Lab 2: Summarizing and Visualizing Data

Overview

In this lab, we begin working with different types of variables. We obtain summaries of numeric and categorical variables, and we also do so by groups. We create new numeric and categorical variables. Lastly, we produce basic graphs for numeric and categorical variables

The Data: Babies' birth weight

Beginning in 1985, the US Surgeon General required the following warning to be placed on the side of cigarette packages:"SURGEON GENERAL'S WARNING: Smoking by Pregnant Women May Result in Fetal Injury, Premature Birth, and Low Birth Weight." Prior to the placement of the warning, studies had to be conducted to investigate the effects of smoking during pregnancy.

The data provided are part of the Child Health and Development Studies, a comprehensive investigation of all pregnancies between 1960 and 1967 among women in the Kaiser Foundation Health Plan in the San Francisco-East Bay area. Despite the warnings that went into effect in 1985, the National Center for Health Statistics found that 15% of women who gave birth in 1996 smoked during pregnancy.

Why do we care about baby birth weight? Birth weight is a measure of a baby's maturity. Typically, smaller babies have lower survival rates than larger babies. Babies that weigh under 5.5 pounds are considered low birth weight. In this study, the rate at which babies died within 28 days of birth was 150 per thousand births for low birth weight babies, compared to 5 per thousand for babies of 'normal' weight. The data set `babies.txt` contains seven variables:

variable	description
`bwt`	baby's weight at birth in ounces
`gestation`	length of pregnancy in days
`parity`	0=first born, 1=otherwise
`age`	mother's age in years
`height`	mother's height in inches
`weight`	mother's pregnancy weight in pounds
`smoke`	smoking status of mother: 0=not now, 1=yes now

Importing the data

You will need to download the text file `babies.txt` from Canvas to begin this lab. You should set up a QTM 100 folder for this class; furthermore, you should set up a QTM 100 datasets sub-folder within that folder. Store all of your datasets into this folder, including `babies.txt.`

To import the dataset, set your working directory to the folder that contains it, either using the RStudio interface (`More -> Set As Working Directory`) or `setwd` function and then use the `read.table` function as follows.

```r
setwd("~/Desktop/QTM_100_Spring_2023/Lab_Datasets/") # change this to your folder

babies <- read.table("babies.txt", header = TRUE)
```

Some points to remember:

1. The object's name (data frame) that contains your data need not necessarily be `babies`. You can choose any convenient name.
2. You need to use forward slashes "/" when providing folder or file paths. You must specify the paths in quotations as any other `string` variable.
3. To obtain the file path on Windows, locate the file. Then, hold the `Shift` key and right-click the file. Select `Copy as path`, then paste (`Ctrl + V`) it into your script, and replace backward slashes with forward slashes.
4. To get the file path on Mac, use 'Get Info. More details are available [here: (https://www.google.com/search?q=how+to+get+a+file+path+on+mac)].

Variable types

The data set structure reveals all variables in the data set, the types of variables (how they are stored in R), and the different values they take on.

```r
str(babies)
```

Here, we see that all variables are listed as `int` for integers because they take on whole number values. R uses this classification to determine how to treat the variable. Now view a summary of the data set.

```r
summary(babies)
```

Note that all variables are summarized numerically by providing means and quartiles, etc. However, the `parity` and `smoke` variables do not represent numeric measurements. Although the data values are stored as 0's and 1's, in reality, these 0's and 1's represent categories. The mean of the 0's and 1's is not a relevant summary of the `smoke` variable—instead, we would like to know how many mothers were smoking. For R to treat these appropriately as categorical variables, we need to recode them as factor variables.

```r
babies$parityf <- factor(babies$parity,labels=c("first born","otherwise"))
babies$smokef <- factor(babies$smoke,labels=c("not now","yes now"))
```

These two commands create two new variables in the `babies` data set. Here, we create a factor variable called `parityf` based on the original integer variable `parity`. The first label, firstborn corresponds to `parity = 0`, and the second label corresponds to `parity = 1`.

Now repeat the `str` and `summary` commands. With `str`, you should see that two new factor variables have been added to the data set. With `summary`, you should now see an appropriate summary of the two categorical variables, `parityf` and `smokef`. We can identify how many babies were firstborn and how many mothers were smoking. In the summary output, you should also notice that several variables have values of `NA`. It is how R stores missing values. For example, there were 13 babies that we do not know the gestation period.

Numeric Variables

Summarizing Numeric Variables

The `summary` command provides a useful overall numeric summary of the data set. You can use this for the entire data set, as we have already done, or just for single variables in the data set, for example.

```
summary(babies$bwt)
```

For example, if we want the average birth weight of babies, we use the `mean` command.

```
mean(babies$bwt)
```

Note that the summary of the data set did not provide the standard deviation of any numeric variables. To get the standard deviation, use the `sd` command.

```
sd(babies$bwt)
```

Other useful single numeric summaries include `min`, `max`, `median`, `range`, and `IQR`.

We can compare a numeric variable and the groups of a categorical variable. For example, if you wanted to determine if birth weight differs by whether or not the mother smoked? We should summarize birth weight separately for smoking and non-smoking mothers to answer this. More specifically, the typical approach compares the average birth weight among smoking and non-smoking mothers. We can use the `tapply` function for this purpose.

```
tapply(X = babies$bwt, INDEX = babies$smokef, FUN = sd)
```

The first argument is the numeric variable, and the second is the categorical variable. The third argument is the statistics you want to calculate—the name of the appropriate R function. Instead of `sd`, you could specify `mean`, `median`, or any other function that can be applied to a numeric variable.

If you want to compare the mean of `bwt` in groups of `parityf`, use

```
tapply(X = babies$bwt, INDEX = babies$parityf, FUN = mean)
```

Visualizing Numeric Variables

Common visualizations for numeric variables include histograms, boxplots, and stem and leaf plots.

```
hist(babies$bwt)
boxplot(babies$bwt)
```

You can add additional arguments to modify the graphs. For example, you can change the color or the axis labels for `hist` and `boxplot`.

Side-by-side boxplots are commonly used to visualize the distribution of a numeric variable by groups. To do this in R, use the same `boxplot` function. For example,

```
boxplot(babies$bwt ~ babies$smokef)
```

Here, the tilde means that birth weight is broken down by smoking status.

To visualize the relationship between two numeric variables, we can use the scatter plot. For example, a scatterplot of birth weight by gestation reveals that babies with more extended gestation periods tend to weigh more at birth.

```
plot(babies$gestation, babies$bwt)
```

The first argument specifies the x-axis variable, and the second is the y-axis.

Categorical Variables

Summarizing Categorical Variables

When summarizing categorical variables, it is interesting to know the frequency of occurrences of each level of the categorical variable. We can summarize this using the `table` command.

```
table(babies$smokef)
```

The result shows how many smoking and non-smoking mothers are present in the data. To see also the sum of observations (margins), we can use the `addmargins` function. This function takes a table as an argument. For convenience, we can create a variable that contains the former table and then apply `addmargins` to this variable.

```
smk.tab <- table(babies$smokef)
smk.tab
addmargins(smk.tab)
```

To see the proportions of observations in each group, use the `prop.table` function.

```
prop.table(smk.tab)
```

Now we know that 742 non-smokers out of 1226 mothers in the data set represent 60.5% of the mothers in the study.

What if we wanted to know about the relationship between two categorical variables? It is summarized in a two-by-two contingency table. For example,

```
table(babies$smokef, babies$parityf)
```

You can obtain the margins and proportions for a 2-by-2 table the same way as for a one-variable table.

```
smk.par.tab <- table(babies$smokef,babies$parityf)
addmargins(smk.par.tab)
prop.table(smk.par.tab)
```

Notice that `prop.table` above gives you the overall proportions, i.e., the values in the table add up to 1. For example, 44.7% of the mothers had firstborn babies and were not smokers among all mothers in the data set: 44.7% = 548/1226 * 100%.

Alternatively, we can calculate row or column proportions as follows.

```
prop.table(smk.par.tab, margin = 1)    # row proportions
prop.table(smk.par.tab, margin = 2)    # column proportions
```

When calculating a row proportion, the denominator is the sum for the row, i.e., values in each row add up to 1. From the row proportions, we can see that out of the 742 non-smoking mothers, 548 had a firstborn baby or 73.9%. In the case of column proportions, the values in each column add up to 1. From the column proportions, we can see that out of the 911 firstborn babies, 548 had a non-smoking mother or 60.2%. You would use row proportions, column proportions, or overall proportions, depending on your research question.

Visualizing Categorical Variables

Basic visualizations for categorical variables include pie charts and bar plots. When creating these graphs, we need to produce them based on the `table` of the variable(s). For example,

```
barplot(smk.tab)
```

Side-by-side bar plots can be produced using the same function, using the command `beside.` Whenever we use counts, we always use side-by-side bar plots. It is not acceptable to utilize stacked bar plots.

```
barplot(smk.par.tab, beside = T, legend.text = T)
```

Notice that `T` is equivalent to `TRUE`; you can use either of them. The `beside = T` option places the bars beside each other (the default is `beside = F`, which creates a "stacked" bar plot). The `legend.text = T` argument produces a corresponding legend.

When comparing two groups in a bar chart, it is often best to use proportions in your bar plots instead of counts. We can do this by taking the proportions from our counts table, just like earlier in this manual. Just make sure that your outcome/response variable is listed second and select *column* proportions.

```
# Create barplot with proportions
barplot(prop.table(smk.par.tab, margin=2), beside = F, legend.text = T)
```

When creating a barplot with proportions, it is okay to use stacked bars. If you did your code correctly, each bar should add up to 1. Either side-by-side/dodged or stacked bars are acceptable to submit for proportional bar charts.