POD Translation by *pod2pdf*

ajf@afco.demon.co.uk

Tkg2_1.5

Table of Contents
Tkg2_1.5

TKG2 BY WILLIAM H. ASQUITH

In the beginning there was a free charting package called USGS-G2 (G2) by the remarkable Jim Fulton of the Water Resources Division of the U.S. Geological Survey. Though I never knew him, some of those who knew him said that he was nearly immortal. For nearly a decade G2 was the defacto standard for the data visualization and charting needs of hundreds of USGS scientists in thousands of reports and journal articles. The number of figures both published and unpublished using G2 is likely uncountable. G2 inconjunction with Abobe FrameMaker were a devastating combination for scientific pursuits. Furthermore, G2 files when coupled with Perl or shell programming provided a very powerful scripting environment without having to learn some archaic graphics library. Scripting G2 was not the most trivial activity. However, it was not unusual for a single G2 template to be used to generate hundreds of publication ready figures by a dozen line Perl program and hundreds of ASCII data files. This was made possible because G2 could be linked to external data files and was relatively easy to use for simple to incredibly complex batch processing jobs.

Sadly, due to technological changes and agency-wide move toward shrink wrapped software, in-house development and distribution of software packages fell out of favor. Thus, G2 and the G2 graphics model were allowed to gradually disappear. G2 was not going to be ported from its SmallTalk environment on Data General Unix to other operating systems namely NT and Solaris.

I was heartbroken and worried extensively since I first learned of the demise of G2. I simply could not see how I could execute my job duties without a package like G2 and now Tkg2. My concerns were so great that over 1,000 personnel hours have been devoted to Tkg2 since the Fall of 1998 in studying Perl and actual programming and over 3,000 hours of total time have been devoted to Tkg2. The fact of the matter is that G2 was the mission critical tool for me and many other USGS scientists. G2 represented a complete package for nearly all 2-D charting needs and G2 could have nicely fit into the bag-of-tricks for nearly all scientists without ever needing to be upgraded and it was totally FREE. Unfortunately, because it was in SmallTalk it was hard to modify and extend.

In the spirit of the original G2 package, I have resurrected G2 in Perl/Tk from scratch coping (with many deviations both purposeful and otherwise) of the published G2 interface. A fantastic number of functions and features not found in the original G2 have been built into Tkg2 by feedback from the user community. It is my hope that Tkg2 will someday see widespread open source (free) use on all Perl capable operating systems including those embedded in my beloved agency. Tkg2 has become a serious application, just spend sometime sand push its limits. With Headquarters support over the last couple of years, Tkg2 has become the main graphic rendering engine for ADAPs and other custom applications built on top of the NWIS system.

I can truly say that it has been a pleasure to serve mother Survey and to show that the best software can only be designed by the users themselves.

```
William H. Asquith, wasquith@usgs.gov
  Hydrologist, Statistician, Atmospheric Scientist,
  Computer Programmer, Student, Teacher,
  Husband, and Father. . .
USGS, Austin, Texas, January 2002
```

WHAT IS TKG2?

Tkg2 is a full featured, interactive, and batch processing capable 2-D charting package whose goals are to provide professional quality camera-ready charts for immediate publication. The intended users of Tkg2 are nonprogrammers who need to exploit several of its unique features, such as producing hundreds of charts without having to possess programming skills beyond elementary Perl or some other scripting language. Through more complex scripting, Tkg2 is easily extended by many intermediate to advanced Perl or Shell programmers into a powerful charting engine.

TKG2 IS PERL

Tkg2 is written entirely in Perl and has in excess of 90 written-from-scratch modules. The Tk graphics module provides the graphical foundation of the application, and provides widgets for over 30 dialog boxes and the canvas for drawing. Several other modules are used for data persistence, date and time

calculations, and other internal features. The primary output of Tkg2 is postscript as that is the native output of the Tk::Canvas. However additional formats are supported by some non-Perl utilities, which provide support for MIF, PDF, and PNG.

PLOTS AND ANNOTATION

Tkg2 can produce one or more charts on a single canvas, and charts are permitted to overlap. Tkg2 supports numerous plot types and most can be combined. Tkg2 has five continuous axis types: linear, log, normal probability, Gumbel probability, and time; and has two categorical axis types: stacked discrete and clustered discrete. Time axis support provides professional looking ticking and labeling from years down to seconds. Tkg2 supports several plot styles which include variations of: scatter, line, bar, Y-error line, X-Y error line, text (annotation), accumulation, and shade. Tkg2 supports very fine granular control of each linear, log, probability, and time axis. Line, symbol, and text annotation is supported, and numerous symbol types are provided. Finally, Tkg2 has the beginnings of incredible box plot power.

INPUT DATA

ASCII Data is inputted into Tkg2 through simple text files, but the format of the text files can vary greatly as long as the data is delimited in a column and row fashion. Missing values are appropriately supported. Data can either be hard loaded or loaded at run-time, and files can be specified in either an absolute or a relative fashion. Tkg2 can either dynamically determine variable (a column of input data) types, which are number, string, or time, or variable types can be directly specified to increase performance. Tkg2 can recognize virtually any time format. The `checktime.pl` program at Tkg2/Util can help you learn what time formats can be parsed by the Date::Manip module and hence Tkg2 itself.

BATCH PROCESSING AND APPLICATION GLUE

A wide variety of tasks can be performed with Tkg2 using the command line including file name globbing, printer spooling, exporting, and recursive directory processing. Tkg2 has full support for reading Tkg2 files from STDIN and producing them along STDOUT; thus, Tkg2 can be inserted in to pipelines. Tkg2 even has three simple display modes that permit only the viewing and exporting of graphics and does not permit interaction with them as permitted in its normal operational mode. Finally, Tkg2 has an operational mode that traces program entry of the major methods and subroutines to assist other developers in understanding the underlying data model.

TKG2 MAIN DIALOG HELP

This is an extremely brief summary of Tkg2. More documentation can be found in the plain old documentation files (POD) found in the Tkg2/Help directory. Help can be accessed from a HELP menu button and from the command line via the —help option. The —help option does take arguments to help you find documention further. The USGS internal Tkg2 web site *http://tx.cr.usgs.gov/tkg2* should contains a large PDF and an HTML version of the documentation as well. This documentation file is located at `Tkg2/Help/main.pod`.

WHAT IS TKG2?

Tkg2 is a full featured, interactive, and batch processing capable 2-D charting package modeled after the venerable USGS-G2. Tkg2 is entirely written in Perl and the Tk graphics module. Other nonstandard Perl modules are required *http://www.CPAN.org*. Tkg2 features include: linear; log; normal and Gumbel probability; time series; x, y and x-y error bar; x, y, and x-y error limit; bar, and text plots. Other plot types include remarkably flexible boxplots, x or y accumulator plots, and time variation plots. Other features include nearly unlimited ASCII file input types, hard loading or run-time loading of data, and use of absolute or relative path names in the input data files. Exported graphics formats include Framemaker Interchange Format (MIF), Portable Network Graphics (PNG), Portable Document Format (PDF), and Postscript. Nearly eighty command line options are available and an external instruction language to control Tkg2 objects is provided to support scripting and application glueing. Tkg2 has a fantastic array of cool and unique features built because only the user base as suggested which features should be included. Tkg2 might be rough on the edges, but it is extremely stable and very powerful.

GETTING STARTED

`tkg2 –help` will provide a brief tutorial and list the available command line arguments. You may view or print the command line help from /usr/local/Tkg2/Help/CmdLine.pod. You might have to tweak line-wrapping on.

ADDITIONAL RESOURCES

The HELP menu button on the right hand side of every Tkg2 drawing sheet accesses additional help features. Screen shots can be viewed with the SCREEN SHOTS menu to the left of the help menu.

AUTHOR

William H. Asquith, wasquith@usgs.gov, Austin, Texas, USA

BUG REPORTS

Please report bugs, suggestions, and thanks pertinent to Tkg2 only to wasquith@usgs.gov. Please use whatever bug reporting mechanism exists for the systems that are dependent on Tkg2.

REQUIREMENTS

Tkg2 has been built and fully tested on **Solaris**, **Linux**, and **MacOSX** with the following Perl and Perl modules:

```
perl 5.8.+
Tk800.027
Tk::Pod 4.26
Text::Wrap
Data::Dumper-2.101
Storable-2.08
Date::Manip 5.42
Date::Calc 5.3
And the dependencies on these modules.
Other modules of the standard Perl distribution are used.
```

The following external utilities provide additional features:

```
ghostscript (Postscript viewing and conversion engine)

pstoedit    (postscript to MIF converter among other things)
    http:/www.pstoedit.net/pstoedit/

ps2pdf      (Postscript to PDF utility)

acroread    (Acrobat Reader, PDF viewing)
```

The `pstoedit` version 3.21 for Solaris binary now (Jan, 11, 2002) requires a shared library for execution. This library is distributed with the binary and is called `libplotter.so.2`. This library should be copied to `/usr/opt/lib` on the Solaris machines and then a soft link in `/usr/lib` need to be created `ln -s /usr/opt/lib/libplotter.so.2 .`. As long as the library can be found along the LD_LIBRARY_PATH environmental variable, `pstoedit` should work. Please make sure that the `libplotter.so` name space is not already taken!

```
% cd /tmp
% unzip pstoedit_bin_solaris.zip
% echo "pstoedit is binary and libplotter.so.2 is a library"
% su
# cp /tmp/libplotter.so.2 /usr/opt/lib/.
# cd /usr/opt/lib
#  ls -l libplotter*
-rwx------   1 root     other    1209720 Jan 11 06:51 libplotter.so.2
# chmod 0755 libplotter.so.2
# chgrp root libplotter.so.2
# cd /usr/lib
```

```
# ln -s /usr/opt/lib/libplotter.so.2 .
# ln -s ./libplotter.so.2 libplotter.so
# exit
% cd /tmp
% ./pstoedit
returning 1 for drvsvg
returning 1 for drvmif
returning 1 for drvcgm
returning 1 for drvcgm
returning 1 for drvcgm
pstoedit: version 3.31 / DLL interface 108 (build Jan  4 2002) :
  Copyright (C) 1993 - 2001 Wolfgang Glunz
  No backend specified

usage: pstoedit [-help] [-bo] [-df fontname] [-dt] [-adt] [-dis]
[-flat nn] [-fontmap mapfile] [-gstest] [-include file] [-merge]
[-pagesize pagesize(e.g. a4)]
[-scale nn] [-nb] [-nc] [-nomaptoisolatin1] [-noclip] [-nq] [-nfr]
[-page nn] [-psarg string] [-pti] [-pta] [-rgb] [-rotate angle]
[-sclip] [-split] [-ssp] [-t2fontsast1] [-uchar char ] [-v]
-f format [infile [outfile]]
```

TKG2 COMMAND LINE

Tkg2 has an incredibly extensive and flexible command line to help everyone get their jobs done. The command line is best viewed by % tkg2 —help. This documentation file will only list the options that are available.

```
--autoexit or --autoexit=<integer>

--batch
--batchsave

--checkcolors
--clear
--colormode=<color | mono>
--cycle=<integer --test>

--debug=<stdout | file>
--destination=<printer>
--display=[host]:server[.screen] or
--DISPLAY=[host]:server[.screen]
--dnload_data  ** NOT IMPLEMENTED YET
--drawdata or --nodrawdata
--dumpboxes

--echocommandline
--exportfile[v]=<filename>
--exportrotate=<0 | 90>
--exportview=<integer, pixels per inch>?

--fixcolors ** NOT IMPLEMENTED YET
--format=< (mif | MIF) |
           (pdf | PDF) |
           (png | PNG) |
           (ps | PS) >

--geometry=<'valid unix geometry string'>
```

```
--glob=<'pattern'>?

--height=<X.X inches>
--help [optional topics]
--home=<directory path>

--importdata
--inst=<file name | - >
--instv=<file name | - >
--instb4data

--justdisplay
--justdisplayone

--mkplot=<string of margins, #x#x#x#>?
--mktemp=<dimension string, #x#>
--megacmd_args=<string>
--megacmd_keep
--megacmd_show
--message=<string>

--nobind
--nodec
--nodialogrescale
--noexport
--nofieldcheck
--nomenus
--nomw
--noprint
--norulers
--nosave
--noundo
--nozoom2unity

--optimize

--pause=<integer>
--presenter
--presenterlabels=<string, delimited by :>
--pretty_data **  NOT IMPLEMENTED YET
--printer_queue=<string>

--readonly
--redodata
--redoinst
--relative_paths or
--norelative_paths ** NOT IMPLEMENTED YET

--scaling=<real no. > 0>
--showme=<real no. > 0>
--showmesubs
--splash or --nosplash
--stdin
--stdout

--test=<integer>
--time_cache_delete
--time_cache_ignore
```

```
--time_cache_nosave
--time_cache_view

--walkatree=<directory>?

--verbose or --noverbose
--version or --v or --V
--viewconfig

--width=<X.X inches>
--withdraw
--withdrawmw or --nomw

--zoom=<real no.>
```

TKG2RC FILES

This POD file provides detailed description of the tkg2rc files that reside with the Tkg2 distribution, in the usual system dependent places, and the user‘s home directory. This documentation file is located at **Tkg2/Help/Tkg2rc.pod**.

INTRODUCTION

The tkg2rc files are very similar to the typical Xresources files. The tkg2rc file is almost uniformally used to control the %::TKG2_CONFIG global hash. The tkg2rc files are handled by the Tkg2/Tkg2rc.pm module. A typical parameter is set in the following fashion.

```
[application]*[parameter]:[value]

         Tkg2*height:        4
```

In this case the height of all pages is set to 4 inches regardless of the original height in the tkg2 graphic files. Leading ‘#’ and ‘!’ are treated as comment lines and politely ignored.

Three potential tkg2rc files are read in successive fashion. Each file potentially overrides settings from the previous. In most cases resources set by the tkg2rc file can be overridden by command line options (see tkg2 —help). The first tkg2rc file that is read and normally would always be present is the **tkg2rc** file that comes with the tkg2 distribution. The file would normally be located at ‘/usr/local/Tkg2/tkg2rc’. The second tkg2rc file read is the system or site specific version as is called **tkg2.rc**. This file should contain parameters required by the system administrator for tweaking operation. This file location is partially operating system dependent. A summary of os-dependent path prefixes for the tkg2.rc file follows:

```
Solaris: /usr/openwin/lib/app-defaults/tkg2.rc

Linux:  /usr/X11R6/lib/X11/app-defaults/tkg2.rc
```

(at least on Red Hat 6.+)
The third tkg2rc file read is in the user‘s home directory and uses the familiar dot file naming convention, **.tkg2rc**. This file is intended primarily for the user to override parameters set by the other files or more commonly by command line options provided by tkg2 wrapping programs. For example, if a wrapping script calls tkg2 for the user in a display only mode and some of the fonts on the canvas are too small to read, then the user could choose to scale up all fonts by 20-percent by setting Tkg2*zoom: to 1.2 in their .tkg2rc file.

PARAMETER LIST

The following list of parameters are those currently supported in a tkg2rc file. Some Tkg2*parameter have a single argument whereas others have a list. Furthermore, a couple of parameters when repeated continue to add argument values to themselves. Just as a parameter can occur in each of the three tkg2rc

files, a parameter can be unlimited repeated in the file, but the last parameter read sets the value. Some of the parameters are composed of two or more words. These are separated by an underscore '_'. The POD markup by the tkpod utility does not display underscores when the parameter is made in bold text. For example, the parameter 'Tkg2*delete_loaded_data' is headed with 'Tkg2*delete loaded data'. The tkg2rc files will not accept that. HTML or other POD markup does not exhibit this behavior.

Tkg2*colors:—*list context, repeating adds to list*
Colors sets the list of available colors seen in the dialog box color selection widgets.

```
Tkg2*colors:    red green white
Tkg2*colors:    blue black
```

Tkg2*debug:—*scalar context, last setting used, '0', 'stdout' or 'file'*
Debug toggles the built in reporting mechanism for tkg2. If stdout is used, then debug messages printed along STDOUT through the BUGS filehandle. When the debug value is 'file' then debug messages are appended to /tmp/tkg2.bugs.

```
Tkg2*debug:  0
```

to turn off, or

```
Tkg2*debug:  1
```

to turn on.

Tkg2*delete_loaded_data:—*scalar context, last setting used, '1' or '0'*
When a file is saved by tkg2, a method is called on the template (page) that removes all dynamically loaded data from before the template is dumped into a string and then written to a file. By default delete_loaded_data is 1 or true, but if the user requires that dynamic loading be turned off, then set delete_loaded_data to 0 or using the —importdata command line option. A conscious decision to not make this feature available via the GUI has been made.

```
Tkg2*delete_loaded_data:      0
```

to set to false, or

```
Tkg2*delete_loaded_data:      1
```

to set to truth.

Tkg2*delimiters:—*list context, repeating adds to list*
Delimiters builds the list of strings that are used by tkg2 to `split` files into columns of data. Normally, the delimiters supplied by the Tkg2/tkg2rc file will be sufficient for just about all installations. The user is given the ability through the GUI to select custom delimiters, so relatively little need exists for controlling these by the resource file method. Note: foreign tkg2 files might use one or more delimiters that an installed version does not explicitly show. This situation is **not** a problem because the tkg2 file stores the exact string that it needs to delimit each file.

```
Tkg2*delimiters:  \s \s+ \s\s+ \t , : |
```

This example provides delimiters of one space, one or more spaces, two or more spaces, tab, comma, colon, and pipe.

Tkg2*fileformat:—*scalar context, last setting used, 'DataDumper', 'Storable'*
Fileformat sets the default file format for new template creation. The default is to use the hash-like DataDumper format. This is the preferred format as it is highly readable ASCII, cross-platform,

archivable, and easily editable by users. The file format can be changed using the File menu radiobuttons in the Save section.

```
Tkg2*fileformat:    Storable
```

Tkg2*fonts:—`list context, repeating adds to list`
Fonts are the font families that are to be shown in tkg2 font family widget selection buttons. Great care must be taken when selecting font families to use as available fonts on a given platform or X server vary greatly. There are even issues involving the compatability of X server and X client font support. The default list (shown below) is recommended and testing has shown little trouble across platforms. Perhaps the greatest danger is using a font in a tkg2 file on one platform that isn't supported on another. Currently tkg2 makes no provision that a given font can actually be rendered until run-time. Errors can the occur. Change the default font families at your own risk.

```
Tkg2*fonts:       Courier Helvetica Times Palatino Symbol
```

Tkg2*geometry:—*scalar context, last setting used*
Geometry specifies the dimensions and location of the window provided by the window manager containing the template (page). Either just the dimensions, just the location, or both dimensions and location can be controlled. The format for the string is the usual X-windows geometry string `(\d+[x]\d+[+-]\d+[+-]\d+)`. The '\d+' denotes an integer. The [x] is the character x and [+-] indicates the plus or minus sign. Here are some examples:

```
Tkg2*geometry: 300x450
```

This sets a 300 pixel width and 450 pixel height.

```
Tkg2*geometry: 45x300-100+20
```

This sets 45 pixels by 300 pixels placed 100 pixels from left and 20 pixels down from top.

```
Tkg2*geometry: +100-20
```

This sets a default width and height placed 100 pixels from right and 20 pixels up from bottom.

```
Tkg2*geometry: 100x500
```

This sets 100 pixels by 500 pixels at the default location on the screen.
Some variation on whether you will detect the effects of the geometry depend on the window manager that you are using and the natural size of the tkg2 sheet.

Tkg2*height:—*scalar context, last setting used*
Height specifies the height of the template (page) in inches that is to be used regardless of the value in the tkg2 file. All graphic elements are remapped on the drawing canvas according to their percentage down from the top on the original. This feature is useful because a single tkg2 file can take on an essentially unlimited range of heights. The height can be equivalently set from the command line with —height=value (see —help). If the value is 0 in the tkg2rc file, then the original height is used. For width adjustment, see width later in this file and the —width=value option in the command line help.

```
Tkg2*height:   4
```

This example sets the height of the page to 4 inches.

Tkg2*increment_dialog_fonts:—*scalar context, last setting used*
Increment dialog fonts increments the small, medium, and large font sizes used in tkg2 dialog boxes. Most of the time this setting is not ever required, but occassionally on large monitors with high

resolution, font size might be too small for older eyes. The increment can be either positive of negative integers and are added to all fonts sizes. For example, `Tkg2*increment_dialog_fonts: 3` would increment each font up by 3 points. The X server will probably use the closest available font for rendering, so don‘t expect to see a 17 point or other atypical font sizes.

```
Tkg2*increment_dialog_fonts: 2
```

Tkg2*linethicks:—*list context, repeating adds to list*
Linethicks sets the line thicks values in inches that are to be made available through the GUI on the line thickness or width widgets. The default values shown in the example here have been carefully selected as each should be visually distinct in postscript output. These setttings will less often show differences on the monitor.

```
Tkg2*linethicks:  0.005i 0.010i 0.015i 0.020i 0.025i 0.035i
Tkg2*linethicks:  0.045i 0.055i 0.065i 0.075i 0.085i 0.095i
```

Tkg2*nozoom2unity:—*0|1*
This parameter turns off the default behavior that —zoom is set to unity prior to postscript rendering. This setting was added to bypass the switch to —zoom=1. The checkbuttons on the Exporting and Printing dialog boxes are properly switched off—nice touch. There is also a —nozoom2unity command line option. The addition of the nozoom2unity feature is important for certain batch processing tasks.

```
Tkg2*nozoom2unity: 0
```

Tkg2*plotting_position_coe:—*scalar context, last setting used*
The plotting position coefficient is used for direct probability calculations from a single column of inputted data. A global variable is set and remains in effect until potentially changed with the next launching of tkg2. This might not be the best design model, but a more elegant solution remains elusive. The default value is 0.40, the Weibull plotting position coefficient. See a good book on statisics and frequency analysis for further discussion.

```
Tkg2*plotting_position_coe:   0.40
```

Tkg2*printers:—*list context, repeating adds to list*
Printers sets the printers to be visible in the GUI regardless of whether the host can actually talk to a particular printer. The printers list has been superceded on Solaris by the use of the `‘lpstat -v’` command. This command might not be available on other systems. The last printer identified in the tkg2rc file is used as the default and shown at the top of the dialog box.

```
Tkg2*printers:  hpblackandwhite  hpcolor
```

Tkg2*queue_options:—*scalar context, last setting used*
Additional command line arguments for the printer queue, usually plain-old and familiar ‘lp’ can be set by the queue_options token. The value is set as one large string and not parsed, and the individual options checked. No checking on the validity of the arguments therefore is made by Tkg2. You will have to rely on the whether it works or not. An example usage follows.

```
Tkg2*queue_options:   -o11x17 -y landscape
```

You can check that the arguments get passed to the queue by invoking Tkg2 with the `–verbose` command line option. An example output is shown below.

```
% tkg2 -batch -verbose multiplots.tkg2
Tkg2-RenderPostscript: 11 x 8.5,
              /home/wasquith/xxxxxxxtkg2printfile, color
Tkg2-correctTkPostscript for psfile =
             /home/wasquith/xxxxxxxtkg2printfile
```

```
Tkg2-Postscript2Printer
       Queue = lp -c -d lasertx -o11x17 -y landscape
               /home/wasquith/xxxxxxxtkg2printfile 2>&1
Tkg2-Postscript2Printer: successful spool
```

The contents of the queue options are shown in the 'Additional printer command line options' field of the print dialog box when it is launched from the File menu. Tkg2 inserts the `-c` and `-d printername` arguments on all spooling operations. Neither `-c` or `-d printername` can be modified by the user—only in Tkg2 source code. The queue however can be changed using the $PRINTER_QUEUE environmental variable of the user's shell. Normally, only specialized users would set this different from 'lp', so you can likely ignore the previous sentence. For example, 'pdq' is an alternative print spooler in the market place. Additionally, the queue options can not be set using the Tkg2 command line.

A final word of warning, because Tkg2 *always* inserts the `-c` and `-d` options to the queue, if you have a `-d printer` in the queue_options of the tkg2rc file, then this printer is used instead of the printer specified by the dialog box or by the `–destination` command line option. This is because the spooling command would look something like this.

```
Queue = lp -c -d lasertx -d elvis /home/wasquith/xxxxxxxtkg2printfile 2>&1
```

As far as your author knows, `elvis` becomes the printer destination instead of `lastertx`. Again the `–verbose` command line option will really help you in those weird circumstances in which something is not working.

Tkg2*redrawdata:—*scalar context, last setting used, '0' or '1'*
Redrawdata is used to toggle the 'Draw Data on Canvas Update' feature seen in the DataEdit menu during normal GUI operation. This is a handy feature for plots with large amounts of data because all editing to plots and other canvas components can be made without have to wait for the canvas to be totally redrawn each time. If the tkg2 file is saved or printed then the data is drawn regard less of value of redrawdata. The value can be altered at run-time by the checkbutton in the DataEdit menu. At the first rendering of the tkg2 file on the screen the data is drawn regardless of the value of redrawdata. The data is not drawn on subsequent updates of the canvas. The command line option —drawdata and —nodrawdata provide an equivalent toggling.

```
Tkg2*redrawdata:   0
```

to set to false, or

```
Tkg2*redrawdata:   1
```

to set to truth.

Tkg2*scaling:—*scalar context, last setting used, real number gt zero*
Scaling alters the start up scaling constant of Tk (and Tkg2) by multiplying the constant by value of the parameter. So if scaling is equal to 0.9, then the scaling constant of the whole Tkg2 session is 10 percent smaller. The scaling constant represents the number of pixels per point where point is 1/72 of an inch. This switch can make it easier to use tkg2 on a low resolution monitor and/or a small diagonal monitor. Tkg2 does have some limited built rules to alter scaling dependent on the host resolution (see Tkg2/Base.pm, ResolutionHandler). The author has limited machines to test monitor size and resolutions on, so the reader is encouraged to experiment with

```
% tkg2 --scaling=? --mktemp=8.5x11
```

until the page fits satisfactorily on their screen and report their monitor size and scaling value to `wasquith@usgs.gov`. Small monitor users will likely want to add the Tkg2*scaling to their home

directory tkg2rc file. Another way to control the scaling constant is with the —scaling command line option. A scaling value in the tkg2rc file overrides any built in rule, but is itself overridden by the command line option. A value of zero can be used within the tkg2rc file to toggle the scaling setting back to the built in rules without having to delete a line from the tkg2rc file.

```
Tkg2*scaling:    0.94
```

Tkg2*showme:—*scalar context, last setting used, real number*
Showme toggles the internal tracing of major subroutine entry along stdout. This is functionally the same as the —showme=f command line option. The real number is the second delay to add between subroutine calls. Showme is the primary tool to determine the location in the code of major tkg2 crashes.

Tkg2*splash:—*scalar context, last setting used, '0' or '1'*
Splash toggles a SPLASH filehandle to either /dev/null or /dev/stderr. Nearly every module in tkg2 prints an '=' to the SPLASH filehandle when it is first compiled by Perl. The SPLASH filehandle is a mechanism by which the user can see that tkg2 is actually doing something. SPLASH can also be controlled using the —splash or —nosplash command line options. The default is for the splash to be visible.

```
Tkg2*splash:   0
```

to turn the splash off, or

```
Tkg2*splash:   1
```

to turn the splash on.

Tkg2*verbose:—*scalar context, last setting used, '0' or '1'*
Verbose toggles a VERBOSE filehandle to either /dev/null or /dev/stdout. With verbose, the user can see which plot is being rendered and what data files are being read. Startup times are also reported. VERBOSE can also be controlled using the —verbose or —noverbose command line options. The default is for verbose to be turned off. See —help for more details.

```
Tkg2*verbose:   0
```

to turn verbose off, or

```
Tkg2*verbose:   1
```

to turn verbose on.

Tkg2*width:—*scalar context, last setting used*
Width specifies the width of the template (page) in inches that is to be used regardless of the value in the tkg2 file. All graphic elements are remapped on the drawing canvas according to their percentage right from the left on the original. This feature is useful because a single tkg2 file can take on an essentially unlimited range of widths. The width can be equivalently set from the command line with —width=value (see —help). If the value is 0 in the tkg2rc file, then the original width is used. For height adjustment, see height earlier in this file and the —height=value option in the command line help.

```
Tkg2*width:   4.55
```

This example sets the width of the page to 4.55 inches.

Tkg2*wm_override_pod_geometry:—*scalar context, last setting used*
Window manager override on geometry of pod placement occurs when this value is true. The default is false, which cause tkg2 to use the Tk geometry method to place the marked up pod in the upper left corner. Under some Linux systems using KDE this does not work correctly, so those user's might want to disable the geometry call with the following in their homespace tkg2rc file.

```
Tkg2*wm_override_pod_geometry:   1
```

Tkg2*zoom:
Zoom is a partially misnamed parameter. It is **not** a zoom in/out factor seen in just about every graphical application. Zoom is a multiplier on **font** sizes created by tkg2 when rendering text on the canvas. Zoom is a neat feature to use in combination with —width or —height. A regular pagesize (8.5x11) can easily be enlarged to say 20x20 and the fonts will be disproportionately small unless zoomed by say by a factor of 2. The command line option, —zoom, will override tkg2rc file value.

```
Tkg2*zoom:     1.1
```

TKG2 ENVIRONMENT AND CONFIGURATION SETTINGS

This file documents the TKG2_ENV and TKG2_CONFIG hashes. These hashes contain information about the environment that Tkg2 is running in and various configuration parameters too. Although the information contained here is slanted to documentation of the internals of Tkg2, these features are highly relavent to the user experience as how tkg2 interacts with the operating system and the user often dictated by settings described here. Many of the settings are highly integrated with the command line options and the Tkg2rc files. The reader should also consult those sections of the documentation.
This file is located at **Tkg2/Help/Environment.pod** and is accessible through the ViewENV button on the main tkg2 window.

TKG2 ENVIRONMENTAL HASH

-SCALING
-COMMANDLINE
-TKG2HOME
-PRINTER_QUEUE
-DISPLAY
-XRESOLUTION
-RC_FILES
    -SYSTEM
    -USER
    -DISTRIBUTION
-MEGACMD_FILES
-UTILITIES
    -TKMIFFIX_EXEC
    -BYPASS_MIFFIX
    -PNGVIEWER_EXEC
    -PSTOEDIT_EXEC
    -PDFVIEWER_EXEC
    -PS2PNG_EXEC
    -BYPASS_PSFIX
    -TKPSFIX_EXEC
    -PSVIEWER_EXEC
-HOST

The name of the host running Tkg2.

-SAFE_KEEPING_FOR_TKG2_FILE_HEADINGS

Each Tkg2 file has a heading before the actual contents that are needed to load and draw the graphics. This heading has several documentation comments and the beginnings of a shell script. Tkg2 files can have arbitrary code placed in their headings. When files are read in, the headings are stored in this environmental hash so that the heading can be reproduced verbatim when the tkg2 graphics are saved to either the same file name or another file name.

-DEFAULT_PRINTER

Name of the default printer.

-YRESOLUTION

-HOME

-EXECUTABLE

-BUGFILE

Location that the BUG filehandle is directed too, which is likely something like /tmp/tkg2.bugs. The filehandle is open to /dev/null unless the —debug command line option is used with a 'file' argument (—debug=file). The —debug option can also use 'stdout' as an argument and in which case the BUG filehandle is written to standard out.

-INPUT_RECORD_SEPARATOR

Defaults to the newline character.

-ORIGINAL_SCALING

-LOGFILE

-MEGACMD_BASENAME

-USERHOME

-OSNAME

-PAGER

TKG2 CONFIGURATION HASH

-DEFAULT_BOXPLOT_MOMENT_CALC_METHOD

-MONITORSIZE

-SHOWME

-PROB_BASE_MAJOR_TICKS

-PROB_BASE_MAJOR_LABEL

-OWNER

William H. Asquith.

-FILEFORMAT

-DELIMITERS

This is an array reference of the built-in file delimiters such as space, tab, comma. The delimiters are set by the distribution tkg2rc file, but the user could choose their own.

-BUILDDATE

The date of an install at root level or other date set by the author as needed.

-VERBOSE

-INCREMENT_DIALOG_FONTS

A font size increment in integer points of the dialog boxes. Positive numbers will yield larger dialog font sizes and negative numbers will yield smaller sizes.

-DIALOG_FONTS

-large

-medium

-small

-mediumB

-largeB

-smallB

-WM_OVERRIDE_POD_GEOMETRY

-PLOTTING_POSITION_COEFFICIENT

Tkg2 uses a global variable for the plotting position coefficient, which is used to generate X-Probability and Y-Probability plots on the fly. The Weibull plotting coefficient is the default. This is changeable while Tkg2 is running, but any data already loaded into the plot will not see a change. It is arguably not a good design to have this as a global variable. However, the author was not comfortable placing this with the -file hash. The -file hash controls how data is read in.

-FORCE_PAGE_HEIGHT

-RC_SCALING

-REDRAWDATA

-QUEQUE_OPTIONS

See the discussion related to the queue_options token in the Tkg2rc documentation.

-COLORS

This is an array reference of 'valid' color names, black, white, reb, ... Valid is an interesting issue as the system hosting Tkg2 could have different color names than the client and cause problems. The —checkcolors command line options should identify any problems.

-LOG_BASE_MINOR_TICKS

-DEFAULT_BOXPLOT_TRANSFORMATION_METHOD

-FORECOLOR

-LINETHICKNESS

This is an array reference of suitable linethicknesses qw(0.05i 0.15i etc) in inches. Testing has indicated that the postscript driver in tk can only resolve certain differences between lines. Only those thicknesses that show up as different on several laser printers have been selected.

-LOG_BASE_MAJOR_TICKS

This is an array reference of the default 'major' cycles to draw a major tick at. For example, '2' would produce ticks at 0.02, .2, 2, 20, . . .

-BACKCOLOR

-LOG_BASE_MAJOR_LABEL

Similar to -LOG_BASE_MAJOR_TICKS, this is an array reference of the default 'major' cycles to label the axis at. For example, '2' would produce labels centered at 0.02, .2, 2, 20, . . .

-FORCE_PAGE_WIDTH

-GEOMETRY

The —geometry command line option can place tkg2 windows at a various points on the screen and dimensions. See the commmand line option help for more details.

-ZOOM

-DELETE_LOADED_DATA

-SPLASH

Show the splash, the series of '=' signs, to the terminal during startup.

-EXTENDED_WARNINGS

-DEBUG

-PROB_BASE_MINOR_TICKS

-FONTS

This is an array reference of font family names that seem to be universally available. The author see little justification for more than just a few, but suggestions for family additions are always welcome.

-PRINTERS

-DATA_DUMPER_INDENT

-VERSION

The official version number of Tkg2. This number should match the versioning shown on the RPM package too.

FURTHER DISCUSSION

THE TKG2 INSTRUCTIONS LANGUAGE

This POD file provides detailed description of the external language to control Tkg2 objects. The Instructions are fed into Tkg2 using the —inst command line option that either reads an instruction file or reads from standard input. See also —inst and —instv within the command line help. This

documentation file is located at **Tkg2/Help/Instructions.pod**.

INTRODUCTION

For lack of better term, **Instructions** is a powerful external scripting language—better yet a configuration language—that provides runtime access to all Tkg2 objects. Each Tkg2 object is just a large complex data structure of hashes, lists, and scalars. These hashes contain all the relevant information to draw plots, annotation, and load data etc.

With the Instructions the user can access every element in the hashes, but for some of the elements it is very difficult to make meaningful access and field element modifications as Tkg2 really prefers to edit certain fields on its own to keep things from crashing. For the time being it is possible to clobber a Tkg2 file with the Instructions to a state that either Tkg2 crashes entirely or Tkg2 is not able to render your graphics. Because the Instructions are applied to the Tkg2 file contents after they have been read into memory, there is a nature safety factor in the Instructions language.

Instructions are provided by instruction files with the suggested, but not mandatory `.inst` extension. The Instructions are applied at the last moment before plots and annotation are rendered on the canvas for the first time. The `–redoinst` command line option provides a means to re-apply Instructions to already opened Tkg2 files. The redo of the Instructions can be really handy at times.

The internal code that applies the Instructions do not currently and are unlikely to ever have the field checking that the dialog boxes have. Thus, the instruction user should be careful otherwise figures will potentially be unrenderable. Fortunately because there is no permanent adjustment to Tkg2 file unless the template is saved after the Instructions are applied, there is little immediate danger that bad Instructions would mess things up.

BASIC INSTRUCTION FORMAT

The instruction format is simple and easily parsed by Tkg2. The basic format looks like this: `Object / Name of object given by user :` The `/` is used as a field separator and the `:` indicates the end of the object identification component. Spaces can be liberally used for readability except in the object name field. Tkg2 considers spaces when trying to match on that field. After the `:` comes a suite of progressively deeper hash keys that are use to identify which parameter of the object is to be changed. The key list is terminated by a double equal sign string, `==`. The value(s) for the parameter are listed next. Here is how an instruction for a plot named MY DATA would change the x axis type to logarithmic and set the y axis minimum and maximums (assuming that the auto limit configuration is turned off).

```
Plot2D / MY DATA :
 x type == log
 y min  == 0.034
 y max  == 134
```

As mentioned earlier, the instruction format is very space insensitive. In the example above, multiple adjustments to the x and y axis hashes are made to the MY DATA plot. The Instructions remember the object type and name until a line containing an other object name is encountered. Thus, the following is equivalent to the preceding example.

```
Plot2D / MY DATA :x type == log
Plot2D / MY DATA :  y min  ==0.034

Plot2D/MY DATA :y max  == 134
```

Comment lines can be inserted into the instruction file by placing # or ! at the beginning—that is the first character—of a line. Empty lines as in the above example are ignored.

The instruction file can be prematurely terminated by line containing the __END__ token starting as the first character in the file.

```
Plot2D / MY DATA :
  x type == log
  y min  == 0.034
__END__
 y max  == 134
```

All lines following an __END__ are ignored.

Some values in the object hashes are arrays or lists and not simple scalars, such as those in the examples so far. Arrays are provided as a space delimited list and blank arrays are set with double brackets []. The brackets can have spaces between them.

```
Plot2D / MY DATA:
   x major == 0.45 12
   y major == [ ]
```

OBJECT TYPES

The following is an exhaustive list of the eight object type (class) labels and their hierarchy. The `**` is where the users would insert their names for the preceding object type.

```
Plot2D / ** /:

Plot2D / ** / DataSet / ** /:

Plot2D / ** / DataSet / ** / Data / ** /:

Plot2D / ** / RefLine / ** /:

Plot2D / ** / QQLine:

AnnoLine / ** /:

AnnoSymbol / ** /:

AnnoText / ** /:
```

OBJECT NAMES (-username)

Each object is classified as one of eight types, which were presented in the previous section. Since any Tkg2 file can have one or more of the object types a mechanism is needed in which an object of a given type can be distinguished from others of the same type. The mechanism is known as the user name for the object. The user gives the object a persistent string that names or otherwise identifies the object. Internally, the object name is contained in the `-username` hash field.

The objects are named through their respective editors, except for the DataSet and Data objects. Plot2D objects are named using the **Plot Name** entry at the bottom of the Page tab. Annotation objects are named with the **User Name** entry field at the bottom using their respective annotation editors. Annotation editors are typically launched by double clicking on the annotation with the third or right mouse button. Reference lines are named using the Reference line editors accessible through the Data menu. Quantile-Quantile lines do not require a user name since there are exactly two or four quantile-quantile lines per plot. The syntax in the previous section for the QQLine thus is missing a trailing name field.

INSTRUCTIONS TUTORIAL

This section presents a brief tutorial to assist the first time instruction user. First we create two plots on the canvas. The first plot is has left, right, top, and bottom margins of 1.5, 2, 1, and 6 inches, respectively. The second plot has left, right, top, and bottom margins of 1.5, 3, 5.5, 2 inches, respectively. The creation of these plots is readily accomplished and most often is using the graphical interface, but in the interests of this tutorial, the plots can be created directly from the command line. The page is 8.5x11 inches.

```
% tkg2 --mktemp=8.5x11 --mkplot=1.5x2x1x6 --mkplot=1.5x3x5.5x2
```

After the command finishes running, the user will be presented with two nearly identical **Add Data File to Plot: plot dimensions** dialog boxes. These dialog boxes are requesting data files to load into each of the plots. Please create the following data files that are one or more space (\s+) delimited.

test1.out

```
X_Data  Y1_Data  Y2_Data
56       34       35
24       32       28
16       24       27
```

test2.out

```
Date_Time  Streamflow
10/04/2000   15000
10/05/2000   16000
10/30/2000   3500
```

Each of the data files is 4 lines long, with the first line being a 'label' line in Tkg2 parlance. The default settings of the dialog box are to read a one or more space delimited file with one label line into a scatter plot on the first y axis.

In the **Add Data File to Plot: 1.5x3x5.5x2** dialog box, change the plot type to 'X-Y Line' and hit ok. The **Open a Data File** dialog box is then launched. Selected the data file *test1.out* and hit ok. The **Load Data into first y-axis X-Y Line Plot** dialog box is shown. Select `X_Data:number` in the left list box and hit the arrow to the Abscissa entry. This will use the X_Data field for the X axis. Next select the `Y1_Data:number` entry and hit the lower arrow to make this the first column of data for the Y axis. Do the same for the `Y2_Data:number` entry. Tkg2 does not requires that each and every column in a data file be loaded into a plot, unlike some other charting software. For this tutorial, we will loaded all the data never the less. Hit the ok button. Finally, this data will be loaded and plotted as a line plot on the bottom (1.5x3x5.5x2) plot.

Repeat the previous suite of data loading steps for the remaining **Add Data File to Plot: 1.5x2x1x6** dialog box. This time note that the first column 'Date_Time' was recognized as date or time field, and Tkg2 appends the `:time` in the left list box on the **Load Data into first y-axis Scatter Plot** dialog box. Load the `Date_Time:time` into the Abscissa and load the `Streamflow:number` into the Ordinate Variables. Hit the ok button. If all went well, then the top plot on the page should have three circles plotted on it as a Scatter plot was just created.

Naming the Plots

The next step toward applying Instructions is to name the plot objects. This is most easily done by double clicking with the first (left) mouse button on each of the plots and launching the Plot Editor (see PlotEditor.pod). Launch the Plot Editor for the top plot and name this plot TIME PLOT in the **Plot Name** entry field near the bottom of the **Plot** tab. Hit the ok button. The user will see no change when the canvas is updated. Your author prefers that object names be given in all capital letters for visual clarity when reading and writing instruction files, but capital letters are not required.

Name the bottom plot ANALYSIS in the same fashion by double clicking on the bottom plot and setting the **Plot Name** entry to ANALYSIS. Instead on hitting the ok button and moving on, lets name the data objects for this plot.

Naming the Data Objects

Naming the data objects is slightly more complicated as there are two levels of nesting of data objects. With the Plot Editor still open, select the **Explanation** tab. Next hit the **Show/Hide Explanation Entries** button. The **Hide/Show Entries** dialog box is launched. It will show three checkbuttons. The checkbuttons control whether the data from a particular file is to be hidden from the explanation or whether individual entries in the explanation will be hidden. The checkbuttons are not really relevant to our instruction tutorial, but we will be using the Instructions

to toggle them.

The first checkbutton **test1.out** is the dataset object, *DataSet / ** /* corresponding to the `test1.out` data file (see Object Types section). The remaining checkbuttons are for the individual entries in the explanation and are offset to the right to show that they are contained in the `test1.out` file. These are *DataSet / ** / Data / ** /* objects (see Object Types section).

The **NameIt** buttons on the right side of the dialog box, provide access to the object naming entry fields. First, let us name the DataSet object. A dataset object is a container for all the data from a single reading of a file. If a file is read a second time (or more), a new dataset object will be created. Since we have read just one file, `test1.out`, into the ANALYSIS plot, there is only one dataset shown in the dialog box. It is left justified in the window and has a raised **NameIt** button on the right side. Hit the raised **NameIt** button associated with the test1.out line, and name the DataSet object, `DATA FROM TEST1.OUT`. Next name the `Y1_Data:number_from...` data object, `Y1 DATA` and name the `Y2_Data:number_from...` data object, `Y2 DATA`. Yes it is a slight hassle to name the objects similar to what they might be commonly referenced as, but more flexibility is garnered if the user can name objects any thing they so desire. Hit the Exit button and they exit the Plot Editor by the **Cancel** button.

Finally, save the file as `test.tkg2` using **File / SaveAs** and exit Tkg2. We are now ready to write our first official Tkg2 instruction file. However, let us look at the instruction command line first and the Instructions file itself second.

INSTRUCTIONS COMMAND LINE

The relevant command line options for the Instructions are `–inst` and `–instv`. These are equivalent commands except that the addition of the `v` turns on `–verbose` so that the user can see a verbose application of the Instructions on the Tkg2 objects. Because the effects of the Instructions on the Tkg2 files can be small or perhaps not even seen by the user, it can be difficult to see problems in the parsing and applications of individual lines of Instructions. The `–debug` command line option can toggle extremely detailed tracing of the instruction parsing. Assume that we have a Tkg2 file titled `junk.tkg2` and Instructions in the file `junk.inst`, here are some examples of verbose and debug operation.

Use the verbose reporting mechanism.

```
% tkg2 --instv=junk.inst junk.tkg2
```

Use even more verbose reporting with a verbose dump to standard output.

```
% tkg2 --instv=junk.inst --debug=stdout junk.tkg2
```

Read in the Instructions using —inst=- by standard input.

```
% cat junk.inst | tkg2 --inst=- junk.tkg2
```

WRITING THE INSTRUCTIONS FILE

To avoid any possible confusion, the reader is reminded that any configuration performed by the Instructions can be done using the graphical interface. Again, the Instructions are a means in switch Tkg2 files can be glued into other applications. Most users will not require the Instructions when they use Tkg2.

It is assumed for this section that the following files have been created from the previous sections: `test1.out`, `test2.out`, and `test.tkg2`. These will be needed to test or see the effects of the Instructions developed here. Using your favorite text editor, create a file called `test.inst` in the same directory as the three above mentioned files. Next add a header or comments to the beginning of the file that looks something like this:

```
# Instruction file for test.tkg2
# test.tkg2 reads data from test1.out and test2.out
# test.tkg2 has two plots named TIME PLOT and ANALYSIS
# The ANALYSIS plot is at the bottom of the page and has
# one DataSet object named DATA FROM TEST1.OUT and two
# data objects in that set named Y1 DATA and Y2 DATA.
# The data objects for TIME PLOT are unnamed.
```

Let us first use the Instructions to manipulate the TIME PLOT. Suppose we wanted to change the y axis type from linear to log.

Begin section on configuring TIME PLOT

```
Plot2D / TIME PLOT :
   y type == log
```

Save the `test.inst` instruction file and lets test it out by invoking Tkg2.

```
% tkg2 --instv=test.inst test.tkg2
```

The following output will be seen in the terminal.

```
Tkg2 processing test.tkg2
   DataOnTheFly Plot name 'TIME PLOT':
      Loading 'test2.out' 4 lines
   DataOnTheFly Plot name 'ANALYSIS':
      Loading 'test1.out' 4 lines
      Reading Instructions from file 'test.inst'
       --inst: on plot 'TIME PLOT'
       --inst: TIME PLOT => -y -type = log
     Inst Message: calling routeAutoLimits because one or more
                   axis types were changed for plot 'TIME PLOT'.
       --inst: on plot 'ANALYSIS'
```

For our example, the line

```
--inst: TIME PLOT => -y -type = log
```

is the most important as it shows that the instruction was parsed and then applied to the TIME PLOT. The previous line –`inst: on plot 'TIME PLOT'` shows that the instruction algorithm identified a plot named TIME PLOT in the `test.tkg2` file. The message line `Inst Message:` line shows

Lets continue to make our instruction file more complex. Let say we want to (1) reverse the y axis, (2) label the y axis on the right as well as the default left, (3) add commas to the y axis numbers, and (4) change the number font size to 12 points. Add the following lines to `test.inst`.

```
   y reverse == 1
   y doublelabel == 1
   y numcommify == 1
   y numfont size == 20
```

Try loading `test.tkg2` again with the above y axis configuration settings. The TIME PLOT should have the y axis reversed, y axis number on the left and right, comma inserted into the numbers, and the number font size increased to 20 points.

The instruction parser is smart enough to apply the values on the right hand side of the double equal sign (==) in either scalar or list context. For example, there is a key for each axis hash called

`-basemajortolabel`, which is used to control the density of labeling on an logarithmic or probability axis. This key is an anonymous array and thus multiple simultaneous values are possible. An empty list is specified by double brackets []. Here is the usage for the second y axis:

```
y2 basemajortolabel == 1 2 3 4 5 6 7 8
```

So the y2 axis, assuming it is logarithmic, will be labeled at .., 1, 10,.. and ..,2, 20, 200,.. and so on. The `-basemajortolabel` provides tremendous flexability in the density hence resolution of the logarithmic axis to a degree beyond that provided by other software.

OBJECT PARAMETER REFERENCE

This section provides a comprehensive overview of the various parameters that can be configured using the Instructions.

Plot2D / ** /:

Not yet written sorry, you could consult any .tkg2 file and look at the various key values pairs. I will get around to this part of the documentation.

Here is a brief example showing how the margins of a plot named HYD2 might look. Two ensembles are not required, but the author believes that some important concepts of the Instructions parser are represented. The leading pound is a comment. The first ensemble changes the left margin and lower margins by 50 and 46 pixels, respectively. The second ensemble changes the upper and right margins by 3.055 inches and 0.69444 inches, respectively. Note that a point is 1/72 of an inches and since the —scaling multiplier is the same division of the pixels by 72 yields equivalent inches when the graphics are actually rendered on the screen.

```
Plot2D / HYD2 :
#yumargin == 220
#xrmargin == 50
xlmargin == 50
ylmargin == 46

Plot2D / HYD2 :
yumargin == 3.055i
xrmargin == .69444i
#xlmargin == .69444i
#ylmargin == .63888i
```

Plot2D / ** / DataSet / ** /:

Not yet written sorry, you could consult any .tkg2 file and look at the various key values pairs. I will get around to this part of the documentation.

Plot2D / ** / DataSet / ** / Data / ** /:

Not yet written sorry, you could consult any .tkg2 file and look at the various key values pairs. I will get around to this part of the documentation.

Plot2D / ** / RefLine / ** /:

Not yet written sorry, you could consult any .tkg2 file and look at the various key values pairs. I will get around to this part of the documentation.

Plot2D / ** / QQLine:

Not yet written sorry, you could consult any .tkg2 file and look at the various key values pairs. I will get around to this part of the documentation.

AnnoLine / ** /:

Not yet written sorry, you could consult any .tkg2 file and look at the various key values pairs. I will get around to this part of the documentation.

AnnoSymbol / ** /:

Symbol annotation has the following keys and values that are built during object construction. The range of possible values or suggested values are shown to the right of the double equal sign.

```
-xorigin == 72
-yorigin == 3i
```

Sets the coordinate of the annotation in canvas units, which are basically pixels. If the coordinate ends in 'i' for inches, then the inches are internally converted to canvas coordinates. Tkg2 stores the coordinates in canvas units.

```
-doit          == 1 | 0, true or false
```

Toggles the drawing of the annotation object on and off. It does not delete or remove the object from storage in the Tkg2 file.

```
-username      == IMPORTANT POINT LOCATION ON PLOT
```

The name of the symbol annotation object given by the user. See the detailed discussion of object naming in `Object Names`.

```
-symbol        == Circle | Square | Triangle | Cross |
                  Star   | Horz Bar | Vert Bar

-outlinewidth == 0.01i
```

Sets the outline or border width of the symbol. Width is in inches and must end in 'i'.

```
-outlinecolor == black,

-fillcolor     == white
```

Sets the fill color of the inside of the symbol, if applicable. Colors can be and usually are expressed by names found in the `rgb.txt` file of the X server. Alternatively, hex representation of a color, even those not in `rgb.txt`, is possible.

```
-fillstyle     == experimental
```

Sets the filling style of the color in the inside of the symbol, if applicable.

```
-dashstyle     == experimental
```

Sets the dash style of the border. The dash style has no effect if the outline color is none.

```
-size          == 10
```

Sets the size of the symbol in either canvas units, which are basically pixels. If the size ends in 'i' for inches, then the inches are internally converted to canvas units. Tkg2 stores the size in canvas units.

```
-angle         == 0
```

Sets the angle of the symbol in degrees clockwise. Angle has no effect on circle, cross, star, or either of the bar symbols.

AnnoText / ** /:

Text annotation has the following keys and values that are built during object construction. The range of possible values or suggested values are shown to the right of the double equal sign.

```
-xorigin       == 72
-yorigin       == 3i
```

Sets the coordinate of the annotation in canvas units, which are basically pixels. If the coordinate ends in ‘i’ for inches, then the inches are internally converted to canvas coordinates. Tkg2 stores the coordinates in canvas units.

```
-doit          == 1 | 0, true or false
```

Toggles the drawing of the annotation object on and off. It does not delete or remove the object from storage in the Tkg2 file.

```
-username      == STATION NUMBER
```

The name of the text annotation object given by the user. See the detailed discussion of object naming in `Object Names`.

```
-text          == String \n to show.
```

The string of text to display. As with most other text strings draw on the canvas, multiple lines are possible. The Instructions pick up on new lines by internally performing a substitution of the characters ‘\n’ with the "\n" new line character. See the above example. Leading and trailing white space is stripped.

```
-justify       == left | right | center
```

Controls the justification of multiple lines of text. Justification has little to no effect when the text is a single line.

```
-anchor        == nw | n | ne | e | se | s | sw | w | center
```

Controls the location of the origin of the text relative to a hypothetical box encompassing the text. The default and recommended value is ‘nw‘, or northwest, which means that the text will start drawing from the upper left hand corner.

```
-borderwidth   == 0.025i
```

Sets the border width of a box surrounding the text. Border width is in inches and must end in ‘i’. If the outline color is none, then a border will not be seen regardless of the border width.

```
-dashstyle     == -- . --
```

Sets the dash style of the border. The dashstyle has no effect if the outline color is none. Tk uses a neat non-graphical method to denote the dashing type. Hyphens, dots, and spaces are used to set up the pattern. Please consult the `getDashList` subroutine in the `Tkg2::Base` module for more details. If in doubt, you are sure to get a solid line with ‘Solid’ or ‘solid’ and ‘—’ makes a reasonable dash.

```
-outlinecolor == black
```

Sets the outline color of the border. Colors can be and usually are expressed by names found in the `rgb.txt` file of the X server. Alternatively, hex representation of any color, even those not in `rgb.txt`, is possible.

```
-fillcolor     == blue
```

Sets the fill color of the inside of the box. Colors can be and usually are expressed by names found in the `rgb.txt` file of the X server. Alternatively, hex representation of a color, even

those not in `rgb.txt`, is possible.

```
 -fillstyle    == experimental
```

Sets the filling style of the color in the inside of the box. This is none operational as of the 0.++ series of Tkg2.

```
 -font -family    == courier, helvetica, symbol
       -size      == 10 (pts), avail. pt's depend on X server
       -weight    == normal | bold
       -slant     == roman | italic
       -color     == blue
       -rotation  == 0 -- Not used in current version of Tkg2
       -stackit   == 1 | 0, true or false
       -custom1   == not used
       -custom2   == not used
```

The font of the text annotation is controlled by the nine keys in the `-font` hash. Possible or suggested values are shown. The `-rotation` key is reserved for forward compatibility. The rotation in the Tk::Canvas is not well supported at this time.

FURTHER DISCUSSION

Developers interested in the Instructions should consult the following module:

```
 Tkg2/DeskTop/Instructions.pm
```

A note about changing axis types using the Instructions is needed. When an axis type is changed say from linear to log with the following:

```
 -x -type == log
```

Tkg2 is geared to make a call to the algorithm that automatically sets the axis limits depending upon the values of the data. This is not always desired. Thus, a very subtle trick is provided. If the axis type is requested twice, then the automatic limit algorithm is bypassed.

```
 -x -type == log
 -x -type == log
```

If the autoconfiguration algorithms are bypassed, the user is responsible for providing a minimum and maximum that are appropriate for the plot. (For example, no negative limits when a linear axis is changed to log). Tkg2 will throw a run-time error with a suitably verbose warning message if the limits are not suitable. You have been warned. The warning is not that big a deal since the Instructions themselves do not adjust the original Tkg2 file.

Unfortunately that is not the whole story as the Instructions have wide ranging effects that connect to the darkest reaches of the Tkg2 code base. The Instructions module is really just a parser and linker of the language into the Tkg2 objects.

INSTRUCTIONS SYNTAX EXAMPLES

This section provides examples of instruction files and how the syntax can be varied.

```
Plot2D/MAIN PLOT/DataSet/FILE1/Data/MEAN:
 attributes lines doit == 0
   attributes points symbol == Square
 attributes points doit == 1
```

```
__END__
Plot2D/MAIN PLOT/QQLine:
 one2one y doit == 1
Plot2D / MAIN PLOT : plottitle == Title of my\nFAVORITE PLOT
```

GENERAL UTILITIES

This POD file provides description of the external utilities that are shipped with the tkg2 distribution that are specifically written to do neat stuff, to make tkg2 more powerful, and to make your job easier. This documentation file is located at **Tkg2/Help/GeneralUtilities.pod**.

There is a very important caveat that needs description. The tkpod utility for viewing POD files does not display the underscore '_' in first or second headings or when the next is bold face. Some of the utilities described below have underscores in their name. The usage examples shown below should correct any confusion.

Most of the utilities described in the following sections have soft links in /usr/local/bin pointing back to the Tkg2/Util directory structure. The motivation of course is that /usr/local/bin is likely to be on the user's `$PATH` variable. The following soft links are created by the Tkg2 RPM distribution in /usr/local/bin.

```
checktkg2time.pl
daysbetween.pl
dtgaprdb.pl
rdb_dt2d_t.pl
rdb_ymd2d_t.pl
rdbtc.pl
strip_nonzero_from_rdb.pl
sumtkg2log.pl
```

and the following soft links are created to add in rapid creation of multiple plots per template.

```
tkg2p2.pl
tkg2p3.pl
tkg2p4.pl
tkg2pd2.pl
tkg2pd3.pl
tkg2pd4.pl
```

Your system administrator will probably not have /usr/local/Tkg2/Util set along your PATH. You can do this yourself on a per terminal basis or you could modify one of your rc files in your home directory. The `.bashrc` is used by the Bash shell.

For Bash shell users, to view and then to set the path if needed:

```
echo $PATH
export PATH="$PATH:/usr/local/Tkg2/Util"
```

For C-shell users, to view and then to set the path if needed:

```
echo $PATH
setenv PATH "$PATH:/usr/local/Tkg2/Util"
```

In the `.bashrc` file, you can insert the above export command after the system profile rc files are called. This usually means that you need to place the export near the end of the file. Contact your system administrator for assistance.

UTILITIES IN Tkg2/Util

checktkg2time.pl

This script can be used to check whether or not a specific date-time format can be properly parsed by Tkg2. Tkg2 can handle a huge variety of date-time formats because it uses the `ParseDateString` subroutine of the Date::Manip module.

```
checktkg2time.pl 10.04.1969
```

You should see that the test date '10.04.1969' parsed to '1969100400:00:00'.

daysbetween.pl

This utility is a quick to compute the floating point offset of the number of days between two dates provided on the command line. The -help command line option provides further details.

The offset can be useful in determining the date-time offset for Tkg2 or other software. For example, it is often convenient to plot two hydrographs for different locations on the same river in an over lapping fashion. The `daysbetween.pl` program could compute the offset between the flood peak times and provide a better offset than just guessing.

```
daysbetween.pl date1 date2
```

dtgaprdb.pl

A utility to insert fake missing records based on defined jumps of time for RDB files. `dtgaprdb.pl` takes an RDB file and if and only if there is a column that case insensitively matches DATE or TIME, scans the file and inserts fake missing values and a fake time stamp if the interval between two consecutive date-time values is larger than a user defined value. This is a handy utility because Tkg2 does not know how to read file headers to determine whether or not a file has a constant time step. The fake missing values line cause Tkg2 to 'lift the pen' when drawing line points.

```
dtgaprdb.pl --help
```

ldat

The script provides a convenient means to list *.dat and *.txt files for editing by nedit (default) or the editor of your choice. The files are found by recursive finding of all *.dat or *.txt files from the directory of execution downward. The *.dat files are listed by default or the `-d` command line option and *.txt files are listed by the `-t` command line option. The editor is as a command line argument.

```
ldat -t xemacs
```

Launches a list box showing *.txt files and xemacs is used to edit them.

```
ldat
```

Launches a list box showing *.dat files and nedit is ued to edit them. The ldat utility has very little to do with Tkg2, but is a handy tool never the less and I will know that it will be available on platforms with Tkg2 installed.

leapyears.pl

This script lists each year between 1900 and 2050 and reports whether or not the year is a leap year. You can really get clever with a quick grep. You will have to call this program explicitly using the full path.

```
/usr/local/Tkg2/Util/leapyears.pl | grep 1969
1969 no
```

lineindex.pl

This script inserts the index or line count of each 'data' record on the left-hand side of the data file. The word INDEX is used for each label line in the file. Lines beginning with # are not modified. The file delimiter is user specified using a prompt along standard error. The number of label lines is also user specified. A null label line count from the user is treated as 1.

```
lineindex.pl  data_file.dat  > data_file_with_index.pl
```

This program is handy if you need to fake a time series plot an you only have the data columns.

rdbtc.pl

This script takes an RDB file piped along standard input and converts any and all date columns (designated by a 'd' or 'D' in the format line) to the Tkg2 internal date-time representation along standard output. The Tkg2 internal date-time representation is fractional days since 19000101. Considerable overhead is required inside Tkg2 to parse date-time strings, the `rdbtc.pl` script provides you a way to bypass the parsing if you know what the format of the date-time strings in your data files are. A quasi-compliant RDB file is created but is certainly readable by Tkg2. The script requires Tkg2/Util/RDBtools.pm.

```
rdbtc.pl -h
```

Launches the built in help.

```
cat my.rdb | rdbtc.pl 'MM/DD/YYYY@HH:MM:SS' \
    > my_converted.rdb
```

rdb_dt2d_t.pl

A number of RDB files have the DATE and the TIME component in separate columns. This is unacceptable to the Tkg2 data model that was developed early on. Instead of writing or developing a cumbersome interface within Tkg2 to handle this, the `rdb_dt2d_t.pl` script was developed for preprocessing. `rdb_dt2d_t.pl` takes an RDB file and if and only if there is a DATE field and a TIME field these columns are replaced with a single DATE_TIME field. The script requires Tkg2/Util/RDBtools.pm.

```
rdb_dt2d_t.pl -h
```

Launches the built in help.

```
rdb_dt2d_t.pl orig.rdb > my_converted.rdb

cat my.rdb | rdb_dt2d_t.pl \
    > my_converted.rdb
```

rdb_ymd2d_t.pl

The script is very similar to the `rdb_dt2d_t.pl` described above. A number of RDB files have the YEAR, MONTH, DAY, and option TIME fields in separate columns. Again, this is at odds with the Tkg2 data model. `rdb_ymd2d_t.pl` takes an RDB file and if and only if there is are YEAR, MONTH, DAY fields and an optional TIME field these columns are replaced with a DATE_TIME field. The script requires Tkg2/Util/RDBtools.pm.

```
rdb_ymd2d_t.pl -h
```

Launches the built in help.

```
rdb_ymd2d_t.pl orig.rdb > my_converted.rdb

cat my.rdb | rdb_ymd2d_t.pl \
```

```
> my_converted.rdb
```

RDBtools.pm
: A module with many RDB specific methods for development of RDB utilities for Tkg2.

strip_nonzero_from_rdb.pl
: Occassionally, one encounters very large RDB files with many zero values. It can be desirable to remove the zero values before plotting or otherwise processing the data. This program reads a RDB file or reads standard input and prints to standard output only the records with a non-zero value in the VALUE field. The field identifier VALUE is hard wired and can only be changed by modifying the program itself.

```
strip_nonzero_from_rdb.pl original.rdb > my_converted.rdb

cat my.rdb | strip_nonzero_from_rdb.pl > my_converted.rdb
```

sumtkg2log.pl
: The script summarizes the **/tmp/tkg2.log** file. The number of users is determined as well as the number of executions of Tkg2 for each. Tab delimited output suitable for time series plotting is produced showing the number of executions per day. Summary statistics are also reported. The output is quasi-RDB compliant.

```
sumtkg2log.pl > grabbed_output.rdb
```

tkg2lines.pl
: The script can be used to see what line thickness resolution is possible and that the postscript options is working on default printer. The script was developed during the early stages of Tkg2 development for testing what line thicknesses where distinguishable when the postscript was rendered.

tkmiffix.pl
: A postprocessing script that enhances the Framemaker Interchange Format. By default this script is executed on all MIF format exports from Tkg2, but it can be toggled off.

tkpsfix.pl
: A postprocessing script that fixes ‘bugs’ in the Postscript output from the Tk tool kit. It is unknown whether there really are bugs in the Postscript, but this script makes some adjustments to the `%PageSize` region of the Postscript file to make printers work better. The original portions of the file are written to a **.tkg2_tkpsfix_debug** file in the user‘s home directory if and only if this file exists. By default this script is executed on all Postscript format exports from Tkg2, but it can be toggled off.

ps2png.pl
: A Postscript to PNG format conversion utilities based on Ghostscript and other dependencies described in the script.

xvfb_driver.pl
: The Tk toolkit requires a connection to an X server in order to work. This implies that a DISPLAY variable must be set. This is a major restriction if one desires to run Tkg2 in a `cron` job fashion. The Xvfb (X virtual frame buffer) program is a possible option. Your author does not have much experience with this like others have, but the `xvfb_driver.pl` script should provide a starting point in using Xvfb.

UTILITIES in Tkg2/Util/PreMades

The utilities here are extremely short scripts that build up a Tkg2 command line to create portrait or landscape sheets with one or more plots on them. There are no command line options for these utilities. You can run the usually Tkg2 command line options through these. These utilities require that your system supports the `exec` command in Perl—all Unix-like do.

The ‘p’ in the script name denotes a portrait (8.5x11 inch) page; whereas the ‘l’ denotes a landscape (11x8.5 inch) page. The pagesize is set by the —mktemp Tkg2 command line options. The ‘d’ in the

name reflects a 0.5 inch vertical spacing between the plots. If the 'd' is is not present, then there is a 0.25 inch vertical spacing between the plots. The right and left margins of the plots are 1.5 and 1 inch, respectively. The upper margin of the upper or top plot is 1 inch and the bottom margin of the bottom or lower plot is 1.5 inches. The lower margin is bigger so that a figure title can be inserted. The 0.25 inch space plots are intended for situations in which all of the X axis are the same type and will have the same range. The axis title can labeling can readily be turned off. The plots are set by multiple —mkplot=#x#x#x# command line options.

Finally, a neat feature is that the AddDataFile dialog box is provided for each plot, and the applicable plot dimensions are shown at the top of the dialog box.

tkg2p2.pl

```
Tkg2_PreMade, Tkg2/Util/PreMades/tkg2p2.pl:
  /usr/local/bin/tkg2 -mktemp=portrait
                      -mkplot=1.5x1x1.000x5.875
                      -mkplot=1.5x1x5.375x1.50
```

tkg2p3.pl

```
Tkg2_PreMade, Tkg2/Util/PreMades/tkg2p3.pl:
  /usr/local/bin/tkg2 -mktemp=portrait
                      -mkplot=1.5x1x1.0000x7.3333
                      -mkplot=1.5x1x3.9167x4.4166
                      -mkplot=1.5x1x6.8333x1.5000
```

tkg2p4.pl

```
Tkg2_PreMade, Tkg2/Util/PreMades/tkg2p4.pl:
  /usr/local/bin/tkg2 -mktemp=portrait
                      -mkplot=1.5x1x1.0000x8.0625
                      -mkplot=1.5x1x3.1875x5.8750
                      -mkplot=1.5x1x5.3750x3.6875
                      -mkplot=1.5x1x7.5625x1.5000
```

tkg2pd2.pl

```
Tkg2_PreMade, Tkg2/Util/PreMades/tkg2pd2.pl:
  /usr/local/bin/tkg2 -mktemp=portrait
                      -mkplot=1.5x1x1.00x6
                      -mkplot=1.5x1x5.50x1.50
```

tkg2pd3.pl

```
Tkg2_PreMade, Tkg2/Util/PreMades/tkg2pd3.pl:
  /usr/local/bin/tkg2 -mktemp=portrait
                      -mkplot=1.5x1x1.00x7.50
                      -mkplot=1.5x1x4.00x4.50
                      -mkplot=1.5x1x7.00x1.50
```

tkg2pd4.pl

```
Tkg2_PreMade, Tkg2/Util/PreMades/tkg2pd4.pl:
  /usr/local/bin/tkg2 -mktemp=portrait
                      -mkplot=1.5x1x1.00x8.25
                      -mkplot=1.5x1x3.25x6.00
                      -mkplot=1.5x1x5.50x3.75
                      -mkplot=1.5x1x7.75x1.50
```

NWIS HOST UTILITIES

This POD file provides description of the external utilities that are shipped with the Tkg2 distribution that are specifically written to interface data retrievals on the USGS-NWIS host machines. This utilities are generally wrappers on top of other command line programs such as `nwts2rdb`. This

documentation file is located at **Tkg2/Help/NWISUtilities.pod**.

There is a very important caveat that needs description. The tkpod utility for viewing POD files does not display the underscore '_' in first or second headings or when the next is bold face. Some of the utilities described below have underscores in their name. The usage examples should correct any confusion.

Most of the utilities described in the following sections have soft links in /usr/local/bin pointing back to the Tkg2/Util directory structure. The motivation of course is that /usr/local/bin is likely to be on the user's $PATH variable. The following links are created by the tkg2 RPM distribution in /usr/local/bin.

```
gwsi_std2rdb.pl
outwat2rdb.pl
DVgetem.pl
DVgetpor.pl
DVlastwk.pl
UVgetem.pl
UVlastwk.pl
UVgetpor.pl
```

For details on setting your PATH to include /usr/local/Tkg2/Util contact your system administrator or see introductory discussion in the GeneralUtilties help file.

INTRODUCTION

Tkg2 can not be built with an interface that satisfies both independent users and users needing a graphical plotting engine on a large corporate data base. These users commonly have different requirements. For example, an independent user might need Tkg2 to read a variety of loosely structured file formats; whereas the data base user might require rigidly structured formats. It also is difficult to built interfaces in Tkg2 itself that talk directly to a particular data base because the prime directive of Tkg2 is to by a flexible stand-alone interactive graphics package. In this light, several utilities are distributed with Tkg2 to make interaction with the USGS-NWIS data base easier without making Tkg2 itself an extension of NWIS.

THE PROGRAMS

dtgaprdb.pl

A utility to insert fake missing records based on defined jumps of time for RDB files. See documentation in GeneralUtilities.

gwsi_std2rdb.pl

The script converts GWSI Standard Table format files to a quasi-RDB compliant format with converted date field for easier parsing by Tkg2. The input file is the first argument on the command line or standard input is used. The output file is the second argument on the command line or standard input is used. If the output file name is '-', then '.rdb' is added to the input file name to create the output file.

```
cat input.gwsi | gwsi_std2rdb.pl > out.rdb
gwsi_std2rdb.pl input.gwsi -
   # output written to input.rdb
gwsi_std2rdb.pl input.gwsi gwsi_as.rdb
```

outwat2rdb.pl

This script can be used to convert an OUTWAT fixed format ASCII (text) file to a quasi-compliant, but Tkg2 readable RDB file (a type of tab delimited file). The biggest problem with OUTWAT files is that the year, month, and day of a data values are in separate columns. Because Tkg2 is very column oriented in its file parsing (so are many spread sheet like applications), it is impossible for Tkg2 to see the three columns as a date.

Usage: outwat2rdb.pl outwat.in rdb.out
Usage: outwat2rdb.pl
To view the help page.

DVgetem.pl (mature, but still experimental)

A utility to assist in retrieving daily values starting from right now backwards in time a specified number of days using the `nwts2rdb` program that resides on NWIS servers. This program is really just a wrapper on top of the `nwts2rdb` program, but has some special shortcuts and tweaks more inline with expected Tkg2 usage. The program is readily used to build Tkg2 templates by using the megacommand functionality. This allows one to quickly retrieve one more more stations and various data descriptors for plotting. Much more efficient than using the common web interfaces because you can have more than one station or data descriptor per plot. Build a Tkg2 template with built-in data retrieval and set back and drink your coffee in the morning. Consult the help page for more details and see the description for `DVlastwk.pl` below. You should also consult the manpage for the `nwts2rdb` command if you are not familiar with it; however, knowledge of the `nwts2rdb` command is not required to use `DVgetem.pl`.

```
DVgetem.pl -h

DVgetem.pl -d=1 -s=00003 -b=45 08167000
```

The last command retrieves an RDB file on station 08167000 for the last 45 days (`-b=45`, backwards 45 days) using statistic code 00003 (daily mean, `-s=00003`) and data descriptor 1 (`-d=1`).

A note about `nwts2rdb` is needed. Apparently, an undocumented or underdocumented feature is a period of record pull with `-b` and `-e` values that look something like the following. Zero for beginning and eight ‘9’ digits for the ending will pull the period of record in whole multiples of water year.

```
nwts2rdb -tdv -aUSGS -n08167000 -d1 -s00003 -b0 -e99999999
```

Here is what the `nwts2rdb` author Scott Bartholoma `sbarthol@usgs.gov` says about this functionality.
For Daily values *DV*, the `-b` and `-e` arguments are dates, not datetimes, so only eight characters are used. Anything from `-b0` to `-b00000000` should act the same. The other data types only output data that was found in the database, so the all zeros and all nines worked well. For DV, I coded it to provide "complete" data with missing values included where needed to fill out the date range. This caused an unanticipated problem when zeros and nines were used to get period of record. It tried to write a complete file of daily values from year 0000 through year 9999—over 3.6 million rows. So, the behavior of the DV module was changed to "see" `-b00000000` to mean beginning of period and `-e99999999` to mean end of period, so that they would work the same for DV as for the other data types.

DVgetpor.pl (mature, but still experimental)

Another daily value retrieval utility. In the spirit of the `DVgetem.pl` program, `DVgetpor.pl` can be used to make period of record retrievals for a station.

```
DVgetpor.pl -h

DVgetpor.pl -d1 08167000
```

See the closing notes on `DVgetem.pl` for discussion on how `DVgetpor.pl` wraps `nwts2rdb` and uses `-b0` and `-e99999999` (8 nines) to make the retrieval. Note that `nwts2rdb` pulls on a water year by water year basis in with these options. Hence, the first water year is likely to be incomplete because the station probably did not go active on October 1. The last water year is likely to be incomplete too because today—literally the time the program is run—is not September 30.

DVlastwk.pl (mature, but still experimental)

A utility to assist in retrieving daily values for the last week using the `nwts2rdb` program that resides on NWIS servers. This program is really just a wrapper on top of the `nwts2rdb` program, but has some special shortcuts and tweaks more inline with expected Tkg2 usage. The program is also a special case of the more general `DVgetem.pl` program. Integer multiples of one week can be retrieved by command line switch. Consult the help page for more details and see the description for `DVlastwk.pl` shown above.

```
DVlastwk.pl -h

DVlastwk.pl -s=00003 -d=1 -o=2 08167000
```

The last command retrieves an rdb file on station 08167000 for the last 14 days (`-o=2`) using statistic code 00003 (daily mean, `-s=00003`) and data descriptor 1 (`-d=1`).

UVgetem.pl (mature, but still experimental)

A utility to assist in retrieving unit values starting from right now backwards in time a specified number of days using the `nwts2rdb` program that resides on NWIS servers. This command parallel is `DVgetem.pl`.

```
UVgetem.pl -h

UVgetem.pl -b=30 -d=1 -s=C 08167000
```

The last command retrieves an rdb file on station 08167000 for the last 30 days (`-b=30`, backwards 30 days) using computed (-s=C) record on data descriptor 1 (`-d=1`).

UVlastwk.pl (mature, but still experimental)

A utility to assist in retrieving unit values for the last week using the `nwts2rdb` program that resides on NWIS servers. Consult the help page for more details and see the description for `DVlastwk.pl` shown above. You should also consult the manpage for the `nwts2rdb` command if you are not familiar with it; however, knowledge of the `nwts2rdb` command is not required to use `UVlastwk.pl`.

```
UVlastwk.pl -h

UVlastwk.pl -d=1 -s=C 08167000
```

The last command retrieves an rdb file on station 08167000 for the last 7 days using computed (`-s=C`) record on data descriptor 1 (`-d=1`).

UVgetpor.pl (mature, but still experimental)

Another unit-value retrieval utility. In the spirit of the `UVgetem.pl` program, `UVgetpor.pl` can be used to make period of record retrievals for a station.

```
UVgetpor.pl -h

UVgetpor.pl -d1 08167000
```

See the closing notes on `UVgetem.pl` for discussion on how `UVgetpor.pl` wraps `nwts2rdb` and uses `-b0` and `-e99999999999999` (14 nines) to make the retrieval. Note that `nwts2rdb` pulls on a water year by water year basis in with these options. Hence, the first water year is likely to be incomplete because the station probably did not go active on October 1. The last water year is likely to be incomplete too because today—literally the time the program is run—is not September 30.

TUTORIAL

This section provides a quick tutorial or example of the `DVgetem.pl` and `UVgetem.pl` utilities. We will create a single plot on a Tkg2 template or sheet. We will use the megacommand function to call these two utilities to pull the last 30 days of daily value and unit value data for a USGS station in Texas.

1. Create a portrait template with a single large plot

```
% tkg2 -mktemp=p
```

The Add Data File to Plot: DEFAULT dialog box should now shown. This dialog box is used to read a data file or output from a command. The DEFAULT is a plot with a 1.75, 1.0, 1.0, and 2.0 inch, left, right, top, and bottom margin, respectively. This plot size can be set by the —mkplot command line option.

2. Set the Plot Type

We want to draw a line plot, so set the Plot Type to X-Y Line. The Plot Type menubutton is the second button down from the top of the Basic tab.

3. Set the File Type to RDB

We will be reading a RDB file type, so toggle the 'File is RDB' checkbutton on by clicking on the little square to the left of the label. Several of the fields will grey out and can be edited with this button is on.

4. Set the field checking

We know that the NWIS data base will provide well defined data fields. Therefore to speed up the importation of the data, toggle the 'Do not verify/test field types (fast reading)' checkbutton on.

5. Set up the unit-value retrieval

Click on the Advanced tab and find the 'megacommand' entry field. In this field you can run arbitrary programs. The output of these programs is used for the data to plot instead of a file. Temporary files are written to your directory. In the entry field type the following.

```
/usr/local/Tkg2/Util/UVgetem.pl -b=30 -d=1 -s=C 08167000
```

or

```
UVgetem.pl -b=30 -d=1 -s=C 08167000
```

if a link to /usr/local/Tkg2/Util/UVgetem.pl has been made in a directory along your `$PATH`. It likely has been. Your station number, data descriptor (`-d`) and statistic code (`-s`) will likely be different. After you have entered the command, hit OK.

6. Load Data into first y-axis X-Y Line Plot

The command from step 5 immediate runs and should rapidly return an another dialog box. This dialog is used to set which values are plotted against on another. Select the DATETIME:time entry in the left listbox and hit the arrow to the Abscissa or X axis. Select the VALUE:number entry and hit the arrow to the Ordinate Variables or Y axis and hit OK. You should now see some data plotted into your plot.

7. Plot the daily-value data

We are now going to repeat most of the above steps with some minor variation. First, select the plot with the left mouse button. Little black squares should mark the perimeter of the plot showing you that you have selected the plot. Next go to the DATA menu and select the 'Add Data File to Selected Plot' action. The now familiar Add Data to Plot dialog box is launched. The Plot Type should have remained X-Y Line and the File is RDB and field checking checkbuttons should have remained toggled on.

Click on the Advanced tab and in the megacommand entry type the following.

```
/usr/local/Tkg2/Util/DVgetem.pl -d=1 -s=00003 -b=30 08167000
```

or

```
DVgetem.pl -d=1 -s=00003 -b=30 08167000
```

The data retrieved are daily values, which do not have a time component, e.g. 10/01/1978. By default Tkg2 assumes a 00:00:00 time component if one is absent. This is the most logical from a computational standpoint, but might be at odds with interpretive or stylistic desires of the user. To mitigate, Tkg2 can be set to do a step plot across the day. However, let us assume that you do not want this, but instead want the daily values plotting at noon or 12:00:00. This can be done by setting the following into the 'Convert date-time ...' entry.

```
-:-:-:12:00:00
```

After that is done, hit OK.

The Load Data into ... Plot dialog is now displayed. Place the DATE:time in the X axis and the VALUE:number in the Y axis and hit OK. You should now see the daily values plotted. It is hard to distinguish between the two lines. Let us change the plotting style of the daily values.

8. Change daily-value plotting style
In the explanation to the right of the plot, double-left click with the mouse on the bottom line symbol. You will see the cursor change to a hand. Do not click on the text strings in the explanation as this will launch a dialog box that we do not want at this time. The Edit Data Drawing Styles dialog box should be displayed now with the Points tab raised.

Click on the DoIt checkbutton and hit Apply. White-filled circles are no added to the figure and the circles should coincide with 12:00:00 on the time axis. Also change the color of the circle edge to red and the fill to red too. You may hit Apply at any time to see your changes.

Next, click on the Lines tab so that we can edit the line drawing style. Let us change the line color to red too. Finally, hit OK.

9. Save the template
Save the template. Go to the FILE menu and select either the Save or the Save As (filename requested). Name this sheet 'sta08167000_30days.tkg2' or something similar and save it. Do not use the Save As (with imported data) as this saving function hard loads the pulled data into the plot. We want this template to go an get the data every time it is opened or 'run'. Finally, exit Tkg2.

10. Test the template
Before we spend more time configuring our plot with descriptive axis titles and moving the explanation around (right mouse click, by the way), we should test whether the template is capable of pulling data again.

```
% sta08167000_30days.tkg2 -verbose
```

The —verbose has been added to showing reporting of the retrieval calls. If you encounter problems, adding the —megacmd_show option might help by showing the raw data pulled.

```
% sta08167000_30days.tkg2 -verbose --megacmd_show
```

You can even spool this plot directly to the printer—try the following command.

```
% sta08167000_30days.tkg2 -batch -destination=laserx
```

The —batch option does the spooling and adds the —autoexit command line option for convenience. The —destination option sets the printer. If the —destination is left off, then the default printer is used.

You can now open the tkg2 file, edit it how you wish as save it. Every time the file is opened, the last 30 days of daily and unit values will be displayed. Is that cool or what? The tutorial is complete.

ADD DATA FILE TO PLOT

This file documents the dialog box for importing data from a file into a plot and documents the various subtle behind-the-scenes data file reading activities. This file is located at **Tkg2/Help/AddDataFile.pod**. Because data file reading is so important to Tkg2, extensive discussion of ancillary data issues than just the settings in the dialog box are discussed as well.

INTRODUCTION AND TKG2 DATA FILES

Tkg2 reads data from ASCII or plain text files. ASCII text is a very convenient file storage format for the type of tasks that Tkg2 provides. ASCII is cross platform, archivable, and certainly years from now your ASCII files will continue to be readable and adaptable to other software. Tkg2 views data files as a collection of delimited columns and not fixed format files per se. These files are readily produced by or imported into spreadsheets and other computer programs. The Tkg2 philosophy is to keep everthing in ASCII.

Tkg2‘s founding principle is that the graphics package should basically be an interactive graphing package and *also* a markup engine for 2-D data much like a web browser is a markup engine for HTML. Tkg2 is not intended to be an interactive data editor like a spreadsheet or a data base. Further, although there are many statistics that Tkg2 computes with its boxplot algorithms and thus Tkg2 can be used as a statistical analysis package too, Tkg2 is not intended to be an statistical package either.

In the spirit of becomming a unique 2-D plotting engine, Tkg2 has some very powerful and flexible data file parsing features as well as scriptability features. These are discussed in some detail in later parts of this section. First let us quickly focus two features and data file concepts that might be surprizing to the first time user.

Dynamic Data File Loading

One of the coolest features of Tkg2 and the feature that impressed me so much with a package that we had at the USGS in the 1990‘s was the ability to have a plotting package read in the data everytime the graphics file was opened up. This has the immediate benefit of detaching the graphic from the data—if the data file changes, which they do often, then the graphic is automatically updated. With this in mind, the default behavior for Tkg2 is to not hard load or import the data into the graphics file, but to read the files on the fly. You do not have to use this feature and during the initial file loading the hard loading can be turned on. Also the FILE menu provides a way to save a graphics file with the data hard loaded even though some or all of the data might have been read in on the fly.

Automatic Axis Configuration

Because the dynamic loading is encouraged, Tkg2 by default does dynamic configuration on axis settings such as minimum, maximum, and number of labels. This behavior can be turned off on a per axis basis and individually on the minimum and maximum of the axis. The autoconfiguration will likely be surprizing and possibly annoying to the first time user (judging from my email). However, the default autoconfiguration is considered a feature. Specifically, the new user will load their data in, reconfigure the axis, and either still have the dynamic loading working or will add more data to the plot. This cause the configuration algorithms to be executed and likely the manually choosensettings to be overrided.

Data File Concepts

Tkg2 considers four distinct portions of a data file: the header or comments, labels, optional format, and the data. The header or comments are lines of the file that are completely ignored by Tkg2. The header is the portion in the file before the label lines and comments are all other lines to skip. Tkg2 can not use the contents in a file header like other software might. By default lines beginning with a leading # are ignored, this can be overridden.

The label lines denote the titles of the data columns. More than one label can be used or the label line can left off altogether, in which case v1, v2, etc. are used to distinguish between the columns. The labels on concatenated together. The column count in each of the labels lines must match or an error is generated.

The optional format line immediately follows the label lines. If a format line is present, its column count must match the count in the label lines. By default Tkg2 does automatic type detection on the data values. Four data types are recognized. These are numeric, string, date-time, and computed date-time. Most users should only concern themselves with the first three. The computed date-time is a precomputed date-time value in the internal format used internally by Tkg2. Internally Tkg2 converts a date-time value to a floating point number of days since January 1, 1900. The computed date-time can be useful in speeding up the data reading operations.

Lastly, the data lines are present. Comments or empty lines can be interspersed witin in the data provided that the file reading settings know to skip the lines. The column count in the data lines must match the counts from the labels.

Each of the last three paragraphs mentions that the column count must remain constant through out the file. Columns are determined solely on the choice of file delimiter. Visually this can be hard to distinguish with some file types such as tab delimited files as the delimiter is hard to seen. Violation of the constant column requirement will spawn errors during the read in phase. Read these error messages carefully. Tkg2 should help you determine what parts of your file needs 'correction'. The most common gotcha occurs with space delimited files—especially hand entered ones. Trailing spaces at the end of a record but not all records will throw an exception. This is necessary so that Tkg2 can parse space delimited files in which the last column is null. If you suspect that your data file suffers from inconsistent trailing spaces, you can strip them with the following perl code that works for Unix files.

```
% perl -pi -e 's/\s+\n$/\n' your_data_file.txt
```

Some examples of 'valid' data files are needed to further the discussion. The following file has a two line header with an blank line embedded between the two lines, one label line, no format line, and three data lines. The file is also space delimited.

```
# header here

# more header
data1  temperature
 45      23
 23      65
# whoops data logger went bad
 24      34
```

The following file has no header, no label, no format line, and is comma delimited.

```
 45,23
 23,65
 24,34
```

The following file has a header, multiple label lines, a format line, and is tab delimited. The tab is denoted by the <t>.

```
# Header
DATE<t>FLOW<t>TEMP
TIME<t>CFS<t>F
d<t>n<t>n
19701001<t>45<t>23
```

```
19800906<t><t>34
```

The format line and the ‘d’ is needed in the above file because the date-time values look like integer numbers although they need to parse into a date. The ‘d’ tells Tkg2 to cast the contents of this column as a date. The column titles are DATE_TIME, FLOW_CFS, and TEMP_F. Tkg2 concatenates the underscore for you. Also note that two tabs are touching in the last line. Tkg2 will recognize the FLOW as a missing value in this line.

The __END__ token

In Perl and hence naturally in Tkg2, reading of a file ceases if an __END__ token is placed in a file. This token as two leading and two trailing underscores. This is handy for testing features of Tkg2 on large files, but you do not need the entire file read in to see that you tweaks are working properly or maybe you simply want to comment out the tail of the file.

THE DIALOG BOX

With the AddDatatoPlot dialog box the location of the axis titles, the plot type, and other plot parameters are set. The dialog box also provides a comprehensive interface into data file reading activities including such parameters as file delimiter and file length. The toggle whether to import or hard load the data or have the data dynamically loaded is provided. Furthermore, this dialog box provides a powerful interface to additional user specified data processing. This is a very important dialog box in which the regular user will see quite often. Only through this dialog box can data be loaded for the first time into a plot. The Perl module that provides the dialog box is located at Tkg2/Data/Class/AddDataToPlot.pm.

ACCESSING THE DIALOG BOX

The dialog box can be launched by either selecting a plot first by clicking Button-1 inside the plot. Eight black squares demark the corners and edges of a plot when it is selected. Next from the *DataEdit* menu select *Add Dataset to Plot*. An alternative method to launch the dialog box is from the command line when —mktemp and optionally —mkplot options are used. These options allow quick construction of a template (—mktemp) and one or more plots (—mkplot). For each plot so generated, the AddDataToPlot dialog box is provided for convenience.

BASIC tab

Position Axes

Place the axis title and labeling along the desired axis. The settings can be changed using the Continuous or Discrete Axis Editors, which are usually accessed by double-clicking mouse button-1 on the axis. Changing the axis position does not create a double-X plot nor a double-Y plot. "Double" meaning that each X axis line or Y axis line is independent. Actually, Tkg2 does not support double-X axis, but does support double-Y axis. If you want a double-Y plot, toggle the checkbutton labeled **Data for second Y axis?**.

Plot Type

Numerous plot types are currently supported. These are briefly discussed here. Just go and try them out and see whether you like them or not. Axis types such as linear, log, probability, gumbel, and time series are controlled using the axis editors.

```
Scatter
```

Requires X and Y data points, no connecting line.

```
X-Y Line
```

Requires X and Y data points with a connecting line.

```
Text (Annotation)
```

Requires X, Y, and another column in data file for text source, and plots as points, lines, and text near the points. Leader lines and other advanced options for the text are provided by the

DrawDataEditor. If the text has the following special content `pscale:###`, in which ### is a number, then that number is multiplied on the symbol size of the scatter plot. The text is literally still drawn on the plot; users will likely turn the text drawing off in the DrawDataEditor. An example is useful. The following file has a single label line and is one or more space delimited. The lines beginning with # are ignored by default. The file contents when plotted as a text plot would have three different symbols sizes drawn for the five data points. The text labels are drawn by default and can be toggled off; the changes in symbol size will still be used.

```
XVALUE   YVALUE   SIZE
# Make a symbol half the symbol size
150       610     pscale:0.5
# Make a symbol twice the symbol size
155       780     pscale:2.0000
# Make a symbol at the same size as the symbol size
160       520     pscale:1
# Make two additional symbols at the symbol size
130       390     some_other_label
153       272     yet_another_label
```

Y-Accumulation

All the Y data values are accumulated as the data is read in. The accumulation occurs on all columns. The accumulator is initially set to zero.

Y-Accumulation(text)

Just like a regular Text plot except that all the Y data values are accumulated as the data is read in like the regular Y-Accumulation plot.

X-Probability

Requires only a single data column, Tkg2 computes the plotting positions, and plots the plotting position probabilities on the X axis and the data on the Y axis. The plotting position is set globally through the Tkg2rc file or the **Global Settings** menu. An important note about missing data is needed. It is very complicated to compute plotting positions in the presence of missing data; therefore, tkg2 **throws out** missing values. This means that the original data array contain both values and missing values is truncated or trimmed to another array containing only values; the plotting positions are then computed from this new array. The user is not provided any warnings that this has occurred; however, the user can be assured that this is a necessary step. Moral—know your data sets?

Y-Probability

Opposite of X-Probability, requires only a single data column, Tkg2 computes the plotting positions, and plots the plotting position probabilities on the X axis and the data on the Y axis. See X-probability plot for discussion about missing values.

Bar

Like a Scatter plot, but draws bars from either the top, bottom, left, or right plot edges to the points. This is not a histogram plot. Tkg2 does not support histogram plots, as your author (statistician by training) does not condone the use of histograms as they easily distort the true nature of the data. Cumulative probability curves are better.

Shade

Like a Line plot, but also shades from lines to either the top, bottom, left, or right plot edges. It is not possible to shade between two ‘columns’ of data. In order to gain the capability to plot shade

between two columns, a ‘Shade Between’ plot is required (see below).

Shade Between

A Shade Between plot is quite different in external appearance and certainly internal representation than a shade plot. Because there is more going on behind the scenes, a Shade Between plot can be noticably slower in rendering than a Shade plot. A data set in which one might want between shading is:

```
DAY     MIN_TEMPERATURE    MAX_TEMPERATURE
1          34                 56
2          37                 58
3          45                 65
4          51                 72
```

The day is desired along the X axis and a shaded region between the minimum and maximum temperatures is needed. During the data loading or actually in the data loading dialog box, the DAY would be choosen for the X-axis, say MIN_TEMPERATURE would be choosen for the Y-axis and MAX_TEMPERATURE choosen for the ‘Value to shade between’ entry field. The usual point symbols and lines are drawn between the DAY and the MIN_TEMPERATURE and a shaded region extending up (uniformally in this example) to the MAX_TEMPERATURE is drawn. A line on the border of the shaded region is not drawn to provide greater flexbility. If a line between the DAY and the MAX_TEMPERATURE values was desired too, then a separate X-Y Line plot would have to be overlayed.

Shade Between Accumulation

Like a Shade Between plot, except that accumulation of both the Y values and the values for the third column or the values to shade too are accumulated as well. There may or may not be problems if your data has missing Y values, but not the shade too values. Or when the shade too values have missing values and the Y values do not.

Y-Error Bar

Like an X-Y Line plot; however, error lines can be drawn and later configured. A third column of data, which sets the error, is requested. The error is subtracted from and added to the Y data values to produce the bar. For example, a Y value equals 50 and the corresponding error is 6. Error lines from 50 to 56 and from 50 to 44 are drawn. This is a symmetrical bar. You can get an asymmetrical plot by having your errors delimited with a ‘<=>’ string as in 5<=>6. Thus, 5 units will be subtracted from the Y data point and 6 units will be added. The error line for the symmetrical and then asymmetrical example would like like the following.

```
44------50------56
 45-----50------56
```

Non-numeric errors values are treated as missing values. The missing value string that is specified for the other data columns can be used with the error lines as well. Consult the Error Line Data discussion in the Further Discussion section below for more discussion.

Y-Error Limits

Like a Y-Error Bar plot, expect that the third column of data is the terminal point of the error line. For example, a Y value equals 50 and the corresponding error limit is 56. An error line extending from 50 to 56 is drawn. If the error limit is 45, then an error line extending from 50 to 45 is drawn. Unlike the ‘Error Bar’ plots, the error line is one sided by default. Double-sided error lines can be generated using the ‘<=>’ string as a delimiter between the two limits. The error line

for the delimited example would look like the following. Consult the Error Line Data discussion in the Further Discussion section below for more discussion.

```
              50------56
        45-----50
        45-----50------56
```

`X-Y Error Bar`

Just like an Y-Error Bar plot except that error lines in both X and Y can be drawn. See discussion associated with Y-Error Bar.

`X-Y Error Limits`

Just like an Y-Error Limit plot except that error lines in both X and Y can be drawn. See discussion associated with X-Y Error Limits.

`X-Discrete`

Usually, Tkg2 only changes an axis to discrete if the data column is determined to be a string, that is, not a number or a valid time format. However, if a number is to be treated as a string, the user can either wrap quotes around each in the data file or use X-Discrete, or specify the format of the columns. X-Discrete forces the X-axis to be discrete.

`Y-Discrete`

Exactly like X-Discrete, but works on the Y-axis.

`XY-Discrete`

Exactly like X- or Y-Discrete, but both the X- and Y-axis are turned into discrete.

Data for Second-Y Axis?

This checkbutton will toggle the second Y-axis on and plot this data using it. The second Y axis become the Y axis line on the right hand side of the plot. This type of plot is also known as a double Y plot. The discussion under the **Position Axes** is also relevant. Tkg2 will not permit a double-Y plot, meaning that data will be loaded into the second Y axis, before data is loaded into the first Y axis. A warning message window is provided to prevent this. After data is loaded into the first axis, choosing between first or second Y axis is permitted with all additional data loading.

Number of Lines to Skip

How many lines at the top of the file need to be outright skipped as data is being read in. For example, the following file probably needs a number of skip lines equaling 2.

```
# EXAMPLE DATA SET
# FOR TKG2 add_data_set.pod
 X_DATA Y_DATA
 (feet) (gallons)
   |        |
   |      -9999
 # |        |
   V        V
```

Missing Value Identifier

Occasionally, a data file will have missing values in it. Whatever this value is can be controlled using this entry field. For example in the ‘Example Data Set’ file seen above, -9999 is a missing

value. Missing value identifier is thus '-9999'. In all cases actual undefined or null values in a data file are loaded in, but quietly ignored during the graphics rendering. The missing values also provide Tkg2 with a mechanism to 'lift the pen' when drawing line plots. For example, the following data set has a six year wide gapb between the years of periodic measurements.

```
DATE_TIME    DISCHARGE
10/04/1991    560
11/21/1992    670
04/23/1999    700
08/19/2000    450
```

If this data file if plotted with DATE_TIME on the X axis and DISCHARGE on the Y axis, then a straight line connecting 670 and 700 will be drawn. This is inappropriate because a jump in the interval is present. Tkg2 has no mechanism to handle fixed interval time series data so internally it has no way of determining whether it should 'lift the pen' between 1992 and 1999. You can trick Tkg2 into lifting the pen by inserting a time value between 11/21/1992 and 04/23/1999 and provide no discharge. In the example '—' is the missing value string. Each of the following are equivalent. Only a single missing value line is needed for pen lifting.

```
DATE_TIME    DISCHARGE
10/04/1991    560
11/21/1992    670
11/22/1992     --
04/23/1999    700
08/19/2000    450
```

or

```
DATE_TIME    DISCHARGE
10/04/1991    560
11/21/1992    670
01/01/1998     --
04/23/1999    700
08/19/2000    450
```

or

```
DATE_TIME    DISCHARGE
10/04/1991    560
11/21/1992    670
04/22/1999@12:13:21     --
04/23/1999    700
08/19/2000    450
```

Consult the Tkg2/Util/dtgaprdb.pl utility for assistance with missing value insertion in rdb files.

File Delimiter

Specify the file delimiter such as tab, comma, or space. Some of the delimiters are shown in regular expression context. The following table defines each. The Tkg2rc file specifies the delimiters and the default delimiter at runtime. If custom is chosen, then the 'Custom File Delimiter' field needs to be filled in.

```
\s       a single space
\s+      one or more spaces
\s\s+    two or more spaces
\t       tab
|        a pipe
```

```
,        a comma
:        a colon
custom   now use 'Custom File Delimiter'
```

Custom File Delimiter

Specify the custom file delimiter such as 'CustomBreakPoint'. If given as this '3 \n' or '6 \r\n', then a newline (\n, Unix files), or a carriage return newline (\r\n, Windows files) becomes the delimiter and 3 or 6 lines of the file are read in at a time.

Number of Label Lines

Specify the number of label or column heading lines in the file. The labels for a given column are catenated together. If the same column header is used, then Tkg2 catentates a counter on the end of the label. For example, in the 'Example Data Set' file seen above, the number of label lines is 2. Thus, the first column is labeled as X-DATA(feet).

Skip Line Identifier

Specify how optional commenting of the data is in the file. Perl regular expression syntax is used. The default of '^#', tells Perl to just ignore any lines in the data part of the file that start (^) with a 'X'. For example, in the 'Example Data Set' file seen in the 'Missing Value Identifier' section, the line after the line containing -9999 is ignored.

Because the full power of Perl's regular expressions (a language unto itself), you can have more complicated logic. Suppose you have a comma delimited file from a data logger. You have added comments in the data file concerning the data and related matters. Also these files have three distinct record identifiers.

```
# Battery voltage remained nominal
ID110,YEAR,DAY OF YEAR,TEMP
ID245,YEAR,DAY OF YEAR,HUMIDITY,WIND
ID262,YEAR,DAY OF YEAR,RAINFALL1,RAINFALL2
110,2001,345,45
245,2001,345,23,1.1234
262,2001,345,0,0
110,2001,346,56
245,2001,346,16,0.942
262,2001,346,0,0
110,2001,347,38
245,2001,347,78,3.780
262,2001,347,.56,.85
```

Clearly in this file we have three distinct record types and each has a different number of columns. Plotting this is indeed a problem. We can not directly produce a time series plot because the year and the day of the year are not in the same column nor can Tkg2's parser in its present state make the conversion for you. That being said, we can still plot any of the data values against the day of the year. Here are some examples of Skip Line Identifiers that will permit reading and plotting of data in this file.

```
^#|^\w+110|^\w+245|^110|^245
```

This regular expression is summarized as: 1) Skip the leading # lines (^#), 2) Skip the ID110 line (^\w+110) with one or more letters (^\w+) and the number 110, 3) Skip the ID245 line (^\w+245), 3) Skip the records starting with 245 (^245), and 4) Skip the records starting with 262 (^262). The pipe symbol '|' denotes logical OR and the caret '^' means 'starting with'.

With the multiple OR regular expression, a number of label lines setting of 1, and a comma as the delimiter, the above file could be read in. Variations on the regular expression of course are possible to read in just the 110 line or the 262 line.

The example should demonstrate a really powerful tool. Consult the Perl documentation itself for further discussion of regular expression syntax.

The sense of the skip line regular expression can be inverted (only for non-RDB files) by the "invert it" checkbutton to the right of the entry field. This toggle greatly simplifies the above example. To illustrate, suppose that the lines containing the 262 record id were the only ones desired. The skip line identifier could be set to `^262`; then only those lines contain 262 in the first three columns would be read in. This makes it even easier to have multiple 'data files' contained within one file. Cool or what? A suggested framework for file construction would be the following.

```
# Battery voltage remained nominal
110,YEAR,DAY OF YEAR,TEMP
245,YEAR,DAY OF YEAR,HUMIDITY,WIND
262,YEAR,DAY OF YEAR,RAINFALL1,RAINFALL2
110,2001,345,45
245,2001,345,23,1.1234
262,2001,345,0,0
110,2001,346,56
245,2001,346,16,0.942
262,2001,346,0,0
110,2001,347,38
245,2001,347,78,3.780
262,2001,347,.56,.85
```

Notice how the "ID" was removed from the record labels shown in the preceding file? Now with ^262 and inversion on, the line

```
262,YEAR,DAY OF YEAR,RAINFALL1,RAINFALL2
```

will get picked up as the label line.

Invert it

This checkbutton reverses the sense of the skip line identifier. This results in the extraction of the 262 record of the previous examples as easy as setting the skip line identifier to '^262' and toggling the Invert it on.

Column types (a|DSNT|f)

As described in the Introduction, Tkg2 identifies four data types. These are numbers, strings, date-times, and computed date-time values. The format abbreviation for each are n, s, d, and t, respectively. The abbreviation is case insensitive so N, S, D, and T are equivalent. By default the Column Types of a file are set to 'auto'. The default is abbreviated as 'a' in the entry field.
The 'auto' setting means that Tkg2 will successively test each data value as a number the as a date-time value. If it files to parse either of these the value becomes a string. Testing for the computed date-time is not possible (see discussion under the Data File Concepts section for more details about computed date-time values). As soon as a value is determined to be a string, then the whole column is identified as a string even if the other data types are present. Tkg2 then internally and externally maintains the data type distinction on each data column. This type casting is preserved for dynamic data loading. The automatic type determination can be CPU expessive. Therefore, one can specify the column types too.

Suppose you have the following file.

```
ID,DATE,RAIN1,RAIN2,COMMENTS
ID110,20010609,.25,.10,
ID110,20010610,.56,.85,last valid value
```

The ID and COMMENTS columns are strings. The two RAIN columns are numbers. These would are compatible with the auto type determination. However, since the DATE column is really just a number, you must override the auto type detection with your on types. Three suitable entries for the Column types are

```
SDNNS
sDnnS
sdnns
```

One character per column is required. An alternative type casting mechanism is to specify the column type in the file itself.

```
ID,DATE,RAIN1,RAIN2,COMMENTS
s,d,n,n,s
ID110,20010609,.25,.10,
ID110,20010610,.56,.85,last valid value
```

In this situation, the Column Types entry would be

```
f
```

or

```
file
```

for 'file'. The line immediately following the label line(s) is considered the format. An error will be thrown if no format line is in place.

RDB files are the only file types in which Tkg2 insists that the column types are specified by the format line in the file.

File is RDB (quasi compliant, d|D for dates)

If the file is a RDB file, a tab delimited file with special restrictions, then all other file settings other than missing value and skip line identifier are reset behind the scenes. Note that Tkg2 is only quasi-RDB compliant as discussed in this section and will remain so. The biggest hangup is that RDB format does not distinguish between numbers and time. If time is a number, then to RDB the type is n, and if the time has non-numeric characterics, then the type is a string or s. Tkg2 requires d or D for proper date-time casting. Tkg2 will also handle the t or T data type—computed date-time. A basic RDB looks something like this:

```
# Stations in this file include:
#  08167000 GUADALUPE RIVER AT COMFORT, TX
#
datetime<t>value<t>code
10d<t>12n<t>3s
19390601<t>36<t>
19390602<t>32<t>
```

Notice in this example, that the first column has the 'd' type. An RDB file has an arbitrary number of lines in the header. These are the lines starting with #. A single label line is used and is followed by the format line. In RDB, the format line as numeric 'widths' to assist RDB compliant commands to produce formatted output. Tkg2 ignores these. In RDB the width number is to come before the type (10d), but Tkg2 does not care if the number follows the type (n12) because the number are not used by Tkg2. Also RDB does not permit comments inside the data. Tkg2 does because the Skip Line Identifier remains functional.

Import data (no dynamic loading)

Should the data be loaded into the Tkg2 file that you are about to create? If data is not loaded, then at runtime, Tkg2 will try to load the data in on the fly. This allows dynamic loading of data and template creation of Tkg2 files. This feature is extremely powerful and flexible. The dynamic loading of data is one of the most critically useful features of Tkg2. Therefore, it is turned on by default.

Skip handy x-y axis configuration on 1st data loaded

By default Tkg2 forces auto configuration of the all the axes when the first data is loaded into a plot. This is done because Tkg2 needs to track parameters such as the minimum and maximum data values. This is done regardless of whether the auto configuration options on the individual axes are toggled off. Occasionally, a user will have already configured the axes before data is loaded and does not want the axes changed. This checkbutton is not seen after the first data is loaded. Also, skipping the axis configuration had no meaning when Tkg2 dynamically loads data files automatically.

Use relative path for file Name

When data is dynamically loaded at runtime, you have a choice on having Tkg2 look for the data file along a absolute path (/u/wasquith/projects/datafile.txt) or along a relative path starting from the current directory of the Tkg2 file (projects/datafile.txt)—notice no leading '/'. If you are accessing data files that are at or below the current directory of the Tkg2 file or location where this Tkg2 file you are creating is saved, you can use the relative path. However, if you need to 'go up and over' or 'go up, over, and down' to find your data files, you must use absolute path names. Hence, turn this checkbutton off. Tkg2 will issue a warning blurb if you try to go up etc without the checkbutton off. The warning is not issued until after this dialog box is Ok'd. This functional behavior is needed because of the way in which Tkg2 does its dynamic loading. Tkg2 must be able to find data files in such a fashion that you can ship a Tkg2 file and subordinate data files to other directories and still have things work.

Do not verify/test field types (fast reading)

By default Tkg2 tests each and every entry in the data file as to whether it is a number, time, or string value. This is CPU expensive and is particularly so for time as Tkg2 uses an extremely flexible, but lengthy algorithm for conversion of time fields to a common basis for further internal processing. You usually can leave this option off until files get over about 75,000 values; unless you are working on a busy machine. The option provided in this dialog box is only applicable during the first importation. You can control the field verification with the —nofieldcheck command-line option, which of course only has meaning if you have not imported the data either by Import Data (see above) or later saving by File—SaveAs(with imported data) menu command.

Show me the data file contents

Forget what the data file looks like? Take a peek at it with this button. You are not able to modify the file as to slow you down and force you to work in an external text editor. The idea is to limit the potential of messing up important data.

ADVANCED tab

Convert ordinates to single column

A really cool feature for certain kinds of data sets. Multiple columns of ordinate data can be collated into a single column. The abscissa values get duplicated. The following diagram best demonstrates what this feature does.

```
Original Data Set
 X  Y1  Y2  Y3
 1  11  21  31
 2  12  22  32

Converted Data Set
 X  Y1:Y2:Y3
 1     11
 2     12
```

```
1       21
2       22
1       31
2       32
```

Thresholds — EXPERIMENTAL!!!!!!!!!!!!

Tkg2 permits some control (Ignore, Substitute, and Make missing) over what it should do with thresholds for box plots and encounters with data values having or < signs prepended to them. This is a highly experimental feature as there are logical complications in how to present the user interface and how to properly apply statistical adjustments to box plots. Because they are experimental, you the regular user might not even detect that something has or has not happened.

Route data through external program

A potentially very powerful utility. The read in data, which is stored in a single complex hash, can be dumped to the file system. An external program can pick up this dumped hash, processes it as needed, and return a revised dumped hash back to the file system. Tkg2 then substitutes the original hash with this new one. The script in Tkg2/Scripts/DoNoTransform_JustTest.pl script provides an example of the necessary frontend and backend to an external processing program.

Use a 'megacommand' in liew of a file

This is a cool feature that permits Tkg2 to grab the data from commands or pipe lines without having to actually create a file (well sort of, see below). For example to read the vertical bar (pipe symbol) delimited Tkg2 log file, one could type 'cat /tmp/tkg2.log' in the entry field. All other options on the dialog box such as delimiter, number of label lines, and column types apply. A really popular command to run might be something like /usr/local/bin/get_me_some_rdb_data.pl arg1, where arg1 is some argument for the program. This command retrieves some data in rdb format, so all you have to do is toggled the File is quasi-RDB checkbutton.

Megacommand has some command line options: —megacmd_args=<string> —megacmd_keep, and —megacmd_show. The args option permits you to pass arguments to the command. For example, suppose we used just /usr/local/Tkg2/Util/DVlastwk.pl without the station number. We could pass the number in via —megacmd_args=08167000. Additional arguments are provided by an underscore delimitation (—megacmd_args=08167000_-c30_45). One caveat is that Tkg2 must see some actual data from the command the first time around. This means that providing the station number for a arbitrary station is at first. Then you have to manual edit the Tkg2 file, seach for '-file' and then '-megacommand' to find the command and remove the station number. Then you can use the —megacmd_args.

The —megacmd_keep keeps the data files pulled by Tkg2 into the current directory without deleting these temporary files as Tkg2 exits. The —megacmd_show provides a verbose mechanism to see what Tkg2 is doing and how execution of the command is proceeding.

Either then —verbose or —megacmd_show command line options will show output to the terminal that looks like this.

```
A MegaCommand is retrieving the data.
  MEGA: %% /usr/local/Tkg2/Util/UVlastwk.pl -d=2 08167353  %%
  MEGA: Results of STDOUT directed to
  MEGA: /u/wasquith/tkg2_megacommand_file_4
  MEGA: which had 2057 lines.
```

To hit you see the exact command that was used as will as the location in which the megacommand temporary file was written to. The temporary files (like the postscript files) are written into the user's home directory.

Sort

Line plots are very ugly on unsorted X data. Click sort here to fix that. You are provided with some options for the sorting. Furthermore, sorting of alphabetical characteristics is possible

perhaps one would need this for some categorical or discrete plots.

Optional plot user name

You can give the plot a totally optional user name. The user name is a human readable and persistent string identifying the currently selected plot. You only need to specify the name if you plan on using the Tkg2 Instructions language for external control of Tkg2 graphic objects (see Instructions Help). The name provide here is not useful in any portion of an interactive Tkg2 session other than setting it and does not contribute to any type of labeling on the screen.

DATE-TIME tab

Convert date-time to a common base

This is another extremely cool and useful feature when you need it. Almost universally time series data is plotted at the time that the data occurs. At times, however, one might want to overlay all the data along a pseudo year in order to see the annual trajectories of the data or one might want to overlay all the data along a pseudo day in order to better visualize diurnal variations. Instead of having to preprocess your data files and risk tampering with your good data, let Tkg2 do it for you; albeit in a CPU expensive fashion. The feature works by providing a colon (:) delimited list specifying the common date-time base to use instead of the actual component as the date-time is internally processed just prior to graphic rendering. This feature works just as you would expect with dynamic loading and on a per data file basis.

For example, to visualize annual daily variation for a 1927 to 1990 data set on top of an already loaded 2001 daily data set, just enter 2001 in the field and all the years for the 1927-1990 data set will be converted to 2001. This will draw the data on top of the 2001 data, assuming the the 2001 data was already been loaded.

To view the diurnal variation of some subdaily increment data, entry 2001:10:01 to move all the data to October 1 of 2001. The hourly, minute, and seconds are left alone.

The general field look like the following: yyyy, yyyy:mn, yyyy:mn:dy, yyyy:mn:dy:hr, yyyy:mn:dy:hr:mi, or yyyy:mn:dy:hr:mi:sc. Fields can be left blank, set to '-', or set to zero if you do not want a particular field modified. For example, suppose you wanted all the data plotted into a single month (say October), then -:10 would be the field.

If the yyyy component is preceeded by the characters 'wy' or 'WY' then the water year is used and not the calendar year. This means that the year used for the months of October, November, and December will by yyyy-1 and not yyyy as will be done for the remaining nine months. This common water year feature is extremely handy if one desires to plot a water year (October to September) of data against another water year without having to preprocess the data file. For example, suppose you wanted to plot all the values for dates such as 1969/10/04 as the 2000 water year with the values plotted at noon and not 00:00:00, then set the common date time to 'wy2000:-:-:12:00:00'.

As a note, leap years are properly handled. If you provide a leap year, then Februrary 29th is shown, but if you do not then it is not.

Finally, your author finds this feature incredibly useful and has never seen it on commercial software.

Use noon

This button sets the common date-time to noon, -:-:-:12:00:00. This is handy when your date values do not contain a time compontent. Tkg2 defaults to 00:00:00 when there is no time component. Setting the -:-:-:12:00:00 sets the time component to 12:00:00 to all date values. Tkg2 does not dynamically determine whether your date-time values actually have a time component.

As WatYr

This button appends ‘wy’ to the beginning of the common date time value. This button acts to facilitate the water year conversion for users who have not read this particular help page. If the ‘wy’ is already prepended then the ‘wy’ is not added. If the common date time entry field is empty then a helpful message is placed into the field to tell the user what this button does.

Default

This button clears the common date-time field.

Date-Time Offset

Yet another extremely cool and useful feature when you need it. One can uniformally offset all time columns in a given data file by a floating point value for days. To pull a time series back one day and a half, enter -1.5 in the field and to push a time series forward seven days an a quarter, enter 7.5. This is great funcationality to have when one needs to compare the timing or overlap streamflow hydrographs and other ‘traveling’ phenomena. A message box is displayed with a warning if a non-numeric value is specified.

At present no other techniques exist to modify numeric or string column types during the data loading. It is hoped that this funcationality will also be included in future Tkg2 releases. For example, converting feet into meters on the fly.

The date-time conversion calculator within the Time Axis Editor will assist you in converting usual dates into the internal time representation of fractional days since January 1, 1900. Thus, to determine a suitable offset, all that is needed is to compute the days representation of two dates and take the difference between them. The days between 2001/10/01@00:12:00 and 2001/10/03@05:16:00 is (37165.219444 - 37163.008333) = 2.211111 days.

Days Calculator

Because computing the difference between two date-time values is hard, we have placed a calculator to compute the difference in floating point days between two arbitrary dates. This is really handy for determining suitable a date-time offset.

Compute date2 - date1

This button computes the offset and displays the result in the sunken field to the right of the button. Errors are shown inside the field.

Load days to offset

This button loads the computed days offset into the date-time offset field for convenience.

FURTHER DISCUSSION

Error Line Data

Additional description of the processing of error limits (and equivalently bars) is needed with specific attention to the handling of asymmetrical values denoted by the <=> string combination. In the following two or more space delimited data example, we have a record ID column, a column of Y-VALUES and their associated error limits (Y-RANGE), a column of X-VALUES and their associated error limits (X-RANGE). Notice that the <=> string is not provided in all fields and does not have to be. The — (double hyphen) string is a missing value, but this string is not available in all fields. Tkg2 considers null fields as missing regardless of the setting of the missing value string. Our attention here is on the X-VALUES and the X-RANGE fields. Pairs or groups of records were created to better distriguish between cases. Following each pair are a series of commented lines, which are shown by the leading #, that graphically illustrate what the X Error Lines will look like. The pipe symbol or vertical bar represents the whisker on the line and the equal signs represent the error lines. The graphics assume that X increases from left to right. The asterisk illustrates the data point when both X and Y values are known—notice that values for X and Y are not known in all cases. When the X or Y values are not known, the error line does not originate from respective axis coordinate, but instead an attempt is made to draw the error line between the two limits (see records 136 and 137). (Actually internally to tkg2, a fake X values is created on-the-fly halfway between the limits so two line renderings are still performed.) In the examples below, when this is done the asterisk is *not* shown. An individual description of each

situation follows the graphic. Further, the illustrations are in the X direction, but the generalization to the Y direction is obvious.

```
ID  Y-VALUES  Y-RANGE  X-VALUES  X-RANGE
125  2.90  1.35<=>      -0.05  -0.07<=>
126  2.80  1.46<=>1.46  -0.20  -0.22<=>--
#      |
#      |======*
#      |
```

One-sided with no whisker through data point because the maximum limit is not provided.

```
128  2.70    --<=>2.86  -0.53      <=>-0.50
129  2.60  2.69<=>2.69  -0.57    --<=>-0.55
#                    |
#             *=======|
#                    |
```

One-sided with no whisker through data point because the minimum limit is not provided.

```
130  2.50  2.40<=>--   -0.66       <=>
231  2.40  2.48        -0.39     --<=>--
#             *
```

Lone data point because of missing limits.

```
132  2.30  2.06<=>2.06  -0.27  -0.27<=>-0.26
133  2.20  2.20<=>2.20  -0.22  -0.22<=>-0.21
#             |        |
#             *========|
#             |        |
```

One-sided with whisker through the data point; the whisker is drawn through the point because -0.27 or -0.22 is provided.

```
134  2.10      <=>2.2   0.18   0.14<=>0.18
135  2.00  2.60<=>2.60  0.00  -0.03<=>0.00
#      |      |
#      |======*
#      |      |
```

One-sided with whisker through the data point; the whisker is drawn through the point because 0.18 or 0.00 is provided.

```
136  1.90  1.64<=>1.95   --  -0.06<=>-0.02
137  1.80  1.29<=>1.29   --   0.05<=>0.07
#      |           |
#      |===========|
#      |           |
```

Asymmetrical error line but the data point is unknown so the line is drawn from the minimum to the maximum. An extremely important note is needed about this case. Although the error line is drawn there is *no* data point to draw. This means that a subsequent text plot of ID plotted next to the point symbol for each X and Y VALUE could *not* be plotted for these two records.

```
138  1.70    --     -0.02  -0.03<=>0.00
139  1.60  1.34<=>1.70   0.00  -0.01<=>0.01
#    |      |
#    |======*====|
#    |      |
```

A normal(?) asymmetrical error line.

```
140   1.50   1.34<=>1.34   -0.50   -0.50<=>-0.50
141   1.40   1.34<=>1.34    0.05    0.05<=>0.05
#                |
#                *
#                |
```

Single whisker through data point because the values for the X-RANGE are equal to the X-VALUE.

```
142    1.30   1.34<=>1.34     --         <=>
143    1.25    --              --       --<=>--
```

Neither 142 or 143 can be plotted as both X and Y are unknown and can not be faked because both minimum and maximum limits are unavailable.

THE FILE MENU

This POD file provides detailed description of the **File** menu available at the top of each template container. This documentation file is located at **Tkg2/Help/FileMenu.pod**.

INTRODUCTION

The Tkg2 file menu closely follows the traditional file menu by providing new, open, save, and exit functions. The menu has the following entries:

```
New
Open
---------
Save
Save As (filename requested)
Save As (with imported data)
HASH Format (preferred)
  Compact the hash
---------
Export
---------
Print (postscript)
Print and Exit
---------
Close
Exit Tkg2
```

THE ACTIONS

The following are the actions or commands supported by the menu.

New and Open

The New and Open features work like expected. New launches the create template dialog box in which the page size of the template is specified. New functionality is provided by the Tkg2/DeskTop/CreateTemplate.pm module. Open launches a getOpenFile Tk widget group and the user is provided a familiar dialog box to specify the file to open by Tkg2. Open functionality is provided by the Tkg2/DeskTop/OpenSave.pm module.

Save, Save As, and Formats

The Save and Save As features work like expected. Save will save the template into the existing file name. Template has not been given a file name yet, then the Save As is called automatically. The Save As prompts the user for a file name. Save functionality is provided by the Tkg2/DeskTop/OpenSave.pm module. Tkg2 currently supports hash (ASCII) storage.

The Save As (with imported data) hard loads all dynamically loaded data into the file and toggles the embedded switches not to load the data at run-time. This is a particularly useful feature when one desires to send another Tkg2 user a file without having to ship them the data files.

The hash format is the preferred format as the Tkg2 graphics objects are dumped into a hash that is really just Perl code. The compact switch strips all newlines and padding spaces from the hash format and reduces the Tkg2 objects to the minimal size that is still executable Perl code. Hand editing of hash format is relatively simple. Plus, years into the future, this format will still be readable and the image contents derivable even if Tkg2 is not around or long since forgotten. Archival of data is a very important feature of Tkg2 as everything is ASCII based. The compact the hash option produces the smallest hash possible—-no extra 'whitespace'. The compact representation might be useful on some systems for speeding up the initial startup sequence.]

Exporting

Exporting of Tkg2 files to other formats is not accomplished using a selection list in the Save As like many other applications, but is instead performed through separate interface. The native export format is postscript as the Tk::Canvas only supports a postscript dump. Fortunately, there are numerous utilities to convert postscript to other formats. The are four principle output formats: MIF, PDF, PNG, and PS (postscript).

Framemaker Interchange Format (MIF) is for importation into Adobe Framemaker. Framemaker is an incredible application that sadly does not seem to garner the attention that other Adobe products do. MIF is a very simple ASCII vector graphics format that is highly tagged. These tags make editing by external scripts much easier than trying to edit postscript directly. The `pstoedit` open-source utility is used to convert the Tk postscript to MIF. Another utility distributed with Tkg2 is the `tkmiffix.pl` script that provides some adjustments to the `pstoedit` MIF for better handling for Framemaker importation and graphics editing—this is partially motivated by your author's habits in Framemaker use. All adjustments to the MIF file performed by the `tkmiffix.pl` script are identified by comments in the file. Relevant Tkg2 codes is found in Tkg2/DeskTop/Rendering/RenderMIF.pm, Tkg2/DeskTop/Batch.pm, and Tkg2/DeskTop/Printing.pm.

The Portable Document Format (PDF) is a common format that most users should already by familiar with. In the author's opinion PDF is under utilized for document transmission than it should be compared to MS Word. PDF generation was originally done by `pstoedit`, but Ghostscript now provides the postscript to PDF conversion. Relevant Tkg2 code is found in Tkg2/DeskTop/Rendering/RenderPDF.pm, Tkg2/DeskTop/Batch.pm, and Tkg2/DeskTop/Printing.pm.

The Portable Network Graphics (PNG) format is used for the raster export format of Tkg2. Support for gif will not be provided as PNG is the nature successor of GIF without the patented compression issues. PNG generation is provided by the `wpng` utility along with Ghostscript, but this may change in future versions. Relevant Tkg2 code is found in Tkg2/DeskTop/Rendering/RenderPDF.pm, Tkg2/DeskTop/Batch.pm, and Tkg2/DeskTop/Printing.pm.

The postscript (PS) format is the native dump of Tk::Canvas graphics. At the present time (fall 2000), there appears to be a bug in the postscript output. This bug involves the translate command in the file. The first translate command need to be modified to properly support non-8.5x11 page sizes. The script, `Tkg2/Util/tkpsfix.pl`, provides a conversion filter. More research in the postscript generation by Tk is needed. Relevant Tkg2 code is found in Tkg2/DeskTop/Rendering/RenderPS.pm, Tkg2/DeskTop/Batch.pm, and Tkg2/DeskTop/Printing.pm. The `tkpsfix.pl` script writes a debug file in user's home directory titled **.tkg2_tkpsfix_debug** if an only if this file already exists. The contents of this file

contain the original lines in the postscript file that are now 'fixed' in the postscript file. You can create the debug file like this:

```
% cd ~
% touch .tkg2_tkpsfix_debug
```

See discussion in the Printing section of this help page for more information on some of the features provided in the Export dialog box.

The `tkpsfix.pl` fixes many problems, but there are reports of large format printers (HP755CM) wasting copious amounts of paper when printing from Tkg2 with or without the `tkpsfix.pl` program being run. Thus, additional tweaks to the postscript output are needed. Some Tkg2 users report that if they placed the following commands after the `%%EndComments` line near the beginning or header of the postscript file, the file spooled properly to the printer.

```
%%Beginfeature: *PageRegion AnsiC
2 dict dup /PageSize [2448 1775] put dup
/ImagingBBox null put setpagedevice
%%End Feature
```

It is possible that the two values after the `/PageSize` will require adjustment for specific page sizes or printer dimensions.

Printing

Printing (see Exporting) is performed by dispatching the the dumped postscript output to a specified printer along the user's printer queue. The printer queue can be changed using the tkg2rc files. See the Tkg2rc.pod for more details. The printers are determined by the tkg2rc files or the `lpstat -v` command on Solaris. On unix there are a few variations on the printing queue. For example, Solaris uses 'lp', while Linux uses 'lpr'. Another popular print utility is 'pdq' For this reason, Tkg2 consults a environment variable $PRINTER_QUEUE to determine the command to use to spool Tkg2 files. In the absence of the variable, Tkg2 uses `lp -c`. Relevant Tkg2 code is found in Tkg2/DeskTop/Rendering/RenderPS.pm, Tkg2/DeskTop/Printing, Tkg2/Tkg2rc.pm, and Tkg2/DeskTop/Batch.pm.

There are four additional printing options that are set below the selection of the printer. First, two options (no rotate or 90 rotate) of postscript orientation are provided. Tkg2 usually defaults to the most appropriate for printer spooling, but in the export dialog box, one might want to turn off. This is especially true for non-postscript exporting.

Second, two options (rescale fonts or do not rescale fonts) on the canvas are provided. The default is for fonts to be rescaled. The Tk canvas on which Tkg2 renders graphics is not quite as advanced as one would want so zooming in and out is not easy. For this reason, Tkg2 has a —zoom multiplier on font sizes to make fonts readable on the screen. However, when one goes to print a file, this zooming on the font size is not desirable. Tkg2 temporarily sets the global variable —zoom to unity, redraws the canvas, spools to the printer, and then redraws the canvas with the original —zoom setting. If your canvas is huge, redrawing might be irritating, but this is a nice feature to have as default.

Third, two options (bypass the postscript correction script or not) on postscript postprocessing are provided. See the discussion in the Exporting section of this help page.

Fourth, two options (color or mono) on the postscript rendering are provided. Use color if you want color in the postscript or use mono if you do not want color. Tk supports greyscale, but there appears to be a bug on some systems(?), so Tkg2 does not provide this utility as of August 2001.

Exiting

The Close action exits the current template with confirmation, but does not entirely exit Tkg2. The Exit Tkg2 action exits all templates and Tkg2 entirely. The Exit action is not provided in the menu when the —nomw commandline option is used. Relevant code is found in Tkg2/DeskTop/Exit.pm.

FURTHER DISCUSSION

The Tkg2/MenusRulersScrolls/Menus.pm module generates the actual menus seen. One of the great design features of Tkg2 is that the main dialog is small and easily shuffled on the desktop and each Tkg2 template is in its own window and entirely self sufficient.

THE EDIT MENU

This POD file provides detailed description of the **Edit** menu available at the top of each template container. This documentation file is located at **Tkg2/Help/EditMenu.pod**.

INTRODUCTION

The Tkg2 edit menu is still an area of research and development as it is not entirely parallel to edit menus seen in other applications. The menu has the following entries:

```
Undo 1
Undo 2
---------
Update Canvas
Step-Wise Update
---------
View Dumped Template
```

THE ACTIONS

The following are the actions or commands supported by the menu.

Undo 1 and 2

Tkg2 has two levels of undo. The code could be modified for unlimited levels of undo. Your author has not spent much time working on a redo, never the less, undo 1 can be used in a redo sense. Undo functionality is provided by the Tkg2/DeskTop/Undo.pm module.

Update Canvas

The `Update Canvas` instructs tkg2 to redraw everything on the canvas. This is a seldom used feature, as all dialog boxes exit with an update of the canvas. The main use of update canvas is when the user switches the 'Draw Data on Canvas Update' variable in the 'Global Settings' menu. For moderate to large data sets, turning the data drawing off can make editing the plots and annotation more efficient. The switch will be temporarily turned on when printing. The update canvas code is located in Tkg2/TemplateUtilities.pm::UpdateCanvas.

Step-Wise Update

This action functions just like the `Update Canvas` except that all major steps of the canvas drawing are incremented through. The user is prompted along standard output whether the step was properly performed or at least believed to be properly performed. The user answers yes or no. Details are written to the `~/.tkg2message` file. The `Step-Wise Update` is intended to help the author diagnosis problems that users might experience.

View Dumped Template

Performs a Data::Dumper dump to ASCII of the template hash an recursively traverses deep data structures and loads the text into a text widget. This feature is primarily for the developer during debugging, but could be informative to other contributors. The output is essentially identical to a Save with the Hash format except that anonymous subroutines and some process dependent information are not removed.

FURTHER DISCUSSION

THE PLOT MENU

Detailed description of the **Plot** menu is provided below. This documentation file is located at **Tkg2/Help/PlotMenu.pod**.

INTRODUCTION

The Tkg2 plot menu is used to create and otherwise edit plots on the canvas. The menu has the following entries:

```
Add Plot by Dragging
Add Plot by Editor
---------
Select Plot
Plot Editor
X-Axis Editor
Y-Axis Editor
Y2-Axis Editor
---------
Copy Plot
Cut Plot
Paste Plot
Delete Plot
---------
Raise Plot
Lower Plot
---------
Show explanation
Hide explanation
Auto axis configuration ON
Auto axis configuration OFF
```

THE ACTIONS

The following are the actions or commands supported by the menu.

Adding a Plot

Plots can be added to the canvas by two methods. The first method, 'Add Plot by Dragging' is by dragging the mouse with start and ending clicks of the first mouse button. In this way, a plot is created in an interactive fashion. The second method, 'Add Plot by Editor' is by launching the Plot Editor, setting the margins, and hitting the OK button. Either method is identical in its results. Multiple plots are easily create. Creating one or more plots automatically from the command line is also possible, see the —mktemp and —mkplot command line options. Command line options are reviewed by typing 'tkg2 —help' at the command line.

Select Plot

The select plot feature provides a scale or selection dialog for the plots. The scale identifies all plots that are on the canvas, whether visible or not. This is an important feature, because it provides a means to grab and edit plots otherwise untouchable by the mouse. Examples of untouchable plots include plots that were made too small, or plots in which the -doit checkbutton has been turned off (see the Plot Editor).

Plot Editor

The plot editor is used to edit plot-wide parameters such as margins, background color, or border thickness. The plot editor is a relatively complex dialog box that controls many features. Thus, the plot editor is not discussed in detail here, (see Tkg2/Help/PlotEditor.pod file).

Axis Editors

The axis editors are used to edit axis specific parameters such as minimum and maximum limits, tick lengths, or axis titles. The axis editors are relatively complex dialog boxes with many features. Thus, the axis editors are not discussed in detail here, (see Tkg2/Help/AxisEditor.pod file).

Copy, Cut, Paste, and Delete

The common editing operations of copying, cutting, pasting, and deleting a plot are performed here and not in the Edit menu. The partial motivation for this is that the editing functions for a major drawing object (a plot) are compartmentalized with other plot editing features. Also, you author has not figured out a suitably simple way in which to implement general cut, copy, paste, and delete functions in the Edit menu.

A plot must be selected first for these operations to work. A plot is selected by clicking the first mouse button once somewhere in the interior of a plot or using the Select Plot feature. When a plot is pasted a small horizonal and vertical offset to the right and down is made. The plot will have to be moved by clicking with the third mouse button. It is not currently possible to paste a plot at the current location of the mouse.

Raising and Lowering Plots

Raise plot and lower plot are simply means to change the drawing order of the selected plot. See the previous section on selecting a plot. A bring to front or move to back feature is not provided as these have been deemed unnecessary.

Show explanation

Toggle the explanation *on* for the selected plot. A similar action is found in the Data menu, but operates on all plots on the screen. This is can also be performed, and historically was, by the Explanation tab in the PlotEditor (double-left click on plot).

Hide explanation

Toggle the explanation *off* for the selected plot. A similar action is found in the Data menu, but operates on all plots on the screen. This is can also be performed, and historically was, by the Explanation tab in the PlotEditor (double-left click on plot).

Auto axis configuration ON

Toggle all the automatic axis configurations *on* for the selected plot. A similar action is found in the Data menu, but operates on all plots on the screen. This is can also be performed, and historically was, by the Plot tab in the PlotEditor (double-left click on plot) or in the AxisEditors (double-left click on an axis).

Auto axis configuration OFF

Toggle all the automatic axis configurations *off* for the selected plot. A similar action is found in the Data menu, but operates on all plots on the screen. This is can also be performed, and historically was, by the Plot tab in the PlotEditor (double-left click on plot) or in the AxisEditors (double-left click on an axis).

FURTHER DISCUSSION

Much of the editing code is found in `Tkg2/MenusRulersScrolls/Menus.pm` which then calls methods:

```
Tkg2/Plot/Editors/ContinuousAxisEditor.pm
Tkg2/Plot/Editors/DiscreteAxisEditor
Tkg2/Plot/Editors/PlotEditor.pm.
```

Relavent code to moving or resizing a plot around the screen is:

```
Tkg2/Plot/Movements/DraggingPlot.pm
Tkg2/Plot/Movements/MovingPlot.pm
Tkg2/Plot/Movements/ResizingPlot.pm.
```

THE DATA MENU

This POD file provides detailed description of the **Data** menu available at the top of each template container. This documentation file is located at **Tkg2/Help/PlotMenu.pod**.

INTRODUCTION

The Tkg2 data menu is used to add and remove data from plots and to perform other data oriented operations. The menu has the following entries:

```
Add Data File to Selected Plot
[] Do not update canvas when data added
---------
Edit Data or Do Statistics
View Internal Data
---------
Edit Y1 Reference Lines
Edit Y2 Reference Lines
---------
```

```
Edit Y1 Quantile-Quantile Lines
Edit Y1 Quantile-Quantile Lines
---------
Show explanations for all plots
Hide explanations for all plots
Auto axis config. ON for all plots
Auto axis config. OFF for all plots
```

THE ACTIONS

The following are the actions or commands supported by the menu.

Add Data File to Selected Plot

The add dataset provides the only interface for actually loading data into a plot for the first time. This feature launches the 'Add Data File to Plot' editor. Numerous features regarding the plot type to create, which y axis the data is to be plotted against, and the myriad of issues involving the reading and parsing of data files are specified. Because there are so many features provided by the dialog, consult `Tkg2/Help/AddDataFile.pod` for more details. Relevant code is contained in `Tkg2/DataMethods/Class/AddDataToPlot.pm`.

[] Do not update canvas when data added

This checkbutton provides the setting of a feature to not redraw the canvas when data is finished loading into the plot. This is a feature that if true allows the user to load all of their data in without the time consuming redrawings. This is particularly helpful on slow remote links. The Update Canvas method in the EDIT menu turns this temporarily on. The default is for the data to be drawn, which means that this checkbutton off.

Edit Data or Do Statistics

Data is loaded into a file in a file and then column pair centric fashion. When edit data is selected the Data Class (File) Editor, `Tkg2/DataMethods/Class/DataClassEditor.pm`, is launched. This editor provides a list of all the data files read into the selected plot. If a data file was read more than once, then a number is appended to the file name. The plotting order of the data from the files can be changed with the buttons on the right. The data from the file can be delete for removed from the plot by selecting the **Delete from Class** button. All data for a plot can be removed by the **Delete All** button. The **Edit Data Set** button launches the Data Set Editor (`Tkg2/DataMethods/Set/DataSetEditor.pm`), which permits similar control of the data on a column by column basis. For example, the Data Set Editor can control the plotting order of the data from a given file, the text in the explanation, and launch the editor to modify the plotting style.

View Internal Data

This feature is not yet implemented. In time, it will be possible to view the data that has been loaded. This feature will be tightly related to the —pretty_data command line option or some command like that.

Edit Reference Lines

The Y1 and Y2 options launch the Reference Line editor. The editor holds a table of reference lines for a given y axis. Each y axis has its own table. Reference lines are a potentially very useful tool for marking exact locations on the plot. Reference lines are like annotation lines in the sense that they annotate a plot and their drawing style can be altered. However, references lines differ in that they are drawn on a per plot basis using the units of the plot scales and not in page units. If a plot is resized or its axis limits altered, then the location or length of the line will change.

Like the other annotation types, reference lines can be given user names and toggled on an off using the Instructions. See the discussion about Annotation in general in the `Tkg2/Help/AnnoMenu.pod` help file. Code relevant to reference lines is in `Tkg2/Anno/ReferenceLines.pm`.

Tkg2 might eventually support reference symbols.

Edit Quantile-Quantile Lines

Quantile-quantile lines are very simple. Basically a quantile line is a line that plots at a +/- 1 to 1 slope. Each axis can have both -1 and 1 sloped quantile-quantile lines turned on. A double y plot can have thus four quantile-quantile lines shown. Quantile-quantile lines are useful when

predicted vs observed plots are made. Relevant code is found in `Tkg2/Anno/QQLine.pm`.

Show explanations for all plots

The explanations for all plots will be toggled *on* by this action. Extremely handy feature to have when multiple plots are on the screen and adjustments to plotting styles are to be made. Relevant code is found in `Tkg2/Plot/Plot2D.pm showExplanation`.

Hide explanations for all plots

The explanations for all plots will be toggled *off* by this action. Extremely handy feature to have when multiple plots are on the screen but the explanations are not desired. Relevant code is found in `Tkg2/Plot/Plot2D.pm showExplanation`.

Auto axis config. ON for all plots

Opposite of turning the auto axis configuration OFF, see discussion below.

Auto axis config. OFF for all plots

By default Tkg2 uses its internal algorithms to determine appropriate axis limits, ticking steps, labeling steps, and other axis settings for X, Y, and second Y axes. This is done to promote the use of Tkg2 as a batch processing graphical markup engine. However, it is convenient to have a way to turn all of them off for all the plots because often one wants either all dynamic configuration or not. Individual minimums and maximums for individual axes can still be controlled via the PlotEditors (double-left click in plot) or the AxisEditors (double-left click on axis). Relevant code is found in `Tkg2/Plot/Plot2D.pm toggleAxisConfigurations`.

FURTHER DISCUSSION

THE ANNOTATION MENU

This POD file provides detailed description of the **Annotation** menu available at the top of each template container. This documentation file is located at **Tkg2/Help/AnnoMenu.pod**.

INTRODUCTION

Tkg2 support three types of annotation on the canvas: text, line, and symbol. Each of these types is created from the Annotation menu. The menu has the following entries:

```
Text
Line
Symbol
---------
Select Text
Select Symbol
Select Line
---------
draw Anno first
---------
  draw Text first
           second
            third
draw Symbol first
           second
            third
  draw Line first
           second
            third
```

The first three entries create a new annotation object. The user will see that the cursor changes to a plus sign. When the first mouse button is pressed, the text and symbol annotation is placed—its origin determined. Subsequently, the Editor for the corresponding annotation type is launched. The user then can configure the annotation and pressing the OK button will show the results. Line annotation is slightly different, on the first press of the mouse button the line begins and is moved until the mouse button is pressed again. The line annotation editor is not automatically launched.

The second three entries provide a scale or selection dialog for the corresponding annotation type. Each scale identifies all annotation objects that are on the canvas, whether visible or not. This is an important feature, because it provides a means to grab and edit objects otherwise untouchable by the mouse. Examples of untouchable annotation include annotation that is made too small, empty text annotation, annotation color that matches the background, or annotation in which the -doit checkbutton has been turned off.

The *draw Anno first* checkbutton toggles the annotation drawing ahead of the plot object drawing. In the Tkg2 UpdateCanvas subroutine all plot objects by default are drawn ahead of the annotation object. This includes all three object types.

The remaining 9 radiobutton entries pertain to the drawing order of the individual annotation types when annotation is being drawn. Be careful, at the present time, there are no checks on the uniqueness of the selections. For example, if all types were to be drawn first, then only the text annotation will be drawn. This is considered a feature, in that the user and relatively easily toggle symbol and line annotation on or off. Since text is the first annotation considered in the list of drawing, it isn‘t possible to toggle it on and off with this version of Tkg2. The following warning is issued along stderr.

```
Tkg2-warning: One or more of the annotation elements (text, symbol,
line) have the same drawing order.  Thus, one or more of these
elements will not be visible on the screen.  Please consider revising
your drawing order in the ANNOTATION menu.
```

GENERAL MOUSE BEHAVIOR

The annotation editors for each of the annotation types are launched by double clicking the third mouse button and not the first. Copying and Deleting of annotation is performed by copy and delete buttons in the editor.

The text and symbol annotation are selected and moved using the first mouse button. The nodes of the line annotation also are selected and moved using the same button. At the present time, an entire line can not be moved at once.

Text Annotation

Text annotation is perhaps the most complex of the annotation objects. The text annotation editor provides a seven line text widget for typing in one or more lines of text. The doit checkbutton toggles the drawing of the widget on and off—a feature most popular with batch processing and external scripting. The anchor menubutton controls the compass direction in which the text will be drawn around the origin. The origin is set by the mouse during placement and object moving. A nw (northwest) origin is probably easiest to use and is the default. Center is another popular choice as all of the text will be centered horizontally and vertically around the origin. The remainder of the text options should be self explanatory.

The Tk::Canvas does not presently support text rotation so the text angle has no effect. However, it is important to know that if a file is being exported to the Framemaker Interchange Format (MIF), that each and every text in containing the string ‘<Ang 90> or <Angle -45>’ other small variations on the characters making up ‘Angle’ are possible. When this mif file is imported into Framemaker, the text will be rotated around the left edge of the text (see Tkg2/Util/tkmiffix.pl script). The stack text option adds a new line after each character in an effort to mitigate for the lack of text angle specification.

The user name entry field is not self explanatory. All of the Tkg2 drawing objects can be given an internal name, this is primarily for the benefit of the user. The name provides a means to persistently label an object when a Tkg2 file is saved, and most importantly the user name provides a mechanism for the external scripting Tkg2 instruction language (see —inst in the command line). The user name has no effect on the drawing.

The text annotation has the ability to load external files as text or run external commands (on Unix-like systems) and capture the standard output. There are five types of behavior that need description.

Soft File Loading
If the text from the dialog box or from the Tkg2 file matches the following '<softcat: filename>' at the beginning of the text field, then the contents of the file (if it exists) are read and inserted as the text at draw time. This file will be read with each update of the canvas. Filename can include a path.

```
<softcat: /tmp/tkg2.log>
```

Hard File Loading
If the text from the dialog box or from the Tkg2 file matches the following '<hardcat: filename>' at the beginning of the text field, then the contents of the file (if it exists) are read and inserted as the text at draw time. This new text is permanently loaded into the Tkg2 file and will be preserved if the Tkg2 file is saved. Filename can include a path.

```
<hardcat: /tmp/tkg2.log>
```

Soft External Command
If the text from the dialog box or from the Tkg2 file matches the following '<softeval: expression possibly with pipes>' at the beginning of the text field, then the string after the 'eval:' is invoked as a shell command whose STDOUT is piped into the read by Tkg2. The command is run each time. Here is an example:
Insert the output from the `date` command.

```
<softeval: date>
```

Hard External Command
If the text from the dialog box or from the Tkg2 file matches the following '<hardeval: expression possibly with pipes>' at the beginning of the text field, then the string after the 'eval:' is invoked as a shell command whose STDOUT is piped into the read by Tkg2. The output from the command is permanently loaded (see 2). Here is an example:
Insert filtered output of the `last` command—

```
<hardeval: last | grep asquith>
```

Simply use the Text
If one of the previous four behaviors are not triggered, then the contents of the text field are simply drawn on the screen.

Symbol Annotation

The symbol annotation provides the doit checkbutton and the user name entry field like the other annotation types (see Text Annotation). The symbol type is selected with the symbol menubutton. The list of symbols includes: circle, square, triangle, cross, star, horizontal bar, and vertical bar. These are the same symbol types support in the plots. The remainder of the options include symbol fill color, outline color, size (in inches), outline width, and angle. The angle is operational unlike the angle in the text annotation.

Line Annotation

The line annotation is likely the second most complex annotation object owing to the support for arrows and differing line styles. Like the text annotation, a doit checkbutton is provided for toggling the drawing on and off. A user name entry field is provided (see Text Annotation). The arrow style should be self explanatory except for the arrow distances. The first distance is the distance along the line from the neck of the arrow to the tip. The second distance is the distance along the line from the trailing points of the arrow to the tip, and the third distance is the distance from the outside edge of the line to the trailing points. The distances are in inches.

FURTHER DISCUSSION

Developers interested in Tkg2 annotation workings should consult the following modules:

```
Tkg2/Anno/Text.pm, Line.pm, Symbol.pm, SelectAnno.pm
Tkg2/MenusRulersScrolls/Menus.pm
```

```
Tkg2/TemplateUtilities.pm::UpdateCanvas
Tkg2/DeskTop/Instructions.pm
```

While the annotation is reasonably well throughout and highly compartmentalized, some improvement is needed. Areas of improvement include better mouse handling and global cut, copy, and pasting. Box annotation needs to be added.

THE GLOBAL SETTINGS MENU

This POD file provides detailed description of the **Global Settings** menu available at the top of each template container. This documentation file is located at **Tkg2/Help/SettingsMenu.pod**.

INTRODUCTION

The Tkg2 global settings menu is used to control several global variables. These settings across all templates in the currently running tkg2 process (see Further Discussion). The menu has the following entries:

```
Draw Data on Canvas Update
Delete Loaded Data when Saved
Snap to Grid
---------
Edit some Global Variables
```

THE ACTIONS

The following are the actions or commands supported by the menu.

Draw Data on Canvas Update

When the variable is turned **on**, then data are drawn on their respective plots. When the variable is turned **off**, then data is not drawn. Regardless of the variable setting, data is drawn on the first rendering of the canvas and if the canvas is exported or printed. Turning data drawing on and off is a means to make editing of plots and annotation easier when a large amount of data make the rendering process take more than a couple of seconds to complete. The variable can be controlled from the command line with `–drawdata or –nodrawdata` and the tkg2rc file with `Tkg2*redrawdata`, see CmdLine.pod and Tkg2rc.pod help files. Relevant code is found in `Tkg2/TemplateUtilities.pm::UpdateCanvas`. The variable is named `$::TKG2_CONFIG{-REDRAWDATA}`.

Delete Loaded Data when Saved

When the variable is turned **on**, then any dynamically loaded data is deleted when a template is save. The data will be re-read from the files during the next opening of the file. This is the default procedure. However, there are circumstances in which it is needed that the data be hard loaded or permanently loaded into a tkg2 file even though it was originally dynamically loaded. Turning the variable **off** will turn the internal key `-dataimported` on (set it equal to one) for each data set. The variable can be controlled from the command line with `–importdata` and the tkg2rc file with `Tkg2*delete_loaded_data`, see CmdLine.pod and Tkg2rc.pod help files. Relevant code is found in `Tkg2/TemplateUtilities.pm::DataOnTheFly` and `Tkg2/DeskTop/Batch.pm::DeleteLoadedData`. The variable is named `$::TKG2_CONFIG{-DELETE_LOADED_DATA}`.

Snap to Grid

The grid snapping is toggled on an off. Like the other two global variables, the toggle applies to all templates. It is not possible to alter the grid snapping distance. The distance is 0.125 inch. Relavent code is found in `Tkg2/TemplateUtilities.pm::snap_to_grid` and `Tkg2/MenusRulersScrolls/Rulers.pm::_drawRulers`.

Edit some Global Variables

Edit some Global Variables launches an entry based editor to change the values of some of the global variables. Currently, the only global variables that make sense to configure once tkg2 is up a running is the plotting position coefficient for direct probability axis support and the zoom factor on the fonts.

FURTHER DISCUSSION

It is perhaps a bad design decision that these variables are truly global across the tkg2 process. This issue has a deep legacy. Future versions of tkg2 might have just 'Settings' and not 'Global Settings', which will apply on a per template basis.

THE PLOT EDITOR

This POD file provides detailed description of the **Plot Editor** dialog. The plot editor is accessed either by selecting a plot and using the `Plot / Plot Editor` menu option or double clicking within the borders of a plot. This documentation file is located at **Tkg2/Help/PlotEditor.pod**.

INTRODUCTION

The Plot Editor is used to control plot-wide parameters such as plot margins, border color and thickness, and much of the explanation. The Plot Editor consistents of three note book tabs: Plot, Plot Title, and Explanation. Each of these tabs contains widgets to control applicable parameters. The buttons at the bottom of the dialog: Apply, OK, Cancel, Edit X-Axis, Edit Y-Axis, and Help are always available.

The **Apply** button redraws the canvas and any potential changes in the dialog box are applied. The **OK** button is identical to Apply except that the dialog box is exited. The **Cancel** button exits the dialog box without redrawing, hence applying any changes. The **Edit X-axis** button launches the X axis editor and the **Edit Y-axis** button launches the Y axis editor. A button for the second or Y2 axis is not provided in the interests of space and its rare usage. The all of the axis editors are accessable by either double clicking on the respective axis with the first mouse button or queried from the PLOT menu.

PLOT tab

Plot Margins

The left, right, top, and bottom margins are set by the entry widgets. The units are in inches and the 'i' can be left off when changing a setting.

Page Color

The page color is controlled by the 'Page: Color' widget. The identical widget is made available through the plot editor of each plot on the page.

Autoconfigure Axis Limits

Tkg2 has a very powerful mechanism in which nice looking axis limits and other axis settings are determined on-the-fly as data is loaded. By default this behavior is turned on. If the user changes axis settings such as minimum, maximum, step length, or log cycles to label and does not turn the autoconfigure off, then the next time that tkg2 starts up the changes by the user will be overwritten. This might be surprizing, but this is considered a feature. If all the data was hard loaded into the plot, then this is not an issue.

Either the automatic determination of either or both the minimum and maximum is possible. When the center is toggled and an axis is linear, then the linear axis will have the origin in the center of the plot. This is a neat feature for residual analysis.

Often the user will start by have the auto turned on, load the data into their plots, turn the auto off, and then change the axis limits and other settings. If the file is then saved and tkg2 re-opens the file, then the user's settings will be used. This works whether the data is hard or soft loaded.

The relavent code is found in `Tkg2/Plot/AxisConfiguration.pm`.

Square Axis

On rare occassions it is desirable that the Y or Y2 axis be square relative to the X axis, that is the number of linear units per inch or log cycles per inch is the same for each axis. If the checkbutton is toggled then Tkg2 will square the axis if and only if (iff) the axis types are equal or the same. It is impossible to make a linear-log combination square. Squareness can be applied separately for the first and second y axis. The squareness feature is handy if you need to produce a map view of horizontal coordinates from a topographic survey. User feedback indicates considerable need for square log plots as only when these are square can certain manual mathematical operations be performed from the graphs.

Border Width and Color

The border width or thickness and the color are controlled with the 'Border: Width and Color' widgets.

Plot Background Color

The back ground color of the plot is controlled with the 'Background: Color' widget.

Switch X/Y Axis

The X and Y axes are switched. This is experimental and not yet fully implemented. Sorry.

All Axes to Percent Base

The X, Y, and Y2 axes are converted to a percentage basis. Each axis type is converted to linear. The minimum is set to zero; the maximum is set to 100; the major tick interval is set to 10; and the no. of minor ticks is set to 4.

All Axes to Frac. Percent Base

The X, Y, and Y2 axes are converted to a factional percentage basis. Each axis type is converted to linear. The minimum is set to zero; the maximum is set to 1; the major tick interval is set to .1; and the no. of minor ticks is set to 4.

Plot Name

Tkg2 has a powerful external 'scripting' language provided by the `Tkg2/DeskTop/Instructions.pm` module that can be used to control all tkg2 objects. One of these objects is the plot object. The Plot Name entry field is a means to attach a user defined string to the plot so that the instruction language can identify the plot. The plot name will be written to the tkg2 file during a save. There is no reason to name a plot unless the user want to use the external Instructions. Further details about Instructions is available in the 'Instructions.pod' help file.

DoIt

The DoIt check buttons turns the drawing of this plot on and off. A handy feature indeed, especially when tkg2 plots are modified by the external Instructions. When the DoIt is turned off, then the plot will not be seen and hence not selectable by the mouse. However the plot can be selected and the plot editor launched from the Plot menu using the 'Select Plot' feature.

PLOT TITLE tab

The Plot Title tab provides a quick and convenient method to add a text annotation to the plot. The location of the text is set by a vertical and horizonal offset from the top middle border of the plot. As the plot is moved or resized then text will be moved along as well. Regular text annotation does not have a spatial connection to the plot. The text is set by the **Title** widget. Multiple lines of text can be made by adding a literal '\n' to the string.

The text font, size, weight, slant, and color are controlled by their respective widgets. The horizonal (x) and vertical (y) offset is set by the Title X-offset and Y-offset entry fields. The offset units are in inches, which is identified by the appended 'i'. The Stack Text checkbutton will add a newline after each and every character in the title. This is supposed to mitigate for the lack of text rotation capabilities by the Tk::Canvas.

EXPLANATION tab

Hide Explanation

The checkbutton hides, but does not destroy, the explanation when toggled on. The default is off so that the explanation is shown when a plot is first created. Because it is so common to toggle a explanation on and off, menu short cuts are provided to make life a little easier. These short cuts are found near the bottom of the PLOT menu, the Show explanation and Hide explanation commands. Further short cuts for toggling on and off all explanations on the sheet are found in the DATA menu.

Rest Position

The right mouse button is used to move the explanation around on the sheet. Sometimes though rarely the explanation can get lost on the sheet or moved off the edge. When this happens, the 'Reset Position' button is handy to return the explanation to its default location near the middle right side of the plot.

Number of columns

The number of columns that the explanation will be drawn with is controlled by this entry field. This is a very nice feature to have in many circumstances when plots becomes choked with data.

Many commericial plotting packages do not support mulitple columned explanations. Tkg2 should also permit multiple independent explanations per plot, but such capability is unlikely to occur without substantial internal object reconstruction.

Column spacing

The spacing between two or more columns is controlled by this menubutton. The values available range from 0.1 to 1.0 inches. The Instructions language can give you arbitrary distances.

Explanation Title

Set the title of the explanation. Usually this is 'Explanation', which is the default, but some might want 'Legend' or nothing at all. The preferred style is to not title the explanation if it is shown within the confines of the plot and to title the explanation as 'Explanation' if it is shown outside the confines of the plot.

Title X-offset

Control the horizontal distance between the base of the explanation title and the remainder of the explanation components. The preferred units are in inches. For example, 0.2i for 0.2 inches. The string 'auto' will trigger internal logic to determine a visually pleasing spacing based on the font size.

Title Y-offset

Control the vertical distance between the base of the explanation title and the remainder of the explanation components. The preferred units are in inches. For example, 0.2i for 0.2 inches. The string 'auto' will trigger internal logic to determine a visually pleasing spacing based on the font size.

Vertical spacing

The vertical spacing or separation between successive lines of the explanation contents are controlled with this entry field. The line spacing of multiple lines of text beside a plotting symbol are not controlled by the vertical spacing—that line spacing is predicated purely on the font size. The vertical spacing controls the separation between what could be called 'explanation entries'. Like the X and Y offset, the value is in inches or the string 'auto' will trigger internal logical to determine a visually please spacing based on the font size that is larger than the line spacing of multiple lines of text.

Horizontal Gap

The horizontal gap is the separation between the right edge of the symbology and the left edge of the beginning line of text explaining what the symbol means. Again the value is in inches or the string 'auto' can be used (see discussion above).

Border color, width, style

The border color, line width, and style is controlled with these three menubuttons.

Background color

The background color of the explanation is controlled with this menubutton.

Font, Size, Weight, Slant, Color

The text font, size, weight, slade, and color is set by these four menubuttons and on entry field.

Show/Hide Explanation Entries

This button launches another dialog box in which individual columns of data plotted from a data file can be toggled on or off from the explanation or all (all of the columns) of the data plotted from z file can be toggled on or off. The actual drawing of the data in the plot is still drawn. Drawing of the data is controlled in the symbology configuration dialog better known as the DrawDataEditor. This is another neat feature of Tkg2. It can be extremely handy to hide specific entries in the explanation to make more room for really important entries. Although this mechansim is a little cumbersome, it does work. The NameIt buttons can be used to give a user name to each of the entries. This can be very important when using the Instructions language for tkg2 application building be developers.

FURTHER DISCUSSION

Developers interested in the Plot Editor should consult the following module:

```
Tkg2/Plot/Editors/PlotEditor.pm
```

THE CONTINUOUS AXIS EDITOR

This POD file provides detailed description of the Continuous Axis Editor dialog. This documentation file is located at Tkg2/Help/ContinuousAxisEditor.pod.

INTRODUCTION

The Continuous Axis Editor is the only means for configuring the settings specific to the axis on the five continuous as opposed to discrete axis types supported by Tkg2. The axis types are Linear, Logarithmic (base-10), Probability, Gumbel, and Time Series. There is a Discrete Axis Editor for configuration of discrete axis types, and is described inits own documentation file.

The editor is accessed either by double-left clicking on an axis or using the buttons at the bottom of the Plot Editor. The Plot Editor is accessed by double clicking within the borders of a plot.

LINEAR AXIS PARAMETERS tab

Axis Title

Specify the title of the axis in one or more lines. Current versions of Tkg2 do not support text rotation.

Reverse Axis

Reverse the sense of the axis. Instead of the minimum and the maximum being on the left (bottom) and right (top) of the plot, respectively, place the minimum and the maximum on the right (top) and left (bottom) of the plot, respectively.

Hide Numbers and Title

Hide the axis labeling (numbers) and the axis title.

Double Label

Label the axis on both sides of the plot.

Autoconfigure X,Y,Y2-Axis Limits:

By default Tkg2 does dynamic axis limit, labeling, and ticking determination based on the range of your data. Both the minimum and maximum are determined individually. For example, suppose you need a zero lower limit for you axis. This is a common occurrence in the physical sciences as many phenomena have a zero lower bounds. You would then set the lower axis limit or minimum to zero and toggle the Minimum checkbutton off. The maximum works in a similar fashion. If either the minimum or the maximum are dynamically determined then the labeling and the ticking potentially will change. There is no way to over ride this. The Tkg2 Instructions language does provide a way to tweak the axis during Tkg2 startup. The Center checkbutton is applicable on linear axis types only and the origin is placed in the middle of the plot. This is a handy feature for residual analysis from statistical regression procedures.

Minimum

Set the minimum axis limit. If the entry is blank, the value defaults to the last dynamically determined axis limit. If a plot is brand new, and no data has been read in, and the entry is blank, then an exception is thrown.

Maximum

Set the maximum axis limit. See Minimum.

Major Tick Interval

Set the interval between major tick marks in the same unit base as the axis.

No. of Minor Ticks

Set the number of minor ticks between major tick marks.

No. of Numbers to Skip

Set the interval jump between labeled major ticks. Sometimes it is desirable to have a lot of major tick marks, but the numeric text with each is so long as to cause the labels to run across each other. You can trim back the labeling but not the ticking by setting a positive number in place of the zero default for this entry.

Axis to percent basis

The axis is converted to a percentage basis. The type is converted to linear. The minimum is set to zero; the maximum is set to 100; the major tick interval is set to 10; and the no. of minor ticks is set to 4. The Apply or Ok buttons will have to be pressed for the effects to be seen, but the corresponding entries will change when this button is pressed. The PlotEditor has a button the changes all axes to fractional percent basis and immediately updated the screen.

Axis to fractional percent basis

The axis is converted to a factional percentage basis. The type is converted to linear. The minimum is set to zero; the maximum is set to 1; the major tick interval is set to .1; and the no. of minor ticks is set to 4. The Apply or Ok buttons will have to be pressed for the effects to be seen, but the corresponding entries will change when this button is pressed. The PlotEditor has a button the changes all axes to fractional percent basis and immediately updated the screen.

Label Transform Equation

The label transform equation is another one of the little features in Tkg2 that is really rarely used, but extremely nice to have under special circumstances. You can route the labeling through an arbitrary complex equation to produce alternative labeling. For example, you can convert a scale in meters to feet or other unit conversion. The equation uses one or more '$x' or '$X' to represent the original untransformed value. The '$x' is used even if the axis is either the first or the second Y axis. The following label transform equation converts the axis values to the equivalent circumference of a circle.

```
2*3.1415926*$x
```

Sometimes it is helpful to have one axis in the base units, but the opposite axis in another unit. Sort of like the nomographs of years past. A semicolon ';' can be used to separate a unique string and the label equation. When the semicolon is present, the transformation is only attempted on the opposite from usual axis (right or top). The following label transform equation on a Y axis converts the right axis values to equivalent circumferences of a circle, but the left axis remains in the radius. The string 'Circumference of Circle' is used as the right axis title. The axis title is optional, in which case the string after the ';' begins the label transform equation.

```
Circumference of Circle;2*3.1415926*$x
```

This feature is cool because it mitigates the problems with manuscript production in English and SI units and helps with international journal publication.

Special Major Ticks

Set by one or more space delimited, the specific locations in which special major ticks are to be drawn. This is a handy feature to set specific locations on the axis for descriptive purposes, varying units, or for manual editing in graphic editing software. For example, suppose you have an axis in units of days ranging from 0.5 to 2.5, yet you want to have specific ticks at certain hours of the first day near 6AM only, you can set the special major ticks as 1.25 1.125 1.375. Comma delimiting in just the dialog box can be used instead of spaces. However, the commas are replaced with a space as soon as the Appy or OK buttons are pressed. The Tkg2 Instructions language can only use one or more space delimiting.

Special Minor Ticks

Set by space delimiting, the specific locations in which special minor ticks are to be drawn. See Special Major Ticks.

LOG AXIS PARAMETERS tab

Many of the settings of the log axis are identical to the linear axis and are described above. This section only discusses features specific to the log axis.

Simple Log Scale

Toggles on a simple log scale like one might see in a spreadsheet graphic. This means that only the major tick of 1 or integer log cycles are labeled and no minor ticks are drawn. If the log cycle range is greater than or equal to seven, the major ticks drawn are at 1 and 5. If the range is greater than equal to 5 and less than 7, the major ticks are drawn at 1, 2, 4, 6, and 8. If the range is less than 5, the major ticks are drawn at 1, 2, 3, 4, 5, 6, 7, 8, and 9. The simple log scale settings are not preserved in the axis hash and hence shown in the dialog box upon clicking Apply or Ok. The settings are dynamically determined and set within the LogLabels.pm module.

Min w/offset, Max w/offset, Offset

The first two fields are non-editable fields and show the minimum and the maximum of the axis with the offset is applied. The actual offset is shown in the editable entry field with the `update`

`min/max` button immediately to the right. The button revises the min and max is offset using the value for the offset. By default the offset is zero. Essentially just the labeling and plotting location of the data change with the offset. The stacking of the tick marks for a log scale remain the same. As a example, suppose the minimum of an axis is .1, the maximum is 20. If the offset is 0.6, then the shown that is labeled minimum is 0.7 and the shown maximum is 20.6. The data will be plotting in the correct positions relative to what the labeling on the log scale indicates. The offset on a log scale can be really handy for particular manual operations on log paper. Log offsets are really popular with streamflow stage-discharge relations otherwise known as rating curves.

Base Major Ticks to DRAW

The major ticks to actually draw are specified here. Typically the major ticks are 1, 2, 3, 4, 5, 6, 7, 8, 9 in multiples of powers of 10. The 1-9 range is typical of most graphics packages and often constitutes the only scale resolution possible. Sometimes the ticks for 7, 8, and 9 get so scrunched up when the axis ranges over a few orders of magnitude that it is nice to turn ed 7 and 9 ticking off in favor of 8 alone. This would be accomplished by a base major ticks to draw of 1, 2, 3, 4, 5, 6, and 8. The list is space delimited. Arbitrary precision is possible. For example, 1.5 would place a major tick at .15, 1.5, 15, 150, and so on.

Base Major Ticks to LABEL

The major ticks to label are specified here in the same fashion as those to draw in the entry field above. It is a very nice feature to have the capability of specifying the ticks to label separately from the ticks to draw. The values for drawing and for labeling can even be mutually exclusive. Such a situation would arguably be a stupid arrangement. (See Base Major Ticks to DRAW).

Base Minor Ticks to DRAW

Just like the major ticks to draw, the minor ticks to draw can be individually specified. The premade minor ticks in the menubutton below this entry field is extremely handy as the lists can get quite long.

Premade Minor Ticks

This menubutton contains premade minor ticks to draw. (See Base Minor Ticks to DRAW).

PROBABILITY AXIS PARAMETERS tab

Many of the settings of the two probability axis are identical to the linear axis and are described above. A "Probability" axis is a normal probability axis that is symmetrical about the median and a normally distributed data set will plot as a straight line. The Gumbel probability axis is an asymmetrical axis about the median and is sometimes convenient in hydrologic frequency analysis.

This section only discusses features specific to the two probability axis. Tkg2 handles probability entirely in the traditional 0 to 1 sense. Exactly zero and exactly unity (1) probabilities are not available on the probability axis. You will have to resort to a linear axis if your application so dictates axis limits of 0 and 1. Tkg2 considers all probability values in a cumulative or nonexceedance perspective, including the data. The odd thing is that probability is labeled in percent on the axis as this is most common on probability paper that the author has seen and the easiest perspective to share with the less statistically inclined general public.

(1-Prob)

Toggle the axis from a nonexceedance perspective to an exceedance perspective. Although, Tkg2 considers nonexceedance internally, you can quickly produce the compliment of the data probability with this checkbutton. You will likely choose an axis title as "Exceedance Probability" instead of "Nonexceedance Probability".

RI style

Toggle on the RI (Recurrence Interval) style of probability, which means that equivalent recurrence intervals for the upper or right tail of the distribution are shown. The common recurrence intervals of 2, 5, 10, 25, 50, 100, 250, and 500 years are shown and can not be user configured. Although mathematically and certainly statistically it is preferable to use probability, recurrence intervals saturate the popular hydrologic literature. Because Tkg2 strives to produce graphics with great visual appeal, the RI style of probability axis shows probability and recurrence interval simultaneously. This promotes the linkage between recurrence interval and probability.

Major Ticks to DRAW in probability

Space delimited list of major ticks to draw. The tick location is specified in nonexceedance probability. (See Base Major Ticks to DRAW associated with log axis).

Major Ticks to LABEL in probability

Space delimited list of major ticks to label. The location is specified in nonexceedance probability. A tick is not drawn. (See Base Major Ticks to LABEL associated with log axis).

Minor Ticks to DRAW in probability

Space delimited list of major ticks to label. The location is specified in nonexceedance probability. (See Base Minor Ticks to DRAW associated with the log axis).

TIME SERIES AXIS PARAMETERS tab

Many of the settings of the time series axis are identical to the linear axis and are described above. This section only discusses features specific to the time series axis.

Several handy time calculators are available in the dialog box. First, Tkg2 uses an floating point number of days since January 1, 1900 for the drawing of the time series data and the axis. The 'Computed date-time representations' non-editable fields near the bottom of the dialog box show the axis minimum and maximum date time fields converted to the floating point days. Really handy for determine date-time offsets and a few other things. Make sure that you hit the Apply button as the calculator does not automatically update itself.

Another handy time calculator is the determination of whether a given year is a leap year. The execution button is originally titled "PRESS". After it is pressed the button will say Yes or No as whether the year to the left of the button is a leap year. Yet another handy time calculator is the determination of the number of days into a year (DOY) that a given date is. As with the leap year calculator, the execution button is originally title "PRESS". The DOY of the date to the left of the button is reflecting as the button title after pressing. This is a nice calculator to have around when your data loggers report DOY and not a date-time string in their output. Neither the leap year or DOY calculator is connected to any portion of the dialog box.

Show year

Toggle the display of the year on or off. For some types of plotting applications, it can be extremely useful to turn the year off. For example, if two years of data are plotted for a common time period, say to show between year variation with a fake year for one of the data, then it is inappropriate to show the year on the time axis.

Show day of week

Toggle the display of the day of the week string such as Saturday or Mon. Abbreviations are used as the number of displayed days increases. At some point, Tkg2 decides that it is not visually pleasing to show the the day of the week and the option is toggled off during the rendering although the dialog will continue to show it on. This behavior is considered a feature for batch processing operations.

Show day of year instead of date

The day of the year 1-365 or 1-366 is shown instead of the date.

Abbreviated months in pub. style (periods, June, July, Sept.)

A period is added to the month abbreviations and the really short months are fully spelled out. The default is to not show the period and to show various abbreviations as space permits. Toggling on the periods can be handy in automated publication situations.

Label Depth

Set the depth of labeling on the time axis. Tkg2 algorithms for time axis generation are extremely complex and can not satisfy all tastes, but the Label Depth can be used to get some variation labeling styles. Tkg2 will override the setting here based on a set of complicated rules in an attempt to garner visually pleasing spacing and resolution.

Label Density

Set the density of labeling and ticking on the time axis, see the Label Depth configuration. Tkg2 will override the setting here based on a set of complicated rules in an attempt to garner visually pleasing spacing and resolution.

SHORTCUTS ON DATE-TIME VALUE ENTRY

In the time axis editor, the date and time components of the axis minimum and maximum are controllable. Because date and time components are laborious to enter, numerous shortcuts have been

provided. Suggestions for more are always welcome.

Date Field (yyyy/mm/dd)

Date/time field means that the following shortcuts can be typed in either the date entry or the time entry. Whether the shortcut is specific for the beginning or ending is identified.

zero, null, or whitespace

January 1 of current year for the beginning field.

zero, null, or whitespace

December 31 of current year for the ending field.

now

Right now using Perl's scalar localtime function. Applicable in either the date or the time field.

then

One year ago from right now. Applicable in either the date or the time field.

yr-

One year (-365 days) ago from right now. Applicable in either the date or the time field.

yr+

One year (+365 days) from right now. Applicable in either the date or the time field.

yesterday

One day ago from right now and can be shortened to 'yes'. Applicable in either the date or the time field.

tomorrow

One day from right now and can be shortened to 'tom'. Applicable in either the date or the time field.

wk+

One week (+7 days) in the future from right now. Applicable in either the date or the time field.

wk-

One week (-7 days) in the past from right now. Applicable in either the date or the time field.

mn+

One month (+30 days) in the future from right now. Applicable in either the date or the time field.

mn-

One month (-30 days) in the past from right now. Applicable in either the date or the time field.

wyr

In the beginning fields provides October 1 of current water year. In the ending fields provides September 30 of current water year.

wyr#

In the beginning fields provides a number of days (#) past October 1 of current water year. In the ending fields provides a number of days past September 30 of current water year.

wyr-#

In the beginning fields provides a number of days (#) before October 1 of current water year. In the ending fields provides a number of days before September 30 of current water year.

Note that wyr can be shortened to wy in all the above wyr cases. To clarify the wyr shortcut a bit further. Assume that today's date is September 5, 2001. We will consider the beginning date field first. Typing wyr in either begin date or begin time will yield: Oct. 01, 2001; typing wyr-5 in either begin date or begin time will yield: Sept. 26, 2001; typing wyr5 in either begin date or begin time will yield: Oct. 06, 2001. For the ending date or begin time, wyr, wyr-5, and wyr5 yield: Sept. 30, 2001, Sept 25, 2001, and October 07, 2001, respectively.

Also, note that you get January 1 on the beginning date if you enter just a year and December 31 on the ending date if you enter just a year. You get the first of the month if you enter a year and a

month for the beginning field and you get the last day of the month in an ending date field—yes leap year supported.

Time Field (hh:mm:ss)

The time component entry space has the following shortcuts that are specific to it. Other short cuts are identified in the Date Field shortcut description.

zero, null, or whitespace

Sets time to 00:00:00.

number

Sets time to number of hours. For example, 11 provides 11:00:00 and 4 provides 04:00:00.

number1:number2

Sets the time to number of hours and number2 of minutes. For example 13:20 provides 13:20:00.

noon

Sets the time to 12:00:00.

tea

Sets the time to 15:00:00.

dinner

Sets the time to 18:00:00.

supper

Sets the time to 18:00:00.

bed

Sets the time to 22:00:00.

TITLE, LABELS, AND TICKS tab

Axis title font style

The axis title font, size, weight, slant, and color are configurable.

Vertically stack text of the title

This checkbutton places a newline behind each character for mitigate against the current absence of text rotation.

Axis labels (numbers) font style

The axis labels or numbers font, size, weight, slant, and color are configurable.

Vertically stack text of the labels

This checkbutton places a newline behind each character for mitigate against the current absence of text rotation.

Label minimum

The Label minimum checkbutton insures that the exact minimum of the axis is labeled. Some publication requirements, and justifiably so, dictate that the value for the axis end points be shown.

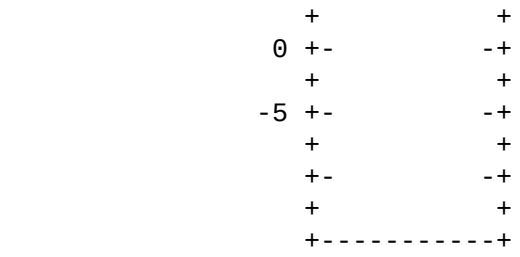
Label maximum

The Label maximum checkbutton insures that the exact maximum of the axis is labeled (see Label minimum).

Minimum to begin labels

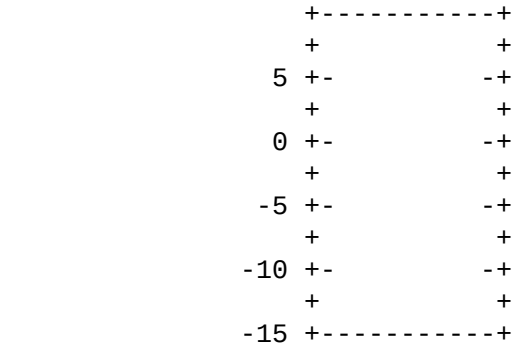
A minimum other than the actual minimum set on the Axis Parameters tab can be defined. The labeling will not begin at or near the actual minimum (see Label minimum), but it will begin at the value provided in the entry field. Some error trapping is provided and invalid values such as a value less than the actual minimum or greater than the actual maximum. If errors occur the value is set back to null. Setting a beginning label other than the actual is really handy in double-Y axis applications to show a plot within a plot—sometimes one has to have ridiculous limits for visual appeal, but the data never comes close to the actual minimum and maximum. (See Maximum to end labels. See Tick to actual minimum and maximum of the axis.) For example, if the minimum to begin labels is -5 then the following plot would be produced.

```
  10 +-----------+
     +           +
   5 +-          -+
```



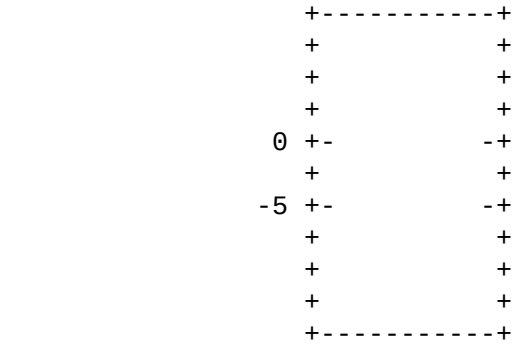
Maximum to end labels

Opposite sense of "Minimum to begin labels" (see above). For example, if the maximum to end labels is 5 then the following plot would be produced.

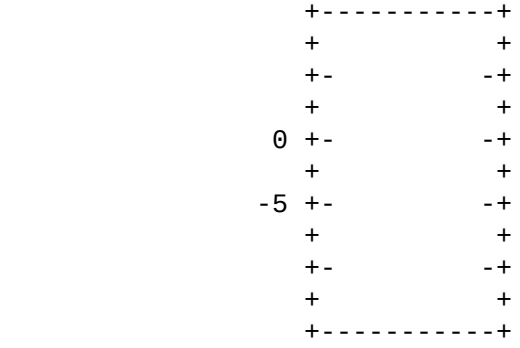


Tick to actual minimum and maximum of the axis

Since the limits of tick labeling can be controlled, it makes since that the ticking past the minimum to begin labels and maximum to end labelss is controllable as well. By default the ticks are drawn to the axis minimum and maximum. Consider a Y-axis and suppose the minimum to begin labels is -5 and the maximum to end labels is 1. If the ticking to axis minimum and maximum is turned off, the following plot is created.



If the ticking to axis minimum and maximum is turned on, the following plot is created.



Title and Label location

The location of the axis title is controlled by this menu button. For X axis, the choices are Top and Bottom; and for Y axis, the choices are Left and Right. This setting is overrided for the labeling by the Double Label checkbutton on the Axis Parameters tab, but the title still honors the location

setting. Double rendering of the axis title is not supported. A double Y axis plot nullifies the location setting.

Title Offset

Controls the vertical (X axis) or horizontal (Y axis) offset for the axis title from the axis itself in units of inches. Negative values are permissible.

Title2 Offset

Controls the horizontal (X axis) or vertical (Y axis) offset from the center of the axis in units of inches. A zero value is default and a nonzero value is seldom used, but can be handy in situations in which the label minimum and label maximum are considerably different from the actual minimum and maximum of the axis. This way you can have the axis title centered within the labeled bounds of the axis.

Label Offset

Controls the vertical (X axis) or horizontal (Y axis) offset for the axis title from the axis itself in units of inches. Negative values are permissible. The sense of the Label Offset is the same as the Title Offset described above.

Major Tick Length

Set the length in inches of the major tick marks. Even log cycle tick marks are made proportionally longer than the value given here for clarity. The tick mark length on a time axis has variable meaning.

Tick Width

Set the line thickness width of the tick marks. Both major and minor ticks have the same width.

Minor/Major

Set the ratio of the minor tick length to the major tick length. The default is 0.6.

Axis Format, Decimals

Set the format for the labels as free, fixed, scientific, or significant and set the number of decimal places to show. The significant format is experimental and does not quite support the concept of ‘significant figures"—the ‘g’ is used in the Perl sprintf function call. For scientific format or notation an ‘e’ is used and can not be changed to ‘x10’ or ‘E‘, but this is a feature that could quickly be added.

Commify Numbers

Add commas to the number portion of a label. For example, 1000 is changed to 1,000. Turning commas on and off in axis labeling is a feature not generally seen in other graphics packages.

GRID AND ORIGIN tab

Major Grid Lines:

Major grid lines can be toggled on and off, and their line thickness, color, and style (solid or various dashes) controlled.

Minor Grid Lines:

Minor grid lines can be toggled on and off, and their line thickness, color, and style (solid or various dashes) controlled.

Origin Line:

The origin line passes through zero on a linear axis and through .50 or 50 percent on a probability axis. The origin has not meaning with log or time axis. The origin is toggled on and off with the Doit checkbutton. The line width, color, and style (solid or various dashes) are each configurable. Having a quick way to draw the origin is an extremely handy feature that is commonly not seen in other graphics software.

FURTHER DISCUSSION

Developers interested in the Continuous Axis Editor should consult the following module:

```
Tkg2/Plot/Editors/ContinuousAxisEditor.pm
```

THE DISCRETE AXIS EDITOR

This POD file provides detailed description of the **Discrete Axis Editor** dialog. The plot editor is accessed either by selecting a plot and using the `Plot / Plot Editor` menu option or double clicking within the borders of a plot. This documentation file is located at

Tkg2/Help/DiscreteAxisEditor.pod.

INTRODUCTION

A discrete axis is created when the data incoming to the axis is a string category and not a number or time. The axis itself is built around the plotting of integers on the axis to determine spacing and other features. However, the axis is labeled with strings.

The editor is accessed either by double-left clicking on an axis or using the buttons at the bottom of the Plot Editor dialog. The Plot Editor is accessed by double clicking within the borders of a plot.

Because of its very nature, there are fewer parameters to configure for a discrete axis than there are for the Continuous Axis.

AXIS PARAMETERS

Axis Title

Specify the title of the axis in one or more lines. Current versions of tkg2 do not support text rotation.

Minimum

Set the minimum axis limit. If the entry is blank, the value defaults to the last dynamically determined axis limit. If a plot is brand new, and no data has been read in, and the entry is blank, then an exception is thrown.

Maximum

Set the maximum axis limit. See Minimum.

Reverse Axis

Reverse the sense of the axis. Instead of the minimum and the maximum being on the left (bottom) and right (top) of the plot, respectively, place the minimum and the maximum on the right (top) and left (bottom) of the plot, respectively.

Double Label

Label the axis on both sides of the plot.

Hide Numbers and Title

Hide the axis labeling (numbers) and the axis title.

Tick at group

The tick if placed at the 'origin' of each category or group on the axis when this checkbutton is toggled. Otherwise the tick is placed half way between successive categories. For example, the following '|' ticks are placed at the group. Each column of '*' represents a bar for a data point in the corresponding category.

```
  *          *
  **         *
  ***        **
  ***       ***
  *|*       *|*
 -------------
 CatA      CatB
```

Whereas, the following ticks are placed half way between the groups. This is also the default setting.

```
  *          *
  **         *
  ***        **
  ***       ***
  ***   |   ***
 -------------
 CatA      CatB
```

No. of Categories to Skip

Set the number category labels to skip drawing on the plot.

Stack or Cluster Data

Toggle between the two plotting mechanisms. If data is stacked in offsets from the center or origin of the category are not made. This is called clustering. The clustering is visually the easiest to view. The stacking is likely not quite a complete feature because the stacking is not additive—that is one can not generate bar plots in which the bar for each data value is placed on top of the bar for a previous data value. Perhaps an 'Additive Stack Data' radiobutton will be added for this feature in the future. The three bars for each category in the description of the Tick at Group checkbutton above represent a cluster. Here is a stacked example. Each symbol represents the a bar for from the bottom of the plot up to the data value.

```
          +
  +       +
  +       +
  *       +
  *       *
  %       *
  %       %
 _____________
 CatA    CatB
```

An additive stack data plot would look like the following using the same 'data' in the above figure.

```
  +
  +       +
  +       +
  +       +
  +       +
  +       +
  *       +
  *       +
  *       *
  *       *
  %       *
  %       %
 _____________
 CatA    CatB
```

Cluster Spacing

The spacing between catagorical elements in a clustered group is set by this entry field. The default is zero inches. This parameter has no effect unless the Cluster Data is toggled on. It will be hard to see the effects of this parameter unless the bars for a plot are toggled on. Normally this option would only be used for bar plots. A word of warning is needed. Because of internal design, it is not possible to build a smart cluster spacing setting, too large of cluster spacing can run the contents of one group onto and across another and give a false impression of the data contents.

TITLE AND LABELS

Axis title font style

The axis title font, size, weight, slant, and color are configurable as are the same parameters for the axis labels.

Vertically stack text of the title and labels.

This checkbutton places a newline behind each character for mitigate against the current absence of text rotation.

Do text blanking with with color

Set the color of the background of the text for the title and the labels.

Title Offset

Controls the vertical (X axis) or horizontal (Y axis) offset for the axis title from the axis itself in units of inches. Negative values are permissible.

Title2 Offset

Controls the horizontal (X axis) or vertical (Y axis) offset from the center of the axis in units of inches. A zero value is default and a nonzero value is seldom used, but can be handy in situations in which the label minimum and label maximum are considerably different from the actual minimum and maximum of the axis. This way you can have the axis title centered within the labeled bounds of the axis.

Label Offset

Controls the vertical (X axis) or horizontal (Y axis) offset for the axis title from the axis itself in units of inches. Negative values are permissible. The sense of the Label Offset is the same as the Title Offset described above.

Title and Label location

The location of the axis title is controlled by this menu button. For X axis, the choices are Top and Bottom; and for Y axis, the choices are Left and Right. This setting is overrided for the labeling by the Double Label checkbutton on the Axis Parameters tab, but the title still honors the location setting. Double rendering of the axis title is not supported. A double Y axis plot nullifies the location setting.

TICKS AND GRID

The tick marks on the discrete axis are referred to a major ticks to stay parallel with the internal implementation. There are no minor ticks for a discrete axis. The same situation holds for the grid lines as well.

Major Tick Length

Set the length in inches of the major tick marks.

Tick Width

Set the line thickness width of the tick marks.

Major Grid Lines:

Grid lines can be toggled on and off, and their line thickness, color, and style (solid or various dashes) controlled.

Further Discussion

Developers interested in the Discrete Axis Editor should consult the following module:

```
Tkg2/Plot/Editors/DiscreteAxisEditor.pm
```

THE DATA CLASS (FILE) EDITOR

This POD file provides detailed description of the **Data Class (File) Editor** dialog. The editor is accessed by selecting a plot and accessing the `Data / Edit Data or Do Statistics` menu option. This documentation file is located at **Tkg2/Help/DataClassEditor.pod**.

INTRODUCTION

A Data Class or File object is created each time a data file is read and the data actually plotted. The object is a simple container for one or more Data Set Objects. Each Data Set object contains the data to plot, settings for symbology, explanation entries, and ancillary components. The Data Class Editor is used to control the plotting order of the data files, remove data files from the plot, or even delete all files from a plot. Finally, the Data Set Editor can be launched. These features are provided by several buttons, which are outlined below.

BUTTONS

Raise

The selected data file in the list box is raised to the top of the list—its Data Sets will be drawn first (on the bottom of everything else in the plot).

Lower

The selected data file in the list box is lowered to the bottom of the list—its Data Sets will be drawn last (on the top of everything else in the plot).

Delete One

The selected data file in the list box is removed from the plot.

Edit Data Set (launch Data Set Editor)

The Data Set Editor is launched on selected data file (Data Class). See documentation on the Data Set Editor. From the Data Set Editor, then text in the explanation can be modified, the plot order of individual columns of data changed, the plotting symbology changed and even columns of data (single plots) removed.

Delete All

All data files in the listbox are immediately deleted from the plot—the plot will be empty.

OBJECT DESCRIPTION

The Data Class object is an array of Data Sets. The beginning of the object can be found in a Tkg2 file by searching for the `-dataclass` key. A snippet of a Tkg2 file is shown below.

```
'-dataclass' =>
      bless( [ bless( { ... }, 'Tkg2::DataMethods::DataSet' ),
               bless( { ... }, 'Tkg2::DataMethods::DataSet' )
             ], Tkg2::DataMethods::DataClass), ...
```

The example object is an array because of the [ ] and contains to subordinate Data Set objects. A plot as a single Data Class. The -dataclass is located within the `-plots` key of the Tkg2 file.

FURTHER DISCUSSION

Developers interested in the Data Class (File) Editor should consult the following module:

`Tkg2/DataMethods/Class/DataClassEditor.pm`

THE DATA SET EDITOR

This POD file provides detailed description of the **Data Set Editor** dialog. The editor is accessed by selecting a plot and accessing the Edit Data Set button from the Data Class (File) Editor. The Data Class (File) Editor is accessed using the `Data / Edit Data or Do Statistics` menu option. Another often more efficient access method is to double-left click on the explanation text (not the explanation title). This documentation file is located at **Tkg2/Help/DataSetEditor.pod**.

INTRODUCTION

A Data Set object is the primary structure for storage of data to plot, where and how to find the data or at least how the data was originally read in, explanation text, plotting order of individual columns of data, the plotting symbology. The Data Set Editor provides access into the object to change the text in the explanation, the plotting order, removal of objects, and a method to access the Draw Data Editor.

Edit Plot Style

The selected entry in either "Ordinates (Y-title)" linked list boxes is made available to the Draw Data Editor and the editor is freshly launched. The symbology can be changed.

Raise in Set

The selected entry in either "Ordinates (Y-title)" linked list boxes is raised to the top of the list—its Data Sets will be drawn first (on the bottom of everything else in the plot).

Lower in Set

The selected is raised to the top of the list—its Data Sets will be drawn is lowered to the bottom of the list—its Data Sets will be drawn last (on the top of everything else in the plot).

Delete from Set

The selected entry in either "Ordinates (Y-title)" linked list boxes is removed from the Data Set. Note that it is possible to delete all data associated with the data file identified in the title bar of the dialog box, but the object itself is not. The removal of the Data Set object is made in the Data

Class (File) Editor.

Modify Name

The selected entry in either "Ordinates (Y-title)" linked list boxes is made available for modification of the "New (show on plot)" text. This text is shown in the plot explanation. The button turns red and the text is placed into the "Modify the selected ordinate title" widget. The text can not be changed. New lines are acceptable. The text is committed by pressing the button again at which point it returns to the default foreground. Note that the "Original" entry is not modified. The "Original" entries are the column titles from the data file with `:type` appended. By making both entries (Original and the New) visible it is easier for the user to keep track of what data corresponds to what graphics. (The USGS-G2 program did not possess this feature, which occassionally made the program hard to use.)

Clear Field

The button clears or empties the "Modify the selected ordinate title" widget. This button is a feature to increase interface efficiency.

OK and Exit DataSet Editor

This button performs the usual acceptance of the changes if any and closes the dialog box.

OK and Exit DataClass Editor

This button provides the same function as the previous button; however, the Data Class (File) Editor is simultaneously closed if it is open. This button provides a pinch of efficiency by not requiring the user to perform a secondary action.

OBJECT DESCRIPTION

The Data Set object is a complex object. A full example is shown an the end of this section. Many of the key-value pairs in the object are self explanatory and regrettably some are not. The Data Set and Draw Data Editors provide tools to modify many of the non `-file` related or non `-data` fields. Note that in the example shown in this section, `-dataimported` is false thus the `-data` later on is empty. This is an example of a dynamic data loading Data Set—the author‘s preferred method. The contents of `-file` are controlled by the Add Data File to Plot dialog box (documented elsewhere). Users often find it useful to modify the `-relativefilename` or the `-fullfilename` (absolute path and file name) values when the data file name changes but the overall appearance of the plot remains the same. The default is for the `-relativefilename` to be used (see that `-userelativepath` is true). This means that the Tkg2 process will started from its directory of launch for the data files.

```
bless( {
       '-file' => {
         '-sorttype' => 'numeric',
         '-skiplineonmatch' => '^#',
         '-numlinestoread' => '',
         '-sortdir' => 'ascend',
         '-datetime_offset' => '0',
         '-numskiplines' => '0',
         '-relativefilename' => 'TestData/example_freq_curves.dat',
         '-fullfilename' =>
   '/home/wasquith/files/tkg2/TestData/example_freq_curves.dat',
         '-missingval' => '',
         '-thresholds' => '0',
         '-ordinates_as1_column' => '0',
         '-megacommand' => 0,
         '-sortdoit' => '0',
         '-numlabellines' => '1',
         '-filedelimiter' => '\\s+',
         '-transform_data' => {
           '-script' => '',
           '-doit' => 0,
```

```
      '-command_line_args' => ''
    },
    '-dataimported' => '0',
    '-common_datetime' => '',
    '-userelativepath' => '1',
    '-fileisRDB' => '0',
    '-numskiplines_afterlabel' => '0',
    '-columntypes' => 'auto'
  },
  '-username' => '',
  '-setname' => 'example_freq_curves.dat',
  '-show_in_explanation' => '1',
  '-DATA' => [
    {
      '-origthirdord' => undef,
      '-username' => '',
      '-showordinate' => 'Natural flood frequency curve',
      '-origabscissa' => 'Nonexceed_Prob:number',
      '-showthirdord' => undef,
      '-origordinate' => 'Nat_Quan(ft3/s):number',
      '-data' => [],
      '-attributes' => {
        '-xerrorbar' => {
          '-color' => 'black',
          '-dashstyle' => undef,
          '-whiskerwidth' => '0.05i',
          '-width' => '0.005i'
        },
        '-plotstyle' => 'X-Y Line',
        '-yerrorbar' => {
          '-color' => 'black',
          '-dashstyle' => undef,
          '-whiskerwidth' => '0.05i',
          '-width' => '0.005i'
        },
        '-points' => {
          '-blankit' => '0',
          '-num2skip' => '0',
          '-fillstyle' => undef,
          '-doit' => '1',
          '-outlinewidth' => '0.015i',
          '-fillcolor' => 'black',
          '-symbol' => 'Circle',
          '-outlinecolor' => 'black',
          '-angle' => '0',
          '-dashstyle' => undef,
          '-size' => '4.193',
          '-blankcolor' => 'white'
        },
        '-special_plot' => undef,
        '-shade' => {
          '-fillstyle' => undef,
          '-shadedirection' => 'below',
          '-doit' => '0',
          '-fillcolor' => 'black'
        },
        '-text' => {
          '-anchor' => 'center',
```

```
  '-numcommify' => 0,
  '-yoffset' => '3.353',
  '-doit' => 1,
  '-numdecimal' => 0,
  '-leaderline' => {
    '-blankit' => 0,
    '-color' => 'black',
    '-endoffset' => '5.84485393258427',
    '-doit' => 0,
    '-beginoffset' => '5.84485393258427',
    '-width' => '0.005i',
    '-blankcolor' => 'white',
    '-dashstyle' => undef,
    '-lines' => [
      {
        '-length' => '23.3794157303371',
        '-angle' => '225'
      },
      {
        '-length' => '14.6121348314607',
        '-angle' => '180'
      }
    ]
  },
  '-justify' => 'left',
  '-xoffset' => '1.196',
  '-numformat' => 'free',
  '-font' => {
    '-blankit' => 0,
    '-rotation' => 0,
    '-color' => 'black',
    '-family' => 'Helvetica',
    '-slant' => 'roman',
    '-custom1' => undef,
    '-weight' => 'normal',
    '-stackit' => 0,
    '-blankcolor' => 'white',
    '-custom2' => undef,
    '-size' => 8
  }
},
'-lines' => {
  '-linewidth' => '0.035i',
  '-doit' => '1',
  '-linecolor' => 'black',
  '-stepit' => '0',
  '-dashstyle' => undef
},
'-shades' => {
  '-fillcolor' => 'white'
},
'-bars' => {
  '-fillstyle' => undef,
  '-direction' => 'below',
  '-doit' => '0',
  '-outlinewidth' => '0.015i',
  '-barwidth' => '5.84485393258427',
  '-fillcolor' => 'white',
```

```
        '-outlinecolor' => 'black',
        '-dashstyle' => undef
      },
      '-which_y_axis' => '1'
    },
    '-showfourthord' => undef,
    '-showabscissa' => 'Nonexceed_Prob:number',
    '-show_in_explanation' => '1',
    '-origfourthord' => undef
  },
  {
    '-origthirdord' => undef,
    '-username' => '',
    '-showordinate' => 'Regulated flood frequency curve',
    '-origabscissa' => 'Nonexceed_Prob:number',
    '-showthirdord' => undef,
    '-origordinate' => 'Reg_Quan(ft3/s):number',
    '-data' => [],
    '-attributes' => {
      '-xerrorbar' => {
        '-color' => 'black',
        '-dashstyle' => undef,
        '-whiskerwidth' => '0.05i',
        '-width' => '0.005i'
      },
      '-plotstyle' => 'X-Y Line',
      '-yerrorbar' => {
        '-color' => 'black',
        '-dashstyle' => undef,
        '-whiskerwidth' => '0.05i',
        '-width' => '0.005i'
      },
      '-points' => {
        '-blankit' => '0',
        '-num2skip' => '0',
        '-fillstyle' => undef,
        '-doit' => '1',
        '-outlinewidth' => '0.015i',
        '-fillcolor' => 'white',
        '-symbol' => 'Square',
        '-outlinecolor' => 'black',
        '-angle' => '0',
        '-dashstyle' => undef,
        '-size' => '4.193',
        '-blankcolor' => 'white'
      },
      '-special_plot' => undef,
      '-shade' => {
        '-fillstyle' => undef,
        '-shadedirection' => 'below',
        '-doit' => '0',
        '-fillcolor' => 'grey85'
      },
      '-text' => {
        '-anchor' => 'center',
        '-numcommify' => 0,
        '-yoffset' => '3.353',
        '-doit' => 1,
```

```
  '-numdecimal' => 0,
  '-leaderline' => {
    '-blankit' => 0,
    '-color' => 'black',
    '-endoffset' => '5.84485393258427',
    '-doit' => 0,
    '-beginoffset' => '5.84485393258427',
    '-width' => '0.005i',
    '-blankcolor' => 'white',
    '-dashstyle' => undef,
    '-lines' => [
      {
        '-length' => '23.3794157303371',
        '-angle' => '225'
      },
      {
        '-length' => '14.6121348314607',
        '-angle' => '180'
      }
    ]
  },
  '-justify' => 'left',
  '-xoffset' => '1.196',
  '-numformat' => 'free',
  '-font' => {
    '-blankit' => 0,
    '-rotation' => 0,
    '-color' => 'black',
    '-family' => 'Helvetica',
    '-slant' => 'roman',
    '-custom1' => undef,
    '-weight' => 'normal',
    '-stackit' => 0,
    '-blankcolor' => 'white',
    '-custom2' => undef,
    '-size' => 8
  }
},
'-lines' => {
  '-linewidth' => '0.035i',
  '-doit' => '1',
  '-linecolor' => 'black',
  '-stepit' => '0',
  '-dashstyle' => undef
},
'-shades' => {
  '-fillcolor' => 'black'
},
'-bars' => {
  '-fillstyle' => undef,
  '-direction' => 'below',
  '-doit' => '0',
  '-outlinewidth' => '0.015i',
  '-barwidth' => '5.84485393258427',
  '-fillcolor' => 'black',
  '-outlinecolor' => 'black',
  '-dashstyle' => undef
},
```

```
                '-which_y_axis' => '1'
              },
              '-showfourthord' => undef,
              '-showabscissa' => 'Nonexceed_Prob:number',
              '-show_in_explanation' => '1',
              '-origfourthord' => undef
            }
          ]
        }, 'Tkg2::DataMethods::DataSet' ),
```

FURTHER DISCUSSION

Developers interested in the Data Set Editor should consult the following module:

```
Tkg2/DataMethods/Class/DataSetEditor.pm
```

THE DRAW DATA EDITOR

This POD file provides detailed description of the **Draw Data Editor** dialog. The editor is accessed by double-left clicking on the symbology for an entry in the plot explanation or accessed through the Data Set Editor by selecting a plot and accessing the `Data / Edit Data or Do Statistics` menu option and proceeding from there. This documentation file is located at **Tkg2/Help/DrawDataEditor.pod**.

INTRODUCTION

The instructions for how to draw the data on a plot are extensive. This editor provides the interface to adjust line widths, colors, and many other features.

POINTS tab

DoIt

Turn the point drawing on and off.

Type

Set the type of point symbol: Circle, Square, Triangle, Arrow, Phoenix, Thin Burst, Burst, FatBurst, Cross, Star, Horizontal Bar, and Vertical bar. (see `package Tkg2::Draw::DrawPointStuff`).

Size

The size in inches or other valid Tk units (inches, i) of the symbol. A scale factor can be applied (multiplied) to the size if the plot was originally loaded as a 'Text' plot and the third column (the text column) contains entries such as `pscale:1.1`, in which 1.1 will be multiplied on the size before the individual symbol is drawn. Thus, a half size symbol would have `pscale:0.5` as its corresponding text.

Edge

The line thickness of the outline of the symbol or the line segments comprising the symbol.

Angle

Set the angle in degrees of the symbol.

Edge Color

The color of the outline of the symbol or the line segments comprising the symbol.

Fill

The fill color of the symbol if applicable.

No. to skip drawing

For plots with a lot of points, it can be unacceptable to plot each symbol. If the value of this field is a non zero integer, then few symbols will be plotted. However line segments between the points are still plotted. For example, suppose that every other point should be plotted—set the no. to skip drawing to unity (1).

Do blanking below with color

Draw a colored rectangle below the symbol in the corresponding color.

The **RUG PLOT** capabilities of Tkg2 are now described. Rug plotting is the drawing of the data points not at the coordinate pairing (x,y), but drawing the x-values along the top/bottom of the x-axis or drawing the y-values along the left/right y-axis. The idea for rug plotting stems from WHA toying with the R environment for statistical computing in April 2005. These features are listed under the Points tab as the major point drawing logic is fundamentally the same. Finally, for historical purposes, rug plotting was added in May 2005 with the movement to the 1.+ versions of Tkg2—almost five years after Tkg2 was born. The rug plot features are loaded for backwards compatability when the DrawDataEditor is launched.

Rug X-axis DoIt
Turns on the rug plotting along the X-axis. This is a handy feature to show the general density of the data along the axis.

Both axes
The bottom axis is rugged by default. If this checkbutton is checked then the rug plot is produced on the both the bottom and top axes.

Invert the fibers
By default the rug plot is drawn on the axis and towards the center of the plot. If this checkbutton is checked then the rug plot is drawn on the axis and away from the plot interior.

Rug X-axis Color
Set the color that the rug plot is to be drawn in.

Edge
The line width of the rug plot.

Size
The size in inches of the rug plot elements.

Rug Y-axis DoIt
Turns on the rug plotting along the Y-axis. This is a handy feature to show the general density of the data along the axis.

Both axes
The left axis is rugged by default if the plot is for the first y-axis; the right axis is rugged by default if the plot is for the second y-axis. In either case, if this checkbutton is checked then the rug plot is produced on the both the left and right axes. It is important to remark that in each case the rugging is not aware of other rugs on the plot; therefore if you have a double-y plot and you rug each y-axis and have both checked then some really weird looking plots can be generated.

Invert the fibers
By default the rug plot is drawn on the axis and towards the center of the plot. If this checkbutton is checked then the rug plot is drawn on the axis and away from the plot interior.

Rug Y-axis Color
Set the color that the rug plot is to be drawn in.

Edge
The line width of the rug plot.

Size
The size in inches of the rug plot elements.

LINES tab

DoIt

Turn the line drawing on and off.

Width
: Set the line width.

Color
: Set the line color.

Style
: Set the line style (solid or dashed).

Step Type
: Set the stepping type. This is an unusual but potentially important feature. The default is no stepping in which an straight line is drawn between two data points exactly like all other graphing software. The step it option draws a stair case between each point—one horizontal line and then one vertical line. The over-step it option adds an additional horizontal having the same length as the previous horizontal line length. (Difficult to explain). The benefit of the stepping is that data that are averages can be properly represented. For example, suppose one has a time series of daily averages plotted at 00:00 hours. It is most appropriate to show a horizontal line across an entire day width instead of an angled line across the day as the angled line implies linear change in the data value. The over-stepping provides a horizontal bar on the last data value. The stepping is not confined to constant spaced data—a feature. The over-stepping is likely not appropriate with variably spaced data on the X-axis. (Stepping is harmless and does not change data values—try it out.)

Arrow distances
: The first distance is the distance along the line from the neck of the arrow to the tip. The second distance is the distance along the line from the trailing points of the arrow to the tip, and the third distance is the distance from the outside edge of the line to the trailing points. The distances are in inches.

Arrow style
: The arrow style: none, first, last, or both. Note that the arrowheads are potentially drawn at the ends of each ensemble of line segments. Thus, if all of the data having at least five data points is shown on the plot and the third data values are missing, then four arrowheads are drawn if the arrow style is `both`.

TEXT tab

Text annotation besides the data points requires that the plot type was originally set to "Text" in the Add Data File to Plot dialog box. If the text has the following special content `pscale:###`, in which ### is a number, then that number is multiplied on the symbol size (see the POINTS tab).

DoIt
: Toggle the text annotation on and off.

Anchor
: Set the anchor of the text string from is origin. The origin is specified by the location of the data point and the values for the offsets.

Font, Size, Weight, Slant, and Color
: Set the font family, size, weight, slant, and color.

X-offset and Y-offset
: Set the X and Y offsets for the text relative to the origin specified by the location of the data point. Any valid Tk unit (inches, i) can be used.

Do blanking with color
: Blank underneather the text in the specified color. Note that because of the font size conversion issues when postscript is rendered, this might not print or export properly.

Format and Decimals
: The format for numbers and the number of decimals to show. This is a very convenient feature when one wants to show numeric values on the plot without showing the entire numeric value for appearance.

TEXT tab, Leader Lines

DoIt

Toggle the leader lines annotation on and off. Turning leader lines on requires that the DoIt for the text annotation at the top of the dialog box is turned on.

Automatic overlap correction

Leader lines can easily over lap on another. Tkg2 has some quasi-smart routines to help minimize overlap. However, these routines are partially disabled as of the Tkg2 0.76+ release because of buggy behavior on some large applications dependent on Tkg2. (In general the overlap correction is experimental.)

ShuffleIt

Yet another toggle to help minimize line overlapping.

Flip every other line when shuffling

Yet another toggle to help minimize line overlapping.

Width and Color

Set the leader line width and color.

Begin and End offsets

Set the offsets between the end of the line and the data point and the end of the line and the text with these offsets in valid Tk units (inches, i).

Lines

Tkg2 supports multi-segmented leader lines. This is a potentially powerful feature in some circumstances. Each line is specified by a length and an angle. An example of the specification is provided by default in the text field.

SHADING tab

DoIt

Toggling the shading beside lines on and off.

Shade To Origin

Shade only as far as the origin if the direction is set towards the origin. The origin is taken as zero for a linear axis.

Direction

Set which direction below, above, left, right, or shade between to shade. The shade between requires that the original plot style from the Add Data File to Plot dialog was "Shade Between" or "Shade Between Accumulation". The ability to shade between and shade to the origin is considered a powerful interpretive feature of Tkg2.

Fill

Color to shade with.

BARS tab

DoIt

Toggle the bar drawing on and off.

Direction

Set which direction below, above, left, right, or horizontal bar between to draw the bar. The bar between feature appears broken (Tkg2 0.76+).

Bar width

Set the bar width in valid Tk units (inches, i).

Color

Set the border color of the bars.

Fill

Set the fill color of the bars.

ERROR LINES tab

Error line plots require that the "Y-Error Bar", "Y-Error Limits", "X-Y Error Bar", or "X-Y Error Limits" plot types be choosen in the Add Data File to Plot daillog to make this tab active. The tab has a section for the X-Error Lines if applicable and always has a Y-Error Line section. The settings for either are described here.

Width, Color, Style
: Set the line width (thickness), color, and style (solid or dashed)

Whisker width
: Specify the width in valid Tk units (inches, i) of the whiskers at the ends of the lines.

SPECIAL PLOT tab

See Special Plot section.

SPECIAL PLOTS

BOX PLOTS

Tkg2 box plots are very powerful, highly complex, but fully configurable. Several topics require discussion. It is hoped the the bulk of the settings on the dialog box are self explanatory or sufficiently similar to settings already discussed in this documentation file that detailing each on is not necessary. In order to plot box plots the box plot plotting type needs to be selected in the Add Data File to Plot dialog box when the data is read in. Tkg2 supports log transformation on the data prior to construction of the boxes as well as both product and L-moment computations. (MOST USERS WILL WANT PRODUCT MOMENTS.) Box plot drawing milage may vary when log-axes are used—be careful.

A major topic to remark on regarding the box plots is that the "center" or origin of the box is defined by the mean of the data which corresponds to the Points if the DoIt for point symbology (see above) is turned on. The `Location` tab defaults to the median of the box, but the location can be changed to the mean. When this is done the DoIt for the point symbology is not needed. All this means is that it is possible to for Tkg2 to plot the mean, median, or both in on the box. Another topic to briefly describe are the ‘ciles. This are the percenticles of the data—Tkg2 recognizes the tercile, quartile, pentacile, decile, and centacile of the data (if definable) according to USGS-WRD policy. The tercile is the 33rd percentile, the quartile is the 25th percentile, and so on.

Users should be aware of the different definitation of the `Tails`: range of the data, interquartile range (IQR), 1.5*IQR, or 3*IQR. The author perfers the range of the data to be the default. The statistics behind each box can be looked up in the `Statistics` tab. If you are really interested in the statistics, check out the —dumpboxes command line options of Tkg2. `Detection Limits` are so experimental that nothing works regarding detection limits. Some configuration on how the box is displayed in the explanation is provided in the `Show Data` tab.

OTHER SPECIAL PLOTS

Tkg2 does not currently support other fancy complex objects like box plots such as rose diagrams or stiff diagrams, but theoretically the Tkg2 data model and hooks deep in the code base could support other special plot objects (types).

OBJECT DESCRIPTION

See the Object Description section in the Data Set Editor documentation.

FURTHER DISCUSSION

Developers interested in the Draw Data Editor should consult the following module:

```
Tkg2/DataMethods/Class/DrawDataEditor.pm
```

FREQUENTLY ASKED QUESTIONS — FAQs

Are there any web resources for Tkg2?

```
http://tx.usgs.gov/usgs/tkg2
http://wwwnwis.er.usgs.gov/graphics/nwisgraphics.htm
http://wwwqvarsa.er.usgs.gov/unix/solaris/tkg2.html
```

I am having two problems with Tkg2 for some users. Tkg2 graphs appear briefly and then disappear when tkg2 is forked from another process, and Tkg2 graphs disappear when run by itself.

This is a known problem with Tkg2 and Reflection X (X-client software). As a short term solution, check the font order in the Client Manager - Setup - fonts and make sure 100dpi comes before 75dpi. This change seems to get things running.

My saved tkg2 files keep reconfiguring my custom settings whenever I open them—I don't want this to happen.

Tkg2 has a different approach to graphics handling. Tkg2 is geared for the batch processor, scriptor, and the person who want to avoid repetitive tasks. Tkg2 believes that it should be a markup engine on data files that are likely to change often. By providing the best behavior for change data files on default, the user is alerted to a different perspective of data plotting, research, and exploratory analysis. Hence, each axis is set for automatic configuration. This can be disabled by the following checkbuttons that are found on the axis editors (double left click on an axis) or in the plot editor (double left click inside a plot).

```
Autoconfigure *-Axis Limits: [] Minimum  [] Maximum  []Center?
```

As you can see, minimums and maximums can be treated separately. Often in environmental sciences, zero is the physical minimum for a plot, and one does not want tkg2 to configure an axis (linear only) with negative numbers. Setting a hard zero is done by setting the axis minimum to zero and toggling the minimum checkbutton off. Center is used to center the origin line on the axis during the autoconfiguration—a neat feature for residual plots common in regression analysis.

Can Tkg2 do multiple Y-axis plots with different data?

Yes. First create a plot and add some data to it. By default the data is placed on the first (left) Y axis". Next add more data to said plot, but click on the "Data for Second Y Axis" just below the plot type specification on the AddDataToPlot Dialog box. The incoming data is now placed on the second or right Y-axis. You can not add data to the second Y axis before data is added to the Y axis. Each axis, dubbed Y1 and Y2, are fully and independently configurable. You might be interested in the next FAQ.

Can Tkg2 draw a second Y-axis in order to show different units for the same data?

Yes. An example plot would be a time series of ground water elevations above sea level simultaneous with a depth below land surface axis. The data is the same, just a transformation to the other axis is needed.

First create a plot and add some data to it. By default the data is placed on the first (left) Y axis". Now double click on the Y-axis. Toggle the "Double Label" checkbutton on. Edit the "Label Transform Equation" entry field on the axis editor. Here is an example:

```
Different Label;5*$x+10
```

If a semicolon is present, the preceeding string is used as the second label. The equation following is the conversion or transformation of the label values on the left axis to label values on the right axis. As shown on the dialog box, $x (dollar sign x) is a surrogate for the tick values along the axis. Separate tick control on the second labeling is not currently available. This type of *independent* double labeling is only supported on linear and log axis plots. Transformation is also possible on the X axis too, but much less common.

I am using tkg2 across a slow network. Do I have other options?

Possibly. Remote display of graphics only is usually slow, but still acceptable. However, if tkg2 is then used in an interactive fashion to reconfigure objects, such as plots, it would be best if the user did a "Save As (with imported data)" from the file menu. Then the user sftp's the tkg2 file to their local machine. Tkg2 files are relative small without data, so transfer times are tremendously faster than graphics and not much slower than downloading data files from the remote machine. Then the user would launch tkg2 locally and open their files. For example,

```
%user@local: tkg2 ratplot_file_with_loaded_data.tkg2
```

Tkg2 can read pipes of data. Hence, networking of tkg2 processes is possible, not recently tested, not ever tested with secure shell, and not implemented by the graphics team at this time. Here is what the tkg2 help says (-withdraw is the key as the remote server does not pipe graphics across the network):

```
--display=[host]:server[.screen] or
--DISPLAY=[host]:server[.screen]
       The display that tkg2 is to operate under can be changed
       if and only if --display is the very first command line
       option.  Tkg2 requires some special internal handling
       of display before the other command line options can be
       processed.  Using --display allows variations on the
       following remote shell application:
          % tkg2 -stdout -withdraw -autoexit -importdata \
          tkg2file.tkg2 | rsh server.domain \
          "tkg2 -display=local:0.0 -stdin"
       This is a clear example of the shear power provided by
       simple redirection of STDOUT and STDIN channels, and
       why all good software should have both graphical and
       commandline interfaces.
```

The font size in dialog boxes is too small or too large to read on my screen. Can I change this?

Yes. You will need to create or modify your .tkg2rc file in your home directory or have your system administrator modify tkg2‘s system level tkg2rc file if numerous users have problems seeing the fonts. The default font size is oddly dependent on the X-server/client relationship. The details are vague at best, but the fix is to increment the font size up or down as needed. This is done with the

```
Tkg2*increment_dialog_fonts:  number
```

resource in the tkg2rc files. For a my laptops, set number to 3, which gives me very large and easily readable letters and numbers. The underscores are mandatory. The capital T in Tkg2 is mandatory too.

I can‘t read the dialog boxes, white text on white backgrounds and I am using Solaris with the CDE desktop.

In CDE on the Sun, if you select a palette in the style manager that has a dark xterm/window background, the text automatically changes to white. Then any software with light backgrounds become almost impossible to read. This works for Framemaker, apparently TKG2, and another I can‘t remember. The only fix we found was to change the window background to a lighter color so the text stays black.

Tkg2 crashes Reflection X.

Suggest removing the 100dpi and 75dpi from the fonts for reflection settings. If you do this and are a user of AIS, the AIS menu‘s are very ugly. What we found worked was to move the 75dpi to later in the list (100dpi we did remove) and that things worked fine.

Can Tkg2 left the pen on line plots? I have missing data, but this fact is not reflected in the data files.

Tkg2 is not smart enough to test of incremental stepping of your data. This includes time series too. Tkg2 will ‘lift’ the pen if and only if one or more missing values are encountered in the data file. Tkg2 has no mechanism to ask itself, if the data is say daily values and there is a break of two or more days. We are inserting missing fields for our groundwater level plots in Texas for the data reports. Here is example

```
DATE  Q
10/01/1969   45
10/02/1969   50
10/03/1969   --
10/04/1969   55
```

The pen is thus lifted on Oct 3rd if the missing value string is set to '—'. You might have to preprocess your data to find the gap around Oct 3rd. Tkg2 implementation of the pen lifting in this fashion provides logical extension to non-time series data.

Could I have a tutorial on batch processing tkg2?

Batch processing with tkg2 is really easy once you do it the first time, and incredibly complex or advanced scripting can be done. Tkg2 is suitable for use as a graphics engine on top of data bases—as long as data base produces nice ASCII files. The easiest model to to create your 'template' on temporary or junk file names of your data. Search your example tkg2 file for '-relativefilename'. If that name points to a temporary file, then you can script something like this.

1) You want three plots. You have three files for station X.
sta08177600_daily.rdb
sta08177600_mean.rdb
sta08177600_rain.rdb

2) You want to create generic tkg2 file to plot this type of data.

3) copy these files to some temporary names
copy("sta08177600_daily.rdb", "TMP_daily.rdb"); # daily mean streamflow (daily values)
copy("sta08177600_mean.rdb", "TMP_mean.rdb"); # mean daily streamflow
copy("sta08177600_rain.rdb", "TMP_rain.rdb"); # daily rainfall

4) Create your tkg2 plot(s) using the TMP* files. Call the created file: joes_plots.tkg2

5) Make sure that you are using relative paths and dynamic loading of data (these are defaults so you don't have to do anything).

6) Write a wrapping script.
% touch joes_plots.pl # create the file
% chmod 755 joes_plots.pl # make it executable
% nedit joes_plots.pl # nedit is my favorite editor

In joes_plots.pl type this

```
#!/usr/bin/perl -w
use strict;
use File::Copy;
die "Please provide station number on command line\n" unless(@ARGV);
my $station = shift(@ARGV);
my $daily = "sta".$station."_daily.rdb";
my $mean  = "sta".$station."_mean.rdb";
my $rain  = "sta".$station."_rain.rdb";
print "Using these files: $daily\n $mean\n  $rain\n";
my $tmpdaily = "TMP_daily.rdb";
my $tmpmean = "TMP_mean.rdb";
my $tmprain = "TMP_rain.rdb";

copy($daily, $tmpdaily);
copy($mean, $tmpmean);
copy($rain, $tmprain);

system("tkg2 --justdisplayone joes_plots.tkg2");

unlink($tmpdaily, $tmpmean, $tmprain);  # delete the tmp files
exit;
#EOF
```

Ok that should do it. Now you need to figure out how to get your data in to the sta########_*.rdb name. Your users do this, then type joes_plots.pl 08177600 and off you go.

PDF looks bad.

Tkg2 uses GhostScript to convert the Postscript to PDF and does not have its own PDF rendering engine. The most usual problem is that some fonts get bitmapped instead of rendered. At other times all text lines with a hyphen will get bitmapped and not rendered. This is not a Tkg2 bug, but appears to be a bug in GhostScript? Your author does not know. This appears to be a bug in GhostScript postscript to PDF generation because Postscript to printers looks good every time—Fonts are never bitmapped. As a fix, could you possibly important the postscript to Illustrator and save as PDF there?

Text rotation is possible when using Framemaker.

By default tkg2 will run a /usr/local/Tkg2/Util/tkmiffix.pl script on the raw outputted mif. The mif is generated from the pstoedit command from the postscript that tkg2 produces. Tkg2 only produces postscript—conversion provided by external utilities. If you are heading to Frame you can hack the rotation by adding a string like the following to ANY line of text including the Y-axis label.

```
<Angle 90>, <ang90>, <Ang90>,
```

So on the screen your Y-axis title might look like this.

```
Y-DATA<Ang90>
```

Note, have to have the text stacking turned off.
Now if you export this to mif, the text will be rotated 90 degrees counterclockwise. The caveat is that each line of text requires rotation.

```
FLOW,<Ang90>
IN CFS<Ang90>
```

Yet another problem is that the rotation occurs on the upper left hand corner of the text and not in the center. I can not get around this at the present time. You will have to reposition the text as necessary to get the rotation to come out right—experimentation by you is required. Also manual editing from Frame might be needed, but that is not too hard to time consuming. You might prefer to delete the title for the Y-axis in favor of using Text Annotation as you have greater flexibility in position control.

NWIS/ADAPS FAQs

Shift Analysis Plotting

While generating a shift analysis and shift-bar plot and graph for station "05514840 Dardenne Creek at O‘Fallon", the following message appeared in the terminal window:

Bareword found where operator expected at (eval 208) line 28, near "‘USGS 05514840 Dardenne Creek at O‘Fallon" (Missing operator before Fallon?) Bareword "Fallon" not allowed while "strict subs" in use at (eval 208) line STARTING " (Missing operator before STARTING?) String found where operator expected at (eval 208) line 29, near ") ‘’ , ,, (Might be a runaway multi-line ’ ’ string starting on line 28) (Missing operator before ‘, ‘?)

SOLUTION:

The single quote (’) in the station name was causing unbalanced quotes. Remove the single quote (’) from station name O‘Fallon in the header file (using GWSI).

More notes about Reflection Fonts, Tkg2, and ADAPS

From: Shawn C Noble <scnoble@usgs.gov> Subject: Reflection Fonts, Tkg2, ADAPS

This is an FYI on a problem that the Iowa District has been having with segmentation faults when opening Tkg2 plots. We noticed it creating hydrograph and rating plots via ADAPS, though it‘s not really an ADAPS issue, it was a reflection font settings issue. It ended up we needed to have the "hp" font set included in the reflection "Subdirectories and font servers" section. We had removed this in the past thinking it was not needed.

Tkg2 not working with Reflection X on XDMCP connection—Tkg2 freezes.

This problem is found when using XDMCP (in some, not all cases).
So the workaround is to use OPENSSH connection method (not XDMCP) in Reflection X.
This presents another problem in that, for some, this solution does not give the user his Solaris CDE that s/he‘s become used to.

Tkg2 collapsing characters of a font (metrics of the boxes around the characters seems to be screwed up).

Inspection of the postscript file shows that a font substitution is being made. In the problem discussed below, the Courier font was being substituted for Fixed.

SOLUTION: Date: Wed, 10 Dec 2008 09:03:00 -0600 From: Shawn C Noble <scnoble@usgs.gov> To: William Asquith <wasquith@usgs.gov> Subject: Re: Text Problem with Tkg2 file

William,

I should have checked my FAQs also. I found the following in my notes that looks to have fixed our problem.

Shawn

Ran into a problem with the graphs produced by swreview. Specifically it appeared as though some fonts were incorrect.

The fix was in Reflection X, settings-fonts. Needed to deselect "Allow font scaling" and "Try font server on client host".

It worked after that.