`slapnap`: Super LeArner Prediction of NAb Panels

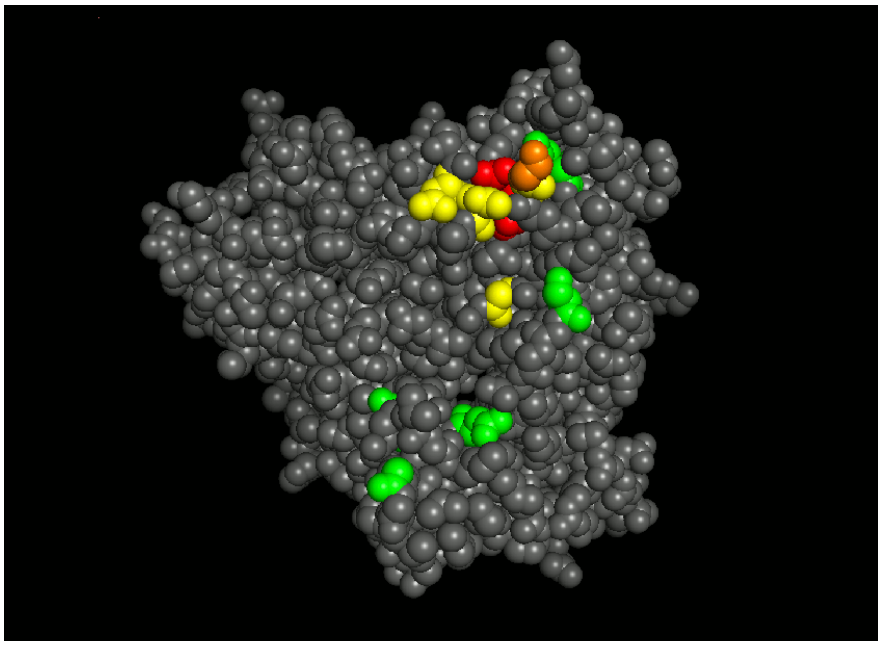
David Benkeser, Brian D. Williamson, Craig A. Magaret, Bhavesh R. Borate, Peter B. Gilbert

May 27, 2020

Contents

Welcome

The `slapnap` container is a tool for using the Compile, Analyze and Tally NAb Panels (CATNAP) database to develop predictive models of HIV-1 neutralization sensitivity to one or several broadly neutralizing antibodies (bNAbs).



Crystal structure of HIV-1 gp120 glycoprotein. Highlighted residues indicating sites most-predictive of VRC01 neutralization resistance. [@magaret2019prediction]

In its simplest form, `slapnap` can be used simply to access and format data from CATNAP in a way that is usable for machine learning analysis. However, the tool also offers fully automated and customizable machine learning analyses based on up to five different neutralization endpoints, complete with automated report generation to summarize results and identify the most predictive features.

This document serves as the user manual for the `slapnap` container. Here, we describe everything needed to utilize the `slapnap` container and understand its output. The documentation is organized into the following sections:

- Section 1 provides a brief overview of Docker, including information on installing Docker and downloading the `slapnap` container.
- Section 2 provides a brief overview of the CATNAP database and the specifics of how the data were accessed to build the `slapnap` container.
- Section 3 provides a detailed description of how to make calls to the slapnap repository, including descriptions of all options that are available.
- Section 4 includes several example calls to the `slapnap` container and descriptions of their output.
- Section 6 provides a description of the data set created in the `slapnap` container.
- Section 7 provides an overview of the methodology that is used in within the `slapnap` analysis.

If you have any issues or questions about using `slapnap`, please file an issue on GitHub.

1 Docker

Docker is a free platform for building containers. Containers are standard units of software that package code and all its dependencies, so that the code can be executed reliably irrespective of computing environment. The `slapnap` tool relies on machine learning implemented in the `R` language and relies on several packages. Achieving full reproducibility for such analyses is challenging in that it requires synchronization across the specific version of `R` and dependent packages. In other words, two users running two versions of `R` or two versions of the same `R` package may arrive at different output when running the same code. Containerization ensures that this does not happen. Any two runs of `slapnap` with the same input options will yield the same output every time.

Installing Docker is necessary for running the `slapnap` tool. While it is not necessary for execution of the `slapnap` container, readers interested in learning more about Docker should consult the Docker documentation for information about getting started using Docker.

Once Docker has been installed on your local computer, you can download `slapnap` using the following command.

```
docker pull slapnap/slapnap
```

This command pulls the image from DockerHub. Once the image has been downloaded, we are ready to learn about how to execute `slapnap` jobs. The next section contains information on the source data used by `slapnap`. Users familiar with the CATNAP data may wish to skip directly to Section 3.1.

2 CATNAP Database

The CATNAP database is a web server hosted by Los Alamos National Laboratory [Yoon et al., 2015]. The database integrates antibody neutralization and HIV-1 sequence data from published studies. Neutralization is measured in terms of half maximal inhibitory concentration (IC_{50}) and 80% inhibitory concentration (IC_{80}). These measures of neutralization against HIV envelope pseudoviruses are available for many broadly neutralizing antibodies (bNAbs) and for some combination bNAbs. Also available on each pseudovirus are amino acid (AA) sequence features for the gp160 protein. These are detailed in Section 6.

During each build of the `slapnap` container, all raw data are downloaded from CATNAP. At run time, the relevant data are selected and processed into a format that is amenable for predictive machine learning analyses. The CATNAP data are updated periodically. To check the date the raw data were pulled from CATNAP to `slapnap`, you can check the date of the `latest` build here.

3 Running the `slapnap` container

To run the `slapnap` container, we make use of the `docker run` command. Note that administrator (`sudo`) privileges are needed to execute this command.

There are several options that are necessary to include in this command to control the behavior of `slapnap`. These are discussed in separate subsections below.

3.1 `slapnap` options

The user has control over many aspects of `slapnap`'s behavior. These options are passed in using the `-e` option[1]. Semi-colon separated strings are used to set options. For example, to provide input for the option `option_name`, we would used `-e option_name="a;semi-colon;separated;string"`. Note that there are no spaces between the option name and its value and no spaces after semi-colons in the separated list. See Section 4 for full syntax.

Each description below lists the default value that is assumed if the option is not specified. Note that many of the default values are chosen simply so that na{"i}ve calls to `slapnap` compile quickly. Proper values should be determined based on scientific context.

-e options for `slapnap`

- **`nab`**: A semicolon-separated list of bNAbs (default = `"VRC01"`). A list of possible bNAbs can be found here. If multiple bNAbs are listed, it is assumed that the analysis should be of estimated `outcomes` for

[1]This sets an environment variable in the container environment. These variables are accessed by the various `R` and `bash` scripts in the container to dictate how the container executes code.

a combination of bNAbs (see Section 7.1 for details on how estimated outcomes for multiple bNAbs are computed).

- `outcomes`: A semicolon-separated string of outcomes to include in the analysis. Possible values are `"ic50"` (included in default), `"ic80"`, `"iip"`, `"sens"` (included in default), `"estsens"`, `"multsens"`. If only a single `nab` is specified, use `sens` to include a dichotomous endpoint. If multiple `nab`s are specified, use `estsens` and/or `multsens`. For detailed definitions of outcomes see Section 7.1).
- `sens_thresh` A numeric value defining the IC_{50} threshold for defining a sensitive versus resistant pseudovirus (default = 1). The dichotomous sensitivity/resistant `outcome`s are defined as the indicator that (estimated) IC_{50} is greater than or equal to `sens_thresh`.
- `multsens_nab` A numeric value used for defining whether a pseudovirus is resistant to a multi-nAb cocktail. The dichotomous outcome `multsens` is defined as the indicator that a virus has (estimated) IC_{50} greater than `sens_thresh` for at least `multsens_nab` nAbs.
- `learners`: A semicolon-separated string of machine learning algorithms to include in the analysis. Possible values include `"rf"` (random forest, default), `"xgboost"` (eXtreme gradient boosting), and `"lasso"` (elastic net). If more than one algorithm is included, then it is assumed that a cross-validated-based ensemble (i.e., a super learner) is desired (see Section 7.2).
- `cvtune`: A boolean string (i.e., either `"TRUE"` or `"FALSE"` [default]) indicating whether the `learners` should be tuned using cross-validation and a small grid search. Defaults to `"FALSE"`. If multiple `learners` are specified, then the super learner ensemble includes three versions of each of the requested `learners` with different tuning parameters.
- `cvperf`: A boolean string (i.e., either `"TRUE"` or `"FALSE"` [default]) indicating whether the `learners` performance should be evaluated using cross-validation. If `cvtune="TRUE"` or `learners` includes multiple algorithms, then nested cross-validation is used to evaluate the performance of the cross-validation-selected best value of tuning parameters for the specified algorithm or the super learner, respectively.
- `nfolds`: A numeric string indicating the number of folds to use in cross-validation procedures (default = `"2"`).
- `importance_grp`: A semicolon-separated string indicating which group-level variable importance measures should be computed. Possible values are none `""` (default), marginal `"marg"`, conditional `"cond"`. See Section 7.3.1 for details on these measures.
- `importance_ind`: A semicolon-separated string indicating which individual-level variable importance measures should be computed. Possible values are none `""` (default), learner-level `"pred"`, marginal `"marg"` and conditional `"cond"`. The latter two take significant computation time to compute.
- `report_name`: A string indicating the desired name of the output report (default = `report_[_-separated list of nabs]_[date].html`).
- `return`: A semicolon-separated string of the desired output. Possible values are `"report"` (default), `"learner"` for the trained algorithm, `"data"` for the analysis dataset, `"figures"` for all figures from the report, and `"vimp"` for variable importance objects.
- `view_port`: A boolean string indicating whether the compiled report should be made viewable on `localhost` (default `"FALSE"`). If `"TRUE"` then `-p` option should be used in the `docker run` command to identify the port. See example in Section **??** for details.

3.2 Mounting a local directory

At the end of a `slapnap` run, user-specified output will be saved (see option `return` in Section 3.1). To retrieve these files from inside the container, we can *mount* a local directory to an output directory (`/home/output/`) in the container using the `-v` option. That is, all files in the mounted local directory will be visible to programs running inside the container and any items saved to the output directory in the container (file path in the container `/home/output/`) will be available in the mounted directory.

Suppose `/path/to/local/dir` is the file path on a local computer in which we wish to save the output files from a `slapnap` run. A `docker run` of `slapnap` would include the option `-v /path/to/local/dir:/home/output`. After a run completes, the requested output should be viewable in `/path/to/local/dir`. See Section 4 for full syntax.

3.3 Viewing report in web browser

An alternative option to mounting local directories for viewing and downloading the report is to set the `view_port` option to `"TRUE"` and open a port to the container via the `-p` option in the `docker run` statement. In this case, rather than exiting upon completion of the analysis, the container will continuing to run and broadcast the compiled report to `localhost` at the specified port (see examples below). The report can be downloaded from the web browser directly in this way.

3.4 Interactive sessions

To simply enter the container and poke around, use an interactive session by including `-it` and overriding the container's entry-point.

```
docker run -it slapnap/slapnap /bin/bash
```

This will enter you into the container in a bash terminal. This may be useful for exploring the file structure, examining versions of `R` packages that are included in the container, etc.

4 Examples

4.1 Basic call to `slapnap`

A call to `slapnap` with all default options can be run using the following command.

```
docker run -v /path/to/local/dir:/home/output slapnap/slapnap
```

Note that this call mounts the local directory `path/to/local/dir` to receive output from the container (Section 3.2).

When this command is executed, messages will print to indicate the progress of the container. The first message will report the name of the log file, which will appear in `/path/to/local/dir`. The container will then compile an analytic data set from the CATNAP database for the default antibody (VRC01), train the default learner (random forest [Breiman, 2001]) for the default outcomes (`ic50` and `sens`), evaluate its performance using two-fold (default for `nfolds`) cross-validation, and compile a report detailing the results, and place the compiled report in `path/to/local/dir`.

4.2 Viewing results in a web browser {sec:webbrowse}

To have the results viewable in a web browser execute the following command (note the use of `\` to break the `bash` command over multiple lines).

```
docker run -v /path/to/local/dir:/home/output \
           -e view_port="TRUE" -p 80:80 \
           slapnap/slapnap
```

This command opens port 80 on the container. Once the report has been compiled, the container will not close down automatically. Instead it will continue to run, broadcasting the report to port 80. Open a web browser on your computer and navigate to `localhost:80` and you should see the compiled report. Many web browsers should allow downloading of the report (e.g., by right-clicking and selecting save `Save As...`).

The container will continue to run until you tell it to `stop`. To do that, retrieve the container ID by executing `docker container ps`. Copy the ID of the running container (say `MY_CONTAINER_ID`) and then execute `docker stop MY_CONTAINER_ID` to shut down the container.

Note that in the above command, we have still mounted a local directory, which may be considered best practice in case other output besides the report is desired to be returned.

4.3 Super learning

If multiple `learners` are specified, then a super learner ensemble [van der Laan et al., 2007] is constructed. In the following command, we construct an ensemble based on a random forest [Breiman, 2001] and elastic net [Zou and Hastie, 2005]. Note that the execution time for super learners can be considerably greater than for single `learners` because of the extra layer of cross-validation needed to construct the ensemble.

```
docker run -v /path/to/local/dir:/home/output \
           -e learners="rf;lasso" \
           slapnap/slapnap
```

4.4 Train an algorithm

The previous commands train learners and evaluate their performance using cross-validation. However, at times we may wish only to use `slapnap` to train a particular algorithm, while avoiding the additional computational time associated with evaluating its cross-validated performance and compiling a report. We show an example of this below using sensitivity as the outcome.

```
docker run -v /path/to/local/dir:/home/output \
           -e learners="rf" \
           -e return="learner" \
           -e cvperf="FALSE" \
           -e outcome="sens" \
           slapnap/slapnap
```

After completion of this run, `learner_sens.rds` will appear in `/path/to/local/dir` that contains an `R` object of class `ranger` (the package used by `slapnap` to fit random forests).

4.5 Pull and clean data

The `slapnap` container can also be used to return cleaned CATNAP data suitable for analyses not supported by the `slapnap` pipeline. In this case, the container avoids training machine learning algorithms and report generation, returning a data set and associated documentation. In the following call, `return` only includes `"data"`; thus, options pertaining to the machine learning portions of `slapnap` are essentially ignored by `slapnap`. The inputted `outcomes` are also irrelevant, as all `outcomes` are included in the resultant data set.

```
docker run -v /path/to/local/dir:/home/output \
           -e return="data" \
           slapnap/slapnap
```

5 Report details

6 Data details

7 Method details

7.1 Outcome definitions

7.2 Super learner details

7.3 Variable importance details

If `importance_grp` or `importance_ind` is specified, then variable importance estimates are computed based on the fitted `learners`. Both biological and prediction importance can be obtained; we discuss each in the following two sections.

7.3.1 Biological importance

Biological importance may be obtained by specifying `importance_grp`, `importance_ind`, or both. We provide two types of biological importance: marginal and conditional, accessed by passing `"marg"` and `"cond"`, respectively, to one of the importance variables. Both types of biological importance are based on the population prediction potential of features [Williamson et al., 2020]. We measure prediction potential using nonparametric R^2 for continuous outcomes (i.e., IC-50, IC-80, or IIP) and using the nonparametric area under the receiver operating characteristic curve (AUC) for binary outcomes (i.e., sensitivity, estimated sensitivity, or multiple sensitivity). In both marginal and conditional importance, we compare the population prediction potential including the feature(s) of interest to the population prediction potential excluding the feature(s) or interest; this provides a measure of the biological importance of the feature(s). The two types of biological importance differ only in the other adjustment variables that we consider: conditional importance compares the prediction potential of all features to the prediction potential of all features excluding the feature(s) of interest, and thus importance must be interpreted conditionally; whereas marginal importance compares the prediction potential of the feature(s) of interest plus geographic confounders to the prediction potential of the geographic confounders alone.

Both marginal and conditional biological importance can be computed for groups of features or individual features. The available feature groups are detailed in 6. Execution time may increase dramatically when biological importance is requested: a separate `learner` (or super learner ensemble) must be trained for each feature group (or individual feature) of interest. Marginal importance tends to be computed more quickly than conditional importance, but both types of importance provide useful information about the population of interest and the underlying biology.

If feature importance is requested, then point estimates, confidence intervals, and p-values (for a test of the null hypothesis that the biological importance is equal to zero) will be computed and displayed for each feature or group of features of interest.

In the following command, we request marginal biological importance for the feature groups defined in 6. We do not specify a super learner ensemble to reduce computation time; however, in most problems we recommend an ensemble to protect against model misspecification.

```
docker run -v /path/to/local/dir:/home/output \
           -e importance_grp="marg" \
           slapnap/slapnap
```

The raw `R` objects (saved as `.rds` files) containing the point estimates, confidence intervals, and p-values for biological importance can be saved by passing `"vimp"` to `return`.

7.3.2 Predictive importance

In addition to the biological importance, it may be of interest to understand how the fitted `learner` depends on the measured features. Learner-level predictive importance may be obtained by passing `"pred"` to the variable `importance_ind`. If a single `learner` is fit, then the predictive importance is the `R` default for that type of learner: for the elastic net, importance is defined using the absolute value of the estimated regression coefficient; for random forests, importance is defined using the Gini index (binary outcomes) or the variance of the outcome (continuous outcomes); for eXtreme gradient boosting [Chen and Guestrin, 2016], importance is defined using the fractional contribution of total gain of the feature's splits. If a super learner ensemble is fit, then the best-fitting algorithm in the ensemble is used to compute the predictive importance.

In the following command, we request predictive importance for a simple scenario. Predictive importance is displayed for the top 20 features.

```
docker run -v /path/to/local/dir:/home/output \
           -e importance_ind="pred" \
           slapnap/slapnap
```

8 References

References

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

Mark J van der Laan, Eric C Polley, and Alan E Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

Brian D Williamson, Peter B Gilbert, Noah R Simon, and Marco Carone. A unified approach for inference on algorithm-agnostic variable importance. *arXiv preprint arXiv:2004.03683*, 2020.

Hyejin Yoon, Jennifer Macke, Anthony P West Jr, Brian Foley, Pamela J Bjorkman, Bette Korber, and Karina Yusim. CATNAP: a tool to compile, analyze and tally neutralizing antibody panels. *Nucleic Acids Research*, 43(W1):W213–W219, 2015. doi: 10.1093/nar/gkv404.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.