Statistical Computing with SAS

P6110: Lecture Notes

Jihui Lee

Department of Biostatistics

Columbia University

COLUMBIA UNIVERSITY | MAILMAN SCHOOL of PUBLIC HEALTH BIOSTATISTICS

Chapter 1. A Dive into SAS

1.1. Statistical Analysis System (SAS)

- Initially started as an agricultural research project at North Carolina State University in 1966.
- Primarily leased to other agricultural departments to analyze the effect soil, weather and seed varieties had on crop yields in the early 1970s.
- SAS Institute Inc. was founded as a private company in 1976.
- "Best Company to Work For" in Fortune's annual rankings each year since 1997.
- Used at more than 75,000 sites in 147 countries.
- Some components
  - Base SAS: Basic procedures and data management
  - SAS/STAT: Statistical analysis
  - SAS/GRAPH: Graphics and presentation
  - SAS/OR: Operations research

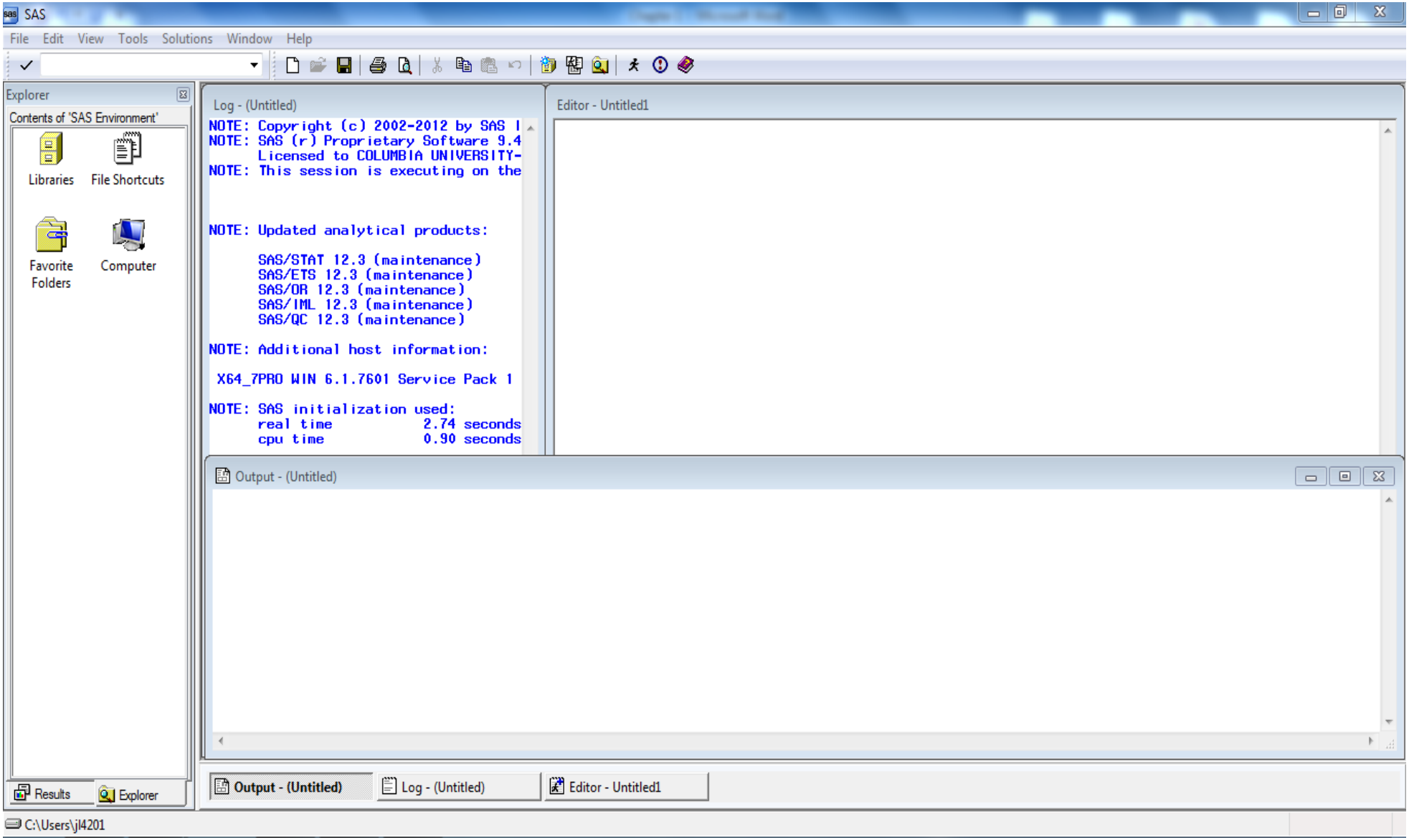
- SAS/ETS: Econometrics and time series analysis
- SAS/IML: Interactive matrix language
- SAS/QC: Quality control
- SAS/INSIGHT: Data mining

1.2. Why SAS?

- History: Long history, wide range of procedures
- Popularity: Useful in job market
- Reliability: Quality control and customer service by experts
- Big data: Great for (large) data manipulation
- Documentation: Neat and well-structured results
- Help: Detailed help documentations and web pages

1.3. SAS Windows

- Editor: Write, edit, and submit SAS programs
- Log: Notes about the SAS session including errors and warnings related to the submitted SAS programs
- Output: Any printable results
  - HTML (Result Viewer; Default for SAS 9.3 or later) vs listing
  - Tools → Options → Preferences → Results → Select 'Create listing'
- Results: Tree list of contents for the Output window
- Explorer: Access to SAS data files and libraries


SAS
File Edit View Tools Solutions Window Help
Explorer
Contents of 'SAS Environment'
Libraries
File Shortcuts
Favorite Folders
Computer
Results
Explorer
Log - (Untitled)
NOTE: Copyright (c) 2002-2012 by SAS I
NOTE: SAS (r) Proprietary Software 9.4
Licensed to COLUMBIA UNIVERSITY-
NOTE: This session is executing on the
NOTE: Updated analytical products:
SAS/STAT 12.3 (maintenance)
SAS/ETS 12.3 (maintenance)
SAS/OR 12.3 (maintenance)
SAS/IML 12.3 (maintenance)
SAS/QC 12.3 (maintenance)
NOTE: Additional host information:
X64_7PRO WIN 6.1.7601 Service Pack 1
NOTE: SAS initialization used:
real time 2.74 seconds
cpu time 0.90 seconds
Editor - Untitled1
Output - (Untitled)
Output - (Untitled)
Log - (Untitled)
Editor - Untitled1
C:\Users\jl4201

1.4. Layout of SAS Programs

Step	Role	Example

DATA step	Read, create, and manipulate data	`data Distance;`
		`  Miles = 26.22;`
		`  Kilometers = 1.61 * Miles;`
		`run;`

PROC step	Perform analysis and generate output	`proc print data = Distance; run;`

- Every statement ends with a semicolon (;).
- A statement can be more than one line.
- A statement can be on the same line as other statements.
- A statement can start in any column.
- NOT case sensitive! (except “quoted” strings)
- Color-coded!
- Check SAS log for important notes.

Example

Editor

```
* Create a SAS dataset named 'distance';
data Distance;
     Miles = 26.22;
     Kilometers = 1.61 * Miles; * Convert miles into kilometers;
run;

* Print the results;
proc print data= Distance;
run;
```

Log

```
NOTE: Copyright (c) 2002-2012 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.4 (TS1M0)
      Licensed to COLUMBIA UNIVERSITY-HEALTH SCIENCES CAMPUS-T&R SFA, Site 70080790.
NOTE: This session is executing on the X64_7PRO platform.
NOTE: Additional host information:
 X64_7PRO WIN 6.1.7601 Service Pack 1 Workstation
NOTE: SAS initialization used:
      real time           0.57 seconds
      cpu time            0.51 seconds

1    * Create a SAS dataset named 'distance';
2    data Distance;
3        Miles = 26.22;
4        Kilometers = 1.61 * Miles;
5    run;
NOTE: The data set WORK.DISTANCE has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds

6
7    * Print the results;
8    proc print data= Distance;
NOTE: Writing HTML Body file: sashtml.htm
9    run;
NOTE: There were 1 observations read from the data set WORK.DISTANCE.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.60 seconds
      cpu time            0.54 seconds
```

Output

Result Viewer

Obs	Miles	Kilometers
1	26.22	42.2142

Listing

Obs	Miles	Kilometers
1	26.22	42.2142

Example

SAS Code

```
data Exam;
     input Name $ Exam1 Exam2 Exam3;
     cards;
     Emma 95 75 85
     Noah 89 . 99
     Liam 88 98 78
     Olivia . 70 80
     ;
run;
proc print data=Exam; run;
```

Output

Obs	Name	Exam1	Exam2	Exam3
1	Emma	95	75	85
2	Noah	89	.	99
3	Liam	88	98	78
4	Olivia	.	70	80

SAS Code

```
proc print data=Exam (obs=3);
     title 'Exam Dataset';
run;
```

Output

Exam Dataset

Obs	Name	Exam1	Exam2	Exam3
1	Emma	95	75	85
2	Noah	89	.	99
3	Liam	88	98	78

SAS Code

```
proc print data=Exam noobs;
     var name exam2;
run;
```

Output

Name	Exam2
Emma	75
Noah	.
Liam	98
Olivia	70

```
proc print data=Exam
style(data)={background=yellow};
     var name / style(data)={font_style=italic
font_weight=bold};
     var exam1-exam2;
run;
```

Obs	Name	Exam1	Exam2
1	*Emma*	95	75
2	*Noah*	89	.
3	*Liam*	88	98
4	*Olivia*	.	70

1.5. SAS Statements: Rules

- SAS variable/dataset names
  - Must be 32 characters or fewer in length.
  - Contain letters, numbers, and underscores (_).
  - Start with a letter or an underscore.
  - If no name is given for a dataset, SAS creates default names of the form 'data*n*' where *n* is an integer. (e.g. data1, data2, …)
- Missing values are represented by a period (.).
- Comments can be added in two ways: `* Comment;` or `/* Comment */`

1.6. Useful Tips

- Include a header for every program: Title, purpose, author, date
- Name your dataset and variables in a concise yet informative way.
- Select the part of programs you want to run before clicking the running button. Otherwise, SAS runs the whole program.
- Write neat and straightforward programs.
  - One SAS statement on one line
  - Include detailed comments: Easy to understand the logistics of SAS codes.
  - Avoid too many loops. (e.g. DO, IF-ELSE)
- Test each part of your program: Check datasets, output, and *log*.
- Macros can be useful when some codes should be repeatedly executed.
- Take advantage of HELP menu and worldwide network of SAS users.

Chapter 2. Data Structure

Creating a SAS dataset can be done by

- ❖ Entering the data directly on the SAS editor
- ❖ Creating SAS datasets from raw data files (e.g. .txt, .csv, .xlsx, .dat, .sas7bdat)
- ❖ Converting data files from other software into SAS datasets

2.1. Library

Example

```
libname P6110 "C:\Users\jl4201\Desktop\P6110\SAS";
```

- SAS Library: Location where (SAS or other types) datasets are stored

  - A folder or directory on the computer
  - Flash drive or CD

- Simply make up a name for a library and tell SAS where it is.
  - LIBNAME
  - Tools → New Library → Specify the path*
  - Explorer → Libraries → (Right click) New → Specify the path*

* Check 'Enable at startup' box to avoid defining the library reference every time you start up SAS.

- Check if the library is successfully specified: Explorer → Libraries
- Datasets are saved as 'dataset-name.sas7bdat'.

2.2. SAS Dataset: Enter directly

Example

```
data P6110.Exam1;
     input Name $ Exam1-Exam3 @@;
     * 'cards' or 'datalines';
     cards;
     Emma 95 75 85 Noah 89 . 99
     Liam 88 98 78 Olivia . 70 80
     ;
run;

proc print data=P6110.Exam1;
title 'Exam Dataset';
run;
```

Exam Dataset

Obs	Name	Exam1	Exam2	Exam3
1	Emma	95	75	85
2	Noah	89	.	99
3	Liam	88	98	78
4	Olivia	.	70	80

- DATA: Name the dataset.
- INPUT: List variable names.
  - List (free): Data separated by at least one blank
  - Column: Data arranged in columns

  - Formatted input: Data in nonstandard formats

- CARDS (DATALINES): List data.
- RUN: Tell SAS to execute the block of code after the DATA statement.

2.3. SAS Dataset: Importing raw data files

Example

Raw Data

	A	B	C	D
1	Name	Exam1	Exam2	Exam3
2	Emma	95	75	85
3	Noah	89		99
4	Liam	88	98	78
5	Olivia		70	80

Output

Obs	Name	Exam1	Exam2	Exam3
1	Emma	95	75	85
2	Noah	89	.	99
3	Liam	88	98	78
4	Olivia	.	70	80

SAS Code

```
* xlsx;
proc import out=P6110.Exam2
     datafile="C:\Users\jl4201\Desktop\
P6110\SAS\Chapter 2\Exam.xlsx"
     dbms=xlsx replace;
     sheet="Sheet1";
     getnames=yes;
run;

* txt;
proc import out=P6110.Exam3
     datafile="C:\Users\jl4201\Desktop\
P6110\SAS\Chapter 2\Exam.txt"
```

```
* csv;
proc import out=P6110.Exam4
     datafile="C:\Users\jl4201\Desktop\
P6110\SAS\Chapter 2\Exam.csv"
     dbms=csv replace;
     getnames=yes;
run;


* sas7bdat;
data P6110.Exam5;
```

```
        dbms=tab replace;
        getnames=yes;
run;
```

```
set
"C:\Users\jl4201\Desktop\P6110\SAS\Chap
ter 2\Exam.sas7bdat";
run;
```

- Import/Export Wizard: File → Import/Export Data (Not recommended)
- DATA Step: INFILE Options
  - DLM= (DELIMITER=): Specify which delimiter is used. Default is a blank space.

    (e.g. ‘,’, ‘/’, ‘&’, ‘09’X)
  - DSD: Comma-separated values (CSV) files
- PROC IMPORT Options
  - DMBS= : Specify the file extension (e.g. CSV, TAB, DLM, XLSX)
  - REPLACE: Overwrite an existing dataset named in the OUT= option if it already exists.
  - DELIMITER=: Specify which delimiter is used. Default is a blank space.

    (e.g. ‘,’, ‘/’, ‘&’, ‘09’X)

  - GETNAMES=NO: Do not get variable names from the first line of input file. Default is YES. If NO, the variables are named VAR1, VAR2, …
  - DATAROWS=*n*: Start reading data in row *n*. Default is 1.
  - SHEET=: Specify which sheet to read in the file.

- http://r4stats.com/examples/data-import/

2.4. Modifiers and Pointers (DATA step)

- &: Use two whitespaces characters to signal the end of a character variable.
- @: Hold the line to allow further input statements in the iteration of the data step.
- @@: Hold the line to allow continued reading from the line on subsequent iteration of the data step. (Multiple observations per line)
- @*n*: Move the pointer to column *n*.
- /: Skip to the next line of raw data.
- #*n*: Move the pointer to the *n*-th line for each observation.

- @'character': Useful when an observation always comes after a particular character or a word.
- +*n*: Move the pointer to the right *n* columns.

2.5. Input Options

- FIRSTOBS=*n*: Tell SAS at what line to begin reading data.
- OBS=*n*: Tell SAS to stop reading after *n* data lines.
- MISSOVER: If it runs out of data, instead of going to the next line, assign missing values to any remaining variables.

- TRUNCOVER: Useful when reading data using column or formatted input and some data lines are shorter than others. TRUNCOVER takes as much as is there when the data line ends in the middle of a variable field.

2.6. Formatted Input

Example

```
data Score;
     input Name $16. +1 Age 2. +1 Type $1. +1 Date MMDDYY9. Score1-Score5;
     datalines;
Alicia Grossman  13 c 10-28-19 7.8 6.5 7.2 8.0 7.9
Matthew Lee       9 D 10-30-19 6.5 5.9 6.8 6.0 8.1
Elizabeth Garcia 10 C 10-29-19 8.9 7.9 8.5 9.0 8.8
Lori Newcombe     6 D 10-30-19 6.7 5.6 4.9 5.2 6.1
```

```
Brian Williams   11 C 10-29-19 7.8 8.4 8.5 7.9 8.0
;
run;

* SAS date values are the number of days since January 1, 1960.;
* Time values are the number of seconds past midnight, and
* daytime values are the number of seconds past midnight January 1, 1960.;

proc print data=P6110.Score;
     format Date MMDDYY9.; run;
```

Obs	Name	Age	Type	Date	Score1	Score2	Score3	Score4	Score5
1	Alicia Grossman	13	c	10/28/19	7.8	6.5	7.2	8.0	7.9
2	Matthew Lee	9	D	10/30/19	6.5	5.9	6.8	6.0	8.1
3	Elizabeth Garcia	10	C	10/29/19	8.9	7.9	8.5	9.0	8.8
4	Lori Newcombe	6	D	10/30/19	6.7	5.6	4.9	5.2	6.1
5	Brian Williams	11	C	10/29/19	7.8	8.4	8.5	7.9	8.0

- Data not in standard format can be such as
  - Numbers with commas
  - Numbers that contain dollar sign
  - Dates / Times of day
- Each variable is followed by its input format, referred as ‘informat’.

- https://support.sas.com/documentation/cdl/en/leforinforref/63324/HTML/default/viewer.htm#n0verk17pchh4vn1akrrv0b5w3r0.htm

Example

	Informat	Definition
Character	$*w*.	Specify the width of the variable
$**INFORMAT**w.	$QUOTE*w*.	Remove matching quotation marks
	$UPCASE*w*.	Convert character data to uppercase
Numeric	*w.d*	Specify the width *w* and the number of decimal places *d*
INFORMATw.d	COMMA*w.d*	Remove embedded commas and $
	PERCENT*w.d*	Convert percentages to numeric values
Date/Time	DATE*w*.	Read dates in the form: *ddmmmyy* or *ddmmmyyyy*
INFORMATw.	MMDDYY*w*.	Read dates in the form: *mmddyy* or *mmddyyyy*
	TIME*w*.	Read time in form: *hh:mm:ss.ss* or *hh:mm*
	DATETIME*w*.	Read datetime values in the form: *ddmmmyy hh:mm:ss.ss*

Chapter 3. Data Manipulation

3.1. Sort Datasets: PROC SORT

- A dataset can be sorted by one or more variables.
- Overwrite the existing dataset unless using out= option.
- By default, sort in ascending order.
- Sort with respect to the order of variable list.

General Syntax

```
proc sort data=dataset out=new-data;
     * var1: ascending, var2: descending;
     by var1 descending var2;
run;
```

3.2. Subset Datasets: IF or WHERE

- Select observations from one dataset by defining selection criteria.

General Syntax

```
data new-dataset;
     set dataset;
     where condition;
     if condition;
run;
```

3.3. IF-THEN/ELSE Statement

- Useful when grouping observations based on multiple conditions

General Syntax

```
* Multiple criteria;
if condition then action1;
else if condition then action2;
else action3;

* DO & END: Execute multiple actions;
if condition then do;
action1; action2; action3;
end;
```

Example

Raw Data	SAS Code

Data1

Obs	ID	TREAT	INITWT	WT3MOS	AGE
1	1	Other1	166.28	146.98	35
2	2	Other2	214.42	210.22	30
3	3	Other2	172.46	159.42	33
4	5	Other2	175.41	160.66	30
5	6	Other2	173.13	169.40	20
6	7	Other1	181.25	170.94	30
7	10	Other1	239.83	214.48	48
8	11	Other1	175.32	162.66	51
9	12	Other2	227.01	211.06	29
10	13	Other2	274.82	251.82	31

```
proc sort data=data1;
     by ID;
run;

proc sort data=data1 out=data1_sort;
     by age descending initwt;
run;

data subset1;
     set data1;
     where TREAT = "Other1";
run;

data subset2;
     set data1;
     if AGE > 30;
run;

data subset3;
     set data1;
     if AGE <= 30 then delete;
run;

data ifelse;
     set data1;
     length agegroup $5.;
     if age >= 50 then agegroup = "50+";
     else if age >= 30 & age < 50 then agegroup = "30-50";
     else agegroup = "-30";
run;
```

3.4. Combine Datasets

- SET statement
  - Concatenate (stack) datasets.
  - If one of the datasets has a variable not contained in the other, missing values will be added instead.
  - Add BY statement after sorting datasets to interleave datasets.

General Syntax

```
data new-dataset;
     set dataset1 ... datasetn;
run;
```

- PROC APPEND

- Useful when the two datasets contain exactly same variables (If not, ERROR).

General Syntax

```
proc append base=dataset1 data=dataset2;
run;
```

- MERGE statement
  - Useful when combining datasets from different sources
  - All datasets must be *sorted* first by the matching variables.
  - If you merge two datasets that have other variables in common, then the variables from the second dataset will overwrite the variables with the same name in the first dataset.
  - One-to-one: Only one observation for each value of the BY variable in all datasets.
  - One-to-many: One dataset has one observation for each value of the BY variable, while the other has multiple observations.
  - Many-to-many: More than one observation with a given BY variable in each dataset.

General Syntax

```
proc sort data=dataset1;
     by ID-Variable; run;
...
proc sort data=datasetn;
     by ID-Variable; run;

data new-dataset;
     merge dataset1 ... datasetn;
     by ID-Variable;
run;
```

- Divide a dataset into multiple datasets

General Syntax

```
data new-dataset1 new-dataset2; * Create 2 datasets;
     set from-dataset;
     if condition then output new-dataset1;
     else output new-dataset2;
run;
```

- MERGE (IN= Option)
  - Helpful to know which dataset an observation comes from
  - Create an indicator variable (0 / 1) that indicates whether the current observation comes from the input dataset or not.
  - Make sure that only complete records are collected in one dataset, and create another dataset with partially missing observations.

General Syntax

```
data compete missing;
merge dataset1(in=in1) dataset2(in=in2);
by id-variable;
if in1 and in2 then output complete; * Check for complete observations;
else output missing;
run;
```

Example

Raw Data

```
data data1;
input ID TREAT $
      INITWT WT3MOS AGE;
cards;
1 Other1 166.28 146.98 35
2 Other2 214.42 210.22 30
3 Other2 172.46 159.42 33
5 Other2 175.41 160.66 30
6 Other2 173.13 169.40 20
7 Other1 181.25 170.94 30
10 Other1 239.83 214.48 48
11 Other1 175.32 162.66 51
12 Other2 227.01 211.06 29
13 Other2 274.82 251.82 31
;
run;
```

```
data data2;
input ID TREAT $ INITWT
      WT3MOS AGE;
cards;
14 Surgery 203.60 169.78 38
17 Surgery 171.52 150.33 42
18 Surgery 207.46 155.22 41
;
run;
```

```
data data3;
input ID GENDER $ AREA
      $ @@;
cards;
1 F NY 1 F NJ
6 F CA 8 M PA
11 M CT 12 M AZ
14 F GA 16 M IL
17 M NC 18 F OH
;
run;
```

SAS Code

```
* Case1) Stacking;
data set1;
     set data1 data2;
*    by ID;
run;

* Case2) PROC APPEND;
proc append base=data1 data=data2;
run;

* Case3) Merging;
data merge1;
     merge set1 data3;
     by ID;
run;
```

```
data complete missing;
     merge set1(in=in1) data3(in=in2);
     by id;
     if in1 and in2 then output complete;
     else output missing;
run;

data heavy light;
     set set1;
     if initwt > 180 then output heavy;
     else output light;
run;
```

3.5. Operators in SAS

Operator	Definition	Operator		Definition
		Symbolic	Mnemonic	
*	Multiplication	=	EQ	Equal to
+	Addition	^=	NE	Not equal to
-	Subtraction	>	GT	Greater than
**	Exponentiation	>=	GE	Greater than or equal to
/	Division	<	LT	Less than
		<=	LE	Less than or equal to
			IN	Equal to one of the list
		&	AND	All comparisons must be true.
		|, ¦, !	OR	At least one comparison must be true.

3.6. Modify, Delete and Rename Variables

- The assignment statement can be used to create/modify/delete variables in the DATA step.
- KEEP = list-of-variables: Tell SAS which variables to keep.
- DROP = list-of-variables: Tell SAS which variables to drop.
- RENAME (old-var = new-var): Tell SAS to rename certain variables.

General Syntax

```
* Case 1) DATA step: RENAME, DROP, KEEP statements;
data new-dataset;
     set dataset;
     rename old-var=new-var;
     drop list-of-variables;
     keep list-of-variables;
run;

* Case 2) DATA step: Either next to dataset name or on SET statement;
data new-dataset (keep=list-of-variables drop=list-of-variables rename=(var=new-var);
     set dataset (keep=list-of-variables drop=list-of-variables rename=(var=new-var);
run;

* Case 3) PROC step;
proc print data=dataset (keep=list-of-variables drop=list-of-variables
                         rename=(var=new-var));
run;
```

Example

Raw Data

Obs	ID	TREAT	INITWT	WT3MOS	AGE
1	1	Other1	166.28	146.98	35
2	2	Other2	214.42	210.22	30
3	3	Other2	172.46	159.42	33
4	5	Other2	175.41	160.66	30
5	6	Other2	173.13	169.40	20
6	7	Other1	181.25	170.94	30
7	10	Other1	239.83	214.48	48
8	11	Other1	175.32	162.66	51
9	12	Other2	227.01	211.06	29
10	13	Other2	274.82	251.82	31
11	14	Surgery	171.52	150.33	42
12	17	Surgery	203.60	169.78	38
13	18	Surgery	207.46	155.22	41

SAS Code

```
data data4;
    set set1;
    rename INITWT = InitialWeight
           WT3MOS = Weight3Months;

    * Define new variables;
    WTdiff = WT3MOS - INITWT;
    INITWT_kg = INITWT * 0.4536;
    INITWT_kg2 = INITWT / 2.2046;

    length agegroup $10.;
    if age >= 50 then agegroup="50+";
    else if age >= 30 then agegroup="30-50";
    else agegroup="-30";
```

```
        if agegroup in ("50+", "-30") then delete;

        drop agegroup;
        keep ID INITWT WT3MOS WTdiff INITWT_kg INITWT_kg2 Age;
        format INITWT_kg 6.2 INITWT_kg2 8.4;

run;
```

Output

Obs	ID	InitialWeight	Weight3Months	AGE	WTdiff	INITWT_kg	INITWT_kg2
1	1	166.28	146.98	35	-19.30	75.42	75.4241
2	2	214.42	210.22	30	-4.20	97.26	97.2603
3	3	172.46	159.42	33	-13.04	78.23	78.2273
4	5	175.41	160.66	30	-14.75	79.57	79.5655
5	7	181.25	170.94	30	-10.31	82.22	82.2145
6	10	239.83	214.48	48	-25.35	108.79	108.7862
7	13	274.82	251.82	31	-23.00	124.66	124.6575
8	14	171.52	150.33	42	-21.19	77.80	77.8010
9	17	203.60	169.78	38	-33.82	92.35	92.3524
10	18	207.46	155.22	41	-52.24	94.10	94.1032

3.7. Labels

- Make the output more readable and informative.
- How the *variables* appear changes, not the variable names.
- (DATA step) LABEL statement: Labels remain associated with the respective variables.
- (PROC step) LABEL statement: Only used for that procedure

3.8. Formats

- Specify how we want the data *values* to look.
- Use either 1) SAS built-in formats or 2) user-defined formats
- FORMAT statement specified in a DATA step sets the variable format *permanently*.
- FORMAT statement specified in a PROC is only used in that *specific procedure*.
- PROC FORMAT: Define your own formats.
- "format-name" is the name of the format that is used in a FORMAT statement.
- Formats for character start with a $.

- NO semicolon (;) in the VALUE statement until you have covered all possible values.
- Regrouping values using FORMAT: Specify range of values
- For non-integer values, make sure there are no cracks in your ranges.
- For convenience, you can specify user-defined permanent formats under your library.

General Syntax

```
data new-dataset;
     set dataset;
     label var-name-1 = "Label-1"
           ...
           var-name-k = "Label-k";
run;

proc format;
     value format-name  category-1 = "Formatted text-1"
                        ...
                        category-k = "Formatted text-k";
run;

proc procedurename data=dataset;
     format var-name format-name.;
run;
```

Example

SAS Code

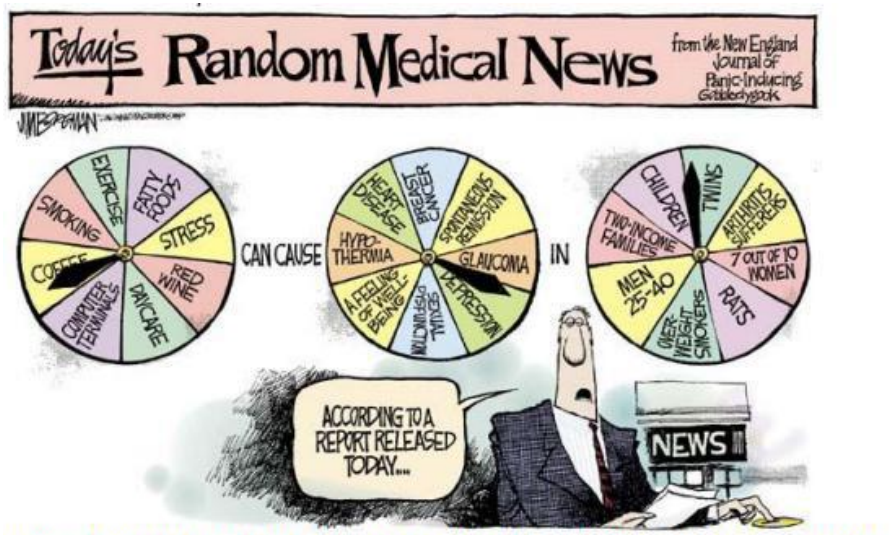
```
/* Labeling */
data label;
set set1;
label ID = "Patient ID"
      TREAT = "Treatment"
      INITWT = "Initial Weight"
      WT3MOS = "Weight after 3 Months"
      AGE = "Age";
run;

/* Formatting */
proc format;
value agegroup  0-<30 = "Less than 30"
               30-<50 = "Between 30 and 50"
               50- HIGH = "Greater than or
                           equal to 50";
value $treatment "Surgery" = "Surgical Treatment"
                 "Other1" = "Other Treatment"
                 "Other2" = "Other Treatment";
* $ for character variable;
run;

proc print data=set1;
     format age agegroup. treat $treatment.;
run;
```

Output

Obs	ID	TREAT	INITWT	WT3MOS	AGE
1	1	Other Treatment	166.28	146.98	Between 30 and 50
2	2	Other Treatment	214.42	210.22	Greater than or equal to 50
3	3	Other Treatment	172.46	159.42	Between 30 and 50
4	5	Other Treatment	175.41	160.66	Between 30 and 50
5	6	Other Treatment	173.13	169.40	Less than 30
6	7	Other Treatment	181.25	170.94	Between 30 and 50
7	10	Other Treatment	239.83	214.48	Between 30 and 50
8	11	Other Treatment	175.32	162.66	Greater than or equal to 50
9	12	Other Treatment	227.01	211.06	Less than 30
10	13	Other Treatment	274.82	251.82	Between 30 and 50
11	14	Surgical Treatment	203.60	169.78	Between 30 and 50
12	17	Surgical Treatment	171.52	150.33	Between 30 and 50
13	18	Surgical Treatment	207.46	155.22	Between 30 and 50

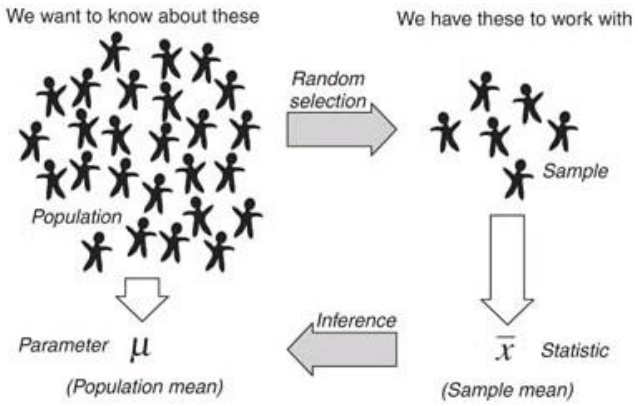
Chapter 4. Descriptive Statistics

4.1. What is Statistics?

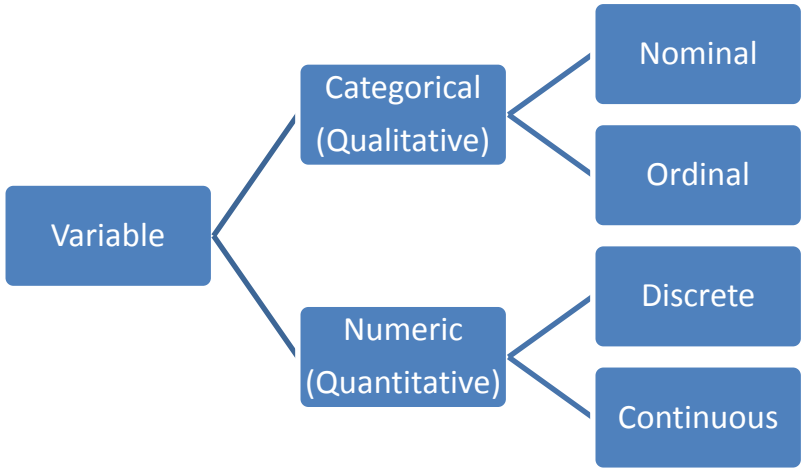


Cartoon by Jim Borgman, first published by the Cincinnati Inquirer and King Features Syndicate 1997 Apr 27; Forum section: 1
Reprinted in the New York Times, 27 April 1997, E4.

4.2. Population (Parameter) and Sample (Statistic)



4.3. Descriptive Statistics



- Distribution of a variable tells us what values it takes and how often it takes these values.
- Tabular description: Frequency table (one-way, two-way, …)
- Graphical description
  - Stem-and-leaf plot
  - Dot plot
  - Bar graph (Categorical)
  - Histogram (Continuous)
  - Box plot (cf. 5-number summary)
  - Scatterplot
- Measure of location
  - Mean
  - Median
  - Mode
- Measure of dispersion
  - Range
  - Quantile (Percentile)
  - 5-number summary (Min, Q1, Median, Q3, Maximum), Interquartile range (IQR)
  - Variance / Standard deviation
  - Coefficient of variation (CV)

4.4. Summarize Categorical Variables: PROC FREQ

- Count frequencies of both *character* and *numeric* variables in one-, two-, …, *n*-way tables.
- For n-way contingency table, separate each name with '*' in TABLES statement.
- Create output datasets containing counts and percentages.
- Compute various statistics such as chi-squared test, Fisher's exact test and odds ratio.
- The first listed variable forms the rows of the table, and the second forms the columns.
- The third variable creates multiple tables (stratification).

  e.g. var1 * var2 * var3: Create tables of var2 (row) and var3 (col) for each level of the var1

General Syntax

```
proc freq data=dataset;
     tables variable-combinations / <options>;
     * e.g. var1 var1*var2 var1*var2*var3 ...;
run;
```

- PROC FREQ options (Appear after a slash in the TABLES)

Option	Description
LIST	Print cross-tabulations in list format rather than grid
MISSPRINT	Include missing values in frequencies but not in percentages
MISSING	Include missing values in frequencies and percentages
NOCOL	Suppress printing of column percentage in cross-tabulations
NOROW	Suppress printing of row percentage in cross-tabulations
NOPERCENT	Suppress printing of global percentages
OUT = `out-dataset`	Write a dataset containing frequencies

4.5. Summarize Continuous Variables: PROC MEANS

- Primarily used for reporting various summary statistics of *numeric* variables.
- Without options, it will calculate the summary statistics for all numeric variables.

  (Default statistics: N (number of non-missing obs), Mean, Standard deviation, Min and Max)

General Syntax

```
proc means data=dataset;
     by list-of-variables;
     class list-of-variables;
     var list-of-variables;
     output out=out-dataset;
run;
```

- PROC MEANS options

Option	Description	Option	Description
MAX	Maximum value	N	Number of non-missings
MIN	Minimum value	NMISS	Number of missings
MEAN	Mean	RANGE	Range
MEDIAN	Median	STDDEV	Standard deviation
MODE	Mode	SUM	Sum
MAXDEC=*n*	Number of decimal places to be displayed	MISSING	Treat missing values as valid summary groups.
P20	20% quantile	NOPRINT	Do not print the means result.

- Optional statements

Option	Description
BY `list-of-variables`	Perform separate analyses for each level of the variables in the list. The dataset must first be *sorted* by these variables.
CLASS `list-of-variables`	Perform the same thing as BY statement, but the output is more compact. No sorting needed.
VAR `list-of-variables`	Specify which numeric variables to use in the analysis. If not specified, then SAS uses all numeric variables.

4.6. Examine Distribution of Continuous Variables: PROC UNIVARIATE

- Explore a dataset *before* conducting any statistical test.
- Produce statistics and graphs describing the distribution of a single variable.

  (e.g. mean, median, mode, standard deviation, skewness, kurtosis[1])
- Good for checking distributional assumptions (Normality).
- Without VAR statement, SAS will calculate statistics for all numeric variables in the dataset.

General Syntax

```
proc univariate data=dataset;
     var list-of-variables;
run;
```

[1] Skewness indicates how asymmetrical the distribution is; Kurtosis indicates how flat or peaked the distribution is.

Example

Raw Data

Obs	pregnant	blood	insulin	bmi	pedigree	age	test	BMIlevel
1	6	72	.	33.6	0.627	50	Positive	Obese
2	1	66	.	26.6	0.351	31	Negative	Overweight
3	8	64	.	23.3	0.672	32	Positive	Healthy
4	1	66	94	28.1	0.167	21	Negative	Overweight
5	0	40	168	43.1	2.288	33	Positive	Obese

SAS Code

```
proc freq data=pima;
     tables BMIlevel * test /
nocol missing out=freqout;
run;
```

```
proc means data=pima n nmiss mean std range;
     class test;
     var insulin blood bmi age;
     output out=meansout1;
run;

proc univariate data=pima normal;
     var insulin blood;
run;
```

Output

Frequency
Percent
Row Pct

Table of BMIlevel by test

BMIlevel	test(test) Negative	Positive	Total
	9 1.17 81.82	2 0.26 18.18	11 1.43
Healthy	95 12.37 93.14	7 0.91 6.86	102 13.28
Obese	253 32.94 53.60	219 28.52 46.40	472 61.46
Overweight	139 18.10 77.65	40 5.21 22.35	179 23.31
Underweight	4 0.52 100.00	0 0.00 0.00	4 0.52
Total	500 65.10	268 34.90	768 100.00

test	N Obs	Variable	Label	N	N Miss	Mean	Std Dev	Range
Negative	500	insulin	insulin	264	236	130.2878788	102.4822366	729.0000000
		blood	blood	481	19	70.8773389	12.1612228	98.0000000
		bmi	bmi	491	9	30.8596741	6.5607369	39.1000000
		age	age	500	0	31.1900000	11.6676548	60.0000000
Positive	268	insulin	insulin	130	138	206.8461538	132.6998982	832.0000000
		blood	blood	252	16	75.3214286	12.2998663	84.0000000
		bmi	bmi	266	2	35.4067669	6.6149824	44.2000000
		age	age	268	0	37.0671642	10.9682537	49.0000000

Chapter 5. Graphical Visualization

5.1. Describe Distribution of Continuous Variables: PROC UNIVARIATE

General Syntax

```
proc univariate data=dataset;
     var list-of-variables;
     plot-request list-of-variables / <plot-options>;
run;
```

- Plot requests

Statement	Description
CDFPLOT	Request a cumulative distribution function plot.
HISTOGRAM	Request a histogram.
PPPLOT	Request a probability-probability plot.
PROBPLOT	Request a probability plot.
QQPLOT	Request a quantile-quantile plot.

- Plot options
  - Overlay a curve showing a standard distribution.
  - Specify the desired distribution with a plot option.
  - Plot option: NORMAL, BETA, EXPONENTIAL, GAMMA, LOGNORMAL, WEIBULL.

5.2. Graphics: PROC SGPLOT

- Create one or more plots and overlay them on a single set of axes.
- Scatterplot, line plot, histogram, boxplot, regression plot, etc.
- Statement specifies the type of graph to construct.

General Syntax

```
proc sgplot data=dataset;
     statement variable-name / <options>;
run;
```

5.3. PROC SGPLOT: Distribution of Categorical Variables

- Bar chart

General Syntax

```
proc sgplot data=dataset;
     vbar variable-name / <barchart-options>;
run;
```

  - Show the distribution of a *categorical* variable.
  - Length of each bar is proportional to the number of observations in that category.
  - VBAR: Vertical bar chart / HBAR: Horizontal bar chart

Option	Description
BARWIDTH = *n*	Specify the width of bars. Values range from 0.1 to 1. Default is 0.8.
DATALABEL = `variable-name`	Display a label for each bar.
GROUP = `variable-name`	Specify a variable for grouping data.
GROUPDISPLAY = `type`	Specify how to display grouped bars. STACK (default) or CLUSTER.
MISSING	Include a bar for missing values.
DISCRETEOFFSET = *n*	Offset bars from midpoints. Useful for overlaying bar charts. Between -0.5 (left) and +0.5 (right). Default is 0 (No offset).
ALPHA = *n*	Specify the level for the confidence limits. Between 0 (100% confidence) and 1 (0% confidence). Default is 0.05 (95% confidence limits).
TRANSPARENCY = *n*	Specify the degree of transparency. Between 0 (default; completely opaque) and 1 (completely transparent).

5.4. PROC SGPLOT: Distribution of Continuous Variables

- Histogram

General Syntax

```
proc sgplot data=dataset;
     histogram variable-name / <histogram-options>;
run;
```

- The data are divided into discrete intervals called bins.
- cf. bar chart (categorical variable)

Option	Description
BINSTART = *n*	Specify the midpoint for the first bin.
BINWIDTH = *n*	Specify the bin width. Ignored if NBINS is specified.
NBINS = *n*	Specify the number of bins.
SCALE = `scaling-type`	Specify the scale for the vertical axis. PERCENT (default), COUNT, or PROPORTION.
SHOWBINS	Place tick marks at the midpoints of the bins.
TRANSPARENCY = *n*	Specify the degree of transparency. Between 0 (default; completely opaque) and 1 (completely transparent).

- Density curve

General Syntax

```
proc sgplot data=dataset;
     density variable-name / <density-options>;
run;
```

  - HISTOGRAM and DENSITY statements can be used together, but not with other types of graphs.
  - When overlaying graphs, the order of statements is important.

    The second graph will be drawn on the top of the first and could hide the first graph.

Option	Description
TYPE = `distribution-type`	Specify the type of distribution curve. Type = NORMAL (default), or KERNEL.
TRANSPARENCY = *n*	Specify the degree of transparency. Between 0 (default; completely opaque) and 1 (completely transparent).

- Boxplot

General Syntax

```
proc sgplot data=dataset;
     vbox variable-name / <boxplot-options>;
run;
```

  - Show the distribution of continuous variables by using 5-number summary.
  - Box-and-whisker plot: By default, the whiskers cannot be longer than 1.5 times the length of the box (= IQR = Q3 – Q1).
  - VBOX: Vertical boxplot / HBOX: Horizontal boxplot

Option	Description
CATEGORY = `variable-name`	Specify a categorical variable. One boxplot will be created for each value of this variable.
GROUP = `variable-name`	Specify a second categorical variable.
EXTREME	Specify that the whiskers should extend to the true min and max values. That is, outliers will not be identified.
MISSING	Include a box for missing values for the group or category variable.
TRANSPARENCY = *n*	Specify the degree of transparency. Between 0 (default; completely opaque) and 1 (completely transparent).

PROC BOXPLOT:

- Another way to create a boxplot.
- The dataset must be sorted by the categorical variable.

General Syntax

```
proc boxplot data=dataset;
     by stratified-variable; * Stratify by the 3rd variable;
     plot continuous-variable * categorical-variable;
run;
```

- Scatterplot

General Syntax

```
proc sgplot data=dataset;
     scatter x= x-variable-name y= y-variable-name / <scatter-options>;
run;
```

– Efficient way to show the relationship between two continuous variables.

Option	Description
DATALABEL = `variable-name`	Display a label for each data point.
GROUP = `variable-name`	Specify a variable for grouping data.
NOMISSINGGROUP	Specify that observations with missing values for the group variable should not be included.
TRANSPARENCY = *n*	Specify the degree of transparency. Between 0 (default; completely opaque) and 1 (completely transparent).

Example

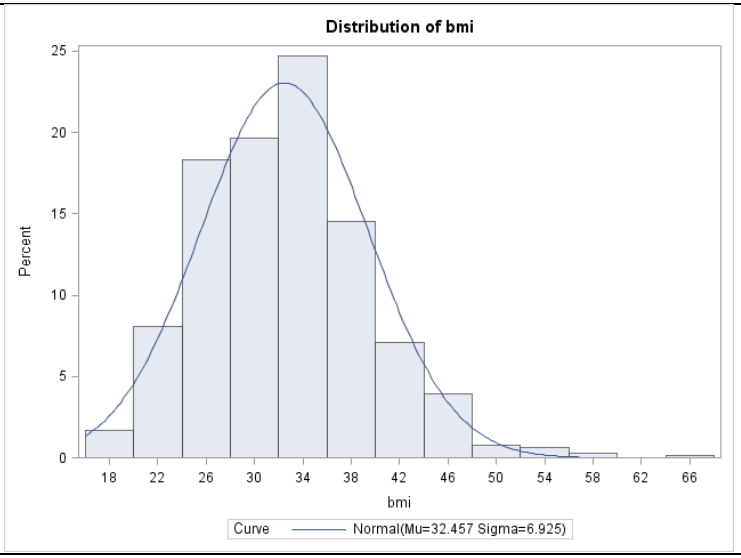
Raw Data

Obs	pregnant	blood	insulin	bmi	pedigree	age	test	BMIlevel
1	6	72	.	33.6	0.627	50	Positive	Obese
2	1	66	.	26.6	0.351	31	Negative	Overweight
3	8	64	.	23.3	0.672	32	Positive	Healthy
4	1	66	94	28.1	0.167	21	Negative	Overweight
5	0	40	168	43.1	2.288	33	Positive	Obese

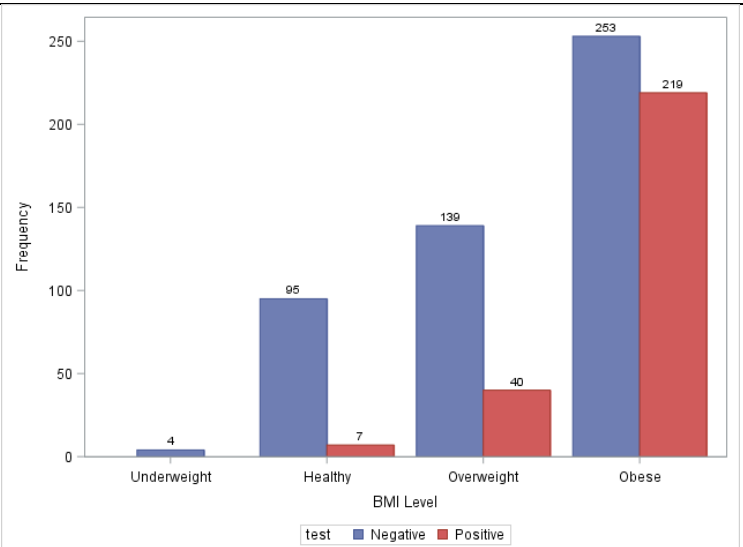
SAS Code

```
proc univariate data=pima plots;
var bmi;
histogram bmi / normal;
run;
```

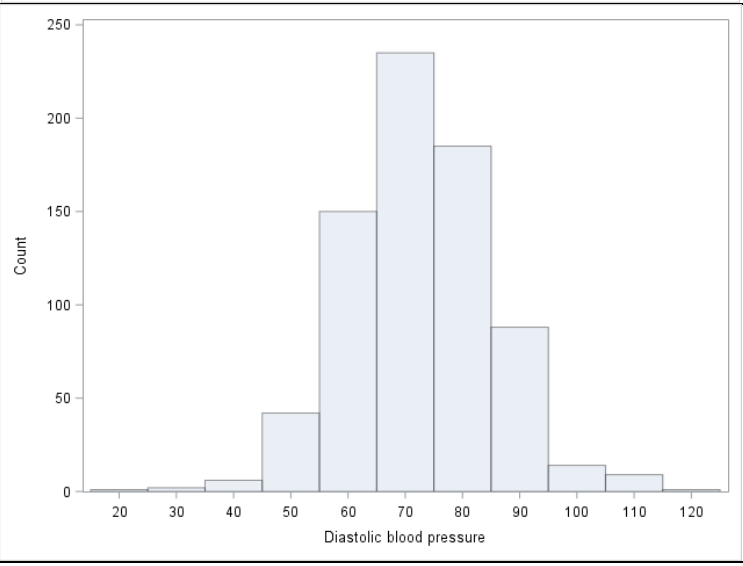
Output



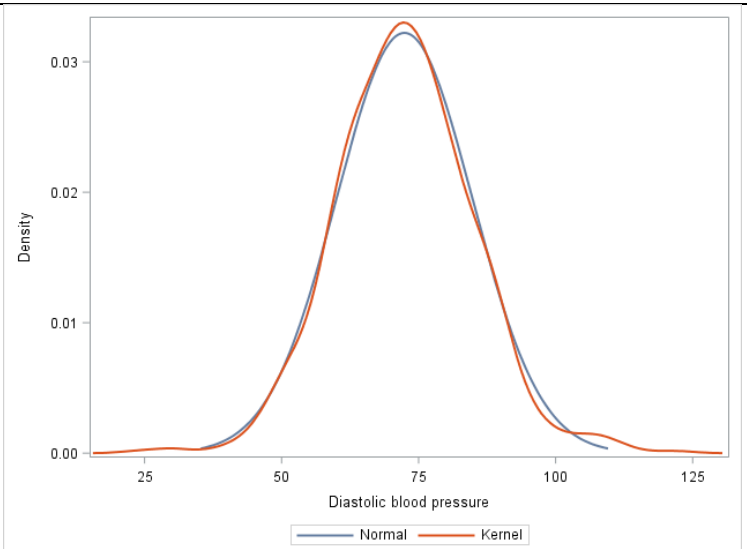
```
* Bar chart;
proc sgplot data=pima;
vbar BMIlevel / datalabel group=test
groupdisplay=cluster;
xaxis label="BMI Level";
run;
```



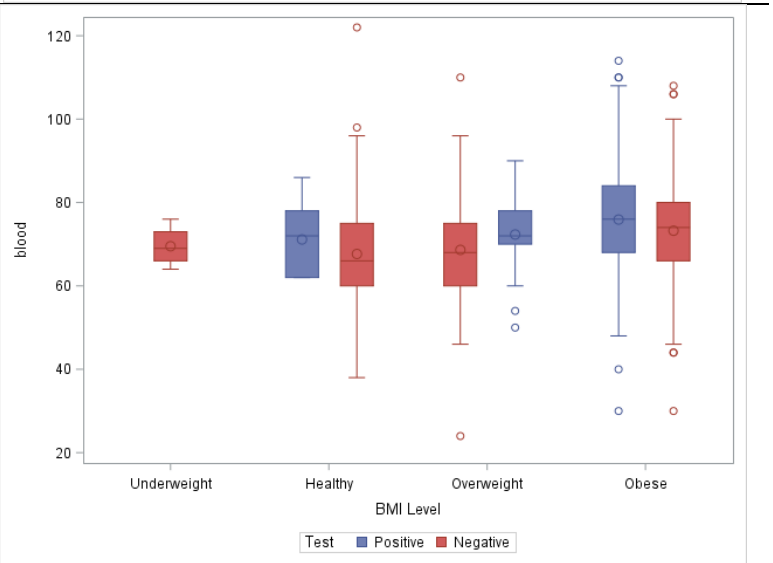

```
* Histogram;
proc sgplot data=pima;
histogram blood / binwidth=10 transparency=0.6
showbins scale=count;
xaxis label="Diastolic blood pressure";
run;
```



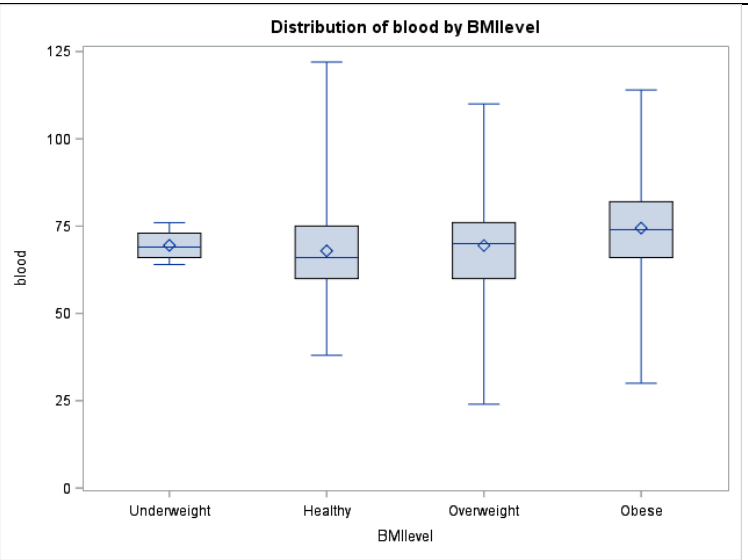
```
* Density curve;
proc sgplot data=pima;
density blood;
density blood / type=kernel;
xaxis label="Diastolic blood pressure";
run;
```



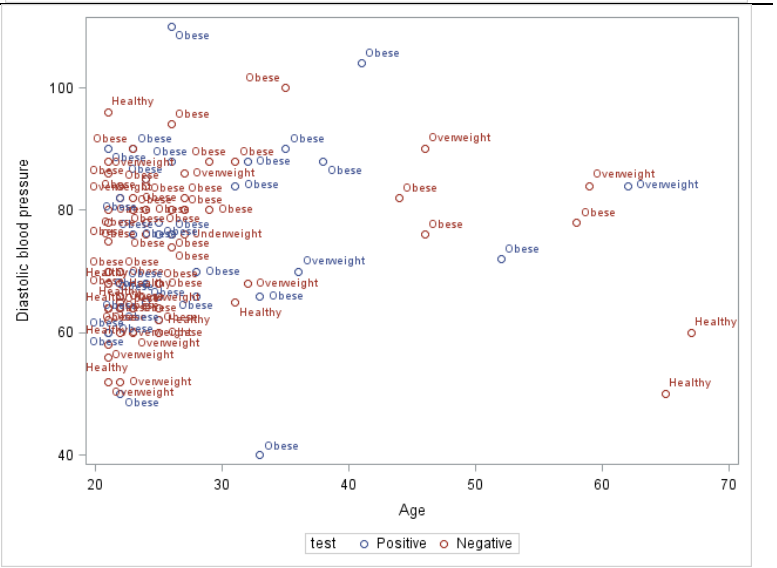

```
* Boxplot;
proc sgplot data=pima;
vbox blood / category=BMILevel group=test;
xaxis label="BMI Level";
run;
```



```
* PROC BOXPLOT;
proc sort data=pima out=pima2;
     by BMILevel;
run;

proc boxplot data=pima2;
     plot blood * BMILevel;
run;
```




```
* Scatterplot;
proc sgplot data=pima;
     where pregnant = 0;
     scatter  x=age y=blood /
datalabel=bmilevel group=test;
     xaxis label="Age";
     yaxis label="Diastolic blood pressure";
run;
```



5.5. PROC SGPLOT: Details

- Axes: Specify options for the horizontal axis (XAXIS) and vertical axis (YAXIS).

General Syntax

```
proc sgplot data=dataset;
     xaxis <options>;
     yaxis <options>;
run;
```

Option	Description
GRID	Create a line at each tick mark on the axis.
LABEL = 'text-string'	Specify a label for the axis.
TYPE = axis-type	Specify the type of axis. DISCRETE (default for character variable), LINEAR (default for numeric variable), TIME (default for time/date variable), LOG (logarithm scale).
VALUES = (values-list)	Specify values for tick marks on axes. Either as a list (0 5 10 15) or a range (0 TO 15 BY 5).

- Reference lines: Add reference lines to a graph.

General Syntax

```
proc sgplot data=dataset;
     refline values / <options>;
run;
```

  - Values can be specified either as a list (0 5 10 15) or a range (0 TO 15 BY 5).
  - If specified before any plot statements, then the line will be drawn behind the plot elements. If afterwards, then the line will be drawn in front of the plot elements.

Option	Description
AXIS = `axis`	Specify the axis that contains the reference line values. Either X or Y (default for horizontal lines).
LABEL = (`label-list`)	Specify one or more text strings as labels for the reference lines.
TRANSPARENCY = *n*	Specify the degree of transparency. Between 0 (default; completely opaque) and 1 (completely transparent).

- Legends

General Syntax

```
proc sgplot data=dataset;
     keylegend / <options>;
run;
```

– Remove legends: Add NOAUTOLEGEND to the PROC SGPLOT statement.

Option	Description
ACROSS = *n*	Specify the number of columns in the legend.
DOWN = *n*	Specify the number of rows in the legend.
LOCATION = `value`	Specify the location for the legend. Either INSIDE the axis area or OUTSIDE (default).
NOBORDER	Remove the border.
POSITION = `value`	Specify the position of the legend. TOP, TOPLEFT, TOPRIGHT, BOTTOM (default), BOTTOMLEFT, BOTTOMRIGHT, LEFT, or RIGHT.

- Insets: Place text in the axis area

General Syntax

```
proc sgplot data=dataset;
     inset 'text-string-1' ... 'text-string-k' / <options>;
run;
```

  - If more than one text string, then the strings will be placed one below the other.

Option	Description
BORDER	Add a border.
POSITION = `value`	Specify the position of the inset. TOP, TOPLEFT, TOPRIGHT, BOTTOM (default), BOTTOMLEFT, BOTTOMRIGHT, LEFT, or RIGHT.

5.6. Other Plotting Procedures

- PROC GCHART
  - Create simple bar charts and pie charts.
  - https://support.sas.com/sassamples/graphgallery/PROC_GCHART.html
- PROC GPLOT
  - Great for scatterplots with overlay straight lines (regression) or curves (smoothing lines).
  - https://support.sas.com/sassamples/graphgallery/PROC_GPLOT.html

5.7. Paneled Graphs: PROC SGPANEL

- Produce nearly all the same types of graphs as PROC SGPLOT.
- While PROC SGPLOT produces single-celled graphs, PROC SGPANEL can produce multi-celled graphs.

General Syntax

```
proc sgpanel data=dataset;
     panelby variable-name / <options>;
     plot-statement;
run;
```

- PANELBY statement must appear before any statements that create plots.
- Variable specified in the PANELBY statement is analogous to the variable in GROUP/CATEGORY option in PROC SGPLOT.
- Instead of XAXIS and YAXIS statements, PROC SGPANEL uses COLAXIS and ROWASIX statements to control axis.

Option	Description
COLUMNS = *n*	Specify the number of columns in the panel.
MISSING	Specify that observations with missing values for the PANELBY variable should be included.
NOVARNAME	Remove the variable name for cell headings.
ROWS = *n*	Specify the number of rows in the panel.
SPACING = *n*	Specify the number of pixels between rows and columns in the panel. Default is 0.
UNISCALE = `value`	Specify which axes will share the same range of values. COLUMN, ROW, and ALL (default).

Example

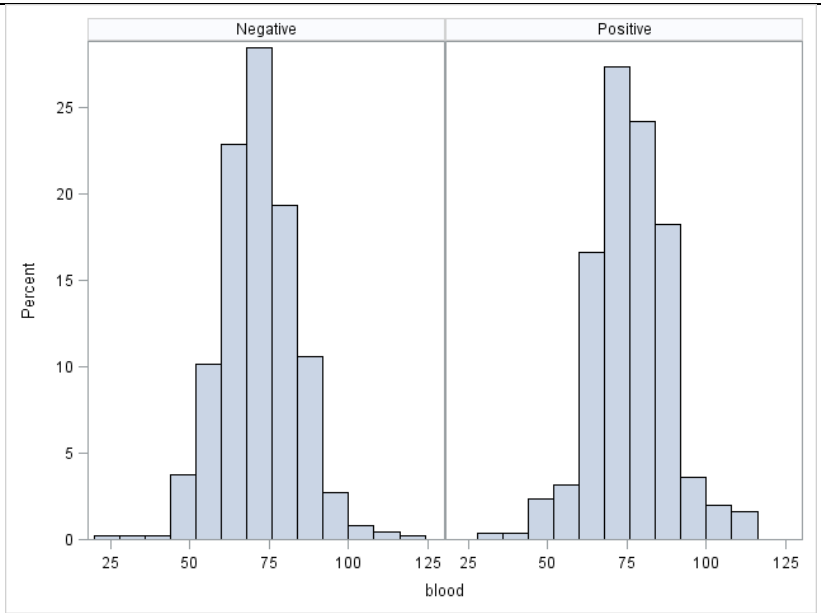
Raw Data

Obs	pregnant	blood	insulin	bmi	pedigree	age	test	BMIlevel
1	6	72	.	33.6	0.627	50	Positive	Obese
2	1	66	.	26.6	0.351	31	Negative	Overweight
3	8	64	.	23.3	0.672	32	Positive	Healthy
4	1	66	94	28.1	0.167	21	Negative	Overweight
5	0	40	168	43.1	2.288	33	Positive	Obese

SAS Code

```
proc sgpanel data=pima;
     panelby test / novarname;
     histogram blood;
run;
```

Output



Chapter 6. SAS Reporting

6.1. Produce Tabular Reports: PROC TABULATE

- Produce a variety of tabular reports, displaying frequencies and descriptive statistics.
- Similar to PROC PRINT, PROC MEANS, PROC FREQ, etc., but PROC TABULATE produces prettier reports.

General Syntax

```
proc tabulate data=dataset;
     class list-of-categorical-variables / <options>;
     var list-of-numeric-variables / <options>;
     table page-variable, row-variable, column-variable / <options>;
run;
```

- CLASS: Specify *categorical* variables to be used for dividing observations into groups.
- VAR: Specify *numeric* variables of which you will get the summary statistics.

- TABLE
  - Tell SAS how to organize a table.
  - Specify the dimensions of the table up to 3 dimensions.
  - Separate each dimension of the table by putting a comma (,) between variable names.
  - If 2 dimensions are specified, then you get rows and columns;

    If only 1 is specified, then that becomes, by default, the column dimension.
  - One TABLE statement defines only one table, but it is possible to use multiple TABLE statements in one procedure.
  - Use an asterisk (*) between variable names if including multiple variables in one dimension.
- Missing data
  - By default, observations are excluded from tables if they have missing values for variables listed in CLASS statement.
  - If you want to keep these observations, simply add missing option:

```
proc tabulate data=dataset MISSING;
```

- Keyword
  - By default, PROC TABULATE produces simple *counts* of observations in each category.
  - For other statistics (listed below), include keyword in TABLE Statement.
  - Include an asterisk (*) after/before variable names.

Option	Description	Option	Description
MAX	Maximum	ALL	Add a row, column, or page showing the total.
MIN	Minimum	N	Number of non-missings
MEAN	Mean	NMISS	Number of missings
MEDIAN	Median	SUM	Sum
MODE	Mode	PCTN	Percentage of observations for the group
STDDEV	Standard deviation	PCTSUM	Percentage of a total sum represented by the group

- Customizing table
  - FORMAT=: Change the format of all data cells in the table.
  - Associate different format with each of the variables (*f=*format*)
  - KEYLABEL: Allow to provide a label for any of the keywords used by the procedure.
  - TABLE option

    BOX=: Allow to write text in the upper left corner of the table (usually empty).

    MISSTEXT=: Specify a value for SAS to print in empty data cells.

Example

Raw Data

Obs	obs	Gender	Type	Agegroup	White blood cell	Red blood cell	Cholesterol
1	1	Female	AB	Young	7710	7.40	258
2	2	Male	AB	Old	6560	4.70	.
3	3	Male	A	Young	5690	7.53	184
4	4	Male	B	Old	6680	6.85	.
5	5	Male	A	Young	.	7.72	187

SAS Code

```
* 1-dimensional table;
proc tabulate data=blood;
     class Gender;
     table Gender;
run;
```

Output

Gender	
Female	Male
N	N
440	560

```
* 2-dimensional table;
proc tabulate data=blood;
     class Gender Type;
     table Gender, Type;
run;
```

	Type			
	A	AB	B	O
	N	N	N	N
Gender				
Female	178	20	34	208
Male	234	24	62	240

```
* Crossing, grouping, and concatenating;
proc tabulate data=blood;
      class Gender Type Agegroup;
      var wbc;
      table Gender All, mean*wbc*(Type Agegroup
All);
run;
```

	Mean						
	White blood cell						
	Type				Agegroup		All
	A	AB	B	O	Old	Young	
Gender							
Female	7218.13	7420.56	6716.07	7049.63	7105.98	7121.36	7112.43
Male	7051.01	6893.00	6990.53	6930.43	6939.35	7061.66	6987.54
All	7123.45	7142.89	6900.12	6987.06	7011.56	7089.08	7042.97

```
* Customizing your table;
proc tabulate data=blood format=comma9.2;
      class Gender AgeGroup;
      var rbc wbc Chol;
      table (Gender=' ' ALL)*(AgeGroup=' ' All),
             rbc*(n*f=3. mean*f=5.1)
             wbc*(n*f=3. mean*f=comma7.)
             chol*(n*f=4. mean*f=7.1);
   keylabel ALL = 'Total';
run;
```

		Red blood cell		White blood cell		Cholesterol	
		N	Mean	N	Mean	N	Mean
Female	Old	242	5.5	234	7,106	208	195.9
	Young	167	5.5	169	7,121	141	212.3
	Total	409	5.5	403	7,112	349	202.5
Male	Old	309	5.4	306	6,939	279	199.1
	Young	198	5.5	199	7,062	167	203.1
	Total	507	5.5	505	6,988	446	200.6
Total	Old	551	5.5	540	7,012	487	197.7
	Young	365	5.5	368	7,089	308	207.3
	Total	916	5.5	908	7,043	795	201.4

6.2. Produce Simple Outputs: PROC REPORT

- Produce output that is similar to PROC PRINT, PROC MEANS, PROC FREQ, etc., but more visually appealing.

General Syntax

```
proc report data=dataset NOWINDOWS;
... <options>;
run;
```

- Without any options, it generates the same output as PROC PRINT.
  - Except there is no observation number (obs).
  - PROC PRINT prints the variable names as column headings;

    PROC REPORT uses variable labels if they exist.
- If you have at least one character variable in your report, then, by default, SAS produces a detail report with one row per observation.
- If the report includes only numeric variables, then, by default, PROC REPORT will *sum* those variables.

- Report window
  - If you have already run PROC REPORT, you need to close the interactive Report window before re-running it.
  - NOWINDOWS or NOWD: Turn off the report output and send it to the output screen.
  - WINDOWS: Turn the default back on.
  - HEADLINE: Place an underline underneath column headings
  - HEADSKIP: Place a blank line underneath column headings.
  - Using HEADLINE and HEADSKIP together: Create a blank line underneath the underline.
  - SPLIT= ‘’: Tell SAS that you want to split the comments between the words (blank). Otherwise, other characters (slashes) are possible as line breaks.
  - MISSING: By default, observations are excluded from reports if they have missing values for variables listed in ORDER, CROUP, ACROSS statement. Use MISSING option to keep missing observations.

- Statements

Statement	Description
COLUMN	List specific variables that you want to include in the report.
WHERE	Print observations that meet specific condition.
DEFINE	Specify options to specific variables.
BREAK	Add a break for each unique value of the variable you specified.
RBREAK	Report statistics at the top/bottom of report
COMPUTE ENDCOMPUTE	Create a compute block. All variables used to compute the new variable need to be listed in the COLUMN statement.

- DEFINE options

General Syntax

```
proc report data=dataset nowindows;
     column list-of-variables;
     define variable-name / <options> 'column-header';
run;
```

Option	Description
ANALYSIS	Calculate statistics for the variable. This is the default usage for numeric variables, and the default statistic is sum.
DISPLAY	Create one row for each observation in the dataset. This is the default usage for character variable.
ACROSS	Create a column for each unique value of the variable. Combine observations by the variable and provide a sum for numeric variable or a frequency for character variable.
GROUP	Create one row for each unique value of the variable. By default, grouping by character variables produces the sum of numeric values.
ORDER	Order the rows by ascending order of the variable (default). ORDER DESCENDING: Order by descending order.
CENTER / LEFT / RIGHT	Center, left, or right alignment.
FORMAT=	Apply standard or user-defined formats.
PAGE	Put the variable on a separate page.
WIDTH=	Provide extra space. Character: The default spacing is the length of the variable. Numeric: The default spacing is 9.
COMPUTED	Create a new variable whose value you calculate is a compute block.

- (R)BREAK

General Syntax

```
proc report data=dataset;
     column list-of-variables;
     define variable-name / <options> 'column-header';
     break location variable-name / <options>;
     rbreak location / <options>;
run;
```

- LOCATION: BEFORE or AFTER, depending on whether you want the break to precede or follow the particular section of the report.
- Options: PAGE (Start a new page),

  SUMMARIZE (Insert summary statistics for numeric variables)
- BREAK: One break for every unique value of the variable you specify. The variable must be listed in a DEFINE statement with either a GROUP or ORDER option.
- RBREAK: Produce only one break at the beginning or end.

- Adding statistics

Option	Description	Option	Description
MAX/MIN	Maximum / Minimum	N	Number of non-missings
MEAN	Mean	NMISS	Number of missings
MEDIAN	Median	SUM	Sum
MODE	Mode	PCTN	Percentage of observations for the group
STD	Standard deviation	PCTSUM	Percentage of a total sum represented by the group

General Syntax

```
proc report data=dataset;
     column list-of-variables;
     column var-1 N var-2,statistic ... var-k,statistics;
run;
```

- COMPUTE & ENDCOMPUTE

General Syntax

```
proc report data=dataset;
     column list-of-variables;
     define new-variable-name / computed;

     compute new-variable-name / <options>;
          computing statements
     endcomp;
run;
```

Example

Raw Data

```
* Dataset: National parks and monuments in the USA;
data park;
     input Name $21. Type $ Region $ Museums Campings;
     datalines;
Dinosaur               NM West 2 6
Ellis Island           NM East 1 0
Everglades             NP East 5 2
Grand Canyon           NP West 5 3
Great Smoky Mountains  NP East 3 10
Hawaii Volcanoes       NP West 2 2
Lava Beds              NM West 1 1
Statue of Liberty      NM East 1 0
Theodore Roosevelt     NP .    2 2
Yellowstone            NP West 2 11
Yosemite               NP West 2 13
;
run;
```

SAS Code

```
* PROC REPORT with only numeric
variables;
proc report data=park nowindows;
     column Museums Campings;
run;
```

Output

Museums	Campings
26	50

```
* ORDER;
proc report data=park nowindows;
     column Region Name Museums
Campings;
     define Region / order;
run;
```

Region	Name	Museums	Campings
East	Ellis Island	1	0
	Everglades	5	2
	Great Smoky Mountains	3	10
	Statue of Liberty	1	0
West	Dinosaur	2	6
	Grand Canyon	5	3
	Hawaii Volcanoes	2	2
	Lava Beds	1	1
	Yellowstone	2	11
	Yosemite	2	13

```
* GROUP;
proc report data=park nowindows;
     column Region Type
            Museums Campings;
     define Region / group;
     define Type / group;
run;
```

Region	Type	Museums	Campings
East	National monument	2	0
	National park	8	12
West	National monument	3	7
	National park	11	29

```
* ACROSS & GROUP;
proc report data=park nowindows;
     column Region
            Type,(Museums Campings);
     define Region / group;
     define Type / across;
run;
```

	Type			
	National monument		National park	
Region	Museums	Campings	Museums	Campings
East	2	0	8	12
West	3	7	11	29

```
* (R)BREAK options;
proc report data=park nowindows;
     column Name Region
          Museums=museums_mean
Campings;
     define Region / order;
     define museums_mean / mean
     'Mean number of museums'
format=4.2;

     break after Region / summarize;
     rbreak after / summarize;
run;
```

Name	Region	Mean number of museums	Campings
Ellis Island	East	1.00	0
Everglades		5.00	2
Great Smoky Mountains		3.00	10
Statue of Liberty		1.00	0
	East	2.50	12
Dinosaur	West	2.00	6
Grand Canyon		5.00	3
Hawaii Volcanoes		2.00	2
Lava Beds		1.00	1
Yellowstone		2.00	11
Yosemite		2.00	13
	West	2.33	36
		2.40	48

```
* Produce N for each Region and (MEAN
and STD) for each Region x Type
category;
proc report data=park nowindows;
     column Region N Type ,
      (Museums Campings),(mean std);
     define Region / group;
     define Type / across;
run;
```

		Type							
		National monument				National park			
		Museums		Campings		Museums		Campings	
Region	N	mean	std	mean	std	mean	std	mean	std
East	4	1	0	0	0	4	1.4142136	6	5.6568542
West	6	1.5	0.7071068	3.5	3.5355339	2.75	1.5	7.25	5.5602758

```
* Compute variables;
proc report data=park nowindows;
     column Name Region Museums
Campings          Facilities Note;
     define Museums/ analysis sum
noprint;
     define Campings/ analysis sum
noprint;
     define Facilities/ computed

"Campings/and/Museums";
     define Note / computed;

     compute Facilities;
          Facilities = Museums.sum +

     Campings.sum;
     endcomp;

     compute Note / char length=10;
          if campings.sum=0
          then Note ="No Camping";
     endcomp;
run;
```

Name	Region	Museums	Campings	Campings and Museums	Note
Dinosaur	West	2	6	8	
Ellis Island	East	1	0	1	No Camping
Everglades	East	5	2	7	
Grand Canyon	West	5	3	8	
Great Smoky Mountains	East	3	10	13	
Hawaii Volcanoes	West	2	2	4	
Lava Beds	West	1	1	2	
Statue of Liberty	East	1	0	1	No Camping
Theodore Roosevelt		2	2	4	
Yellowstone	West	2	11	13	
Yosemite	West	2	13	15	

6.3. Writing Simple Custom Reports

- Useful either when reporting your result as filled in as a complete sentence or when you want one page per observation.
- FILE statement: Create a report.
- PUT statement
  - List, column, or formatted style
  - No need to worry about putting $ after character variable.
  - Control spacing with the same pointer controls that INPUT statement uses.
    (cf. 2.4. Modifiers and Pointers)
  - In addition to printing variables, you can insert text strings by simply enclosing them in quotation marks.

Example

Raw Data

Obs	Name	Type	Region	Museums	Campings
1	Dinosaur	National monument	West	2	6
2	Ellis Island	National monument	East	1	0
3	Everglades	National park	East	5	2
4	Grand Canyon	National park	West	5	3
5	Great Smoky Mountains	National park	East	3	10
6	Hawaii Volcanoes	National park	West	2	2
7	Lava Beds	National monument	West	1	1
8	Statue of Liberty	National monument	East	1	0
9	Theodore Roosevelt	National park		2	2
10	Yellowstone	National park	West	2	11
11	Yosemite	National park	West	2	13

SAS Code

```
data _NULL_;
     set park;

     Total = Museums + Campings;

     file "C:\Users\jl4201\Desktop\P6110\SAS\Chapter 6\Report.txt" print;
     title "National parks and monuments in the USA";

     put @5 Name "is a " Type "in " Region "region."
          / @5 "The number of Museums in " Name "is " Museums ","
          / @3 "and the number of camp grounds in " Name "is " Campings "."
          / @5 "That is, there are " Total "facilities in " Name "." //;

     PUT _PAGE_;
run;
```

Output

```
                National parks and monuments in the USA
1
                                                21:59 Wednesday, September 27, 2017
Dinosaur is a National monument in West region.
    The number of Museums in Dinosaur is 2 ,
  and the number of camp grounds in Dinosaur is 6 .
    That is, there are 8 facilities in Dinosaur .
                National parks and monuments in the USA
2
                                                21:59 Wednesday, September 27, 2017
Ellis Island is a National monument in East region.
    The number of Museums in Ellis Island is 1 ,
  and the number of camp grounds in Ellis Island is 0 .
    That is, there are 1 facilities in Ellis Island

....

                National parks and monuments in the USA
11
                                                21:59 Wednesday, September 27, 2017
Yosemite is a National park in West region.
    The number of Museums in Yosemite is 2 ,
  and the number of camp grounds in Yosemite is 13 .
    That is, there are 15 facilities in Yosemite .
```

Chapter 7. Output Delivery System (ODS)

7.1. Output Delivery System (ODS)

- Technically, procedures produce only data and send that data to the Output Delivery System (ODS) which determines where the output should go (destination) and what it should look like (template).
- The question is not whether you want to use ODS (You always use ODS), but whether you want to accept default output or choose something else.
- Destination
  - Type of ODS output
  - Each SAS procedure creates output objects that can be sent (separately) to destinations.
  - If not specified, the output will be sent, by default (for SAS 9.3 or later), to HTML when using SAS windowing environment.

Destination	Description
HTML	Hypertext Markup Language; Files can be used as web pages.
LISTING	Text output; What appears in the Output window
PDF	Portable Document Format
PS	PostScript
RTF	Rich Text Format; Easy to incorporate into Word document.
PRINTER	High-resolution printer output
MARKUP	Markup languages including XML, EXCELXP, LaTex, CSV
DOCUMENT	Output document; Create a reusable output document.
OUTPUT	SAS dataset

- Template
  - Template tells ODS how to format and present the data.
  - The two most common types of templates are TABLE and STYLE templates.

    TABLE: Specify the basic structure of the output.

    STYLE: Specify how the output will look.

Output object (= Data from procedure + Table template) + Style template = (ODS) = Output

7.2. ODS TRACE

General Syntax

```
ods trace on;
     Any procedure
ods trace off;
```

- Tell SAS to print information about output objects in your SAS log.
- If BY statement is used, then procedures produce one output object for each BY group.

7.3. ODS SELECT (or EXCLUDE)

General Syntax

```
proc procedurename data=dataset;
     ods select list-of-output-objects;
run;
```

- Allow you to choose just the output objects you want.
- `list-of-output-objects`: Name, label, or path of one or more output objects

7.4. ODS OUTPUT

General Syntax

```
proc procedurename data=dataset;
     ods output output-object = new-dataset;
run;
```

- Put the results from a procedure into a SAS dataset.
- Some procedures have OUTPUT statement or OUT = options.
- With ODS, you can save almost any part of procedure output as a SAS dataset by sending it to the OUTPUT destination.
  - First, use ODS TRACE statement to check the name of output object you want.
  - Then, use ODS OUTPUT statement to send that object to the OUTPUT destination.
- `output-object`: Name, label, or path of the piece of output you want to save.
- `new-dataset`: Name of the SAS dataset you want to create.

- Not belong to either DATA or PROC step.
- Open a SAS dataset and wait for the correct procedure output. The dataset remains open until the next encounter with the end of a PROC step.
- Apply to whatever PROC currently being processed, or will apply to the next PROC if there is not a current PROC.

Example

Raw Data

Obs	obs	Gender	Type	Agegroup	wbc	rbc	chol
1	1	Female	AB	Young	7710	7.40	258
2	2	Male	AB	Old	6560	4.70	.
3	3	Male	A	Young	5690	7.53	184
4	4	Male	B	Old	6680	6.85	.
5	5	Male	A	Young	.	7.72	187

SAS Code

```
* ODS TRACE;

proc sort data=blood;
     by agegroup;
run;

ods trace on;
proc means data=blood;
     var chol;
     by agegroup;
run;
ods trace off;
```

Log

```
Output Added:
-------------
Name:        Summary
Label:       Summary statistics
Template:    base.summary
Path:        Means.ByGroup1.Summary
-------------
NOTE: The above message was for the following BY group:
      Agegroup=Old

Output Added:
-------------
Name:        Summary
Label:       Summary statistics
Template:    base.summary
Path:        Means.ByGroup2.Summary
-------------
NOTE: The above message was for the following BY group:
                              Agegroup=Young
```

```
* ODS SELECT (or EXCLUDE);
ods trace on;
proc means data=blood;
     var chol;
     by agegroup;
     ods select means.bygroup1.summary;
run;
ods trace off;
```

```
Output Added:
-------------
Name:        Summary
Label:       Summary statistics
Template:    base.summary
Path:        Means.ByGroup1.Summary
-------------
```

```
* ODS OUTPUT;

* First, use ODS TRACE to check the name of
output object you want;
ods trace on;
proc freq data=blood;
     table agegroup * type / chisq fisher;
run;
ods trace off;

* Then, use ODS OUTPUT to send that object to
the OUTPUT destination;
proc freq data=blood;
     table agegroup * type / chisq fisher;
     ods output CrossTabFreqs = Freq;
run;
```

```
Output Added:
-------------
Name:        CrossTabFreqs
Label:       Cross-Tabular Freq Table
Template:    Base.Freq.CrossTabFreqs
Path:        Freq.Table1.CrossTabFreqs
-------------

Output Added:
-------------
Name:        ChiSq
Label:       Chi-Square Tests
Template:    Base.Freq.ChiSq
Path:        Freq.Table1.ChiSq
-------------

Output Added:
-------------
Name:        FishersExact
Label:       Fisher's Exact Test
Template:    Base.Freq.ChisqExactFactoid
Path:        Freq.Table1.FishersExact
-------------
```

7.5. Creating TEXT/HTML/RTF/PDF Output

- Text
  - LISTING: Create simple text output
  - To produce LISTING output in SAS windowing environment, you must open the LISTING destination.

    a) Tools → Options → Preferences → Results → Select 'Create listing'

    b) ODS LISTING; / ODS LISTING CLOSE;
  - Text output consists of basic characters without special formatting added.
  - Highly portable, compact when printed, and easily edited.

- HTML

General Syntax

```
ods html body = 'filename.html' ... <options>;
     Any procedure
ods html close;
```

  - Get files ready to be posted on a website.
  - Can be read into spreadsheets, and printed or imported into word processors.
  - HTML output is the default in SAS 9.3 or later, so no need to use ODS statements to open or close the destination.

Option	Description
BODY = 'filename'	Contain the results.
FILE = 'filename'	Synonymous with BODY=.
CONTENTS = 'filename'	Create a table of contents with links to the body file.
PAGE = 'filename'	Create a table of contents with links by page number.
FRAME = 'filename'	Create a frame that allows you to view the body file and the contents or the page file at the same time.
STYLE = style-name	Specify the style template. Default is HTMLBLUE.

- RTF

General Syntax

```
ods rtf file = 'filename.rtf' ... <options>;
     Any procedure
ods rtf close;
```

  - Developed by Microsoft for document interchange.
  - Can copy RTF output into a Word document and edit it.

Option	Description
BODYTITLE	Put titles and footnotes in the main part of the RTF documents.
COLUMNS = *n*	Request columnar output where *n* is the number of columns.
STARTPAGE = `value`	Control page breaks. Default is YES, inserting a page break between procedures. NO turns off page breaks. NOW inserts a page break at that point.
STYLE = `style-name`	Specify the style template. Default is RTF.

- PDF

General Syntax

```
ods pdf file = 'filename.pdf' ... <options>;
     Any procedure
ods pdf close;
```

  - A member of the PRINTER family of ODS destinations

Option	Description
COLUMNS = *n*	Request columnar output where *n* is the number of columns.
STARTPAGE = `value`	Control page breaks. Default is YES, inserting a page break between procedures. NO turns off page breaks. NOW inserts a page break at that point.
STYLE = `style-name`	Specify the style template. Default is PRINTER.

7.6. ODS GRAPHICS

General Syntax

```
ods graphics on;
ods graphics / <options>;
ods destination-name <options>;
     Any procedure
ods graphics off;
```

- It is on by default in SAS 9.3 or later in SAS windowing environment.
- ODS GRAPHICS is not a destination.
- Default size = 640 pixels x 480 pixels. If only one dimension is specified, then SAS will adjust the other dimension to maintain a default aspect ratio of 4:3.

- ODS GRAPHICS options

Option	Description
HEIGHT = *n*	Specify the image height in CM, IN, MM, PT, or PX.
IMAGENAME = `'filename'`	Specify the base image filename. Default is the name of its ODS output object.
OUTPUTFMT = `file-type`	Specify the graph format. Default varies by destination. BMP, GIF, JPEG, PDF, PNG, PS, SVG, TIFF, etc.
RESET	Reset options to defaults.
WIDTH = *n*	Specify the image width in CM, IN, MM, PT, or PX.

- Saving graphical output
  - When saving image files, SAS will append numerals to the end of the image name.
  - `destination-name`: ODS destination such as HTML, LISTING, PDF, or RTF.
  - For some destinations including PDF and RTF, graphs and tabular output are integrated together in a single file.
  - For other destinations including LISTING and HTML, graphs are saved separately from tabular output.

Option	Description
FILE= ’`filename`’	Specify a path and filename for saving output images from PDF and RTF destinations. Images will be saved in a single file along with tabular output.
GPATH = ’`path`’	Specify a path for saving output images from LISTING and HTML destinations. Images will be saved in individual files.
DPI = *n*	Specify the image resolution for PDF destination. Default is 200.
IMAGE_DPI = *n*	Specify the image resolution for HTML, LISTING, and RTF destinations.
STYLE = `style-name`	Specify a style template.

Chapter 8. Loops and Arrays

8.1. Array

- An array is an ordered *group* of similar items/variables.
- Array elements do not need to be contiguous, the same length, or even related at all.
- All elements should be either all numeric or all character.
- Arrays *simplify* and *shorten* your program when doing the same thing to many variables.
- Examples
  - Take the log of every numeric variable.
  - Set missing values to 999 or vice versa.
  - Write a series of assignment statements or IF statements.
  - Set up new indexed names for variables.
  - Compute new variables. (e.g. Transformation from continuous to binary)
  - Reshape datasets. (e.g. From wide to long format or vice versa)

8.2. ARRAY Statement

- Arrays are defined in DATA step.
- Either refer to already defined variables (Note: Not datasets or values of a variable) or create new variables.
- For character variables, you must include $.
- Basic structure of an array statement
  - Array name
  - Array size in brackets {}, [] or parentheses ()
  - Names of variables in the array (optional)
- Limitations
  - Arrays can only be used in a DATA step, not a PROC step.
  - Arrays are not used to combine data over multiple subjects.
  - Array references cannot be used as an input to a MACRO parameter or in FORMAT, LABEL, DROP, KEEP, LENGTH, or OUTPUT statement.

General Syntax

```
array array-name [array-size] <$> <length> list-of-array-elements;
```

- If array-size is specified, it is an *indexed* array. Otherwise, it is a *non-indexed* array.

  SAS can figure out the size based on the elements (variables) used to define the array.

  - ARRAY: SAS keyword that specifies that an array is being defined
  - array-name: Valid SAS name that is *not* a variable name in the dataset
  - array-size: Number of elements in the array
  - <$>: Specify if the new variables created in the array are character variables. Default type is numeric.
  - <length>: Length of new variables created in the array (optional)
  - list-of-array-elements: List of variables of the same type (all numeric or all character) to be included in the array

- Arrays are not stored with the dataset; it is defined only for the duration of the DATA step.
- You can give arrays any name, as long as they do not match any of the variable names in the dataset or any SAS keywords.

Example	
SAS Code	`array STORE (4) Macys Pennys Sears Target;`
Array	STORE(1) is the variable Macys. STORE(2) is the variable Pennys. STORE(3) is the variable Sears. STORE(4) is the variable Target.

- If omit the variable list, SAS automatically creates variable names, using the array name as base and adding numbers from 1 to *n* (=array-size).

  (e.g. array new {8}: Variables new1, new2, …, new8 will be created by SAS.)
- Arrays are most useful if there are many variables for each subjects.

- Special group of variables
  - _NUMERIC_: Use all the numeric variables as array elements.
  - _CHARACTER_: Use all the character variables as array elements.
  - _ALL_: Use all variables in the dataset as array elements. All variable should have the same type (either all numeric or all character).

Example

```
array all_numeric {*} _NUMERIC_;
```

  - Asterisk (*) is used when you do not know the number of variables in your array.

- Temporary array _TEMPORARY_
  - Useful for storing constant/character variables for calculation.
  - Require less storage than regular variables.
  - No corresponding variables to identify the array elements.

Example

IF

```
if month = 1 then balance = balance + (balance * 0.05);
else if month = 2 then balance = balance + (balance * 0.08);
else if month = 3 then balance = balance + (balance * 0.12);
else if month = 4 then balance = balance + (balance * 0.20);
else if month = 5 then balance = balance + (balance * 0.27);
else if month = 6 then balance = balance + (balance * 0.35);
```

Array

```
array rate {6} _temporary_ (0.05 0.08 0.12 0.20 0.27 0.35);
if month ge 1 and month le 6 then
balance = balance + (balance * rate{month});
```

8.3. Loop

- Loops are useful for *repeating* blocks of action, by avoiding writing the same statement multiple times.
- Commonly used with arrays.
- Specify conditions for repeating an action:
  - Repeat for a set number of times.
  - Repeat if certain conditions are met.

8.4. DO Loop

- Structure of a DO loop: DO statement and END statement
- Programming statements within the loop will be performed once each time through the loop.
- DO statement
  - Index: Variable which tracks the number of times through the loop
  - Bounds: Upper and lower limits for the index
  - By default, index variable (called 'counter') counts up by 1. (e.g. do i=1 to 5;)
  - It can be set up to count in different increments. (e.g. do i=1 to 5 by 0.5;)
  - It can skip values. (e.g. do i=1, 4, 7, 10;)
- How a DO loop runs:
  - Index is set to the lower bound.
  - Statements in the loop are executed until the end statement.
  - 1 (default increment) is added to the index at the *end* of the loop.
  - If index is not above the upper bound, go to step 2.

- Other types of DO loops

Type	Description
DO UNTIL (logical condition)	Perform the operations in the do loop until the logical condition is satisfied.
DO WHILE (logical condition)	Perform the operations in the do loop while the logical condition is satisfied.
DO OVER	Perform the operations in the do loop over all elements in the array.

  - Initialize the logical condition before entering the loop.
  - The (stopping) logical condition is checked *before* the loop is executed.

SAS Code

```
data squares;
     do i = 1 to 5;
          y = i * i;
     output;
     end;
run;
```

Output

Obs	i	y
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25

```
data array;
input A B C D E;
datalines;
1 . 1 0 1
0 . . 1 1
1 1 0 1 .
;
run;
```

```
data array1;
set array;
array vars[5] A -- E;
do i=1 to dim(vars);
if vars[i]=. then vars[i] = 999;
end;
drop i;
run;
```

Obs	A	B	C	D	E
1	1	999	1	0	1
2	0	999	999	1	1
3	1	1	0	1	999

```
data careless;
input Score
LastName $ (Ans1-
Ans3)($1.);
datalines;
100 John Abc
65 sMITH CCd
95 scerbo DeD
;
run;
```

```
data careful;
set careless;
array all_chars{*}_character_;
do i = 1 to dim(all_chars);
all_chars{i} =
lowcase(all_chars{i});
end;
drop i;
run;
```

Obs	Score	LastName	Ans1	Ans2	Ans3
1	100	john	a	b	c
2	65	smith	c	c	d
3	95	scerbo	d	e	d

```
data spirometry;
input PatientID FVC1 FEV1 FVC2 FEV2 FVC3 FEV3;
cards;
     1     3.4   2.5   3.6   2.7   3.2   2.4
     2     4.2   3.3   3.9   3     4.3   3.3
     3     4.5   3.1   4.7   3     4.7   2.8
     4     3.8   3     3.7   2.8   4     3.1
     5     5     2.7   4.9   2.9   4.7   2.7
     ;
run;
data spirometryratios;
set spirometry;
array FVC{3} ; array FEV{3} ; array Ratio{3} ;
do i = 1 to 3;
Ratio{i} = round(FEV{i} / FVC{i}, .01);
end; drop i FVC1 -- FEV3; run;
```

Obs	PatientID	Ratio1	Ratio2	Ratio3
1	1	0.74	0.75	0.75
2	2	0.79	0.77	0.77
3	3	0.69	0.64	0.60
4	4	0.79	0.76	0.78
5	5	0.54	0.59	0.57

```
data score;
array Ans{10} $ 1 ; * 1=length (optional);
array key[10] $ 1 _temporary_
('A','B','C','D','E','E','D','C','B','A');
input ID (Ans1-Ans10)($1.);
RawScore = 0;
do Ques = 1 to 10;
RawScore = RawScore + (key{Ques} eq Ans{Ques});
end;
Percent = 100 * RawScore/10;
keep ID RawScore Percent;
datalines;
 123 ABCDEDDDCA
 126 ABCDEEDCBA
 129 DBCBCEDDEB
;
run;
```

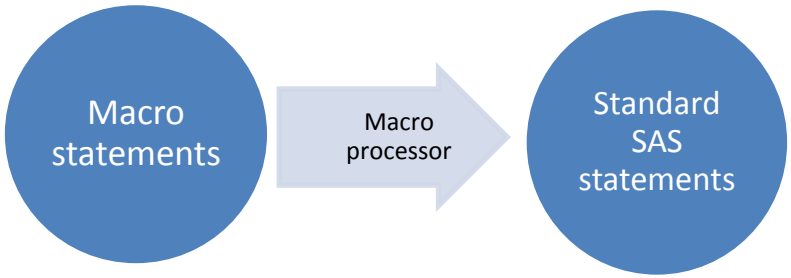
Obs	ID	RawScore	Percent
1	123	7	70
2	126	10	100
3	129	4	40

Chapter 9. Macros

9.1. Macros

- A macro is a way to *automate* a task you perform repeatedly or on a regular basis.
- A series of commands and cations can be stored and run whenever needed.
- Macros can make the development and maintenance of production programs much easier.
  - Avoid repetitious SAS code.
  - Create generalizable and flexible SAS code.
  - Conditionally execute DATA steps and PROC steps.
  - Pass information from one part of a SAS job to another.
  - Make one small change and have SAS echo that change throughout the program.
  - Store macros in a central location and share them between programs and between programmers.

- Macro processor



  - Standard SAS program: SAS complies and immediately executes it.
  - Macro: SAS must pass the macro statements to the macro processor that resolves them, generating standard SAS code.
  - 'meta-programming': Write a program that writes a program.
- SAS macro code consists of two basic parts: macros and macro variables

9.2. Key Symbols

- *&name* (Macro variable reference)
  - Name of macro variables are prefixed with an ampersand (&).
  - It does not belong to a dataset, and its value is always character.
  - This value could be a variable name, a numeral, or any text that you want to substituted into your program.

- *%name* (Macro call)
  - Name of macros are prefixed with a percent sign (%).
  - Larger piece of a program that may contain complex logic including complete DATA steps and PROC steps and macro statements.
  - e.g. %DO and %END, %IF-%THEN/%ELSE.

9.3. Macro Variables

- Efficient way of replacing text strings in SAS code
- Can be defined within a macro definition (local) or within a statement that is outside a macro definition (global).
- Macro variables defined by SAS: When you invoke SAS, the macro processor creates automatic macro variable that supply information related to the SAS session.
- %LET: Assign a value to a macro variable.

Example

Before

```
%let iterations=10;
%let country = Canada;

do i=1 to &iterations;
title "Addresses in &country";[2]
```

After
(Resolved by macro processor)

```
do i=1 to 10;
title "Addresses in Canada";
```

[2] In open code (anywhere outside a macro definition), the macro variables should be referenced only within double quotation marks. Macro processor does not look for macros inside single quotation mark.

- Some automatic SAS macro variables

Variable	Description
SYSDATE	Current date
SYSDAY	Current day of the week
SYSTIME	Starting time of job
SYSDSN	Last SAS dataset built
SYSINFO	System information given by some PROCs
SYSSCP	Operating system where SAS is running
SYSVER	SAS version

9.4. Macro functions

- Process one or more arguments and produce a result.
- Used in both macro definitions and open code. (i.e. inside or outside the macro)
- Example: %LENGTH, %EVAL, %UPCASE, %PUT

Example

Raw Data

Obs	obs	Gender	Type	Agegroup	wbc	rbc	chol
1	1	Female	AB	Young	7710	7.40	258
2	2	Male	AB	Old	6560	4.70	.
3	3	Male	A	Young	5690	7.53	184
4	4	Male	B	Old	6680	6.85	.
5	5	Male	A	Young	.	7.72	187

SAS Code

```
%let bc= blood cell;

data blood;
     set blood;
     label wbc = "White &bc"
           rbc = "Red &bc"
           chol = "Cholesterol";
run;
```

Output

Obs	obs	Gender	Type	Agegroup	White blood cell	Red blood cell	Cholesterol
1	1	Female	AB	Young	7710	7.40	258
2	2	Male	AB	Old	6560	4.70	.
3	3	Male	A	Young	5690	7.53	184
4	4	Male	B	Old	6680	6.85	.
5	5	Male	A	Young	.	7.72	187

```
%let var_list = rbc wbc chol;
proc means data=blood n mean min max
maxdec=1;
   var &var_list;
run;
```

Variable	Label	N	Mean	Minimum	Maximum
rbc	Red blood cell	916	5.5	1.7	8.8
wbc	White blood cell	908	7043.0	4070.0	10550.0
chol	Cholesterol	795	201.4	17.0	331.0

```
title "It is &systime on &sysday,
&sysdate.";
proc print data=blood (obs=5) noobs;
run;
```

It is 00:23 on Wednesday, 14FEB18

obs	Gender	Type	Agegroup	wbc	rbc	chol
1	Female	AB	Young	7710	7.40	258
2	Male	AB	Old	6560	4.70	.
3	Male	A	Young	5690	7.53	184
4	Male	B	Old	6680	6.85	.
5	Male	A	Young	.	7.72	187

9.5. Macro Programs

General Syntax

```
* Macro programs;
%macro macro-name(list-of-parameters);
Macro-text (Macro definition)
%mend macro-name;

* Invoke the macro;
%macro-name(list-of-parameters);
```

- %MACRO: Tell SAS that this is the beginning of a macro.
- %MEND: Mark the end.
- The macro-name in %MEND statement is optional, but recommended for easier debugging.
- Macro-text: A set of statements
- Invoking a macro: Add the percent sign prefix to its name.
- Macros with conditional logic: Combine macros and macro variables.

9.6. Tips: How to Avoid Macro Errors

- Develop your program in a *piecewise* fashion.
- Write your code in standard SAS code and make sure that it is bug-free. Then, convert it to macro logic by adding one feature at a time.

Example

```
* MACRO with PROC FREQ;
%macro freqmac(datain, var1, var2);
proc freq data=&datain;
     table &var1*&var2;
run;
%mend freqmac;

%freqmac(blood, gender, type);
```

```
* Macros with conditional logic;
%macro report;
%if &sysday = Friday %then %do;
proc print data=blood (obs=10);
     where gender = "Female";
run;
%end;
%else %do;
proc print data=blood (obs=10);
     where gender ne "Female";
run;
%end;
%mend report;

%report;
```

Chapter 10. Restructuring Longitudinal Datasets

10.1. Wide vs Long Format

- Wide format
  - A subject's repeated responses [measurements] will be in a single row.
  - Each response [measurement] is in a separate column.
- Long format
  - Each row is one time-point per subject.
  - A subject with *n* repeated responses [measurements] takes *n* rows of the dataset.
- Restructuring (i.e. converting wide to long or vice versa) a dataset is useful because different analyses require different setups.

	obs	ID	male	exposure	tol1	tol2	tol3	tol4	tol5
1	1	9	0	1.54	2.23	1.79	1.9	2.12	2.66
2	2	45	1	1.16	1.12	1.45	1.45	1.45	1.99
3	3	268	1	0.9	1.45	1.34	1.99	1.79	1.34
4	4	314	0	0.81	1.22	1.22	1.55	1.12	1.12
5	5	442	0	1.13	1.45	1.99	1.45	1.67	1.9
6	6	514	1	0.9	1.34	1.67	2.23	2.12	2.44
7	7	569	0	1.99	1.79	1.9	1.9	1.99	1.99
8	8	624	1	0.98	1.12	1.12	1.22	1.12	1.22
9	9	723	0	0.81	1.22	1.34	1.12	1	1.12
10	10	918	0	1.21	1	1	1.22	1.99	1.22
11	11	949	1	0.93	1.99	1.55	1.12	1.45	1.55
12	12	978	1	1.59	1.22	1.34	2.12	3.46	3.32
13	13	1105	1	1.38	1.34	1.9	1.99	1.9	2.12
14	14	1542	0	1.44	1.22	1.22	1.99	1.79	2.12
15	15	1552	0	1.04	1	1.12	2.23	1.55	1.55
16	16	1653	0	1.25	1.11	1.11	1.34	1.55	2.12

: Wide format

	obs	ID	male	exposure	measure	tolerance
1	1	9	0	1.54	1	2.23
2	1	9	0	1.54	2	1.79
3	1	9	0	1.54	3	1.9
4	1	9	0	1.54	4	2.12
5	1	9	0	1.54	5	2.66
6	2	45	1	1.16	1	1.12
7	2	45	1	1.16	2	1.45
8	2	45	1	1.16	3	1.45
9	2	45	1	1.16	4	1.45
10	2	45	1	1.16	5	1.99
11	3	268	1	0.9	1	1.45
12	3	268	1	0.9	2	1.34
13	3	268	1	0.9	3	1.99
14	3	268	1	0.9	4	1.79
15	3	268	1	0.9	5	1.34
16	4	314	0	0.81	1	1.22
17	4	314	0	0.81	2	1.22
18	4	314	0	0.81	3	1.55
19	4	314	0	0.81	4	1.12
20	4	314	0	0.81	5	1.12
21	5	442	0	1.13	1	1.45
22	5	442	0	1.13	2	1.99
23	5	442	0	1.13	3	1.45
24	5	442	0	1.13	4	1.67
25	5	442	0	1.13	5	1.9

: Long format

10.2. Restructure Datasets: PROC TRANSPOSE

- Quick and simple solution to restructure SAS datasets
- Missing values or duplicates can create problems. (Convert using ARRAY instead.)

General Syntax

```
proc transpose data=old-dataset out=new-dataset prefix=transvar;
     id list-of-id-variables;
     by list-of-by-variables;
     var list-of-variables;
run;
```

- OUT: Specify the new dataset containing the transposed data.
- PREFIX: Create the names for the transposed variables.

- Statement

Statement	Description
ID	Values of this variable will become variable names. If more than one variable is listed, then the values of all variables in the ID statement will concatenated to form the new variable names. If not specified, then the new variable names will be named col1, col2, etc.
BY	Specify the grouping variables that you want to keep as variables. Dataset must be sorted by the BY variable before transposing. The transposed dataset will have one observation for each BY level per variable transposed.
VAR	Specify the variables to transpose. They become rows for each level of the BY variable. SAS creates a new variable '_NAME_' which has as values the names of the variables in the VAR statement.

Example

Raw Data

Obs	obs	ID	male	exposure	measure	tolerance
1	1	9	0	1.54	0	2.23
2	1	9	0	1.54	1	1.79
3	1	9	0	1.54	2	1.90
4	1	9	0	1.54	3	2.12
5	1	9	0	1.54	4	2.66
6	2	45	1	1.16	0	1.12
7	2	45	1	1.16	1	1.45
8	2	45	1	1.16	2	1.45
9	2	45	1	1.16	3	1.45
10	2	45	1	1.16	4	1.99

SAS Code

```
* Covert from long to wide format;
proc transpose data=long out=wide
(drop=_NAME_) prefix=tol;
     by obs ID male exposure;
     var tolerance;
run;
```

Output

Obs	obs	ID	male	exposure	tol1	tol2	tol3	tol4	tol5
1	1	9	0	1.54	2.23	1.79	1.90	2.12	2.66
2	2	45	1	1.16	1.12	1.45	1.45	1.45	1.99
3	3	268	1	0.90	1.45	1.34	1.99	1.79	1.34
4	4	314	0	0.81	1.22	1.22	1.55	1.12	1.12
5	5	442	0	1.13	1.45	1.99	1.45	1.67	1.90

```
* Covert from wide to long format;
proc transpose data=wide out=long2
(rename=(col1=tolerance) drop=_name_);
     by obs ID male exposure;
     var tol1 - tol5;
run;
```

Obs	obs	ID	male	exposure	tolerance
1	1	9	0	1.54	2.23
2	1	9	0	1.54	1.79
3	1	9	0	1.54	1.90
4	1	9	0	1.54	2.12
5	1	9	0	1.54	2.66
6	2	45	1	1.16	1.12
7	2	45	1	1.16	1.45
8	2	45	1	1.16	1.45
9	2	45	1	1.16	1.45
10	2	45	1	1.16	1.99

10.3. SAS Automatic Variables

- _N_
  - Indicate the number of times SAS has looped through the DATA step.
  - Not necessarily equal to the observation number.

    e.g. A simple subsetting IF statement can change the relationship between observation number and the number of iterations of the DATA step.
- FIRST.*variable* and LAST.*variable*
  - Available when BY statement is used in a DATA step.
  - The dataset must be sorted by the BY variable.
  - Used to pick one row out of several rows with the same value of a variable.
  - Especially useful when subjects do not have the same number of observations.
  - FIRST.*variable* [LAST.*variable*] will have a value of 1 when SAS is processing an observation with the *first [last] occurrence* of a new value for that variable and a value of 0 for the other observations.

10.4. RETAIN Statement

- Ordinarily, DATA step in SAS operates by
  - Reading in one row from a source of data
  - Working down through all of the commands in the DATA step
  - Purging the old values of the variables
  - Returning to the top of the DATA step
- The process above starts again when a new row of data is read in.
- RETAIN
  - Variables listed in a RETAIN statement do not have its value purged from memory when the DATA step reaches its end.
  - Allow to *hold* information from one observation to the next.
  - Can be used to summarize information from multiple records.
  - Also used for rearranging the order of variables in the dataset.

Example

Raw Data

```
data one;
     input id visit cost @@;
     cards;
     1 1 12 2 1 3
     3 1 11 1 2 13
     3 2 8 1 3 21
;
run;

proc sort data=one; by id visit; run;
```

Obs	id	visit	cost
1	1	1	12
2	1	2	13
3	1	3	21
4	2	1	3
5	3	1	11
6	3	2	8

SAS Code

```
* No RETAIN;
data two;
set one;
by id;
if first.id then
totcost=cost;
else
totcost=totcost+cost;
run;
```

```
* RETAIN;
data three;
set one;
by id;
retain totcost;
if first.id then
totcost=cost;
else totcost=totcost+cost;
run;
```

```
* RETAIN, last.var;
data four;
set one;
by id;
retain totcost;
if first.id then
totcost=cost;
else totcost=totcost+cost;
if last.id;
run;
```

Output

Obs	id	visit	cost	totcost
1	1	1	12	12
2	1	2	13	.
3	1	3	21	.
4	2	1	3	3
5	3	1	11	11
6	3	2	8	.

Obs	id	visit	cost	totcost
1	1	1	12	12
2	1	2	13	25
3	1	3	21	46
4	2	1	3	3
5	3	1	11	11
6	3	2	8	19

Obs	id	visit	cost	totcost
1	1	3	21	46
2	2	1	3	3
3	3	2	8	19

10.5. Reshape Using ARRAY

- Convert a dataset with several values per row into a dataset with one value per row.
- Combine several rows of data into one.

  (cf. PROC TRANSPOSE)
- Create several datasets within one DATA step.

Example

Raw Data

```
data missing;
     input id x1-x3 @@;
     cards;
1 1 2 . 2 3 5 2
;
run;
```

Obs	id	x1	x2	x3
1	1	1	2	.
2	2	3	5	2

SAS Code

```
* From wide to long;
proc transpose data=missing out=missing_long
(drop = _NAME_ rename=(col1=xx));
     by id;
     var x1-x3;
run;
```

Output

Obs	id	xx
1	1	1
2	1	2
3	1	.
4	2	3
5	2	5
6	2	2

```
data missing_long2;
     set missing;
     array x{3};
     do visit=1 to 3;
          if missing(x{visit}) then leave;
          xx = x{visit};
          output;
     end;
     keep id xx;
run;
```

Obs	id	xx
1	1	1
2	1	2
3	2	3
4	2	5
5	2	2

```
* From long to wide;
proc transpose data=missing_long out=missing_wide
(drop=_NAME_) prefix=x;
     by id;
     var xx;
run;

data missing_wide2;
     array x{3};
     do i=1 to 3 until (last.id);
          set missing_long;
          by id;
          x{i} = xx;
     end;
     keep id x1-x3;
run;
```

Obs	id	x1	x2	x3
1	1	1	2	.
2	2	3	5	2

10.6. Graphical visualization for longitudinal datasets: PROC SGPLOT

- Longitudinal dataset: Data recorded at multiple time points
- Visualize the time trend.
  - Overlay all treatment groups – easy to compare
  - Overlay summary statistics at each time point
- PROC SGPLOT: SERIES

General Syntax

```
proc sgplot data=dataset;
     series x= x-variable-name y= y-variable-name / <series-options>;
run;
```

Option	Description
CURVELABEL = `'text-string'`	Add a label for the curve.
DATALABEL = `variable-name`	Display a label for each data point.
GROUP = `variable-name`	Specify a variable for grouping data.
MARKERS	Add a marker for each data point.
BREAK	Create a break in the line for each missing value for the Y variable.
NOMISSINGGROUP	Specify that observations with missing values for the group variable should not be included.
TRANSPARENCY = *n*	Specify the degree of transparency. Between 0 (default; completely opaque) and 1 (completely transparent).

- PROC SGPLOT: REG (regression line or curve), LOESS (loess curve), PBSPLINE (penalized B-spline curve)

General Syntax

```
proc sgplot data=dataset;
     statement-name x= x-variable-name y= y-variable-name / <options>;
run;
```

Option	Description
ALPHA = *n*	Specify the level for the confidence limits. Between 0 (100% confidence) and 1 (0% confidence). Default is 0.05 (95% confidence limits).
CLI (for REG, PBSPLINE)	Add prediction limits for individual predicted values.
CLM	Add confidence limits for mean predicted values.
CURVELABEL = `'text-string'`	Add a label for the curve.
GROUP = `variable-name`	Specify a variable for grouping data.
NOMARKERS	Remove markers for data points.
CLMTRANSPARENCY = *n*	Specify the degree of transparency for the confidence limits. Between 0 (default; completely opaque) and 1 (completely transparent).

Example

Raw Data

Data 'long'

Obs	obs	ID	male	exposure	measure	tolerance
1	1	9	0	1.54	0	2.23
2	1	9	0	1.54	1	1.79
3	1	9	0	1.54	2	1.90
4	1	9	0	1.54	3	2.12
5	1	9	0	1.54	4	2.66
6	2	45	1	1.16	0	1.12
7	2	45	1	1.16	1	1.45
8	2	45	1	1.16	2	1.45
9	2	45	1	1.16	3	1.45
10	2	45	1	1.16	4	1.99

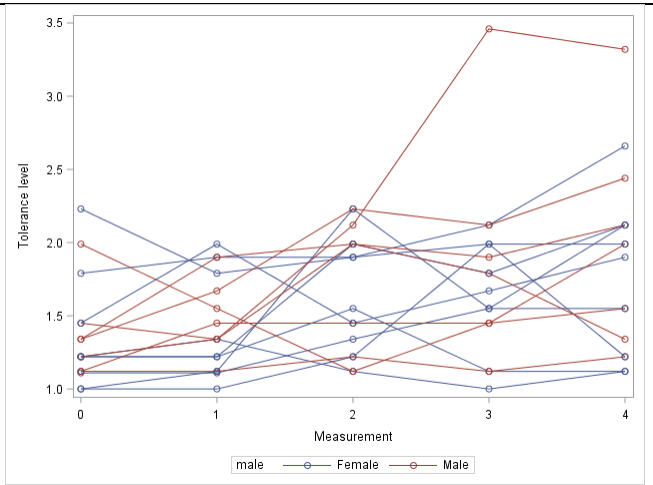
Data 'long_plot'

Obs	obs	ID	male	exposure	measure	tolerance
1	1	9	0	1.54	0	2.23
2	1	9	0	1.54	1	1.79
3	1	9	0	1.54	2	1.90
4	1	9	0	1.54	3	2.12
5	1	9	0	1.54	4	2.66
6	1	9	0	1.54	4	.
7	2	45	1	1.16	0	1.12
8	2	45	1	1.16	1	1.45
9	2	45	1	1.16	2	1.45
10	2	45	1	1.16	3	1.45
11	2	45	1	1.16	4	1.99
12	2	45	1	1.16	4	.

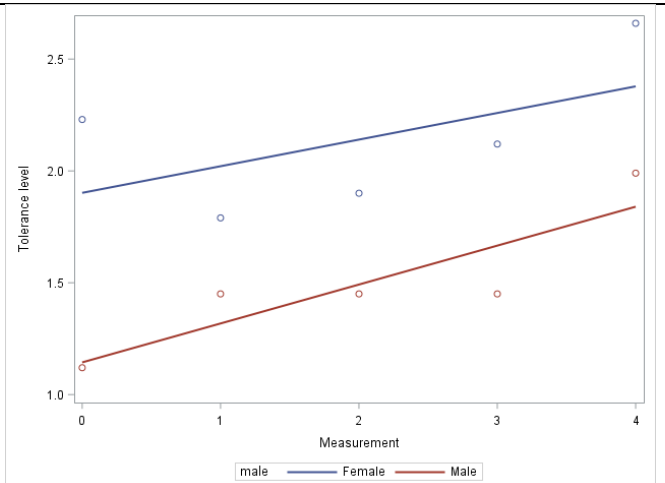
SAS Code

```
* PROC SGPLOT: SERIES;
proc sgplot data=long_plot;
      series x=measure y=tolerance
/markers group=male break;
      xaxis label="Measurement";
      yaxis label="Tolerance level";
      format male malefmt.;
run;
```

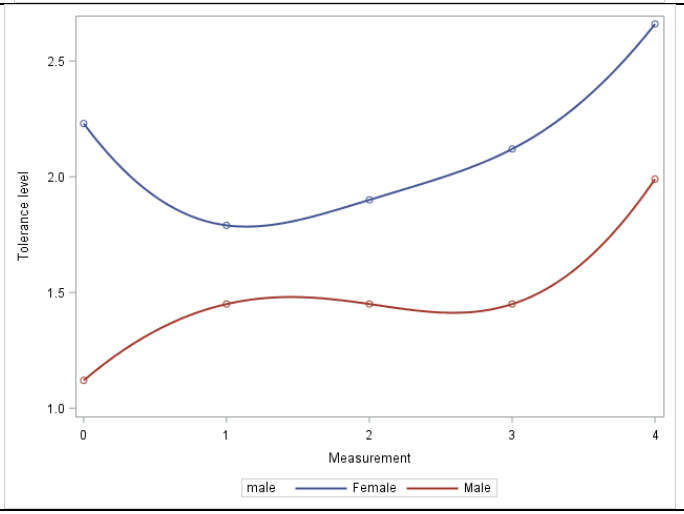
Output



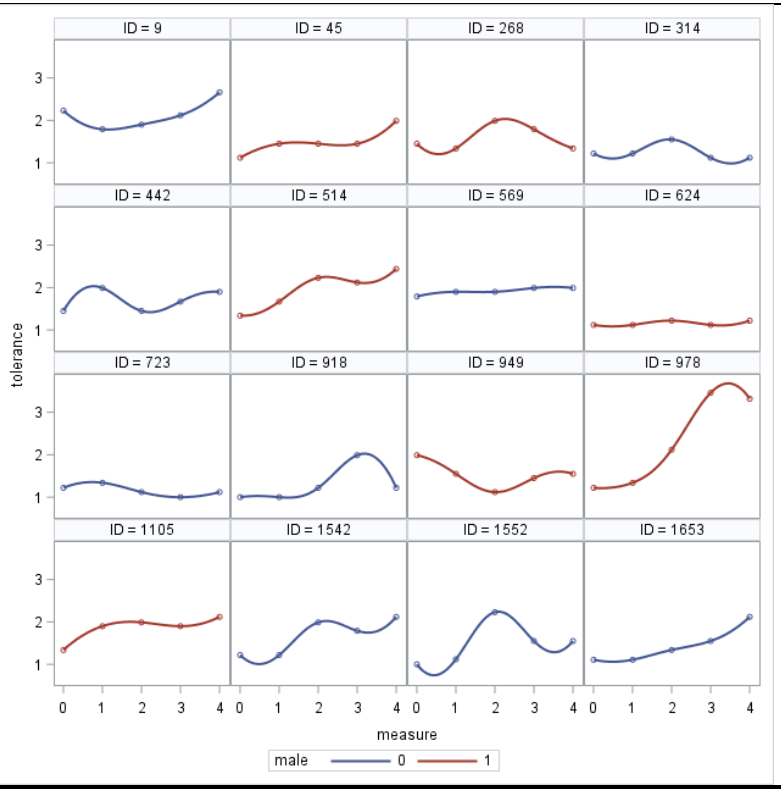
```
* PROC SGPLOT: REG;
proc sgplot data=long;
     reg x=measure y=tolerance
/group=male;
     where obs in (1 2);
     xaxis label="Measurement";
     yaxis label="Tolerance level";
     format male malefmt.;
run;
```




```
* PROC SGPLOT: PBSPLINE;
proc sgplot data=long;
     pbspline x=measure y=tolerance
/group=male;
     where obs in (1 2);
     xaxis label="Measurement";
     yaxis label="Tolerance level";
     format male malefmt.;
run;
```



```
* PROC SGPANEL: Individual;
proc sgpanel data=long;
     panelby ID / columns=4 rows=4;
     pbspline x=measure y=tolerance /
group=male;
run;
```



Chapter 11. Exporting Datasets

11.1. Exporting Dataset: PROC EXPORT

General Syntax

```
proc export data=dataset outfile='file-name'
            dbms= identifier replace;
run;
```

- DBMS identifier
  - Comma-delimited (.csv): CSV
  - Excel (.xlsx): XLSX
  - Tab-delimited (.txt): TAB
- Export wizard: File → Export Data

11.2 Export Using ODS

General Syntax

```
* CSV;
ods csv file="file-name";
     Any procedure
ods csv close;

* HTML, XLS;
ods html file="file-name";
     Any procedure
ods html close;
```

Chapter 12. Hypothesis Testing: Test for Mean(s)

12.1. Hypothesis Testing

- A method for using *sample* data (statistic) to decide between two competing claims (hypotheses) about a *population* characteristic (parameter) to answer research questions.
- Null hypothesis (H_0): Hypothesis to be tested. Usually "Nothing happened/new".
- Alternative hypothesis (H_1): Our question of interest. Usually "Something happened/new".
- 2 decisions: Reject H_0 / Fail to reject H_0 (NEVER say 'accept H_0')
- Possible outcomes in hypothesis testing

		Truth	
		H_0	H_1
Decision	Fail to reject H_0	☺	Type II error
	Reject H_0	Type I error	☺

- Significance level: *Pre-specified threshold* of type I error rate. (i.e. tolerance of type I error.) Often denoted by α.
- *p*-value: Probability of type I error *observed* in the sample.
- Test procedure
  - Explicitly define the population parameter of interest.
  - Clarify the null and alternative hypotheses.
  - Determine the significance level of the test.
  - Consider any necessary assumptions. (e.g. distribution, parameters)
  - State the form of test statistic and its distribution under H_0.
  - Set the decision rule and compute the *p*-value.
  - Make a conclusion in the context of the problem.

12.2. One-Sample Test for a Mean

- Test a hypothesis on a specific value of the population mean μ (e.g. H_0: μ= μ_0).

One-sample z-test (H_0: μ= μ_0)	$Z = \frac{\bar{X} - \mu_0}{\sigma/\sqrt{n}} \sim N(0,1)$
One-sample t-test (H_0: μ= μ_0)	$t = \frac{\bar{X} - \mu_0}{s/\sqrt{n}} \sim t_{n-1}$

- Z-test
  - When the population variance is *known*.
  - Population distribution is normal or sample size is large. (e.g. n≥30)
  - Rejection region

H_1	Rejection region	*p*-value
$\mu \neq \mu_0$	$\lvert z\rvert > z_{\alpha/2}$	$P(\lvert Z\rvert > \lvert z\rvert \lvert H_0)$
$\mu > \mu_0$	$z > z_{\alpha}$	$P(Z > z \lvert H_0)$
$\mu < \mu_0$	$z < -z_{\alpha}$	$P(Z < z \lvert H_0)$

where $P(Z > z_\alpha) = \alpha$ with $Z \sim N(0,1)$.

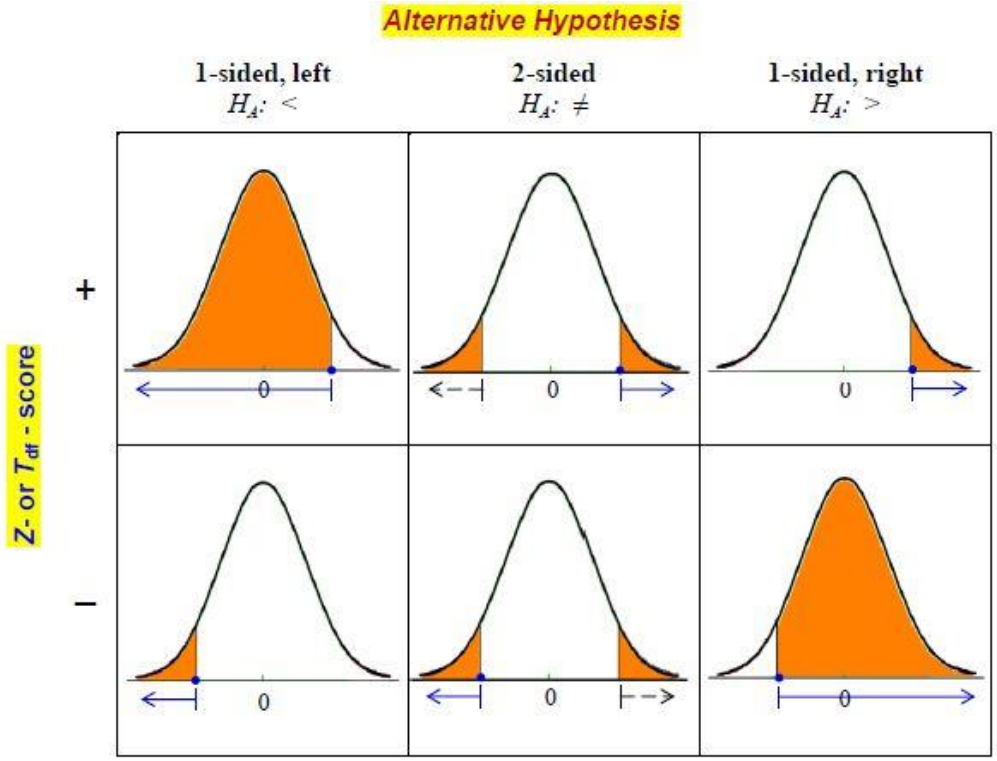
- t-test
  - When the population variance is *unknown*.
  - Use *s* (sample standard deviation) instead of σ.

$$s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}}$$

  - Degrees of freedom (*df*): Number of scores in a sample that are free to vary
  - t_{df}: t-distribution with degrees of freedom *df*. Generally, more spread out than a standard normal curves. (As $df \to \infty$, $t_{df} = N(0,1)$.)
  - Population distribution needs to be normal, but the test is robust.
  - Rejection region

H_1	Rejection region	*p*-value
$\mu \neq \mu_0$	$\lvert t\rvert > t_{n-1,\alpha/2}$	$P(\lvert T\rvert > \lvert t\rvert \lvert H_0)$
$\mu > \mu_0$	$t > t_{n-1,\alpha}$	$P(T > t \lvert H_0)$
$\mu < \mu_0$	$t < -t_{n-1,\alpha}$	$P(T < t \lvert H_0)$

where $P\left(T > t_{n-1,\alpha}\right) = \alpha$ with $T \sim t_{n-1}$.


Alternative Hypothesis
1-sided, left
H_A: <
2-sided
H_A: ≠
1-sided, right
H_A: >
Z- or T_{df} - score
+
–
0
0
0
0
0
0

12.3. Independent Two-Sample Test for Means

- Test a hypothesis to compare two means μ_1 and μ_2. (e.g. H_0: $\mu_1 = \mu_2$)
- Comparison between two samples to see if they are truly different.
- Two samples must be independent and randomly selected.
- Z-test: Each sample size must be at least 30 or, if not, each population must have a normal distribution with a *known* standard deviation.
- Otherwise (i.e. unknown deviation), use t-test.
- F-test of equal variances (Homoscedasticity)
  - $H_0: \sigma_1^2 = \sigma_2^2$ vs $H_1: \sigma_1^2 \neq \sigma_2^2$
  - Under H_0,

$$F = \left. \frac{\frac{s_1^2}{\sigma_1^2}}{\frac{s_2^2}{\sigma_2^2}} \right/ = \frac{s_1^2}{s_2^2} \sim F_{n_1-1,\ n_2-1}.$$

  - Reject H_0 if $F > F_{n_1-1,n_2-1,\alpha/2}$ or $F < F_{n_1-1,n_2-1,1-\alpha/2}$ where $P\left(F > F_{n_1-1,n_2-1,\alpha}\right) = \alpha$.

Two-sample z-test[3] (H_0: $\mu_1 = \mu_2$)		$Z = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \sim N(0,1)$
Two-sample t-test (H_0: $\mu_1 = \mu_2$)	Equal variances	$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t_{n_1+n_2-2}$ $s_p = \sqrt{\frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2}}$
	Unequal variances	$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \sim t_{df}$ $df = \frac{(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2})^2}{\frac{(s_1^2/n_1)^2}{(n_1-1)} + \frac{(s_2^2/n_2)^2}{(n_2-1)}}$

[3] When the samples are large, you may use $\sigma_1 \approx s_1$ and $\sigma_2 \approx s_2$.

12.4. Paired Two-Sample t-test

- Measurements are made on the same subject rather than on two different individuals.
  - Before or after studies (e.g. pre-treatment vs post-treatment)
  - Matched case-control studies
  - Cross-over studies
- The *differences* should be normally distributed. If the distribution deviates from the normal in only minor way, then the t-distribution can still be used.

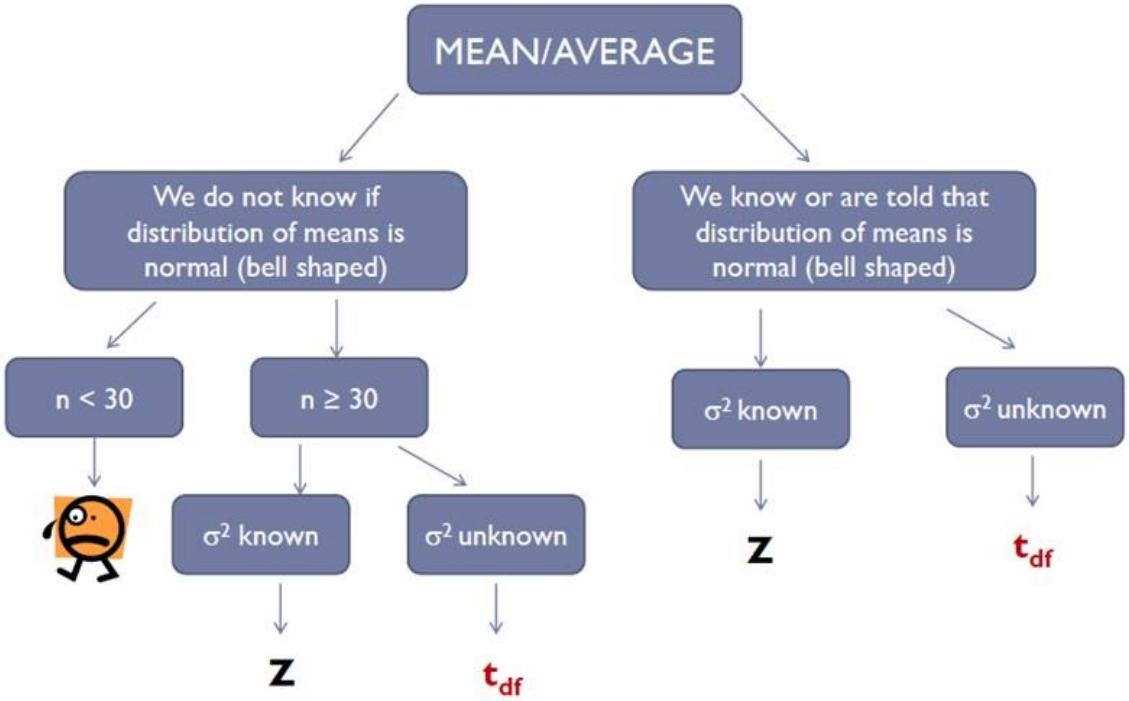
Paired Two-sample t-test
(H_0: $\Delta \equiv \mu_2 - \mu_1 = 0$)

$$t = \frac{\bar{d} - \Delta}{s_d / \sqrt{n}} \sim t_{n-1}$$

where $\bar{d}$ and s_d^2 are the sample mean and variance of the differences.

H_1	Rejection region	*p*-value
$\Delta \neq 0$	$\lvert t \rvert > t_{n-1,\alpha/2}$	$P(\lvert T \rvert > \lvert t \rvert \lvert H_0)$
$\Delta > 0$	$t > t_{n-1,\alpha}$	$P(T > t \lvert H_0)$
$\Delta < 0$	$t < -t_{n-1,\alpha}$	$P(T < t \lvert H_0)$

where $P(T > t_{n-1,\alpha}) = \alpha$ with $T \sim t_{n-1}$.



The central limit theorem (CLT) helps us to assume normality in samples of 30 or more

12.5. PROC TTEST

- Checking normality (H_0: The distribution is normal.)[4]

General Syntax

```
proc univariate data=dataset normal;
     var variable-name;
     qqplot variable-name;
run;
```

- One-sample t-test (H_0: $\mu = \mu_0$)

General Syntax

```
proc ttest data=dataset h0=nullvalue
           alpha=significancelevel sides=sides nobyvar;
     var variable-name;
run;
```

Option	Description
H0 = *n*	Null value (μ_0). Default is 0.
ALPHA = *n*	Significance level. Default is 0.05.
NOBYVAR	Moves the names of the variables from the title to the output table.
SIDES = `type`	Specify the alternative hypothesis. Default is two-sided (≠). SIDES = 2 (≠), L (<), U (>)

[4] In order to guarantee the normality, the QQPLOT should be (approximately) a straight line and the normality tests should generate non-significant *p*-values.

- Independent Two-sample t-test (H_0: $\mu_1 = \mu_2$)
  - Check equal variances: Part of the output when using a CLASS statement

General Syntax

```
proc ttest data=dataset;
     class group_variable;
     var variable_name;
run;
```

- Paired two-sample t-test (H_0: $\mu_1 = \mu_2$)

General Syntax

```
proc ttest data=dataset;
     paired after * before;
run;

proc ttest data=dataset;
     var diff; * diff = after - before;
run;
```

- Graphics with PROC TTEST

General Syntax

```
proc ttest data=dataset plots(ONLY)=(list-of-plot-requests);
     var variable-name;
run;
```

  - By default, the QQPLOT and SUMMARYPLOT plots are generated.
  - If you choose specific plots in the plot-list, the default plots will still be created unless you add the ONLY global option.

Request	Description
ALL	Request all appropriate plots.
BOXPLOT	Create boxplots.
HISTOGRAM	Create histograms overlaid with normal and kernel density curves.
INTERVALPLOT	Create plots of confidence interval of means.
NONE	Suppress all plots.
QQPLOT	Create a normal quantile-quantile (Q-Q) plot.
SUMMARYPLOT	Create one plot that includes both histograms and boxplots.
AGREEMENTPLOT*	Create agreement plots.
PROFILESPLOT*	Create a profiles plot.

* Available for paired t-test

Example

Raw Data

Obs	obs	Gender	Type	Agegroup	White blood cell	Red blood cell	Cholesterol
1	1	Female	AB	Young	7710	7.40	258
2	2	Male	AB	Old	6560	4.70	.
3	3	Male	A	Young	5690	7.53	184
4	4	Male	B	Old	6680	6.85	.
5	5	Male	A	Young	.	7.72	187

SAS Code

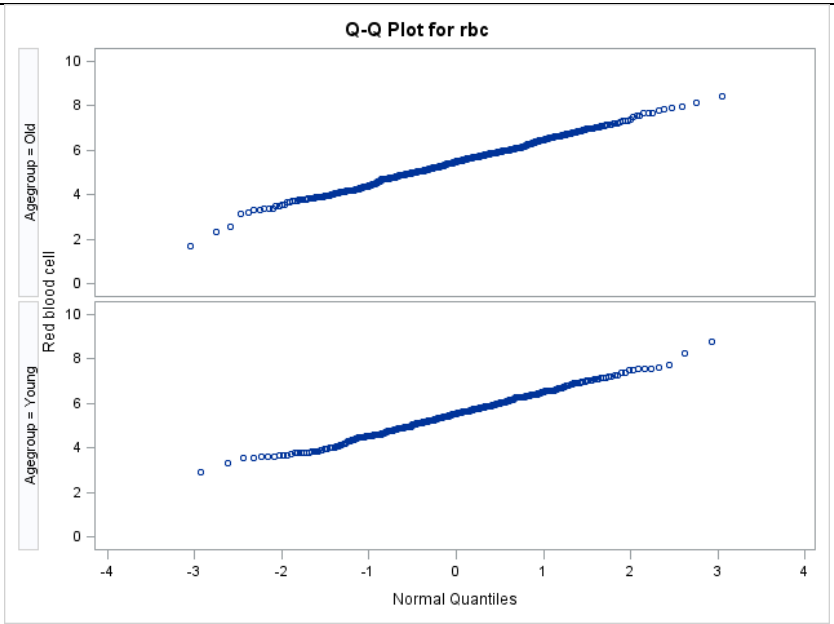
```
/* One-sample t-test */

* Checking normality;
proc univariate data=blood normal;
     var rbc;
     qqplot rbc;
     histogram rbc;
run;

* One-sample t-test;
proc ttest data=blood h0=5.4;
     var rbc;
run;
```

Output

Tests for Normality				
Test	Statistic		p Value	
Shapiro-Wilk	W	0.998442	Pr < W	0.5950
Kolmogorov-Smirnov	D	0.01809	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.038034	Pr > W-Sq	>0.2500
Anderson-Darling	A-Sq	0.30757	Pr > A-Sq	>0.2500

DF	t Value	Pr > |t|
915	2.57	0.0104

```
/* Independent two-sample t-test */

* Checking normality;
proc univariate data=blood normal;
      class agegroup;
      var rbc;
      qqplot rbc;
      histogram rbc;
run;

* Independent two-sample t-test;
proc ttest data=blood;
      class agegroup;
      var rbc;
run;
```



Method	Variances	DF	t Value	Pr > |t|
Pooled	Equal	914	-0.97	0.3310
Satterthwaite	Unequal	787.77	-0.98	0.3296

Equality of Variances				
Method	Num DF	Den DF	F Value	Pr > F
Folded F	550	364	1.03	0.7554

Example

Raw Data

```
data weight;
input wbefore wafter @@;
cards;
 200 185 175 154
 188 176 198 193
 197 198 310 275
 245 224 202 188
 ;
run;

data weight;
     set weight;
     diff = wafter - wbefore;
run;
```

Obs	wbefore	wafter	diff
1	200	185	-15
2	175	154	-21
3	188	176	-12
4	198	193	-5
5	197	198	1
6	310	275	-35
7	245	224	-21
8	202	188	-14

SAS Code

```
/* Paired two-sample t-test */

* Paired two-sample t-test;
proc ttest data=weight;
     paired wafter*wbefore;
run;


* diff = after - before;
proc ttest data=weight;
     var diff;
run;
```

Output

DF	t Value	Pr > |t|
7	-3.94	0.0056

DF	t Value	Pr > |t|
7	-3.94	0.0056

12.6. Analysis of Variance (ANOVA)

- Compare means for multiple (usually ≥ 3) independent populations

  (H_0: $\mu_1 = \mu_2 = \ldots = \mu_k$, $k \geq 3$; H_1: Not H_0)

  - One-way ANOVA: There is only one way to classify the populations of interest.

    (e.g. Compare the effect of three different treatments.)

  - Two-way ANOVA: There is more than one way to classify the populations.

    (i.e. Does the effect due to one factor change as the level of another factor changes?)

- ANOVA assumes *k* independent, equivariant, normally-distributed groups.

Assumption	Description	Check
Independence	The *k* groups are independent and randomly selected.	Usually assumed or given.
Normality	The *k* groups are normally distributed.	Histograms, boxplots, Q-Q plots Shapiro-Wilk[5] or Kolmogorov-Smirnov test
Homoscedasticity	The population variance is the same in each of the *k* groups.	Plot residuals vs fitted values (random) Levene's or Brown-Forsythe's test

[5] If the sample size is ≤ 2000, PROC UNIVARIATE with the NORMAL option computes the Shapiro-Wilk statistic.

- ANOVA is *robust*, so minor violations in these assumptions will not invalidate your analysis.
- Checking homoscedasticity (MEANS statement HOVTEST= option)

Value	Description
BARTLETT	Bartlett's test
BF	Brown and Forsythe's variation of Levene's test
LEVENE	Levene's test
OBRIEN	O'Brien's test

- Assess magnitude of variation attributable to specific sources.
- Partition the total variation according to the source (variability within/between groups)
- Extension of two-sample t-test to multiple groups.

Example: Religion and mean age of getting married	
t-test	Muslims vs Christians
ANOVA	Muslims vs Christians vs Catholic

- Two-way ANOVA
  - Extension of the one-way ANOVA
  - Two independent variables (factors), each with its own levels.
  - The groups must have the same sample size.
  - Null hypothesis

1	The population means of the first factor are equal.
2	The population means of the second factor are equal.
3	There is no interaction between the two factors.

- Sum of Squares (SS)

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk}$$

  - Two-way model where α is the treatment effect of interest.
  - RSS: Residual Sum of Squares
  - Different types of analysis (I, II, and III) assess the treatment effect differently.

Type	Treatment effect	Description
Type I	$SS_{trt}(I) = RSS(\mu) - RSS(\mu, \alpha)$	Add each effect sequentially
Type II	$SS_{trt}(II) = RSS(\mu, \beta) - RSS(\mu, \alpha, \beta)$	Ignore interaction
Type III	$SS_{trt}(III) = RSS(\mu, \beta, \gamma) - RSS(\mu, \alpha, \beta, \gamma)$	Consider interaction

  - When the interaction exists, it is more appropriate to use Type III analysis to assess the treatment effect.
  - If there is no interaction, then Type II is statistically more powerful than Type III.
  - If unbalanced sample sizes, then use Type II.
  - If the interaction is significant, the main effects should not be further analyzed.
  - For a completely balanced design, Type I, II, and III all give the same results.

12.7. PROC ANOVA

General Syntax

```
proc anova data=dataset;
     class list-of-group-variables;
     model numeric-variable = categorical-variable;
     means effects / <options>;
run; quit;
```

- CLASS: Come before the MODEL statement to define the classification variables.
- MODEL: Define the dependent variable (numeric) and the effects (categorical).
- MEANS (optional): Calculate means of the dependent variable for any of the main effects in the MODEL statement. Perform several types of multiple comparison tests.

12.8. PROC GLM

General Syntax

```
proc glm data = dataset;
     class list-of-group-variables;
     model numeric-variable = categorical-variable;
     means effects / <options>;
     contrast "contrast-title" group-variable contrast-values;
     estimate "contrast-title" group-variable contrast-values;
run; quit;
```

- CLASS: Come before the MODEL statement to define the classification variables.
- MODEL: Define the dependent variable (numeric) and the effects (categorical).
- MEANS (optional): Calculate means of the dependent variable for any of the main effects in the MODEL statement. Perform several types of multiple comparison tests.

- CONTRAST/ESTIMATE
  - Test the significance of a specific comparison between levels of a treatment factor that you are particularly interested in.
  - Any level not involved in the contrast takes a value of 0.

Example: A 4-level factor (A, B, C, D)

Contrast				Description
A	B	C	D	
-1	1	0	0	Difference between A and B
0	0	-1	1	Difference between C and D
-1	-1	1	1	Difference between average of A & B and that of C & D
-2	1	1	0	Difference between A and average of B & C
-3	1	1	1	Difference between A and average of B, C & D

Example

Raw Data

Obs	obs	Gender	Type	Agegroup	White blood cell	Red blood cell	Cholesterol
1	1	Female	AB	Young	7710	7.40	258
2	2	Male	AB	Old	6560	4.70	.
3	3	Male	A	Young	5690	7.53	184
4	4	Male	B	Old	6680	6.85	.
5	5	Male	A	Young	.	7.72	187

SAS Code

```
* One-way ANOVA;
proc anova data=blood;
      class type;
      model chol = type;
      means type / hovtest=bf;
run; quit;

proc glm data = blood;
      class type;
      model chol = type;
      means type/ hovtest=bf;
run; quit;
```

Output

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	18699.392	6233.131	2.52	0.0569
Error	791	1957316.022	2474.483		
Corrected Total	794	1976015.414			

Brown and Forsythe's Test for Homogeneity of chol Variance
ANOVA of Absolute Deviations from Group Medians

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Type	3	1673.7	557.9	0.64	0.5882
Error	791	687501	869.2		

Source	DF	Type III SS	Mean Square	F Value	Pr > F
Type	3	18699.39185	6233.13062	2.52	0.0569

```
* Two-way ANOVA with interaction;
proc anova data=blood;
class type agegroup;
model chol = agegroup type
agegroup*type;
run; quit;

proc glm data = blood;
class type agegroup;
model chol = agegroup type
agegroup*type / ss1 ss2 ss3;
run; quit;
```

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	7	42924.205	6132.029	2.50	0.0153
Error	787	1933091.209	2456.279		
Corrected Total	794	1976015.414			

Source	DF	Type I SS	Mean Square	F Value	Pr > F
Agegroup	1	17228.83363	17228.83363	7.01	0.0082
Type	3	19692.89109	6564.29703	2.67	0.0464
Type*Agegroup	3	6002.47979	2000.82660	0.81	0.4860

Source	DF	Type II SS	Mean Square	F Value	Pr > F
Agegroup	1	18222.33287	18222.33287	7.42	0.0066
Type	3	19692.89109	6564.29703	2.67	0.0464
Type*Agegroup	3	6002.47979	2000.82660	0.81	0.4860

Source	DF	Type III SS	Mean Square	F Value	Pr > F
Agegroup	1	5715.72361	5715.72361	2.33	0.1275
Type	3	22747.61527	7582.53842	3.09	0.0266
Type*Agegroup	3	6002.47979	2000.82660	0.81	0.4860

12.9. Multiple Comparisons

- After showing that at least one pair of the group means are significantly different, we may proceed to make individual comparisons.

  e.g. Conduct two-sample t-tests to test for a difference in means for each pair of groups.
- Performing individual comparisons requires multiple hypothesis tests.
- Using α=0.05 for each test will lead the overall Type I error to be elevated above 5%.
- For *n* independent tests, the probability of making a Type I error at least once is $1-0.95^n$.
- Consider multiple comparison procedures to preserve the overall significance level (α).
- Options for MEANS statement

Type	Option	Description
Bonferroni	BON	General and simple but conservative; Good for a few comparisons.
Sidak	SIDAK	Less conservative than Bonferroni for normally distributed statistics.
Tukey's HSD	TUKEY	Most powerful for pairwise comparison.
Dunnett	DUNNETT	Most powerful test for comparisons with control (reference).
Scheffe	SCHEFFE	Exact simultaneous for all contrasts. (Unplanned comparisons)

12.10. Parametric vs Nonparametric Tests for Mean(s)/Median(s)[6]

- If distributional assumptions are not satisfied,
  - Try transformations of the data. (e.g. log, square root)
  - Try nonparametric approaches.
- Parametric tests include assumptions on the underlying distribution of the observed data, while nonparametric tests do not require any assumptions.
- Parametric tests tend to have higher statistical power (ability to find an effect, when there actually exists one) if the assumptions are satisfied.

		Parametric (means)	Nonparametric (medians)
1-sample		1-sample z- (t-) test	One-sample Wilcoxon signed rank test
2-samples	Independent	2-sample z- (t-) test	Mann-Whitney test Kolmogorov-Smirnov test
	Paired	Paired t-test	Paired Wilcoxon signed rank test
3+ sample		ANOVA	Kruskal-Wallis test (One-way) Friedman test (Two-way)

[6] http://www.originlab.com/index.aspx?go=Products/Origin/Statistics/NonparametricTests
http://changingminds.org/explanations/research/analysis/parametric_non-parametric.htm

Chapter 13. Categorical Data Analysis

13.1. Statistical Model for Different Types of Variables[7]

Y	–	X
Response	–	Explanatory
Dependent	–	Independent
Outcome	–	Predictor

Y	X	Test
Nominal	Nominal	χ^2- test
Continuous	Binary	t- test
	Nominal	One-way ANOVA
	Mixed	One-way ANCOVA
Continuous	1 Continuous	Bivariate correlation Simple regression
	2+ Mixed	Multiple regression
Binary	Mixed	Logistic regression

[7] http://stats.idre.ucla.edu/other/mult-pkg/whatstat/

Examples

Example 1: Family history – Aminotransferase (ALT) level

X (Family history)	–	Y (ALT)	
Binary	–	Continuous	→ t-test

Example 2: Age – ALT level

	X (Age)	–	Y (ALT)	
a)	Continuous		Continuous	→ Correlation, Regression
b)	Categorical		Continuous	→ ANOVA

Example 3: Gender – Family history

X (Gender)	Y (Family history) Yes	No	
Male	a	b	
Female	c	d	→ χ^2- test

13.2. One-Sample Test for Binary Proportion

- H_0: $p = p_0$ vs H_1: $p \neq p_0$
- Test statistic: By the normal approximation from $X \sim Bin(n, p)$ with $np \geq 5$ and $n(1-p) \geq 5$,

$$Z = \frac{\hat{p} - p_0}{\sqrt{p_0(1-p_0)/n}} \sim N(0,1) \text{ under } H_0.$$

- Rejection region

H_1	Rejection region	p-value
$p \neq p_0$	$\lvert z \rvert > z_{\alpha/2}$	$P(\lvert Z \rvert > \lvert z \rvert \lvert H_0)$
$p > p_0$	$z > z_{\alpha}$	$P(Z > z \lvert H_0)$
$p < p_0$	$z < -z_{\alpha}$	$P(Z < z \lvert H_0)$

where $P(Z > z_\alpha) = \alpha$ with $Z \sim N(0,1)$.

- PROC FREQ

General Syntax

```
proc freq data=dataset;
     table binary-variable / binomial(p = binary-proportion);
run;
```

13.3. Test of Independence

- Contingency table (R X C)
  - Display data that can be classified by two different (categorical) variables.
  - Each cell represents the number of units with a specific value for each of the two variables.
  - n_{ij}: The number of units in the cell (i, j), $i = 1, 2, \cdots, R$ and $j = 1, 2, \cdots, C$

    (i-th row; j-th column)

X	Y				Total
	1	2	…	C	
1	n_{11}	n_{12}	…	n_{1C}	$n_{1\cdot}$
2	n_{21}	n_{22}	…	n_{2C}	$n_{2\cdot}$
⋮	⋮	⋮	n_{ij}	⋮	⋮
R	n_{R1}	n_{R2}	…	n_{RC}	$n_{R\cdot}$
Total	$n_{\cdot 1}$	$n_{\cdot 2}$	…	$n_{\cdot C}$	n

- Hypothesis testing: Relationship (*association*) between two variables
  - H_0: Variable X is *not* associated with Variable Y.

    (i.e. Variable X and Variable Y are independent.)
  - H_1: Variable X and Variable Y are associated.

    (i.e. Variable X and variable Y are not independent.)

H_0:
$$\begin{array}{ccccccc} p_{11} & = & p_{12} & = & \cdots & = & p_{1C} \\ p_{21} & = & p_{22} & = & \cdots & = & p_{2C} \\ & & \vdots & & & & \\ p_{R1} & = & p_{R2} & = & \cdots & = & p_{RC} \end{array}$$

H_1: $p_{ij} \neq p_{ik}$ for some $i = 1, 2, \cdots, R; j, k = 1, 2, \cdots, C; j \neq k$

where $p_{ij} = P(Y = j \mid X = i)$, $i = 1, 2, \cdots, R$ and $j = 1, 2, \cdots, C$.

- χ^2- test (Chi-squared test)
  - O_{ij}: Observed number of units in the cell (i, j)
  - E_{ij}: Expected number of units in the cell (i, j) under H_0

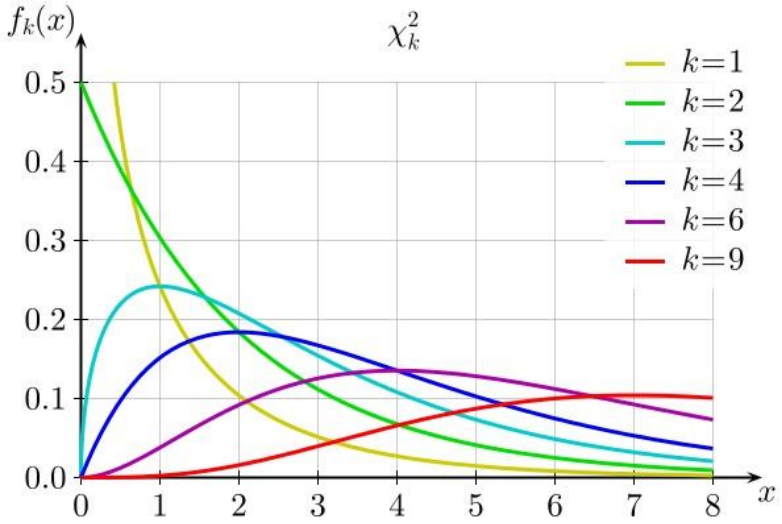
$$E_{ij} = \frac{n_{i\cdot} \times n_{\cdot j}}{n}, i = 1, 2, \cdots R; j = 1,2, \cdots, C$$

  - Test statistic

$$\chi^2 = \sum_{\forall i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \sim \chi^2_{(R-1)(C-1)} \text{ under } H_0.$$

  - Reject H_0 if $\chi^2 > \chi^2_{(R-1)(C-1),\alpha}$ where $P\left(\mathrm{X} > \chi^2_{(R-1)(C-1),\alpha}\right) = \alpha;\ \ \mathrm{X} \sim \chi^2_{(R-1)(C-1)}$.

- χ^2- distribution



- Fisher's exact test
  - In order to use χ^2- test,

    a) No more than 1/5 of the cells have expected values <5.

    b) No cell has expected value <1.
  - If not, use Fisher's exact test instead.

- McNemar's test
  - The χ^2- test relies on data that consist of *independent* observations.
  - Data cannot be repeated measures for same subject or matched pairs.
  - McNemar's test: Two-sample test for binomial proportions for *matched-pair* data
  - Test statistic

$$\chi^2 = \frac{(n_{12}-n_{21})^2}{n_{12}+n_{21}} \sim \chi_1^2 \text{ under } H_0.$$

  - With continuity correction,

$$\chi^2 = \frac{(|n_{12}-n_{21}|-0.5)^2}{n_{12}+n_{21}} \sim \chi_1^2 \text{ under } H_0.$$

  - Reject H_0 if $\chi^2 > \chi_{1,\alpha}^2$ where $P\left(\mathrm{X} > \chi_{1,\alpha}^2\right) = \alpha$; $\mathrm{X} \sim \chi_1^2$.
  - Use this test only if $n_{12} + n_{21} \geq 20$.

- PROC FREQ

General Syntax

```
proc freq data=dataset;
     table variable-combinations / <options>;
run;
```

Option	Description
CHISQ	Conduct χ^2- test of independence.
EXACT (FISHER)	Conduct Fisher's exact test for tables.
AGREE	Conduct McNemar's test for matched pair analysis.
CMH	Conduct Cochran-Mantel-Haenszel test for stratified two-way table. (i.e. χ^2- test for 2X2Xk(strata))
TREND	Conduct Cochran-Armitage trend test for proportions.
RELRISK	Request relative risk measures for 2X2 tables.
RISKDIFF	Provide differences in risk measures.
EXPECTED	Produce expected values in the frequency table.
CL	Produce confidence limits for measures of association.

Example

Raw Data

Obs	obs	Gender	Type	Agegroup	White blood cell	Red blood cell	Cholesterol
1	1	Female	AB	Young	7710	7.40	258
2	2	Male	AB	Old	6560	4.70	.
3	3	Male	A	Young	5690	7.53	184
4	4	Male	B	Old	6680	6.85	.
5	5	Male	A	Young	.	7.72	187

SAS Code

```
* One-sample test for binary
proportion;
proc freq data=blood;
table gender / binomial(p=.3);
* H0: p=0.3;
run;
```

Output

Test of H0: Proportion = 0.3	
ASE under H0	0.0145
Z	9.6609
One-sided Pr > Z	<.0001
Two-sided Pr > |Z|	<.0001

SAS Code

```
* 2 X 2 Test of Independence;
proc freq data=blood;
table gender * agegroup / chisq exact;
run;
```

Output

Statistic	DF	Value	Prob
Chi-Square	1	0.4426	0.5059
Likelihood Ratio Chi-Square	1	0.4423	0.5060
Continuity Adj. Chi-Square	1	0.3603	0.5483
Mantel-Haenszel Chi-Square	1	0.4421	0.5061
Phi Coefficient		-0.0210	
Contingency Coefficient		0.0210	
Cramer's V		-0.0210	

Fisher's Exact Test	
Cell (1,1) Frequency (F)	258
Left-sided Pr <= F	0.2741
Right-sided Pr >= F	0.7674
Table Probability (P)	0.0415
Two-sided Pr <= P	0.5164

```
* 2 X 3 Test of Independence;
proc freq data=blood;
table gender * type / chisq exact;
run;
```

Statistic	DF	Value	Prob
Chi-Square	3	4.0865	0.2523
Likelihood Ratio Chi-Square	3	4.1389	0.2469
Mantel-Haenszel Chi-Square	1	0.5828	0.4452
Phi Coefficient		0.0639	
Contingency Coefficient		0.0638	
Cramer's V		0.0639	

Fisher's Exact Test	
Table Probability (P)	<.0001
Pr <= P	0.2516

Example

Raw Data

```
data Paired;
     input Dr1 $ Dr2 $ Count;
     cards;
Good Good 526
Good Bad 95
Bad Good 515
Bad Bad 106
;
run;
```

Table of Dr1 by Dr2			
	Dr2		
Dr1	Bad	Good	Total
Bad	106	515	621
Good	95	526	621
Total	201	1041	1242

SAS Code

```
* McNemar's Test;
* AGREE: Paired sample;
proc freq data=Paired;
     tables Dr1*Dr2 / agree;
     weight count;
run;
```

Output

McNemar's Test	
Statistic (S)	289.1803
DF	1
Pr > S	<.0001

Chapter 14. Correlation and Linear Regression

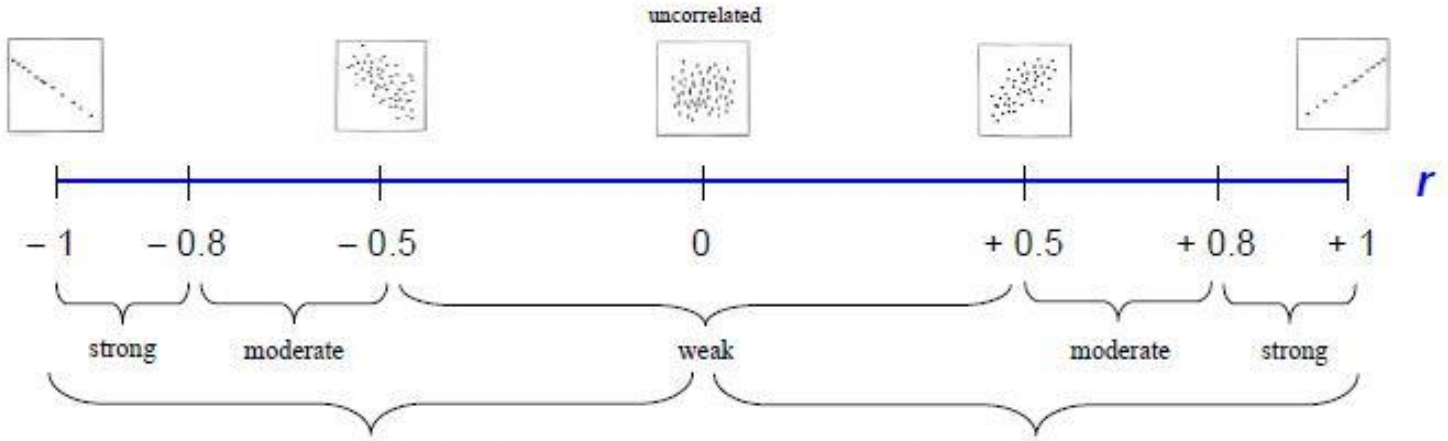
14.1. Correlation

- Measure of *association* between two numeric variables
- NOT about predicting one variable from another, but rather about investigating whether there is a relationship between the two variables. (cf. Association vs Causation)
- Correlation coefficient
  - Take values from -1 (perfectly negative) to +1 (perfectly positive).
  - The larger the *absolute* value is, the stronger the relationship is.
  - Sensitive to outliers
  - Should not be extrapolated beyond the range of observed values of X and Y as the relationship between the two variables may change outside this range.
  - A high correlation coefficient does not imply a causal relationship.

- Pearson's correlation coefficient
  - Measure of the *linear* correlation (dependence) between two variables X and Y
  - Sensitive only to a linear relationship between the two variables

Population	$\rho = \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} = \frac{E[(X-E(X))(Y-E(Y))]}{\sqrt{Var(X)}\sqrt{Var(Y)}}$
Sample	$r = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$

- Spearman's rank coefficient (Nonparametric)
  - Assess how well the relationship between two variables can be described using a monotonic function.
  - A perfect Spearman of +1 or -1 occurs when each of the variables is a perfect monotone function of the other.
- Kendall's τ coefficient (Nonparametric)
  - If the rankings of two variables are exactly the same (reverse), then τ = 1 (-1).
  - If the two variables are independent, then one can expect τ ≈ 0.



- Some important exceptions to the "typical" cases above:

$r = 0$, but X and Y are correlated, *non*linearly

$r > 0$ in each of the two individual subgroups, but $r < 0$ when combined

$r > 0$, only due to the effect of one *influential* outlier; if removed, then data are uncorrelated ($r = 0$)

- PROC CORR

General Syntax

```
proc corr data=dataset <nonparametric-options> plots = <plot-request>;
     var list-of-variables;
     with list-of-variables;
run;
```

- By default, compute Pearson's correlation coefficient.
- Nonparametric options: SPEARMAN, KENDALL, HOEFFDING
- Plot requests

Request	Description
SCATTER	Create scatterplots for pairs of variables. Prediction or confidence ellipses are overlaid on the plot. (ELLIPSE= PREDICTION, CONFIDENCE, or NONE)
MATRIX	Create a matrix of scatterplots for all variables.

Example

Raw Data

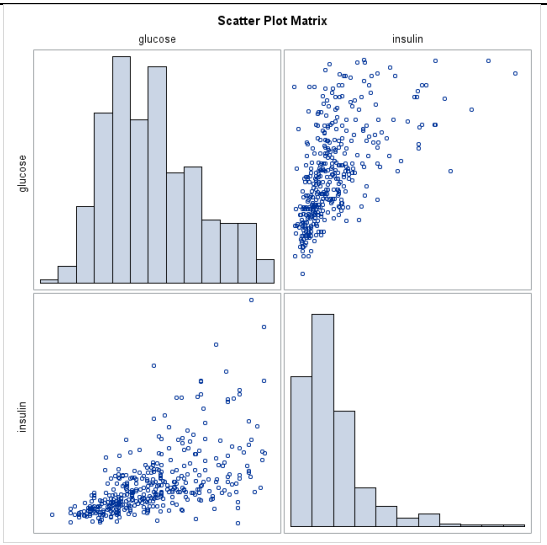
Obs	id	pregnant	glucose	blood	triceps	insulin	bmi	pedigree	age	test
1	1	1	89	66	23	94	28.1	0.167	21	Negative
2	2	0	137	40	35	168	43.1	2.288	33	Positive
3	3	3	78	50	32	88	31	0.248	26	Positive
4	4	2	197	70	45	543	30.5	0.158	53	Positive
5	5	1	189	60	23	846	30.1	0.398	59	Positive

SAS Code

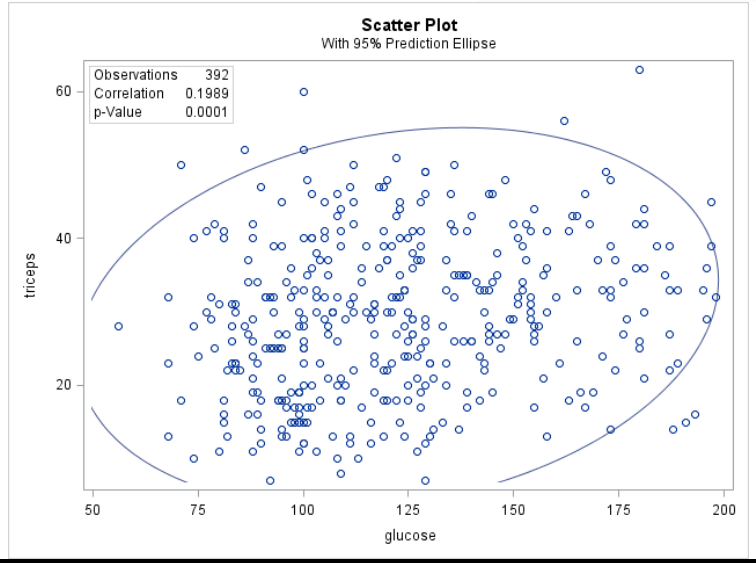
```
proc corr data=pima plots=matrix(histogram);
var glucose insulin;
run;

proc corr data=pima plots=scatter(ellipse=prediction) spearman;
     var glucose insulin;
     with triceps;
run;
```

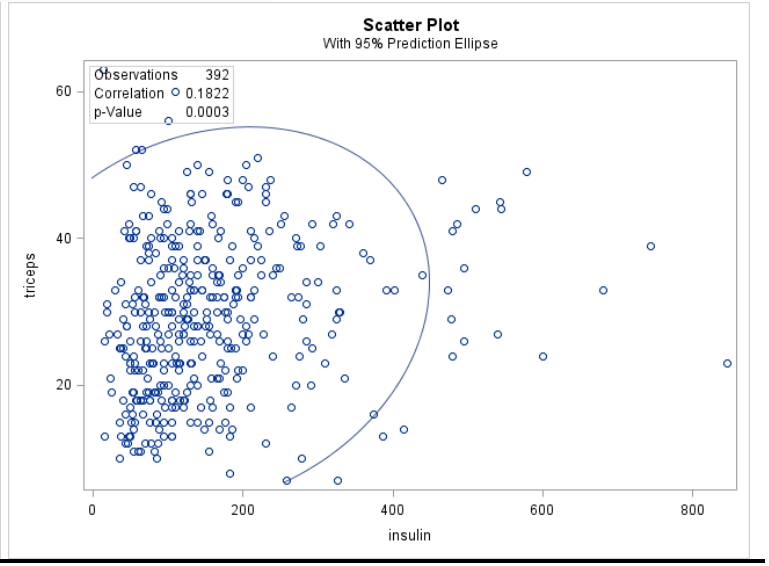
Output

Pearson Correlation Coefficients, N = 392 Prob > |r| under H0: Rho=0	glucose	insulin
glucose glucose	1.00000	0.58122 <.0001
insulin insulin	0.58122 <.0001	1.00000



Spearman Correlation Coefficients, N = 392 Prob > |r| under H0: Rho=0		
	glucose	insulin
triceps	0.21584	0.24114
triceps	<.0001	<.0001





14.2. Linear Regression

- Linear relationship

$$Y = \alpha + \beta\, x$$

  - Slope (β): Change in *y* when *x* changes by one unit
  - Intercept (α): Where the line crosses the *y*-axis.
  - A linear relationship is the simplest non-trivial relationship that can be imagined.
  - Appropriate if the "true" relationship between X and Y are linear. (cf. Transformation)

- Simple linear regression

$$\begin{array}{ccccc} Y & = & \alpha + \beta\, x & + & \varepsilon \\ \text{(Response)} & = & \text{(Linear Model)} & + & \text{(Error)} \end{array}$$

  where $\varepsilon \sim N(0, \sigma^2)$.

  - Fitted model: $\hat{Y} = \hat{\alpha} + \hat{\beta}\, x$ where $\hat{\alpha}$ and $\hat{\beta}$ are sample-based estimators.

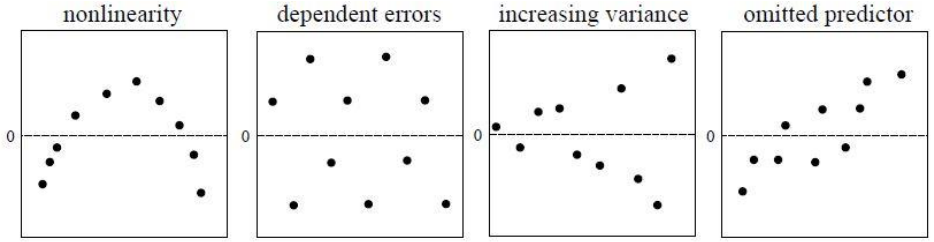
- Assumptions in linear regression[8]

Assumption	Description
Linearity	The variables (*x*, *Y*) actually exhibit a linear relationship.
Independency	Observations should be independent.
Homoscedasticity	For each value of the predictor (*x*), the variance of the response (*Y*) should be the same.
Normality	For each value of the predictor (*x*), the distribution of the response (*Y*) is normal.
Normality	The errors should be normally distributed.

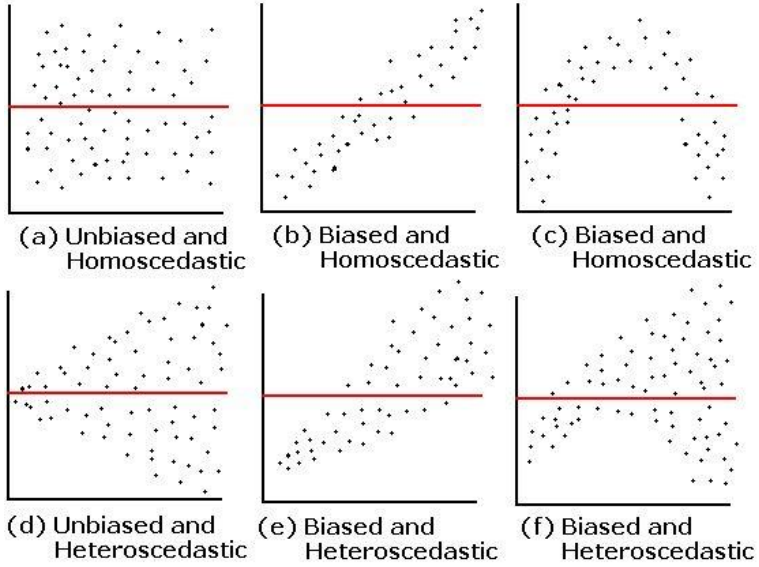
  - Check correlation and/or scatterplots of the variables.
  - Check normality (qqplot) and homoscedasticity (residuals vs fitted values plot) of residuals.

[8] The last three assumptions can be summarized as $\varepsilon_i \sim iid\ N(0, \sigma^2),\ i = 1, 2, \cdots, n$

- Residual plot for model checking



  - Nonlinear trend: Try polynomial regression model.
  - Non-constant variance: Try Weighted Least Squares (WLS) or variable transformation.



- Coefficient of Determination (R^2)

$$R^2 = \frac{SS_{Regression}}{SS_{Total}} = 1 - \frac{SS_{Residual}}{SS_{Total}}$$

  - Indicate how well data fit a statistical model.
  - Proportion of total response variation explained by the regressors in the model
  - $0 \leq R^2 \leq 1$
  - $R^2 = 1$: Fitted model explains all variability in Y
  - $R^2 = 0$: No linear relationship for straight line regression

- Multiple Linear Regression

$$\begin{array}{ccccc} Y & = & \beta_0 + \beta_1\, x_1 + \beta_2\, x_2 + \cdots + \beta_p\, x_p & + & \varepsilon \\ \text{(Response)} & = & \text{(Linear Model)} & + & \text{(Error)} \end{array}$$

where $\varepsilon \sim N(0, \sigma^2)$.

  - Include $p \geq 2$ independent variables in the regression model.
  - Fitted model: $\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j$ where $\hat{\beta}_j, j = 0, 1, 2, \cdots, p$, are sample-based estimators.
  - Multi-collinearity: Additional assumption to check for multiple linear regression
  - Variation Influential Factor (VIF): Measure how much the variance of estimated regression coefficient increases because of collinearity. (Problematic if >10)

- Model selection
  - Explain the data in the simplest way by excluding unnecessary predictors.
  - Prevent collinearity that is caused by having too many variables doing the same job.
  - Backward, forward, stepwise selection

- More on regression…
  - Dummy variables
  - Interaction
  - Variable transformation
  - Interpretation / Prediction
  - Model comparison: AIC, BIC, Adjusted R^2, Mallow's C_p

14.3. PROC REG

General Syntax

```
proc reg data=dataset plots(options) = (list-of-plot-requests);
     model dependent-variable = list-of-independent-variables / <options>;
run; quit;
```

Option	Description
VIF	Produce Variation Influential Factor (VIF).
NOINT	Fit a model without intercept.
NOPRINT	Do not print the result.
SELECTION = `method`	Specify a model selection method. e.g. FORWARD, BACKWARD, STEPWISE, RSQUARE, CP
P	Calculate predicted values from the input data and the estimated model.
R	Request detailed information about residuals.
CLM	Display the 100(1 – α)% confidence limits for the mean predicted values.
CLI	Display the 100(1 – α)% confidence limits for individual predicted values.
ALPHA = *n*	Specify the level for the confidence limits (intervals). Between 0 (100% confidence) and 1 (0% confidence). Default is 0.05 (95% confidence limits).
INFLUENCE	Request a detailed analysis of the influence of each observation on the estimates and the predicted values.

- By default, the RESIDUALS and DIAGNOSTICS are automatically generated.
- For a simple linear regression, a FITPLOT is additionally generated automatically.

Plot Request	Description
FITPLOT	Scatterplot with regression line and confidence and prediction bands
RESIDUALS	Residuals plotted against independent variable
DIAGNOSTICS	Diagnostics panel including all of the following plots
COOKSD	Cook's D statistic by observation number
OBSERVEDBYPREDICTED	Dependent variable by predicted values
QQPLOT	Normal quantile plot of residuals
RESIDUALBYPREDICTED	Residuals by predicted values
RESIDUALHISTOGRAM	Histogram of residuals
RFPLOT	Residual fit plot
RSTUDENTBYLEVERAGE	Studentized residuals by leverage
RSTUDENTBYPREDICTED	Studentized residuals by predicted values

14.4. PROC GLM[9]

General Syntax

```
proc glm data=dataset;
     class list-of-categorical-variables;
     model dependent-variable = list-of-independent-variables;
run;
```

- CLASS: Specify the list of categorical variables included in the model. (cf) Dummy variables)
- Useful for various analysis
  - Linear regression
  - Analysis of variance (ANOVA)
  - Analysis of covariance (ANCOVA)
  - Weighted regression (Weighted Least Squares: WLS)
  - Multivariate analysis of variance (MANOVA)
- PROC GLMSELECT: Conduct model selection

[9] Check Chapter 12.8. for PROC GLM statements.

Example: Simple Linear Regression

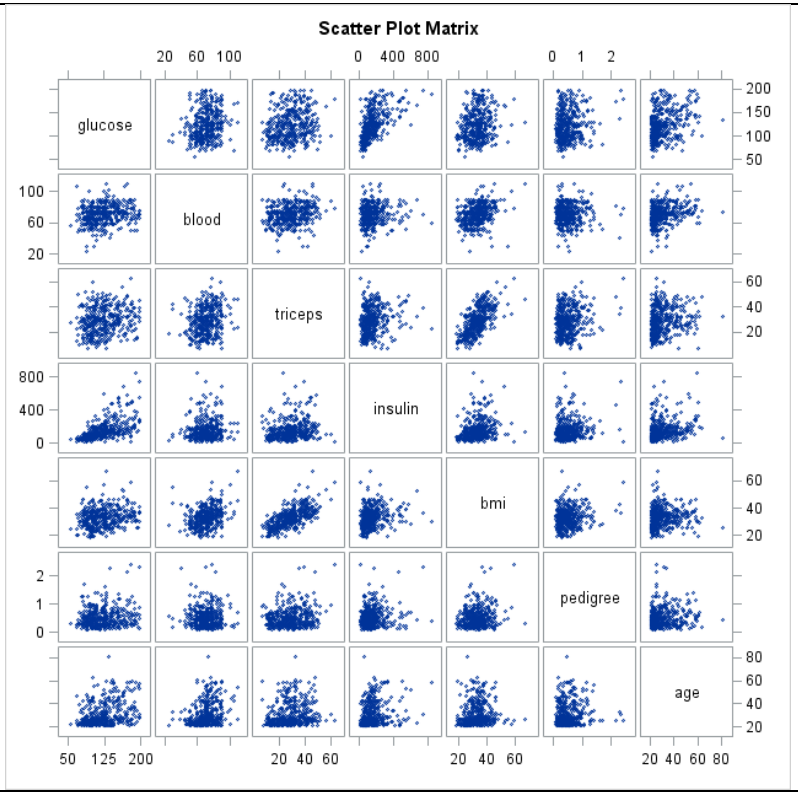
Raw Data

Obs	id	pregnant	glucose	blood	triceps	insulin	bmi	pedigree	age	test
1	1	1	89	66	23	94	28.1	0.167	21	Negative
2	2	0	137	40	35	168	43.1	2.288	33	Positive
3	3	3	78	50	32	88	31	0.248	26	Positive
4	4	2	197	70	45	543	30.5	0.158	53	Positive
5	5	1	189	60	23	846	30.1	0.398	59	Positive

SAS Code

```
* Check the linear correlation;
proc corr data=pima
plots(maxpoints=100000000)=matrix(nvar=7);
     var glucose -- age;
run;
```

Output



```
* Simple linear regression: triceps (Y) ~
bmi (x);
proc reg data=pima;
     model triceps = bmi;
     output out=regout p=yhat r=resid;
run; quit;
```

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	19086	19086	308.13	<.0001
Error	390	24157	61.94048		
Corrected Total	391	43243			

Root MSE	7.87023	R-Square	0.4414
Dependent Mean	29.14541	Adj R-Sq	0.4399
Coeff Var	27.00332		

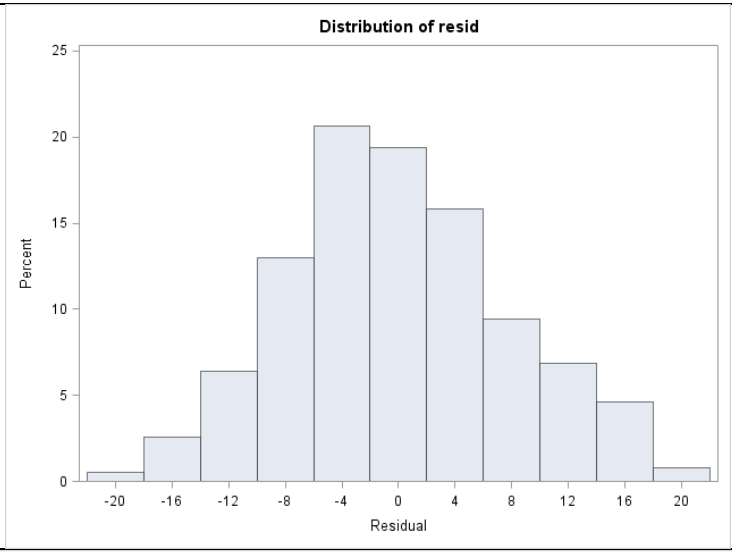
Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	Intercept	1	-3.74769	1.91555	-1.96	0.0511
bmi	bmi	1	0.99416	0.05664	17.55	<.0001

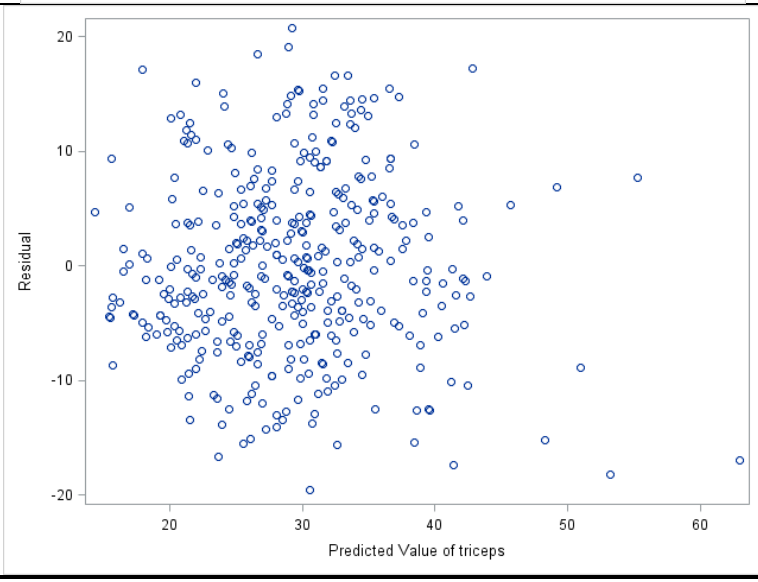
```
* Residual: normality check;
proc univariate data=regout normal;
     var resid;
     histogram resid;
     qqplot resid;
run;
```

Tests for Normality

Test	Statistic		p Value	
Shapiro-Wilk	W	0.992185	Pr < W	0.0377
Kolmogorov-Smirnov	D	0.04267	Pr > D	0.0821
Cramer-von Mises	W-Sq	0.131234	Pr > W-Sq	0.0439
Anderson-Darling	A-Sq	0.819541	Pr > A-Sq	0.0357




```
* Scatterplot: predicted values vs
residuals;
proc sgplot data=regout;
     scatter x=yhat y=resid;
run;
```



Example: Multiple Linear Regression

Raw Data

Obs	id	pregnant	glucose	blood	triceps	insulin	bmi	pedigree	age	test
1	1	1	89	66	23	94	28.1	0.167	21	Negative
2	2	0	137	40	35	168	43.1	2.288	33	Positive
3	3	3	78	50	32	88	31	0.248	26	Positive
4	4	2	197	70	45	543	30.5	0.158	53	Positive
5	5	1	189	60	23	846	30.1	0.398	59	Positive

SAS Code

```
* Multiple regression;
proc reg data=pima;
     model triceps = glucose blood
insulin bmi pedigree age / vif;
run; quit;
```

Output

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	19835	3305.84505	54.37	<.0001
Error	385	23408	60.79907		
Corrected Total	391	43243			

Root MSE	7.79738	R-Square	0.4587
Dependent Mean	29.14541	Adj R-Sq	0.4503
Coeff Var	26.75336		

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation
Intercept	Intercept	1	-7.96647	2.84577	-2.80	0.0054	0
glucose	glucose	1	0.00793	0.01651	0.48	0.6313	1.66898
blood	blood	1	-0.00125	0.03500	-0.04	0.9716	1.23002
insulin	insulin	1	-0.00073948	0.00413	-0.18	0.8580	1.54959
bmi	bmi	1	0.96762	0.06116	15.82	<.0001	1.18806
pedigree	pedigree	1	1.40261	1.17122	1.20	0.2318	1.05298
age	age	1	0.11646	0.04274	2.72	0.0067	1.22231

```
* Stepwise selection;
proc reg data=pima;
     model triceps = glucose blood
insulin bmi pedigree age /
selection=stepwise;
     output out=regout p=yhat
r=resid;
run; quit;
```

Analysis of Variance

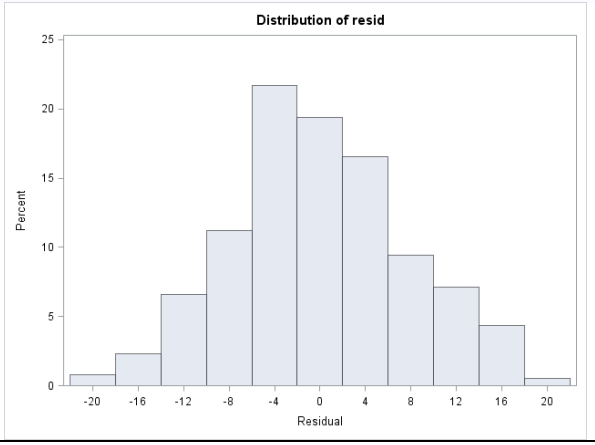
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	19726	9863.07131	163.15	<.0001
Error	389	23517	60.45391		
Corrected Total	391	43243			

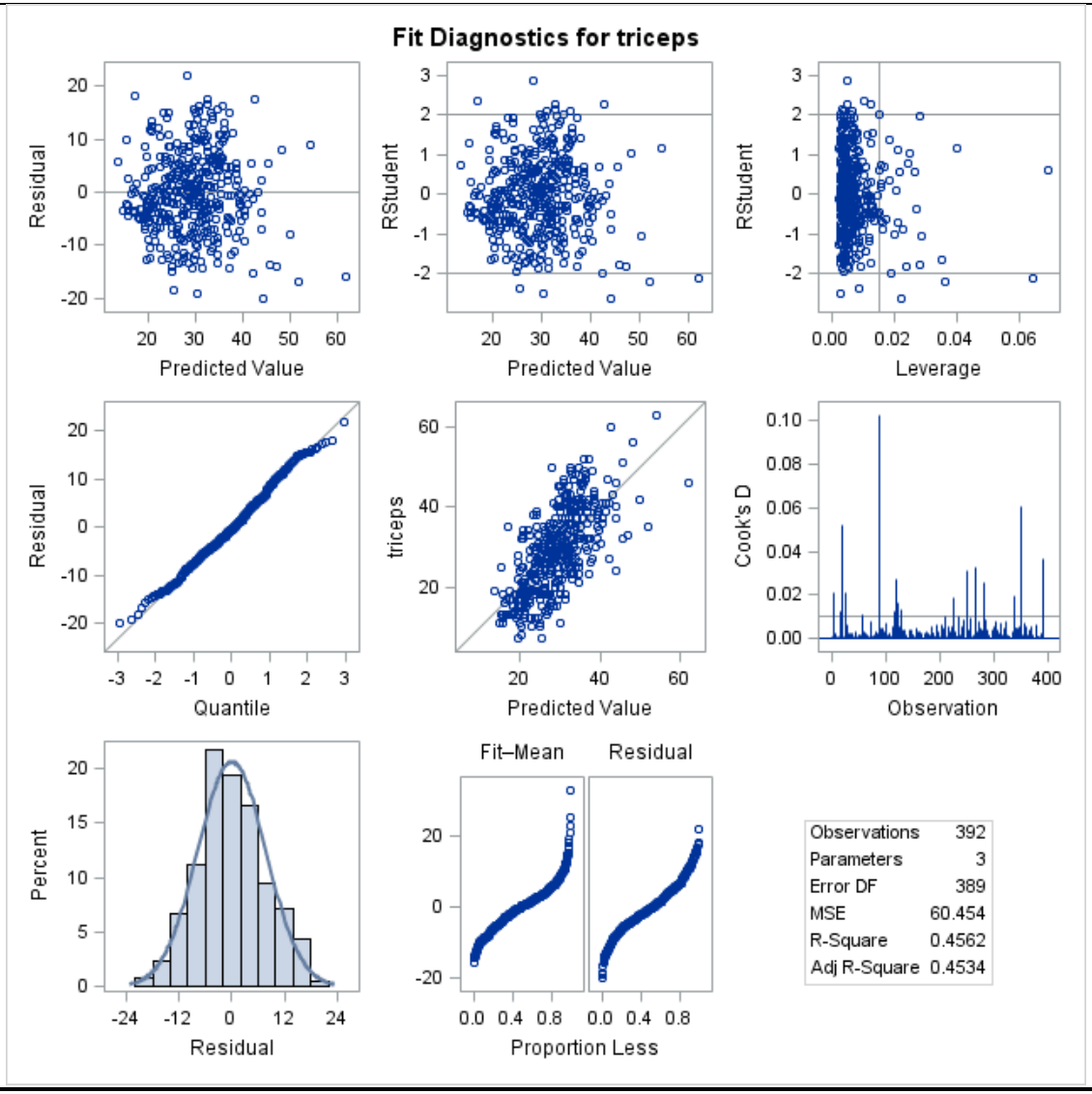
Variable	Parameter Estimate	Standard Error	Type II SS	F Value	Pr > F
Intercept	-7.20728	2.17059	666.51997	11.03	0.0010
bmi	0.98142	0.05609	18509	306.17	<.0001
age	0.12575	0.03864	640.21985	10.59	0.0012

```
* Residual: normality check;
proc univariate data=regout normal;
     var resid;
     histogram resid;
     qqplot resid;
run;
```

Tests for Normality

Test	Statistic		p Value	
Shapiro-Wilk	W	0.993879	Pr < W	0.1156
Kolmogorov-Smirnov	D	0.038037	Pr > D	>0.1500
Cramer-von Mises	W-Sq	0.104089	Pr > W-Sq	0.0995
Anderson-Darling	A-Sq	0.650564	Pr > A-Sq	0.0914




Fit Diagnostics for triceps
Residual
Predicted Value
RStudent
Predicted Value
RStudent
Leverage
Residual
Quantile
triceps
Predicted Value
Cook's D
Observation
Percent
Residual
Fit–Mean
Residual
Proportion Less
Observations 392
Parameters 3
Error DF 389
MSE 60.454
R-Square 0.4562
Adj R-Square 0.4534

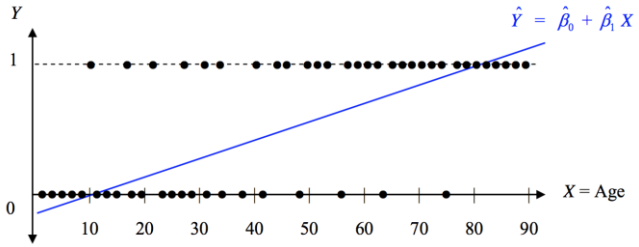
Chapter 15. Generalized Linear Models (GLM)

15.1. Motivation: Why GLM?

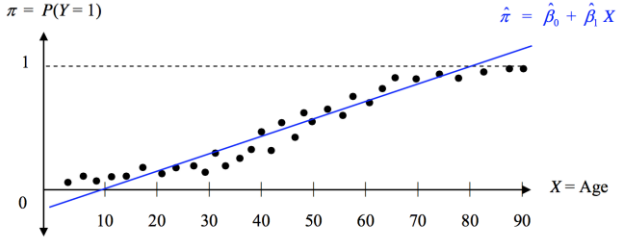
Example: *"If you live long enough, you will need a surgery."*

X = Age

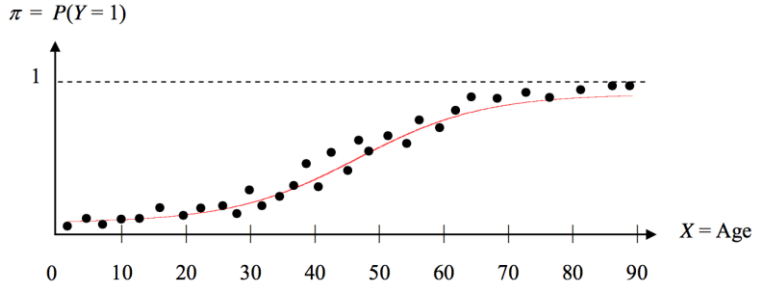
Y = Ever had a major surgery (1 = Yes, 0 = No)



Simple linear regression: Little predictive value for the response (either 0 or 1)



Modeling the probability of Y: Restricted to the finite interval / Violation of assumptions



Transform the probability π:

$$g(\pi) = \log\left(\frac{\pi}{1-\pi}\right) \in (-\infty, +\infty)$$

15.2. Generalized Linear Model (GLM)

- Framework to *generalize* the methods in linear models to the wide class of distributions
- Model functions of the mean
- Components

Component	Description
Random	Response variable Y with independent observations (Y_1, Y_2, …, Y_n) forms a distribution in a natural exponential family. $$f(y;\theta) = h(y)\exp[T(y)\,b(\theta) - A(\theta)]$$ e.g. Poisson, binomial, normal
Systematic	Systematic component involves the explanatory variables $x_1, x_2, \dots, x_p$ as linear predictors. $$g(\mu) = \eta = \sum_{j=1}^{p} \beta_j x_j = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$ where $E(Y_i) = \mu_i, i = 1, 2, \dots, n$.
Link	Link function $g(\cdot)$ describes the relationship between the random and systematic components. $$g(\mu) = \eta$$ e.g. $g(\mu) = \mu$: Identity link

- Types of GLM

Random	Support	Link		Model
Normal	$(-\infty, +\infty)$	Identity	$g(\mu) = \mu = X\beta$	Linear-response regression
Exponential Gamma	$(0, +\infty)$	Inverse	$g(\mu) = \frac{1}{\mu} = X\beta$	Exponential-response regression
Poisson	{0, 1, 2, …}	Log	$g(\mu) = \log(\mu) = X\beta$	Log-linear regression
Bernoulli Binomial	{0, 1} {0, 1, 2, … N}	Logit	$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right) = X\beta$	Logistic regression
Multinomial	K outcomes	Logit	$\log\left(\frac{\Pr(Y=k)}{\Pr(Y=K)}\right) = \beta_k X$ $k = 1, 2, \ldots, K-1$	Multinomial logistic regression

  - In case of over-dispersion, consider negative binomial distribution instead of Poisson.
  - Multinomial distribution with orders: Ordinal logistic regression
  - Predictors (X) can take on any form: Binary, categorical, and/or continuous
  - Log: Natural log (i.e. *ln*)

15.3. PROC GENMOD

General Syntax

```
proc genmod data=dataset;
     class categorical-variable(ref="Reference");
     model dependent-variable = list-of-independent-variables
           / dist = distribution link = link-function;
     lsmeans categorical-variable / <options>;
run;
```

- More flexible than PROC GLM with a choice of link functions
- CLASS: Specify categorical variables and their reference category.
- (Distribution) DIST = normal (default), poisson, bin, negbin
- (Link function) LINK = identity (default), log, logit, probit, cloglog
- LSMEANS: Compute least squares means corresponding to the specified effects.

Option	Description
ALPHA = *n*	Specify the level for the confidence limits. Between 0 (100% confidence) and 1 (0% confidence). Default is 0.05 (95% confidence limits).
CL	Request the confidence limits for each of the LS-means.
CORR [COV]	Request the estimated correlation [covariance] matrix of the LS-means.

- PROC HPGENSELECT: Conduct model selection

15.4. Log-linear Regression

- Random component

$$Y_i \mid X \sim Poisson(\lambda_i),\ \ E(Y_i \mid X) = \lambda_i,\ \ i = 1, 2, \ldots, n$$

- Systematic component: Linear predictor $(x_1, x_2, \ldots, x_p)$

$$\eta_i = \sum_{j=1}^{p} \beta_j x_{ij} = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

- Link function (log)

$$g(\lambda_i) = \log(\lambda_i) \in (-\infty, +\infty)$$

- Log-linear regression

$$g(\lambda_i) = \log(\lambda_i) = \sum_{j=1}^{p} \beta_j x_{ij} = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

- SAS: PROC GENMOD

General Syntax

```
proc genmod data=dataset;
     class categorical-variable(ref="Reference");
     model dependent-variable = list-of-independent-variables
           / dist = poisson link = log;
run;
```

Example: Log-linear regression

Raw Data

Obs	id	pregnant	glucose	blood	triceps	insulin	bmi	pedigree	age	test
1	1	1	89	66	23	94	28.1	0.167	21	Negative
2	2	0	137	40	35	168	43.1	2.288	33	Positive
3	3	3	78	50	32	88	31	0.248	26	Positive
4	4	2	197	70	45	543	30.5	0.158	53	Positive
5	5	1	189	60	23	846	30.1	0.398	59	Positive

SAS Code

```
* Poisson distribution / Log link;
proc genmod data=pima;
     class test(ref="Negative");
     model pregnant = insulin|test age / dist = poisson link = log;
run;
```

Output

Analysis Of Maximum Likelihood Parameter Estimates

Parameter		DF	Estimate	Standard Error	Wald 95% Confidence Limits		Wald Chi-Square	Pr > ChiSq
Intercept		1	-0.4195	0.0964	-0.6084	-0.2306	18.95	<.0001
insulin		1	-0.0002	0.0004	-0.0009	0.0006	0.20	0.6585
test	Positive	1	0.3462	0.1005	0.1492	0.5431	11.87	0.0006
test	Negative	0	0.0000	0.0000	0.0000	0.0000	.	.
insulin*test	Positive	1	-0.0009	0.0005	-0.0019	0.0001	3.23	0.0723
insulin*test	Negative	0	0.0000	0.0000	0.0000	0.0000	.	.
age		1	0.0465	0.0021	0.0424	0.0507	476.97	<.0001
Scale		0	1.0000	0.0000	1.0000	1.0000		

15.5. Logistic Regression

- Random component

$$Y_i \mid X \sim Binomial(n_i, p_i),\ \ E(Y_i/n_i \mid X) = \ p_i,\ \ i = 1, 2, \dots, n$$

- Systematic component: Linear predictor $(x_1, x_2, \dots, x_p)$

$$\eta_i = \sum_{j=1}^{p} \beta_j x_{ij} = \beta_1 x_{i1} + \ \beta_2 x_{i2} + \cdots + \ \beta_p x_{ip}$$

- Link function (Logit)

$$g(p_i) = logit(p_i) = \log\left(\frac{p_i}{1 - \ p_i}\right) \in (-\infty, +\infty)$$

- Logistic regression

$$g(p_i) = \text{logit}(p_i) = \sum_{j=1}^{p} \beta_j x_{ij} = \beta_1 x_{i1} + \ \beta_2 x_{i2} + \cdots + \ \beta_p x_{ip}$$

- SAS: PROC GENMOD

General Syntax

```
proc genmod data=dataset;
     class categorical-variable(ref="Reference");
     model dependent-variable = list-of-independent-variables
           / dist = bin link = logit;
run;
```

- SAS: PROC LOGISTIC

General Syntax

```
proc logistic data=dataset descending;
     class categorical-variable(ref="Reference") / param = ref;
     model dependent-variable = list-of-independent-variables / lackfit;
run;
```

- DESCENDING: Sort the response variable from highest to lowest.
- By default, SAS models the probability of the lower category.
- PARAM = REF: Use the specified reference values for modeling.
- LACKFIT: Provide the Hosmer-Lemeshow for goodness-of-fit test

  H_0: The logistic regression fits well.

- Interpretation
  - The sign of β determines whether the log odds of Y is increasing or decreasing.
  - If $\beta = 0$, then there is no linear relationship between the *log odds* of Y and X.
  - Odds ratio (OR) = e^{β}

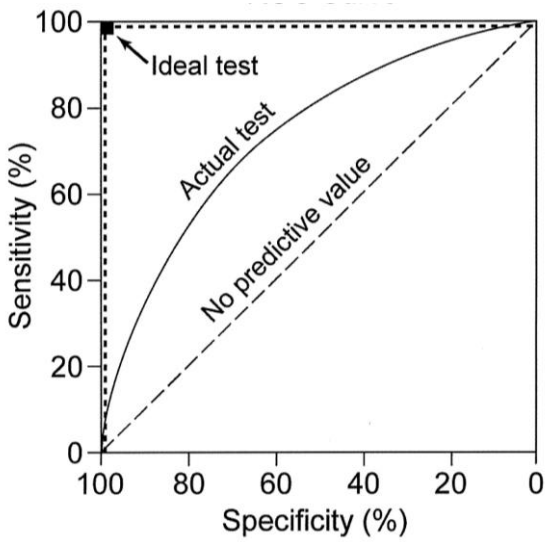
    1) Ratio of the probability of success (group 1) and that of failure (group 2)

    2) $OR \in [0, +\infty)$

    3) OR = 1: There is no difference between the groups compared.

    4) OR > 1: Group 1 has a greater probability than group 2.

- Receiver operating characteristic (ROC) curve
  - Sensitivity (True positive rate) / Specificity (True negative rate)
  - A model with high discrimination ability will have high sensitivity and specificity simultaneously, leading to the ROC curve getting close to the top left corner of the plot.
  - Area under the curve (AUC): Provide the probability that a randomly selected pair of subjects (one truly positive and one truly negative) will be correctly ordered by the test.
  - AUC $\in$ [0.5 (No discrimination), 1 (Perfect discrimination)]



15.6. Comparison between Procedures

Procedure	Description
PROC REG	Perform a linear regression with diagnostic tests.
PROC GLM	Perform a simple/multiple/polynomial/weighted regression. Provide a wide range of options for analysis with limited model-checking capacity.
PROC LOGISTIC	Perform logistic regression with diagnostic tests.
PROC GENMOD	Fit a generalized linear model using MLE.

Example: Logistic regression

Raw Data

Obs	id	pregnant	glucose	blood	triceps	insulin	bmi	pedigree	age	test
1	1	1	89	66	23	94	28.1	0.167	21	Negative
2	2	0	137	40	35	168	43.1	2.288	33	Positive
3	3	3	78	50	32	88	31	0.248	26	Positive
4	4	2	197	70	45	543	30.5	0.158	53	Positive
5	5	1	189	60	23	846	30.1	0.398	59	Positive

SAS Code

```
* Binomial distribution / Logit link;
proc genmod data=pima descending;
     model test = glucose bmi pedigree age / dist = bin link = logit;
run;

* PROC LOGISTIC;
proc logistic data=pima plots(only)=(roc effect);
     class test (ref="Negative") / param=ref;
     model test = glucose bmi pedigree age / lackfit outroc=roc;
run;
```

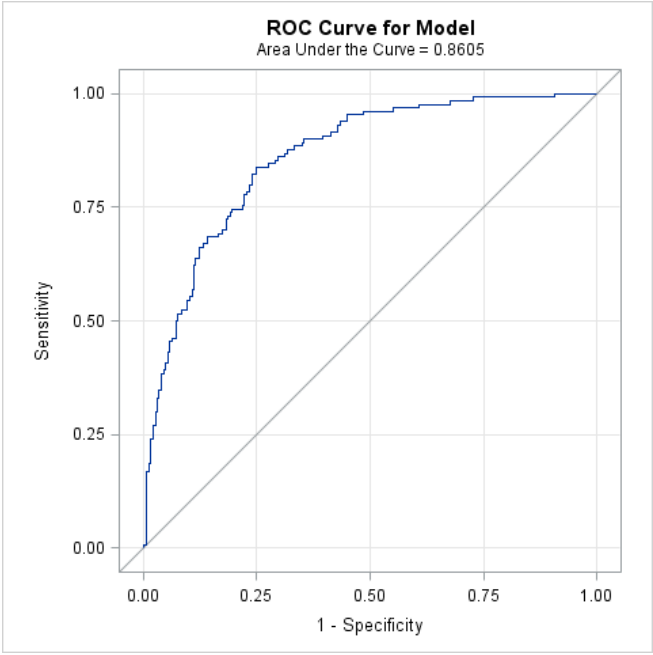
Output

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-10.0920	1.0802	87.2780	<.0001
glucose	1	0.0362	0.00498	52.7658	<.0001
bmi	1	0.0744	0.0203	13.4940	0.0002
pedigree	1	1.0871	0.4194	6.7186	0.0095
age	1	0.0530	0.0134	15.5590	<.0001

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
glucose	1.037	1.027	1.047
bmi	1.077	1.035	1.121
pedigree	2.966	1.304	6.747
age	1.054	1.027	1.083

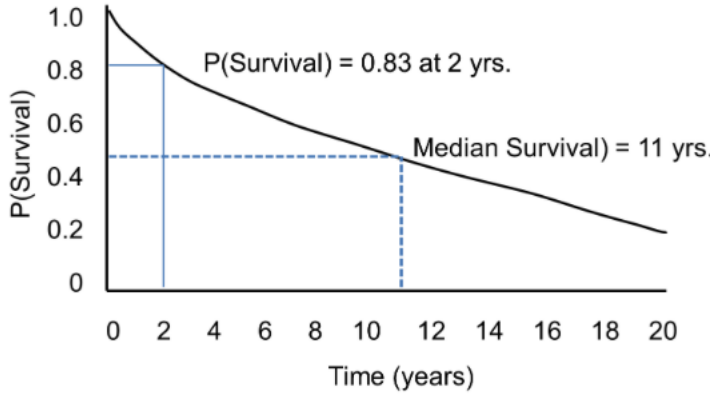


Chapter 16. Survival Analysis

16.1. Time-to-Event

- In many clinical/medical researches, the 'time to event' *T* is a variable of primary interest.
  - *T*: None-negative random variable
  - Event: Death, failure, equipment breakdown, development of some disease, etc.
  - Clinical endpoint, survival time, of failure time
- Generally, not symmetrically distributed.
  - Only few subjects survive longer compared to the majority.
- Survival time is right censored.
  - At the end of the study, some subjects may not have reached the endpoint of interest.
  - Assumption: Time-to-event is independent of the censoring mechanism.

- Example
  - Time until cardiovascular death after some treatment intervention
  - Time until tumor recurrence
  - Remission duration of certain disease in clinical trials
  - Incubation time of certain disease (e.g. AIDS, Hepatitis C)



  - The 10-year survival rate of patients with stage 3 colon cancer after the diagnosis
  - Which gender is more likely to survive 3 years after a surgery?

16.2. Incomplete Data: Censoring

- Censoring happens when a value occurs outside the range of a measuring instrument.
- Reasons of censoring: Withdrawal, lost to follow-up, event-free at last follow-up, death due to another cause, etc.

Type	Description
Right censoring	The individual is still alive or has not experienced the event of interest at the end of the study.
Left censoring	The individual has already experienced the event of interest prior to the start of the study. We know that the event occurred, but are not sure when exactly it happened. e.g. First time smoking, Alzheimer disease (onset hard to determine)
Interval censoring	The event occurs within some interval. Due to discrete observation times, actual event time is unknown.
Type I censoring	An experiment has a set number of subjects and the study ends at a predetermined *time*. Some subjects remain right-censored.
Type II censoring	An experiment has a set of number of subjects and the study ends when a predetermined *number* of subjects experience the event of interest. Some subjects remain right-censored.

16.3. Incomplete Data: Truncation

- Truncation ≠ Censoring
- Truncation occurs when the incomplete nature of the observation is due to a systematic selection process inherent to the study design (sampling bias).
- Only those individuals whose time of event lies within a certain interval $[Y_L, Y_R]$ are included.

Type	Description
Right truncation	Only individuals who have experienced the event by a specified time are included in the sample. e.g. Patients with AIDS from transfusion (Only patients who were infected with AIDS virus after March 1, 2005 and developed AIDS by June 30, 2014 are included.)
Left truncation	Only individuals who survive a sufficient time are included in the sample. e.g. Death time of elderly residents of a retirement community (Only the elderly people of a certain age can be admitted into the community. People died before this age cannot be included.)

16.4. Important Functions

- Let *T* be the survival time (time-to-event) with pdf $f(t),\ t \in [0, \infty)$.
- Cumulative distribution function $F(t)$

$$F(t) = P(T \leq t) = \int_0^t f(s)\, ds$$

- Survival function $S(t)$

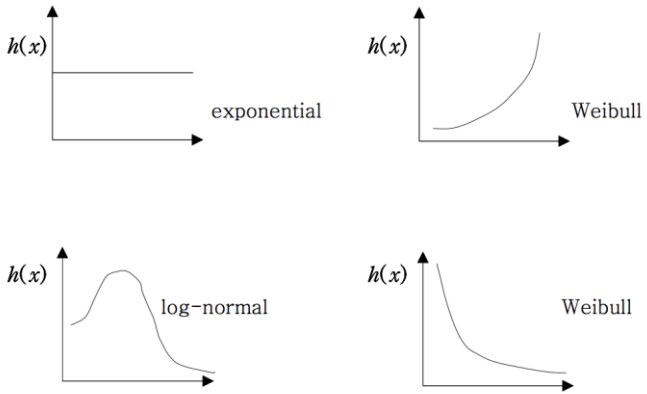
$$S(t) = P(T > t) = \int_t^\infty f(s)\, ds = 1 - P(T \leq t) = 1 - F(t)$$

  - $S(0) = 1$
  - $S(\infty) = \lim_{t \to \infty} S(t) = 0$
  - $S(t)$ is non-increasing in t and right-continuous.

- Hazard function $\lambda(t)$

$$\lambda(t) = \lim_{h \to 0+} \frac{P(t \leq T < t + h \mid T \geq t)}{h}$$

  - *Instantaneous* failure rate at *t* given survival up to *t*
  - Conditional failure rate / Intensity function / Force of mortality / Instantaneous hazard

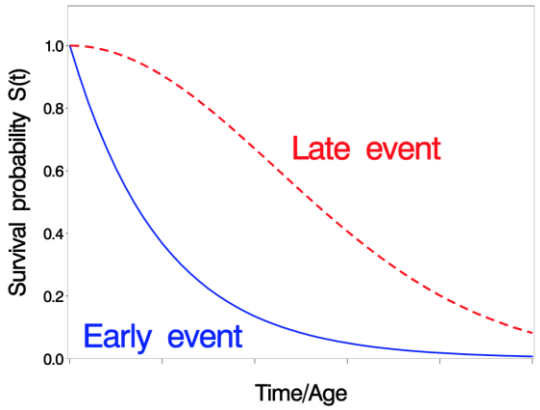


- Cumulative hazard function $\Lambda(t)$

$$\Lambda(t) = \int_0^t \lambda(s)\, ds$$

- Relationship between functions
  - $f(t) = \frac{dF(t)}{dt} = -\frac{dS(t)}{dt}$
  - $\lambda(t) = \frac{d\Lambda(t)}{dt} = -\frac{d\log S(t)}{dt} = \frac{f(t)}{S(t)}$
  - $\Lambda(t) = -\log S(t)$
  - $\mathrm{S(t)} = e^{-\Lambda(t)} = \exp\left(-\int_0^t \lambda(s)\, ds\right)$
  - $\Lambda(\infty) = \infty\ (\because S(\infty) = 0)$

- Example: Compare survival or hazard functions of different groups. (e.g. treatment / gender)



Survival analysis considers *censoring* and *time-dependent* covariates.

- ✓ Compare mean time to events (t-test or linear regression) → Ignore censoring
- ✓ Compare proportion of events (Relative risk, OR, or logistic regression) → Ignore time

16.5. Survival Analysis: Notation

- Mainly focus on right censored data
  - *X*: True survival time
  - *C*: Censoring time
  - $\Delta = I(X \leq C)$: Censoring indicator (0 if censored)
  - $Y = \min(X, C)$: What we actually observe
  - Censoring time is independent of the event of interest. (i.e. *X* is independent of *C*)
- Survival analysis

Approach	Description
Parametric	e.g. Exponential / Weibull distribution
Nonparametric	No specification about the distribution of survival time Kaplan-Meier (product-limit) estimator

16.6. Kaplan-Meier (K-M) Estimator

- Suppose for *n* subjects,
  - Observed at distinct, ordered time points $t_1 < t_2 < \cdots < t_k,\ k \leq n$
  - d_i: The number of failures at time t_i
  - n_i: The number of subjects at risk (i.e. no event and not censored) just prior to t_i

    (Size of the risk set)
- If there exist right censored individuals,
  - Estimated hazard function

$$\hat{\lambda}(t_i) = \frac{d_i}{n_i}, \qquad i = 1, 2, \cdots k$$

  - Kaplan-Meier estimator of survival function

$$\hat{S}(t) = \prod_{i:\, t_i \leq t} \left(1 - \hat{\lambda}(t_i)\right) = \prod_{i:\, t_i \leq t} \left(1 - \frac{d_i}{n_i}\right)$$

- Note that the K-M estimator is undefined after the largest observed failure time.

  DO NOT extrapolate!

Example

Data A small study is looking at time to relapse after a cancer treatment.
Data from 10 patients is shown below; censored observations are marked by (+):

10, 20+, 35, 40+, 50+, 55, 70+, 71+, 80, 90+

K-M Estimator

Time (t)	Died (d_i)	At risk (n_i)	$\hat{\lambda}(t_i) = d_i/n_i$	$1 - \hat{\lambda}(t_i)$	$\hat{S}(t)$
10	1	10	1/10	9/10	0.9
20	0	9	0	1	0.9 X 1 = 0.9
35	1	8	1/8	7/8	0.9 X 7/8 = 0.79
40	0	7	0	1	0.79 X 1 = 0.79
50	0	6	0	1	0.79 X 1 = 0.79
55	1	5	1/5	4/5	0.79 X 4/5 = 0.63
70	0	4	0	1	0.63 X 1 = 0.63
71	0	3	0	1	0.63 X 1 = 0.63
80	1	2	1/2	1/2	0.63 X 1/2 = 0.32
90	0	1	0	1	0.32 X 1 = 0.32

- Median survival time
  - The median survival time ($t_{50\%}$) is the time beyond which 50% of the individuals in the population of interest are expected to survive. (i.e. $S(t_{50\%}) = 0.5$)
  - The estimated median survival time ($\hat{t}_{50\%}$) is defined as the smallest observed time for which the estimated survival function is less than 0.5.
  - Sometimes, the estimated survival function is greater than 0.5 for all values of *t*, then there is no median survival time.

- PROC LIFETEST
  - Estimate survival functions / K-M table
  - SAS output includes K-M table.
  - Generate graphs and confidence limits.
  - If dataset contains only complete and right-censored observations, PROC LIFETEST requires two components:

    a) Time (of event or censoring)

    b) Censoring indicator (1: event / 0: censored)

General Syntax

```
proc lifetest data=dataset plots=(survival);
     time time-variable*censoring-indicator(0);
run;
```

  - Time variable always comes first.
  - The censoring indicator needs to be numeric.
  - SAS needs to know which observations are censored.

Example

Raw Data

Obs	Day	Treatment	Status
1	4	1	1
2	5	1	1
3	9	1	1
4	10	1	1
5	11	1	1

SAS Code

```
* Kaplan-Meier (K-M) Estimator;
proc lifetest data=leukemia plots=(survival(cl));
     time day*status(0);
run;
```

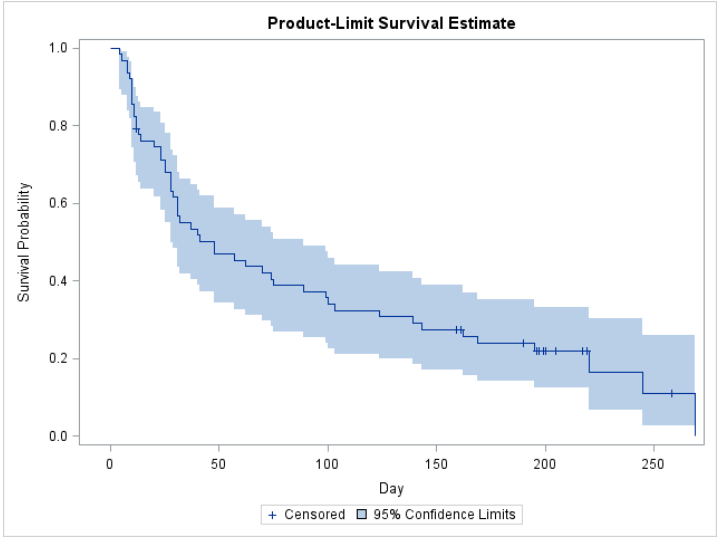
Output

Product-Limit Survival Estimates

Days		Survival	Failure	Survival Standard Error	Number Failed	Number Left
0.000		1.0000	0	0	0	63
4.000		0.9841	0.0159	0.0157	1	62
5.000		0.9683	0.0317	0.0221	2	61
8.000		.	.	.	3	60
8.000		0.9365	0.0635	0.0307	4	59

Quartile Estimates

Percent	Point Estimate	95% Confidence Interval		
		Transform	[Lower	Upper)
75	169.000	LOGLOG	99.000	269.000
50	48.000	LOGLOG	28.000	89.000
25	20.000	LOGLOG	10.000	28.000



16.7. Log-Rank Test

- Compare the survival functions between (2+) groups.

  (e.g. treatment, some demographic characteristics)

- No assumptions about the distribution of survival functions are required.
- Suppose there are *J* groups. The null hypothesis here will be

$$H_0: S_1(t) = \cdots = S_J(t) \text{ for all } t.$$

- Optimal power for detecting differences when hazards are proportional.

General Syntax

```
proc lifetest data=dataset;
     time time-variable*censoring-indicator(0);
     strata strata-variable / test=(all) adjust=mc-method diff=control("ref");
run;
```

- STRATA: Specify the grouping variable.
- TEST=(all): Provide all the available nonparametric tests.
- ADJUST=: Select the multiple comparison adjustment method.[10]
- DIFF=: Declare the reference group.
- If hazards cross, the log-rank test may *not* be suitable.
- Weighting allows the test to depend on the event time and the censoring distribution. (e.g. Gehan, Peto-Peto, Fleming-Harrington)
- Stratified analysis
  - Compare survival functions among one category across other categories.
  - e.g. Clinic (multicenter clinical trial), age group, gender

[10] Check Chapter 12.9. for possible multiple comparison options.

Example

Raw Data

Obs	Day	Treatment	Status
1	4	1	1
2	5	1	1
3	9	1	1
4	10	1	1
5	11	1	1

SAS Code

```
* Log-Rank Test;
proc lifetest data=leukemia plots=(survival);
     time day*status(0);
     strata treatment / test=(all);
run;
```

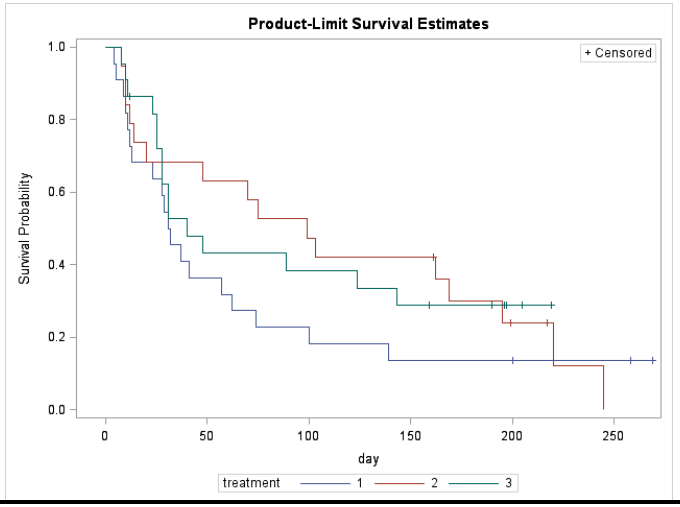
Output

Rank Statistics

treatment	Log-Rank	Wilcoxon	Tarone	Peto	ModifiedPeto	Fleming
1	4.0501	213.00	31.26	3.2454	3.2074	3.2399
2	-1.9887	-137.00	-19.57	-1.9661	-1.9631	-2.0043
3	-2.0614	-76.00	-11.69	-1.2793	-1.2443	-1.2356

Test of Equality over Strata

Test	Chi-Square	DF	Pr > Chi-Square
Log-Rank	1.6425	2	0.4399
Wilcoxon	2.6843	2	0.2613
Tarone	2.5914	2	0.2737
Peto	2.5625	2	0.2777
Modified Peto	2.6242	2	0.2693
Fleming(1)	2.3804	2	0.3042



16.8. Proportional Hazards (PH) Model

- Also known as Cox's regression
- Link the survival functions to multiple covariates (explanatory variables).
- Quantify the *effect* of a certain predictor on survival function.
- Allow to predict survival functions based on a set of covariates.
- Semi-parametric model
  - Make a parametric assumption on the effect of predictors on hazard function.
  - No assumption regarding the nature of the hazard function itself
- With non-time-varying covariates $Z = (Z_1, Z_2, \cdots, Z_k)$, the PH model specifies that

$$\lambda(t|Z) = \lambda_0(t)\, e^{\sum_{i=1}^{k} \beta_i Z_i}$$

  - $\lambda_0(t)$: Arbitrary baseline hazard rate (nonparametric).
  - $\beta = (\beta_1, \beta_2, \cdots, \beta_k)$: Regression coefficients
  - More complicated case: Time-varying covariates $Z_i(t), i = 1, 2, \cdots, k$.

- Hazard ratio

$$\frac{\lambda_{z_1=1}(t|Z_2,\cdots,Z_k)}{\lambda_{z_1=0}(t|Z_2,\cdots,Z_k)} = \frac{\lambda_0(t)\, e^{\beta_1+\sum_{i=2}^{k}\beta_i Z_i}}{\lambda_0(t)\, e^{\sum_{i=2}^{k}\beta_i Z_i}} = e^{\beta_1}$$

  - HR > 1 ($\beta_1 > 0$) : Greater hazard / Worse survival
  - HR < 1 ($\beta_1 < 0$) : Less hazard / Better survival / Protective

- PROC PHREG

General Syntax

```
proc phreg data=dataset;
     model time-variable*censoring-indicator(0) = list-of-independent-variables;
run;
```

- Analysis steps

1. Start by checking the K-M estimates.
2. Fit the Cox proportional hazard (PH) model and get the hazard ratio (HR).
3. Test the proportionality assumption.

   a) The hazard ratio is constant over time, but proportional (multiplicative factor).
   b) The risk does not depend on time. That is, the risk is constant over time.
   ⇒ For each covariate,
   i) Plot survival functions / cumulative hazard functions / log(cumulative hazard).
   ii) Include an interaction term with time (usually log(time)).
   Proportionality condition is met if the interaction terms are not significant.

3.* If the proportionality assumption is not satisfied,

   a) Stratify the model by the non-proportional covariates.
   b) Run Cox models on time intervals rather than on entire time domain.
   c) Include a covariate interaction with time as a predictor.

4. Check the functional form of continuous variables.
   (e.g. Linear, quadratic, categorized form)
5. Look at the residuals. (Random pattern of residuals evenly distributed around zero)

Example

Raw Data

Obs	id	age	beck	hercoc	ivhx	ndrugtx	race	treat	site	los	time	censor
1	1	39	9	4	3	1	0	1	0	123	188	1
2	2	33	34	4	2	8	0	1	0	25	26	1
3	3	33	10	2	3	3	0	1	0	7	207	1
4	4	32	20	4	3	1	0	0	0	66	144	1
5	5	24	5	2	1	5	1	1	0	173	551	0

SAS Code

```
* Cox PHM - multiple predictors;
proc phreg data=uis;
      class treat(ref="0") race(ref="1");
      model time*censor(0)
             = treat age race/rl;
run;
```

Output

Testing Global Null Hypothesis: BETA=0			
Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	15.8131	3	0.0012
Score	15.5781	3	0.0014
Wald	15.5092	3	0.0014

Analysis of Maximum Likelihood Estimates									
Parameter		DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio	95% Hazard Ratio Confidence Limits		Label
treat	1	1	-0.21981	0.08984	5.9861	0.0144	0.803	0.673	0.957	treat 1
age		1	-0.01210	0.00720	2.8219	0.0930	0.988	0.974	1.002	
race	0	1	0.25989	0.10653	5.9515	0.0147	1.297	1.052	1.598	race 0

Chapter 17. Longitudinal Data Analysis

17.1. Longitudinal Study[11]

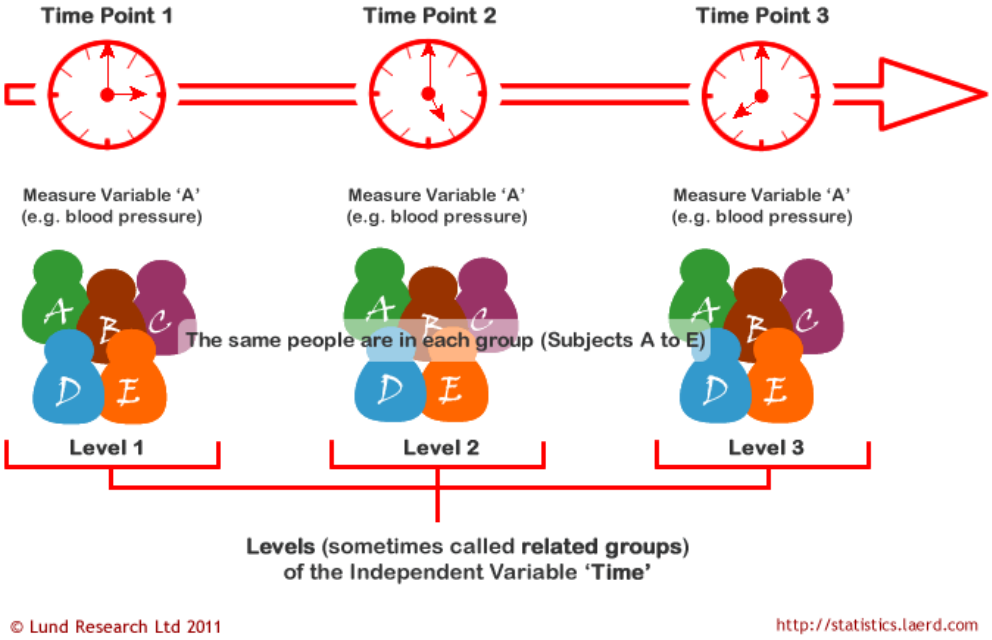
- Participant outcomes (and possibly treatments or exposures) are collected at multiple *follow-up* times (repeated measurements at multiple time points).
- Verify inter-individual differences and how they influence response over time.
- Repeated observations at the individual level exclude the time-invariant unobserved individual differences and observe the temporal order of events (cf. cross-sectional study).
- Cohort: A group of subjects who have shared a particular characteristic or experience during a particular time span
- Prospective (follow-up) / Retrospective (back in time)

[11] See Chapter 10 for manipulating and visualizing the longitudinal data.

- Benefits
  - The timing of disease onset [event] can be correlated with recent changes in exposure and/or with chronic exposure.
  - Multiple follow-up measurements can alleviate recall bias.
  - Measuring the change in outcomes at the individual level provides the opportunity to observe individual patterns of change.
  - The cohort under study is fixed, so changes in time are not confounded by cohort differences. (i.e. Separate cohort and age effects.)
- Challenges
  - There is a risk of bias due to incomplete follow-up or dropout of study participants.
  - Analyzing the correlated data requires a method that can properly account for the intra-subject correlation of response measures.
  - The direction of causality can be complicated by feedback between outcome and exposure (time-varying covariates).

17.2. Longitudinal Data Analysis

- Assuming independence between observations are not appropriate.
  - Measurements within a subject are dependent.
  - Measurements between subjects can be independent.



- Notation
  - Number of subjects: $i = 1, 2, \cdots I$
  - Number of repeated measurements: $j = 1, 2, \cdots J_i$
  - Times of measurement: $t_{ij},\ i = 1, 2, \cdots I,\ j = 1, 2, \cdots J_i$
  - Outcome measured on subject i at time t_{ij}: $y_{ij},\ i = 1, 2, \cdots I,\ j = 1, 2, \cdots J_i$
- Model
  - For subject i, $Y_i = X_i\beta + \varepsilon_i$, where $Var(\varepsilon_i) = \sigma^2 V_i$, $i = 1, 2, \cdots I$.
  - Incorporating all $i = 1, 2, \cdots I$,

$$Y = X\beta + \varepsilon,\ \ Var(\epsilon) = \sigma^2 V = \sigma^2 \begin{bmatrix} V_1 & & \cdots & & 0 \\ & V_2 & & & \\ \vdots & & \ddots & & \vdots \\ & & & V_{I-1} & \\ 0 & & \cdots & & V_I \end{bmatrix}$$

  - The covariates X can be either 1) fixed at the subject level or 2) time-varying.

- Correlation structure[12]

Correlation structure	Description
Unstructured	All elements are unconstrained. ($Var(Y_i) = \Sigma_i = \Sigma$) $n + \frac{n(n-1)}{2} = \frac{n(n+1)}{2}$ parameters
Compound symmetry (Exchangeable)	$Var(Y_i) = \sigma^2 \begin{bmatrix} 1 & \rho & \cdots & \cdots & \rho \\ \rho & 1 & \rho & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho & \rho & \cdots & \rho & 1 \end{bmatrix}$
Toeplitz	$Cov(Y_{ij}, Y_{i(j+k)}) = \rho_k;\ 1 + (n-1) = n$ parameters $Var(Y_i) = \sigma^2 \begin{bmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{n-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho_{n-1} & \rho_{n-2} & \cdots & \rho_1 & 1 \end{bmatrix}$
Banded	Correlation is zero beyond the certain time interval k. i.e. $Cov(Y_{ij}, Y_{i(j+l)}) = 0$ for $l \geq k$ e.g. $k = 2$; $Var(Y_i) = \sigma^2 \begin{bmatrix} 1 & \rho_1 & 0 & \cdots & 0 \\ \rho_1 & 1 & \rho_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \rho_1 & 1 \end{bmatrix}$

- Other examples: AR(p), MA(q), exponential

[12] Assume $j = 1, 2, \cdots, n$ for the ith subject

17.3. Random Effects Model vs Generalized Estimating Equation

Approach	Description
Random effects model	- Incorporate correlation structure in the model by introducing random quantities into the mean. - Introduce random subject effects, coming from a distribution. - Change the intercept/slope of the model between individuals. - Estimate between and within subject variance components.
Generalized estimating equation (Marginal model)	- Model the correlation directly. - Separate the mean structure and correlation. - Focus on estimating the main effects (population average effect) and variance matrices. - Estimate a within-subject variance and a covariance matrix.

- Both random effects model and GEE partition total variability into

  1) subject-level and 2) population-level variance.

- Fundamental difference between random effects model and GEE is in the interpretation of the coefficients.
- GEE is "robust": Provide valid asymptotic confidence intervals of β even if the correlation structure in the model is miss-specified through robust SE estimates.

17.4. Random Effects Model

- Fixed effects & random effects

Fixed Effects	Random Effects
Constant across individuals.	Vary across individuals.
Levels of each factor are fixed in advance.	Levels of factor are meant to be representative of a general population of possible levels.
Estimated using least squares or maximum likelihood.	Estimated with shrinkage.
Marginal interpretation	Conditional interpretation

- Mixed effects model if a model contains both fixed and random effects.
- Random effects models are similar mathematically to introducing penalization.
- Using randomness 1) decreases the number of parameters and 2) induces correlation structures.

- Random intercept model

$$Y_{ij} = \beta_0 + \beta_1 x_{ij} + b_i + \varepsilon_{ij}$$

where $b_i \sim N(0, \sigma_b^2)$ and $\varepsilon_{ij} \sim N(0, \sigma^2)$.[13]

  - Random subject effects (intercept) b_i introduces heterogeneity.
  - Assume correlated subject-level errors.
  - Induce a compound symmetry (exchangeable) within-subject correlation structure.

$$Var(Y_i) = \begin{bmatrix} \sigma_b^2 + \sigma^2 & \sigma_b^2 & \cdots & \cdots & \sigma_b^2 \\ \sigma_b^2 & \sigma_b^2 + \sigma^2 & \sigma_b^2 & \cdots & \sigma_b^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_b^2 & \sigma_b^2 & \cdots & \sigma_b^2 & \sigma_b^2 + \sigma^2 \end{bmatrix} = (\sigma_b^2 + \sigma^2) \begin{bmatrix} 1 & \rho & \cdots & \cdots & \rho \\ \rho & 1 & \rho & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \rho & \rho & \cdots & \rho & 1 \end{bmatrix}$$

where $\rho = \frac{\sigma_b^2}{\sigma_b^2 + \sigma^2}$.

[13] This is a Gaussian case. Generalized random effects model involves a link function similar to GLM.
The compound symmetry correlation structure will not be induced for any other generalized random effects models.

- Random slope model

$$Y_{ij} = \beta_0 + \beta_1 x_{ij} + b_i x_{ij} + \varepsilon_{ij}$$

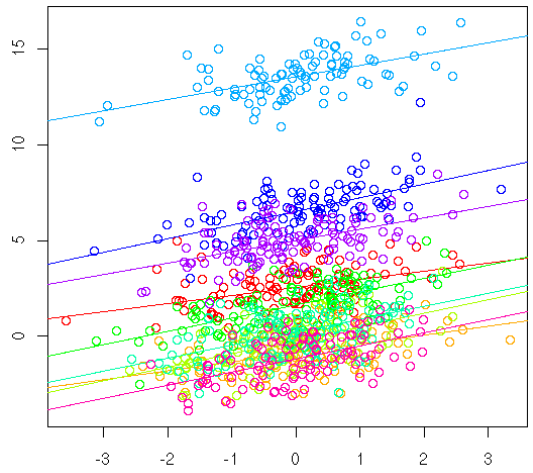
where $b_i \sim N(0, \sigma_b^2)$ and $\varepsilon_{ij} \sim N(0, \sigma^2)$.

- Random intercept & slope model

$$Y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0i} + b_{1i} x_{ij} + \varepsilon_{ij}$$

where

$$\begin{pmatrix} b_{0i} \\ b_{1i} \end{pmatrix} \sim N\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_{b0}^2 & \sigma_{01} \\ \sigma_{01} & \sigma_{b1}^2 \end{pmatrix}\right] \text{ and } \varepsilon_{ij} \sim N(0, \sigma^2).$$

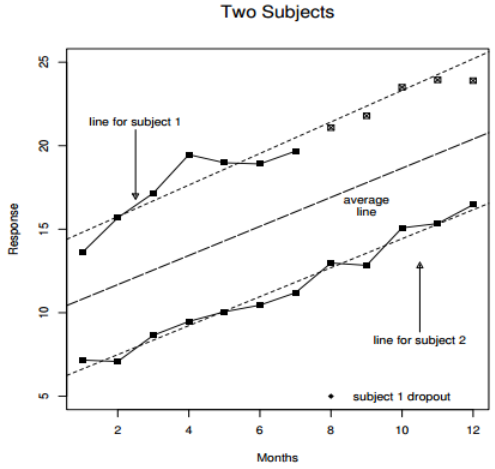


- Interpretation

$$Y_{ij} = \beta_0 + \beta_1 x_{ij} + b_{0i} + b_{1i} x_{ij} + \varepsilon_{ij}$$

  - Time-varying covariates: Effect for an average subject (within-subject covariates)
  - Time-invariant covariates: Individual sharing similar characteristics (between-subject covariates)
  - $E(Y_{ij}) = \beta_0 + \beta_1 x_{ij}$
  - $E(Y_{ij}|b_{0i}, b_{1i}) = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i}) x_{ij}$

    (i.e. b_{0i}, b_{1i}: Effect for a particular subject conditional on the random effects)



- Interclass Correlation Coefficient (ICC)

$$ICC = \frac{\sigma_b^2}{\sigma_b^2 + \sigma^2}$$

  - Quantify how strongly observations within a subject resemble each other.
  - Proportion of variability explained by within-subject variation
  - Assess the consistency or reproducibility of quantitative measurements made via multiple visits or by different observers measuring the same quantity.

- PROC MIXED

General Syntax

```
proc mixed data=dataset;
     model dependent-variable = list-of-independent-variables;
     random random-effects <options>;
     repeated repeated-effect <options>;
run;
```

Statement	Description
RANDOM	Specify the effects in the model that represent repeated measurements and impose a particular covariance structure. Multiple RANDOM statements can be added; Effects in the same statement may be correlated, but independent in different statements.
REPEATED	Specify the random effects and their covariance structures. Control the covariance structure of the residuals.

Option	Description
TYPE = `structure`	Specify the covariance structure. TYPE = VC (default), AR, TOEP, UN, CS
SUBJECT = `effect`	Identify the subjects in the mixed model. Complete independence is assumed across subjects.

17.5. Generalized Estimating Equation (GEE)

- Model[14]

$$Y = X\beta + \varepsilon \quad \text{where } \varepsilon \sim N(0, \sigma^2 V)$$

  - Focus on the marginal distribution (population-averaged effect) of Y rather than on a subject-level conditional distribution.
  - Coefficients are interpreted marginally: Compare subjects based on covariate values.
- Consistent irrespective of the true underlying correlation structure
- Limitations
  - Difficult to assess the goodness-of-fit models due to lack of an inference function
  - Parameter estimates are sensitive to the presence of outliers.
  - Non-convergence and multiple roots problem

[14] This is a Gaussian case. Generalized marginal model involves a link function similar to GLM.

Example

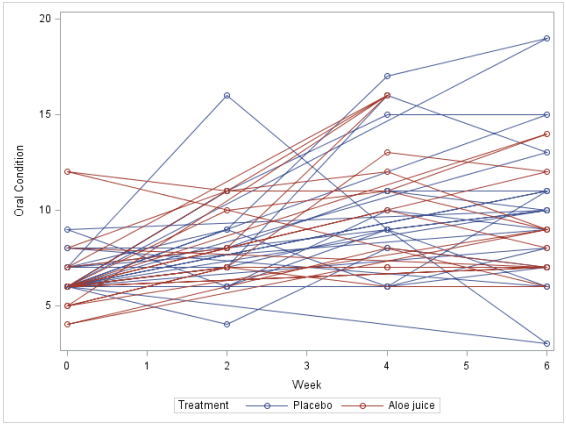
Raw Data

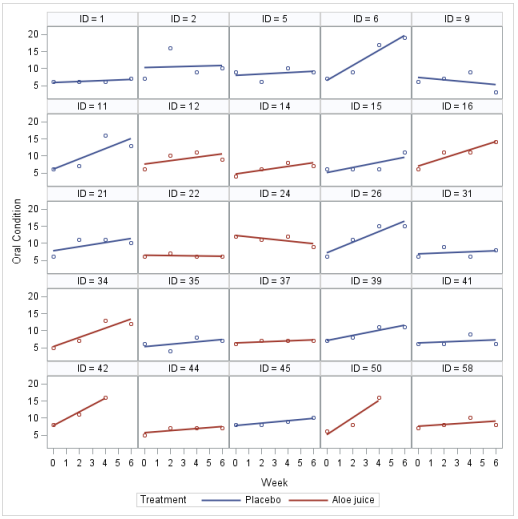
Obs	ID	Treatment	AGE	Initial weight	STAGE	Oral condition in week 0	Oral condition in week 2	Oral condition in week 4	Oral condition in week 6
1	1	Placebo	52	124	2	6	6	6	7
2	5	Placebo	77	160	1	9	6	10	9
3	6	Placebo	60	136.5	4	7	9	17	19
4	9	Placebo	61	179.6	1	6	7	9	3
5	11	Placebo	59	175.8	2	6	7	16	13

SAS Code

```
* Spaghetti plot;
proc sgplot data=Cancer_long;
series x=week y=totalc /
markers group=trt;
xaxis label="Week";
yaxis label="Oral Condition";
run;

* Individuals;
proc sgpanel data=Cancer_long;
panelby ID / columns=5 rows=5;
reg x=week y=totalc
/group=trt;
colaxis label="Week";
rowaxis label="Oral
Condition";
run;
```

Output

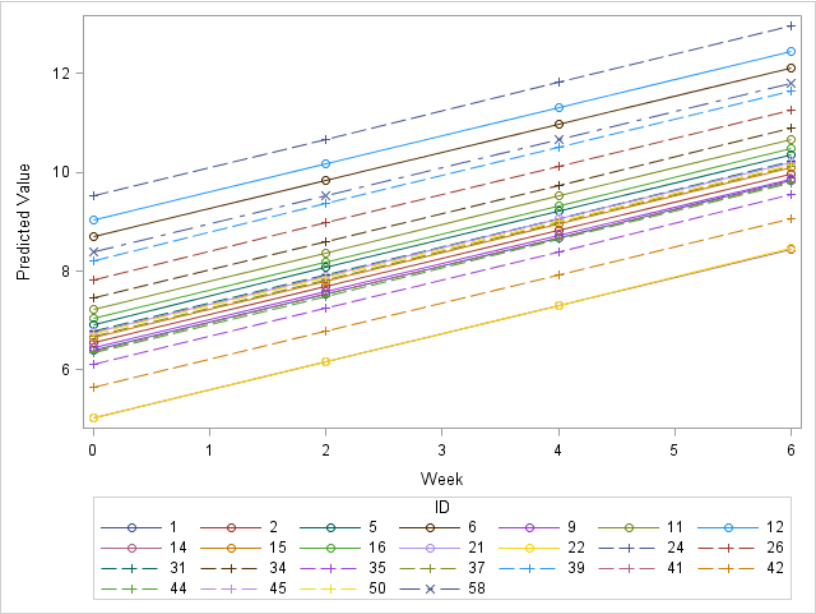




```
* PROC MIXED;
proc mixed data=cancer_long;
class id trt(ref="Placebo");
model totalc = stage weightin
age trt week / solution
outpm=pred;
random intercept / subject=id
type=un;
run;

proc sgplot data=pred;
series x=week y=pred / markers
group=id;
xaxis label="Week";
yaxis label="Predicted Value";
run;
```

Solution for Fixed Effects

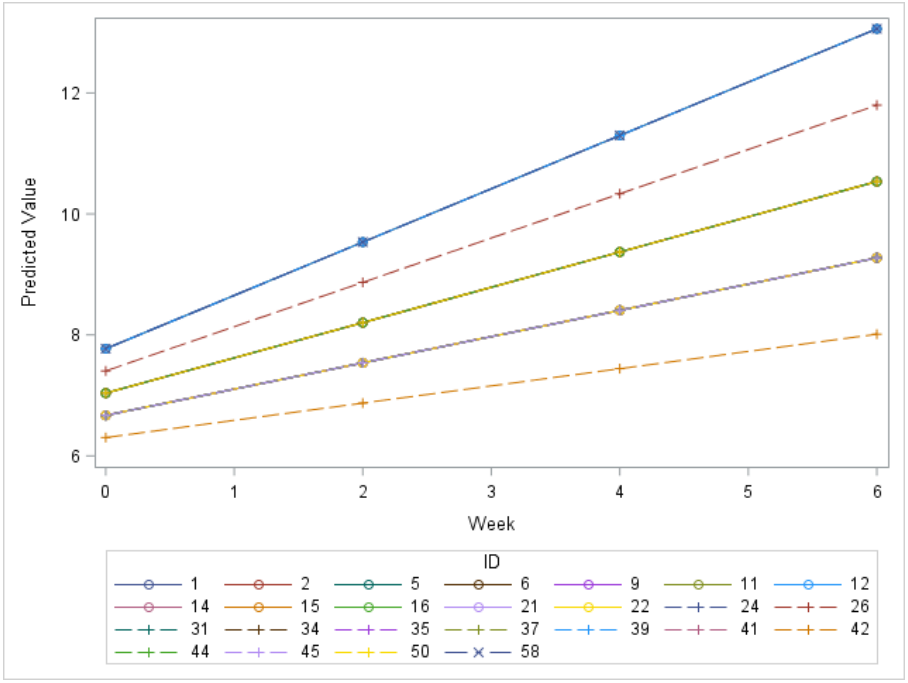
Effect	Treatment	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept		1.0495	3.7169	20	0.28	0.7806
STAGE		0.9116	0.3282	72	2.78	0.0070
WEIGHTIN		0.01032	0.01402	72	0.74	0.4639
AGE		0.04306	0.03266	72	1.32	0.1915
TRT	Aloe juice	-0.4264	0.8543	72	-0.50	0.6192
TRT	Placebo	0	.	.	.	.
WEEK		0.5718	0.1099	72	5.20	<.0001



```
* Interaction;
proc mixed data=cancer_long
covtest;
model totalc = stage|week /
solution notest outpm=pred;
     random intercept /
subject=id type=un;
run;

proc sgplot data=pred;
     series x=week y=pred /
markers group=id;
     xaxis label="Week";
     yaxis label="Predicted
Value";
run;
```

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	6.3027	0.9025	23	6.98	<.0001
STAGE	0.3671	0.4031	71	0.91	0.3655
WEEK	0.2848	0.2053	71	1.39	0.1697
STAGE*WEEK	0.1492	0.09059	71	1.65	0.1041

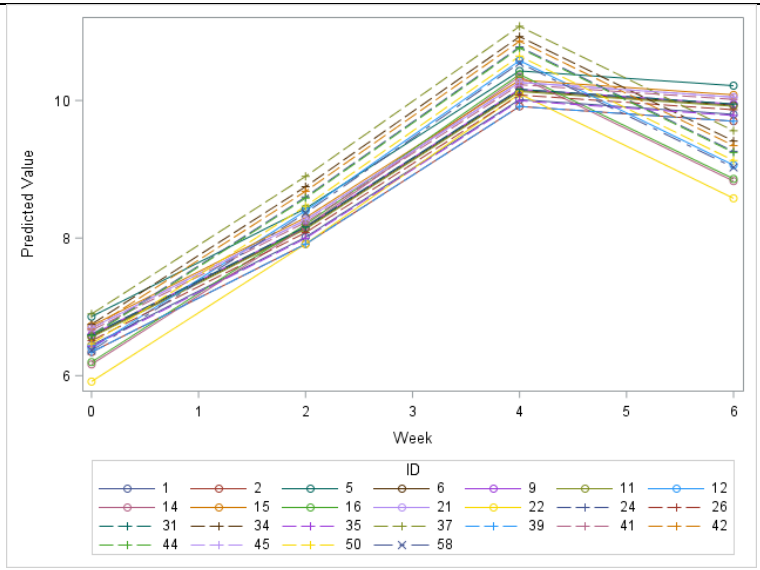


```
* Discrete time;
* Interaction: trt, age;
proc mixed data=cancer_long2
covtest;
class trt(ref="Placebo")
week(ref="0");
model totalc = age trt|week /
solution notest outpm=pred;
random intercept / subject=id
type=un;
run;
```

Solution for Fixed Effects

Effect	Treatment	WEEK	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept			5.5762	2.1694	22	2.57	0.0175
AGE			0.01665	0.03383	67	0.49	0.6243
TRT	Aloe juice		-0.1114	1.1826	67	-0.09	0.9253
TRT	Placebo		0	.	.	.	.
WEEK		2	1.5714	0.8775	67	1.79	0.0778
WEEK		4	3.5714	0.8775	67	4.07	0.0001
WEEK		6	3.3571	0.8775	67	3.83	0.0003
WEEK		0	0	.	.	.	.
TRT*WEEK	Aloe juice	2	0.4286	1.3228	67	0.32	0.7470
TRT*WEEK	Aloe juice	4	0.6104	1.3228	67	0.46	0.6460
TRT*WEEK	Aloe juice	6	-0.6938	1.3720	67	-0.51	0.6148
TRT*WEEK	Aloe juice	0	0	.	.	.	.
TRT*WEEK	Placebo	2	0	.	.	.	.
TRT*WEEK	Placebo	4	0	.	.	.	.
TRT*WEEK	Placebo	6	0	.	.	.	.
TRT*WEEK	Placebo	0	0	.	.	.	.

```
proc sgplot data=pred;
series x=week y=pred / markers
group=id;
xaxis label="Week";
yaxis label="Predicted Value";
run;
```



```
* GEE with exchangeable
covariance matrix;
proc genmod data=cancer_long;
class id trt(ref="Placebo");
model totalc = age trt stage
week;
repeated  subject=id /
type=exch covb corrw;
run;
```

Working Correlation Matrix				
	Col1	Col2	Col3	Col4
Row1	1.0000	0.1837	0.1837	0.1837
Row2	0.1837	1.0000	0.1837	0.1837
Row3	0.1837	0.1837	1.0000	0.1837
Row4	0.1837	0.1837	0.1837	1.0000

Analysis Of GEE Parameter Estimates

Empirical Standard Error Estimates

Parameter		Estimate	Standard Error	95% Confidence Limits		Z	Pr > |Z|
Intercept		3.3137	1.4753	0.4222	6.2053	2.25	0.0247
AGE		0.0349	0.0226	-0.0093	0.0791	1.55	0.1219
TRT	Aloe juice	-0.1900	0.7530	-1.6658	1.2858	-0.25	0.8008
TRT	Placebo	0.0000	0.0000	0.0000	0.0000	.	.
STAGE		0.8927	0.3214	0.2628	1.5227	2.78	0.0055
WEEK		0.5671	0.1292	0.3138	0.8204	4.39	<.0001

Chapter 18. Power and Sample Size Calculation

18.1. Statistical Power

- Power of a test
  - Probability of rejecting the null hypothesis (H_0)
  - Power = 1 – P(Type II error): Correctly reject H_0 and identify a truly significant result.
  - Determine the *usefulness* of a test.
  - Usually, 80% is considered a 'decent' power.
- Various factors affect the power of the test:
  - The larger the significance level (α), the higher power of the test.
  - The larger the sample size, the higher the power of the test.
  - The larger the size of the discrepancy between hypothesized and true values, the higher the power.

18.2. Power and Sample Size Analysis

- Optimize the design of a study. (Save money and time.)
- Improve chances of conclusive results with maximum efficiency.
- Achieve a desired balance between Type I and Type II errors.
- Minimize risks for subjects.
- Primary objectives
  - Determine the sample size to achieve a certain power.
  - Determine the power of a test for a given sample size.
  - Characterize the power of a study to detect a minimum meaningful effect.
- Planning for a future (prospective) study

Sample size	Description
Too small	Insufficient power Difference between groups is clinically important, but it may not reject H_0.
Too large	Excess power Difference between groups is not clinically important, but it may reject H_0.

- Parameters needed for power and sample size calculation[15]

Parameter	Description
Type I error (α)	Usually set at 5%. Power increases as α increases.
Standard deviation (σ)	Variance of the data If small, the power will be greater.
Effect size (Δ)	Minimum (clinically) significant difference (e.g. means, proportions) Big effect sizes are easier to detect and thus have greater power.
Sample size (*n*)	Usually driven by cost, time, etc.

Example: If we are interested in conducting a level α test and wish to have 100(1-β) % power,

Test	One-sample t-test (One-sided)	One-sample t-test (Two-sided)	Two-sample t-test[16] (Two-sided)
Sample size	$n \geq \left(\frac{(z_\alpha + z_\beta)\sigma}{\Delta}\right)^2$	$n \geq \left(\frac{(z_{\alpha/2} + z_\beta)\sigma}{\Delta}\right)^2$	$n \geq 2 \times \left(\frac{(z_{\alpha/2} + z_\beta)\sigma}{\Delta}\right)^2$

[15] Effect size and standard deviation are usually obtained through pilot studies or previously published data.

[16] Sample size in each group (Assume equal-sized groups)

18.3. PROC POWER

General Syntax

```
proc power <options>;
     <statements>  <options>;
run;
```

Statement	Description
ONESAMPLEFREQ	Tests, confidence interval precision, and equivalence tests of a single binomial proportion TEST = Z / EXACT
ONESAMPLEMEANS	One-sample t-test, confidence interval precision, or equivalence test TEST = T
PAIREDFREQ	McNemar's test for paired proportions DIST = NORMAL
PAIREDMEANS	Paired t-test, confidence interval precision, or equivalence test TEST = DIFF / EQUIV_DIFF
TWOSAMPLEFREQ	Chi-square, likelihood ratio, and Fisher's exact tests for two independent proportions TEST = PCHI / LRCHI / FISHER
TWOSAMPLEMEANS	Two-sample t-test, confidence interval precision, or equivalence test TEST = DIFF / EQUIV / DIFF_SATT

TWOSAMPLEWILCOXON	Wilcoxon-Mann-Whitney (rank-sum) test for 2 independent groups
TWOSAMPLESURVIVAL	Log-rank, Gehan, and Tarone-Ware tests for comparing two survival curves TEST = LOGRANK / GEHAN / TARONEWARE
ONEWAYANOVA	One-way ANOVA including single-degree-of-freedom contrasts TEST = OVERALL / CONTRAST
MULTREG	Tests of one or more coefficients in multiple linear regression
ONECORR	Fisher's z-test and t-test of (partial) correlation DIST = FISHERZ / T
PLOT	Display plots for previous sample size analysis

- PROC GLMPOWER: Prospective power and sample size analysis for linear models.
- For analyses not supported directly in SAS, write your own program.
- PASS: Specialized software for power and sample size analysis

Example

SAS Code

```
* One-sample t-test;
* Sample size calculation with power = 80%;
proc power;
onesamplemeans
test = t
mean = 5
stddev = 20
ntotal = .
power = 0.8
;
run;
```

```
* Two-sample t-test;
* Power calculation with sample size = 200;
proc power;
twosamplemeans
test = diff
meandiff = 5
stddev = 12
ntotal = 200
power = .
;
run;
```

```
* One-way ANOVA: Balanced groups;
* Power (overall test);
proc power;
onewayanova test = overall
groupmeans = 59|66|42
std = 12
nperg = 4
power = .
;
run;
```

Output

Fixed Scenario Elements	
Distribution	Normal
Method	Exact
Mean	5
Standard Deviation	20
Nominal Power	0.8
Number of Sides	2
Null Mean	0
Alpha	0.05

Computed N Total	
Actual Power	N Total
0.802	128

Fixed Scenario Elements	
Distribution	Normal
Method	Exact
Mean Difference	5
Standard Deviation	12
Total Sample Size	200
Number of Sides	2
Null Difference	0
Alpha	0.05
Group 1 Weight	1
Group 2 Weight	1

Computed Power
Power
0.834

Fixed Scenario Elements	
Method	Exact
Group Means	59 66 42
Standard Deviation	12
Sample Size Per Group	4
Alpha	0.05

Computed Power
Power
0.585

SAS Code

```
* Chi-squared test;
* Power calculation with a series of different npergroup;
proc power;
twosamplefreq test=pchi
grouppproportions = (0.6 0.8)
nullproportiondiff = 0
npergroup = 25 50 75 100 200
power = .;
run;
```

```
* Multiple linear regression;
proc power;
multreg
model = fixed
nfullpredictors = 7
ntestpredictors = 3
rsquarefull = 0.9
rsquarediff = 0.1
ntotal = .
power = 0.9;
run;
```

```
* Survival analysis;
* Compare two groups - based on median survivals;
proc power;
twosamplesurvival
accrualtime = 12
followuptime = 24
groupmedsurvtimes = 15 | 20 22 24
npergroup = .
power = 0.8
;
run;
```

Output

Fixed Scenario Elements	
Distribution	Asymptotic normal
Method	Normal approximation
Null Proportion Difference	0
Group 1 Proportion	0.6
Group 2 Proportion	0.8
Number of Sides	2
Alpha	0.05

Computed Power		
Index	N per Group	Power
1	25	0.335
2	50	0.590
3	75	0.767
4	100	0.876
5	200	0.993

Fixed Scenario Elements	
Method	Exact
Model	Fixed X
Number of Predictors in Full Model	7
Number of Test Predictors	3
R-square of Full Model	0.9
Difference in R-square	0.1
Nominal Power	0.9
Alpha	0.05

Computed N Total	
Actual Power	N Total
0.903	20

Fixed Scenario Elements	
Method	Lakatos normal approximation
Form of Survival Curve 1	Exponential
Form of Survival Curve 2	Exponential
Accrual Time	12
Follow-up Time	24
Group 1 Median Survival Time	15
Nominal Power	0.8
Number of Sides	2
Number of Time Sub-Intervals	12
Group 1 Loss Exponential Hazard	0
Group 2 Loss Exponential Hazard	0
Alpha	0.05

Computed N Per Group			
Index	Med Surv Time 2	Actual Power	N Per Group
1	20	0.801	273
2	22	0.800	158
3	24	0.802	108

Chapter 19. Introduction to PROC SQL

19.1. Standard Query Language (SQL)

- Standard language for relational database management systems
- Communicate with a database
  - Update data on a database.
  - Retrieve data from a database.
  - Request information from database to answer questions
- Common database management systems: Oracle, Sybase, Microsoft SQL Server, Access
- Standard commands: Select, Insert, Update, Delete, Create, Drop

19.2. PROC SQL

- Base SAS procedure: Combine the functionality of DATA and PROC steps in a *single* step.
  - Sort, summarize, subset, merge, and concatenate datasets.
  - Create new variables, or produce a new table.
  - Retrieve, update and report on information from SAS datasets or other database.
- PROC SQL can do the same task with fewer and shorter statements than traditional SAS code.
- It often uses fewer resources than conventional DATA and PROC steps.
- SAS has fewer data types than standard SQL.
  - Character
  - Numeric (numeric, decimal, integer, smallint, float, real, double, precision, and date)
- PROC SQL follows the guidelines set by the American National Standards Institute (ANSI).
- A SQL view is a stored SELECT statement executed at run time. (cf) NOPRINT)

General Syntax

```
proc sql <options>;
select column(s)
from table-name | view-name
where expression
group by column(s)
having expression
order by column(s);
quit;
```

- SQL statement
  - You can have as many SQL statements as you want in a single PROC SQL.

SQL Statement	Description
ALTER TABLE	Add, drop, and modify columns in a table.
CREATE	Build new tables, views, or indexes.
DELETE	Eliminate unwanted rows from a table or view.
DESCRIBE	Display table and view attributes.
DROP	Eliminate entire tables, views, or indexes.
INSERT	Add rows of data to tables or views.
RESET *<options>*	Add to or change PROC SQL options without re-invoking the procedure.
UPDATE	Modify data values in existing rows of a table or view.
JOIN *tables* on *variable(s)*	Merge datasets based on certain variable(s).

Example

Raw Data

sports

Obs	CustomerID	Name	Address
1	101	Murphy's Sports	115 Main St.
2	102	Sun N Ski	2016 Newberry Ave.
3	103	Sports Outfitters	19 Cary Way
4	104	Cramer & Johnson	4106 Arlington Blvd.
5	105	Sports Savers	2708 Broadway

pbc1

Obs	ID	Treatment	Age	Gender	Stage
1	1	1	58.7652	1	4
2	2	1	56.4463	1	3
3	3	1	70.0726	0	4
4	4	1	54.7406	1	4
5	9	1	42.5079	1	2

pbc2

Obs	ID	Ascites	Hepato	Spiders	Bili	Chol	Albu	Copp	Alka	SGOT	Trig	Platelet	Protime
1	1	1	1	1	14.5	261	2.6	156	1718	137.95	172	190	12.2
2	2	0	1	1	1.1	302	4.14	54	7394.8	113.52	88	221	10.6
3	3	0	0	0	1.4	176	3.48	210	516	96.1	55	151	12
4	4	0	1	1	1.8	244	2.54	64	6121.8	60.63	92	183	10.3
5	5	0	1	1	3.4	279	3.53	143	671	113.15	72	136	10.9

SAS Code

```
/* Create a table + Print */
proc sql;
create table work.sports0
(CustomerID num, Name char(17), Address char(20));
insert into work.sports0
 values (101, "Murphy's Sports", "115 Main St.")
 values (102, "Sun N Ski", "2016 Newberry Ave.")
 values (103, "Sports Outfitters", "19 Cary Way")
 values (104, "Cramer & Johnson", "4106 Arlington
Blvd.");
select * from work.sports0; quit;
```

Output

CustomerID	Name	Address
101	Murphy's Sports	115 Main St.
102	Sun N Ski	2016 Newberry Ave.
103	Sports Outfitters	19 Cary Way
104	Cramer & Johnson	4106 Arlington Blvd.

```
/* Concatenate two tables */
* Another table;
proc sql;
create table sports00
(CustomerID num, Name char(13), Address char(13));
insert into sports00
 values (105, "Sports Savers", "2708 Broadway");
quit;
* Concatenate + print;
proc sql;
create table sports as
select * from sports0
union all
select * from sports00;
select name, address from sports; quit;
```

Name	Address
Murphy's Sports	115 Main St.
Sun N Ski	2016 Newberry Ave.
Sports Outfitters	19 Cary Way
Cramer & Johnson	4106 Arlington Blvd.
Sports Savers	2708 Broadway

```
/* Read an existing table */
proc sql;
select name, address from sports
where customerID = 102; quit;
```

Name	Address
Sun N Ski	2016 Newberry Ave.

```
/* Create a new table (+ sort) + print */
proc sql;
create table sports3 as
select *, customerID - 100 as SimpleID,
substr(Name,1,1) as Initial
from sports
order by name;
select SimpleID, Initial, Address
from sports3; quit;
```

SimpleID	Initial	Address
4	C	4106 Arlington Blvd.
1	M	115 Main St.
3	S	19 Cary Way
5	S	2708 Broadway
2	S	2016 Newberry Ave.

```sql
/* Merge two tables */
proc sql;
create table pbc as
select *
from pbc1, pbc2
where pbc1.id = pbc2.id
order by pbc1.id;
select id, age, stage, hepato, albu
from pbc where id <= 5; quit;
```

ID	Age	Stage	Hepato	Albu
1	58.7652	4	1	2.6
2	56.4463	3	1	4.14
3	70.0726	4	0	3.48
4	54.7406	4	1	2.54
5	38.1054	3	1	3.53

```sql
/* Rename, Label, Format, New variable */
proc format;
value mffmt 0 = "Male"
            1 = "Female";
run;

proc sql;
create table pbc1_new as
select ID label = "Patient ID",
       stage as pbcstage,
       age format = 5.1,
       round(age) as age2,
       gender format = mffmt.
from pbc1;
select *
from pbc1_new
where pbcstage = 1; quit;
```

Patient ID	Stage	Age	age2	Gender
52	1	50.5	51	Male
58	1	44.6	45	Male
65	1	40.2	40	Female
98	1	28.9	29	Female
102	1	56.6	57	Female
153	1	49.6	50	Female
174	1	55.6	56	Female
206	1	62.0	62	Female
218	1	34.6	35	Female
258	1	51.5	51	Female
272	1	38.4	38	Female
285	1	46.3	46	Female
61	1	43.9	44	Male
73	1	38.5	38	Female
107	1	62.5	63	Female
150	1	35.0	35	Female

```
/* Having, Group by */
proc sql;
select ID, age, stage, hepato
from pbc
where Trig <= 200 AND 3.5 <= Albu <= 6
group by stage, hepato, id
having 11 <= Protime <= 14;
quit;
```

ID	Age	Stage	Hepato
61	43.8987	1	0
73	38.4942	1	0
153	49.6044	1	0
206	61.9904	1	0
25	45.0732	2	0
90	33.4757	2	0
93	36.5339	2	0
104	43.0171	2	0
135	42.9678	2	0
89	52.4435	2	1