
ARC-Finder – A simple, locally-deployed tool to find your peer's research data

Projektmodul (Modul 9) des Zertifikatskurs FDM (15.07)
2021 / 2022

Dominik Brilhaus,
<https://orcid.org/0000-0001-9021-3197>

2022-06-09

Contents

Introduction	3
Motivation	3
DataPLANT and the Annotated Research Context	3
Approach	4
The ARC-Finder workflow	4
The ARC-Finder GUI	5
SQLite database as alternative output	5
Data safety	5
Technical back-end	6
Discussion	6
Caveats and places for future improvements	6
isa.investigation.xlsx	6
group-associated ARCs	7
branches	7
efficiency	7
Scalability	7
Supplemental Material	8
Availability	8
Dependencies	8
Software	8
R libraries	8
Platform	9
Checks and tests	9
Deviation from the original concept	9
Scripts	10
arcFinder.sh	10
README.md	13
ARC-Finder – A simple, locally-deployed tool to find your peer’s research data	13
scripts/01_install_dependencies.R	14
scripts/01_restore_dependencies.R	14
scripts/02_read_from_gitlab.sh	14
scripts/scripts/03_parse_isaInvxlsx.R	17
scripts/scripts/03_pull_together.R	20

scripts/scripts/05_pull_together_sql.R	21
scripts/scripts/04_searchApp/app.R	21

Introduction

Motivation

Research is a highly collaborative endeavor that builds on the synergistic interaction between different stakeholders enabled by efficient knowledge exchange. Gaining a prompt overview of the ongoing research efforts – both pre- and post-publication – is oftentimes hindered for social, legal or technical reasons. This often holds true even between parties of spatially closest and well trusted surroundings of a collaborative consortium such as the Cluster of Excellence on Plant Sciences (CEPLAS¹). The key to enable discussion on and exchange of research data is *findability*, the first layer of the FAIR principles² of data stewardship (Wilkinson *et al.*, 2016³). The project presented here aims to address this layer, by making CEPLAS research easily findable and visible amongst CEPLAS researchers and showcase the beauty and ease of data sharing to spike fruitful collaborations with peers.

DataPLANT and the Annotated Research Context

Research data management (RDM) within CEPLAS is closely aligned with DataPLANT⁴, the NFDI⁵ consortium for plant sciences. At the heart of DataPLANT’s RDM strategy lies the Annotated Research Context (ARC⁶), a directory structure that packages research data together with associated metadata and computational workflows into self-sustained research objects. Annotation of research data in the ARC is based on the metadata schema ISA⁷ (for investigation – study – assay). Serialized in spread sheet format as *ISA-tab* this enables intuitive, flexible and yet structured and conclusive metadata annotation of the versatile data types produced in plant sciences. ARCs are git⁸ repositories that can be shared via DataPLANT’s DataHUB⁹, a customized GitLab¹⁰ instance with a federated authentication interface to allow controlled access across institute borders.

Although the ARC environment is continuously being developed, the choice of these key technical pillars are set: (a) ARC as the structure, (b) ISA as the metadata language, (c) git as version control logic and (d) gitlab for ARC collaboration and user management. This allows to leverage the ARC environment and develop (intermediate) solutions for data findability, knowing that time and efforts are well-invested, since both (meta)data ingest into as well as secondary outputs dependent on the

¹CEPLAS, <https://ceplas.eu>

²GO-FAIR, <https://www.go-fair.org/fair-principles/>

³Wilkinson, M., Dumontier, M., Aalbersberg, I. et al. The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* 3, 160018 (2016). <https://doi.org/10.1038/sdata.2016.18>

⁴DataPLANT, <https://nfdi4plants.de>

⁵Nationale Forschungsdaten Infrastruktur, <https://www.nfdi.de/>

⁶ARC specifications, <https://github.com/nfdi4plants/ARC-specification/>

⁷ISA Metadata Schema, <https://isa-tools.org/>

⁸Git, <https://git-scm.com/>

⁹DataPLANT DataHUB, <https://git.nfdi4plants.org>

¹⁰GitLab, <https://gitlab.com>

ARC will be adoptable and migratable in the future.

While (contents of the) ARCs can be searched via standard GitLab-implemented mechanisms within the DataHUB or via standard routines on a user’s system where the ARCs are locally cloned and stored, a structured and user-friendly search interface tailored to metadata stored in multiple ARCs is currently unavailable. With the ARC-Finder presented here, I seek to close this gap.

Approach

The ARC-Finder workflow

The ARC-Finder employs three concerted, but independent modules of metadata retrieval, restructure, and representation (Fig. 1).

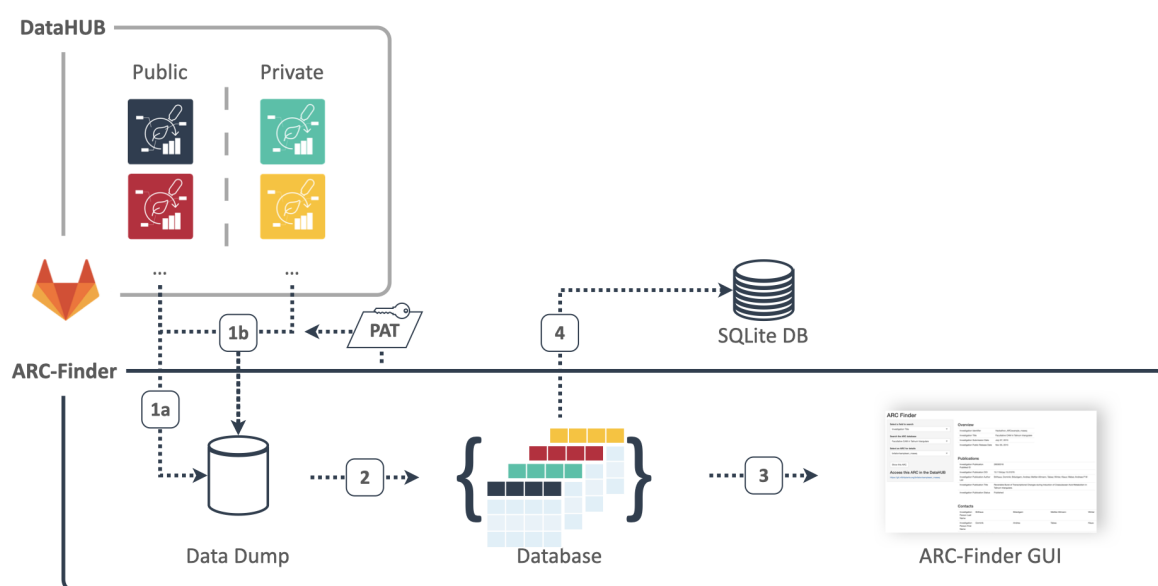


Figure 1: ARC-Finder Workflow. Depending whether the user provided a gitlab personal access token (PAT), the ARC-Finder retrieves publicly (1a) or publicly and privately (1b) accessible metadata from the DataHUB and stores it in a local data dump. The metadata is re-structured into a searchable database (2) and fed into the ARC-Finder GUI (3, details see Fig. 2) as well as provided as an SQLite database (DB) (4).

Depending on user-choice and gitlab personal access token (PAT) either publicly available (1a) or public plus privately shared (1b) ARCs are read and stored in a local data dump (see also user instructions in

README.md).

From the ISA investigation workbooks ([isa.investigation.xlsx](#)) stored at the roots of every ARC. The CEPLAS-ARC-Finder selectively retrieves, downloads and dumps the metadata locally on the user’s machine. The CEPLAS-ARC-Finder then restructures the metadata into a simple spreadsheet-based database. From the database the investigation data is finally read and represented by a user interface that enables finding the data available to the individual user.

The ARC-Finder GUI

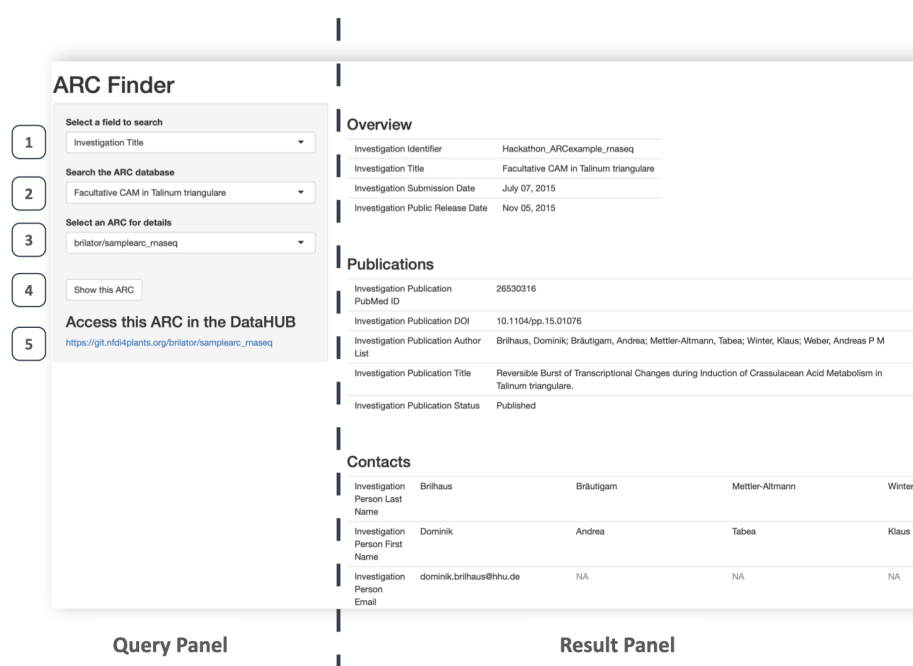


Figure 2: ARC-Finder GUI.

SQLite database as alternative output

Data safety

This project focuses on metadata at the highest project and least sensitive (i.e. ISA’s “investigation”) level to minimize user input or possible discomfort with data sharing - outsourced to DataHUB

Here, access to the ARCs can be controlled to share them publicly or with invited collaborators.

Technical back-end

The technical back-end of the ARC-Finder is a combination of shell and R scripts and leveraging on the GitLab API (version 4). The idea was to rely on as few code environments as possible. The actual code work is attached in the supplemental materials (see scripts) and available online (see availability). Software dependencies are listed in the supplemental materials (see dependencies).

Discussion

- The arcFinder provides a comparably easy approach
- Can immediately be used
 - I can add my ARC to the DataHUB and have it directly listed by the arcFinder
- One of the major hurdles in advocating FAIR data stewardship to the users is the continuously changing plethora of platforms, tools and schema.
- Integrable with future developments to avoid user friction
- Extensibility (within DataPLANT: beauty of standards)
 - Designed on purpose limited. User.
 - Show-case how use of standards can boost sustainable research data management.
 - Vehicle for communication
 - While most tools in DataPLANT are based on other programming languages...
- SQL Alternative

Caveats and places for future improvements

isa.investigation.xlsx

- read from isa.json rather than isa.investigation.xlsx
- Rationale: yet another detour / dependency to produce isa.json
- direct user-input to isa.investigation.xlsx can be read immediately
- xlsx can become big and needs to be dumped
- json could be read on-the-fly

group-associated ARCs

- are currently excluded

branches

- reading only from default git branch `main` (not e.g. master or others)

efficiency

- tool dumps, pulls freshly every time it is called
 - nothing memorized and updated
 - by design sqlite db is always overwritten -> data could be appended
- selectivity
 - not all ARCs, but just selection (e.g. group)
 - error-prone: non-clean ARCs

Scalability

- CI / CD, public access / deployment

Supplemental Material

Availability

The ARC-Finder is available for download at <https://github.com/Brilator/arcFinder>.

Dependencies

Software

Table 1: Software used during development, testing and writing.

Software	Version	Platform
GNU bash	3.2.57(1)-release	x86_64-apple-darwin21
curl	7.79.1	x86_64-apple-darwin21.0
R	4.2.0	x86_64-apple-darwin17.0
RStudio	2022.02.2 Build 485	-
Visual Studio Code	1.67.2	-
Codes Spell Checker (VS Code Extension)	2.03	-
pandoc	2.18	-
TeX Live 2022	MacTeX-2022	-

R libraries

To provide best reproducibility, R package dependencies are handled via [renv](#)¹¹ (version 0.15.3) and stored in the root file “renv.lock”. In the first step of [arcFinder](#), the virtual environment is automatically restored, including installation of all required dependencies. Depending on the local setup (installation of R and packages), this may take some time. However, [renv](#) prevents interference with the local setup, thus keeping the system intact.

Table 2: R packages specifically loaded for individual R scripts

Package (version)	Main purpose	Used in script(s)
renv_0.15.4	Manage R package dependencies	01_install_dependencies.R 01_restore_dependencies.R

¹¹R package “renv”, <https://rstudio.github.io/renv/>

Table 2: R packages specifically loaded for individual R scripts

Package (version)	Main purpose	Used in script(s)
readxl_1.4.0 (part of ‘tidyverse’)	Read data from Microsoft Excel workbooks	03_parse_isaInvxlsx.R
tidyverse_1.3.1	Tidy data into a useful format	04_searchApp/app.R
shiny_1.7.1	Prepare and launch a shiny app	04_searchApp/app.R
DBI_1.1.2	Write data to an ‘*.sqlite’ object	05_pull_together_sql.R

Platform

The DataPLANT’s DataHUB¹² is a customized instance of GitLab¹³, currently running under version 14.10.2, hosted and maintained by the DataPLANT node at Albert-Ludwigs-University Freiburg. Data is retrieved from the DataHUB via GitLab API version 4.

After registration¹⁴ with DataPLANT, users can share and access non-public ARCs via the DataHUB. As explained in the [arcFinder](#)’s README, a GitLab private access token (PAT) needs to be generated within the DataHUB and provided to [arcFinder](#).

Checks and tests

Currently tested only under macOS Monterey 12.3.1 (x86_64-apple-darwin17.0, 64-bit) with software versions specified under Dependencies.

Deviation from the original concept

The originally proposed concept targeted an automated workflow for easier metadata-ingestion from previously published manuscripts into an the ISA model of an ARC. As this workflow (a) targets a completely other “side” of the ARC and DataHUB environment and thus (b) comes with multiple additional and more complicated dependencies, it was omitted from the ARC-Finder presented here.

¹²DataPLANT DataHUB, <https://git.nfdi4plants.org>

¹³GitLab, <https://gitlab.com>

¹⁴DataPLANT registration, <https://register.nfdi4plants.org/>

Scripts

arcFinder.sh

```

1 #####
2 ### Create root folder for temporary data
3 #####
4
5 ### If exists, remove and create fresh.
6 ### This is to prevent data piling.
7 ### TODO Should probably be replaced with safer / better logic for
  debugging. TODO
8
9 if [ -d ".tmp/" ]; then
10     rm -r .tmp/
11     mkdir .tmp/
12 else
13     mkdir .tmp/
14 fi
15
16 #####
17 ### Restore env session
18 #####
19
20 echo "### Restore virtual environment"
21 echo "-----"
22
23 Rscript ./scripts/01_restore_dependencies.R 2>&1 >> .tmp/01.log
24
25 #####
26 ### Read GitLab personal access token (PAT)
27 #####
28
29 ### Read GitLab PAT from -p flag
30
31 while getopts p: flag
32 do
33     case "${flag}" in
34         p) gitlab_pat=${OPTARG};;
35         esac
36     done
37
38 ### Check if argument supplied with `-p` is a file.
39 ### If yes, read that file.
40 ### If not, use the input (PAT as a string) directly
41
42 if [ -f "$gitlab_pat" ]; then
43     echo "Using GitLab token stored in '$gitlab_pat'."
44     gitlab_pat=$(< $gitlab_pat)
45 fi

```

```

46
47 ### check if string is empty
48
49 [ -z "$gitlab_pat" ] && printf "No GitLab token supplied or GitLab
    token is empty. \nReading from public ARCs only.\n"
50
51 #####
52 ### Run gitlab reader
53 #####
54
55 echo "-----"
56 echo "### Step 01: Downloading metadata of available ARCs from the
    DataHUB."
57 echo "-----"
58
59 echo "log of 02_read_from_gitlab.sh" > .tmp/02.log
60 bash ./scripts/02_read_from_gitlab.sh -p "${gitlab_pat}" 2>&1 >> .tmp
    /02.log
61
62 #####
63 ### Run xlsx parser
64 #####
65
66 ## store paths of isa.investigation.xlsx files into variable
67 ## while loop
68 ## - extract arc id from part of path
69 ## - run script with arc id and path
70
71 echo "### Step 02: Structuring ARC metadata."
72 echo "-----"
73 echo "log of 03_parse_isaInvxlsx.R" > .tmp/03.log
74
75 invs=$(find .tmp/02_investigations -name '*.xlsx' | sort -n)
76 echo "$invs" | while IFS= read -r current_inv_path;
77 do
78     arc_id=$(echo $current_inv_path | cut -d/ -f3 | cut -d"_" -f1)
79
80     Rscript ./scripts/03_parse_isaInvxlsx.R "$arc_id" $current_inv_path
        2>&1 >> .tmp/03.log
81
82 done
83
84
85 #####
86 ### Pull together data
87 #####
88
89 echo "### Step 03: Building a searchable database of ARC metadata"
90 echo "-----"
91
92 Rscript ./scripts/03_pull_together.R 2>&1 >> .tmp/03.log

```

```
93
94 #####
95 ### Optional: Prepare SQLite database
96 #####
97
98 Rscript ./scripts/05_pull_together_sql.R 2>&1 >> .tmp/05.log
99
100 #####
101 ### Run the search APP
102 #####
103
104 echo "### Voila: The ARC Finder is running in your default browser."
105 echo "### Close the browser window or tab to shut down the app."
106
107 Rscript -e 'load(".tmp/03_allARCs.RData"); shiny::runApp("./scripts/04
    _searchApp/app.R", launch.browser = TRUE)' 2>&1 >> .tmp/04.log
```

README.md

ARC-Finder – A simple, locally-deployed tool to find your peer’s research data

This is a tool to help you find metadata about ARCs stored in the DataPLANT DataHUB. Visit the DataPLANT website for more information about ARCs (annotated research contexts).

Usage

- Git clone or download this repository.
- Open a command line or terminal and navigate to the `arcFinder` directory.
- Run one of the following two options:

Option 1: Search public ARCs only

```
1 ./arcFinder.sh
```

Option 2: Search Public + privately shared ARCs

Note: Replace `<gitlab pat>` with the path pointing to a file which stores a GitLab personal access token (PAT).

```
1 ./arcFinder.sh -p <gitlab pat>
```

Registration with DataPLANT In order to use the `<gitlab pat>` option, please follow these steps:

1. Sign up with DataPLANT.
2. Generate a personal access token in the DataHUB PAT settings
 - Provide a “Token name”, e.g. `arcFinder`
 - Select either option “api” or “read_api” and click “Create personal access token”
 - Copy the generated token on top of the page.
3. Paste the bare token into a text file and save it (e.g. `gitlab_token` stored in the root of this directory)
4. Supply the file path to `arcFinder`, e.g.:

```
1 ./arcFinder.sh -p gitlab_token
```

scripts/01_install_dependencies.R

```
1 #####
2 ### Script to install all R dependencies
3 #####
4
5
6 if(!require(tidyverse, quietly = TRUE)){install.packages("tidyverse")}
7 if(!require(DBI, quietly = TRUE)){install.packages("DBI")}
8 if(!require(shiny, quietly = TRUE)){install.packages("shiny")}
9 if(!require(renv, quietly = TRUE)){install.packages("renv")}
10
11 renv::init(bare = T)
12
13 # save library state to lockfile
14 renv::snapshot()
15
16 renv::status()
```

scripts/01_restore_dependencies.R

```
1
2 #####
3 ### Restore R virtual environment via renv
4 #####
5
6 # restore lockfile, thereby installing dependencies from renv.lock
7
8 renv::restore()
```

scripts/02_read_from_gitlab.sh

```
1 #!/usr/bin/env bash
2
3 #####
4 ### Script to skim GitLab for accessible ARCs and
5 ### retrieve their isa.investigation.xlsx's
6 #####
7
8 ### Goal
9 # 1. read gitlab pat
10 # 2. Store a list of all available projects (e.g. tab)
11 # 3. Iterate over project trees
12 #   - check for isa.investigation file
13 #   - present: download raw and dump to temp
14 #   - absent: leave loop
```

```
15
16 #####
17 ### Read GitLab token (PAT)
18 #####
19
20 ### Read GitLab PAT from -p flag
21
22 while getopts p: flag
23 do
24     case "${flag}" in
25         p) gitlab_pat=${OPTARG};;
26     esac
27 done
28
29 # ### Check if argument supplied with `-p` is a file.
30 # ### If yes, read that file.
31 # ### If not, use the input (PAT as a string) directly
32
33 # if [ -f "$gitlab_pat" ]; then
34 #     echo "Using GitLab token stored in '$gitlab_pat'."
35 #     gitlab_pat=$(< $gitlab_pat)
36 # else
37 #     echo "Using supplied GitLab token"
38 #     # This would be gitlab_pat=$gitlab_pat    ### TODO: probably safer
39 #     # to change this
40 # fi
41
42 # ### check if string is empty
43 # [ -z "$gitlab_pat" ] && printf "No GitLab token supplied or GitLab
44 # token is empty. \nReading from public ARCs only.\n"
45
46 #####
47 ### List available ARCs
48 #####
49
50 # Writing to json first
51 curl --silent --request GET --header "PRIVATE-TOKEN: $gitlab_pat" "
52     https://git.nfdi4plants.org/api/v4/projects/" > .tmp/02
53     _arcs_available.json
54
55 # grepping project IDs
56 grep -oE '"id":[0-9]{1,},"description"' .tmp/02_arcs_available.json |
57     grep -oE '[0-9]{1,}' > .tmp/02_arcs_ids
58
59 # Could be piped directly (without the temporary .json)
60 # But will keep the json, for trouble-shooting
61 # curl --request GET --header "PRIVATE-TOKEN: $gitlab_pat" "https://git
62 # .nfdi4plants.org/api/v4/projects/" | grep -oE '"id":[0-9]{1,},"
63 # description"' | grep -oE '[0-9]{1,}' > projects_list
```



```
59
60
61 #####
62 ### Iterate over ARCS
63 #####
64
65 ### create a dump directory for the isa.investigation.xlsx files
66 if ! [ -d ".tmp/02_investigations" ]; then mkdir ".tmp/02
    _investigations"; fi
67
68 ### write a table to collect ARC id and path with namespace
69 printf "ARC id\tARC path\tcomment" > .tmp/02_investigations/arc_list.
    tsv
70
71 all_arc_IDs=$(< .tmp/02_arcs_ids)
72 echo "$all_arc_IDs" | while IFS= read -r arc_id;
73 do
74     # echo $arc_id
75
76     ### get project info
77     curl --silent --header "PRIVATE-TOKEN: $gitlab_pat" "https://git.
        nfdi4plants.org/api/v4/projects/$arc_id" > .tmp/02
        _current_arc_info.json
78
79     ### extract git path with namespace
80     arc_path=$(grep -oE '"path_with_namespace": ".*", "created_at' .tmp/02
        _current_arc_info.json | cut -d'"' -f 4)
81
82     echo $arc_path
83
84     ### get project tree
85     curl --silent --header "PRIVATE-TOKEN: $gitlab_pat" "https://git.
        nfdi4plants.org/api/v4/projects/$arc_id/repository/tree" > .tmp/02
        _current_arc_tree.json
86
87     ### check that file `isa.investigation.xlsx` exists at ARC root
88     ### if yes: download and dump
89     ### if no: error message
90
91     inv_path=$(grep -oE '"path": "isa.investigation.xlsx",' .tmp/02
        _current_arc_tree.json)
92
93     ### check if variable is empty
94     if [ -z "$inv_path" ]
95     then
96         printf "Missing 'isa.investigation.xlsx' at the root of $arc_path\n
            "
97         printf "\n$arc_id\t$arc_path\tisa.investigation.xlsx missing" >> .
            tmp/02_investigations/arc_list.tsv
98     else
```

```

99     curl -L --silent --request GET --header "PRIVATE-TOKEN: $gitlab_pat
      " "https://git.nfdi4plants.org/api/v4/projects/$arc_id/
      repository/files/isa%2Einvestigation%2Exlsx/raw?ref=main" -o .
      tmp/02_investigations/$arc_id'_isa.inv.xlsx'
100     printf "\n$arc_id\t$arc_path\tisa.investigation.xlsx detected" >> .
      tmp/02_investigations/arc_list.tsv
101     fi
102
103     rm .tmp/02_current_arc*
104
105 done

```

scripts/scripts/03_parse_isaInvxlsx.R

```

1
2 #####
3 ### Script to read metadata from an isa.investigation.xlsx
4 #####
5
6 ## rough idea:
7
8 # 0. Takes two arguments as CLI input: <ARC id> and <isa.investigation.
   xlsx>
9 # 1. Check whether its an investigation sheet or loop over sheets
10 # 2. focus on investigation only
11 # 3. subset into investigation sections
12 # 4. Store JSON-like as (nested) lists
13 #   - column 1 = keys
14 #   - column(s) 2:n = values as a list
15 # 5. put out as RData for further processing
16
17
18 #####
19 ### Setup
20 #####
21
22 ### If package "readxl" is not installed, install it.
23 # if(!require("readxl", quietly = TRUE)){install.packages("readxl")}
24
25 ### load the package
26 library(readxl)
27
28
29 #####
30 ### Inputs
31 #####
32
33 args = commandArgs(trailingOnly=TRUE)
34

```

```

35 # test if arguments are supplied: if not, return an error
36 if (length(args)!=2) {
37
38     stop("<ARC id> and <isa.investigation.xlsx> must be supplied as
        arguments", call.=FALSE)
39
40 } else if (length(args)==2) {
41
42     # default output file
43     isa_inv_wb <- args[2]
44     print(paste("Reading file", isa_inv_wb))
45     arc_id <- args[1]
46 }
47
48 #####
49 ### read data from excel
50 #####
51
52 ### loop over sheets of workbook
53 for(sheet in excel_sheets(isa_inv_wb)){
54
55     ### read sheet
56     current_sheet <- as.data.frame(read_xlsx(isa_inv_wb, col_names = F,
        sheet = sheet, .name_repair = "minimal"))
57
58     ### Simple sanity check for ISA investigation format
59
60     if(current_sheet[1, 1] == "ONTOLOGY SOURCE REFERENCE" & current_sheet
        [2, 1] == "Term Source Name"){
61
62         print(paste("Reading from excel sheet", sheet))
63         invdata <- current_sheet
64
65     }else{
66         print(paste("Excel sheet", sheet, "is not in ISA investigation
            format"))
67         invdata <- NULL
68     }
69
70 }
71
72 ### Stop if no proper ISA sheet detected
73 if(is.null(invdata)){stop(simpleError("No valid ISA investigation sheet
        detected"))}
74
75
76 #####
77 ### wrangle / extract only relevant data
78 #####
79
80

```

```

81 ### subset to investigation only (excluding study, assay layers)
82 investigation_data <- invdata[grepl("^investigation", invdata[, 1],
83   ignore.case = T), ]
84
85 ### first column as row names
86 rownames(investigation_data) <- investigation_data[,1]
87 investigation_data2 <- investigation_data[,-1, drop = F]
88
89 ### extract investigation subsections (some redundancy with above)
90 inv_sections <- which(grepl("^INVESTIGATION", row.names(investigation_
91   data2), ignore.case = F))
92
93 investigation_list <- list()
94 for(i in 1:length(inv_sections))
95 {
96   if(i == length(inv_sections))
97   {
98     section_range <- (inv_sections[i] + 1):nrow(investigation_data2)
99   }else{
100     section_range <- (inv_sections[i] + 1):(inv_sections[i+1] - 1)
101   }
102
103   current_section <- investigation_data2[section_range, , drop =F]
104
105   ### remove columns that are only NA
106   current_section <- current_section[, apply(current_section, 2,
107     function(x){sum(is.na(x)) != nrow(current_section)}), drop = F]
108
109   ### transpose / pivot data to transform into list
110   current_section_transposed <- as.data.frame(t(current_section))
111   rownames(current_section_transposed) <- NULL
112
113   # TODO stupid work-around to circumvent the bug with a section having
114   # only NAs
115   if(nrow(current_section_transposed) == 0){current_section_transposed
116     [1, ] = NA}
117
118   current_section_transposed$arc_id <- arc_id
119
120   ### extract current section name
121   current_section_name <- row.names(investigation_data2)[inv_sections[i]]
122
123   # ### transform to named list, omitting NAs
124   # investigation_list[[current_section_name]] <- lapply(current_
125     section_transposed, function(v){v[!is.na(v)]})
126
127   #
128   # stack(current_section_transposed)

```

```

124     investigation_list[[current_section_name]] <- current_section_
        transposed
125
126 }
127
128 #####
129 ### output to .RData
130 #####
131
132 if(!dir.exists(".tmp/03_rdata_dumps/")){dir.create(".tmp/03_rdata_dumps
    /")}
133
134 print(paste0("Storing outputs in: .tmp/03_rdata_dumps/", arc_id, ".
    Rdata"))
135
136 save(investigation_list, isa_inv_wb, arc_id, file = paste0(".tmp/03_
    rdata_dumps/", arc_id, ".RData"))

```

scripts/scripts/03_pull_together.R

```

1 #####
2 ### Script to pull together output of previous scripts
3 #####
4
5 ### for loop over available RData dumps from 03_parse_isaInvxlsx.R
6
7 all_arcs <- list()
8
9 for(i in dir(".tmp/03_rdata_dumps/", full.names = T, pattern = ".RData"
    ))
10 {
11     ### load the data
12     load(i)
13
14     ### store in named list
15
16     all_arcs[[arc_id]] <- investigation_list
17 }
18
19
20 ### row-bind the second-level (i.e. INVESTIGATION "sections") of the
    lists, respectively
21
22 all_arcs_db <- do.call(Map, c(f = rbind, all_arcs))
23
24 ### read the arc_list (translating the ARC id to the ARC path) produced
    by 02_read_from_gitlab.sh
25 ### and append to above list
26

```

```
27 arc_list <- read.table(".tmp/02_investigations/arc_list.tsv", sep = "\t", header = T)
28 colnames(arc_list)[1] <- "arc_id"
29
30 # all_arcs_db[["arc_list"]] <- arc_list
31
32 ### store the output as RData
33
34 save(all_arcs, all_arcs_db, arc_list, file = ".tmp/03_allARCs.RData")
```

scripts/scripts/05_pull_together_sql.R

```
1 #####
2 ### Convert data into SQLite database
3 #####
4
5 # if(!require("DBI", quietly = TRUE)){install.packages("DBI")}
6 library(DBI)
7
8 load(".tmp/03_allARCs.RData")
9
10 ### Write into an SQLite DB file
11
12 mydb <- dbConnect(RSQLite::SQLite(), "yourARCs_database.sqlite")
13
14 for(i in names(all_arcs_db))
15 {
16   dbWriteTable(mydb, i, all_arcs_db[[i]], overwrite = T)
17 }
18
19 dbWriteTable(mydb, "ARC list", arc_list, overwrite = T)
20
21 dbListTables(mydb)
22 dbDisconnect(mydb)
```

scripts/scripts/04_searchApp/app.R

```
1 #####
2 ### The ARC-Finder GUI Shiny App
3 #####
4
5 suppressMessages(library(tidyverse))
6 suppressMessages(library(shiny))
7
8 # load(".tmp/03_allARCs.RData")
9
```

```
10  ### Flatten ALL values into a 3-column (arc_id | key | value) df to
    provide search across "any field"
11
12  all_values <- do.call(
13    rbind.data.frame, lapply(all_arcs_db, function(x){
14      pivot_longer(x, cols = setdiff(colnames(x), "arc_id"), values_drop_na
        = T)}))
15
16  all_values <- as.data.frame(all_values)
17
18  arc_list <- unique(arc_list)
19
20  shinyApp(
21    ui = pageWithSidebar(
22      headerPanel("ARC Finder"),
23      sidebarPanel(
24        selectizeInput('search_key', 'Select a field to search',
          choices = c("Any field", unique(all_values$name))),
25        uiOutput("search_field"),
26        selectInput(inputId = "arc_path", label = "Select an ARC
          for details", choices = NULL),
27        br(),
28        actionButton("go", "Show this ARC"),
29        h3("Access this ARC in the DataHUB"),
30        uiOutput("arc_gitlab"),
31      ),
32    ),
33
34    mainPanel(
35      h3("Overview"),
36      tableOutput("table_INV"),
37      br(),
38      h3("Publications"),
39      tableOutput("table_INV_PUBS"),
40      br(),
41      h3("Contacts"),
42      tableOutput("table_INV_Contacts")
43    )
44  ),
45
46  server = function(input, output, session) {
47
48
49    ##### reactive input field for text-search
50
51
52    output$search_field <- renderUI({
53
54      # check whether user wants to filter by cyl;
55      # if not, then filter by selection
56      if ('Any field' %in% input$search_key) {
```

```
57     df <- all_values
58   } else {
59     df <- subset(all_values, name == input$search_key)
60   }
61
62
63   selectizeInput('search_value', 'Search the ARC database', choices
64     = c("", sort(unique(df$value))))
65 })
66
67 ##### ARC choices (arc_id) matching user-input
68
69   arc_choices_id <- reactive({
70
71     if ('Any field' %in% input$search_key) {
72
73       subset(all_values, value == input$search_value, arc_id, drop
74         = T)
75
76     } else {
77
78       subset(all_values, name == input$search_key & value == input$
79         search_value, arc_id, drop = T)
80
81     }
82   })
83
84   ### retrieve path for matching ARCs from arc list
85
86   arc_choices_path <- reactive({
87
88     subset(arc_list, arc_id %in% arc_choices_id(), ARC.path, drop =
89       T)
90
91   })
92
93   ##### reactive ARC selection: updated Input to let user pick from
94
95   observe({
96     updateSelectInput(session = session, inputId = "arc_path",
97       choices = arc_choices_path())
98   })
99
100   ##### User's ARC choice
101
102   selected_arc <- eventReactive(input$go, {
```



```

103
104     all_arcs[[as.character(subset(arc_list, ARC.path %in% input$arc
105                               _path, arc_id, drop = T))]]
106   })
107
108
109   ##### render table INVESTIGATION
110
111   output$table_INV <- renderTable(colnames = F, rownames = F, {
112
113     selected_table <- selected_arc()$INVESTIGATION
114     selected_table <- selected_table[, -which(colnames(selected_
115       table) == "arc_id")]
116
117     selected_table$pivot_col <- row.names(selected_table)
118     long <- pivot_longer(selected_table, setdiff(colnames(
119       selected_table), "pivot_col"), values_drop_na = T)
120
121     pivot_wider(long, names_from = pivot_col)
122   })
123
124   ##### render table INVESTIGATION PUBLICATIONS
125
126   output$table_INV_PUBS <- renderTable(colnames = F, rownames = F
127     , {
128
129     selected_table <- selected_arc()$`INVESTIGATION PUBLICATIONS`
130     selected_table <- selected_table[, -which(colnames(selected_
131       table) == "arc_id")]
132
133     selected_table$pivot_col <- row.names(selected_table)
134     long <- pivot_longer(selected_table, setdiff(colnames(
135       selected_table), "pivot_col"), values_drop_na = T)
136
137     pivot_wider(long, names_from = pivot_col)
138   })
139
140   ##### render table INVESTIGATION CONTACTS
141
142   output$table_INV_Contacts <- renderTable(colnames = F, rownames
143     = F, {
144
145     selected_table <- selected_arc()$`INVESTIGATION CONTACTS`
146     selected_table <- selected_table[, -which(colnames(selected_
147       table) == "arc_id")]

```

```
146     selected_table$pivot_col <- row.names(selected_table)
147     long <- pivot_longer(selected_table, setdiff(colnames(
148         selected_table), "pivot_col"), values_drop_na = T)
149     pivot_wider(long, names_from = pivot_col)
150
151   })
152
153   ##### render link to gitlab
154
155     output$arc_gitlab <- renderUI(a(href = paste0('https://git.
156         nfdi4plants.org/', input$arc_path),
157         paste0('https://git.nfdi4plants
158             .org/', input$arc_path) ,
159         target="_blank"))
160
161     session$onSessionEnded(function() {
162         stopApp()
163     })
164 }
```