contextual: Simulating Contextual Multi-Armed Bandit Problems in R

Robin van Emden
JADS

Eric Postma
Tilburg University

Maurits Kaptein
Tilburg University

Abstract

Contextual multi-armed bandits have been gaining ever more popularity due to their effectiveness in solving previously computationally intractable partial information sequential decision problems - from online advertising and recommender systems to clinical trial design and personalized medicine. A popularity both inspired by and inspiring an ever-growing body of predominantly analytically oriented research on ever more sophisticated Contextual Bandit algorithms. At the same time, there are as of yet surprisingly few options that enable researchers and practitioners to simulate and compare the wealth of new and existing Bandit algorithms in a practical, standardized and extensible way. To help close this gap between analytical research and real-life application the current paper introduces the R package **contextual**: a user-friendly and easily extensible framework that facilitates the comparison of, amongst others, contextual and non-contextual Bandit policies through both simulation and offline analysis.

Keywords: contextual multi-armed bandits, simulation, sequential experimentation, R.

A vignette for the van Emden, Kaptein, and Postma (2018) paper.

1. Introduction

There are many real-world situations in which we repeatedly have to decide between a set of options, yet only learn about the best course of action by testing one choice after the other, one step at a time. Such problems are deceptively easy to state but have proven to have broad statistical and practical implications and applications. To get a better grip on such decision problems, and to learn why specific strategies might be more successful than others, they have been studied extensively under the moniker of “Multi-Armed Bandit” problems. Here, these multi-armed bandits are defined as a statistical and machine learning concept in which a so-called agent follows the advice of an algorithm or “policy” to optimize the overall reward it receives in a sequential decision problem with limited information. That is, a MAB policy suggests an agent when to explore new options and when to exploit known ones – where, importantly, for each decision, at each time step t, the only new information the agent acquires is the reward for its latest decision. The agent remains in the dark about the potential rewards of the unchosen options and about any other information outside of current and past rewards and choices made.

In that respect, MAB problems reflect dilemmas we all encounter on a daily basis: do you stick to what you know and receive an expected result ("exploit") or choose something you don’t know all that much about and potentially learn something new ("explore")?

- Do you feed your next coin to the one-armed bandit that paid out last time, or do you test your luck on another arm, on another machine?

- When going out to dinner, do you explore new restaurants, or do you exploit familiar ones?
- Do you stick to your current job, or explore and hunt around?
- Do I keep my current stocks, or change my portfolio and pick some new ones?
- As an online marketer, do you try a new ad, or keep the current one?
- As a doctor, do you treat your patients with tried and tested medication, or do you prescribe a new and promising experimental treatment?

Though MAB models have already proven powerful of their own accord, a recent generalization, known as the **contextual** Multi-Armed Bandit (cMAB), adds one important element to the equation: in addition to past decisions and their rewards, cMAB agents are able to make use of side information about the state of the world at each t before making their decision. In other words, an agent that follows the advice of a cMAB policy may decide differently in different contexts.

This access to side information makes cMAB algorithms even more adept to many real-life decision problems than its MAB progenitors: do you show a certain add to returning customers, to new ones, or both? Do you prescribe a different treatment to male patients, female patients, or both? In the real world, it appears almost no choice exists without a context. So it may be no surprise that cMAB algorithms have found many applications: from recommender systems and advertising to health apps and personalized medicine. A practical applicability that has led to a multitude of new, often analytically derived bandit algorithms or policies, each with their own strengths and weaknesses.

Yet though cMAB algorithms have gained much traction in both research and industry, they have mostly been studied mathematically and analytically – as of yet, comparisons on simulated, and, importantly, real-life large-scale offline "partial label" data sets have been lacking. To this end, the current paper introduces the **contextual** R package. A package that aims to facilitate the simulation, offline comparison, and evaluation of (Contextual) Multi-Armed bandit policies. Though there exists one R package for basic MAB analysis, there is, as of yet, no extensible and widely applicable R package that is able to analyze and compare, respectively, basic K-armed, Continuum, Adversarial and Contextual Multi-Armed Bandit Algorithms on either simulated or online data.

In section 2, this paper will continue with a more formal definition of MAB and a CMAB problems. In section 3, we will continue with an overview of **contextual**'s general implementation. In section 4, we list our implemented polices, and simulate a MAB and a cMAB policy. In section 5, we demonstrate how easy it is to add and simulate your own custom policy. In section 6, we replicate two papers, thereby demonstrating how to test policies on offline data sets. Finally, in section 7, we will go over some of the additional features in the package, and conclude with some comments on the current state of the package and possible enhancements.

2. Contextual Multi-Armed Bandits

On formalizing our Multi-Armed Bandit problem, the aforementioned sequential decision maker's exploit/explore dilemma can be captured by defining a finite set (or **bandit)** of K i.i.d. options (the **arms** of the bandit) each with their own, unknown**,** reward distribution $v_1,\ldots,v_k$ with means $\mu \ldots \mu_k$. Next we define an **agent**, who has to decide between the exploration of unknown arms and

the exploitation of known arms in K in order to maximize its total **reward** (that is, to maximize its cumulative reward $\sum_{t=1}^{T} r_t$ [1]) over a period of time T by following the advice of a **policy** $\boldsymbol{\pi}$ which keeps track of **parameters** $\boldsymbol{\theta}$ that are updated when new information (reward r awarded by the bandit when the agent has chosen an arm) becomes available. This process is repeated T times, where T is often defined as the Bandit's "horizon".
That is, an agent repeats the following lines one at a time at each time step t in t=1,2,...,T:

1a) Agent asks policy π which of the bandit's K arms to choose

1b) Policy π advices action a_t based on the state of a set of parameters θ_t

2a) Agent does action a_t by playing the suggested bandit arm.

2b) Bandit rewards the agent with reward r_t for action a_t,

3a) Agent sends the reward r_t to policy π

3b) Policy π uses r_t to update the policy's set of parameters θ_t

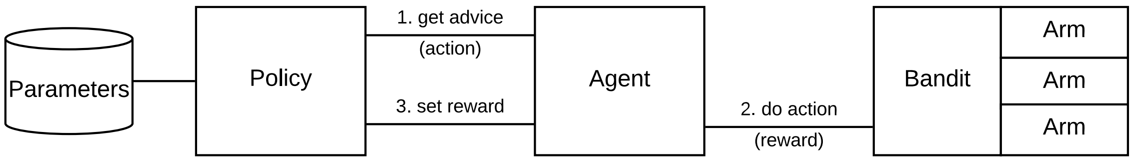

Figure 1: Overview MAB formalization towards **contextual**'s implementation

To allow for side information, that is, to generalize this formalization to a *contextual* Multi-Armed Bandit model, the model needs just one additional step. Again, an agent repeats the following lines for each time step t in t=1,2,...,T:

1a) Agent checks the bandit for side information that might influence the expression of its arms

1b) Bandit returns feature vector Xt

2a) Agent asks policy π which of the bandit's K arms to choose given Xt

2b) Given Xt, policy π advices action a_t based on the state of a set of parameters θ_t

3a) Agent does action a_t by playing the suggested bandit arm.

3b) Bandit rewards the agent with reward r_t for action a_t,

4a) The agent sends the reward r_t to policy π

4b) Policy π uses r_t to update the policy's set of parameters $\theta_t$given Xt

[1]or to minimize its cumulative or expected regret

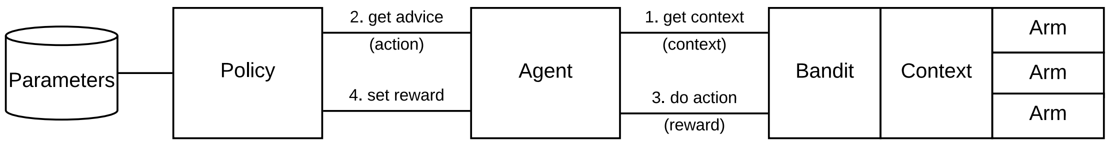

Figure 2: Overview of cMAB formalization towards **contextual**'s implementation

As a matter of fact, by setting feature vector X to [1] for each t in step 1b, the suggested cMAB model perfectly emulates a non-contextual MAB model, easing the comparison and implementation of both MAB and cMAB substantially

3. Implementation

Here, explain how implementation flows naturally from the above

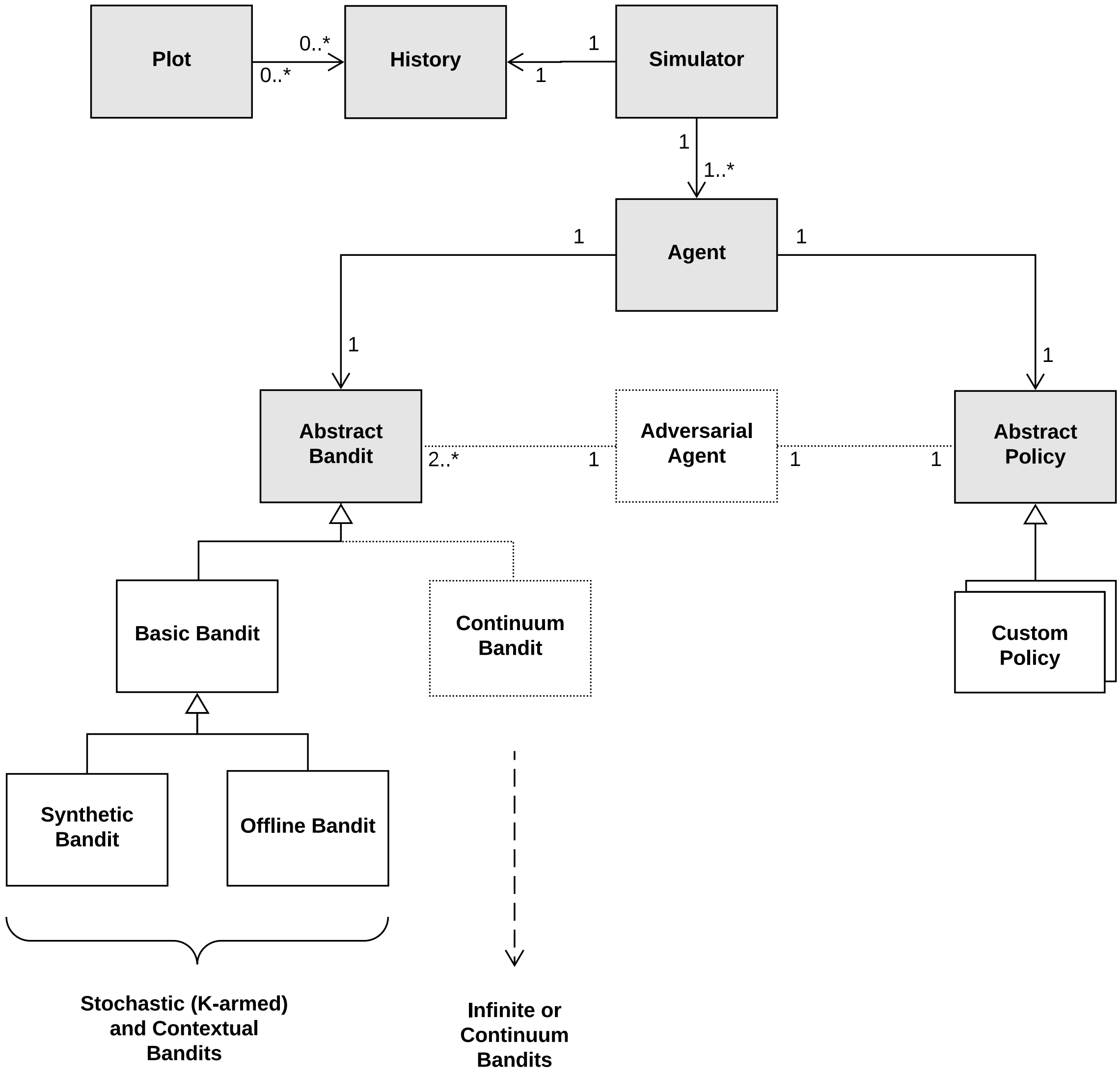

Figure 3: **contextual** UML Class Diagram

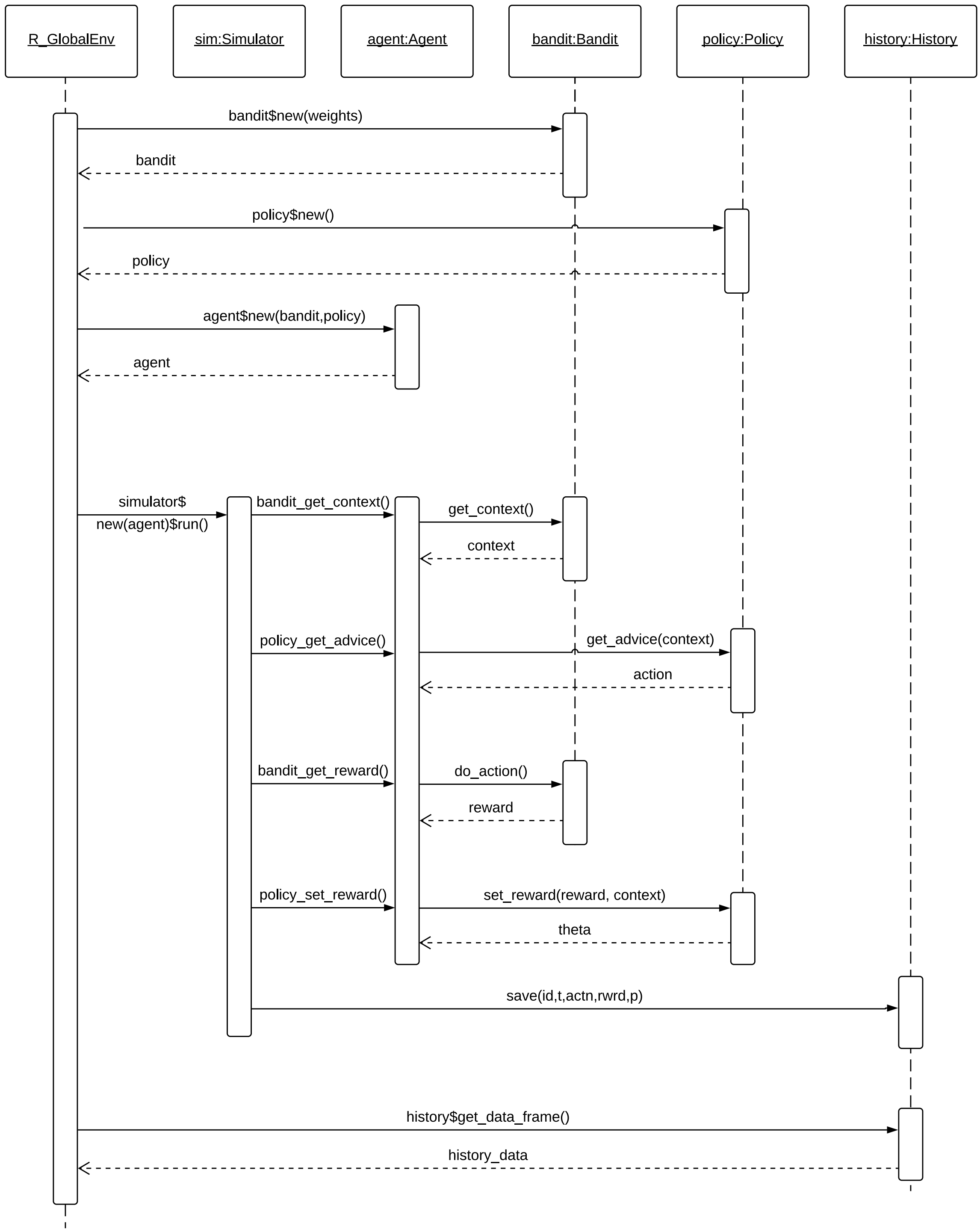

Figure 4: **contextual** UML Sequence Diagram

```
library("contextual")
```

```
horizon             <- 100
simulations         <- 100
weight_per_arm      <- c(0.9, 0.1, 0.1)

policy              <- EpsilonGreedyPolicy$new(epsilon = 0.1, name = "EG")
bandit              <- SyntheticBandit$new(weights = weight_per_arm)

agent               <- Agent$new(policy,bandit)

simulator           <- Simulator$new(agents = agent,
                                     horizon = horizon,
                                     simulations = simulations)
history             <- simulator$run()

plot(history, type = "cumulative", regret = TRUE)
plot(history, type = "arms")
```

For results, see Figure **??** on page **??**.

4. Object orientation: extending contextual

The R6 package allows the creation of classes with reference semantics, similar to R's built-in reference classes. Compared to reference classes, R6 classes are simpler and lighter-weight, and they are not built on S4 classes so they do not require the methods package. These classes allow public and private members, and they support inheritance, even when the classes are defined in different packages.

One R6 class can inherit from another. In other words, you can have super- and sub-classes.

Subclasses can have additional methods, and they can also have methods that override the superclass methods. In this example of a custom **contextual** bandit, we'll extend BasicBandit and override the initialize() method..

5. Special features

For instance, quantifying variance..

6. The art of optimal parallelisation

There is a very intersting trade of between the amount of parallelisation (how many cores, nodes used) the resources needed to compute a certain model, and the amount of data going to and fro the cores.

PERFORMANCE DATA ——————————————————————

on 58 cores: k3*d3 * 5 policies * 300 * 10000 –> 132 seconds

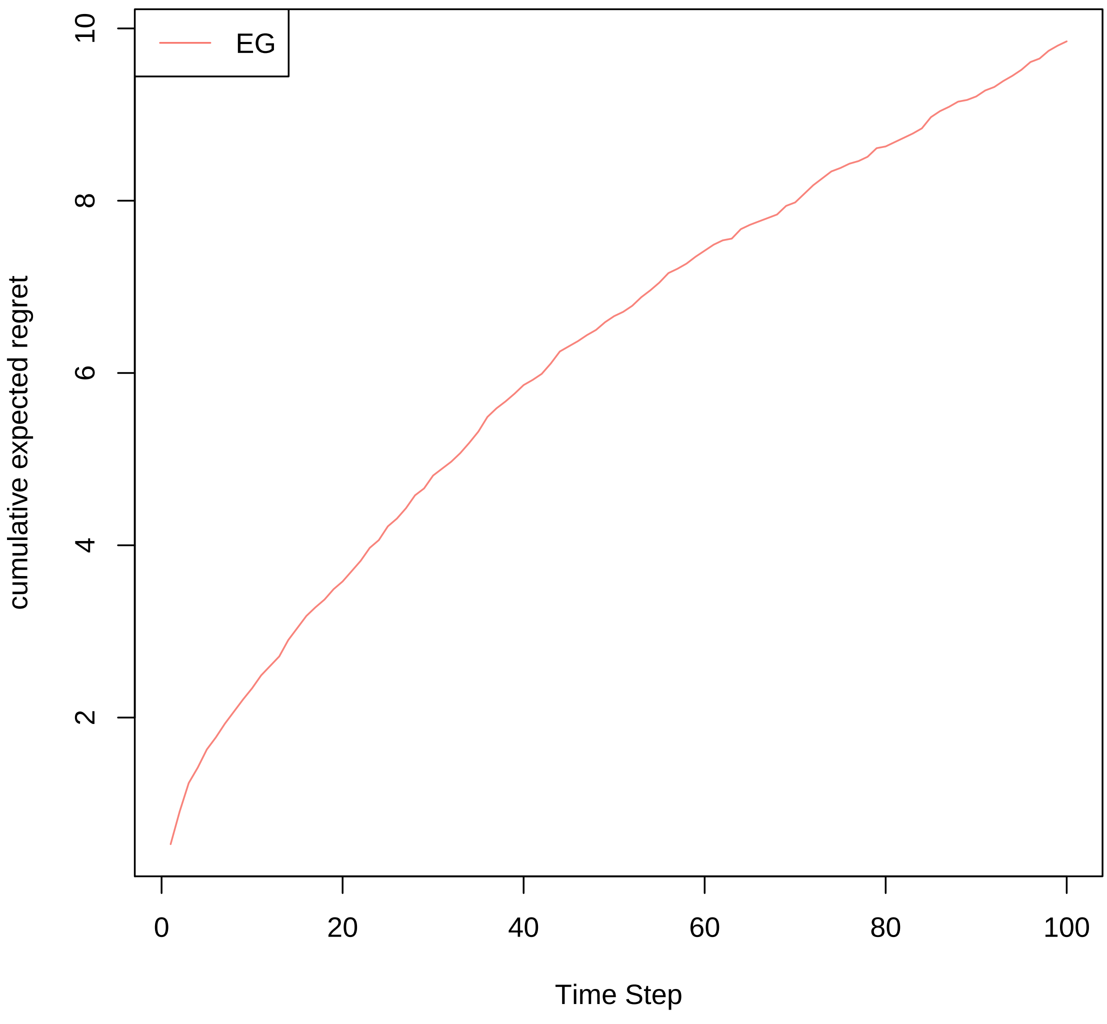

Figure 5: Epsilon Greedy

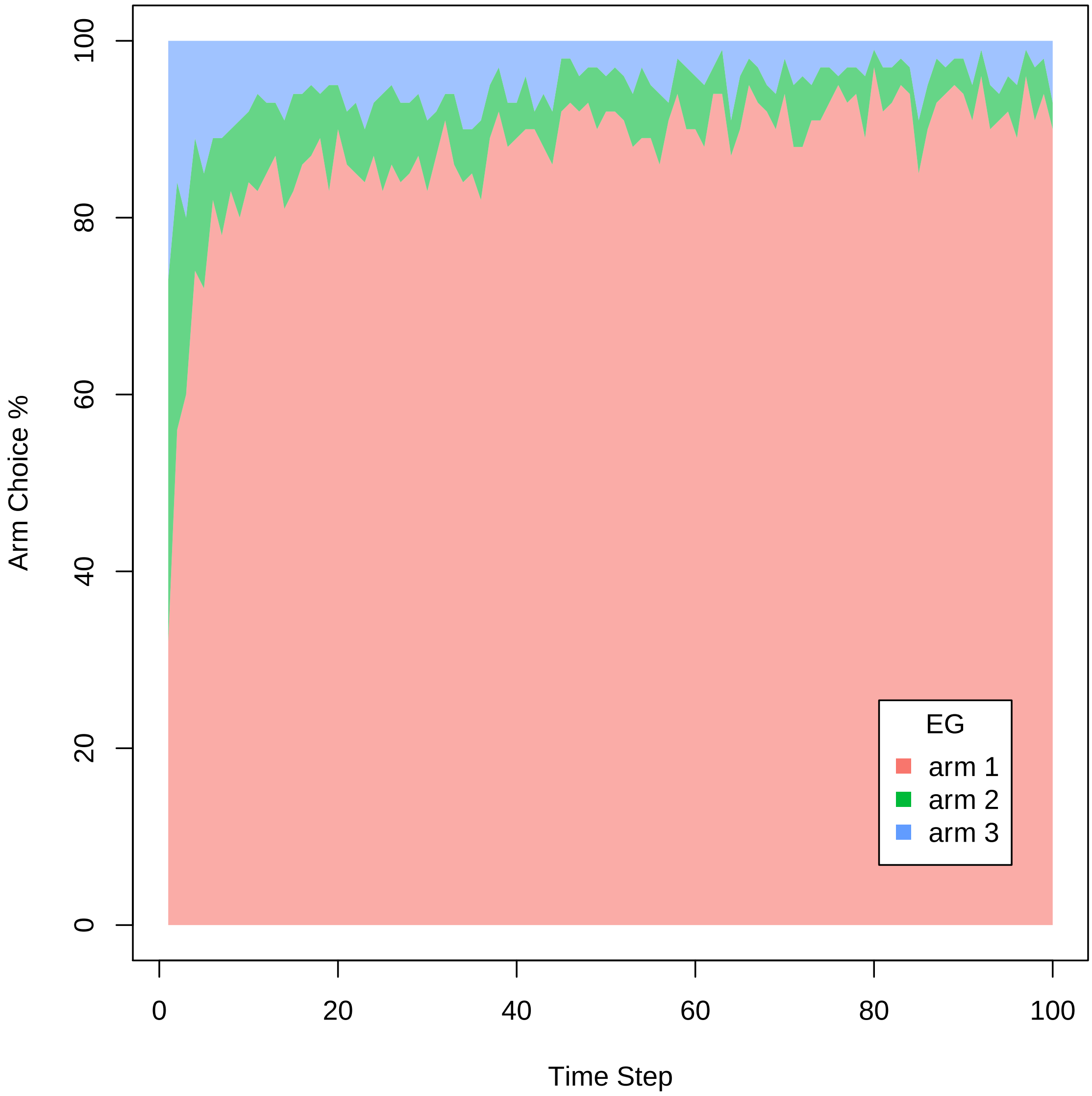

Figure 6: Epsilon Greedy

on 120 cores: k3*d3 * 5 policies * 300 * 10000 –> 390 seconds

—

on 58 cores: k3*d3 * 5 policies * 3000 * 10000 –> 930 seconds

on 120 cores: k3*d3 * 5 policies * 3000 * 10000 –> 691 seconds

7. Extra greedy UCB

Ladila bladibla.

8. Conclusions

Placeholder... the goal of a data analysis is not only to answer a research question based on data but also to collect findings that support that answer. These findings usually take the form of a table, plot or regression/classification model and are usually presented in articles or reports.

9. Acknowledgments

Thanks go to CCC.

References

van Emden R, Kaptein M, Postma E (2018). *contextual: Simulating Contextual Multi-Armed Bandit Problems in R*. Jheronimus Academy of Data Science.

Affiliation:

Robin van Emden
Jheronimus Academy of Data Science
Den Bosch, the Netherlands
E-mail: robin@pwy.nl
URL: pavlov.tech

Eric O. Postma
Tilburg University
Communication and Information Sciences
Tilburg, the Netherlands
E-mail: e.o.postma@tilburguniversity.edu

Maurits C. Kaptein
Tilburg University
Statistics and Research Methods
Tilburg, the Netherlands

E-mail: m.c.kaptein@uvt.nl
URL: www.mauritskaptein.com