

Tools for supporting the development of robust SAS macros

Martin Gregory

PhUSE 2010

Berlin, 18-20 October 2010

Overview

- Review of techniques for robust macros
- Emacs – the extensible editor
- Emacs Speaks Statistics
- Emacs extensions for robust macro techniques

Techniques for writing robust SAS macros*

- parameter design
- ✓ parameter validation
- ✓ checking and reacting to outcomes
- ✓ returning status to the caller
- restoring the environment
- ✓ testing
- ✓ macro variable scope
- documentation

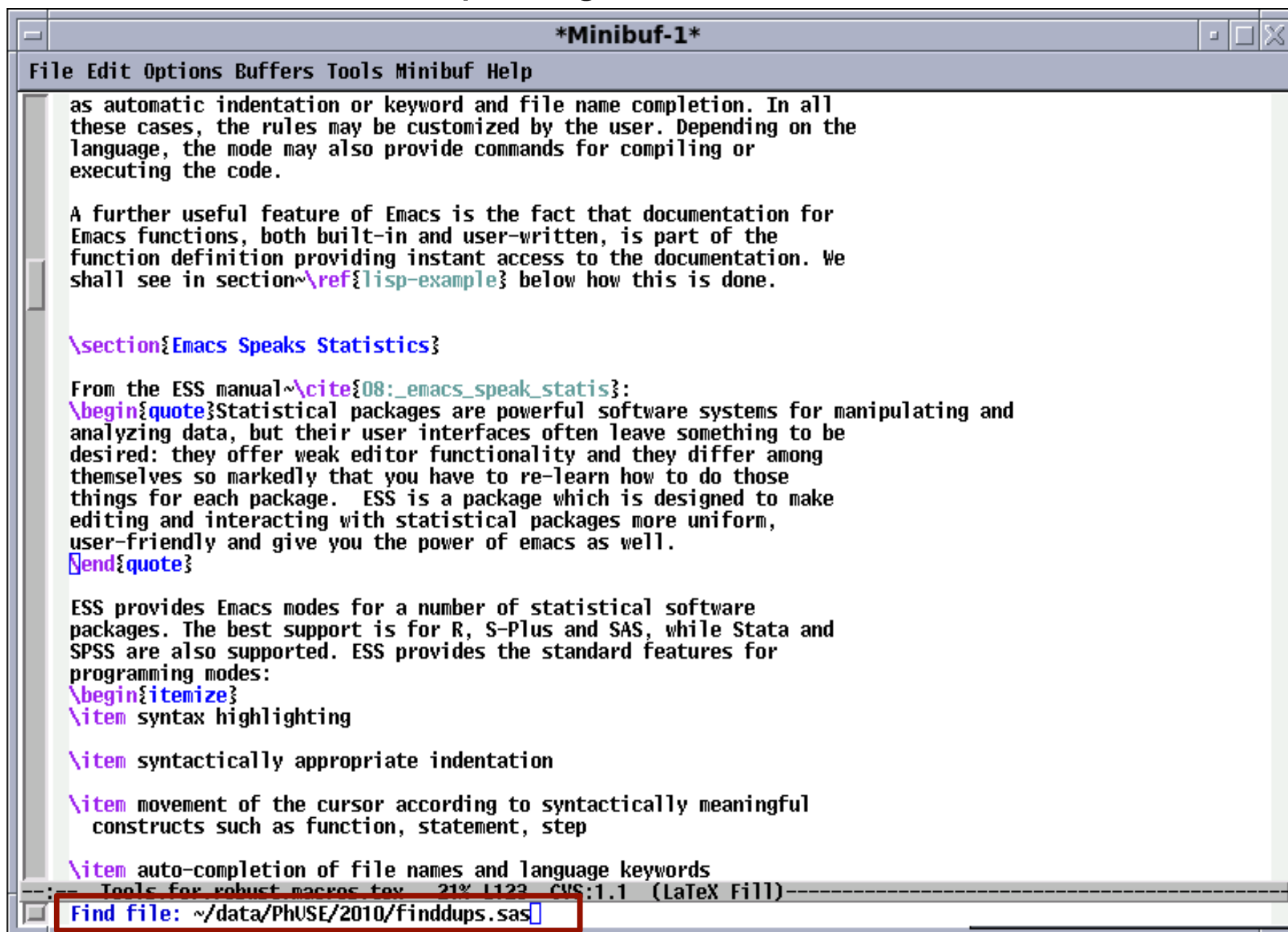
Tools support standards

- Standards are useful:
 - no need to re-invent the wheel each time
 - taking over someone else's work
- Standards are necessary
 - certainly in the pharmaceutical industry
- Standards must be implemented by people
 - different people
 - people under pressure
- Tools make following standards easy

GNU Emacs – the extensible programmer's editor

- Examples of what emacs does really well:
 - process multiple files
 - modes for almost all programming languages
 - ease of extension
 - built-in documentation
 - version control interface

Basic Emacs features: opening files



The screenshot shows an Emacs window titled `*Minibuf-1*`. The menu bar includes `File Edit Options Buffers Tools Minibuf Help`. The main text area contains the following content:

as automatic indentation or keyword and file name completion. In all these cases, the rules may be customized by the user. Depending on the language, the mode may also provide commands for compiling or executing the code.

A further useful feature of Emacs is the fact that documentation for Emacs functions, both built-in and user-written, is part of the function definition providing instant access to the documentation. We shall see in section`\ref{lisp-example}` below how this is done.

`\section{Emacs Speaks Statistics}`

From the ESS manual`\cite{08:_emacs_speak_statist}`:

`\begin{quote}`Statistical packages are powerful software systems for manipulating and analyzing data, but their user interfaces often leave something to be desired: they offer weak editor functionality and they differ among themselves so markedly that you have to re-learn how to do those things for each package. ESS is a package which is designed to make editing and interacting with statistical packages more uniform, user-friendly and give you the power of emacs as well.

`\end{quote}`

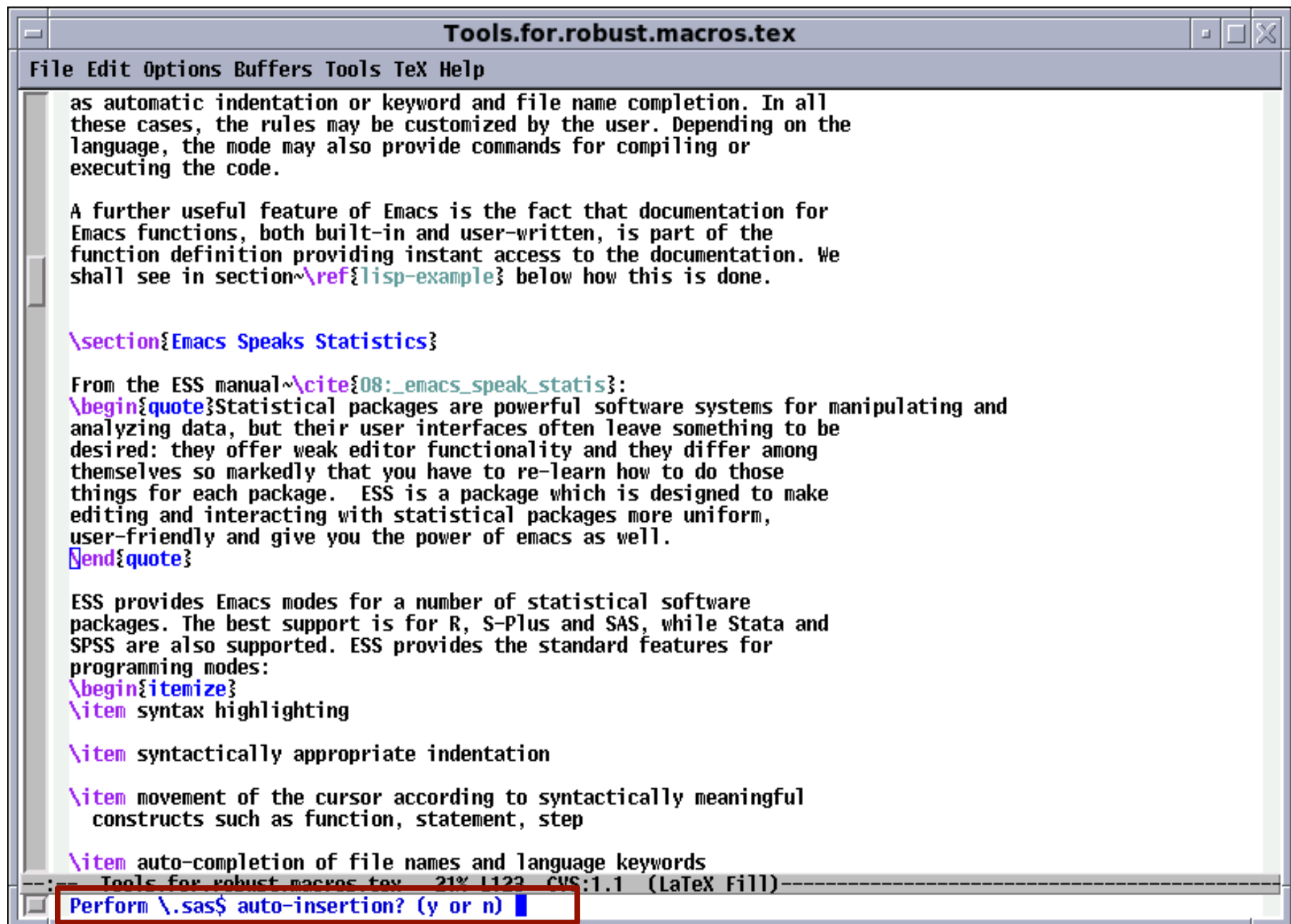
ESS provides Emacs modes for a number of statistical software packages. The best support is for R, S-Plus and SAS, while Stata and SPSS are also supported. ESS provides the standard features for programming modes:

- `\begin{itemize}`
- `\item` syntax highlighting
- `\item` syntactically appropriate indentation
- `\item` movement of the cursor according to syntactically meaningful constructs such as function, statement, step
- `\item` auto-completion of file names and language keywords

Tools for robust macros tex 21% 1123 CVS:1.1 (LaTeX Fill)

Find file: `~/data/PhUSE/2010/finddups.sas`

Basic Emacs features: automatic template insertion



The screenshot shows an Emacs window titled "Tools.for.robust.macros.tex". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "TeX", and "Help". The main text area contains LaTeX-style macros for document structure and lists of features. At the bottom, a status bar shows "Tools for robust macros.tex 21% 1122 CUS:1.1 (LaTeX Fill)". A red rectangular box highlights a confirmation prompt at the bottom of the window.

```
as automatic indentation or keyword and file name completion. In all
these cases, the rules may be customized by the user. Depending on the
language, the mode may also provide commands for compiling or
executing the code.

A further useful feature of Emacs is the fact that documentation for
Emacs functions, both built-in and user-written, is part of the
function definition providing instant access to the documentation. We
shall see in section~\ref{lisp-example} below how this is done.

\section{Emacs Speaks Statistics}

From the ESS manual~\cite{08:_emacs_speak_statist}:
\begin{quote}Statistical packages are powerful software systems for manipulating and
analyzing data, but their user interfaces often leave something to be
desired: they offer weak editor functionality and they differ among
themselves so markedly that you have to re-learn how to do those
things for each package. ESS is a package which is designed to make
editing and interacting with statistical packages more uniform,
user-friendly and give you the power of emacs as well.
\end{quote}

ESS provides Emacs modes for a number of statistical software
packages. The best support is for R, S-Plus and SAS, while Stata and
SPSS are also supported. ESS provides the standard features for
programming modes:
\begin{itemize}
\item syntax highlighting

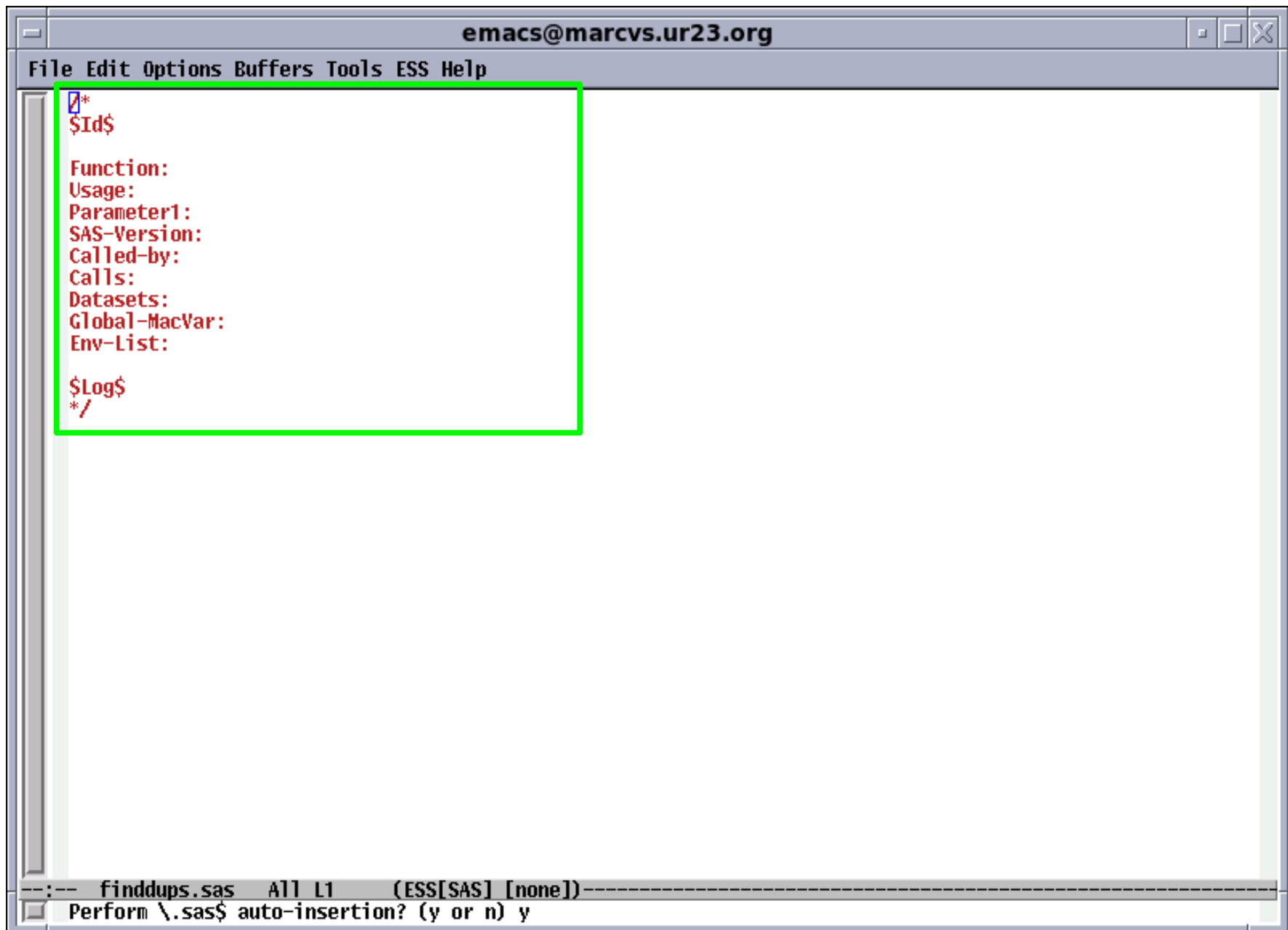
\item syntactically appropriate indentation

\item movement of the cursor according to syntactically meaningful
constructs such as function, statement, step

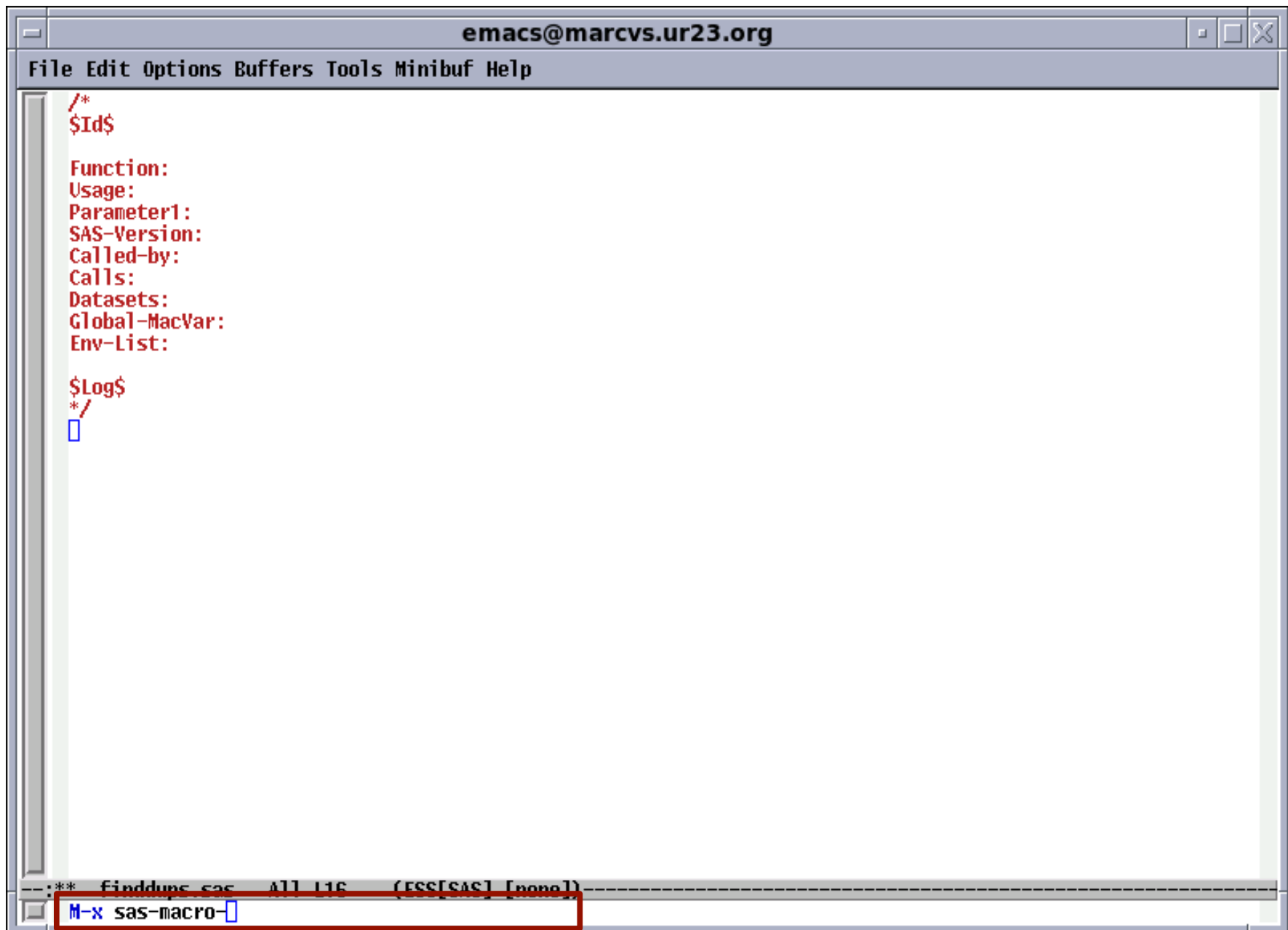
\item auto-completion of file names and language keywords
\end{itemize}

Tools for robust macros.tex 21% 1122 CUS:1.1 (LaTeX Fill)
Perform \.sas$ auto-insertion? (y or n) █
```

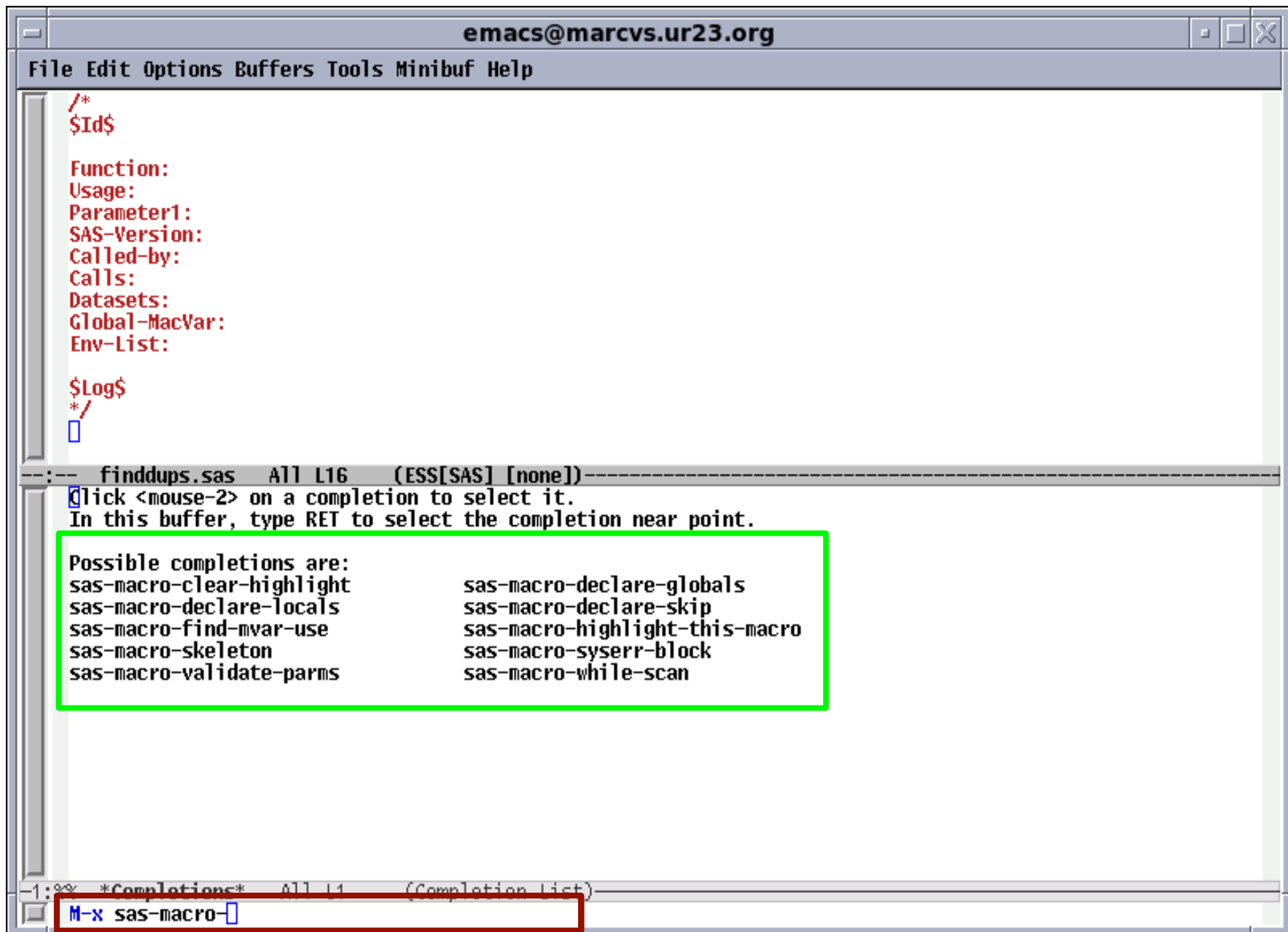
Basic Emacs features: automatic template insertion



Basic Emacs features: command completion



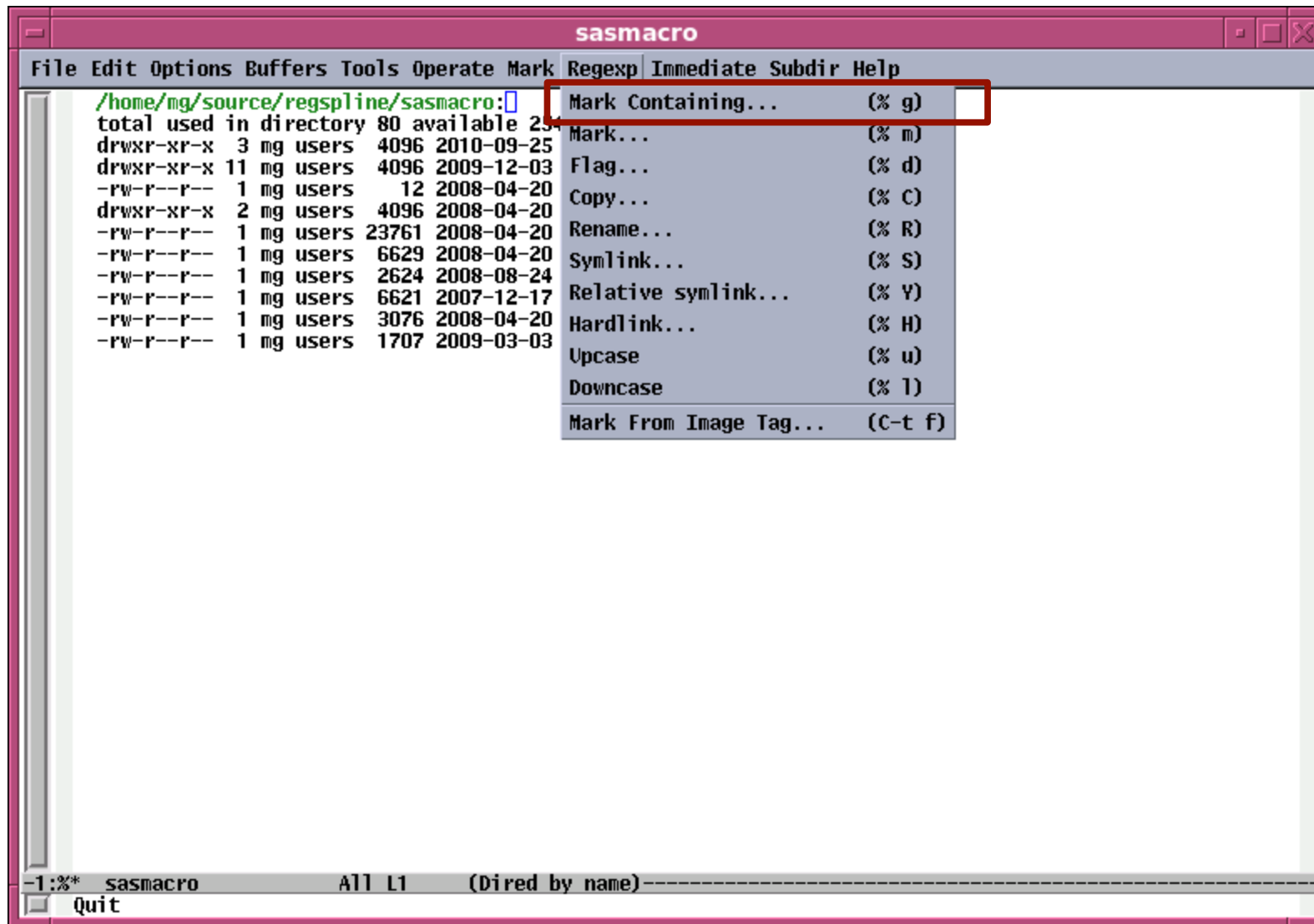
Basic Emacs features: command completion



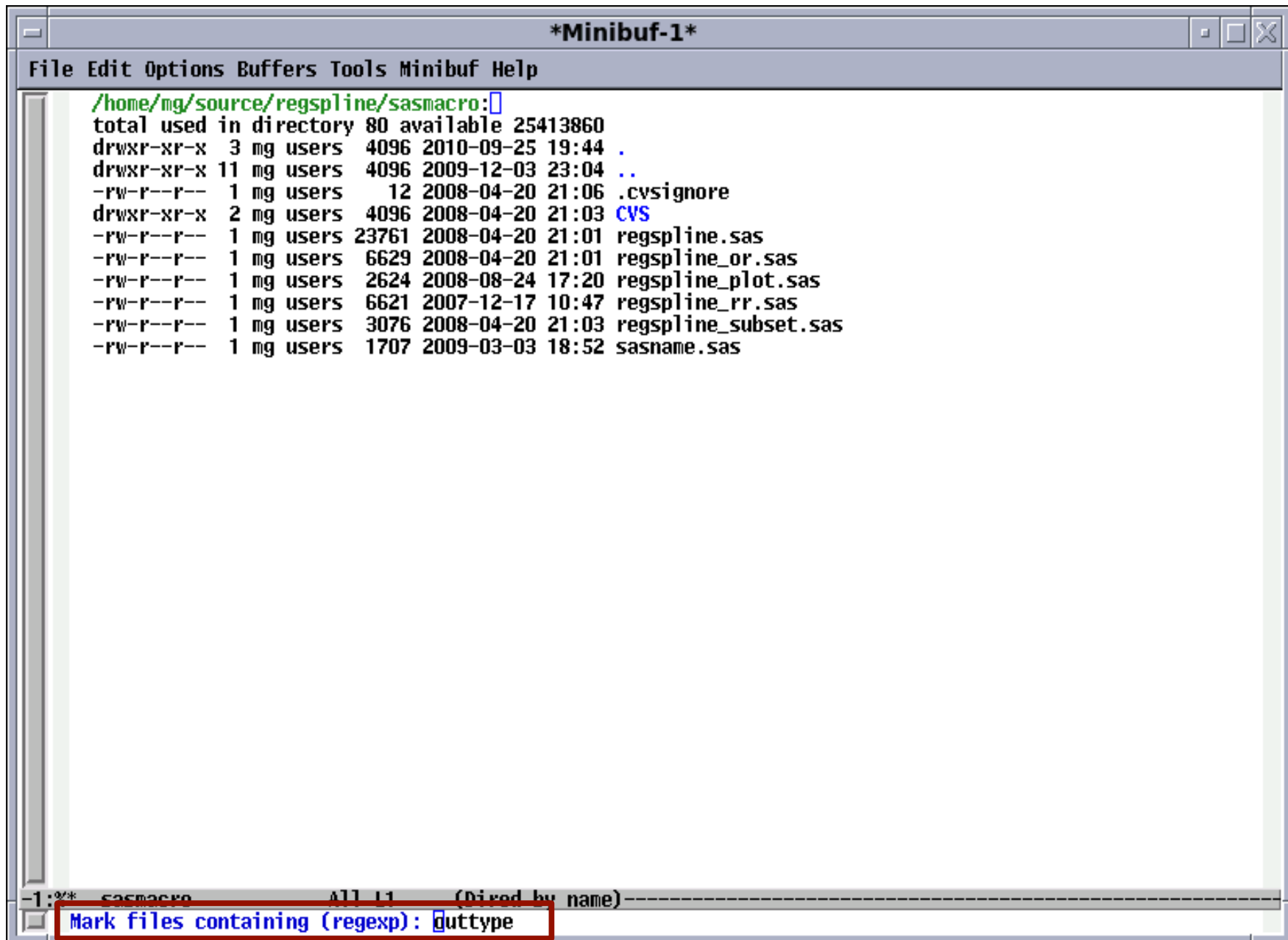
GNU Emacs – the extensible programmer's editor

- Examples of what emacs does really well:
 - process multiple files
 - modes for almost all programming languages
 - ease of extension
 - built-in documentation
 - version control interface

Processing multiple files: search



Processing multiple files: search

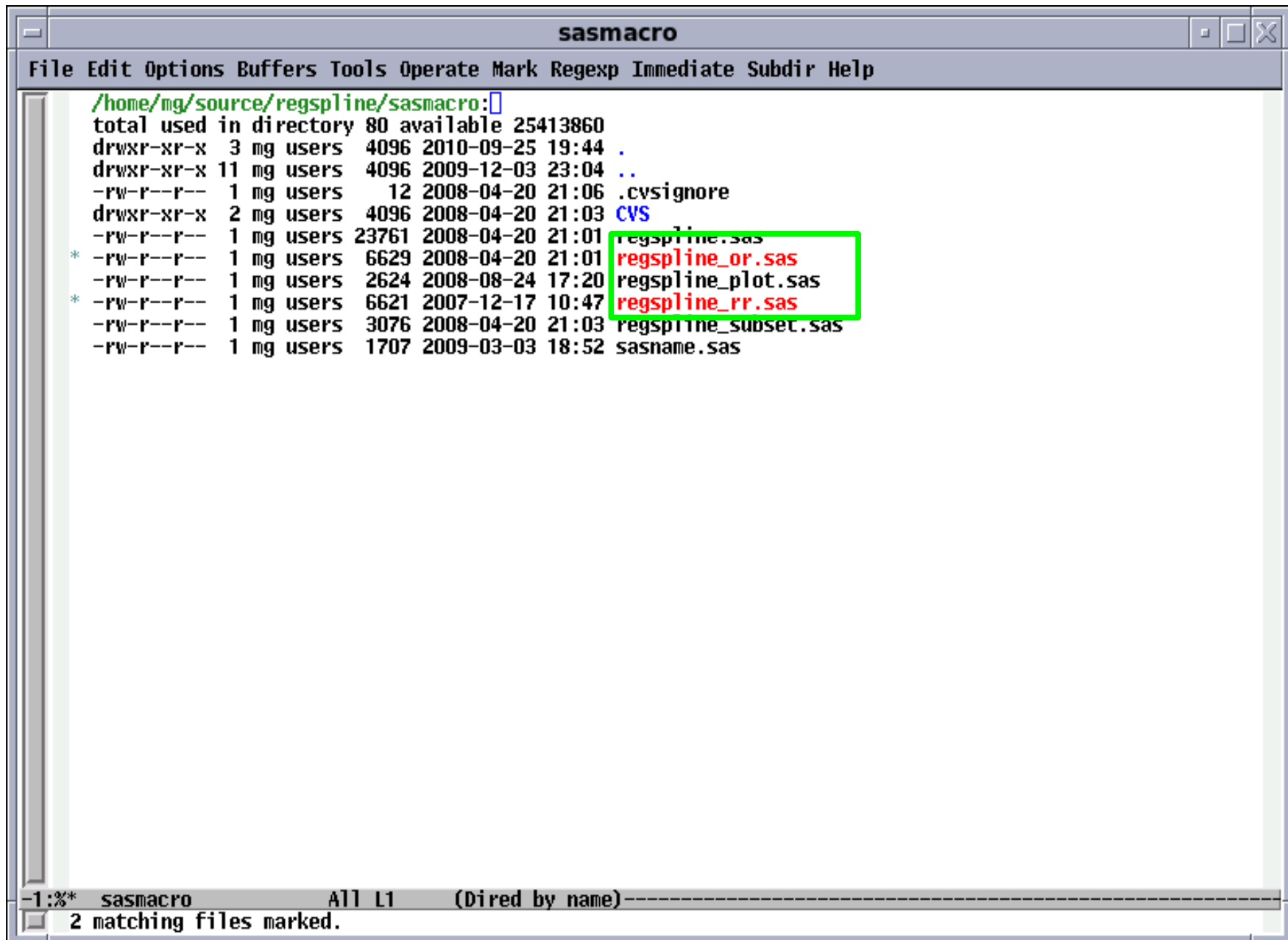


```
File Edit Options Buffers Tools Minibuf Help

/home/mg/source/regspline/sasmacro:
total used in directory 80 available 25413860
drwxr-xr-x  3 mg users  4096 2010-09-25 19:44 .
drwxr-xr-x 11 mg users  4096 2009-12-03 23:04 ..
-rw-r--r--  1 mg users   12 2008-04-20 21:06 .cvsignore
drwxr-xr-x  2 mg users  4096 2008-04-20 21:03 CVS
-rw-r--r--  1 mg users 23761 2008-04-20 21:01 regspline.sas
-rw-r--r--  1 mg users  6629 2008-04-20 21:01 regspline_or.sas
-rw-r--r--  1 mg users  2624 2008-08-24 17:20 regspline_plot.sas
-rw-r--r--  1 mg users  6621 2007-12-17 10:47 regspline_rr.sas
-rw-r--r--  1 mg users  3076 2008-04-20 21:03 regspline_subset.sas
-rw-r--r--  1 mg users  1707 2009-03-03 18:52 sasname.sas

-1: %* sasmacro All 11 (Dired by name)
Mark files containing (regexp): gutfype
```

Processing multiple files: search

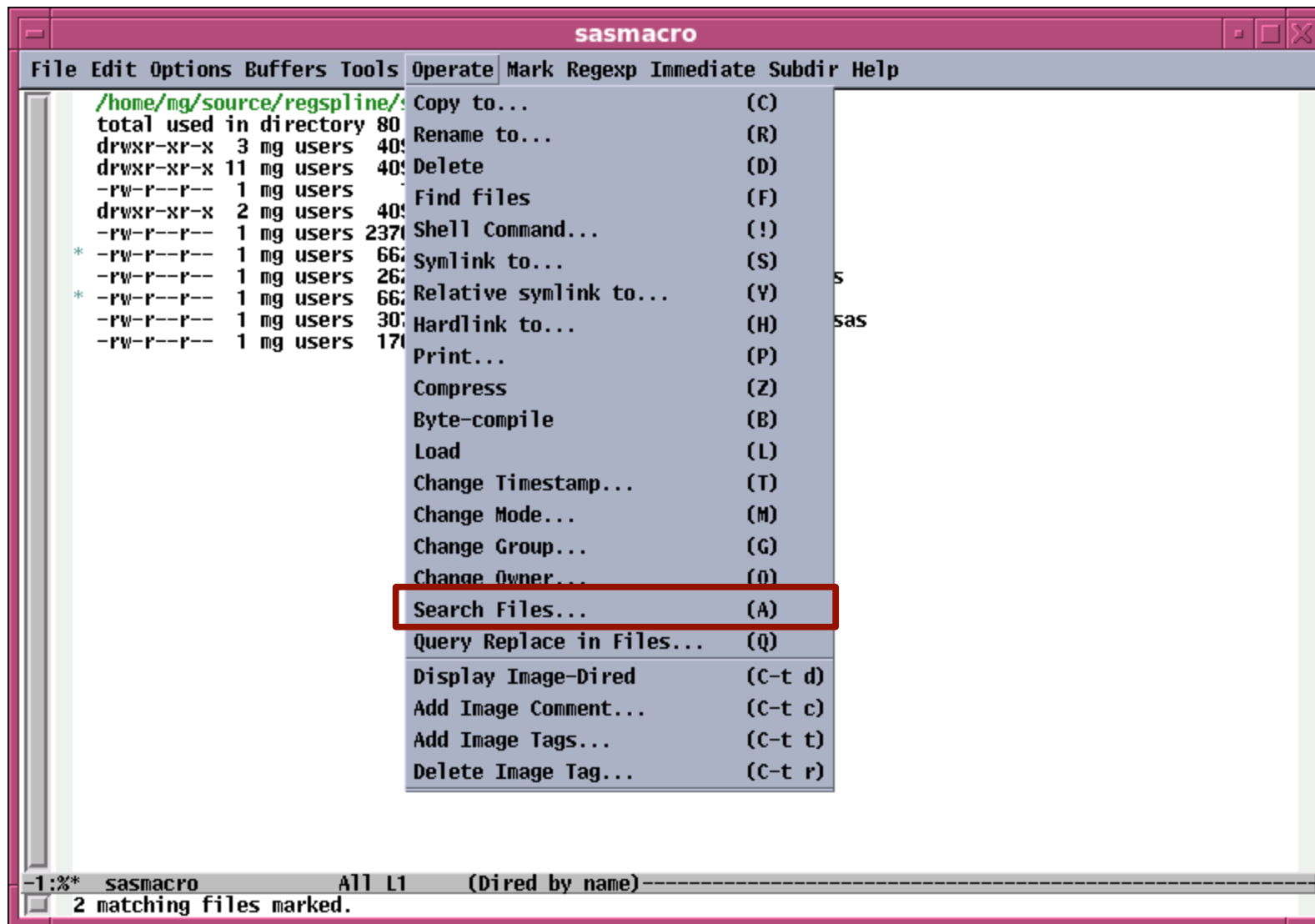


The screenshot shows a window titled "sasmacro" with a menu bar (File, Edit, Options, Buffers, Tools, Operate, Mark, Regexp, Immediate, Subdir, Help). The main text area displays a directory listing for the path `/home/mg/source/regspline/sasmacro:`. The listing shows various files and directories with their permissions, sizes, owners, and timestamps. Two files, `regspline_or.sas` and `regspline_rr.sas`, are highlighted in red text and enclosed in a green rectangular box. The status bar at the bottom indicates the search criteria as `-1:;%* sasmacro All L1 (Dired by name)` and shows that 2 matching files are marked.

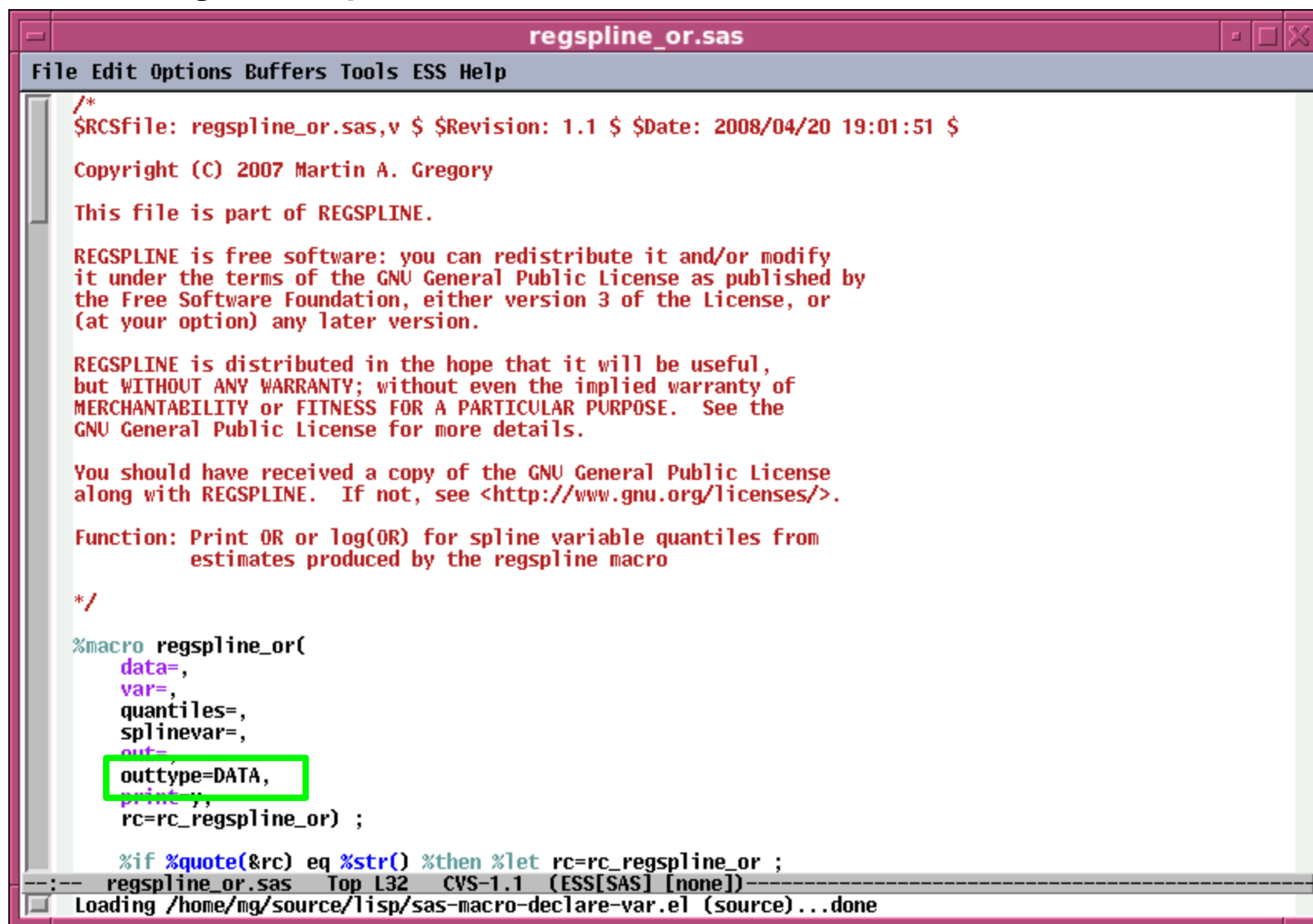
```
/home/mg/source/regspline/sasmacro:
total used in directory 80 available 25413860
drwxr-xr-x  3 mg users  4096 2010-09-25 19:44 .
drwxr-xr-x 11 mg users  4096 2009-12-03 23:04 ..
-rw-r--r--  1 mg users   12 2008-04-20 21:06 .cvsignore
drwxr-xr-x  2 mg users  4096 2008-04-20 21:03 CVS
-rw-r--r--  1 mg users 23761 2008-04-20 21:01 regspline.sas
* -rw-r--r--  1 mg users  6629 2008-04-20 21:01 regspline_or.sas
-rw-r--r--  1 mg users  2624 2008-08-24 17:20 regspline_plot.sas
* -rw-r--r--  1 mg users  6621 2007-12-17 10:47 regspline_rr.sas
-rw-r--r--  1 mg users  3076 2008-04-20 21:03 regspline_subset.sas
-rw-r--r--  1 mg users  1707 2009-03-03 18:52 sasname.sas
```

-1:;%* sasmacro All L1 (Dired by name)
2 matching files marked.

Processing multiple files: search



Processing multiple files: search



```
regspline_or.sas
File Edit Options Buffers Tools ESS Help

/*
$RCSfile: regspline_or.sas,v $ $Revision: 1.1 $ $Date: 2008/04/20 19:01:51 $

Copyright (C) 2007 Martin A. Gregory

This file is part of REGSPLINE.

REGSPLINE is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

REGSPLINE is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with REGSPLINE. If not, see <http://www.gnu.org/licenses/>.

Function: Print OR or log(OR) for spline variable quantiles from
         estimates produced by the regspline macro

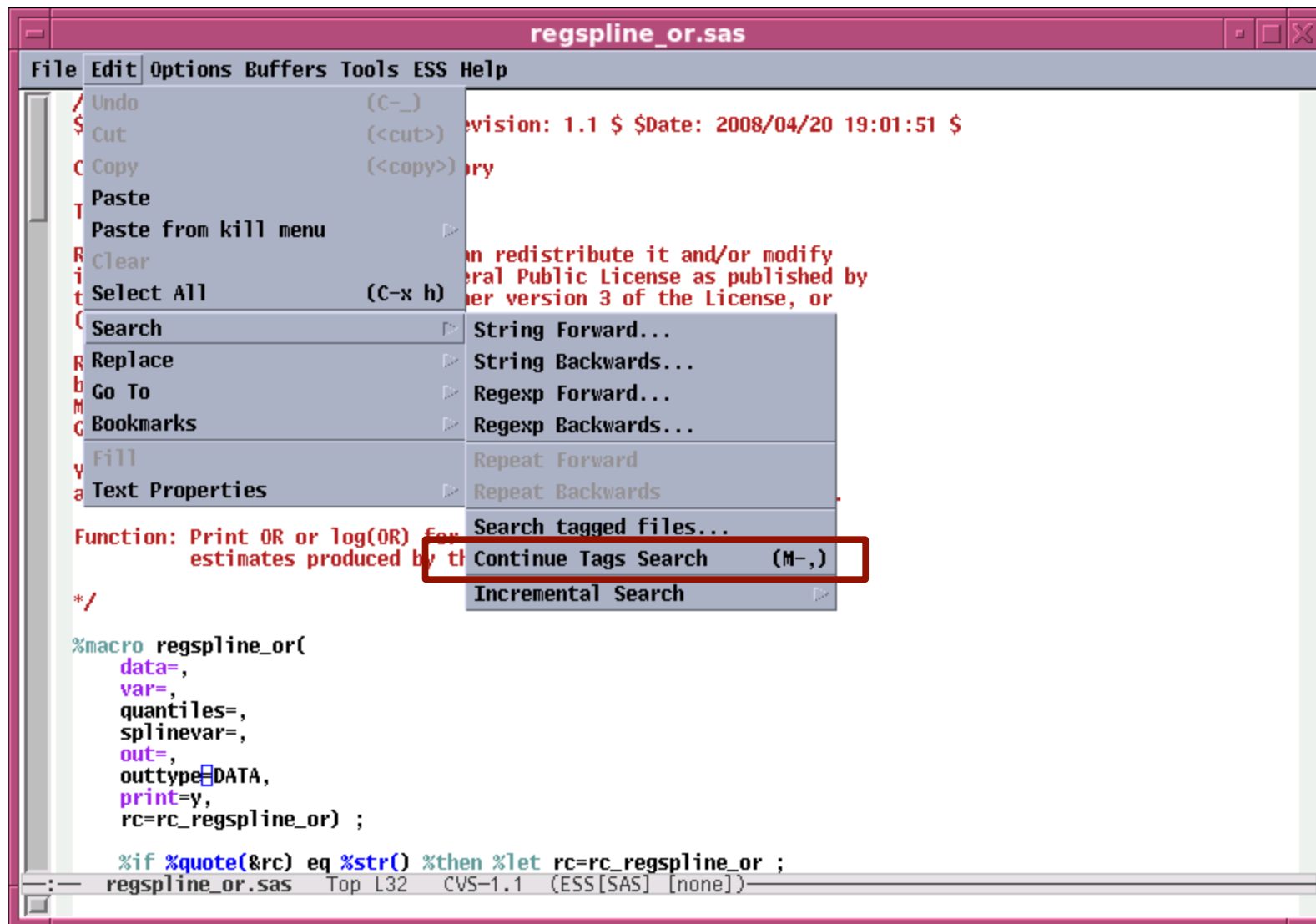
*/

%macro regspline_or(
  data=,
  var=,
  quantiles=,
  splinevar=,
  out=,
  outtype=DATA,
  print=,
  rc=rc_regspline_or) ;

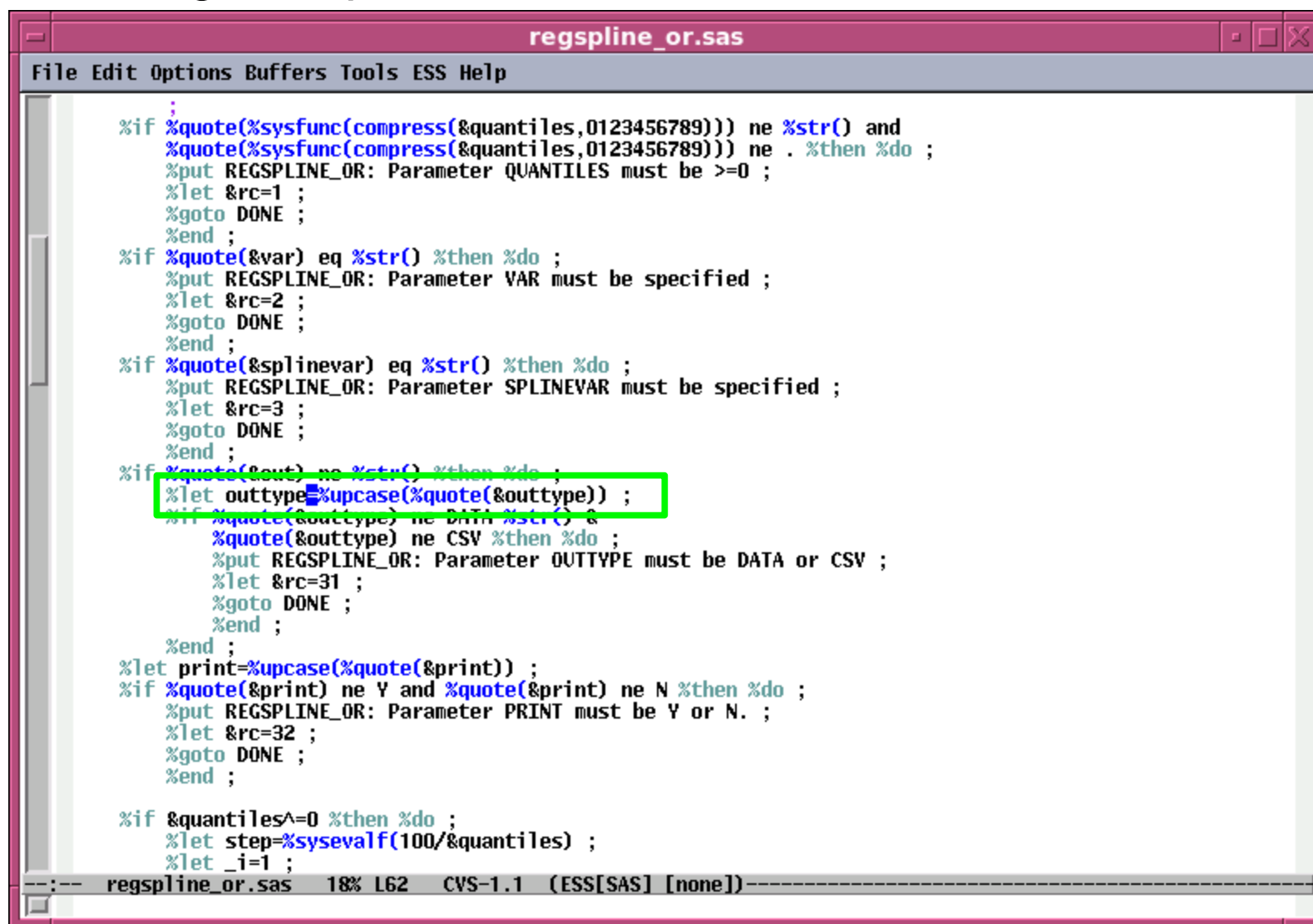
  %if %quote(&rc) eq %str() %then %let rc=rc_regspline_or ;

--:-- regspline_or.sas Top L32 CVS-1.1 (ESS[SAS] [none])
Loading /home/mg/source/lisp/sas-macro-declare-var.el (source)...done
```

Processing multiple files: search



Processing multiple files: search



```
regspline_or.sas
File Edit Options Buffers Tools ESS Help

;
%if %quote(%sysfunc(compress(&quantiles,0123456789))) ne %str() and
%quote(%sysfunc(compress(&quantiles,0123456789))) ne . %then %do ;
%put REGSPLINE_OR: Parameter QUANTILES must be >=0 ;
%let &rc=1 ;
%goto DONE ;
%end ;
%if %quote(&var) eq %str() %then %do ;
%put REGSPLINE_OR: Parameter VAR must be specified ;
%let &rc=2 ;
%goto DONE ;
%end ;
%if %quote(&splinevar) eq %str() %then %do ;
%put REGSPLINE_OR: Parameter SPLINEVAR must be specified ;
%let &rc=3 ;
%goto DONE ;
%end ;
%if %quote(&out) ne %str() %then %do ;
%let outtype=%upcase(%quote(&outtype));
%if %quote(&outtype) ne DATA %str() &
%quote(&outtype) ne CSV %then %do ;
%put REGSPLINE_OR: Parameter OUTTYPE must be DATA or CSV ;
%let &rc=31 ;
%goto DONE ;
%end ;
%end ;
%let print=%upcase(%quote(&print));
%if %quote(&print) ne Y and %quote(&print) ne N %then %do ;
%put REGSPLINE_OR: Parameter PRINT must be Y or N. ;
%let &rc=32 ;
%goto DONE ;
%end ;

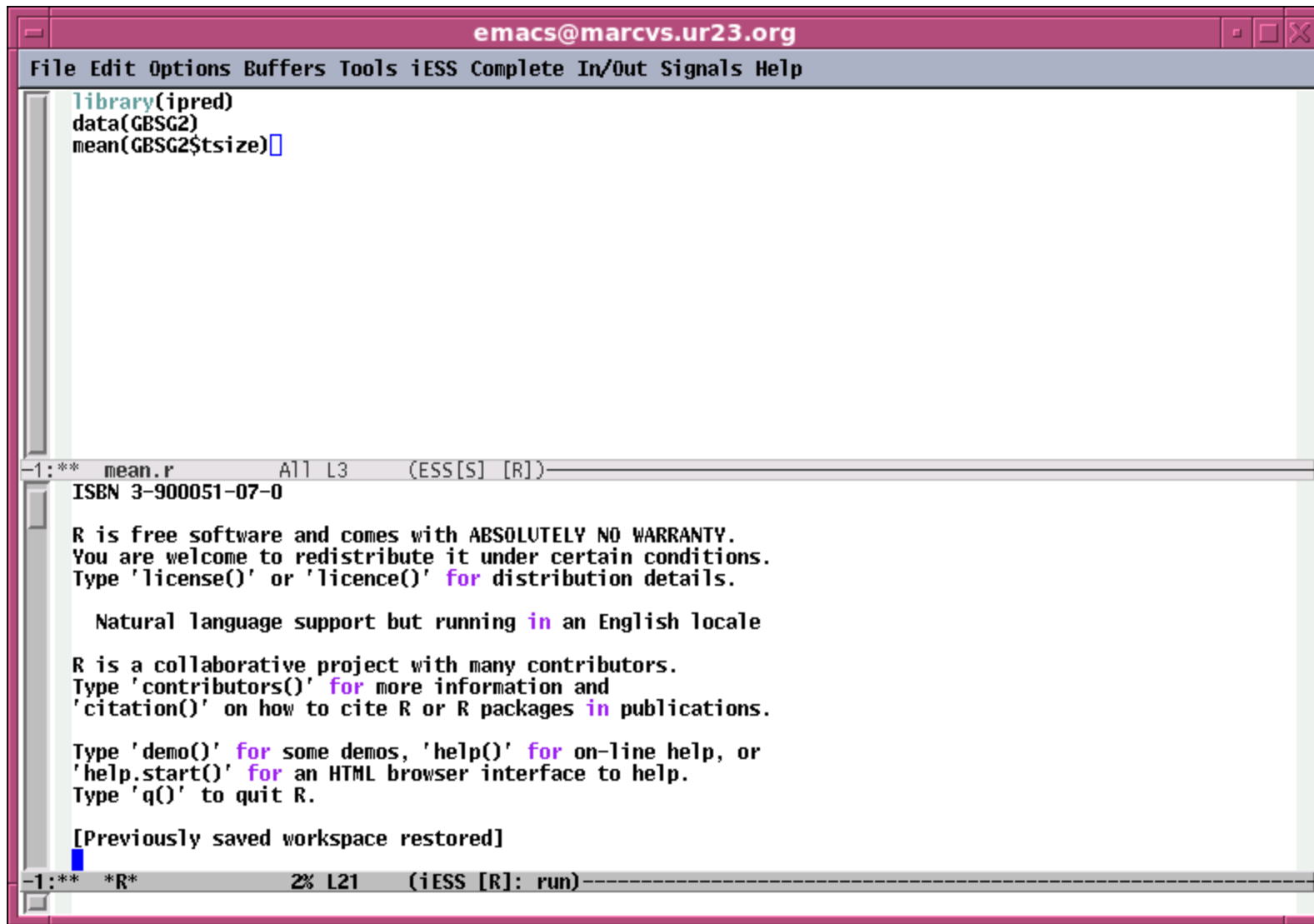
%if &quantiles^=0 %then %do ;
%let step=%sysvalf(100/&quantiles);
%let _i=1 ;

--:-- regspline_or.sas 18% L62 CVS-1.1 (ESS[SAS] [none])
```

Emacs Speaks Statistics - ESS

- development environment for statistical packages
- supports R, SAS, S-PLUS, Stata, SPSS
- uniform interface across different software packages
- syntax highlighting
- syntactically appropriate indentation
- cursor movement by function, statement, step, etc.
- auto-completion of file names and language keywords
- formatting of comment sections
- code execution controlled from Emacs

ESS: code execution



The screenshot shows the ESS (Erlang Shell) interface. The top window is titled 'emacs@marcv.s.ur23.org' and contains a menu bar with 'File Edit Options Buffers Tools iESS Complete In/Out Signals Help'. The main text area shows the following R code:

```
library(ipred)
data(GBSG2)
mean(GBSG2$size)
```

Below the code, the output of the execution is displayed. The first line is '-1:** mean.r' followed by 'A11 L3 (ESS[S] [R])'. The output text is:

ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

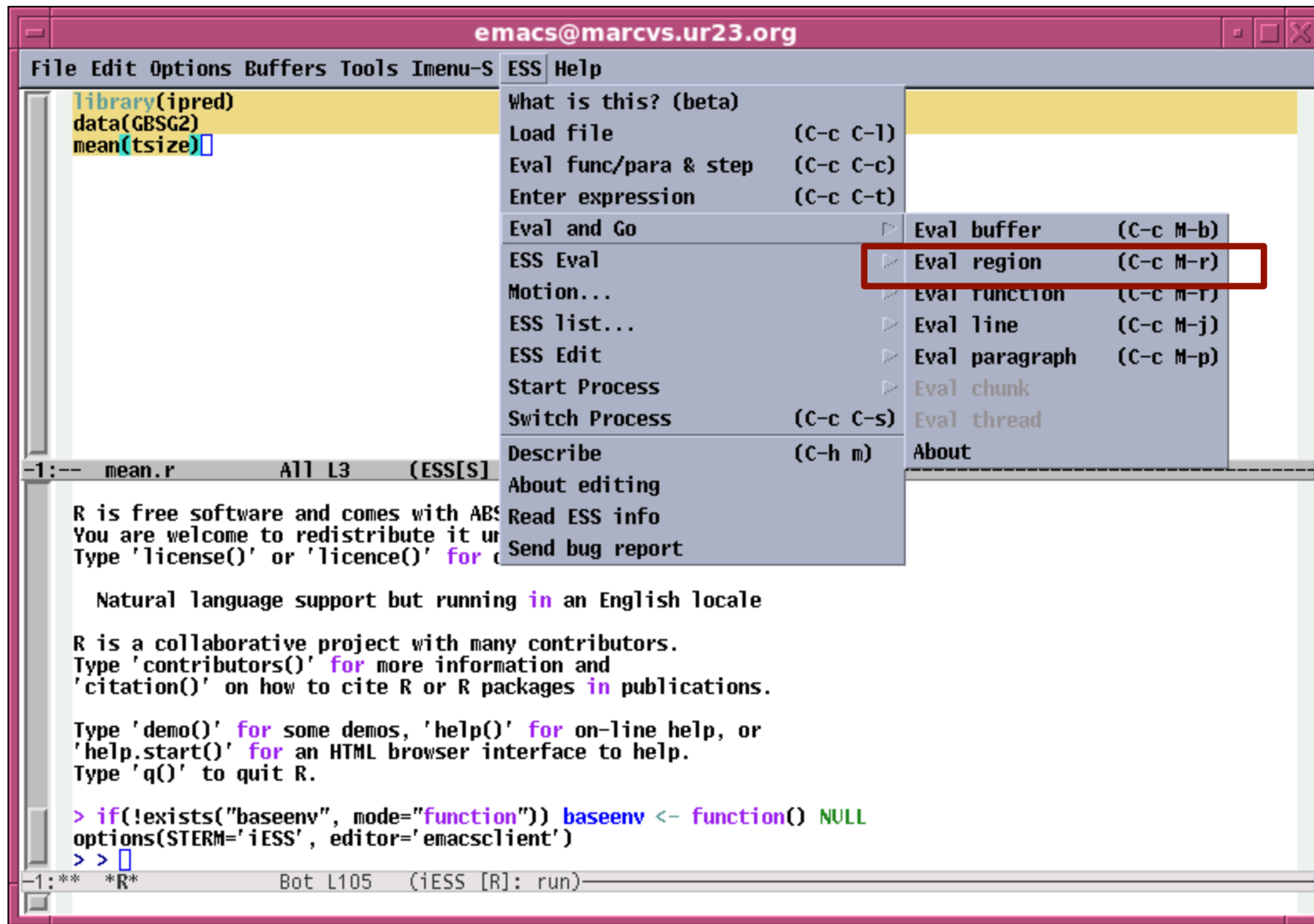
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

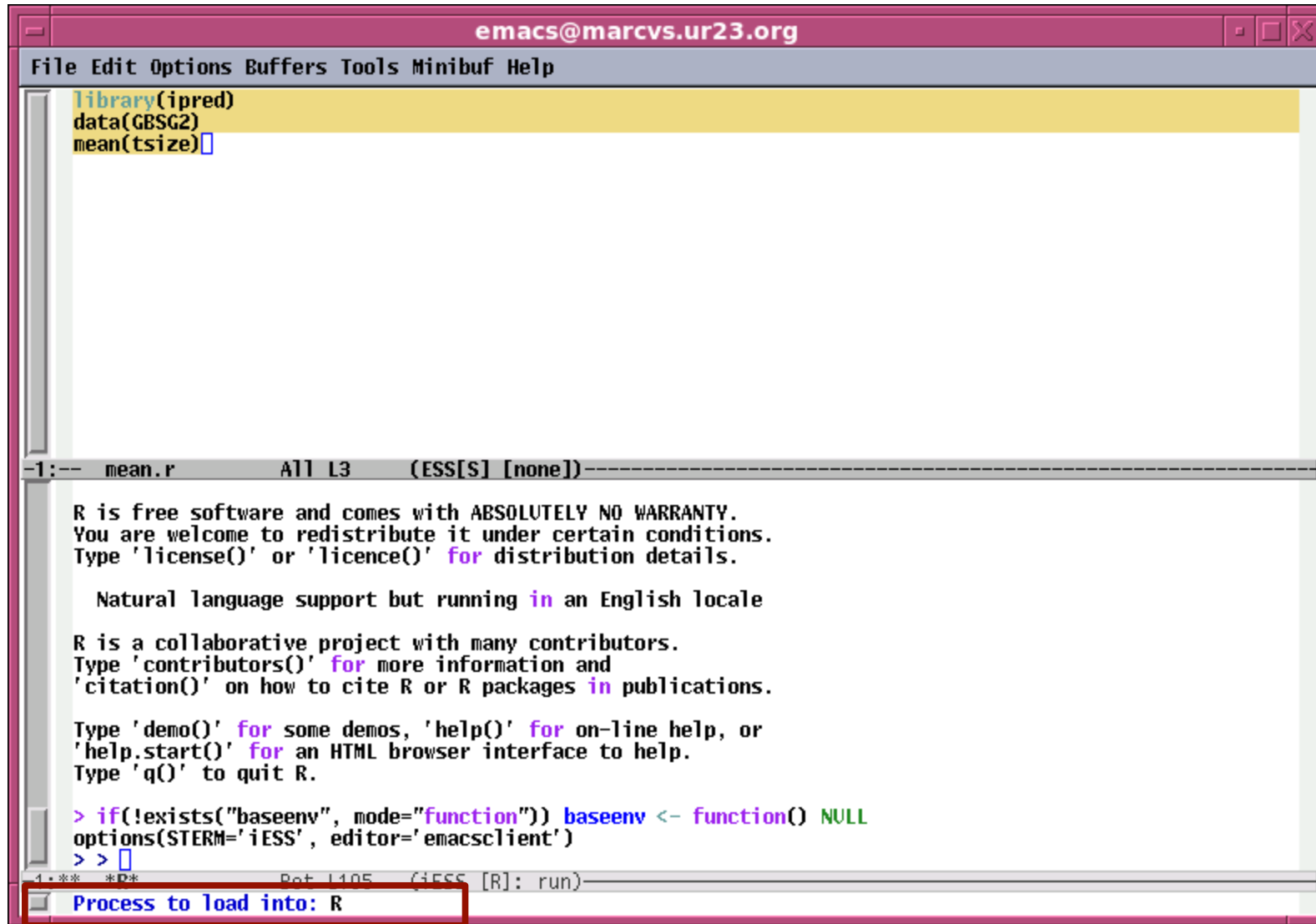
[Previously saved workspace restored]

The bottom status bar shows '-1:** *R*' followed by '2% L21 (iESS [R]: run)'.

ESS: code execution



ESS: code execution



The screenshot shows the ESS interface within an Emacs window. The window title is 'emacs@marcv.s.ur23.org'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Minibuf', and 'Help'. The main text area contains the following R code:

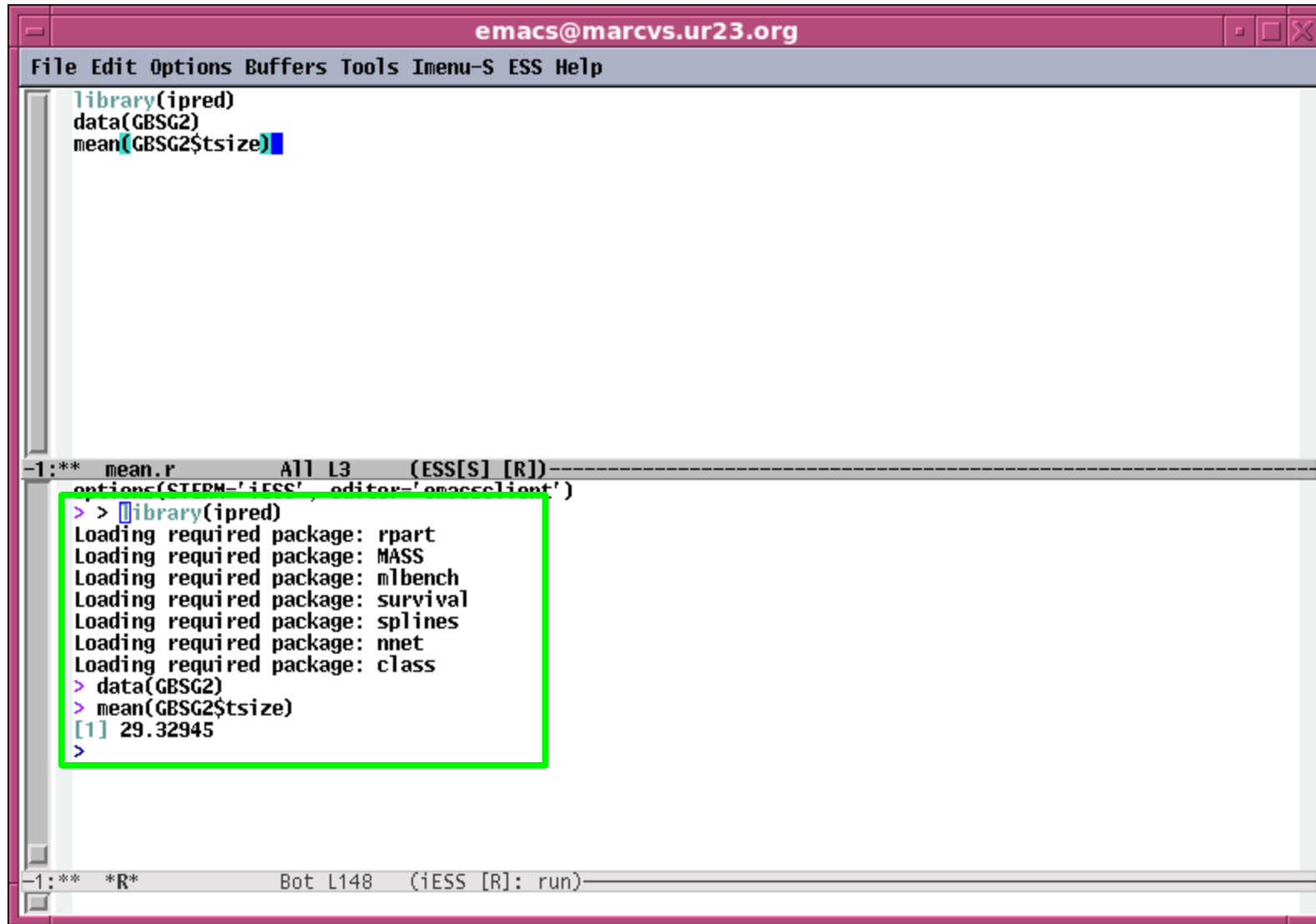
```
library(ipred)
data(GBSG2)
mean(tsize)
```

Below the code area, a status bar shows '-1:-- mean.r All L3 (ESS[S] [none])'. The output area displays the R startup message, including the license and help information, followed by the execution of the code:

```
> if(!exists("baseenv", mode="function")) baseenv <- function() NULL
options(STERM='iESS', editor='emacsclient')
> >
```

At the bottom, a red box highlights the text 'Process to load into: R'.

ESS: code execution



The screenshot shows the ESS interface within an Emacs window. The window title is 'emacs@marcv.s.ur23.org'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Imenu-S', 'ESS', and 'Help'. The main text area contains the following R code:

```
library(ipred)
data(GBSG2)
mean(GBSG2$size)
```

Below the main text area is a console window. The console title is '-1:** mean.r All L3 (ESS[S] [R])'. The console output shows the execution of the code, with a green box highlighting the results:

```
> > library(ipred)
Loading required package: rpart
Loading required package: MASS
Loading required package: mlbench
Loading required package: survival
Loading required package: splines
Loading required package: nnet
Loading required package: class
> data(GBSG2)
> mean(GBSG2$size)
[1] 29.32945
>
```

At the bottom of the console window, the status bar shows '-1:** *R* Bot L148 (iESS [R]: run)'.

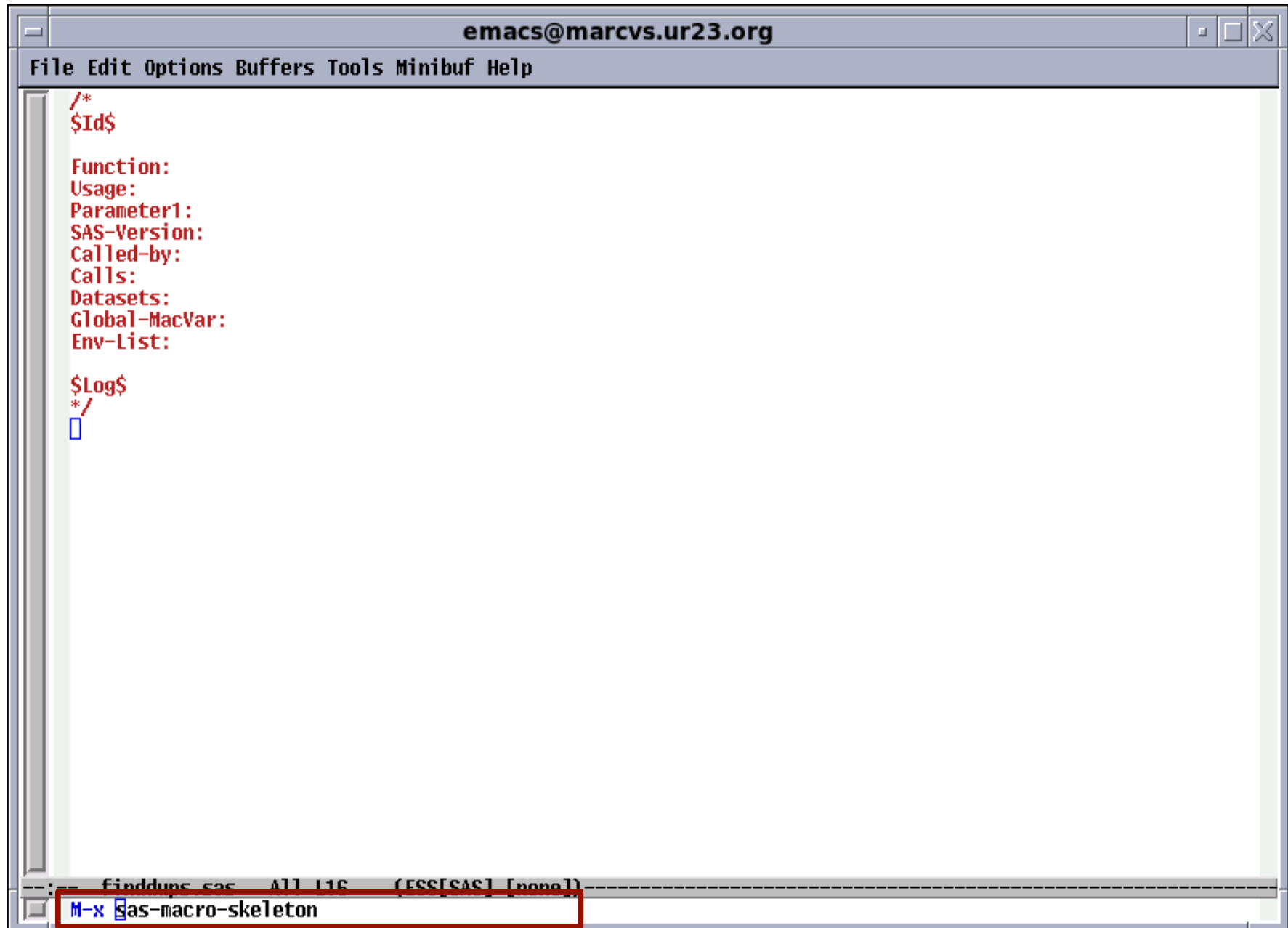
Extensions for robust macro techniques

- Template insertion: macro skeleton
- Template insertion: parameter validation
- Checking for undeclared macro variables
- Highlighting the current macro
- Template insertion: checking data and proc step return codes

Extensions for robust macro techniques

- Template insertion: macro skeleton
- Template insertion: parameter validation
- Checking for undeclared macro variables
- Highlighting the current macro
- Template insertion: checking data and proc step return codes

Insert SAS macro template

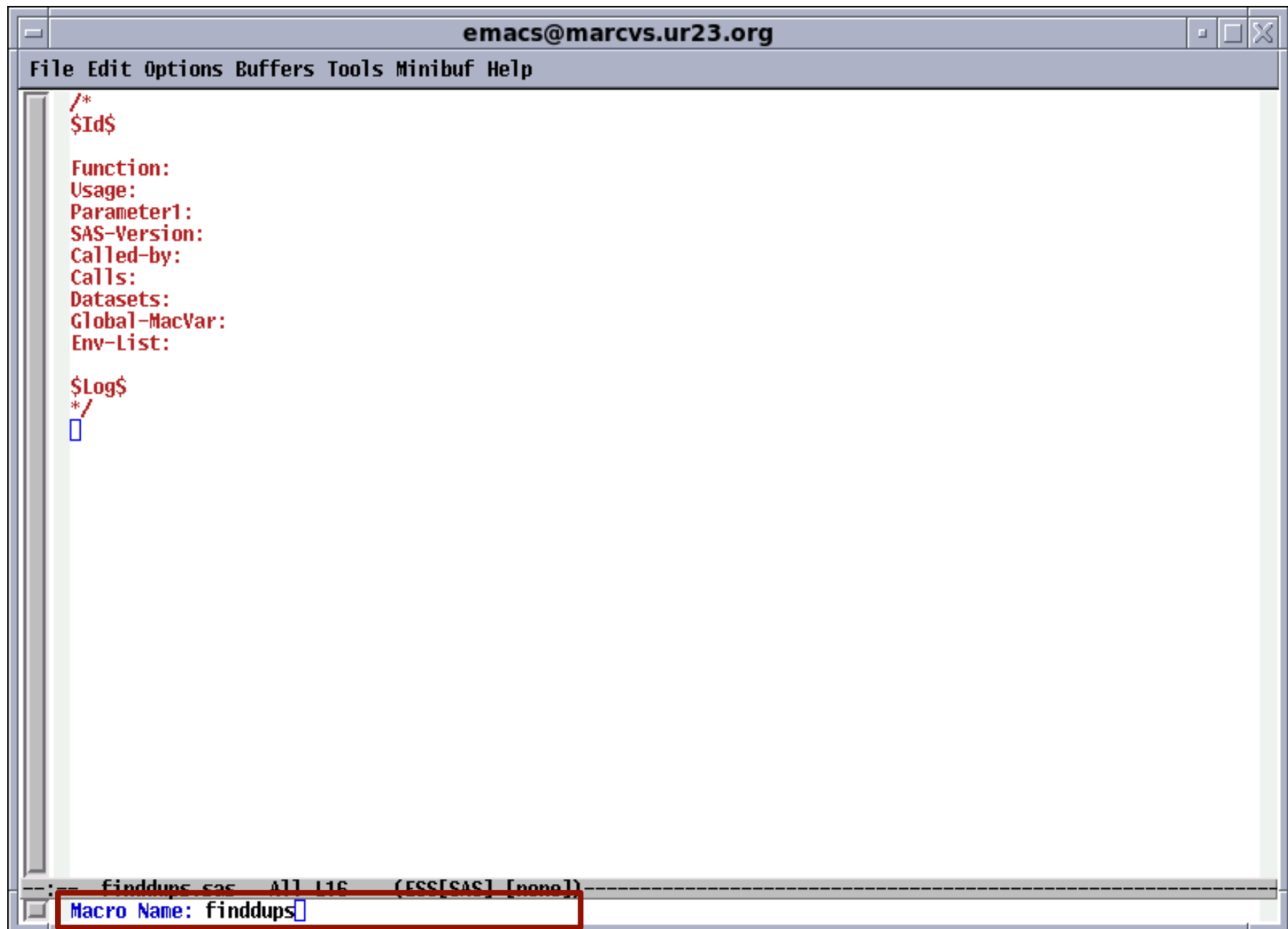


The screenshot shows an Emacs window titled "emacs@marcv.s.ur23.org". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Minibuf", and "Help". The main text area contains a SAS macro template with the following text:

```
/*  
$Id$  
  
Function:  
Usage:  
Parameter1:  
SAS-Version:  
Called-by:  
Calls:  
Datasets:  
Global-MacVar:  
Env-List:  
  
$Log$  
*/  
[
```

The status bar at the bottom shows "finddups.cas All 116 (ESS[SAS] [none])". A red box highlights the command "M-x sas-macro-skeleton" in the minibuffer.

Insert SAS macro template – enter macro name



Insert SAS macro template

emacs@marcv.s.uu.se

File Edit Options Buffers Tools ESS Help

```

/*
$Id$

Function:
Usage:
Parameter1:
SAS-Version:
Called-by:
Calls:
Datasets:
Global-MacVar:
Env-List:

$Log$
*/
macro finddups(
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
  %global &rc ;
  %let &rc=9999 ;

  /* Set RC=0: if we arrive here all should be ok;
  %let &rc=0;
%DONE:
%mend;

```

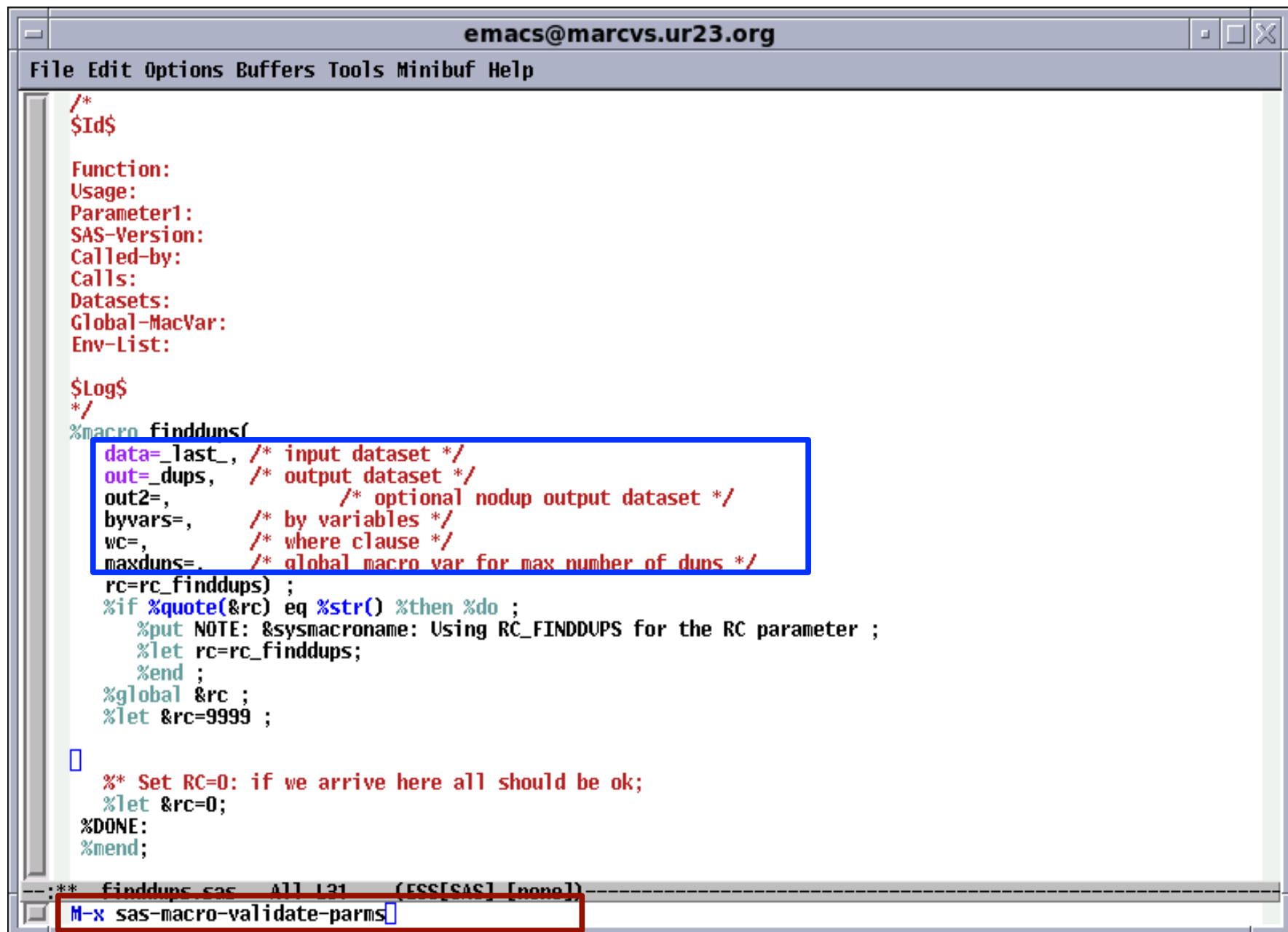
;;** finddups.sas All 116 (ESS[SAS] [none])

You can run the command 'sas-macro-skeleton' with C-c m

Extensions for robust macro techniques

- Template insertion: macro skeleton
- Template insertion: parameter validation
- Checking for undeclared macro variables
- Highlighting the current macro
- Template insertion: checking data and proc step return codes

Insert parameter validation section



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools Minibuf Help

/*
$Id$

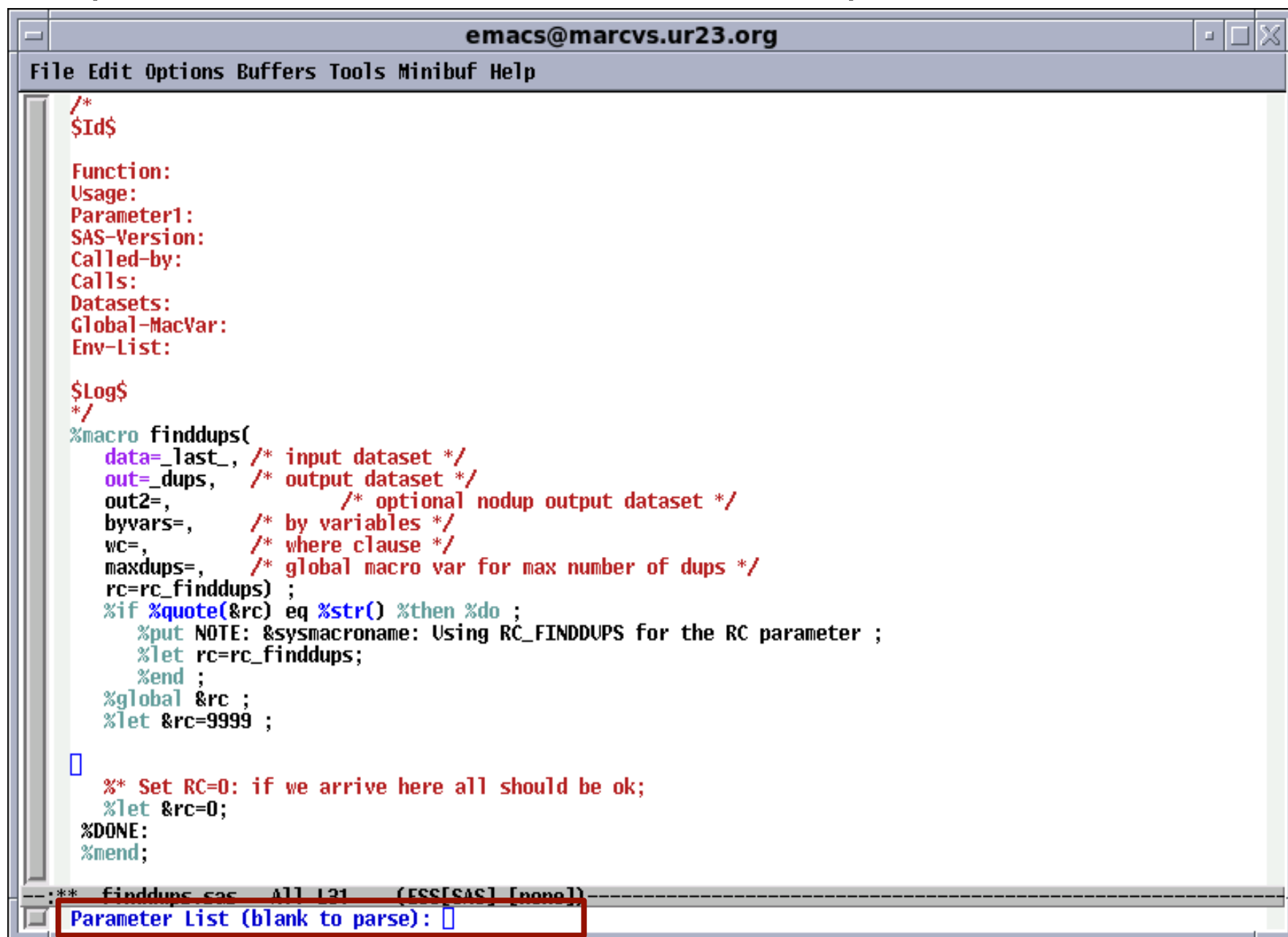
Function:
Usage:
Parameter1:
SAS-Version:
Called-by:
Calls:
Datasets:
Global-MacVar:
Env-List:

$Log$
*/
%macro finddups(
  data=_last_ /* input dataset */
  out=_dups,   /* output dataset */
  out2=,       /* optional nodup output dataset */
  byvars=,     /* by variables */
  wc=,         /* where clause */
  maxdups=,    /* global macro var for max number of dups */
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
  %global &rc ;
  %let &rc=9999 ;

  /* Set RC=0: if we arrive here all should be ok;
  %let &rc=0;
%DONE:
%mend;

** finddups.sas All 131 (ESS[SAS] [none])
M-x sas-macro-validate-params
```

Insert parameter validation section – enter parameter list



```
/*
$Id$

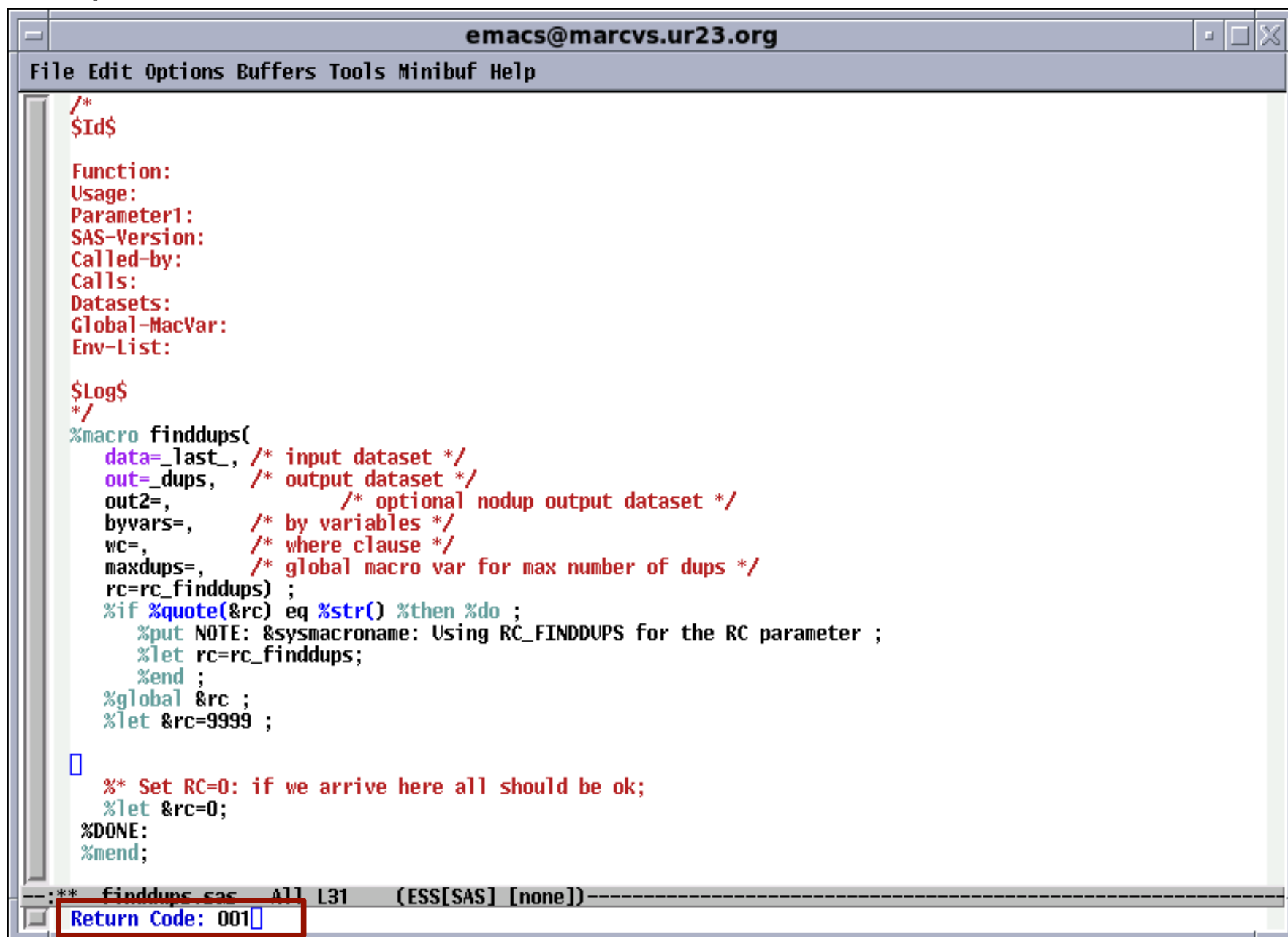
Function:
Usage:
Parameter1:
SAS-Version:
Called-by:
Calls:
Datasets:
Global-MacVar:
Env-List:

$Log$
*/
%macro finddups(
  data=_last_, /* input dataset */
  out=_dups,   /* output dataset */
  out2=,       /* optional nodup output dataset */
  byvars=,     /* by variables */
  wc=,         /* where clause */
  maxdups=,    /* global macro var for max number of dups */
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
  %global &rc ;
  %let &rc=9999 ;

  /* Set RC=0: if we arrive here all should be ok;
  %let &rc=0;
%DONE:
%mend;

** finddups.sas All 131 (ESS[SAS] [none])
Parameter List (blank to parse):
```

Insert parameter validation section – enter return code



```
emacs@marcv.s.ur23.org
File Edit Options Buffers Tools Minibuf Help

/*
$Id$

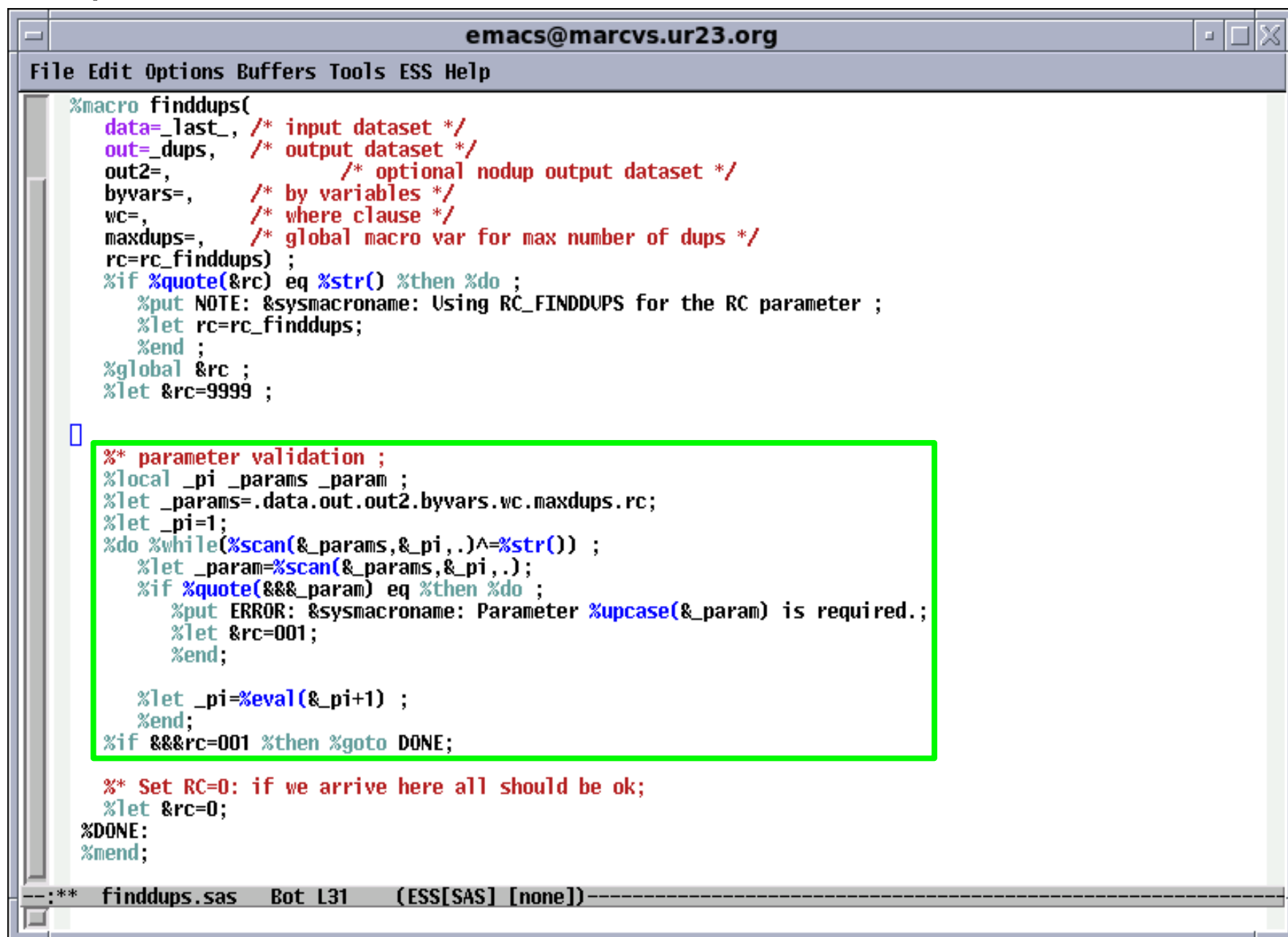
Function:
Usage:
Parameter1:
SAS-Version:
Called-by:
Calls:
Datasets:
Global-MacVar:
Env-List:

$Log$
*/
%macro finddups(
  data=_last_, /* input dataset */
  out=_dups,   /* output dataset */
  out2=,       /* optional nodup output dataset */
  byvars=,     /* by variables */
  wc=,         /* where clause */
  maxdups=,    /* global macro var for max number of dups */
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
  %global &rc ;
  %let &rc=9999 ;

  /* Set RC=0: if we arrive here all should be ok;
  %let &rc=0;
%DONE:
%mend;

** finddups.sas All L31 (ESS[SAS] [none])
Return Code: 001
```

Insert parameter validation section



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools ESS Help

%macro finddups(
  data=_last_, /* input dataset */
  out=_dups,   /* output dataset */
  out2=,       /* optional nodup output dataset */
  byvars=,     /* by variables */
  wc=,         /* where clause */
  maxdups=,    /* global macro var for max number of dups */
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
  %global &rc ;
  %let &rc=9999 ;

  /* parameter validation ;
  %local _pi _params _param ;
  %let _params=.data.out.out2.byvars.wc.maxdups.rc;
  %let _pi=1;
  %do %while(%scan(&_params,&_pi,.)^=%str()) ;
    %let _param=%scan(&_params,&_pi,.);
    %if %quote(&&&_param) eq %then %do ;
      %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
      %let &rc=001;
    %end;

    %let _pi=%eval(&_pi+1) ;
  %end;
  %if &&&rc=001 %then %goto DONE;

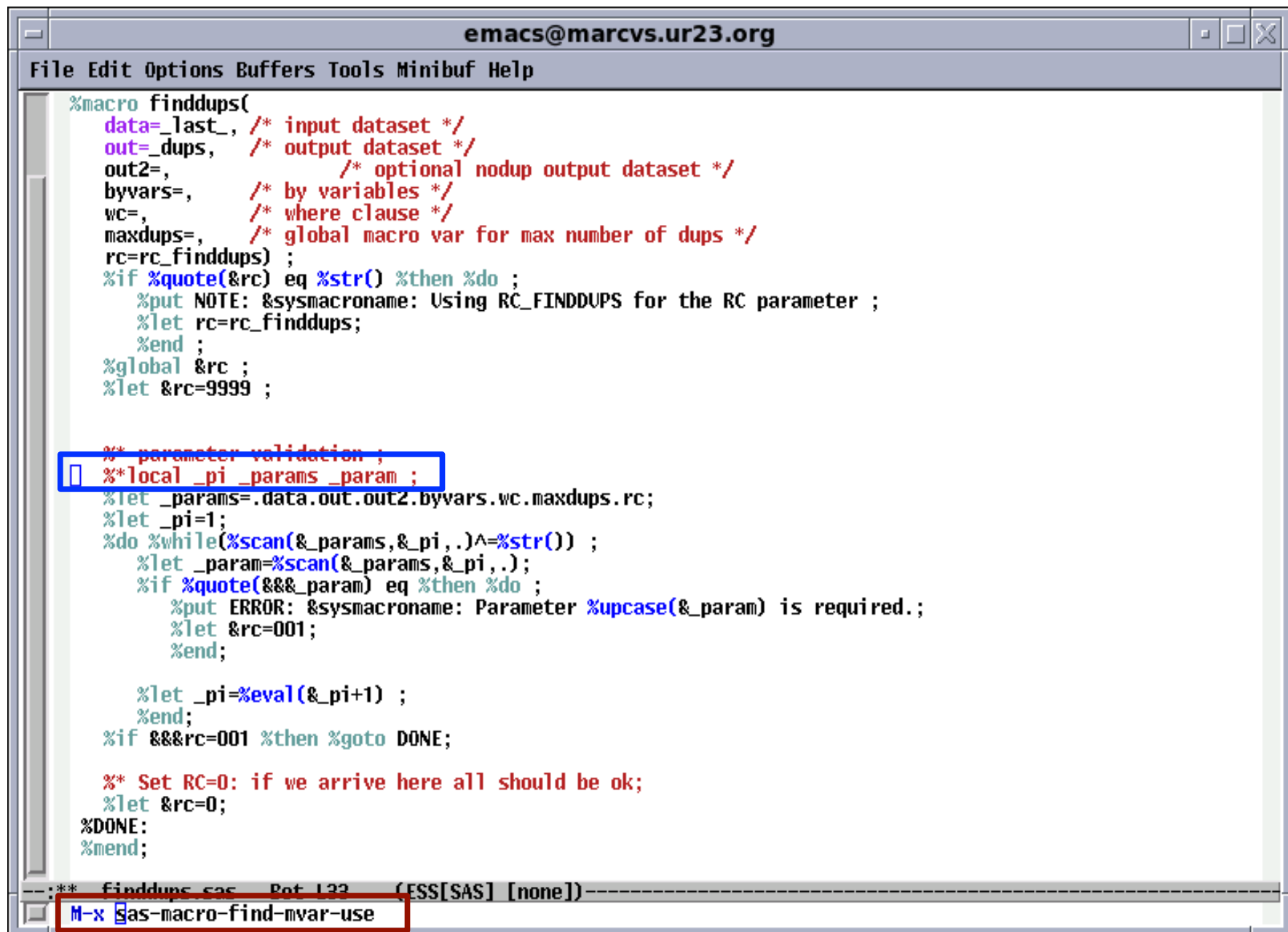
  /* Set RC=0: if we arrive here all should be ok;
  %let &rc=0;
%DONE:
%mend;

--:** finddups.sas Bot L31 (ESS[SAS] [none])
```

Extensions for robust macro techniques

- Template insertion: macro skeleton
- Template insertion: parameter validation
- Checking for undeclared macro variables
- Highlighting the current macro
- Template insertion: checking data and proc step return codes

Check for undeclared variables



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools Minibuf Help

%macro finddups(
  data=_last_ /* input dataset */
  out=_dups, /* output dataset */
  out2=, /* optional nodup output dataset */
  byvars=, /* by variables */
  wc=, /* where clause */
  maxdups=, /* global macro var for max number of dups */
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
  %global &rc ;
  %let &rc=9999 ;

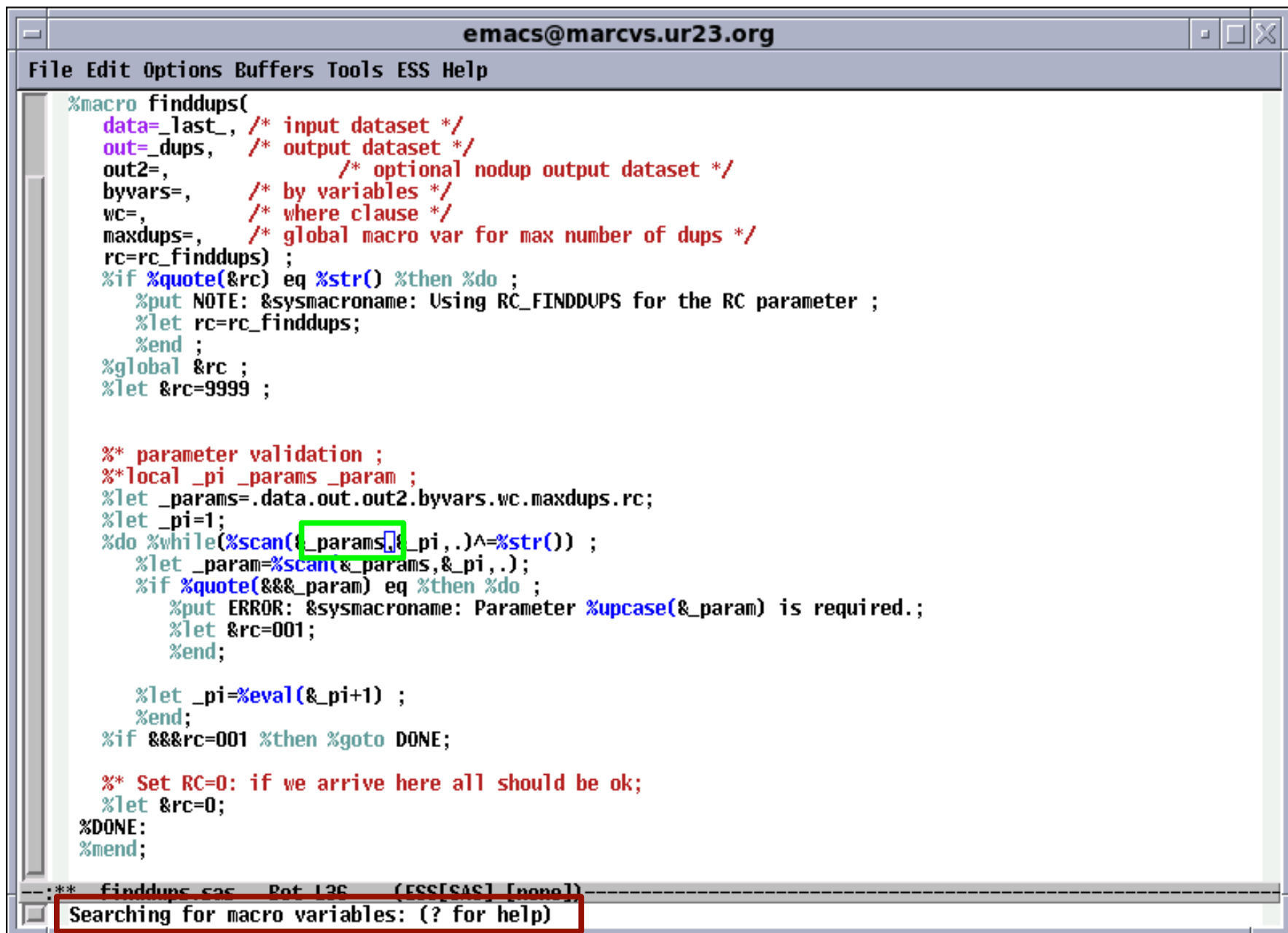
  /* parameter validation ;
  %*local _pi _params _param ;
  %let _params=.data.out.out2.byvars.wc.maxdups.rc;
  %let _pi=1;
  %do %while(%scan(&_params,&_pi,.)^=%str()) ;
    %let _param=%scan(&_params,&_pi,.);
    %if %quote(&&&_param) eq %then %do ;
      %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
      %let &rc=001;
    %end;

    %let _pi=%eval(&_pi+1) ;
  %end;
  %if &&&rc=001 %then %goto DONE;

  /* Set RC=0: if we arrive here all should be ok;
  %let &rc=0;
%DONE:
%mend;

** finddups sas Pot 133 (ESS[SAS] [none])
M-x sas-macro-find-mvar-use
```

Check for undeclared variables – first instance found



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools ESS Help

%macro finddups(
  data=_last_, /* input dataset */
  out=_dups,   /* output dataset */
  out2=,       /* optional nodup output dataset */
  byvars=,     /* by variables */
  wc=,         /* where clause */
  maxdups=,    /* global macro var for max number of dups */
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
  %global &rc ;
  %let &rc=9999 ;

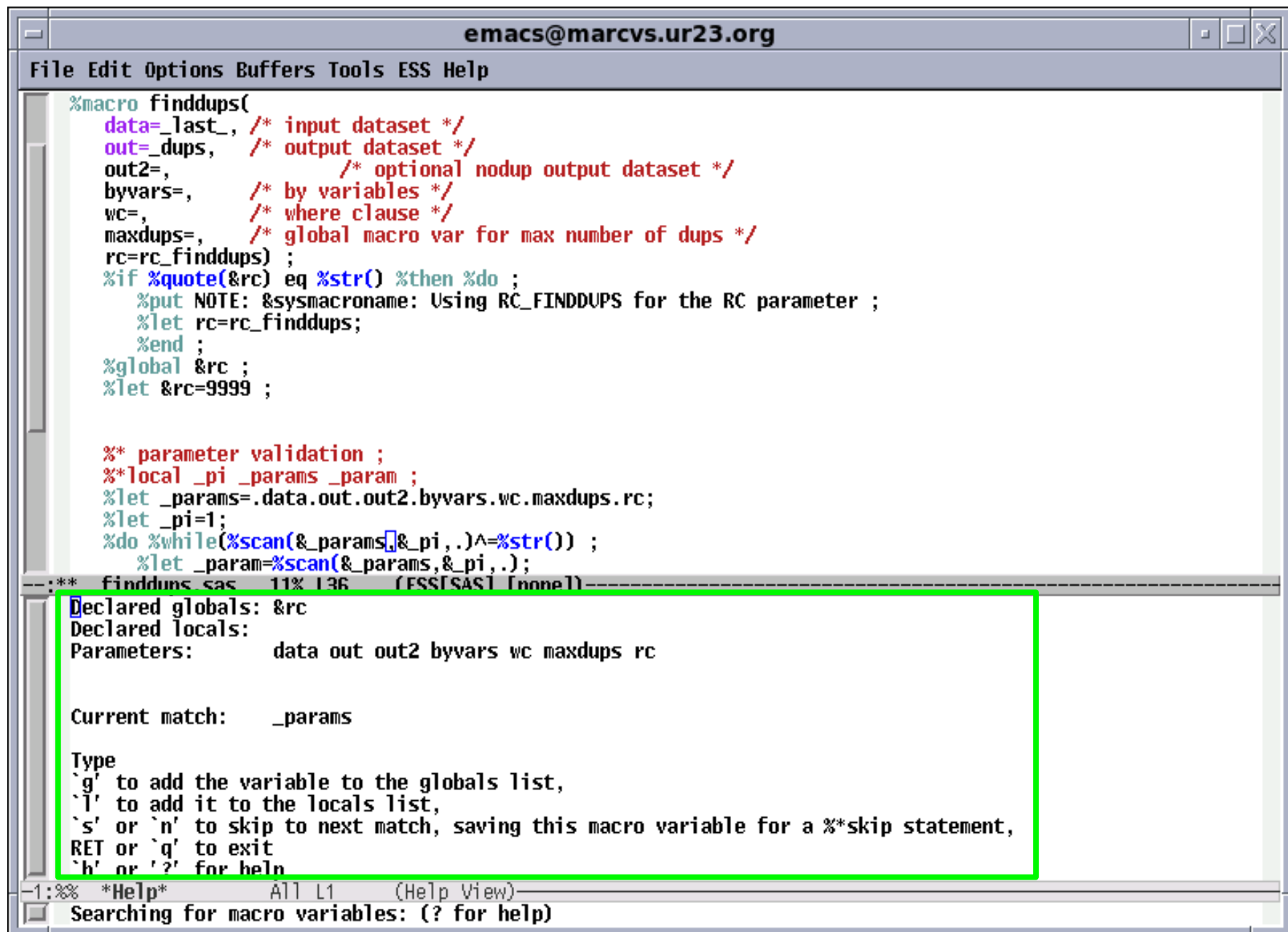
  /* parameter validation ;
  %*local _pi _params _param ;
  %let _params=.data.out.out2.byvars.wc.maxdups.rc;
  %let _pi=1;
  %do %while(%scan(_params,_pi,.)^=%str()) ;
    %let _param=%scan(_params,&_pi,.);
    %if %quote(&&&_param) eq %then %do ;
      %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
      %let &rc=001;
    %end;

    %let _pi=%eval(&_pi+1) ;
  %end;
  %if &&&rc=001 %then %goto DONE;

  /* Set RC=0: if we arrive here all should be ok;
  %let &rc=0;
  %DONE:
  %mend;

** finddups.sas Pot 136 (ESS[SAS] [none])
Searching for macro variables: (? for help)
```

Check for undeclared variables – interactive documentation



The screenshot shows an Emacs window titled 'emacs@marcv.s.ur23.org'. The menu bar includes 'File Edit Options Buffers Tools ESS Help'. The main text area contains SAS macro code for a macro named 'finddups'. The code includes parameter declarations, a conditional check for the 'rc' parameter, and a validation loop. A green rectangular box highlights the interactive documentation for the macro, which lists declared globals, locals, parameters, the current match, and a list of navigation keys.

```
%macro finddups(  
  data=_last_, /* input dataset */  
  out=_dups,   /* output dataset */  
  out2=,       /* optional nodup output dataset */  
  byvars=,     /* by variables */  
  wc=,         /* where clause */  
  maxdups=,    /* global macro var for max number of dups */  
  rc=rc_finddups) ;  
  %if %quote(&rc) eq %str() %then %do ;  
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;  
    %let rc=rc_finddups;  
  %end ;  
  %global &rc ;  
  %let &rc=9999 ;  
  
  /* parameter validation ;  
  /*local _pi _params _param ;  
  %let _params=.data.out.out2.byvars.wc.maxdups.rc;  
  %let _pi=1;  
  %do %while(%scan(&_params,&_pi,.)^=%str()) ;  
    %let _param=%scan(&_params,&_pi,.);  
  %end;  
--:** finddups sas 11% 136 (ESS[SAS] [none])--
```

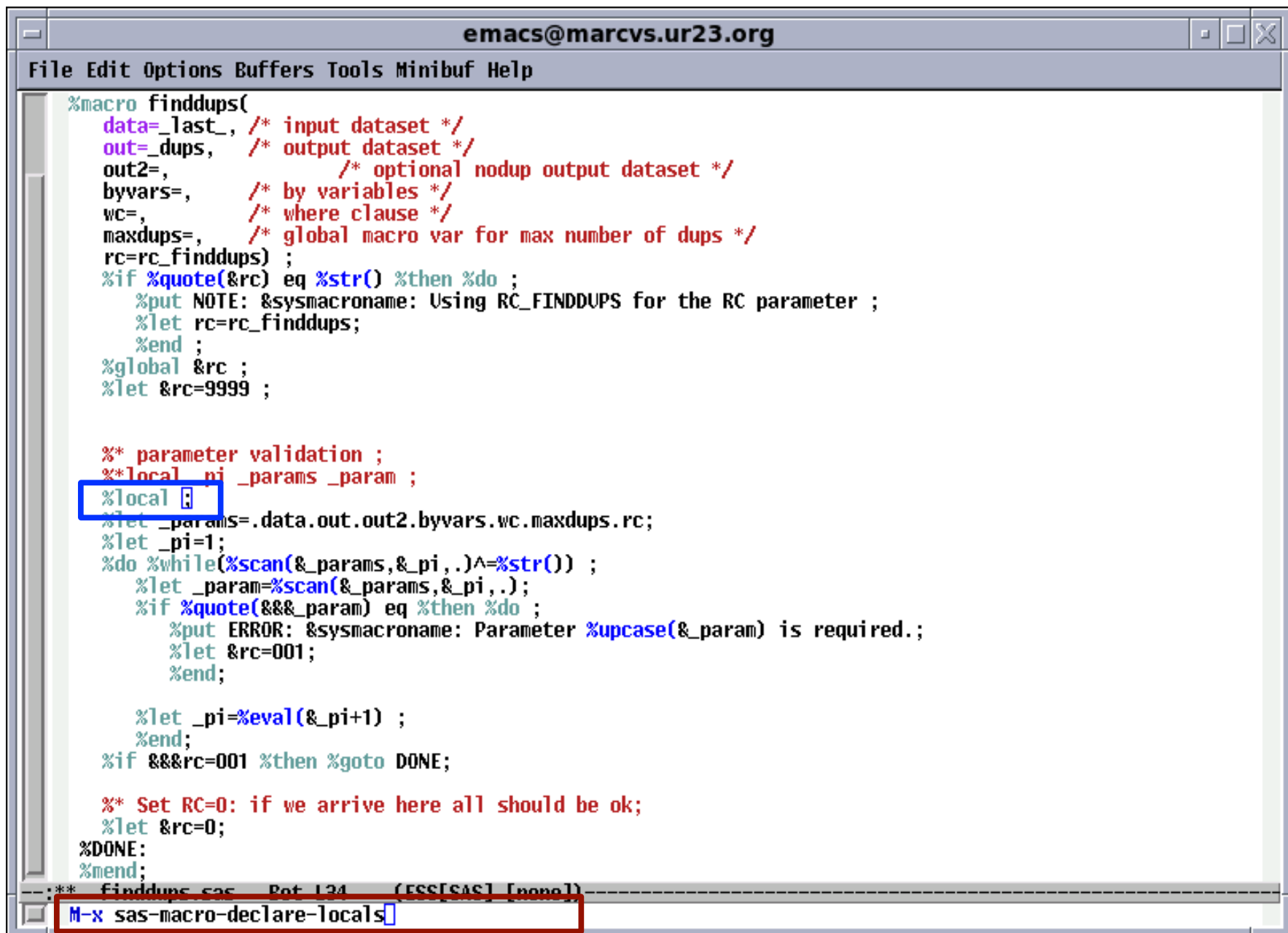
Declared globals: &rc
Declared locals:
Parameters: data out out2 byvars wc maxdups rc

Current match: _params

Type
'g' to add the variable to the globals list,
'l' to add it to the locals list,
's' or 'n' to skip to next match, saving this macro variable for a %*skip statement,
RET or 'q' to exit
'h' or '?' for help

-1:%% *Help* All L1 (Help View)
Searching for macro variables: (? for help)

Check for undeclared variables – insert list



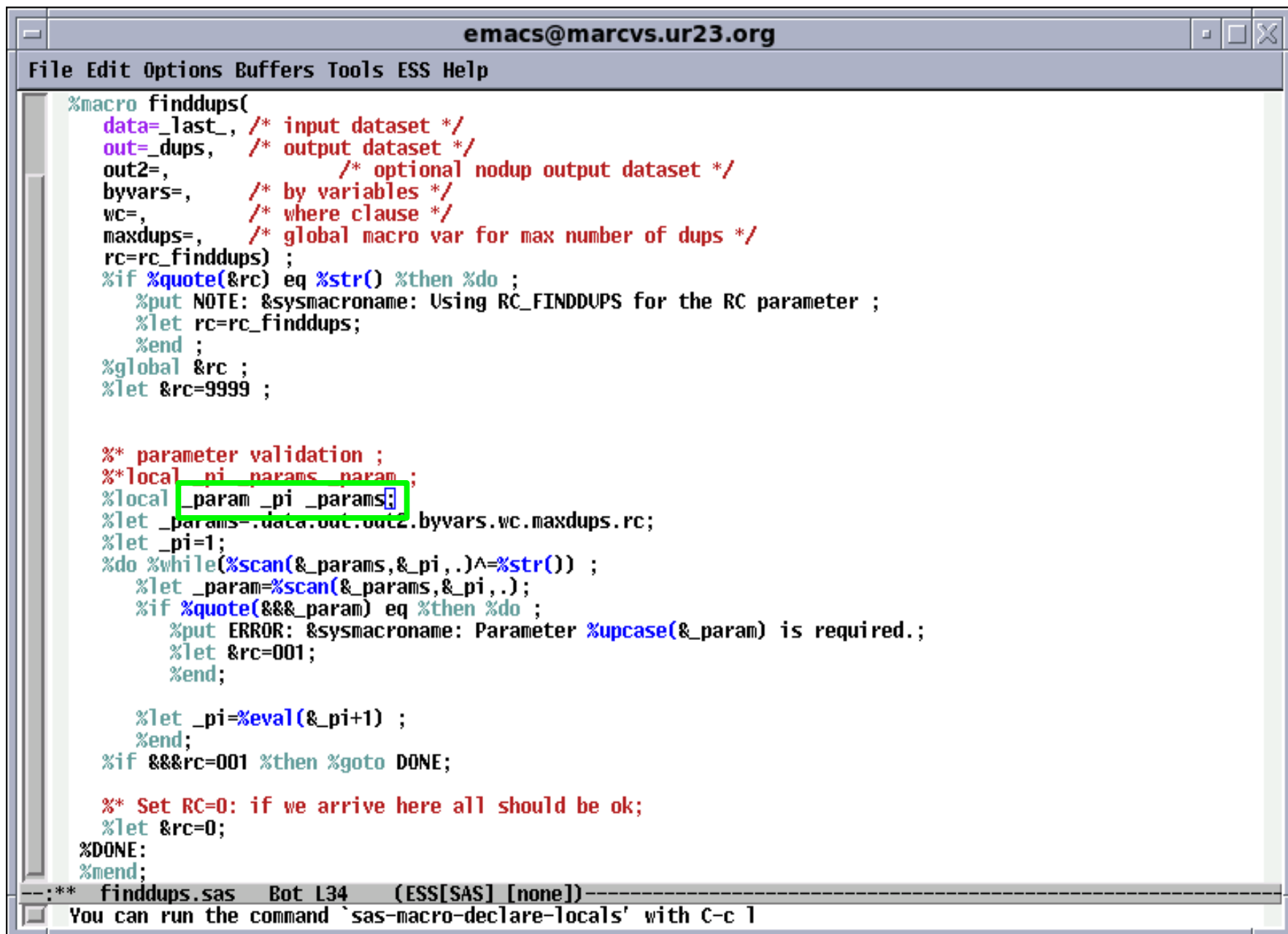
```
%macro finddups(
  data=_last_, /* input dataset */
  out=_dups,   /* output dataset */
  out2=,       /* optional nodup output dataset */
  byvars=,     /* by variables */
  wc=,         /* where clause */
  maxdups=,    /* global macro var for max number of dups */
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
  %global &rc ;
  %let &rc=9999 ;

  /* parameter validation ;
  **local _pi _params _param ;
  %local ;
  %let _params=.data.out.out2.byvars.wc.maxdups.rc;
  %let _pi=1;
  %do %while(%scan(&_params,&_pi,.)^=%str()) ;
    %let _param=%scan(&_params,&_pi,.);
    %if %quote(&&&_param) eq %then %do ;
      %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
      %let &rc=001;
    %end;

    %let _pi=%eval(&_pi+1) ;
  %end;
  %if &&&rc=001 %then %goto DONE;

  /* Set RC=0: if we arrive here all should be ok;
  %let &rc=0;
  %DONE:
  %mend;
** finddups sas Pot 134 (ESS[SAS] [none])
M-x sas-macro-declare-locals
```

Check for undeclared variables



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools ESS Help

%macro finddups(
  data=_last_ /* input dataset */
  out=_dups,   /* output dataset */
  out2=,       /* optional nodup output dataset */
  byvars=,     /* by variables */
  wc=,         /* where clause */
  maxdups=,    /* global macro var for max number of dups */
  rc=rc_finddups) ;
  %if %quote(&rc) eq %str() %then %do ;
    %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
    %let rc=rc_finddups;
  %end ;
%global &rc ;
%let &rc=9999 ;

/* parameter validation ;
%local _pi _params _param ;
%local _param _pi _params;
%let _params=.data.out.out2.byvars.wc.maxdups.rc;
%let _pi=1;
%do %while(%scan(&_params,&_pi,.)^=%str()) ;
  %let _param=%scan(&_params,&_pi,.);
  %if %quote(&&&_param) eq %then %do ;
    %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
    %let &rc=001;
  %end;

  %let _pi=%eval(&_pi+1) ;
%end;
%if &&&rc=001 %then %goto DONE;

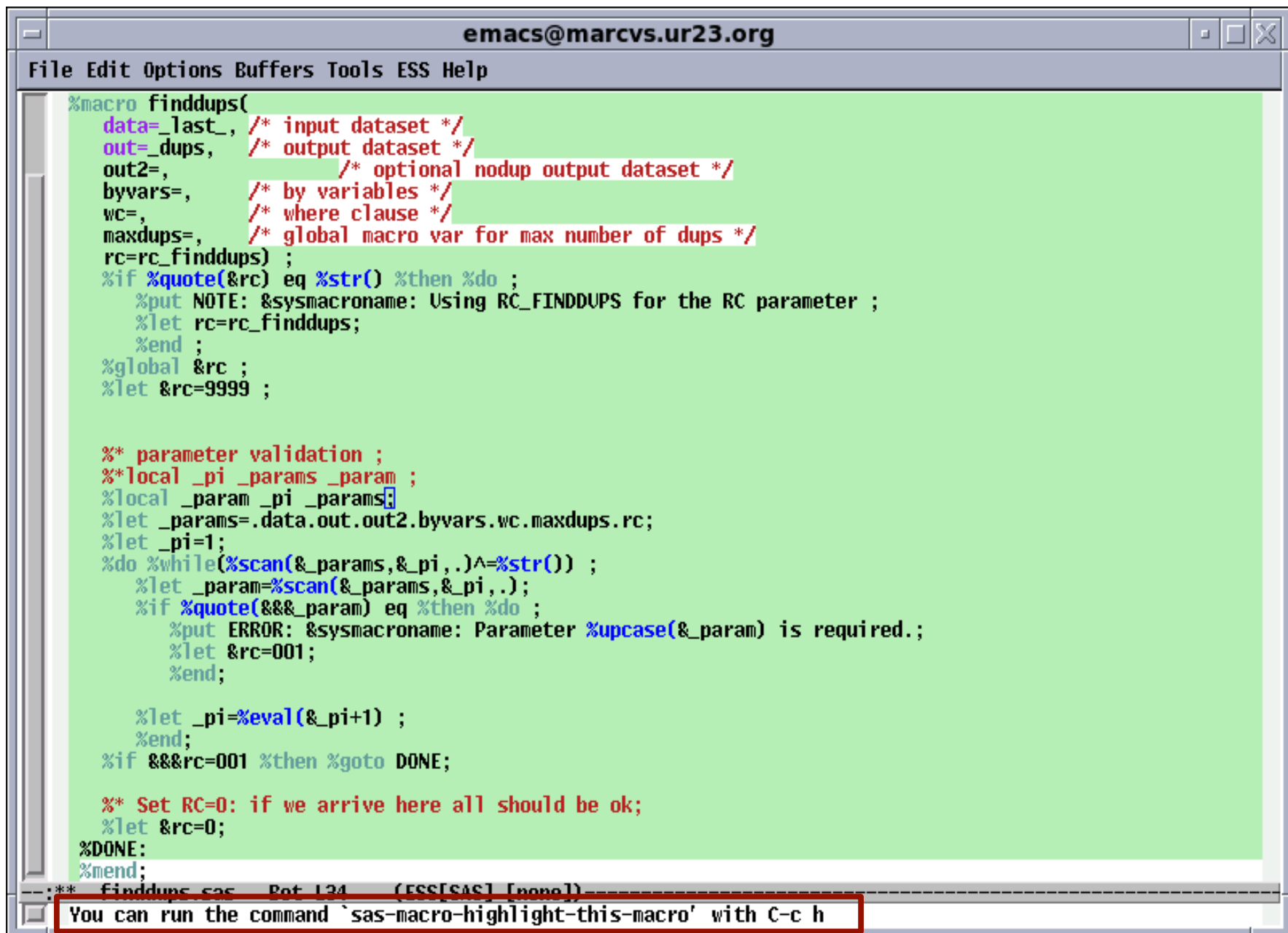
/* Set RC=0: if we arrive here all should be ok;
%let &rc=0;
%DONE:
%mend;

--:** finddups.sas Bot L34 (ESS[SAS] [none])--
You can run the command 'sas-macro-declare-locals' with C-c l
```

Extensions for robust macro techniques

- Template insertion: macro skeleton
- Template insertion: parameter validation
- Checking for undeclared macro variables
- Highlighting the current macro
- Template insertion: checking data and proc step return codes

Highlight the current macro



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools ESS Help

%macro finddups(
  data=_last_, /* input dataset */
  out=_dups,    /* output dataset */
  out2=,        /* optional nodup output dataset */
  byvars=,      /* by variables */
  wc=,          /* where clause */
  maxdups=,     /* global macro var for max number of dups */
  rc=rc_finddups) ;
%if %quote(&rc) eq %str() %then %do ;
  %put NOTE: &sysmacroname: Using RC_FINDDUPS for the RC parameter ;
  %let rc=rc_finddups;
%end ;
%global &rc ;
%let &rc=9999 ;

/* parameter validation ;
%local _pi _params _param ;
%local _param _pi _params;
%let _params=.data.out.out2.byvars.wc.maxdups.rc;
%let _pi=1;
%do %while(%scan(&_params,&_pi,.)^=%str()) ;
  %let _param=%scan(&_params,&_pi,.);
  %if %quote(&&&_param) eq %then %do ;
    %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
    %let &rc=001;
  %end;

  %let _pi=%eval(&_pi+1) ;
%end;
%if &&&rc=001 %then %goto DONE;

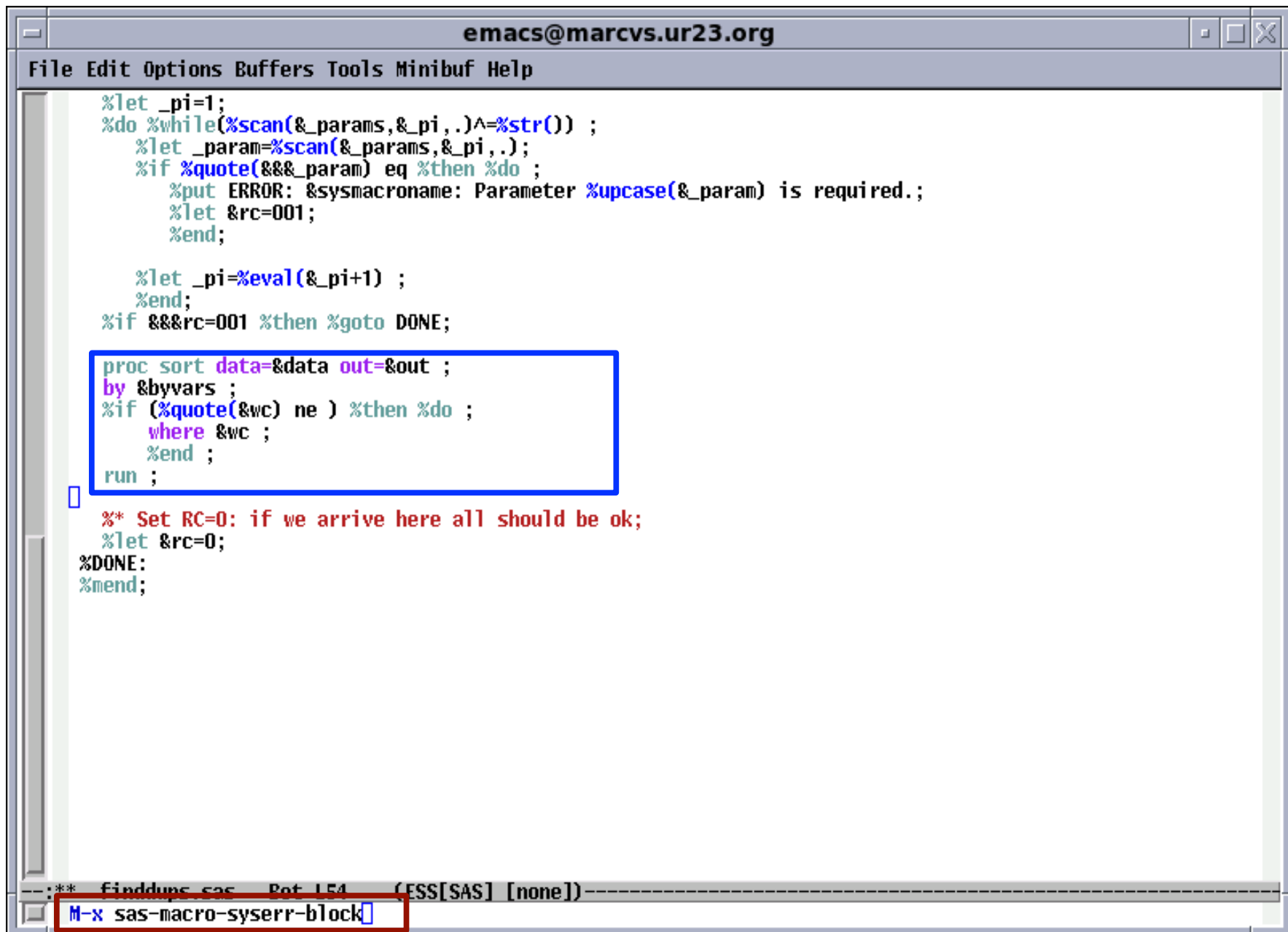
/* Set RC=0: if we arrive here all should be ok;
%let &rc=0;
%DONE:
%mend;

** finddups.sas  Pot 134  (ESS[SAS] [none])
You can run the command 'sas-macro-highlight-this-macro' with C-c h
```

Extensions for robust macro techniques

- Template insertion: macro skeleton
- Template insertion: parameter validation
- Checking for undeclared macro variables
- Highlighting the current macro
- Template insertion: checking data and proc step return codes

Check for error in proc or data step



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools Minibuf Help

%let _pi=1;
%do %while(%scan(&_params,&_pi,.)^=%str())) ;
  %let _param=%scan(&_params,&_pi,.);
  %if (%quote(&&&_param) eq %then %do ;
    %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
    %let &rc=001;
  %end;

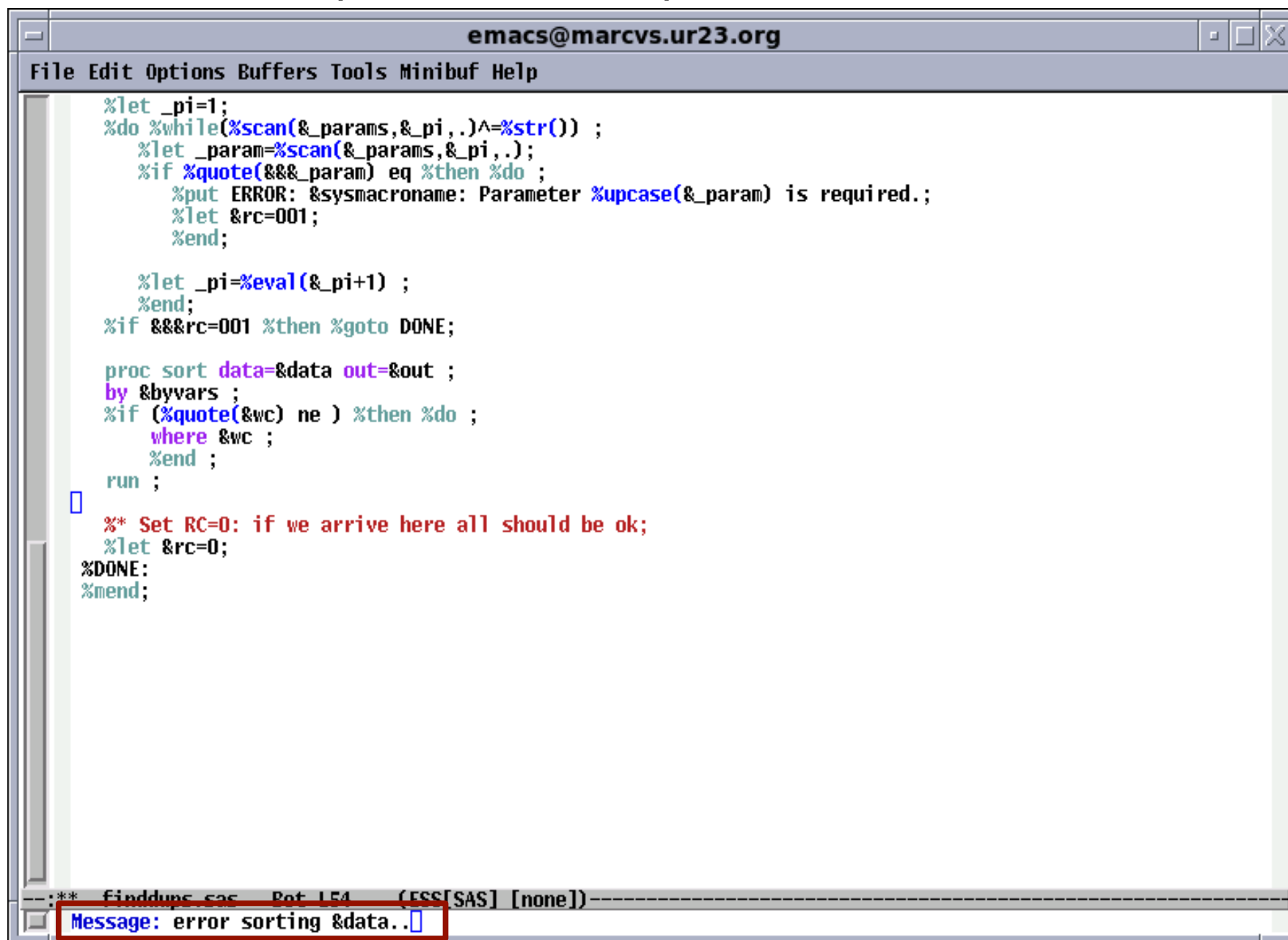
  %let _pi=%eval(&_pi+1) ;
%end;
%if &&&rc=001 %then %goto DONE;

proc sort data=&data out=&out ;
by &byvars ;
%if (%quote(&wc) ne ) %then %do ;
  where &wc ;
%end ;
run ;

%* Set RC=0: if we arrive here all should be ok;
%let &rc=0;
%DONE:
%mend;

** finddups.sas  Pot 154  (ESS[SAS] [none])
M-x sas-macro-syserr-block
```

Check for error in proc or data step



The screenshot shows an Emacs editor window titled 'emacs@marcv.s.ur23.org'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Minibuf', and 'Help'. The main text area contains SAS code with syntax highlighting. The code includes a loop to process parameters, a PROC SORT step, and a RUN statement. A red comment line states: '/* Set RC=0: if we arrive here all should be ok;'. Below the code, a status bar shows 'Finddups.sas', 'Buf 154', and '(ESS[SAS] [none])'. A red box highlights the message 'Message: error sorting &data..' in the status bar.

```
%let _pi=1;
%do %while(%scan(&_params,&_pi,.)^=%str()) ;
  %let _param=%scan(&_params,&_pi,.);
  %if %quote(&&&_param) eq %then %do ;
    %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
    %let &rc=001;
  %end;

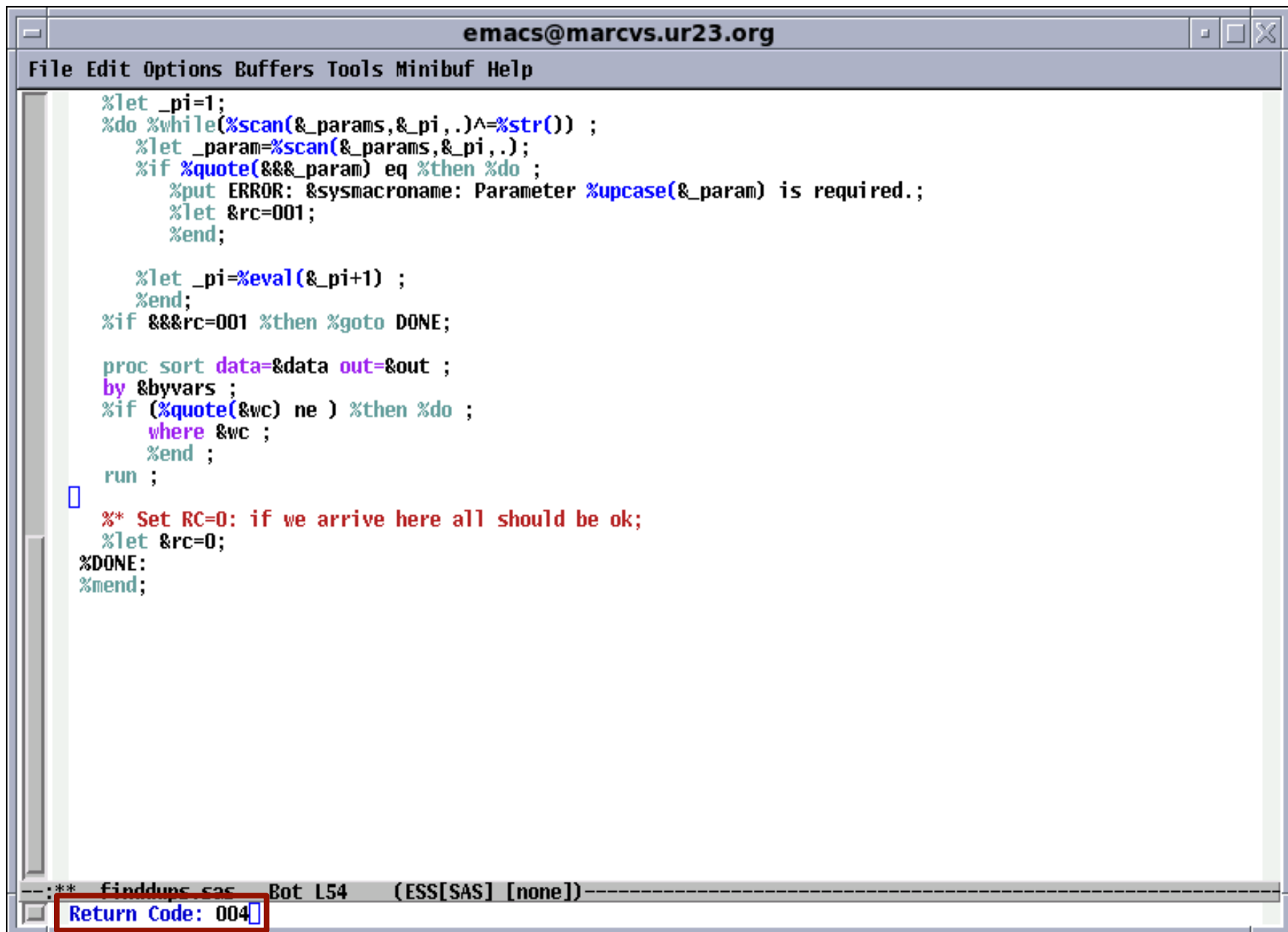
  %let _pi=%eval(&_pi+1) ;
%end;
%if &&&rc=001 %then %goto DONE;

proc sort data=&data out=&out ;
by &byvars ;
%if (%quote(&wc) ne ) %then %do ;
  where &wc ;
%end ;
run ;

/* Set RC=0: if we arrive here all should be ok;
%let &rc=0;
%DONE:
%mend;
```

Message: error sorting &data..

Check for error in proc or data step



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools Minibuf Help

%let _pi=1;
%do %while(%scan(&_params,&_pi,.)^=%str())) ;
  %let _param=%scan(&_params,&_pi,.);
  %if (%quote(&&&_param) eq %then %do ;
    %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
    %let &rc=001;
  %end;

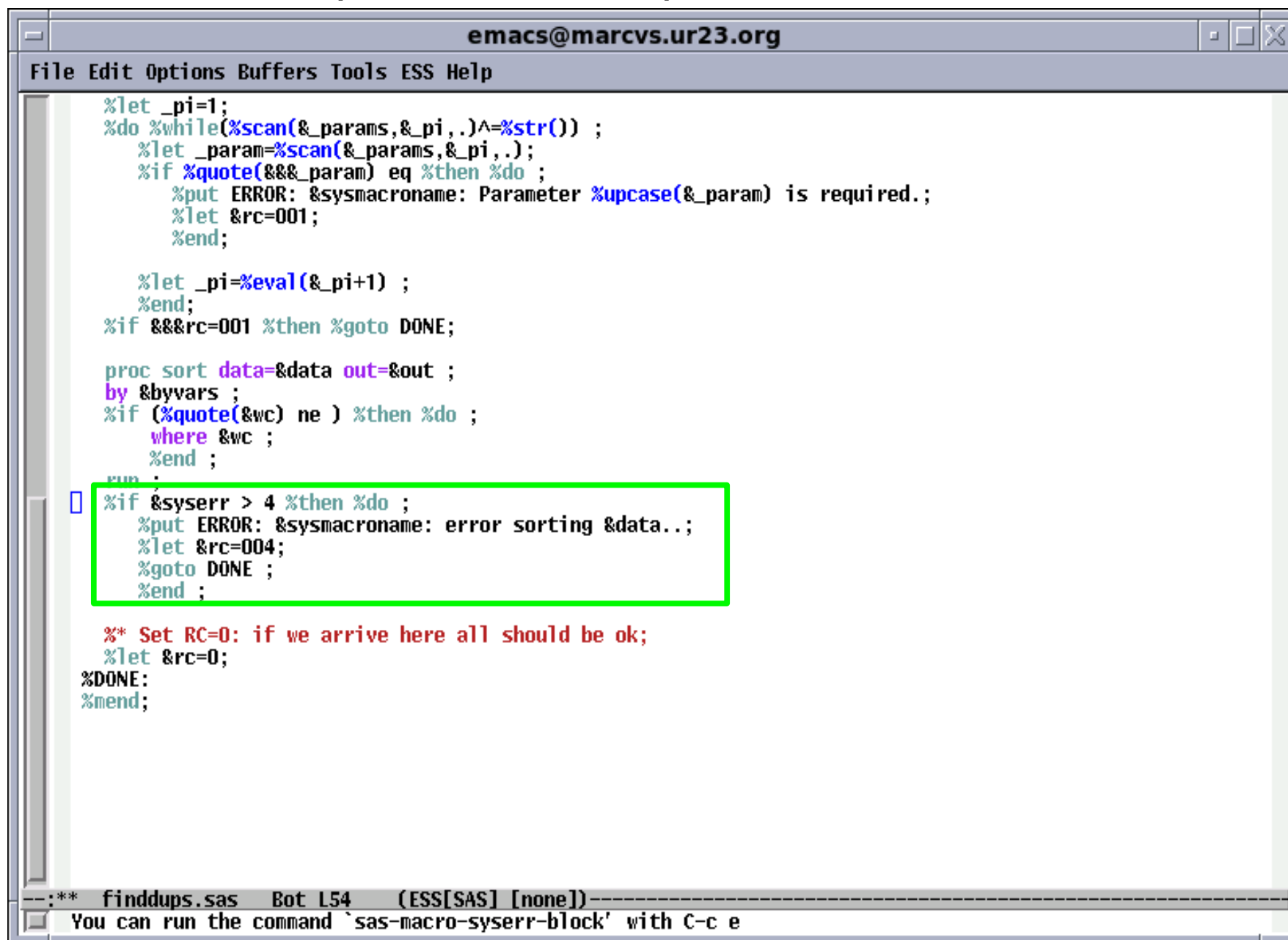
  %let _pi=%eval(&_pi+1) ;
%end;
%if &&&rc=001 %then %goto DONE;

proc sort data=&data out=&out ;
by &byvars ;
%if (%quote(&wc) ne ) %then %do ;
  where &wc ;
%end ;
run ;

%* Set RC=0: if we arrive here all should be ok;
%let &rc=0;
%DONE:
%mend;
```

Return Code: 004

Check for error in proc or data step



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools ESS Help

%let _pi=1;
%do %while(%scan(&_params,&_pi,.)^=%str())) ;
  %let _param=%scan(&_params,&_pi,.);
  %if (%quote(&&&_param) eq %then %do ;
    %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
    %let &rc=001;
    %end;

    %let _pi=%eval(&_pi+1) ;
    %end;
  %if &&&rc=001 %then %goto DONE;

  proc sort data=&data out=&out ;
  by &byvars ;
  %if (%quote(&wc) ne ) %then %do ;
    where &wc ;
  %end ;
run ;
%if &syserr > 4 %then %do ;
  %put ERROR: &sysmacroname: error sorting &data..;
  %let &rc=004;
  %goto DONE ;
%end ;

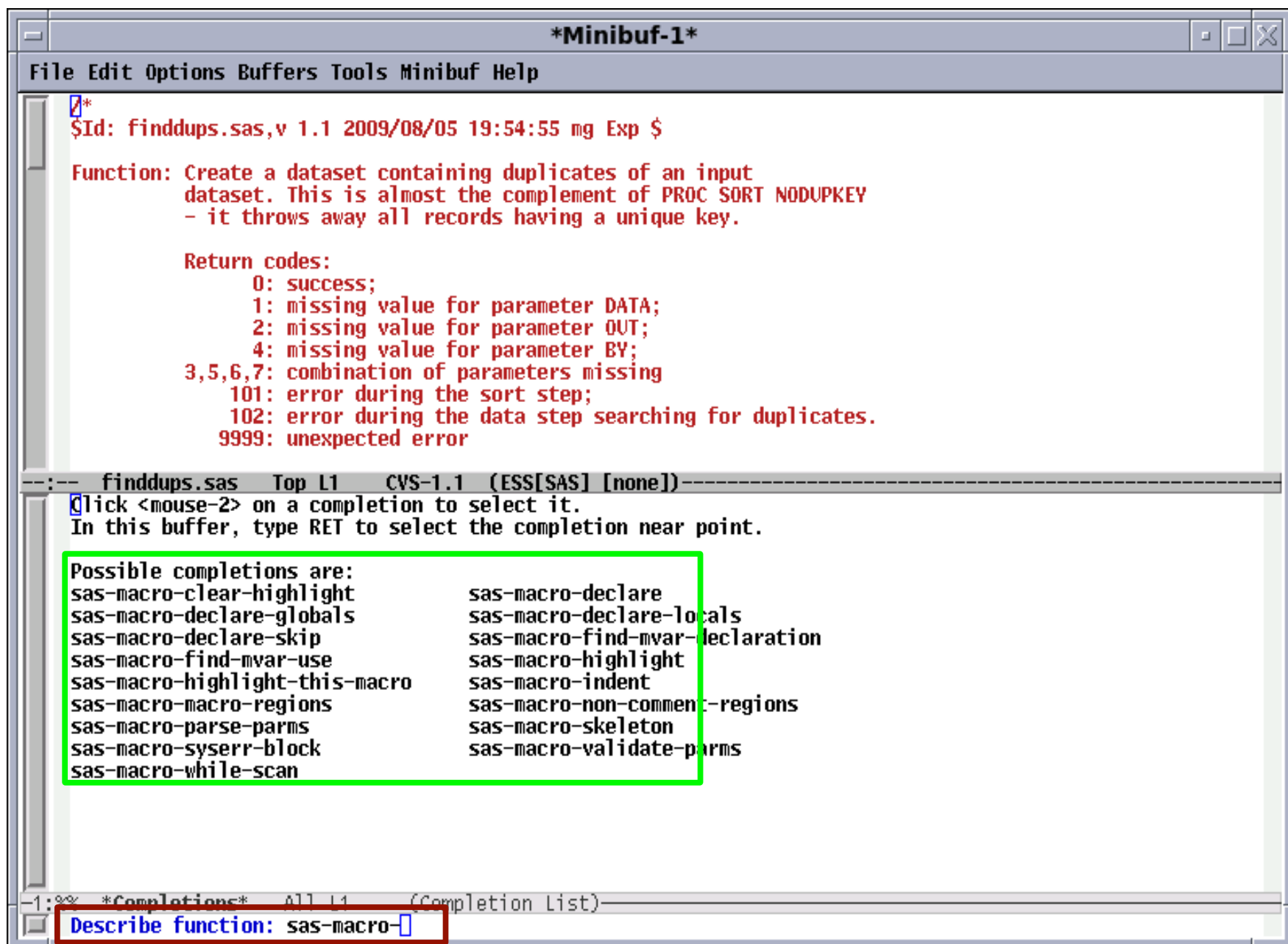
%* Set RC=0: if we arrive here all should be ok;
%let &rc=0;
%DONE:
%mend;
```

--:** finddups.sas Bot L54 (ESS[SAS] [none])--
You can run the command 'sas-macro-syserr-block' with C-c e

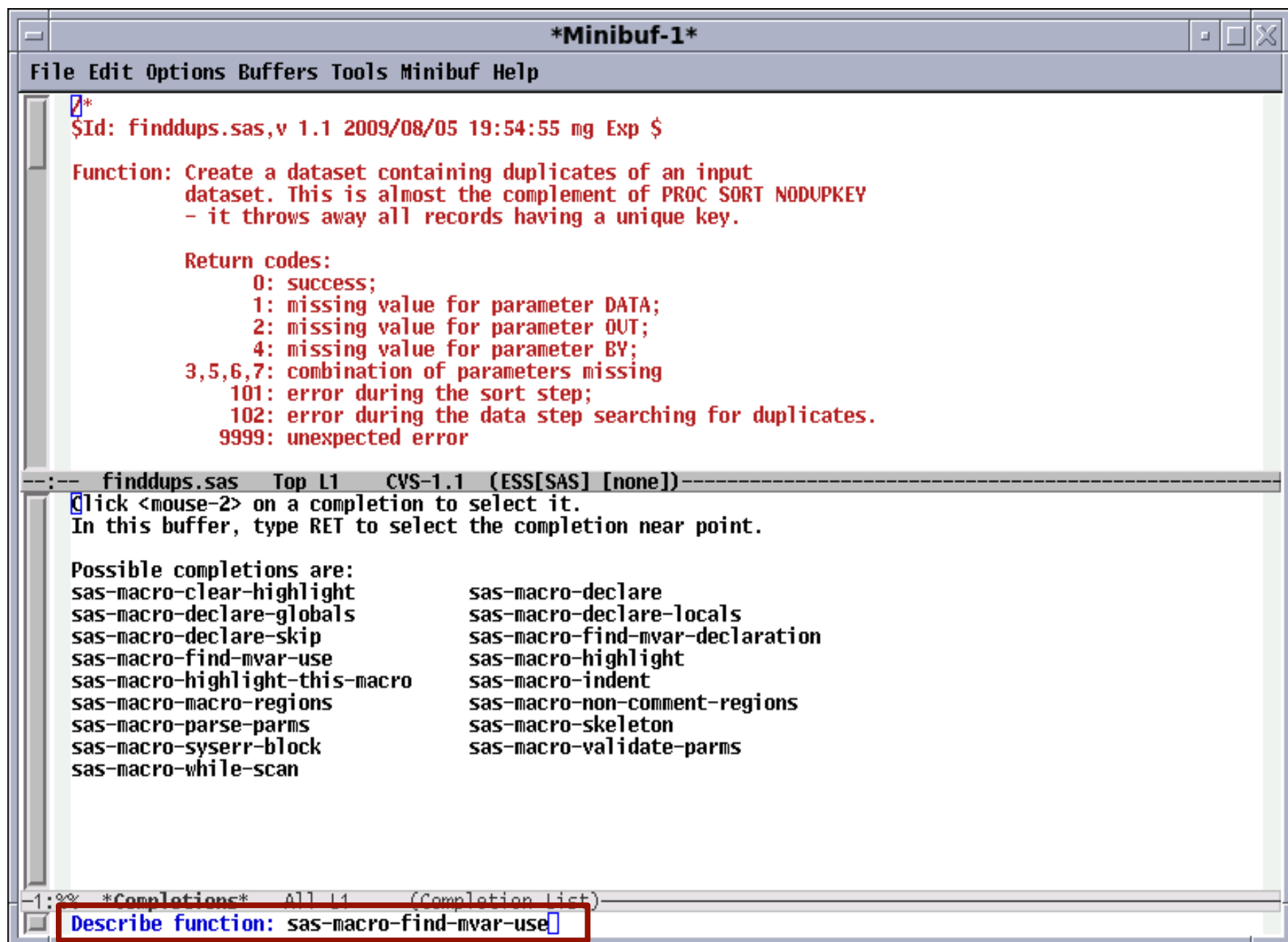
GNU Emacs – the extensible programmer's editor

- Examples of what emacs does really well:
 - process multiple files
 - modes for almost all programming languages
 - ease of extension
 - built-in documentation
 - version control interface

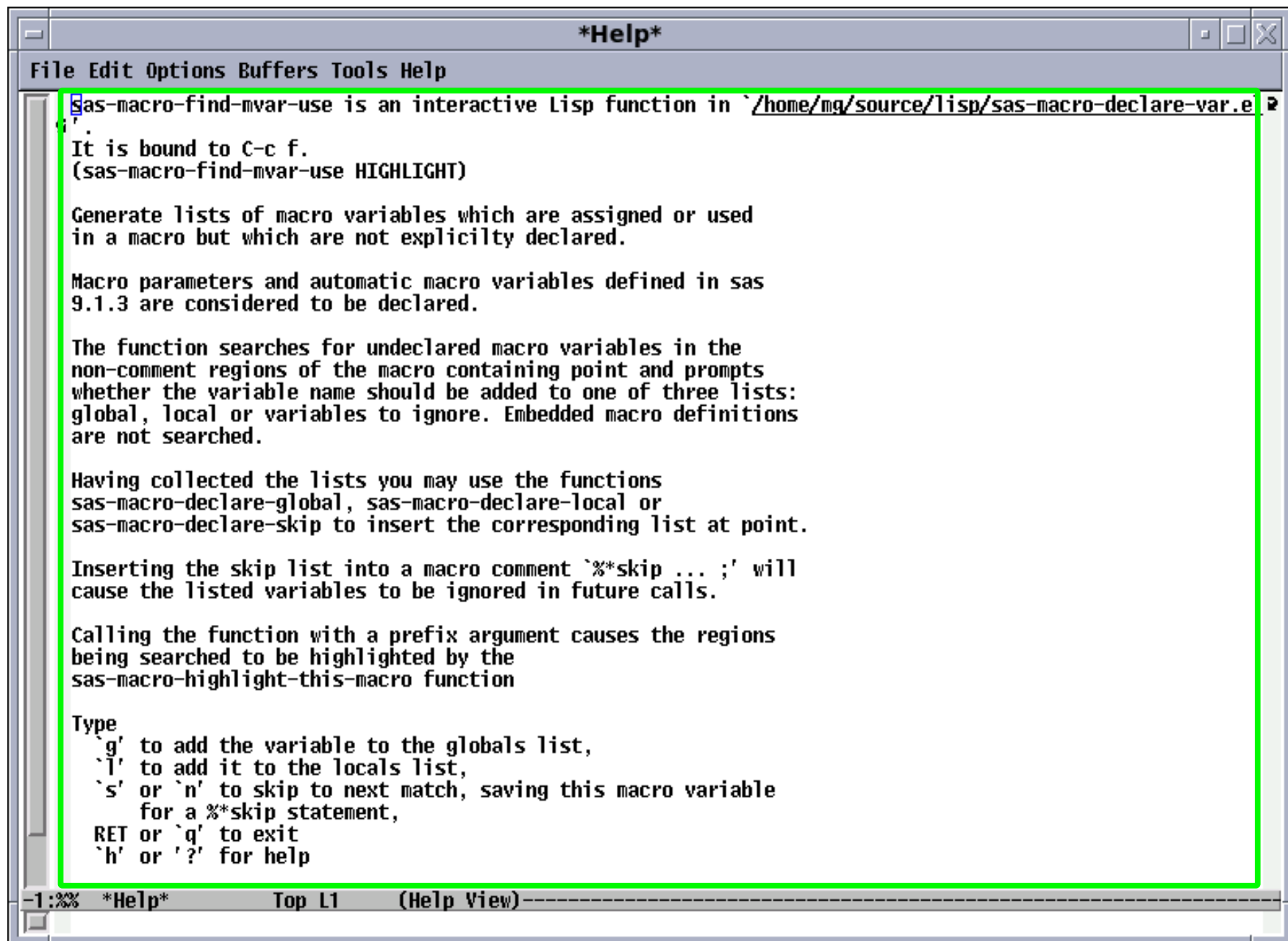
Built-in documentation



Built-in documentation



Built-in documentation



The screenshot shows an Emacs window titled `*Help*` with a menu bar containing `File Edit Options Buffers Tools Help`. The main text area displays the documentation for the `sas-macro-find-mvar-use` function. The text is as follows:

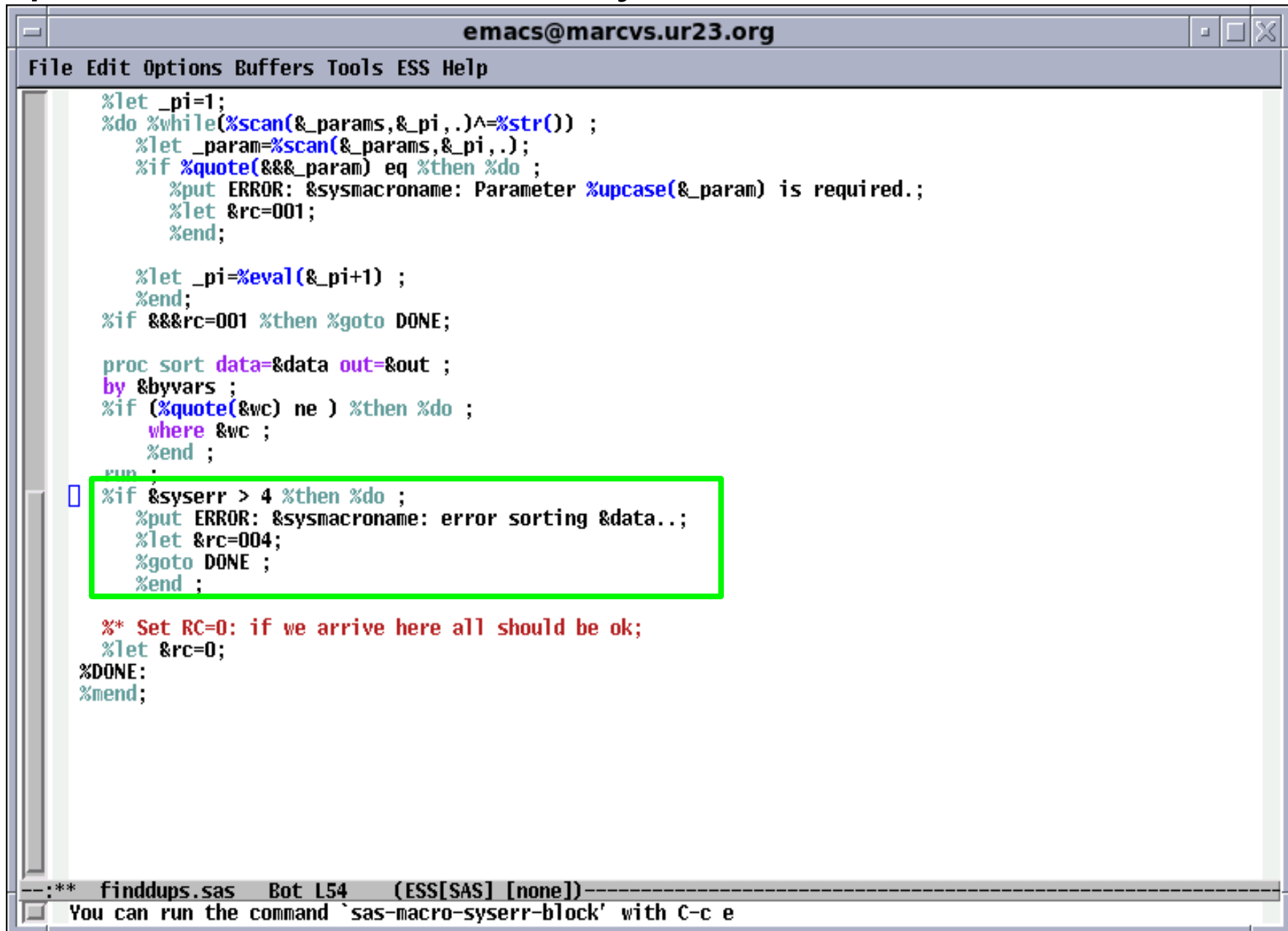
```
sas-macro-find-mvar-use is an interactive Lisp function in ~/home/mg/source/lisp/sas-macro-declare-var.el.  
It is bound to C-c f.  
(sas-macro-find-mvar-use HIGHLIGHT)  
  
Generate lists of macro variables which are assigned or used  
in a macro but which are not explicitly declared.  
  
Macro parameters and automatic macro variables defined in sas  
9.1.3 are considered to be declared.  
  
The function searches for undeclared macro variables in the  
non-comment regions of the macro containing point and prompts  
whether the variable name should be added to one of three lists:  
global, local or variables to ignore. Embedded macro definitions  
are not searched.  
  
Having collected the lists you may use the functions  
sas-macro-declare-global, sas-macro-declare-local or  
sas-macro-declare-skip to insert the corresponding list at point.  
  
Inserting the skip list into a macro comment %*skip ... ; will  
cause the listed variables to be ignored in future calls.  
  
Calling the function with a prefix argument causes the regions  
being searched to be highlighted by the  
sas-macro-highlight-this-macro function  
  
Type  
  'g' to add the variable to the globals list,  
  'l' to add it to the locals list,  
  's' or 'n' to skip to next match, saving this macro variable  
      for a %*skip statement,  
RET or 'q' to exit  
  'h' or '?' for help
```

The status bar at the bottom of the window shows `-1:%% *Help* Top L1 (Help View)`.

GNU Emacs – the extensible programmer's editor

- Examples of what emacs does really well:
 - process multiple files
 - modes for almost all programming languages
 - ease of extension
 - built-in documentation
 - version control interface

Lisp extensions can be easy



```
emacs@marcv.sur23.org
File Edit Options Buffers Tools ESS Help

%let _pi=1;
%do %while(%scan(&_params,&_pi,.)^=%str()) ;
  %let _param=%scan(&_params,&_pi,.);
  %if %quote(&&&_param) eq %then %do ;
    %put ERROR: &sysmacroname: Parameter %upcase(&_param) is required.;
    %let &rc=001;
    %end;

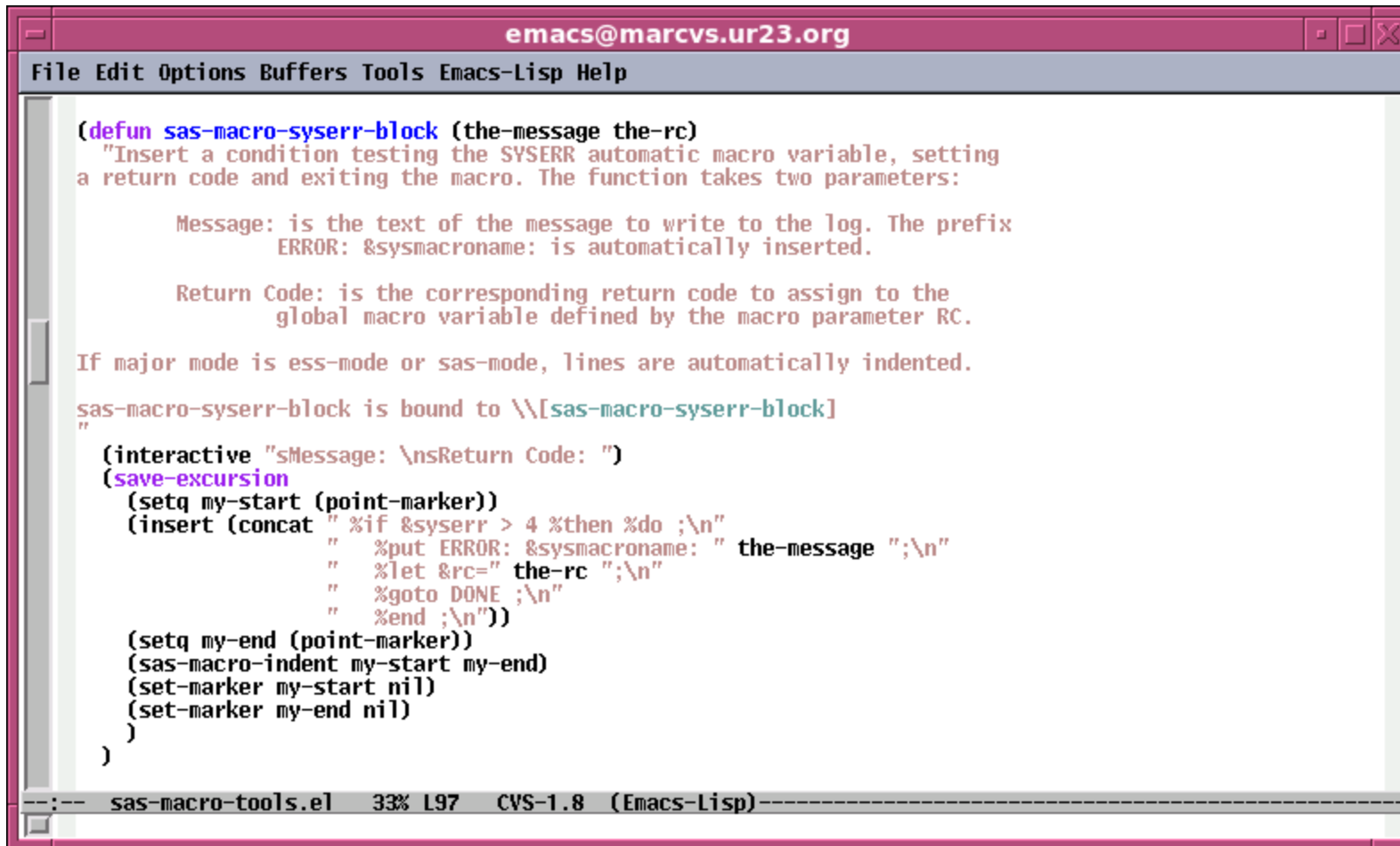
    %let _pi=%eval(&_pi+1) ;
  %end;
%if &&&rc=001 %then %goto DONE;

proc sort data=&data out=&out ;
by &byvars ;
%if (%quote(&wc) ne ) %then %do ;
  where &wc ;
%end ;
run ;
%if &syserr > 4 %then %do ;
  %put ERROR: &sysmacroname: error sorting &data..;
  %let &rc=004;
  %goto DONE ;
%end ;

/* Set RC=0: if we arrive here all should be ok;
%let &rc=0;
%DONE:
%mend;
```

--:** finddups.sas Bot L54 (ESS[SAS] [none])--
You can run the command 'sas-macro-syserr-block' with C-c e

Lisp extensions can be easy



The screenshot shows an Emacs window titled 'emacs@marcv.sur23.org'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Emacs-Lisp', and 'Help'. The main text area contains the following Lisp code and documentation:

```
(defun sas-macro-syserr-block (the-message the-rc)
  "Insert a condition testing the SYSERR automatic macro variable, setting
  a return code and exiting the macro. The function takes two parameters:

      Message: is the text of the message to write to the log. The prefix
      ERROR: &sysmacroname: is automatically inserted.

      Return Code: is the corresponding return code to assign to the
      global macro variable defined by the macro parameter RC.

  If major mode is ess-mode or sas-mode, lines are automatically indented.

  sas-macro-syserr-block is bound to \\[sas-macro-syserr-block]
  "
  (interactive "sMessage: \nsReturn Code: ")
  (save-excursion
    (setq my-start (point-marker))
    (insert (concat " %if &syserr > 4 %then %do ;\n"
                    " %put ERROR: &sysmacroname: " the-message ";\n"
                    " %let &rc=" the-rc ";\n"
                    " %goto DONE ;\n"
                    " %end ;\n"))
    (setq my-end (point-marker))
    (sas-macro-indent my-start my-end)
    (set-marker my-start nil)
    (set-marker my-end nil)
  )
)
```

The status bar at the bottom shows: '---:-- sas-macro-tools.el 33% L97 CVS-1.8 (Emacs-Lisp)-----'.

Conclusions

- Tools support standards
- Emacs supports programming standards generally
- ESS has specific support for SAS
- Extensions make it easy to implement robust macro techniques
- Building extensions is relatively easy