Google Cloud Queries on 1000 Genomes Dataset

Advanced Guide to Analyzing Variants with BigQuery

```
SELECT COUNT(1) AS number_of_rows
FROM
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823`
```

This query returns the number of rows that are in the table and labels this column 'number_of_rows'. The output is 105,923,159. The scan size output 0.0gb.

```
SELECT
  SUM(ARRAY_LENGTH(call)) AS number_of_calls
FROM

`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823`
```

This query outputs the length of each call array which is 182,104,652. The call column is an array of variant cells that contain the name and other values. The scan size output 0.0gb.

```
SELECT
  COUNT(call) AS number_of_calls
FROM
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v, v.call
```

The estimated query size of this was 0.0gb. This outputs 182,104,652 as it is just another method of counting the calls by unnesting this column through a join.

```
SELECT
  COUNT(call.name) AS number_of_calls
FROM
```

```
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v, v.call call
```

The estimated size for this query was 1.52gb. This outputs the number of ‘name’ values within the ‘call’ column which is the same as the previous two (182,104,652) as there should be one ‘name’ for each call.

```
SELECT
  COUNT(1) AS number_of_real_variants
FROM

`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v, v.call call
WHERE
  EXISTS (SELECT 1
            FROM UNNEST(v.alternate_bases) AS alt
           WHERE
            alt.alt NOT IN ("<NON_REF>", "<*>"))
```

This query takes approximately 0.53gb. It counts the number of true variants, so variants that are not labeled as <non-ref> or <*>. It returns 38,549,388.

```
SELECT
  COUNT(1) AS number_of_non_variants
FROM

`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v, v.call call
WHERE
  NOT EXISTS (SELECT 1
                FROM UNNEST(v.alternate_bases) AS alt
               WHERE                 alt.alt NOT IN (
"<NON_REF>", "<*>"))
```

This query also takes approximately 0.53gb. It counts the number of non variants, so variants that are labeled as <non-ref> or <*>. It returns 143,555,264.

```
--------------- SELECT
call.name AS call_name,
  COUNT(call.name) AS call_count_for_call_set
FROM
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v, v.call
GROUP BY
  call_name ORDER
BY
  Call_name
```

In this query 1.53 GB of data was processed. This query counts the number of variants called by each sample. It is used to count the number of rows in which each call set occurs.

```
SELECT   call.name AS
call_name,
  COUNT(call.name) AS call_count_for_call_set
FROM
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v, v.call
WHERE
  EXISTS (SELECT 1
            FROM UNNEST(v.alternate_bases) AS alt
          WHERE
            alt.alt NOT IN ("<NON_REF>", "<*>"))
GROUP BY   call_name ORDER BY
  Call_name
```

The Bytes processed for this query was 2.06 GB and completed in 4.5s. This query filtering out the non-variant segments to count just the variant rows.

```
SELECT   call.name AS
call_name,
  COUNT(call.name) AS call_count_for_call_set
FROM
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v, v.call
WHERE
  EXISTS (SELECT 1 FROM UNNEST(call.genotype) AS gt WHERE gt > 0)
  AND NOT EXISTS (SELECT 1 FROM UNNEST(call.genotype) AS gt WHERE gt < 0)
GROUP BY
call_name ORDER
BY
  Call_name
```

Query completed in 3.5s elapsed and used 4.24 GB of data. This query filter out the variants that are not considered true variants by genotype for individuals and count just the variant rows.

```
SELECT
  COUNT(DISTINCT call.name) AS number_of_callsets
FROM
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v,  v.call
```

The size of this query was 1.53 GB and completed in 3.1s elapsed. This query is used to count the samples in the table and get the value for that number of rows.

```
SELECT
call_filter,
  COUNT(call_filter) AS number_of_calls
FROM
```

```
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v,
  v.call,
  UNNEST(call.FILTER) AS call_filter
GROUP BY   call_filter ORDER BY
  Number_of_calls
```

The size used for this query was 254 MB of data This query was used for counting high-quality variants per sample and shows how to view the per-variant-call FILTER values for the dataset.

```
SELECT   reference_name,
start_position,
end_position,
reference_bases,
call.name AS call_name,
  (SELECT STRING_AGG(call_filter) FROM UNNEST(call.FILTER) AS call_filter)
AS filters,
  ARRAY_LENGTH(call.FILTER) AS filter_count FROM bigquery-public-
data.human_genome_variants.platinum_genomes_deepvariant_va
riants_20180823` v, v.call
WHERE
  EXISTS (SELECT 1 FROM UNNEST(call.FILTER) AS call_filter WHERE
call_filter = 'PASS')
  AND ARRAY_LENGTH(call.FILTER) > 1 ORDER BY   filter_count DESC,
reference_name, start_position, end_position, reference_bases,
call_name
LIMIT
  10
```

This query size was 4.28 GB . This query is used to verify If the FILTER column contains the value PASS. This also omits calls that do not contain a PASS value under FILTER and It returns zero as output.

```
SELECT   call.name AS
call_name,
  COUNT(1) AS number_of_calls
FROM
`bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_v
ariants_20180823` v, v.call
WHERE
  NOT EXISTS (SELECT 1 FROM UNNEST(call.FILTER) AS call_filter WHERE
call_filter != 'PASS') GROUP BY   call_name
ORDER BY
  Call_name
```

This query used 1.77 GB of size. This query shows how to count all calls including variants and non-variants for each call set and omits any call with a non-PASS filter and counts all high quality calls for each sample.

```
SELECT   call.name AS
call_name,   COUNT(1) AS
number_of_calls
FROM
`bigquery-public-
data.human_genome_variants.platinum_genomes_deepvariant_varian ts_20180823`
v, v.call
WHERE
 NOT EXISTS (SELECT 1 FROM UNNEST(call.FILTER) AS call_filter WHERE
call_filter != 'PASS')
 AND EXISTS (SELECT 1 FROM UNNEST(call.genotype) as gt WHERE gt >
0) GROUP BY   call_name ORDER BY
  Call_name
```

Query size for this was 4.49 GB. This query shows how to count all calls for each sample . It omits any call with a non-pass filter and only includes calls with at least one true variant.

```
SELECT
  REGEXP_REPLACE(reference_name, '^chr', '') AS chromosome,
  COUNT(reference_name) AS number_of_variant_rows
FROM
`bigquery-public-
data.human_genome_variants.platinum_genomes_deepvariant_varian ts_20180823` v
WHERE
```

```
  EXISTS (SELECT 1
            FROM UNNEST(v.call) AS call, UNNEST(call.genotype) AS gt
           WHERE gt >
0) GROUP BY
chromosome ORDER BY
chromosome
```

Estimated size for this query was 3.34 GB. The query uses the REGEXP_REPLACE function to replace the "chr" prefix string with an empty string. It then changes the GROUP BY and ORDER BY functions to use the computed chromosome alias.

```
SELECT
  reference_name,
  COUNT(reference_name) AS number_of_variant_rows
FROM
`bigquery-public-
data.human_genome_variants.platinum_genomes_deepvariant_varian ts_20180823` v
WHERE
  EXISTS (SELECT 1
            FROM UNNEST(v.call) AS call
           WHERE EXISTS (SELECT 1
                           FROM UNNEST(call.genotype) AS
gt GROUP BY   reference_name ORDER BY
  Reference_name
```

The query size was 3.34 GB. This query is used to produce a per-variant result. This allows you to query over an Array. The function returns one row for each element of Array.

Data Stories for the 1000 Genomes Project

```
SELECT
  COUNT(sample) AS all_samples,
  SUM(IF(In_Phase1_Integrated_Variant_Set = TRUE, 1, 0)) AS
samples_in_variants_table
FROM
  `bigquery-public-data.human_genome_variants.1000_genomes_sample_info`
```

This query took less than 1gb to run and it returned the total number of samples in the dataset (3500) and the number of samples which had a label of ‘true’ in the column ‘In_Phase1_Integrated_Variant_Set.’ This column determines whether or not there is a variant in the sample.

```
SELECT
gender,
gender_count,
  (gender_count / SUM(gender_count)
OVER
  (
ORDER BY
    gender_count)) AS gender_ratio
FROM (
  SELECT
gender,
    COUNT(gender) AS gender_count
  FROM
    `bigquery-public-data.human_genome_variants.1000_genomes_sample_info`
WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
gender)
```

This query took less than 1gb to run and it returned the ratio of 51.9 females over those with variants. This shows us that the set is not biased in terms of gender and variants.

```
 SELECT   population,
population_description,
population_count,
  RATIO_TO_REPORT(population_count)
OVER
  (   ORDER BY     population_count) AS
population_ratio,   super_population,
super_population_description, from(
SELECT     population,
population_description,
super_population,
super_population_description,
```

```
    COUNT(population) AS population_count,
  FROM
    [genomics-public-data:1000_genomes.sample_info]
  WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY     population,
population_description,
super_population,
    super_population_description)
```

This query shows us the breakdown of populations (including the code name, description, and count) with variants. Iberian populations in Spain are especially under represented with only 1% of the share of data. The query is about 1mb.

```
# Ratios of ethnicities grouped by gender
SELECT   population,   gender,
population_count,
  RATIO_TO_REPORT(population_count) OVER(
  PARTITION BY
population   ORDER
BY     gender)
  AS population_ratio
from(   SELECT
gender,
population,
    COUNT(population) AS population_count,
  FROM
    [genomics-public-data:1000_genomes.sample_info]
  WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
gender,
population) ORDER
```

```
BY   population,
gender
```

This query took about .4mb to run. It selects the columns population and gender and generates the population count as the column ‘population_count.’ It only selects those with variant data and outputs the ratio of genders within each population. These were mostly close to even between genders.

```
# Compute the distribution of family sizes
SELECT
num_family_members AS family_size,
COUNT(num_family_members) AS num_families_of_size
FROM (
  SELECT
family_id,
  COUNT(family_id) AS num_family_members,
  FROM
  [genomics-public-data:1000_genomes.sample_info]
  WHERE
  In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
family_id) GROUP
BY family_size
```

The scan size of this query is about .2mb. It gets the number of family members by counting the number of family_id’s for each unique id. It only selects those with variants. Most of the families were of size 1 (636) and the least were of the size 4 (3).

```
SELECT
   INTEGER(reference_name) AS chromosome,
   MIN(start) AS min,
   MAX(start) AS max
 FROM
```

```
   [genomics-public-data:1000_genomes.variants]
OMIT RECORD IF    reference_name IN ("X", "Y",
"MT")
 GROUP BY
chromosome
```

The size of this query was 400mb. It turns each reference name into an integer and labels it as ‘chromosome’ and gets the minimum start value and maximum start values. It omits those with reference names X, Y, and MT. The minimum start is on chromosome 17 (55).

```
SELECT
  INTEGER(reference_name) AS chromosome,
vt AS variant_type,
  COUNT(1) AS cnt
 FROM
   [genomics-public-data:1000_genomes.variants]
OMIT RECORD IF    reference_name IN ("X", "Y",
"MT")
 GROUP BY
chromosome,
variant_type
```

This query takes about 300mb. It reports the number of each variant type on each chromosome and groups by chromosome and variant type. The majority of these for every chromosome were of type ‘SNP’ (single nucleotide polymorphism) and next greatest amount were of type ‘Indel’ (insertion and deletion) and the least SV (structural variants).