RESEARCH

Introduction to the profile areas of data sciences: project 5

Rohan Chitte] Natalja Amiridze] Michael Onyebuchi Ohaya

Abstract

Goal of the project: Analyse data using the BigQuery system from the 1000 Genome project.

Main results of the project: The queries of the two tutorials "Advanced guide to analyzing variants using BigQuery" and "Exploring the phenotypic data" were correctly formulated and the output data of the queries could be analysed and visualised with functions of the class BigQueryHelper

Personal key learnings: We learned how to use the BigQuery system to analyze data using SQL queries and writing user defined functions(though UDF was not used in the project due to implementation issues).

Estimated working hours: 10

Project evaluation: 1

Number of words: 826

1 Scientific Background

Genomics is the study of whole genomes (the entire set of genetic material) of organisms and incorporates elements of genetics. Genomics uses a combination of recombinant DNA, DNA sequencing methods, and bioinformatics to sequence, assemble and analyse the structure and function of genomes. DNA (deoxyribonucleic acid) contains the instructions an organism needs to develop, survive and reproduce [Figure 1]. DNA is located in the cell nucleus, and other complex organisms also have a small amount of DNA in cell structures called mitochondria. It consists of 4 nucleotides A(Adenine),T(Thymine),C(Cytosine) and G(Guanine). The nucleotides are connected to each other by covalent bonds. The sequencing of these bonds is specific - i.e adenine bonds only with thymine and cytosine only with guanine. An important property of DNA is that it can replicate or make copies of itself. The DNA encodes the genetic information for the transmission of genetic material. The order or sequence of these bases determines which biological instructions are contained in a DNA strand. For example, the base sequence "ATCGTT" could instruct for blue eyes, while "ATCGCT" might instruct for brown. These order of the bases make humans genetically different from each other. Such genetic variations are observed by DNA sequencing. Interestingly, for data analysis, only a very small portion of the genome can carry meaningful information and only that region is required for analysis. Such regions are called Exomes. They are known to encode proteins and include approximately 1%–2% of the genome but are thought to contain approximately 85% of important information like disease-causing variants.

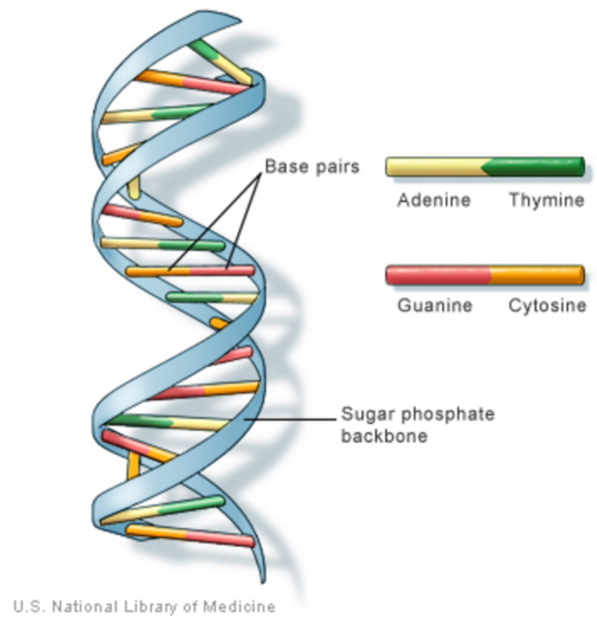


Figure 1 DNA

2 Goal

The primary goal of this project is to use BigQuery System to analyze data from the 1000 Genomes project. To analyze the data, it is vital to familiarise oneself with Genomics which is an interdisciplinary field of biology focusing on the structure, function, evolution, mapping, and editing of genomes. Further, We executed one of the Data Stories for the 1000 Genomes Project called "Exploring the Sample Information" using Google's BigQuery in python.

3 Data and Preprocessing

Genome sequencing is the process of determining the complete DNA sequence of an organism's genome at a single time. Earlier Sequencing was performed by sequencing nearly an entire human genome through the use of shotgun sequencing. technology [Figure 2]. This method involves randomly breaking up DNA sequences into lots of small pieces and then reassembling the sequence by looking for regions of overlap in order to determine the sequence of an entire genome. However, this approach of sequencing comes with some drawbacks. It requires computer processing power beyond what an ordinary laboratory would possess. It can introduce errors in the assembly process and requires a reference genome. Modern NGS (Next Generation Sequencing) Techniques, adapts to Sequencing DNA Fragments using method parallelism. This results in a faster sequencing of hundreds of millions of DNA molecules to be at a time. Such methods fall under the category of high-throughput sequencing and are capable of sequencing the entire genome in less than a day. To aim of data analysis is to restore full sequence by performing tasks of alignment and variant calling. Firstly, exomes sequence are read and then aligned to a reference genome. This is called Assembly. Secondly, individual samples are checked by variant calling function. This process which is also called Annotation, tells us where the genetic sequence of a person differs from the reference genome(average person's genome). For this project we used Google's Genomics and Google's BigQuery to perform analysis on Genomics data. Google's Genomics contains full 1,000 Genomes dataset for 3,500 individuals. This dataset includes information about individual's Ethnicity, gender, and family relationship. To execute BigQueries in python, we imported bigquery and BigQueryHelper. Later, we set project id for access to the project on google cloud platform. Since we are dealing with huge chunks of data, it is essential to check how much information a query is processing as Google's BigQuery comes with a limit for trail account. Therefore, before executing the query, we checked each query using estimate_query_size method of BigQueries which shows the size of query as output in Gigabytes. Then, to execute the queries in python we used query_to_pandas of BigQueries which executes the query and stores the result in form a pandas dataframe.

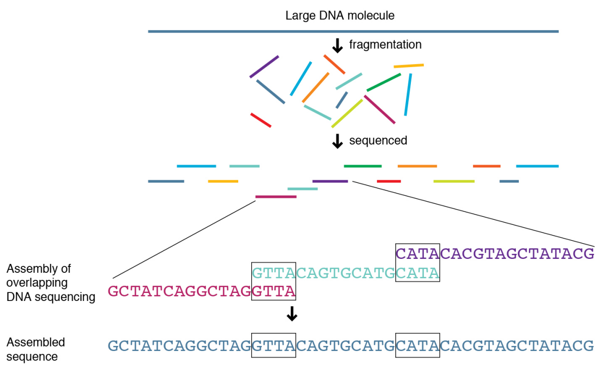


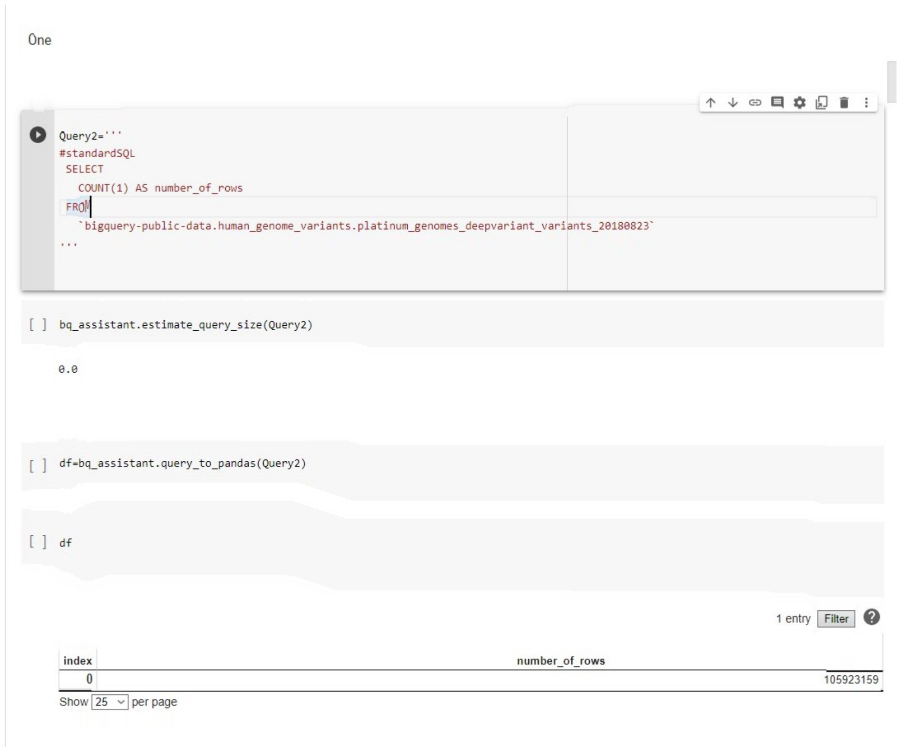
Figure 2 DNA Sequencing

4 Methods and Results

4.1 Task 1

The main goal of the first task was to redo one of BigQuery's "Advanced guide to analyzing variants using BigQuery" (at least 5 queries) in our colab notebook. In other to process the queries we had to use a helper function which was part of the prerequisite of the project and also followed the guideline of setting up a project in BigQuery and Colab platforms both from Google. The helper function simplifies common BigQuery's tasks therby removing a lot of bottle-neck while querying the Dataset. The first query basically gave us the number of rows in the dataset.

- Query 1 (see Figure 3)
- **IT Interpretation:** The query counts the number of rows in the first column of the dataset and outputs its value as number_of_rows in the displayed in figure 3
  `COUNT(1) AS number_of_rows`
- **Biological Interpretation:** The only biological interpretation here is that there is 105,923,159 sample collected in the 'platinum_genomes_deepvariant_variants_20180823' dataset



One

```
Query2='''
#standardSQL
 SELECT
   COUNT(1) AS number_of_rows
 FROM
   `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823`
'''
```

```
[ ] bq_assistant.estimate_query_size(Query2)

    0.0
```

```
[ ] df=bq_assistant.query_to_pandas(Query2)
```

```
[ ] df
```

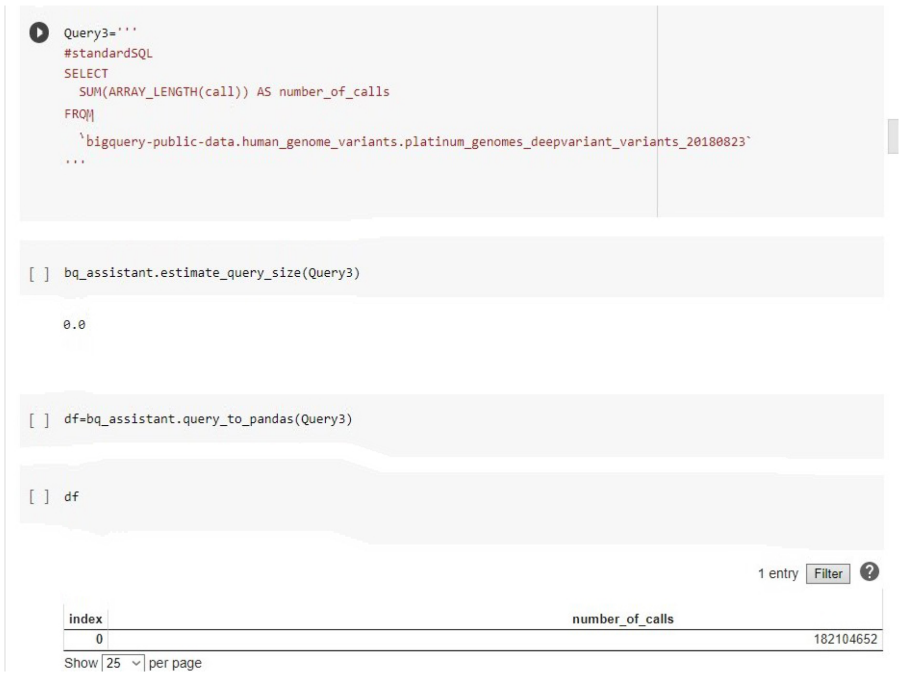
1 entry

index	number_of_rows
0	105923159

Show 25 per page

Figure 3 Tutorial1, Query 1

- Query 2 (see Figure 4)



```
Query3='''
#standardSQL
SELECT
  SUM(ARRAY_LENGTH(call)) AS number_of_calls
FROM
  `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823`
'''
```

```
[ ] bq_assistant.estimate_query_size(Query3)

    0.0
```

```
[ ] df=bq_assistant.query_to_pandas(Query3)
```

```
[ ] df
```

1 entry

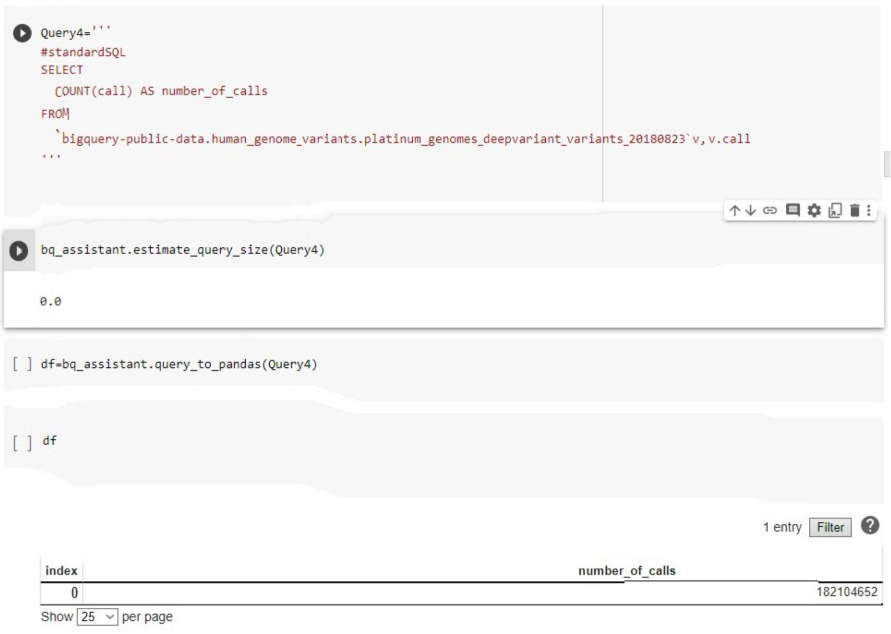
index	number_of_calls
0	182104652

Show 25 per page

Figure 4 Tutorial1, Query 2

- **IT Interpretation:** This query get the total number of variant call across all samples. It does this by getting the sum of the 'ARRAY_LENGTH' which is one of the variables in the call column of the sample. Subsequent queries separates the real-variants from the non-variants.
  `SUM(ARRAY_LENGTH(call)) AS number_of_calls`
- **Biological Interpretation:** This query gives us the total number of real_variant and non_variant in the sample.

- Query 3 (see Figure 5)



```
Query4='''
#standardSQL
SELECT
  COUNT(call) AS number_of_calls
FROM
  `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823` v, v.call
'''
```

```
bq_assistant.estimate_query_size(Query4)

0.0
```

```
[ ] df=bq_assistant.query_to_pandas(Query4)
```

```
[ ] df
```

1 entry Filter

index	number_of_calls
0	182104652

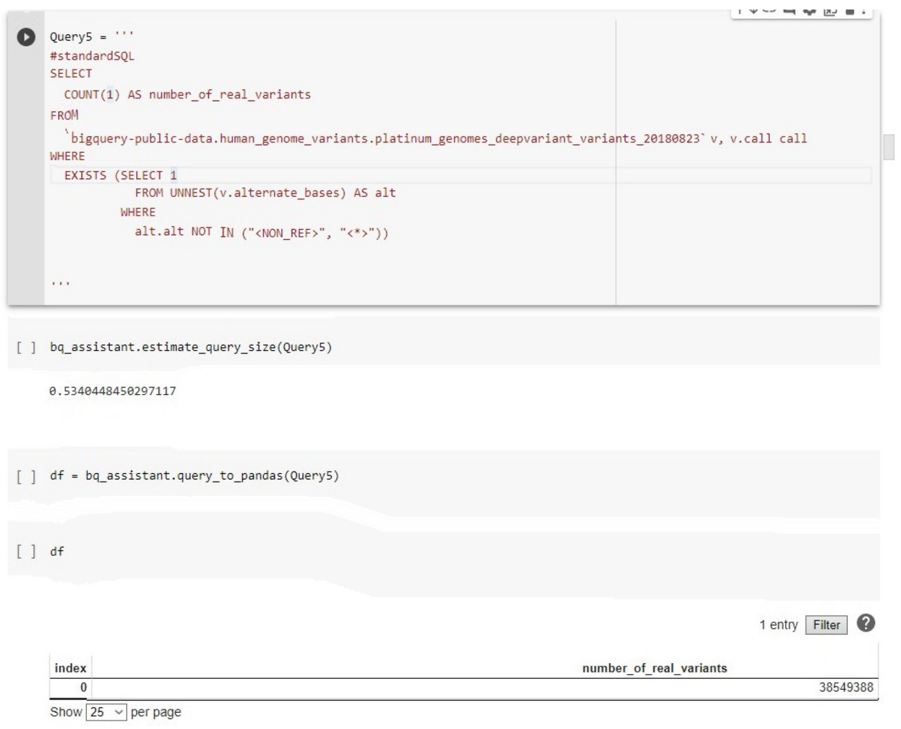
Show 25 per page

Figure 5 Tutorial1, Query 3

- **IT Interpretation:** This query get the total number of variant call across all samples. It does this by joining each row with the call column.The joining is done implicitly in this case with the use of ',' operator in the 'FROM' section of the query.
  `COUNT(call) AS number_of_calls`
- **Biological Interpretation:** This query gives us the total number of real_variant and non_variant in the sample just as in the query in figure 4.

- Query 4 (see Figure 6)
- **IT Interpretation:** This counts the number of real_variant segment in the table. It counts from the table column number_of_real_variants the '1' and outputs it as the real_variants in the sample.

```
COUNT(1) AS number_of_real_variants
        WHERE
        EXISTS (SELECT 1
```



```
Query5 = '''
#standardSQL
SELECT
  COUNT(1) AS number_of_real_variants
FROM
  `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823` v, v.call call
WHERE
  EXISTS (SELECT 1
            FROM UNNEST(v.alternate_bases) AS alt
          WHERE
            alt.alt NOT IN ("<NON_REF>", "<*>"))

'''
```

```
[ ] bq_assistant.estimate_query_size(Query5)

    0.5340448450297117
```

```
[ ] df = bq_assistant.query_to_pandas(Query5)
```

```
[ ] df
```

1 entry Filter

index	number_of_real_variants
0	38549388

Show 25 per page

Figure 6 Tutorial1, Query 4

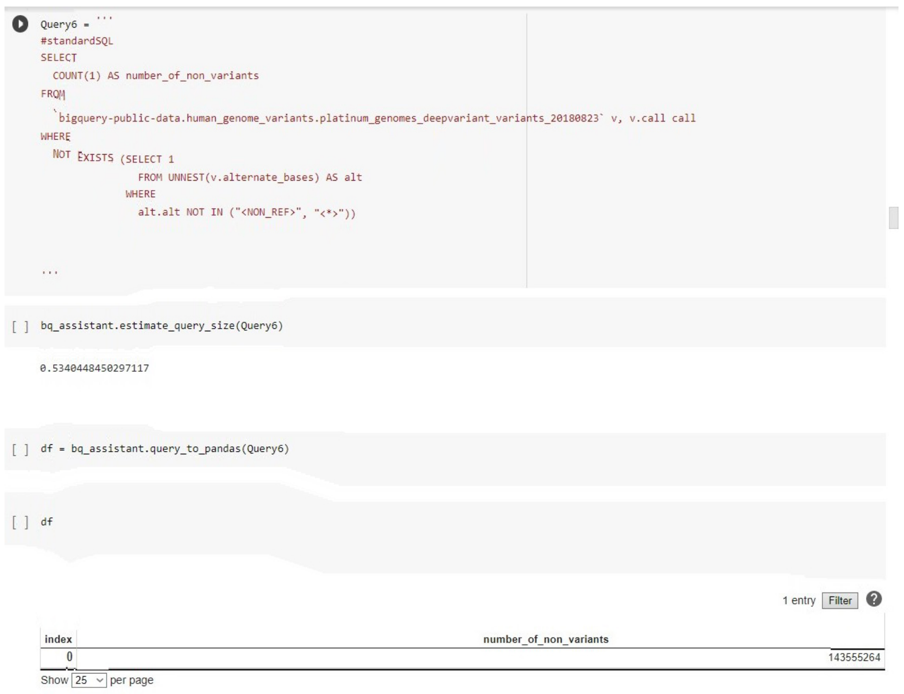
```
    FROM UNNEST(v.alternate_bases) AS alt
  WHERE
    alt.alt NOT IN ("<NON_REF>", "<*>"))
```

The 'WHERE' clause extracts from the 'v.alternate_base' column '1' if alt.alt is not in '¡NON_REF¿' or '¡*¿'. In essence, from the returned column count the number of '1' and output the total.

- **Biological Interpretation:** Get the number of distinct variant from each sample in the dataset.

- Query 5 (see Figure 7)
- **IT Interpretation:** This counts the number of non_real_variant segment in the table. It counts from the table column number_of_real_variants the '1' and outputs it as the real_variants in the sample.

```
COUNT(1) AS number_of_non_variants
        WHERE
        NOT EXISTS (SELECT 1
            FROM UNNEST(v.alternate_bases) AS alt
          WHERE
            alt.alt NOT IN ("<NON_REF>", "<*>"))
```



```
Query6 = '''
#standardSQL
SELECT
  COUNT(1) AS number_of_non_variants
FROM
  `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823` v, v.call call
WHERE
  NOT EXISTS (SELECT 1
                FROM UNNEST(v.alternate_bases) AS alt
              WHERE
                alt.alt NOT IN ("<NON_REF>", "<*>"))


'''
```

```
[ ] bq_assistant.estimate_query_size(Query6)
```

```
0.5340448450297117
```

```
[ ] df = bq_assistant.query_to_pandas(Query6)
```

```
[ ] df
```

1 entry Filter

index	number_of_non_variants
0	143555264

Show 25 per page

Figure 7 Tutorial1, Query 5

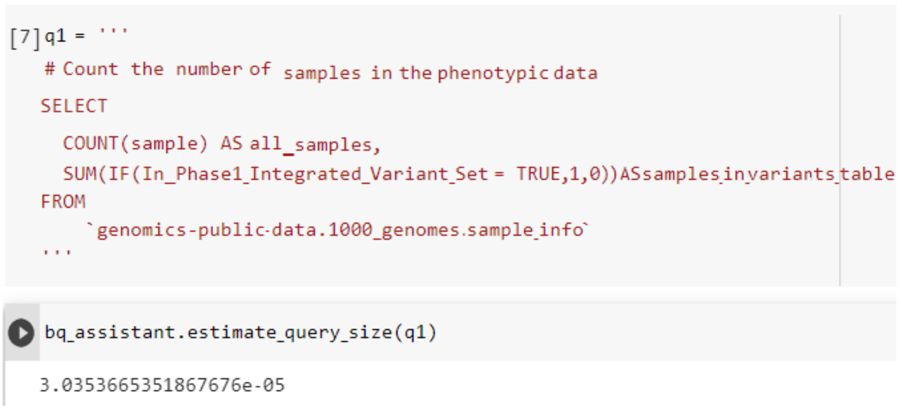
The 'WHERE' clause extracts from the 'v.alternate_base' column '0' if alt.alt is not in '¡NON_REF¿' or '¡*¿'. In essence, from the returned column count the number of '1' and output the total.

- **Biological Interpretation:** Get the number of non-variant from each sample in the dataset. The sum of the real_variants and non_variants equals the number_of_calls as seen above. Query 1,2 and 3 all finished in 0.0s. As the query instruction increased so does the time to query, the fourth and first queries (number_of_real_variants and number_of_non_variants) took 0.534s to finish the queries. Other queries which we did showed a steady increase in time as the filter parameters becomes more and more specific (details can be seen in the attached notebook).

4.2 Task 2

For the 2nd part of this project we decided to do the first part "Exploration of phenotypic data" and also 2 queries for the second part "Exploration of variant data" of the tutorial "Data stories for the 1000genome project". The first part was to collect data on ethnicity, gender and family relationships, and the second part was to collect data on genome variants available for the 1000 genome data set. The IT and biological interpretation of the queries used follows.

- Query 1 (see Figure 8)
  - **IT Interpretation:** This query counts the number of samples in the data table ‘genomics-public-data.1000_genomes.variants‘
    `COUNT(sample) AS all_samples`



```
[7] q1 = '''
 # Count the number of samples in the phenotypic data
 SELECT
   COUNT(sample) AS all_samples,
   SUM(IF(In_Phase1_Integrated_Variant_Set = TRUE,1,0))ASsamples_in_variants_table
 FROM
     `genomics-public-data.1000_genomes.sample_info`
 '''
```

```
bq_assistant.estimate_query_size(q1)
```

```
3.0353665351867676e-05
```

Figure 8 Tutorial 2, Query 1, size=3.03e-05 GB

Therefore, all lines in which there is at least one sample and the sum of the samples are calculated _in_variants_table.

```
SUM(IF(In_Phase1_Integrated_Variant_Set = TRUE, 1, 0))
        AS samples_in_variants_table
```

if the variable "In_Phase1_Integrated_Variant_Set" is TRUE, 1 or 0. With the help of SELECT it is possible to get the counts of data. As output we get with the help of BigQueryHelper class we are able to estimate size of query (.estimate_query_size() and convert query into pandas DataFrame table (.query_to_pandas()) containing calculated counts of all_samples and counts of samples_in_variants_table.

– **Biological Interpretation:** Biologically this means that the complete set of 1000 genomes contains data for 3500 individuals but the data in the table of low-coverage variants are valid only for a subset of these individuals (see Figure 9). For analyses across all samples in the table of variants, the sample size is therefore 1092.

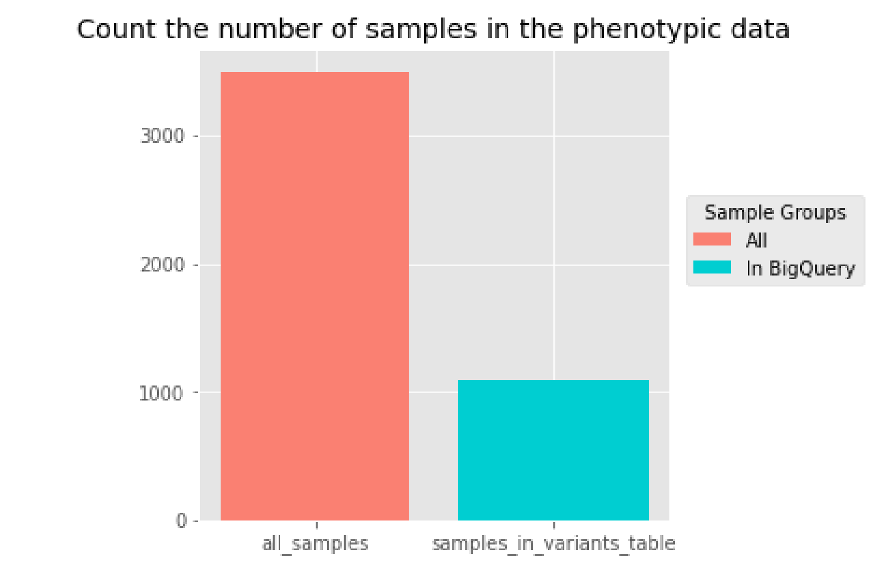
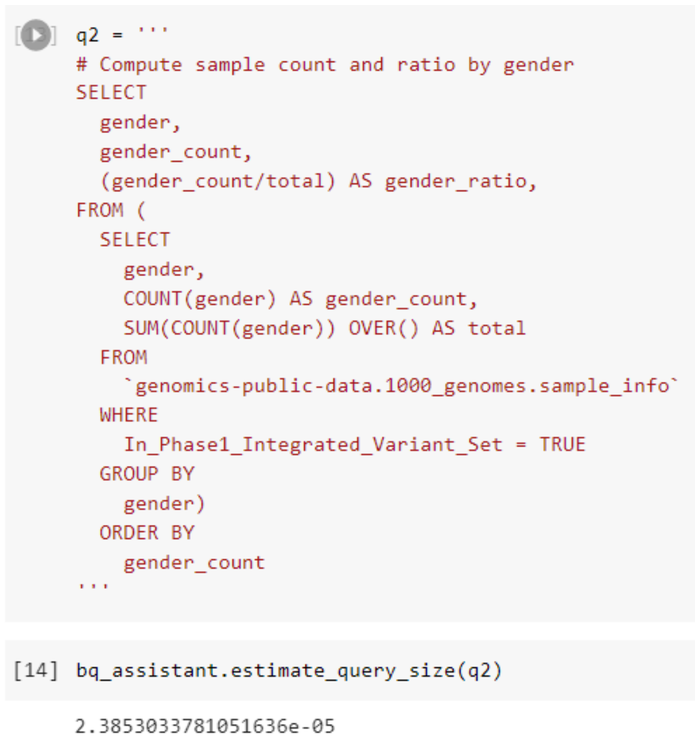


Figure 9 Plot to the Tutorial 2, Query 1

- Query 2 (see Figure 10)



```
[ ] q2 = '''
    # Compute sample count and ratio by gender
    SELECT
      gender,
      gender_count,
      (gender_count/total) AS gender_ratio,
    FROM (
      SELECT
        gender,
        COUNT(gender) AS gender_count,
        SUM(COUNT(gender)) OVER() AS total
      FROM
        `genomics-public-data.1000_genomes.sample_info`
      WHERE
        In_Phase1_Integrated_Variant_Set = TRUE
      GROUP BY
        gender)
      ORDER BY
        gender_count
    '''
```

```
[14] bq_assistant.estimate_query_size(q2)

     2.3853033781051636e-05
```

Figure 10 Tutorial 2, Query 2, Size = 2.38e-05 GB

- **IT Interpretation:** This query counts the number of each gender `COUNT(gender) AS gender_count` from the data table ‘genomics-public-data.1000_genomes.variants‘. It also calculates the gender_ratio

```
(gender_count/total) AS gender_ratio
```

Hereby total is calculated as sum of all gender counts calculated by

```
SUM(COUNT(gender)) OVER() AS total
```

As we want to only analyse individuals in the gene variant data, the WHERE clause is used to filter data with the attribute

```
In_Phase1_Integrated_Variant_Set = TRUE
```

which are also grouped by gender with GROUP BY. Finally, the data (gender, gender_count and gender_ratio calculated for each gender as gender_count divided by total) is ordered by gender_count and selected using the SELECT clause.

– **Biological Interpretation:** From a biological point of view we get gender data and conclude that the samples are almost equally divided between male (48.08%) and female (51.92%) . We are therefore able to study gene variants in both gender.

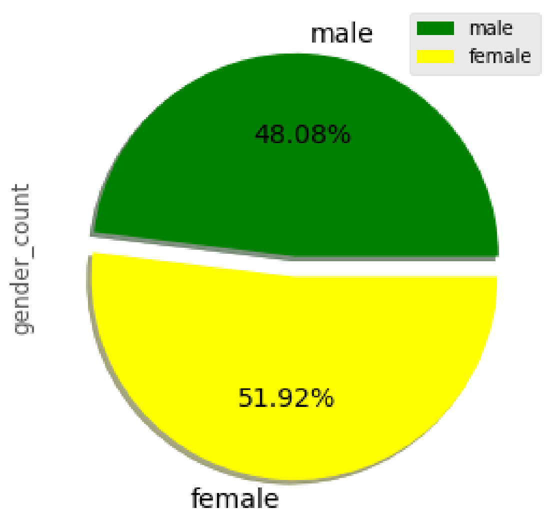
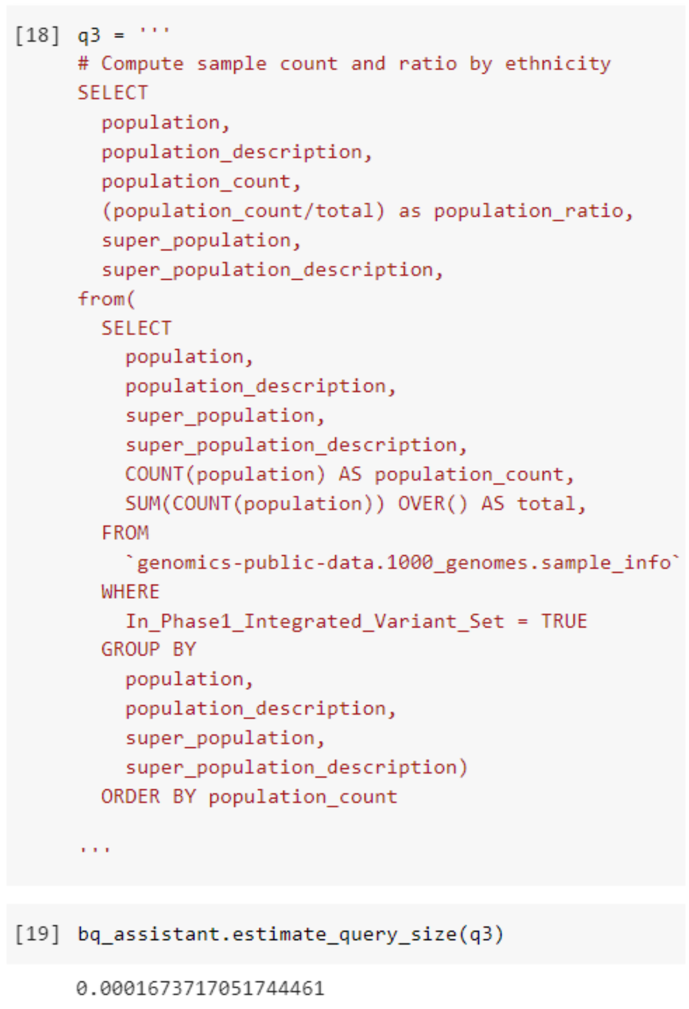


Figure 11 Plot to the Tutorial 2, Query 2

- Query 3 (see Figure 12)



```
[18] q3 = '''
     # Compute sample count and ratio by ethnicity
     SELECT
       population,
       population_description,
       population_count,
       (population_count/total) as population_ratio,
       super_population,
       super_population_description,
     from(
       SELECT
         population,
         population_description,
         super_population,
         super_population_description,
         COUNT(population) AS population_count,
         SUM(COUNT(population)) OVER() AS total,
       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)
       ORDER BY population_count

     '''
```

```
[19] bq_assistant.estimate_query_size(q3)

     0.00016737170517444461
```

Figure 12 Tutorial 2, Query 3, Size=1.6e-4

  - **IT Interpretation:** To analyse ethnicity of individuals, values such as population, population description, population_count, population_ratio, super_population and super_population_description are needed. These data are selected from the data
    `‘genomics-public-data.1000\_genomes.sample\_info‘`
    for which, the attribute is

    `In_Phase1_Integrated_Variant_Set = TRUE`

    . Similar as for the Query 2 the population number is calculated using COUNT AS function and for population_ratio the total number of all populations is calculated. These data are grouped by population, population_description,super_population and super_population_description. Finally, the needed data can be selected, whereby population_ratio is calculated by dividing each population_count by total and all data to be selected are ordered by population_count.

– **Biological Interpretation:** Analysing across all ethnic groups it can be concluded that the sample sizes are between 55 and 100, with an outlier of 14 by IBS (see Figure 13. So we cannot compare the gene variants between all the populations in the given regions.

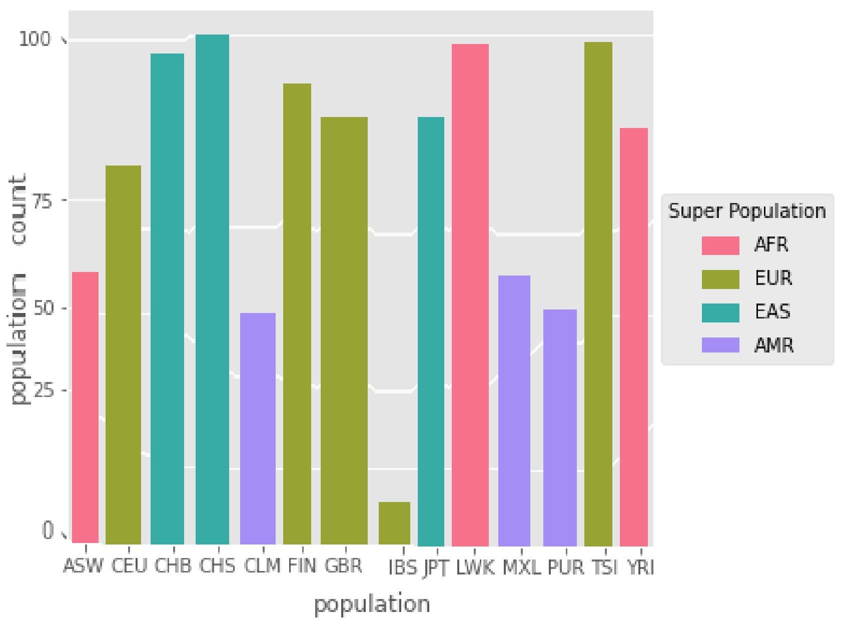
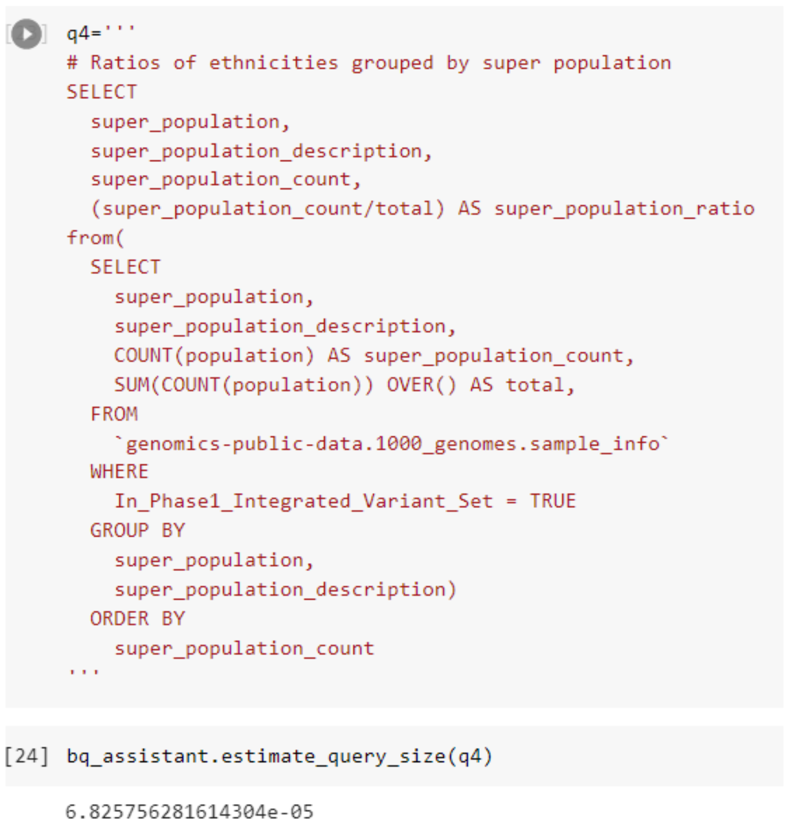


Figure 13 Plot to the Tutorial 2, Query 3

- Query 4 (see Figure 14)



```
q4='''
# Ratios of ethnicities grouped by super population
SELECT
  super_population,
  super_population_description,
  super_population_count,
  (super_population_count/total) AS super_population_ratio
from(
  SELECT
    super_population,
    super_population_description,
    COUNT(population) AS super_population_count,
    SUM(COUNT(population)) OVER() AS total,
  FROM
    `genomics-public-data.1000_genomes.sample_info`
  WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
    super_population,
    super_population_description)
  ORDER BY
    super_population_count
'''
```

```
[24] bq_assistant.estimate_query_size(q4)
```

```
6.825756281614304e-05
```

Figure 14 Tutorial 2, Query 4, Size=6.83e-05

  - **IT Interpretation:** To further analyse ethnicity of individuals the values such as super_population, super_population description, super_population_count, super_population_ratio are needed. These data are selected from the data
`‘genomics-public-data.1000\_genomes.sample\_info‘`
for which, the attribute is
`In_Phase1_Integrated_Variant_Set = TRUE`
. Similar as for the Query 2 and the population number is calculated using COUNT AS clause and for super_population_ratio the total number of all populations is calculated. These data are grouped by super_population and super_population_description. Finally, the needed data can be selected, whereby super_population_ratio is calculated by dividing each super_population_count by total and all data to be selected are ordered by super_population_count.
  - **Biological Interpretation:** As it can be seen in 15 the ratios are between 16.6% and 34.7% of the samples per super_population and are not even. So we can not compare gene_variants between different super_populations.

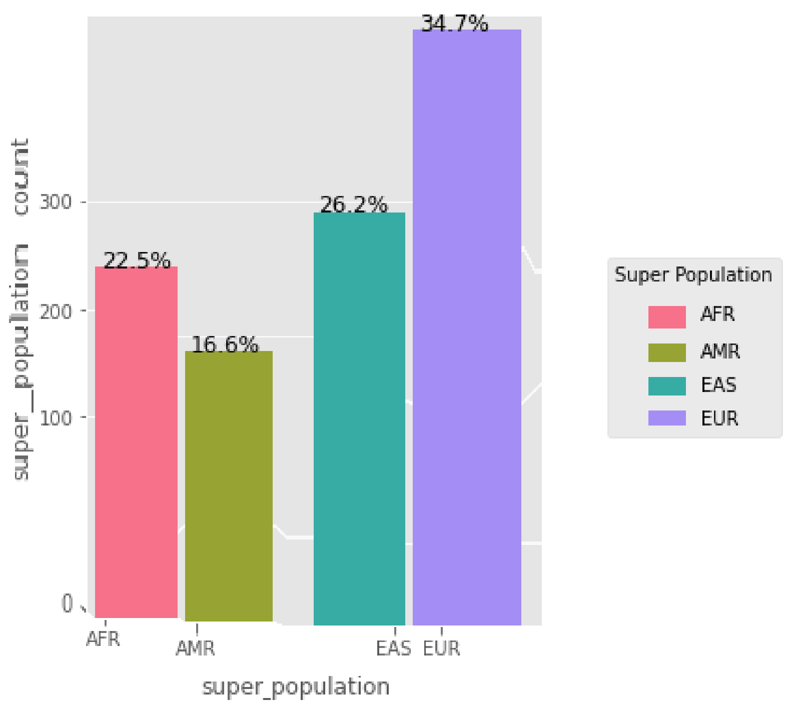
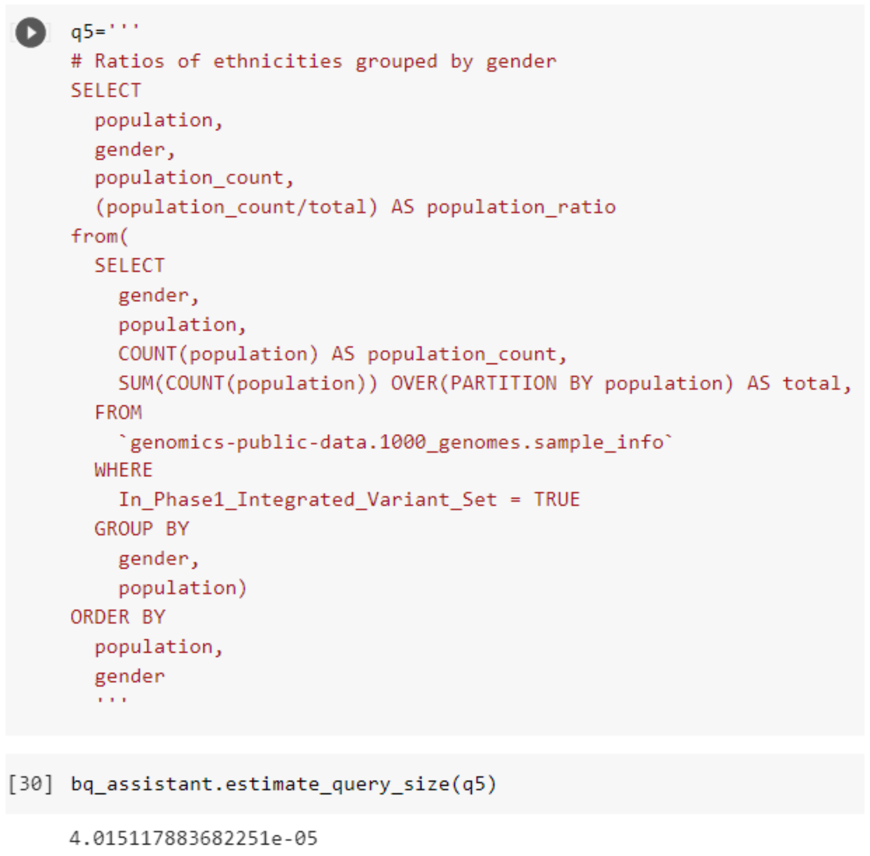


Figure 15 Plot to the Tutorial 2, Query 4

- Query 5 (see Figure 16)
  - **IT Interpretation:** With this query the ratios of ethnicity grouped by gender be calculated and with population, gender, population_count selected using the clause SELECT. Before it, these data are achieved from selected gender, population from the data table, calculated population_count and calculated sum of population counts for each population as a partition using these SQL functions
    `SUM(COUNT(population)) OVER(PARTITION BY population) AS total`
    and for which, the attribute is
    `In_Phase1_Integrated_Variant_Set = TRUE`
    and grouped by gender and population. Finally, the needed data can be selected, whereby population_ratio is calculated by dividing population_count by total for each population and gender and all data to be selected are ordered by population and gender.
  - **Biological Interpretation:** The genders are not equally distributed among different ethnic groups and also the sample sizes are small ranging from 7 to 50 see Figure 17.

x



```
q5='''
# Ratios of ethnicities grouped by gender
SELECT
  population,
  gender,
  population_count,
  (population_count/total) AS population_ratio
from(
  SELECT
    gender,
    population,
    COUNT(population) AS population_count,
    SUM(COUNT(population)) OVER(PARTITION BY population) AS total,
  FROM
    `genomics-public-data.1000_genomes.sample_info`
  WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
    gender,
    population)
ORDER BY
  population,
  gender
  '''
```

```
[30] bq_assistant.estimate_query_size(q5)

     4.015117883682251e-05
```

Figure 16 Tutorial 2, Query 5, Size=4.015e-05

- Query 6 (see Figure 18)
  - **IT Interpretation:** To investigate the relation between family_size and number_of_families, this query selects num_family_members and counts of each num_family_members which from primarily selected counts of each family_id and put in the num_family_members from the data table and for which, the attribute is

    `In_Phase1_Integrated_Variant_Set = TRUE`

    grouped by family_id (means for each family)

    `COUNT(family_id) AS num_family_members`

    Finally, the needed data are grouped by family_size and can be selected
  - **Biological Interpretation:** Figure 19 shows that about two thirds of the families consist of only one family member. It can be concluded that the most two thirds samples are from quite different genotypes which may have different unique gene variants.

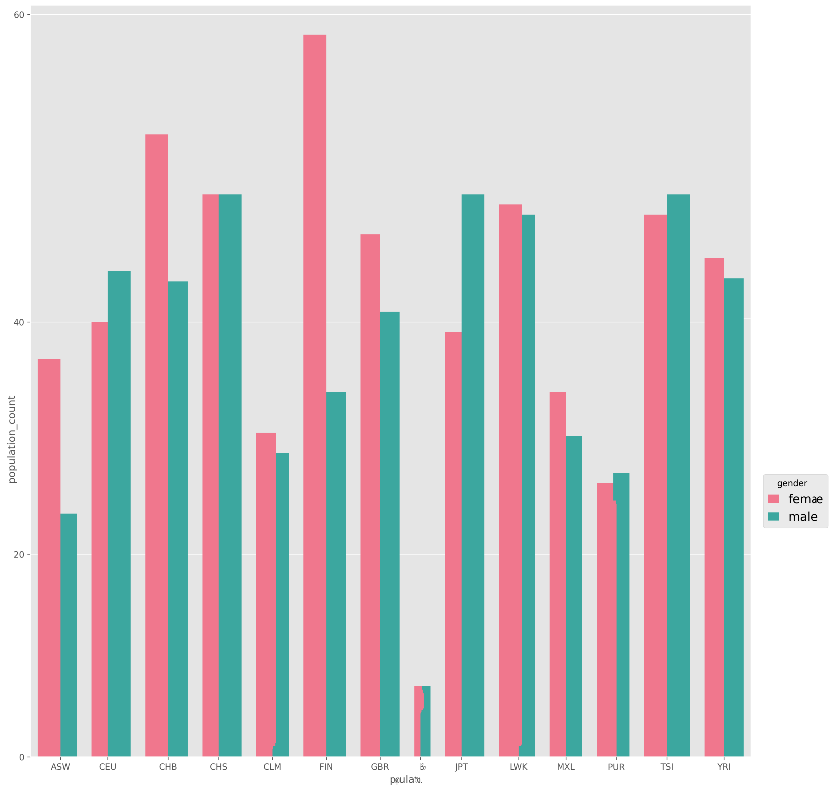
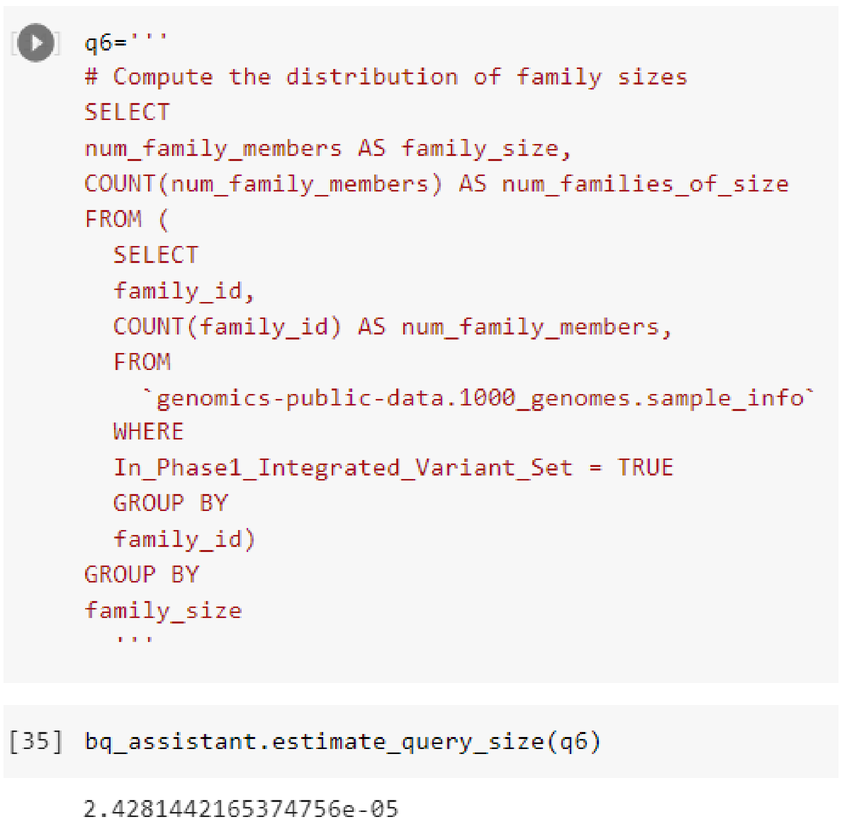


Figure 17 Plot to the Tutorial 2, Query 5

- Query 7 (see Figure 20)
  - **IT Interpretation:** To get an overview over min/max chromosomal positions of gene variants this query selects reference_name as integer using function SAFE_CAST and saves it in a variable chromosome. Also it minimum and maximum of the start with the help of MIN MAX functions. These data are selected from the data table

    `‘genomics-public-data.1000_genomes.variants‘`

    , for which reference_name is unequal ”X”, ”Y” and ”MT” and is grouped by chromosome
  - **Biological Interpretation:** Figure 21 shows that maximal chromosomal positions are nearly continuously and lie by nearly 10^8 slowly decreasing over the chromosomes from 1 to 22. The minimal chromosomal positions are not continuous over the chromosomes. At 17 it is the smallest, from chromosome 1 to 12 and 16 and from 18 to 20.They are by 10^5 and from 13 to 15 and 21,22 they are big and are by 10^7



```
q6='''
# 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
  '''
```

```
[35] bq_assistant.estimate_query_size(q6)

     2.4281442165374756e-05
```

Figure 18 Tutorial 2, Query 6, Size=2.43e-05

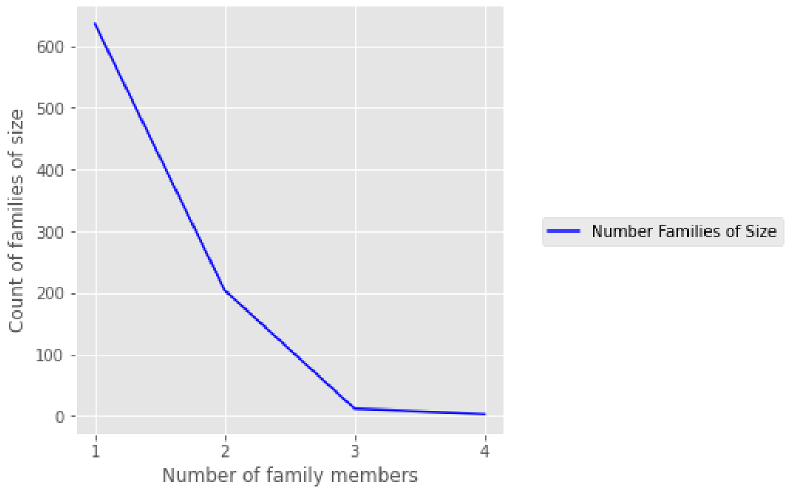
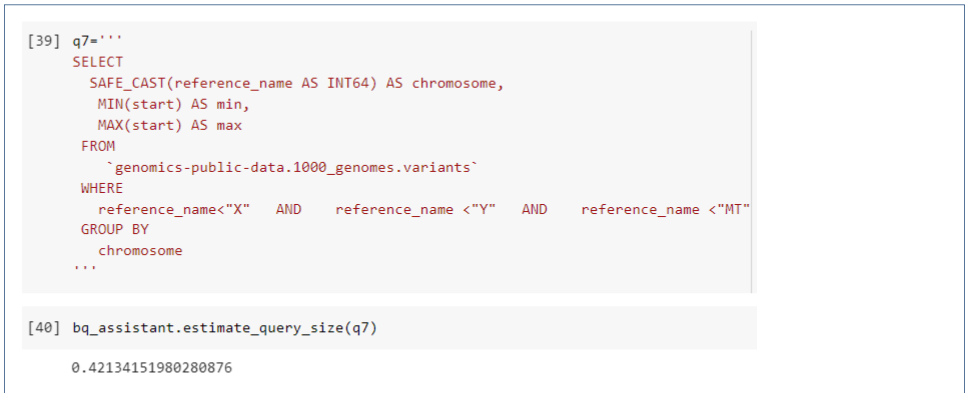


Figure 19 Plot to the Tutorial 2, Query 6

- Query 8 (see Figure 22)
  - **IT Interpretation:** This query selects reference_name as integer using function SAFE_CAST and saves it in a variable chromosome. It also



```
[39] q7='''
     SELECT
       SAFE_CAST(reference_name AS INT64) AS chromosome,
        MIN(start) AS min,
        MAX(start) AS max
      FROM
          `genomics-public-data.1000_genomes.variants`
      WHERE
        reference_name<"X"   AND    reference_name <"Y"   AND    reference_name <"MT"
      GROUP BY
        chromosome
     '''

[40] bq_assistant.estimate_query_size(q7)

     0.42134151980280876
```

Figure 20 Tutorial 2, Query 7, Size=0.42 GB

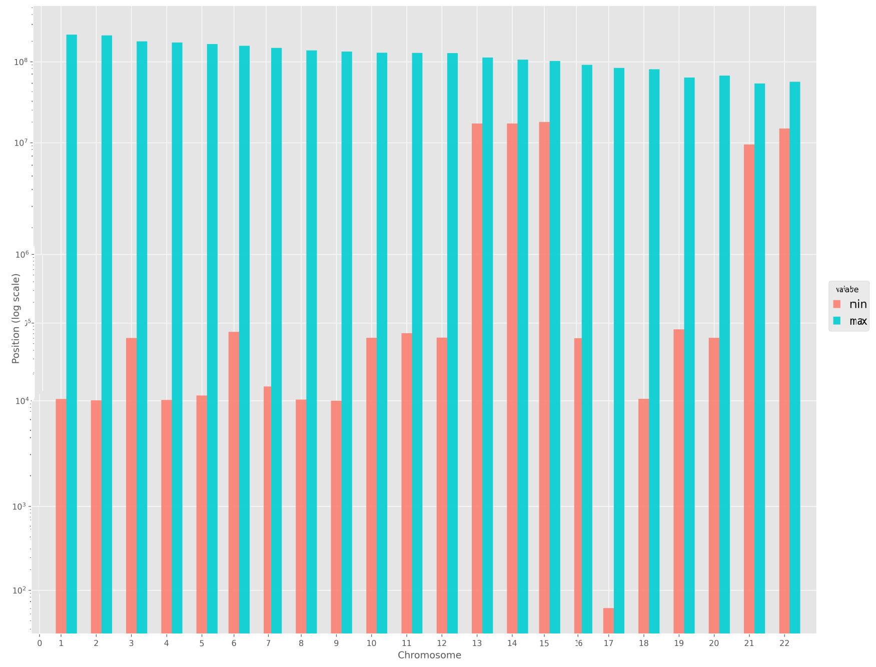


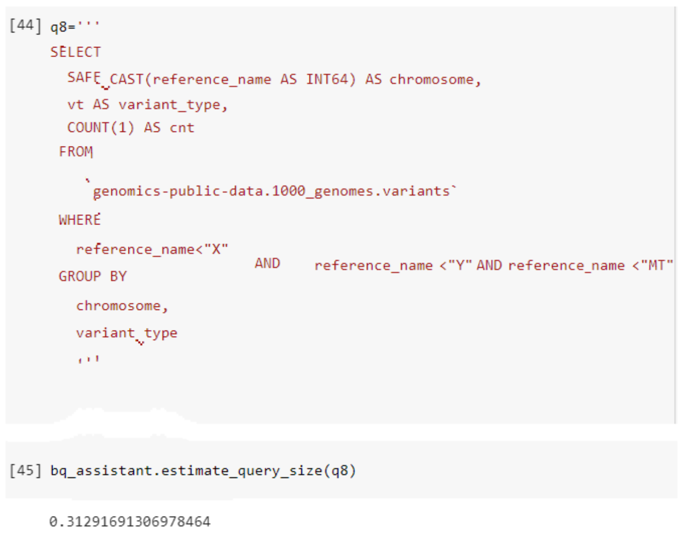
Figure 21 Plot to the Tutorial 2, Query 7

selects vt and saves it in variant_type and counts for how many times each variant_type appears by each chromosome. These data are selected from the data table

`‘genomics-public-data.1000_genomes.variants‘`

, for which reference_name is unequal ”X”, ”Y” and ”MT” and is grouped by chromosome and variant_type

– **Biological Interpretation:** The Figure 23 shows that SNP are proportionally are much more present. INDELS are rare and SV are much more few.



```
[44] q8='''
    SELECT
      SAFE_CAST(reference_name AS INT64) AS chromosome,
      vt AS variant_type,
      COUNT(1) AS cnt
     FROM
        `genomics-public-data.1000_genomes.variants`
     WHERE
       reference_name<"X"          AND    reference_name <"Y" AND reference_name <"MT"
     GROUP BY
       chromosome,
       variant_type
       '''
```

```
[45] bq_assistant.estimate_query_size(q8)
```

```
    0.31291691306978464
```

Figure 22 Tutorial 2, Query 8, Size=0.31 GB

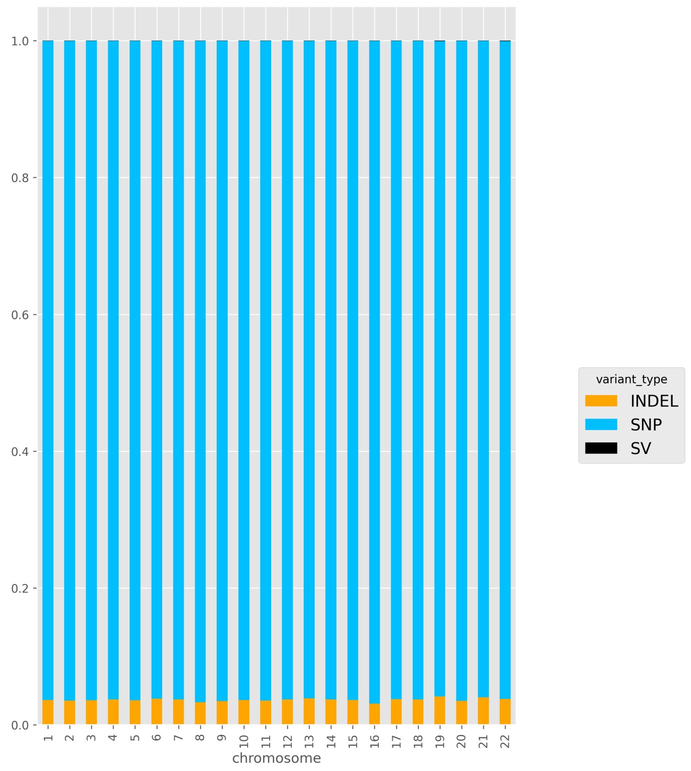


Figure 23 Plot to the Tutorial 2, Query 8

5 Discussion

5.1 Results

Using BigQuery it was possible for this project to get the data by formulating the queries with clauses and functions. The functions of BigQuery like .estimate_query_size() and convert query to pandas DataFrame table via .query_to_pandas() it was also possible to visualize.

5.2 Importance of the project for datascience

With BigQuery it is possible to analyze big datasets using Python. BigQuery is very useful for datascience as it makes possible to query Terabytes amounts of data in a short time.

6 Appendix

Name	Work Description
Michael	Method and Results of task 1, code for task 1 and some for task 2
Rohan	Scientific Background, Goal, Data and some part of code for task 2
Natalja	Methods and Results of task 2, Discussion, code for task 2, some part of the code for task 1

Table 1 Task responsibilities

References