Estimagic

Janoś Gabler

September 12, 2019

Introduction

Structure of the Next Two Talks

- This talk:
  - What is estimagic
  - Using estimagic in a simple example
- Next talk:
  - What is respy
  - How respy has become more flexible recently
  - How estimagic helped to achieve that

For Those Who Saw the Last Presentation

- Less focus on design and implementation
- More focus on using estimagic
- Reflect questions we got from users

What is Estimagic

- **Estimagic** is a tool for estimation and inference of (structural) econometric models
- Model ≡ Criterion function for extremum-estimatior
- It's a meta package!
- Todays focus:
  - The maximize and minimize function
  - The params DataFrame
  - Convenience functions for parameter handling
  - Specification of constraints

Traditional Optimizers

- Take an array of start parameters
  - Encourage position based parameter handling
- Run for hours or days without feedback
  - User writes/prints logs for monitoring
- Lose all information when they crash
  - User saves parameters after every step
- Don’t support constraints typical in economics
  - User implements them by reparametrizations
- Only do minimization
  - User flips sign manually

Tutorial: Multinomial Probit

Goal

```
def mprobit(formula, data, algorithm='nlopt_bobyqa', dashboard=True):
    """Estimate ordered logit model described by *formula* on *data*.

    Args:
        forumla (str): A valid patsy formula
        data (DataFrame)
        algorithm (str): The optimization algorithm
        dashboard (bool)

    Returns:
        params (DataFrame): Estimated parameters

    """
    pass

# usage
formula = "insure ~ age + male + nonwhite + site_2 + site_3"
data = pd.read_stata('sysdsn1.dta')
mprobit(formula, data)
```

Tasks

- Process the user input
  - Process data using the formula
  - Set up start parameters
- Likelihood function
  - Parse parameters into quantities we need
  - Evaluate likelihood contributions
- Maximization
  - Set-up constraints
  - Do maximization
- **Same steps occur in any structural model**

Example Model

- Taken from Stata documentation
- Outcome: Type of insurance (Indemnity, Prepaid or Uninsured)
- Variables
  - age, male and nonwhite dummies
  - site (takes values 1, 2, 3)
- 3 Outcomes
  - 2 Identified beta vectors
  - Identified 2 x 2 covariance matrix with one normalized variance

look at mprobit.ipynb – Processing

Notes on Params DataFrame

- One object that bundles:
  - Parameter values
  - Information how to parse parameters
  - Lower and upper bounds
  - Information for visualization
- User is free to choose any index
- Should it be exposed to / written by the user?
  - Multinomial probit: No!
  - Skillmodels: partially
  - Respy: Yes!

What's A Good Index for Params

- How many levels?
  - Capture all relevant partitions
  - Have most frequent partitions as first level
  - Two to four levels work for most models
- Name your levels, so you can use them in queries
- Optimize mainly for ease of selection
  - Name column takes care of readability
- Don't mix int and string in one index level

look at mprobit.ipynb – Likelihood

Working With Parameters in Likelihood

- Use estimagic's utilities for covariance matrices
- Don't write too specific code
  - Never hard-code length of anything
  - (Almost) never extract parameters into scalars
- Bundle all parameter parsing in one place

look at mprobit.ipynb – Optimization

Specifying Constraints

- Constraint = dictionary
- Selecting parameters:
  - 'loc' to select based on index
  - 'query' for complex selections based on index and columns. If you need it a lot, choose a better index!
- Specify type of constraint:
  - covariance, sdcorr, sum, probability, increasing, equality, pairwise equality, fixed
- More information in the ▸ Documentation

maximize and minimize

- Both function have Identical Interface
  - criterion: function
  - params: DataFrame
  - algorithm: string
  - criterion_args (list) and criterion_kwargs (dict)
  - constraints: list
  - algo_options: dict
  - dashboard: bool
  - db_options: dict
- If you call maximize or minimize for the user, let him pass arguments through to estimagic!

What Algorithms Are Implemented

- Scipy Algorithms:
  - L-BFGS-B, TNC, SLSQP
- NLOPT Algorithms:
  - cobyla, bobyqa, newuoa, newuoa_bound, praxis, neldermead, sbplx, mma, ccsaq, slsqp, lbfgs, tnewton_precond_restart, tnewton_precond, tnewton_restart, tnewton, var2, var1, auglag, auglag_eq
- Pygmo Algorithms:
  - de, sade, de1220, ihs, pso, pso_gen, sea, sga, simulated_annealing, bee_colony, cmaes, xnes, nsga2, moead
- If you need another algorithm, let us know!

What Happened Since Last Time

- Numerical derivatives + MSM standard errors (Max)
- Likelihood Standard Errors (Linda)
- Improved dashboard + comparison plot (Klara)
- More options for optimizers, remote dashboard (Tobias)
- More constraints, more utilities, robust_cholesky (Janos)
- Many code reviews (Hans-Martin, Tobias, Klara, Janos)
- Almost: pounders algorithm (Moritz)
- Almost: persistent logging (Radost)