simon-5507-04-slides

Topics to be covered

- What you will learn
  - Convert string to numeric
  - Labels for number codes
  - Drawing bar charts
  - Creating categories
  - Modifying categories
  - Crosstabulations

Review definitions

- Categorical
  - Small number of possible values
  - Each value associated with a category
- Continuous
  - Large number of possible values
  - Potentially any value in an interval

Speaker notes

Before we start, let’s review a couple of definitions.

A **categorical variable** is a variable that can only take on a small number of values. Each value is usually associated with a particular category.

Examples of categorical variables are

- sex (Male or Female),
- race (White, Black, Native American, etc.),
- cancer stage (I, II, III, or IV),
- birth delivery type (Vaginal, C-section).

A **continuous variable** is a variable that can take on a large number of possible values, potentially any value in some interval.

Examples of continuous variables are

- Birth weight in grams,
- gestational age,
- fasting LDL level.

There are some variables that are on the boundary between categorical and continuous, but it is not worth quibbling about here.

The point to remember is that the types of graphs that you use and the types of statistics that you compute are dependent on many things, but first and foremost on whether the variables are categorical, continuous, or a mixture.

Today, you will see examples involving mostly categorical variables.

Documenting your program (1)

```
* 5507-04-simon-demo.sas
  author: Steve Simon
  date: created 2018-10-22
  purpose: to work with categorical variables
  license: public domain;

* datasets created in this program
    titanic, original data
    titanic1, age converted to numeric
```

Speaker notes

- Comments on the code: Documenting your program

This is my standard documentation.;

File locations (2)

```
%let path=q:/5507-2025b/04;

ods pdf
  file="&path/results/5507-04-simon-demo.pdf";

filename raw_data
  "&path/data/titanic.txt";

libname perm
  "&path/data";
```

Speaker notes

- Comments on the code: File locations

These are my standard file locations.;

The import procedure (3)

```
proc import
    datafile=raw_data
    out=perm.titanic
    dbms=dlm
    replace;
  delimiter='09'x;
  getnames=yes;
run;
```

Speaker notes

- Comments on the code: The import procedure

As a general rule, proc import works best for simple delimited files where the first row contains the variable names. With complicated text files (such as files where the data for an individual extends across more than one line) or files without variable names in the first row are usually better handled by a data step.

There is a small problem with the imported data as you shall see below.;

The print and contents procedures (4)

```
proc print
    data=perm.titanic(obs=10);
  title1 "The first ten rows of the Titanic dataset";
  footnote1 "Created by Steve Simon on &sysdate using SAS &sysver";
run;

proc contents
    data=perm.titanic;
  title1 "Internal description of Titanic dataset";
run;
```

Speaker notes

- Comments on the code: The print and contents procedures

Notice how the age variable is left justified. Most of the time, strings are left justified, but age is a numeric variable. Or it should have been. This is caused by the non-standard (non-standard to SAS) missing value code of NA, as noted in the data dictionary. It would have been easier to anticipate this ahead of time, but we'll fix things up after the fact.;

Using the input function (5)

```
data perm.titanic1;
  set perm.titanic;
  age_c = input(age, ?? 8.);
run;
```

Speaker notes

- Comments on the code: Using the input function

For the one continuous variable (age) you need to convert the code “NA” to the SAS missing value code, which is a dot. The easiest way to do this is to force the data to numeric with a simple arithmetic equation like adding a zero. But you get a warning message for each occurence of NA, which can get tedious. The input function with two question marks avoids this issue.;

Break #1

- What you have learned
  - Convert string to numeric
- What's coming next
  - Labels for number codes

Coding categories

- Use number codes (1, 2, 9)
- Use single letter codes (F, M, U)
- Use strings (Female, Male, Unspecified)

Speaker notes

When you have categorical data, you can choose how to represent it. There are several possible choices, each with advantages and disadvantages.

Number codes are useful for controlling the order in which the categories are displayed. If you want to present tables and graphs with males frist, then females, then unspecified, then assign the codes accordingly. The problem with number codes is consistency. If half of your data entry team use 1 for males and half uses 1 for females, you have a serious problem.

Single letter codes are easy to type and easy to remember. A problem occurs with the default in SAS and most other software. They will use alphabetical order to display your data in tables and graphs. Putting females before males might be okay, but it might not. There is also a potential for ambiguity. Your might be okay with W for White, but A could refer to African-American, American Indian, or Asian.

Strings are very easy to remember and avoid many of the ambiguity and inconsistency problems with other representation. The typing can be a bit tedious and error prone.

With both letter codes and strings, please be consistent with capitalization. Most of SAS is fine with upper case, lower case, or a mix. But not with category codes. Capital F female is a different category that lower case F female.

If you use number codes, you should always use proc format to insure that your output is labeled clearly and unambiguously.

The format procedure (6)

```
proc format;
  value f_survived
    0 = "No"
    1 = "Yes";
run;

proc freq
    data=perm.titanic1;
  tables Survived / nocum;
  format Survived f_survived.;
  title1 "More than half of all the passengers died";
run;
```

Speaker notes

- Comments on the code: The format procedure

You define category values using the format procedure. This is similar to the value labels in SPSS and the factor function in R.

The value subcommand designates codes and assigns those codes to a format name, in this case, f_survived. You can use any reasonable name. I choose to prefix any category format with f_.

Once category codes are defined, you specify which variable or variables use those codes with a format statement. This insures that the number codes are replaced by the proper labels.;

Break #2

- What you have learned
  - Labels for number codes
- What’s coming next
  - Drawing bar charts

Drawing bar charts

- Many features are important
  - Horizontal bars are often better
  - Adjust aspect ratio to keep bars thin

Speaker notes

Bar charts can display counts or percentages for your categorical data. The default options for these charts should almost always be modified.

For most bar charts a horizontal orientation is often better. The category labels fit better and are easier to read. A horizontal bar chart will often fit better on the page of a manuscript.

There are exceptions. Charts with many bars, especially with clustering (coding two or more catgorical variables) will often be better in a vertical orientation.

Try things both ways if you are unsure.

The aspect ratio (the ratio the the height of the graph to the width) is also important. Try to avoid a small number of very fat bars. For a horizontal bar chart this often means an aspect ratio less than 1. How much less depends on how many bars you have. For two or three bars, a height of about 2 inches to a width of about 6 inches works well.

Again, try different options and see which one looks best.

Bar charts (7)

```
ods graphics / height=1.5 in width=6 in;

proc sgplot
    data=perm.titanic1;
  hbar Sex /
      nofill
      stat=percent;
  xaxis
      label = " "
      values = (0 0.2 0.4 0.6 0.8 1)
      valuesdisplay = ("0%" "20%" "40%" "60%" "80%" "100%")
      max = 1;
  yaxis label = " ";
  title1 "Most of the passengers were male";
```

Speaker notes

- Comments on the code: Bar charts

The ods graphics options controls some general options for the graphs produced by SAS.

The height and width options control the vertical and horizontal dimentions of the graphing window.

The hbar subcommand draws horizontal bars.

The nofill option uses no color for the interior of the bar.

The stat=percent makes the length of the bars equal to the percentage.;

Break #3

- What you have learned
  - Drawing bar charts
- What’s coming next
  - Creating categories

Creating categories

- Converting continuous variables
  - Ease of interpretation
  - Loss of information

Speaker notes

Often you may be asked to convert a continuous variable into a categorical variable.

Creating categories, number codes (8)

Speaker notes

```
data age_codes; set perm.titanic1; if age_c = . then age_cat = 9; else if age_c < 6 then age_cat = 1; else if age_c < 13 then age_cat = 2;
else if age_c < 21 then age_cat = 3; else age_cat = 4; run;
```

The format procedure (9)

```
* Comments on the code: Creating categories, number codes

Create new categories using the

  if ... then ... else subcommand.

Create a series of conditions that can
evaluate to true or false and assign
codes. Always account for missing
values first.;
```

Speaker notes

```
proc format; value f_age 1 = “toddler” 2 = “pre-teen” 3 = “teenager” 4 = “adult” 9 = “unknown”; run;
```

Sort the data for a quality check; (10)

```
* Comments on the code: The format procedure

The format statement attaches a label to each number code.;
```

Speaker notes

```
proc sort data=age_codes; by age_cat; run;
```

Compute ranges for each category; (11)

```
* Comments on the code: Sort the data for a quality check;
```

Speaker notes

```
proc means min max data=age_codes; by age_cat; var age_c; format age_cat f_age.; title1 “Quality check for conversion”; title2 “Coding
appears to be fine”; run;
```

Quality check, 2 (12)

```
* Comments on the code: Compute ranges for each category;
```

Speaker notes

- Comments on the code: Quality check, 2

Always check the new categorical variable against the range of values of the original continuous variable.;

Break #4

- What you have learned
  - Creating categories
- What's coming next
  - Modifying categories

04-05

Modifying categories (13)

```
data first_class;
  set perm.titanic1;
  if PClass = "1st"
    then first_class = "Yes";
    else first_class = "No ";
run;

proc freq
    data=first_class;
  table PClass*first_class /
    norow nocol nopercent;
run;
```

Speaker notes

- Comments on the code: Modifying categories

Here’s how you combine categories for a categorical variable. It uses the same “if - then - else” code.;

Break #5

- What you have learned
  - Modifying categories
- What’s coming next
  - Crosstabulations

Crosstabulations

Default crosstabulation (14)

```
proc freq
    data=perm.titanic1;
  tables Sex*Survived;
  format Survived f_survived.;
  title1 "Crosstabulation with all percentages";
run;
```

Speaker notes

- Comments on the code: Default crosstabulation

To examine relationships among categorical variables use a two dimensional crosstabulation.

Notice that SAS provides three different percentages. I do NOT recommend that you show every percentage.;

Row percentages (15)

```
proc freq
    data=perm.titanic1;
  tables Sex*Survived / nocol nopercent;
  format Survived f_survived.;
  title1 "Crosstabulation with row percentages";
run;
```

Speaker notes

- Comments on the code: Row percentages

You get row percentages by excluding column percentages (nocol) and cell percentages (nopercent).

Notice that among the men, 83% died and 17% survived. 83% and 17% adds up to 100% within that row. Among the women 33% died and 67% survived. 33% and 67% addes up to 100% within that row.;

Column percentages (16)

```
proc freq
    data=perm.titanic1;
  tables Sex*Survived / norow nopercent;
  format Survived f_survived.;
  title1 "Crosstabulation with column percentages";
run;
```

Speaker notes

- Comments on the code: Column percentages

You get column percentages by excluding row percentages (norow) and cell percentages (nopercent).

Column percentages add up to 100% within each column. Among those who died, 18% were women and 82% were men. 18% and 82% adds up to 100%. Among the survivors, 68% were women and 32% were men. 68% and 32% adds up to 100%.;

Cell percentages (17)

```
proc freq
    data=perm.titanic1;
  tables Sex*Survived / norow nocol;
  format Survived f_survived.;
  title1 "Crosstabulation with cell percentages";
run;

ods pdf close;
```

Speaker notes

- Comments on the code: Cell percentages

You get cell percentages by excluding row percents (norow) and column percents (nocol).

Cell percentages add up to 100% across the entire table. There were 12% female deaths among all the passengers, 54% male deaths, 23% female survivors, and 11% male survivors. These four percentages (12%, 54%, 23%, and 11%) all add up to 100%. Which is best: row, column, or cell percents. The answer is “it depends.” I have a handout that talks about these issues.;

Summary

- What you have learned
  - Convert string to numeric
  - Labels for number codes
  - Drawing bar charts
  - Creating categories
  - Modifying categories
  - Crosstabulations