01-Hail-common-variant-analysis

June 16, 2021

1 Institute for Behavioral Genetics International Statistical Genetics 2021 Workshop

1.1 Common Variant Analysis of Sequencing Data with Hail

You have reviewed the lecture videos, and you are ready to get hands-on with Hail to analyze real sequencing data.

In this practical, we will learn how to:

1) Use simple python code and Jupyter notebooks.
2) Use Hail to import a VCF and run basic queries over sequencing data.
3) Use Hail to perform basic quality control on sequencing data.
4) Use Hail to run a basic genome-wide association study on common variants in sequencing data.

2 Introduction

2.0.1 It doesn't all need to "stick" today

This practical contains a lot of new material, and the goal of this workbook is not for you to be able to reproduce from memory all the various capabilities demonstrated here. Instead, the goal is for you to get a sense for the kind of analysis tasks that sequencing data requires, and to gain some exposure to what these analyses look like in Hail.

There is no one-size-fits-all sequencing analysis pipeline, because each sequencing dataset will have unique properties that need to be understood and accounted for in QC and analysis. Hail can empower you to interrogate sequencing data, but it cannot give you all the questions to ask!

Some of the questions and exercises in this notebook might seem unrelated to the specific task of analyzing sequencing data, but that is intentional – Hail is a computational tool that hopes to help you indulge your scientific curiosity, and asking and answering a variety of questions about many aspects of your data is the best way to learn *how to Hail*.

We don't expect you to be able to run a full GWAS on your own data in Hail tomorrow. If this is something you want to do, there are **lots more** resources available – documentation, cookbooks, tutorials, and most importantly, the Hail community on the forum and zulip chatroom.

2.0.2 We encourage you to play

Hail is a highly expressive library with lots of functionality – you'll see just a small fraction of it today. Throughout this notebook and especially in the denoted **exercises**, we encourage you to experiment with the code being run to see what happens! Sometimes it will be an error, but sometimes you will encounter new pieces of functionality. If you're curious about how to use Hail to ask slightly different questions than the code or exercises here, please ask the faculty! We are eager to help.

2.0.3 Interactive analysis on the cloud

Part of what we think is so exciting about Hail is that Hail development has coincided with other technological shifts in the data science community.

Five years ago, most computational biologists analyzed sequencing data using command-line tools, and took advantage of research compute clusters by explicitly using scheduling frameworks like LSF or Sun Grid Engine. These clusters are powerful resources, but it requires a great deal of extra thought and effort to manage pipelines running on them.

Today, most Hail users run Hail from within interactive Python notebooks (like this one!) backed by thousands of cores on public compute clouds like Google Cloud, Amazon Web Services, or Microsoft Azure. You don't need to share a cluster with hundreds of other scientists, and you only need to pay for the resources that you use.

You won't get hands-on experience with this kind of usage today, but there are lots of resources to help you get started if you're interested in that. Please stay in touch with us after the workshop ends!

3 1. Using Jupyter

The notebook software that you are using right now is called Jupyter, which came from a combination of the languages **Ju**lia, **Pyt**hon, and **R**.

Learning objectives

- be comfortable running, editing, adding, and deleting code cells.
- learn techniques for unblocking yourself if Jupyter acts up.

3.0.1 Running cells

Evaluate cells using SHIFT + ENTER. Select the next cell and run it. If you prefer clicking, you can select the cell and click the "Run" button in the toolbar above.

```
[1]: print('Hello, world')
```

```
Hello, world
```

3.0.2 Modes

Jupyter has two modes, a **navigation mode** and an **editor mode**.

Navigation mode:

- BLUE cell borders
- `UP` / `DOWN` move between cells
- `ENTER` while a cell is selected will move to **editing mode**.
- Many letters are keyboard shortcuts! This is a common trap.

Editor mode:

- GREEN cell borders
- `UP` / `DOWN`/ move within cells before moving between cells.
- `ESC` will return to **navigation mode**.
- `SHIFT + ENTER` will evaluate a cell and return to **navigation mode**.

Try editing this markdown cell by double clicking, then re-rendering it by "running" the cell.

3.0.3 Cell types

There are several types of cells in Jupyter notebooks. The two you will see in this notebook are **Markdown** (text) and **Code**.

[2]:
```
# This is a code cell
my_variable = 5
```

This is a markdown cell, so even if something looks like code (as below), it won't get executed!

my_variable += 1

3.0.4 Shell commands

It is possible to call command-line utilities from Jupyter by prefixing a line with a `!`. For instance, we can print the current directory:

[3]:
```
! pwd
```

```
/Users/kumar/Dropbox (Partners
HealthCare)/HailTeam/Workshops/2021_Boulder/2021_IBG_Hail/resources
```

3.0.5 Tips and tricks

Keyboard shortcuts:

- `SHIFT + ENTER` to evaluate a cell
- `ESC` to return to navigation mode
- `y` to turn a markdown cell into code
- `m` to turn a code cell into markdown
- `a` to add a new cell **above** the currently selected cell
- `b` to add a new cell **below** the currently selected cell
- `d, d` (repeated) to delete the currently selected cell
- `TAB` to activate code completion

To try this out, create a new cell below this one using `b`, and print `my_variable` by starting with `print(my` and pressing `TAB`!

3.1 Resetting Jupyter if you get stuck

If at any point during this practical, you are unable to successfully run cells, it is possible that your Python interpreter is in a bad state due to cells being run in an incorrect order. If this happens, you can recover a working session by doing the following:

1. Navigate to the "Kernel" menu at the top, and select "Restart and clear output".
2. Select the cell you were working on, then select "Run all above" from the "Cell" menu at the top.
3. If the problem persists, reach out to the faculty for help!

4 2. Import and initialize Hail

In addition to Hail, we import a few methods from the Hail plotting library. We'll see examples soon!

[4]:
```
import hail as hl
from hail.plot import output_notebook, show
```

Now we initialize Hail and set up plotting to display inline in the notebook.

[5]:
```
hl.init()
output_notebook()
```

```
Running on Apache Spark version 3.1.1
SparkUI available at http://10.0.0.72:4040
Welcome to
     __  __     <>__
    / /_/ /__  __/ /
   / __  / _ `/ / /
  /_/ /_/\_,_/_/_/   version 0.2.68-13190f0b6103
LOGGING: writing to /Users/kumar/Dropbox (Partners HealthCare)/HailTeam/Workshop
s/2021_Boulder/2021_IBG_Hail/resources/hail-20210616-1647-0.2.68-13190f0b6103.lo
g
```

This notebook works on a small (~16MB) downsampled chunk of the publically available Human Genome Diversity Project (HGDP) dataset. HGDP is a super-set of the well-known 1000 genomes dataset, with a broader group of represented populations.

We can see the files used using `ls` below:

[6]:
```
! ls -lh resources/
```

```
total 38656
-rwxr-xr-x@ 1 kumar  staff    15K Jun 16 16:31 HGDP_sample_data.tsv
-rwxr-xr-x@ 1 kumar  staff   2.5M Jun 16 16:31
```

```
ensembl_gene_annotations.txt
-rwxr-xr-x@ 1 kumar  staff   382K Jun 16 16:31
hgdp_gene_annotations.tsv
-rwxr-xr-x@ 1 kumar  staff    16M Jun 16 16:31 hgdp_subset.vcf.bgz
```

5 3. Explore genetic data with Hail

Learning Objectives:

- To be comfortable exploring Hail data structures, especially the `MatrixTable`.
- To understand categories of functionality for performing QC.

5.0.1 Import data from VCF

The Variant Call Format (VCF) is a common file format for representing genetic data collected on multiple individuals (samples).

Hail has an import_vcf function that reads this file to a Hail `MatrixTable`, which is a general-purpose data structure that is often used to represent a matrix of genetic data.

Why not work directly on the VCF? While VCF is a text format that is easy for humans to read, it is inefficient to process on a computer.

The first thing we do is import (`import_vcf`) and convert the `VCF` file into a Hail native file format. This is done by using the `write` method below. Any queries that follow will now run much more quickly.

[7]:
```
hl.import_vcf('resources/hgdp_subset.vcf.bgz', min_partitions=4,␣
 ↪reference_genome='GRCh38')\
.write('resources/hgdp.mt', overwrite=True)
```

```
2021-06-16 16:48:04 Hail: INFO: Coerced sorted dataset
2021-06-16 16:48:33 Hail: INFO: wrote matrix table with 10441 rows and 392
columns in 4 partitions to resources/hgdp.mt
    Total size: 15.96 MiB
    * Rows/entries: 15.96 MiB
    * Columns: 1.85 KiB
    * Globals: 11.00 B
    * Smallest partition: 2628 rows (3.98 MiB)
    * Largest partition:  2570 rows (4.01 MiB)
```

5.0.2 HGDP as a Hail `MatrixTable`

We represent genetic data as a Hail `MatrixTable`, and name our variable `mt` to indicate this.

[8]:
```
mt = hl.read_matrix_table('resources/hgdp.mt')
```

5.0.3 What is a `MatrixTable`?

Let’s explore it!

You can see: - **numeric** types: - integers (`int32`, `int64`), e.g. `5` - floating point numbers (`float32`, `float64`), e.g. `5.5` or `3e-8` - **strings** (`str`), e.g. `"Foo"` - **boolean** values (`bool`) e.g. `True` - **collections**: - arrays (`array`), e.g. `[1,1,2,3]` - sets (`set`), e.g. `{1,3}` - dictionaries (`dict`), e.g. `{'Foo': 5, 'Bar': 10}` - **genetic data types**: - loci (`locus`), e.g. `[GRCh37] 1:10000` or `[GRCh38] chr1:10024` - genotype calls (`call`), e.g. `0/2` or `1|0`

[9]:
```
mt.describe(widget=True)
```

```
VBox(children=(HBox(children=(Button(description='globals', layout=Layout(height='30px', width=
```

```
Tab(children=(VBox(children=(HTML(value='<p><big>Global fields, with one value in the dataset.
```

5.0.4 Exercise

Take a few moments to explore the interactive representation of the matrix table above.

- Where is the variant information (`locus` and `alleles`)?
- Where is the sample identifier (`s`)?
- Where is the genotype quality `GQ`?

5.0.5 `show`

Hail has a variety of functionality to help you quickly interrogate a dataset. The `show()` method prints the first few values of any field, and even prints in pretty HTML output in a Jupyter notebook!

[10]:
```
mt.s.show()
```

```
+----------------------+
| s                    |
+----------------------+
| str                  |
+----------------------+
| "LP6005441-DNA_F08" |
| "LP6005441-DNA_C05" |
| "HGDP00961"          |
| "HGDP00804"          |
| "HGDP00926"          |
| "HGDP00716"          |
| "HGDP01269"          |
| "HGDP00241"          |
| "HGDP00110"          |
| "LP6005441-DNA_F02" |
| "HGDP01299"          |
| "HGDP00098"          |
| "LP6005441-DNA_D09" |
| "HGDP00817"          |
| "HGDP00611"          |
| "LP6005441-DNA_E09" |
| "HGDP01231"          |
| "HGDP01162"          |
| "HGDP00762"          |
| "LP6005442-DNA_H01" |
| "LP6005441-DNA_A05" |
| "LP6005441-DNA_A09" |
| "HGDP01384"          |
| "HGDP00893"          |
+----------------------+
showing top 24 rows
```

It is also possible to show() the matrix table itself, which prints a portion of the top-left corner of the variant-by-sample matrix:

[11]:
```
# show() works fine with no arguments, but can print too little data by default␣
↪on small screens!
mt.show(n_cols=3)
```

```
+---------------+------------+------------------------+------------------------+--------------
| locus         | alleles    | 'LP6005441-DNA_F08'.GT | 'LP6005441-DNA_F08'.DP | 'LP6005441-DN
+---------------+------------+------------------------+------------------------+--------------
| locus<GRCh38> | array<str> | call                   |                  int32 |
+---------------+------------+------------------------+------------------------+--------------
| chr1:17379    | ["G","A"]  | 0/0                    |                     11 |
| chr1:95068    | ["G","A"]  | 0/0                    |                     15 |
| chr1:111735   | ["C","A"]  | 0/0                    |                     14 |
| chr1:134610   | ["G","A"]  | 0/0                    |                      8 |
| chr1:414783   | ["T","C"]  | NA                     |                     NA |
| chr1:1130877  | ["C","G"]  | 0/0                    |                     24 |
| chr1:1226707  | ["C","G"]  | 0/0                    |                     26 |
| chr1:1491494  | ["G","A"]  | 0/0                    |                     27 |
| chr1:1618118  | ["G","A"]  | 0/0                    |                     27 |
| chr1:2078529  | ["G","A"]  | 0/0                    |                     30 |
| chr1:2512104  | ["G","A"]  | 0/0                    |                     25 |
| chr1:2683695  | ["C","G"]  | 0/1                    |                     55 |
| chr1:2689763  | ["A","C"]  | NA                     |                     NA |
| chr1:2758837  | ["C","A"]  | NA                     |                     NA |
| chr1:3008858  | ["A","G"]  | 0/0                    |                     31 |
| chr1:3254545  | ["T","C"]  | 0/0                    |                     23 |
| chr1:3299310  | ["A","C"]  | 0/1                    |                     27 |
| chr1:3779436  | ["T","C"]  | 1/1                    |                     27 |
| chr1:3848254  | ["C","T"]  | 0/0                    |                     34 |
| chr1:4156721  | ["C","T"]  | 0/0                    |                     27 |
| chr1:4510922  | ["C","T"]  | 0/0                    |                     32 |
| chr1:4748394  | ["G","A"]  | 0/0                    |                     27 |
| chr1:5008942  | ["A","G"]  | 0/0                    |                     36 |
| chr1:5331415  | ["G","T"]  | 0/0                    |                     33 |
+---------------+------------+------------------------+------------------------+----------------

+------------------------+------------------------+------------------------+--------------------
| 'LP6005441-DNA_C05'.DP | 'LP6005441-DNA_C05'.GQ | 'LP6005441-DNA_C05'.AD | 'LP6005441-DNA_C0
+------------------------+------------------------+------------------------+--------------------
|                  int32 |                  int32 | array<int32>           | array<int32>
+------------------------+------------------------+------------------------+--------------------
|                     15 |                     30 | NA                     | NA
|                     27 |                     30 | NA                     | NA
|                     15 |                     20 | NA                     | NA
|                     NA |                     NA | NA                     | NA
|                      9 |                     20 | NA                     | NA
|                     27 |                     30 | NA                     | NA
|                     25 |                     20 | NA                     | NA
|                     30 |                     20 | NA                     | NA
|                     30 |                     40 | NA                     | NA
|                     37 |                     60 | NA                     | NA
|                     27 |                     40 | NA                     | NA
|                     76 |                     99 | [45,31]                | [782,0,1258]
|                     NA |                     NA | NA                     | NA
|                     NA |                     NA | NA                     | NA
|                     26 |                     30 | NA                     | NA
|                     37 |                     99 | [14,23]                | [672,0,372]
|                     26 |                     20 | NA                     | NA
|                     27 |                     20 | NA                     | NA
|                     25 |                     20 | NA                     | NA
|                     27 |                     30 | NA                     | NA
|                     28 |                     40 | NA                     | NA
|                     28 |                     30 | NA                     | NA
|                     34 |                     50 | NA                     | NA
|                     36 |                     40 | NA                     | NA
+------------------------+------------------------+------------------------+--------------------
showing top 24 rows
showing the first 3 of 392 columns
```

The above output is visually noisy because the matrix table has as lot of information in it. `show`ing just the called genotype (`GT`) is a bit more friendly.

The printed representation of GT is explained below, where `a` is the reference allele and `A` is the alternate allele:

`0/0` : homozygous reference or `aa`

`0/1` : heterozygous or `Aa`

`1/1` : homozygous alternate or `AA`

[12]:
```
mt.GT.show()
```

```
+---------------+------------+------------------------+------------------------+-------------
| locus         | alleles    | 'LP6005441-DNA_F08'.GT | 'LP6005441-DNA_C05'.GT | 'HGDP00961'.G
+---------------+------------+------------------------+------------------------+-------------
| locus<GRCh38> | array<str> | call                   | call                   | call
+---------------+------------+------------------------+------------------------+-------------
| chr1:17379    | ["G","A"]  | 0/0                    | 0/0                    | 0/0
| chr1:95068    | ["G","A"]  | 0/0                    | 0/0                    | NA
| chr1:111735   | ["C","A"]  | 0/0                    | 0/0                    | 0/1
| chr1:134610   | ["G","A"]  | 0/0                    | NA                     | NA
| chr1:414783   | ["T","C"]  | NA                     | 0/0                    | NA
| chr1:1130877  | ["C","G"]  | 0/0                    | 0/0                    | 0/0
| chr1:1226707  | ["C","G"]  | 0/0                    | 0/0                    | 0/0
| chr1:1491494  | ["G","A"]  | 0/0                    | 0/0                    | 0/0
| chr1:1618118  | ["G","A"]  | 0/0                    | 0/0                    | 0/1
| chr1:2078529  | ["G","A"]  | 0/0                    | 0/0                    | 0/0
| chr1:2512104  | ["G","A"]  | 0/0                    | 0/0                    | 0/0
| chr1:2683695  | ["C","G"]  | 0/1                    | 0/1                    | 0/0
| chr1:2689763  | ["A","C"]  | NA                     | NA                     | 0/0
| chr1:2758837  | ["C","A"]  | NA                     | NA                     | NA
| chr1:3008858  | ["A","G"]  | 0/0                    | 0/0                    | 0/0
| chr1:3254545  | ["T","C"]  | 0/0                    | 0/1                    | 0/0
| chr1:3299310  | ["A","C"]  | 0/1                    | 0/0                    | 1/1
| chr1:3779436  | ["T","C"]  | 1/1                    | 0/0                    | 0/0
| chr1:3848254  | ["C","T"]  | 0/0                    | 0/0                    | 0/0
| chr1:4156721  | ["C","T"]  | 0/0                    | 0/0                    | 0/1
| chr1:4510922  | ["C","T"]  | 0/0                    | 0/0                    | 0/1
| chr1:4748394  | ["G","A"]  | 0/0                    | 0/0                    | 0/1
| chr1:5008942  | ["A","G"]  | 0/0                    | 0/0                    | NA
| chr1:5331415  | ["G","T"]  | 0/0                    | 0/0                    | 0/0
+---------------+------------+------------------------+------------------------+-------------

+-----------------+------------------------+-----------------+-----------------+-------------
| 'HGDP00110'.GT | 'LP6005441-DNA_F02'.GT | 'HGDP01299'.GT | 'HGDP00098'.GT | 'LP6005441-DNA_D0
+-----------------+------------------------+-----------------+-----------------+-------------
| call            | call                   | call            | call            | call
+-----------------+------------------------+-----------------+-----------------+-------------
| 0/0             | 0/0                    | 0/0             | 0/0             | 0/0
| 0/1             | 0/0                    | 0/1             | 0/0             | 0/0
| 0/0             | NA                     | NA              | NA              | NA
| 0/0             | NA                     | NA              | 0/0             | NA
| NA              | NA                     | NA              | NA              | NA
| 0/0             | 0/0                    | 0/0             | 0/0             | 0/0
| 0/0             | 0/0                    | 0/0             | 0/0             | 0/0
| 0/1             | 0/0                    | 0/0             | 0/0             | 0/0
| 0/0             | 0/0                    | 0/0             | 0/0             | 0/1
| 0/0             | 0/0                    | 0/0             | 0/0             | 0/0
| 0/0             | 0/0                    | 0/0             | 0/0             | 0/0
| 0/1             | 0/0                    | 0/1             | 0/0             | 0/1
| NA              | NA                       | NA                | 0/0               | NA
| 0/0             | NA                       | NA                | NA                | NA
| 0/0             | 0/1                      | 0/1               | 0/1               | 0/0
| 0/1             | 0/1                      | 0/1               | 0/1               | 1/1
| NA              | NA                       | NA                | NA                | 0/1
| 0/1             | 0/0                      | 1/1               | 0/1               | 0/1
| 0/0             | 0/0                      | 0/0               | 0/0               | 0/0
| 0/0             | 0/0                      | 0/0               | 0/0               | 0/0
| 0/1             | 0/0                      | 1/1               | 0/0               | 0/0
| 0/1             | 0/1                      | 0/1               | 1/1               | 0/0
| 0/0             | 0/0                      | 0/0               | NA                | 0/0
| 0/0             | 0/0                      | 0/0               | 0/0               | 0/0
+-----------------+--------------------------+-----------------+-----------------+-----------------
showing top 24 rows
showing the first 14 of 392 columns
```

5.0.6 Exercise

There is a fourth value seen above, other than `0/0`, `0/1`, `1/1`. What is it?

5.0.7 `summarize`

`summarize` Prints (potentially) useful information about any field or object:

`DP` is the read depth (number of short reads spanning a position for a given sample). Let's summarize all values of DP:

[13]:
```python
mt.DP.summarize()
```

```
4092872 records.

- DP (int32):
  Non-missing: 3851673 (94.11%)
      Missing: 241199 (5.89%)
      Minimum: 0
      Maximum: 5057
         Mean: 33.02
      Std Dev: 30.20
```

`AD` is the array of allelic depth per allele at a called genotype. Note especially the missingness properties:

[14]:
```python
mt.AD.summarize()
```

```
4092872 records.

- AD (array<int32>):
  Non-missing: 1164892 (28.46%)
```

```
      Missing: 2927980 (71.54%)
     Min Size: 2
     Max Size: 2
    Mean Size: 2.00

  - AD[<elements>] (int32):
    Non-missing: 2329784 (100.00%)
        Missing: 0
        Minimum: 0
        Maximum: 1299
           Mean: 17.09
        Std Dev: 15.13
```

5.0.8 Exercise

In the empty cell below, summarize some of the other fields on the matrix table. You can use the interactive widget above to find the names of some of the other fields.

Share any interesting findings with your colleagues!

[ ]:

```

```

5.0.9 `count`

`MatrixTable.count` returns a tuple with the number of rows (variants) and number of columns (samples).

[15]:

```
mt.count()
```

[15]:

```
(10441, 392)
```

The count above tells us that we have 10,441 variants and 392 samples. This is just a tiny slice of a real sequencing dataset. The largest sequencing datasets today comprise hundreds of thousands of samples and more than a billion variants.

5.1 Hail has a large library of genetics functionality

Hail can be used to analyze any kind of data (Hail team members have used Hail to analyze household financial data, USA election polling data, and even to build a bot that posts real-time updates about the Euro 2020 tournament to Slack). However, Hail does not have *only* general-purpose analysis functionality. Hail has a large set of functionality built for genetics and genomics.

For example, `hl.summarize_variants` prints useful statistics about the variants in the dataset. These are not part of the generic `summarize()` function, which must support all kinds of data, not just variant data!

[16]:

```
hl.summarize_variants(mt)
```

```
==============================
Number of variants: 10441
==============================
Alleles per variant
-------------------
  2 alleles: 10441 variants
==============================
Variants per contig
-------------------
   chr1: 881 variants
   chr2: 799 variants
   chr3: 728 variants
   chr4: 659 variants
   chr5: 618 variants
   chr6: 572 variants
   chr7: 576 variants
   chr8: 525 variants
   chr9: 476 variants
  chr10: 516 variants
  chr11: 483 variants
  chr12: 411 variants
  chr13: 397 variants
  chr14: 332 variants
  chr15: 316 variants
  chr16: 319 variants
  chr17: 312 variants
  chr18: 270 variants
  chr19: 252 variants
  chr20: 263 variants
  chr21: 194 variants
  chr22: 170 variants
   chrX: 361 variants
   chrY: 11 variants
==============================
Allele type distribution
------------------------
  SNP: 10441 alternate alleles (Ti: 6602, Tv: 3839, ratio: 1.72)
==============================
```

6 4. Annotation and quality control

6.1 Integrate sample information

Our dataset currently only has sample IDs and genetic data. In order to run a toy GWAS, we need phenotype information.

We can find it in the following file:

[17]:
```
! head resources/HGDP_sample_data.tsv
```

```
sample_id       pop     continental_pop sex_karyotype   sleep_duration
tea_intake_daily        general_happiness       screen_time_per_day
HG00107 gbr     nfe     XY      6       3       3.2895e+00      11
HG00114 gbr     nfe     XY      5       3       3.5099e+00      10
HG00121 gbr     nfe     XX      6       6       2.0851e+00      6
HG00127 gbr     nfe     XX      6       3       2.7580e+00      6
HG00132 gbr     nfe     XX      5       6       2.2454e+00      5
HG00149 gbr     nfe     XY      5       6       2.8159e+00      9
HG00177 fin     fin     XX      7       9       3.3661e+00      8
HG00190 fin     fin     XY      5       6       2.9159e+00      6
HG00233 gbr     nfe     XX      8       3       3.9002e+00      10
```

We can import it as a Hail Table with hl.import_table.

We call it `sd` for “sample data”.

[18]:
```
sd = hl.import_table('resources/HGDP_sample_data.tsv',
                     key='sample_id',
                     impute=True)
```

```
2021-06-16 16:49:57 Hail: INFO: Reading table to impute column types
2021-06-16 16:49:59 Hail: INFO: Finished type imputation
  Loading field 'sample_id' as type str (imputed)
  Loading field 'pop' as type str (imputed)
  Loading field 'continental_pop' as type str (imputed)
  Loading field 'sex_karyotype' as type str (imputed)
  Loading field 'sleep_duration' as type int32 (imputed)
  Loading field 'tea_intake_daily' as type int32 (imputed)
  Loading field 'general_happiness' as type float64 (imputed)
  Loading field 'screen_time_per_day' as type int32 (imputed)
```

The “key” argument tells Hail to use the `sample_id` field as the table key, which is used to find matching values in joins. In a moment, we will be joining the `sd` table onto our matrix table so that we can use the sample data fields in our QC and analysis. It is also possible to specify a new key for an existing table using the `.key_by(...)` method.

The “impute” argument tells Hail to impute the data types of the fields on the table. What does this mean? It means that you can ask Hail to figure out what is the data type in each column field such as `str` (string or just characters), `bool` (boolean or just true and false), `float64` (float or numbers with decimals), or `int32` (integer or numbers without decimals/whole numbers). If you don’t use the `impute` flag or specify types manually with the `types` argument, each field will be imported as a string.

While we can see the names and types of fields in the logging messages and in the `head` output above, we can also `show` this table:

[19]:
```
sd.show()
```

```
+-----------+-------+----------------+--------------+----------------+-----------------+---
```

```
| sample_id | pop   | continental_pop | sex_karyotype | sleep_duration | tea_intake_daily | ger
+-----------+-------+-----------------+---------------+----------------+------------------+----
| str       | str   | str             | str           |          int32 |            int32 |
+-----------+-------+-----------------+---------------+----------------+------------------+----
| "HG00107" | "gbr" | "nfe"           | "XY"          |              6 |                3 |
| "HG00114" | "gbr" | "nfe"           | "XY"          |              5 |                3 |
| "HG00121" | "gbr" | "nfe"           | "XX"          |              6 |                6 |
| "HG00127" | "gbr" | "nfe"           | "XX"          |              6 |                3 |
| "HG00132" | "gbr" | "nfe"           | "XX"          |              5 |                6 |
| "HG00149" | "gbr" | "nfe"           | "XY"          |              5 |                6 |
| "HG00177" | "fin" | "fin"           | "XX"          |              7 |                9 |
| "HG00190" | "fin" | "fin"           | "XY"          |              5 |                6 |
| "HG00233" | "gbr" | "nfe"           | "XX"          |              8 |                3 |
| "HG00252" | "gbr" | "nfe"           | "XY"          |              5 |                0 |
| "HG00260" | "gbr" | "nfe"           | "XY"          |              5 |                6 |
| "HG00280" | "fin" | "fin"           | "XY"          |              4 |                6 |
| "HG00332" | "fin" | "fin"           | "XX"          |              6 |                3 |
| "HG00342" | "fin" | "fin"           | "XY"          |              8 |                3 |
| "HG00344" | "fin" | "nfe"           | "XX"          |              6 |                6 |
| "HG00353" | "fin" | "fin"           | "XX"          |              6 |                6 |
| "HG00366" | "fin" | "fin"           | "XY"          |              6 |                6 |
| "HG00371" | "fin" | "fin"           | "XY"          |              5 |                6 |
| "HG00437" | "chs" | "eas"           | "XX"          |              6 |                9 |
| "HG00438" | "chs" | "eas"           | "XX"          |              6 |                9 |
| "HG00473" | "chs" | "eas"           | "XX"          |              6 |                6 |
| "HG00478" | "chs" | "eas"           | "XY"          |              8 |                6 |
| "HG00582" | "chs" | "eas"           | "XX"          |              6 |                9 |
| "HG00595" | "chs" | "eas"           | "XY"          |              8 |                6 |
+-----------+-------+-----------------+---------------+----------------+------------------+----
showing top 24 rows
```

And we can `summarize` each field in `sd`:

```
[20]: sd.summarize()
```

```
2021-06-16 16:50:05 Hail: INFO: Coerced sorted dataset

392 records.

  - sample_id (str):
      Non-missing: 392 (100.00%)
          Missing: 0
         Min Size: 7
         Max Size: 17
        Mean Size: 7.92
    Sample Values: ['HG00107', 'HG00114', 'HG00121', 'HG00127', 'HG00132']

  - pop (str):
      Non-missing: 392 (100.00%)
          Missing: 0
         Min Size: 2
         Max Size: 11
        Mean Size: 3.84
    Sample Values: ['gbr', 'gbr', 'gbr', 'gbr', 'gbr']

  - continental_pop (str):
      Non-missing: 392 (100.00%)
          Missing: 0
         Min Size: 3
         Max Size: 3
        Mean Size: 3.00
    Sample Values: ['nfe', 'nfe', 'nfe', 'nfe', 'nfe']

  - sex_karyotype (str):
      Non-missing: 392 (100.00%)
          Missing: 0
         Min Size: 2
         Max Size: 2
        Mean Size: 2.00
    Sample Values: ['XY', 'XY', 'XX', 'XX', 'XX']

  - sleep_duration (int32):
    Non-missing: 392 (100.00%)
        Missing: 0
        Minimum: 2
        Maximum: 10
           Mean: 6.03
        Std Dev: 1.44

  - tea_intake_daily (int32):
    Non-missing: 392 (100.00%)
        Missing: 0
        Minimum: 0
        Maximum: 12
           Mean: 5.37
        Std Dev: 1.97

  - general_happiness (float64):
    Non-missing: 392 (100.00%)
        Missing: 0
        Minimum: 1.81
        Maximum: 4.50
           Mean: 2.94
        Std Dev: 0.51

  - screen_time_per_day (int32):
```

```
Non-missing: 392 (100.00%)
    Missing: 0
    Minimum: 0
    Maximum: 13
       Mean: 6.57
    Std Dev: 2.50
```

6.2 Add sample data to our HGDP `MatrixTable`

Let's now merge our genetic data (`mt`) with our sample data (`sd`).

This is a join between the `sd` table and the columns of our matrix table. It just takes one line:

[21]:
```python
mt = mt.annotate_cols(sample_data = sd[mt.s])
```

6.2.1 What's going on here?

Understanding what's going on here is a bit more difficult. To understand, we need to understand a few pieces:

1. `annotate` methods In Hail, `annotate` methods refer to **adding new fields**.

- `MatrixTable`'s `annotate_cols` adds new column (**sample**) fields.
- `MatrixTable`'s `annotate_rows` adds new row (**variant**) fields.
- `MatrixTable`'s `annotate_entries` adds new entry (**genotype**) fields.
- `Table`'s `annotate` adds new row fields.

In the above cell, we are adding a new column (**sample**) field called "sample_data". This field should be the values in our table `sd` associated with the sample ID `s` in our `MatrixTable` - that is, this is performing a **join**.

Python uses square brackets to look up values in dictionaries:

```
>>> d = {'foo': 5, 'bar': 10}

>>> d['foo']
'bar'
```

You should think of this in much the same way - for each column of `mt`, we are looking up the fields in `sd` using the sample ID `s`.

Let's see how the matrix table has changed:

[22]:
```python
mt.describe(widget=True)
```

```
VBox(children=(HBox(children=(Button(description='globals', layout=Layout(height='30px', width=


Tab(children=(VBox(children=(HTML(value='<p><big>Global fields, with one value in the dataset.
```

6.2.2 Cheat sheets

More information about matrix tables and tables can be found in a graphical representation as Hail cheat sheets:

- MatrixTable
- Table

6.3 Query the sample data

We will use some of the general-purpose query functionality to interrogate the sample data we have imported.

The code below uses the `aggregate_cols` method on our matrix table, which computes aggregate statistics about column (sample) data. There are also methods for `aggregate_rows` (aggregate over row data) and `aggregate_entries` aggregate over all of the entries in our matrix, one per variant per sample).

Hail **aggregators** can be recognized by the `hl.agg` prefix. Some examples:

- `hl.agg.fraction(CONDITION)` - compute the fraction of values at which `CONDITION` is true.
- `hl.agg.count_where(CONDITION)` - compute the number of values at which `CONDITION` is true.
- `hl.agg.stats(X)` - compute a few useful statistics about `X`.
- `hl.agg.counter(X)` - compute the number of occurrences of each unique value of `X`. Useful for categorical fields like strings, not as useful for numbers!
- `hl.agg.corr(X, Y)` - compute the Pearson correlation coefficient between X and Y values.
- For more adventurous students, see the full list of aggregators.

6.3.1 Sex karyotype

To start, we will compute the occurrences of each value of `sex_karyotype`:

```
[23]: mt.aggregate_cols(hl.agg.counter(mt.sample_data.sex_karyotype))
```

```
[23]: frozendict({'XX': 188, 'XY': 204})
```

The above result tells us that slightly more than half of our samples are XY, and the rest are XX. What should you do if some of your samples are neither XX or XY? That depends on the analysis you are trying to do, but you should be ready to think about this case!

6.3.2 Ancestry

How many people are in each self-reported major continental ancestry group?

```
[24]: mt.aggregate_cols(hl.agg.counter(mt.sample_data.continental_pop))
```

```
[24]: frozendict({'nfe': 60, 'fin': 8, 'mid': 16, 'oth': 8, 'afr': 93, 'eas': 72,
      'sas': 87, 'amr': 48})
```

6.3.3 Exercise

Try changing `continental_pop` to `pop` and rerunning the cell above. Most of the populations are abbreviated, but see if you can find an ancestral population from each continent among the non-abbreviated ones!

6.3.4 Numeric aggregations

The numeric aggregators are used the same way:

[25]:
```
mt.aggregate_cols(hl.agg.stats(mt.sample_data.sleep_duration))
```

[25]:
```
Struct(mean=6.030612244897959, stdev=1.44246162636083, min=2.0, max=10.0, n=392,
sum=2364.0)
```

6.3.5 Exercise

Use the `fraction` and `count_where` aggregators to answer the following questions in the cells below:

1. How many samples drink more than 8 cups of tea per day? *Hint: the CONDITION will take the form `SOMETHING > 8`.*
2. What fraction of samples sleep less than 4 hours per day?

[ ]:
```

```

[ ]:
```

```

6.4 Checkpoint #1.

You've finished the annotation section. Now is a good time to take a bio break or ask any pertinent questions to the faculty!

7 5. Interrogating SNP distributions

You might have noticed in the above `hl.summarize_variants()` printout that the only "allele type" present in our data is SNP. In a real sequencing dataset, you'll see insertions, deletions, "star" alleles (upstream spanning deletions), and complex variation (none of the above).

We selected only SNPs from the HGDP dataset for simplicity. However, since we've got a few thousand SNPs, we might as well look at the frequency of the twelve SNP polymorphisms.

We can do this with an aggregator!

The `collections.Counter` class we use below is a builtin Python data structure that helps print out our query result sorted by number of occurrences.

[26]:
```
from collections import Counter
Counter(mt.aggregate_rows(hl.agg.counter((mt.alleles[0], mt.alleles[1])))).
↪most_common()
```

```
[26]: [(('C', 'T'), 1877),
       (('G', 'A'), 1848),
       (('T', 'C'), 1460),
       (('A', 'G'), 1417),
       (('G', 'T'), 560),
       (('C', 'A'), 516),
       (('C', 'G'), 497),
       (('A', 'C'), 482),
       (('T', 'G'), 476),
       (('A', 'T'), 461),
       (('G', 'C'), 436),
       (('T', 'A'), 411)]
```

7.0.1 Exercise

Discuss these two questions as a group:

Question 1 - Why do the C/T and G/A SNPs have roughly the same frequency? Why do the T/C and A/G SNPs have roughly the same frequency?

Question 2 - Why does the C/T SNP occur so much more frequently than a T/A SNP?

Hint: The answer to these questions doesn't involve statistics, but basic biology!

7.0.2 More complicated aggregators

You've seen and tried a taste of aggregator functionality, but aggregators in Hail can be very expressive. As an example, we're going to compute the sample IDs of the five happiest XX individuals. Don't worry about being able to reproduce this on your own right now!

The answer involves using the filter aggregator in combination with the take aggregator, and using the `take` aggregator's optional `ordering` argument to indicate that we want 5 sample IDs sorted by happiness.

```
[27]: mt.aggregate_cols(
          hl.agg.filter(
              mt.sample_data.sex_karyotype == 'XX',
              hl.agg.take(hl.struct(sample_id=mt.s, happiness=mt.sample_data.
       ↪general_happiness),
                          5,
                          ordering=-mt.sample_data.general_happiness)
          )
      )
```

```
2021-06-16 16:50:30 Hail: INFO: Coerced sorted dataset
```

```
[27]: [Struct(sample_id='HG01323', happiness=4.2144),
       Struct(sample_id='HG04054', happiness=4.0774),
       Struct(sample_id='HG03951', happiness=4.072),
       Struct(sample_id='HG03049', happiness=4.0379),
Struct(sample_id='HG02126', happiness=4.0227)]
```

7.1 Checkpoint #2.

You've finished the basic aggregation materials!

8 6. Sample QC

In past workshop sessions you have learned the important adage: *garbage in, garbage out*. Good QC takes time and thoughtfulness but is necessary for robust results.

Here, we run through some simple sample qc steps, but **these steps are not a one-size-fits-all solution for QC on your own data!**

Hail has the function hl.sample_qc to compute a list of useful statistics about samples from sequencing data. This function adds a new column field, `sample_qc`, with the computed statistics. Note that this doesn't actually remove samples for you – those decisions are up to you. The `sample_qc` method gives you some data you can use as a starting point.

Click the sample_qc link above to see the documentation, which lists the computed fields and their descriptions.

[28]:
```
mt = hl.sample_qc(mt)
```

[29]:
```
mt.describe(widget=True)
```

```
VBox(children=(HBox(children=(Button(description='globals', layout=Layout(height='30px', width=
```

```
Tab(children=(VBox(children=(HTML(value='<p><big>Global fields, with one value in the dataset.
```

Hail includes a plotting library built on bokeh that makes it easy to visualize fields of Hail tables and matrix tables.

Let's visualize the pairwise distribution of `Mean DP` (Read Depth) and `Call Rate`.

Note that you can **hover over points with your mouse to see the sample IDs!**

[30]:
```
p = hl.plot.scatter(x=mt.sample_qc.dp_stats.mean,
                    y=mt.sample_qc.call_rate,
                    xlabel='Mean DP',
                    ylabel='Call Rate',
                    hover_fields={'ID': mt.s},
                    size=8)
show(p)
```

8.0.1 Exercise

Try adding the following argument into the plot function argument list above below the `hover_fields=...` line:

```
label=mt.sample_data.sex_karyotype,
```

8.1 Filter columns using generated QC statistics

Before filtering samples, we should compute a raw sample count:

```
[31]: mt.count_cols()
```

```
[31]: 392
```

`filter_cols` removes entire columns from the matrix table. Here, we keep columns (samples) where the `call_rate` is over 92%:

```
[32]: mt = mt.filter_cols(mt.sample_qc.call_rate >= 0.92)
```

We can compute a final sample count:

```
[33]: mt.count_cols()
```

```
[33]: 385
```

How many samples did not meet your QC criteria?

9 7. Variant QC

Now that we have successfully gone through basic sample QC using the function `sample_qc` and general-purpose filtering methods, let's do variant QC.

Hail has the function hl.variant_qc to compute a list of useful statistics about **variants** from sequencing data.

Once again, **Click the link** above to see the documentation!

```
[34]: mt = hl.variant_qc(mt)
```

```
[35]: mt.describe(widget=True)
```

```
VBox(children=(HBox(children=(Button(description='globals', layout=Layout(height='30px', width=

Tab(children=(VBox(children=(HTML(value='<p><big>Global fields, with one value in the dataset.
```

Find the `variant_qc` output!

We can `show()` the computed information:

```
[36]: mt.variant_qc.show()
```

```
+---------------+------------+-------------------------+---------------------------+---------
| locus         | alleles    | variant_qc.dp_stats.mean | variant_qc.dp_stats.stdev | variant_
+---------------+------------+-------------------------+---------------------------+---------
```

```
| locus<GRCh38> | array<str> |                  float64 |                  float64 |
+---------------+------------+--------------------------+--------------------------+---------
| chr1:17379    | ["G","A"]  |                 4.95e+01 |                 2.17e+01 |
| chr1:95068    | ["G","A"]  |                 2.28e+01 |                 1.19e+01 |
| chr1:111735   | ["C","A"]  |                 1.42e+01 |                 6.02e+00 |
| chr1:134610   | ["G","A"]  |                 1.36e+01 |                 5.46e+00 |
| chr1:414783   | ["T","C"]  |                 6.27e+00 |                 2.65e+00 |
| chr1:1130877  | ["C","G"]  |                 3.75e+01 |                 7.77e+00 |
| chr1:1226707  | ["C","G"]  |                 3.61e+01 |                 6.28e+00 |
| chr1:1491494  | ["G","A"]  |                 3.41e+01 |                 5.39e+00 |
| chr1:1618118  | ["G","A"]  |                 3.63e+01 |                 6.23e+00 |
| chr1:2078529  | ["G","A"]  |                 3.41e+01 |                 4.28e+00 |
| chr1:2512104  | ["G","A"]  |                 3.47e+01 |                 5.52e+00 |
| chr1:2683695  | ["C","G"]  |                 5.00e+01 |                 2.22e+01 |
| chr1:2689763  | ["A","C"]  |                 1.44e+01 |                 4.80e+00 |
| chr1:2758837  | ["C","A"]  |                 1.63e+01 |                 1.18e+01 |
| chr1:3008858  | ["A","G"]  |                 3.54e+01 |                 7.13e+00 |
| chr1:3254545  | ["T","C"]  |                 3.15e+01 |                 5.85e+00 |
| chr1:3299310  | ["A","C"]  |                 2.54e+01 |                 9.66e+00 |
| chr1:3779436  | ["T","C"]  |                 3.45e+01 |                 6.73e+00 |
| chr1:3848254  | ["C","T"]  |                 3.16e+01 |                 5.76e+00 |
| chr1:4156721  | ["C","T"]  |                 3.24e+01 |                 4.77e+00 |
| chr1:4510922  | ["C","T"]  |                 3.26e+01 |                 5.06e+00 |
| chr1:4748394  | ["G","A"]  |                 3.28e+01 |                 6.36e+00 |
| chr1:5008942  | ["A","G"]  |                 2.63e+01 |                 7.11e+00 |
| chr1:5331415  | ["G","T"]  |                 3.46e+01 |                 6.12e+00 |
+---------------+------------+--------------------------+--------------------------+---------

+------------------------+------------------------+---------------+----------------------+---
| variant_qc.gq_stats.min | variant_qc.gq_stats.max | variant_qc.AC | variant_qc.AF        | va
+------------------------+------------------------+---------------+----------------------+---
|                float64 |                float64 | array<int32>  | array<float64>       |
+------------------------+------------------------+---------------+----------------------+---
|               2.00e+01 |               9.90e+01 | [752,10]      | [9.87e-01,1.31e-02] |
|               1.20e+01 |               9.90e+01 | [556,122]     | [8.20e-01,1.80e-01] |
|               1.00e+00 |               9.90e+01 | [538,84]      | [8.65e-01,1.35e-01] |
|               6.00e+00 |               5.60e+01 | [439,15]      | [9.67e-01,3.30e-02] |
|               3.00e+00 |               2.60e+01 | [110,8]       | [9.32e-01,6.78e-02] |
|               2.00e+01 |               9.90e+01 | [758,12]      | [9.84e-01,1.56e-02] |
|               2.00e+01 |               9.90e+01 | [768,2]       | [9.97e-01,2.60e-03] |
|               2.00e+01 |               9.90e+01 | [761,9]       | [9.88e-01,1.17e-02] |
|               2.00e+01 |               9.90e+01 | [639,129]     | [8.32e-01,1.68e-01] |
|               2.00e+01 |               9.90e+01 | [770,0]       | [1.00e+00,0.00e+00] |
|               2.00e+01 |               9.90e+01 | [769,1]       | [9.99e-01,1.30e-03] |
|               1.10e+01 |               9.90e+01 | [578,174]     | [7.69e-01,2.31e-01] |
|               1.10e+01 |               6.00e+01 | [340,0]       | [1.00e+00,0.00e+00] |
|               3.00e+00 |               9.90e+01 | [304,50]      | [8.59e-01,1.41e-01] |
|               2.00e+01 |               9.90e+01 | [537,233]     | [6.97e-01,3.03e-01] |
|              2.00e+01 |              9.90e+01 | [487,283]     | [6.32e-01,3.68e-01] |
|              9.00e+00 |              9.90e+01 | [362,72]      | [8.34e-01,1.66e-01] |
|              2.00e+01 |              9.90e+01 | [398,372]     | [5.17e-01,4.83e-01] |
|              2.00e+01 |              9.90e+01 | [769,1]       | [9.99e-01,1.30e-03] |
|              2.00e+01 |              9.90e+01 | [743,23]      | [9.70e-01,3.00e-02] |
|              2.00e+01 |              9.90e+01 | [677,93]      | [8.79e-01,1.21e-01] |
|              2.00e+01 |              9.90e+01 | [451,317]     | [5.87e-01,4.13e-01] |
|              1.10e+01 |              9.90e+01 | [636,0]       | [1.00e+00,0.00e+00] |
|              2.00e+01 |              9.90e+01 | [767,3]       | [9.96e-01,3.90e-03] |
+-----------------------+-----------------------+---------------+---------------------+---

+-----------------------+----------------------+-------------------+---------------------+--
| variant_qc.n_not_called | variant_qc.n_filtered | variant_qc.n_het | variant_qc.n_non_ref | v
+-----------------------+----------------------+-------------------+---------------------+--
|                  int64 |                 int64 |             int64 |                int64 |
+-----------------------+----------------------+-------------------+---------------------+--
|                      4 |                     0 |                10 |                   10 |
|                     46 |                     0 |               118 |                  120 |
|                     74 |                     0 |                68 |                   76 |
|                    158 |                     0 |                13 |                   14 |
|                    326 |                     0 |                 2 |                    5 |
|                      0 |                     0 |                12 |                   12 |
|                      0 |                     0 |                 2 |                    2 |
|                      0 |                     0 |                 9 |                    9 |
|                      1 |                     0 |                85 |                  107 |
|                      0 |                     0 |                 0 |                    0 |
|                      0 |                     0 |                 1 |                    1 |
|                      9 |                     0 |               174 |                  174 |
|                    215 |                     0 |                 0 |                    0 |
|                    208 |                     0 |                18 |                   34 |
|                      0 |                     0 |               117 |                  175 |
|                      0 |                     0 |               139 |                  211 |
|                    168 |                     0 |                44 |                   58 |
|                      0 |                     0 |               124 |                  248 |
|                      0 |                     0 |                 1 |                    1 |
|                      2 |                     0 |                21 |                   22 |
|                      0 |                     0 |                69 |                   81 |
|                      1 |                     0 |               175 |                  246 |
|                     67 |                     0 |                 0 |                    0 |
|                      0 |                     0 |                 3 |                    3 |
+-----------------------+----------------------+-------------------+---------------------+--
showing top 24 rows
```

Metrics like `call_rate` are important for QC. Let's plot the cumulative density function of call rate per variant:

[37]:
```
show(hl.plot.cdf(mt.variant_qc.call_rate))
```

```
BokehUserWarning: ColumnDataSource's columns must be of the same length. Current
lengths: ('left', 124), ('right', 124), ('top', 126)
BokehUserWarning: ColumnDataSource's columns must be of the same length. Current
lengths: ('bottom', 126), ('left', 124), ('right', 124), ('top', 126)
BokehUserWarning: ColumnDataSource's columns must be of the same length. Current
lengths: ('x', 126), ('y', 128)
```

Before filtering variants, we should compute a raw variant count:

[38]:
```python
# pre-qc variant count
mt.count_rows()
```

[38]:
```
10441
```

`filter_rows` removes entire rows of the matrix table. Here, we keep rows where the `call_rate` is over 95%:

[39]:
```python
mt = mt.filter_rows(mt.variant_qc.call_rate > 0.95)
```

9.0.1 Hardy-Weinberg filter

In past sessions, you have seen filters on Hardy-Weinberg equilibrium using tools like PLINK.

Hail can do the same. First, let's plot the log-scaled HWE p-values per variant:

[40]:
```python
p = hl.plot.histogram(hl.log10(mt.variant_qc.p_value_hwe))
show(p)
```

There are some **extremely** bad variants here, with p-values smaller than 1e-100. Let's look at some of these variants:

[41]:
```python
rows = mt.rows()
rows.order_by(rows.variant_qc.p_value_hwe).show()
```

```
+----------------+------------+----------------+-----------+-------------------------------+---
| locus          | alleles    | rsid           |      qual | filters                       | i
+----------------+------------+----------------+-----------+-------------------------------+---
| locus<GRCh38>  | array<str> | str            |   float64 | set<str>                      |
+----------------+------------+----------------+-----------+-------------------------------+---
| chr1:125165544 | ["G","T"]  | NA             | -1.00e+01 | {"InbreedingCoeff"}           |
| chr2:94565722  | ["T","C"]  | "rs1273701572" | -1.00e+01 | {"InbreedingCoeff"}           |
| chr2:113591601 | ["G","A"]  | "rs6419550"    | -1.00e+01 | {"InbreedingCoeff"}           |
| chr3:77780883  | ["A","T"]  | "rs75079699"   | -1.00e+01 | {"InbreedingCoeff"}           |
| chr5:46433521  | ["G","C"]  | NA             | -1.00e+01 | {"InbreedingCoeff"}           |
| chr5:170833888 | ["C","T"]  | "rs79126011"   | -1.00e+01 | {"InbreedingCoeff"}           |
| chr9:63771196  | ["C","T"]  | "rs4310313"    | -1.00e+01 | {"InbreedingCoeff"}           |
| chr10:46546502 | ["G","A"]  | "rs3127692"    | -1.00e+01 | {"InbreedingCoeff"}           |
| chr17:21291540 | ["G","A"]  | "rs72840058"   | -1.00e+01 | {"InbreedingCoeff"}           |
| chr17:21302646 | ["A","G"]  | "rs62057725"   | -1.00e+01 | {"InbreedingCoeff"}           |
| chr17:21414148 | ["G","A"]  | "rs79180191"   | -1.00e+01 | {"InbreedingCoeff"}           |
| chr17:21848624 | ["T","C"]  | "rs1228136826" | -1.00e+01 | {"InbreedingCoeff"}            |
| chr17:22124419 | ["T","C"]  | "rs1447696797" | -1.00e+01 | {"InbreedingCoeff"}            |
| chr17:22137702 | ["G","A"]  | "rs77080468"   | -1.00e+01 | {"InbreedingCoeff"}            |
| chr17:22150402 | ["C","T"]  | "rs374143516"  | -1.00e+01 | {"InbreedingCoeff"}            |
| chr17:22156917 | ["A","G"]  | "rs78017709"   | -1.00e+01 | {"InbreedingCoeff"}            |
| chr17:26970132 | ["G","C"]  | NA             | -1.00e+01 | {"InbreedingCoeff"}            |
| chr20:29118659 | ["A","C"]  | "rs1329786266" | -1.00e+01 | NA                             |
| chr20:30424474 | ["T","C"]  | "rs1375387936" | -1.00e+01 | {"AS_VQSR","InbreedingCoeff"} |
| chr20:30824828 | ["G","T"]  | NA             | -1.00e+01 | {"InbreedingCoeff"}            |
| chr21:5218463  | ["G","A"]  | "rs1296322493" | -1.00e+01 | {"InbreedingCoeff"}            |
| chr21:10380584 | ["C","T"]  | "rs75142325"   | -1.00e+01 | {"InbreedingCoeff"}            |
| chr21:10801323 | ["G","C"]  | "rs62219214"   | -1.00e+01 | {"InbreedingCoeff"}            |
| chr21:10810094 | ["G","C"]  | "rs28827310"   | -1.00e+01 | {"InbreedingCoeff"}            |
+----------------+------------+----------------+-----------+--------------------------------+---

+-----------------------+------------------+------------+------------+-----------+--------------
| info.AS_ReadPosRankSum | info.AS_pab_max | info.AS_QD | info.AS_MQ |   info.QD | info.AS_MQRan
+-----------------------+------------------+------------+------------+-----------+--------------
|                float64 |          float64 |    float64 |    float64 |   float64 |            flo
+-----------------------+------------------+------------+------------+-----------+--------------
|              5.29e-01 |         1.00e+00 |   2.17e+01 |   5.92e+01 | 2.17e+01 |          -1.77
|              1.06e+00 |         1.00e+00 |   1.52e+01 |   5.38e+01 | 1.52e+01 |          -5.61
|              9.33e-01 |         1.00e+00 |   1.06e+01 |   4.69e+01 | 1.06e+01 |          -2.29
|             -1.42e-01 |         1.00e+00 |   1.88e+01 |   6.00e+01 | 1.88e+01 |          -6.00
|              9.49e-01 |         1.00e+00 |   1.44e+01 |   5.32e+01 | 1.44e+01 |          -2.19
|             -7.80e-02 |         1.00e+00 |   2.04e+01 |   5.99e+01 | 2.04e+01 |          -1.60
|              5.55e-01 |         1.00e+00 |   5.72e+00 |   4.26e+01 | 5.72e+00 |          -6.74
|              4.96e-01 |         1.00e+00 |   1.39e+01 |   6.00e+01 | 1.39e+01 |           0.00
|              3.76e-01 |         1.00e+00 |   1.93e+01 |   6.00e+01 | 1.93e+01 |           0.00
|              7.45e-01 |         1.00e+00 |   1.60e+01 |   6.00e+01 | 1.60e+01 |           1.10
|              1.89e-01 |         1.00e+00 |   1.61e+01 |   5.37e+01 | 1.61e+01 |           1.31
|             -2.51e-01 |         1.00e+00 |   2.73e+01 |   5.18e+01 | 2.73e+01 |          -4.79
|             -2.47e-01 |         1.00e+00 |   2.92e+01 |   5.79e+01 | 2.92e+01 |          -1.13
|              9.57e-01 |         1.00e+00 |   2.41e+01 |   5.61e+01 | 2.41e+01 |          -1.65
|              2.45e-01 |         1.00e+00 |   2.59e+01 |   5.06e+01 | 2.59e+01 |          -7.60
|              7.78e-01 |         1.00e+00 |   2.93e+01 |   4.99e+01 | 2.93e+01 |          -7.64
|              1.28e+00 |         1.00e+00 |   1.27e+01 |   5.16e+01 | 1.27e+01 |          -5.48
|              2.04e-01 |         1.00e+00 |   2.81e+01 |   5.11e+01 | 2.81e+01 |          -6.26
|              1.39e+00 |         1.00e+00 |   1.66e+01 |   4.89e+01 | 1.66e+01 |          -7.50
|              3.10e-02 |         1.00e+00 |   2.03e+01 |   4.87e+01 | 2.03e+01 |          -7.80
|              1.12e+00 |         1.00e+00 |   1.46e+01 |   5.12e+01 | 1.46e+01 |          -9.54
|              1.07e+00 |         1.00e+00 |   1.86e+01 |   5.73e+01 | 1.86e+01 |           7.92
|              6.22e-01 |         1.00e+00 |   1.08e+01 |   5.33e+01 | 1.08e+01 |          -8.62
|              1.76e+00 |         1.00e+00 |   1.53e+01 |   5.30e+01 | 1.53e+01 |          -8.21
+-----------------------+------------------+------------+------------+-----------+--------------

+------------------------------------+--------------+-------------+----------+-----------------
| info.AS_SB_TABLE                   | info.AS_VarDP | info.AS_SOR | info.SOR | info.transmit
```

```
+-------------------------------------+--------------+-------------+----------+----------------
| array<int32>                        |        int32 |     float64 |  float64 |
+-------------------------------------+--------------+-------------+----------+----------------
| [2530759,2658686,4674032,5445216]   |     15307914 |    8.02e-01 | 8.02e-01 |
| [2744392,2448688,2632227,938989]    |      8764281 |    1.98e+00 | 1.98e+00 |
| [3447869,3213349,2778556,2338468]   |     11778240 |    8.00e-01 | 8.00e-01 |
| [2431382,2598153,2418400,2106424]   |      9554359 |    7.86e-01 | 7.86e-01 |
| [2362886,2344431,2204225,1144085]   |      8055567 |    1.54e+00 | 1.54e+00 |
| [1038041,2758564,1388629,2553316]   |      7738550 |    3.91e-01 | 3.91e-01 |
| [9577761,6255003,2256377,1811696]   |     19900836 |    5.08e-01 | 5.08e-01 |
| [2875919,2474509,3025151,2601452]   |     10976952 |    6.94e-01 | 6.94e-01 |
| [2818670,2658428,2716858,2427104]   |     10621059 |    7.49e-01 | 7.49e-01 |
| [2745828,2529213,2725172,2431983]   |     10432195 |    7.25e-01 | 7.25e-01 |
| [2321338,2589743,1394068,2129163]   |      8434310 |    1.06e+00 | 1.06e+00 |
| [2496577,2373285,4567711,5490225]   |     14927795 |    8.54e-01 | 8.54e-01 |
| [2221011,1957977,5510476,4187514]   |     13876953 |    8.53e-01 | 8.53e-01 |
| [2528293,2556611,5295354,3029366]   |     13409621 |    1.39e+00 | 1.40e+00 |
| [2484580,2523034,3043663,5751562]   |     13802821 |    1.50e+00 | 1.50e+00 |
| [2561863,2554369,6616266,5953375]   |     17685857 |    8.01e-01 | 8.01e-01 |
| [2542373,2345840,924998,1716039]    |      7528980 |    1.46e+00 | 1.46e+00 |
| [71917,68939,179325,119964]         |       440145 |          NA |       NA |
| [2750906,2663080,6547727,313127]    |     12274837 |    6.02e+00 | 6.02e+00 |
| [2551010,2575757,4836637,4862321]   |     14824547 |    6.89e-01 | 6.89e-01 |
| [6610464,6405873,5996632,5422600]   |     24435387 |    7.65e-01 | 7.65e-01 |
| [5815928,4934192,9364131,10088334]  |     30202570 |    6.31e-01 | 6.31e-01 |
| [10422190,10508662,6732289,7564224] |     35227357 |    8.07e-01 | 8.07e-01 |
| [8687734,9712091,5974764,6967867]   |     31342455 |    7.36e-01 | 7.36e-01 |
+-------------------------------------+--------------+-------------+----------+----------------

+---------------------+-------------------------+--------------------------+----------------
| info.InbreedingCoeff | variant_qc.dp_stats.mean | variant_qc.dp_stats.stdev | variant_qc.dp_s
+---------------------+-------------------------+--------------------------+----------------
|             float64 |                 float64 |                  float64 |
+---------------------+-------------------------+--------------------------+----------------
|           -9.98e-01 |                9.05e+01 |                 2.26e+01 |
|           -9.99e-01 |                5.63e+01 |                 1.36e+01 |
|           -1.00e+00 |                9.17e+01 |                 1.96e+01 |
|           -1.00e+00 |                5.81e+01 |                 1.15e+01 |
|           -9.97e-01 |                4.87e+01 |                 1.02e+01 |
|           -1.00e+00 |                4.53e+01 |                 9.72e+00 |
|           -9.98e-01 |                1.08e+02 |                 2.95e+01 |
|           -9.91e-01 |                7.18e+01 |                 1.52e+01 |
|           -1.00e+00 |                6.83e+01 |                 1.17e+01 |
|           -1.00e+00 |                7.04e+01 |                 1.26e+01 |
|           -1.00e+00 |                5.73e+01 |                 1.10e+01 |
|           -9.88e-01 |                8.11e+01 |                 1.72e+01 |
|           -9.97e-01 |                8.69e+01 |                 1.73e+01 |
|           -1.00e+00 |                8.75e+01 |                 1.87e+01 |
|            -1.00e+00 |                 8.88e+01 |                  1.82e+01 |
|            -1.00e+00 |                 1.10e+02 |                  2.12e+01 |
|            -9.95e-01 |                 5.75e+01 |                  1.09e+01 |
|            -1.00e+00 |                 9.78e+01 |                  1.70e+01 |
|            -1.00e+00 |                 7.15e+01 |                  2.23e+01 |
|            -1.00e+00 |                 9.41e+01 |                  1.57e+01 |
|            -1.00e+00 |                 1.59e+02 |                  3.13e+01 |
|            -1.00e+00 |                 1.93e+02 |                  3.53e+01 |
|            -1.00e+00 |                 2.01e+02 |                  4.38e+01 |
|            -1.00e+00 |                 1.90e+02 |                  3.43e+01 |
+----------------------+--------------------------+---------------------------+---------------

+-------------------------+-------------------------+---------------+---------------------+---
| variant_qc.gq_stats.min | variant_qc.gq_stats.max | variant_qc.AC | variant_qc.AF       | va
+-------------------------+-------------------------+---------------+---------------------+---
|                 float64 |                 float64 | array<int32>  | array<float64>      |
+-------------------------+-------------------------+---------------+---------------------+---
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.20e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
|                9.90e+01 |                9.90e+01 | [385,385]     | [5.00e-01,5.00e-01] |
+-------------------------+-------------------------+---------------+---------------------+---

+-------------------------+-----------------------+------------------+----------------------+-
| variant_qc.n_not_called | variant_qc.n_filtered | variant_qc.n_het | variant_qc.n_non_ref | v
+-------------------------+-----------------------+------------------+----------------------+-
|                   int64 |                 int64 |            int64 |                int64 |
+-------------------------+-----------------------+------------------+----------------------+-
```

```
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
|                       0 |                      0 |              385 |                  385 |
+-------------------------+------------------------+------------------+----------------------+--
showing top 24 rows
```

9.0.2 Exercise

There's a lot of information in the above output. Take a moment to look through, and remember, these are **bad-quality variants**. Can you see why these variants had such low HWE p-values? *Hint: scroll all the way to the right to the variant_qc output.*

9.0.3 The VCF `FILTERS` field

You're not entirely on your own for variant QC – most variant calling software generates its own filtering annotations that are present in the `FILTERS` field of the VCF. Lines of the VCF might have reasons to be filtered, or might be `PASS`. Above, almost every one of these bad variants has the `"InbreedingCoeff"` filter listed in its `mt.filters` field!

We can remove all of these pre-filtered variants by keeping only variants which have no filters applied.

[42]:
```
mt = mt.filter_rows(hl.len(mt.filters) == 0)
```

We can then compute the final sample and variant count:

[43]:
```
mt.count()
```

[43]:
```
(8538, 385)
```

How many variants did you lose from your variant QC?

9.1 Checkpoint #3.

You've finished the QC materials!

10 8. Association Testing and PCA

The goal of a GWAS is to find associations between genotypes and a trait of interest.

First, we filter to common variants (those with a minor allele frequency over 1%). GWAS is not well-powered to detect signal from extremely rare variants, like those only observed in one individual.

Our filter below removes variants where the minimum value of AF is below 1%. The AF array computed by `hl.variant_qc` contains one value for each allele, **including the reference**, and sums to 1.0.

A variant with 5% MAF might have AF values `[0.95, 0.05]` or `[0.05, 0.95]`. A variant with 0.1% MAF might have AF values `[0.999, 0.001]` or `[0.001, 0.999]`. This filter ensures that we account for both cases.

[44]:
```
base = mt
mt = base.filter_rows(hl.min(base.variant_qc.AF) > 0.01)
```

[45]:
```
mt.count()
```

[45]:
```
(5862, 385)
```

How many variants did you lose from your common variant filter?

Remember that in a GWAS we're running independent association tests on each variant.

In Hail, the method we use is hl.linear_regression_rows. Why isn't this called `hl.gwas`? Modularity! There are applications for this statistical method other than genome wide association studies.

We will start by using the `sleep_duration` phenotype. At the end, you will have a chance to try the others (you can do that by editing the string below to one of the other phenotypes and rerunning the rest of the notebook). The other phenotypes are:

- 'tea_intake_daily'
- 'general_happiness'
- 'screen_time_per_day'

[46]:
```
phenotype = 'sleep_duration'
```

[47]:
```
gwas = hl.linear_regression_rows(y=mt.sample_data[phenotype],
                                 x=mt.GT.n_alt_alleles(),
                                  covariates=[1.0]  # intercept term
                                 )
```

```
2021-06-16 16:52:17 Hail: INFO: linear_regression_rows: running on 385 samples
for 1 response variable y,
    with input variable x, and 1 additional covariate…
```

The resulting output from the line above is a `table`. Let's look at what the table looks like.

Out of all of the fields, we would recommend focusing your understanding of the p-value and beta effect when determining if you have a GWAS signal.

However, keep in mind that the results thus far may need your model to be adjusted.

[48]:
```
gwas.show()
```

```
+---------------+------------+-------+----------+---------------+-----------+----------------+-
| locus         | alleles    |     n |    sum_x | y_transpose_x |      beta | standard_error |
+---------------+------------+-------+----------+---------------+-----------+----------------+-
| locus<GRCh38> | array<str> | int32 |  float64 |       float64 |   float64 |        float64 |
+---------------+------------+-------+----------+---------------+-----------+----------------+-
| chr1:17379    | ["G","A"]  |   385 | 1.01e+01 |      5.18e+01 | -9.35e-01 |       4.64e-01 |
| chr1:1130877  | ["C","G"]  |   385 | 1.20e+01 |      7.20e+01 | -2.41e-02 |       4.27e-01 |
| chr1:1491494  | ["G","A"]  |   385 | 9.00e+00 |      5.30e+01 | -1.38e-01 |       4.91e-01 |
| chr1:1618118  | ["G","A"]  |   385 | 1.29e+02 |      7.67e+02 | -9.27e-02 |       1.28e-01 |
| chr1:3008858  | ["A","G"]  |   385 | 2.33e+02 |      1.44e+03 |  1.52e-01 |       1.01e-01 |
| chr1:3254545  | ["T","C"]  |   385 | 2.83e+02 |      1.76e+03 |  2.53e-01 |       9.75e-02 |
| chr1:3779436  | ["T","C"]  |   385 | 3.72e+02 |      2.17e+03 | -2.60e-01 |       8.92e-02 |
| chr1:4156721  | ["C","T"]  |   385 | 2.31e+01 |      1.46e+02 |  2.76e-01 |       2.99e-01 |
| chr1:4510922  | ["C","T"]  |   385 | 9.30e+01 |      5.67e+02 |  7.22e-02 |       1.50e-01 |
| chr1:4748394  | ["G","A"]  |   385 | 3.18e+02 |      1.85e+03 | -3.37e-01 |       1.02e-01 |
| chr1:5620931  | ["C","G"]  |   385 | 1.23e+02 |      7.00e+02 | -3.59e-01 |       1.35e-01 |
| chr1:5647439  | ["A","T"]  |   385 | 1.54e+02 |      9.78e+02 |  4.05e-01 |       1.29e-01 |
| chr1:6135622  | ["C","T"]  |   385 | 4.07e+02 |      2.50e+03 |  2.31e-01 |       9.77e-02 |
| chr1:8930685  | ["T","A"]  |   385 | 3.44e+01 |      2.08e+02 |  2.92e-02 |       2.27e-01 |
| chr1:9381890  | ["C","A"]  |   385 | 3.03e+02 |      1.87e+03 |  2.29e-01 |       1.08e-01 |
| chr1:10273901 | ["G","T"]  |   385 | 4.32e+02 |      2.57e+03 | -1.75e-01 |       1.04e-01 |
| chr1:10473368 | ["C","T"]  |   385 | 3.70e+01 |      2.08e+02 | -3.42e-01 |       2.20e-01 |
| chr1:10880445 | ["A","C"]  |   385 | 2.14e+02 |      1.30e+03 |  6.90e-02 |       1.06e-01 |
| chr1:11783783 | ["G","A"]  |   385 | 4.00e+01 |      2.32e+02 | -2.36e-01 |       2.36e-01 |
| chr1:12182736 | ["T","G"]  |   385 | 1.54e+02 |      9.15e+02 | -9.52e-02 |       1.26e-01 |
| chr1:12714480 | ["G","A"]  |   385 | 1.70e+01 |      1.00e+02 | -1.31e-01 |       3.41e-01 |
| chr1:12862886 | ["C","T"]  |   385 | 1.71e+01 |      1.01e+02 | -1.32e-01 |       3.41e-01 |
| chr1:13383554 | ["G","A"]  |   385 | 6.70e+01 |      3.85e+02 | -3.03e-01 |       1.85e-01 |
| chr1:13436332 | ["C","G"]  |   385 | 9.00e+00 |      5.00e+01 | -4.79e-01 |       4.90e-01 |
+---------------+------------+-------+----------+---------------+-----------+----------------+-
showing top 24 rows
```

11 9. Visualization

Let's visualize our association test results from the linear regression. We can do so by creating 2 common plots: a Manhattan plot and a Q-Q plot.

We'll start with the Manhattan plot:

[49]:
```
p = hl.plot.manhattan(gwas.p_value)
show(p)
```

11.0.1 Manhattan plot interpretation

The dashed red line is the line for genome-wide significance with a typical Bonferroni correction: 5e-8.

We have several points above the line – mouse over to see the loci they refer to. This means we're ready to publish our sleep GWAS in *Nature*, right?

…right?

11.0.2 Q-Q plot

The plot that accompanies a Manhattan plot is the Q-Q (quantile-quantile) plot.

[50]:
```
p = hl.plot.qq(gwas.p_value)
show(p)
```

```
2021-06-16 16:52:29 Hail: INFO: Ordering unsorted dataset with network shuffle
```

The Q-Q plot can clearly indicate widespread inflation or deflation of p-values. Is either of those properties present here?

11.0.3 Exercise

Is this GWAS well controlled? Discuss with your group.

Wikipedia has a good description of genomic control estimation (lambda GC) to read later.

11.1 Checkpoint #4

You've used Hail to run a basic GWAS! It's more complicated than running in PLINK, but hopefully you can see that each part of the process is flexible and configurable!

12 10. Confounded! PCA to the rescue.

If you've done a GWAS before, you've probably included a few other covariates that might be confounders – age, sex, and principal components.

Principal components are a measure of genetic ancestry, and can be used to control for population stratification.

Principal component analysis (PCA) is a very general statistical method for reducing high dimensional data to a small number of dimensions which capture most of the variation in the data. Hail has the function pca for performing generic PCA.

PCA typically works best on normalized data (e.g. mean centered). Hail provides the specialized function hwe_normalized_pca which first normalizes the genotypes according to the Hardy-Weinberg Equilibium model.

[51]:
```
pca_eigenvalues, pca_scores, pca_loadings = hl.hwe_normalized_pca(mt.GT,␣
↪compute_loadings=True)
```

```
2021-06-16 16:52:43 Hail: INFO: hwe_normalized_pca: running PCA using 5862
variants.
2021-06-16 16:53:04 Hail: INFO: pca: running PCA with 10 components…
2021-06-16 16:53:19 Hail: INFO: Coerced sorted dataset
```

[52]:
```
pca_scores = pca_scores.checkpoint('resources/pca_scores.ht', overwrite=True)
```

```
2021-06-16 16:53:21 Hail: INFO: Ordering unsorted dataset with network shuffle
2021-06-16 16:53:27 Hail: INFO: wrote table with 385 rows in 16 partitions to
resources/pca_scores.ht
    Total size: 36.15 KiB
    * Rows: 36.14 KiB
    * Globals: 11.00 B
    * Smallest partition: 21 rows (1.97 KiB)
    * Largest partition:  27 rows (2.53 KiB)
```

The pca function returns three things: * **eigenvalues**: an array of doubles * **scores**: a table keyed by sample * **loadings**: a table keyed by variant

The **loadings** are the *principal directions*, each of which is a weighted sum of variants (a "virtual variant"). By default, `hwe_normalized_pca` gives us 10 principal directions.

12.0.1 Joining computed PCs onto our matrix table

Using the same syntax as we used to join the sample data table `sd` above, we can join the computed scores onto our matrix table:

[53]:
```
mt = mt.annotate_cols(pca = pca_scores[mt.s])
```

12.1 Visualize principal components

Let's make plots!

[54]:
```
p1 = hl.plot.scatter(x=mt.pca.scores[0],
                     y=mt.pca.scores[1],
                     xlabel='PC1',
                     ylabel='PC2',
                     hover_fields={'ID': mt.s, 'pop': mt.sample_data.pop},
                     label=mt.sample_data.continental_pop,
                     size=8)

 p2 = hl.plot.scatter(x=mt.pca.scores[2],
                     y=mt.pca.scores[3],
                     xlabel='PC3',
                     ylabel='PC4',
                     hover_fields={'ID': mt.s, 'pop': mt.sample_data.pop},
                     label=mt.sample_data.continental_pop,
                     size=8)

 show(p1)
 show(p2)
```

12.1.1 Exercise

Compare your plots with your group. **Besides the randomly-chosen coloration**, do they look the same? If not, how do they differ?

12.2 Checkpoint #5

You've finished the PCA materials!

13 11. Control confounders and run another GWAS

Now that we have computed principal components and saved it into our `mt`, let's run another GWAS that includes PCs as covariates to account for confounding due to population stratification.

[55]:
```
gwas2 = hl.linear_regression_rows(
    y=mt.sample_data[phenotype],
    x=mt.GT.n_alt_alleles(),
    covariates=[1.0, mt.pca.scores[0], mt.pca.scores[1], mt.pca.scores[2]])
```

```
2021-06-16 16:54:04 Hail: INFO: linear_regression_rows: running on 385 samples
for 1 response variable y,
    with input variable x, and 4 additional covariates…
```

[56]:
```
p = hl.plot.qq(gwas2.p_value)
show(p)
```

```
2021-06-16 16:54:09 Hail: INFO: Ordering unsorted dataset with network shuffle
```

The above Q-Q plot indicates a much better-controlled GWAS. Let's try the Manhattan plot:

[57]:
```
p = hl.plot.manhattan(gwas2.p_value)
show(p)
```

13.0.1 Are there any significant hits? Do you trust these results? Why or why not?

Discuss with your group - roughly how many samples would you need to discover rare variant signal? polygenic signal?

13.0.2 Exercise

Go back to the “Association Testing and PCA” section above and rerun the rest of the notebook with a different phenotype. You will need to edit the cell where `phenotype = 'sleep_duration'`, but rerun all the cells starting from the “Association testing and PCA” heading beginning with `base = mt....`

13.0.3 Exercise

Change the “gwas2” cell to experiment with how many principal components are needed as covariates to properly control this GWAS. How many are needed here in our tiny simulated example? How many are needed in a typical GWAS?

13.1 The end!

If you still have time and desire, there is another practical: `02-Hail-rare-variant-analysis`. You can access that notebook from the Jupyter home page. But please don’t rush straight to that notebook – you might learn more reviewing the code you ran here, digesting it, experimenting with the code, and asking questions!

14 When the workshop ends and you return to your life

The hosted notebook service that is running this notebook will be turned off in a few hours, but you can continue using Hail!

The Hail website has a page with information about getting started. If you have a MacOS or Linux computer, or have access to a Linux server, you can run Hail.

It is also possible to run Hail on Google Cloud. See the video lectures for guidance on how to do that, or reach out to the team for help!

14.1 The Hail community

Although Hail has a steeper learning curve than many command-line tools, you won’t be learning it alone! Hail has a forum and Zulip chatroom full of like-minded users of all experience levels. Please stop by to say hello!

[ ]:

```

```