1.Strings

General goal: the ability to produce automatically generated labels based on what is being plotted.
For example:
X/Y Axis for vcurves of rvectors in quantity [units] format. eg Azimuth [^o] vs Time [Seconds]
X/Y Axis for psds: In the correct units based on base units, eg Amplitude [V/Hz^{1/2}] vs Frequency [Hz]
X/Y Axis for Histograms : counts vs Aximuth [^o]

To do this, vectors need to export the following things as automatic scalars or automatic strings in the same way as vectors currently export automatic scalars:
[datasource/vectorname/QUANTITY] (eg, "Time"; default to tagname)
[datasource/vectorname/UNITS] (eg, "Seconds"; default to "")
[datasource/vectorname/RATE] (eg, 100.16, default to 1.0)
[datasource/vectorname/RATEUNITS] (eg, "Hz", default to "")

These need to be defined for all vectors from their source:
for RVectors, from their data source (see section 2.3)
for SVectors, from their owner: eg, psds need to generate Quantity and Units
note: if we were infinitely ambitious, equations could even try to generate these!

Datasources will also be able to export datasource-wide strings and scalars, as described in section 2.2.

2.Data source changes

2.1.Data sources with heirarchy

Some data sources are broken up heirarchically (eg, piolib, some fits files). To support this:
Option 1 (preferred):

- the URL/File requester will be modified to allow browsing into the data source
    - auto completion in the line entry
    - the dialog is enhanced/modified
- the datasource name is not considered complete until a group has been selected.
- define '**:**' as a special character to delineate groups. eg "`/data/file.fits:hdu1`"
- if the datasource only has one (named) group, this group is silently assumed. So, with a *fits* file with one HDU, named *hdu1*, either `/data/file.fits` or `/data/file.fits:hdu1` will work. If the fits file has *hdu1* and *hdu2*, then `/data/file.fits` will be incomplete.
- Should we allow arbitrarily deep Heirachies?

Option 2:

- If a data source which has groups is selected, a group selection combo appears.
- This option does not seem to allow arbitrarily deep heirarchies.

2.2.Datasource-wide Scalars and Strings (per-file metadata)

Data sources must provide all 4 primitive types: Matrixes, Vectors, Scalars, and Strings.
Datasource-wide 'metadata' will appear as scalars or strings which can be accessed in the same way that Vectors and Matrixes currently are (ie, with the newScalarDialog or newStringDialog.) Like Vectors, they are not read automatically.
Data sources will provide some automatic primitives:

Automatic Vectors:

- `INDEX`: counts frame numbers. If you ask for f=200, n = 5, you get 200,201...204. It can be from 0 to LASTFRAME-1.
- `_TIME`: Time. The default data source should provide a dialog which allows this to be defined in terms of existing other vectors. This default behavior can be overridden by individual data sources. Defaults to `INDEX`.

Automatic Scalars:

- `LASTFRAME`: the number of the last defined frame in the source.
    - The datasource may chose to make this number be the size of the largest vector, if it decides to allow vectors of different number of frames. It must always be possible to read any vector up to `LASTFRAME`, even if the data source decides to return `NaN`s.
- `FRAMERATE`: Rate for frames. Defaults to 1, but can be overridden by the datasource. Used by PSDs.

Automatic Strings:

- `SOURCENAME:` a readable name of the data source. Typically the full ile name. For indirect files, it is the file that the indirect file points to.
- `MODTIME:` the date/time the data source was last modified, if available. '\0' otherwise.

`RATEUNITS:` units for `FRAMERATE`: defaults to '\0'

2.3.Scalars and Strings associated with individual vectors

Data sources may provide scalars and strings associated with individual vectors (eg, calibration coefficients, units). The scalars should be added to the list of scalars already attached to vectors (max, min, etc), and the strings should be put in their own analogous list.

Automatic Field Strings:

`NAME:`The editable tagname of the field (so it can be used in labels)

`UNITS`: The units of the vector. Defaults to "", but should be defined so it can be used.

`QUANTITY`: What quantity the vector is measuring (eg, "Time", "rotation rate".)

`RATE`: The sample rate of the vector. For RVectors, will normally be [datasource/FRAMERATE]*samples_per_frame or [datasource/FRAMERATE]/skip.

`RATEUNITS`: The units of the sample rate. For RVectors will be [datasource/RATEUNITS]

2.4.Default datasource configuration dialog

By default, the datasource configuration dialog needs to be able to define the TIME related settings.

```
_TIME vector:     [field selection combo box]
Interpret as:     [interpretation combo box: CTIME, JD, Etc...]
Frame Rate:       [float entry]     Units:   [rate units]

[] Apply as default to all files ending in [.yyz]
[] Apply to this file only
[] Apply to all (ascii) files
```

Individual data sources may add to or override these settings. The base class of the data source should provide tools for these things. In particular, storing defaults to associate with files and extensions. Wherever it makes sense, data sources should adapt this feature. Some data sources may have their own way of identifying sub-types, so this may need to be extended.

3.KstPlot overlays

All capabilities of kst1.4 need to be ported to kst 2.0
We also need the ability to overlay different logical plots into the same space. So:

All plots live in a rectangular content region (the part of the plot where lines/images appear).
Plots may be non-Cartesian.
Plots need to be able to draw non-cartesian axis (eg, polar plot, etc).
Plots need to be able to project an image or a curve into its space.
Plots need to report/accept their ranges in the natural units of the plot (eg, r and θ for a polar plot).
From the user's point of view, the projection needs to live with the plot, not the curve/image.
When in a grid, plots need to be able to align their borders/content region with neighboring plots
Plots need to be able to report their minimum required borders.
Plots need to be able to accept a border size to draw into.
Plots need to be able to suppress any of their borders.
If a plot has suppressed a border, its minimum border size will be zero for the suppressed border.
If a plot (and all overlays) have suppressed borders, then, when on a grid, they need to dynamically resize themselves to keep the same size content region as neighboring plots.
Child view objects with active points inside the content region need to be aligned with the content region, not the overall plot.