Accessing WoSIS from R – 'Snapshot' Version

D G Rossiter

20-July-2021

Table of Contents

This document shows how to access WoSIS "Snapshot" data from R. For access to WoSIS "Latest" data from R, see `WoSIS_Latest_with_R.Rmd` at https://git.wur.nl/Batje001/wosis/-/tree/master/R_scripts.

The "Snapshot" datasets are static, containing the standardised soil profile (point observation) data available at a given moment (e.g. July 2016). So far there are two of these, registered with Digital Object Identifiers (DOI):

- 2016: https://dx.doi.org/10.17027/isric-wdcsoils.20160003
- 2019: https://dx.doi.org/10.17027/isric-wdcsoils.20190901

The reason to have snapshots, as opposed to just the latest information, is to allow comparisons of datasets as they evolve over time.

For an overview of WoSIS, see https://www.isric.org/explore/wosis. This links to https://www.isric.org/explore/wosis/accessing-wosis-derived-datasets which explains the difference between snapshot and dynamic datasets, and how to access them.

The Procedures Manual describes how the database was built.

Packages

If you do not have these on your system, install with `install.packages(..., dependencies=TRUE)` or via the R Studio package manager.

```
library(rgdal)          # interface to GDAL Geographic Data Abstraction
Language

## Loading required package: sp

## rgdal: version: 1.5-23, (SVN revision 1121)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.1.4, released 2020/10/20
## Path to GDAL shared files:
/Library/Frameworks/R.framework/Versions/4.0/Resources/library/rgdal/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 6.3.1, February 10th, 2020, [PJ_VERSION: 631]
## Path to PROJ shared files:
/Library/Frameworks/R.framework/Versions/4.0/Resources/library/rgdal/proj
## Linking to sp version:1.4-5
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading
rgdal.

library(gdalUtils)      # some useful utilities for GDAL
library(readr)          # tidyverse functions to read files
library(sf, warn.conflicts = FALSE)             # Simple Features spatial
data

## Linking to GEOS 3.8.1, GDAL 3.2.1, PROJ 7.2.1

library(sp)             # spatial data types in R
library(dplyr, warn.conflicts = FALSE)          # another way to handle
tabular data
library(dbplyr, warn.conflicts = FALSE)    # databases from dplyr
library(DBI)            # R database interface
library(RSQLite)        # R interface to SQLite databses
```

GDAL is used for spatial data import/export, coordinate systems, etc. Check for a valid GDAL installation with the following code (not run here):

```
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
```

```
if (valid_install)
   print("Valid GDAL found") else stop("No valid GDAL")
```

Downloading a WoSIS Snapshot

A "Snapshot" is downloaded as a compressed file from the stable data location given by its DOI, for example the 2019 version: https://dx.doi.org/10.17027/isric-wdcsoils.20190901. This link is to the page which describes the dataset, its metadata, a WMS (Web Mapservice) link, and a download link, to work with the data in R. It is not possible to download only part of the database; any subsetting must be done after download.

Download the 2019 snapshot into a subdirectory relative to the current working directory, creating the subdirectory if necessary. This is a *very large file*, about 146.5 Mb, so if it has already been downloaded, do not do it again.

```
wosis.dir.name <- "./wosis2019"
if (!file.exists(wosis.dir.name)) dir.create(wosis.dir.name)
zip.file.name <- "WoSIS_2019_September.zip"
snapshot.zip <- paste0("https://files.isric.org/public/wosis_snapshot/",
zip.file.name)
target.zip <- paste0(wosis.dir.name, "/", zip.file.name)
if (!file.exists(target.zip)) {
  download.file(snapshot.zip, destfile=target.zip)
}
```

Unpack the file; this will take some time. If already unpacked from a previous run, no need to do it again.

```
if (!file.exists(paste0(wosis.dir.name, "/wosis_201909.gpkg"))) {
  system.time(unzip(target.zip, exdir=wosis.dir.name, junkpaths=TRUE))
}
```

```
##    user  system elapsed
##   6.019   7.955  25.604
```

```
list.files(wosis.dir.name)
```

```
## [1] "ReadmeFirst_WoSIS_2019dec04.pdf"  "WoSIS_2019_September.zip"
## [3] "wosis_201909_attributes.tsv"      "wosis_201909_layers_chemical.tsv"
## [5] "wosis_201909_layers_physical.tsv" "wosis_201909_profiles.tsv"
## [7] "wosis_201909.gpkg"
```

This results in about *20x more storage, 3.8Gb*. It includes four *tab-delimited* flat text files with extension `.tsv` (1.8 Gb) and one *Geopackage*[1] with extension `.gpkg` (2.2 Gb). These provide two ways to access the same information.

[1] https://www.geopackage.org

- `wosis_201909.gpkg`: the Geopackage

- `wosis_201909_attributes.tsv` : List of attributes with their codes, whether each is a site or horizon property, the unit of measurement, the number of profiles or layers, the inferred uncertainty; these attributes are used in the other files

- `wosis_201909_layers_chemical.tsv` : Chemical properties indexed by profile and layer
- `wosis_201909_layers_physical.tsv` : Physical properties indexed by profile and layer

- `wosis_201909_profiles.tsv` : Profile information, including coordinates, primary key, and classification
  The file `Readme_first_WoSIS_snapshot_September_2019.pdf` explains this dataset, please take some time to read it.

Working with the tab-separated value tables

WoSIS profiles

The profile-level information is stored in file `wosis_201909_profiles.tsv`.

```
profiles <- read_tsv(paste0(wosis.dir.name, "/wosis_201909_profiles.tsv"))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
## cols(
##   .default = col_character(),
##   profile_id = col_double(),
##   geom_accuracy = col_double(),
##   latitude = col_double(),
##   longitude = col_double(),
##   dsds = col_double(),
##   cfao_version = col_double(),
##   cwrb_version = col_double(),
##   cstx_version = col_logical()
## )
## ℹ Use `spec()` for the full column specifications.

## Warning: 21314 parsing failures.
##  row          col           expected actual
file
## 1063 cstx_version 1/0/T/F/TRUE/FALSE   1990
'./wosis2019/wosis_201909_profiles.tsv'
## 2650 cstx_version 1/0/T/F/TRUE/FALSE   1975
'./wosis2019/wosis_201909_profiles.tsv'
```

```
## 2674 cstx_version 1/0/T/F/TRUE/FALSE   1975
'./wosis2019/wosis_201909_profiles.tsv'
## 3725 cstx_version 1/0/T/F/TRUE/FALSE   1975
'./wosis2019/wosis_201909_profiles.tsv'
## 3764 cstx_version 1/0/T/F/TRUE/FALSE   1999
'./wosis2019/wosis_201909_profiles.tsv'
## .... ............ .................. ......
.....................................
## See problems(...) for more details.

dim(profiles)

## [1] 196498     23

names(profiles)

##  [1] "profile_id"                    "dataset_id"
##  [3] "country_id"                    "country_name"
##  [5] "geom_accuracy"                 "latitude"
##  [7] "longitude"                     "dsds"
##  [9] "cfao_version"                  "cfao_major_group_code"
## [11] "cfao_major_group"              "cfao_soil_unit_code"
## [13] "cfao_soil_unit"                "cwrb_version"
## [15] "cwrb_reference_soil_group_code" "cwrb_reference_soil_group"
## [17] "cwrb_prefix_qualifier"         "cwrb_suffix_qualifier"
## [19] "cstx_version"                  "cstx_order_name"
## [21] "cstx_suborder"                 "cstx_great_group"
## [23] "cstx_subgroup"
```

This has the same information as the geopackage, but in addition the profile ID, which can be used to link with the attribute tables.

List the countries and contributing datasets:

```
length(unique(profiles$country_name))

## [1] 175

head(table(profiles$country_name))

##
## Afghanistan     Albania     Algeria      Angola  Antarctica   Argentina
##          19          97          10        1169           9         244

length(unique(profiles$dataset_id))

## [1] 167

head(table(profiles$dataset_id))

##
##                   {ACTD,AF-AfSP}                {AF-AfSIS-phase1}
##                              794                             1902
```

```
## {AF-AfSP,AGIS,SAF-SOTER,ZA-SOTER}           {AF-AfSP,AGIS,ZA-SOTER}
##                                325                               284
##         {AF-AfSP,BJSOTER,WD-WISE}                 {AF-AfSP,BORENA}
##                                710                               210
```

Profiles come from 175 countries (variously defined) and 167 contributing datasets. The list of sources (i.e., databases contributing to WoSIS) is internal to ISRIC, please ask.

Profiles may be classified in one or more of the the three soil classification systems, as specified when the profiles were added to WoSIS. Note that there had been no attempt to re-classify or correlate.

```
table(profiles$cstx_order_name)

##
##    Alfisol    Andisol   Aridisol    Entisol    Gelisol   Histosol
Inceptisol
##       8303        586        935       2914         76        557
3958
##   Mollisol     Oxisol   Spodosol  Spodosols    Ultisol   Vertisol
Vertisols
##       6547        460        756        148       3712        776
5

table(profiles$cwrb_reference_soil_group)

##
##     Acrisols Albeluvisols      Alisols     Andosols   Anthrosols
Arenosols
##         1227          109          459          408          259
1608
##    Calcisols    Cambisols   Chernozems     Cryosols     Durisols
Ferralsols
##         1434         3035          728          120           43
875
##    Fluvisols     Gleysols    Gypsisols    Histosols  Kastanozems
Leptosols
##         1076         1085          118          244          308
1425
##     Lixisols     Luvisols     Nitisols    Phaeozems    Planosols
Plinthosols
##          789         3276          333         1441          313
142
##      Podzols     Regosols     Retisols   Solonchaks     Solonetz
Stagnosols
##          375         1919            6          333          374
73
##    Umbrisols    Vertisols
##          522         2207

table(profiles$cfao_major_group)
```

```
##
##      Acrisols       Alisols      Andosols    Anthrosols     Arenosols
##          1381           326           392           241          1450
##     Calcisols     Cambisols    Chernozems    Ferralsols     Fluvisols
##           545          2843           580           840          1064
##      Gleysols      Greyzems     Gypsisols     Histosols   Kastanozems
##          1082            76            44           225           304
##     Leptosols     Lithosols      Lixisols      Luvisols      Nitisols
##           462           104           449          2999           220
##      Nitosols     Phaeozems     Planosols   Plinthosols       Podzols
##           114          1305           307            27           300
## Podzoluvisols       Rankers      Regosols     Rendzinas    Solonchaks
##           308            48          1740           501           400
##      Solonetz     Vertisols      Xerosols     Yermosols
##           354          1897           632           330

sum(is.na(profiles$cfao_major_group))

## [1] 172608
```

Most profiles are missing classifications in any system; the percentage w/o any classification is:

```
round(100*(length(intersect(which(is.na(profiles$cfao_major_group)),
          intersect(which(is.na(profiles$cwrb_reference_soil_group)),
          which(is.na(profiles$cstx_order_name))))))/dim(profiles)[1],1)

## [1] 73.2
```

The profiles all have coördinates (fields `longitude`, `latitude`) and so can be converted to spatial objects (Simple Features or `sp`); the Coördinate Reference System (CRS) is given as geographic coördinates on the WGS84 datum the WoSIS documentation. However, the points come from many sources and may have used other CRS, and many were not georeferenced with high accuracy.

The accuracy of the geographical coördinates is given in decimal degrees, according to the precision reported in the original source, which may have been in degrees-minutes-seconds or decimal degrees. This does not take into account any datum shifts.

```
table(profiles$geom_accuracy)

##
##    1e-07    1e-06    9e-06    1e-05    2e-05  2.8e-05    5e-05    1e-04
##     1345    84728      217    71925      276     1202      621     9158
## 0.000278    4e-04    5e-04 0.000556 0.000909    0.001 0.001389   0.0014
##     2882        1      545       63       10     4607        7      238
## 0.001667    0.002   0.0025 0.002778    0.003   0.0035    0.004   0.0045
##        6      236       83       59      110       57      166       56
##    0.005    0.006 0.008333   0.0085 0.009091     0.01    0.014    0.015
##      276       18       25       16       55     9507        1        6
## 0.016667  0.01667    0.017     0.02    0.025     0.03  0.03333     0.04
```

```
##     1843        4        1      148      102      161        5       76
##     0.05     0.08  0.08333 0.083333    0.085      0.1     0.15      0.2
##      148       19       22       36        1     3885        3       44
##     0.24     0.25      0.3      0.4      0.5      0.7      0.9        1
##        1       10        4        9       14        1        1     1458
```

So, you could select only the high-precision points for spatial modelling, but all points for statistical summaries.

Make a spatial version of the profile database:

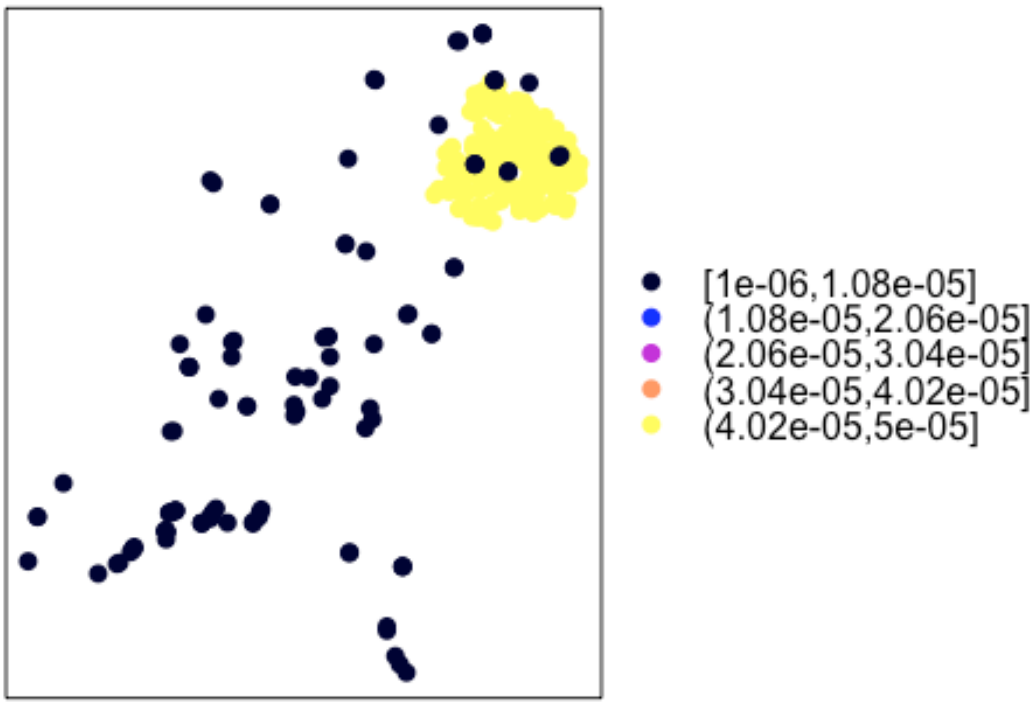
```
profiles.sp <- data.frame(profiles)
coordinates(profiles.sp) <- c("longitude", "latitude")
proj4string(profiles.sp) <- CRS("+init=epsg:4326")
str(profiles.sp)
```

```
## Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
##   ..@ data       :'data.frame':  196498 obs. of  21 variables:
##   .. ..$ profile_id                   : num [1:196498] 36897 36898 36899
36900 36901 ...
##   .. ..$ dataset_id                   : chr [1:196498] "{BE-UplandsI}"
"{BE-UplandsI}" "{BE-UplandsI}" "{BE-UplandsI}" ...
##   .. ..$ country_id                   : chr [1:196498] "BE" "BE" "BE"
"BE" ...
##   .. ..$ country_name                 : chr [1:196498] "Belgium"
"Belgium" "Belgium" "Belgium" ...
##   .. ..$ geom_accuracy                : num [1:196498] 1e-06 1e-06 1e-06
1e-06 1e-06 1e-06 1e-06 1e-06 1e-06 1e-06 ...
##   .. ..$ dsds                         : num [1:196498] 100 97 109 94 100
103 103 94 106 100 ...
##   .. ..$ cfao_version                 : num [1:196498] NA NA NA NA NA NA
NA NA NA NA ...
##   .. ..$ cfao_major_group_code        : chr [1:196498] NA NA NA NA ...
##   .. ..$ cfao_major_group             : chr [1:196498] NA NA NA NA ...
##   .. ..$ cfao_soil_unit_code          : chr [1:196498] NA NA NA NA ...
##   .. ..$ cfao_soil_unit               : chr [1:196498] NA NA NA NA ...
##   .. ..$ cwrb_version                 : num [1:196498] NA NA NA NA NA NA
NA NA NA NA ...
##   .. ..$ cwrb_reference_soil_group_code: chr [1:196498] NA NA NA NA ...
##   .. ..$ cwrb_reference_soil_group    : chr [1:196498] NA NA NA NA ...
##   .. ..$ cwrb_prefix_qualifier        : chr [1:196498] NA NA NA NA ...
##   .. ..$ cwrb_suffix_qualifier        : chr [1:196498] NA NA NA NA ...
##   .. ..$ cstx_version                 : logi [1:196498] NA NA NA NA NA NA
...
##   .. ..$ cstx_order_name              : chr [1:196498] NA NA NA NA ...
##   .. ..$ cstx_suborder                : chr [1:196498] NA NA NA NA ...
##   .. ..$ cstx_great_group             : chr [1:196498] NA NA NA NA ...
##   .. ..$ cstx_subgroup                : chr [1:196498] NA NA NA NA ...
##   ..@ coords.nrs : int [1:2] 7 6
##   ..@ coords     : num [1:196498, 1:2] 4.67 4.46 4.69 4.68 4.47 ...
##   .. ..- attr(*, "dimnames")=List of 2
```

```
##   .. .. ..$ : NULL
##   .. .. ..$ : chr [1:2] "longitude" "latitude"
##   ..@ bbox       : num [1:2, 1:2] -172.4 -77.8 179.2 81.4
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:2] "longitude" "latitude"
##   .. .. ..$ : chr [1:2] "min" "max"
##   ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
##   .. .. ..@ projargs: chr "+proj=longlat +datum=WGS84 +no_defs"
```

Show a map of the higher-precision profiles in the Netherlands:

```
dim(profiles.hi <- profiles %>%
      dplyr::filter(country_id=="NL") %>%
      dplyr::filter(geom_accuracy < 1/3600))

## [1] 256  23

coordinates(profiles.hi) <- c("longitude", "latitude")
proj4string(profiles.hi) <- CRS("+init=epsg:4326")
spplot(profiles.hi, zcol="geom_accuracy", key.space="right")
```



An important attribute at the site level is the sampling depth:

```
profiles %>% select(profile_id, country_id, longitude, latitude,
geom_accuracy, dsds)
```

```
## # A tibble: 196,498 x 6
##    profile_id country_id longitude latitude geom_accuracy  dsds
##         <dbl> <chr>          <dbl>    <dbl>         <dbl> <dbl>
##  1      36897 BE              4.67     50.6      0.000001   100
##  2      36898 BE              4.46     50.6      0.000001    97
##  3      36899 BE              4.69     50.6      0.000001   109
##  4      36900 BE              4.68     50.6      0.000001    94
##  5      36901 BE              4.47     50.6      0.000001   100
##  6      36902 BE              4.62     50.6      0.000001   103
##  7      36903 BE              4.77     50.6      0.000001   103
##  8      36904 BE              4.86     50.6      0.000001    94
##  9      36905 BE              4.64     50.5      0.000001   106
## 10      36906 BE              4.61     50.5      0.000001   100
## # … with 196,488 more rows
```

WoSIS attribute tables

The values of the attributes are at either the profile (site) level or the layer (usually a pedogenetic horizon) level. There is also a table with the list of attributes and their description. The profiles attributes were discussed in the previous section.

The layer level attributes are in two (very large) text files, each about 850 Mb.

The fields are separated ('delimited') by tabulation characters ('tabs'). These files can be read into R with the `read.table` function from the `utils` package, with appropriate arguments: the separator `sep` is a tabulation mark `\t`, and there is a header line. Strings are read as-is, not converted to factors (categorical variables).

List of attributes

The first table is the list of attributes, which points to the other files with the attribute values and the corresponding profile and layer:

```
attributes <- read.table(paste0(wosis.dir.name,
"/wosis_201909_attributes.tsv"),
                         header=TRUE,
                         sep="\t",
                         stringsAsFactors=FALSE)
str(attributes)

## 'data.frame':    52 obs. of  8 variables:
##  $ code       : chr  "BDFI33" "BDFIAD" "BDFIFM" "BDFIOD" ...
##  $ type       : chr  "Horizon" "Horizon" "Horizon" "Horizon" ...
##  $ attribute  : chr  "Bulk density fine earth - 33 kPa" "Bulk density fine
earth - air dry" "Bulk density fine earth - field moist" "Bulk density fine
earth - oven dry" ...
##  $ unit       : chr  "kg/dm³" "kg/dm³" "kg/dm³" "kg/dm³" ...
##  $ profiles   : int  14924 1786 5279 25124 26268 0 0 14588 54278 6422 ...
##  $ layers     : int  78215 8471 14219 122693 154901 0 0 75422 295688 23691
...
##  $ description: chr  "Bulk density of the fine earth fraction*,
```

```
equilibrated at 33 kPa" "Bulk density of the fine earth fraction*, air dried"
"Bulk density of the fine earth fraction*, field moist" "Bulk density of the
fine earth fraction*, oven dry" ...
##  $ accuracy   : num  35 35 35 35 35 35 35 35 20 20 ...
```

List the attribute codes, names, and units of measure:

```
attributes[, c("code", "type", "attribute", "unit")]
```

```
##      code    type                                    attribute       unit
## 1  BDFI33 Horizon           Bulk density fine earth - 33 kPa     kg/dm³
## 2  BDFIAD Horizon          Bulk density fine earth - air dry     kg/dm³
## 3  BDFIFM Horizon      Bulk density fine earth - field moist     kg/dm³
## 4  BDFIOD Horizon         Bulk density fine earth - oven dry     kg/dm³
## 5  BDWS33 Horizon           Bulk density whole soil - 33 kPa     kg/dm³
## 6  BDWSAD Horizon          Bulk density whole soil - air dry     kg/dm³
## 7  BDWSFM Horizon      Bulk density whole soil - field moist     kg/dm³
## 8  BDWSOD Horizon         Bulk density whole soil - oven dry     kg/dm³
## 9  CECPH7 Horizon Cation exchange capacity - buffered at pH7 cmol(c)/kg
## 10 CECPH8 Horizon Cation exchange capacity - buffered at pH8 cmol(c)/kg
## 11   CFAO    Site                      Soil classification FAO   unitless
## 12   CFGR Horizon         Coarse fragments gravimetric total     g/100g
## 13   CFVO Horizon          Coarse fragments volumetric total cm³/100cm³
## 14   CLAY Horizon                                  Clay total     g/100g
## 15   CSTX    Site          Soil classification Soil taxonomy   unitless
## 16   CWRB    Site                    Soil classification WRB   unitless
## 17   DSDS    Site                    Depth of soil - sampled         cm
## 18   ECEC Horizon         Effective cation exchange capacity cmol(c)/kg
## 19 ELCO20 Horizon        Electrical conductivity - ratio 1:2       dS/m
## 20 ELCO25 Horizon      Electrical conductivity - ratio 1:2.5       dS/m
## 21 ELCO50 Horizon        Electrical conductivity - ratio 1:5       dS/m
## 22 ELCOSP Horizon  Electrical conductivity - saturated paste       dS/m
## 23 NITKJD Horizon                          Total nitrogen (N)       g/kg
## 24   ORGC Horizon                              Organic carbon       g/kg
## 25   PHAQ Horizon                                      pH H2O   unitless
## 26   PHCA Horizon                                   pH CaCl2   unitless
## 27   PHKC Horizon                                     pH KCl   unitless
## 28   PHNF Horizon                                     pH NaF   unitless
## 29 PHPBYI Horizon                   Phosphorus (P) - Bray I      mg/kg
## 30 PHPMH3 Horizon                 Phosphorus (P) - Mehlich 3      mg/kg
## 31 PHPOLS Horizon                     Phosphorus (P) - Olsen      mg/kg
## 32 PHPRTN Horizon                 Phosphorus (P) - retention      mg/kg
## 33 PHPTOT Horizon                     Phosphorus (P) - total      mg/kg
## 34 PHPWSL Horizon             Phosphorus (P) - water soluble      mg/kg
## 35   SAND Horizon                                  Sand total     g/100g
## 36   SILT Horizon                                  Silt total     g/100g
## 37   TCEQ Horizon         Calcium carbonate equivalent total       g/kg
## 38   TOTC Horizon                          Total carbon (C)       g/kg
## 39 WG0006 Horizon        Water retention gravimetric - 6 kPa     g/100g
## 40 WG0010 Horizon       Water retention gravimetric - 10 kPa     g/100g
## 41 WG0033 Horizon       Water retention gravimetric - 33 kPa     g/100g
```

```
## 42 WG0100 Horizon      Water retention gravimetric - 100 kPa     g/100g
## 43 WG0200 Horizon      Water retention gravimetric - 200 kPa     g/100g
## 44 WG0500 Horizon      Water retention gravimetric - 500 kPa     g/100g
## 45 WG1500 Horizon     Water retention gravimetric - 1500 kPa     g/100g
## 46 WV0006 Horizon         Water retention volumetric - 6 kPa cm³/100cm³
## 47 WV0010 Horizon        Water retention volumetric - 10 kPa cm³/100cm³
## 48 WV0033 Horizon        Water retention volumetric - 33 kPa cm³/100cm³
## 49 WV0100 Horizon       Water retention volumetric - 100 kPa cm³/100cm³
## 50 WV0200 Horizon       Water retention volumetric - 200 kPa cm³/100cm³
## 51 WV0500 Horizon       Water retention volumetric - 500 kPa cm³/100cm³
## 52 WV1500 Horizon      Water retention volumetric - 1500 kPa cm³/100cm³

table(attributes$type)

##
## Horizon    Site
##      48       4
```

Four attributes are at "site" level, and are found in the `profiles` table discussed in the previous section. The other 48 are per-"horizon" and are found in the `physical` or `chemical` attribute tables, see below.

The codes are the first part of seven field names per attribute in the attribute tables. For example `CLAY` becomes part of names like `clay_method` in the physical attributes table. Each attribute has several fields, with the tail of the name as:

- `value` – one or more values, in the format {1:value; 2:value…}, which are duplicate measurements
- `value_avg` – the average of the values
- `method` – text description of the analytical method
- `date` – one or more values, in the format {1:yyyy-mm-dd; 2:yyyy-mm-dd…}, which are the dates each of the duplicate measurements was added to the database (not the original measurement date, nor the field sampling date)
- `dataset_id` – text code of original database
- `profile_code` – text code of profile from original database
- `licence` – text string of the Creative Commons[2] license for this value, e.g. `CC-BY-NC`

So for example in the `physical` table (see below) for the first attribute `bdfi33`, there are the following fields:

- `bdfi33_value`
- `bdfi33_value_avg`
- `bdfi33_method`
- `bdfi33_date`

---

[2] https://creativecommons.org/licenses/

- `bdfi33_dataset_id`
- `bdfi33_profile_code`
- `bdfi33_licence`

How many profiles/layers of each?

```
attributes[, c("code", "profiles", "layers")]
```

```
##      code profiles layers
## 1  BDFI33    14924  78215
## 2  BDFIAD     1786   8471
## 3  BDFIFM     5279  14219
## 4  BDFIOD    25124 122693
## 5  BDWS33    26268 154901
## 6  BDWSAD        0      0
## 7  BDWSFM        0      0
## 8  BDWSOD    14588  75422
## 9  CECPH7    54278 295688
## 10 CECPH8     6422  23691
## 11   CFAO    23890      0
## 12   CFGR    39527 203083
## 13   CFVO    45918 235002
## 14   CLAY   141640 607861
## 15   CSTX    21314      0
## 16   CWRB    26664      0
## 17   DSDS   196381      0
## 18   ECEC    31708 132922
## 19 ELCO20     8010  44596
## 20 ELCO25     3313  15134
## 21 ELCO50    23093  90944
## 22 ELCOSP    19434  73517
## 23 NITKJD    65356 216362
## 24   ORGC   110856 471301
## 25   PHAQ   130986 613322
## 26   PHCA    66921 314230
## 27   PHKC    32920 150447
## 28   PHNF     4978  25448
## 29 PHPBYI    10735  40486
## 30 PHPMH3     1446   7242
## 31 PHPOLS     2162   8434
## 32 PHPRTN     4636  23917
## 33 PHPTOT     4022  12976
## 34 PHPWSL      283   1242
## 35   SAND   105547 491810
## 36   SILT   133938 575913
## 37   TCEQ    51991 222242
## 38   TOTC    32662 109953
## 39 WG0006      863   4264
## 40 WG0010     3357  14739
## 41 WG0033    21116  96354
```

```
## 42 WG0100      696   3762
## 43 WG0200     4418  28239
## 44 WG0500      344   1716
## 45 WG1500    34365 187176
## 46 WV0006        9     26
## 47 WV0010     1469   5434
## 48 WV0033     5987  17801
## 49 WV0100      747   2559
## 50 WV0200        3      9
## 51 WV0500      703   1763
## 52 WV1500     6149  17542
```

Each one has a description, e.g.,

```
attributes[1:5, c("code",  "description")]
```

```
##     code
## 1 BDFI33
## 2 BDFIAD
## 3 BDFIFM
## 4 BDFIOD
## 5 BDWS33
##
description
## 1                 Bulk density of the fine earth fraction*, equilibrated
at 33 kPa
## 2                              Bulk density of the fine earth fraction*,
air dried
## 3                            Bulk density of the fine earth fraction*,
field moist
## 4                               Bulk density of the fine earth fraction*,
oven dry
## 5 Bulk density of the whole soil including coarse fragments, equilibrated
at 33 kPa
```

And each has an estimated accuracy (see below for explanation):

```
attributes[1:5, c("code", "attribute", "accuracy")]
```

```
##     code                              attribute accuracy
## 1 BDFI33      Bulk density fine earth - 33 kPa       35
## 2 BDFIAD     Bulk density fine earth - air dry       35
## 3 BDFIFM Bulk density fine earth - field moist       35
## 4 BDFIOD    Bulk density fine earth - oven dry       35
## 5 BDWS33      Bulk density whole soil - 33 kPa       35
```

Find the attributes related to P:

```
ix <- grep("Phosphorus", attributes$attribute)
attributes[ix, c("code", "attribute", "profiles", "layers", "unit",
"accuracy")]
```

```
##      code                       attribute profiles layers  unit accuracy
## 29 PHPBYI        Phosphorus (P) - Bray I    10735  40486 mg/kg       40
## 30 PHPMH3     Phosphorus (P) - Mehlich 3     1446   7242 mg/kg       25
## 31 PHPOLS         Phosphorus (P) - Olsen     2162   8434 mg/kg       25
## 32 PHPRTN     Phosphorus (P) - retention     4636  23917 mg/kg       20
## 33 PHPTOT         Phosphorus (P) - total     4022  12976 mg/kg       15
## 34 PHPWSL Phosphorus (P) - water soluble      283   1242 mg/kg       15
```

```
attributes[ix, "description"]
```

```
## [1] "Measured according to the Bray-I method, a combination of HCl and NH4
F to remove easily acid soluble P forms, largely Al- and Fe-phosphates (for
acid soils)"
## [2] "Measured according to the Mehlich-3 extractant, a combination of
acids (acetic [HOAc] and nitric [HNO3]), salts (ammonium fluoride [NH4F] and
ammonium nitrate [NH4 NO3]), and the chelating agent
ethylenediaminetetraacetic acid (EDTA); considered suitable for removing P
and other elements in acid and neutral soils"
## [3] "Measured according to the P-Olsen method:  0.5 M sodium bicarbonate
(NaHCO3) solution at a pH of 8.5 to extract P from calcareous, alkaline, and
neutral soils"
## [4] "Retention measured according to the New Zealand method"
## [5] "Determined with a very strong acid (aqua regia and sulfuric
acid/nitric acid)"
## [6] "Measured in 1:x soil:water solution (mainly determines P in dissolved
forms)"
```

These are total P, or extractable P by various strengths of extractant. Especially interesting here is the `accuracy` field, explained in $\S 2.2.3$ of the procedures manual.

“The precision and accuracy of results from laboratory measurements can be derived from the random error and systematic error in repeated experiments on reference materials or with reference methods… For measurements that use other devices, such as GPS and soil moisture sensors, the accuracy can be obtained from manufacturers, literature and even expert knowledge.”

In the WoSIS snapshot, there is no attempt to give the accuracy of each measurement individually. Instead, expert knowledge applied to various datasets of repeated measurements has been used to estimate a typical accuracy of each method, see the `attributes` table, above.

This is given in the same units as the attribute, here mg kg^{-1}.

So for the P-related measurements, total and water-soluble are considered in general the most accurate, 15mg kg^{-1}, compared to a total P median value 118 and a mean value of 284.7 in the database (see below under “Chemical attributes”). Bray I is considerably less accurate than Mehlich 3 or Olsen.

Physical attributes

The remaining two attributes files are text files with per-layer attribute. Each entry has a two-field key: profile and layer. They can be linked to the profiles via a foreign key. They must be read in to a single structure, there is no way to subset them during import.

To read in the physical attributes to the R workspace,we use the `readr` function `read_tsv`, i.e. "read tab-delimited text file." This function makes a guess of the data type of each field, by reading the first "few" records (by default 1000). However in tests of this, because of the variety of value formats in the fields, the guesses do not work very well. Therefore we were forced to define an explicit specification of each of the 195 column's data type, using the optional `col_types` argument, and referring to the documentation.

Each column is specified as onne character: `c` = character, `i` = integer, `n` = number, `d` = double-precision number, `l` = logical, `f` = factor, `D` = date, `T` = date time, `t` = time, `?` = guess, or _ to skip the column.

```
physical <- readr::read_tsv(paste0(wosis.dir.name,
"/wosis_201909_layers_physical.tsv"),

col_types="iiddclcdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdcc
ccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdcc
ccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdcc
ccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdcc
ccccdccccccdccccccdccccccdccccc")

## Warning: Unnamed `col_types` should have the same length as `col_names`.
Using
## smaller of the two.

dim(physical)

## [1] 702698    195
```

There are 702698, 195 layers of profiles with physical properties.

These are the attributes, as explained in the `attributes` table (see above), along with profile and horizon identificatiojn:

```
names(physical)

##   [1] "profile_id"          "profile_layer_id"    "upper_depth"
##   [4] "lower_depth"         "layer_name"          "litter"
##   [7] "bdfi33_value"        "bdfi33_value_avg"    "bdfi33_method"
##  [10] "bdfi33_date"         "bdfi33_dataset_id"   "bdfi33_profile_code"
##  [13] "bdfi33_licence"      "bdfiad_value"        "bdfiad_value_avg"
##  [16] "bdfiad_method"       "bdfiad_date"         "bdfiad_dataset_id"
##  [19] "bdfiad_profile_code" "bdfiad_licence"      "bdfifm_value"
##  [22] "bdfifm_value_avg"    "bdfifm_method"       "bdfifm_date"
##  [25] "bdfifm_dataset_id"   "bdfifm_profile_code" "bdfifm_licence"
##  [28] "bdfiod_value"        "bdfiod_value_avg"    "bdfiod_method"
```

```
##  [31] "bdfiod_date"          "bdfiod_dataset_id"   "bdfiod_profile_code"
##  [34] "bdfiod_licence"       "bdws33_value"        "bdws33_value_avg"
##  [37] "bdws33_method"        "bdws33_date"         "bdws33_dataset_id"
##  [40] "bdws33_profile_code" "bdws33_licence"      "bdwsad_value"
##  [43] "bdwsad_value_avg"     "bdwsad_method"       "bdwsad_date"
##  [46] "bdwsad_dataset_id"    "bdwsad_profile_code" "bdwsad_licence"
##  [49] "bdwsfm_value"         "bdwsfm_value_avg"    "bdwsfm_method"
##  [52] "bdwsfm_date"          "bdwsfm_dataset_id"   "bdwsfm_profile_code"
##  [55] "bdwsfm_licence"       "bdwsod_value"        "bdwsod_value_avg"
##  [58] "bdwsod_method"        "bdwsod_date"         "bdwsod_dataset_id"
##  [61] "bdwsod_profile_code" "bdwsod_licence"      "clay_value"
##  [64] "clay_value_avg"       "clay_method"         "clay_date"
##  [67] "clay_dataset_id"      "clay_profile_code"   "clay_licence"
##  [70] "cfgr_value"           "cfgr_value_avg"      "cfgr_method"
##  [73] "cfgr_date"            "cfgr_dataset_id"     "cfgr_profile_code"
##  [76] "cfgr_licence"         "cfvo_value"          "cfvo_value_avg"
##  [79] "cfvo_method"          "cfvo_date"           "cfvo_dataset_id"
##  [82] "cfvo_profile_code"    "cfvo_licence"        "sand_value"
##  [85] "sand_value_avg"       "sand_method"         "sand_date"
##  [88] "sand_dataset_id"      "sand_profile_code"   "sand_licence"
##  [91] "silt_value"           "silt_value_avg"      "silt_method"
##  [94] "silt_date"            "silt_dataset_id"     "silt_profile_code"
##  [97] "silt_licence"         "wg0100_value"        "wg0100_value_avg"
## [100] "wg0100_method"        "wg0100_date"         "wg0100_dataset_id"
## [103] "wg0100_profile_code" "wg0100_licence"      "wg0010_value"
## [106] "wg0010_value_avg"     "wg0010_method"       "wg0010_date"
## [109] "wg0010_dataset_id"    "wg0010_profile_code" "wg0010_licence"
## [112] "wg1500_value"         "wg1500_value_avg"    "wg1500_method"
## [115] "wg1500_date"          "wg1500_dataset_id"   "wg1500_profile_code"
## [118] "wg1500_licence"       "wg0200_value"        "wg0200_value_avg"
## [121] "wg0200_method"        "wg0200_date"         "wg0200_dataset_id"
## [124] "wg0200_profile_code" "wg0200_licence"      "wg0033_value"
## [127] "wg0033_value_avg"     "wg0033_method"       "wg0033_date"
## [130] "wg0033_dataset_id"    "wg0033_profile_code" "wg0033_licence"
## [133] "wg0500_value"         "wg0500_value_avg"    "wg0500_method"
## [136] "wg0500_date"          "wg0500_dataset_id"   "wg0500_profile_code"
## [139] "wg0500_licence"       "wg0006_value"        "wg0006_value_avg"
## [142] "wg0006_method"        "wg0006_date"         "wg0006_dataset_id"
## [145] "wg0006_profile_code" "wg0006_licence"      "wv0100_value"
## [148] "wv0100_value_avg"     "wv0100_method"       "wv0100_date"
## [151] "wv0100_dataset_id"    "wv0100_profile_code" "wv0100_licence"
## [154] "wv0010_value"         "wv0010_value_avg"    "wv0010_method"
## [157] "wv0010_date"          "wv0010_dataset_id"   "wv0010_profile_code"
## [160] "wv0010_licence"       "wv1500_value"        "wv1500_value_avg"
## [163] "wv1500_method"        "wv1500_date"         "wv1500_dataset_id"
## [166] "wv1500_profile_code" "wv1500_licence"      "wv0200_value"
## [169] "wv0200_value_avg"     "wv0200_method"       "wv0200_date"
## [172] "wv0200_dataset_id"    "wv0200_profile_code" "wv0200_licence"
## [175] "wv0033_value"         "wv0033_value_avg"    "wv0033_method"
## [178] "wv0033_date"          "wv0033_dataset_id"   "wv0033_profile_code"
```

```
## [181] "wv0033_licence"      "wv0500_value"        "wv0500_value_avg"
## [184] "wv0500_method"       "wv0500_date"         "wv0500_dataset_id"
## [187] "wv0500_profile_code" "wv0500_licence"      "wv0006_value"
## [190] "wv0006_value_avg"    "wv0006_method"       "wv0006_date"
## [193] "wv0006_dataset_id"   "wv0006_profile_code" "wv0006_licence"
```

Examine the format of a single attribute along some profiles:

```
(.clay.fields <- which(substr(names(physical), 1, 4)=="clay"))

## [1] 63 64 65 66 67 68 69

data.frame(physical[1, .clay.fields])

##   clay_value clay_value_avg
## 1   {1:5.90}            5.9
##
clay_method
## 1 {"1:instrument = pipette, size = 0 - 0.002 mm, dispersion = Sodium
hexametaphosphate [(NaPO3)6] - Calgon type (ultrasonic treatment might be
included), treatment = Hydrogen peroxide [H2O2] plus mild Acetic acid
[CH3COOH] / Sodium acetate [CH3COONa] buffer treatments (if pH-H2O > 6.5),
sample pretreatment = sieved over 2 mm sieve"}
##        clay_date clay_dataset_id clay_profile_code clay_licence
## 1 {1:1997-09-01}         WD-ISIS             BF001     CC-BY-NC

(.clay.values.fields <- which(substr(names(physical), 1, 10)=="clay_value"))

## [1] 63 64

data.frame(physical[,1:5], .clay.values.fields)[1:12,]

##    profile_id profile_layer_id upper_depth lower_depth layer_name
## 1       47010                1           0          21         Ap
## 2       47010                2          21          35         E1
## 3       47010                3          35          56         E2
## 4       47010                4          56          88         EB
## 5       47010                5          88         120         Bv
## 6       47381                6           0           9       <NA>
## 7       47381                7           9          20       <NA>
## 8       47381                8          20          35       <NA>
## 9       47381                9          35          60       <NA>
## 10      47381               10          60          90       <NA>
## 11      47381               11          90         116       <NA>
## 12      47555               12           0          17         Ap
##    .clay.values.fields
## 1                   63
## 2                   64
## 3                   63
## 4                   64
## 5                   63
## 6                   64
```

```
## 7                   63
## 8                   64
## 9                   63
## 10                  64
## 11                  63
## 12                  64
```

The format for an attribute is `{seq:val[,seq:val]}` where the `seq` is an integer on `[1...]` indicating which measurement number – note that there can be more than one measurement per property, e.g., repeated lab. measurements, and `val` is the numeric value.

The average of all measurements has its own field, here `clay_value_avg`, so if we only want the average, it is prepared for us.

With `dplyr` functions we can easily subset by attribute name. For example to see the hydrometer-based methods of measuring clay:

```
(clay.values <- physical %>% select(contains("clay")))
```

```
## # A tibble: 702,698 x 7
##    clay_value clay_value_avg clay_method               clay_date
clay_dataset_id
##    <chr>               <dbl> <chr>                     <chr>     <chr>
##  1 {1:5.90}              5.9 "{\"1:instrument = pipet… {1:1997-… WD-ISIS
##  2 {1:10.90}            10.9 "{\"1:instrument = pipet… {1:1997-… WD-ISIS
##  3 {1:19.00}            19   "{\"1:instrument = pipet… {1:1997-… WD-ISIS
##  4 {1:30.00}            30   "{\"1:instrument = pipet… {1:1997-… WD-ISIS
##  5 {1:31.50}            31.5 "{\"1:instrument = pipet… {1:1997-… WD-ISIS
##  6 {1:10.50}            10.5 "{\"1:instrument = pipet… {1:1997-… WD-ISIS
##  7 {1:6.90}              6.9 "{\"1:instrument = pipet… {1:1997-… WD-ISIS
##  8 {1:13.60}            13.6 "{\"1:instrument = pipet… {1:1997-… WD-ISIS
##  9 {1:32.00}            32   "{\"1:instrument = pipet… {1:1997-… WD-ISIS
## 10 {1:37.20}            37.2 "{\"1:instrument = pipet… {1:1997-… WD-ISIS
## # … with 702,688 more rows, and 2 more variables: clay_profile_code <chr>,
## #   clay_licence <chr>
```

```
length(clay.methods <- unique(clay.values$clay_method))
```

```
## [1] 133
```

```
head(clay.methods)
```

```
## [1] "{\"1:instrument = pipette, size = 0 - 0.002 mm, dispersion = Sodium
hexametaphosphate [(NaPO3)6] - Calgon type (ultrasonic treatment might be
included), treatment = Hydrogen peroxide [H2O2] plus mild Acetic acid
[CH3COOH] / Sodium acetate [CH3COONa] buffer treatments (if pH-H2O > 6.5),
sample pretreatment = sieved over 2 mm sieve\"}"
## [2] NA
## [3] "{\"1:size = 0 - 0.002 mm, instrument = pipette, sample pretreatment =
sieved over 2 mm sieve, treatment = Hydrogen peroxide [H2O2] plus mild Acetic
acid [CH3COOH] / Sodium acetate [CH3COONa] buffer treatments (if pH-H2O >
6.5), dispersion = Sodium hexametaphosphate [(NaPO3)6] - Calgon type
```

```
(ultrasonic treatment might be included)\"}"
## [4] "{\"1:size = 0 - 0.002 mm, instrument = pipette, dispersion = Sodium
hexametaphosphate [(NaPO3)6] - Calgon type (ultrasonic treatment might be
included), treatment = Hydrogen peroxide [H2O2] plus mild Acetic acid
[CH3COOH] / Sodium acetate [CH3COONa] buffer treatments (if pH-H2O > 6.5),
sample pretreatment = sieved over 2 mm sieve\"}"
## [5] "{\"1:size = 0 - 0.002 mm, instrument = pipette, dispersion = Sodium
hexametaphosphate [(NaPO3)6] - Calgon type (ultrasonic treatment might be
included), treatment = Hydrogen peroxide [H2O2] plus mild Acetic acid
[CH3COOH] / Sodium acetate [CH3COONa] buffer treatments (if pH-H2O > 6.5),
sample pretreatment = sieved over 2 mm sieve\",\"2:size = 0 - 0.002 mm,
instrument = pipette, dispersion = Sodium hexametaphosphate [(NaPO3)6] -
Calgon type (ultrasonic treatment might be included), treatment = Hydrogen
peroxide [H2O2] plus mild Acetic acid [CH3COOH] / Sodium acetate [CH3COONa]
buffer treatments (if pH-H2O > 6.5), sample pretreatment = sieved over 2 mm
sieve\"}"
## [6] "{\"1:instrument = pipette, size = 0 - 0.002 mm, dispersion = Sodium
hexametaphosphate [(NaPO3)6] - Calgon type (ultrasonic treatment might be
included), treatment = Hydrogen peroxide [H2O2] plus mild Acetic acid
[CH3COOH] / Sodium acetate [CH3COONa] buffer treatments (if pH-H2O > 6.5),
sample pretreatment = sieved over 2 mm sieve\",\"2:size = 0 - 0.002 mm,
instrument = pipette, dispersion = Sodium hexametaphosphate [(NaPO3)6] -
Calgon type (ultrasonic treatment might be included), treatment = Hydrogen
peroxide [H2O2] plus mild Acetic acid [CH3COOH] / Sodium acetate [CH3COONa]
buffer treatments (if pH-H2O > 6.5), sample pretreatment = sieved over 2 mm
sieve\"}"
```

```
length(clay.method.hydrometer.ix <- grep("hydrometer", clay.methods))
```

```
## [1] 18
```

```
clay.methods[clay.method.hydrometer.ix][1:3]
```

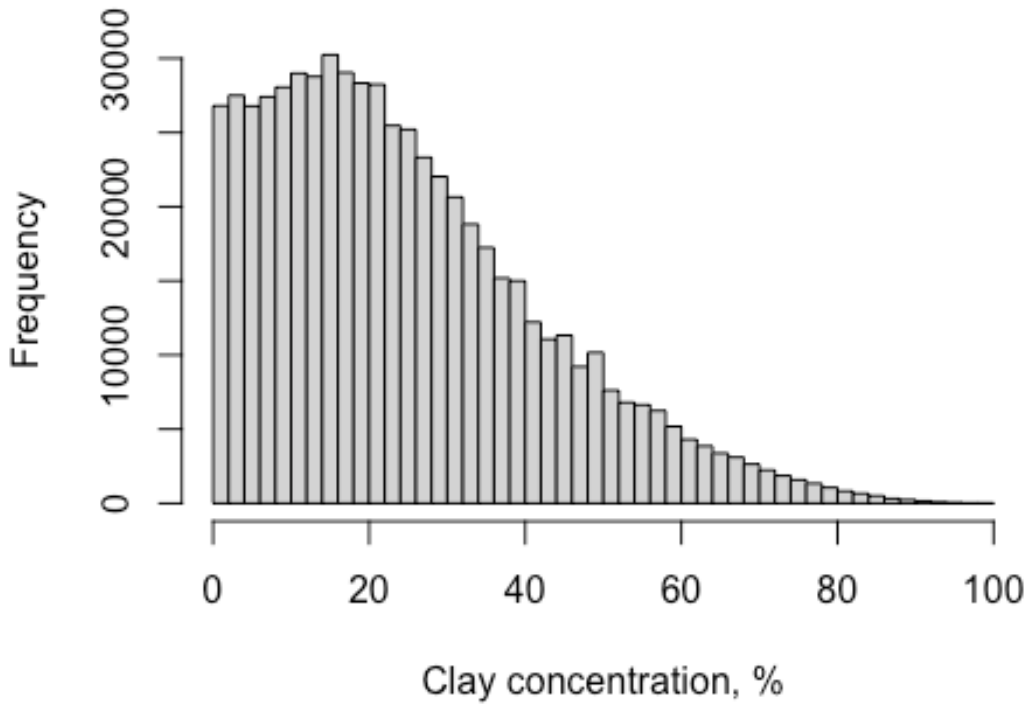
```
## [1] "{\"1:dispersion = not specified, treatment = not specified, size = 0
- 0.002 mm, sample pretreatment = sieved over 2 mm sieve, instrument =
hydrometer\"}"
## [2] "{\"1:treatment = not specified, dispersion = not specified, size = 0
- 0.002 mm, sample pretreatment = sieved over 2 mm sieve, instrument =
hydrometer\"}"
## [3] "{\"1:dispersion = not specified, treatment = not specified, size = 0
- 0.002 mm, instrument = hydrometer, sample pretreatment = sieved over 2 mm
sieve\"}"
```

We see the list of values of individual measurements, the average, the method used, the date of addition to WoSIS, the dataset ID, and the profile. This allows us to select by measurement method and dataset.

If we are satisfied with just the average, we make a table with that value, the profile, and the layer:

```
(clay.values <- physical %>% select(profile_id:layer_name, clay_value_avg))
```

```
## # A tibble: 702,698 x 6
##    profile_id profile_layer_id upper_depth lower_depth layer_name
clay_value_avg
##         <int>            <int>       <dbl>       <dbl> <chr>
<dbl>
##  1      47010                1           0          21 Ap
5.9
##  2      47010                2          21          35 E1
10.9
##  3      47010                3          35          56 E2
19
##  4      47010                4          56          88 EB
30
##  5      47010                5          88         120 Bv
31.5
##  6      47381                6           0           9 <NA>
10.5
##  7      47381                7           9          20 <NA>
6.9
##  8      47381                8          20          35 <NA>
13.6
##  9      47381                9          35          60 <NA>
32
## 10      47381               10          60          90 <NA>
37.2
## # … with 702,688 more rows

summary(clay.values)

##    profile_id     profile_layer_id   upper_depth       lower_depth
##  Min.   : 36897   Min.   :      1   Min.   :   0.00   Min.   :   0.00
##  1st Qu.:152941   1st Qu.: 698686   1st Qu.:   7.00   1st Qu.:  25.00
##  Median :187221   Median : 919160   Median :  32.00   Median :  58.00
##  Mean   :264068   Mean   :1303557   Mean   :  49.45   Mean   :  74.64
##  3rd Qu.:395019   3rd Qu.:1883754   3rd Qu.:  74.00   3rd Qu.: 105.00
##  Max.   :623817   Max.   :3178451   Max.   :3277.00   Max.   :3292.00
##                                     NA's   :444       NA's   :1003
##   layer_name        clay_value_avg
##  Length:702698      Min.   :  0.00
##  Class :character   1st Qu.: 11.30
##  Mode  :character   Median : 22.00
##                     Mean   : 25.09
##                     3rd Qu.: 35.30
##                     Max.   :100.00
##                     NA's   :94837

hist(clay.values$clay_value_avg, breaks=seq(0,100, by=2),
     xlab="Clay concentration, %", main="")
```



Chemical attributes

To read in the chemical attributes to the R workspace, we again need to supply the optional `col_types` argument to `readr::read_tsv`, thereby specifying the data type of each field.

```
chemical <- readr::read_tsv(paste0(wosis.dir.name,
"/wosis_201909_layers_chemical.tsv"),

col_types="iiddclcdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdcc
ccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdccccccdcc
ccccdccccc")
dim(chemical)

## [1] 788538    153

# spec(chemical)
names(chemical)

##   [1] "profile_id"          "profile_layer_id"    "upper_depth"
##   [4] "lower_depth"         "layer_name"          "litter"
##   [7] "tceq_value"          "tceq_value_avg"      "tceq_method"
##  [10] "tceq_date"           "tceq_dataset_id"     "tceq_profile_code"
##  [13] "tceq_licence"        "cecph7_value"        "cecph7_value_avg"
##  [16] "cecph7_method"       "cecph7_date"         "cecph7_dataset_id"
```

```
##  [19] "cecph7_profile_code" "cecph7_licence"      "cecph8_value"
##  [22] "cecph8_value_avg"    "cecph8_method"       "cecph8_date"
##  [25] "cecph8_dataset_id"   "cecph8_profile_code" "cecph8_licence"
##  [28] "ecec_value"          "ecec_value_avg"      "ecec_method"
##  [31] "ecec_date"           "ecec_dataset_id"     "ecec_profile_code"
##  [34] "ecec_licence"        "elco20_value"        "elco20_value_avg"
##  [37] "elco20_method"       "elco20_date"         "elco20_dataset_id"
##  [40] "elco20_profile_code" "elco20_licence"      "elco25_value"
##  [43] "elco25_value_avg"    "elco25_method"       "elco25_date"
##  [46] "elco25_dataset_id"   "elco25_profile_code" "elco25_licence"
##  [49] "elco50_value"        "elco50_value_avg"    "elco50_method"
##  [52] "elco50_date"         "elco50_dataset_id"   "elco50_profile_code"
##  [55] "elco50_licence"      "elcosp_value"        "elcosp_value_avg"
##  [58] "elcosp_method"       "elcosp_date"         "elcosp_dataset_id"
##  [61] "elcosp_profile_code" "elcosp_licence"      "orgc_value"
##  [64] "orgc_value_avg"      "orgc_method"         "orgc_date"
##  [67] "orgc_dataset_id"     "orgc_profile_code"   "orgc_licence"
##  [70] "phca_value"          "phca_value_avg"      "phca_method"
##  [73] "phca_date"           "phca_dataset_id"     "phca_profile_code"
##  [76] "phca_licence"        "phaq_value"          "phaq_value_avg"
##  [79] "phaq_method"         "phaq_date"           "phaq_dataset_id"
##  [82] "phaq_profile_code"   "phaq_licence"        "phkc_value"
##  [85] "phkc_value_avg"      "phkc_method"         "phkc_date"
##  [88] "phkc_dataset_id"     "phkc_profile_code"   "phkc_licence"
##  [91] "phnf_value"          "phnf_value_avg"      "phnf_method"
##  [94] "phnf_date"           "phnf_dataset_id"     "phnf_profile_code"
##  [97] "phnf_licence"        "phpbyi_value"        "phpbyi_value_avg"
## [100] "phpbyi_method"       "phpbyi_date"         "phpbyi_dataset_id"
## [103] "phpbyi_profile_code" "phpbyi_licence"      "phpmh3_value"
## [106] "phpmh3_value_avg"    "phpmh3_method"       "phpmh3_date"
## [109] "phpmh3_dataset_id"   "phpmh3_profile_code" "phpmh3_licence"
## [112] "phpols_value"        "phpols_value_avg"    "phpols_method"
## [115] "phpols_date"         "phpols_dataset_id"   "phpols_profile_code"
## [118] "phpols_licence"      "phprtn_value"        "phprtn_value_avg"
## [121] "phprtn_method"       "phprtn_date"         "phprtn_dataset_id"
## [124] "phprtn_profile_code" "phprtn_licence"      "phptot_value"
## [127] "phptot_value_avg"    "phptot_method"       "phptot_date"
## [130] "phptot_dataset_id"   "phptot_profile_code" "phptot_licence"
## [133] "phpwsl_value"        "phpwsl_value_avg"    "phpwsl_method"
## [136] "phpwsl_date"         "phpwsl_dataset_id"   "phpwsl_profile_code"
## [139] "phpwsl_licence"      "totc_value"          "totc_value_avg"
## [142] "totc_method"         "totc_date"           "totc_dataset_id"
## [145] "totc_profile_code"   "totc_licence"        "nitkjd_value"
## [148] "nitkjd_value_avg"    "nitkjd_method"       "nitkjd_date"
## [151] "nitkjd_dataset_id"   "nitkjd_profile_code" "nitkjd_licence"
```

There are 788538, 153 layers of profiles with chemical properties.

Select layers with total P values:

```
total.P <- chemical %>%
   dplyr::filter(!is.na(phptot_value_avg)) %>%
   select(profile_id:layer_name, phptot_value:phptot_licence)
summary(total.P)
```

```
##    profile_id     profile_layer_id   upper_depth       lower_depth
##  Min.   : 45102   Min.   :  77586   Min.   :   0.00   Min.   :   1.00
##  1st Qu.: 49614   1st Qu.:  97538   1st Qu.:   0.00   1st Qu.:  18.00
##  Median : 61450   Median : 144506   Median :  16.00   Median :  43.00
##  Mean   :277981   Mean   :1290239   Mean   :  34.68   Mean   :  62.73
##  3rd Qu.:524156   3rd Qu.:2295953   3rd Qu.:  50.00   3rd Qu.:  90.00
##  Max.   :613997   Max.   :3133662   Max.   :1000.00   Max.   :1050.00
##   layer_name         phptot_value       phptot_value_avg  phptot_method
##  Length:12976       Length:12976       Min.   :    0.0   Length:12976
##  Class :character   Class :character   1st Qu.:   39.0   Class :character
##  Mode  :character   Mode  :character   Median :  118.0   Mode  :character
##                                        Mean   :  284.7
##                                        3rd Qu.:  250.0
##                                        Max.   :11521.0
##  phptot_date        phptot_dataset_id  phptot_profile_code phptot_licence
##  Length:12976       Length:12976       Length:12976        Length:12976
##  Class :character   Class :character   Class :character    Class
:character
##  Mode  :character   Mode  :character   Mode  :character    Mode
:character
##
##
##
```

Joining profile and attribute information

At the profile (site) level, we can select by location or country or bounding box. But to summarize or display attribute values by these, we need to *join* the profile and attribute tables.

The primary key in the profiles table is `profile_id`; the `profile_id` field in the two attribute tables is the foreign key to link the profiles with their attributes. These along with the `profile_layer_id` to specify the soil layer result in the full key to the attribute tables.

It's simplest to select the profiles in the profile table, and then use these profile IDs to select out of the attribute tables.

For example, to analyze the particle-size distribution of Indian soil profiles we first find the profiles from India:

```
(profiles.india <- dplyr::filter(profiles, country_name=="India") %>%
    select(profile_id, longitude, latitude))
```

```
## # A tibble: 199 x 3
##    profile_id longitude latitude
##         <dbl>     <dbl>    <dbl>
```

```
##  1      66473      79.5     29.0
##  2      66474      87.3     24.3
##  3      66475      83.3     25.3
##  4      66476      87.3     24.3
##  5      66477      87.2     24.3
##  6      66490      75.8     22.7
##  7      66492      78.4     17.5
##  8      66531      89.2     26.3
##  9      66732      87.2     24.3
## 10      66733      79.5     29.0
## # … with 189 more rows
```

We then use the `left_join` function to add the layers to each profile. This will repeat the primary key `profile_id` for each value of the secondary key `profile_layer_id` (i.e., the same table structure as the table being joined), with any fields selected from the main (profile) table repeated.

```
(layers.india <- left_join(profiles.india, physical) %>%
      select(profile_id, upper_depth:layer_name,sand_value_avg,
silt_value_avg, clay_value_avg))

## Joining, by = "profile_id"

## # A tibble: 1,093 x 7
##    profile_id upper_depth lower_depth layer_name sand_value_avg
silt_value_avg
##         <dbl>       <dbl>       <dbl> <chr>               <dbl>
<dbl>
##  1      66473           0          20 Ap                     37
45
##  2      66473          20          27 A12                    35
47
##  3      66473          27          43 A3                     36
46
##  4      66473          43          80 B21                    37
44
##  5      66473          80         119 <NA>                   35
47
##  6      66473         119         128 IIC                    57
32
##  7      66474           0          13 Ap                     32
52
##  8      66474          13          24 A3                     28
53
##  9      66474          41          71 B22t                   19
41
## 10      66474          71         101 B23t                   25
44
## # … with 1,083 more rows, and 1 more variable: clay_value_avg <dbl>
```

Convert this to a `data.frame`:

```
layers.india <- as.data.frame(layers.india)
```

Working with the Geopackage

For an introduction to Geopackage data structures, see "Getting Started With GeoPackage".

A Geopackage stores data in SQL tables. To access these, we first establish a connection to the SQL database with the `DBI::dbConnect` function of the `DBI` "data base interface" R package.

```
source <- paste0(wosis.dir.name, "/", "wosis_201909.gpkg")
(gpkg <- DBI::dbConnect(RSQLite::SQLite(), source))
```

```
## <SQLiteConnection>
##   Path: /Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg
##   Extensions: TRUE
```

Once the connection has been established, we can list the tables in the database with the `DBI::dbListTables` function.

```
DBI::dbListTables(gpkg)
```

```
##  [1] "gpkg_contents"
##  [2] "gpkg_extensions"
##  [3] "gpkg_geometry_columns"
##  [4] "gpkg_metadata"
##  [5] "gpkg_metadata_reference"
##  [6] "gpkg_ogr_contents"
##  [7] "gpkg_spatial_ref_sys"
##  [8] "gpkg_tile_matrix"
##  [9] "gpkg_tile_matrix_set"
## [10] "rtree_wosis_201909_profiles_geom"
## [11] "rtree_wosis_201909_profiles_geom_node"
## [12] "rtree_wosis_201909_profiles_geom_parent"
## [13] "rtree_wosis_201909_profiles_geom_rowid"
## [14] "sqlite_sequence"
## [15] "wosis_201909_attributes"
## [16] "wosis_201909_layers_chemical"
## [17] "wosis_201909_layers_physical"
## [18] "wosis_201909_profiles"
```

There are tables with the internal structure of the geopackage and others with geographic data (`wosis_201909_profiles`), attribute descriptions (`wosis_201909_attributes`), and the attributes themselves (`wosis_201909_layers_chemical`, `wosis_201909_layers_physical`). The information in the profiles and attribute tables is the same as in the text files (above).

The `gpkg_geometry_columns` table has only one record, showing the spatial reference. Show its contents with `dplyr::tbl`:

```
dplyr::tbl(gpkg, "gpkg_geometry_columns")
```

```
## # Source:   table<gpkg_geometry_columns> [?? x 6]
## # Database: sqlite 3.35.5
## #   [/Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg]
##   table_name            column_name geometry_type_name srs_id     z     m
##   <chr>                 <chr>       <chr>                <int> <int> <int>
## 1 wosis_201909_profiles geom        POINT                 4326     0     0
```

The `wosis_201909_profiles` table contains the site information, including profile ID, country of origin, dataset ID, and soil classification:

```
dplyr::tbl(gpkg, "wosis_201909_profiles")
```

```
## # Source:   table<wosis_201909_profiles> [?? x 24]
## # Database: sqlite 3.35.5
## #   [/Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg]
##    profile_id       geom dataset_id    country_id country_name
geom_accuracy
##         <int>     <blob> <chr>         <chr>      <chr>
<dbl>
##  1      36897 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
##  2      36898 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
##  3      36899 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
##  4      36900 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
##  5      36901 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
##  6      36902 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
##  7      36903 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
##  8      36904 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
##  9      36905 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
## 10      36906 <raw 29 B> {BE-UplandsI} BE         Belgium
0.000001
## # … with more rows, and 18 more variables: latitude <dbl>, longitude
<dbl>,
## #   dsds <int>, cfao_version <int>, cfao_major_group_code <chr>,
## #   cfao_major_group <chr>, cfao_soil_unit_code <chr>, cfao_soil_unit
<chr>,
## #   cwrb_version <int>, cwrb_reference_soil_group_code <chr>,
## #   cwrb_reference_soil_group <chr>, cwrb_prefix_qualifier <chr>,
## #   cwrb_suffix_qualifier <chr>, cstx_version <int>, cstx_order_name
<chr>,
## #   cstx_suborder <chr>, cstx_great_group <chr>, cstx_subgroup <chr>
```

The accuracy of the geographical coördinates (field `geom_accuracy`) is given in decimal degrees, according to the precision reported in the original source, which may have been in degrees-minutes-seconds or decimal degrees. This does not take into account any datum shifts.

There are several internal R formats for spatial data; we show how to use two of them: Simple Features and `sp` classes.

Geometry with Simple Features

Simple Features is a relatively new standard for representing spatial data.[3] The `sf` package (E. Pebesma 2018) provides R access to this representation.

To read the Geopackage into an R spatial object as Simple Features we use the `sf::st_read` function. We must specify the optional `fid_column_name` argument to include the profile ID (primary key) as a column in the attribute table.

The only GIS layer in the Geodatabase (i.e., with coördinates) is the profiles table.

```
st_layers(dsn=source)
```

```
## Driver: GPKG
## Available layers:
##                      layer_name geometry_type features fields
## 1        wosis_201909_profiles         Point   196498     22
## 2 wosis_201909_layers_chemical            NA   788538    152
## 3 wosis_201909_layers_physical            NA   702698    194
## 4      wosis_201909_attributes            NA       52      8
```

```
wosis.sf <- st_read(source, stringsAsFactors=FALSE,
                    fid_column_name="profile_id")
```

```
## Multiple layers are present in data source
/Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg, reading
layer `wosis_201909_profiles'.
## Use `st_layers' to list all layer names and their type in a data source.
## Set the `layer' argument in `st_read' to read a particular layer.
```

```
## Warning in evalq((function (..., call. = TRUE, immediate. = FALSE,
noBreaks. =
## FALSE, : automatically selected the first layer in a data source
containing more
## than one.
```

```
## Reading layer `wosis_201909_profiles' from data source
##   `/Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg'
##   using driver `GPKG'
## Simple feature collection with 196498 features and 23 fields
```

[3] https://github.com/r-spatial/sf/

```
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -172.3633 ymin: -77.84866 xmax: 179.25 ymax: 81.3956
## Geodetic CRS:  WGS 84

class(wosis.sf)

## [1] "sf"         "data.frame"

dim(wosis.sf)

## [1] 196498     24

names(wosis.sf)

##  [1] "dataset_id"                    "country_id"
##  [3] "country_name"                  "geom_accuracy"
##  [5] "latitude"                      "longitude"
##  [7] "dsds"                          "cfao_version"
##  [9] "cfao_major_group_code"         "cfao_major_group"
## [11] "cfao_soil_unit_code"           "cfao_soil_unit"
## [13] "cwrb_version"                  "cwrb_reference_soil_group_code"
## [15] "cwrb_reference_soil_group"     "cwrb_prefix_qualifier"
## [17] "cwrb_suffix_qualifier"         "cstx_version"
## [19] "cstx_order_name"               "cstx_suborder"
## [21] "cstx_great_group"              "cstx_subgroup"
## [23] "profile_id"                    "geom"
```

There are almost 200k observations.

The second-to-last column `profile_id` is the primary key, as specified with `st_read`. The final column `geom` contains the geometry of each item, here the point coördinates.

```
class(wosis.sf$geom)

## [1] "sfc_POINT" "sfc"

str(wosis.sf$geom)

## sfc_POINT of length 196498; first list element:  'XY' num [1:2] 4.67 50.65

st_bbox(wosis.sf$geom)

##       xmin       ymin       xmax       ymax
## -172.36333  -77.84866  179.25000   81.39560

st_crs(wosis.sf$geom)

## Coordinate Reference System:
##   User input: WGS 84
##   wkt:
## GEOGCRS["WGS 84",
##     DATUM["World Geodetic System 1984",
```

```
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["geodetic latitude (Lat)",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["geodetic longitude (Lon)",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     USAGE[
##         SCOPE["Horizontal component of 3D system."],
##         AREA["World."],
##         BBOX[-90,-180,90,180]],
##     ID["EPSG",4326]]
```

```
head(wosis.sf$geom, 4)
```

```
## Geometry set for 4 features
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 4.462114 ymin: 50.58396 xmax: 4.687607 ymax: 50.64989
## Geodetic CRS:  WGS 84

## POINT (4.666901 50.64989)

## POINT (4.462114 50.58396)

## POINT (4.687607 50.59788)

## POINT (4.681783 50.6336)
```

We see the geometry type, dimensions, bounding box, and coördinate reference system (CRS).

Each row is a single record, for example here a profile from Angola:

```
wosis.sf[1024,]
```

```
## Simple feature collection with 1 feature and 23 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 15.7976 ymin: -14.55055 xmax: 15.7976 ymax: -14.55055
## Geodetic CRS:  WGS 84
##          dataset_id country_id country_name geom_accuracy  latitude
longitude
## 1024 {ACTD,AF-AfSP}         AO       Angola       2.8e-05 -14.55055
15.7976
##      dsds cfao_version cfao_major_group_code cfao_major_group
## 1024  165           NA                  <NA>             <NA>
##      cfao_soil_unit_code cfao_soil_unit cwrb_version
```

```
## 1024                <NA>           <NA>           NA
##      cwrb_reference_soil_group_code cwrb_reference_soil_group
## 1024                           <NA>                      <NA>
##      cwrb_prefix_qualifier cwrb_suffix_qualifier cstx_version
cstx_order_name
## 1024                  <NA>                  <NA>           NA
<NA>
##      cstx_suborder cstx_great_group cstx_subgroup profile_id
## 1024          <NA>             <NA>          <NA>      45820
##                           geom
## 1024 POINT (15.7976 -14.55055)
```

To display a particular profile, find its `profile_id`:

```
wosis.sf[which(wosis.sf$profile_id==45820),]
```

```
## Simple feature collection with 1 feature and 23 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 15.7976 ymin: -14.55055 xmax: 15.7976 ymax: -14.55055
## Geodetic CRS:  WGS 84
##          dataset_id country_id country_name geom_accuracy  latitude
longitude
## 1024 {ACTD,AF-AfSP}         AO       Angola       2.8e-05 -14.55055
15.7976
##      dsds cfao_version cfao_major_group_code cfao_major_group
## 1024  165           NA                  <NA>             <NA>
##      cfao_soil_unit_code cfao_soil_unit cwrb_version
## 1024                <NA>           <NA>           NA
##      cwrb_reference_soil_group_code cwrb_reference_soil_group
## 1024                           <NA>                      <NA>
##      cwrb_prefix_qualifier cwrb_suffix_qualifier cstx_version
cstx_order_name
## 1024                  <NA>                  <NA>           NA
<NA>
##      cstx_suborder cstx_great_group cstx_subgroup profile_id
## 1024          <NA>             <NA>          <NA>      45820
##                           geom
## 1024 POINT (15.7976 -14.55055)
```

Each column is an attribute; these can be summarized.

For example, the dataset source:

```
length(unique(wosis.sf$country_name))
```

```
## [1] 175
```

```
head(table(wosis.sf$country_name))
```

```
##
## Afghanistan     Albania     Algeria      Angola  Antarctica   Argentina
##          19          97          10        1169           9         244
```

```
length(unique(wosis.sf$dataset_id))
```

```
## [1] 167
```

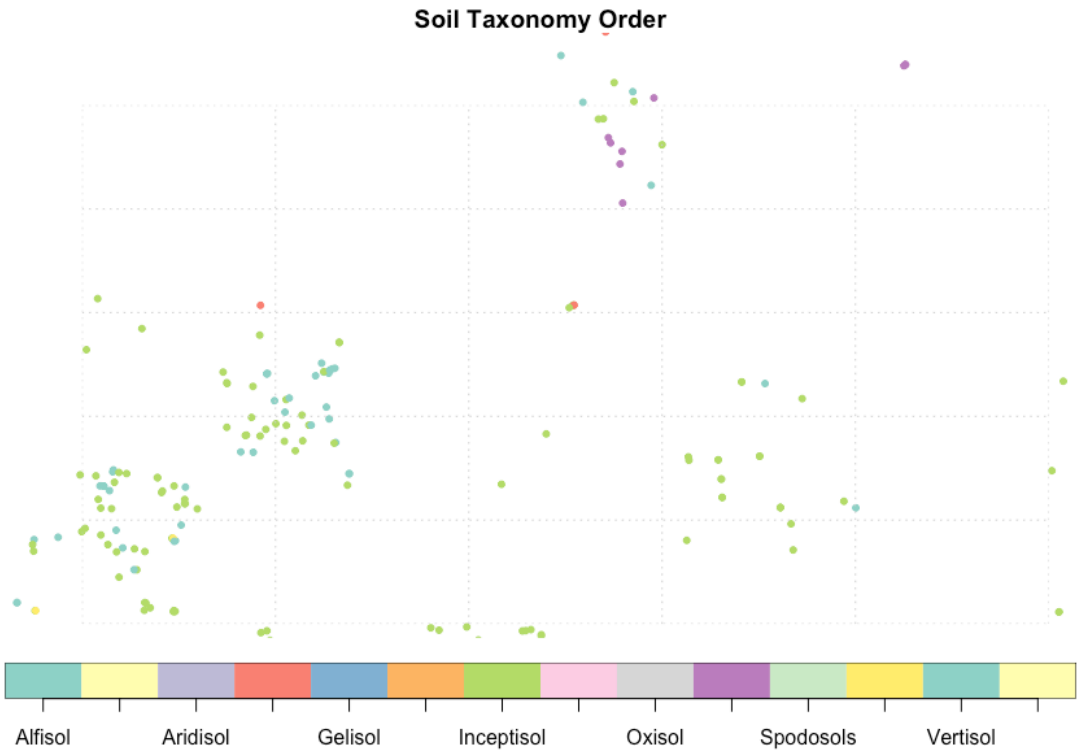
```
head(table(wosis.sf$dataset_id))
```

```
##
##                     {ACTD,AF-AfSP}                 {AF-AfSIS-phase1}
##                                794                              1902
## {AF-AfSP,AGIS,SAF-SOTER,ZA-SOTER}           {AF-AfSP,AGIS,ZA-SOTER}
##                                325                               284
##         {AF-AfSP,BJSOTER,WD-WISE}                  {AF-AfSP,BORENA}
##                                710                               210
```

Profiles come from 175 ISO countries (variously defined) and 167 contributing datasets. The list of sources (i.e., databases contributing to WoSIS) is internal to ISRIC, please ask.

The profile-level attributes can also be plotted as maps, for example Soil Taxonomy Order in a 4x2 degree tile in central NY State (USA):

```
plot(wosis.sf["cstx_order_name"],
     xlim=c(-78, -74), ylim=c(42, 44),
     pch=20,
     key.length=1, # make the legend wide enough to show all classes
     main="Soil Taxonomy Order")
grid()
```



Profiles can be subsetted by profile-level attribute, e.g., to work with just the Indian data:

```
(wosis.sf.india <- wosis.sf %>% dplyr::filter(country_name=="India"))

## Simple feature collection with 199 features and 23 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 69.8 ymin: 8.483333 xmax: 94.05 ymax: 32
## Geodetic CRS:  WGS 84
## First 10 features:
##           dataset_id country_id country_name geom_accuracy latitude
longitude
## 1          {WD-WISE}         IN        India          0.01 29.02222
79.48889
## 2          {WD-WISE}         IN        India          0.01 24.29583
87.25639
## 3          {WD-WISE}         IN        India          0.01 25.25833
83.26111
## 4          {WD-WISE}         IN        India          0.01 24.29583
87.25639
## 5          {WD-WISE}         IN        India          0.01 24.29167
87.23972
## 6  {US-NCSS,WD-WISE}         IN        India          0.01 22.72639
75.81389
## 7  {US-NCSS,WD-WISE}         IN        India          0.01 17.54167
78.40000
## 8          {WD-WISE}         IN        India          0.10 26.33333
89.16667
```

```
## 9          {WD-WISE}         IN        India          0.01 24.29167
87.23972
## 10         {WD-WISE}         IN        India          0.01 29.02222
79.48889
##    dsds cfao_version cfao_major_group_code cfao_major_group
cfao_soil_unit_code
## 1   128         1974                    J        Fluvisols
e
## 2   127         1974                    L         Luvisols
g
## 3   132         1974                    L         Luvisols
g
## 4   127         1974                    L         Luvisols
o
## 5   186         1974                    L         Luvisols
c
## 6   167           NA                 <NA>             <NA>
<NA>
## 7   193           NA                 <NA>             <NA>
<NA>
## 8   137         1974                    B        Cambisols
h
## 9   186         1974                    N         Nitosols
e
## 10  128         1974                    H        Phaeozems
h
##    cfao_soil_unit cwrb_version cwrb_reference_soil_group_code
## 1          Eutric         2006                             FL
## 2          Gleyic         2006                             LV
## 3          Gleyic         2006                             LV
## 4          Orthic         2006                             LV
## 5         Chromic         2006                             LV
## 6            <NA>         2006                             LV
## 7            <NA>         2006                             LV
## 8           Humic         2006                             UM
## 9          Eutric         2006                             NT
## 10         Haplic         2006                             PH
##    cwrb_reference_soil_group cwrb_prefix_qualifier cwrb_suffix_qualifier
## 1                  Fluvisols                  <NA>                  <NA>
## 2                   Luvisols                  <NA>                  <NA>
## 3                   Luvisols                  <NA>                  <NA>
## 4                   Luvisols                  <NA>                  <NA>
## 5                   Luvisols                  <NA>                  <NA>
## 6                   Luvisols                  <NA>                  <NA>
## 7                   Luvisols                  <NA>                  <NA>
## 8                  Umbrisols                  <NA>                  <NA>
## 9                   Nitisols                  <NA>                  <NA>
## 10                 Phaeozems                  <NA>                  <NA>
##    cstx_version cstx_order_name cstx_suborder cstx_great_group
cstx_subgroup
```

```
## 1          1975        Mollisol         Udoll        Hapludoll
Typic
## 2          1987         Alfisol        Aqualf       Ochraqualf
Typic
## 3          1975         Alfisol        Aqualf       Ochraqualf
Aeric
## 4          1975         Alfisol        Aqualf       Ochraqualf
Typic
## 5          1975         Alfisol        Ustalf       Paleustalf
Ultic
## 6            NA         Alfisol        Ustalf       Rhodustalf
Udic
## 7            NA         Alfisol        Ustalf       Haplustalf
Typic
## 8            NA            <NA>          <NA>             <NA>
<NA>
## 9          1987         Alfisol        Ustalf       Paleustalf
Ultic
## 10         1987        Mollisol         Udoll        Hapludoll
Typic
##    profile_id                       geom
## 1       66473 POINT (79.48889 29.02222)
## 2       66474 POINT (87.25639 24.29583)
## 3       66475 POINT (83.26111 25.25833)
## 4       66476 POINT (87.25639 24.29583)
## 5       66477 POINT (87.23972 24.29167)
## 6       66490 POINT (75.81389 22.72639)
## 7       66492     POINT (78.4 17.54167)
## 8       66531 POINT (89.16667 26.33333)
## 9       66732 POINT (87.23972 24.29167)
## 10      66733 POINT (79.48889 29.02222)

table(wosis.sf.india$dataset_id)

##
## {US-NCSS,WD-WISE}         {US-NCSS}         {WD-ISCN} {WD-ISIS,WD-WISE}
##                10                 10                28                12
##    {WD-Mangroves}    {WD-NWAFU-SCS}         {WD-WISE}
##                29                  2               108

wosis.sf.india %>% count(cstx_order_name)

## Simple feature collection with 9 features and 2 fields
## Geometry type: MULTIPOINT
## Dimension:     XY
## Bounding box:  xmin: 69.8 ymin: 8.483333 xmax: 94.05 ymax: 32
## Geodetic CRS:  WGS 84
##   cstx_order_name  n                             geom
## 1         Alfisol 34 MULTIPOINT ((93.53333 25.01...
## 2        Aridisol  9 MULTIPOINT ((72.83333 26.08...
## 3         Entisol 13 MULTIPOINT ((93.05 23.83333...
```

```
## 4      Inceptisol 27 MULTIPOINT ((94.05 27.33333...
## 5        Mollisol  4 MULTIPOINT ((79.48889 29.02...
## 6          Oxisol  3 MULTIPOINT ((77.23333 9.083...
## 7         Ultisol  3 MULTIPOINT ((87.26667 23.18...
## 8        Vertisol 35 MULTIPOINT ((75.8 22.71667)...
## 9            <NA> 71 MULTIPOINT ((77.5 32), (78 ...
```

Geometry with sp

An older R spatial representation than Simple Features is sp "Classes and methods for spatial data in R," explained in detail in (E. J. Pebesma and Bivand 2005) and (Bivand, Pebesma, and Gómez-Rubio 2013).

We can read the Geopackage into an R sp object with the readOGR function of the rgdal package. By default readOGR reads the first layer from a Geopackage; here that is the profiles (the only layer with geometry).

In this dataset strings are to be interpreted as R factors, i.e., categorical variables.

```
ogrInfo(dsn=source)
```

```
## Warning in ogrInfo(dsn = source): First layer wosis_201909_profiles read;
multiple layers present in
## /Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg, check
layers with ogrListLayers()
```

```
## Source:
"/Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg", layer:
"wosis_201909_profiles"
## Driver: GPKG; number of rows: 196498
## Feature type: wkbPoint with 2 dimensions
## Extent: (-172.363 -77.8487) - (179.25 81.3956)
## CRS: +proj=longlat +datum=WGS84 +no_defs
## Number of fields: 22
##                              name type length typeName
## 1                      dataset_id    4      0   String
## 2                      country_id    4      2   String
## 3                    country_name    4      0   String
## 4                   geom_accuracy    2      0     Real
## 5                        latitude    2      0     Real
## 6                       longitude    2      0     Real
## 7                            dsds    0      0  Integer
## 8                    cfao_version    0      0  Integer
## 9           cfao_major_group_code    4      2   String
## 10               cfao_major_group    4      0   String
## 11            cfao_soil_unit_code    4      1   String
## 12                 cfao_soil_unit    4      0   String
## 13                   cwrb_version    0      0  Integer
## 14 cwrb_reference_soil_group_code    4      4   String
## 15      cwrb_reference_soil_group    4      0   String
## 16          cwrb_prefix_qualifier    4      0   String
```

```
## 17          cwrb_suffix_qualifier    4      0   String
## 18                   cstx_version    0      0  Integer
## 19                cstx_order_name    4      0   String
## 20                  cstx_suborder    4      0   String
## 21               cstx_great_group    4      0   String
## 22                  cstx_subgroup    4      0   String
```

```
wosis.sp <- readOGR(dsn=source,
                stringsAsFactors = TRUE)
```

```
## Warning in readOGR(dsn = source, stringsAsFactors = TRUE): First layer
wosis_201909_profiles read; multiple layers present in
## /Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg, check
layers with ogrListLayers()
```

```
## OGR data source with driver: GPKG
## Source:
"/Users/rossiter/data/ISRIC/ISRIC_WoSIS/wosis2019/wosis_201909.gpkg", layer:
"wosis_201909_profiles"
## with 196498 features
## It has 22 fields
```

```
class(wosis.sp)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
bbox(wosis.sp)
```

```
##                  min      max
## coords.x1 -172.36333 179.2500
## coords.x2  -77.84866  81.3956
```

```
proj4string(wosis.sp)
```

```
## Warning in proj4string(wosis.sp): CRS object has comment, which is lost in
## output
```

```
## [1] "+proj=longlat +datum=WGS84 +no_defs"
```

```
dim(wosis.sp)
```

```
## [1] 196498     22
```

```
summary(wosis.sp)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##                  min      max
## coords.x1 -172.36333 179.2500
## coords.x2  -77.84866  81.3956
## Is projected: FALSE
```

```
## proj4string : [+proj=longlat +datum=WGS84 +no_defs]
## Number of points: 196498
## Data attributes:
##         dataset_id      country_id                      country_name
##  {US-NCSS}   :50353   US     :56277   United States of America:56277
##  {AU-CSIRO}  :42523   AU     :42758   Australia               :42758
##  {CH-NABODAT}:10869   CH     :10943   Switzerland             :10943
##  {WD-ISCN}   : 7973   BR     : 8883   Brazil                  : 8883
##  {MX-INEGI}  : 7461   CA     : 8516   Canada                  : 8516
##  {BE-VASPDB} : 6820   MX     : 7554   Mexico                  : 7554
##  (Other)     :70499   (Other):61567   (Other)                 :61567
##  geom_accuracy          latitude        longitude             dsds
##  Min.   :0.0000001   Min.   :-77.85   Min.   :-172.363   Min.   :   0
##  1st Qu.:0.0000010   1st Qu.:-16.94   1st Qu.: -91.302   1st Qu.:  56
##  Median :0.0000100   Median : 32.90   Median :   4.534   Median : 110
##  Mean   :0.0103703   Mean   : 17.42   Mean   :  -1.436   Mean   : 117
##  3rd Qu.:0.0000100   3rd Qu.: 45.30   3rd Qu.:  51.997   3rd Qu.: 152
##  Max.   :1.0000000   Max.   : 81.40   Max.   : 179.250   Max.   :3292
##                                                          NA's   :117
##   cfao_version    cfao_major_group_code  cfao_major_group
cfao_soil_unit_code
##  Min.   :1974     CM     :  1748        Luvisols :  2999   h      :  3326
##  1st Qu.:1974     LV     :  1689        Cambisols:  2843   e      :  2902
##  Median :1997     L      :  1310        Vertisols:  1897   c      :  2662
##  Mean   :1986     R      :  1276        Regosols :  1740   o      :  1586
##  3rd Qu.:1997     B      :  1095        Arenosols:  1450   k      :  1193
##  Max.   :1997     (Other): 16772        (Other)  : 12961   (Other):  8182
##  NA's   :172608   NA's   :172608        NA's     :172608   NA's   :176647
##  cfao_soil_unit    cwrb_version    cwrb_reference_soil_group_code
##  Haplic :  2991   Min.   :1998     LV     :  3276
##  Eutric :  2902   1st Qu.:2006     CM     :  3035
##  Chromic:  1616   Median :2007     VR     :  2207
##  Calcic :  1166   Mean   :2007     RG     :  1919
##  Orthic :  1128   3rd Qu.:2007     AR     :  1608
##  (Other): 10048   Max.   :2015     (Other): 14618
##  NA's   :176647   NA's   :169834   NA's   :169835
##  cwrb_reference_soil_group cwrb_prefix_qualifier cwrb_suffix_qualifier
##  Luvisols :  3276          Endoleptic:   537     Humic     :   716
##  Cambisols:  3035          Epileptic :   529     Esqueletic:   368
##  Vertisols:  2207          Esqueletic:   317     Calcaric  :   361
##  Regosols :  1919          Haplic    :   311     Luvic     :   272
##  Arenosols:  1608          Eutric    :   271     Aridic    :   252
##  (Other)  : 14619          (Other)   :  5616     (Other)   :  2727
##  NA's     :169834          NA's      :188917     NA's      :191802
##   cstx_version      cstx_order_name   cstx_suborder     cstx_great_group
##  Min.   : 199     Alfisol   :  8303   Udalf  :  5081   Hapludalf:  3503
##  1st Qu.:2003     Mollisol  :  6547   Udult  :  2892   Hapludoll:  1308
##  Median :2009     Inceptisol:  3958   Udoll  :  2492   Hapludult:  1094
##  Mean   :2004     Ultisol   :  3712   Aqualf :  1802   Argiudoll:  1050
##  3rd Qu.:2012     Entisol   :  2914   Aquoll :  1654   Paleudult:   920
```

```
##  Max.   :2015     (Other)   :  4299   (Other): 17834   (Other)  : 21469
##  NA's   :175184   NA's      :166765   NA's   :164743   NA's     :167154
##   cstx_subgroup
##  Typic   : 11238
##  Aquic   :  2136
##  Oxyaquic:  1310
##  Aeric   :  1187
##  Mollic  :   934
##  (Other) : 11709
##  NA's    :167984

names(wosis.sp@data)

##  [1] "dataset_id"                    "country_id"
##  [3] "country_name"                  "geom_accuracy"
##  [5] "latitude"                      "longitude"
##  [7] "dsds"                          "cfao_version"
##  [9] "cfao_major_group_code"         "cfao_major_group"
## [11] "cfao_soil_unit_code"           "cfao_soil_unit"
## [13] "cwrb_version"                  "cwrb_reference_soil_group_code"
## [15] "cwrb_reference_soil_group"     "cwrb_prefix_qualifier"
## [17] "cwrb_suffix_qualifier"         "cstx_version"
## [19] "cstx_order_name"               "cstx_suborder"
## [21] "cstx_great_group"              "cstx_subgroup"
```

The shapefile has been imported as a `SpatialPointsDataFrame` with the correct CRS. In the `sp` data structure the coördinates are not stored as an attribute (as in Simple Features), instead, they are in their own slot.

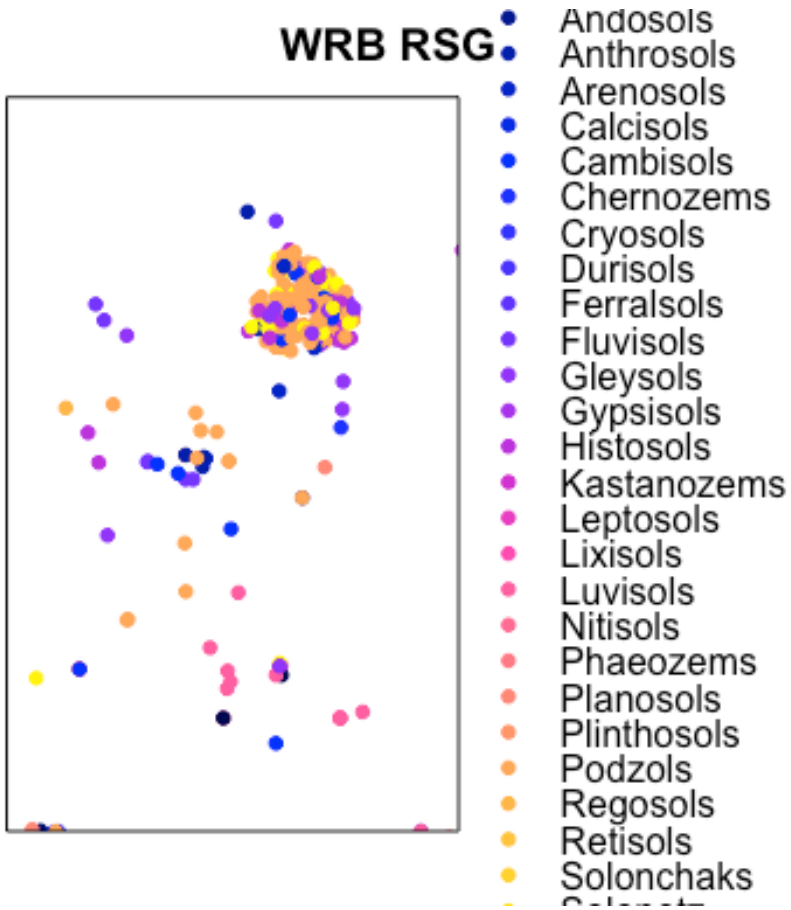
The profile data can be summarized:

```
unique(wosis.sp$cwrb_reference_soil_group)

##  [1] <NA>         Solonchaks   Calcisols    Podzols      Lixisols
##  [6] Luvisols     Arenosols    Acrisols     Umbrisols    Ferralsols
## [11] Cambisols    Gypsisols    Vertisols    Nitisols     Phaeozems
## [16] Solonetz     Alisols      Plinthosols  Gleysols     Regosols
## [21] Stagnosols   Planosols    Fluvisols    Leptosols    Histosols
## [26] Chernozems   Andosols     Kastanozems  Anthrosols   Cryosols
## [31] Retisols     Albeluvisols Durisols
## 32 Levels: Acrisols Albeluvisols Alisols Andosols Anthrosols ... Vertisols

summary(is.na(wosis.sp$cwrb_reference_soil_group))

##    Mode   FALSE    TRUE
## logical   26664  169834

table(wosis.sp$cwrb_reference_soil_group)

##
##     Acrisols Albeluvisols      Alisols     Andosols   Anthrosols
Arenosols
```

```
##         1227          109          459          408          259
1608
##    Calcisols    Cambisols   Chernozems     Cryosols     Durisols
Ferralsols
##         1434         3035          728          120           43
875
##    Fluvisols     Gleysols    Gypsisols    Histosols  Kastanozems
Leptosols
##         1076         1085          118          244          308
1425
##     Lixisols     Luvisols     Nitisols    Phaeozems    Planosols
Plinthosols
##          789         3276          333         1441          313
142
##      Podzols     Regosols     Retisols   Solonchaks     Solonetz
Stagnosols
##          375         1919            6          333          374
73
##    Umbrisols    Vertisols
##          522         2207
```

Here is a map of the profiles with a WRB classification in a 4°×4° tile including the Netherlands:

```
spplot(wosis.sp, zcol="cwrb_reference_soil_group",
     xlim=c(4, 8), ylim=c(50, 54),
     pch=20, key.space="right",
     main="WRB RSG")
```



Attributes

The Geopackage also contains SQL tables with the physical and chemical attributes. These are accessed as SQL connections:

```
class(dplyr::tbl(gpkg, "wosis_201909_layers_chemical"))

## [1] "tbl_SQLiteConnection" "tbl_dbi"              "tbl_sql"
## [4] "tbl_lazy"             "tbl"

class(dplyr::tbl(gpkg, "wosis_201909_layers_physical"))

## [1] "tbl_SQLiteConnection" "tbl_dbi"              "tbl_sql"
## [4] "tbl_lazy"             "tbl"
```

Read these into R and display the variable names:

```
wosis.chemical <- dplyr::tbl(gpkg, "wosis_201909_layers_chemical")
(wosis.chemical$ops$vars)

##   [1] "profile_layer_id"    "profile_id"          "upper_depth"
##   [4] "lower_depth"         "layer_name"          "litter"
##   [7] "tceq_value"          "tceq_value_avg"      "tceq_method"
##  [10] "tceq_date"           "tceq_dataset_id"     "tceq_profile_code"
##  [13] "tceq_licence"        "cecph7_value"        "cecph7_value_avg"
```

```
##  [16] "cecph7_method"       "cecph7_date"         "cecph7_dataset_id"
##  [19] "cecph7_profile_code" "cecph7_licence"      "cecph8_value"
##  [22] "cecph8_value_avg"    "cecph8_method"       "cecph8_date"
##  [25] "cecph8_dataset_id"   "cecph8_profile_code" "cecph8_licence"
##  [28] "ecec_value"          "ecec_value_avg"      "ecec_method"
##  [31] "ecec_date"           "ecec_dataset_id"     "ecec_profile_code"
##  [34] "ecec_licence"        "elco20_value"        "elco20_value_avg"
##  [37] "elco20_method"       "elco20_date"         "elco20_dataset_id"
##  [40] "elco20_profile_code" "elco20_licence"      "elco25_value"
##  [43] "elco25_value_avg"    "elco25_method"       "elco25_date"
##  [46] "elco25_dataset_id"   "elco25_profile_code" "elco25_licence"
##  [49] "elco50_value"        "elco50_value_avg"    "elco50_method"
##  [52] "elco50_date"         "elco50_dataset_id"   "elco50_profile_code"
##  [55] "elco50_licence"      "elcosp_value"        "elcosp_value_avg"
##  [58] "elcosp_method"       "elcosp_date"         "elcosp_dataset_id"
##  [61] "elcosp_profile_code" "elcosp_licence"      "orgc_value"
##  [64] "orgc_value_avg"      "orgc_method"         "orgc_date"
##  [67] "orgc_dataset_id"     "orgc_profile_code"   "orgc_licence"
##  [70] "phca_value"          "phca_value_avg"      "phca_method"
##  [73] "phca_date"           "phca_dataset_id"     "phca_profile_code"
##  [76] "phca_licence"        "phaq_value"          "phaq_value_avg"
##  [79] "phaq_method"         "phaq_date"           "phaq_dataset_id"
##  [82] "phaq_profile_code"   "phaq_licence"        "phkc_value"
##  [85] "phkc_value_avg"      "phkc_method"         "phkc_date"
##  [88] "phkc_dataset_id"     "phkc_profile_code"   "phkc_licence"
##  [91] "phnf_value"          "phnf_value_avg"      "phnf_method"
##  [94] "phnf_date"           "phnf_dataset_id"     "phnf_profile_code"
##  [97] "phnf_licence"        "phpbyi_value"        "phpbyi_value_avg"
## [100] "phpbyi_method"       "phpbyi_date"         "phpbyi_dataset_id"
## [103] "phpbyi_profile_code" "phpbyi_licence"      "phpmh3_value"
## [106] "phpmh3_value_avg"    "phpmh3_method"       "phpmh3_date"
## [109] "phpmh3_dataset_id"   "phpmh3_profile_code" "phpmh3_licence"
## [112] "phpols_value"        "phpols_value_avg"    "phpols_method"
## [115] "phpols_date"         "phpols_dataset_id"   "phpols_profile_code"
## [118] "phpols_licence"      "phprtn_value"        "phprtn_value_avg"
## [121] "phprtn_method"       "phprtn_date"         "phprtn_dataset_id"
## [124] "phprtn_profile_code" "phprtn_licence"      "phptot_value"
## [127] "phptot_value_avg"    "phptot_method"       "phptot_date"
## [130] "phptot_dataset_id"   "phptot_profile_code" "phptot_licence"
## [133] "phpwsl_value"        "phpwsl_value_avg"    "phpwsl_method"
## [136] "phpwsl_date"         "phpwsl_dataset_id"   "phpwsl_profile_code"
## [139] "phpwsl_licence"      "totc_value"          "totc_value_avg"
## [142] "totc_method"         "totc_date"           "totc_dataset_id"
## [145] "totc_profile_code"   "totc_licence"        "nitkjd_value"
## [148] "nitkjd_value_avg"    "nitkjd_method"       "nitkjd_date"
## [151] "nitkjd_dataset_id"   "nitkjd_profile_code" "nitkjd_licence"
```

Further these can be processed as for the tables, see above.

Working with WoSIS as a `SoilProfileCollection`

The `aqp` "Algorithms for Quantitive Pedology" package provides many functions for working with soil profile data. Its principal data structure is the `SoilProfileCollection`, which stores profiles and their per-horizon attributes.

Load this package:

```
library(aqp)            # Algorithms for Quantitative Pedology

## This is aqp 1.29

##
## Attaching package: 'aqp'

## The following objects are masked from 'package:dplyr':
##
##     combine, filter, group_by, mutate, slice, summarize

## The following object is masked from 'package:stats':
##
##     filter
```

In this example we convert the small dataset for India to a `SoilProfileCollection`.

The `aqp::depth` function initializes the `SoilProfileCollection` object. The formula has the field name of the profile on the left, and the the field names of the horizon boundaries on the right. These fields are in the WoSIS layer.

Note that the object to be converted to a `SoilProfileCollection` must be a `data.frame` only, not also a `dpylr` object.

```
ds.aqp <- as.data.frame(layers.india)
depths(ds.aqp) <- profile_id ~ upper_depth + lower_depth

## Warning: Horizon top depths contain NA! Check depth logic with
## aqp::checkHzDepthLogic()

## Warning: Horizon bottom depths contain NA! Check depth logic with
## aqp::checkHzDepthLogic()

is(ds.aqp)

## [1] "SoilProfileCollection"

slotNames(ds.aqp)

## [1] "idcol"        "hzidcol"      "depthcols"    "metadata"     "horizons"
## [6] "site"         "sp"           "diagnostic"   "restrictions"

str(ds.aqp@site)
```

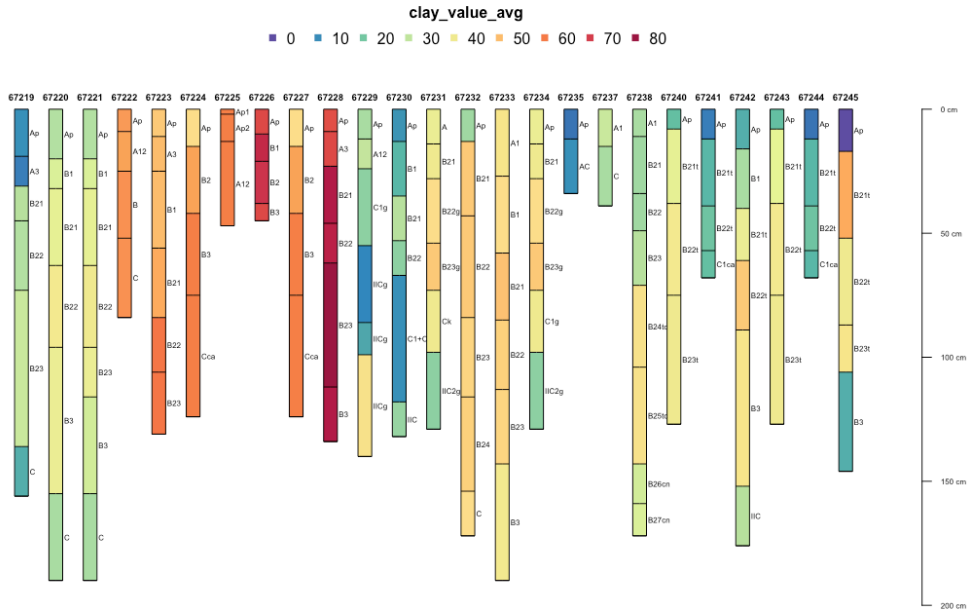
```
## 'data.frame':    199 obs. of  1 variable:
##  $ profile_id: chr  "170991" "170992" "170993" "170994" ...

str(ds.aqp@horizons)

## 'data.frame':    1093 obs. of  8 variables:
##  $ profile_id    : num  170991 170991 170991 170991 170991 ...
##  $ upper_depth   : num  0 0 10 10 26 38 58 64 83 0 ...
##  $ lower_depth   : num  10 10 26 26 38 58 64 83 130 20 ...
##  $ layer_name    : chr  "Ap1" NA "Ap2" NA ...
##  $ sand_value_avg: num  38 50.5 37.3 50.6 44.5 73.8 22.1 80.2 84.6 40.8
...
##  $ silt_value_avg: num  28.5 26.1 29.5 26.8 26.6 17.7 48.3 14.6 11.8 21.8
...
##  $ clay_value_avg: num  33.5 23.4 33.2 22.6 28.9 8.5 29.6 5.2 3.6 37.4 ...
##  $ hzID          : chr  "1" "2" "3" "4" ...
##  - attr(*, "problems")= tibble [21,314 × 5] (S3: tbl_df/tbl/data.frame)
##   ..$ row     : int [1:21314] 1063 2650 2674 3725 3764 3765 3986 4488 4492
6748 ...
##   ..$ col     : chr [1:21314] "cstx_version" "cstx_version" "cstx_version"
"cstx_version" ...
##   ..$ expected: chr [1:21314] "1/0/T/F/TRUE/FALSE" "1/0/T/F/TRUE/FALSE"
"1/0/T/F/TRUE/FALSE" "1/0/T/F/TRUE/FALSE" ...
##   ..$ actual  : chr [1:21314] "1990" "1975" "1975" "1975" ...
##   ..$ file    : chr [1:21314] "'./wosis2019/wosis_201909_profiles.tsv'"
"'./wosis2019/wosis_201909_profiles.tsv'"
"'./wosis2019/wosis_201909_profiles.tsv'"
"'./wosis2019/wosis_201909_profiles.tsv'" ...
```

Note how the horizons have been grouped into sites, in the `@site` slot, and the per-horizon (by depth) values are in the `@horizons` slot. Here we have 1093 horizons in 199 profiles.

Now this `SoilProfileCollection` can be used for many `aqp` functions. For example, here is the depth distribution of average bulk density of the components for the first 24 listed profiles, labelled by genetic horizon.

```
ds.aqp[100,]

## SoilProfileCollection with 1 profiles and 6 horizons
## profile ID: profile_id  |  horizon ID: hzID
## Depth range: 156 - 156 cm
##
## ----- Horizons (6 / 6 rows  |  8 / 8 columns) -----
##  profile_id hzID upper_depth lower_depth layer_name sand_value_avg
##       67219  522           0          19         Ap             34
##       67219  523          19          31         A3             35
##       67219  524          31          45        B21             24
##       67219  525          45          73        B22             23
##       67219  526          73         136        B23             27
##       67219  527         136         156          C             42
##  silt_value_avg clay_value_avg
```

```
##               50              16
##               51              14
##               45              31
##               47              30
##               40              33
##               38              20
##
## ----- Sites (1 / 1 rows  |  1 / 1 columns) -----
##  profile_id
##       67219
##
## Spatial Data:
##      [,1]
## [1,]   NA
## CRS:  NA

plotSPC(ds.aqp[100:124,], name="layer_name", color='clay_value_avg')
```



Notice tha the profiles have different thickness.

References

Bivand, Roger, Edzer Pebesma, and Virgilio Gómez-Rubio. 2013. *Applied Spatial Data Analysis with R*. 2nd ed. Use R! 10. New York: Springer. http://link.springer.com/book/10.1007%2F978-1-4614-7618-4.

Pebesma, Edzer. 2018. “Simple Features for R: Standardized Support for Spatial Vector Data.” *The R Journal* 10 (1): 439–46. https://doi.org/10.32614/RJ-2018-009.

Pebesma, Edzer J., and Roger S. Bivand. 2005. “Classes and Methods for Spatial Data in R.” *R News* 5 (2): 9–13.