COSGAP

Release 1.9.0dev

Richard Zetterberg, John Shorter, Espen Hagen, Bayram Cevdet A

Apr 15, 2024

TABLE OF CONTENTS

This is the online documentation for the COSGAP project, hosted at GitHub.com/CoMorMent/containers.

CHAPTER

ONE

INTRODUCTION

1.1 COSGAP: COntainerized Statistical Genetics Analysis Pipelines

1.1.1 Documentation

The main documentation for COSGAP is hosted at cosgap.rtfd.io

1.1.2 Project status

1.1.3 Information

The goal of this github repository (https://github.com/comorment/containers) is to distribute software tools for statistical genetics analysis, alongside with their respective reference data and scripts ("analysis pipelines") to facilitate application of these tools. The scope of this project is currently limited to genome-wide association studies (GWAS) and post-GWAS statistical-genetics analyses, including polygenic scoring (PGS). This project builds on earlier work by Tryggve consortium, with most recent major development done as part of the CoMorMent EU H2020 project (comorment.eu). For more information see our preprint manuscript, this presentation on PGC WWL meeting (Feb 9, 2024), or our online documentation here.

For an overview of available software, see here.

Most of these tools are packaged into singularity containers (https://sylabs.io/singularity/) and shared in the singularity folder of this repository. You can download individual containers using github's `Download` button, or clone the entire repository from command line as described in the *INSTALL.md* file.

Many of the tools require additional reference data provided in the reference folder of this repository. Certain reference data can not be made publicly available, in which case we provide access instructions in a separate GitHub repository: https://github.com/comorment/reference. This repository is private - please approach your contact within CoMorMent project to enable your access.

Description of containers and usage instructions are provided in the docs folder.

More extensive use cases of containers, focusing on real data analysis, are provided in the usecases folder.

The history of changes is available in the *CHANGELOG.md* file.

If you would like to contribute to developing these containers, please see the *CONTRIBUTING* file.

Additional tools are available in separate repositories:

- https://github.com/comorment/ldsc - LD score regression

- https://github.com/comorment/mixer - cross-trait MiXeR analysis
- https://github.com/comorment/popcorn - cross-ancestry genetic correlations
- https://github.com/comorment/magma - MAGMA, LAVA, lava-partitioning tools
- https://github.com/comorment/HDL - High-Definition Likelihood
- https://github.com/comorment/ldpred2_ref - reference files for LDpred2. The tool itself is included in `r.sif` (more info).

1.1.4 Cite

If you use the software provided here, please cite our Zenodo.org code deposit (change version accordingly):

```
Oleksandr Frei, Andreas Jangmo, Espen Hagen, bayramakdeniz, ttfiliz, Richard Zetterberg,␣
↪& John Shorter. (2024). comorment/containers: Comorment-Containers-v1.8.1 (v1.8.1).␣
↪Zenodo. https://doi.org/10.5281/zenodo.10782180
```

Bibtex format:

```
@software{oleksandr_frei_2024_10782180,
  author       = {Oleksandr Frei and
                  Andreas Jangmo and
                  Espen Hagen and
                  bayramakdeniz and
                  ttfiliz and
                  Richard Zetterberg and
                  John Shorter},
  title        = {comorment/containers: Comorment-Containers-v1.8.1},
  month        = mar,
  year         = 2024,
  publisher    = {Zenodo},
  version      = {v1.8.1},
  doi          = {10.5281/zenodo.10782180},
  url          = {https://doi.org/10.5281/zenodo.10782180}
}
```

Please also cite our preprint:

```
Akdeniz, B.C., Frei, O., Hagen, E., Filiz, T.T., Karthikeyan, S., Pasman, J.A., Jangmo,␣
↪A., Bergsted, J., Shorter, J.R., Zetterberg, R., Meijsen, J.J., Sønderby, I.E., Buil,␣
↪A., Tesli, M., Lu, Y., Sullivan, P., Andreassen, O.A., & Hovig, E. (2022). COGEDAP: A␣
↪COmprehensive GEnomic Data Analysis Platform. arXiv:2212.14103 [q-bio.GN]. DOI: [10.
↪48550/arXiv.2212.14103](https://doi.org/)
```

Bibtex format:

```
@misc{akdeniz2022cogedap,
      title={COGEDAP: A COmprehensive GEnomic Data Analysis Platform},
      author={Bayram Cevdet Akdeniz and Oleksandr Frei and Espen Hagen and Tahir Tekin␣
↪Filiz and Sandeep Karthikeyan and Joelle Pasman and Andreas Jangmo and Jacob Bergsted␣
↪and John R. Shorter and Richard Zetterberg and Joeri Meijsen and Ida Elken Sonderby␣
↪and Alfonso Buil and Martin Tesli and Yi Lu and Patrick Sullivan and Ole Andreassen␣
↪and Eivind Hovig},
```

(continues on next page)

(continued from previous page)

```
      year={2022},
      eprint={2212.14103},
      archivePrefix={arXiv},
      primaryClass={q-bio.GN}
}
```

Note that this project will soon be renamed "COSGAP", and that the citation info will be updated accordingly.

1.1.5 Installation

See the *INSTALL.md* file for installation instructions.

1.1.6 Legacy

Earlier version (prior to April 2021) of all containers and refrence data was distributed on Google Drive. This is no longer the case, the folder on Google drive is no longer maintained. ALl containers and reference data are released through this repository.

1.1.7 Source files

The source files for configuring and building the container files provided here are found in the docker directory. See the corresponding *README* file therein for details.

1.1.8 Documentation build instructions

The online documentation hosted at cosgap.rtfd.io can be built locally using Sphinx in a conda environment as

```
cd sphinx-docs/source  # documentation source/config directory
conda env create -f environment.yml  # creates environment "sphinx"
conda activate sphinx
make html  # make html-documentation in $PWD/_build/html/
```

The resulting file(s) `$PWD/_build/html/index.html` can be viewed in any web browser.

In order to make a pdf with the documentation, issue

```
make pdflatex
```

and open `$PWD/_build/latex/cosgap.pdf` in a pdf viewer.

CHAPTER

TWO

GETTING STARTED

To get started using a bare-bones container setup, please confer the following pages:

2.1 `hello.sif` container

2.1.1 Description

You may use `hello.sif` container to familirize yourself with Singularity (https://sylabs.io/docs/), and the way it works on your secure HPC environment (TSD, Bianca, Computerome, or similar). This singularity container is indented as a demo. It only contains Plink 1.9 (http://zzz.bwh.harvard.edu/plink/) software.

2.1.2 Getting Started

- Download `hello.sif` from here
- Download `chr21.[bed,bim,fam]` files from here
- Import these files to your secure HPC environment
- Run `singularity exec --no-home hello.sif plink --help`, to validate that you can run singularity. This command is expected to produce the standard plink help message, starting like this:

```
PLINK v1.90b6.18 64-bit (16 Jun 2020)          www.cog-genomics.org/plink/1.9/
(C) 2005-2020 Shaun Purcell, Christopher Chang   GNU General Public License v3
```

2.1.3 Helpful links to singularity documentation

It's good idea to familiraze with basics of the singularity, such as these:

- "singularity shell" options
- Bind paths and mounts.

2.1.4 Installing Docker and Singularity on your local machine

While you're getting up to speed with singularity, it might be reasonable to have it install on your local machine (laptop or desktop), and try out containers locally before importing them to your HPC environment.

To install singularity on Ubuntu follow steps described here: https://sylabs.io/guides/3.7/user-guide/quick_start.html Note that `sudo apt-get` can give only a very old version of singularity, which isn't sufficient. Therefore it's best to build singularity locally. Note that singularity depends on GO, so it must be installed first. If you discovered more speciifc instructions, please submit an issue or pull request to update this documentation.

2.1.5 Mapping your data to singularity containers

There are several ways to give singularity container access to your data. Here are few examples:

1. `singularity exec --home $PWD:/home hello.sif plink --bfile chr21 --freq --out chr21` - this command will map your current folder (`$PWD`) into `/home` folder within container, and set it as active working directory. In this way in your plink command you can refer to the files as if they are in your local folder, i.e. `chr21` without specifying the path. The command will then use plink to calculate allele frequencies, and save the result in current folder.

2. `singularity exec --home $PWD:/home hello.sif plink --bfile /home/chr21 --freq --out /home/chr21` Same as above command, but more explicitly refer to `/home/chr21` files, without relying on it being the active working directory. Here you can also choose to use `--bind` argument instead of `--home`, which allow to map multiple folders if needed (comma-separated).

3. Now, let's assume that instead of downloading `chr21.[bim/bed/fam]` files and `hello.sif` container you've cloned the entire github repo (`git clone git@github.com:comorment/containers.git`), and have transfered it to your HPC environment. Then change your folder to the root of the `containers` repository, and run these commands:

   ```
   mkdir out_dir && singularity exec --bind reference/:/ref:ro,out_dir:/out:rw␣
   ↪singularity/hello.sif plink --bfile /ref/hapgen/chr21 --freq --out /out/chr21
   ```

   Note that input paths are relative to the current folder. Also, we specified `ro` and `rw` access, to have reference data as read-only, but explicitly allow the container to write into `/out` folder (mapped to `out_dir` on the host).

4. Run `singularity shell --home $PWD:/home -B $(pwd)/data:/data hello.sif` to use singularity in an interactive mode. In this mode you can interactively run plink commands. Note that it will consume resources of the machine where you currently run the singulairty comand (i.e., most likely, the login node of your HPC cluster).

2.1.6 Running as SLURM job

- Run singularity container within SLURM job scheduler, by creating a `hello_slurm.sh` file (by adjusting the example below), and running `sbatch hello_slurm.sh`:

  ```
  #!/bin/bash
  #SBATCH --job-name=hello
  #SBATCH --account=p697
  #SBATCH --time=00:10:00
  #SBATCH --cpus-per-task=1
  #SBATCH --mem-per-cpu=8000M
  module load singularity/3.7.1
  singularity exec --no-home hello.sif plink --help
  singularity exec --home $PWD:/home hello.sif plink --bfile chr21 --freq --out chr21
  ```

Please let us know if you face any problems.

2.1.7 TSD-specific instructions

The official documentation for singularity on TSD is available here. Here are more important notes about singularity on TSD:

- `module load singularity/2.6.1` is going to be deprecated soon; instead, there will be a local installation of singularity on each Colossus node
- Singularity might be unavailable on some of the interactive nodes. For example, in `p33` project it is recommended to run singularity on `p33-appn-norment01` node. You may also find it in `p33-submit` nodes.
- You may want to run `module purge`, to make sure you use locally installed singularity. It is good idea to run `which singularity` to validate this.
- Use `singularity --version` to find the version of singularity
- Generally, it is a good idea to add `--no-home` argument to your singularity commands, to make sure that that scripts such as `.bashrc` do not interfere with singularity container. This also applies if you have custom software installed in your home folder. For other options that control isolation of the containers (i.e. `--containall` option) see here.
- If you are a developer, and you would like to generate a singularity container, you may want to do it outside of TSD, and then bring just a `.sif` file to TSD. Also, building singularity containers is much easier by building a Docker container first (using `Dockerfile`), and converting such Docker container to a singularity container.

2.1.8 Software

List of software included in the container:

OS/tool	version	license
ubuntu	20.04	Creative Commons CC-BY-SA version 3.0 UK licence
plink[1]	v1.90b6.18 64-bit (16 Jun 2020)	GPLv3

References

[1] Christopher C Chang, Carson C Chow, Laurent CAM Tellier, Shashaank Vattikuti, Shaun M Purcell, James J Lee, Second-generation PLINK: rising to the challenge of larger and richer datasets, GigaScience, Volume 4, Issue 1, December 2015, s13742–015–0047–8, https://doi.org/10.1186/s13742-015-0047-8

CHAPTER

THREE

INSTALLATION

We recommend to clone this entire repository using `git clone`. However, you need to install the Git LFS extension. This is done by downloading and unpacking the GitLFS package, adding `git-lfs` binary to a folder that is in your `PATH`, and running `git lfs install` command.

```
mkdir ~/bin
export PATH="/home/$USER/bin:$PATH"        # good idea to put this in your ~/.bashrc or ~
↪/.bash_profile
wget https://github.com/git-lfs/git-lfs/releases/download/v2.13.2/git-lfs-linux-amd64-v2.
↪13.2.tar.gz
tar -xzvf git-lfs-linux-amd64-v2.13.2.tar.gz
cp git-lfs /home/$USER/bin
git lfs install
```

Now you're all set to clone this repository (note that adding `--depth 1` to your command as shown below will limit the amount of data transfered from github to your machine):

```
git clone --depth 1 https://github.com/comorment/containers.git
```

At this point you may want to run the following find&grep command to check that all git lfs files were downloaded successfully (i.e. you got an actual content of each file, and not just its git lfs reference). The command searches for and lists all files within $COMORMENT folder which contain a string like `oid sha`, likely indicating that git lfs file hasn't been downloaded. If the following commands doesn't find any files that you're good to go. Otherwise you may want to re-run your `git clone` commands or investigate why the're failing to download the actual file.

```
find $COMORMENT -type f -not -path '*/.*' -exec sh -c 'head -c 100 "{}" | if grep -H
↪"oid sha"; then echo {}; fi ' \; | grep -v "oid sha256"
```

For TSD system, a read-only copy of $COMORMENT containers is maintained at these locations (please read github/README.md file before using these copies):

```
# for p33 project
export COMORMENT=/cluster/projects/p33/github/comorment

# for p697 project
export COMORMENT=/ess/p697/data/durable/s3-api/github/comorment
```

Once you have a clone of this repository on your system, you may proceed with docs/singularity/hello.md example. Take a look at the README file in the docs/singularity folder, as well as detailed use cases in usecases.

To simplify instructions throughout this repository we use certain variables (it's a good idea to include them in your `.bashrc` or similar):

- `$COMORMENT` refers to a folder with `comorment` and `reference` subfolders, containing a clone of the containers and reference repositories from GitHub. Cloning `reference` repository is optional, and it's only needed for internal work within the CoMorMent project - for normal use you may proceed without it.
- `$SIF` refers to `$COMORMENT/containers/singularity` folder, containing singulairty containers (the `.sif` files)
- `SINGULARITY_BIND="$COMORMENT/containers/reference:/REF:ro,$COMORMENT/reference:/REF2:ro"` defines default bindings within container (`/REF`, `/REF2`). If you don't have access to private reference, try out commands without mapping `$COMORMENT/reference:/REF2:ro` - most (if not all) of the exmples don't require private reference data.
- We assume that all containers run with `--home $PWD:/home`, mounting current folder mounted as `/home` within container
- We also recommend using `--contain` argument to better isolate container from the environment in your host machine. If you choose not to mount `--home $PWD:/home`, you may want to add `--no-home` argument.
- You can choose to exclude passing environment variables from the host into the container with the `--cleanenv` option. Read more about it here.

CHAPTER

FOUR

SINGULARITY

4.1 gwas.sif container

4.1.1 Description

The gwas.sif container file has multiple tools related to imputation and GWAS analysis, as summarized in the *Sofware* table below.

Note that some specific tools (e.g. bolt) are added to the path directly from their /tools folder (e.g. /tools/bolt) due to hard-linked dependencies. Either way, all tools can be invoked by their name, as listed above. For example:

```
>singularity exec gwas.sif regenie
              |==============================|
              |      REGENIE v2.0.2.gz       |
              |==============================|

Copyright (c) 2020 Joelle Mbatchou and Jonathan Marchini.
Distributed under the MIT License.
Compiled with Boost Iostream library.
Using Intel MKL with Eigen.

ERROR: You must provide an output prefix using '--out'
For more information, use option '--help' or visit the website: https://rgcgithub.github.
↪io/regenie/
```

4.1.2 Software

List of software included in the container:

OS/tool	version	license
ubuntu	20.04	Creative Commons CC-BY-SA version 3.0 UK licence
bcftools[1]	1.19	MIT/Expat/GPLv3
bedtools[2]	2.31.1	MIT
beagle[34]	22Jul22.46e	GPLv3
bgenix[5]	1.1.7	Boost
bolt[6]	v2.4.1	GPLv3
cat-bgen[7]	same version as bgenix	Boost

continues on next page

Table 1 – continued from previous page

OS/tool	version	license
duohmm[8]	95bd395	MIT
eagle[9]	v2.4.1	GPLv3
edit-bgen[10]	same version as bgenix	Boost
flashpca_x86-64[11]	2.0	GPLv3
gcta64[12]	1.94.1	GPLv3
gctb[13]	2.04.3	MIT
GWAMA[14]	2.2.2	BSD-3-Clause
HTSlib[15]	1.19.1	MIT/Expat/Modified-BSD
king[16]	2.3.2	permissive
ldak[17]	5.2	GPLv3
liftOver[18]	latest	permissive
metal[19]	2020-05-05	-
minimac4[20]	v4.1.6	GPLv3
plink[21]	v1.90b7.2 64-bit (11 Dec 2023)	GPLv3
plink2[22]	v2.00a5.10LM 64-bit Intel (5 Jan 2024)	GPLv3
plink2_avx2[Page 15, 22]	v2.00a5.10LM AVX2 Intel (5 Jan 2024)	GPLv3
PRSice_linux[23]	2.3.5	GPLv3
qctool[24]	2.2.2, revision e5723df2c0c85959	Boost
regenie[25]	v3.4	MIT/Boost
samtools[1]	v1.19.2	MIT/ExpatD
shapeit4.2[26]	v4.2.2	MIT
shapeit5[27] phase_rare	v5.1.1	MIT
shapeit5[Page 15, 27] phase_common	v5.1.1	MIT
shapeit5[Page 15, 27] ligate	v5.1.1	MIT
shapeit5[Page 15, 27] switch	v5.1.1	MIT
shapeit5[Page 15, 27] xcftools	v5.1.1	MIT
simu_linux[28]	v0.9.4	GPLv3
snptest[29]	v2.5.6	permissive
switchError[30]	6e688b1	MIT
vcftools[31]	0.1.17 (git SHA: d511f469e)	GPLv3

4.2 `python3.sif` container

4.2.1 Description

`python3.sif` container runs Python packaged by Conda-forge, and has many useful python modules already installed, including pandas, numpy, scipy, matplotlib, jupyter and few others (see here for full details). Basic usage is very simple:

```
>singularity exec --contain --home $PWD:/home python3.sif python
Python 3.10.6 | packaged by conda-forge | (main, Aug 22 2022, 20:35:26) [GCC 10.4.0] on␣
↪linux
Type "help", "copyright", "credits" or "license" for more information.
```

You may also use jupyter notebook like this:

[1] Petr Danecek, James K Bonfield, Jennifer Liddle, John Marshall, Valeriu Ohan, Martin O Pollard, Andrew Whitwham, Thomas Keane, Shane A McCarthy, Robert M Davies, Heng Li, Twelve years of SAMtools and BCFtools, GigaScience, Volume 10, Issue 2, February 2021, giab008,

```
singularity exec --contain --home $PWD:/home $SIF/python3.sif jupyter notebook --no-
↪browser --port 8890
```

The port is optional, but you may want to specify it if you'd like to run jupyter on a remote server - in which case you need to enable port forwarding as described here. This also works if you connect from Windows using Putty as described here.

`python3.sif` container has few additional tools installed:

- `/tools/ukb/ukb_helper.py` - https://github.com/precimed/ukb/
- `/tools/python_convert` - https://github.com/precimed/python_convert

https://doi.org/10.1093/gigascience/giab008

[2] Aaron R. Quinlan, Ira M. Hall, BEDTools: a flexible suite of utilities for comparing genomic features, Bioinformatics, Volume 26, Issue 6, March 2010, Pages 841–842, https://doi.org/10.1093/bioinformatics/btq033

[3] B L Browning, X Tian, Y Zhou, and S R Browning (2021) Fast two-stage phasing of large-scale sequence data. Am J Hum Genet 108(10):1880-1890. doi:10.1016/j.ajhg.2021.08.005

[4] B L Browning, Y Zhou, and S R Browning (2018). A one-penny imputed genome from next generation reference panels. Am J Hum Genet 103(3):338-348. doi:10.1016/j.ajhg.2018.07.015

[5] https://enkre.net/cgi-bin/code/bgen/wiki?name=bgenix

[6] Loh, P.-R. et al. Efficient Bayesian mixed model analysis increases association power in large cohorts. Nature Genetics 47, 284–290 (2015).

[7] https://enkre.net/cgi-bin/code/bgen/wiki?name=cat-bgen

[8] O'Connell J, Gurdasani D, Delaneau O, Pirastu N, Ulivi S, et al. (2014) A General Approach for Haplotype Phasing across the Full Spectrum of Relatedness. PLOS Genetics 10(4): e1004234. https://doi.org/10.1371/journal.pgen.1004234

[9] Loh, P.-R. et al. Reference-based phasing using the Haplotype Reference Consortium panel. Nature Genetics 48, 1443–1448 (2016)

[10] https://enkre.net/cgi-bin/code/bgen/wiki?name=edit-bgen

[11] Gad Abraham, Yixuan Qiu, Michael Inouye, FlashPCA2: principal component analysis of Biobank-scale genotype datasets, Bioinformatics, Volume 33, Issue 17, September 2017, Pages 2776–2778, https://doi.org/10.1093/bioinformatics/btx299

[12] Yang J, Lee SH, Goddard ME, Visscher PM. GCTA: a tool for genome-wide complex trait analysis. Am J Hum Genet. 2011 Jan 7;88(1):76-82. doi: 10.1016/j.ajhg.2010.11.011. Epub 2010 Dec 17.

[13] Zeng, J., de Vlaming, R., Wu, Y. et al. Signatures of negative selection in the genetic architecture of human complex traits. Nat Genet 50, 746–753 (2018). https://doi.org/10.1038/s41588-018-0101-4

[14] Mägi, R., Morris, A.P. GWAMA: software for genome-wide association meta-analysis. BMC Bioinformatics 11, 288 (2010). https://doi.org/10.1186/1471-2105-11-288

[15] James K Bonfield, John Marshall, Petr Danecek, Heng Li, Valeriu Ohan, Andrew Whitwham, Thomas Keane, Robert M Davies, HTSlib: C library for reading/writing high-throughput sequencing data, GigaScience, Volume 10, Issue 2, February 2021, giab007, https://doi.org/10.1093/gigascience/giab007

[16] Ani Manichaikul, Josyf C. Mychaleckyj, Stephen S. Rich, Kathy Daly, Michèle Sale, Wei-Min Chen, Robust relationship inference in genome-wide association studies, Bioinformatics, Volume 26, Issue 22, November 2010, Pages 2867–2873, https://doi.org/10.1093/bioinformatics/btq559

[17] Speed, D., Cai, N., the UCLEB Consortium. et al. Reevaluation of SNP heritability in complex human traits. Nat Genet 49, 986–992 (2017). https://doi.org/10.1038/ng.3865

[18] Phuc-Loi Luu, Phuc-Thinh Ong, Thanh-Phuoc Dinh, Susan J Clark, Benchmark study comparing liftover tools for genome conversion of epigenome sequencing data, NAR Genomics and Bioinformatics, Volume 2, Issue 3, September 2020, lqaa054, https://doi.org/10.1093/nargab/lqaa054

[19] Cristen J. Willer, Yun Li, Gonçalo R. Abecasis, METAL: fast and efficient meta-analysis of genomewide association scans, Bioinformatics, Volume 26, Issue 17, September 2010, Pages 2190–2191, https://doi.org/10.1093/bioinformatics/btq340

[20] https://genome.sph.umich.edu/wiki/Minimac4

[21] Christopher C Chang, Carson C Chow, Laurent CAM Tellier, Shashaank Vattikuti, Shaun M Purcell, James J Lee, Second-generation PLINK: rising to the challenge of larger and richer datasets, GigaScience, Volume 4, Issue 1, December 2015, s13742–015–0047–8, https://doi.org/10.1186/s13742-015-0047-8

[22] https://www.cog-genomics.org/plink/2.0/

[23] Shing Wan Choi, Paul F O'Reilly, PRSice-2: Polygenic Risk Score software for biobank-scale data, GigaScience, Volume 8, Issue 7, July 2019, giz082, https://doi.org/10.1093/gigascience/giz082

[24] https://code.enkre.net/qctool

[25] Mbatchou, J., Barnard, L., Backman, J. et al. Computationally efficient whole-genome regression for quantitative and binary traits. Nat Genet 53, 1097–1103 (2021). https://doi.org/10.1038/s41588-021-00870-7

[26] Delaneau, O., Zagury, JF., Robinson, M.R. et al. Accurate, scalable and integrative haplotype estimation. Nat Commun 10, 5436 (2019). https://doi.org/10.1038/s41467-019-13225-y

[27] Hofmeister, R.J., Ribeiro, D.M., Rubinacci, S. et al. Accurate rare variant phasing of whole-genome and whole-exome sequencing data in the UK Biobank. Nat Genet 55, 1243–1249 (2023). https://doi.org/10.1038/s41588-023-01415-w

[28] https://github.com/precimed/simu

[29] https://www.chg.ox.ac.uk/~gav/snptest/#download

[30] O'Connell J, Gurdasani D, Delaneau O, Pirastu N, Ulivi S, et al. (2014) A General Approach for Haplotype Phasing across the Full Spectrum of Relatedness. PLOS Genetics 10(4): e1004234. https://doi.org/10.1371/journal.pgen.1004234

[31] Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R;

- `ldpred` - https://github.com/bvilhjal/ldpred is installed via pip, simply run 'ldpred –help' to get started

4.2.2 Software

List of software in the container:

OS/tool	version	license
ubuntu	20.04 (LTS)	Creative Commons CC-BY-SA version 3.0 UK licence
LDpred[32]	1.0.11	MIT
plink[Page 15, 21]	v1.90b6.18 64-bit (16 Jun 2020)	GPLv3
python3[33]	python 3.10.6 + numpy, pandas, etc.	PSF
python_convert[34]	git SHA bcde562	GPLv3

4.3 `r.sif` container

4.3.1 Description

The `r.sif` container has multiple genetics tools based or relying on R, with a full R environment and Rstudio-server, based on the Rocker Project `rocker/verse` image. Please refer to the *Software* table below for details. In addition, several standard R packages are also included (e.g. data.table, ggplot2, rmarkdown, etc.)

Please report an issue if you encounter errors that have not been reported.

For GSMR, the example data (`http://cnsgenomics.com/software/gsmr/static/test_data.zip`) is available in `$COMORMENT/containers/reference/example/gsmr` folder. You may start the container like this:

```
cd $COMORMENT/containers/reference/examples/gsmr
singularity shell --home $PWD:/home $SIF/r.sif
```

and then follow the official tutorial https://cnsgenomics.com/software/gsmr/ . Note that `gcta64` tool is also included in `r.sif` container, as the tutorial depends on it.

4.3.2 Invoking Rstudio-server

The `r.sif` container includes Rstudio-server, which can be accessed in a browser running on the host machine by

1. Start Rstudio-server on the local or remote machine as:

```
cd <working/dir>
mkdir -p run var-lib-rstudio-server
printf 'provider=sqlite\ndirectory=/var/lib/rstudio-server\n' > database.conf
singularity exec --bind run:/run,var-lib-rstudio-server:/var/lib/rstudio-server,database.
↪conf:/etc/rstudio/database.conf <path/to/r.sif /usr/lib/rstudio-server/bin/rserver --
↪www-address=127.0.0.1
```

where `<working/dir>` is the directory where you want to start Rstudio-server, and `<path/to/r.sif>` is the path to the `r.sif` container.

[32] Bjarni J. Vilhjálmsson, et al. Modeling Linkage Disequilibrium Increases Accuracy of Polygenic Risk Scores, The American Journal of Human Genetics Volume 97, Issue 4, 2015, Pages 576-592, https://doi.org/10.1016/j.ajhg.2015.09.001.
[33] Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.
[34] https://github.com/precimed/python_convert

2. (Optional) Create SSH tunnel using port 8787 from the local host to the remote machine

```
ssh -N -f -L "localhost:8787:localhost:8787" <remote/machine/address>  # replace <remote/
↪machine/address> as necessary
```

3. Then, open 0.0.0.0:8787 in a web browser on the host.

Please refer to the Rocker Project documentation for more details.

4.3.3 Software

Genetic analysis software

List of main software in the container:

OS/tool	version	license
ubuntu	20.04	Creative Commons CC-BY-SA version 3.0 UK licence
R[35]	4.0.5 (2021-03-31) + data.table, ggplot, etc.	misc
gcta64[Page 15, 12]	1.94.1	GPLv3
GenomicSEM[36]	GenomicSEM/GenomicSEM@bcbbaff	GPLv3
GSMR[37]	v1.0.9	GPL>=v2
rareGWAMA[38]	dajiangliu/rareGWAMA@72e962d	-
seqminer[39]	zhanxw/seqminer@142204d	GPL
PRSice_linux[Page 15, 2	2.3.5	GPLv3
TwoSampleMR[40]	MRCIEU/TwoSampleMR@c174107	unknown/MIT
snpStats[40]	v1.40.0	GPLv3

R packages

In addition to the `rocker/verse` image and the above genomics tools listed above there are a host of additional R packages and dependencies installed in the container. See the installer scripts for CRAN, Bioconductor, GitHub, and source packages for details.

[35] https://www.r-project.org

[36] Grotzinger, A.D., Rhemtulla, M., de Vlaming, R. et al. Genomic structural equation modelling provides insights into the multivariate genetic architecture of complex traits. Nat Hum Behav 3, 513–525 (2019). https://doi.org/10.1038/s41562-019-0566-x

[37] Zhu, Z., Zheng, Z., Zhang, F. et al. Causal associations between risk factors and common diseases inferred from GWAS summary data. Nat Commun 9, 224 (2018). https://doi.org/10.1038/s41467-017-02317-2

[38] Liu, D., Peloso, G., Zhan, X. et al. Meta-analysis of gene-level tests for rare variant association. Nat Genet 46, 200–204 (2014). https://doi.org/10.1038/ng.2852

[39] Lina Yang, Shuang Jiang, Bibo Jiang, Dajiang J Liu, Xiaowei Zhan, Seqminer2: an efficient tool to query and retrieve genotypes for statistical genetics analyses from biobank scale sequence dataset, Bioinformatics, Volume 36, Issue 19, October 2020, Pages 4951–4954, https://doi.org/10.1093/bioinformatics/btaa628

[40] Hemani, G., Haycock, P., Zheng, J., Gaunt, T., Elsworth, B., & Palmer, T. (2024). TwoSampleMR R package (v0.5.10). Zenodo. https://doi.org/10.5281/zenodo.10684540

4.4 `ldsc.sif` container

LD score regression. For details, see github.com/comorment/ldsc.

4.5 `HDL.sif` container

High-Definition Likelihood. For details, see github.com/comorment/HDL.

4.6 `MAGMA.sif` container

MAGMA, LAVA, and LAVA-partitioning tools. For details, see github.com/comorment/MAGMA.

4.7 `MiXeR.sif` container

Causal Mixed Effect Models for Cross-Trait and Cross-Ancestry Analysis. For details, see github.com/comorment/MiXeR.

4.8 References

CHAPTER

FIVE

SPECIFICATIONS

5.1 Genotype data spec

We expect imputed genotype data, which may be split into multiple *cohorts* at each site. For example, MoBa imputed genotype data is currently split into three cohorts, one per genotype array: GSA, OMNI and HCE. In this context, a *cohort* is a unit of GWAS analysis, and we do not make distinction between studies (i.e. TOP, DemGENE, HUNT, MoBa), and sub-cohorts within each study. If you have multiple studies, each with a set of sub-cohorts, we suggest to organize it into folders as follows <STUDY>_<COHORT> (for example, `MOBA_GSA`, `MOBA_OMNI`, `MOBA_HCE`, `TOP`, `DemGENE`, `HUNT`).

We expect the data to be in plink format (.bed/.bim.fam), split per chromosomes, organized for example as follows:

```
<BASEPATH>/<COHORT>/chr@.[bed,bim,fam]        # hard calls in plink format (@ indicates␣
↪chr label)
<BASEPATH>/<COHORT>/chr@.[vcf.gz,vcf.gz.tbi]  # dosages (either compressed .vcf files,␣
↪or .bgen format)
<BASEPATH>/<COHORT>/chr@.[bgen,sample]
```

It is recommended (but not required) that all genetic data within cohort is placed into it's own folder. A strict requirement is that within each cohort the files are only different by chromosome label, so it is possible to specify them by a single prefix with `@` symbol indicating the location of a chromosome label. If your data is organized differently, we recommend to use symbolic links, rather than making a full copy of the data. We also recommend to set the data as **read-only** using `chmod 0444 $BASEPATH/$COHORT/chr*` command.

Many analyses use only plink files. However, dosage files are required for some analysis, for example SAIGE. For each analysis you need to provide dosage data in a compatible format (but we will provide a set of scripts or examples to help converting data between different formats). For example, SAIG recognize either compressed `.vcf.gz` files (with corresponding `.vcf.gz.tbi` index), or `.bgen / .sample` formats. For `.vcf.gz`, please note that they should be compressed with `bgzip` (see here)

```
bgzip -c file.vcf > file.vcf.gz
tabix -p vcf file.vcf.gz
```

In the `.fam` files, we require `IID` column to be globally unique (not just unique within families). Currently there is no need to provide family annotations, sex information, or phenotype information in `.fam` files, this information is currently not used in the downstream analysis. In the future we will consider adding a separate file to add pedigree information, to accomodate more complex family structures than what is feasible with `.fam` file. Currently we do not require `IID` values to be unique across cohorts.

At of now, we only support the analysis for autosomes (chr 1. . . 22). Support for other chromosomes will came later. We expect the same set of individuals across all autosomes (chr 1. . . 22).

5.1.1 Change log

- `v0.9` - first version of this document

5.2 Phenotypes and covariates spec

For phenotypes and covariates, we expect the data to be organized in a single delimiter-separated file (hereinafter referred to as *phenotype file*), with rows corresponding to individuals, and columns corresponding to relevant variables of interest or covariates. By default the delimiter is expected to be a comma, but also can be tab, semicolon, space, or a white-space delimited file. Phenotype file should be accompanied by a *data dictionary* file, as described below. We expect a single phenotype file and a single data dictionary file for each cohort:

```
<BASEPATH>/<COHORT>/pheno.csv
<BASEPATH>/<COHORT>/pheno.dict
```

Off note: when we run GWAS analysis on a given cohort, we use subjects that has both genetic and phenotype data available, thus it's fine to include subjects without genetic data in the phenotype file. If you have sub-cohorts of the same study, it is OK to re-use one phenotype file containing information for all sub-cohorts, as long all subjects have a unique IID across cohorts.

The phenotype file must include a subject IID column, containing identifiers that matches the IID in genetic data (i.e. the `IID` column in plink `.fam` files). If `FID` column is included in the phenotype file it will be simply ignored. All subjects should be uniquely identified by their `IID`. This is against plink specification for the `.fam` file, however other software may not support subject identification through a pair of (FID, IID). Because of this we require all subject IIDs to be unique across families.

The phenotype file must contain all covariates needed for GWAS analysis, including age, sex, principal genetic components, and other confounters such as genetic batch or plate, if needed. Column names in the phenotype file must be unique. It is OK to include other relevant columns in the phenotype file - a GWAS analysis can be customized to use a subset of columns, as well as a subset of subjects.

Missing values should be encoded by empty string (see example below). It is allowed to use `#` to comment out first lines. Columns required in phenotype file: `IID` and `SEX` (note that use of `IID`, not `ID`, to match plink nomenclature).

Phenotype file should be accompanied by a *data dictionary* file, which define whether each variable is a binary (case/control), nominal (a discrete set of values) or continuous. The data dictionary should be a file with two columns, one row per variable (listed in the first column), with second column having values *BINARY*, *NOMINAL*, *ORDINAL*, *CONTINUOUS* or *IID*. Exactly one column must be marked with *IID* type. The file may have other optional columns, i.e. description of each variable. The file should have column names, first two columns must have names `FIELD` and `TYPE`.

The purpose of the `pheno.dict` file is to allow scripts to choose correct analysis: for example, if target variable is continous, we can run GWAS with linear regression mode, while if target variable is binary, we will run logistic regression; similarly, if a nominal variable is used as covariate, then it will be included as factor.

Binary variables must be encoded as 1 (cases) and 0 (controls). This is default in regenie. For plink, such coding can be used with `--1` argument. If you have a binary variable such as SEX and you want to keep the actual labels (e.g. "male" and "female"), then you should mark it as "NOMINAL" in the dictionary file.

Example `MoBa/pheno.csv` file. Subject `IID=3` have missing values for `SEX` and `MDD`.

```
# optional comments or description
IID,SEX,MDD,PC1,PC2,PC3
1,M,0,0.1,0.2,0.3
2,F,1,0.4,0.5,0.6
```

(continues on next page)

(continued from previous page)

```
3,,,0.6,0.7,0.8
4,M,0,0.9,0.1,0.2
...
```

Example `MoBa/pheno.dict` file:

```
FIELD,TYPE,DESCRIPTION
IID,IID,Identifier
SEX,NOMINAL,Sex (M - male, F - female)
MDD,BINARY,Major depression diagnosis
PC1,CONTINUOUS,First principal component
PC2,CONTINUOUS,2nd principal component
PC3,CONTINUOUS,3rd principal component
...
```

5.2.1 Change log

- `v0.9` - first version of this document
- `v0.9.1` - specify case/control coding and rename COLUMN->FIELD in the dictionary file

5.3 Summary statistics spec

The results of GWAS are represented as summary statistics, with the following columns:

- `SNP` - marker name, for example rs#.
- `CHR` - chromosome label
- `BP` - base-pair position
- `A1` - effect allele for `Z` and `BETA` columns
- `A2` - other allele
- `N` - sample size
- `CaseN`, `ControlN` - sample size for cases and controls (logistic regression only)
- `FRQ` - frequency of A1 allele
- `Z` - z-score (or t-score) of association
- `BETA` - effect size; for logistic regression, this contains `log(OR)`
- `SE` - standard error of the `BETA` column
- `L95`, `U95` - lower and upper 95% confidence interval of the `BETA`.
- `P` - p-value

For `SNP`, `CHR`, `BP`, `A1` and `A2` columns the scripts/gwas/gwas.py script will simply copy over the information from the genetic file, i.e. from `.bgen` or `.bim` files. This means that `SNP` is likely to be dbSNP rs#, or some other form of identifyied such as `CHR:BP:A1:A2`. For `CHR` and `BP`, there we don't enforce a specific genomic build - it all depends on what build was used by the genotype data. Finally, `A1` and `A2` are not guarantied to be minor or major alleles, but `A1` will be used as an effect allele for signed summary statistics (i.e. `Z` and `BETA` columns).

The sample size `N` is as reported by the software (`plink2` or `regenie`). For case-control traits, this appears to be a sum of cases and controls (not the effective sample size which would take into account imbalance between cases and controls).

`L95` and `U95` columns are only provided for `plink2` results. `CaseN` and `ControlN` columns are only provided for `plink2` results for logistic regression. If you need these columns for `regenie` analysis consider also running `plink2` analysis, and copy over the columns into your `regenie` output.

5.3.1 Comparison of columns names

- CoMorMent: this file
- LDSC: https://github.com/precimed/ldsc/blob/master/munge_sumstats.py
- BioPsyk: https://github.com/BioPsyk/cleansumstats/blob/dev/assets/schemas/cleaned-sumstats.yaml
- NORMENT: https://github.com/precimed/python_convert/blob/master/sumstats_utils.py

CoMorMent	daner	LDSC	BioPsyk	NORMENT	Description
missing	?	missing	0	missing	good idea to provide this column and referencing a line in .bim file
CHR	CHR	CHR	CHR	CHR	OK
BP	BP	BP	POS	BP	keep BP which is more informative (“POS” could also stand for genomic position)
SNP	SNP	SNP	RSID	SNP	keep SNP which makes more sense as we copy over marker name from genetic file
A1	A1	A2	EffectAllele	A1	keep A1 for consistency with LDSC even thought EffectAllele is more informative
A2	A2	A2	OtherAllele	A2	keep A2 for consistency with LDSC even though OtherAllele is more informative
P	P	P	P	PVAL	OK
SE	SE	SE	SE	SE	OK
L95	?	missing	ORL95	missing	keep “L95” as confidence interval may also be for the BETA or LOG(OR)
U95	?	missing	ORU95	missing	keep “U95”
N	?	N	N	N	OK
CaseN	Nca	N_CAS	CaseN	NCASE	OK
ControlN	Nco	N_CON	ControlN	NCONTROL	OK
INFO	INFO	INFO	INFO	INFO	OK
Direction	Direction	missing	Direction	DIRECTION	OK
BETA	BETA or OR	BETA	B	BETA or OR	keep “BETA” for consistency with LDSC (and also BETA is more informative)
Z	?	Z	Z	Z	OK
FRQ	FRQ_A_NI	FRQ	EAF	FRQ	keep “FRQ” which makes more sense for non-EUR populations
missing	?	missing	EAF_1K(	missing	not needed

5.3.2 Change log

- `v0.9` - first version of this document

CHAPTER SIX

REFERENCE

Reference data descriptions should go here.

6.1 openSNP example data

This project relies on data from openSNP, a platform for sharing personal genomics data. This public data is licensed under the CC0.

To obtain the datas for use in various examples provided here, see detailed instructions at https://github.com/comorment/opensnp.

6.2 Summary statistics

This folder contains summary statistics that we use in use cases throughout containers.

If you use this data you must comply with requirements established by the authors of these datasets. The links and a summary of these requirements is provided below.

- `clozuk_pgc2.meta.sumstats.txt.gz`, obtained from https://walters.psycm.cf.ac.uk/

```
Citation: Pardiñas AF, Holmans P, Pocklington AJ, Escott-Price V, Ripke S, Carrera␣
↪N, et al.
Common schizophrenia alleles are enriched in mutation-intolerant genes and in␣
↪regions under strong
background selection. Nat Genet. 2018. doi: 10.1038/s41588-018-0059-2.

DISCLAIMER: These data are provided "as is", and without warranty, for scientific␣
↪and educational use only.
If you download these data, you acknowledge that these data will be used only for␣
↪non-commercial
research purposes; that the investigator is in compliance with all applicable state,
↪ local, and
federal laws or regulations and institutional policies regarding human subjects and␣
↪genetics
research; that secondary distribution of the data without registration by secondary␣
↪parties is
prohibited; and that the investigator will cite the appropriate publication in any␣
↪communications
or publications arising directly or indirectly from these data. You also␣
↪acknowledge that yourself
```

(continues on next page)

(continued from previous page)

```
or any member of your research team will never attempt to identify any participant␣
↪in these studies.
```

- `SavageJansen_2018_intelligence_metaanalysis.txt.gz`, obtained from https://ctg.cncr.nl/software/summary_statistics (original file named `SavageJansen_IntMeta_sumstats.zip`, extracted and re-packed into `.gz`)

```
Summary statistics for intelligence, wave 2 from Jeanne Savage et al., 2018,
Genome-wiide association meta-analysis (N=269,867) identifies new genetic and
functional links to intelligence. Nature Genetics, 2018 Jul;50(7):912-919

Please cite this reference when using the summary statistics.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike␣
↪4.0 International License
(https://creativecommons.org/licenses/by-nc-sa/4.0/). In addition, when downloading␣
↪the sumstats you agree not
to attempt to identify individual participants and not to use the sumstats for␣
↪projects that may lead to
stigmatizing individuals or groups of individuals.
```

- `Morningness_sumstats_Jansenetal.txt.gz`, obtained from https://ctg.cncr.nl/software/summary_statistics (original file named `Morningness_sumstats_Jansenetal.txt.gz`, extracted and re-packed into `.gz`)

```
 Summary statistics for Insomnia, wave 2 from Philip Jansen et al., 2019
 Jansen PR, Watanabe K, Stringer S, Skene N, Bryois J, Hammerschlag AR, de Leeuw CA,
↪ Benjamins JS, Muñoz-Manchado AB, Nagel M, Savage JE, Tiemeier H, White T;     ␣
↪23andMe Research Team, Tung JY, Hinds DA, Vacic V, Wang X, Sullivan PF, van der␣
↪Sluis S, Polderman TJC, Smit AB, Hjerling-Leffler J, Van Someren EJW, Posthuma  ␣
↪   D. Genome-wide analysis of insomnia in 1,331,010 individuals identifies new␣
↪risk loci and functional pathways . Nature Genetics 2019 Mar;51(3):394-403. doi: ␣
↪       10.1038/s41588-018-0333-3.

Please cite this reference when using the summary statistics.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike␣
↪4.0 International License
(https://creativecommons.org/licenses/by-nc-sa/4.0/). In addition, when downloading␣
↪the sumstats you agree not
to attempt to identify individual participants and not to use the sumstats for␣
↪projects that may lead to
stigmatizing individuals or groups of individuals.
```

CHAPTER SEVEN

SCRIPTS

Documentation of helper scripts provided with this project

7.1 GWAS

For indepth explanation of the `gwas.py` and `config.yaml` codes provided here, please see the *usecases*:

- gwas_demo
- gwas_real

7.2 LDpred2

The files in this directory exemplifies how to run the LDpred2 analysis using the `bigsnpr` R library, using ldpred2.R script developed by Andreas Jangmo, Espen Hagen and Oleksandr Frei. The script is based on this tutorial. The LDpred2 method is explained in the publication:

- Florian Privé, Julyan Arbel, Bjarni J Vilhjálmsson, LDpred2: better, faster, stronger, Bioinformatics, Volume 36, Issue 22-23, 1 December 2020, Pages 5424–5431, https://doi.org/10.1093/bioinformatics/btaa1029

7.2.1 Prerequisites

This README assumes the following two repositories are cloned using git:

- http://github.com/comorment/containers
- https://github.com/comorment/ldpred2_ref
- https://github.com/comorment/opensnp

We also assume the following commands are executed from the current folder (the one containing createBackingFile.R and ldpred2.R scripts).

Note on help functions

The main R scripts contained in this directory (`ldpred2.R`, `createBackingFile.R`, `imputeGenotypes.R`, `complementSum`) are set up using the argparser package for parsing command line arguments. The help output from each script can printed to the terminal, issuing:

```
export SIF=$COMORMENT/containers/singularity
export RSCRIPT="singularity exec --home=$PWD:/home $SIF/r.sif Rscript"
# invoke ldpred2.R input options:
$RSCRIPT ldpred2.R --help
```

yielding:

```
usage: ldpred2.R [--] [--help] [--out-merge] [--geno-impute-zero]
       [--merge-by-rsid] [--opts OPTS] [--geno-file-rds GENO-FILE-RDS]
       [--sumstats SUMSTATS] [--out OUT] [--out-merge-ids
       OUT-MERGE-IDS] [--file-keep-snps FILE-KEEP-SNPS] [--ld-file
       LD-FILE] [--ld-meta-file LD-META-FILE] [--chr2use CHR2USE]
       [--col-chr COL-CHR] [--col-snp-id COL-SNP-ID] [--col-A1 COL-A1]
       [--col-A2 COL-A2] [--col-bp COL-BP] [--col-stat COL-STAT]
       [--col-stat-se COL-STAT-SE] [--col-pvalue COL-PVALUE] [--col-n
       COL-N] [--stat-type STAT-TYPE] [--effective-sample-size
       EFFECTIVE-SAMPLE-SIZE] [--n-cases N-CASES] [--n-controls
       N-CONTROLS] [--name-score NAME-SCORE] [--hyper-p-length
       HYPER-P-LENGTH] [--hyper-p-max HYPER-P-MAX] [--ldpred-mode
       LDPRED-MODE] [--cores CORES] [--set-seed SET-SEED]
       [--genomic-build GENOMIC-BUILD] [--tmp-dir TMP-DIR]

Calculate polygenic scores using ldpred2

flags:
  -h, --help                    show this help message and exit
  --out-merge                   Merge output with existing file.
  --geno-impute-zero            Set missing genotypes to zero.
...
```

Note on filtering of genotype data

The current version of these scripts performs no filtering of genotype data (e.g., minor allele frequency, imputation quality) prior to calculating linkage disequilibrium or polygenic scores. This should be done for polygenic score analyses intended for publication.

Note on missing genotypes

If genotypes are missing LDpred2 will stop and return an error (`Error:  You can't have missing values in 'X'.`). One can either pass `--geno-impute-zero` to replace missing genotypes with zero or impute with any other tool such as plink, or use imputeGenotypes.R that works for `bigSNPR` (.rds/.bk) files. Currently, only "simple" imputation with mode, mean, random or zero is supported by this script. For documentation on these methods see snp_fastImputeSimple.

First, note that using `--geno-impute-zero` is costly in computational time so it's better to impute prior to running ldpred2.R. Second, imputeGenotypes.R does not create a copy of the genotypes, thus the imputation performed persists. If you wish to keep the original .rds/.bk files you should copy these prior to imputing.

An example use of imputeGenotypes.R:

```
# Convert from plink format to bigSNPR .rds/.bk files
$RSCRIPT createBackingFile.R --file-input <fileGeno>.nomiss.bed --file-output <fileGeno>.
↪nomiss.rds

# Copy these files if you wish to leave the original files unchanged
cp <fileGeno>.rds <fileGeno>.nomiss.rds
cp <fileGeno>.bk <fileGeo>.nomiss.bk
$RSCRIPT imputeGenotypes.R --impute-simple mean0 --geno-file-rds <fileGeno>.nomiss.rds
```

Another option is to use PLINK's `--fill-missing-a2` option, and re-run `createBackingFile.R`:

```
export PLINK="singularity exec --home=$PWD:/home $SIF/gwas.sif plink"
$PLINK --bfile /REF/examples/prsice2/EUR --fill-missing-a2 --make-bed --out EUR.nomiss
$RSCRIPT createBackingFile.R --file-input EUR.nomiss.bed --file-output EUR.nomiss.rds
$RSCRIPT ldpred2.R --geno-file-rds EUR.nomiss.rds ...
```

Note on genomic builds

By default, the LDpred2 scripts assume that the genotype data and summary statistics use build GRCh37/hg19, but there are no explicit checks for consistent builds across input files. If the genotype data and summary statistics file use another build, the `--genomic-build <build>` flag should be used to specify build version, parsing either `hg18`, `hg19` or `hg38` as an argument. As of now, setting this argument will affect the loading of LD metadata only, but not the genotype data or summary statistics. A symptom of using the wrong build is that the script will match only a small fraction of variants between the genotype data, summary statistics file and/or LD reference data.

Optional: Estimating linkage disequilibrium (LD)

LDpred2 uses the LD structure when calculating polygenic scores. By default, the LDpred2.R script uses LD structure based on European samples provided by the LDpred2 authors. Instead of calculating LD on your own, the `calculateLD.R` script can be used. The output from this script can then be used as input to `LDpred2.R` (with the optional `--ld-file` flag).

It should be noted that creating these LD matrixes may require several steps that are dependent on what type of genotypic data you have. These are not covered in detail here, but a first step is to ensure that you filter out related individuals, and use a reasonably sized set of genotyped or well-imputed SNPs. How to restrict individuals and SNPs is covered below.

First, to use `calculateLD.R` you need to download genetic maps from 1000 genomes in order to convert each SNPs physical position to genomic position. If you don't provide these files, LDpred2 will try to download these automatically which will cause an error without an internet connection. To prevent this behavior, these should be downloaded manually and the folder where they are stored should be passed to the LDpred2-script using the flag `--dir-genetic-maps your-genetic/maps-directory`.

Two parameters that can be passed to `calculateLD.R` and affect the LD estimation are `--window-size` (region around index SNP in base pairs) and `--thres-r2` (threshold for including a SNP correlation in the LD). The default for `--thres-r2` is 0 in `bigsnpr::snp_cor`, but `calculateLD.R` has a default of 0.01.

The example script below will output one file per chromosome (`output/ld-chr-1.rds`, `output/ld-chr-2.rds`, . . .) and a "map" indicating the SNPs used in LD estimation (`output/map.rds`). The flag `--sumstats` can be used to filter SNPs to use where the first argument is the file and the second the column name or position of the RSID of the SNP (ie it does not need to be a proper sumstats file). The `--extract` argument does similar but expects a file that is just a list of RSIDs. These arguments can be combined and the SNPs used will then be limited to those that overlap in both files.

By a similar principle, `--extract-individuals` can be used together with `--sample-individuals` to limit the set of individuals (e.g., unrelated only), and then to draw a sample from this set.

```
# point to input/output files
export fileGeno=/REF/examples/ldpred2/g1000_eur_chr21to22_hm3rnd1.bed
export fileGenoRDS=g1000_eur_chr21to22_hm3rnd1.rds
export filePheno=/REF/examples/ldpred2/simu.pheno
export fileSumstats=/REF/examples/ldpred2/trait1.sumstats.gz
export fileOutLD=ld-chr-@.rds
export fileOutLDMap=ld-map.rds

# set environmental variables. Replace "<path/to/comorment>" with
# the full path to the folder containing cloned "containers" and "ldpred2_ref"␣
↪repositories
export COMORMENT=<path/to/comorment>
export SIF=$COMORMENT/containers/singularity
export REFERENCE=$COMORMENT/containers/reference
export LDPRED2_REF=$COMORMENT/ldpred2_ref
export SINGULARITY_BIND=$REFERENCE:/REF,${LDPRED2_REF}:/ldpred2_ref

export RSCRIPT="singularity exec --home=$PWD:/home $SIF/r.sif Rscript"

# convert genotype to LDpred2 format
$RSCRIPT createBackingFile.R --file-input $fileGeno --file-output $fileGenoRDS

# create genetics maps directory, download and process
mkdir -p 100genomes/maps
$RSCRIPT calculateLD.R --geno-file-rds $fileGenoRDS \
 --dir-genetic-maps 100genomes/maps \
 --chr2use 21 22  --sumstats $fileSumstats SNP \
 --file-ld-blocks $fileOutLD --file-ld-map $fileOutLDMap
```

Note that "bad" LD matrixes may result in optimization failures as these when running the LDpred2 scoring:

```
Running LDPRED2 auto model
Erorr in { : task 1 failed - "L-BFGS-B needs finite values of 'fn'"
Calls: snp_ldpred2_auto -> %dorng% -> do.call -> <Anonymous> -> <Anonymous>
```

The LDpred2 creators recommend creating independent LD blocks in these matrixes. the `splitLD.R` script can be used for this purpose. The setup is the same as the example above, but we add a modified `$RSCRIPT [...]` statement using the outputted matrixes from `calculateLD.R` as input to `splitLD.R`. There are several parameters to this script that will affect the "shape" of these blocks (thus subsequent performance in LDpred2). Consult `splitLD.R` and `bigsnpr::snp_ldsplit` for details.

```
$RSCRIPT splitLD.R --file-ld-blocks $fileOutLD \
 --file-ld-map $fileOutLDMap \
 --file-output-ld-blocks ld-blocked-chr@.rds
```

The script `analyzeLD.R` can be used to visualize these matrixes and provide summary statistics. We don't cover its use in detail here, but if you experience issues with LD matrixes one way may be to compare plots and statistics between your matrixes and those provided by by the LDpred2 creators. `Rscript analyzeLD.R --help` provides an overview of usage.

7.2.2 Running LDpred2 analysis

Effective sample-size

LDpred2 requires information on effective sample size. There are three ways to provide this to LDpred2:

- As a column in the summary statistics, defaulting to column `N`. If it is a different column, provide with argument `--col-n`.
- Manually calculated by providing this number with `--effective-sample-size`.
- Manually specified by providing the number of cases and controls with arguments `--n-cases` and `--n-controls`.

Specifying the effective sample size manually will override any sample size column in the sumstats. Providing both `--effective-sample-size` and `--n-cases/--n-controls` will throw an error.

Summary statistics

LDpred2 requires chromosome number, effective allele (eg A1), reference allele (eg A2, A0), and either SNP ID (RSID) or genomic position. If the summary statistics lack any of this information, the software will not run. Commonly, output from meta-analysis software such as metal do not contain this information. The complementSumstats.R script can be used to add these columns. In the example below, this script is used to append this information in a set of gzipped files inside a directory, and output these as gzipped files:

```
# set environmental variables. Replace "<path/to/comorment>" with
# the full path to the folder containing cloned "containers" and "ldpred2_ref"␣
↪repositories
export COMORMENT=<path/to/comorment>
export SIF=$COMORMENT/containers/singularity
export REFERENCE=$COMORMENT/containers/reference
export SINGULARITY_BIND=$REFERENCE:/REF,${LDPRED2_REF}:/ldpred2_ref

export RSCRIPT="singularity exec --home=$PWD:/home $SIF/r.sif Rscript"

# Directory with possibly gzipped sumstat files
dirSumstats=directory/sumstats
# Directory to direct output
dirOutput=directory/sumstats/processed
if [ ! -d $dirOutput ]; then mkdir $dirOutput; fi;

for fileSumstats in `ls $dirSumstats`; do
 echo "Processing file $fileSumstats"
 $RSCRIPT $COMORMENT/containers/LDpred2/complementSumstats.R --col-sumstats-snp-id␣
↪MarkerName --sumstats $dirSumstats/$fileSumstats --file-output $dirOutput/$fileSumstats
 gzip $dirOutput/$fileSumstats
done
```

When working within the container, the `--reference argument` can be omitted, but can be replaced with anything else along with `--col-reference-snp-id` to set the SNP ID column in the reference file. The argument `--columns-append` controls which columns to append and default to the `#CHROM` and `POS` which are the columns of chromosome and position in the `HRC` reference data (default of `--reference` argument). This script will fail if there are duplicate SNPs in any of these files that are matched. In the example below, output is piped to gzip. To write directly to a file the arguments `--file-output <output file>` and `--file-output-col-sep` controls the location of the output file and the column separator used (defaults to tab, "\t").

NOTE: In case the summary statistics file (or any other file used by the scripts) is outside the working directory, make sure to append its directory to the `SINGULARITY_BIND` environment variable as above, and refer to the file accordingly - otherwise the running container won't see the file.

Synthetic example (chr21 and chr22)

The following set of commands gives an example of how to apply LDpred2 on a synthetic example generated here. This only uses chr21 and chr22, so it runs much faster than the previous example. This example requires only `map_hm3_plus.rds`, `ldref_hm3_plus/LD_with_blocks_chr21.rds`, and `ldref_hm3_plus/LD_with_blocks_chr22.rds` files from the ldpred2_ref repository, so you may download them separately rather than clone the entire repo.

```
# point to input/output files
export fileGeno=/REF/examples/ldpred2/g1000_eur_chr21to22_hm3rnd1.bed
export fileGenoRDS=g1000_eur_chr21to22_hm3rnd1.rds
export fileSumstats=/REF/examples/ldpred2/trait1.sumstats.gz
export fileOut=simu

# set environmental variables. Replace "<path/to/comorment>" with
# the full path to the folder containing cloned "containers" and "ldpred2_ref"␣
↪repositories
export COMORMENT=<path/to/comorment>
export SIF=$COMORMENT/containers/singularity
export REFERENCE=$COMORMENT/containers/reference
export LDPRED2_REF=$COMORMENT/ldpred2_ref
export SINGULARITY_BIND=$REFERENCE:/REF,${LDPRED2_REF}:/ldpred2_ref

export RSCRIPT="singularity exec --home=$PWD:/home $SIF/r.sif Rscript"

# convert genotype to LDpred2 format
$RSCRIPT createBackingFile.R --file-input $fileGeno --file-output $fileGenoRDS

# run LDpred2 infinitesimal mode
$RSCRIPT ldpred2.R --ldpred-mode inf \
 --chr2use 21 22 \
 --geno-file-rds $fileGenoRDS \
 --sumstats $fileSumstats \
 --out $fileOut.inf

# run LDpred2 automatic mode
$RSCRIPT ldpred2.R --ldpred-mode auto \
 --chr2use 21 22 \
 --geno-file-rds $fileGenoRDS \
 --sumstats $fileSumstats \
 --out $fileOut.auto
```

Optional: Append score to existing file

It is possible to merge the calculated score to an existing file. For example, you might have a file looking like this:

```
FID IID SEX PC1 ...
1 1 M 1.1 ...
2 2 F 0.3 ...
```

By replacing the `$fileOut.inf` and `$fileOut.auto` argument above with `<myfile>` and using the options `--name-score myScoreInf` for the `--ldpred-mode inf` statement and `--name-score myScoreAuto` for the other, and add the flag `--out-merge` you end up with these scores in the existing file.

```
FID IID SEX PC1 ... myScoreInf myScoreAuto
1 1 M 1.1 ... 0.3  0.4
2 2 F 0.3 ... -0.2  0.1
```

Note that by default, merging is based on the columns `IID` and `FID` in the output file. If these columns are named differently the option `--out-merge-ids <FID column> <IID column` should be used to specify their names.

Height example

The following set of commands gives an example of how to apply LDpred2 on genetic data from the OpenSNP project, and a height GWAS sumstats file. This example requires the `opensnp_hm3.*` and `UKB_NEALELAB_2018_HEIGHT.GRCh37.hm3.gz` files from the opensnp repository, so you may download them separately rather than clone the entire repo, and place them according to the paths in the script below.

```
# Set environmental variables. Replace "<path/to/comorment>" with
# the full path to the folder containing cloned "containers" and "ldpred2_ref"␣
↪repositories
export COMORMENT=<path/to/comorment>
export SIF=$COMORMENT/containers/singularity
export REFERENCE=$COMORMENT/containers/reference
export LDPRED2_REF=$COMORMENT/ldpred2_ref
export OPENSNP=$COMORMENT/opensnp
export SINGULARITY_BIND=$REFERENCE:/REF,${LDPRED2_REF}:/ldpred2_ref,${OPENSNP}:/opensnp

# Point to LDpred2.R input/output files
export fileGeno=/opensnp/imputed/opensnp_hm3.bed
export fileGenoRDS=opensnp_hm3.rds
export fileSumstats=/opensnp/gwas/UKB_NEALELAB_2018_HEIGHT.GRCh37.hm3.gz
export fileOut=Height

export RSCRIPT="singularity exec --home=$PWD:/home $SIF/r.sif Rscript"

# convert genotype to LDpred2 format
$RSCRIPT createBackingFile.R --file-input $fileGeno --file-output $fileGenoRDS

# impute
$RSCRIPT imputeGenotypes.R --impute-simple mean0 --geno-file-rds $fileGenoRDS

# Generate PGS usign LDPRED-inf
$RSCRIPT ldpred2.R \
 --ldpred-mode inf \
```

(continues on next page)

(continued from previous page)

```
 --col-stat BETA \
 --col-stat-se SE \
 --stat-type BETA \
 --geno-file-rds $fileGenoRDS \
 --sumstats $fileSumstats \
 --out $fileOut.inf

# Generate PGS using LDPRED2-auto
$RSCRIPT ldpred2.R \
 --ldpred-mode auto \
 --col-stat BETA \
 --col-stat-se SE \
 --stat-type BETA \
 --geno-file-rds $fileGenoRDS \
 --sumstats $fileSumstats \
 --out $fileOut.auto
```

7.2.3 Output

The main LDpred2 output files are `Height.score.inf` and `Height.score.auto` put in this directory. The files are text files with tables formatted as

```
FID IID score
HG00096 HG00096 -0.733896062346436
HG00097 HG00097 0.688693127521599
HG00099 HG00099 0.203279440703434
HG00100 HG00100 0.0890499485064315
...
```

The script will also output `.bk` and `.rds` binary files with prefix `EUR` in this directory.

7.2.4 Slurm job

On an HPC resource, the same analysis can be run by first writing a job script run_ldpred2_slurm.job. In order to run the job, first make sure that the `SBATCH_ACCOUNT` environment variable is defined:

```
export SBATCH_ACCOUNT=project_ID
```

where `project_ID` is the granted project that computing time is allocated. As above, <`path/to/containers` should point to the cloned `containers` repository. Entries like `--partition=normal` may also be adapted for different HPC resources. Then, the job can be submitted to the queue by issuing `sbatch run_ldpred2_slurm.job`. The status of running jobs can usually be enquired by issuing `squeue -u $USER`.

Redirect temporary file output

By default, the LDpred2.R script will put large file(s) in the system temporary directory (using `base::tempdir()`). For use on HPC resources, use of the designated `$SCRATCH`, `$LOCALTMP`, or `$TMPDIR` directories is recommended to avoid filling up the system temporary directory.

One can redirect the temporary file output by setting the `TMPDIR` environment variable to a mounted directory on the HPC resource, by incorporating the following lines into the job script:

```
export SINGULARITY_BIND=$REFERENCE:/REF,${LDPRED2_REF}:/ldpred2_ref,$SCRATCH:/scratch
export SINGULARITYENV_TMPDIR=/scratch
```

Otherwise, the location of temporary files can be specified by the `--tmp-dir` argument to the `ldpred2.R` script.

7.3 PGS toolset

More on PGS: https://choishingwan.github.io/PRS-Tutorial/

7.3.1 Codes

- `README.md`: this file
- `pgs/pgs.py`: Python class definitions setup to create bash commands for different PGS tools (`plink`, `PRSice2`, `LDpred2`, etc.)
- `run_pgs_synthetic.py`: Python run script setup to run PGS tools on synthetic datas provided in this repository, mainly for testing purposes.
- `run_pgs_w_QC.py`: Python run script setup to run PGS tools on datas provided in this repository, mainly for testing purposes. Performs basic QC steps.
- `run_pgs_MoBa_*.py`: Python run scripts to run PGS tools on MoBa datas.
- `pgs_exec.py`: `gwas.py` like command line tool that can be used to run, create bash and slurm job scripts for different PGS tools based on user input.
- `vis_pgs_synthetic.ipynb`: Jupyter notebook plotting/comparing the output PGS scores generated by the `run_pgs_synthetic.py` script.
- `vis_pgs_w_QC.ipynb`: Jupyter notebook plotting/comparing the output PGS scores generated by the `run_pgs_w_QC.py` script.
- `start_jupyter_server.py`: start a Jupyter server using the `python3.sif` container (allows jupyter notebooks within VSCode with the Jupyter extension)
- `config.yaml`: YAML file defining some parameters for Slurm jobscripts and PGS methods
- `pgs_exec_example_*.sh`: Example bash scripts for `pgs_exec.py`
- `requirements.txt`: Python package requirements. Install with `pip install -r requirements.txt`
- `Rscripts/*.R`: misc. R scripts defining or being used by the PGS tools or optional QC steps.
- `tests/`: directory for unit tests, executable with `py.test -v tests` in this directory

7.3.2 Requirements

The basic requirements for running these codes (sans project specific genomics data) are the same as for the rest of this project, i.e., a basic Python environment and a working Singularity/Apptainer installation.

7.3.3 Running the codes

Running these codes requires Python 3.8+ (tested/developed mainly using Python 3.9+) and a working Singularity/Apptainer installation.

Input files

To work with these codes, some input files are required. These are:

Summary statistics

GWAS summary files formatted according to the *sumstats specification*

Phenotypic data

Phenotypic data formatted according to the *phenotype specification*

Genotypic data

Genotypic data formatted according to the *genotype specification*

Covariate data

Covariate data formatted according to the *covariate specification*

Output files

The output will be written to a user-specified `output/` directory, which is created if it does not exist. The output directory will contain subdirectories for each PGS method, e.g., `output/PGS_synthetic_plink/` as specified in the runtime scripts.

PGS scores

The individ level output file shared by all PGS methods is named `test.score`. The text file contains the PGS scores for each individual in the phenotype file. The first two columns are `FID` and `IID`. The third column `score` is the PGS score.

> ***NOTE:*** For PLINK and PRSice-2, the `score` column contains the "best" PGS score, i.e., the one with the highest R2 for the tested range of p-value thresholds. The scores for each p-value threshold are also written to the output directory as separate files.

Summary statistics

The file `test_summary.txt` contains the R (generalized) linear model (LM/GLM) summary statistics for the PGS model in plain-text format, while `test_summary.csv` contains the LM/GLM summary statistics in tabular (.csv) format. Both the full model and the null model are reported, typically assumed to be on the form

full model $y \sim PGS_{\text{score}} + PC_1 + PC_2 + ... + PC_n + SEX$

null model $y_{\text{null}} \sim PC_1 + PC_2 + ... + PC_n + SEX$

nocov model $y_{\text{nocov}} \sim \ PGS_{\text{score}}$

For binary traits, the GLM should use the binomial family and the logit link function to fit the model.

For binary traits, we also summarize Odds Ratios (OR) and 95% confidence intervals (CI) for the PGS models. These outputs are written to plaintext and tabular files named `test_summary.or.txt` and `test_summary.<null/full/nocov>.or.csv`, respectively.

Python runtime scripts

`run_pgs_synthetic.py`

Run PGS using PLINK, PRSice2 and LDpred2 on synthetic data provided in this repository, namely the files:

- summary statistics: `/REF/examples/ldpred2/trait1.sumstats.gz`
- phenotype data: `/REF/examples/ldpred2/simu.pheno`
- genotype data: `/REF/examples/ldpred2/g1000_eur_chr21to22_hm3rnd1.bed/bim/fam`
- covariate data: `/REF/examples/prsice2/EUR.cov`

  NOTE: Files from a full clone of https://github.com/comorment/ldpred2_ref with LDpred2 reference data is required and should be located in a directory `ldpred2_ref` as defined in the `config.yaml` file.

Running the script:

```
$ python3 run_pgs_synthetic.py
environment variables in use:
        ROOT_DIR: /nrec/space/espenh
        CONTAINERS: /nrec/space/espenh/containers
...
# A tibble: 1 × 12
  r.squared adj.r.squ...¹ sigma stati...²   p.value    df logLik   AIC   BIC devia...³
      <dbl>        <dbl> <dbl>    <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl>
1     0.765        0.762 0.494    201. 3.47e-150     8  -354.  728.  770.    120.
# ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
#   variable names ¹adj.r.squared, ²statistic, ³deviance
[1] "R2 (null model): 0.0250978564710413"
[1] "R2 (full model): 0.765387984851408"
```

NOTE: The script may require other packages than Python builtins, such as `pandas` and `pyyaml`. Install these by issuing `pip install <package>` in the current Python environment. There is also `pip install -r requirements.txt`

The output will be added to the `output/PGS_synthetic_<method>/` directories.

run_pgs_w_QC.py

Run PGS using PLINK, PRSice2 and LDpred2 on tutorial data provided in this repository, including some basic QC steps on the data based on suggestions from this tutorial.

The main input files are:

- summary statistics: /REF/examples/prsice2/Height.gwas.txt.gz
- phenotype data: /REF/examples/prsice2/EUR.height
- genotype data: /REF/examples/prsice2/EUR.bed/bim/fam
- covariates: /REF/examples/prsice2/EUR.cov
- eigenvectors: /REF/examples/prsice2/EUR.eigenvec

  NOTE: Files from a full clone of https://github.com/comorment/ldpred2_ref with LDpred2 reference data is required and should be located in a directory `ldpred2_ref` as defined in the `config.yaml` file.

Running the script:

```
python3 run_pgs_w_QC.py
```

run_pgs_MoBa_*.py

Run basic PGS for height using PLINK, PRSice2 and LDpred2, respectively on MoBa child sample data (only on TSD p697).

pgs_exec.py command line tool

Run, create bash and slurm job scripts for different PGS tools

Get a list of options:

```
$ python3 pgs_exec.py --help
usage: PGS [-h] [--method {plink,prsice2,ldpred2-inf,ldpred2-auto}] [--config CONFIG] [--
↪sumstats-file SUMSTATS_FILE] [--pheno-file PHENO_FILE] [--phenotype PHENOTYPE] [--
↪phenotype-class {CONTINUOUS,BINARY}] [--geno-file-prefix GENO_FILE_PREFIX] [--output-
↪dir OUTPUT_DIR] [--runtype {sh,slurm,subprocess}]
           {plink,prsice2,ldpred2-inf,ldpred2-auto} ...

A pipeline for PGS analysis

positional arguments:
  {plink,prsice2,ldpred2-inf,ldpred2-auto}

optional arguments:
  -h, --help            show this help message and exit
  --method {plink,prsice2,ldpred2-inf,ldpred2-auto}
                        Method for PGS
  --config CONFIG       config YAML file
  --sumstats-file SUMSTATS_FILE
                        summary statistics file
  --pheno-file PHENO_FILE
                        phenotype file
```

(continues on next page)

(continued from previous page)

```
  --phenotype PHENOTYPE
                        phenotype name (must be a column header in ``pheno_file``)
  --phenotype-class {CONTINUOUS,BINARY}
                        phenotype class
  --geno-file-prefix GENO_FILE_PREFIX
                        file path to .bed, .bim, .fam, etc. files
  --output-dir OUTPUT_DIR
                        Output file directory
  --runtype {sh,slurm,subprocess}
                        operation mode
```

Example w. PRSice2 as subprocess on synthetic dataset (`pgs_exec_example_1.sh`):

```
python3 pgs_exec.py \
    --sumstats-file /REF/examples/ldpred2/trait1.sumstats.gz \
    --pheno-file /REF/examples/ldpred2/simu.pheno \
    --phenotype trait1 \
    --phenotype-class CONTINUOUS \
    --geno-file-prefix /REF/examples/ldpred2/g1000_eur_chr21to22_hm3rnd1 \
    --output-dir output/PGS_synthetic_prsice2 \
    --runtype subprocess \
    prsice2 \
    --covariate-file /REF/examples/prsice2/EUR.cov \
    --eigenvec-file output/PGS_synthetic_prsice2/g1000_eur_chr21to22_hm3rnd1.eigenvec
```

> ***NOTE:*** The last two lines will override settings for `method: prsice2` in `config.yaml` file, being parsed to the `PRSice2.r` script

Example w. LDpred2-inf via shell (`sh`) script on synthetic dataset (`pgs_exec_example_2.sh`):

```
python3 pgs_exec.py \
    --sumstats-file /REF/examples/ldpred2/trait1.sumstats.gz \
    --pheno-file /REF/examples/ldpred2/simu.pheno \
    --phenotype trait1 \
    --phenotype-class CONTINUOUS \
    --geno-file-prefix /REF/examples/ldpred2/g1000_eur_chr21to22_hm3rnd1 \
    --output-dir output/PGS_synthetic_LDpred2_inf \
    --runtype sh \
    ldpred2-inf \
    --covariate-file /REF/examples/prsice2/EUR.cov \
    --eigenvec-file output/PGS_synthetic_LDpred2_inf/g1000_eur_chr21to22_hm3rnd1.
↪eigenvec \
    --file-geno-rds output/PGS_synthetic_LDpred2_inf/g1000_eur_chr21to22_hm3rnd1.rds \
    file-keep-snps /REF/hapmap3/w_hm3.justrs \
    chr2use 21,22
```

Which generates a shell script that can be run as

```
bash bash_scripts/ldpred2-inf-230918-12:26:35.sh  # YYMMDD-HH:MM:SS is appended to file␣
↪name
```

> ***NOTE*** Replacing `--runtype 'sh'` with `--runtype 'slurm'` and `'ldpred2-inf'` by `'ldpred2-auto'` generates a slurm jobscript using LDpred2-auto which can be submitted by issuing `bash slurm_job_scripts/ldpred2-auto.job` (cf. `pgs_exec_example_3.sh`)

7.3.4 Config file

The `config.yaml` file defines some parameters for Slurm jobscripts, job environment, and PGS methods in a YAML file. The parameters defined throughout the file are:

parameters to pass for SLURM jobs

Entries that will be put in the SLURM job scripts. These are:

```
slurm:
  job_name: pgs  #
  account: p697_norment  #
  time: "00:30:00"  # expected runtime
  cpus_per_task: 4  # number of CPU cores per task
  mem_per_cpu: 4000MB
  partition: normal

  # list of modules to load in SLURM jobs
  module_load:
    - singularity/3.7.3  # cf. the output of: "module spider singularity"
```

environment variables (edit as necessary)

Control where the PGS tools are located, where the reference data is located, etc. These are:

```
environ:
  # mandatory root directory containing all inferred directories (edit as necessary).
  ROOT_DIR: "/nrec/space/espenh"

# dependent environment variables (edit as necessary)
# NB: "SIF" is mandatory
environ_inferred:
  # folder containing full clone of https://github.com/comorment/containers
  CONTAINERS: '$ROOT_DIR/containers'
  # reference data within containers repo
  REFERENCE: "$CONTAINERS/reference"
  # directory with singularity containers (.sif files)
  SIF: "$CONTAINERS/singularity"
  # folder containing  full clone of https://github.com/comorment/ldpred2_ref with␣
↪LDpred2 reference data
  LDPRED2_REF: "$ROOT_DIR/ldpred2_ref"
  # folder containing LDpred2 R scripts
  LDPRED2_SCRIPTS: "$CONTAINERS/scripts/pgs/LDpred2"

# for SINGULARITY_BIND variable to set in job scripts
# NB! will be set as "export SINGULARITY_BIND=value0:/key0,value1:/key1,..."
# NB! Also mandatory
SINGULARITY_BIND:
  REF: '$REFERENCE'
  ldpred2_ref: '$LDPRED2_REF'
  ldpred2_scripts: '$LDPRED2_SCRIPTS'
```

Parameters specific to each PGS calculating tool

Refer class documentation of each tool for details

Plink

used by class PGS_PLINK

```
plink:
  clump_p1: 1
  clump_r2: 0.1
  clump_kb: 250
  range_list: [0.001, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 1]
  strat_indep_pairwise: [250, 50, 0.25]
  nPCs: 6
  score_columns: [SNP, A1, BETA]
  threads: 4
```

PRSice-2

used by class PGS_PRSice2

```
prsice2:
  MAF: 0.01
  INFO: 0.8
  nPCs: 6
  thread: 4
```

LDpred2

used by class PGS_LDpred2

```
ldpred2:
  nPCs: 6
  cores: 4
```

CHAPTER

EIGHT

USECASES

The following use cases illustrate applications of CoMorMent containers:

- gwas_demo.md describes how to run a demo GWAS analysis with `plink2` and `regenie` following the *CoMorMent specifications*.
- gwas_real.md illustrates usage of gwas.py analysis on MoBa height and depression phenotypes
- meta_simu.md describe use cases related to meta-analysis
- bolt-lmm_demo.md describes how to run bolt-lmm
- *scripts/pgs/LDpred2/README.md* describes how to run PRS analyses using LDpred2

CHAPTER NINE

API REFERENCE

9.1 gwas.py

class gwas.**ActionAppendDeprecated**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Methods

`__call__`(parser, namespace, values[, ...])	Call self as a function.

format_usage

class gwas.**ActionStoreDeprecated**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Methods

`__call__`(parser, namespace, values[, ...])	Call self as a function.

format_usage

class gwas.**LoadFromFile**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Methods

`__call__`(parser, namespace, values[, ...])	Call self as a function.

format_usage

`class gwas.`**`Logger`**`(`*fh*, *mode*`)`

Lightweight logging.

Methods

`error`(msg)	Print to log file, error file and stdout with a single command.
`log`(msg)	Print to log file and stdout with a single command.

`error``(`*msg*`)`

Print to log file, error file and stdout with a single command.

`log``(`*msg*`)`

Print to log file and stdout with a single command.

`class gwas.`**`NumpyEncoder`**`(`*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*`)`

Methods

`default`(obj)	Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).
`encode`(o)	Return a JSON string representation of a Python data structure.
`iterencode`(o[, _one_shot])	Encode the given object and yield each string representation as available.

`default``(`*obj*`)`

Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
```

(continues on next page)

(continued from previous page)

```
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

`gwas.sec_to_str(t)`

Convert seconds to days:hours:minutes:seconds

9.2 pgs.pgs

`class pgs.pgs.BasePGS(`*sumstats_file='/REF/examples/prsice2/Height.gwas.txt.gz', pheno_file='/REF/examples/prsice2/EUR.height', phenotype='Height', phenotype_class='CONTINUOUS', geno_file_prefix='/REF/examples/prsice2/EUR', output_dir='qc-output', **kwargs*`)`

Base PGS object declaration with some shared properties for subclassing

Parameters

sumstats_file: str
summary statistics file (.gz)

pheno_file: str
phenotype file (for instance, .height)

phenotype: str or None
if not `None`, phenotype name (must be a column header in pheno_file)

phenotype_class: str
phenotype class, either `CONTINUOUS` or `BINARY`

geno_file_prefix: str
path to QC'd .bed, .bim, .fam files (w.o. file ending) (</ENV/path/to/data/file>)

output_dir: str
path for output files (<path>)

****kwargs**

Attributes

data_prefix: str
file name prefix of .bed, .bim, etc. files

Methods

get_str:	abstract method for returning string with commands

`abstract get_str()`

Required public method

`class pgs.pgs.PGS_LDpred2(`*sumstats_file='/REF/examples/prsice2/Height.gwas.txt.gz', pheno_file='/REF/examples/prsice2/EUR.height', phenotype='Height', phenotype_class='CONTINUOUS', geno_file_prefix='/REF/examples/prsice2/EUR', output_dir='PGS_ldpred2_inf', method='auto', file_geno_rds='PGS_ldpred2_inf/EUR.rds', **kwargs*`)`

Helper class for setting up LDpred2 PRS analysis. Inherited from class `BasePGS`

Parameters

sumstats_file: str
summary statistics file (.gz)

pheno_file: str
phenotype file (for instance, .height)

phenotype: str or None
if not `None`, phenotype name (must be a column header in `pheno_file`)

phenotype_class: str
phenotype class, either 'CONTINUOUS' or 'BINARY'

geno_file_prefix: str
path to QC'd .bed, .bim, .fam files (w.o. file ending) (</ENV/path/to/data/file>)

output_dir: str
path for output files (<path>)

method: str
LDpred2 method, either "auto" (default) or "inf" for infinitesimal

file_geno_rds: str
base name for .rds file output

****kwargs**
dict of additional keyword/arguments pairs parsed to the `$LDPRED2_SCRIPTS/ldpred2.R` script (see file for full set of options). If the option is only a flag without value, set value as None-type or empty string.

Methods

generate_eigenvec_eigenval_files:
get_model_evaluation_str:
get_str:

`generate_eigenvec_eigenval_files`(*nPCs=6*)

Return string which can be included in job script for generating .eigenvec and .eigenval files in the output directory using PLINK

Parameters

nPCs: int
number of PCs to account for

`get_model_evaluation_str`(*eigenvec_file=None*, *nPCs=None*, *covariate_file=None*)

Return callable string for fitting a simple linear model between PGS score and phenotype data using R stats::lm, printing stats::lm.fit.summary output to file

Parameters

eigenvec_file: path
path to file with PCs (no header, columns FID, IID, PC1, PC2, . . .)

nPCs: int
number of PCs to account for

covariate_file: path
path to file with covariates (header, columns FID, IID, <covariate>)

Returns

str

`get_str`(*create_backing_file=True*)

Public method to create commands

Parameters

create_backing_file: bool
if True (default), prepend statements for running the `$LDPRED2_SCRIPTS/createBackingFile.R` script, generating `file_geno_rds`

Returns

list of str
list of command line statements for analysis run

`class pgs.pgs.PGS_PRSice2`(*sumstats_file='/REF/examples/prsice2/Height.gwas.txt.gz'*, *pheno_file='/REF/examples/prsice2/EUR.height'*, *phenotype='Height'*, *phenotype_class='CONTINUOUS'*, *geno_file_prefix='/REF/examples/prsice2/EUR'*, *output_dir='PGS_prsice2'*, *covariate_file='/REF/examples/prsice2/EUR.cov'*, *eigenvec_file='/REF/examples/prsice2/EUR.eigenvec'*, *nPCs=6*, *MAF=0.01*, *INFO=0.8*, ***kwargs*)

Helper class for setting up PRSice-2 PRS analysis. Inherited from class `BasePGS`

Parameters

sumstats_file: str
summary statistics file (.gz)

pheno_file: str
phenotype file (for instance, .height)

phenotype: str or None
if not `None`, phenotype name (must be a column header in `pheno_file`)

phenotype_class: str
phenotype class, either `CONTINUOUS` or `BINARY`

geno_file_prefix: str
path to QC'd .bed, .bim, .fam files (w.o. file ending) (</ENV/path/to/data/file>)

output_dir: str
path for output files (<path>)

covariate_file: str or None
path to covariate file (.cov)

eigenvec_file: str or None
path to eigenvec file (.eig) with PCs

nPCs: int
number of Principal Components (PCs) to include in covariate generation

MAF: float
base-MAF upper threshold value (0.01)

INFO: float
base-INFO upper threshold value (0.8)

****kwargs**
dict of additional keyword/arguments pairs parsed to the Rscripts/PRSice.R script (see file for full set of options). If the option is only a flag without value, set value as None-type or empty string.

Attributes

data_prefix: str
file name prefix of .bed, .bim, etc. files

Methods

get_model_evaluation_str:	
get_str:	

`get_model_evaluation_str()`

Return callable string for fitting a simple linear model between PGS score and phenotype data using R stats::lm, printing stats::lm.fit.summary output to file

Returns

str

`get_str()`

Public method to create commands

Returns

list of str
list of command line statements for analysis run

`class pgs.pgs.PGS_Plink`(*sumstats_file='/REF/examples/prsice2/Height.gwas.txt.gz'*, *pheno_file='/REF/examples/prsice2/EUR.height'*, *phenotype='Height'*, *phenotype_class='CONTINUOUS'*, *geno_file_prefix='QC_data/EUR'*, *output_dir='PGS_plink'*, *covariate_file='/REF/examples/prsice2/EUR.cov'*, *eigenvec_file='/REF/examples/prsice2/EUR.eigenvec'*, *clump_p1=1*, *clump_r2=0.1*, *clump_kb=250*, *clump_snp_field='SNP'*, *clump_field='P'*, *range_list=None*, *strat_indep_pairwise=None*, *nPCs=6*, *score_columns=None*, ***kwargs*)

Helper class for setting up Plink PRS analysis. Inherited from class `BasePGS`

Parameters

sumstats_file: str
summary statistics file (.gz)

pheno_file: str
phenotype file (for instance, .height)

phenotype: str or None
if not `None`, phenotype name (must be a column header in `pheno_file`)

phenotype_class: str
phenotype class, either `CONTINUOUS` or `BINARY`

geno_file_prefix: str
path to QC'd .bed, .bim, .fam files (w.o. file ending) (</ENV/path/to/data/file>)

output_dir: str
path for output files (<path>)

covariate_file: str
path to covariance file (.cov)

eigenvec_file: str or None
None, or path to eigenvec file (.eigenvec)

clump_p1: float
plink –clump-p1 parameter value (default: 1)

clump_r2: float
plink –clump-r2 parameter value (default: 0.1)

clump_kb: float
plink –clump-r2 parameter value (default: 250)

clump_snp_field: str
plink –clump-snp-field parameter value (default: 'SNP')

clump_field: str
plink –clump-field parameter value (default: 'P')

range_list: list of floats
list of p-value ranges for plink –q-score-range arg. (default: [0.001, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5])

strat_indep_pairwise: list of scalars
plink –indep-pairwise parameters for stratification describing window size (kb), step size (variant ct), r^2 threshold (default: [250, 50, 0.25])

nPCs: int
plink –pca parameter value (default: 6)

# score_args: list
# plink –score arguments (default: [3, 4, 12, 'header'])
score_columns: list of str
for plink's –score, column names in sumstats_file. Requires header. Default: ['SNP', 'A1', 'BETA']

****kwargs**

Attributes

data_prefix: str
file name prefix of .bed, .bim, etc. files

Methods

get_model_evaluation_str:	
get_str:	

`get_model_evaluation_str()`

Return callable string for fitting a simple linear model between PGS score and phenotype data using R stats::lm, printing stats::lm.fit.summary output to file

Returns

str

`get_str`(*mode='basic'*, *update_effect_size=False*)

Parameters

mode: str
‘basic’ or ‘stratification’

update_effect_size: bool
if True, compute PGS using OR

`class pgs.pgs.Standard_GWAS_QC`(*sumstats_file='/REF/examples/prsice2/Height.gwas.txt.gz'*, *pheno_file='/REF/examples/prsice2/EUR.height'*, *geno_file_prefix='/REF/examples/prsice2/EUR'*, *output_dir='QC_data'*, *phenotype='Height'*, *data_postfix='.QC'*, *QC_target_kwargs=None*, *QC_prune_kwargs=None*, *QC_relatedness_prune_kwargs=None*, ***kwargs*)

Helper class for common GWAS QC. Inherited from class `BasePGS`

Based on the tutorial https://choishingwan.github.io/PRS-Tutorial/target/#qc-of-target-data

Use with caution. This class is not fully tested.

Parameters

sumstats_file: str
summary statistics file (.gz)

pheno_file: str
phenotype file (for instance, .height)

geno_file_prefix: str
path to (raw) .bed, .bim, .fam files (w.o. file ending) (</ENV/path/to/data/file>)

output_dir: str
path for output files (<path>)

phenotype: str
default: ‘Height’

data_postfix: str
default: ‘.QC’

QC_target_kwargs: dict
default: {‘maf’: 0.01, ‘hwe’: 1e-6, ‘geno’: 0.01, ‘mind’: 0.01}

QC_prune_kwargs: dict
default: {‘indep-pairwise’: [200, 50, 0.25]}

QC_relatedness_prune_kwargs: dict
defaultL: {'rel-cutoff': 0.125}

****kwargs**

Methods

get_str:

`get_str()`
Standard GWAS QC

`pgs.pgs.convert_dict_to_str(`*d*, *key_prefix='--'*`)`

Parameters

d: dict
key, value pairs

key_prefix: str
string prefix for key names. Default: "–"

Returns

str
string formatted as "–key0 value0 –key1 value1 …". In case values are iterable, it will be formatted as "–key0 value0[0] value0[1] … –key0"

`pgs.pgs.df_colums_to_file(`*source_file*, *output_file*, *usecols=None*, *delim_whitespace=True*, *delimiter=None*, ***kwargs*`)`
Extract columns from dataframe (.csv) on file to output_file

Parameters

source_file: file path
.csv (or similar) input file read by pandas.read_csv.

output_file: file path
output file to be written

usecols: list of str or None
columns to read and write

delim_whitespace: bool
parsed to df.read_csv. Default: True

delimiter: None or str
delimiter. Default: None

****kwargs**
keyword arguments parsed to pd.read_csv()

`pgs.pgs.post_run_plink(`*output_dir*, *data_prefix*, *best_fit_file='best_fit_prs.csv'*, *score_file='test.score'*`)`
Read best-fit predictions and export standardized `test.score` file to output_dir from class PGS_Plink output

Parameters

output_dir: path
path to output directory

data_prefix: str
standard file name prefix (for .bed, .bim, .fam, etc.)

best_fit_file: str
.csv file in `output_dir` with best fit Threshold value. Default: 'best_fit_prs.csv'

score_file: str
test score file in `output_dir`. Default: 'test.score'

`pgs.pgs.`**`post_run_prsice2`**`(`*output_dir*, *data_prefix*, *score_file='test.score'*`)`

Read best-fit predictions and export standardized `test.score` file to output_dir from class PGS_PRSice2 output

Parameters

output_dir: path
path to output directory

data_prefix: str
standard file name prefix (for .bed, .bim, .fam, etc.)

score_file: str
test score file in `output_dir`. Default: 'test.score'

`pgs.pgs.`**`run_call`**`(`*call*`)`

run subprocess call

`pgs.pgs.`**`set_env`**`(`*config*`)`

Function to set environment variables from config.yaml

TODO: add defaults

Parameters

config: dict
config dictionary from config.yaml (or similar file)

CHAPTER

TEN

CONTRIBUTING

Thanks for considering contributing! Please read this document to learn the various ways you can contribute to this project and how to go about doing it.

10.1 Bug reports and feature requests

10.1.1 Did you find a bug?

First, do a quick search to see whether your issue has already been reported. If your issue has already been reported, please comment on the existing issue.

Otherwise, open a new GitHub issue. Be sure to include a clear title and description. The description should include as much relevant information as possible. The description should explain how to reproduce the erroneous behavior as well as the behavior you expect to see. Ideally you would include a code sample or an executable test case demonstrating the expected behavior.

10.1.2 Do you have a suggestion for an enhancement or new feature?

We use GitHub issues to track feature requests. Before you create a feature request:

- Make sure you have a clear idea of the enhancement you would like. If you have a vague idea, consider discussing it first on a GitHub issue.
- Check the documentation to make sure your feature does not already exist.
- Do a quick search to see whether your feature has already been suggested.

When creating your request, please:

- Provide a clear title and description.
- Explain why the enhancement would be useful. It may be helpful to highlight the feature in other libraries.
- Include code examples to demonstrate how the enhancement would be used.

10.2 Making a pull request

When you're ready to contribute code to address an open issue, please follow these guidelines to help us be able to review your pull request (PR) quickly.

1. **Initial setup** (only do this once)

   If you haven't already done so, please fork this repository on GitHub.

   Then clone your fork locally with

   ```
   git clone https://github.com/USERNAME/containers.git
   ```

   or

   ```
   git clone git@github.com:USERNAME/containers.git
   ```

   At this point the local clone of your fork only knows that it came from *your* repo, github.com/USERNAME/containers.git, but doesn't know anything the *main* repo, https://github.com/comorment/containers.git. You can see this by running

   ```
   git remote -v
   ```

   which will output something like this:

   ```
   origin https://github.com/USERNAME/containers.git (fetch)
   origin https://github.com/USERNAME/containers.git (push)
   ```

   This means that your local clone can only track changes from your fork, but not from the main repo, and so you won't be able to keep your fork up-to-date with the main repo over time. Therefore you'll need to add another "remote" to your clone that points to https://github.com/comorment/containers.git. To do this, run the following:

   ```
   git remote add upstream https://github.com/comorment/containers.git
   ```

   Now if you do `git remote -v` again, you'll see

   ```
   origin https://github.com/USERNAME/containers.git (fetch)
   origin https://github.com/USERNAME/containers.git (push)
   upstream https://github.com/comorment/containers.git (fetch)
   upstream https://github.com/comorment/containers.git (push)
   ```

2. **Ensure your fork is up-to-date**

   Once you've added an "upstream" remote pointing to https://github.com/comorment/containers.git, keeping your fork up-to-date is easy:

   ```
   git checkout main  # if not already on main
   git pull --rebase upstream main
   git push
   ```

3. **Create a new branch to work on your fix or enhancement**

   Committing directly to the main branch of your fork is not recommended. It will be easier to keep your fork clean if you work on a separate branch for each contribution you intend to make.

   You can create a new branch with

```
# replace BRANCH with whatever name you want to give it
git checkout -b BRANCH
git push -u origin BRANCH
```

4. **Test your changes**

   Our continuous integration (CI) testing runs a number of checks for each pull request on GitHub Actions. You can run most of these tests locally, which is something you should do *before* opening a PR to help speed up the review process and make it easier for us.

   And finally, please update the *CHANGELOG* with notes on your contribution in the "Unreleased" section at the top.

   After all of the above checks have passed, you can now open a new GitHub pull request. Make sure you have a clear description of the problem and the solution, and include a link to relevant issues.

   We look forward to reviewing your PR!

10.3 Information for developers

The list of tools included in the different Dockerfiles and installer bash scripts for each container is provided *here*. Please keep this up to date when pushing new container builds.

10.3.1 Sphinx

We use sphinx to generate online documentation from README.md files of this repository. This uses MyST package to generate links in the documentation. Here are few rules that we follow across `.md` files to make it work well:

- use full path to the file in this repository

10.3.2 Folder structure

These folders are relevant to the users:

- `docs` folder contain user documentation
- `usecases` folder contain extended examples / tutorials
- `singularity` folder contain pre-build containers
- `reference` folder contain reference data used in use-cases
- `scripts` folder contain pipelines such as `gwas.py` and `pgs-toolkit`, as well as other helper scripts.

These folders are relevant to developers:

- `docker` folder contains several `Dockerfile` files (container definitions) and relevant shell scripts (in `docker/scripts/`) used within those Dockerfile's. Unit-tests validating functionality of the resulting containers are available in the `tests` folder.
- `sphinx-docs` provides scripts used to build sphinx documentation.

10.3.3 Note about NREC machine

We use NREC machine to develop and build containers. NREC machine has small local disk (~20 TB) and a larger external volume attached (~400 TB) If you use NREC machine, it's important to not store large data or install large software to your home folder which is located on a small disk, using `/nrec/projects space` instead:

```
Filesystem                              Size  Used Avail Use% Mounted on
/dev/sda1                                20G  9.6G  9.7G  50% /
/dev/mapper/nrec_extvol-comorment       393G  346G   28G  93% /nrec/projects
/dev/mapper/nrec_extvol_2-comorment_2  935G  609G  279G  69% /nrec/space
```

Both docker and singularity were configured to avoid placing cached files into local file system. For docker this involves changing `/etc/docker/daemon.json` file by adding this:

```
{
    "data-root": "/nrec/projects/docker_root"
}
```

(as described https://tienbm90.medium.com/how-to-change-docker-root-data-directory-89a39be1a70b ; you may use `docker info` command to check the data-root)

For singularity, the configuration is described here https://sylabs.io/guides/3.6/user-guide/build_env.html and it was done for the root user by adding the following line into /etc/environment

```
export SINGULARITY_CACHEDIR="/nrec/projects/singularity_cache"
```

Common software, such as git-lfs, is installed to /nrec/projects/bin. Therefore it's reasonable for all users of the NREC comorment instance to add this folder to the path by changing `~/.bashrc` and `~/.bash_profile`.

```
export PATH="/nrec/projects/bin:$PATH"
```

A cloned version of comorment repositories is available here:

```
/nrec/projects/github/comorment/containers
/nrec/projects/github/comorment/reference
```

Feel free to change these folders and use git pull / git push. TBD: currently the folder is cloned as 'ofrei' user - I'm not sure if it will actually work to pull & push. But let's figure this out.

10.3.4 Testing container builds

Some basic checks for the functionality of the different container builds are provided in `<containers>/tests/`, implemented in Python. The tests can be executed using the Pytest testing framework.

To install Pytest in the current Python environment, issue:

```
pip install pytest  # --user optional
```

New virtual environment using conda:

```
conda create -n pytest python=3 pytest -y  # creates env "pytest"
conda activate pytest  # activates env "pytest"
```

Then, all checks can be executed by issuing:

```
cd <containers>
py.test -v tests  # with verbose output
```

Checks for individual containers (e.g., `gwas.sif`) can be executed by issuing:

```
py.test -v tests/test_<container-prefix>.py
```

Note that the proper container files (*.sif files) corresponding to the different test scripts must exist in `<containers>/singularity/>`, not only git LFS pointer files.

10.3.5 Git clone ignoring LFS

See stackoverflow.com/questions/42019529/how-to-clone-pull-a-git-repository-ignoring-lfs

```
GIT_LFS_SKIP_SMUDGE=1 git clone git@github.com:comorment/containers.git
```

CHAPTER

ELEVEN

CHANGELOG

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

Note that CoMorMent containers are organized using several GitHub repositories:

- https://github.com/comorment/containers - .sif files, public reference data, documentation, common scripts
- https://github.com/comorment/reference - private reference data with access restricted to CoMorMent collaborator

All of the above repositories are covered by this CHANGELOG. They will have the same version tags on github. In addition, we have repositories containing specific tools, e.g. https://github.com/comorment/HDL, which will be covered by their own CHANGELOG.md file.

To identify the version of a .sif file, run `md5sum <container>.sif` command and find the MD5 checksum in the list below. If MD5 sum is not listed for a certain release then it means that the container hasn't been changed from the previous release.

11.1 [Unreleased]

11.1.1 Added

- Added options `--extract`, `--extract-step1`, `--extract-step2`, `--exclude`, `--exclude-step1`, and `--exclude-step2` to the `gwas.py` script to enable inclusion and exclusion of SNPs
- Added Rstudio-server and R packages info to `r.sif` container documentation

11.1.2 Updated

- Rebuilt `gwas.sif` container with md5sum checksum:

```
4e295149f3a5e25588cc4a1f1d39876c  singularity/gwas.sif
```

- Compile regenie with `HAS_BOOST_IOSTREAM=1` and `HTSLIB_PATH` options
- Change LDpred2 usage example to use the OpenSNP based datasets
- Bundle of sphinx documentation build updates/restructures
- Refer to the project as "COSGAP-containers"
- Minor changes to documentation + suggestion of TOC

- migrate online documentation to cosgap.readthedocs.io
- updated documentation to reflect the new project name
- added references/urls to software tables in the documentation for singularity containers
- update citation info

11.1.3 Fixed

- Fixed brittle tests if `TMPDIR` is not `/tmp`

11.1.4 Removed

- Removed Saige support and Saige-related files

11.1.5 Misc

- Miscellaneous goes here

11.2 [1.8.1] - 2024-03-05

11.2.1 Fixed

- Fixed parsing of `IID` field in `pheno.dict`
- Fixed issue with files with different suffixes produced by plink2 for binary phenotypes in `gwas.py`

11.3 [1.8.0] - 2024-02-22

11.3.1 Added

- Added scripts to analyze and filter bigSNPR LD matrixes (`scripts/pgs/LDpred2/analyzeLD.R`, `scripts/pgs/LDpred2/splitLD.R`).

11.4 [1.7.2] - 2024-02-14

11.4.1 Updated

- Rebuilt `r.sif` container with md5sum checksum:

```
3d69fc2168ef98d1eda3da05391cd6e4  singularity/r.sif
```

11.4.2 Added

- added `CC-GWAS` R package to `r.sif` container

11.5 [1.7.1] - 2024-02-06

11.5.1 Fixed

- Fixed parsing of `--genomic-build hg18/hg38` in `ldpred2.R`

11.6 [1.7.0] - 2024-02-02

11.6.1 Added

- Added `samtools 1.19.2`, `bedtools 2.31.1`, `liftOver (latest)` to `gwas.sif` container
- Added corresponding unit tests

11.6.2 Updated

- Updated the following binaries (not listing apt package updates) in gwas.sif built
  - bcftools to 1.19
  - bolt to 2.4.1
  - gcta to 1.94.1
  - gctb to 2.04.3
  - htslib to 1.19.1
  - king to 2.3.2
  - minimac4 to 4.1.6
  - plink to v1.90b7.2 64-bit (11 Dec 2023)
  - plink2 to v2.00a5.10LM 64-bit Intel (5 Jan 2024)
  - plink2_avx2 to v2.00a5.10LM AVX2 Intel (5 Jan 2024)
  - PRSice_linux to 2.3.5
  - regenie to 3.4.1
  - vcftools to git SHA: d511f469e87c2ac9779bcdc3670b2b51667935fe (0.1.17dev)
- Rebuilt `gwas.sif` w. md5sum checksum:

```
a775f4216b15b731471821d0c2a0da43  singularity/gwas.sif
```

- updated installer scripts

11.6.3 Fixed

- Broken `docker/scripts/build_docker.sh` script

11.7 [1.6.0] - 2023-12-12

11.7.1 Added

- Added `gdb` debugger, `ldak` and `snptest` binaries to `gwas.sif` container
- Added tests for `ldak` and `snptest` binaries in `gwas.sif` container

11.7.2 Updated

- updated `metal` to version `2020-05-05` in `gwas.sif`
- updated `qctool` to `v2.2.2` and added related binaries `inthinnerator`, `hptest`, `ldbird` and `selfmap` to `gwas.sif`
- rebuilt `gwas.sif` (md5 checksum b6104b58d21f862f9d61a86d9d4802a6)

11.8 [1.5.1] - 2023-10-20

11.8.1 Fixed

- Fixed broken ReadTheDocs documentation build

11.9 [1.5.0] - 2023-10-17

11.9.1 Added

- Added `<containers>/scripts/pgs/pgs_toolkit`, a Python toolkit for computing PGS using LDpred2, PRSice2 or PLINK
- Added `<containers>docker/scripts/build_docker.sh` script replacing corresponding build statement in `Makefile`
- Added test for `gcta`

11.9.2 Updated

- Updated `r.sif` build with many additional R packages, with corresponding updates to build recipes and tests
- Use `https://packagemanager.posit.co/cran/__linux__/focal/2023-02-16` as main R package repo
- `r.sif` md5 checksum:

```
1280ba24d99664d450b2e4c4a9c00587  singularity/r.sif
```

- Updated GitHub workflow versions to current versions

11.9.3 Removed

- removed logging of `docker build ...` in `docker/Makefile` (issues with piping to `tee` in case of build errors)

11.10 [1.4.0] - 2023-10-17

11.10.1 Added

- Added phasing/imputation tools `beagle`, `duohmm`, `eagle`, `shapeit5`, `switchError`, to `gwas.sif` container + updated tests

11.10.2 Fixed

- Fix issue that shell script wouldn't capture failing statements

11.10.3 Updated

- Updated `gwas.sif` Dockerfile and installed shell scripts (misc. dependencies updates, installing `gcta` version 1.93.3beta2)
- Rebuilt `gwas.sif` using Docker `--no-cache` option to fix missing `minimac4` binary, w. md5 checksum:

```
a1dd235221902741bf5773945a584e47  singularity/gwas.sif
```

11.10.4 Removed

- Removed unused `install_miniconda.sh` script from `src/scripts` folder

11.11 [1.3.9] - 2023-10-17

11.11.1 Added

- User-set directory option for temporary files during LDpred2 runs, by default `base::tempdir()`

11.12 [1.3.8] - 2023-10-17

11.12.1 Fixed

- Added `--genomic-build hg18/hg19/hg38` option to `ldpred2.R` to use correct LD reference meta file `pos` column name

11.13 [1.3.7] - 2023-10-17

11.13.1 Added

- Added a feature to read and convert BGEN (.bgen) files to `scripts/pgs/LDpred2/createBackingFile.R`

11.14 [1.3.7] - 2023-10-12

- User-set directory for temporary files during LDpred2 runs, by default `base::tempdir()`

11.15 [1.3.6] - 2023-08-17

11.15.1 Fixed

- Ignore LDpred2 `--col-bp <column>` arg in case `--merge-by-rsid` is used

11.16 [1.3.5] - 2023-08-17

11.16.1 Updated

- Updated LDpred2 README file

11.17 [1.3.4] - 2023-06-22

11.17.1 Updated

- Update regenie to v3.2.8

11.17.2 Fixed

- #187 - Regression in gwas.py in handling of info, maf, hwe and geno filters

11.18 [1.3.3] - 2023-06-14

11.18.1 Updated

- Removed time consuming genotype missingness check from `ldpred2.R`.

11.19 [1.3.2] - 2023-06-12

11.19.1 Fixed

- Fixed misc. issues with cross references in online documentation

11.20 [1.3.1] - 2023-06-07

11.20.1 Added

- Added unittest for uppercase chromosome column name in sumstats files, that may also contain chromosomes encoded as character(s)

11.20.2 Fixed

- Fixed issue with character encoding in sumstats files, in case chromosome column name is uppercase.

11.21 [1.3.0] - 2023-05-19

11.21.1 Added

- Added to `ldpred2.R`: Multi-threading of `snp_ldsc`, arguments for parameters to `snp_ldpred2_auto`, and alternative effective sample-size calculation through `--n-cases` and `n-controls`.

11.21.2 Fixed

- Solved error due to case-sensitive handling of `--col-chr` in `ldpred2.R` and naming of diagnostic plot when using `--name-score`.

11.22 [1.2] - 2023-05-11

11.22.1 Added

- Added `RELEASES.md` file explaining steps needed to make releases.
- Added `PRSice_linux` to `r.sif`
- Added tests for `gwas.py`
- Added package `GWASTools` to `r.sif`.
- Added confidence intervals to qq plots created by `gwas.py` using `GWASTools` R package.
- Added status badges and citation.cff file

11.22.2 Updated

- Updated file and folder layout, fixing minor documentation issues. Moving from `m2r2` to `Myst-parser` for Sphinx-generated online docs.
- Rebuilt the R container

-
  ```
  5ecbfc50f96bc6b25f61858927283e2d  singularity/r.sif
  ```
- Rebuilt the R container

  ```
  23d195a10b84603b15d0e8c42df40fbd  singularity/r.sif
  ```

11.22.3 Fixed

- Set version file info to 1.2.dev (was 0.1.1dev)
- Fixed bad parsing of arbitrary length list of args in `usecases/LDpred2/complementSumstats.R`
- Made `usecases/LDpred2/complementSumstats.R` write output file by default, not stdout.
- Fixed print statement in `usecases/LDpred2/complementSumstats.R` causing crash w. `--file-output` arg.
- Fixed `ldpred2.R` script in case `--file-pheno`/`--col-pheno`/`--col-pheno-from-fam` args were used, by removing these options altogether.
- Use packagemanager.rstudio.com/cran/**linux**/focal/2023-02-16 as main R package repo
- `gwas.py --variance-standardize` option now throws an error when applied to columns with no variance

11.22.4 Removed

- Removed redundant `usecases/LDpred2_tutorial` files

11.22.5 Misc

- Python code max line length of 120 chars, ignore number of newlines between functions

11.22.6 Misc

- Python code max line length of 120 chars, ignore number of newlines between functions

11.23 [1.1] - 2022-12-01

Maintenance/feature release with the following main software incorporated into each container:

container	OS/tool	version	license
hello.sif	ubuntu	20.04	Creative Commons CC-BY-SA version 3.0 UK licenc
hello.sif	plink	v1.90b6.18 64-bit (16 Jun 2020)	GPLv3
gwas.sif	ubuntu	20.04	Creative Commons CC-BY-SA version 3.0 UK licenc

continues on next pag

Table 1 – continued from previous page

container	OS/tool	version	license
gwas.sif	plink	v1.90b6.18 64-bit (16 Jun 2020)	GPLv3
gwas.sif	plink2	v2.00a3.6LM 64-bit Intel (14 Aug 2022)	GPLv3
gwas.sif	plink2_avx2	v2.00a3.6LM AVX2 Intel (24 Jan 2020)	GPLv3
gwas.sif	PRSice_linux	2.3.3 (2020-08-05)	GPLv3
gwas.sif	simu_linux	v0.9.4	GPLv3
gwas.sif	bolt	v2.4 July 22, 2022	GPLv3
gwas.sif	gcta64	version 1.93.2 beta Linux	GPLv3
gwas.sif	gctb	2.02	MIT
gwas.sif	qctool	2.0.6, revision 18b8f17	Boost
gwas.sif	king	2.2.9 - ©	permissive
gwas.sif	metal	version released on 2011-03-25	-
gwas.sif	vcftools	0.1.17	GPLv3
gwas.sif	bcftools	1.12 (using htslib 1.12)	MIT/Expat/GPLv3
gwas.sif	flashpca_x86-64	2.0	GPLv3
gwas.sif	regenie	v2.0.2.gz	MIT/Boost
gwas.sif	GWAMA	2.2.2	BSD-3-Clause
gwas.sif	minimac4	v4.1.0	GPLv3
gwas.sif	bgenix	1.1.7	Boost
gwas.sif	cat-bgen	same version as bgenix	Boost
gwas.sif	edit-bgen	same version as bgenix	Boost
gwas.sif	HTSlib	1.12	MIT/Expat/Modified-BSD
gwas.sif	shapeit4.2	v4.2.2	MIT
python3.sif	ubuntu	20.04 (LTS)	Creative Commons CC-BY-SA version 3.0 UK licenc
python3.sif	python3	python 3.10.6 + numpy, pandas, etc.	PSF
python3.sif	LDpred	1.0.11	MIT
python3.sif	python_convert	github commit bcde562	GPLv3
python3.sif	plink	v1.90b6.18 64-bit (16 Jun 2020)	GPLv3
r.sif	ubuntu	20.04	Creative Commons CC-BY-SA version 3.0 UK licenc
r.sif	R	4.0.5 (2021-03-31) + data.table, ggplot, etc.	misc
r.sif	gcta64	version 1.93.2 beta Linux	GPLv3
r.sif	PRSice_linux	2.3.3 (2020-08-05)	GPLv3
r.sif	rareGWAMA	dajiangliu/rareGWAMA@72e962d	-
r.sif	GenomicSEM	GenomicSEM/GenomicSEM@bcbbaff	GPLv3
r.sif	TwoSampleMR	MRCIEU/TwoSampleMR@c174107	unknown/MIT
r.sif	GSMR	v1.0.9	GPL>=v2
r.sif	snpStats	v1.40.0	GPLv3
saige.sif	ubuntu	16.04	Creative Commons CC-BY-SA version 3.0 UK licenc
saige.sif	SAIGE	version 0.43	GPLv3

Main changes since release version 1.0.0:

11.23.1 Added

- add option to append `usecases/LDpred2/ldpred.R` score output to an existing file
- add script `usecases/LDpred2/complementSumstats.R` to append chromosome and position to summary statistics
- add polygenic score output tests for `usecases/LDpred2/ldpred.R`
- add `usecases/LDpred2/imputeGenotypes.R` for imputing genotypes using R-package bigSNPR
- add `usecases/LDpred2/calculateLD.R` for calculation LD using R-package bigSNPR.
- add autobuilt online documentation from repository sources at https://comorment-containers.readthedocs.io/en/latest/
- add R libraries for LDpred2 analysis to `r.sif` + corresponding example.
- add tests for `metal` and `qctool` in `gwas.sif` build
- add basic GitHub actions from https://github.com/precimed/container_template.git
- add `FaST-LMM` (version 0.6.3) to future `python3.sif`, and corresponding test
- add `shapeit4.2` binary (shapeit4 v.4.2.2) and HTSlib (1.11) to future `gwas.sif` builds, and corresponding test
- added additional tests for software in `gwas.sif`, `python3.sif` builds
- add versions identifiers for all explicitly installed software across `hello.sif`, `gwas.sif`, `python3.sif`, `r.sif`, listed in *docker/README.md*
- replaced Ubuntu 18.04 with 20.04 (LTS) as base image for `hello.sif`, `gwas.sif`, `python3.sif`
- replaced `src/scripts/install_miniconda3.sh` by `scr/scripts/install_mambaforge.sh` which is now used in future `python3.sif` builds
- add tests for bgenix and Minimac4 software in `gwas.sif`, removing build-time dependencies for these from container
- add basic test that KING software runs in `gwas.sif`
- add Dockerfiles and install scripts for `gwas.sif`, `hello.sif`, `python3.sif`, `r.sif`, `saige.sif` from gwas.
- add CHANGELOG.md (this file)
- add `gwas.py --analysis saige` option, allowing to run SAIGE analysis
- add `gwas.py --analysis figures` option, using R qqman for QQ and manhattan plots
- add `gwas.py --pheno-sep` and `--dict-sep` options to specify delimiter for the phenotype file and phenotype dictionary file
- add package `qqman` to `r.sif`
- add package `yaml` to `python3.sif`
- add `gctb_2.0_tutorial.zip` reference files under `reference/examples/gctb_2.0_tutorial`
- add `config.yaml` file with configuration options, which can be specified via `gwas.py --config` option
- add `--chunk-size-bp` and `--bim` option, allowing to run SAIGE analysis in smaller chunks
- add `--keep` and `--remove` options to `gwas.py`, allowing to keep and remove subsets of individuals from analysis; the functions work similarly to plink2 as described here.

11.23.2 Updated

- rebuilt the following containers following version pinning in Dockerfiles, install scripts, etc. (see above additions):

```
bb7a8e0b977e29e03067d75d19803913  singularity/gwas.sif
11ac9e8fe69df07d650bd5e1e7cdeee5  singularity/hello.sif
c78d57397471ee802d37837ca5f8b797  singularity/python3.sif
e8f26b23a8b44f15f3dfff2b02623780  singularity/r.sif
a3f1d8411e1e3cf8670551b7f334a58d  singularity/saige.sif
```

11.23.3 Fixed

- `usecases/LDpred2/ldpred2.R` error when sumstats contain characters in chromosome column.
- use `afterok` spec instead of `afterany` in SLURM dependencies so that next steps of the pipeline don't run if a previous step has failed (fix #26)
- use SLURM's `cpus_per_task=1` for SAIGE step2, because it doesn't support –nThreads (see https://github.com/saigegit/SAIGE/issues/9)

11.23.4 Removed

- removed `--geno-impute` from `usecases/LDpred2/ldpred2.R`. Functionality replaced by `--geno-impute-zero` and `usecases/LDpred2/imputeGenotypes.R`
- removed misc. source/data files in /tools/* from container builds
- removed unused `libquadmath0` library from builds (affecting future `gwas.sif`, `hello.sif`, and `python3.sif` builds)
- the following command-line options are removed; instead, they can be specified via `config.yaml` file: `--slurm-job-name`, `--slurm-account`, `--slurm-time`, `--slurm-cpus-per-task`, `--slurm-mem-per-cpu`, `--module-load`, `--comorment-folder`, `--singularity-bind`. Note that `config.yaml` file is now required.
- `gwas.py --analysis loci manh qq` options as removed (fix #22)
- `--bed-fit`, `--bed-test`, `--bgen-fit`, `--bgen-test` options of `gwas.py` are removed; use new options `--geno-fit-file` and `--geno-file` instead
- remove `regenie.sif` and `regenie3.sif`, because regenie software is also included in `gwas.sif`
- remove MiXeR package from `python3.sif` container, because MiXeR is now available as a separate container (https://github.com/comorment/mixer). This is also where you will find MiXeR's use-cases.
- MAGMA, LAVA and ldblock software is moved to https://github.com/comorment/magma. MAGMA reference files are also moved to this repository.
- enigma-cnv.sif and enigma-cnv.sif is moved to https://github.com/comorment/iPsychCNV enigma-cnv.sif is also available here: in https://github.com/ENIGMA-git/ENIGMA-CNV/tree/main/CNVCalling/containers
- tryggve_query.sif is moved to https://github.com/comorment/Tryggve_psych
- `matlabruntime.sif` container is moved to https://github.com/comorment/matlabruntime. pleioFDR reference files are also moved to this repository.

11.24 [1.0.0] - 2020-10-20

11.24.1 Added

- initial release of the following containers:

```
70502c11d662218181ac79a846a0937a  enigma-cnv.sif
1ddd2831fcab99371a0ff61a8b2b0970  gwas.sif
b02fe60c087ea83aaf1b5f8c14e71bdf  hello.sif
1ab5d82cf9d03ee770b4539bda44a5ba  ipsychcnv.sif
6d024aed591d8612e1cc628f97d889cc  ldsc.sif
2e638d1acb584b42c6bab569676a92f8  matlabruntime.sif
331688fb4fb386aadaee90f443b50f8c  python3.sif
cdbfbddc9e5827ad9ef2ad8d346e6b82  r.sif
b8c1727227dc07e3006c0c8070f4e22e  regenie.sif
97f75a45a39f0a2b3d728f0b8e85a401  regenie3.sif
20e01618bfb4b0825ef8246c5a63aec5  saige.sif
5de579f750fb5633753bfda549822a32  tryggve_query.sif
```

Here is the list of tools available in prebuilt containers:

container	tool	version
hello.sif	demo example	
gwas.sif	plink	v1.90b6.18 64-bit (16 Jun 2020)
gwas.sif	plink2	v2.00a2.3LM 64-bit Intel (24 Jan 2020)
gwas.sif	plink2_avx2	v2.00a2.3LM AVX2 Intel (24 Jan 2020)
gwas.sif	PRSice_linux	2.3.3 (2020-08-05)
gwas.sif	simu_linux	Version v0.9.4
gwas.sif	bolt	v2.3.5 March 20, 2021
gwas.sif	gcta64	version 1.93.2 beta Linux
gwas.sif	gctb	GCTB 2.02
gwas.sif	qctool	version: 2.0.6, revision 18b8f17
gwas.sif	king	KING 2.2.6 - ©
gwas.sif	metal	version released on 2011-03-25
gwas.sif	vcftools	VCFtools (0.1.17)
gwas.sif	bcftools	Version: 1.12 (using htslib 1.12)
gwas.sif	flashpca_x86-64	flashpca 2.0
gwas.sif	regenie	REGENIE v2.0.2.gz
gwas.sif	GWAMA	GWAMA_v2.2.2.zip
gwas.sif	magma	magma_v1.09a_static.zip
gwas.sif	shapeit2	Version : v2.r904
gwas.sif	impute4	impute4.1.2_r300.3
gwas.sif	minimac4	Version: 1.0.2; Built: Fri Sep 3 13:25:51
gwas.sif	bgenix	version: 1.1.7, revision
gwas.sif	cat-bgen	same version as bgenix
gwas.sif	edit-bgen	same version as bgenix
python3.sif	python3	python 3.10 + standard packages (numpy, pandas, etc)
python3.sif	ldpred	?
python3.sif	mixer	mixer v1.3
python3.sif	python_convert	github commit bcde562f0286f3ff271dbb54d486d4ca1d40ae36
r.sif	R	version 4.0.3 + standard packages (data.table, ggplot, etc)

continues on next page

Table 2 – continued from previous page

container	tool	version
r.sif	seqminer	?
r.sif	rareGWAMA	?
r.sif	GenomicSEM	?
r.sif	TwoSampleMR	?
r.sif	GSMR	v1.0.9
r.sif	LAVA	?
r.sif	LAVA partitioning	?
saige.sif	SAIGE	version 0.43
enigma-cnv.sif	PennCNV	version 1.0.5
ldsc.sif	LDSC	version 1.0.1
ipsychcnv.sif	???	missing Dockerfile
matlabruntime.sif	???	work in progress
regenie.sif	???	?
regenie3.sif	???	?

CHAPTER

TWELVE

INTERNAL USAGE

Miscellaneous notes for internal usage.

12.1 Docker

Build recipes for containers using Docker and Singularity.

12.1.1 Software versions

Please confer and update accordingly the software version tables in the respective *singularity* files for each container.

12.1.2 Feedback

If you face any issues, or if you need additional software, please let us know by creating an issue.

12.1.3 Note about NREC machine

We use NREC machine to develop and build containers. NREC machine has small local disk (~20 TB) and a larger external volume attached (~400 TB) If you use NREC machine, it's important to not store large data or install large software to your home folder which is located on a small disk, using `/nrec/projects space` instead:

```
Filesystem                          Size  Used Avail Use% Mounted on
/dev/sda1                            20G  9.6G  9.7G  50% /
/dev/mapper/nrec_extvol-comorment   393G  346G   28G  93% /nrec/projects
/dev/mapper/nrec_extvol_2-comorment_2  935G  609G  279G  69% /nrec/space
```

Both docker and singularity were configured to avoid placing cached files into local file system. For docker this involves changing `/etc/docker/daemon.json` file by adding this:

```
{
    "data-root": "/nrec/projects/docker_root"
}
```

(as described https://tienbm90.medium.com/how-to-change-docker-root-data-directory-89a39be1a70b ; you may use `docker info` command to check the data-root)

For singularity, the configuration is described here https://sylabs.io/guides/3.6/user-guide/build_env.html and it was done for the root user by adding the following line into /etc/environment

```
export SINGULARITY_CACHEDIR="/nrec/projects/singularity_cache"
```

Common software, such as git-lfs, is installed to /nrec/projects/bin. Therefore it's reasonable for all users of the NREC comorment instance to add this folder to the path by changing `~/.bashrc` and `~/.bash_profile`.

```
export PATH="/nrec/projects/bin:$PATH"
```

A cloned version of comorment repositories is available here:

```
/nrec/projects/github/comorment/containers
/nrec/projects/github/comorment/reference
```

Feel free to change these folders and use git pull / git push. TBD: currently the folder is cloned as 'ofrei' user - I'm not sure if it will actually work to pull & push. But let's figure this out.

12.1.4 Testing container builds

Some basic checks for the functionality of the different container builds are provided in `<containers>/tests/`, implemented in Python. The tests can be executed using the Pytest testing framework.

To install Pytest in the current Python environment, issue:

```
pip install pytest  # --user optional
```

New virtual environment using conda:

```
conda create -n pytest python=3 pytest -y  # creates env "pytest"
conda activate pytest  # activates env "pytest"
```

Then, all checks can be executed by issuing:

```
cd <containers>
py.test -v tests  # with verbose output
```

Checks for individual containers (e.g., `gwas.sif`) can be executed by issuing:

```
py.test -v tests/test_<container-prefix>.py
```

Note that the proper container files (*.sif files) corresponding to the different test scripts must exist in `<containers>/singularity/>`, not only git LFS pointer files.

12.1.5 Git clone ignoring LFS

See stackoverflow.com/questions/42019529/how-to-clone-pull-a-git-repository-ignoring-lfs

```
GIT_LFS_SKIP_SMUDGE=1 git clone git@github.com:comorment/containers.git
```

CHAPTER

THIRTEEN

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

INDEX