Bike Share

PLEASE FIND YOUR NAME on the list of today's studio groups. Then find your partner(s), seat yourselves comfortably according to your group number, and read through the rest of this document.

This worksheet will cover the in-class work for the next *two* class periods, plus an optional mini project to complete outside of class. The worksheet is broken into several sections. If you finish a section, you are welcome to move on to the next one. If you need additional time, you are also welcome to work on it outside of class.

Preliminaries: Studio Groups

Whenever we are in studios, we will ask you to work in a small group (most likely a pair). Often this work will entail completing a worksheet like this one. Each member of the group should complete their own worksheet, but we encourage you to collaborate closely and discuss each section together. Free to also interact with the people at your table in any way that is helpful. You are welcome to get up and move around the studio, make use of the whiteboards / blackboards, etc.

How did we form the studio groups? Glad you asked! We used a very simple algorithm, namely uniform random shuffling. This algorithm makes no effort to create groups that are matched or balanced in any way. Its efficacy relies on two important facts: (1) the studio is filled with people who are both motivated to learn and capable of helping *you* learn, and (2) we will be changing groups often throughout the semester. Taken together, these facts imply that it doesn't matter very much who you work with, because you are likely to have a good experience no matter who you are grouped with — and even if you don't, the downside is likely to be outweighed by the good experiences you have in other groups. Furthermore, we believe that each of these experiences (good, bad, or just different) will help you learn about yourself and develop your skills at working with others.

Preliminaries: Pair Programming

The Jupyter notebook tasks associated with this worksheet should be done using pair programming. Pair programming involves two roles:

Driver: types and runs code.

Navigator: provides high-level guidance and helps debug.

While the driver is the one typing, the navigator(s) should be the one(s) leading how that code is structured and implemented. The other group member(s) will be navigator(s). On Friday, you should switch who is the driver. It may also be helpful to jointly sketch the code on this worksheet before actually implementing it. Remember that if you are an experienced programmer you can always further develop your skills by helping your fellow team member(s) learn.

Agenda and Checklist

Each day, we will start with a discussion about the modeling concepts in the worksheet and any questions that arose. The rest of the studio time you will be free to work through the worksheet, which includes going through parts of the notebooks associated with Chapters 2-4 in the book. We will also reconvene at the end of each class period to discuss and debrief.

Our main goals are:

- To introduce modeling concepts (states and state changes, model parameters, time steps, randomness) and tasks (running a simulation, storing the output as a time series, plotting results graphically).
- To introduce related programming concepts (functions and function parameters, conditionals, iteration) and objects in the ModSimPy library (State, TimeSeries).
- To practice working with Jupyter notebooks.

The following to-do's should be on your radar.

- [x] Write your name here: Lilo
- [x] Write your name(s) of your studio partner(s) here: Kei
- [x] By Sunday, September 8: Scan this worksheet and submit it on Canvas. (Make sure that you scan all pages of this document, and that each page is legible and oriented correctly.)
- [x] By Monday, September 9: Have a NINJA check-in. scheduled for tuesday
- [ ] *Optional:* Complete the mini project at the end of this worksheet.
- [ ] Read Chapters 5-8 for next week.
- [x] Meet in the AUDITORIUM on Tuesday, September 10.

Chapter Notebooks

Modeling a Bike Share System

Work through the first three sections of the Chapter 2 notebook (*Modeling a bikeshare system, Updating, Functions*). Think about and answer the following questions:

Q1: What happens when this code runs?

```
def a_function():
    print('Hello')

a_function()
```

Hello is printed

Q2: What does a *State* object represent?

holds variables and sets them to a starting value

Constructing a Step Function

Work through the next three sections of the Chapter 2 notebook (*Conditionals, Step, Parameters*), and answer the following questions:

Q3: What happens when this code runs?

```
def step(p1, p2):
    if flip(p1): 0.5
        bike_to_wellesley()
        print('Moving a bike to Wellesley')

    if flip(p2): 0.4
        bike_to_olin()
        print('Moving a bike to Olin')

step(0.5, 0.4)
```

50% of the time, a bike moves to wellesley and 40% of the time, a bike moves to olin

Q4: In the context of the bike share model, why is it useful to move the probabilities outside the step function?

It's useful to move the probabilities out of the step function to make them easily changeable as parameters, moving them outside the scope of the function

Reading Reflection Questions (15 minutes)

1. In Chapters 2-4, you read about the design and simulation of a bike share model. What does that mean? What are the key steps?

Some key steps to creating the bikeshare model include:
- creating starting state specifying bikes at each location as well as other values and metrics
- writing functions to modify the state such as to move bikes from one location to another
- writing a step function to model the movement of the bikes based on probability
- simulate the bikeshare by calling the step function

2. In Chapter 3, we updated the model to avoid negative bikes. How important do you think addressing negative bikes is? To what extent does this depend on the intended purpose of the model?

negative bikes are bad for two reasons:
- they represent customers that would have been lost
- it breaks the math in the model. if you have -3 bikes and you add 1, is it right to have -2 bikes?

however, if the model's goal is to check if the # bikes will stay balanced and above 0, or the goal is to see how many more bikes they might need to meet customer demand, it could be alright to leave in negative bikes.

3. Think about the final version of the bike share model that appears in Chapter 4. What features does this model include? What does it ignore? Of the things it leaves out, which seem most important? Which might be good enough?

features: - state includes bikes@each location, # times there weren't bikes @ a location, a clock counter and the first time a customer couldn't get a bike
- a function to run the simulation with variable probabilities and # steps

missing: - bike docks, if necessary
- travel time
- busy times of day / demand changing

Running a Simulation

Work through the next three sections of the Chapter 2 notebook (*For loop, TimeSeries, and Plotting*).

Q5: Why might you want to vary the parameter `num_steps`?

running the step function multiple times is how you can simulate the bikeshare for an extended period of time, which can produce more accurate results and stronger trends.

Adding a Clock

Work through the *Comparison operators* and *Metrics* sections of the Chapter 3 notebook. Then, work through the first exercise of the Chapter 3 notebook.

Q6: One part of this task is done for you, namely adding a state variable called `clock`. What is the other part?[1]

The other part is making the clock value increase by 1 each time step is called

[1] You can write down the code to be added to the step function; it should be a single line.

state.clock += 1

Modifying the Simulation

Work through the first three sections of the Chapter 4 notebook (*Returning values, Running simulations, More for loops*). Be sure you understand both how to return a value from a function, and how to use a value that is returned.

Q7: What is the difference between `linspace` and `linrange`?

linspace returns array of space # vals spaced out equally whereas
linrange returns array of vals spaced out by step amount

Q8: Why is it useful to test different values of *p1*? Write down an explanatory, predictive, and descriptive modeling question that we could answer by using this simulation.

For what values of p1 are there the least number of unhappy customers?

What is the maximum demand that the bikeshare can handle before there are unhappy customers?

Parameter Sweeping

Work through the next section of the Chapter 4 notebook (*Sweeping parameters*).

Q9: You now know how to get two kinds of results from a simulation: a time series for a single set of parameter values, and a metric swept over multiple parameter values. How are these kinds of results different? What kinds of questions are more usefully answered by one than the other?

a time series shows what happened over time in a certain simulation specifically, while a sweep series shows the general trends when one of the parameters changes.

If you are interested in knowing how many bikes are at dock at a given time, you need a time series, whereas if you instead want to know how the number of unhappy customers in total changes with a certain parameter, a sweep series is better.

Rebalancing the System (15 minutes)

Help! Citi Bike in NYC is experiencing a chronic imbalance between supply and demand at different stations (`https://www.wnyc.org/story/citi-bike-deserts/`). For example, Penn Station typically runs out of bikes on weekday mornings, while so many bikes are taken to the Flatiron District that there aren't enough docks to accept them. Use your experience with the Olin–Wellesley bike share model to think about how you would build a new model to help Citi Bike solve this problem. What kind of work would you want it to do? Recall that we've talked about three kinds of work models can do: prediction, explanation, and design. This is not an exhaustive list!

State: What are the most important elements of the Citi Bike system to include in the model's state? What changes would you have to make to the Olin–Wellesley model to keep track of them?

- need to add # docks at each station
- # bikes available = # docks taken
- need to add more stations?

Parameters: What quantities do you need to describe the condition of the system? Which are the most important for addressing the imbalance problem?

- flatiron, penn (# bikes)
- flatiron_docks, penn_docks (# docks)

this quantifies how many bikes and how many docks were available

Actions: What needs to happen in a typical time step of the model? Are all time steps the same, or are there special actions that occur in only some time steps?

- at each station, the demand for bikes and for docks should each change based on time of day, determined by the data collected by citi
- could use # of trips data from one station to another to account for demand
- add in travel time based on stop distance?

Metrics: What metrics would you want to collect? How would you compute them? How would Citi Bike use them to address the supply-demand imbalance?

- flatiron_empty, penn_empty (# times a customer found no bikes available)
- flatiron_full, penn_full (# times a customer found nowhere to park their bike)

citi bike could use these metrics to determine the optimal # of docks and bikes for each station

Worksheet Reflection Questions (15 minutes)

SweepSeries(
data = mp.empty(0),
index = [0, 0.1 ... 0.9])
↑
the p's
for example

1. **Which programming concepts (if any) were new to you? What strategies were effective in helping you become familiar with them? If you were comfortable with these concepts already, in what ways were you able to help others in the studio become more comfortable with them (perhaps through your pair programming experience)?**

the python syntax was new to me.
I was able to help my programming partner become more comfortable with programming concepts by explaining how the code works to the best of my ability, such as classes and returns.

2. **How did pair programming work for you? Who did what, and how did you decide? How did it feel? How would you expect it to feel with some additional practice?**

Kei was driver and I was navigator because I'm more familiar with programming.
Although I felt frustrated at not being able to move forward, I was glad I could help answer her questions and work well together.

3. **The bike share model you've created in Python is still a long way from a model that would be useful to the folks at Citi Bike — or is it? Briefly sketch a strategy for expanding the bike share model from the book to address the imbalance between supply and demand at different stations.**

- write a function to run a full day in the simulation
- run full-day simulations for different starting states

4. **Which parts of this expansion do you see as easiest or hardest? Why?**

writing the function to run one full day is going to be the more difficult part, because I need to figure out what should happen throughout the day in the function, while running the day function is easier, because I just need to run it with sweeping parameters & graph the results

Mini Project: Rebalancing Tactics (Optional)

assumption: no docks necessary?

Now that you've had some good practice with the Olin–Wellesley bike share model, let's return to the question of rebalancing between stations. Suppose that Citi Bike wants to reduce the number of customers who arrive to an empty station by redistributing bikes between two popular neighborhoods: Midtown and the East Village. They want to answer the question of how best to do this using modeling. Below is a table of the average number of daily trips between these two neighborhoods broken down by AM and PM trips for August 2013. This data comes from Citi Bike's publicly available data.

`https://www.citibikenyc.com/system-data`

	Midtown to East Village	East Village to Midtown
AM	93	228
PM	477	281

* does not include unhappy customers

For simplicity, we will restrict our attention of the Citi Bike system to these two neighborhoods and ignore demand for other possible trips. Further, we will assume that there are a total of 200 bikes available.

Think of a model for this system that will allow Citi Bike to answer some version of their rebalancing question. One possibility is to look at redistributing the bikes overnight. In other words, you could design a model that can compare different starting configurations at the beginning of the day in terms of the number of unhappy customers (similar to how we evaluated different values of $p1$ in the Olin-Wellesley model). You may need to make some assumptions to simplify your question or model. Sketch a schematic diagram of your proposed model below. Also, write down how you plan to use this model to systematically answer your proposed question.

starting state
↓
day(state) → runs am trip probabilities, then pm
└→ returns total empties from the whole day

sweep-state() → runs days with diff # bikes starting midtown
→ saves # empties & graphs it
└→ can also save & graph # trips each way in am & pm

Next, create a Jupyter notebook that presents and implements your proposed model. In particular, you should address the QMRI questions in your deliverable:

1 • Question: What is the motivating question? What starting state is most effective at keeping the bike distribution balanced during the day?
2 • Methodology/Model: How are you going to answer it?
3 • Result: What happened?
4 • Interpretation: How does that answer the question?

(2.) write a day function, sweep through diff bike distributions and plot to see what distribution causes the least unhappy customers

(3.) The graph was inconsistent due to using rnd(), so I averaged each over 10 days to get a smoother graph. Approximately 130-140 bikes at midtown seems to be the minimum on the graph. (4.) The answer is that 130-140 bikes starting at midtown is the most effective distribution of the bikes, on average

If you would like to read more about rebalancing:

1. Kaufman, Sarah M., et al. "Citi Bike: the first two years." 2015. https://wagner.nyu.edu/files/rudincenter/2015/06/Citi_Bike_First_Two_Years_RudinCenter.pdf
2. Jian, Nanjing, et al. "Simulation optimization for a large-scale bike-sharing system." Proceedings of the 2016 Winter Simulation Conference, IEEE Press, 2016. (Sections 1-2.3) https://www.informs-sim.org/wsc16papers/054.pdf
3. Visualization of the Citi Bike system: https://urbica.github.io/citibike/