\* All files referenced here as available in the “macrosystems-FIA” folder are currently living at <https://duke.box.com/macrosystems-FIA>.

\* Text in code font are meant to be typed in a bash shell with **bold text** replaced appropriately. \* Code snippets are included with additional examples in the file “Import FIA data v01.sh”, available in the [duke.box FIA folder](https://duke.box.com/macrosystems-FIA).

**Obtaining FIA data**

*From USFS DataMart*

The USFS provides FIA data for download at their DataMart website: <http://apps.fs.fed.us/fiadb-downloads/datamart.html>. The data are available as Microsoft Access database files, so if you have MS Access and want to use it, that’s an option. If you are interested in an open source solution, the data are also available as .csv files, one for each table in the FIA database. These are provided in several packaging options:

* Individual data tables for individual states, either zipped up or plain text. This format is useful if you want to quickly download a small subset of variables, but otherwise skip it.
* State-level .zip files containing all tables for the state. Useful if you need data from only a few states and want to avoid downloading the whole thing.
* The whole thing. I.e. a big .zip file containing all data for all states (i.e. <http://apps.fs.fed.us/fiadb-downloads/ENTIRE.zip>, linked at the bottom of the DataMart page).

The DataMart site is slow and buggy (as of Feb. 2015), but otherwise downloading these files is straightforward.

*From Us*

We are also hosting a copy of the complete .csv dataset at [macrosystems-FIA](https://duke.box.com/macrosystems-FIA), which may be quicker to download than trying to get through to the DataMart.

**Obtaining FIA data as a PSQL database**

If you do not have or want to use Microsoft Access, but you do want to use the FIA data as an actual database rather than a pile of .csv files, we’ve prepared a PSQL version that can be downloaded from [macrosystems-FIA](https://duke.box.com/macrosystems-FIA). To use it, you must first have access to a PSQL server (*see Installing PostgreSQL below*). Then import the downloaded file to your database:

gzcat **downloaded\_file\_ending\_in.psql.gz** | psql -d **new\_database\_name**

\*\*\*\* Note: The PSQL database I’ve currently made available is Rob Kooper’s version described [here](https://github.com/PecanProject/pecan/wiki/Installing-PEcAn-Data#fia-database) and used in Pecan. It has two shortcomings though. First, it depends on particular users (‘bety’ and ‘rkooper’) existing. Without creating those users first, errors are generated during the import, though it’s possible that they are harmless (I haven’t looked much into it). Second, that version of the database differs from importing the complete and current ENTIRE.csv. I don’t know how exactly—maybe it’s an older version, or maybe Rob pulled in only a subset of the data, etc.—but it seems like we should try to provide a complete and up to date copy. I’m sure it’s easy for me to export my own PSQL dump, I just haven’t gotten around to it / am waiting to talk to M. Dietze to find out if there’s a *good* reason that Rob’s db differs from mine.

**Converting FIA data to a PSQL database**

You may want to convert the FIA database to PSQL yourself, e.g. if the file linked above becomes obsolete, or you only need a few states worth of data and so don’t want to import the whole thing. In that case...

*Create an empty PSQL database following the FIA schema*

USFS provides an empty MS Access file that specifies the structure of the database. This can be converted for use in setting up a PSQL version of the database, but unfortunately the only way we’ve been able to do so is using a Windows-only application. Instructions follow, but as an alternative you can download a pre-converted PSQL schema file here \*\*\*\*\*\*\*\*\*\*\*\*\*.

*Converting the FIA Microsoft Access database shell to PSQL*

Download the .mdb defining the FIA database at <http://apps.fs.fed.us/fiadb-downloads/images/FIADB_version5_1.accdb>. Obtain the free Windows program *Access to PostgreSQL* at <http://www.bullzip.com/products/a2p/info.php>. Use the program to convert the downloaded .mdb file to PSQL format. Brief instructions:

* Run *Access to PostgreSQL*
* On the *Source Database* screen, choose the source file (the downloaded .mdb) but don’t worry about security settings. (We’re going to write to a file, not to an existing DB, so no security needed.)
* On the *Destination Database* screen select *Create dump file*, and set a file name for the output.
* On the *Select Tables* screen select all tables (should be default).
* On the *Transfer Options* screen the default values are fine.

The result of this procedure is a text file containing a complete description of the tables and variables in the FIA database, ready to be imported to your PSQL server. Actually, it’s not quite ready. First, two hacks:

Open the file in a text editor and delete the line (first one that doesn’t befin with ‘--‘):

CREATE DATABASE "";

Also, we’ve found that keeping all table and variable names lowercase will avoid some problems, but in the file you just created they are all uppercase. One way to fix this is:

cat **mixed\_case\_input\_file** | tr '[A-Z]' '[a-z]' > **all\_lowercase\_output\_file**

This step may not be necessary, but some of the tutorial below and some of our included code may not work as expected if you skip it.

*Creating and populating the PSQL database*

Import the PSQL database shell (downloaded or created above) to your PSQL server at the command line using (replace bold variables as appropriate):

psql -U **username** -d **new\_database\_name** < **PSQL\_shell\_file**

Now, raw FIA data downloaded from USFS in .csv format can be inserted into this database using the command:

psql -U **username** -d **database\_name** -c "\COPY **table\_name** FROM '**csv\_file\_name**' WITH CSV HEADER DELIMITER AS ',' NULL AS '' ENCODING 'UTF-8'"

Note that **table\_name** can be obtained from the .csv file name. E.g. for statewide files ‘XX\_BOUNDARY.CSV’ contains table ‘BOUNDARY’ for state ‘XX’. Remember though that if you followed our advice above the table name in your database will be lowercase (e.g. ‘boundary’).

In some cases this procedure may give errors, usually referring to a missing table (“relation XXXXX does not exist”) or an improper value (e.g. “invalid input syntax for integer: ###”). As far as I can tell these errors reflect actual problems with the FIA database. In the first case, for example, the .csv datasets really do contain files for tables that don’t exist in the MS Access schema. However, the errors I’ve checked out have been innocuous (e.g. the .csv files pointing to “missing” tables have all turned out to be empty).

See ‘Import FIA data v01.r’ in [macrosystems-FIA](https://duke.box.com/macrosystems-FIA) for an example of how to automate adding multiple tables from multiple sites.

**Converting FIA data to a different database format, e.g. MySQL**

While not covered here, the procedure for importing FIA to a format other than PSQL are probably similar. The same software company that provides the Microsoft Access --> PSQL converter even has [equivalent programs for some other database formats](http://www.bullzip.com/download.php).

**Installing PostgreSQL**

You’ll need a PSQL server running locally or access to a remote one in order to install and query a PSQL database. Binaries are available at <http://www.postgresql.org/download/>. That site has good documentation, and installation is fairly straightforward.

**Querying FIA data from a PSQL database**

Once the data are in a PSQL database they can be queried with standard commands. The script ‘FIA example queries v01.r’ in [macrosystems-FIA](https://duke.box.com/macrosystems-FIA) contains some examples using an R interface to PSQL, including extraction of plot- and tree-level information subsetted by census year, lat/lon, and species. To create your own queries, you may need to consult the FIADB User Guide (available from USFS, and also in macrosystems-FIA folder), which exhaustively defines every field of every table in the database.