Toronto Area SAS Society (TASS)

September 14, 2018 | Toronto, ON
SAS Institute - Canada

An Easier and Faster Way to Untranspose a Wide File

Arthur Tabachneck
Thornhill, ON

Joe Matise
Chicago, IL

Matt Kastin
Chicago, IL

Gerhard Svolba
Wien, Austria

Toronto Area SAS Society

September 14, 2018

Presentation Overview

- What the %untranspose macro does
- How to use the macro
- Where to get the %untranspose macro (p.s., it's free!)
- Some useful techniques we used in creating the macro

Question

Have you ever needed to *untranspose* a wide file back to the long or not-quite-as-wide file that was used to create it?

For Example

You were given/sent the following file

income2015	income2016	income2017	expenses2015	expenses2016	expenses2017	debt2015	debt2016	debt2017
70000	75500	80000	60000	70000	81000	no	no	yes
50000	52000	55000	42000	53000	60000	no	yes	yes
80000	90000	99000	70000	75000	85000	no	no	no

label: Yearly Income

label: Yearly Expenses

label: Expenses>Income

and you needed to untranspose it to look like

id	year	income	expenses	debt
1	2015	70000	60000	no
1	2016	75500	70000	no
1	2017	80000	81000	yes
2	2015	50000	42000	no
2	2016	52000	53000	yes
2	2017	55000	60000	yes
3	2015	80000	70000	no
3	2016	90000	75000	no
3	2017	99000	85000	no

One Typical Method - Use Proc Transpose

create id

```
data long;  set have;  id=_n_;   run;
proc transpose data=long out=long;
  by id;   var income2015--debt2017;
run;
data long;
  set long;  year=input(substr(_NAME_,anydigit(_NAME_)),8.);
  _NAME_=substr(_NAME_,1,anydigit(_NAME_)-1);
run;
proc sort data=long;
  by id year;  run;
proc transpose data=long out=want;
  by id year;   var COL1;  run;
data _null_;
  set long (where=(id eq 1)) end=last;
  if _n_ eq 1 then
   call execute('proc datasets lib=work nolist; modify want;');
  call execute('label '||_NAME_||'='||_LABEL_||';');
  if last then call execute('run;quit;');
run;
data want (drop=_:);
  set want (rename=(income=_income expenses=_expenses));
  income=input(left(_income),8.);
expenses=input(left(_expenses),8.);  run;
```

One Typical Method - Use Proc Transpose

```
data long;  set have;  id=_n_;   run;
proc transpose data=long out=long;
  by id;   var income2015--debt2017;
run;
data long;
  set long;  year=input(substr(_NAME_,anydigit(_NAME_)),8.);
  _NAME_=substr(_NAME_,1,anydigit(_NAME_)-1);
run;
proc sort data=long;
  by id year;  run;
proc transpose data=long out=want;
  by id year;   var COL1;  run;
data _null_;
  set long (where=(id eq 1)) end=last;
  if _n_ eq 1 then
   call execute('proc datasets lib=work nolist; modify want;');
  call execute('label '||_NAME_||'='||_LABEL_||';');
  if last then call execute('run;quit;');
run;
data want (drop=_:);
  set want (rename=(income=_income expenses=_expenses));
  income=input(left(_income),8.);
expenses=input(left(_expenses),8.);  run;
```

make file long

One Typical Method

```
data long;  set have;  id=_n_;   run;
proc transpose data=long out=long;
  by id;   var income2015--debt2017;
run;
data long;
  set long;  year=input(substr(_NAME
  _NAME_=substr(_NAME_,1,anydigi
run;
proc sort data=long;
  by id year;  run;
proc transpose data=long out=want;
  by id year;   var COL1;  run;
data _null_;
  set long (where=(id eq 1)) end=last;
  if _n_ eq 1 then
   call execute('proc datasets lib=wor
  call execute('label '||_NAME_||'='||_L
  if last then call execute('run;quit;');
run;
data want (drop=_:);
  set want (rename=(income=_incom
  income=input(left(_income),8.);
expenses=input(left(_expenses),8.);  run;
```

	id	_NAME_	_LABEL_	COL1
1	1	income2015	Yearly Income	70000
2	1	income2016	Yearly Income	75500
3	1	income2017	Yearly Income	80000
4	1	expenses2015	Yearly Expenses	60000
5	1	expenses2016	Yearly Expenses	70000
6	1	expenses2017	Yearly Expenses	81000
7	1	debt2015	Yearly Dept	no
8	1	debt2016	Yearly Dept	no
9	1	debt2017	Yearly Dept	yes
10	2	income2015	Yearly Income	50000
11	2	income2016	Yearly Income	52000
12	2	income2017	Yearly Income	55000
13	2	expenses2015	Yearly Expenses	42000

One Typical Method - Use Proc Transpose

```
data long;  set have;  id=_n_;   run;
proc transpose data=long out=long;
  by id;   var income2015--debt2017;
run;
data long;
  set long;  year=input(substr(_NAME_,anydigit(_NAME_)),8.);
  _NAME_=substr(_NAME_,1,anydigit(_NAME_)-1);
run;
proc sort data=long;
  by id year;  run;
proc transpose data=long out=want;
  by id year;   var COL1;  run;
data _null_;
  set long (where=(id eq 1)) end=last;
  if _n_ eq 1 then
   call execute('proc datasets lib=work nolist; modify want;');
  call execute('label '||_NAME_||'='||_LABEL_||';');
  if last then call execute('run;quit;');
run;
data want (drop=_:);
  set want (rename=(income=_income expenses=_expenses));
  income=input(left(_income),8.);
  expenses=input(left(_expenses),8.);  run;
```

parse year and variable names from _NAME_

sort the file

make file wide

add variable labels

correct for numeric variables

One Typical Method - Use Proc Transpose

```
data long;  set have;  id=_n_;   run;
proc transpose data=long out=long;
  by id;   var income2015--debt2017;
run;
data long;
  set long;  year=input
  _NAME_=substr(_NA
run;
proc sort data=long;
  by id year;  run;
proc transpose data=l
  by id year;   var COL
data _null_;
  set long (where=(id e
  if _n_ eq 1 then
   call execute('proc da
  call execute('label '||
  if last then call execu
run;
data want (drop=_:);
  set want (rename=(in
  income=input(left(_income),8.);
expenses=input(left(_expenses),8.);  run;
```

Gets the job done, but

requires a lot of coding

loses formats

converts numeric fields to character

while the method is extensible, the code isn't

will output records even if they only contain missing values

creates an output file that contains a lot of redundant metadata

Alternatively- Transpose Each Variable Separately

```
data long;  set have;  id=_n_; run;
proc transpose data=long out=long_income prefix=income;
  by id;  var income2015-income2017;  run;
data _null_;  set long_income (obs=1);
  call execute('proc datasets lib=work nolist;modify long_income;
     label income1='|| quote(strip(_LABEL_)) || ';run;quit;');   run;
proc transpose data=long out=long_expenses prefix=expenses;
  by id;  var expenses2015-expenses2017;  run;
data _null_;   set long_expenses (obs=1);
  call execute('proc datasets lib=work nolist;modify long_expenses;
     label expenses1='|| quote(strip(_LABEL_)) || ';run;quit;');
run;
proc transpose data=long out=long_debt prefix=debt;
  by id;   var debt2015-debt2017;  run;
data _null_;   set long_debt (obs=1);
  call execute('proc datasets lib=work nolist;modify long_debt;
     label debt1='|| quote(strip(_LABEL_)) || ';run;quit;');  run;
data want (drop=_:);
  set long_income(rename=(income1=income) drop=_:);
  set long_expenses(rename=(expenses1=expenses) drop=_:);
  set long_debt(rename=(debt1=debt));
  year=input(substr(_NAME_,5),4.);  run;
```

create id

transpose income

assign label

transpose expenses

assign label

transpose debt

assign label

merge files, rename variables, and parse year from _NAME_

Alternatively- Transpose Each Variable Separately

```
data long;  set have;  id=_n_; run;
proc transpose data=long out=long_income prefix=income;
  by id;  var income2015-income2017;  run;
data _null_;  set long_inco
  call execute('proc dataset
     label income1='|| quote(
proc transpose data=long
  by id;  var expenses2015-
data _null_;    set long_exp
  call execute('proc dataset
     label expenses1='|| quo
run;
proc transpose data=long
  by id;   var debt2015-debt
data _null_;   set long_deb
  call execute('proc dataset
     label debt1='|| quote(str
data want (drop=_:);
  set long_income(rename=
  set long_expenses(renam
  set long_debt(rename=(debt1=debt));
  year=input(substr(_NAME_,5),4.);  run;
```

Fewer Problems and Runs Faster, but:

code requires more thought

greater chance of introducing error

code isn't extensible

will output records even if they only contain missing values

creates output files that contain a lot of redundant metadata

A Much Easier Method

Use Gerhard Svolba's makelong macro

```
data long;  set have;  id=_n_; run;
%makelong(data=have,out=want,id=id,
   measurement=year,root=income expenses debt)
```

- Not applicable because the macro can't handle a combination of character and numeric variables
- Otherwise, would have performed as well as the second example, but with less code and thought required

Preferred Method

the %untranspose Macro

```
%untranspose(data=have,out=want, id=year,
  var=income expenses debt, create_byvar=id)
```

- Less code thus less time to prepare and lower chance of user error
- Very Extensible
- Can handle any variable name that PROC TRANSPOSE can create
- Works with any combination of character and/or numeric variables
- You choose whether to output records with all missing values
- All variables maintain their original types, lengths, formats, informats and labels
- Parameter names are the same as PROC TRANSPOSE's options and statements
- Lets you create a non-redundant file of metadata (if desired)
- Can create a new by variable that will contain the sequential record numbers
- Much faster than the other methods

Preferred Method

Use the %untranspose Macro

In addition to the time saved not having to write code the %untranspose macro runs faster than the other methods

with 500,000 records and three variables

72.5 times faster than Method 1

13 times faster than Method 2 or the %makelong macro

Speed differentials increase with both number of records AND/OR the number of variables to be untransposed

How to use the %UNTRANSPOSE macro

Toronto Area SAS Society

September 14, 2018

But, first, what is a macro?

- SAS code that begins with a %macro statement , ends with a %mend statement, and conforms to the SAS Macro language
- Macros generate text which (like in this case) can be an entire set of SAS programs
- Macros have to be both compiled and executed
- They can be compiled by either *opening and running them*, *specifying them in a %include statement*, *storing them as compiled stored macros*, or by *using the SASAUTOCALL facility*. Instructions for each method is described in a paper, by Harry Droogendyk, *Which SASAUTOS Macros Are Available to My SAS® Session* (https://analytics.ncsu.edu/sesug/2008/SBC-126.pdf)
- Macros can be difficult to write unless you're very familiar with the SAS macro language
- But, you don't have to know the SAS macro language to use macros that others have written

How to Run the %untranspose Macro

As with running any macro you only have to type:

%untranspose( )

a % (percent) sign

the macro's name

a left parenthesis

a right parenthesis
no ending semi-colon required

any parameters for which you want to use a value that is different than the parameter's default value
(e.g., data=have, out=want)
comma separated

The %untranspose macro uses Named Parameters
(i.e., parameters whose names are followed by an = sign)

Benefits

- Before compiling you get to determine the default values for each parameter
- When running the macro you only need to specify those parameters for which you want to assign values that are different than the parameters' default values
- No need to differentiate between options and statements like you have to with PROC TRANSPOSE. Parameter order is irrelevant
- No relearning is necessary because the macro's parameters have the same names as PROC TRANSPOSE's options and statements

How to Use the %untranspose Macro

Named Parameters and their initial Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

Note: If you want parameters to have different default values you simply have to change them before compiling the macro

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
```

```
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

you can change the libnames used for one-level file names

e.g. libname_in = some libname

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

you can specify one- or two-level file names
and, for either, you can include any desired data step options

e.g. data=have OR data=myfiles.have OR data=have (obs=10)

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
```

```
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

the parameter that lets you specify the variable or variables that define the level of the wide data= dataset

by=id

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix =_,
var=,
id=,
id_informat=8.,
create_byvar=)
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

example transposed variable name → **_04APR2018_Weight_Actual**

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix =_,
var=,
id=,
id_informat=8.,
create_byvar=)
```

```
id_format=8.,
var_first =no,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

example transposed variable name → **_04APR2018_Weight_Actual**

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix =_,
var=,
id= date,
id_informat= date9.,
create_byvar=)
```

```
id_format=date9.,
var_first =no,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

example transposed variable name → _04APR2018_Weight_Actual

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix =_,
var=,
id= date,
id_informat= date9.,
create_byvar=)
```

```
id_format=date9.,
var_first =no,
delimiter =_,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

example transposed variable name → _04APR2018_Weight_Actual

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix =_,
var=weight,
id= date,
id_informat= date9.,
create_byvar=)
```

```
id_format=date9.,
var_first =no,
delimiter =_,
suffix=,
copy=,
makelong=NO,
max_length=,
```

Note: can be a variable or any variable list

example transposed variable name → **_04APR2018_Weight_Actual**

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix =_,
var =weight,
id= date,
id_informat= date9.,
create_byvar=)
```

```
id_format=date9.,
var_first =no,
delimiter=_,
suffix=_Actual,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

example transposed variable name → _04APR2018_Weight_Actual

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

Specify any variable(s) that should be copied rather than transposed

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

Specify whether the macro should output records that only have missing values for the variables listed in the var parameter

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

If you want the macro to output a dataset containing all of the variable names, types, lengths, formats, informats and labels, set this parameter to equal the one or two-level dataset name

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
```

```
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

The default (i.e., no) will output records at the by variable, id variable level.
=yes will output records at the by variable, id variable, var variable level

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

Minimize processing time (when untransposing to a long format) by indicating the maximum variable length

How to Use the %untranspose Macro

Named Parameters and their Default Values

```
(libname_in=work,
libname_out=work,
data=,
out=,
by=,
prefix=,
var=,
id=,
id_informat=8.,
create_byvar=)
id_format=8.,
var_first=yes,
delimiter=,
suffix=,
copy=,
missing=NO,
metadata=,
makelong=NO,
max_length=,
```

If you don't have a by variable, specifying a variable name will cause the macro to create a sequential variable

Examples

The macro comes with a tip sheet that includes six pages of examples

Untranspose Macro Tip Sheet

Purpose: The macro untransposes wider SAS datasets back to either the less wide state that existed before the file was transposed, or to a long file. The macro can accommodate any prefixes, variable names, delimiters, ID values, and suffixes that may exist in the transposed variable names.

Named Parameters: The macro uses named parameters so that (1) default values can be assigned and (2) the various parameters only have to be specified when values other than the default values are required. We attempted, as closely as possible, to use the same option names and statements as those used for PROC TRANSPOSE. When calling the macro, the default values will be used unless you specify the desired value. Thus, if you wanted the macro to typically get your data from a libname called mydata, you would modify the parameter by specifying it in the macro declaration.

Parameter	Required	Possible Values	Default Value	Description
libname_in	No	Any valid libname	work	The parameter to which you can assign the name of the SAS library that contains the dataset you want to untranspose
libname_out	No	Any valid libname	work	The parameter to which you can assign the name of the SAS library where you want the untransposed file written
data	Yes	Any valid filename	None	The parameter to which you assign the one or two-level name of the file that you want to untranspose
out	Yes	Any valid filename	None	The parameter to which you assign the name of the file that you want the macro to create
by	No	Any variable name from the file specified in the data parameter	None	The parameter to which you would assign the name of the original dataset's by variable(s)
prefix	No	Any valid SAS name characters	None	The parameter to which you assign the string (if any) that the transposed variable names begin with
var	Yes	Any valid SAS name	None	The parameter to which you assign the name(s) of the original variables that had been transposed
id	No	Any valid SAS name	None	The parameter to which you specify the variable name that was used as the ID variable (if any) when the transposed file was created. Only one variable can be assigned
Id_informat	No	Any valid SAS informat	8.	The parameter to which you can assign the informat to be used to extract the id variable's values
Id_format	No	Any valid SAS format	8.	The parameter to which you can indicate the format you want assigned to the id variable
var_first	No	YES=<prefix>var<delimiter>id<suffix> NO=<prefix>id<delimiter>var<suffix> N/A=<prefix>var<suffix>	Yes	The parameter that defines whether var names precede id values in the transposed variable names
delimiter	No	Any valid SAS name characters	None	The parameter to which you assign the string (if any) that was used to separate var and ID values in the transposed variable names
suffix	No	Any valid SAS name characters	None	The parameter to which you can assign a string (if any) that the transposed variable names end with
copy	No	Any valid SAS name	None	The parameter to which you can assign the name(s) of any variables that had been copied
missing	No	Yes or no (case insensitive)	No	The parameter to indicate whether a record should be output if the only non-missing variables are the BY, ID and COPY variables
metadata	No	Any valid filename	None	The parameter to which you can specify the one or two-level SAS dataset the you want created to contain the untransposed variables' metdata
makelong	No	Yes or no (case insensitive)	No	The parameter to which you can specify that you want the macro to output records at the BY variable, ID variable value, var variable(s) level
max_length	No	Any number between 1 and 32767	None	The parameter to which you can specify the length of the _value_ variable
create_byvar	No	Any valid SAS variable name	None	The parameter to which you can have the macro create a by variable and assign sequential numbers

Usage Examples: The following are some examples of how you might use the macro. For each example the wide dataset's name is *have* and resides in the work library, and the less wide or long dataset created by the macro is called *want* and also resides in the work library.

dataset: ***have***

id	income2015	income2016	income2017	expenses2015	expenses2016	expenses2017	debt2015	debt2016	debt2017
1	70000	75500	80000	60000	70000	81000	no	no	yes
2	50000	52000	55000	42000	53000	60000	no	yes	yes
3	80000	90000	99000	70000	75000	85000	no	no	no

macro call: ***%untranspose*(data=have, out=want, by=id, id=year, var=income expenses debt)**

dataset: ***want***

id	year	income	expenses	debt
1	2015	70000	60000	no
1	2016	75500	70000	no
1	2017	80000	81000	yes
2	2015	50000	42000	no
2	2016	52000	53000	yes
2	2017	55000	60000	yes
3	2015	80000	70000	no
3	2016	90000	75000	no
3	2017	99000	85000	no

dataset: ***have***

id	income2015	income2016	income2017	expenses2015	expenses2016	expenses2017	debt2015	debt2016	debt2017
1	70000	75500	80000	60000	70000	81000	no	no	yes
2	50000	52000	55000	42000	53000	60000	no	yes	yes
3	80000	90000	99000	70000	75000	85000	no	no	no

macro call: ***%untranspose*(data=have (obs=1), out=want, by=id, id=year, var=income expenses debt)**

dataset: ***want***

id	year	income	expenses	debt
1	2015	70000	60000	no
1	2016	75500	70000	no
1	2017	80000	81000	yes

dataset: ***have***

id	income2015	income2016	income2017	expenses2015	expenses2016	expenses2017	debt2015	debt2016	debt2017
1	70000	75500	80000	60000	70000	81000	no	no	yes
2	50000	52000	55000	42000	53000	60000	no	yes	yes
3	80000	90000	99000	70000	75000	85000	no	no	no

macro call: ***%untranspose*(data=have (where=(id eq 2)), out=want, by=id, id=year, var=income expenses debt)**

dataset: ***want***

id	year	income	expenses	debt
2	2015	50000	42000	no
2	2016	52000	53000	yes
2	2017	55000	60000	yes

Where Can You Get the Macro?

On the internet, of course!

https://github.com/gerhard1050/Untranspose-a-Wide-File

The page includes the paper, the macro, the Powerpoint, and a tip sheet

BREAKING NEWS!

Last month we discovered a 5th way one can download, compile and run a macro

```
filename ut url 'http://tiny.cc/untranspose_macro';
%include ut ;
%untranspose(data=have, out=want, by=id, id=year,
 var=income expenses debt)
```

Both downloads and compiles the macro

where tiny.cc/untranspose points to a github *raw* file:

https://raw.githubusercontent.com/FriedEgg/Papers/master/An_Easier_and_Faster_Way_to_Untranspose_a_Wide_File/src/untranspose.sas

BREAKING NEWS!

Last month we discovered a 5th way one can download, compile and run a macro

```
filename ut url 'http://tiny.cc/untranspose';
%include ut ;
%untranspose(data=have, out=want, by=id, id=year,
 var=income expenses debt)
```

Runs the macro

where tiny.cc/untranspose points to a github *raw* file:

https://raw.githubusercontent.com/FriedEgg/Papers/master/An_Easier_and_Faster_Way_to_Untranspose_a_Wide_File/src/untranspose.sas

development: some techniques we used

Parse dataset options and Identify whether 1 or 2-level name was used

```
%let lp=%sysfunc(findc(%superq(data),%str(%())); 
%if &lp. %then %do;
  %let rp=                                  a),%str(%)),b));
  %let dso                                  onrstr(%superq(data)),
     &lp+1,
  %let data                                 %superq(data)),1,%eval(&lp-1)));
%end;
%else %let dsoptions=;
%if %sysfunc(countw(&data.)) eq 2 %then %do;
   %let libna
   %let data
 %end;
 %else %if %
   %let libna
 %end;
```

for example
from : &data=mylib.mydata;
create: &libname_in=mylib
and
&data=mydata

for example
from : &data=mylib.mydata (rename=(var1=age));
create: &dsoptions=rename=(var1=age)
and
&data=mylib.mydata

development: some techniques we used

Creating one record datasets and then using PROC SQL to create macro variables from dictionary.columns

```
data t_e_m_p;
   set &libname_in..&data. (obs=1 keep=&copy.);
run;

proc sql noprint;
   select name
      into :to_copy separated by " "
        from dictionary.columns
          where libname="WORK" and
                memname="T_E_M_P"
  ;
quit;
```

development: some techniques we used

Using a macro variable that contains a space-separated list of variable names to create a SAS dataset

```
data t_e_m_p;
   array vars(*) &var.;
   output;
run;
```

development: some techniques we used

The SAS Global Forum paper contains a section that describes every aspect of the macro's code

Presentation Overview

- What the %untranspose macro does ✓
- How to use the macro ✓
- Where to get the %untranspose macro (p.s., it's free!) ✓
- Some interesting techniques we used in creating the macro ✓

Questions?

Your comments and questions are valued and encouraged

Contact the Authors

Arthur Tabachneck, Ph.D., CEO
Analyst Finder, Inc.
Thornhill, ON
art@analystfinder.com

Joe Matise
NORC @ the University of Chicago
Chicago, IL
snoopy369@gmail.com

Gerhard Svolba, Ph.D.
SAS Institute
Wien, Austria
Gerhard.Svolba@sas.com

Matt Kastin
NORC @ the University of Chicago
Chicago, IL
fried.egg@verizon.com