Flags

/C Tells PauseForUser to return immediately after handling any pending events. See Details.

Details

During execution of PauseForUser, only mouse and keyboard activity directed toward either *mainWindowName* or *targetWindowName* is allowed.

While waiting for user action, PauseForUser disables double-clicks and any contextual menus that can lead to dialogs in order to prevent changes on the command line. It also disables killing windows by clicking the close icon in the title bar unless the window was originally created with the /K=1 flag (kill with no dialog).

If /C is omitted, PauseForUser returns only when the main window has been killed.

If /C is present, PauseForUser handles any pending events, sets V_Flag to the truth the target window still exists, and then returns control to the calling user-defined function. Use PauseForUser/C in a loop if you need to do something while waiting for the user to finish interacting with the target window, such as updating a progress indicator.

As of Igor Pro 8.00, the PauseForUser operation is allowed only in user-defined functions. It returns an error if called from a macro or from the command line.

See Also

Pause For User on page IV-152 for examples and further discussion.

PauseUpdate

PauseUpdate

The PauseUpdate operation delays the updating of graphs and tables until you invoke a corresponding ResumeUpdate command.

Details

PauseUpdate is useful in a macro that changes a number of things relating to the appearance of a graph. It prevents the graph from being updated after each change. Its effect ends when the macro in which it occurs ends. It also affects updating of tables.

This operation is not allowed from the command line. It is allowed but has no effect in user-defined functions. During execution of a user-defined function, windows update only when you explicitly call the DoUpdate operation.

See Also

The **DelayUpdate**, **DoUpdate**, **ResumeUpdate,** and **Silent** operations.

PCA

PCA [*flags*][*wave0, wave1,… wave99*]

The PCA operation performs principal component analysis. Input data can be in the form of a list of 1D waves, a single 2D wave, or a string containing a list of 1D waves. The operation can produce multiple output waves depending on the specified flags.

Flags

/ALL	Shortcut for the combination of commonly used flags: /CVAR, /SL, /NF, /IND, /IE, and /RMS.
/COV	Calculates the input wave(s) covariance matrix, which as the input for the remainder of the analysis. The covariance matrix is computed by first creating a matrix copying each input 1D wave into sequential columns and then multiplying that matrix by its transpose.
/CVAR	Computes the cumulative percent variance defined as 100 * sum of first *m* eigenvalues divided by the sum of all eigenvalues. The results are stored in the wave W_CumulativeVAR in the current data folder. See also /VAR.

/IE — Computes the imbedded error. Returns errors in the wave W_IE in the current data folder. The wave is scaled using `SetScale/P x 1,1,"", W_IE`. The imbedded error is a function of the number of factors, the number of rows and columns and the sum of the eigenvectors not included in the significant factors. The behavior of IE determines the number of significant factors.

/IND — Computes the factor indicator function. Note that if you specify /IND the residual standard deviation will also be calculated. Returns results in the wave W_IND in the current data folder. The wave is scaled using `SetScale/P x 1,1,"", W_IND`.

/LEIV — Limits eigenvalues so that the SVD calculation does not require too much memory. The limit is set to the minimum of the number of rows or columns of the input.

/NF — Finds the number of significant factors and stores it in the variable V_npnts. You must use /IND in order to compute the significant factors.

/O — Overwrites input waves.

/Q — Suppresses printing of factors in the history area.

/RSD[=*rsdMode*] — Computes the Residual Standard Deviation (RSD) and returns the RSD in the wave W_RSD in the current data folder. The first element in W_RSD is NaN and all remaining wave elements correspond to the number of significant factors.

rsdMode =0: Covariance about the origin.

rsdMode =1: Correlation about the origin.

/RMS — Computes the RMS error. Returns results in the wave W_RMS in the current data folder. The wave is scaled using `SetScale/P x 1,1,"", W_RMS`.

/SCMT — Saves C matrix after the singular value decomposition (SVD) in the wave M_C in the current data folder.

/SCR — Converts the individual wave input into standard scores. Does not work when the input is a single 2D wave. It is an error to convert to standard scores when one or more entries in the waves are NaN or INF. If you use this feature make sure to use the appropriate form of the RSD calculation.

/SDM — Saves a copy of the data matrix at the end of the calculation. This is useful if your input consists of individual waves or if you want to save the computed standard scores. If the input is a 2D matrix, you will get a copy of the input matrix in the wave M_D.

/SEVC — Saves the eigenvalue vector in the wave W_Eigen, which are the raw eigenvalues generated by the SVD, in the current data folder. Normally, if the SVD was applied to a raw data matrix, i.e., not covariance or correlation matrix, you must square each element of the wave to obtain the PCA eigenvalues. Note that this wave has default wave scaling.

/SL — Computes percent significance level and stores it in the wave W_PSL in the current data folder.

/SQEV — Does not square SVD eigenvalues. If you specify /COV there is no need to use this flag.

Use only if your input is already a covariance matrix. In this case the results of the SVD are the eigenvalues not their square roots.

/SRMT — Saves R matrix after the SVD in the wave M_R in the current data folder.

/U — Leaves the input waves unchanged only when the input is a 2D wave. Note that covariance calculations will not be made even if the appropriate flag is used.

/VAR — Computes the variance associated with each eigenvalue. The variance is defined as the ratio of the eigenvalue to the sum of all eigenvalues. The results are stored in the wave W_VAR in the current data folder. See also /CVAR above.

/WSTR=*waveListStr*