# faoswsSeed: A package for the imputation of the seed domain of the Statistical Working System

**Joshua M. Browning, Michael. C. J. Kao**
Food and Agriculture Organization
of the United Nations

**Abstract**

This vignette provides detailed description of the usage of functions in the **faoswsSeed** package.

There are two sections to this paper. The first is introductory, and provides a breif overview of the algorithm followed by this package. The second section shows a sample execution of the module, and describes what each function is doing as execution proceeds.

*Keywords*: Seed, Agriculture.



## 1. Introduction

This algorithm follows this general process:

1. Pull agricultural data from the database.
2. Estimate the area sown for each year.
3. Estimate the seed rate.
4. Estimate the total seed used by multiplying the area sown by the seed rate.
5. Push the updated data.table back to the database.

## 2. Example

Before we begin, we will need to load the required library

```
## Load libraries
library(faosws)
library(faoswsSeed)
library(faoswsImputation)
library(faoswsUtil)
library(data.table)
library(ggplot2)
```

### 2.1. Pull Data

Now, we need to get a data.table object from the working system. To do this, we'll need a token for the R session. To get this token, you'll need to create an .xml file that references the

appropriate dataset: see the .xml file in the sws_seed repository. Alternatively, you can create an .xml file with the Code tag set to agriculture and four SelectableDimension's: geographicAreaM49, measuredItemCPC, measuredElement, timePointYears. Once you get this token, you can run GetTestEnvironment (see below, baseUrl is the same) to load some variables (usually starting with swsContext.*) into your workspace.

Now, we may change the dataset of interest by updating the keys in the *swsContext.datasets[[1]]* object. For example, let's look at Austria's data (geographicAreaM49 code of "40"). Note: multiple keys are allowed.

Lastly, the getAreaData function will pull in the seed data needed. The code below shows an example of what these calls look like, although it is not executable as the token is no longer valid.

```
## Pull in necessary swsContext parameters, see faosws documentation
GetTestEnvironment(
    baseUrl = "https://hqlqasws1.hq.un.fao.org:8181/sws",
    token = "ec5a4b0e-0ffa-432e-9db0-ba08072c924b"
)
swsContext.datasets[[1]]@dimensions$geographicAreaM49@keys = c("40")
data = getAreaData(dataContext = swsContext.datasets[[1]],
                   areaSownElementCode = "5212",
                   areaHarvestedElementCode = "5312",
                   seedElementCode = "5525")
```

We can't pull this data directly, but the faoswsSeed package has a default dataset that would result from this kind of a call:

```
seedData

##       geographicAreaM49 measuredItemCPC timePointYears
##    1:               100            0111           1994
##    2:               100            0111           1995
##    3:               100            0111           1996
##    4:               100            0111           1997
##    5:               100            0111           1998
##   ---
## 8956:               348        26190.01           2009
## 8957:               348        26190.01           2010
## 8958:               348        26190.01           2011
## 8959:               348        26190.01           2012
## 8960:               348        26190.01           2013
##       Value_measuredElement_5212 flagObservationStatus_measuredElement_5212
##    1:                         NA                                          M
##    2:                         NA                                          M
##    3:                         NA                                          M
##    4:                         NA                                          M
##    5:                         NA                                          M
##   ---
## 8956:                         NA                                          M
## 8957:                         NA                                          M
## 8958:                         NA                                          M
## 8959:                         NA                                          M
```

```
## 8960:                           NA                                          M
##       flagMethod_measuredElement_5212 Value_measuredElement_5312
##    1:                               u                     1319760
##    2:                               u                     1181120
##    3:                               u                      957670
##    4:                               u                     1211720
##    5:                               u                     1141682
##   ---
## 8956:                               u                          NA
## 8957:                               u                          NA
## 8958:                               u                          NA
## 8959:                               u                          NA
## 8960:                               u                          NA
##       flagObservationStatus_measuredElement_5312
##    1:
##    2:
##    3:
##    4:
##    5:
##   ---
## 8956:                                         M
## 8957:                                         M
## 8958:                                         M
## 8959:                                         M
## 8960:                                         M
##       flagMethod_measuredElement_5312 Value_measuredElement_5525
##    1:                               -                          NA
##    2:                               -                          NA
##    3:                               -                          NA
##    4:                               -                          NA
##    5:                               -                          NA
##   ---
## 8956:                               u                          NA
## 8957:                               u                          NA
## 8958:                               u                          NA
## 8959:                               u                          NA
## 8960:                               u                          NA
##       flagObservationStatus_measuredElement_5525
##    1:                                         M
##    2:                                         M
##    3:                                         M
##    4:                                         M
##    5:                                         M
##   ---
## 8956:                                         M
## 8957:                                         M
## 8958:                                         M
## 8959:                                         M
## 8960:                                         M
##       flagMethod_measuredElement_5525
##    1:                               u
```

```
##    2:                                u
##    3:                                u
##    4:                                u
##    5:                                u
##   ---
## 8956:                                u
## 8957:                                u
## 8958:                                u
## 8959:                                u
## 8960:                                u
```

The data is somewhat cryptic, as there are alot of codes used, but it's there.

### 2.2. Estimate Area Sown

Next, we need to impute the missing values. Imputation follows one of several methods:

1. If no area sown values exist, the area sown is imputed as the area harvested.
2. If some area sown values exist, then imputation is performed based on input parameters. If the imputationParameters argument is NULL, then an average ratio is computed across all non-missing values within each byKey group (by default, byKey is NULL and so the ratio is computed with all the data): $R$ =(area sown) / (area harvested). Missing values for area sown are then imputed by taking the area harvested and multiplying by this ratio.
3. If some area sown values exist and imputationParameters is not NULL, then imputation is performed via ensemble imputation from the faoswsImputation package. All of the elegant models in that framework are not likely to be useful here, as most countries have no data. However, two models such as a local and global mean may be helpful.

In this example, we have the first scenario (remember, 5212 is the area sown code and 5312 is the area harvested).

```
seedData[geographicAreaM49 == 348 & measuredItemCPC == "01330",
         .(time = timePointYears, areaSown = Value_measuredElement_5212,
           areaHarvested = Value_measuredElement_5312)]
```

```
##     time areaSown areaHarvested
##  1: 1994   131916        101212
##  2: 1995   131334        100108
##  3: 1996   130934         99660
##  4: 1997   130874         98901
##  5: 1998   129658         99099
##  6: 1999   127066         99000
##  7: 2000   106000         88672
##  8: 2001    93000         82186
##  9: 2002    93000         82846
## 10: 2003    93100         93032
## 11: 2004    93000         93217
## 12: 2005   100000         86028
## 13: 2006    80000         75634
```

```
## 14: 2007    80000         75260
## 15: 2008    83000         75776
## 16: 2009    83000         75933
## 17: 2010       NA         73922
## 18: 2011    83000         75511
## 19: 2012       NA            NA
## 20: 2013       NA            NA

temp = copy(seedData)
imputeAreaSown(data = temp)
temp[geographicAreaM49 == 348 & measuredItemCPC == "01330",
     .(time = timePointYears, areaSown = Value_measuredElement_5212,
       areaHarvested = Value_measuredElement_5312, Value_areaSownRatio)]

##     time  areaSown areaHarvested Value_areaSownRatio
##  1: 1994 131916.00        101212            1.008237
##  2: 1995 131334.00        100108            1.008237
##  3: 1996 130934.00         99660            1.008237
##  4: 1997 130874.00         98901            1.008237
##  5: 1998 129658.00         99099            1.008237
##  6: 1999 127066.00         99000            1.008237
##  7: 2000 106000.00         88672            1.008237
##  8: 2001  93000.00         82186            1.008237
##  9: 2002  93000.00         82846            1.008237
## 10: 2003  93100.00         93032            1.008237
## 11: 2004  93000.00         93217            1.008237
## 12: 2005 100000.00         86028            1.008237
## 13: 2006  80000.00         75634            1.008237
## 14: 2007  80000.00         75260            1.008237
## 15: 2008  83000.00         75776            1.008237
## 16: 2009  83000.00         75933            1.008237
## 17: 2010  74530.91         73922            1.008237
## 18: 2011  83000.00         75511            1.008237
## 19: 2012        NA            NA            1.008237
## 20: 2013        NA            NA            1.008237
```

The areaSownRatio is estimated globally by default. We could also estimate the ratio within each country individually:

```
temp = copy(seedData)
imputeAreaSown(data = temp, valueAreaSown = "Value_measuredElement_5212",
               byKey = "geographicAreaM49")
temp[geographicAreaM49 == 348 & measuredItemCPC == "01330",
     .(time = timePointYears, areaSown = Value_measuredElement_5212,
       areaHarvested = Value_measuredElement_5312, Value_areaSownRatio)]

##     time  areaSown areaHarvested Value_areaSownRatio
##  1: 1994 131916.00        101212            1.013193
##  2: 1995 131334.00        100108            1.013193
##  3: 1996 130934.00         99660            1.013193
##  4: 1997 130874.00         98901            1.013193
```

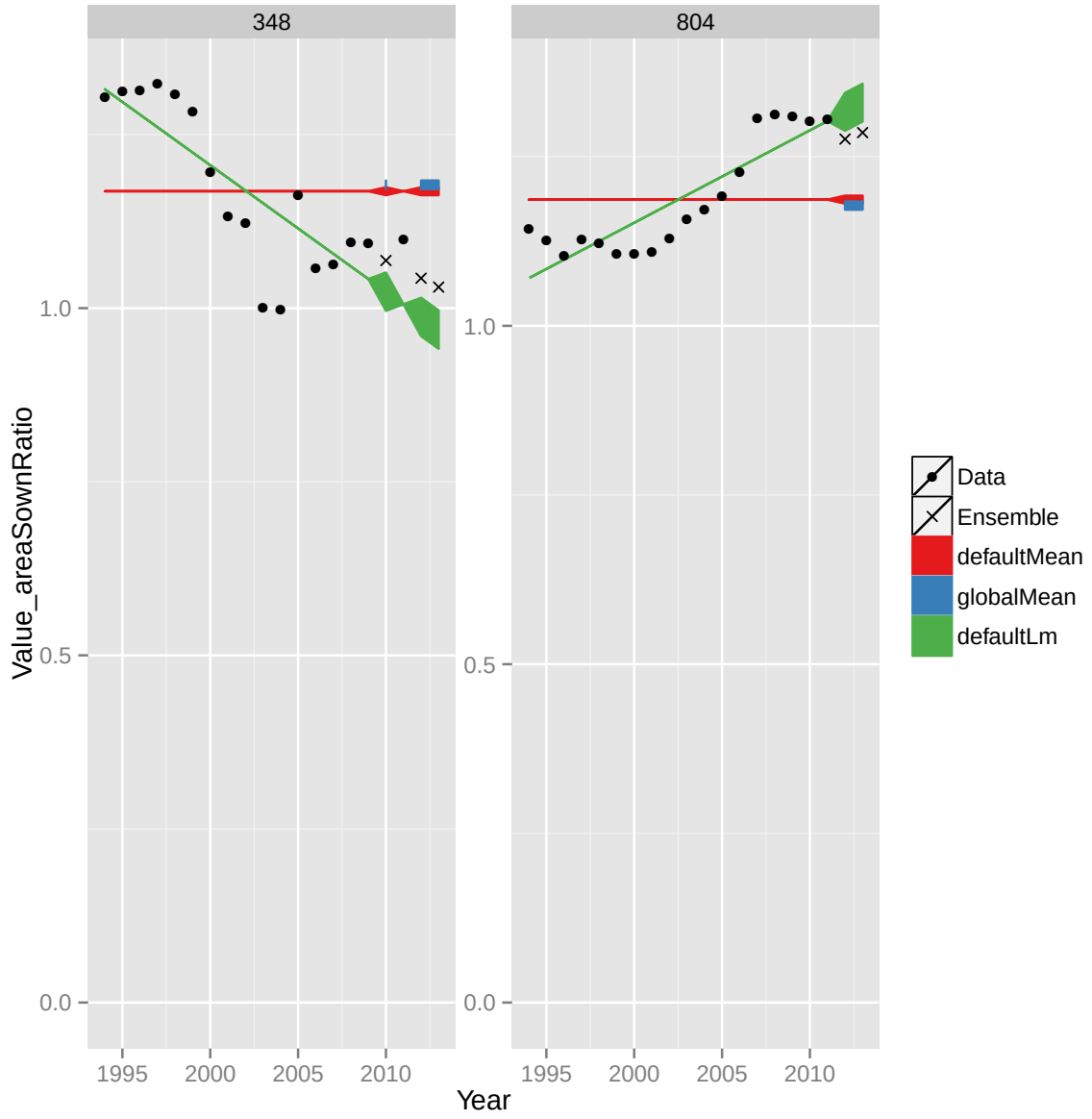
```
##  5: 1998 129658.00          99099            1.013193
##  6: 1999 127066.00          99000            1.013193
##  7: 2000 106000.00          88672            1.013193
##  8: 2001  93000.00          82186            1.013193
##  9: 2002  93000.00          82846            1.013193
## 10: 2003  93100.00          93032            1.013193
## 11: 2004  93000.00          93217            1.013193
## 12: 2005 100000.00          86028            1.013193
## 13: 2006  80000.00          75634            1.013193
## 14: 2007  80000.00          75260            1.013193
## 15: 2008  83000.00          75776            1.013193
## 16: 2009  83000.00          75933            1.013193
## 17: 2010  74897.26          73922            1.013193
## 18: 2011  83000.00          75511            1.013193
## 19: 2012        NA             NA            1.013193
## 20: 2013        NA             NA            1.013193
```

Or, we can estimate the ratio via an ensemble method (from faoswsImputation):

```
imputationParams = defaultImputationParameters(variable = "seed")
## Coerce type to character instead of default factor type
imputationParams$flagTable$flagObservationStatus =
    as.character(imputationParams$flagTable$flagObservationStatus)
imputationParams$ensembleModels = list(
    defaultMean = ensembleModel(model = defaultMean, extrapolationRange = Inf,
                                level = "countryCommodity"),
    globalMean = ensembleModel(model = defaultGlobalMean,
                               extrapolationRange = Inf, level = "commodity"),
    defaultLm = ensembleModel(model = defaultLm, extrapolationRange = Inf,
                              level = "countryCommodity"))
temp = seedData[measuredItemCPC == "01330", ]
imputeAreaSown(data = temp, imputationParameters = imputationParams)
temp[geographicAreaM49 == 348 & measuredItemCPC == "01330",
     .(time = timePointYears, areaSown = Value_measuredElement_5212,
       areaHarvested = Value_measuredElement_5312, Value_areaSownRatio)]
```

```
##     time  areaSown areaHarvested Value_areaSownRatio
##  1: 1994 131916.00        101212            1.303363
##  2: 1995 131334.00        100108            1.311923
##  3: 1996 130934.00         99660            1.313807
##  4: 1997 130874.00         98901            1.323283
##  5: 1998 129658.00         99099            1.308368
##  6: 1999 127066.00         99000            1.283495
##  7: 2000 106000.00         88672            1.195417
##  8: 2001  93000.00         82186            1.131580
##  9: 2002  93000.00         82846            1.122565
## 10: 2003  93100.00         93032            1.000731
## 11: 2004  93000.00         93217            1.000000
## 12: 2005 100000.00         86028            1.162412
## 13: 2006  80000.00         75634            1.057725
## 14: 2007  80000.00         75260            1.062982
```

```
## 15: 2008  83000.00          75776            1.095334
## 16: 2009  83000.00          75933            1.093069
## 17: 2010  78994.25          73922            1.068616
## 18: 2011  83000.00          75511            1.099178
## 19: 2012        NA             NA            1.043119
## 20: 2013        NA             NA            1.030370
```




In this example, notice that we filtered the seedData set to one specific measuredItemCPC code. This is a requirement for imputeAreaSown: it can only handle one CPC code at a time. A future goal of this package is to add an additional function that calls imputeAreaSown for each individual CPC code.

Additionally, note that the defaultLm model could be problematic: predictions in the later years for the left graph give ratios of less than 1. These don't make sense: you can't sow less area than your harvest. There is a check within the code that corrects any imputed values less than 1 (by setting it to 1); however, the analyst should ensure that the models they use do not impute values smaller than 1 as a good practice.

Note that this ensemble model seems to generate improved estimates of area sown rates in

this scenario. However, in most scenarios, area sown is not available. Moreover, if it is, it is generally available for almost all years. The usual scenario would be more like the following example:

```
data = seedData[measuredItemCPC == "0111", ]
data[geographicAreaM49 == 100,
     .(time = timePointYears, areaSown = Value_measuredElement_5212,
       areaHarvested = Value_measuredElement_5312)]

##      time areaSown areaHarvested
##  1: 1994       NA       1319760
##  2: 1995       NA       1181120
##  3: 1996       NA        957670
##  4: 1997       NA       1211720
##  5: 1998       NA       1141682
##  6: 1999       NA        966282
##  7: 2000       NA        978575
##  8: 2001       NA       1355500
##  9: 2002       NA       1368627
## 10: 2003       NA        841014
## 11: 2004       NA       1039680
## 12: 2005       NA       1101807
## 13: 2006       NA        970392
## 14: 2007       NA       1087996
## 15: 2008       NA       1111533
## 16: 2009       NA       1247718
## 17: 2010       NA       1137650
## 18: 2011       NA       1137642
## 19: 2012       NA       1090000
## 20: 2013       NA            NA

imputeAreaSown(data = data, imputationParameters = imputationParams)
data[geographicAreaM49 == 100,
     .(time = timePointYears, areaSown = Value_measuredElement_5212,
       areaHarvested = Value_measuredElement_5312, Value_areaSownRatio)]

##      time areaSown areaHarvested Value_areaSownRatio
##  1: 1994  1319760       1319760                   1
##  2: 1995  1181120       1181120                   1
##  3: 1996   957670        957670                   1
##  4: 1997  1211720       1211720                   1
##  5: 1998  1141682       1141682                   1
##  6: 1999   966282        966282                   1
##  7: 2000   978575        978575                   1
##  8: 2001  1355500       1355500                   1
##  9: 2002  1368627       1368627                   1
## 10: 2003   841014        841014                   1
## 11: 2004  1039680       1039680                   1
## 12: 2005  1101807       1101807                   1
## 13: 2006   970392        970392                   1
## 14: 2007  1087996       1087996                   1
```

```
## 15: 2008  1111533       1111533                    1
## 16: 2009  1247718       1247718                    1
## 17: 2010  1137650       1137650                    1
## 18: 2011  1137642       1137642                    1
## 19: 2012  1090000       1090000                    1
## 20: 2013       NA            NA                    1
```

```
imputeAreaSown(data = data)
data[geographicAreaM49 == 100,
     .(time = timePointYears, areaSown = Value_measuredElement_5212,
       areaHarvested = Value_measuredElement_5312, Value_areaSownRatio)]
```

```
##     time areaSown areaHarvested Value_areaSownRatio
##  1: 1994  1319760       1319760                    1
##  2: 1995  1181120       1181120                    1
##  3: 1996   957670        957670                    1
##  4: 1997  1211720       1211720                    1
##  5: 1998  1141682       1141682                    1
##  6: 1999   966282        966282                    1
##  7: 2000   978575        978575                    1
##  8: 2001  1355500       1355500                    1
##  9: 2002  1368627       1368627                    1
## 10: 2003   841014        841014                    1
## 11: 2004  1039680       1039680                    1
## 12: 2005  1101807       1101807                    1
## 13: 2006   970392        970392                    1
## 14: 2007  1087996       1087996                    1
## 15: 2008  1111533       1111533                    1
## 16: 2009  1247718       1247718                    1
## 17: 2010  1137650       1137650                    1
## 18: 2011  1137642       1137642                    1
## 19: 2012  1090000       1090000                    1
## 20: 2013       NA            NA                    1
```

The two cases above are identical. Using an ensemble provides no advantage as no area sown data is originally available.

### 2.3. Estimate Seed Rate

For this vignette, we will proceed with the dataset defined above: seedData with a CPC code of 0111.

The next step in getting the seed usage is to estimate the seeding rate. The database stores two tables, default_seed_rate and specific_seed_rate, which contain estimates for seed rates. The specific seed rate table contains values for country commodity pairs, while the default seed rate contains average values for commodities overall. It would therefore be preferable to use the specific_seed_rate table, but values are not always available for all countries. Thus, the default_seed_rate table is used when entries are not available in the specific_seed_rate table.

```
# countrySpecificData = getCountrySpecificSeedRate()
countrySpecificData = data.table(
    geographicAreaM49 = c("100", "348", "400"),
```

```
    measuredItemCPC = "0111",
    Value_seedRate = c(222, 213, 115),
    flagObservationStatus_seedRate = c("E", "E", ""))
setkeyv(countrySpecificData, c("geographicAreaM49", "measuredItemCPC"))
fillCountrySpecificSeedRate(data = data,
                            countrySpecificData = countrySpecificData)
head(data, 1)
data[, Value_seedRate]
```

```
##   [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [61]   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
##  [76]   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
##  [91]   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
```

The getCountrySpecificSeedRate function simply pulls the specific_seed_rate table from the database. This table is also the default value for countrySpecificData in the fillCountrySpecificSeedRate function, so generally it will not need to be created as here (although it could be created manually from the commented out line above). However, for vignette creation, it was simpler to just create the part of the table we used at the time of the writing, and this is what is done here. The fillCountrySpecificSeedRate function adds an additional two columns to data with the seedRate value and observation flags.

```
# generalSeedData = getCountryGeneralSeedRate()
generalSeedData = data.table(measuredItemCPC = "0111", Value_seedRate = 151.14,
                             flagObservationStatus_seedRate = "")
setkeyv(generalSeedData, "measuredItemCPC")
fillGeneralSeedRate(data = data,
                    generalSeedData = generalSeedData)
head(data, 1)
data[, Value_seedRate]
```

```
##   [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [76] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [91] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

This function updates all the NA values to the ("global") default seeding rate for this commodity.

### 2.4. Estimate Seed Usage

The seed usage is estimated by way of the imputeSeed function:

```
data[, oldSeed := Value_measuredElement_5525]
imputeSeed(data)
data[geographicAreaM49 == 100, .(timePointYears,
                                 AreaSown = Value_measuredElement_5212,
                                 AreaHarvested = Value_measuredElement_5312,
                                 Seed = Value_measuredElement_5525,
                                 oldSeed)]
```

```
##     timePointYears AreaSown AreaHarvested       Seed oldSeed
##  1:           1994  1319760       1319760   1181.120      NA
##  2:           1995  1181120       1181120    957.670      NA
##  3:           1996   957670        957670   1211.720      NA
##  4:           1997  1211720       1211720   1141.682      NA
##  5:           1998  1141682       1141682    966.282      NA
##  6:           1999   966282        966282    978.575      NA
##  7:           2000   978575        978575   1355.500      NA
##  8:           2001  1355500       1355500   1368.627      NA
##  9:           2002  1368627       1368627    841.014      NA
## 10:           2003   841014        841014   1039.680      NA
## 11:           2004  1039680       1039680   1101.807      NA
## 12:           2005  1101807       1101807    970.392      NA
## 13:           2006   970392        970392   1087.996      NA
## 14:           2007  1087996       1087996   1111.533      NA
## 15:           2008  1111533       1111533 396000.000  396000
## 16:           2009  1247718       1247718 753000.000  753000
## 17:           2010  1137650       1137650   1137.642      NA
## 18:           2011  1137642       1137642   1090.000      NA
## 19:           2012  1090000       1090000         NA      NA
## 20:           2013       NA            NA   1058.749      NA
```

Not all seeds have been imputed. To understand why, note that area sown is available for all years except the last year. But, area sown on year $t$ corresponds to seed usage on year $t-1$, thus we don't have enough valid observations to impute either of the last two seed usages. These values remain missing, as seen above.

### 2.5. Push data back to database

Lastly, we must push the imputed data back to the database. This is done via the saveSeedData function (not evaluated, as the token would need to be valid).

```
saveSeedData(data)
```

Note: this function assumes certain swsContext files exist within your workspace, although it does not require you to pass them to it. So, be careful not to rm those files!

**Affiliation:**

Joshua M. Browning
Economics and Social Statistics Division (ESS)
Economic and Social Development Department (ES)

Food and Agriculture Organization of the United Nations (FAO)
Viale delle Terme di Caracalla 00153 Rome, Italy
E-mail: joshua.browning@fao.org
URL: https://github.com/rockclimber112358/sws_seed