Basic Concepts #5
Customizing I/O Format by Proc Format

JC Wang
Outline

1. PROC FORMAT
   - Overview
   - User-Defined Format
   - User-Defined Informat
   - Numeric Value Printing Template

2. PROC Format Statement
   - Format Library
   - Input Control Data Set

3. Examples
   - Selected Examples
Three statements for user-defined informats/formats

- Use `VALUE` statement to create user-defined format.
- Use `INVALUE` statement to create user-defined informat.
- Use `PICTURE` statement to create template for printing SAS numeric values.
Three statements for user-defined informats/formats

- Use **VALUE** statement to create user-defined format.
- Use **INVALU** statement to create user-defined informat.
- Use **PICTURE** statement to create template for printing SAS numeric values.
Three statements for user-defined informats/formats

- Use `VALUE` statement to create user-defined format.
- Use `INVALUE` statement to create user-defined informat.
- Use `PICTURE` statement to create template for printing SAS numeric values.
Three statements for user-defined informats/formats

- Use `VALUE` statement to create user-defined format.
- Use `INVALUE` statement to create user-defined informat.
- Use `PICTURE` statement to create template for printing SAS numeric values.
User-Defined Format: Value Statement

VALUE <$>name <(format-option(s))> 
<value-range-set(s)>;

- name is a format name which is a valid SAS name with additional restrictions—*not ending with digits*, starting with a $ for character format (so that character format name is at most 31 characters long following the leading $), and must *not* be a SAS-supplied format name.

- Format options include DEFAULT=, MIN=, and MAX=; FUZZ=; NOTSORTED; and MULTILABEL.

- Each value-range-set is in the following form: 
  value-or-range='formatted-value'|[existing-format] 
  where *formatted-value*, enclosed in matching quotation marks, can be up to 32,767 characters long. (Note, however, some procedures print only the first 8 or 16 characters.)
User-Defined Format: Value Statement

VALUE <$(name (format-option(s)))>
<value-range-set(s)>;

- name is a format name which is a valid SAS name with additional restrictions—*not ending with digits*, starting with a $ for character format (so that character format name is at most 31 characters long following the leading $), and must *not* be a SAS-supplied format name.

- Format options include DEFAULT=, MIN=, and MAX=; FUZZ=; NOTSORTED; and MULTILABEL.

- Each value-range-set is in the following form:
  value-or-range='formatted-value'|[existing-format]
where formatted-value, enclosed in matching quotation marks, can be up to 32,767 characters long. (Note, however, some procedures print only the first 8 or 16 characters.)
User-Defined Format: Value Statement

VALUE <$>name <(format-option(s))> <value-range-set(s)>;

- *name* is a format name which is a valid SAS name with additional restrictions—*not ending with digits*, starting with a $ for character format (so that character format name is at most 31 characters long following the leading $), and must *not* be a SAS-supplied format name.

- Format options include `DEFAULT=`, `MIN=`, and `MAX=`; `FUZZ=`; `NOTSORTED`; and `MULTILABEL`.

- Each *value-range-set* is in the following form:
  
  `value-or-range='formatted-value'|[existing-format]`

  where *formatted-value*, enclosed in matching quotation marks, can be up to 32,767 characters long. (Note, however, some procedures print only the first 8 or 16 characters.)
**User-Defined Format: Value Statement**

\[
\text{VALUE } <\$> \text{name } <(\text{format}-\text{option}(s))> \\
<\text{value-range-set}(s)>;
\]

- *name* is a format name which is a valid SAS name with additional restrictions—*not ending with digits*, starting with a $ for character format (so that character format name is at most 31 characters long following the leading $), and must *not* be a SAS-supplied format name.

- Format options include `DEFAULT=`, `MIN=`, and `MAX=`; `FUZZ=`; `NOTSORTED`; and `MULTILABEL`.

- Each *value-range-set* is in the following form: `value-or-range='formatted-value'|[existing-format]` where *formatted-value*, enclosed in matching quotation marks, can be up to 32,767 characters long. (Note, however, some procedures print only the first 8 or 16 characters.)
**Value-Range-Set**

Value-range-set is one or more occurrences of value-or-range separated by comma in which

- value is a single value (numeric or character string) or keyword `OTHER` (numeric only for PICTURE statement)
- range is a range of values (numeric values or characters) or keywords `LOW` and `HIGH`, use `<` to exclude end point(s) (numeric variable only for PICTURE statement)

**Examples:**
1. `LOW,12-15` (i.e., minimum, or values between 12 & 15, inclusive)
2. `'M'‐<HIGH` (Strings start from M and greater (lexical order) to less than maximum)
3. `0<‐<10` (the open interval `(0,10)`)
4. `'m'‐'z~','M'‐'Z~` (strings lead by m or M or higher) the use of `~` is significant (otherwise, it stops at z or Z (single letter))
Value-Range-Set

Value-range-set is one or more occurrences of value-or-range separated by comma in which

- **value** is a single value (numeric or character string) or keyword **OTHER** (numeric only for PICTURE statement)
- **range** is a range of values (numeric values or characters) or keywords **LOW** and **HIGH**, use `<` to exclude end point(s) (numeric variable only for PICTURE statement)

Examples:

1. LOW,12-15 (i.e., minimum, or values between 12 & 15, inclusive)
2. ‘M’-<HIGH (Strings start from M and greater (lexical order) to less than maximum)
3. 0<->10 (the open interval (0,10))
4. ‘m’-’z~’,’M’-’Z~’ (strings lead by m or M or higher) the use of ~ is significant (otherwise, it stops at z or Z (single letter))
Value-Range-Set

Value-range-set is one or more occurrences of value-or-range separated by comma in which

- **value** is a single value (numeric or character string) or keyword OTHER (numeric only for PICTURE statement)
- **range** is a range of values (numeric values or characters) or keywords LOW and HIGH, use < to exclude end point(s) (numeric variable only for PICTURE statement)

Examples:

1. LOW,12-15 (i.e., minimum, or values between 12 & 15, inclusive)
2. ’M’-<HIGH (Strings start from M and greater (lexical order) to less than maximum)
3. 0<-<10 (the open interval (0,10))
4. ’m’-’z~’,’M’-’Z~’ (strings lead by m or M or higher) the use of ~ is significant (otherwise, it stops at z or Z (single letter))
Value-Range-Set

Value-range-set is one or more occurrences of value-or-range separated by comma in which

- **value** is a single value (numeric or character string) or keyword `OTHER` (numeric only for PICTURE statement)
- **range** is a range of values (numeric values or characters) or keywords `LOW` and `HIGH`, use `<` to exclude end point(s) (numeric variable only for PICTURE statement)

Examples:

1. LOW,12-15 (i.e., minimum, or values between 12 & 15, inclusive)
2. ’M’-<HIGH (Strings start from M and greater (lexical order) to less than maximum)
3. 0<-<10 (the open interval (0,10))
4. ’m’-’z∼’,’M’-’Z∼’ (strings lead by m or M or higher) the use of ∼ is significant (otherwise, it stops at z or Z (single letter))
**User-Defined Informat: Invalue Statement**

```
INVALUE <$(name (informat-option(s)))>
<value-range-set(s)>;
```

- `name` is an informat name which is a valid SAS name with additional restrictions—at most 31 characters long, not ending with digits, starting with a $ for character format (so that character format name is at most 30 characters long following the leading $), and must not be a SAS-supplied informat name. Detail: in SAS log, the character ’@’ prefixes a user-defined informat.
- Informat options include `DEFAULT=`, `MIN=`, and `MAX=`; `FUZZ=`; `NOTSORTED`; `JUST` and `UPCASE`.
- Each `value-range-set` is in the following form: `value-or-range=informatted-value|[existing-informat]`.
  - `informatted-value` can be ’character-string’ (enclosed in matching quotation marks, and can be up to 32,767 characters long), `number`, `_ERROR_`, and `_SAME_`. 
User-Defined Informat: Invalue Statement

INVALUE <$(name<(informat-option(s))>)>(value-range-set(s));

- *name* is an informat name which is a valid SAS name with additional restrictions—at most 31 characters long, not ending with digits, starting with a $ for character format (so that character format name is at most 30 characters long following the leading $), and must not be a SAS-supplied informat name. Detail: in SAS log, the character ’@’ prefixes a user-defined informat.

- Informat options include DEFAULT=, MIN=, and MAX=; FUZZ=; NOTSORTED; JUST and UPCASE.

- Each *value-range-set* is in the following form:

  value-or-range=informatted-value|[existing-informat]

- *Informatted-value* can be ’character-string’ (enclosed in matching quotation marks, and can be up to 32,767 characters long), *number*, _ERROR_, and _SAME_.

wang (WMU) Basic Concepts 5 Lecture 7 6 / 12
User-Defined Informat: INVALUE Statement

INVALUE <$>name <(informat-option(s))> <value-range-set(s)>;

- *name* is an informat name which is a valid SAS name with additional restrictions—at most 31 characters long, not ending with digits, starting with a $ for character format (so that character format name is at most 30 characters long following the leading $), and must not be a SAS-supplied informat name. Detail: in SAS log, the character '@' prefixes a user-defined informat.
- Informat options include *DEFAULT=*, *MIN=*, and *MAX=*; *FUZZ=*, *NOTSORTED*, *JUST* and *UPCASE*.
- Each *value-range-set* is in the following form: *value-or-range=*informed-value|[existing-informat] informed-value can be *character-string* (enclosed in matching quotation marks, and can be up to 32,767 characters long), *number*, _ERROR_, and _SAME_.

wang (WMU) Basic Concepts 5
Lecture 7 6 / 12
User-Defined Informat: Invalue Statement

INVALUE <$(name (<(informat-option(s))>))> <value-range-set(s)>;

- *name* is an informat name which is a valid SAS name with additional restrictions—at most 31 characters long, not ending with digits, starting with a $ for character format (so that character format name is at most 30 characters long following the leading $), and must not be a SAS-supplied informat name. Detail: in SAS log, the character '@' prefixes a user-defined informat.

- Informat options include `DEFAULT=`, `MIN=`, and `MAX=`; `FUZZ=`; `NOTSORTED`; `JUST` and `UPCASE`.

- Each *value-range-set* is in the following form:
  - `value-or-range=informatted-value|[existing-informat]`
  - `informatted-value` can be `'character-string'` (enclosed in matching quotation marks, and can be up to 32,767 characters long), `number`, `_ERROR_`, and `_SAME_`. 
Numeric Value Printing Template: Picture Statement

PICTURE name <(format-option(s))>
<value-range-set-1 <(picture-1-option(s))>
<...value-range-set-n <(picture-n-option(s))>>;

- Format options include DEFAULT=, MIN=, and MAX=; FUZZ=; NOTSORTED; MULTILABEL; DATATYPE=; DECSEP= and DIG3SEP; ROUND; MULTIPLIER= (or MULT=), FILL=, NOEDIT and PREFIX=. (Note: MULT is applied first, then ROUND, and last the format.)

- Each value-range-set is in the following form:
  value-or-range-1 <...value-or-range-n>=’picture’

  picture is a character string (at most 40 characters) enclosed in match quotation marks. Detail — 3 types of characters: digit selectors (up to 16 digit selectors), message selectors (printed as is), and directives (used with DATATYPE= format option only).
Numeric Value Printing Template: Picture Statement

PICTURE name (format-option(s)) >
<value-range-set-1 (picture-1-option(s))>
<... value-range-set-n (picture-n-option(s))>>>>;

- Format options include DEFAULT=, MIN=, and MAX=; FUZZ=; NOTSORTED; MULTILABEL; DATATYPE=; DECSEP= and DIG3SEP; ROUND; MULTIPLIER= (or MULT=), FILL=, NOEDIT and PREFIX=. (Note: MULi is applied first, then ROUND, and last the format.)

- Each value-range-set is in the following form:
  value-or-range-1 <..., value-or-range-n>=’picture’
  picture is a character string (at most 40 characters) enclosed in match quotation marks. Detail — 3 types of characters: digit selectors (up to 16 digit selectors), message selectors (printed as is), and directives (used with DATATYPE= format option only).
Numeric Value Printing Template: Picture Statement

PICTURE name <(format-option(s))>
<value-range-set-1 <(picture-1-option(s))> 
<...value-range-set-n <(picture-n-option(s))>>>

- Format options include DEFAULT=, MIN=, and MAX=; FUZZ=; NOTSORTED; MULTILABEL; DATATYPE=; DECSEP= and DIG3SEP; ROUND; MULTIPLIER= (or MULT=), FILL=, NOEDIT and PREFIX=. (Note: MULT is applied first, then ROUND, and last the format.)

- Each value-range-set is in the following form:
  value-or-range-1 <..., value-or-range-n>=’picture’

picture is a character string (at most 40 characters) enclosed in match quotation marks. Detail — 3 types of characters: digit selectors (up to 16 digit selectors), message selectors (printed as is), and directives (used with DATATYPE= format option only).
Outline

1. PROC FORMAT
   - Overview
   - User-Defined Format
   - User-Defined Informat
   - Numeric Value Printing Template

2. PROC Format Statement
   - Format Library
   - Input Control Data Set

3. Examples
   - Selected Examples
Permanent Format Library

- Created by
  
  ```
  PROC FORMAT LIBRARY=libref<.catalog>
  ```

- Referenced using `FMTSEARCH=` system option.
Permanent Format Library

- Created by
  
  \[ \text{PROC FORMAT LIBRARY=libref<.catalog>} \]

- Referenced using \textit{FMTSEARCH=} system option.
Permanent Format Library

- Created by
  PROC FORMAT LIBRARY=libref.<.catalog>
- Referenced using FMTSEARCH= system option.
Input Control Data Set

User-defined formats can be created at run time by
PROC FORMAT CNTLIN=input-control-SAS-data-set;
Outline

1. PROC FORMAT
   - Overview
   - User-Defined Format
   - User-Defined Informat
   - Numeric Value Printing Template

2. PROC Format Statement
   - Format Library
   - Input Control Data Set

3. Examples
   - Selected Examples
Selected Examples

- Grouping/recoding data
- Tallying missing data
- Displaying date/time/datetime values using directives
- Displaying a variable several ways all at once
- Table lookup
- Creating permanent format catalog
- Reporting negative percent values
- Control data sets
- Creating/Using Multilabel Format

See example ProcFormat.sas
Selected Examples

- Grouping/recoding data
- Tallying missing data
- Displaying date/time/datetime values using directives
- Displaying a variable several ways all at once
- Table lookup
- Creating permanent format catalog
- Reporting negative percent values
- Control data sets
- Creating/Using Multilabel Format

See example ProcFormat.sas
Selected Examples

- Grouping/recoding data
- Tallying missing data
- Displaying date/time/datetime values using directives
- Displaying a variable several ways all at once
- Table lookup
- Creating permanent format catalog
- Reporting negative percent values
- Control data sets
- Creating/Using Multilabel Format

See example ProcFormat.sas