How to Efficiently Handle Large Datasets in SAS

Large datasets occupy tons of storage, take an annoyingly long time to process, and consequently become a bottleneck in meeting deadlines. Compared to Excel, SAS supports larger datasets and accomplishes more complex analyses. Therefore, SAS is widely used for business analytics and statistical analysis, and is often the preferred platform for handling large datasets. In SAS, one can easily process files with millions of rows and hundreds of columns, such as company transactions, insurance claims, international trade data, investment flows, and stock prices. Here are some tips for boosting your productivity when processing large datasets in SAS.

STEP 1. IMPORTING DATA

SAS is powerful in manipulating and analyzing large datasets. Importing those huge files into SAS, however, can be frustratingly slow. Luckily, there are ways to speed up the process!

- To save time, convert **xlsx** or **xlsb** files into **csv** or **txt** files before importing. Note that each sheet in an **xlsx** file needs to be saved in a separate **csv** file.

XLXS	CSV ✓

`PROC IMPORT DATAFILE="Test.xlsx"`
`OUT=test_xlsx DBMS=EXCEL REPLACE;`
`SHEET=”Sheet1”;`
`RUN;`
	`PROC IMPORT DATAFILE="Test_1.csv"`
	`OUT=test_csv_1 DBMS=CSV REPLACE;`
	`RUN;`

Time: 8:08.29 *minutes*	Time: 12.02 seconds

- To save space, use commands such as “COMPRESS” on character variables and “LENGTH” on numeric variables. The compressed data will save run time in subsequent steps.

COMPRESS=NO	COMPRESS=YES ✓

`PROC IMPORT DATAFILE="Test_1.csv"`
`OUT=test_csv_1 DBMS=CSV REPLACE;`
`RUN;`
	`OPTIONS COMPRESS=YES;`
	`PROC IMPORT DATAFILE="Test_1.csv"`
	`OUT=test_csv_1 DBMS=CSV REPLACE;`
	`RUN;`

Data Size: 370.4MB	**Data Size: 224.9MB** (size decreased by 39.29 percent)

STEP 2. CLEANING DATA

Data files are likely to be formatted differently and difficult to compile. To organize the imported datasets, you may have to concatenate, standardize, or aggregate them. Again, this data cleaning process can be made efficient.

- To save space, use “DROP” and “KEEP” to subset the data and keep only the key information.

CONCATENATE	CONCATENATE + DROP ✓

`DATA test;`
`SET test_csv_1 test_csv_2 test_csv_3 test_csv_4 test_csv_5 test_csv_6 test_csv_7 test_csv_8 test_csv_9 test_csv_10;`
`RUN;`
	`DATA test_drop(DROP=VAR1-VAR5);`
	`SET test_csv_1 test_csv_2 test_csv_3 test_csv_4 test_csv_5 test_csv_6 test_csv_7 test_csv_8 test_csv_9 test_csv_10;`
	`RUN;`

Data Size: 2.2GB	Data Size: 1.9GB (size decreased by 13.36 percent)

- To save time, favor "WHERE" over "IF." "WHERE" loads only the subset of data, whereas "IF" loads the entire dataset and then subsets.

IF	WHERE ✓

`DATA test_if;`
`SET test_drop;`
`IF QTY<10000;`
`RUN;`
	`DATA test_where;`
	`SET test_drop (WHERE=(QTY<10000));`
	`RUN;`

Time: 58.30 seconds	Time: 39.07 seconds

- To save space, aggregate transaction records while preserving sufficient details (i.e. key identifiers such as time and name). Some handy tools are "PROC SQL," "PROC MEANS," and "PROC SUMMARY."

Before Aggregation							After Aggregation ✓					
DATE	**DRUG**	QTY	COST	PAY	COPAY		**DATE**	**DRUG**	QTY	COST	PAY	COPAY
1/2/2010	**A**	30	$167	$150	$17		**1/2/2010**	**A**	90	$501	$450	$51
1/2/2010	**A**	60	$334	$300	$34		**1/2/2010**	**B**	60	$178	$174	$4
1/2/2010	**B**	30	$89	$87	$2							
1/2/2010	**B**	30	$89	$87	$2							

STEP 3. ANALYZING DATA

With the cleaned datasets, you are now ready to perform your analysis! You can still easily cut run time when computing summary statistics and joining datasets.

- To save time, avoid sorting data when you only need to group them. For instance, use "CLASS" rather than "BY" in "PROC MEANS" and "PROC SUMMARY."

PROC MEANS + BY	PROC MEANS + CLASS ✓

`PROC SORT DATA=test_drop OUT=test_drop_s;`
`BY DATE DRUG;`
`RUN;`
 	
`PROC MEANS DATA=test_drop_s NWAY NOPRINT;`
`BY DATE DRUG;`
`VAR UNITS BILL COST PAY COPAY;`
`OUTPUT OUT=test_means_by SUM= ;`
`RUN;`
	`PROC MEANS DATA=test_drop NWAY NOPRINT;`
	`CLASS DATE DRUG;`
	`VAR QTY BILL COST PAY COPAY;`
	`OUTPUT OUT=test_means_class SUM= ;`
	`RUN;`

Time: 1:44.28 *minutes* (1:23.69 minutes + 20.59 seconds)	Time: 28.62 seconds

- To save time, be smart about the tools you choose to merge datasets. Typically, merging datasets using “PROC SQL + JOIN” requires simpler code and results in shorter run time than “DATA + MERGE.” However, for large datasets, merging with “DATA + MERGE” is relatively faster.

DATA + MERGE	PROC SQL + JOIN

`PROC SORT DATA=test_drop OUT=test_drop_s;`
`BY DRUG GROUP ID_NO; RUN;`
 	
`PROC SORT DATA=drug_index OUT=drug_index_s;`
`BY DRUG GROUP ID_NO; RUN;`
 	
`DATA test_merge;`
`MERGE test_drop_s drug_index_s;`
`BY DRUG GROUP ID_NO;`
`RUN;`
	`PROC SQL;`
	`CREATE TABLE test_SQL AS`
	`SELECT a.*, b._FREQ_`
	`FROM test_drop AS a LEFT JOIN drug_index AS b`
	`ON a.DRUG =b.DRUG AND a.GROUP=b.GROUP AND`
	`a.ID_NO =b.ID_NO;`
	`QUIT;`

Data Size: 200.6MG	
Time: 24.27 seconds (22.02 seconds + 0.00 seconds + 2.25 seconds)	Time: 9.28 seconds ✓
Data Size: 2GB	
Time: 2:16.07 *minutes* ✓ (1:32.66 *minutes* + 0.86 seconds + 42.55 seconds)	Time: 2:45.27 *minutes*

Finally, here are some additional tips:

- Run heavy batches overnight. SAS never sleeps, but your annoying colleagues who clog up the server CPU do.
- Do test runs with only a few rows (e.g. “OBS=1000”). This saves time in editing and debugging your code.
- Set up SAS “EMAIL” notifications, which will notify you when your program has run.

Kylie Zhang is an analyst at Epsilon Economics specializing in data analytics and statistical analysis. She enjoys turning data into actionable insights. She received her M.P.P. degree with a focus on applied Economics and Econometrics from the University of Chicago. Epsilon Economics is a firm of economists and financial analysts specializing in providing economic analysis, analytics, and strategy to Fortune 500 companies and leading law firms around the world.