**Design Thoughts & Initial Exercises:**

1. Write simple Python code with Physicians, Consumers and their daily interactions for 2 years (say 250days/year\*2years = 500 time slots).
2. Extend above to include Pharmacists.
3. Extend above to include Targets, Profiles.
4. Add in Promotions.
5. Stop here and revisit.

**Focusing on Problem 1:**

Create two Classes for Entities: Physician, Consumer

Create One Class for setting up and running simulation models: SimModel

Run Simulation by calling corresponding methods of SimModel

Parallel Idea on Tagging (Tags): org\_tag, formulary\_tag, promo\_tag, age\_group\_tag, Group\_Practice\_Tag

**Class 1: Consumer**

**Attributes:**

Cust\_id, age, gender, lat, long, zip, zip3, primary\_ins,

Phys List, Comorbidity List, fill\_probability

Prevnar [Status, Date Taken], P23 [Status, Date Taken], Vax[Status, Date Taken], V116[Status, Date Taken],

Transactions [type, source, date, treatment\_desc] {ex: Med\_Claims, Pharm\_Rx, Lab\_Claims as Transaction Type},

HC\_appointments{appointment\_date:doc\_id}

**Methods:**

visit\_doc() – logs when a patients visits a doctor

get\_presc() – logs when a patient gets a prescription

fill\_presc() – fills using fill\_probabilty and then logs if successful

schedule\_appointment() – used to search physician’s availability and take that slot. Log it.

**Class 2: Physician**

**Attributes:**

specialty, spec\_group, target\_seg, org\_tag, optionally -- age, gender

patient\_list, appointments{date\_slot : [patient ids]}, max\_appointments{date\_slot : num\_pats}, available\_appointment\_slots[list of date slots]

hcp\_insurances[insurance ids], Rx\_Counts{brand : counts} – Prevnar, P23, Vax, V116

**Methods:**

assign\_patients(patient list) – assigns a set of patients to physician

initialize\_appointments() – fill all appointment slots to be used before simulation starts running.

set\_appointment(patient\_id, date\_slot) – checks if appointment is available on that day and if available assigns the patient to physician’s appointment list and returns true. If appointment is not available returns false. Updates available\_appointment\_slots[] if all appointments are full for the day.

decide\_prescription(patient\_id) – decides if the patient needs prescription and if so which one and then prescribes for the patient (i.e., call get\_prescription(brand) of the patient). Then call fill \_presc() of the patient. Log the decide counts.

**Simulation Execution:**

Class 3: **SimModel**

**Attributes:**

N\_Time\_Slots, time\_slot\_range – range from 1 to N (used for looping through)

Phys\_Count. Phys\_List

spec\_group\_list[], org\_tag\_list(), zip3\_list()

Patients per Physician distribution – Normal Mean and Std Dev say, (1200, 300)

ref\_patient\_appointments - a table with columns pat\_age\_group, visits\_per\_year\_mean, visits\_per\_year\_stdev

ref\_phys\_max\_daily\_appointments – a table with columns spec\_grp, mean, stdev

**Methods:**

init() – initializes environment at time 0.

simulate() – loops through each day of simulation.

step(date\_slot) – specifies what happens on each day slot of the simulation

report\_slot\_activities(date\_slot) – reports aggregate metrics after end of each day slot.

report\_sim\_summary() – reports aggregate metrics after end of simulation.

Details for init() method:

1. Initialize N\_Time\_Slots, time\_slot\_range
2. Create Phys Count amount of physicians and add to Phys\_List. Assign spec\_grp, org\_tag, zip3

For each physician:

1. Call phys.assign\_patients() This does the following
   1. Assign patients for each physician (based on patients per physician distribution)
   2. Each of Physician’s patient gets same zip3 as the physician
   3. Assign age, gender, claim\_type (govt, commercial), ins\_tag for each patient
2. Call phys.initialize\_appointments() This does the following
   1. Initializes phys.appointments attribute with date\_slot key and empty lists as values
   2. Initializes max\_appointments{date\_slot : num\_pats}, available\_appointment\_slots[list of date slots]
   3. For each patient of the physician:
      1. If available\_appointment\_slots[] is empty then break out of the loop.
      2. Get number of total visits per year and project to total slot years (i.e., 2 years)
      3. Draw random date slot from available\_appointment\_slots[] and call set\_appointment() for the patient and given date slot. This set\_appointment() internally does the following:

Updates physician appointment schedule, updates available\_appointment\_slots[]

* + 1. Repeat until number of appointemnts needed in step ii. If no more appointments are available, then break out of patient loop [(c) above]

Details about simulate() method:

1. For each slot in date\_slots call step(slot\_index)
2. Report\_aggr\_stats()

Details about step() method:

Step(slot\_idx) method defines what happens on each day of the simulation.

1. For each phys:
   1. For each patient
      1. pat.visit\_doc()
      2. phys.decide\_rx()
      3. pat.get\_presc()
      4. pat.fill\_rx()
      5. pat.schedule\_appointment()
   2. summarize daily rxs for each phys and log it
2. Summarize all phys activity for the slot day and log it (report\_slot\_activity())