WEBVTT

1 00:00:00.610 --> 00:00:02.074 Welcome to video 2.1.

2 00:00:02.650 --> 00:00:06.579 The first of our lab 2 videos in our SAS lab video series.

3 00:00:07.150 --> 00:00:10.059 My name is Maria Ciarleglio, and I'm from the Department of

4 00:00:10.060 --> 00:00:12.880 Biostatistics at the Yale School of Public Health.

5 00:00:13.780 --> 00:00:16.870 In this lab we'll cover how to work with variables in SAS.

6 00:00:17.050 --> 00:00:21.129 Specifically, how to create new variables from existing variables

7 00:00:21.250 --> 00:00:22.250 in a dataset.

8 00:00:25.660 --> 00:00:29.519 The topics covered in this lab will be presented over three videos.

9 00:00:30.130 --> 00:00:32.727 Video 2.1 will begin by reviewing proc import to read

10 00:00:33.850 --> 00:00:35.439 a data file into SAS.

11 00:00:36.010 --> 00:00:38.853 We learned how to use this technique in video 1.2.

12 00:00:39.700 --> 00:00:42.909 Then we will begin looking at a number of different ways of creating

13 00:00:42.910 --> 00:00:43.910 variables.

14 00:00:44.500 --> 00:00:48.729 We will cover arithmetic operations, such as addition, subtraction,

15 00:00:48.820 --> 00:00:50.929 multiplication, division, and exponentiation.

16 00:00:52.570 --> 00:00:56.090 Next, we'll look at some commonly used mathematical functions that

17 00:00:56.110 --> 00:00:57.670 can transform a variable.

18 00:00:57.790 --> 00:01:01.073 For example, the exponential function, square root, and logarithmic

19 00:01:01.300 --> 00:01:02.300 functions.

20 00:01:03.170 --> 00:01:06.355 We'll also discuss some arithmetic and statistical functions that

21 00:01:06.530 --> 00:01:10.040 allow us to sum and find the mean standard deviation,

22 00:01:10.160 --> 00:01:12.806 minimum or maximum value of several variables for each

23 00:01:13.730 --> 00:01:14.730 subject.

24 00:01:15.230 --> 00:01:16.646 At the end of video 2.1.

25 00:01:17.000 --> 00:01:19.640 We will discuss how to work with date variables.

26 00:01:20.000 --> 00:01:23.719 We will add a date variable to a SAS data set, calculate the length

27 00:01:23.720 --> 00:01:26.758 of time between two date variables, and discuss different ways

28 00:01:27.350 --> 00:01:30.470 of displaying or formatting date variables.

29 00:01:31.070 --> 00:01:34.909 In video 2.2, we'll continue to learn about creating variables

30 00:01:34.970 --> 00:01:38.419 using comparison operators such as less than,

31 00:01:38.510 --> 00:01:40.489 greater than, or equal to.

32 00:01:40.760 --> 00:01:43.819 These are commonly used in if-then statements.

33 00:01:43.850 --> 00:01:47.329 For example, if a certain condition is true, then a

34 00:01:47.330 --> 00:01:50.060 new variable will take on a certain value.

35 00:01:50.630 --> 00:01:54.079 And lastly, we'll discuss logical operators such as

36 00:01:54.140 --> 00:01:56.150 and, or, and not.

37 00:01:56.960 --> 00:01:59.557 Video 2.3 will show you how to save this new analysis

38 00:02:00.560 --> 00:02:04.430 dataset that you've created containing all of your new variables.

39 00:02:06.190 --> 00:02:09.369 We will again look at data from the Framingham Heart Study.

40 00:02:09.910 --> 00:02:13.749 The data file that we will use in this video is an Excel file called

41 00:02:13.810 --> 00:02:17.220 Framingham subset 2 which is available in the lab 2

42 00:02:17.410 --> 00:02:19.930 Resources folder on the course web page.

43 00:02:20.290 --> 00:02:23.083 This dataset is similar to Framingham subset that we used

44 00:02:23.800 --> 00:02:25.319 in videos 1.2 and 1.3, but some

45 00:02:27.280 --> 00:02:30.465 of the variables have been modified and I've also added a few new

46 00:02:30.850 --> 00:02:31.850 variables.

47 00:02:32.860 --> 00:02:36.520 There are 30 variables in the data set shown in columns and

48 00:02:38.320 --> 00:02:41.679 4434 individual observations shown as rows.

49 00:02:41.830 --> 00:02:44.949 Notice that the number of rows in our Excel spreadsheet is

50 00:02:46.720 --> 00:02:50.289 4435. This is because the first row of the spreadsheet contains

51 00:02:50.290 --> 00:02:54.039 the variable names, and the individual observations begin

52 00:02:54.040 --> 00:02:56.588 on row 2. This version of Framingham subset includes

53 00:02:57.610 --> 00:03:00.749 a variable I created describing participant race.

54 00:03:01.030 --> 00:03:04.120 And for those who died, I have generated a date of death

55 00:03:04.870 --> 00:03:08.289 in order to give us an opportunity to work with missing data, the 10th

56 00:03:08.290 --> 00:03:10.691 subject shown in row 11 has been modified to give

57 00:03:11.770 --> 00:03:13.480 him some missing data values.

58 00:03:13.630 --> 00:03:16.962 This will allow you to see how missing values in the data affect the

59 00:03:17.080 --> 00:03:18.699 creation of new variables.

60 00:03:19.390 --> 00:03:22.575 Additionally instead of only having information on baseline total

61 00:03:22.900 --> 00:03:26.529 cholesterol, we now have total cholesterol information collected

62 00:03:26.530 --> 00:03:29.200 during three examination periods.

63 00:03:29.500 --> 00:03:33.159 In this study, the three examination periods were approximately six

64 00:03:33.160 --> 00:03:34.160 years apart.

65 00:03:34.810 --> 00:03:36.340 Touching again on missing data.

66 00:03:36.430 --> 00:03:39.639 Notice that when an individual does not have cholesterol level

67 00:03:39.670 --> 00:03:42.512 measured at one of the follow up examinations, the cell is

68 00:03:43.150 --> 00:03:46.709 left blank. When SAS imports these data, SAS

69 00:03:46.720 --> 00:03:49.689 knows that a blank cell indicates a missing value.

70 00:03:49.870 --> 00:03:53.055 So if you're collecting data in an Excel spreadsheet simply leave

71 00:03:53.500 --> 00:03:56.439 the cell blank if a subject has a missing value.

72 00:03:57.090 --> 00:04:00.789 We will use the import procedure to read this Excel file

73 00:04:00.790 --> 00:04:03.879 into SAS which will create a SAS dataset.

74 00:04:04.180 --> 00:04:07.809 The SAS dataset will be given the name Framingham in SAS.

75 00:04:07.900 --> 00:04:11.036 The imported data set will have the same number of variables and

76 00:04:11.290 --> 00:04:14.560 observations as the original Excel spreadsheet.

77 00:04:15.460 --> 00:04:18.547 The first step to working with data and adding new variables to

78 00:04:19.180 --> 00:04:21.819 a data set is to get your data into SAS.

79 00:04:21.970 --> 00:04:25.600 We will use the import procedure to reading our external data file.

80 00:04:25.990 --> 00:04:28.631 This method was discussed in video 1.2.

81 00:04:29.620 --> 00:04:32.266 Here is what typical SAS code for this procedure looks

82 00:04:33.160 --> 00:04:36.639 like. Remember our trick of using the import wizard

83 00:04:36.940 --> 00:04:40.730 and saving the code generated by the wizard for use later.

84 00:04:41.620 --> 00:04:44.380 The generated code is what I have pasted here.

85 00:04:45.070 --> 00:04:48.639 The first line begins with proc import which is the SAS

86 00:04:48.640 --> 00:04:51.471 procedure name. Out = work.Framingham

87 00:04:52.780 --> 00:04:55.622 is where we can specify the library where the imported SAS

88 00:04:56.410 --> 00:04:59.790 data set will be saved, and we can also name the data set.

89 00:05:00.190 --> 00:05:03.819 Here we're saving this data set to the temporary work library

90 00:05:03.850 --> 00:05:05.730 that we discussed in video 1.3.

91 00:05:06.130 --> 00:05:09.364 Remember that the contents of the work library will disappear once

92 00:05:09.940 --> 00:05:12.635 we exit SAS. This means that if we exit our current SAS

93 00:05:13.510 --> 00:05:17.169 session, any data sets that we created in the work library

94 00:05:17.350 --> 00:05:21.220 will need to be re-imported, and any new variables that we created

95 00:05:21.340 --> 00:05:25.029 will need to be recreated once we launch SAS if we want

96 00:05:25.030 --> 00:05:27.129 to continue analysing these data.

97 00:05:27.730 --> 00:05:30.523 We can also create a permanent library using the lib name

98 00:05:31.420 --> 00:05:34.066 statement to save our imported data as a permanent SAS

99 00:05:34.930 --> 00:05:38.559 data set. In video 2.3, I will give you a few

100 00:05:38.560 --> 00:05:41.451 options for saving your modified data set containing all of

101 00:05:42.130 --> 00:05:45.769 your new variables, so that you can directly use that analysis data

102 00:05:45.770 --> 00:05:49.359 set later without having to repeat your data management

103 00:05:49.360 --> 00:05:50.360 steps.

104 00:05:51.010 --> 00:05:54.489 Note that if we had omitted the library name here and just

105 00:05:54.490 --> 00:05:58.089 written out equals Framingham, the Framingham dataset would

106 00:05:58.090 --> 00:06:01.151 have by default been saved to the work library anyway.

107 00:06:02.910 --> 00:06:06.149 The name we are giving the data set is the name Framingham.

108 00:06:06.210 --> 00:06:09.101 So when we refer to this data set in our code we can simply

109 00:06:09.720 --> 00:06:12.219 call it Framingham and SAS will know to look in the

110 00:06:13.290 --> 00:06:16.800 work library if no other library is specified.

111 00:06:17.520 --> 00:06:20.754 On the second line we see the word data file followed by the equal

112 00:06:21.220 --> 00:06:23.759 sign and the physical location of the file.

113 00:06:23.880 --> 00:06:27.480 In this case, the excel file that we're going to import into SAS.

114 00:06:27.600 --> 00:06:31.139 Notice that we put the path of our excel file in double quotes.

115 00:06:31.320 --> 00:06:34.950 You can also use single quotes but just be sure to be consistent

116 00:06:35.070 --> 00:06:36.070 on both ends.

117 00:06:36.870 --> 00:06:39.381 On the 3rd line of code, we have the acronym DBMS

118 00:06:40.380 --> 00:06:42.329 for database management systems.

119 00:06:42.480 --> 00:06:45.518 We need this line of code to tell SAS what type of file we are

120 00:06:46.050 --> 00:06:49.769 importing. Here we're importing an Excel file in this example.

121 00:06:50.040 --> 00:06:53.579 The word replace is necessary to overwrite

122 00:06:53.670 --> 00:06:57.119 any other existing files in SAS by the same name.

123 00:06:57.540 --> 00:07:00.480 Here is where we also put our first semicolon.

124 00:07:01.200 --> 00:07:04.042 The fourth line of code tells SAS which Excel worksheet to

125 00:07:04.740 --> 00:07:08.160 read in. Here we're specifying the name that appears

126 00:07:08.280 --> 00:07:11.339 on the tab of the specific worksheet of interest.

127 00:07:11.790 --> 00:07:15.479 In our excel file Framingham subset 2, our worksheet

128 00:07:15.540 --> 00:07:18.480 is named Framingham. So that is what we specify in the range

129 00:07:19.230 --> 00:07:22.470 statement. We also end this line with a semicolon.

130 00:07:23.070 --> 00:07:25.422 On the fifth line, get names equals yes tell SAS

131 00:07:26.850 --> 00:07:29.937 that there is a header row in the Excel file which contains the

132 00:07:30.060 --> 00:07:33.749 variable names. Line 6 through 9 specify how

133 00:07:33.780 --> 00:07:37.139 text, date, and time variables are imported.

134 00:07:37.770 --> 00:07:41.250 Finally we end the import procedure with the word run,

135 00:07:41.370 --> 00:07:42.689 followed by a semicolon.

136 00:07:43.800 --> 00:07:47.369 Now let's open SAS and run this procedure to

137 00:07:47.400 --> 00:07:49.179 import Framingham subset 2.

138 00:07:49.680 --> 00:07:52.865 I'm going to open SAS by directly double clicking on the existing

139 00:07:53.430 --> 00:07:57.029 program file for this lab called SAS code Video

140 00:07:57.030 --> 00:08:00.989 2. This program file is available in the lab 2 Resources

141 00:08:00.990 --> 00:08:04.679 folder on the course web page so that you can follow along.

142 00:08:06.190 --> 00:08:10.270 In the program editor, I like to begin my code with some comments.

143 00:08:10.390 --> 00:08:13.036 I've also pasted the proc import code generated by the

144 00:08:13.990 --> 00:08:17.769 import wizard and inserted a comment about the procedure

145 00:08:17.770 --> 00:08:21.151 to document what I'm doing and to help me remember my thought process

146 00:08:21.281 --> 00:08:23.794 if I return to this code later.

147 00:08:24.100 --> 00:08:26.844 Most of the comments in this file are to help you better

148 00:08:27.610 --> 00:08:30.790 understand and locate the different pieces of code.

149 00:08:31.000 --> 00:08:33.820 But the use of comments is completely up to you.

150 00:08:33.909 --> 00:08:37.649 You can make them as detailed as you want or omit them altogether.

151 00:08:37.990 --> 00:08:41.077 You will likely have to modify the data file statement to point

152 00:08:41.470 --> 00:08:44.749 to the location where you have Framingham subset 2

153 00:08:45.040 --> 00:08:46.750 saved on your computer.

154 00:08:46.960 --> 00:08:50.409 I have the Framingham subset 2 excel files saved

155 00:08:50.470 --> 00:08:54.039 in the SAS video 2 folder on my USP flash drive

156 00:08:54.100 --> 00:08:57.250 which is mapped as drive G on my computer.

157 00:08:57.790 --> 00:09:01.569 I'm going to highlight and submit or run the import procedure.

158 00:09:01.750 --> 00:09:05.082 Remember you can run your highlighted code by either clicking on the

159 00:09:05.230 --> 00:09:08.709 running man button in the menu bar or by using the keyboard

160 00:09:08.710 --> 00:09:10.660 shortcuts F3 or F8.

161 00:09:11.680 --> 00:09:13.070 Let's check our log window.

162 00:09:13.270 --> 00:09:17.139 The lines in black are the lines of code we highlighted and submitted

163 00:09:17.230 --> 00:09:20.219 and the lines in blue tell us that our SAS data set worked up

164 00:09:20.680 --> 00:09:22.840 Framingham was successfully created.

165 00:09:23.110 --> 00:09:26.679 The log also reports the number of observations and the number

166 00:09:26.680 --> 00:09:28.720 of variables in the imported dataset.

167 00:09:28.960 --> 00:09:32.679 This should match the number in the original imported excel file.

168 00:09:33.610 --> 00:09:36.870 Next let's check the contents of the Framingham dataset.

169 00:09:37.060 --> 00:09:40.990 We will use the contents procedure to give us a list of the variables

170 00:09:41.050 --> 00:09:42.169 in our dataset.

171 00:09:42.700 --> 00:09:46.179 We'll also be able to see if the variable was imported as

172 00:09:46.180 --> 00:09:48.490 a numeric or character variable.

173 00:09:48.700 --> 00:09:52.419 I would like the variables to be listed in the order they are included

174 00:09:52.420 --> 00:09:55.409 in the file rather than in alphabetical order, so I specified

175 00:09:56.230 --> 00:09:58.778 the order equals var num option in the first line of

176 00:09:59.740 --> 00:10:03.309 PROC CONTENTS. I find this option useful because there tends to

177 00:10:03.310 --> 00:10:06.380 be a reason for the ordering of variables in a dataset.

178 00:10:06.610 --> 00:10:10.179 Although if I'm looking to see if the dataset contains a specific

179 00:10:10.180 --> 00:10:13.960 variable, then looking at the variable list in alphabetical order

180 00:10:14.020 --> 00:10:17.079 would be more helpful. Again highlight and run the code.

181 00:10:19.350 --> 00:10:22.059 Checking the log, I don't see any errors.

182 00:10:22.110 --> 00:10:23.880 But I do see a new window.

183 00:10:23.970 --> 00:10:25.499 The results viewer window.

184 00:10:25.800 --> 00:10:28.529 This is the window that contains our output.

185 00:10:28.770 --> 00:10:32.249 Here we see the data file Framingham is saved in the work

186 00:10:32.250 --> 00:10:35.328 library because under dataset name we see work.Framingham.

187 00:10:36.750 --> 00:10:39.119 This dataset has 30 variables and

188 00:10:40.650 --> 00:10:41.909 4434 observations.

189 00:10:42.510 --> 00:10:45.890 I also see the intake and date of death variables were

190 00:10:46.050 --> 00:10:48.500 imported correctly as a date, and the sex and race

191 00:10:49.740 --> 00:10:53.549 variables were imported correctly as character variables.

192 00:10:54.120 --> 00:10:57.354 It's always a good idea to check that your data are being imported

193 00:10:57.690 --> 00:10:58.690 correctly.

194 00:10:59.840 --> 00:11:02.878 We will take the Framingham data set that we just imported and

195 00:11:03.320 --> 00:11:05.240 create several new variables.

196 00:11:05.360 --> 00:11:08.202 The new variables will appear as new columns at the end of

197 00:11:08.870 --> 00:11:11.026 the data set. This process creates a new SAS

198 00:11:12.350 --> 00:11:15.889 data file that contains both the original variables and

199 00:11:15.890 --> 00:11:16.999 the new variables.

200 00:11:17.600 --> 00:11:21.140 I call this new data set my analysis data set because

201 00:11:21.170 --> 00:11:24.208 oftentimes we are performing this data management and creating

202 00:11:24.740 --> 00:11:28.460 these new variables for the purpose of conducting data analysis

203 00:11:28.520 --> 00:11:32.149 later on. The data set that then contains the variables we

204 00:11:32.150 --> 00:11:35.660 need for a future analysis is my analysis data set.

205 00:11:36.200 --> 00:11:38.999 We have two options for saving this new data set.

206 00:11:39.260 --> 00:11:42.829 We can overwrite the existing Framingham data set by giving

207 00:11:42.830 --> 00:11:46.640 the new data set. the name Framingham, the same as the original.

208 00:11:46.880 --> 00:11:49.624 Alternatively, we can create a completely new dataset by

209 00:11:50.540 --> 00:11:52.160 assigning it a different name.

210 00:11:52.400 --> 00:11:55.879 In our example we will name our new dataset Framingham

211 00:11:55.880 --> 00:11:58.379 new. This approach allows us to keep the Framingham

212 00:11:59.840 --> 00:12:02.299 SAS data file in its original form.

213 00:12:02.810 --> 00:12:06.499 Our new dataset is a copy of Framingham plus any newly

214 00:12:06.500 --> 00:12:08.869 created or modified variables.

215 00:12:09.080 --> 00:12:12.889 Be careful when using option one because you will overwrite

216 00:12:12.950 --> 00:12:15.950 the version of Framingham that you originally imported.

217 00:12:16.130 --> 00:12:18.889 I don't mean that you will overwrite the excel file; you will

218 00:12:18.890 --> 00:12:21.781 overwrite the SAS data file that was created as a result of

219 00:12:22.460 --> 00:12:23.720 the import procedure.

220 00:12:24.440 --> 00:12:27.478 Once you overwrite a data set you cannot simply go back to the

221 00:12:27.650 --> 00:12:31.370 previous version. For example, if we overwrite Framingham

222 00:12:31.460 --> 00:12:35.570 and add 10 new variables, the dataset will now contain 40 variables.

223 00:12:35.720 --> 00:12:39.003 This means that we've lost the original version of the dataset with

224 00:12:39.230 --> 00:12:42.799 30 variables. To get the original back, we would have to rerun

225 00:12:42.800 --> 00:12:44.090 the import procedure.

226 00:12:44.360 --> 00:12:47.929 My preference is to always leave the original data file as

227 00:12:47.930 --> 00:12:50.723 is and create a second analysis dataset that contains any

228 00:12:51.470 --> 00:12:54.500 modifications changes or new variables.

229 00:12:54.830 --> 00:12:58.309 This way if I make a mistake in creating a new variable or if I

230 00:12:58.310 --> 00:13:02.330 accidentally overwrite an existing variable in the original dataset

231 00:13:02.390 --> 00:13:05.869 I can correct my code and use the original dataset to try

232 00:13:05.870 --> 00:13:06.870 again.

233 00:13:07.380 --> 00:13:10.590 We will create new variables using a data step.

234 00:13:10.680 --> 00:13:14.159 We used a data step in video 1.2 to read data

235 00:13:14.210 --> 00:13:16.679 into SAS using manual data entry.

236 00:13:17.170 --> 00:13:20.849 On line one, we begin the DATA step with the word data

237 00:13:20.910 --> 00:13:24.193 followed by the name of the data set that we want to create in this

238 00:13:24.450 --> 00:13:26.459 data step. Because I prefer option 2, I'm

239 00:13:27.960 --> 00:13:31.626 giving the modified data set a new name, Framingham new.

240 00:13:32.160 --> 00:13:35.759 Because I'm not specifying a library, this new data set will

241 00:13:35.760 --> 00:13:37.799 be saved in the work library.

242 00:13:38.490 --> 00:13:42.389 You could explicitly specify the work library as the location

243 00:13:42.390 --> 00:13:45.750 of Framingham new as we see here but it's not necessary.

244 00:13:45.990 --> 00:13:49.559 When you don't specify a library., SAS will default to

245 00:13:49.590 --> 00:13:52.040 the work library. We end the data statement with a

246 00:13:53.220 --> 00:13:54.220 semicolon.

247 00:13:55.030 --> 00:13:57.549 On the second line I begin the SET statement.

248 00:13:57.640 --> 00:14:01.090 This is where I identify the dataset that will be read

249 00:14:01.210 --> 00:14:02.339 into the data step.

250 00:14:02.830 --> 00:14:06.669 We want to read the original Framingham data set into

251 00:14:06.700 --> 00:14:10.240 the DATA step, create a copy called Framingham new,

252 00:14:10.330 --> 00:14:13.929 and make any of the modifications to the dataset specified

253 00:14:13.930 --> 00:14:16.029 within the body of the DATA step.

254 00:14:16.360 --> 00:14:19.545 This is where we will define new variables, usually by performing

255 00:14:20.470 --> 00:14:22.900 operations on existing variables.

256 00:14:23.140 --> 00:14:26.178 Again, we are reading the Framingham data set in from the work

257 00:14:26.860 --> 00:14:29.898 library. You could have also put the SET statement on the same

258 00:14:30.340 --> 00:14:32.079 line as the data statement.

259 00:14:32.520 --> 00:14:35.166 In SAS, it's not the line breaks that indicate the end

260 00:14:36.070 --> 00:14:38.320 of each statement, it's the semicolons.

261 00:14:38.710 --> 00:14:42.429 End the DATA step with the word run followed by a semicolon.

262 00:14:43.090 --> 00:14:45.785 Now we can discuss SAS operators and functions that are

263 00:14:46.570 --> 00:14:50.590 used in SAS programming statements to create new variables.

264 00:14:51.430 --> 00:14:54.309 We will begin by discussing arithmetic operators.

265 00:14:54.760 --> 00:14:58.359 The first arithmetic operation we will discuss is addition.

266 00:14:58.780 --> 00:15:02.559 We will use addition to add two or more variables together

267 00:15:02.560 --> 00:15:06.039 to make a new variable, or to add a constant

268 00:15:06.070 --> 00:15:07.990 to one or more variables.

269 00:15:08.290 --> 00:15:11.320 We type the plus symbol between the terms we are summing.

270 00:15:12.100 --> 00:15:15.580 We have five binary variables in our Framingham dataset

271 00:15:15.640 --> 00:15:18.874 that report the presence of prevalent disease at the time of entry

272 00:15:19.360 --> 00:15:22.104 into the study: prevalent coronary heart disease, angina

273 00:15:22.960 --> 00:15:25.557 pectoris, myocardial infarction stroke, and prevalent

274 00:15:26.710 --> 00:15:27.710 hypertension.

275 00:15:28.030 --> 00:15:32.049 The variable equals one, if the individual has the prevalent disease,

276 00:15:32.200 --> 00:15:35.350 and zero if the individual is free from the disease.

277 00:15:35.560 --> 00:15:38.794 We see the count and percentage of individuals with each prevalent

278 00:15:39.040 --> 00:15:41.931 condition reported in these frequency tables with prevalent

279 00:15:42.520 --> 00:15:45.607 hypertension being the most common prevalent condition in these

280 00:15:46.120 --> 00:15:50.080 individuals. We want to create a new variable in the dataset

281 00:15:50.140 --> 00:15:53.729 that counts the number of prevalent conditions that a participant

282 00:15:53.730 --> 00:15:57.309 has. Because the variables indicating presence or

283 00:15:57.310 --> 00:16:00.999 absence of disease are binary, if we sum these indicator

284 00:16:01.000 --> 00:16:04.185 variables, we will be counting the number of prevalent conditions

285 00:16:04.630 --> 00:16:05.830 for each individual.

286 00:16:06.160 --> 00:16:09.642 Here we are naming this new account variable prev cond,

287 00:16:10.090 --> 00:16:12.639 and this variable will be in Framingham new.

288 00:16:13.090 --> 00:16:15.834 Also notice that when it comes to variable names, SAS is

289 00:16:16.600 --> 00:16:17.740 not case sensitive.

290 00:16:18.130 --> 00:16:21.670 Although the prevalent condition variables are all capitalized

291 00:16:21.730 --> 00:16:25.239 in the dataset Framingham, I do not need to use all caps

292 00:16:25.300 --> 00:16:27.070 when referencing the variables.

293 00:16:27.760 --> 00:16:31.059 Just so it's clear I want you to see that these operations will be

294 00:16:31.060 --> 00:16:32.919 applied to each subject.

295 00:16:33.130 --> 00:16:36.140 The subjects are represented in the rows of the data set.

296 00:16:36.180 --> 00:16:37.779 Seven subjects are shown here.

297 00:16:38.200 --> 00:16:41.889 The last subject happens to be the 10th subject in the original

298 00:16:41.890 --> 00:16:44.879 dataset. This subject has several missing data values so that

299 00:16:45.460 --> 00:16:48.400 you can see how variable creation behaves in the presence of

300 00:16:48.970 --> 00:16:50.379 missing input values.

301 00:16:50.950 --> 00:16:54.249 When you follow along in print some of the new variables we will

302 00:16:54.250 --> 00:16:57.970 create using the code in the SAS program file, the obs

303 00:16:58.000 --> 00:17:01.839 equals ten option in PROC PRINT will print the first

304 00:17:01.870 --> 00:17:03.789 10 subjects in the dataset.

305 00:17:04.329 --> 00:17:08.019 The new variable we're creating sums the values of

306 00:17:08.079 --> 00:17:12.159 prevalent CHD, AP, MY, stroke, and hypertension

307 00:17:12.190 --> 00:17:15.228 for each subject, essentially counting the number of prevalent

308 00:17:15.579 --> 00:17:17.559 conditions each subject has.

309 00:17:18.020 --> 00:17:21.549 Prev Cond equals zero for the first subject shown because

310 00:17:21.550 --> 00:17:25.209 he has no prevalent conditions. One for the second subject,

311 00:17:25.670 --> 00:17:27.875 three for the third, two for the fourth, zero

312 00:17:29.190 --> 00:17:31.510 for the fifth, and one for the six.

313 00:17:32.110 --> 00:17:35.589 When it comes to subject seven we see that his value of

314 00:17:35.860 --> 00:17:37.089 Prev Cond is missing.

315 00:17:37.570 --> 00:17:40.510 This is because one of the variables involved in calculating

316 00:17:41.110 --> 00:17:42.429 the sum is missing.

317 00:17:43.390 --> 00:17:44.860 Let's move on to subtraction.

318 00:17:45.280 --> 00:17:49.329 Here we create a new variable called pulse pressure in Framingham

319 00:17:49.330 --> 00:17:52.466 new which is the difference between systolic and diastolic blood

320 00:17:52.930 --> 00:17:56.559 pressure. The next operation is multiplication which

321 00:17:56.560 --> 00:17:58.449 uses the asterisk symbol.

322 00:17:58.660 --> 00:18:02.289 We are creating a variable age months which is simply

323 00:18:02.290 --> 00:18:05.319 age expressed in months and not in years.

324 00:18:05.800 --> 00:18:08.740 We do this by multiplying the variable age in the Framingham

325 00:18:09.460 --> 00:18:12.559 dataset which is aging years by the number 12.

326 00:18:13.270 --> 00:18:16.869 The constant twelve will be multiplied by each individual's

327 00:18:16.900 --> 00:18:18.970 age to give us our new variable.

328 00:18:19.120 --> 00:18:22.780 Again age in months is missing when age itself is missing.

329 00:18:23.840 --> 00:18:27.619 Looking next at Division we can calculate mean arterial pressure

330 00:18:27.680 --> 00:18:31.219 the average pressure in a patient's arteries during one cardiac

331 00:18:31.220 --> 00:18:34.111 cycle. By adding systolic blood pressure to twice diastolic

332 00:18:35.180 --> 00:18:38.119 blood pressure and dividing that quantity by three.

333 00:18:38.720 --> 00:18:41.929 This example illustrates how you can use multiple arithmetic

334 00:18:41.930 --> 00:18:44.569 operations together when defining a variable.

335 00:18:44.930 --> 00:18:48.289 Here we are using addition, multiplication, and division.

336 00:18:49.160 --> 00:18:51.440 Also notice the use of parentheses.

337 00:18:51.760 --> 00:18:55.429 Here, we want to divide the quantity systolic plus

338 00:18:55.430 --> 00:18:57.380 twice diastolic by three.

339 00:18:57.980 --> 00:19:01.067 If we had admitted the parentheses, SAS would have only divided

340 00:19:01.730 --> 00:19:04.670 the quantity, two times diastolic by three.

341 00:19:05.740 --> 00:19:09.549 A second formula for map involves first calculating

342 00:19:09.550 --> 00:19:13.239 pulse pressure, which we have, and dividing that by three, then

343 00:19:13.240 --> 00:19:14.950 adding diastolic blood pressure.

344 00:19:15.250 --> 00:19:18.386 Even though we have not yet run the DATA step to formally create

345 00:19:18.880 --> 00:19:22.359 pulse pressure, you can still reference variables that you

346 00:19:22.360 --> 00:19:25.650 defined on an earlier line in that same data step.

347 00:19:26.680 --> 00:19:30.730 Finally, we apply exponentiation using two asterisks.

348 00:19:31.090 --> 00:19:34.344 We calculate age squared or age to the power 2.

349 00:19:35.500 --> 00:19:39.009 As a reminder if one of the arguments for an arithmetic

350 00:19:39.010 --> 00:19:41.680 operator is missing, the result is missing.

351 00:19:43.570 --> 00:19:47.109 I want to take a moment before we get into the next topic to show

352 00:19:47.110 --> 00:19:49.450 you how to use SAS as a calculator.

353 00:19:49.780 --> 00:19:52.965 SAS can be used to perform calculations such as simple arithmetic

354 00:19:53.650 --> 00:19:57.579 operations that we've seen or more complex calculations

355 00:19:57.580 --> 00:20:01.029 involving different built in functions which we'll discuss shortly.

356 00:20:01.630 --> 00:20:05.260 First let me show you how not to perform calculations

357 00:20:05.320 --> 00:20:06.519 in your SAS program.

358 00:20:06.880 --> 00:20:10.281 Suppose you wanted SAS to compute the sum of 1 in 1,

359 00:20:10.540 --> 00:20:11.540 so 1 plus 1.

360 00:20:12.130 --> 00:20:15.609 Typing 1 plus 1 in your program and running this line of

361 00:20:15.610 --> 00:20:17.589 code will not return an answer.

362 00:20:18.010 --> 00:20:20.410 In fact, you'll receive an error in your log.

363 00:20:21.190 --> 00:20:24.220 Maybe we need to assign the result to a variable.

364 00:20:24.490 --> 00:20:27.479 We want the variable called answer to equal the result of the

365 00:20:27.940 --> 00:20:30.586 calculation. Unfortunately, this also isn't the way to

366 00:20:31.420 --> 00:20:33.940 do it. You will again receive an error in your log.

367 00:20:34.900 --> 00:20:39.040 SAS is different from other object oriented programming languages.

368 00:20:39.130 --> 00:20:42.339 For the most part you can't have variables floating around.

369 00:20:42.430 --> 00:20:44.782 You need to work within data sets which means we

370 00:20:45.940 --> 00:20:49.719 need to perform our calculations within a data step.

371 00:20:50.170 --> 00:20:53.159 We do this even though we're not reading in a data file using

372 00:20:53.740 --> 00:20:57.309 the SET statement. We still need to work within a data

373 00:20:57.310 --> 00:21:00.969 step when using SAS as a calculator .This means

374 00:21:00.970 --> 00:21:03.518 that we need to print the data set that we create in

375 00:21:04.510 --> 00:21:08.170 our data step or navigate to the created data set

376 00:21:08.260 --> 00:21:10.710 in the Explorer tab in order to see the results of

377 00:21:11.800 --> 00:21:12.939 our calculations.

378 00:21:13.890 --> 00:21:17.339 The calculated value which is the output of the function

379 00:21:17.370 --> 00:21:20.790 or the operation, will need to be assigned a variable name,

380 00:21:20.880 --> 00:21:23.967 and that variable which holds the result will be the last value

381 00:21:24.660 --> 00:21:26.119 in your created data set.

382 00:21:26.250 --> 00:21:29.489 Or perhaps it will be the only value in your data set, as we'll see

383 00:21:29.490 --> 00:21:30.690 with this first example.

384 00:21:30.870 --> 00:21:33.712 As an example of how to use SAS as a calculator, let's use

385 00:21:34.650 --> 00:21:38.309 some of the arithmetic operations that we're already familiar with.

386 00:21:38.910 --> 00:21:41.801 I'm naming the data set that will contain the results of my

387 00:21:41.970 --> 00:21:43.186 calculations, calc.

388 00:21:44.160 --> 00:21:47.729 I want to compute an individual's body mass index, or

389 00:21:47.730 --> 00:21:51.329 BMI, using the formula, weight in kilograms divided

390 00:21:51.330 --> 00:21:53.130 by height in meters squared.

391 00:21:53.430 --> 00:21:57.210 The BMI I want to calculate is for an individual with a weight

392 00:21:57.240 --> 00:22:00.170 of 70 kilograms and a height equal to 1.6

393 00:22:01.080 --> 00:22:04.619 meters. The result of the calculation will be in a

394 00:22:04.620 --> 00:22:06.420 variable called BMI.

395 00:22:06.990 --> 00:22:10.739 After running the DATA step when I run the print procedure to print

396 00:22:10.740 --> 00:22:13.484 the data set called calc, I see a single row in a single

397 00:22:14.400 --> 00:22:17.400 variable, the variable that we created called BMI.

398 00:22:18.000 --> 00:22:20.254 The no OBS option in PROC PRINT suppresses SAS

399 00:22:21.660 --> 00:22:23.450 is customary observation number.

400 00:22:24.360 --> 00:22:28.259 Another way of performing this calculation is to define a variable

401 00:22:28.260 --> 00:22:30.955 for weight and a variable for height and then use those

402 00:22:31.920 --> 00:22:34.259 variables in the formula for BMI.

403 00:22:34.620 --> 00:22:38.160 We can compare this to the calculation that we performed on line 2

404 00:22:38.520 --> 00:22:41.699 where we plug the raw values directly into the formula.

405 00:22:42.030 --> 00:22:44.279 As you would expect, we get the same result.

406 00:22:44.850 --> 00:22:48.329 Notice here that the data set Calc 2 contains all

407 00:22:48.330 --> 00:22:51.510 of the variables that we created in this second data step.

408 00:22:53.130 --> 00:22:56.880 Next we'll discuss using mathematical functions in SAS.

409 00:22:57.150 --> 00:23:01.130 Functions can be applied to a single variable or to a set of variables

410 00:23:01.170 --> 00:23:03.720 to help us transform or analyze data.

411 00:23:03.870 --> 00:23:07.300 There are many built in functions in SAS, and we'll be looking at some

412 00:23:07.320 --> 00:23:11.160 of the most frequently used mathematical and statistical functions.

413 00:23:11.850 --> 00:23:14.937 Functions are applied in a data step in the same way we carried

414 00:23:15.540 --> 00:23:17.219 out arithmetic operations.

415 00:23:17.550 --> 00:23:20.219 Each function takes one or more arguments.

416 00:23:20.580 --> 00:23:23.910 These arguments can be existing variables in a data set.

417 00:23:24.150 --> 00:23:27.139 In this case you want to use the values of those variables to

418 00:23:27.660 --> 00:23:30.502 evaluate the function. SAS will take the input values, the

419 00:23:31.170 --> 00:23:34.208 arguments for each subject, apply the function, and create the

420 00:23:34.680 --> 00:23:36.539 new variable in Framingham new.

421 00:23:37.140 --> 00:23:40.390 For example, we have BMI in our Framingham dataset.

422 00:23:40.770 --> 00:23:44.279 If I want to compute the natural log of BMI for all subjects

423 00:23:44.280 --> 00:23:47.699 in my dataset, I would create a variable in Framingham new

424 00:23:47.760 --> 00:23:51.359 that applies the log function, which is natural log in SAS, to

425 00:23:51.360 --> 00:23:53.160 the existing variable BMI.

426 00:23:53.550 --> 00:23:56.699 The name I chose for this new variable is log BMI.

427 00:23:57.450 --> 00:24:00.684 Notice that the log function applied to a missing value, as we see

428 00:24:01.020 --> 00:24:04.107 here in the last subject, returns a missing value, as you would

429 00:24:04.200 --> 00:24:07.619 expect. A function that can take multiple arguments

430 00:24:07.680 --> 00:24:09.329 is the max function.

431 00:24:09.700 --> 00:24:13.559 Recall that we have total cholesterol collected over three study

432 00:24:13.560 --> 00:24:17.369 periods. The largest non missing value of total cholesterol

433 00:24:17.370 --> 00:24:20.820 for each subject would be the max of existing variables,

434 00:24:20.970 --> 00:24:23.309 Total cholesterol 1, 2, and 3.

435 00:24:23.730 --> 00:24:26.910 The arguments of the function are separated by commas.

436 00:24:27.360 --> 00:24:30.779 Again, notice that if all of the input values of the function are

437 00:24:30.780 --> 00:24:33.622 missing, as we see in the last subject, and the maximum of

438 00:24:34.290 --> 00:24:37.619 a set of missing values is also a missing value.

439 00:24:38.100 --> 00:24:41.040 In this data step we are producing new variables as a result

440 00:24:41.640 --> 00:24:42.900 of the calculations.

441 00:24:43.230 --> 00:24:46.799 The new variables will be appended as columns to the dataset

442 00:24:46.830 --> 00:24:48.240 created in the DATA step.

443 00:24:49.170 --> 00:24:52.619 Log BMI and Max cholesterol are the final columns

444 00:24:52.680 --> 00:24:53.759 of Framingham new.

445 00:24:55.660 --> 00:24:59.309 Here is a list of some commonly used mathematical functions that can

446 00:24:59.310 --> 00:25:00.869 be applied to a number.

447 00:25:00.990 --> 00:25:02.940 Here X is a real number.

448 00:25:03.420 --> 00:25:06.479 The first function we'll discuss is the exponential function.

449 00:25:06.870 --> 00:25:10.469 The exponential function E to the X calculates the value

450 00:25:10.470 --> 00:25:13.459 of E raised to the power of your argument where E is the base

451 00:25:14.130 --> 00:25:15.166 of the natural logarithm 2.718.

452 00:25:17.190 --> 00:25:20.699 Let's.switch to SAS to apply the exponential function to

453 00:25:20.730 --> 00:25:21.730 a variable.

454 00:25:22.920 --> 00:25:26.460 Here I used a DATA step to create a variable X

455 00:25:26.550 --> 00:25:28.829 in a data set that I call Example 1.

456 00:25:29.250 --> 00:25:32.670 I have nine observations here but one observation is missing.

457 00:25:33.020 --> 00:25:36.690 I'm including the missing X value to show you how the functions deal

458 00:25:36.720 --> 00:25:39.513 with missing values. Normally in a data step I enter each

459 00:25:40.230 --> 00:25:43.709 observation on a separate line after the word data lines.

460 00:25:43.800 --> 00:25:47.670 SAS will run a complete iteration of the DATA step to construct

461 00:25:47.700 --> 00:25:50.460 one observation using one raw line of data.

462 00:25:50.790 --> 00:25:54.479 The one line of data can contain one or more variables.

463 00:25:54.630 --> 00:25:57.899 The data step then repeats this process again and again until there

464 00:25:57.900 --> 00:26:00.269 are no raw data lines left to read.

465 00:26:00.570 --> 00:26:04.410 But you see here that I list all nine values of X on the same line.

466 00:26:04.690 --> 00:26:08.094 I am able to do this because I include the double trailing at sign

467 00:26:08.520 --> 00:26:09.900 in the input statement.

468 00:26:10.380 --> 00:26:13.859 Basically this allows me to create several observations from

469 00:26:13.860 --> 00:26:14.880 one line of data.

470 00:26:15.390 --> 00:26:18.749 I could also break the data across more than one line and SAS will

471 00:26:18.750 --> 00:26:21.935 continue to read the next observation in the order the values are

472 00:26:22.140 --> 00:26:25.980 entered. Let's run the data step and check the log for errors.

473 00:26:29.680 --> 00:26:33.279 SAS tells us that we have nine observations and one variable in

474 00:26:33.280 --> 00:26:34.684 our data set Example 1.

475 00:26:35.860 --> 00:26:39.549 We can either print the data set or navigate to it in the Explorer

476 00:26:39.550 --> 00:26:41.380 tab to view it as a table.

477 00:26:42.040 --> 00:26:45.449 Since it's a small data set, let's print it to our results viewer

478 00:26:45.460 --> 00:26:46.460 window.

479 00:26:47.270 --> 00:26:50.779 I see that all nine of my observations have been read in and are

480 00:26:50.780 --> 00:26:53.180 contained in the data setmExample 1.

481 00:26:53.690 --> 00:26:57.140 Notice that we see SAS's automatic observation numbering

482 00:26:57.170 --> 00:27:00.680 because I did not use the no OBS option in PROC PRINT.

483 00:27:00.920 --> 00:27:02.150 Let's go back to our code.

484 00:27:03.230 --> 00:27:05.745 We want to transform X by exponentiating.

485 00:27:06.470 --> 00:27:10.069 I can do this by writing a second data step that reads in the data

486 00:27:10.070 --> 00:27:13.609 set Example 1, and creates a new variable which is the

487 00:27:13.610 --> 00:27:17.089 exponentiated x value, and this would be a fine way to do it.

488 00:27:17.360 --> 00:27:20.959 I'm preserving the original data set Example 1 and creating

489 00:27:20.990 --> 00:27:24.559 a copy called example 1A which will contain e

490 00:27:24.560 --> 00:27:25.560 to the X.

491 00:27:26.180 --> 00:27:27.640 When we print our new data set

492 00:27:27.710 --> 00:27:29.817 Example 1A, We see it contains both X and E

493 00:27:31.250 --> 00:27:34.760 to the X. Notice that exponentiating a missing value

494 00:27:34.880 --> 00:27:36.470 results in a missing value.

495 00:27:36.920 --> 00:27:40.579 SAS also tells us that we try to perform an operation on a missing

496 00:27:40.580 --> 00:27:41.809 value in the log.

497 00:27:42.890 --> 00:27:46.039 I want to show you that you can also create new variables in the

498 00:27:46.040 --> 00:27:48.740 original data step where I'm reading in the data.

499 00:27:48.980 --> 00:27:52.480 I can define any new variables that I also want this dataset

500 00:27:52.490 --> 00:27:56.179 to contain right after the input statement and before the data

501 00:27:56.180 --> 00:27:59.929 line statement. For example I know that this dataset will contain

502 00:27:59.960 --> 00:28:02.655 the variable x because I specified this variable in the

503 00:28:03.440 --> 00:28:04.440 input statement.

504 00:28:05.030 --> 00:28:08.119 I want this data set to also contain e to the X.

505 00:28:08.210 --> 00:28:10.313 So I define a new variable which I name EXPX,

506 00:28:12.020 --> 00:28:15.880 equal to the exponential function applied to the variable x.

507 00:28:16.250 --> 00:28:19.160 This saves us from having to run an additional data step.

508 00:28:20.740 --> 00:28:24.699 Let's print the data set to see if it contains our two variables.

509 00:28:24.910 --> 00:28:25.930 And it does.

510 00:28:27.420 --> 00:28:29.968 The function for natural log or log base e is simply

511 00:28:31.290 --> 00:28:32.290 log in SAS.

512 00:28:32.970 --> 00:28:36.509 You might be used to using LN to refer to natural log

513 00:28:36.540 --> 00:28:38.500 but know that in SAS log is natural log.

514 00:28:39.090 --> 00:28:42.720 The function for log base ten is log ten

515 00:28:43.320 --> 00:28:46.260 both log and log ten must have positive arguments, otherwise

516 00:28:47.190 --> 00:28:50.489 the function will return a missing value and you will see a note about

517 00:28:50.490 --> 00:28:51.900 this in the SAS log.

518 00:28:52.680 --> 00:28:55.349 Next we see absolute value and square root.

519 00:28:55.530 --> 00:28:58.650 Square root of a negative number will return a missing value.

520 00:28:59.190 --> 00:29:02.498 In order to round a number to a certain number of decimal places, we

521 00:29:03.060 --> 00:29:04.499 use the round function.

522 00:29:04.620 --> 00:29:08.219 Specify the argument and then specify to how many decimals

523 00:29:08.220 --> 00:29:09.510 we'd like to round the number.

524 00:29:10.020 --> 00:29:13.949 In this example we would like to round X to the hundredth place.

525 00:29:14.910 --> 00:29:17.850 Int basically cuts off the decimal from a number and returns

526 00:29:18.630 --> 00:29:21.570 the integer only. This function does not round.

527 00:29:22.070 --> 00:29:25.499 Ceiling returns the smallest integer that is greater than

528 00:29:25.560 --> 00:29:27.390 or equal to the argument.

529 00:29:27.780 --> 00:29:30.426 Floor returns the largest integer that is less than or

530 00:29:31.260 --> 00:29:32.490 equal to the argument.

531 00:29:32.970 --> 00:29:36.106 We'll see how these functions behave when we apply them to our x

532 00:29:36.420 --> 00:29:37.829 variable in SAS.

533 00:29:39.110 --> 00:29:42.590 I'm going to repeat the process of creating these new transformed

534 00:29:42.620 --> 00:29:45.903 X variables by applying the remainder of the functions that we just

535 00:29:46.130 --> 00:29:49.579 covered in the original data step where we entered the raw data.

536 00:29:49.940 --> 00:29:52.309 I'm naming this dataset Example 1.

537 00:29:52.370 --> 00:29:55.999 So after I run this data step SAS will overwrite

538 00:29:56.030 --> 00:29:59.569 Example 1 that currently exists in the work library.

539 00:29:59.930 --> 00:30:02.625 To keep our original example 1, we would need to assign

540 00:30:03.440 --> 00:30:05.229 a different name to this dataset.

541 00:30:06.050 --> 00:30:07.050 Let's run this code.

542 00:30:08.820 --> 00:30:12.390 Checking the log we do see several messages telling us

543 00:30:12.420 --> 00:30:15.809 when we are trying to perform a calculation that is not defined.

544 00:30:16.260 --> 00:30:19.799 Each time SAS is asked to perform a calculation using

545 00:30:19.800 --> 00:30:22.451 an invalid argument, it lets you know in the log.

546 00:30:23.040 --> 00:30:26.759 SAS also notifies you when it's asked to perform calculations

547 00:30:26.760 --> 00:30:30.359 using missing values and tells you that this action generated

548 00:30:30.450 --> 00:30:34.469 a missing value. However, these are not program halting errors

549 00:30:34.530 --> 00:30:37.079 and do not stop SAS from running the DATA step.

550 00:30:37.710 --> 00:30:41.549 As we see at the very bottom, Example One was successfully created

551 00:30:41.670 --> 00:30:45.239 and it contains nine observations and 10 variables.

552 00:30:45.690 --> 00:30:49.229 Again, keep an eye on your log after each run because it can help

553 00:30:49.230 --> 00:30:51.299 you identify problems with your data.

554 00:30:53.610 --> 00:30:56.158 Here we see some common mainly statistical functions

555 00:30:57.240 --> 00:31:00.809 that are applied to either a list of raw numbers entered as

556 00:31:00.810 --> 00:31:04.800 arguments or to more than one variable in an existing dataset.

557 00:31:05.280 --> 00:31:08.369 The first function listed here is the sum function.

558 00:31:08.670 --> 00:31:11.819 I'll point out the difference between the SUM function and simply

559 00:31:11.820 --> 00:31:14.670 using the addition arithmetic operation shortly.

560 00:31:15.090 --> 00:31:18.869 Next we have mean, median, min max, number

561 00:31:18.870 --> 00:31:21.850 of non missing values, number of missing values, standard

562 00:31:22.530 --> 00:31:24.330 deviation, and variance.

563 00:31:24.420 --> 00:31:28.410 The arguments are listed inside the parentheses separated by commas.

564 00:31:29.160 --> 00:31:32.443 Note that the result of these functions is based on the non missing

565 00:31:32.670 --> 00:31:34.439 values of the arguments.

566 00:31:34.800 --> 00:31:38.339 Even if a subject has a missing value for one of the arguments

567 00:31:38.370 --> 00:31:41.849 these functions will use the non missing values to perform the

568 00:31:41.850 --> 00:31:42.850 calculation.

569 00:31:43.200 --> 00:31:46.529 This is the difference between the basic arithmetic operations we

570 00:31:46.530 --> 00:31:50.069 performed previously and the sum function, for example.

571 00:31:50.160 --> 00:31:53.002 The SUM function will add all available data whereas if we

572 00:31:53.670 --> 00:31:56.316 use the plus sign to define the sum if any argument of

573 00:31:57.180 --> 00:32:00.710 the arithmetic operation is missing the sum itself will be missing.

574 00:32:02.450 --> 00:32:05.537 Moving to SAS, we will apply these functions to our three total

575 00:32:06.170 --> 00:32:09.859 cholesterol measurements taken over the three periods in this study.

576 00:32:10.010 --> 00:32:13.293 I'm going to read in the original Framingham dataset and apply each

577 00:32:13.520 --> 00:32:17.059 of these functions. The dataset Framingham new will

578 00:32:17.060 --> 00:32:19.519 contain the results of the calculations.

579 00:32:20.240 --> 00:32:24.079 It might be interesting to also create a few duplicate variables

580 00:32:24.080 --> 00:32:27.289 that check the calculations performed by the functions.

581 00:32:27.380 --> 00:32:31.400 For example let's compare a manual arithmetic calculation

582 00:32:31.460 --> 00:32:33.714 of the sum of total cholesterol 1, 2, and 3 to

583 00:32:35.030 --> 00:32:37.970 the sum calculated using the SUM function.

584 00:32:38.120 --> 00:32:42.380 The end function reports the number of non missing observations

585 00:32:42.410 --> 00:32:46.039 in the arguments for each, and Miss reports the number

586 00:32:46.040 --> 00:32:47.689 of missing observations.

587 00:32:48.320 --> 00:32:52.009 Since we have three total cholesterol variables the sum of

588 00:32:52.010 --> 00:32:54.349 N and N Miss should equal three.

589 00:32:54.590 --> 00:32:58.220 Let's also check if the mean cholesterol calculated using the

590 00:32:58.250 --> 00:33:00.994 mean function equals the sum of all observations divided

591 00:33:02.060 --> 00:33:04.859 by the number of observations used in the sum.

592 00:33:04.940 --> 00:33:08.076 That is the number of non missing values in total cholesterol 1,

593 00:33:08.660 --> 00:33:11.502 2, and 3. Printing the first 10 observations in Framingham

594 00:33:12.440 --> 00:33:15.086 new, we see that individuals tend not to have complete

595 00:33:16.130 --> 00:33:18.559 cholesterol data in all three periods.

596 00:33:18.590 --> 00:33:22.369 But the arithmetic and statistical functions apply to these variables

597 00:33:22.460 --> 00:33:23.900 are not equal to missing.

598 00:33:24.020 --> 00:33:27.589 In contrast, the manual SUM check Sum is missing if

599 00:33:27.590 --> 00:33:30.320 any of the arguments in the manual some is missing.

600 00:33:30.440 --> 00:33:34.339 Notice that these functions analyze the data within each subject.

601 00:33:34.400 --> 00:33:37.820 We are not finding the means, say, over all subjects.

602 00:33:38.690 --> 00:33:41.728 You can again use SAS as a calculator to manually enter a list

603 00:33:42.440 --> 00:33:45.170 of values as arguments into the function.

604 00:33:45.490 --> 00:33:49.390 Here, I'm computing the mean of the values 1 2 3

605 00:33:49.490 --> 00:33:52.970 4 5 and the standard deviation of those data values.

606 00:33:53.090 --> 00:33:54.890 You can also nest functions.

607 00:33:54.950 --> 00:33:58.729 For example suppose I want to compute the square root of the sum

608 00:33:58.790 --> 00:34:00.489 of 1 2 3 4 5.

609 00:34:00.620 --> 00:34:03.560 We can do that without creating an intermediate variable for

610 00:34:04.130 --> 00:34:06.350 the sum. But you could also do it that way.

611 00:34:06.680 --> 00:34:10.218 Again print the data set to view the results of the calculation.

612 00:34:11.350 --> 00:34:14.919 As we would expect, the mean of that sequence is 3 and

613 00:34:14.920 --> 00:34:18.459 the standard deviation of those five numbers is reported along

614 00:34:18.460 --> 00:34:19.989 with the square root of the sum.

615 00:34:21.710 --> 00:34:25.099 Lastly, we're going to talk about working with date variables.

616 00:34:25.420 --> 00:34:28.939 In SAS, date variables are stored as numbers

617 00:34:29.000 --> 00:34:32.179 and are displayed to us using different date formats.

618 00:34:32.960 --> 00:34:36.468 We see here that our 2 date variables in Framingham,

619 00:34:36.560 --> 00:34:40.459 intake and date of death, were imported as numeric variables.

620 00:34:41.110 --> 00:34:44.629 The numeric value of the date is the number of days between

621 00:34:44.630 --> 00:34:48.439 January 1st 1960 and the specified date,

622 00:34:48.620 --> 00:34:52.219 that is day 0 in SAS is January 1st

623 00:34:52.340 --> 00:34:53.359 1960.

624 00:34:53.960 --> 00:34:57.469 Dates before January 1st 1960 are represented

625 00:34:57.470 --> 00:35:01.130 as negative numbers, and dates after as positive numbers.

626 00:35:01.220 --> 00:35:04.429 The variables date and SAS date are the same.

627 00:35:04.490 --> 00:35:08.329 The variable on the left has been formatted to display in a format

628 00:35:08.390 --> 00:35:11.420 we're used to seeing, while the variable on the right is an

629 00:35:11.480 --> 00:35:14.929 unformatted date showing you how SAS represents a date

630 00:35:14.960 --> 00:35:15.960 numerically.

631 00:35:16.700 --> 00:35:19.444 This numeric representation of dates allows us to easily

632 00:35:20.360 --> 00:35:23.300 find the number of days between two dates.

633 00:35:23.870 --> 00:35:27.409 You can use the arithmetic operation of subtraction to find the

634 00:35:27.410 --> 00:35:31.189 number of days between an end date and a start date.

635 00:35:31.250 --> 00:35:34.849 For example, to find the number of days between January 30th

636 00:35:35.340 --> 00:35:36.908 1948 and April 18th 1964, we see

637 00:35:38.960 --> 00:35:40.489 that the end date is

638 00:35:42.440 --> 00:35:44.253 1569 days after January 1st 1960, and

639 00:35:46.010 --> 00:35:49.519 the start date is 4354 days

640 00:35:49.610 --> 00:35:51.678 before January 1st 1960.

641 00:35:52.520 --> 00:35:56.059 To find the number of days in between we subtract the start date

642 00:35:56.060 --> 00:35:58.094 from the end date, which gives us 5923

643 00:36:00.110 --> 00:36:03.920 days. We can also convert those days to months

644 00:36:03.950 --> 00:36:06.769 or years to give a more meaningful timescale.

645 00:36:07.700 --> 00:36:10.689 In the Framingham data we have two dates, date of intake into

646 00:36:11.330 --> 00:36:14.564 the study and date of death when a death was observed during study

647 00:36:14.780 --> 00:36:18.659 follow up. To find the number of days between a participant's

648 00:36:18.660 --> 00:36:21.502 date of death and their intake date, we subtract the start

649 00:36:22.370 --> 00:36:25.760 date from the end date or intake from date of death.

650 00:36:26.770 --> 00:36:29.857 The new variable we're defining is called days to death, and we

651 00:36:30.310 --> 00:36:33.544 see that it is only defined for those individuals with non missing

652 00:36:34.090 --> 00:36:36.849 values for both intake and date of death.

653 00:36:37.300 --> 00:36:40.750 We can convert days to years by dividing the number of days

654 00:36:40.810 --> 00:36:42.970 by 365.25.

655 00:36:43.990 --> 00:36:47.439 With this many days a timescale of years is more appropriate.

656 00:36:48.310 --> 00:36:51.849 Finally, since we also have the variable called age that tells

657 00:36:51.850 --> 00:36:55.929 us the subject's age in years at intake, we can calculate

658 00:36:55.990 --> 00:36:58.685 an approximate age at death by adding years to death to

659 00:36:59.560 --> 00:37:00.670 the age variable.

660 00:37:01.330 --> 00:37:04.689 Note this is an approximate age of death because we do not have the

661 00:37:04.690 --> 00:37:07.030 individual's date of birth in these data.

662 00:37:07.720 --> 00:37:11.800 If we had date of birth we could calculate age of death by subtracting

663 00:37:11.950 --> 00:37:13.780 date of birth from date of death.

664 00:37:14.940 --> 00:37:18.389 If we want to define a new date variable in our data set to use in

665 00:37:18.390 --> 00:37:22.140 calculations, we can do so directly in a data step.

666 00:37:22.650 --> 00:37:26.399 For example, if I know that follow up for long term endpoints ended

667 00:37:26.400 --> 00:37:29.969 on December 31st 1972 in this study, I

668 00:37:29.970 --> 00:37:32.665 can define a new variable and follow up equals December

669 00:37:33.840 --> 00:37:35.909 31st 1972.

670 00:37:36.690 --> 00:37:39.510 I will show you two ways of defining the date variable.

671 00:37:39.690 --> 00:37:42.728 In the first method you enclose the date in quotes followed by

672 00:37:43.320 --> 00:37:46.920 the letter D. It is important to specify the day,

673 00:37:46.950 --> 00:37:48.809 followed by month, and then year.

674 00:37:49.470 --> 00:37:52.979 You must also use the three letter abbreviation for the month.

675 00:37:54.400 --> 00:37:57.909 The second method of defining a date variable uses the N

676 00:37:57.910 --> 00:37:59.690 D Y function in SAS.

677 00:38:00.190 --> 00:38:03.940 The arguments of the function are numerical month, followed by da,y

678 00:38:04.120 --> 00:38:07.239 and then year with each argument separated by a comma.

679 00:38:07.600 --> 00:38:10.060 Both methods produce identical results.

680 00:38:10.840 --> 00:38:14.650 Another useful function is the today function in SAS.

681 00:38:14.770 --> 00:38:18.550 This function will define a date variable using the current date.

682 00:38:18.670 --> 00:38:20.560 There are no arguments of this function.

683 00:38:21.130 --> 00:38:24.579 Because we're reading in the Framingham dataset you're using the SET

684 00:38:24.580 --> 00:38:28.059 statement, these new date variables will be defined for every

685 00:38:28.060 --> 00:38:29.570 subject in the dataset.

686 00:38:30.070 --> 00:38:33.579 We will see three new columns at the end of our dataset but the value

687 00:38:33.580 --> 00:38:36.070 of each date will be constant for all subjects.

688 00:38:36.640 --> 00:38:40.300 This is necessary if we want to use these newly defined variables

689 00:38:40.330 --> 00:38:44.469 to perform calculations involving existing date variables

690 00:38:44.560 --> 00:38:47.304 such as date of intake because SAS evaluates expressions

691 00:38:48.130 --> 00:38:51.520 involving variables in the data set for each observation.

692 00:38:52.530 --> 00:38:56.039 All of these methods for defining date variables will produce

693 00:38:56.070 --> 00:38:58.710 the date in the form of a SAS date value.

694 00:38:58.920 --> 00:39:02.699 This is fine if you just want to use the date to perform calculations.

695 00:39:02.820 --> 00:39:05.809 However, if you want to print the date variables it's helpful

696 00:39:06.360 --> 00:39:09.055 to apply a date format to the variables so that you can

697 00:39:09.900 --> 00:39:12.989 see the date presented in a way we are used to seeing.

698 00:39:13.530 --> 00:39:16.380 For example, the way intake is displayed here.

699 00:39:17.070 --> 00:39:20.609 Since the date variable intake was an existing variable

700 00:39:20.670 --> 00:39:23.169 in the imported Excel workbook, SAS recognized that

701 00:39:24.210 --> 00:39:26.807 this variable was a date and applied a date format to

702 00:39:27.750 --> 00:39:29.880 the variable when reading in the data.

703 00:39:30.300 --> 00:39:32.970 This is why it appears as a formatted date.

704 00:39:33.930 --> 00:39:36.180 We see this when we look at PROC CONTENTS.

705 00:39:36.240 --> 00:39:39.389 The format date 9 was applied to the variable intake.

706 00:39:40.110 --> 00:39:43.148 Formats are applied to variables in a data step using a format

707 00:39:43.770 --> 00:39:46.514 statement. After the word format specify the name of the

708 00:39:47.340 --> 00:39:50.489 variable that you would like to format followed by the name of the

709 00:39:50.490 --> 00:39:54.300 format. The format name must be followed by a period

710 00:39:54.390 --> 00:39:57.959 or decimal point. We closed the format statement with a

711 00:39:57.960 --> 00:39:58.960 semicolon.

712 00:39:59.490 --> 00:40:02.579 You can apply the same format to more than one variable.

713 00:40:02.610 --> 00:40:06.330 By including the format name after more than one variable name.

714 00:40:07.110 --> 00:40:10.770 You can also apply formats to multiple variables in your dataset

715 00:40:10.830 --> 00:40:12.690 using one format statement.

716 00:40:13.230 --> 00:40:16.709 SAS has many predefined date formats to choose from,

717 00:40:16.830 --> 00:40:18.600 even more than I have listed here.

718 00:40:19.410 --> 00:40:22.497 We are applying a different date format to each of these 4 date

719 00:40:23.010 --> 00:40:26.670 variables. Intake is formatted as date 9.

720 00:40:26.790 --> 00:40:30.450 9 refers to the length of the date variable under this format.

721 00:40:30.900 --> 00:40:33.018 End follow up is formatted as MMDDyy.

722 00:40:34.800 --> 00:40:38.429 The thing to notice about this format is that it uses a two

723 00:40:38.430 --> 00:40:42.110 digit year and follow up version to use its

724 00:40:42.180 --> 00:40:43.217 date format MMDDyy10.

725 00:40:45.680 --> 00:40:49.320 10 again refers to the length of the resulting formatted date

726 00:40:49.470 --> 00:40:53.219 including the forward slashes between month day and year.

727 00:40:53.880 --> 00:40:57.630 Finally today date uses date format word date

728 00:40:57.720 --> 00:41:00.840 which displays the date as we would in text form.

729 00:41:01.380 --> 00:41:04.409 Formats do not change the value of the variable.

730 00:41:04.470 --> 00:41:07.289 These dates are still thought of as numbers by SAS.

731 00:41:07.500 --> 00:41:11.340 Formats only change the way values of the variable are displayed.

732 00:41:12.930 --> 00:41:16.849 This concludes video 2.1 In this video, we reviewed

733 00:41:16.860 --> 00:41:19.604 PROC import. We also covered how to perform calculations

734 00:41:20.340 --> 00:41:23.939 and construct new variables using arithmetic operations and

735 00:41:23.940 --> 00:41:27.630 using mathematical, arithmetic, and statistical functions.

736 00:41:28.260 --> 00:41:31.170 Finally we discussed working with dates in SAS.

737 00:41:32.190 --> 00:41:34.689 In video 2.2 We will discuss creating new variables

738 00:41:35.880 --> 00:41:38.520 using comparison and logical operators.