Process Southeast Utah Group Long-term Vegetation Monitoring Data

true

June 22, 2022

Contents

1 Introduction

This package processes plant and ground cover data collected in the field and exports them to the Southeast Utah Group (SEUG) Long-term Vegetation Monitoring Program (LTVMP) database. Plant cover and frequency and ground cover data are collected in the field using paper datasheets, then transcribed into Excel workbooks. This package imports the workbook files into R, restructures the data, then exports them to the database. Onset data loggers collect temperature, relative humidity, and precipitation data that are exported to comma delimited files (csv) using the HOBOware application from Onset. The raindancer package is used to import these data into R and summarize them. This package then exports the processed data to the database.

2 Installation

This package is available on GitHub at https://github.com/scoyoc/dataprocessR. Dependent packages include dplyr, glue, lubridate, raindancer, RODBC, stringr, tibble, and tidyr. Suggested packages include knitr and rmarkdown.

```
# Install the devtools package
if (!"devtools" %in% installed.packages()[, "Package"]) {
  # install.packages("devtools")
}
```

```
# Install dataprocessR
devtools::install_github("scoyoc/dataprocessR")
library("dataprocessR")

# Install raindancer
devtools::install_github("scoyoc/raindancer")
library("raindancer")
```

3 Connecting to a Database

First, we need to connect R to a database. If you haven't connected a database to R before you might need to add it to Windows ODBC Data Sources:
1. Search "ODBC" on the Windows Start page and open "ODBC Data Sources (64-bit)."
2. Under the 'User DSN' tab, click the 'Add' button and follow the prompts to connect the database.
3. Here's a helpful document with detailed instructions: https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/open-the-odbc-data-source-administrator

There is an example database included with this package that we'll be using for the vignette. We'll be using the RODBC package to connect the database to R.

```
dat_dir <- system.file("extdata", package = "dataprocessR")
db_name <- "example_db.accdb"
my_db <- RODBC::odbcConnectAccess2007(paste(dat_dir, db_name, sep = "/"))
```

3.1 File Paths

The functions included in this package require full path names to data files. For this vignette we will use *system.file()* to extract the full path name to data included with the package. In your own work, replace the *system.file()* line with the path to your data (e.g., "C:/path/to/data").

4 Plant Cover and Frequency and Ground Cover Data

Paper datasheets are used to record field data. The data are then transcribed into Excel workbooks back in the office. The paper datasheets are easy and cost effective way to collect field data and have proven to be valuable references when questions about plant data arise. The Excel workbooks are effective ways for field technicians to enter and check data back in the office, but the layout of the spreadsheets are not conducive for data analysis. This package imports the data in the Excel workbooks into R, restructures the data, then exports them to the SEUG LTVMP database.

4.1 Importing Excel Workbooks into R

The *import_xls()* function imports data from the Excel workbooks into R. The function requires the full path name to the Excel workbook. Lets start by using *list.files()* to read a list of Excel files into R. There are example Excel iles included in this package.

```
veg_files <- list.files(path = dat_dir, pattern = ".xls", full.names = TRUE,
                        recursive = FALSE)
print(basename(veg_files))
#> [1] "Veg_A03_2020.xls" "Veg_A05_2020.xls" "Veg_A06_2020.xls" "Veg_A08_2020.xls"
```

Now that we have a list of full path names to the Excel files we can read them into R using the *import_xls()* function.

```
dat <- import_xls("C:/path/to/data")
```

The function returns a list with three components:
1. **file_info** is information about the Excel file.
2. **sampling_event** is information about the sampling event.
3. **data** is a data frame of plant cover and frequency and ground cover data.

Let's start with the first file.

```
dat <- import_xls(veg_files[1])
paste("class = ", class(dat)); paste("length = ", length(dat)); names(dat)
#> [1] "class =  list"
#> [1] "length =  3"
#> [1] "file_info"      "sampling_event" "data"
```

Let's examine the components of the list returned by the *import_xls()* function.

file_info is information about the Excel file.

```
str(dat$file_info)
#> 'data.frame':    1 obs. of  4 variables:
#>  $ DataFile  : chr "Veg_A03_2020.xls"
#>  $ ImportDate: chr "2022-06-22"
#>  $ PlotID    : chr "A03"
#>  $ SampleYear: int 2020
```

sampling_event is information about the sampling event.

```
str(dat$sampling_event)
#> 'data.frame':    1 obs. of  4 variables:
#>  $ SampleYear: int 2020
#>  $ PlotID    : chr "A03"
#>  $ SampleDate: chr "2020-04-16"
#>  $ Observers : chr "Josh Doucette;Morgan Cannon;Kenna Ryan"
```

data is a data frame of raw data.

```
str(dat$data)
#> tibble [661 x 7] (S3: tbl_df/tbl/data.frame)
#>  $ Key       : chr [1:661] "2020.A03.1.CRUSTAGE" "2020.A03.1.GRAVEL" "2020.A03.1.LITTER" "2020.A03.1
#>  $ SampleYear: int [1:661] 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
#>  $ PlotID    : chr [1:661] "A03" "A03" "A03" "A03" ...
#>  $ Quad      : int [1:661] 1 1 1 1 1 1 1 1 2 2 ...
#>  $ SppCode   : chr [1:661] "CRUSTAGE" "GRAVEL" "LITTER" "BROTEC" ...
#>  $ F         : int [1:661] 2 1 1 1 3 3 1 3 4 1 ...
#>  $ C         : int [1:661] NA 1 5 1 1 1 2 1 NA 1 ...
```

4.2 Exporting Data to the Database

The *export_xls()* function uses *import_xls()* to read the Excel workbooks into R, then exports the components returned by *import_xls()* to separate tables in the database.

First, let's assign the names of the tables in the database to objects in the R work space to simplify our code.

```
veg_data_table = "tblData"
veg_sampling_event_table = "tblSamplingEvent"
veg_import_table = "tblImportRecord"
```

Now there's one place in our R script to change the table names.

The *export_xls()* function uses *import_xls()* to read the data in the workbooks into R, then exports them to the database. This function relies on RODBC package to write the data to the database; it does not return an R object. This function requires five arguments and there are two optional arguments:
1. **my_xls** - The full path name to the Excel file.
2. **my_db** - The name of the database object in R.
3. **data_table** - The name of the plant cover and frequency and ground cover data table in the database.
4. **sampling_event_table** - The name of the sampling event table in the database.
5. **import_table** - The name of the Excel file import log table in the database.
6. **verbose** - Optional. Prints messages to the console showing function progress. Default is TRUE. If FALSE, messages are suppressed.
7. **view** - Optional. Prints data to console before writing them to the database. Default is TRUE. If FALSE, data are not printed and there is no prompt before writing data to the database.

For this exercise we'll let the function print messages and suppress viewing the data. Viewing can be useful if you want to visually examine the data before writing them to the database.

```
export_xls(veg_files[1],
           my_db = my_db,
           data_table = veg_data_table,
           sampling_event_table = veg_sampling_event_table,
           import_table = veg_import_table,
           verbose = TRUE, view = FALSE)
#> Processing Veg_A03_2020.xls
#> - Writing data to database
#> - Writing sampling event to database
#> - Writing import log to database
#> [1] 1
```

That's it. The data have been exported to the database.

Processing one file at a time is time consuming and inefficient. The *lapply()* function can be used to iterate *export_xls()* through all the Excel files in a folder.

```
lapply(veg_files[2:length(veg_files)], function(this_xls){
  export_xls(my_xls = this_xls,
             my_db = my_db,
             data_table = veg_data_table,
             sampling_event_table = veg_sampling_event_table,
             import_table = veg_import_table,
             verbose = TRUE, view = FALSE)
})
#> Processing Veg_A05_2020.xls
#> - Writing data to database
#> - Writing sampling event to database
#> - Writing import log to database
#> Processing Veg_A06_2020.xls
#> - Writing data to database
#> - Writing sampling event to database
#> - Writing import log to database
#> Processing Veg_A08_2020.xls
#> - Writing data to database
#> - Writing sampling event to database
#> - Writing import log to database
#> [[1]]
#> [1] 1
#>
```

```
#> [[2]]
#> [1] 1
#>
#> [[3]]
#> [1] 1
```

Let's take a look at the database tables now that we've exported all of the plant anc ground cover data.

tblImportRecord is a log of Excel files imported to the database.

```
dplyr::glimpse(RODBC::sqlFetch(my_db, veg_import_table,
                               stringsAsFactors = F))
#> Rows: 4
#> Columns: 4
#> $ DataFile   <chr> "Veg_A03_2020.xls", "Veg_A05_2020.xls", "Veg_A06_2020.xls",~
#> $ ImportDate <chr> "2022-06-22", "2022-06-22", "2022-06-22", "2022-06-22"
#> $ PlotID     <chr> "A03", "A05", "A06", "A08"
#> $ SampleYear <int> 2020, 2020, 2020, 2020
```

tblSamplingEvent is a log of vegetation plot sampling events.

```
dplyr::glimpse(RODBC::sqlFetch(my_db, veg_sampling_event_table,
                               stringsAsFactors = F))
#> Rows: 4
#> Columns: 4
#> $ SampleYear <int> 2020, 2020, 2020, 2020
#> $ PlotID     <chr> "A03", "A05", "A06", "A08"
#> $ SampleDate <chr> "2020-04-16", "2020-04-13", "2020-04-15", "2020-04-16"
#> $ Observers  <chr> "Josh Doucette;Morgan Cannon;Kenna Ryan", "Morgan Cannon;Fa~
```

tblData is a table of raw plant and ground cover data.

```
head(RODBC::sqlFetch(my_db, veg_data_table, stringsAsFactors = F))
#>                   Key SampleYear PlotID Quad  SppCode F  C
#> 1 2020.A03.1.CRUSTAGE       2020    A03    1 CRUSTAGE 2 NA
#> 2   2020.A03.1.GRAVEL       2020    A03    1   GRAVEL 1  1
#> 3   2020.A03.1.LITTER       2020    A03    1   LITTER 1  5
#> 4   2020.A03.1.BROTEC       2020    A03    1   BROTEC 1  1
#> 5   2020.A03.1.SPOCRY       2020    A03    1   SPOCRY 3  1
#> 6   2020.A03.1.HELPET       2020    A03    1   HELPET 3  1
```

5 Weather Data

Weather data are collected in the field at the SEUG LTVMP plots using Onset loggers. These data are exported to comma delimited (csv) files using the Onset HOBOware application. The raindancer package is used to import these data into R and summarize them. See the raindancer vignette for a detailed description of package. This package, dataprocessR, then exports the data to the database using *export_hobo()*.

First, let's pull a list of files into R. Several example csv files are included with this package.

```
wx_files <- list.files(path = dat_dir, pattern = ".csv", full.names = TRUE,
                       recursive = FALSE)
print(basename(wx_files))
#> [1] "20200415_A05_PRCP.csv"    "20200415_A05_TEMP.csv"
#> [3] "20200415_A06_PRCP.csv"    "20200415_A06_Temp.csv"
```

```
#> [5] "20200416_A03_PRCP.csv"    "20200416_A03_TEMP_RH.csv"
#> [7] "20200416_A08_PRCP.csv"    "20200416_A08_TEMP_RH.csv"
```

And let's assign table names again to simplify our code.

```
# Set table names
wx_import_table = "tblWxImportLog"
wx_raw_data_table = "tblWxData_raw"
wx_prcp_data_table = "tblWxData_PRCP"
wx_temp_rh_data_table = "tblWxData_TEMP_RH"
wx_details_table = "tblWxLoggerDetails"
```

Now let's use *export_hobo()* to write these data to the database. Like *export_xls()*, this function relies on RODBC package to write the data to the database and it does not return an R object. This function requires seven arguments and has two optional arguments:
1. **my_file** - The full path name to the csv file.
2. **my_db** - The name of the database object in R.
3. **import_table** - The name of the csv file import log table in the database.
4. **raw_data_table** - The name of the raw weather data table in the database.
5. **prcp_data_table** - The name of the summarized precipitation data table in the database.
6. **temp_rh_data_table** - The name of the summarized temperature and relative humidity data table in the database.
7. **details_table** - The name of the logger settings log table in the database.
8. **verbose** - Optional. Prints messages to the console showing function progress. Default is TRUE. If FALSE, messages are suppressed.
8. **view** - Optional. Prints data to console before writing them to the database. Default is TRUE. If FALSE, data are not printed and there is no prompt before writing data to the database.

Like before, we'll let the function print messages and suppress viewing the data.

```
export_hobo(wx_files[1],
            my_db = my_db,
            import_table = wx_import_table,
            raw_data_table = wx_raw_data_table,
            prcp_data_table = wx_prcp_data_table,
            temp_rh_data_table = wx_temp_rh_data_table,
            details_table = wx_details_table,
            verbose = TRUE,
            view = FALSE)
#> Processing 20200415_A05_PRCP.csv
#> - Writing raw data to database
#> - Writing processed data to database
#> - Writing logger details to database
#> - Writing import log to database
#> [1] 1
```

And like before, we can improve our efficiency by using the *lapply()* function to iterate *export_hobo()* through all csv files in a folder.

```
lapply(wx_files[2:length(wx_files)], function(this_file){
  export_hobo(this_file,
              my_db = my_db,
              import_table = wx_import_table,
              raw_data_table = wx_raw_data_table,
              prcp_data_table = wx_prcp_data_table,
              temp_rh_data_table = wx_temp_rh_data_table,
```

```
              details_table = wx_details_table,
              verbose = TRUE,
              view = FALSE)
})
#> Processing 20200415_A05_TEMP.csv
#> - Writing raw data to database
#> - Writing processed data to database
#> - Writing logger details to database
#> - Writing import log to database
#> Processing 20200415_A06_PRCP.csv
#> - Writing raw data to database
#> - Writing processed data to database
#> - Writing logger details to database
#> - Writing import log to database
#> Processing 20200415_A06_Temp.csv
#> - Writing raw data to database
#> - Writing processed data to database
#> - Writing logger details to database
#> - Writing import log to database
#> Processing 20200416_A03_PRCP.csv
#> - Writing raw data to database
#> - Writing processed data to database
#> - Writing logger details to database
#> - Writing import log to database
#> Processing 20200416_A03_TEMP_RH.csv
#> - Writing raw data to database
#> - Writing processed data to database
#> - Writing logger details to database
#> - Writing import log to database
#> Processing 20200416_A08_PRCP.csv
#> - Writing raw data to database
#> - Writing processed data to database
#> - Writing logger details to database
#> - Writing import log to database
#> Processing 20200416_A08_TEMP_RH.csv
#> - Writing raw data to database
#> - Writing processed data to database
#> - Writing logger details to database
#> - Writing import log to database
#> [[1]]
#> [1] 1
#>
#> [[2]]
#> [1] 1
#>
#> [[3]]
#> [1] 1
#>
#> [[4]]
#> [1] 1
#>
#> [[5]]
#> [1] 1
```

```
#>
#> [[6]]
#> [1] 1
#>
#> [[7]]
#> [1] 1
```

Let's take a look at the database tables now that we've exported all of the weather data.

raw_data_table = "tblWxData_raw" prcp_data_table = "tblWxData_PRCP" temp_rh_data_table = "tblWxData_TEMP_RH"

tblWxImportLog is a log of csv files imported to the database.

```
dplyr::glimpse(RODBC::sqlFetch(my_db, wx_import_table,
                               stringsAsFactors = F))
#> Rows: 10
#> Columns: 11
#> $ FileName         <chr> "20200415_A05_PRCP.csv", "20200415_A05_TEMP.csv", "20~
#> $ PlotID           <chr> "A05", "A05", "A06", "A06", "A03", "A03", "A03", "A08~
#> $ Element          <chr> "PRCP", "TEMP", "PRCP", "TEMP", "PRCP", "TEMP", "RH",~
#> $ Product          <chr> "HOBO UA-003-64 Pendant Temp/Event", "HOBO UA-001-64 ~
#> $ SerialNumber     <int> 20551774, 20561144, 20551794, 20561146, 20551795, 205~
#> $ LaunchName       <chr> "A05_PRCP", "A05_TEMP", "A06 PRCP", "A06 Temp", "A03_~
#> $ DeploymentNumber <int> 3, 3, 1, 1, 1, 1, 1, 1, 1, 1
#> $ LaunchTime       <chr> "03/25/19 07:37:49 AM GMT-06:00", "03/25/19 07:37:24 ~
#> $ FirstSampleTime  <chr> "03/25/19 05:00:00 PM GMT-06:00", "03/25/19 05:00:00 ~
#> $ LastSampleTime   <chr> "07/31/19 07:17:23 PM GMT-06:00", "10/31/19 03:00:00 ~
#> $ ImportDate       <chr> "2022-06-22", "2022-06-22", "2022-06-22", "2022-06-22~
```

tblWxLoggerDetails is a log of logger settings.

```
head(RODBC::sqlFetch(my_db, wx_details_table, stringsAsFactors = F))
#>                FileName        Details                                Value
#> 1 20200415_A05_PRCP.csv        Product HOBO UA-003-64 Pendant Temp/Event
#> 2 20200415_A05_PRCP.csv  Serial Number                             20551774
#> 3 20200415_A05_PRCP.csv Version Number                                1.17
#> 4 20200415_A05_PRCP.csv   Manufacturer                Onset Computer Corp.
#> 5 20200415_A05_PRCP.csv  Device Memory                               65536
#> 6 20200415_A05_PRCP.csv Header Created    09/04/13 09:40:47 AM GMT-06:00
```

tblWxData_PRCP is a table of summarized precipitation data.

```
head(RODBC::sqlFetch(my_db, wx_prcp_data_table, stringsAsFactors = F))
#>   PlotID            DateTime Element PRCP_mm Tips MaxTips_min
#> 1    A05 2019-03-25 00:00:00    PRCP       0    0           0
#> 2    A05 2019-03-25 01:00:00    PRCP       0    0           0
#> 3    A05 2019-03-25 02:00:00    PRCP       0    0           0
#> 4    A05 2019-03-25 03:00:00    PRCP       0    0           0
#> 5    A05 2019-03-25 04:00:00    PRCP       0    0           0
#> 6    A05 2019-03-25 05:00:00    PRCP       0    0           0
```

tblWxData_TEMP_RH is a table of summarized temperature and relative humidity data.

```
head(RODBC::sqlFetch(my_db, wx_temp_rh_data_table, stringsAsFactors = F))
#>   PlotID                Date Element      Mean    Min    Max  n MinTime MaxTime
#> 1    A05 2019-03-25 00:00:00    TEMP 11.797786  7.381 16.713 14   23:30   17:00
#> 2    A05 2019-03-26 00:00:00    TEMP 10.977229  3.893 20.996 48   07:00   17:30
```

```
#> 3    A05 2019-03-27 00:00:00    TEMP 13.732958  6.166 21.187 48   06:30   15:30
#> 4    A05 2019-03-28 00:00:00    TEMP 15.678417 11.334 20.710 48   04:00   14:30
#> 5    A05 2019-03-29 00:00:00    TEMP  9.227063  4.934 12.787 48   23:30   15:00
#> 6    A05 2019-03-30 00:00:00    TEMP  6.445458  1.330 11.625 48   07:00   17:00
#>   Units
#> 1     F
#> 2     F
#> 3     F
#> 4     F
#> 5     F
#> 6     F
```

tblWxData_raw is a table of the raw, not-summarized, weather data.

```
head(RODBC::sqlFetch(my_db, wx_raw_data_table, stringsAsFactors = F))
#>                FileName PlotID            DateTime Element Value Units
#> 1 20200415_A05_PRCP.csv    A05 2019-03-25 17:00:00    PRCP     0 Event
#> 2 20200415_A05_PRCP.csv    A05 2019-04-10 03:43:11    PRCP     1 Event
#> 3 20200415_A05_PRCP.csv    A05 2019-04-10 04:17:37    PRCP     2 Event
#> 4 20200415_A05_PRCP.csv    A05 2019-04-10 04:23:01    PRCP     3 Event
#> 5 20200415_A05_PRCP.csv    A05 2019-04-10 04:30:08    PRCP     4 Event
#> 6 20200415_A05_PRCP.csv    A05 2019-04-10 04:38:03    PRCP     5 Event
```

5.1 Troubleshooting Weather Data

The structure of the csv files generated from HOBOware vary wildly. They can have anywhere from 4 to 10 columns and the logger details are usually in two "hidden" columns following the data. This makes data from the temperature, relative humidity, and precipitation data difficult to read into R. Below are some tips to troubleshoot the csv files when they fail to read into R correctly and *export_hobo()* fails.

1. First open the *.csv file in a text editor and examine the data. Most of the time the data are simply invalid for some reason, it's probably noted in the Field Notes. If this is the case, move the file to the "not processed" directory and move on.

2. If the data look valid, use the code below to find where a file is not reading in to R correctly. Sometimes this is fairly easy to fix in the *.csv, other times is a real bugger.

More hints:

Files before 2020. A common bug in earlier *.csv files is when the logger name on line 13, "Launch Name", has an apostrophe in it (e.g. Devil's Garden). Remove the apostrophe and the file will read into R.

Files 2020 and later. A common bug in more recent csv files, HoboWare does not write column names for the last two columns, Details and units. This error reports "more columns than column names". Open the file and add column names to the last two columns of row two. The file should read into R smoothly.

```
#-- Isolate file and begin troubleshooting
file_index <- 11                        # Index number of the trouble file. This number will change.
file_list[file_index]                   # Prints name of trouble file.
my_file <- file_list[file_index]        # Assigns file path to R object.
raindancer::import_hobo(my_file)   # Test raindancer function for error.

#-- Problems with parsing date-time
read.table(my_file, sep = ",", skip = 1) |> # skip argument may differ
```

```
  tibble::tibble() |>
  dplyr::rename("DateTime" = V2) |> # Variable name (V2) may differ
  dplyr::mutate("DateTime" = lubridate::parse_date_time(DateTime,
                                                        orders = "%m/%d/%y hh:mm"))

#-- Problems with csv file headers
# Play around with skip argument until csv read into R correctly
read.csv(my_file, skip = 1)  |> tibble::tibble()
readr::read_csv(my_file, skip = 1, show_col_types = FALSE)  |> tibble::tibble()

#-- Testing data summary functions
raindancer::import_hobo(my_file) |> raindancer::raindance(dat)
raindancer::import_hobo(my_file) |> raindancer::sundance(dat)

raindancer::import_hobo(file_list[file_index]) |> raindancer::process_hobo()

# Continue processing files in directory after an invalid file breaks the
#    lapply() is sorted out.
# After troubleshooting the file that caused export_hobo() to  crash.
# Change the number to the index of the next file and continue to process the
#    rest of the files in the directory.
lapply(file_list[file_index:length(file_list)], function(this_file){
  export_hobo(this_file, my_db = my_db,
              import_table = import_table,
              raw_data_table = raw_data_table,
              prcp_data_table = prcp_data_table,
              temp_rh_data_table = temp_rh_data_table,
              details_table = details_table,
              view = FALSE)
})
```

6 Reporting Errors and Issues

Please submit any problems on the Issues page of this GitHub repository, https://github.com/scoyoc/dataprocessR/issues, or contact the author of the package if this happens.