PRELIMINARY TEST PROCEDURE FOR IRAF

Part 1. Image and dataio tests

Jeannette V. Barnes
Central Computer Services
National Optical Astronomy Observatories
Tucson, Arizona

Revised May 26, 1989

The following pages describe a short test procedure that new users can execute to test some basic image functions within IRAF for a new installation. This process will help verify that everything is working correctly and also help the first time user gain familiarity with the system.

We will assume that you have started IRAF and are residing in an empty directory from which you wish to work. You will need to load the following packages: dataio, noao (loads images and plot), and lists. We do this by simply typing the package names:

```
dataio
noao
lists
```

1. Type **package** to be sure you have loaded all the necessary packages. The following should be displayed.

```
dataio
user
system
lists
clpackage
language
noao
plot
images
```

For this particular procedure we want to use all the default parameters for the various tasks. Thus we will ‘unlearn’ each of the packages that we have loaded. Note that this is not the usual procedure - generally, you want IRAF to ‘learn’ or remember the hidden parameters that were used previously. To do this, type

```
unlearn  dataio
unlearn  images
unlearn  plot
unlearn  lists
```

or

```
unlearn  dataio  images  plot  lists
```

2. An IRAF image exists in the ‘dev’ directory. Let’s first make a copy of this image, putting the header file into the current default directory and the pixel file into your default image directory.

```
imcopy  dev$pix  image.short
```

3. Let’s look at the header information for this image.

imhead image.short long+

should produce a listing similar to the following:

```
image.short[512,512][short]: m51  B  600s
    No bad pixels, no histogram, min=unknown, max=unknown
    Line storage mode, physdim [512,512], length of user area 1621 s.u.
    Created Thu 06:13:36 05-Jul-90, Last modified Thu 06:13:37 05-Jul-90
    Pixel file 'draco!SCR$1:[JBARNES]imagej7short.pix' [ok]
    'KPNO-IRAF'  /
    '20-10-88'  /
    'KPNO-IRAF'  /
    '24-04-87'  /
    New copy of one035.imh
    New copy of one035
    New copy of m51
    New copy of m513
    New copy of m51
    New copy of dev$pix
    IRAF-MAX=           1.993600E4  /  DATA MAX
    IRAF-MIN=          -1.000000E0  /  DATA MIN
    IRAF-B/P=                   16  /  DATA BITS/PIXEL
    IRAFTYPE= 'INTEGER '  /
    IRAF-MAX=           1.229817E4  /  DATA MAX
    IRAF-MIN=          -6.053954E0  /  DATA MIN
    IRAF-B/P=                   32  /  DATA BITS/PIXEL
    IRAFTYPE= 'FLOATING'  /
    CCDPICNO=                   53  /  ORIGINAL CCD PICTURE NUMBER
    ITIME   =                  600  /  REQUESTED INTEGRATION TIME (SECS)
    TTIME   =                  600  /  TOTAL ELAPSED TIME (SECS)
    OTIME   =                  600  /  ACTUAL INTEGRATION TIME (SECS)
    DATA-TYP= 'OBJECT (0)'  /  OBJECT,DARK,BIAS,ETC.
    DATE-OBS= '05/04/87'  /  DATE DD/MM/YY
    RA      = '13:29:24'  /  right ascension
    DEC     = '47:15:34'  /  declination
    EPOCH   =                 0.00  /  EPOCH OF RA AND DEC
    ZD      = '22:14:00'  /  zenith distance
    UT      = ' 9:27:27'  /  universal time
    ST      = '14:53:42'  /  sidereal time
    CAM-ID  =                    1  /  CAMERA HEAD ID
    CAM-TEMP=              -106.22  /  CAMERA TEMPERATURE, DEG C
    DEW-TEMP=              -180.95  /  DEWAR TEMPRATURE, DEG C
    F1POS   =                    2  /  FILTER BOLT I POSITION
    F2POS   =                    0  /  FILTER BOLT II POSITION
    TVFILT  =                    0  /  TV FILTER
    CMP-LAMP=                    0  /  COMPARISON LAMP
    TILT-POS=                    0  /  TILT POSITION
    BIAS-PIX=                    0  /
    BI-FLAG =                    0  /  BIAS SUBTRACT FLAG
    BP-FLAG =                    0  /  BAD PIXEL FLAG
    CR-FLAG =                    0  /  BAD PIXEL FLAG
    DK-FLAG =                    0  /  DARK SUBTRACT FLAG
    FR-FLAG =                    0  /  FRINGE FLAG
    FR-SCALE=                 0.00  /  FRINGE SCALING PARAMETER
    TRIM    = 'Apr 22 14:11 Trim image section is [3:510,3:510]'
    BT-FLAG = 'Apr 22 14:11 Overscan correction strip is [515:544,3:510]'
    FF-FLAG = 'Apr 22 14:11 Flat field image is Flat1.imh with scale=183.9447'
    CCDPROC = 'Apr 22 14:11 CCD processing done'
    AIRMASS =    1.08015632629395  / AIRMASS
```

Note that the pixels are short integers (=16-bits). Notice also that your pixel file path (line 5) is different from the one above - that, of course, is to be expected. (In the above case, IRAF has converted the "." in the pixel file path name to "j7", to produce a legal VMS filename.)

It would be useful to generate two more copies of this image but with different pixel types - one with 32-bit floating point pixels (called reals) and one with 64-bit double precision floating point pixels (called double). Note that IRAF also supports other pixel data types - 32-bit integers called long and complex numbers. Execute the following:

```
imarith  image.short  \  1  image.real  pixtype=r
imarith  image.short  \  1  image.dbl  pixtype=d
imhead  image.*
```

Your terminal should now display the following, generated from the ‘imheader’ task above:

```
image.short.imh[512,512][short]: m51  B  600s
image.real.imh[512,512][real]: m51  B  600s
image.dbl.imh[512,512][double]: m51  B  600s
```

Note the ‘.imh’ extension to the image name - this declares that the image was written as an IRAF image rather than an SDAS image (denoted by ‘.hhh’). In many cases, this extension is transparent to the user.

4. Let’s execute a couple of more tasks that will exercise some image operators. Typing

```
minmax  image.dbl,image.real,image.short
```

Should display the following:

```
image.dbl [77,4] -1. [348,189] 19936.
image.real [77,4] -1. [348,189] 19936.
image.short [77,4] -1. [348,189] 19936.
```

Now execute

```
listpix  image.short[300:310,200:205]  |  table
```

The following table of pixel values should be displayed.

```
 1   1  145.       7   2  130.       2   4  149.       8   5  140.
 2   1  143.       8   2  132.       3   4  146.       9   5  143.
 3   1  141.       9   2  128.       4   4  143.      10   5  131.
 4   1  142.      10   2  132.       5   4  145.      11   5  148.
 5   1  135.      11   2  139.       6   4  140.       1   6  138.
 6   1  138.       1   3  162.       7   4  133.       2   6  139.
 7   1  134.       2   3  145.       8   4  129.       3   6  145.
 8   1  125.       3   3  146.       9   4  128.       4   6  141.
 9   1  130.       4   3  144.      10   4  141.       5   6  141.
10   1  123.       5   3  135.      11   4  137.       6   6  149.
11   1  132.       6   3  141.       1   5  144.       7   6  149.
 1   2  147.       7   3  127.       2   5  145.       8   6  147.
 2   2  147.       8   3  129.       3   5  133.       9   6  144
 3   2  145.       9   3  131.       4   5  144.      10   6  143
 4   2  141.      10   3  133.       5   5  145.      11   6  151.
 5   2  132.      11   3  135.       6   5  144.
 6   2  130.       1   4  149.       7   5  143.
```

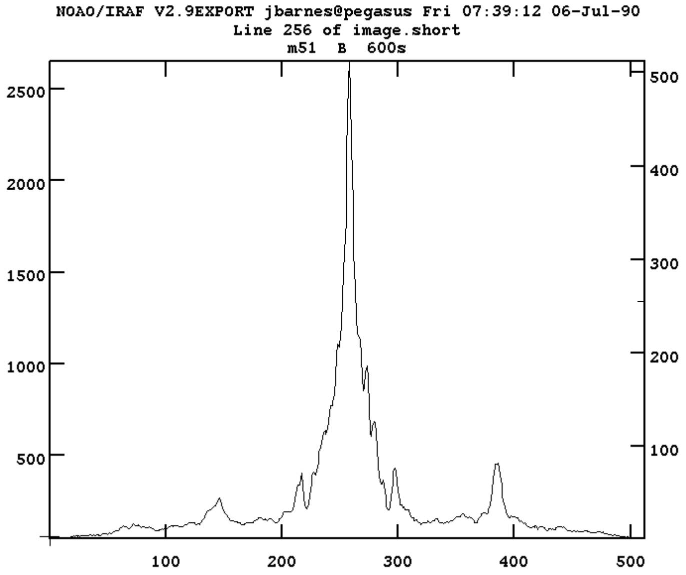
Similarly, the following executions of ‘listpixels’ should generate the same table as above.

```
listpix  image.real[300:310,200:205]   |  table
listpix  image.dbl[300:310,200:205]   |  table
```

5. Now let’s check some plotting options. Type

```
implot   image.short
```

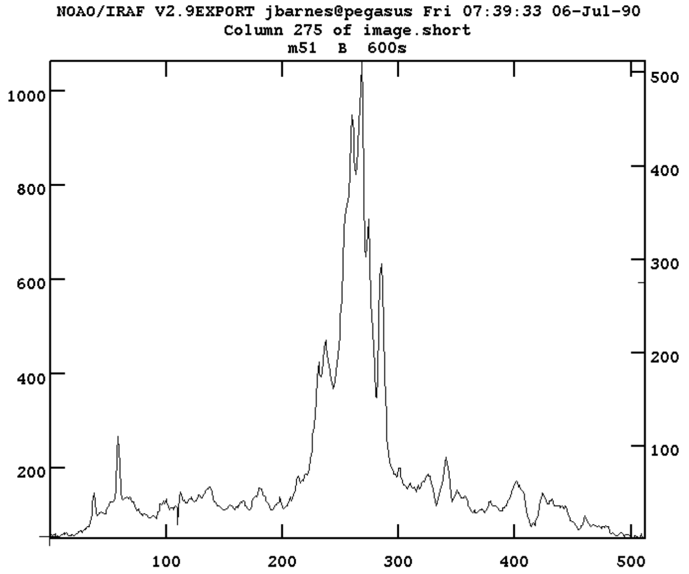
The following plot should be displayed - note that ‘implot’ defaults to the center line (row) of the image.



While the cursor is being displayed type

```
:c  275
```

and the following plot should be displayed.



While still in ‘implot’ and with the cursors displayed, executing the following set of commands should display the same plots for the other images.

```
:i  image.real
:l  256
:c  275
:i  image.dbl
:l  256
:c  275
```

Note that if you lose the plot for some reason while you are in ‘implot’ that you can get the plot back with **0**, the ‘zero’. You can also generate a list of the cursor options available in **implot** by typing **?** while you are in cursor mode - the ‘return’ key will get the plot back. Exit ‘implot’ with **q**. Type **clear** to clear the screen.

6. Now let’s test the use of image sections. Type

```
imcopy  image.real[200:300,200:300]  image.sect
imhead  image.sect
```

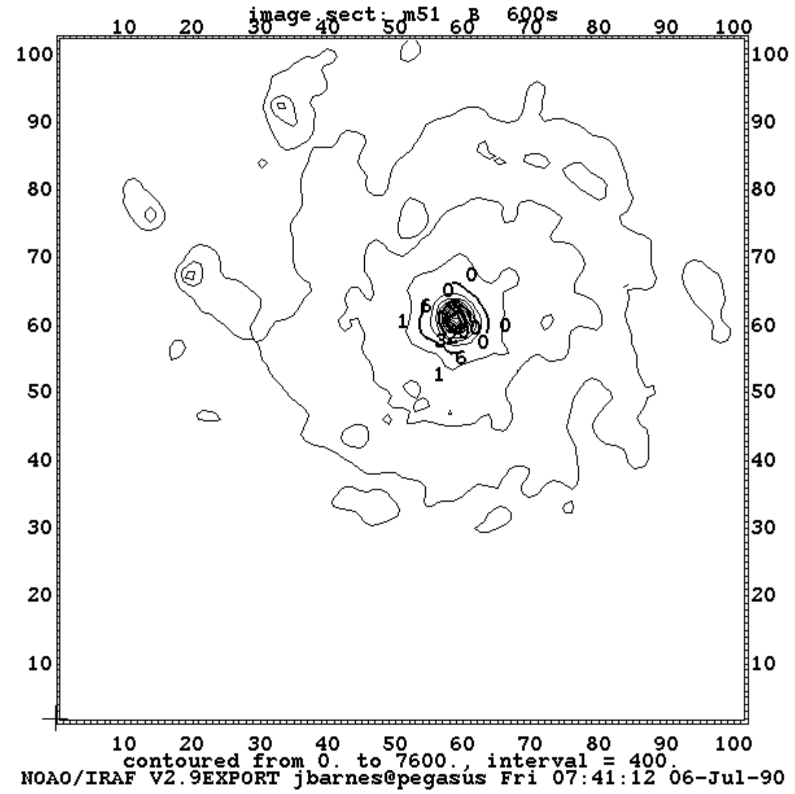
The result of the ‘imheader’ task should show

```
image.sect[101,101][real]:  m51  B  600s
```

7. Let’s make a contour plot of this new image.

```
contour  image.sect
```

The following plot should be displayed. Note that the contour levels may be marked differently.

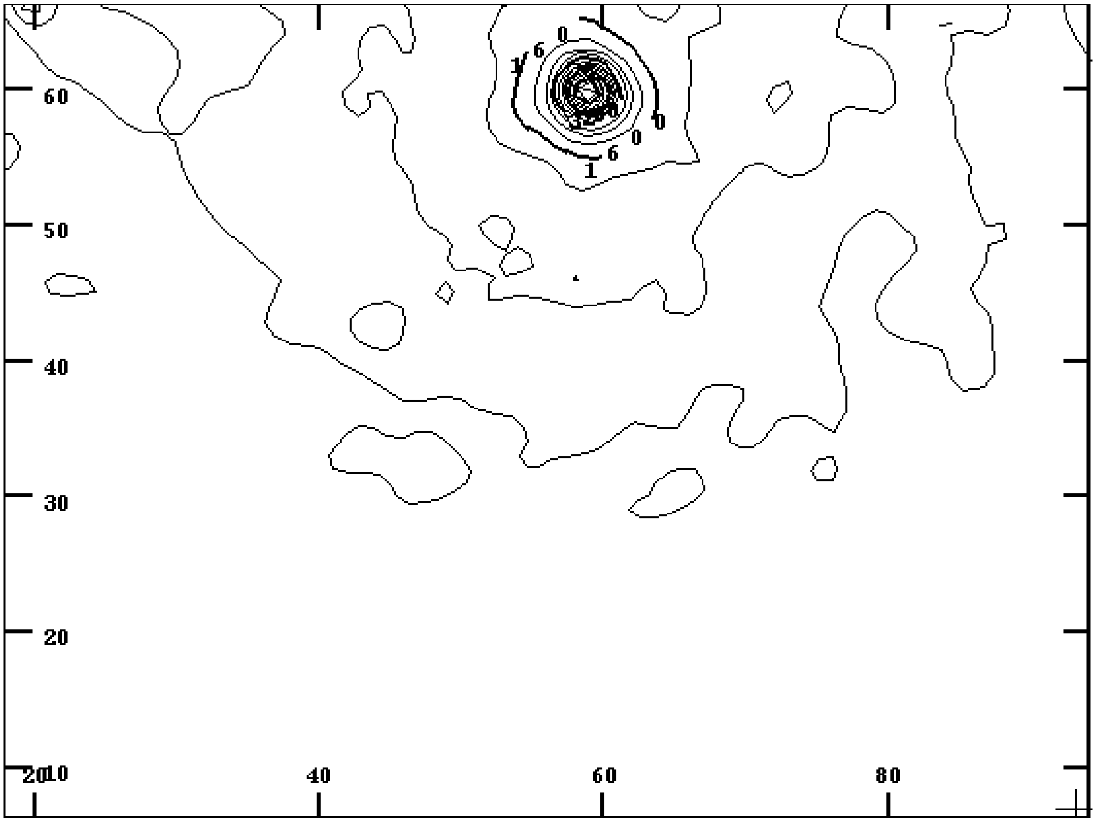


```
image.sect: m51  B  600s
contoured from 0. to 7600., interval = 400.
NOAO/IRAF V2.9EXPORT jbarnes@pegasus Fri 07:41:12 06-Jul-90
```

Let's again check some of the basic cursor plot options available in all of the plotting packages. While the contour plot is displayed on your screen, type

```
=gcur
```

The cursors should appear on your screen. Now move the cursors so that they intersect on the contour marked at x=57 and y=37. You can check to see if you have the right feature by typing '**C**' - that's a capital !! Once you are set on the right feature type '**Z**', again a capital, and the following 'zoomed' plot should be displayed. Use '**A**' to put the proper axes on the plot. Note that you can generate a listing of the cursor options by typing **:.help** while you are in cursor mode - these cursor options are basic to most plotting tasks in IRAF. Exit the 'cursor help mode' by **q** followed by a 'return' to replot. A **q** or any small letter keystroke will exit **=gcur**. Type **clear** to clear the screen.



8. At this time, let's modify a couple of the image titles. Run the task 'hedit', as below. The user input is underlined.

```
li> hedit
images to be edited: image.real
fields to be edited: title
value expression: m51 real
image.real,i_title ("m51  B  600s" -> "m51 real"):
image.real,i_title: "m51  B  600s" -> "m51 real"
update image.real ? (yes):
image.real updated
```

```
li> hedit
images to be edited (image.real): image.dbl
fields to be edited (title):
value expression (m51 real): m51 double
image.dbl,i_title ("m51  B  600s" -> "m51 double"):
image.dbl,i_title: "m51  B  600s" -> "m51 double"
update image.dbl ? (yes):
image.dbl updated
```

Typing **imhead image*** should now display the following.

```
image.short.imh[512,512][short]: m51  B  600s
image.real.imh[512,512][real]: m51 real
image.dbl.imh[512,512][double]: m51 double
image.sect.imh[101,101][real]: m51  B  600s
```

9. We will now concentrate on writing and reading images onto tape in FITS format. You may want to physically mount your tape on the drive first if you have a VMS system since the IRAF ‘allocate’ command does a ‘MOUNT/FOR’ in VMS. Note also that IRAF calls the tape devices ‘mta’, etc. So, please replace in the following the proper drive notation.

```
allocate  mtc
wfits  image.short  mtc  new+
```

10. Let’s make a quick check of the tape writing. Type

```
mtexamine  mtc
```

The following should be displayed on your terminal.

```
File mtc[1]:
        185 2880-byte records
        Total 185 records, 532800 bytes
File mtc[2]:
        Total 0 records, 0 bytes
Tape at EOT
```

Now type

```
devstatus  mtc
```

As a minimum, the following should be displayed. The first line, will of course, be different. What is important is the tape position.

```
# Magtape unit ‘mtc’ allocated to ‘jbarnes’ Sat 09:55:13 07-Jul-90
file = 2
record = 1 (EOT)
```

11. Let’s read the image back onto disk from the tape and check it.

```
rfits  mtc  1  new.short  long+
```

As the image is read in the FITS header should be displayed on your terminal.

```
File: new.short
SIMPLE  =                    T  /  FITS STANDARD
BITPIX  =                   16  /  FITS BITS/PIXEL
NAXIS   =                    2  /  NUMBER OF AXES
NAXIS1  =                  512  /
NAXIS2  =                  512  /
BSCALE  =       1.0000000000E0  /  REAL = TAPE*BSCALE + BZERO
BZERO   =       0.0000000000E0  /
OBJECT  = ’m51  B  600s’  /
ORIGIN  = ’KPNO-IRAF’  /
DATE    = ’07-07-90’  /
IRAFNAME= ’image.short’  /  NAME OF IRAF IMAGE FILE
IRAF-MAX=           1.993600E4  /  DATA MAX
IRAF-MIN=          -1.000000E0  /  DATA MIN
IRAF-B/P=                   16  /  DATA BITS/PIXEL
IRAFTYPE= ’INTEGER ’  /
CCDPICNO=                   53  /  ORIGINAL CCD PICTURE NUMBER
ITIME   =                  600  /  REQUESTED INTEGRATION TIME (SECS)
TTIME   =                  600  /  TOTAL ELAPSED TIME (SECS)
OTIME   =                  600  /  ACTUAL INTEGRATION TIME (SECS)
DATA-TYP= ’OBJECT (0)’  /  OBJECT,DARK,BIAS,ETC.
DATE-OBS= ’05/04/87’  /  DATE DD/MM/YY
RA      = ’13:29:24’  /  RIGHT ASCENSION
DEC     = ’47:15:34’  /  DECLINATION
EPOCH   =                 0.00  /  EPOCH OF RA AND DEC
ZD      = ’22:14:00’  /  ZENITH DISTANCE
UT      = ’09:27:27’  /  UNIVERSAL TIME
ST      = ’14:53:42’  /  SIDEREAL TIME
CAM-ID  =                    1  /  CAMERA HEAD ID
CAM-TEMP=              -106.22  /  CAMERA TEMPERATURE, DEG C
DEW-TEMP=              -180.95  /  DEWAR TEMPRATURE, DEG C
F1POS   =                    2  /  FILTER BOLT I POSITION
F2POS   =                    0  /  FILTER BOLT II POSITION
TVFILT  =                    0  /  TV FILTER
CMP-LAMP=                    0  /  COMPARISON LAMP
TILT-POS=                    0  /  TILT POSITION
BIAS-PIX=                    0  /
BI-FLAG =                    0  /  BIAS SUBTRACT FLAG
BP-FLAG =                    0  /  BAD PIXEL FLAG
CR-FLAG =                    0  /  BAD PIXEL FLAG
DK-FLAG =                    0  /  DARK SUBTRACT FLAG
FR-FLAG =                    0  /  FRINGE FLAG
FR-SCALE=                 0.00  /  FRINGE SCALING PARAMETER
TRIM    = ’Apr 22 14:11 Trim image section is [3:510,3:510]’
BT-FLAG = ’Apr 22 14:11 Overscan correction strip is [515:544,3:510]’
FF-FLAG = ’Apr 22 14:11 Flat field image is Flat1.imh with scale=183.9447’
CCDPROC = ’Apr 22 14:11 CCD processing done’
AIRMASS =    1.08015632629395  / AIRMASS
HISTORY ’KPNO-IRAF’  /
HISTORY ’24-04-87’  /
HISTORY New copy of one035.imh
HISTORY New copy of one035
HISTORY New copy of m51
HISTORY New copy of m513
HISTORY New copy of m51
HISTORY New copy of dev$pix
END
```

12. Let’s check the image now. Do

```
imarith  image.short  /  new.short  div.short
minmax  div.short
```

The results of ‘minmax’ should be

```
div.short  [1,1] 1. [1,1] 1.
```

13. Something new! Lets’ make an ‘@file’. Type

```
files  image.r*  >  fitsout
files  image.d*  >>  fitsout
type  fitsout
```

The file ‘fitsout’ should contain the following list. Note that the use of the ‘>’ sign redirects output to a new file. The ‘>>’ appends to a file.

```
image.real.imh
image.dbl.imh
```

14. Using the file that we just created, let’s add those images to our tape.

```
wfits  @fitsout  mtc  new-  bit=16
mtex  mtc
```

The ‘mtexamine’ task should now display the following.

```
File mtc[1]:
        185 2880-byte records
        Total 185 records, 532800 bytes
File mtc[2]:
        185 2880-byte records
        Total 185 records, 532800 bytes
File mtc[3]:
        185 2880-byte records
        Total 185 records, 532800 bytes
File mtc[4]:
        Total 0 records, 0 bytes
Tape at EOT
```

Type

```
devstat  mtc
```

and check the tape position. The following should be displayed.

```
# Magtape unit ‘mtc’ allocated to ‘jbarnes’ Sat 09:55:13 07-Jul-90
file = 4
record = 1 (EOT)
```

15. Let’s make one more check on ‘rfits’. Type

```
rfits  mtc  1,3  check
imhead  check*
```

and the following should be displayed from ‘imheader’. Note that we did not specify the ‘datatype’ parameter and ‘rfits’ chose an appropriate type for us for each image. As you probably remember, the first image on the tape was written in 9. and was ‘image.short’; the third image on the tape was written in 14. and was ‘image.dbl’ from our directory.

```
        check001.imh[512,512][short]: m51  B  600s
        check003.imh[512,512][real]: m51 double
```

The ‘minmax’ task should verify our results.

```
        minmax  check*

        check001.imh  -1.  19936
        check003.imh  -0.7057446  19935.71
```

16. Hopefully all went well to this point. Let’s clean things up a little.

```
        rew  mtc
        deallocate  mtc
        dir
```

Remember that if you want to delete any images you must use the task ‘imdelete’. The task ‘delete’ will delete your text files. If the improper task is used to delete files a warning message is printed and no files are deleted.

If discrepancies occur during any of these steps, please look at the examples closely. It might be advisable to backtrack a few steps and verify things again. If the discrepancies are repeatable there could indeed be a problem. Please document the discrepancy and feel free to give us a call, if some advice or help is needed.