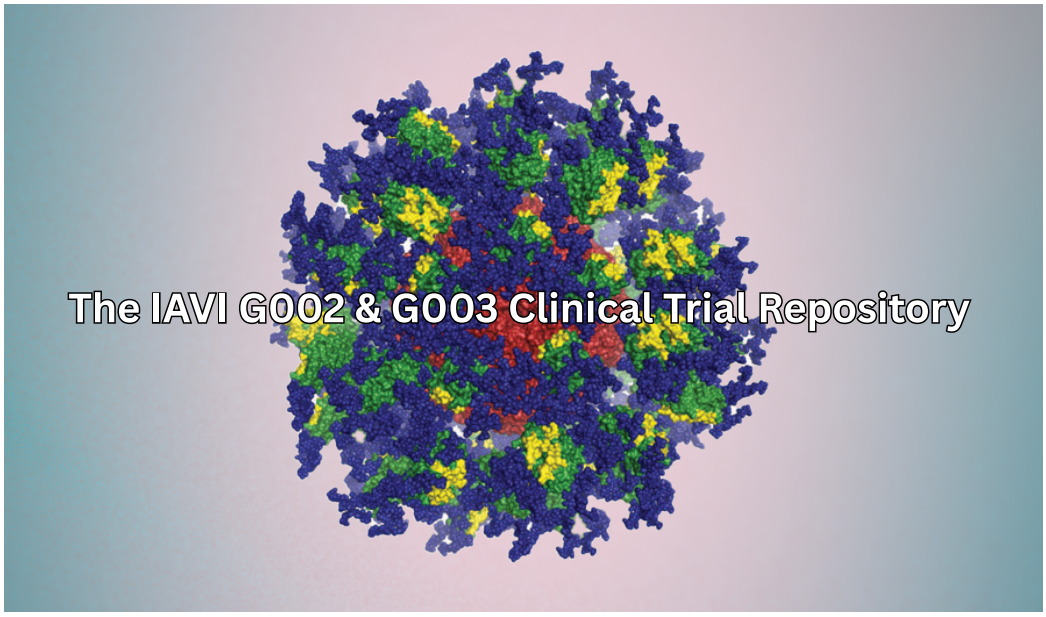


Python 3.10 code style black pre-commit enabled

## G00x - Generalizable Germline-Targeting Clinical Trial Pipeline



# About / Summary

This repository serves two major purposes.

First, this repository contains the data for the publication "Vaccination with mRNA-encoded nanoparticles drives early maturation of HIV bnAb precursors in humans", Willis, Prabhakaran, Muthui, Naidoo, Sincomb, Wu, Cottrell, Landais, deCamp, et al., Science, 2025. This publication reports the safety and immunogenicity data from the clinical trials IAVI G002 (in the United States) and IAVI G003 (in Rwanda and South Africa). The data in the repository includes:

- Antigen- and epitope-specific memory B cell frequencies (FACS data)
- 10x-based sequencing data from epitope-specific memory B cells
- Frequencies of VRC01-class bnAb precursor memory B cells
- SPR $K_D$, $k_{on}$, and $k_{off}$ values for VRC01-class and non-VRC01-class competitor antibodies (derived from sorted memory B cells) binding to various antigens

- Code for generating main text figures 2 through 7.
- Code for generating the spreadsheet Data S10. “Source data”.

Second, the repository presents the G00x pipeline designed to facilitate the analysis of germline-targeting vaccine clinical trials. It is an all-in-one pipeline and analysis platform that parses, validates, calculates B cell frequencies, runs cellranger and combines all analyses into a plottable dataframe. The pipeline supports datasets for both G002 and G003 trials (related to germline targeting to elicit VRC01-class antibodies against HIV) and ensures that all data is processed and validated to maintain the integrity of the clinical trial results. The pipeline is readily modifiable to enable handling of other types of germline-targeting trials, with other types of analyses, and can also be modified to enable data storage/data flow pathways different from that used in G002 and G003.

---

# Data Access

If you don't want to run the pipeline, you can access the important data via the following links.

- The annotated, filtered and paired antibody sequences:
  - G002 final_df.feather
  - G003 final_df.feather
- Merged summary file with all frequencies reported in this study:
  - G002 flow_and_sequencing_long_names.csv
  - G003 flow_and_sequencing_long_names.csv

## G002

To run testing, you will need a `g002` directory in the root of the project. You can get this with the sync command:

```
# this will sync the entire contents of the files into g002 directory
# Beware, this file is 1.5TB
aws sync --delete s3 s3://iavig002public/g002/ ./g002'
```

However, we have also setup so you don't need to download the entire directory. You can run the following to get a subset of the data.

```
# get the sorting directory
aws s3 cp --recursive s3://iavig002public/g002/G002/sorting ./g002/G002/sorting

# get the sequencing directory excluding large bcl and fastq files
aws s3 cp --recursive s3://iavig002public/g002/G002/sequencing ./g002/G002/sequencing --exclude *working_directory/*
--exclude *.fastq.gz --exclude *.tif --exclude *.cbcl --exclude *.imf1 --exclude *.filter --exclude *.bin --exclude
*Logs/* --exclude *_stdout --exclude *_stderr --exclude *Autofocus/* --exclude *Intensities/*
```

## G003

To run testing, you will need a `g003` directory in the root of the project. You can get this with the sync command

```
# this will sync the entire contents of the files into g003 directory
# Beware, this file is 1.5TB
aws sync --delete s3 s3://iavig003public/g003/ ./g003'
```

However, we have also setup so you don't need to download the entire directory. You can run the following to get a subset of the data.

```
# get the sorting directory
aws s3 cp --recursive s3://iavig003public/g003/G003/sorting ./g003/G003/sorting

# get the sequencing directory excluding large bcl and fastq files
aws s3 cp --recursive s3://iavig003public/g003/G003/sequencing ./g003/G003/sequencing --exclude *working_directory/*
--exclude *.fastq.gz --exclude *.tif --exclude *.cbcl --exclude *.imf1 --exclude *.filter --exclude *.bin --exclude
*Logs/* --exclude *_stdout --exclude *_stderr --exclude *Autofocus/* --exclude *Intensities/*
```

# Pipeline

## Installation pre-requisites

While not necessary, we highly recommend using the conda open-source package and environment manager. For the purposes of this repository, only a minimal installer for anaconda is necessary (Miniconda).

Miniconda command line installers

## Installation

This installation assumes that `git` and `conda` are in your path.

```
# clone the repository
git clone https://github.com/SchiefLab/G00x.git

# change directory
cd G00x

# this will create a conda environment called g00x and install the package
./install.sh
```

## Validation

The most important part of this clinical trial other than it working, is that everything is validated. That means that the data is validated, the code is validated, and the results are validated. This is a very important part of the process and should not be skipped.

### FACS/Sorting validation

#### G002

These are instructions on how to validate the file structure containing the FACS/Sorting data before uploading to box.

```
#Run the validator for flow from the command line
g00x g002 validate flow my_path/to/box/G002/


# If you used the AWS sync command above, you can use the command:
g00x g002 validate flow ./g002/G002/sorting/G002
```

Your folder structure on box should look like this.

```
# if you used the above commands to sync the data. The folder structure will look like this

g002/G002/sorting
└── G002
    ├── Prescreens
    │   ├── Prescreen_RunDate220825_UploadDate221021
    │   ├── Prescreen_RunDate220826_UploadDate221021
    ....
    └── Sorts
        ├── Sort_RunDate220927_UploadDate221013
        ├── Sort_RunDate220928_UploadDate221014
        ├── Sort_RunDate220929_UploadDate221014
        ├── Sort_RunDate220930_UploadDate221014
```

#### G003

```
#Run the validator for flow from the command line
g00x g003 validate flow my_path/to/G003/


# If you used the AWS sync command above, you can use th e
g00x g003 validate flow ./g003/G003/sorting/G003
```

For G003, all the data is available in S3 bucket. The structure should look as below:

```
g003/G003/sorting
└── G003
    ├── Prescreens
    │   ├── Sort_RunDate230807_UploadDate230807
    │   ├── Sort_RunDate230808_UploadDate230808
    ....
    └── Sorts
        ├── Sort_RunDate230807_UploadDate230807
        ├── Sort_RunDate230808_UploadDate230808
        ├── Sort_RunDate230809_UploadDate230809
        ├── Sort_RunDate231003_UploadDate231027
```

Sequencing files validation

**G002**

In order to validate the sequencing files, it must be merged with the flow data. Thus, you will need both Globus and Box access. Validation can be accomplished by validating the merge command

```
g00x g002 sequencing -f /path/to/flow -s /path/to/sequencing -o output_file


# if you have the AWS structure
g00x g002 merge -s ./g002/G002/sequencing/G002 -f ./g002/G002/sorting/G002 -o my_merged_file
```

The sequencing folder structure should be as follows:

```
G002
├── run0002
│   ├── 221006_VH00497_31_AAAVKCLHV
│   └── sample_manifest.csv
├── run0003
│   ├── 221019_VH00497_32_AAANGGVM5
│   └── sample_manifest.csv
└── run0004
    ├── 221101_VL00414_3_AACFYLCM5
    ├── 221103_VL00414_4_AAATJGYM5
    └── sample_manifest.csv
```

**G003**

In G003, all the data are stored in the S3 bucket; thus, you only need access.

```
g00x g003 sequencing -f /path/to/flow -s /path/to/sequencing -o output_file


# if you have the AWS structure
g00x g003 merge -s ./g003/G003/sequencing/G003 -f ./g003/G003/sorting/G003 -o my_merged_file
```

```
G003
├── run0001
│   ├── 230818_NB552490_0059_AHL7GFBGXM
│   └── sequencing_manifest.csv
├── run0002
│   ├── 230821_NB552490_0060_AHL7GGBGXM
│   └── sequencing_manifest.csv
├── run0003
│   ├── 231108_NB552490_0065_AHL7CCBGXM
│   └── sequencing_manifest.csv
```

Using the above command, you will generate a file called `my_merged_file.csv` which will have the following fields.

```
ptid
group
weeks
visit_id
probe_set
sample_type
run_date
sort_pool
hashtag
run_dir_path
pool_number
sorted_date
vdj_sequencing_replicate
cso_sequencing_replicate
vdj_lirary_replicate
cso_library_replicate
bio_replicate
vdj_index
```

```
feature_index
vdj_run_id
cso_run_id
```

No data fields should have NA or empty values. If that happens, that means there are non-merged files.

## FACS/Flow

### G002

```
# run in one command
g00x g002 flow /path/to/box/G002/ -o path/to/flow_output
```

### G003

```
# run in one command
g00x g003 flow /path/to/G003/ -o path/to/flow_output
```

This will output two dataframes—one in CSV and the other in feather. The dataframe looks like the following.

| | run_purpose | run_date | sort_id | ptid | group | weeks | visit_id | probe_set | sample_type | sort_software_dv | sort_file_type | samp |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | PreS | 2022-08-26 | S6C | G002611 | 4 | 8 | V600 | Cg28v2 | PBMC | DV | Summary | T1 |
| 1 | PreS | 2022-08-26 | S6C | G002710 | 4 | -5 | V091 | Cg28v2 | PBMC | DV | Summary | T1 |
| 2 | PreS | 2022-08-26 | S6C | G002611 | 4 | 4 | V160 | Cg28v2 | PBMC | DV | Summary | T1 |
| 3 | PreS | 2022-08-26 | S6C | G002710 | 4 | 8 | V600 | Cg28v2 | PBMC | DV | Summary | T1 |

Each row is a single flow measurement. For example, the count of how many B cells were in the sort. There are many meta data columns as well. A single donor, time point and flow experiment is repeated for each measurement we are after. This makes it very easy to extract and plot data.

These are the fields from the previous dataframe.

| fields | description |
|---|---|
| run_purpose | Sort or Preclinical (PreS) |
| run_date | The date the sort was completed |
| sort_id | internal sort id, e.g. S6 |
| pubID | the patient id, e.g. G002611 |
| group | the vaccine group, 1,2,3,4 |
| weeks | the week from the sample |
| visit_id | the visit id corresponding with the group |
| probe_set | the group this flow sample was sorted with |
| sample_type | PBMC or FNA |
| sort_software_dv | Diva software, DV |
| sort_file_type | Only Summary at this point. But could be FCS |
| sample_tube | the labeling of the sample tube |
| file_subset | if the sorter is started or stopped, this can divide the file, e.g a,b,c.. |
| extension | the file extension |
| file_path | the path of the file |
| gate | the gate name, e.g P1 |
| gate_parent | The parent gate of the gate |
| easy_name | an easier more programmatic name for the gate measurement, e.g antigenic_count |
| verbose_name | A more verbose name for labeling |
| value_type | if the gate a frequency or a count |

| fields | description |
|---|---|
| value | what is the value of the gate or frequency |
| hashtag | if this sample has been sorted, what is the hashtag |
| sort_pool | the sorting pool, P01 - P10 |

### BCR sequence analysis

G002

```
g00x g002 pipeline demultiplex -f ./g002/G002/sorting/G002/ -s ./g002/G002/sequencing/G002/ -o
./g002/G002/output/demultiplex

g00x g002 pipeline vdj -d ./g002/G002/output/demultiplex.feather -o ./g002/G002/output/vdj

g00x g002 pipeline cso -d ./g002/G002/output/demultiplex.feather -o ./g002/G002/output/cso

# Merge VDJ and CSO dataframes, run the output through SADIE for
g00x g002 pipeline airr -k 3 -c ./g002/G002/output/cso.feather -v ./g002/G002/output/vdj.feather -o
./g002/G002/output/final_df

# Output merged.feather not used to generate figures, but is a sanity check.
g00x g002 validate merge -s ./g002/G002/sequencing/G002/ -f ./g002/G002/sorting/G002 -o ./g002/G002/output/merged

# Output flow_and_sequencing.feather used to generate figures; will contain counts, frequencies, and metadata from
flow and sequencing data.
g00x g002 analysis report -s ./g002/G002/output/final_df.feather -f ./g002/G002/output/flow_output.feather -o
./g002/G002/output/flow_and_sequencing
```

G003

```
g00x g003 pipeline demultiplex -f ./g003/G003/sorting/G003/ -s ./g003/G003/sequencing/G003/ -o
./g003/G003/output/demultiplex

g00x g003 pipeline vdj -d ./g003/G003/output/demultiplex.feather -o ./g003/G003/output/vdj

g00x g003 pipeline cso -d ./g003/G003/output/demultiplex.feather -o ./g003/G003/output/cso

# Merge VDJ and CSO dataframes, run the output through SADIE for
g00x g003 pipeline airr -k 3 -c ./g003/G003/output/cso.feather -v ./g003/G003/output/vdj.feather -o
./g003/G003/output/final_df

# Output merged.feather not used to generate figures, but is a sanity check.
g00x g003 validate merge -s ./g003/G003/sequencing/G003/ -f ./g003/G003/sorting/G003 -o ./g003/G003/output/merged

# Output flow_and_sequencing.feather used to generate figures; will contain counts, frequencies, and metadata from
flow and sequencing data.
g00x g003 analysis report -s ./g003/G003/output/final_df.feather -f ./g003/G003/output/flow_output.feather -o
./g003/G003/output/flow_and_sequencing
```

### Development

#### Testing

If you'd like to help develop. Please use pre-commit before commiting any files.

```
# runs pre-commit for syntax,formatting and static type checking
poetry run pre-commit run --all-files
```

Now you can run the tests via poetry

```
poetry run pytest -sv --log-cli-level DEBUG tests
```

## Figures and Tables

### Main Figures

Main figures as they are displayed in the G002 and G003 paper. Figure 1 is not included as it is manually generated.

```
g00x plots fig2
g00x plots fig3
g00x plots fig4
g00x plots fig5
g00x plots fig6
g00x plots fig7
```

### Supplementary Figures

Supplementary figures are in no particular order. They are generated as needed.

```
g00x plots suppfigures --all #TODO: Update later
```

## Issues

Please submit any issues to the issues page and we are happy to help.

## License