# PH614: 02. Data wrangling

Professional Skills for Public Health

2022

## Learning Objectives

- Create new data sets
- Rename variables
- Select variables
- Drop variables
- Understand operators
- Filter data with logic
- Creating new variables
- Sort data

## The Data step

- SAS lets you create new data sets each time you use the **DATA** step
- Typical structure of the DATA step is:

```
*This is a blank temporary dataset;
DATA new_data;
RUN;
```

- New data sets not prefixed with a library name will be temporary!
- To make a permanent dataset make sure the library you want it to be in exists then...

```
DATA PH614.new_data;
RUN;
```

## Creating a new data set from an existing data set

- Often we use existing data as the basis for our dataset in SAS. To do this we use the **SET** keyword

```
*Create a new data set;
DATA PH614.new_data2;
SET PH614.NHANES_mini;
RUN;

*Look at your log and next run PROC CONTENTS to see if the new dataset was created correctly;
PROC CONTENTS DATA=PH614.new_data2;
RUN;
```

- Note every time you use the **DATA** step you create a new dataset and if the name already exists it will overwrite the existing dataset!!! Watch out for this.

## Rename Variables

- There are times you want your variable to have a more representative name

- You can rename the variables, the caveat is there can only be alphabets, numbers, and underscores in the variable's names and they should only begin with letters
- Use the **RENAME** keyword followed by old_var_name=new_var_name
- You can have multiple of these rename pairs as you need

```
DATA PH614.new_data3;
SET PH614.NHANES_mini;
RENAME Sex=Gender Ethnicity=Race Education=Tranining;
RUN;

*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data3;
RUN;
```

## Keeping variables

- There are times when you are not interested in all the variables in a dataset
- You can decide to keep some out of several variables in a new dataset without losing your original dataset
- To keep some variables use the keyword **KEEP**

```
DATA PH614.new_data4;
SET PH614.NHANES_mini;
KEEP Age Sex Race Education;
RUN;

*Look at your log and next run PROC CONTENTS to see what variables are kept;
PROC CONTENTS DATA=PH614.new_data4;
RUN;
```

## Dropping variables

- This is the opposite of keeping and serves where you want to drop a few variables you are not interested in
- The **DROP** keyword is what is required to achieve this

```
DATA PH614.new_data5;
SET PH614.NHANES_mini;
DROP Weight Height;
RUN;

*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data5;
RUN;
```

## Combining variable manipulation

- You can combine the variable manipulation steps as required.
- Make sure you end each step with a semicolon

```
DATA PH614.new_data6;
SET PH614.NHANES_mini;
KEEP Age Sex Race Education Weight Height;
RENAME Sex=Gender;
RUN;
```

```
*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data6;
RUN;
```

## Operators

- SAS lets you carry out a couple of operations
- First, let's use the assignment operator = to create a new variable and set its value

```
DATA PH614.new_data7;
SET PH614.NHANES_mini;
Test = 4; *Numbers are written without quotes;
Annother_test = "Hello"; *Non-numbers are written with quotes;
RUN;

*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data7;
RUN;
```

## Mathematical operators

- SAS supports mathematical operators including +, -, /, *
- SAS also supports operator precedence (PEMDAS)

- let's explore this

```
DATA PH614.new_data8;
SET PH614.NHANES_mini;
BMI = Weight/(Height*Height);
RUN;

*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data8;
RUN;
```

## SAS IF-THEN-ELSE logic

- Apart from mathematical operators, SAS supports comparison operators such as >, <, >=, <=
- SAS as a programing language has provisions for logic and decision points
- Using this construct we can check for truth and decide to do something ifa properpere condition is met
- the typical construct uses the words **IF** condition **THEN** do something; Let's see an example

```
DATA PH614.new_data9;
SET PH614.NHANES_mini;
IF Age < 18 THEN Age_group = "Child";
ELSE IF 18 <= Age <= 60 THEN Age_group = "Adult";
ELSE Age_group = "Elder";
RUN;

*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data9;
RUN;
```

- The statement may be written without the **ELSE IF** and **ELSE** portions

## Filtering data using the IF statement

- We can filter based on a variable using the IF statement

```
DATA PH614.new_data10;
SET PH614.NHANES_mini;
IF Sex = 'Male';
RUN;

*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data10;
RUN;
```

## Filtering data using the WHERE clause

- We can filter based on a variable using the IF statement

```
DATA PH614.new_data11;
SET PH614.NHANES_mini;
WHERE Sex = 'Male';
RUN;

*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data11;
RUN;
```

## Sorting datasets

- To sort a dataset you use the **PROC** command not the DATA command
- You can determine which variable to sort on

```
PROC SORT DATA=PH614.new_data10;
BY Age;
RUN;

*Look at your log and next run PROC CONTENTS to see what changed;
PROC CONTENTS DATA=PH614.new_data10;
RUN;
```

## Review of Learning Objectives

- Create new data sets
- Rename variables
- Select variables
- Drop variables
- Understand operators
- Creating new variables
- Filter data with logic
- Sort data

## Q&A

## Next...

- Univariate Analysis
- PROC UNIVARIATE

- PROC MEANS
- PROC FREQ
- Formatting variables