## Preparing stage

Load naive-, memory-, and plasma-cells results from https://www.ncbi.nlm.nih.gov/igblast/igblast.cgi (https://www.ncbi.nlm.nih.gov/igblast/igblast.cgi)
Settings:
  Number of germline gene: 1 for V, J, and D
  Alignment format: AIRR
  Number of clonotypes to show: 1000

In [2]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:

```
def cut_variations(gene):
    return gene.split('*')[0]

def calc_mutations(seq_germ_t):
    cell_seq = np.array(list(seq_germ_t[0]))
    germ_seq = np.array(list(seq_germ_t[1]))
    return sum(cell_seq != germ_seq)
```

In [4]:

```
def get_normalized_dataset(filename):
    df = pd.read_csv(filename, sep='\t')
    df['v_gene'] = df['v_call'].map(cut_variations)
    df['d_gene'] = df['d_call'].fillna("N/A").map(cut_variations)
    df['j_gene'] = df['j_call'].map(cut_variations)
    df['v_mutations_cnt'] = df[['v_sequence_alignment', 'v_germline_alignment']].apply(calc_mutations, axis=1)
    df['cdr3_len'] = df['cdr3'].str.len()
    return df
```

In [5]:

```
df_nai = get_normalized_dataset("./naive_germline_genes_full.csv")
df_mem = get_normalized_dataset("./memory_germline_genes_full.csv")
df_pla = get_normalized_dataset("./plasma_germline_genes_full.csv")
```

## Task 1

Analyze the joint usage of V and J genes: for each sequence find the closest germline V and J genes, list of all VJ pairs occurring in the sample, and create a plot (e.g., heatmap) showing the number of sequences for each VJ pair.

Compare VJ usages in three samples and find samples with the smallest and highest number of VJ combinations.

In [6]:

```
def get_VJ_pivot(src_df):
    tmp_df = src_df[['v_gene', 'j_gene', 'sequence_id']].groupby(['v_gene', 'j_gene']).count().reset_index()
    tmp_df.columns = ['v_gene', 'j_gene', 'count']
    genes_pivot = tmp_df.pivot(index='v_gene', columns='j_gene', values='count')
    genes_pivot.fillna(0, inplace=True)
    return genes_pivot

def draw_VJ_heatmap(df_nai, df_mem, df_pla):
    fig, axs = plt.subplots(ncols=3)
    axs[0].set_title('Naive cells')
    axs[1].set_title('Memory cells')
    axs[2].set_title('Plasma cells')
    sns.set(rc={'figure.figsize':(24,24)})
    sns.heatmap(get_VJ_pivot(df_nai), annot=True, cmap="YlOrRd", ax=axs[0])
    sns.heatmap(get_VJ_pivot(df_mem), annot=True, cmap="YlOrRd", ax=axs[1])
    sns.heatmap(get_VJ_pivot(df_pla), annot=True, cmap="YlOrRd", ax=axs[2])
    plt.show()
```

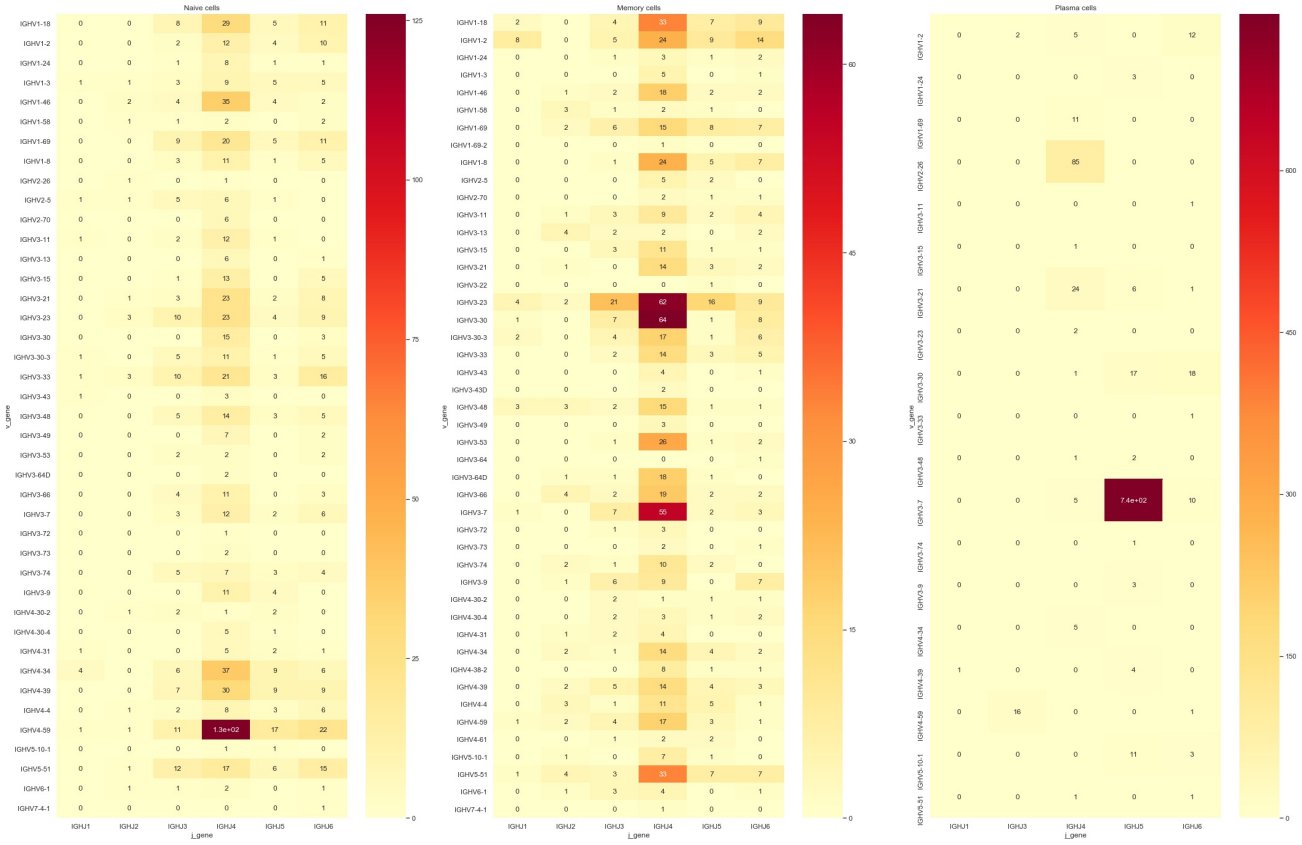
In [11]:

```
print(f"Naive cells VJ-usages: {df_nai.groupby(['v_gene', 'j_gene']).count().shape[0]}")
print(f"Memory cells VJ-usages: {df_mem.groupby(['v_gene', 'j_gene']).count().shape[0]}")
print(f"Plasma cells VJ-usages: {df_pla.groupby(['v_gene', 'j_gene']).count().shape[0]}")
```

```
Naive cells VJ-usages: 144
Memory cells VJ-usages: 171
Plasma cells VJ-usages: 32
```

In [28]:

```
draw_VJ_heatmap(df_nai, df_mem, df_pla)
```




As we can see, plasma cells have the lowest amount of VJ combinations (32 combinations against 144 for naive cells and 171 for memory cells).
This can happen because plasma cells produce the first wave of protective antibodies and help clear infection immediately. They should have the highest affinity towards their target antigen.
This leads to the lowest variation of VJ-usage. Memory cells may be more variative in order to protect from mutated antigens next time.

## Task 2

Find 10 most used V genes in the sample and analyze their mutability. For each gene, analyze sequences aligned to it and compute the number of differences in each alignment. Mutability is the distribution of the number of differences. Visualize mutability of 10 most used V genes in any convenient form (e.g., using boxplot).

Compare mutability of V genes, find samples with the smallest and highest mutability, and provide a biological explanation of differences between samples.

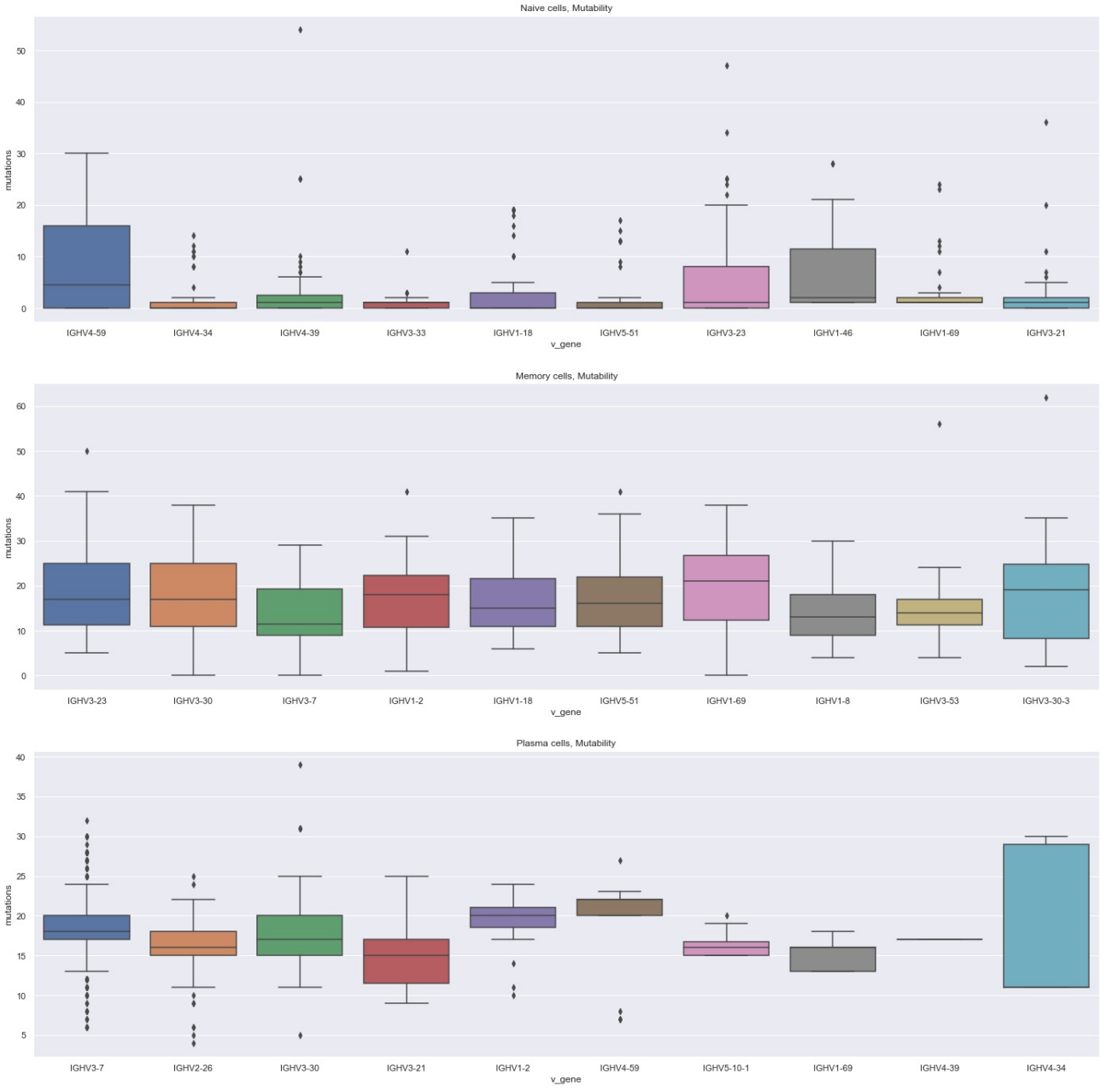
In [29]:

```
def get_mutations_distributions(df, n):
    tmp_df = df.groupby('v_gene')['v_mutations_cnt'].apply(list).reset_index(name='mutations')
    tmp_df['cnt'] = tmp_df.mutations.map(lambda l: len(l))
    tmp_df = tmp_df.sort_values('cnt', ascending=False)[:n].explode('mutations')
    tmp_df['mutations'] = tmp_df['mutations'].astype(float)
    return tmp_df

def draw_V_boxplots(nai_distr, mem_distr, pla_distr):
    fig, axs = plt.subplots(nrows=3)
    axs[0].set_title('Naive cells, Mutability')
    axs[1].set_title('Memory cells, Mutability')
    axs[2].set_title('Plasma cells, Mutability')
    sns.set(rc={'figure.figsize':(36,24)})
    sns.boxplot(x='v_gene', y='mutations', data=nai_distr, ax=axs[0])
    sns.boxplot(x='v_gene', y='mutations', data=mem_distr, ax=axs[1])
    sns.boxplot(x='v_gene', y='mutations', data=pla_distr, ax=axs[2])
    plt.show()
```

In [30]:

```
nai_distr = get_mutations_distributions(df_nai, 10)
mem_distr = get_mutations_distributions(df_mem, 10)
pla_distr = get_mutations_distributions(df_pla, 10)
draw_V_boxplots(nai_distr, mem_distr, pla_distr)
```




The lowest mutability is in the naive-cells sample. Both of the memory-cells and plasma-cells samples have high mutability. As a difference, the memory-cells mutability has a higher variance.
I can assume that these results depend on somatic hypermutation (SMH) - the process in which activated B-cells get more mutations in their coding regions to increase or decrease their complementarity. Naive cells have not reached these stage yet, so they have fewer mutations.

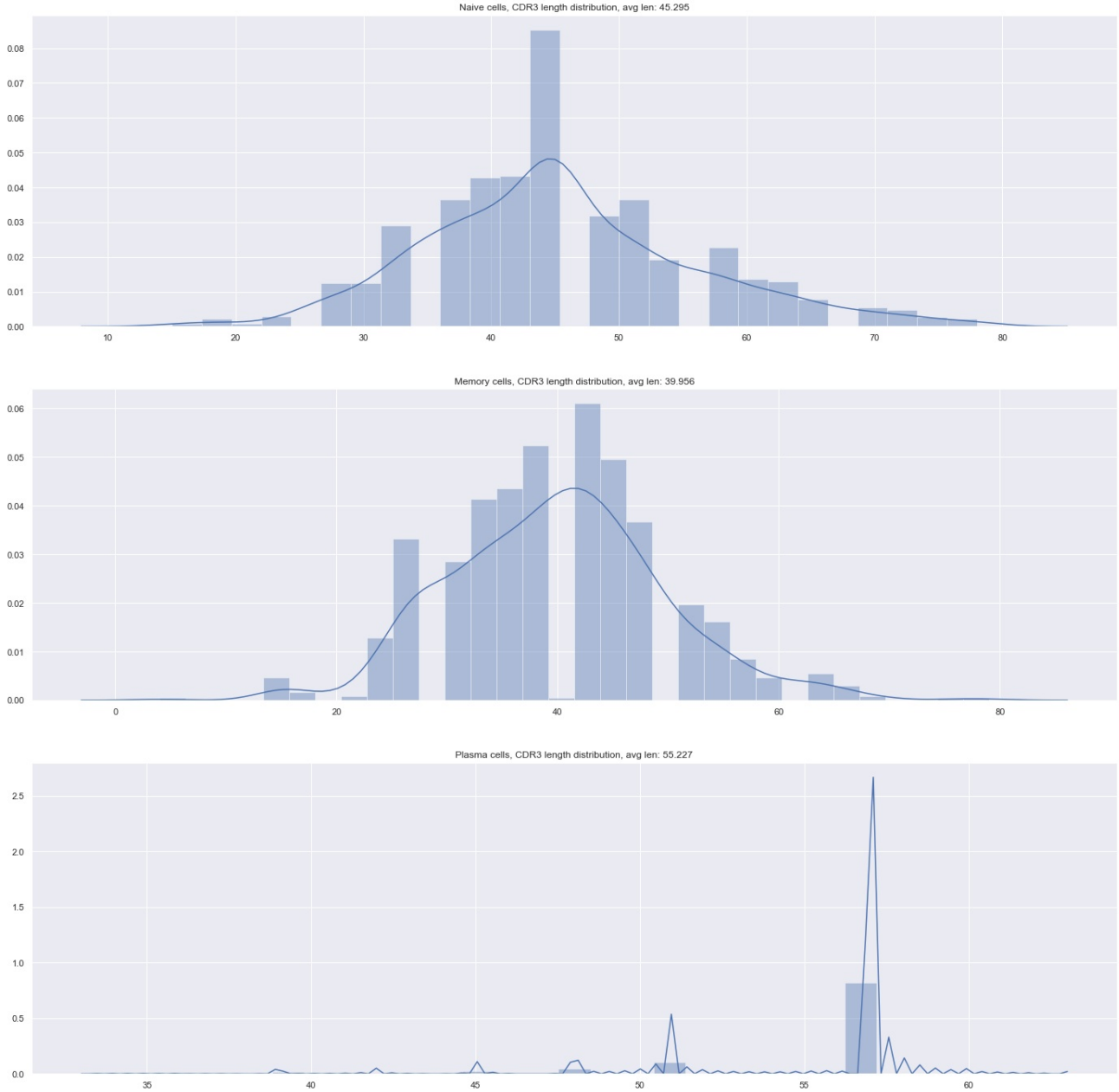
## Task 3

Visualize distributions of CDR3 lengths.

Compare distributions of CDR3 lengths and explain why CDR3 lengths in the PLASMA sample differ from the NAIVE and MEMORY samples.

In [41]:

```
fig, axs = plt.subplots(nrows=3)
axs[0].set_title(f'Naive cells, CDR3 length distribution, avg len: {df_nai.cdr3_len.values.mean()}')
axs[1].set_title(f'Memory cells, CDR3 length distribution, avg len: {df_mem.cdr3_len.values.mean()}')
axs[2].set_title(f'Plasma cells, CDR3 length distribution, avg len: {df_pla.cdr3_len.values.mean()}')
sns.set(rc={'figure.figsize':(24,24)})
sns.distplot(df_nai.cdr3_len.values, ax=axs[0])
sns.distplot(df_mem.cdr3_len.values, ax=axs[1])
sns.distplot(df_pla.cdr3_len.values, ax=axs[2])
plt.show()
```




The plasma cells should be much more specific to the current antigen, so maybe just the cells with this CDR3 length are enough complementary.

## Task 4

Compute the fraction of non-productive sequences in the sample. Both IgBlast and DiversityAnalyzer report productiveness of input sequences as a part of the output.

Compare fractions of productive sequences in all samples.

In [45]:

```
nai_cnt = df_nai[df_nai.productive == 'F'].shape[0]
mem_cnt = df_mem[df_mem.productive == 'F'].shape[0]
pla_cnt = df_pla[df_pla.productive == 'F'].shape[0]
print(f"Naive cells, non-productive sequences: {nai_cnt/df_nai.shape[0]*100}% ({nai_cnt}/{df_nai.shape[0]})")
print(f"Memory cells, non-productive sequences: {mem_cnt/df_mem.shape[0]*100}% ({mem_cnt}/{df_mem.shape[0]})")
print(f"Plasma cells, non-productive sequences: {pla_cnt/df_pla.shape[0]*100}% ({pla_cnt}/{df_pla.shape[0]})")
```

```
Naive cells, non-productive sequences: 3.1% (31/1000)
Memory cells, non-productive sequences: 7.9% (79/1000)
Plasma cells, non-productive sequences: 3.2% (32/1000)
```

We see, that naive-cells and plasma-cells samples include fewer non-productive sequences, meanwhile memory-cells sample has twice-higher percent of them.
It can happen because memory cells can go through several cycles of SHM and become less productive. Naive cells are already preselected in the bone marrow and have no extra mutations. Plasma cells are also preselected in germinal centers, they are highly targeted to the current antigen and usually can't adapt to the new one.