Data summary and analysis

All programs and data sets used for these notes are available from my course website. New files that we have not used before are cereal_DataSummary.sas, gpa_DataSummary.sas, placekick.sas, and placekick.csv.

Means and more

We will examine `proc means` (Base SAS) in depth here to see commonly used syntax in procedures, to specify particular options, and to explore the help. Below is the syntax given in the help:

PROC MEANS <*option(s)*> <*statistic-keyword(s)*>;

BY <DESCENDING> *variable-1* <<DESCENDING> *variable-2* ...> <NOTSORTED>;

CLASS *variable(s)* </ *option(s)*>;

FREQ *variable*;

ID *variable(s)*;

OUTPUT <OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
<*id-group-specification(s)*> <*maximum-id-specification(s)*>
<*minimum-id-specification(s)*> </ *option(s)*> ;

TYPES *request(s)*;

VAR *variable(s)* </ WEIGHT=*weight-variable*>;

WAYS *list*;

WEIGHT *variable*;

Descriptions of commonly used statements:

- `var`: Variables to use in the calculations
- `class`: Perform calculations on groups defined by these variables
- `by`: Perform calculations on groups defined by these variables (somewhat similar to class); the data needs to be sorted

- `output`: Create a new data set with some of the calculations; this can be very useful for a further analysis. Another way to create a new data set will be shown later using the output delivery system.

These types of statements are available in most procedures! Also, while the `where` statement is not shown in the syntax, this can be used here too.

Below are a few examples using the cereal data:

```
title1 "Chris Bilder, STAT 850";

proc import out=cereal datafile="C:\data\cereal.csv"
    DBMS=CSV replace;
  getnames=yes;
  datarow=2;
run;

data set1;
  set cereal;
  sugar = sugar_g/size_g;  *sugar content per cereal divided by
    serving size;
  fat = fat_g/size_g;
  sodium = sodium_mg/size_g;
  keep ID Shelf Cereal sugar fat sodium;
run;

title2 "Cereal data adjusted for serving size";
proc print data=set1(obs = 5);
run;

title2 "Means for cereal data";
proc means data=set1;
  var sugar fat sodium;
  output out = out_set1;
run;

title2 "Out data set from proc means";
proc print data=out_set1;
run;
```

Chris Bilder, STAT 850
Cereal data adjusted for serving size

Obs	ID	Shelf	Cereal	sugar	fat	sodium
1	1	1	Kellog's Razzle Dazzle Rice Crispies	0.35714	0.000000	6.0714
2	2	1	Post Toasties Corn Flakes	0.07143	0.000000	9.6429
3	3	1	Kellogg's Corn Flakes	0.07143	0.000000	10.7143
4	4	1	Food Club Toasted Oats	0.06250	0.062500	8.7500
5	5	1	Frosted Cheerios	0.43333	0.033333	7.0000

Chris Bilder, STAT 850
Means for cereal data

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
sugar	40	0.2894156	0.1495599	0	0.5555556
fat	40	0.0321828	0.0276879	0	0.0925926
sodium	40	5.6147038	2.4625271	0	10.7142857

Chris Bilder, STAT 850
Out data set from proc means

Obs	_TYPE_	_FREQ_	_STAT_	sugar	fat	sodium
1	0	40	N	40.0000	40.0000	40.0000
2	0	40	MIN	0.0000	0.0000	0.0000
3	0	40	MAX	0.5556	0.0926	10.7143
4	0	40	MEAN	0.2894	0.0322	5.6147
5	0	40	STD	0.1496	0.0277	2.4625

```
title2 "Means for cereal data for shelf class";
proc means data=set1;
  class shelf;
  var sugar fat sodium;
```

```
run;
```

Chris Bilder, STAT 850
Means for cereal data for shelf class

The MEANS Procedure

Shelf	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
1	10	sugar	10	0.2568366	0.1672957	0.0606061	0.4444444
		fat	10	0.0261223	0.0335836	0	0.0925926
		sodium	10	8.0386045	1.6672840	5.8064516	10.7142857
2	10	sugar	10	0.4149686	0.0900102	0.3000000	0.5555556
		fat	10	0.0448224	0.0271435	0.0166667	0.0925926
		sodium	10	5.2731677	1.7455984	1.8518519	7.6666667
3	10	sugar	10	0.2303732	0.1577006	0	0.4516129
		fat	10	0.0296143	0.0289136	0	0.0925926
		sodium	10	4.4610216	2.8862230	0	9.3548387
4	10	sugar	10	0.2554839	0.1101023	0.1000000	0.4000000
		fat	10	0.0281720	0.0194388	0	0.0545455
		sodium	10	4.6860215	1.7393711	1.8181818	7.0967742

```
title2 "Means for cereal data by shelf";
proc means data=set1;
  var sugar fat sodium;
  by shelf;
run;
```

Chris Bilder, STAT 850
Means for cereal data by shelf

The MEANS Procedure

Shelf=1

Variable	N	Mean	Std Dev	Minimum	Maximum
sugar	10	0.2568366	0.1672957	0.0606061	0.4444444
fat	10	0.0261223	0.0335836	0	0.0925926
sodium	10	8.0386045	1.6672840	5.8064516	10.7142857

Shelf=2

Variable	N	Mean	Std Dev	Minimum	Maximum
sugar	10	0.4149686	0.0900102	0.3000000	0.5555556
fat	10	0.0448224	0.0271435	0.0166667	0.0925926
sodium	10	5.2731677	1.7455984	1.8518519	7.6666667

Shelf=3

Variable	N	Mean	Std Dev	Minimum	Maximum
sugar	10	0.2303732	0.1577006	0	0.4516129
fat	10	0.0296143	0.0289136	0	0.0925926
sodium	10	4.4610216	2.8862230	0	9.3548387

Shelf=4

Variable	N	Mean	Std Dev	Minimum	Maximum
sugar	10	0.2554839	0.1101023	0.1000000	0.4000000
fat	10	0.0281720	0.0194388	0	0.0545455
sodium	10	4.6860215	1.7393711	1.8181818	7.0967742

On your own, investigate what happens with the last two procedure implementations when `output out=out_set1` is added before `run`.

Most procedures have a number of options that are available on the `proc` line. By selecting the <options> item in the syntax help, you will obtain a longggg list of options. Below is how I use

a few of them.

```
title2 "Show how to use options in proc means";
proc means data=set1 mean alpha=0.05 clm median p50 std;
  class shelf;
  var sugar fat sodium;
  output out=out_set1;
run;

title2 "Out data set from proc means";
proc print data=out_set1;
run;
```

Chris Bilder, STAT 850
Show how to use options in proc means

The MEANS Procedure

Shelf	N Obs	Variable	Mean	Lower 95% CL for Mean	Upper 95% CL for Mean	Median	50th Pctl	Std Dev
1	10	sugar	0.2568366	0.1371605	0.3765127	0.3440860	0.3440860	0.1672957
		fat	0.0261223	0.0020981	0.0501465	0.0086207	0.0086207	0.0335836
		sodium	8.0386045	6.8459014	9.2313076	7.4968072	7.4968072	1.6672840
2	10	sugar	0.4149686	0.3505792	0.4793580	0.4203704	0.4203704	0.0900102
		fat	0.0448224	0.0254051	0.0642398	0.0339080	0.0339080	0.0271435
		sodium	5.2731677	4.0244419	6.5218936	5.5363985	5.5363985	1.7455984
3	10	sugar	0.2303732	0.1175610	0.3431854	0.2569024	0.2569024	0.1577006
		fat	0.0296143	0.0089308	0.0502979	0.0192593	0.0192593	0.0289136
		sodium	4.4610216	2.3963420	6.5257011	5.1349655	5.1349655	2.8862230
4	10	sugar	0.2554839	0.1767215	0.3342463	0.2818182	0.2818182	0.1101023
		fat	0.0281720	0.0142663	0.0420778	0.0327957	0.0327957	0.0194388
		sodium	4.6860215	3.4417504	5.9302926	4.8636364	4.8636364	1.7393711

Chris Bilder, STAT 850
Out data set from proc means

Obs	_TYPE_	_FREQ_	_STAT_	sugar	fat	sodium
1	0	40	N	40.0000	40.0000	40.0000
2	0	40	MIN	0.0000	0.0000	0.0000
3	0	40	MAX	0.5556	0.0926	10.7143
4	0	40	MEAN	0.2894	0.0322	5.6147
5	0	40	STD	0.1496	0.0277	2.4625

Comments:

- The order of the statements or options do not matter for most procedures.
- Can you make some informal statements regarding differences in the nutritional content (sugar, fat, sodium) across the shelves?
- The `noprint` option is helpful if you just want the output data set without any information sent to the Results Viewer.
- More than one `output` statement can be used in a procedure.
- Notice that the output data set did not change. Options need to be specified again to have particular items put into the data set. What do you think the output data set will look like with the following `output` statements?:

```
output out=out_set2 mean=mean LCLM=lower UCLM=upper;
output out=out_set3 mean(sodium) = mean LCLM = lower UCLM =
  upper;
output out=out_set4 mean(sugar) = mean1 mean(sodium) = mean2
  LCLM = lower UCLM = upper;
output out=out_set5 mean=mean LCLM=lower UCLM=upper /
  autoname;
```

  The last output statement shows how options can be included. Many statements within a procedure have options given after a forward slash.
- The confidence levels used by the intervals generated from the `output` statement are given by the options in the `proc` line.

Regression

The purpose of the GPA data set was to examine the relationship between high school and college GPA with a simple linear regression model. Let Y denote the college GPA, and let x denote the high school GPA. I want to estimate the model $E(Y) = \beta_0 + \beta_1 x$ where Y has a normal distribution with a variance σ^2, β_0 is a

y-intercept parameter, and β_1 is a slope parameter. We have 20 observations in the data set, so we have 20 observed values of Y: $y_1 = 3.1, \ldots, y_{20} = 2.6$ and 20 observed values of x: $x_1 = 3.04, \ldots, x_{20} = 2.88$. I can estimate this model using `proc reg` (SAS STAT):

```
title1 "Chris Bilder, STAT 850";

data set1;
  infile "C:\data\gpa.csv" firstobs=2 delimiter=",";
  input HS College;
run;

title2 "Estimate model for GPA data";
proc reg data=set1;
  model College = HS;
run;
```

Chris Bilder, STAT 850
Estimate model for GPA data

The REG Procedure
Model: MODEL1
Dependent Variable: College

Number of Observations Read	20
Number of Observations Used	20

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	2.89817	2.89817	24.54	0.0001
Error	18	2.12621	0.11812		
Corrected Total	19	5.02437			

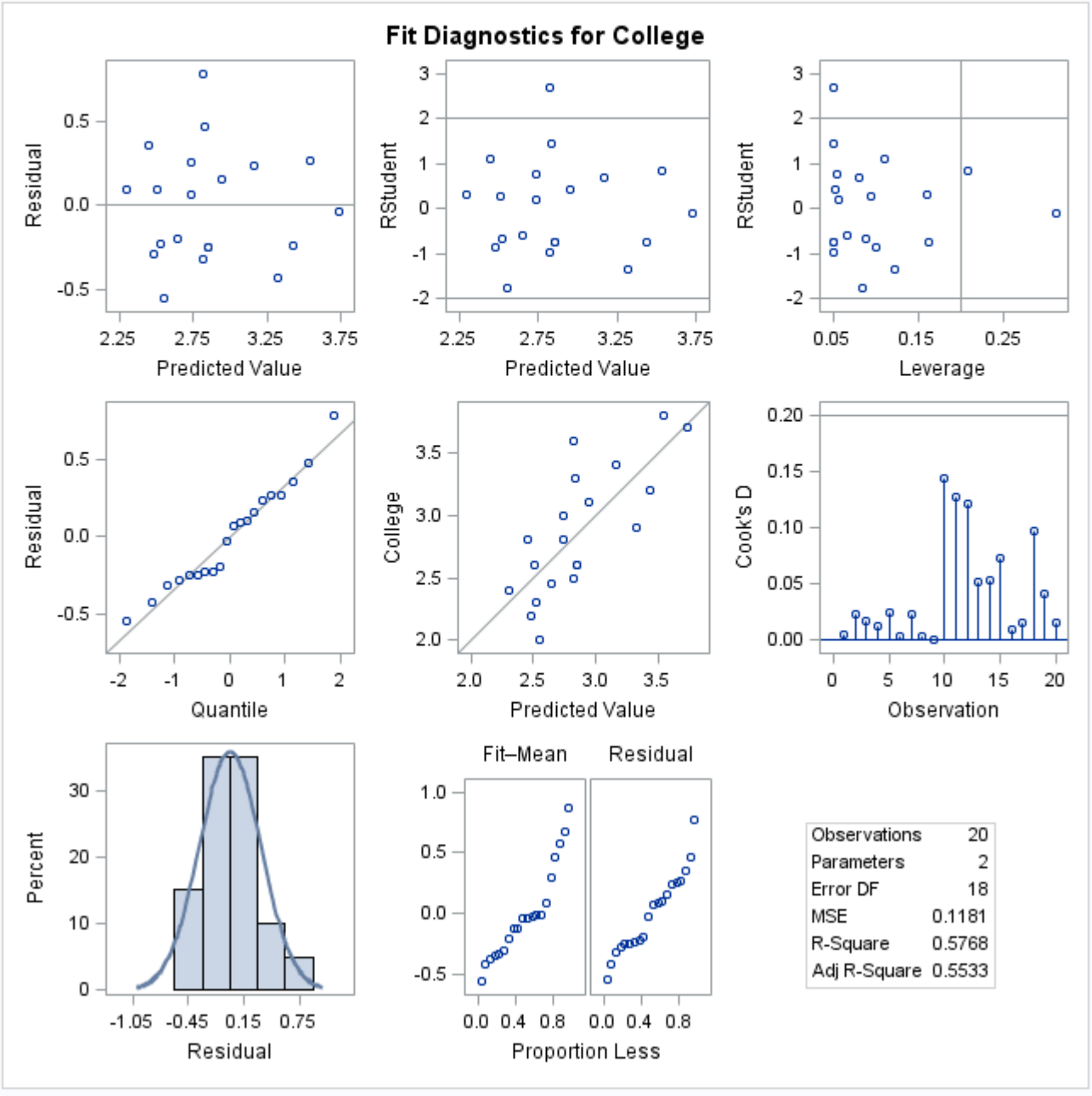
Root MSE	0.34369	R-Square	0.5768
Dependent Mean	2.86250	Adj R-Sq	0.5533
Coeff Var	12.00662		

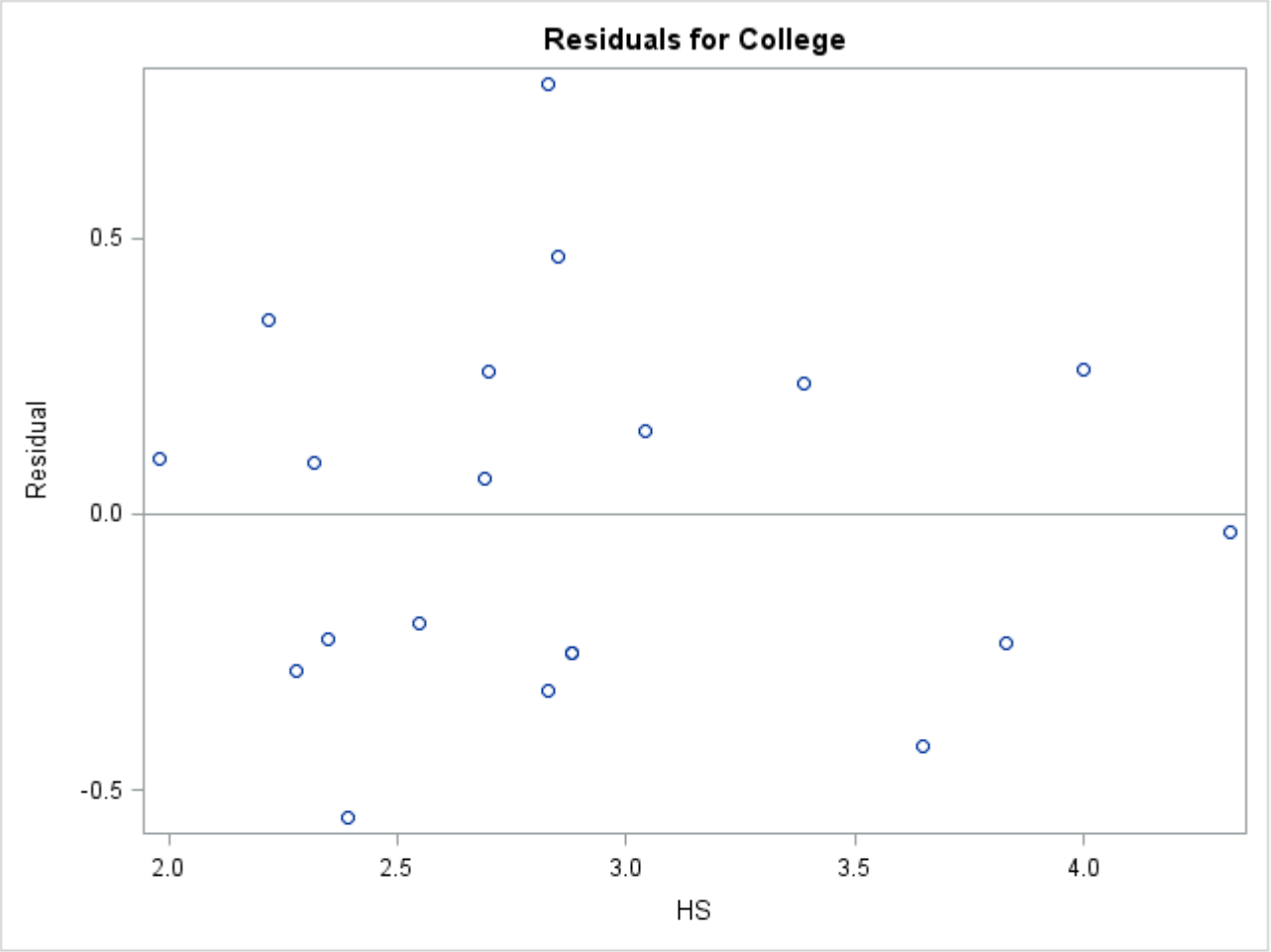
Parameter Estimates

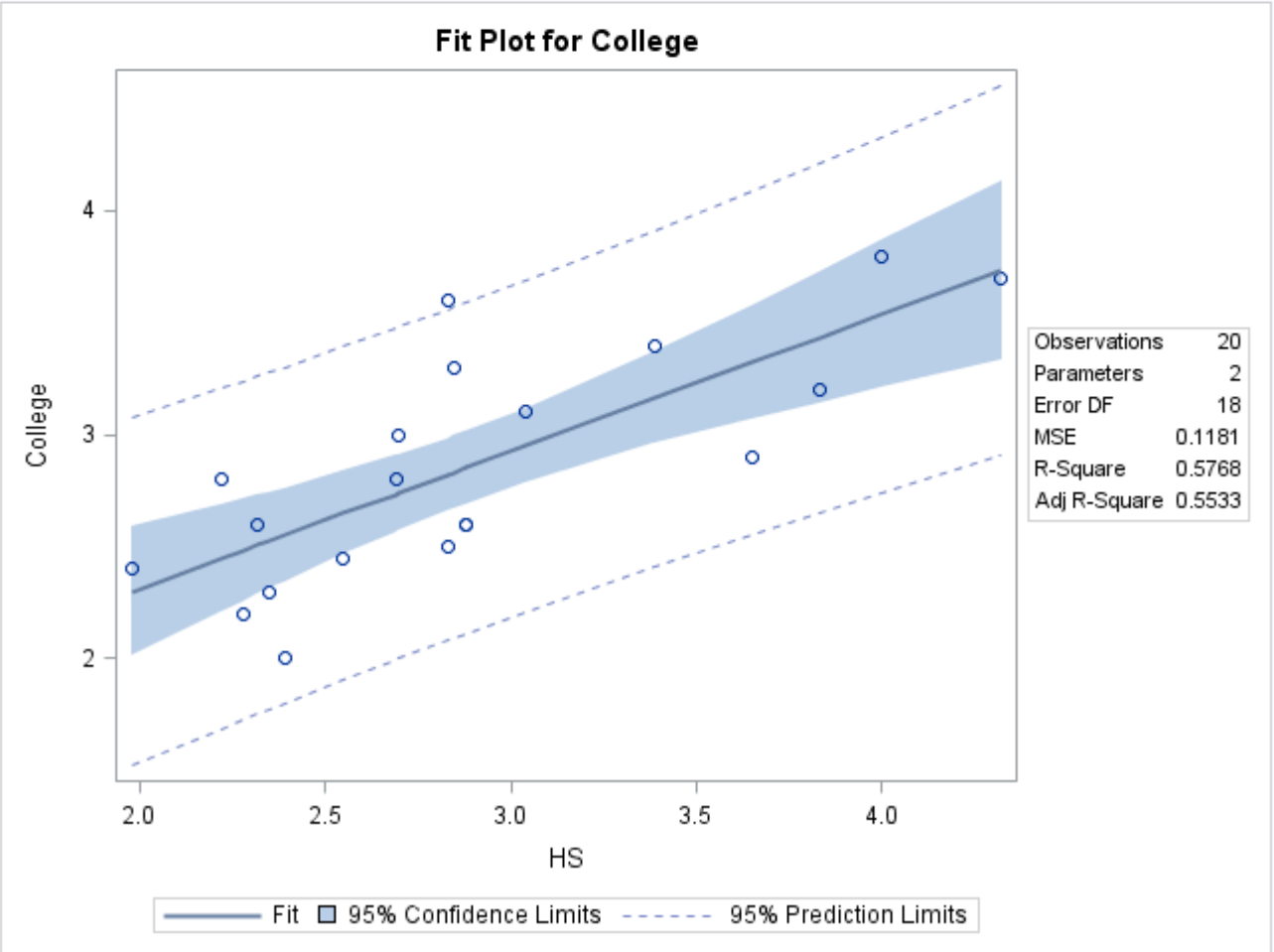
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1.08688	0.36662	2.96	0.0083
HS	1	0.61249	0.12365	4.95	0.0001

Chris Bilder, STAT 850
Estimate model for GPA data

The REG Procedure
Model: MODEL1
Dependent Variable: College




Residuals for College
Residual
0.5
0.0
-0.5
2.0
2.5
3.0
3.5
4.0
HS


Fit Plot for College
College
4
3
2
2.0
2.5
3.0
3.5
4.0
HS
Observations 20
Parameters 2
Error DF 18
MSE 0.1181
R-Square 0.5768
Adj R-Square 0.5533
Fit
95% Confidence Limits
95% Prediction Limits

The estimated regression model is $\hat{Y} = 1.0869 + 0.6125x$. Below are a few questions that can be answered by examining the output:

- How well does the model fit the data?
- Is there sufficient evidence to indicate a linear relationship between college and high school GPA?
- What do you think about the normality assumption for Y?
- What is the 95% confidence interval for the mean college GPA for students with a high school GPA of 3.0?

SAS can take a few minutes to run the previous code due to the plots that are created by default. I find this behavior unacceptable! To prevent these plots from being created, one can run the same `proc reg` code again but with "ODS statements":

```
ods graphics off;

proc reg data=set1;
  model College = HS;
run;

ods graphics on; *default;
```

The output now will be generated very quickly. We will discuss ods statements in more detail later.

Below is the syntax for `proc reg` from the help:

Syntax: REG Procedure

The following statements are available in the REG procedure:

PROC REG <*options*>;
- <*label*:> **MODEL** *dependents* = <*regressors*> </ *options*>;
- **BY** *variables*;
- **FREQ** *variable*;
- **ID** *variables*;
- **VAR** *variables*;
- **WEIGHT** *variable*;
- **ADD** *variables*;
- **CODE** <*options*>;
- **DELETE** *variables*;
- <*label*:> **MTEST** <*equation, …, equation*> </ *options*>;
- **OUTPUT** <OUT=*SAS-data-set*> <*keyword=names*> <*…keyword=names*>;
- **PAINT** <*condition* |ALLOBS> </ *options*> |<STATUS |UNDO>;
- **PLOT** <*yvariable*xvariable*> <=*symbol*> <*…yvariable*xvariable*> <=*symbol*> </ *options*>;
- **PRINT** <*options*> <ANOVA> <MODELDATA>;
- **REFIT** ;
- **RESTRICT** *equation, …, equation*;
- **REWEIGHT** <*condition* |ALLOBS> </ *options*> |<STATUS |UNDO>;
- **STORE** <*options*>;
- <*label*:> **TEST** *equation*, <, …, *equation*> </ *option*>;

Selecting the `proc reg` line leads to a display of the following options:

The REG Procedure

PROC REG Statement

PROC REG *<options>*;

The PROC REG statement invokes the REG procedure. The PROC REG statement is required. If you want to fit a model to the data, you must also use a MODEL statement. If you want to use only the PROC REG options, you do not need a MODEL statement, but you must use a VAR statement. If you do not use a MODEL statement, then the COVOUT and OUTEST= options are not available.

Table 85.1 summarizes the *options* available in the PROC REG statement. Note that any *option* specified in the PROC REG statement applies to all MODEL statements.

Table 85.1: PROC REG Statement Options

Option	Description
Data Set Options	
DATA=	Names a data set to use for the regression
OUTEST=	Outputs a data set that contains parameter estimates and other model fit summary statistics
OUTSSCP=	Outputs a data set that contains sums of squares and crossproducts
COVOUT	Outputs the covariance matrix for parameter estimates to the OUTEST= data set
EDF	Outputs the number of regressors, the error degrees of freedom, and the model R square to the OUTEST= data set
OUTSEB	Outputs standard errors of the parameter estimates to the OUTEST= data set
OUTSTB	Outputs standardized parameter estimates to the OUTEST= data set. Use only with the RIDGE= or PCOMIT= option.
OUTVIF	Outputs the variance inflation factors to the OUTEST= data set. Use only with the RIDGE= or PCOMIT= option.
PCOMIT=	Performs incomplete principal component analysis and outputs estimates to the OUTEST= data set
PRESS	Outputs the PRESS statistic to the OUTEST= data set
RIDGE=	Performs ridge regression analysis and outputs estimates to the OUTEST= data set
RSQUARE	Same effect as the EDF option
TABLEOUT	Outputs standard errors, confidence limits, and associated test statistics of the parameter estimates to the OUTEST= data set
ODS Graphics Options	
PLOTS=	Produces ODS graphical displays
Display Options	
CORR	Displays correlation matrix for variables listed in MODEL and VAR statements
SIMPLE	Displays simple statistics for each variable listed in MODEL and VAR statements
USSCP	Displays uncorrected sums of squares and crossproducts matrix
ALL	Displays all statistics (CORR, SIMPLE, and USSCP)
NOPRINT	Suppresses output
Other Options	
ALPHA=	Sets significance value for confidence and prediction intervals and tests
SINGULAR=	Sets criterion for checking for singularity

Comments:

- The `outest` option will create a data set with information about the estimation.
- The `plots` option controls what types of plots will be created. By using `plots = none`, this will prevent all plots from being created similar to what was done earlier with ods statements.

The most important statement is `model`, which gives a syntax representation of the regression model to be estimated. Selecting `MODEL` in the help leads to further information about it:

The REG Procedure

Overview | Getting Started | Syntax | Details | Examples | References

MODEL Statement

<label:> **MODEL** *dependents = <regressors> </ options>*;

After the keyword MODEL, the dependent (response) variables are specified, followed by an equal sign and the regresso variables. Variables specified in the MODEL statement must be numeric variables in the data set being analyzed. For example, if you want to specify a quadratic term for variable *X1* in the model, you cannot use X1*X1 in the MODEL statement but must create a new variable (for example, X1SQUARE=X1*X1) in a DATA step and use this new variable in the MODEL statement. The label in the MODEL statement is optional.

Table 85.5 summarizes the *options* available in the MODEL statement. Equations for the statistics available are given in the section Model Fit and Diagnostic Statistics.

Table 85.5: MODEL Statement Options

Option	Description
Model Selection and Details of Selection	
SELECTION=	Specifies model selection method
BEST=	Specifies maximum number of subset models displayed or output to the OUTEST= data set
DETAILS	Produces summary statistics at each step
DETAILS=	Specifies the display details for FORWARD, BACKWARD, and STEPWISE methods
GROUPNAMES=	Provides names for groups of variables
INCLUDE=	Includes first n variables in the model
MAXSTEP=	Specifies maximum number of steps that might be performed
NOINT	Fits a model without the intercept term
PCOMIT=	Performs incomplete principal component analysis and outputs estimates to the OUTEST= data set
RIDGE=	Performs ridge regression analysis and outputs estimates to the OUTEST= data set
SLE=	Sets criterion for entry into model
SLS=	Sets criterion for staying in model
START=	Specifies number of variables in model to begin the comparing and switching process
STOP=	Stops selection criterion
Statistics	
ADJRSQ	Computes adjusted R square
AIC	Computes Akaike's information criterion
B	Computes parameter estimates for each model
BIC	Computes Sawa's Bayesian information criterion
CP	Computes Mallows' C_p statistic
GMSEP	Computes estimated MSE of prediction assuming multivariate normality
JP	Computes J_p, the final prediction error
MSE	Computes MSE for each model
PC	Computes Amemiya's prediction criterion
RMSE	Displays root MSE for each model
SBC	Computes the SBC statistic
SP	Computes S_p statistic for each model
SSE	Computes error sum of squares for each model
Data Set Options	
EDF	Outputs the number of regressors, the error degrees of freedom, and the model R square to the OUTEST= data set
OUTSEB	Outputs standard errors of the parameter estimates to the OUTEST= data set
OUTSTB	Outputs standardized parameter estimates to the OUTEST= data set. Use only with the RIDGE= or PCOMIT= option.
OUTVIF	Outputs the variance inflation factors to the OUTEST= data set. Use only with the RIDGE= or PCOMIT= option.
PRESS	Outputs the PRESS statistic to the OUTEST= data set
RSQUARE	Has same effect as the EDF option
Regression Calculations	

Comments:

- Different models can be "named" by specifying a label before

a `model` statement. This can be helpful when more than one model is estimated at a time within `proc reg` (I rarely do this).

- If more than one explanatory variable (i.e., regressor, independent variable, covariate) is needed for a model, then one can simply provide these after the equal sign by separating them with spaces. For example, if ACT score was an additional explanatory variable in the model, then the syntax would be `model College = HS ACT`.
- Various options to be given after the `/` are displayed in the help. For example, the `noint` option prevents β_0 from being estimated. Also, notice the "data set options" portion of the help which allows additional items to be put in the data set specified by `outest` in the `proc reg` line of code.

Below is a demonstration for some of these and additional statements and options:

```
title2 "Estimate model for GPA data";
proc reg data=set1 outest=out_set1 alpha=0.05 plots=none;
  MyModel: model College = HS / outseb clm p r;
run;

title2 "Information resulting from outest option";
proc print data=out_set1;
run;
```

Chris Bilder, STAT 850
Estimate model for GPA data

The REG Procedure
Model: MyModel
Dependent Variable: College

Number of Observations Read	20
Number of Observations Used	20

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	2.89817	2.89817	24.54	0.0001
Error	18	2.12621	0.11812		
Corrected Total	19	5.02437			

Root MSE	0.34369	R-Square	0.5768
Dependent Mean	2.86250	Adj R-Sq	0.5533
Coeff Var	12.00662		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1.08688	0.36662	2.96	0.0083
HS	1	0.61249	0.12365	4.95	0.0001

Chris Bilder, STAT 850
Estimate model for GPA data

The REG Procedure
Model: MyModel
Dependent Variable: College

Output Statistics						
Obs	Dependent Variable	Predicted Value	Std Error Mean Predict	95% CL Mean		Residual
1	3.10	2.9489	0.0788	2.7833	3.1144	0.1511
2	2.30	2.5262	0.1025	2.3108	2.7417	-0.2262
3	3.00	2.7406	0.0807	2.5711	2.9101	0.2594
4	2.45	2.6487	0.0881	2.4636	2.8339	-0.1987
5	2.50	2.8202	0.0773	2.6578	2.9827	-0.3202
6	3.70	3.7329	0.1918	3.3299	4.1358	-0.0329
7	3.40	3.1632	0.0979	2.9575	3.3690	0.2368
8	2.60	2.5079	0.1050	2.2872	2.7285	0.0921
9	2.80	2.7345	0.0811	2.5641	2.9048	0.0655
10	3.60	2.8202	0.0773	2.6578	2.9827	0.7798
11	2.00	2.5507	0.0993	2.3420	2.7594	-0.5507
12	2.90	3.3225	0.1205	3.0692	3.5757	-0.4225
13	3.30	2.8325	0.0771	2.6705	2.9944	0.4675
14	3.20	3.4327	0.1384	3.1419	3.7235	-0.2327
15	2.80	2.4466	0.1138	2.2075	2.6857	0.3534
16	2.40	2.2996	0.1372	2.0114	2.5878	0.1004
17	2.60	2.8509	0.0769	2.6893	3.0124	-0.2509
18	3.80	3.5369	0.1563	3.2084	3.8653	0.2631
19	2.20	2.4834	0.1085	2.2555	2.7112	-0.2834
20	2.60	2.8509	0.0769	2.6893	3.0124	-0.2509

Sum of Residuals	0
Sum of Squared Residuals	2.12621
Predicted Residual SS (PRESS)	2.51305

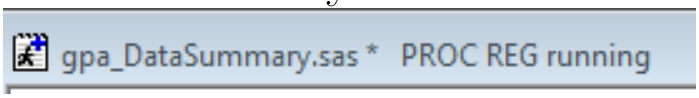
Chris Bilder, STAT 850
Information resulting from outest option

Obs	_MODEL_	_TYPE_	_DEPVAR_	_RMSE_	Intercept	HS	College
1	MyModel	PARMS	College	0.34369	1.08688	0.61249	-1
2	MyModel	SEB	College	0.34369	0.36662	0.12365	-1

Questions:

- What option led to the predicted values ($\hat{Y}$) being printed?
- What option led to the confidence intervals for $E(Y)$ being printed?
- Where is the standard error for $\hat{\beta}_1$ (i.e., $Var(\hat{\beta}_1)^{1/2}$) printed?
- If the `output` statement was included in the `proc reg` code, what do you think this would help do?

It is common to run `proc reg` multiple times back-to-back in order to investigate particular aspects of a model or to get the code/output "correct". When this is done, you will notice that SAS indicates that `proc reg` is still running despite output being generated already:



This will happen with some other SAS procedures as well. Unfortuately, any SAS data set being created in these instances cannot be viewed. To end the running of a procedure, one can issue a "`quit;`" line of code or run another procedure.

Overall, a great way to learn how to use SAS or other statistical software packages is to examine the help for a procedure to find interesting new statements or options. Put these statements/options into your own code to see what happens!

Output delivery system

The `output` statement is a traditional way to include computations performed by a procedure in a data set. Starting in SAS version 8, all aspects of the output from a procedure are considered to be in a *table* and these tables can be put into a data set. This is done through the output delivery system (ODS). Statements starting with `ods` can exist inside or outside of the `proc` and `run` code. To obtain a list of what information can be extracted from a procedure's output, use `ods trace on` before and `ods trace off` after the execution of code. Below is an example for `proc means` with the cereal data:

```
ods trace on;  *Print ODS table names in log window;

title2 "Means for cereal data";
proc means data=set1 mean;
  var sugar fat sodium;
run;

ods trace off;  *ODS names are no longer printed;
```

Rather than looking at the Results Viewer, the important information is displayed in the log window:

```
695     ods trace off;
696
697     title2 "Means for cereal data";
698     proc means data=set1 mean;
699       var sugar fat sodium;
700       ods output summary=ods_set1;
701     run;

Output Added:
-------------
Name:         Summary
Label:        Summary statistics
Template:     base.summary
Path:         Means.Summary
```

```
-------------
NOTE: There were 40 observations read from the data set
      WORK.SET1.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.10 seconds
      cpu time            0.01 seconds
702
703   ods trace off;
```

A table of information can be created by using the name `summary` in an `ods output` statement. Below is how essentially the same code is run with this new `ods` statement.

```
title2 "Means for cereal data";
proc means data=set1 mean;
  var sugar fat sodium;
  ods output summary=ods_set1;
run;

title2 "ODS generated table";
proc print data=ods_set1;
run;
```

Chris Bilder, STAT 850
Means for cereal data

The MEANS Procedure

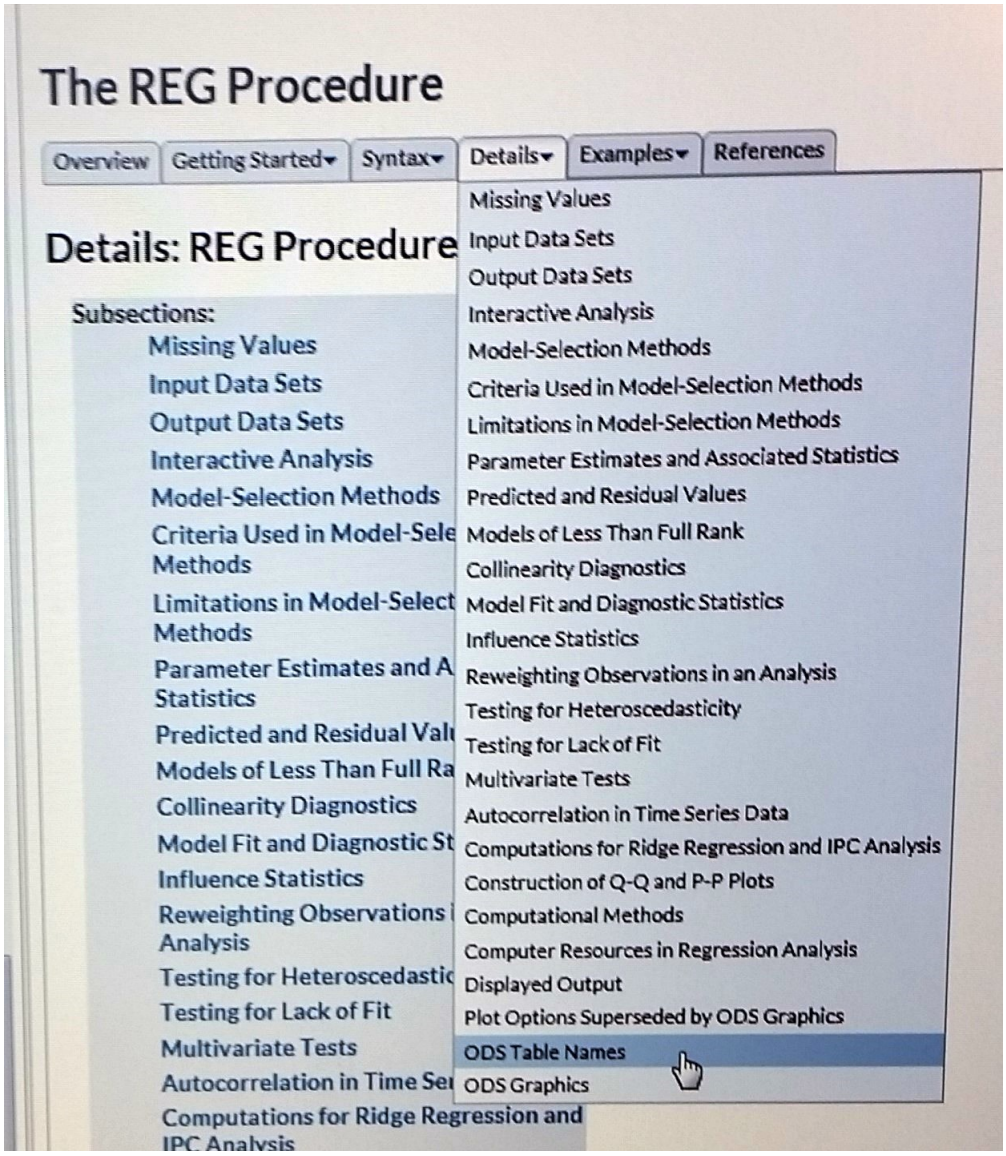
Variable	Mean
sugar	0.2894156
fat	0.0321828
sodium	5.6147038

Chris Bilder, STAT 850
ODS generated table

Obs	VName_sugar	sugar_Mean	VName_fat	fat_Mean	VName_sodium	sodium_Mean
1	sugar	0.2894155586	fat	0.0321827691	sodium	5.6147038182

How does one know what information is available in the data sets that can be created by ODS?

- Information regarding the table names is available in the help for most procedures (look under "Details"). The image below shows the help for `proc reg`:



  The ODS Graphics portion of the help describes the plots that can be produced with the `plots` option on the `proc reg` line.

- Create the data set and then print it.

- Have SAS label the output with their corresponding ODS table names. Unfortunately, SAS will not create these labels for output going to the Results Viewer. Instead, the output needs to go to the output window. Below is how this is accomplished using `proc reg` and the GPA data:

```
ods listing;  *Output also goes to Output window;
```

```
ods trace on / listing;  *Put ODS table names in output
  window;

proc reg data=set1 plots=none;
  model College = HS;
run;

ods trace off;  *ODS table names are no longer printed;
ods listing close;   *Output now only goes to Results Viewer;
```

Output - (Untitled)

Chris Bilder, STAT 850
Estimate model for GPA data

The REG Procedure
Model: MODEL1
Dependent Variable: College

Output Added:

Name: NObs
Label: Number of Observations
Template: Stat.Reg.NObs
Path: Reg.MODEL1.Fit.College.NObs

Number of Observations Read	20
Number of Observations Used	20

Output Added:

Name: ANOVA
Label: Analysis of Variance
Template: Stat.REG.ANOVA
Path: Reg.MODEL1.Fit.College.ANOVA

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value
Model	1	2.89817	2.89817	24.54
Error	18	2.12621	0.11812	
Corrected Total	19	5.02437		

Analysis of Variance

Source	Pr > F
Model	0.0001
Error	
Corrected Total	

Output Added:

Name: FitStatistics
Label: Fit Statistics
Template: Stat.REG.FitStatistics
Path: Reg.MODEL1.Fit.College.FitStatistics

Output - (Untitled)

Chris Bilder, STAT 850
Estimate model for GPA data

The REG Procedure
Model: MODEL1
Dependent Variable: College

Root MSE	0.34369	R-Square	0.5768
Dependent Mean	2.86250	Adj R-Sq	0.5533
Coeff Var	12.00662		

Output Added:

Name: ParameterEstimates
Label: Parameter Estimates
Template: Stat.REG.ParameterEstimates
Path: Reg.MODEL1.Fit.College.ParameterEstimates

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1.08688	0.36662	2.96	0.0083
HS	1	0.61249	0.12365	4.95	0.0001

Below is how one of these tables can be extracted and then printed:

```
proc reg data=set1 plots=none; *Cannot use noprint option!;
  model College = HS;
  ods output ParameterEstimates=ods_set1;
run;

title2 "ODS ParameterEstimates data set";
proc print data=ods_set1;
run;
```

Chris Bilder, STAT 850
Estimate model for GPA data

The REG Procedure
Model: MODEL1
Dependent Variable: College

Number of Observations Read	20
Number of Observations Used	20

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	2.89817	2.89817	24.54	0.0001
Error	18	2.12621	0.11812		
Corrected Total	19	5.02437			

Root MSE	0.34369	R-Square	0.5768
Dependent Mean	2.86250	Adj R-Sq	0.5533
Coeff Var	12.00662		

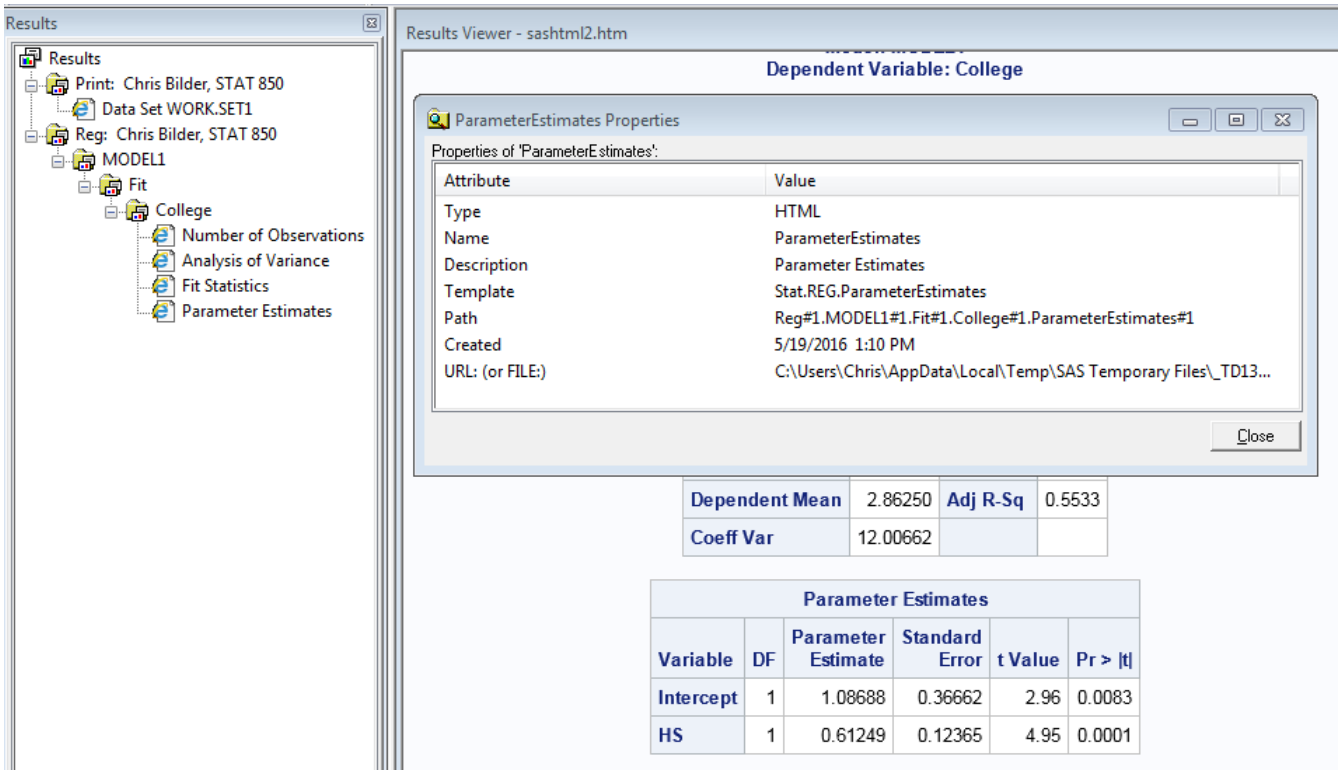
Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1.08688	0.36662	2.96	0.0083
HS	1	0.61249	0.12365	4.95	0.0001

Chris Bilder, STAT 850
ODS ParameterEstimates data set

Obs	Model	Dependent	Variable	DF	Estimate	StdErr	tValue	Probt
1	MODEL1	College	Intercept	1	1.08688	0.36662	2.96	0.0083
2	MODEL1	College	HS	1	0.61249	0.12365	4.95	0.0001

- Use the information in the Results Explorer to determine a table name. For example, I right clicked on "Parameter Estimates" and selected Properties to bring up the ParameterEstimates Properties window:



  This window also shows where SAS stores a temporary HTML file that contains the output displayed in the Results Viewer.

The ODS can only print information that is displayed in the output. Thus, one cannot use a `noprint` option in the first line of a call to a procedure and still have a data set created by the ODS. Also, this means that specific output needs to be requested in a procedure call for a particular table to be available. For example, the second `proc reg` call below generates two additional tables (see log window after running it yourself) when compared to the first `proc reg` call:

```
ods trace on;

proc reg data=set1 plots=none;
  model College = HS;
run;
```

```
proc reg data=set1 plots=none;
  model College = HS / clm p;
run;

ods trace off;
```

Contingency tables

The responses for categorical variables are often summarized as counts in a contingency table format. To illustrate the process of how to create a contingency table in SAS, we are going to examine data that I collected for Bilder and Loughin (*Chance*, 1998). The overall purpose of this paper was to determine what factors affect the probability of success for a placekick in the National Football League (NFL). The variables in the data set are:

- Week: Week of the season
- Distance: Distance of the placekick in yards
- Change: Binary variable denoting lead-change (1) versus non-lead-change (0) placekicks; successful lead-change placekicks are those that change which team is winning the game.
- Elap30: Number of minutes remaining before the end of the half with overtime placekicks receiving a value of 0
- PAT: Binary variable denoting the type of placekick where a point after touchdown (PAT) is a 1 and a field goal is a 0
- Type: Binary variable denoting dome (0) versus outdoor (1) placekicks
- Field: Binary variable denoting grass (1) versus artificial turf (0) placekicks
- Wind: Binary variable for placekicks attempted in windy conditions (1) versus non-windy conditions (0); I define windy as

a wind stronger than 15 miles per hour at kickoff in an outdoor stadium

- Good: This is the response (dependent) variable; it is a 1 for successful placekicks and a 0 for failed placekicks.

There are 1,425 placekick observations from the 1995 NFL season that are within this data set.

For the purpose here, we will first examine a contingency table summarizing the number of placekicks (counts) that are cross-classified by the good and change variables:

```
title1 "Chris Bilder, STAT 850";

proc import out=placekick datafile="C:\data\placekick.csv"
                DBMS=CSV replace;
  getnames=yes;
  datarow=2;
run;

title2 "The placekicking data set";
proc print data=placekick(obs=5);
run;

title2 "Contingency table for good vs. change";
proc freq data=placekick;
  tables good*change;
run;

title2 "Contingency table for good vs. change";
proc freq data=placekick;
  tables good*change / norow nocol nocum nopercent;
run;
```

Chris Bilder, STAT 850
The placekicking data set

Obs	week	distance	change	elap30	PAT	type	field	wind	good
1	1	21	1	24.7167	0	1	1	0	1
2	1	21	0	15.85	0	1	1	0	1
3	1	20	0	0.45	1	1	1	0	1
4	1	28	0	13.55	0	1	1	0	1
5	1	20	0	21.8667	1	0	0	0	1

Chris Bilder, STAT 850
Contingency table for good vs. change

The FREQ Procedure

Frequency Percent Row Pct Col Pct	Table of change by good		
	good		
change	0	1	Total
0	95 6.67 8.91 58.28	971 68.14 91.09 76.94	1066 74.81
1	68 4.77 18.94 41.72	291 20.42 81.06 23.06	359 25.19
Total	163 11.44	1262 88.56	1425 100.00

Chris Bilder, STAT 850
Contingency table for good vs. change

The FREQ Procedure

Frequency	Table of change by good		
	good		
change	0	1	Total
0	95	971	1066
1	68	291	359
Total	163	1262	1425

The procedure which creates the contingency tables is `proc freq` (SAS STAT). By using a few options in the `tables` statement, I was able to remove some of the distracting information that appeared in the output from the first procedure call.

The syntax for `proc freq` as displayed in the help is

Syntax: FREQ Procedure

The following statements are available in the FREQ procedure:

PROC FREQ < *options* > ;
BY *variables*;
EXACT *statistic-options* < / *computation-options* > ;
OUTPUT <OUT=*SAS-data-set* > *output-options*;
TABLES *requests* < / *options* > ;
TEST *options*;
WEIGHT *variable* < / *option* > ;

Some of the same types of statements and the same syntax structure that we have seen before is present again. Differences include the `tables` statement which replaces the `model` or `var` statements that we have seen previously. Simply, this procedure produces contingency tables so this is why "tables" is used instead.

Questions:

- How can a data set be created which contains the counts from the contingency table?
- Pearson chi-square tests for independence are often performed for contingency tables like this. How can we have SAS compute the necessary information for this test with `proc freq`?

An alternative way to summarize this cross-classification of the data is through using the `list` option in the `tables` statement:

```
title2 "Cross-classifications of good vs. change";
proc freq data=placekick;
  tables change*good / norow nocol nocum nopercent list;
run;
```

Chris Bilder, STAT 850
Cross-classifications of good vs. change

The FREQ Procedure

change	good	Frequency
0	0	95
0	1	971
1	0	68
1	1	291

This display format can be especially helpful when there is a categorical variable with many levels and/or more than two categorical variables.

Final comment

When running a long program, there may be times when you would like to skip portions of it. One way is to simply put an asterisk in front of each line to "comment over" the code that you do not want to run. A more simple way is to enclose the

corresponding code by `/*` at the start and `*/` at the end of the code that you would like to skip.