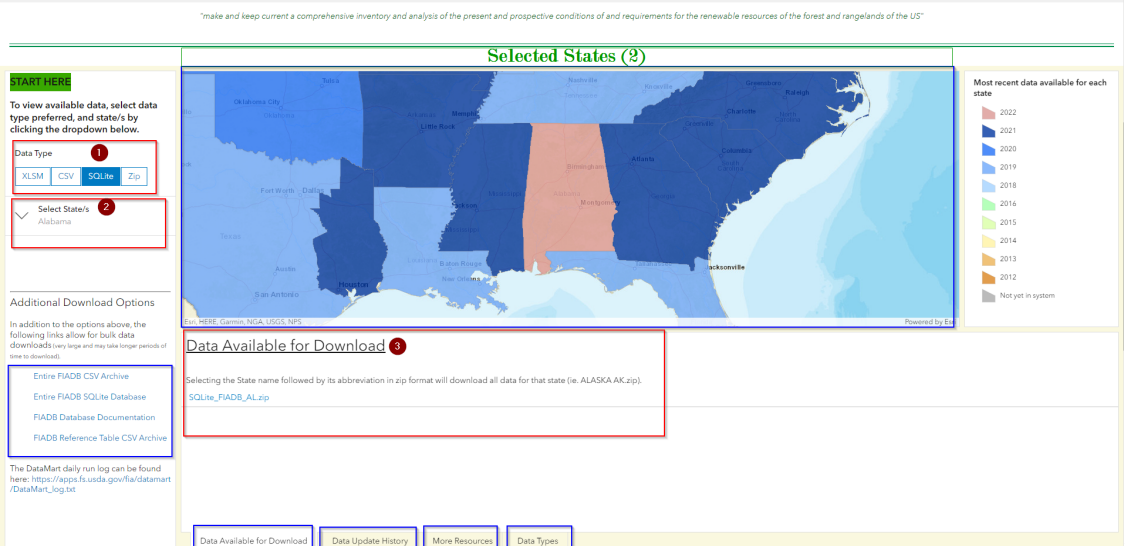
Extracting FIA tables from SQLite databases and unique plots in each state

Mukti Subedi

2023-08-31

Downloading compressed state SQlite database and extract

Forest Inventory and Analysis (FIA) data can be downloaded from FIA DataMart. On the left panel, select Data Type under the heading "START HERE," Once you select sate(s) of your interests, available data for download will be displayed on the bottom part of the central panel under "Data Available for Download." Click and save your data in your directory. Please check the screenshot below.



Required R packages

1) RsQulite
2) DBI

`RSQlite` package embeds the SQLite database engine in R and provides an interface compliant with the DBI package. `DBI` package is a R database interface definition for communication between R and relational database management systems

Extracting SQLite databases into state tables (`*.csv`)

Compressed zipfiles can be extracted using `7zip`software with a single click. Extracted databases are stored in external drive at: "F:\00_00_All_ind_states_march2023\00_02_allSttatsCSVS\". The code chunk below is the function that takes the `.db` extension files iteratively from the `00_02-allStatatsCSVS` folder.

```r
extractFiaDatabase <- function(dir = getwd(), dbpattern = "*.db$",
                               recursive = TRUE, full.namees = TRUE,
                               locToExtract = "./") {
  #--------------- list all FIA files by state interactively

  all_files <- list.files(
    path = dir, pattern = dbpattern, recursive = TRUE,
    full.names = TRUE
  )

  #---------- loop through each state database
  for (db in all_files) {
    #--------------------- derive index for each database
    ind <- which(all_files == db)

    # print in console which database is being processing
    print(paste0("processing database for: ", db, "
                 remaining dbs =  :", length(all_files) - ind))

    # ----------- read the connection for sqlite database
    conn <- dbConnect(drv = SQLite(), dbname = all_files[ind])

    # extract names of tables/ this will be used to print each
    # table into csv
    fname <- invisible(paste(dbListTables(conn), sep = ","))

    # extract state abbreviation based on file name
    state <- substr(
      x = all_files[ind], start = (nchar(all_files[ind]) - 4),
      stop = (nchar(all_files[ind]) - 3)
    )
    # database path for each state
    path_csv <- paste0(locToExtract, "FIA_CSVS_", state)
    # check if path is already exists
    if (!dir.exists(path_csv)) {
      dir.create(path_csv)
    } else {
      print(paste0("path already exist:", path_csv))
    }

    # writing tables based on database file extension.
    #-------------------------------------------------------------------------#
    for (file in fname) {
      f_ind <- which(fname == file)

      print(paste0("printing table ", file, " of ", state,
                   " remaining tables = ", length(fname) - f_ind))
      dtable <- dbReadTable(conn = conn, name = file)

      nm <- paste0(state, "_", file, ".csv")

      write.csv(dtable, na = "", sep = ",", row.names = FALSE,
                file = paste0(path_csv, "/", nm))
```

```
    }
  }
}
# location of directories
extpath <- "F:\\00_00_All_ind_states_march2023\\00_02_allSttatsCSVS\\"
# csv files to be extracted
dirpath <- "F:\\00_00_All_ind_states_march2023\\00_01_allStatesExtract"

extractFiaDatabase(dir = dirpath, dbpattern = "*.db$", recursive = TRUE,
                   full.namees = TRUE, locToExtract = extpath)
```

`extractFiaDatabase` Function arguments

The `extractFiaDatabase()` function designed to extract data from multiple FIA (Forest Inventory and Analysis) database in SQLite format, and save each table from each database as a separate CSV file. The function takes the following arguments

1. `dir`: specifies the directory where the FIA databases are located. If not specified, the default is the current working directory.
2. `dbpattern`: a regular expression pattern used to identify the FIA databases to be extracted. By default, it looks for any file with a **.db** extension.
3. `recursive`: a logical value indicating whether sub-directories of the specified dir should be searched for FIA databases as well.
4. `full.names`: a logical value indicating whether the full path and file names of the FIA databases should be returned.
5. `locToExtract`: the directory where the extracted CSV files will be saved. By default, it creates a sub-directory called **"FIA_CSVS_"** in the current working directory.

Function running steps

The function first uses the `list.files()` function to retrieve a list of all files in the specified directory (**dir**) that match the **dbpattern**. These files are assumed to be FIA databases in SQLite format.

It then loops through each file in the list and performs the following operations:

- Prints a message indicating which database is being processed and how many databases are left to be processed.
- Connects to the database using `dbConnect()` from the `RSQLite` package.
- Extracts the names of all tables in the database using `dbListTables()`.
- Extracts the state abbreviation from the file name.
- Specifies a path for the extracted CSV files based on the state abbreviation.
- Creates the directory for the CSV files if it does not already exist.
- Loops through each table in the database and performs the following operations:
  - Prints a message indicating which table is being printed and how many tables are left to be printed.
  - Reads the table from the database using `dbReadTable()`.
  - Specifies a file name for the CSV file based on the state abbreviation and the table name.
  - Writes the table as a CSV file to the specified directory using `write.csv()`.

Extracting unique plots from FIA state tables

Extracted **csv** are now stored in "F:\00_00_All_ind_states_march2023\00_01_allStatesExtract". The code chunk below is the function that takes the `.csv` extension files; especially **PLOT**, **COND**, **REF_SPECIES**, and **TREE** tables.

```
# removing columns that are not necessary
remCols <- function() {
  dropCol <- c(
    "CREATED_BY", "CREATED_IN_INSTANCE", "CREATED_DATE",
    "MODIFIED_BY", "MODIFIED_IN_INSTANCE", "MODIFIED_DATE"
  )
  return(dropCol)
}
#------------------------------------------------------------------------------ #

#---------- loop through each state database

filterUniquePlot <- function(dir.list, write.location) {
  listDir <- list.dirs(path = dir.list)[-1]

  for (db in listDir) {
    ind <- which(listDir == db)

    st_abbr <- substr(
      x = listDir[ind], start = (nchar(listDir[ind]) - 1),
      stop = (nchar(listDir[ind]) - 0)
    )

    print(paste0("currently working on state = ", st_abbr,
                 " remaining = ", length(listDir) - ind))

    # print(st_abbr)
    tab <- c("PLOT", "COND", "REF_SPECIES", "TREE")

    #------ read plot
    PLOT <- data.table::fread(
      input = paste0(db, "/", st_abbr, "_", tab[1], ".csv"), sep = ",", header = TRUE,
      integer64 = "double", logical01 = FALSE, nThread = getDTthreads(),
      drop = remCols(), showProgress = TRUE
    )
    #------ cond

    COND <- data.table::fread(
      input = paste0(db, "/", st_abbr, "_", tab[2], ".csv"), sep = ",", header = TRUE,
      integer64 = "double", logical01 = FALSE, nThread = getDTthreads(),
      drop = remCols(), showProgress = TRUE
    )
    #----- ref species
    REF_SPECIES <- data.table::fread(
      input = paste0(db, "/", st_abbr, "_", tab[3], ".csv"), sep = ",", header = TRUE,
      integer64 = "double", logical01 = FALSE, nThread = getDTthreads(),
      drop = remCols(), showProgress = TRUE
    )
```

```
#----- TREE
TREE <- data.table::fread(
  input = paste0(db, "/", st_abbr, "_", tab[4], ".csv"), sep = ",", header = TRUE,
  integer64 = "double", logical01 = FALSE, nThread = getDTthreads(),
  drop = remCols(), showProgress = TRUE
)
#-------------------------------------------------------------------------------#

# create PLT_CN field in plot table using plot number
# pltID is the unique id across the census

PLOT <- PLOT %>%
  dplyr::mutate(
    PLT_CN = CN,
    pltID = stringr::str_c(UNITCD, STATECD, COUNTYCD, PLOT, sep = "_")
  ) %>%
  collect() # to convert data.table structure into data.table format

# -------------------------------------------------------------------------------#
# Join species name in Tree table

#
refsel <- REF_SPECIES %>%
  dplyr::select(c("SPCD", "COMMON_NAME", "GENUS", "SPECIES")) %>%
  collect() %>%
  data.table()

# TREE[refsel, on = "SPCD"]

TREE <- merge(refsel, TREE, all.x = TRUE, by = "SPCD")

TREE[, SCIENTIFIC_NAME := paste(GENUS, SPECIES)] %>% collect()

#-------------------------------------------------------------------------------#
# Dropping names that are not required
#----------- names from plot
nam.plt <- c(
  "PLT_CN", "pltID", "STATECD", "UNITCD", "COUNTYCD", "PLOT",
  "MACRO_BREAKPOINT_DIA", "INVYR", "MEASYEAR", "PLOT_STATUS_CD",
  "REMPER", "LAT", "LON", "ELEV", "GROW_TYP_CD", "ECOSUBCD"
)

#---------- names from cond
nam.cond <- c(
  "PLT_CN", "CONDPROP_UNADJ", "PROP_BASIS", "COND_STATUS_CD", "CONDID",
  "RESERVCD", "OWNGRPCD", "FORINDCD", "ADFORCD", "FORTYPCD", "FLDTYPCD",
  "MAPDEN", "STDAGE", "STDSZCD", "FLDSZCD", "SITECLCD", "SICOND",
  "SIBASE", "SISP", "STDORGCD", "STDORGSP", "PROP_BASIS", "SLOPE",
  "ASPECT", "PHYSCLCD", "GSSTKCD", "CYCLE"
)
#----------- names from tree
nam.tre <- c(
  "PLT_CN", "CONDID", "DIA", "SPCD", "TPA_UNADJ",
```

```
  "SUBP", "TREE", "STATUSCD", "SPCD", "SPGRPCD",
  "HT", "HTCD", "ACTUALHT", "TREECLCD", "CCLCD", "CR",
  "TPA_UNADJ", "TPAMORT_UNADJ", "TPAREMV_UNADJ", "TPAGROW_UNADJ",
  "DRYBIO_BOLE", "DRYBIO_TOP", "DRYBIO_STUMP", "DRYBIO_SAPLING",
  "DRYBIO_WDLD_SPP", "DRYBIO_BG", "CARBON_AG", "CARBON_BG",
  "COMMON_NAME", "GENUS", "SPECIES", "SCIENTIFIC_NAME"
)

#------------------------------------------------------- Full Tree list -----#
## ----- Plot filter

COND <- COND %>%
  dplyr::select(all_of(nam.cond)) %>%
  collect()

TREE <- TREE %>%
  dplyr::select(all_of(nam.tre)) %>%
  collect()

PLOT <- PLOT %>%
  dplyr::select(all_of(nam.plt)) %>%
  collect()
## Load FIA table {#header1}
data <- PLOT %>%
  dplyr::left_join(COND, by = c("PLT_CN")) %>%
  dplyr::left_join(TREE, by = c("PLT_CN", "CONDID"))

#-----------------------------------------------------------------------------#

nrow(data)

# FIA started annual inventory in 1999 after farm bill

# lets filter plots that were measured after 1999

data.sel <- data %>%
  filter(MEASYEAR > 1999) %>%
  # filter period inventory selection
  filter(TREECLCD == 0 | TREECLCD == 2) %>%
  # at least one solid 8-foot section, TREECLCD ==3 or 4 are rotten or dead
  # dead trees are also assigned the number
  filter(STATUSCD == 1)
# filter live trees only
# filter(DIA>= 5) subplot by default should use dia threshold of 5"

nrow(data.sel)

new <- data.sel %>%
  dplyr::select(all_of(c(
    "PLT_CN", "pltID", "STATECD", "UNITCD", "COUNTYCD", "PLOT",
    "INVYR", "MEASYEAR", "LAT", "LON", "ELEV", "REMPER", "ECOSUBCD",
    "CONDID", "RESERVCD", "OWNGRPCD", "FORTYPCD", "FLDTYPCD", "MAPDEN", "STDAGE",
    "STDSZCD", "FLDSZCD", "SITECLCD", "SICOND", "SIBASE", "SISP",
```

```
        "STDORGCD", "STDORGSP", "SLOPE", "ASPECT", "PHYSCLCD", "GSSTKCD",
        "CYCLE", "DIA", "SPCD", "TPA_UNADJ", "SUBP", "TREE", "STATUSCD",
        "SPGRPCD", "HT", "HTCD", "ACTUALHT", "TREECLCD", "CCLCD", "CR", "COMMON_NAME",
        "GENUS", "SPECIES", "SCIENTIFIC_NAME"
      )))

    #--------------- create variable of interests

    fil_new <- new %>%
      dplyr::filter(SUBP == 1) %>%
      group_by(pltID) %>%
      dplyr::filter(row_number() == 1) %>%
      # reframe(unique(pltID))
      dplyr::select(all_of(c(
        "PLT_CN", "pltID", "STATECD", "UNITCD", "COUNTYCD", "PLOT",
        "INVYR", "MEASYEAR", "LAT", "LON", "ELEV", "ECOSUBCD",
        "OWNGRPCD", "MAPDEN", "STDAGE", "SITECLCD"

        # "SICOND", "SIBASE","GENUS", "SPECIES", "SCIENTIFIC_NAME"
      ))) %>%
      filter(!is.na("OWNGRPCD"))
    #-------------------------------------------------------------------------------#
    # write csv file with lat long and other key
    write.csv(x = fil_new, file = paste0(write.location, "loc_info_", st_abbr, ".csv"),
              row.names = FALSE)
  }
}

filterUniquePlot(dir.list = "../00_02_allSttatsCSVS/",
                 write.location = "../00_03_UniquePlotByState/")
# wrlocation <- "F:\\00_00_All_ind_states_march2023\\00_03_UniquePlotByState\\"
#-------------------------------------------------------------------------------#
```

1. The function `filterUniquePlot()` takes two arguments: 1) dir.list (the directory to search for data) and 2) write.location (the directory to write the processed data).
2. Inside the function, the sub-directories of the input directory are identified and stored in `listDir`
3. For each sub-directory, the function reads in four CSV files (PLOT, COND, REF_SPECIES, and TREE) using the function **fread** from `data.table()` package; `data.table::fread`. The files are read in using the path `paste0(db, "/", st_abbr, "_", tab[i], ".csv")`, where **db** is the name of the sub-directory, **st_abbr** is the state abbreviation extracted from the sub-directory name, and **tab[i]** is one of the four table names.
4. Creates a **PLT_CN** field in the **PLOT** table using the **PLOT** number, and a **pltID** field as a unique ID across the census.
5. Joins the **REF_SPECIES** table with the **TREE** table on the **SPCD** field, creating a new **SCIENTIFIC_NAME** field in the **TREE** table by pasting together the **GENUS** and **SPECIES** fields.
6. Drops certain columns from the **PLOT**, **COND**, and **TREE** tables.

The function writes the processed data to the specified `write.location` directory.

Visualize unique plots

We have `length(list.files('../00_03_UniquePlotByState/',pattern = ".csv$"))` states. Lets read them all and combine into one dataframe, As each state table has state abbreviation in its file name, we want to extract it and make one variable as state. However, **statecd** variable is available in the table.

```
library(dplyr)
library(sf)
library(magrittr)

# list state tables

combineAllStates <- function(tblLocation = getwd(), pattern = "*.csv$",
                             full.names = TRUE) {
  # read list of files
  tbls <- list.files(tblLocation, pattern = pattern, full.names = full.names)

  # create empty list

  slist <- list()

  # create st_abb columns and fill state abbreviation

  for (tab in tbls) {
    # extract state abbreviation
    ind <- which(tbls == tab)

    st_abbr <- substr(x = tab, start = (nchar(tbls[ind]) - 5),
                      stop = (nchar(tbls[ind]) - 4))

    stbl <- read.csv(file = tbls[ind]) %>% mutate(state_abb = st_abbr)

    slist[[ind]] <- stbl
  } # bind data frame
  allStateTable <- do.call("rbind", slist)

  return(allStateTable)
}


all_tables <- combineAllStates(tblLocation = "../00_03_UniquePlotByState/")
```

The function `combineAllStates` above code chunks takes in a directory location where state tables are saved, along with a file pattern and whether to include full names or not. It reads in all the `*.CSV` files in the directory with the specified pattern and creates an empty list (`slist`) to hold the state tables.

The function then loops through the list of files, extracting the state abbreviation from the file name and creating a new column in the table with the abbreviation. The modified table is then added to the list.

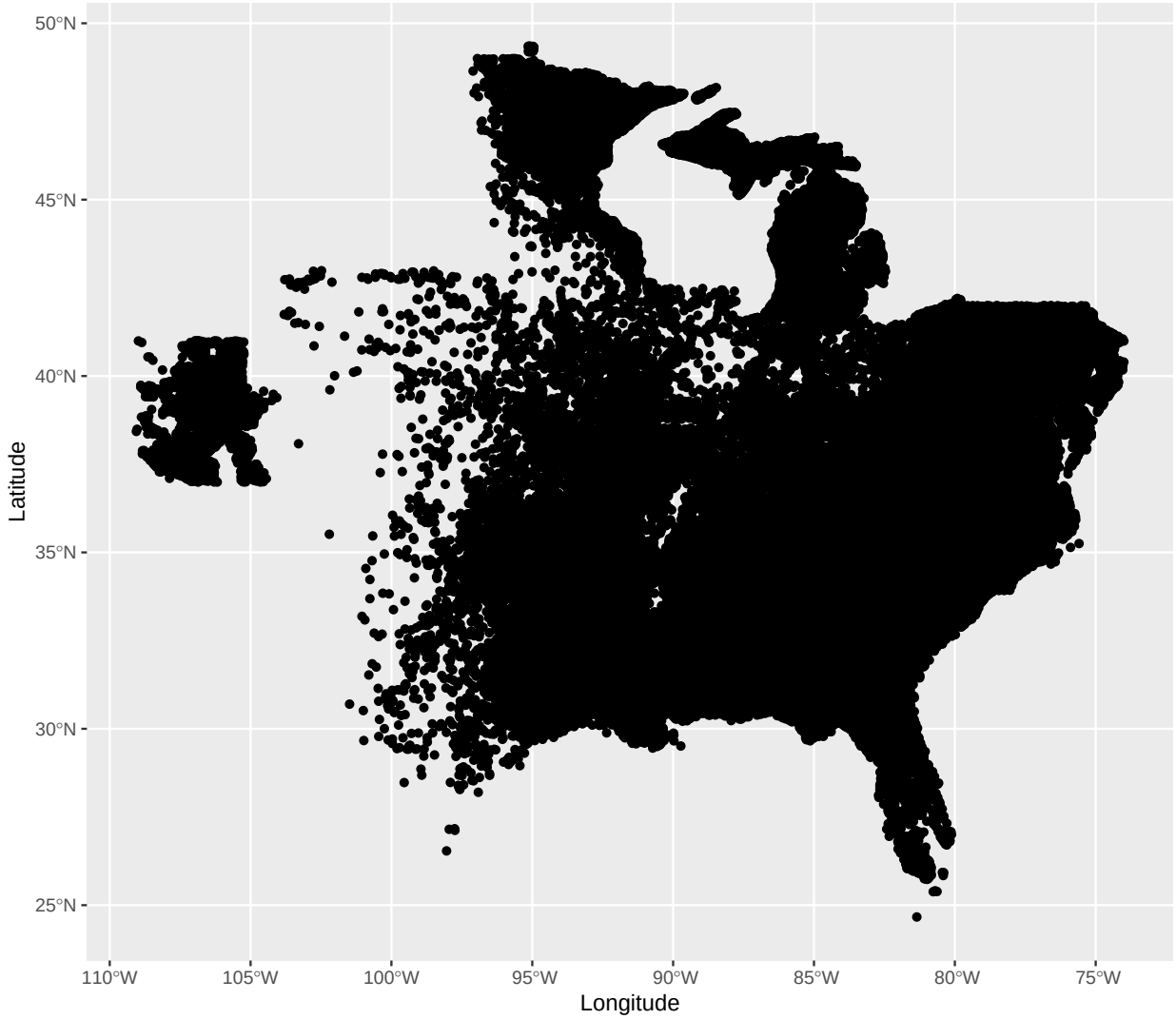
Finally, the list of modified tables is combined using `do.call("rbind", slist)` and returned as a single data frame containing all state tables with the added column for state abbreviation.

Now we have one single table with repeated plot information from all states. We can visualize them

```r
library(ggplot2)
library(USA.state.boundaries)


# Read data in as an sf object
sf_dat <- st_as_sf(all_tables, coords = c("LON", "LAT"), crs = 4326)

# Plot the locations of the data points
ggplot(data = sf_dat) +
  geom_sf() +
  labs(x = "Longitude", y = "Latitude")
```




```
##--- add state boundary

# load the map
state_bdr<- USA.state.boundaries::state_boundaries_wgs84
```

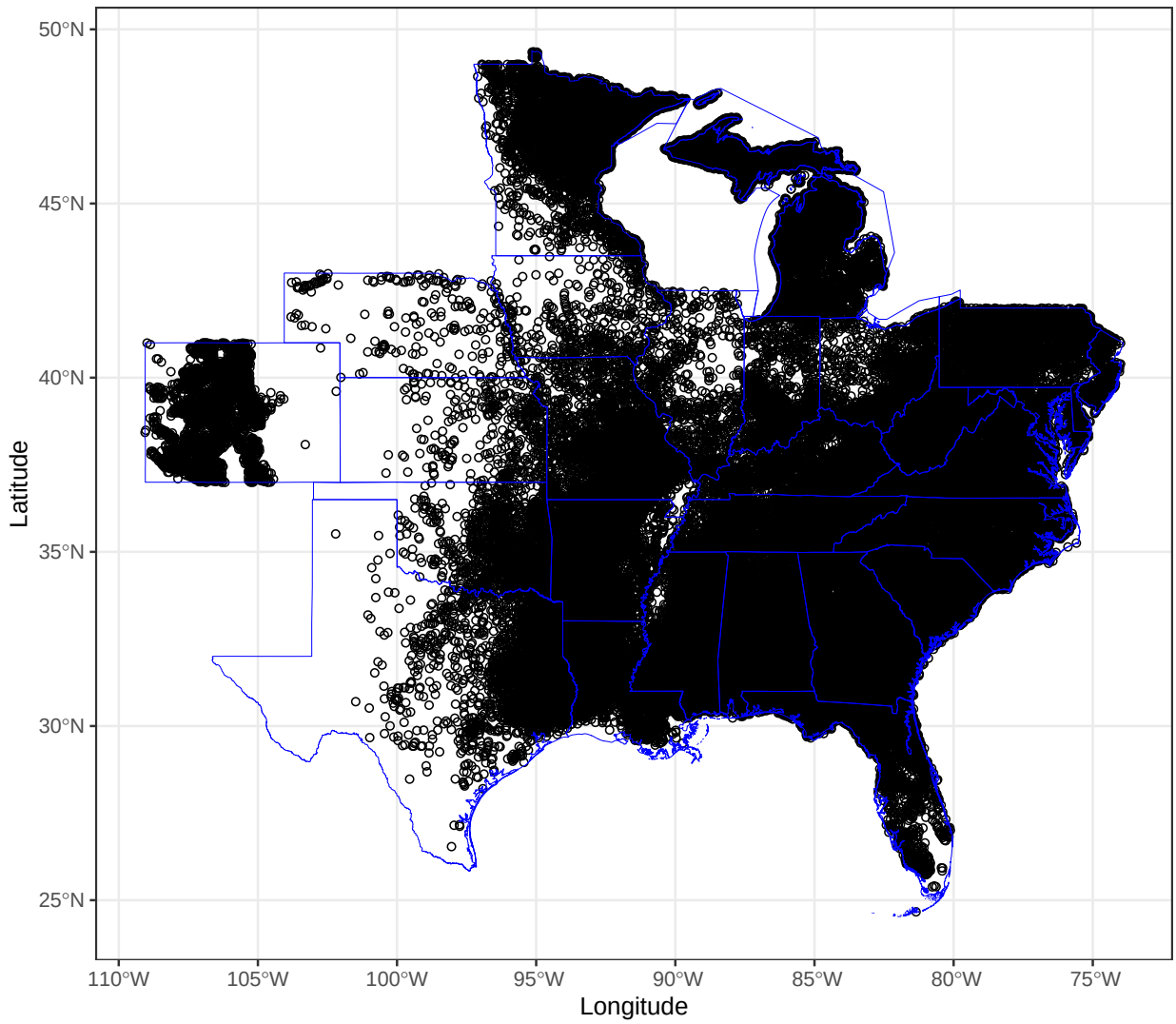
```r
## select states
state_abbr <- paste0(unique(all_tables$state_abb),collapse = "|")

sel_state <- dplyr::filter(state_bdr, grepl(state_abbr, STATE_ABBR))

## or dplyr::filter(state_bdr, str_detect(STATE_ABBR, state_abbr))

# st_write(sel_state,"../State_28.shp")

ggplot()+
  geom_sf(data = sf_dat, color = "black", shape = 21)+
  geom_sf(data = sel_state, color = "blue", fill = NA)+
 labs(x = "Longitude", y = "Latitude")+
  theme_bw(12)
```


50°N
45°N
40°N
35°N
30°N
25°N
Latitude
110°W
105°W
100°W
95°W
90°W
85°W
80°W
75°W
Longitude