Text Mining in JMP with R

Andrew T. Karl, Senior Management Consultant, Adsurgo LLC
Heath Rushing, Principal Consultant and Co-Founder, Adsurgo LLC

1. Introduction

A popular rule of thumb suggests that 80% of data in most organizations is unstructured, such as text. Text mining is the process of finding interesting and relevant information from this unstructured data and determining if there are any meaningful relationships by transforming it into a structured format and applying classical multivariate statistical techniques. Many companies use this methodology on a daily basis for pattern discovery (e.g. warranty analysis, electronic medical records analysis) and predictive modeling (e.g. insurance fraud) using text from various sources such as email, survey comments, incident reports, free form data fields, websites, research reports, blogs, and social media. For these frequent users, SAS offers a comprehensive text mining tool, SAS Text Miner. However, the infrequency of use in some organizations does not warrant the cost associated with this software.

For example, some organizations within the Department of Defense need a basic, low-cost text mining capability to augment their existing analytical suite of tools: the US Army requires periodic text mining in their operational analysis of improvised explosive devices (IEDs). Additionally, the US Air Force requires occasional text mining to find information from pilot comments from operational tests. For such organizations, a JSL script accessing the R language offers a viable, low-cost alternative.

This paper highlights how text mining capabilities furnished by the R language may be wrapped into a JMP JSL script. R is an open source language for statistical computing. A variety of add-on packages may be downloaded from the Comprehensive R Archive Network (CRAN) to supplement the base R system. Using these packages (such as “tm”), R is capable of gathering a collection of documents into a corpus and then building a document term matrix from that corpus. R also supports sparse matrix algebra, which is necessary for text mining.

We refer readers elsewhere for a detailed description of the principles of text mining, also known as natural language processing. A nice introduction to text mining is provided by Weiss, S., et al. (2009) *Text Mining: Predictive Methods for Analyzing Unstructured Information*. We will illustrate our script with a data set on National Transportation Safety Board (NTSB) airplane accident reports provided by Miner, G., et al. (2012) *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*. We will assume a basic knowledge of these topics and focus on presenting these capabilities through JMP with R.

2. What is text mining?

By text mining, we refer here to the process of reducing a collection of documents (also known as a corpus) into a document-term matrix (DTM), with one row for each document and one column for each word that appears in the corpus. Once represented in this form, the corpus may be analyzed using existing data-mining methods (with some modifications), treating documents as observations and words as variables. This “bag-of-words” approach assumes that the order that the words in a document appear

in, as well as their parts of speech, may be ignored. A major challenge in text mining problems is that the DTM is often extremely large; however, due to the relatively infrequent occurrence of most words, the DTM is also sparse, allowing it to be stored efficiently.

In some applications, there are more terms present in a corpus than documents, causing problems for some model building routines. Even in cases where there are more documents than terms, the large number of terms will slow down the model building process for some procedures. Since many of the terms are likely to be irrelevant, the larger number of irrelevant terms will increase the variance of the estimates. A rank-reduced singular value decomposition (SVD) may be applied to the DTM to produce a matrix with fewer columns. For example, a DTM with ten thousand words may be reduced to a matrix with only 50 columns. These 50 columns are formed by taking linear combinations of the original 10,000 columns in a way that preserves as much of the original information as possible. The smaller matrix resulting from the SVD is produced by default with our JMP script. While the DTM may be used directly with either supervised or unsupervised learning methods (with some special modifications to standard methods), importing this matrix into JMP would require that it be treated as a dense matrix, which is not feasible for larger applications (though this option is provided by the script).

In some cases, it may be important to be able to predict responses for future observations. For example, a logistic regression may be used to predict whether or not an insurance claim is fraudulent based on the written report filed by the agent. This model would depend on vectors from the SVD of the DTM resulting from the training corpus of claims. Special care must be taken when handling new observations: they need to be transformed to the space spanned by the SVD on the training data.

Popular text mining questions include, “which documents are most similar?” and, “which documents are most similar to this particular document?” There are a couple major advantages of using vectors from the SVD of the DTM rather than the DTM itself to answer these questions. First, the DTM is usually large enough that it cannot be manipulated without accounting for its sparse structure. JMP does not offer this capability. Secondly, the entries of the DTM are non-negative, and we are more concerned about overlap of positive entries (shared words between two documents) than we are about overlap of zero entries (words that are absent from both of two documents). This requires the use of the cosine metric, which is not available within the JMP Cluster platform. By contrast, the document summaries resulting from the SVD may be analyzed with the Euclidean metric. A third advantage is that the dimensionality reduction provided by the SVD eliminates redundant/irrelevant variables that can often cause problems with clustering algorithms.

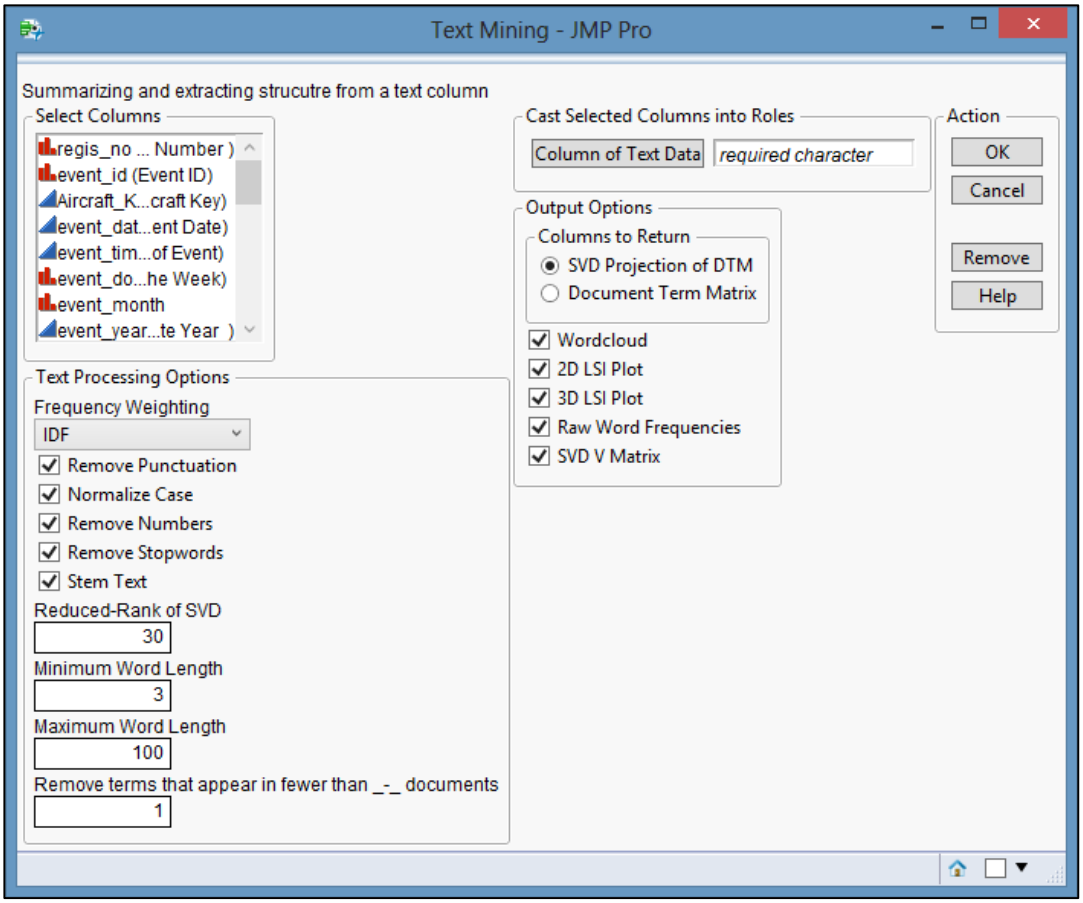
Just as documents may be clustered, the output of the SVD may be used to cluster terms. This can detect which words occur commonly together throughout the corpus. More details about the SVD appear in the Appendix.

The ability to perform text mining in JMP should provide opportunities to explore columns of unstructured text information that would otherwise be ignored. The visual exploration of summaries of a collection of documents – including latent semantic analysis, raw counts, and clustering on the documents and terms – provides a low-cost ability to search for new patterns and identify previously unknown relationships in your data. Thanks to the efficient routines for text mining and matrix algebra provided by R, this JMP script scales well to large collections of long documents.

3. JMP Script and Application

To illustrate our script, we will analyze a collection of NTSB accident reports that are available from Miner, G., et al. (2012) *Practical Text Mining and Statistical Analysis for Non-structured Text Data Application.* The accident reports contain columns of structured information along with columns of unstructured text. The structured information includes the time and location of the accident, as well as whether there were any fatalities. The text columns contain the written accounts of the accidents along with succinct descriptions of the causes of the accidents. For our text mining examples, we will focus on the column "narr_cause," which contains the equivalent of 209 single-spaced pages (with one blank line between each report) using Times New Roman 12 point font. There are a total of 84370 words, with an average of 26 words/report for 3235 reports.

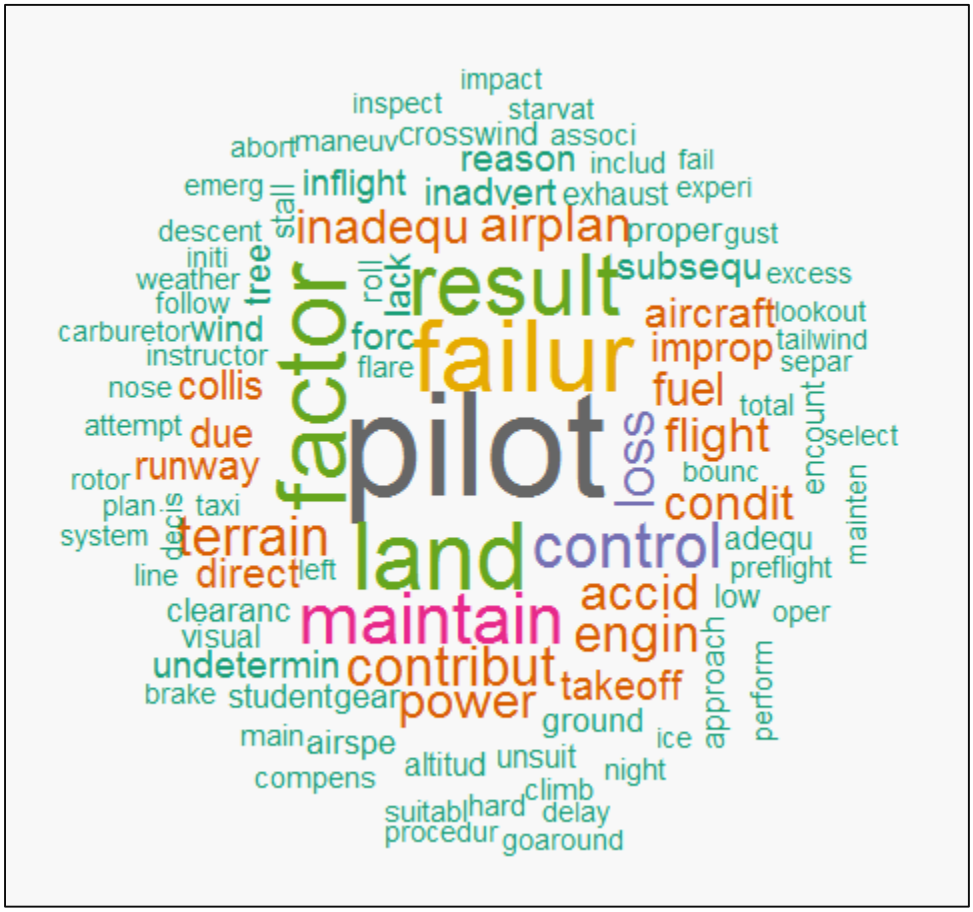
Regarding the scalability of the program, we note that the longer accounts appearing in the "narr_accf (NTSB Final Narrative (6120.4).)" column contain the equivalent of 810 single-spaced pages (with one blank line between each report) using Times New Roman 12 point font. There are a total of 496542 words, with average of 153 words/report for 3235 reports. The R code takes 15 seconds to process the narr_cause column and 35 seconds to process the narr_accf column.



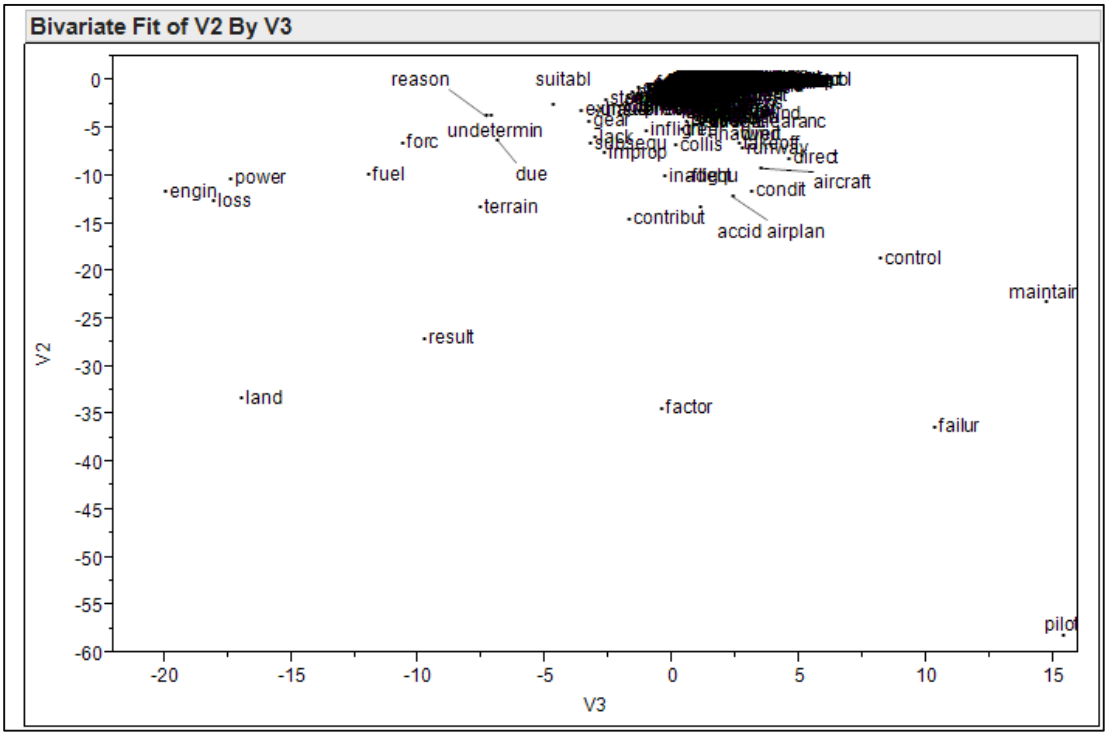
- **Select Columns area** – The script assumes that the documents appear in a single row of a data table, with one document per row. Once the appropriate column is selected, clicking the **Column of Text Data** button will instruct the script to use that column.
- **Frequency Weighting –** The DTM is constructed using the counts of each term within each document. Depending on the application, a transformation of these raw term frequencies may yield better results.
    - **IDF** – Inverse document frequency. De-emphasizes words that appear in many documents.
    - **Document Frequency** – raw counts
    - **Binary** – The non-zero components of the document frequency DTM are replaced by 1.
    - **Ternary –** The components of the document frequency DTM with values greater than 2 are replaced by 2.
    - **Log –** The non-zero components, *x*, of the document frequency DTM are replaced by $1 + \log(x)$.

Note: The next five options are executed in the order they are listed.

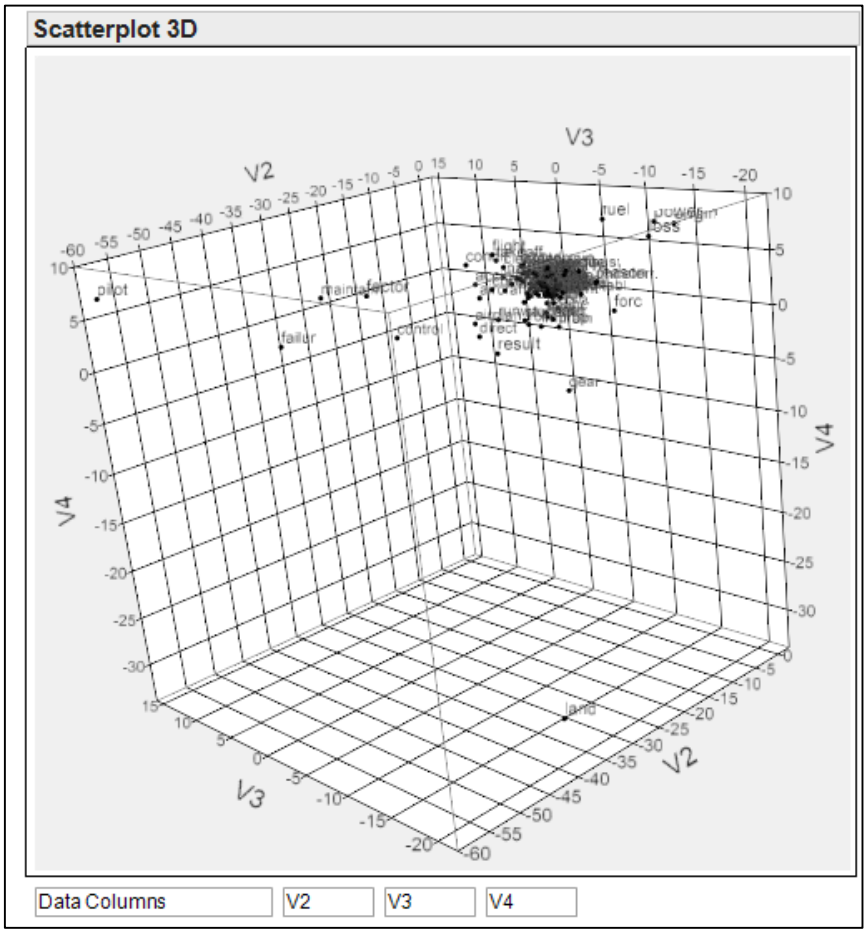
- **Remove Punctuation –** Removes punctuation from each document. If this option is not checked, the punctuation is attached to the word it appears next to. For example, the last word of the first sentence in this paragraph would be read as "document." Instead of "document".
- **Normalize Case** – converts the text of each document to all lowercase.
- **Remove Numbers** – Removes the numbers from each document in the corpus.
- **Remove Stopwords –** Removes commonly used words that are likely to appear in many documents. Since they are so common, they are not helpful in differentiating between the documents. These are stopwords from the SMART information retrieval system (http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop).
- **Stem Text** – reduce each word in the corpus to its base. For example, *jumped* and *jumping* would both be reduced to *jump*. See http://snowball.tartarus.org/ for details of the algorithm.
- **Reduced Rank of SVD** – To how many columns should the SVD reduce the DTM? This number is referred to as *s* in the Appendix.
- **Minimum Word Length** – Words shorter than this length will not be included in the DTM.
- **Maximum Word Length** – Words longer than this length will not be included in the DTM.
- **Remove terms that appear in fewer than _-_ documents** – "Terms" that only appear in a few documents are frequently typos.
- **Column of Text Data –** See the entry for **Select Columns area**.
- **Columns to Return** – Should the reduced form of the DTM ("SVD Projection of DTM") or the DTM itself ("Document Term Matrix") be returned?
- **Wordcloud** – Should a wordcloud be produced? The most frequent terms appear in the center of the cloud and are larger. The colors that the terms are printed in also changes as the frequency decreases. The wordcloud uses the DTM, and depends on the chosen Frequency weighting.



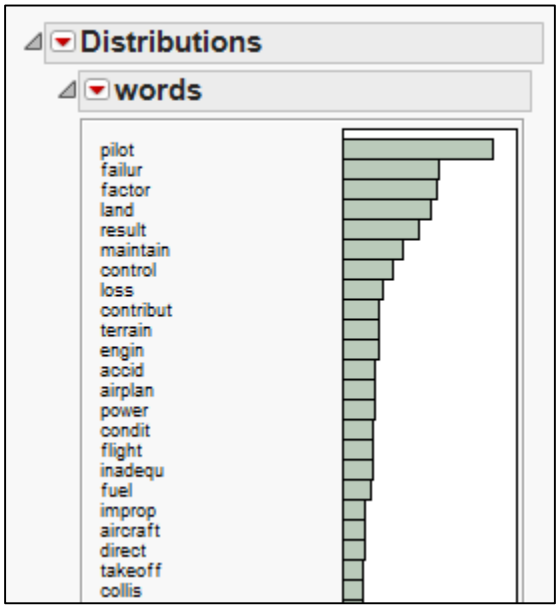
- **2D LSI Plot** –Terms that appear close together appear in the same document frequently.



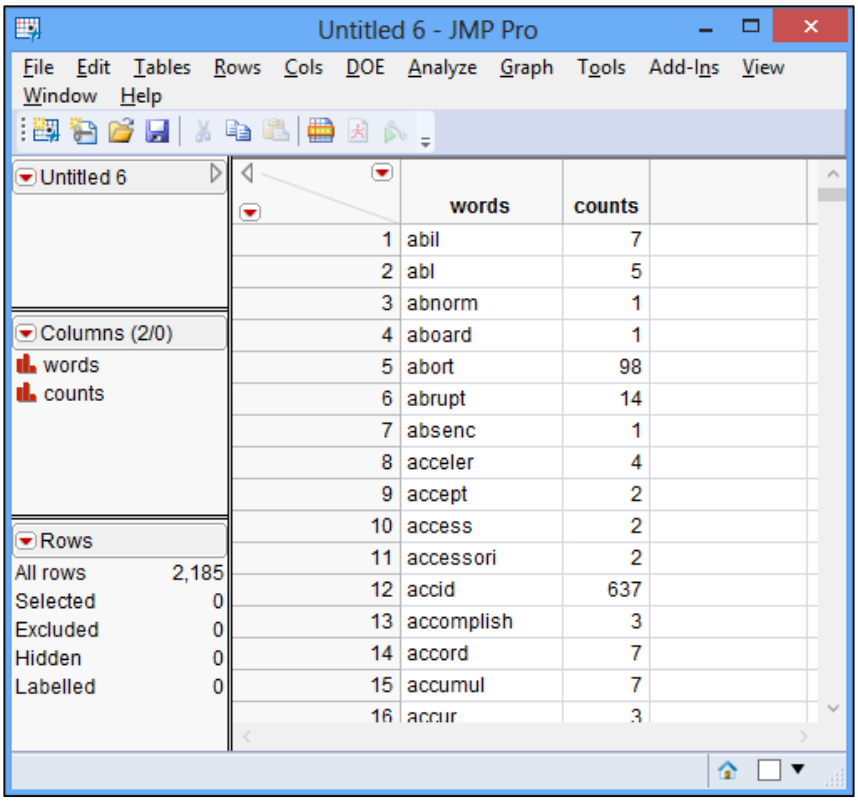
- **3D LSI Plot** –This plot may be rotated by clicking and dragging with the mouse. Terms that appear close together appear in the same document frequently.



- **Raw Word Frequencies** – Returns a plot of the word frequencies. These are raw term counts, and do not depend on the Frequency Weighting.

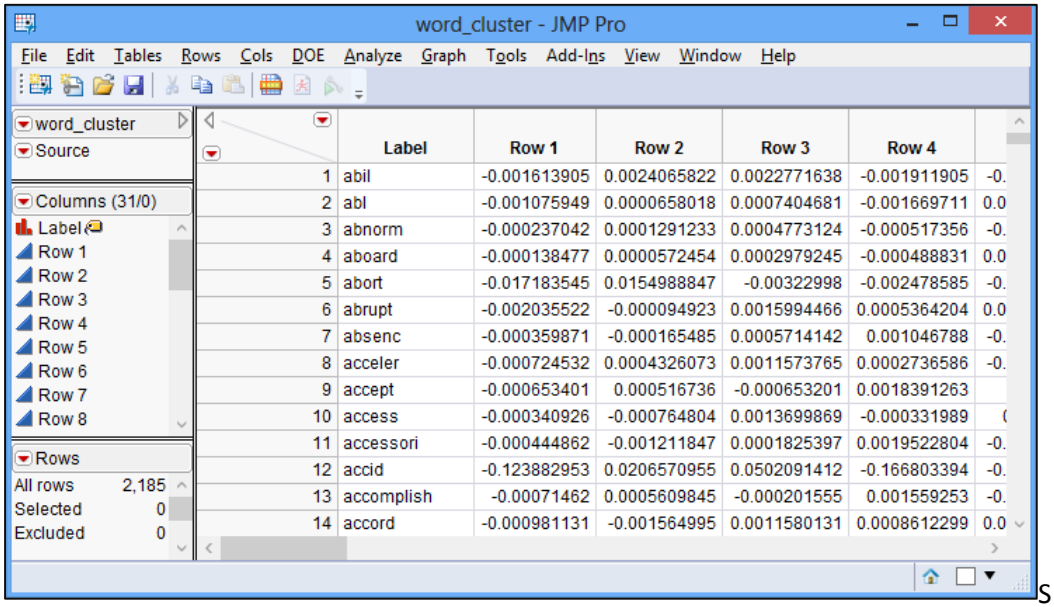


The counts may be obtained by clicking the red triangle next to Distributions and selecting **Script > Data Table Window**.



	words	counts
1	abil	7
2	abl	5
3	abnorm	1
4	aboard	1
5	abort	98
6	abrupt	14
7	absenc	1
8	acceler	4
9	accept	2
10	access	2
11	accessori	2
12	accid	637
13	accomplish	3
14	accord	7
15	accumul	7
16	accur	3

- **SVD V Matrix –** Returns a table containing a matrix from the SVD. This matrix may be used for finding clusters of terms, or for transforming new observations into the same space spanned by the SVD of the original DTM.

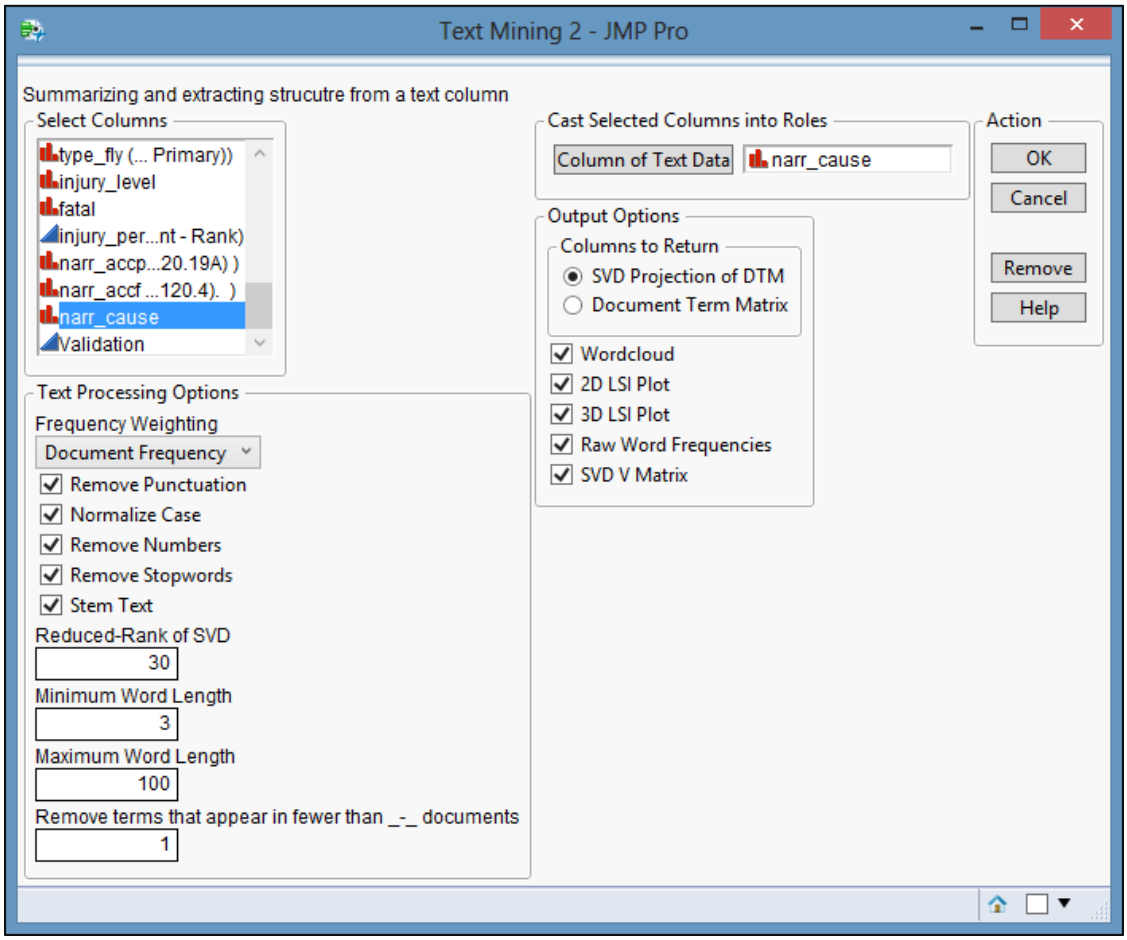


	Label	Row 1	Row 2	Row 3	Row 4	
1	abil	-0.001613905	0.0024065822	0.0022771638	-0.001911905	-0.
2	abl	-0.001075949	0.0000658018	0.0007404681	-0.001669711	0.0
3	abnorm	-0.000237042	0.0001291233	0.0004773124	-0.000517356	-0.
4	aboard	-0.000138477	0.0000572454	0.0002979245	-0.000488831	0.0
5	abort	-0.017183545	0.0154988847	-0.00322998	-0.002478585	-0.
6	abrupt	-0.002035522	-0.000094923	0.0015994466	0.0005364204	0.0
7	absenc	-0.000359871	-0.000165485	0.0005714142	0.001046788	-0.
8	acceler	-0.000724532	0.0004326073	0.0011573765	0.0002736586	-0.
9	accept	-0.000653401	0.000516736	-0.000653201	0.0018391263	
10	access	-0.000340926	-0.000764804	0.0013699869	-0.000331989	(
11	accessori	-0.000444862	-0.001211847	0.0001825397	0.0019522804	-0.
12	accid	-0.123882953	0.0206570955	0.0502091412	-0.166803394	-0.
13	accomplish	-0.00071462	0.0005609845	-0.000201555	0.001559253	-0.
14	accord	-0.000981131	-0.001564995	0.0011580131	0.0008612299	0.0

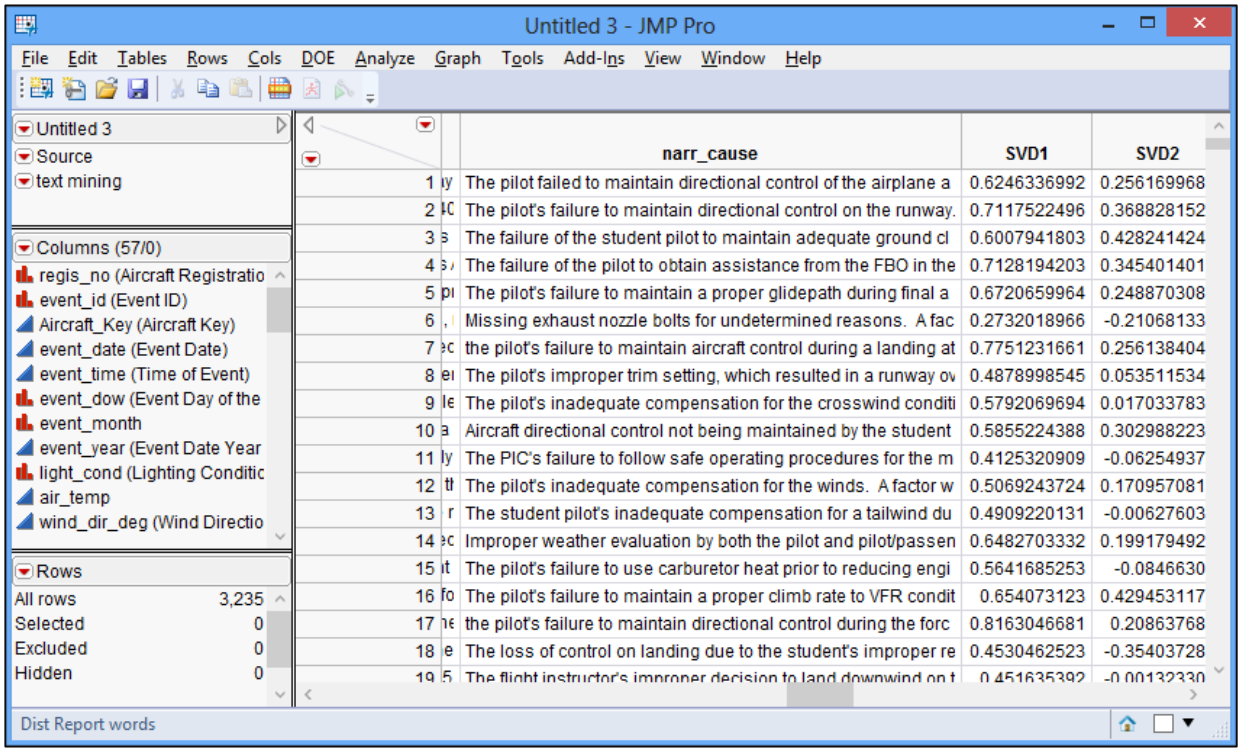
S

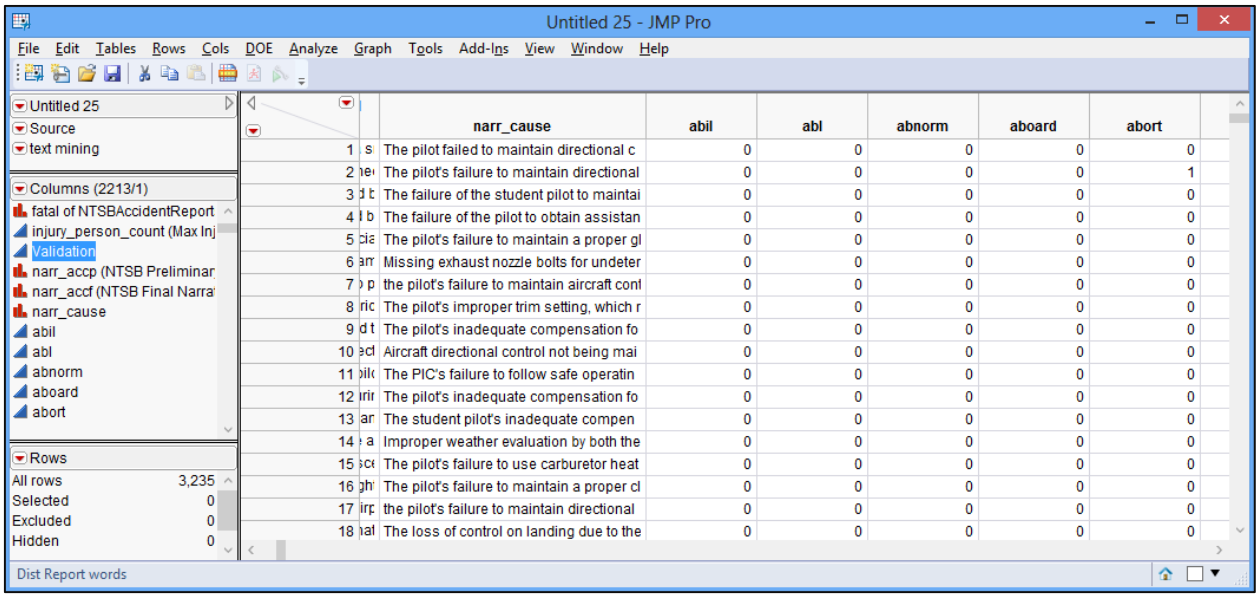
a. Supervised Learning

We first run the text mining script to extract 30 vectors from the SVD of the document frequency DTM for the NTSB data.



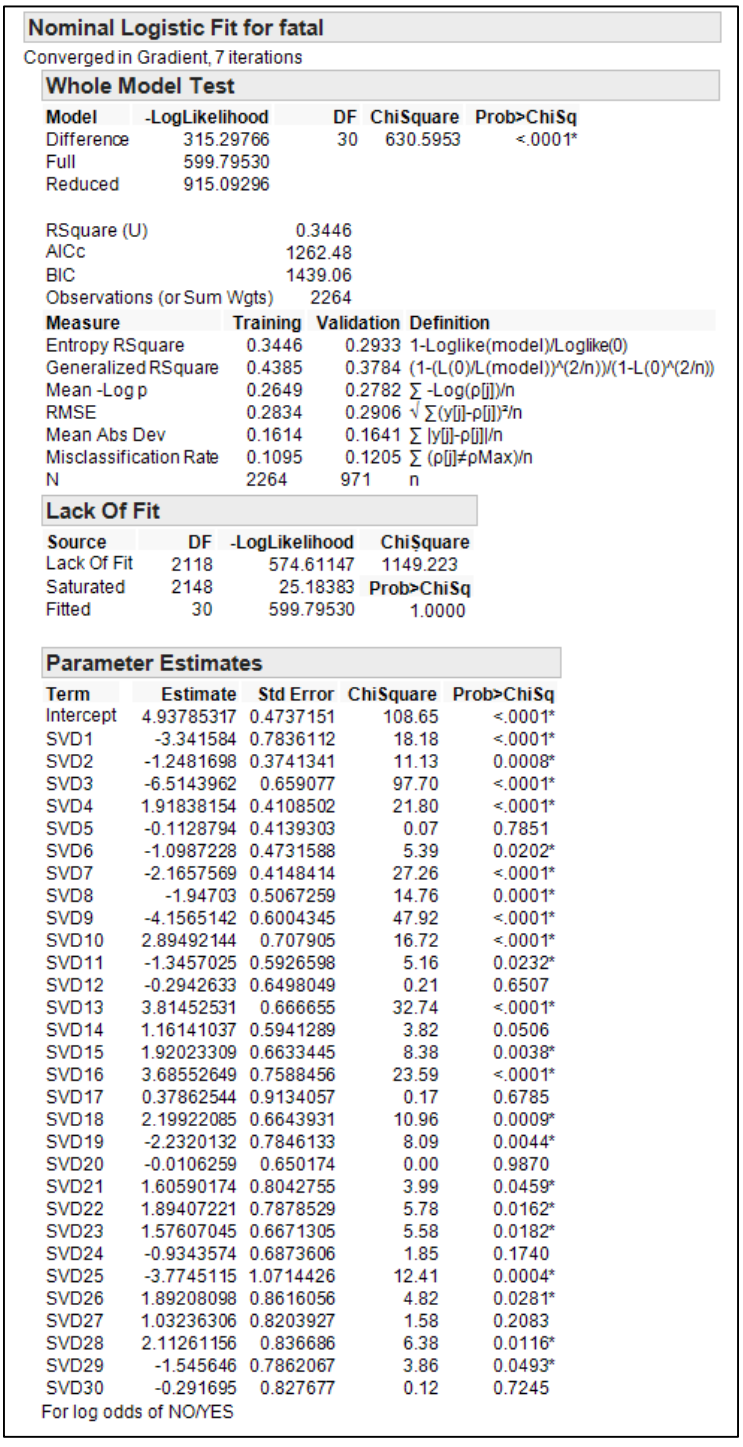
A new data table is created by appending the vectors from the SVD (*SVD1 – SVD30*) to the columns of the original table. Contrast this with the over 2000 additional columns required for the full DTM.





We will first fit a logistic regression (using the **Fit Model** platform) to *SVD1-SVD30* for whether or not the crash was fatal. This binary response is contained in the "fatal" column. We add a column, *Validation*, that contains 70% 0's and 30% 1's. The 1's correspond to observations that will be held out from the model building for the purpose of error assessment. Strictly speaking, in order to truly simulate the process of obtaining new data, we should first build the DTM for the training data, perform the SVD, and

transform the new DTM for the validation data into this space. For simplicity, we will here just hold out the validation data.



Nominal Logistic Fit for fatal

Converged in Gradient, 7 iterations

Whole Model Test

Model	-LogLikelihood	DF	ChiSquare	Prob>ChiSq
Difference	315.29766	30	630.5953	<.0001*
Full	599.79530			
Reduced	915.09296			

RSquare (U)	0.3446
AICc	1262.48
BIC	1439.06
Observations (or Sum Wgts)	2264

Measure	Training	Validation	Definition
Entropy RSquare	0.3446	0.2933	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.4385	0.3784	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.2649	0.2782	∑ -Log(ρ[j])/n
RMSE	0.2834	0.2906	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.1614	0.1641	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.1095	0.1205	∑ (ρ[j]≠ρMax)/n
N	2264	971	n

Lack Of Fit

Source	DF	-LogLikelihood	ChiSquare
Lack Of Fit	2118	574.61147	1149.223
Saturated	2148	25.18383	Prob>ChiSq
Fitted	30	599.79530	1.0000

Parameter Estimates

Term	Estimate	Std Error	ChiSquare	Prob>ChiSq
Intercept	4.93785317	0.4737151	108.65	<.0001*
SVD1	-3.341584	0.7836112	18.18	<.0001*
SVD2	-1.2481698	0.3741341	11.13	0.0008*
SVD3	-6.5143962	0.659077	97.70	<.0001*
SVD4	1.91838154	0.4108502	21.80	<.0001*
SVD5	-0.1128794	0.4139303	0.07	0.7851
SVD6	-1.0987228	0.4731588	5.39	0.0202*
SVD7	-2.1657569	0.4148414	27.26	<.0001*
SVD8	-1.94703	0.5067259	14.76	0.0001*
SVD9	-4.1565142	0.6004345	47.92	<.0001*
SVD10	2.89492144	0.707905	16.72	<.0001*
SVD11	-1.3457025	0.5926598	5.16	0.0232*
SVD12	-0.2942633	0.6498049	0.21	0.6507
SVD13	3.81452531	0.666655	32.74	<.0001*
SVD14	1.16141037	0.5941289	3.82	0.0506
SVD15	1.92023309	0.6633445	8.38	0.0038*
SVD16	3.68552649	0.7588456	23.59	<.0001*
SVD17	0.37862544	0.9134057	0.17	0.6785
SVD18	2.19922085	0.6643931	10.96	0.0009*
SVD19	-2.2320132	0.7846133	8.09	0.0044*
SVD20	-0.0106259	0.650174	0.00	0.9870
SVD21	1.60590174	0.8042755	3.99	0.0459*
SVD22	1.89407221	0.7878529	5.78	0.0162*
SVD23	1.57607045	0.6671305	5.58	0.0182*
SVD24	-0.9343574	0.6873606	1.85	0.1740
SVD25	-3.7745115	1.0714426	12.41	0.0004*
SVD26	1.89208098	0.8616056	4.82	0.0281*
SVD27	1.03236306	0.8203927	1.58	0.2083
SVD28	2.11261156	0.836686	6.38	0.0116*
SVD29	-1.545646	0.7862067	3.86	0.0493*
SVD30	-0.291695	0.827677	0.12	0.7245

For log odds of NO/YES

There is no evidence of lack-of-fit. Click the red triangle next to Nominal Logistic Fit for fatal and select **Confusion Matrix**.

Confusion Matrix

Actual		Predicted	Actual		Predicted
Training	**NO**	**YES**	**Validation**	**NO**	**YES**
NO	1885	63	NO	806	35
YES	185	131	YES	82	48

On the validation data, logistic regression achieved an accuracy of 88%, a precision of 58%, and a recall of 37%.

An artificial neural network (using the **Neural** platform) performs slightly better.

Model NTanH(3)NTanH2(3)

Training

fatal	Measures
Generalized RSquare	0.4719436
Entropy RSquare	0.3752491
RMSE	0.2772469
Mean Abs Dev	0.1585455
Misclassification Rate	0.1068905
-LogLikelihood	571.70511
Sum Freq	2264

Confusion Matrix

Actual		Predicted
fatal	**NO**	**YES**
NO	1892	56
YES	186	130

Validation

fatal	Measures
Generalized RSquare	0.3710908
Entropy RSquare	0.2869453
RMSE	0.2915425
Mean Abs Dev	0.1663743
Misclassification Rate	0.1143151
-LogLikelihood	272.58937
Sum Freq	971

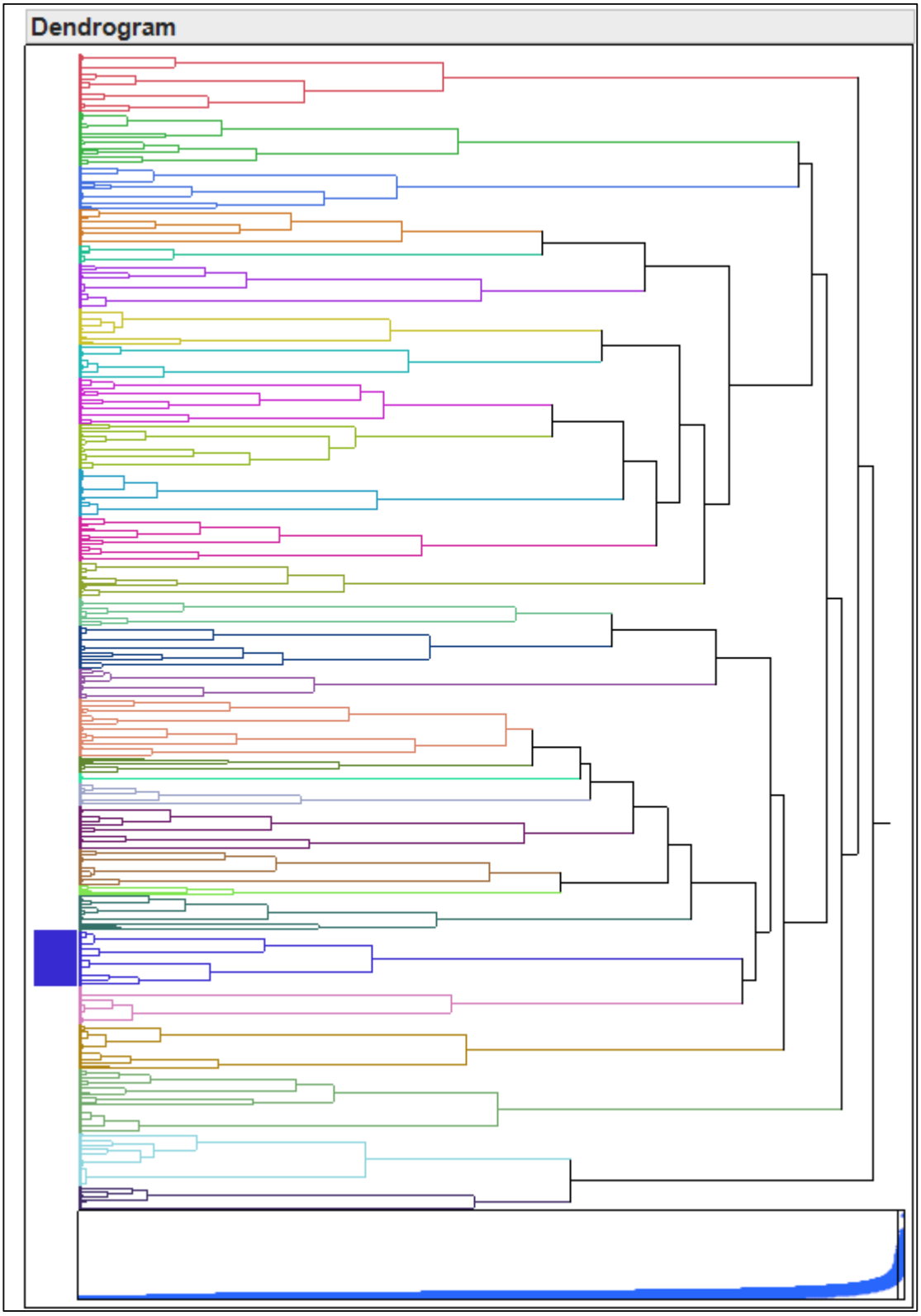
Confusion Matrix

Actual		Predicted
fatal	**NO**	**YES**
NO	811	30
YES	81	49

On the validation data, the neural network achieved an accuracy of 89%, a precision of 62%, and a recall of 38%.

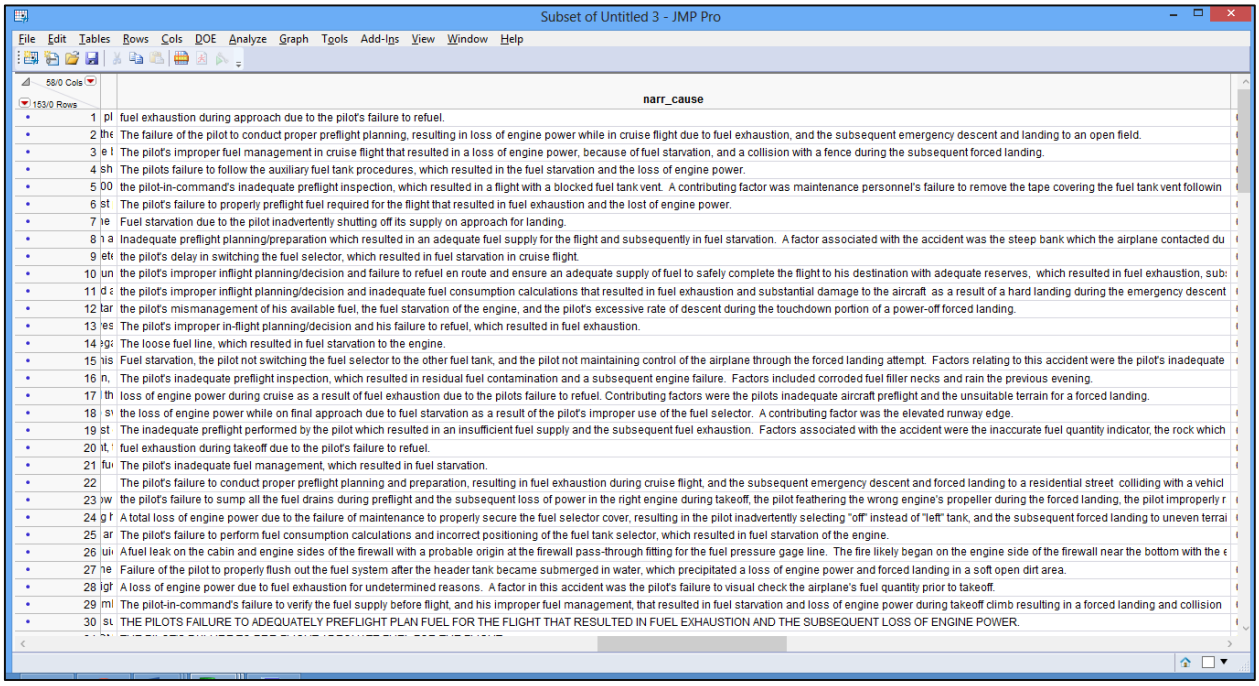
b. Clustering Documents

In some cases, we may not have a target response we need to predict. A common unsupervised learning problem for text mining involves finding clusters of similar documents. Using the **Cluster** platform, we find

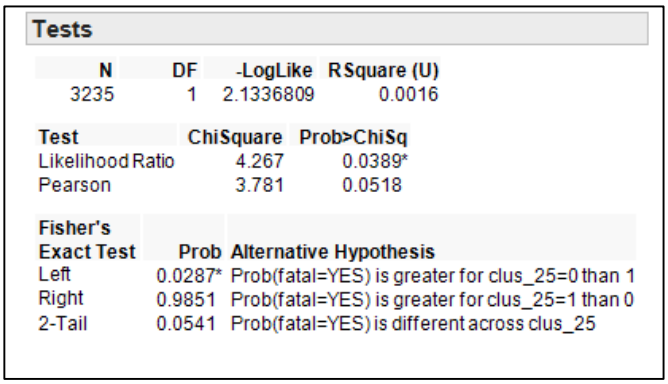


For illustration, we will examine the members of the blue cluster below indicated with the blue rectangle to the left of the histogram. The rows belonging to this cluster in the data table are automatically selected when the cluster is selected in the dendogram. The command **Table > Subset**

(click **OK**) creates a new table with only reports from this cluster. The first 30 (out of 153) of the reports from this cluster appear after the dendogram.



After reading through the reports, we see that this cluster contains accidents due to fuel exhaustion, usually due to pilot error. 8.5% of these accidents were fatal, compared with the 14% rate seen in the reports not contained in this cluster. Fisher's exact test indicates this difference is marginally significant, with a p-value of 0.0541.



Tests

N	DF	-LogLike	RSquare (U)
3235	1	2.1336809	0.0016

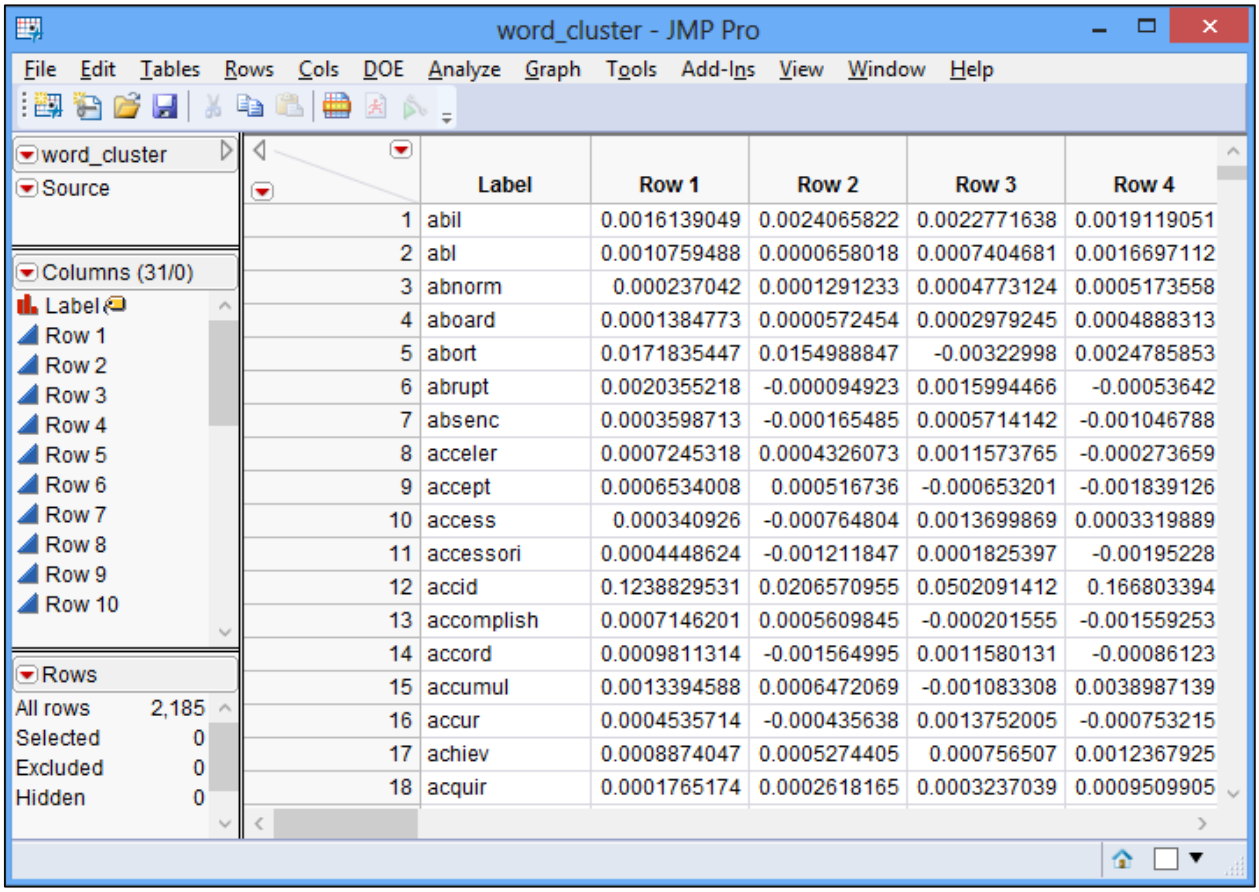
Test	ChiSquare	Prob>ChiSq
Likelihood Ratio	4.267	0.0389*
Pearson	3.781	0.0518

Fisher's Exact Test	Prob	Alternative Hypothesis
Left	0.0287*	Prob(fatal=YES) is greater for clus_25=0 than 1
Right	0.9851	Prob(fatal=YES) is greater for clus_25=1 than 0
2-Tail	0.0541	Prob(fatal=YES) is different across clus_25

Of course, our interpretation of the meaning of this cluster is subjective, which affects the appropriate interpretation of the hypothesis test for difference in proportion of fatal outcomes between clusters. Still, this example illustrates how text mining may be used for hypothesis formation when *a priori* subject matter knowledge is limited. If this finding were of interest to the NTSB, they could add a section to their accident report such as "Fuel Exhaustion a Factor?" which would allow for a retrospective analysis on fuel exhaustion in future reports. Though it would likely be difficult to get IRB approval for a randomized experiment to study the relationship between fuel exhaustion and fatal accidents, other hypothesize suggested by text mining will be amenable to such studies.

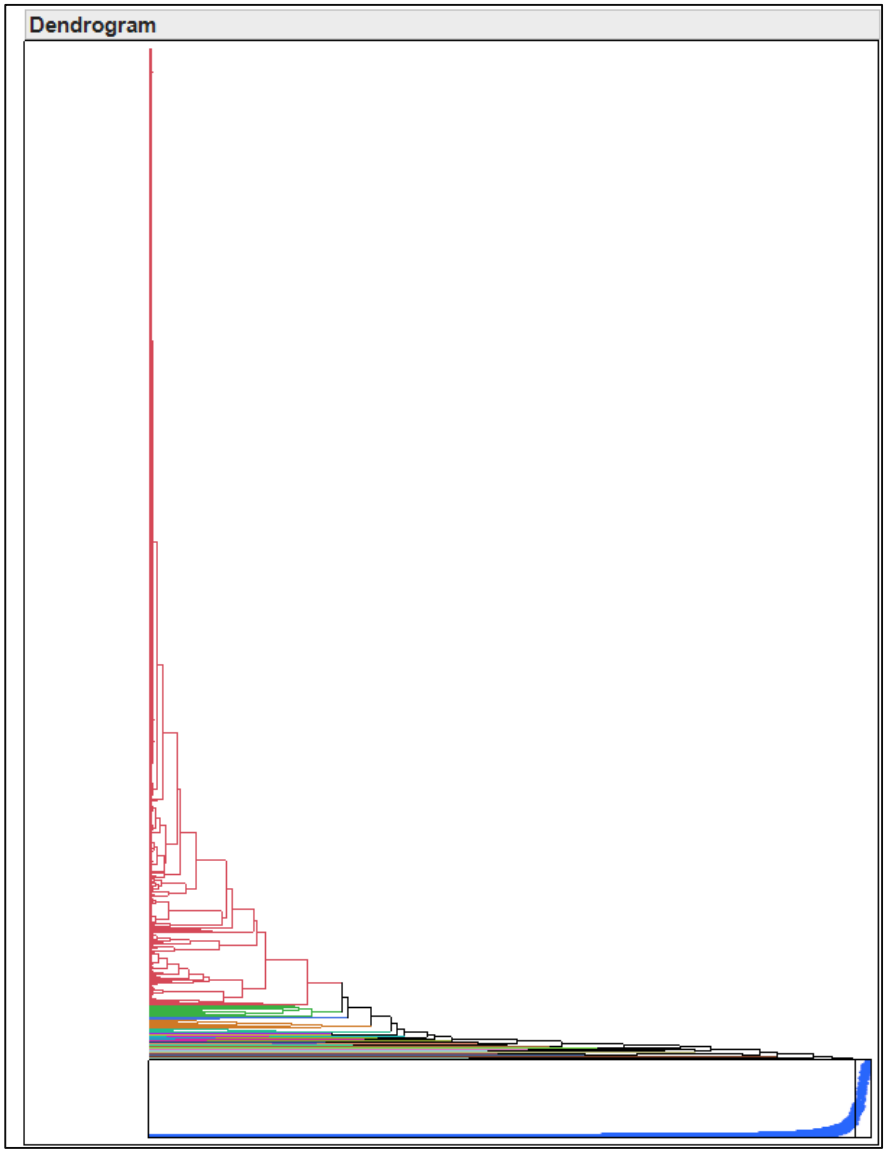
c. Clustering Terms

We perform a cluster analysis on the terms, again using the **Cluster** platform.

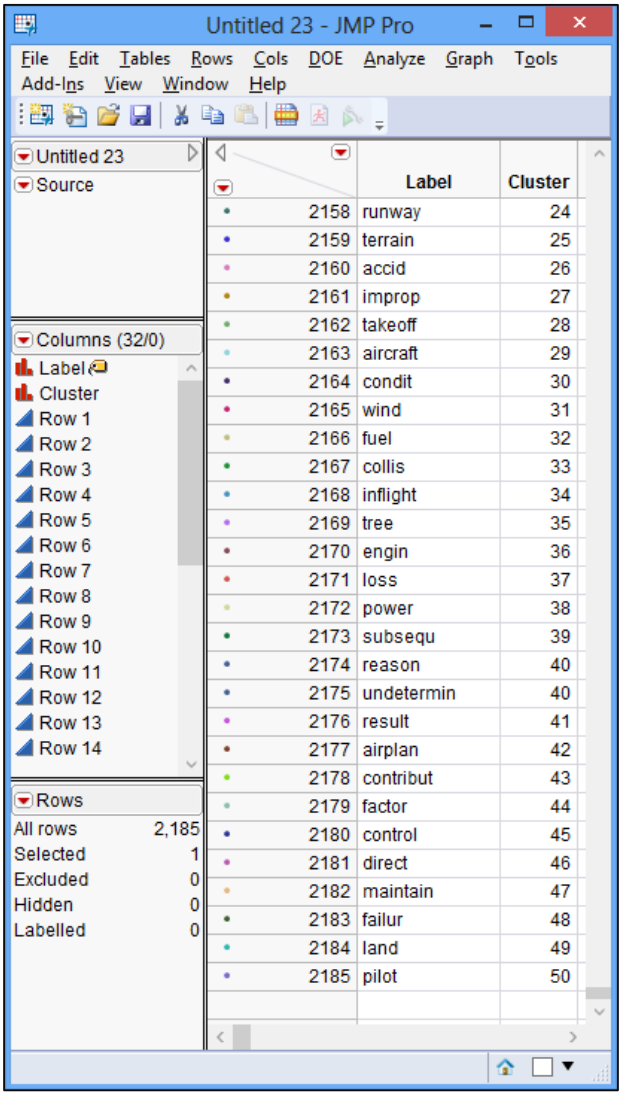


	Label	Row 1	Row 2	Row 3	Row 4
1	abil	0.0016139049	0.0024065822	0.0022771638	0.0019119051
2	abl	0.0010759488	0.0000658018	0.0007404681	0.0016697112
3	abnorm	0.000237042	0.0001291233	0.0004773124	0.0005173558
4	aboard	0.0001384773	0.0000572454	0.0002979245	0.0004888313
5	abort	0.0171835447	0.0154988847	-0.00322998	0.0024785853
6	abrupt	0.0020355218	-0.000094923	0.0015994466	-0.00053642
7	absenc	0.0003598713	-0.000165485	0.0005714142	-0.001046788
8	acceler	0.0007245318	0.0004326073	0.0011573765	-0.000273659
9	accept	0.0006534008	0.000516736	-0.000653201	-0.001839126
10	access	0.000340926	-0.000764804	0.0013699869	0.0003319889
11	accessori	0.0004448624	-0.001211847	0.0001825397	-0.00195228
12	accid	0.1238829531	0.0206570955	0.0502091412	0.166803394
13	accomplish	0.0007146201	0.0005609845	-0.000201555	-0.001559253
14	accord	0.0009811314	-0.001564995	0.0011580131	-0.00086123
15	accumul	0.0013394588	0.0006472069	-0.001083308	0.0038987139
16	accur	0.0004535714	-0.000435638	0.0013752005	-0.000753215
17	achiev	0.0008874047	0.0005274405	0.000756507	0.0012367925
18	acquir	0.0001765174	0.0002618165	0.0003237039	0.0009509905

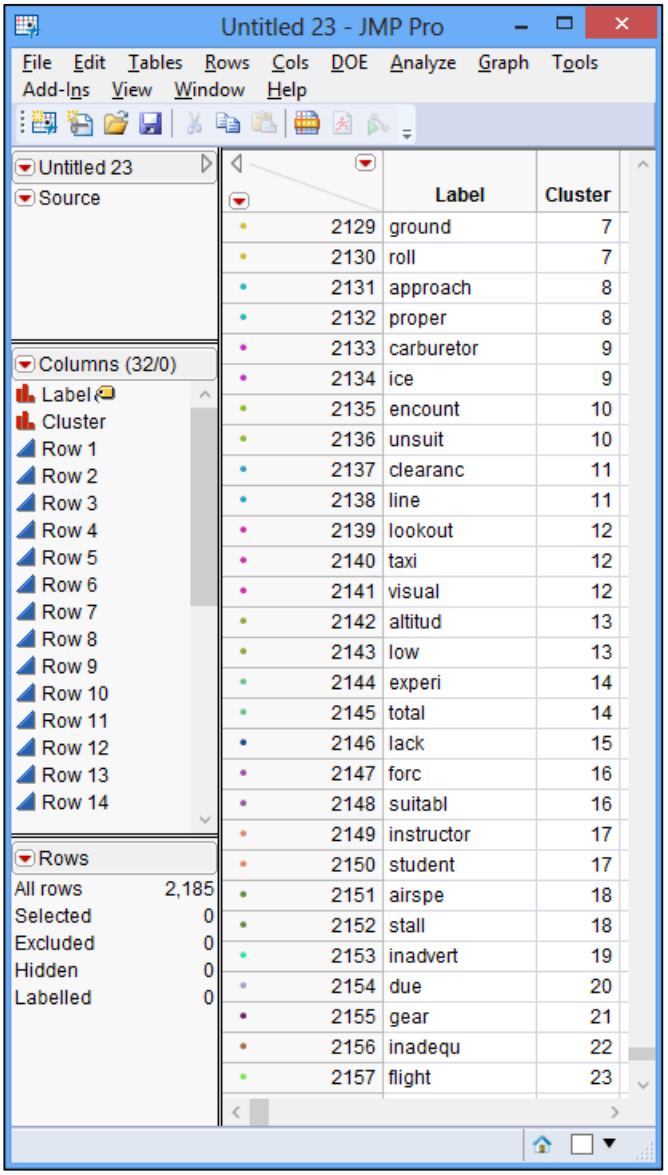
After setting the number of clusters to 50, it appears that there are a number of isolated terms that are grouped into the top (red) cluster, some clusters of only a few terms, and then several clusters consisting of a single term (apparent from the data table).

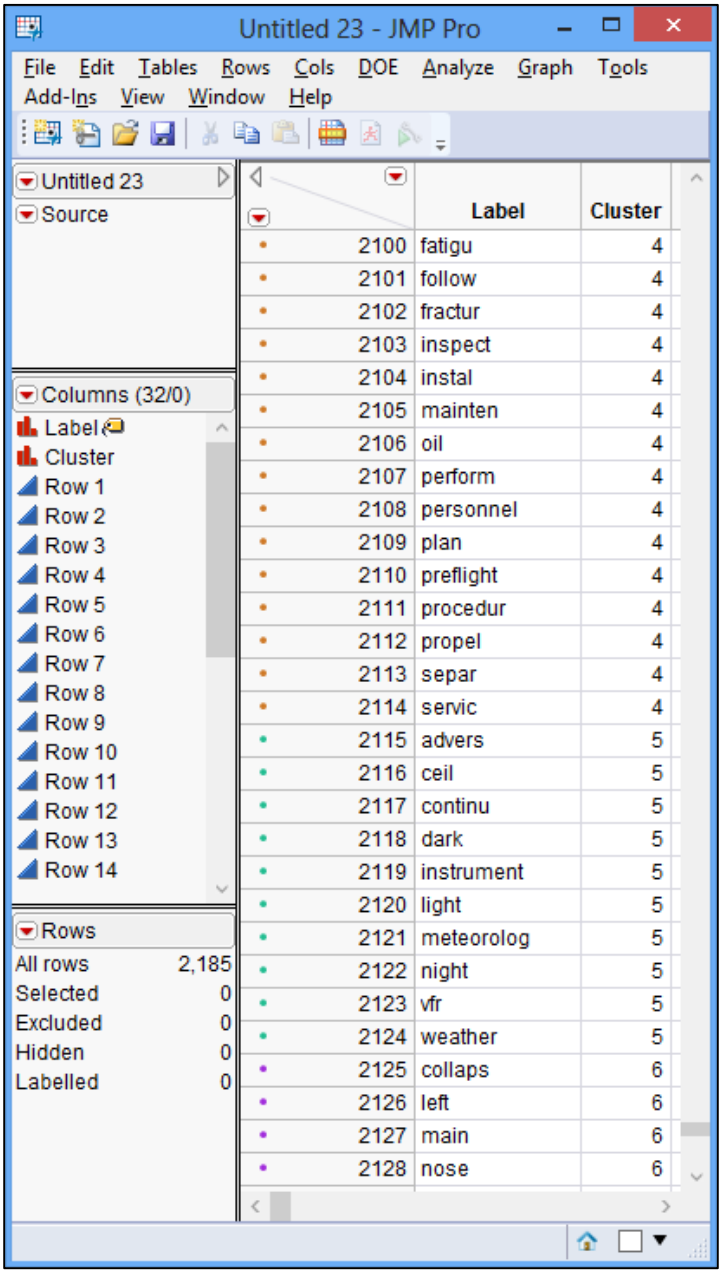


Clusters 24-50 are mostly single term clusters, except for cluster 40, which contains “reason” and “undetermin.”



We find several two- and three-term clusters (e.g. stall and airspe; ice and carburetor). These groupings would likely be unsurprising to aviation experts, but might be informative for others.


Untitled 23 - JMP Pro
File Edit Tables Rows Cols DOE Analyze Graph Tools
Add-Ins View Window Help
Untitled 23
Source
Columns (32/0)
Label
Cluster
Row 1
Row 2
Row 3
Row 4
Row 5
Row 6
Row 7
Row 8
Row 9
Row 10
Row 11
Row 12
Row 13
Row 14
Rows
All rows 2,185
Selected 0
Excluded 0
Hidden 0
Labelled 0
Label
Cluster
2129 ground 7
2130 roll 7
2131 approach 8
2132 proper 8
2133 carburetor 9
2134 ice 9
2135 encount 10
2136 unsuit 10
2137 clearanc 11
2138 line 11
2139 lookout 12
2140 taxi 12
2141 visual 12
2142 altitud 13
2143 low 13
2144 experi 14
2145 total 14
2146 lack 15
2147 forc 16
2148 suitabl 16
2149 instructor 17
2150 student 17
2151 airspe 18
2152 stall 18
2153 inadvert 19
2154 due 20
2155 gear 21
2156 inadequ 22
2157 flight 23



4. Appendix

The script returns a rank-reduced singular value decomposition (SVD) of the DTM X. The SVD factorization is

$$X \approx UDV^t,$$

where

- *U* is a dense *d* by *s* orthogonal matrix, where *s* is the rank of the SVD factorization (*s=1,…,min(d,w)*).
- *D* is a diagonal matrix with nonnegative entries.

- V^t is a dense *s* by *w* orthogonal matrix, where *s* is the rank of the SVD factorization (*s=1,…,min(d,w)*), and the superscript *t* indicates “transpose.”

The appropriate value of *s* is a matter of debate, and is application dependent. Smaller values of *s* represent a greater extent of dimensionality reduction at the cost of a loss of structure of the original DTM. In our examples, we use $s = 30$, though values anywhere from 30 to 500 are commonly used.