

Introduction to R, module04

Steve Simon

Created 2016-08-13

Introduction

This Powerpoint presentation was created using an R Markdown file. It is always a good idea to print the version of R and the current date as the very first few lines of any R program.

```
R.version.string  
## [1] "R version 4.0.0 (2020-04-24) "  
Sys.Date()  
## [1] "2021-03-07"
```

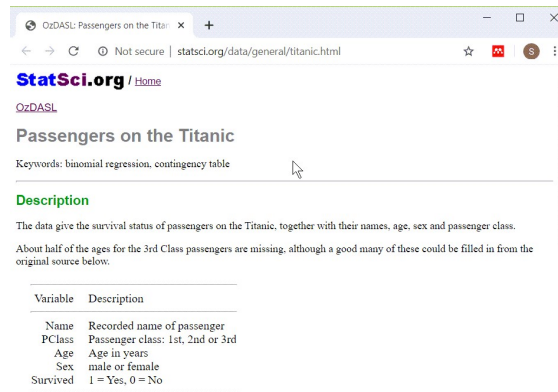
The version of R is stored in an internal constant, `R.version.string`. You can find the current data with the `Sys.Date()` function.

The Titanic dataset

- Titanic dataset
 - <http://www.statsci.org/data/general/titanic.txt>
- Titanic data dictionary
 - <http://www.statsci.org/data/general/titanic.html>

You should find the data file on mortality among passengers of the Titanic at the website listed here. There is a pretty good data dictionary as well.

Titanic data dictionary



StatSci.org / Home

OzDASL

Passengers on the Titanic

Keywords: binomial regression, contingency table

Description

The data give the survival status of passengers on the Titanic, together with their names, age, sex and passenger class.

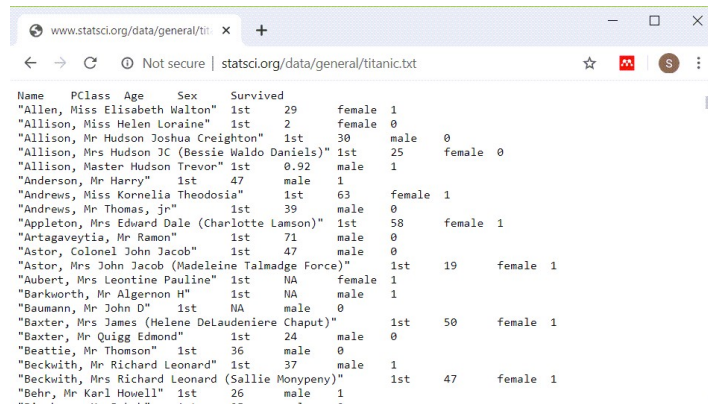
About half of the ages for the 3rd Class passengers are missing, although a good many of these could be filled in from the original source below.

Variable	Description
Name	Recorded name of passenger
PClass	Passenger class: 1st, 2nd or 3rd
Age	Age in years
Sex	male or female
Survived	1 = Yes, 0 = No

Screenshot from Titanic data dictionary

There are five variables in this data set, the name of the passenger, the passenger class, the passenger age, sex, and whether they survived.

What the file looks like



Name	PClass	Age	Sex	Survived					
"Allen, Miss Elisabeth Walton"	1st	29	female	1					
"Allison, Miss Helen Lovaine"	1st	2	female	0					
"Allison, Mr Hudson Joshua Creighton"	1st	30	male	0					
"Allison, Mrs Hudson JC (Bessie Waldo Daniels)"	1st	25	female	0					
"Allison, Master Hudson Trevor"	1st	0.92	male	1					
"Anderson, Mr Harry"	1st	47	male	1					
"Andrews, Miss Kornelia Theodosia"	1st	63	female	1					
"Andrews, Mr Thomas, jr"	1st	39	male	0					
"Appleton, Mrs Edward Dale (Charlotte Lamson)"	1st	58	female	1					
"Antagaveytia, Mr Ramon"	1st	71	male	0					
"Astor, Colonel John Jacob"	1st	47	male	0					
"Astor, Mrs John Jacob (Madeleine Talmadge Force)"	1st	19	female	1					
"Aubert, Mrs Leontine Pauline"	1st	NA	female	1					
"Barkworth, Mr Algernon H"	1st	NA	male	1					
"Baumann, Mr John D"	1st	NA	male	0					
"Baxter, Mrs James (Helene DeLauniere Chaput)"	1st	50	female	1					
"Baxter, Mr Quigg Edmond"	1st	24	male	0					
"Beattie, Mr Thomson"	1st	36	male	0					
"Beckwith, Mr Richard Leonard"	1st	37	male	1					
"Beckwith, Mrs Richard Leonard (Sallie Monypeny)"	1st	47	female	1					
"Behr, Mr Karl Howell"	1st	26	male	1					

Image of original dataset

There are several hints that you may notice as you look at this file. Passengers with longer name seem to be misaligned. Everything is left justified. It looks like when things do line up it is columns eight characters apart.

So let's try reading it in as a tab delimited file.

Reading in the Titanic data set

```
fn <-  
"http://www.statsci.org/data/general/titanic.txt"  
ti <- read.table(  
  file=fn, header=TRUE,  
  stringsAsFactors=FALSE, sep="\t")  
head(ti, n=4)  
##  
Name PClass Age    Sex Survived  
## 1              Allen, Miss Elisabeth  
Walton    1st  29 female         1  
## 2              Allison, Miss Helen  
Lorraine  1st   2 female         0  
## 3              Allison, Mr Hudson Joshua  
Creighton 1st  30  male         0  
## 4 Allison, Mrs Hudson JC (Bessie Waldo  
Daniels)  1st  25 female         0
```

It reads in fairly well, it seems. The first few lines of data are hard to read because the passenger names are so long.

Displaying without the names

```
head(ti[, -1])  
##      PClass   Age    Sex Survived  
## 1      1st 29.00 female         1  
## 2      1st  2.00 female         0  
## 3      1st 30.00   male         0  
## 4      1st 25.00 female         0  
## 5      1st  0.92   male         1  
## 6      1st 47.00   male         1
```

Notice how much nicer things look when you take out the column of names.

Peek at the bottom also

```
tail(ti[ , -1])  
##      PClass Age    Sex Survived  
## 1308    3rd  NA female         0  
## 1309    3rd  27   male         0  
## 1310    3rd  26   male         0  
## 1311    3rd  22   male         0  
## 1312    3rd  24   male         0  
## 1313    3rd  29   male         0
```

It's also a good idea to peek at the bottom of a data set right after you read it in. Sometimes a line in the middle gets out of sync and it throws the rest of the data off. If that happens then you will see weird things at the bottom of the data.

Notice that Miss Tamini Zabour has an "NA" for age. We'll take a careful look at this.

Descriptive statistics on age

```
summary(ti$Age)
##      Min. 1st Qu.  Median    Mean 3rd Qu.
Max.      NA's
##      0.17  21.00   28.00   30.40   39.00
71.00      557
```

You learned how to get descriptive statistics for continuous variables in a previous module. The youngest passenger is 0.17 years old and the oldest is 71 years old. Let's peek at their data values.

Print out information on youngest and oldest

```
ti[which.min(ti$Age), ]
##
PClass Age Sex Survived Name
## 764 Dean, Miss Elizabeth Gladys (Millvena)
3rd 0.17 female 1
ti[which.max(ti$Age), ]
##
Survived Name PClass Age Sex
## 10 Artagaveytia, Mr Ramon 1st 71 male
0
```

The `which.min` and `which.max` functions simplify the task of finding the row or rows where a data value reaches its minimum or maximum.

Notice that the little baby survived, but not the old man.

First break

- What have you learned
 - Reading in the Titanic dataset
- What's next
 - Categorical versus continuous variables
 - Counts, proportions, and percentage
 - Factors
 - Crosstabulations
 - Barplots
 - New categorical variables

Categorical versus continuous variables

- Categorical
 - Small number of possible values
 - Each value has a name or label
- Continuous
 - Large number of possible values
 - Potentially any value in a range.

A categorical variable is defined (loosely) as a variable that has a small number of possible values. Each value is usually associated with a particular category or label. In contrast, a continuous variable is defined as a variable that has a large number of possible values, potentially any value in a particular interval.

In a previous module, almost all of the variables that you used were continuous. Today, almost all of the variables that you will use will be categorical.

The distinction between continuous and categorical variables is important in deciding what types of descriptive and inferential statistics you should use. But, there is often gray and fuzzy line between categorical and continuous variables. Don't worry too much about this today. If you're not sure whether a variable is categorical or continuous, try some simple descriptive statistics and graphs appropriate for categorical data and then try some simple descriptive statistics and graphs for continuous data. You will usually figure out quickly whether treating your variable as categorical or continuous makes the most sense.

There are other types of variables also, such as count variables, that have their own special features.

Frequency counts

```
PClass_counts <- table(ti$PClass)
PClass_counts
##
## 1st 2nd 3rd
## 322 280 711
```

For categorical variables, you should first get frequency counts. A mean and standard deviation are usually meaningless for categorical data.

Unlike most other statistical packages, R tends to have a minimalist approach to statistics. If you asked for frequency counts in SAS or SPSS, these systems would automatically add percentages. R doesn't add percentages automatically.

This is something that you will either love or hate. You might think that SAS and SPSS are more thoughtful because almost every time you want a count, you'd also want the corresponding percentage. Or you might find it annoying to tell those programs to not clutter up your output with information you didn't want.

Personally, I don't like software deciding for me what I want. I'd rather ask for percentages explicitly when I need them rather than have them come as the default.

Now this is a rather trivial issue, but it does illustrate an important difference in philosophy. R makes you ask for the extras that you might need. SAS and SPSS force you to ask to NOT include things that they think are important.

So the table function produces only counts.

Proportions

```
PClass_proportions <- prop.table(PClass_counts)
PClass_proportions
##
##      1st      2nd      3rd
## 0.2452399 0.2132521 0.5415080
```

If you want percentages in addition to counts, there are several approaches.

The `prop.table` function takes a frequency table and converts it to a proportion.

Percentages

```
PClass_percents <- round(100*PClass_proportions)
PClass_percents
##
## 1st 2nd 3rd
##  25  21  54
```

Multiply by 100 and round to get percentages.

Percentages

```
pct.sign <- "%"  
PClass_nice_percents <- paste(PClass_percents,  
pct.sign)  
PClass_nice_percents  
## [1] "25 %" "21 %" "54 %"
```

You can get even fancier. The paste function concatenates several string variables and if something is not a string, R will convert it to a string before concatenating.

Percentages

```
PClass_nicer_percents <- paste(PClass_percents,  
pct.sign, sep="")  
PClass_nicer_percents  
## [1] "25%" "21%" "54%"
```

The “sep” parameter tells R what to put between the two values you are pasting together. By default it is a single blank character, but we don’t want a blank between the number and the percent sign.

Percentages

```
colon <- ": "  
PClass_nicest_percents <-  
  paste(  
    names(PClass_percents),  
    colon,  
    PClass_percents,  
    pct.sign, sep="")  
PClass_nicest_percents  
## [1] "1st: 25%" "2nd: 21%" "3rd: 54%"
```

Somehow, the names of the passenger class got lost. So let's add them back in. A colon and a space between the names and the numbers helps as well.

Fractions and percentages

```
n <- sum(PClass_counts)
slash <- "/"
comma <- ", "
percents_and_fractions <- paste(
  PClass_nicest_percents, comma,
  PClass_counts, slash, n,
  sep="")
percents_and_fractions
## [1] "1st: 25%, 322/1313" "2nd: 21%, 280/1313"
"3rd: 54%, 711/1313"
```

Finally, you can combine the counts and the total with the percents to make things look really nice.

This shows how I like to program in R. First, get something simple. Then slowly add layers to it until you get a nice polished product.

In contrast, most other statistical packages try to produce polished results right from the start. This has some advantages, but the approach used by R, where many of the functions do something simple and basic, allows you to polish the results the way you want them.

Tables that include counts of missing values.

```
table(ti$PClass, useNA="always")  
##  
## 1st  2nd  3rd <NA>  
## 322  280  711    0
```

The default option in the table function is to not list missing values. During the initial data screening, you should always look for missing values. This is done with the useNA parameter.

Total count

```
addmargins(table(ti$PClass, useNA="always"))  
##  
## 1st 2nd 3rd <NA> Sum  
## 322 280 711 0 1313
```

You can add a total count to the frequency table with the `addmargins` function.

Second break

- What have you learned
 - Reading in the Titanic dataset
 - Categorical versus continuous variables
 - Counts, proportions, and percentage
- What's next
 - Factors
 - Crosstabulations
 - Barplots
 - New categorical variables

Factors

```
ti$urv_factor <- factor(ti$Survived, levels=0:1,  
labels=c("No", "Yes"))  
table(ti$urv_factor)  
##  
##   No  Yes  
## 863 450
```

The factor function is similar to the variable label in SPSS and the format statement in SAS. It assigns category names (No and Yes) to numeric codes (0 and 1). It also has other features that are helpful in linear and logistic regression models, such as the ability to specify contrasts.

Crosstabulation

```
pclass_by_gender <- table(ti$PClass, ti$Sex)
pclass_by_gender
##
##      female male
## 1st      143  179
## 2nd      107  173
## 3rd      212  499
```

A crosstabulation shows counts across the combination of two different categorical variables. This table shows that there are 143 female first class passengers, 179 male first class passengers, 107 female second class passengers, etc.

Row proportions

```
row_proportions <- prop.table(pclass_by_gender,  
margin=1)  
row_proportions  
##  
##           female           male  
## 1st 0.4440994 0.5559006  
## 2nd 0.3821429 0.6178571  
## 3rd 0.2981716 0.7018284
```

The `prop.table` function converts counts into proportions. Here is an example of row proportions.

Row percentages

```
row_pcts <- round(100*row_proportions)
row_pcts
##
##      female male
## 1st      44   56
## 2nd      38   62
## 3rd      30   70
```

It is a lot easier to read if you convert the proportions into percentages and then round. In first class, 44% of the passengers are female. In second class, 38% are female, and so forth.

Column percentages

```
col_proportions <- prop.table(pclass_by_gender,  
margin=2)  
col_pcts <- round(100*col_proportions)  
col_pcts  
##  
##      female male  
## 1st      31   21  
## 2nd      23   20  
## 3rd      46   59
```

Setting the margin parameter to 2 gives column proportions. Column proportions add up to 1 within each column. Again, converting to percents and rounding helps a lot. Among the female passengers, 31% traveled in first class, 23% in second class, and 46% in third class.

Column percentages

```
cell_proportions <- prop.table(pclass_by_gender)
cell_pcts <- round(100*cell_proportions)
cell_pcts
##
##      female male
## 1st      11   14
## 2nd       8   13
## 3rd      16   38
```

If you omit the margin parameter, R produces cell proportions. Cell proportions add up to 1 across all the cells in the table. Converting to percents, you can see that 11% of the passengers were first class females, 14% were first class males, 8% were second class females, etc.

Which percentages should you use

- General guidance
 - Set the rows to your treatment/exposure
 - Set the columns to your outcome
 - Compute row percentages
- why not try several formats
 - Revised your tables as often as you revise your writing

It's beyond the scope of this class, but with crosstabulations, you have choices as to what should be the rows and what should be the columns. Then you can compute row, column, or cell percentages.

I've found that nine times out of ten, the best choice depends on what is your treatment variable and what is your outcome. It usually works best if you place the treatment variable in the rows and the outcome in the columns and compute row percentages. That shows how often you see a particular outcome in the treatment group and the percentage in the control group is right beneath it.

That being said, I would also encourage you to try several different approaches.

Practical example

```
gender_by_survival <- table(ti$Sex,
ti$urv_factor)
survival_proportions <-
prop.table(gender_by_survival, margin=1)
survival_percents <-
round(100*survival_proportions)
survival_percents
##
##           No Yes
##   female  33  67
##   male   83  17
```

The key question in the Titanic dataset is survival. Who lived and who died. Kate Winslet lived, but sad to say, Leonardo diCaprio didn't. So what was the survival rate for all women? For all men?

Using the rules shown on the previous slide, survival is obviously the outcome. Make that the columns and gender the rows. Then compute row percentages (margin=1).

For this, you can see that only 0.6666667% of the women survived, but 0.1668625% of the men did.

Third break

- What have you learned
 - Reading in the Titanic dataset
 - Categorical versus continuous variables
 - Counts, proportions, and percentage
 - Factors
 - Crosstabulations
- What's next
 - Barplots
 - New categorical variables

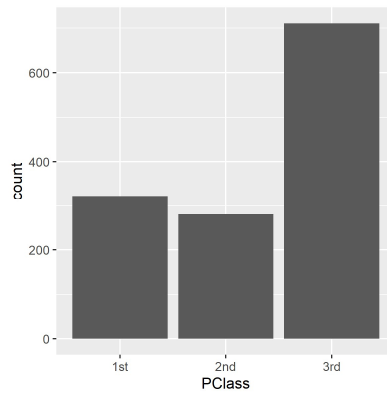
Barplot of counts, code only

```
library(ggplot2)
simple_barplot <-
  ggplot(ti, aes(x=PClass)) +
  geom_bar()
ggsave("../images/barplot1.png", simple_barplot,
width=4, height=4)
```

I'm not a big fan of bar plots, but they sometimes have their uses. You can get a barplot for the frequency count.

Because of the way Rmarkdown displays graphs, I have to put the code on a separate slide from the graph.

Barplot of counts, graph only



Barplot of counts

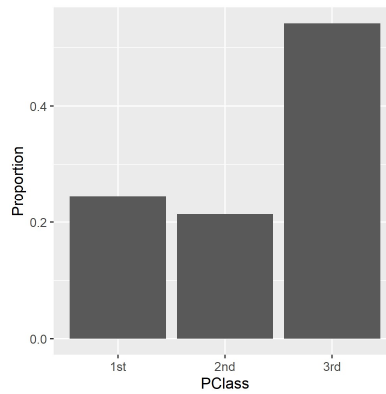
Here is what the plot looks like.

Barplot of proportions, code only

```
proportions_barplot <-  
  ggplot(ti, aes(x=PClass)) +  
    geom_bar(aes(weight=1/n)) +  
    scale_y_continuous("Proportion")  
ggsave("../images/barplot2.png",  
proportions_barplot, width=4, height=4)
```

Use weights to get a bar chart of proportions.

Barplot of proportions, graph only



Barplot of proportions

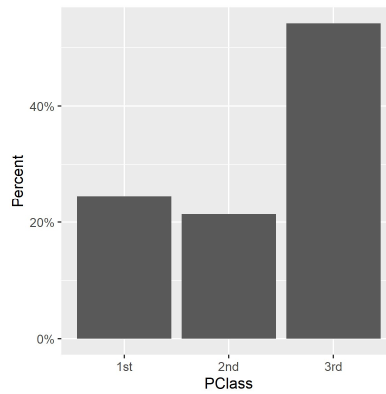
Here is what the plot looks like.

Barplot of percentages, code only

```
proportions_barplot <-  
  ggplot(ti, aes(x=PClass)) +  
    geom_bar(aes(weight=100/n)) +  
    scale_y_continuous("Percent", breaks=c(0, 20,  
40), labels=c("0%", "20%", "40%"))  
ggsave("../images/barplot3.png",  
proportions_barplot, width=4, height=4)
```

Use different weights to get a bar chart of proportions.

Barplot of counts, graph



Barplot of counts

Here is what the plot looks like.

Fourth break

- What have you learned
 - Reading in the Titanic dataset
 - Categorical versus continuous variables
 - Counts, proportions, and percentage
 - Factors
 - Crosstabulations
 - Barplots
- What's next
 - New categorical variables

New categorical variables

```
ti$child <- as.numeric(ti$Age<18)
table(ti$child, useNA="always")
##
##      0      1 <NA>
## 660    96   557
```

There are several ways to create a new categorical variable in R, but the easiest way is through the use of logical comparisons.

Suppose you wanted to compute a categorical variable called “child” which is equal to 1 if a passenger is less than 18 years old and 0 if a passenger is 18 or older. Here’s one way to do it.

Quality check

```
range(ti[which(ti$child==1), "Age"])  
## [1] 0.17 17.00  
range(ti[which(ti$child==0), "Age"])  
## [1] 18 71
```

Anytime you create a new categorical variable, you should run a quality check to make sure nothing is wrong. The range function is useful for this.

Quality check

```
chk <- table(ti$Age, ti$child, useNA="always")
head(chk)
##
##           0 1 <NA>
##  0.17 0 1      0
##  0.33 0 1      0
##  0.8   0 1      0
##  0.83 0 2      0
##  0.92 0 1      0
##    1   0 5      0
```

An alternative is to run a crosstabulation. I am only showing the first few rows, but if you print the entire crosstabulation, you will see the switch over from child=1 to child=0 at 18 years of age.

More on recoding

```
ti$lower.class <- as.numeric(ti$PClass=="2nd" |
ti$PClass=="3rd")
ti$lower.class <- factor(ti$lower.class,
levels=c(0,1), labels=c("No", "Yes"))
table(ti$PClass, ti$lower.class)

##
##           No Yes
## 1st 322    0
## 2nd   0 280
## 3rd   0 711
```

Sometimes you want to combine two or more of the categories together. Here is how you might create a new passenger class variable with values of 1 for second and third class and 0 for first class.

The vertical bar (|) is the R symbol for logical OR.

As before, it's a good idea to run a crosstabulation to verify that the new coding was done properly.

Creating multi-level categories

```
cut_points <- c(0, 1.5, 12, 19, 99)
ti$age_groups <- cut(ti$Age, breaks=cut_points)
table(ti$age_groups, useNA="always")
##
## (0,1.5] (1.5,12] (12,19] (19,99] <NA>
##      13      47      89     607     557
```

You can create multi-level categories using a complex logic system, but you can do it simpler and faster with the cut function.

If you are familiar with math notation, the square bracket means that the value on the boundary belongs in the set, and a parentheses means that it does not belong in the set. So (1.5, 12] means don't include the 1.5 in this set, but do include the 12.

Creating multi-level categories

```
lb <- c("Infant", "Child", "Teenager", "Adult")
ti$age_groups <- cut(ti$Age, breaks=cut_points,
  labels=lb)
table(ti$age_groups, useNA="always")
##
##   Infant   Child Teenager   Adult   <NA>
##      13      47       89     607     557
```

It might be better to include labels to describe the age groups.

Quality check

```
range(ti$Age[which(ti$age_groups=="Infant")])  
## [1] 0.17 1.50  
range(ti$Age[which(ti$age_groups=="Child")])  
## [1] 2 12  
range(ti$Age[which(ti$age_groups=="Teenager")])  
## [1] 13 19  
range(ti$Age[which(ti$age_groups=="Adult")])  
## [1] 20 71
```

Summary

- What have you learned
 - Reading in the Titanic dataset
 - Categorical versus continuous variables
 - Counts, proportions, and percentage
 - Factors
 - Crosstabulations
 - Barplots
 - New categorical variables