PharmaSUG 2019 - Paper DV-276

Not That Dummy Data

Yuliia Bahatska, PRA Health Sciences;

ABSTRACT

When it comes to data visualization it is expected that the data is presented the way it is stored in the analysis dataset. Labels may be changed, formats may be applied, or some derivations may be made, but definitely programmers are not expected to add the values to the data. But is it always true? SAS® has quite a wide range of ways to create figures and usually at least one of the graphical procedures or Graph Template Language (GTL) can provide you with the required output, but sometimes you still need a workaround. In this paper we will concentrate on the cases when in order to present the data in the desired way it is helpful to add artificial or dummy values.

INTRODUCTION

Figures are often used to present the data in all sorts of posters, publications and manuscripts. In this case rather than using a pre-defined shell it is a back-and-forth process with a statistician when a programmer is constantly asked to change little things. And while an update might seem minor to the person who did not write the program, sometimes a tiny alteration in the appearance requires quite a lot of code rewriting because it turns out that the procedure you are using cannot do that one thing or the two options you need are not compatible. In such situations two scenarios are possible. You can tell your statistician that the update is impossible to make, or you can be creative and consider it as a challenge. In this paper I would like to share with you the ways I used to cope with such challenges.

ADDING DESCRIPTIVE STATISTICS TO A SERIESPLOTST

I would start with an example, you've probably already come across. I had a seriesplot of subjects' individual values over time. This is the way my code looked like:

```
proc template;
 define statgraph seriesplot;
  begingraph;
   entrytitle "Subject values over time";
   layout overlay / xaxisopts=(offsetmin=.05 label='Visit')
                    yaxisopts=(label='Value');
     seriesplot x=VISIT y= VALUE / group=USUBJID name="subjects";
     discretelegend "subjects";
   endlayout;
  endgraph;
 end;
run;
```

Then the statistician asked me to add the descriptive statistics. The easiest way was to make a set of the original dataset and the dataset with descriptive statistics and to create one variable that combined USUBJID and descriptive statistics name and the other that would combine individual values. This way the datasets below:

USUBJID	VISIT	VALUE
Subject 1	Visit 1	Value 1
Subject 1	Visit 2	Value 2
Subject 1	Visit 3	Value 3…

Table 1: Individual values

STAT_NAME	VISIT	STAT
Mean	Visit 1	Value 4
Mean	Visit 2	Value 5
Mean	Visit 3	Value 6…

Table 2: Descriptive statistics

turned to:

USUBJID	VISIT	Value	STAT_NAME	STAT	USUBJID_STAT	VALUE_STAT
Subject 1	Visit 1	Value 1			Subject1	Value 1
Subject 1	Visit 2	Value 2			Subject 1	Value 2
Subject 1	Visit 3	Value 3			Subject 1	Value 3
	Visit 1		Mean	Value 4	Mean	Value 4
	Visit 2		Mean	Value 5	Mean	Value 5
	Visit 3		Mean	Value 6	Mean	Value 6

Table 3: Values and statistics combined

And the code required minimum modifications:

```
proc template;
  define statgraph seriesplot;
   begingraph;
    entrytitle "Subject values over time with descriptive statistics";
    layout overlay / xaxisopts=(offsetmin=.05 label='Visit')
                     yaxisopts=(label='Value');
      seriesplot x=VISIT y=VALUE_STAT / group=USUBJID_STAT name="subjects";
      discretelegend "subjects";
    endlayout;
   endgraph;
  end;
run;
```

Below is a primitive example with just four subjects and five visits that illustrates the general idea. The upper figure is just the individual subject data and the figure in the bottom has the descriptive statistics added so that individual values can be compared against the average.

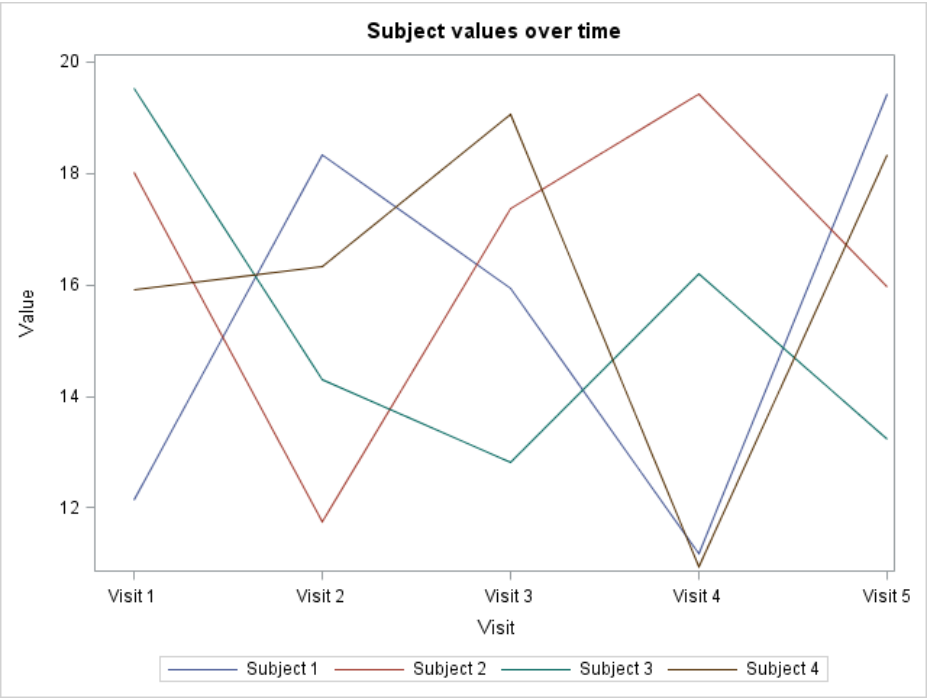

Figure 1: Subject values over time

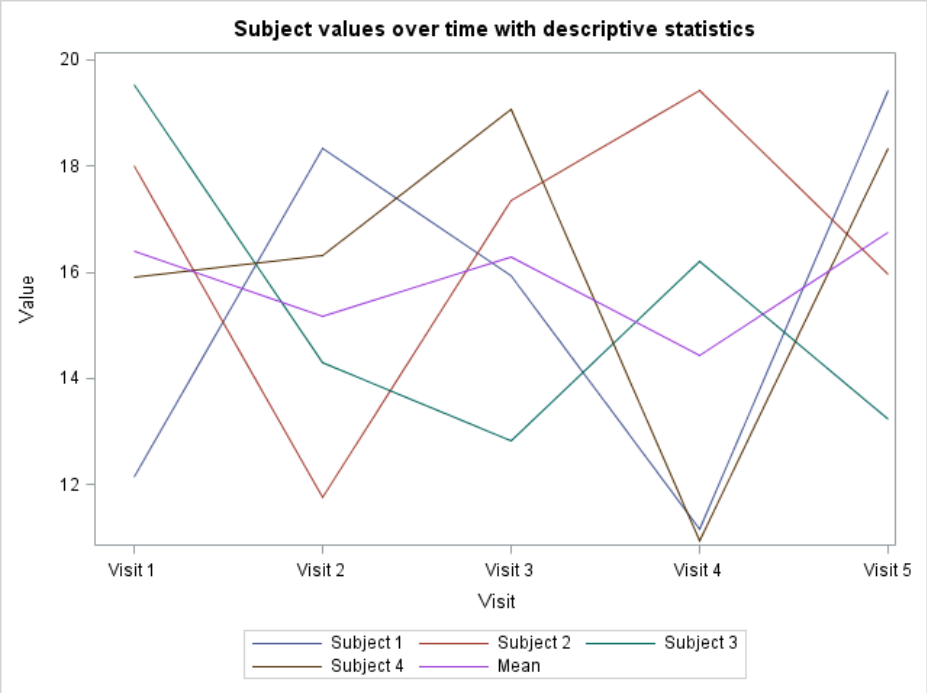

Figure 2: Subject values over time with descriptive statistics

SERIESPLOT WITH DIFFERENT COLORS FOR ONE LINE

In this section I would like to share with you the situation I faced while creating another seriesplot. The visits in this study were split into two periods and the request from the statistician was not only to have a different color for each subject but to have a different color for each subject for each period. I decided to use the SERIESPLOT in GTL to create lines and the SCATTERPLOT to plot the values. In order to have two different colors per subject I added a new variable USUBJID_PERIOD that combined USUBJID and PERIOD.

USUBJID	VISIT	VALUE	PERIOD	USUBJID_PERIOD
Subject 1	Visit 1	Value 1	1	Subject 1,1
Subject 1	Visit 2	Value 2	1	Subject 1,1
Subject 1	Visit 3	Value 3	2	Subject 1,2…

Table 4: Individual values by period and visit

I then passed the new variable to the GROUP:

```
proc template;
 define statgraph seriesplot;
  begingraph;
   entrytitle "Subject values over time split by period";
   layout overlay / xaxisopts=(offsetmin=.05 label='Visit')
                    yaxisopts=(label='Value');
      seriesplot x=VISIT y=VALUE / group=USUBJID_PERIOD name='subjects';
      scatterplot x=VISIT y=VALUE / group=USUBJID_PERIOD name='scatter';
      mergedlegend 'subjects' 'scatter';
   endlayout;
  endgraph;
 end;
```

```
run;
```

The result was not yet the one I needed as the values between period 1 and period 2 are not connected. Moreover, as you can see from the figure below, Subject 2 has no value on Visit 3.

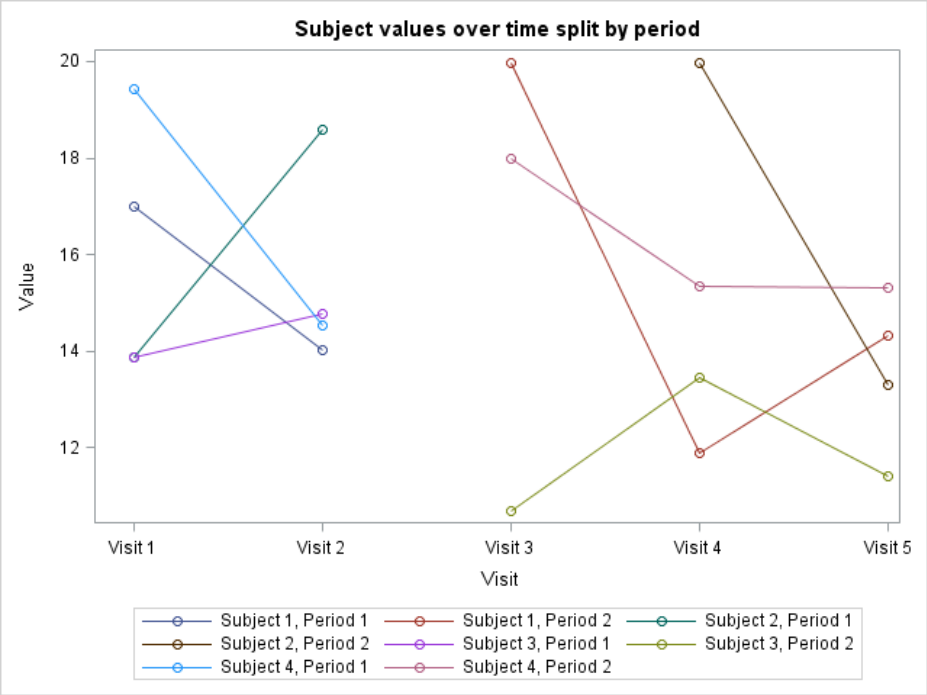

Figure 3: Individual values, periods are not connected

In order to have the connection lines between period 1 and period 2 I added artificial values to my data. As everything before visit 3 in my study was considered as period 1, I needed these lines to have the color of period 1. So, for each subject that had both period 1 and 2 values I duplicated the record with VISIT=Visit 3 with PERIOD=1 and then rederived USUBJID_PERIOD. I also created a new variable VALUE_SCATTER that was empty for artificial values so that the symbols in scatterplot don't get overlaid.

USUBJID	VISIT	VALUE	VALUE_SCATTER	PERIOD	USUBJID_PERIOD
Subject 1	Visit 1	Value 1	Value 1	1	Subject 1,1
Subject 1	Visit 2	Value 2	Value 2	1	Subject 1,1
Subject 1	**Visit 3**	**Value 3**	**.**	**1**	**Subject 1,1**
Subject 1	Visit 3	Value 3	Value 3	2	Subject 1,2…

Table 5: Individual values and artificial values

My only problem now was Subject 2. As there was no value for Visit 3, there was nothing to duplicate, but I still needed to have the connection line and moreover, it needed to change color at Visit 3. Given the values for two points (Visit 2 and Visit 4) it was possible to calculate the equation of the line. So, if the coordinates of two points are (x_1, y_1) and (x_2, y_2) the equation looks like:

$$\frac{x-x_1}{x_2-x_1}=\frac{y-y_1}{y_2-y_1}$$

Passing the values from Visit 2 and Visit 4 I got the y value for the point where the colors needed to change. In this particular example Visit 3 is in the exact middle between Visit 2 and Visit 4 which means I could have used the formula for the middle of the line segment, but the formula above works for all potential cases. I added two records for Visit 3 with the derived value: one for period 1 and one for period

2 and that was it. VALUE_SCATTER was also missing for these two records. After applying the exact same code with just one line modified:

```
scatterplot x=VISIT y=VALUE_SCATTER / group=USUBJID_PERIOD name='scatter';
```

I got the required result:

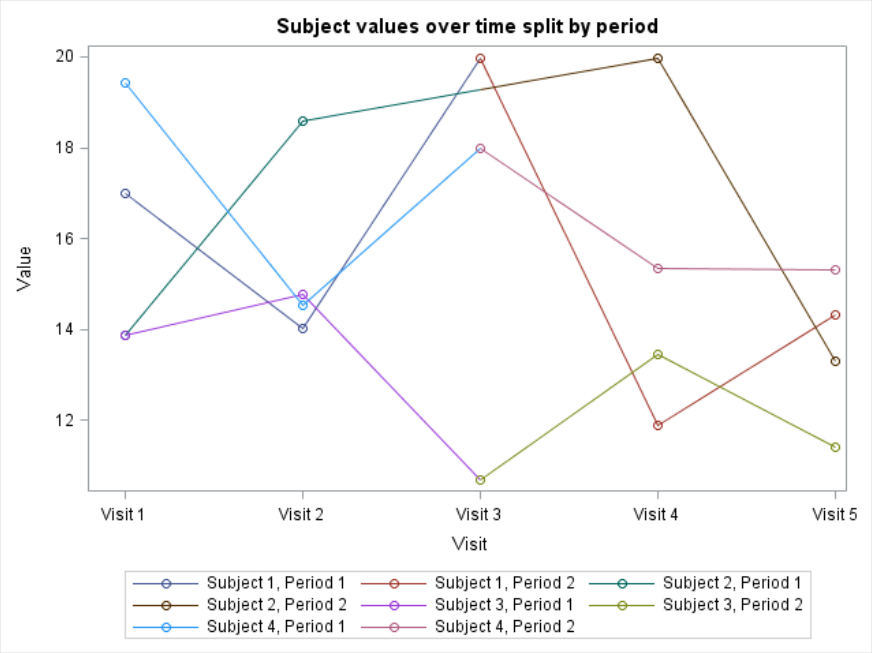

Figure 4: Individual values, periods are connected

Note how the line for Subject 2 changes color at Visit 3.

DISPLAYING ALL VALUES IN THE LEGEND FOR SG-PROCEDURES

It is so easy to create a figure when all the required values are present in the data. Incomplete data can become a nightmare for programmers in case they are asked to display all the values in the legend. If they are using GTL it is possible using the DISCRETEATTRMAP statement in combination with the TYPE= option, however if for some reason GTL cannot be used, dummy values could solve this problem. I needed to create a figure that would show distribution of age group by gender and although one of the age groups was not present in the data, I needed to have it in the legend. Here to make the understanding easier I will use dataset SASHELP.CLASS and the age groups: 10-12, 13-16, 17-20. The last group is not present in the data (just like in SASHELP.CLASS there are no students in 17-20 group):

SEX	AGEGROUP	COUNT
F	10-12	3
F	13-16	6
M	10-12	4
M	13-16	6

Table 6: Counts for age group by gender

By the person who was making the code review I was asked to use the SGPLOT procedure. So, when I ran my code:

```
proc sgplot data=counts;
  styleattrs datacolors=(green red blue);
  vbar sex/response=COUNT group=AGEGROUP stat=sum;
```

```
    xaxis display=(nolabel);
    yaxis grid label='Count';
run;
```

I didn't have age group 17-20 in the legend:

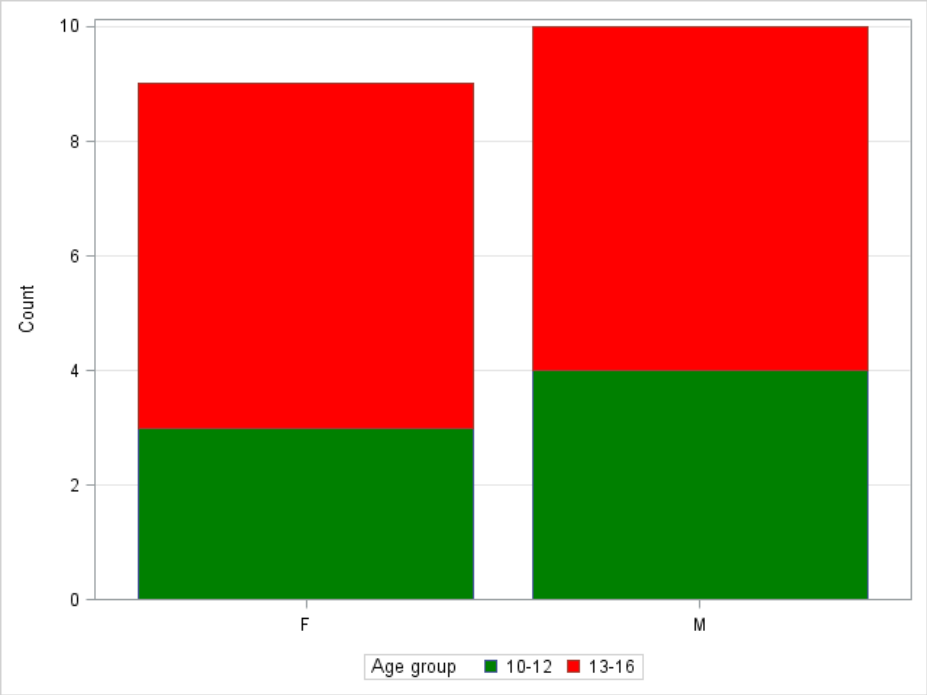

Figure 5: Age group by gender

I added dummy values to my dataset so that each group would be present. Note that for 17-20 count is missing, so nothing extra is displayed in the plot:

SEX	AGEGROUP	COUNT
F	10-12	.
F	13-16	.
F	17-20	.
F	10-12	3
F	13-16	6
M	10-12	4
M	13-16	6

Table 7: counts for age group by gender with dummy values

And after running the same code I got the desired result:

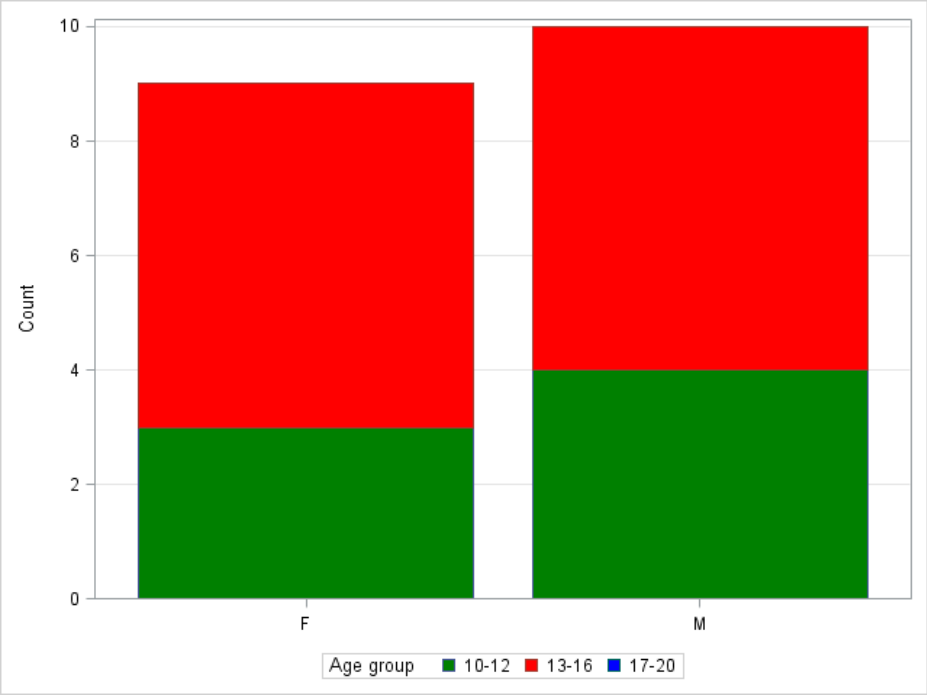

Figure 6: Age group by gender, complete legend

DISPLAYING ALL VALUES IN THE MERGED LEGEND FOR GTL

Switching to the GTL and using the DISCRETEATTRMAP statement in combination with the TYPE= option in the previous example would have solved the problem. However, this solution also has its limitations. In case the MERGEDLEGEND statement needs to be used, it will not work. In one of the projects I worked on I needed to create two plots with the linear regression line together with the individual values by gender. Here for illustration I will again use SASHELP.CLASS dataset and the age groups: 10-12, 13-14, 15-20. In order to make it applicable for my purpose I will only include male students with age group 15-20, so that there are no female students in this group. The code I used:

```
proc template;
  define statgraph mergedLegend;
    begingraph;
      layout overlay;
        scatterplot x=WEIGHT y=HEIGHT / group=AGEGROUP name='scatter';
          regressionplot x=WEIGHT y=HEIGHT / group=AGEGROUP name=line;
        mergedlegend 'scatter' 'line';
      endlayout;
    endgraph;
 end;
run;

proc sgrender data=class template=mergedLegend;
      by SEX;
run;
```

produced the following figures:

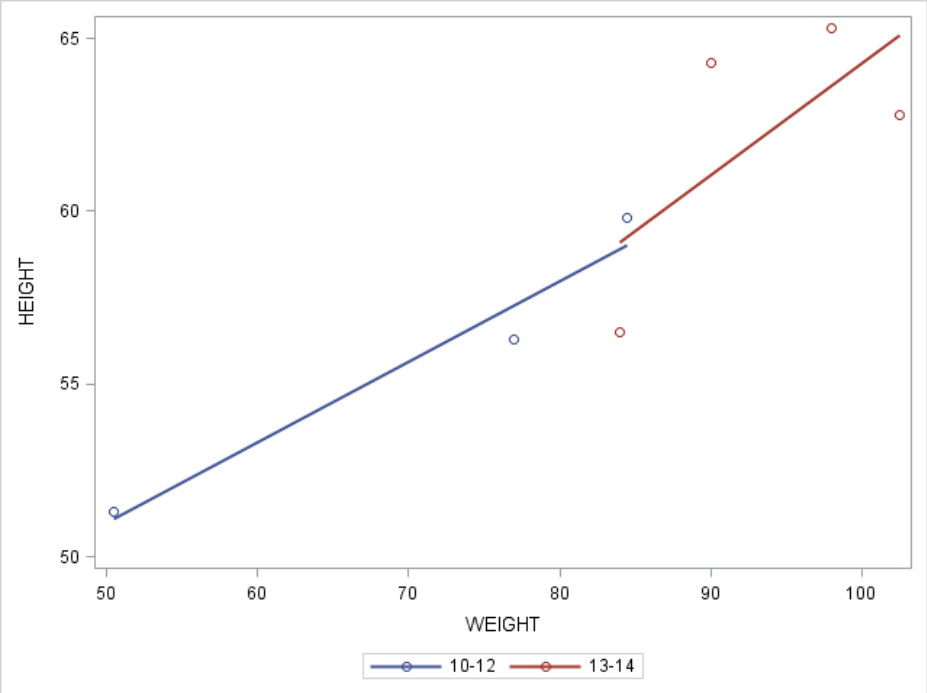

Figure 7: Male, incomplete legend

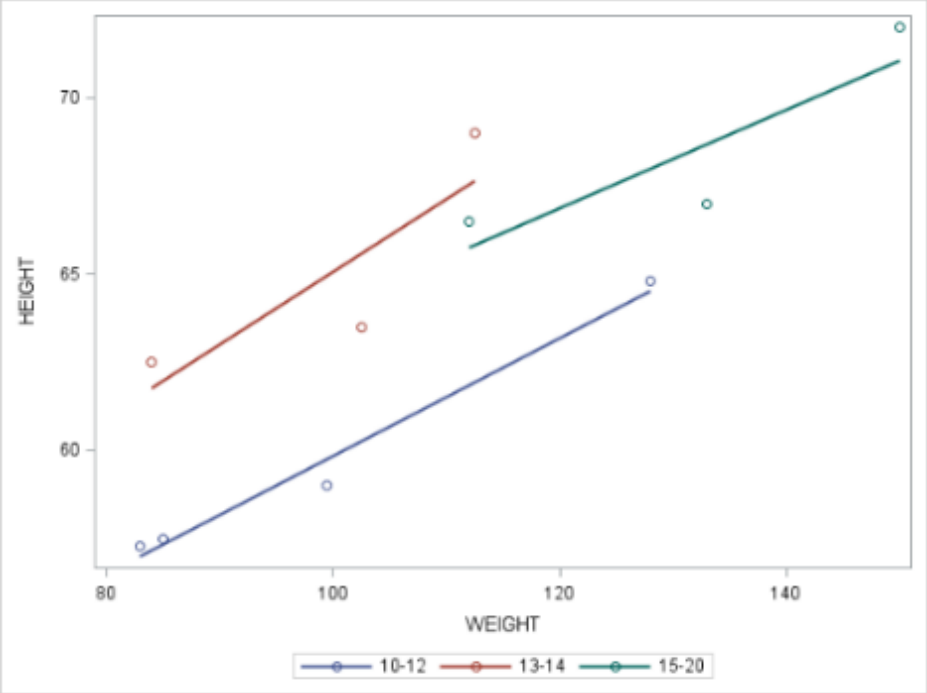

Figure 8: Female, complete legend

and the problem was that the statistician wanted to have all groups present in the legend for both of them. To achieve this, I added 6 rows with empty HEIGHT and WEIGHT variables to my dataset: one for each gender and age group. And when I ran the same code:

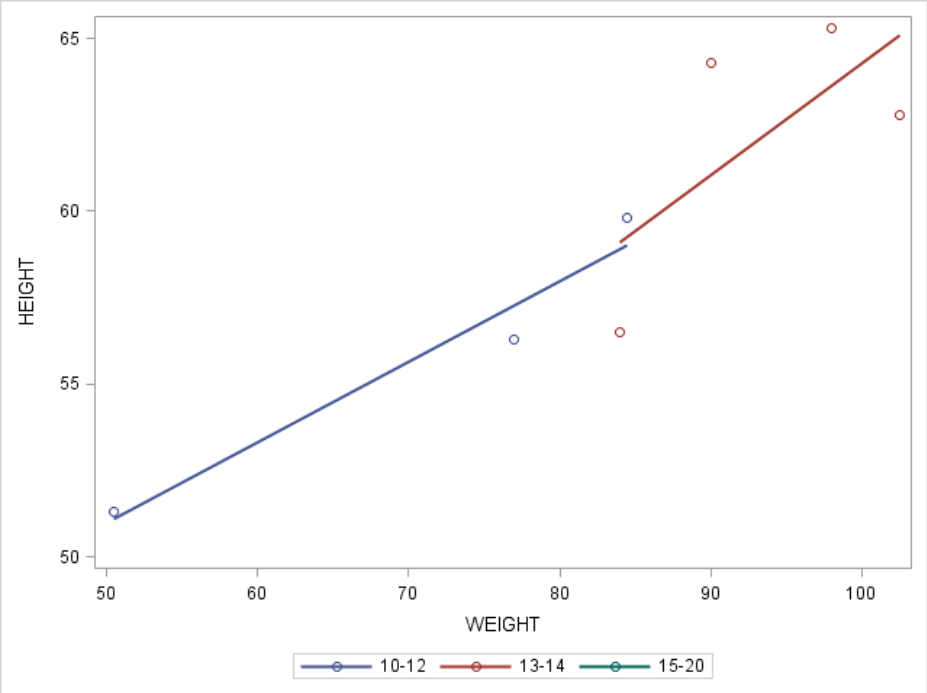

Figure 9: Male, complete legend

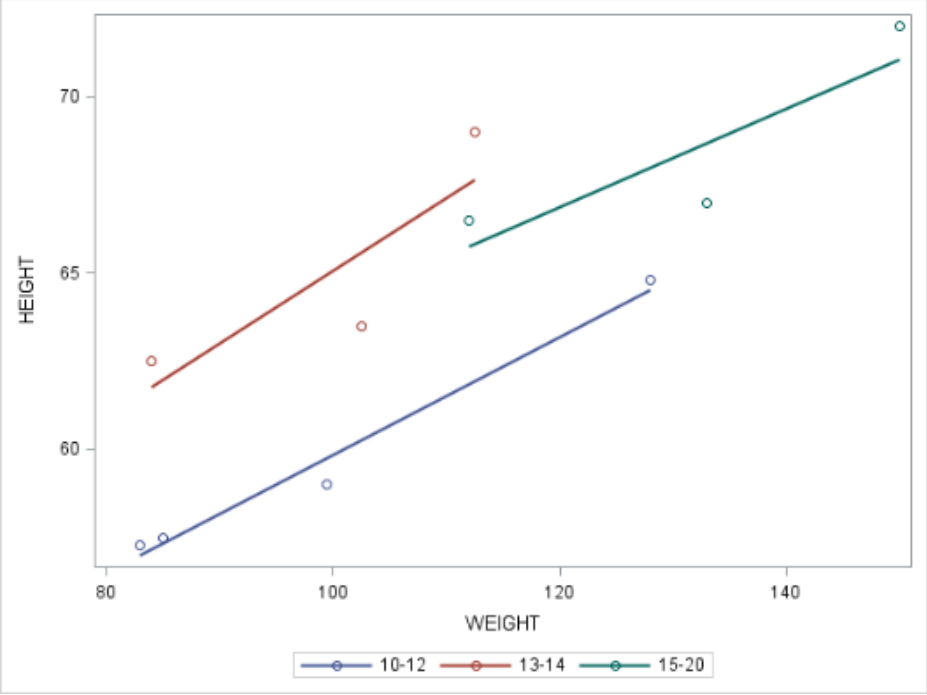

Figure 10: Female, complete legend

all the values were present in the legend.

CONCLUSION

Although adding artificial values, especially with non-missing data, is sometimes very helpful, it should be done with caution and rather be considered as a last resort. Otherwise, it can cause a mistake which can easily be missed, as figure validation is often done by reviewing the output.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Yuliia Bahatska
PRA Health Sciences
bahatskayuliia@prahs.com