



THE  
POWER  
TO KNOW.

**SAS<sup>®</sup> Enterprise Miner<sup>™</sup> and  
SAS<sup>®</sup> Text Miner Procedures  
Reference for SAS<sup>®</sup> 9.1.3  
The TMUTIL Procedure  
(Book Excerpt)**

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS® *Enterprise Miner™* and SAS® *Text Miner Procedures: Reference for SAS® 9.1.3*, Cary, NC: SAS Institute Inc.

**SAS® Enterprise Miner™ and SAS® Text Miner Procedures: Reference for SAS 9.1.3**

Copyright © 2008 by SAS Institute Inc., Cary, NC, USA.

All rights reserved. Produced in the United States of America.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

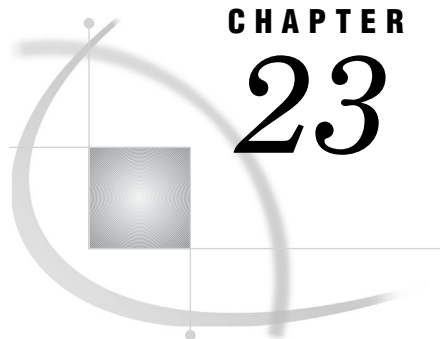
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, October 2008

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/publishing](http://support.sas.com/publishing) or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.



# CHAPTER 23

## The TMUTIL Procedure

---

<i>Overview: TMUTIL Procedure</i>	<b>451</b>
<i>Syntax: TMUTIL Procedure</i>	<b>451</b>
<i>PROC TMUTIL Statement</i>	<b>452</b>
<i>CONTROL Statement</i>	<b>453</b>
<i>FILTER Statement</i>	<b>454</b>
<i>OUTPUT Statement</i>	<b>456</b>
<i>WEIGHT Statement</i>	<b>457</b>
<i>SYN Statement</i>	<b>459</b>
<i>SELECT Statement</i>	<b>460</b>
<i>Examples: TMUTIL Procedure</i>	<b>461</b>
<i>Example 1: PROC TMUTIL and PROC DOCPARSE</i>	<b>461</b>
<i>Example 2: CONTROL Statement</i>	<b>461</b>
<i>Example 3: OUTPUT Statement</i>	<b>462</b>
<i>Example 4: CONTROL and OUTPUT Statements</i>	<b>462</b>
<i>Example 5: WEIGHT Statement</i>	<b>463</b>
<i>Example 6: SYN Statement</i>	<b>463</b>
<i>Example 7: FILTER Statement</i>	<b>464</b>
<i>Example 8: SELECT Statement</i>	<b>466</b>
<i>Example 9: FACTOR Statement</i>	<b>466</b>

---

### Overview: TMUTIL Procedure

PROC TMUTIL is a utility procedure for SAS Text Miner. The procedure is designed to support the text mining nodes found in Enterprise Miner but is also available for users to build custom text mining solutions. Its primary input source is the output from PROC DOCPARSE. The TMUTIL procedure uses an efficient internal representation of the output from PROC DOCPARSE called an inverted index. The inverted index can be stored in memory as long as the user desires. This allows the user to manipulate the data with several TMUTIL calls without having to reload potentially very large data sets each time. The procedure has several important functions:

- Maintain the inverted index structure that represents the document collection.
- Alter the inverted index structure based on interactive input (that is setting synonyms, dropping terms, and filtering).
- Apply transformations to the data via weightings and factorizations.
- Write out various data set representations of the inverted index structure.

---

### Syntax: TMUTIL Procedure

```
PROC TMUTIL <option(s)>;
```

**CONTROL**<*option(s)*>;  
**FILTER**<*option(s)*>;  
**OUTPUT**<*option(s)*>;  
**SELECT**<*option(s)*>;  
**SYN**<*option(s)*>;  
**WEIGHT**<*option(s)*>;

---

## PROC TMUTIL Statement

**PROC TMUTIL** <*option(s)*>;

### Options

#### **DATA=** <*term-doc freq data set*>

Input DATA table is the OUT data from PROC DOCPARSE. This data set is a compressed representation of a sparse term-document frequency matrix. There are three required variables on the data set.

- **\_TERMNUM\_** — Contains integer indices greater than zero that corresponding to the KEY on the KEY data set. These integers are the unique identifier for a term.
- **\_DOCUMENT\_** — An integer that corresponds to the document index in the DOC data set.
- **\_COUNT\_** — An integer specifying the number of times that term **\_TERMNUM\_** occurred in DOC **\_DOCUMENT\_**.

*Note:* The DATA data set must be used in combination with an INIT option in the CONTROL statement.  $\Delta$

#### **DOC | DOCUMENTS=**<*doc data set*>

Input DOCUMENT data set. While PROC TMUTIL does not store any of the actual text of the documents internally, it can use and store the contents of the following two variables:

- **\_DOCUMENT\_** — A positive integer identifying the document ID. If this is not specified, the **\_DOCUMENT\_** entry will be generated sequentially beginning with 1.
- **Target variable** — Any nominal or ordinal variable that is identified with the TARGET option. This variable is used when calculating the categorical weightings and is specified with the TARGET option.

#### **KEY | TERMS=**<*libref*>**SAS-data-set**

Specifies the SAS data set that contains the terms. The table is typically the output KEY data set created by the DOCPARSE procedure. The KEY data set is essential if synonyms, stems, or start or stop lists have been applied when the DOCPARSE data sets were generated. If the KEY data set is not specified, every term on the OUT data set will be treated as kept and there will be no parents or synonyms assigned. The following is a list of variables in the SAS data set that can be used by PROC TMUTIL:

- `_TERMNUM_ | KEY | INDEX` — the index of the term. This must be an integer greater than zero. If the KEY data set is used, this variable is required.
- `FREQ` — an integer indicating the number of times the term occurs in the document collection. If this variable is left off, PROC TMUTIL will compute it from the OUT data set and subsequently write the variable to requested output KEY data sets..
- `NUMDOCS` — an integer representing the number of documents in which the term occurs in. If this variable does not exist, PROC TMUTIL will compute it from the OUT data set and subsequently write the variable to requested output KEY data sets.
- `KEEP` — a character variable “Y” or “N”, indicating if the term is currently kept or dropped. If this variable does not exist, all terms are considered to be kept and future KEY data sets that are written out will indicate this.
- `PARENT` — an integer or missing value that indicates whether the term is to be mapped to a representative term. If the value is missing, the term has no parent. If the value is an integer, the value is the key to the term that is its representative. It is possible for the key and the parent value to be the same. In this case, NUMDOCS and FREQ refer to the number of times the term itself appeared in the document (that is the frequency of the surface form of the term). The `_ISPAR` variable is required when the PARENT variable is used.
- `_ISPAR` — a character variable indicating if the term is a parent, child, or neither. The variable is required whenever there are parent child relationships on the input KEY data set. The PARENT variable is required when the `_ISPAR` variable is used. The variable can take on one of three values:
  - "+" (plus sign) if the term is functioning as a parent (representative) term.
  - " " (space char) if the term is functioning as neither a parent or a child and
  - "." (period) if the term is functioning as a child and
- `FILTER` — An integer valued variable that indicates which term is to be filtered at what level. A "0" indicates the term is not filtered. If the variable is not on the input KEY every term is at filter level "0" (unfiltered). See the FILTER statement and the FILTERVAR option in the OUTPUT statement for additional information.

**TARGET=** < *variable—name* >

Indicate which variable on DOC data set is to be used in computing the target-specific weightings, MI | IG | CHI. Input into this variable is a string that makes up the variable name. See the documentation for the weightings under the WEIGHT statement.

---

## CONTROL Statement

PROC TMUTIL is unusual in that the data that it uses persists in memory until the user explicitly asks that it be cleared or until sas has exited. After the initial call to PROC TMUTIL, subsequent calls can be made without the expensive overhead of reading in the list of documents and terms. When the TMUTIL procedure is called again, it can reconnect to the data that was stored in memory on the initial call.

The CONTROL statement has options to tell the proc if this is the first or last time it is being called. It also has an option to indicate the name of the SAS macro variable that holds the memory. At times memory might be tight, and you might want to release the memory in order to perform other tasks. If you want to be able to start PROC TMUTIL up with the same data at a later time, be sure to include an output KEY and

OUT data set with the OUTPUT statement. Later, TMUTIL can be reinitialized with this output KEY and OUT data set. A further savings in space can be realized if the RENUMBER option in the OUTPUT statement is also used.

**CONTROL** <option(s)>;

## Options

### INIT|INITIAL

Indicates the initial time the proc is called and memory needs to be allocated so that the input data can be stored. An input DATA data set is required with an INIT call and these options must be provided on the first call to the proc.

### MEMLOC=<string>

String for use as macro variable name. The macro variable holds the memory location of the allocated memory. If several TMUTIL instances are to be used simultaneously, a different MEMLOC string should be used for each of those instances. This parameter is required on all calls to the proc unless INIT and RELEASE are used simultaneously.

### RELEASE

Tells the proc to release all allocated memory upon completion. If not specified, the memory will not be released and its address will reside in the MEMLOC macro variable. Subsequent calls to the proc will access the already allocated memory via this macro variable. The user should issue RELEASE on the last call, otherwise system memory will not be freed and will be unavailable to SAS until SAS is terminated and restarted.

---

## FILTER Statement

**FILTER** <option(s)>;

The FILTER statement is used to subset the collection based on a list of terms or documents. The FILTER statement allows for the temporary exclusion of specific terms or documents. Unlike the select statement, FILTER statements can be undone with the undo command. This allows for a quick examination of a subset of the collection and gives the user the ability to return to the original collection when the subsetting is finished.

The FILTER statement enables the user to identify which terms or documents should be used in future analysis. All remaining terms or documents are set to a filter level and are not used in the analysis. If the COMP (compliment) option is used, filters work in reverse and filter out the terms and docs that would otherwise have been used.

The knowledge of each requested filter is retained by the procedure and can be undone with the UNDO and UNDOALL commands. The REDUCEN, REDUCEW, REDUCEF options allow one to filter in and out terms based on the number you want kept, the weight, or the frequency, respectively. The REDUCE options in the FILTER statement allow the filters to be undone.

### COMPLEMENT|COMP

Boolean option that is to be used in combination with TERMDATA, TERMLIST, DOCDATA, or DOCLIST. When it is used the specified set is filtered out rather than filtered in.

**DOCDATA=<doc id data set>**

Data set of docs to use in the analysis. All other docs are filtered out. The data set requires a single variable called either `_DOCUMENT_`, `KEY`, or `INDEX`. All docs not on the list are removed from the analysis and corresponding counts for the terms are updated. If the `COMP` option is used, the docs listed are removed from the analysis rather than kept.

**DOCLIST**

An alternative method to `DOCDATA` for filtering terms. The approach accepts a list of positive integers to be filtered from use. If the `COMP` option is used the complement of the set of `DOCS` on this data set are filtered from use.

**REDUCEF=X**

Filter out Terms that are not contained in at least `X` docs. Can be used in conjunction with `REDUCEW`.

**REDUCEN=X**

Filter out all but the highest `X` weighted terms. This option allows the user to control exactly how many terms are left unfiltered. If less than `X` kept terms exist, then the option will have no effect. This option cannot be used with `REDUCEW` or `REDUCEF`.

**REDUCEP=X**

Filter out all terms that belong to more than `X%` of the documents. This option is for removing terms that occur too frequently in your collection.

**REDUCEW=X**

Filter out all terms that are weighted less than `X`. Can be used in conjunction with `REDUCEF` in the `FILTER` statement.

**TERMDATA=<term id data set>**

Data set of terms to filter in. The data set requires a single variable called `_TERMNUM_`, `KEY`, or `INDEX`. All terms on the list are kept for analysis and those that are not listed are set to the current filter level. If the `COMP` option is used the listed terms are removed from the analysis rather than kept.

**TERMLIST**

An alternative method to `TERMDATA` for filtering terms. The approach accepts a list of positive integers to be maintained for use. If the `COMP` option is used the listed terms are set to the current filter level and removed from the analysis.

**UNDO**

Undoes the immediately preceding filter, whether it was a doc filter or a term filter.

**UNDOALL**

Undoes all preceding filters.

**WITHALL**

This option can be used in conjunction with the `TERMDATA` data set or the `TERMLIST` list to filter in documents that contain all of the terms specified. This option functions like a query with a logical "AND" of all the terms listed. The `COMP` option can be used in conjunction with the `WITHALL` option to reverse the set of documents that are filtered.

**WITHANY**

This option can be used in conjunction with the `TERMDATA` data set or the `TERMLIST` list to filters in documents that contain any of the terms specified. This option functions like a query with a logical "OR" of all the terms listed. The `COMP` option can be used in conjunction with the `WITHANY` option to reverse the set of documents that are filtered.

## OUTPUT Statement

The OUTPUT Statement is used to specify the data sets that the user wishes to be written out along with some options that control that output. In general the user will manipulate the key and out data set with FILTER statements, SELECT statements and KEEP and DROP statements, among other things. This manipulation will alter the KEY, OUT, and DOC data sets and the OUTPUT statement allows the user to request updated copies of these data sets. Also, current versions of the stop or synonym list can be output.

**OUTPUT** *<option(s)>*;

### **DOC=<doc-data-set>**

Output the data set of the documents. The data set will contain a `_DOCUMENT_` variable that identifies the document and additional variables for SVD dimensions, Roll-up-terms, and so on.

### **KEEPONLY**

Specifies the output key data set is to have only the kept terms on it.

### **KEY= <key-data-set>**

The KEY data set that is output produces the same set of variables that are read in on input. See the proc option documentation for KEY above for more information.

### **KEYFORMAT=DEFAULT | SIMPLE | TMSCORE**

Controls the variables and contents that are placed on the KEY data set.

DEFAULT	Places variables on the data set that the Text Miner product requires.
SIMPLE	Does not place parent terms on the key data set twice as the DEFAULT method does. Instead, it adds two variables, PFREQ and PNUMDOC to indicate the representative counts. The SIMPLE option only outputs kept terms.
TMSCORE	Places only the variables necessary for scoring with the TMSCORE function. Only kept terms are placed on this data set, and frequencies and numdocs are not included because they are not necessary for scoring a new data set.

### **OUT=<out-data-set>**

Output OUT table. The OUT data set has the same variables as the input DATA data set but the counts can be weighted if requested. See the proc option documentation for DATA above for more information.

### **REINDEX**

This option instructs PROC TMUTIL to renumber all of the kept, terms sequentially. The option can be used only when there are no current filters, when an output OUT and KEY data set are specified, and when the RELEASE option is used. The option, followed up by a new INIT call to PROC TMUTIL with the reindexed data, can be used to save considerable space.

### **STOP=<term ID-data-set>**

Output all terms that are currently set to KEEP="No". There is one variable on the data set `_TERMNUM_`. The data set, when merged with the actual terms, can be used by Text Miner as a stop list.

### **SYN=<synonym-data-set>**

Will contain all synonyms that exist in the inverted index. The format of this data set is two columns `_TERMNUM_` and `PARENT`. The data set can be used by Text Miner as its synonym list.

### UNWEIGHTED

Indicate whether weights are to be applied to OUT table above.

**Default:** WEIGHTED

## WEIGHT Statement

**WEIGHT** *<option(s)>*;

The weight statement enables you to apply weightings to the term-document frequency table. There are two types of weights `TERM` and `CELL` weights. The `TERM` weights option calculates and assigns a positive number to each term that occurs in the collection. The `TERM` weight is based on the distribution of the term across the collection and can be interpreted as indicating the importance of the particular term for the collection. The `CELL` weight option, on the other hand, is a function that is applied to each entry in each cell of the term-document frequency matrix. It is not a function of how frequently a term occurs across the collection. Rather, it is a function that moderates the effect of a term that is repeated within a document. If the  $ij$ th entry in the term doc frequency matrix is  $f_{ij}$ , the `TERM` weight of term  $i$  is  $w_i$ , and the cell weight is the function  $g()$ , then the weighted frequency of the  $ij$ th entry is  $w_i * g(f_{ij})$ .

### CELLWGT = BINARY | LOG | NONE

Apply requested cell weightings.

**BINARY**            If the term occurs in the document, then cell weighting is the indicator function of  $f_{ij}$ . That is, it assigns a 1 if  $f_{ij}$  is nonzero. Otherwise, it leaves the frequency as 0.

**LOG**                 $cellweight = \log_2(f_{ij} + 1)$

**NONE**              No weighting function is applied to the cell.

### TERMWGT = NORMAL | GFIDF | IDF | ENTROPY | MI | IG | CHI

Apply appropriate global item weightings. One `TERM` weight exists for each term. If no `TERM` weight is applied, the default weight is 1 for each term.

**NORMAL**

$$w_i = \frac{1}{\sqrt{\sum_j f_{ij}^2}}$$

**GFIDF** (Global Frequency Inverse Document Frequency)

$$w_i = \frac{g_i}{d_i}$$

**IDF** (Inverse Document Frequency)

$$w_i = \log_2 \left( \frac{n}{d_i} \right) + 1$$

## ENTROPY

$$w_i = 1 + \sum_j \frac{p_{ij} \log_2(p_{ij})}{\log_2(n)}$$

For the above equations,

- $f_{ij}$  is the frequency of term  $i$  in document  $j$ .
- $d_i$  is the number of documents in which term  $i$  appears.
- $g_i$  is the number of times that term  $i$  appears in the whole document collection.
- $n$  is the number of documents in the collection.
- 

$$p_{ij} = \frac{f_{ij}}{g_i}$$

The final three weights, MI, IG, and CHI are category-dependent weightings and require that a target variable be specified with the TARGET option in the TMUTIL procedure.

MI (Mutual Information)

$$w_i = \max_{C_k} \left[ \log \left( \frac{P(t_i, C_k)}{P(t_i) P(C_k)} \right) \right]$$

IG (Information Gain)

$$w_i = - \sum_k P(C_k) \log P(C_k) + P(t_i) \sum_k P(C_k|t_i) \log P(C_k|t_i) \\ + P(\bar{t}_i) \sum_k P(C_k|\bar{t}_i) \log P(C_k|\bar{t}_i)$$

CHI

the value of the chi-squared test statistic for a one-way table that consists of the number of documents containing the term for each level of the target variable.

NONE

$$w_i = 1$$

For the above equations,

- $t_i$  is the  $i$ th term
- $\bar{t}_i$  indicates the absence of term  $t_i$

□

$$P(t_i) = \frac{d_i}{n}$$

- $C_k$  is the set of documents assigned to category  $k$ .
- $P(C_k)$  = percentage of documents belonging to category  $C_k$ .
- $P(t_i, C_k)$  percentage of documents containing both  $t_i$  and belonging to category  $C_k$ .
- $P(C_k|t_i)$  is the probability that term  $i$ , when it occurs, belongs to a document in the set  $C_k$ .

---

## SYN Statement

**SYN** <option(s)>;

The SYN statement is used to set and unset equivalent terms. While it is possible that some synonyms were assigned on the input KEY data set via the PARENT variable, the SYN statement enables the user to add additional synonyms or remove synonyms interactively. Assigning synonyms with the SYN statement requires that either the PARENTID and CHILDLIST option are used together or that a SYNDATA data set be used. The default behavior is to assign the children to the parents; however, if, in addition, the UNSET parameter is used, the specified terms are removed from their assigned equivalent group.

### **CHILDLIST | TERMLIST=<list of ints>**

Specifies a list of positive integers indicating the terms to be used as children. This option is used in combination with the PARENT variable. The default behavior will assign all terms in the CHILDLIST to the PARENT. If UNSET is used, the children will be removed from any existing assignment.

### **PARENT | PARENTID=*N***

A positive integer that identifies the term that is to function as the parent. When the UNSET option is used, TMUTIL will ungroup any children currently assigned to this parent. By using the PARENT and CHILDLIST options, only one equivalent group can be set at a time.

### **SYNDATA=<term id data set>**

Data set ID pairs that identify which terms are to be treated as children of representative terms. When a parent-child relationship is set, the child frequencies will be attributed to the parent term. The Synonym list has two required variables, `_TERMNUM_` and PARENT. Using the SYNDATA command, you can create different equivalent groups, each with its own parent and children, at one time.

`_TERMNUM_`      A positive integer indicating the ID of the child term.

PARENT            A positive integer indicating the ID of the parent term.

When the UNSET option is used, TMUTIL will unset the pairs as equivalent rather than assign them.

### **UNSET**

Unsets the synonym relationship between children and the parent. This option is used in combination with a SYNDATA data set or the PARENT and CHILDLIST. If a SYNDATA data set is input, the option takes every child-parent pair and unsets

them. If the term is a parent and the child is a missing value, all children of the parent are removed as synonyms. If the term number is a child and the parent is missing, then the term is removed as a child of its parent.

---

## SELECT Statement

**SELECT** *<option(s)>*;

The SELECT statement enables you to keep and drop terms. This process is similar to filtering, but the results cannot be undone generically with an undo option. Instead the actual term numbers must be submitted with KEEP="Y" in order to undo them.

**DROP | DROPLIST**=*<list of ints>*

Same behavior as for the DROPDATA option except inputs are on the command line as a list of positive integers.

**DROPDATA**=*<term id data set>*

This option changes the KEEP status from kept to dropped by passing in a data set of term indices. The required variable on the data set is either `_TERMNUM_`, `KEY`, or `INDEX`. The option will change all specified term numbers from KEEP="Y" to KEEP="N". The option has no effect on terms that are listed that are already dropped or filtered.

**KEEP | KEEPLIST**=*<list of ints>*

Same behavior as for the KEEPDATA option except inputs are on the command line as a list of positive integers.

**KEEPDATA**=*<term id data set>*

This option changes the KEEP status from dropped to kept by passing a data set of term indices. The required variable on the data set is either `_TERMNUM_`, `KEY`, or `INDEX`. The option will change all specified term numbers from KEEP="N" to KEEP="Y". The option has no effect on terms that are listed that are already kept or filtered.

**REDUCEF**=*X*

Keep only terms that are in at least X docs. Can be used in conjunction with REDUCEW.

**REDUCEN**=*X*

Keep only the X highest weighted terms. This option enables the user to control exactly how many terms are kept. If fewer than X kept terms exist, then the option will have no effect. This option cannot be used with REDUCEW or REDUCEF.

**REDUCEP**=*X*

Keep only terms that belong to at most X% of the documents. This option is for removing terms that occur too frequently in your collection.

**REDUCEW**=*X*

Keep only terms that are weighted higher than or equal to X. Can be used in conjunction with REDUCEF.

---

## Examples: TMUTIL Procedure

---

### Example 1: PROC TMUTIL and PROC DOCPARSE

PROC TMUTIL was designed to be used immediately following PROC DOCPARSE. The primary input to TMUTIL is the out data set from DOCPARSE. If parent information exists on the key data set, as in this PROC DOCPARSE call, then the key should also be passed into PROC TMUTIL. A CONTROL statement, documented below, is required for every call to PROC TMUTIL.

```
proc docparse data=SAMPSIO.abstract
  stemming=yes
  tagging=yes
  key=abstractkey
  out=abstractout;
  var text;
run;

proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
  output out=newout key=newkey;
run;
```

---

### Example 2: CONTROL Statement

The following examples show the four ways that the CONTROL statement can be used. The first three calls in this example demonstrate using the proc interactively. The first time tmutil is called, the init option is used. Then, additional calls can be made that use neither option. The third proc call below uses the release option, which clears the memory of all previous work. Finally, in the fourth proc call, an example is presented where the proc is not used interactively. The init and release option are used at the same time. In this case, the memloc variable is not necessary because the memory is not being stored or retrieved from the macro variable.

```
/* initial allocation of memory */
proc tmutil data=;
  control init memloc="here";
run;

/* subsequent calls reconnect to the memory */
proc tmutil;
  control memloc="here";
;
run;

/* release the memory */
proc tmutil;
  control memloc="here" release;
```

```

run;

/* init and release on same call */
proc tmutil data=;
  control init release;
run;

```

---

### Example 3: OUTPUT Statement

The following example outputs the term-doc frequency table (OUT data set), the term list with statistics (KEY data set), the terms that have been stopped (STOP data set), and the Synonym assignments (SYN data set).

```

proc docparse data=SAMPSIO.abstract
  stemming=yes
  tagging=yes
  key=abstractkey
  out=abstractout;
  var text;
run;
proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
  output out=newout key=newkey syn=newsyn stop=newstop;
run;

```

The following example outputs the term-doc frequency table (OUT data set), the term list with statistics (KEY data set), the terms that have been stopped (STOP data set), and the Synonym assignments (SYN data set).

```

proc docparse data=SAMPSIO.abstract
  stemming=yes
  tagging=yes
  key=abstractkey
  out=abstractout;
  var text;
run;
proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
  output out=newout key=newkey syn=newsyn stop=newstop;
run;

```

---

### Example 4: CONTROL and OUTPUT Statements

Example 4 parses with a stop list. The stopped terms are still passed along to tmutil in the first call to tmutil. This enables the user to change his or her mind about stopping the terms at a later time. However, it also forces extra computation and storage to be done by tmutil. On the second call to tmutil, the REINDEX option is used and re-indexed out and key data sets are output. Finally, on the last call, the re-indexed sets are read back into PROC TMUTIL.

```

proc docparse data=SAMPSIO.abstract
  stemming=yes
  tagging=yes
  key=abstractkey
  stop=sashelp.stoplst
  out=abstractout;
  var text;
run;

proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" ;
run;

proc tmutil ;
  control memloc="abstract" release;
  output out=reindexedout key=reindexedkey reindex;
run;

proc tmutil data=reindexedout key=reindexedkey;
  control init memloc="reindexed" ;
run;

```

---

## Example 5: WEIGHT Statement

The following example follows the PROC DOCPARSE code given above and outputs weighted versions of the term-doc frequency table (OUT data set) and the term list with statistics (KEY data set). The weighting chosen is the default for Text Miner, log-entropy.

```

proc docparse data=SAMPSIO.abstract
  stemming=yes
  tagging=yes
  key=abstractkey
  out=abstractout;
  var text;
run;

proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
  weight cellwgt=log termwgt=entropy;
  output out=weightedout key=weightedkey;
run;

```

---

## Example 6: SYN Statement

There are two ways to set synonyms. For the following example, you want to treat "airplane/Noun" (key=8733) as the representative term for "aircraft/Noun" (key=2401) and "airline/Noun"(key=8000). Both are demonstrated below. For both examples, we demonstrate unsetting the synonym that was just set.

```

proc docparse data=SAMPSIO.abstract
  stemming=yes
  tagging=yes
  key=abstractkey
  out=abstractout;
  var text;
run;

proc tutil data=abstractout key=abstractkey;
  control init memloc="abstract" ;
  syn parent=8733 childlist=2401 8000;
  output out=synout key=synkey;
run;

proc tutil data;
  control memloc="abstract" release;
  syn parent=8733 childlist=2401 8000 unset;
  output out=synout key=synkey;
run;

data synonyms;
INPUT _TERMNUM_ PARENT;
datalines;
2401 8733
8000 8733
;
run;

proc tutil data=abstractout key=abstractkey;
  control init memloc="abstract" ;
  syn syndata=synonyms;
  output out=synout key=synkey;
run;

proc tutil;
  control memloc="abstract" release;
  syn syndata=synonyms unset;
  output out=synout key=synkey;
run;

```

---

## Example 7: FILTER Statement

In this example "efficient/Adj" (key=1), "cross/Prop" (key=2), and the last document (key=1238) on the Abstract data set are filtered from use using the command line. The second proc call undoes the filtering and releases.

```

proc docparse data=SAMPSIO.abstract
  stemming=yes
  tagging=yes
  key=abstractkey
  out=abstractout;
  var text;
run;

```

```

proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" ;
  filter termlist=1 2 doclist=1238 comp;
  output out=filterout key=filterkey;
run;

proc tmutil ;
  control memloc="abstract" release;
  filter undo;
  output out=out key=key;
run;

```

In the following example, the same filter as above is applied as an input data set. Then another filter is applied, leaving an empty set of terms and docs.

```

INPUT _TERMNUM_ @@;
datalines;
1
2

;
run;

data docfilt;
INPUT _DOCUMENT_ @@;
datalines;
1238

;
run;

proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
  filter termdata=termfilt docdata=docfilt comp;
  output out=filterout key=filterkey;
run;

proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
  filter termdata=termfilt docdata=docfilt;
  output out=emptyout key=emptykey;
run;

```

The last example in this section demonstrates filtering out in the highest 100 weighted terms:

```

proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
  weight cellwgt=log termwgt=entropy;
  filter reduceN =100;
  output out=topNout key=topNkey;
run;

```

---

## Example 8: SELECT Statement

In the following example, the first two terms are dropped with the dropdata statement. Then they are included again with the keeplist statement.

```
proc docparse data=SAMPSIO.abstract
  stemming=yes
  tagging=yes
  key=abstractkey
  out=abstractout;
  var text;
run;

data dropterm;
  INPUT _TERMNUM_ @@;
  datalines;
  1
  2
  ;
run;

proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" ;
  select dropdata=dropterm;
  output out=dropout key=dropkey;
run;

proc tmutil ;
  control memloc="abstract" release;
  select keeplist= 1 2;
  output out=keepout key=keepkey;
run;
```

The last example in this section demonstrates dropping all but the terms weighted greater than or equal to .25:

```
proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
  weight cellwgt=log termwgt=entropy;
  select reduceW =.25;
  output out=reduceWout key=reduceWkey;
run;
```

Notice that on the KEY data set, any term with a weight of larger than .25 has been set to keep="N" and these terms do not appear on the OUT data set.

---

## Example 9: FACTOR Statement

```
proc tmutil data=abstractout key=abstractkey;
  control init memloc="abstract" release;
```

```
output out=newout key=newkey doc=svdabstract;  
factor type=svd k=3;  
run;
```