MIND STEP

MODELLING INDIVIDUAL DECISIONS TO SUPPORT THE EUROPEAN POLICIES RELATED TO AGRICULTURE

Deliverable D2.2: A guide/handbook to build an interface for accessing the data in the project required by partners WP 2-6

AUTHORS	Gocht, Alexander (THÜNEN), Sebastian Neuenfeldt (THÜNEN) Xinxin Yang (THÜNEN), Marc Müller (WR) John Helming (WR), Gerbert Roerink (WR) Janssen Sander (WR), Diti Oudendag (WR), Dimitrios Kremmydas (JRC) Albert Brouwer (IIASA)
APPROVED BY WP MANAGER:	Gocht, Alexander (THÜNEN)
DATE OF APPROVAL:	30.04.2021
APPROVED BY PROJECT COORDINATOR:	John Helming (WR)
DATE OF APPROVAL:	30.04.2021
WP NUMBER & TITLE	WP 2 Data requirements for indicators on European policies related to agriculture and data management
DISSIMINATION LEVEL	Public
PROJECT WEB SITE:	https://mind-step.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

This document was produced under the terms and conditions of Grant Agreement No. 817566 for the European Commission. It does not necessary reflect the view of the European Union and in no way anticipates the Commission's future policy in this area.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 773901.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

TABLE OF CONTENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

LIST OF FIGURES

LIST OF TABLES

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

ACRONYMS

ABM	Agent-based model
ADC	AgroDataCube
Agripolis	Agricultural Policy Simulator
AgroDataCube	AgroDataCube provides a large collection of both open data and derived data for
API	Application Programming Interface
Best4Soil	The network of practitioners for sharing knowledge on prevention and control of
BIN	Bedrijfsinformatienet (Dutch version of FADN)
CAP	Common Agricultural Policy
CAPRI	Common Agricultural Policy Regionalised Impact Modelling System
CGE	Computable general equilibrium
CIRCABC	Communication and Information Resource Centre for Administrations, Businesses
CLC	CORINE land cover
COM	Component Object Model
CORINE	Coordination of Information on the Environment
CSV	Comma separated file
DEM	Digital Elevation Data
DevOps	A set of practices that combines software development (Dev) and IT operations
DG-AGRI	The Commission's Directorate-General for Agriculture and Rural Development
DSS	Dynamic Steering System
DWH	Data warehouse
EAA	Economic accounts for agriculture
EPIC	Environmental Policy Integrated Climate Model
ESDB	European Soil Database
EU	European Union
EU COM	European Commission
FADN	Farm Accountancy Data Network
FAO	The Food and Agriculture Organization
FAOSTAT	Food and Agriculture Organization Corporate Statistical Database
FARMBOSS	FARM Betriebs Optimierungs und Simulations Software (farm level optimization and
FARMDYN	A dynamic mixed integer bio-economic farm scale model
FSS	Farm Structure Survey
GAMS	General Algebraic Modeling System
GDX	GAMS Data eXchange
GEMPACK	General Equilibrium Modelling PACKage
GGIG	GAMS Graphical User Interface Generator
GIO	A Graphical User Interface for GLOBIOM
GIS	Geographical Information System
Git	A software for tracking changes in any set of files, usually used for coordinating work
GitHub	A provider of Internet hosting for software development and version control using
GitLab	A web-based DevOps lifecycle tool that provides a Git-repository manager providing
GLOBIOM	The Global Biosphere Management Model
GMS	A GAMS script file with GAMS code
GNU	An extensive collection of free software, which can be used as an operating system
GPL	General Public License
GSAA	Geospatial Aid Application
GTAP	Global Trade Analysis Project

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

GUI	Graphical User Interface
HNVF	High Nature Value Farmland
IACS	Integrated Administration and Control System
IBA	Important Bird Area
IDM	Individual decision making
IES	Institute for Environment and Sustainability
IFM-CAP	Individual Farm Model for Common Agricultural Policy Analysis
IMAGE	Integrated Model to Assess the Global Environment
IT	Information Technology
JRC	Joint Research Centre
JSON	JavaScript Object Notation
KWIN	Kwantitatieve Informatie voor de Akkerbouw (Dutch management data)
Landsat-MSS1	Landsat Multi Spectral Scanner
LBT	Landbouwtelling (Dutch agricultural census)
LEITAP	A further development of the GTAP model
LPIS	The Land Parcel Identification System
LUCAS	Land Use/Cover Area frame Statistical Survey
MAGNET	Modular Applied GeNeral Equilibrium Tool
MIP	Mixed-Integer Programming
mmU	minimum mapping units
MS	Member States
NASA	National Aeronautics and Space Administration
NDVI	Normalized Difference Vegetation Index
NIR	near infrared
NUTS	Nomenclature of Territorial Units for Statistics
OECD	The Organisation for Economic Co-operation and Development
OS	Operating system
PBA	Prime Butterfly Areas
PowerBI	A business analytics service by Microsoft
PRIMES	Price-Induced Market Equilibrium System
PUF	Public use files
R	Programming language and free software environment for statistical computing and
RDC	Research Data Centre
REST	Representational state transfer
SAPM	The Survey on agricultural production methods
SGDBE	Soil Geographical Database of Eurasia
SMU	Soil Mapping Units
SPAM	Spatial Production Allocation Model
SSP	Shared Socioeconomic Pathways
STU	Soil Typological Units
SUF	Scientific use file
SVN	Apache Subversion
UNFCCC	United Nations Framework Convention on Climate Change
VHR	very high resolution
XML	Extensible Markup Language
XSD	XML Schema Definition
YML	A human-readable data-serialization language

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

EXECUTIVE SUMMARY

1. INTRODUCTION

   The aim of this deliverable is to build a conceptual framework for database interfaces for relevant information in MIND STEP. The purpose is to give a guide for interface development, which is setup such that i) it integrates data from multiple heterogeneous sources at flexible geographic and regional scales and ii) that it supports analytical reporting and allow structured and/or ad hoc queries. The approach of the deliverable is to review exiting approaches in the field of 1) harmonized databases for socio-economic micro data 2) exiting geo-reference and geo-spatial databases, like, IACS reporting and monitoring system for agricultural subsidies and remote sensing-based products as well as the AgroDataCube and 3) exiting databases and data interfaces for already established models. Based on that we propose a concept, by summarizing the requirements for the project and deriving suitable interface solution. This concept is then used to build different interfaces, delivered in D2.4, and documented using, so called, “use cases” in Chapter 3.

2. REVIEW

3. Chapter: 2.1. Review of socio-economic databases and interfaces for IDM modelling

   We discuss mainly socio-economic database interfaces for the farm data accounting data network (FADN) and the farm structure survey (FSS). The interfaces so far have been developed at ad-hoc basis and have not been made so far available to other potential users via clearly defined channels like GITHUB, GITLAB or SVN. In addition, missing documentation made it hard or impossible to adjust for the purpose of other research projects. Also, a list of use cases, which document how to use and apply the interface were missing. Another problem identified in the review was that when financial support ended, and confidential data had to be deleted, the maintenance of the interfaces could not be continued. Given that a micro data access to EU wide FSS data of EUROSTAT is still in an earlier phase we decided to first develop a micro data interface to the FADN data, building on the interfaces developed for IFM-CAP. The interface was tested with the FADN data provided in April 2021. However, to also prepare for the work of micro FSS data we build a prototype for micro FSS data from the national data provider in Germany, for which THÜNEN has access.

4. Chapter: 2.2. Review of bio-physical and environmental impact relevant data at high resolution and related interfaces for IDM modelling

   For the review of bio-physical and environmental impact relevant data at high resolution we first discuss exiting EU-wide geo databases potentially relevant for IDM modelling and provide references and interfaces where accessible. We discuss the Integrated Administration and Control System of the EU for the CAP and the terms of condition to use it for IDM development. The 2013 CAP reform made it compulsory to use this **Land Parcel Identification System** (LPIS) together with a **Geospatial Aid Application** (GSAA) as components of the paying agencies integrated administrative and control systems (IACS), introduced progressively from 2015 to enhance checks of aid applications. IACS can be a valuable resource for analysing the structure and the land use of certain region at the parcel level and hence also a valuable source to parameterize farm level model. As the access is quite restrictive and organised at regional administrative level of the single member states but make the data public, we included a table summarizes sources freely available for download. In addition, we discussed remote sensing data nowadays freely available. The new imaging technologies with the EU-owned Copernicus Sentinel satellites, known as Sentinels, had become a new source of data for monitoring the

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

Common Agricultural Policy and hence also for IDM modelling because automated processing of time series throughout the growing season makes it possible to identify crops and monitor certain agricultural practices on individual parcels or even at higher resolution (10x10 meters) EU-wide (Devos, 2017,2018a,2018b). Although the Commission promoted the technology through many conferences and workshops in 2019 only 15 out of 66 paying agencies used the Copernicus Sentinel to check aid applications and an audit of the Commission revealed that many paying agencies consider that there are obstacles to wider use of the new technologies (ECA, 2020). Currently more and more products, like crop maps and moving events at larger regional scale and continuedly updated are available also for parameterization. As so far, no EU-wide products are currently available, but under development, using the interfaces and packages from Sen4CAP[1] we focus on the application and use of a national development of the Netherland, named the green monitor, which was later integrated in the **AgroDataCube**. The **Green Monitor** data platform (www.groenmonitor.nl) started in 2012 to map the Netherlands with high resolution satellite imagery. The Green Monitor is developed as an easy-to-use webtool for visualization and interpretation of time series of NDVI satellite images covering the Netherlands. The NDVI is a measure of the amount of green biomass. From 2016 onwards the Green Monitor is collecting and processing high resolution imagery from Sentinel-2 and Landsat satellites. Here interesting for IDM-modelling is the product on the growing season of the crop. In case of grassland the derived markers are the number of mowing cuts, the date of first, second and later mowing events, the grass ploughing and renewal and a yield indicator. In 2018, version 2 of the AgroDataCube has been developed. Through integration with Green Monitor, the AgroDataCube now also provides a remote sensing-based vegetation index (NDVI) at sub-parcel resolution. Such vegetation indices are used for research, e.g., crop modelling and yield forecasting, by farmers to monitor the development of their crops, or to monitor agricultural practice, e.g., complying with CAP regulations. The AgroDataCube functions as a hub that brings together these heterogeneous data streams, enriches them, adding in-house analytics, and publishes the result as harmonized, up-to-date, standardized datasets accessible through an open **REST API** (agrodatacube.wur.nl). The deliverable reviews the current approach and is an interesting application also with respect to the technical implementation via a REST API.

5. Review interfaces for IDM modelling to existing and established modelling databases

   Besides the micro economic data bases and geo-spatial data, a large pool of interesting information for IDM modelling is provided in the databases developed for existing modelling platforms like Magnet, GLOBIOM and CAPRI.

   We first review the interfaces development of the Global Biosphere Management Model (**GLOBIOM**), which has been developed and used by the International Institute for Applied Systems Analysis (IIASA) since the late 2000s. Through stages of filtering and aggregation, the output data is distributed across a series of GAMS parameters that are the representational near equivalent of Python/R data frames as well as a series of supportive set definitions. To interface the GLOBIOM database a R routine have been developed, which allows GDX content to be explored, read, and written. On top of a GLOBIOM visualization interface is provided by the globiomvis **R package**. It supports analysis and generation of a variety of scenario plots. In addition, globiomvis enables creation of maps for the various regionally and spatially explicit representations of the model. In addition, and for an interactive exploration a graphical user interface (GIO) based on **GGIG** is available as an alternative way of performing analysis, visualization, and parameterizing and running the core version of the model.[2] GGIG is Java-

[1] http://esa-sen4cap.org/content/download-package-description
[2] https://github.com/iiasa/GLOBIOM_GUI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

based and orchestrates numerous Java libraries that provide rich visualization and analytical functionality.

Another example is the **CAPRI** data base. The CAPRI agricultural economic model has been developed since 1999 with the help of several EU research projects. The model supports the policy-making process by means of quantitative analyses of the EU Common Agricultural Policy (CAP) at global and regional levels. The aim is to estimate in advance the impact of agricultural policy decisions on production, income, trade, and the environment using the model.

The interfaces to access the databases are currently based on a graphical user **interface GGIG** (Britz, 2014), like the way GLOBIOM assess the data.[3] The GUI comprises a generic and powerful tool for exploitation, accessing and exporting of results. In addition to the GGIG there exists an interface to import the result database of CAPRI into Excel using an **COM add for Excel** based on the GDX API of GAMS and the MS .net framework and a **R package** interface to access the CAPRI database developed by Mihaly Himics. The R package reads GDX files and prepares figures and charts. It includes functions for processing, visualizing, and analysing both the model databases and simulation results. It has been designed to complement (rather than to replace) already existing GIGG for CAPRI.

The **MAGNET** model is a global general equilibrium model. MAGNET is based on the LEITAP model which has been used extensively in policy analyses. MAGNET uses a series of additional databases. The database is constructed in several steps. A new concept has been developed for MAGNET output: writing MAGNET scenario results to the central Datawarehouse using SQL Server Integration services. The data is converted and processed and finally stored into the Data warehouse (DW). Finally, the data is accessible (authoring) using different applications **like SQL, Power-BI, R, Python**. One of the applications is **Power BI** with additionally PBI-report server. With this latest tool visualisations of project data can be made available for users/clients outside Wageningen Economic Research. However, these reports contain predefined figures and tables. Therefore, and already in development, is **OData webservices** is introduced with which users from outside Wageningen Economic Research can also query the data.

6. Chapter 3 CONCEPTUAL AND TECHNICAL FRAMEWORK INTERFACE DEVELOPMENT

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. GUI, or UI, stands for Graphical User Interface. Modelling systems reviewed in Chapter 1 often using the graphical user interface, but it requires numerous and time-consuming interactions. As software doesn't need a graphical user interface to communicate software products can also exchange data and functionalities via machine-readable interfaces. R packages have been reviewed in Chapter 1, as an example of an API. A particular API which operates via the internet to communicate between application is the REST API. In Chapter 1 we have observed that the AgroDataCube, MAGNET with OData, but also the service to sentinel data uses API Rest technologies. However, it is somehow limiting for highly sensitive data like FADN.
After two workshops we opted for the following API interface concept: As a graphical user interfaces are too restrictive and API in a classical form, programmed in .net, C++ or java requires IT knowledge, which is often not available amongst modellers in the field of agricultural economics, we identified R packages as the best alternative. The programming

[3] https://www.ilr.uni-bonn.de/em/rsrch/ggig/GGIG_user_Guide.pdf & https://www.ilr.uni-bonn.de/em/rsrch/ggig/GGIG_programming_guide.pdf

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

language R is quite open and less restrictive, well-known, maybe also taught already at the university, free of charge, it runs cross platform and is easily applicable by learning from internet sources. It also provides the functionality to include Python. R also can be interlinked to GAMS, which was identified in WP7 as the main language for agricultural IDM modelling.
With R packages we can account for different needs of groups involved in development, maintenance, and application of the interfaces. As example the developers of the interface, needs to have a shared distribution system to commonly develop and extent the interface and deploy. The user group will use the API in a more applicable way by loading the API and applying without any need of changing the function of the interface itself. With the package deployment in R this concept is easily applicable. For both clients a **good documentation** is required. To support the documentation process for the interface, it shall be to a certain extent generated in an automated way and build up on inline documentation. Besides the documentation of the interface also **a use case documentation** is of importance. The R-Mark down approach allows easily to compile use cases and make them in different formats available, as provided in the next Chapter.

7. Chapter 4 DESCRIPTION OF THE DEVELOPED PROTOTYPE INTERAFCE VIA USE CASES

In this Chapter the use cases are presented for five developed data interfaces based on the concept developer in Chapter 3. The first interface is the FADN data interface. To use the package, you can install it directly from R. The **fadnUtils package** facilitates the efficient handling of FADN data within the R language framework. This means that there is a specific temporal pattern of how a user interacts with the package. More specifically, after a request for FADN data in csv format. The first step is to import the data into an R-friendly format. In a next step you can convert the R in a human-readable file. In this file the user can define what columns to keep and what transformations or new calculations to make. Using a JSON file the raw can been converted into a structured human readable file, whereas the JSON file defines the rules how the raw that is modified. The use case provides then various examples of the powerfulness of using the fadnUtils package to perform data analysis on FADN data. In A second subsection we perform using the fadnUtils package basic analyses on the provide FADN data, e.g., analyse the number of countries and years - compare with the requested data, check the number of provided variables, check for missing variables, mean of all provided variables, number of sample farms or count the exit, stay and entry of farms for the years of identical farms.

The second API is a **FSS package** and has the aim to provide functions to analyse German micro–Farm Structure Survey (FSS) data. As of the end of April 2021 there are two functions so far implemented in this package. One converts the (usually in csv format) provided raw data into a Rdata file, which is easier and faster to handle in R. The second one is presented in this chapter and provides a fake data set to use and test some analysis before touching the real FSS data.

One exception is the **interface to management data**. The parameterisation of the different technologies and investments is data demanding and require the knowledge of management data, technical coefficients usually collected by different players in the field and are not available in statistics such as FADN or FSS. Instead of providing an explicit API we provide a summary of information on accessibility of management data and their respective access policies. Various types of management handbook data that can be found in the case study regions are only available on paper and/or in their own language. Availability of the management handbooks in the different case study regions in MIND STEP and the possibility to use are summarised in this subsection. We identified 13 different sources for management data across the case study regions, covering operations investment machinery costs, biogas

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Content of nutrient for feed and crop product, renewable resources used to produce bio-based products and/or bioenergy, guide prices for new construction and reconstruction, of agricultural farm buildings and rural dwellings, agricultural machinery catalogue, investment costs, earnings and expense ratios, detailed gross margin and labour requirement calculations for a large number of conventional and organic arable and open-air vegetable crops, fixed costs for land, buildings, tractors, machinery, labour and hired labour and much more. For each of the identified sources the internet link is provided. We also provide an example how these is connected to the IDM model FARMDYN.

Within the MIND STEP project, FARMDYN has been adopted for the Netherlands by using the Dutch version of FADN for general farm characteristics like endowments, yields, and cost structure. Further sources of information are the KWIN for management data as discussed in the previous section. However, detailed data on grassland activities is missing. A solution is the use of the Dutch AgroDataCube (ADC) database, which includes the timing of mowing events and indications of annual dry-matter yields. This permit deriving the grassland management tables required by FARMDYN. In this subsection we describe the structure of the used data and the processing steps required to align the AgroDataCube and BIN data to populate the grassland management attributes for the FARMDYN model. The use of satellite images from the **AgroDataCube interface** has proven to be very useful for deriving grassland-related parameters for the FARMDYN model because it permits to distribute average farm-level yields to several management systems, distinguished by the number of cuts.

About half the models involved in MIND STEP use GAMS as the core language. Over half of the models use R as an additional scripting language or are even based on R. This, together with the capabilities offered by R and its package ecosystem, makes R the natural choice for implementing code that transforms, manipulates, and analyses data derived from models and for code that glues models together. In the last interface, we present how an R package can be used to extract GAMS data and making it available for processing in R, using **GLOBIOM** data as an example. Note that the discussed R processing methods are not unique to GAMS and can serve any model-derived data that can be represented as a data frame, hence also for CAPRI output.

8. Chapter 5 CONCLUSION
   With this deliverable we also finish the prototype phase for the interface development in MIND STEP. We will review with the MIND STEP partners the current interfaces and define further adjustment. Besides, the following issues will be relevant for the finalisation: The FSS interface need to be adjusted to the micro FSS data from EUROSTAT. In addition, we will centralize all code developments on the GITLAB of IIASA, which was not operational for the prototypes. Further FadnUtils will be finalized such that the structured human readable file based on JSON fits to the requirements of the modelling teams in MIND STEP. The prototype for accessing GDX files using the example of GLOBIOM will be used, in combination with other already developed packages, to provide a comprehensive interface for the CAPRI data base.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

1. INTRODUCTION

The aim of this deliverable is to build a conceptual framework for database interfaces for relevant information in MIND STEP. It will allow an easier access to information for different IDM developments and farm modelling approaches. Given the independent existence and continuous changes of the database, including the increasing availability of high spatial and temporal resolution farm and biophysical data, we aim at building database specific interfaces instead of building one big database, to avoid the risk that the work is soon outdated and hence difficult to maintain. The purpose is to give a guide for interface development, which is setup such that i) it integrates data from multiple heterogeneous sources at flexible geographic and regional scales and ii) that it supports analytical reporting and allow structured and/or ad hoc queries. The guide for interface development is structures like the task included in the work package. Chapter 1 discusses potential databases and existing solution as a review. The first subchapter has a strong focus on EU and national wide harmonized databases for socio-economic micro data, which are based on farm surveys. It starts with a discussion about interfaces for FADN and FSS micro data. We examine under which circumstances FSS and FADN data has been made accessible to the research community. In addition, we review models which made intensive use of this data and describe the developed interface solutions. In this scope and although FSS and FADN include a lot of useful information some important information is not collected but needed to parameterize IDM models. This holds particular for production method and related technology information. Hence, we also review the accessibility of management handbook data and interfaces to be comprehensive. In a second subchapter we shortly review exiting geo-reference and geo-spatial databases as LUCAS and CORINE used for describing the natural conditions of a farm. Further and initiated by the EU Common Agricultural Policy (CAP) we review the IACS reporting and monitoring system for agricultural subsidies. IACS includes at high-resolution payments and related farming activity across Europa partially at the parcel or plot level. Unfortunately, IACS is organized by each country in the EU and hence do not share the same technical and organisational systems. This and the high level of data confidentiality often restricts the use of IACS for research. But there are circumstances and countries, where such data are publicly accessible and part of a comprehensive geospatial database solution. AgroDataCube is one example which we review. In the last subsection of Chapter 1 we discuss exiting databases and data interfaces for already established models and their database. In Chapter 2 we develop from a technical point of view, what we understand as an interface and list and discuss desirable properties using the review as a starting point. Chapter 3 will then combine the conceptual approach of Chapter 2 using different use cases to build application programming interfaces. The work in WP 2 for interfaces is structures in a first phase prototype interfaces development and in a second phase to finalize the work. The code of the programming interfaces for the prototype are delivered D4.2. This structure allows a feedback with the partners of the consortium before finalising the interfaces. To streamline the discussion, we list potential further adjustments of the interfaces in Chapter 3 in the conclusions.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

2. REVIEW

2.1. Review of socio-economic databases and interfaces for IDM modelling

{lead THÜNEN}

The discussion in MIND STEP emphasised that access to agricultural micro data is of high relevance to develop IDM models, particular to increase the flexibility and to analyse distributions rather than average aggregated values. EUROSTAT has received **farm structure survey** micro data from the member states of the EU[4] since 1990 (each 3-5 years with full surveys). A detained catalogue of variable and related meta data is available online[5]. From an email exchange with EUROSTAT[6] , they confirmed that the Farm Structure Survey (FSS) micro data is now also accessible for research. The data is granted only for scientific purposes and when the organisation is accepted by the EU as a research entity[7]. To access the data there are two ways. The data can be partially anonymised or non-anonymised. The latter is only accessible in Eurostat's "Safe centre" in Luxembourg. Scientific use files (SUF) containing records on individual farms in such a way that the risk of identification of the farm is reduced, but the data has been modified. Such files are usually sent to the researchers on CDs or can be downloaded. The non-anonymised files are highly confidential and still a combination of some variables may lead to an identification of the surveyed farm. The potential results of the analysis at the "Safe centre" in Luxembourg are controlled by the staff of the statistical office before it is published and provided to the researcher.

In addition, there are public use files (PUF) anonymised in such that the respondent cannot be identified either directly or indirectly. PUF are not confidential and in principle public domain, however, due to extensive anonymisation they are not very useful for scientific purposes [8]. Nevertheless, this file might be used as dummy data set for interface solutions.

Another option to access the FSS micro data is offered via national data providers[9]. But this restricts the analysis to the national domain. The access condition varies amongst the national agencies. In some cases, the regulations are strict, and access is only possible in so called "Safe centre". In addition, the format, and items of the FSS catalogue can differ compared to the EU data format. A summary how the different countries do provide access to the micro data is unfortunately not available.

The second micro database on economic performance in the agricultural domain is the **farm accountancy data network** (FADN)[10]. It monitors farm' income and business activities. It is also an important informative source for understanding the impact of the measures taken under the CAP. FADN is the only survey of microeconomic data based on harmonised bookkeeping principles across Europe. All relevant documents and information on FADN are accessible in CIRCABC in a public domain area [11]. As bookkeeping principles are evolving the documentations also changes. The newest document for the description of the catalogue of variables was provided in May 2020 for the farm

[4] In 2010 a special survey, the Survey on agricultural production methods (SAPM) was carried out. SAPM was carried out together with the 2010 census in some countries, whereas in other countries SAPM was carried out as a sample survey and data were linked to data of the census at the level of the individual holding to enable cross comparisons of variables collected in both SAPM and the census (https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:Survey_on_agricultural_production_methods_(SAPM)). See also https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32008R1166. To the knowledge of the authors the data is available at national data providers, at least in Germany, but not at EU level.

[5] https://ec.europa.eu/eurostat/cache/metadata/en/ef_esms.htm

[6] 25/02/2021 information received from ESTAT-Microdata-access@ec.europa.eu

[7] https://ec.europa.eu/eurostat/documents/203647/11490892/FSS_anonymisation.pdf/be9e69c2-554f-191d-ace6-af0e03fdb231

[8] https://ec.europa.eu/eurostat/web/microdata/public-microdata

[9] http://www.forschungsdatenzentrum.de/en/latest-news

[10] The Farm Return is specified in Commission Regulation (EEC) No 2237/77 of 23 September 1977 and subsequent amendments until the year 2008 accounting included, then in Regulation (EC) 868/2008 from the financial year 2009.

11https://circabc.europa.eu/ui/group/880bbb5b-abc9-4c4c-9259-5c58867c27f5/library/34866b54-76e8-4984-9eea-5aad44e3ffa7?p=1&n=10&sort=versionLabel_ASC

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

return in 2019 but also older versions are available. For MIND STEP we applied for data until 2017/18. At this location document for the methodology can be downloaded. Like FSS the Commission does not directly collect data itself. This is the responsibility of a Liaison Agency in each Member State. To ensure that this sample reflects the heterogeneity of farming it is stratify by three criteria: region, economic size and type of farming. Farms are selected in the sample according to a selection plan that assures its representativity of the farm population, represented by FSS. FADN adopts an approach by including only farms deemed to be commercial above a cut-off limits, measured in economic size. An extrapolation factor (weight) is calculated for each surveyed farm. In addition, standard results are a set of indicators, calculated from the catalogue and are available for download from CIRCABC. They describe the economic situation of farmers by different groups and regions. Data at the level of individual farms are normally not released outside the Directorate General for Agriculture of the Commission, exception are granted for research projects. The FADN survey catalogue are extracted from inventory, cash book or journal kept by the farmer or field officer and are grouped into 13 tables[12]. Compared to FSS, which consists of structural information for all farms in the, FADN consists of bookkeeping data for a subset of farms. The data is collected for each year.

There are two main models in the EU which make intensive use of FSS and FADN at different aggregations. With the availability of adaptable aggregated data from the farm accountancy data network (FADN) and aggregated data the farm structure surveys (FSS) the farm type module for Common Agricultural Policy Regionalized Impact model (CAPRI-FT) was developed (Britz and Witzke 2014, Gocht and Britz 2011). A higher resolution of the supply model has been achieved by defining farm types, which in turn are represented by a nonlinear programming model that captures all activities associated with the farms of a certain typology in a specific region. The CAPRI-FT share the same model template (economic modelling assumptions) across regions, only the parameterization differs. The model comprises about 2,900 farm types in the EU. Each is characterised by its specialisation and economic size. To parameterize the model's interfaces to three main data sources has been established in a java-based program[13]: to a comprehensive aggregated data set from the Farm Structure Survey (FSS)[14] (cropping hectares, animal herds) by type and economic size and Nuts2 region, aggreged FADN[15] data for yields and costs of particular by type, size and Nuts2. Contrary to the CAPR-FT, where the main data source was FSS the IFM-CAP model further increased the resolution by building a strong link to the micro data of FADN. This was possible as IFM-CAP is an EU-inhouse JRC development with access to micro level FADN data (Louhichi et al. 2015, Louhichi, Espinosa, et al. 2018). The research group of IFM-CAP used in the first version the java interface also used by CAPRI-FT to covert the data into a GAMS readable format. This recently changed and a new interface realized in R as package library was developed called FADNUtils (reference Dimitri). The IFM-CAP model aims at an EU-wide individual farm-level model coverage. The model is applied to each of the 80.000 individual FADN farms. The primary data source is the micro data FADN complemented by additional EU-wide data sources such as aggregated Eurostat data and the CAPRI model database (Louhichi, Espinosa, et al. 2018). Like CAPRI-FT, IFM-CAP applies a GGIG based user interface to configure and run working steps (raw-FADN data processing, construction of the model database, calibration, scenario runs). Although this has been standardized to some extent by using GGIG, less attention has drawn on the interfaces to input data. The interfaces have been developed at ad-hoc basis and have not been made so far available to other potential users. In addition, missing documentation made it

[12] Table A: General information; Table B Type of occupation and Breakdown of the farm area: owned, rented or sharecropped; Table C: Labor; Table D: Assets; Table E: Quotas and other rights; Table F: Debts; Table G: Value added tax (VAT). Table H: Inputs; Table I: Crops; Table J: Livestock production: Table K: Animal products and services: Table L: Other gainful activities; Table M: Subsidies.
[13] Part of GIG java source code
[14] FSS provides harmonized data regarding the structure of agricultural holdings in terms of land use, livestock numbers, farm labor force, machinery and equipment, and participation in rural development programs. The complete agricultural census is updated every 10 years (with intermediate sample surveys).
[15] FADN provides accounting data for a sample of commercial agricultural holdings. The survey is conducted annually.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

hard or impossible to adjust it for the purpose of other research projects. A list of use cases, which document how to use and apply the interface were missing. Besides, the data format, content and structure of the databases often also changed and became outdated. When financial support ended, and confidential data had to be deleted. The developed interfaces for FADN data operated on micro data (Neuenfeldt, 2014). In CAPRI-FT and IFM-CAP the bio-physical representation is limited and hence no interfaces to management and production data bases have been developed. Although the methodology to depict technologies in IDM models[16] is known, gathering, and collecting the required information is cumbersome, when focusing on several regions. Often information on management, related costs and technical coefficients is when available only in national languages and rarely[17] accessible in digital form. Most data collections on management are based on expert interviews and field experiments and information on the adaptation in the farm community is limited. In this context, production surveys like SAPM can be useful to establish a good understanding on the existing and dominating technologies to interlink management data with from FSS and FADN.

The interface development in WP2 is split into two development phases. This allows us to start with interfaces for which the project members have access. Given that we still in an earlier phase to apply for micro data access for FSS (and maybe for SAPM) we will first further develop the micro data interface to the FADN data, building on the interfaces developed for IFM-CAP. In addition, we focus on an interface for micro data from the national data provider in Germany, for which THÜNEN has access. In addition, we collect amongst the partners information on the accessibility to management data to recommend on how potential data access and interfaces could look like in the future.

2.2. Review of bio-physical and environmental impact relevant data at high resolution and related interfaces for IDM modelling

{lead, WR, S. Janssen}

We first name exiting EU-wide geo databases potentially relevant for IDM modelling and provide references and interfaces where accessible. Then we discuss the Integrated Administration and Control System of the EU for the CAP and the terms of condition to use it for IDM development. We follow with a review of existing remote sensing products and at the end discuss the AgrodataCube, as an example for combining different geo spatial data sources in a way to provide a consistent geo spatial data cube for data analysis in the context of the MIND STEP model toolbox.

2.2.1. Geo data at EU level

This gives a summary of the available databases in the EU and are described in subsequent sub-sections.

Table 1: Summary table for the spatial data basis for the allocation of farms

Name of the geo database	Description	Link and reference	Classes description
Corine Land Cover	Visual interpretation of satellite images and divides land use 44 classes	http://land.copernicus.eu/pan-european/corine-land-cover	44 classes
LUCAS	(Land Use/Cover Area frame	http://esdac.jrc.ec.europa.eu/projects/lucas	Lucas classes

[16] field operations, manure handling, ruminant production and grass land management, fertilizer application technologies, housing and feeding regimes

[17] Exceptions exists as for the management data from KTBL, which have been made available as bulk to the THÜNEN Institute in form of a database. The use and access however are restricted. However, there a web interface, whereby manual selection management data is provided as wen table. www.KTBL.de

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

	Statistical Survey)		
Ecosystem type map v2.1	based on CORINE Land Cover 2006	https://www.eea.europa.eu/data-and-maps/data/ecosystem-types-of-europe-1	Relevant classes: I1: Arable land and market garden, I2: Cultivated areas of gardens and parks
High Nature Value Farmland	based on CORINE Land Cover 2006	http://www.eea.europa.eu/data-and-maps/figures/estimated-high-nature-hnv-presence	For each combination of country and environmental zone, CLC classes are identified (out of in total 19 CLC classes) that are likely to contain primarily HNV land.
SMU	Soil mapping Units	http://esdac.jrc.ec.europa.eu/content/european-soil-database-v20-vector-and-attribute-data	Single soil component and similar soils
DEM	European Commission, JRC-IES Digital Elevation Model (CCM DEM, 250 meters), received 2004	European Commission, JRC-IES Digital Elevation Model (CCM DEM, 250 meters)	Own compilation of altitude zone 0–300 m, 300–600 m and >600 m based on Digital Elevation Model
Less Favoured Areas (LFA)	LFA boundaries map	European Commission, JRC, LFA boundaries map, received 2006, http://www.eea.europa.eu/data-and-maps/figures/less-favoured-areas/fig11_less-favoured-areas3_graphic.eps/image_large (12 Jan 2005)	Mountain/hill area (article 3.3); Less favoured areas in danger of depopulation (3.4); areas with specific handicaps (3.5); non less favoured areas

Source: Own compilation.

2.2.1.1. Corine Land Cover (CLC)

CORINE Land Cover (CLC) is a geographic land cover/land use database encompassing the countries of the European Community. CLC was elaborated based on the visual interpretation of satellite images. Proposed in 1985 by the European Commission, CLC describes land cover (and partly land use) according to a nomenclature of 44 classes. First estimation was undertaken in 1990. An update of the CLC database has been launched in 2000 and 2006. Ancillary data (aerial photographs, topographic or vegetation maps, statistics, local knowledge) were used to refine interpretation and the assignment of the territory into the categories of the CORINE Land Cover nomenclature. The smallest surfaces mapped (minimum mapping Units-mmU) is 25 hectares (Linear features less than 100 m in width are not considered. The scale of the output product was fixed at 1:100.000. Thus, the location precision of the CLC database is 100 m). The following CCL classes refer to agriculture: 211 agriculture, 212 irrigate agriculture, 221 vineyards, 231 pastures, and various mixed classes like class, heterogenous agricultural areas (4 sub-classes). The main drawbacks of the using CLC approach is beside the large time span between the surveys (1990 to 2000, update 2006, 2015) and the mmU of 25 ha that does not consider small agricultural structures (areas) the fact that the classes are rather aggregated and therefore less suitable for describing a particular farming system which needs to be explicitly known cropping shares. Furthermore, no direct information of livestock herd sizes is there and needs to be approximated somehow from the fodder areas. However, grassland classes are not suitable for determination of agricultural use.

2.2.1.2. Land Use/Cover Area frame Statistical Survey (LUCAS)

LUCAS is an EU field survey programme supervised by EUROSTAT. LUCAS provides information on land cover and land use based on a sample survey. The specific land cover/use statistics are collected in co-operation with authorities from all EU countries. Since 2006 the survey has been undertaken to estimate the state of land use and cover in the EU and the dynamics of change. The LUCAS surveys are carried out in-situ every three years. This means that observations are made and registered on the ground.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Data on land cover and land use are collected at the point of observation and for a transect. For 250 meter long transects (normally oriented West to East) all changes in land cover are recorded. The surveyors record both areal land cover classes and linear elements. The LUCAS transect database consists of two samples. The first and larger data base (LTDr) contains 270,277 data points on the sequence of land uses and land cover. For a sub sample 1,283 data points not only the sequence of the land uses, and land cover changes is recorded but also the features' respective width (LTDw). In 2009, the European Commission extended the periodic Survey to sample and analyse the main properties of topsoil in 23 EU Member States. This topsoil survey represents the first attempt to build a consistent spatial database of the soil cover across the EU based on standard sampling and analytical procedures, with the analysis of all soil samples being carried out in a single laboratory (http://eusoils.jrc.ec.europa.eu/projects/Lucas/). Approximately 20,000 points were selected out of the main LUCAS grid for the collection of soil samples. A standardised sampling procedure was used to collect around 0.5 kg of topsoil ranging from 0-20 cm. The samples were dispatched to a central laboratory for physical and chemical analyses. The latest LUCAS survey (2012) covers all EU countries and observations on more than 270 000 points.

Three types of information are obtained:

1. Micro data: land cover, land use and environmental parameters associated to the single surveyed points,
2. Point and landscape photos in the four cardinal directions,
3. Statistical tables with aggregated results by land cover, land use at the geographical level; these estimates are based on the point data appropriately weighted.

The LUCAS data and other sources of specific land cover/use data are not always comparable mainly due to methodological differences. In Austria for example a comparison showed relatively little accordance between LUCAS data and national data (FSS and IACS). For example, LUCAS data estimated an increase in cropland over a period of years when the cropland area was known to have decreased, or LUCAS estimates of the extent of land use change towards settlements were ten time higher than the real ones. This is due to methodological differences many related to sample size; for example, the LUCAS grid of survey points is too coarse for tracking national detailed change at the national level and there have been insufficient adjustments with national data.

2.2.1.3. Ecosystem type map v2.1

The ecosystem classification is based on EUNIS and the proposal of ecosystem typology for the MAES working group. It considers mapping feasibility at European scale and keep compatibility with national mapping approaches (nested scales).

The basic geometric reference for the mapping of the ecosystem types is CORINE Land Cover transformed into the 100*100 m grid (using the CORINE land cover value of the pixel centroid as pixel class label). CORINE Land Cover classes are transformed into EUNIS classes based on detailed expert analysis, starting with the m:n crosswalk between EUNIS and CLC, additional georeferenced data (higher resolution compared to CLC) and thematic relation between land cover classes and the EUNIS classification system (improving the thematic resolution of CLC). The crosswalk between EUNIS classes and CORINE land cover classes was already developed from the ETC-BD and was used as starting point. The current version v2.1 of the Ecosystem type map v2.1 can be downloaded here: http://www.eea.europa.eu/data-and-maps/data/external/ecosystem-type-map-v2.1. A new version 3 is planned to be produced by in 2017. Though the ecosystem type map is mainly based on CORINE Land Cover 2006 it shows a significant higher geometric accuracy than CORINE Land Cover. A drawback is that grassland classes are not suitable for determination of agricultural use.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

2.2.1.4. High Nature Value Farmland (HNVF)

Some types of farmland are, because of their broad characteristics, inherently high in biodiversity. They are described by the general characteristics of low-input farming systems in terms of biodiversity and management practices and named as high nature value farmland (Paracchini et al., 2008). The conservation of biodiversity on these farmlands depends on the continuation of low intensity farming practices. The concept of HNV farmland ties together the biodiversity to the continuation of farming on certain types of land and the maintenance of specific farming systems. Typical examples include semi-natural grassland systems, traditional olive, vine, and fruit production, Dehesa, Montado and other wood pasture systems and extensive farming in boscage landscapes. The overall mapping effort is based as much as possible on existing Europe-wide datasets (CLC 2006, Natura 2000 sites, IBAs, PBAs, environmental zones). For some countries national specific information was used (specific examples are referred to subsequently). The current version of the European HVN farmland layer is based on CORINE Land Cover 2006 and can be downloaded. It is about to be updated based on CLC 2012 in 2016. The HNV dataset is suitable for pan-European analyses, e.g., CAP assessment. As based on CORINCE Land Cover the spatial resolution is quite coarse as the minimum mapping unit is 25 hectares.

2.2.1.5. Soil Mapping Units (SMU)

The European Soil Database (ESDB) consists of the most detailed and comprehensive soil data, distributed by the European Soil Portal of the Joint Research Centre (JRC) of the European Commission (Panagos er al., 2012). The database consists of four components, of which the Soil Geographical Database of Eurasia (SGDBE) at scale of 1:1 000 000 is of interest. The SGDBE contains a list of Soil Typological Units (STUs), which are described by variables (attributes) specifying the nature and properties of the soils, e.g., texture, water regime, stoniness. To represent this database in spatial layers at a scale corresponding to 1: 1 000 000, the STUs are grouped into Soil Mapping Units (SMUs) to form soil associations. Each SMU corresponds to a part of the mapped territory and as such is represented by one or more polygons. The data are freely available after registration via the JRC (http://esdac.jrc.ec.europa.eu/resource-type/european-soil-database-soil-properties) in vector data from 2001 and raster data from 2006 with a resolution of 1 km and 10 km (European Commission and the European Soil Bureau Network, 2004).

2.2.1.6. Digital Elevation Data (DEM)

The elevation data was provided by the Institute for Environment and Sustainability (IES) of the JRC. This digital elevation model (DEM) is the result of merging several DEMs with varying resolution to provide ample coverage across Europe, Turkey, and part of Russia. The DEM has a resolution of 250 m.

2.2.2. Integrated Administration and Control System (IACS)

{THÜNEN, A. Gocht}

To ensure regular payments, the CAP relies on the Integrated Administration and Control System (IACS), a set of comprehensive administrative and on-the-spot checks. The Land Parcel Identification System (LPIS) is a key component of IACS. It is system based on ortho-imagery (aerial or satellite photographs) which records all agricultural parcels in the Member States. It serves two main purposes: to clearly locate all eligible agricultural land contained within *reference parcels* and to calculate their maximum eligible area. Reference parcels are a uniquely identified and geographically delimited agricultural area. Farmers are expected to examine and to identify and exclude from their applications all non-agricultural land, and ineligible features on parcels. The LPIS's technical specifications vary amongst Member State as various types of reference parcel systems exist. The **Agricultural parcel system:** which covers a single field and s single famer; **cadastral parcel**: which can relate to one or

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

more farmers, based on ownership, and can cover one or more crops; **farmer's block**: which belong to a farmer but covers one or more crops without natural boundaries and **physical and topographical block** which Area bordered by certain features (ditches, hedges, walls, etc.) and can cover one or more crop groups (ECA, 2016). Belgium, Germany, and the United Kingdom have an LPIS for each region. All other EU Member States have one each covering the whole country. There are currently 44 LPISs in total, containing over 135 million reference parcels.

The LPISs are managed by the Member States, which are responsible for the quality of the data entered in their systems. The EU COM provides guidance to the Member States, audits the effectiveness, may apply financial corrections if there are failures in the LPIS. The LPIS ortho-images have a very high spatial resolution - mostly 25-50 cm per pixel - and are in general updated every three years. Due to the low frequency of updates to LPIS imagery, paying agencies could not use them to verify activities taking place on the parcel during the year (planting, harvesting, mowing, etc.). To verify farmers' declarations and adherence to eligibility rules, paying agencies have had to carry out field inspections for a sample of around 5 % of farmers. Field inspections are time-consuming and costly and provide a one-off record of the situation on the field. The Commission developed, since 1992, an alternative approach for inspecting agricultural parcels with satellite images from commercial providers (such as SPOT, WorldView, PlanetScope) taken at different times throughout the year, called 'checks with remote sensing'. According to JRC 80 % of field inspections are now performed using remote sensing. If the paying agency cannot draw a conclusion based on RS images an inspector carries out a 'field visit' to the land parcels concerned. They still require human intervention in the form of operators, who interpret very high- resolution (VHR) satellite images, using computer-assisted photointerpretation.

An upgrade was hence needed to meet the new requirements of the new CAP after 2014. A new set of LPIS-related rules was adopted, together with the greening requirements. For identifying all agricultural parcels used a famer, independently of the LPIS reference parcel system, the Member State authorities must provide all beneficiaries with a pre-established geospatial application form and the corresponding graphic material by 2018.

The 2013 CAP reform made it compulsory to use this Land Parcel Identification System (LPIS) together with a Geospatial Aid Application (GSAA) as components of the paying agencies integrated administrative and control systems (IACS), introduced progressively from 2015 to enhance checks of aid applications. In principle, farmers should since 2018 submitting their aid applications using geospatial methods, i.e., the position and size of their parcels must be derived from imagery captured in the LPIS. Some Member States implemented electronic claims with a geo-spatial component before GSAA became compulsory. LPIS including the information from the GSAA can be a valuable resource for analysing the structure and the land use of certain region at the parcel level and hence also a valuable source to parameterize farm level model. The next table summarizes sources freely available for download.

Table 2: FLIK and GSAA links for download, own compilation

Countries	Link	GSAA LPIS	Temporal coverage	Crop code
Luxemburg	https://data.public.lu/en/datasets/referentiel-des-parcelles-flik/	YES / YES	Several years	NO
Germany Lower Saxony	Sla.niedersachsen.de/landentwicklung/LEA	YES	recent	Yes
Germany - Brandenburg	https://geobroker.geobasis-bb.de/gbss.php?MODE=GetProductInformat	YES	recent	YES

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

	ion&PRODUCTID=996f8fd1-c662-4975-b680-3b611fcb5d1f			
Germany North Rhine Westphalia	available via DIAS	YES	2019	YES
The Netherlands	https://www.nationaalgeoregister.nl/geonetwork/srv/dut/catalog.search#/metadata/dd8e0fb8-0f09-40ba-a884-7e23c0680ae2	YES	Several years	YES
Austria	https://inspire.lfrz.gv.at/009501/ds/inspire_schlaege_20XX_polygon.gpkg.zip	YES	2019/2018	YES

Source: Own compilation.

2.2.3. Remote sensing products

{WR, G. Roerink}

In 1972 NASA launched the first satellite, which was specifically equipped for land observation. The first Landsat Multi Spectral Scanner (Landsat-MSS1) was able to map the earth surface with a resolution of 80 m in four spectral bands (Green, Red, Red/NIR, NIR). The satellite had a repeat cycle of 18 days, but due to data recording limitations not every image was acquired (in fact only 1600 images were acquired during its lifetime of 6 years). Crop monitoring at field level was not possible with this satellite, as with 80 m resolution only the bigger fields could be distinguished and only one or two images per year were acquired. However, the red and NIR spectral bands allowed it to quantify biomass. In practice this satellite was used for land cover classification at a coarser scale.

This example illustrated already very good that the power and usability of satellite sensors is characterised by the spatial, spectral, and temporal resolution. If you want to map house extensions, you need other specifications than when you want to monitor crop growth at field level.

2.2.3.1. Spatial resolution

The spatial pixel resolution of the land observation satellites nowadays ranges between 0.5 m and 250 m. There is a trade-off between the pixel resolution and the covered area. The highest resolution covers only small areas of a few square km's, while the coarser resolution satellites map areas of hundreds of km's wide. In case of crop monitoring, you want to map the farmer plots individually, so the pixel resolution must be substantially smaller than the average plot size. For a field size of 0.5 to 1 ha a pixel resolution of 10 to 20 m will do.

2.2.3.2. Spectral resolution

The electromagnetic spectrum ranges from shortwave gamma rays to long wave radio waves (see Figure 1). Typical land observation satellite sensors measure radiation in three domains:

- The optical sensors measure reflected solar radiation in the visible and NIR domain (0.4-2.5 µm wavelength). From crop monitoring purposes spectral bands in the red and NIR are required, with the principle that a green canopy needs red light for the photosynthesis process and NIR is reflected and transmitted almost completely, while for other land covers the red and NIR reflectance are more similar. Disadvantage is that the land surface only can be observed at clear-sky conditions.
- The thermal sensors measure emitted longwave radiation in the thermal infrared domain (8-12 µm wavelength). The land and sea surface temperature can be determined; besides that, forest fires are detected with thermal radiation. Also, thermal sensors have problems with clouds.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

- The radar sensors measure radiation in the microwave domain (1-10 cm wavelength). The large advantage of radar waves is that it passes through clouds, so image acquisition is guaranteed. The radar signal is sensitive for soil and plant structure and moisture.

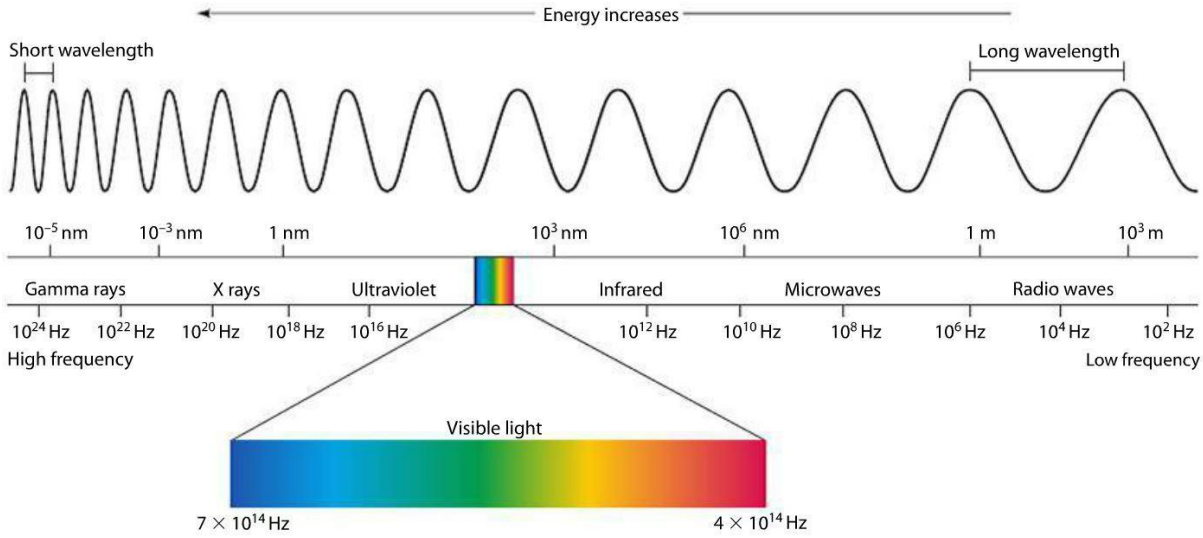


Figure 1: The Electro-magnetic spectrum.

Source: https://www.miniphysics.com/electromagnetic-spectrum_25.html.

2.2.3.3. Temporal resolution

The third dimension of satellite resolution is the temporal resolution. Satellite either have a polar orbit or an equatorial orbit. The latter one follows the earth equator with the same inclination speed as the earth orbit, so it stays at the same spot above the equator. As a result, its position in space is about 35000 km above the earth surface. So, pixel resolution is coarse (1 km), but it can make constantly image acquisitions. These satellites are typical weather monitoring satellites.

In case of a polar orbit the satellite makes orbit from pole to pole. Each orbit follows a new path over the earth. The revisit time is the number of days that is takes for the satellite to map again the same pathway over earth. The altitude in space is between 300-1000 km; much lower than for the geostationary satellite, so the pixel resolution can be much higher. So only polar orbiting satellites are suitable for crop monitoring purposes at field level.

For crop monitoring purposes resolutions between 10 and 20 m are required. Satellites with these pixel resolutions have typical revisit times between 10 and 20 days, which is not enough for adequate crop monitoring, especially if one considers the cloud occurrence in many parts of the world. The solution for this problem is to launch more identical satellites (for example Sentinal-2A and Sentinal-2B) which multiplies the number of acquisitions and assures lower revisit times.

2.2.3.4. Copernicus program

Although the Common Agricultural Policy has a long history of using satellite or aerial images for checking area-based aid, these images mainly before 2017 usually were not available with sufficient temporal frequency. New imaging technologies with the EU-owned Copernicus Sentinel satellites, known as Sentinels, had become a new source of data for monitoring the Common Agricultural Policy because automated processing of time series throughout the growing season makes it possible to identify crops and monitor certain agricultural practices on individual parcels or even at higher resolution (10x10 meters) Eu-wide (Devos, 2017,2018a,2018b). Although the Commission promoted the technology through many conferences and workshops in 2019 only 15 out of 66 paying agencies used the Copernicus Sentinel to check aid applications and an audit of the Commission revealed that

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

many paying agencies consider that there are obstacles to wider use of the new technologies (ECA, 2020).

2.2.4. Green Monitor

{WR, Roerink}

The Green Monitor data platform (www.groenmonitor.nl) started in 2012 to map the Netherlands with high resolution satellite imagery. The Green Monitor is developed as an easy-to-use webtool for visualization and interpretation of time series of NDVI satellite images covering the Netherlands.

Figure 2: Screenshot of the Green Monitor (www.groenmonitor.nl). The graph shows the annual NDVI behaviour in 2019 of the blue marked grassland parcels (note that 5 mowing events can be distinguished).

Source: www.groenmonitor.nl.

From 2016 onwards the Green Monitor is collecting and processing high resolution imagery from Sentinel-2 and Landsat satellites into uniform Normalized Difference Vegetation Index (NDVI) images of 10 m resolution over the Netherlands. The NDVI is a measure of the amount of green biomass and is defined as a ratio of the NIR and Red spectral bands with values between 0 (water, bare soil) and 1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

(crops with multiple layers of green leaves). The atmospherically corrected level 2A product is used and pixels are resampled in a uniform grid (the Dutch Rijksdriehoekstelsel projection). The major effort in pre-processing the data is the cloud and shadow screening and masking procedure. For a reliable crop monitoring system, the cloud and shadow detection is crucial. In the beginning the cloud masking was done manually; since 2019 it is performed by AI technology which recognises and removes the black and white cloud and shadow patters. In case of hazy cloud patterns, the clouds and shadows are checked also manually and wherever necessary additional corrections are made.

Table 3: Available satellite imagery for the Green Monitor

Satellite	Period	Spectral bands	Resolution	Revisit time
Sentinel-2	2016 onwards	B, G, R, NIR	10 m	5 days (overlap 2x per 5 days)
		Red Edge, NIR, SWIR, MIR	20 m	
		Deep Blue, Cirrus, Water vapour	60 m	
Landsat-8	2013 onwards	Panchromatic	15 m	16 days (overlap 2x per 16 days)
		Deep Blue, B, G, R, NIR, SWIR, MIR, Cirrus	30 m	
		TIR-1, TIR-2	100 m	
Landsat-7	1999 onwards	Panchromatic	15 m	16 days (overlap 2x per 16 days)
		B, G, R, NIR, SWIR, MIR	30 m	
		TIR	60 m	

Source: Own compilation.

2.2.4.1. Grassland markers

The cloud free satellite images in the Green Monitor provide the opportunity to monitor each parcel in the Netherlands. For each parcel the temporal NDVI profile throughout the year is extracted. The NDVI curve reveals quantitative information on the growing season of the crop. A distinction is made between arable crops and permanent grassland. In case of grassland the derived markers are:

- Number of mowing cuts
- Date of first, second and later mowing events
- Grass ploughing and renewal
- Yield indicator

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

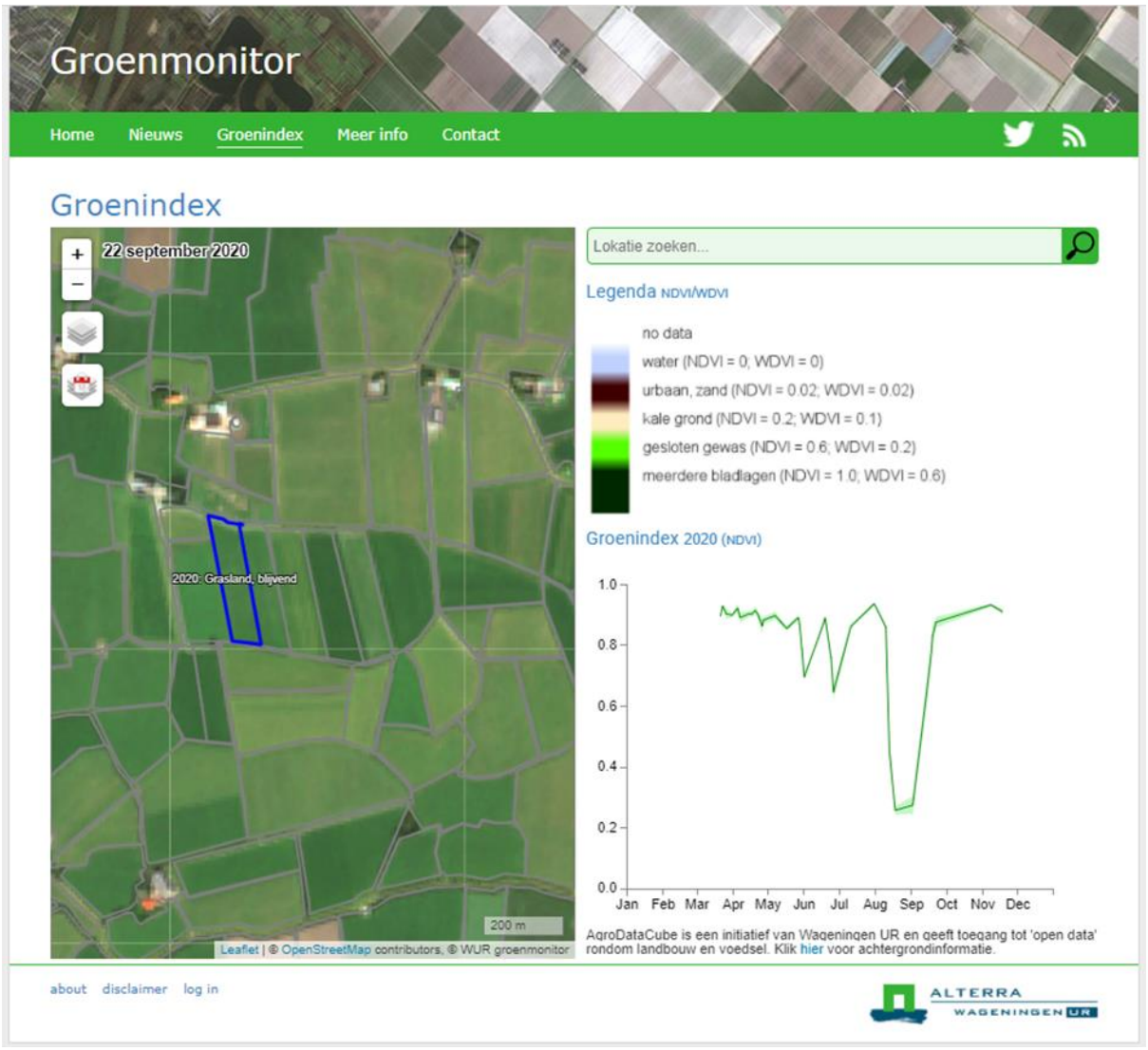


Figure 3: Screenshot of the Green Monitor (www.groenmonitor.nl). The graph shows the annual NDVI behaviour in 2020 of the blue marked grassland parcel with two mowing cuts and a grass ploughing and renewal event

Source: www.groenmonitor.nl.

Figure 3 explains how the grassland marker detection algorithm works. In principle permanent grassland has high NDVI values throughout the year. Small dips in the time series are detected as mowing cuts. The dips are relatively small as not all grass is removed; the straws remain on the field and regrowth starts again. Grassland ploughing and renewal is detected as a large dip in the curve, as the ploughing reveals the bare soil with very low NDVI values. The cumulative NDVI value is used as yield indicator. Through an empirical relationship the NDVI values can be 'translated' to grass length (in cm). The grassland markers are stored and made available again in the AgroDataCube.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

2.2.5. AgroDataCube

{WR, S. Janssen}

Many valuable open data sources are available for the Netherlands that can improve data science and decision making in agriculture and food. However, these data sources are still scattered and are published using a range of different, standardized, and non-standardized formats and protocols. This means that substantial efforts are required to find, collect, and combine such data repeatedly, to feed the many applications that use such data. The AgroDataCube functions as a hub that brings together these heterogeneous data streams, enriches them, adding in-house analytics, and publishes the result as harmonized, up-to-date, standardized datasets accessible through an open REST API (agrodatacube.wur.nl).

In 2018, version 2 of the AgroDataCube has been developed. Through integration with Green Monitor, the AgroDataCube now also provides a remote sensing-based vegetation index (NDVI) at sub-parcel resolution. Such vegetation indices are used for research, e.g., crop modelling and yield forecasting, by farmers to monitor the development of their crops, or to monitor agricultural practice, e.g., complying with CAP regulations.

The approach: Merge, harmonize and publish

Many distributed data services relevant for the agri-food domain already feed into the AgroDataCube. These sources are heterogeneous about different aspects. While for instance remote sensing data or weather data are voluminous, available daily and are processed near-real time, soil data and parcel data are smaller and relatively static. The AgroDataCube automatically structures and harmonizes the incoming data streams and links their spatial and temporal dimensions. This means that for example time-series of weather data or NDVI (Normalized Difference Vegetation Index) data can be retrieved on the level of agricultural parcels. Data is delivered in a standardized format and therefore easily reusable, for instance in data analytics tools and decision support systems.

AgroDataCube currently provide data services that publish spatially and temporally explicit data from the following resources:

- Agricultural parcels and parcel attributes (parcel geometries and crop information from BRP, AAN)
- Soil data (Soil map 1:50.000, BOFEK)
- Weather data (observations from KNMI stations)
- Elevation (AHN)
- Administrative regions (NUTS and postal codes)
- Green Monitor satellite data
    - NDVI, WDVI vegetation indices (mean and standard deviation)
    - Grassland markers: mowing dates, ploughing date, management intensity
    - Arable land markers: ploughing date, sowing date, emergence, harvest, catch crop
- Radar coherence (Sentinel-1)

The ADC is filled near real-time with current data (weather data, green monitor satellite data), so that the current situation in the field is always available and there is a perspective for action.

The AgroDataCube is an innovation in big open data. It is one of the first real results of combining data from different batches and make them unequivocally available to the user. It is based on open data principles and is open documented.

The AgroDataCube:

- makes an innovative contribution to the aspect of interoperability of data in the agri-food domain

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

- is of great importance for different interest groups
- makes new research and new business and consumer-oriented solutions

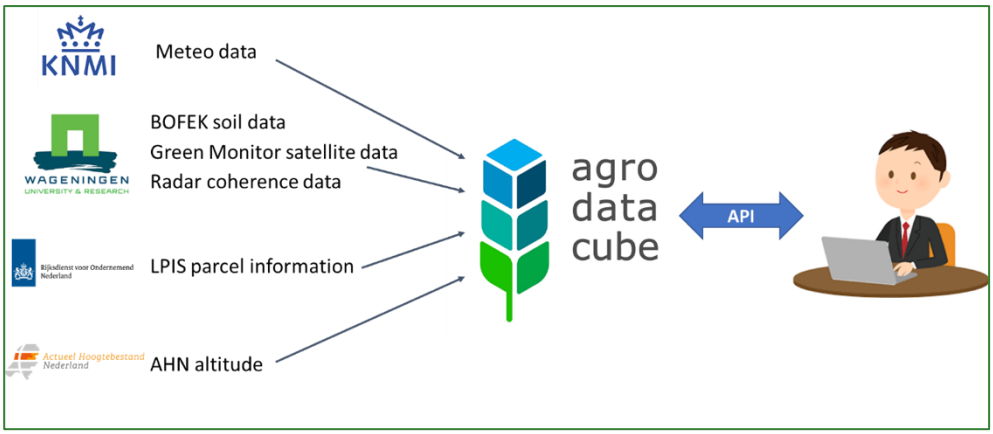


Figure 4: Schematic representation of the AgroDataCube

Source: Own compilation.

2.3. Review interfaces for IDM modelling to existing and established modelling databases

2.3.1. GLOBIOM database and interfaces

{IIASA, A. Brouwer}

The Global Biosphere Management Model (GLOBIOM) has been developed and used by the International Institute for Applied Systems Analysis (IIASA) since the late 2000s. It is a partial-equilibrium model that represents various land use-based activities, including the agriculture, forestry, and bioenergy sectors. The model is built following a bottom-up setting based on detailed grid-cell information, providing the biophysical and technical cost information. This detailed structure allows a rich set of environmental parameters to be accounted for. Its spatial equilibrium modelling approach represents bilateral trade based on cost competitiveness.

GLOBIOM takes land use inputs for the agriculture, livestock, forestry, and bioenergy sectors from FAOSTAT and the Spatial Production Allocation Model (SPAM). SPAM data is processed with aid of the mapspam2globiom R package[18]. In the case of the EU, crops are allocated across NUTS2 regions using data from EUROSTAT. Harvested areas are based on FAOSTAT statistics but spatially allocated using data from the SPAM. Yields for all locations and crops are determined in a geographically explicit framework by the Environmental Policy Integrated Climate Model (EPIC). The yields are distinguished by crop management system and land characteristics by spatial unit.

Total forest area in GLOBIOM is calibrated according to FAO Global Forest Resources Assessments (FRA). The available woody biomass resources are provided by the forest model G4M for each forest area unit and are presented by mean annual increments that are divided into commercial roundwood, non-commercial roundwood and harvest losses, thereby covering the main sources of woody biomass supply.

[18] https://iiasa.github.io/mapspam2globiom/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

The model is primarily implemented in GAMS. Prior to running GLOBIOM, EPIC and G4M are run, and a pre-compilation stage converts raw input data into GAMS-ready data that serves as input for the model and scenario stages. This includes GDX, GMS data-definition, and CSV files.

On conclusion of the scenario runs, output data for the various scenarios is collected and merged. Through stages of filtering and aggregation, the output data is distributed across a series of GAMS parameters that are the representational near equivalent of Python/R data frames as well as a series of supportive set definitions. Parameters and sets are all included into a single output GDX. Data dimensions include a spatial specifier for the various grid cells countries, or regions; indicator; unit; item; scenario indices; and time. Interfaces primarily access these output parameters and sets.

To interface GLOBIOM to R, the gdxrrw R package is used[19]. It allows GDX content to be explored, read, and written. Read GDX data is represented as R data frames which are readily converted into the modernized data frame representation called tibble that is at the core of the tidyverse R framework.[20] The tidyverse is a coherent and lucidly designed set of R packages that cover various aspects of data analysis such as tidying, transformation, manipulation, and declarative visualization. On top of gdxrrw and the tidyverse, a GLOBIOM visualization interface is provided by the globiomvis R package.[21] It supports analysis and generation of a variety of scenario plots. In addition, globiomvis enables creation of maps for the various regionally and spatially explicit representations of the model.

For interactive exploration—and intended primarily for training and outreach purposes—a graphical user interface (GIO) based on GGIG (Britz, 2014)[22] is available as an alternative way of performing analysis, visualization, and parameterizing and running the core version of the model.[23] GGIG is Java-based and orchestrates numerous Java libraries that provide rich visualization and analytical functionality. GGIG is specialized to GLOBIOM purposes via a hierarchy of XML configuration files. To compensate for their fragility, these XML configuration files are validated against and further documented (beyond the reach of the regular GGIG documentation) through annotated XSD schema files. To ease maintenance and updates of the GUI, the XML files are to a large degree generated from the GLOBIOM code base as well as from the GGIG state persistence INI file by means of Python scripts.

2.3.2. CAPRI database and interfaces

{THÜNEN, A. Gocht, M. Himics}

The CAPRI agricultural economic model has been developed since 1999 with the help of several EU research projects. The model supports the policy-making process by means of quantitative analyses of the EU Common Agricultural Policy (CAP) at global and regional levels. The aim is to estimate in advance the impact of agricultural policy decisions on production, income, trade, and the environment using the model. To guarantee comparability of results between member countries and over time, CAPRI uses standardized and harmonized data sources from EUROSTAT, the EU Commission and the FAO or OECD, as far as possible. A large part of the development is devoted to data preparation or standardization to reference units that are thus comparable over time and across regions in Europe. The database consolidation steps are designed in such a way that all data changes can be repeated so that newly available data sets or improved statistics can be integrated without problems. Such a concept allows the data consolidation to evolve continuously without having to accept methodological breaks. This means a high level of hardware requirements and methodological competence. For example, the creation of the database requires the computing power of a high-performance computer

[19] https://support.gams.com/gdxrrw:interfacing_gams_and_r
[20] https://www.tidyverse.org/
[21] https://iiasa.github.io/globiomvis
[22] https://www.ilr.uni-bonn.de/em/rsrch/ggig/GGIG_user_Guide.pdf & https://www.ilr.uni-bonn.de/em/rsrch/ggig/GGIG_programming_guide.pdf
[23] https://github.com/iiasa/GLOBIOM_GUI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

for about one day, although many of the processes run in parallel. The result of the database consolidation steps are time series for the agricultural sector at the regional Nuts2 level for the topics of agricultural accounts (EAA), land use, animal stocking density, factor income, prices, market balance, nutrient requirements, and nutrient suppliers. In addition, the dataset contains a consistent representation of regional feed requirements and feed resources. Special features are, apart from the balancing consistency framework, the bundling across regions (Nuts2, Nuts1, MS, EU) and the detailed environmental indicators in greenhouse gas emissions and nutrient balancing. An additional gain is the accompanying archiving of European datasets, as it is not uncommon for statistics to become unavailable after a few years. As experience shows, the dataset has a high usefulness for many analyses and projects, especially in the European context, although better-resolved data sources are sometimes available at regional or national level. The generation of consistent time series is part of the data work. The time series are also used to derive trends for future development and to create reference points for the future. These future projections, often referred to as baselines, use official agricultural projections from the EU Commission in addition to trend projections from the time series. The rule here is that the regional trends in the sum of all regions should reflect the projection of the EU Commission. These projections have a projection period of about 10 years and serve as a reference data set in many European analyses in the agricultural and environmental sector. In recent years, long-term projections with reference to climate development have also been produced. Here, external projections to the SSPs from the GLOBIOM and PRIMES models are used. Data consolidation uses statistical or mathematical estimation methods with the goal of adjusting the values of the statistics only when economic and bio-physical relationships or other statistics require it. If, for example, the yield multiplied by the area does not correspond to the production volume from the statistics, the yield is adjusted accordingly. The original data and the new estimated data are stored together for comparisons and better tracking. A metadata model allows the necessary information on the statistics and on the processing steps to be summarized and stored efficiently. Traceability in case of changes in the routines is ensured by the version control software SVN. The network is supported by eight institutions, including 3 universities, one company and four research institutes. In addition, the EU Commission-JRC supports the network with positions and corresponding calls for proposals. For each data consolidation step in the model there are responsible developers. Many institutions support the network with personnel as well as with hardware and software infrastructures. Currently, all data is freely available and are downloadable.[24] The ***interfaces*** to access the databases are currently based on a graphical user interface GGIG (Britz, 2014).[25] The GUI comprises a generic and powerful tool for exploitation, accessing and exporting of results. Results from the different work steps of CAPRI generated by GAMS are stored in GDX format as multi-dimensional sparse data cubes. The regional time series data base of CAPRI covers almost 15 Mio non-zeros values. To access these huge data quantities in a user-friendly and efficient way, an XML file defines views in the data. Each view is firstly characterised by a selection or filter for the different dimensions such as regions, activities, items, or scenarios. Secondly, a pivot is defined which maps the data base dimension to viewport dimensions, such as the columns or rows of a table, or the regions shown in a map. And thirdly, it defines the view type: a table, different type of graphs or a map. Fourthly, views may comprise links to other views, like the concept of hyperlinks in WEB pages, which allows a "drill-down" like exploitation from general to specific aspects, or vice versa.

But the user maintains his freedom: he may tune the view to his own needs, by adding his own selections, change the pivot or the view type. Equally, fonts, colour, cell sizes or properties of the graphs may be set by the user. His personal settings can be stored for future session. And finally, the mapping viewer allows for rather flexible classifications and colouring options. The details with

[24] https://www.capri-model.org/dokuwiki_help/doku.php?id=getting_started_with_capri
[25] https://www.ilr.uni-bonn.de/em/rsrch/ggig/GGIG_user_Guide.pdf & https://www.ilr.uni-bonn.de/em/rsrch/ggig/GGIG_programming_guide.pdf

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

examples are discussed in a chapter of the CAPRI documentation.[26] In addition to the GGIG there exists an ***interface*** to import the result database of CAPRI into Excel using an COM add for Excel based on the GDX API of GAMS and the MS .net framework. After opening Excel, the COM add-is loaded and a new Ribbon "CAPRI REPORTS" can be used to import formatted result table with long texts into Excel, ready for reporting.[27] Similarly, a R package interface to access the CAPRI database exists developed by Mihaly Himics. The R package reads GDX files and prepares figures and charts. [28] It includes functions for processing, visualizing, and analysing both the model databases and simulation results. It has been designed to complement (rather than to replace) already existing GIGG for CAPRI. The advantages of capriR compared to graphical User Interface options include dissemination of model-databases and simulation results, automated reporting requiring additional (post-model) calculations, creating publication-quality maps and other data visualizations. As CAPRI covers EU agricultural production activities with fine geographical detail, spatial data analysis and visualization are of a particular importance for capriR. CAPRI results are linked to commonly used spatial data packages thus enabling the user to create high-resolution static or interactive maps. With capriR you can also access rapidly the databases and simulation results of the CAPRI modelling system. The modularity of the R programming language allows for directly applying advanced econometric and statistical techniques from other (open-source) R packages on the data sets retrieved from CAPRI. Although capriR is only directly useful for the relatively small user base of CAPRI, some of the general strategies for rapid package development might be re-used for similar, large scale economic models.

2.3.3. MAGNET

{WR, J. Helming, M. Müller}

2.3.3.1. Model and background

The policy landscape is becoming increasingly complex with interrelated global challenges stretching across domains previously handled in relative isolation. An example is the Paris agreement of 2016 (UNFCCC 2016) which will have widespread repercussions for the way in which the world economy operates. Such commitments require policymakers to look at impacts beyond their own domain and decades ahead. With feedback loops abound impacts of interventions become theoretically ambiguous requiring ex-ante integrated modelling tools to explore expected impacts of policy interventions, trade-offs, and synergies across multiple domains. Such challenges are faced by MAGNET (Modular Applied GeNeral Equilibrium Tool). The MAGNET model is a global general equilibrium model. MAGNET is based on the LEITAP model which has been used extensively in policy analyses. MAGNET offers more flexibility in model aggregation (definition of regions and sectors) and more options for changing a model's structure. The main aim of MAGNET is to offer a global applied general equilibrium modelling framework which can be easily tailored to specific research questions and regions and products of interest. This flexibility allows researchers to adjust the complexity of a model to the questions at hand as well as to their own level of understanding of global CGE models. The core of the MAGNET database is the GTAP database. MAGNET uses a series of additional databases, such as GTAP satellite databases, FAOSTAT (commodity balances, land use, land cover and fertilizer), data on biofuels from the International Energy Agency and land use parameters taken from the IMAGE model. On average for almost each module in MAGNET (Figure 5) additional data is included in the MAGNET-database, complemented with scenario data like projections of GDP and population. The database is constructed in several steps: the first step includes gathering raw data from the web or other sources and reshape to standard input formats by hand or using code. The next

[26] https://www.capri-model.org/docs/capri_documentation.pdf#search="Exploitation"
[27] https://www.capri-model.org/dokuwiki/doku.php?id=capri:team:alex_gocht
[28] https://github.com/trialsolution/caprir

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

step is performed by using DSS (Figure 5). You can select the desired GTAP database, add other databases and formulate new regions, sectors, commodities, and endowments. These steps lead to the desired database for a specific project. The processing of data is for almost 100% coded to make it easier to update the original source data, to track how data are processed and to maintain flexibility using different GTAP database versions.

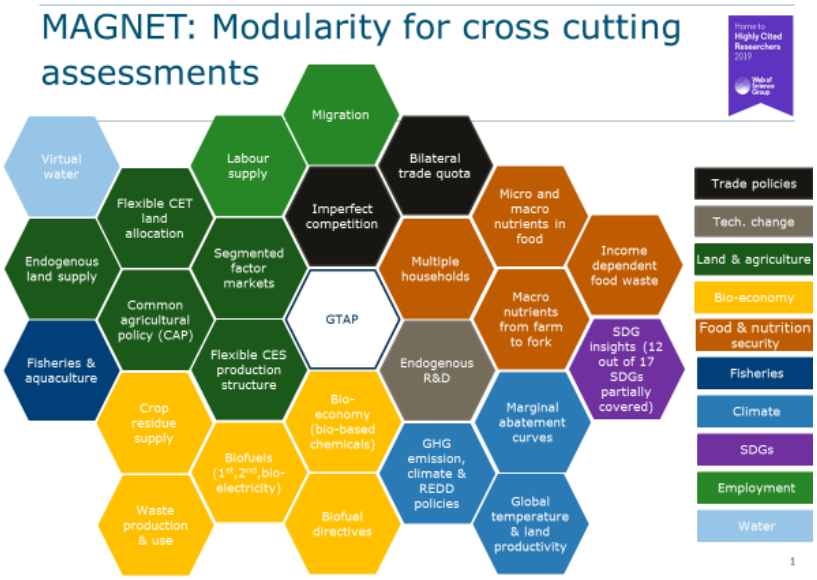


Figure 5: MAGNET modules

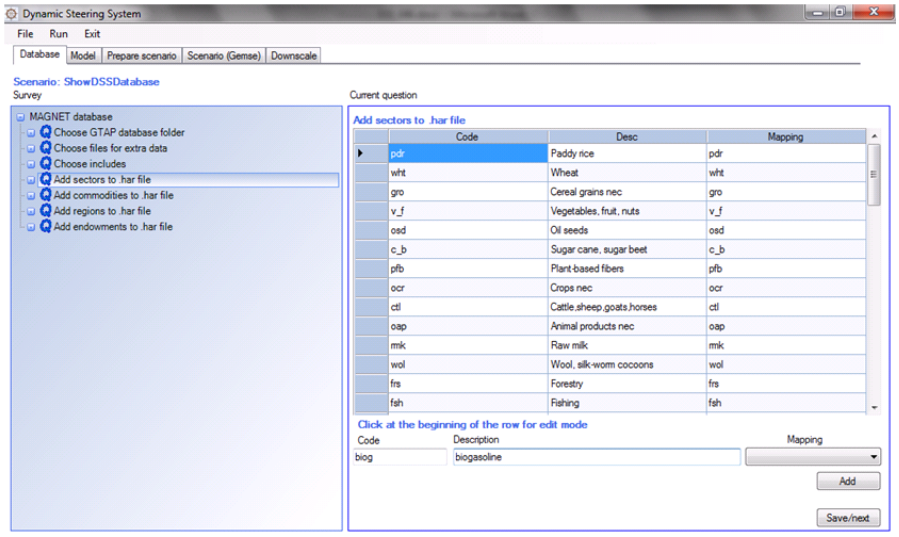


Figure 6: DSS – Example of selections for creating a database

However, the first step is not coded as the raw data needs to be checked on consistency and naming.

2.3.3.2. Handling multiple users and developers

Magnet has been developed by multiple persons at different locations and furthermore results are used by multiple internal and external users or clients. This requires a solid infra structure to guarantee consistent and reproducible outcomes. As MAGNET is used in a range of projects, we need to assure that there is a working version of the model that can be used as-is. Therefore, development of the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

MAGNET model and database take place under version control system SVN working with a trunk and branches. The trunk forms the stable core of MAGNET. It has no loose ends or broken parts. All its components are documented and tested with the production version of MAGNET (a specific aggregation and set of simulations). It can therefore be used "as-is" for applied work. If the trunk changes it is a single revision that is a state-change: it moves from a working state to a new working state where all included components and answer files work. There is thus not a series of revisions introducing loose ends/ unfinished code or which break already included parts of the system. Developments of the code system is done in branches that originate from the trunk. Here intermediate versions can be committed that not yet work and all other files that aid the development of the code. For example, manually created database files used for developments can be stored and shared with other team members. WEcR developed a dedicated software package called DSS (Dynamic Steering System) to ease the use of MAGNET. In essence, DSS 'asks questions' on which files to use and then creates appropriate command files for running GEMPACK code (referred to as Scenarios in DSS). Specifically, DSS is used to create the MAGNET database (Figure 6), to create the MAGNET model and to construct scenarios. This allows the MAGNET-users to use the same code for different databases (different releases of the GTAP database, or different aggregations for specific applications) and different model structures. This flexibility comes at the cost of requiring the user to make a large number of choices before being able to run the model. Answers of a model setup are stored in so called answer files. These answer files can be shared with other users to reproduce the specific model versions. Working with DSS facilitates also a full separation of model code and data files. A new concept has been developed for MAGNET output: writing MAGNET scenario results to the central Datawarehouse of WecR (Figure 7) complemented with links to predefined PowerBI reports. The Datawarehouse of Wageningen Economic Research starts with including data into the system using SQL Server Integration services (SSIS) (Figure 7). The data is converted and processed using Data Quality Services (DQS) and Meta Data Services (MDS) and finally stored into the Data warehouse (DW). In the semantic layer derived data is created. Finally, the data is accessible (authoring) using different applications like SQL, Power-BI, R, Python and other. One of the applications is Power BI with additionally PBI-report server. With this latest tool visualisations of project data can be made available for users/clients outside Wageningen Economic Research. However, these reports contain predefined figures and tables. Therefore, yet not in Figure 7. However already in development, is OData webservices is introduced with which users from outside Wageningen Economic Research can also query the data. Users need to be authorized for this.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

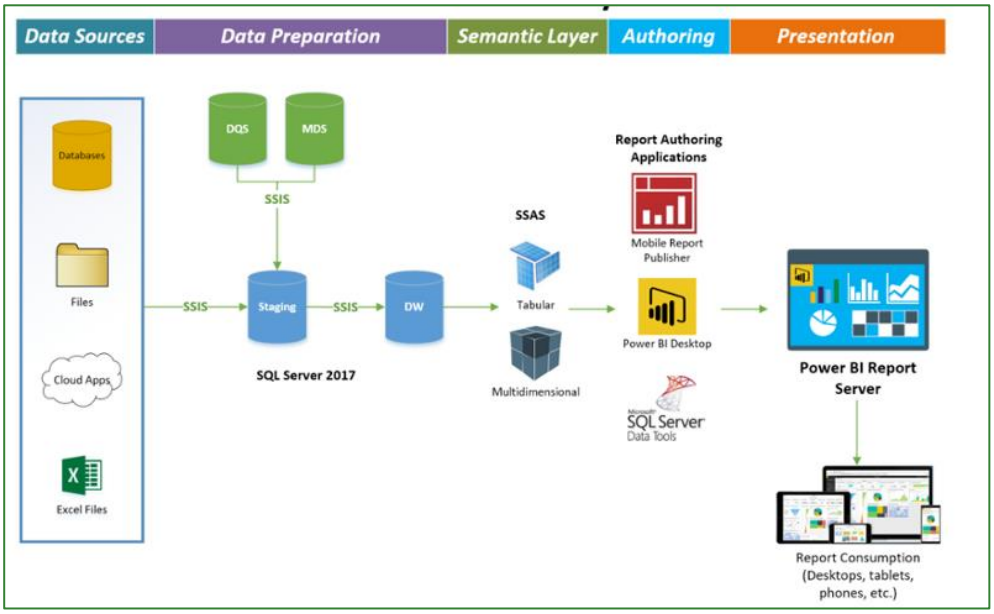


Figure 7:Overview of the Datawarehouse system of Wageningen Economic Research

Scenarios of MAGNET can be stored in the Datawarehouse by added functionalities in DSS. In DSS a tab is added where you can select from which scenario the data should be added to DW. After including the data, MAGNET developers can create PBI reports to release these data and clients and internal users can use these reports for own use. An example of such a report is presented in Figure 8.

Figure 8: Example of a Power-BI dashboard showing MAGNET results for a project

Magnet-GRID is an additional program in which had been developed by WecR for simulating the spatial patterns of agricultural land use resulting from economic decisions on the use of land (Vasco et al, 2020). MAGNET-GRID combines scenario-based projections derived from MAGNET output with spatially explicit projections on the biophysical suitability for agricultural projections and shows results on 1-km by 10 km for primary production like wheat, cattle farming and other.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

3. CONCEPTUAL AND TECHNICAL FRAMEWORK INTERFACE DEVELOPMENT

{lead THÜNEN}

API is the acronym for **Application Programming Interface**, which is a software intermediary that allows two applications to talk to each other. GUI, or UI, stands for **Graphical User Interface**, a software platform that presents the back-end data in a visually coherent way to users. Modelling systems reviewed in Chapter 1 using often the GGIG graphical user interface, which can be dynamically adjusted by the user, by a text-based xml definition file, and hence does not need code chances, when new controls and views are required. Many economic models offer visual aid and easy data access possibilities for their users via graphical user interfaces (GUI). Most GUIs still require the user to do numerous and time-consuming interactions with the software, slowing down the analysis of simulation results, and sometimes even hindering model users to find the relevant drivers and other causality chains in model results. Software or its elements don't need a graphical user interface to communicate with each other. Software products exchange data and functionalities via machine-readable interfaces – APIs. R packages have been reviewed in Chapter 1, which are one way that the user interacts in a programming environment, like R, using a standardized interface, provided by a certain R-package, like capriR or globiomvis. A particular API which operates via the internet to communicate between application is the REST API (also known as RESTful API) It conforms to the constraints of **REST architectural** style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding. **REST** is a set of architectural constraints, not a protocol or a standard. For an API to be considered RESTful, it must conform to certain criteria. In Chapter 1 we have seen that the AgroDataCube, MAGNET with the **OData webservices,** but also the service to obtain sentinel data using the API Rest technologies. They are interesting when the data sources are huge, and the user is interested parts of the provided data. Such services also require a good security concept when the data is not public domain, which also restricts the use of Rest APIs for highly sensitive data like FADN.

In MIND STEP particular in WP2 an interface for data used for modelling requires to have certain properties:

- Given the complexity of task in modelling to combine and merge different data sources, check for consistency, and understand the process of data generation **a graphical user interface is too restrictive** and will not offer the flexibility required for the different data modifications for the different models.
- **An API in a classical form programmed in .net, C++ or java requires IT knowledge** not existing amongst modellers in the field of agricultural economics.
- The **programming language** need to be more **open and less restrictive, well-known,** maybe also taught already at the university, **free of charge**, **running cross platform** and easily applicable by learning from internet sources. Example in that field are python and R, whereas R also provides the functionality to include Python. Particular the R programming language provides complementary data exploitation possibilities with its command line interface (CUI) and with many optional packages for analysing and visualizing large datasets. R can be executed repeatedly with a minimum effort for interaction with the software interface, each time simulation results have been updated. Data visualizations, statistical and econometric analyses based on of third-party R packages can be easily replicated and repeated in this manner.
- The language shall in the optimal case also allow to include GAMS, which was identified in WP7 as the main language for agricultural IDM modelling.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

- We also need to **account for different groups** involved in development, maintenance, and application of the interfaces.
    - There are **developers of the interface**, and which need to have a shared distribution system to commonly develop and extent the interface. the system shall allow that developers can track changes for their local environment but also push the developments to a central repository.
    - Another group is the **user group**, who will use the API in a more applicable way by loading the API and applying without any need of changing the function of the interface itself. For this it also requires a ***good documentation*** of the interface and the offered functionality that the user knows how to apply the interface. In an optimal scenario such a documentation of the interface shall be to a certain extent generated in an automated way and with a similar structure for all different interfaces made available.
- Besides the documentation of the interface also a ***use case documentation*** is of importance. It should show how the functions can be applied by providing examples. As example for FADN how to apply the API to merge or convert, load, or select and filter data sets. As most of the data is, however, not commonly available, interface shall provide a **dummy data set** to allow test without having the data at hand.
- Even if this is to a certain extent clear but the user of the interface shall clearly understand under which software setup the interface was developed.
- In addition, the interface shall make clear for which **version of the data base** it was tested and developed and further **links shall be provided** to understand the structure and the meaning of the data items.
- It shall also include meta-data, version info, Affiliation and how to refer in scientific publication to gibe appropriate credentials.
- It shall support Unit testing. This also requires a high modularization also import for maintenance.

3.1. Proposed workflow for the interface development

{THÜNEN, A. Gocht, X. Yang, S. Neuenfeldt}

Given the identified requirements we identified a workflow (concept) for the mind step project that might fit and satisfy most of the requirements listed above. Given that most of the modelling groups are working on Windows operating systems we outline the workflow for Windows, but it is equally applicable for Mac OS and Linux.

The core is R and R-Studio, the latter is an IDE for R which provides some nice features for interacting with the versioning system, viewing data and create automated documentations. The versing system we propose is a Git approach. The server is installed and currently operating under the THÜNEN domain as server as GitLab server. Available for all mind step partners. This allows to deploy the solution and record the history. GitLab interact with a local installation of Git. This allows for track changing local modification, a superior property of Git compared to SVN, where all commits target the server repository directly. The installation or setup of R is based on anaconda, which (in an optimal case) allows the package management for R (and much more) and installation without administration rights and provides a GUI for loading different environments. It also allows to copy and import environment from different sources.

The key to the proposal is that we build up on **R packages**. After installing r and R-Studio the following steps are proposed to develop the interfaces for mindstep:

1. Setup a new R package project for your interface

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

2. Link this to an empty GitLab project
3. Write some *super nice* interface functions for your data (can be remote (suing other RESTful API) or local)
4. Distribute the interface
    a. Push to the GitLab (official deliverable)
    b. Providing a documentation (will go to the annex of this Deliverable)
5. Using R and install the library for your modelling project, e.g., filter and modify local FADN data and export as GDX for GAMS.
6. Develop a user case document, will be part of Chapter 4 by each interface, using this approach.

What makes the approach possibly interesting for a wider economic modelling public is that R packages can be built with limited efforts, and under limited time, and is well known by students and hence potential new modellers. It is distributed under the GNU license. Such R packages can be developed for interfacing complex databases, but as well as for dissemination, visualization, and complex data analysis. Also, model linkages can benefit a lot from model-specific rapid package development. The need for exchanging large amounts of data between different model architectures poses a practical challenge to modelling groups. Model-specific R packages for data exchange offer a common software platform. The large user-base of the R programming language in the broader scientific community makes such packages efficient in disseminating models and interfaces for a general scientific audience, increasing at the same time the transparency of modelling exercises.

3.2. Technical Implementation

We descript the technical implementation of step 1- step 6 and the software required for the proposal in the Annex 9.1-9.5. We first present the setup of the software. Then how to link R and GIT for a package and afterwards explain, from the view of a developer, how to setup the GITLAB repository, create a R package project and interlink it. Then we explain how the developer, with the intention to adapt or add to the package functions, can work with the package, and deploy it. In Annex 9.4. we present how to build a corresponding manual. At the end we show how “use cases” and their documentation can be built.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

4. DESCRIPTION OF THE DEVELOPED PROTOTYPE INTERAFCE VIA USE CASES

4.1. FADNutils: Data interface to EU FADN data -Use Case 1

{THÜNEN, Yang; JRC, Kremmydas}

The `fadnUtils` package facilitates the efficient handling of FADN data within the R language framework. This means that there is a specific temporal pattern of how a user interacts with the package (see Figure 9). More specifically, after a request for FADN data from DG-AGRI, this data is delivered in csv format. The first step is to import the data into an R-friendly format using the `fadnUtils` package. A detailed explanation of this step is given in Use Case 4.1.2.2.

After importing the data, the user can proceed on analysing data based on his individual needs/targets. Use cases 4.1.2.3 and 4.1.2.4 provide details on how the package facilitates further data analysis.

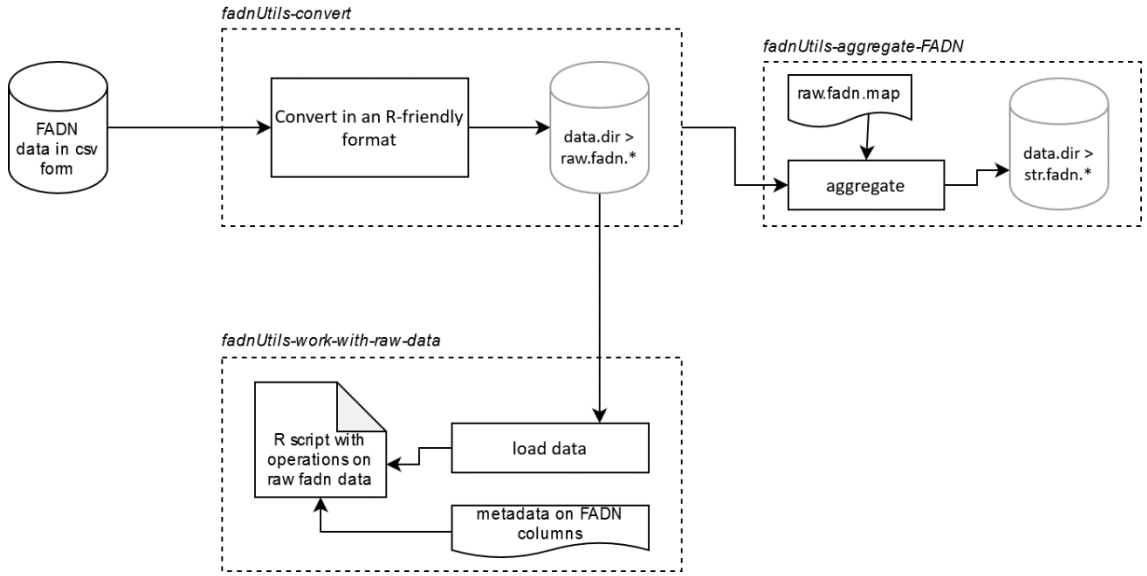


Figure 9: Temporal overview of how the user interacts with the package.

Source: Own compilation.

Importing the FADN csv files is done a single time and the imported r-friendly data is stored permanently in a data directory. Then, at any time, different researchers can use this data directory to spin off different types of analysis.

Finally, `fadnUtils` extensively uses the `data.table` R package and thus the user is expected to have a basic knowledge of it.[29]

4.1.1. Installation

You can install the development version from GitLab with:

```
# install fadnUtils package
devtools::install_git("https://git-dmz.THUENEN.de/MIND STEP/fadnutilspackages")
```

29 For more on data.table package, see documentation in https://cran.r-project.org/web/packages/data.table/index.html and a swift introduction in https://www.datacamp.com/courses/data-analysis-the-data-table-way

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
# install related R packages
requiredPackages = c('fadnUtils','data.table', 'devtools','jsonlite', 'ggplot2')
for(p in requiredPackages){
  if(!require(p,character.only = TRUE)) install.packages(p)
  library(p,character.only = TRUE)
}
```

4.1.2. Use cases

4.1.2.1. Create a working directory

First, the user sets a working directory. Make sure the relative path stays within CurrentProjectDirectory. A working directory is specified arbitrarily by the user. This structure is helpful for data management and maintenance.

```
# ..................Create data dir.....................................................#

# fadnUtils always work with a user defined data.dir
# Let's assume that the user has not created one yet.
# The following line creates a data.dir folder somewhere in our computer
# We must also have created the raw_str_map.file and pass it as an argument
#   to the function. This file is copied to the data.dir folder. Thus, we can
#   see the structure of the data contained in a data.dir folder by inspecting
#   the raw_str_map.file residing in it.

# a local directory
CurrentProjectDirectory = "D:/public/yang/MIND_STEP/New_test_fadnUtils"

# the path of the fadn files for loading
fadn.data.dir = "D:/public/data/fadn/lieferung_20210414/csv/"

# ceate a data.dir
create.data.dir(folder.path = CurrentProjectDirectory)

# Once the data.dir is created, we must declare that we are working with it
set.data.dir(CurrentProjectDirectory)

get.data.dir()
# After you create a data dir, below is a list of "real-world" example files:
# CurrentProjectDirectory/
# +-- csv
# +-- fadnUtils.metadata.json
# +-- rds
# \-- spool
#     \-- readme.txt
```

4.1.2.2. Import csv FADN data

After the successful request of FADN data from DG-AGRI, data was delivered in csv format. Usually this is done through one csv file for each country and each year; thus, we expect the raw data provided to reside on many different files. There is a possibility for variation of the csv format. Thus, the package allows the user to define the configuration of the csv for a specific request (delimiter, number of columns, symbol of decimal point, etc.). However, we expect that the csv format will be consistent for all the different files sent as a response to a single request.

The first step for using the raw FADN data is to import the data into a data.dir folder. A data.dir is an ordinary computer folder whose location is specified arbitrarily by the user. However, `fadnUtils` imposes to this folder a specific structure. This structure ensures good data management and maintenance. A data.dir folder is created with the `create.data.dir` function. In case we have already created the data.dir once before, we must specify which data.dir to work with, using the `set.data.dir` function.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

In order to work efficiently with R, we first need to convert the csv-data to an r friendly format, hereafter termed as Rdata. This is done using the `convert.to.fadn.raw.rds` function. The function converts the csv data (i.e. the data provided by DG-AGRI) to Rdata keeping the same original data structure (same columns, same rows). Next, since most often for the needs of an analysis, not all columns are needed and/or new columns are required to be calculated, the package facilitates the refinement of the original csv structure. This is done using the `convert.to.fadn.str.rds` function. This function transforms a raw Rdata file to a structured Rdata file. The raw Rdata file is a single table containing all columns and rows of the original csv-file. Instead, the structured Rdata file separates the original information to categories like identification information of each farm (id, weight, NUTS, etc.), costs, crops, livestock, etc.

The conversion of the raw Rdata file to a structured Rdata file is driven by a human-readable file, called `raw_str_map.file`. In this file the user can define what columns to keep and what transformations or new calculations to make. Below we present a script exhibiting the procedure for importing csv FADN data with the `fadnUtils` package.

Using a JSON file (raw_str_map.json), a raw Rdata can been converted into a structured Rdata. In order to convert this structured Rdata smoothly, it is necessary to check this user-defined JSON file, whether the needed column names/variables of JSON file are all in raw Rdata or in a csv file. This `check.column` function makes it flawless to calculate the aggregate variables.

```
# ............... IMPORT DATA IN TWO STEPS .....................................#

# You can import the file in two steps, one for converting
# the csv to fadn.raw.str (csv-data to raw Rdata) and
# one for converting the fadn.raw.rds to fadn.str.rds (raw Rdata
# to structured Rdata).
######################################################################
##             STEP 1: CONVERT CSV TO FADN.RAW.RDS              ##
######################################################################

##-----------------------------
## load each file separately
##-----------------------------
# load for a specific country germany: "DEU" and from a specific year: 2009
convert.to.fadn.raw.rds(
  file.path = paste0(fadn.data.dir ,"DEU2009.csv"),
  sepS = ",",
  fadn.country = "DEU",
  fadn.year = 2009
  #keep.csv = T # copy csv file in csv.dir
)

##-----------------------------
## load all csv files in a folder
##-----------------------------
"csv2raw function takes csv files in a folder and converts them into raw data"
allcsv2raw <- function(LocationofCSVFiles){

  # list all csv files
  csv_file_names <- list.files(path = LocationofCSVFiles, pattern= "*.csv$")

  #csv_file_names <- "DEU, BEL"
  for (file in csv_file_names){
    # extract first 3 char
    country = substr(file, 1, 3)

    # extract 4-7 char
    year = substr(file, 4, 7)
    #year = as.numeric(gsub("\\D+", "", file))
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
    convert.to.fadn.raw.rds(
      file.path = paste0(fadn.data.dir,file),
      sepS = ",",
      fadn.country = country,
      fadn.year = year
      #keep.csv = T # copy csv file in csv.dir

    )

  }
}

# load all csv
#allcsv2raw(fadn.data.dir)

##----------------------------
## load specific year and country
##----------------------------

"C.Y2raw function takes selected countries and years, then converts them into raw data"
C.Y2raw <- function(countries, years){
  for (country in countries){
    for (year in years){

      file = paste0(country,year,".csv")

      convert.to.fadn.raw.rds(
        file.path = paste0(fadn.data.dir,file),
        sepS = ",",
        fadn.country = country,
        fadn.year = year
        #keep.csv = T # copy csv file in csv.dir

      )

    }
  }
}

# load countries: BEL, DEU and NED
countriesList = c("BEL", "DEU", "NED")
yearsList = c(2009,2010,2011,2018)
#C.Y2raw(countries = countriesList, years =yearsList )

show.data.dir.contents()

# If you converted the csv to raw Rdata successfully, raw Rdata files are saved in "rds" fo
lder,
# the project's files and folders look like this:

# New_test_fadnUtils/
# +-- csv
# +-- fadnUtils.metadata.json
# +-- rds
# |   +-- fadn.raw.2009.BEL.compressed.rds
# |   +-- fadn.raw.2009.BEL.rds
# |   +-- fadn.raw.2010.BEL.compressed.rds
# |   +-- fadn.raw.2010.BEL.rds
# |   +-- fadn.raw.2011.BEL.compressed.rds
# |   +-- fadn.raw.2011.BEL.rds
# |   +-- fadn.raw.2012.BEL.compressed.rds
# |   \-- fadn.raw.2012.BEL.rds
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
# \-- spool
#     \-- readme.txt

#####################################################################
##         STEP 2: CONVERT FADN.RAW.RDS TO FADN.STR.RDS          ##
#####################################################################

#########################################################################################################
############
# Notices:#
############
## Before converting raw Rdata into str Rdata, it is recommended to use check.column() meth
od
## so that all variables in this json file can be converted.
## The conversion of the raw Rdata file to a structured Rdata file is driven by a human-rea
dable file,
## called raw_str_map.json.
## This json file is saved in extraction_dir by default.
## if you want to use raw_str_map.json by default, please put this file in extraction_dir.
## Or the user can define a external json file where it is
## and how to caculate the str Rdata.
#########################################################################################################
############

rds.dir = paste0(get.data.dir(),"/rds/")
# set a str name for for saving the str Rdata in rds.dir
new.str.name = "test"

# set a extraction_dir
dir.create(paste0(rds.dir, new.str.name))
new.extraction.dir = paste0(rds.dir, new.str.name)

##------------------------------------------------------------------------
##  Step 2.0: Check the variables of loaded a  raw rds data and a json file  --
##------------------------------------------------------------------------

# Save the modifed json file
list_vars = check.column(importfilepath = paste0(rds.dir, "fadn.raw.2009.BEL.rds"), # a rds
 file or a csv file
                         jsonfile = "D:/public/yang/MIND_STEP/2014_after_copy.json", # a js
on file
                         rewrite_json = TRUE, # write a new json file without unmatched var
iables
                         extraction_dir = new.extraction.dir # save the new json in extract
ion_dir
)

# Let's see the unmathcted variables in this json file
print(list_vars)

##-------------------------------------------------------------
##  Step 2.1: convert convert the raw Rdata into str Rdata   --
##-------------------------------------------------------------

#check the default json file in extraction_dir
if ("raw_str_map.json" %in% list.files(new.extraction.dir, pattern = "\\.json$")){
  cat(new.extraction.dir, "has a raw_str_map.json.", "\n")
}else{warning("please put a raw_str_map.json in ", new.extraction.dir,"\n", "Or using a ext
ernal json file (option 2)", "\n")}

##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
##  option 1: convert the file separately using a raw_str_map.json in extraction_dir
##  making sure that a raw_str_map.json is in extraction_dir
##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
convert.to.fadn.str.rds(fadn.country = "BEL",
                        fadn.year = 2009,
                        str.name = new.str.name # extraction_dir
)

##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
##  option 2: convert the file separately using a external json file
##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

# If force_external_raw_str_map is TRUE
# this external json file will be copied to extraction_dir as raw_str_map.json,
convert.to.fadn.str.rds(fadn.country = "BEL",
                        fadn.year = 2009,
                        raw_str_map.file = "D:/public/yang/MIND_STEP/new_sample/raw_str_ma
p.json", # a external json file
                        str.name = new.str.name, # extraction_dir
                        force_external_raw_str_map = T,
                        DEBUG = F
)

#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
#option 3: convert mutilple raw Rdata files in rds.dir into str Rdata in extraction_dir
#:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

"raw2str function will help us to convert the raw Rdata into the str Rdata in rds.dir compl
etely.
This function takes a user-defined raw_str_map.file and a logical constant which is FALSE b
y default,
and converting raw data to str data."
raw2str <- function(Current_raw_str_map.file = NULL, overwrite_external_json = F){


  rds.dir  = paste0(get.data.dir(), "/rds")

  raw_file_names <- dir(rds.dir, pattern ="\\.rds$" )

  for (file in raw_file_names){

    if (!grepl("compressed", file)){

      # extract first 3 char
      country = substr(file, 15, 17)

      # extract number
      year = as.numeric(gsub("\\D+", "", file))

      cat("converting the str data for country: ", country, " and year: ", year, "\n")
      tryCatch(
        expr = {
          convert.to.fadn.str.rds(fadn.country = country,
                                  fadn.year = year,
                                  raw_str_map.file = Current_raw_str_map.file,
                                  force_external_raw_str_map =  overwrite_external_json,
                                  str.name = new.str.name
          )
        },
        warning = function(w){
          message('Caught an warning!')
          print(w)
        },
        error = function(e) {
          message("Caught an error! Please check the objects in json file using check.colum
n() (see more in USE_CASE_4.R).")
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
          #cat("Wrong, can't convert the str Rdata!",sep = "\n")
          print(e)
        }
      )

    }

    else{cat("It's compressed data!",sep = "\n")}


  }
}
#-----------------------------------------------------------------
# option 3.1: using a raw_str_map.json by defaut
#-----------------------------------------------------------------
# convert str data with a default json file, make sure that a raw_str_map.json in extractio
n_dir
#raw2str()
# or
#raw2str(Current_raw_str_map.file = "D:/public/yang/MIND_STEP/new_sample/test01/raw_str_ma
p.json", overwrite_external_json = F)

#-----------------------------------------------------------------
# option 3.2: using a external json file
#-----------------------------------------------------------------

# convert str data using a external json file
#raw2str(Current_raw_str_map.file = "D:/public/yang/MIND_STEP/new_sample/test01/raw_str_ma
p.json", overwrite_external_json = T)

show.data.dir.contents()


# If str Rdata was converted, the str Rdata is saved in "test"(new.str.name) folder as belo
w.

# New_test_fadnUtils/
# +-- csv
# +-- fadnUtils.metadata.json
# +-- rds
# |   +-- fadn.raw.2009.BEL.compressed.rds
# |   +-- fadn.raw.2009.BEL.rds
# |   +-- fadn.raw.2010.BEL.compressed.rds
# |   +-- fadn.raw.2010.BEL.rds
# |   +-- fadn.raw.2011.BEL.compressed.rds
# |   +-- fadn.raw.2011.BEL.rds
# |   +-- fadn.raw.2012.BEL.compressed.rds
# |   +-- fadn.raw.2012.BEL.rds
# |   \-- test
# |       +-- fadn.str.2009.BEL.rds
# |       +-- raw_str_map.json
# |       \-- rewrite_2014_after_copy.json
# \-- spool
#     +-- my_logfile.txt
#     \-- readme.txt
```

4.1.2.3. Load Rdata from a data.dir

In order to initiate any analysis with `fadnUtils`, we first need to load r-friendly data. We can only load data for countries and years that that has already been imported into a data.dir folder (see use case 4.1.2.1). In order to inspect the available country-year pairs in the current data.dir, we use the `show.data.dir.contents` function. In order to load raw Rdata (fadn csv data in r-friendly format) we use the `load.fadn.raw.rds`, while for structured Rdata we use the `load.fadn.str.rds`

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

function. In both functions the user can define the combinations of countries and years to be loaded. The result of this process is a data structure that contains all the requested countries and years. More specifically, when loading raw Rdata, using `load.fadn.raw.rds`, the result is a single `data.table` having all the columns and all the rows of the original csv files requested. When loading structured Rdata, using `load.fadn.str.rds`, the result is a list containing individual data tables on general farm properties, costs, crops and livestock for all years and countries requested. Details on the structure returned from loading data are given in the package reference section. Below we provide an example of how to load data.

```
#####################################################################
##                        LOAD RAW RDATA                           ##
#####################################################################
### We can either load raw Rdata files (the original FADN csv in r-friendly format),
### or structured Rdata files (the original data transformed into meaningful
### information)
# To load raw Rdata, only for BEL and 2009
my.data = load.fadn.raw.rds(
  countries = "BEL",
  years = 2009
)

# my.data is a single large data.table, with the original csv columns and rows
nrow(my.data)  #Number of rows
names(my.data)  #Column names
length(names(my.data)) #Number of columns
str(my.data) #Overall structure

#####################################################################
##                      LOAD STRUCTURED RDATA                      ##
#####################################################################

#To load structured data, for BEL and 2009
my.data.2009 = load.fadn.str.rds(
  countries = "BEL",
  years = 2009,
  extraction_dir = new.str.name # Location of the str Rdata
)


# You can see that my.data is a list, with three elements: info, costs, crops
str(my.data.2009)

# You can access each individual element like this
str(my.data.2009$info)
#str(my.data.2009$costs)
#str(my.data.2009$crops)


# The first columns of each of the above elements (info, costs, crops)
#   are created according to the ID section of the raw_str_map
names(my.data.2009$info)
names(my.data.2009$costs)
names(my.data.2009$crops)


# info and costs data.tables are in wide-format (each observation in a single row,
#   all attributes of a single observation in different columns).
# crops element is in long format (one observation is in many rows,
#
#
# See https://seananderson.ca/2013/10/19/reshape/ for
#   discussion of the two types of data formats
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
head(my.data.2009$info)
head(my.data.2009$costs)
head(my.data.2009$crops)



# Also on the attributes section of each of the above elements, we can access
#   the column formulas and descriptions, as defined in the raw_str_map file.
# View(
#   attr(my.data.2009$info,"column.descriptions")
# )
# View(
#   attr(my.data.2009$costs,"column.descriptions")
# )
# View(
#   attr(my.data.2009$crops,"column.descriptions")
# )


# Especially for the crops element, we can also see the description
#   CROP column
# View(
#   attr(my.data.2009$crops,"crops.descriptions")
# )

######################################################################
##              LOAD COUNTRIES-YEARS COMBINATIONS                   ##
######################################################################
###  In the previous examples, we showed how to load data for one country and
###  one year In the following examples we show more combinations.

#To load for DEU and NED for year 2015
my.data = load.fadn.str.rds(countries = c("DEU","NED"), years = c(2009,2010,2011), extracti
on_dir = new.str.name )


#To load for DEU and NED for all years
my.data = load.fadn.str.rds(countries = c("DEU","NED"),extraction_dir = new.str.name )


#To load all available countries for year 2015
my.data = load.fadn.str.rds(years = 2015, extraction_dir = new.str.name)


#To load all availabel data
# my.data = load.fadn.str.rds(extraction_dir = new.str.name)
```

4.1.2.4. Perform analysis/transformations

Below we provide various examples of the powerfulness of using the `fadnUtils` package to perform data analysis on FADN data.

```
# We load structured data for all available years and countries
my.str.data = load.fadn.str.rds(extraction_dir = new.str.name)

# We load structured data for all available years for Germany, Niederlande and Belgien.
my.str.data.DEU.NED.BEL = load.fadn.str.rds( countries =c("DEU", "NED", "BEL"),extraction_d
ir = new.str.name)

##---------------------------------------------------------------
##         HOW MANY FARMS FOR EACH COUNTY AND EACH YEAR         --
##---------------------------------------------------------------
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
# we use the info DT, and group by YEAR-COUNTRY
my.str.data$info[,.N,by=list(YEAR,COUNTRY)]

##      YEAR COUNTRY    N
##   1: 2004     NED 1397
..
## 333: 2018     UKI 2848

#We can also use dcast, to show a more tabular format
dcast(
  my.str.data$info,
  YEAR~COUNTRY,
  fun.aggregate = length,
  value.var =
)
##     YEAR  BEL  BGR CYP  CZE  DAN  DEU  ELL  ESP EST  FRA  HRV  HUN  IRE   ITA
##  1: 2004    0    0   0    0    0    0    0    0   0    0    0    0    0     0
..
## 15: 1166 448 998 489 1498 1872 12271 2056 5085 722 1010 559  890 2848

# We can also export to clipboard, using the write.excel utility function
#   After running the following command, open excel and paste. The result will appear.
write.excel(
  dcast(
    my.str.data$info,
    YEAR~COUNTRY,
    fun.aggregate = length,
    value.var =
  )
)
##------------------------------------------------------------------
##               ALL CROP AREAS PER COUNTRY-YEAR                --
##------------------------------------------------------------------

# First, calculate the weighted area
my.str.data$crops[
  VARIABLE=="LEVL",
  VALUE.w:=WEIGHT*VALUE/1000
  ]

# Then dcast that variable
dcast(
  my.str.data$crops[VARIABLE=="LEVL"],
  COUNTRY+CROP~YEAR,
  value.var = "VALUE.w",
  fun.aggregate = sum,
  na.rm = T
)

##------------------------------------------------------------------
##             ALL CROP PRODUCTION PER COUNTRY-YEAR             --
##------------------------------------------------------------------


dcast(
  my.str.data$crops[VARIABLE=="GROF",VALUE.w:=WEIGHT*VALUE/1000],
  COUNTRY+CROP~YEAR,
  value.var = "VALUE.w",
  fun.aggregate = sum,
  na.rm = T
)

##------------------------------------------------------------------
##             BARLEY PRODUCTION PER COUNTRY-YEAR             --
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
##-------------------------------------------------------------------
dcast(
  my.str.data$crops[
    VARIABLE=="GROF" & CROP=="BARL",
    VALUE.w:=WEIGHT*VALUE/1000
    ],
  COUNTRY~YEAR,
  value.var = "VALUE.w",
  fun.aggregate = sum,
  na.rm = T
)

##-------------------------------------------------------------------
##       DISTRIBUTION OF NUMBER OF CROPS PER COUNTRY-YEAR       --
##-------------------------------------------------------------------
# for countries: NED, BEL and DEU
crops.data = my.str.data.DEU.NED.BEL$crops #catering for easier access at next steps

#this contains the number of crops for each farm-country-year/
#   Be carefule, we hav to filter to count only the LEVL variable
crops.data.Ncrops = crops.data[VARIABLE=="LEVL",.N,by=list(COUNTRY,YEAR,ID)]

# This displays the quantiles of the number of crops
crops.data.Ncrops[,as.list(quantile(N)),by=list(YEAR,COUNTRY)][order(COUNTRY)]

# R excels on graphic representation of results
library(ggplot2)

ggplot(crops.data.Ncrops,aes(y=N,x=1)) +
  geom_boxplot() +
  facet_grid(YEAR~COUNTRY) +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank()
  )+
  ylab("Number of Crops")
```

4.1.2.5. Collect common id, some calculations, and plots

Using `collect.common.id` function, common id will be collected over years for further development. In this sub-section, some examples will be represented. Below are examples of collection id based on already loaded raw Rdata or str Rdata. This function returns all the common id over years.

```
##-------------------------------------------------------------------------
##  COLLECT COMMON ID FROM LOADED STRUCTURED RDATA AND LOADED RAW R_DATA  --
##-------------------------------------------------------------------------

# Collection the common id from loaded str Rdata
collected.common.id_str = collect.common.id(my.str.data)

## Tranforming  list  to data table....
##  [1] 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018
## 15  year(s) is/are selected.

print(collected.common.id_str)

##            common_id
##   1: xxx
..
## 446:  xxx
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
# Collection the common id from loaded raw Rdata (Year: 2015 and all countries)
collected.common.id_raw = collect.common.id(my.data)
## Tranforming  list  to data table....
## [1] 2015
## 1  year(s) is/are selected.

print(collected.common.id_raw)

##             common_id
##     1:    xxx
..
## 82068: xxx

##-----------------------------------------------------------------
##          CALCULATION BASED ON COLLECETED COMMON ID          --
##-----------------------------------------------------------------

# sommaries for infos

# sample and representend number of farms
my.str.data$info[,list(Nobs_sample=.N,Nobs_represented=sum(WEIGHT)),
                 by=.(COUNTRY,YEAR)]

##      COUNTRY YEAR Nobs_sample Nobs_represented
##   1:     NED 2004        1397            60644
##   2:     NED 2005        1446            60598


# only for full sample (common id over years in selected data)
my.str.data$info[ID %in% collected.common.id_str[[1]],
                 list(Nobs_sample=.N,
                      Nobs_represented=sum(WEIGHT)),
                 by=.(COUNTRY,YEAR)]

##     COUNTRY YEAR Nobs_sample Nobs_represented
##  1:     NED 2004         446         20358.73
##  2:     NED 2005         446         20209.66
..


# some summaries for crops

# unweighted and weighted sum over countries, years, crops and variables (EAAP  GROF  INTF
 LEVL SHARE  UVAG  UVSA)
# EAA: Economic Accounts of Agriculture
# EAAP: EAA value at producer prices
# GROF

my.str.data$crops[ ,list(VALUE=sum(VALUE),
                         VALUE_weighted=sum(VALUE*WEIGHT)),
                   by=.(COUNTRY,YEAR,CROP,VARIABLE)]

##        COUNTRY YEAR CROP VARIABLE       VALUE VALUE_weighted
##     1:     NED 2004 SWHE     LEVL     3506.61   1.033372e+05
##     2:     NED 2004 SWHE     GROF    30460.00   8.763908e+05
...

# only for full sample (common id over years in selected data)
my.str.data$crops[ID %in% collected.common.id_str[[1]],
                  list(VALUE=sum(VALUE),
                       VALUE_weighted=sum(VALUE*WEIGHT)),
                  by=.(COUNTRY,YEAR,CROP,VARIABLE)]
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

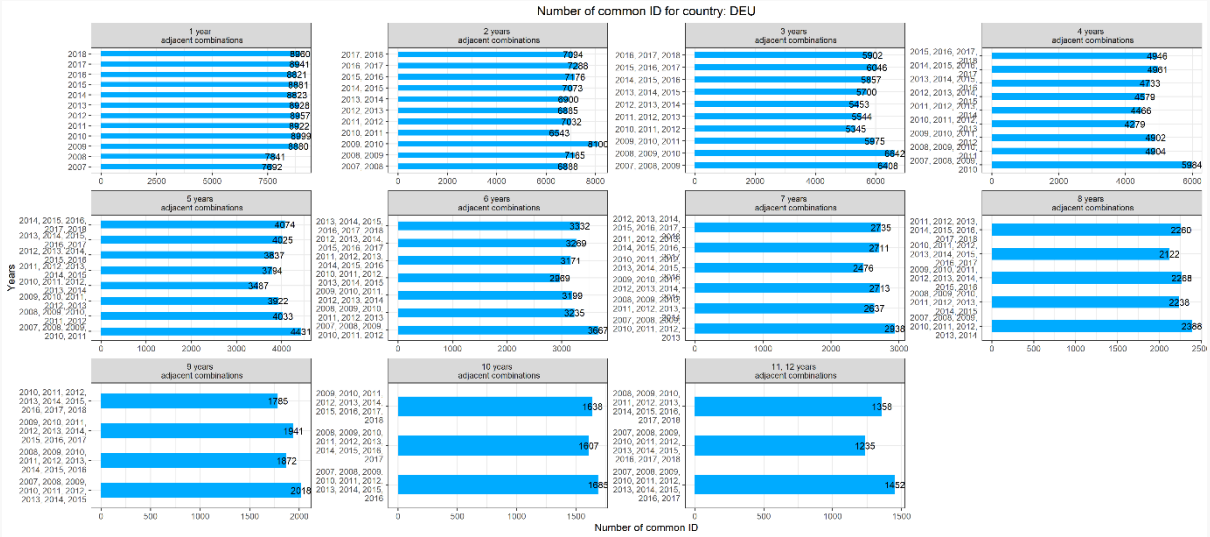
```
##        COUNTRY YEAR CROP VARIABLE        VALUE VALUE_weighted
##     1:     NED 2004 SWHE     LEVL    1570.010       41569.95
##     2:     NED 2004 SWHE     GROF   13910.200      361948.09



my.str.data$crops[ID %in% collected.common.id_str[,common_id],
                  list(VALUE=sum(VALUE),
                       VALUE_weighted=sum(VALUE*WEIGHT)),
                  by=.(COUNTRY,YEAR,CROP,VARIABLE,ID)]

##        COUNTRY YEAR CROP VARIABLE             ID VALUE VALUE_weighted
##     1:     NED 2004 SWHE     LEVL 26000000015782 12.00      918.98734
##     2:     NED 2004 SWHE     LEVL 26000000026867  3.50      175.00000


##-------------------------------------------------------------
## Load fadn raw data and search the the number of common id for adjacent combination years
##-------------------------------------------------------------

"find all adjacent combinations in a list"
myFun <- function(Data) {
  A <- lapply(1:(length(Data)), sequence)
  B <- lapply(rev(lengths(A))-1L, function(x) c(0, sequence(x)))
  unlist(lapply(seq_along(A), function(x) {
    lapply(B[[x]], function(y) Data[A[[x]]+y])
  }), recursive = FALSE, use.names = FALSE)
}

"add a string to the facet label text and split it in two lines"
label_facet <- function(original_var, custom_name){
  lev <- levels(as.factor(original_var))
  lab <- paste0(lev, " \n ",custom_name)
  names(lab) <- lev
  return(lab)
}

"multi-panel plots using facet_wrap() for dynamic choice"
figure <- function(country, df, n){
  p = df %>%
    # reorder by Num_id
    #ggplot( aes(x = reorder(Years, -Num_id) ,y=Num_id))
    ggplot( aes(x = Years ,y=Num_id))+
    geom_bar( stat="identity",
              position = position_dodge(width = 0.8),
              width=0.5,
              #fill = rainbow(n=length(df$Num_id))
              fill = "#00abff"
    ) +
    coord_flip()+
    #labs(title = paste0("Plot of the Number of common ID for country: ", country ), fill =
 "Years") +
    xlab("Years") +
    ylab("Number of common ID") +
    geom_text(aes(label=Num_id), vjust=0.5, colour="black", size=3.5)+
    theme(axis.text.x=element_text(color = "black", size=6, angle=0, vjust=.8, hjust=0.8))
+
    scale_x_discrete(labels = function(x) str_wrap(x, width = n)) +
    facet_wrap( ~ group, scales="free",
                #labeller=names
                labeller = labeller(group = label_facet(df$group, "adjacent combinations"))
    )+
    ggtitle(paste0("Number of common ID for country: ", country )) +
    theme_bw() +
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
    theme(plot.title = element_text(hjust = 0.5))
  p
}

"load raw data and get the number of common id for selected countries over all exist adjace
nt combinations years.
then save the number of common id to an excel sheet and plot"
output_common_id <- function(countries_list, saveExcel = TRUE, excelname , savePlots = TRU
E){

  rds.dir = paste0(get.data.dir(),"/rds/")
  plots.dir = paste0(get.data.dir(), "/plots/")
  if (!dir.exists(plots.dir)) dir.create(plots.dir)
  library(xlsx)
  library(openxlsx)
  xlsx_file_dir <- paste0(get.data.dir(), "/spool/")
  if (saveExcel==TRUE) {wb <- createWorkbook(paste0(xlsx_file_dir, excelname))}
  outlist = list()
  for (country in countries_list){
    cat("Country:", country, '\n')
    raw_file_names <- dir(rds.dir, pattern = paste0(country, "\\.rds$") )
    years_list = as.numeric(gsub("\\D+", "", raw_file_names))
    adjacent_list = myFun(years_list)
    my.data = list()
    for (year_items in adjacent_list) {
      name = toString(year_items)
      data = load.fadn.raw.rds(countries = country, years = year_items)
      my.data[[name]] = data
    }
    Big.Num.Common.id = list()
    for (data_list in names(my.data)){
      common.id = collect.common.id(my.data[[data_list]])
      Big.Num.Common.id[[data_list]] = nrow(common.id)
    }
    DF = do.call(rbind, Big.Num.Common.id)
    DF = data.frame(DF)
    colnames(DF) <-  "Num_id"
    DF$Years <-  row.names(DF)
    outlist[[country]] = DF
    if (!is.null(wb)) {
      if (!(country %in% names(wb))) {
        addWorksheet(wb, country)}
      writeData(wb,country, DF)
      }
    if (savePlots == TRUE){
      library(ggplot2)
      library(stringr)
      DF$Length <- str_count(DF$Years)
      DF$group <- cut(DF$Length, breaks=c(1,5,14,18,25,30,35,40,48,55,58,70,Inf))
      levels(DF$group) <- c("1 year","2 years","3 years","4 years","5 years","6 years","7 y
ears",
                                      "8 years","9 years","10 years","11, 12 years",">12 ye
ars")
      if (length(years_list) >=15){
        p <- figure(country,DF, 35)}
      else{
        p <- figure(country,DF, 20)}

      ggsave(plot = p,
             filename = paste0(plots.dir,country ,"_plot.png"),
             width = 18, height = 8)}
    }
    if (saveExcel == TRUE) {
      saveWorkbook(wb, paste0(xlsx_file_dir, excelname), overwrite = T)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
      cat(excelname," is saved in ",xlsx_file_dir, "\n")}
    if (savePlots == TRUE) cat("plots are saved in", plots.dir, "\n")

  return(list(outlist,p))
  }

# get all countires in fadn str data
countries = unique(my.str.data$info$COUNTRY)
#ID_list <- output_common_id(countries_list = countries)

# get Germany: DEU
DEU_list <- output_common_id("DEU"), saveExcel = TRUE, excelname = "DEU.xlsx", savePlots =
TRUE)

# plot
DEU_list[[2]]
```



4.1.3. Data directory structure

Table 4 describes the imposed structure of the data.dir directory which is build by `create.data.dir` function.

Table 4: Data.dir imposed structure

Type	Name	Description
Directory	./csv	The original FADN csv files are stored here
Directory	./rds	RDS of the original csv files are stored here (*.fadn.raw.rds)
Directory	./rds/<FOLDER>	An extraction of the raw data (*.fadn.extr.rds)
Directory	./spool	The user can keep here any related file, e.g. the data warehouse excel request file
File	fadnUtils.metadata.json	Contains metadata on the data.dir (description, etc.)
File	raw_str_map.json	Contains the mapping from the fadn.raw.rds to the fadn.str.rds data

Source: Own compilation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

4.2. FADNutils: Data interface to EU FADN data: User Case 2

This chapter uses functions developed in the fadnUtils package to have a first look at the provided FADN data. There are also some functions and code that is not part of the fadnUtils package. Some code snippets do not contain R output as this is not necessary for this use case.

4.2.1. Preliminaries

4.2.1.1. Clean environment

```
rm(list=ls())
```

4.2.1.2. Load libraries

```
requiredPackages = c('tidyverse','readxl','skimr','data.table','fadnUtils')
for(p in requiredPackages){
 if(!require(p,character.only = TRUE)) install.packages(p)
 library(p,character.only = TRUE)
}
```

4.2.2. Convert csv FADN files to Rdata

4.2.2.1. Declare folder structure and settings

We start by converting the csv FADN files to Rdata files. Therefore,

- we provide the project directory, in which the Rdata files should be copied to
- we provide the location of the raw csv FADN files
- we run create.data.dir to prepare the project directory
- we run set.data.dir to declare that we are working with it
- we finish this by defining and running the allcsv2raw function

> *If you have already done this conversion, you can skip it.*

```
# set project directory
CurrentProjectDirectory <- "D:/data/fadn/lieferung_20210414/neuenfeldt/test_fadnUtils"
# the path of the fadn files for loading
fadn.data.dir <- "D:/data/fadn/lieferung_20210414/csv/"
# ceate a data.dir
create.data.dir(folder.path = CurrentProjectDirectory)
# Once the data.dir is created, we must declare that we are working with it
set.data.dir(CurrentProjectDirectory)

get.data.dir()
```

4.2.3. Load raw Rdata

Load raw FADN data.

```
## LOAD RAW R-DATA
countries_all <- get.available.fadn.raw.rds() %>%
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
 filter(str_detect(pattern = "compressed",string = COUNTRY,negate = TRUE)) %>%
 pull(COUNTRY) %>% unique()
system.time({
 my.data <- load.fadn.raw.rds(countries = countries_all)
}) # ~ 1620.42 sec
```

Dimension of the data (rows and columns):

```
dim(my.data)
```

```
## [1] 988163   2277
```

4.2.4. Use cases

4.2.4.1. Test number of countries and years - compare with request

Therefore load request xlsx and prepare for comparison with provided data.

```
file_names <- get.available.fadn.raw.rds() %>%
 filter(str_detect(pattern = "compressed",
                   string = COUNTRY,
                   negate = TRUE)) %>%
 rename(year=YEAR,country=COUNTRY) %>%
 mutate(provided="y")
data_requested_general <- read_xlsx(
 path = "D:/public/data/fadn/lieferung_20210414/FADN data request forms_March2021_MM210326.xlsx",
 sheet = 2,
 range = "A365:P393")
colnames(data_requested_general)[1] <- "country"
data_requested_general <- data_requested_general %>%
 mutate(country=str_sub(country,2,4))
data_requested_general_long <- data_requested_general %>%
 pivot_longer(-country)
data_requested_general_long <- data_requested_general_long %>%
 filter(value=="y") %>%
 rename(year=name,requested=value)
requested_and_provided <- data_requested_general_long %>% full_join(file_names,by=c("country","year")) %>%
 filter(is.na(requested ) | is.na(provided))
## if empty, than everything is fine
requested_and_provided
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: country <chr>, year <chr>, requested <chr>,
## #   provided <chr>
```

4.2.4.2. Check number of provided variables

```
files <- my.data
files %>% select(YEAR,COUNTRY,countryyear)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
##        YEAR COUNTRY countryyear
##     1: 2004     NED     NED2004
..
## 988163: 2018     UKI     UKI2018
```

```
## Number of sum of sample farms - observation - not unique
files %>% select(YEAR,COUNTRY) %>% group_by(COUNTRY) %>% count() %>% print(n=28)
```

```
## # A tibble: 28 x 2
## # Groups:   COUNTRY [28]
##    COUNTRY      n
##    <chr>    <int>
##  1 BEL      14189
..## 28 UKI      33577
```

```
## Combination of country and year in data set
files %>% select(YEAR,COUNTRY) %>% group_by(COUNTRY) %>% distinct()
```

```
## # A tibble: 333 x 2
## # Groups:   COUNTRY [28]
##     YEAR COUNTRY
##    <int> <chr>
##  1  2004 NED
..
## 10  2007 ELL
## # ... with 323 more rows
```

4.2.4.3. Check missing variables

```
data_requested_variables <- read_xlsx(
 path = "D:/public/data/fadn/lieferung_20210414/FADN data request forms_March2021_MM210326.xlsx",
 sheet = 3,
 range = "A3:O4885")
data_requested_variables <- data_requested_variables %>%
 select(`name from 2014`,DESCRIPTION,`COMMON name`,`Combined Request`) %>% filter(`Combined Request`==1 & !is.na(`Combined Request`))
data_requested_variables %>% skimr::skim()
```

Data summary

Name	Piped data
Number of rows	2278
Number of columns	4
_____	
Column type frequency:	
character	3
numeric	1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
name from 2014	279	0.88	2	17	0	1998	0
DESCRIPTION	0	1.00	4	147	0	2269	0
COMMON name	0	1.00	2	25	0	2278	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Combined Request	0	1	1	0	1	1	1	1	1	▁▁▇▁

```
## Which variables are missing
vars_requested_and_missing <- data_requested_variables %>%
 pull(`COMMON name`) %>% setdiff(.,colnames(files))
vars_requested_and_missing

## [1] "CYNG_FUV"  "CYNG_IRTA" "CYNG_PRQ"  "CYNG_SV"

## Which variables are in advance
vars_provided_not_requested <- data_requested_variables %>%
 pull(`COMMON name`) %>% setdiff(colnames(files),.)
vars_provided_not_requested

## [1] "countryyear"  "load.YEAR"    "load.COUNTRY"

## Number of requested and provided variables
data_requested_variables %>% pull(`COMMON name`) %>%
 length()

## [1] 2278

colnames(files) %>% length()

## [1] 2277

## Print and export difference in requested and provided variables
vars_requested_and_missing <- tibble(
 vars_requested_and_missing=vars_requested_and_missing,
 name=data_requested_variables %>%
  filter(`COMMON name` %in% vars_requested_and_missing) %>%
  pull(DESCRIPTION))
vars_requested_and_missing

## # A tibble: 4 x 2
##   vars_requested_and_missing name
##   <chr>                      <chr>
## 1 CYNG_FUV                   Growth of young plantations Farm use
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
## 2 CYNG_IRTA          Growth of young plantations Irrigated crop total a~
## 3 CYNG_PRQ           Growth of young plantations Production
## 4 CYNG_SV            Growth of young plantations Sales value
```

4.2.4.4. Are the ID'S and farm type variables provided?

```
files %>% select(ID)
```

4.2.4.5. Is type of farming (TF14) provided?

```
## Is type of farming (TF14) provided?
files %>% select(contains("TF"))
TF_vars <- files %>% select(contains("TF")) %>% colnames()
data_requested_variables %>% filter(`COMMON name` %in% TF_vars)
## How many different farm types are provided?
files %>% select(TF) %>% distinct() %>% count()
files %>% select(TF14) %>% distinct() %>% count()
files %>% select(TF8) %>% distinct() %>% count()
files %>% select(TF) %>% distinct() %>% arrange(TF)
```

4.2.4.6. Are the NUTS information provided?

```
files %>% select(contains("NUTS"))
## Are there NA's in the NUTS3?
files %>% select(COUNTRY,YEAR,NUTS3) %>% filter(is.na(NUTS3))
## Are there zeros in the NUTS3?
files %>% select(COUNTRY,YEAR,NUTS3) %>% filter(NUTS3==0)
## Are there no NUTS3 information provided?
files %>% select(COUNTRY,YEAR,NUTS2:NUTS3) %>% filter(NUTS3=="")
## Show all NUTS3
files %>% pull(NUTS3) %>% unique()
```

4.2.4.7. Are there any NA's?

```
system.time({
  files_na <- files %>% summarise_all(.funs = function(x)sum(is.na(x)))
}) # 11 sec

##    user  system elapsed
##    7.84    2.65   10.54

files_na %>% pivot_longer(-YEAR) %>% filter(value!=0) # no NA's

## # A tibble: 0 x 3
## # ... with 3 variables: YEAR <int>, name <chr>, value <int>
```

4.2.4.8. Means over all variables

Means over all variables to get a feeling of variables that have ONLY zeros - variables that provide no content.

4.2.4.9. Over years and countries

Over years and countries - does only work for numeric variables, but that is not a problem here.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
files_summary <- tibble(files) %>%
 summarise_all(.funs=function(x)mean(x,na.rm=TRUE))
## Number of variables which have only zeros
files_summary %>% group_by(YEAR) %>%
 select(-ID) %>% pivot_longer(-YEAR) %>%
 filter(value==0) %>% count(YEAR)
```

```
## # A tibble: 1 x 2
## # Groups:   YEAR [1]
##    YEAR     n
##   <dbl> <int>
## 1 2013.   121
```

```
## Over years for each country
files_summary_COUNTRY <- tibble(files) %>%
 group_by(COUNTRY) %>%
 summarise_all(.funs=function(x)mean(x,na.rm=TRUE))
files_summary_COUNTRY
```

```
## # A tibble: 28 x 2,277
##    COUNTRY ACCTYP AFRMBLD_DY AFRMBLD_IP AFRMBLD_OV AFRMBLD_SUB AFRMBLD_SV
##  * <chr>   <dbl>     <dbl>     <dbl>     <dbl>      <dbl>      <dbl>
##  1 BEL     0.789     8630.     8862.     87199.     803.       208.
..

## #   CCITRFRUT_IRA_X <dbl>, CCITRFRUT_IRPRQ_X <dbl>, ...
```

```
## Number of variables which have only zeros - per country
files_summary_COUNTRY %>% select(-ID) %>%
 pivot_longer(-COUNTRY) %>% filter(value==0) %>%
 select(-value) %>% unique() %>%
 count(COUNTRY) %>% print(n=28)
```

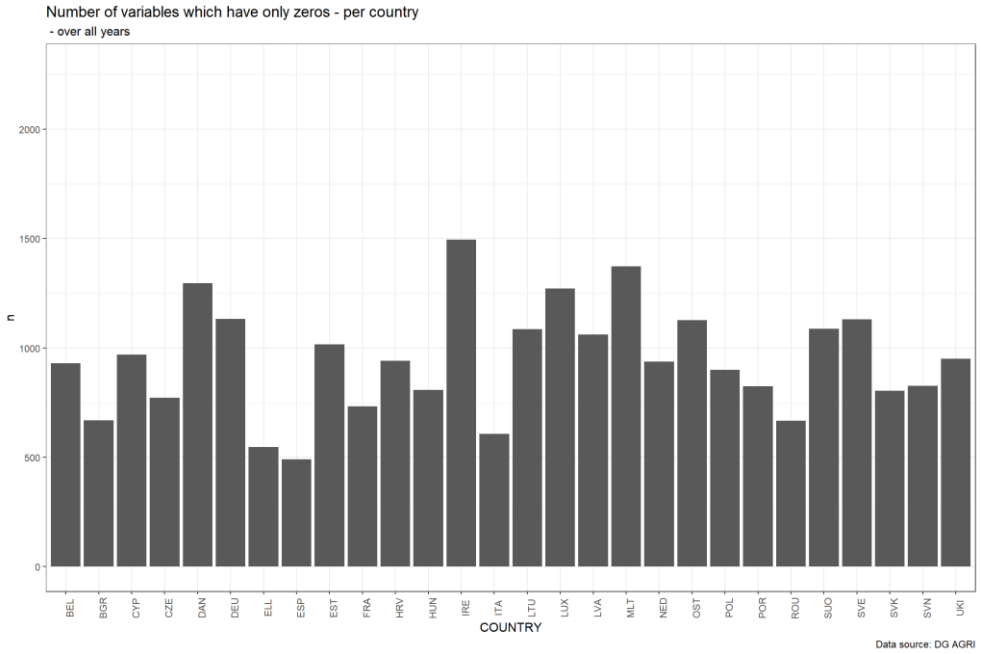
```
## # A tibble: 28 x 2
##    COUNTRY     n
##  * <chr>   <int>
##  1 BEL       930
##  2 BGR       669
..

## 28 UKI       951
```

4.2.4.10. Number of variables which have only zeros - per country

```
p <- files_summary_COUNTRY %>% select(-ID) %>%
 pivot_longer(-COUNTRY) %>% filter(value==0) %>%
 select(-value) %>% unique() %>%
 count(COUNTRY) %>%
 ggplot(aes(x=COUNTRY,y=n)) +
 geom_col() +
 ylim(c(0,ncol(files))) +
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
 labs(title = "Number of variables which have only zeros - per country",
     subtitle = " - over all years",
     caption = "Data source: DG AGRI") +
 theme_bw() +
 theme(axis.text.x = element_text(angle = 90))
p
```




```
ggsave(plot = p,
     filename = paste0(CurrentProjectDirectory,"/vars_zero_country.png"),
     width = 12, height = 8)
```

4.2.4.11. Number of variables which have only zeros - per year

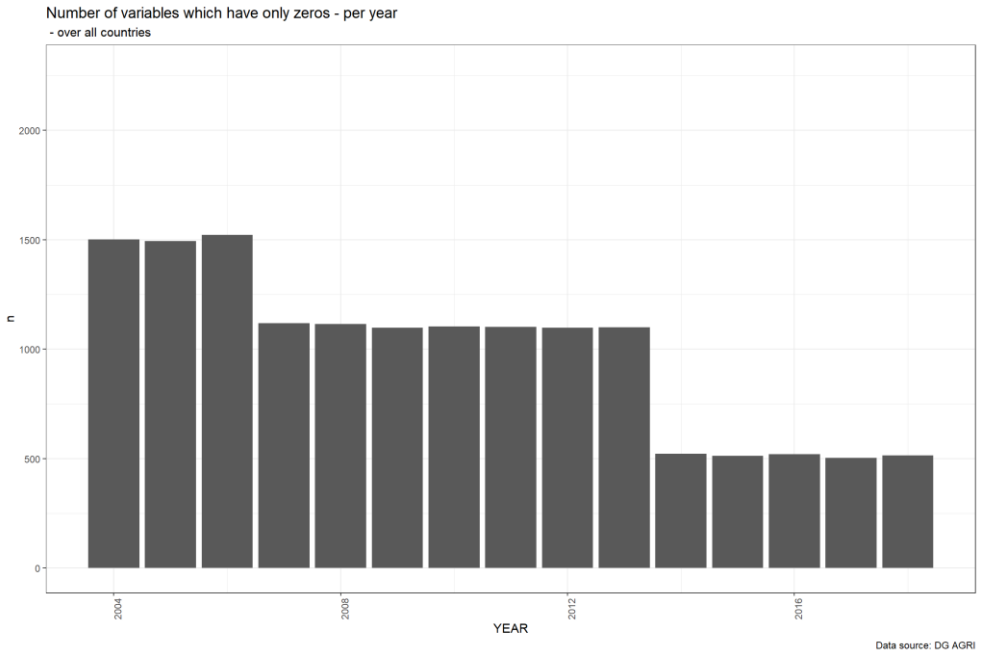
```
files_summary_YEAR <- tibble(files) %>%
 group_by(YEAR) %>%
 summarise_all(.funs=function(x)mean(x,na.rm=TRUE))
files_summary_YEAR %>% select(-ID) %>%
 pivot_longer(-YEAR) %>%
 filter(value==0) %>%
 count(YEAR)
```

```
## # A tibble: 15 x 2
##    YEAR    n
##  * <int> <int>
##  1  2004  1501
..

## 15  2018   515
```

```
## figure  - Number of variables which have only zeros - per year
p <- files_summary_YEAR %>%
 select(-ID) %>% pivot_longer(-YEAR) %>%
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
 filter(value==0) %>% select(-value) %>%
 unique() %>%
 count(YEAR) %>%
 ggplot(aes(x=YEAR,y=n)) +
 geom_col() +
 ylim(c(0,ncol(files))) +
 labs(title = "Number of variables which have only zeros - per year",
     subtitle = " - over all countries",
     caption = "Data source: DG AGRI") +
 theme_bw() +
 theme(axis.text.x = element_text(angle = 90))
p
```

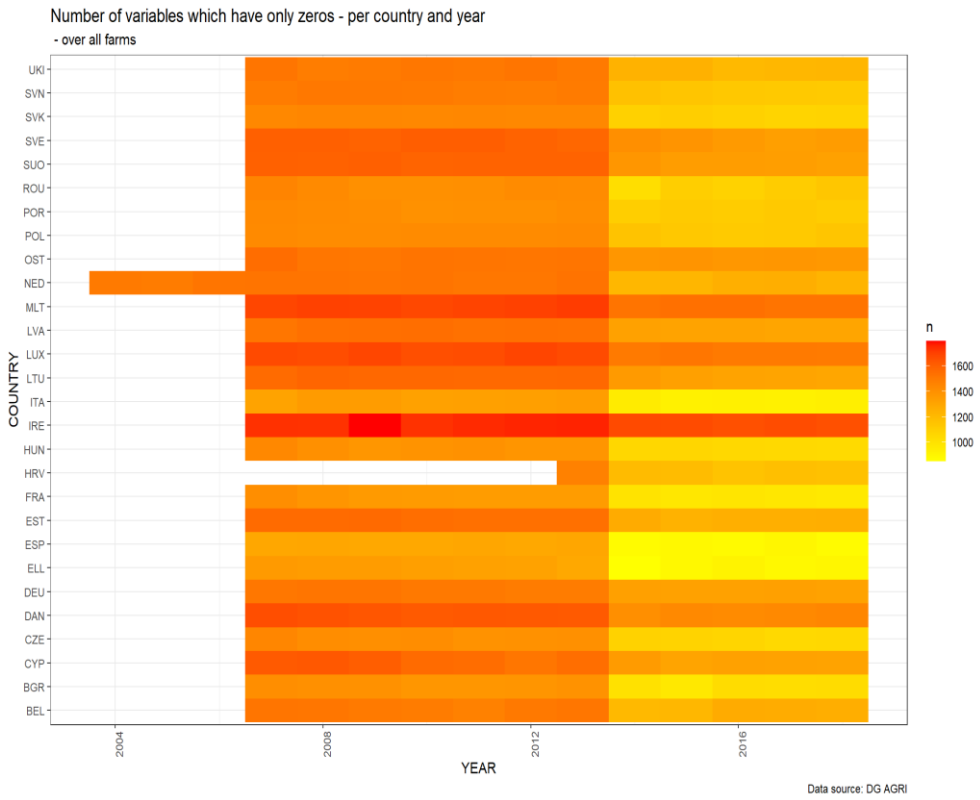



```
ggsave(plot = p,
     filename = paste0(CurrentProjectDirectory,"/vars_zero_years.png"),
     width = 12, height = 8)
```

4.2.4.12. Number of variables which have only zeros - per year and country

```
files_summary_COUNTRY_YEAR <- tibble(files) %>% group_by(COUNTRY,YEAR) %>% summarise_all(.
funs=function(x)mean(x,na.rm=TRUE))
result <- files_summary_COUNTRY_YEAR %>% select(-ID) %>% pivot_longer(-c(COUNTRY,YEAR)) %
>% filter(value==0) %>% count(COUNTRY,YEAR)
## Figure - Number of variables which have only zeros - per year and country
p <- result  %>% ggplot(aes(x=YEAR,y=COUNTRY,fill=n)) + geom_tile() +
 labs(title = "Number of variables which have only zeros - per country and year",
     subtitle = " - over all farms",
     caption = "Data source: DG AGRI") +
 theme_bw() + scale_fill_gradient(low="yellow", high="red") +
 theme(axis.text.x = element_text(angle = 90))
p
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

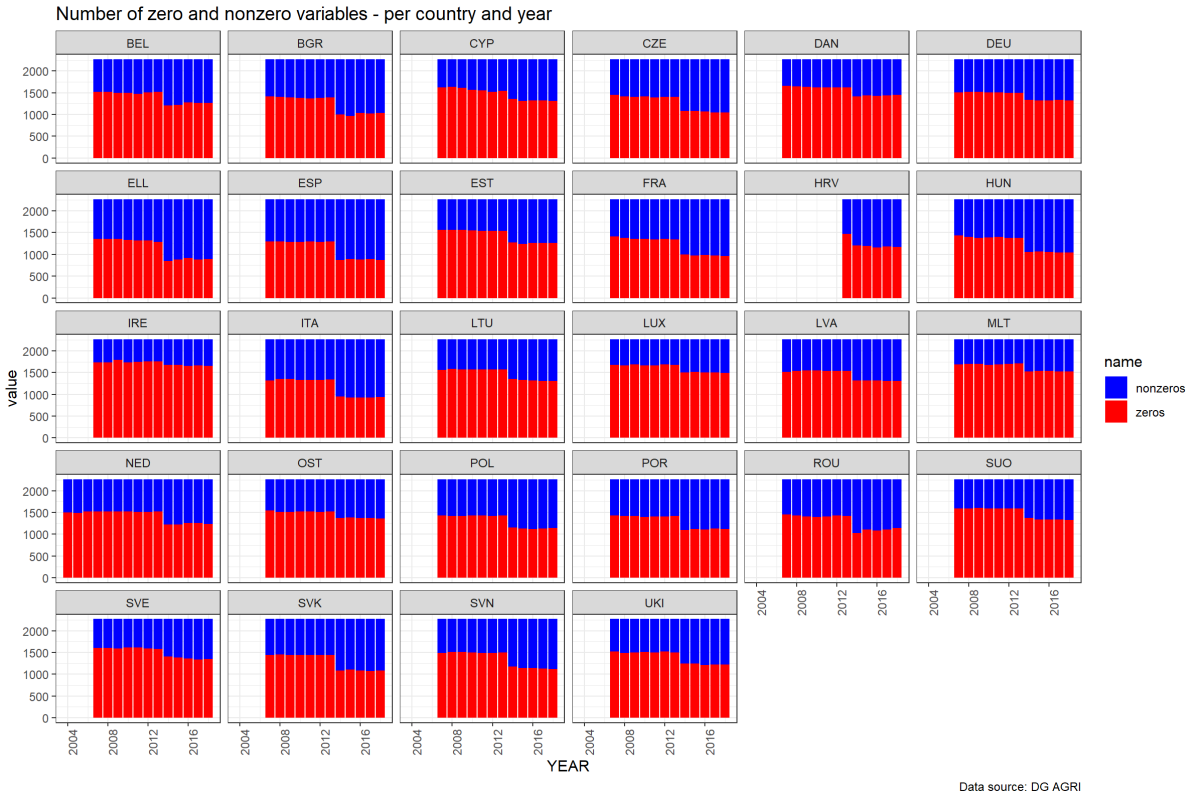



```
ggsave(plot = p,filename = paste0(CurrentProjectDirectory,"/vars_zero_country_years.png"),
    width = 12, height = 8)
```

4.2.4.13. Number of zero and nonzero variables - per year and country

```
## # A tibble: 333 x 4
## # Rowwise:  COUNTRY, YEAR
##    COUNTRY  YEAR zeros nonzeros
##    <chr>   <int> <int>    <int>
##  1 BEL     2007  1522     746
..

## 10 BEL     2016  1276     992
## # ... with 323 more rows
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.



4.2.4.14. Number of sample farms - Germany

```
files %>% filter(COUNTRY=="DEU") %>%
 select(COUNTRY,YEAR,ID) %>%
 count(YEAR)

##    YEAR    n
##  1: 2007 7692
..

## 12: 2018 8960

files %>% filter(COUNTRY=="DEU") %>%
 select(COUNTRY,YEAR,ID) %>%
 pivot_wider(names_from = YEAR,
             values_from = ID,
             values_fn = length)

## # A tibble: 1 x 13
##   COUNTRY `2007` `2008` `2009` `2010` `2011` `2012` `2013` `2014` `2015` `2016`
##   <chr>    <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>
## 1 DEU       7692   7841   8880   8999   8922   8957   8928   8823   8881   8821
## # ... with 2 more variables: `2017` <int>, `2018` <int>

## ID's for years 2007-2008 to 2009 - nx -> x times
files %>% select(COUNTRY,YEAR,ID) %>%
 group_by(COUNTRY) %>%
 filter(YEAR %in% c(2007:2009)) %>%
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
 count(ID) %>%
 summarise(n3=sum(n==3),
           n2=sum(n==2),
           n1=sum(n==1),
           n0=sum(!n %in% c(1:3) )) %>%
 print(n=27)
```

```
## # A tibble: 27 x 5
##    COUNTRY    n3    n2    n1    n0
##  * <chr>   <int> <int> <int> <int>
##  1 BEL     1030   165   139     0
..

## 27 UKI     2021   847   702     0
```

```
## Function to count exit, stay and entry
count_setdiff <- function(x,y){
 n1 <- length(setdiff(x,y))
 n2 <- length(setdiff(y,x))
 n3 <- length(intersect(x,y))
 return(c(n1,n3,n2))
}
```

4.2.4.15. Test for Belgium

Only testing if `count_setdiff()` works.

```
aa_ <- files %>%
 filter(COUNTRY=="BEL" & YEAR==2007) %>%
 pull(ID) %>% as.character()
bb_ <- files %>%
 filter(COUNTRY=="BEL" & YEAR==2008) %>%
 pull(ID) %>% as.character()
count_setdiff(aa_,bb_)
setdiff(aa_,bb_) %>% length()
```

4.2.4.16. Count exit, stay and entry for all countries

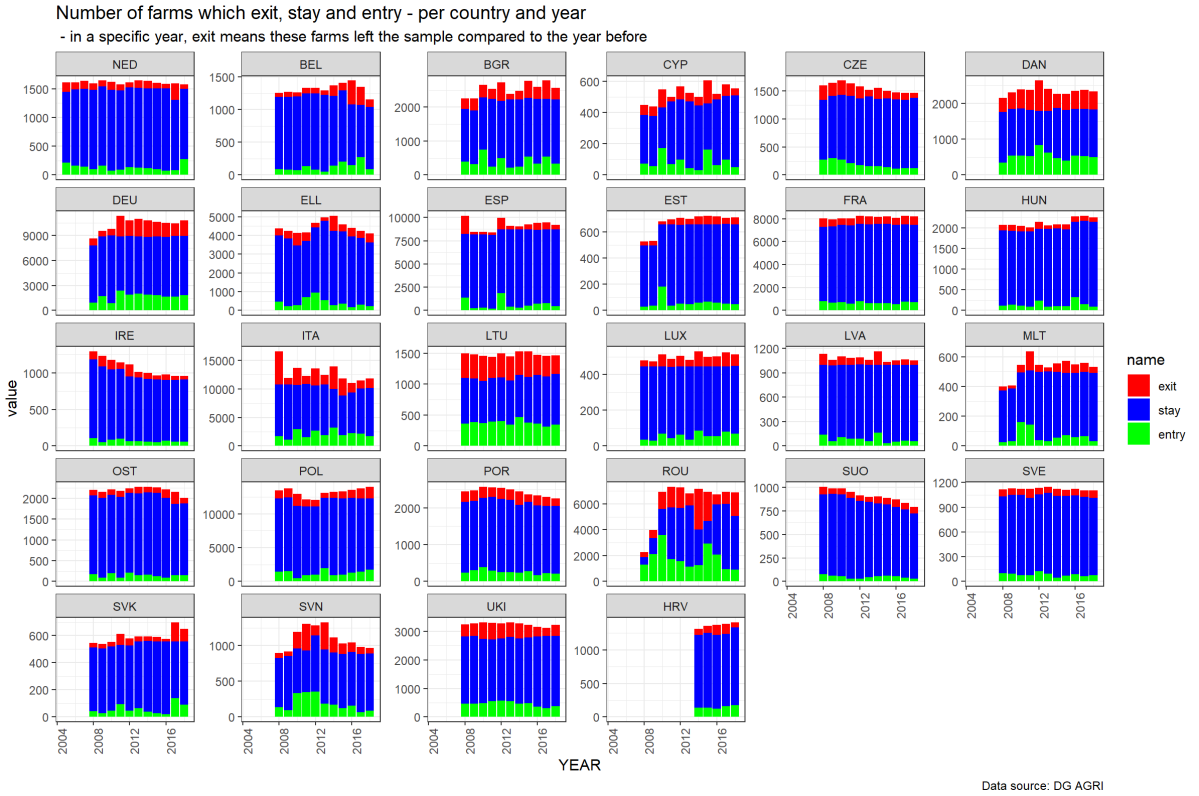
```
data_df <- setNames(
 data.frame(
  matrix(ncol = 5,
         nrow = 0)),
 c("COUNTRY","YEAR","exit","stay","entry"))
for(i in unique(files$COUNTRY)){
 years_selected <- files[COUNTRY==i,list(unique(YEAR))][[1]]

 for(j in years_selected[-length(years_selected)]){
  vec1 <- files[COUNTRY==i & YEAR==j,list(ID)][[1]]
  vec2 <- files[COUNTRY==i & YEAR==(j+1),list(ID)][[1]]
  data_df <- rbind(
   data_df,
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
    data.frame("COUNTRY"=i,"YEAR"=(j+1),
               "exit"=count_setdiff(vec1,vec2)[1],
               "stay"=count_setdiff(vec1,vec2)[2],
               "entry"=count_setdiff(vec1,vec2)[3]))

  }

}
transitions <- data_df
```

4.2.4.17. Count exit, stay and entry for all countries

```
p <- transitions %>%
  pivot_longer(-c(COUNTRY,YEAR)) %>%
  mutate(name=factor(name,levels = c("exit","stay","entry"))) %>%
  ggplot(data=.,aes(x=YEAR,y=value,fill=name)) +
  geom_col(position = "stack") +
  facet_wrap(~COUNTRY,scales="free_y") +
  scale_fill_manual(values = c("red","blue","green")) +
  labs(title = "Number of farms which exit, stay and entry - per country and year",
       subtitle = " - in a specific year, exit means these farms left the sample compared to the year befo
re",
       caption = "Data source: DG AGRI") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90))
p
```

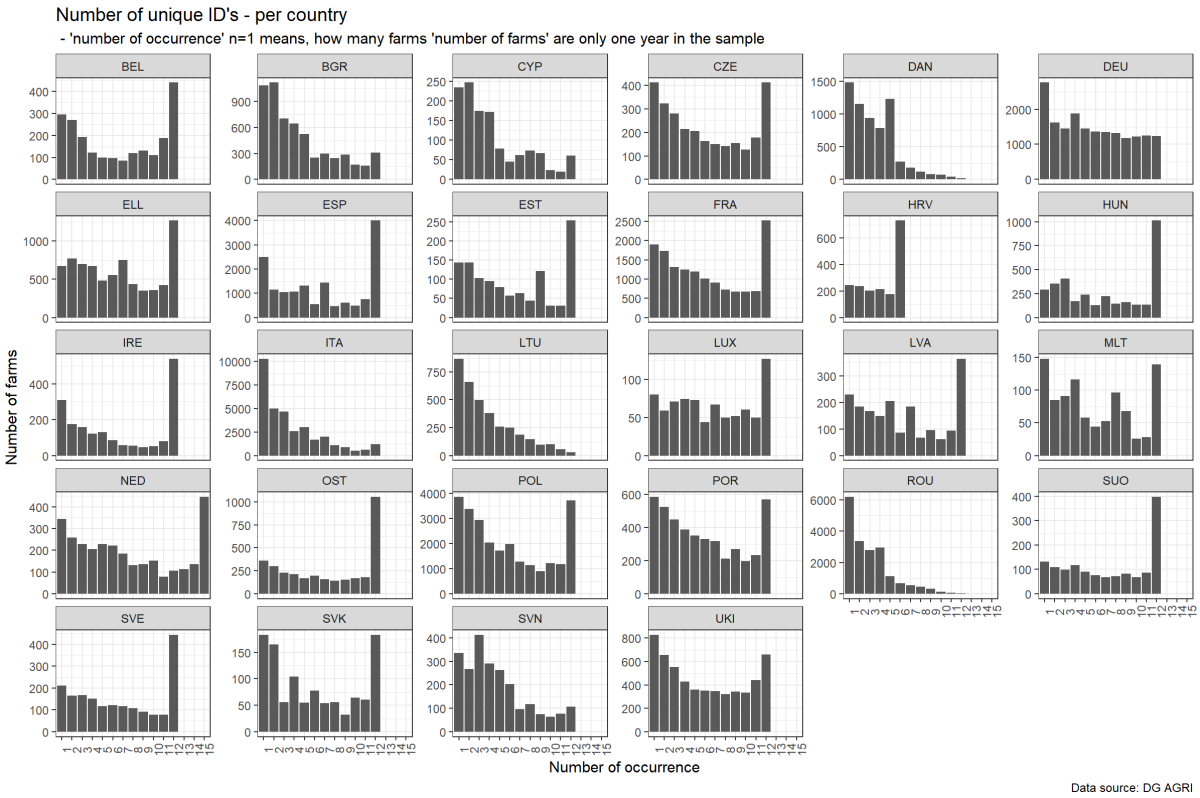


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
ggsave(plot = p,
       filename = paste0(CurrentProjectDirectory,"/Number_of_ID_entry-stay-exit_country_year.png
"),
       width = 12, height = 8)
```

4.2.4.18. Length of farm occurrence

```
unique_farms <- files %>%
  select(COUNTRY,YEAR,ID) %>%
  group_by(COUNTRY) %>%
  count(ID) %>%
  group_by(COUNTRY,n) %>%
  count()
p <- unique_farms %>%
  ggplot(aes(x=as.factor(n),y=nn)) +
  geom_col() +
  labs(title = "Number of unique ID's - per country",
       subtitle = " - 'number of occurrence' n=1 means, how many farms 'number of farms' are only on
e year in the sample",
       caption = "Data source: DG AGRI") +
  xlab("Number of occurrence") +
  ylab("Number of farms")+
  theme_bw() +
  facet_wrap(~COUNTRY,scales = "free_y") +
  theme(axis.text.x = element_text(angle = 90))
p
```



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
ggsave(plot = p,
    filename = paste0(CurrentProjectDirectory,"/Number_of_unique_ID_country.png"),
    width = 12, height = 8)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

4.3. Interface to the national FSS micro data

{THÜNEN, S. Neuenfeldt}

The goal of the FSS package is to provide some functions to analyse (German) Farm Structure Survey (FSS) data. As of the end of April 2021 there are two functions so far implemented in this package. One converts the (usually in csv format) provided raw data into a Rdata file, which is easier and faster to handle in R.

The second one is presented in this chapter and provides a fake data set to use and test some analysis before touching the real FSS data. In Germany, the FSS data must be requested by application and this is subject to specific conditions. Accessing the official data is possible on-site and off-site. On-site use, which is probably the most useful one, is possible via a safe centre or by remote execution of code. Both means, that it is almost always preferable to prepare code for specific analysis. Therefore, the researcher can build a fake data set and make sure, that the code has the proper syntax. Semantic analysis with this code is rather impossible, of course. This chapter presents a use case how to build a fake data set which has the correct form and is quite consistent in terms of number of observations. A more detailed description of the functions is presented in the Deliverable D2.4.

4.3.1. Installation

You can install the GitLab version of `FSS` from GitLab with:

```
# devtools::install_git("https://git-dmz.THUENEN.de/MIND STEP/fss") -->
```

4.3.2. Example

4.3.2.1. Load FSS package

```
library(FSS)
```

4.3.2.2. Test a function

Here we will test one of the provided functions: `generateFakeFSSData_DE()`. We generate some basic variables, which are automatically produced. We set an additional variable to be generated, namely C0300. The variable will be a continuous variable.

```
FSS_data_DE <- generateFakeFSSData_DE(C0codes = "C0300")

##
##  Build fake data
```

4.3.2.3. Add a specific categorical variable

Now we will add a specific variable, i.e. a categorical variable if a farm is an organic farm. This will be randomly distributed. The probability of a farm to be organic is 20%.

```
library(tidyverse)
FSS_data_DE <- FSS_data_DE %>%
  mutate(C0501=sample(c(0,1),
                      size = nrow(FSS_data_DE),
                      replace = TRUE,
                      prob = c(0.8,0.2)))
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

4.3.2.4. Some analysis

After that we analyse the data regarding the number of farms that are organic (C0501) for each year (C0008U1) and farm type (C0060UG1).

```
FSS_data_DE %>% group_by(C0008U1,C0060UG1,C0501) %>%
  count() %>%
  pivot_wider(names_from = C0060UG1, values_from = n) %>%
  knitr::kable()
```

C0008U1	C0501	1	2_3	45	46_48	5	6_7_8
1999	0	57491	16693	29578	44526	33620	55483
2020	1	11938	3420	6248	9284	7030	11512

4.3.3. Conclusion

This could be the final output of an overview of conventional (C0501=0) and organic (C0501=1) farms for each year and the provided farm types. As this is a fake data set, these numbers make no sense.

Nevertheless, it is a process that must be made each time FSS data wants to be analysed. The researcher must write a R program beforehand and check its syntax thoroughly. One the one hand, the researcher should be prepared when visiting the RDC for working with the data. On the other hand, if remote execution of code is preferred or necessary it is even more important to have running program. There will be a lot of communication with the RDC to get the program running, notwithstanding that you have no clue about what is really behind your produced features, numbers, tables or figures as you have no chance to look into the real data. Data protection does not permit it. Working with proper fake data beforehand is without alternative.

4.4. Interfaces for Management data

{WR, J. Helming}

Highly detailed farm models like FARMDYN (Britz et al. 2014) or FARMBOSS (Muench, 2002) depict compared to the more aggregated models in rich detail the inventory of machines, land, stables and other structures, demographic relations between different herds, labour and feed requirements and nutrient flows as well as the financial dimension. In addition, such models have high temporal resolution, partially at the weekly or day basis, to account for management options. The models also cover investment in assets, whether machinery, land, stables or bio-gas facilities. The parameterisation of the different technologies and investments is data demanding and require the knowledge of management data, technical coefficients usually collected by different players in the field and are not available in statistics such as FADN or FSS. This section aims to provide a summary of information on accessibility of management data and their respective access policies. Availability of the management handbooks in the different case study regions in MIND STEP (Germany, the Netherlands, France and Italy) and the possibility to use it are summarised in table 4. Various types of management handbook data that can be found in the case study regions are only available on paper and/or in the own language.

Table 5: Summary table for actors and access to management handbook data to parametrize IDM models

Country	Body	Scope of data	Access conditions	Description	Webpage or database link

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

DE	Board of Technology and Construction in Agriculture e.V. KTBL	As a non-profit association, we promote future-oriented agriculture by advising all stakeholders and providing facts and figures.	Part of the data are online accessible but in exceptions also as bulk data base for partners based on contract conditions	operations investment machinery costs (MSKost) biogas	https://www.ktbl.de/webanwendungen https://daten.ktbl.de/makost
	German Agricultural So ciety (DLG)	DLG is an open network and the professional voice of agriculture, agribusiness and the food sector.	Online data base	Content of nutrient for feed and crop product	https://datenbank.futtermittel.net
	Agency for Renewable Resources (FNR)	FNR is the central coordinating institution for funding research, development and demonstration projects in the field of renewable resources used to produce bio-based products and/or bioenergy.	Print media	renewable resources used to produce bio-based products and/or bioenergy	https://international.fnr.de https://mediathek.fnr.de/leitfaden-biogas.html
	ALB	Working Group for Rationalization, Agricultural Engineering and Construction in Hesse e.V. (ALB-Hesse)	Print media	Guide prices for new construction and reconstruction of agricultural farm buildings and rural dwellings	http://www.alb-hessen.de/veroeffentlichungen/richtpreise.html
	Profi	With profi, readers interested in agricultural technology receive thorough and practical information month after month; Landwirtschaftsverlag GmbH	Web formulae	agricultural machinery catalogue, investment costs	https://www.profi.de/management/landmaschinen-katalog
	Land data		Excel download	LAND-DATA's agricultural industry comparison contains important key figures on profitability, stability and liquidity as well as earnings and expense ratios:	https://www.landdata.de/beispielauswertungen
NL	KWIN-AGV 2018	Arable and vegetable crops	Print media (paid)	Detaild gross margin and labour requirement calculations for a large number of conventional and organic arable and open-air vegetable crops. As well as fixed costs for land, buildings, tractors, machinery, labour and hired labour.	https://www.wur.nl/nl/show/kwin-agv.htm
	Branche Organisation Arable and ministry of Agriculture	The website 'Handbook Soil and Fertilisation' offers advice and possibilities regarding fertilisation and soil management for producers and advisers.	Print media	Operations (e.g fertilisation, cultivation, organic matter and water management, rotation), Arable crops (legal fertilisation levels) and Soil (quality, fertility, health, structure and water availability)	https://www.handboekbodemenbemesting.nl/nl/handboekbodemenbemesting.htm
	Wageningen Livestock Research	Wageningen Livestock Research offers a wide range of products and services, such as handbooks with practical information, useful software that allows you to make calculations and current prices, for example of animal feed.	Print media (e.g. KWIN Livestock), free and paid and tools and software products (Excretion Information (BEX), Feed Print, Dairy Wise, etc.).	WLR translates scientific knowledge into practical solutions in livestock farming systems, nutrition, fertilisation and crop growth (limited to roughage and feed crops), genetics, animal welfare, and the impact of domestic animals on the environment.	https://www.wur.nl/nl/Onderzoek-Resultaten/Onderzoeksinstituten/livestock-research/Producten.htm

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

	Different partners among which Akkerweb, Wageningen University and Research	There is a wealth of data associated with your farm, information that is usually registered and recorded by various trading and other partners. However, these records do not help you move your farm or crops forward. We looked for a solution to this.	Free apps and commercial applications	Each field carries an enormous amount of information that growers could use for a tailor-made approach of thelr management. Unlocking field information via Akkerweb and combining this information with knowledge of crop management, enables the farmer to increase profit via targeted application of irrigation, crop protection and soil tillage.	www.akkerweb.eu
FR	Chambres d'Agriculture	"*Barème d'entraide*" ("mutual aid scale") is a document distributed each year by the French *Chambres d'Agriculture* (there is usually one *Chambre d'Agriculture* per region). There also exists one document entitled "*Coûts des opérations culturales des Matériels Agricoles*" (Costs of farming operations for agricultural machinery) synthetizing the information at the national scale.	*Barèmes d'entraide* at regional scale are available upon request from the corresponding *Chambres d'Agriculture* and are also made available for free online for some regions. The document synthetizing the information at the national scale is available for free online.	Set of tables allowing to calculate the costs of cultivation operations according to the agricultural equipment used.	Document at the national scale: https://chambres-agriculture.fr/publications/toutes-les-publications/la-publication-en-detail/actualites/couts-des-operations-culturales-2020-des-materiels-agricoles/
IT	Disciplinare di produzione integrata Emilia Romagna	Legislative documents to provide indications on crop management to be respected	Online, public access	Detailed information on crop nutrient and water requirements and on the use of plant protection products.	https://agricoltura.regione.emilia-romagna.it/produzioni-agroalimentari/temi/bio-agro-climambiente/agricoltura-integrata/disciplinari-produzione-integrata-vegetale/Collezione-dpi/dpi_2020/disciplinari-2020
	Centro ricerche produzioni animali (2001). Liquami zootecnici, Manuale per l'utilizzazione agronomica, Ed. L'Informatore Agrario	Livestock management agricultural handbook	Print media (paid)	Information on manure/slurry management in livestock farms	https://www.libreriauniversitaria.it/liquami-zootecnici-manuale-utilizzazione-agronomica/libro/9788872201428

Source: Own compilation.

Within MIND STEP management handbooks are used in the ABM Agripolis and the bio-economic farm model FARMDYN. A crucial set of data for FARMDYN are the machinery requirements for field operations and the associated labour requirements and costs. This information is not available in FADN, at least not in the needed detail. As shown in Figure 10 FARMDYN permits the selection of data-files related to crop operations (cropop_XX.gms). In this GAMS file, data can be loaded or stored, and re-shaped to be accommodated in the subsequent model code. For the Netherlands, data on field

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

operations was collected from the KWIN database and stored in gdx-format and the file cropop_NL was created for inclusion in FARMDYN. The workflow is relatively simple as field operation data are assumed to be equal for all farms, so no aggregation or sub-setting is required.

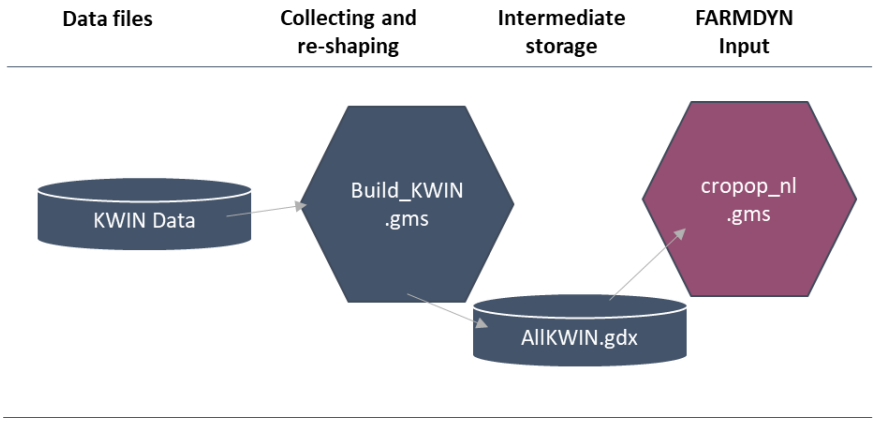


Figure 10: KWIN Data Processing

Source: Own compilation.

Typical data-parameters included in AllKWIN.gdx are for instance operation attributes in the GAMS parameter "opp_attr" and machinery attributes as prices, lifetime and costs of maintenance and insurance in the GAMS parameter p_machAttr. Figure 4 shows the elements of the GAMS parameter "opp_attr".

Cultivateren	labTime	1.05714285714286
	nPers	1
diepwoelen	labTime	2.1
	nPers	1
frezen	labTime	2.78333333333333
	nPers	1
ploegen	labTime	1.3
	nPers	1
spitten	labTime	2.8
	nPers	1
stoppeleggen	labTime	0.566666666666667
	nPers	1

Figure 11: Elements of opp_attr. Operations in first column (in Dutch for Dutch version of FARMDYN), Second column includes: nPers, number of persons (heads per ha) and labTime in hours per ha.

Source: Own compilation from KWIN.

Table 5 also refers to akkerweb (www.akkerweb.eu). This is an open service platform that can be used for different precision farming applications. It provides maps, services, data, decision support and connections. It provides background maps, services for weather data, satellite images, soil maps, but also a task map generator and crop growth models. The applications that are available are focused tools for crop protection such as nematode and weed management, *Phytophthora*, but also on nutrient management and grass production. Akkerweb also includes tools for *soil borne parasitic nematodes. These tools* can be used to find the most optimal order of crops in the rotation, so that a crop that is sensitive to damage will not be grown after a crop that generates a high multiplication of the nematodes. Further, an estimate is given of the effect of nematode (both models) or fungal (Best4Soil) species, when present in high densities, on crop growth. These types of information can e.g., be used as input for FARMDYN.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

4.5. AgroDataCube

{WR, M. Müller, S. Janssen}

4.5.1. Use case

The single farm model FARMDYN (Britz et al., 2014) allows simulating optimal farm management and investment decision under changes in boundary conditions such as prices, technology, or policy instruments, for a wide range of farming branches (arable, pig fattening, sows, dairy, beef cattle including suckler cow-calf systems, biogas). It is based on a model template for a fully dynamic or comparative-static bio-economic simulation building on Mixed-Integer Programming (MIP). The fully dynamic version can be extended to a stochastic dynamic model, combined with different risk behavioural models. Farm branches and other elements such as e.g., fertilization and manure policy restriction can be added in a modular fashion to the core model, as well as a module for large-scale sensitivity analysis. The model's default data and parameterization comprise detailed engineering data for Germany which cover field operations, a crop calendar for over hundred crops, detailed by tillage system, conventional versus organic farming and by plot size and farm-plot distance.

Within the MIND STEP project, FARMDYN has been adopted for the Netherlands by using the Dutch version of FADN (Bedrijfsinformatienet, BIN) for general farm characteristics like endowments, yields, and cost structure. Further sources of information are the "Kwantitatieve Informatie voor de Akkerbouw" (KWIN) for management data as discussed in the previous chapter and the Dutch agricultural census (Landbouwtelling, LBT). BIN is an extensive database for farm statistics that is maintained at Wageningen Economic Research. It includes a panel of around 1.500 agricultural and horticultural enterprises. The database can be accessed either through a dedicated database management system (ARTIS) or meanwhile through a data warehouse system.

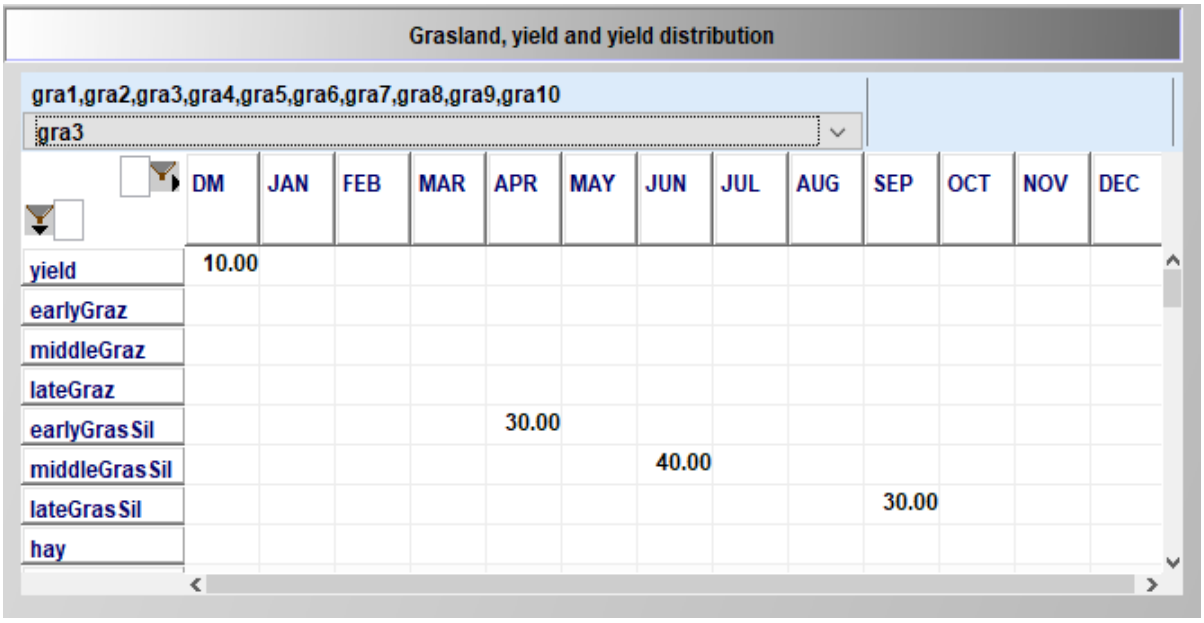


Figure 12: Grassland management representation in FARMDYN for grass silage with 3 mowing events

Source: FARMDYN GUI, Britz et al., 2014.

While data about crop production on arable land is readily available from BIN, detailed data on grassland activities is more difficult to obtain. FARMDYN permits the simulation of a wide range of

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

grassland management options, differentiated by types of harvesting (grazing, mowing), by dry-matter and nutrient yields, and timing of field operations. Since grassland is an important, sometimes the only, source of on-farm feed and due to the high potential for generating not only nutrients for animals, but also positive environmental outcomes, an empirical grounding of the grassland representation in FARMDYN is desirable. The model's default representation of grassland data is depicted in Figure 12: Each management option is described by their cumulative annual dry-matter yield (top-left corner in Figure 12) and the distribution of seven grassland outputs (3 types of grazing and silage, hay) throughout the year. This level of detail is very difficult to obtain for a large sample of farms and management options.

A solution to compensate for this lack of information is the Dutch AgroDataCube (ADC) database, which provides a large collection of both open data and derived data for use in agri-food applications. The data from the groenmonitor (Figure 2), which includes the timing of mowing events and indications of annual dry-matter yields. This permit deriving crucial parts of the grassland management tables required by FARMDYN. The following sections describe the structure of the used data and the processing steps required to align ADC and BIN data to populate the grassland management attributes for the FARMDYN model.

4.5.2. Data bases and workflow

4.5.2.1. Extraction of grassland data from ADC

The ADC the following relevant data at parcel level:

- NDVI time series
- Annual cumulative NDVI and cumulative grass length
- Number of grass mowing cuts
- Mowing dates

All data stored in the ADC is anonymized. However, using the parcel level data as input for the FARMDYN model requires associating the IDs of the parcels with the corresponding farm ID, if possible, or at least with another meaningful classification (e.g. agro-ecological zones), to which the farm-level data can also be linked. For this case study we used a sample of farms across the Netherlands. The extraction of the relevant parcel data from the ADC was done manually with the following processing steps:

1. Provision of the officially registered farm ID's to Wageningen Environmental Research
2. Request for authorized use of the personalized LPIS parcels information (BRP crop parcel dataset with username and farm ID) to the Dutch authorities (RVO)
3. Selection of the relevant field ID's with corresponding farm ID's in the personalized LPIS parcels information system. In total 10000 field ID's were queried.
4. Extraction of the relevant ADC data (see above) for the selected 10000 field ID's
5. Creation of one dataset with Farm ID, Field ID and ADC data

For the GIS coupling of the datasets in this case the ArcGis software was used. Once established as a practical workflow, the process can be fully automated in the future, provided that the request for authorized use of the personalized LPIS parcels information is granted by either the authorities or on an individual basis by the involved farmers.

The information available from ADC is summarized in Figure 13. The most important variables are those related to the number of mowing cuts (no_mowing_cuts), the range of days during which the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

mowing took place (m1-m6, q1-q6), and the total cumulative height (cum_height) as indicator for total dry-matter yield. This permits an immediate linkage to the FARMDYN tables in Figure 12.

ADC Data	Data type / unit	Description
Crop_type	Factor	
Farm_ID	ID	
Area_m2	m2	Plot area
Field_ID	ID	
Extraction	date	Date of the satellite image
no_sat_img	n	Number of NVDI satellite observations per year
no_25m_pix	n	Number of 25 m resolution pixels that encompass the field
no_mowing_cuts	n	Total number of mowing cuts
cum_NDVI	index	cumulative NDVI
avg_NDVI	index	average NDVI
max_NDVI	index	max. NDVI
min_NDVI	index	min NDVI
cum_height	cm	Cumulative height of the grass
m1..6	day_of_year	Day of the year .. mowing cut
q1..6	days_since_last_ob	Number of days between the image where ploughing occurred and the previous image

Figure 13: AgroDataCube: Grassland data indicators

Source: ADC, own summary compilation.

4.5.2.2. Extraction of grassland data from BIN

Like the case of ADC, the primary extraction of the required data from the BIN database was conducted by a domain expert. This is largely due to the size and complexity of the database, and it proved to be more practical to make use of established workflows rather than developing own automated routines, particularly in the present case which concerns the development of a prototype. The data extracted from the BIN database was grassland area and dry matter yields at farm level. It must be noted here that the farm IDs used in BIN are different from those to which the parcel-level information was provided. Combining BIN and ADC thus also requires a correspondence between the farm IDs used in BIN and ADC.

4.5.2.3. Combining the data sources

The extraction of datasets from BIN and ADC by the respective domain experts resulted essentially in 3 data tables as shown in Figure 14. Processing ADC and BIN data requires to drop several observations that do not provide useful information for the present task. Examples are very small parcels or parcels for which no yield is reported. Also, the ADC data includes data on crops acreages and areas devoted to trees, which are generally important but not needed for the derivation of grassland data. After the data table are sufficiently cleaned, the parcel level information is aggregated to the farm-level by adding up areas, distinguished by the number of mowing cuts. Average yields (cumulative heights) are also aggregated to farm level. An important step is also the re-scaling of grassland yields measured in cumulative height, as reported in ADC, to match the information in BIN, which is given in tons of dry matter. The prepared data tables can now be merged by using the correspondence table ("C" in Figure 14). The described steps (sub-setting, aggregation, and merging) are implemented as a procedure implemented in R.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

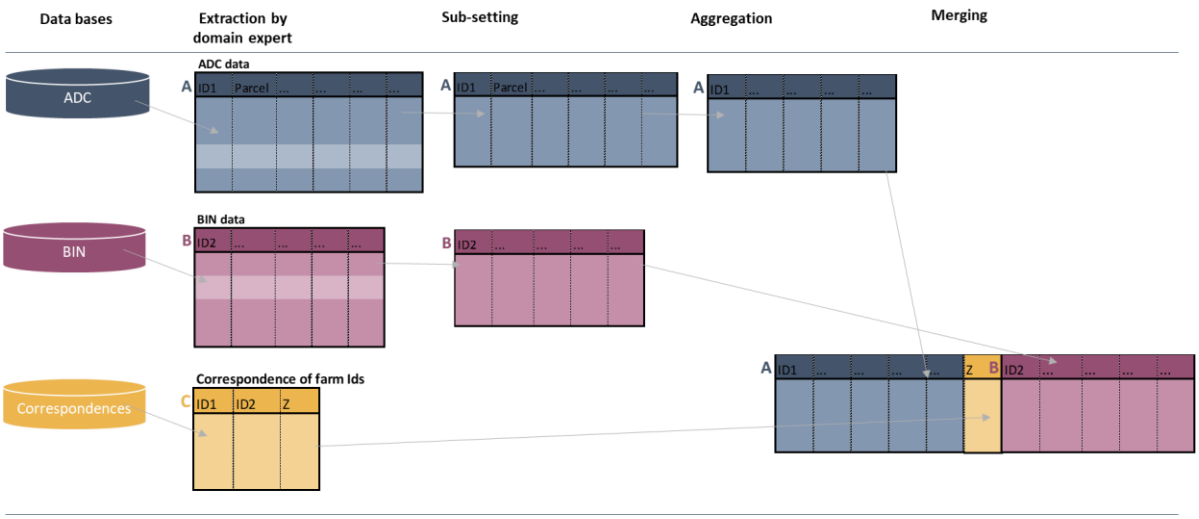


Figure 14: Combination of data sources

Source: Own compilation.

4.5.2.4. Consistency checks

To ensure that the combination of satellite- and survey-data at farm level are consistent, the aggregated farm-level grassland areas were compared (Figure 15). Despite some deviations originating from definition and measurement of the relevant variables, the datasets appear to be consistent in the sample cases. A further test was to check the average farm-level yields depending on the number of cuts. In general, the more intensively the grassland is managed, the higher the yields should be. This is nicely reflected in the ADC data and depicted in Figure 16.

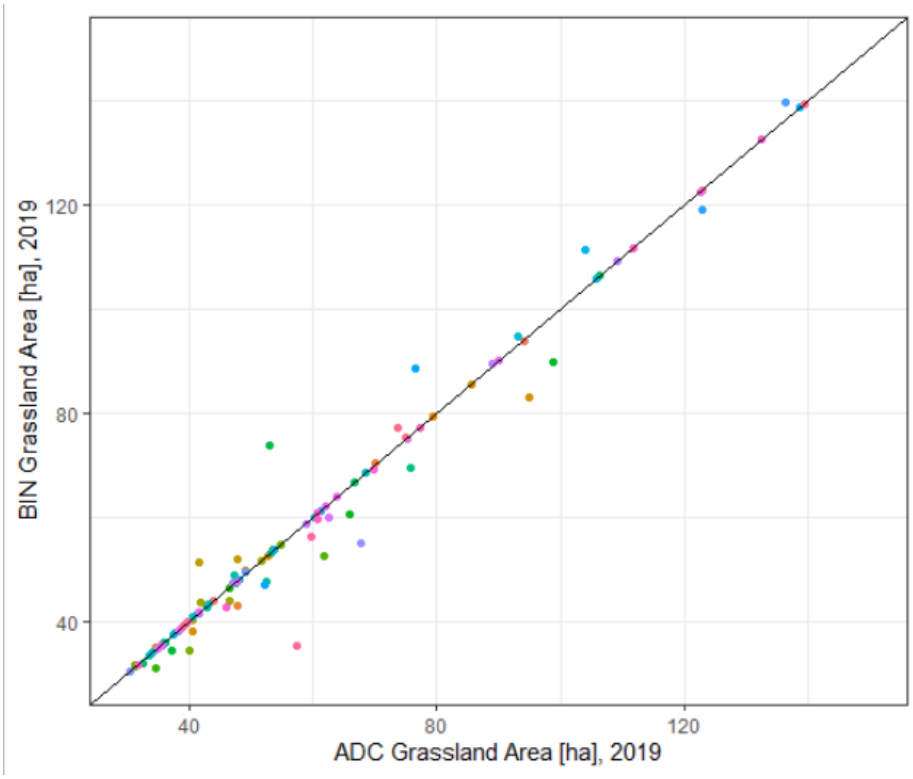


Figure 15: Grassland Area from ADC and BIN (2019, 180 dairy farms)

Source: ADC and BIN, own compilation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Most importantly, the ADC grassland dataset permits linking the number of mowing events with the total dry-matter yield of grassland, thus providing crucial information for the construction of the FARMDYN table in Figure 12: First, the number of cuts gives an estimate on the management system put in place by the farmer. One or no cutting event at all indicate that a plot was planned for grazing only. Three or more cuts indicate that a farmer was producing grass silage. It might be possible to identify mixed grazing/silage systems based on these data using additional information.

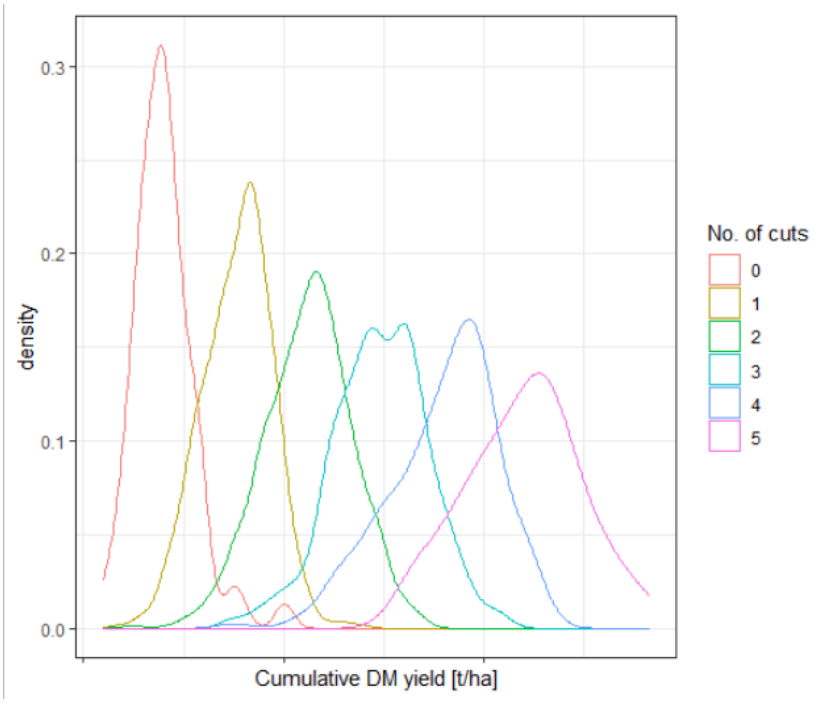


Figure 16: Grassland Yield Densities by Number of Cuts

Source: ADC and BIN, own compilation.

4.5.1. R-Code

4.5.1.1. Load required libraries

```
library(ggplot2)
library(data.table)
library(readxl)
```

4.5.1.2. Specify data and working directories - this is user specific!

```
ADCDir      = "C:/FARMDYNDATA/ADC"
BINDir      = "C:/FARMDYNDATA/BIN"
LMMDir      = "W:\\WR\\PIA\\PW\\DZK\\GHG2020\\output Jakob"
MyLMMDir      = "C:/FARMDYNDATA/LMM"
MyBINDir      = "C:/FARMDYNDATA/MYBIN"
MyDRAMDir      = "C:/FARMDYNDATA/DRAM"

#setwd(ADCDir)
```

4.5.1.3. Load data - file names are user specific!

```
# ADC data
ADCFile <- "brp2019_182farms_mowing_regime.xlsx"
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
dat_brp2019 <- as.data.table(read_xlsx(paste(ADCDir,ADCFile, sep="/"), she
et = "brp2019_182_farms_maaisnedes",range = "a1:AA4545", col_names = TRU
E))
ADC_COLS <- as.data.table(colnames(dat_brp2019))

# BIN data
lmm <- fread(paste(MyLMMDir,"lmm_06_19.csv", sep="/"))
LMM_06_19_COLS <- as.data.table(colnames(lmm))

# Correspondence tables
LBT2BIN <- read_xlsx(paste(MyBINDir,"parcelExchange.xlsx", sep="/"), sheet
 = "LBT2BIN",range = "X3:Y22016", col_names = TRUE)
DAY2MNTH <- read_xlsx(paste(MyBINDir,"parcelExchange.xlsx", sep="/"), shee
t = "DAY2MNTH",range = "G2:I14", col_names = TRUE)
```

4.5.1.4. Process BIN data

```
# Specify grassland data columns
lmmGrassLandCols <- c("BIN-nummer","A24 NSO type","B02 hoofdgrondsoort","Y
ear",
                      "B07 oppervlakte cultuurgrond grasland [ha]",
                      "L15 opbrengst grasland in kg ds per ha",
                      "L18 opbrengst grasland in kVEM per ha",
                      "L11 opbrengst (snij)mais in kg ds per ha",
                      "L14 opbrengst (snij)mais in kVEM per ha")
# Define shorter names for columns (also without blanks)
lmmGrassLandColsNew <- c("BIN_ID","NSOType","mainSoilType","Year","grasLan
dArea_ha","grasLandYield_kg","grasLandYield_kVEM","snijMaisYield_kg","snij
MaisYield_kVEM")

# Define numeric columns
lmmGrassLandColsNewNum <- c("Year","grasLandArea_ha","grasLandYield_kg","g
rasLandYield_kVEM","snijMaisYield_kg","snijMaisYield_kVEM")

# Extract only grassland related data
lmmGrassLand <- lmm[,lmmGrassLandCols,with=FALSE]

# replace long column names with shorter ones
setnames(lmmGrassLand, old = lmmGrassLandCols, new = lmmGrassLandColsNew,
skip_absent=TRUE)

# convert to numeric
lmmGrassLand[, (lmmGrassLandColsNewNum) := lapply(.SD, as.numeric), .SDcol
s = lmmGrassLandColsNewNum]

# subset non-zero values
lmmGrassLand <- lmmGrassLand[grasLandArea_ha>0 & grasLandYield_kg>0,lmmGra
ssLandColsNew,with=FALSE]

# Add ADC farm IDs
lmmGrassLand <- merge(lmmGrassLand, LBT2BIN, by.x="BIN_ID", by.y="BIN_ID")
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

4.5.1.5. Prepare ADC data

```
# Unique set elements
tFarm_ID <- (unique(dat_brp2019$Farm_ID))
tCrop_type <- (unique(dat_brp2019$Crop_type))

# ADC data to be used for grassland
tGrassLandType <- c("Grasland, blijvend","Grasland, natuurlijk. Hoofdfunct
ie landbouw.","Grasland, tijdelijk")

# Ensure farm IDs and number of mowing cuts are factors
dat_brp2019$LBT_ID = as.factor(dat_brp2019$Farm_ID)
dat_brp2019$fact_no_mowing_cuts = as.factor(dat_brp2019$no_mowing_cuts)


# Add an approximation for DM yields
# References:
#     https://www.lfl.bayern.de/mam/cms07/publikationen/daten/informatione
n/p_22478.pdf
#     https://www.teagasc.ie/media/website/crops/grassland/How-to-measure-
grass-right.pdf

dat_brp2019$cum_DMY = dat_brp2019$cum_Height*250/1000
```

4.5.1.6. Test consistency between BIN and ADC

```
# test data table from ADC
tst_brp2019=subset(dat_brp2019, Crop_type %in% tGrassLandType) [, list(
                                brpGrasLandArea_ha = sum(Area_m2/10000, n
a.rm = T),
                                brpCumDryProd_kg = sum(cum_DMY*Area_m2/100
00, na.rm = T)
                               ),
                           by=LBT_ID ]
tst_brp2019$brpCumDryYield_kg2ha = tst_brp2019$brpCumDryProd_kg / tst_brp2
019$brpGrasLandArea_ha

# Target data table from BIN
lmmGrassLand2brp <- subset(lmmGrassLand, LBT_ID %in% tFarm_ID & Year %in%
c(2017,2018,2019) )

# merge with ADC data
lmmGrassLand2brp <- merge(lmmGrassLand2brp, tst_brp2019, by.x="LBT_ID", b
y.y="LBT_ID")

# visualize area data
ggplot(subset(lmmGrassLand2brp, Year %in% c(2019) ) )  + geom_point(aes(x=
brpGrasLandArea_ha,y=grasLandArea_ha, color=LBT_ID)) +
  geom_abline(slope=1,intercept=0)+
  labs(x = "ADC Grassland Area [ha], 2019", y="BIN Grassland Area [ha], 20
19") +
  theme_bw()+theme(legend.position="none") + xlim(30, 150) + ylim(30, 150)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

4.5.1.7. Aggregate ADC data to farm level

```
stat_brp2019=subset(dat_brp2019, Crop_type %in% tGrassLandType) [, list(
  brpGrasLandArea_ha = sum(Area_m2/10000, na.rm = T),
  brpCumDryProd_kg = sum(cum_DMY*Area_m2/10000, na.rm = T),
  brpAvrDayM1 = mean(m1, na.rm = T),
  brpAvrDayM2 = mean(m2, na.rm = T),
  brpAvrDayM3 = mean(m3, na.rm = T),
  brpAvrDayM4 = mean(m4, na.rm = T),
  brpAvrDayQ1 = mean(m1-q1, na.rm = T),
  brpAvrDayQ2 = mean(m2-q2, na.rm = T),
  brpAvrDayQ3 = mean(m3-q3, na.rm = T),
  brpAvrDayQ4 = mean(m4-q4, na.rm = T)
), by=c("LBT_ID","fact_no_mowing_cuts")]
stat_brp2019$brpCumDryYield_kg2ha = stat_brp2019$brpCumDryProd_kg / stat_b
rp2019$brpGrasLandArea_ha
# str(stat_brp2019)

stat_brp2019[, brpGrasLandAreaTot_ha := sum(brpGrasLandArea_ha, na.rm=T),
by=c("LBT_ID")]
stat_brp2019[, brpGrasLandAreaTot_shr := brpGrasLandArea_ha/sum(brpGrasLan
dArea_ha, na.rm=T), by=c("LBT_ID")]
stat_brp2019[, brpCumDryProdTot_kg := sum(brpCumDryProd_kg, na.rm=T), by=c
("LBT_ID")]
stat_brp2019[, brpCumDryYieldAvr_kg2ha := brpCumDryProdTot_kg / brpGrasLan
dAreaTot_ha]

# Append BIN farm IDs
stat_brp2019 <- merge(stat_brp2019, LBT2BIN, by.x="LBT_ID", by.y="LBT_ID")
```

4.5.1.8. Export to file

```
fwrite(stat_brp2019, paste(ADCDir,"stat_ADCbyFarmandNrCuts.csv", sep="/")
 )
```

4.5.2. Conclusion

The use of satellite images from the ADC database has proven to be very useful for deriving grassland-related parameters for the FARMDYN model because it permits to distribute average farm-level yields to several management systems, distinguished by the number of cuts. Unfortunately, it is not possible at this stage to distinguish the grass yields per mowing event, only the average annual yields. For the integration in FARMDYN, additional information has to be used, e.g. the FARMDYN default shares. While they appear to be valid approximations, they are not farm-specific, which would be a valuable improvement of the FARMDYN database. Also, a distinction of types of grass according to the vegetation stage (here: early, middle, and late) is not directly possible. Using the differences between mowing events is currently being tested as an approximation.

4.6. Macro data from established models Like GLOBIOM

{IIASA, Brouwer}

About half the models involved in MIND STEP use GAMS as the core language. Over half of the models use R as an additional scripting language or are even based on R. This—together with the capabilities offered by R and its package ecosystem—makes R the natural choice for implementing code that

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

transforms, manipulates, and analyses data derived from models and also for code that glues models together.

Here we give a hands-on review of extracting GAMS data and making it available for processing in R, using GLOBIOM data as an example. Note that the discussed R processing methods are not unique to GAMS and can serve any model-derived data that can be represented as a data frame.

4.6.1. GDX data

GAMS Data eXchange (GDX) is a GAMS-native file format hat stores the values of one or more GAMS symbols such as sets, parameters variables and equations. GDX files can be used to prepare data for a GAMS model, present results of a GAMS model, store results of the same model using different parameters, and so on. GDX files are binary files that are portable between different platforms.

Libraries to access GDX file ship with GAMS and can be used from other languages. Hence, accessing GDX files requires a GAMS installation, or at least the GDX-pertinent subset of dynamically-loaded libraries that ship with GAMS. These do not require a GAMS license.

4.6.2. The gdxrrw R package

The **gdxrrw** R package (https://github.com/GAMS-dev/gdxrrw) allows exchange of data between GAMS and R via GDX files. Hosting of gdxrrw was recently moved to GitHub. On the GitHub **gdxrrw** wiki (https://github.com/GAMS-dev/gdxrrw/wiki) you can determine what package version will work with what R version. As yet undocumented at the time of this writing is that as of **gdxrrw** V1.0.8, libraries from GAMS >= V33 are required on account of the use of a new more memory-efficient GDX API.

After installing **gdxrrw**, the package can be loaded and pointed to a GAMS installation directory from where to load the GDX libraries:

```
library(gdxrrw)
igdx("C:/GAMS/33")

## GDX API loaded from gamsSysDir=C:/GAMS/33
## The GDX library has been loaded
## GDX library load path: C:/GAMS/33
```

To browse the package documentation and learn about its functions, issue:

```
??gdxrrw

## starting httpd help server ... done
```

4.6.3. Exploring GDX data

Examine the content of a GLOBIOM scenario output file:

```
gdx_file <- "output.gdx"
info <- gdxInfo(gdx_file,
                dump=FALSE,
                returnList=FALSE,
                returnDF=TRUE)
info$sets

##                  name index dim card           text doms     domnames
## 1              VAR_ID     5   1   59                  0            *
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
## 2           VAR_UNITs     6   1   39 Variable units    0           *
## 3           REGION_AG     7   1   11                   0           *
## 4       REGION_AG_MAP     8   2   74                0, 0 REGION_A....
## 5            ITEM_AG     9   1   67                   0           *
## 6        ITEM_AG_MAP    10   2  195                0, 0   ITEM_AG, *
## 7          MacroScen    11   1    5                   0 AllMacroScen
## 8           IEA_SCEN    12   1   14                   0  ALLIEA_SCEN
## 9          BioenScen    13   1    4                   0 ALLBioenScen
## 10          ScenYear    14   1   11                   0  AllScenYear
## 11 SYSTEM_ATRIB_VAL_4    19   1    1                   0           *

info$parameters

##                                name index dim    card
## 1                           OUTPUT_AG     1   8  241236
..

## 28           Water_reg_SW_GW_Compare    39   6    4136
##                                          text         doms     domnames
## 1       Aggregated output for item and region 0, 0, 0,.... VAR_ID, ....
..

## 28                                          0, 0, 0,.... ANYREGIO....
```

4.6.4. The Tidyverse

The Tidyverse (https://www.tidyverse.org/) is a coherent set of well-designed packages that together provide a more modern and consistent way of manipulating, analyzing, and visualizing data. It is a major innovation in the R ecosystem. To allow the packages to work together, they all accept so-called tidy data (https://tidyr.tidyverse.org/articles/tidy-data.html). Tidy data is tabular. R data frames are the classic data type for holding tabular data. Tibbles (https://tibble.tidyverse.org/) as used by the Tidyverse are modernized data frames:

- fewer surprising behaviors
- stricter
- enhanced print()

The ease of composing data processing steps from different packages in the tidyverse can also ease the interfacing of MIND STEP models provided that the model data exchange plays by Tidyverse rules: use Tidy data represented as tibbles and construct higher-level functionality as much as possible from functionality in Tidyverse packages.

4.6.5. Tidy GDX data

Parameters can be read from GDX files as dataframes:

```
data <- rgdx.param(gdx_file,
                   "OUTPUT_AG",
                   names=NULL,
                   compress=TRUE,
                   ts=FALSE,
                   squeeze=TRUE,
                   useDomInfo=TRUE,
                   check.names=TRUE)
is.data.frame(data)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
## [1] TRUE
```

After loading the Tidyverse, data frames can be cast to tibbles:

```
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 -
-

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() -
-
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

tidy_data <- as_tibble(data)
tidy_data

## # A tibble: 241,228 x 9
##    VAR_ID VAR_UNIT REGION_AG ITEM_AG MacroScen BioenScen IEA_SCEN ScenYear
##    <fct>  <fct>    <fct>     <fct>   <fct>     <fct>     <fct>    <fct>
##  1 Anim   1000 TLU World     Meat    SSP2      SPA0      scenRCP~ 2000
..

## 10 Anim   1000 TLU World     Meat    SSP2      SPA0      scenRCP~ 2090
## # ... with 241,218 more rows, and 1 more variable: OUTPUT_AG <dbl>
```

Note that the default print behavior of tibbles shows only the first few rows to prevent output flooding in case of large data sets.

4.6.6. Non-GDX data

Data for processing in the Tidyverse can be obtained from other storage formats than GDX. For example, the **readr** Tidyverse package (https://readr.tidyverse.org/) enables reading of rectangular data (like csv, tsv, and fwf). The use of Excel files is a deplorable but depressingly common practice. The **readxl** Tidyverse package (https://readxl.tidyverse.org/) makes it easy to get data ouf of Excel and into R.

Since GDX, though GAMS proprietary, is a relatively efficient data storage format that can represent sparse data, another approach is to convert your data to GDX—using GDX as an intermediate format—and process that with R. For example Excel data can be converted into GDX data using GDXXRW. Unfortunately, GDXXRW is Windows/Office specific. The xl2gdx (https://github.com/iiasa/xl2gdx) script provides a platform-independent alternative to GDXXRW.

4.6.7. Tidyverse processing

After importing data, Tidyverse processing is likely to involve cleanup, transformation, and visualization. Each of these activities is supported by a dedicated package.

4.6.7.1. Cleanup: tidyr

Imported data is typically not tidy data.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

The **tidyr** package helps clean it up.

As of **tidyr** 1.0.0, `pivot_longer()` and `pivot_wider()` replace `spread()` and `gather()`.

4.6.7.2. Transformation: dplyr

The **dplyr** package (https://dplyr.tidyverse.org/) provides a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges

- `filter()`: pick observations by their values..
- `arrange()`: reorder the rows.
- `select()`: pick variables by their names.
- `mutate()`: create new variables with functions of existing variables.
- `summarize()`: collapse many values down to a single summary.
- …

Composed with the `%>%` pipe operator.

4.6.7.3. Visualization: ggplot2

ggplot (https://ggplot2.tidyverse.org/): declaratively create graphics.

Based on the "Layered Grammar of Graphics" (https://www.tandfonline.com/doi/abs/10.1198/jcgs.2009.07098). Layers, Aesthetics, Facets, Stats, Geoms, Mappings composed with `+`.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

5. CONCLUSION

This deliverable provides a concept for the development of data interfaces in MIND STEP. We propose and apply R packages in combination with a version control system and a semi-automated way of documenting the interface functions, as well as the use cases. The use cases are applications of the R packages for problems modellers are usually confronted with. A review of exiting interfaces in different domains revealed that R scrips but also R packages have been already partially applied in the domain of existing models like IFM-CAP, CAPRI and GLOBIOM. It also crystalized that REST APIs become more relevant in the context of data access for geo- but as well modelling data bases.

With this deliverable we also finish the prototype phase for the interface development in MIND STEP. We will review with the MIND STEP partners the current interfaces and define further adjustment. Besides, the following issues will be relevant for the finalisation: The FSS interface need to be adjusted to the micro FSS data from EUROSTAT. Currently the application process is ongoing, and we right now register as research entity to propose a data analysis for MIND STEP. When the data access is provided, we will also adjust the R packages, such that it works with EUSTAT FSS data, as data formats and codes probably are different compared to the German interface to FSS. In addition, we will centralize all code developments on the GITLAB of IIASA, which was not operational for the prototypes. Further FadnUtils will be finalized such that the structured human readable file based on JSON fits to the requirements of the modelling teams in MIND STEP. The prototype for accessing GDX files using the example of GLOBIOM will be used, in combination with other already developed packages, to provide a comprehensive interface for the CAPRI data base. We also will assess how in the R package environment a REST API can be developed.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

6. ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 817566. Neither the European Commission nor any person acting on behalf of the Commission is responsible for how the following information is used. The views expressed in this publication are the sole responsibility of the author and do not necessarily reflect the views of the European Commission.

7. REFERENCES

Britz W., Lengers, B,.Kuhn, T. and Schäfer, D., 2014. A highly detailed template model for dynamic optimization of farms - FARMDYN, University of Bonn, Institute for Food and Resource Economics, Version September 2016, 147 pages.

Britz, W. , 2014. A New Graphical User Interface Generator for Economic Models and its Comparison to Existing Approaches, German Journal of Agricultural Economics, 63(4): 271-285.

Devos, W., Fasbender, D., Lemoine, G., Loudjani, P., Milenov, P., Wirnhardt, C., 2017. Discussion document on the introduction of monitoring to substitute OTSC - Supporting non-paper DS/CDP/2017/03 revising R2017/809, EUR 28830 EN, Publications Office of the European Union, Ispra, 2017, doi: 10.2760/258531.

Devos, W., Lemoine, G., Milenov, P., Fasbender, D., Loudjani, P., Wirnhardt, C., Sima, A. and Griffiths, P., 2018a. "Second discussion document on the introduction of monitoring to substitute OTSC: rules for processing applications in 2018-2019". EUR 29369 EN. Publications Office of the European Union, Ispra, 2018a, doi:10.2760/344612.

Devos, W., Lemoine, G., Milenov, P., Fasbender, D., 2018b. Technical Guidance on the decision to go for substitution of OTSC with checks by monitoring. Document. DS/CDP/2018/17, EUR 29370 EN. Publications Office of the European Union, Ispra, 2018b, doi: 10.2760/693101.

ECA, 2016, European Court of Auditors, 2016. The Land Parcel Identification System: a Useful Tool to Determine the Eligibility of Agricultural Land – but Its Management Could Be Further Improved Special report PL 2016 No. 25.

ECA, 2020. Using new imaging technologies to monitor the Common Agricultural Policy: steady progress overall, but slower for climate and environment monitoring, Special Report 04/2020

European Commission and the European Soil Bureau Network, 2004. The European Soil Database distribution version 2.0.

GitHub Docs (2021). Webpage: https://docs.github.com/en. GitHub, Inc. Accessed: 23rd April 2021.

Gocht, A. and Britz, W., 2011. EU-wide farm type supply models in CAPRI--How to consistently disaggregate sector models into farm type models, Journal of Policy Modeling, Elsevier, vol. 33(1), pages 146-167, January.

Kuiper, M., Shutes, L., 2014. MAGNET in SVN; Tips and tricks for developing under version control. The Hague, Wageningen Economic Research. Internal Document

Louhichi, K., Ciaian, P., Espinosa, M., Colen, L., Perni, A. and Gomez y Paloma, S., 2015. An EU-Wide Individual Farm Model for Common Agricultural Policy Analysis (IFM-CAP). JRC-IPTS-AGRILIFE Unit, European Commission.

Louhichi, K., Espinosa, M., Ciaian, P., Perni, A., Vosough Ahmadi, B., Colen, L. and Gomez y Paloma, S., 2018. The EU-Wide Individual Farm Model for Common Agricultural Policy Analysis (IFM-CAP v.1): Economic Impacts of CAP Greening, European Commission, Joint Research Centre, EUR 28829 EN, doi:10.2760/218047.

Meijl. H., 2021. Notes for new MAGNET website. Internal

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Münch, T., 2002. "Anpassungsstrategien für Marktfruchtunternehmen an zukünftige externe und interne Rahmenbedingungen am Beispiel der sächsischen Marktfruchtunternehmen", Dissertation, Halle, Germany

Neuenfeldt, S. and Gocht, A., 2014. A handbook on the use of FADN database in programming models, THÜNEN Working Papers 35, Johann Heinrich von THÜNEN Institute, Federal Research Institute for Rural Areas, Forestry and Fisheries.

Paracchini, M. L., Petersen, J.-E., Hoogeveen, Y., Bamps, C., Burfield, I. and van Swaay, C., 2008. High Nature Value Farmland in Europe. An estimate of the distribution patterns on the basis of land cover and biodiversity data. JRC Scientific and Technical Reports. European Communities, Luxembourg.

Panagos, P., Van Liedekerke, M., Jones, A., and Montanarella, L., 2012. European Soil Data Centre: Response to European policy support and public data requirements. Land Use Policy, 29(2): 329-338.

UNFCCC, 2016. Report of the Conference of the Parties on its twenty-first session, heldin Paris from 30 November to 13 December 2015.

Vogt, J., Colombo, R., Paracchini, M. L., de Jager, A., and Soille, P., 2003. CCM River and Catchment Database Version 1.0. JRC-IES EUR 20756 EN, European Commission. http://www.ec-gis.org/Workshops/9ec-gis/papers/demos_dejager.pdf

Woltjer, G., Rutten, M. and M. Kuiper, 2013. Working with Magnet: User guide. The Hague, Wageningen Economic Research. Internal document.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

8. APPENDIX

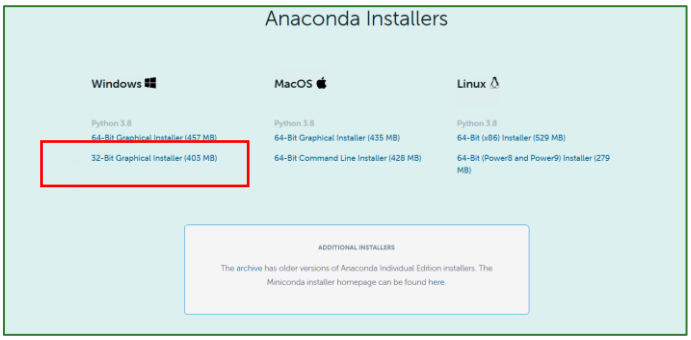
8.1. Requirements on the local machine: Anaconda, R with R Studio, Git installation

{THÜNEN, X. Yang}

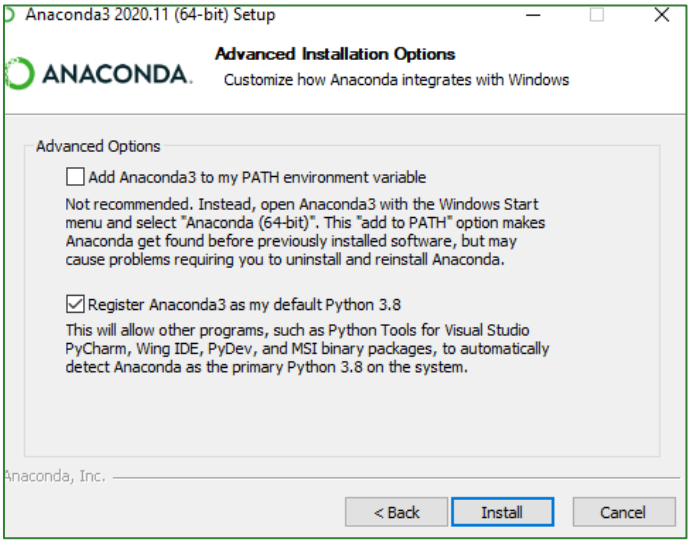
8.1.1. Installation Anaconda for Windows OS

Anaconda is a package manager, an environment manager, and Python distribution that contains a collection of many open-source packages.

Go to the Anaconda Website and choose a Python 3.x graphical installer (A) or a Python 2.x graphical installer (B). If you aren't sure which Python version you want to install, choose Python 3. Do not choose both.



We recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.



Click on Next. You can install Pycarm if you like. Click on Next. Click on Finish.

8.1.2. Installation R and R Studio from anaconda

RStudio is an integrated development environment for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

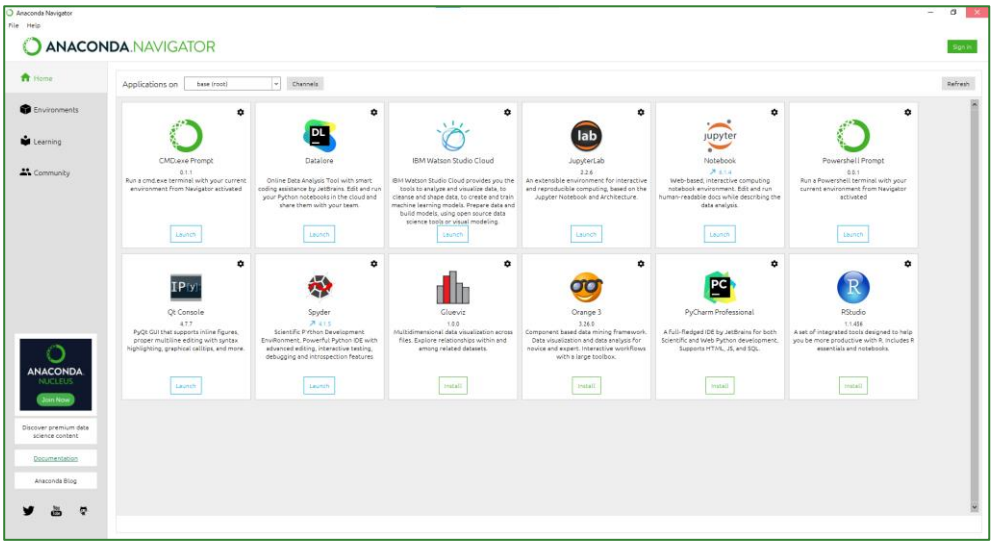
- Create a new environment for R and R-Studio. Open Anaconda Prompt.


```
(base) C:\>conda create  -n r_env python=3.7.1
```

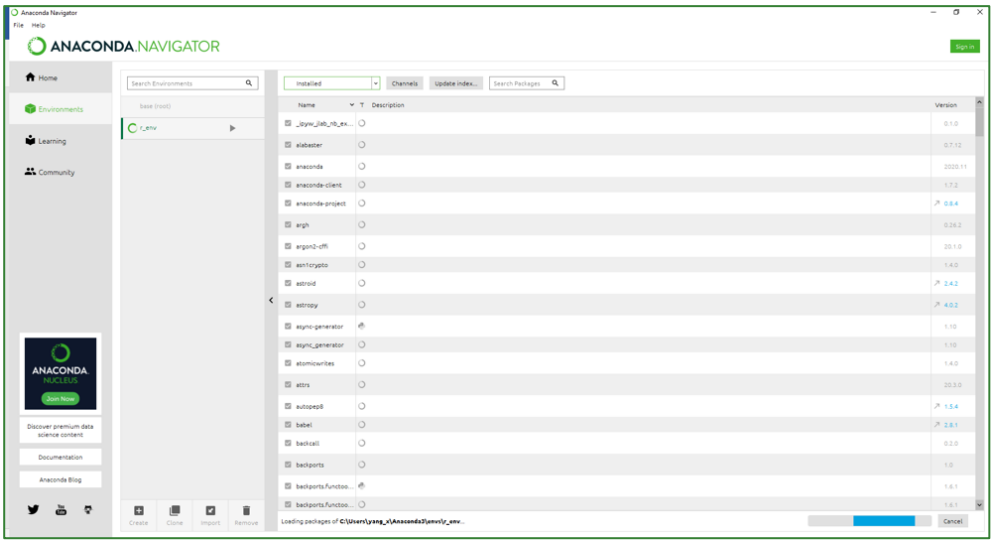
- Install R-Studio. We can install R-Studio with following commands:

```
(base) C:\>conda install -n r_env -c defaults rstudio
```

- Launch R-Studio in anaconda. Open Anaconda Navigator.

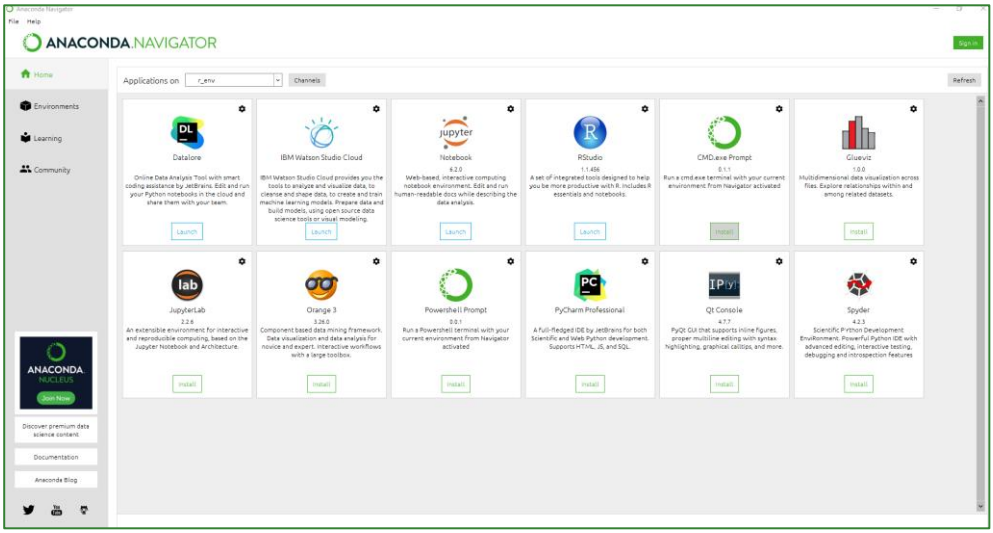


We have created a new R environment “r_env”. Navigator activates it, as shown by the highlighted green bar. All actions take place in the active environment.

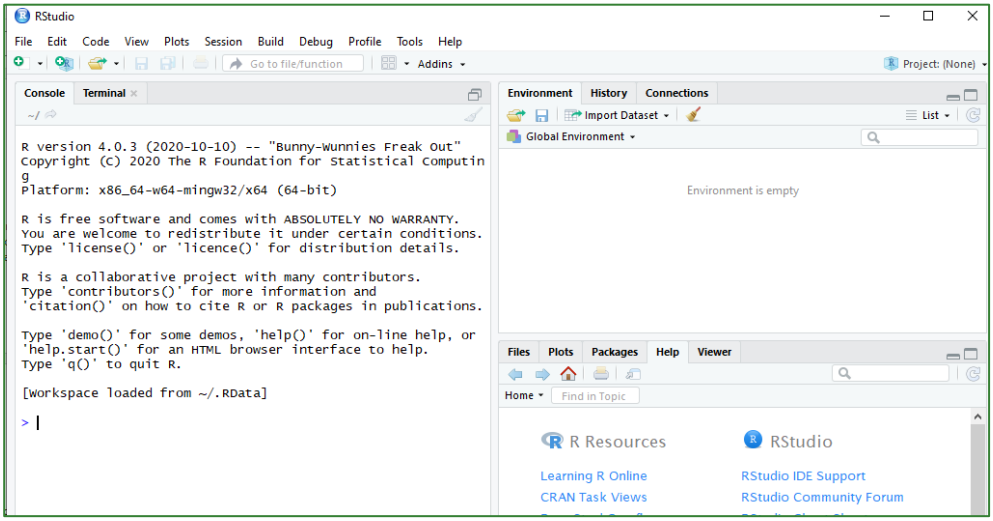


With the new environment active, click the home button and then click the “Launch” button.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

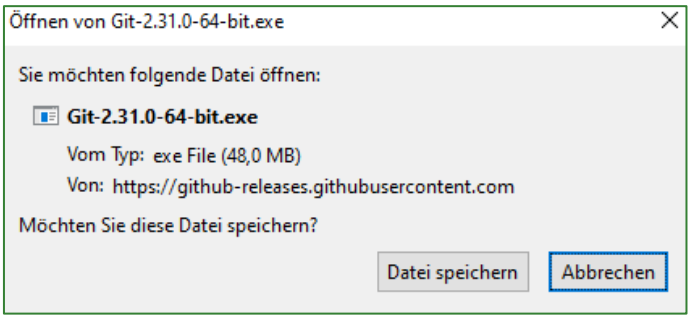


You will see R-Studio running from Navigator.



8.1.3. Installation Git

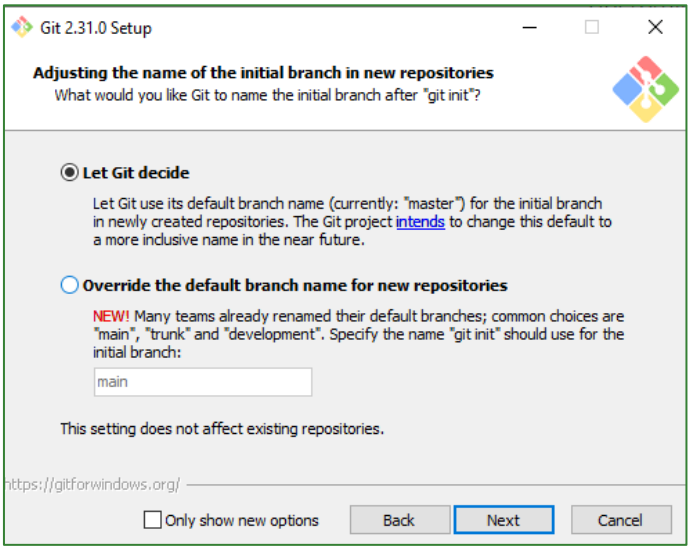
Go to https://git-scm.com/download and download Git on different OS. Click on “Save file”.



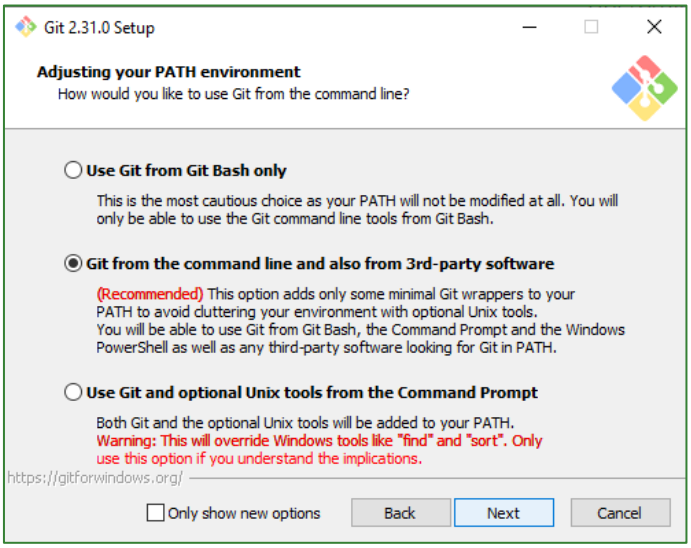
Double click the file.

Choose “Let Git decide “, then click on Next.

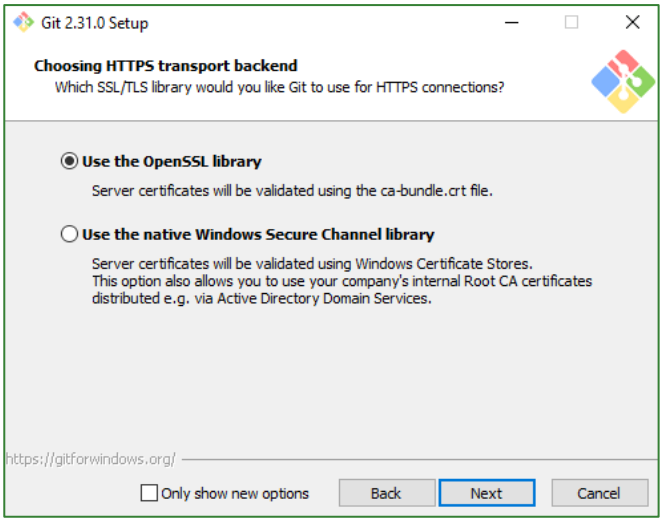
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.



Choose “Git from the command line and from 3rd-patry software”, then click on “Next”.

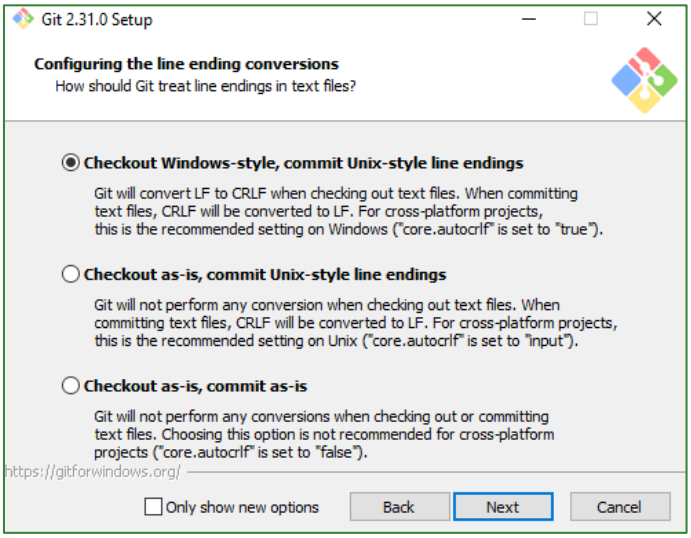


Choose “Use the openSSL library”, click on “Next”.

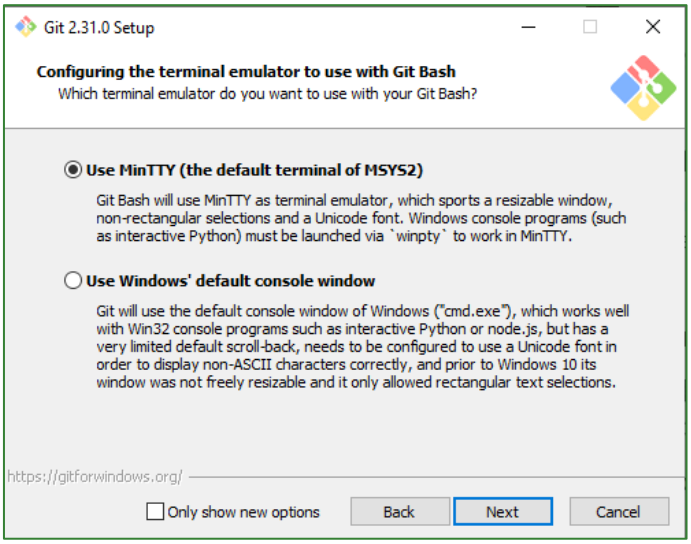


Choose the default choice, click on Next.

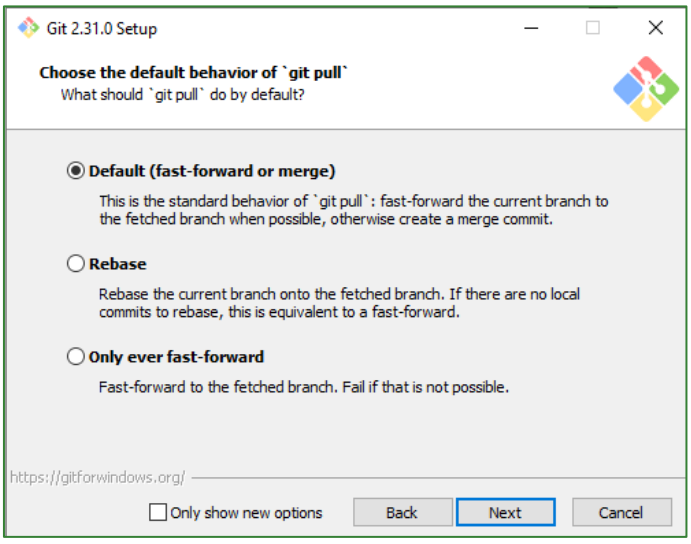
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.



Choose “Use MinTTY”, click on Next.

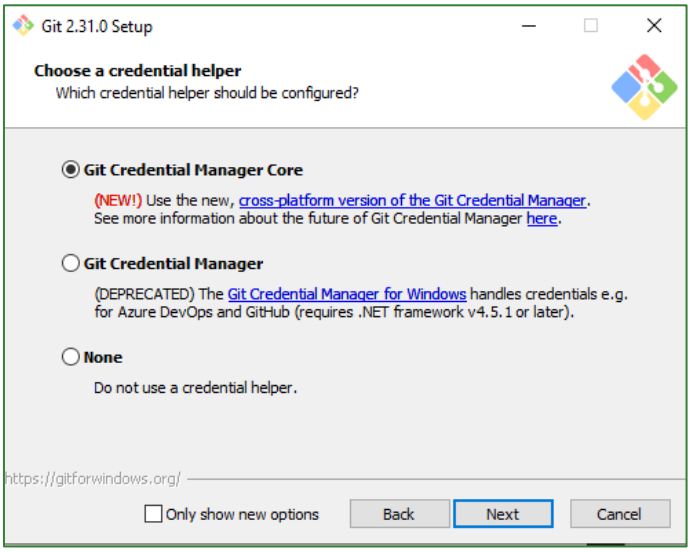


Click on Next.

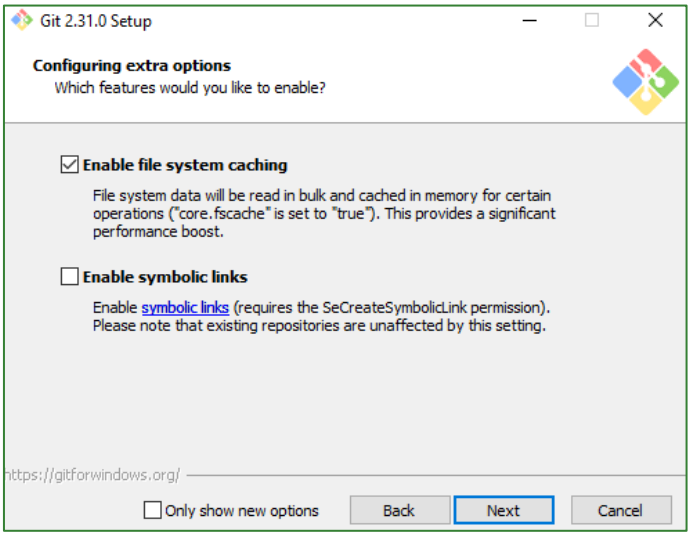


Click on Next.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.



Click on Next.



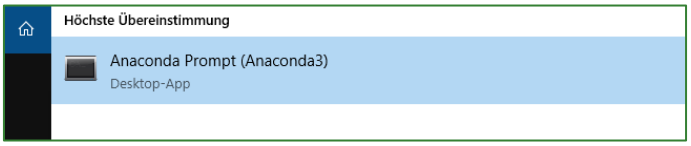
Click on Next. Click on “Finish”.

8.1.4. Packaging a conda environment

We can run multiple versions of different software at the same time by creating different environments. The directory of the newly created software environment is *anaconda_root/envs/environment_name.* We can copy the packaged environment to target machine and unzip it. This is useful for deploying code in a consistent environment.

1. Pack your conda environment

- Open Anaconda Prompt



- List all installed environments with command line `$ conda info --envs`

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

```
(base) C:\>conda info --envs
# conda environments:
#
base                  *  C:\Users\[redacted]\Anaconda3
env_python_3.8           C:\Users\[redacted]\Anaconda3\envs\env_python_3.8
mypython36_env           C:\Users\[redacted]\Anaconda3\envs\mypython36_env
new_env_r                C:\Users\[redacted]\Anaconda3\envs\new_env_r
r_3.6.0                  C:\Users\[redacted]\Anaconda3\envs\r_3.6.0
r_env                    C:\Users\[redacted]\Anaconda3\envs\r_env
```

- Export your environment without the build info by using the `--no-builds` flag from windows to yml file.

  `$ conda env export -n mypython36_env -f env.yml --no-builds`

2. Unpack a conda environment

- Create a new environment named `new_env` from the `yml` file on the target machine, replace the `yml file path` to location of `env.yml` file.

  `$ conda env create --name new_env --file=yml file path\env.yml`

```
(base) C:\>conda env create --name new_env --file=C:\Users\yang_x\Desktop\Git\irgq\R_env\env.yml
Collecting package metadata (repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.9.2
  latest version: 4.10.0

Please update conda by running

    $ conda update -n base -c defaults conda


Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate new_env
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

- After creating your new environment, you can run `$ conda info --envs` to check your new environment. The environment created by conda is always located in */Users/.../anaconda3/envs/*. A new Python environment has been successfully created. Now we can activate and use it.

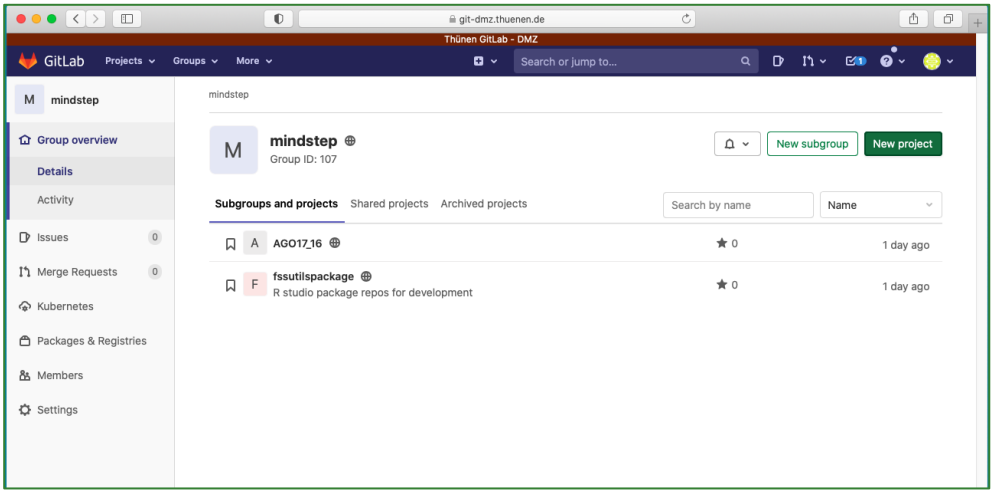
```
(base) C:\>conda info --envs
# conda environments:
#
base                  *  C:\Users\yang_x\Anaconda3
env_python_3.8           C:\Users\yang_x\Anaconda3\envs\env_python_3.8
mypython36_env           C:\Users\yang_x\Anaconda3\envs\mypython36_env
new_env                  C:\Users\yang_x\Anaconda3\envs\new_env
new_env_r                C:\Users\yang_x\Anaconda3\envs\new_env_r
r_3.6.0                  C:\Users\yang_x\Anaconda3\envs\r_3.6.0
r_env                    C:\Users\yang_x\Anaconda3\envs\r_env
```

8.2. Group: Developer - Setup of the GitLab repository, R package project and interlink GitLab with the R package

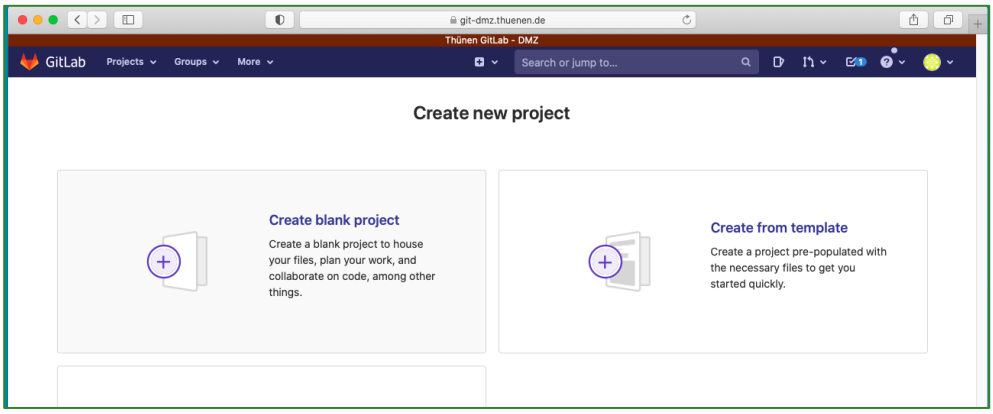
{THÜNEN, Yang}

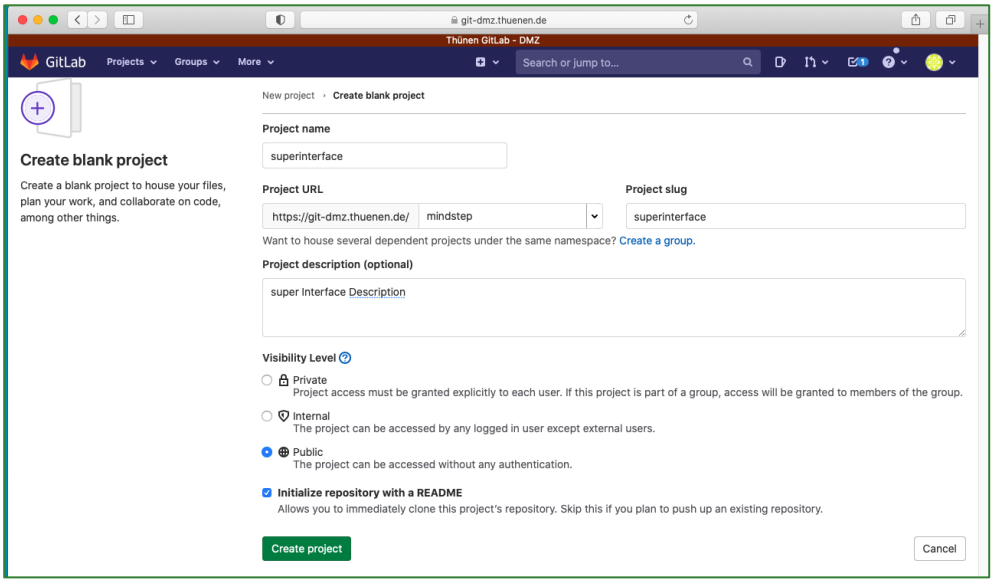
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

After registration of the mind step users in https://git-dmz.THUENEN.de. Get username and password from XinXin.Yang@THUENEN.de. You will be added to the group of MIND STEP in GitLab and can create your only project.

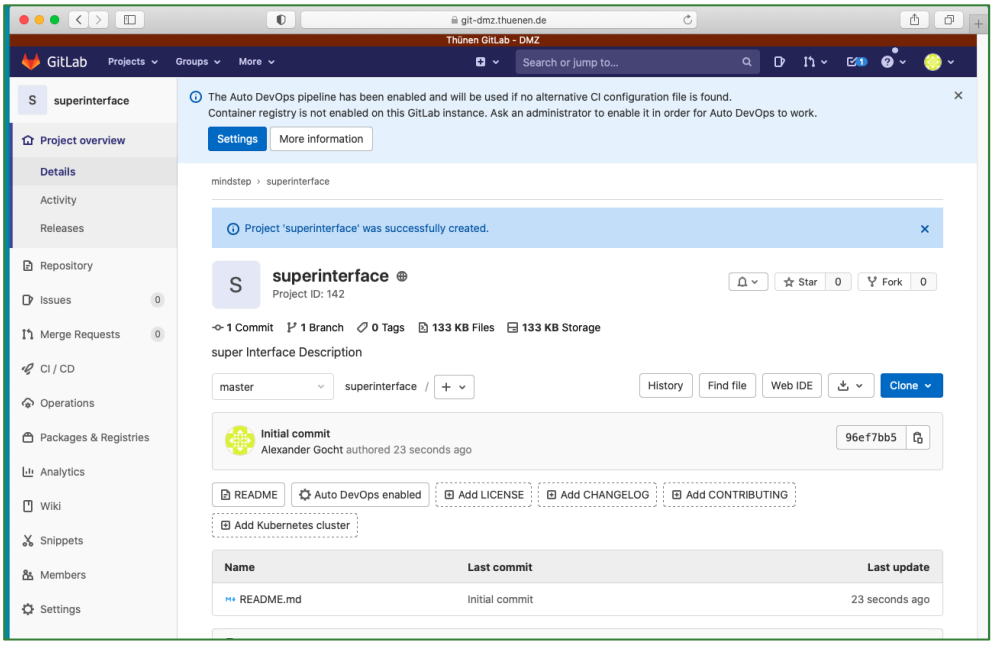


Create your project at https://GitLab-dmz.THUENEN.de/MIND STEP/superinterface.giz

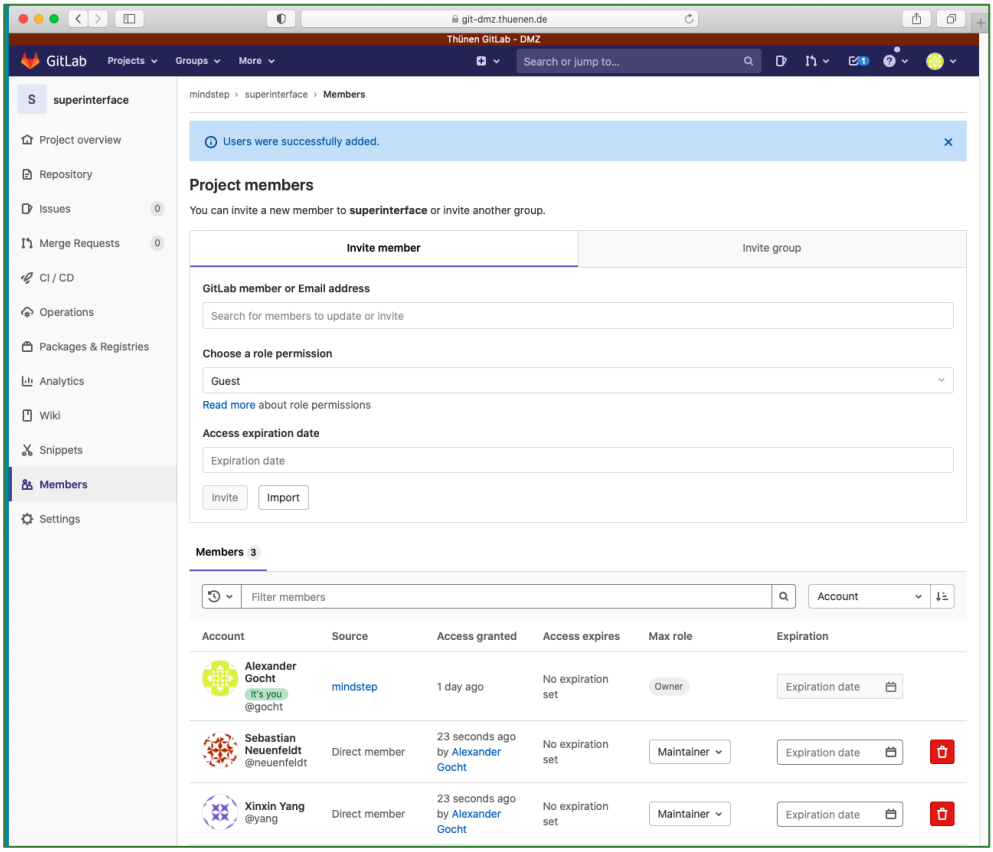




This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

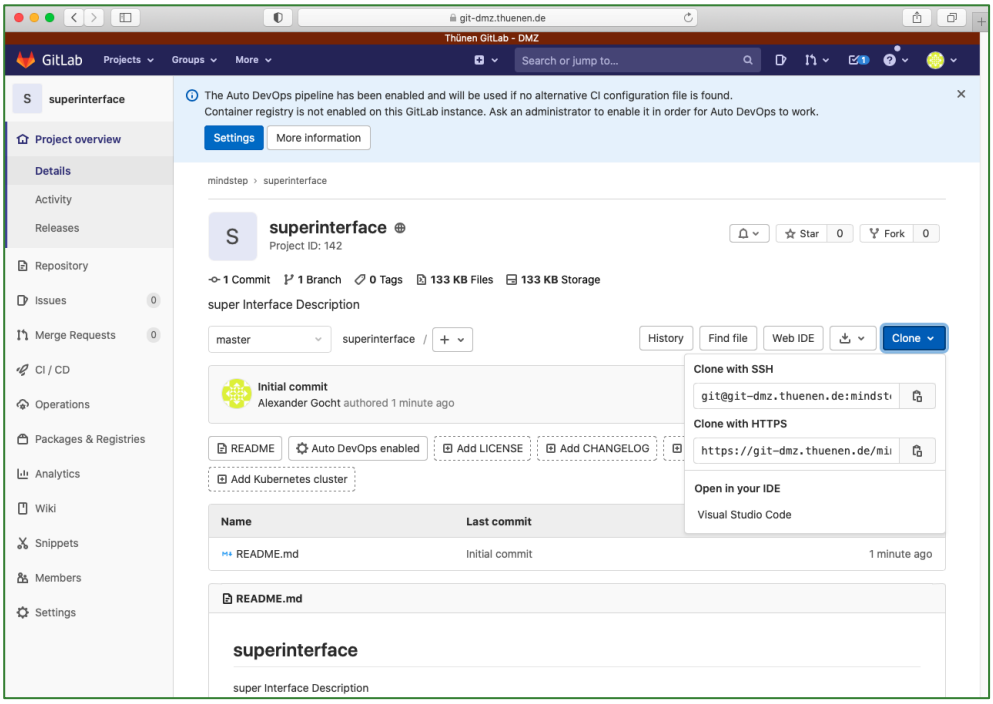


Add user for the project.

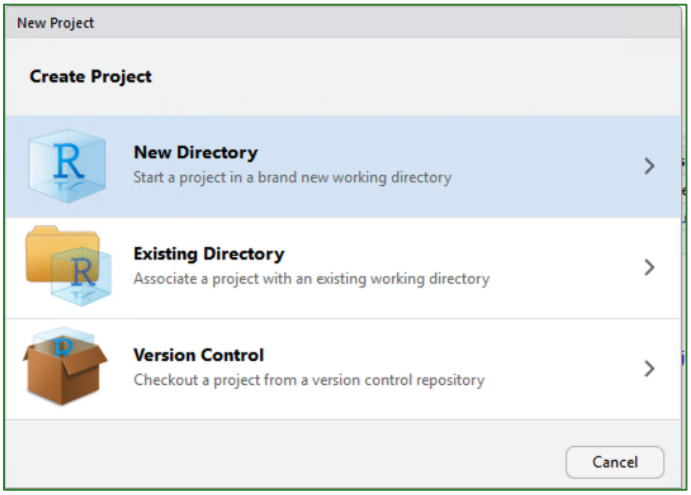


Get the URL link for the empty project:

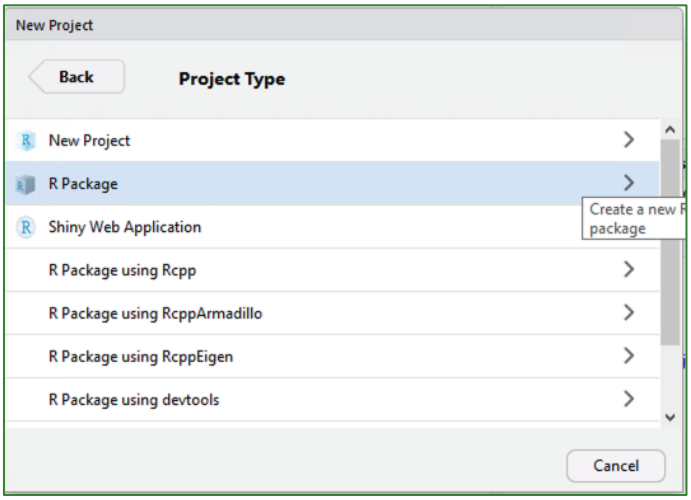
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

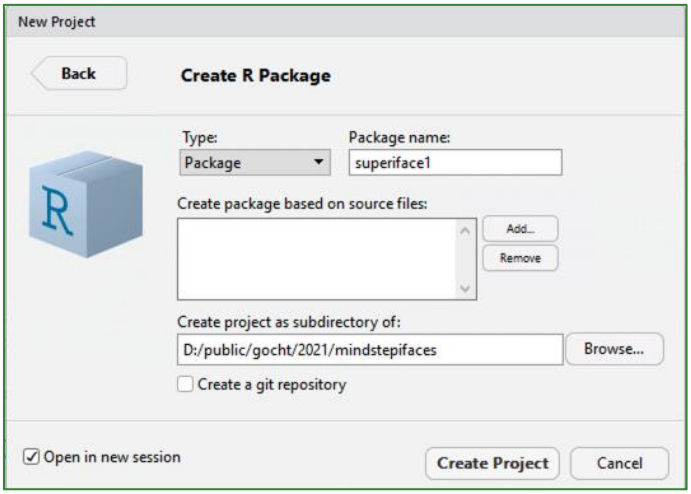


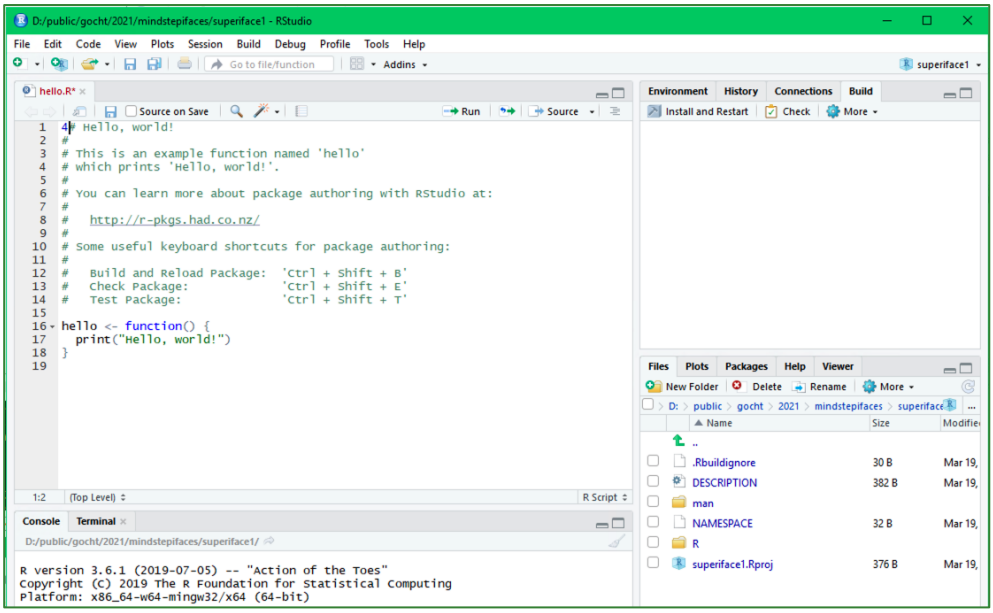
Create project locally with R-Studio for an interface package let's call it "superiface1". Please note select name without special characters. Please create a project in R-Studio for a new package.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

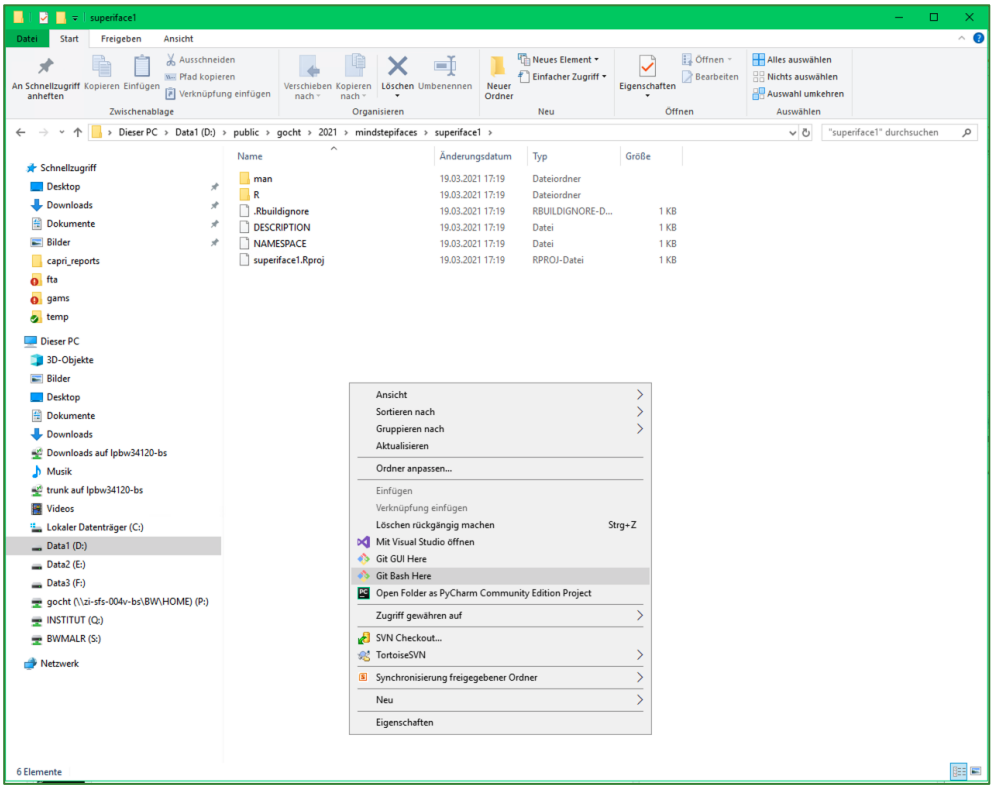


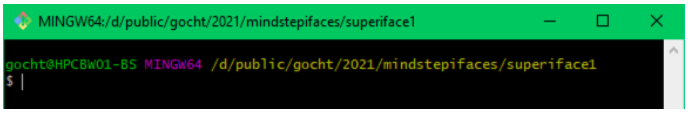




Go to git batch:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

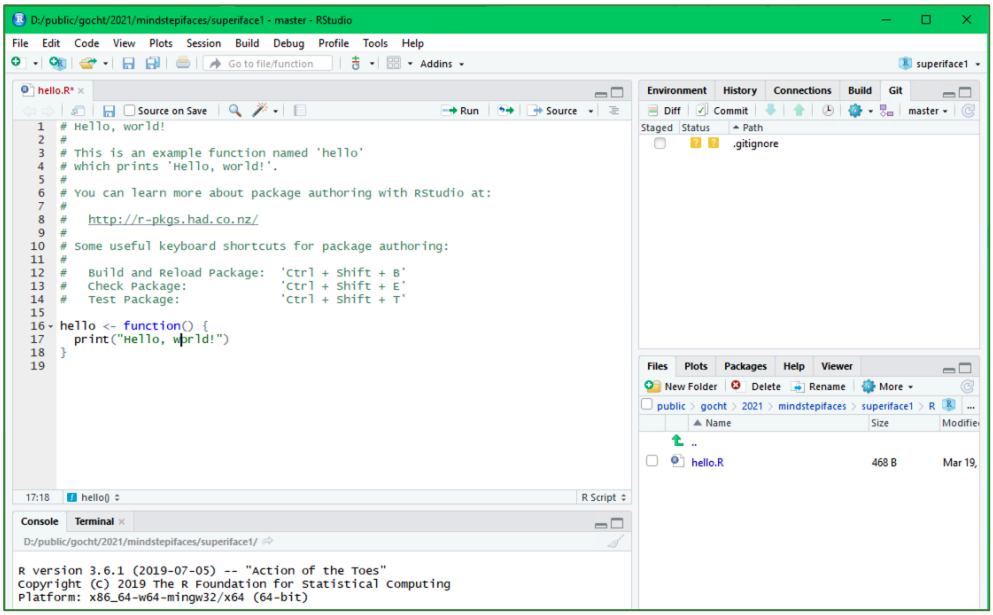





```
gocht@HPCBW01-BS MINGW64 /d/public/gocht/2021/MIND STEPifaces/superiface1

git init

git remote add origin https://git-dmz.THUENEN.de/MIND STEP/superinterface1.git

git add .

git commit -m "initial test for package commit"

# if the GitLab repository is not empty use:

git pull https://git-dmz.THUENEN.de/MIND STEP/superinterface1.git master --allow-unrelated-histories

# if empty use

git pull https://git-dmz.THUENEN.de/MIND STEP/superinterface1.git master
```

open R project in R Studio

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.



In the git console link the R package project to the GitLab of THUENEN using the following command:

Commit and Push package project or use the Git console like:

```
git add .  # to add all files
git commit -m “some nice description”
git pull origin master
# to upload files to the GIUTLAB
git push -u origin master
# to understand which remote repository was links
git remote -v
```

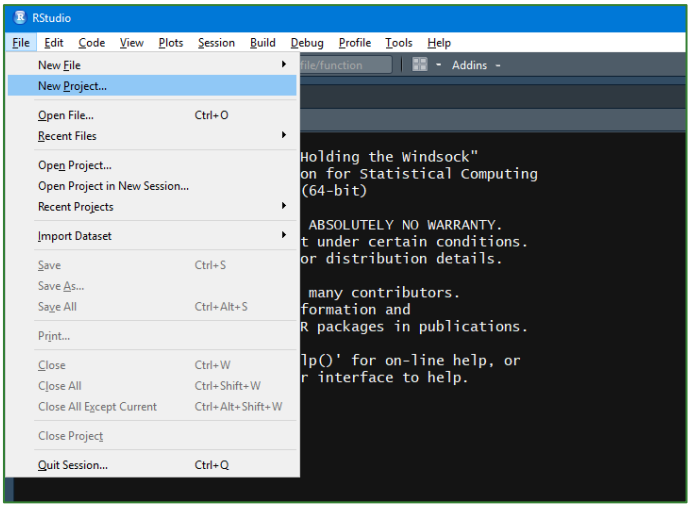
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

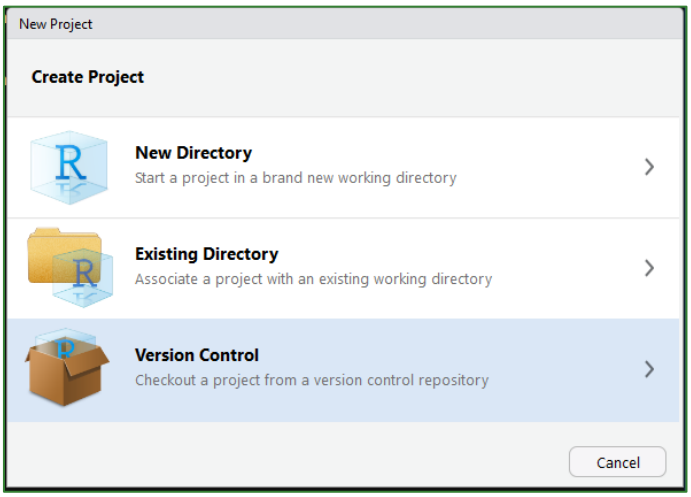
8.3. Group: Developer with the intention to adapt or add to the package functions and compile a manual

{THÜNEN, S. Neuenfeldt}

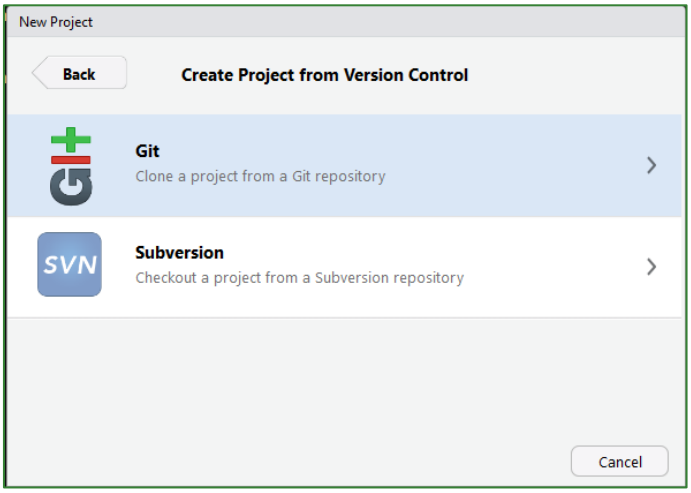
8.3.1. General instructions

Open new Project in R from repository download from "https://git-dmz.THUENEN.de/MINDSTEP/superiface1.git".

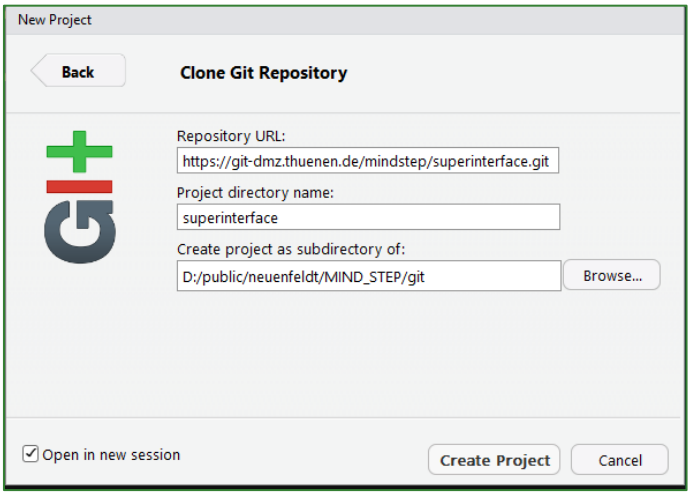




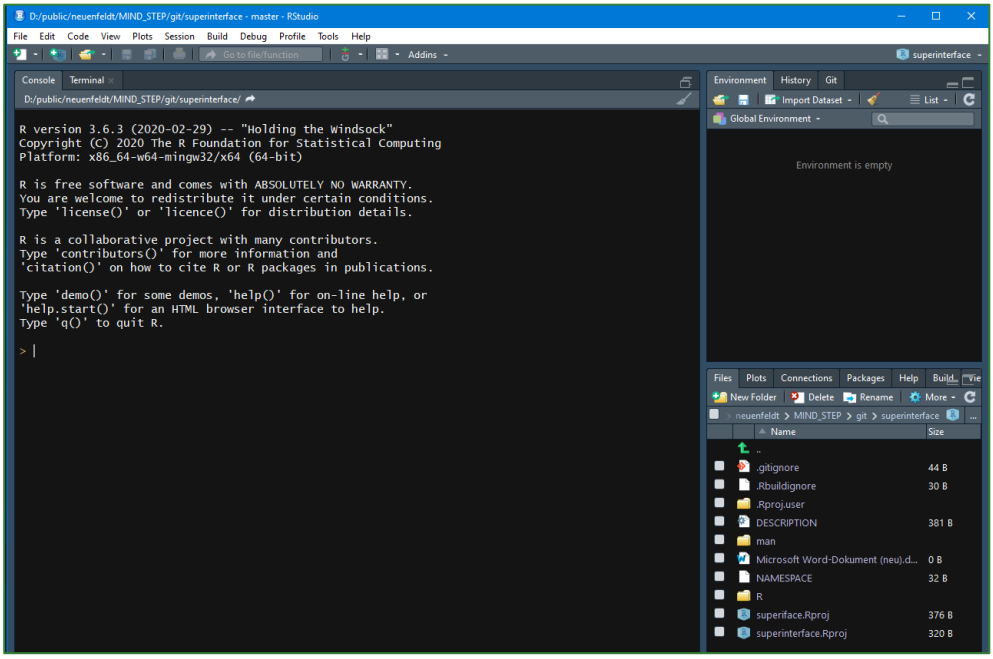
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.



Important: correct repository URL starting with “https://”.

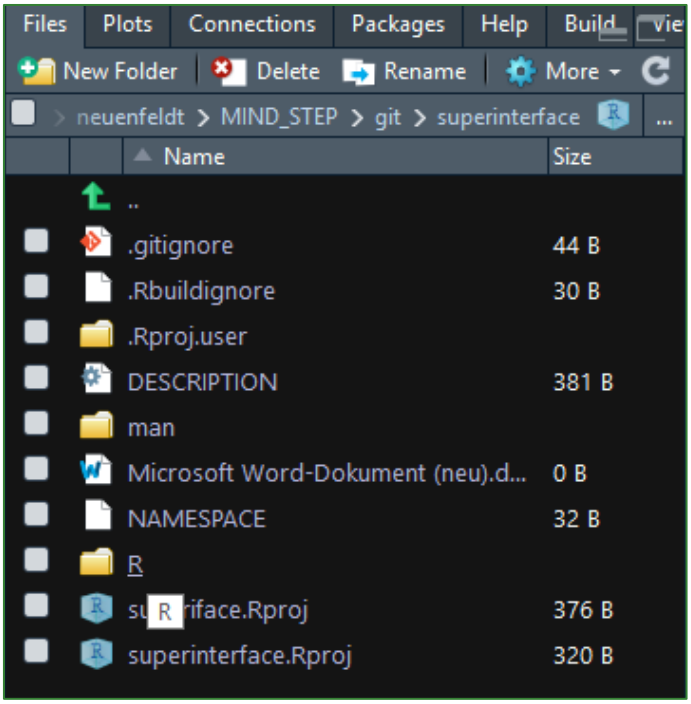


This should end up in a new session of R-Studio with the project:

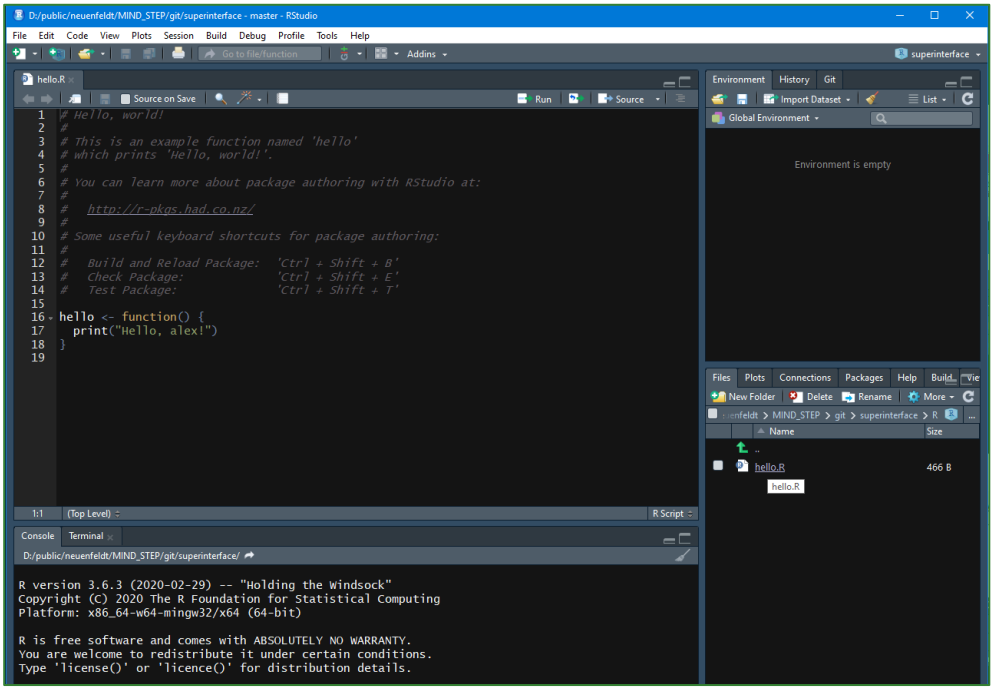


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

Do some outstanding work by changing the functions, therefore go into the folder “R” as normally part of the lower right panel:

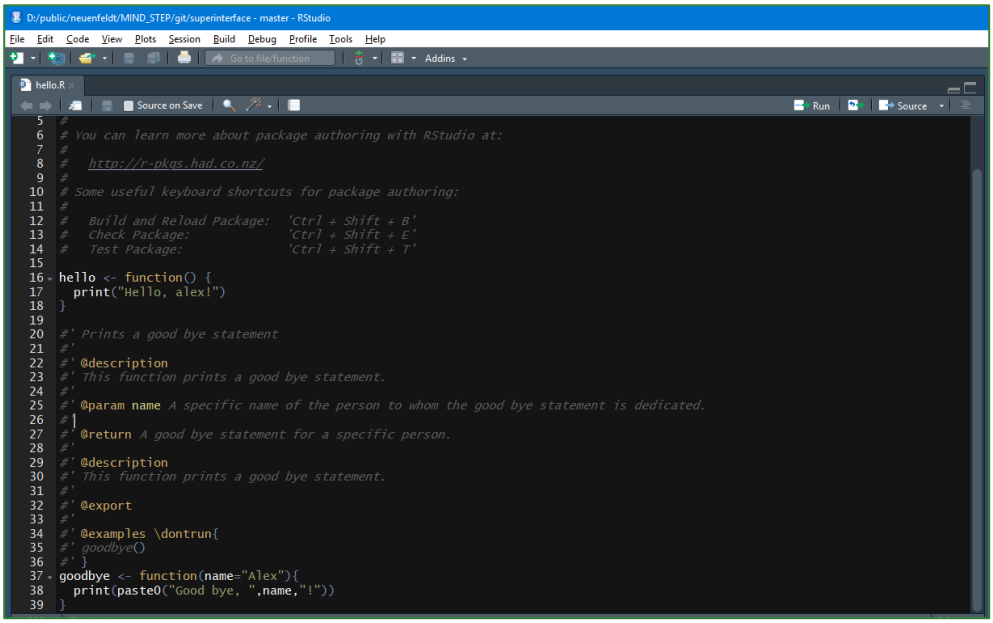


Then left-click on “hello.R” function and it should automatically be opened in the upper left panel.



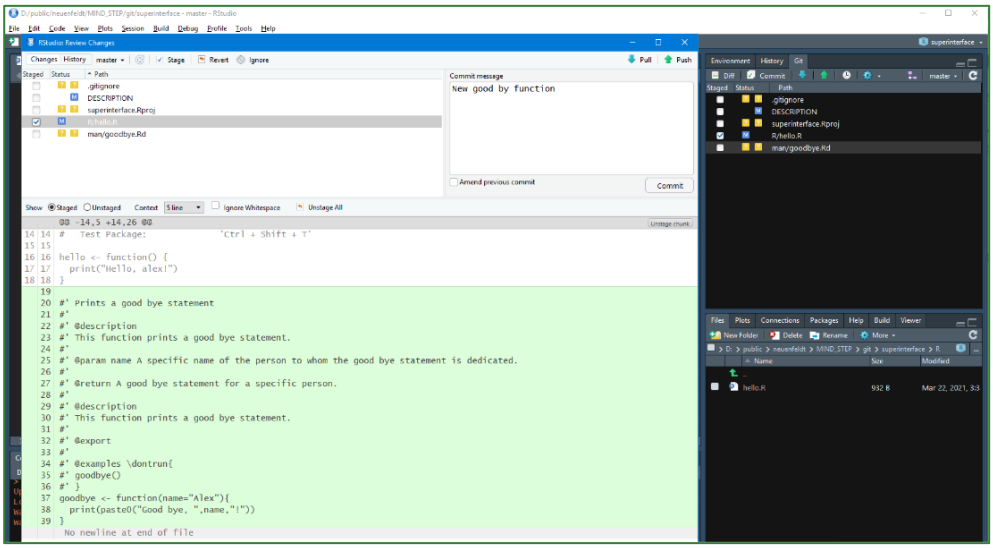
Add some function, in this case the function `goodby()` in line 37-39 as part of “hello.R”. Add description to the function, here in line 20 to 36. This is important when the manual or readme will be produced based on the meta information.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

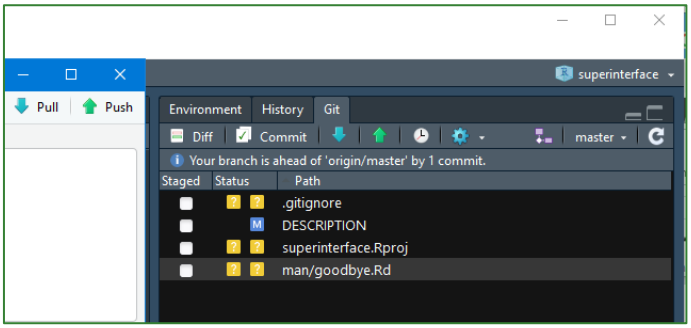


Explain and comment the code if description is not enough.

Then first commit and push afterwards (in case conflicts -> remove those): select the adapted “hello.R” file and have a look at the lower panel which shows the changes. Add a useful comment to describe the changes.



When commit is done, click on “push” which can be done in the panel of the commit window.

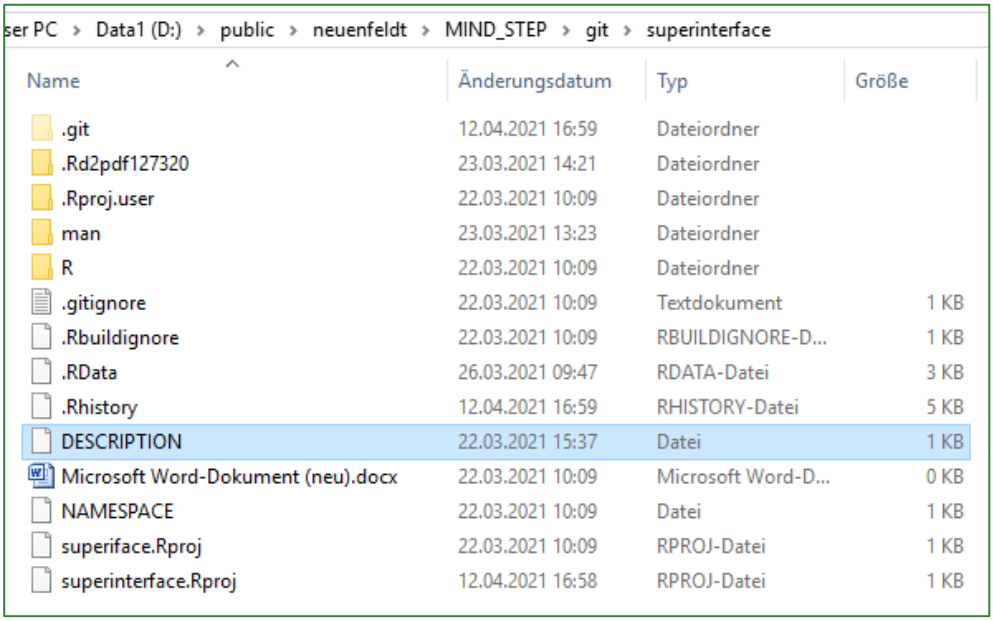


Successful pushing should look like this:

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

8.4. Building manual

Before building the pdf manual you should fill in some information about the package in the DESCRIPTION file in the package folder.



You can open it with an editor or just in RStudio. The file should look like this:

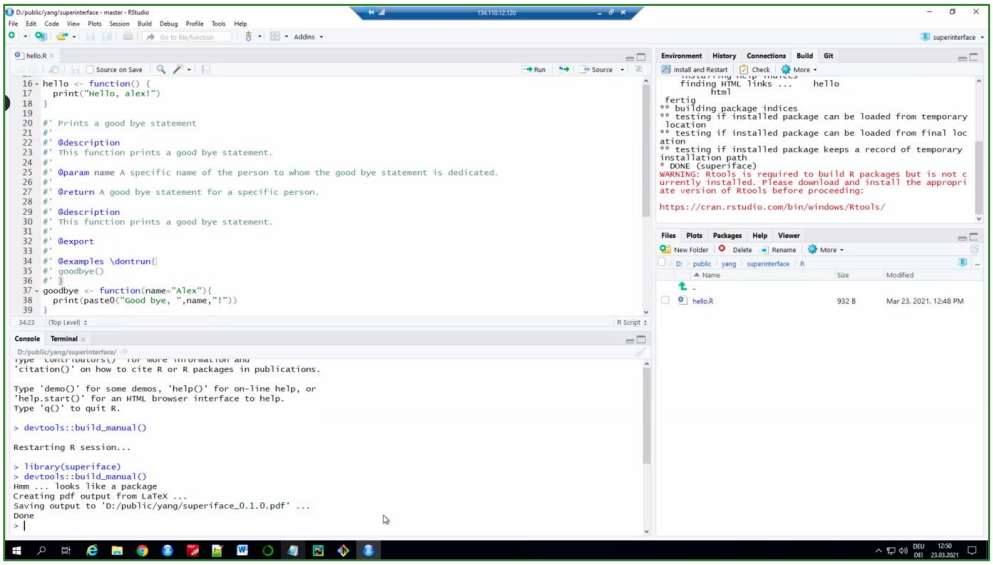
```
Package: superiface
Type: Package
Title: What the Package Does (Title Case)
Version: 0.1.0
Author: Who wrote it
Maintainer: The package maintainer <yourself@somewhere.net>
Description: More about what it does (maybe more than one line)
    Use four spaces when indenting paragraphs within the Description.
License: What license is it under?
Encoding: UTF-8
LazyData: true
RoxygenNote: 7.1.1
```

How to fill in meaningful statements can be elaborated at https://cran.r-project.org/doc/manuals/r-release/R-exts.html#The-DESCRIPTION-file or at https://www.mzes.uni-mannheim.de/socialsciencedatalab/article/r-package/#subsection4-3 for instance.

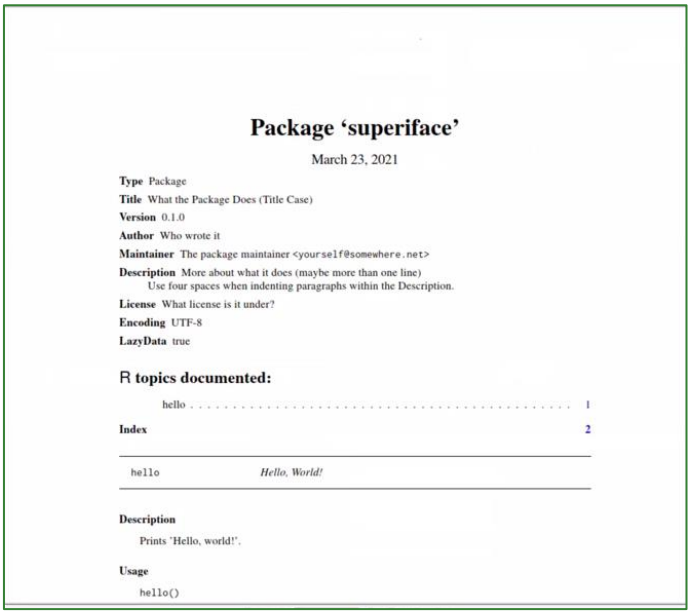
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

As can be seen in the description file, a license should be given. This regulates how your code can be used by others. Licensing is not the easiest task. Have a look at https://choosealicense.com/ what licenses are possible. If you have something in mind like "I care about sharing improvements.", than type in your Rstudio console use_gpl_license(). This means, "…GNU GPLv3 also lets people do almost anything they want with your project, except distributing closed source versions." (https://choosealicense.com/).

First run `devtools::document()` to generate R-documents (.Rd) in the "man" folder of the package. This will be needed to generate the PDF manual. All R-Files that are produced and stored in "R" folder of the package are recognized. Please describe as best as possible your R-files as explained above. Building the PDF manual for hypertext and PDF documentation for Annex in Del2.1 can be done via `devtools::build_manual()`:



This will generate a nicely formatted pdf documentation:



Package 'superiface'

March 23, 2021

Type Package
Title What the Package Does (Title Case)
Version 0.1.0
Author Who wrote it
Maintainer The package maintainer <yourself@somewhere.net>
Description More about what it does (maybe more than one line)
Use four spaces when indenting paragraphs within the Description.
License What license is it under?
Encoding UTF-8
LazyData true

R topics documented:

hello 1
Index 2

hello *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

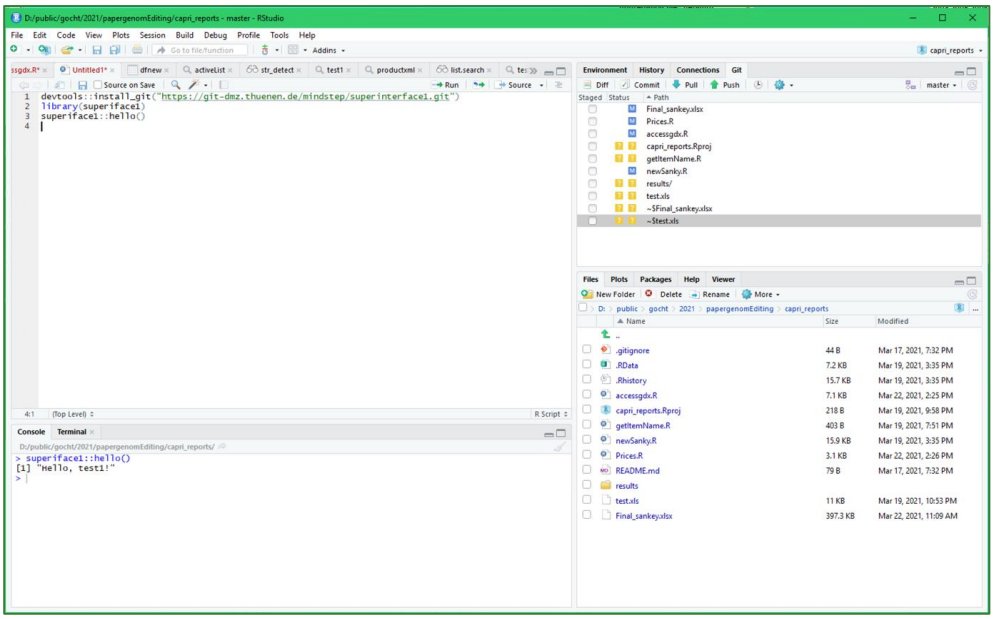
```
hello()
```

8.5. Use case documentation (group user or developer)

{THÜNEN, Yang}

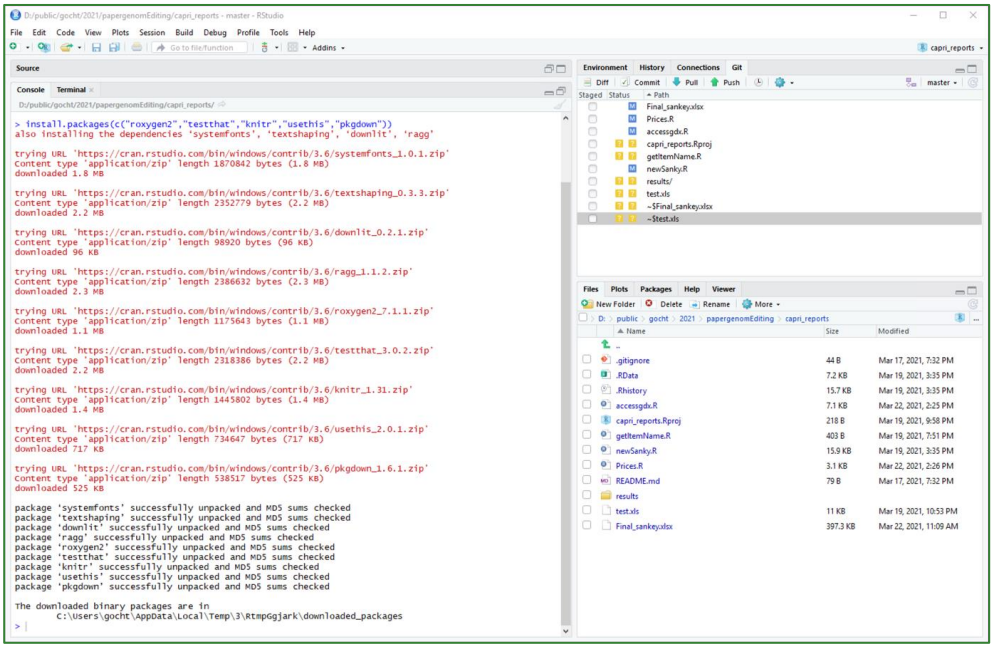
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Install package from https://GitLab-dmz.THUENEN.de/MIND STEP/superinterface.giz and load. But before you need to install package “devtools” using the command `install.packages(“devtools”)`. Then you can use the function from the package:



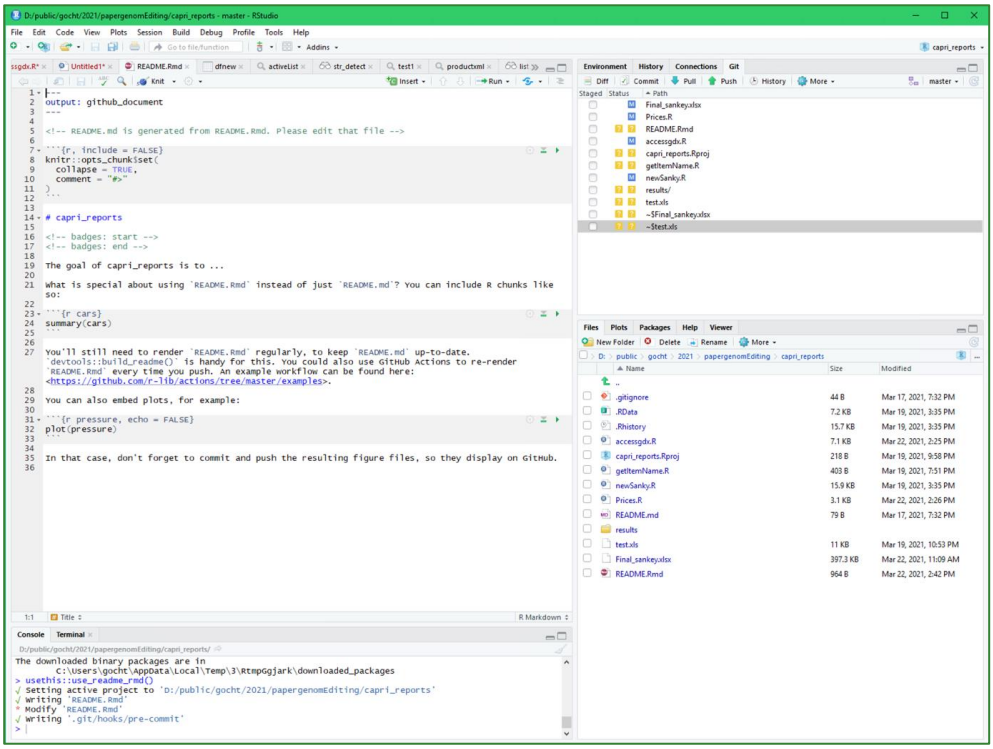
Install package for mark down: Put into the console the following statement to install the packages required for using mark down for use case documentation:

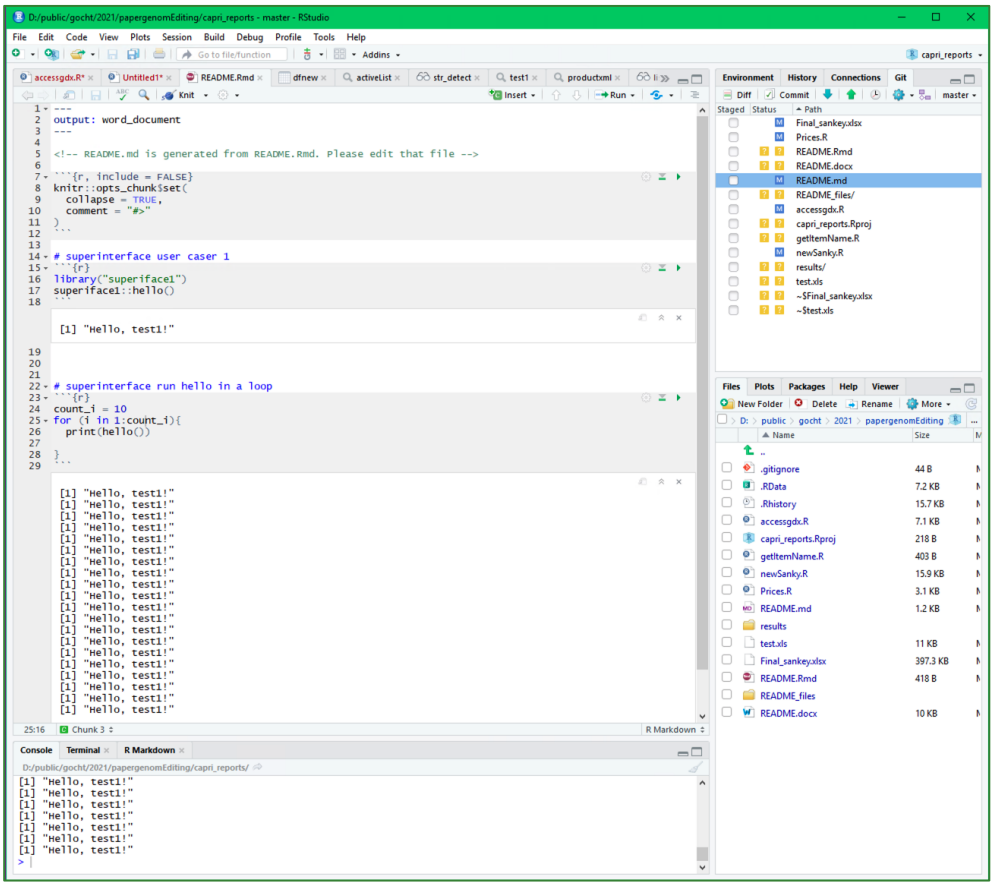
`install.packages(c("roxygen2","testthat","knitr","usethis","pkgdown"))`



Then create new mark down with: `usethis::use_readme_rmd()` creates default mark down language project. Insert R cell using green button. Write a use case of the developed function:

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.





For more see: https://rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf

To export as word put output: `word_document` and to start to compile the word document use control + shift + K.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

8.6. Useful hints and troubleshooting

8.6.1. Building PDF manual

{THÜNEN, Neuenfeldt}

If you have problems with making the PDF manual due to missing “pdflatex”, please check the following instructions.

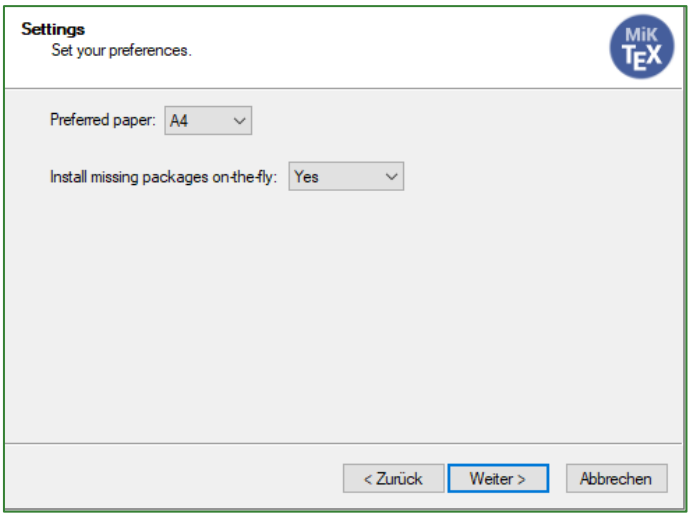
If calling `devtools::build_manual()` in your project/package in the console folder gives:

```
Converting Rd files to LaTeX

Error in texi2dvi(file = file, pdf = TRUE, clean = clean, quiet =
quiet,  :
```

pdflatex is not available

```
Error in texi2dvi(file = file, pdf = TRUE, clean = clean, quiet =
quiet,  :

  pdflatex is not available

Error in running tools::texi2pdf()

Error: Failed to build manual
```

Be sure you have installed MiKTeX, which is a Latex to PDF converter. If you have to install MiKTeX, let it automatically install packages (if possible).[30]



Then check if the path in the R environment is correctly specified. `Sys.getenv("PATH")` shows all folders of programs in the R environment and MiKTeX should be part of it. Then check if the path is correct. If the path is not correct, R does not know where “pdflatex.exe” is located and `Sys.which("pdflatex")` gives an empty path (“”). In my case there was mistakenly an quotation mark at the end of the folder of MiXTeK.

```
Sys.getenv("PATH")

C:\\Users\\neuenfeldt\\AppData\\Local\\Programs\\MiKTeX\\miktex\\bin
\\x64\""
```

30 If not, you have to install some packages when you run devtools::build_manual() the first time.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

Must be

```
C:\\Users\\neuenfeldt\\AppData\\Local\\Programs\\MiKTeX\\miktex\\bin
\\x64\\"
Sys.which("pdflatex")
pdflatex
      ""
```

Solution:

Call `Sys.getenv("R_USER")` in console to find out where the ".Renviron" file is located. Recognise the path and move into that folder and open it with an editor. In my case, this file has on entry:

`PATH="${RTOOLS40_HOME}\usr\bin;${PATH}"`

You have to put the folder of MiKTeX where pdflatex.exe is located. Open the path of MiKTeX from above and take care that pdflatex.exe is available there. Then this entry in ".Renviron" file must be adapted to have that folder in, but be precise (like the one given in red color):

`PATH="${RTOOLS40_HOME}\usr\bin;C:\Users\neuenfeldt\AppData\Local\Programs\MiKTeX\miktex\bin\x64;${PATH}"`

Close RStudio and start again. If everything went fine:

```
Sys.which("pdflatex")
```

pdflatex

```
"C:\\Users\\NEUENF~3\\AppData\\Local\\Programs\\MiKTeX\\miktex\\bin\
\x64\\pdflatex.exe"
```

Gives the right path.

Then, open your project and redo:

```
devtools::build_manual()
Hmm ... looks like a package
```

Creating pdf output from LaTeX ...

```
Saving output to 'C:/git/superiface_0.1.0.pdf' ...
Done
```

8.6.2. Importing a Git repository using the command line

{THÜNEN, Neuenfeldt}

In this sub-chapter we show how to import a Git repository from one group to another. This explanation is based on GitHub Docs (2021) (https://docs.github.com/en/github/importing-your-projects-to-github/importing-a-git-repository-using-the-command-line).

This action is quite useful if you have already produced a repository or package and want to work together with members of another group.

1. Create a new project/repository in the group (see 8.2) in which the package/project/repository should be imported.
2. On your local system go into the folder in which the package should have a directory and open "Git bash" via right click.

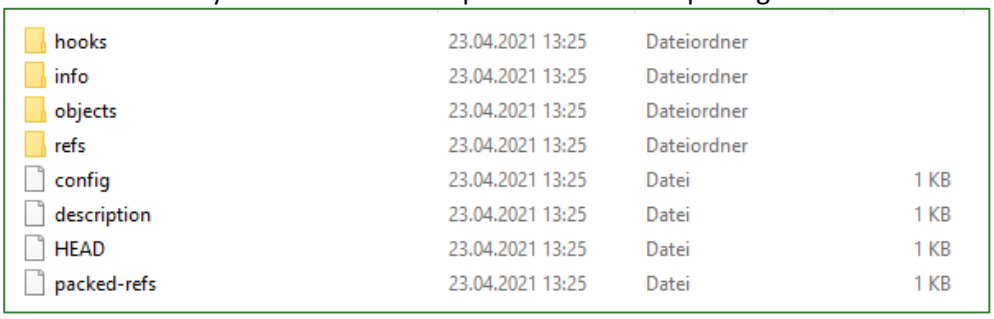
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

The Git command line opens. Now make a bare clone of the repository into a local directory. Use the URL in which your package you want to transfer is actually stored.

```
$ git clone --bare https://some-host.com/someuser/somepackage.git
```

As GitHub Docs (2021) states, "[t]his creates a full copy of the data, but without a working directory for editing files, and ensures a clean, fresh export of all the old data."

This new directory looks different compared to the usual package folder structure:



3. After the local directory "../somepackage.git" is build, change to this directory and push it to GitLab using the "mirror" option, which ensures that all references, such as branches and tags, are copied to the imported repository. Take the name of the created project from 1.

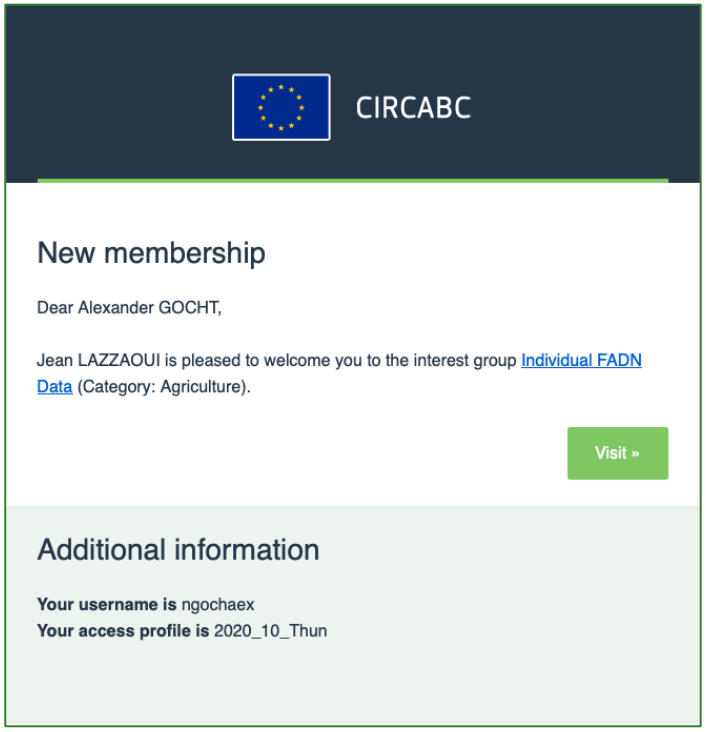
```
$ cd somepackage.git
$ git push --mirror https://some-other-host.com/anothergroup/somepackage.git
```

4. Now you can remove the local directory "../somepackage.git".

8.7. FADN data download and decoding

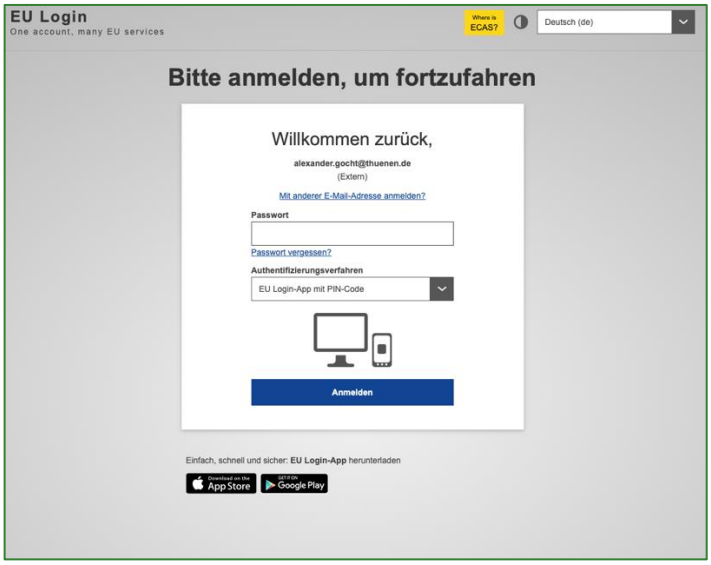
{THÜNEN, Gocht}

1. After creating the public key with GPG4Win send the public key to the commission officer. Then you will receive a phone call to provide the fingerprint of your keys. This allows the officers to encrypt and upload the files for your access on CIRCABC.
2. When the data is ready you will receive an invitation mail (check you spam folder)



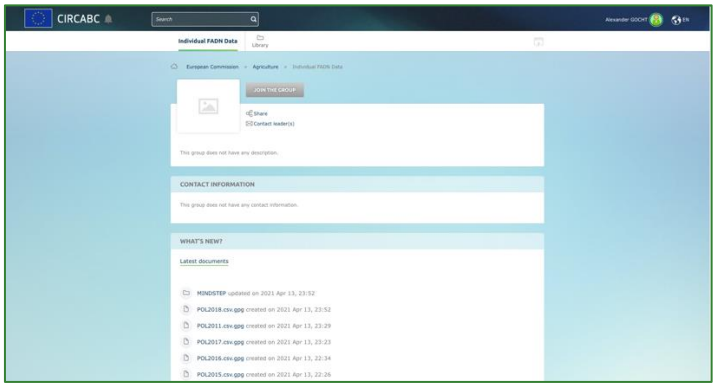
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

3. From there you need to log in using a two-factor authentication. You need to register your mobile phone first if it has not been done and install the app on it “EU login”.

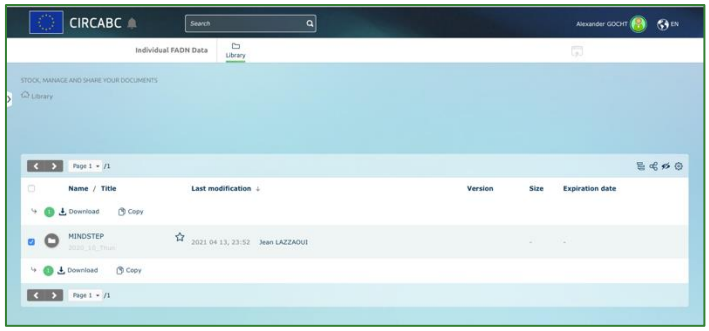


4. With your username (usually your E-mail address) your password and together with the EU log in app on your mobile phone you should enter the portal. In case no mobile phone was registered use the following link:

https://ecas.ec.europa.eu/cas/login?loginRequestId=ECAS_LR-107656466-53O1QRzkzaZu8zzYlfj7MwNwZzd4C9ATAztfGuawG4VkCs3bYd9sOEq0n7bevFyILWAUKK6fhIhgtYeE0WUqLb9m-yntOf97TTHqpWj1k3JizzhW-HRUgl720hAoYGwzLNszRPeRfiTyeAge4MhzZqEFHnG9b2fsW3gUzO1IcSQfS31IF4cxQ3kd5RShGVzzZcqbLeh8G to register it.

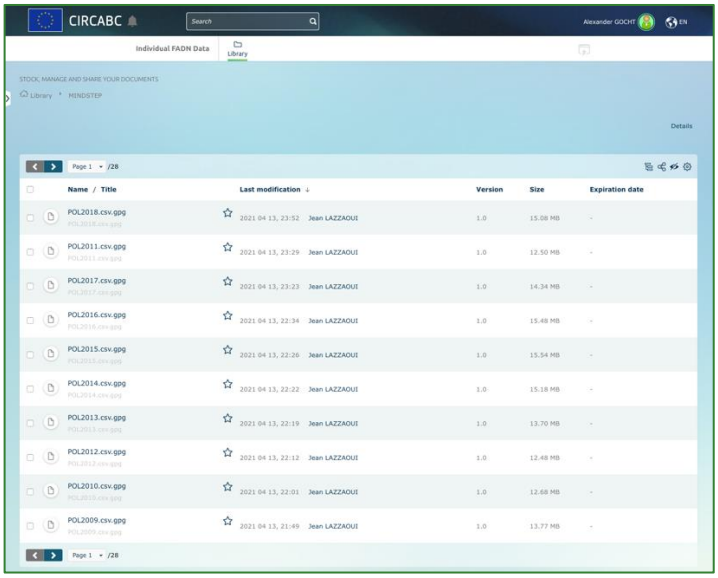


5. Download the data by clicking on library



This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 817566.

6. The bulk download for all data is not working as the size is too big. You need to download 28 smaller bulk downloads per page!



7. After download please unzip the files on the computer you created the public key.
8. Open Kleopatra and move all downloaded gpg files into the program.
9. Enter the password of your public key and decode the data
10. Store the csv files for further processing.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

MIND STEP WP2 TEAM

Mr Alexander Gocht (THÜNEN)

Mr Sebastian Neuenfeldt (THÜNEN)

Ms Xinxin Yang (THÜNEN)

Mr Hugo Storm (UBO)

Mr John Helming (WR)

Mr Marc Müller (WR)

Ms Diti Oudendag (WR)

Mr Sander Janssen (WR)

Mr Gerbert Roerink (WR)

Mr Albert Brouwer (IIASA)

Dimitrios Kremmydas (JRC)

Consortium description

The consortium of MIND STEP consists of 11 partners from 7 countries in Europe (the Netherlands, Germany, Austria (IIASA), Italy, France, Spain (JRC-Seville), Norway and Hungary). It includes partners from the private and public sector representing:

- Academia and higher education (UBO, UCSC, WU).
- SME dealing with research consultancy, data collection, strategic advice, normalization and policy in the field of energy, environment and sustainable development. This SME has also a strong track record in the field of communication, stakeholder engagement and exploitation (GEO)
- Public government bodies dealing with agricultural and environmental research and data collection and building agricultural models at different scales (WR, IIASA, IAMO, THÜNEN, INRA, NIBIO, JRC)

The consortium has been carefully constructed in such a way that it is capable of jointly managing all activities and risks involved in all project stages. Each partner contributes its own particular skills, (inter) nationally wide network and expertise, and has a critical role in MIND STEP. Partner expertise smoothly complements each other and all together form the full set of capabilities necessary to lead MIND STEP to a success. Achieving the overall objective is determined by all partners in the consortium as well as their ability to involve other interested stakeholders in the process of developing, validating and disseminating the IDM models, indicators and methodologies (WR, UBO, IAMO, UCSC, WU, THÜNEN and INRA) and linking IDM models to current agricultural policy models (WR, IIASA, UBO) included in the MIND STEP model toolbox. Dissemination and communication activities are steered by partner GEO who has graphic design, IT and marketing communication teams to deliver out-of-the-box and novel solutions for dissemination and communication and JRC who has a large network with policy makers. GEO has experience in leading comparable activities in H2020 projects as UNISECO and COASTAL. The coordinator WR is part of Stichting Wageningen Research (Wageningen Research Foundation, WR). WR consists of a number specialised institutes for applied research in the domain of healthy food and living environment. WR collaborates with Wageningen University (WU) under the external brand name Wageningen University & Research. One of the strengths of Wageningen University & Research (including WR) is that its structure facilitates and encourages close cooperation between different disciplines. The institutes Wageningen Economic Research (proposed coordinator

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

of MIND STEP, WR) and Wageningen Environmental Research (WR) are involved in this proposal. The One-Wageningen approach will also be applied to MIND STEP. WR has a long-standing reputation of leading large scale EU projects, such as SUPREMA, Foodsecure, SUSFANS, FLINT, SAT-BBE, and SIM4NEXUS.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.