OVERVIEW of BASIC PROCS [1]

General Remarks about SAS PROCs

The purpose in life of the DATA step is to *create* a SAS ds (usually).
The purpose in life of the PROC step is to *analyze* a SAS ds (usually).
DATA step processes data worksheet by rows (obs).
PROC step analyzes data worksheet by columns (vars).

Output of PROCs is placed in OUTPUT window.

Features in common to most analytical PROCs:

- You can say which SAS ds you want to analyze (**DATA=**)
- You can say if you want it analyzed in subsets (**BY**)
- You can say which variables you want analyzed (**VAR**)

Defaults for these features:

- Analyze most recently created SAS ds
- Do not analyze it in subsets
- Analyze all the (numeric) variables

Additional features are available for most PROCs.

[1] The Department of Statistics and Data Science has an excellent tutorial on this material at **https://stat.utexas.edu/images/SSC/documents/SoftwareTutorials/SAS_InferentialStats.pdf**.

IMPORTANT FEATURES of SPECIFIC PROCS

PROC SORT

Function:

- To arrange a SAS dataset in numerical order
- To arrange a SAS dataset in alphabetical order
- To sort SAS ds into subsets so a later proc can analyze it in subsets

Syntax:

```
PROC SORT DATA=<sas ds> OUT=<sas ds>;
  BY <var(s)>;
```

Notes:

Sorts from least to greatest (by default)

Missing values precede all other values in sort order.

Use **DESCENDING** to reverse the sort order

Ex: **BY DESCENDING SCORE;**

Use more than one **BY** var for nested sort

Ex: **BY SEX NAME;**

This sorts alphabetically by name within each sex.

In Windows, the sorted SAS ds will replace and destroy the unsorted version unless the optional **OUT=** feature is used. If **OUT=** is used, then both the sorted and unsorted ds are preserved.

A system option that may affect results of PROC SORT: **OPTIONS REPLACE;** [2]

- Purpose is to force replacement of existing SAS ds by new SAS ds when new SAS dataset has same name as old SAS dataset
- Old SAS ds is replaced if **OPTIONS REPLACE** (the default in Windows) is in effect (old version of SAS ds is destroyed)
- Old SAS ds remains if **OPTIONS NOREPLACE** is in effect (new version of SAS ds is destroyed)

BY is required.

BY variables must be in the SAS ds being sorted.

SORT puts nothing in the OUTPUT window.

---

[2] The **REPLACE** / **NOREPLACE** option is one of many SAS System options that can be set by the local systems administrator who installs SAS. This system option can be overridden by the individual user through the **OPTIONS** statement. If **REPLACE** is in effect (either by default through the system's administrator's choice, or through the individual user's use of **OPTIONS REPLACE**), then any newly created SAS dataset with the same name as an existing SAS dataset will overwrite that existing SAS dataset. If **NOREPLACE** is in effect, then an unexpected consequence ensues: A newly created SAS dataset with the same complete name (including the path) as an existing SAS dataset will be destroyed immediately -- without triggering an error message! The reason is that **NOREPLACE** prevents the replacement of the existing SAS dataset with the eponymous new SAS dataset; SAS has no place to put the new dataset because its name is already in use; so SAS destroys the new dataset. In some operating systems, like VMS, however, a newly created SAS dataset is always given a fresh version number, so there is never a new SAS dataset with exactly the same name as an existing dataset, so the potential trap just outlined cannot happen. However, in SAS for Windows on a PC, there are no version numbers (unless **MAXGEN** option has been activated), so the trap could arise. Probably, **REPLACE** is the default in your SAS for Windows installation. Run **PROC OPTIONS;** to get a list of the default settings for your installation.

Examples: [The examples given here run consecutively as part of the same program.] (Assume OPTIONS REPLACE is in effect on a Windows system.)

```
DATA;  * Creates WORK.DATA1;
  INPUT NAME $ SEX $ AGE;
CARDS;
BILL M 32
MARY F 48
...
```

Ex.1: ** Alphabetize in name order;
** (WORK.DATA1 is re-ordered, losing its original order);

```
PROC SORT;
  BY NAME;
```

Ex.2: ** Order by age, oldest first, name the sorted version WORK.RECORDS;
** (The version previously sorted by NAME is preserved in WORK.DATA1 – the version newly sorted by descending AGE is WORK.RECORDS);

```
PROC SORT OUT=RECORDS;
  BY DESCENDING AGE;
```

Ex.3: ** Order females first, then males, by decreasing age within each sex;
** (WORK.RECORDS is re-ordered, losing its previous order);

```
PROC SORT DATA=RECORDS;
  BY SEX DESCENDING AGE;
```

Ex.4: ** Order males first, then females, by increasing age within each sex, call sorted version EMPLOYEE.RECORDS (permanent SAS dataset);
** (WORK.RECORDS retains its previous order -- sorted version is EMPLOYEE.RECORDS);

```
LIBNAME EMPLOYEE 'C:\MYFOLDER';
PROC SORT DATA=RECORDS OUT=EMPLOYEE.RECORDS;
  BY DESCENDING SEX AGE;
```

PROC PRINT

Function: to produce English-readable listing of data in SAS ds

Syntax:

```
PROC PRINT DATA=<sas ds>;
  BY <vars>;
  VAR <vars>;
```

Notes:

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

If use **VAR**, only those variables named are printed, and they are printed in the order given in **<vars>**.

PRINT puts a listing of the data into the OUTPUT window. No hard copy is produced immediately.

To get hard copy, print the OUTPUT window (click print icon). Or copy the window into a text editor and print.

Example:

** Print only name and age of people in WORK.RECORDS, first females, then males (assuming WORK.RECORDS is already sorted by sex);

```
PROC PRINT DATA=RECORDS;
  VAR NAME AGE;
  BY SEX;
```

PROC PLOT [3]

Function: produces low resolution scatterplots

Syntax:

```
PROC PLOT DATA=<sas ds>;
  PLOT <vert> * <horiz> = { '< char >' | < var > }  <vert> * <horiz> ... / options>;
  BY <vars
```

Notes:

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

PLOT statement is required.

Default plotting char is A (1 point), B (2 points), etc. – denotes number of data points at that position.

Can specify plotting char: **= '<char>'**

Can plot with left-most char or digit of a var: **= <var>**

Options:

- **OVERLAY** puts multiple plots on same axes.
- **VPOS=<n>** and **HPOS=<n>** control dimensions of plot.
- Can label tickmarks and draw vertical and horizontal lines.

Examples:

Ex.1: ** Plot Y (vertical axis) vs X (horizontal) using A (default) as plotting char;

```
PROC PLOT;
  PLOT Y * X;
```

Ex.2: ** Plot Y (vertical axis) vs X (horizontal) using * as plotting char;

```
PROC PLOT;
  PLOT Y * X = '*';
```

Ex.3: ** Plot Y (vertical axis) vs X (horizontal) using M or F as plotting char;

```
PROC PLOT;
  PLOT Y * X = SEX;
```

Ex.4: ** Plot Y (vertical axis) vs X (horizontal) on one plot, and INCOME (vertical) vs X (horizontal) on separate plots, using A as plotting char in both plots;

```
PROC PLOT;
  PLOT Y * X  INCOME * X;
```

Ex.5: ** Plot Y (vertical axis) vs X (horizontal) on one plot, and INCOME (vertical) vs X (horizontal) on a separate plot, using A as plotting char in both plots, with both plots reduced to dimensions of 16 keystrokes vertically by 40 keystrokes horizontally;

```
PROC PLOT;
  PLOT Y * X  INCOME * X / VPOS=16 HPOS=40;
```

---

[3] PROC GPLOT produces high-resolution plots, but the syntax is somewhat different.

Ex.6: ** Plot Y and INCOME (both on vertical axis) vs X (horizontal), overlaying the two plots on same paper, using A as plotting char for Y and * for INCOME;

```
PROC PLOT;
  PLOT Y * X  INCOME * X = '*' / OVERLAY;
```

PROC CHART [4]

Function: Produces 5 different kinds of low resolution charts: horizontal bar, vertical bar, block, pie, star

Syntax:
```
PROC CHART DATA=<sas ds>;
    <kind of chart> <vars> / <options>;
    BY <vars>;
```

Notes:

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

5 kinds of charts:

VBAR <vars>	vertical bar charts (histograms)
HBAR <vars>	horizontal bar charts (histograms)
BLOCK <vars>	"city block" charts
PIE <vars>	pie charts
STAR <vars>	star charts

The variable(s) being charted is the one following the VBAR, HBAR, BLOCK, PIE, or STAR. This is the variable that determines the number of bars, blocks, wedges, or points.

A. Options for all 5 kinds of charts:

MISSING	missing values treated as valid data
DISCRETE	charted variable treated as discrete
TYPE =	tells what the bars, wedges, points represent on charts
FREQ (the default TYPE)	
CFREQ	
PCT	
CPCT	
SUM	(**SUM** & **MEAN** used only with **SUMVAR=**)
MEAN	
SUMVAR = <var>	shows mean or sum of <var> by charted variable's groups
MIDPOINTS = <values>	determines number of groups and their midpoints

B. Options for **VBAR**, **HBAR**, **BLOCK** only:

GROUP = <var>	produces side-by-side charts, one for each value of <var>
SUBGROUP = <var>	subdivides each bar by values of <var>

C. Option for **VBAR** only:

NOSPACE	no space between vertical bars of chart

Examples:

Ex.1: ** Two horizontal bars, one showing the number of males, the other the number of females in the dataset;

```
PROC CHART;
  HBAR SEX;
```

---

[4] PROC GCHART produces high-resolution charts, but the syntax is somewhat different.

Ex.2: ** Vertical histograms of AGE and INCOME, and pie chart of SEX (3 wedges: # males, # females, and # missing);

```
PROC CHART;
  VBAR AGE INCOME;
  PIE SEX / MISSING;
```

Ex.3: ** Show number of apartments having 1 room, 2 rooms, 3 rooms, etc. -- i.e. one bar for each value of NROOMS (the DISCRETE option forces one bar for each and prevents grouping multiple values into the same bar, like one bar for 1-3 rooms, one bar for 4-6 rooms, for example);

```
PROC CHART DATA=RENTALS;
  HBAR NROOMS / DISCRETE;
```

Ex.4: ** Two bars, one showing total female income, one showing total male income;

```
PROC CHART;
  HBAR SEX / SUMVAR=INCOME;
```

Ex.5: ** Two horizontal bars, one showing mean female income, one showing mean male income;

```
PROC CHART;
  HBAR SEX / SUMVAR=INCOME TYPE=MEAN;
```

Ex.6: ** Three bars, showing # people with income from 0 - 20000, 20000 - 40000, 40000 - up;

```
PROC CHART;
  VBAR INCOME / MIDPOINTS=10000, 30000, 50000;
```

Ex.7: ** Two side-by-side histograms of income, one for males, one for females;

```
PROC CHART;
  HBAR INCOME / GROUP=SEX;
```

Ex.8: ** F and M symbols appear in the bars for each department, representing # of employees of each sex;

```
PROC CHART;
  HBAR DEPT / SUBGROUP=SEX;
```

Ex.9: ** Side-by-side “city block”-style histograms of income for all departments (each city block being the histogram for a different department), with F and M symbols in each bar representing number of females and males in that bar;

```
PROC CHART;
  BLOCK INCOME / SUBGROUP=SEX GROUP=DEPT;
```

Ex.10: ** Side-by-side sets of three bars (one set of three bars for each department) representing total income in the three income ranges 0 - 20000, 20000 - 40000, 40000 - 60000, with F and M symbols in each bar representing total female and total male income in that bar;

```
PROC CHART;
  HBAR INCOME / GROUP=DEPT SUBGROUP=SEX SUMVAR=INCOME
     MIDPOINTS=10000, 30000, 50000;
```

PROC MEANS

Function:

- Produces simple summary statistics on numeric variables in SAS ds
- Does one-sample or matched-pairs T-tests

Syntax:
```
PROC MEANS DATA=<sas ds> <stat list>;
  CLASS <vars>;
  VAR <vars>;
  BY <vars>;
```

Notes:

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

If use **VAR**, only those vars named are analyzed.

If use **<stat list>**, only those statistics named are calculated; otherwise, up to 18 stats (depending on space -- usually only 6) are given for each variable analyzed.

To do one-sample T-test and get p-value, put **T PRT** in **<stat list>**. Note that this does (one-sample) two-tailed T-test of H_0: mean = 0.

CLASS statement has same effect as **BY**, but without requiring the SAS ds to be sorted.

Examples:

Ex.1 ** Means, stdevs, etc. for variables AGE and INCOME;
```
PROC MEANS;
  VAR AGE INCOME;
```

Ex.2 ** Test H_0: mean of score1 = mean of score2;
```
DATA RECORDS;
  INPUT NAME $ SEX $ SCORE1 SCORE2;
  DIFF = SCORE1 - SCORE2;
CARDS;
BILL M 95 88
  ...
PROC MEANS DATA=RECORDS T PRT;
  VAR DIFF;
```

Ex.3 ** Means, stdevs, etc. for variables AGE and INCOME, separately for females and males;
```
PROC MEANS;
  VAR AGE INCOME;
  CLASS SEX;
```

PROC UNIVARIATE

Function:

- Produces wealth of statistics on numeric vars in SAS ds
- Does one-sample or matched-pairs T-tests

Syntax:
```
PROC UNIVARIATE DATA=<sas ds> PLOT FREQ NORMAL;
    BY <vars>;
    VAR <vars>;
```

Notes:

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

If use **VAR**, only those variables named are analyzed.

Produces at least 27 statistics for each variable analyzed.

PLOT produces 3 plots: stem-and-leaf, box, normal probability.

FREQ produces frequency distribution of values.

NORMAL tests null hypothesis of normality for each variable.

Examples:

Ex.1: ** Get summary statistics on RESIDS, and test H_0: RESIDS are normally distributed;

```
PROC UNIVARIATE DATA=REGRESS  NORMAL;
  VAR RESIDS;
```

Ex. 2: ** Test H_0: mean of score1 = mean of score2;

```
DATA;
  INPUT NAME $ SEX $ SCORE1 SCORE2;
  DIFF = SCORE1 - SCORE2;
CARDS;
BILL M 95 88
  ...
PROC UNIVARIATE;
  VAR DIFF;
```

** Look at "Student's t" and "Pr > |t|" for the variable DIFF on the output. The T-test and p-value are automatically part of the output of UNIVARIATE.

PROC CORR

Function: computes correlations between pairs of variables in a SAS ds

Syntax: **PROC CORR DATA=<sas ds> PEARSON SPEARMAN KENDALL NOSIMPLE NOPROB;**
BY <vars>;
VAR <vars>;

Notes:

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

If use **VAR**, only those variables named are correlated.

Produces correlation table for all pairs of variables analyzed

- **PEARSON** is the default (ordinary correlations)
- **SPEARMAN** produces rank correlation
- **KENDALL** produces Kendall's tau

Also calculates many of PROC MEANS summary statistics. (Can suppress with **NOSIMPLE**.)

Also tests H_0: correlation = 0 for each correlation. (Can suppress with **NOPROB**.)

Example:

** Calculate ordinary correlations of all (nine) pairings of AGE, INCOME, and TAXES, and for each pairing test the hypothesis that the correlation is zero, suppressing PROC MEANS output;

```
PROC CORR NOSIMPLE;
  VAR AGE INCOME TAXES;
```

PROC TTEST

Function: does two-sample T-tests

Syntax:
```
PROC TTEST DATA=<sas ds>;
  CLASS <var>;
  BY <vars>;
  VAR <vars>;
```

Notes:

To use PROC TTEST properly, your dataset must contain a variable (like SEX) that distinguishes the observations in one sample from those in the other. This variable must be declared in the **CLASS** statement.

Do not confuse the two-sample T-test of PROC TTEST with the paired comparisons T-test that can be produced by PROC UNIVARIATE.

CLASS variable identifies the two samples.

CLASS variable must have two and only two values.

CLASS statement is required.

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

If use **VAR**, T-tests will be produced only for those variables named.

Each T-test is done twice:

- assuming the two pop variances are equal (EQUAL: homoscedastic)
- assuming the two pop variances are not equal (UNEQUAL: heteroscedastic)

Also tests whether the two pop variances are equal (F ' test)

Recommendation: If you do not know whether the two population variances are equal, then use the UNEQUAL test.

Example:

** Test H_0: mean male income = mean female income;

```
PROC TTEST;
  CLASS SEX;
  VAR INCOME;
```

PROC GLM

Function: regression, analysis of variance, analysis of covariance, specialized general linear models

Syntax: (Set-up for Regression)

```
PROC GLM DATA=<sas ds>;
  MODEL <dep var> = <ind var1> <ind var2> ... / <options>;
  BY <vars>;
  OUTPUT OUT=<out sas ds> P=<pred var> R=<res var>;
```

Notes:

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

MODEL is necessary and the dependent and independent variables must be in <sas ds>.

To regress transformed variables, the transformed variables must already be in <sas ds> in transformed condition (transform them in a previous DATA step).

The option **P** causes printing of observed, predicted, and residual values in the OUTPUT window.

Use **OUTPUT OUT=** to enable plotting residuals; **P=** adds predicted values to **<out sas ds>**; and **R=** adds residuals to **<out sas ds>**.

There are 4 parts to the output on the listing:

1) Overall ANOVA table
2) Miscellaneous statistics
3) Type I and Type III SS's
4) Parameter estimates

Examples:

Ex.1: ** Regress Y on X; yields $\hat{Y} = \alpha + \beta X$; print observed, predicted, and residual values;

```
PROC GLM;
  MODEL Y = X / P;
```

Ex.2: ** Multiple regression of Y on X_1 and X_2; yields $\hat{Y} = \alpha + \beta_1 X_1 + \beta_2 X_2$;

```
PROC GLM;
  MODEL Y = X1 X2;
```

Ex.3: ** Polynomial regression; yields $\hat{Y} = \alpha + \beta_1 X + \beta_2 X^2$ (X^2 need not be in SAS ds);

```
PROC GLM;
  MODEL Y = X X*X;
```

Ex.4: ** Use transformations; yields $\log(\hat{Y}) = \alpha + \beta_1 X + \beta_2 \sqrt{X}$;

```
DATA;
  INFILE 'TEST.DAT';
  INPUT Y X;
  LOGY = LOG(Y);
  SQRTX = SQRT(X);
PROC GLM;
  MODEL LOGY = X SQRTX;
```

Ex.5: ** Plot residuals vs predicted values and test H_0: residuals are normal;
** WORK.NEW = WORK.OLD + PREDY + RESIDS;

```
PROC GLM DATA=OLD;
  MODEL Y = U V;
  OUTPUT OUT=NEW P=PREDY R=RESIDS;
PROC PLOT DATA = OLD;
  PLOT RESIDS*PREDY;
PROC UNIVARIATE DATA=OLD  NORMAL;
  VAR RESIDS;
```

Syntax: (Set-up for Analysis of Variance)

```
PROC GLM DATA=<sas ds>;
  CLASSES <ind var1> <ind var2> ... ;
  MODEL <dep var> = <ind var1> <ind var2> ... / <options>;
  BY <vars>;
  OUTPUT OUT=<out sas ds> P=<pred var> R=<res var>;
```

Notes:
If use **BY**, SAS ds must be already sorted by the **BY** var(s).
MODEL is necessary and the dependent and independent variables must be in <sas ds>.
CLASSES is necessary and lists all of the independent vars.
Independent vars should be categorical or discrete vars with only a few distinct values.
OUTPUT OUT= not often used in ANOVA set-up.
There are 3 parts to the output on the listing:

1) Overall ANOVA table
2) Miscellaneous statistics
3) Type I and Type III SS's

Most common option is **SOLUTION**, which restores part 4 (parameter estimates) of output.
Examples:
Ex.1: ** Test H_0: mean male income = mean female income (assuming homoscedasticity);

```
PROC GLM;
  CLASS SEX;
  MODEL INCOME = SEX;
```

Ex.2: ** One-way ANOVA: Test H_0: Dept sales means are all equal;

```
PROC GLM;
  CLASS DEPT;
  MODEL SALES = DEPT;
```

Ex.3: ** Two-way ANOVA: Test H_0: Dept sales means are all equal; Test H_0: mean female sales = mean male sales;

```
PROC GLM;
  CLASSES DEPT SEX;
  MODEL SALES = DEPT SEX;
```

Ex.4: ** Two-way ANOVA with interactions: Test H_0: Dept sales means are all equal; Test H_0: mean female sales = mean male sales; Test H_0: All interaction means are zero;

```
PROC GLM;
  CLASSES DEPT SEX;
  MODEL SALES = DEPT SEX DEPT*SEX;
```

Ex.5: ** Estimate parameters of two-way ANOVA – **SOLUTION** option restores parameter estimates that are suppressed by default when a **CLASSES** statement is present;

```
PROC GLM;
  CLASSES DEPT SEX / SOLUTION;
  MODEL SALES = DEPT SEX;
```

Syntax: (Set-up for Analysis of Covariance)

```
PROC GLM DATA=<sas ds>;
  CLASSES <categorical var(s)> ;
  MODEL <dep var> = <ind var1> <ind var2> ... / <options>;
  BY <vars>;
  OUTPUT OUT=<out sas ds> P=<pred var> R=<res var>;
```

Notes:

In analysis of covariance (anacova), the independent variables are a mix of categorical and continuous variables. Essentially, anacova fits a set of regression lines – one line for each value of the categorical variable combinations.

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

MODEL is necessary and the dependent and independent variables must be in <sas ds>.

CLASSES is necessary and lists all of the categorical independent variables.

There are 3 parts to the output on the listing:

1) Overall ANOVA table
2) Miscellaneous statistics
3) Type I and Type III SS's

Most common option is **SOLUTION**, which restores part 4 (parameter estimates) of output.

Example:

Ex.1: ** Estimate equations of two parallel lines (one for males, one for females):

$Salary = \alpha_m + \beta \cdot Sales$ and $Salary = \alpha_f + \beta \cdot Sales$;

```
PROC GLM;
  CLASSES SEX;
  MODEL SALARY = SEX SALES;
```

Ex.2: ** Estimate equations of two lines (one for males, one for females): $Salary = \alpha_m + \beta_m \cdot Sales$ and $Salary = \alpha_f + \beta_f \cdot Sales$; and test $H_0 : \alpha_m = \alpha_f$ and $H_0 : \beta_m = \beta_f$; and estimate parameters;

```
PROC GLM;
  CLASSES SEX;
  MODEL SALARY = SEX SALES SEX*SALES / SOLUTION;
```

PROC FREQ

Function: produces tables, cross-tabulations, chi-square test of independence

Syntax:
```
PROC FREQ;
  TABLES <var1 * var2> <var3 * var4> ... / <options>;
  WEIGHT <var>;  BY <var(s)>;
```

Notes:

If use **BY**, SAS ds must be already sorted by the **BY** var(s).

Options:

- **EXPECTED** expected cell counts (f_e's) under Ho: independent
- **DEVIATION** prints $f_o - f_e$'s (observed counts – expected counts)
- **CELLCHI2** cell's contribution [$(f_o - f_e)^2 / f_e$] to chisquare statistic
- **CHISQ** chisquare test for independence
- **ALL** produces a set of measures of association
- **LIST** prints tables in list rather than crosstabulation format
- **MISSING** includes counts of missing values in frequencies

WEIGHT variable is used when cell counts are already known.

Examples:

Ex.1: ** List frequency counts of males and females;
```
PROC FREQ;
  TABLES SEX;
```

Ex.2: ** List frequency counts of males, females, and missing sex category;
```
PROC FREQ;
  TABLES SEX / MISSING;
```

Ex.3: ** List frequency counts for each distinct value of sex, department, state (separate list for each variable);
```
PROC FREQ;
  TABLES SEX DEPT STATE;
```

Ex.4: ** Cross-tab table of sex by department (counts for every combination of sex and department);
```
PROC FREQ;
  TABLES SEX*DEPT;
```

Ex.5: ** Three-way cross-tab (counts for all combinations of sex, dept, and state);
```
PROC FREQ;
  TABLES SEX*DEPT*STATE;
```

Ex.6: ** Cross-tab of sex by department and separate list of observation counts by state;
```
PROC FREQ;
  TABLES SEX*DEPT STATE;
```

Ex.7: ** Cross-tab of sex by dept, and test H_0: sex and dept are independent;

```
PROC FREQ;
  TABLES SEX*DEPT / CHISQ;
```

Ex.8: ** Cross-tab, and test independence of sex and dept, and show each cell's contribution to chi-square statistic (sum of contributions across all cells = chi-square statistic);

```
PROC FREQ;
  TABLES SEX*DEPT / CHISQ CELLCHI2;
```

Ex.9: ** Chi-square test of independence for table with already known cell counts – each row of input data shows counts for a different combination of sex and department;

```
DATA;
  INPUT SEX $ DEPT $ COUNT;
CARDS;
F Billing 43
F Payroll 32
M Billing 51
M Payroll 17
PROC FREQ;
  TABLES SEX*DEPT / CHISQ;
  WEIGHT COUNT;
```