test of the main lsfit functions

Synopsis
test()

1.5.31 datafilter - Tool

Package utility
Module mathematics

Filtering operations on data arrays
include "image.g"

Constructors

datafilter	Construct a Datafilter tool

Functions

done	Destroy this tool
filter	Filter a data array
medianclip	Median clip a data array

Summary
The Datafilter `tool` provides some simple filtering operations for Glish arrays. Here is an example where we create a noisy sinusoid with outliers. First we filter the data; you can see how the median filter removes outliers (as well as compressing the noise). Then we also median clip the original data showing how the outliers are effectively removed.

Example

```
# Generate Gaussian distribution

- include 'randomnumbers.g'
- n := 500
- r := randomnumbers();
- g := r.normal(0.0, 0.5, n);
- r.done()
```

```
# Generate data ; sin wave plus noise

- x := (1:n) / 10 / pi
- y := sin(x) + g
#
- include 'pgplotter.g'
- p0 := pgplotter ()
- p0.plotxy (x, y, title='Sin + noise', plotlines=F)

# Now add some outliers

- ii := 1:n
- for (i in ii[ii%50 == 0]) {
  y[i] *:= 20;
}
#
- p1 := pgplotter()
- p1.plotxy (x, y, title='Sin + noise + outliers', plotlines=F)

# Generate Datafilter tool

- include 'datafilter.g'
- df := datafilter();

# Median filter returns filtered data

- y2 := df.filter (y, method='median', width=10)
#
- p2 := pgplotter()
- p2.plotxy (x, y2, title='Median filtered', plotlines=F)

# Median clip returns a Boolean mask

- m := df.medianclip (y, width=10, clip=5)
#
- p3 := pgplotter()
- p3.plotxy (x, y, title='Median clipped', plotlines=F, mask=m)
```

datafilter - Constructor

Package	**utility**
Module	**mathematics**
Tool	**datafilter**

Construct a Datafilter tool

Synopsis
datafilter()

Returns
datafilter `tool` or fail

Description

Construct a Datafilter tool.

Example

```
- include 'datafilter.g'
- df := datafilter()
```

---

datafilter.done - Function

Package	**utility**
Module	**mathematics**
Tool	**datafilter**

Destroy this tool

Synopsis
done()

Returns
T or fail

Description

If you no longer need to use the Datafilter `tool`, calling this function will destroy it.

Example

```
- include 'datafilter.g'
- df := datafilter()
- df.done()
```

---

datafilter.filter - Function

Package	**utility**
Module	**mathematics**
Tool	**datafilter**

Filter a data array

Synopsis
filter(data, width, method, progress)

Arguments

data	in	Input numeric data array Allowed:Numeric Glish vector
width	in	Width of running median filter Allowed:Positive integer Default: 5 points
method	in	The method type Allowed:String (case insensitive) from 'median' or 'mean' Default: 'median'
progress	in	See progress reports on median filter every progress points Allowed:Positive integer Default: No reports

Returns
Numeric vector or fail

Description

This function running-filters a data array with a variety of kernels over the specified width. It returns the filtered data vector.

As yet, input masks are not handled.

Example

```
- include 'datafilter.g'
- df := datafilter()
- d2 := df.filter(d, method='median', width=10)     # Data array 'd' assumed pre-made
```

---

datafilter.medianclip - Function

Package	**utility**
Module	**mathematics**
Tool	**datafilter**

Median clip a data array

Synopsis

medianclip(data, width, clip, progress)

Arguments

data	in	Input numeric data array Allowed:Numeric Glish vector
width	in	Width of running median filter Allowed:Positive integer Default: 5 points
clip	in	The clipping amount Allowed:Positive numeric value Default: 5
progress	in	See reports on median filter computation every progress points Allowed:Positive integer Default: No reports

Returns

Boolean mask or fail

Description

This function generates a median clipped mask from a data array. It does this by computing the median filtered array (d1). Then it finds the absolute differences from this (d2). Then it computes the median filter of d2 forming d3. The absolute differences (d2) are then compared point by point with d3. Any value $d2 > clip * d3$ is flagged as bad (mask==F).
The function returns a Boolean mask vector. Values of T are good and values of F are the ones detected as outliers (bad).
As yet, input masks are not handled.

Example

```
- include 'datafilter.g'
- df := datafilter()
- m := df.medianclip(d, width=10, clip=5)     # Data array 'd' assumed pre-made
```

1.6 misc - Module

Miscellaneous utilities
include "misc.g"

Description
The misc module contains a number of generally useful tools:

`misc` contains a grab-bag of useful functions.

`os` contains system-specific functions like file and directory handling.

`aipsrc` is available for querying the AIPS++ resources (".aipsrc" files). `aipsrcedit` allows interactive editing of AIPS++ resources.

`getrc` is available for querying the AIPS++ resources (".aisprc" files) during system startup to avoid recursive dependencies involving aipsrc.

`sysinfo` allows interrogation of other miscellaneous AIPS++ information.

`printer` is available for printing files and Glish values.

`assay` is available for testing the installation.

`stopwatch` implements a set of timing functions in the style of a traditional stopwatch.

`sh` creates a persistent shell client, amortizing startup over many different function calls.

`substitute` is available for substituting Glish variables and expressions in a string.

`itemcontainer` is a generic closure tool container.

`aips2loc` plots the lines of code in AIPS++ as a function of time.

`ftp` allows ftp operation from inside AIPS++.

`make` provides a facility equivalent to Unix make inside Glish.

`executive` provides a facility for running scripts in a queue.

Tools

aipsrc	tool to find aips resources from .aipsrc files
aipsrcedit	Tool to edit aipsrc file
assay	Assay test and demo functions
executive	Tool for running scripts in a queue
ftp	Tool to ftp files
getrc	find aips recources from .aipsrc files using the getrc utility