Imperial College
London

From datasets to metadatasets in Stata

Roger B. Newson
r.newson@imperial.ac.uk
http://www.rogernewsonresources.org.uk

Department of Primary Care and Public Health, Imperial College London

2020 London Stata Conference, 10–11 September, 2020
Downloadable from the conference website at
http://ideas.repec.org/s/boc/usug20.html

What are metadatasets?

- In Stata, a **metadataset** is a dataset (in a frame or a file), containing data about data.
- Currently, there are 3 principal kinds, with 1 observation per dataset, 1 observation per value (per value label), and 1 observation per variable (per dataset), respectively.
- They may be created as output datasets (or **resultssets**) by programs like `xdir`, `describe` or `descsave`[1, 2, 3].
- Alternatively, they may be created in a worksheet (using software such as LibreOffice Calc) and imported into Stata datasets using `import delimited` or `import excel`.
- And they may be output for **self–documentation** of a database.
- Alternatively (or even additionally), they may be input to specify **modifications** of a database.

What are metadatasets?

- In Stata, a **metadataset** is a dataset (in a frame or a file), containing data about data.
- Currently, there are 3 principal kinds, with 1 observation per dataset, 1 observation per value (per value label), and 1 observation per variable (per dataset), respectively.
- They may be created as output datasets (or **resultssets**) by programs like `xdir`, `describe` or `descsave`[1, 2, 3].
- Alternatively, they may be created in a worksheet (using software such as LibreOffice Calc) and imported into Stata datasets using `import delimited` or `import excel`.
- And they may be output for **self–documentation** of a database.
- Alternatively (or even additionally), they may be input to specify **modifications** of a database.

What are metadatasets?

- In Stata, a **metadataset** is a dataset (in a frame or a file), containing data about data.
- Currently, there are 3 principal kinds, with 1 observation per dataset, 1 observation per value (per value label), and 1 observation per variable (per dataset), respectively.
- They may be created as output datasets (or **resultssets**) by programs like `xdir`, `describe` or `descsave`[1, 2, 3].
- Alternatively, they may be created in a worksheet (using software such as LibreOffice Calc) and imported into Stata datasets using `import delimited` or `import excel`.
- And they may be output for **self–documentation** of a database.
- Alternatively (or even additionally), they may be input to specify **modifications** of a database.

What are metadatasets?

- In Stata, a **metadataset** is a dataset (in a frame or a file), containing data about data.
- Currently, there are 3 principal kinds, with 1 observation per dataset, 1 observation per value (per value label), and 1 observation per variable (per dataset), respectively.
- They may be created as output datasets (or **resultssets**) by programs like `xdir`, `describe` or `descsave`[1, 2, 3].
- Alternatively, they may be created in a worksheet (using software such as LibreOffice Calc) and imported into Stata datasets using `import delimited` or `import excel`.
- And they may be output for **self–documentation** of a database.
- Alternatively (or even additionally), they may be input to specify **modifications** of a database.

What are metadatasets?

- In Stata, a **metadataset** is a dataset (in a frame or a file), containing data about data.
- Currently, there are 3 principal kinds, with 1 observation per dataset, 1 observation per value (per value label), and 1 observation per variable (per dataset), respectively.
- They may be created as output datasets (or **resultssets**) by programs like `xdir`, `describe` or `descsave`[1, 2, 3].
- Alternatively, they may be created in a worksheet (using software such as LibreOffice Calc) and imported into Stata datasets using `import delimited` or `import excel`.
- And they may be output for **self–documentation** of a database.
- Alternatively (or even additionally), they may be input to specify **modifications** of a database.

What are metadatasets?

- In Stata, a **metadataset** is a dataset (in a frame or a file), containing data about data.
- Currently, there are 3 principal kinds, with 1 observation per dataset, 1 observation per value (per value label), and 1 observation per variable (per dataset), respectively.
- They may be created as output datasets (or **resultssets**) by programs like `xdir`, `describe` or `descsave`[1, 2, 3].
- Alternatively, they may be created in a worksheet (using software such as LibreOffice Calc) and imported into Stata datasets using `import delimited` or `import excel`.
- And they may be output for **self–documentation** of a database.
- Alternatively (or even additionally), they may be input to specify **modifications** of a database.

What are metadatasets?

- In Stata, a **metadataset** is a dataset (in a frame or a file), containing data about data.
- Currently, there are 3 principal kinds, with 1 observation per dataset, 1 observation per value (per value label), and 1 observation per variable (per dataset), respectively.
- They may be created as output datasets (or **resultssets**) by programs like `xdir`, `describe` or `descsave`[1, 2, 3].
- Alternatively, they may be created in a worksheet (using software such as LibreOffice Calc) and imported into Stata datasets using `import delimited` or `import excel`.
- And they may be output for **self–documentation** of a database.
- Alternatively (or even additionally), they may be input to specify **modifications** of a database.

Metadatasets with 1 observation per dataset

- The SSC package `xdir` creates an output dataset (or resultsset), with 1 observation for each file in a user–specified directory with a name conforming to a user–specified pattern, or (alternatively) with 1 observation for each frame in memory.
- This resultsset may be listed, output to a frame or a disk file, or saved in the current frame (overwriting any pre–existing dataset).
- The SSC package `descgen` is used in a `xdir` resultsset, and adds variables containing Stata dataset attributes (if applicable) for the listed files or frames.
- These dataset attributes include numbers of observations and variables, sort–lists of variables, dataset labels, and (optionally) dataset characteristics.
- We will demonstrate `xdir` and `descgen`, using a simple example.

Metadatasets with 1 observation per dataset

- The SSC package `xdir` creates an output dataset (or resultsset), with 1 observation for each file in a user–specified directory with a name conforming to a user–specified pattern, or (alternatively) with 1 observation for each frame in memory.
- This resultsset may be listed, output to a frame or a disk file, or saved in the current frame (overwriting any pre–existing dataset).
- The SSC package `descgen` is used in a `xdir` resultsset, and adds variables containing Stata dataset attributes (if applicable) for the listed files or frames.
- These dataset attributes include numbers of observations and variables, sort–lists of variables, dataset labels, and (optionally) dataset characteristics.
- We will demonstrate `xdir` and `descgen`, using a simple example.

Metadatasets with 1 observation per dataset

- The SSC package `xdir` creates an output dataset (or resultsset), with 1 observation for each file in a user–specified directory with a name conforming to a user–specified pattern, or (alternatively) with 1 observation for each frame in memory.
- This resultsset may be listed, output to a frame or a disk file, or saved in the current frame (overwriting any pre–existing dataset).
- The SSC package `descgen` is used in a `xdir` resultsset, and adds variables containing Stata dataset attributes (if applicable) for the listed files or frames.
- These dataset attributes include numbers of observations and variables, sort–lists of variables, dataset labels, and (optionally) dataset characteristics.
- We will demonstrate `xdir` and `descgen`, using a simple example.

Metadatasets with 1 observation per dataset

- The SSC package `xdir` creates an output dataset (or resultsset), with 1 observation for each file in a user–specified directory with a name conforming to a user–specified pattern, or (alternatively) with 1 observation for each frame in memory.
- This resultsset may be listed, output to a frame or a disk file, or saved in the current frame (overwriting any pre–existing dataset).
- The SSC package `descgen` is used in a `xdir` resultsset, and adds variables containing Stata dataset attributes (if applicable) for the listed files or frames.
- These dataset attributes include numbers of observations and variables, sort–lists of variables, dataset labels, and (optionally) dataset characteristics.
- We will demonstrate `xdir` and `descgen`, using a simple example.

Metadatasets with 1 observation per dataset

- The SSC package `xdir` creates an output dataset (or resultsset), with 1 observation for each file in a user–specified directory with a name conforming to a user–specified pattern, or (alternatively) with 1 observation for each frame in memory.
- This resultsset may be listed, output to a frame or a disk file, or saved in the current frame (overwriting any pre–existing dataset).
- The SSC package `descgen` is used in a `xdir` resultsset, and adds variables containing Stata dataset attributes (if applicable) for the listed files or frames.
- These dataset attributes include numbers of observations and variables, sort–lists of variables, dataset labels, and (optionally) dataset characteristics.
- We will demonstrate `xdir` and `descgen`, using a simple example.

Metadatasets with 1 observation per dataset

- The SSC package `xdir` creates an output dataset (or resultsset), with 1 observation for each file in a user–specified directory with a name conforming to a user–specified pattern, or (alternatively) with 1 observation for each frame in memory.
- This resultsset may be listed, output to a frame or a disk file, or saved in the current frame (overwriting any pre–existing dataset).
- The SSC package `descgen` is used in a `xdir` resultsset, and adds variables containing Stata dataset attributes (if applicable) for the listed files or frames.
- These dataset attributes include numbers of observations and variables, sort–lists of variables, dataset labels, and (optionally) dataset characteristics.
- We will demonstrate `xdir` and `descgen`, using a simple example.

Using `xdir` to create a dataset of `.dta` files in the current folder

We use `xdir` to create a resultsset in the current frame, with 1 observation per file in the working folder with the extension `.dta`. We then `describe` and `list` this resultsset.

```
. xdir, dir(.) pattern(*.dta) fast;

. desc, fu;

Contains data
  obs:             3
 vars:             2
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
dirname         str1    %-9s                  Directory name
filename        str13   %-13s                 File name
-------------------------------------------------------------------------------
Sorted by: dirname  filename
     Note: Dataset has changed since last saved.

. list, abbr(32);

     +-------------------------+
     | dirname   filename      |
     |-------------------------|
  1. | .         mydesc.dta    |
  2. | .         myvallabs.dta |
  3. | .         myxauto.dta   |
     +-------------------------+
```

Using `descgen` to generate Stata dataset attribute variables

We then use `descgen` to generate a list of additional variables, containing Stata dataset attributes of the files, and `describe` this extended resultsset.

```
. descgen, label;
File names input from variable: filename

. desc, fu;

Contains data
  obs:             3
 vars:            10
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
dirname         str1    %-9s                  Directory name
filename        str13   %-13s                 File name
isdta           byte    %8.0g                 Stata dataset status indicator
nobs            byte    %8.0g                 N of observations
nvar            byte    %8.0g                 N of variables
width           byte    %8.0g                 Width of observation (bytes)
size            int     %8.0g                 Size of dataset (bytes)
sortedby        str16   %-16s                 Sort list of variables
allvars         str87   %-87s                 List of all variables
dslabel         str25   %-25s                 Dataset label
-------------------------------------------------------------------------------
Sorted by: dirname  filename
     Note: Dataset has changed since last saved.
```

Some attributes of the datasets in the `.dta` files

We then `list` a selection of the attributes, documenting the `.dta` files in the current folder:

```
. list filename nobs nvar size sortedby dslabel, abbr(32);

     +-------------------------------------------------------------------------------+
     | filename        nobs   nvar   size   sortedby           dslabel                |
     |-------------------------------------------------------------------------------|
  1. | mydesc.dta        23      8   1725   dset order         Variables in all datasets |
  2. | myvallabs.dta      4      3     60   vallabname value   Value labels           |
  3. | myxauto.dta       74     13   4144   foreign make       Extended auto data     |
     +-------------------------------------------------------------------------------+
```

We will discuss how these datasets were made later!

Metadatasets with 1 observation per value (per value label)

- Stata uses named **value labels** to map integer values to text labels.
- One or more value labels may be stored in a metadataset with one observation per integer value per value label, and 3 variables, containing the label name, the integer code, and the text label, respectively.
- Such a metadataset can be input from a delimited text file (using `import delimited`), or even from a Microsoft Excel spreadsheet (using `import excel`).
- In this metadataset, the SSC package `vallabdef` can input these 3 variables, and create the corresponding value labels.
- These value labels are not often useful in the current frame.
- *However*, they can be transferred to other frames, where they *might* be useful, using the SSC package `vallabsave`, or using other means that we will see later.

Metadatasets with 1 observation per value (per value label)

- Stata uses named **value labels** to map integer values to text labels.
- One or more value labels may be stored in a metadataset with one observation per integer value per value label, and 3 variables, containing the label name, the integer code, and the text label, respectively.
- Such a metadataset can be input from a delimited text file (using `import delimited`), or even from a Microsoft Excel spreadsheet (using `import excel`).
- In this metadataset, the SSC package `vallabdef` can input these 3 variables, and create the corresponding value labels.
- These value labels are not often useful in the current frame.
- *However*, they can be transferred to other frames, where they *might* be useful, using the SSC package `vallabsave`, or using other means that we will see later.

Metadatasets with 1 observation per value (per value label)

- Stata uses named **value labels** to map integer values to text labels.
- One or more value labels may be stored in a metadataset with one observation per integer value per value label, and 3 variables, containing the label name, the integer code, and the text label, respectively.
- Such a metadataset can be input from a delimited text file (using `import delimited`), or even from a Microsoft Excel spreadsheet (using `import excel`).
- In this metadataset, the SSC package `vallabdef` can input these 3 variables, and create the corresponding value labels.
- These value labels are not often useful in the current frame.
- *However*, they can be transferred to other frames, where they *might* be useful, using the SSC package `vallabsave`, or using other means that we will see later.

Metadatasets with 1 observation per value (per value label)

- Stata uses named **value labels** to map integer values to text labels.
- One or more value labels may be stored in a metadataset with one observation per integer value per value label, and 3 variables, containing the label name, the integer code, and the text label, respectively.
- Such a metadataset can be input from a delimited text file (using `import delimited`), or even from a Microsoft Excel spreadsheet (using `import excel`).
- In this metadataset, the SSC package `vallabdef` can input these 3 variables, and create the corresponding value labels.
- These value labels are not often useful in the current frame.
- *However*, they can be transferred to other frames, where they *might* be useful, using the SSC package `vallabsave`, or using other means that we will see later.

Metadatasets with 1 observation per value (per value label)

- Stata uses named **value labels** to map integer values to text labels.
- One or more value labels may be stored in a metadataset with one observation per integer value per value label, and 3 variables, containing the label name, the integer code, and the text label, respectively.
- Such a metadataset can be input from a delimited text file (using `import delimited`), or even from a Microsoft Excel spreadsheet (using `import excel`).
- In this metadataset, the SSC package `vallabdef` can input these 3 variables, and create the corresponding value labels.
- These value labels are not often useful in the current frame.
- *However*, they can be transferred to other frames, where they *might* be useful, using the SSC package `vallabsave`, or using other means that we will see later.

Metadatasets with 1 observation per value (per value label)

- Stata uses named **value labels** to map integer values to text labels.
- One or more value labels may be stored in a metadataset with one observation per integer value per value label, and 3 variables, containing the label name, the integer code, and the text label, respectively.
- Such a metadataset can be input from a delimited text file (using `import delimited`), or even from a Microsoft Excel spreadsheet (using `import excel`).
- In this metadataset, the SSC package `vallabdef` can input these 3 variables, and create the corresponding value labels.
- These value labels are not often useful in the current frame.
- *However*, they can be transferred to other frames, where they *might* be useful, using the SSC package `vallabsave`, or using other means that we will see later.

Metadatasets with 1 observation per value (per value label)

- Stata uses named **value labels** to map integer values to text labels.
- One or more value labels may be stored in a metadataset with one observation per integer value per value label, and 3 variables, containing the label name, the integer code, and the text label, respectively.
- Such a metadataset can be input from a delimited text file (using `import delimited`), or even from a Microsoft Excel spreadsheet (using `import excel`).
- In this metadataset, the SSC package `vallabdef` can input these 3 variables, and create the corresponding value labels.
- These value labels are not often useful in the current frame.
- *However*, they can be transferred to other frames, where they *might* be useful, using the SSC package `vallabsave`, or using other means that we will see later.

A metadataset with 1 observation per value (per value label)

We start by `describe`ing and `list`ing a dataset in the current frame, with 1 observation per value per value label.

```
.   desc, fu;

Contains data
  obs:             4                          Value labels
 vars:             3
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
vallabname      str6    %9s                   Value label name
value           byte    %10.0g                Numeric value
label           str8    %9s                   Label
-------------------------------------------------------------------------------
Sorted by: vallabname  value
     Note: Dataset has changed since last saved.
.   list, abbr(32) sepby(vallabname);

     +--------------------------------+
     | vallabname   value      label |
     |--------------------------------|
  1. |     origin       0   Domestic |
  2. |     origin       1    Foreign |
     |--------------------------------|
  3. |         us       0     Non-US |
  4. |         us       1         US |
     +--------------------------------+
```

Creating value labels in the metadataset

This dataset contains no value labels. *However*, these value labels can be created in the current frame, using `vallabdef` to extract them from the 3 variables:

```
.   label list;
.   vallabdef vallabname value label;
.   label list;
origin:
           0 Domestic
           1 Foreign
us:
           0 Non-US
           1 US
```

These value labels are not very useful in the current data frame. *However*, they can be can be transferred to other data frames where they might be useful. This can be done using the `vallabtran` module of the SSC package `vallabsave`. Other ways of doing this will be revealed later!

Metadatasets with 1 observation per variable (per dataset)

- Traditionally, these are created using the official Stata command `describe`, or the SSC package `descsave`[1, 2, 3].
- They have 1 observation per variable, and data on variable attributes, such as names, modes (numeric or string), storage types, formats, value labels, variable labels, and sometimes even variable characteristics.
- With Stata Version 16, we now have the SSC package `invdesc`, which modifies the attributes of variables in the current data frame, using attributes from a `descsave` resultsset in a second data frame, and value labels from a third data frame.
- Note that the datasets in the second and third data frames may be created manually in a spreadsheet.
- The `invdesc` package has a second module `sinvdesc`, which tries to find the new variable attributes for the current dataset in the observations of the current dataset.

Metadatasets with 1 observation per variable (per dataset)

- Traditionally, these are created using the official Stata command `describe`, or the SSC package `descsave`[1, 2, 3].
- They have 1 observation per variable, and data on variable attributes, such as names, modes (numeric or string), storage types, formats, value labels, variable labels, and sometimes even variable characteristics.
- With Stata Version 16, we now have the SSC package `invdesc`, which modifies the attributes of variables in the current data frame, using attributes from a `descsave` resultsset in a second data frame, and value labels from a third data frame.
- Note that the datasets in the second and third data frames may be created manually in a spreadsheet.
- The `invdesc` package has a second module `sinvdesc`, which tries to find the new variable attributes for the current dataset in the observations of the current dataset.

Metadatasets with 1 observation per variable (per dataset)

- Traditionally, these are created using the official Stata command `describe`, or the SSC package `descsave`[1, 2, 3].
- They have 1 observation per variable, and data on variable attributes, such as names, modes (numeric or string), storage types, formats, value labels, variable labels, and sometimes even variable characteristics.
- With Stata Version 16, we now have the SSC package `invdesc`, which modifies the attributes of variables in the current data frame, using attributes from a `descsave` resultsset in a second data frame, and value labels from a third data frame.
- Note that the datasets in the second and third data frames may be created manually in a spreadsheet.
- The `invdesc` package has a second module `sinvdesc`, which tries to find the new variable attributes for the current dataset in the observations of the current dataset.

Metadatasets with 1 observation per variable (per dataset)

- Traditionally, these are created using the official Stata command `describe`, or the SSC package `descsave`[1, 2, 3].
- They have 1 observation per variable, and data on variable attributes, such as names, modes (numeric or string), storage types, formats, value labels, variable labels, and sometimes even variable characteristics.
- With Stata Version 16, we now have the SSC package `invdesc`, which modifies the attributes of variables in the current data frame, using attributes from a `descsave` resultsset in a second data frame, and value labels from a third data frame.
- Note that the datasets in the second and third data frames may be created manually in a spreadsheet.
- The `invdesc` package has a second module `sinvdesc`, which tries to find the new variable attributes for the current dataset in the observations of the current dataset.

Metadatasets with 1 observation per variable (per dataset)

- Traditionally, these are created using the official Stata command `describe`, or the SSC package `descsave`[1, 2, 3].
- They have 1 observation per variable, and data on variable attributes, such as names, modes (numeric or string), storage types, formats, value labels, variable labels, and sometimes even variable characteristics.
- With Stata Version 16, we now have the SSC package `invdesc`, which modifies the attributes of variables in the current data frame, using attributes from a `descsave` resultsset in a second data frame, and value labels from a third data frame.
- Note that the datasets in the second and third data frames may be created manually in a spreadsheet.
- The `invdesc` package has a second module `sinvdesc`, which tries to find the new variable attributes for the current dataset in the observations of the current dataset.

Metadatasets with 1 observation per variable (per dataset)

- Traditionally, these are created using the official Stata command `describe`, or the SSC package `descsave`[1, 2, 3].
- They have 1 observation per variable, and data on variable attributes, such as names, modes (numeric or string), storage types, formats, value labels, variable labels, and sometimes even variable characteristics.
- With Stata Version 16, we now have the SSC package `invdesc`, which modifies the attributes of variables in the current data frame, using attributes from a `descsave` resultsset in a second data frame, and value labels from a third data frame.
- Note that the datasets in the second and third data frames may be created manually in a spreadsheet.
- The `invdesc` package has a second module `sinvdesc`, which tries to find the new variable attributes for the current dataset in the observations of the current dataset.

Example: Creating the 3 datasets that we saw earlier using `invdesc`

- This Example includes all previous Examples in this presentation, and creates the 3 datasets that we listed earlier (`mydesc`, `myvallabs` and `myxauto`) in frames.
- We start with 3 tab–delimited `.txt` files with the same names.
- We create the metadataset `mydesc` from `mydesc.txt`, using the module `sinvdesc` to create a metadataset with one observation per variable per dataset to be created.
- We then create the metadataset `myvallabs` from `myvallabs.txt`, using `invdesc` to create the dataset and `vallabdef` to create the value labels.
- We then create the dataset `myxauto` from `myxauto.txt`, using `invdesc` to get the variable attributes from frame `mydesc` and the value labels from frame `myvallabs`.
- Finally, after saving all the frames to disk files, we use `xdir` and `descgen` to create a metadataset with 1 observation per file.

Example: Creating the 3 datasets that we saw earlier using `invdesc`

- This Example includes all previous Examples in this presentation, and creates the 3 datasets that we listed earlier (`mydesc`, `myvallabs` and `myxauto`) in frames.
- We start with 3 tab–delimited `.txt` files with the same names.
- We create the metadataset `mydesc` from `mydesc.txt`, using the module `sinvdesc` to create a metadataset with one observation per variable per dataset to be created.
- We then create the metadataset `myvallabs` from `myvallabs.txt`, using `invdesc` to create the dataset and `vallabdef` to create the value labels.
- We then create the dataset `myxauto` from `myxauto.txt`, using `invdesc` to get the variable attributes from frame `mydesc` and the value labels from frame `myvallabs`.
- Finally, after saving all the frames to disk files, we use `xdir` and `descgen` to create a metadataset with 1 observation per file.

Example: Creating the 3 datasets that we saw earlier using `invdesc`

- This Example includes all previous Examples in this presentation, and creates the 3 datasets that we listed earlier (`mydesc`, `myvallabs` and `myxauto`) in frames.
- We start with 3 tab–delimited `.txt` files with the same names.
- We create the metadataset `mydesc` from `mydesc.txt`, using the module `sinvdesc` to create a metadataset with one observation per variable per dataset to be created.
- We then create the metadataset `myvallabs` from `myvallabs.txt`, using `invdesc` to create the dataset and `vallabdef` to create the value labels.
- We then create the dataset `myxauto` from `myxauto.txt`, using `invdesc` to get the variable attributes from frame `mydesc` and the value labels from frame `myvallabs`.
- Finally, after saving all the frames to disk files, we use `xdir` and `descgen` to create a metadataset with 1 observation per file.

Example: Creating the 3 datasets that we saw earlier using `invdesc`

- This Example includes all previous Examples in this presentation, and creates the 3 datasets that we listed earlier (`mydesc`, `myvallabs` and `myxauto`) in frames.
- We start with 3 tab–delimited `.txt` files with the same names.
- We create the metadataset `mydesc` from `mydesc.txt`, using the module `sinvdesc` to create a metadataset with one observation per variable per dataset to be created.
- We then create the metadataset `myvallabs` from `myvallabs.txt`, using `invdesc` to create the dataset and `vallabdef` to create the value labels.
- We then create the dataset `myxauto` from `myxauto.txt`, using `invdesc` to get the variable attributes from frame `mydesc` and the value labels from frame `myvallabs`.
- Finally, after saving all the frames to disk files, we use `xdir` and `descgen` to create a metadataset with 1 observation per file.

Example: Creating the 3 datasets that we saw earlier using `invdesc`

- This Example includes all previous Examples in this presentation, and creates the 3 datasets that we listed earlier (`mydesc`, `myvallabs` and `myxauto`) in frames.
- We start with 3 tab–delimited `.txt` files with the same names.
- We create the metadataset `mydesc` from `mydesc.txt`, using the module `sinvdesc` to create a metadataset with one observation per variable per dataset to be created.
- We then create the metadataset `myvallabs` from `myvallabs.txt`, using `invdesc` to create the dataset and `vallabdef` to create the value labels.
- We then create the dataset `myxauto` from `myxauto.txt`, using `invdesc` to get the variable attributes from frame `mydesc` and the value labels from frame `myvallabs`.
- Finally, after saving all the frames to disk files, we use `xdir` and `descgen` to create a metadataset with 1 observation per file.

Example: Creating the 3 datasets that we saw earlier using `invdesc`

- This Example includes all previous Examples in this presentation, and creates the 3 datasets that we listed earlier (`mydesc`, `myvallabs` and `myxauto`) in frames.
- We start with 3 tab–delimited `.txt` files with the same names.
- We create the metadataset `mydesc` from `mydesc.txt`, using the module `sinvdesc` to create a metadataset with one observation per variable per dataset to be created.
- We then create the metadataset `myvallabs` from `myvallabs.txt`, using `invdesc` to create the dataset and `vallabdef` to create the value labels.
- We then create the dataset `myxauto` from `myxauto.txt`, using `invdesc` to get the variable attributes from frame `mydesc` and the value labels from frame `myvallabs`.
- Finally, after saving all the frames to disk files, we use `xdir` and `descgen` to create a metadataset with 1 observation per file.

Example: Creating the 3 datasets that we saw earlier using `invdesc`

- This Example includes all previous Examples in this presentation, and creates the 3 datasets that we listed earlier (`mydesc`, `myvallabs` and `myxauto`) in frames.
- We start with 3 tab–delimited `.txt` files with the same names.
- We create the metadataset `mydesc` from `mydesc.txt`, using the module `sinvdesc` to create a metadataset with one observation per variable per dataset to be created.
- We then create the metadataset `myvallabs` from `myvallabs.txt`, using `invdesc` to create the dataset and `vallabdef` to create the value labels.
- We then create the dataset `myxauto` from `myxauto.txt`, using `invdesc` to get the variable attributes from frame `mydesc` and the value labels from frame `myvallabs`.
- Finally, after saving all the frames to disk files, we use `xdir` and `descgen` to create a metadataset with 1 observation per file.

Importing `mydesc.txt` into frame `mydesc`

We start by importing the text file into the frame as a string–only dataset (using `import delimited` with the `stringcols(_all)` option), and `describe` it:

```
.   import delimited using mydesc.txt, stringcols(_all) varnames(1) clear;
(7 vars, 23 obs)
.   desc, fu;

Contains data
  obs:            23
 vars:             7
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
dset            str9    %9s
name            str12   %12s
type            str6    %9s
isnumeric       str1    %9s
format          str6    %9s
vallab          str6    %9s
varlab          str34   %34s
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

We see that the new dataset contains only unlabelled string variables, *mostly* named like those in a `describe` resultsset.

Converting the string–only dataset to a `describe` resultsset

We then type the magic word `sinvdesc`, and `describe` the dataset again:

```
.   sinvdesc;
.   desc, fu;

Contains data
  obs:            23
 vars:             7
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
dset            str9    %9s                   Data set
name            str12   %12s                  Variable name
type            str6    %9s                   Storage type
isnumeric       byte    %10.0g                Numeric indicator
format          str6    %9s                   Display format
vallab          str6    %9s                   Value label
varlab          str34   %34s                  Variable label
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

We see that the variables are now labelled, and `isnumeric` is now a `byte` variable. The dataset is now a modified `describe` resultsset, without the numeric variable `order`, but with an added string variable `dset`, identifying the dataset in which each variable will live.

Listing the synthetic `describe` resultsset

After sorting the resultsset using the SSC package `keyby`, we can `list` it, grouping by `dset`:

```
.   by dset: list name type format vallab varlab, abbr(32);

-------------------------------------------------------------------------------
-> dset = mydesc

     +-----------------------------------------------------+
     |      name   type   format   vallab              varlab |
     |-----------------------------------------------------|
  1. |      dset                                    Data set |
  2. |      name                                Variable name |
  3. |      type                                 Storage type |
  4. | isnumeric                               Numeric indicator |
  5. |    format                                  Display format |
     |-----------------------------------------------------|
  6. |    vallab                                    Value label |
  7. |    varlab                                  Variable label |
     +-----------------------------------------------------+

-------------------------------------------------------------------------------
-> dset = myvallabs

     +-----------------------------------------------------+
     |       name   type   format   vallab              varlab |
     |-----------------------------------------------------|
  1. | vallabname                             Value label name |
  2. |      value                                Numeric value |
  3. |      label                                        Label |
     +-----------------------------------------------------+
```

(These are only the variables in the first 2 datasets.)

Listing the synthetic `describe` resultsset (continued)

And here are the variables for the last dataset `myxauto`:

```
-------------------------------------------------------------------------------
-> dset = myxauto

     +-------------------------------------------------------------------------+
     |          name     type   format   vallab                       varlab |
     |-------------------------------------------------------------------------|
  1. |       foreign     byte    %8.0g   origin                     Car type |
  2. |          make    str17    %-17s                     Make and Model |
  3. |         price      int   %8.0gc                             Price |
  4. |         rep78     byte    %8.0g                 Repair Record 1978 |
  5. |      headroom    float    %6.1f                     Headroom (in.) |
     |-------------------------------------------------------------------------|
  6. |         trunk     byte    %8.0g                Trunk space (cu. ft.) |
  7. |        length      int    %8.0g                       Length (in.) |
  8. |          turn     byte    %8.0g                  Turn Circle (ft.)  |
  9. | displacement      int    %8.0g               Displacement (cu. in.) |
 10. |   gear_ratio    float    %6.2f                          Gear Ratio |
     |-------------------------------------------------------------------------|
 11. |           us     byte    %8.0g       us            US or non-US model |
 12. |         tons   double   %10.0g                     Weight (US tons) |
 13. |          npm   double   %10.0g   Fuel consumption (nipperkins/mile) |
     +-------------------------------------------------------------------------+
```

These variables look as if they will belong to a modified `auto` dataset.

Importing `myvallabs.txt` into frame `myvallabs`

We start by importing the text file into the frame as a string–only dataset, and `describe` it:

```
.   import delimited using myvallabs.txt, stringcols(_all) varnames(1) clear;
(3 vars, 4 obs)
.   desc, fu;

Contains data
  obs:             4
 vars:             3
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
vallabname      str6    %9s
value           str1    %9s
label           str8    %9s
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

We see that the new dataset contains 3 unlabelled string variables.

Converting the string–only dataset to a metadataset of value labels

We then use `invdesc` to convert the unlabelled string variables, using the frame `mydesc` (created earlier) as the descriptive frame:

```
.   invdesc, dframe(mydesc);
.   desc, fu;

Contains data
  obs:             4
 vars:             3
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
vallabname      str6    %9s                   Value label name
value           byte    %10.0g                Numeric value
label           str8    %9s                   Label
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

We see that the variables are now labelled, and `value` is now numeric.

Creating value labels in the metadataset (revisited)

We can now `list` the dataset, `save` it to a disk file, and create the value labels in the current data frame, using `vallabdef` as before:

```
.   list, abbr(32) sepby(vallabname);

     +--------------------------------+
     | vallabname   value      label |
     |--------------------------------|
  1. |     origin       0   Domestic |
  2. |     origin       1    Foreign |
     |--------------------------------|
  3. |         us       0     Non-US |
  4. |         us       1         US |
     +--------------------------------+
.   save myvallabs.dta, replace;
file myvallabs.dta saved
.   label list;
.   vallabdef vallabname value label;
.   label list;
origin:
           0 Domestic
           1 Foreign
us:
           0 Non-US
           1 US
```

As we said before, these value labels are not very useful in the current data frame. *However*, we will be using them in the next dataset, which we will now create in another data frame!

Importing `myxauto.txt` into frame `default`

Again, we `import` and `describe` a string–only dataset:

```
. import delimited using myxauto.txt, stringcols(_all) varnames(1) clear;
(13 vars, 74 obs)

. desc, fu;

Contains data
  obs:            74
 vars:            13
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
foreign         str1    %9s
make            str17   %17s
price           str5    %9s
rep78           str1    %9s
headroom        str4    %9s
trunk           str2    %9s
length          str3    %9s
turn            str2    %9s
displacement    str3    %9s
gear_ratio      str9    %9s
us              str1    %9s
tons            str5    %9s
npm             str16   %16s
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

This time, *most* of the string variables have familiar–looking names.

Converting the string–only dataset to a modified `auto` dataset

And when we call `invdesc`, we specify descriptives and value labels from metadatasets in frames `mydesc` and `myvallabs`, which we created earlier:

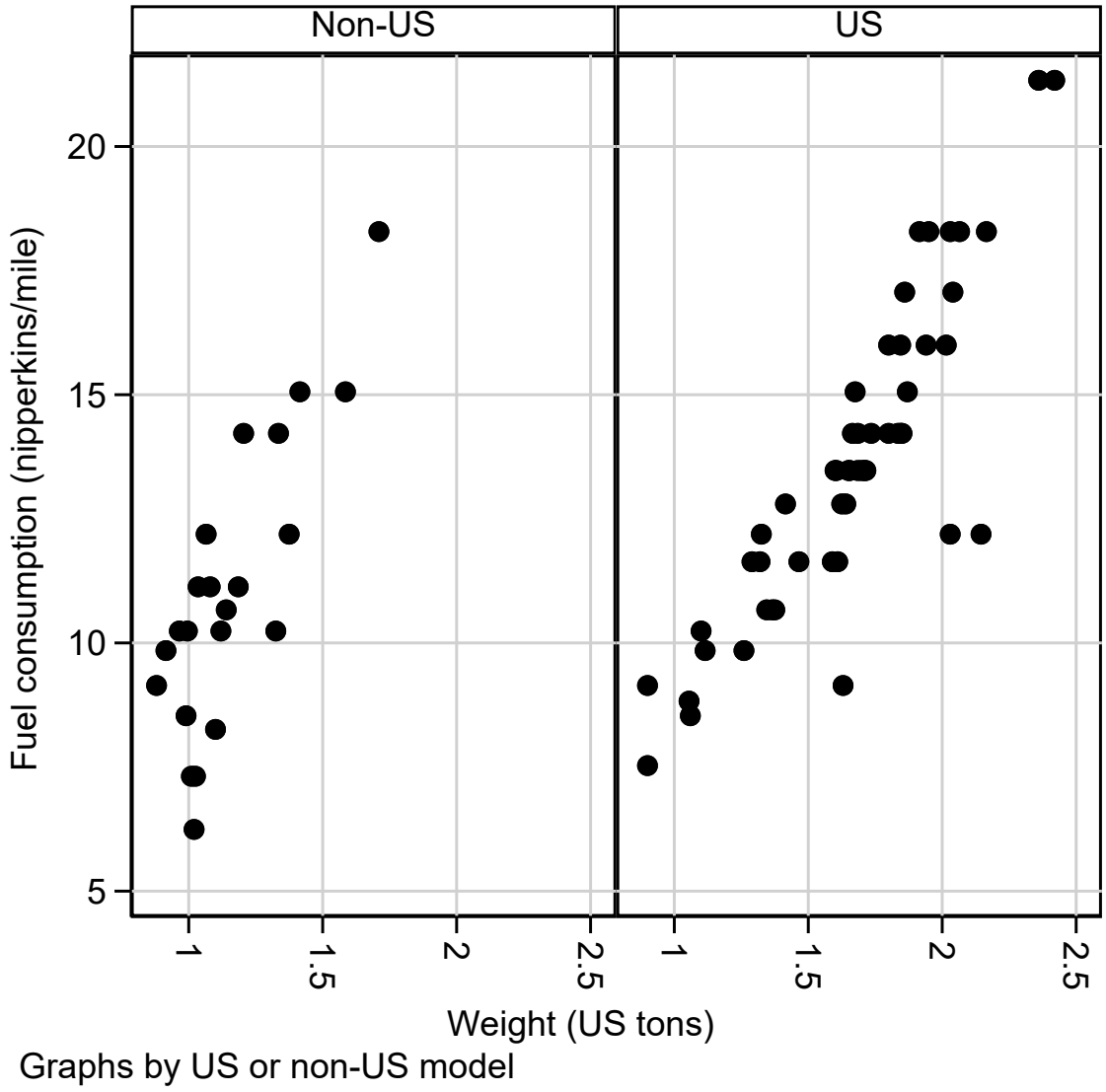
```
. invdesc, dframe(mydesc) lframe(myvallabs);

. desc, fu;

Contains data
  obs:            74
 vars:            13
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
foreign         byte    %8.0g      origin     Car type
make            str17   %-17s                 Make and Model
price           int     %8.0gc                Price
rep78           byte    %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           byte    %8.0g                 Trunk space (cu. ft.)
length          int     %8.0g                 Length (in.)
turn            byte    %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      double  %6.2f                 Gear Ratio
us              byte    %8.0g      us         US or non-US model
tons            double  %10.0g                Weight (US tons)
npm             double  %10.0g                Fuel consumption (nipperkins/mile)
-------------------------------------------------------------------------------
Sorted by:
     Note: Dataset has changed since last saved.
```

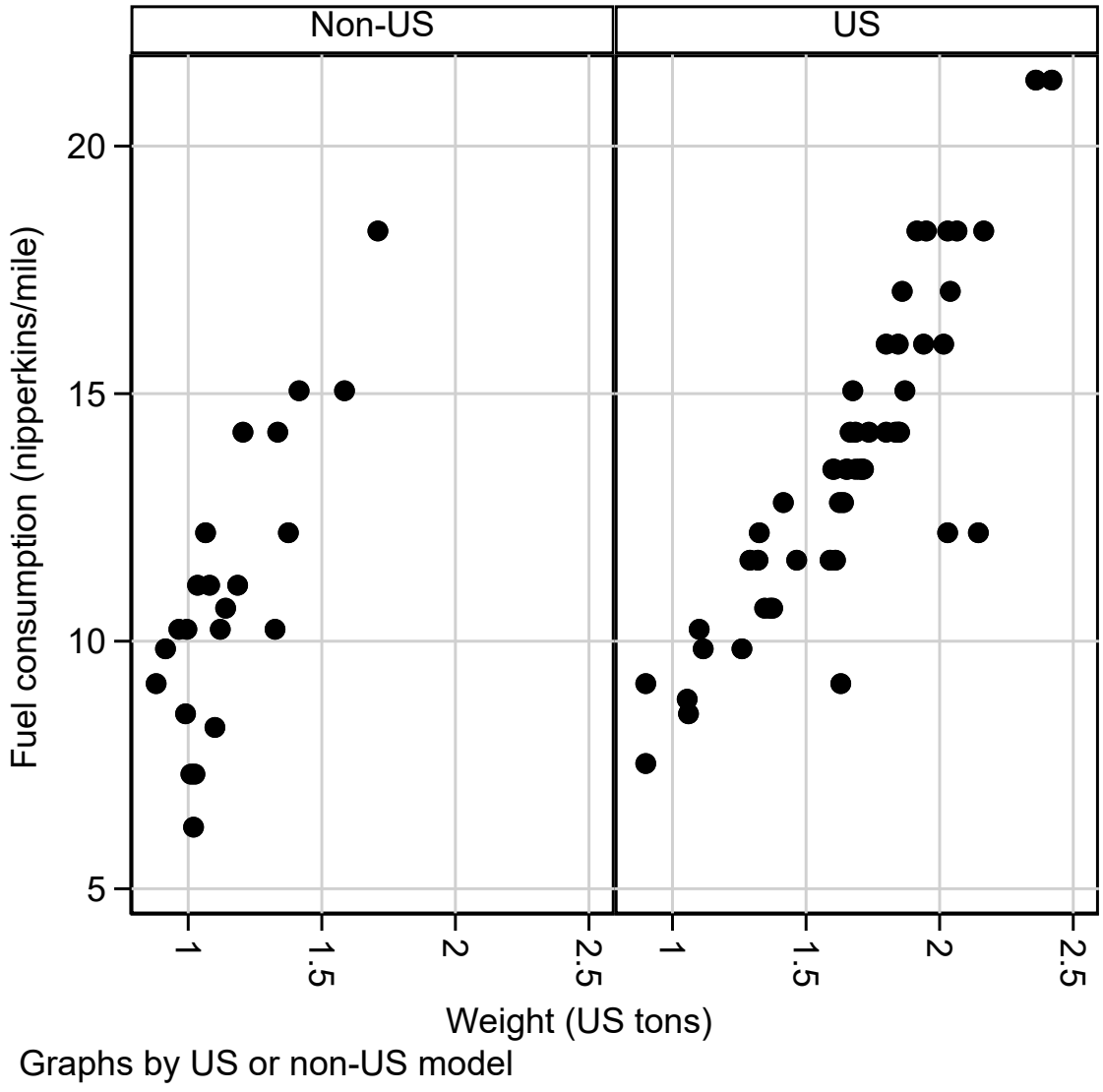
Scatter plot of `npm` against `tons` by `us`

- The vertical axis gives fuel consumption in nipperkins per mile (where a nipperkin is 1/256 of a gallon in the binary system of Imperial/US fluid measures).
- The horizontal axis gives weight in US tons (which are smaller than metric tonnes, which in turn are smaller than Imperial tons).
- And the plots are separated by US origin.



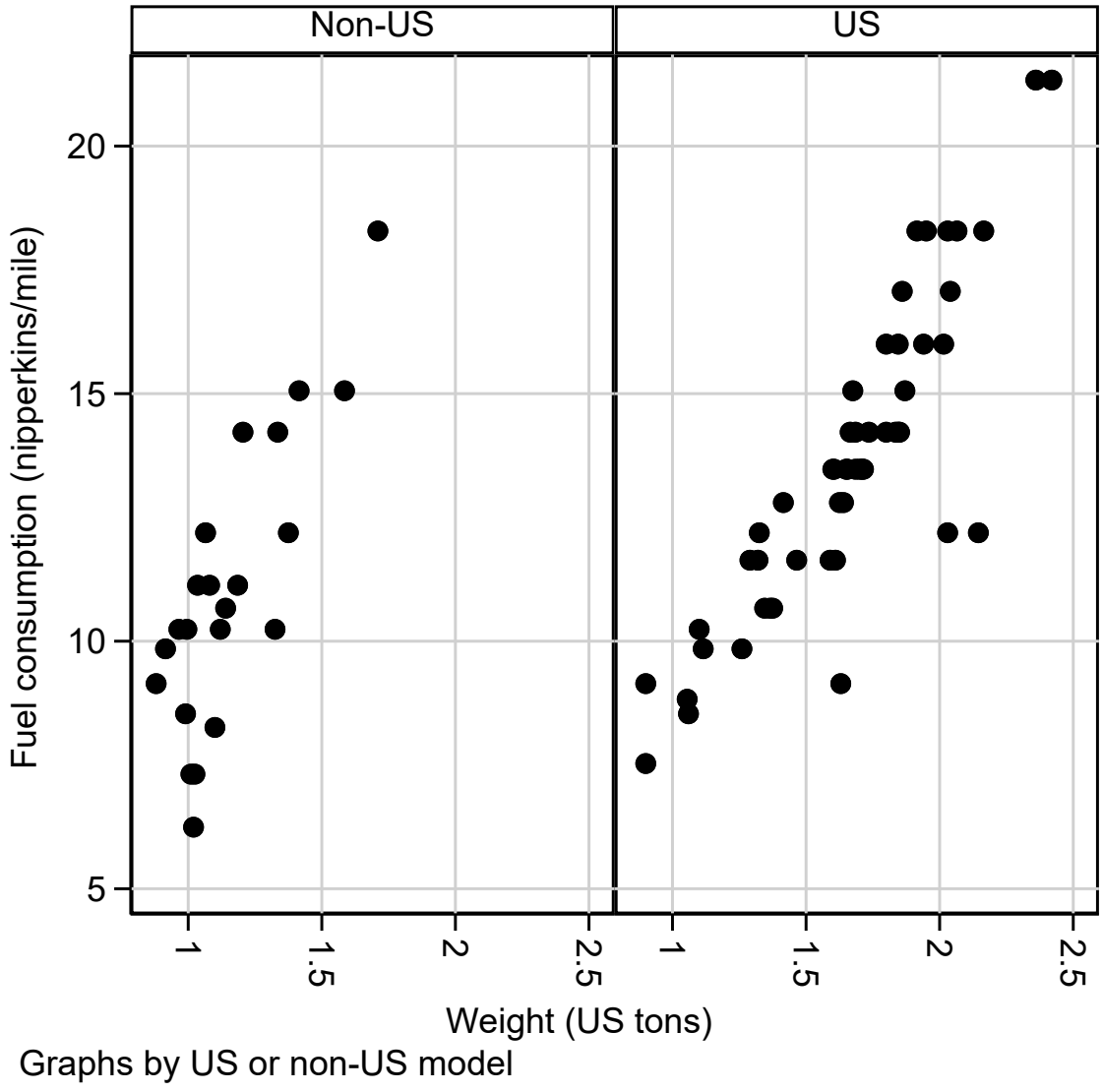
Scatter plot of `npm` against `tons` by `us`

- The vertical axis gives fuel consumption in nipperkins per mile (where a nipperkin is 1/256 of a gallon in the binary system of Imperial/US fluid measures).
- The horizontal axis gives weight in US tons (which are smaller than metric tonnes, which in turn are smaller than Imperial tons).
- And the plots are separated by US origin.



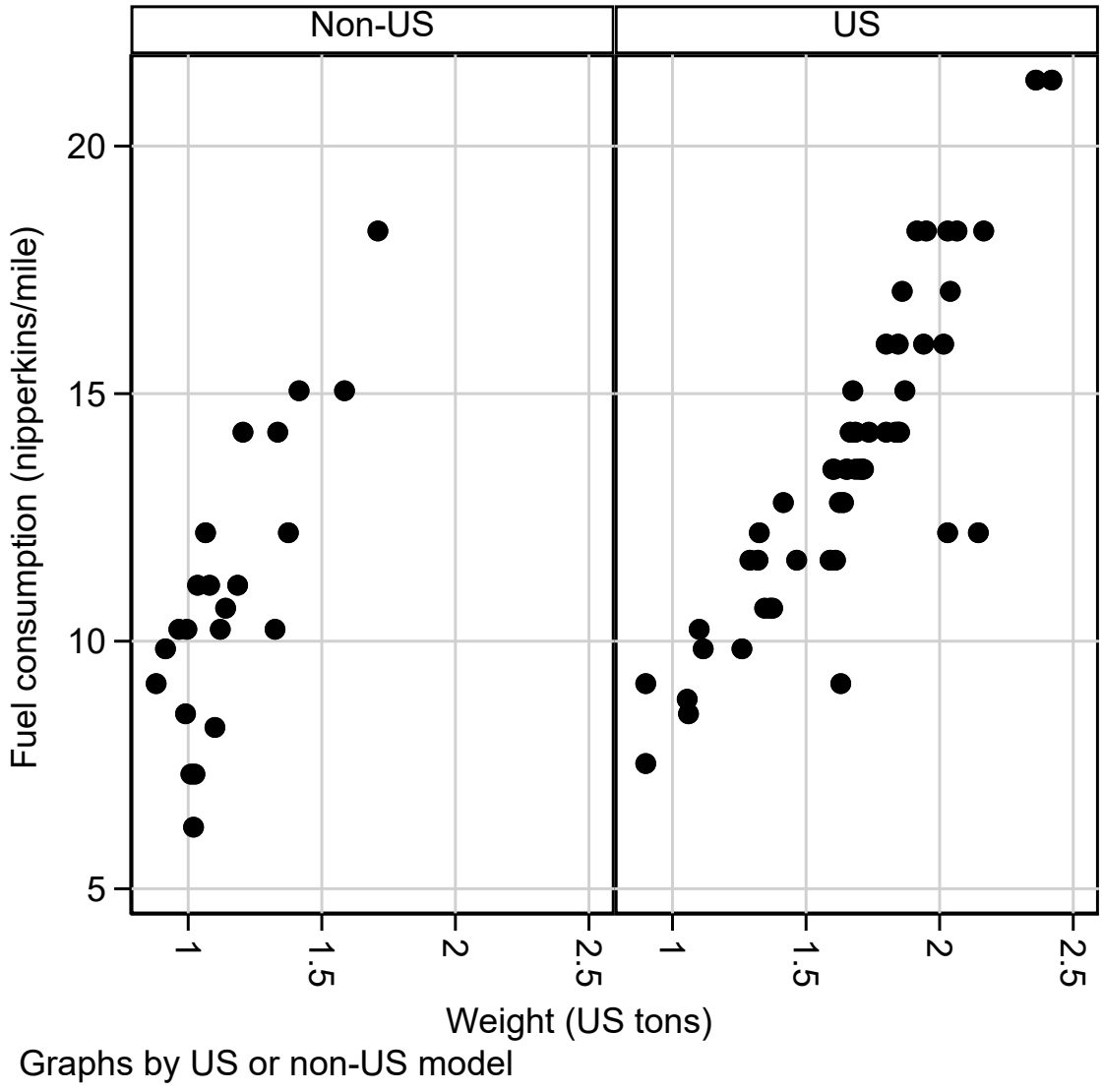
Scatter plot of `npm` against `tons` by `us`

- The vertical axis gives fuel consumption in nipperkins per mile (where a nipperkin is 1/256 of a gallon in the binary system of Imperial/US fluid measures).
- The horizontal axis gives weight in US tons (which are smaller than metric tonnes, which in turn are smaller than Imperial tons).
- And the plots are separated by US origin.



Scatter plot of `npm` against `tons` by `us`

- The vertical axis gives fuel consumption in nipperkins per mile (where a nipperkin is 1/256 of a gallon in the binary system of Imperial/US fluid measures).
- The horizontal axis gives weight in US tons (which are smaller than metric tonnes, which in turn are smaller than Imperial tons).
- And the plots are separated by US origin.



The 3 datasets we eventually created

And, after the example sequence (of which only a few highlights have been shown), here are the 3 datasets we saw earlier. These comprise 2 metadatasets (`mydesc` containing variable descriptives and `myvallabs` containing value labels), and one non–meta–dataset `myxauto` (created using the 2 metadatasets made earlier).

```
. list filename nobs nvar size sortedby dslabel, abbr(32);

     +-------------------------------------------------------------------------------+
     | filename        nobs   nvar   size   sortedby           dslabel              |
     |-------------------------------------------------------------------------------|
  1. | mydesc.dta        23      8   1725   dset order         Variables in all datasets |
  2. | myvallabs.dta      4      3     60   vallabname value   Value labels          |
  3. | myxauto.dta       74     13   4144   foreign make       Extended auto data   |
     +-------------------------------------------------------------------------------+
```

Note that these were generated using definitive information stored in generic spreadsheets, as definitive information always should be. Many variations on this theme are possible. (Most of them will have more than one non–meta–dataset!)

References

[1] Newson RB. Post–`parmest` peripherals: `fvregen`, `invcise`, and `qqvalue`. Presented at the *16th UK Stata User Meeting*, 9–10 September, 2010. Downloadable from the conference website at *http://ideas.repec.org/p/boc/usug10/01.html*.

[2] Newson R. Resultssets, resultsspreadsheets and resultsplots in Stata. Presented at the *4th German Stata User Meeting*, 31 March, 2006. Downloadable from the conference website at *http://ideas.repec.org/p/boc/dsug06/01.html*.

[3] Newson R. From datasets to resultssets in Stata. Presented at the *10th UK Stata User Meeting*, 28–29 June, 2004. Downloadable from the conference website at *https://ideas.repec.org/p/boc/usug04/16.html*.

The presentation, and the example do–file and input tab–delimited worksheets, can be downloaded from the conference website at *http://ideas.repec.org/s/boc/usug20.html*

The packages used can be downloaded from SSC.