Paper CC45

@/@@ This Text file. Importing Non-Standard Text Files using @,@@ and / Operators.

Russell Woods, Wells Fargo Home Mortgage, Fort Mill SC

ABSTRACT

SAS® in recent years has done a fantastic job at ad ding newer and more powerful tools to the analyst and developer tool boxes, but these options are only as effective as the data that is available to them. Most of you would have no trouble dealing with a simple delimited or column formatted text file. However, data can often be provided in a non-standard format that must be parsed before analysis can be performed, or the final results have very specific formatting rules that must be adhered to. In these cases we can use some simple SAS operators, '@', '@@' and '/' in conjunction with conditional statements to interact with any flat file that has some kind of distinguishable pattern. In this presentation I will demonstrate the step-by-step process I used to analyze several non-standard text files and create import specifications capable of importing them into SAS. It is my hope that once you master t hese techniques it will not matter if you are preparing an audit report for the federal government or a side-effects analysis for the FDA you will easily be able to accommodate any specifications they may have.

INTRODUCTION

Living in a data-driven world requires being able to work with data in various formats, not only data that is clean and simplistic. The majority of programmers probably know how to import simple text files where one row correlates to one record and the format is either delimited or each column is well defined. That is relatively easy. However, consider encountering a file where your input statement has to be different based on some indicator present in the file? This will require you to read ahead and apply some logic statements to import the file successfully. Luckily, SAS provides all the tools needed to work with thi s kind of complex text file. In the following paper a primer on file importing for the uninitiated will be provided then a step by step process will be laid out to solve the problem of importing complex text files.

PRIMER ON FILE IMPORTING

To start, it is important to establish the fundamentals of importing text files. Whether a simple text file or something more exotic is being imported, the basic elements are going to be the same. The DATA step combined with the infile and input statements will be doing the heavy lifting for any text importing. Importing a simple text file where each row of text corresponds to 1 record can often times be done with the import data wizard or some very basic code. For the following examples the Theatre.Expenses SAS training dataset was used as a basis.

File Edit Format View Help

Theatre.Expenses SAS Learning Dataset 06/30/2014

FlightID	Flight Date	Cost of Flight
IA03400	02DEC1999	89155
IA03400	14DEC1999	39599
IA03400	26DEC1999	66800
IA03401	09DEC1999	33076
IA03401	21DEC1999	106032
IA10500	04DEC1999	47870
IA10500	16DEC1999	16106

Textfile1: Simple Text File

VIEWTABLE: Theater.Expenses

	FlightID	Date	Expenses
1	IA03400	02DEC1999	89155
2	IA03400	14DEC1999	39599
3	IA03400	26DEC1999	66800
4	IA03401	09DEC1999	33076
5	IA03401	21DEC1999	106032
6	IA10500	04DEC1999	47870

Dataset1: Simple text file after import

```
DATA SimpleImport;
infile '\\Simple.txt' firstobs = 4;
format FlightID $10. Date date9. Expenses best12.;
input
       @1           FlightID           $10.
                                       date9
       @20          Date               .
       @40          Expenses           ;
run;
```

Hopefully this is relatively straight forward. The desired dataset has three columns, so the input statement has three variables. No matter how complicated the text file you are importing, the basic structure of the input statement is not going to change. The only wrinkle added, as the title suggests, are the @, @@ and / operators.

SAS Operator	Operator function
@	Holds an input record for the execution of the next INPUT statement within the same iteration of the DATA step.[1]
@@	Holds the input record for the execution of the next INPUT statement across iterations of the DATA step. [1]
/	Advances to the next line in the text file.

Table 1. Operator definitions

By default, when you end an input statement, SAS advances to the next line of input. Therefore, if you are required to move through the text file in a non-sequential way or multiple records are on a single line you will need to use the trailing @ and double trailing @@.

SEEING @ AND @@ IN ACTION

In the following scenario, it is demonstrated how @ and @@ can be used when importing text. They are similar in function, but each fulfills a specific role

```
Theatre.Expenses SAS Learning Dataset          06/30/2014

FlightID        Flight Date     Cost of Flight
IA03400         02DEC1999       89155     IA03400     14DEC1999     39599
IA03400         26DEC1999       66800     IA03401     09DEC1999     33076
IA03401         21DEC1999       106032    IA10500     04DEC1999     47870
IA10500         16DEC1999       16106     IA10500     28DEC1999     29206
IA10501         11DEC1999       36028     IA10501     23DEC1999     23105
```

Textfile2: Multiple records on one row

	FlightID	Date	Expenses
1	IA03400	02DEC1999	89155
2	IA03400	14DEC1999	39599
3	IA03400	26DEC1999	66800
4	IA03401	09DEC1999	33076
5	IA03401	21DEC1999	106032
6	IA10500	04DEC1999	47870
7	IA10500	16DEC1999	16106
8	IA10500	28DEC1999	29206
9	IA10501	11DEC1999	36028
10	IA10501	23DEC1999	23105

Table2: Multiple records on one row after import.

It is possible to use the @ or the @@ to import this text file successfully, though in this particular case it makes a lot more sense to just use the @.

[1] SAS Support documentation:
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000146292.htm

```
DATA AmpersandImport;
infile '\\Simple ampersand.txt' firstobs = 4; format
FlightID $10. Date date9. Expenses best12.;

input
       @1                  FlightID            $10.
       @20                 Date                date9.
       @40                 Expenses            @;
output;
input
       @50                 FlightID            $10.
       @69                 Date                date9.
       @89                 Expenses            ;
output;
run;
```

In the above code sample, the first record in the line is imported identical to what was done in the previous example. The line then ends with a @, which causes SAS to hold the input record and stay at the current location. The next record on that line is then imported using another straightforward input statement. The output statement must be used because a single iteration of the DATA step is creating two records. The single ampersand does not hold the location across iterations of the DATA step so both records in the line must be imported on each pass.

```
DATA AmpersandImport2;
infile '\\Simple ampersand.txt' firstobs = 4; format
FlightID $10. Date date9. Expenses best12.;

if Counter = 0 then
       input
       @1                  FlightID            $10.
       @20                 Date                date9.
       @40                 Expenses            @@;
else
       input
       @50                 FlightID            $10.
       @69                 Date                date9.
       @89                 Expenses            ;
Counter +1;
if Counter = 2 then Counter = 0;
run;
```

While there are a lot of similarities between using @ and @@, this code example should illustrate some important distinctions. The most important one being that the first example runs through the DATA step 5 times, while the second actually runs through it 10. This is due to the flexibility of @@ holding the input record across iterations of the DATA step. It becomes obvious that in this particular example the @ is more efficient and looks cleaner, but it is an important academic exercise to see that both can be successfully used.

One possible case where the @@ can be very powerful is when a delimited file has varying numbers of records in each row. Using the @@ it is possible to write code to import one single complete record then hold the pointer at the point where the next record starts. This makes it possible to ignore the number of records in a single row as the DATA step is only concerned with importing 1 record at a time. When it reaches a line break it starts importing on the next line.

```
Theatre.Expenses SAS Learning Dataset            06/30/2014

FlightID        Flight Date     Cost of Flight
IA03400 02DEC1999       89155   IA03400 14DEC1999       39599   IA10101 19DEC1999       166744  IA11100 04DEC1999       136366
IA03400 26DEC1999       66800   IA03401 09DEC1999       33076
IA03401 21DEC1999       106032  IA10500 04DEC1999       47870
IA10500 16DEC1999       16106   IA10500 28DEC1999       29206   IA11101 23DEC1999       109002
IA10501 11DEC1999       36028   IA10501 23DEC1999       23105   IA03001 25DEC1999       32840
IA08700 11DEC1999       118044  IA08700 23DEC1999       30288
IA00400 30DEC1999       85973   IA00401 13DEC1999       25243   IA00401 25DEC1999       111484
```

Textfile3: Varying number of records on one row (tab delimited).

	FlightID	Date	Expenses
1	IA03400	02DEC1999	$89,155.00
2	IA03400	14DEC1999	$39,599.00
3	IA10101	19DEC1999	$166,744.00
4	IA11100	04DEC1999	$136,366.00
5	IA03400	26DEC1999	$66,800.00
6	IA03401	09DEC1999	$33,076.00
7	IA03401	21DEC1999	$106,032.00
8	IA10500	04DEC1999	$47,870.00
9	IA10500	16DEC1999	$16,106.00
10	IA10500	28DEC1999	$29,206.00
11	IA11101	23DEC1999	$109,002.00
12	IA10501	11DEC1999	$36,028.00
13	IA10501	23DEC1999	$23,105.00
14	IA03001	25DEC1999	$32,840.00
15	IA08700	11DEC1999	$118,044.00
16	IA08700	23DEC1999	$30,288.00
17	IA00400	30DEC1999	$85,973.00
18	IA00401	13DEC1999	$25,243.00
19	IA00401	25DEC1999	$111,484.00

Table3: All 19 rows are imported into the table successfully.

```
DATA MixedLineImport;
infile '\\Mixed Line Width.txt' firstobs = 4 dlm = '09'x;
informat FlightID $10. Date date9. Expenses best12.;
format Date date9. Expenses dollar12.2;

input
       FlightID     $
       Date
       Expenses     @@;

run;
```

This import is slightly different than the other examples due to the source file being tab delimited. Still, by adding '@@' this curious text file immediately becomes easy to read into a dataset. If the '@@' were removedthis DATA statement would create a table with only 7 records instead of the correct, 19. SAS would import the f irst record in a row, see no @@ and upon the ending of the INPUT statement, advance to the next line of the text file, skipping any other entries in that row. The @@ prevents this by holding the input record across iterations of the DATA step.

IMPORTING COMPLEX DATA

Taking stock (get to know your data):

American football career statistics for the 2014 NFL season were chosen to serve as the sample dataset. This dataset has a few components that make it non-trivial to import, but is still relatively straightforward. Step 1 in any data analysis process is to get to know your data, know the patterns and make decisions about what the final output should look like.

Football Statistics 06/30/2014
SESUG2014 Page 1

Player Name	Position	Season	Team	Rushing G	Att	yds	y/g	avg	TD	Receiving Rec	Tgt	yds	y/g	avg	YAC	TD	Fumbles Fum	FumL
Jamaal Charles	RB	2008	Chiefs	16	67	357	22.3	5.3	0	27	40	272	17.0	10.1	11.6	1	2	2
	RB	2009	Chiefs	15	190	1120	74.7	5.9	7	40	56	297	19.8	7.4	6.0	1	2	2
	RB	2010	Chiefs	16	230	1467	91.7	6.4	5	45	64	468	29.3	10.4	9.7	3	3	2
	RB	2011	Chiefs	2	12	83	41.5	6.9	0	5	6	9	4.5	1.8	-0.4	1	1	1
	RB	2012	Chiefs	16	285	1509	94.3	5.3	5	35	48	48	14.8	6.7	7.7	1	5	3
	RB	2013	Chiefs	15	259	1287	85.8	5.0	12	70	104	693	46.2	9.9	9.6	7	4	2
Totals				80	1043	5823	72.8	5.6	29	222	318	1975	24.7	8.7	8.7	14	17	12
LeSean McCoy	RB	2009	Eagles	16	155	637	39.8	4.1	4	40	55	308	19.3	7.7	9.9	0	2	1
	RB	2010	Eagles	15	207	1080	72.0	5.2	7	78	90	592	39.5	7.6	9.7	2	2	1
	RB	2011	Eagles	15	273	1309	87.3	4.8	17	48	69	315	21.0	6.6	8.8	3	1	1
	RB	2012	Eagles	12	200	840	70.0	4.2	2	54	67	373	31.1	6.9	9.6	3	4	3
	RB	2013	Eagles	16	314	1607	100.4	5.1	9	52	64	539	33.7	10.4	12.0	2	1	1
Totals				74	1149	5473	74.0	4.8	39	272	345	2127	28.7	7.8	10.0	10	10	7
Adrian Peterson	RB	2007	Vikings	14	238	1341	95.8	5.6	12	19	28	268	19.1	14.1	15.0	1	4	3
	RB	2008	Vikings	16	363	1760	110.0	4.8	10	21	39	125	7.8	6.0	6.6	0	9	4
	RB	2009	Vikings	16	314	1383	86.4	4.4	18	43	57	436	27.3	10.1	10.4	0	7	6
	RB	2010	Vikings	15	283	1298	86.5	4.6	12	36	50	341	22.7	9.5	9.7	1	1	1
	RB	2011	Vikings	12	208	970	80.8	4.7	12	18	23	139	11.6	7.7	7.0	1	1	0
	RB	2012	Vikings	16	348	2097	131.1	6.0	12	40	51	217	13.6	5.4	4.9	1	4	2
	RB	2013	Vikings	14	279	1266	90.4	4.5	10	29	40	171	12.2	5.9	5.7	1	5	3
Totals				103	2033	10115	98.2	5.0	86	206	288	1697	16.5	8.2	8.3	5	31	19

football Statistics 06/30/2014
SESUG2014 Page 2

Player Name	Position	Season	Team	Rushing G	Att	yds	y/g	avg	TD	Receiving Rec	Tgt	yds	y/g	avg	YAC	TD	Fumbles Fum	FumL
Matt forte	RB	2008	Bears	16	316	1238	77.4	3.9	8	63	76	477	29.8	7.6	7.0	4	1	1
	RB	2009	Bears	16	258	929	58.1	3.6	4	57	72	471	29.4	8.3	8.4	0	6	3
	RB	2010	Bears	16	237	1069	66.8	4.5	6	51	70	547	34.2	10.7	9.5	3	3	2
	RB	2011	Bears	12	203	997	83.1	4.9	3	52	76	490	40.8	9.4	8.8	1	2	2
	RB	2012	Bears	15	248	1094	72.9	4.4	5	44	60	340	22.7	7.7	7.3	1	2	1
	RB	2013	Bears	16	289	1339	83.7	4.6	9	74	94	594	37.1	8.0	7.5	3	2	2
Totals				91	1551	6666	73.3	4.3	35	341	448	2919	32.1	8.6	8.0	12	16	11
Eddie Lacy	RB	2013	Packers	15	284	1178	78.5	4.1	11	35	44	257	17.1	7.3	9.2	0	0	0
Totals				15	284	1178	78.5	4.1	11	35	44	257	17.1	7.3	9.2	0	0	0
Montee Ball	RB	2013	Broncos	16	120	559	34.9	4.7	4	20	27	145	9.1	7.3	7.2	0	3	3
Totals				16	120	559	34.9	4.7	4	20	27	145	9.1	7.3	7.2	0	3	3
Marshawn Lynch	RB	2007	Bills	13	280	1115	85.8	4.0	7	18	26	184	14.2	10.2	11.3	0	2	1
	RB	2008	Bills	15	250	1036	69.1	4.1	8	47	67	300	20.0	6.4	8.4	1	2	1
	RB	2009	Bills	13	120	450	34.6	3.8	2	28	37	179	13.8	6.4	7.3	0	3	1
	RB	2010	Seahawks	12	165	573	47.8	3.5	6	21	25	138	11.5	6.6	7.0	0	3	3
	RB	2010	Bills	4	37	164	41.0	4.4	0	1	3	7	1.8	7.0	4.0	0	1	1
	RB	2011	Seahawks	16	285	1204	80.3	4.2	12	28	41	212	14.1	7.6	8.1	1	3	2
	RB	2012	Seahawks	16	315	1590	99.4	5.0	11	23	30	196	12.3	8.5	9.0	1	5	2
	RB	2013	Seahawks	16	301	1257	78.6	4.2	12	36	44	316	19.8	8.8	8.1	2	4	1
Totals				104	1753	7389	71.0	4.2	58	202	273	1532	14.7	7.6	8.3	5	23	12

Textfile4: A subset of complex text file football statistics (Sourced from NFL.com career statistics).

A few items to notice right away are that the player name is only mentioned once, the player can have a varying number of seasons and there is a total line included. It should also be noted that it is possible for a player to have two rows of data for one season (Marshawn Lynch was traded in 2010 so played for two teams).

Criteria to be used when importing this file:

1. Create two tables. One table will have player name and the total line associated with them. The second table will be comprised of a separate record for each year that a player played. In the event that a player was with two teams, the two seasons will be combined into one. Statistics for the year will be summed and averaged accordingly and the team name column will be concatenated.
2. Player name will be carried through for each record even though in the text file it is only listed once.

DECIDE THE TABLE STRUCTURE (COLUMN ORDER AND FORMATS):

Once the text file is analyzed the next logical step is to finalize the format of the final data set. It is much easier to code your input statements if the formatting decisions have already been made. For the two datasets being created the following formatting decisions were made:

```
informat
      PlayerName $20. Position $2. Season 4. Team $25. GamesPlayed 4.
      Ru_Attempts 5. Ru_yds 10. ru_yards_game 6.2 ru_avg 6.2 ru_td 4.
      re_Receptions 5. re_targets 5. re_yds 10. re_yards_game 6.2
            re_avg 6.2 re_YAC 6.2 re_td
      4. Fumbles 3. Fumbles_Lost 3.;
```

The difference between using informat, format or a combination of the two is an important one. In this example by using informat the resulting dataset will look identical to the text file, making for easy comparisons.

START SMALL BY IMPORTING A SINGLE LINE OF TEXT THEN EXPAND

Starting small and breaking up a project into small, easily testable segments is a best practice useful in creating any complex solution; importing complex datasets is no exception. In the example provided, applying our decided upon formats and simply importing every line is an easy and effective way to test that the formats and column locations line up correctly.

```
DATA FootballInfo;
infile '\\Football base file.txt';

informat
      PlayerName $20. Position $2. Season 4. Team $25. GamesPlayed 4.
      Ru_Attempts 5. Ru_yds 10. ru_yards_game 6.2 ru_avg 6.2 ru_td 4.
      re_Receptions 5. re_targets 5. re_yds 10. re_yards_game 6.2
            re_avg 6.2 re_YAC 6.2 re_td
      4. Fumbles 3. Fumbles_Lost 3.;

input
      @1          PlayerName              $20.
      @21         Position                $2.
      @27         Season                  4.
      @34         Team                    $11.
      @45         GamesPlayed             4.
      @50         Ru_Attempts             5.
      @56         Ru_yds                  6.
      @62         ru_yards_game           6.2
      @68         ru_avg                  6.2
      @75         ru_td                   4.
      @80         re_receptions           5.
      @85         re_targets              5.
      @91         re_yds                  6.
      @97         re_yards_game           6.2
      @103        re_avg                  6.2
      @109        re_YAC                  6.2
      @116        re_td                   4.
      @123        Fumbles                 3.
      @127        Fumbles_Lost            3.;
run;
```

	PlayerName	Position	Season	Team	GamesPlayed	Ru_Attempts	Ru_yds	ru_yards_game	ru_avg	ru_td	re_receptions	re_targets	re_yds	re_yards_game	re_avg	re_YAC	re_td	Fumbles	Fumbles_Lost
1	Football Statistics																		201
2	SESUG2014																		
3	----------	--		----------															
4	Player	Po		Team															
5	Name																		
6	----------	--		----------															
7	Jamaal Charles	RB	2008	Chiefs	16	67	357	22.3	5.3	0	27	40	272	17	10.1	11.6	1	2	2
8		RB	2009	Chiefs	15	190	1120	74.7	5.9	7	40	56	297	19.8	7.4	6	1	2	2
9		RB	2010	Chiefs	16	230	1467	91.7	6.4	5	45	64	468	29.3	10.4	9.7	3	3	2
10		RB	2011	Chiefs	2	12	83	41.5	6.9	0	5	6	9	4.5	1.8	-0.4	1	1	1
11		RB	2012	Chiefs	16	285	1509	94.3	5.3	5	35	48	48	14.8	6.7	7.7	1	5	3
12		RB	2013	Chiefs	15	259	1287	85.8	5	12	70	104	693	46.2	9.9	9.6	7	4	2
13	Totals				80	1043	5823	72.8	5.6	29	222	318	1575	24.7	9.7	9.7	14	17	12
14	LeSean McCoy	RB	2009	Eagles	16	155	637	39.8	4.1	4	40	55	308	19.3	7.7	9.9	0	2	1
15		RB	2010	Eagles	15	207	1080	72	5.2	7	78	90	592	39.5	7.6	9.7	2	2	1
16		RB	2011	Eagles	15	273	1309	87.3	4.8	17	48	69	315	21	6.6	8.8	3	1	1
17		RB	2012	Eagles	12	200	840	70	4.2	2	54	67	373	31.1	6.9	9.6	3	4	3
18		RB	2013	Eagles	16	314	1607	100.4	5.1	9	52	64	539	33.7	10.4	12	2	1	1
19	Totals				74	1149	5473	74	4.8	39	272	345	2127	28.7	7.8	10	10	10	7
20	Adrian Peterson	RB	2007	Vikings	14	238	1341	95.8	5.6	12	19	28	268	19.1	14.1	15	1	4	3

Table4: Importing each row as one record regardless of all other factors.

Using this strategy of starting small it is possible to easily validate that the starting positions and informats chosen for each column are correct. A quick analysis of each observation shows that every column was imported correctly and

the data is in the correct formats and is displaying correctly. With this established, the task of eliminating the extraneous rows, breaking the text file into two datasets and combining seasons can be performed.

IDENTIFY THE CONDITIONAL LOGIC THAT MUST BE APPLIED

In our example there are a few condition we must identify in order to import the line correctly.

Conditions to identify in order to import the line correctly:

1. Identify the first row and import the date of the file (06/30/2014).
2. Identify the start or the end of a page.
3. Identify the total line associated with a particular player.
4. Continue assigning a player's name until all seasons are completed.

1: Identify the first row and import the date of the file (06/30/2014).

The first line of the text file will always be a header row so the code to grab the date is nothing special.

```
format ReportDate date9.;
retain ReportDate;

if _n_ = 1 then
      input
            @121        ReportDate              mmddyy10.
      /////;
else;
```

A format statement is used so the date is not displayed as number of days and _n_ = 1 establishes when the first line of text is being imported. All the header information is extraneous, therefore the '/' is used to dropdown 5 lines. Additional logic statements could be used in lieu of the 5 '/', however this solution is very simple and as long as the page headers are standardized should work in all cases. The first line of data is 6 lines down from the start of the page header, but the input statement itself will move to the next row of text upon completion. The 5 '/' and the input statement itself advancing make up the 6 rows required to get to the first row of data.

2: Identify the start or the end of a page.

Similar to what was done with the very first line of the text file, it will be necessary to skip the header row for each page of the file. This can be done by either identifying the start or the end of a page. The start of a page can be identified by the text 'Football Statistics'. The end of a page is a little more involved and can be done in a number of different ways. For this example the end of a page is going to be identified as a Totals line that is immediately followed by a blank line.

3: Identify the total line associated with a particular player.

The Totals line is also easy to identify. We can import the same range as PlayerName and look for a value equal to 'Totals'.

4: Continue assigning a player's name until all seasons are completed.

Each player has at least two lines associated with him. To be included on this file they have played for at least one season and will include a total line where each value will be equal to the value for their one season of play. The logic for this item will again check the same range as PlayerName and will continue importing seasons until it finds a value of 'Totals'

If the different logical conditions are broken up into small enough pieces they become very effortless to code. The key is analyzing all the patterns beforehand and separating them into their individual components.

PUTTING EVERYTHING TOGETHER

From our four conditional statements above we now know the general form that the import process will take.

1. Import the report date from the first row of the file.
2. Skip down 6 rows to the first row of data (skip the header).
3. Import player data for one season.
4. Output dataset for season statistics.
5. Look ahead to next line, if it is another season of data, go back to step #3, otherwise continue.
6. Import the totals line.
7. Output dataset for career totals.
8. Look ahead to next line, if it is another player go back to step #3, if it is the end of a page go back to step #2, if it is the end of the file import completed.

Once broken down to the most basic elements, each additional step and condition is simple. The final code is shown below:

```
DATA
       FootballInfo (drop = LookAhead _Season _Team
              _GamesPlayed  _Ru_Attempts  _Ru_yds  _ru_yards_game  _ru_avg
              _ru_td  _re_Receptions  _re_targets  _re_yds  _re_yards_game
              _re_avg _re_YAC _re_td _Fumbles _Fumbles_Lost)
       Totals (drop = position Season _Season _Team _GamesPlayed
              _Ru_Attempts _Ru_yds _ru_yards_game _ru_avg _ru_td
              _re_Receptions _re_targets _re_yds _re_yards_game
              _re_avg _re_YAC _re_td _Fumbles _Fumbles_Lost LookAhead);
infile '\\Footbal base file.txt' truncover;

format ReportDate date9.;

informat
       /*Establishing formats for imported variables*/
       LookAhead $20.
       PlayerName $20. Position $2. Season 4. Team $25.
       GamesPlayed 4. Ru_Attempts 5. Ru_yds 10. ru_yards_game 6.2
       ru_avg 6.2 ru_td 4. re_Receptions 5. re_targets 5.
       re_yds 10. re_yards_game 6.2 re_avg 6.2 re_YAC 6.2 re_td 4.
       Fumbles 3. Fumbles_Lost 3.

       /*Temporary variables in the event a player was traded.     When
       this occurs it is necessary to import an additional ros of
       data and combine the two rows into one. These are dropped
       in the final data set*/
       _Season 4. _Team $25. _GamesPlayed 4.
       _Ru_Attempts 5. _Ru_yds 10. _ru_yards_game 6.2 _ru_avg 6.2
       _ru_td 4. _re_Receptions 5. _re_targets 5. _re_yds 10.
       _re_yards_game 6.2 _re_avg 6.2 _re_YAC 6.2 _re_td 4.
       _Fumbles 3. _Fumbles_Lost 3.;
```

```
      /*ReportDate: Is imported from the first row and applied to every row in the
      dataset.
      PlayerName: This variable is imported from the first row of each player and
      applied to each row of data.*/
retain ReportDate PlayerName;

/*Read in first 20 characters and hold the input record.
this will be used by conditional logic to decide how to continue.*/
input
      @1           Lookahead           $20.   @;

/*You are at the first row of the report.
Grabs ReportDate and jumps down to the first row of data.*/ if _n_ = 1 then
do;
      input
            @120         ReportDate          mmddyy10.
            //////
            @1           PlayerName          $20.   @;

end;

/*Importing a standard row of player data*/
input
      @21          Position                 $2.
      @27          Season              4.
      @34          Team                     $11.
      @45          GamesPlayed              4.
      @50          Ru_Attempts              5.
      @56          Ru_yds              6.
      @62          ru_yards_game       6.2
      @68          ru_avg              6.2
      @75          ru_td                    4.
      @80          re_receptions            5.
      @85          re_targets               5.
      @91          re_yds              6.
      @97          re_yards_game       6.2
      @103         re_avg              6.2
      @109         re_YAC              6.2
      @116         re_td                    4.
      @123         Fumbles                  3.
      @128         Fumbles_Lost        2.;

/*Read in first 20 characters and hold the input record.
The @@ must be used here because the first 20 characters can be
                   blanks. The @@ ensures
entirely made up of  won't advance to
the next row looking for data and skip observations. This will
be used by conditional logic to decide how to continue.*/
input
      @1           Lookahead                $20.
      @27          _season                  4.     @@;

/*This checks for a player who has been traded and played for multiple teams in
the same season. The additional season will be imported then combined.*/
if _season = Season then do; input
            @34          _Team                    $11.
            @45          _GamesPlayed        4.
            @50          _Ru_Attempts        5.
            @56          _Ru_yds                  6.
            @62          _ru_yards_game           6.2
            @68          _ru_avg                  6.2
```

```
            @75         _ru_td              4.
            @80         _re_receptions            5.
            @85         _re_targets               5.
            @91         _re_yds                   6.
            @97         _re_yards_game            6.2
            @103        _re_avg                   6.2
            @109        _re_YAC                   6.2
            @116        _re_td              4.
            @123        _Fumbles                  3.
            @128        _Fumbles_Lost       2.;

      Team  =  strip(Team)  ||  "/"  ||  strip(_Team);  GamesPlayed  =
      GamesPlayed  +  _GamesPlayed;  Ru_Attempts  =  Ru_Attempts  +
      _Ru_Attempts; ru_yds = ru_yds + _ru_yds;
      ru_yards_game = ru_yards_game + _ru_yards_game; ru_avg = ru_avg +
      _ru_avg;
      ru_td = ru_td + _ru_td;
      re_receptions = re_receptions + _re_receptions; re_targets =
      re_targets + _re_targets;
      re_yds = re_yds + _re_yds; re_YAC = re_YAC +
      _re_YAC; re_td = re_td + _re_td; fumbles =
      fumbles + _fumbles;
      Fumbles_lost = Fumbles_lost + _Fumbles_lost;

      /*The data step should always be ended looking at a row of player data. After
      importing and combining, the next row is imported to be used by conditional
      logic to decide how to continue.*/
      input
            @1          Lookahead                 $20.
            @27         _season                   4.    @@;
end;

/*The FootballInfo observation is output. This has to go after the code section
looking at multiple seasons to ensure the correct record is generated.*/
output FootballInfo;

/*Identifies totals row and imports according.*/ if Lookahead =
'Totals' then do;
      input
            @21         Position                  $2.
            @27         Season              4.
            @45         GamesPlayed               4.
            @50         Ru_Attempts               5.
            @56         Ru_yds              6.
            @62         ru_yards_game       6.2
            @68         ru_avg              6.2
            @75         ru_td                     4.
            @80         re_receptions             5.
            @85         re_targets                5.
            @91         re_yds              6.
            @97         re_yards_game       6.2
            @103        re_avg              6.2
            @109        re_YAC              6.2
            @116        re_td                     4.
            @123        Fumbles                   3.
            @127        Fumbles_Lost        3.;
      output Totals;

      /*Looks ahead to be able to determine if it is the end of
```

```
        a page or there is another row of player data.*/
        input
        @1                  Lookahead                   $20.   @@;

        /*If the end of a page, skip down to the next row of data
        and grab Playername.*/
        if Lookahead = '' then
               input
                      ////////
                      @1 PlayerName $20. @@; /*If not at the end of a
        page, then the lookahead variable
                      is equal to the next players name.*/
        else
               PlayerName = Lookahead;
end;

run;
```

SAS affords the flexibility to do things many diff erent ways, and there is any number of solutions to importing the sample file. The code provided is one possible way that is very straight forward and does not have a lot of code repetition. One of the major design decisions is to decide if the iteration of the DATA step should end on the next valid line of data or if the DATA step should determine the correct place to import at the beginning of the iteration. That choice has the most impact on the structure of the end result; neither is right or wrong. In this example we choose the former and ensured we were ending the iteration of the DATA step at the start of a valid observation.

CONCLUSION

The example provided was a good demonstration of the power of using @ and @@. The football statistics file was a straightforward example, but hopefully it can be seen how the process used can be extrapolated and expanded upon to fit a variety of business cases. A high-level real world example is provided below to compare and contrast.

The following credit reporting file is
generated: **Page starts with Header row.
**First line of record is primary mortgagor relevant information.
**Record continues with additional mortgagor relevant
information. **Record finishes with general account information.

The primary mortgagor information is included on every account. If any additional mortgagors are present they are included one line at a time following the primary mortgagor. Immediately following the last additional mortgagor or immediately following the primary mortgagor if no additional mortgagors are present, is the general account information. This is a very realistic general format for a flat file to take and is not unlike our football statistic example. The rules for this import spec would most likely look something like this:

1. Identify the start of the page and drop down the correct number of lines to the data.
2. Import the primary account information
3. Check next line for additional mortgagor. If additional mortgagor present import data and repeat this step.
4. All additional mortgagors imported so this line must be general account information. Import.
5. Advance to next observation or start of a new page so DATA step can process the next import.

From a high-level these solutions are identical. The code to solve both of these problems is included in this paper and would only require minor changes to solve either problem. This general structure established here should be able to be modified for the vast majority of business cases involving any complex text file. If the complex file can be broken down into the small pieces that are easy to validate and the criteria that must be followed the process becomes virtually effortless to implement.

CONTACT INFORMATION

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

Name:	Russell Woods
Enterprise:	Wells Fargo Home Mortgage
City, State ZIP:	Fort Mill, South Carolina
E-mail:	rjwood02@gmail.com