Bogdan Romocea
June 29, 2011

Quality Assurance (QA)

Being a data analyst and making mistakes go hand in hand, with the latter being essentially an unavoidable consequence of the former. This relationship can however be disrupted and maybe even severed if we are careful to play the role not just of producers, but also of *skeptical consumers* of all the output and insights that we produce. In other words, we can make any mistakes we wish - as long as we are the first to catch them, in a timely fashion (ideally, before anybody else gets a chance to notice anything).

Setting up a good corrective feedback loop is one of the most important things that we can do to enhance our skills and effectiveness as data analysts:

1. A careful examination of the results - ranging from a quick read & compare in some cases, to automated testing and validations in other cases - allows us to catch mistakes.
2. Finding mistakes allows us to learn and enhance our controls and processes.
3. Better controls and improved processes will naturally lead to fewer mistakes in the future.

But this feedback loop cannot start and work its magic unless we are *proactive* about catching and learning from our errors.

QA is very important, given that mistakes can have a significant negative impact on:

- Projects, business initiatives and decisions;
- Others' perceptions about our abilities, competence and reliability;
- Other folks' willingness to work with us.

QA is very cheap, compared to the potential consequences above

- Obtaining correct results is always the first and foremost concern.
- Doing things quickly is meaningless, and of zero – or even negative – value, if subsequently the results turn out to be incorrect.
- The time spent on various QA techniques is a very good investment.
- QA doesn't take much time once the initial QA process is set up.
- There is no contradiction between paying attention to QA and getting things done quickly. In fact, the opposite is true. Applying QA techniques from the start – such as exploring the data, checking the intermediate steps, and writing concise, modular, high-quality code – not only helps ensure that the results are correct, but also decreases (in some cases dramatically) the time needed to finish the job.
- Any kind of task, from the simplest to the most complicated, is equally worthy of careful and diligent attention. The impact that a mistake could have on the business may vary; however, the risk of losing credibility and the potential impact on us stay the same.

QA techniques

- Pay close attention to the Exploratory Data Analysis: missing values, duplicates / number of unique values, summary statistics, plots, comparisons between distributions, etc.
- Keep asking yourself whether the numbers make sense (in agreement with other sources, etc.); if not, chances are that something is wrong.

- Don’t invent explanations as a substitute for taking the time to understand the data and to verify the code. It’s always better to check and correct, rather than imagine reasons and speculate about consequences.
- If in doubt, ask as many questions as needed.
- Keep verifying the data, the code and the results all along, step by step – rather than just once at the end (or maybe even not at all!).
- Avoid manual steps that would be difficult or time consuming to run again, such as copying and pasting several pieces of output into Excel.
- Set up a separate, automated validation step for recurring processes that run daily, weekly or monthly, by saving the summary results to a master dataset. For example if a process usually returns 1,500 +/- 250 leads but the latest run produced 7,500 or 100, then email yourself an alert and pause the distribution of the report until you get a chance to troubleshoot and understand what's going on.
- Use good programming techniques that pay attention to the following dimensions: code formatting, comments, modularization, simplicity, error-checking, automation, efficiency and so on.
- Never copy and paste code from macros and functions which can, and are meant to, be used directly, without any modifications. Doing so totally defeats the purpose of modularizing code, and creates vulnerabilities that serve no purpose and are simply unacceptable.
- Be very careful with what you get when joining or merging tables by columns with duplicate values.
- Do not make assumptions without stating them clearly upfront, and without bothering to check whether they may hold true. Other people may have encountered the same issues in the past, and thus may be in a much better position to confirm or reject our assumptions.
- Pay close attention to the process. Obtaining (approximately) correct results is no excuse for employing sloppy, deficient procedures that happen to have little or no impact this time. Next time, the same sloppy process could easily cause big mistakes.