This python code below can run at Google Colab platform
https://colab.research.google.com/drive/1mtlk_d8XLw49f-vQdUL3N7g-jhzgIR1t?usp=sharing

```
# Run  to install AlphaGenome.

from IPython.display import clear_output

! pip install alphagenome

clear_output()


# module import.

from alphagenome import colab_utils

from alphagenome.data import gene_annotation

from alphagenome.data import genome

from alphagenome.data import transcript as transcript_utils

from alphagenome.interpretation import ism

from alphagenome.models import dna_client

from alphagenome.models import variant_scorers

from alphagenome.visualization import plot_components

import matplotlib.pyplot as plt

import pandas as pd


# Take the API key.

dna_model = dna_client.create(colab_utils.get_api_key())

[output.name for output in dna_client.OutputType]


# read the information in genome assembly hg38 for human.

gtf = pd.read_feather(

   'https://storage.googleapis.com/alphagenome/reference/gencode/'
```

```
    'hg38/gencode.v46.annotation.gtf.gz.feather'
)

# Set up transcript extractors using the information in the GTF file.
# Mane select transcripts consists of of one curated transcript per locus.
gtf_transcripts = gene_annotation.filter_protein_coding(gtf)
gtf_transcripts = gene_annotation.filter_to_mane_select_transcript(gtf_transcripts)
transcript_extractor = transcript_utils.TranscriptExtractor(gtf_transcripts)

# set the variant of SLC26A4 c.997_998insA .
variant = genome.Variant(
    chromosome='chr7',
    position=107683532,
    reference_bases='A',  # from CCCAAGGGGG to CCAAAGGGGG
    alternate_bases='AA',
)
#Expand the reference interval around the variant to a 1-Mb window.
interval = variant.reference_interval.resize(dna_client.SEQUENCE_LENGTH_1MB)

# get RNA-seq predictions in the interval for "Thyroid gland" tissue
variant_output = dna_model.predict_variant(
    interval=interval,
    variant=variant,
    requested_outputs=[dna_client.OutputType.RNA_SEQ],
    ontology_terms=['UBERON:0002046'],
)  # thyroid gland

# plot
```

```
transcripts = transcript_extractor.extract(interval)


plot_components.plot(
  [
     plot_components.TranscriptAnnotation(transcripts),
     plot_components.OverlaidTracks(
        tdata={
           'REF': variant_output.reference.rna_seq,
           'ALT': variant_output.alternate.rna_seq,
        },
        colors={'REF': 'dimgrey', 'ALT': 'red'},
     ),
  ],
  interval=variant_output.reference.rna_seq.interval.resize(80000),
  # Annotate the location of the variant as a vertical line.
  annotations=[plot_components.VariantAnnotation([variant], alpha=0.8)],
)
plt.savefig(
  "image.svg",
  bbox_inches="tight",
  pad_inches=0.2,
  transparent=True
)# Save the plot as an SVG file  margin 0.2
plt.show()


# get RNA-seq predictions in the interval for “whole blood” tissue
variant_output = dna_model.predict_variant(
  interval=interval,
```

```
    variant=variant,
    requested_outputs=[dna_client.OutputType.RNA_SEQ],
    ontology_terms=['UBERON:0013756'],
)  # venous blood


# plot
transcripts = transcript_extractor.extract(interval)
plot_components.plot(
    [
        plot_components.TranscriptAnnotation(transcripts),
        plot_components.OverlaidTracks(
            tdata={
                'REF': variant_output.reference.rna_seq,
                'ALT': variant_output.alternate.rna_seq,
            },
            colors={'REF': 'dimgrey', 'ALT': 'red'},
        ),
    ],
    interval=variant_output.reference.rna_seq.interval.resize(80000),
    # Annotate the location of the variant as a vertical line.
    annotations=[plot_components.VariantAnnotation([variant], alpha=0.8)],
)
plt.savefig(
    "image.svg",
    format="svg",
    bbox_inches="tight",
    pad_inches=0.2,
```

```
    transparent=True
) # Save the plot as an SVG file
plt.show()
```