It's 11 pm - Do you know what your SAS® users are doing tonight?

June Genis

System Product Support Group
Stanford University Data Center

Abstract

The Stanford Data Center has installed code to utilize one of the user exits which are currently defined in *Technical Report Y-108: User Exit Facilities for the SAS System Version 5*. Our exit produces a local SMF record type when a SAS PROC or DATA step is invoked in MVS batch or TSO. These records are in turn analyzed with SAS. This paper summarizes the difficulties we encountered in implementing the exit, what we have been able to learn about our SAS user community from the data, and what additional data we would like to have but does not seem to be available.

Introduction to the User Exit Facility

SAS defines a series of user exits that allow an installation to tailor and extend the SAS system supervisor functions on IBM operating systems. These exits are documented in *Technical Report Y-108* for version 5.18 of the SAS System.[1] They are implemented in the module SASUSER either by direct modification to the SASUSER code or by calls to user written assembler routines of predefined name. The calls to the external routines in the distributed version of SASUSER are normally disabled, but can be enabled by using IMASPZAP. The exits allow the installation to take control at any of the following points in the execution of a SAS job:

- at initialization (i.e. on entry to SASUSER)
- at termination (just prior to supervisor termination)
- after default and initiation time system options are processed
- when raw data files are accessed
- when a TP monitor command (e.g. TSO or CMS) is issued
- when a statement can not be recognized
- after the procedure name has been scanned but prior to its invocation (to check for procedure authorization)
- when a SAS library is accessed
- at the end of a procedure (for step accounting)
- after an INFILE record has been read
- at a number of places when using SAS/SHARE®

The implementation discussed in this paper makes use of only one of these exits, the step accounting exit, however the general procedures and caveats for working with the exits apply to all of them.

The Stanford Data Center Environment

At the time we undertook this project the Stanford Data Center was operating an IBM 3090-200E, since upgraded to a 3090-400E, running the MVS/XA operating system. Although our primary charge is to support the University's Administrative Computing function, we also supply open shop access to any Stanford community member with a billable source of funds. We are a charge back facility with a highly developed accounting system based on the generation and processing of SMF records. Already existing reports based on type 4 jobstep accounting records were able to provide information on the number of SAS invocations each month, their total cpu consumption and the account to be charged for it. These reports show that in a typical month we will have between 160 and 180 accounts running 7000 to 9000 SAS jobsteps which consume 2500 to 3000 cpu minutes.

Although type 4 records do not include TSO usage, we already knew from other data that almost all SAS usage is batch mode, or more accurately is run via remote job submission and retrieval from WYLBUR®, a component of the Stanford Timesharing System. SAS has for some time been the single largest consumer of cycles outside of the time sharing system.

[1] The original documentation on which this work was based was Technical Report Y-103 of the same title.

The existing reports were unable, however, to supply any information on how usage divided between component products such as SAS/ETS® and SAS/OR® or what specific procedures were being invoked from each product.

Data Center Implementation

What we wanted was something which summarized information about the resources consumed by particular procedures. Since we already had a system in place to manage SMF records, including procedures for creating new local record types through an SMFWRITE SVC, we decided to create a local SMF record, type 209, from the step accounting exit, SASUPRAC. Actual coding of the exit was done by Susan Plass, currently manager of the System Products Support Group. The analysis programs were done by the author. Both are available on request to the author.

The step accounting exit is invoked at the end of each SAS procedure or DATA step. Many type 209 records could therefore be anticipated from a single MVS jobstep. We had no idea at the beginning of this effort how many that might turn out to be. We considered the possibility of storing intermediate results in memory and creating a single combined record at the end of the jobstep. This approach would have been considerably more complex, involving at least the use of the initiation and termination exits in addition to the step accounting exit and would have resulted in a more complicated record to be parsed by the analysis programs. As all of these factors add up to additional programmer time against the expense of additional record storage space we decided to go with the simpler approach until space became a problem.

The SASUPRAC exit

When SAUPRAC is invoked the address of an accounting parameter block (mapped by the SASUAPB macro, found in the macro library on the distribution tape) is passed to the exit in R1. This control block contains the name of the procedure (SASDATA is used for a DATA step), the cpu time used in OS timer units, the required memory in bytes, and the number of pages printed to the SAS PRINT file. SASUPRAC is only invoked when an enabling zap had been applied to SASUSER.

The following zap, which can be applied with IMASPZAP, SUPERZAP or other such utility causes SASUSER to load and call the installation-written step accounting exit module SASUPRAC:

```
NAME SAS SASUSER
VER 0490 47F0
REP 0490 4700
```

The address shown above is for Version 5.18. This address has changed for every release since 5.08 and will without doubt be different for Version 6 as well.

Record description

The result of a call to our SASUPRAC exit is the following SMF record as defined by the INPUT statement used to read it in the analysis program.

```
  /*  DEFINITION OF SAS USER EXIT RECORD                */
  /*  TYPE 209                                          */
INPUT
  /* HEADER FOR ALL SMF RECORDS */
     @2 ID PIB1.              /* RECORD TYPE (209) */
     @3 TIMEDATE SMFSTAMP8. /* time and date written */
     @11 SYSTEM   $4.
  /* HASPSERV DATA */
     @15 JOBNUMBR $4.
     @19 JOBNAME $8.
     @27 USER $3.
     @30 GROUP $2.
     @32 SUBGROUP $2.
  /* STANFORD SAS RECORD HEADER */
     @34 SUBTYPE PIB1.    /* X'01' is step exit */
     @35 RESERVED $2.     /* force full word alignment */
  /* SAS USER ACCOUNTING PARAMETER BLOCK */
     @37 PROCNAME $8.     /* SAS procedure name */
     @45 CPU  TU4.        /* TCB time in timer units */
     @49 REGION   PIB4.   /* in bytes */
     @53 PAGES  PIB4.     /* pages printed by this proc */
;
```

User data such as job name and number are retrieved with another standardized routine, implemented through an SVC, which retrieves information from JES2. A subtype structure was included to allow for the addition of future exits all collected under a single SAS SMF record type.

Implementation Problems

Although *Technical Report Y-108* describes the way exits are invoked it does not specify anything about general register conventions on entry and return from the exit.

It can not be assumed that standard linkage conventions should be used, since IBM's own exits do not use standard IBM subroutine linkage conventions. This means that an implementer must go to the source code of SASUSER in an attempt to determine the linkage conventions in effect.

The following comments were extracted from the SASUSER source code:

```
*************************************************************
*                                                           *
*  EXCEPT FOR THE INIT AND TERM EXITS, ALL EXITS ARE        *
*  INVOKED AS FOLLOWS:                                      *
*                                                           *
*           <LOAD PARAMETERS>                               *
*           SASEXIT <ENTRY CODE>                            *
*          +LA     R0,<ENTRY CODE>                          *
*          +L      R15,$USERXIT                             *
*          +BALR   R14,R15                                  *
*                                                           *
*  EXITS ENTERED IN THIS WAY DO NOT REQUIRE A SAVEAREA      *
*  AND ON RETURN WILL RESTORE ALL REGISTERS EXCEPT R15.     *
*                                                           *
*************************************************************
```

The last two lines of this comment are incorrect.

In fact, code in SASUSER immediately following the call to SASUPRAC and before registers are restored, counts on the contents of R11, R13, and possibly R12 being the same as they were when the exit was called. All other registers except R15 are restored. We therefore adopted the following conventions.

Registers at entry:

R0 Contents are unpredictable

R1 Address of SASUAPB control block

R2-R13 Contents are unpredictable

R14 Return address

R15 Entry address

Registers at exit:

R0-R10 Contents are unpredictable

R11-R13 Contents are unchanged

R14 SASUSER return address

R15 Return code indicating:

0 record written without error

Sample Graphical Reports

We are still experimenting with analysis of the exit records. The following reports are ones which have proven interesting.

The two plots in Figure 1 represent summary data, by both frequency of invocation and total cpu consumed, for all of 1988. The bar for each month is sub-grouped by the contribution of each SAS product component which we currently license. This sub-grouping was made possible by merging the SMF records with a data set whose records contain the name of each SAS PROC and the product in which it is found.[2]

It is readily visible that measured by either parameter the vast bulk of our usage results from the BASE product. This can be further observed in the PIE charts shown in Figures 2 and 3 where usage for the month of July is broken down by component products. The single plot at the top of each figure represents total SAS System usage, by frequency and cpu sum respectively. The two rows of plots below it are the result of a BY grouping on product. Even at this rather compact scale it can easily be seen that the profile for the BASE product is almost identical to that for total usage.

Looking at the enlarged display for the BASE product in Figure 4 we can further see that the vast bulk of this usage is in turn the result of DATA step use. It would indeed be most inappropriate, therefore, to categorize the SAS system as a "statistical analysis system," at least as it is used at our installation. In fact, when PROC SORT, clearly another data management function, is combined with the DATA step they comprise two thirds of the number of steps and three quarters of the cpu time consumed.

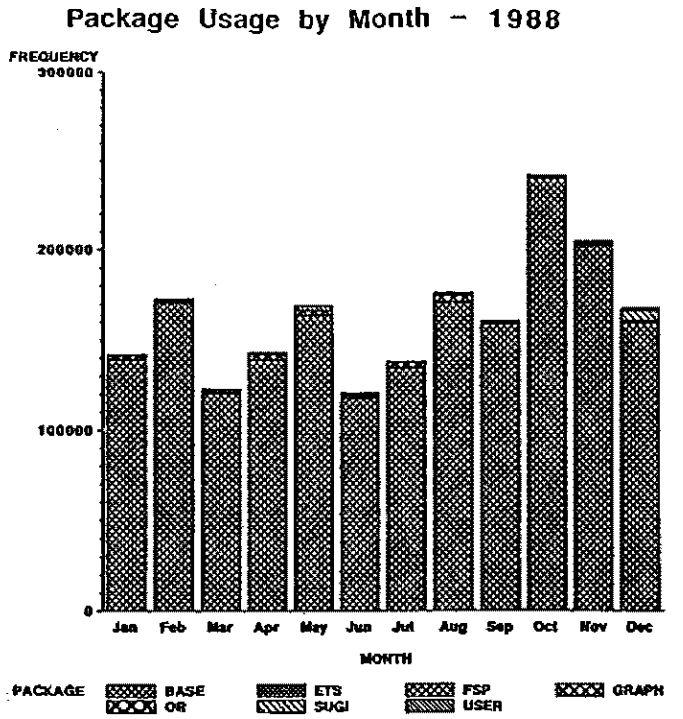
What we have learned from the graphs

Since we are a charge back facility, the cost of using SAS is never far from the minds of our users. Given the profile revealed by this data it appears that our users have a vested interest in improved efficiency of data step processing. This will be reflected in the votes of support staff on future software ballots.

We are also being asked to do more zero based budgeting for software expenditures. Since all SAS components are priced the same we are clearly getting a much better return on BASE than on the other product components we license. In the future we will at a minimum have to demonstrate that the additional products are at least bringing in sufficient revenue to cover their licensing costs. If not, we may have to use our new record type as the basis for surcharging in order to keep

---

[2] This file was originally built for 5.08 by dumping and editing the directory information for each of the HELP files found on the distribution tape. It has since been updated manually as needed.



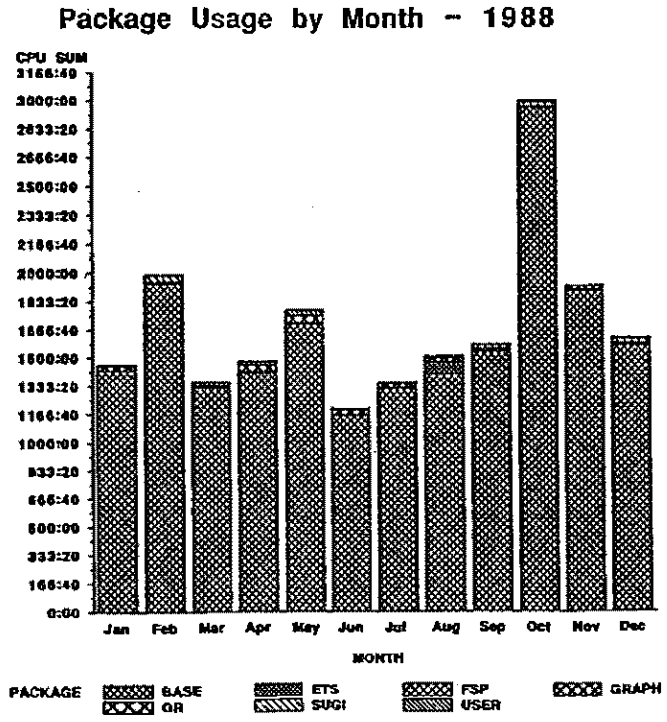


Figure 1: Total SAS Usage for 1988

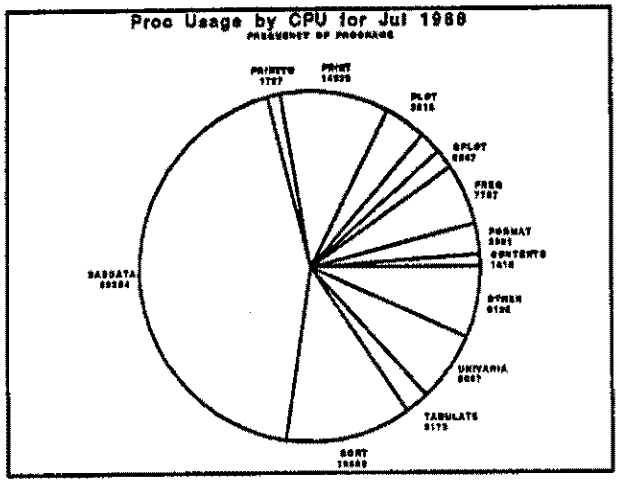
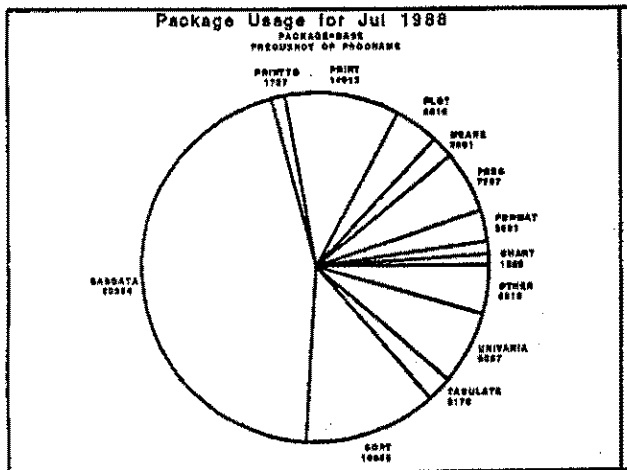
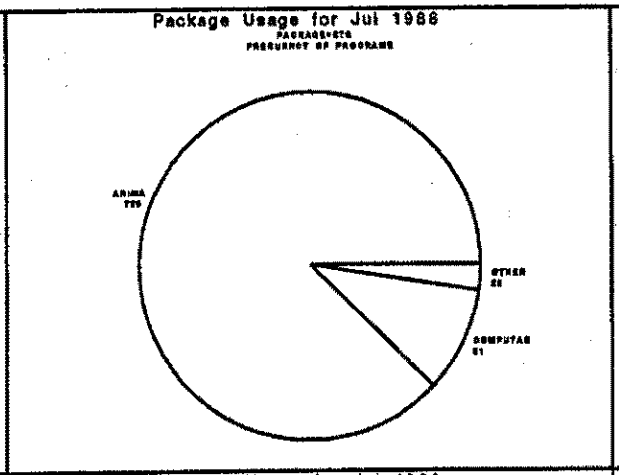
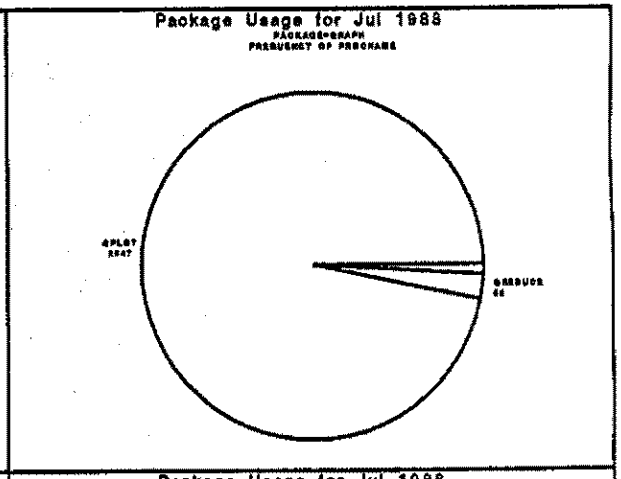
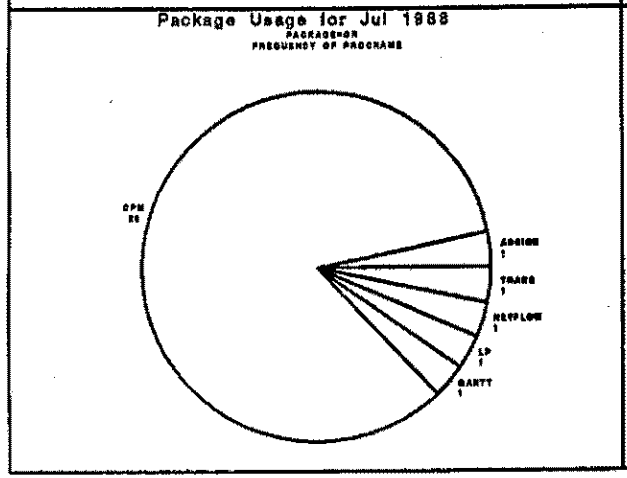


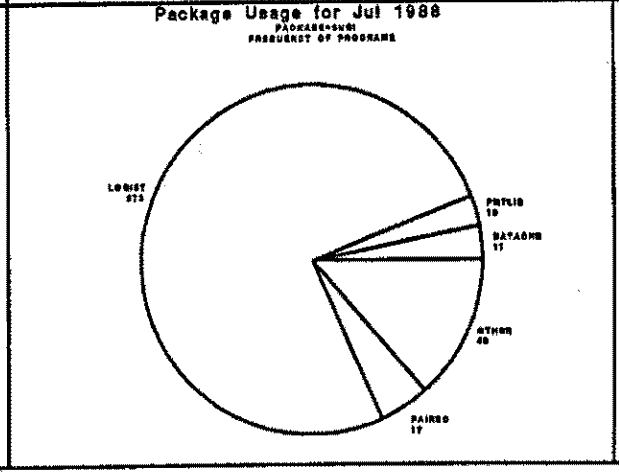
Figure 2: SAS Usage for July 1988 by Frequency

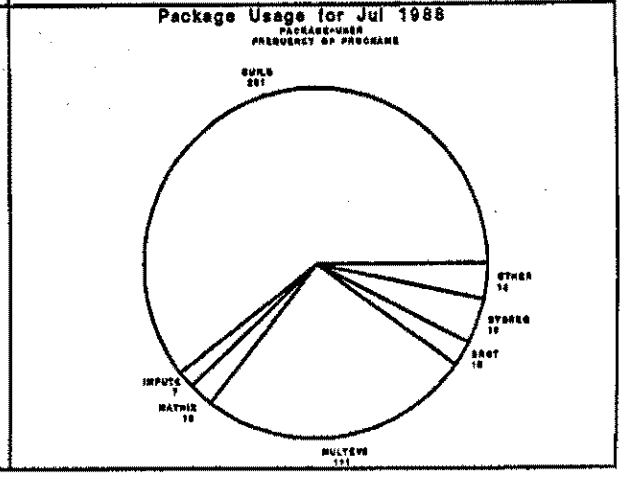












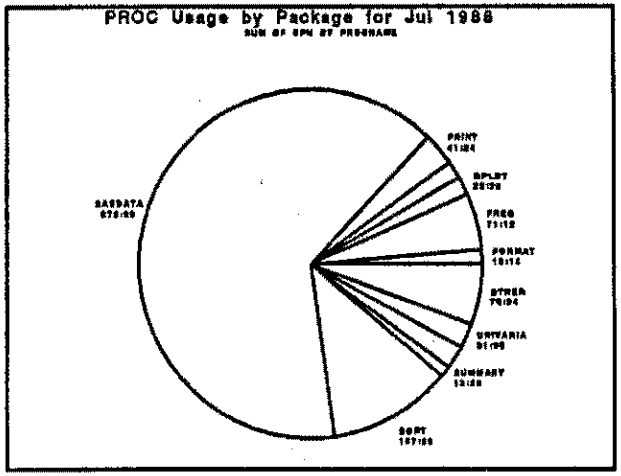
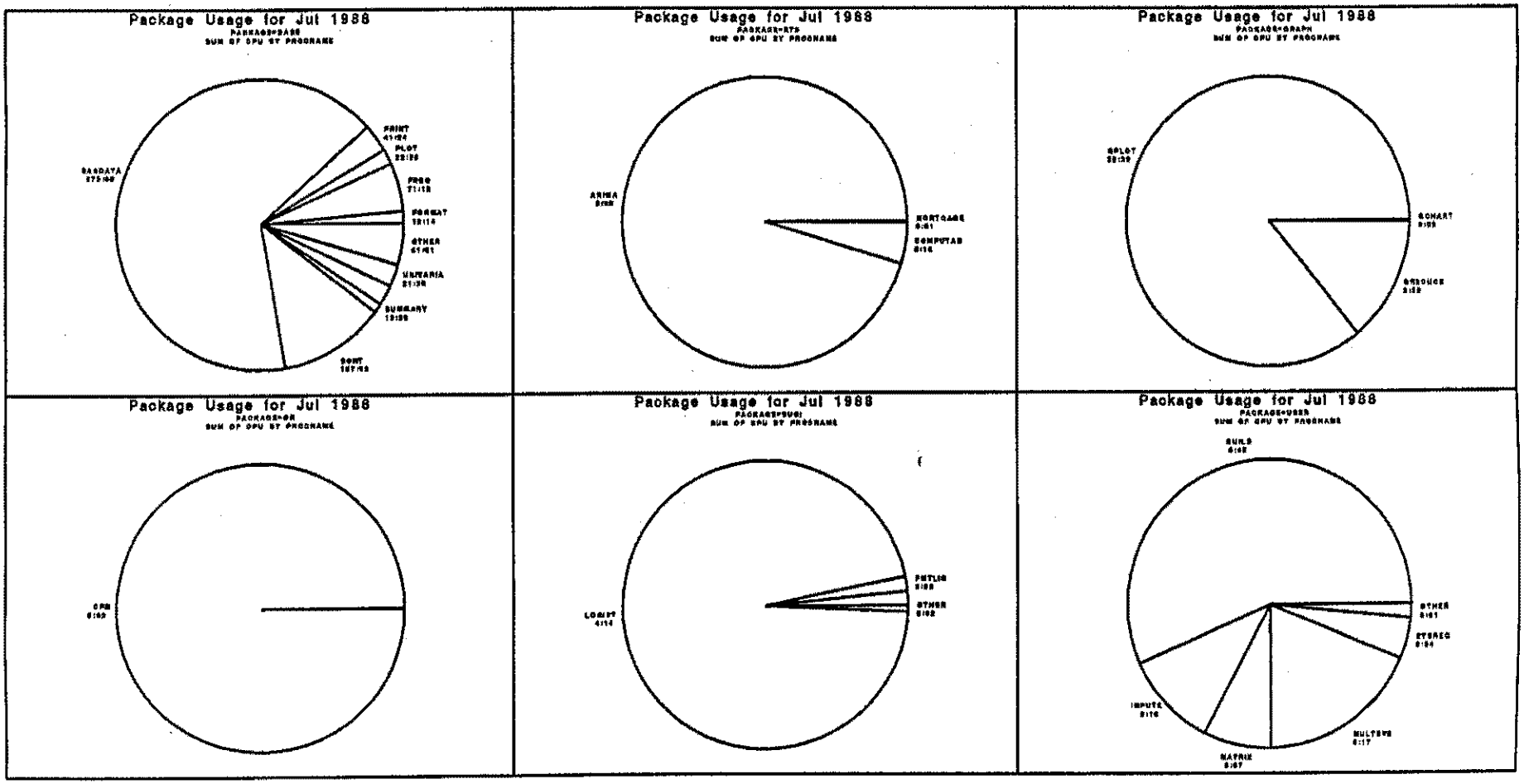


Figure 3: SAS Usage for July 1988 by CPU Sum



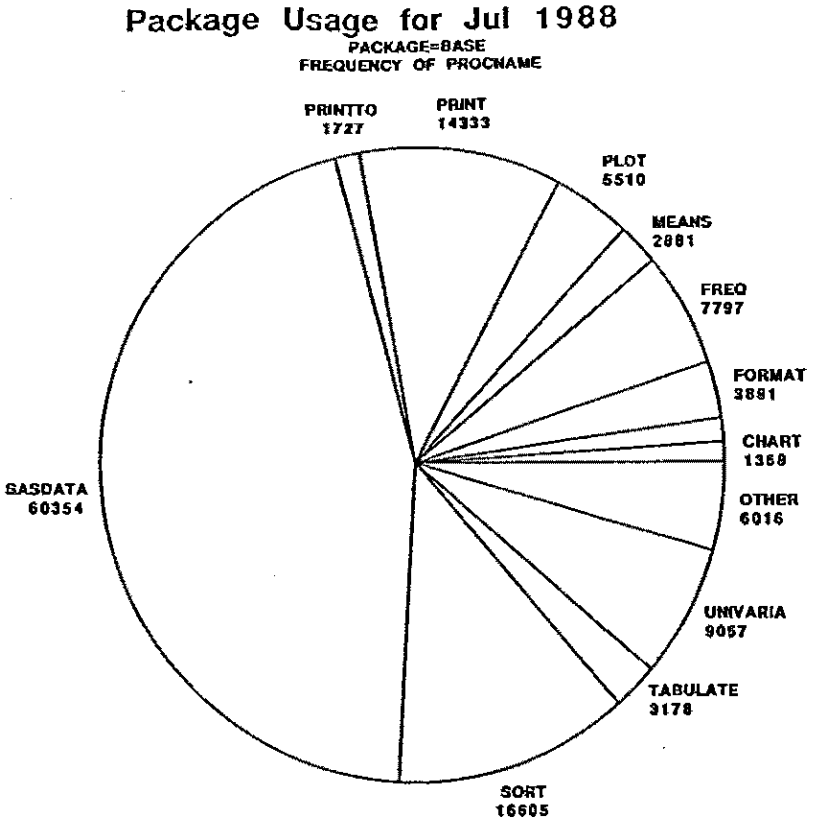
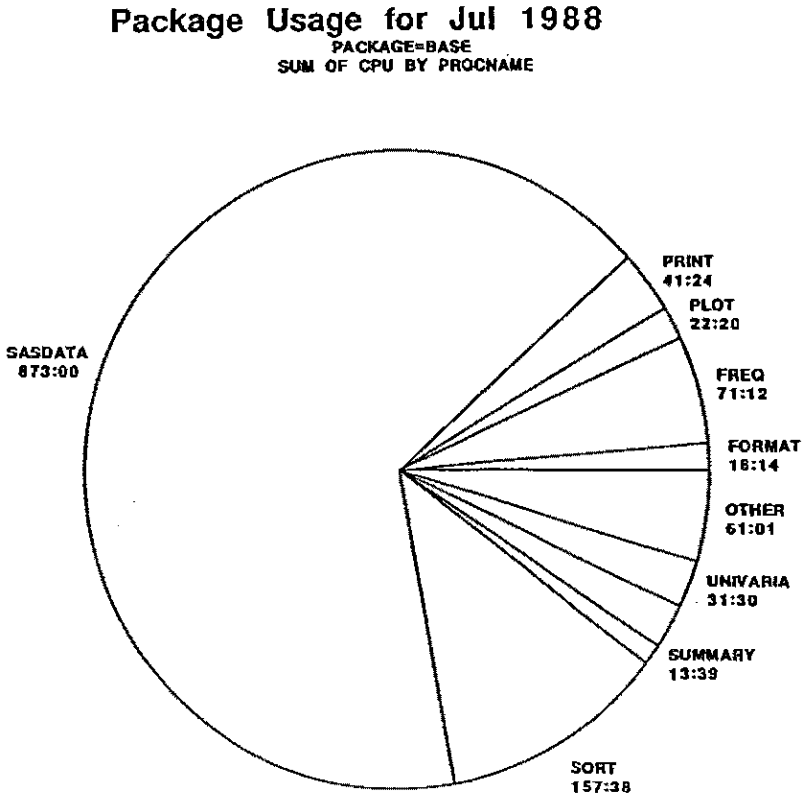


Figure 4: BASE Product Usage for July 1988



these products. If surcharging is not acceptable as a means of recovering costs, then the continuation of the products may be in jeopardy.

If we are going to consider dropping any component we will need more specific information than the summary plots contain since we will need to find all users of the product to discuss alternative ways to accomplish their tasks. For such detailed data we must drop back from graphics and look at a TABULATE analysis.

TABULATE Reports

Figure 5 shows a sample page from a TABULATE report which is summarized by user account within each procedure. For the purpose of understanding the report we will focus on the FMTLIB usage of account GA.ALS which belongs to a staff member. On this report we are able to identify that:

- user GA.ALS is a user of FMTLIB
- in the month under examination he ran one job with two invocations of FMTLIB
- that his usage represents 18.50% of all FMTLIB usage
- which in turn is .31% of all use of the Supplemental Library
- and is a negligible amount of total SAS usage.

To understand the importance of FMTLIB in this user's overall SAS work profile we must now look at a breakdown by user which is shown in Figure 6. Here we can see that FMTLIB is the only Supplemental routine which he used in July 1988 and that it represents only a fraction of one percent of his total SAS usage.

We can also use TABULATE to provide more detail about the data found in figures 1 through 4. In Figure 7 we see a portion of a monthly breakdown summarized by procedure within component which has not been further subdivided by individual user. Here we can see both that BASE represents over 97% of total cpu usage for the month as well as details about procedures invoked from the other product components. The previous page of this report would have shown the same sort of detail for the BASE product.

It must be pointed out that some of the fields on these TABULATE reports are actually meaningless. TABULATE is unable to tell that the job in which GA.ALS used FMTLIB was the same one in which he also ran a SORT and a PRINT step. Therefore, it will double count "Jobs" in the Product Total summary and triple count it when summarizing all GA.ALS usage. The author of this paper knows of no current cure for this problem except white-out and would appreciate hearing from anyone with a proposed solution.[3]

What's still missing

Most of what we can not currently obtain appears to be related to insufficient information being placed in the SASUAPB control block which is made available to the step accounting exit. In particular we would like more information about graphics usage such as what device driver was being used, how many separate plots were produced and whether the output was directed to a catalog, a local graphics device or a remote one. Since a graphics device must be opened before use, it may be more appropriate to capture some of this information at that point rather than at step termination.

We would also like to know more about how users are are making use of the RLINK facility. Because TSO usage is currently so low we have not yet been able to determine if SMF records are being generated when services such as RSUBMIT, UPLOAD and DOWNLOAD are requested via an RLINK connection. Even if mainframe records are created, they would not tell us anything about the amount of data being transmitted over the link.

More information about data set usage by procedures would also be useful for mainframe only jobs. Inclusion of the data set names referenced for the input and output of each step might allow us to detect situations were users consistently made more passes than necessary over their files. Such problems could then be addressed by modifying our training programs and local documentation. Reporting the magnitude of the data sets involved would give us some feeling for how much of the workload might be migrated to other platforms such as the IBM/PC.

Conclusion

The User Exit Facility provides a useful way of obtaining more information on SAS workload characteristics than can be obtained from standard operating system facilities. It could be made even more useful by improved documentation, more exits and increased information passed to the step accounting exit.

[3] The Jobs column is computed by counting a variable which is set to 1 for the first occurrence of a procedure name within a job and 0 elsewhere.

YEAR 1988
AND MONTH July

			Occurrence in		CPU Time in seconds						
			Jobs	Steps	MEAN	MIN	MAX	SUM	% of Proc	% of Product	% of all SAS
PACKAGE	PROCNAME	ACCOUNT									
GRAPH	GREDUCE	XB.656	21	60	3.65	0.07	32.86	218.96	100.00%	13.91%	.26%
		PROC Total	21	60	3.65	0.07	32.86	218.96	100.00%	13.91%	.26%
	Product Total		303	2617	0.60	0.00	32.86	1573.86	100.00%	100.00%	1.93%
OR	PROCNAME	ACCOUNT									
	CPM	AU.DGP	13	26	0.11	0.09	0.12	2.84	100.00%	100.00%	
		PROC Total	13	26	0.11	0.09	0.12	2.84	100.00%	100.00%	
	Product Total		13	26	0.11	0.09	0.12	2.84	100.00%	100.00%	
SUGI	PROCNAME	ACCOUNT									
	DATACHK	XG.B81	2	3	0.12	0.04	0.17	0.37	9.27%	.14%	
		XG.C12	4	8	0.46	0.07	0.70	3.66	90.72%	1.38%	
		PROC Total	6	11	0.37	0.04	0.70	4.04	100.00%	1.52%	
	FMTLIB	ACCOUNT									
		AM.ZR1	2	2	0.41	0.40	0.41	0.81	17.93%	.30%	
		GA.ALS	1	2	0.42	0.42	0.42	0.84	18.50%	.31%	
		MR.JHI	1	2	0.38	0.38	0.38	0.77	16.93%	.28%	
		XT.P88	1	2	0.73	0.73	0.73	1.46	32.36%	.55%	
		XT.P89	1	2	0.32	0.32	0.32	0.64	14.26%	.24%	
		PROC Total	6	10	0.45	0.32	0.73	4.52	100.00%	1.70%	
	LOGIST	ACCOUNT									
		CG.KXB	6	49	3.60	0.15	4.75	176.61	69.46%	66.73%	.21%
		ET.YCA	10	47	0.14	0.05	0.42	6.49	2.55%	2.45%	
		MA.MJS	36	70	0.14	0.04	0.40	9.79	3.84%	3.69%	.01%

(CONTINUED)

Figure 5: FMTLIB Usage by User for July 1988

YEAR 1988
AND MONTH July

			Occurrence in		CPU Time in seconds			
			Jobs	Steps	MEAN	SUM	% of Product	% of all SAS
ACCOUNT	PACKAGE	PROCNAME						
GA.ALS	BASE	CONTENTS	7	14	0.21	3.00	.90%	.90%
		FORMAT	16	31	0.14	4.35	1.31%	1.31%
		FREQ	4	10	0.10	1.05	.31%	.31%
		MEANS	7	13	0.20	2.55	.77%	.76%
		OPTIONS	6	12	0.04	0.45	.13%	.13%
		PDSCOPY	1	2	0.17	0.33	.10%	.10%
		PRINT	14	32	0.32	10.19	3.07%	3.06%
		PRINTTO	1	2	0.05	0.10	.03%	.03%
		SASDATA	42	126	1.95	246.02	74.27%	74.09%
		SORT	10	36	0.89	32.03	9.66%	9.64%
		TABULATE	9	260	0.12	30.26	9.13%	9.11%
		UNIVARIA	1	2	0.27	0.54	.16%	.16%
		Product Total	119	542	0.61	331.21	100.00%	99.74%
	SUGI	PROCNAME						
		FMTLIB	1	2	0.42	0.84	100.00%	.25%
		Product Total	1	2	0.42	0.84	100.00%	.25%
	ALL		120	544	0.61	332.04	100.00%	100.00%
GA.GRL	PACKAGE	PROCNAME						
	BASE	FORMAT	2	4	0.09	0.37	1.95%	1.95%
		FREQ	3	6	0.31	1.87	9.78%	9.78%
		MEANS	4	8	0.12	0.93	4.87%	4.87%

(CONTINUED)

Figure 6: Usage for User GA.ALS July 1988

YEAR 1988
AND MONTH July

		Occurrence in		CPU Time in seconds					
		Jobs	Steps	MEAN	MIN	MAX	SUM	% of Product	% of all SAS
PACKAGE	Product Total								
BASE		20619	134303	0.59	0.00	561.66	79181.8	100.00%	97.19%
ETS	PROCNAME								
	ARIMA	47	729	0.46	0.10	6.21	335.28	95.25%	.41%
	COMPUTAB	37	81	0.20	0.00	0.41	16.02	4.55%	.01%
	MORTGAGE	4	8	0.08	0.08	0.08	0.67	.19%	
	Product Total	88	818	0.43	0.00	6.21	351.97	100.00%	.43%
FSP	PROCNAME								
	FSEDIT	1	4	0.23	0.19	0.27	0.92	100.00%	
	Product Total	1	4	0.23	0.19	0.27	0.92	100.00%	
GRAPH	PROCNAME								
	GCHART	5	10	0.30	0.00	0.39	2.98	.18%	
	GPLOT	277	2547	0.53	0.00	21.07	1351.92	85.89%	1.65%
	GREDUCE	21	60	3.65	0.07	32.86	218.96	13.91%	.26%
	Product Total	303	2617	0.60	0.00	32.86	1573.86	100.00%	1.93%
OR	PROCNAME								
	CPM	13	26	0.11	0.09	0.12	2.84	100.00%	
	Product Total	13	26	0.11	0.09	0.12	2.84	100.00%	
SUGI	PROCNAME								
	DATACHK	6	11	0.37	0.04	0.70	4.04	1.52%	
	FMTLIB	6	10	0.45	0.32	0.73	4.52	1.70%	
	LOGIST	71	272	0.93	0.04	4.75	254.25	96.07%	.31%
	PAIRED	10	17	0.09	0.04	0.14	1.58	.59%	

(CONTINUED)

Figure 7: Summary Data for Procedures by Product