Computational Statistics with Python

Topic 1: Introduction to Computational Statistics

Expected lecture time: 1 hour

Giancarlo Manzi

Department of Economics, Management and Quantitative Methods

University of Milan, Milan, Italy

Course level and shape

- Course on **statistical simulation** when theory might fail.
- Special focus on **Monte Carlo methods** for random value and variable generation.
- Very few rigorous proofs and mathematical details, a lot of intuition and **see-how it-happens applications**.
- Special focus on **Python**.
- This is a huge and continuously developing field; therefore not enough time to see everything; need to **choose selected topics**.
- Basic **statistical knowledge** required.

Course Goals

- **Goal 1**: To introduce the Python language and the Anaconda platform (mainly Jupyter).
- **Goal 2**: To present a few Monte-Carlo methods through Python, namely:
  - Resampling methods.
  - Random numbers and random variable generation.
  - Basic MCMC methods.
  - Other MC methods.

Details

- **Basic resampling** (mainly bootstrap, but also jackknife, modified bootstrap)
- **Pseudo-Random number generators** (Examples: linear congruential generators, multiply with carry generators, lagged Fibonacci generators, etc.).
- **Random variable generation** (Examples: the inverse transform method; the accept-reject method; the Box-Muller method).
- **Monte Carlo numerical methods** (the Newton-Raphson algorithm, Monte Carlo integration, the EM algorithm, importance sampling, etc.).
- Review of **basic Bayesian analysis**.
- **Mixture models** (just an outline).
- **Monte Carlo Markov Chain basics**: the **Metropolis-Hastings algorithm**.
- **MCMC basics**: the **Gibbs sampler**.
- **Hierarchical Bayesian models**.

Textbooks and other material

- Efron, B., Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. New York: CRC Press.
- Kenett, R., Zacks, S., Gedeck, P. (2022). Modern Statistics: A Computer Based Approach with Python. *Forthcoming*.
- Guttag, J. V. (2020). Introduction to Computation and Programming Using Python.
- Lunn, D., Jackson, C., Best, N., Thomas, A., Spiegelhalter, D. (2012). *The BUGS Book*. New York: CRC Press.
- Martin, O. A., Kumar, R., Lao, J. (2022). *Bayesian Modeling and Computation in Python*. CRC Press.
- Notebooks, R scripts, other material, etc.

What is computational statistics?

- Definition 1 (Wegman, 1988): *CS is a collection of techniques that have a strong focus on the exploitation of computing in the creation of new statistical methodology*.
- Definition 2 (Efron and Tibshirani, 1991): *CS is a collection of computer-intensive statistical methods*.
- Definition 3 (Rao, 1993): *CS refers to the trend in modern statistics of basic methodology supported by the state-of-the-art computational and graphical facilities*.
- Definition 4 (Gentle, 2005): *CS is a class of statistical methods characterized by computational methods*.
- My personal definition - part I (also including simulation and Monte Carlo techniques): **CS is a set of tools to "survive" when traditional statistical theory doesn't help**.
- My personal definition - part II (also including simulation and Monte Carlo techniques): **CS is a set of tools to "solve" challenging theoretical problems**.

A computational statistics word cloud

statistics
computer
science
statistical
computational
methods
area
problems
computations
includes
application
section
mathematical
done
modern
computing
use
applied
variable
intractable
required
authors
obtaining
ever-growing
grounded
statistician
performance
related
might
bayesian
statisticians
mcmc
datasets
study
investigation
system
tools
paper
discuss
simulation
complicated
randomization
explore
bootstrap
entails
analysis
dealing
practical
development
computationally
one
feasible
power
perform
plea
translation
intensive
age
even
solutions
revolutionized
selection
impact
arise
carlo
medium
estimation
view
will
data
goal
areas
density
model
expanding
computers
intensity
experience
research
realize
important
size
notion
introducing
form
high
monte
advice
much
editor
genre
concerned
journal
numerical
detection
theory
completely
core
somewhat
classify
list
contributions
theories
tests
center
based
respectively
emphatic
reveal
inter-disciplinary
formulated
gives
quantitative
intrusion
connections
four
applications
connection
teaching
computation
quantitatively
rapidly
become
refined
represent
informatics
advances
network
developing
advance
discussed
previous

An interesting comparison

Feature	Traditional statistics	Computational statistics
Sample size	Small to moderate	Large to very large
Sample selection	Independent Identically Distributed	Nonhomogeneous
Number of random variables	One or a few	High number
Type of computation	Close to manual	Intensive
Mathematical burden	Theoretically tractable	Numerically tractable
Type of research question	Precise	Imprecise
Assumptions	Strong	Weak or nonexistent
Relationships	Generally linear or linearizable	Generally nonlinear
Population distribution	Known	Unknown
Errors	Normal	Distribution free
Type of inference	Mainly frequentist	Mainly Bayesian
Type of used algorithm	In closed forms	Mainly iterative
Statistical properties	Optimal	Robust

(Source: adapted from Wegman, 1988)

An example of problems with integration

- An integral like the following:

$$\int_0^1 \frac{1}{2\sqrt{2\pi}} x^4(1-x)^3 e^{-\frac{1}{8}(log(\frac{x}{1-x})-4)^2} dx$$

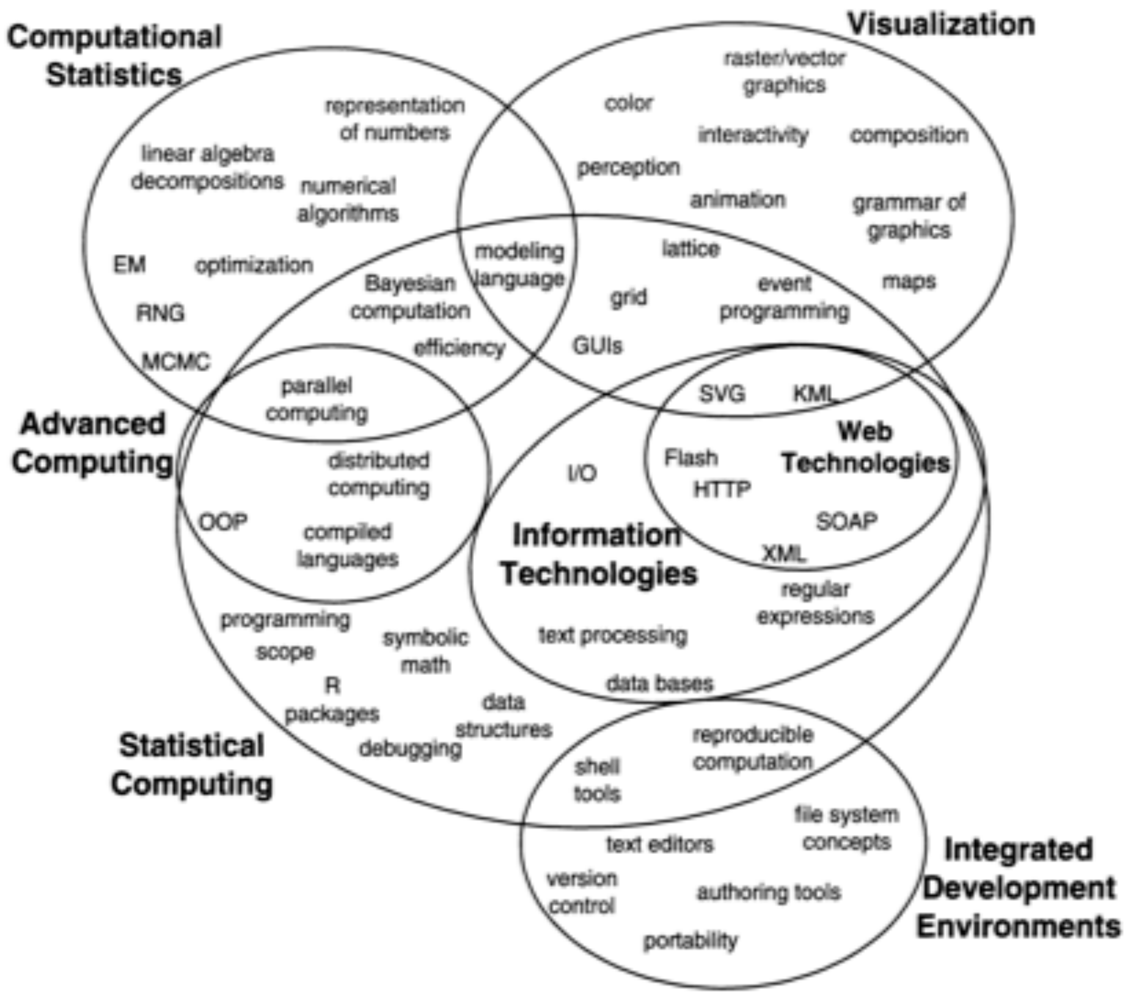
might be considered *analytically intractable*, meaning that either we cannot solve it or it has has no closed-form solution.

- This was the problem pioneer Bayesians had until the very end of last century because often the posterior densities they obtained from Bayesian analysis were intractable.
- Therefore we have to *approximate* it via simulation or numerical solutions.

Another example of an approximate solution

- You probably remember that in *R* there is the constant *pi* giving the value 3.141593.
- π is a real number and therefore *R* gives us an approximation of this real number with a lot of decimals (according to the floating point number memory).
- How is this approximation done?
- This is a numerical approximation exploiting geometry features and the uniform distribution.

A broad (and incomplete) vision of computational statistics and statistical computing



A few step- by-step example of an approx solution used in computational Statistics

- The maximum likelihood estimation (MLE) method is the most popular method to obtain estimators for parameters in statistics.
- It implies the computation of derivatives of sometimes too complicated functions to be solved analytically.
- Closed form expressions for the MLEs might generally not exist for most of the models, especially in the regression context.
- For example to find the estimators in the logit model one should use the MLE method.
- Sometimes to solve for MLE is difficult and one has to rely on numerical approximation.
- Then, some numerical approximation algorithms are used, such as the Newton-Raphson algorithm, the Fisher scoring method, the Iterative least square method.
- Let's see here how the Newton-Raphson works in general and with a binomial example.

The Newton-Rapshon algorithm

- The basic idea behind the NR algorithm is the following.
  - (i) Construct a quadratic approximation to the function of interest around some initial parameter value (hopefully close to the MLE).
  - (ii) Adjust the parameter value to that which maximizes the quadratic approximation.
  - This two steps are iterated until the parameter values stabilize.

Finding the maximum of a function....

Taylor series approximation.

- Let's start from the basic Taylor's theorem: *Suppose f is $k+1$ times differentiable on an interval I. For any points x and $x+h$ in I there exists a point w between x and $x+h$ s.t.:

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \cdots + \frac{1}{k!}f^{[k]}(x)h^k + \frac{1}{(k+1)!}f^{[k+1]}(w$$

- It can be shown that as h goes to 0, the higher order terms in equation 1 go to 0 much faster than h goes to 0. In this way we obtain (for small values of h) the *first order Taylor approximation of f at x*:

$$f(x+h) \approx f(x) + f'(x)h.$$

- Similarly, we obtain the *second order Taylor approximation of f at x* in this way:

$$f(x+h) \approx f(x) + f'(x)h + \frac{1}{2}f''(x)h^2.$$

Finding the maximum of a function.... (cont'd)

Taylor series approximation (cont'd).

- We can axpress the first order Taylor's approximation in terms of a linear function in h: :

$$f(x+h) \approx a + bh,$$

where $a = f(x)$ and $b = f'(x)$.

- Similarly, we can express the second order Taylor's approximation in terms of a polynomial in h:

$$f(x+h) \approx a + bh + \frac{1}{2}ch^2,$$

where $a = f(x)$, $b = f'(x)$ and $c = f''(x)$.

Finding the maximum of a function.... (cont'd)

Taylor series approximation (cont'd): finding a maximum of a second order polynomial.

- Suppose we want to find the value of x that maximizes:

$$f(x) = a + bx + cx^2.$$

- Let's compute the first derivative of f:

$$f'(x) = b + 2cx,$$

- Solving for $f'(\hat{x}) = 0$ we obtain the first condition for a maximum:

$$b + 2c\hat{x} = 0,$$

  that is, $\hat{x} = -\frac{b}{2c}$.

- Solving for $f''(x) < 0$ we obtain the second condition for a maximum:

$$2c < 0.$$

- This means that in $f(-\frac{b}{2c})$ we have a maximum if $c < 0$.

The NR algorithm to finding maxima of functions

- Suppose we want to find the value of x that maximizes some twice continuously differentiable function $f(x)$.
- Since

$$f(x+h) \approx a + bh + \frac{1}{2}ch^2,$$

with $a = f(x)$, $b = f'(x)$ and $c = f''(x)$, we have:

$$f'(x+h) \approx b + ch.$$

- The first order condition for the value of h (denoted $\hat{h}$) that maximizes $f(x+h)$ is

$$b + c\hat{h} = 0,$$

i.e. $\hat{h} = -\frac{b}{c}$.

- Therefore, the value that maximizes the second order Taylor approximation to f at x is:

$$x + \hat{h} = x - \frac{b}{c} = x - \frac{f'(x)}{f''(x)}.$$

and this is the basis for the NR algorithm.

The NR algorithm to finding maxima of functions (cont'd)

- With this in mind we can specify the Newton Raphson algorithm for 1 dimensional function optimization.

NR ALGORITHM: find the value $\hat{x}$ of x that maximizes $f(x)$

- [Input]: a function f, an initial value x_0, a tolerance level tol, an iterative value i set to 0.
- While $|f'(x_i)| > tol$ do:
  - $i = i + 1$;
  - $x_i = x_{i-1} - \frac{f'(x_{i-1})}{f''(x_{i-1})}$;
  - $\hat{x} = x_i$.
- Return ($\hat{x}$)

A BIG WARNING: the NR algorithm doesn't check the second order conditions necessary for $\hat{x}$ to be a maximizer. This means that if you give the algorithm a bad starting value for x_0 you end up with a min rather than a max!

The NR algorithm to finding maxima of functions: a Binomial example

- Recall the density of a binomial random variable y with parameter n and π:

$$f(y|n,\pi) = \binom{n}{y}\pi^y(1-\pi)^{n-y},$$

  for $y = 0, 1, 2, \ldots, n$.

- The log-likelihood function with respect to π (and without considering the binomial coefficient in which there is not a π term, so not contributing to the derivatives) is:

$$\ell(\pi|n,y) = y\ln(\pi) + (n-y)\ln(1-\pi).$$

- Let's compute the first derivative with respect to π:

$$\ell'(\pi|n,y) = \frac{y}{\pi} + (n-y)\frac{-1}{1-\pi}.$$

- Let's compute the second derivative with respect to π:

$$\ell''(\pi|n,y) = -\frac{y}{\pi^2} - \frac{n-y}{(1-\pi)^2}.$$

The NR algorithm to finding maxima of functions: a Binomial example (cont'd)

- Analytically, we know that the MLE is $\hat{\pi} = \frac{y}{n}$.
- Let's see if with the NR we obtain a good approximation in the case of $y = 2$ and $n = 5$ for which then $\hat{\pi} = 0.4$.
- We begin by setting a tolerance level.
- In this case, let′s set it to 0.01 (In practice you probably want something closer to 0.00001).
- Next we make an initial guess (denoted π_0) as to the MLE.
- Suppose $\pi_0 = 0.55$. Then $\ell'(\pi_0|y) \approx -3.03$ which is larger in absolute value than our tolerance of 0.01. Thus we set:

$$\pi_1 = \pi_0 - \frac{\ell'(\pi_0|t)}{\ell''(\pi_0|y)} \approx 0.40857.$$

The NR algorithm to finding maxima of functions: a Binomial example (cont'd)

- Now we calculate $\ell'(\pi_1|y) \approx -0.1774$ which is still larger in absolute value than our tolerance of 0.01. Thus we set:

$$\pi_2 = \pi_1 - \frac{\ell'(\pi_1|y)}{\ell''(\pi_1|y)} \approx 0.39994.$$

- Now $\ell'(\pi_2|y) \approx 0.0012$ which is smaller in absolute value than our tolerance of 0.01 so we can stop and we are very close to the analytical solution after only two iterations.

The Fisher scoring method

- In its simplest form (for example with one parameter only) the Fisher scoring method uses the expected Fisher information instead of the second derivative in the NR algorithm.

- So if we have a parameter π, for a given step in the algorithm:

$$\pi_i = \pi_{i-1} + \frac{\ell'(\pi_{i-1}|y)}{E\left[\ell''(\pi_{i-1}|y)\right]}$$

and the algorithm go ahead as the NR until the stopping rule (i.e. considering the tolerance value) is reached.

The Iterative Least Squares algorithm

- Apart from some details, the Fisher scoring in the context of model fitting, particularly in the regression context, is equivalent to the Iterative Least Squares algorithm (this was proved by McCullagh and Nelder (1989)).

In [2]:

```
#This is to let you have larger fonts...
from IPython.core.display import HTML
HTML("""
<style>x

div.cell { /* Tunes the space between cells */
margin-top:1em;
margin-bottom:1em;
}

div.text_cell_render h1 { /* Main titles bigger, centered */
font-size: 2.2em;
line-height:1.4em;
text-align:center;
}

div.text_cell_render h2 { /*  Parts names nearer from text */
margin-bottom: -0.4em;
}


div.text_cell_render { /* Customize text cells */
font-family: 'Times New Roman';
font-size:1.5em;
line-height:1.4em;
padding-left:3em;
padding-right:3em;
}
</style>
""")
```

Out[2]:

In [ ]:

```
```