K2 Analytics
Building Skills, Building Individuals

Advanced SAS

- Rajesh Jakhotia

1 Oct 2017

Earning is in Learning
- Rajesh Jakhotia

About K2 Analytics

At K2 Analytics, we believe that skill development is very important for the growth of an individual, which in turn leads to the growth of Society & Industry and ultimately the Nation as a whole. For this it is important that access to knowledge and skill development trainings should be made available easily and economically to every individual.

Our Vision: *"To be the preferred partner for training and skill development"*

Our Mission: *"To provide training and skill development training to individuals, make them skilled & industry ready and create a pool of skilled resources readily available for the industry"*

*We have chosen Business Intelligence and Analytics as our focus area. With this endeavour we make this "**Advanced SAS Module**" accessible to all those who wish to learn SAS. We hope it is of help to you. For any feedback / suggestion feel free to write back to us at ar.jakhotia@k2analytics.co.in*

Welcome to Base SAS!!!

K2 Analytics
Building Skills, Building Individuals

Welcome to Advanced SAS

Earning is in Learning
- Rajesh Jakhotia

SAS Functions

Trim, Left, Right
Compress & Compbl
Substr
Cat
Scan & Index
Put & Input
intnx & intck

Earning is in Learning
- Rajesh Jakhotia

Trim & Strip

- TRIM – Removes trailing blanks; if the value is a series of blanks then it returns one blank;
- STRIP – Removes Leading and Trailing Spaces
-

```
data _null_;
A = "     A1     A2      ";
B = "                    ";
C = "                   C";
x = trim(A) || trim(B) || "~" ;
y = "~" || trim(B) || trim(C) || "~" ;

put x;

put y;

run;
```

```
A1     A2 ~
~                    C~
```

```
data _null_;
A = "     A1     A2  ";
B = "                ";
C = "                C";
x = strip(A) || strip(B) || "~" ;
y = "~" || strip(B) || strip(C) || "~" ;

put x;

put y;

run;
```

```
A1     A2~
~C~
```

Left & Right

- LEFT– LEFT returns an argument with leading blanks moved to the end of the value
- RIGHT - RIGHT returns an argument with trailing blanks moved to the start of the value

```
data _null_;
A = "     A1     A2      ";
B = "                     ";
C = "                    C";
lx = "~" || left(A) || left(B) || "~" ;
ly = "~" || left(B) || left(C) || "~" ;

put lx;
put ly;
run;
```

```
~A1     A2                    ~
~                     C       ~
```

```
data _null_;
A = "     A1     A2      ";
B = "                    ";
C = "                   C";
rx = "~" || right(A) || right(B) || "~" ;
ry = "~" || right(B) || right(C) || "~" ;

put rx;
put ry;
run;
```

```
~          A1     A2          ~
~                             C~
```

COMPRESS & COMPBL

- COMPRESS – Removes all blanks; or alternatively removes the specified character from the string variable
- COMPBL – Removes multiple blanks in a character string to single blank

```
data _null_;
A = "~     A ~       B !     C    $";
x = compbl(A);
y = compress(A);
z = compress(A, "~A");
put x;
put y;
put z;
run;
```

```
~ A ~ B ! C $
~A~B!C$
B !     C   $
```

SUBSTR

- SUBSTR – Usage

  SUBSTR(*variable, position*<,*length*>)=*characters-to-replace*

```
data _null_;
     a='KIDNAP';
     substr(a, 1, 3)='CAT';
     put a;
     b=a;
     substr(b, 4)="TY";
     put b;
     c=substr(b, 2, 3);
     put c;
run;
```

⇐ **a** changes to CATNAP

⇐ **b** changes to CATTY

⇐ **c** is assigned the value ATT

CAT function

- CAT – used for Concatenation
- Variants of CAT functions are – CAT, CATX, CATT, CATS

```
data _null_;
   separator='@_';
   x='K2 Analytics  ';
   y='   Finishing School ';
   z='   Private ';
   a='Limited.';
   cat_r=cat( x, y, z, a);
   put cat_r $char.;
   catx_r=catx(separator, x, y, z, a);
   put catx_r $char.;
   catt_r=catt( x, y, z, a);
   put catt_r $char.;
   cats_r=cats( x, y, z, a);
   put cats_r $char.;
run;
```

```
K2 Analytics     Finishing School    Private Limited.
K2 Analytics@_Finishing School@_Private@_Limited.
K2 Analytics   Finishing School   PrivateLimited.
K2 AnalyticsFinishing SchoolPrivateLimited.
```

Scan & Index

- SCAN – Used to parse a string based on some rules
- Usage

  SCAN (char_val, n, 'list of delimeters')

  returns the n^{th} word from the char_val, where a word is defined as anything between two delimiters; If there are fewer than n words than scan function will return blank

- INDEX – Used to search a string in longer string
- Usage

  INDEX (char_val, 'search-string')

  returns the position at which search-string is located in the char_val; If search-string does not exist in char_val then returns 0

```
data _null_;
     a='ABCDEFGHDEF123';
     i=index(a, "DEF");
     e=index(a, "-");
     s=scan(a, 2, "DEF");
     put i;
     put e;
     put s;
run;
```

```
4
0
GH
```

Put & Input

- INPUT – Equivalent to INFORMAT;

  Takes 2 argument; 1st argument is the value and 2nd argument defines the informat of the 1st argument

- PUT – Equivalent to FORMAT;

  Takes 2 argument;

  1st argument is the value and 2nd argument defines the format in which the 1st argument should be outputted.

  The output value of PUT will be Character

Put & Input… contd

```
data put_input;
    input dd mon $ yy;
    concat_col=compress(dd||mon||yy);
    input_col=input(concat_col, date7.);
    put_cal=put(input_col, date9.);
    datalines;
01 JAN 14
01 JAN 15
26 JAN 14
26 JAN 15
;
run;
```

Data type is Date (Numeric)

Data type is Character

dd	mon	yy	concat_col	input_col	put_col
1	JAN	14	1JAN14	19724	01JAN2014
1	JAN	15	1JAN15	20089	01JAN2015
26	JAN	14	26JAN14	19749	26JAN2014
26	JAN	15	26JAN15	20114	26JAN2015

INTCK

- INTCK – Returns the integer difference between two Dates / two Times / two Datetimes
- Usage : intck (interval_measure, Date 1, Date 2)

```
data tmp_intck;
Format date1 date2 date9.;
date1 = '01Jan2016'd;
date2 = '26Jan2017'd;
years=intck('year',date1,date2);
SEMIYEAR=intck('SEMIYEAR',date1,date2);
quarters=intck('qtr',date1,date2);
months=intck('month',date1,date2);
weeks=intck('week',date1,date2);
days=intck('day',date1,date2);
run;
```

	date1	date2	years	SEMIYEAR	quarters	months	weeks	days
1	01JAN2016	26JAN2017	1	2	4	12	56	391

http://www.sascommunity.org/wiki/INTCK_function_examples

INTNX

- INTNX – Returns a date which is specified number of time units away from a specified date
- Usage : intnx(interval_measure, ref_date, time_units, <b | s | m | e>)

```
data tmp_intnx;
format dt beginning_dt middle_dt end_dt sameday_dt date9.;
dt = '15Aug2017'd;
beginning_dt = intnx('month', dt, 1, 'b');
middle_dt = intnx('month', dt, 1, 'm');
end_dt = intnx('month', dt, 1, 'e');
sameday_dt = intnx('month', dt, 1, 's');
run;
```

dt	beginning_dt	middle_dt	end_dt	sameday_dt
15AUG2017	01SEP2017	15SEP2017	30SEP2017	15SEP2017

http://www.sascommunity.org/wiki/INTNX_Function_examples

QUIZ

Earning is in Learning
- Rajesh Jakhotia

Quiz

- Match the following

A) Strip()	1) Concatenation function
B) Cat()	2) Used for Text Mining – to search a string in another string
C) Put()	3) Strips leading and trailing white spaces
D) Index()	4) Put the input value in desired format and output is Character
E) Intnx()	5) Used to increment / decrement date by certain specified number of units
F) Intck()	6) By default suppresses all blanks in character string
G) Compress()	7) Used to find difference in two dates

SAS PROCs

Proc Format
Proc Contents
Proc Means & Proc Univariate
Proc Freq
Proc Sort – NODUP & NODUPKEY
Proc Rank
Proc Corr
Proc Transpose
Proc SQL
Proc Delete

Earning is in Learning
- Rajesh Jakhotia

Proc Format

- Proc Format is used to display values in certain Format
- Syntax

```
Proc Format;
Value User_Defined_Format_Name
        Range_1 = “Label_1”
        Range_2 = “Label_2”
        ……
        ……;
Run;
```

Range can be single value, range of values, list of values

PROC Format …contd

```
Proc format;
value age_buckets
low - 21 = "<=21"
21 < - 31 = "21-31"
31 < - 41 = "31-41"
41 < - high = ">41"
;

data LR1;
set LR1;
format Age_FMT_BKT age_buckets.;
Age_FMT_BKT = Age;
run;

data LR1;
set LR1;
format Age_FMT_BKT int.;
run;
```

Age_FMT_BKT
21-31
>41
>41
>41
31-41
31-41

Age_FMT_BKT
30
43
53
45
37
41

```
data LR1;
set LR1;
Age_FMT_BKT_CHAR = put(Age, age_buckets.);
run;
```

Age_FMT_BKT_CHAR
21-31
>41
>41
>41
31-41
31-41

Proc Contents

- Proc Contents help get the metadata of the dataset
- Syntax

```
proc contents data=<dataset name>
out=out dataset name
noprint;
run;
```

Use NOPRINT if you do not wish to have the output printed in Results Tab

```
proc contents data=PUT_INPUT;
run;
```

Alphabetic List of Variables and Attributes

Number	Variable	Type	Len	Pos
4	concat_col	Char	32	32
1	dd	Num	8	0
5	input_col	Num	8	16
2	mon	Char	8	24
6	put_col	Char	9	64
3	yy	Num	8	8

Proc Means

```
proc means data=<dataset name>;
run;
```

Most simple form…it will return N (No. of Records), Mean, Min, Max and Std. Dev for all numeric variables in the dataset

---

```
proc means data=<dataset name>;
    var var_1 var_2 ...;
run;
```

If we desire the statistics for only specific variables

```
proc means data=<dataset name>;
    var var_name;
    output out=<out_dst_name> n=n min=min max=max mean=mean p1=p1 p5=p5 p10=p10 p10=p10
        median=median p75=p75 p90=p90 p95=p95 p99=p99;
run;
```

To get more detailed statistics

```
proc means data=<dataset name>;
    var var_name;
    class cls_var_1 cla_var_2....;
    output out=<out_dst_name> n=n min=min max=max mean=mean p1=p1 p5=p5 p10=p10 p10=p10
        median=median p75=p75 p90=p90 p95=p95 p99=p99;
run;
```

To get statistics w.r.t each segment / class

Proc Means… contd

```
proc means data=LR1;
run;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Target	20000	0.0444000	0.2059873	0	1.0000000
Age	20000	38.3962000	9.6001788	21.0000000	55.0000000
Balance	20000	146181.31	169812.53	0	1246966.77
No_OF_CR_TXNS	20000	16.6179500	12.9699498	0	50.0000000

Proc Univariate

```
proc univariate data=<dataset name>;
run;
```

Provides detailed statistics for each numeric variable in the dataset

```
proc univariate data=<dataset name>;
    var var_1 var_2 ...;
run;
```

If we desire the statistics for only specific variables

```
proc univariate data=<dataset name>;
    var var_1;
    output out=csv_uni n=n nmiss=nmiss mean=mean min=min p1=p1 p5=p5 median=median
        p90=p90 p95=p95 p99=p99 max=max range=range stderr=stderr var=var;
run;
```

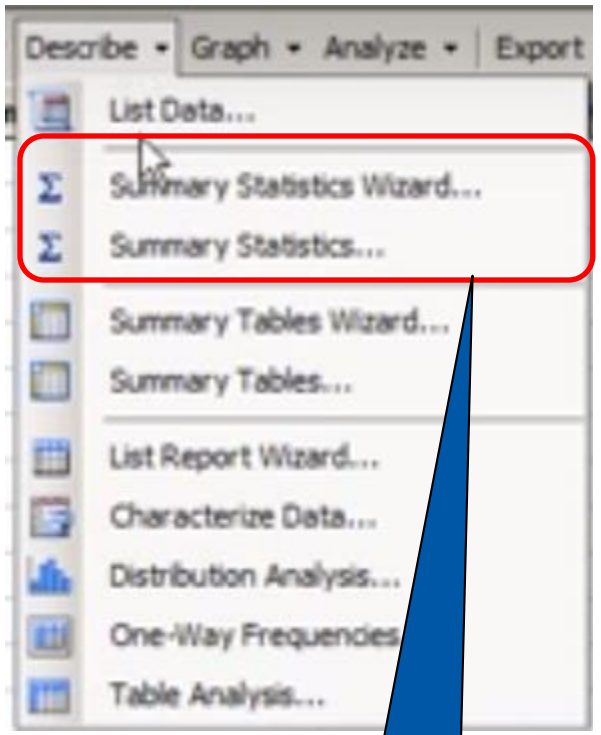
To get the output written to some dataset

Proc Univariate… contd

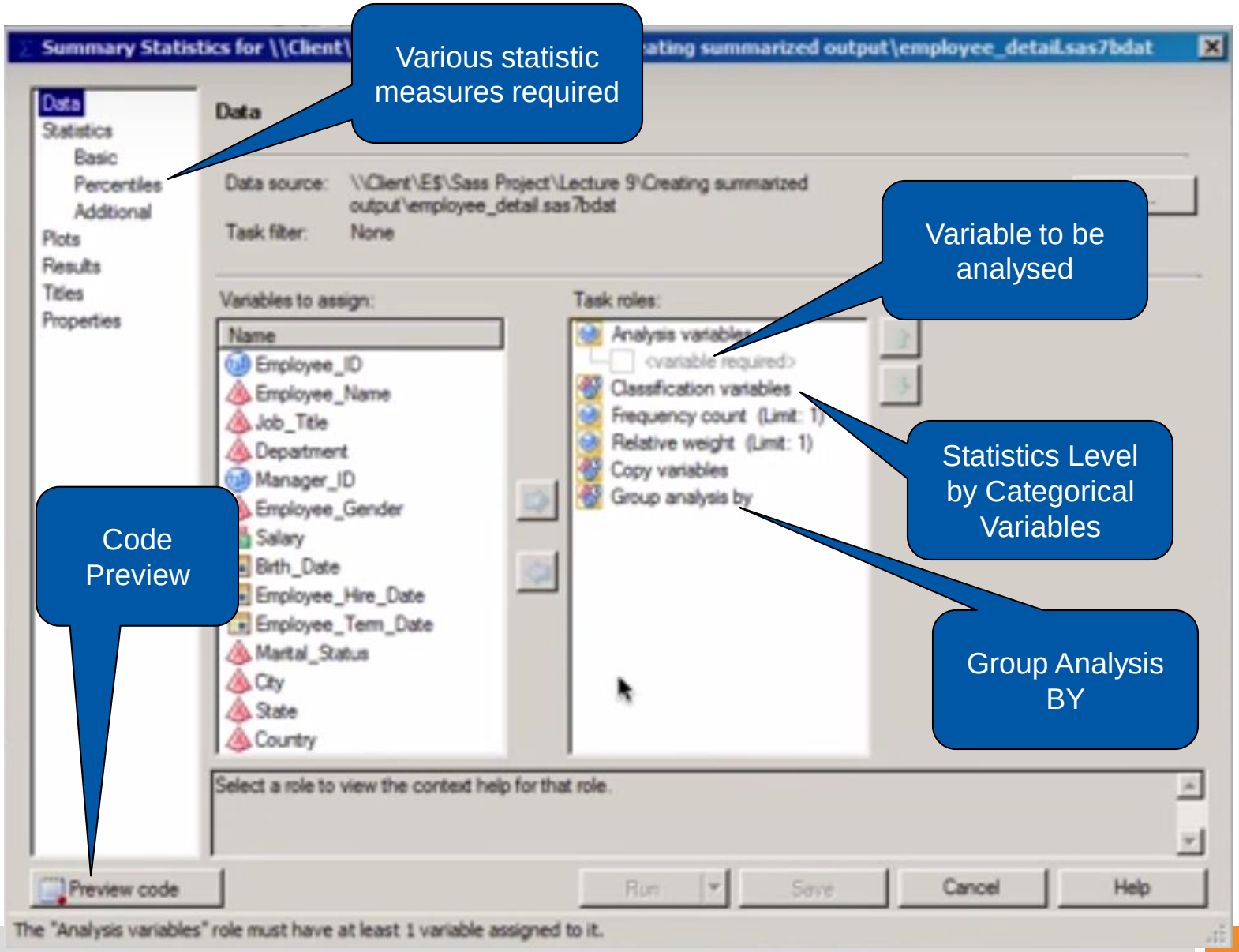
```
proc univariate data=LR1;
    var Balance;
    output out=uni_out n=n min=min max=max mean=mean p1=p1 p5=p5 p10=p10
    median=median p75=p75 p90=p90 p95=p95 p99=p99;
run;
```

n	mean	max	p99	p95	p90	median	p10	p5	p1	min
20000	146181.30563	1246966.77	726693.58	517448.325	392425.115	79755.745	7249.835	3821.755	572.175	0

Summary Statistics using SAS EG



Select Summary Statistics



Various statistic measures required

Variable to be analysed

Statistics Level by Categorical Variables

Group Analysis BY

Code Preview

Proc Freq

```
proc freq data=<dataset name>;
run;
```

Do not run this type of code… it will run the frequency distribution for all columns (Numeric or Character)

```
proc freq data=<dataset name>;
    table var_1 var_2 ... ;
run;
```

Generates one-way frequency table for each variables passed as parameter in TABLE option

```
proc freq data=<dataset name>;
    table var_1*var_2 ;
run;
```

Generates Cross-Tab between Var_1 and Var_2

```
proc freq data=<dataset name>;
    table var_1*var_2 / norow nocol nopercent chisq;
run;
```

Generates Cross-Tab with Chi-Sq statistics; The row percent, column percent and overall percentage will not be displayed

Proc Freq… contd

```
proc freq data=LR1;
    table Gender Age_bkt;
run;
```

The FREQ Procedure

Gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	5525	27.63	5525	27.63
M	14279	71.40	19804	99.02
O	196	0.98	20000	100.00

AGE_BKT	Frequency	Percent	Cumulative Frequency	Cumulative Percent
26-30	3404	17.02	3404	17.02
31-35	3488	17.44	6892	34.46
36-40	2756	13.78	9648	48.24
41-45	3016	15.08	12664	63.32
46-50	2532	12.66	15196	75.98
<25	1784	8.92	16980	84.90
>50	3020	15.10	20000	100.00

Proc Sort with NoDUP / NoDUPKEY

- NODUP – No Duplicate

```
proc sort data= <dataset name>
nodup
out=<out_dataset name>;
by <var>;
run;
```

- NODUPKEY – No Duplicate Key

```
proc sort data= <dataset name>
nodupkey
out=<out_dataset name>;
by <var>;
run;
```

Sample Dataset to Test NODUP & NODUPKEY

```
data student;
    input roll_no 1-4 subject $6-14 mark 16-18 dt;
    informat dt ddmmyy10.;
    format dt date9.;
    datalines;
1    Maths     95  01-12-2013
1    Physics   91  01-12-2013
1    Chemistry 100 01-12-2013
2    Maths     55  01-12-2013
2    Physics   61  01-12-2013
2    Chemistry 73  01-12-2013
2    Chemistry 73  01-12-2013
1    Maths     95  01-12-2013
;
run;
```

Proc Rank

```
proc rank data=<dataset name>;
    var <var name>;
    ranks <rank_col_name>;
run;
```

Variable mentioned in **var** will be ranked
The ranked value will be stored in column name mentioned in **ranks** option

In Rank Proc, if there is a **tie** between values then by default the **MEAN** rank is returned for the tied values

```
proc rank data=<dataset name>
    out=<out_dst>
    ties=low groups=<no_of_groups>;
    var <var name>;
    ranks <rank_col_name>;
run;
```

Using GROUPS Option to create Deciles or desired number of groups

Proc Rank… contd

```
proc rank data=LR1 out=LR1_RANK;
    var Age;
    ranks RAnk_Age;
run;
```

```
proc rank data=LR_DF out=LR_DF_DECILE groups=10;
    var Age;
    ranks decile_age;
run;
```

Proc Corr

```
proc corr data=<dataset name>;
run;
```

Provides correlation analysis between all numeric variables

```
proc corr data=<dataset name>;
    var var_1 var_2 ...;
run;
```

Gives correlation only for the variables passed in var option

```
proc corr data=<dataset name>;
    by by_var_1 by_var_2 ...;
    var var_1 var_2 ...;
run;
```

Gives correlation w.r.t BY Variables;
Note: Data should be sorted by BY Variables

Proc Corr… contd

```
proc corr data=LR_DF;
    var Age No_OF_CR_TXNS Balance;
run;
```

Pearson Correlation Coefficients, N = 20000 Prob > |r| under H0: Rho=0			
	Age	No_OF_CR_TXNS	Balance
Age	1.00000	0.05777	-0.14692
	1.0000	<.0001	<.0001
No_OF_CR_TXNS	0.05777	1.00000	-0.13414
	<.0001	1.0000	<.0001
Balance	-0.14692	-0.13414	1.00000
	<.0001	<.0001	1.0000

Proc Transpose

- Transpose – Convert Row to Columns or Columns to Rows

```
data LR_DF;
    set LR_DF;
    flag=1;
run;

proc sort data=LR_DF;
    by Cust_ID;
run;

proc transpose data=lr_df
    out=trans_out;
    var flag;
    id occupation;
    by Cust_ID;
run;
```

One of the application can be to create dummy variables

Cust_ID	_NAME_	SAL	PROF	SELF_EMP	SENP
C1	Flag	1	.	.	.
C10	Flag	.	1	.	.
C100	Flag	.	1	.	.
C1000	Flag	1	.	.	.
C10000	Flag	.	.	.	.
C10001	Flag	1	.	.	.
C10002	Flag	.	1	.	.
C10003	Flag	1	.	.	.
C10004	Flag	.	1	.	.
C10005	Flag	.	1	.	.
C10006	Flag	1	.	.	.
C10007	Flag	.	1	.	.
C10008	Flag	.	1	.	.
C10009	Flag	.	1	.	.
C1001	Flag	.	1	.	.

Proc Transpose

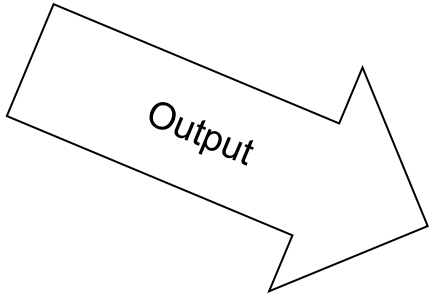
Quick Reference Sheet – Proc Transpose

- Proc Transpose changes multiple *values* in rows (for a column) into *columns*, and can also change multiple *columns'* values into multiple *rows* values for a single column
- It knows how many columns to create for your output file based on the maximum number of values in a column to be transposed (to do this with a Data Step is tedious)
- **ID** statement names the column in the input file whose row values provide the column names in the output file. There should only be one variable in an ID statement. Also, the column used for the ID statement cannot have any duplicate values. For example, Mr. Black and Mr. White cannot both have cats. If this is the case, one solution is to create a different input dataset by using Proc Means to sum the values so that there is only one row for each pet. What if the values of ID are numeric and can't be used as SAS column names? Then use the prefix= option with the ID statement to create variables like "mile140", "mile150", etc.
- **VAR** statement specifies which variables' values are to be transposed; can be character and/or numeric variables; if VAR is omitted, Transpose transposes all numeric vars
- **BY** statement names row-identification variable(s) whose values are not transposed; it requires a preliminary Proc Sort
- Transpose includes some default variables in the output dataset such as **_NAME_**, **_LABEL_**. You can override them with statement options or drop them in a dataset drop option
- **Prefix** option provides a prefix to the transposed column names instead of COL1, COL2, etc
- **Name** option provides the name for an output file column which tells which input variables were transposed
- **Transposing two times –** it is sometimes necessary to transpose an input file two or more times, then merge the output files together due to the only-1-ID-per-transpose limitation.

Proc SQL

- In SAS you can execute standard SQL queries by writing it between PROC SQL – QUIT block

```
proc sql ;
    select Age, count(1) as cnt, sum(Balance) as sum_bal
    from LR_DF
    where Gender='M'
    Group by Age
    Having cnt>500
    order by Age desc;
quit;
```



Age	Cnt	sum_bal
53	506	47610108
32	573	93159897
31	525	73347281
30	525	81047765
28	524	73866552

Proc Delete

```
proc delete data= <dataset_name>;
run;
```

Deletes the dataset

```
proc delete data=student;
run;
```

First.

Last.

Lag

Earning is in Learning
- Rajesh Jakhotia

.First and .Last

Processing Observations in a BY Group

In the DATA step, SAS identifies the beginning and end of each BY group by creating two temporary variables for each BY variable: FIRST.*variable* and LAST.*variable*. These temporary variables are available for DATA step programming but are not added to the output data set. Their values indicate whether an observation is one of the following positions:

- the first one in a BY group
- the last one in a BY group
- neither the first nor the last one in a BY group
- both first and last, as is the case when there is only one observation in a BY group

You can take actions conditionally, based on whether you are processing the first or the last observation in a BY group.

http://support.sas.com/documentation/cdl/en/lrcon/69852/HTML/default/viewer.htm#n01a08zkzy5igbn173zjz82zsi1s.htm

.First and .Last …contd

```
proc import datafile= "/folders/myfolders/raw_data/SA_Transactions.csv"
        out=SA_TXNS
        dbms=dlm
        replace;
    delimiter=',';
    getnames=yes;
run;
```

```
data SA_TXNS;
set SA_TXNS;
MTH_ID = year(Date) * 100 + month(Date);
run;
```

- Flag the First Transaction of Account for every Month combination as FLG_FIRST = 1
- Flag the Last Transaction of Account as FLG_LAST = 1

Txn_ID	Account_No	Date	Narration	Dr__Amount	Cr__Amount	Chq_Ref_Number	Closing_Balance	MTH_ID
1	10851232494	01APR2011	Opening Balance	.	.		10000	201104
2	10851232494	01APR2011	Salary	.	100000	3455	110000	201104
3	10851232494	02APR2011	NEFT-INDIA INFOLINE	.	10500	100000	120500	201104
4	10851232494	04APR2011	NEFT TRF TO OTHER BANK	100000	60000	958	80500	201104
5	10851232494	11APR2011	LOAN EMI Payment	35000	.	1316	45500	201104
6	10851232494	16APR2011	ECS CLG RELIANCE	.	5861.39	200371	51361.39	201104

.First and .Last …contd

```
data SA_TXNS;
set SA_TXNS;
by Account_No MTH_ID;
FLG_FIRST = First.Account_No OR First.MTH_ID;
/*
    if (First.Account_No OR First.MTH_ID) then FLG_FIRST = 1
    else FLG_FIRST = 0;
*/

FLG_LAST =  Last.Account_No;
/*
    if Last.Account_No then FLG_LAST = 1
    else FLG_LAST = 0;
*/
run;
```

Txn_ID	Account_No	Date	Narration	Dr__Amount	Cr__Amount	Chq_Ref_Number	Closing_Balance	MTH_ID	FLG_FIRST	FLG_LAST
1	10851232494	01APR2011	Opening Balance	.	.		10000	201104	1	0
2	10851232494	01APR2011	Salary	.	100000	3455	110000	201104	0	0
3	10851232494	02APR2011	NEFT-INDIA INFOLINE	.	10500	100000	120500	201104	0	0
4	10851232494	04APR2011	NEFT TRF TO OTHER BANK	100000	60000	958	80500	201104	0	0
5	10851232494	11APR2011	LOAN EMI Payment	35000	.	1316	45500	201104	0	0
6	10851232494	16APR2011	ECS CLG RELIANCE	.	5861.39	200371	51361.39	201104	0	0
7	10851232494	18APR2011	DD Issue-On HDFC-KOLKATTA - 108513000553	7500	.	975950	43861.39	201104	0	0
8	10851232494	01MAY2011	Reimbursements	.	5607	18011	49468.39	201105	1	0
9	10851232494	01MAY2011	Salary	.	98900	959	148368.39	201105	0	0
10	10851232494	04MAY2011	NEFT TRF TO OTHER BANK	60000	.	1601	88368.39	201105	0	0
11	10851232494	11MAY2011	LOAN EMI Payment	35000	.	0758A1	53368.39	201105	0	0
12	10851232494	15MAY2011	CHQ DEPOSIT - Mangalam Cements	.	1000	1662A1	54368.39	201105	0	0
13	10851232494	16MAY2011	LIC INSURANCE PREMIUM	22343	.	160928	32025.39	201105	0	0
14	10851232494	08JUN2011	ATM CHQ Deposit	.	20000	180112	52025.39	201106	1	0
15	10851232494	11JUN2011	LOAN EMI Payment	35000	.	1001	17025.39	201106	0	0
16	10851232494	23JUN2011	ECS CLG NTPC	.	1000	49594	18025.39	201106	0	0
17	10851232494	09JUL2011	NEFT CR	.	20000	160930	38025.39	201107	1	0
18	10851232494	11JUL2011	LOAN EMI Payment	35000	.	11234	3025.39	201107	0	0
19	10851232494	10AUG2011	ATM CHQ Deposit	.	35000	160929	38025.39	201108	1	0
20	10851232494	12AUG2011	LOAN EMI Payment	35000	.		3025.39	201108	0	0
21	10851232494	20AUG2011	ATM CASH WDL	2000	.	1001	1025.39	201108	0	0
22	10851232494	01SEP2011	ATM CASH WDL	1000	.	1002	25.39	201109	1	0
23	10851232494	09SEP2011	CHQ DEPOSIT	.	35000	180123	35025.39	201109	0	0
24	10851232494	11SEP2011	LOAN EMI Payment	35000	.	160931	25.39	201109	0	0
25	10851232494	01OCT2011	AQB NON MAINTENANCE CHARGES	750	.		-724.61	201110	1	0
26	10851232494	01OCT2011	Service Charge & Edu Cess	76.5	.		-801.11	201110	0	1

Lag()

```
proc sort data = SA_TXNS; by Account_No Date; run;

data SA_TXNS;
set SA_TXNS;
by Account_No;
FORMAT Prev_Date Date9.;
Prev_Balance = lag(Closing_Balance);
Prev_Date = lag(Date);
Date_Diff = Date - Prev_Date;
run;
```

Obs	Txn_ID	Account_No	Date	Narration	Dr__Amount	Cr__Amount	Chq_Ref_Number	Closing_Balance	MTH_ID	Prev_Date	Prev_Balance	Date_Diff
1	1	10851232494	01APR2011	Opening Balance	.	.		10000	201104	.	.	.
2	2	10851232494	01APR2011	Salary	.	100000	3455	110000	201104	01APR2011	10000.00	0
3	3	10851232494	02APR2011	NEFT-INDIA INFOLINE	.	10500	100000	120500	201104	01APR2011	110000.00	1
4	4	10851232494	04APR2011	NEFT TRF TO OTHER BANK	100000	60000	958	80500	201104	02APR2011	120500.00	2
5	5	10851232494	11APR2011	LOAN EMI Payment	35000	.	1316	45500	201104	04APR2011	80500.00	7
6	6	10851232494	16APR2011	ECS CLG RELIANCE	.	5861.39	200371	51361.39	201104	11APR2011	45500.00	5
7	7	10851232494	18APR2011	DD Issue-On HDFC-KOLKATTA - 108513000553	7500	.	975950	43861.39	201104	16APR2011	51361.39	2
8	8	10851232494	01MAY2011	Reimbursements	.	5607	18011	49468.39	201105	18APR2011	43861.39	13
9	9	10851232494	01MAY2011	Salary	.	98900	959	148368.39	201105	01MAY2011	49468.39	0

SAS ARRAY

Earning is in Learning
- Rajesh Jakhotia

What is an ARRAY in SAS?

- SAS arrays are another way to temporarily group and refer to SAS variables
- A SAS array is not a new data structure, the array name is not a variable, and arrays do not define additional variables
- Rather, a SAS array provides a different name to reference a group of variables
- The ARRAY statement defines variables to be processed as a group
- The variables referenced by the array are called elements
- Once an array is defined, the array name and an index reference the elements of the array
- **Note: Arrays within SAS are different than arrays in other languages**

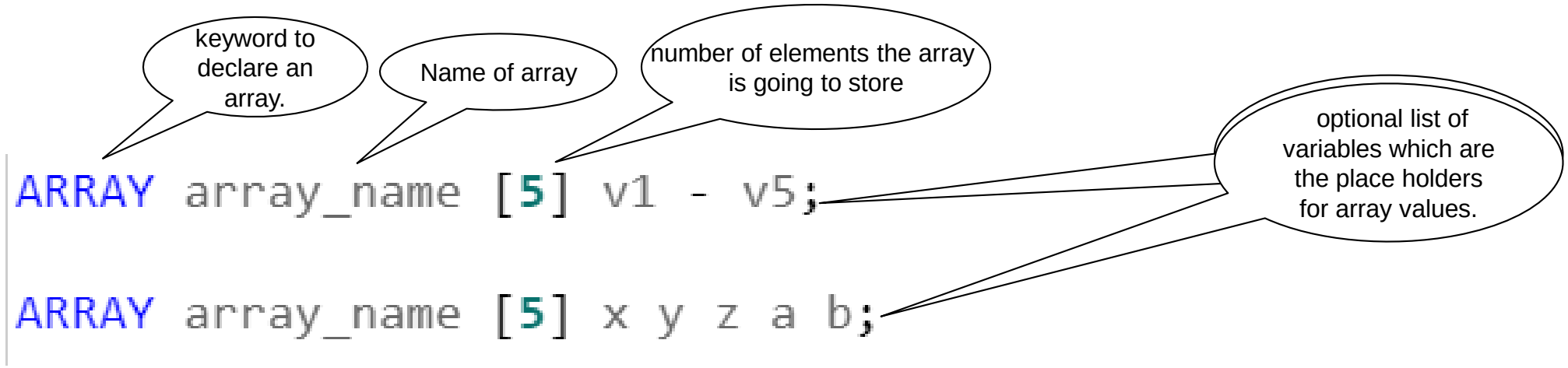
http://www2.sas.com/proceedings/sugi30/242-30.pdf

Array… contd

Why use Array?

- Arrays help simplify code
- Useful in analysing repetitive data with minimum coding
- A convenient way of temporarily identifying a group of variables for processing within a data step

Sample Array Declaration



Array Example 1

```
data ARRAY_EXAMPLE;
Input A1 - A4 X Y Z;
Array A [4] A1 - A4;
Array XYZ[3] X Y Z;

A_SUM = sum ( of A(*));
A_SUM_1 = sum ( of A1 - A4);

XYZ_SUM = sum ( of XYZ(*));

/* The below statement will not run */
*XYZ_SUM_1 = sum ( of X - Z);

Datalines;
1 2 3 4 10 20 30
11 22 33 44 100 200 300
;
run;
Proc print data = ARRAY_EXAMPLE;
run;
```

Note the usage of “OF” option in SUM Function

Obs	A1	A2	A3	A4	X	Y	Z	A_SUM	A_SUM_1	XYZ_SUM
1	1	2	3	4	10	20	30	10	10	60
2	11	22	33	44	100	200	300	110	110	600

Array Example 2

- Convert all Missing Values to 0 (Zero)

```
data SA_TXNS_1;
set SA_TXNS;
array change _numeric_;
do over change;
    if change=. then change=0;
end;
run ;
```

Txn_ID	Account_No	Date	Narration	Dr__Amount	Cr__Amount	Chq_Ref_Number	Closing_Balance
1	10851232494	01APR2011	Opening Balance	0	0		10000
2	10851232494	01APR2011	Salary	0	100000	3455	110000
3	10851232494	02APR2011	NEFT-INDIA INFOLINE	0	10500	100000	120500
4	10851232494	04APR2011	NEFT TRF TO OTHER BANK	100000	60000	958	80500
5	10851232494	11APR2011	LOAN EMI Payment	35000	0	1316	45500
6	10851232494	16APR2011	ECS CLG RELIANCE	0	5861.39	200371	51361.39
7	10851232494	18APR2011	DD Issue-On HDFC-KOLKATTA - 108513000553	7500	0	975950	43861.39
8	10851232494	01MAY2011	Reimbursements	0	5607	18011	49468.39
9	10851232494	01MAY2011	Salary	0	98900	959	148368.39
10	10851232494	04MAY2011	NEFT TRF TO OTHER BANK	60000	0	1601	88368.39
11	10851232494	11MAY2011	LOAN EMI Payment	35000	0	0758A1	53368.39
12	10851232494	15MAY2011	CHQ DEPOSIT - Mangalam Cements	0	1000	1662A1	54368.39

Practical Application involving

INTCK Function
PROC SQL – for Aggregation
PROC TRANSPOSE – for Transposition
SUM (OF) Concept for Row Wise Aggregation
DATA MERGE

Earning is in Learning
- Rajesh Jakhotia

Business Problem Statement

- You have been given Retail Banking Transaction Data. You have to Aggregate the data at ACCOUNT level to find out "**Number of Credit Transactions**" and "**Total Amount Credited**" at Account Level and Month-on-Month, Quarter and Ratio of the two Quarters.
- The output has to be one row per account format
- **Best Practice**: Aggregated Variables should have self-explanatory names

- **Steps:**
  - *Aggregate Data at Month Level*
  - *Transpose by Account Number*
  - *Sum the Monthly Variables to create Quarterly Variables*
  - *Create the Ratio Variable*
  - *Data Merge*

Step 1: Aggregation

```
data SA_TXNS;
set SA_TXNS;
REF_MTH_ID = intck('month',"01Jan2011"d,date);
run;
```

```
proc sql;
create table ACC_SUMMARY as
select Account_No, REF_MTH_ID,
count(cr__Amount) as CNT_CR_TXNS,
sum(cr__Amount) as AMT_CR_TXNS
from SA_TXNS
group by Account_No, REF_MTH_ID;
quit;
```

Obs	Account_No	REF_MTH_ID	CNT_CR_TXNS	AMT_CR_TXNS
1	10851232494	3	4	176361.39
2	10851232494	4	3	105507.00
3	10851232494	5	2	21000.00
4	10851232494	6	1	20000.00
5	10851232494	7	1	35000.00
6	10851232494	8	1	35000.00
7	10851232494	9	0	.

Step 2: Transpose and sum (of)

```
proc transpose data=ACC_SUMMARY out = ACC_SUMMARY_T name= var  ;
by Account_No;
ID  REF_MTH_ID;
var CNT_CR_TXNS AMT_CR_TXNS;
run;

data ACC_SUMMARY_T;
set ACC_SUMMARY_T;
RQ1 = sum(of _3 - _5);
RQ2 = sum(of _6 - _8);
RQ2_by_RQ1 = round(RQ2 / RQ1,0.001);
run;
```

Obs	Account_No	var	_3	_4	_5	_6	_7	_8	_9	RQ1	RQ2	RQ2_by_RQ1
1	10851232494	CNT_CR_TXNS	4.00	3	2	1	1	1	0	9.00	3	0.333
2	10851232494	AMT_CR_TXNS	176361.39	105507	21000	20000	35000	35000	.	302868.39	90000	0.297

Step 3: Data Merge

```
data ACC_SUMMARY_F (drop = var _9);
merge ACC_SUMMARY_T ( where = (var = 'CNT_CR_TXNS')
    rename= (   _3 = CNT_CR_TXNS_3   _4 = CNT_CR_TXNS_4   _5 = CNT_CR_TXNS_5
                _6 = CNT_CR_TXNS_6   _7 = CNT_CR_TXNS_7   _8 = CNT_CR_TXNS_8
                RQ1 = CNT_CR_TXNS_RQ1  RQ2 = CNT_CR_TXNS_RQ2   RQ2_by_RQ1 = CNT_CR_TXNS_RQ2_by_RQ1
                )
        )


    ACC_SUMMARY_T ( where = (var = 'AMT_CR_TXNS')
    rename= (   _3 = AMT_CR_TXNS_3   _4 = AMT_CR_TXNS_4   _5 = AMT_CR_TXNS_5
                _6 = AMT_CR_TXNS_6   _7 = AMT_CR_TXNS_7   _8 = AMT_CR_TXNS_8
                RQ1 = AMT_CR_TXNS_RQ1  RQ2 = AMT_CR_TXNS_RQ2   RQ2_by_RQ1 = AMT_CR_TXNS_RQ2_by_RQ1
                )
        );
by  Account_No;
run;
```

Account_No	CNT_CR_TXNS_3	CNT_CR_TXNS_4	CNT_CR_TXNS_5	CNT_CR_TXNS_6	CNT_CR_TXNS_7	CNT_CR_TXNS_8	CNT_CR_TXNS_RQ1	CNT_CR_TXNS_RQ2	CNT_CR_TXNS_RQ2_by_RQ1
10851232494	4	3	2	1	1	1	9	3	0.333

AMT_CR_TXNS_3	AMT_CR_TXNS_4	AMT_CR_TXNS_5	AMT_CR_TXNS_6	AMT_CR_TXNS_7	AMT_CR_TXNS_8	AMT_CR_TXNS_RQ1	AMT_CR_TXNS_RQ2	AMT_CR_TXNS_RQ2_by_RQ1
176361.39	105507	21000	20000	35000	35000	302868.39	90000	0.297

K2 Analytics
Building Skills, Building Individuals

Thank you

Name: Rajesh Jakhotia
Email : ar.jakhotia@k2analytics.co.in
Mobile: 89396 94874

Earning is in Learning
- Rajesh Jakhotia