

Installing a data set onto the TDR Web Retrieval System

The following pages are a first draft at outlining the steps involved in getting data up and accessible on the web: The following is a brief summary of the steps. Details follow.

1. Create a location on the Unix system for the data.
2. FTP the data from Stats Canada or UoT's ICPSR site.
3. Unbundle the data.
4. Create or modify a SAS program.
5. Create the SAS data set
6. Create the corresponding 'fm' and 'index' files.
7. Organize and create documentation.
8. Update the master index.

Overview

Once data set files have been transferred by ftp, the files must be formatted following specific guidelines in order for them to be mounted on the web for use. Some of this process is now 'automated'.

Each data set is stored in its own directory in *stats.uoguelph.ca* under */users/d*. The file system is only accessible from *stats.uoguelph.ca*. Directories within the */users/d/* directory, group data files according to source or major subject area (eg. LABOUR, SLID, SPECIAL, ICSPR).

The following pages give examples of all the files which must exist or be created for each data set, as well as notes on what type of editing of files may/must occur in order for the data set to run.

The producers of data sets do not all follow the same formats when compiling the data. Even within an organization such as STATS CANADA, data files come in a variety of ways. Some files arrive with all codebooks, users guides, SAS and SPSS programs intact. Some arrive with only an SPSS file and some rudimentary notes. In some instances the data may already be in a SAS or SPSS dataset or have a procedure in SAS capable of reading it (example - IMF).

When mounting a data set, some or all of the editing listed in the following pages will need to be performed. Begin by checking files to see what is included. Occasionally, codebooks, etc. can be obtained in paper format by contacting the supplier (eg. DLI, ICPSR etc). Some files arrive in a zipped format and must be unzipped to locate individual files within the set. Files should be transferred to the appropriate directory within */users/d*.

When editing files, check previously mounted data sets to make sure the format you are following contains all necessary information. Check with the appendix to make sure all files have been created and are located in the appropriate directory. **When each directory has been completed, ensure that all members of the data group can access files to modify them if necessary.** To do this type, *ls -l* to list all files and their modes. If necessary type *chmod -R 775 ** to ensure all files are read, write and execute by all members. If any files indicate the group is libdata or something else type *chgrp ugdata **.

1. Create a location

Logon to *stats.uoguelph.ca* and go to */users/d*. Locate the appropriate sub-directory to install the data. Once you change to this location determine a name for the directory to create. It should be associated with the data set, like *slid94*, *vaw93*, *gss9*.... Simply enter the command:

```
/users/d/HELP/create <datasetname>
```

It will take care of everything. See *Appendix A*, on Directory Structure, for an outline of how each data set is set up. Aside from the 'default' doc files, only the directories will be created with the appropriate rights and ownerships.

2. FTP the data

Use tools such as WSFTP or Rapid filer to move the data from STATS Canada or UoT onto the Unix system. You can do this in 2 steps, DLI to PC , PC to Unix, if you prefer. Remember to move compressed/binary files (zip, gz, PDF, WP) in binary mode. You may choose to ftp directly from the Unix prompt to the DLI site. This is a little more difficult.

3. Unbundle the data

The data is usually compressed and you will need to uncompress it. There are tools to do this on Unix. If the file has a zip extension, try *unzip <filename>*. If it has a gz extension try *gunzip <filename>*. If you end up with a tar file use, *tar -xvf <filename>*. Remember to clean up the compressed files. At this stage you should also rename the file using: *mv <dataset>.data*

4. Create SAS program

This can be the most difficult part. If you are lucky, then there will be a SAS program. It will probably still need to have modifications to it. Essentially there should be a header, libnames, proc formats, data set naming and compression, input statements, labels, format assignments and indexing. The best way to understand this is to look at examples. The SCF and GSS files are straight forward. See *Appendix B* for an example.

Some notes to follow are:

- Remove all introductory lines and insert header lines as in *Appendix B*. This includes, title, date, options and libnames.
- Prepare Proc Formats as the second step, BEFORE reading from rawdata files.
- In value statement ensure all values end in ' '; and note that it can't end in numbers. We pad it with an 'f' when you get variable names like q1 q2 q3.... It also can't exceed 8 characters. These are common problems with supplied SAS code.
- If the SAS file does not contain a format statement, one can be created using an editor like *xedit* or *vi*. You will need to be able to make global changes.
- SPSS files are close to SAS and can easily be modified using *Xedit* or *vi*

5. Create SAS dataset

Simply run the sas program by typing `sas <progname>`. This will create, in the data root, the `ssd01` file (sas dataset), the `snx01` file (index file) and the `sct01` file (formats file). The later can be very large if you have many pairings. In the PROGS directory it will also create a log file. It is extremely important to check this for error messages and compare the counts to the documentation. There will also be a listing file, containing the output from the contents procedure. Simply copy this to the root directory for the dataset, renaming it `<datasetname>/contents`. For example:

```
cp gss9.lst ../gss9.contents
```

6. Create fm and index files

This is now fully automated by a script, provided the SAS program is properly formatted. It will automatically create the *.fm files and a long and short index file. At the prompt type in:

```
/users/d/HELP/formmaker
```

It will ask for the location of the SAS program to use. Note you must already have created the appropriate FORMS directory.

See the sample index file in appendix C. You will need to edit this file to include the variables you want to subset by. By default it adds the first 4 in the SAS program, which is likely NOT what you want. Be sure this list is the same as the one in the Index procedure of your SAS program. This is simply used to speed up the subsetting procedure.

7. Documentation

Make sure all the documents are copied into the documents directory. You should set up a file called `<dataname>-readme.html` in the root for the data. There is a sample in `/users/d/HELP/readme.html` that you can copy over and edit. Update the description and include any HTML links to PDF, WP or similar files. If there are ASCII versions of the `cdbk`, `recl`, `quest`, or `ug` simply rename them appropriately.

8. Update the master index and clean-up.

Edit the file `/users/d/WWWDOCS/FORMS/index.fm`. You need to include the index file name, a title, a directory location and whether or not there is a 'weight' variable. If there is no weight variable then type in `NONE` at the end of the line, otherwise enter the variable name. This list is alphabetical.

The 2nd last check is to make sure you have not overfilled the DISK. Issue the command:

```
bdf /users/d/<directoryname> (For example bdf /users/d/SCF)
```

This will return the disk utilization. If we are over 95% we can move old files to the AUSPEX. Contact the DRC at Guelph for assistance here. The command `bdf`, without any options, will list ALL our directories and their utilization.

That's ALL :-)

Appendix A: Directory Structure

The Finished Data Set should include the following. Compare this to one of directories using the WWW.

Main/Root Directory

- Sub-directories - DOCS, PROGS, FORMS
- optionally RAWDATA, FRENCH
- Files - Format statement
 - <dataname>-readme.html
 - <dataname>.contents
 - <dataname>.data
 - <dataname>.ssd01
 - <dataname>.snx01
 - <dataname>.sct01

-substitute for dataname - ie gss9.ssd01

Sub-directories

DOCS

- Files - <dataname>.quest
 - <dataname>.cdbk
 - <dataname>.ug
 - <dataname>.recl

-If they don't exist as text files create using */users/d/HELP/doc dataname*.

PROGS

- Files - <dataname>.sas
 - <dataname>.log
 - <dataname>.lst

-optionally, there are spss program files, if supplied

FORMS

- Files - <dataname>.index
 - <dataname>.var.fm
 - <dataname>.varby.fm (created by copying var.fm file to varby.fm)

-also have several *.fm files, one for each subset variable

NOTE:

All files associated with a data set should contain the same file name as the directory in which they are stored.

e.g. Violence Against Women Survey resides in '/users/d/SPECIAL/VAW93'

Therefore, all corresponding files will begin with vaw93

- vaw93.data
- vaw93.sas
- vaw93.cdbk

This means that the full file name for vaw93.data is:

'/users/d/SPECIAL/VAW93/vaw93.data'

Appendix B: SAMPLE SAS FILE (using LABOUR/SLID94/slid94cj as sample data set)

slid94cj.sas:

Header typed in:

```
/*Survey of Labour and Income Dynamics 1994 Cross-Sectional Job File*/  
/* date and name of creator */  
options linesize=72; (can be larger where necessary)  
*options obs=500; (Used for testing)  
libname sasdata '/users/d/LABOUR/SLID94';  
libname library '/users/d/LABOUR/SLID94';
```

Format statement:

```
proc format library=library;
```

```
    value V037_F  
        1 = "Yes"  
        2 = "No" ;
```

Input statement:

```
data sasdata.slid94cj (compress=yes);  
    infile '/users/d/LABOUR/SLID94/slid94cj.data' Lrecl=513;  
  
    input  
        510 YRSCH18C  
        xyz 12-13;
```

Label:

```
label  
    NBABSCc ="No, absences this job"  
    ;
```

Format Assignment:

```
format  
    PUPID16c V001_F.  
    ;
```

Indexing and contents: the 'pairs' are the ones used to subset by - speeds things up.

```
proc datasets library=sasdata;  
    modify slid94cj;  
    index create pair=(pvreg25c regre25c sex21 marst26c);  
    contents data=slid94cj;
```

Appendix C: Example of index file:

PUREG25C/prov.fm/Province, scroll/1^8, true, slid94prov
REGRE25C/reg.fm/Region, scroll/1^8, true, slid94prov
MARST26C/mar.fm,Marital Status,scroll/1^8, true,formhelp
SEX21/Gender.sex.fm,scroll/2^8,true,formhelp
vars/slid94civar.fm, scroll,true

prov.fm = variable file for provincial designations
1^8 = means the box for this variable will be on the first line and spans 1 column for 8 rows
(*This can be adjusted to conform to size of boxes needed for different data sets*)
PUREG25C = Variable name
Province = title of selection box since PUREG25C is unrecognizable to users
scroll = scroll menu of choices (also input, radio...)
true = multiple selections

slid94civar.fm = is the selection box containing all variables held in the data set

formhelp = is a help file located in /users/d/HELP (*generic file is created as necessary*)

GLOSSARY OF COMMANDS

<p>mkdir <dirname> chmod -R 775 * chgrp -R ugdata cd & mv <from > <to> cd .. /users/d/HELP/refmt <fn> !<first letter of command> ~ X <fn> pico <fn> cp <from> <to> ls ls -l man <command> remove or rm rmdir sas unzip, gunzip, tar -xvf bdf</p>	<p>make directory change mode to read, write, execute for owner and group set group to ugdata change directory background job move move up one level reformat (awk script to reformat <i>var</i> files) back one command -ex. !s - execute last job beginning with s home directory - edit edit copy list files list mode of files manual remove file remove directory run sas uncompress tools check disk usage</p>
<p>X EDIT</p>	<p>BASIC COMMANDS</p>
<p><< >> dd ? / f F2 F3 qqit save file c/from/to/* 1 set scale on set arbchar on ^ z n1 n2</p>	<p>beginning of lines to be moved/removed end of lines to be moved/removed delete lines between these marks move back one command insert to indicate current line insert to indicate line insertion should follow inserts new line exits file where no changes have been made quit file without saving changes made saves changes without closing file saves changes and closes file change from something to something else the first time it appears show scale create an arbitrary character for editing set zone range from one number to another number</p>

