STA 410/2102

Statistical Computation | Prof S Schwartz

For all course recources see the Course Site on Quercus https://q.utoronto.ca/courses/244990

UNIVERSITY OF TORONTO

Statistical Sciences

Synopsis

This course takes the perspective that, most fundamentally, *statistics* is **Integration** and **Differentiation** mathematics for a particular set of functions (e.g., distributions):

- Expectation (e.g., Probability) calculations are **Integration** problems
- Model Fitting (on data) is an **Optimization** problem, frequently addressed using **Differentiation**

With this in mind, the interest of the course is in leveraging numerical computations in pursuit of these fundamental calculations within the hierarchy of

- **Applied Statistics**: statistical analysis concerned with real world data and related issues
  - **Computational Statistics**: statistical calculations based on computational approaches (e.g., bootstrapping)
    - **Statistical Computation**: the careful and accurate implemention of computational statistical methods

which relies upon well known **mathematical** and **computational** foundations.

> **Regarding Mathematics versus Computing**, the author James E. Gentle who features heavily in this course is fond of saying:
>
> - "*The form of a mathematical expression and the way the expression should be evaluated in practice may be quite different.*"
> - "*Computer numbers are not the same as real numbers, and the arithmatic operations on computer numbers are not exactly the same as those of ordinary arithmetic.*"

For some additional introductory comments examining the domain of this course, the first two pages (vii-viii) of the the preface to Gentle's ***Statistical Computing*** *textbook are helpful. Further insightful clarification of the distinction between* ***Statistical Computing*** *and* ***Computational Statistics*** *is also available in the same preface in the sections titled* ***Statistical Computing Courses*** *and* ***Computational Statistics Courses*** *(pages xiii-xiv) and in the* ***Introduction to Part II*** *(pages 83-84).*

Course Objectives

1. **Programming Experience:** *develop comfort programming in new frameworks*.

   > "Programming is only learned by programming. (Read that again.)" -- James E. Gentle (**Computational Statistics**, page 134).

   Coding will be done in *Python* via the UofT JupyterHub (see the support pages for additional information) or google colab, and will utilize many modern computational frameworks, e.g., TensorFlow (TF) and TensorFlow Probability (TFP), and PyMC.

   - If you *can code* (e.g., in *R*) you will be given plenty of examples in this course which will allow you to effectively use *Python*.
   - If you *can't code* you are still welcome to take this course, but it will necessitate your learning to code quickly.

   Ability in *Python* (and coding more generally) is a skill that will serve you well in your career; and, *Python* is a very natural choice for an introductory course on computation.

   - While *Python* and *R* are both *high level programming languages*, *Python* is designed as a general purpose language for similar use cases (e.g., computational algorithm design) as *low level programming languages* (such as C, C++, and Fortran); whereas, *R* is designed to facilitate user guided data analysis rather than algorithmic development *per se*.
   - Because of its general applicability, *Python* has appealed to an extensive and diverse user base which has developed a prolific and synergistic computational ecosystem addressing an extremely rich set of computational domains (such as the aforementioned TF, TFP, and PyMC); so, competence with *Python* provides access to powerful programatic resources for essentially any problem or application outside the scope of *R*.
   - Beause of *Python's* ubiquity and usefulness, premier computational resources and capabilities (e.g., TensorFlow for automatic differentiation) have been made readily freely available for *Python* computation with any computer that can comfortably browse a webpage, e.g., on google colab (although, it must be noted that comparable support for R is also increasingly available).

   > It's not necessary, but if you wish to have a local *Python* installation on your computer, this can be easily done with conda (with mamba if you need it to be faster) and a code editor (such as PyCharm, Atom, pico/nano vi, emacs, etc.).

2. **Reading Experience:** *develop comfort reading advanced mathematical material*.

   Comfort (or ability to quickly develop comfort) in multivariate calculus and linear algebra *is required* for this course.

   > For the conscientious student:
   >
   > - Review material to get more than sufficiently up to speed for this course is available in the *Chapter 1: Mathematical and Statistical Preliminaries* of James E. Gentle's **Computational Statistics** Textbook (pages 3-73).
   > - And I additionally quite recommend the gleefully brief introduction of *Chapter 1: Review* from G. H. Givens and J. A. Hoeting in their textbook, also titled **Computational Statistics** (Second Edition, pages 1-18).

   A. The course notes endeavor to be a sufficiently self-contained primer for all content covered in this class. *Students should increasingly develop capability to read and understand the course notes as the course progresses.*

   B. The resources used to create the course notes are extensively referenced. *Students should increasingly develop the ability to read and understand advanced textbook and miscellaneous references provided in this course.*

   > The two resouces most utilized to create the materials of this course are freely available online via the UofT Library
   >
   > - **Computational Statistics** by James E. Gentle (Springer)
   > - **Computational Statistics** (Second Edition) by G. H. Givens and J. A. Hoeting (Wiley)
   >
   > but many additional resources will also be frequently encountered (and referenced) throughout the course.

   Developing comfort reading and deriving understanding from advanced textbooks or other challenging written exposition is a transformative right of passage that represents a threshold of significant maturity and ability in your development as a statistician, etc.. Consulting the references provided in the course will be of value in assessing and supporting concept mastery, as well as calibrating feedback and growth opportunity regarding mathematical maturity.

3. **Course Topcs:** *gain expertise in the fundamental concepts underlying* ***Statistical Computation***.

   After covering the classical *numerical mathematics* topics of

   - Floating-point number pitfalls
   - Useful Linear Algebra
   - Function Approximation

   the focus of the course shifts to the modern developments in

   - Differentiation and how it is used for optimization
   - Integration in high (and low) dimensional spaces

   > The empahsis firstly on foundations and then on modern differentiation and integration topics is made **at the expense** of several topics typially found in a ***Statistical Computation*** course; notably, through the **exclusion** of
   >
   > - Select linear algebra topics, e.g., sparse matrices, fast hadamard transforms, randomized numerical linear algebra
   > - Sampling from specific statistical distributions and nonparametric inference with bootstrapping
   > - Combinatorial (discrete) optimzation, e.g., with simulated annealing
   > - Constrained optimization, e.g., Lagrange multipliers, and interior and exterior point algorithms (e.g., for quantile regression)
   > - Expectation/minorization-maximization, e.g., for censoring and mixture models
   >
   > which are topics conspicuously **not covered in this course**, but which the successful student of this course would certainly be capable (and encouraged) to pursue upon completion of the course.

   An enumeration of the topics covered in this course are given at increasing levels of detail in the Schedule and Course Topics sections below.

Schedule

The Schedule for each class section is as follows.

- In accordance with University Covid Policy and Subject to Change
  - All lectures through January 31 will be given online through Zoom meetings available fomr the Quercus Course Page.
  - In-person lectures are tentatively scheduled to resume after January 31.

> In person lectures will not be recorded. Online lectures will not be set to record automatically; however, if remembered (or reminded) the "Zoom record meeting" can be turned on during Zoom lectures and (assuming no techical errors are encountered) the recorded video and chat (including class participation) will be made available to students in the course for viewing throughout the term of the course.
>
> Course videos and materials belong to your instructor, the University, and/or other sources depending on the specific facts of each situation, and are protected by copyright. Do not download, copy, or share any course videos without the explicit permission of the instructor.

L0101 (MP 202)	*L5101/L2501* (BA 1130)	Expected Lecture Topics / Class Day Activity	Programming Portfolio (40%)
Tue 3:10-6:00 PM	**Wed 6:10-9:00 PM**		**due end of calendar day (EoD)**
Online Zoom/Quercus	Online Zoom/Quercus		
Jan 11	Jan 12	Syllabus, Floating-Point, Numberical Errors	
Jan 18	Jan 19	Numerical Limits, Careful Computation, Big-O	
Jan 25	Jan 26	Lin. Alg., Decomposition, Gaussian Elimination	Programming Assignment #0
In-person MP 202 resumes*	In-person BA 1130 resumes*		Jan 17-23 enrollments get an
Feb 1	Feb 2	Decomposition, Iterative Methods, Gradients	extended due date of Feb 1/2
Feb 8	Feb 9	Vector/Function Spaces, Fourier Series, FFT	Programming Assignment #1
Feb 15 (**Online**)	Feb 16 (**Online**)	Coding Challenge I (15%) ~2.5hr with 20min for breaks and submission by end of lecture period	
~Feb 22~	~Feb 23~	READING WEEK	
Mar 1 (**In-person**)	Mar 2 (**In-person**)	Midterm Examination (10%)	Programming Assignment #2
Mar 8	Mar 9	Polynomial Bases, Optimization, Root-Finding	
Mar 15	Mar 16	Fixed-Point, Newton-Related, & Other Methods	
Mar 22	Mar 23	Integral Approximation, Sampling/Estimation	Programming Assignment #3
Mar 29	Mar 30	Copulas, Transformation/Bijection, MCMC/HMC	
Apr 5 (**Online**)	Apr 6 (**Online**)	Coding Challenge II (15%) ~2.5hr with 20min for breaks and submission by end of lecture period	Programming Assignment #4
Apr 11-29 (**In-person**)	Apr 11-29 (**In-person**)	Final Examination (20%)	

Communication

Announcements

Course related updates will be made via announcements on the Quercus Course Page.

Asynchronous (written) Communication

Course Discussion Boards and sta410@utoronto.ca will be monitored Monday through Friday and have response timeframes of approximately 48 hours.

1. For all community relevant questions of general interest to the class, use the Course Discussion Boards on the Quercus Course Page.
2. For private communications, e.g., regarding accomodations, etc., use the course email sta410@utoronto.ca.

   Emails sent to any other address may or may not be forwarded to the course email sta410@utoronto.ca. When forwarded they will have the response timeframes of approximately 48 hours from the time of forwarding.

Synchronous (verbal) Communication

Professor office hours occur immediately after course lectures

- Tuesday 6:10-7PM
- Wednesday 9:10-10PM

TA zoom help sessions will additionally be intially available for the first three weeks of the term (with a possibility of continuation further into the term) and are tentatively scheduled for

- Friday 11AM-12PM
- Friday 2-3PM

Grading

Undergraduate and Graduate course grades follow the Standard University Scale.

	0 - 49%	**50 - 52%**	**53 - 56%**	**57 - 59%**	**60 - 62%**	**63 - 66%**	**67 - 69%**	**70 - 72%**	**73 - 76%**	**77 - 79%**	**80 - 84%**	**85 - 100%**
GPA	0.0	0.7	1.0	1.3	1.7	2.0	2.3	2.7	3.0	3.3	3.7	4.0
	Failing	Passing					Failing	Passing				
	STA410	STA410					STA2102	STA2102				

Course grading is based on the following factors.

Programming Portolio	Coding Challenges	Midterm Exam	Final Exam
40%	30%	10%	20%

Graduate (STA2102) students (only) have the option to select an alternative weighting scheme which reassigns 10% of each of the first two coding-related categories to each of the exams.

40% (Optionally Paired) *Programming Portfolio Assignments*

This computational course requires frequent programming. There will be a *Programming Portfolio Assignment* due approximately every two weeks, with a total of five (5) *Programming Portfolio Assignments* submitted over the course of the semester.

1. *Programming Portfolio Assignments* are due before the calendar due date indicated for the assignment expires, i.e., before the end of the day (EoD).
2. Grading will be automated with MarkUs based on predefined tests of code correctness and conceptual understanding.
   - Only code submissions made through MarkUs will be graded.
   - Only completed and apppropriately formatted submissions will be graded.
   - All instructions for each submission component (including point values and formatting requirements) will be clearly specified with each assignment.
3. ***Code work and submission must be your own individual or paired creation; but, sharing "small hints" with classmates is allowed.***
   - Students may work individually.
     - Students choosing to work individually must work in accordance with the University Student Academic Integrity values of "honesty, trust, fairness, respect, responsibility and courage."
   - Students may self-select pairs for each assignment.
     - Paired students work together and may share work without restriction within their pair; but, must otherwise work in accordance with the University Student Academic Integrity values noted above.
     - **Paired students must BOTH separately submit their (common) work**, including (agreeing) contribution of work statements for each problem.
       - ***Automatic grading with MarkUs will be done on a per student basis and will not take student pairing into account*** so while students may work in pairs they must still submit their assignment individually.
       - *Failure to submit assignments on behalf of both students in a pair may result in a score of 0 being assigned to the student in the pair who did not submit their work.*

   Please seek partners in person or on the course discussion board on Quercus. Groups of three (or more) are not allowed; however, students are welcome to amicably seek new partners for each new assignment.
4. Regrade requests will be available through MarkUs for up to seven (7) days upon recieving returned scores for the following circumstances:
   - Grade awards not matching automated testing results
   - Automated test wrongly failing for some students but not others
   - Automated test wrongly failing for nearly all students

   If a regarde request is made in which an automated test wrongly thought to be failing is actually working appropriately, the problem and all related questions will be regraded on the basis of the correctness of the submitted code:

   - correct code: full points will be awarded
   - incorrect code: all questions related to the regrade request (e.g., all questions involving automated tests of submitted code in question) will be scored as 0.

15%+15% Coding Challenges I and II

This computational course requires development of programming competence. The class time on two class days will be used for *Coding Challenges* simulating job interview exercises.

1. The *Coding Challenges* will both take place during the full class period on the sixth and twelfth class days of the term

   > i.e., on the last class day before reading week and on the last class of the term
2. *Coding Challenges* are available 10 minutes after the hour and conclude after approximately 2.5 hours of effort.
   - Submissions will be expected to be received during the final 20 minutes of the scheduled lecture period, and late submissions will not be accepted and will result in an assigned score of 0 points.
   - Two five-minute breaks may be taken during the *Coding Challenges*; but, students must be mindful to meet submission deadline at the end of the lecture period.
3. ***Students must be able to work from a computer with reliable internet connectivity during the normal three (3) hour class lecture period on course days with Coding Challenges***.
   - To avoid access and connectivity problems, students may complete the *Coding Challenges* in any location with reliable connectivity for online coding work, e.g., home, dorm rooms, library, etc.

     > The classroom will be available for seating (limited by power outlet accessibility) for students working on laptops.
   - A scheduling accomodation may be requested at sta410@utoronto.ca for students who with reasonable planning have attempted to identify a suitable workspace for the *Coding Challenge* but been unable to do so.

     > Accomodation requests must be made in advance of the *Coding Challenge*. Accomodation requests made on the day of the *Coding Challenge* will only be considered for unexpected and unforseen circumstances.
   - The professor will be accessible at sta410@utoronto.ca and in the classroom during the *Coding Challenges* should any unexpected technical difficulties arise; however, the students are expected to be comfortable with the workflow required for *Coding Challenges* due to its similarity with the *Programming Portfolio Assignments*.
4. **Receiving or sharing information about the *Coding Challenge* during the *Coding Challenge* is cheating.**
   - **Students may not receive or share information about the *Coding Challenges* as a result of physical proximity to any other individuals during the *Coding Challenges*.**
   - **Students are not allowed to partake in (verbal, written, electronic, signed, etc.) communication about the *Coding Challenges* with any individuals during the *Coding Challenge* period.**
5. The *Coding Challenges* will have a similar feel to the programming assignments above and grading will be automated with MarkUs in the same manner as the *Programming Portfolio Assignments* above.

10%+20% Midterm and Final Examinations.

Questions for the *Midterm and Final Examinations* are derived from the course notes and evaluate appropriate course progress and learning.

1. The *Midterm and Final Exams* will both be two (2) hour examinations.
2. The *Midterm Exam* will be an in-person examination administered in class (starting 10 minutes after the hour, as usual) during the seventh class day of the term.

   > I.e., immediately after reading week.
3. The *Final Exam* will be an in-person examination administered by the Faculty of Arts and Science.
   - The *Final Exam* will be similar in style to the *Midterm Exam*, and will included questions covering the whole (not just the second half) of the course.
4. Students may bring a single standard 216 x 279 mm (8.5 x 11.0 in) paper "notes cheat sheet" for both the *Midterm and Final Exams*, and perform calculations on scratch paper or a calculator.

- **Students will not sit to the left or the right of other students during examinations**.
- **Giving an answer to other students who then submit this answer as their own is cheating on both counts**.
  - **Students are not allowed to communicate with other students during the examination period**.
  - **No cell phones, computers, or any communication medium may be used during the examation period.**

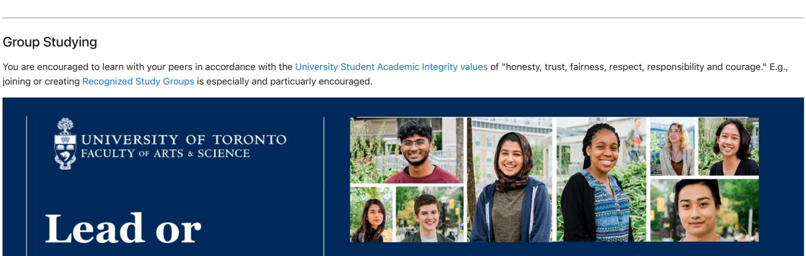
1. Regrade or grade check requests regarding the *Midterm Exam* must be made to sta410@utoronto.ca within seven (7) of receiving returned *Midterm Exam* scores.

   > Your request must clearly identify the question you have concerns about and contain a detailed justification for your concern with specific references to your answer and to the relevant course material. It is possible for your adjusted grade to be lowered if the request exposes related problematic aspects of your initial submission.

   Regrade requests regarding the *Final Exam* must be made following the guidelines of the Faculty of Arts and Science Registrars Office.

Group Studying

You are encouraged to learn with your peers in accordance with the University Student Academic Integrity values of "honesty, trust, fairness, respect, responsibility and courage." E.g., joining or creating Recognized Study Groups is especially and particuarly encouraged.



Missed Assignments

- ***Programming Portfolio Assignments*** may be completed over a two-week period and so extensions on the basis of short-term illness will not be granted; however
  - Late submissions will be accepted for up to three (3) days (at most) after a due date with a subtraction of 20% per day late, after which time the assignment will recieve a score of 0.

    > E.g., a score of 90% will recieve a 50% mark if turned in on the second calendar date after a due date.
  - Students who join the course late during the period of January 17-23 will have the due date of January 25/26 of Programming Portfolio Assignment 0 extended one week to February 1/2.
- Missed ***Midterm Exams*** and ***Coding Challenges*** will not be accomodated without submitting an Absence Declaration through ACORN and notifying STA410@utoronto.ca within seven (7) days of the missed assessment.
  - ***Midterm Exams*** (10% of course grade) will not be available for reschedule, and in the event of a missed ***Midterm Exam***, the ***Coding Challenges*** and ***Final Exam*** will respectively be upweighted to 35% (from 30%) and 25% (from 20%) of the course grade.
  - ***Coding Challenges*** must be rescheduled and completed within seven (7) days of their originally scheduled date; otherwise, missed ***Coding Challenges*** will recieve a score of 0.

    > ***Coding Challenge*** scheduling accomodations made in advance for students who have reasonably attempted to identify a suitable workspace but been unable to do so are detailed in the ***Coding Challenges*** Section above.
- Missed ***Final Exams*** require a petition to the Faculty of Arts and Sciences.

Further Accomodations

Specific exceptions to the schedule requirements of this course must be initiated through the registrar's office or Student Accessibility Services.

> E.g., if you are missing more than one week of class due to illness or emergency notify your registrar's office as soon as possible.

Course Topics

0. Computer Numbers

- ***Programming Homework Assignment 0***
- ***Core Textbook Readings and References***

1. Floating-Point Numbers $\mathbb{F}$
   A. Fixed-Point Numbers $\mathbb{I}$
   B. Pseudorandom Numbers
      - Modulus Recursion
      - Bit Manipulation
      - Period and Seed
      - Randomization Libraries and Python Modules
   C. Bit Arithmatic I: Multiplication (e.g., with `0` and `1`)
      - Numeric Underflow
   D. Bit Arithmatic II: Special "Numbers" $\pm\infty$ and NaN
      - Numeric Overflow
   E. Bit Arithmatic III: Breaking Addition
      - Roundoff Error
      - Catestrophic Cancellation
2. Density of $\mathbb{F}$
3. Modeling $\mathbb{R}$ with $\mathbb{F}$
   A. Limits of Precision: Root Finding with Bisection
   B. Selective Computation: The Quadratic Formula
   C. Avoiding Numerical Error Accumulation
      - Machine-Epsilon $\epsilon_{machine}$
      - Symplectic Integration
   D. Tricks: Logs and Taylor Series Expansions
      - Taylor Series Expansions
        a. Algorithmic Efficiency: The Time-Space Tradeoff
   - Big "O" Notation I
     - little "o" Notation
     - A Counting Example with Convolutions

1. Solving Equations I

- ***Programming Homework Assignment 1***
- ***Core Textbook Readings and References***

1. Linear Equations as Mathematics $Ax = b$
   A. Essential Linear Algebra Concepts
      - Matrix Inversion and Transposition
      - Linear Independence and Rank
      - Orthogonality and Normal Vectors
      - Singular Value Decomposition (SVD)
      - Multicollinearity
      - Eigendecomposition
      - Principal Components Analysis (PCA)
      - Eigenvalues and Eigenvectors
      - Vector and Matrix Norms
   B. Condition
      - Example Applications
      - Stability and Error Analysis
   C. Generalized Inverses
      - The Gramian Matrix
      - Sherman-Morrison-Woodbury Formula
2. Linear Equations as Computation $Ax = b$
   A. Gaussian Elimination
      - Elementary Operations
      - LU Decomposition
   B. Other Factorizations
      - "Big O" Notation II
      - QR Decomposition
        - Least Squares I: Minimizing Error $Ax \approx b$
      - Cholesky Decomposition and C^{-T}
3. Iterative Methods
   A. Gauss-Seidel
      - Successive Overrelaxation
   B. Least Squares II: Objective Functions $x^T Ax + x^t b$
      - Gradients
      - Automatic Differentiation
   C. Coordinate Descent
   D. Gradient (Steepest Direction) Descent
   E. Conjugate Gradient Methods

2. Representating Functions

- ***Programming Homework Assignment 2***
- ***Textbook Readings***

1. Function Spaces
   A. Spline Basis Functions
   B. Smoothing Matrices
   C. Backfitting Additive Regression Models
   D. Regularization
   E. Data Smoothing
   F. Generalizability and the Variance-Bias Tradeoff
2. Vector Spaces
   A. Inner (dot) Products
   B. Norms
   C. Metrics
   D. Vector Space Basis Sets
   E. Fourier Coefficients
   F. Convergence Terminology
3. Fourier Series and Transforms
   A. Truncated Fourier Series Basis Expansions
   B. Characteristic and Moment Generating Functions
   C. Convolution Theorem
   D. Discrete Fourier Transform (DFT)
   E. Transformation Applicability
   F. Primitive n^{th} Roots of Unity
   G. Fast Fourier Transform (FFT)
4. Polynomial Bases
   A. Taylor Series and Monomials
      - Taylor Series: Laplace Approximation
   B. Standard Bases
      - Legendre Polynomials P_k
      - Chebyshev Polynomials T_k
      - Jacobi Polynomials Q_k
      - Laguerre Polynomials $L_k^{(\alpha-1)}$
      - Chebyshev-Hermite Polynomials H_k^e
        - Hermite Polynomials H_k
        - Hermite Exponentially Tilted Polynomial Functions
   C. Comparing Bases: Lagendre versus Laguerre Polynomials for e^{-x}
   D. Computationally Efficiency in Standard Polynomial Bases
      - Christoffel-Carboux Formula for Evaluating $f(x)f(y)$
      - Three-Term Recurrence Relation for Evaluating $f(x)$
   E. Density Approximation with Chebyshev-Hermite Polynomials
      - Gram–Charlier and Edgeworth Series and Cornish-Fisher Expansion
5. Deep Neural Networks (DNNs)

3. Solving Equations II (Nonlinear Optimization/Parameter Estimation)

- ***Programming Assignments***
- ***Textbook Readings***

1. Nonlinearity $b = f(x) \neq Ax$
   A. Minima, Maxima, and the Hessian $H_{f(\theta)}$
      - Multivariate Taylor Series Approximations
   B. Maximum Likelihood Estimates (MLEs)
      - The Jacobian $J_{g(z)}$
   C. (Bayesian) Variational Inference
2. Nonlinear Root-Finding $0 = f(x) \neq Ax$
   A. Bisection
   B. Netwon's Method
      - Multivariate Taylor Series
   C. Fisher Scoring
      - Iteratively Reweighted Least Squares (IRLS)
   D. The Secant Method
   E. Fixed-Point Iteration
      - Contractive Mapping Theorem
      - Scaling
3. Newton-Like Methods
   A. Ensuring Monotonic Iteration
      - Modified Newton Methods
   B. Common Optimization Algorithm
   C. Quasi-Newton Methods
4. Other Methods
   A. Gauss-Newton
   B. Nonlinear Gauss-Seidel
   C. Acceleration
   D. Line Search
      - Golden Section Search
   E. Nelder-Mead
5. ~~Combinatorial (Discrete) Optimzation~~
   - ~~Simulated Annealing~~
6. ~~Constrained optimization~~
   - ~~Expectation-Maximization~~
   - ~~Interior and Exterior Point Algorithms~~

4. Numerical Integration

- ***Programming Assignments***
- ***Textbook Readings***

1. Approximation: Integration by Function Simplification
   A. Newton-Cotes Quadrature
      a. Riemann Integrals
      b. Trapezoidal Quadrature
      c. Simpson's Rule
   B. Improving Approximation: Romberg Acceleration
   C. Gaussian Quadrature
2. Estimation: Integration by Statistical Sampling
   A. Monte Carlo Integration
      a. Inverse CDF Sampling
      b. Rejection Sampling
      c. Importance Sampling
   B. Improving Estimation: Variance Reduction
      - Biased Sampling
      - Antithetical Sampling
      - Control Variates
      - Roa-Blackwellization
3. Advanced Sampling Techniques
   A. Copulas: Conditional Distributions via Differentiation
   B. Bijections: Transformation of Variables
      a. Normalizing Flows
      b. Masked Autoregressive Flows (MAFs)
      c. Masked Autoencoder Density Estimation (MADE)
      d. Inverse Autoregressive Flows (IAFs)
      e. Real NVP: Easy Jacobian with Affine Transformations
   C. Markov Chain Monte Carlo (MCMC)
      a. Gibbs Sampling
      b. Convergence: Markov Chain Mixing
      c. Metropolis-Hastings
      d. Detailed Balance
      e. Hamiltonian Monte Carlo (HMC)
      f. Diagnostics

1. ~~Bootstraping Methods~~