

Utah State University

DigitalCommons@USU

All Graduate Plan B and other Reports

Graduate Studies

5-2012

Ignoring the Spatial Context in Intro Statistics Classes - and some Simple Graphical Remedies

Nathan Voge
Utah State University

Follow this and additional works at: <https://digitalcommons.usu.edu/gradreports>



Part of the [Mathematics Commons](#), and the [Statistics and Probability Commons](#)

Recommended Citation

Voge, Nathan, "Ignoring the Spatial Context in Intro Statistics Classes - and some Simple Graphical Remedies" (2012). *All Graduate Plan B and other Reports*. 327.

<https://digitalcommons.usu.edu/gradreports/327>

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Plan B and other Reports by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.



IGNORING THE SPATIAL CONTEXT IN INTRO STATISTICS CLASSES –
AND SOME SIMPLE GRAPHICAL REMEDIES

by

Nathan Voge

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Statistics

Approved:

Dr. Jürgen Symanzik
Major Professor

Dr. Cristopher Corcoran
Committee Member

Dr. Kady Schneiter
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah

2012

ABSTRACT

Ignoring the Spatial Context in Intro Statistics Classes – And Some Simple
Graphical Remedies

by

Nathan Voge, Master of Science

Utah State University, 2012

Major Professor: Dr. Jürgen Symanzik
Department: Mathematics and Statistics

Statistical data often have a spatial (geographic) context, be it countries of the world, states in the US, counties within a state, cities across the globe, or locations where measurements have been taken. However, most introductory statistics books do not even suggest that such data often are not independent from location, but rather are effected by some spatial association. Remedies are simple: Display data via various map views and briefly discuss which additional information can be extracted from such a graphical representation. In this report, we will visit a variety of popular introductory statistics textbooks and show how some of the data used in examples and exercises can be initially displayed via various map views, such as choropleth maps or micromaps. Students familiar with R should be able to create similar map displays by themselves via several R packages.

(72 pages)

ACKNOWLEDGMENTS

- Figures 6, 7, 9, 10, 12-15, 17, 18, 20, 21, 23, and 24 were created in R Version 12.2.2 (figures also compatible with current R Version 2.15.0, *R Development Core Team* (2012)) with R packages: maptools (See *Lewin-Koh et al.* (2012)), Rgooglemaps (See *Loecher* (2012)), maps, RColorBrewer (See *Neuwirth* (2011)), and PBSmapping (See *Schnute et al.* (2010)).
- Figure 21 and 24 are based on micromap templates from Michael Minnotte and Daniel B. Carr, respectively.
- Latex template was provided by Mark Schmelter.
- I would like to deeply thank my masters advisor Dr. Jürgen Symanzik for skillfully guiding me along this long and sometimes difficult process. His cheerful and encouraging attitude as made this process much more bearable and at times quite enjoyable! I would like to also thank all my cohorts in cell block 10 (also known as the basement of the Lund Bldg). I have enjoyed their interactions and the sharing of the pain and triumph experienced earning this degree! I would like to acknowledge the faculty and staff of the Math and Stat Dept here at USU. Thank you Meredith, Cindy, and Nancy! You guys are awesome and have been so helpful and great to work with. I would also like to thank all the other faculty in the department for their care, example, and dedication to teaching and research. Finally, I would like to thank my friends, parents, and family. Thank you for your encouragement and support and especially for giving me a welcome haven away from statistics!

CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGMENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
1 INTRODUCTION	1
2 THE FOUR TEXTBOOKS	3
2.1 Probability and Statistics for Engineers and Scientists	3
2.2 Introduction to the Practice of Statistics	4
2.3 Essentials of Business Statistics	6
2.4 Statistics	7
3 EXAMPLES OF SPATIAL ASSOCIATION	9
3.1 Choropleth Maps	9
3.1.1 Mortgage Delinquency Rates	9
3.1.2 CO ₂ Emissions	11
3.2 Google Maps	13
3.2.1 Public University Tuition	13
3.2.2 Major League Baseball Team Valuations and Revenue	17
3.3 Micromaps	19
3.3.1 Damage Due to Tornadoes	19
3.3.2 Fruit and Vegetable Consumption and Smoking	22
4 DISCUSSION	26
REFERENCES	27
APPENDICES	30
APPENDIX A ADDITIONAL SPATIAL EXAMPLES FROM TEXTBOOKS	31
A.1 Introduction to the Practice of Statistics	31
A.2 Essentials of Business Statistics	32
A.3 Statistics	34
APPENDIX B R CODE	35
B.1 Choropleth Map and Stem and Leaf Plot: Mortgage Delinquency	35
B.2 World Choropleth Map: CO ₂ Emissions	36
B.2.1 Histogram: CO ₂ Emissions	38

B.3	Rayglyph Googlemap: University Tuition	38
B.3.1	Scatterplot and Residual Plot: University Tuition	45
B.4	Rayglyph Googlemap: MLB Value vs Revenue	46
B.4.1	Histogram: MLB Value and Revenue	53
B.5	One-Panel Linked Micromap: Tornado Damage	54
B.5.1	Tornado Histogram	57
B.6	Two-Panel Linked Micromap: Fruit/Vegetable Consumption and Smoking	58

LIST OF TABLES

Table		Page
1	Table from <i>Moore et al.</i> (2009) book containing spatial examples where a map display seems beneficial, but was not attempted in this report. The parentheses following its description is the exercise number or example found in the book. (1.28) refers to chapter one, exercise 28 etc. (Ex. 1.13) refers to Example 1.13. “n” is the sample size and “p” is the number of variables.	31
2	Table from <i>Moore et al.</i> (2009) book containing spatial examples that are too small and/or are categorical and would not benefit from a map. See Table 1 for additional details.	31
3	Table from <i>Bowerman et al.</i> (2009) book containing spatial examples where a map display seems beneficial, but was not attempted in this paper. See Table 1 for additional details.	32
4	Table from <i>Bowerman et al.</i> (2009) book containing spatial examples that are too small and/or are categorical and would not benefit from a map. See Table 1 for additional details.	33
5	Table from <i>Freedman et al.</i> (2007) book containing spatial examples where a map display seems beneficial, but was not attempted in this paper.	34

LIST OF FIGURES

Figure		Page
1	Illustration showing a map of the Nile River damming system and water flow control. (<i>Hayter</i> , 2006, p. 537). [From HAYTER. <i>Probability and Statistics for Engineers and Scientists (with CD-ROM)</i> , 3E. ©2007 Brooks/Cole, a part of Cengage Learning, Inc. Reproduced by permission. www.cengage.com/permissions]	3
2	State mean SAT scores plotted against the percent of high school seniors in each state who take the SAT exams. The data is separated into northeastern states (plot symbol “e”) and the mid-western states (plot symbol “m”). (<i>Moore et al.</i> , 2009, p. 89). [From <i>Introduction to the Practice of Statistics</i> (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]	5
3	Illustration of pipelines and cities served by the Columbia Gas System. The yellow dots look like they might be data points, but on closer inspection, they are just locations of storage fields. [From <i>Essentials of Business Statistics</i> (third edn) by B. L. Bowerman, R. T. OConnell, J. B. Orris, and E. S. Murphree. ©2009 by McGraw-Hill/Irwin. Used with permission.] (<i>Bowerman et al.</i> , 2009, p. 500).	6
4	Map of a sampling design for the 1995 population survey (<i>Freedman et al.</i> , 2007, p. 397). [From <i>STATISTICS</i> 4th edition by David Freedman, Robert Pisani, and Roger Purves. Copyright ©2007, 1998, 1991, 1991, 1978 by W. W. Norton & Company, Inc. Used by permission of W.W. Norton & Company, Inc.]	7
5	Original data set of mortgage delinquency rates (<i>Bowerman et al.</i> , 2009, p. 74). [From <i>Essentials of Business Statistics</i> (third edn) by B. L. Bowerman, R. T. OConnell, J. B. Orris, and E. S. Murphree. ©2009 by McGraw-Hill/Irwin. Used with permission.]	10
6	Stem-and-leaf plot of the mortgage data	10
7	Choropleth map of the mortgage delinquency rates. D.C., Alaska, and Hawaii were not included on the map.	11

8	Original data set of CO ₂ emissions per person from countries with at least 20 million people (<i>Moore et al.</i> , 2009, p. 26). [From <i>Introduction to the Practice of Statistics</i> (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]	12
9	Histogram of CO ₂ data.	12
10	Choropleth map of the CO ₂ emissions data (in metric tons per person).	13
11	Original data set of in-state undergraduate tuition and fees for 32 universities between 2000 and 2005 (<i>Moore et al.</i> , 2009, p. 595). [From <i>Introduction to the Practice of Statistics</i> (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]	14
12	Scatterplot of the tuition data.	14
13	Google map showing the percent increase of tuition from 2000 to 2005.	15
14	Google map adding ray-glyph lines to the percent increase dots of the previous map (Figure 13).	16
15	Residual plot of tuition data.	16
16	Original data set of franchise value and 2006 revenues for each of the 30 major league baseball (MLB) teams as reported by Forbes magazine and Forbes.com. (<i>Bowerman et al.</i> , 2009, p. 67). [From <i>Essentials of Business Statistics</i> (third edn) by B. L. Bowerman, R. T. OConnell, J. B. Orris, and E. S. Murphree. ©2009 by McGraw-Hill/Irwin. Used with permission.]	17
17	Histogram of the MLB data.	18
18	Google map with ray-glyph lines showing Major League Baseball team revenues and values. Teams that were located in close proximity were manually moved to fix over-plotting.	18
19	Original data set of average property damage due to tornadoes from 1950 to 1999, adjusted for inflation (<i>Moore et al.</i> , 2009, p. 25). [From <i>Introduction to the Practice of Statistics</i> (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]	20
20	Histogram of the tornado data.	20

21	Micromap of the tornado data. This map is for the continental U.S. only and does not show data for Hawaii, Alaska, and Puerto Rico. The right column gives us a color-coded dot plot that corresponds to a state in the middle column that is plotted on a map in the column on the left. States with an average property damage above the median are shaded grey in the five small maps above the separating line while states with an average property damage below the median are shaded in grey in the five small maps below the separating line.	21
22	Original data set of fruit and vegetable consumption and daily smoking for all 50 states (<i>Moore et al.</i> , 2009, p. 161-162). [From <i>Introduction to the Practice of Statistics</i> (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]	23
23	Scatterplot and regression line related to the fruit and vegetable and smoking data set (<i>Moore et al.</i> , 2009, p. 162).	23
24	Micromap of fruit and vegetable consumption and smoking data. . .	25

CHAPTER 1

INTRODUCTION

A picture is worth a thousand words. Although this saying is usually attributed to paintings and art, it is highly applicable to data analysis. From initial exploration of a data set to the final presentation of results to the end user, statistical graphics play a vital role in shaping the understanding of our data. Through proper use of graphics we can make critical discoveries and communicate them clearly. Graphics are often a necessary step before conducting an analysis. Linear regression relies completely on the data being linear. We determine this through the use of a scatterplot. In fact, anyone who has ever seen the Anscombe data set (*Anscombe* (1973)) would agree that it is essential to look at the data first, at least via a scatterplot, before conducting any linear regression analysis. Moreover, plots are often used as the primary source to confirm normality and constant variance.

Conversely, poor use or misuse of graphics can seriously mislead the intended audience (by accident or design). The same is true for spatial data. Data in a spatial context is highly dependent on location and should be viewed on a map before any analysis is performed. *Wainer* (1997) lists twelve rules on how to display data badly. Specifically, his 5th rule, “Graph data out of context” (p. 25) and his 2nd rule “Hide what data you do show” (p. 16) are applicable to spatial data. “Graph data out of context” would basically mean that we remove data from their natural geographic environment and “hide what data you do show” would mean that we are hiding parts of the data, in this case, the spatial locations.

In this report, we looked at the four textbooks used in introductory statistics courses at Utah State University (USU) to determine how well these books deal with

the spatial components of the data sets used in the books. The textbooks we analyzed are:

1. Probability and Statistics for Engineers and Scientists by A. Hayter (*Hayter* (2006))
2. Introduction to the Practice of Statistics by D. S. Moore, G. P. McCabe, and B. Craig (*Moore et al.* (2009))
3. Essentials of Business Statistics by B. L. Bowerman, R. T. O'Connell, J. B. Orris, and E. S. Murphree (*Bowerman et al.* (2009))
4. Statistics by D. Freedman, R. Pisani, and R. Purves (*Freedman et al.* (2007))

For each book, we took an in-depth look at the data presented and we identified data sets with a spatial component. A portion of these data sets were then selected and graphs representing the data were produced. Multiple ways to display data using a map are presented. Some books provided websites where a link to the data was found. Otherwise, data was entered manually.

In the remainder of this report, we took a closer look at the four textbooks in Chapter 2. In Chapter 3, we looked at data sets from these textbooks where we expected some spatial association and described via various maps how this spatial association can be easily revealed. We finish with a brief discussion in Chapter 4. Appendix A provides an overview of additional spatial examples from the four textbooks. Appendix B contains the R code to reproduce all figures presented in this report. This report is an extended version of *Voge and Symanzik* (2011).

CHAPTER 2

THE FOUR TEXTBOOKS

2.1 Probability and Statistics for Engineers and Scientists

In *Hayter* (2006), there weren't any examples of spatial data. This can be explained by the fact that most of the examples were related to data from engineering, manufacturing and medical backgrounds.

In Figure 1, we see our first example of a map in a textbook. The Aswan High Dam, situated at the northern end of Lake Nasser, controls the flow of water into the fertile regions bordering the Nile river in Egypt. The proper control of the water

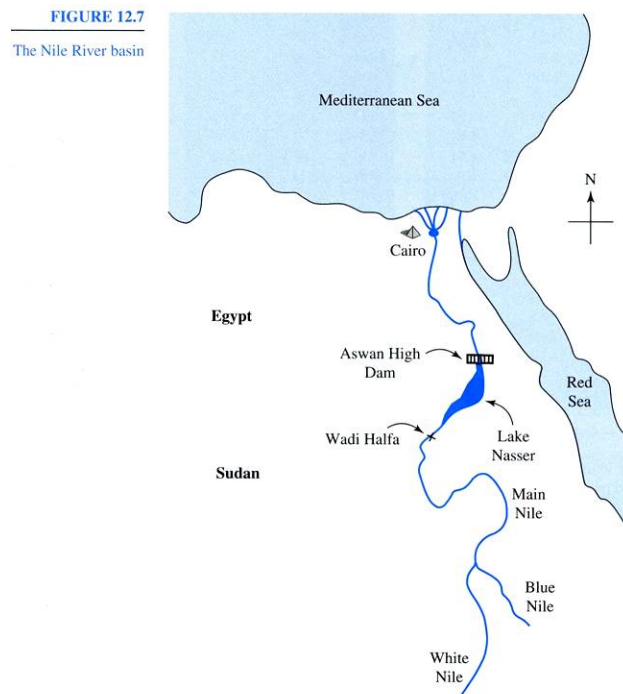


Fig. 1: Illustration showing a map of the Nile River damming system and water flow control. (*Hayter*, 2006, p. 537). [From HAYTER. *Probability and Statistics for Engineers and Scientists (with CD-ROM)*, 3E. ©2007 Brooks/Cole, a part of Cengage Learning, Inc. Reproduced by permission. www.cengage.com/permissions]

passing through the dam is critical to the well-being of agriculture, industry, and population centers. The amount of water reserves in Lake Nasser depends upon the flow of water into the lake at Wadi Halfa. Egyptian engineers are able to predict the future Nile River flowrates at Wadi Halfa by measuring its monthly inflow. Figure 1 solely illustrates this water control system. The data used in this example is concerned with the inflow at Wadi Halfa over the past 100 years and is not related to the map. It would be interesting to use this map to compare the flowrates at different locations along the Nile River. But this map serves just to motivate the example and not to display data.

2.2 Introduction to the Practice of Statistics

Moore et al. (2009) contained seven “major” data sets with spatial components. Four of these were further discussed in this report. The three excluded examples are “Literacy Rates (percent) in Islamic Nations” (p. 10), “Park Space and Population” (p. 78), and “NBA Teams as Businesses” (p. 98). More information about these data sets can be found in Table 1 in Appendix A.1. In addition to the “major” data sets, *Moore et al.* (2009) also contained several “minor” data sets with just a few, sometimes as little three to five, observations presented with spatial locations (See Table 2 in Appendix A.1). For such “minor” data sets, maps probably won’t provide any further insights. Data were mainly from environmental, economic, social, and health backgrounds.

On page 89, *Moore et al.* (2009) actually came across the issue of spatial dependence while looking at SAT scores. They included a scatterplot of mean SAT scores vs percent taking the SAT (see Figure 2) that further categorized states into a Northeast group and a Midwest group. The graph clearly showed a strong separation between these two regions. This would be an excellent opportunity to introduce spatial de-

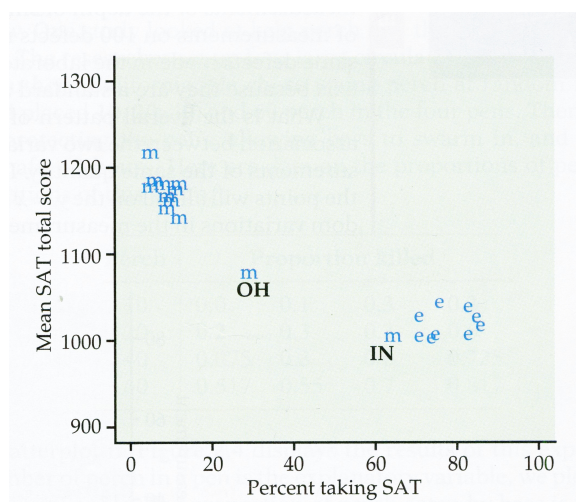


Fig. 2: State mean SAT scores plotted against the percent of high school seniors in each state who take the SAT exams. The data is separated into northeastern states (plot symbol “e”) and the mid-western states (plot symbol “m”). (*Moore et al.*, 2009, p. 89). [From *Introduction to the Practice of Statistics* (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]

pendence and some map items. In fact, this book already has a section designated to introduce spatial dependence. The book has “Challenge Exercises” that are used in the book to better challenge the student. Some exercises are mathematical and some require an open-ended investigation, etc. A statement about spatial association could be added here. Extra information about spatial dependence and more views could also be added to the book’s “Supplemental Material” section on the book’s website (<http://www.whfreeman.com/ips6e>).

Note: There is a seventh edition of this book (*Moore et al.* (2012)) that is currently being used in statistics classes. It was also reviewed (at a later date) and no such maps were found in this latest edition.

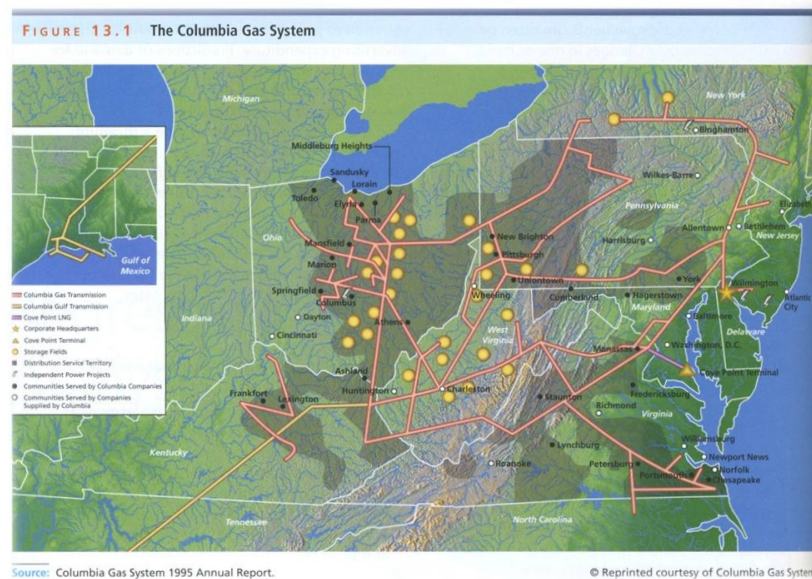


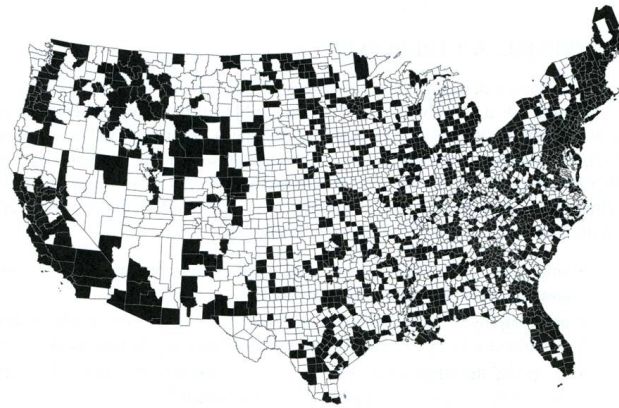
Fig. 3: Illustration of pipelines and cities served by the Columbia Gas System. The yellow dots look like they might be data points, but on closer inspection, they are just locations of storage fields. [From *Essentials of Business Statistics* (third edn) by B. L. Bowerman, R. T. OConnell, J. B. Orris, and E. S. Murphree. ©2009 by McGraw-Hill/Irwin. Used with permission.] (*Bowerman et al.*, 2009, p. 500).

2.3 Essentials of Business Statistics

Bowerman et al. (2009) contained five major data sets with spatial components. Two of these are further discussed in this report. The excluded examples are “Housing Affordability in Texas” (p. 134), “New Jersey Bank Data” (p. 560), and “Accounting Rates” (p. 559) (See Table 3 in Appendix A.2). *Bowerman et al.* (2009) also contained several “minor” data sets where maps probably won’t provide any further insights (summarized in Table 4 in Appendix A.2). Data were mainly from economic and social backgrounds.

In Figure 3 we see another example of a map in a textbook. There is shading and even different kinds of markers on the map, but to our disappointment this map does not display any data. All it does is illustrate the pipelines of and the cities served by the Columbia Gas System. The *Bowerman et al.* (2009) book states that natural gas

Figure 2. Primary Sampling Units for the Current Population Survey: the 1995 sample design with 792 PSUs.



Note: Alaska and Hawaii not shown.
Source: Bureau of the Census, Statistical Methods Division.

Fig. 4: Map of a sampling design for the 1995 population survey (*Freedman et al.*, 2007, p. 397). [From *STATISTICS* 4th edition by David Freeman, Robert Pisani, and Roger Purves. Copyright ©2007, 1998, 1991, 1991, 1978 by W. W. Norton & Company, Inc. Used by permission of W.W. Norton & Company, Inc.]

companies use prediction models when ordering to avoid transmission and storage fees aligned with over/under consumption. The data used in this example does not relate to the map view. This map easily could be intended to show the actual size of the storage fields and the capacity of the pipelines. This indicates that authors do use maps, just not to display data.

2.4 Statistics

Freedman et al. (2007) contained two major data sets with spatial components, “Smoking and Health” (p. 150) and “Death Rates from Breast Cancer” (p. 152) (See Table 5 in Appendix A.3). None of these are discussed in our report.

The *Freedman et al.* (2007) book also contained a map (see Figure 4). This map used a strong coloring scheme which might suggest data are represented. But the coloring didn’t show any strong patterns because it was used to show the 1995 population survey sampling design. This map actually has a strong resemblance to

a choropleth map which we will exhibit in the next section, and it can easily be transformed into such a map by displaying some actual measurements obtained in these sampling units.

CHAPTER 3

EXAMPLES OF SPATIAL ASSOCIATION


In this chapter we took six data sets from our selection of textbooks that contained strong spatial components and we created maps depicting some reasonable spatial association. Displaying spatial data on a map makes it easier to know the data, identify patterns, and identify outliers in space. As mentioned earlier, through the use of proper graphics, we can make critical discoveries and communicate them clearly.

3.1 Choropleth Maps

Symanzik and Carr (2008) stated: “Choropleth maps use the color or shading of regions in a map to represent region values. Choropleth maps have proved very popular but have many problems and limitations as indicated by writers such as *Robinson et al.* (1978), *Dent* (1993), and *Harris* (1999).” As with histograms, choropleth maps use class intervals to convert continuous estimates into an ordered variable. The choice of the colors and class breaks can severely influence the pattern that appears on the map. With this in mind, choropleth maps are one of easiest ways to put data on a map. They are essentially a color-by-number where each range of numbers is translated into a particular color.

3.1.1 Mortgage Delinquency Rates

The first choropleth map we created is based on data from the *Bowerman et al.* (2009) book. It includes a numerically sorted list of mortgage delinquency rates for all 50 states and Washington D.C. as reported by USA Today.com on March 13, 2007

TABLE 2.15 Mortgage Delinquency Rates for Each of the 50 States and the District of Columbia as Reported by USA Today.com on March 13, 2007 (for Exercise 2.40) 

Mississippi	10.6%	North Carolina	6.1%	Delaware	4.5%	Arizona	3.5%
Louisiana	9.1%	Arkansas	6.1%	Iowa	4.4%	Vermont	3.4%
Michigan	7.9%	Missouri	6.1%	New Hampshire	4.4%	Idaho	3.4%
Indiana	7.8%	Oklahoma	6.1%	Colorado	4.4%	California	3.3%
Georgia	7.5%	Illinois	5.4%	New Mexico	4.3%	Alaska	3.1%
West Virginia	7.4%	Kansas	5.1%	Connecticut	4.3%	Washington	2.9%
Texas	7.4%	Rhode Island	5.0%	Maryland	4.3%	South Dakota	2.9%
Tennessee	7.3%	Maine	4.9%	Wisconsin	4.1%	Wyoming	2.9%
Ohio	7.3%	Florida	4.9%	Nevada	4.1%	Montana	2.8%
Alabama	7.1%	New York	4.8%	Utah	4.0%	North Dakota	2.7%
Kentucky	6.3%	Nebraska	4.7%	Minnesota	4.0%	Oregon	2.6%
South Carolina	6.3%	Massachusetts	4.5%	Dist. of Columbia	3.7%	Hawaii	2.4%
Pennsylvania	6.3%	New Jersey	4.5%	Virginia	3.7%		

Source: Mortgage Bankers Association as reported by Noelle Knox, "Record Foreclosures Hit Mortgage Lenders," USA Today, March 13, 2007, http://www.usatoday.com/money/economy/housing/2007-03-13-foreclosures_N.htm.

Fig. 5: Original data set of mortgage delinquency rates (*Bowerman et al.*, 2009, p. 74). [From *Essentials of Business Statistics* (third edn) by B. L. Bowerman, R. T. OConnell, J. B. Orris, and E. S. Murphree. ©2009 by McGraw-Hill/Irwin. Used with permission.]

- The decimal point is at the |

```

2 | 4678999
3 | 134457
4 | 00113334445557899
5 | 014
6 | 1111333
7 | 13344589
8 |
9 | 1
10 | 6

```

Fig. 6: Stem-and-leaf plot of the mortgage data

(see Figure 5). The author asked the students to describe the distribution of the data and find any outliers by creating a stem-and-leaf plot.

We created a stem-and-leaf plot (Figure 6) for these data that provided us with some useful information. It tells us that the data are right-skewed, they appear multi-modal, there are likely a couple of outliers, and there is no consistent pattern. Although the purpose of this exercise is to practice using stem-and-leaf plots, much more information can be obtained from the data set by plotting the data for these states on a map.

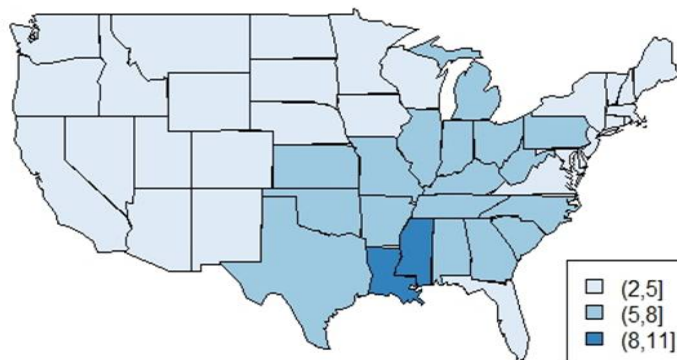


Fig. 7: Choropleth map of the mortgage delinquency rates. D.C., Alaska, and Hawaii were not included on the map.

Figure 7 shows what we get! It took only about ten lines of R code to produce Figure 7 and the results are astounding. Just by looking at the data in a table (see Figure 5), few people would be able to see such a strong spatial pattern. Figure 7 makes it clear south-eastern states have higher percentage of foreclosures. Moreover, those states that looked like they may have been outliers in the stem-and-leaf plot, are actually a part of a larger spatial pattern. These so-called outliers are the epicenter with extremely high mortgage delinquency rates. A band of medium-high mortgage delinquency rates surrounds these two states. Obviously the map leads to quite a different interpretation than the stem-and-leaf plot. R code for Figures 6 and 7 can be found in Appendix B.1.

3.1.2 CO₂ Emissions

For our next example, we considered the alphabetically sorted data set in Figure 8 from the *Moore et al.* (2009) book containing CO₂ emissions. *Moore et al.* (2009) asked the students to describe the distribution of the data, provide some summary statistics, identify outliers, and display the data using a graph. By looking at the adjacent problems, we assumed this means a histogram. The histogram in Figure 9 tells us that the data are right-skewed with a possibility of three outliers.

TABLE 1.6			
Carbon dioxide emissions (metric tons per person)			
Country	CO ₂	Country	CO ₂
Algeria	2.3	Mexico	3.7
Argentina	3.9	Morocco	1.0
Australia	17.0	Myanmar	0.2
Bangladesh	0.2	Nepal	0.1
Brazil	1.8	Nigeria	0.3
Canada	16.0	Pakistan	0.7
China	2.5	Peru	0.8
Columbia	1.4	Tanzania	0.1
Congo	0.0	Philippines	0.9
Egypt	1.7	Poland	8.0
Ethiopia	0.0	Romania	3.9
France	6.1	Russia	10.2
Germany	10.0	Saudi Arabia	11.0
Ghana	0.2	South Africa	8.1
India	0.9	Spain	6.8
Indonesia	1.2	Sudan	0.2
Iran	3.8	Thailand	2.5
Iraq	3.6	Turkey	2.8
Italy	7.3	Ukraine	7.6
Japan	9.1	United Kingdom	9.0
Kenya	0.3	United States	19.9
Korea, North	9.7	Uzbekistan	4.8
Korea, South	8.8	Venezuela	5.1
Malaysia	4.6	Vietnam	0.5

Fig. 8: Original data set of CO₂ emissions per person from countries with at least 20 million people (*Moore et al.*, 2009, p. 26). [From *Introduction to the Practice of Statistics* (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]

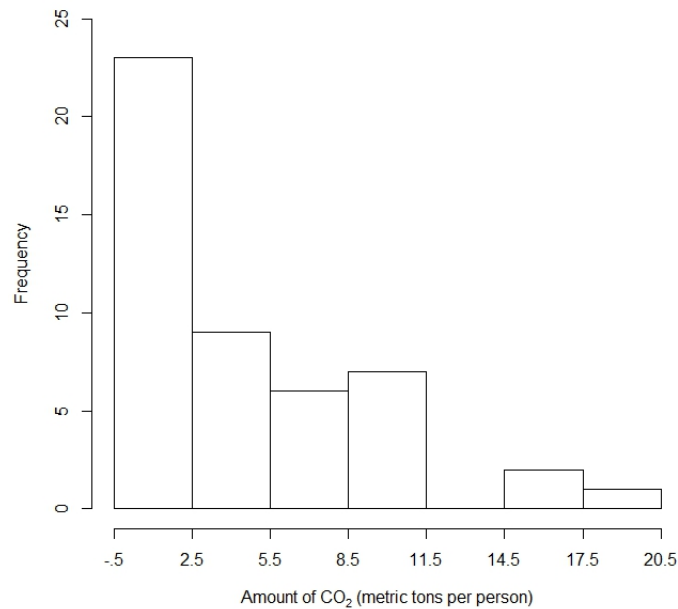


Fig. 9: Histogram of CO₂ data.

When we show the data from Figure 8 on a map (see Figure 10), we can see much more than just some skewed data. In Figure 10 we are able to see strong grouping among continents and geographical areas. More importantly, we are able to identify spatial outliers such as South Africa, Saudi Arabia, and Germany that weren't previously visible in the histogram. Their CO₂ emissions were rather different from the CO₂ emissions of their geographic neighbors, but they weren't unusual on a global (non-spatial) scheme. Again the countries possibly identified as outliers are actually part of a larger spatial pattern when viewed on a map. R code for Figure 9 can be found in Appendix B.2.1 and R code for Figure 10 can be found in Appendix B.2.

3.2 Google Maps

In this section we looked at displaying data using a different kind of map. Here we utilized the detail and resourcefulness of a Google map by plotting point data that have a latitude and longitude associated with them.

3.2.1 Public University Tuition

Moore et al. (2009) asked the students to plot the data from Figure 11 and check

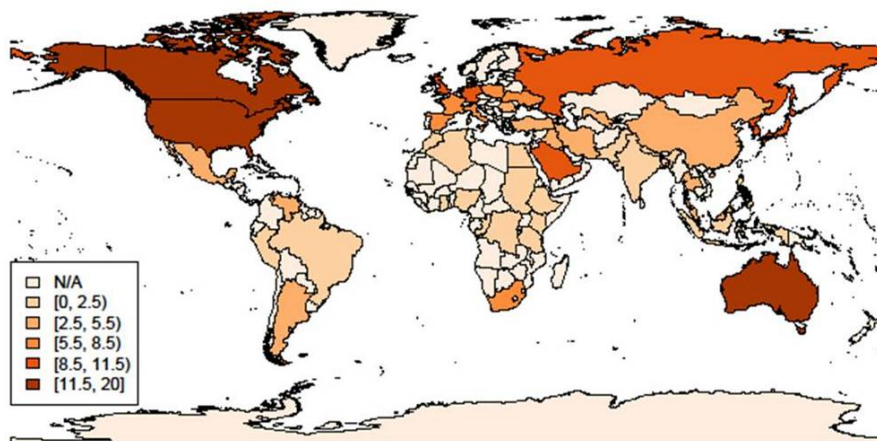


Fig. 10: Choropleth map of the CO₂ emissions data (in metric tons per person).

TABLE 10.1
In-state tuition and fees (in dollars) for 32 public universities

School	2000	2005	School	2000	2005	School	2000	2005
Penn State	7,018	11,508	Virginia	4,335	7,370	Iowa State	3,132	5,634
Pittsburgh	7,002	11,436	Indiana	4,405	7,112	Oregon	3,819	5,613
Michigan	6,926	9,798	Cal-Santa Barbara	3,832	6,997	Iowa	3,204	5,612
Rutgers	6,333	9,221	Texas	3,575	6,972	Washington	3,761	5,610
Illinois	4,994	8,634	Cal-Irvine	3,970	6,770	Nebraska	3,450	5,540
Minnesota	4,877	8,622	Cal-San Diego	3,848	6,685	Kansas	2,725	5,413
Michigan State	5,432	8,108	Cal-Berkeley	4,047	6,512	Colorado	3,188	5,372
Ohio State	4,383	8,082	UCLA	3,698	6,504	North Carolina	2,768	4,613
Maryland	5,136	7,821	Purdue	3,872	6,458	Arizona	2,348	4,498
Cal-Davis	4,072	7,457	Wisconsin	3,791	6,284	Florida	2,256	3,094
Missouri	4,726	7,415	Buffalo	4,715	6,068			

Fig. 11: Original data set of in-state undergraduate tuition and fees for 32 universities between 2000 and 2005 (*Moore et al.*, 2009, p. 595). [From *Introduction to the Practice of Statistics* (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]

whether a linear relationship existed between the two variables. Students also had to identify outliers, run a simple linear regression, and obtain the residuals.

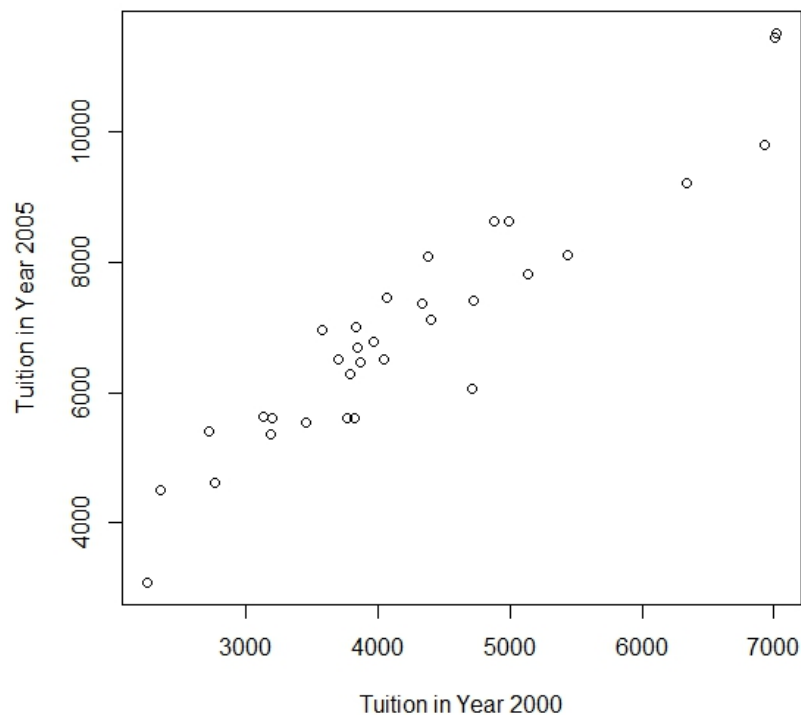


Fig. 12: Scatterplot of the tuition data.

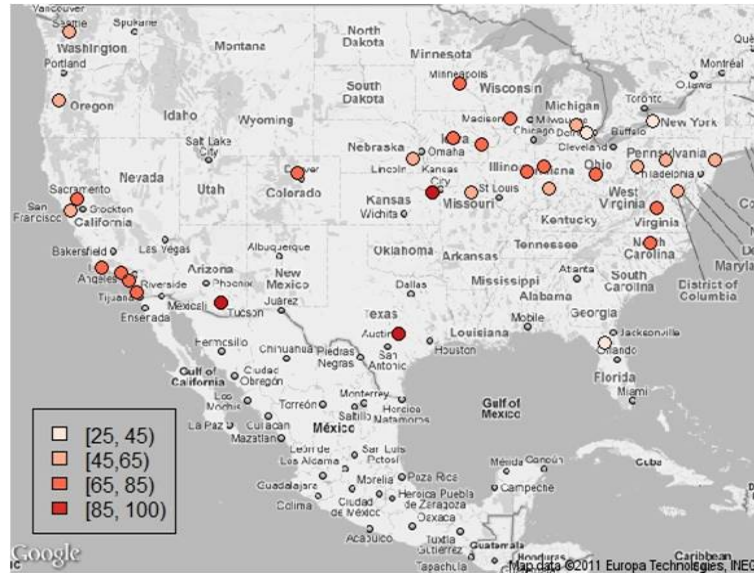


Fig. 13: Google map showing the percent increase of tuition from 2000 to 2005.

We first produced a scatterplot (Figure 12) of the data in Figure 11 and fitted the following linear regression model: $\hat{Y}r_{2005} = 1059 + (1.4) * Yr_{2000}$. Figure 12 suggests a strong linear relationship and possibly a few outliers.

Using the internet, we were able to obtain the latitude and longitude for each of the universities and plotted them on a Google map of the United States. To represent the difference of tuition and fees between the two years, we thought it made more sense to look at the percent increase for each university from 2000 to 2005. In Figure 13, the dark red colors represent the highest percent increase of tuition and fees as seen in Texas, Kansas, and Arizona. This map also shows some strong spatial grouping in different areas of the country and again some spatial outliers.

Adding a ray-glyph plot (*Carr et al. (1992)*) to the map allows us to look at multiple variables on one map. In Figure 14, the green lines represent the tuition for 2000 and 2005 respectively, with a line pointing down representing the lowest tuition and a line pointing up representing the highest tuition in that year. Universities whose lines are quite asymmetric are representing outliers. As such, it appears that

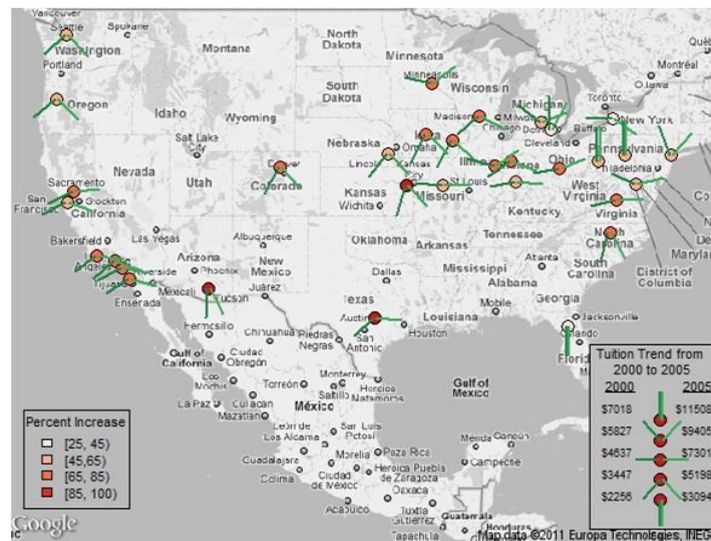


Fig. 14: Google map adding ray-glyph lines to the percent increase dots of the previous map (Figure 13).

Universities in Buffalo, Ohio, and Texas are outliers. A residual plot as in Figure 15, confirms that Buffalo, Ohio, and Texas are outliers as they have the highest/lowest residuals. R code for Figures 12 and 15 can be found in Appendix B.3.1 and R code

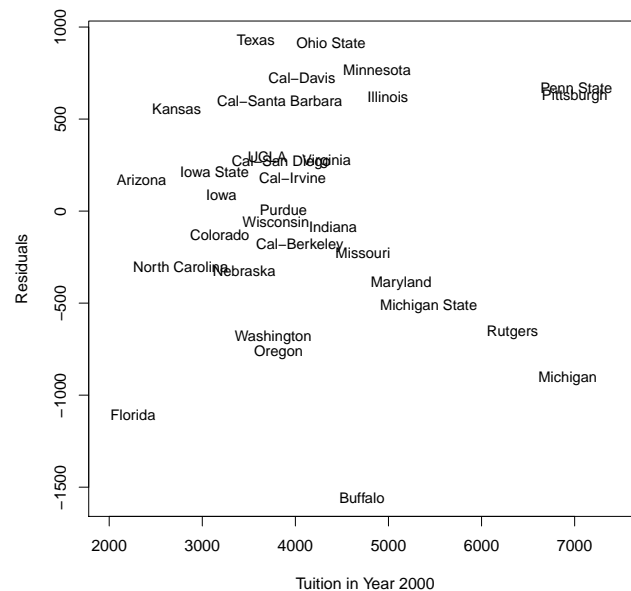



Fig. 15: Residual plot of tuition data.

TABLE 2.12 Major League Baseball Team Valuations and Revenues as Given on the Forbes.com Website on February 25, 2007 (for Exercise 2.24)  MLBTeams

Rank	Team	Value (\$mil)	Revenues (\$mil)	Rank	Team	Value (\$mil)	Revenues (\$mil)
1	New York Yankees	1026	277	16	Texas Rangers	353	153
2	Boston Red Sox	617	206	17	Cleveland Indians	352	150
3	New York Mets	604	195	18	Chicago White Sox	315	157
4	Los Angeles Dodgers	482	189	19	Arizona Diamondbacks	305	145
5	Chicago Cubs	448	179	20	Colorado Rockies	298	145
6	Washington Nationals	440	145	21	Detroit Tigers	292	146
7	St Louis Cardinals	429	165	22	Toronto Blue Jays	286	136
8	Seattle Mariners	428	179	23	Cincinnati Reds	274	137
9	Philadelphia Phillies	424	176	24	Pittsburgh Pirates	250	125
10	Houston Astros	416	173	25	Kansas City Royals	239	117
11	San Francisco Giants	410	171	26	Milwaukee Brewers	235	131
12	Atlanta Braves	405	172	27	Oakland Athletics	234	134
13	Los Angeles Angels @ Anaheim	368	167	28	Florida Marlins	226	119
14	Baltimore Orioles	359	156	29	Minnesota Twins	216	114
15	San Diego Padres	354	158	30	Tampa Bay Devil Rays	209	116

Source: http://www.forbes.com/lists/2006/33/Rank_1.html (accessed February 25, 2007).

Fig. 16: Original data set of franchise value and 2006 revenues for each of the 30 major league baseball (MLB) teams as reported by Forbes magazine and Forbes.com. (*Bowerman et al.*, 2009, p. 67). [From Essentials of Business Statistics (third edn) by B. L. Bowerman, R. T. OConnell, J. B. Orris, and E. S. Murphree. ©2009 by McGraw-Hill/Irwin. Used with permission.]

for Figures 13 and 14 can be found in Appendix B.3.

3.2.2 Major League Baseball Team Valuations and Revenue

From the data in Figure 16, the *Bowerman et al.* (2009) book gives us another beneficial use of a Google map to display data. Here the students are asked to develop several types of histograms and to describe the distribution of team values.

We produced just one histogram as seen in Figure 17. The data appears to be right-skewed and there is a possibility of an outlier. The color scheme seen here is the same as in Figure 18 and represents the distribution of team values by its saturation.

Using the internet, we were again able to obtain the latitude and longitude for each of the baseball teams and plotted them on a Google map of the United States. In Figure 18, the dark red colors represent the teams with relatively high value. There are a high number of lower valued teams and there appears to be some spatial

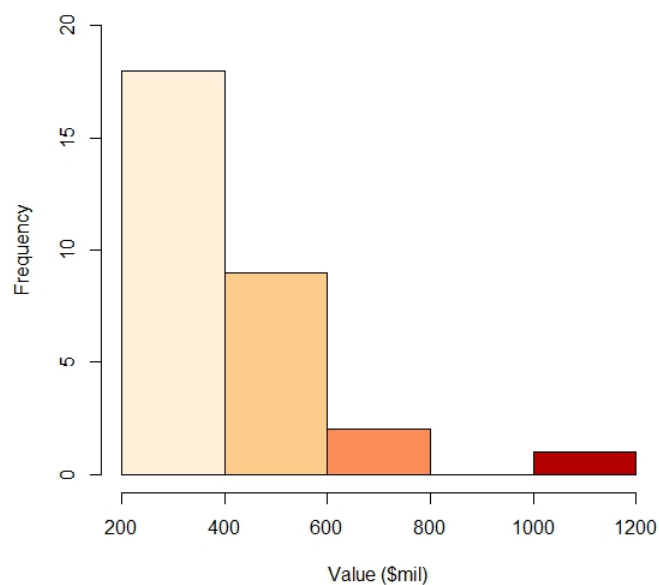


Fig. 17: Histogram of the MLB data.

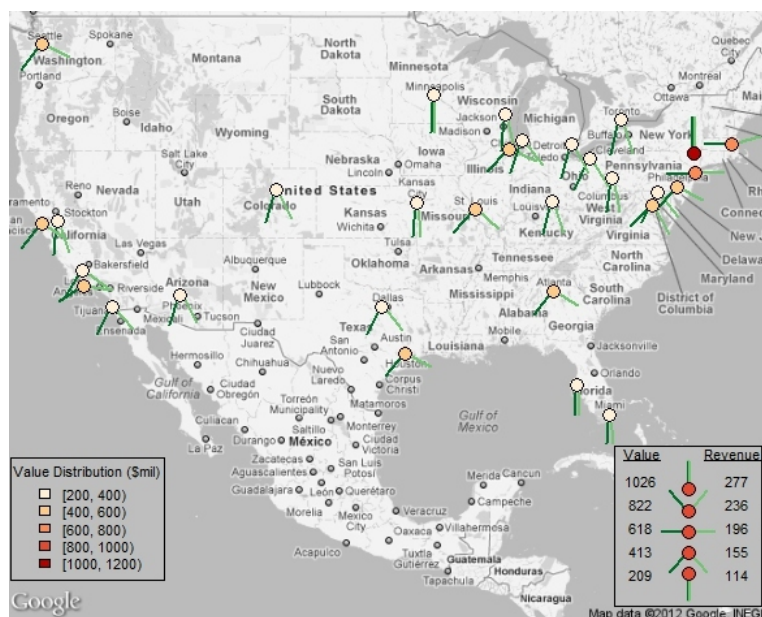


Fig. 18: Google map with ray-glyph lines showing Major League Baseball team revenues and values. Teams that were located in close proximity were manually moved to fix over-plotting.

patterns. The highest valued teams are in the Northeast United States. Areas that have two teams in close proximity to each other such as Los Angeles, San Francisco, Maryland, and Chicago seem to compete with each other resulting in one team having less value. There are other groupings in the Mid-east and Florida. The darkest red team is the New York Yankees and is a possible outlier.

The ray-glyph lines (*Carr et al.* (1992)) in this map gives us additional information a histogram can't. Again teams whose lines are asymmetric represent outliers. In this ray-glyph plot, it appears the Washington Nationals is an obvious outlier, with the LA Dodgers, Chicago White Sox, Chicago Cubs, Atlanta Braves, Boston Red Sox, Seattle Mariners, and Houston Astros as possible outliers. R code for Figure 17 can be found in Appendix B.4.1 and R code for Figure 18 can be found in Appendix B.4.

A ray-glyph map (see Figure 14 and Figure 18) allows us to display multiple variables at one time. It also allows us to identify the location of universities/teams that are not so well known. Finally, this kind of map allows us to combine a variable like percent increase by using a color scheme with the ray-glyph lines that represent actual tuition data for two years, all within a single map.

3.3 Micromaps

In this section, we worked with another type of map that also allowed us to view actual data values on a map. It is essentially a combination of multiple choropleth maps and a dot plot that is commonly called a linked micromap plot or just a micromap (*Carr et al.* (1998)).

3.3.1 Damage Due to Tornadoes

Our first example of a micromap comes from the data set in Figure 19. *Moore et al.* (2009) asked the students to identify the top five and bottom five states, make

TABLE 1.5
Average property damage per year due to tornadoes

State	Damage (\$millions)	State	Damage (\$millions)	State	Damage (\$millions)
Alabama	51.88	Louisiana	27.75	Ohio	44.36
Alaska	0.00	Maine	0.53	Oklahoma	81.94
Arizona	3.47	Maryland	2.33	Oregon	5.52
Arkansas	40.96	Massachusetts	4.42	Pennsylvania	17.11
California	3.68	Michigan	29.88	Puerto Rico	0.05
Colorado	4.62	Minnesota	84.84	Rhode Island	0.09
Connecticut	2.26	Mississippi	43.62	South Carolina	17.19
Delaware	0.27	Missouri	68.93	South Dakota	10.64
Florida	37.32	Montana	2.27	Tennessee	23.47
Georgia	51.68	Nebraska	30.26	Texas	88.60
Hawaii	0.34	Nevada	0.10	Utah	3.57
Idaho	0.26	New Hampshire	0.66	Vermont	0.24
Illinois	62.94	New Jersey	2.94	Virginia	7.42
Indiana	53.13	New Mexico	1.49	Washington	2.37
Iowa	49.51	New York	15.73	West Virginia	2.14
Kansas	49.28	North Carolina	14.90	Wisconsin	31.33
Kentucky	24.84	North Dakota	14.69	Wyoming	1.78

Fig. 19: Original data set of average property damage due to tornadoes from 1950 to 1999, adjusted for inflation (*Moore et al.*, 2009, p. 25). [From *Introduction to the Practice of Statistics* (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]

a histogram using software, compare it with a histogram with classes in increments of 10, and to identify outliers. This histogram in Figure 20 tells us that the data are right-skewed and that there are possibly three outliers.

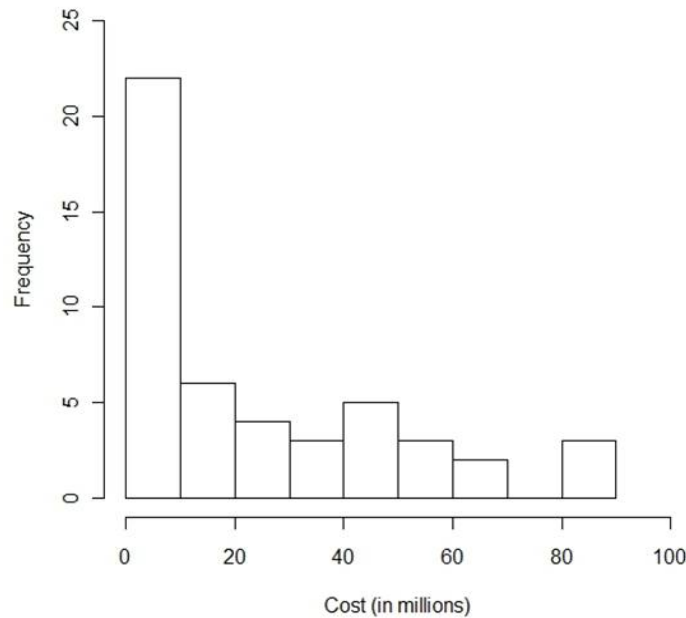


Fig. 20: Histogram of the tornado data.

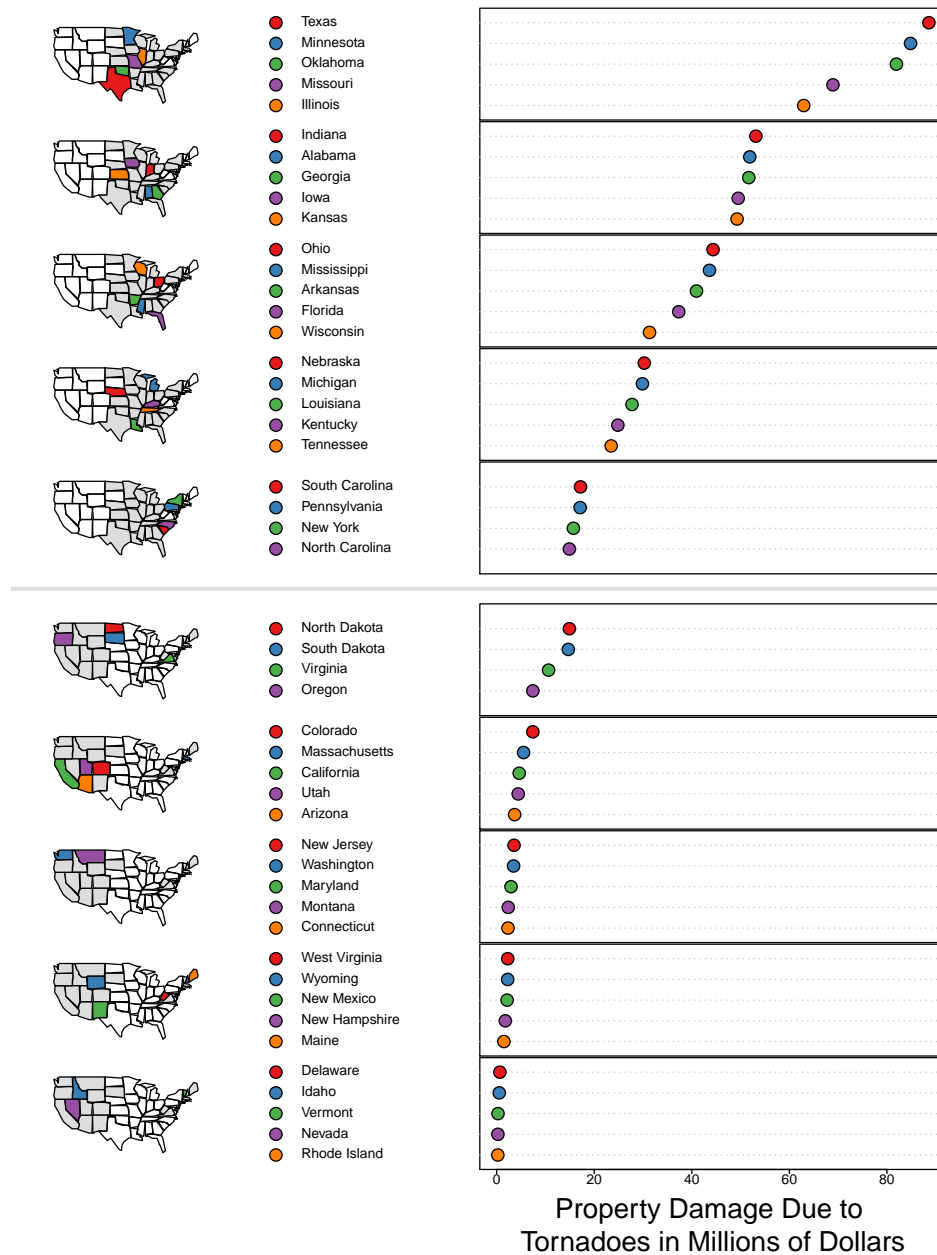


Fig. 21: Micromap of the tornado data. This map is for the continental U.S. only and does not show data for Hawaii, Alaska, and Puerto Rico. The right column gives us a color-coded dot plot that corresponds to a state in the middle column that is plotted on a map in the column on the left. States with an average property damage above the median are shaded grey in the five small maps above the separating line while states with an average property damage below the median are shaded in grey in the five small maps below the separating line.

As we follow the map in Figure 21 down from the top, we are able to visually see where and how the data are distributed. It appears that property damage is highest in the central United States (U.S.) and spreads out to the eastern and western coasts. Spatial outliers are visible and we are able to identify the top and bottom five states and see where they are located. The benefits of such maps are that they minimize loss of information by eliminating the need for breaks, provide better use of color in small regions, provide a better way to observe a continuous variable, and provide another way to show more than one variable on a choropleth map as demonstrated in the next section. R code for Figure 20 can be found in Appendix B.5.1 and R code for Figure 21 can be found in Appendix B.5.

We note that some ratio of property damage per square mile or property damage per inhabitant would be beneficial here as it is likely that large states will have higher total damage.

3.3.2 Fruit and Vegetable Consumption and Smoking

Our second example of a micromap is based on data that was collected by the Center for Disease Control (CDC) and the Behavioral Risk Factor Surveillance System (BRFSS). A link to the website housing the data in Figure 22 was found in the *Moore et al.* (2009), Appendix N-4, 53. Data were collected for 29 demographic characteristics and risk factors for each state (risk factors are listed in Appendix D-1 in the book). For the authors' purpose, only two risk factors were chosen: Fruits and Vegetables is the percent of adults in the state who reported eating at least five servings of fruits and vegetables per day. Smoking is the percent who smoked every day.

Moore et al. (2009) included the scatterplot shown in Figure 23 and asked the students to describe the relationship between the two variables and to identify certain

State	Fruits & Vegetables (percent)	Smoking (percent)	State	Fruits & Vegetables (percent)	Smoking (percent)
Alabama	20.1	18.8	Montana	24.7	14.5
Alaska	24.8	18.8	Nebraska	20.2	16.1
Arizona	23.7	13.7	Nevada	22.5	16.6
Arkansas	21.0	18.1	New Hampshire	29.1	15.4
California	28.9	9.8	New Jersey	25.9	12.8
Colorado	24.5	13.5	New Mexico	21.5	14.6
Connecticut	27.4	12.4	New York	26.0	14.6
Delaware	21.3	15.5	North Carolina	22.5	17.1
Florida	26.2	15.2	North Dakota	21.8	15.0
Georgia	23.2	16.4	Ohio	22.6	17.6
Hawaii	24.5	12.1	Oklahoma	15.7	19.0
Idaho	23.2	13.3	Oregon	25.9	13.4
Illinois	24.0	14.2	Pennsylvania	23.9	17.9
Indiana	22.0	20.8	Rhode Island	26.8	15.3
Iowa	19.5	16.1	South Carolina	21.2	17.0
Kansas	19.9	13.6	South Dakota	20.5	13.8
Kentucky	16.8	23.5	Tennessee	26.5	20.4
Louisiana	20.2	16.4	Texas	22.6	13.2
Maine	28.7	15.9	Utah	22.1	8.5
Maryland	28.7	13.4	Vermont	30.8	14.4
Massachusetts	28.6	13.5	Virginia	26.2	15.3
Michigan	22.8	16.7	Washington	25.2	12.5
Minnesota	24.5	14.9	West Virginia	20.0	21.3
Mississippi	16.5	18.6	Wisconsin	22.2	15.9
Missouri	22.6	18.5	Wyoming	21.8	16.3

Fig. 22: Original data set of fruit and vegetable consumption and daily smoking for all 50 states (*Moore et al.*, 2009, p. 161-162). [From *Introduction to the Practice of Statistics* (sixth edn) by D. S. Moore, G. P. McCabe, and B. Craig. ©2009 by W. H. Freeman and Company. Used with permission.]

points on the plot. Utah is the furthest to the left and California the second furthest. Both of these states have less than average smoking values, but California has an above average fruit and vegetable consumption. The plot shows a negative linear association; as fruit and vegetable consumption decreases, smoking increases.

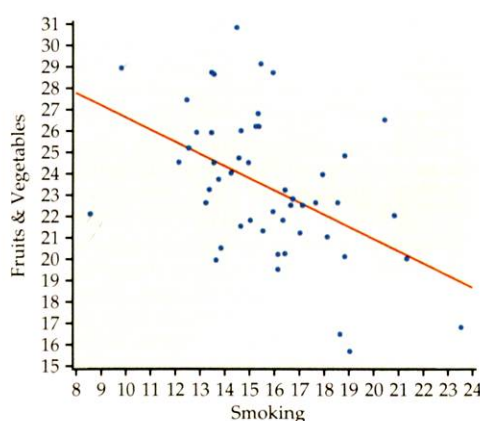


FIGURE 2.29 Fruits & Vegetables versus Smoking with least-squares regression line, for Exercise 2.136.

Fig. 23: Scatterplot and regression line related to the fruit and vegetable and smoking data set (*Moore et al.*, 2009, p. 162).

In Figure 24, we applied a more advanced micromap design that is able to include Hawaii and Alaska (and even Washington D.C. if needed) on the fruit and vegetable and smoking data set. This micromap enlarges the smaller states on the eastern coast and gives better dimensions to the rest of the states. If we look at California and Utah on the micromap we see two different stories. California is an obvious spatial outlier with respect to fruit and vegetable consumption, but it is not dramatically different from the other data points in the dot plot. Utah on the other hand is not a spatial outlier, but its smoking pattern is dramatically different from the other points in the dot plot. This tells us something might be different about Utah, possibly due to the high percentage of its religious population who don't smoke. We refer to *Gebreab et al.* (2008) for a detailed discussion on how to interpret a micromap similar to the one shown in Figure 21. R code for Figure 24 can be found in Appendix B.6.

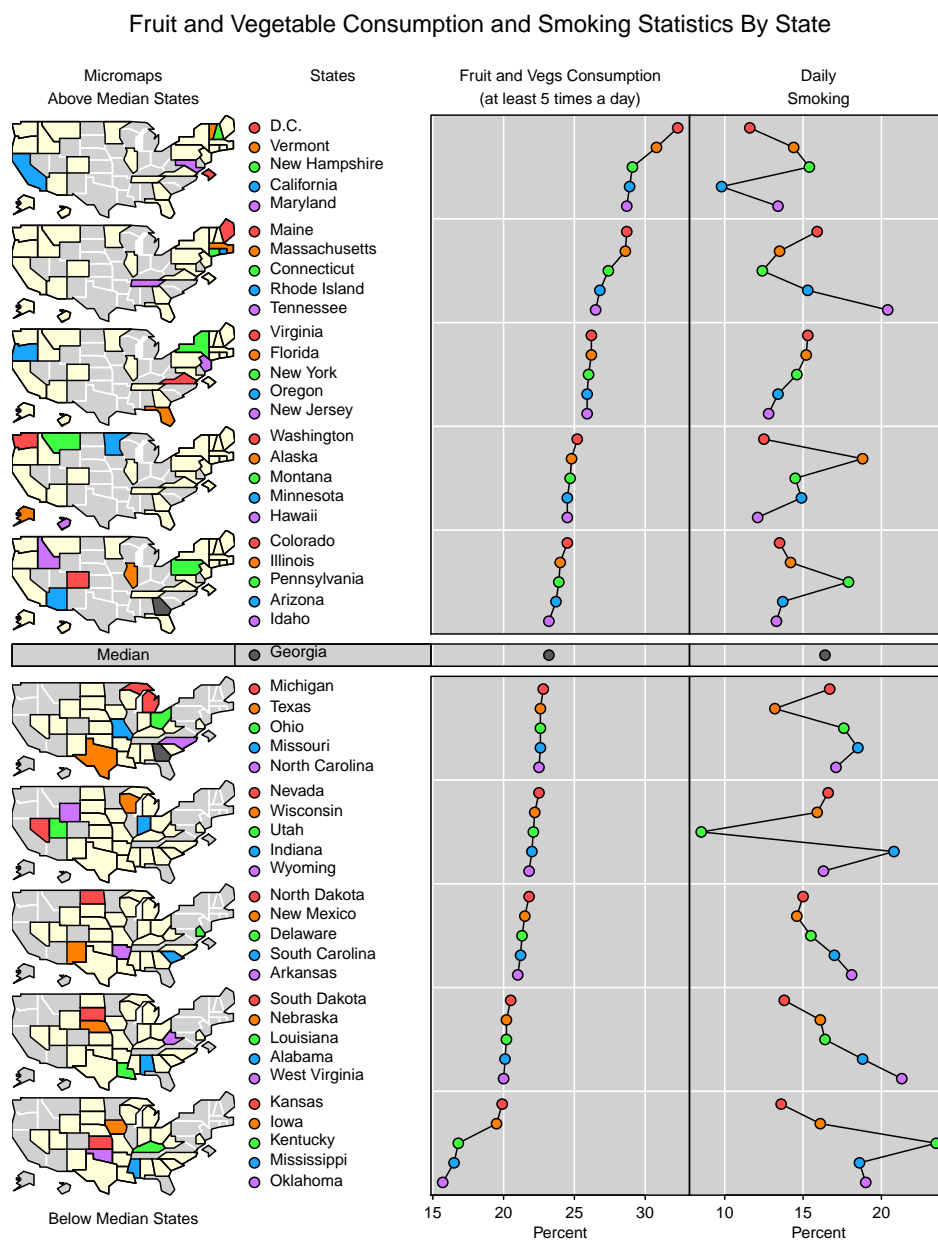


Fig. 24: Micromap of fruit and vegetable consumption and smoking data.

CHAPTER 4

DISCUSSION

Displaying spatial data via maps enriches understanding. We are surrounded by data. Collecting is only part of the battle. Displaying all components of a data set allows us to explore and discover valuable information that is otherwise lost when only sets of the data are displayed. Through displaying data with a spatial component via maps, many outliers or otherwise hard-to-interpret features evident in other plotting mechanisms can be explained by spatial context.

It makes sense to display data sets with a spatial component in a spatial framework such as a map. Maps allow us to better understand the data when interpreted in its geographic context. Also, maps can reveal spatial outliers that aren't readily visible in the data. Much of the information obtained from other plotting techniques can be extended by using maps. This can give us better insights in how to treat a particular location or try to identify its peculiarity.

We clearly do not expect from textbook authors to make major changes to their existing textbooks. However, we would strongly encourage textbook authors to add question parts that ask students to create a map or interpret a given map in the book. Many textbooks already include sections to challenge the students and have a platform to introduce advanced subjects. Textbooks could simply suggest the use of a map to plot the data and provide some examples so that students know the importance of using maps. After all, we are confronted with statistical maps almost on a daily basis; the best known example is the weather map that can be found on many newspapers, TV, and on the web.

REFERENCES

- Anscombe, F. J. (1973), Graphs in Statistical Analysis, *The American Statistician*, 27(1), 17–21.
- Bowerman, B. L., R. T. O’Connell, J. B. Orris, and E. Murphree (2009), *Essentials of Business Statistics*, third ed., McGraw-Hill/Irwin, New York.
- Carr, D. B., A. R. Olsen, and D. White (1992), Hexagon Mosaic Maps for Displays of Univariate and Bivariate Geographical Data, *Cartography and Geographic Information Systems*, 19(4), 228–236, 271.
- Carr, D. B., A. R. Olsen, J. P. Courbois, S. M. Pierson, and D. A. Carr (1998), Linked Micromap Plots: Named and Described, *Statistical Computing and Statistical Graphics Newsletter*, 9(1), 24–32.
- Dent, B. D. (1993), *Cartography: Thematic Map Design (Third Edition)*, William C. Brown, Dubuque, IA.
- Freedman, D., D. Pisani, and R. Purves (2007), *Statistics*, fourth ed., W. W. Norton & Company, New York.
- Gebreab, S. Y., R. R. Gillies, R. G. Munger, and J. Symanzik (2008), Visualization and Interpretation of Birth Defects Data Using Linked Micromap Plots, *Birth Defects Research (Part A): Clinical and Molecular Teratology*, 82, 110–119.
- Harris, R. L. (1999), *Information Graphics — A Comprehensive Illustrated Reference*, Oxford University Press, New York, NY.
- Hayter, A. (2006), *Probability and Statistics for Engineers and Scientists*, third ed., Duxbury Press, Belmont, CA.

- Lewin-Koh, N. J., R. Bivand, contributions by Edzer J. Pebesma, E. Archer, A. Baddeley, H.-J. Bibiko, J. Callahan, S. Dray, D. Forrest, M. Friendly, P. Giraudoux, D. Golicher, V. G. Rubio, P. Hausmann, K. O. Hufthammer, T. Jagger, S. P. Luque, D. MacQueen, A. Niccolai, T. Short, G. Snow, B. Stabler, and R. Turner (2012), *maptools: Tools for reading and handling spatial objects*, r package version 0.8-14.
- Loecher, M. (2012), *RgoogleMaps: Overlays on Google map tiles in R*, Berlin School of Economics and Law (BSEL), r package version 1.2.0.
- Moore, D. S., G. P. McCabe, and B. Craig (2009), *Introduction to the Practice of Statistics*, sixth ed., W. H. Freeman, New York.
- Moore, D. S., G. P. McCabe, and B. Craig (2012), *Introduction to the Practice of Statistics*, seventh ed., W. H. Freeman, New York.
- Neuwirth, E. (2011), *RColorBrewer: ColorBrewer palettes*, r package version 1.0-5.
- R Development Core Team (2012), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.
- Robinson, A., R. Sale, and J. Morrison (1978), *Elements of Cartography (Fourth Edition)*, John Wiley and Sons, New York, NY.
- Schnute, J. T., N. Boers, R. Haigh, and A. Couture-Beil. (2010), *PBSmapping: Mapping Fisheries Data and Spatial Analysis Tools*, r package version 2.61.9.

- Symanzik, J., and D. B. Carr (2008), Interactive Linked Micromap Plots for the Display of Geographically Referenced Statistical Data, in *Handbook of Data Visualization*, edited by C. Chen, W. Härdle, and A. Unwin, pp. 267–294 & 2 Color Plates, Springer, Berlin, Heidelberg.
- Voge, N. D., and J. Symanzik (2011), Ignoring the Spatial Context in Intro Statistics Classes – And Some Simple Graphical Remedies, in *2011 JSM Proceedings*, American Statistical Association, Alexandria, VA, (CD).
- Wainer, H. (1997), *Visual Revelations: Graphical Tales of Fate and Deception from Napoleon Bonaparte to Ross Perot*, Copernicus/Springer, New York, NY.

APPENDICES

APPENDIX A

ADDITIONAL SPATIAL EXAMPLES FROM TEXTBOOKS

A.1 Introduction to the Practice of Statistics

Table 1: Table from *Moore et al.* (2009) book containing spatial examples where a map display seems beneficial, but was not attempted in this report. The parentheses following its description is the exercise number or example found in the book. (1.28) refers to chapter one, exercise 28 etc. (Ex. 1.13) refers to Example 1.13. “n” is the sample size and “p” is the number of variables.

Pg	Title	n	p	Description
10	Literacy Rates (percent) in Islamic Nations (Ex. 1.7)	17	2	Compares male and female literacy rates among major Islamic nations in 2002. Countries with population less than 3 million were omitted. Data for some nations weren’t available. From earthtrends.wri.org.
78	Park Space and Population (1.152)	12	2	Compares population, and park and open space of several U.S. cities with high population density. Population is reported in thousands of people, and open space in acres. From www.oasisync.net.
98	NBA Teams as Businesses (2.19)	29	3	Shows the values (in \$millions), revenue (in millions of dollars), and income (in millions of dollars) of teams in the NBA. Forbes, 2004.

Table 2: Table from *Moore et al.* (2009) book containing spatial examples that are too small and/or are categorical and would not benefit from a map. See Table 1 for additional details.

Pg	Title	n	p	Description
167	University Degrees in Asia (2.156)	3	3	Compares university degrees in engineering, natural science, and social science among three regions: United States, Western Europe, and Asia. Data is categorical.
702	The Effects of Peer Pressure on Mathematics Education (13.12)	3	3	Compares the results of a questionnaire that asked male and female students to rate how often the fear being called a nerd or teacher’s pet on a 4-point scale. Countries were chosen where high achievement is not valued highly. Such as Germany, Canada, and Israel. Data is categorical.

A.2 Essentials of Business Statistics

Table 3: Table from *Bowerman et al.* (2009) book containing spatial examples where a map display seems beneficial, but was not attempted in this paper. See Table 1 for additional details.

Pg	Title	n	p	Description
134	Housing Affordability in Texas (3.20)	6	1	Compares the percentage of homes sold during the fourth quarter of 2006 that a median income household could afford to purchase at the prevailing mortgage interest rate for six metro areas in Texas.
559	Accounting Rates (13.67)	54	2	Accounting rates on stocks and market returns for 54 companies.
560	New Jersey Bank Data (13.68)	21	2	Compares the percent of minority population vs the number of residents per bank branch.

Table 4: Table from *Bowerman et al.* (2009) book containing spatial examples that are too small and/or are categorical and would not benefit from a map. See Table 1 for additional details.

Pg	Title	n	p	Description
89	J. D. Power Quality Study of 2006 Automobiles (2.67 - 2.74)	5	3	Compares 35 automobile manufactures and reports their manufacturing and design quality and country of origin. More categorical in nature.
124	Comparing Lifestyles in US and Eight Other Countries (3.9 - 3.13)	9	5	Compares voting percentage, income tax, video rentals, number of PC's, and religion. Has too many variables.
150	June 2001 Unemployment Rates (Section 3.5)	4	2	June 2001 unemployment rates for various regions in the United States.
153	Unemployment Rates (3.44)	5	2	Compares the January 2005 civilian labor force sizes and unemployment rates in five Midwestern states.
205	Airline Delays (4.67)	5	3	Compares the number of on time, delayed, and total flights of two airlines in five major airports.

A.3 Statistics

Table 5: Table from *Freedman et al.* (2007) book containing spatial examples where a map display seems beneficial, but was not attempted in this paper.

Pg	Title	n	p	Description
150	Smoking and Health (Ch. 9, Set D - 1.)	11	2	Adapted from a 1955 article by Sir Richard Doll on the relationship between per capita consumption of cigarette smoking in various countries in 1930 and the death rates (per million) from lung cancer for men in 1950.
152	Death rates from breast cancer due to fat in diet (Ch. 9, Sec. 5, Ex. 3)	40	2	List of 40 countries that eat a lot of fat.

APPENDIX B

R CODE

B.1 Choropleth Map and Stem and Leaf Plot: Mortgage Delinquency

```

# United States Choropleth Map
# By          Juergen Symanzik (revised for current data by Nathan Voge)
#
#
#          1 Import functions and data
#          2 Sort the data and assign colors
#          3 Draw map, plot data, make titles, and create a legend

# 1. Import needed functions and data-----

# load required packages
library(RColorBrewer)
library(maps)

# read in data

## Note, D.C was in our data set, but not on the map we are using
## so it is deleted from the data set

State = c("Mississippi","Louisiana","Michigan","Indiana","Georgia",
          "West Virginia","Texas","Tennessee","Ohio","Alabama",
          "Kentucky","South Carolina","Pennsylvania","North Carolina",
          "Arkansas","Missouri","Oklahoma","Illinois","Kansas",
          "Rhode Island","Maine","Florida","New York","Nebraska",
          "Massachusetts","New Jersey","Delaware","Iowa","New Hampshire",
          "Colorado","New Mexico","Connecticut","Maryland","Wisconsin",
          "Nevada","Utah","Minnesota","Virginia","Arizona","Vermont",
          "Idaho","California","Alaska","Washington","South Dakota",
          "Wyoming","Montana","North Dakota","Oregon","Hawaii")

Delinquency = c(10.6,9.1,7.9,7.8,7.5,7.4,7.4,7.3,7.3,7.1,6.3,6.3,6.3,6.1,6.1,
               6.1,6.1,5.4,5.1,5.4,4.9,4.9,4.8,4.7,4.5,4.5,4.5,4.4,4.4,4.4,4.3,
               4.3,4.3,4.1,4.1,4.4,4.3,3.5,3.4,3.4,3.3,3.1,2.9,2.9,2.9,2.8,
               2.7,2.6,2.4)

mort = cbind(State, Delinquency)

# read in data via a file, mort = read.csv("DelinqRate_edit.csv", header=TRUE)

# 2. Sort the data and assign colors-----

# set how to divide up the data
breaks = c(2, 5, 8, 11)

# apply the breaks to the data set
m.class = cut(as.numeric(mort[, 2]), breaks)

```

```

# pick colors and assign colors to breaks
m.col = brewer.pal(3, "Blues")[m.class]

# match states in map to states in the data set
## Note, states in the map are "characters, so we must make the
## the states in our data set "characters" as well
map.m.col = m.col[match.map("state", as.character(mort[, 1]))]

# 3. Draw map, plot data, make titles, and create a legend-----

# save map as a .pdf
pdf("Mortgages_18Jul_Map.pdf", width = 11, paper = "USr")

# create the map
map("state", fill = T, col = map.m.col)
legend("bottomright", legend = levels(m.class), fill = brewer.pal(3, "Blues"))

# optional title: title("Mortgage Delinquency Rates (%) in 2007")

dev.off()

```

B.2 World Choropleth Map: CO2 Emissions

```

# World Choropleth Map
# By          Juergen Symanzik (revised for current data by Nathan Voge)
#
#
#           1 Import functions and data
#           2 Sort the data and assign colors
#           3 Draw map, plot data, make titles, and create a legend

# 1. Import needed functions and data-----

library(RColorBrewer)
brew.color = brewer.pal(6, "Oranges")

library(mapttools)
data(wrld_simpl)

# read in data

Country = c("United States", "Australia", "Canada", "Saudi Arabia",
            "Russia", "Germany", "Korea, Republic of", "Japan",
            "United Kingdom", "Korea, Democratic People's Republic of",
            "South Africa", "Poland", "Ukraine", "Italy", "Spain", "France",
            "Venezuela", "Uzbekistan", "Malaysia", "Argentina", "Romania", "Iran",
            "Mexico", "Iraq", "Turkey", "China", "Thailand", "Algeria", "Brazil",
            "Egypt", "Columbia", "Indonesia", "Morocco", "India", "Philippines",
            "Peru", "Pakistan", "Vietnam", "Kenya", "Nigeria", "Bangladesh",
            "Ghana", "Myanmar", "Sudan", "Nepal", "Tanzania", "Congo", "Ethiopia")

# changed some of the country names in the data set to match the
# map. Ex. United States, Russia, Rep. of Korea

CO.2 = c(19.9, 17, 16, 11, 10.2, 10, 9.7, 9.1, 9, 8.8, 8.1, 8, 7.6, 7.3, 6.8, 6.1, 5.1, 4.8, 4.6,
        3.9, 3.9, 3.8, 3.7, 3.6, 2.8, 2.5, 2.5, 2.3, 1.8, 1.7, 1.4, 1.2, 1, 0.9, 0.9, 0.8, 0.7,
        0.5, 0.3, 0.3, 0.2, 0.2, 0.2, 0.2, 0.1, 0.1, 0, 0)

```

```

co2.2 = data.frame(Country, CO2.2)

# read in data via a file, co2 = read.csv("CO2_2.csv", header = TRUE)

# extract country names and CO2 values
country.data = (co2.2[, 1]) # countries of data set, 48 here
country.co2 = (co2.2[, 2])
country.map = wrld_simpl$NAME # countries' names as coded in wrld_simpl data
                                # set, contains 246 countries. Make sure
                                # countries' names in data sets match (spelled
                                # the same etc.), use: for (i in 1:104)
                                # print(grep(country.data[i],country.map))

n.map = length(country.map)
color.map = numeric(n.map)

# 2. Sort the data and assign colors_-----

# use "grep" to match country.data with country.map and assign a color level
for (i in 1:length(country.data)){
  if (country.co2[i] >= 16)
    color.map[grep(country.data[i], country.map)] = 6
  else if (country.co2[i] >= 8.8)
    color.map[grep(country.data[i], country.map)] = 5
  else if (country.co2[i] >= 6.1)
    color.map[grep(country.data[i], country.map)] = 4
  else if (country.co2[i] >= 2.5)
    color.map[grep(country.data[i], country.map)] = 3
  else if (country.co2[i] >= 0)
    color.map[grep(country.data[i], country.map)] = 2
}

# collect countries with actual scores (not equal to zero) as listed in
# country.data
collected.countries = (color.map != 0)
color.map2 = color.map

# assign colors
color.map2[collected.countries] = brew.color[1] # use light grey for countries
                                                    # not listed in country.data
                                                    # object
color.map2[!collected.countries] = brew.color[color.map[!collected.countries]]

# 3. Draw map, plot data, make titles, and create a legend_-----

# save it as a .pdf (looks better)
pdf("co2.2_18Jul_Map.pdf", width = 11, paper = "USr")

# plot map
plot(wrld_simpl, col = color.map2, axes = FALSE, ylim = c(-55, 90))

# create a legend
legend(-180, -15, legend = c("N/A", "[0, 2.5)", "[2.5, 5.5)", "[5.5, 8.5)",
  "[8.5, 11.5)", "[11.5, 20)"), fill = brewer.pal(6, "Oranges"), bg = "white")

# make a title at the top
# par(mar = c(0, 0, 12.5, 0))
# title("Carbon Dioxide Emissions (metric tons per person)")

# shut down the current graphic device so can save as a pdf
dev.off()

```

B.2.1 Histogram: CO2 Emissions

```
# CO2 Histogram
# By      Nathan Voge
#
#
#      1 Import functions and data
#      2 Draw plots, define colors, create breaks

# 1. Import needed functions and data_____

library(grDevices)
library(RColorBrewer)
library(mapttools)

data(wrld_simpl)

CO2 = c(19.9,17,16,11,10.2,10,9.7,9.1,9,8.8,8.1,8,7.6,7.3,6.8,6.1,5.1,4.8,4.6,
        3.9,3.9,3.8,3.7,3.6,2.8,2.5,2.5,2.3,1.8,1.7,1.4,1.2,1,0.9,0.9,0.8,0.7,
        0.5,0.3,0.3,0.2,0.2,0.2,0.2,0.1,0.1,0,0)

co2.2 = as.numeric(CO2)

# read in data via a file, co2.2 = read.csv("CO2_2.csv", header = TRUE)

# 2. Draw plots, define colors, create breaks_____

brewcol = brewer.pal(6, "Oranges") # define colors

breaks = c(-.5, 2.5, 5.5, 8.5, 11.5, 14.5, 17.5, 20.5)

print(paste("Histogram of World", ~CO[2], "Emissions"))

# save it as a .pdf (looks better)
pdf("co2.2_18Jul_hist.pdf", width = 11, paper = "USr")

hist(co2.2, breaks = breaks,
      xlab = "Amount of"~CO[2]~"(metric tons per person)", ylim = c(0, 25), xlim = range(breaks),
      xaxt = "n", main="")

# optional title: ("Histogram of World" ~CO[2]~"Emissions")

axis(1, at = breaks, labels = c("-.5", "2.5", "5.5", "8.5",
                                "11.5", "14.5", "17.5", "20.5"))

# shut down the current graphic device so can save as a pdf
dev.off()
```

B.3 Rayglyph Googlemap: University Tuition


```

# Google Map Featuring Ray-Glyph Lines
# By      Nathan Voge (based on work by Daniel B. Carr (ray-glyphs) and
#          Markus Loecher (RgoogleMaps))
#
#
#          1 Import functions and data
#          2 Create a Google map
#          3 Add ray-glyph lines to the map
#          4 Add colored circles depicting percent increase to the map
#          5 Create percent increase legend
#          6 Create ray-glyph legend
#          7 Plot title

# 1. Import needed functions and data-----

# load in required packages
library(RColorBrewer)
library(RgoogleMaps)
library(PBSmapping)

# read in data
University = c("Kansas","Texas","Arizona","Ohio State","Cal-Davis",
               "Cal-Santa Barbara","Iowa State","Minnesota","UCLA","Iowa",
               "Illinois","Cal-San Diego","Virginia","Cal-Irvine","Colorado",
               "Purdue","Wisconsin","North Carolina","Penn State","Pittsburgh",
               "Indiana","Cal-Berkeley","Nebraska","Missouri","Maryland",
               "Michigan State","Washington","Oregon","Rutgers","Michigan",
               "Florida","Buffalo")

Year_2000 = c(2725,3575,2348,4383,4072,3832,3132,4877,3698,3204,4994,3848,4335,
              3970,3188,3872,3791,2768,7018,7002,4405,4047,3450,4726,5136,5432,
              3761,3819,6333,6926,2256,4715)

Year_2005 = c(5413,6972,4498,8082,7457,6997,5634,8622,6504,5612,8634,6685,
              7370,6770,5372,6458,6284,4613,11508,11436,7112,6512,5540,7415,
              7821,8108,5610,5613,9221,9798,3094,6068)

Latitude = c(38.953611,30.28614,32.231667,40.38.54,34.41254,42.023949,
             44.975278,34.072222,41.655833,40.110539,32.881,38.035,33.64535,
             40.006667,40.424,43.075,35.908333,40.796036,40.444565,39.1661583,
             37.87,40.8175,38.9453,38.9875,42.723387,47.6599,44.044044,
             40.741632,42.283,29.64833,42.9286688)

Longitude = c(-95.26,-97.73942,-110.951944,-83.0145,-121.75,-119.84813,
              -93.647595,-93.234167,-118.444097,-91.525,-88.228411,-117.238,
              -78.505,-117.842642,-105.267222,-86.929,-89.417222,-79.05,
              -77.862739,-79.953274,-86.5263857,-122.259,-96.701389,-92.3288,
              -76.94,-84.481366,-122.306,-123.075736,-74.17486,-83.735,
              -82.34944,-78.8480905)

Percent_Increase = c(0.9864,0.9502,0.9157,0.8439,0.8313,0.8259,0.7989,0.7679,
                     0.7588,0.7516,0.7289,0.7373,0.7001,0.7053,0.6851,0.6679,
                     0.6576,0.6665,0.6398,0.6332,0.6145,0.6091,0.6058,0.5690,
                     0.5228,0.4926,0.4916,0.4698,0.4560,0.4147,0.3715,0.2870)

data = data.frame(University, Year_2000, Year_2005, Latitude, Longitude,
                  Percent_Increase)

# read in data via a file, tuit = read.csv("tuition.csv", header = TRUE)

# 2. Create a Google map-----

# Define x and y coordinates on map. In all RgoogleMaps packages, the latitude
# is read first followed by the longitude. For our data set, we looked up the

```

```

# latitude and longitude online for each university.
x = data["Latitude"]
y = data["Longitude"]

# Define the bounding box (bb) for the map using the latitude and longitude from
# our data set.
bb = qbbox(x, y, TYPE = "all", margin = list(m = rep(5, 4),
      TYPE = c("perc", "abs")[1]))

# Create a Google map by loading a static map from the package and define
# some options. bb$lonR, bb$latR are our latitude and longitude from the
# previous line of code. We have a few choices on the type of map we want to use
# such as "roadmap", "satellite", "terrain", etc. The default is a color map,
# here we want it to be gray so we use the "GRAYSCALE" argument. If needed, we
# can also use the "zoom" argument. This zooms in/out around the area we
# define by using the argument "center".

# save it as a .pdf (looks better)
pdf("tuit_18Jul_map.pdf", width = 7, paper = "USr")

MyMap = GetMap.bbox(bb$lonR, bb$latR, maptype = "roadmap", GRAYSCALE = TRUE)

# This command plots the map in the graphics window in R
PlotOnStaticMap(MyMap)

## Note: Content is added to the static map by using "PlotOnStaticMap".
# "PlotOnStaticMap" is repeated for each color/item added to the map. "FUN"
# identifies the plotting function, usually points or lines. "pch" and "cex"
# normal plotting parameters.

# 3. Add Ray-glyph lines to the map-----

# Function to Calculate the Latitude and Longitude for Ray-glyph Lines
# It uses the two variables you want to compare (var1, var2), the latitude
# and longitude for these variables (lat, lon), and the minimum and maximum
# of the two variables (min1, max1, min2, max2) respectively. It returns the
# latitude and longitude of the point away from the origin for each variable.

CalcRayGlyphMapLatLon = function(var1, var2, lat, lon, min1, max1, min2,
      max2) {
  num1 = var1 - min1
  ratio1 = num1 / (max1 - min1)
  angle1 = 270 - ratio1 * 180
  rad1 = angle1 * pi / 180
  lon1 = lon + 2 * cos(rad1)
  lat1 = lat + 2 * sin(rad1)

  num2 = var2 - min2
  ratio2 = num2 / (max2 - min2)
  angle2 = 270 + ratio2 * 180
  rad2 = angle2 * pi / 180
  lon2 = lon + 2 * cos(rad2)
  lat2 = lat + 2 * sin(rad2)

  list = c(lat1 = lat1, lon1 = lon1, lat2 = lat2, lon2 = lon2)
}

# Function to Plot Ray-glyph lines.
# "variable" is the plotting parameter. "lncolor1" is the color of the left
# line. "lncolor2" is the color of the right line. "cexln" is the cex of the
# line. "lwdln" is the line width. "ltyln1" and "ltyln2" and the respective
# line types.

```

```

PlotRayGlyphLines = function(data, variable, lncolor1, lncolor2, cexln = 1.5,
                             lwdln = 2, ltyln1 = 1, ltyln2 = 1) {
  colselect = subset(data, select = variable)
  for (i in 1:nrow(colselect)) {
    loc = data[colselect[i, ], ] # select each location
    PlotOnStaticMap(MyMap,
                    lat = c(as.numeric(loc[4]),
                           as.numeric(CalcRayGlyphMapLatLon(loc[2], loc[3], loc[4], loc[5],
                                                                min(data[2]), max(data[2]), min(data[3]),
                                                                max(data[3]))[1])),
                    lon = c(as.numeric(loc[5]) - .1,
                           as.numeric(CalcRayGlyphMapLatLon(loc[2], loc[3], loc[4], loc[5],
                                                                min(data[2]), max(data[2]), min(data[3]),
                                                                max(data[3]))[2]) - .1),
                    FUN = lines, col = lncolor1, add = TRUE, pch = 1, cex = cexln,
                    lwd = lwdln, lty = ltyln1) #Line on left
    PlotOnStaticMap(MyMap,
                    lat = c(as.numeric(loc[4]),
                           as.numeric(CalcRayGlyphMapLatLon(loc[2], loc[3], loc[4], loc[5],
                                                                min(data[2]), max(data[2]), min(data[3]),
                                                                max(data[3]))[3])),
                    lon = c(as.numeric(loc[5]) + .1,
                           as.numeric(CalcRayGlyphMapLatLon(loc[2], loc[3], loc[4], loc[5],
                                                                min(data[2]), max(data[2]), min(data[3]),
                                                                max(data[3]))[4]) + .1),
                    FUN = lines, col = lncolor2, add = TRUE, pch = 1, cex = cexln,
                    lwd = lwdln, lty = ltyln2) #Line on right
  }
}

# Define color scheme for ray-glyph lines
palgreen = brewer.pal(9, "Greens")
lncolor1 = palgreen[8]
lncolor2 = palgreen[5]

# Run PlotRayGlyphLines function and plot ray-glyph lines. Change plotting
# variable. ("University" here)
PlotRayGlyphLines(data = data, variable = "University", lncolor1, lncolor2)

# 4. Add colored circles depicting percent increase to the map-----

# Function to Plot Colored Circles Outlined in Black.
# Data*[4] and data*[5] contain the latitude and longitude for each
# respective level (*), which are our breaks in the data.
# After many tries, we found 4 levels most useful
# for our data set. "subset" allows us to identify how we want to divide up the
# data with "Percent_Increase" being our variable of interest.
# Here we put black circles around the colors to make them more visible.
# "pch = 1" is a circle. "cexcir" is the cex of the circles. "pchdot" is the pch
# of the dot. "pchcir" is the pch of the circle

## Note: The data in "Percent_Increase" was not provided by the original data
## set.

PlotPoints = function(data, cexcir = 1.5, pchdot = 19, pchcir = 1) {
  for (i in 1:4) {
    if (i == 1) {
      data1 = subset(data, Percent_Increase >= .25 & Percent_Increase < .45)
      PlotOnStaticMap(MyMap, data1[, 4], data1[, 5], FUN = points,
                      col = palcircle[i], add = TRUE, pch = pchdot, cex = cexcir)
      PlotOnStaticMap(MyMap, data1[, 4], data1[, 5], FUN = points,
                      col = "black", add = TRUE, pch = pchcir, cex = cexcir,
                      lwd = 1)
    }
  }
}

```

```

    }
  else if (i == 2) {
    data2 = subset(data, Percent_Increase >= .45 & Percent_Increase < .65)
    PlotOnStaticMap(MyMap, data2[,4], data2[,5], FUN = points,
                    col = palcircle[i], add = TRUE, pch = pchdot,
                    cex = cexcir)
    PlotOnStaticMap(MyMap, data2[,4], data2[,5], FUN = points,
                    col = "black", add = TRUE, pch = pchcir, cex = cexcir,
                    lwd = 1)
  }
  else if (i == 3) {
    data3 = subset(data, Percent_Increase >= .65 & Percent_Increase < .85)
    PlotOnStaticMap(MyMap, data3[,4], data3[,5], FUN = points,
                    col = palcircle[i], add = TRUE, pch = pchdot,
                    cex = cexcir)
    PlotOnStaticMap(MyMap, data3[,4], data3[,5], FUN = points,
                    col = "black", add = TRUE, pch = pchcir, cex = cexcir,
                    lwd = 1)
  }
  else if (i == 4) {
    data4 = subset(data, Percent_Increase >= .85 & Percent_Increase < 1.0)
    PlotOnStaticMap(MyMap, data4[,4], data4[,5], FUN = points,
                    col = palcircle[i], add = TRUE, pch = pchdot,
                    cex = cexcir)
    PlotOnStaticMap(MyMap, data4[,4], data4[,5], FUN = points,
                    col = "black", add = TRUE, pch = pchcir, cex = cexcir,
                    lwd = 1)
  }
}
}

# Run PlotPoints function and plot dots/circles.

# Define color scheme for our circles. We used the website
# http://colorbrewer2.org to identify an appropriate color scheme with 4 levels.
palcircle = brewer.pal(4, "OrRd")
cexcir = 1.5
pchdot = 19
pchcir = 1

# Run colored circles function
PlotPoints(data, cexcir, pchdot, pchcir)

## Note: If data doesn't have a percent increase variable, use the below code
## to plot the circles

# PlotOnStaticMap(MyMap, data[,4], data[,5], FUN = points,
#                 col = palcircle[4], add = TRUE, pch = pchdot, cex = cexcir)
# PlotOnStaticMap(MyMap, data[,4], data[,5], FUN = points,
#                 col = "black", add = TRUE, pch = pchcir, cex = cexcir,
#                 lwd = 1)

# 5. Create percent increase legend-----

# The x and y coordinates of the legend box are based on
# the latitude and longitude of the static map from Google. The
# function "LatLon2XY.centered" transforms latitude and longitude into x and
# y coordinates on the map.
# LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ll[2])$newX and
# LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ll[2])$newY give the
# lower left-corner coordinates of the static map. We use a fraction of
# them (.995, .61 respectively) to shift the legend to its default location.

```

```

# Define computed coordinates used on map (not RgoogleMaps/MyMap)
xcoord.max = LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ur[2])$newX
xcoord.min = LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ll[2])$newX
ycoord.max = LatLon2XY.centered(MyMap, MyMap$BBOX$ur[1], MyMap$BBOX$ll[2])$newY
ycoord.min = LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ll[2])$newY

# Legend for percent increase
legend(xcoord.min * .995, ycoord.min * .61,
       legend = c("[25, 45]", "[45, 65]", "[65, 85]", "[85, 100]"),
       fill = brewer.pal(4, "OrRd"), title = "Percent Increase", cex = .8,
       bg = "grey")

## Note: To put the legend in the top left corner we need change the Y and use
## MyMap$BBOX$ur[1] as the new reference point that gives us the Y coordinate
## in the "upper right" corner. But we just want Y (the latitude). So
## MyMap$BBOX$ll[2] stays the same. We also change the Y proportion to .995.

# legend(xcoord.min*.995, ycoord.max*.995,
# legend = c("[25, 45]", "[45, 65]", "[65, 85]", "[85, 100]"),
# fill = brewer.pal(4, "OrRd"), title = "Percent Increase", cex = .7,
# bg = "green")

## Note: To place the legend in the left-center, do the same, but change the
## proportion on the Y coordinate to .15.

# legend(xcoord.min*.995, ycoord.max*.15,
# legend = c("[25, 45]", "[45, 65]", "[65, 85]", "[85, 100]"),
# fill = brewer.pal(5, "Reds"), title = "Percent Increase", cex = .7,
# bg = "blue")

# 6. Create ray-glyph legend-----

# Define coordinates of ray-glyph legend box
x.left = xcoord.max*.6
y.bottom = ycoord.min*.961
x.right = xcoord.max*.99
y.top = ycoord.min*.45

# Convert rectangle x,y coordinates to latitude and longitude for use with MyMap
lat.bottom = XY2LatLon(MyMap, x.left, y.bottom)[1]
lon.left = XY2LatLon(MyMap, x.left, y.bottom)[2]
lat.top = XY2LatLon(MyMap, x.right, y.top)[1]
lon.right = XY2LatLon(MyMap, x.right, y.top)[2]

# Function to draw lines for the ray-glyph legend
CalcRayGlyphLegendLatLon = function(angle1, angle2, lat, lon) {

  rad1 = angle1 * pi / 180
  lon1 = lon + 2 * cos(rad1)
  lat1 = lat + 2 * sin(rad1)

  rad2 = angle2 * pi / 180
  lon2 = lon + 2 * cos(rad2)
  lat2 = lat + 2 * sin(rad2)

  list = c(lat1 = lat1, lon1 = lon1, lat2 = lat2, lon2 = lon2)
}

# Ray-glyph legend function. "cexln" is the cex of the line. "lwdln" is the
# line width. "ltyln1" and "ltyln2" are the respective line types.
PlotRayGlyphLegend = function(llabel, rlabel, llevels, rlevels, lncolor1,

```

```

lncolor2, circolor, title, levelscex = .8,
titlecex = .8, labelscex = .8, circex = cexcir,
lwdln = 2, lncircolor = "black") {

#title
text((x.right + x.left) / 2, y.top * 1.1, labels = title, bg = "gray",
     cex = labelscex)
# Left side variable
text((x.right + x.left) / 2 * .84, y.bottom * .58, labels = llabel,
     cex = labelscex)
# Right side variable
text((x.right + x.left) / 2 * 1.14, y.bottom * .58, labels = rlabel,
     cex = labelscex)
for (i in 0:4) {
  # Left lines
  PlotOnStaticMap(MyMap,
    lat = c(lat.bottom * 1.19 + i * 1.5,
      CalcRayGlyphLegendLatLon(270 - i * 45, 270 - i * 45,
        lat.bottom * 1.19 + i * 1.5,
        (lon.right + lon.left) / 2)[3]),
    lon = c((lon.right + lon.left)/2 - .1,
      CalcRayGlyphLegendLatLon(270 - i * 45, 270 - i * 45,
        lat.bottom * 1.15 + i * 1.5,
        (lon.right + lon.left) / 2)[4]),
    FUN = lines, col = lncolor1, add = TRUE, pch = pchcir, cex = circex,
    lwd = lwdln)
  # Right lines
  PlotOnStaticMap(MyMap,
    lat = c(lat.bottom * 1.19 + i * 1.5,
      CalcRayGlyphLegendLatLon(270 + i * 45, 270 + i * 45,
        lat.bottom * 1.19 + i * 1.5, (lon.right + lon.left) / 2)[3]),
    lon = c((lon.right + lon.left) / 2 + .1,
      CalcRayGlyphLegendLatLon(270 + i * 45, 270 + i * 45,
        lat.bottom * 1.15 + i * 1.5,
        (lon.right + lon.left) / 2)[4]),
    FUN = lines, col = lncolor2, add = TRUE, pch = pchcir, cex = circex,
    lwd = lwdln)
  # circles
  PlotOnStaticMap(MyMap, lat.bottom * 1.19 + i * 1.5,
    (lon.right + lon.left) / 2,
    FUN = points, col = circolor, add = TRUE, pch = pchdot, cex = circex)
  # outline circles
  PlotOnStaticMap(MyMap, lat.bottom * 1.19 + i * 1.5,
    (lon.right + lon.left) / 2,
    FUN = points, col = lncircolor, add = TRUE, pch = pchcir, cex = circex)
  # Left side levels
  text((x.right + x.left) / 2 * .84, y.top * 1.95 + i * 20,
    labels = llevels[1 + i], cex = labelscex)
  # Right side levels
  text((x.right + x.left) / 2 * 1.16, y.top * 1.95 + i * 20,
    labels = rlevels[1 + i], cex = labelscex)
}
}

# Rectangle background, with form
rect(x.left, y.bottom, x.right, y.top, col = "grey")

# Define values & generic labels
min1 = min(data[2])
max1 = max(data[2])
qtr1 = round((max1 - min1) * .25 + min1)
qtr2 = round((max1 - min1) * .5 + min1)
qtr3 = round((max1 - min1) * .75 + min1)
min2 = min(data[3])
max2 = max(data[3])
qtr21 = round((max2 - min2) * .25 + min2)

```

```

qtr22 = round((max2 - min2) * .5 + min2)
qtr23 = round((max2 - min2) * .75 + min2)
levelscex = .8
titlecex = .8
labelscex = .8

# Run the ray-glyph legend function. Change title and labels.

## Note: If labels run outside the bounds of the legend box you will have to
## manually adjust the rectangle background above.

PlotRayglyphLegend(llabel = ~underline(2000), rlabel = ~underline(2005),
                    title = "Tuition Trend from\n 2000 to 2005",
                    llevels = c(min1, qtr1, qtr2, qtr3, max1),
                    rlevels = c(min2, qtr21, qtr22, qtr23, max2),
                    circolor = palcircle[4],
                    lncolor1 = lncolor1,
                    lncolor2 = lncolor2
)

# 7. Plot title-----

# First shift down the margins so the title appears closer to the top of
# United States.

# par(mar = c(0, 0, 12.5, 0))
# title(main = "Ray-Glyph Plot Showing Cost of Tuition and\n Percent Increase from 2000 to 2005")
# \n starts a new line

dev.off()

```

B.3.1 Scatterplot and Residual Plot: University Tuition

```

# University Tuition Scatterplot and Plot of Residuals
# By          Nathan Voge
#
#
#          1 Import functions and data
#          2 Draw scatterplot
#          3 Plot residuals

# 1. Import needed functions and data-----

library(RColorBrewer)
library(RgoogleMaps)
library(PBSmapping)

University = c("Kansas", "Texas", "Arizona", "Ohio State", "Cal-Davis",
               "Cal-Santa Barbara", "Iowa State", "Minnesota", "UCLA", "Iowa",
               "Illinois", "Cal-San Diego", "Virginia", "Cal-Irvine", "Colorado",
               "Purdue", "Wisconsin", "North Carolina", "Penn State", "Pittsburgh",
               "Indiana", "Cal-Berkeley", "Nebraska", "Missouri", "Maryland",
               "Michigan State", "Washington", "Oregon", "Rutgers", "Michigan",
               "Florida", "Buffalo")

Year_2000 = c(2725, 3575, 2348, 4383, 4072, 3832, 3132, 4877, 3698, 3204, 4994, 3848, 4335,

```

```

3970,3188,3872,3791,2768,7018,7002,4405,4047,3450,4726,5136,5432,
3761,3819,6333,6926,2256,4715)

Year_2005 = c(5413,6972,4498,8082,7457,6997,5634,8622,6504,5612,8634,6685,
7370,6770,5372,6458,6284,4613,11508,11436,7112,6512,5540,7415,
7821,8108,5610,5613,9221,9798,3094,6068)

tuit = data.frame(University, Year_2000, Year_2005)

# read in data via a file, tuit = read.csv("tuition.csv", header = TRUE)

# 2. Draw scatterplot-----

# save it as a .pdf
pdf("tuit_18Jul_hist.pdf", width = 7)

plot(tuit$Year_2000, tuit$Year_2005, main = "", xlab = "Tuition in Year 2000",
      ylab = "Tuition in Year 2005")

dev.off()

# Scatterplot with names
# plot(tuit[, 2], tuit[, 3], type="n")
# text(tuit[, 2], tuit[, 3], tuit[, 1])

# 3. Plot residuals-----

glmmdat = glm(tuit$Year_2005 ~ tuit$Year_2000)
glmresid = residuals(glmmdat)
glmpred = predict(glmmdat)

# save it as a .pdf
pdf("tuit_22Jul_resid.pdf", width = 7)

plot(tuit$Year_2000, glmresid, type = "n", xlim = c(2000,7500), xlab = "Tuition in Year 2000",
      main = "", ylab = "Residuals", cex.lab = 1)

# optional title: Residual plot of Tuition in Year 2000

text(tuit$Year_2000, glmresid, tuit[, 1], cex = .9)

dev.off()

```

B.4 Rayglyph Googlemap: MLB Value vs Revenue

```

# Google Map Featuring Ray-Glyph Lines
# By          Nathan Voge (based on work by Daniel B. Carr (ray-glyphs) and
#              Markus Loecher (RgoogleMaps))
#
#
#          1 Import functions and data
#          2 Create a Google map
#          3 Add ray-glyph lines to the map
#          4 Add colored circles depicting percent increase to the map
#          5 Create percent increase legend
#          6 Create ray-glyph legend
#          7 Plot title

```



```

# 1. Import needed functions and data-----

# load in required packages
library(RColorBrewer)
library(RgoogleMaps)
library(PBSmapping)

# read in data
team = c('Arizona Diamondbacks','Atlanta Braves','Baltimore Orioles',
        'Boston Red Sox','Chicago Cubs','Chicago White Sox','Cincinnati Reds',
        'Cleveland Indians','Colorado Rockies','Detroit Tigers',
        'Florida Marlins','Houston Astros','Kansas City Royals',
        'Los Angeles Angels','Los Angeles Dodgers','Milwaukee Brewers',
        'Minnesota Twins','New York Mets','New York Yankees',
        'Oakland Athletics','Philadelphia Phillies','Pittsburgh Pirates',
        'San Diego Padres','San Francisco Giants','Seattle Mariners',
        'St Louis Cardinals','Tampa Bay Devil Rays','Texas Rangers',
        'Toronto Blue Jays','Washington Nationals')

value = c(305,405,359,617,448,315,274,352,298,292,226,416,239,368,482,235,216,
        604,1026,234,424,250,354,410,428,429,209,353,286,440)

revenue = c(145,172,156,206,179,157,137,150,145,146,119,173,117,167,189,131,114,
        195,277,134,176,125,158,171,179,165,116,153,136,145)

lat = c(33.445278,33.735278,39.583889,42.346389,42.42.53,39.0975,41.495833,
        39.756111,42.339167,25.778056,29.756944,39.051389,35.00278,34.073611,
        43.948333,44.981667,40.756944,41.829167,37.951667,39.905833,40.446944,
        32.7073,37.778333,47.591389,38.6225,27.768333,32.751389,43.641389,
        38.872778)

lon = c(-112.066944,-84.389444,-76.621667,-71.0975,-87.655556,-86.7,-84.506667,
        -81.685278,-104.994167,-83.048611,-80.219722,-95.355556,-94.480556,
        -119.32778,-119.24,-87.971111,-93.278333,-73.845833,-73.926389,
        -121.200556,-75.166389,-80.005833,-117.1566,-122.389444,-122.3325,
        -90.193056,-82.653333,-97.082778,-79.389167,-77.0075)

data = data.frame(team, value, revenue, lat, lon)

# read in data via a file, data = read.csv("mlb_coordchg.csv", header = TRUE)

# 2. Create a Google map-----

# Define x and y coordinates on map. In all RgoogleMaps packages, the latitude
# is read first followed by the longitude. For our data set, we looked up the
# latitude and longitude online for each university.
x = data["lat"]
y = data["lon"]

# Define the bounding box (bb) for the map using the latitude and longitude from
# our data set.
bb = qbbox(x, y, TYPE = "all", margin = list(m = rep(5, 4),
        TYPE = c("perc", "abs")[1]))

# Create a Google map by loading a static map from the package and define
# some options. bb$lonR, bb$latR are our latitude and longitude from the
# previous line of code. We have a few choices on the type of map we want to use
# such as "roadmap", "satellite", "terrain", etc. The default is a color map,
# here we want it to be gray so we use the "GRAYSCALE" argument. If needed, we
# can also use the "zoom" argument. This zooms in/out around the area we
# define by using the argument "center".

# save it as a .pdf (looks better)

```

```

pdf("mlb_18Jul_map.pdf", width = 7, paper = "USr")

MyMap = GetMap.bbox(bb$lonR, bb$latR, maptype = "roadmap", GRAYSCALE = TRUE)

# This command plots the map in the graphics window in R
PlotOnStaticMap(MyMap)

## Note: Content is added to the static map by using "PlotOnStaticMap".
# "PlotOnStaticMap" is repeated for each color/item added to the map. "FUN"
# identifies the plotting function, usually points or lines. "pch" and "cex"
# normal plotting parameters.

# 3. Add Ray-glyph lines to the map-----

# Function to Calculate the Latitude and Longitude for Ray-glyph Lines
# It uses the two variables you want to compare (var1, var2), the latitude
# and longitude for these variables (lat, lon), and the minimum and maximum
# of the two variables (min1, max1, min2, max2) respectively. It returns the
# latitude and longitude of the point away from the origin for each variable.

CalcRayGlyphMapLatLon = function(var1, var2, lat, lon, min1, max1, min2,
                                max2) {
  num1 = var1 - min1
  ratio1 = num1 / (max1 - min1)
  angle1 = 270 - ratio1 * 180
  rad1 = angle1 * pi / 180
  lon1 = lon + 2 * cos(rad1)
  lat1 = lat + 2 * sin(rad1)

  num2 = var2 - min2
  ratio2 = num2 / (max2 - min2)
  angle2 = 270 + ratio2 * 180
  rad2 = angle2 * pi / 180
  lon2 = lon + 2 * cos(rad2)
  lat2 = lat + 2 * sin(rad2)

  list = c(lat1 = lat1, lon1 = lon1, lat2 = lat2, lon2 = lon2)
}

# Function to Plot Ray-glyph lines.
# "variable" is the plotting parameter. "lncolor1" is the color of the left
# line. "lncolor2" is the color of the right line. "cexln" is the cex of the
# line. "lwdln" is the line width. "ltyln1" and "ltyln2" and the respective
# line types.

PlotRayGlyphLines = function(data, variable, lncolor1, lncolor2, cexln = 1.5,
                             lwdln = 2, ltyln1 = 1, ltyln2 = 1) {
  colselect = subset(data, select = variable)
  for (i in 1:nrow(colselect)) {
    loc = data[colselect[i, ], ] # select each location
    PlotOnStaticMap(MyMap,
      lat = c(as.numeric(loc[4]),
        as.numeric(CalcRayGlyphMapLatLon(loc[2], loc[3], loc[4], loc[5],
          min(data[2]), max(data[2]), min(data[3]),
          max(data[3]))[1])),
      lon = c(as.numeric(loc[5]) - .1,
        as.numeric(CalcRayGlyphMapLatLon(loc[2], loc[3], loc[4], loc[5],
          min(data[2]), max(data[2]), min(data[3]),
          max(data[3]))[2]) - .1),
      FUN = lines, col = lncolor1, add = TRUE, pch = 1, cex = cexln, lwd = lwdln,
      lty = ltyln1) #Line on left
    PlotOnStaticMap(MyMap,
      lat = c(as.numeric(loc[4]),
        as.numeric(CalcRayGlyphMapLatLon(loc[2], loc[3], loc[4], loc[5],

```

```

        min(data[2]), max(data[2]), min(data[3]),
        max(data[3]))[3])),
lon = c(as.numeric(loc[5]) + .1,
        as.numeric(CalcRayGlyphMapLatLon(loc[2], loc[3], loc[4], loc[5],
        min(data[2]), max(data[2]), min(data[3]),
        max(data[3]))[4]) + .1),
        FUN = lines, col = lncolor2, add = TRUE, pch = 1, cex = cexln, lwd = lwdln,
        lty = ltyln2) #Line on right
    }
}

# Define color scheme for ray-glyph lines
palgreen = brewer.pal(9, "Greens")
lncolor1 = palgreen[8]
lncolor2 = palgreen[5]

# Run PlotRayGlyphLines function and plot ray-glyph lines. Change plotting
# variable. ("team" here)
PlotRayGlyphLines(data = data, variable = "team", lncolor1, lncolor2)

# 4. Add colored circles depicting percent increase to the map-----

# Function to Plot Colored Circles Outlined in Black.
# Data*[4] and data*[5] contain the latitude and longitude for each
# respective level (*), which are our breaks in the data.
# After many tries, we found 4 levels most useful
# for our data set. "subset" allows us to identify how we want to divide up the
# data with "Percent_Increase" being our variable of interest.
# Here we put black circles around the colors to make them more visible.
# "pch = 1" is a circle. "cexcir" is the cex of the circles. "pchdot" is the pch
# of the dot. "pchcir" is the pch of the circle

## Note: The data in "Percent_Increase" was not provided by the original data
## set.

PlotPoints = function(data, num.breaks, cexcir = 1.5, pchdot = 19, pchcir = 1) {
  for (i in 1:num.breaks) {
    if (i == 1) {#i=1; variable = "value"
      data1 = subset(data, value >= 200 & value < 400)
      PlotOnStaticMap(MyMap, data1[, 4], data1[, 5], FUN = points,
        col = palcircle[i], add = TRUE, pch = pchdot, cex = cexcir)
      PlotOnStaticMap(MyMap, data1[, 4], data1[, 5], FUN = points, col = "black",
        add = TRUE, pch = pchcir, cex = cexcir, lwd = 1)
    }
    else if (i == 2) {#i=2; variable = "value"
      data2 = subset(data, value >= 400 & value < 600)
      PlotOnStaticMap(MyMap, data2[, 4], data2[, 5], FUN = points,
        col = palcircle[i], add = TRUE, pch = pchdot, cex = cexcir)
      PlotOnStaticMap(MyMap, data2[, 4], data2[, 5], FUN = points, col = "black",
        add = TRUE, pch = pchcir, cex = cexcir, lwd = 1)
    }
    else if (i == 3) {
      data3 = subset(data, value >= 600 & value < 800)
      PlotOnStaticMap(MyMap, data3[, 4], data3[, 5], FUN = points,
        col = palcircle[i], add = TRUE, pch = pchdot, cex = cexcir)
      PlotOnStaticMap(MyMap, data3[, 4], data3[, 5], FUN = points, col = "black",
        add = TRUE, pch = pchcir, cex = cexcir, lwd = 1)
    }
    else if (i == 4) {
      data4 = subset(data, value >= 800 & value < 1000)
      PlotOnStaticMap(MyMap, data4[, 4], data4[, 5], FUN = points,

```

```

        col = palcircle[i], add = TRUE, pch = pchdot, cex = cexcir)
    PlotOnStaticMap(MyMap, data4[,4], data4[,5], FUN = points,
        col = "black", add = TRUE, pch = pchcir, cex = cexcir,
        lwd = 1)
  }
  else if (i == 5) {
    data5 = subset(data, value >= 1000 & value < 1200)
    PlotOnStaticMap(MyMap, data5[,4], data5[,5], FUN = points,
        col = palcircle[i], add = TRUE, pch = pchdot, cex = cexcir)
    PlotOnStaticMap(MyMap, data5[,4], data5[,5], FUN = points,
        col = "black", add = TRUE, pch = pchcir, cex = cexcir,
        lwd = 1)
  }
}
}

# Run PlotPoints function and plot dots/circles.

# Define color scheme for our circles. We used the website
# http://colorbrewer2.org to identify an appropriate color scheme with 4 levels.
# variable = "value"
num.breaks = 5
palcircle = brewer.pal(5, "OrRd")
cexcir = 1.5
pchdot = 19
pchcir = 1

# Run colored circles function
PlotPoints(data, num.breaks, cexcir, pchdot, pchcir)

## Note: If data doesn't have a percent increase variable, use the below code
## to plot the circles

# PlotOnStaticMap(MyMap, data[,4], data[,5], FUN = points,
#                 col = palcircle[4], add = TRUE, pch = pchdot, cex = cexcir)
# PlotOnStaticMap(MyMap, data[,4], data[,5], FUN = points,
#                 col = "black", add = TRUE, pch = pchcir, cex = cexcir,
#                 lwd = 1)

# 5. Create percent increase legend_____

# The x and y coordinates of the legend box are based on
# the latitude and longitude of the static map from Google. The
# function "LatLon2XY.centered" transforms latitude and longitude into x and
# y coordinates on the map.
# LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ll[2])$newX and
# LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ll[2])$newY give the
# lower left-corner coordinates of the static map. We use a fraction of
# them (.995, .61 respectively) to shift the legend to its default location.

# Define computed coordinates used on map (not RgoogleMaps/MyMap)
xcoord.max = LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ur[2])$newX
xcoord.min = LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ll[2])$newX
ycoord.max = LatLon2XY.centered(MyMap, MyMap$BBOX$ur[1], MyMap$BBOX$ll[2])$newY
ycoord.min = LatLon2XY.centered(MyMap, MyMap$BBOX$ll[1], MyMap$BBOX$ll[2])$newY

# Legend for percent increase
legend(xcoord.min * .995, ycoord.min * .57,
      legend = c("[200, 400]", "[400, 600]", "[600, 800]", "[800, 1000]",
        "[1000, 1200]"),
      fill = palcircle, title = "Value Distribution ($mil)", cex = .9,
      bg = "grey")

```

```

## Note: To put the legend in the top left corner we need change the Y and use
## MyMap$BBOX$ur[1] as the new reference point that gives us the Y coordinate
## in the "upper right" corner. But we just want Y (the latitude). So
## MyMap$BBOX$ll[2] stays the same. We also change the Y proportion to .995.

# legend(xcoord.min*.995, ycoord.max*.995,
# legend = c("[25, 45)", "[45,65)", "[65, 85)", "[85, 100)"),
# fill = brewer.pal(4, "OrRd"), title = "Percent Increase", cex = .7,
# bg = "green")

## Note: To place the legend in the left-center, do the same, but change the
## proportion on the Y coordinate to .15.

# legend(xcoord.min*.995, ycoord.max*.15,
# legend = c("[25, 45)", "[45,65)", "[65, 85)", "[85, 100)"),
# fill = brewer.pal(5, "Reds"), title = "Percent Increase", cex = .7,
# bg = "blue")

# 6. Create ray-glyph legend-----

# Define coordinates of ray-glyph legend box
x.left = xcoord.max*.6
y.bottom = ycoord.min*.961
x.right = xcoord.max*.99
y.top = ycoord.min*.45

# Convert rectangle x,y coordinates to latitude and longitude for use with MyMap
lat.bottom = XY2LatLon(MyMap, x.left, y.bottom)[1]
lon.left = XY2LatLon(MyMap, x.left, y.bottom)[2]
lat.top = XY2LatLon(MyMap, x.right, y.top)[1]
lon.right = XY2LatLon(MyMap, x.right, y.top)[2]

# Function to draw lines for the ray-glyph legend
CalcRayGlyphLegendLatLon = function(angle1, angle2, lat, lon) {

  rad1 = angle1 * pi / 180
  lon1 = lon + 2 * cos(rad1)
  lat1 = lat + 2 * sin(rad1)

  rad2 = angle2 * pi / 180
  lon2 = lon + 2 * cos(rad2)
  lat2 = lat + 2 * sin(rad2)

  list = c(lat1 = lat1, lon1 = lon1, lat2 = lat2, lon2 = lon2)
}

# Ray-glyph legend function. "cexln" is the cex of the line. "lwdln" is the
# line width. "ltyln1" and "ltyln2" are the respective line types.
PlotRayGlyphLegend = function(llabel, rlabel, llevels, rlevels, lncolor1,
                             lncolor2, circolor, title, levelscex = .9,
                             titlecex = .9, labelscex = .9, circex = cexcir,
                             lwdln = 2, lncircolor = "black") {

  #title
  text((x.right + x.left) / 2, y.top * 1.1, labels = title, bg = "gray",
       cex = labelscex)
  # Left side variable
  text((x.right + x.left) / 2 * .84, y.bottom * .58, labels = llabel,
       cex = labelscex)
  # Right side variable
  text((x.right + x.left) / 2 * 1.14, y.bottom * .58, labels = rlabel,
       cex = labelscex)

```

```

for (i in 0:4) {
  # Left lines
  PlotOnStaticMap(MyMap,
    lat = c(lat.bottom * 1.19 + i * 1.5,
      CalcRayGlyphLegendLatLon(270 - i * 45, 270 - i * 45,
        lat.bottom * 1.19 + i * 1.5,
        (lon.right + lon.left) / 2)[3]),
    lon = c((lon.right + lon.left)/2 - .1,
      CalcRayGlyphLegendLatLon(270 - i * 45, 270 - i * 45,
        lat.bottom * 1.15 + i * 1.5,
        (lon.right + lon.left) / 2)[4]),
    FUN = lines, col = lncolor1, add = TRUE, pch = pchcir, cex = circex,
    lwd = lwdln)
  # Right lines
  PlotOnStaticMap(MyMap,
    lat = c(lat.bottom * 1.19 + i * 1.5,
      CalcRayGlyphLegendLatLon(270 + i * 45, 270 + i * 45,
        lat.bottom * 1.19 + i * 1.5, (lon.right + lon.left) / 2)[3]),
    lon = c((lon.right + lon.left) / 2 + .1,
      CalcRayGlyphLegendLatLon(270 + i * 45, 270 + i * 45,
        lat.bottom * 1.15 + i * 1.5,
        (lon.right + lon.left) / 2)[4]),
    FUN = lines, col = lncolor2, add = TRUE, pch = pchcir, cex = circex,
    lwd = lwdln)
  # circles
  PlotOnStaticMap(MyMap, lat.bottom * 1.19 + i * 1.5,
    (lon.right + lon.left) / 2,
    FUN = points, col = circolor, add = TRUE, pch = pchdot, cex = circex)
  # outline circles
  PlotOnStaticMap(MyMap, lat.bottom * 1.19 + i * 1.5,
    (lon.right + lon.left) / 2,
    FUN = points, col = lncircolor, add = TRUE, pch = pchcir, cex = circex)
  # Left side levels
  text((x.right + x.left) / 2 * .84, y.top * 1.95 + i * 20,
    labels = llevels[1 + i], cex = labelsxex)
  # Right side levels
  text((x.right + x.left) / 2 * 1.16, y.top * 1.95 + i * 20,
    labels = rlevels[1 + i], cex = labelsxex)
}
}

# Rectangle background, with form
rect(x.left, y.bottom, x.right, y.top-28, col = "grey")

# Define values & generic labels
min1 = min(data[2])
max1 = max(data[2])
qtr1 = round((max1 - min1) * .25 + min1)
qtr2 = round((max1 - min1) * .5 + min1)
qtr3 = round((max1 - min1) * .75 + min1)
min2 = min(data[3])
max2 = max(data[3])
qtr21 = round((max2 - min2) * .25 + min2)
qtr22 = round((max2 - min2) * .5 + min2)
qtr23 = round((max2 - min2) * .75 + min2)

# Run the ray-glyph legend function. Change title and labels.

## Note: If labels run outside the bounds of the legend box you will have to
## manually adjust the rectangle background above.

PlotRayGlyphLegend(llabel = ~underline(Value), rlabel = ~underline(Revenue),
  title = "",
  llevels = c(min1, qtr1, qtr2, qtr3, max1),

```

```

        rlevels = c(min2, qtr21, qtr22, qtr23, max2),
        circolor = palcircle[4],
        lncolor1 = lncolor1,
        lncolor2 = lncolor2
    )

# 7. Plot title-----

# First shift down the margins so the title appears closer to the top of
# United States.

# par(mar = c(0, 0, 12.5, 0))
# title(main = "Ray-Glyph Plot Showing Major League Baseball Valuations \nand
#         Revenue ($mil) in 2007 (Forbes.com)" # \n starts a new line

dev.off()

```

B.4.1 Histogram: MLB Value and Revenue

```

# MLB Histogram
# By          Nathan Voge
#
#
#           1 Import functions and data
#           2 Draw plots, define colors, create breaks

# 1. Import needed functions and data-----

library(RColorBrewer)
library(RgoogleMaps)
library(PBSmapping)

value = c(305,405,359,617,448,315,274,352,298,292,226,416,239,368,482,235,216,
        604,1026,234,424,250,354,410,428,429,209,353,286,440)

revenue = c(145,172,156,206,179,157,137,150,145,146,119,173,117,167,189,131,114,
        195,277,134,176,125,158,171,179,165,116,153,136,145)

mlb = data.frame(value, revenue)

# read in data via a file, data = read.csv("mlb_coordchg.csv", header = TRUE)

# 2. Draw plots, define colors, create breaks-----

# save it as a .pdf
pdf("mlb_18Jul_valhist.pdf", width = 11, paper = "USr")

value.breaks5 = c(200, 400, 600, 800, 1000, 1200)
palette5 = brewer.pal(5, "OrRd") # define colors
hist(mlb$value, xlim = c(200, 1200), ylim = c(0, 20), xlab = "Value ($mil)",
     main = "", col = palette5, breaks = value.breaks5)

dev.off()

```

```

# save it as a .pdf
pdf("mlb_18Jul_revhist.pdf", width = 11, paper = "USr")

revenue.breaks5 = seq(100, 300, 40)
hist(mlb$revenue, main = "", col = palette5, xlab = "Revenue ($mil)",
     breaks = revenue.breaks5, ylim = c(0, 20), xlim = range(revenue.breaks5),
     xaxt = "n") # xaxt = "n" turns off the numbering for the x-axis

# reformat the x-axis with tick marks that fit our breaks
axis(1, at = revenue.breaks5, labels = c("100", "140", "180", "220", "260",
                                         "300"))

dev.off()

# Extra histograms
#value.breaks2 = c(200, 350, 500, 650, 800, 950, 1100)
#palette6 = brewer.pal(7, "OrRd")
#hist(mlb$value, xlim = c(200, 1200), ylim = c(0, 15), xlab = "Value ($mil)",
#     main = "", col = palette6, breaks = value.breaks2)
#
#palette7 = brewer.pal(9, "OrRd")
#hist(mlb$value, xlim = c(200, 1200), ylim = c(0, 12), xlab = "Value ($mil)",
#     main = "", col = palette7)

##Revenue
#palette9 = brewer.pal(9, "OrRd")
#hist(mlb$revenue, main = "", col = palette9, xlab = "Revenue ($mil)",
#     xlim = c(100, 300), ylim = c(0, 10))

```

B.5 One-Panel Linked Micromap: Tornado Damage

```

# Linked micromap plots (one statistical panel)
# By Mike Minnotte (revised for current data by Nathan Voge)
#
#
# 1 Import functions and data
# 2 Extract state names and sort the data
# 3 Colors and Graphics Device

# 1. Import needed functions and data-----

# read in data
# web link to data: http://sciencepolicy.colorado.edu/sourcebook/tornadoes.html

State = c("Alabama", "Arizona", "Arkansas", "California", "Colorado", "Connecticut",
          "Delaware", "Florida", "Georgia", "Idaho", "Illinois", "Indiana", "Iowa",
          "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts",
          "Michigan", "Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska",
          "Nevada", "New Hampshire", "New Jersey", "New Mexico", "New York",
          "North Carolina", "North Dakota", "Ohio", "Oklahoma", "Oregon",
          "Pennsylvania", "Rhode Island", "South Carolina", "South Dakota",
          "Tennessee", "Texas", "Utah", "Vermont", "Virginia", "Washington",
          "West Virginia", "Wisconsin", "Wyoming")
# Puerto Rico, Hawaii, Alaska were eliminated from the data set as

```



```

# they aren't included on the map

Cost = c(51.88,3.469,40.96,3.682,4.623,2.26,0.2744,37.32,51.68,0.2552,62.94,
        53.13,49.51,49.28,24.84,27.75,0.5252,2.329,4.418,29.88,84.84,43.62,
        68.93,2.266,30.26,0.0953,0.6592,2.94,1.485,15.73,14.9,14.69,44.36,
        81.94,5.52,17.11,0.0898,17.19,10.64,23.47,88.6,3.565,0.2416,7.416,
        2.374,2.143,31.33,1.779)

torn = data.frame(State, Cost)

# read in data via file, torn = read.csv("tornado_alpha.csv", header = TRUE)

library(RColorBrewer)
library(maps) # load map data

data(state) # load state data

# 2. Extract state names and sort the data (high to low)-----

tornado = torn[, 2] # extract data
tornado.state = torn[, 1] # get state names
tornado.name <- tornado.state[order(tornado, decreasing = T)]
tornado = sort(tornado, decreasing = TRUE)

# 3. Start Graphics Device and Define Colors-----

pal = brewer.pal(6, "Set1") # rainbow colors
pal[6] = "#DDDDDD" # gray

# save it as a .pdf (looks better)
pdf("Tornado_micromap_18Jul.pdf", width = 7.5, height = 10)

# create the layout
layout(matrix(1:36, nrow = 12, byrow = TRUE), widths = c(1, 1, 2),
        heights = c(rep(4, 5), 1, rep(4, 5), 3))

# visualize the layout: layout.show(36)

# loop that draws maps and plot everything
for (i in 1:10) { # start of loop for each of the i panels, ends on line 159

  # compute colors
  par(mex = 0.5, mar = rep(.01, 4))

  # define colors above/below median to be gray (pal[6])
  if (i <= 5)
    m.col = c(rep(pal[6], 25), rep(0, 25))
  else
    m.col = c(rep(0, 25), rep(pal[6], 25))

  # define other colors on map (column 1)
  if (i <= 4)
    m.col[(i-1) * 5 + 1:5] = pal[1:5]
  else if (i == 5)
    m.col[(i-1) * 5 + 1:4] = pal[1:4]
  else if (i == 6)
    m.col[(i-1) * 5 - 1 + 1:4] = pal[1:4]
  else
    m.col[(i-1)*5 - 2 + 1:5] = pal[1:5]

  map.m.col = m.col[match.map("state", as.character(tornado.name))]

```

```

# plot map
map("state", fill = TRUE, col = map.m.col, border = 1,
    xlim = c(-125, -65), ylim = c(25, 50))

# plot labels (column 2)
par(mar = rep(.1, 4))
plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n", bty = "n",
     xaxt = "n", yaxt = "n", xlab = "", ylab = "")

if (i <= 4)
  points(rep(.1, 5), seq(.9, .1, by = -.2), pch = 21, bg = pal[1:5], cex = 2)
else if (i == 5) # notice in i==5/6, we change seq(...) and pal[...] because
                  # there are only 4 states/dots here. This is repeated below.
  points(rep(.1, 4), seq(.8, .2, by = -.2), pch = 21, bg = pal[1:4], cex = 2)
else if (i == 6)
  points(rep(.1, 4), seq(.8, .2, by = -.2), pch = 21, bg = pal[1:4], cex = 2)
else
  points(rep(.1, 5), seq(.9, .1, by = -.2), pch = 21, bg = pal[1:5], cex = 2)

if (i <= 4)
  text(rep(.18, 5), seq(.9, .1, by = -.2), tornado.name[(i-1) * 5 + 1:5],
       pos = 4, cex = 1.0)
else if (i == 5)
  text(rep(.18, 4), seq(.8, .2, by = -.2), tornado.name[(i-1) * 5 + 1:4],
       pos = 4, cex = 1.0)
else if (i == 6)
  text(rep(.18, 4), seq(.8, .2, by = -.2), tornado.name[(i-1) * 5 - 1 + 1:4],
       pos = 4, cex = 1.0)
else
  text(rep(.18, 5), seq(.9, .1, by = -.2), tornado.name[(i-1) * 5 - 2 + 1:5],
       pos = 4, cex = 1.0)

# plot dotplot of values (column 3)
par(mar = rep(.1, 4))
if (i == 10)
  plot(0, 0, xlim = range(tornado), ylim = c(0, 1), type = "n", yaxt = "n",
       xlab = "", ylab = "")
else
  plot(0, 0, xlim = range(tornado), ylim = c(0, 1), type = "n", xaxt = "n",
       yaxt = "n", xlab = "", ylab = "")

if (i <= 4)
  abline(h = seq(.9, .1, by = -.2), lty = 3, col = "grey")
else if (i == 5)
  abline(h = seq(.8, .2, by = -.2), lty = 3, col = "grey")
else if (i == 6)
  abline(h = seq(.8, .2, by = -.2), lty = 3, col = "grey")
else
  abline(h = seq(.9, .1, by = -.2), lty = 3, col = "grey")

if (i <= 4)
  points(tornado[(i-1) * 5 + 1:5], seq(.9, .1, by = -.2), pch = 21,
        bg = pal[1:5], cex = 2)
else if (i == 5)
  points(tornado[(i-1) * 5 + 1:4], seq(.8, .2, by = -.2), pch = 21,
        bg = pal[1:4], cex = 2)
else if (i == 6)
  points(tornado[(i-1) * 5 - 1 + 0:3], seq(.8, .2, by = -.2), pch = 21,
        bg = pal[1:4], cex = 2)
else
  points(tornado[(i-1) * 5 - 2 - 2 + 1:5], seq(.9, .1, by = -.2), pch = 21,
        bg = pal[1:5], cex = 2)

# separate states above and below median, j is the columns, i the rows
if (i == 5) {

```

```

    for (j in 1:3){
      plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n", bty = "n",
           xaxt = "n", yaxt = "n", xlab = "", ylab = "")
      abline(h = .5, lwd = 3, col = pal[6])
    }
  }

# Plot through remaining (empty) cells
if (i == 10)
  for (j in 1:3)
    plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = "n", bty = "n",
         xaxt = "n", yaxt = "n", xlab = "", ylab = "")
} # end of loop

# Label for dot plots
text(0.5, 0.35, "Property Damage Due to \nTornadoes in Millions of Dollars",
     cex = 1.5)

dev.off()

```

B.5.1 Tornado Histogram

```

# Tornado Damage Histogram
# By          Nathan Voge
#
#
#           1 Import functions and data
#           2 Draw plot

# 1. Import needed functions and data_____

State = c("Alabama", "Arizona", "Arkansas", "California", "Colorado", "Connecticut",
          "Delaware", "Florida", "Georgia", "Idaho", "Illinois", "Indiana", "Iowa",
          "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts",
          "Michigan", "Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska",
          "Nevada", "New Hampshire", "New Jersey", "New Mexico", "New York",
          "North Carolina", "North Dakota", "Ohio", "Oklahoma", "Oregon",
          "Pennsylvania", "Rhode Island", "South Carolina", "South Dakota",
          "Tennessee", "Texas", "Utah", "Vermont", "Virginia", "Washington",
          "West Virginia", "Wisconsin", "Wyoming")
# Puerto Rico, Hawaii, and Alaska were eliminated

Cost = c(51.88, 3.469, 40.96, 3.682, 4.623, 2.26, 0.2744, 37.32, 51.68, 0.2552, 62.94,
         53.13, 49.51, 49.28, 24.84, 27.75, 0.5252, 2.329, 4.418, 29.88, 84.84, 43.62,
         68.93, 2.266, 30.26, 0.0953, 0.6592, 2.94, 1.485, 15.73, 14.9, 14.69, 44.36,
         81.94, 5.52, 17.11, 0.0898, 17.19, 10.64, 23.47, 88.6, 3.565, 0.2416, 7.416,
         2.374, 2.143, 31.33, 1.779)

torn = data.frame(State, Cost)

# read in data via a file, torn = read.csv("tornado_alpha.csv", header = TRUE)

# 2. Draw plot_____

# save as a .pdf
pdf("torn_18Jul_hist.pdf", width = 11, paper = "USr")

```

```
hist(torn$Cost, xlim = c(0, 100), ylim = c(0, 25), main = "",
     xlab = "Cost (in millions)")

dev.off()
```

B.6 Two-Panel Linked Micromap: Fruit/Vegetable Consumption and Smoking

```
# Linked micromap plots (two statistical panels)
# By          Daniel B. Carr (revised for current data by Nathan Voge)
#
#
#           1 Import functions and data
#           2 Sort data and extract abbreviations
#           3 Colors and Graphics Device
#           4 Define panel layouts
#
#           5 Define graphics parameters
#           6 Set up for processing in loops
#           7 Draw maps
#           8 Draw linking dots and state names
#
#           9 Draw estimates and confidence bounds
#          10 Draw counts
#          11 Draw title and legend
#          12 Outline groups of panels

# 1. Import needed functions and data-----

# read in data
fvsmk = read.csv("fvsmk_DC.csv", row.names = 1, header = TRUE) # row.names = 1
                                                                # makes row
                                                                # numbers go
                                                                # away

load("Ch9_panelLayout.Rdata") # workspace that has all the panel layout
                              # functions

# read in polygon shapes for map
stateVBorders = read.csv("stateVisibilityBorders.txt", row.names = NULL,
                        header = TRUE) # "st" = state ID's, "x" and "y" are
                                      # polygon coordinates
nationVBorders = read.csv("nationVisibilityBorders.txt",
                          blank.lines.skip = FALSE, row.names = NULL,
                          header = TRUE)

# get Fips codes which are numbers the government uses to identify states
stateNamesFips = read.csv('stateNamesFips.txt', row.names = 1, header = TRUE)

# 2. Sort the data.frame by fruit and veg consumption (high to low) and
#    extract the state abbreviations for later use-----

ord = rev(order(fvsmk$Fruit_Vegs))
fvsmk = fvsmk[ord, ]
```

```

stateDataId = row.names(fvsmk)

# 3. Start Graphics Device and Define Colors-----

#windows(width = 7.5,height = 10)

wgray = rgb(.82, .82, .82) #white gray

rgbColors = matrix(c(
  1.00, .30, .30,
  1.00, .50, .00,
  .25,1.00, .25,
  .10, .65, 1.00,
  .80, .45, 1.00,
  .35, .35, .35,
  .85, .85, .85,
  .50, .50, .50,
  1.00, 1.00, .85,
  .85, .85, .85), ncol = 3, byrow = TRUE)
hdColors = rgb(rgbColors[, 1], rgbColors[, 2], rgbColors[, 3])

#4. Define panel layouts-----

bot = .77
top = .73
left = 0
right = 0
panels = panellayout(nrow = 11, ncol = 4,
  topMar = top, bottomMar = bot,
  leftMar = left, rightMar = right,
  rowSep = c(0, 0, 0, 0, 0, .07, .07, 0, 0, 0, 0, 0),
  rowSize = c(7, 7, 7, 7, 7, 1.5, 7, 7, 7, 7, 7),
  colSize = c(2.5, 2.2, 2.90, 2.90),
  colSep = c(0, 0, 0, 0))

panelBlock = panellayout(nrow = 3, ncol = 3,
  topMar = top, bottomMar = bot,
  leftMar = left, rightMar = right,
  rowSep = c(0, .07, .07, 0),
  rowSize = c(35, 1.5, 35),
  colSize = c(4.7, 2.90, 2.90),
  colSep = c(0, 0, 0, 0))

#5. Define graphics parameters-----

dcex = .95
tCex = 1.08
cex = .65
fontsize = 12
font = 1
line1 = .2
line2 = 1.0
line3 = .2
ypad = .65
nameShift = .12

```

```

#6. Set up indices for perceptual groups of states-----

iBegin = c(1, 6, 11, 16, 21, 26, 27, 32, 37, 42, 47) # group beg subscript
iEnd   = c(5, 10, 15, 20, 25, 26, 31, 36, 41, 46, 51) # group ending subscript
nGroups = length(iEnd)

#7. Plot maps-----

# range for scale the polygon panels
rxpoly = range(stateVBorders$x, na.rm = TRUE)
rypoly = range(stateVBorders$y, na.rm = TRUE)

# polygon ID's, one per polygon
polygonId = stateVBorders$st[is.na(stateVBorders$x)]

# panel titles
panelSelect(panels, 1, 1)
panelScale()
mtext(side = 3, line = line1, 'Above Median States', cex = cex)
mtext(side = 3, line = line2, 'Micromaps', cex = cex)

panelSelect(panels, 11, 1)
panelScale()
mtext(side = 1, line = line3, 'Below Median States', cex = cex)

# drawing the maps
for (i in 1:nGroups) {
  if (i == 6) { #map not drawn, median state in adjacent panels
    panelSelect(panels, 6, 1)
    panelScale()
    panelFill(col = wgray)
    panelOutline(col = "black")
    text(.5, .55, 'Median', cex = cex)
    next # 'next' halts the processing of the current iteration and advances
        # the looping index.
  }
  panelSelect(panels, i, 1)
  panelScale(rxpoly, rypoly)
  gsubs = iBegin[i]:iEnd[i]
  if (i == 5)
    gsubs = c(gsubs, 26) # median state added here
  if (i == 7)
    gsubs = c(gsubs, 26) # median state added here
  if (i < 6)
    cont = stateDataId[1:26]
  else
    cont = stateDataId[26:51]
    # cont means above or below median contour
  panelNames = stateDataId[gsubs]

# plot background (out of contour) states in gray with white outlines
back = is.na(match(stateVBorders$st, cont))
polygon(stateVBorders$x[back], stateVBorders$y[back], col = wgray,
        border = FALSE)
polygon(stateVBorders$x[back], stateVBorders$y[back], col = "white",
        density = 0)

# outline states

# plot foreground states for the panel in their special colors pens 1:5

```

```
# and other in contour states in light yellow pen 9
fore = !back
pen = match(polygonId, panelNames, nomatch = 9)[!is.na(match(polygonId, cont))]
polygon(stateVBorders$x[fore], stateVBorders$y[fore], col = hdColors[pen],
        border = FALSE) # outline states
polygon(stateVBorders$x[fore], stateVBorders$y[fore], col = "black",
        density = 0, lwd = 1) # outline states

# outline U.S.
polygon(nationVBorders$x, nationVBorders$y, col = "black", density = 0,
        lwd = 1)

# outside boundary
}
```

#8. Plot labels_____

```
# get full state names in rate order
ord = match(stateDataId, stateNamesFips$ab) # match state names
stateNames = row.names(stateNamesFips)[ord] # get the full state names except
# D.C.
cbind(stateDataId, stateNames) # check that the matching has
# worked

# title column
panelSelect(panels, 1, 2)
panelScale()
mtext(side = 3, line = line1, '', cex = cex)
mtext(side = 3, line = line2, 'States', cex = cex)

# draw state names
for (i in 1:nGroups) {
  gsubs = iBegin[i]:iEnd[i]
  gnams = stateNames[gsubs]
  nsubs = length(gnams)
  pen = 1:nsubs
  laby = nsubs:1
  panelSelect(panels, i, 2)
  panelScale(c(0, 1), c(1-ypad, nsubs + ypad))

  if (i == 6) {
    pen = 6
    panelFill(col = wgray)
    panelOutline()
  }

  for (j in 1:length(pen)) {
    points(.1, laby[j], pch = 16, col = hdColors[pen[j]], cex = dcex)
    points(.1, laby[j], pch = 1, col = "black", cex = dcex)
    text(.18, laby[j] + nameShift, gnams[j], cex = cex, adj = 0,
         col = "black", font = font)
  }
}
```

#9 Plot rates with confidence bounds_____

```
countRange1 = range(fvsmk$Fruit_Vegs)
countRange1 = mean(countRange1) + 1.10 * diff(countRange1) * c(-.5, .5)
countGrid1 = panelInbounds(countRange1) # used pretty values that are in bounds
```

```

panelSelect(panels, 1, 3)
panelScale()
mtext(side = 3, line = line1, '(at least 5 times a day)', cex = cex)
mtext(side = 3, line = line2, 'Fruit and Veggies Consumption', cex = cex)

for (i in 1:nGroups) {
  gsubs = iBegin[i]:iEnd[i]
  nsubs = length(gsubs)
  pen = 1:nsubs
  laby = nsubs:1
  panelSelect(panels, i, 3)
  panelScale(countRange1, c(1-ypad, nsubs + ypad))
  panelFill(col = wgray)
  panelGrid(x = countGrid1, col = "white", lwd = 1)
  panelGrid(x = 5.6, col = "black", lty = 2) # hard code U.S. average
  panelOutline(col = "white")

  if (i == nGroups) {
    axis(side = 1, at = countGrid1, labels = as.character(countGrid1),
          col = "black", mgp = c(1, 0, 0), tck = -.04, cex.axis = cex)
    mtext(side = 1, line = .7, "Percent", cex = cex)
  }

  if (i == 6)
    pen = 6

  lines(fvsmk$Fruit_Vegs[gsubs], laby, col = "black", lwd = 1)

  for (j in 1:length(pen)) {
    points(fvsmk$Fruit_Vegs[gsubs[j]], laby[j], pch = 16,
           cex = dcex, col = hdColors[pen[j]])
    points(fvsmk$Fruit_Vegs[gsubs[j]], laby[j], pch = 1,
           cex = dcex, col = "black")
  }
}

```

#10. Plot counts-----

```

countRange = range(fvsmk$Smoke)
countRange = mean(countRange) + 1.10 * diff(countRange) * c(-.5, .5)
countGrid = panelInbounds(countRange) # used pretty values that are in bounds

panelSelect(panels, 1, 4)
panelScale()
mtext(side = 3, line = line1, 'Smoking', cex = cex)
mtext(side = 3, line = line2, 'Daily', cex = cex)

for (i in 1:nGroups) {
  gsubs = iBegin[i]:iEnd[i]
  nsubs = length(gsubs)
  pen = 1:nsubs
  laby = nsubs:1
  panelSelect(panels, i, 4)
  panelScale(countRange, c(1-ypad, nsubs + ypad))
  panelFill(col = wgray)
  panelGrid(x = countGrid, col = "white")
  panelOutline(col = "white")

  if (i == nGroups) {
    axis(side = 1, at = countGrid, labels = as.character(countGrid),
          col = "black", mgp = c(1, 0, 0), tck = -.04, cex.axis = cex)
    mtext(side = 1, line = .7, "Percent", cex = cex)
  }
}

```



```

    }

    if(i == 6)
      pen = 6

    lines(fvsmk$Smoke[gsubs], laby, col = "black", lwd = 1)

    for (j in 1:length(pen)) {
      points(fvsmk$Smoke[gsubs[j]], laby[j], pch = 16,
             cex = dcex, col = hdColors[pen[j]])
      points(fvsmk$Smoke[gsubs[j]], laby[j], pch = 1,
             cex = dcex, col = "black")
    }
  }
}

#11 Add Title and Legend -----

panelSelect(panels, margin = 'top')
panelScale()
text(.5, .85, 'Fruit and Vegetable Consumption and Smoking Statistics By State',
     cex = 1)

panelSelect(panels, margin = 'bottom')
panelScale(inches = TRUE)
xs = 2.8
ys = -.42
# text(.01 + xs, .70 + ys, "U.S. Average", cex = cex, adj = 0, font = 1)
# lines(c(.78, 1.50) + xs, c(.70, .70) + ys, lty = 2, col = "black")
# text(.01 + xs, .53 + ys, "90% Confidence Interval", cex = cex, adj = 0,
#      font = 1)
# lines(c(1.35, 1.61) + xs, c(.53, .53) + ys, lwd = 2, col = hdColors[1])

#12 Outline groups of Panels-----

for (i in 1:3) {
  for (j in 2:3) {
    panelSelect(panelBlock, i, j)
    panelScale()
    panelOutline(col = "black")
  }
}

panelSelect(panelBlock, 2, 1)
panelScale()
panelOutline(col = "black")

```