

## 10 *Stata* Coding and Birthdays

### The Data

The Birthday Problem is a famous probability problem that helps us understand why coincidences occur. Suppose there are  $n$  people in the room. What's the probability that at least two have the same birthday? Under certain assumptions, and disregarding leap years, we can compute the probability to be

$$1 - \frac{365 \times 364 \times \dots \times (365 - n + 1)}{365^n}$$

This tells us that in a classroom of 32 people (substitute 32 for  $n$ ) the probability that at least two will share the same birthday is .753, and if there are 40 people in the classroom this probability goes up to .891. If there are 60 people, it is a near certainty.

One of the main assumptions behind this solution to this famous problem is that birthdays are uniformly distributed. In other words, people are just as likely to be born on one day of the year as on any other. But this assumption clashes with conventional wisdom that says, for example, more babies are born 9 months after Winter, or that more are born on New Year's Day. In this lab, we'll look at data for births in the U.S. for the year 1978 and examine just what the distribution is.

A side issue we must deal with is how to properly "code" data. Data often come to us in an un-useable form (or un-useable from the computer's perspective). Today you'll learn some techniques for "cleaning" data so that we can examine its distribution.

Some questions we'll investigate: **How does the distribution of births vary from month to month? How does this distribution vary from day to day?**

The data come from an article by Dr. Geoffrey Berresford: "The uniformity assumption in the birthday problem" published in *Mathematics Magazine*, 53(5): 286-288, 1980.

## Cleaning and Exploring

Load the data into *Stata* with the use command.

```
. use http://www.stat.ucla.edu/labs/datasets/birthdays.dta
```

It is always a good idea to begin by typing the **describe** command to get a general idea of what the data look like.

```
. describe
```

You should now see in your results window that you have two variables with 365 observations. The first variable is a string variable named *var1* and the second variable, *var2* is an integer variable. We can then use the *list* command to see the actual data.

```
. list
```

With 365 observations, we likely will not want to look at all of it, but just the first screen or so gives us an idea of what we are dealing with. Hitting the “q” or the apple key and the period at the same time will break scrolling run-away data screens. Any other key-stroke will work as a page-down command.

Use the rename command to give var1 and var2 descriptive names. Let’s call var1 “birthday” and var2 “numbirths.” This is done as follows:

```
. rename var1 birthday
```

Now rename var2 on your own.

Type the describe command again. Though this is an improvement over the original description, we can improve it even further using the **label** command.

```
. label variable birthday "Date of birth"  
. label variable numbirths "Number of births recorded"  
. describe
```

*Question 1: How has the output for the describe command changed?*

It is good practice to label and document data as much as possible. You

might think you will remember everything there is to remember, but a week later a lot might have changed.

*Question 2: Type “graph birthday” and “summarize birthday.” What happens?*

Before we can analyze this data, more has to take place than these simple cosmetic changes. The birthday variable, for instance, can't really be processed by most of our favorite functions. One approach to make this dataset more palatable is to turn the birthday variable into several different variables. In particular, since we want to study how the distribution varies from month to month, we might want to have a separate variable to indicate the month separately.

To do this, we need to use the date function in *Stata* combined with the generate and format functions. This takes several steps, so hang on.

The date function takes a string variable containing date information, and converts it to a form more easily manipulated by *Stata*. When calling the date function, you need to provide the name of the string variable you're converting as well as information about the format of that string. Since our birthday variable is in the form *January 1, 1978* we specify *mdy* since this has the month listed first, followed by the day, and lastly the year with all four digits. Noting this order before using the date command is important – remember to do this when completing your assignment for this lab. For more information about the date function and its use with other formats, you can type **help date** in your command window.

```
. generate bday=date(birthday, "mdy")  
. list birthday bday
```

*Note: it is important that “mdy” be enclosed in quotes.*

The dates are now integers in the new bday variable. They represent the number of days between that date and January 1, 1960. Though this may seem odd, such handling of dates is quite common in computer packages. One benefit to it is that computing the number of days between say June 3, 1983

and November 18, 1992 becomes nothing more than a simple subtraction problem.

We can improve this further by typing

```
. format bday %d
```

By reformatting to the %d format, the integer dates are converted to the form you see when you type the list command.

```
. list birthday bday
```

To extract the month and day information from this new date formatted variable we use the month and day functions, respectively.

```
. generate month = month(bday)
. generate day = day(bday)
. list birthday bday month day
```

Using similar commands in *Stata*, we can extract other useful date information.

```
. generate dayyear = doy(bday)
```

creates a variable that contains the number, 1 through 365, that corresponds to the day of the year. For instance, the first of January would take on the value 1 since it is the first day of the year. *Note: the middle letter of the command is an “o”; it is an acronym for Day Of Year.*

```
. generate dayweek = dow(bday)
```

creates a variable indicating what day of the week a date is. Sundays take on the value of 0, Mondays are coded as a 1, and so on. We can also create a variable describing which one of the 52 weeks in the year the day falls in, what half of the year a day falls in, or what quarter a day is in.

```
. generate week = week(bday)
. generate yearhalf = halfyear(bday)
. generate quarter = quarter(bday)
```

*Question 3: What day of the week did your birthday fall in 1978? What day of the year was that? In what week? (Hint: Using the **if** command can*

help us subset the data and concentrate on a smaller portion. For example, if your birthday is in July, typing `. list birthday quarter if month == 7` will list only the birthdays in July and what quarter each corresponding date is in.)

Now when you type in the describe command, you should have 10 variables with 365 observations.

*Question 4: Check to make sure that your data set now contains 10 variables and create descriptive labels for each of the newly generated variables as we did at the start of this lab. Are these new variables characters or numeric?*

## Daily Variations in Number of Births

How did the number of births vary from day to day in 1978?

*Question 5: What do you think a plot of number of births against the day of the year will look like?*

```
. graph numbirths dayyear
```

*Question 6: Describe this graph. What time of year has the most births? The least number of births? How does this graph differ from what you expected?*

*Question 7: The most surprising feature of this graph is the “shadow” curve. What do you think caused this?*

Before going on, we want to point out some of the choices you have to display graphs. In general, you want your plots to be as simple as possible, but sometimes it makes a graph easier to read if you can change the color, for example. To change what symbol is plotted, what color is plotted, and how our axes are labelled, try the command:

```
. graph numbirths dayyear, symbol(x) pen(6) xlabel ylabel  
title(Number of babies born on each day of the year in 1978)
```

*Question 8: How did the graph change with these additional commands? What color is pen color 6?*

Other plotting symbols include O for large circles, o for small circles, p for plus signs, d for diamonds, T for triangles, S for squares, . for dots, i for invisible (this will come in handy in other contexts that you'll come across), \_n for the observation numbers, and varname if you want to use the value of a variable as a plotting symbol. You can experiment to find other colors.

One of the exciting things about science is discovering a surprising pattern. We set out just to answer the simple question about the daily variation in the number of births. Instead, we find a mystery: two parallel sets of points. So before going on, we investigate.

One approach to investigating this is to “blow up” the scale and examine only a small portion of the graph, say the first 3 months:

```
. graph numbirths dayyear if quarter==1, symbol(p) pen(1)  
xlabel ylabel title(Number of babies born each day first  
quarter of 1978)
```

Now lets just graph the month of August to refine the scale even more.

```
. graph numbirths dayyear if month==8, symbol(T) pen(2) xlabel  
ylabel title(Number of babies born on each day in August of  
1978)
```

*Question 9: Looking at these two graphs with smaller scales, what do you think accounts for the parallel curve effect seen with this data? (Hint: how*

many lower points follow how many high points? You should notice a pattern. It might aid your eye even more to add the option “c(l)” to this graph. This option connects the dots: `. graph numbirths dayyear if month==8, symbol(T) pen(2) xlabel ylabel title(Number of babies born on each day in August of 1978) c(l)`

The plots we’ve seen so far are fine for looking at the day-to-day trend. But what if we want to answer more general questions, such as: How does the number of births vary by month? Or by season?

With these questions, we’re not so much interested in daily changes, but instead we want to summarize each month and compare months. Boxplots are one useful tool for comparing values from different groups.

```
. sort month
. graph numbirths, box by(month) title(Box-plots of births by month)
```

*Question 10: Is a randomly selected person more or less likely to be born in April than in September? What can you say about the distribution of the number of births within each month?*

*Question 11: Which season has the most births? Which season has the least?*

*Question 12: Create a boxplot that compares the distribution of the number of births for each quarter. How does it compare with what you thought it would look like?*

*Question 13: Describe what a boxplot would look like if we made a side-by-side boxplot of each day of the week. What would each box look like? Which*

*days would have the lowest median number of births? Do you think they'll all have the same amount of spread?*

*Question 14: Make a boxplot for each day of the week. Describe how close your prediction was in the last question. Notice that some days of the week have outliers.*

Let's say that instead of wanting box-plots for each day of the week separately, we want one box-plot for weekends and one box-plot for Monday through Friday. For this we need to create a new variable, let's call it *weekend*.

```
. generate weekend = dayweek
```

So far, all we've done is made a new variable called weekend that is identical to the dayweek variable we already had. Now what we want to do is recode the weekend variable.

```
. recode weekend 0=1 1=0 2=0 3=0 4=0 5=0 6=1
```

Now our weekend variable takes on a value of 1 if the day is a weekend, and 0 if the day is a weekday. Graphically we can see how weekends compare to weekdays using side-by-side box-plots again.

```
. sort weekend  
. graph numbirths, box by(weekend) title(Box-plots for  
weekdays vs weekends) ylabel
```

*Question 15: Based on your boxplot, about what would you guess was the average number of births for weekdays? For weekends?*

To get numerical summaries, we use the **by** option slightly differently.

```
. by weekend: summarize numbirths
```

*Question 16: How many babies are born during a weekend on average? How many babies are born on an average Tuesday? (You will need to sort and then summarize with another variable using the **by** command to answer the second question.)*

*Question 17: What accounts for those outliers on weekdays? Remember that during the year, there were a few weekdays that had a very low number of births compared to other weekdays. Justify your answer with the appropriate graph. What is peculiar about these particular dates? (Hint: To create the appropriate plot, we want to take advantage of a particular symbol option. Re-graph the boxplot you created for question ??, but add as one of the options **symbol([bday])**. This will label the points with the value stored in the variable **bday**.)*

## Assignment

The data set you will use for the assignment is rather famous. Our source for this assignment is the webpage of the Journal of Statistics Education, but the dataset is also found in a variety of other sources. The data involves the draft lottery of 1970. During the Vietnam war, young men were drafted for military service. In an attempt to expose them fairly to the risk of being drafted, a lottery was held on December 1, 1969, to allocate birth dates at random: 366 capsules, each containing a unique day of the year, were successively drawn from a container. The first date drawn (September 14) was assigned rank 1, the second date drawn (April 24) was assigned rank 2, and so on. Those eligible for the draft who were born on September 14 were called first for physicals, then those born on April 24 were tapped, and so on.

Soon after this lottery, an observed pattern of unfairness in the results led to considerable publicity. For the purposes of this lab, the year 1952 has been added to all of the listed dates to enable you to look at the day of the week variable. This year was selected both because it was a leap year, and because people born in 1952 first became eligible for the draft in 1970, the year they became 18 years old.

Enter the dataset into *Stata* by typing:

```
. use http://www.stat.ucla.edu/labs/datasets/draftlottery.dta
```

and use it to answer the following questions. Some alterations to the format of the data using the coding methods discussed earlier will be necessary.

*Question 18: What do var1 and var2 represent? Explain how you reached this conclusion.*

*Question 19: Using appropriate graphs **and** numerical summaries, describe any evidence of unfairness you see in the 1969 draft lottery. Explain what you mean by “unfair” and why the graphs and summaries you present support your conclusion.*