

# Notebook

June 5, 2025

## 1 Central dogma of molecular biology in our own cells

### 1.1 Primary Authors:

- Go, Daphne Janelyn
- Gilo, Joshua
- Lasala, Kyle Carlo

### 1.2 Notebook Version Control

```
[ ]: !pip install nbdime
```

```
[ ]: # Version Check
!nbdiff-web Bioinformatics_Ex1.ipynb https://raw.githubusercontent.com/
↳KyleCarlo/gene-annotation-analysis-of-human-genome/refs/heads/main/
↳Bioinformatics_Ex1.ipynb
```

~C

```
[7]: # Merging
!nbmerge-web Bioinformatics_Ex1.ipynb Bioinformatics_Ex1.ipynb https://raw.
↳githubusercontent.com/KyleCarlo/gene-annotation-analysis-of-human-genome/
↳refs/heads/main/Bioinformatics_Ex1.ipynb --out merged.ipynb
```

~C

### 1.3 Prerequisites: Downloading the necessary data and importing the libraries.

```
[ ]: # gffutils for reading gtf
!pip install gffutils
```

```
[1]: import gffutils
import pandas as pd
import matplotlib.pyplot as plt

import urllib.request
import os
import gzip
import shutil
```

## 1.4 Objective:

1. Explore the human genome, transcriptome, and proteome.
2. Investigate specific, well-studied human genes by analyzing their genomic location, expression patterns, isoform diversity, and biological functions.

## 1.5 Part 1. Human genome, transcriptome, proteome

There are multiple resources listed towards the bottom of this page, which provide information about the human genome. For example, you can download information about the human genome, transcriptome, and proteome from [RefSeq](#) or from [GENCODE](#).

Using these resources, report on the following questions:

1. **How many genes are there in the human genome?**
2. **How many transcripts are there in the human transcriptome?**
3. **What are the different kinds/classes of RNA present in the human transcriptome?**
4. **How many protein-coding genes are there in the human genome?**
5. **How many proteins are there in the human proteome?**
6. **Plot the distribution of lengths of genes.**
7. **Plot the distribution of the length of exons.**
8. **Plot the distribution of the length of introns.**

In line with findings from Frankish et al. (2015), we compared and analyzed human gene annotations from both RefSeq and GENCODE, recognizing that while GENCODE offers broader transcript diversity and genomic coverage, RefSeq provides a more conservative and streamlined annotation set. The downloaded gene annotations is version *Genome Reference Consortium Human Build 38 (GRCh38.p14)*

We specifically downloaded the [Comprehensive gene annotation \(CHR\)](#) from GENCODE (GTF), which includes the full set of gene annotations mapped to the reference chromosomes only. Additionally, we downloaded the gene annotation feature (GTF) source files from both RefSeq and GenBank for comparison. See the image below.

### Select file source

- ☒ All
- ☐ RefSeq only
- ☐ GenBank only

### Select file types

- ☐ Genome sequences (FASTA)
- ☒ Annotation features (GTF)
- ☐ Annotation features (GFF)

Given that we have two reference datasets, we want to explore the differences between the two. With this, we answered each question to both datasets and compared our results.

```
[4]: urls = {
    "gencode.v48.annotation.gtf.gz":
        "ftp://ftp.ebi.ac.uk/pub/databases/gencode/"
        "Gencode_human/release_48/gencode.v48.annotation.gtf.gz",

    "GCF_000001405.40_GRCh38.p14_genomic.gtf.gz":
        "https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/405/"
        "GCF_000001405.40_GRCh38.p14/GCF_000001405.40_GRCh38.p14_genomic.gtf.gz"
}

for filename, url in urls.items():
    if os.path.exists(filename):
        print(f"{filename} already exists, skipping download.")
    else:
        print(f"Downloading {filename} ...")
        urllib.request.urlretrieve(url, filename)
        print(f"Saved to {filename}")

for fn in urls:
    out_fn = fn[:-3]
    if os.path.exists(out_fn):
        print(f"{out_fn} already exists, skipping decompression.")
    else:
        print(f"Decompressing {fn} to {out_fn} ...")
        with gzip.open(fn, 'rb') as f_in, open(out_fn, 'wb') as f_out:
            shutil.copyfileobj(f_in, f_out)
        print(f"Decompressed to {out_fn}")
        os.remove(fn)
```

```
Downloading gencode.v48.annotation.gtf.gz ...
Saved to gencode.v48.annotation.gtf.gz
Downloading GCF_000001405.40_GRCh38.p14_genomic.gtf.gz ...
Saved to GCF_000001405.40_GRCh38.p14_genomic.gtf.gz
gencode.v48.annotation.gtf already exists, skipping decompression.
GCF_000001405.40_GRCh38.p14_genomic.gtf already exists, skipping decompression.
```

### 1.5.1 Creating a DB for GTF files

These creates a database for both datasets, you only run this once. Once you have the .db files, you can now use it to query data.

```
[ ]: gencode_file = 'gencode.v48.annotation.gtf'
db_file = 'gencode.db'
gencode_db = gffutils.create_db(gencode_file, dbfn=db_file,
    ↪disable_infer_genes=True, disable_infer_transcripts=True, force=True,
    ↪keep_order=True, merge_strategy='merge', sort_attribute_values=True)
```

```
[5]: refseq_file = 'GCF_000001405.40_GRCh38.p14_genomic.gtf'
db_file = 'refseq.db'
refseq_db = gffutils.create_db(refseq_file, dbfn=db_file,
    ↳disable_infer_genes=True, disable_infer_transcripts=True, force=True,
    ↳keep_order=True, merge_strategy='merge', sort_attribute_values=True)
```

### 1.5.2 Reading GTF DB

Run these cells to read the generated .db files.

```
[2]: db_file = 'gencode.db'
gencode = gffutils.FeatureDB(db_file)
```

```
[3]: db_file = 'refseq.db'
refseq = gffutils.FeatureDB(db_file)
```

### 1.5.3 How many genes are there in the human genome?

**Data Source: GENCODE** To count the number of genes in the dataset, we filtered the rows where the `featuretype` is "gene" and counted the distinct number of rows from the column `id`. According to the [GENCODE release statistics](#) for this gene annotation, there are **78,686 genes**, which matches the number we obtained from our analysis.

```
[4]: cursor = gencode.conn.execute("""
    SELECT COUNT(DISTINCT id) FROM features
    WHERE featuretype='gene';
    """)

cursor.fetchall()[0][0]
```

```
[4]: 78686
```

**Without Pseudogenes** According to [National Human Genome Research Institute](#), a pseudogene is a segment of DNA that structurally resembles a gene but is not capable of coding for a protein. Pseudogenes are most often derived from genes that have lost their protein-coding ability due to accumulated mutations that have occurred over the course of evolution. In this case, we wanted to count the number of genes excluding pseudogenes.

We filtered out pseudogenes by removing rows and adding a condition where the `gene_type` contains the substring "pseudogene" and counted the distinct number of rows from the column `id`. According to the [GENCODE release statistics](#), there are 14,695 pseudogenes, and subtracting this from the total gene count (78,686) yields 63,991 genes, which is already close to our analysis. We got **63,494 genes excluding pseudogenes**.

```
[108]: cursor = gencode.conn.execute("""
    SELECT COUNT(DISTINCT id) FROM features
    WHERE json_extract(attributes, '$.gene_type') NOT LIKE '%pseudogene%'
    AND featuretype='gene'
    """)
```

```
cursor.fetchall()[0][0]
```

[108]: 63494

**Data Source: RefSeq** To count the number of genes in the dataset, we just counted the number of distinct elements in the `gene`. According to the [annotation report from NCBI](#), there are 59,715 genes. This closely matches our results, where we identified **59,701 genes**.

```
[6]: cursor = refseq.conn.execute("""
      SELECT COUNT(DISTINCT json_extract(attributes, '$.gene')) FROM features;
      """)

      refseq_gene_count = cursor.fetchall()[0][0]
      print(refseq_gene_count)
```

59701

**Without pseudogenes** Similar to GENCODE, we also want to count the number of genes excluding the pseudogenes.

We added a condition to filter out pseudogenes by counting the number of rows where `pseudo` is tagged as `true` and then subtracted it to the original gene count. According to the [annotation report from NCBI](#), there are 42,257 genes. This closely matches our results, where we identified **42,622 genes excluding pseudogenes**.

```
[8]: cursor = refseq.conn.execute("""
      SELECT COUNT(DISTINCT json_extract(attributes, '$.gene')) FROM features
      WHERE json_extract(attributes, '$.pseudo')='["true"]';
      """)

      print(refseq_gene_count - cursor.fetchall()[0][0])
```

42622

#### 1.5.4 How many transcripts are there in the human transcriptome?

**Data Source: GENCODE** To count the number of transcripts, we filtered out the rows where the `featuretype` is "transcript" and counted the distinct number of rows from the column `id`. According to the [GENCODE release statistics](#) for this gene annotation, there are **385,669 transcripts**, which matches the number we obtained from our results.

```
[22]: cursor = gencode.conn.execute("""
      SELECT COUNT(DISTINCT id) FROM features
      WHERE featuretype='transcript';
      """)

      cursor.fetchall()[0][0]
```

[22]: 385669

**Data Source: RefSeq (NCBI)** Similar to GENCODE, we filtered out the rows where the `featuretype` is "transcript" and then counted the number of distinct ids. According to the [annotation report from NCBI](#), there are 185,558 transcripts. This is slightly deviated with our results, where we identified **201,189 transcripts**.

```
[13]: cursor = refseq.conn.execute("""
        SELECT COUNT(DISTINCT id) FROM features
        WHERE featuretype='transcript';
        """)

        cursor.fetchall()[0][0]
```

[13]: 201189

### 1.5.5 What are the different kinds/classes of RNA present in the human transcriptome?

**Data Source: GENCODE** There are 3 main types of RNA involved in protein synthesis: mRNA (messenger RNA), tRNA (transfer RNA), and rRNA (ribosomal RNA). However, we have to consider other types of RNA. In the GENCODE dataset, there are:

Mt\_rRNA, Mt\_tRNA, miRNA, misc\_RNA, scRNA, snRNA, snoRNA, sRNA, and scaRNA are non-coding RNA predicted using sequences from Rfam and miRBase

lncRNA is a generic long non-coding RNA biotype

Non-coding RNAs are RNA molecules that are not translated into proteins, but they play critical roles in cellular function and regulation. Even though they don't participate directly in protein synthesis like mRNAs, their biological significance is profound.

Now, to get all the types of RNA, we filtered out the rows that only includes the substring "RNA" from the `gene_type` feature and displayed the distinct elements from the same feature. According to the [GENCODE release statistics](#), the RNA types are **lncRNA**, **miRNA**, **misc\_RNA**, **Mt\_rRNA**, **Mt\_tRNA**, **rRNA**, **rRNA\_pseudogene**, **scaRNA**, **snoRNA**, **snRNA**, **sRNA**, **vault\_RNA**, which matches our results.

```
[30]: cursor = gencode.conn.execute("""
        SELECT DISTINCT json_extract(attributes, '$.gene_type') FROM features
        WHERE json_extract(attributes, '$.gene_type') LIKE '%RNA%'
        """)

        for each in cursor.fetchall():
            for each_2 in each:
                print(each_2)
```

["lncRNA"]

["miRNA"]

["snRNA"]

```

["misc_RNA"]
["snoRNA"]
["scaRNA"]
["rRNA_pseudogene"]
["rRNA"]
["sRNA"]
["vault_RNA"]
["Mt_tRNA"]
["Mt_rRNA"]

```

**Data Source: RefSeq (NCBI)** There are RNA types that exists on both RefSeq and GENCODE datasets such as lncRNA, miRNA, snRNA, misc\_RNA, snoRNA, and rRNA. These RNA have the same meaning. For RNAs not included in GENCODE, tRNA, antisense\_RNA, vault\_RNA, telomerase\_RNA, RNase\_MRP\_RNA, RNase\_P\_RNA, Y\_RNA, and scRNA are non-coding RNAs.

To get the types of RNA, we filtered out the rows that only includes the substring "RNA" from the `transcript_biotype` feature and displayed the distinct elements from the same feature. According to the [annotation report from NCBI](#), the RNA types are **mRNA**, **misc\_RNA**, **miRNA**, **tRNA**, **lncRNA**, **snoRNA**, **snRNA**, **antisense\_RNA**, **rRNA**, **vault\_RNA**, **telomerase\_RNA**, **RNase\_MRP\_RNA**, **RNase\_P\_RNA**, **Y\_RNA**, **scRNA**, which matches the result of our query.

```

[33]: cursor = refseq.conn.execute("""
        SELECT DISTINCT json_extract(attributes, '$.transcript_biotype') FROM
        ↪features
        WHERE json_extract(attributes, '$.transcript_biotype') LIKE '%RNA%'
        """)

for each in cursor.fetchall():
    for each_2 in each:
        print(each_2)

```

```

["miRNA"]
["lnc_RNA"]
["mRNA"]
["snRNA"]
["snoRNA"]
["ncRNA"]
["antisense_RNA"]
["tRNA"]
["rRNA"]
["telomerase_RNA"]
["vault_RNA"]
["Y_RNA"]
["RNase_MRP_RNA"]
["scRNA"]
["RNase_P_RNA"]

```

### 1.5.6 How many protein-coding genes are there in the human genome?

**Data Source: GENCODE** To get the number of protein-coding genes, we filtered out the `gene_type` into "protein\_coding" only and filtered out the `tag` does not contain the substring "read". Then we counted the number of distinct elements in the `gene_id` column. According to the [GENCODE release statistics](#), there are **19,435** protein-coding genes, which exactly matches our result.

```
[42]: cursor = gencode.conn.execute("""
        SELECT COUNT(DISTINCT json_extract(attributes, '$.gene_id')) FROM features
        WHERE json_extract(attributes, '$.gene_type')='["protein_coding"]'
        AND json_extract(attributes, '$.tag') NOT LIKE '%read%';
    """)

    cursor.fetchall()[0][0]
```

[42]: 19452

**Data Source: RefSeq (NCBI)** To get the number of protein-coding genes, we filtered out the `gene_biotype` into "protein-coding" only and counted the number of distinct elements in the `gene` column. According to the [annotation report from NCBI](#), there are **20,078 protein-coding genes**, which matches exactly our result.

```
[47]: cursor = refseq.conn.execute("""
        SELECT COUNT(DISTINCT json_extract(attributes, '$.gene')) FROM features
        WHERE json_extract(attributes, '$.gene_biotype')='["protein_coding"]'
    """)

    cursor.fetchall()[0][0]
```

[47]: 20078

### 1.5.7 How many proteins are there in the human proteome?

**Data Source: GENCODE** To get the number of proteins, we counted all the distinct elements from the `protein_id` column. The GENCODE statistics did not include this from the report, but our query results into **112,346 proteins**.

```
[50]: cursor = gencode.conn.execute("""
        SELECT COUNT(DISTINCT json_extract(attributes, '$.protein_id')) FROM_
        ↪features;
    """)

    cursor.fetchall()[0][0]
```

[50]: 112346

**Data Source: RefSeq (NCBI)** Similar to GENCODE, we counted all distinct elements from the `protein_id` column. The annotation report of NCBI did not include this from the report, but



our query results into **136,282 proteins**.

```
[51]: cursor = refseq.conn.execute("""
        SELECT COUNT(DISTINCT json_extract(attributes, '$.protein_id')) FROM_
        ↪features;
        """)

        cursor.fetchall()[0][0]
```

```
[51]: 136282
```

### 1.5.8 Plot the distribution of lengths of genes.

**Data Source: GENCODE** In this part, we constructed a SQL query to extract relevant information from the GENCODE SQLite database. Specifically, we selected the `featuretype`, a cleaned version of the `transcript_id`, and the `start` and `end` genomic coordinates from the `features` table. We then loaded the result of the query into a pandas DataFrame named `gencode_df` using `pd.read_sql`. We renamed the columns to `type`, `transcript_id`, `start`, and `end` for clarity. Finally, we added a new column called `length`, which calculates the length of each transcript (or feature) by subtracting the `start` position from the `end` position. This allowed us to compute the genomic feature lengths for further analysis, such as plotting the distribution of transcript lengths.

```
[5]: query = """
        SELECT featuretype, SUBSTRING(json_extract(attributes, '$.transcript_id'),_
        ↪3, LENGTH(json_extract(attributes, '$.transcript_id'))-4),
        start, end FROM features;
        """

        gencode_df = pd.read_sql(query, gencode.conn)
        gencode_df.columns = ['type', 'transcript_id', 'start', 'end']
        gencode_df['length'] = gencode_df['end'] - gencode_df['start']
```

```
[6]: gencode_df[gencode_df["type"]=="gene"][['start', 'end', 'length']].describe()
```

```
[6]:
```

	start	end	length
count	7.868600e+04	7.868600e+04	7.868600e+04
mean	7.474854e+07	7.477998e+07	3.143851e+04
std	5.588770e+07	5.589030e+07	8.614636e+04
min	5.770000e+02	6.470000e+02	7.000000e+00
25%	3.038022e+07	3.040348e+07	7.610000e+02
50%	6.218170e+07	6.220718e+07	4.596000e+03
75%	1.104158e+08	1.104551e+08	2.596550e+04
max	2.489366e+08	2.489371e+08	2.473538e+06

```
[7]: gene_lengths = gencode_df[gencode_df['type'] == 'gene']['length']

        bin_edges = (
            list(range(0, 1000, 100)) +
            list(range(1000, 10000, 1000)) +
```

```

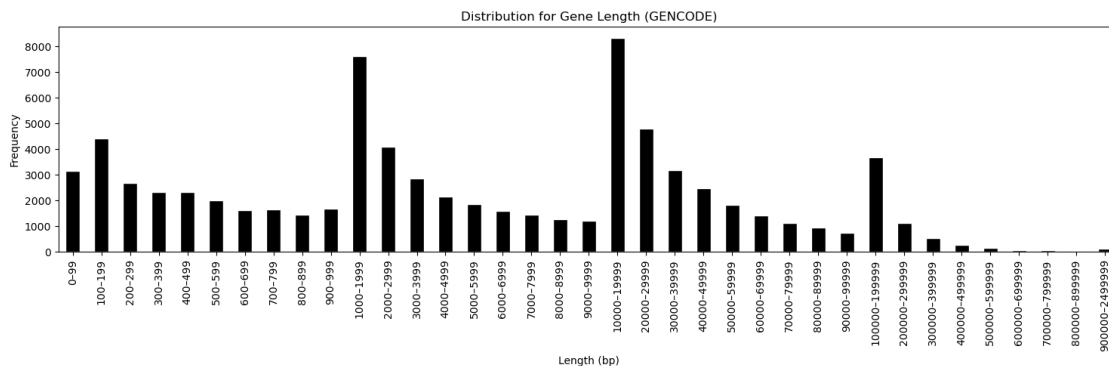
list(range(10000, 100000, 10000)) +
list(range(100000, 1000000, 100000)) +
[2500000]
)
bin_labels = [f"{bin_edges[i]}-{bin_edges[i+1]-1}" for i in_
↪range(len(bin_edges)-1)]

binned = pd.cut(gene_lengths, bins=bin_edges, labels=bin_labels, right=True)

bin_counts = binned.value_counts(sort=False)

plt.figure(figsize=(15, 5))
bin_counts.plot(kind='bar', color='black', edgecolor='white')
plt.xlabel("Length (bp)")
plt.ylabel("Frequency")
plt.title("Distribution for Gene Length (GENCODE)")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

```



```
[8]: min(gene_lengths), max(gene_lengths)
```

```
[8]: (7, 2473538)
```

As per the analysis, the shortest gene length is 7 bp while the longest is 2,473,538 bp. We also observed that **some genes are relatively short**, with a large number falling within the 0–1999 base pair (bp) range. There are **distinct peaks in the 100–199 bp and 1000–1999 bp bins**, suggesting that short genes are highly prevalent. Interestingly, there is also a **strong spike in the 10,000–19,999 bp range**, indicating a substantial number of moderately long genes. While the frequency of genes generally decreases with increasing length, the **distribution exhibits a heavy right tail extending up to 2.5 million bp**, representing a small subset of very long genes. This wide variation in gene length likely reflects the complexity of gene structure, where some genes contain numerous introns or regulatory elements, while others are more compact. Overall, the distribution appears somewhat bimodal, with one mode in the short gene region and another

around 10–20 kb, followed by a gradual decline, illustrating the diverse architecture of human genes.

**Data Source: RefSeq (NCBI)** The logic is almost the same with the GENCODE database. We start by defining a SQL query to select specific information from the `features` table: the feature type, a cleaned-up version of the `transcript_id` extracted from a JSON field called `attributes`, and the start and end coordinates of each feature. Then, we execute this query using pandas' `read_sql` function to load the data into a DataFrame named `refseq_df`. After loading, we rename the columns to more intuitive names — `type`, `transcript_id`, `start`, and `end`. Finally, we add a new column `length` by calculating the difference between the `end` and `start` positions, giving us the length of each feature.

```
[9]: query = """
      SELECT featuretype, SUBSTRING(json_extract(attributes, '$.transcript_id'),
      ↪3, LENGTH(json_extract(attributes, '$.transcript_id'))-4), start, end FROM
      ↪features;
      """
      refseq_df = pd.read_sql(query, refseq.conn)
      refseq_df.columns = ['type', 'transcript_id', 'start', 'end']
      refseq_df['length'] = refseq_df['end'] - refseq_df['start']
```

```
[10]: refseq_df[refseq_df['type'] == 'gene'][['start', 'end', 'length']].describe()
```

```
[10]:
```

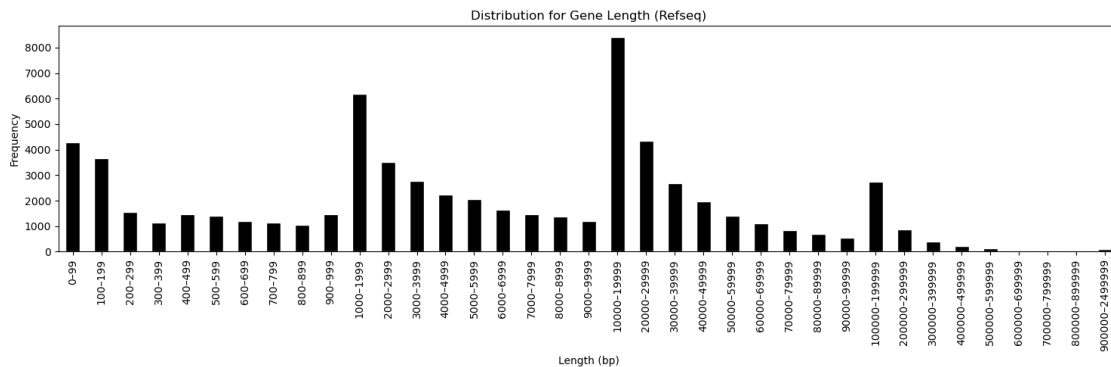
	start	end	length
count	6.742900e+04	6.742900e+04	6.742900e+04
mean	6.600083e+07	6.602978e+07	2.894632e+04
std	5.792598e+07	5.793032e+07	8.293749e+04
min	1.000000e+00	1.070000e+02	7.000000e+00
25%	1.834998e+07	1.837848e+07	8.990000e+02
50%	5.159851e+07	5.162277e+07	5.292000e+03
75%	1.029000e+08	1.029508e+08	2.268400e+04
max	2.489366e+08	2.489371e+08	2.473619e+06

```
[11]: gene_lengths = refseq_df[refseq_df['type'] == 'gene']['length']
      bin_edges = (
          list(range(0, 1000, 100)) +
          list(range(1000, 10000, 1000)) +
          list(range(10000, 100000, 10000)) +
          list(range(100000, 1000000, 100000)) +
          [2500000]
      )
      bin_labels = [f"{bin_edges[i]}-{bin_edges[i+1]-1}" for i in
      ↪range(len(bin_edges)-1)]

      binned = pd.cut(gene_lengths, bins=bin_edges, labels=bin_labels, right=True)

      bin_counts = binned.value_counts(sort=False)
```

```
plt.figure(figsize=(15, 5))
bin_counts.plot(kind='bar', color='black', edgecolor='white')
plt.xlabel("Length (bp)")
plt.ylabel("Frequency")
plt.title("Distribution for Gene Length (Refseq)")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



```
[12]: min(gene_lengths), max(gene_lengths)
```

```
[12]: (7, 2473619)
```

As per the analysis, the shortest gene length is 7 bp while the longest is 2,473,619 bp. There are **some genes that are relatively short**, with a high frequency of genes falling within the initial bins. Specifically, there is a large number of genes in the 0-99 bp range, indicating a significant presence of very short genes. Following this, there are distinct peaks in the 1000–1999 bp bin, suggesting a **strong prevalence of short to medium-length genes**. Interestingly, there is also a strong spike, representing the most dominant peak, in the 10,000–19,999 bp range, indicating a substantial number of **moderately long genes**. While the frequency of genes generally decreases with increasing length beyond this point, the distribution exhibits a **heavy right tail extending to much longer lengths**, representing a smaller subset of very long genes, including a noticeable peak around 100,000–199,999 bp. This wide variation in gene length likely reflects the diverse functional and structural requirements of genes. Overall, the distribution appears to be multimodal, with prominent modes in the very short (0-99 bp), short-to-medium (1000-1999 bp), and moderately long (10,000-19,999 bp) gene regions, followed by a gradual decline, illustrating the varied architecture of genes in the RefSeq database.

**Comparison between GENCODE and Refseq gene length distribution** The distributions of gene length from GENCODE and RefSeq are **similar to each other**. Both graphs are multimodal and show frequency peaks at 1,000–1,999 bp, 10,000–19,999 bp, and 100,000–199,999 bp. Overall, the distributions are right-skewed, exhibiting a heavy tail.

### 1.5.9 Plot the distribution of the length of exons.

We used the previously generated dataframes, `gencode_df` for GENCODE and `refseq_df` for Refseq and selected all rows where all `type` is "exon". We are only interested in the column `length` that shows the length of a particular exon.

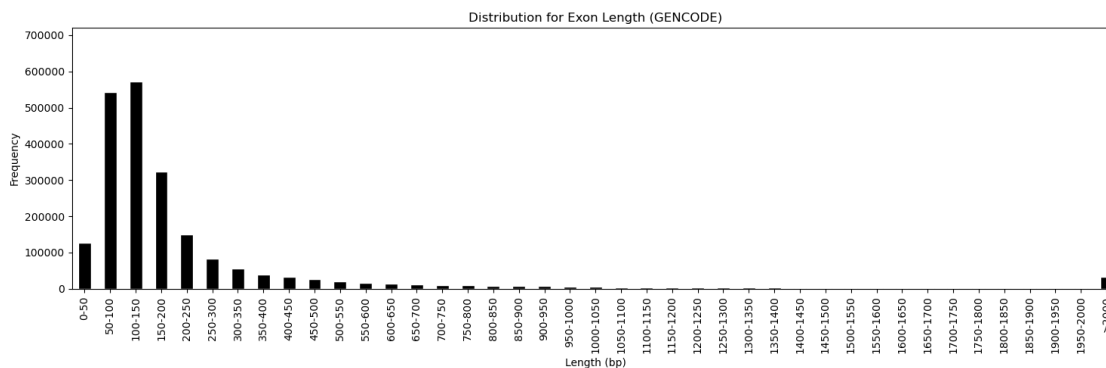
#### Data Source: GENCODE

```
[13]: exon_lengths = gencode_df[gencode_df['type'] == 'exon']['length']

bin_edges = list(range(0, 2001, 50)) + [10000]
bin_labels = [f"{i}-{i+50}" for i in range(0, 2000, 50)] + ['>2000']
binned = pd.cut(exon_lengths, bins=bin_edges, labels=bin_labels, right=True)

bin_counts = binned.value_counts(sort=False)

plt.figure(figsize=(15, 5))
bin_counts.plot(kind='bar', color='black', edgecolor='white')
plt.xlabel("Length (bp)")
plt.ylabel("Frequency")
plt.title("Distribution for Exon Length (GENCODE)")
plt.xticks(rotation=90)
plt.tight_layout()
plt.ylim(0, 720_000)
plt.show()
```



```
[14]: min(exon_lengths), max(exon_lengths)
```

```
[14]: (0, 347299)
```

The exon length distribution based on GENCODE, is heavily skewed to the right with a heavy tail end. It has a minimum length of 0 bp and maximum length of 347,299 bp. The vast **majority of exons are short**, with the highest frequencies observed in the 50-100 bp bin (approaching 600,000 exons), followed closely by the 0-50 bp and 100-150 bp categories. Frequencies decline steeply as exon length increases, although a **notable number of exons persist in much longer length categories**, extending to over 2000 bp. This indicates that while short exons are overwhelmingly

more common, longer exons do exist, albeit in significantly smaller quantities.

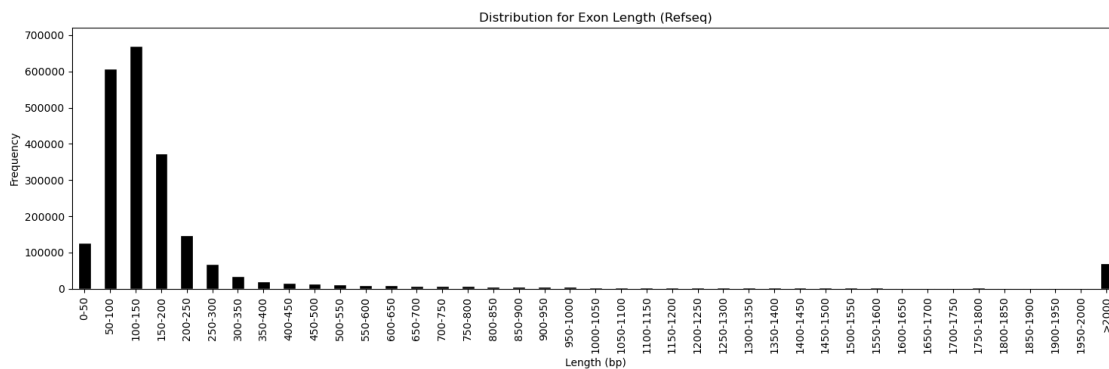
#### Data Source: RefSeq (NCBI)

```
[15]: exon_lengths = refseq_df[refseq_df['type'] == 'exon']['length']

bin_edges = list(range(0, 2001, 50)) + [10000]
bin_labels = [f"{i}-{i+50}" for i in range(0, 2000, 50)] + ['>2000']
binned = pd.cut(exon_lengths, bins=bin_edges, labels=bin_labels, right=True)

bin_counts = binned.value_counts(sort=False)

plt.figure(figsize=(15, 5))
bin_counts.plot(kind='bar', color='black', edgecolor='white')
plt.xlabel("Length (bp)")
plt.ylabel("Frequency")
plt.title("Distribution for Exon Length (Refseq)")
plt.xticks(rotation=90)
plt.tight_layout()
plt.ylim(0, 720_000)
plt.show()
```



```
[16]: min(exon_lengths), max(exon_lengths)
```

```
[16]: (0, 91666)
```

The exon length distribution based on Refseq is heavily skewed to the right, signifying that the vast majority of exons are relatively short. The highest frequencies are concentrated in the shortest length categories: the 50-100 bp bin shows the highest frequency, approaching 700,000 exons, followed by the 0-50 bp and 100-150 bp bins, both well over 300,000. As exon length increases beyond these initial bins, the frequency drops off rapidly, forming a long and attenuated tail. Despite the sharp decrease in frequency for longer lengths, a notable number of exons are still observed in the very long categories, including the >2000 bp bin, which contains approximately 70,000 exons. This pattern clearly demonstrates that while short exons are overwhelmingly more common, longer exons do exist, though in significantly smaller quantities.

**Comparison between GENCODE and Refseq exon length distribution** The distributions of exon length from GENCODE and RefSeq are **similar to each other**. A subtle difference can be seen in the peaks. GENCODE's highest peak is approximately 600,000 while Refseq's peak is approximately 700,000. Both follow a unimodal pattern and have the same peak at 50-150 bp. Using the same binning, both distributions closely follow the same distribution reported by Sakharkar et. al (2005). However, based on the distribution of Sakharkar et. al, the peaks are approximately 75,000 which is significantly lower compared to GENCODE and Refseq.

This discrepancy in frequency might be caused by different annotation versions. Sakharkar et al. used genome annotations that were available in the early 2000s, when both GENCODE and RefSeq had fewer annotated genes and transcripts. Our uses modern annotations, which are far more comprehensive, with many more transcripts and alternative splicing events included. This results in a higher number of exons, higher counts per bin, even if the shape remains the same.

Sakharkar, M., Perumal, B., Sakharkar, K., & Kanguane, P. (2005). An analysis on gene architecture in human and mouse genomes. In *Silico Biology*, 5, 347–365. <https://doi.org/10.3233/ISB-00192>

#### 1.5.10 Plot the distribution of the length of introns.

To get the intron lengths, we have to filter first our data where `type` is "exon" to get all the exons with coordinates `start` and `end`. Next, we grouped them according to `transcript_id` and sorted according to `transcript_id` and `start`. Since the exons are grouped according to its corresponding transcript and already sorted, we use the exon coordinates `start` and `end` to infer the intron length. We subtract each element in the `start` at index  $i+1$  to the element in the `end` at index  $i$ .

$$intron[i] = start[i + 1] - end[i]$$

#### Data Source: GENCODE

```
[17]: gencode_introns_df = gencode_df[gencode_df['type'] == 'exon'].reset_index().
      ↪drop('index', axis=1)[['transcript_id', 'start', 'end']]
gencode_introns_df = gencode_introns_df.sort_values(['transcript_id', 'start']).
      ↪reset_index().drop('index', axis=1)
gencode_introns_df = gencode_introns_df.groupby('transcript_id').agg(list)
```

```
[18]: introns = []
i = 0
for each in gencode_introns_df.itertuples():
    _, start, end = each
    start = pd.Series(start + [0])
    end = pd.Series([0] + end)
    introns += list(start-end)[1:-1]
```

```
[19]: intron_lengths = pd.Series(introns)

bin_edges = list(range(0, 20001, 500)) + [2_000_000]
```

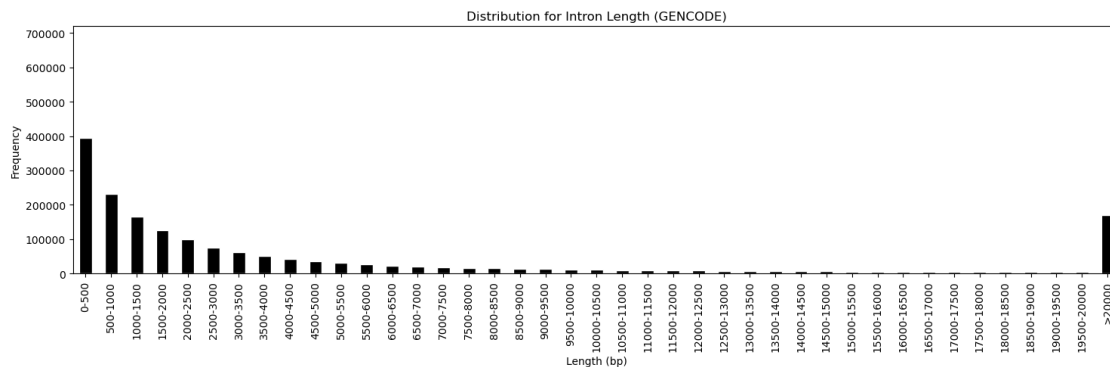
```

bin_labels = [f"{i}-{i+500}" for i in range(0, 20000, 500)] + ['>20000']
binned = pd.cut(intron_lengths, bins=bin_edges, labels=bin_labels, right=True)

bin_counts = binned.value_counts(sort=False)

plt.figure(figsize=(15, 5))
bin_counts.plot(kind='bar', color='black', edgecolor='white')
plt.xlabel("Length (bp)")
plt.ylabel("Frequency")
plt.title("Distribution for Intron Length (GENCODE)")
plt.xticks(rotation=90)
plt.tight_layout()
plt.ylim(0,720_000)
plt.show()

```



```
[20]: min(intron_lengths), max(intron_lengths)
```

```
[20]: (2, 1240121)
```

The intron length distribution of GENCODE is highly skewed to the right, indicating a **strong prevalence of shorter introns**. The 0-500 bp bin has the highest frequency, approaching 400,000 introns. There's a rapid decrease in frequency as intron length increases, with subsequent bins like 500-1000 bp and 1000-1500 bp showing considerably lower, though still substantial, frequencies.

Despite the sharp decline, the distribution exhibits a very long and heavy tail, extending to exceptionally long intron lengths. Notably, the >20000 bp bin still contains a significant number of introns, with a frequency exceeding 150,000. This indicates that while the vast majority of introns are relatively short, a substantial population of very long introns also exists within the Refseq annotation.

**Note that the binning of values is per 500 bp.** This means that compared to the exon length distributions, introns are longer.

**Data Source: RefSeq (NCBI)**



```
[21]: refseq_introns_df = refseq_df[refseq_df['type'] == 'exon'].reset_index().
↳drop('index', axis=1)[['transcript_id', 'start', 'end']]
refseq_introns_df = refseq_introns_df.sort_values(['transcript_id', 'start']).
↳reset_index().drop('index', axis=1)
refseq_introns_df = refseq_introns_df.groupby('transcript_id').agg(list)
```

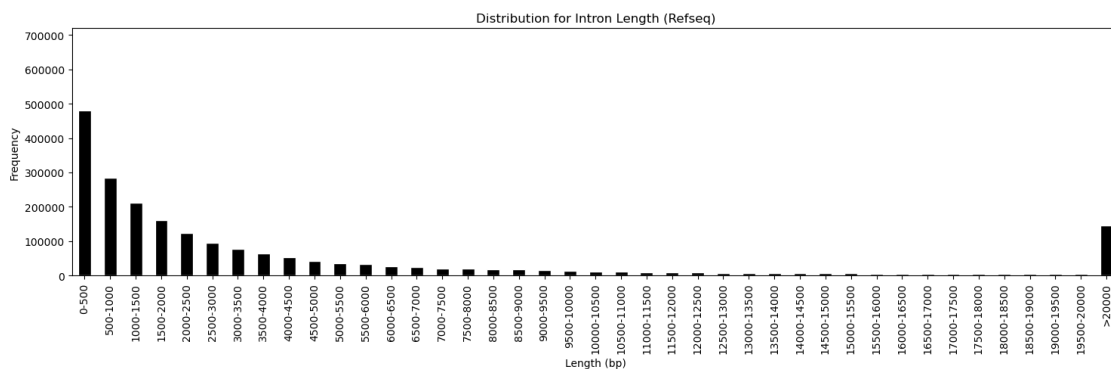
```
[22]: introns = []
i = 0
for each in refseq_introns_df.itertuples():
    _, start, end = each
    start = pd.Series(start + [0])
    end = pd.Series([0] + end)
    introns += list(start-end)[1:-1]
```

```
[23]: intron_lengths = pd.Series(introns)

bin_edges = list(range(0, 20001, 500)) + [2_000_000]
bin_labels = [f"{i}-{i+500}" for i in range(0, 20000, 500)] + ['>20000']
binned = pd.cut(intron_lengths, bins=bin_edges, labels=bin_labels, right=True)

bin_counts = binned.value_counts(sort=False)

plt.figure(figsize=(15, 5))
bin_counts.plot(kind='bar', color='black', edgecolor='white')
plt.xlabel("Length (bp)")
plt.ylabel("Frequency")
plt.title("Distribution for Intron Length (Refseq)")
plt.xticks(rotation=90)
plt.tight_layout()
plt.ylim(0,720_000)
plt.show()
```



```
[24]: min(intron_lengths), max(intron_lengths)
```

[24]: (2, 1160412)

The intron length distribution of Refseq is overwhelmingly skewed to the right, signifying that the vast majority of introns are relatively short. The 0-500 bp bin shows the highest frequency, with almost 500,000 introns. Following this, the frequency rapidly decreases as intron length increases. For example, the 500-1000 bp bin has a frequency below 300,000, and subsequent bins show a continuous decline.

Despite the sharp initial drop, the distribution exhibits a very long and heavy tail, extending to exceptionally long intron lengths. Notably, even though the frequencies are very low for most of the longer bins, there's a significant resurgence in the final >20000 bp bin, which contains a substantial number of introns, exceeding 100,000.

**Similar to GENCODE, the binning of values is per 500 bp.** This means that introns are longer than exons

**Comparison between GENCODE and Refseq intron length distribution** The distributions of intron length from GENCODE and Refseq are **similar to each other**. A subtle difference can be seen in the peaks. GENCODE's highest peak is approximately 450,000 while Refseq's peak is approximately 500,000. Both follows a unimodal pattern and have the same peak at 0-500 bp. Comparing these distributions to Sakharhar et. al (2005), the skewness and tails are the same although different binning of values are shown. The distribution of Sakharhar et. al is also significantly lower compared to the GENCODE and Refseq with probably the same case to the exon distribution.

Relative to the exon length distribution, **introns are considered longer compared to exons**. There are several reasons why introns are longer. According to Jo and Choi (2015), exons, which contain the protein-coding sequences, are often relatively short and conserved across species while introns, on the other hand, can vary greatly in length and may contain sequences that regulate gene expression or splicing.

Jo, B. S., & Choi, S. S. (2015). Introns: The Functional Benefits of Introns in Genomes. *Genomics & informatics*, 13(4), 112–118. <https://doi.org/10.5808/GI.2015.13.4.112>

## 1.6 Part 2. Greatest hits of the human genome

Although maybe a bit outdated, [Dolgin, 2017] compiled a list of the top 10 most studied genes in the human genome. Choose 3 out of the 10; and for each gene, answer the following questions:

1. **Where is it located in the genome, based on hg38 reference genome coordinates?**
2. **In which human tissues is the gene known to be expressed?**
3. **How many gene isoforms (alternatively spliced mRNAs) are known for each gene?**
4. **What is the average length of exons and average length of introns, across all isoforms?**
5. **How many protein isoforms are known for each gene?**
6. **What functions of the genes/proteins are known?**

Elie Dolgin. The most popular genes in the human genome. *Nature*, 551(7681):427–431, November 2017.

For this part, we referred to the RefSeq human genome annotation instead of GENCODE because NCBI provides more structured information about genes, making it easier to cross-check specific details.

### 1.6.1 Selected Genes

1. **TP53**
2. **TNF**
3. **EGFR**

To get the necessary details of the selected genes (**TP53**, **TNF**, **EGFR**), we select the rows that has a `gene` feature equal to "TP53" or "TNF" or "EGFR". Then, we extract the necessary information for each rows i.e. `gene_id`, `gene`, `transcript_id`, `featuretype`, `start`, and `end`. Finally, the `pd.read_sql()` function executes this query and loads the resulting data into a Pandas DataFrame called `genome_3_df` for further analysis.

```
[25]: query = """
        SELECT seqid, SUBSTRING(json_extract(attributes, '$.gene_id'), 3,
        ↳LENGTH(json_extract(attributes, '$.gene_id'))-4) as gene_id,
        SUBSTRING(json_extract(attributes, '$.gene'), 3,
        ↳LENGTH(json_extract(attributes, '$.gene'))-4) as gene,
        json_extract(attributes, '$.transcript_id') as transcript_id, featuretype
        ↳as type, start, end FROM features
        WHERE json_extract(attributes, '$.gene')= '["TP53"]'
        OR json_extract(attributes, '$.gene')= '["TNF"]'
        OR json_extract(attributes, '$.gene')= '["EGFR"]'
        """
        genome_3_df = pd.read_sql(query, refseq.conn)
```

To infer the average length of exons, we first calculate the length of each row by subtracting each element in the column `end` to each element in the column `start`. Then, group the exons according to its `gene` and `transcript_id`, and aggregate the data and calculate the average exon length.

To infer the average length of introns, we used the same formula as mentioned above to get the length of introns. We also sort the values according to `start` and `end`. We also group the exons according to its `transcript_id`, `start`, and `end`. We aggregate the data by compiling each data in a list. Lastly, `get_intron_mean` is implemented to get the average intron length. The functionality of `get_intron_mean` is the same as above.

```
[26]: def get_intron_mean(gene, introns_df):
        intron_mean = []
        for each in introns_df.loc[gene].itertuples():
            _, start, end = each
            start = pd.Series(start + [0])
            end = pd.Series([0] + end)
            introns = start - end
            intron_mean.append(introns[1:-1].mean())
        return pd.Series(intron_mean).mean()
```

```

genome_3_df['length'] = genome_3_df['end'] - genome_3_df['start']
exons_df = genome_3_df[(genome_3_df['type'] == '
↳ 'exon')][['gene', 'transcript_id', 'length']]
exons_mean_df = exons_df.groupby(['gene', 'transcript_id']).mean('length')
introns_df = genome_3_df[(genome_3_df['type'] == '
↳ 'exon')][['gene', 'transcript_id', 'start', 'end']]
introns_df = introns_df.sort_values(['gene', 'transcript_id', 'start', 'end'])
introns_df = introns_df.groupby(['gene', 'transcript_id']).agg(list)

```

### 1.6.2 TP53

Where is it located in the genome, based on hg38 reference genome coordinates? According to the [NCBI report on TP53 gene](#), the location of the TP53 gene in GRCh38.p14 is **7668421..7687490**. The query results of TP53 gene start and end columns are within the bounds of the reported location.

```

[27]: tp53_df = genome_3_df[(genome_3_df['type'] == 'transcript') &
↳ (genome_3_df['gene'] == 'TP53')]
tp53_df

```

```

[27]:
      seqid  gene_id  gene      transcript_id      type      start  \
556  NC_000017.11    TP53  TP53  ["NR_176326.1"]  transcript  7668421
567  NC_000017.11    TP53  TP53  ["NM_001276761.3"]  transcript  7668421
589  NC_000017.11    TP53  TP53  ["NM_001126112.3"]  transcript  7668421
613  NC_000017.11    TP53  TP53  ["NM_001407269.1"]  transcript  7668421
635  NC_000017.11    TP53  TP53  ["NM_001407268.1"]  transcript  7668421
659  NC_000017.11    TP53  TP53  ["NM_000546.6"]    transcript  7668421
683  NC_000017.11    TP53  TP53  ["NM_001276760.3"]  transcript  7668421
705  NC_000017.11    TP53  TP53  ["NM_001276695.3"]  transcript  7668421
727  NC_000017.11    TP53  TP53  ["NM_001126113.3"]  transcript  7668421
751  NC_000017.11    TP53  TP53  ["NM_001276696.3"]  transcript  7668421
773  NC_000017.11    TP53  TP53  ["NM_001126114.3"]  transcript  7668421
797  NC_000017.11    TP53  TP53  ["NM_001126118.2"]  transcript  7668421
818  NC_000017.11    TP53  TP53  ["NM_001407262.1"]  transcript  7668421
843  NC_000017.11    TP53  TP53  ["NM_001407263.1"]  transcript  7668421
866  NC_000017.11    TP53  TP53  ["NM_001407265.1"]  transcript  7668421
888  NC_000017.11    TP53  TP53  ["NM_001407264.1"]  transcript  7668421
912  NC_000017.11    TP53  TP53  ["NM_001407266.1"]  transcript  7668421
936  NC_000017.11    TP53  TP53  ["NM_001407267.1"]  transcript  7668421
958  NC_000017.11    TP53  TP53  ["NM_001407270.1"]  transcript  7668421
982  NC_000017.11    TP53  TP53  ["NM_001407271.1"]  transcript  7668421
1004  NC_000017.11    TP53  TP53  ["NM_001276697.3"]  transcript  7668421
1021  NC_000017.11    TP53  TP53  ["NM_001126115.2"]  transcript  7668421
1038  NC_000017.11    TP53  TP53  ["NM_001276699.3"]  transcript  7668421
1055  NC_000017.11    TP53  TP53  ["NM_001126117.2"]  transcript  7668421

```

1072	NC_000017.11	TP53	TP53	["NM_001276698.3"]	transcript	7668421
1089	NC_000017.11	TP53	TP53	["NM_001126116.2"]	transcript	7668421

	end	length
556	7687490	19069
567	7687490	19069
589	7687490	19069
613	7687490	19069
635	7687490	19069
659	7687490	19069
683	7687490	19069
705	7687490	19069
727	7687490	19069
751	7687490	19069
773	7687490	19069
797	7687490	19069
818	7687490	19069
843	7687490	19069
866	7677451	9030
888	7677451	9030
912	7677451	9030
936	7677451	9030
958	7677451	9030
982	7677451	9030
1004	7675244	6823
1021	7675244	6823
1038	7675244	6823
1055	7675244	6823
1072	7675244	6823
1089	7675244	6823

**In which human tissues is the gene known to be expressed?** Based on the consensus dataset, the TP53 gene was described as having **low-specificity**. This means that the tissue is expressed at biologically relevant levels in most tissues. However, as seen in the bar plot above, the tissues in which they are expressed the most are:

### 1.6.3 Thymus

The thymus is responsible for the production and maturation of immune cells; including small lymphocytes that protects the body against foreign antigens [1]. During the development of T-Cells, the integrity of thymic epithelial cells are essential to their development. This integrity is maintained by p53, a protein which TP53 encodes. A study revealed that deactivating this gene in mice within these cells would impair T-Cell maturation and compromise the function of the immune system [2]. Furthermore, the absence of functional p53 has been linked to the development

of thymic lymphomas. A study observed that p53 knockout mice predominantly developed thymic lymphomas [3].

#### **1.6.4 Tonsils**

The tonsils are tissues that help the body fight infection by filtering bacteria and viruses which enter through the mouth and nose. Inside the tonsils, B-cells rapidly divide and inadvertently damage their own DNA during this replication process. To prevent these mutations from spreading, these cells rely on the p53 protein [5]. Furthermore, a study also noted that in 8 out of 8 tonsil samples, MDM2 parallels p53 nuclear expression which confirms that p53 is active and regulated even in healthy tissue [6].

#### **1.6.5 Skin**

The epidermis primarily acts as the first line of defense against ultraviolet radiation. Therefore, keratinocytes maintain a steady amount of p53 that can be rapidly stabilised when DNA photolisions appear. After an hour of UVB exposure, the levels of p53 protein starts to increase and reached maximum levels by 8-12 hours [7].

#### **1.6.6 Lymph Nodes**

The lymph nodes orchestrate the activation of immune cells. They push lymphocytes through rapid proliferation and somatic hyper-mutation, risking DNA breaks. p53 functions as the gatekeeper that culls these clones which have errors. High p53 transcription allows swift cell-cycle arrest and apoptosis if errors occur during replication [8].

#### **1.6.7 Fallopian Tubes**

The secretory epithelium of the distal fallopian tube is now known to be as the cell of origin for most high-grade serous ovarian carcinoms. Because of this, it carries TP53 and relies on p53 to safeguard against oxidative and hormonal stress. The lack of TP53 triggers chromosomal instability in fallopian-tube epithelial cells [9].

### **1.7 References**

- [1] Zdrojewicz Z, Pachura E, Pachura P. The Thymus: A Forgotten, But Very Important Organ. *Adv Clin Exp Med*. 2016 Mar-Apr;25(2):369-75.
- [2] Rodrigues, P. M., Ribeiro, A. R., Perrod, C., Landry, J. J. M., Araújo, L., Pereira-Castro, I., Benes, V., Moreira, A., Xavier-Ferreira, H., Meireles, C., & Alves, N. L. (2017). Thymic epithelial cells require p53 to support their long-term function in thymopoiesis in mice. *Blood*, 130(4), 478–488. <https://doi.org/10.1182/blood-2016-12-758961>
- [3] Dudgeon, C., Chan, C., Kang, W., Sun, Y., Emerson, R., Robins, H., & Levine, A. J. (2014). The evolution of thymic lymphomas in p53 knockout mice. *Genes & development*, 28(23), 2613–2620. <https://doi.org/10.1101/gad.252148.114>
- [4] Martinez-Valdez, H., Guret, C., de Bouteiller, O., Fugier, I., Banchereau, J., & Liu, Y. J. (1996). Human germinal center B cells express the apoptosis-inducing genes Fas, c-myc, P53, and Bax but not the survival gene bcl-2. *The Journal of experimental medicine*, 183(3), 971–977. <https://doi.org/10.1084/jem.183.3.971>

[5] Martinez-Valdez, H., Guret, C., de Bouteiller, O., Fugier, I., Banchereau, J., & Liu, Y. J. (1996). Human germinal center B cells express the apoptosis-inducing genes Fas, c-myc, P53, and Bax but not the survival gene bcl-2. *The Journal of experimental medicine*, 183(3), 971–977. <https://doi.org/10.1084/jem.183.3.971>

[6] Martinez, J. C., Mateo, M., Sánchez-Beato, M., Villuendas, R., Orradre, J. L., Algara, P., Sánchez-Verde, L., García, P., López, C., & Martínez, P. (1995). MDM2 expression in lymphoid cells and reactive and neoplastic lymphoid tissue. Comparative study with p53 expression. *The Journal of pathology*, 177(1), 27–34. <https://doi.org/10.1002/path.1711770106>

[7] Lu, Y. P., Lou, Y. R., Yen, P., Mitchell, D., Huang, M. T., & Conney, A. H. (1999). Time course for early adaptive responses to ultraviolet B light in the epidermis of SKH-1 mice. *Cancer research*, 59(18), 4591–4602.

[8] Martinez, J. C., Mateo, M., Sánchez-Beato, M., Villuendas, R., Orradre, J. L., Algara, P., Sánchez-Verde, L., García, P., López, C., & Martínez, P. (1995). MDM2 expression in lymphoid cells and reactive and neoplastic lymphoid tissue. Comparative study with p53 expression. *The Journal of pathology*, 177(1), 27–34. <https://doi.org/10.1002/path.1711770106>

[9] Bronder, D., Tighe, A., Wangsa, D., Zong, D., Meyer, T. J., Wardenaar, R., Minshall, P., Hirsch, D., Heselmeyer-Haddad, K., Nelson, L., Spierings, D., McGrail, J. C., Cam, M., Nussen-zweig, A., Foijer, F., Ried, T., & Taylor, S. S. (2021). TP53 loss initiates chromosomal instability in fallopian tube epithelial cells. *Disease models & mechanisms*, 14(11), dmm049001. <https://doi.org/10.1242/dmm.049001>

**How many gene isoforms (alternatively spliced mRNAs) are known for each gene?** Based on the queried results, there are **26 isoforms of TP53 gene**.

```
[28]: tp53_df['transcript_id'].nunique()
```

```
[28]: 26
```

**What is the average length of exons and average length of introns, across all isoforms?**

```
[29]: print(f"Average Exon Length: {exons_mean_df.loc['TP53']['length'].mean():.2f}")
      print(f"Average Intron Length: {get_intron_mean('TP53', introns_df):.2f}")
```

```
Average Exon Length: 235.31
```

```
Average Intron Length: 1167.77
```

**How many protein isoforms are known for each gene?** According to Khoury et. al, the TP53 gene has **12 protein isoforms** due to alternative splicing. These different isoforms occurred through a combination of internal promoters, alternative splicing of intron 9, and alternative translation initiation.

Reference: Khoury MP, Bourdon JC. p53 Isoforms: An Intracellular Microprocessor? *Genes Cancer*. 2011 Apr;2(4):453-65. doi: 10.1177/1947601911408893. PMID: 21779513; PMCID: PMC3135639.

**What functions of the genes/proteins are known?** TP53 is a transcription factor which regulates the cell during interphase. It is responsible for tasks such as cell cycle arrest, DNA

repair, senescence, or apoptosis upon binding to specific DNA sequences in response to various stimuli which signals DNA damage. Outside of its primary role as a tumor suppressor, it also influences metabolism, autophagy, ferroptosis and aging processes.

### 1.7.1 TNF

**Where is it located in the genome, based on hg38 reference genome coordinates?**

The TNF gene mapped to the **NC\_\_** sequence represents its location on the main human genome reference, providing the standard version of the gene used in most annotations. In contrast, the mappings to **NT\_\_** sequences show alternative genomic versions of the same TNF gene—these may come from different individuals, represent alternate haplotypes, or be from unplaced or structurally variable regions of the genome. This does not mean there are more TNF genes, but rather that the same gene is being shown in different genomic contexts to reflect natural genetic variation across the human population.

In this case we removed all the **NT\_\_** and retain only the **NC\_\_** sequence because it represents the primary, standardized reference genome, ensuring consistency and comparability across analyses. Therefore, there is 1 TNF and this matches the [NCBI report on TNF gene](#) which is located at **31575565..31578336**.

```
[30]: tnf_df = genome_3_df[(genome_3_df['type'] == 'transcript') &
    ↪(genome_3_df['gene'] == 'TNF')]
tnf_df.iloc[[0]]
```

```
[30]:
```

	seqid	gene_id	gene	transcript_id	type	start	end	\
1	NC_000006.12	TNF	TNF	["NM_000594.4"]	transcript	31575565	31578336	
								length
1								2771

According to the [NCBI report on TNF gene](#), the location of the TNF gene in GRCh38.p14 is **31575565..31578336**. Not all the query results of TNF gene **start** and **end** columns match this reported location. However, one row with a **gene\_id** of "TNF" and **transcript\_id** of "NM\_000594.4", exactly matches the reported location. Looking at the **seqid**, the rows that does not match the reported location might be in a different sequence.

Although there are multiple "TNF" genes in the dataset, these are multiple copies of the same transcript. This means that there are only **1 isoform for TNF in the dataset**.

**In which human tissues is the gene known to be expressed?** Based on the consensus dataset used by the Human Protein Atlas (HPA), the TNF gene is classified as tissue-enhanced, meaning its strongest signals are concentrated on specific organs. Notably, the organs with the most expression belong to hematopo-lymphoid organs such as the bone marrow and other lymphoid tissues such as the lymph nodes, spleen and appendix. These organs are the sites where the body initiates shapes and terminates immune response.



### 1.7.2 Bone Marrow

The bone marrow shows the highest TNF signal among all tissues. The bone marrow is where new blood and immune cells are made. During infection, TNF pushes hematopoietic stem cells into limited proliferation when infection happens.

### 1.7.3 Lymph Nodes

Lymph nodes are small structures that help the body filter out harmful substances and coordinate immune response. Inside the lymph nodes, its follicles rely on a form of TNF delivered by B and T cells to keep the germinal-center network intact, where these cells also come from. In mice lacking TNF, this network collapses and antibody response is compromised [2].

### 1.7.4 Spleen

The spleen helps removed damaged blood cells and supports immune responses. The follicles of the spleen depend on the signaling mechanisms provided by the TNF. Furthermore, the amount of TNF also determines how the spleen works, affecting how immune cells grow and interact [3].

## 1.8 References

[1] Schuettpeitz, L. G., & Link, D. C. (2013). Regulation of hematopoietic stem cell activity by inflammation. *Frontiers in immunology*, 4, 204. <https://doi.org/10.3389/fimmu.2013.00204>

[2] Alexei V. Tumanov, Sergei I. Grivennikov, Andrei A. Kruglov, Yuriy V. Shebzukhov, Ekaterina P. Koroleva, Yulan Piao, Chang-Yi Cui, Dmitry V. Kuprash, Sergei A. Nedospasov; Cellular source and molecular form of TNF specify its distinct functions in organization of secondary lymphoid organs. *Blood* 2010; 116 (18): 3456–3464. doi: <https://doi.org/10.1182/blood-2009-10-249177>

[3] Jeucken, K. C. M., Kaaij, M. H., Rip, J., van Rooijen, C. C. N., Kan, Y. Y., Corneth, O. B. J., van Hamburg, J. P., & Tas, S. W. (2022). Splenic Architecture and Function Requires Tight Control of Transmembrane TNF Expression. *International journal of molecular sciences*, 23(4), 2229. <https://doi.org/10.3390/ijms23042229>

**How many gene isoforms (alternatively spliced mRNAs) are known for each gene?**  
Based on the queried results, there is only **1 isoform** for the TNF gene.

```
[37]: tnf_df.iloc[[0]]
```

```
[37]:      seqid gene_id gene  transcript_id      type      start      end \
1  NC_000006.12    TNF  TNF  ["NM_000594.4"]  transcript  31575565  31578336

      length
1      2771
```

**What is the average length of exons and average length of introns, across all isoforms?**

```
[30]: print(f"Average Exon Length: {exons_mean_df.loc['TNF']['length'].mean():.2f}")
      print(f"Average Intron Length: {get_intron_mean('TNF', introns_df):.2f}")
```

Average Exon Length: 421.06

Average Intron Length: 350.67

**How many protein isoforms are known for each gene?** The terms “TNF” and “TNF- ” are often used interchangeably to refer to tumor necrosis factor alpha, a key pro-inflammatory cytokine. TNF is encoded by a single gene and has only **one true protein isoform**, derived from a single mRNA transcript. However, this protein can exist in two distinct forms: a membrane-bound form (mTNF ) and a soluble form (sTNF ). The soluble form is not generated through alternative splicing or transcriptional variation, but rather through post-translational cleavage of mTNF by the metalloprotease ADAM17 (TACE), which removes its membrane-bound property.

Black, R. A., Rauch, C. T., Kozlosky, C. J., Peschon, J. J., Slack, J. L., Wolfson, M. F., Castner, B. J., Stocking, K. L., Reddy, P., Srinivasan, S., Nelson, N., Boiani, N., Schooley, K. A., Gerhart, M., Davis, R., Fitzner, J. N., Johnson, R. S., Paxton, R. J., March, C. J., & Cerretti, D. P. (1997). A metalloproteinase disintegrin that releases tumour-necrosis factor-alpha from cells. *Nature*, 385(6618), 729–733. <https://doi.org/10.1038/385729a0>

Grimstad, Ø. (2016). Tumor Necrosis Factor and the Tenacious . *JAMA Dermatology*, 152(5), 557. <https://doi.org/10.1001/jamadermatol.2015.4322>

**What functions of the genes/proteins are known?** The TNF gene is mainly produced by macrophages which binds to TNFRSF1A and TNFRSF1A to activate various pathways responsible for chronic inflammatory responses. Furthermore, other isoforms of TNF such as tmTNF can also engage in reverse signaling, modulating apoptosis, cell survival, and cytokine production in various immune cell types. Finally, it is also a pyrogen, inducing fever via hypothalamic prostaglandin synthesis, and contributes to the pathology of autoimmune diseases, sepsis, and insulin resistance.

### 1.8.1 EGFR

```
[18]: egfr_df = genome_3_df[(genome_3_df['type'] == 'transcript') &
    ↪(genome_3_df['gene'] == 'EGFR')]
egfr_df
```

```
[18]:
```

	seqid	gene_id	gene	transcript_id	type	start	\
13	NC_000007.14	EGFR	EGFR	["NM_005228.5"]	transcript	55019017	
72	NC_000007.14	EGFR	EGFR	["NM_001346899.2"]	transcript	55019017	
129	NC_000007.14	EGFR	EGFR	["NM_001346941.2"]	transcript	55019017	
176	NC_000007.14	EGFR	EGFR	["NM_001346898.2"]	transcript	55019017	
233	NC_000007.14	EGFR	EGFR	["NM_001346897.2"]	transcript	55019017	
288	NC_000007.14	EGFR	EGFR	["NM_201284.2"]	transcript	55019017	
323	NC_000007.14	EGFR	EGFR	["NM_201282.2"]	transcript	55019017	
358	NC_000007.14	EGFR	EGFR	["NM_201283.2"]	transcript	55019017	
381	NC_000007.14	EGFR	EGFR	["XM_047419953.1"]	transcript	55080469	
439	NC_000007.14	EGFR	EGFR	["XM_047419952.1"]	transcript	55095532	
497	NC_000007.14	EGFR	EGFR	["NM_001346900.2"]	transcript	55109839	

```

end
13  55211628
72  55211628
129 55211628
176 55203076
```

233 55203076  
 288 55171037  
 323 55168635  
 358 55156939  
 381 55211628  
 439 55211628  
 497 55211628

According to the [NCBI report on EGFR gene](#), the location of the EGFR gene in GRCh38.p14 is **55019017..55211628**. The query results of EGFR gene **start** and **end** columns are within the bounds of the reported location.

**In which human tissues is the gene known to be expressed?** Based on the consensus dataset, the EGFR gene is classified as tissue-enhanced meaning it is found everywhere, with the most levels in the placenta. EGFR encodes the epidermal growth-factor receptor, a surface protein that tells cells when to grow, divide, or fuse.

## 1.8.2 The Placenta

The placenta is an organ that forms in the uterus during pregnancy. The placenta is connected to fetus through the umbilical cord. It serves multiple functions related to endocrine, immune and physiological processes [1] During the first trimester, EGFR sits mainly on cytotrophoblasts which sit on the surface of the placenta. By weeks 6-12, it shifts to the overlying syncytiotrophoblast which handles nutrient and gas exchange between the mother and the fetus. In mice that lack EGFR, placentas are undersized and can't support normal fetal growth [2].

When EGFR signals drop too low in humans, doctors often see fetal-growth restriction and preeclampsia which are linked to poor trophoblast invasion and high blood pressure in pregnancy [3]. Conversely, over-active or “hijacked” EGFR pathways can push trophoblasts to proliferate abnormally, which researchers are exploring as one root of gestational trophoblastic disease.

At term, studies show plenty of EGFR along the syncytiotrophoblast membrane which responds to maternal growth factors in the blood [4]. Because of this, high specific expression, drug-delivery teams now design EGFR-targeted lipid nanoparticles to ferry mRNA or drugs straight to the placenta while sparing other organs which in the future could treat FGR or preeclampsia without harming the fetus [5].

[1] Herrick EJ, Bordonni B. Embryology, Placenta. [Updated 2023 May 1]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2025 Jan-. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK551634/>

[2] Renshall, L. J., Beards, F., Evangelinos, A., Greenwood, S. L., Brownbill, P., Stevens, A., Sibley, C. P., Aplin, J. D., Johnstone, E. D., Teesalu, T., & Harris, L. K. (2021). Targeted Delivery of Epidermal Growth Factor to the Human Placenta to Treat Fetal Growth Restriction. *Pharmaceutics*, 13(11), 1778. <https://doi.org/10.3390/pharmaceutics13111778>

[3] Clemente, L., & Bird, I. M. (2022). The epidermal growth factor receptor in healthy pregnancy and preeclampsia. *Journal of molecular endocrinology*, 70(1), e220105.

<https://doi.org/10.1530/JME-22-0105>

[4] Magid, M., Nanney, L. B., Stoscheck, C. M., & King, L. E., Jr (1985). Epidermal growth factor binding and receptor distribution in term human placenta. *Placenta*, 6(6), 519–526. [https://doi.org/10.1016/s0143-4004\(85\)80005-9](https://doi.org/10.1016/s0143-4004(85)80005-9)

[5] Geisler, H. C., Ghalsasi, A. A., Safford, H. C., Swingle, K. L., Thatte, A. S., Mukalel, A. J., Gong, N., Hamilton, A. G., Han, E. L., Nachod, B. E., Padilla, M. S., & Mitchell, M. J. (2024). EGFR-targeted ionizable lipid nanoparticles enhance in vivo mRNA delivery to the placenta. *Journal of Controlled Release*, 371, 455–469. <https://doi.org/10.1016/j.jconrel.2024.05.036>

**How many gene isoforms (alternatively spliced mRNAs) are known for each gene?**  
Based on the queried results, there are **11 isoforms of EGFR gene**.

```
[22]: egfr_df['transcript_id'].nunique()
```

```
[22]: 11
```

**What is the average length of exons and average length of introns, across all isoforms?**

```
[31]: print(f"Average Exon Length: {exons_mean_df.loc['EGFR']['length'].mean():.2f}")  
      print(f"Average Intron Length: {get_intron_mean('EGFR', introns_df):.2f}")
```

Average Exon Length: 395.23

Average Intron Length: 7452.88

**How many protein isoforms are known for each gene?** EGFR protein has **4 isoforms** which are results of alternative splicing. Isoform 1 is the canonical one depicted above. Other isoforms are isoform 2 (p60, 405 aa, 44,664 Da), isoform 3 (p110, 705 aa, 77,312 Da) and isoform 4 (628 aa, 69,228 Da).

Ayca Circir Hatil ; Esra Cicek ; Merve Oyken ; A.Elif Erson-Bensan EGFR (Epidermal Growth Factor Receptor) Atlas Genet Cytogenet Oncol Haematol. 2019-12-01 Online version: [http://atlasgeneticsoncology.org/gene/147/egfr-\(epidermal-growth-factor-receptor\)](http://atlasgeneticsoncology.org/gene/147/egfr-(epidermal-growth-factor-receptor))

According to [UniProt EGFR Entry](#), isoform 1 is the main, active version of the receptor that responds to EGF and related ligands by triggering important signaling pathways inside the cell, such as those for growth, survival, and movement. Isoform 2, however, is produced through alternative splicing and lacks some critical parts needed for signaling. As a result, it cannot activate these pathways and may even block EGF signaling by acting as a decoy—either by binding to EGF without sending a signal or by preventing isoform 1 from working properly. This makes isoform 2 a natural inhibitor of EGF action.

**What functions of the genes/proteins are known?** EGFR binds ligands such as epidermal growth factor and transforming growth factor- $\alpha$  which is essential for normal embryonic development, wound healing, and maintenance of epithelial tissue integrity. Aberrations in EGFR expression or activation are implicated in various cancers (e.g., lung, glioblastoma, head and neck), making EGFR a major therapeutic target in oncology.

This notebook was converted with [convert.ploomber.io](#)