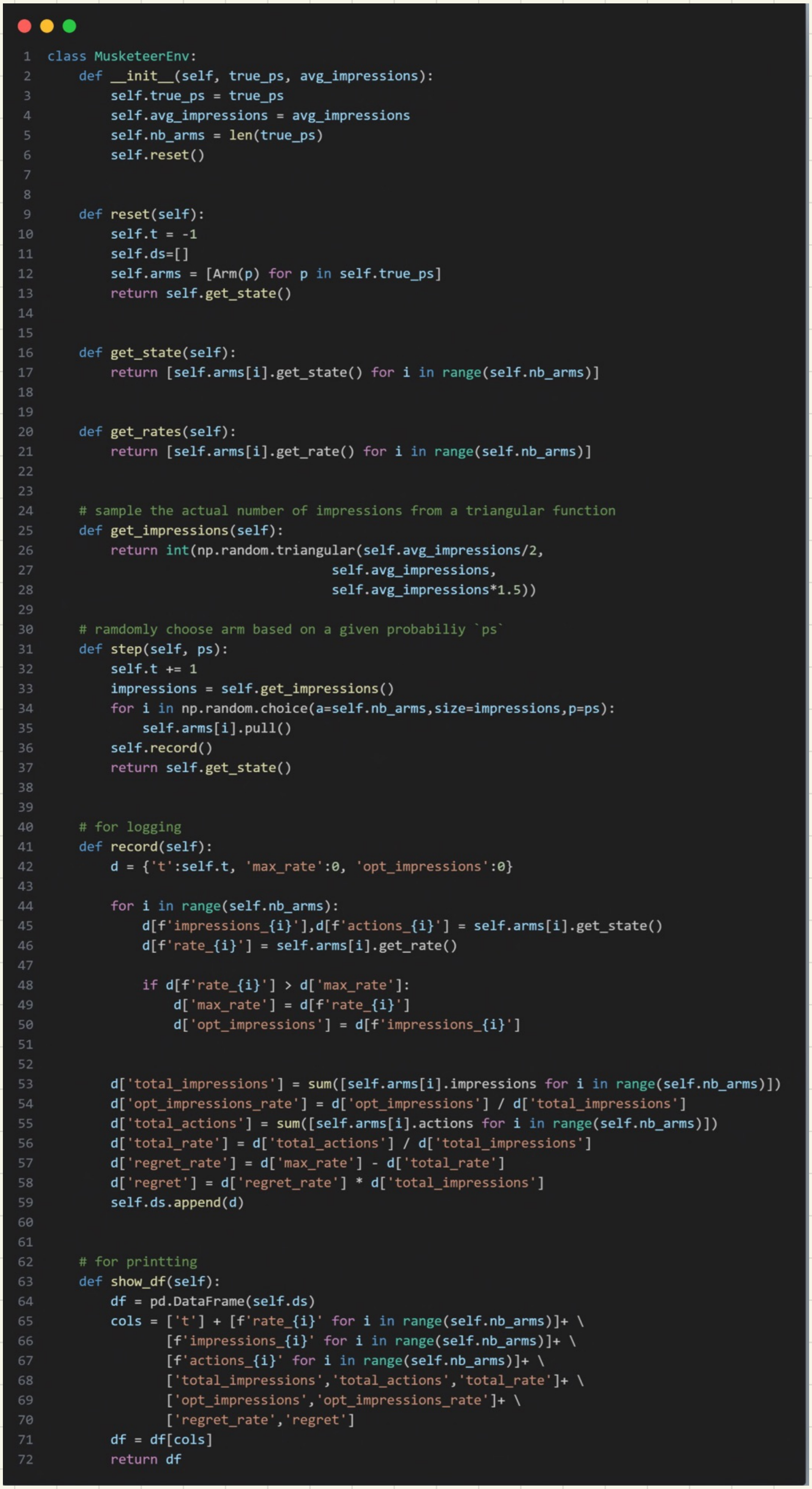
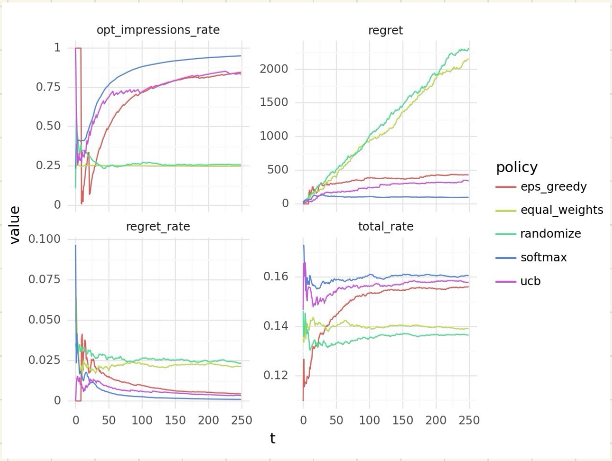


```python
class MusketeerEnv:
    def __init__(self, true_ps, avg_impressions):
        self.true_ps = true_ps
        self.avg_impressions = avg_impressions
        self.nb_arms = len(true_ps)
        self.reset()


    def reset(self):
        self.t = -1
        self.ds=[]
        self.arms = [Arm(p) for p in self.true_ps]
        return self.get_state()


    def get_state(self):
        return [self.arms[i].get_state() for i in range(self.nb_arms)]


    def get_rates(self):
        return [self.arms[i].get_rate() for i in range(self.nb_arms)]


    # sample the actual number of impressions from a triangular function
    def get_impressions(self):
        return int(np.random.triangular(self.avg_impressions/2,
                                        self.avg_impressions,
                                        self.avg_impressions*1.5))

    # ramdomly choose arm based on a given probabiliy `ps`
    def step(self, ps):
        self.t += 1
        impressions = self.get_impressions()
        for i in np.random.choice(a=self.nb_arms,size=impressions,p=ps):
            self.arms[i].pull()
        self.record()
        return self.get_state()


    # for logging
    def record(self):
        d = {'t':self.t, 'max_rate':0, 'opt_impressions':0}

        for i in range(self.nb_arms):
            d[f'impressions_{i}'],d[f'actions_{i}'] = self.arms[i].get_state()
            d[f'rate_{i}'] = self.arms[i].get_rate()

            if d[f'rate_{i}'] > d['max_rate']:
                d['max_rate'] = d[f'rate_{i}']
                d['opt_impressions'] = d[f'impressions_{i}']


        d['total_impressions'] = sum([self.arms[i].impressions for i in range(self.nb_arms)])
        d['opt_impressions_rate'] = d['opt_impressions'] / d['total_impressions']
        d['total_actions'] = sum([self.arms[i].actions for i in range(self.nb_arms)])
        d['total_rate'] = d['total_actions'] / d['total_impressions']
        d['regret_rate'] = d['max_rate'] - d['total_rate']
        d['regret'] = d['regret_rate'] * d['total_impressions']
        self.ds.append(d)


    # for printting
    def show_df(self):
        df = pd.DataFrame(self.ds)
        cols = ['t'] + [f'rate_{i}' for i in range(self.nb_arms)]+ \
               [f'impressions_{i}' for i in range(self.nb_arms)]+ \
               [f'actions_{i}' for i in range(self.nb_arms)]+ \
               ['total_impressions','total_actions','total_rate']+ \
               ['opt_impressions','opt_impressions_rate']+ \
               ['regret_rate','regret']
        df = df[cols]
        return df
```



Todo s: According to the graph, Softmax has the best performance