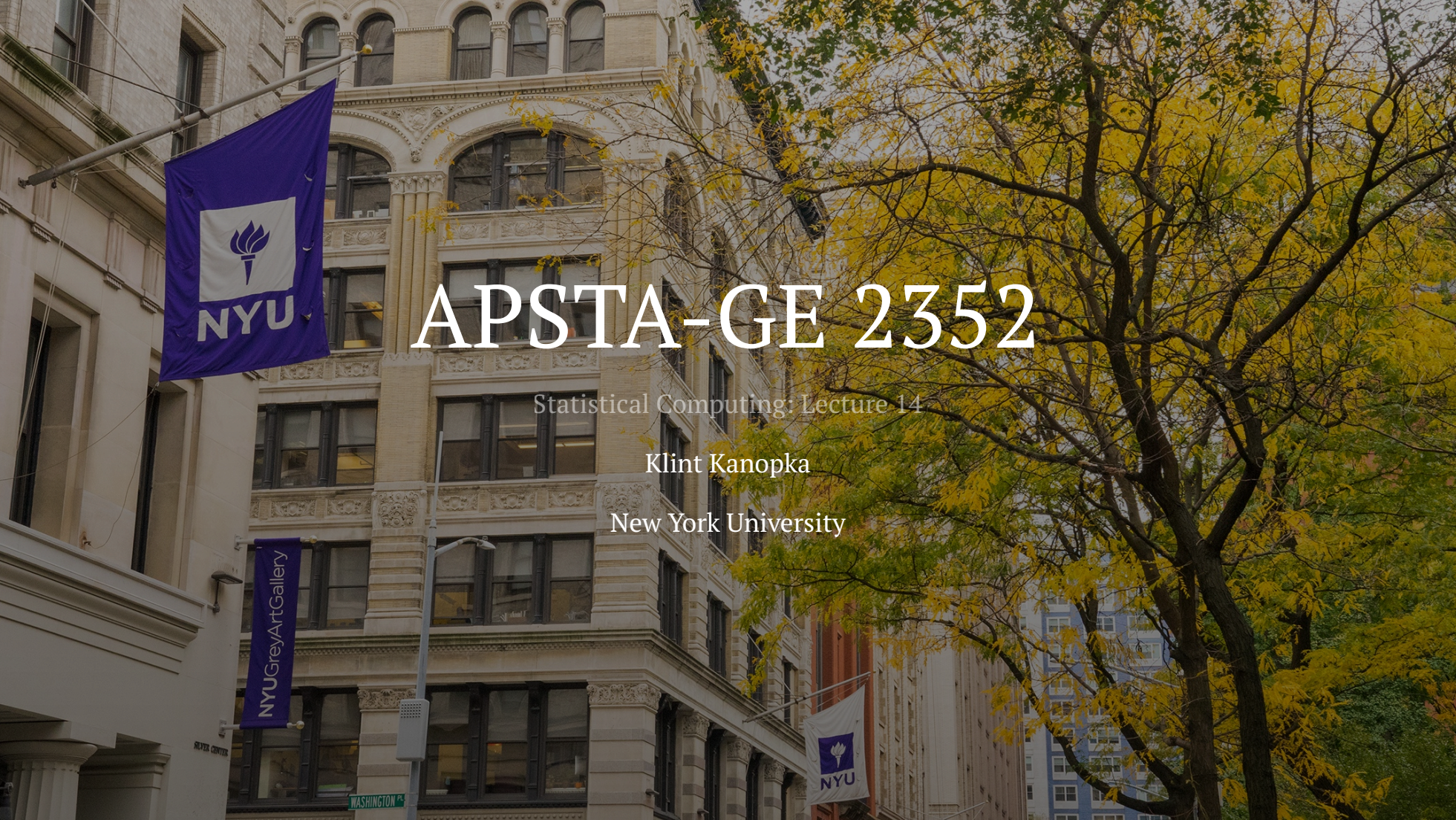

APSTA-GE 2352

Statistical Computing: Lecture 14

Klint Kanopka

New York University

Table of Contents

1. Statistical Computing - Week 14
    1. Table of Contents
    2. Announcements
2. Wrapping Up the Semester
    1. Revisiting Formal Course Goals
    2. The *Real* Course Goals
    3. What could have gone better?
    4. What went well?
    5. Course Evaluations
3. Final Review
    1. What to study
    2. Conceptual Questions
    3. Numerical Optimization
    4. Model Evaluation
4. Wrap Up

Announcements

- PS7 is due on Monday, December 15 @11.59p
  - **Must be submitted on Gradescope!**
- Final Exam is Thursday, December 18
  - In this room
  - Bring one page of notes (both sides of a single 8.5×11 sheet of paper)
- Content may vary from the practice final, but format will be very similar
- Errata points are reconciled for lectures, not yet for PS3-PS7

Wrapping Up the Semester

Revisiting Formal Course Goals

1. Students will implement literate programming to produce coherent and reproducible code.
2. Students will verify code function through the implementation of unit tests.
3. Students will write more efficient code by applying optimization techniques (e.g., vectorization, parallelization).
4. Students will solve problems by implementing and modifying algorithms.
5. Students will answer statistical questions by implementing Monte Carlo simulations.

The *Real* Course Goals

1. Be more comfortable with using `R` like a programming language

- Loops and control flow (`if`, `else`, `while`, `for`)
- Boolean logic, `TRUE` and `FALSE` tricks
- Going from a code sketch in words to functional code
- Fixing and diagnosing your errors

2. Start seeing the world as full of computable problems

- If you're clever about framing, everything is a search or optimization problem
- Most things can be solved with `optim()` or MCMC

The *Real* Course Goals (continued)

3. Discuss some things you don't necessarily learn elsewhere in the program

- Some of the nuts and bolts of numerical optimization
- Regularization, cross-validation
- Clustering and PCA

4. Help you understand that you don't have to be fully reliant on the software that's already been written!

- You can implement algorithms in `R`!

What could have gone better?

- I don’t feel great about the PCA and SVD lectures
  - Either requires more scaffolding or a total revamp/replacement
- Assignment length could be more consistent
- Lab attendance was poor
- Office hours weren’t efficiently utilized

What went well?

- You are all **much** stronger programmers than you were in September!
- Everyone did really well on assignments
- Lab quality improved over the course of the semester
  - Still room for improvement here! If you want to teach lab and help me improve this next year, let me know!
- Some activities, assignments, and lectures feel really good
- Slack continues to seem really helpful for course-related communication and community building

Course Evaluations

- The formal eval for NYU: https://coursefeedback.nyu.edu/nyu/
- The informal eval for us: PollEv.com/klintkanopka

Final Review

What to study

- Big ideas: `if`, `else`, `while`, `for`
- Indexing and how to do selection with indexing
- Recycling behavior
- Big ideas about:
  - Simulations
  - Optimization
  - Randomized algorithms

Conceptual Questions

1. Describe a task that you can easily parallelize. Describe a task that you cannot parallelize.
2. Given a dataframe, `d`, with columns `X1, X2, X3, X4`:
    1. Write code using a `for()` loop that computes the mean of each row and returns a vector of row means
    2. Write code using an `apply()` family function that computes the mean of each row and returns a vector of row means
    3. Write *vectorized* code to compute the mean of each row using only basic arithmetic operations and returns a vector of row means
3. For a dataframe, `d`, what is the difference between `d[4]` and `d[[4]]`?
4. What is the purpose of regularization? How are the LASSO and ridge regularization similar/different? In what cases might you use each?
5. In what situation is Markov Chain Monte Carlo most useful for numeric optimization?
6. How could you apply parallelization to k-Means clustering to improve runtime?

Numerical Optimization

- We will construct our own version of `optim()` using ternary search, designed to find the maximum value of a *convex* function of one variable, $f(x)$. Because we guarantee $f(x)$ is convex, it only has a single maximum (or minimum) value. Ternary search is useful for solving *bounded* optimization problems on convex functions. Bounded specifically means that you are looking for x^*, where $x^* = \arg\max_x f(x)$, subject to the constraint that $x_{left} \leq x^* \leq x_{right}$. Ternary search accomplishes this task by dividing up the search space into three pieces by selecting two boundary guesses, b_1, b_2. Because $f(x)$ is convex, there are exactly three things that can happen:

  1. $f(b_1) < f(b_2)$: This means the function is *increasing* on the interval $[b_1, b_2]$, so the maximum can't be on the interval $[x_{left}, b_1)$ and must be on the interval $[b_1, x_{right}]$
  2. $f(b_1) > f(b_2)$: This means the function is *decreasing* on the interval $[b_1, b_2]$, so the maximum can't be on the interval $(b_2, x_{right}]$ and must be on the interval $[x_{left}, b_2]$
  3. $f(b_1) = f(b_2)$: The function has to move from *increasing* to *decreasing* on the interval $[b_1, b_2]$, so the maximum must be on the interval $[b_1, b_2]$

Draw a diagram to prove these facts to yourself.

Numerical Optimization

- To carry out ternary search, follow the algorithm:

  1. Select $x_{left} < b_1 < b_2 < x_{right}$
  2. Compute $f(b_1), f(b_2)$
  3. Reassign x_{left} and/or x_{right} based on the criteria above
  4. If $\frac{x_{left}+x_{right}}{2} - x^* \leq \varepsilon$, return $\frac{x_{left}+x_{right}}{2}$, otherwise repeat

- Your task is to write a single function that carries out ternary search. Your function should take as arguments:

  - `f` : the function to minimize
  - `left` : the starting left bound
  - `right` : the starting right bound
  - `eps` : ε, the error tolerance

Numerical Optimization

- Questions:

1. How do you guarantee that $\left|x^* - \hat{x}^*\right| \leq \varepsilon$?
2. Does this algorithm appear efficient? Why or why not?
3. Describe the worst possible case for the run time of this algorithm in terms of k, the number of iterations until convergence. Write an expression that would allow you to solve for the worst case number of iterations required for convergence in terms of k, ε and the starting guesses of x_{left}, x_{right}

Model Evaluation

- One way that classification models can be evaluated is using the F_1 score, which is the harmonic mean of precision and recall, where precision is the proportion of predicted positive cases that were actually positive and recall is the proportion of actually positive cases that were predicted positive. The F_1 score is constructed as:

$$F_1 = \frac{2}{\text{precision}^{-1} + \text{recall}^{-1}}$$

- Write a function that takes in a vector of binary labels $(y_i \in \{0, 1\})$ and a vector of binary predictions $(\hat{y}_i \in \{0, 1\})$ and computes the F_1 score.

Model Evaluation

- Now assume you have a function, `FancyModel(train_data, test_data, lambda)` that takes in training data and some tuning parameter $\lambda \in [0, 1]$, fits a model, and outputs predicted labels applied to the test data. Don't worry about how it works, just know that if you want predictions for the data that created the model, you can supply the same argument for `train_data` and `test_data`.

- Write code to:

  1. Create an 80/20 train/test split of your original data, `d`. There is a column in `d` called `label` that contains the true label for each observation.
  2. Carry out ten-fold cross validation on the training set for ten potential values of λ, finding the F_1 score for each held out fold and λ combination
  3. Select the value of λ with the highest mean F_1 score across folds
  4. Generate predictions for the test set using your chosen value of λ
  5. Compute the out of sample F_1 score

Wrap Up

Recap

- Final exam
  - You can bring one 8.5×11 sheet of notes
  - The focus is less on memorizing specific algorithms and more on writing and understanding efficient and effective `R` code
  - Think about stuff we've hammered all semester: `if`, `else`, `while`, `for` are most of it

Final Final Thoughts

- PollEv.com/klintkanopka