**Question 13.2**

*In this problem you, can simulate a simplified airport security system at a busy airport. Passengers arrive according to a Poisson distribution with λ1 = 5 per minute (i.e., mean interarrival rate μ1 = 0.2 minutes) to the ID/boarding-pass check queue, where there are several servers who each have exponential service time with mean rate μ2 = 0.75 minutes. [Hint: model them as one block that has more than one resource.] After that, the passengers are assigned to the shortest of the several personal-check queues, where they go through the personal scanner (time is uniformly distributed between 0.5 minutes and 1 minute).*

I used the Python package Simpy to complete this problem. The code is shown below.

import simpy

import numpy as np

from functools import partial, wraps

import matplotlib.pyplot as plt

import seaborn as sns

class Passenger(object):

def \_\_init\_\_(self, env, name, bpc, pc, travel\_time, board\_pass\_wait\_time, personal\_check\_wait\_time):

self.env = env

self.name = name

self.bpc = bpc

self.pc = pc

self.travel\_time = travel\_time

self.board\_pass\_wait\_time = board\_pass\_wait\_time

self.personal\_check\_wait\_time = personal\_check\_wait\_time

# Start the begin\_trip process everytime an instance is created.

self.action = env.process(self.begin\_trip())

def begin\_trip(self):

# Simulate driving to the airport

yield self.env.timeout(self.travel\_time)

start\_time = self.env.now

# Begin waiting in line for ID/boarding-pass check

print('%s arriving at boarding-pass check queue at %d' % (self.name, self.env.now))

with self.bpc.request() as req:

yield req

# Begin ID/boarding-pass check

print('%s starting boarding pass check at %s' % (self.name, self.env.now))

yield self.env.timeout(self.board\_pass\_wait\_time)

print('%s leaving the boarding pass check queue %s' % (self.name, self.env.now))

# Begin waiting in personal-check queue

with self.pc.request() as req:

yield req

# Begin personal-check

print('%s starting personal-check at %s' % (self.name, self.env.now))

yield self.env.timeout(self.personal\_check\_wait\_time)

print('%s leaving the personal-check queue %s' % (self.name, self.env.now))

end\_time = self.env.now

self.wait\_time = end\_time - start\_time

def patch\_resource(resource, pre=None, post=None):

"""Patch \*resource\* so that it calls the callable \*pre\* before each

put/get/request/release operation and the callable \*post\* after each

operation. The only argument to these functions is the resource

instance.

"""

def get\_wrapper(func):

# Generate a wrapper for put/get/request/release

@wraps(func)

def wrapper(\*args, \*\*kwargs):

# This is the actual wrapper

# Call "pre" callback

if pre:

pre(resource)

# Perform actual operation

ret = func(\*args, \*\*kwargs)

# Call "post" callback

if post:

post(resource)

return ret

return wrapper

# Replace the original operations with our wrapper

for name in ['put', 'get', 'request', 'release']:

if hasattr(resource, name):

setattr(resource, name, get\_wrapper(getattr(resource, name)))

def monitor(data, resource):

"""This is our monitoring callback."""

item = (

resource.\_env.now, # The current simulation time

resource.count, # The number of users

len(resource.queue), # The number of queued processes

)

data.append(item)

def run\_airport\_simulation(num\_passengers, pass\_mean\_interarrival\_rate, num\_board\_pass\_checkers, num\_personal\_check\_checkers):

data\_bpc = []

data\_pc = []

env = simpy.Environment()

bpc = simpy.Resource(env, capacity=num\_board\_pass\_checkers)

pc = simpy.Resource(env, capacity=num\_personal\_check\_checkers)

bpc\_monitor = partial(monitor, data\_bpc)

patch\_resource(bpc, post=bpc\_monitor)

pc\_monitor = partial(monitor, data\_pc)

patch\_resource(pc, post=pc\_monitor)

passenger\_list = []

travtime = 0

for i in range(num\_passengers):

bptime = np.random.exponential(scale=0.75)

pctime = np.random.uniform(low=0.5, high=1.0)

passenger\_list.append(Passenger(env, 'Passenger %d' % i, bpc, pc, travtime, bptime, pctime))

travtime = travtime + np.random.exponential(scale=pass\_mean\_interarrival\_rate)

env.run()

return data\_bpc, data\_pc, passenger\_list

num\_passengers = 1000

pass\_mean\_interarrival\_rate = 0.02

num\_board\_pass\_checkers\_list = [15, 16, 17]

num\_personal\_check\_checkers\_list = [15, 16, 17]

fig, axtuple = plt.subplots(len(num\_board\_pass\_checkers\_list), len(num\_personal\_check\_checkers\_list), figsize=(18, 9))

plotnum = 0

for n0, num\_board\_pass\_checkers in enumerate(num\_board\_pass\_checkers\_list):

for n1, num\_personal\_check\_checkers in enumerate(num\_personal\_check\_checkers\_list):

datbpc, datpc, pl = run\_airport\_simulation(num\_passengers, pass\_mean\_interarrival\_rate, num\_board\_pass\_checkers, num\_personal\_check\_checkers)

wait\_time\_list = [passenger\_instance.wait\_time for passenger\_instance in pl]

sns.distplot(wait\_time\_list, ax=axtuple[n0, n1])

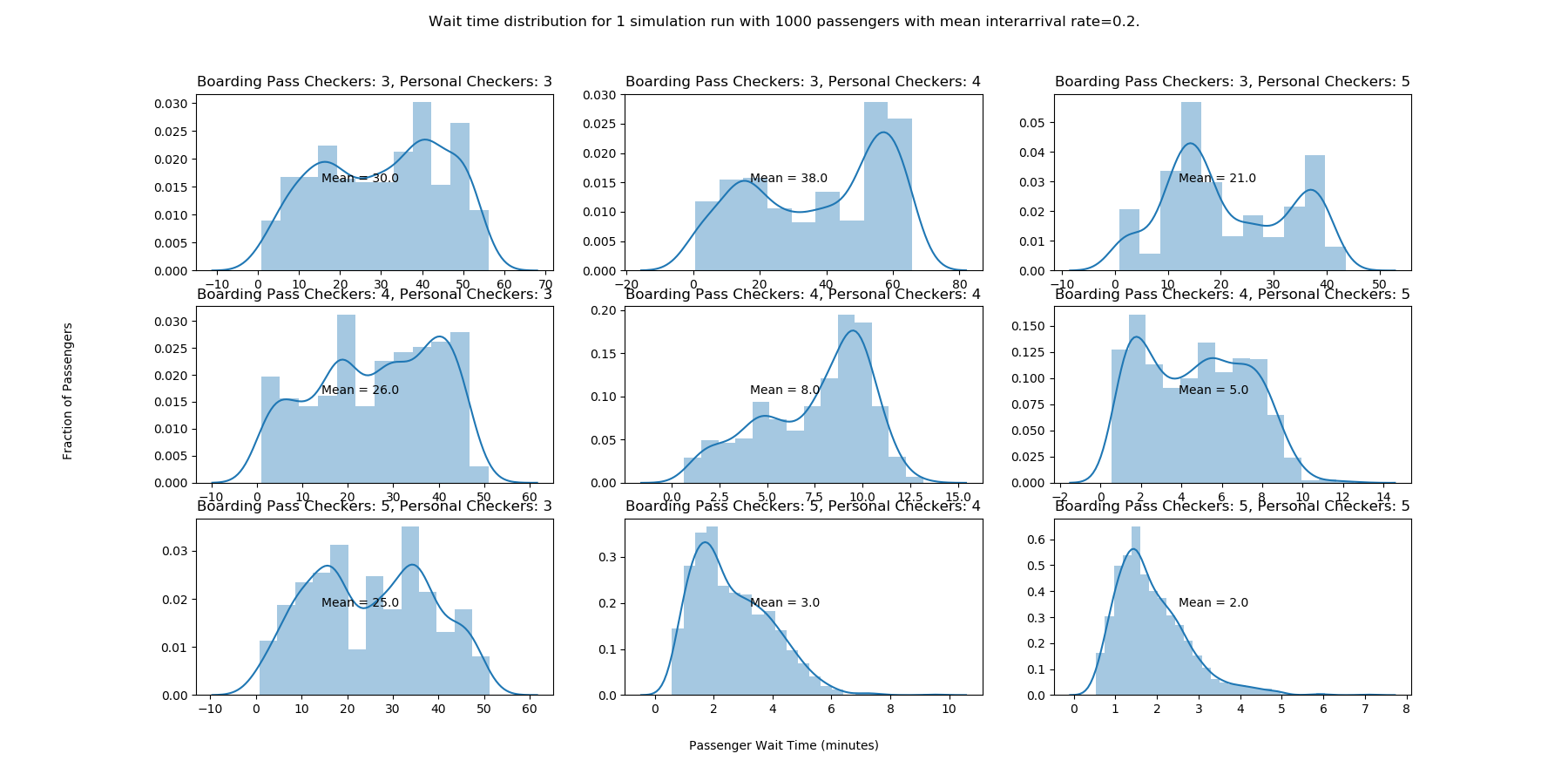
axtuple[n0, n1].set\_title("Boarding Pass Checkers: {}, Personal Checkers: {}".format(num\_board\_pass\_checkers, num\_personal\_check\_checkers))

axtuple[n0, n1].text(0.35, 0.5, "Mean = {}".format(round(np.mean(wait\_time\_list))), transform=axtuple[n0, n1].transAxes)

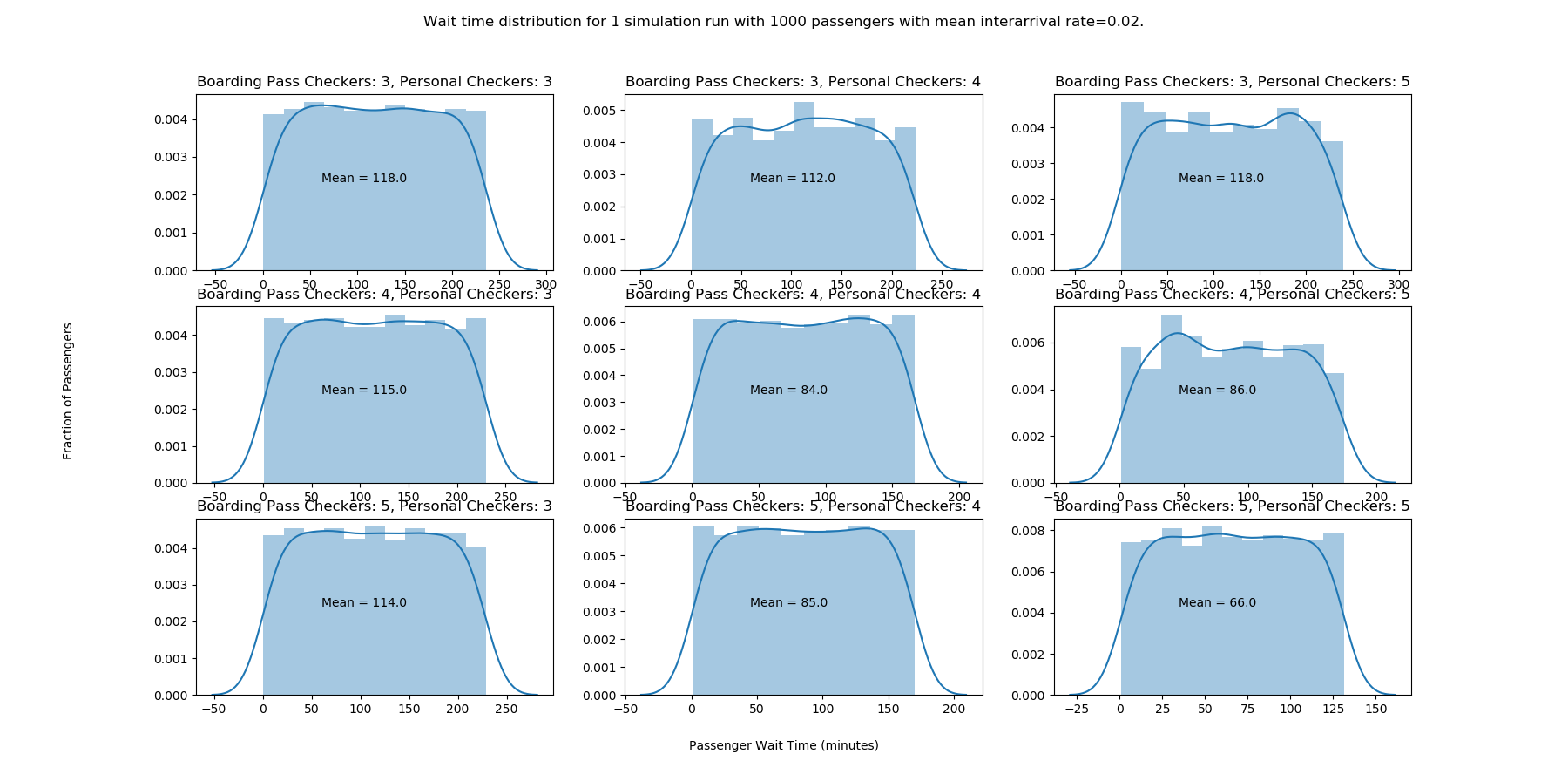
fig.text(0.5, 0.04, "Passenger Wait Time (minutes)", ha='center')

fig.text(0.04, 0.5, "Fraction of Passengers", va='center', rotation='vertical')

fig.suptitle('Wait time distribution for 1 simulation run with {} passengers with mean interarrival rate={}.'.format(num\_passengers, pass\_mean\_interarrival\_rate))

The first analysis was completed with a mean passenger arrival rate of 5 passengers per minute (mean passenger interarrival time of 0.2 minutes). This was simulated for 1000 passengers with all 9 permutations of 3, 4 or 5 boarding-pass checkers and personal-checkers. Shown below are the plots of the distribution of wait times for all 1000 passengers for one run of each permutation. 

From this we see that 4 boarding-pass checkers and 4 personal-checkers is the minimum required for the average wait time to be below 15 minutes. Let’s take a look at how the wait time changes for these amounts of resources if we bump the passenger arrival rate to 50 passengers per minute (interarrival time of .02 minutes):



It is clear that the 10x increase in passenger arrival rate has overrun the resource combinations which produced average wait times less than 15 minutes in the first scenario. In fact, with this higher arrival rate, the average wait time does not dip below 15 minutes until we supply the airport check in with 17 boarding-pass checkers and 17 personal-checkers. See below:

