Association Testing Draft : `pheno_qt1`, `pheno_qt2`

H3Agwas Association Testing Pipeline

Tue Jul 12 09:15:27 SAST 2022

1 Introduction

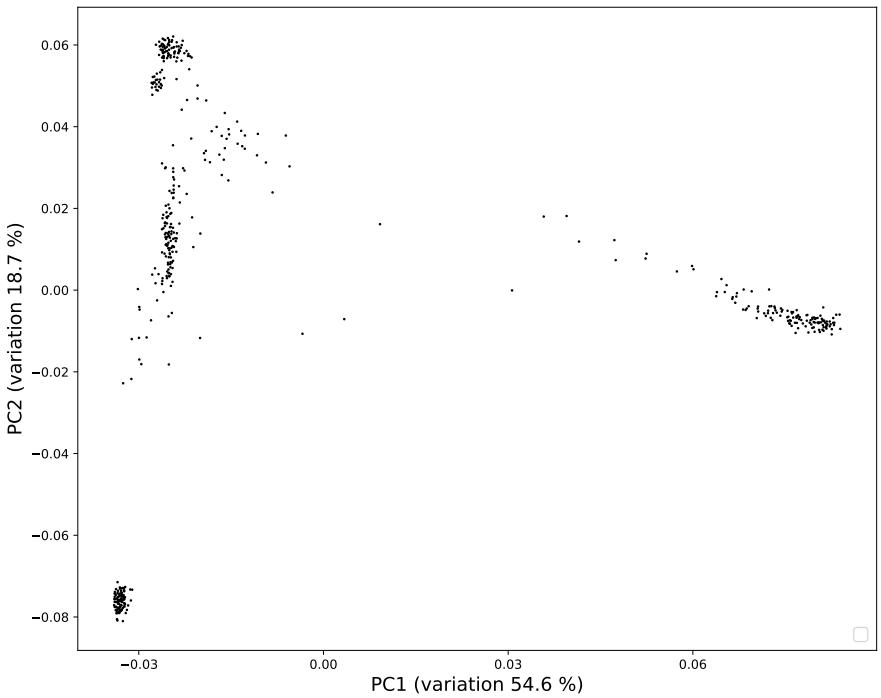
This report gives a brief overview of the run of the association testing pipeline.

- You were testing for the following phenotypes `pheno_qt1`, `pheno_qt2`
- You were using the following covariates []

2 Principal Component Analysis of Participants

Figure 1 shows a PCA of the participants. This should be examined for possible structure.

Figure 1 PCA of participants

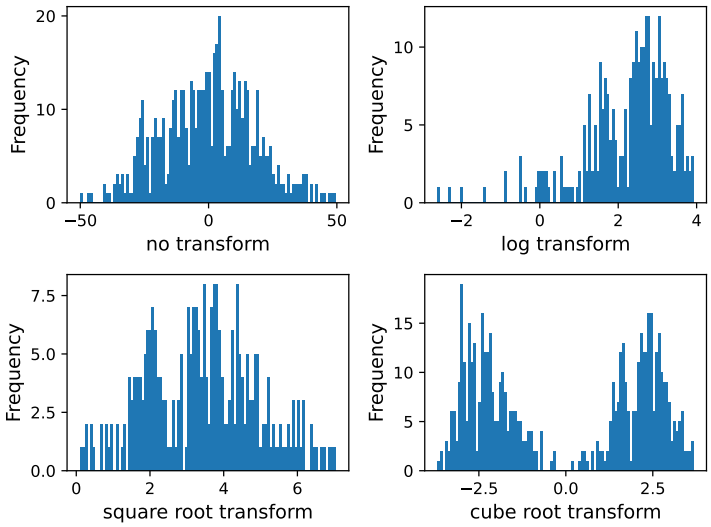


A summary of the data for `pheno_qt1` can be found in the Table 1, transformed using different transforms. A histogram is found in Figure 2.

Table 1 Overview of phenotype `pheno_qt1` distribution

Data	Count	Min	Max	Ave	StdDev
no transform	550	$-5.0158E+00$	4.9065	$-1.0720E-01$	1.7198
log transform	292	$-2.6090E+00$	3.3992	2.25	1.10
square root transform	279	$9.7775E-02$	7.0124	3.44	1.49
cube root transform	550	$-3.6879E+00$	3.3667	$1.3392E-02$	1.2235

Figure 2 Histogram of `pheno_qt1` values under different transforms [File is `B050-pheno-qt1.pdf`]

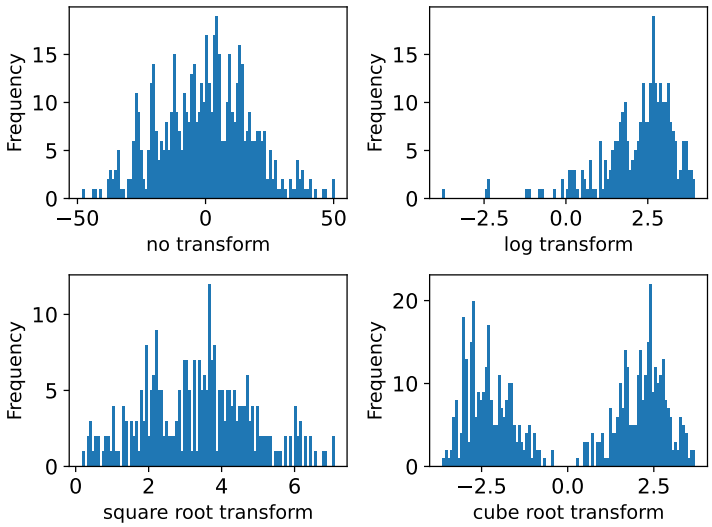


A summary of the data for `pheno_qt2` can be found in the Table 2, transformed using different transforms. A histogram is found in Figure 3.

Table 2 Overview of phenotype `pheno_qt2` distribution

Data	Count	Min	Max	Ave	StdDev
no transform	550	$-4.8290E+00$	5.0031	$-2.1362E-01$	1.7172
log transform	297	$-3.7771E+00$	3.3993	2.21	1.13
square root transform	286	0.16	7.09	3.35	1.48
cube root transform	550	$-3.6416E+00$	3.3669	$3.3940E-02$	1.2234

Figure 3 Histogram of `pheno_qt2` values under different transforms [File is `B050-pheno-qt2.pdf`]



3 PLINK Results for phenotype pheno-qt1, test assoc

The raw results can be found in the **assoc** directory. The QQ plot can be found in Figure 4, and the Manhatten plot in Figure 5.

- No correction

Figure 4 QQ plot for PLINK testing – pheno-qt1, test assoc – pheno_qt1.qassoc

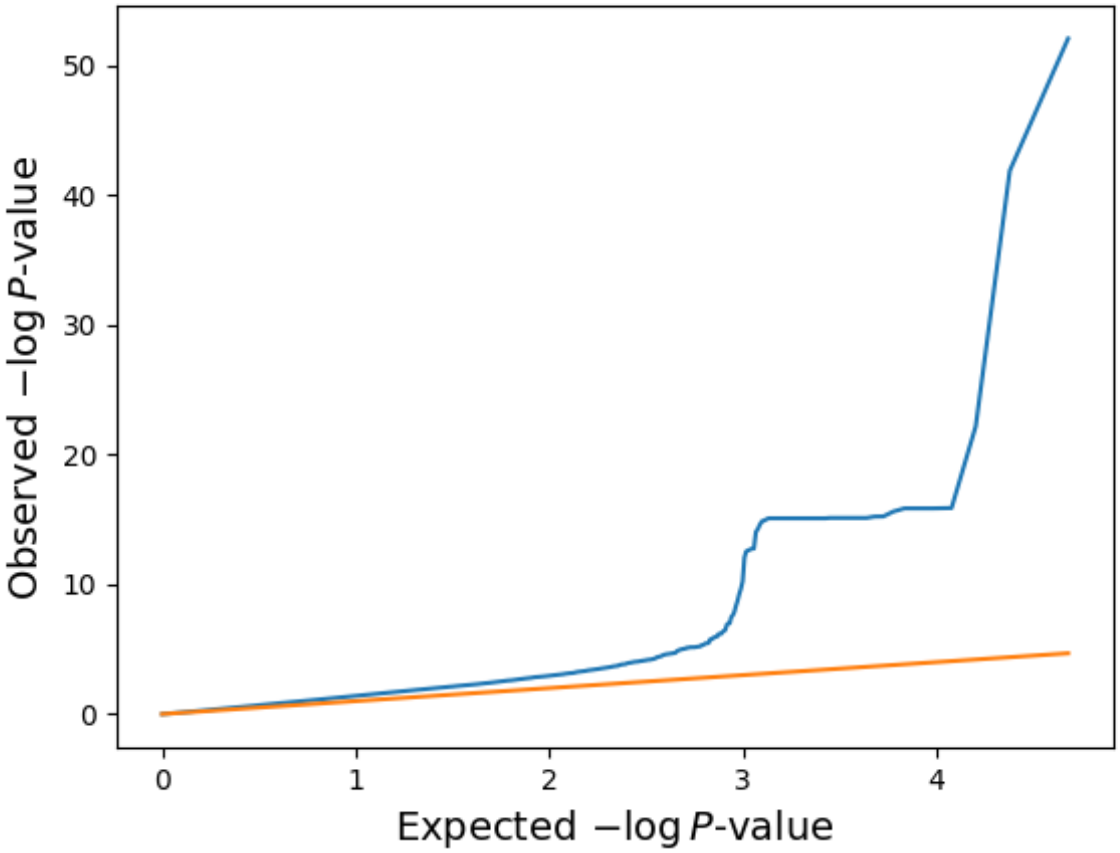
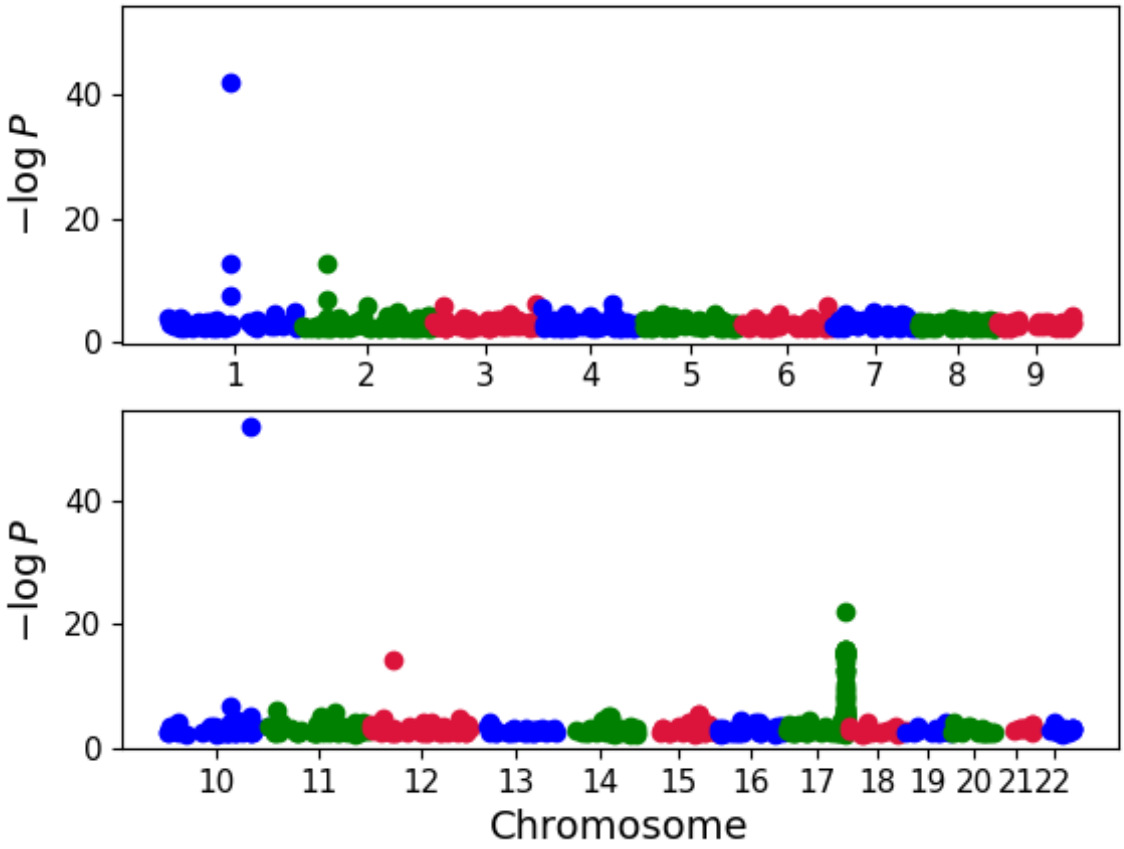


Figure 5 PLINK testing: Manhatten plot for – `pheno-qt1`, test assoc – `pheno_qt1.qassoc`



4 PLINK Results for phenotype pheno-qt1, test linear

The raw results can be found in the **linear** directory. The QQ plot can be found in Figure 6, and the Manhatten plot in Figure 7.

- No correction

Figure 6 QQ plot for PLINK testing – pheno-qt1, test linear – pheno_qt1.assoc.linear

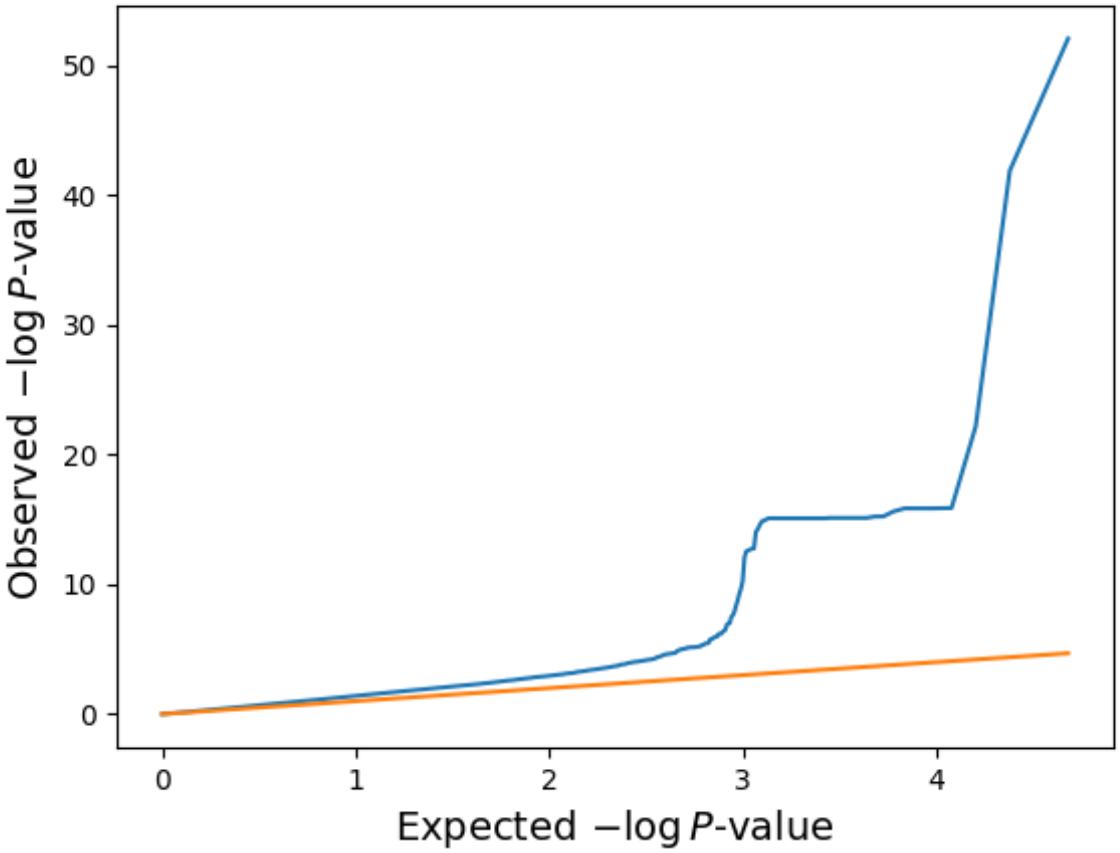
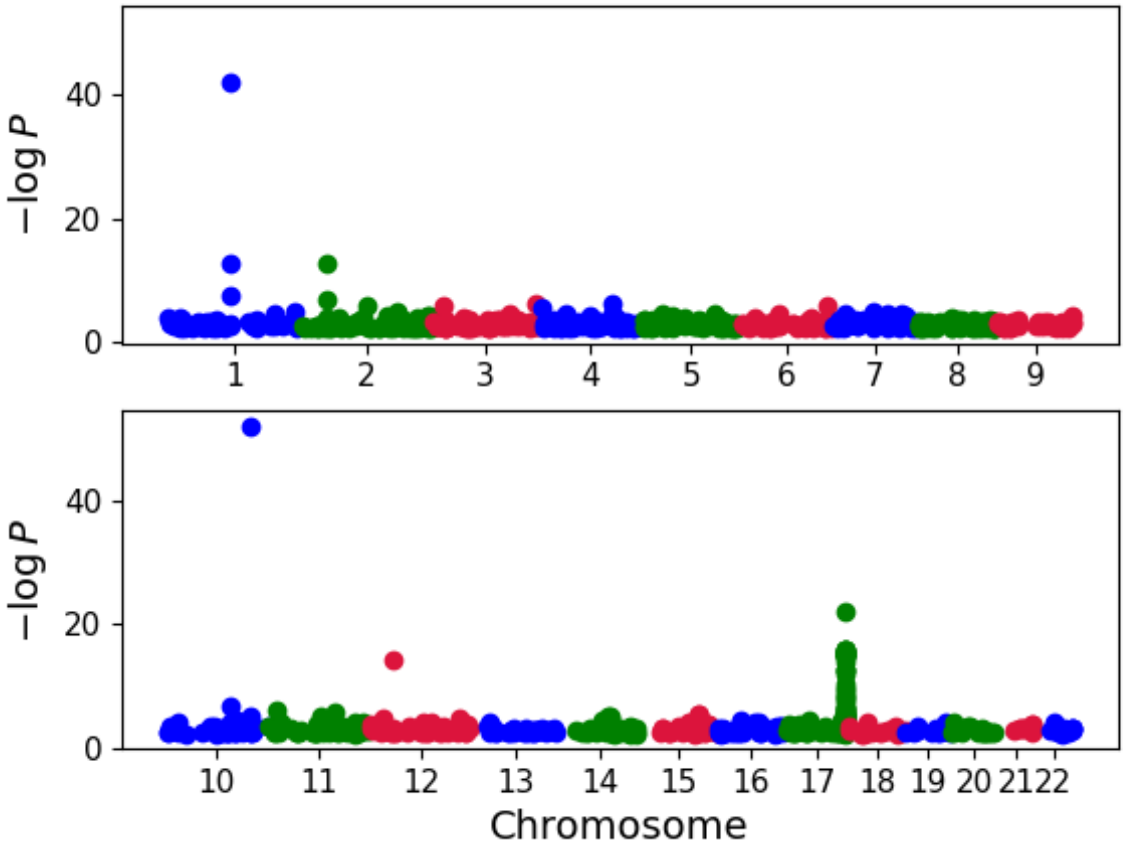


Figure 7 PLINK testing: Manhatten plot for – `pheno-qt1`, test linear – `pheno_qt1.assoc.linear`



5 PLINK Results for phenotype pheno-qt2, test assoc

The raw results can be found in the **assoc** directory. The QQ plot can be found in Figure 8, and the Manhatten plot in Figure 9.

- No correction

Figure 8 QQ plot for PLINK testing – pheno-qt2, test assoc – pheno_qt2.qassoc

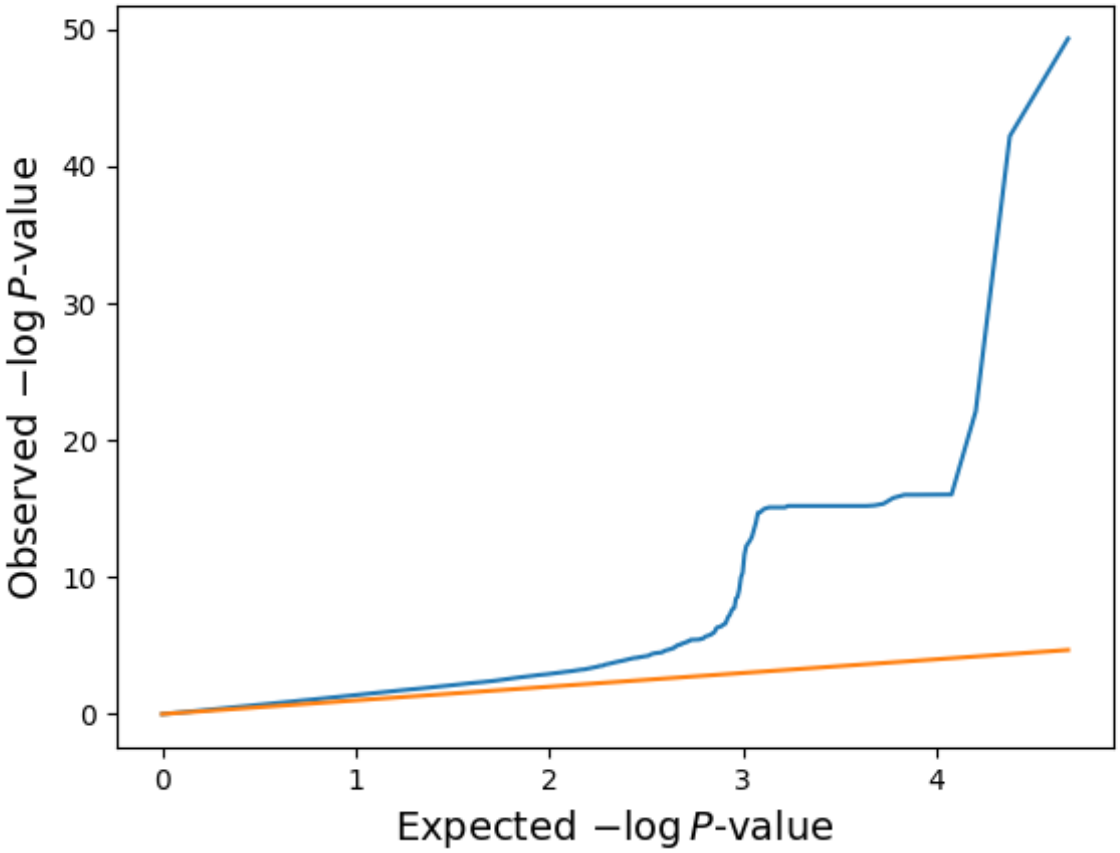
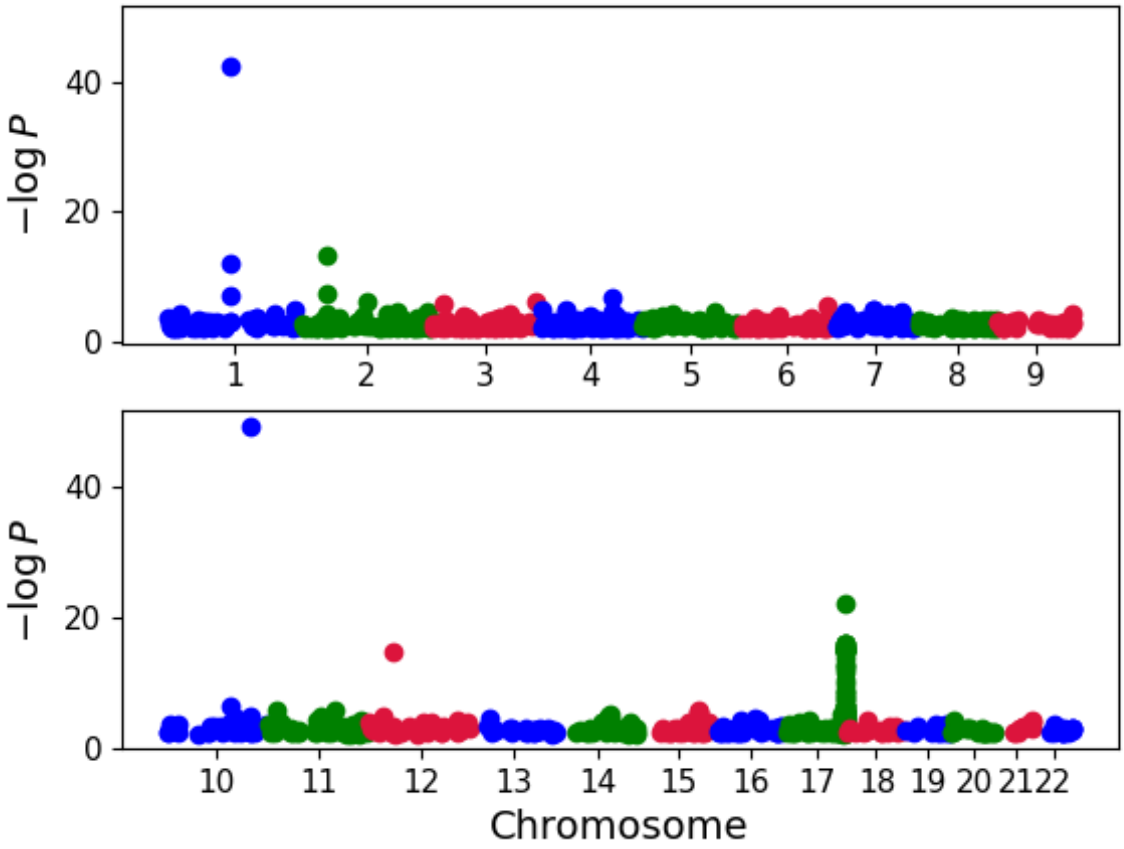


Figure 9 PLINK testing: Manhatten plot for – `pheno-qt2`, test assoc – `pheno_qt2.qassoc`



6 PLINK Results for phenotype pheno-qt2, test linear

The raw results can be found in the **linear** directory. The QQ plot can be found in Figure 10, and the Manhatten plot in Figure 11.

- No correction

Figure 10 QQ plot for PLINK testing – pheno-qt2, test linear – pheno_qt2.assoc.linear

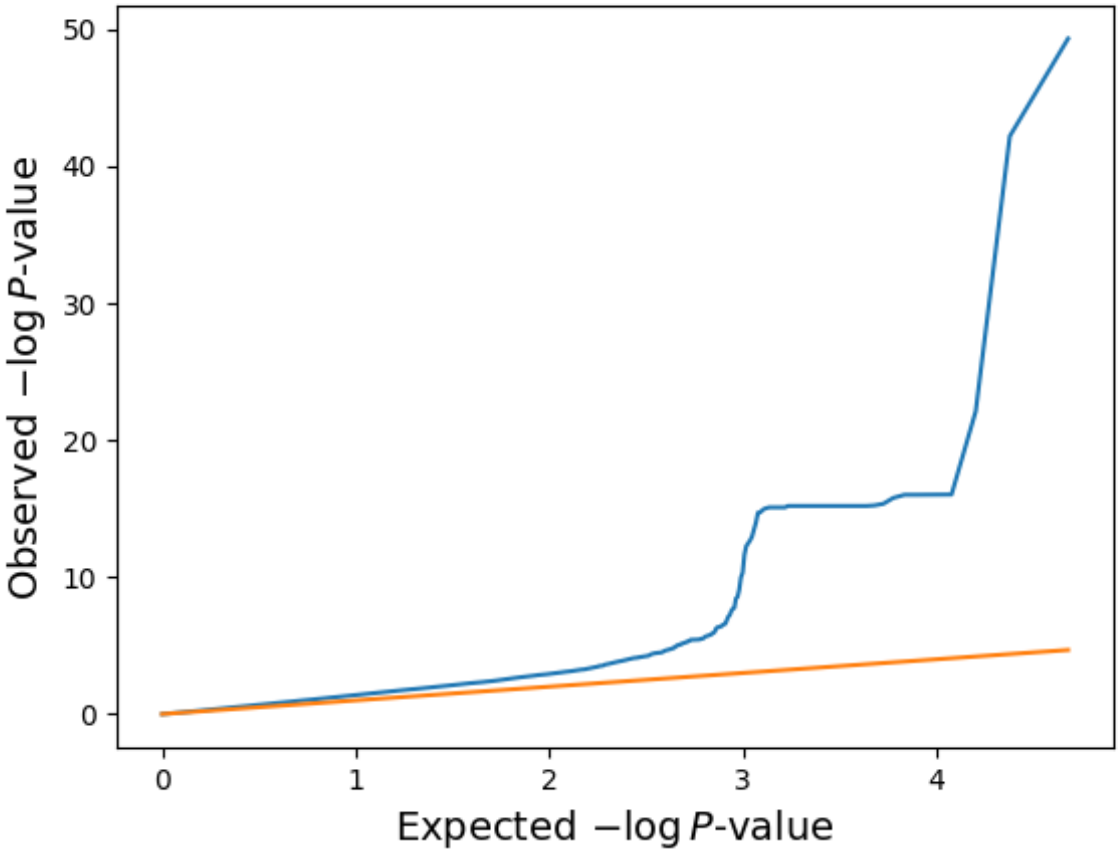


Figure 11 PLINK testing: Manhatten plot for – `pheno-qt2`, test linear – `pheno_qt2.assoc.linear`

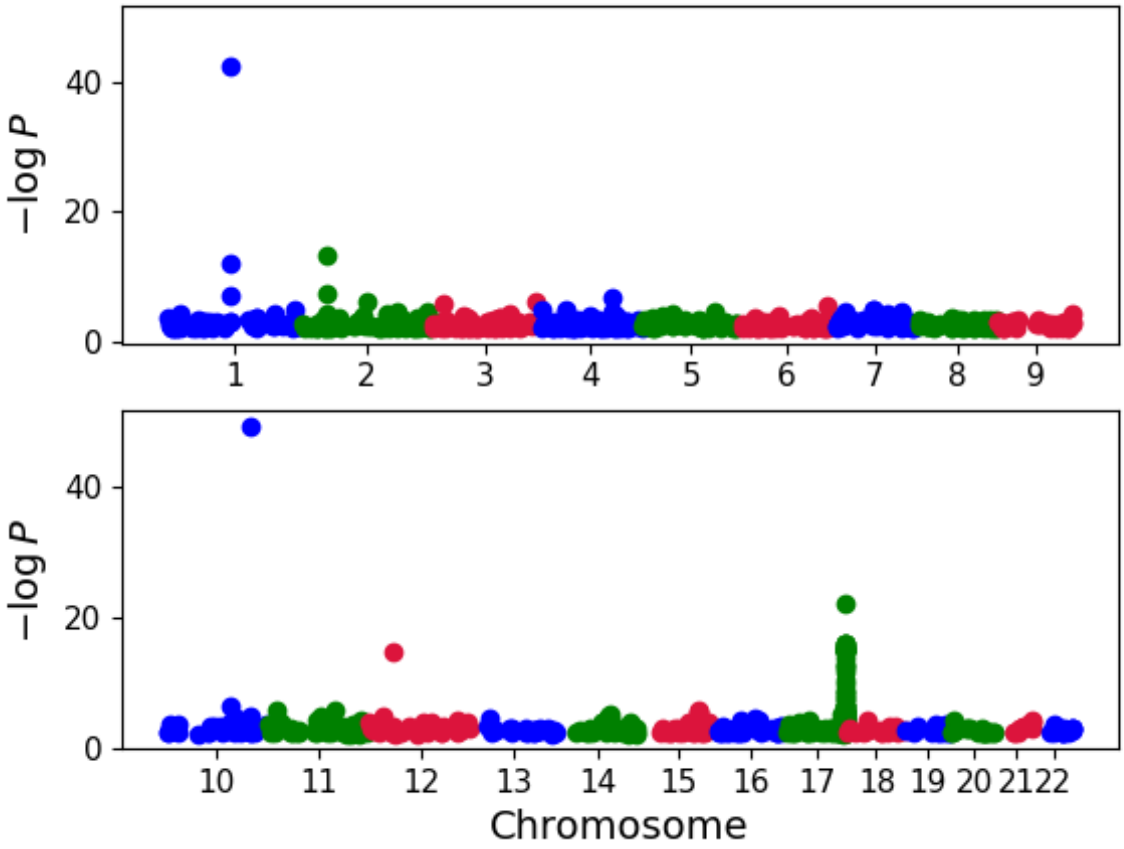


Table 3 The top 10 SNPs found by GEMMA analysis of phenotype `pheno-qt1`

Chr	SNP	Pos	Beta	P
10	10:112678657:G:T	112678657	-15.7000	1.974E-48
1	1:117532790:C:T	117532790	-14.9724	1.776E-45
17	17:78757626:A:G	78757626	12.3403	3.100E-21
12	12:31367856:A:C	31367856	9.4800	7.856E-17
2	2:45832137:A:G	45832137	13.3723	6.954E-15
17	17:78727734:T:C	78727734	10.8048	4.522E-14
17	17:78728813:G:A	78728813	10.6729	8.474E-14
17	17:78716122:G:A	78716122	10.3920	2.107E-13
17	17:78716417:A:G	78716417	10.3920	2.107E-13
17	17:78714793:C:T	78714793	10.3920	2.107E-13

7 Result of Gemma analysis : phenotype pheno-qt1

All the results from the GEMMA analysis can be found in the **gemma** directory. The result of the GEMMA analysis is shown for phentoytype `pheno-qt1`. The file with association statistics is found in `imput_data-pheno-qt1.assoc.txt`. The top 10 results are shown in Table 3:
The Manhatten plot can be found in Figure 12. The corresponding QQ-plot can be found in Figure 13.

Figure 12 Gemma testing: Manhatten plot for phenotype pheno-qt1

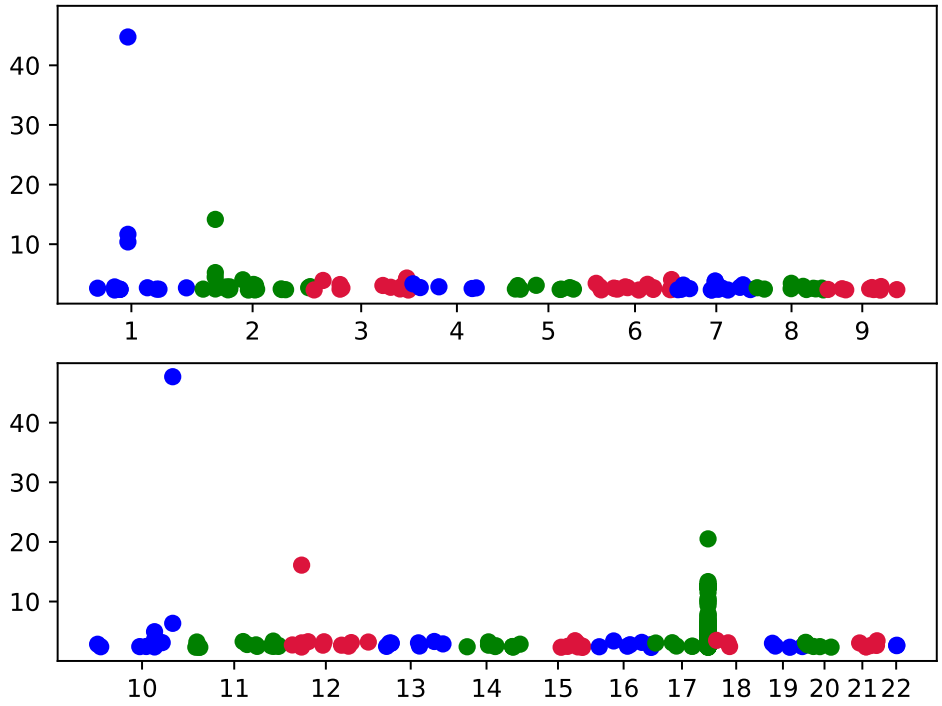


Figure 13 Gemma testing: QQ-plot for phenotype pheno-qt1

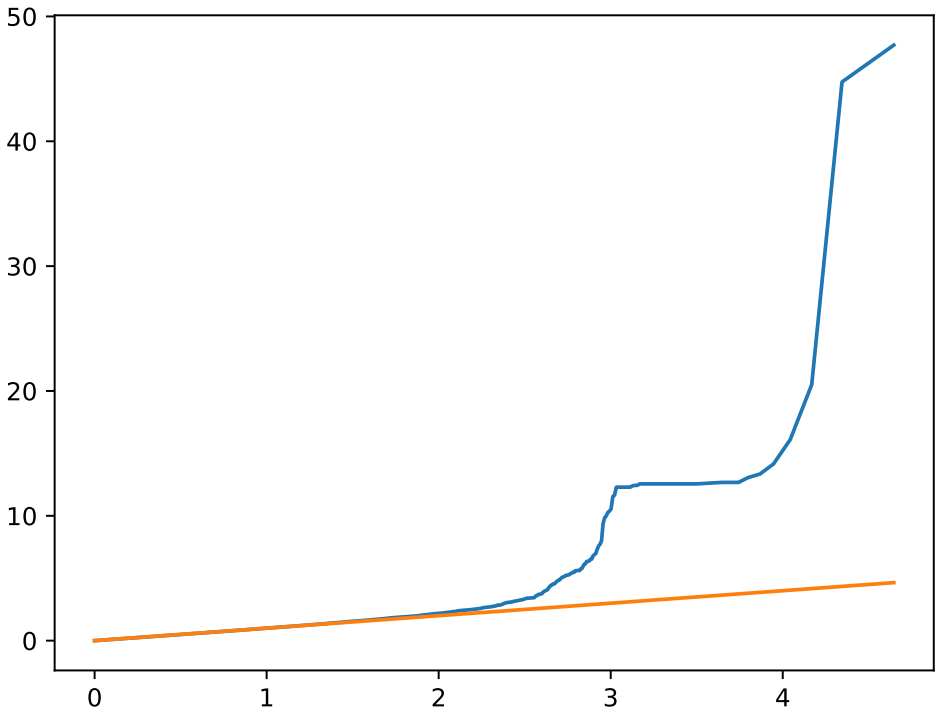


Table 4 The top 10 SNPs found by GEMMA analysis of phenotype `pheno-qt2`

Chr	SNP	Pos	Beta	P
1	1:117532790:C:T	117532790	-14.9551	3.122E-46
10	10:112678657:G:T	112678657	-15.1765	1.732E-45
17	17:78757626:A:G	78757626	12.2282	3.115E-21
12	12:31367856:A:C	31367856	9.5014	3.502E-17
2	2:45832137:A:G	45832137	13.5996	1.276E-15
17	17:78727734:T:C	78727734	10.8065	2.504E-14
17	17:78728813:G:A	78728813	10.6795	4.655E-14
17	17:78716122:G:A	78716122	10.3850	1.273E-13
17	17:78716417:A:G	78716417	10.3850	1.273E-13
17	17:78714793:C:T	78714793	10.3850	1.273E-13

8 Result of Gemma analysis : phenotype pheno-qt2

All the results from the GEMMA analysis can be found in the **gemma** directory. The result of the GEMMA analysis is shown for phentoytype `pheno-qt2`. The file with association statistics is found in `imput_data-pheno-qt2.assoc.txt`. The top 10 results are shown in Table 4:
The Manhatten plot can be found in Figure 14. The corresponding QQ-plot can be found in Figure 15.

Figure 14 Gemma testing: Manhatten plot for phenotype pheno-qt2

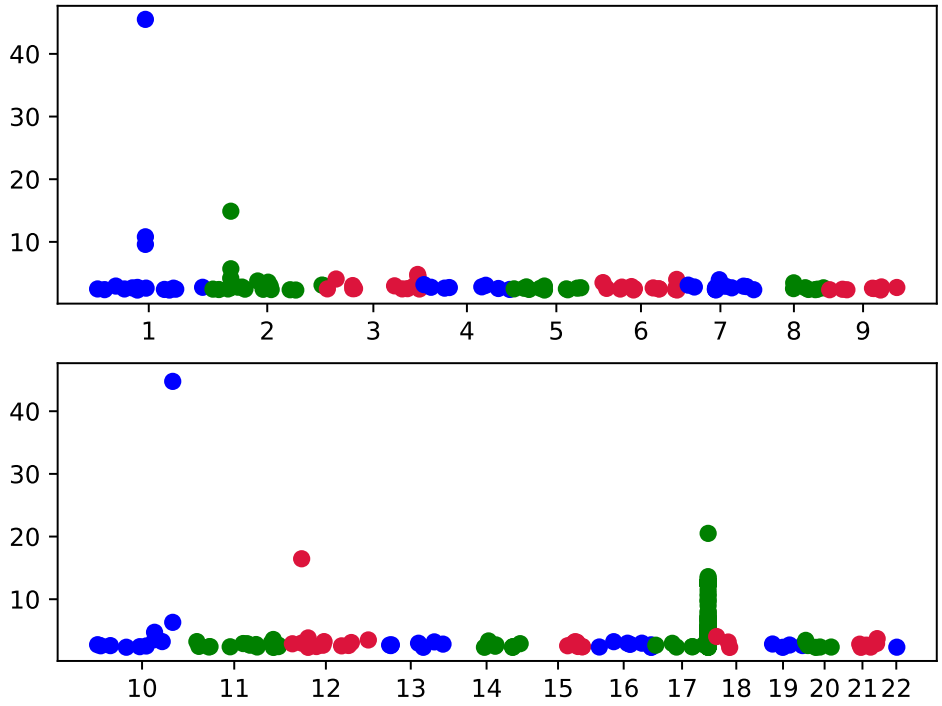


Figure 15 Gemma testing: QQ-plot for phenotype pheno-qt2

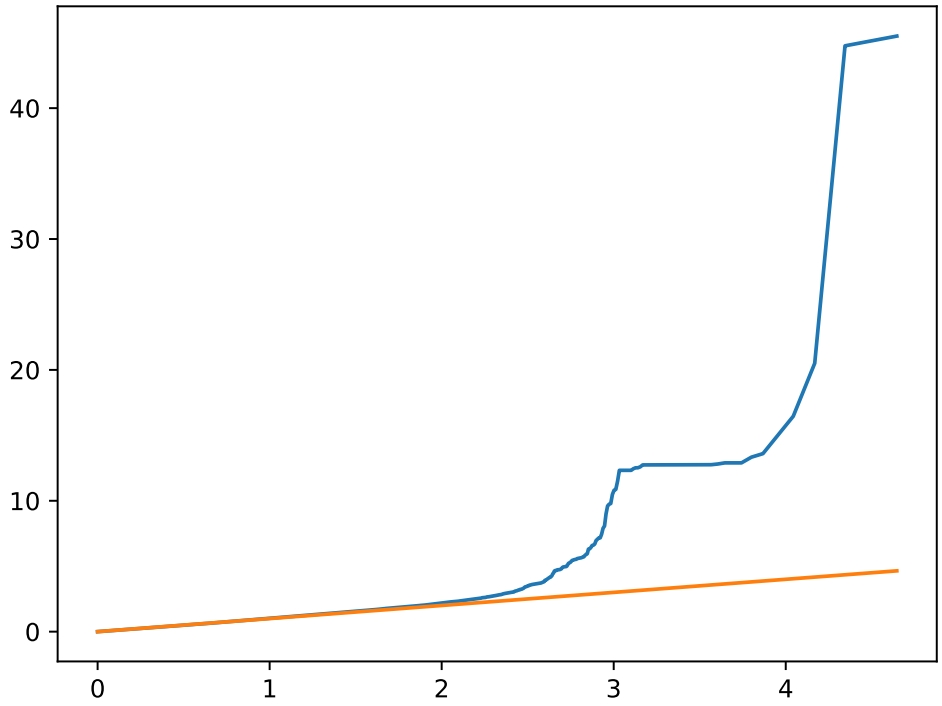


Table 5 Docker Images Used

Nextflow process	Docker Image
default	quay.io/h3abionet_org/py3plink

9 Technical details

The analysis and report was produced by the h3aGWAS pipeline (http://github.com/h3abionet/h3agwas) produced by the Pan-African Bioinformatics Network for H3Africa (http://www.h3abionet.org).

The following tools were used:

- 22.04.3 [Di Tommaso et al, 2017]
- A local copy of the workflow was used
- The command line below was called [NB: if the command line is long, the linebreak may break oddly after a hyphen or dash so take care.]

```
nextflow run /home/jeantristan/Travail/git/h3agwas/assoc --input_dir data/imputed/ --
    input_pat imput_data --data data/pheno/pheno_test.all --pheno pheno_qt1,pheno_qt2
    --output_dir assoc --output assoc --gemma 1 --assoc 1 --sample_snps_rel 1 --linear
    1 -profile slurmSingularity --bgen data/imputed/bgen/out.bgen --bgen_sample data/
    imputed/bgen/out.sample
```

- The profile slurmSingularity was used: the docker images used are found in Table 5
- The full configuration can be found in the appendix.

A Configuration

The Nextflow configuration files are shown below.

A.1 config

```
singularity.cacheDir = "/home/jeantristan/.singularity/"
```

A.2 nextflow

```
plugins {
  id 'nf-azure'
}



py3Image = "quay.io/h3abionet_org/py3plink"
gemmaImage="quay.io/h3abionet_org/py3plink"
boltImage="quay.io/h3abionet_org/py3fastlmm"
latexImage="quay.io/h3abionet_org/h3agwas-texlive"
//py3saige="quay.io/h3abionet_org/py3saige"
py3saige="wzhou88/saige:1.1.2"
gctaImage="quay.io/h3abionet_org/gcta"
MetaAnImage="quay.io/h3abionet_org/py3metagwas"
fastlmmImage="quay.io/h3abionet_org/py3fastlmm"
rImage="quay.io/h3abionet_org/py3rproject"
py3utils="quay.io/h3abionet_org/py3utils"
swarmPort = '2376'
queue = 'batch'

manifest {
    homePage = 'http://github.com/h3abionet/h3agwas'
    description = 'GWAS Pipeline for H3Africa'
    mainScript = 'main.nf'
}


aws {
accessKey='*******'
secretKey='*******'
    region    ='eu-west-1'
}

cloud {
    imageId = "ami-710b9108"      // specify your AMI id here
    instanceType = "m4.xlarge"
    subnetId = "null"
    sharedStorageId   = "null"
    sharedStorageMount = "/mnt/shared"
    bootStorageSize = "20GB"     // Size of disk for images spawned
    //   instanceStorageMount = ""   // Set a common mount point for images
    //   instanceStorageDevice = ""  // Set a common block device for images
    autoscale {
        enabled = true
        maxInstances = 1
        terminateWhenIdle = true
    }

}


params {

    // Directories
    work_dir                = "/$PWD"
    input_dir               = ""
```

```
    output_dir              = "${params.work_dir}/output"
    scripts                 = "${params.work_dir}/scripts"
    output                  = "out"


    max_forks            = 95
    // Data
    input_pat            = ""


    high_ld_regions_fname = ""
    sexinfo_available    = true
    cut_het_high         = 0.343
    cut_het_low           = 0.15
    cut_diff_miss         = "0.05"
    cut_maf                = "0.01"
    cut_mind              = "0.02"
    cut_geno              = 0.01
    cut_hwe               = 0.008
    case_control         = "${params.input_dir}/sample.phe"
    case_control_col     = "PHE"


    phenotype = 0
    pheno_col = "all"
    batch = 0
    batch_col = 0

    samplesize           = 0  // How many individuals in each genotype report 0=ALL
    strandreport         =""
    manifest             =  ""
    idpat                =  0  // or "(\w+)-DNA_(\w+)_.*" or ".*_(.*)"


    // Needed for topbottom.nf -- uncomment and put in details
    // reference        = ""
    // output_align     = ""
    // samplesheet      = ""
    // chipdescription  = ""

accessKey='*******'
secretKey='*******'
    region               = "eu-west-1"
    AMI                  = "ami-710b9108"
    instanceType         = "m4.xlarge"
    bootStorageSize      = "20GB"
    maxInstances         = "5"

    plink_mem_req        = "750MB"
    other_mem_req        = "750MB"
    big_time             = '1000h'
    sharedStorageMount   = "/mnt/shared"
    max_plink_cores      = 4


}
profiles {

    // For execution on a local machine, no containerization. -- Default
    standard {
        process.executor = 'local'
    }

    slurm {
       process.executor = 'slurm'
       process.queue = queue
```

```
}

awsbatch {
     process.executor = "awsbatch"
     aws.region    =’us-east-1’
     aws.uploadStorageClass = ’ONEZONE_IA’
     process.queue = ’h3a’
}




// Execute pipeline with Docker locally
docker {
    process.executor = ’local’
    docker.remove      = true
    docker.runOptions  = ’--rm’
    docker.registry    = ’quay.io’
    docker.enabled     = true
    docker.temp        = ’auto’
    docker.fixOwnership= true
    docker.process.executor = ’local’
}


// Execute pipeline with Docker Swarm setup
dockerSwarm {
    docker.remove      = true
    docker.runOptions  = ’--rm’
    docker.registry    = ’quay.io’
    docker.enabled     = true
    docker.temp        = ’auto’
    docker.fixOwnership= true
    docker.process.executor = ’local’
    docker.engineOptions = "-H :$swarmPort"
}

// For execution on a PBS scheduler, no containerization.
pbs {
    process.executor = ’pbs’
    process.queue = queue
}


// For execution on a PBS scheduler with containerization.
pbsDocker {

    process.executor = ’pbs’
    docker.remove      = true
    docker.runOptions  = ’--rm’
    docker.registry    = ’quay.io’
    docker.enabled     = true
    docker.temp        = ’auto’
    docker.fixOwnership= true
}


// For execution on a SLURM scheduler, no containerization.
slurm {
    process.executor = ’slurm’
    process.queue = queue
}


// For execution on a PBS scheduler with containerisation.
slurmDocker {
```

```
        process.executor = 'slurm'
        docker.remove      = true
        docker.runOptions  = '--rm'
        docker.registry    = 'quay.io'
        docker.enabled     = true
        docker.temp        = 'auto'
        docker.fixOwnership= true
    }



    singularity {
        singularity.cacheDir = "${HOME}/.singularity"
        singularity.autoMounts = true
        singularity.enabled = true
        process.executor = 'local'
     }

    // For execution on a SLURM scheduler with singularity
    slurmSingularity {
        singularity.cacheDir = "${HOME}/.singularity"
        process.executor = 'slurm'
        singularity.autoMounts = true
        singularity.enabled = true
        singularity.runOption = "--cleanenv"
        process.queue = queue

    }

    slurmDocker {
        process.executor = 'slurm'
        process.queue = queue
        docker.remove      = true
        docker.runOptions  = '--rm'
        docker.registry    = 'quay.io'
        docker.enabled     = true
        docker.temp        = 'auto'
        docker.fixOwnership= true

    }
    // For execution on a PBS scheduler, no containerization.
    pbs {
        process.executor = 'pbs'
        process.queue = queue
    }


    // For execution on a PBS scheduler with containerization.
    pbsDocker {

        process.executor = 'pbs'
        docker.remove      = true
        docker.runOptions  = '--rm'
        docker.registry    = 'quay.io'
        docker.enabled     = true
        docker.temp        = 'auto'
        docker.fixOwnership= true
  }

  // For execution on Azure Batch
  azurebatch {
        process.executor = 'azurebatch'
  }

}
```

```
process {
    container = py3Image

    withLabel:bigMem {
      memory = '8GB'
    }

    withLabel: gemma {
          container = gemmaImage
    }

    withLabel: latex {
          container = latexImage
    }
    withLabel: gcta{
          container = gctaImage
    }
    withLabel: metaanalyse {
          container = MetaAnImage
    }
//    withLabel: py2fast {
//          container = py2fastImage
//    }
    withLabel: bolt {
          container = boltImage
    }
    withLabel: R{
          container = rImage
    }

    withLabel : fastlmm{
          container = fastlmmImage
    }
    withLabel : saige{
          container = py3saige
    }
    withLabel : utils{
          container = py3utils
    }

}

//timeline {
//    enabled=true
//    file = "nextflow_reports/timeline.html"
//}

//report {
//    enabled = true
//    file = "nextflow_reports/report.html"
//}
```