Paper 4630-2020

Making Long Data Wide with a Flexible Macro

Stephanie R. Thompson, Jenzabar / Datamum

ABSTRACT

When you need to make a wide dataset from multiple sources where the data are long instead, this macro approach is for you. This macro has several variations that allow for customization and the addition of dynamic prefixes to the new variable names. The purpose of the macro was to take period-based data and put it into a format to be used for predictive modeling. Over 10 different data sources were used and this macro was easily applied to each. See how macros can make your life easier!

INTRODUCTION

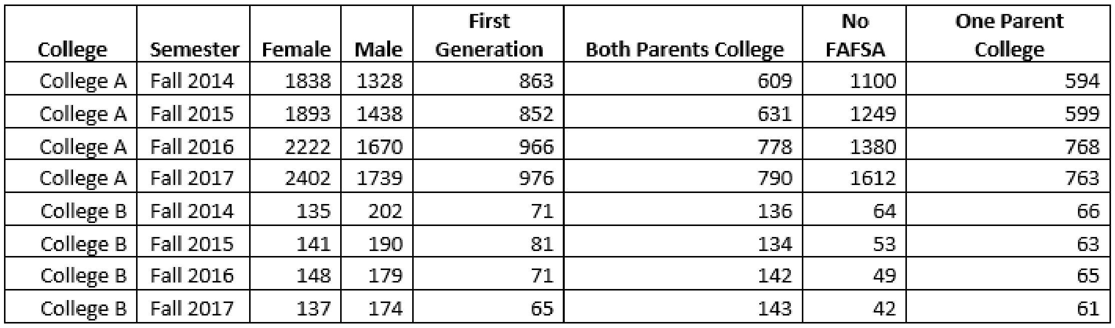
Getting data into the proper format for modeling can be the biggest part of the project. Many models require wide data as opposed to long data. Getting the data reorganized can take time especially if you have a large number of datasets and periods to rearrange.

Another challenge with data preparation for modeling is that you may have variables with the same name but for different time periods. In this case, you need to add something to the variable name to make sure you can differentiate them. The macro in this paper will add a prefix to all of the variables in the dataset automatically. To facilitate the macro, the dataset is split into multiple datasets for each time period of interest to make sure the proper prefix is associated with the new dataset.

PROC TRANSPOSE is another option to make long data wide where you can add a prefix – or even a suffix - to the variable names. However, PROC TRANSPOSE was not chosen for this particular situation because it does not handle multiple variables as easily. Developing a macro to handle each variable and prefix could be done and produce similar results. Basic code for PROC TRANSPOSE will be covered for reference and comparison.

THE DATA

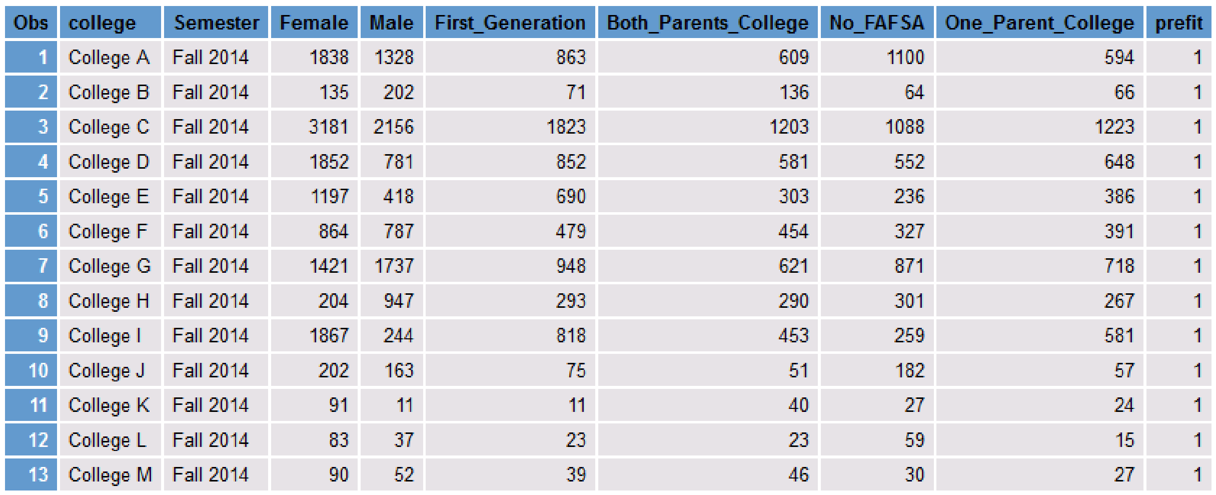
The data used for this project was created to mimic data for four fall semesters of enrollment numbers for a number of colleges across six variables. The data contain all of the colleges and all of the semesters in a long file. The desired output is a dataset that contains all of the colleges with each of the values for the 6 variables for each fall semester as individual variables. This type of structure may be desirable for modeling college values over time or to easily compare or preform operations on values year to year (e.g., the change in females from fall 2014 to 2015 by college). A partial view of the data is shown below in Figure 1 and the full data are in Appendix D. The Excel workbook containing the data is embedded into the paper so you can access it to test the code.



College	Semester	Female	Male	First Generation	Both Parents College	No FAFSA	One Parent College
College A	Fall 2014	1838	1328	863	609	1100	594
College A	Fall 2015	1893	1438	852	631	1249	599
College A	Fall 2016	2222	1670	966	778	1380	768
College A	Fall 2017	2402	1739	976	790	1612	763
College B	Fall 2014	135	202	71	136	64	66
College B	Fall 2015	141	190	81	134	53	63
College B	Fall 2016	148	179	71	142	49	65
College B	Fall 2017	137	174	65	143	42	61

Figure 1. Sample Data

In order for this particular macro to work, the full, long dataset needs to be split up into individual datasets based on the variable you will use as the prefix for the new variable names. One of the four versions of these datasets, for Fall 2014, is shown in Figure 2.



Obs	college	Semester	Female	Male	First_Generation	Both_Parents_College	No_FAFSA	One_Parent_College	prefit
1	College A	Fall 2014	1838	1328	863	609	1100	594	1
2	College B	Fall 2014	135	202	71	136	64	66	1
3	College C	Fall 2014	3181	2156	1823	1203	1088	1223	1
4	College D	Fall 2014	1852	781	852	581	552	648	1
5	College E	Fall 2014	1197	418	690	303	236	386	1
6	College F	Fall 2014	864	787	479	454	327	391	1
7	College G	Fall 2014	1421	1737	948	621	871	718	1
8	College H	Fall 2014	204	947	293	290	301	267	1
9	College I	Fall 2014	1867	244	818	453	259	581	1
10	College J	Fall 2014	202	163	75	51	182	57	1
11	College K	Fall 2014	91	11	11	40	27	24	1
12	College L	Fall 2014	83	37	23	23	59	15	1
13	College M	Fall 2014	90	52	39	46	30	27	1

Figure 2. AY1 Dataset

The code to generate the individual datasets is quite simple. In this example it is done manually, but it could also be automated by a macro. The code that generated AY1 in Figure 2 is below:

```
data ay1 ay2 ay3 ay4;
set mypaper.sampledata;
if semester = 'Fall 2014' then do;
    prefit = 1;
    output ay1;
end;
else if semester = 'Fall 2015' then do;
    prefit = 2;
    output ay2;
end;
else if semester = 'Fall 2016' then do;
    prefit = 3;
    output ay3;
end;
else do;
    prefit = 4;
    output ay4;
end;
run;
```

It is important to note that all of the output datasets need to have the same name prefix and include a sequential value at the end matching the prefit value. The variable prefit is used in the variable prefix generated by the macro to indicate the order of the variables in the final dataset. Also, the number of semesters - in this case 4 – will be passed to the macro to denote the number of iterations it will run.

THE MACRO

In order to best understand the macro, it will be helpful to see how it is invoked first. Knowing what parameters are used makes following the code easier. Three positional values are passed to the macro and are used within in. The first value is the name of the input datasets without the sequential value at the end (e.g. just the prefix). In this example it is “AY.” The next value in the macro is the number of iterations or the maximum prefit value. The third value is the prefix part of the name of the output datasets that the sequential value will be added to. The reason for keeping the names of the datasets similar is to take advantage of the colon to reference multiple datasets with the same beginning characters. Using “AY:” in a DATA STEP will use AY1, AY2, AY3, and AY4 in this example.

The statement to invoke the macro is:

```
%vars(ay,4,aynew);
```

The macro will process all of the variables in the dataset by adding a prefix to them. You do not need to know how many variables are in the dataset or even their names. This is an advantage to PROC TRANSPOSE.

```
%macro vars(dsn,num,out);
   %do j = 1 %to &num;
   %let dsid=%sysfunc(open(&dsn&j));
   %let n=%sysfunc(attrn(&dsid,nvars));
   %let pref = semester&j._;
       data &out&j;
          set &dsn&j(rename=(
          %do i = 1 %to &n;
             %let var=%sysfunc(varname(&dsid,&i));
                 &var=&pref&var
          %end;));
          %let rc=%sysfunc(close(&dsid));
          where &pref.prefit = &j;
          drop &pref.prefit &pref.semester ;
          rename &pref.college = college;
     run;

     proc sort data = &out&j;
     by college;
     run;

    %end;
%mend vars;
```

Some of the key lines of code in the macro relate to how it opens the dataset and determines the number of variables. The automatic count and processing of all of the variables in the dataset is what gives this an advantage to other methods. The line "%let n=%sysfunc(attrn(&dsid,nvars)); is where the number of variables the first dataset – AY1 in the example - is assigned the count of variable it contains. The variable n is then used to limit a loop that changes the name of the variables by adding in the determined prefix.
The prefix for the variables is assigned in %let pref = semester&j._; It is important to note the period is needed after the “&j” to end the reference to the macro variable j. Without the period, SAS® will include the underscore in the macro variable name and will not generate the desired result. The underscore is added to the prefix only to make it more easily read. It is not required.

The rename dataset option in the macro is quite long and is this section of the code:

```
(rename=(
          %do i = 1 %to &n;
             %let var=%sysfunc(varname(&dsid,&i));
                 &var=&pref&var
          %end;))
```

The loop over the number of variables is part of the statements and allows for the renaming of each variable.

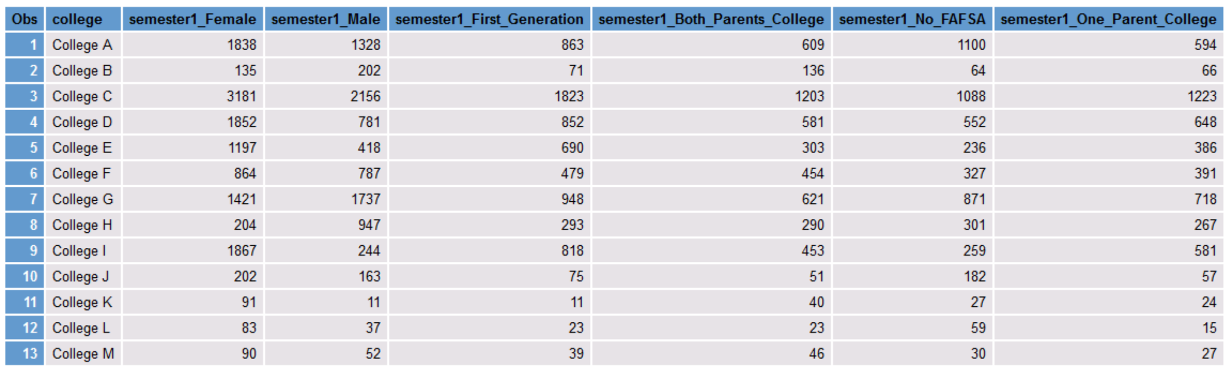
The dataset must also be closed since it has been explicitly opened. This is done in this statement: %let rc=%sysfunc(close(&dsid));

The remainder of the macro drops, renames, and sorts the new dataset. Some of the variables are only needed for processing and are not necessary in the new dataset. The rename is used for college as that is the ID variable that will be used to merge the new datasets on. If you wanted to skip particular variables in the rename or if you know your ID variable is variable one in all of the datasets, you can adjust the starting value or list of values in the do loop for i.

Hopefully you can see why the datasets were named sequentially as it facilitates the macro. The macro could be modified to read in a list of dataset names to iterate over if you would prefer. Another potential modification to the macro is to generate the prefit values automatically. These steps were not done in the original version of the macro as they were not needed to obtain the desired results.

THE RESULT

Once the macro has run, the resulting dataset looks like the one in Figure 3.

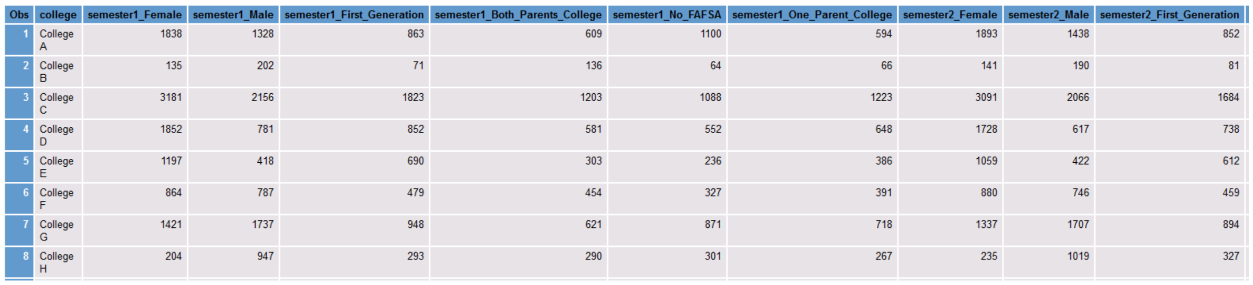


Obs	college	semester1_Female	semester1_Male	semester1_First_Generation	semester1_Both_Parents_College	semester1_No_FAFSA	semester1_One_Parent_College
1	College A	1838	1328	863	609	1100	594
2	College B	135	202	71	136	64	66
3	College C	3181	2156	1823	1203	1088	1223
4	College D	1852	781	852	581	552	648
5	College E	1197	418	690	303	236	386
6	College F	864	787	479	454	327	391
7	College G	1421	1737	948	621	871	718
8	College H	204	947	293	290	301	267
9	College I	1867	244	818	453	259	581
10	College J	202	163	75	51	182	57
11	College K	91	11	11	40	27	24
12	College L	83	37	23	23	59	15
13	College M	90	52	39	46	30	27

Figure 3. AYNEW1 Dataset

Each variable, other than college which is the ID variable, now has a prefix of "semester1_" to indicate the time period of the data.

When all 4 datasets are combined, you will get a dataset that looks like the one in Figure 4.



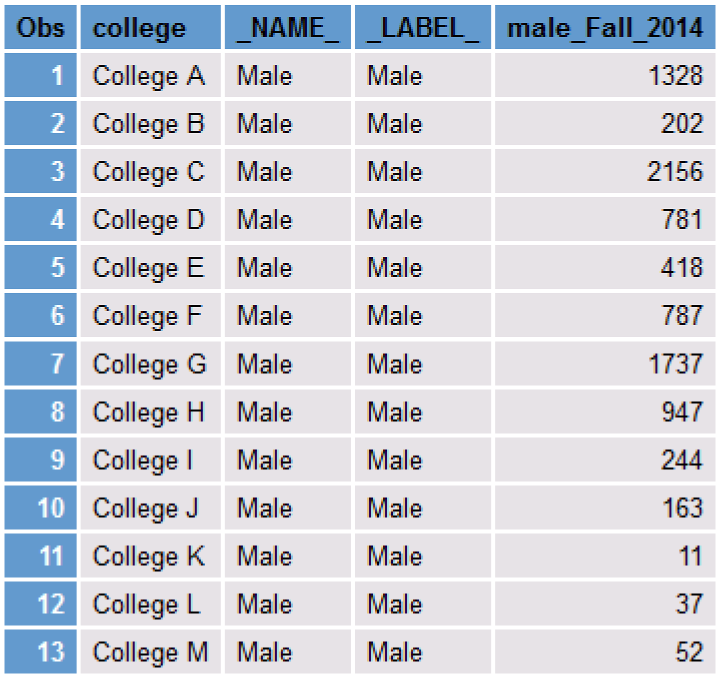
Obs	college	semester1_Female	semester1_Male	semester1_First_Generation	semester1_Both_Parents_College	semester1_No_FAFSA	semester1_One_Parent_College	semester2_Female	semester2_Male	semester2_First_Generation
1	College A	1838	1328	863	609	1100	594	1893	1438	852
2	College B	135	202	71	136	64	66	141	190	81
3	College C	3181	2156	1823	1203	1088	1223	3091	2066	1684
4	College D	1852	781	852	581	552	648	1728	617	738
5	College E	1197	418	690	303	236	386	1059	422	612
6	College F	864	787	479	454	327	391	880	746	459
7	College G	1421	1737	948	621	871	718	1337	1707	894
8	College H	204	947	293	290	301	267	235	1019	327

Figure 4. COLLEGENEW Dataset

This is only a partial view of the new dataset which now contains 25 variables whereas the original dataset has 8. Having all of the semester data values in columns will make modeling and comparisons easier.

PROC TRANSPOSE

Considering that there are multiple ways to accomplish most tasks in SAS, PROC TRANSPOSE is presented as an option. The syntax for PROC TRANSPOSE uses the names of the variables and can add a prefix to the transposed variables. PROC TRANSPOSE also generates the variables _NAME_ and _LABEL_ which are many times dropped in the resulting dataset. The transpose from the end of the full sample code in Appendix A generates the following dataset:



Obs	college	_NAME_	_LABEL_	male_Fall_2014
1	College A	Male	Male	1328
2	College B	Male	Male	202
3	College C	Male	Male	2156
4	College D	Male	Male	781
5	College E	Male	Male	418
6	College F	Male	Male	787
7	College G	Male	Male	1737
8	College H	Male	Male	947
9	College I	Male	Male	244
10	College J	Male	Male	163
11	College K	Male	Male	11
12	College L	Male	Male	37
13	College M	Male	Male	52

Figure 5. AY_TRANSPOSED Dataset

This use of PROC TRANSPOSE does not transpose all of the variables and it puts the semester identifier at the end of the variable instead of at the beginning. It may be possible to generate a dataset like the above with all of the variables in one pass of the transpose procedure. Another option is running multiple transpose procedures for each variable in a macro loop which might more easily accomplish the task. A fully developed alternative to the macro presented in this paper has not been developed. PROC TRANSPOSE is only being mentioned as a potential alternative.

ADDITIONAL PARAMETER MACRO

An additional parameter was added to the original macro to allow it to run on datasets that had an additional variable in the long data. The full code is in Appendix B. If, for example, the variable male and female was included in a variable called gender, the original dataset could be slit out by semesters by gender. Invoking the macro with four parameters would be done as shown below:

```
%vars2(ay_v2m,4,male,ay_v3m);

%vars2(ay_v2f,4,female,ay_v3f);
```

When originally writing the program, the source data was mixed in terms of how the data were structured. Having two versions of the macro that could be invoked made it easier to process all of the data. The two versions meant that I did not have to manipulate all of the datasets into a single format and this saved time in the overall coding. Also, since each dataset run through the macro needed to be for a single semester and have the prefit variable added, that code was easily reused throughout the program. Again, this bit of code could also have been transformed into a macro to make the final program even more streamlined.

CONCLUSION

The goal of this paper is to provide a method of getting multiple datasets into a format so they can be easily combined for modeling or some other use. The additional benefit of this method is that prefixes can be added to the variable names to make sequencing and comparisons easier. The macro is flexible and allows users to alter it to meet additional needs.

ACKNOWLEDGMENTS

Some of the ideas for this macro were found online. Unfortunately, the specific references have been lost. However, I wish to thank the SAS user community and SAS Institute Inc., for their willingness to share code and ideas through various online forums and websites.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Stephanie R. Thompson
Sr. Data Scientist / Principal
Jenzabar / Datamum
stephanie@datamum.com
http://www.Jenzabar.com
http://www.datamum.com

APPENDIX A
FULL SAS CODE

SAS Code File

Program 1.sas

```
/** First parameter is the data set that contains all of the variables.  **/
/** Second parameter is the number of loops.              **/
/** Third parameter is the new data set that contains the new variables. **/

libname mypaper '/folders/myfolders/Long to Wide Paper';

/* dataset mypaper.sampledata */

data ay1 ay2 ay3 ay4;
set mypaper.sampledata;
if semester = 'Fall 2014' then do;
        prefit = 1;
        output ay1;
end;
else if semester = 'Fall 2015' then do;
        prefit = 2;
        output ay2;
end;
else if semester = 'Fall 2016' then do;
        prefit = 3;
        output ay3;
end;
else do;
        prefit = 4;
        output ay4;
end;
run;
```

```
%macro vars(dsn,num,out);
   %do j = 1 %to &num;
   %let dsid=%sysfunc(open(&dsn&j));
   %let n=%sysfunc(attrn(&dsid,nvars));
   %let pref = semester&j._;
                data &out&j;
                   set &dsn&j(rename=(
                   %do i = 1 %to &n;
                      %let var=%sysfunc(varname(&dsid,&i));
                                        &var=&pref&var
                   %end;));
                   %let rc=%sysfunc(close(&dsid));
                          where &pref.prefit = &j;
                          drop &pref.prefit &pref.semester ;
                          rename &pref.college = college;
              run;

              proc sort data = &out&j;
              by college;
              run;

             %end;
%mend vars;

%vars(ay,4,aynew);

data newcolleges;
merge aynew: ;
by college;
run;

/* proc transpose comparison */

proc sort data=mypaper.sampledata out=ay_sort;
by college ;
run;
```

```
proc transpose data=ay_sort out=ay_transposed prefix = male_;
id semester;
by college;
var male ;*female Both_Parents_College First_Generation No_FAFSA One_Parent_College;
where semester = 'Fall 2014';
run;
```

APPENDIX B
VERSION TO RUN BY ADDITIONAL PARAMETER

Code included on embedded SAS code file

```
/* second version if gender as a row split out into additional datasets */
/* sample data not configured for this macro */



%macro vars2(dsn,num,typ,out);
  %do j = 1 %to &num;
  %let dsid=%sysfunc(open(&dsn&j));
  %let n=%sysfunc(attrn(&dsid,nvars));
  %let pref = semester&j._&typ._;
           data &out&j;
             set &dsn&j(rename=(
             %do i = 1 %to &n;
               %let var=%sysfunc(varname(&dsid,&i));
                           &var=&pref&var
             %end;));
             %let rc=%sysfunc(close(&dsid));
                  where &pref.prefit = &j;
                  drop &pref.prefit &pref.semester ;
                  rename &pref.college = college;
          run;

          proc sort data = &out&j;
          by college;
          run;

         %end;
%mend vars2;



%vars2(ay_v2m,4,male,ay_v3m);


%vars2(ay_v2f,4,female,ay_v3f);
```

APPENDIX C
IMPORT CODE

SAS File

import excel data.sas

```
/* Generated Code (IMPORT) */
/* Source File: sample data.xlsx */
/* Source Path: /folders/myfolders/Long to Wide Paper */
/* Code generated on: 2/21/20, 10:46 AM */

libname mypaper '/folders/myfolders/Long to Wide Paper';

%web_drop_table(mypaper.sampledata);



FILENAME REFFILE '/folders/myfolders/Long to Wide Paper/sample data.xlsx';

PROC IMPORT DATAFILE=REFFILE
        DBMS=XLSX
        OUT=mypaper.sampledata;
        GETNAMES=YES;
RUN;

PROC CONTENTS DATA=mypaper.sampledata; RUN;



%web_open_table(mypaper.sampledata);
```

APPENDIX D
DATA

Excel Workbook of Sample Data

sample data.xlsx

College	Semester	Female	Male	First Generation	Both Parents College	No FAFSA	One Parent College
College A	Fall 2014	1838	1328	863	609	1100	594
College A	Fall 2015	1893	1438	852	631	1249	599
College A	Fall 2016	2222	1670	966	778	1380	768
College A	Fall 2017	2402	1739	976	790	1612	763
College B	Fall 2014	135	202	71	136	64	66
College B	Fall 2015	141	190	81	134	53	63
College B	Fall 2016	148	179	71	142	49	65
College B	Fall 2017	137	174	65	143	42	61
College C	Fall 2014	3181	2156	1823	1203	1088	1223
College C	Fall 2015	3091	2066	1684	1141	1141	1191
College C	Fall 2016	3035	1992	1612	1101	1119	1195
College C	Fall 2017	3034	1862	1549	1125	1057	1165
College D	Fall 2014	1852	781	852	581	552	648
College D	Fall 2015	1728	617	738	537	490	580
College D	Fall 2016	1137	261	373	295	419	311
College D	Fall 2017	1100	267	403	278	408	278
College E	Fall 2014	1197	418	690	303	236	386
College E	Fall 2015	1059	422	612	291	257	321
College E	Fall 2016	927	393	536	248	238	298
College E	Fall 2017	830	378	460	233	228	287
College F	Fall 2014	864	787	479	454	327	391
College F	Fall 2015	880	746	459	467	307	393
College F	Fall 2016	892	800	451	491	332	418
College F	Fall 2017	944	778	460	514	318	430
College G	Fall 2014	1421	1737	948	621	871	718
College G	Fall 2015	1337	1707	894	635	861	654
College G	Fall 2016	1306	1774	875	652	914	639
College G	Fall 2017	1287	1796	845	698	883	657
College H	Fall 2014	204	947	293	290	301	267
College H	Fall 2015	235	1019	327	320	339	268

College H	Fall 2016	261	1157	358	378	359	323
College H	Fall 2017	294	1203	360	402	412	323
College I	Fall 2014	1867	244	818	453	259	581
College I	Fall 2015	1837	267	843	465	250	546
College I	Fall 2016	1933	297	903	486	260	581
College I	Fall 2017	1868	289	866	490	232	569
College J	Fall 2014	202	163	75	51	182	57
College J	Fall 2015	107	68	22	17	127	9
College J	Fall 2016	137	109	35	26	171	14
College J	Fall 2017	170	143	34	35	224	20
College K	Fall 2014	91	11	11	40	27	24
College K	Fall 2015	89	8	8	37	35	17
College K	Fall 2016	97	6	9	45	32	17
College K	Fall 2017	103	8	5	62	30	14
College L	Fall 2014	83	37	23	23	59	15
College L	Fall 2015	109	50	23	33	85	18
College L	Fall 2016	129	62	34	44	85	28
College L	Fall 2017	137	55	31	37	92	32
College M	Fall 2014	90	52	39	46	30	27
College M	Fall 2015	147	174	94	95	58	74
College M	Fall 2016	114	181	69	87	65	74
College M	Fall 2017	129	211	98	98	76	68