As I shift from a teaching-and-consulting career to a research-and-consulting career, a few questions pop in my mind: "What general lessons did I learn in the past few years? What kind of general heuristics I should use in upcoming projects?". Working on consulting projects in the past few years.

1 - Do not trust the data.

When it comes to data, never assume that you will receive the data, spend a few minutes integrating it into the model you are building and you will be done. It is silly to think that data assembled by people who don’t know your model will be perfect for the model. Yet, that’s exactly what I expected in several occasions. Also, most of the project schedules I´ve seen implicitly assume that. If you are working with data that will be fed into a model, the first few minutes you spend looking at the data should be devoted for a dimension and reality check.

2 - Know where you stand in the agility vs. “waterfall” project management continuum, and adjust your behavior accordingly.

Everyone wants to be “agile”, but not everyone understand what it means. Also, not everyone understands that being agile may involve tradeoffs (you can find some). Sometimes, being agile may involve saying “we are OK if we trash this beautiful Gantt chart we thought would be great pursuing, because someone had an incredible idea or because doing what I thought initially turns out to be unfeasible. We are also Ok with dumping 25% of our work in the trash because we are embracing change in scope”. If you can’t imagine your \*\*client\*\* saying these words, you’re probably closer a waterfall project, and you should adjust your behavior accordingly.

“Know if you are playing in an Agile Team or Not.”

3 – “Over-generalization” may be as bad as “Under-Generalization”

You heard this before: “Don’t repeat yourself”. In programming, this means that you should seek to code at a level in which you will not need to repeat a set of instructions, so you need to generalize. In some circumstances, generalizing means that instead of creating predictive models, you may want to program a generic model builder, and store the models in a list. Extreme programming calls this principle YAGNI ([You Aren’t Gonna Need It](<http://wiki.c2.com/?YouArentGonnaNeedIt)>). The principle states:

"Always implement things when you actually need them, never when you just foresee that you need them."

2- Do not "over-generalize." your code.

The project scope is to build a simulation model of a manufacturing plant, not a flexible simulation environment suitable for all manufacturing plants. Of course, no expe following good practice (e.g. building an R package to

The following items are not meant to be generalized to all circumstances, but are a subset of practical experience over the