Collective Knowledge Applied to March Madness Experiment

Patrick Hesse

***Abstract*—If the essence of scientific discovery could be reduced to a single term it would be reproducibility. Experiments hold merit in the scientific community only when their conclusions can be recreated by peers. The current standard for peer-reviewed submission does not carry the degree of trust that weve come to expect from other scientific disciplines. Our work will incorporate transparency into this process through a series of examples. We have created a series of reproducible workflows with a varying level of complexity aimed at guiding members of the computing community through a series of best practices. Standardizing how computational experiments are shared with the broader scientific community will bring the body of computer science computational research closer to the ideal cherished throughout the broader scientific community; namely trust.**

CONTENTS

LIST OF FIGURES

I. Collective Knowledge

Several tools have emerged to address reproducibility, most notably Docker [1] and Popper. Docker and other virtual environment software have gained popularity due to their ability to take a snapshot of the environment where a software experiment was conducted. Unfortunately, environment snapshots limit researchers' need to validate and build upon previous experiments [2]. Collective Knowledge (CK) is a framework that provides a standard application program interface (API) that enables researchers to share their projects and artifacts in a common format [3]. Researchers using CK can share their entire workflow or components, such as source code and data sets, through repositories like GitHub and Bitbucket [2]. The software service abstracts away access to hardware allowing another researcher to reproduce an experiment under similar conditions [2]. If a researcher is unable to reproduce an experiment due to an incomplete description of software dependencies, they can debug the workflow and share the fixed repository with the Collective Knowledge community. CK's long term goal is to enable a collaborative, reproducible, and sustainable research environment based on DevOps principles [3].

A. Features

Entities, repositories, actions, and modules form the basic vocabulary of collective CK [2]. Entries refer to individual components of a workflow, like source code, data sets, and scripts [8]. CK facilitates sharing and organization by assigning unique identifiers (UIDs) to every entry within the project [8]. Each entry is stored in a sperate directory with its information stored in a JavaScript Object Notation (JSON) file located in a subdirectory entitled .cm [3]. Modules are a specific type of entry that implements the functionality of CK. Modules act as a collection of entries and the actions that operate on them [8]. This creates a two-level directory structure where the top-level directories represent CK modules [8]. The second-level directory store program source code, datasets, and experiments. Actions are CK functionalities offered by modules that operate on lower level entries [8].

B. Code Dependency

CK reduces code dependency by automatically detecting and rebuilding the environment of a workflow and installing missing software packages [4]. All software, data, and models are represented by packages serialized by automatically generated UID, semantic tags, and information about versioning and supported platforms [4]. By cataloging this information, CK uses the JSON API to automatically detect already installed software and install missing software and other source-specific packages automatically [6].

C. Code Rot

Members of the Collective knowledge community who download a solution and have trouble running the experiment due to missing environment specification can debug the workflow and post a corrected version to the CK database [8]. This feature reduces the likelihood of out-of-date repositories making code rot less likely over time.

D. Demonstration

The first step in creating and running a CK repository is downloading the CK source code. This can be done on Linux, Windows, or MacOS [5]. For a Linux system `pip install ck` will download and install CK. After installing CK, `ck pull repo` will download a CK repository.

If the repository has a compile action the following command will trigger the compilation of the source code and CKs automated software dependency detection. At t his point, extra plugins, modules, and packages can be added to the existing project. Once the source code has been compiled `ck run [module_name]: [entity(optional)]` will run the experiment.

E. Collective Knowledge Drawbacks

The creators of CodeReef chose to use the open Collective Knowledge format to share components and workflows because of its continued use in academia and industrial projects. However, CodeReef addresses two major limitations of Collective Knowledge when applied to machine learning workflow [7]. CK technology is a distributed system without a centralized location for components. This makes keeping track of all community contributions difficult [7]. Additionally, the distributed nature of CK makes adding new components, assembling workflows, and automatic testing across different platforms difficult [7]. CK lacks repository and entity versioning, making it challenging to maintain stable workflows. A bug in one CK component can break dependent workflows in adjacent project [7]. Improving on these allows issues allows CodeReef to tackle code dependency and code rot in a similar way to Collective Knowledge while lowering the barrier of entry through a simplification of the software service.a

F. Display results

After the compile and run commmands are run, a post-process script is executed building the table of results that follows. The table shows the top 10 teams as predicted by the program compared to the actual top 10 teams of the 2019 season.

II. Concluding Remarks

The problem of reproducibility in the computer science community can be broken down into two sub-problems. The first is a need for the technical tools necessary for replicating experiments. Without these tools, there is no guarantee that experiments can be trusted in the future. The second is a lack of incentives for creating replicable work.

Researchers face significant barriers to entry in learning new tools. They lack incentives to cultivate an understanding of tools outside of those needed to conduct their experiments. Without being easy to use and adapt to existing workflows, no reproducible solution will be successful [1]. To gain

TABLE I
CK SIMULATED RESULTS

2019 Results	Home-Field Advantage 0	Home-Field Advantage 4	Home-Field Advantage 8
Virginia	Gonzaga	Gonzaga	Gonzaga
Texas Tech	Virginia	Duke	Duke
Auburn	Duke	Virginia	Virginia
Michigan St	North Carolina	North Carolina	North Carolina
Purdue	Texas Tech	Texas Tech	Texas Tech
Duke	Michigan St	Michigan St	Michigan St
Gonzaga	Houston	Kentucky	Kentucky
Kentucky	Kentucky	Houston	Houston
Oregon	Tennessee	Tennessee	Tennessee
Tennessee	Youngstown St	Michigan	Michigan

widespread adoption, reproducible solutions must make it easier for researchers to perform tasks [1]. Unfortunately, tools that provide more functionality have increased in complexity. To combat this, resources must exist to educate researchers about the tools available to them. Ultimately, Researchers have options, at times more than they have time to invest in. A central location for cataloging these tools and demonstrating how they can be applied to existing workflows seamlessly is crucial for the adoption of reproducibility in the computational science community.

REFERENCES

[1] Carl Boettiger. An introduction to docker for reproducible research, with examples from the R environment. *CoRR*, abs/1410.0846, 2014.

[2] G. Fursin, A. Lokhmotov, and E. Plowman. Collective knowledge: Towards r d sustainability. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 864–869, 2016.

[3] Grigori Fursin. Collective knowledge framework (ck). 2017.

[4] Grigori Fursin. Ck features. 2018.

[5] Grigori Fursin. Demonstrating portable and customizable ck benchmarking workflows. 2019.

[6] Grigori Fursin. Portable workflows. 2019.

[7] Grigori Fursin, Herve Guillou, and Nicolas Essayan. Codereef: an open platform for portable mlops, reusable automation actions and reproducible benchmarking, 2020.

[8] Michel Steuwer. Ck: Collective knowledge. 2017.