

Why Dummy Variable Makes You **SMART**, and How to Do it **SEXY**



By Brian Sun, M.Sc.
Rick Hansen Institute
May 4, 2011



Predictive Modeler Using
SAS® Enterprise Miner™ 5



Categorical Variable as IV

- ▶ Model Length of Stay (LOS) in Hospital ~ Age + Injury Level
(high, medium, low)

- ▶ Class Statement

```
proc glm data=patient;  
  class injury_level;  
  model LOS=age injury_level;  
quit;
```

- ▶ Dummy Variable Injury_high = $\begin{cases} 1 & \text{if injury_level="high"} \\ 0 & \text{otherwise} \end{cases}$

```
proc glm data=patient;  
  model LOS=age injury_high injury_medium;  
quit;
```

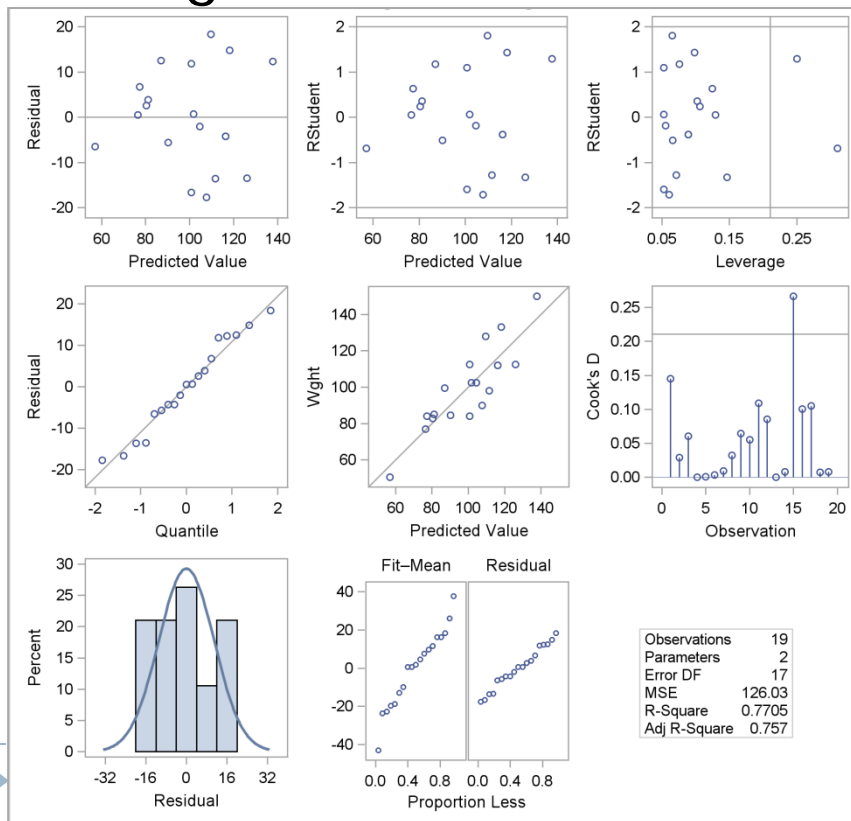
Theoretically, no difference in term of estimation

Big Fan of Proc Reg?

```
ods graphics;  
proc reg data=patient;  
    model LOS=age injury_high injury_medium;  
quit;
```

<= No Class Statement

▶ Diagnostics Plot



▶ Model Selection Options

Backward/forward/stepwise/
MaxR/MinR/RSquare/CP/AdjRSQ

▶ Allow Multiple Models

You can do multiple model
statements in one procedure

▶ Proc Reg V.S. Proc Glm

Take Full Control of Baseline

```
proc glm data=patient;  
  class injury_level;  
  model LOS=age injury_level;  
quit;
```

<= No control of baseline

Recode the variable

Injury_level {
 "High" => "3-High"
 "Medium" => "2-Medium"
 "Low" => "1-Low"

- ▶ More flexibility to control if you do dummy variable.

```
proc glm data=patient;  
  model LOS=age injury_high injury_medium;  
quit;
```

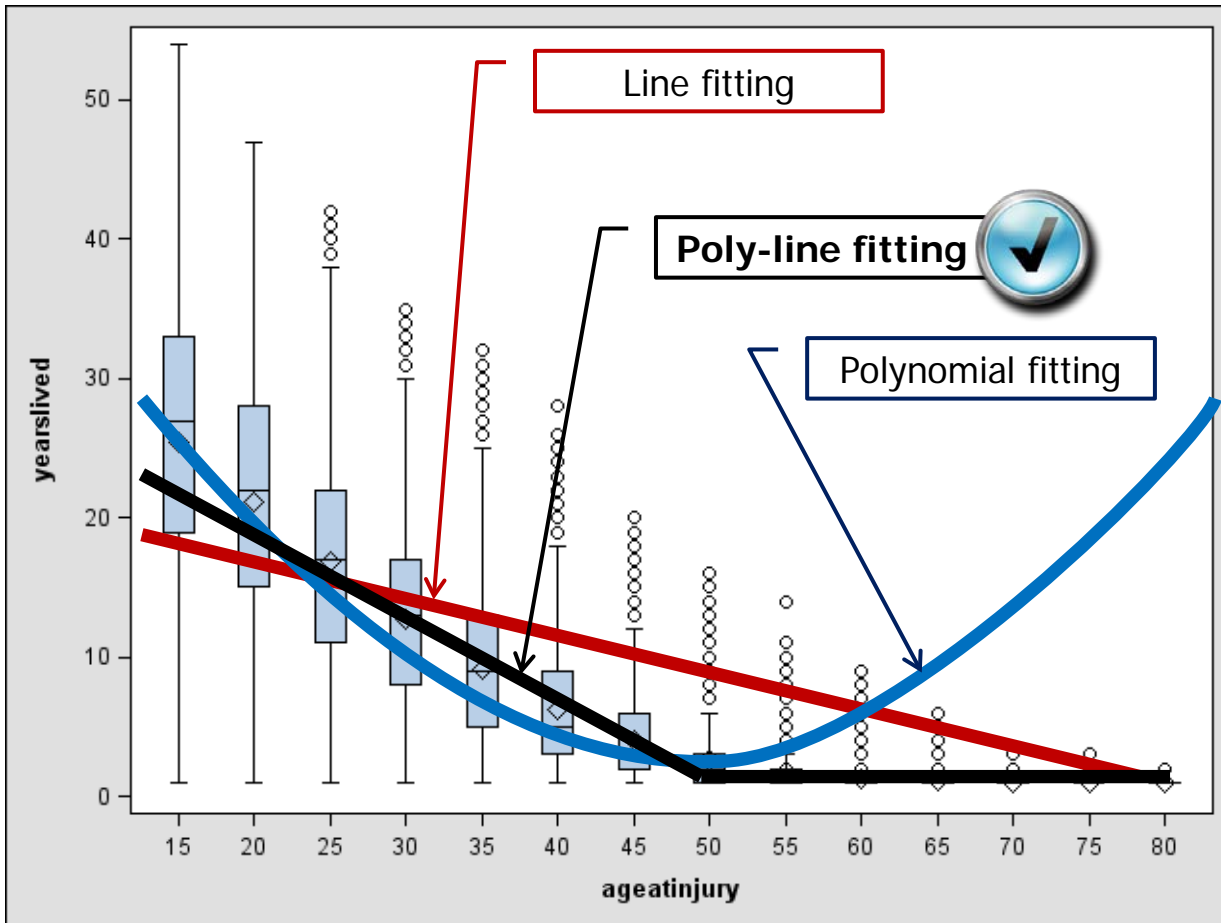
<= "Low" is the baseline

```
proc glm data=patient;  
  model LOS=age injury_high;  
quit;
```

<= "Low and medium" is the baseline

Make the Fitting Smart

Years Lived After Injury ~ Age at Injury + Age_50 + Age_50_inter + Other Covariates



$$\text{Age}_{50} = \begin{cases} 1 & \text{if age at injury} \\ & \leq 50 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Age}_{50_inter} = \text{Age}_{50} * \text{Age}$$

Adjusted R-SQ

Linear fitness: 0.52

Polynomial fitness: 0.63

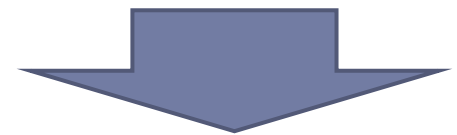
Poly-line fitness: 0.75

Code the Dummy Variable

```
Data patient;  
  set patient_raw;  
  if gender="male" then gender_male=1;  
    Else gender_male=0;  
  if gender="female" then gender_female=1;  
    Else gender_female=0;  
  if education="Primary School"  
    then edu_primary=1;  
    Else edu_primary=0;  
  if education="Middle School"  
    then edu_Mid=1;  
    Else edu_Mid=0;  
  if education="High School"  
    then edu_hig=1;  
    Else edu_hig=0;  
  if education="University Diploma"  
    then edu_UDiploma=1;  
    Else edu_UDiploma=0;  
  .....
```

It's so tedious!

How to do it SEXY?



SAS Macro!

The Automated “Toy”

▶ Key Features

- Automatically detecting all levels of categorical variable, create dummy variables accordingly.
- Automatically naming and labeling the dummy variables
- Automatically making dummy variables for all character/numerical variables in a table.

```
%Auto_Dummy_Variable (
```

```
    tablename=,
```

```
    variablename=,
```

```
    outtablename=,
```

```
    missing=,
```

```
    MaxLevel=,
```

```
    delimiter=
```

```
);
```

▶ Syntax

→ Input table name

→ Input variable name OR
ALL, _NUM_, _CHAR_

→ Output table name

→ Whether code missing value as dummy
variable: Yes, No (default)

→ Max number of levels considered for coding
dummy variables, the default is 10

→ Delimiter for coding multiple options text
variable

Example of Use

- ▶ create dummy variables for one variable

```
%Auto_Dummy_Variable (tablename=patient, variablename=gender, outtablename=patient);
```

- ▶ create dummy variables for every variable in a table

```
%Auto_Dummy_Variable (tablename=patient, variablename=_All_, outtablename=patient);
```

If the number of distinct values is greater than 10, the variable would be automatically excluded from generating dummy variables

- ▶ create dummy variable for multiple options text variable

```
%Auto_Dummy_Variable (tablename=patient, variablename=complications,  
outtablename=patient, delimiter=|);
```

Example:

Complications="Pneumonia | Neuro Pain"



Complications_Pneumonia=1

Complications_NeuroPain=1

Key Note 1 – Macro Array

- ▶ Automatically detect all levels of categorical variable, and create dummy variables accordingly.

Get all level texts by Proc SQL, join texts of both variable name and level name to make dummy variable name

Gender	dummy_name
female	gender_female
male	gender_male

Table: level_list

Generate macro arrays for level text and dummy variable name text from Level_list table

Looping the macro matrix, and generating corresponding if then clauses in data step.

```
Proc SQL noprint;  
  create table level_list as  
  select distinct &variablename.,  
  cats("&variablename._",&variablename) as  
  dummy_name  
  from &tablename.; quit;
```

```
Data _null_;  
  set level_list;  
  call symputx('level' || left(put(_n_,8.)),&variablename.);  
  call symputx('dummysname' || left(put(_n_,8.)),dummy_name);  
  call symputx('N_level',left(put(_n_,8.)));  
  quit;
```

```
Data &outputtable.;  
  set &tablename.;  
  %do i=1 %to &N_level;  
    if &variablename.="&&level&i"  
    then &&dummysname&1.=1;  
    Else &&dummysname&1.=0;  
  %end;  
run;
```

Key Note 2 – Use of function

▶ Automatically name dummy variables

Rule of Automatic Naming: Variablename_LevelText

○ Limitation of SAS name length: 32

example: variable name – ComplicationsAtFollowUP (24 bytes)

level text – “Pneumonia” (9 bytes)

Solution: trim text if length exceed 32

```
dummy_name = trim(substr(cats("& variablename._",&variablename.),1,min(32,length(cats("& variablename._",&variablename.)))));
```

○ Blank, special characters in level text

example: variable name – WorkExp level text – “20++ years”

Solution: remove all blank and special characters with compress function

```
dummy_name = compress(dummy_name, 'adk')
```

Modifiers: a – alphabetic

d – number

k – keep

Question

