11

Para saber mais

Aqui estão algumas informações adicionais interessantes sobre o que vimos em aula:

- No `PROC FREQ` vimos como fazer o filtro a partir de comandos adicionais na chamada da base (logo após o nome da base), colocando o filtro todo e os critérios entre parênteses, e colocando um "igual" após o `where`, na forma `(where = (<critério do filtro>))`. Mas este procedimento também possui a opção de receber um parâmetro (como o parâmetro `table`) que justamente é um filtro, de uma forma ainda mais parecida com o que usamos em um *data step*. Ele segue a seguinte estrutura:

```
PROC FREQ
  data=alura.cadastro_produto;
  where data = .;
  table nome;
RUN;
```

  Basta colocar como uma linha de comando adicional, e neste caso não são colocados nem parênteses nem o igual.

- Vimos como fazer atribuições de valores de variáveis diretamente, mas podemos também criar variáveis a partir de outras variáveis, basta usar o nome da variável que desejo utilizar na minha conta. Inclusive posso realizar operações matemáticas, como `+` (adição), `-` (subtração), `*` (multiplicação), `/` (divisão) e `**` (exponenciação). Por exemplo, se eu quero que a `variavel2` receba o valor da `variavel1` ao quadrado, posso escrever a seguinte expressão:

```
variavel2 = variavel1**2;
```

  Apenas precisamos ter cuidado em escrever corretamente o nome das variáveis, pois podemos chegar em um erro se tentarmos usar uma variável que não exista ainda na base, ou sobrescrever uma variável da base se usarmos um nome igual.

- Atribuímos valores numéricos a variáveis numéricas, mas como podemos atribuir um texto a uma variável do tipo caractere? Por exemplo, se eu quiser que minha variável de lançamento possua os valores `Sim` e `Não` ao invés de zero ou um? Sabemos que se eu escrever `flag_lancamento = Sim;` eu provavelmente obterei um erro pois dificilmente tenho uma variável chamada `Sim` na minha base. Para resolver este problema é só colocar meu texto entre aspas, assim o SAS sabe que é um texto e não um comando ou o nome de uma variável:

```
if data > 201606
  then flag_lancamento2 = "Sim";
  else flag_lancamento2 = "Não";
```

- Vimos que para variáveis numéricas o *missing* é representado como um ponto. Nas variáveis de texto (do tipo caractere) ele é representado simplesmente como um espaço em branco. Juntando isso com o fato de que colocamos os textos entre aspas (como vimos acima), se queremos filtrar os *missings* de uma variável do tipo caractere, utilizaríamos o código abaixo:

```
where variavel_caractere = " ";
```

- Vimos que se eu quiser realizar mais de uma operação que dependa de um único condicional eu posso usar a estrutura

```
if <condição> then do;
  <diversas operações>
end;
```

Mas podemos usar esse comando de `do;` inclusive depois do meu comando `else` quando eu quiser realizar diversas operações inclusive quando minha condição inicial não for atendida (no meu "senão"), da seguinte forma:

```
if <condição> then do;
  <diversas operações>
end;
else do;
  <diversas operações>
end;
```

Repare que eu sempre preciso usar o comando `end;` para indicar onde o comando `do;` acaba.