Categorical Data

Lt Col Ken Horton Professor Bradley Warner

27 August, 2020

Objectives

1) Define and use properly in context all new terminology.
2) Generate in `R` tables for categorical variable(s).
3) Generate in `R` appropriate graphical summaries of categorical and numerical variables.
4) Be able to interpret and explain output both graphically and numerically.

Categorical data

Like numerical data, categorical data can also be organized and analyzed. This section introduces tables and other basic tools for categorical data. Remember at the beginning of this block of material, our case study had categorical data so we have seen some of the ideas in this lesson.

The `email50` data set represents a sample from a larger email data set called `email`. This larger data set contains information on 3,921 emails. In this section we will use the email data set to examine whether the presence of numbers, small or large, in an email provides any useful value in classifying email as spam or not spam.

Contingency tables and bar plots

In the `email` data set we have two variables: `spam` and `number` that we want to summarize. Let's use `inspect()` to get information and insight about the two variables. We can also type `?email` to learn more about the data. First load the `openintro` library.

```r
library(openintro)
```

```r
email %>%
  select(spam,number) %>%
  inspect()
```

```
##
## categorical variables:
##     name  class levels    n missing
## 1 number factor      3 3921       0
##                                    distribution
## 1 small (72.1%), none (14%) ...
##
```

```
## quantitative variables:
##      name   class min Q1 median Q3 max       mean        sd    n missing
## ...1 spam numeric   0  0      0  0   1 0.09359857 0.2913066 3921       0
```

Notice the use of the `pipe` operator and how it adds to the ease of reading the code. The `select()` function allows us to narrow the variables down to the two of interest. Then `inspect()` gives us information about those variables. We read from top line; we start with the data set `email`, input it into `select()` and select variables from it, and then use `inspect()` to summarize the variables.

As is indicated `number` is a categorical variable that describes whether an email contains no numbers, only small numbers (values under 1 million), or at least one big number (a value of 1 million or more). The variable `spam` is a numeric variable where `1` indicates the email is spam. To treat it as categorical we will want to change it to a **factor** but first we will build a table that summarizes data for the two variables. This table is called a **contingency table**. Each value in the table represents the number of times a particular combination of variable outcomes occurred. We will show you the code to generate the contingency table.

```
tally(~spam+number,data=email,margins = TRUE)
```

```
##        number
## spam    none small  big Total
##   0      400  2659  495  3554
##   1      149   168   50   367
##   Total  549  2827  545  3921
```

The value 149 corresponds to the number of emails in the data set that are spam *and* had no number listed in the email. Row and column totals are also included. The **row totals** provide the total counts across each row (e.g. $149 + 168 + 50 = 367$), and **column totals** are total counts down each column. The row and column totals are known as **marginal** counts and the values in the table, such as 149, as **joint** counts.

Let's turn `spam` into a factor and update the `email` data object. We will use `mutate()` to do this.

```
email <- email %>%
  mutate(spam = factor(email$spam,levels=c(1,0),labels=c("spam","not spam")))
```

Now checking the data again.

```
email %>%
  select(spam,number) %>%
  inspect()
```

```
##
## categorical variables:
##     name  class levels    n missing
## 1   spam factor      2 3921       0
## 2 number factor      3 3921       0
##                                    distribution
## 1 not spam (90.6%), spam (9.4%)
## 2 small (72.1%), none (14%) ...
```

Let's generate the table again.

```r
tally(~spam+number,data=email,margins = TRUE)
```

```
##           number
## spam       none small  big Total
##   spam      149   168   50   367
##   not spam  400  2659  495  3554
##   Total     549  2827  545  3921
```

A table for a single variable is called a **frequency table**. The table below is a frequency table for the `number` variable.

```r
tally(~number,data=email)
```

```
## number
##  none small   big
##   549  2827   545
```

If we replaced the counts with percentages or proportions, the table would be called a **relative frequency table**.

```r
tally(~number,data=email,format='proportion')
```

```
## number
##      none     small       big
## 0.1400153 0.7209895 0.1389952
```

```r
round(tally(~number,data=email,format='percent'),2)
```

```
## number
##  none small   big
##  14.0  72.1  13.9
```

A bar plot is a common way to display a single categorical variable. Figure 1 shows a **bar plot** for the `number` variable.

```r
email %>%
  gf_bar(~number)
```

Next the counts are converted into proportions (e.g. 549/3921 = 0.140 for `none`) in Figure 2.

```r
email %>%
  gf_props(~number)
```

Again, let's clean up the plot

```r
email %>%
  gf_props(~number,title="The proportions of emails with a number in it",
           subtitle="From 2012",xlab="Type of number in the email",ylab="Proportion of emails") %>%
  gf_theme(theme_bw())
```

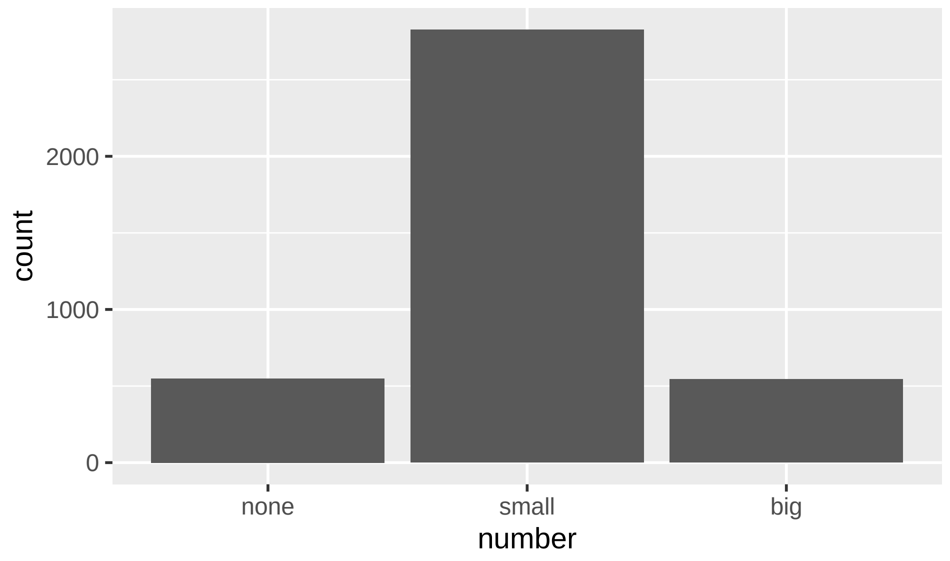


Figure 1: Bar plot of the variable number.

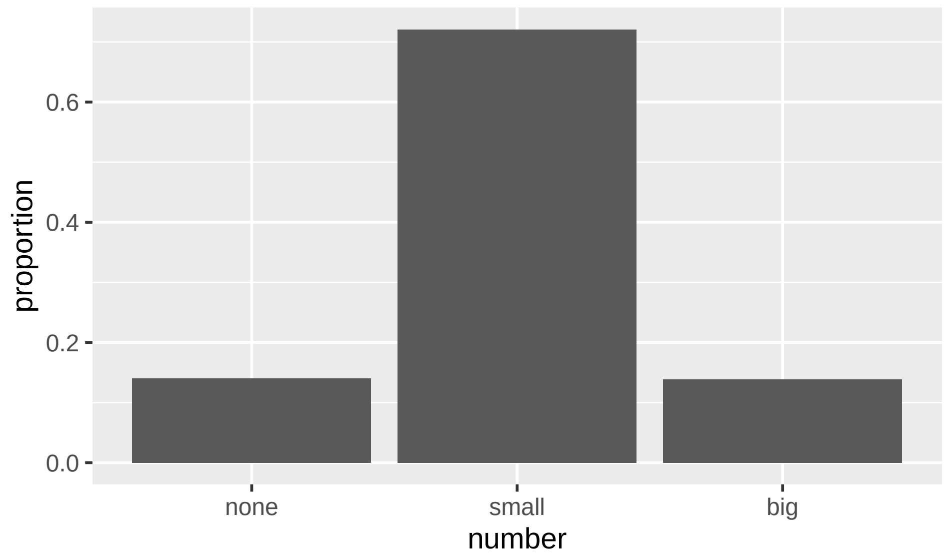
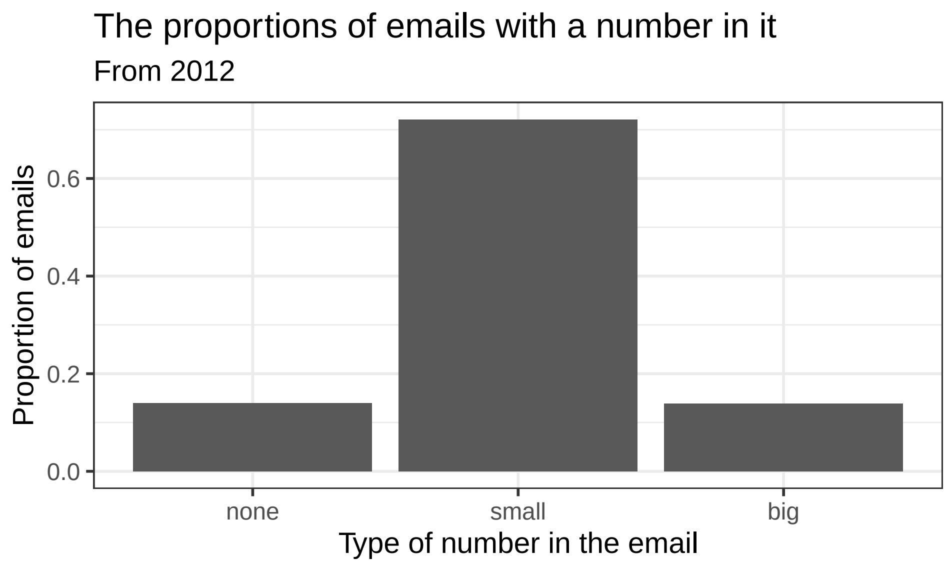


Figure 2: Bar plot of the variable number as a proportion.



Column proportions

The table below shows the column proportions. The **column proportions** are computed as the counts divided by their column totals. The value 149 at the intersection of *spam* and *none* is replaced by 149/549 = 0.271, i.e. 149 divided by its row total, 549. So what does 0.271 represent? It corresponds to the proportion of emails in the sample with no numbers that are spam. We are **conditioning**, restricting, on emails with no number. This rate of spam is much higher than emails with only small numbers (5.9%) or big numbers (9.2%). Because these spam rates vary between the three levels of `number` (*none*, *small*, *big*), this provides evidence that the `spam` and `number` variables are associated.

```
tally(spam~number,data=email,margins = TRUE,format='proportion')
```

```
##           number
## spam             none      small        big
##   spam     0.27140255 0.05942695 0.09174312
##   not spam 0.72859745 0.94057305 0.90825688
##   Total    1.00000000 1.00000000 1.00000000
```

The `tally()` function will always condition on the variable on the right hand side of the tilde, ~, when calculating proportions and thus only generate column proportions. The more general `table()` function of `R` will allow either column or row proportions.

> **Exercise**:
> Create a table of column proportions where the variable `spam` is the column variable.

```
tally(number~spam,data=email,margins = TRUE,format='proportion')
```

```
##        spam
## number       spam  not spam
##   none  0.4059946 0.1125492
##   small 0.4577657 0.7481711
##   big   0.1362398 0.1392797
##   Total 1.0000000 1.0000000
```

Exercise:
In the table you just created, what does 0.748 represent?[1]

Example:
Data scientists use statistics to filter spam from incoming email messages. By noting specific characteristics of an email, a data scientist may be able to classify some emails as spam or not spam with high accuracy. One of those characteristics is whether the email contains no numbers, small numbers, or big numbers. Another characteristic is whether or not an email has any HTML content. A contingency table for the `spam` and `format` variables is needed.
1 Make `format` into a categorical factor variable.The levels should be "text" and "HTML".[2]
2 Create a contingency table from the `email` data set with `format` in the columns and `spam` in the rows.

In deciding which variable to use as a column, the data scientist would be interested in how the proportion of spam changes within each email format. This corresponds to column proportions based on `format`: the proportion of spam in plain text emails and the proportion of spam in HTML emails.

```
email <- email %>%
  mutate(format = factor(email$format,levels=c(1,0),labels=c("HTML","text")))
```

```
tally(spam~format,data=email,margins = TRUE,format="proportion")
```

```
##           format
## spam             HTML       text
##   spam     0.05796038 0.17489540
##   not spam 0.94203962 0.82510460
##   Total    1.00000000 1.00000000
```

In generating the column proportions, we can see that a higher fraction of plain text emails are spam (209/1195 = 17.5%) than compared to HTML emails (158/2726 = 5.8%). This information on its own is insufficient to classify an email as spam or not spam, as over 80% of plain text emails are not spam. Yet, when we carefully combine this information with many other characteristics, such as `number` and other variables, we stand a reasonable chance of being able to classify some email as spam or not spam.

In constructing a table, we need to think about which variable we want in the column and which in the row. The formula format in some way makes us think about the response and predictor variables. However in some cases, it is not clear which variable should be in the column and row and the analyst must decide the point to be made with the table. Before settling on one form for a table, it is important to consider the audience and the message they are to receive from the table.

Exercise:
Create two tables with `number` and `spam` where each are in the column, so two table where you change which variable is in the column. Which would be more useful to someone hoping to identify spam emails using the `number` variable?[3]

```
tally(spam~number,email,format='proportion',margin=TRUE)
```

[1]0.748 represents the proportions of emails with no spam that had a small number in it.
[2]From the help menu on the data HTML is coded as a 1
[3]The column proportions with `number` in the columns will probably be most useful, which makes it easier to see that emails with small numbers are spam about 5.9% of the time (relatively rare). We would also see that about 27.1% of emails with no numbers are spam, and 9.2% of emails with big numbers are spam.

```
##            number
## spam              none      small        big
##   spam      0.27140255 0.05942695 0.09174312
##   not spam 0.72859745 0.94057305 0.90825688
##   Total     1.00000000 1.00000000 1.00000000
```

```
tally(number~spam,email,format='proportion',margin=TRUE)
```

```
##         spam
## number        spam  not spam
##   none   0.4059946 0.1125492
##   small 0.4577657 0.7481711
##   big    0.1362398 0.1392797
##   Total 1.0000000 1.0000000
```

Segmented bar and mosaic plots

Contingency tables using column proportions are especially useful for examining how two categorical variables are related. Segmented bar and mosaic plots provide a way to visualize the information in these tables.

A **segmented bar plot** is a graphical display of contingency table information. For example, a segmented bar plot representing the table with `number` in the column is shown in Figure 3, where we have first created a bar plot using the `number` variable and then separated each group by the levels of `spam`.

```
email %>%
  gf_bar(~number,fill=~spam)
```

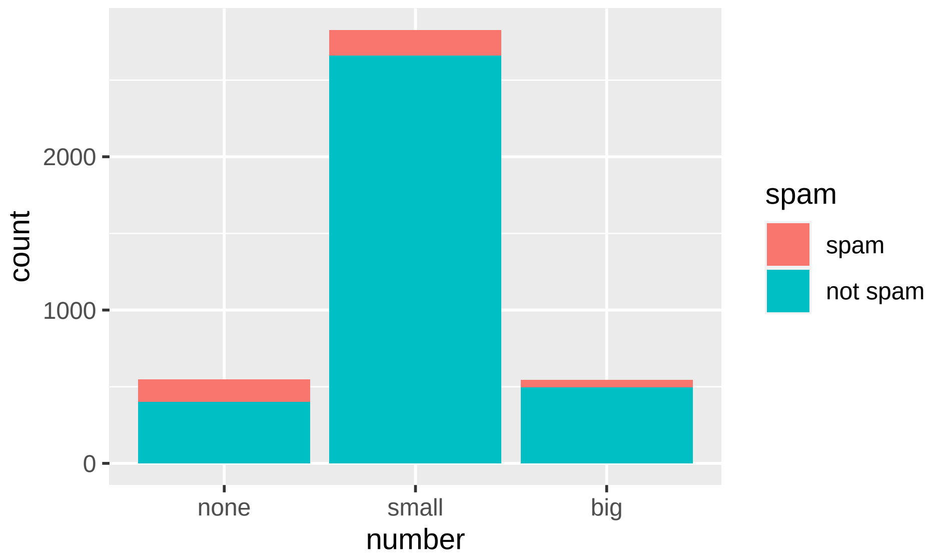


Figure 3: Segmented bar plot for numbers found in emails, where the counts have been further broken down by spam

The column proportions of the table have been translated into a standardized segmented bar plot in Figure 4, which is a helpful visualization of the fraction of spam emails in each level of `number`.

```
email %>%
  gf_props(~number,fill=~spam,position='fill')
```

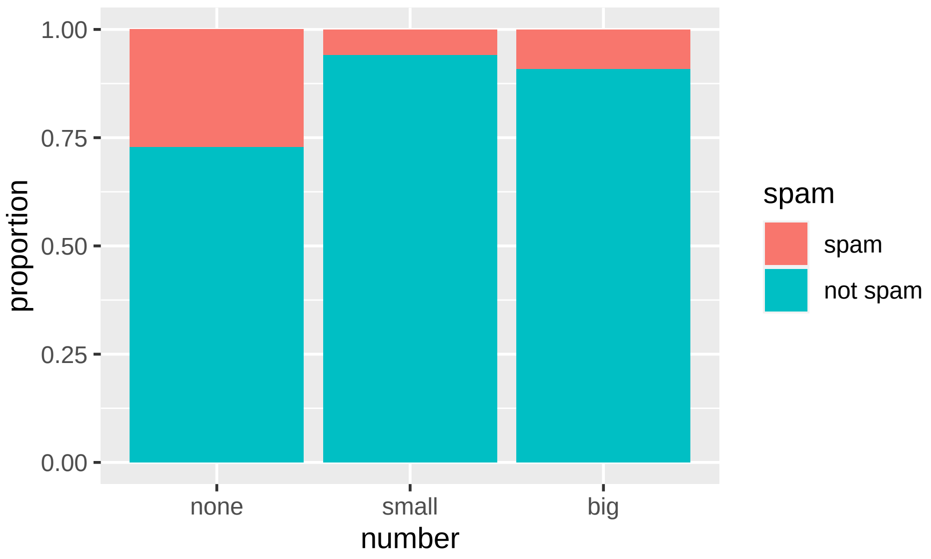


Figure 4: Standardized version of Figure 3

> *Example*:
> Examine both of the segmented bar plots. Which is more useful?

Figure 3 contains more information, but Figure 4 presents the information more clearly. This second plot makes it clear that emails with no number have a relatively high rate of spam email – about 27%! On the other hand, less than 10% of email with small or big numbers are spam.

Since the proportion of spam changes across the groups in Figure 4, we can conclude the variables are dependent, which is something we were also able to discern using table proportions. Because both the `none` and `big` groups have relatively few observations compared to the `small` group, the association is more difficult to see in Figure 3.

In some other cases, a segmented bar plot that is not standardized will be more useful in communicating important information. Before settling on a particular segmented bar plot, create standardized and non-standardized forms and decide which is more effective at communicating features of the data.

A **mosaic plot** is a graphical display of contingency table information that is similar to a bar plot for one variable or a segmented bar plot when using two variables. It seems strange, but mosaic plots are not part of the `mosaic` package. We must load another set of packages called `vcd` and `vcdExtra`. Mosaic displays help to visualize the pattern of associations among variables in two-way and larger tables. Mosaic plots are controversial since they rely on the perception of area. Human vision is not good at distinguishing areas.

We will introduce mosaic plots because it is another way to visualize contingency tables. Figure 5 shows a mosaic plot for the `number` variable. Each row represents a level of `number`, and the row heights correspond to the proportion of emails of each number type. For instance, there are fewer emails with no numbers than emails with only small numbers, so the `none` outcome row is shorter in height. In general, mosaic plots use box *areas* to represent the number of observations. Since there is only one variable, the widths are all constant. Thus area is simply related to row height making this visual easy to read.

```
library(vcd)
```

```
mosaic(~number,data=email)
```

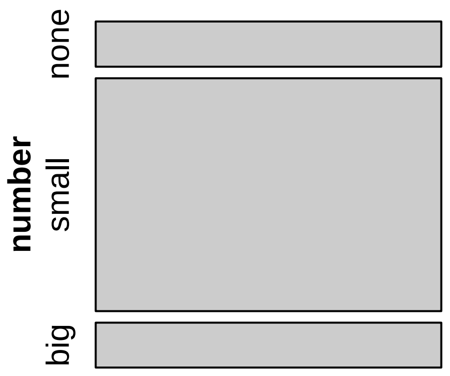


Figure 5: Mosaic plot where emails are grouped by the number variable.

This one-variable mosaic plot can be further divided into pieces as in Figure 6 using the `spam` variable. The first variable in the formula is used to determine row height. That is, each row is split proportionally according to the fraction of emails in each number category, these heights are similar to Figure 5. Next each row is split horizontally according to the proportion of emails that were spam in that number group. For example, the second row, representing emails with only small numbers, was divided into emails that were spam (left) and not spam (right). The area of the rectangles is proportional to the proportions in the table where each cell count is divided by the total count. First we will generate the table and then represent it as a mosaic plot.

```
tally(~number+spam,data=email,format='proportion')
```

```
##        spam
## number        spam   not spam
##   none  0.03800051 0.10201479
##   small 0.04284621 0.67814333
##   big   0.01275185 0.12624331
```

```
mosaic(~number+spam,data=email)
```

These plots are hard to use in a visual comparison of area. For example, is the area for *small* number *spam* emails different from *none* number *spam* emails? The rectangles have different shapes but from the table we can tell the areas are close.

An important use of the mosaic plot is to determine if an association between variables may be present. The bottom of the first column represents spam emails that had big numbers, and the bottom row of the second column represents regular emails that had big numbers. We can again use this plot to see that the `spam` and `number` variables are associated since some rows are divided in different vertical locations than others,

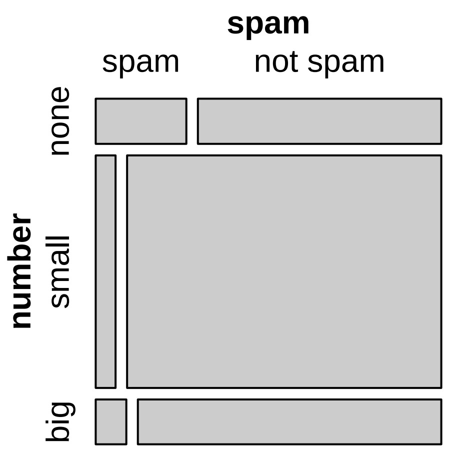


Figure 6: Mosaic plot where emails are grouped by the number variable.

which was the same technique used for checking an association in the standardized version of the segmented bar plot.

In a similar way, a mosaic plot representing column proportions where *spam* is in the column could be constructed. To completely understand the mosaic plot as shown in Figure 7 let's first find the proportions of `spam`.

```
tally(~spam,data=email,format="proportion")
```

```
## spam
##       spam   not spam
## 0.09359857 0.90640143
```

So the row heights will be split 90-10. Next let's find the proportions of number within each value of spam. In the spam row, *none* will be 41%, *small* will be 46%, and *big* will be 13%.

```
tally(number~spam,data=email,margins = TRUE,format="proportion")
```

```
##        spam
## number       spam  not spam
##   none  0.4059946 0.1125492
##   small 0.4577657 0.7481711
##   big   0.1362398 0.1392797
##   Total 1.0000000 1.0000000
```

However, because it is more insightful for this application to consider the fraction of spam in each category of the `number` variable, we prefer Figure 6.

```
mosaic(~spam+number,data=email)
```

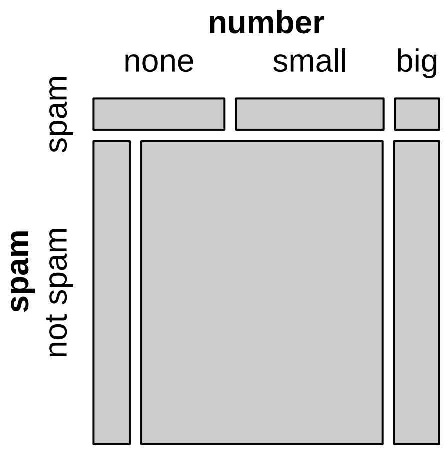


Figure 7: Mosaic plot with spam as the first variable

The only pie chart you will see in this course, hopefully

While pie charts are well known, they are not typically as useful as other charts in a data analysis. A **pie chart** is shown in Figure 8. It is generally more difficult to compare group sizes in a pie chart than in a bar plot, especially when categories have nearly identical counts or proportions. In the case of the *none* and *big* categories, the difference is so slight you may be unable to distinguish any difference in group sizes.

```
pie(table(email$number), col=COL[c(3,1,2)], radius=0.75)
```

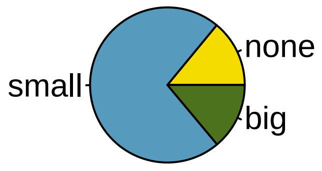


Figure 8: A pie chart number for the email data set.

Pie charts are popular in the Air Force due to the ease of generating them in Excel and PowerPoint. However, the values for each slice are often printed on top of the chart making the chart irrelevant. We recommend a minimum use of pie charts in your work.

Comparing numerical data across groups

Some of the more interesting investigations can be considered by examining numerical data across groups. This is the case where one variable is categorical and the other is numerical. The methods required here aren't really new. All that is required is to make a numerical plot for each group. Here two convenient methods are introduced: side-by-side box plots and density plots.

We will take a look again at the subset of `county_complete` data set and compare the median household income for counties that gained population from 2000 to 2010 versus counties that had no gain. While we might like to make a causal connection here, remember that these are observational data and so such an interpretation would be unjustified.

This section will give us a chance to perform some data wrangling. We will be using the `tidyverse` verbs in the process. Data wrangling is an important part of analysis work and typically takes a significant portion of the analysis work.

Here is the code to generate the data we need.

```
library(usdata)
```

```
county_M377 <- county_complete %>%
  select(name, state, pop2000, pop2010, fed_spend=fed_spending_2009, poverty=poverty_2010,
         homeownership = homeownership_2010, multi_unit = housing_multi_unit_2010,
         income = per_capita_income_2010, med_income = median_household_income_2010) %>%
  mutate(fed_spend=fed_spend/pop2010)
```

First, as a reminder, let's look at the data.

What do we want `R` *to do?* We want to select the variables `pop2000`, `pop2010`, and `med_income`.

What does `R` *need?* It needs the data object, and variable names.

We will use the `select()` and `inspect()` functions.

```
county_M377 %>%
  select(pop2000,pop2010,med_income) %>%
  inspect()
```

```
##
## quantitative variables:
##            name   class   min       Q1 median    Q3     max     mean        sd
## ...1    pop2000 numeric    67 11223.50  24621 61775 9519338 89649.99 292547.67
## ...2    pop2010 numeric    82 11114.50  25872 66780 9818605 98262.04 312946.70
## ...3 med_income numeric 19351 36956.25  42450 49144  115574 44274.12  11547.49
##         n missing
## ...1 3139       3
## ...2 3142       0
## ...3 3142       0
```

Notice that three counties are missing population values, reported as `NA`. Let's remove them and find which counties increased population by creating a new variable.

```r
cc_reduced <- county_M377 %>%
  drop_na(pop2000) %>%
  select(pop2000,pop2010,med_income) %>%
  mutate(pop_gain = sign(pop2010-pop2000))
```

```r
tally(~pop_gain,data=cc_reduced)
```

```
## pop_gain
##   -1    0    1
## 1097    1 2041
```

There were 2,041 counties where the population increased from 2000 to 2010, and there were 1,098 counties with no gain, only 1 county had a net of zero, or a loss. Let's just look at the counties with a gain or loss in side-by-side boxplot. Again, we will use `filter()` to select the two groups and then make the variable `pop_gain` into a categorical variable, more data wrangling.

```r
cc_reduced <- cc_reduced %>%
  filter(pop_gain != 0) %>%
  mutate(pop_gain = factor(pop_gain,levels=c(-1,1),labels=c("Loss","Gain")))
```

```r
inspect(cc_reduced)
```

```
##
## categorical variables:
##       name  class levels    n missing
## 1 pop_gain factor      2 3138       0
##                                    distribution
## 1 Gain (65%), Loss (35%)
##
## quantitative variables:
##            name   class   min       Q1  median      Q3     max     mean
## ...1    pop2000 numeric    67 11217.25 24608.0 61783.5 9519338 89669.37
## ...2    pop2010 numeric    82 11127.00 25872.0 66972.0 9818605 98359.23
## ...3 med_income numeric 19351 36950.00 42443.5 49120.0  115574 44253.24
##             sd    n missing
## ...1 292592.28 3138       0
## ...2 313133.28 3138       0
## ...3  11528.95 3138       0
```

The **side-by-side box plot** is a traditional tool for comparing across groups. An example is shown in Figure 9 where there are two box plots, one for each group and drawn on the same scale.

```r
cc_reduced %>%
  gf_boxplot(med_income~pop_gain,
             subtitle="The income data were collected between 2006 and 2010.",
             xlab="Population change from 2000 to 2010",
             ylab="Median Household Income")
```

Another useful plotting method uses **density plots** to compare numerical data across groups. A histogram bins data but is highly dependent on the number and boundary of the bins. A density plot also estimates the distribution of a numerical variable but does this by estimating the density of data points in a small

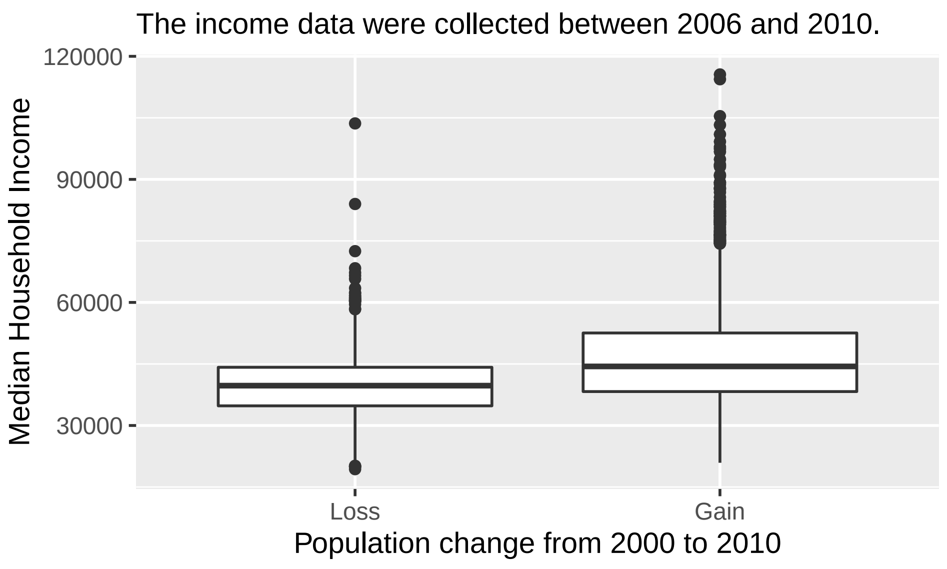


Figure 9: Side-by-side box plot for median household income, where the counties are split by whether there was a population gain or loss from 2000 to 2010.

window around each data point. The overall curve is the sum of this small density estimate. A density plot can be thought of as a smooth version of the histogram. Several options go into a density estimate such as the width of the window and type of smoothing function. These ideas are beyond the point here and we will just use the default options. Figure 10 is a plot of the two density curves.

```
cc_reduced %>%
  gf_dens(~med_income,color=~pop_gain,lwd=1)
```

Exercise:
Use the box plots and density plots to compare the incomes for counties across the two groups. What do you notice about the approximate center of each group? What do you notice about the variability between groups? Is the shape relatively consistent between groups? How many *prominent* modes are there for each group?[4]

Exercise:
What components of each plot in Figures 8 and 9 do you find most useful?[5]

File Creation Information

- File creation date: 2020-08-27
- Windows version: Windows 10 x64 (build 18362)

[4]Answers may vary a little. The counties with population gains tend to have higher income (median of about $45,000) versus counties without a gain (median of about $40,000). The variability is also slightly larger for the population gain group. This is evident in the IQR, which is about 50% bigger in the *gain* group. Both distributions show slight to moderate right skew and are unimodal. There is a secondary small bump at about $60,000 for the *no gain* group, visible in the density plot, that seems out of place. (Looking into the data set, we would find that 8 of these 15 counties are in Alaska and Texas.) The box plots indicate there are many observations far above the median in each group, though we should anticipate that many observations will fall beyond the whiskers when using such a large data set.

[5]The side-by-side box plots are especially useful for comparing centers and spreads, while the density plots are more useful for seeing distribution shape, skew, and groups of anomalies.

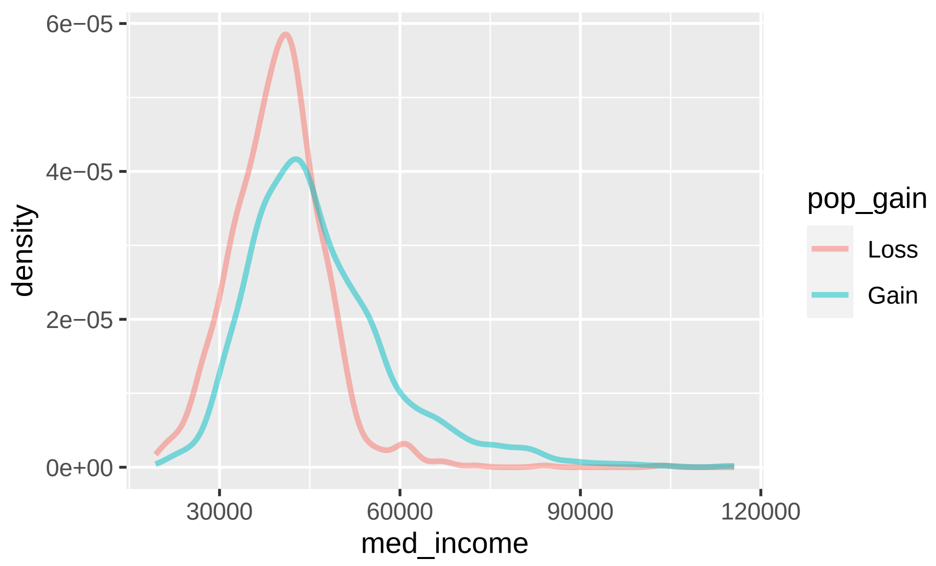


Figure 10: Density plots ofmedian household income for counties with population gain versus population loss

- R version 3.6.3 (2020-02-29)
- `mosaic` package version: 1.7.0
- `tidyverse` package version: 1.3.0