CSSS 569 · Visualizing Data and Models

Problem Set 3

Professor: Chris Adolph, Political Science and CSSS

Winter Quarter 2020

Due in class on Thursday, 12 March 2020

General instructions for homeworks: Homework can be handwritten or typed. For any exercises done with R or other statistical packages, you should attach all code you have written and all (interesting) output. Materials should be stapled together in order by problem. The most readable and elegant format for homework answers incorporates student comments, code, output, and graphics into a seamless narrative, as one would see in a textbook.

On this problem set, chose to complete *either* Problem 1 or Problem 2. Either way, this problem set will be easiest to do using R.

Problem 1: From EDA to Model

In Problem Set 2, I asked you to produce two or three graphics showing relationships among variables in your own data, or in a dataset created from `gapminder.org`. This week, we will reuse these data, but with a more model-based approach. *Important Note: this problem only really "works" if some of your covariates are continuous or might plausibly be treated as such (e.g., an ordinal scale which you are willing to tentatively treat as interval level data). If all of your covariates are already discrete, use a different dataset or tackle Problem 2, below, instead.*

Based on your knowledge, hunches, and last week's exploratory data analysis, choose a regression model to fit one of the variables in your dataset. The model should involve a single dependent variable and more than one covariate. You may use any co-

variates you wish and transform them in any way you see fit, but at least one covariate *must* be continuous, or at least potentially treated as such. You may use any tools you have gathered in your past methodological training, including linear regression, generalized linear models, maximum likelihood estimation, time series analysis, etc. Choose a model that you find interesting and that you feel reasonably confident in fitting. You will not be graded on how "advanced" that model is.

a. Write out the model formally *in mathematical notation* so that I know what you have done. (Note: R formulas are *not* mathematical notation. Do not use them to describe a model in a paper.)

b. Estimate the model and report a table of parameter estimates. Tell me what method of estimation you used.

c. Models impose assumptions to gain traction over data. As noted in class, sometimes the assumptions are close in spirit to the data and thus worth making to simplify the patterns in the data. In other cases, even slightly weakening the assumptions of a model strongly changes the results. We want to avoid conclusions that depend strongly on model assumptions.

To investigate the robustness of our results, let's relax the assumption that (some) of our continuous (or treated-as-interval) covariates have simple, interval effects on the response variable. Replace one or more of these covariates with "smooths" of the respective variables, and refit the model. You may do this using a Generalized Additive Model or by adding smoothers (like splines) to another model. (Try your best to approximate your initial model in these frameworks, within reason.)

d. Show graphically whether using a smoother "makes a difference" substantively; i.e., whether or not the response variable reacts to the smoothed variable in an approximately linear fashion.[1] Do you have reason to doubt the appropriateness or usefulness of your original model? Why or why not?

e. If you now doubt the original model's treatment of explanatory variable x_i, can you think of a simple transformation, polynomial, or recoding of x_i that approximates the smooth function produced by GAM or a smoothing spline? That is,

1 The easiest way to do this is using the `gam()` command in `R` package `mgcv`; see especially `plot.gam()` and `vis.gam()`.

can you think of a parametric equivalent to the smoother you fit? If so, (1) refit the model using this transformation or recoding, (2) show that the reformulated model fits better than the original model using the goodness of fit metric of your choice, and (3) show graphically the similarity of this new fit to the smoothed model.

If you do not doubt the original model, demonstrate graphically that the original model and the smoothed model make similar predictions.

Problem 2: Creating an Interactive Graphic

Create an interactive plot of social science data, deployable as a webpage which responds in some useful way to user input. "Useful" covers a lot of ground: users might be able to subset the data displayed, change the variables plotted, transform one or more of the variables, add fits or other summaries, and/or learn about user-selected cases.[2] You are not limited to these examples. Rather, you will be marked based on the extent to which your interactive display provides significant utility for data exploration not available in a similar static plot.

Data. You may use any dataset and any graphical elements you find interesting and helpful in your interactive display. If you do not have a dataset of your own, you may use the `gapminder.org` data from Problem Set 2, though you should be careful not to simply replicate interactives available on `gapminder.org`.

Tools. ***You must use a programming language to create your interactive display.*** Subject to this restriction, you may use whatever tools you like (such as `D3` or `processing`), but if your primary exposure to programming is through `R`, I strongly recommend using the `Shiny` package for `R` discussed in class.[3]

Submitting your interactive. Students have several options for submitting their interactive displays. All students should send a brief (maximum 3 paragraph) description of the

2 What might constitute an unhelpful interactive? Again, there is no simple definition, but there are examples, such as an interactive that does no more than provide a flat menu from which a series of premade displays are selected.

3 A tutorial for `Shiny` can be found at `http://shiny.rstudio.com/tutorial`. A good way to get started is to run on your own computer the tutorial examples, then progressively rewrite one or more of the examples to apply to your own data. However, you should not feel limited by the structure of the `Shiny` examples, which are intentionally limited and simple compared to the potential power of the package.

underlying data, the purpose of the interactive display, the goals achieved, and the methods used to achieve them.

Additionally, students should provide the interactive itself and all underlying code. In some cases, this can be done by placing the interactive online and providing the instructor a link to it as well as the code (note this requires server support, which can be pricy and complex). In most cases, it will be simpler to submit the interactive as a set of executable scripts. In particular, users of `Shiny` should send a zip file of the folder containing their `ui.R` and `server.R` files along with any required additional scripts or data. If in doubt, please make sure I know what software I need to run your interactive.

You must ensure your code will run on another computer. *If you have any* `setwd()` *commands in your code, this is unlikely to be the case. Check on a friend's computer* ***before*** *submitting your interactive!*

If your code doesn't work, but you have made substantial progress, send me what you have and add to your write-up a description of the remaining barriers to implementation. *Be sure to indicate in your write-up whether your code should run as submitted!*

Evaluation. Students choosing to tackle Problem 2 will be evaluated on their success in creating a usable interactive graphic as well as how much utility (or promise) that interactivity provides for understanding the data. Significant allowance will be made for students will to take the plunge into a new set of tools.