Starting R with Data and Graphics

Introduction to Statistical Modeling, Math 155

Abstract

This activity will help you get started with R using commands to make statistical graphics and exploring the different modes of graphics. Your deliverable this activity is to construct a brief report in Google Docs answering the questions posed below.

Many of the operations we will use in this course are extensions to R distributed by the `mosaic` package through the official R network. If you are using the server, this has already been installed. If you are using your own computer, ask the instructor to show you how to install the package. Then, load the package with this command:

To start, we'll work with the `Births78` dataframe. This is already built in to `mosaic`, so you already have it and can refer to it simply as `Births78`.

Here are some commands you can use to generate graphics. It's important that you realize that there are different modes — scatterplots, histograms, box-and-whisker plots.

Built-in datasets such as `Births78` have their codebook available via the `help()` command:

```
help(Births78)
```

In general, you will be given the names of any dataset you are to work with. These can be loaded with the `fetchData()` function, which works either for built-in data or for data available in CSV files. When you use `fetchData()`, you should do assignment to a named object so that you can refer to the fetched data in further commands. Some examples:

```
births = fetchData("Births78")
grades = fetchData("grades.csv")
feet = fetchData("KidsFeet")
```

Important commands with dataframes are:

```
names(Births78)  # names of the variables
```

```
## [1] "date"      "births"    "dayofyear"
```

```
nrow(Births78)  # how many cases
```

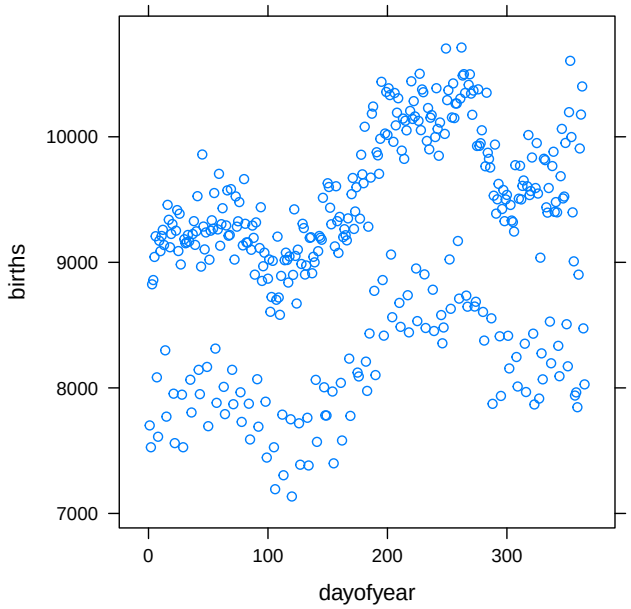
```
## [1] 365
```

```
head(Births78)  # show a snippet
```

```
##     date births dayofyear
## 1 1/1/78   7701         1
## 2 1/2/78   7527         2
## 3 1/3/78   8825         3
## 4 1/4/78   8859         4
## 5 1/5/78   9043         5
## 6 1/6/78   9208         6
```

Scatterplots are made with `xyplot()`. The first argument follows a very specific syntax called a "formula" that uses the `~` ("tilde") character. With `xyplot()`, the formula gives the y versus x variables. Try it:

```
xyplot(births ~ dayofyear, data = Births78)
```



Here are other graphics commands for some other common modes:

```
histogram(~births, data = Births78)
densityplot(~births, data = Births78)
bwplot(~births, data = Births78)
```

You might be wondering why there is nothing to the left of the `~`. That's because these graphics modes involve only one variable. When there is only one variable involved, it should go to the right of the tilde.

In your report:

- Make the four different modes of graphics as shown above — scatter plot, histogram, density plot, box-and-whisker plot — and paste the graphics into your report. Under each, explain what the graph is showing and why it looks the way it does.
- Make a histogram or density plot of the `dayofyear` variable. Explain why it looks the way it does.

As you may know, the number of births depends on the day of the week — Sunday, Monday, Tuesday, etc. — as well as the time of year. But the `dayofyear` variable does not directly give the day of year. A simple way to calculate the day of the week from `dayofyear` is to use a mathematical operation called the "modulo" or "remainder of division". (You don't have to master this. We're just using it here because it does the job at hand.) Here's the command that calculates the remainder of division by 7 and calls it `dayofweek`

```
Births78=transform(Births78,dayofweek=dayofyear%%7)
```

The division remainder is calculated by the **%%** operator. As it happens, for 1979 the first day of the year was a Sunday, so a value of 1 in **dayofweek** means Sunday, 2 means Monday, and so on.

When you want to break down a densityplot of one variable by a second variable like **dayofweek**, you can use a command like this:

```
densityplot( ~ births, groups=dayofweek, data=Births78 )
```

- Make this density plot, include it in your report, along with an explanation of why it looks the way it does. (Histograms do not let you do this, since you can't draw one histogram over another.)
- Use the **groups=dayofweek** argument in a scatter plot of births against **dayofyear**. Include the graph in your report and explain why it looks the way it does and how it relates to the density plot.

The formula syntax for graphics extends to allow you to specify a "conditioning" variable, making the graphics separately for each level of the conditioning variable. It works like this:

```
xyplot(births ~ dayofyear | dayofweek, data = Births78)
bwplot(~births | dayofweek, data = Births78)
densityplot(~births | dayofweek, data = Births78)
```

- Make each of the above plots, include them in your report, and explain how they are different from the corresponding graphs that don't condition on **dayofweek**.

You can calculate numerical measures such as the mean, standard deviation (**sd()**), median, maximum (**max()**), minimum (**min()**), or IQR using a similar formula syntax, for instance:

```
mean(~births, data = Births78)

## [1] 9132

mean(births ~ dayofweek, data = Births78)

##    0    1    2    3    4    5    6
## 8309 7951 9371 9709 9498 9484 9626
```

- In your report, say which day of the week has the smallest median. (Remember, Sunday is 1, Monday is 2, etc.)
- In your report, explain why

  ```
  mean(births ~ dayofyear, data = Births78)
  ```

  doesn't do anything useful for these data.

You can use **tally()** to count the number of cases that have a particular level of a variable.

- In your report, explain why **tally(~dayofweek,data=Births78)** gives the result it does.

Remember to hand in your report.