PPS data management recipe

Version 1.1

Instructions for setting up a data management structure for a new project

Authors:

Joost van Heerwaarden

Gerrie W.J. van de Ven

Marcel Lubbers

Antonius G.T. Schut

[Introduction 2](#_Toc43384670)

[Directory structure 3](#_Toc43384671)

[Data formatting requirements 4](#_Toc43384672)

[Meta data 4](#_Toc43384673)

[Data privacy issues 5](#_Toc43384674)

[Set-up script 5](#_Toc43384675)

## Introduction

A good data management plan with a data storage and processing structure is an essential part of research. This document describes the recipe that is developed for research at PPS. This recipe helps to ensure that data, analyses and results are stored and documented in a systematic way and facilitates good scientific practices with re-usable data, transparent methods and repeatable results.

Proper data management contributes to making research more transparent and accessible and helps ensuring that ethical and quality requirements related to collection, storage and pre‑ and post‑processing data are met. Further, it is an essential part of open science where data can safely be shared with other users and needs full documentation describing legal restrictions on use and sharing, data collection protocols and description of the data with associated metadata.

The recipe aims to help you organise your data and set up data collection protocols in a structured manner which will ensure transparent use of methods and facilitates processing of data in a later stage. The investment in the implementation of the PPS-recipe will pay off in later stages of your research and allows supervisors and colleagues better to provide support.

The PPS-recipe allows flexibility, yet requires that data processing is done with scripts and not within spreadsheets. We strongly recommend to work with ASCII formatted files (.csv). processing of data in spreadsheets needs to be avoided. Our recipe automatically produces a Microsoft Excel (.xlsx) file as last step in the process, containing a single dataset with a sheet including a description of the data, a sheet with processed data and a sheet with metadata providing a unit (e.g. kg/ha) and description of what each value in the data columns represent (e.g. standardized maize grain yield with a 12.5% moisture content).

At PPS, we have developed a set of guidelines and structures that aim to ensure data is handled and stored uniformly. It is based around the following set of principles:

1. Integrity of raw data (stored in ASCII formatted files when possible)
2. Full documentation of the nature and content off all data files by providing meta-data and variable defenitions.
3. Reproducible pathway from (raw) data to results by commented scripts

In principle, each project such as a PhD thesis chapter, MSc thesis, or research paper should be associated with a self-contained directory structure containing all data, scripts, results and written output that can be archived and made available for storage or sharing.

Below is a detailed description of what this directory structure looks like, how data needs to be formatted and documented and how to set up scripts to handle and analyse data transparently. This distribution comes with a helper R script that can be sourced to make an initial project folder structure, and that includes some example data and scripts that demonstrate the entire path from setting up a project directory, to processing raw data to a distributable file with a meta data page to simple analysis that produces and stores results. **A brief description of the implementation of this script can be found at the end of this document**.

## Directory structure

The hierarchical directory structure associated with a project is presented in Figure 1. A central project directory with an informative name contains four main directories: *data*, *results*, *scripts*, and *writing*. The *writing* directory is meant to contain the project output document, i.e. a thesis chapter or research paper.

The *data* directory contains the sub-directories *definitions\_protocols*, *processed* and *raw*. The *raw* directory is the most important one, since it is here that files with all raw data should be stored. All raw data needed for the generation of project results should be stored in this directory, even if they are already stored elsewhere. The important thing to note about raw data is that has to be as close to initial data entry format as possible and should not be edited or handled by the user after it has been stored in the *raw* directory. There are however a set of guidelines that raw data is expected to adhere to. These will be outlined further below.

The *scripts* directory contains all computer code needed for data processing, cleaning and analysis. Ideally, they form a numbered set of scripts (e.g. “1. Raw data cleaning.r”) that, when run successively, reproduces all results referred to in the project output document (e.g. report, paper or chapter) from the raw data. The scripts should contain sufficient commentary to be human readable. Results generated by scripts are written to the *results* directory and include the number of the script (e.g. “1. Cleaned data.csv” or “1. Histograms of raw values.pdf”) for easy traceability. The analysis scripts write the tables and figures referred to in the project output document to the corresponding *results/tables* and *results/figures* directories. As a by-product of running the scripts, a set of intermediary or processed data files may be produced. These are written to the *data/processed* directory. This directory also contains the distributable copies of the data, including complete meta-data. Finally, the *data/definitions\_protocols* directory may contain documents describing the specifics of data collection such as protocols and detailed descriptions of how variables in the data are defined. Ideally, variable definitions that will be distributed with the data refer to these documents when needed. Statements related to ethical and privacy guidelines followed can also be stored here.

Figure 1 Hierarchical directory structure for storing all data, scripts, results and written output for a research project.

## Data formatting requirements

Raw data files need to adhere to a couple of basic formatting rules to ensure that they can be handled efficiently and uniformly. First of all, raw data in a single ASCII file has columns representing one single variable, factor or identifier. Use underscores instead of spaces in column names. Repeated observations on the same object or experimental unit (e.g. yields measured over different seasons) are recorded on separate lines by adding a row for each observation, repeating all information related to the object and adding a time column to indicate which date- and when required a timepoint the information in a row refers to. Dates are formatted as YYYYMMDD or YYYY-MM-DD to support sorting. Experimental data include a location, name of the experiment, treatment, plot, replicate, latitude and longitude, date of observation (etc).

Second, numeric variables have the unit added to the variable name, separated by a “$”, e.g. Dry\_grain\_yield$kg\_ha. Use standard (SI) abbreviations for units but avoid special symbols that may cause problems when handling and storing data. Use clear variable names and be consistent, e.g. Fresh\_grain\_yield$kg\_ha, Dry\_grain\_yield$kg\_ha or GrainYield$kgDMha.

**Please see the example data downloaded by the helper script to see what a data file should look like.**

## Meta data

Each raw data file is linked to a file containing extensive meta-data. Meta-data is meant to provide all information needed to make data usable for others. It includes a data identifier, information on the project and project funding, author and contact information, data type, species or subjects the data refers to, geographic areas and dates covered.

## Data privacy issues

Before distributing the data, it is important to identify data-fields or variables that may be privacy sensitive. This includes all variables that can be traced back to a private individual such as names, phone numbers, location data. These data fields need to be removed or anonymised when data is made available publicly.

## Set-up script

A standard *R* script is available that when run from the main project directory (with this directory set as “home”). It can be sourced from GitHub as follows (if you haven’t done so already):

1. Make an empty folder  
     
   2. Set it to working director in R  
     
   3. Paste the following in the terminal:

source("<https://raw.githubusercontent.com/jvheerwaarden/PPS_data_management/master/Master_Script_V1.1.R>")

This automatically downloads scripts and files that:

1. create the data management structures described above. Briefly, it creates the required directory structure, read any raw data files (.csv or .xlsx) stored in the project directory and place them in the *data/raw* directory.

The script also provides an example for the final processing step and processes the raw data into a corresponding processed .xslx file, containing worksheets with 1. meta-data with all required fields and 2. A list of all variables in the data with appropriate units and extra columns for variable definitions and privacy sensitivity. Please check the “example\_data.csv“ and “Meststof proef WUR.csv” data files in the *data/raw* directory and the associated “example\_data\_metadata.xlsx“ and “Meststof proef WUR\_metadata.xlsx” files in the *data/processed* directory.

2 It places a set of R scripts in the *scripts* directory (numbered 0-2, and a functions file) that provide examples of reading, cleaning and analysing data. They are run automatically (by sourcing the *0\_build\_project.R* file) and output is written to the *results/figures* and the *results/raw* directories. There is also a separate *RawDataConvertScript.R* that when sourced, converts raw files in the working directory or *data/raw* directory into an *.xlsx* file with meta data as described under point 1. Please have a look at what the scripts do to get an idea of the logic behind them.

3. It places the current Document and Readme in *writing* directory.