Resultssets, resultsspreadsheets and resultsplots in Stata

Roger Newson
Imperial College London, London, UK
r.newson@imperial.ac.uk
http://www.imperial.ac.uk/nhli/r.newson/

Presented at the 4th German Stata Users' Group Meeting on 31 March 2006

This presentation, and the do–files containing the examples, can be downloaded from the conference website at
http://ideas.repec.org/s/boc/dsug06.html

Introduction

Introduction

- Most Stata users make their living producing results in a form accessible to end users.

Introduction

- Most Stata users make their living producing results in a form accessible to end users.
- Most of these end users will not understand the results listed in a Stata log.

Introduction

- Most Stata users make their living producing results in a form accessible to end users.
- Most of these end users will not understand the results listed in a Stata log.
- *However*, they can understand tables and plots.

Introduction

- Most Stata users make their living producing results in a form accessible to end users.
- Most of these end users will not understand the results listed in a Stata log.
- *However*, they can understand tables and plots.
- The tables may be in .pdf, .html, word processor, spreadsheet or paper documents.

Introduction

- Most Stata users make their living producing results in a form accessible to end users.
- Most of these end users will not understand the results listed in a Stata log.
- *However*, they can understand tables and plots.
- The tables may be in .pdf, .html, word processor, spreadsheet or paper documents.
- The plots may be produced using Stata or non–Stata software.

Introduction

- Most Stata users make their living producing results in a form accessible to end users.
- Most of these end users will not understand the results listed in a Stata log.
- *However*, they can understand tables and plots.
- The tables may be in .pdf, .html, word processor, spreadsheet or paper documents.
- The plots may be produced using Stata or non–Stata software.
- This presentation will introduce ways of producing these plots and tables in Stata.

Results (1): Countries of origin of cars in the `auto` data

Results (1): Countries of origin of cars in the `auto` data

We have created a labelled numeric variable `country` in the `auto` data. Here are its frequencies as output to the Stata log (using `tabulate`):

Results (1): Countries of origin of cars in the `auto` data

We have created a labelled numeric variable `country` in the `auto` data. Here are its frequencies as output to the Stata log (using `tabulate`):

```
. tab country, gene(c_)

 Country of |
  origin of |
       firm |      Freq.     Percent        Cum.
------------+-----------------------------------
         US |         52       70.27       70.27
      Japan |         11       14.86       85.14
    Germany |          7        9.46       94.59
     France |          2        2.70       97.30
      Italy |          1        1.35       98.65
     Sweden |          1        1.35      100.00
------------+-----------------------------------
      Total |         74      100.00
```

Results (2): Mean weights of cars from each country

Results (2): Mean weights of cars from each country

We now use `regress` to estimate confidence intervals for the mean weights of cars from each country:

Results (2): Mean weights of cars from each country

We now use `regress` to estimate confidence intervals for the mean weights of cars from each country:

```
. regress weight c_*, noconst nohead
------------------------------------------------------------------------------
      weight |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         c_1 |   3317.115   88.26823    37.58   0.000     3140.979    3493.252
         c_2 |   2248.182   191.9154    11.71   0.000     1865.221    2631.143
         c_3 |   2238.571   240.5786     9.30   0.000     1758.504    2718.639
         c_4 |        2625   450.0814     5.83   0.000     1726.877    3523.123
         c_5 |        2130   636.5112     3.35   0.001     859.8616    3400.138
         c_6 |        3170   636.5112     4.98   0.000     1899.862    4440.138
------------------------------------------------------------------------------
```

Results (2): Mean weights of cars from each country

We now use `regress` to estimate confidence intervals for the mean weights of cars from each country:

```
. regress weight c_*, noconst nohead
------------------------------------------------------------------------------
      weight |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         c_1 |   3317.115   88.26823    37.58   0.000     3140.979    3493.252
         c_2 |   2248.182   191.9154    11.71   0.000     1865.221    2631.143
         c_3 |   2238.571   240.5786     9.30   0.000     1758.504    2718.639
         c_4 |        2625   450.0814     5.83   0.000     1726.877    3523.123
         c_5 |        2130   636.5112     3.35   0.001     859.8616    3400.138
         c_6 |        3170   636.5112     4.98   0.000     1899.862    4440.138
------------------------------------------------------------------------------
```

Very few end users will understand this output!

Resultsset of frequencies and mean weights

Resultsset of frequencies and mean weights

This resultsset is a Stata dataset created using `xcontract`, `parmest`, `descsave` and `factext` (downloadable from SSC). It has one observation per country, and data on frequencies, and on estimates and 95% confidence limits for mean weights (in US pounds), for cars made by firms based in that country.

Resultsset of frequencies and mean weights

This resultsset is a Stata dataset created using `xcontract`, `parmest`, `descsave` and `factext` (downloadable from SSC). It has one observation per country, and data on frequencies, and on estimates and 95% confidence limits for mean weights (in US pounds), for cars made by firms based in that country.

```
. list country _freq estimate min95 max95, clean noobs
```

country	_freq	estimate	min95	max95
US	52	3317.12	3140.98	3493.25
Japan	11	2248.18	1865.22	2631.14
Germany	7	2238.57	1758.50	2718.64
France	2	2625.00	1726.88	3523.12
Italy	1	2130.00	859.86	3400.14
Sweden	1	3170.00	1899.86	4440.14

Resultsset of frequencies and mean weights

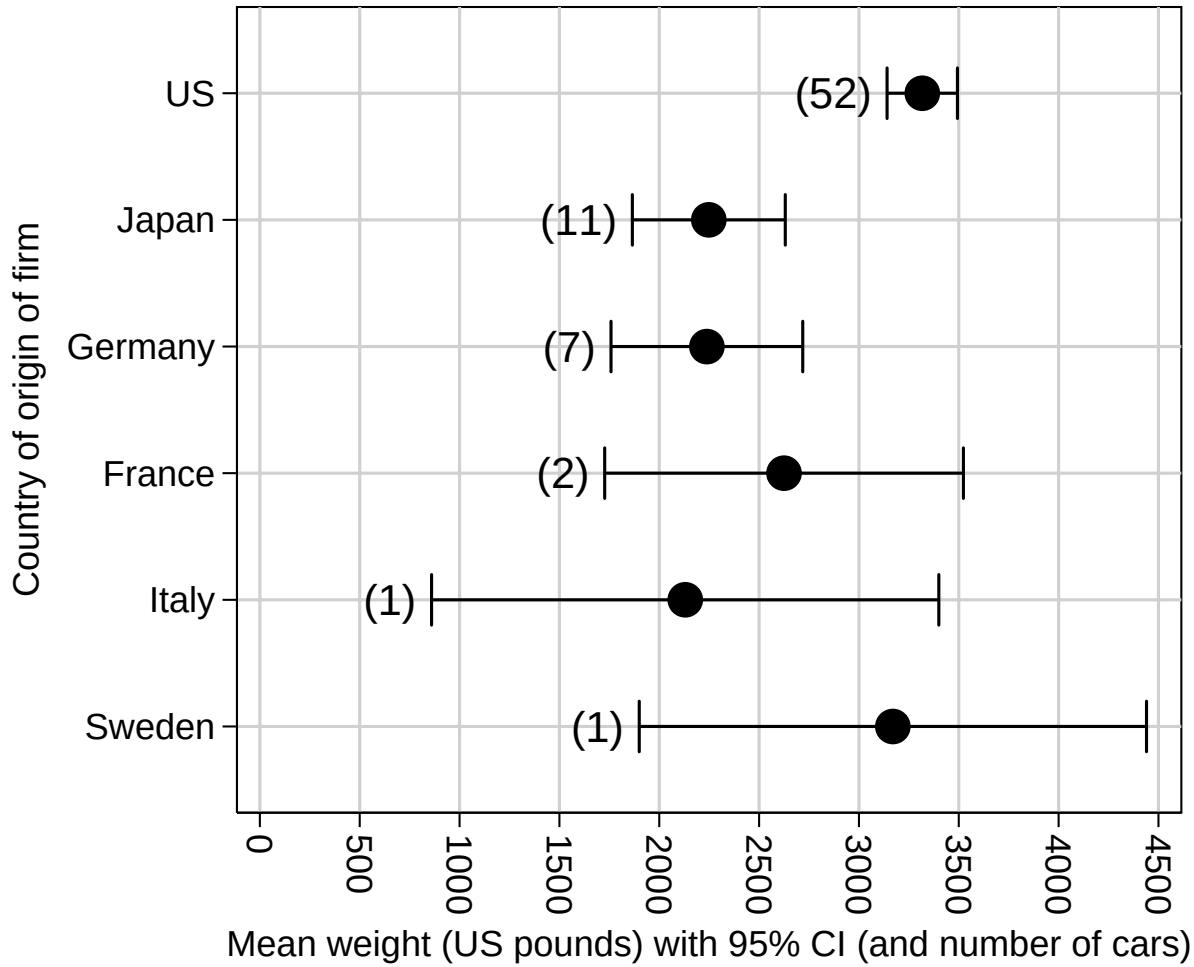
This resultsset is a Stata dataset created using `xcontract`, `parmest`, `descsave` and `factext` (downloadable from SSC). It has one observation per country, and data on frequencies, and on estimates and 95% confidence limits for mean weights (in US pounds), for cars made by firms based in that country.

```
. list country _freq estimate min95 max95, clean noobs
```

country	_freq	estimate	min95	max95
US	52	3317.12	3140.98	3493.25
Japan	11	2248.18	1865.22	2631.14
Germany	7	2238.57	1758.50	2718.64
France	2	2625.00	1726.88	3523.12
Italy	1	2130.00	859.86	3400.14
Sweden	1	3170.00	1899.86	4440.14

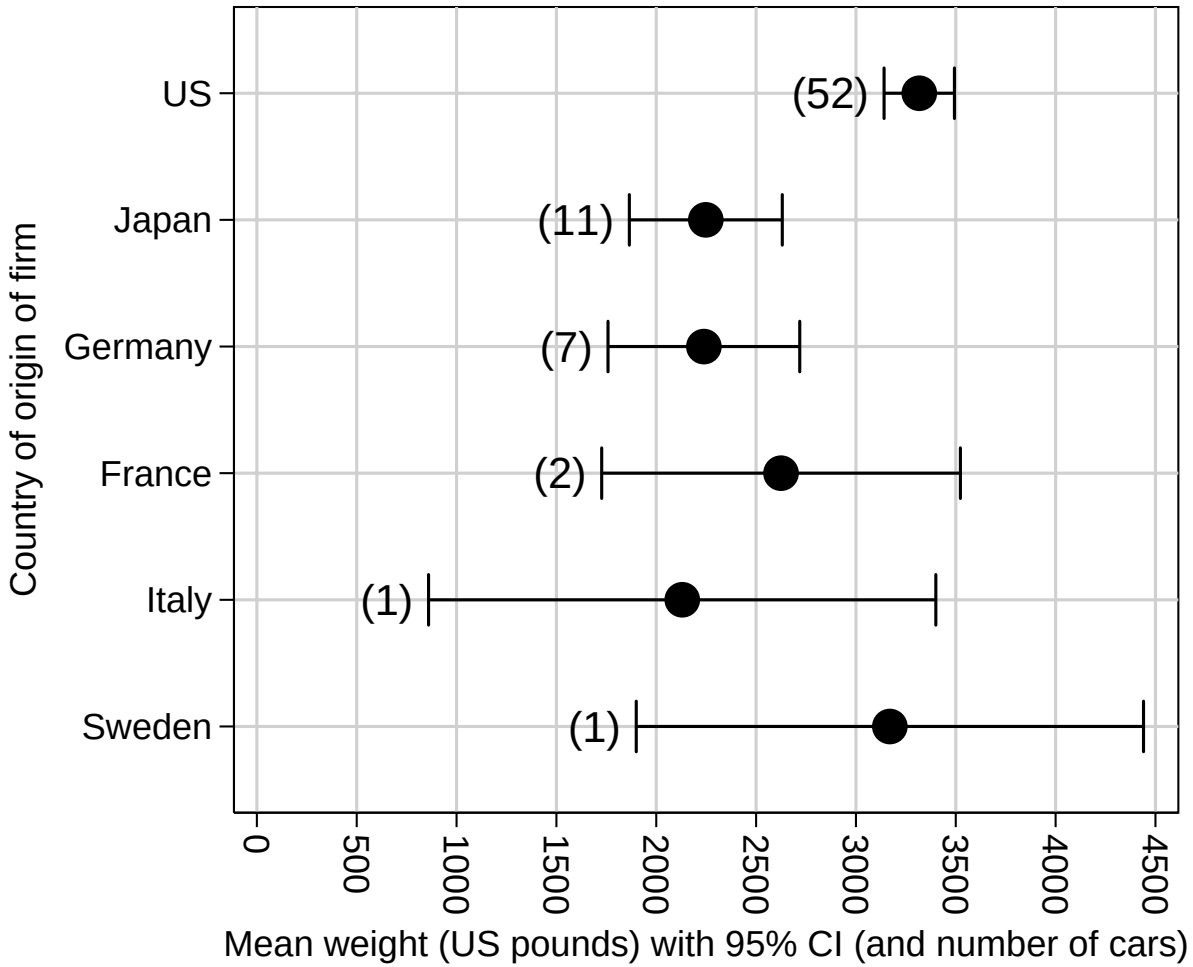
This is easier to understand than the previous output. *However* ...

A resultsplot of the resultsset



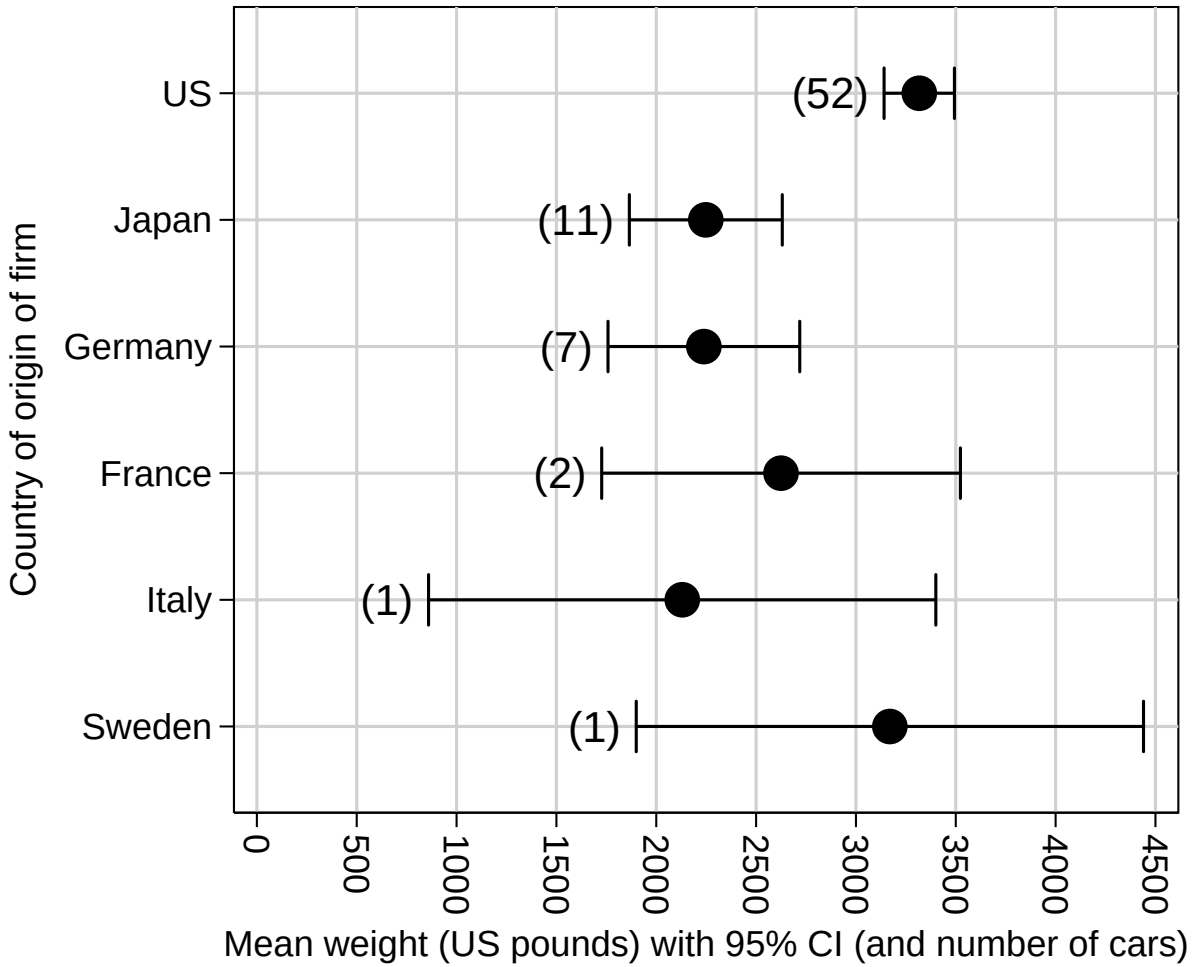
A resultsplot of the resultsset

- ... this plot was produced from the resultsset, using `eclplot` (downloadable from SSC).



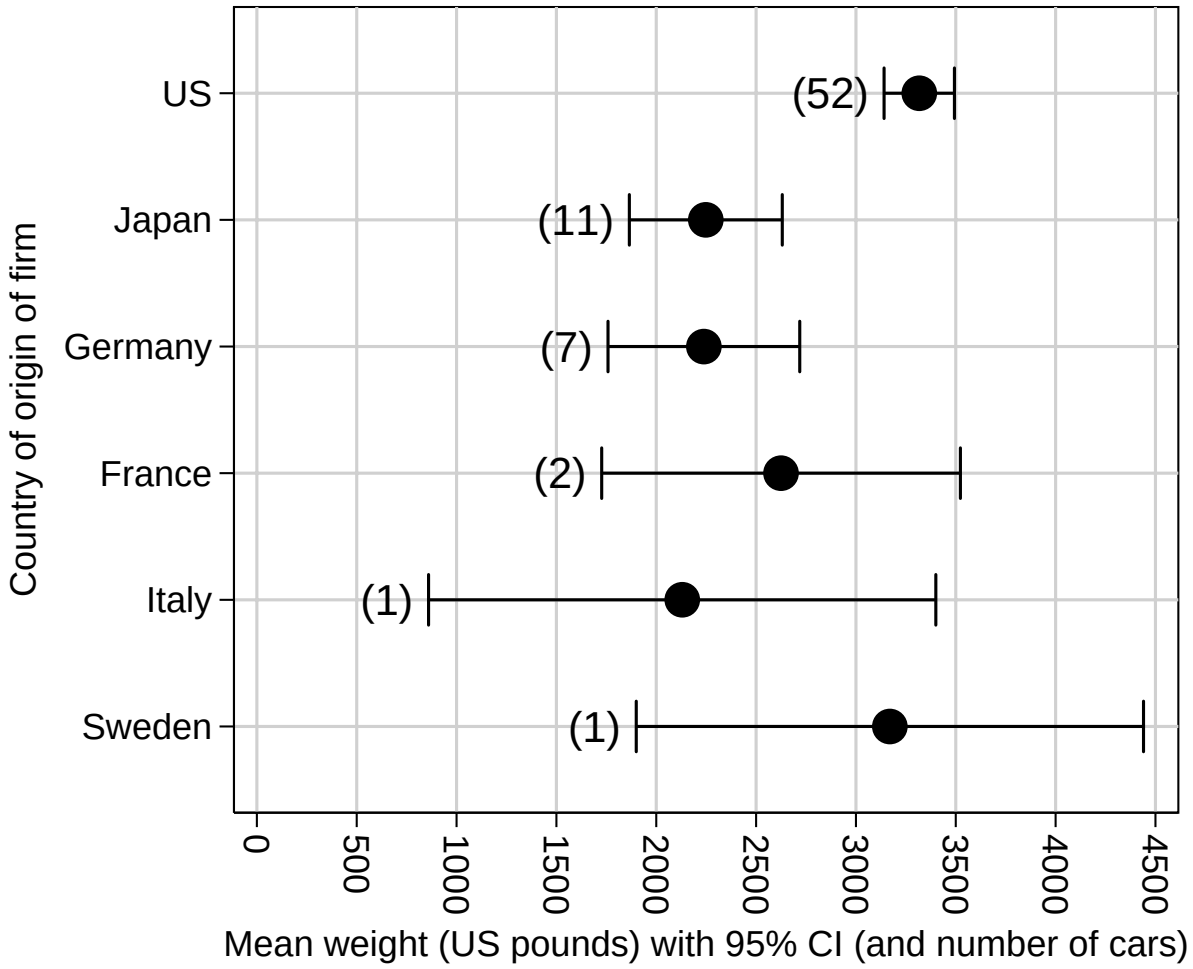
A resultsplot of the resultsset

- ... this plot was produced from the resultsset, using `eclplot` (downloadable from SSC).
- `eclplot` requires an input dataset with one observation per confidence interval and data on estimates and confidence limits.



A resultsplot of the resultsset

- ... this plot was produced from the resultsset, using `eclplot` (downloadable from SSC).
- `eclplot` requires an input dataset with one observation per confidence interval and data on estimates and confidence limits.
- The frequencies were added using the `plot()` option of `eclplot`.



Resultsspreadsheet of frequencies and mean weights

Resultsspreadsheet of frequencies and mean weights

We convert the confidence limits to string, adding commas and parentheses, using the SSC package `sdecode`. Then we use the SSC package `listtex` to output the resultsset to a resultsspreadsheet in the LaTeX `tabular` row style:

Resultsspreadsheet of frequencies and mean weights

We convert the confidence limits to string, adding commas and parentheses, using the SSC package `sdecode`. Then we use the SSC package `listtex` to output the resultsset to a resultsspreadsheet in the LaTeX `tabular` row style:

```
. listtex country _freq estimate min95 max95, rstyle(tabular) type ///
>   head( ///
>     "\begin{tabular}{rrrrr}" ///
>     "\textit{Country}&\textit{N}&\textit{Mean}&\textit{(95\%}&\textit{CI)}\\" ///
>   ) ///
>   foot("\end{tabular}")
\begin{tabular}{rrrrr}
\textit{Country}&\textit{N}&\textit{Mean}&\textit{(95\%}&\textit{CI)}\\
US&52&3317.12&(3140.98,&3493.25)\\
Japan&11&2248.18&(1865.22,&2631.14)\\
Germany&7&2238.57&(1758.50,&2718.64)\\
France&2&2625.00&(1726.88,&3523.12)\\
Italy&1&2130.00&(859.86,&3400.14)\\
Sweden&1&3170.00&(1899.86,&4440.14)\\
\end{tabular}
```

Resultsspreadsheet of frequencies and mean weights

We convert the confidence limits to string, adding commas and parentheses, using the SSC package `sdecode`. Then we use the SSC package `listtex` to output the resultsset to a resultsspreadsheet in the LaTeX `tabular` row style:

```
. listtex country _freq estimate min95 max95, rstyle(tabular) type ///
>   head( ///
>     "\begin{tabular}{rrrrr}" ///
>     "\textit{Country}&\textit{N}&\textit{Mean}&\textit{(95\%}&\textit{CI)}\\" ///
>   ) ///
>   foot("\end{tabular}")
\begin{tabular}{rrrrr}
\textit{Country}&\textit{N}&\textit{Mean}&\textit{(95\%}&\textit{CI)}\\
US&52&3317.12&(3140.98,&3493.25)\\
Japan&11&2248.18&(1865.22,&2631.14)\\
Germany&7&2238.57&(1758.50,&2718.64)\\
France&2&2625.00&(1726.88,&3523.12)\\
Italy&1&2130.00&(859.86,&3400.14)\\
Sweden&1&3170.00&(1899.86,&4440.14)\\
\end{tabular}
```

This is not very easy to understand. *However* ...

Frequencies and mean weights (in US pounds) for countries in the `auto` data

Frequencies and mean weights (in US pounds) for countries in the `auto` data

. . . when the resultsspreadsheet was cut and pasted into the LaTeX version of this presentation, the following table was produced:

Frequencies and mean weights (in US pounds) for countries in the `auto` data

. . . when the resultsspreadsheet was cut and pasted into the LaTeX version of this presentation, the following table was produced:

Country	*N*	*Mean*	*(95%*	*CI)*
US	52	3317.12	(3140.98,	3493.25)
Japan	11	2248.18	(1865.22,	2631.14)
Germany	7	2238.57	(1758.50,	2718.64)
France	2	2625.00	(1726.88,	3523.12)
Italy	1	2130.00	(859.86,	3400.14)
Sweden	1	3170.00	(1899.86,	4440.14)

Frequencies and mean weights (in US pounds) for countries in the `auto` data

. . . when the resultsspreadsheet was cut and pasted into the LaTeX version of this presentation, the following table was produced:

Country	*N*	*Mean*	*(95%*	*CI)*
US	52	3317.12	(3140.98,	3493.25)
Japan	11	2248.18	(1865.22,	2631.14)
Germany	7	2238.57	(1758.50,	2718.64)
France	2	2625.00	(1726.88,	3523.12)
Italy	1	2130.00	(859.86,	3400.14)
Sweden	1	3170.00	(1899.86,	4440.14)

`listtex` (and also another SSC package `estout`) can also produce resultsspreadsheets in other generic text formats, such as plain TeX, HTML, or tab-separated (for easy conversion to Microsoft Excel spreadsheets or Word tables).

Definitions

Definitions

- A **resultsset** is a Stata dataset produced (directly or indirectly) as output by a Stata command. (The term was coined by Nicholas J. Cox of Durham University, UK.)

Definitions

- A **resultsset** is a Stata dataset produced (directly or indirectly) as output by a Stata command. (The term was coined by Nicholas J. Cox of Durham University, UK.)
- A **resultsspreadsheet** is a generic text format spreadsheet produced (directly or indirectly) as output by a Stata command.

Definitions

- A **resultsset** is a Stata dataset produced (directly or indirectly) as output by a Stata command. (The term was coined by Nicholas J. Cox of Durham University, UK.)
- A **resultsspreadsheet** is a generic text format spreadsheet produced (directly or indirectly) as output by a Stata command.
- A **resultsplot** is a plot, with plotted points corresponding to the observations of a resultsset (or to the rows of a resultsspreadsheet).

Resultssets/spreadsheets/plots have one row per *result*

Resultssets/spreadsheets/plots have one row per *result*

- A { dataset / spreadsheet / plot } has one { observation / row / plotted point } per *thing*, and data on *attributes_of_things*.

Resultssets/spreadsheets/plots have one row per *result*

- A { dataset / spreadsheet / plot } has one { observation / row / plotted point } per *thing*, and data on *attributes_of_things*.
- For instance, the `auto` dataset has one observation per car model, and data on car attributes.

Resultssets/spreadsheets/plots have one row per *result*

- A { dataset / spreadsheet / plot } has one { observation / row / plotted point } per *thing*, and data on *attributes_of_things*.
- For instance, the `auto` dataset has one observation per car model, and data on car attributes.
- Resultssets, resultsspreadsheets and resultsplots are all normally produced (directly or indirectly) using a Stata dataset as input.

Resultssets/spreadsheets/plots have one row per *result*

- A { dataset / spreadsheet / plot } has one { observation / row / plotted point } per *thing*, and data on *attributes_of_things*.
- For instance, the `auto` dataset has one observation per car model, and data on car attributes.
- Resultssets, resultsspreadsheets and resultsplots are all normally produced (directly or indirectly) using a Stata dataset as input.
- *However*, the { observations / rows / plotted points } of a { resultsset / resultsspreadsheet / resultsplot } do *not* usually correspond one–for–one to the observations of the original dataset.

Resultssets/spreadsheets/plots have one row per *result*

- A { dataset / spreadsheet / plot } has one { observation / row / plotted point } per *thing*, and data on *attributes_of_things*.
- For instance, the `auto` dataset has one observation per car model, and data on car attributes.
- Resultssets, resultsspreadsheets and resultsplots are all normally produced (directly or indirectly) using a Stata dataset as input.
- *However*, the { observations / rows / plotted points } of a { resultsset / resultsspreadsheet / resultsplot } do *not* usually correspond one–for–one to the observations of the original dataset.
- For instance, the resultsset in the previous example has one observation per *country*, and data on frequencies and confidence intervals.

Resultssets

Resultssets

- The creation and use of resultssets are discussed at length in Newson (2003) and Newson (2004).

Resultssets

- The creation and use of resultssets are discussed at length in Newson (2003) and Newson (2004).
- Programs that create resultssets include the official Stata programs `statsby`, `collapse` and `contract`, and the SSC programs `parmby`, `parmest`, `metaparm`, `xcollapse` and `xcontract`.

Resultssets

- The creation and use of resultssets are discussed at length in Newson (2003) and Newson (2004).
- Programs that create resultssets include the official Stata programs `statsby`, `collapse` and `contract`, and the SSC programs `parmby`, `parmest`, `metaparm`, `xcollapse` and `xcontract`.
- *However*, resultssets can also be created from some types of resultsspreadsheets, using `insheet`.

Resultssets

- The creation and use of resultssets are discussed at length in Newson (2003) and Newson (2004).
- Programs that create resultssets include the official Stata programs `statsby`, `collapse` and `contract`, and the SSC programs `parmby`, `parmest`, `metaparm`, `xcollapse` and `xcontract`.
- *However*, resultssets can also be created from some types of resultsspreadsheets, using `insheet`.
- A resultsset may be listed to the Stata log and/or saved to a disk file and/or written to the memory, replacing any existing dataset.

Resultssets

- The creation and use of resultssets are discussed at length in Newson (2003) and Newson (2004).
- Programs that create resultssets include the official Stata programs `statsby`, `collapse` and `contract`, and the SSC programs `parmby`, `parmest`, `metaparm`, `xcollapse` and `xcontract`.
- *However*, resultssets can also be created from some types of resultsspreadsheets, using `insheet`.
- A resultsset may be listed to the Stata log and/or saved to a disk file and/or written to the memory, replacing any existing dataset.
- *However*, resultssets exist mainly as a means of creating resultsplots and/or resultsspreadsheets.

A resultsset created by the SSC package `xcontract`

A resultsset created by the SSC package `xcontract`

In the `auto` data with the added variable `country`, we use `xcontract` to create a resultsset with one observation per country, and data on frequencies and percentages, which is listed to the Stata log and saved to a temporary file:

A resultsset created by the SSC package `xcontract`

In the `auto` data with the added variable `country`, we use `xcontract` to create a resultsset with one observation per country, and data on frequencies and percentages, which is listed to the Stata log and saved to a temporary file:

```
. tempfile tf1

. xcontract country, saving(‘tf1’) list(, clean noobs)

Listing of results:

    country   _freq   _percent
         US      52      70.27
      Japan      11      14.86
    Germany       7       9.46
     France       2       2.70
      Italy       1       1.35
     Sweden       1       1.35
file C:\DOCUME~1\rnewson\LOCALS~1\Temp\ST_000000m1.tmp saved
```

The SSC package `parmest`

The SSC package `parmest`

The `parmest` package is used with estimation commands such as `regress`. It creates a resultsset with one observation per model parameter, and data on estimates, confidence intervals and p–values. We can start by running the regression model as before:

The SSC package `parmest`

The `parmest` package is used with estimation commands such as `regress`. It creates a resultsset with one observation per model parameter, and data on estimates, confidence intervals and p–values. We can start by running the regression model as before:

```
. regress weight c_*, noconst nohead
------------------------------------------------------------------------------
      weight |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         c_1 |   3317.115   88.26823    37.58   0.000     3140.979    3493.252
         c_2 |   2248.182   191.9154    11.71   0.000     1865.221    2631.143
         c_3 |   2238.571   240.5786     9.30   0.000     1758.504    2718.639
         c_4 |        2625   450.0814     5.83   0.000     1726.877    3523.123
         c_5 |        2130   636.5112     3.35   0.001     859.8616    3400.138
         c_6 |        3170   636.5112     4.98   0.000     1899.862    4440.138
------------------------------------------------------------------------------
```

A resultsset created by the SSC package `parmest`

A resultsset created by the SSC package `parmest`

We then use `parmest` to create a resultsset with one observation per country, and data on mean weights and confidence limits. This is listed to the Stata log, and also written to the memory, replacing the existing data:

A resultsset created by the SSC package `parmest`

We then use `parmest` to create a resultsset with one observation per country, and data on mean weights and confidence limits. This is listed to the Stata log, and also written to the memory, replacing the existing data:

```
. parmest, label norestore  format(estimate min95 max95 %8.2f) ///
>   list(parm label estimate min95 max95, clean noobs)

    parm                  label   estimate     min95     max95
     c_1   country==     1.0000    3317.12   3140.98   3493.25
     c_2   country==     2.0000    2248.18   1865.22   2631.14
     c_3   country==     3.0000    2238.57   1758.50   2718.64
     c_4   country==     4.0000    2625.00   1726.88   3523.12
     c_5   country==     5.0000    2130.00    859.86   3400.14
     c_6   country==     6.0000    3170.00   1899.86   4440.14
```

A resultsset created by the SSC package `parmest`

We then use `parmest` to create a resultsset with one observation per country, and data on mean weights and confidence limits. This is listed to the Stata log, and also written to the memory, replacing the existing data:

```
. parmest, label norestore  format(estimate min95 max95 %8.2f) ///
>   list(parm label estimate min95 max95, clean noobs)
```

parm	label	estimate	min95	max95
c_1	country==	1.0000	3317.12	3140.98	3493.25
c_2	country==	2.0000	2248.18	1865.22	2631.14
c_3	country==	3.0000	2238.57	1758.50	2718.64
c_4	country==	4.0000	2625.00	1726.88	3523.12
c_5	country==	5.0000	2130.00	859.86	3400.14
c_6	country==	6.0000	3170.00	1899.86	4440.14

Note the variable `label`, which contains, for each observation, the variable label of the X–variable of the parameter for that observation. The X–variables here are indicators (dummies) for values of `country` (from 1 to 6).

Resultssets can inherit variables from the original dataset

Resultssets can inherit variables from the original dataset

- Instead of the `label` variable, we might prefer to have the variable `country`, complete with value labels.

Resultssets can inherit variables from the original dataset

- Instead of the `label` variable, we might prefer to have the variable `country`, complete with value labels.
- Many resultsset–generating programs (eg `statsby` and `parmby`) have a `by()` option, allowing the resultsset to inherit by–variables from the dataset.

Resultssets can inherit variables from the original dataset

- Instead of the `label` variable, we might prefer to have the variable `country`, complete with value labels.
- Many resultsset–generating programs (eg `statsby` and `parmby`) have a `by()` option, allowing the resultsset to inherit by–variables from the dataset.
- *However*, `country` is *not* a by–variable, but a categorical predictor variable in the regression model.

Resultssets can inherit variables from the original dataset

- Instead of the `label` variable, we might prefer to have the variable `country`, complete with value labels.
- Many resultsset–generating programs (eg `statsby` and `parmby`) have a `by()` option, allowing the resultsset to inherit by–variables from the dataset.
- *However*, `country` is *not* a by–variable, but a categorical predictor variable in the regression model.
- And *many* users want to plot confidence intervals against predictor variables!

Restoring variables in resultssets using `descsave` and `factext`

Restoring variables in resultssets using `descsave` and `factext`

- The SSC package `descsave` is an extended version of `describe`, which can describe a list of variables and write a do–file.

Restoring variables in resultssets using `descsave` and `factext`

- The SSC package `descsave` is an extended version of `describe`, which can describe a list of variables and write a do–file.
- This do–file, if run in another dataset, will reconstruct the storage types, display formats, value labels and variable labels for any variables of the same names and modes (numeric or string) which may exist in that dataset.

Restoring variables in resultssets using `descsave` and `factext`

- The SSC package `descsave` is an extended version of `describe`, which can describe a list of variables and write a do–file.
- This do–file, if run in another dataset, will reconstruct the storage types, display formats, value labels and variable labels for any variables of the same names and modes (numeric or string) which may exist in that dataset.
- The SSC package `factext` can reconstruct categorical factors such as `country` from the `label` variable in a `parmest` resultsset.

Restoring variables in resultssets using `descsave` and `factext`

- The SSC package `descsave` is an extended version of `describe`, which can describe a list of variables and write a do–file.
- This do–file, if run in another dataset, will reconstruct the storage types, display formats, value labels and variable labels for any variables of the same names and modes (numeric or string) which may exist in that dataset.
- The SSC package `factext` can reconstruct categorical factors such as `country` from the `label` variable in a `parmest` resultsset.
- It can then run a do–file created by `descsave` to reconstruct the storage types, display formats, value labels and variable labels.

Running `descsave` in the extended `auto` data

Running `descsave` in the extended `auto` data

In the extended `auto` dataset, we run `descsave` on the added variable `country`, writing the do–file to a temporary file:

Running `descsave` in the extended `auto` data

In the extended `auto` dataset, we run `descsave` on the added variable `country`, writing the do–file to a temporary file:

```
. tempfile df1

. descsave country, do(‘df1’)

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
country         byte   %8.0g       country    Country of origin of firm
```

Running `descsave` in the extended `auto` data

In the extended `auto` dataset, we run `descsave` on the added variable `country`, writing the do–file to a temporary file:

```
. tempfile df1

. descsave country, do(‘df1’)

              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
country         byte   %8.0g       country    Country of origin of firm
```

`descsave` describes the storage type, display format, value label and variable label of `country`, and creates a temporary do–file ‘`df1`’ to reconstruct these attributes in another dataset.

The do–file created by `descsave`

The do–file created by `descsave`

We `type` the temporary do–file:

The do–file created by `descsave`

We `type` the temporary do–file:

```
. type ‘df1’
cap la de country 1 ‘"US"’, modify
cap la de country 2 ‘"Japan"’, modify
cap la de country 3 ‘"Germany"’, modify
cap la de country 4 ‘"France"’, modify
cap la de country 5 ‘"Italy"’, modify
cap la de country 6 ‘"Sweden"’, modify
cap recast byte country
cap form country %8.0g
cap la val country country
cap la var country ‘"Country of origin of firm"’
```

The do–file created by `descsave`

We `type` the temporary do–file:

```
. type ‘df1’
cap la de country 1 ‘"US"’, modify
cap la de country 2 ‘"Japan"’, modify
cap la de country 3 ‘"Germany"’, modify
cap la de country 4 ‘"France"’, modify
cap la de country 5 ‘"Italy"’, modify
cap la de country 6 ‘"Sweden"’, modify
cap recast byte country
cap form country %8.0g
cap la val country country
cap la var country ‘"Country of origin of firm"’
```

If this do–file is run in another dataset, and a numeric variable `country` exists in that dataset, then that numeric variable `country` will have the storage type, display format, value labels and variable label of the variable `country` in the `auto` data.

Reconstructing `country` in the `parmest` resultsset

Reconstructing `country` in the `parmest` resultsset

In the `parmest` resultsset, we run `factext` to reconstruct `country`, from the `label` variable, and list the resultsset:

Reconstructing `country` in the `parmest` resultsset

In the `parmest` resultsset, we run `factext` to reconstruct `country`, from the `label` variable, and list the resultsset:

```
. factext, do(‘df1’)

. list parm label country estimate min95 max95, clean noobs
```

parm		label	country	estimate	min95	max95
c_1	country==	1.0000	US	3317.12	3140.98	3493.25
c_2	country==	2.0000	Japan	2248.18	1865.22	2631.14
c_3	country==	3.0000	Germany	2238.57	1758.50	2718.64
c_4	country==	4.0000	France	2625.00	1726.88	3523.12
c_5	country==	5.0000	Italy	2130.00	859.86	3400.14
c_6	country==	6.0000	Sweden	3170.00	1899.86	4440.14

Reconstructing `country` in the `parmest` resultsset

In the `parmest` resultsset, we run `factext` to reconstruct `country`, from the `label` variable, and list the resultsset:

```
. factext, do('df1')

. list parm label country estimate min95 max95, clean noobs
```

parm		label	country	estimate	min95	max95
c_1	country==	1.0000	US	3317.12	3140.98	3493.25
c_2	country==	2.0000	Japan	2248.18	1865.22	2631.14
c_3	country==	3.0000	Germany	2238.57	1758.50	2718.64
c_4	country==	4.0000	France	2625.00	1726.88	3523.12
c_5	country==	5.0000	Italy	2130.00	859.86	3400.14
c_6	country==	6.0000	Sweden	3170.00	1899.86	4440.14

The variable `country` has been reconstructed from the variable `label`, with its variable labels. So we can now identify the countries of the confidence intervals.

Match–merging resultssets

Match–merging resultssets

We now merge the `parmest` resultsset with the `xcontract` resultsset that we saw earlier, and add the frequencies to the confidence intervals:

Match–merging resultssets

We now merge the `parmest` resultsset with the `xcontract` resultsset that we saw earlier, and add the frequencies to the confidence intervals:

```
. sort country

. merge country using ‘tf1’
(label country already defined)

. sort country

. list country _freq estimate min95 max95, clean noobs

    country   _freq   estimate      min95      max95
         US      52    3317.12    3140.98    3493.25
      Japan      11    2248.18    1865.22    2631.14
    Germany       7    2238.57    1758.50    2718.64
     France       2    2625.00    1726.88    3523.12
      Italy       1    2130.00     859.86    3400.14
     Sweden       1    3170.00    1899.86    4440.14
```

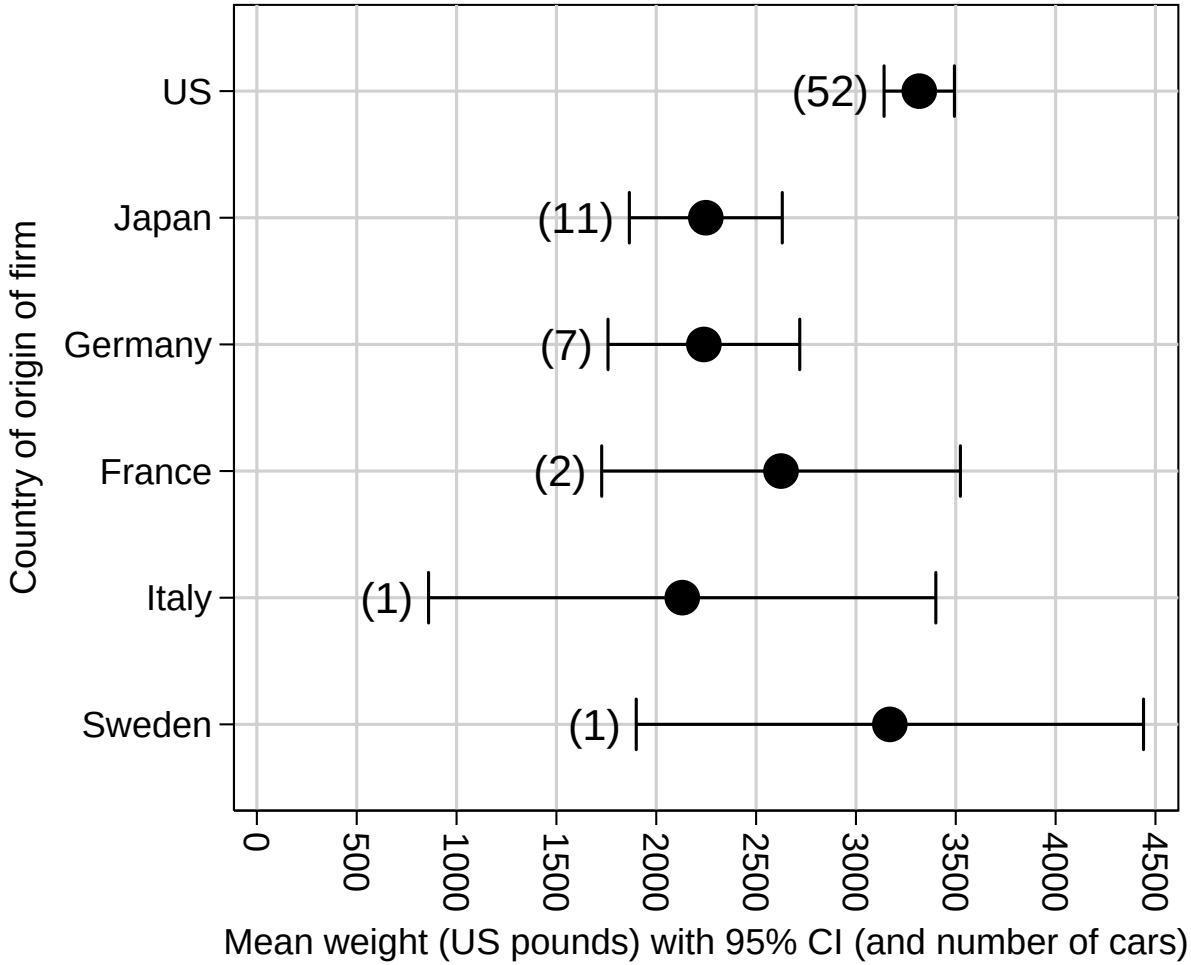
Match–merging resultssets

We now merge the `parmest` resultsset with the `xcontract` resultsset that we saw earlier, and add the frequencies to the confidence intervals:

```
. sort country

. merge country using ‘tf1’
(label country already defined)

. sort country

. list country _freq estimate min95 max95, clean noobs

    country   _freq   estimate      min95      max95
         US      52    3317.12    3140.98    3493.25
      Japan      11    2248.18    1865.22    2631.14
    Germany       7    2238.57    1758.50    2718.64
     France       2    2625.00    1726.88    3523.12
      Italy       1    2130.00     859.86    3400.14
     Sweden       1    3170.00    1899.86    4440.14
```

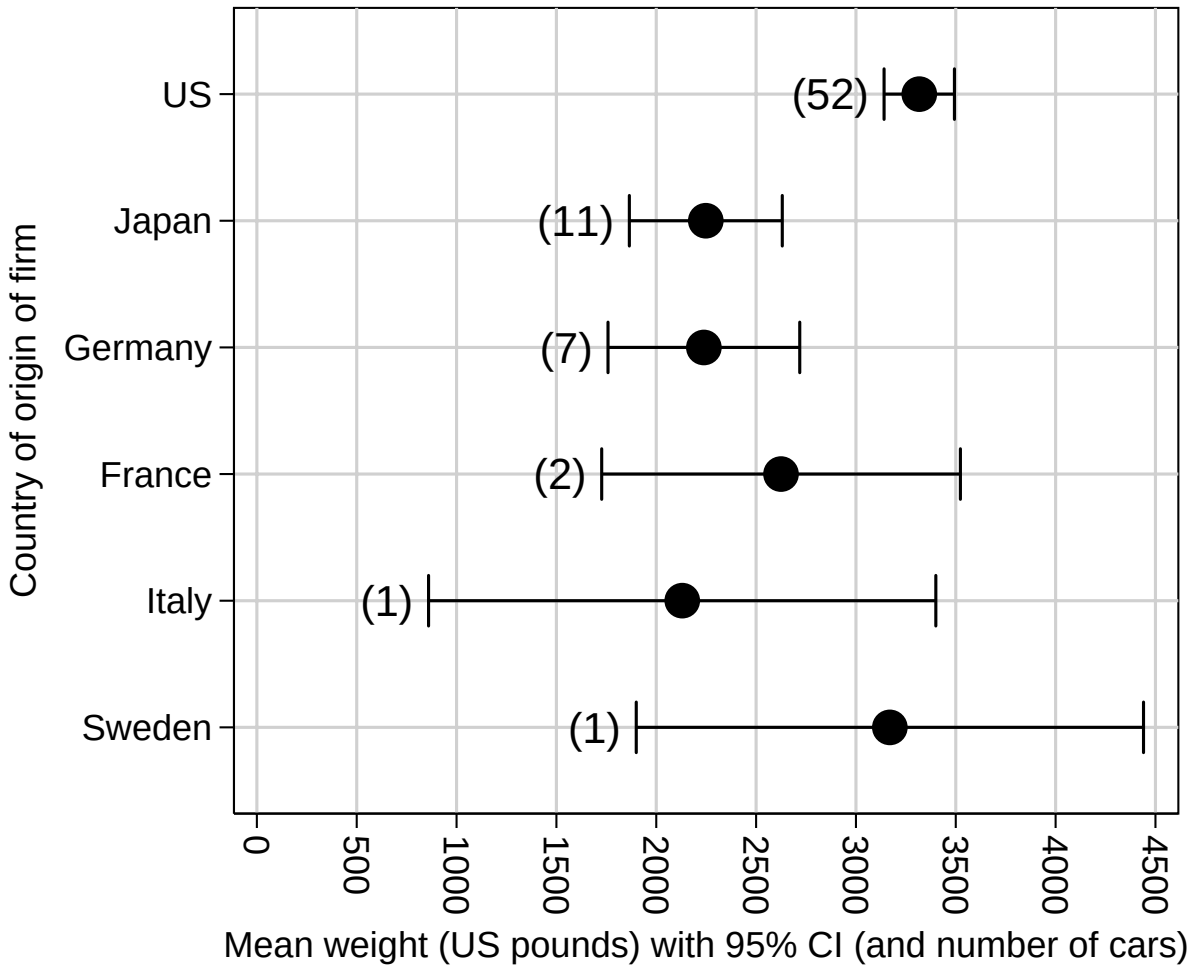
We now have the resultsset which created the resultsplot that we saw earlier.

The power of resultssets



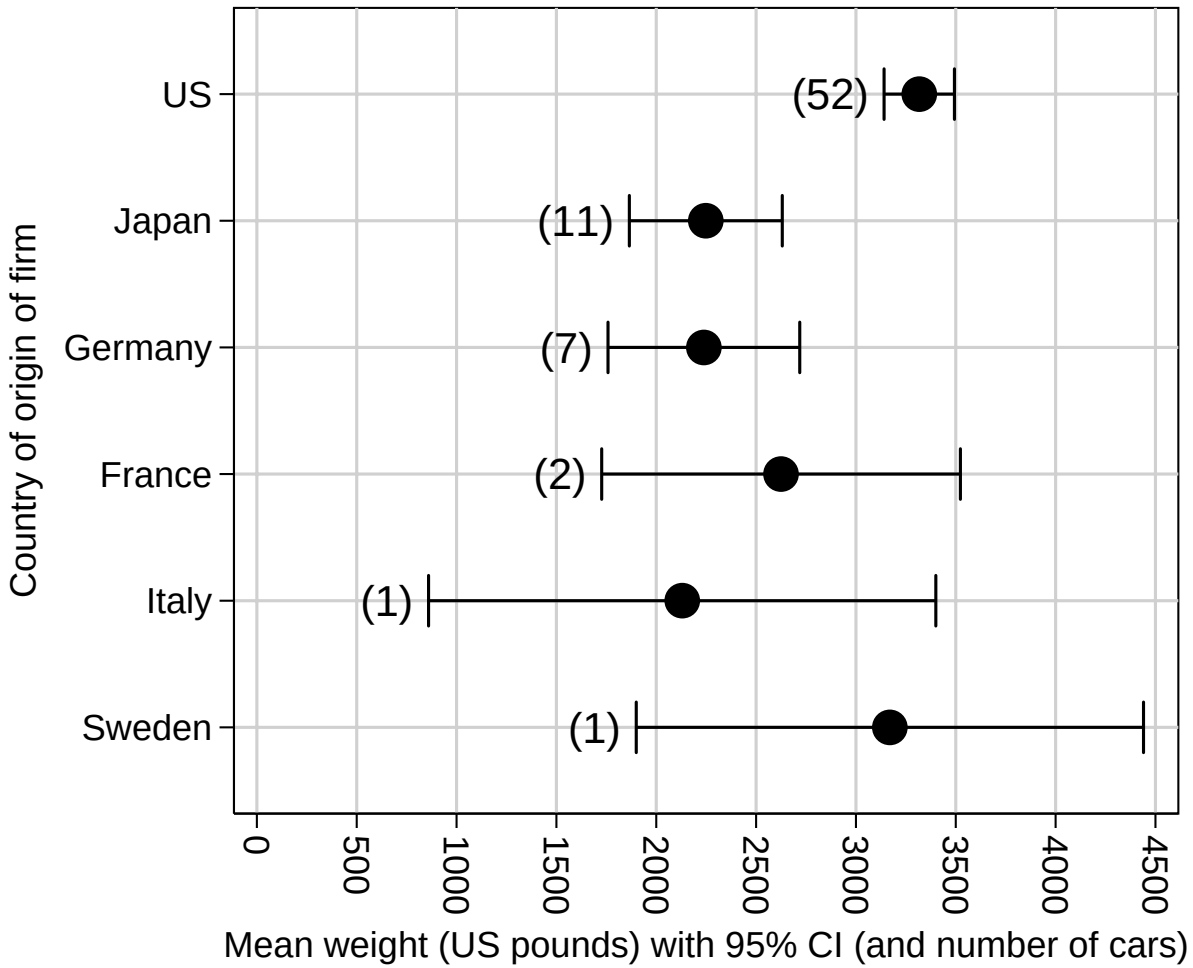
The power of resultssets

- We used `parmest` to produce a resultsset of confidence intervals.



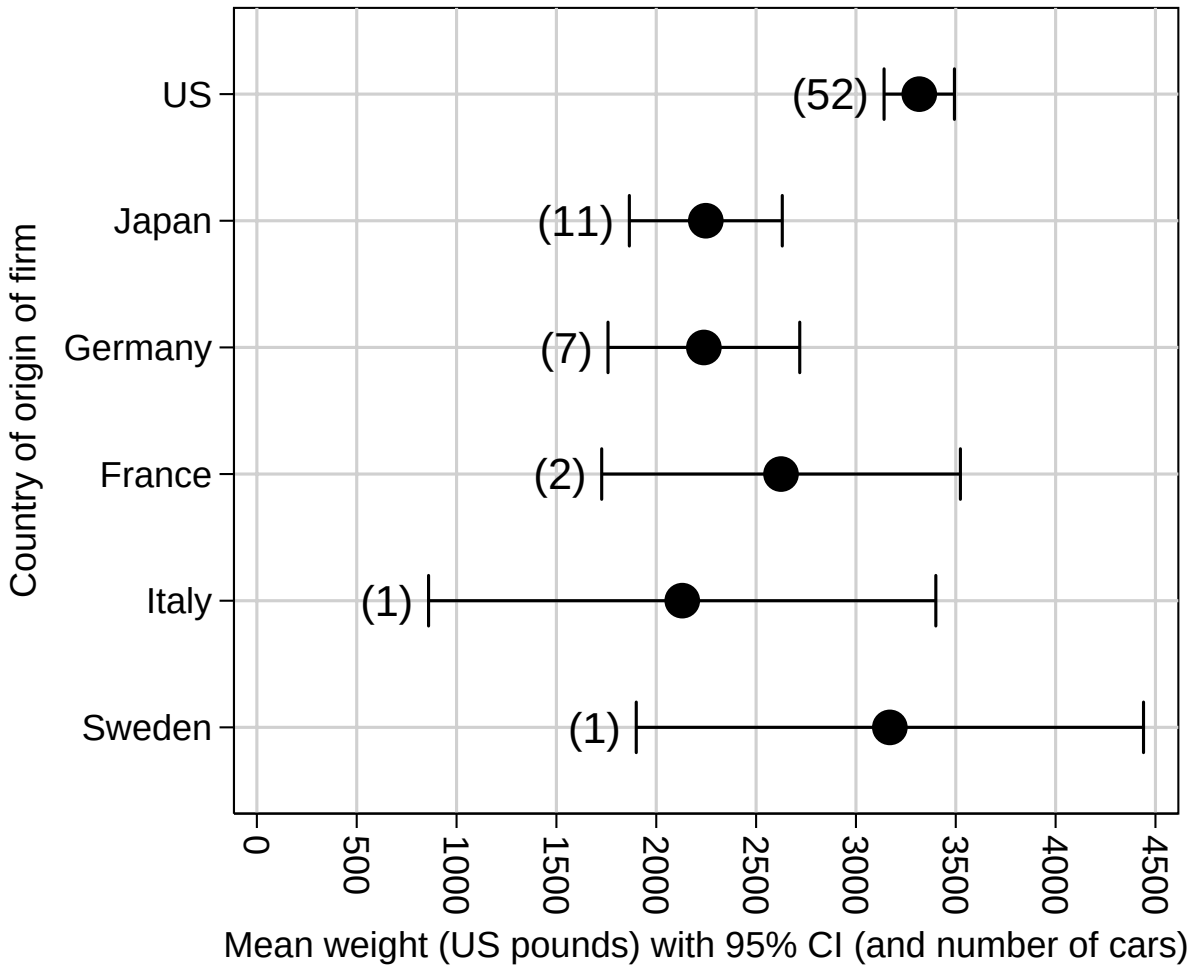
The power of resultssets

- We used `parmest` to produce a resultsset of confidence intervals.
- We reconstructed the vertical–axis variable `country`, using `descsave` and `factext`.



The power of resultssets

- We used `parmest` to produce a resultsset of confidence intervals.
- We reconstructed the vertical–axis variable `country`, using `descsave` and `factext`.
- We then added the frequences by merging in a resultsset produced by `xcontract`.



Resultsspreadsheets

Resultsspreadsheets

- A resultsspreadsheet is a table of results in a generic text format, with cells arrayed into rows and columns.

Resultsspreadsheets

- A resultsspreadsheet is a table of results in a generic text format, with cells arrayed into rows and columns.
- The format (or row style) is defined using a column–delimiter string and (optionally) a row–begin string and/or a row–end string.

Resultsspreadsheets

- A resultsspreadsheet is a table of results in a generic text format, with cells arrayed into rows and columns.
- The format (or row style) is defined using a column–delimiter string and (optionally) a row–begin string and/or a row–end string.
- Examples of row styles include tab–separated, comma–separated, HTML, plain TEX and LATEX `tabular`.

Resultsspreadsheets

- A resultsspreadsheet is a table of results in a generic text format, with cells arrayed into rows and columns.
- The format (or row style) is defined using a column–delimiter string and (optionally) a row–begin string and/or a row–end string.
- Examples of row styles include tab–separated, comma–separated, HTML, plain TEX and LATEX `tabular`.
- Programs that create resultsspreadsheets include the official Stata utility `estimates table` and Ben Jann's SSC package `estout` (Jann, 2005a; Jann, 2005b).

Resultsspreadsheets

- A resultsspreadsheet is a table of results in a generic text format, with cells arrayed into rows and columns.
- The format (or row style) is defined using a column–delimiter string and (optionally) a row–begin string and/or a row–end string.
- Examples of row styles include tab–separated, comma–separated, HTML, plain TEX and LATEX `tabular`.
- Programs that create resultsspreadsheets include the official Stata utility `estimates table` and Ben Jann's SSC package `estout` (Jann, 2005a; Jann, 2005b).
- *However*, resultsspreadsheets can also be created from resultssets (as well as *vice versa*), using `outsheet` for tab–delimited and comma–delimited formats, or the SSC package `listtex` for other formats (eg TEX and HTML).

Resultsspreadsheets

- A resultsspreadsheet is a table of results in a generic text format, with cells arrayed into rows and columns.
- The format (or row style) is defined using a column–delimiter string and (optionally) a row–begin string and/or a row–end string.
- Examples of row styles include tab–separated, comma–separated, HTML, plain TEX and LATEX `tabular`.
- Programs that create resultsspreadsheets include the official Stata utility `estimates table` and Ben Jann's SSC package `estout` (Jann, 2005a; Jann, 2005b).
- *However*, resultsspreadsheets can also be created from resultssets (as well as *vice versa*), using `outsheet` for tab–delimited and comma–delimited formats, or the SSC package `listtex` for other formats (eg TEX and HTML).
- Resultsspreadsheets can be copied (or linked) into TEX or HTML documents, or converted into Microsoft Excel spreadsheets or Word tables.

Common table formats (or row styles) for resultsspreadsheets

Common table formats (or row styles) for resultsspreadsheets

Each of these table row styles has a row–begin string, a column–delimiter string and a row–end string, given here as Stata expressions. The row–begin and row–end strings may be empty strings.

Row style	*Row–begin*	*Column–delimiter*	*Row–end*
Tab–delimited	`""`	`char(9)`	`""`
HTML table	`"<tr><td>"`	`"</td><td>"`	`"</td></tr>"`
LaTeX `tabular`	`""`	`"&"`	`"\\"`
Plain TeX `halign`	`""`	`"&"`	`"\cr"`

Common table formats (or row styles) for resultsspreadsheets

Each of these table row styles has a row–begin string, a column–delimiter string and a row–end string, given here as Stata expressions. The row–begin and row–end strings may be empty strings.

Row style	*Row–begin*	*Column–delimiter*	*Row–end*
Tab–delimited	`""`	`char(9)`	`""`
HTML table	`"<tr><td>"`	`"</td><td>"`	`"</td></tr>"`
LaTeX `tabular`	`""`	`"&"`	`"\\"`
Plain TeX `halign`	`""`	`"&"`	`"\cr"`

Note that resultsspreadsheets of all styles are generic text tables. Therefore, any text can appear between the row–begin, column–delimiter and row–end strings, except for row–begin, column–delimiter and row–end strings.

The SSC package `estout`

The SSC package `estout`

The `estout` package (Jann, 2005a; Jann, 2005b) is very powerful indeed. It produces a very wide range of resultsspreadsheets from the results of any estimation command. We will demonstrate a very simple example. First, we run the regression command that we used earlier:

The SSC package `estout`

The `estout` package (Jann, 2005a; Jann, 2005b) is very powerful indeed. It produces a very wide range of resultsspreadsheets from the results of any estimation command. We will demonstrate a very simple example. First, we run the regression command that we used earlier:

```
. regress weight c_*, noconst nohead
------------------------------------------------------------------------------
      weight |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         c_1 |   3317.115   88.26823    37.58   0.000     3140.979    3493.252
         c_2 |   2248.182   191.9154    11.71   0.000     1865.221    2631.143
         c_3 |   2238.571   240.5786     9.30   0.000     1758.504    2718.639
         c_4 |        2625   450.0814     5.83   0.000     1726.877    3523.123
         c_5 |        2130   636.5112     3.35   0.001     859.8616    3400.138
         c_6 |        3170   636.5112     4.98   0.000     1899.862    4440.138
------------------------------------------------------------------------------
```

A resultsspreadsheet created by the SSC package `estout`

A resultsspreadsheet created by the SSC package `estout`

After the regression command, we run `estout` to produce a tab–delimited resultsspreadsheet with one row per model parameter and data on confidence limits:

A resultsspreadsheet created by the SSC package `estout`

After the regression command, we run `estout` to produce a tab–delimited resultsspreadsheet with one row per model parameter and data on confidence limits:

```
. estout using estout1.txt, ///
>   cells("b(fmt(%8.2f)) ci_l(fmt(%8.2f)) ci_u(fmt(%8.2f))") ///
>   replace label collabels(, lhs(label)) mlabels(, none)

label   b       min95   max95
country==US     3317.12 3140.98 3493.25
country==Japan  2248.18 1865.22 2631.14
country==Germany        2238.57 1758.50 2718.64
country==France 2625.00 1726.88 3523.12
country==Italy  2130.00 859.86  3400.14
country==Sweden 3170.00 1899.86 4440.14
```

A resultsspreadsheet created by the SSC package `estout`

After the regression command, we run `estout` to produce a tab–delimited resultsspreadsheet with one row per model parameter and data on confidence limits:

```
. estout using estout1.txt, ///
>   cells("b(fmt(%8.2f)) ci_l(fmt(%8.2f)) ci_u(fmt(%8.2f))") ///
>   replace label collabels(, lhs(label)) mlabels(, none)

label   b       min95   max95
country==US     3317.12 3140.98 3493.25
country==Japan  2248.18 1865.22 2631.14
country==Germany        2238.57 1758.50 2718.64
country==France 2625.00 1726.88 3523.12
country==Italy  2130.00 859.86  3400.14
country==Sweden 3170.00 1899.86 4440.14
```

The resultsspreadsheet is output to the Stata log, and also to a disk file `estout1.txt`, where it can be accessed and edited by a spreadsheet package (possibly even Microsoft Excel). *However* ...

Converting a resultsspreadsheet to a resultsset

Converting a resultsspreadsheet to a resultsset

. . . alternatively, the resultsspreadsheet can be converted to a resultsset using `insheet`, as follows:

Converting a resultsspreadsheet to a resultsset

. . . alternatively, the resultsspreadsheet can be converted to a resultsset using `insheet`, as follows:

```
. insheet using estout1.txt, clear
(4 vars, 6 obs)

. list, clean noobs
```

label	b	min95	max95
country==US	3317.12	3140.98	3493.25
country==Japan	2248.18	1865.22	2631.14
country==Germany	2238.57	1758.5	2718.64
country==France	2625	1726.88	3523.12
country==Italy	2130	859.86	3400.14
country==Sweden	3170	1899.86	4440.14

Converting a resultsspreadsheet to a resultsset

. . . alternatively, the resultsspreadsheet can be converted to a resultsset using `insheet`, as follows:

```
. insheet using estout1.txt, clear
(4 vars, 6 obs)

. list, clean noobs
```

label	b	min95	max95
country==US	3317.12	3140.98	3493.25
country==Japan	2248.18	1865.22	2631.14
country==Germany	2238.57	1758.5	2718.64
country==France	2625	1726.88	3523.12
country==Italy	2130	859.86	3400.14
country==Sweden	3170	1899.86	4440.14

Or, alternatively, the user may edit `estout1.txt` manually in a spreadsheet package, and *then* convert it to a resultsset. We can then produce . . .

Resultsplots

Resultsplots

- Programs to produce resultsplots include the SSC packages `eclplot` (Newson, 2005), which plots confidence intervals, and `smileplot` (Newson *et al.*, 2003), which plots p–values.

Resultsplots

- Programs to produce resultsplots include the SSC packages `eclplot` (Newson, 2005), which plots confidence intervals, and `smileplot` (Newson *et al.*, 2003), which plots p–values.
- In Stata (at present), a resultsplot can be produced *only* from a resultsset (and *not* directly from a resultsspreadsheet).

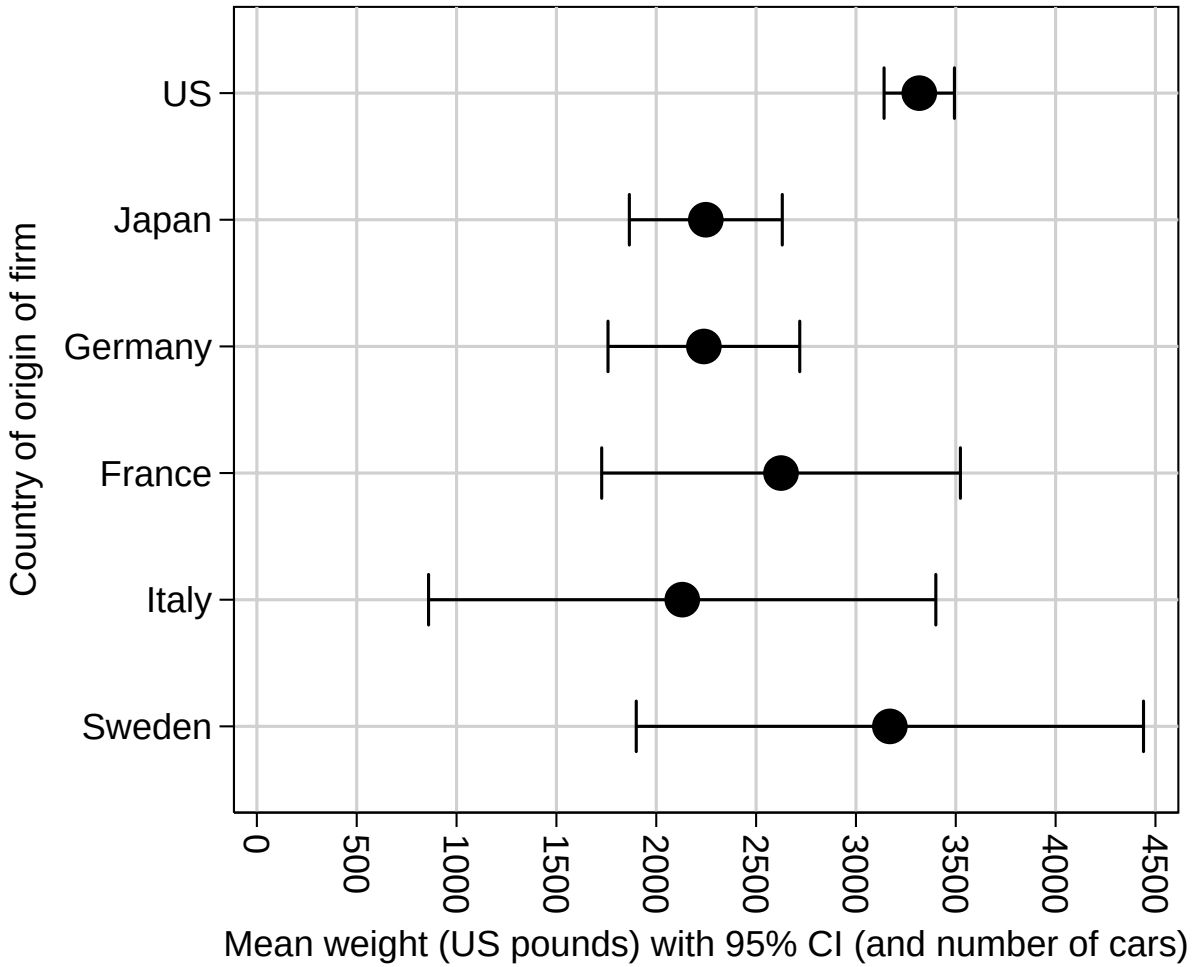
Resultsplots

- Programs to produce resultsplots include the SSC packages `eclplot` (Newson, 2005), which plots confidence intervals, and `smileplot` (Newson *et al.*, 2003), which plots p–values.
- In Stata (at present), a resultsplot can be produced *only* from a resultsset (and *not* directly from a resultsspreadsheet).
- *However*, as we have seen, it is easy to produce a resultsset from a resultsspreadsheet.

Resultsplots

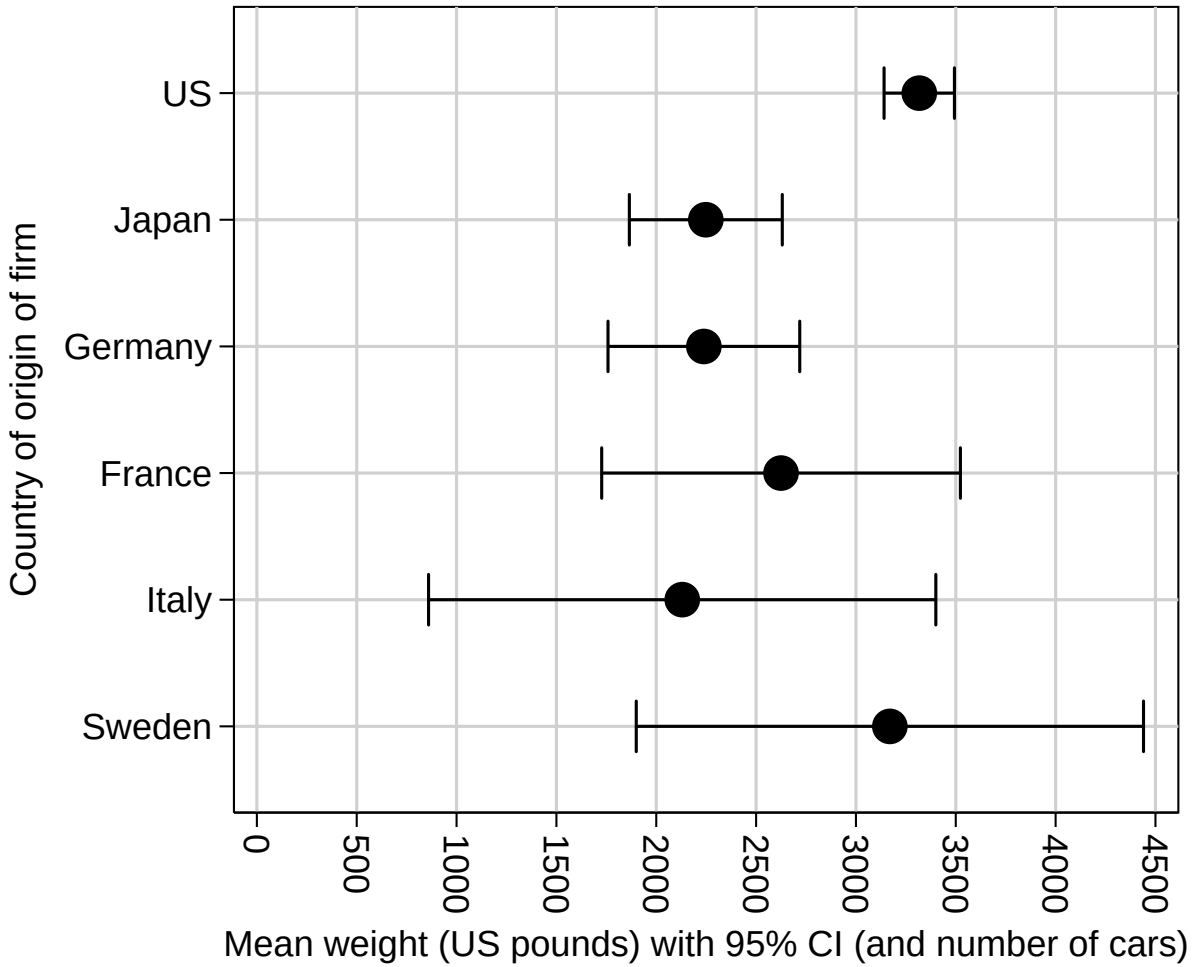
- Programs to produce resultsplots include the SSC packages `eclplot` (Newson, 2005), which plots confidence intervals, and `smileplot` (Newson *et al.*, 2003), which plots p–values.
- In Stata (at present), a resultsplot can be produced *only* from a resultsset (and *not* directly from a resultsspreadsheet).
- *However*, as we have seen, it is easy to produce a resultsset from a resultsspreadsheet.
- This makes life much easier for Stata users who do not like to do a lot of programming.

A resultsplot of the resultsset from the resultsspreadsheet



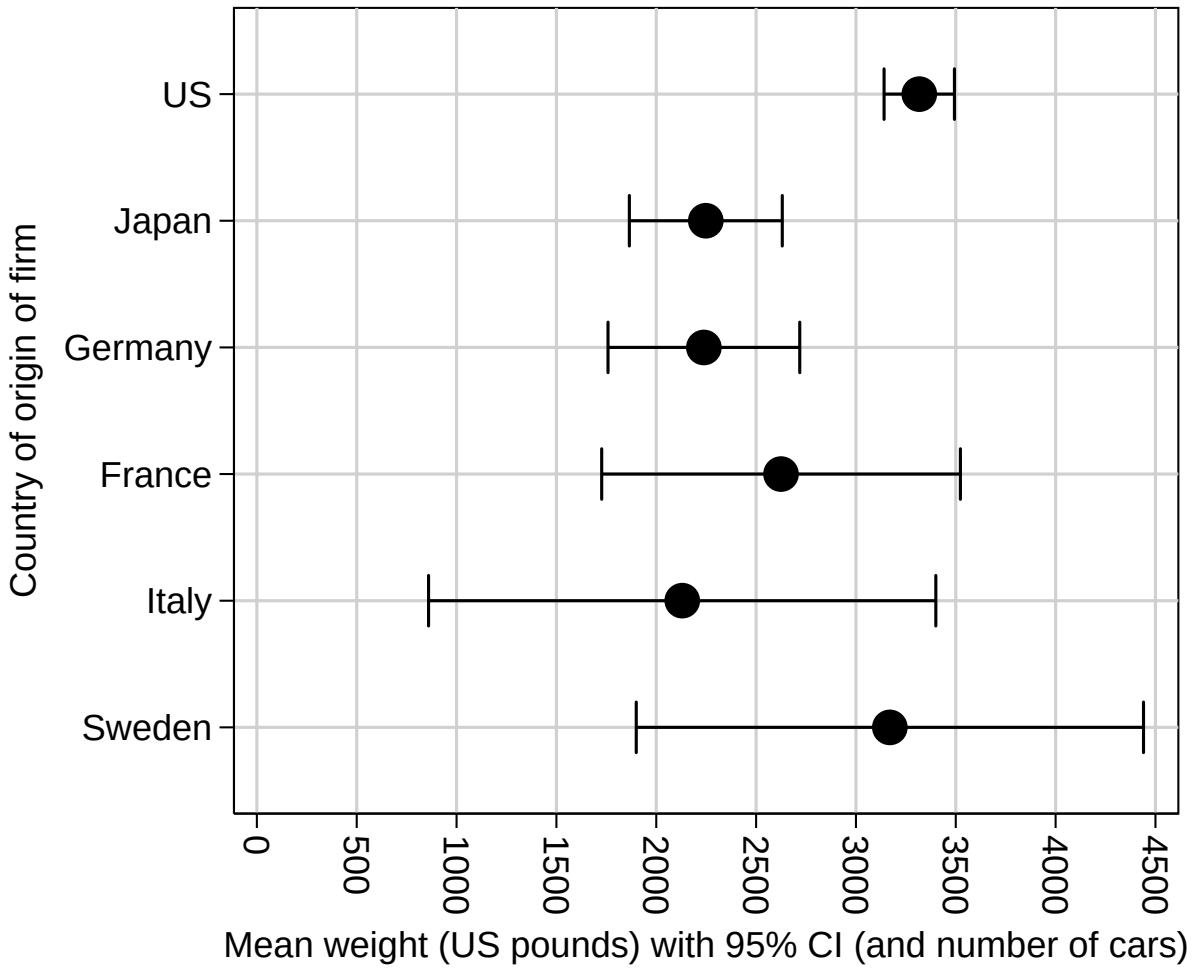
A resultsplot of the resultsset from the resultsspreadsheet

- This plot was produced by `eclplot` from the resultsset generated from the `estout` resultsspreadsheet.



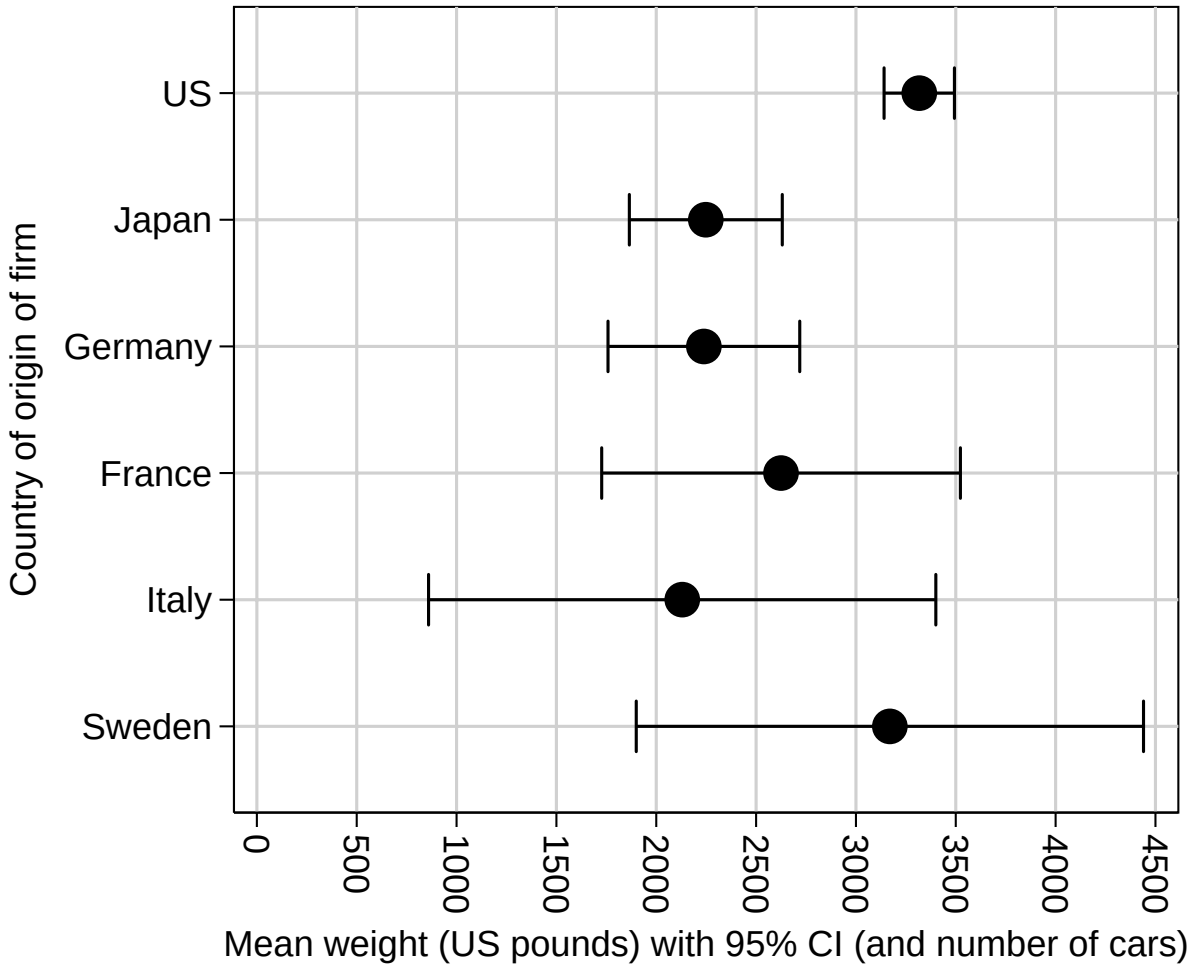
A resultsplot of the resultsset from the resultsspreadsheet

- This plot was produced by `eclplot` from the resultsset generated from the `estout` resultsspreadsheet.
- It has everything in the previous plot, except for the frequencies.



A resultsplot of the resultsset from the resultsspreadsheet

- This plot was produced by `eclplot` from the resultsset generated from the `estout` resultsspreadsheet.
- It has everything in the previous plot, except for the frequencies.
- (And these could have been added using Ben Jann's SSC package `estadd`.)



An alternative to `estout`: killing a resultsset

An alternative to `estout`: killing a resultsset

It is also possible to produce a table as a resultsspreadsheet from a resultsset using `listtex`, as follows:

An alternative to `estout`: killing a resultsset

It is also possible to produce a table as a resultsspreadsheet from a resultsset using `listtex`, as follows:

1. Convert all confidence limits to string variables using the SSC package `sdecode`, adding commas and parentheses.

An alternative to `estout`: killing a resultsset

It is also possible to produce a table as a resultsspreadsheet from a resultsset using `listtex`, as follows:

1. Convert all confidence limits to string variables using the SSC package `sdecode`, adding commas and parentheses.
2. If parallel rows of confidence intervals are required in the table, then use `reshape wide` to reshape the resultsset.

An alternative to `estout`: killing a resultsset

It is also possible to produce a table as a resultsspreadsheet from a resultsset using `listtex`, as follows:

1. Convert all confidence limits to string variables using the SSC package `sdecode`, adding commas and parentheses.
2. If parallel rows of confidence intervals are required in the table, then use `reshape wide` to reshape the resultsset.
3. If gap rows are required in the table, then add them using the SSC package `ingap`.

An alternative to `estout`: killing a resultsset

It is also possible to produce a table as a resultsspreadsheet from a resultsset using `listtex`, as follows:

1. Convert all confidence limits to string variables using the SSC package `sdecode`, adding commas and parentheses.
2. If parallel rows of confidence intervals are required in the table, then use `reshape wide` to reshape the resultsset.
3. If gap rows are required in the table, then add them using the SSC package `ingap`.
4. Use `listtex` to copy the resultsset to a resultsspreadsheet.

An alternative to `estout`: killing a resultsset

It is also possible to produce a table as a resultsspreadsheet from a resultsset using `listtex`, as follows:

1. Convert all confidence limits to string variables using the SSC package `sdecode`, adding commas and parentheses.
2. If parallel rows of confidence intervals are required in the table, then use `reshape wide` to reshape the resultsset.
3. If gap rows are required in the table, then add them using the SSC package `ingap`.
4. Use `listtex` to copy the resultsset to a resultsspreadsheet.

This is a destructive process, and is usually programmed in a do–file between a `preserve` and a `restore`. *However*, it produces the range of tables that I want to produce.

Example: Child lung function and analgesics in pregnancy (ALSPAC child cohort study, Bristol University, UK)

Example: Child lung function and analgesics in pregnancy (ALSPAC child cohort study, Bristol University, UK)

- In this study, we measured forced expiratory volume (FEV1), expressed in "SD units", in 6609 children at 8 years of age.

Example: Child lung function and analgesics in pregnancy (ALSPAC child cohort study, Bristol University, UK)

- In this study, we measured forced expiratory volume (FEV1), expressed in "SD units", in 6609 children at 8 years of age.
- We had previously asked their mothers their levels of use of 2 analgesics (paracetamol and aspirin) at 2 stages of pregnancy (0–20 and 20–32 weeks).

Example: Child lung function and analgesics in pregnancy (ALSPAC child cohort study, Bristol University, UK)

- In this study, we measured forced expiratory volume (FEV1), expressed in "SD units", in 6609 children at 8 years of age.
- We had previously asked their mothers their levels of use of 2 analgesics (paracetamol and aspirin) at 2 stages of pregnancy (0–20 and 20–32 weeks).
- Analgesic exposure levels were "Never", "Some days", "Most days" and "Unknown". We measured differences in mean FEV1, compared to the reference category of never–users.

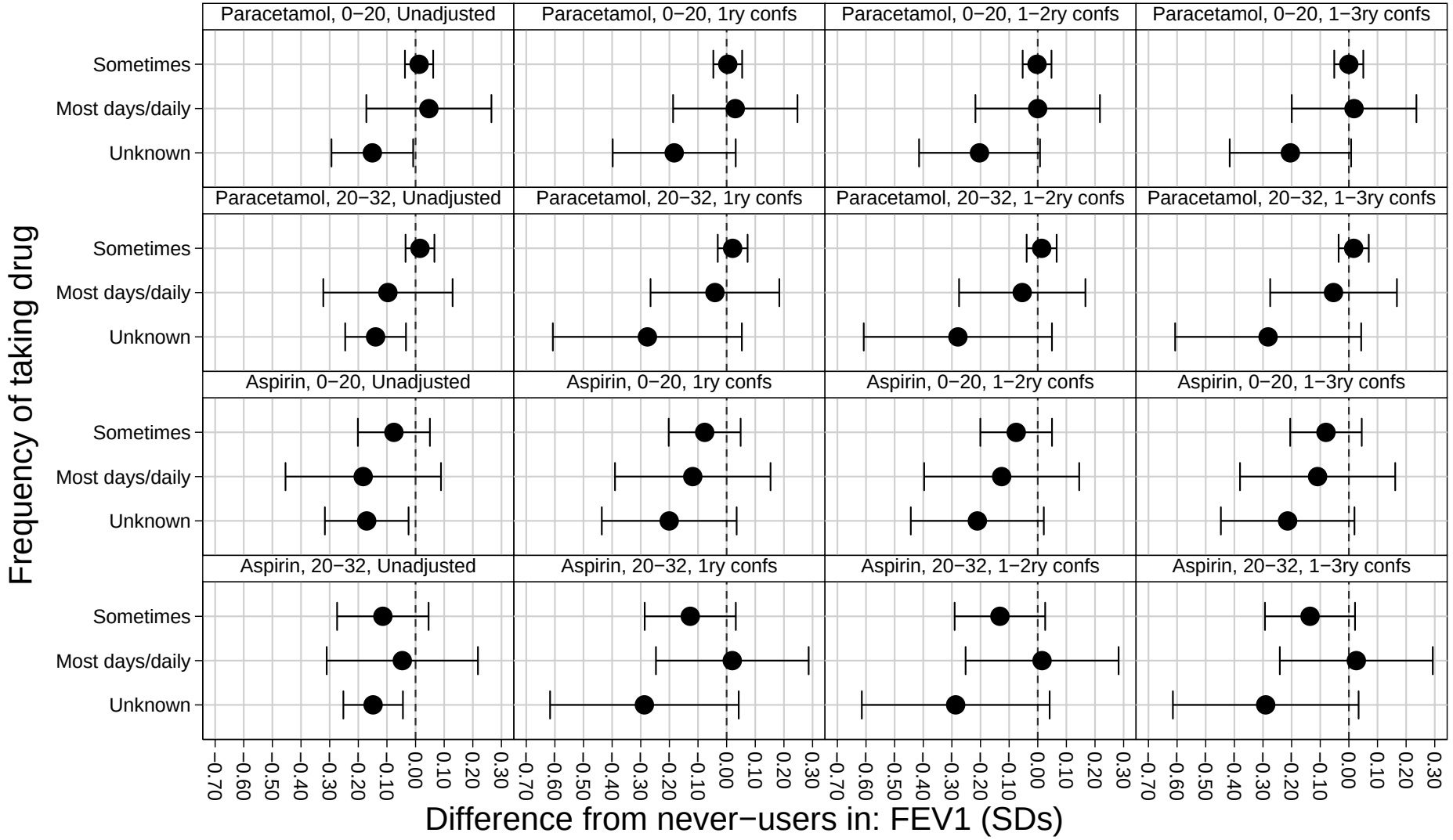
Example: Child lung function and analgesics in pregnancy (ALSPAC child cohort study, Bristol University, UK)

- In this study, we measured forced expiratory volume (FEV1), expressed in "SD units", in 6609 children at 8 years of age.
- We had previously asked their mothers their levels of use of 2 analgesics (paracetamol and aspirin) at 2 stages of pregnancy (0–20 and 20–32 weeks).
- Analgesic exposure levels were "Never", "Some days", "Most days" and "Unknown". We measured differences in mean FEV1, compared to the reference category of never–users.
- These differences could be unadjusted, *or* adjusted for one of 3 non–empty nested sets of confounders.

Example: Child lung function and analgesics in pregnancy (ALSPAC child cohort study, Bristol University, UK)

- In this study, we measured forced expiratory volume (FEV1), expressed in "SD units", in 6609 children at 8 years of age.
- We had previously asked their mothers their levels of use of 2 analgesics (paracetamol and aspirin) at 2 stages of pregnancy (0–20 and 20–32 weeks).
- Analgesic exposure levels were "Never", "Some days", "Most days" and "Unknown". We measured differences in mean FEV1, compared to the reference category of never–users.
- These differences could be unadjusted, *or* adjusted for one of 3 non–empty nested sets of confounders.
- We therefore calculated confidence intervals for $2 \times 2 \times 3 \times 4 = 48$ differences, generated by 2 analgesics, 2 stages of pregnancy, 3 non–reference exposure levels, and 4 confounder sets.

A resultsplot with $2 \times 2 \times 3 \times 4 = 48$ confidence intervals



Graphs by drug, Gestation week, and Confounder set

`metaparm` – confidence intervals for linear combinations of *independently–estimated* parameters

`metaparm` – confidence intervals for linear combinations of *independently–estimated* parameters

- The SSC package `metaparm` is complementary to `lincom`, and to the SSC package `lincomest` (Newson, 2003).

`metaparm` – confidence intervals for linear combinations of *independently–estimated* parameters

- The SSC package `metaparm` is complementary to `lincom`, and to the SSC package `lincomest` (Newson, 2003).
- It inputs a resultsset with 1 observation for each of a set of *independently–estimated* parameters, and data on their estimates and standard errors.

`metaparm` – confidence intervals for linear combinations of *independently–estimated* parameters

- The SSC package `metaparm` is complementary to `lincom`, and to the SSC package `lincomest` (Newson, 2003).
- It inputs a resultsset with 1 observation for each of a set of *independently–estimated* parameters, and data on their estimates and standard errors.
- It outputs a `parmest`–format resultsset, with 1 observation (or 1 observation per by–group), and data on estimates and confidence limits for a linear combination of the parameters, specified by the `weight` expression.

`metaparm` – confidence intervals for linear combinations of *independently–estimated* parameters

- The SSC package `metaparm` is complementary to `lincom`, and to the SSC package `lincomest` (Newson, 2003).
- It inputs a resultsset with 1 observation for each of a set of *independently–estimated* parameters, and data on their estimates and standard errors.
- It outputs a `parmest`–format resultsset, with 1 observation (or 1 observation per by–group), and data on estimates and confidence limits for a linear combination of the parameters, specified by the `weight` expression.
- The obvious application of `metaparm` is meta–analysis.

`metaparm` – confidence intervals for linear combinations of *independently–estimated* parameters

- The SSC package `metaparm` is complementary to `lincom`, and to the SSC package `lincomest` (Newson, 2003).
- It inputs a resultsset with 1 observation for each of a set of *independently–estimated* parameters, and data on their estimates and standard errors.
- It outputs a `parmest`–format resultsset, with 1 observation (or 1 observation per by–group), and data on estimates and confidence limits for a linear combination of the parameters, specified by the `weight` expression.
- The obvious application of `metaparm` is meta–analysis.
- *However*, `metaparm` allows negative `iweights`, and can therefore *also* calculate confidence limits for differences (or ratios) between parameters.

`metaparm` – confidence intervals for linear combinations of *independently–estimated* parameters

- The SSC package `metaparm` is complementary to `lincom`, and to the SSC package `lincomest` (Newson, 2003).
- It inputs a resultsset with 1 observation for each of a set of *independently–estimated* parameters, and data on their estimates and standard errors.
- It outputs a `parmest`–format resultsset, with 1 observation (or 1 observation per by–group), and data on estimates and confidence limits for a linear combination of the parameters, specified by the `weight` expression.
- The obvious application of `metaparm` is meta–analysis.
- *However*, `metaparm` allows negative `iweights`, and can therefore *also* calculate confidence limits for differences (or ratios) between parameters.
- Or even for interactions, defined as differences between differences (or ratios between ratios) between parameters.

Example: Vitamin A supplementation in infectious disease

Example: Vitamin A supplementation in infectious disease

- Glasziou and MacKerras (1993) meta–analysed 5 community trials of vitamin A supplementation in infectious disease patients.

Example: Vitamin A supplementation in infectious disease

- Glasziou and MacKerras (1993) meta–analysed 5 community trials of vitamin A supplementation in infectious disease patients.
- The outcome in all 5 studies was death from all causes.

Example: Vitamin A supplementation in infectious disease

- Glasziou and MacKerras (1993) meta–analysed 5 community trials of vitamin A supplementation in infectious disease patients.
- The outcome in all 5 studies was death from all causes.
- The active treatment regime (vitamin A supplementation) varied between studies in frequency and dose, which was expressed in international units (IU).

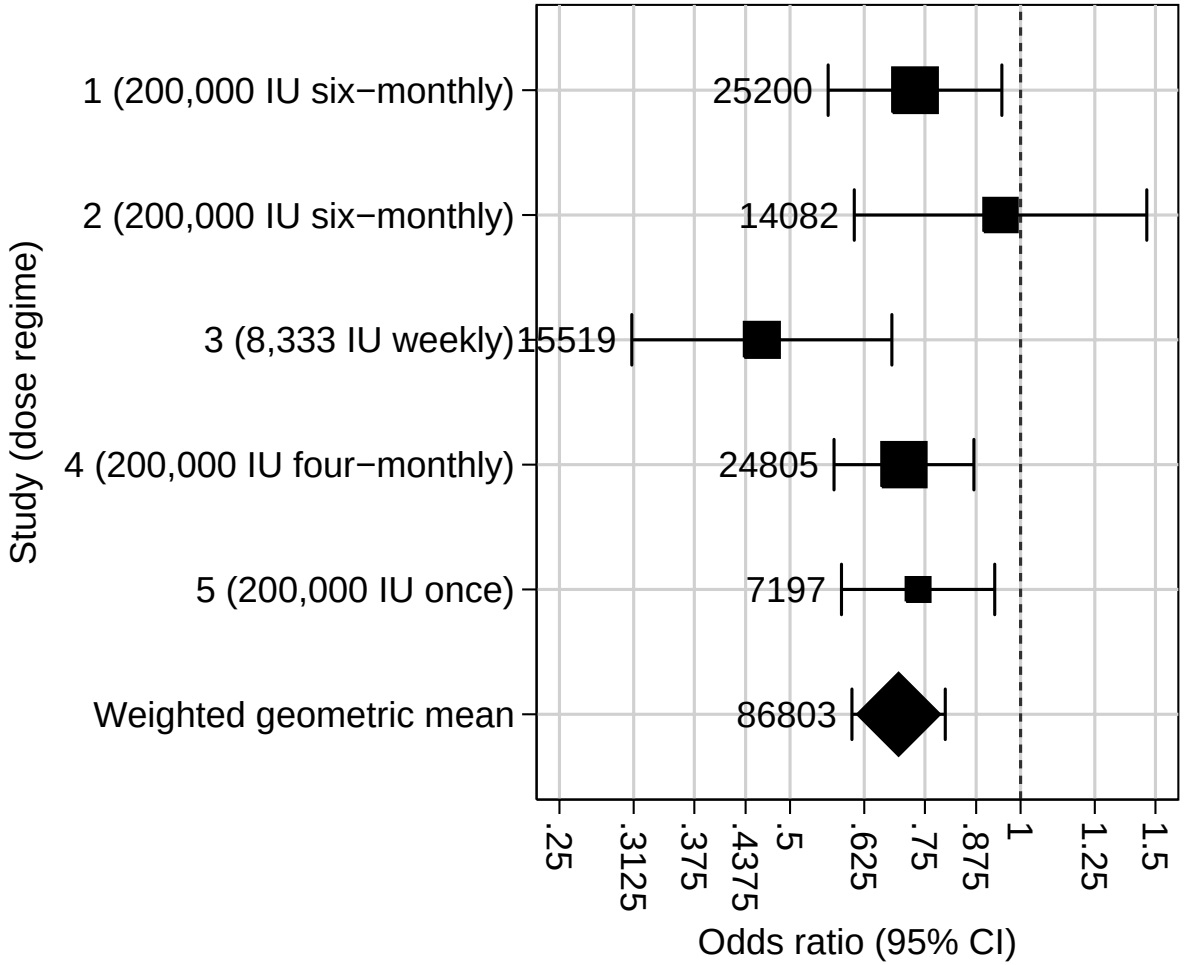
Example: Vitamin A supplementation in infectious disease

- Glasziou and MacKerras (1993) meta–analysed 5 community trials of vitamin A supplementation in infectious disease patients.
- The outcome in all 5 studies was death from all causes.
- The active treatment regime (vitamin A supplementation) varied between studies in frequency and dose, which was expressed in international units (IU).
- The data were reproduced in Bland (2000).

Example: Vitamin A supplementation in infectious disease

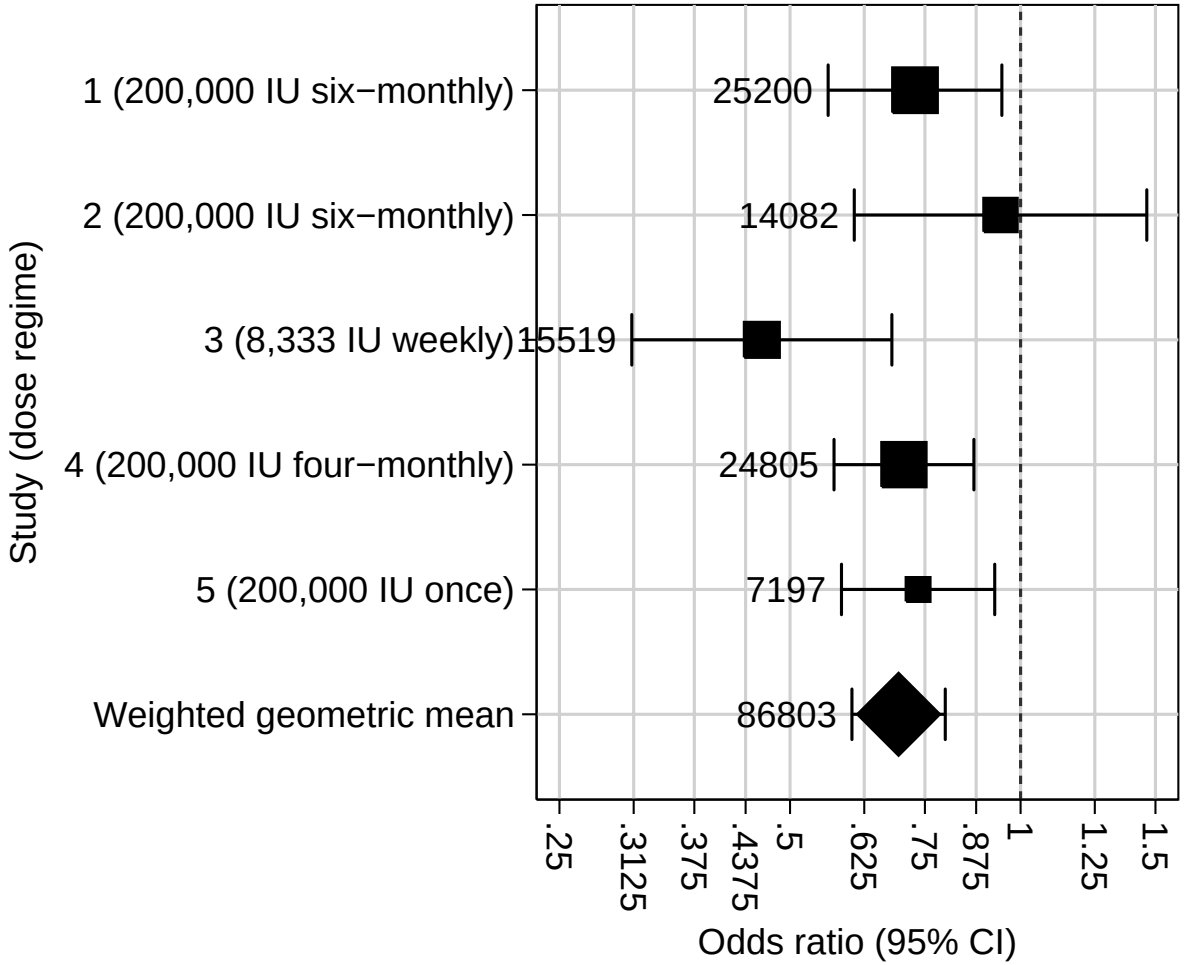
- Glasziou and MacKerras (1993) meta–analysed 5 community trials of vitamin A supplementation in infectious disease patients.
- The outcome in all 5 studies was death from all causes.
- The active treatment regime (vitamin A supplementation) varied between studies in frequency and dose, which was expressed in international units (IU).
- The data were reproduced in Bland (2000).
- We meta–analysed the data using `parmby`, `metaparm` and `eclplot`.

A Cochrane forest plot produced using `metaparm` and `eclplot`



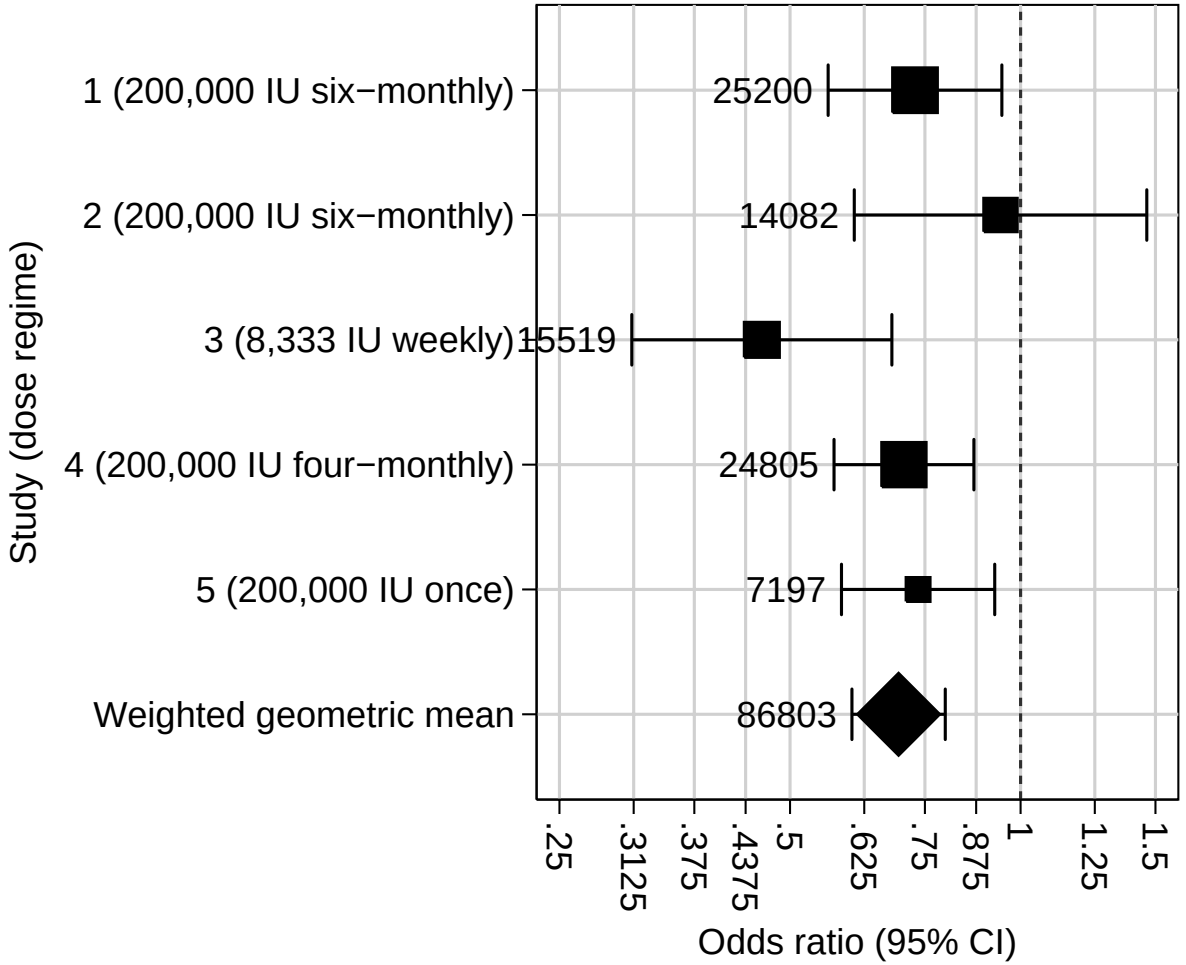
A Cochrane forest plot produced using `metaparm` and `eclplot`

- We used `parmby` to produce a resultsset, with one observation per study and data on odds ratios.



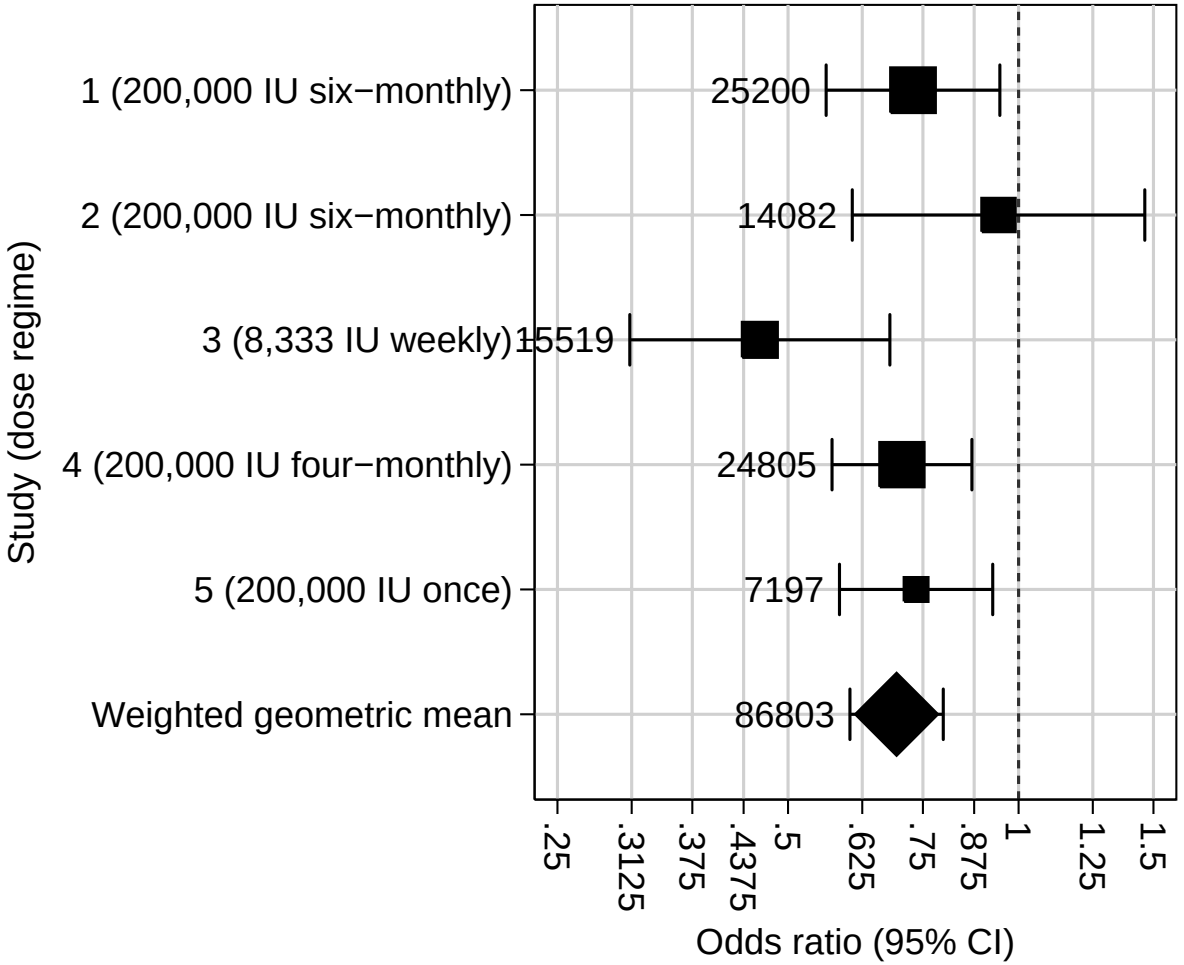
A Cochrane forest plot produced using `metaparm` and `eclplot`

- We used `parmby` to produce a resultsset, with one observation per study and data on odds ratios.
- We used `metaparm` to produce a second resultsset, which we `appended` to the first.



A Cochrane forest plot produced using `metaparm` and `eclplot`

- We used `parmby` to produce a resultsset, with one observation per study and data on odds ratios.
- We used `metaparm` to produce a second resultsset, which we `appended` to the first.
- Note that `eclplot` can weight symbols by study size.



The two "central dogmas" of resultsprocessing

The two "central dogmas" of resultsprocessing

- The **central dogma of molecular genetics** can be stated:

The two "central dogmas" of resultsprocessing

- The **central dogma of molecular genetics** can be stated:
  $DNA \Longrightarrow RNA \Longrightarrow protein$

The two "central dogmas" of resultsprocessing

- The **central dogma of molecular genetics** can be stated:
  $DNA \Longrightarrow RNA \Longrightarrow protein$
- Similarly, in resultsprocessing, the **resultsset–central dogma** is followed by `statsby`, `parmby`, `parmest`, `xcontract` and `xcollapse` users, and states that

The two "central dogmas" of resultsprocessing

- The **central dogma of molecular genetics** can be stated:
  $DNA \Longrightarrow RNA \Longrightarrow protein$
- Similarly, in resultsprocessing, the **resultsset–central dogma** is followed by `statsby`, `parmby`, `parmest`, `xcontract` and `xcollapse` users, and states that

$$datasets \Longrightarrow resultssets \Longrightarrow \begin{cases} resultsplots \\ resultsspreadsheets \end{cases}$$

The two "central dogmas" of resultsprocessing

- The **central dogma of molecular genetics** can be stated:
  $DNA \Longrightarrow RNA \Longrightarrow protein$
- Similarly, in resultsprocessing, the **resultsset–central dogma** is followed by `statsby`, `parmby`, `parmest`, `xcontract` and `xcollapse` users, and states that

$$datasets \Longrightarrow resultssets \Longrightarrow \begin{cases} resultsplots \\ resultsspreadsheets \end{cases}$$

- And the **resultspreadsheet–central dogma** is followed by `estimates table` and `estout` users, and states that

The two "central dogmas" of resultsprocessing

- The **central dogma of molecular genetics** can be stated:
  $DNA \Longrightarrow RNA \Longrightarrow protein$
- Similarly, in resultsprocessing, the **resultsset–central dogma** is followed by `statsby`, `parmby`, `parmest`, `xcontract` and `xcollapse` users, and states that

$$datasets \Longrightarrow resultssets \Longrightarrow \begin{cases} resultsplots \\ resultsspreadsheets \end{cases}$$

- And the **resultspreadsheet–central dogma** is followed by `estimates table` and `estout` users, and states that

$$datasets \Longrightarrow resultsspreadsheets \Longrightarrow resultssets \Longrightarrow resultsplots$$

The two "central dogmas" of resultsprocessing

- The **central dogma of molecular genetics** can be stated:
  $DNA \Longrightarrow RNA \Longrightarrow protein$
- Similarly, in resultsprocessing, the **resultsset–central dogma** is followed by `statsby`, `parmby`, `parmest`, `xcontract` and `xcollapse` users, and states that

$$datasets \Longrightarrow resultssets \Longrightarrow \begin{cases} resultsplots \\ resultsspreadsheets \end{cases}$$

- And the **resultspreadsheet–central dogma** is followed by `estimates table` and `estout` users, and states that

$$datasets \Longrightarrow resultsspreadsheets \Longrightarrow resultssets \Longrightarrow resultsplots$$

- The two resultsprocessing dogmas are complementary, and each one has advantages and disadvantages.

The central role: resultssets *versus* resultsspreadsheets

The central role: resultssets *versus* resultsspreadsheets

- Resultsspreadsheets are *much* easier to edit manually than resultssets (using a spreadsheet package, a text editor or even a word processor).

The central role: resultssets *versus* resultsspreadsheets

- Resultsspreadsheets are *much* easier to edit manually than resultssets (using a spreadsheet package, a text editor or even a word processor).
- And many users do this, because they do not have the time to learn a lot of programming.

The central role: resultssets *versus* resultsspreadsheets

- Resultsspreadsheets are *much* easier to edit manually than resultssets (using a spreadsheet package, a text editor or even a word processor).
- And many users do this, because they do not have the time to learn a lot of programming.
- *On the other hand*, resultssets *may* be more convenient to process in do–files.

The central role: resultssets *versus* resultsspreadsheets

- Resultsspreadsheets are *much* easier to edit manually than resultssets (using a spreadsheet package, a text editor or even a word processor).
- And many users do this, because they do not have the time to learn a lot of programming.
- *On the other hand*, resultssets *may* be more convenient to process in do–files.
- For instance, variables in a resultsset have storage types, display formats, variable labels and value labels, often inherited from variables of the same names in the original dataset.

The central role: resultssets *versus* resultsspreadsheets

- Resultsspreadsheets are *much* easier to edit manually than resultssets (using a spreadsheet package, a text editor or even a word processor).
- And many users do this, because they do not have the time to learn a lot of programming.
- *On the other hand*, resultssets *may* be more convenient to process in do–files.
- For instance, variables in a resultsset have storage types, display formats, variable labels and value labels, often inherited from variables of the same names in the original dataset.
- Resultssets are most useful when plots and tables are being mass–produced, as in the child cohort example.

Conclusions

Conclusions

- The resultsspreadsheet–central dogma is probably preferred by most users, most of the time.

Conclusions

- The resultsspreadsheet–central dogma is probably preferred by most users, most of the time.
- The resultsset–central dogma is followed by a minority of programmers, who want to write a do–file to do *everything*, with little or no manual intervention.

Conclusions

- The resultsspreadsheet–central dogma is probably preferred by most users, most of the time.
- The resultsset–central dogma is followed by a minority of programmers, who want to write a do–file to do *everything*, with little or no manual intervention.
- And the same packages are often useful under either dogma!

References (1)

Bland M. 2000. *An introduction to medical statistics. 3rd ed.* Oxford: Oxford University Press.

Glasziou P.P. and Mackerras D. E. M. 1993. Vitamin A supplementation in infectious disease: a meta–analysis. *British Medical Journal* **306**: 366-370.

Jann B. 2005a. From regression estimates to document tables: output formatting using `estout`. Presented at the 11th UK Stata User Meeting, 17–18 May, 2005.

Jann B. 2005b. Making regression tables from stored estimates. *The Stata Journal* **5(3)**: 288–308.

Newson R. 2002. Creating plots and tables of estimation results using `parmest` and friends. Presented at the 8th UK Stata User Meeting, 20–21 May, 2002.

Newson R and the ALSPAC Study Team. 2003. Multiple–test procedures and smile plots. *The Stata Journal* **3(2)**: 109–132.

References (2)

Newson R. 2003. Confidence intervals and p–values for delivery to the end user. *The Stata Journal* **3(3)**: 245–269.

Newson R. 2004. From datasets to resultssets in Stata. Presented at the 10th UK Stata User Meeting, 28–29 June, 2004.

Newson R. 2005. Generalized confidence interval plots using commands or dialogs. Presented at the 11th UK Stata User Meeting, 17–18 May, 2005.

All Stata User Meeting presentations referenced (and this one) can be downloaded from *http://www.stata.com/meeting/proceedings.html*

The packages `descsave`, `estadd`, `estout`, `factext`, `listtex`, `metaparm`, `parmest`, `regaxis`, `scheme_rbn1mono`, `sencode`, `sdecode`, `xcollapse` and `xcontract` (mentioned and/or used in this presentation) can be downloaded from SSC using the `ssc` command in Stata.