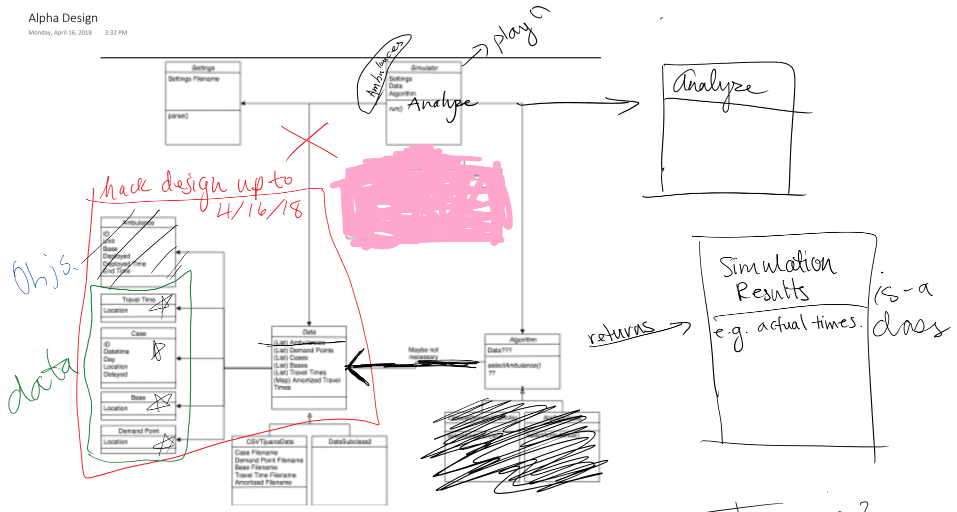
Alpha Design

Monday, April 16, 2018 3:32 PM

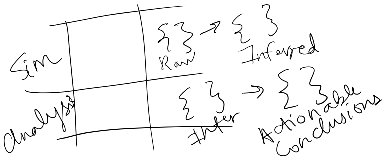


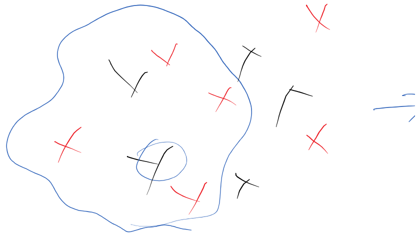
$a1 = \text{Ambulance}()$

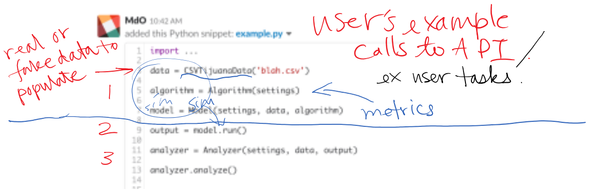
$a1.\text{field1} = \text{object}$

1000

$b_n = 6$







MdO 10:42 AM
added this Python snippet: example.py

```python
import ...

data = CSVTijuanaData('blah.csv')

algorithm = Algorithm(settings)

model = Model(settings, data, algorithm)

output = model.run()

analyzer = Analyzer(settings, data, output)

analyzer.analyze()
```

real or fake data to populate

1

2

3

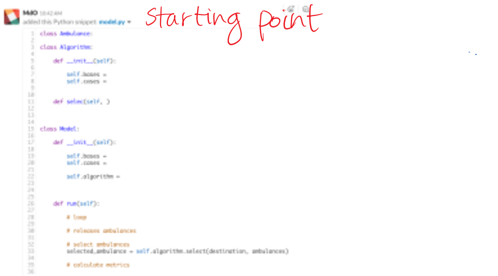
sim

sim

metrics

User's example calls to API

ex user tasks.



MdO 10:42 AM
added this Python snippet: model.py

```python
class Ambulance:

class Algorithm:

    def __init__(self):

        self.bases =
        self.cases =


    def select(self, )



class Model:

    def __init__(self):

        self.bases =
        self.cases =

        self.algorithm =



    def run(self):

        # loop

        # releases ambulances

        # select ambulances
        selected_ambulance = self.algorithm.select(destination, ambulances)

        # calculate metrics
```

Starting point

User Tasks

- Run the simulation by providing the arguments/functions.

Models (Framework)

Simulation has ambulances, bases, demands

Algorithms include travel time, city coverage, and selection.

Analysis includes average, std dev, min, max.

Extra: define choice during tension of conflicting metrics.

*Will work because pass func pointers.

User Tasks

1) Load files to models, go w/ default settings or pick own.
2) Run using algos
3) analyze results

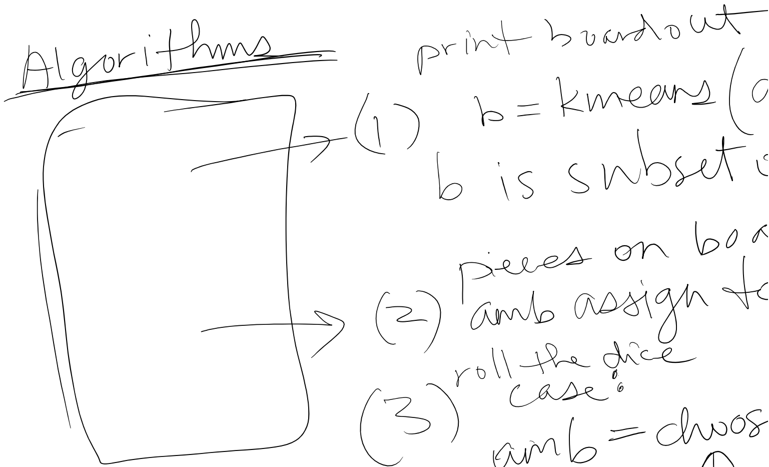
Hack → Design

Readers { Bases
Cases
Demands

1- Open CSV where first row is name of data
2- Determine whether CSV has appropriate fields
3- Throw exception if not.

like [illegible] include numer [illegible] avoid being

Algorithms



print boardout

(1) b = kmeans(

b is subset

(2) pieces on boa

amb assign to

(3) roll the dice

case

amb = choos

case

include travel time.

output = fn (inputs)

[ ] type(base)

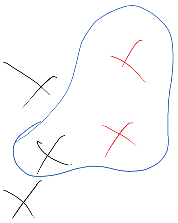
[ ] type (base)

1- bases

Out: bases

params

(bases) returns bases



all bases)

of all bases

rd

o bases

e (ambs)

es.

Demands | CSV has appropri
fields
3 - Throw exception if not.

Libs — include numpy — try ~~to~~ avoid being computationally costly.

Algorithms

Allow picking specific function

$Y = f(x)$

Pass f, check arity of (x) and return type of y. (Cant do this statically: recommend ~~that~~ a small simulation is run first.

When the simulation finishes, there should be a returned object containing all the raw data.

Analysis algorithms

e.g. for each field, calc the mean, median and std dev.

allow passing additional fns so

list of fns? [fn1, fn2, ..., fn n].

Results should output to a new file.

results_n.csv   results_

results_22.csv.

case1 | case 2
[stat 1
stat 2] | [ ]

Stat 1
travel [case 1: 20min , case 2: 40min]

stat 2
ambs [1 , 2 , ...]

Out : bases

def choose (bases / demands) returns bases

params

fn(bases, demands)

def choose(.... func=default_kmeans)

settings.conf ⟶ settings_<datetime>_<h

ds = DispatchSimulation (settings)

results = ds.play(...)
⟶ save to file.

lists of lists

be able to
reproduce the sim
using this file.

ash>.conf