6 Examples of Use

Version 06.10.06, 2005-08-05, Color, Acrobat Distiller

This chapter provides examples for retrieving and manipulating time series data using TSTool. The heading for the section gives an indication of the example purpose. Where appropriate, input and output types are indicated to simplify finding a useful example. General examples are listed first, followed by more complex examples.

6.1 General Examples

This section includes examples related to general TSTool use, which may be appropriate for general users.

6.1.1 General – One-time Time Series Display/Analysis

The following example session illustrates how to query time series data for display, analysis, and viewing.

1. Start TSTool. If the HydroBase or other database input types are enabled, select a database (see **Section 3.2- Select HydroBase Dialog,** for example).
2. To manipulate time series in any way, first select the time series of interest (see **Section 3.3 - Main Interface**). Pick appropriate input type, data type, and filter information. Press ***Get Time Series List*** to list the available time series. After pressing ***Get Time Series List***, a list of time series will be shown in the upper-right corner of the interface.
3. Select one or more time series from the list and transfer to the ***Commands*** list as time series identifiers. Time series identifiers are explained in **Chapter 2 – Introduction**.
4. Press the ***Run All Commands*** button to query the time series. They should now be listed in the ***Time Series Results*** list.

5. Use the ***Results*** and ***Tools*** menus to view the time series or the ***File…Save…Time Series As*** menus to export as files. For example, display a line graph (using ***Results…Graph - Line***) and then view the time series as a summary or table.
6. Go back to the ***Commands*** list and use the ***Commands*** menu to manipulate time series. For example:

   - Insert a `fillInterpolate()` command to fill data

   - Insert a `cumulate()` or `runningAverage()` command to transform the data

7. Repeat steps 4 – 5 to process and view time series.

6.1.2 General – Reproducing an Analysis with a Commands File

To reproduce an analysis, save the commands shown in the ***Commands*** list to a commands file and then reload and run the commands later. For example, assuming that steps similar to the previous section have been executed:

1. Use the ***File…Save…Commands As*** menu to save the commands. It is recommended that commands files be saved with a file extension *.TSTool*.
2. Exit TSTool and restart (alternatively, clear the commands using the ***Clear Commands*** button).
3. Use ***File…Open…Commands File***. Select the file that you previously saved.
4. Then run the commands by pressing the ***Run All Commands*** button. Display the results using the ***Results*** menu.

As the above example shows, reproducing an analysis consists of saving a commands file that can be reused later. The main complications in this approach are that the environment in which the commands are run may change over time. For example if using a HydroBase database, the database version, ODBC data source name, database host, or working directory may be differ between computers. It is recommended that commands use directories relative to a working directory and that the working directory is defined consistently on different computers that will use the commands. Using paths relative to the working directory will consequently allow commands files to be portable. TSTool will internally set the working directory that the directory where a commands file is opened or saved.

6.2 Model Data Processing Examples

Most computer models require data that adhere to a consistent format. TSTool facilitates processing model data files with features that:

- Allow a specific period of record to be output
- Fill missing data
- Produce time series in a specific order

The following examples illustrate how to use TSTool to process model data.

6.2.1 Modeling – Preparing Model Files Using a Commands File

To prepare model files, multiple commands will usually process numerous time series. Modelers often run TSTool in batch mode from a command shell using a command like:

```
tstool -commands commandsfile
```

However, it is recommended that commands files be run using the GUI, if possible, in order to take advantage of additional error-checking and feedback features.

TSTool provides command editor dialogs for every command and helps ensure the integrity of commands by searching for input time series for each command. An effort has been made to make the current TSTool recognize and process old commands. However, there have been some changes that will require updates to commands. It is recommended that old command files be migrated to the new syntax using the following approaches:

1. Be familiar with this documentation and, in particular, the command reference.
2. Run an existing commands file and review the log file for warnings about commands that need to be updated. Then edit the commands in the GUI (see the next step).
3. Open an existing commands file using the ***File…Open…Commands File*** menu and then edit commands using the editor dialogs. These dialogs will help to convert old commands to the new syntax. Most commands files focus on a particular data type and filling technique. Therefore most updates will generally involve only a few changes.

A number of commands have been added/enhanced to promote reuse of commands files in both batch and GUI run modes. For example, the `openHydroBase()` indicates whether the command is active for batch and GUI run modes. Choosing the correct setting simplifies exchange of command files between users and operating environments.

When querying time series, select a subset of the commands for intermediate work to verify filling or other manipulation. General commands (e.g., `setOutputPeriod()`) may be required even if a subset of time series is being processed.

TSTool by default reads all available data. However, the `setQueryPeriod()` is available to limit the period that is read. The `setOutputPeriod()` is now used only to control the period for output products.

6.2.2 Modeling – Processing End of Month Reservoir Data (Input=HydroBase, Output=StateMod)

The commands file shown below was originally generated by an older version of TSTool (before TSTool 05.xx.xx).

```
10/1974 9/1991
-units DFLT
-wy
-ostatemod
-o coloup.tar
# CBT SHADOW MTN GRAND L
Fillconst(513695.SEO.RSTO.MONTH.,0)
513695.SEO.RSTO.MONTH.
# CBT GRANBY RESERVOIR
Fillconst(514620.SEO.RSTO.MONTH.,0)
514620.SEO.RSTO.MONTH.
# DILLON RESERVOIR
Fillconst(364512.SEO.RSTO.MONTH.,0)
364512.SEO.RSTO.MONTH.
# GREEN MOUNTAIN RESERVIOR
Fillconst(363543.SEO.RSTO.MONTH.,0)
363543.SEO.RSTO.MONTH.
# RIFLE GAP RESERVOIR
Fillconst(393508.SEO.RSTO.MONTH.,0)
393508.SEO.RSTO.MONTH.
```

The equivalent commands using new commands syntax are shown below. Note that only a few changes are required to update the file. Although the resulting commands file may be longer in some cases, the fill operations are now separate from the initial query, allowing multiple fill commands to be applied sequentially and also allowing the intermediate time series to be queried and viewed independently in the GUI. The new commands structure also allows the same results to be obtained in multiple ways. For example, the zero-filled time series can be created by using `newTimeSeries()`, or by utilizing `setIncludeMissingTS()`, `fillConstant()`, and `setConstant()`. **Note that since the original commands file was created, greater definition has been given to data sources in time series identifiers. The TSTool GUI will always supply a data source from the HydroBase database and may need to be used to determine the proper data source for a time series identifier.**

```
# Reservoir target file commands
# Each reservoir needs a minimum (zero) and maximum time series (from HydroBase)
setOutputPeriod(10/1974,9/1991)
setOutputYearType(Water)
# CBT SHADOW MTN GRAND L
TS ShadowMtn = newTimeSeries(513695.USBR.ResEOM.MONTH.,"CBT SHADOW MTN GRAND L",*,*,AF,0.0)
513695.USBR.ResEOM.MONTH.
# CBT GRANBY RESERVOIR
TS Granby = newTimeSeries(514620.USBR.ResEOM.MONTH.,"CBT GRANBY RESERVOIR",*,*,AF,0.0)
514620.USBR.ResEOM.MONTH.
# DILLON RESERVOIR
TS Dillon = newTimeSeries(364512.DWB.ResEOM.MONTH.,"DILLON RESERVOIR",*,*,AF,0.0)
364512.DWB.ResEOM.MONTH.
# GREEN MOUNTAIN RESERVIOR
TS GreenMtn = newTimeSeries(363543.USBR.ResEOM.MONTH.,"GREEN MOUNTAIN RESERVOIR",*,*,AF,0.0)
363543.USBR.ResEOM.MONTH.
# RIFLE GAP RESERVOIR
TS RifleGap = newTimeSeries(393508.USBR.ResEOM.MONTH.,"RIFLE GAP RESERVOIR",*,*,AF,0.0)
393508.USBR.ResEOM.MONTH.
writeStateMod("coloup.tar",*)
```

6.2.3 Modeling – Filling Reservoir Targets with a Pattern File (Input=HydroBase, Output=StateMod)

The following example illustrates an old command file (before TSTool 05.xx.xx) for creating a StateMod reservoir target file, using pattern filling.

```
# Phase IIIb modifications
#       Fill missing data using water district indicator gages determined in demandts runs
#       Fill with historical monthly average if no wetness pattern average available
#       Set start dates for reservoirs in March of year listed in Ray A fax (9/8/98)
#
10/1908 9/1998
-ignorelezero
-units DFLT
-wy
-filldata fill.pat
-fillhistave
-ostatemod
-o ../ym2001H.tar
# ELKHEAD RESERVOIR
443902.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern_setconstbefore(443902.StateMod.RSTO.MONTH.../ts_files/443902.stm,9251000,0,03/1975)
# ALLEN BASIN RESERVOIR
583500.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern_setconstbefore(583500.StateMod.RSTO.MONTH.../ts_files/583500.stm,9239500,0,03/1909)
# FISH CREEK RESERVOIR
583508.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern_setconstbefore(583508.StateMod.RSTO.MONTH.../ts_files/583508.stm,9239500,0,03/1956)
# LESTER CREEK RESERVOIR
583521.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern_setconstbefore(583521.StateMod.RSTO.MONTH.../ts_files/583521.stm,9239500,0,03/1975)
# STILLWATER RESERVOIR 1
583540.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern_setconstbefore(583540.StateMod.RSTO.MONTH.../ts_files/583540.stm,9239500,0,03/1939)
# LAKE CATAMOUNT
583631.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern_setconstbefore(583631.StateMod.RSTO.MONTH.../ts_files/583631.stm,9239500,0,03/1974)
# STEAMBOAT LAKE
583787.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern(583787.StateMod.RSTO.MONTH.../ts_files/583787.stm,9239500)
#fillpattern_setconstbefore(583787.StateMod.RSTO.MONTH.../ts_files/583787.stm,9239500,0,03/1974
)
# STAGECOACH RESERVOIR
584213.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern_setconstbefore(584213.StateMod.RSTO.MONTH.../ts_files/584213.stm,9239500,0,03/1988)
# YAMCOLO RESERVOIR
584240.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
fillpattern_setconstbefore(584240.StateMod.RSTO.MONTH.../ts_files/584240.stm,9239500,0,03/1980)
…some commands omitted…
```

The same results are obtained using new commands with the following commands file.

```
# Phase IIIb modifications
#       Fill missing data using water district indicator gages determined in demandts runs
#       Fill with historical monthly average if no wetness pattern average available
#       Set start dates for reservoirs in March of year listed in Ray A fax (9/8/98)
#
#
# Updated by SAM (2001-04-02) during TSTool testing.  Made the following
# changes:
#
# Read time series then use fillPattern() (2 steps rather than 1)
# Comment out -o options and replace with writeStateMod() at end
# Change -fillhistave to fillHistMonthAverage(*) at end
# Change -ignorelezero to setIgnoreLEZero(true)
# remove -units
# replace -filldata with setPatternFile()
# change -wy to setOutputYearType(Water)
# change period to setOutputPeriod()
# Add fillConstant(*,0.0) at end
#
setOutputPeriod(10/1908,9/1998)
setIgnoreLEZero(true)
setOutputYearType(Water)
setPatternFile(fill.pat)
# ELKHEAD RESERVOIR
443902.DelimFile.ResEOM.MONTH.../ts_files/zero_dat.del
TS 443902 = readTimeSeries(443902.StateMod.ResEOM.MONTH.../ts_files/443902.stm)
fillPattern(443902,9251000)
setConstantBefore(443902,0,03/1975)
# ALLEN BASIN RESERVOIR
583500.DelimFile.ResEOM.MONTH.../ts_files/zero_dat.del
TS 583500 = readTimeSeries(583500.StateMod.ResEOM.MONTH.../ts_files/583500.stm)
fillPattern(583500,9239500)
setConstantBefore(583500,0,03/1909)
# FISH CREEK RESERVOIR
583508.DelimFile.ResEOM.MONTH.../ts_files/zero_dat.del
TS 583508 = readTimeSeries(583508.StateMod.ResEOM.MONTH.../ts_files/583508.stm)
fillPattern(583508,9239500)
setConstantBefore(583508,0,03/1956)
# LESTER CREEK RESERVOIR
583521.DelimFile.ResEOM.MONTH.../ts_files/zero_dat.del
TS 583521 = readTimeSeries(583521.StateMod.ResEOM.MONTH.../ts_files/583521.stm)
fillPattern(583521,9239500)
setConstantBefore(583521,0,03/1975)
# STILLWATER RESERVOIR 1
583540.DelimFile.ResEOM.MONTH.../ts_files/zero_dat.del
TS 583540 = readTimeSeries(583540.StateMod.ResEOM.MONTH.../ts_files/583540.stm)
fillPattern(583540,9239500)
setConstantBefore(583540,0,03/1939)
# LAKE CATAMOUNT
583631.DelimFile.ResEOM.MONTH.../ts_files/zero_dat.del
TS 583631 = readTimeSeries(583631.StateMod.ResEOM.MONTH.../ts_files/583631.stm)
fillPattern(583631,9239500)
setConstantBefore(583631,0,03/1974)
# STEAMBOAT LAKE
583787.DelimFile.ResEOM.MONTH.../ts_files/zero_dat.del
TS 583787 = readTimeSeries(583787.StateMod.ResEOM.MONTH.../ts_files/583787.stm)
fillPattern(583787,9239500)
#fillpattern_setconstbefore(583787.StateMod.ResEOM.MONTH.../ts_files/583787.stm,9239500,0,03/1
974)
# STAGECOACH RESERVOIR
584213.DelimFile.RSTO.MONTH.../ts_files/zero_dat.del
TS 584213 = readTimeSeries(584213.StateMod.ResEOM.MONTH.../ts_files/584213.stm)
fillPattern(584213,9239500)
setConstantBefore(584213,0,03/1988)
# YAMCOLO RESERVOIR
584240.DelimFile.ResEOM.MONTH.../ts_files/zero_dat.del
TS 584240 = readTimeSeries(584240.StateMod.ResEOM.MONTH.../ts_files/584240.stm)
fillPattern(584240,9239500)
```

```
setConstantBefore(584240,0,03/1980)
…some commands omitted…
fillHistMonthAverage(*)
fillConstant(*,0)
writeStateMod(../ym2001H.tar,*)
```

6.2.4 Modeling – Using a List File to Automate Time Series Processing (Input=HydroBase, Output=StateMod)

It may be desirable to output a StateMod daily time series file given a list of station/structure identifiers. The following example illustrates the commands file to accomplish this task.

```
#
# Example to illustrate how a list file can be used to generate a StateMod daily
# time series file with for many structures
#
# Read a list file, where the first column is structure identifiers.  Try to get
# daily diversion time series for each structure.  If data are not available, create
# an empty missing time series.
#
createFromList(ListFile="structure_list.txt",DataSource=DWR,DataType=DivTotal,
Interval=Day,InputType=HydroBase,HandleMissingTSHow=DefaultMissingTS)
#
# Now write the results to a StateMod file.
#
setOutputYearType(Calendar)
writeStateMod("structure_list.stm",*)
```

6.2.5 Modeling – Processing Frost Dates (Input=HydroBase, Output=StateCU)

Frost dates are special time series consisting of four dates per year. The dates correspond to:

- Last day in spring that the temperature was 28° F
- Last day in spring that the temperature was 32° F
- First day in fall that the temperature was 32° F
- First day in fall that the temperature was 28° F

These specific dates are currently consistent with the HydroBase and StateCU input types. Older versions of TSTool (before version 06.00.00) treated frost dates as a single time series, where the four components were internally manipulated as dates. The `add()` command and a limited number of features supported manipulating frost date time series. However, other commands could not be used to process the time series. Consequently, display, analysis, and manipulation capabilities were limited.

As of TSTool version 06.00.00, TSTool handles each of the above data items as separate data types and time series, internally treating the values as Julian days from January 1. The StateCU input type, which is used when reading and writing frost date files, converts between Julian days and Month/Year in the file. By handling as four separate numerical time series, all of TSTool's manipulation tools can be used to fill and analyze frost dates. This does require each time series to be specified, whereas before the four were internally handled with a single time series identifier. Because frost dates are internally treated as numerical Julian days, using the generic numerical `add()` command functionality may result in unexpected output if the time series overlap (Julian days will be added). To avoid this situation, use the `fillFromTS()`, `setFromTS()`, or `blend()` commands when merging multiple time series. The following example illustrates how to process a frost dates file for StateCU:

```
setOutputPeriod(1950,2002)
# __________________________________________________________
# 0130 - ALAMOSA SAN LUIS VALLEY RGNL
0130.NOAA.FrostDateL28S.Year~HydroBase
0130.NOAA.FrostDateL32S.Year~HydroBase
0130.NOAA.FrostDateF32F.Year~HydroBase
0130.NOAA.FrostDateF28F.Year~HydroBase
# Add Meeker Stations (5484 and 5487)
# then "free" 5487
5484.NOAA.FrostDateL28S.Year~HydroBase
5484.NOAA.FrostDateL32S.Year~HydroBase
5484.NOAA.FrostDateF32F.Year~HydroBase
5484.NOAA.FrostDateF28F.Year~HydroBase
5487.NOAA.FrostDateL28S.Year~HydroBase
5487.NOAA.FrostDateL32S.Year~HydroBase
5487.NOAA.FrostDateF32F.Year~HydroBase
5487.NOAA.FrostDateF28F.Year~HydroBase
fillFromTS(5484.NOAA.FrostDateL28S.Year,5487.NOAA.FrostDateL28S.Year,*,*)
fillFromTS(5484.NOAA.FrostDateL32S.Year,5487.NOAA.FrostDateL32S.Year,*,*)
fillFromTS(5484.NOAA.FrostDateF32F.Year,5487.NOAA.FrostDateF32F.Year,*,*)
fillFromTS(5484.NOAA.FrostDateF28F.Year,5487.NOAA.FrostDateF28F.Year,*,*)
free(TSID=”5487*”)
#
# __________________________________________________________
fillHistYearAverage(*)
#
# __________________________________________________________
writeStateCU("..\StateCU\Frost2002.stm")
```

6.2.6 Modeling – Filling Streamflow Using MOVE2 (Input=HydroBase)

Data filling is an important activity for modeling. TSTool provides a number of data filling commands, as described in **Section 4.4 – Fill Time Series Data**. Data filling can be accomplished using varying levels of complexity. The approach used for data filling depends on the data type and interval. For example, estimating daily precipitation may be difficult because relationships between daily precipitation time series may not exist. TSTool provides tools for data filling but it does not automatically pick the most appropriate fill methods. Consequently, data filling involves a number of steps:

1. Initial review of the data (e.g., using the ***Results…Report - Data Coverage by Year*** menu, and graphs)
2. Review of the spatial proximity of gages using the TSTool ***View…Map Interface*** capability or GIS software.
3. Comparison of candidate time series (e.g., using the ***Results…Graph - XY-Scatter*** menu)
4. Apply data filling commands
5. Review final results visually and review time series histories (by right-clicking on a time series in the ***Time Series Results*** list and selecting ***Time Series Properties***)

As indicated above, the data filling approach can be simple or complex. An example of a complex data filling technique is to use the `fillMOVE2()` command on daily streamflow data. In particular, consider the following case:

- Time series 1 (TS1) has a long period of gaged unregulated data (e.g., a headwater): 1900 to 2000
- Time series 2 (TS2) has a shorter period of gaged data with 1920 to 1950 being unregulated and 1950 to 2000 being regulated (e.g., due to the construction of a reservoir)

- The goal is to produce an estimate of unregulated flow for TS2 for the full period 1900 to 2000.

This can be accomplished using the following commands:

```
#
# Data filling example – assume daily DateValue time series as input
#
# Set the output period so that time series are guaranteed to have a period to fill…
setOutputPeriod(1900-01-01,2000-12-31)
# Read the dependent time series (TS2), which will receive the final result
TS ts2 = readDateValue(InputFile="ts2.txt")
# Read the independent time series (TS1)
TS ts1 = readDateValue(InputFile="ts1.txt")
# Make a copy of the dependent to analyze and fill after 1950…
# Need this because we need to clear out the regulated data
TS ts2_copy = copy(ts2)
# Set the later period to missing in the copy so it will be filled…
replaceValue(ts2_copy,-10000,10000,-999,1950-01-01,*)
# Analyze and fill the early period.  Transform the data to log10 and
# use monthly equations.  The first pair of dates is the dependent analysis period.
# The second pair of dates is the independent analysis period.  The third pair of
# dates is the fill period.
fillMOVE2(ts2,ts1,MonthlyEquations,Log,1920-01-01,1949-12-31,*,*,*,1919-12-31)
# Analyze and fill the later period.  Transform the data to log10 and
# use monthly equations…
fillMOVE2(ts2_copy,ts1,MonthlyEquations,Log,1920-01-01,1949-12-31,*,*,1950-01-01,*)
# Now set the later period into the main time series…
setFromTS(ts2,ts2_copy,1950-01-01,*)
```

The above example illustrates a somewhat complicated situation where data filling is facilitated by the features of the `fillMOVE2()` command. If the `fillMOVE2()` command is not appropriate, then the `fillRegression()` or other commands can be applied. In some cases, it may be appropriate to fill different parts of the period using different independent time series. A simpler approach may involve only a single filling step (e.g., fill the entire period using a single `fillRegression()` command).

6.3 Time Series Trace Examples

The general term *time series traces* refers to groups of a time series, often shown in overlapping fashion. Common ways to create traces are:

- Split time series into N-year lengths and shift to overlap.
- Run a model multiple times with different input, in order to generate many possible outcomes.
- Generate synthetic data to use as input to a model.

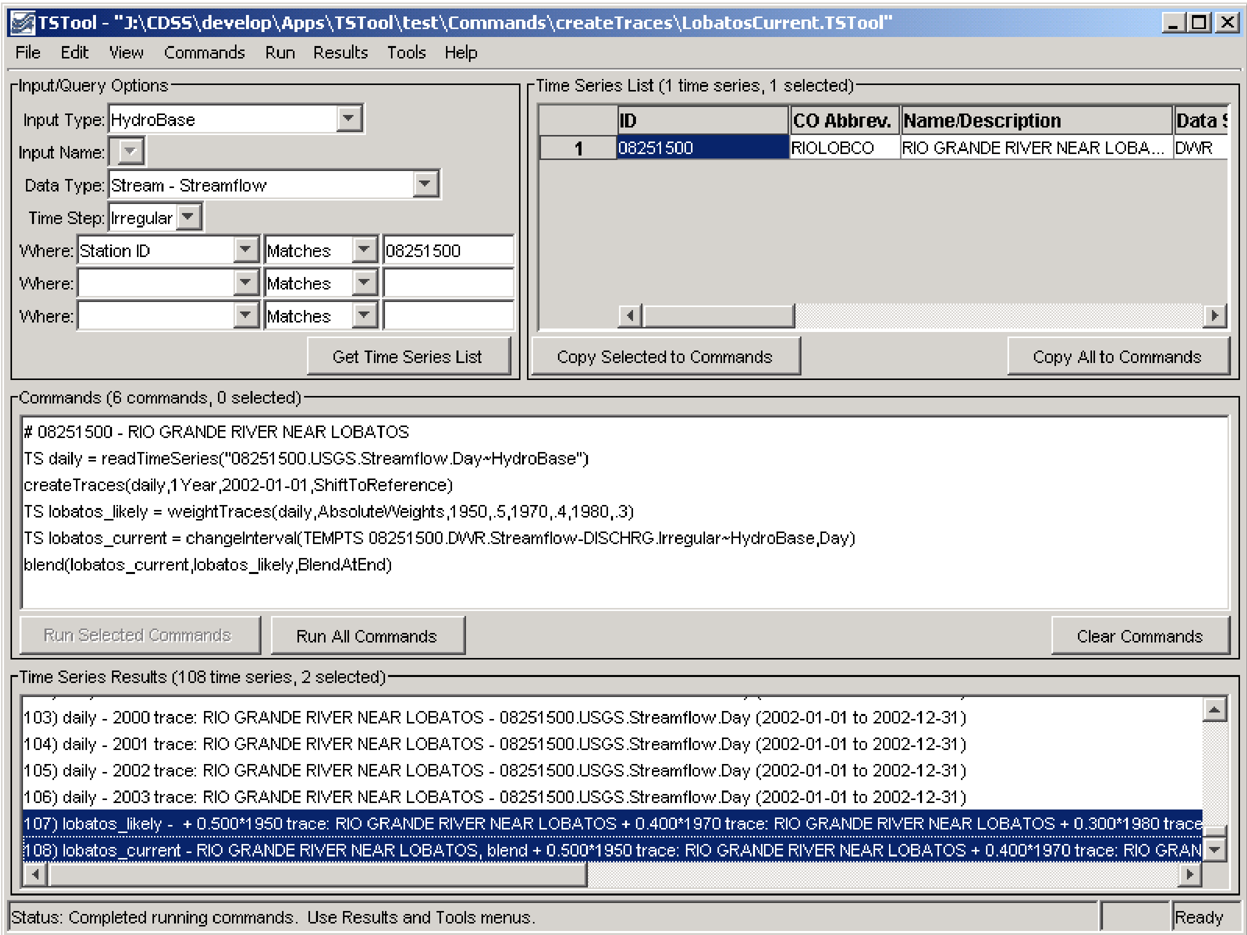
Several TSTool commands have been implemented to create and process time series traces. TSTool manages trace data by using the same identifier for all traces and internally keeping a trace (sequence) number, which is the same as the starting year of the trace. An extension to the time series identifier convention may be made in the future to allow a trace to be uniquely identified by a time series identifier. Currently, you must use the GUI and its time series selection features to select individual traces.

6.3.1 Time Series Traces – Comparing Historical and Current Conditions (Input=HydroBase)

The following commands file illustrates how historic time series traces can be plotted on top of real-time data. The features illustrated in the example were implemented to help determine an estimate of future flow based on current conditions.

```
#
# These commands query historic and real-time data at the lobatos gage and
# compute a weighted "best-guess" for the flows at lobatos for the remainder
# of the current year.
#
# First get the historic daily lobatos gage and convert to traces.  Shift each trace to
# 01/01/2001 so that the data can overlay the current values.  Internally, each trace
# has the identifier 08251500.DWR.QME.Day (which is used in commands) and a
# hidden sequence number that matches the year of the start of the trace.
# Note that the TEMPTS key word is used in the command to query the daily
# time series so the time series is discarded after conversion to traces.
TS daily = readTimeSeries("08251500.USGS.Streamflow.Day~HydroBase")
createTraces(daily,1Year,01/01/2002,ShiftToReference)
#
# Now weight the traces using representative historic years.  This uses absolute
# weights which means that the total of the weights can be different than 1.0.  An
# option to use normalized weights may be added to force the weights to scale to 1.0,
# even with missing data.
#
TS lobatos_likely = weightTraces(daily,AbsoluteWeights,1950,.5,1970,.4,1980,.3)
#
# Now query the current (real-time) flows.  HydroBase may only hold a few weeks or months
# of data.
#
# Uncomment the following line to see the actual real-time values (slower)
# 08251500.DWR.RT_rate.Real-time.DISCHRG
TS lobatos_current = changeInterval(TEMPTS 08251500.DWR.RT_rate.Real-time.DISCHRG,Day)
#
# After the above commands are executed, time series in memory will include the traces and
# the current time series.  You can select only the time series of interest and plot
# OR select many time series and then disable/enable in the plot.  You may need to use
# both approaches to find appropriate time series to weight.
blend(lobatos_current,lobatos_likely,BlendAtEnd)
```

The results of processing the above commands in TSTool are a list of the traces, a weighted time series (based on three traces), and the current daily data, all at the same streamflow gage, as shown in the following figure.

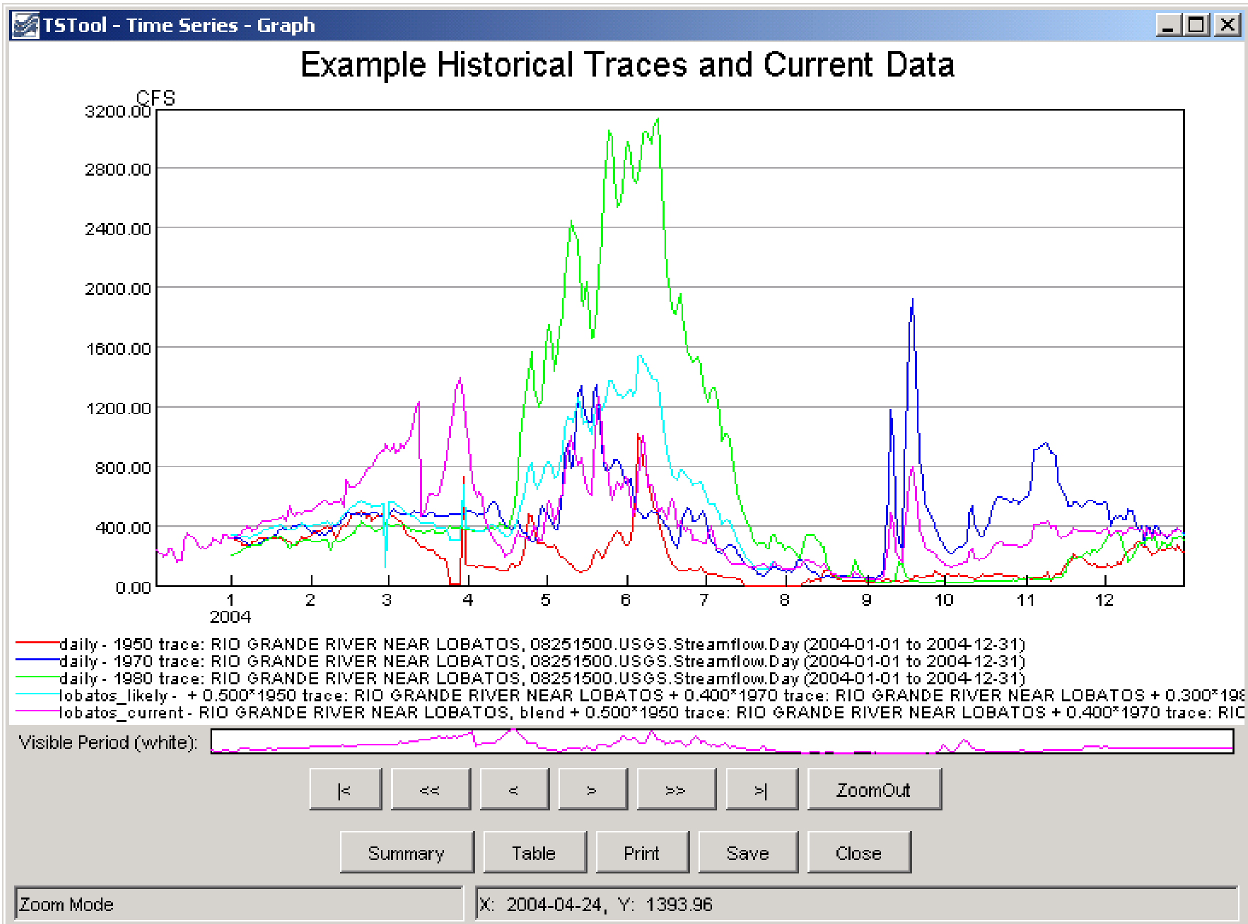


Example_LobatosCurrent_TSTool

Any of the time series can be selected and viewed. The following features are useful for selecting appropriate traces:

1. Convert a time series to traces and then plot all the traces. Use the graph properties to turn traces on/off or use symbols to identify. You can consequently visually select traces that are similar.
2. Use the tools described in **Chapter 5 - Tools** to evaluate time series and traces. For example, the ***Year to Date*** report can be used to determine how well different years compare volumetrically.

Key traces and output time series can be selected and graphed, as shown below.



Example_LobatosCurrent_Graph

6.4 Time Series Product Examples

Time series products are described in the **TSView Time Series Viewing Tools Appendix**. In summary, a time series product file can be generated that uses time series identifiers to indicate data to be processed, and includes other properties (e.g., titles) to configure a graph. TSTool can process time series products in a number of ways, as illustrated by the following examples.

6.4.1 Time Series Product - Using TSTool to Display Graphs from an Application

TSTool is primarily used as an interactive tool or to process commands files in batch mode. However, it is also possible to run TSTool with a commands file, no main graphical user interface, and still display only specific graphs. For example, TSTool can be called from an application to display a graph by reading data from a recognized database or file format. This takes advantage of TSTool's features rather than adding additional features to the application. The following example illustrates how to display a graph of precipitation and streamflow data in a single graph, using data from the State of Colorado's HydroBase database. TSTool should be started using a command line similar to:

```
tstool –commands example.tstool –nomaingui
```

Additionally, the HydroBase database to be used should be configured in the TSTool configuration file (see the **Installation and Configuration Appendix**). This run mode actually runs the GUI; however, the GUI is never made visible. Instead, it is used to control a batch commands file run, with graphical displays. Because the interactive main interface is disabled, the normal HydroBase login dialog is not shown; therefore, the HydroBase information must be defined in the configuration information.

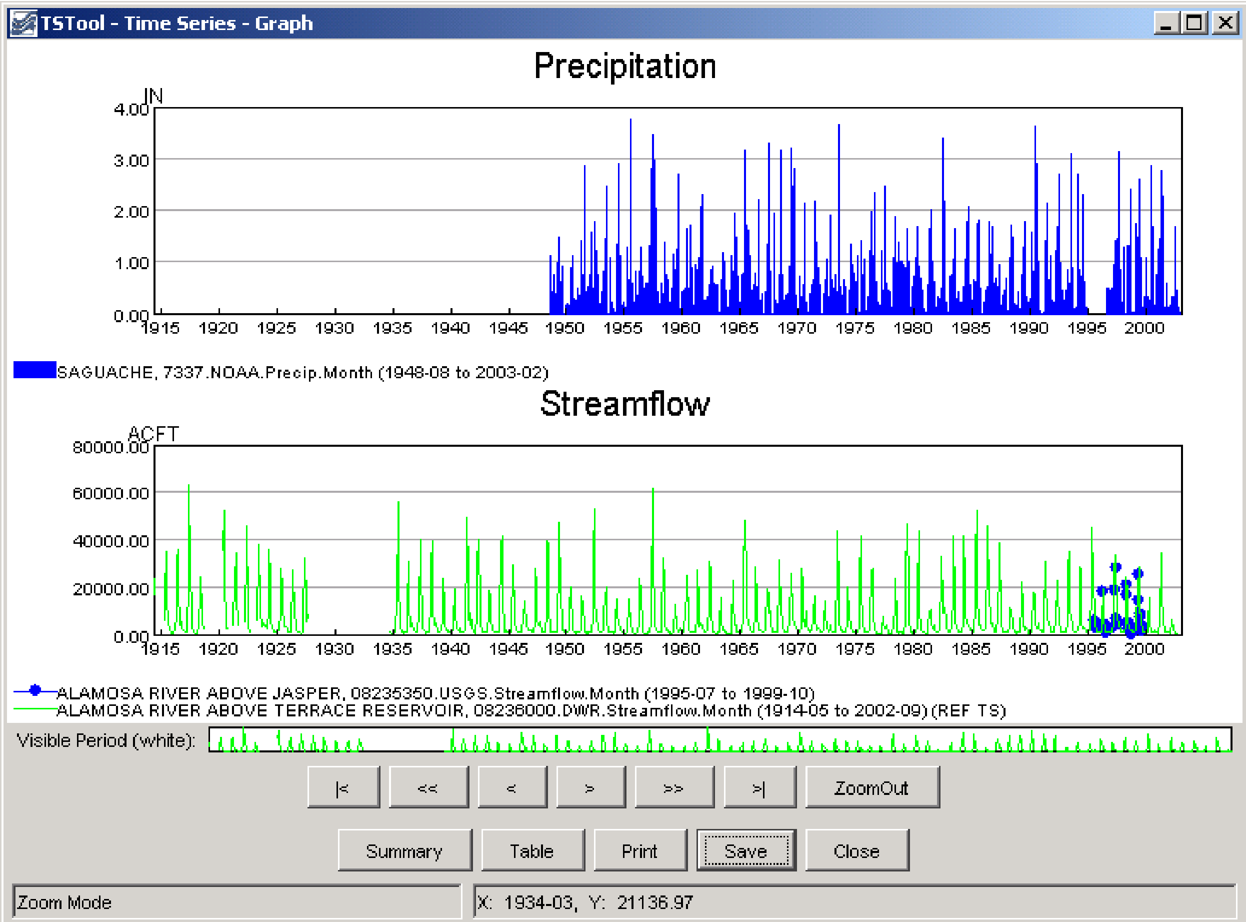
Although it is possible to display several graphs at the same time, it is currently assumed that only one graph will be shown. Closing the graph will close TSTool. The commands file can be complex but in many cases will be simple because an application is calling TSTool to display a single graph. The following example shows a typical commands file for this run mode:

```
# Example commands file to run a commands file without showing the main GUI
# but showing a plot to the screen.  When the plot window closes, TSTool will
# exit without prompting.  TSTool should be called using:
#
# TSTool -commands ThisFile -nomaingui
#
# This is useful for displaying plots from applications that don't have features
# to query HydroBase.
#
# SAM, RTi, 2002-06-02
#
# Process a time series product description file and display a plot window
# to the screen.
processTSProduct("test.tspd",GUIAndBatch,Preview)
```

The `processTSProduct()` command references a time series product file. An example of the file is as follows (see the **TSView Time Series Viewing Tools Appendix** for a full description of time series product file properties):

```
[Product]

ProductType = "Graph"
TotalHeight = "400"
TotalWidth = "600"

[SubProduct 1]

GraphType = "Bar"
MainTitleString = "Precipitation"
BarPosition = "CenteredOnDate"

[Data 1.1]

Color = "Blue"
TSID = "7337.NOAA.Precip.Month~HydroBase"

[SubProduct 2]

GraphType = "Line"
MainTitleString = "Streamflow"

[Data 2.1]

SymbolSize = "7"
SymbolStyle = "Circle-Filled"
TSID = "08235350.USGS.Streamflow.Month~HydroBase"

[Data 2.2]

TSID = "08236000.DWR.Streamflow.Month~HydroBase"
```

The resulting graph is shown in the following figure. Pressing ***Close*** will exit TSTool.



Example_NoMainGUI_Graph

6.4.2 Automating Graphs to Compare Observed and Simulated Time Series

It is often useful to automate comparison of observed and simulated time series (e.g., during model calibration). For example, after making an adjustment to a model during calibration, many comparisons may be made to evaluate the changes. TSTool can help with comparisons. For example, consider the simulated and observed time series stored in a DateValue file (*results.dv*), as follows (in this case the DateValue file was actually created by reading model input and output, and the `TSID` and `DataType` lines were hand-edited in the file to facilitate this example).

```
# DateValueTS 1.3 file
# File generated by...
# program:   TSTool 6.08.02 (2004-07-27) Java
# user:      sam
# date:      Thu Jul 29 09:17:35 MDT 2004
# host:      host unknown
# directory: J:\CDSS\develop\Apps\TSTool\test\Commands\createTraces
# command:   TSTool -home C:\CDSS
#--------------------------------------------------------------------
#
# Commands used to generate output:
#
# 09152500...MONTH~StateMod~J:\CDSS\DataSets\SWSI_Gunnison\StateMod\gunnv.rih
# 09152500.StateMod.River_Outflow.Month~StateModB~J:\CDSS\DataSets\SWSI_Gunnison\StateMod\gunnvb.b43
#
Delimiter   = " "
NumTS       = 2
TSID        = "09152500..StreamFlow_Observed.MONTH" "09152500..Streamflow_Simulated.Month"
Alias       = "" ""
Description = "09152500" "Gunn R. NR GrandJ   _FLO"
DataType    = "Streamflow_Observed" "Streamflow_Simulated"
Units       = "ACFT" "ACFT"
MissingVal  = -999.0000 -999.0000
Start       = 1908-10
End         = 2001-09
#
# Time series comments/histories:
#
#
# Creation history for time series 1 TSID=09152500...MONTH Alias=):
#
# Read StateMod TS for 1908-10 to 2001-09 from "J:\CDSS\DataSets\SWSI_Gunnison\StateMod\gunnv.rih"
#
# Creation history for time series 2 TSID=09152500.StateMod.River_Outflow.Month Alias=):
#
#   Read from "J:\CDSS\DataSets\SWSI_Gunnison\StateMod\gunnvb.b43 for 1908-10 to 2000-09
#
#EndHeader
Date "09152500...MONTH, ACFT" "09152500.StateMod.River_Outflow.Month, ACFT"
1908-10 -999.0000 82035.8828
. . . omitted – periods do not exactly line up . . .
1916-10 61488.5000 95692.6641
1916-11 56529.8000 97254.9297
1916-12 55339.6000 94700.1563
1917-01 52265.2000 47388.2305
1917-02 49984.2000 42303.5938
1917-03 79935.0000 64748.8125
. . . similar to end of file . . .
```

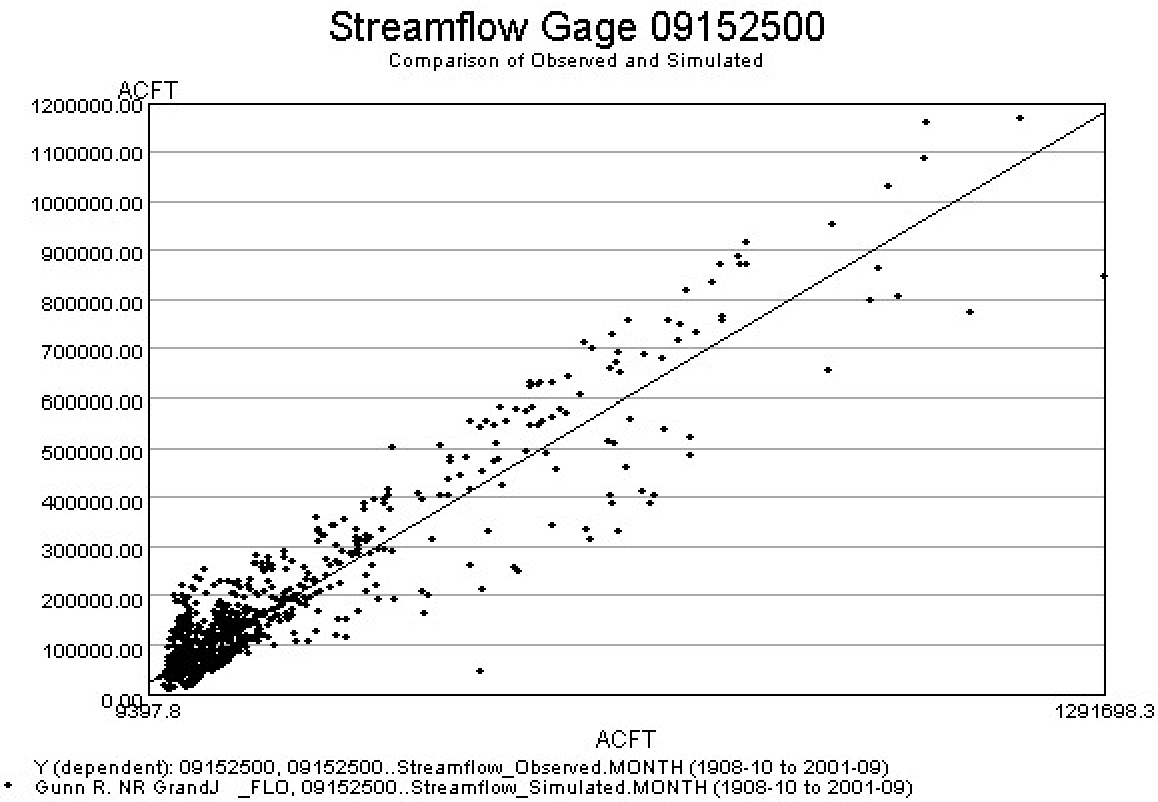
These time series can be read into TSTool, an XY graph produced, and the time series product saved (*results.tsp*), as shown in the following example (note the original TSID properties have been inserted, corresponding to the original data):

```
[Product]

ProductType = "Graph"
TotalWidth = "600"
TotalHeight = "400"
MainTitleString = "Streamflow Gage 09152500"
SubTitleString = "Comparison of Observed and Simulated"

[SubProduct 1]

GraphType = "XY-Scatter"
XYScatterMethod = "OLSRegression"
LegendFormat = "Auto"
MainTitleString = ""

[Data 1.1]

#TSID = "09152500...MONTH~StateMod~J:\CDSS\DataSets\SWSI_Gunnison\StateMod\gunnv.rih"
TSID = "09152500..Streamflow_Observed.MONTH~DateValue~results.dv"

[Data 1.2]

#TSID = "09152500.StateMod.River_Outflow.Month~StateModB~J:\CDSS\DataSets\SWSI_Gunnison\StateMod\gunnvb.b43"
TSID = "09152500..Streamflow_Simulated.MONTH~DateValue~results.dv"
```

Finally, a commands file (*results.TSTool*) can be created that processes the time series and time series product file:

```
processTSProduct("results.tsp",GUIAndBatch,NoPreview,"09152500.jpg")
```

The above commands can be run from the TSTool GUI or in batch mode to produce the following graph (note in this example that the x-axis data values are so large that the software is having difficulty finding good labels):



09152500.jpg

Because this approach relies primarily on the time series identifiers to associate the time series data with the time series product, it is important to establish a concise and clean identifier scheme. Once a working example is established, the example can be scaled up to a larger production either by repeating the example (and changing identifiers) or by automatically changing the example to replace strings. The latter is not currently part of TSTool; however, TSTool can call external programs (see the `runProgram()` command), which could supplement the existing TSTool features.