University of North Carolina
Chapel Hill

Soci252 – Data Analysis

Professor François Nielsen

Using R with the OLI Online Statistics Course

Last modified March 18, 2017

Contents

12 R Bugs and Problems with OLI Probability and Statistics **47**

THIS DOCUMENT is a short introduction to the use of R for the OLI curriculum in Probability and Statistics. Sections are arranged roughly in the order in which they appear in the curriculum, except that instructions corresponding to both the descriptive and inferential aspects of a particular type of analysis are grouped together. The introduction is followed by an Appendix with specific guidelines on how to fix known bugs in R instructions given in OLI.

1 Installing the R Program

For statistical work we will use the powerful program R that is freely downloadable. You should install R on the laptop that you bring to class as early as possible at the beginning of the class. To install the program go to the R site at http://cran.r-project.org/. Click on the appropriate link and follow the instructions to download the version of R for your Windows, Mac (OS X) or Linux system.

In the course of installation you will be requested to choose a mirror site. Choose a U.S. site geographically close to here, probably the MD site which is in Bethesda, MD.

The installation program will automatically create an icon on your desktop. (If you have a 64 bits system there will be two icons, one for 32 bits version and one for the 64 bits version. You can use either.)

I recommend that you also install John Fox's `car` package.

Windows users note: Before installing any package you should first close R (if it is already running) then restart R in "administrator mode" by right-clicking on the desktop icon and selecting Run as administrator on the local menu.

To install the car package start R and enter the following at the prompt:

```
> install.packages("car", dependencies=TRUE)
```

Note that you can simply select and copy the command above (without the prompt) with Ctrl-C and enter it in R at the prompt (>) with Ctrl-V and press Enter.

You can later access the `car` package with the command

```
> library(car)
```

You can later install other packages using the menus (Packages -> Install package(s) ...). If you are using Windows remember to run R as administrator first.

2 Interacting with R

Graphical user interface (GUI) When you open R what you see is the graphical user interface or GUI (Figure 1). It is rather Spartan, with a bunch of boilerplate text followed by the prompt ">". The prompt means "What now?". It is the way R tells you that it expects you to type an instruction. Let's begin by doing some calculations.

R as a super calculator First try this

```
> 2+2
[1] 4
> 2*3
[1] 6
```

It is reassuring to know that R knows that 2+2=4, and that it can multiply. It can do more complicated, compound calculations.

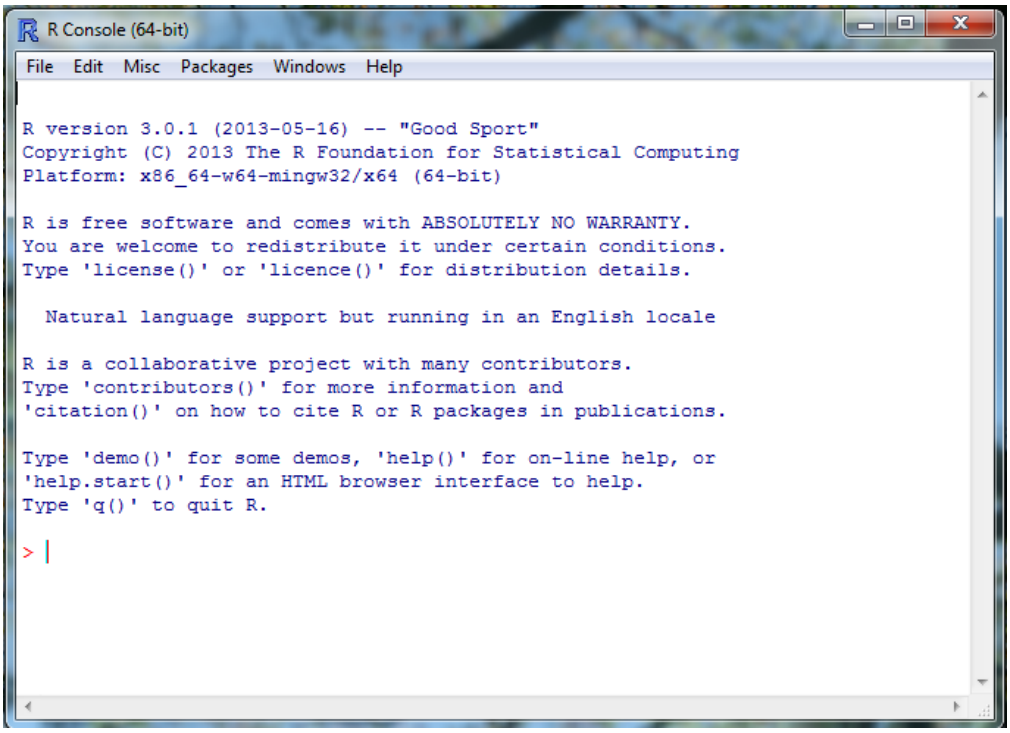

Figure 1: The R GUI

Table 1: Mathematical (scalar) functions.

Function	Description
abs(x)	Absolute value
sqrt(x)	Square root
round(x, digits=n)	Round x to n decimal places
log(x)	Natural logarithm (base $e = 2.718282$)
log10(x)	Common logarithm (base 10)

```
> (2+3)^2
[1] 25
```

R uses a standard order of arithmetic operations. If in doubt you can specify the desired order of operations using parentheses; R will perform operations within the parentheses first. R also has functions for various mathematical calculations.

```
> sqrt(2)
[1] 1.414214
> log10(1000)
[1] 3
```

The mathematical functions you are most likely to encounter in the OLI curriculum are summarized in Table 1.

Creating objects To save the result of a calculation in memory do the following.

```
> x <- (7+5)^2-17
> x
[1] 127
> y <- "my string"
> y
[1] "my string"
```

The composite operator <- is formed on the keyboard by entering the characters < (less than) followed by - (hyphen or minus) with no space in between. <- means that the variable on the left "gets" or "is assigned" the value on the right.[1] When you create a variable R does not automatically print the value. To print the value (on the screen, that is) you need to enter the name of the variable. Note that you can save variables that are textual, by entering the value between quotes.

Variables of all kinds, sizes and shapes are called "objects" in R. You will be handling objects in R all the time.

Vectors A very important kind of objects in R are vectors. These are combinations of elements all of the same type, e.g. all of them numbers.

You can create a vector using the `c()` function.

```
> v <- c(17, 2, 14, 5, 11, 8)
> v
[1] 17  2 14  5 11  8
> sqrt(v)
[1] 4.123106 1.414214 3.741657 2.236068 3.316625 2.828427
```

In this example I created a vector v with six elements, printed the vector by entering its name, and then used the `sqrt()` function. As you can see, again when creating an object one needs to enter its the name to see its value. Note the extremely important (and powerful) fact that in R *scalar functions apply to vectors element by element*. Here we calculated the square root of each element in the vector in one fell swoop.

One can also create a vector as a range, like so.

```
> w <- 6:17
> w
 [1]  6  7  8  9 10 11 12 13 14 15 16 17
```

There are many statistical functions that apply to an entire numerical vector (as opposed to element-wise), summarized in Table 2. Some examples follow, that use the vector v created earlier.

```
> length(v)
[1] 6
> mean(v)
[1] 9.5
> sd(v)
[1] 5.612486
> median(v)
[1] 9.5
> range(v)
[1]  2 17
```

Note how `range(v)` returns two numbers (the minimum and the maximum), not just one.

[1] In most cases you can also use "=" instead of "<-" (as the R instructions in the R curriculum do). But using <- is the most common practice among R users.

Table 2: Statistical functions.

Function	Description
length(v)	Number of elements
mean(v)	Mean
median(v)	Median
sd(v)	Standard deviation
var(v)	Variance
mad(v)	Median absolute deviation
sum(v)	Sum
min(v)	Minimum
max(v)	Minimum
range(v)	Range (Min, Max)

More on entering commands into R A short R command suggested in the OLI course can be copied by highlighting it, copying it with Ctrl-C, and entering it in R at the > prompt with Ctrl-V or by right-clicking at the prompt and selecting Paste from the local menu.

You can also copy an R command from a document, such as a pdf file, that includes the R prompt (>). In fact, you can highlight and copy an entire block of text that includes commands with R prompts, and even the resulting output. In such cases to enter the commands in R you can right-click at the R prompt and select Paste commands only. This trick works in many cases; it is useful to enter examples of R commands provided in documentation. However Paste commands only is not available to paste into the script editor described below.

In some cases you will find that the command you copied from OLI does not run. This is often for the good reason that the command provided is in a generic form that you have to edit by specifying the actual names of the variables. In such cases I find that the simplest method may be to create a script to edit the command, as follows. In the GUI go to the File menu and select New script. R will open a text editor window. Type the desired command in the editor, specifying the variable names as needed. To run the edited command highlight it and press Ctrl-R, or go to the Edit menu and select Run line or selection. R will then run the commands as if they had been entered at the prompt.

Opening a new script is also a useful method to run and modify more complex commands provided by OLI. These can consist of a very long line of commands separated by semi colons, or even blocks of commands. In such cases you can highlight and copy the line or block in OLI, open a new script in R, and copy the command(s) in the editor window. You can sometimes run the long line as is, by highlighting it, and then running it with Ctrl-R or by right-clicking the selection and choosing Run line or selection in the local menu. But I often find it useful to edit a long line by breaking it into a set of shorter commands, by placing the cursor right behind the semi colons (;) and pressing Enter, and then highlighting and running the resulting block of lines.

Using this technique you can also easily edit and rerun a block of commands (as you have to do in some of the exercises) by making a copy of an entire block, inserting the copy beneath the original block, editing the copy by changing the variable names, and running the edited block. You can change repeated text such as variable names in a block by placing the cursor on top of the block to be edited and clicking Edit → Replace... and filling the dialog box. With that method you are creating a script that you can even save (File → Save) if you want to reuse it later. You may not have a reason to save a script file for this course, but you certainly will when you do your own research.

3 Dataframes

Dataframes in OLI Much of the computer work for the OLI curriculum is based on data provided through links to files with extension `.RData`. There are several ways to open such a file in R, one of which should work for you.

1. The simplest way, which should work if R is installed locally on your system, is to double-click the link to the `.RData` file. The computer will then open an R session and automatically load the data set. Even though the data are there, all you will see is the R prompt > and nothing else! See below for suggestions on what you can do to make sure the data set is available and look at the data.

2. A variant of (1) is to click on the link to the data set. A dialog box should open asking you to choose Open or Save. Choose Open with R for Windows GUI.

3. A slightly more circuitous way to download the data, which is the one suggested by OLI, is to right click on the link to the data set and choose Save to the Hard Drive (or Save Link As ...). Then find the downloaded file on your system and double-click on it to open the file in R. One advantage of this method is that R will open in the folder where the data file is located, so you will know later where to find any R output such as graphs or data sets that you decide to save.

The OLI curriculum provides the instructions needed to do the statistical work. In some cases these instructions will consist of R commands that you can highlight and copy in the usual way (with Ctrl-C, or by right-clicking the highlighted text and then selecting Copy on the local menu) and then paste directly in the R console at the > prompt (with Ctrl-V or Paste from the local menu).

There are a few bugs in the R instructions provided by OLI. Fixes for bugs of which I know at the time of writing are listed in an Appendix to this document.

An example: The Afifi and Clark data With either method of opening a data set in R you are left staring at the R prompt and you may well wonder what to do. Here we look at a few useful commands you can use to find your way around a data set you just opened, in the context of a real example. Note that in R everything on a line after the pound sign (#) is considered a comment and ignored.

The data set is located at http://www.unc.edu/ nielsen/soci252/software/AfifiClark.RData. Open the data set using one of the methods described above.

Enter the following commands at the prompt one by one and see what happens on your system.

```
> ls()
[1] "AfifiClark"

> names(AfifiClark)
 [1] "id"       "sex"      "age"      "marital"  "educat"   "employ"
 [7] "income"   "relig"    "cesd"     "cases"    "drink"    "health"
[13] "regdoc"   "treat"    "beddays"  "acuteill" "chronill" "educat.f"

> head(AfifiClark)
  id    sex age   marital educat     employ income      relig cesd  cases drink
1  1 female  68   widowed      2    retired      4 Protestant    0 normal    no
2  2   male  58  divorced      4  full time     15 Protestant    4 normal   yes
3  3 female  45   married      3  full time     28 Protestant    4 normal   yes
4  4 female  50  divorced      3 unemployed      9 Protestant    5 normal    no
```

```
5  5 female  33 separated      3  full time     35 Protestant    6 normal   yes
6  6   male  24   married      3  full time     11 Protestant    7 normal   yes
     health regdoc treat beddays acuteill chronill educat.f
1      good    yes   yes      no       no      yes  some HS
2 excellent    yes   yes      no       no      yes some col
3      good    yes   yes      no       no       no  HS grad
4 excellent    yes    no      no       no      yes  HS grad
5 excellent    yes   yes     yes      yes       no  HS grad
6 excellent    yes   yes      no      yes      yes  HS grad

> summary(AfifiClark)
       id              sex           age                marital
 Min.   :  1.00   male  :111   Min.   :18.00   never married: 73
 1st Qu.: 74.25   female:183   1st Qu.:28.00   married      :127
 Median :147.50                Median :42.50   divorced     : 43
 Mean   :147.50                Mean   :44.41   separated    : 13
 3rd Qu.:220.75                3rd Qu.:59.00   widowed      : 38
 Max.   :294.00                Max.   :89.00

     educat             employ        income             relig
 Min.   :1.00   full time  :167   Min.   : 2.00   Protestant:155
 1st Qu.:3.00   part time  : 42   1st Qu.: 9.00   Catholic  : 51
 Median :3.00   unemployed : 14   Median :15.00   Jewish    : 30
 Mean   :3.48   retired    : 38   Mean   :20.57   None      : 56
 3rd Qu.:4.00   houseperson: 27   3rd Qu.:28.00   Other     :  2
 Max.   :7.00   in school  :  2   Max.   :65.00
                other      :  4
...
```

Explanation: The `ls()` lists the objects that R keeps in memory. (You need to enter `ls()` just as shown, with the empty parentheses.) In this case it reveals the name of the data frame that has been read in memory (`AfifiClark`). You can list the contents of the data frame by just entering its name (`AfifiClark`), but this can get overwhelming when the data frame has thousands of cases. Then it is better to use `head()` with the name of the data frame, which lists the first 6 cases by default. You can specify a different number of cases to be listed by specifying the number you want after the name of the file, as in `head(AfifiClark, 10)` if you wanted the first 10 cases. (Likewise, you can use `tail()` to list the last 6 cases by default.) Use `names()` to show the names of the variables in the data frame. `summary()` summarizes every variable in the data set. (Part of that output was omitted to save space.) Note that `summary()` provides different statistics depending on the type of the variable: minimum, maximum, mean, etc. for numeric variables, and numbers of observations in the different categories for categorical variables (which R calls "factors").

The next commands will create a graph in a separate window, which may be hidden behind other windows (so you may have to retrieve it). The graph we want is a histogram of variable `cesd`, which is a depression score (the higher the score, the more symptoms of depression the person has).

```
> hist(cesd)   # doesn't work
Error in hist(cesd) : object 'cesd' not found
> attach(AfifiClark)
> hist(cesd)   # now R "sees" variable cesd
```

Explanation: The first command doesn't work because R does not "see" the variable `income` within the data frame AfifiClark. So we tell R to look for `income` within AfifiClark with the

Table 3: Dataframe-related functions.

Function	Description
> ls()	lists objects in memory (to reveal name of loaded dataframe)
> names(df)	lists names of variables in dataframe df
> head(df)	lists first 6 records (cases) of dataframe df
> head(df, n)	same thing for first n records
> tail(df), tail(df, n)	lists last 6 (last n) records
> summary(df)	summary statistics for all the variables in dataframe df
> getwd()	find out what work directory (folder) you are in

command `attach(AfifiClark)`. We can now create the histograms we want. Note how skewed the distribution of cesd is: most people have few symptoms of depression, but some are seriously affected.

Next we want to create a pie chart of religious denominations. (Output is not shown.)

```
> tab <- table(relig)
> tab
> pct <- round(100*tab/sum(tab), 1)
> pct
> lab <- paste(names(pct), pct, sep=" ")
> pie(pct, labels=lab)
```

Explanations: The commands above show how to create a labeled pie chart of religious denominations that shows the percentages. Note that we use the "<-" assignment notation to create a table object named tab.

Summary of dataframe-related functions Useful functions related to dataframes are listed in Table 3.

Using an R script to fix or modify commands Next we illustrate how to create a script with the text editor in R. This is very useful to modify commands suggested by OLI. OLI Learn by Doing and StatTutor labs often suggest sequences of commands that, once copies and pasted to the command prompt, form long lines separated by semicolons, like this (I have split one long line into two so it fits on the page).

```
tab <- table(relig); pct <- round(100*tab/sum(tab), 1);
  lab <- paste(names(pct), pct, sep=" "); pie(pct, labels=lab)
```

These long lines of commands are awkward to work with. More importantly, they are difficult to modify to change a variable name or fix a problem in the commands provided by OLI.

The solution is to create an R script (also called an R Document in the Mac GUI). Using the menu create a new R script with File -> New script. A blank window opens. Copy the line(s) above in the new window. Now enter a new line (press Enter) after each semicolon. Optionally, you can then delete the semicolons. Your script should now look like this.

```
tab <- table(relig)
pct <- round(100*tab/sum(tab), 1)
lab <- paste(names(pct), pct, sep=" ")
pie(pct, labels=lab)
```

Now run this script by highlighting the block of commands and running it, from the Edit menu or with Ctrl-R. It should create the same pie chart as we had before.

Now for the payoff of doing this. We also want a pie chart of educational attainment, which is variable `educat.f`. To do this we just make a copy of the block of commands already in the script window and insert it at the end the script. Then we change `relig` in the first line of the new block to `educat.f`. Highlight the new block and run it by entering Ctrl-R or using the menu (Edit -> Run line or selection). You should now see a pie chart for educational attainment.

Checking our preconceptions with R Which religious denomination do you think is most prone to depression? Just for fun, let's look at side-by-side boxplots of depression score by religious denomination, with notches to give us a sense of which differences might be statistically significant.

```
> plot(cesd ~ relig, col=terrain.colors(5), notch=TRUE)
```

Does the graph confirm our theory?

The importance of "detachment" When you are done with a data frame it is good practice to "detach" the data frame with the following, unless you are going to close the R session anyway.

```
> detach()
```

This is to prevent problems with other dataframes you may later attach during the same session, if they contain variables with the same name as the currently attached one.

4 Distribution – One Categorical Variable

4.1 Two Categories

We look at the following example. A survey of the Middle East region in 2012 asked respondents whether they favored or opposed Assad remaining President of Syria. The sample included 98 respondents in Syria; among these 54 said they were in favor of Assad remaining president.

Descriptive Analysis and Inference

In R descriptive analysis and inference for a single proportion are combined in the function `prop.test()`. There are two principal questions of interest here: (1) Finding a point estimate and a confidence interval of support for Assad; (2) Test whether Assad has the support of a majority (i.e., more than 50%) of the Syrian population.

The first question is answered using the template

```
prop.test(x, n)
```

where x is the count of successes and n is the count of trials. In the Assad example we use `prop.test()` as follows.

```
> prop.test(54, 98)

        1-sample proportions test with continuity correction

data:  54 out of 98, null probability 0.5
X-squared = 0.8265, df = 1, p-value = 0.3633
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
```

```
 0.4474730 0.6505232
sample estimates:
        p
0.5510204
```

We see that the point estimate is $\hat{p} = .5510204$, so 55.1% say they support Assad, and the CI is (0.4474730, 0.6505232), so we are 95% confident that Assad support lies between 44.7% and 65.1%. If we wanted another confidence level than 95% we can specify the option `conf.level = 0.95` with a probability other than 0.95. For this use we can ignore the rest of the output.

To test a hypothesis we use

```
prop.test(x, n, p = <null hyp.>, alt = c("two.sided", "less", "greater"))
```

where p is the value specified in the null hypothesis, and alt specifies the direction of the hypothesis. In the Assad example the setup is $H_0 : p = .5, H_a : p > .5$ so we specify `alt="greater"`.

```
> prop.test(54, 98, p=.5, alt="greater")

        1-sample proportions test with continuity correction

data:  54 out of 98, null probability 0.5
X-squared = 0.8265, df = 1, p-value = 0.1816
alternative hypothesis: true p is greater than 0.5
95 percent confidence interval:
 0.4630677 1.0000000
sample estimates:
        p
0.5510204

> # alt can be "two.sided", "less" or "greater"
```

The test gives us a p-value of 0.1816, which is not small. We conclude that we cannot reject the null hypothesis that support for Assad is 50% (or less), in other words that Assad does not have a majority of support.

Notes: (1) prop.test() uses a chi-squared value (`X-squared = 0.8265` in the printout) as a test statistic. The z-statistic can be calculated as the square root of `X-squared`, with the same sign as $(\hat{p} - p_0)$. (2) Inferential results from `prop.test()` differ slightly from those obtained by hand because the function by default applies a correction for continuity called Yates' correction. To avoid this use the `correct = FALSE` option.

4.2 More than Two Categories

We look at the distribution of religious affiliation in the AfifiClark data.

Descriptive Analysis

To describe the distribution of a single categorical variables in numbers from individual data, we use the function `table()` to create a table and then `prop.table()` to calculate the proportions in each category, as in the following.

```
> tab <- table(relig)
> tab
relig
Protestant   Catholic     Jewish       None      Other
       155         51         30         56          2
```

```
> prop.table(tab)
relig
 Protestant    Catholic      Jewish        None       Other
0.527210884 0.173469388 0.102040816 0.190476190 0.006802721
```

The following script shows four graphic representations of a single categorical variable (Figure 2).

```
> pie(tab)
> pct <- round(100*prop.table(tab), 1)
> pct
relig
Protestant   Catholic     Jewish       None      Other
      52.7       17.3       10.2       19.0        0.7
> lbls <- paste(names(pct), " ", pct, "%", sep="")
> lbls
[1] "Protestant 52.7%" "Catholic 17.3%"   "Jewish 10.2%"     "None 19%"
[5] "Other 0.7%"
> pie(pct, labels=lbls)
> barplot(pct)
> dotchart(pct)
Warning message:
In dotchart(pct) :
  'x' is neither a vector nor a matrix: using as.numeric(x)
> layout(matrix(1:1, ncol=1))
```

The simple pie chart is made simply as `pie(tab)`. For the more informative version I first calculated percentages rounded to one decimal and then pasted together the category names and the percentages to create labels, which are then used with the `pie()` function. The bar plot and dot chart ae made directly from the percentages as `barplot()` and `dotchart()`, respectively.

Inference

We can use a chi-squared test to test whether a sample might come from a population with a known distribution. We continue the religious affiliation example to illustrate this. In 2007 PEW estimated religious affiliation for the entire U.S. population. We want to test whether the distribution of religious affiliations in the AfifiClark data, that were collected in a Southern California community, could be the same as that found by PEW for the nation as a whole. We proceed as follows.

```
> pew07 <- c(.513, .239, .017, .161, .07)
> # Note Jewish includes other non Christian
> sum(pew07)
[1] 1
> chisq.test(tab, p=pew07)

        Chi-squared test for given probabilities

data:  tab
X-squared = 148.8292, df = 4, p-value < 2.2e-16

Warning message:
In chisq.test(tab, p = pew07) : Chi-squared approximation may be incorrect
```

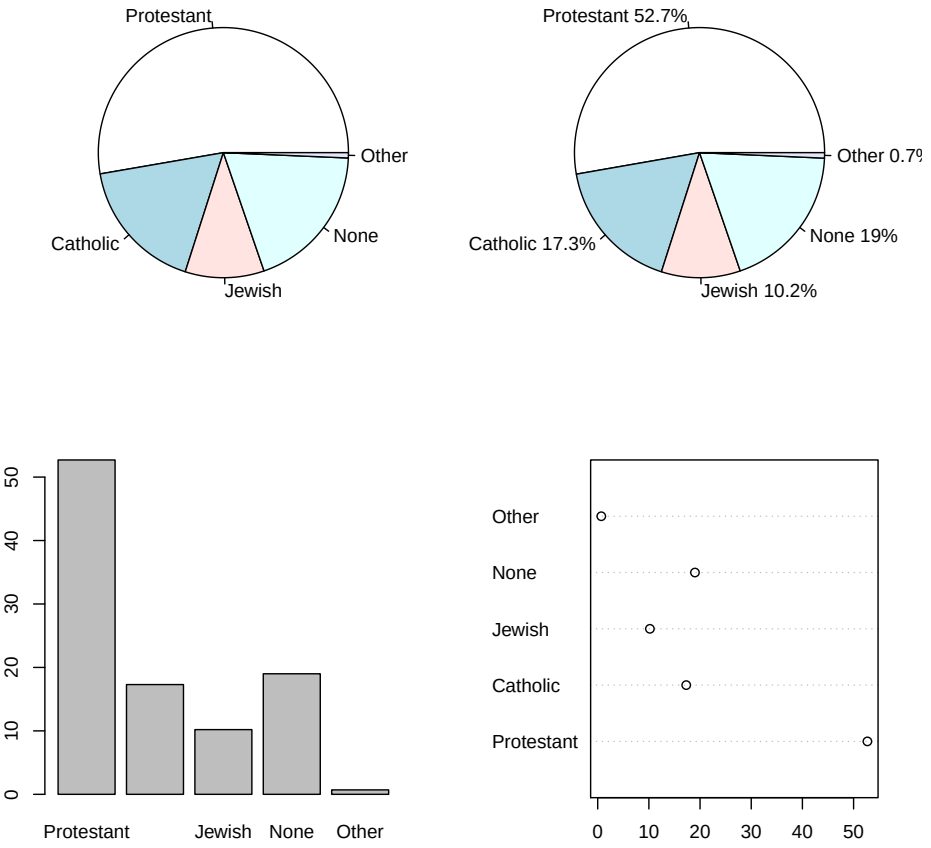

Figure 2: Four graphic representations of religious affiliation. (Clockwise form top left.) A simple pie chart. A more informative pie chart showing percentages. A dot chart (aka Cleveland's dot plot). A bar chart.

The p-value is essentially zero, suggesting that we reject the hypothesis that the distributions of religious affiliation in the AfifiClark data and for the U.S. as a whole are the same. But R gives a warning that assumptions for the use of chi-squared may not be met. To make sure we rerun `chisqu.test()` with the option `simulate.p.value=TRUE`, which uses simulation rather than assuming the chi-squared distribution.

```
> chisq.test(tab, p=pew07, simulate.p.value=TRUE)

        Chi-squared test for given probabilities with simulated p-value (based
        on 2000 replicates)

data:  tab
X-squared = 148.8292, df = NA, p-value = 0.0004998
```

The p-value using simulation is still very small, essentially zero. So we conclude that distribution of affiliations in the AfifiClark data and in the country as a whole are not the same. How do the distributions differ? To find out we calculate the residuals, as follows.

```
> chisq.test(tab, p=pew07, simulate.p.value=TRUE)$residuals
relig
Protestant   Catholic     Jewish       None      Other
 0.3402014 -2.2983647 11.1834712  1.2595983 -4.0956518
```

The residuals are standardized discrepancies between observed and expected counts. A positive residual indicates an excess, and a negative one a deficit, of observed compared to expected. Residuals can be interpreted as z-scores. The largest residual (11.183) is that for Jewish affiliation, suggesting that the proportion of Jewish-affiliated is much higher in the Southern California community where the AfifiClark data were collected than in the country as a whole.

5 Distribution – One Quantitative Variable

5.1 Descriptive Analysis

We look at the distribution of `income` (in thousands of dollars) in the AfifiClark data. I have previously loaded the data and attached the data frame. Here are functions to compute various measures of center and spread. Here the functions work without a glitch, because there are no missing values.

If there were missing values the functions would return `NA` (missing), because R doesn't know what we want to do with the missing observations. In that case we should use the option `na.rm=TRUE`, meaning "remove missing values" and the function will work as intended, e.g. `mean(income, na.rm=TRUE)`.

```
> mean(income)
[1] 20.57483
> median(income)
[1] 15
> sd(income)
[1] 15.29012
> IQR(income)
[1] 19
> length(income)   # number of cases
[1] 294
> summary(income)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

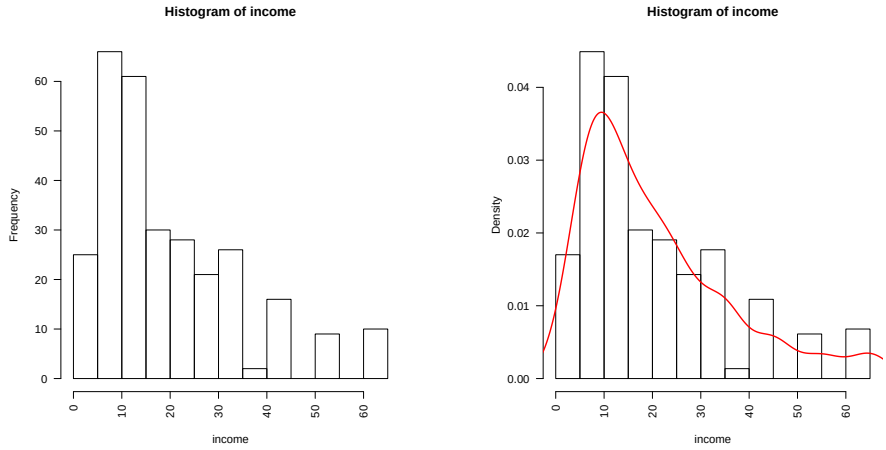

Figure 3: Histograms of income. Simple fequency histogram (left). Density histogram with added density curve (right).

```
   2.00    9.00   15.00   20.57   28.00   65.00
> fivenum(income)
[1]  2  9 15 28 65
```

We note that mean income (20.57483) is greater than the median (15), a clue that the distribution is skewed right. We visualize the distribution of income with two variants of a histogram.

```
> hist(income)
> hist(income, prob=TRUE)
> lines(density(income), col="red", lwd=2)
```

Histograms are shown in Figure 3. The first version (left) is a frequency histogram, with the vertical scale showing the number of observations in each income bin (i.e., interval). We can see that the distribution is indeed right-skewed. In the second histogram (right) the vertical scale measures the density of observations in each bin. Density is the number such that the areas of all the bars, added together, sum up to one. The second histogram adds a density curve on top of the histogram. The option `prob=TRUE`, which switches from the frequency to the density scale is needed, as otherwise the density curve would not show, as it would be completely smooched against the horizontal axis on the frequency scale.

A boxplot is generated as follows.

```
> boxplot(income)
```

The box plot of income is shown in Figure 4. We note what appears to be a single high outlier with an income of 65K. The appearance is misleading, because the point corresponds in fact to 10 observations with the same high income. (You can check that this is so by running `table(income)` at the prompt, which gives you a table of all the values of income in the data with the number of observations for each.)

5.2 Inference

Inference with Individual Data

The principal forms of inference for a single distribution are finding a confidence interval for the mean and testing a hypothesis on the mean. When individual data are available both tasks are performed using the `t.test()` function.

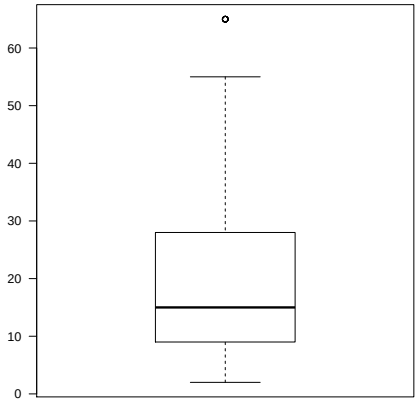

Figure 4: Boxplot of income. The single high outlier at income of 65K is misleading, as the point actually corresponds to 10 stacked observations.

```
> t.test(income)

        One Sample t-test

data:  income
t = 23.0727, df = 293, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 18.81981 22.32985
sample estimates:
mean of x
 20.57483
```

We can ignore the first part of the output, which tests the hypothesis (uninteresting in this case) that mean population income is zero. The estimated CI is (18.81981, 22.32985) which tells us that we can be 95% confident that mean income in the population sampled by Afifi and Clark is between 18,820 and 22,330 dollars. If we wanted a confidence level other than 95% we would explicitly include the option `confi=.95`, changing the default .95 to the desired level.

Suppose that mean income for the country as a whole at the time the data were collected was 15,000 dollars. We want to test the hypothesis that mean income in the population sampled by Afifi and Clark was greater than 15K. The hypothesis setup is $H_0 : \mu = 15; H_a : \mu > 15$. We run `t.test` as follows.

```
> t.test(income, mu=15, alt="greater")

        One Sample t-test

data:  income
t = 6.2516, df = 293, p-value = 7.158e-10
alternative hypothesis: true mean is greater than 15
95 percent confidence interval:
 19.1034     Inf
sample estimates:
```

```
mean of x
 20.57483
```

The p-value is very small, so we reject the null hypothesis that $\mu = 15K$: if it were true the probability of finding a mean income as high as we did (20.57483) would be practically zero. We conclude that population mean income is greater than 15K.

Inference with Summarized Data

When we don't have access to the individual data, we can still carry out inference if we are given three pieces of information: the sample mean, the sample standard deviation (or the variance), and the number of observations. We then use the formulas for the CI and the t-test directly.

Confidence Interval The formula for the CI is

$$\bar{X} \pm t^* SE(\bar{X})$$

where $SE = s/\sqrt{n}$, α is one minus the desired confidence level (of 0.95 in this case), and t^* is the quantile of a t distribution with $n-1$ df for the upper-tail probability of $1-\alpha/2$.

Using R and the values for `xbar`, `s`, and `n` that were calculated earlier, we proceed as follows. (In a realistic example this summarized information would be provided as part of a textbook problem or found in a published study report.)

```
> xbar <- 20.57483; s <- 15.29012; n <- 294
> alpha <- .05
> tstar <- qt(alpha/2, df=n-1, lower.tail=FALSE)  # critical t value
> tstar
[1] 1.968093
> SE <- s/sqrt(n)
> c(xbar - tstar*SE, xbar + tstar*SE)
[1] 18.81981 22.32985
```

This CI is exactly the same as we calculated with `t.test()` using the individual data.

Hypothesis Test The test statistic t for the null hypothesis $H_0 : \mu = \mu_0$ is

$$t = \frac{\bar{X} - \mu_0}{SE(\bar{X})}$$

where $SE = s/\sqrt{n}$. The p-value is calculated as

$$\text{p} - \text{value} = \begin{cases} \text{P}(T_{n-1} \leq t) & \text{for } H_a : \mu < \mu_0 \\ \text{P}(T_{n-1} > t) & \text{for } H_a : \mu > \mu_0 \\ 2 \times \text{P}(T_{n-1} > |t|) & \text{for } H_a : \mu \neq \mu_0 \end{cases}$$

where T_{n-1} refers to a variable distributed as Student t with $(n-1)$ df.

Using R with the income example, we can test the hypothesis that average income is more than 15K (so the alternative hypothesis is $H_a : \mu > \mu_0$) using the previously calculated `xbar`, `s`, and `n` as follows.

```
> xbar <- 20.57483; s <- 15.29012; n <- 294
> mu <- 15
> SE <- s/sqrt(n)
> tstat <- (xbar - mu)/SE
```

```
> tstat                              # test statistic
[1] 6.251646
> pt(tstat, n-1, lower.tail=FALSE)  # p-value
[1] 7.157766e-10
```

The test statistic and p-value are the same as found earlier from the individual data.

Inference When σ is Known When the population standard deviation σ is known (so we don't have to estimate it as the sample standard deviation s) then inference for the sample mean is based on the normal rather than the Student t distribution.

Situations where σ is known are not all that rare: it is the case, for example, for *normed tests,* such as IQ tests, whose distributions have been studied based on large representative samples. The following example is typical.

A professor teaching an undergraduate Data Analysis course administers a standard test of understanding of statistical concepts to his students at semester's end. The test was normed based on a large sample of undergraduates enrolled in college-level first courses in statistics at institutions in the United States. Nationally the test has a mean score (percent correct answers) of 55.770 and standard deviation of 16.134.

One semester for the $n = 39$ students who took the test the average score was $\bar{x} = 71.0$. The professor would like to know three items: (1) The 95% confidence interval for the class average; (2) Whether the class average of 71.0 is significantly greater than the national average of 55.77; (3) How "big" the score advantage of his class is compared to the national average. The analysis is carried out as follows.

```
> xbar <- 71.0; sigma <- 16.134; n <- 39
> mu <- 55.77
> SD <- sigma/sqrt(n)      # "SD" not "SE", since based on sigma
> alpha <- .05
> zstar <- qnorm(alpha/2, lower.tail=FALSE)  # critical z value
> zstar
[1] 1.959964
> c(xbar - zstar*SD, xbar + zstar*SD)        # 95pct CI
[1] 65.93642 76.06358
>
> zstat <- (xbar - mu)/SD
> zstat                                      # test statistic
[1] 5.895086
> pnorm(zstat, lower.tail=FALSE)             # p-value
[1] 1.87243e-09
>
> cohen <- (xbar - mu)/sigma
> cohen                                      # Cohen's d
[1] 0.9439693
```

The professor can be confident at the 95% level that the class average score is between 65.9% and 76.1%. The large z-statistic (5.9) and tiny p-value (≈ 0) indicate that the null hypothesis that the class score is the same as the national average can be rejected. The Cohen's d indicates that the class score is almost one standard deviation higher than the national average. The professor is happy, as he likes to think that the good performance of the class relative to the national average is a reflection of his superior teaching.

Functions from the BSDA Package The package BSDA by Alan T. Arnholt provides functions for tests with summarized data that are not available from base R.

- `tsum.test()` provides CIs and hypothesis tests like `t.test()` but based on summarized rather than individual data. For the `income` example with the AfifiClark data `tsum.test` is called as follows.

```
> xbar <- 20.57483; s <- 15.29012; n <- 294; mu <- 15
> tsum.test(mean.x = xbar, s.x = s, n.x = n, mu = mu)
```

(Output not shown.)

- `zsum.test()` provides CIs and hypothesis tests like `t.test()` but based on summarized rather than individual data and on the use of the normal rather than Student *t* distribution. For the statistics exam example `zsum.test()` is called as follows.

```
> xbar <- 71.0; sigma <- 16.134; n <- 39; mu <- 55.77
> zsum.test(mean.x = xbar, sigma.x = sigma, n.x = n, mu = mu)
```

(Output not shown.)

- `z.test()` provides CIs and hypothesis tests for individual data like `t.test()`, but based on the normal rather than Student *t* distribution. It is called as follows.

```
> mu <- 15; sigma <- sd(income)   # here use s as estimate of sigma
> z.test(x = income, mu = mu, sigma.x = sigma)
```

6 Relationship – Categorical -> Quantitative

Analysis of the relationship between a categorical explanatory variable and a quantitative response. The approach depends on the number and nature of the categories of the explanatory variable.

6.1 Two Independent Samples

We look at the difference in income between men and women in the AfifiClark data.

Descriptive Analysis

The descriptive analysis proceeds as follows.

```
> means <- tapply(income, sex, mean)
> SDs <- tapply(income, sex, sd)
> Ns <- tapply(income, sex, length)
> cbind(Mean=means, SD=SDs, N=Ns)
           Mean       SD   N
male   24.10811 16.28683 111
female 18.43169 14.27649 183
> boxplot(income~sex, notch=TRUE)
```

The mean income for males (24.1K) is substantially larger than mean income for females (18.4K). The standard deviation is somewhat larger for males.

The parallel boxplots (Figure 5) shows that the same pattern of larger male than female income holds for medians. I have used the option `notch=TRUE` in the `boxplot()` function to create the notches in the boxplots. The non-overlapping notches indicate that the medians are significantly different at better than the .05 level.

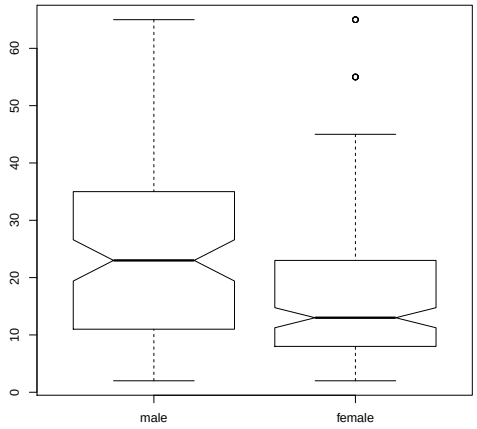

Figure 5: Boxplot of income by sex.

Inference

Testing the null hypothesis that the mean incomes of men and women are the same is done in R using the `t.test()` function. To replicate hand calculations using the pooled estimate of the standard error, I use the option `var.equal=TRUE`.

```
> t.test(income~sex, var.equal=TRUE)

        Two Sample t-test

data:  income by sex
t = 3.1319, df = 292, p-value = 0.001913
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.109297 9.243531
sample estimates:
  mean in group male mean in group female
            24.10811             18.43169
```

In this case the summary statistics showed that the SDs for males and females are similar, so the assumption of equal variances is plausible. In some situations, this assumption is less plausible. Then a variant of the t-test called the Welch test can be used, simply by omitting the `var.equal=TRUE` option, as `var.equal=FALSE` is the default. The result is as follows.

```
> t.test(income~sex)

        Welch Two Sample t-test

data:  income by sex
t = 3.0327, df = 208.99, p-value = 0.002731
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.986457 9.366371
sample estimates:
```

```
  mean in group male mean in group female
           24.10811             18.43169
```

Using either method we see that the p-value is small ($p < .003$ at most), and we conclude by rejecting the null hypothesis that male and female mean incomes are the same.

6.2 Two Paired Samples

We look at a data set consisting of observations on 15 dementia patients in nursing homes. The numbers of disruptive behaviors by a patient were counted and averaged for a three-day period centered around a full moon, and for other days (Moore, David S., George P. McCabe and Bruce A. Craig. 2012. *Introduction to the Practice of Statistics,* 7e. New York: W.H.Freeman and Co.) The data are available at http://www.unc.edu/ nielsen/soci252/software/FullMoon.RData

Descriptive Analysis

We do a simple descriptive analysis as follows.

```
> ls()
[1] "FullMoon"
> head(FullMoon)
  patient aggmoon aggother aggdiff
1       1    3.33     0.27    3.06
2       2    3.67     0.59    3.08
3       3    2.67     0.32    2.35
4       4    3.33     0.19    3.14
5       5    3.33     1.26    2.07
6       6    3.67     0.11    3.56
> attach(FullMoon)
> stem(aggdiff)

  The decimal point is at the |

  -0 | 0
   0 | 11
   1 | 6
   2 | 1347
   3 | 11167
   4 | 44
```

The data frame FullMoon illustrates the typical data arrangement for a paired samples analysis, with the two observations on the same units stored in two different variables (or vectors). `aggdiff` is the precalculated difference for a given patient between average disruptive episodes during moon days (`aggmoon`) and non-moon days (`aggother`). The stem and leaf chart of `aggdiff` shows some skewness but no marked outlier.

Inference

To test the hypothesis that there is no difference in disruptive behavior between moon days and non-moon days we use `t.test()` with the `paired=TRUE` option, as follows.

```
> t.test(aggmoon, aggother, paired=TRUE, mu=0)

        Paired t-test
```

```
data:  aggmoon and aggother
t = 6.4518, df = 14, p-value = 1.518e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.623968 3.241365
sample estimates:
mean of the differences
              2.432667
```

The t statistic of 6.4518 with a very small p-value indicates that we can reject the hypothesis that disruptive behavior is the same on moon days and non-moon days. The test also gives us a 95% confidence interval for the average difference in number of disruptive episodes as (1.62, 3.24).

Note that the paired t-test can be obtained equivalently by doing a one-sample t-test of the difference (`aggdiff`). This is done as follows; you can check that the results are exactly the same.

```
> t.test(aggdiff, mu=0)

        One Sample t-test

data:  aggdiff
t = 6.4518, df = 14, p-value = 1.518e-05
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 1.623968 3.241365
sample estimates:
mean of x
 2.432667
```

6.3 More than Two Categories: Analysis of variance (ANOVA)

I have previously loaded the dataframe AfifiClark at and entered attach(AfifiClark) so R can find the variables within the dataframe.

```
> names(AfifiClark)
 [1] "id"       "sex"      "age"      "marital"  "educat"   "employ"
 [7] "income"   "relig"    "cesd"     "cases"    "drink"    "health"
[13] "regdoc"   "treat"    "beddays"  "acuteill" "chronill" "educat.f"
> table(relig)
relig
Protestant   Catholic     Jewish       None      Other
       155         51         30         56          2
```

We look at the relationship between religious affiliation (`relig`), a categorical variable, and depressive symptoms (`cesd`), a quantitative variable. Are people in some affiliations protected from, or more prone to, depression?

Before beginning the analysis I want to get rid of the Other category of `relig`, which has only two cases. (This step is generally not needed with other data sets.) To do this I create a new variable `relig2`, with the Other category forced to missing, as follows.

```
> relig2 <- factor(relig, levels=c("Protestant", "Catholic", "Jewish", "None"))
> table(relig2)
relig2
```

```
Protestant   Catholic     Jewish       None
       155         51         30         56
```

Descriptive Analysis

A table of cesd mean, SD and number of observations for each denomination is obtained as follows.

```
> means <- tapply(cesd, relig2, mean)
> SDs <- tapply(cesd, relig2, sd)
> Ns <- tapply(cesd, relig2, length)
> cbind(Mean=means, SD=SDs, N=Ns)
               Mean        SD   N
Protestant  7.677419  8.657206 155
Catholic    8.666667  6.760671  51
Jewish     11.266667  8.885996  30
None       10.875000 10.174947  56
```

In this block of commands I use the `tapply()` function to calculate the mean, standard deviation, and number of cases for each of the four denominations. Note how each call of `tapply()` specifies the variable to be summarized (`cesd`), the grouping variable (`relig2`), and the statistic to be computed (mean, sd or length). I saved the results into three vectors (`means`, `SDs`, and `Ns`). Then I used the matrix function `cbind()` ("column bind") to arrange the vectors into a table with three columns, which is then printed. We can see that mean `cesd` score varies across denominations, from 7.7 for Protestant to 11.3 for Jewish.

The side-by-side boxplot is produced as follows.

```
> plot(cesd ~ relig2, notch=TRUE)
```

The plot is shown in Figure 6. I used the `notch=TRUE` option to get a preliminary sense of the significance of the differences in median `cesd` between denominations. The difference in *medians* between two groups is significant at the .05 level if the notches for the groups do not overlap. The Jewish-Protestant and the None-Protestant differences appear significant.

Inference

We use ANOVA with the function `aov()` to test the hypothesis that the mean `cesd` scores for the four groups are all the same, versus the alternative that at least one mean score is different. More formally we test $H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4$ versus the alternative H_a : at least one μ_i is different. This is done as follows.

```
> aovmod1 <- aov(cesd ~ relig2)
> summary(aovmod1)
             Df Sum Sq Mean Sq F value Pr(>F)
relig2        3    620  206.53   2.727 0.0444 *
Residuals   288  21811   75.73
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
2 observations deleted due to missingness
```

I saved the output of the `aov()` function into an object I called `aovmod1`. Then I displayed this object with the `summary()` function. The output shows that the value of the F statistic is 2.727 with p-value .0444, so the result is (just) significant at the .05 level. Thus we have sufficient evidence to reject the null hypothesis that all the means are the same.

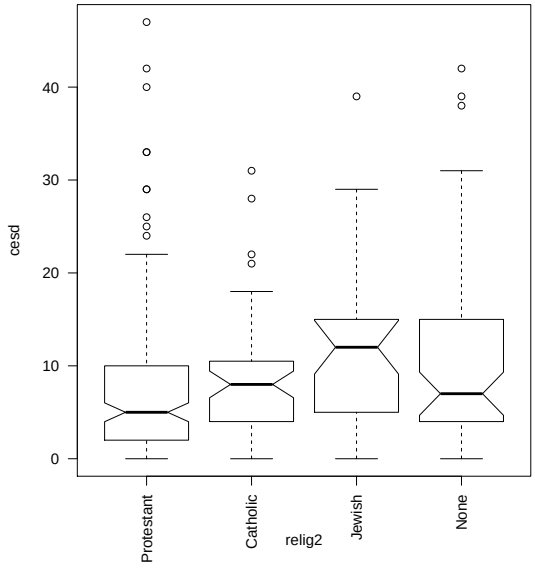

Figure 6: Side-by-side boxplot of cesd depressive symptoms score by religious affiliation.

Additional Analysis I – Effects of Individual Categories

The output provided by `summary()` is extremely terse and gives no information about the nature of the differences in `cesd` scores between affiliations. For a more detailed summary we can apply the `summary.lm()` function to the `aovmod1` object we saved.

```
> summary.lm(aovmod1)

Call:
aov(formula = cesd ~ relig2)

Residuals:
    Min      1Q  Median      3Q     Max
-11.267  -5.677  -2.467   2.831  39.323

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    7.6774     0.6990  10.983   <2e-16 ***
relig2Catholic 0.9892     1.4048   0.704   0.4819
relig2Jewish   3.5892     1.7358   2.068   0.0396 *
relig2None     3.1976     1.3568   2.357   0.0191 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.702 on 288 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.02762,   Adjusted R-squared:  0.01749
F-statistic: 2.727 on 3 and 288 DF,  p-value: 0.04437
```

The `Estimates` in the `Coefficients:` section are interpreted as follows: `(Intercept)` estimates mean `cesd` for the `Protestant` category, which is the omitted category in the model.

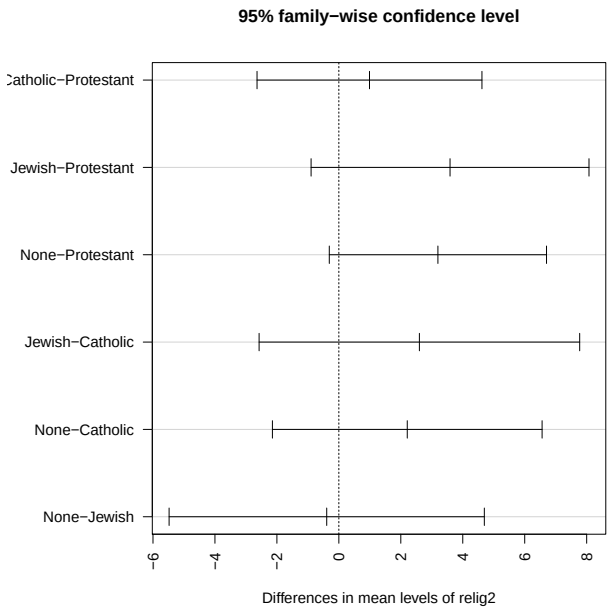

Figure 7: TukeyHSD plot of the cesd score by religious afiliation model aovmod1.

You can check that the value of the coefficient (7.6774) is the same as that calculated in the descriptive analysis table. The coefficients for the other categories represent the differences between the category means and the omitted category, i.e. Protestant. For example, the coefficient of 0.9892 for `relig2Catholic` is the difference between the Catholic mean (8.666667 in the descriptive table) and the Protestant mean (7.6774). The coefficients for the Jewish and for the None categories are significant.

Additional Analysis II – Multiple Comparisons

The `summary.lm()` function provides tests of the difference between category means and the mean of the omitted category, but we may be interested in other group comparisons. To test all possible group differences in `cesd` score we can use the `TukeyHSD()` function, which is named for statistician John Tukey and the name of the method (Honest Significant Differences). The function is applied directly to the model object produced by `aov()` and the results represented graphically, as follows.

```
> par(las=2)
> par(mar=c(5,8,4,2))
> plot(TukeyHSD(aovmod1))
```

(The `par()` commands are used to make the labels along the y axis of the graph horizontal (`las=2`) and to adjust the margins so there is enough room along the y axis to write the names of the comparisons.)

The TukeyHSD plot is shown in Figure 7. For each comparison the TukeyHSD plot shows the CI for the difference in mean `cesd` between the two groups. The vertical line is set at zero. If the CI corresponding to the difference straddles the line, i.e. contains zero, it means that the difference is not significant. In none of the comparisons is the zero line excluded from the CI, indicating that none of the differences is statistically significant.

TukeyHSD can also be printed, as follows. The Tukey HSD p-value for the difference in means between two groups is shown in the column `p adj` ("p adjusted"). Consistent with the graph, no

difference reaches significance, although None-Protestant with p=.088 and Jewish-Protestant with p=.166 come closest.

```
> TukeyHSD(aovmod1)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = cesd ~ relig2)

$relig2
                          diff        lwr      upr     p adj
Catholic-Protestant  0.9892473 -2.6409843 4.619479 0.8953885
Jewish-Protestant    3.5892473 -0.8962546 8.074749 0.1662183
None-Protestant      3.1975806 -0.3085898 6.703751 0.0879800
Jewish-Catholic      2.6000000 -2.5742625 7.774262 0.5646395
None-Catholic        2.2083333 -2.1444239 6.561091 0.5566471
None-Jewish         -0.3916667 -5.4796566 4.696323 0.9972077
```

Why the absence of significant difference, since we had previously found significant group differences for None-Protestant and Jewish-Protestant in the `aovmod1` printout? This is because here we are making several simultaneous comparisons, increasing the chance of finding a significant difference by chance alone. The TukeyHSD method adjusts for this by making it more difficult to find a significant difference. Hence the two differences that originally appeared significant are no longer so.

Additional Analysis III – Simplifying the Model

(See Crawley p.438).

The ANOVA model that we have estimated included all four religious affiliation categories, which together account for a significant proportion of the variance in depressive symptoms, as per the F-test. Could we account for the `cesd` variance as well with fewer categories? To investigate this we note that the coefficients of `relig2Jewish` (3.5892) and of `relig2None` (3.1976) are very similar, indicating that mean `cesd` are almost the same for these two affiliations. Let's see what happens when we rerun the ANOVA after combining the Jewish and None categories into a common category I will label `NonChristian` for simplicity.

```
> relig3 <- relig2
> levels(relig3)[3:4] <- "NonChristian"
> levels(relig3)
[1] "Protestant"   "Catholic"     "NonChristian"
> aovmod3 <- aov(cesd ~ relig3)
> anova(aovmod1, aovmod3)
Analysis of Variance Table

Model 1: cesd ~ relig2
Model 2: cesd ~ relig3
  Res.Df   RSS Df Sum of Sq    F   Pr(>F)
1    288 21811
2    289 21814 -1   -2.9967 0.0396  0.8425
```

The simplified affiliation variable `relig3` is created from a copy of `relig2` by replacing levels 3 and 4 of `relig2` (`Jewish` and `None`) by the single level `NonChristian`. The new variable has only three levels.

I then reran the ANOVA with relig3, saving the output in object aovmod3. I then compared the aovmod1 and aovmod3 models using anova(). The non-significant F-test (p-value 0.8425) means that the two models are statistically identical, so based on the principle of Occam's razor we decide in favor of relig3, which uses up fewer degrees of freedom.

Next we look at the detail of aovmod3 using the summary.lm() function.

```
> summary.lm(aovmod3)

Call:
aov(formula = cesd ~ relig3)

Residuals:
    Min      1Q  Median      3Q     Max
-11.012  -5.677  -2.339   2.988  39.323

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)          7.6774     0.6978  11.002  < 2e-16 ***
relig3Catholic       0.9892     1.4025   0.705  0.48116
relig3NonChristian   3.3342     1.1682   2.854  0.00463 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.688 on 289 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.02749,   Adjusted R-squared:  0.02076
F-statistic: 4.084 on 2 and 289 DF,  p-value: 0.01782
```

The coefficient of relig3NonChristian, which combines Jewish and None identifications, has p-value 0.00463, which is better than the $\alpha = .01$ level (as indicated by the two asterisks). On the other hand the coefficient of relig3Catholic is non-significant, implying that the difference in mean cesd between the Catholic category and the intercept, corresponding to the omitted Protestant category, is not statistically different from zero. So we wonder whether it is possible to simplify the model further by collapsing the Protestant and Catholic categories into a single category.

Proceeding as before, I created a new variable relig4, starting with a copy of relig3 and replacing the first two levels with a single new level called (of course) Christian. I then ran the ANOVA using aov() with relig4, saving the output in aovmod4, and comparing it with aovmod3 using anova().

```
> relig4 <- relig3
> levels(relig4)[1:2] <- "Christian"
> levels(relig4)
[1] "Christian"    "NonChristian"
> aovmod4 <- aov(cesd ~ relig4)
> anova(aovmod3, aovmod4)
Analysis of Variance Table

Model 1: cesd ~ relig3
Model 2: cesd ~ relig4
  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1    289 21814
2    290 21852 -1   -37.553 0.4975 0.4812
```

The F-test in the `anova()` comparison has large p-value (0.4812), so we cannot reject the hypothesis that the two models are the same. Thus as before the principle of parsimony directs us to prefer the model with `relig4`, which has only 2 categories, over the one with `relig3`, which has 3. We look at the detailed results for `aovmod4` using `summary.lm()`.

```
> summary.lm(aovmod4)

Call:
aov(formula = cesd ~ relig4)

Residuals:
    Min      1Q  Median      3Q     Max
-11.012  -5.922  -2.012   3.011  39.078

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)          7.9223     0.6048  13.099  < 2e-16 ***
relig4NonChristian   3.0893     1.1144   2.772  0.00593 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.68 on 290 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.02581,   Adjusted R-squared:  0.02245
F-statistic: 7.684 on 1 and 290 DF,  p-value: 0.00593
```

The summary shows that the intercept, which corresponds to the mean `cesd` score for the omitted Christian category, is about 8, and mean `cesd` for the `NonChristian` category is about 3 points higher, or 11. The difference in category means is significant at better than the .01 level. This final outcome is intellectually satisfying, as it means we have successfully simplified a fairly complex set of differences in depressive symptoms among four religious identifications into a simpler, and presumably more fundamental, difference between Christian and non-Christian affiliations: respondents with non-Christian affiliations report significantly higher depressive symptoms scores (by about 3 cesd points) than respondents with Christian affiliations. With respect to the `cesd` score it makes no difference whether the respondent reports a Protestant or Catholic affiliation, or a Jewish one or None.

How "big" is the effect of the Christian vs. non-Christian contrast on `cesd` score? A common measure of standardized effect size is Cohen's *d*, named after Jacob Cohen (1992). It is calculated from information in the `summary.lm()` printout as the ratio of the coefficient of NonChristian (3.0893) to the residual standard error (8.68).

```
> 3.0893/8.68
[1] 0.3559101
```

The *d* of .356 means that the effect of denomination is equivalent to about a third of a standard deviation in `cesd` score within groups (denominations), which is in the small to medium size range (Cohen 1992).

What do we make of this result? Should we join a Christian denomination to avoid depression? (UNC people: Imagine what the Preacher at the Pit could do with this.) Before we get too excited let's mention a couple of caveats. First, the R-squared for `aovmod4` in the `summary.lm` output is 0.02581, indicating that the religious affiliation dichotomy explains only 2.6% of the variance in `cesd`, which corresponds to a correlation of 0.161 ($= \sqrt{0.02581}$). Thus religious identification has only a fairly weak association with depressive symptoms. Second, we have no strong basis for inferring causation, as the data are observational and the weak association

might be due to lurking variables affecting both affiliation and depressive score, and depressive state may even conceivably affect religious affiliation.

Additional Analysis IV – Comparing Models with AIC or BIC

In the previous section we compared the different models of the impact of religious affiliation on depressive symptoms using the `anova()` function. We could do this because the models we looked at are nested. There are alternative measures of fit that allow models to be compared even when they are not nested. The two most commonly used measures of that type are AIC (Akaike's Information Criterion) and BIC (Bayesian Information Criterion). They are easy to calculate in R from the `aov` model objects that we saved earlier.

```
> AIC(aovmod1, aovmod3, aovmod4)
        df      AIC
aovmod1  5 2098.180
aovmod3  4 2096.220
aovmod4  3 2094.723
> BIC(aovmod1, aovmod3, aovmod4)
        df      BIC
aovmod1  5 2116.564
aovmod3  4 2110.927
aovmod4  3 2105.753
```

For both the AIC and the BIC, when comparing models the smaller the value of the criterion, the better the fit. We see that both AIC and BIC decrease (the fit increases) as we go from `aovmod1`, through `aovmod3`, to `aovmod4`, with the latter model the best fitting. Thus we would have made the same choice of preferred model (the simple `aovmod4` model) had we used AIC or BIC instead of `anova()` comparisons.

6.4 Independent Samples t.test vs. ANOVA

This section on a Categorical -> Quantitative relationship has discussed the independent samples t-test (for a categorical variable with two categories) and analysis of variance (for a variable with more than two categories) separately, as is traditionally done. Is there a true substantive difference between the two methodologies?

The answer is no. The two methods are in fact computational variations of the same general linear model. To see this identity concretely we compare the independent samples t-test with ANOVA for the sex difference in mean income in the AfifiClark data. After loading and attaching the data, `t.test()` was used with option `var.equal=TRUE` as follows (repeated here for convenience).

```
> attach(AfifiClark)
> t.test(income ~ sex, var.equal=TRUE)

        Two Sample t-test

data:  income by sex
t = 3.1319, df = 292, p-value = 0.001913
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.109297 9.243531
sample estimates:
  mean in group male mean in group female
            24.10811             18.43169
```

For ANOVA we save the output of the `aov()` function into a model object (that I choose to call `aov1`), and then look at that object using the detailed `summary.lm()` function, as follows.

```
> aov1 <- aov(income ~ sex)
> summary.lm(aov1)

Call:
aov(formula = income ~ sex)

Residuals:
    Min      1Q  Median      3Q     Max
-22.108 -11.108  -4.270   5.318  46.568

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   24.108      1.430  16.860  < 2e-16 ***
sexfemale     -5.676      1.812  -3.132  0.00191 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.07 on 292 degrees of freedom
Multiple R-squared:  0.0325,    Adjusted R-squared:  0.02919
F-statistic: 9.809 on 1 and 292 DF,  p-value: 0.001913
```

Comparison of the two printouts shows that they provide the same essential information.

- `t.test()` gives estimates of mean income for male sex (24.10811) and female sex (18.43169); `aov()` estimates mean income for male (24.108) as the coefficient for `(Intercept)` and the *difference* between female and male income (−5.676 = 18.43169 − 24.10811) as the coefficient of `sexfemale`.
- the test statistic (`t = 3.1319`), degrees of freedom (`df = 292`) and p-value (`p-value = 0.001913`) of `t.test()` are found in `aov()` as the `t value` for `sexfemale` (−3.132) with p-value `Pr(>|t|)` of 0.00191; the df of 292 are found on the last line of the `aov()` output. (The test statistic is negative rather than positive in `aov()` because it refers to the female − male, rather than the male − female difference.)
- `aov()` provides some information, such as the residual standard error (15.07) that `t.test()` does not.

Differences between `t.test()` and `aov()` are thus superficial, the result of computational strategy choices in pre-computer days.

7 Relationship – Categorical -> Categorical

I have previously loaded the dataframe AfifiClark at .... I am interested in the relationship between religious denomination and drinking habits (whether respondent drinks or not). I use `ls()` to make sure the data is in memory and `attach()` it to make variables accessible by name. I make univariate tables of variables `relig` and `drink`.

```
> ls()
[1] "AfifiClark"
> attach(AfifiClark)
> table(relig)
relig
Protestant   Catholic     Jewish       None      Other
       155         51         30         56          2
```

```
> table(drink)
drink
yes  no
234  60
```

There are only two cases in the `Other` category of `relig`. This is going to create problems later on, so I create a new factor `relig2` that excludes these two cases. This is done by not including `Other` among the levels (categories) of `relig2`.

```
> relig2 <- factor(relig, levels=c("Protestant", "Catholic", "Jewish", "None"))
> table(relig2)
relig2
Protestant   Catholic     Jewish       None
       155         51         30         56
```

Now we have substantial numbers of cases in all categories. (In other situations it may make sense to consolidate categories instead of dropping cases, but here we don't know enough about these Other to assign them to another affiliation.)

7.1 Descriptive Analysis

The basic descriptive analysis using both numbers and graph is as follows.

```
> tab <- table(relig2, drink)
> tab
            drink
relig2       yes  no
  Protestant 112  43
  Catholic    44   7
  Jewish      26   4
  None        50   6
> prop.table(tab, 1)
            drink
relig2             yes        no
  Protestant 0.7225806 0.2774194
  Catholic   0.8627451 0.1372549
  Jewish     0.8666667 0.1333333
  None       0.8928571 0.1071429
> spineplot(tab)
```

Here `relig2` is the explanatory variable, and `drink` is the response variable. I first created a contingency table (which I called `tab`) with the `table()` function. For convenience I listed the explanatory variable first in `table(relig2, drink)`, so categories of `relig2` will sit in the rows of the table.[2]

Then I used `prop.table(tab, 1)` to calculate the proportions of respondents of each denomination who drink or do not drink, so proportions add up to 1 in each row. Another way to describe the numbers is to say that each row shows the conditional distribution of drinking given the religious denomination. To see the association, we compare the rows. We see that only 72% of Protestants drink, with 28% abstaining. Higher (and rather similar) percentages of the other denominations drink: 86% of Catholic, 87% of Jewish, and 89% of the None.

The pattern of association is shown graphically using the function `spineplot(tab)` (Figure 8). The spineplot is similar to a stacked barchart, but additionally represents the relative

[2] Some R functions assume that the explanatory variable sits in the rows and the response variable sits in the columns. However this is not a strong convention, and the opposite arrangement is often used, including for typographical reasons.

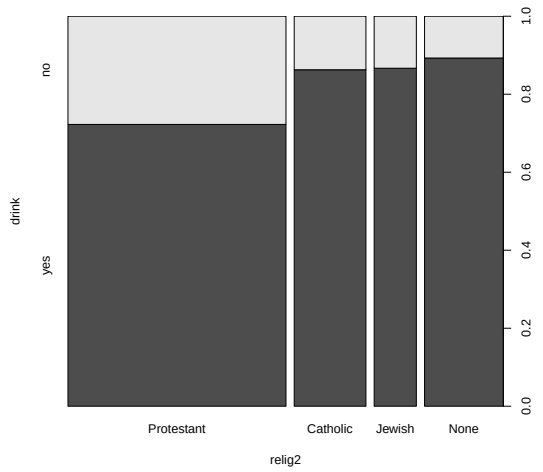

Figure 8: Spineplot of whether respondent drinks by religious denomination

sizes of the categories of the explanatory variable (here, religious denomination) as the widths of the bars, so we can see at a glance that Protestant is by far the largest denomination, Jewish the smallest. The plot shows the same pattern of low drinking rate of Protestant relative to other denominations.

Alternative for Percentaging Table Nicer-looking tables of conditional percentages are obtained with functions `rowPercents()` and `colPercents()` from John Fox's **Rcmdr** package. The function definitions are listed in Appendix A. To use them copy the listings for both functions and enter them at the R prompt. Then you can reproduce the drinking by denomination example as follows.

```
> tab <- table(relig2, drink)
> rowPercents(tab)
           drink
relig2       yes   no Total Count
  Protestant 72.3 27.7   100   155
  Catholic   86.3 13.7   100    51
  Jewish     86.7 13.3   100    30
  None       89.3 10.7   100    56
```

7.2 Inference

The inferential part of the analysis is carried out using `chisq.test()` as follows.

```
> chisq.test(tab)

        Pearson's Chi-squared test

data:  tab
X-squared = 10.6411, df = 3, p-value = 0.01383

> chisq.test(tab)$residuals
           drink
```

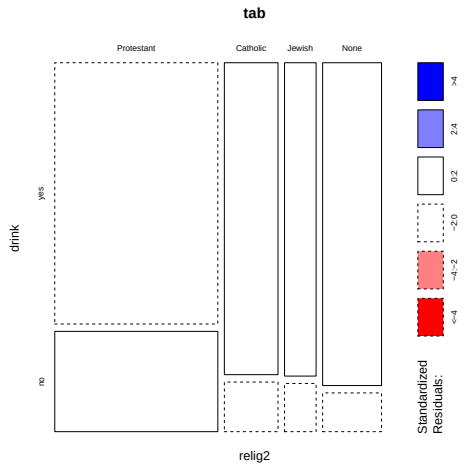

Figure 9: Mosaic plot of drinking by religious denomination. Positive residuals are shown with solid lines; negative residuals with dashed lines. In this case none of the residuals reaches the cutoff of 2 and thus all the panels are blank.

```
relig2              yes         no
  Protestant -1.0048081  1.9758387
  Catholic    0.5466045 -1.0748344
  Jewish      0.4433238 -0.8717449
  None        0.8255752 -1.6233979
> mosaicplot(tab, shade=TRUE)
```

`chisq.test(tab)` finds a chi-squared of 10.6411 with 3 df which has p-value of .014. Thus it would be highly unlikely to find these results (a chi-squared as large as this) if the null hypothesis that the proportion drinking is the same for all denominations holds. Thus we reject the null hypothesis.

The chi-squared test only allows an overall test of the null. To obtain further insight into the nature of the differences in drinking pattern among denominations we can look at the residuals obtained as `chisq.test(tab)$residuals`. Residuals are the standardized discrepancies between observed counts and counts expected under the hypothesis of independence and can be interpreted as z-scores. We see that the largest residual is 1.9758387, significant at the .05 level, corresponding to the Protestant propensity to not drink that we had noticed in the descriptive analysis.

The Mosaic plot (Figure 9) provides a visual representation of the residuals. The area of each cell is proportional to the frequency of observations in the cell. The outline of a cell indicates whether the residual is positive (solid line) or negative (dashed line). The cells are shaded according to the size of the residual, with cutoffs at ±2 and ±4. In this analysis the largest residual (1.9758387) falls just short of the cutoff point of 2 so none of the cells are shaded.

7.3 Advanced Analysis I – Model Simplification

We noticed earlier that while drinking is markedly less prevalent among Protestants, rates for the other three affiliations were quite similar. Could it be that the differences among non-Protestant affiliations are non-significant statistically? If that's the case, we should be able to fit the model just as well with a two-level categorical variable distinguishing only between Protestant and Other affiliation. The null then would be that prevalence is different for Protestant, but the same

for the other three affiliation. We check this by proceeding as follows.

```
> relig3 <- relig2
> levels(relig3)
[1] "Protestant" "Catholic"   "Jewish"     "None"
> levels(relig3)[2:4] <- "Other"
> levels(relig3)
[1] "Protestant" "Other"
> tab3 <- table(relig3, drink)
> tab3
            drink
relig3       yes  no
  Protestant 112  43
  Other      120  17
```

In the listing above I created a new categorical variable relig3 that keeps the Protestant category but replaces the other three categories (levels 2 to 4) by a new `Other` level. I then recalculated the contingency table as `tab3`.

Next we redo the chi-squared test and residual analysis and redraw the mosaic plot.

```
> chisq.test(tab3)

        Pearson's Chi-squared test with Yates' continuity correction

data:  tab3
X-squared = 9.5546, df = 1, p-value = 0.001994

> chisq.test(tab3)$residuals
            drink
relig3             yes        no
  Protestant -1.004808  1.975839
  Other       1.068781 -2.101634
> mosaicplot(tab3, shade=TRUE)
```

This time we obtain a chi-squared statistic of 9.5546 with 1 df, yielding a p-value of .001994 which is even smaller than what we had found earlier. The residuals continue to show an excess of abstaining Protestants, but this time the residual for the Other category is negative and significant, indicating a deficit. The mosaic plot shows the same pattern visually (Figure 10). The cell for Other and not drinking is shaded red, indicating a significant deficit in abstaining.

We can compare the models for `tab` (4 affiliation categories) and `tab3` (2 affiliation categories) more formally by noting that `tab3` is "nested" (i.e., is a constrained version of) within `tab`, so the difference in the chi-squared for the two models is distributed as chi-squared with $2 = 3 - 1$ df (the difference in dfs between the models). The p-value of this difference is calculated in one line as follows.

```
> pchisq(abs(10.6411-9.5546), 3-1, lower.tail=FALSE)
[1] 0.5808574
```

The p-value is large, indicating that the difference in chis-squared is non-significant, so we prefer the simpler model with two categories (`tab3`) by parsimony.

7.4 Advanced Analysis II – Relative Risk and Odds Ratio in Two-By-Two Tables

Agresti (2007, p.26) discusses results from a medical study that he describes as follows.

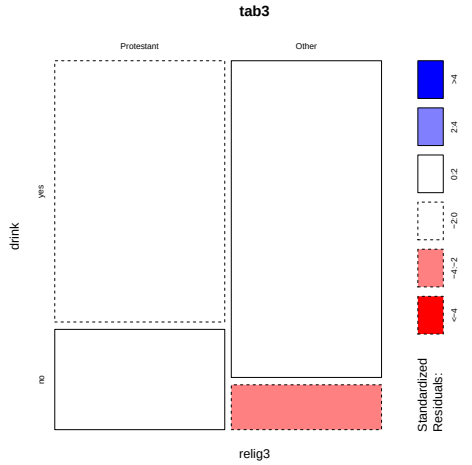

Figure 10: Mosaic plot of drinking by religious affiliation after consolidating Catholic, Jewish and None categories into a single Other category.

Table 4: Cross Classification of Aspirin Use and Myocardial Infarction

	Myocardial Infarction		
	Yes	No	Total
Placebo	189	10,845	11,034
Aspirin	104	10.933	11,037

Source: Agresti (2007), Table 2.3, p.27.

Table [4] is from a report on the relationship between aspirin use and myocardial infarction (heart attacks) by the Physicians' Health Study Research Group at Harvard Medical School. The Physicians' Health Study was a five-year randomized study testing whether regular intake of aspirin reduces mortality from cardiovascular disease. Every other day, the male physicians participating in the study took either one aspirin tablet or a placebo. The study was "blind" – the physicians in the study did not know which type of pill they were taking.

We begin with a standard analysis of the difference in proportions with infarction between Placebo and Aspirin subjects. We first create a labeled table as follows.

```
> tab <- matrix(c(189, 10845, 104, 10933), nrow=2, byrow=TRUE)
> rownames(tab) <- c("Placebo", "Aspirin")
> colnames(tab) <- c("Yes", "No")
> names(dimnames(tab)) <- c("Treatment", "Myocardial Infarction")
> tab
         Myocardial Infarction
Treatment Yes    No
  Placebo 189 10845
  Aspirin 104 10933
```

Next we calculate the conditional probabilities of Infarction given treatment.

```
> round(100*prop.table(tab, 1), digits = 1)
```

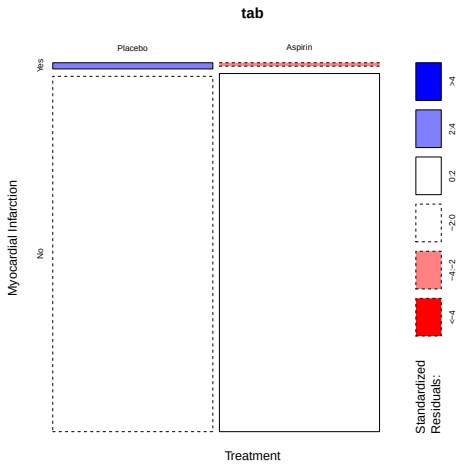

Figure 11: Mosaic plot for cross classification of aspirin use and myocardial infarction. There is a significant excess of the disease in the Placebo group, and a corresponding deficit in the Aspirin group. The overall infarction rate is small.

```
         Myocardial Infarction
Treatment Yes   No
  Placebo 1.7 98.3
  Aspirin 0.9 99.1
```

We continue the standard analysis with a test of the null hypothesis of equality of proportion of infarction in Placebo and Aspirin groups using `prop.test()`.

```
> prop.test(tab, correct=FALSE)

        2-sample test for equality of proportions without continuity correction

data:  tab
X-squared = 25.0139, df = 1, p-value = 5.692e-07
alternative hypothesis: two.sided
95 percent confidence interval:
 0.004687751 0.010724297
sample estimates:
    prop 1     prop 2
0.01712887 0.00942285
```

`prop.test()` gives us a very small p-value, so we reject the null hypothesis of equality of proportions. We can also visualize the comparison using a mosaic plot (Figure 11).

```
> mosaicplot(tab, shade=TRUE)
```

Both the statistical analysis and graph show that there is a highly significant difference in proportions of myocardial infarction between the Placebo and Aspirin groups, so that aspirin appears protective from infarction. However the overall rate of infarction is quite small, which makes substantive interpretation of the results awkward: What does it mean to say that there is a difference of $1.7 - 0.9 = .8\%$ between the treatment groups? This is a very common data configuration in medical research since a disease is typically (and fortunately) a rare event.

A better alternative to the difference in risk of infarction between Placebo and Aspirin groups is the *ratio* of the risks, called the *relative risk* or *risk ratio* (RR).

Using the numbers in the original contingency table (Table 4), the risk is 189/(189+10845) for the Placebo group and 104/(104+10933) for the Aspirin group, so the ratio can be calculated in R as follows.

```
> rr <- (189/(189+10845))/(104/(104+10933))
> rr
[1] 1.817802
```

The relative risk of an attack for Placebo vs Aspirin is 1.82. Thus the risk of an attack is 82% greater in the Placebo group than in the Aspirin group. This phrasing of the results in terms of the risk ratio may be more compelling than that in terms of a difference in risks.

The *odds* of an event are the ratio of the probabilities of an event happening to that of it not happening. With the numbers in Table 4 the odds of infarction are 189/10845 for the Placebo group and 104/10933 for the Aspirin group. The *odds ratio* (OR) is the odds for one group divided by the odds for the other, which we calculate in R as follows.

```
> or <- (189/10845)/(104/10933)
> or
[1] 1.832054
```

We note that the OR (1.832) is very close to the RR (1.818). This is not a coincidence, but the manifestation of an important property of the odds ratio: when the probability of an event is close to zero, as it is in this example, the OR is a good approximation for the risk ratio RR. The reason for this can be understood by comparing the formulas used to compute the RR and the OR: when the number of events in a group is small (e.g., 189), the number of non-events (10845) is close to the total number of observations in a group (10845+189). This property is especially useful in some study designs (beyond the scope of this course) where the OR can be estimated, but not the RR.

The package **propCIs** provides functions to calculate confidence intervals for the three measures of effect: difference in means, RR, and OR. The package has to be installed beforehand. (If using Windows, make sure to run R as administrator when installing.) The numbers in the contingency table have to be entered separately and the desired confidence level specified explicitly (there is no default), as illustrated in the following run.

```
> install.packages("PropCIs")
> library(PropCIs)
> diffscoreci(189, 11034, 104, 11037, 0.95) # CI for difference of proportions

95 percent confidence interval:
 0.004716821 0.010788501

> riskscoreci(189, 11034, 104, 11037, 0.95) # CI for relative risk

95 percent confidence interval:
 1.433904 2.304713

> orscoreci(189, 11034, 104, 11037, 0.95) # CI for odds ratio

95 percent confidence interval:
 1.440802 2.329551
```

For example, we can be 95% confident that the risk of an infarction is between 1.43 and 2.30 times greater for the Placebo group than for the Aspirin group.

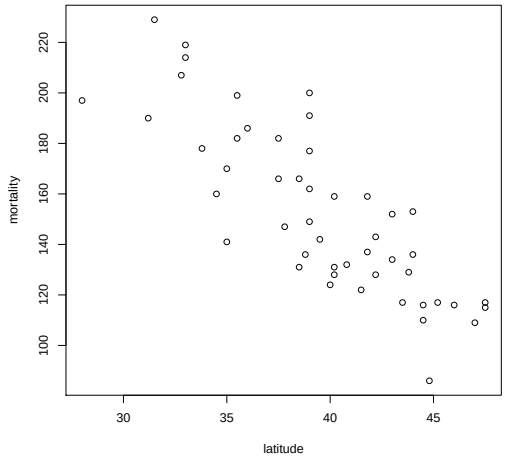

Figure 12: Simple scatterplot of melanoma mortality by latitude.

8 Relationship – Quantitative -> Quantitative

8.1 Descriptive Analysis

The data frame Melanoma can be input as shown. The outcome of interest is `mortality`, the number of white males who died due to malignant melanoma 1950-1969 per one million inhabitants. The explanatory variable is `latitude` of the center of the state in degrees North from the equator (so a higher latitude corresponds to a more northern location of the state).

```
> Melanoma <- read.csv("http://www.unc.edu/%7Enielsen/soci252/activities/USmelanoma.csv")
> ls()
[1] "AfifiClark" "lbls"       "Melanoma"   "oldpar"     "pct"
[6] "tab"
> head(Melanoma)
            mortality latitude longitude ocean
Alabama           219     33.0      87.0   yes
Arizona           160     34.5     112.0    no
Arkansas          170     35.0      92.5    no
California        182     37.5     119.5   yes
Colorado          149     39.0     105.5    no
Connecticut       159     41.8      72.8   yes
> attach(Melanoma)
```

Is there a relationship between mortality and latitude? The scatterplot is produced as follows (Figure 12). It shows a strong negative relationship between mortality and latitude. (Reader, why?) Given the apparent linearity it makes sense to calculate the correlation, which is −.825, confirming the strong negative relationship.

```
> plot(latitude, mortality)
> cor(latitude, mortality)
[1] -0.8245178
```

8.2 Inference

We pursue the analysis by calculating the regression of mortality on latitude, as follows. As typical in R, estimation is done in two steps: first the `lm()` ("linear model") function is used to create a model object, which I call `mod1`; then the essential information about the regression is extracted from `mod1` and printed using the `summary()` function.

```
> mod1 <- lm(mortality~latitude)
> summary(mod1)

Call:
lm(formula = mortality ~ latitude)

Residuals:
    Min      1Q  Median      3Q     Max
-38.972 -13.185   0.972  12.006  43.938

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 389.1894    23.8123   16.34  < 2e-16 ***
latitude     -5.9776     0.5984   -9.99 3.31e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.12 on 47 degrees of freedom
Multiple R-squared:  0.6798,    Adjusted R-squared:  0.673
F-statistic:  99.8 on 1 and 47 DF,  p-value: 3.309e-13
```

The section `Coefficients:` of the summary output provides regression coefficients in the column labeled `Estimate`. The estimate for `(Intercept)` is the coefficient b_0 and the estimate for `latitude` is the coefficient b_1 of the explanatory variable in the prediction equation

$$\widehat{Y} = b_0 + b_1 X.$$

In this example $b_0 = 389.189$ and $b_1 = -5.978$, so that

$$\widehat{Y} = 389.189 - 5.978X$$

where Y denotes melanoma mortality and X latitude.

The most common type of inference in a simple regression like this one is a test of the null hypothesis that the coefficient β_1 of X (the slope) is equal to zero, in other words, there is no linear relationship between X and Y. The `Coefficients:` section provides all the necessary information. Focusing on the coefficient of latitude (-5.9776), one sees that the standard error is 0.5984. The test statistic for the hypothesis that $\beta_1 = 0$ is $(-5.9776 - 0)/0.5984 = -9.99$, and is provided in column `t value`. The two-tailed p-value (probability of obtaining a t value as extreme as this if the slope is really zero) is provided in column `Pr(>|t|)`. Here the p-value is extremely small at 3.31×10^{-13}, or essentially zero. The null hypothesis that there is no linear relationship between melanoma mortality and latitude is resoundingly rejected.

The hypothesis that the intercept is equal to zero can be tested in the same way, but is usually of lesser interest. The significance of the linear relationship can also be tested using the function `corr.test()` as follows. As the output shows, that test is equivalent to that performed by `lm()`.

```
> cor.test(latitude, mortality)

        Pearson's product-moment correlation
```

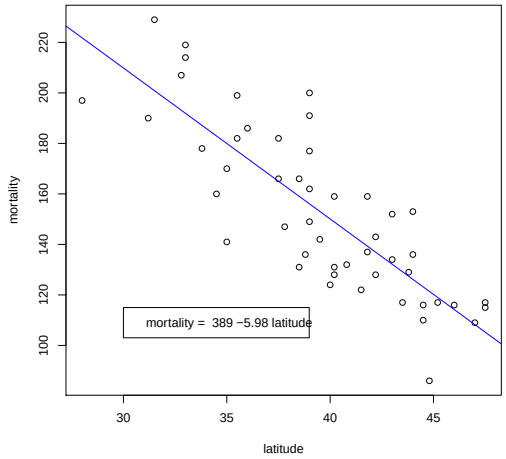

Figure 13: A more elaborate scatterplot of melanoma mortality by latitude.

```
data:  latitude and mortality
t = -9.9898, df = 47, p-value = 3.309e-13
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.8976036 -0.7073128
sample estimates:
       cor
-0.8245178
```

Knowing the estimated regression line, we can prepare a more informative scatterplot as follows (Figure 13).

```
> plot(latitude, mortality)
> abline(mod1, col="blue")
> # add regression equation
> cf=coefficients(mod1)
> legend(30,115,legend=paste("mortality = ",round(cf[1],0),round(cf[2],2),"latitude"))
```

9 Relationship – Quantitative -> Categorical

Analysis and inference of Quantitative -> Categorical relationships involve advanced statistical methods, the most common being logistic regression (for a dichotomous response) and multinomial regression (for a categorical response with more than two categories). These methods are beyond the scope of this course.

However two (relatively novel) types of graphs permit a visual assessment of a Q -> C relationship.

9.1 Descriptive Analysis

A Q -> C relationship can be represented as a *spinogram* (using `spineplot()`) or as a *conditional density plot* (using `cdplot()`). The graphs are created with `spineplot(y ~ x)` and

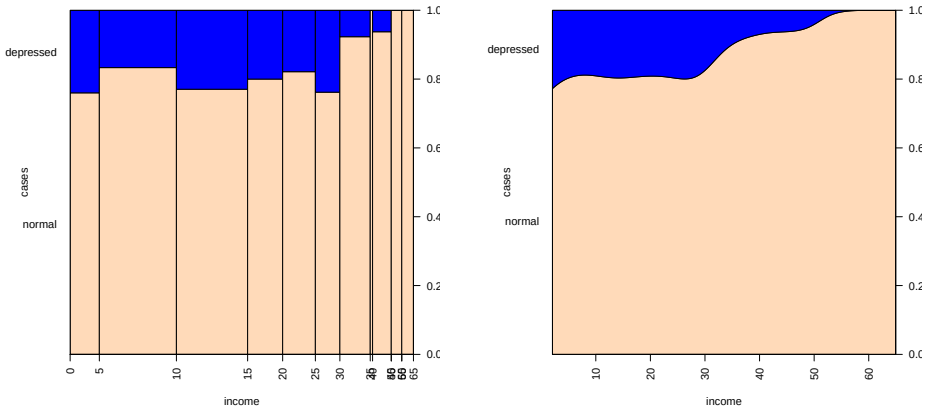

Figure 14: Graphic representations of the Income -> Depression relationship: spinogram (left) and conditional density plot (right). Dark blue area represents proportion depressed.

`cdplot(y ~ x)`, where the response `y` is categorical (a factor) and the explanatory variable `x` is quantitative.[3] The graphs are illustrated for the relationship between depression (factor `cases` with levels `normal` and `depressed`) and `income` in the `AfifiClark` data.

```
> spineplot(cases ~ income, col=c("peachpuff", "blue"))
> cdplot(cases ~ income, col=c("peachpuff", "blue"))
```

Both graphs (Figure 14) show that the proportion depressed declines with higher income. Note that the horizontal scales of the plots differ, as the bin widths of the spinogram are proportional to the number of cases in the interval, while the income scale in the conditional density plot is linear. The conditional density plot shows perhaps most clearly that the rate of depression remains approximately constant at a level of about 20% for incomes less than 30K. Above 30K the depression rate declines sharply with higher income. Both plots show the same pattern but because of the difference in horizontal scales the 30K threshold is situated farther to the right in the `spineplot()` than in the `cdplot()`.

We cannot conclude from this pattern that low income causes depression, as causality may well run in the other direction, with a depressed state causing a reduced income.

9.2 Inference

10 Probability Functions

10.1 Overview

R provides functions to calculate probabilities for a wide range of probability distributions. The functions are systematically named based on a "stem" corresponding to the specific distribution, e.g., `-binom` for binomial, `-norm` for normal, `-t` for Student t. For each distributions there are four functions defined by their prefix:

- `d-` probability P($X = x$) (discrete distribution) or probability density f(x) (continuous distribution)
- `p-` probability P($X \leq x$)
- `q-` quantile function: value of x such that P($X \leq x$) is equal to a given probability p

[3] The same `spineplot(y x)` expression is used to create a *spineplot* proper, when `x` and `y` are both categorical, and a *spinogram* when `x` is quantitative and `y` is categorical.

- `r`– generates random observations from the distribution

Examples are given below for the functions that are most used in the OLI curriculum.

10.2 Binomial Distribution

dbinom() The function `dbinom(x, size, prob)` returns $P(X = x)$, the probability of exactly x successes in a binomial experiment with `size` trials and probability `prob` of success.

- The probability of color-blindness among males is .08. Suppose one samples randomly 20 males from a large population. What is the probability that exactly 2 males (out of 20) are color-blind?

  Use `dbinom(x, size, prob)` with `x=2, size=20, prob=.08`.

```
> dbinom(2, 20, .08)
[1] 0.2710906
```

pbinom() The function `pbinom(x, size, prob)` returns $P(X \leq x)$, the probability of x or fewer successes in a binomial experiment with `size` trials and probability `prob` of success.

- In a sample of 20 males from a large population where the probability of color-blindness in males is .08, what is the probability that 2 males or fewer are color-blind (i.e., 0, 1, or 2 are color-blind)?

  Use `pbinom(x, size, prob)` with `x=2, size=20, prob=.08`.

```
> pbinom(2, 20, .08)
[1] 0.7879462
```

rbinom() The function `rbinom(n, size, prob)` generates `n` random observations from a binomial experiment with `size` trials and probability `prob` of success.

- Simulate the drawing of 10 random samples of 20 males from a large population where the probability of color-blindness among males is .08.

  Use `rbinom(n, size, prob)` with `n=10, size=20, prob=.08`.

```
> rbinom(10, 20, .08)
 [1] 0 3 0 2 3 1 2 0 1 2
```

  These results mean that the first random sample has 0 color-blind males, the second sample has 3, the third 0, etc.

10.3 Normal Distribution

pnorm() The function `pnorm(x, mean = 0, sd = 1, lower.tail = TRUE)` returns the probability $P(X \leq x)$ for a normally-distributed variable with mean `mean` and standard deviation `sd`. By default (when not specified) the mean is 0 and the standard deviation is 1, i.e. the distribution is standardized. By default `pnorm()` returns the lower tail probability $P(X \leq x)$, i.e. the area under the curve to the left of `x`. With the option `lower.tail=FALSE` `pnorm()` returns $P(X > x)$, i.e. the area under the curve to the right of `x`.

- Suppose that X is distributed as a standardized normal distribution (i.e., with mean=0 and sd=1).

  What is $P(X \leq -1.65)$?

  Use `pnorm(x)` with `x=-1.65`.

```
> pnorm(-1.65)
[1] 0.04947147
```

See Figure 15, Panel a.

- For X standard normal, what is P($X > 2$)?

  Use `pnorm(x)` with the option `lower.tail=FALSE`.

```
> pnorm(2.0, lower.tail=FALSE)
[1] 0.02275013
```

  See Figure 15, Panel b.

- For X standard normal, what is P($-0.8 \leq X \leq 1.5$)?

  To find the probability that X is in an interval (x1, x2), subtract the value of `pnorm()` for x1 (lower bound) from the value of `pnorm()` for x2 (upper bound).

```
> pnorm(1.5) - pnorm(-0.8)
[1] 0.7213374
```

  See Figure 15, Panel c.

- For X standard normal, what is P($|X| > 1.723$)?

```
> 2*pnorm(1.723, lower.tail=FALSE)
[1] 0.08488853
```

  See Figure 15, Panel d. This is the way to compute the p-value for a two-tailed (or two-sided) z-test. The value returned by `pnorm()` is multiplied by two because $|X| > 1.723$ refers to both values greater than 1.723 and values less than -1.723.

- According to Wikipedia British musician Brian Jones, founder of the Rolling Stones, had an IQ of 135.

  Assuming that IQ is normally distributed with mean=100 and sd=15, what is the proportion of the population with IQs less than or equal to 135?

  Use `pnorm()` specifying the values of `mean` and `sd`.

```
> pnorm(135, 100, 15)
[1] 0.9901847
```

  Thus Brian Jones' IQ was at the 99th percentile.[4]

qnorm() The function `qnorm(p, mean=0, sd=1, lower.tail = TRUE)` returns the value of x such that P($X \leq x$) is equal to the given probability p. `qnorm()` is called the *quantile function* (hence the q- prefix) and is the inverse of `pnorm()`. Options to specify `mean`, `sd` and `lower.tail` work the same way as for `pnorm()`.

- What is the value of a standardized normal variable X such that P($X \leq x$) is .05? (This is the critical value we would want to calculate a 10% confidence interval).

  Use `qnorm(p)` with p=.05.

[4] Talent and intelligence all went to waste through alcohol and drugs addiction, and Jones died at the age of 27.

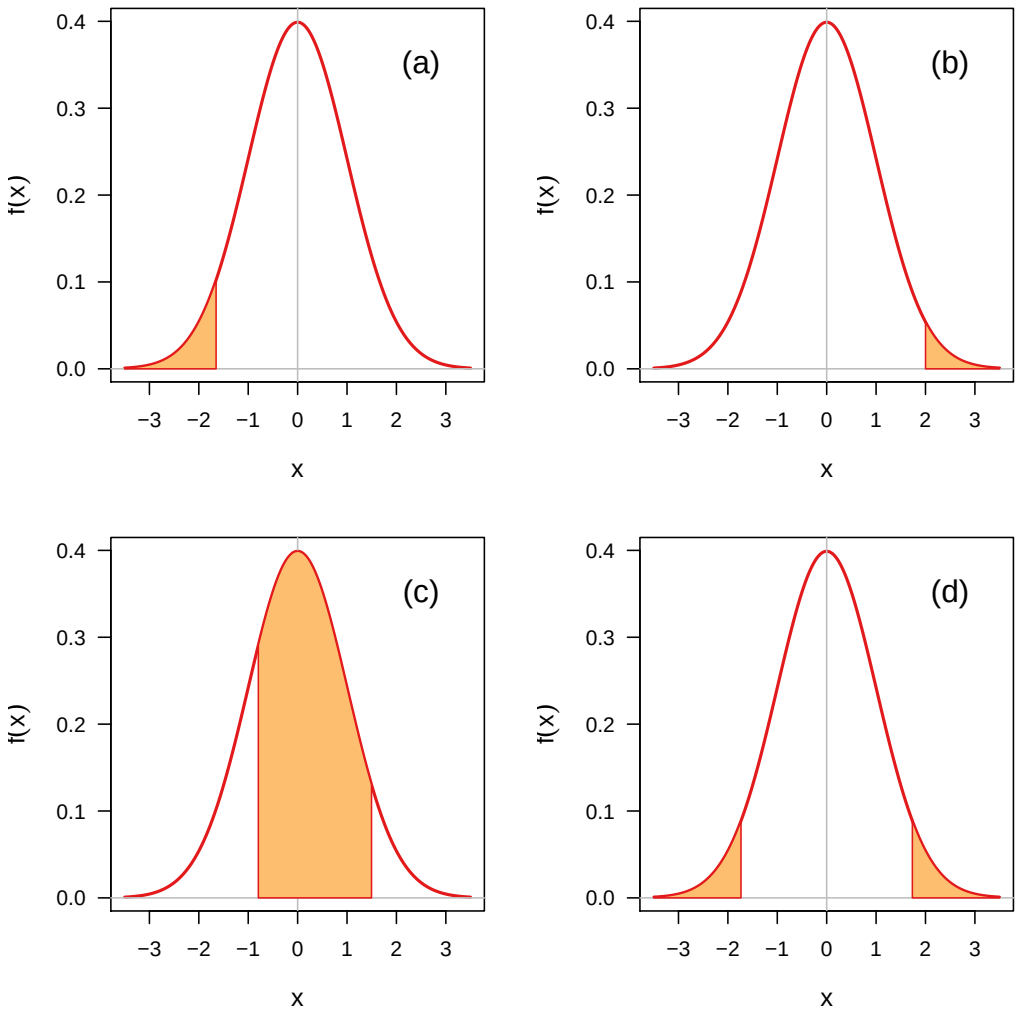

Figure 15: Four common uses of `pnorm()`. (a) P($X \leq -1.65$): `pnorm(-1.65)` `[0.04947147]`; (b) P($X > 2$): `pnorm(2.0, lower.tail=FALSE)` `[0.02275013]`; (c) P($-0.8 \leq X \leq 1.5$): `pnorm(1.5) - pnorm(-0.8)` `[0.7213374]`; (d) P($|X| > 1.723$): `2*pnorm(1.723, lower.tail=FALSE)` `[0.08488853]`

```
> qnorm(.05)
[1] -1.644854
```

We use 1.644854 as the critical value, ignoring the negative sign.

- Suppose the time to complete a 200-meter backstroke swim for female competitive swimmers is normally distributed with a mean $\mu = 141$ seconds and a standard deviation $\sigma = 7$ seconds.

  Suppose the fastest 6% of female swimmers in the nation are offered college scholarships. In order to be given a scholarship, a swimmer must complete the 200-meter backstroke in no more than how many seconds? Use `qnorm(.06)`, specifying the mean and sd of backstroke time.

  ```
  > qnorm(.06, 141, 7)
  [1] 130.1166
  ```

  or 130 seconds maximum.

- Suppose scores on an exam are normally distributed with a mean of 75 points and a standard deviation of 8 points.

  Suppose that the top 4% of the exams will be given an A+. In order to be given an A+, an exam must earn at least what score?

  ```
  > qnorm(.04, 75, 8, lower.tail=FALSE)
  [1] 89.00549
  ```

  Thus scores 89 points or more get an A+. We use the `lower.tail=FALSE` option because what we want here is the value of X corresponding to an upper-tail probability of 4%. Compare with the previous example (backstroke time), where the probability referred to the lower tail of the distribution.

rnorm() The function `rnorm(n, mean = 0, sd = 1)` generates `n` random observations from a normal distribution with mean `mean` and standard deviation `sd`. By default (when mean and sd are not specified) the distribution is standardized normal.

- Simulate drawing a sample of 10 from a normally distributed population of IQs with mean=100 and sd=15.

  Use `rnorm(n, mean, sd)`, specifying the relevant parameters.

  ```
  > rnorm(10, 100, 15)
   [1]  97.52691  83.53285  68.84773 102.95186 108.13373 111.73588  63.80832
   [8] 115.82666 116.83199 111.01693
  ```

10.4 Student *t* Distribution

Members of the Student *t* family of distributions are symmetric around a mean of zero and similar in shape to a normal distribution. *t* distributions differ from the normal in having thicker tails, especially for small degrees of freedom (df). As the df increase the *t* distribution converges quickly to a standard normal, so by df=30 the *t* and the standard normal are equivalent for most purposes (Figure 16, Panel a).

Probability functions for the Student *t* distribution work the same way as the for the normal, except that df has to be specified (and `mean` and `sd` may not be). In particular, the `lower.tail` option works the same as for the normal.

pt() The function `pt(x, df, lower.tail = TRUE)` returns the probability $P(X \leq x)$ for a t-distributed variable with `df` degrees of freedom. `df` has to be specified (there is no default). By default `pt()` returns the lower tail probability $P(X \leq x)$, i.e. the area under the curve to the left of `x`. With the option `lower.tail=FALSE` `pt()` returns the upper tail probability $P(X > x)$, i.e. the area under the curve to the right of `x`.[5]

- Suppose that X is distributed as a Student t distribution with 7 df.

  What is $P(X \leq -1.65)$?

  Use `pt(x, df)` with `x=-1.65` and `df=7`.

  ```
  > pt(-1.65, 7)
  [1] 0.07146457
  ```

  See Figure 16, Panel b.

- For X distributed as t with 7 df, what is $P(X > 2)$?

  Use `pt(x, df)` with the option `lower.tail=FALSE`.

  ```
  > pt(2.0, 7, lower.tail=FALSE)
  [1] 0.04280966
  ```

  See Figure 16, Panel c.

- For X distributed as t with 7 df, what is $P(|X| > 1.723)$?

  ```
  > 2*pt(1.723, 7, lower.tail=FALSE)
  [1] 0.1285541
  ```

  See Figure 16, Panel d. This is the way to compute the p-value for a two-tailed (or two-sided) t-test. The value returned by `pnorm()` is multiplied by two because $|X| > 1.723$ refers to both values greater than 1.723 and values less than -1.723.

qt() The function `qt(p, df, lower.tail = TRUE)` returns the value of x such that $P(X \leq x)$ is equal to the given probability p. `qt()` is called the *quantile function* (hence the `q-` prefix) and is the inverse of `pt()`. Degrees of freedom `df` have to be specified. The option `lower.tail` works the same way as for `pt()`.

- What is the value of a variable X that is t-distributed with 7 df such that $P(X \leq x)$ is .05? (This is the critical value we would want to calculate a 10% confidence interval).

  Use `qt(p, df, lower.tail=FALSE)` with `p=.05` and `df=7`.

  ```
  > qt(.05, 7, lower.tail=FALSE)
  [1] 1.894579
  ```

  We use the `lower.tail=FALSE` option to obtain a positive critical value.

[5] Returning the upper-tail probability is often the default for probability functions in statistical software, a survival of the usage of printed tables, which showed upper-tailed values only to save space. Use of the `lower.tail=FALSE` option emulates that behavior.

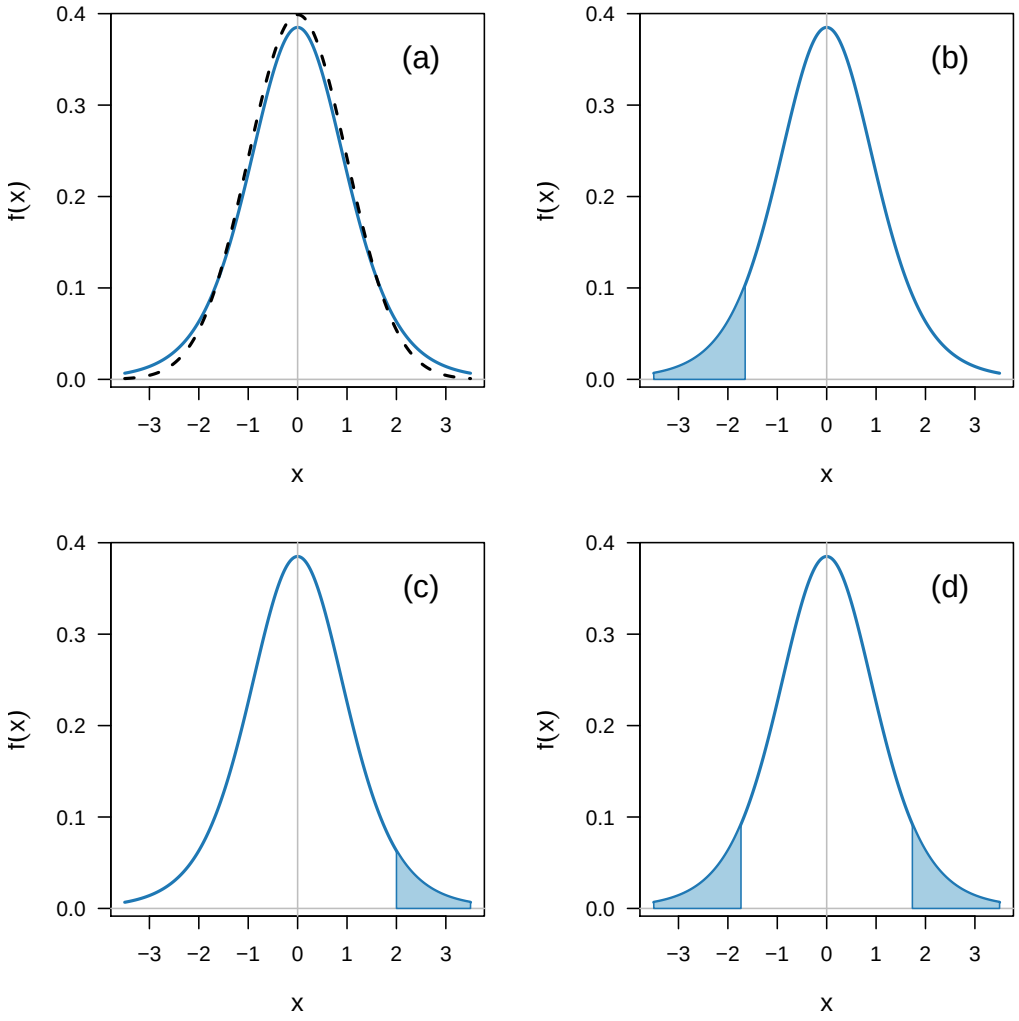

Figure 16: Student t distribution and three uses of `pt()`. (a) Student t distribution with 7 df (solid line) and standard normal distribution (dashed line) compared; (b) $P(X \leq -1.65)$: `pt(-1.65, 7)` `[0.07146457]`; (c) $P(X > 2)$: `pt(2.0, 7, lower.tail=FALSE)` `[0.04280966]`; (d) $P(|X| > 1.723)$: `2*pt(1.723, 7, lower.tail=FALSE)` `[0.1285541]`

rt() The function `rt(n, df)` generates `n` random observations from a Student t distribution with `df` degrees of freedom.

- Simulate drawing a sample of 10 from a t distribution with 7 df.

  Use `rt(n, df)`, specifying the relevant parameters.

```
> rt(10, 7)
 [1]  0.5733087 -0.5408957  0.3988082 -0.2569938 -0.7685501  0.2137492
 [7]  1.3653922 -0.8188979 -0.5634982 -1.3526420
```

11 Appendix A: Functions for Percentaging Tables

From John Fox's **Rcmdr** package. Instructions: copy listings for both functions and paste at the R prompt; then call as `rowPercents(tab)` (row percentages add up to 100%) or `colPercents(tab)` (column percentages add up to 100%), where `tab` is a contingency table.

```
rowPercents function (tab, digits = 1)
{
    dim <- length(dim(tab))
    if (dim == 2)
        return(t(colPercents(t(tab), digits = digits)))
    tab <- aperm(tab, c(2, 1, 3:dim))
    aperm(colPercents(tab, digits = digits), c(2, 1, 3:dim))
}

colPercents function (tab, digits = 1)
{
    dim <- length(dim(tab))
    if (is.null(dimnames(tab))) {
        dims <- dim(tab)
        dimnames(tab) <- lapply(1:dim, function(i) 1:dims[i])
    }
    sums <- apply(tab, 2:dim, sum)
    per <- apply(tab, 1, function(x) x/sums)
    dim(per) <- dim(tab)[c(2:dim, 1)]
    per <- aperm(per, c(dim, 1:(dim - 1)))
    dimnames(per) <- dimnames(tab)
    per <- round(100 * per, digits)
    result <- abind(per, Total = apply(per, 2:dim, sum), Count = sums,
        along = 1)
    names(dimnames(result)) <- names(dimnames(tab))
    result
}
```

12 R Bugs and Problems with OLI Probability and Statistics

Java Applets Blocked

A problem that emerged in Spring 2015 is that some applets do not work. So far the problem has been reported for:

- Measures of center applet in Module 1 (in Learn by Doing, p.21).
- Birthday paradox applet in Module 5 (in Learn by Doing, bottom p.92).
- Binomial distribution demonstration in Module 8 (in Learn by Doing, p.144).

If you experience this problem on your system it may be because Java blocks the sites where applets are located because of stricter security standards in newer (8+) versions of Java.

A workaround is to add the site where the applet is located to the Java exception list. To do this on my system (Windows 7) I do the following.

1. Click the Start button (in lower left corner of the screen) and then All Programs.
2. Scroll down the list to the Java folder and click it.
3. Click on Configure Java.
4. Click on the Security tab.
5. Click on Edit Site List.
6. Click on Add.
7. Click on the field that appears and enter `http://statweb.stanford.edu/`
8. Click OK.
9. Click on Add again, enter `https://statweb.stanford.edu/` and click OK.
10. Click on Add again, enter `http://www.stat.tamu.edu/` and click OK.
11. Click on Add again, enter `https://oli.cmu.edu/` and click OK.
12. Click OK again to leave Configure Java.
13. You may need to leave and then reopen your browser for the changes to take effect.

The procedure for the Mac should be similar.

Module 2 – Learn by Doing p.55

This is the Learn by Doing on using simple regression analysis for winning time in the Olympic Game for the 1,500 meters race as a function of the year of the game.

You will need to add a ";" at the end of the second plot command (or run the command on a separate line), like this:

```
plot(olym$Year[olym$Year!=1896], olym$Time[olym$Year!=1896]);
```

Here is the full set of commands for this activity. Run each block of commands in turn. Consider clicking History → Recording in the graphic window after the first plot is completed to keep it in memory so you can recall it (using PageUp in the graphic window) to compare it with the second plot later.

```
# scatterplot of winning time against Olympic Game year
plot(olym$Year, olym $Time)

# calculate regression and draw regression line
L = lm(olym$Time~olym$Year);
abline(L);
```

```
# add a legend with regression function to plot
cf=coefficients(L);
legend(1940,260,legend=paste("time = ",round(cf[1],0),round(cf[2],2),"year"))

# redraw the plot and recalculate regression after removing outlier
# I have added a ";" at end of the plot command
plot(olym$Year[olym$Year!=1896], olym$Time[olym$Year!=1896]);
L = lm(olym$Time[olym$Year!=1896]~olym$Year[olym$Year!=1896]);
abline(L);
cf=coefficients(L);
legend(1950,240,legend=paste("time = ",round(cf[1],0),round(cf[2],2),"year"))
```

Module 3 – Learn by Doing p.67

This is the Learn by Doing on sampling. It is a comparison of a random sample from a population of students with a subset of the population that is not random (the subset of business students).

First, there is a problem with the OLI instruction

```
summary(population$Verbal) summary(random_sample$Verbal)
```

When you copy these two lines, they appear as one long line. You need to enter the two summary commands separately, or add a semicolon (;) between the two. One way that will certainly work is like this

```
summary(population$Verbal);
summary(random_sample$Verbal)
```

Second, the commands required are intricate, with a number of changes of variables. Here is a script with all the commands you need for the activity. Copy and paste the commands one block at a time, not all at once. Don't copy across the page number and the running head of this document as R will flag this as an error.

Optionally, consider clicking History → Recording in the graphic window after drawing the first pair of piecharts. This will keep the graphs in memory to facilitate comparisons later on. After you have created other graphs you can move back and forth among them using the PageUp and PageDown keys in the graphic window.

```
# select random sample from population
random_sample = population[sample(1:length(population$Course),192),];
head(random_sample)

# random sample and population compared
# side-by-side pie charts for handedness
random_sample_percent = 100*summary(random_sample$Handed)/length(random_sample$Handed);
random_sample_percent;
pop_percent = 100*summary(population$Handed)/length(population $Handed);
pop_percent;
par(mfrow=c(1,2));
pie(pop_percent,labels=paste(c("left=","right="),round(pop_percent,0),"%"),main="Population");
pie(random_sample_percent,labels=paste(c("left=","right="),round(random_sample_percent,0),"%"),
  main="Random Sample");

# side-by-side pie charts for sex
random_sample_percent = 100*summary(random_sample$Sex)/length(random_sample$Sex);
random_sample_percent;
```

```
pop_percent = 100*summary(population$Sex)/length(population $Sex);
pop_percent;
par(mfrow=c(1,2));
pie(pop_percent,labels=paste(c("female=","male="),round(pop_percent,0),"%"),main="Population");
pie(random_sample_percent,labels=paste(c("female=","male="),round(random_sample_percent,0),"%"),
  main="Random Sample");

# summaries for verbal (SAT)
# semicolon missing in original instructions after the first summary command
summary(population$Verbal);
summary(random_sample$Verbal)

# summaries for age (SAT)
summary(population$Age);
summary(random_sample$Age)

# create business students subset
business = population[population$Course=="Business",];
head(business)

# copy the entire block above for pie charts (Handed and Sex) and summaries (Verbal and Age)

# change "random_sample" to "business" and "Random Sample" to "Business Students"
# business students and population compared
# side-by-side pie charts for handedness
business_percent = 100*summary(business$Handed)/length(business$Handed);
business_percent;
pop_percent = 100*summary(population$Handed)/length(population $Handed);
pop_percent;
par(mfrow=c(1,2));
pie(pop_percent,labels=paste(c("left=","right="),round(pop_percent,0),"%"),main="Population");
pie(business_percent,labels=paste(c("left=","right="),round(business_percent,0),"%"),
  main="Business Students");

# side-by-side pie charts for sex
business_percent = 100*summary(business$Sex)/length(business$Sex);
business_percent;
pop_percent = 100*summary(population$Sex)/length(population $Sex);
pop_percent;
par(mfrow=c(1,2));
pie(pop_percent,labels=paste(c("female=","male="),round(pop_percent,0),"%"),main="Population");
pie(business_percent,labels=paste(c("female=","male="),round(business_percent,0),"%"),
  main="Business Students");

# summaries for verbal (SAT)
# semicolon missing in original instructions after the first summary command
summary(population$Verbal)
summary(business$Verbal)

# summaries for age (SAT)
summary(population$Age)
summary(business$Age)
```

```
# optional: side-by-side boxplots for Verbal
# use "par" command to go back to one graph per window
par(mfrow=c(1,1))
boxplot(Verbal ~ Course, data=population, notch=TRUE)
```

Module 8 – Learn by Doing p.143

This is the Learn by Doing on Binomial Random Variables. OLI suggests calculating binomial probabilities using the formulas

```
dbinom(k, n, p) = P(X=k)
pbinom(k, n, p) = P(X<=k)
```

but R reports an error when you enter them.

The problem is that R does not understand these statements, as they are *symbolic representations* of the meaning of the `dbinom` and `pbinom` functions. For the formulas to work in R the symbols have to be replaced by their actual (numerical) values. So in the context of this problem (where n=10, k=4 and p=.2) you should enter instead

```
dbinom(4, 10, .2)
pbinom(4, 10, .2)
```

See if that works for you.

Module 11 – Learn by Doing p.186

In the *first* of *two* Learn by Doing on Confidence Intervals for the Population Mean on p.186 in Module 11 you are invited to define a `conf.int()` function by copying a block of commands and entering it in R.

R on Emily's Mac reported an error when she entered the function definition directly copied from OLI. This seems to be due to R not "seeing" the newlines at the end of the lines and therefore not knowing when one command ends and the other begins.

A successful workaround for Emily was to create a new script in R with File -> New document, paste the block of commands in the script, and making sure each line was ended by a semicolon (;). The block of commands now looked like this.

```
conf.int = function(x,sigma,c) {
n = length(x);
m = mean(x);
me = qnorm((1+c)/2)*sigma/sqrt(n);
print.noquote(paste("sample size: ",n," sample mean: ",m));
print.noquote(paste(c*100,"% confidence interval: (",m-me,", ",m+me,")"));
}
```

She then pasted the new block at the R prompt, and R now accepted the input. Afterwards the command

```
conf.int(birthweight$birthweight,500,0.99)
```

successfully executed.

You can modify the block from OLI yourself or use the already modified block above.