Introduction to SAS Statistical Package

Lecture 4

SAS Procedures

Class 4

Descriptive statistics

- exploratory data analysis is very important from many perspectives
- in SAS there are three procedures used routinely

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>tables for categorical data</td>
</tr>
<tr>
<td>univariate</td>
<td>Descriptive statistics for numeric data</td>
</tr>
<tr>
<td>means</td>
<td>Descriptive statistics for numeric data</td>
</tr>
</tbody>
</table>

Example data: CHOLEX

Subset of the Johns Hopkins Precursors Study. Examine risk factors and development of cardiovascular disease.

Example data: CHOLEX

AGE    graduation age
BMI    body mass index
CHD    CHD
CHDAGE
CHDYR
CHOLEST cholesterol
COFFEE coffee cups
CVD    CVD
CVDAGE
CVDYR
DBP    diastolic BP
FOLLAGE followup age
FOLLTIME followup time
ID     ID
LCY    LCY
MI     MI
MIYR
NA     physical activity
PHYACT
SBP    systolic BP
SNOKER
current smoker
Systolic BP
YOB    year of birth

proc freq

- produces frequency counts and cross-tabulation tables
- computes tests and measures of association

Syntax:

proc freq <options>;
tables requests / <options>;
**proc freq**

Example CLASS4_1.SAS:

```sas
proc freq data=mylib.cholex;
    tables smoker chd;
    tables smoker* chd / chisq relrisk;
run;
```

*data=filename is an option*

*chisq and relrisk are requests for statistics*

---

**proc univariate**

- produces simple descriptive statistics
- use PLOT options on PROC statement
- stem-and-leaf plot
- box plot
- normal probability plot (QQ plot)
- side by side box plots for by variable groups

**Syntax:**

```sas
proc univariate <options>;
    var variables / <options>;
```

**proc univariate : Class4_2.sas**

Example:

```sas
proc univariate data=mylib.cholex plot;
title "Univariate Output for CHOLEX";
var age bmi chdage dbp;
run;
```

---

**proc means**

- similar to univariate – no plots
- nicer output, particularly for more than one variable

**Syntax:**

```sas
proc means <options>;
    class varlist;
    var variables / <options>;
    by varlist;
run;
```
**proc means options**
- `data=dataset`
- `statistic`
  - default is: n mean std min max
  - Others are: nmiss range median clm

**statements**
- `class` statistics produced for each combination of class variable
- `by` statistics produced by each combination of by variables

---

**Example:**
```
proc means data=mylib.cholex n
   nmiss mean median stderr range;
   title "Means Output";
   var age bmi chdage dbp;
run;
```

**Check Output**

---

**SAS Formats**
- It is sometimes useful to store data in one way and display it in another. For example, dates can be stored as integers but displayed in human readable format.
- A **SAS format** changes the way the data stored in a variable is displayed. There are two types of format:
  - Internal formats (SAS already knows about these)
  - User defined formats (you define these yourself).

---

**Example with Class statement:**
```
proc means data=mylib.cholex n
   mean median stderr range;
   class smoker;
   var age bmi chdage dbp;
   title "Means Output by Smoking Status";
run;
```

---

**Internal SAS formats : Class4_5.SAS**

---

```
proc print data=mydata;
   var x y;
run;
```

---

```
proc print data=mydata;
   var x y;
   format x 6.2 y intob6.;
run;
```

---

```
proc print data=mydata;
   var x y;
   format x 5. y n3hyfg.;
run;
```

---

```
proc print data=mydata;
   var x y;
   format x 6.2 y intob6.;
run;
```
Permanent formats: Class4_6.SAS

A format statement added to a datastep permanently connects the format to a variable. The format information is stored in the dataset header.

```sas
data mydata;
  x = 123.456789001234;
  y = "Hello world!"
format x 20.14 y data7.;
run;
```

```sas
proc print data=mydata;
var x y;
run;
```

```sas
0 123.4567890876120001 12A0403
```

User defined formats: Class4_7.SAS

- Define the format using `proc format`
- Tell SAS to use the format with a specific variable by using the `format statement` as before.

```sas
proc freq data=mylib.chol; table cvd;
run;
```

```sas
Frequency Missing = 19
```

User defined formats

`proc format` defines the format `cvdlb`. The format statement applies the format to the variable `cvd`.

```sas
proc format;
   value cvdlb 1='present' 0='absent';
run;
```

```sas
proc freq data=mylib.chol; table cvd;
format cvd cvdlb.;
run;
```

```sas
Cumulative    Cumulative
CVD    Frequency     Percent     Frequency      Percent
            208       72.47           208        72.47
          287       100.00
```

Syntax:

```sas
proc format <options>;
    value FormatName
        range1 = 'formatted value1'
        ....
        rangeN = 'formatted valuen';
run;
```

Used to define a format.

User defined formats: Example

```sas
proc format;
   value gen
       1 = 'male'
       2 = 'female';
   value age
       10-29 = '10 - 29'
       30-39 = '30 - 39'
       40-49 = '40 - 49'
       50-75 = '50 - 75';
   value $dpt
       'A' = 'Dept A.'
       'B' = 'Dept B.';
run;
```

Defines three formats, gen, age and dpt.

Format dpt is a character format suitable for character variables.

Format names:

- must be 8 or fewer characters long
- cannot end with a number
- character formats begin with a $ character
- can not use a SAS internal format name
- refer to format in format statement by using the name followed by a period
format ranges

- You can specify a range of values to be formatted in a given way.

```
proc format;
  value age
    10-29 = '10 - 29'
    30-39 = '30 - 39'
    40-49 = '40 - 49'
    50-75 = '50 - 75';
run;
```

inclusive ranges

- you can use formats as look-up tables to categorize a variable.

Example: Class4_8.sas

```
proc format;
  value sbplb low-100='<=100'
    100<-120='100-120'
    120<-high='>120';
  data a; set mylib.cholex;
    sbpcopy=sbp;
  proc print data=a;
    var id sbp sbpcopy;
    format sbp sbplb. id 6.;run;
```

specifying format ranges

- low  lowest value (excludes missing)
- high  highest value
- other all other values not listed (including missing values)

```
  value1 -  value2  means [value1,value2]
  value1 <- value2  means [value1,value2)
  value1 <- value2  means (value1,value2]
```

Example Class4_9.sas

```
Examine smoking and CHD.
proc format;
  value smokelb 0 ='nonsmoker'
    1='smoker';
  proc freq data=mylib.cholex;
    tables smoker chd /
        chisq relrisk;
    title 'Freq output with formats';
    format smoker  smokelb. chd cvdlb.;
run;
```
Modeling with SAS

- examine relationships between variables
- estimate parameters and their standard errors
- calculate predicted values
- evaluate the fit or lack of fit of a model
- test hypotheses
- design
- outcome

The linear model

\[ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \varepsilon \]
\[ \varepsilon \sim N(0, \sigma^2) \]

Example:

Weight = \beta_0 + \beta_1 \text{Height} + \beta_2 \text{Age} + \varepsilon

Note: outcome variable must be continuous and normal given independent variables

the linear model with proc reg

- estimates parameters by least squares
- produces diagnostics to test model fit (e.g. scatter plots)
- tests hypotheses

Example:

```
proc reg data=mydata;
  model weight = height age;
run;
```

proc reg

Syntax:
```
proc reg <options>;
  model response = effects </options>;
  plot yvariable*xvariable = 'symbol';
  by variable -list;
  output <OUT=SAS data set>
    <output statistic list>;
run;
```
**proc reg**

**proc reg statement syntax:**

- **data = SAS data set name**
  - input data set
- **outest = SAS data set name**
  - creates data set with parameter estimates
- **simple**
  - prints simple statistics

**proc reg**

**the model statement**

- model response=<effects>/options;
  - required
  - variables must be numeric
  - many options
  - can specify more than one model statement

Example:

```sas
model weight = height age;
model weight = height age / p clm cli;
```

**proc reg**

**the plot statement**

- plot yvariable*xvariable <=symbol> </options>;
  - produces scatter plots - yvariable on the vertical axis and xvariable on the horizontal axis
  - can specify several plots
  - optional symbol to mark points
  - yvariable and xvariable can be variables specified in model statements or statistics available in output statement

Example:

```sas
plot weight * age / pred;
plot r. * p. / vref = 0;
plot (weight p. l95. U95.) * age / overlay;
```

**proc reg**

**some statistics available for plotting:**

- P. predicted values
- R. residuals
- L95. lower 95% CI bound for individual prediction
- U95. upper 95% CI bound for individual prediction
- L95M. lower 95% CI bound for mean of dependent variable
- U95M. upper 95% CI bound for mean of dependent variable

Example:

```sas
plot weight * age / pred;
plot r. * p. / vref = 0;
plot (weight p. l95. U95.) * age / overlay;
```

**proc reg**

**the output statement**

- output <OUT=SAS data set> keywords=names;
  - creates SAS data set
  - all original variables included
  - keyword=names specifies the statistics to include

Example:

```sas
output out=pvals p=pred r=resid;
```

**Example: Class4_10.sas**

**variables of interest:**

- **sbp** – systolic blood pressure
- **age** – age
- **bmi** – body mass index
**proc reg example: Class4_10.sas**

```sas
proc reg data=mylib.cholex;
model sbp= age bmi;
plot r.*p.*age ;
output out=regout predicted=pv ;
proc print data=regout (obs=10);
run;

here:
1. model estimate parameters etc
2. plot make two plots
3. output make an output dataset regout
```

**Check Output**

**The run statement**

Many people assume that the `run` statement ends a procedure such as `proc reg`.

This is because when SAS encounters a `run` statement, it executes any outstanding instructions in the program buffer. But it may or may not end the procedure.

```sas
proc reg data=cholex;
model sbp = age bmi;
run;
model sbp = age bmi;
plot r.*bmi;
run;
quit; /* ends the procedure */
```

**Check Output**

**reg and glm**

- `proc reg` and `proc glm` procedures are suitable only when the outcome variable is normally distributed.
- `proc reg` has many regression diagnostic features, while `proc glm` allows you to fit more sophisticated linear models such as random effects models, models for unbalanced designs etc.

**non-normal outcomes**

- In many situations we cannot assume our response variable is normally distributed.
- `proc reg` and `proc glm` are not suitable for modeling such outcomes.

**Example:**

Suppose you are interested in estimating the prevalence of disease in a population. You have an indicator of disease (1 = Yes, 0 = No)
non-normal outcomes

**Example:**
You are interested in estimating how the incidence of infant mortality has changed as a function of time.

---

**proc logistic**

- fits linear logistic regression models for binary or ordinal response data by the method of maximum likelihood
- enables you to specify categorical variables (also known as CLASS variables) as explanatory variables. It also enables you to specify interaction terms in the same way as in the GLM procedure.

**proc logistic**

**Syntax:**

```sas
proc logistic <options>;
   by variables;
   class variables </options>;
   model response = effects </options>;
run;
```

**Example :** Class5_4.sas

---

**proc logistic**

- **Proc logistic** statement says which variables are classification (categorical) variables
- By default, PROC LOGISTIC assigns Ordered Value 1 to response level 0, causing the probability of the nonevent to be modeled.
- specify the DESCENDING option in the PROC LOGISTIC statement, which reverses the default ordering of Y from (0,1) to (1,0), making 1 (the event) the level with Ordered Value 1:

```sas
proc logistic data=mylib.cholex descending;
   model chd = bmi age ;
```

**proc logistic:**

- `class` statement says which variables are classification (categorical) variables
- `by` statement produces a separate analysis for each level of the `by` variables (data must be sorted in the order of the `by` variables)
- `response` variable is the response (dependent) variable in the regression model.
- `<effects>` are a list of variables. These are the independent variables in the regression model. Any independent variables that are categorical must be listed in the Class statement.

**proc logistic:**

- **class** statement
  - classification (categorical) variables
  - Option `param=ref`
    Parameter estimates of CLASS main effects using the reference coding scheme estimate the difference in the effect of each non-reference level compared to the effect of the reference level.

```sas
proc logistic data=mylib.cholex descending;
   model chd = bmi age ;
   class smoker / param=reference;
```
**proc logistic:**

- Check Online Doc for a further description of the options and statements available for the logistic procedure

**example: logistic regression**

- Perform a logistic regression analysis to determine how the odds of CHD are associated with age, BMI, and smoking status in the cholex file

---

**Example: Class4_11.sas**

```sas
proc logistic data=mylib.cholex descending;
class smoker / param=reference ; *add descending option to get 1 vs 0;
model chd = age bmi smoker ;
*add format for smoker;
run;
```

- **descending** option on PROC statement means that we are modeling the probability that chd=1 and not the probability that chd=0.

---

**survival analysis**

- survival analysis is a class of statistical methods for studying the occurrence and timing of events

---

**proc lifetest**

- produces estimates of survivor functions using Kaplan-Meier or actuarial method
- tests the null hypothesis that the survivor functions are identical for two or more groups
**proc lifetest**

Syntax:

```
proc lifetest <options>;
   time variable <*censor(list)>;
   by variables;
   freq variables;
   id variables;
   strata variable <(list)> < ... variable <(list)>>;
   test variable;
run;
```

**proc lifetest options**

```
data = data set
method = KM default
notable suppresses printing of survival function estimates
plots = (S,LS,LLS,H,P)
```

Example:

```
proc lifetest
data=mylib.mydata
   plots=(survival,hazard);
```

**the time statement**

```
time variable*censor(list);
```

where:
- variable - time to event variable
- censor - censoring indicator
- list - list of censored values

Example:

```
time t * flag(1,2);
```

**the strata statement**

```
strata variables;
```

where:
- variables - variables that determine groups

Example:

```
time t * flag(1,2);
strata gender;
```

**Example data: CHOLEX**

The CHOLEX dataset is a subset of the data from the Johns Hopkins Precursors Study.

Want to compare time to cerebrovascular disease (cvd) for persons with different cholesterol levels (cholest).

Strategy:
- create format for cholest variable
- Kaplan-Meier plots
- Use cvdage (age at CVD) as the time variable
- cvd - censoring indicator
- value of 0 indicates a censored individual
- cholest - strata variable categorized into 4 groups by format

**Example: CLASS4_12.SAS**

```
%let data = survival\CVS\class4_12.sas;
%let format :
   value chol low = "<=174"
   175-184 = "174-184"
   185-214 = "185-214"
   high = ">214"

proc lifetest data=survival.choles
   plots=survival,ls,loglog, notable;
   time cvdage*cvd(0);
   strata cholest;
   format chol. choles;
run;
```
**proc phreg**

- uses Cox’s partial likelihood method to estimate regression models with censored data
- handles both continuous-time and discrete-time data
- performs stratified analyses
- allows time-dependent covariates
- allows counting process formulation input and thus recurring outcomes
- perform conditional logistic regression analysis for matched case-control studies

**Syntax:**

```
proc phreg <options>;
    model response*censor(list) = variables / <options>;
    strata variable <list>;
    output out=name <keywords=names>;
run;
```

Note: no class statement, create your own dummy variables for categorical explanatory variables

**Example:**

```
proc phreg data=mylib.mydata;
    model survtime * status(0) = sex treatment;
    strata agegroup;
run;
```

Example: CLASS4_13.SAS

```
/* create dummies for levels of cholest */
data temp; set survival.cholex;
    if cholest = . then group1 = .;
    else group1 = (cholest <= 173);
    if cholest = . then group2 = .;
    else group2 = (cholest > 174 and cholest <= 193);
    if cholest = . then group3 = .;
    group3 = (cholest > 194 and cholest <= 214);
run;

proc phreg data=temp;
    model cvdage*cvd(0) = group1 group2 group3 /risklimits;
    output out=output survival=s logsurv=logs loglogs=loglogs;
run;
```