.3.1
## [13] bibtex_0.4.2    stringi_1.2.2    rstudioapi_0.7  xml2_1.2.0
## [17] rmarkdown_1.10  tools_3.5.0      RefManageR_1.2.0 stringr_1.3.1
## [21] xfun_0.2        yaml_2.1.19      compiler_3.5.0  htmltools_0.3.6
## [25] knitr_1.20
```

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).

5.4 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.

5.5 Heterogenous (R-)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 readibility and reusability of the developed (R-)scripts at KWB (currently: approximately 1000 Rscripts).

5.6 Collaborative version control

Multiple people developing code together.

Proposed solution:

1. Miteinander sprechen,
2. Regelmäßige Updates/Commits,
3. Bei Verwendung/Weiterbearbeitung von Code aus einem Projekt (z.B. OGRE) in einem weiteren (z.B. FLUSSHYGIENE) ist vom Orginalcode eine Kopie mittels `SVN copy` zu erstellen und diese weiterzubearbeiten, da sonst bestehende Prozeduren im Ausgangsprojekt überschrieben werden

Bestehendes Subversion Tutorial (link only accessible from KWB intranet) ist um die fehlenden der oben genannten Aspekte zu ergänzen.

Eine Präsentation mit verschiedenen Tools zur Versionsverwaltung findet sich hier: https://www.fosteropenscience.eu/node/597

5.7 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.

For increasing the portability of R script he recommands the for example the following:

> “Portable R scripts should use unicode code points, to avoid accidental mis-encoding of string literals.”* — Ushey, 2018

Encoding example in R:

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

```
## [1] "Ü"
```

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

```
## [1] "Ü"
```

Chapter 6

Glossary

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

6.1 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)

6.2 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)

6.3 Techniques

- Version control

- Literate Programming
- Data Publication
- Munging (i.e. data cleaning)
- Software Testing
- Continuous Integration
- Workflow Management
- File Format Standards
- Licensing
- Virtualization and Environment Isolation

For details see: Rokem and Chirigati (2018)

6.4 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

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

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.

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

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