Reproducible Computing and Reporting in a Complex Software Environment

E. J. Harner[a], C. Grant[b], and M. Lilback[b]
[a] *West Virginia University,* [b] *Rc²ai*

Statistical projects consisting of code-based text, data analyses, visualizations, and simulations can be "containerized," creating a transparent reproduction of the computational workflow that produced the results. With the advent of container platforms such as Docker and their integration with cloud technologies and standalone operating systems, entire environments can be archived, providing later access to the exact data, methods, platforms, and configurations used to generate the computations and reports. Generally, all processes should be automated by Dockerfiles, makefiles, and shell scripts along with Git to ensure reproducibility.

This talk extends the single container model to multiple containers and Docker applications, which are needed for complex software environments. The base image, 'rcompute,' extends Rocker's 'verse' based on Ubuntu (https://github.com/rocker-org/rocker-versioned2) by adding components required for big data and streaming data computations, including drivers for the Postgres database and Spark, supporting R packages, and various Linux libraries and applications. This base container supports Git version control and connectivity to GitHub and it allows R Markdown documents to be created and edited by RStudio Server or Vim. By itself, 'rcompute' supports single node execution for R, Python, Spark and database access, which is sufficient for reproducible reports, but not at scale.

The second image, 'rpsql', provides Postgres database services, the most powerful open-source database available. Although 'rpsql' can be run as a standalone container, it is generally used in conjunction with 'rcompute'. As such, a Docker application was constructed based on both 'rcompute' and 'rpsql' using 'docker-compose.' Doctor compose specifies dependancies among the images and ports so that the running containers can work seamlessly together and can communicate.

The third component 'rspark' is a Docker application comprising a small standalone Spark cluster, composed of a master and worker images. The principal interface between 'rcompute' and 'rspark' is 'sparklyr,' which provides a 'dplyr' interface to Spark. Typically, the 'rcompute' Docker application, including 'rpsql', is run on an edge node and the 'rspark' Docker application is run in the cloud, e.g., AWS. Future work will extend 'rspark' to a Kubernetes (k8s) Spark cluster for scaling out.

The images underlying the Doctor containers discussed here can be frozen at any time on Docker Hub and thus reproducible computing and reporting is possible even at scale. This talk and the LaTeX verion of the abstract, together with links to the repositories for the 'rcompute' and 'rspark' Docker applications, can be found here:
https://github.com/jharner/DSSV2020rspark

Keywords: Reproducible report, Git, Docker, R markdown, Postgres, Spark