Portal/Hail API

API specification

We propose a total of three endpoints. The first endpoint provides metadata that can be used to request a calculation.

The second and third endpoints are mutually exclusive: the former provides information needed to perform a calculation in the browser, while the latter provides pre-computed results. A helper function is provided to format live calculations to look like the final results- this allows both calculation methods to be used interchangeably and processed in the same way.

Retrieve available datasets/masks

A genotype dataset is a set of variant genotypes and samples (e.g. a VCF.)

A phenotype dataset is a set of phenotypes and samples (PED or tab-delimited file). These phenotypes are linked to one or more genotype datasets.

A mask is defined by:

1. Variant filtering criteria
2. Variant grouping criteria

Variant filtering:

- Allele frequency
- Annotation about a variant (protein-truncating, loss-of-function, etc.)

Grouping:

- Gene
- Chromatin state regions (enhancer, silencer, promoter, insulator, etc.)
- KEGG pathways (variants within genes that belong to the pathway)
- Gene ontology terms (similar to KEGG)

Masks would be available per genotype dataset.

For now, a description of the variant filtering and grouping criteria will probably just have to be plain text without some type of formal grammar.

Request

`GET /api/aggregation/metadata`

Response

```
{
  "data": [
    {
      "genotypeDataset": 1,
      "description": "52K Exomes",
      "genomeBuild": "GRCh37",
      "masks": [
        {
          "id": 1,
          "name": "PTV",
          "description": "Protein truncating variants",
          "groupType": "GENE",
          "identifierType": "ENSEMBL"
        },
        {
          "id": 2,
          "name": "PTV & LoF & AF<0.05",
          "description": "Protein truncating variants with AF < 0.05",
          "groupType": "GENE",
          "identifierType": "ENSEMBL"
        },
        {
          "id": 3,
          "name": "PTV & LoF & AF<0.05",
          "description": "Protein truncating and loss-of-function variants with AF < 0.05",
          "groupType": "GENE",
          "identifierType": "ENSEMBL"
        }
      ],
      "phenotypeDatasets": [
        {
          "phenotypeDataset": 1,
          "description": "52K Exomes - Cardiovascular Traits",
          "phenotypes": [
            {
              "name": "ldl",
              "description": "Low density lipoproteins"
            },
            {
              "name": "hdl",
              "description": "High density lipoproteins"
            }
          ]
        },
        {
          "phenotypeDataset": 2,
          "description": "52K Exomes - Diabetes-related Traits",
          "phenotypes": [
            {
              "name": "T2D",
              "description": "Type 2 diabetes"
            },
            {
              "name": "FG",
              "description": "Fasting glucose"
            }
          ]
        }
      ]
    }
  ]
}
```

Retrieve covariance in region given dataset/mask(s)

Request

Only requesting scores and covariance for the "PTV" mask just to save some space.

```
POST /api/aggregation/covariance
```

```
{
  "chrom": "6",
  "start": 1,
  "stop": 100000,
  "genotypeDataset": 1,
  "phenotypeDataset": 1,
  "phenotype": "rand_qt",
  "samples": "ALL",
  "genomeBuild": "GRCh37",
  "masks": [...]
}
```

The `masks` key can be either a numeric ID, representing a server-side mask file (for example, if a mask was already computed for a particular genotype dataset), or it can be a mask definition itself. This allows the client to send a specific list of variants to the server.

If we want a mask already defined on the server, send a list of IDs:

```
{
  "masks": [1, 2, 3]
}
```

Or instead of a using server-side masks, supply a list of your own generated in the browser or client-side:

```
{
  "masks": [
    {
      "id": 10,
      "name": "PTV+LOF<0.01",
      "description": "A mask generated by the user in the browser, including only variants that are PTV
      "genome_build": "GRCh37",
      "group_type": "GENE",
      "identifier_type": "ENSEMBL",
      "groups": {
        "ENSG000001": ["1:1_A/G", "1:2_C/T", ...],
        "ENSG000002": ["1:43_G/C", ...]
      }
    }
  ]
}
```

In this case, the ID can be any integer, so long as it is unique across all of the masks sent in the request.

Response

We want to be able to retrieve as much of the masks/covariance data all at once to minimize round trips to the server. This structure would allow retrieving covariance for multiple masks/groups all at once.

Note that all scores and covariances are assumed to be counting towards the **alternate allele**. It is also critical that alternate allele frequencies are included so as to be able to orient scores/covariances towards the minor allele when performing the aggregation tests.

The data section of this payload contains two distinct kinds of entities:

- **groups** represent the information required to perform analysis on one or more sets of variants. Often, these variants are based on a particular region (such as `groupType: 'interval'` or `groupType: 'gene'`), filtered according to a specific mask. In the example below, only one set is specified: all protein truncating variantsa (PTV) that lie within the region that defines a gene with ENSEMBL ID `ENSG000001`.
- **variants** contains basic information about individual variants. Some information (such as `altFreq`) is needed for calculations. Optionally, this section may present additional information that can be used to analyze and interpret the calculation results. The same variant may appear in many groups, but is only listed here once.

```
{
  "data": {
    "genotypeDataset": 42,
    "phenotypeDataset": 8,
    "phenotype": "T2D",
    "sigmaSquared": 0.08,
    "nSamples": 3550,
    "variants": [
      {
        "variant": "2:21228642_G/A",
        "altFreq": 0.033,
        "pvalue": 0.000431,
        "score": 0.1
      }
    ],
    "groups": [
      {
        "groupType": "GENE",
        "group": "ENSG000001",
        "mask": 1,
        "variants": ["2:21228642_G/A"],
        "covariance": [0.3]
      }
    ]
  }
}
```

Retrieving pre-computed aggregation test results

In the case where aggregation tests have already been pre-computed for some of the datasets server-side, this endpoint provides access to calculation results (and associated information). This endpoint is mutually exclusive with the one above: if results have already been computed, then there is no need to retrieve the raw data (such as covariance) used as input to the calculation.

Otherwise, the payloads are superficially similar, so that the same visualization experience can work with both on-the-fly and precomputed results. `raremetal.js` has a helper function that will format in-browser calculations as the response below.

Request

Looks identical to the same request used to retrieve covariance.

`POST /api/aggregation/results`

```
{
  "chrom": "6",
  "start": 1,
  "stop": 100000,
  "genotypeDataset": 1,
  "phenotypeDataset": 1,
  "phenotype": "rand_qt",
  "samples": "ALL",
  "genomeBuild": "GRCh37",
  "masks": [
    1
  ]
}
```

Response

If results were already available server-side, say if they were pre-computed, we could accept a response of this format and be able to show the results within LZ without performing any aggregation test computations.

The two types of data in this endpoint are similar to those described in the `covariance` endpoint above. Instead of covariance data, `groups` in this endpoint provide calculation results.

```
{
  "data": {
    "genotypeDataset": 42,
    "description": "52K Exomes",
    "phenotypeDataset": 1,
    "phenotype": "T2D",
    "variants": [
      {
        "variant": "2:21228642_G/A",
        "altFreq": 0.033,
        "pvalue": 0.000431
      }
    ],
    "groups": [
      {
        "groupType": "GENE",
        "group": "ENSG000001",
        "mask": "PTV",
        "variants": ["2:21228642_G/A"],
        "test": "burden",
        "pvalue": 1.8e-09,
        "stat": 0.1
      }
    ]
  }
}
```