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

Motivation

Research is a highly collaborative endeavor that builds on 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) even between parties of spatially closest and well trusted surroundings of a collaborative consortium such as the Cluster of Excellence on Plant Sciences (CEPLAS[1]). The key to enable discussion on and exchange of research data is *findability*, the first layer of the FAIR principles of data stewardship. 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.

State of the art

Research data management within CEPLAS is closely aligned with DataPLANT[2], the NFDI consortium for plant sciences. DataPLANT has developed the Annotated Research Context (ARC[3]), a directory structure for research objects. Annotation of research data in the ARC is based on the metadata schema ISA[4] (for investigation – study – assay). Serialized in spread sheet format as *ISA-tab* this allows intuitive, flexible and yet structured and conclusive metadata annotation of the versatile data types produced in plant sciences. ARCs are git[5] repositories that can be shared via DataPLANT's DataHUB[6], a customized GitLab[7] 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 and develop at least intermediate solutions for data findability, knowing that time and efforts are well-invested, since both (meta)data inputs in as well as secondary outputs dependent on the ARC will be adoptable and migratable in the future.

[1] CEPLAS, https://ceplas.eu
[2] DataPLANT, https://nfdi4plants.de
[3] ARC specifications, https://github.com/nfdi4plants/ARC-specification/
[4] ISA Metadata Schema, https://isa-tools.org/
[5] Git, https://git-scm.com/
[6] DataPLANT DataHUB, https://git.nfdi4plants.org
[7] GitLab, https://gitlab.com

Approach

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).

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 and will be achieved in four concerted, but independent modules of metadata

1. collection,
2. retrieval,
3. restructure, and
4. representation.

First, metadata is collected – manually or supported by automation – in the ISA investigation spread sheet, packaged in ARCs and submitted to the DataHUB by individual volunteers. Here, access to the ARCs can be controlled to share them publicly or with invited collaborators. 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.

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

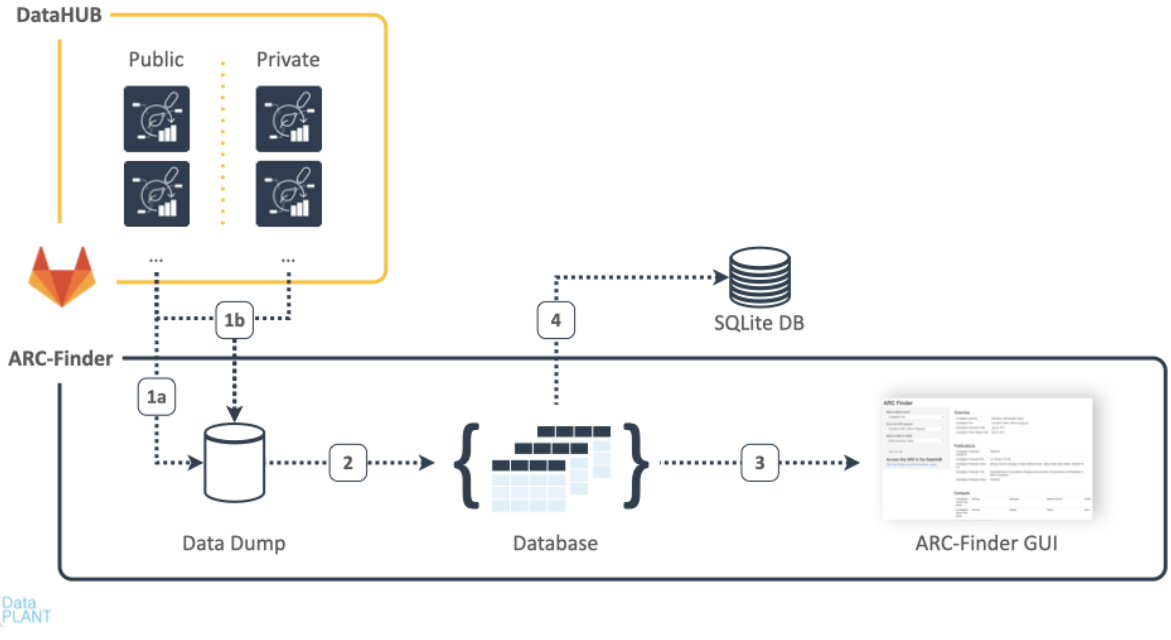

Figure 1: ARC-Finder Workflow. First, the ARC-Finder retrieves accessible data from the DataHUB (1). Depending on user-choice and provided access token either publicly available (1a) or public plus privately shared (1b) ARCs are read and stored in a local data dump.

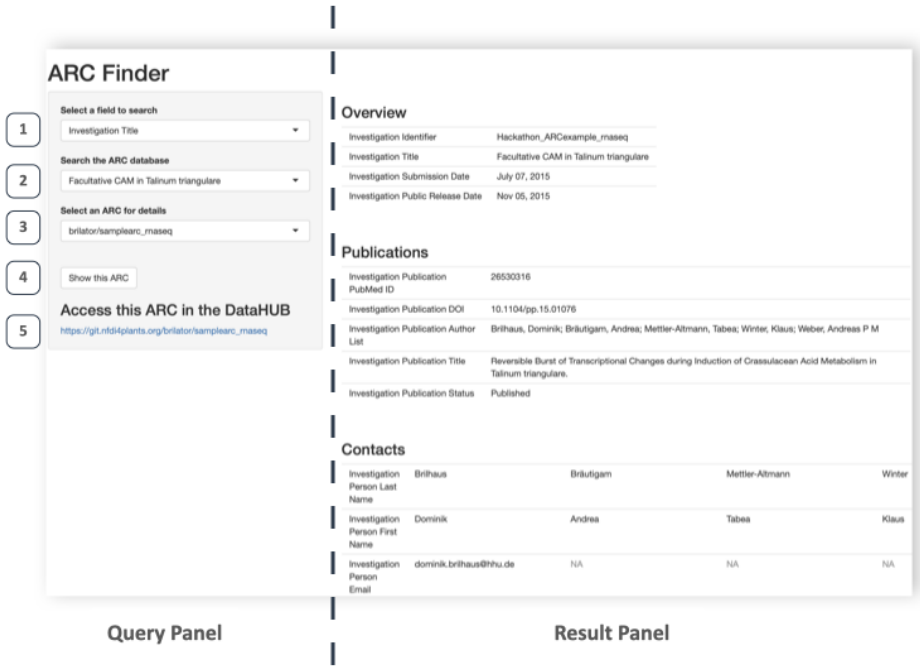

Figure 2: ARC-Finder GUI.

- 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 the `renv`[8] (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

[8] R 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 DataHUB[9] is a customized instance of GitLab[10], 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 registration[11] 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.

Scripts

arcFinder.sh

[9] DataPLANT DataHUB, https://git.nfdi4plants.org
[10] GitLab, https://gitlab.com
[11] DataPLANT registration, https://register.nfdi4plants.org/

```
#############################################################
### Create root folder for temporary data
#############################################################

### If exists, remove and create fresh.
### This is to prevent data piling.
### TODO Should probably be replaced with safer / better logic for
    debugging. TODO

if [ -d ".tmp/" ]; then
  rm -r .tmp/
  mkdir .tmp/
else
  mkdir .tmp/
fi

#############################################################
### Resotre renv session
#############################################################

echo "### Restore virtual environment"
echo "-----------------------"

Rscript ./scripts/01_restore_dependencies.R 2>&1 >> .tmp/01.log

#############################################################
### Read GitLab personal access token (PAT)
#############################################################

### Read GitLab PAT from -p flag

while getopts p: flag
do
    case "${flag}" in
        p) gitlab_pat=${OPTARG};;
    esac
done

### Check if argument supplied with `-p` is a file.
### If yes, read that file.
### If not, use the input (PAT as a string) directly

if [ -f "$gitlab_pat" ]; then
    echo "Using GitLab token stored in '$gitlab_pat'."
    gitlab_pat=$(< $gitlab_pat)
fi

### check if string is empty

```

```
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
```

```
##########################################################

Rscript ./scripts/05_pull_together_sql.R 2>&1 >> .tmp/05.log

##########################################################
### Run the search APP
##########################################################

echo "### Voila: The ARC Finder is running in your default browser."
echo "### Close the browser window or tab to shut down the app."

Rscript -e 'load(".tmp/03_allARCs.RData"); shiny::runApp("./scripts/04
    _searchApp/app.R", launch.browser = TRUE)' 2>&1 >> .tmp/04.log
```

scripts/01_install_dependencies.R

```
################################################################
### Script to install all R dependencies
################################################################


if(!require(tidyverse, quietly = TRUE)){install.packages("tidyverse")}
if(!require(DBI, quietly = TRUE)){install.packages("DBI")}
if(!require(shiny, quietly = TRUE)){install.packages("shiny")}
if(!require(renv, quietly = TRUE)){install.packages("renv")}

renv::init(bare = T)

# save library state to lockfile
renv::snapshot()

renv::status()
```

scripts/01_restore_dependencies.R

```

################################################################
### Restore R virtual environment via renv
################################################################

# restore lockfile, thereby installing dependencies from renv.lock

renv::restore()
```

scripts/02_read_from_gitlab.sh

```
#!/usr/bin/env bash

################################################################
### Script to skim GitLab for accessible ARCs and
### retreive their isa.investigation.xlsx's
################################################################

### Goal
# 1. read gitlab pat
# 2. Store a list of all available projects (e.g. tab)
# 3. Iterate over project trees
#   - check for isa.investigation file
#   - present: download raw and dump to temp
#   - absent: leave loop

############################
### Read GitLab token (PAT)
############################

### Read GitLab PAT from -p flag

while getopts p: flag
do
    case "${flag}" in
        p) gitlab_pat=${OPTARG};;
    esac
done

# ### Check if argument supplied with `-p` is a file.
# ### If yes, read that file.
# ### If not, use the input (PAT as a string) directly

# if [ -f "$gitlab_pat" ]; then
#     echo "Using GitLab token stored in '$gitlab_pat'."
#     gitlab_pat=$(< $gitlab_pat)
# else
#     echo "Using supplied GitLab token"
#     # This would be gitlab_pat=$gitlab_pat   ### TODO: probably safer to change this
# fi

# ### check if string is empty

# [ -z "$gitlab_pat" ] && printf "No GitLab token supplied or GitLab token is empty. \nReading from public ARCs only.\n"


############################
### List available ARCs
```

```
48 ############################
49
50 # Writing to json first
51 curl --silent --request GET --header "PRIVATE-TOKEN: $gitlab_pat" "
       https://git.nfdi4plants.org/api/v4/projects/" > .tmp/02
       _arcs_available.json
52
53 # grepping project IDs
54 grep -oE '"id":[0-9]{1,},"description"' .tmp/02_arcs_available.json |
       grep -oE '[0-9]{1,}' > .tmp/02_arcs_ids
55
56 # Could be piped directly (without the temporary .json)
57 # But will keep the json, for trouble-shooting
58 # curl --request GET --header "PRIVATE-TOKEN: $gitlab_pat" "https://git
       .nfdi4plants.org/api/v4/projects/" | grep -oE '"id":[0-9]{1,},"
       description"' | grep -oE '[0-9]{1,}' > projects_list
59
60
61 ############################
62 ### Iterate over ARCS
63 ############################
64
65 ### create a dump directory for the isa.investigation.xlsx files
66 if ! [ -d ".tmp/02_investigations" ]; then mkdir ".tmp/02
       _investigations"; fi
67
68 ### write a table to collect ARC id and path with namespace
69 printf "ARC id\tARC path\tcomment" > .tmp/02_investigations/arc_list.
       tsv
70
71 all_arc_IDs=$(< .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
```

```
  ### check that file `isa.investigation.xlsx` exists at ARC root
  ### if yes: download and dump
  ### if no: error message

  inv_path=$(grep -oE '"path":"isa.investigation.xlsx",' .tmp/02_current_arc_tree.json)

  ### check if variable is empty
  if [ -z "$inv_path" ]
  then
    printf "Missing 'isa.investigation.xlsx' at the root of $arc_path\n"
    printf "\n$arc_id\t$arc_path\tisa.investigation.xlsx missing" >> .tmp/02_investigations/arc_list.tsv
  else
    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/$arc_id'_isa.inv.xlsx'
    printf "\n$arc_id\t$arc_path\tisa.investigation.xlsx detected" >> .tmp/02_investigations/arc_list.tsv
  fi

  rm .tmp/02_current_arc*

done
```

scripts/scripts/03_parse_isaInvxlsx.R

```
#############################################################
### Script to read metadata from an isa.investigation.xlsx
#############################################################

## rough idea:

# 0. Takes two arguments as CLI input: <ARC id> and <isa.investigation.xlsx>
# 1. Check whether its an investigation sheet or loop over sheets
# 2. focus on investigation only
# 3. subset into investigation sections
# 4. Store JSON-like as (nested) lists
#   - column 1 = keys
#   - column(s) 2:n = values as a list
# 5. put out as RData for further processing


########################
### Setup
```

```
########################

### If package "readxl" is not installed, install it.
# if(!require("readxl", quietly = TRUE)){install.packages("readxl")}

### load the package
library(readxl)


########################
### Inputs
########################

args = commandArgs(trailingOnly=TRUE)

# test if arguments are supplied: if not, return an error
if (length(args)!=2) {

  stop("<ARC id> and <isa.investigation.xlsx> must be supplied as
     arguments", call.=FALSE)

  } else if (length(args)==2) {

  # default output file
  isa_inv_wb <- args[2]
  print(paste("Reading file", isa_inv_wb))
  arc_id <- args[1]
}

########################
### read data from excel
########################

### loop over sheets of workbook
for(sheet in excel_sheets(isa_inv_wb)){

  ### read sheet
  current_sheet <- as.data.frame(read_xlsx(isa_inv_wb, col_names = F,
     sheet = sheet, .name_repair = "minimal"))

  ### Simple sanity check for ISA investigation format

  if(current_sheet[1, 1] == "ONTOLOGY SOURCE REFERENCE" & current_sheet
     [2, 1] == "Term Source Name"){

    print(paste("Reading from excel sheet", sheet))
    invdata <- current_sheet

  }else{
    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],
       ignore.case = T), ]
83
84  ### first column as row names
85  rownames(investigation_data) <- investigation_data[,1]
86  investigation_data2 <- investigation_data[,-1, drop = F]
87
88  ### extract investigation subsections (some redundancy with above)
89  inv_sections <- which(grepl("^INVESTIGATION",  row.names(investigation_
       data2), ignore.case = F))
90
91  investigation_list <- list()
92  for(i in 1:length(inv_sections))
93  {
94
95    if(i == length(inv_sections))
96      {
97      section_range <- (inv_sections[i] + 1):nrow(investigation_data2)
98      }else{
99      section_range <- (inv_sections[i] + 1):(inv_sections[i+1] - 1)
100     }
101
102   current_section <- investigation_data2[section_range, , drop =F]
103
104   ### remove columns that are only NA
105   current_section <- current_section[, apply(current_section, 2,
        function(x){sum(is.na(x)) != nrow(current_section)}), drop = F]
106
107   ### transpose / pivot data to transform into list
108   current_section_transposed <- as.data.frame(t(current_section))
109   rownames(current_section_transposed) <- NULL
110
111   # TODO stupid work-around to circumvent the bug with a section having
          only NAs
```

```
112   if(nrow(current_section_transposed) == 0){current_section_transposed
          [1, ] = NA}
113
114   current_section_transposed$arc_id <- arc_id
115
116   ### extract current section name
117   current_section_name <- row.names(investigation_data2)[inv_sections[i
          ]]
118
119   # ### transform to named list, omitting NAs
120   # investigation_list[[current_section_name]] <- lapply(current_
          section_transposed, function(v){v[!is.na(v)]})
121   #
122   # stack(current_section_transposed)
123
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
```

```
  ### store in named list

  all_arcs[[arc_id]] <- investigation_list

}

### row-bind the second-level (i.e. INVESTIGATION "sections") of the lists, respectively

all_arcs_db <- do.call(Map, c(f = rbind, all_arcs))

### read the arc_list (translating the ARC id to the ARC path) produced by 02_read_from_gitlab.sh
### and append to above list

arc_list <- read.table(".tmp/02_investigations/arc_list.tsv", sep = "\t", header = T)
colnames(arc_list)[1] <- "arc_id"

# all_arcs_db[["arc_list"]] <- arc_list

### store the output as RData

save(all_arcs, all_arcs_db, arc_list, file = ".tmp/03_allARCs.RData")
```

scripts/scripts/05_pull_together_sql.R

```
#################################################################
### Convert data into SQLite database
#################################################################

# if(!require("DBI", quietly = TRUE)){install.packages("DBI")}
library(DBI)

load(".tmp/03_allARCs.RData")

### Write into an SQLite DB file

mydb <- dbConnect(RSQLite::SQLite(), "yourARCs_database.sqlite")

for(i in names(all_arcs_db))
{
  dbWriteTable(mydb, i, all_arcs_db[[i]], overwrite = T)
}

dbWriteTable(mydb, "ARC list", arc_list, overwrite = T)

dbListTables(mydb)
dbDisconnect(mydb)
```

scripts/scripts/04_searchApp/app.R

```
################################################################
### The ARC-Finder GUI Shiny App
################################################################

suppressMessages(library(tidyverse))
suppressMessages(library(shiny))

# load(".tmp/03_allARCs.RData")

### Flatten ALL values into a 3-column (arc_id | key | value) df to
    provide search across "any field"

all_values <- do.call(
  rbind.data.frame, lapply(all_arcs_db, function(x){
  pivot_longer(x, cols = setdiff(colnames(x), "arc_id"), values_drop_na
      = T)}))

all_values <- as.data.frame(all_values)

arc_list <- unique(arc_list)

shinyApp(
    ui = pageWithSidebar(
        headerPanel("ARC Finder"),
        sidebarPanel(
            selectizeInput('search_key', 'Select a field to search',
               choices = c("Any field", unique(all_values$name))),
            uiOutput("search_field"),
            selectInput(inputId = "arc_path", label = "Select an ARC
               for details", choices = NULL),
            br(),
            actionButton("go", "Show this ARC"),
            h3("Access this ARC in the DataHUB"),
            uiOutput("arc_gitlab"),

        ),

        mainPanel(
          h3("Overview"),
          tableOutput("table_INV"),
          br(),
          h3("Publications"),
          tableOutput("table_INV_PUBS"),
          br(),
          h3("Contacts"),
          tableOutput("table_INV_Contacts")
        )
    ),

```

```
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$search_key)
60          }
61
62
63        selectizeInput('search_value', 'Search the ARC database', choices
              = c("", sort(unique(df$value))))
64
65        })
66
67
68        ##### ARC choices (arc_id) matching user-input
69
70          arc_choices_id <- reactive({
71
72          if ('Any field' %in% input$search_key) {
73
74            subset(all_values, value == input$search_value, arc_id, drop
                 = T)
75
76            } else {
77
78            subset(all_values, name == input$search_key & value == input$
                 search_value, arc_id, drop = T)
79            }
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 =
                T)
89
90          })
91
92        ##### reactive ARC selection: updated Input to let user pick from
```

```
93
94
95          observe({
96            updateSelectInput(session = session, inputId = "arc_path",
                 choices = arc_choices_path())
97            })
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
               _path, arc_id, drop = T))]]
105
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_
                 table) == "arc_id")]
115
116           selected_table$pivot_col <- row.names(selected_table)
117           long <- pivot_longer(selected_table, setdiff(colnames(
                 selected_table), "pivot_col"), values_drop_na = T)
118
119           pivot_wider(long, names_from = pivot_col)
120
121         })
122
123        ##### render table INVESTIGATION PUBLICATIONS
124
125         output$table_INV_PUBS <- renderTable(colnames = F, rownames = F
               , {
126
127           selected_table <- selected_arc()$`INVESTIGATION PUBLICATIONS`
128           selected_table <- selected_table[, -which(colnames(selected_
                 table) == "arc_id")]
129
130           selected_table$pivot_col <- row.names(selected_table)
131           long <- pivot_longer(selected_table, setdiff(colnames(
                 selected_table), "pivot_col"), values_drop_na = T)
132
133           pivot_wider(long, names_from = pivot_col)
134
135
136         })
```

```
137
138         ##### render table INVESTIGATION CONTACTS
139
140           output$table_INV_Contacts <- renderTable(colnames = F, rownames
                  = F, {
141
142
143             selected_table <- selected_arc()$`INVESTIGATION CONTACTS`
144             selected_table <- selected_table[, -which(colnames(selected_
                    table) == "arc_id")]
145
146             selected_table$pivot_col <- row.names(selected_table)
147             long <- pivot_longer(selected_table, setdiff(colnames(
                    selected_table), "pivot_col"), values_drop_na = T)
148
149             pivot_wider(long, names_from = pivot_col)
150
151           })
152
153       ##### render link to gitlab
154
155           output$arc_gitlab <- renderUI(a(href = paste0('https://git.
                  nfdi4plants.org/', input$arc_path),
156                                           paste0('https://git.nfdi4plants
                                                  .org/', input$arc_path) ,
                                                  target="_blank"))
157
158
159           session$onSessionEnded(function() {
160             stopApp()
161           })
162       }
163
164   )
```