BDAgram: Exploratory Data Analysis

TO: Hyerim. Kim
cc: IRGN 452 (Big Data Analytics) students
Date: March 2, 2016
From: Prof. Roger Bohn

Exploratory data analysis (EDA) is the concept of open-ended examination of data sets to detect "interesting" patterns. It should be done early in any project, since it is very useful for generating ideas that you can later measure and test more formally. It is a creative process, not an analytic method. It also tends to rely on visual displays (plots and others), since the human brain is very good at spotting patterns visually. Good software makes EDA much easier and faster. With good software it is possible to go through 5 variations of an idea in 5 minutes.

We have not covered this topic systematically in BDA, but we did spend a class on graphical displays of data. The main reference was:

- Chapter 5, "Exploring Data," in *Data Mining for Rattle and R* (DMRR) by Graham Williams, Springer
- At the start of BDA I recommended a tutorial on EDA by Facebook. It's on Udacity: https://www.udacity.com/courses/ud651. It takes 3 to 4 hours, but by now much of it should be familiar so it will take less.
- Another tutorial by well-regarded faculty is from Coursera. Personally, I did not think it was as good. https://www.coursera.org/learn/exploratory-data-analysis
- Googling will show dozens of entries under: *tutorial "exploratory data analysis" R*

A few useful principles for EDA

- Since you are looking for patterns, look at multiple variable simultaneously, using techniques like faceted plots, mosaic plots, color, bubble charts, etc. All of these are covered in Chapter 5 of DMRR, although of course more detailed discussions are available.
- Scatterplots are probably the single most useful tool. There are many tutorials on scatterplots in R.
- Colorful maps are good for *presenting* data, but not for *exploring* it. I would

throw out most of the maps in your presentation. Instead, present the same information in a bar graph or (better) a multiple variable plot of some kind e.g. stacked bar graph, side-by-side bar, scatter, etc.

- When showing a variable for multiple observations (such as diabetes incidence by region of the country or by age), *put the observations in a logical order,* not the (essentially random) order from the original data set. For example, order them from largest to smallest values.
- When showing a variable for multiple observations, use a *consistent* ordering every time, so your eye can compare one plot to another. For example, if you are showing regions of the country in a particular sequence, do all your displays using the same sequence. This partially contradicts the previous suggestion, so you have a choice of which one to follow.
- Line up like-with-like. For example in the tables showing incidence of a condition, you had all the females underneath all the males. It's much more useful to line up both males and females next to each other so that you can quickly notice discrepancies/similarities.
- Use appropriate methods to *reveal* rather than *hide*. Try to design plots that will 1) compare similar things, 2) in ways that will highlight differences between them.
- Go back through your lectures notes and use all the comments and tips I've given about good and bad ways to work with data. For example:
  - When data varies over a wide range of values (10:1, or even 3:1 between biggest and smallest), a linear scale will almost always hide important patterns. (Most of the points will be concentrated in a small area at the left side.) Using logs is the easiest transformation to deal with this, but sometimes others will be even better. (e.g. square root)
  - Overplotting. Don't do it! See previous notes on how to deal with it.
  - Whenever showing a list (of variables, of observations, or anything), think about the *order* in which you show them. Usually, the default order they arrive in is random. Examples of useful orders are chronological, small to large, large to small, sorted by value of a different key variable such as an outcome variable, sorted by a hypothesized causal variable.
- You will usually be showing a 2 or 3 dimensional slice of mute-

dimensional information. Don't just settle for one way of slicing it; look at multiple ways to see if any of them shows patterns. For example in the basic incidence data, we may want to show:

  - By region
  - By age group (Age is always important with health-related data. The others are less certain.)
  - By male/female
  - By cost per claim
  - By cost per year (perhaps shown as a categorical variable)
  - By type of treatment
  - By specific sub-disease
  - Using e.g. a bubble chart, you can show the value of any third variable by any two of these variables.
  - Using colors adds a fourth variable, and using symbol shapes can add a fifth, although at that point a display gets very cluttered. It's usually better to show a faceted plot. A grid of scatter plots, for example, can show 4 dimensions).
- This list is just a start.

As always, it's up to you to reformat and combine raw data. In your case, the logical units of analysis will include claims, but more often probably patients, patient/year, etc. Save yourself time in the long run by constructing data sets that are organized by patient-year (one record per patient year) and patient (all claims for a single patient).

Even when you are reporting data at the claim level of detail, not all claims are equal! For example if you use dollar (won) weights, it will usually be much more informative than just counting how many claims are of particular types.

==========================

R tip: Relabeling and re-ordering categorical variables.
R dataframes are designed to work well with categorical data. You can change the names to make them clearer. For example instead of 1/2, use M/F or Male/Female. Instead of age groups 1, 2, 3…, use "0-5", "6-10", etc.

R refers to such categories as *factors*. In fact much of the time R works with categorical data (and dummy variables), the data is actually stored as factors. Factors can be either ordered (ordinal data), or unordered.

> In R, each possible value of a categorical variable is called a *level*. A vector of levels is called a *factor*. Factors fit very cleanly into the vector orientation of R, and they are used in powerful ways for processing data and building statistical models. …In many situations it is not necessary to call *factor* explicitly. When an R function requires a factor, it usually converts your data to a factor automatically. The *table* function, for instance, works only on factors, so it routinely converts its inputs to factors without asking. You must explicitly create a factor variable when you want to specify the full set of levels or when you want to control the ordering of levels. [From: *The R Cookbook*, by Paul Teeter. This excerpt available on Google books.]

So, learning how to set up factors can save a lot of time for plotting and other purposes. A good tutorial introduction is http://www.r-bloggers.com/data-types-part-3-factors/

Shrinking your data set:
This is not an EDA issue, and it applies to most of the BDA projects. If you are running into problems with slow computer speeds, or running out of memory, build a "shrunk" data set that has only the observations and variables you may want to analyze. (But err on the side of too much, rather than too little. It's always easier to throw out material that you don't want, than to re-build and insert data that you suddenly discover might be relevant.

In the start of the course, I suggested building a data set of 10,000 to 100,000 observations, just to make it easier to work with. We are now past that point, however. You should set aside a random subset of data for testing at the end, but other than that, use most or all of your "interesting" data.

For the Korean health analysis, go through the data set and throw out claims for patients who have zero diabetes-related activity. This should reduce the number of people by a factor of 10 or 20. Just be careful: if a person has a single diabetes claim, then you want to keep all claims for that year. Another caution is that there may still be purposes for which you want the non-diabetes patients. For example, showing diabetes as a percent of total $ expenditures. However for that purpose you don't need individual claims.

Pseudo code to collect all claims from people who have diabetes.

```
• Identify all diabetes related claims.
    ▪ Z1 <-  Dataset$Diagnosticgroup == diabetes
• Collect all the personal IDs associated with those claims.
    ▪ Z2 <- Dataset$ID[Z1]
• Reduce this to a list of unique IDs
    ▪ Z3 <- unique(Z2)
• Identify all claims for those individuals
    ▪ Z4 <- Dataset$ID %in% Z3  #Membership test
• Select all those claims
    ▪ Z5 <- subset(Dataset, Z4)  #Many other ways to subset in R
```

Other references:

- A thorough reference book is *Handbook of Data Visualization*, edited by Chun-houh Chen, Wolfgang Härdle, Antony Unwin, Springer.
- I'm not familiar with this one, but it's also available through UCSD and Springerlink. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach* by Natalia Andrienko, Gennady Andrienko ISBN: 978-3-540-25994-7 (Print) 978-3-540-31190-4 (Online)