2. Поиск сайтов протеолиза белков протеосомой по данным VDJdb.

Смирнов Антон Сергеевич

Jan 20, 2023

Оглавление

Предварительные настройки

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import json
from datetime import datetime
from urllib.error import HTTPError
from Bio import Entrez, SeqIO
from io import StringIO
import logomaker
import re
```

Всегда говори NCBI кто ты

```
Entrez.email = "anton.smirnov.9910@gmail.com"
Entrez.api_key = "67a161eb14f134f9d7e50e111f957429f808"


# Print iterations progress
def printProgressBar (iteration, total, prefix = '', suffix = '', decimals = 1, length = 100, fill = '', printEnd = "\r"):
    """
    Call in a loop to create terminal progress bar
    @params:
        iteration   - Required  : current iteration (Int)
        total       - Required  : total iterations (Int)
        prefix      - Optional  : prefix string (Str)
        suffix      - Optional  : suffix string (Str)
        decimals    - Optional  : positive number of decimals in percent complete (Int)
        length      - Optional  : character length of bar (Int)
        fill        - Optional  : bar fill character (Str)
```

```
        printEnd    - Optional  : end character (e.g. "\r", "\r\n") (Str)
    """
    percent = ("{0:." + str(decimals) + "f}").format(100 * (iteration / float(total)))
    filledLength = int(length * iteration // total)
    bar = fill * filledLength + '-' * (length - filledLength)
    print(f'\r{prefix} |{bar}| {percent}% {suffix}', end = printEnd)
    # Print New Line on Complete
    if iteration == total:
        print()

def isCorrectSequence(seq):
    alphabet = list("ACDEFGHIKLMNPQRSTVWY")
    seq = seq.strip()
    for i in range(0,len(seq)):
        if seq[i] not in alphabet:
            return False
    return True
```

Для описания сайтов разрезания берем С-конец эпитопа. Для полноты картины нужна аминокислота, следующая после С-конца эпитопа. Для этого нужна референсная последовательность и большая часть кода посвящена их извлечению.

```
vdjdb = pd.read_csv("../data/source/VDJdb-2022-11-13-22-41.tsv", sep = "\t", header=0)
vdjdb.head()
```

	complex.id	Gene	...	CDR3fix Score
0	1	TRA	...	{"cdr3": "CIVRAPGRADMRF", "cdr3_old": "CIVRAPG... 2
1	1	TRB	...	{"cdr3": "CASSYLPGQGDHYSNQPQHF", "cdr3_old": "... 2
2	0	TRB	...	{"cdr3": "CASSFEAGQGFFSNQPQHF", "cdr3_old": "C... 2
3	2	TRA	...	{"cdr3": "CAVPSGAGSYQLTF", "cdr3_old": "CAVPSG... 2
4	2	TRB	...	{"cdr3": "CASSFEPGQGFYSNQPQHF", "cdr3_old": "C... 2

[5 rows x 17 columns]

```
len(vdjdb['Epitope'].unique())
```

1150

Первичная фильтрация

Сайты разрезания ищем только для человеческих MHC 1 класса

```
vdjdb_filtered = vdjdb[vdjdb["MHC class"] == "MHCI"]
vdjdb_filtered.shape
```

(86160, 17)

```
vdjdb_filtered = vdjdb_filtered[vdjdb_filtered["Species"] == "HomoSapiens"]
vdjdb_filtered.shape
```

(78770, 17)

Оставляем с нормально записанной последовательностью и не пустыми полями про источник

```
vdjdb_filtered = vdjdb_filtered[vdjdb_filtered["Epitope"].apply(isCorrectSequence)]
vdjdb_filtered.shape
```

(78770, 17)

```
vdjdb_filtered = vdjdb_filtered[~vdjdb_filtered["Epitope gene"].isna()]
vdjdb_filtered.shape
```

(78738, 17)

```
vdjdb_filtered = vdjdb_filtered[~vdjdb_filtered["Epitope species"].isna()]
vdjdb_filtered.shape
```

(78738, 17)

```
vdjdb_filtered = vdjdb_filtered[~vdjdb_filtered["Reference"].isna()]
vdjdb_filtered.shape
```

(77227, 17)

Очистка

Названия генов

```
print(vdjdb_filtered["Epitope gene"].unique())
```

['Nef' 'pp65' 'Nucleocapsid' 'ORF3' 'ORF1ab' 'Spike' 'NDC1' 'TKT' 'SEC24A'
 'AKAP13' 'EXOC8' 'PABPC1' 'MLANA' 'BRLF1' 'Gag' 'IE1' 'EBNA1' 'BZLF1'
 'Tax' 'EBNA3A' 'M' 'NY-ESO-1' 'UL40' 'EBNA6' 'Tel1' 'ABCD3' 'BMLF1'
 'ELAVL4' 'INS' 'Pol' 'TERT' 'NS3' 'EMC' 'WT1' 'MAGE-A3' 'TITIN'
 'synthetic' 'M1-F5L' 'M1-G4E' 'GAG' 'Leader peptide' 'RT' 'NP338'
 'ANKRD30A' 'MAGE-A4' 'PMEL' 'P53' 'ENR' 'TP53' 'PIK3CA' 'NS4B' 'BST2'
 'IE2' 'UL49' 'NSP3' 'INS-DRiP' 'PTPRN' 'EBNA4' 'EBNA3B' 'pp50' 'NP'
 'LMP2A' 'T-Ag' '5T4' 'GANAB' 'GNL3L' 'PGM5' 'SNX24' 'FNDC3B' 'SMARCD3'
 'CDK4' 'NS5B' 'SLC30A8' 'KRAS' 'MAGEA6' 'PDS5A' 'MED13' 'RFC5' 'BRAP'
 'GINS1' 'DPY19L4' 'RNF19B' 'ASTN1' 'MLL2' 'BCL2L1' 'PLA2G6' 'E7' 'LMP1'
 'MAGEA1' 'TYR' 'KanJ' 'MAGEA3' 'KLK3' 'PLCD3' 'PPM1' 'SRPX' 'AHNAK'
 'AFMID' 'HELZ2' 'CENPL' 'TPX2' 'WDR46' 'HIVEP2' 'AMPH' 'Vpr' 'Vif'
 'Matrix' 'RNP' 'NSP12' 'TXNDC11' 'U2AF2' 'GEMIN' 'CD74' 'PDE4A' 'WDR87'
 'FANCI' 'CRISPLD1' 'KLHL7' 'ARMT1' 'gp100' 'SSX2' 'MAGE-A1' 'ABCB5'
 'MART1' 'Tyrosinase' 'NY-ESO' 'PORCN' 'AKAP9' 'ZDBF2' 'GCN1L1' 'CDKN2A'
 'PDE7B' 'POGK' 'MPV17' 'IE-1' 'UL29/28' 'ARHGAP35' 'p53' 'COL18A1'
 'KIF16B' 'KIAA1279' 'XPNPEP1' 'UGGT2' 'PHKA1' 'GNB5' 'FBXO21' 'RECQL5'
 'KIAA1967' 'KIAA0368' 'CADPS2' 'NUP98' 'KARS' 'CASP8' 'TUBGCP2' 'RNF213'
 'SKIV2L' 'H3F3B' 'API5' 'RNF10' 'PHLPP1' 'ZFYVE27' 'NBAS' 'PPM1F' 'ACTN4'
 'ME1' 'SF3B1' 'NRAS' 'ERBB2' 'IGF2BP2' 'ORF10' 'ORF14' 'ORF6' 'ORF7a'
 'ORF7b' 'ORF8' 'ORF9b' 'Envelope']

Исключим синтетические конструкты

```
vdjdb_filtered = vdjdb_filtered[vdjdb_filtered["Epitope gene"] != "synthetic"]
vdjdb_filtered.shape
```

(77161, 17)

Есть ли эпитопы, которые пришли из разных генов?

```
epitopes = vdjdb_filtered["Epitope"].unique()
for e in epitopes:
    v = vdjdb_filtered[vdjdb_filtered["Epitope"] == e]
    if len(v["Epitope gene"].unique()) > 1:
        print(f"{e} {v['Epitope gene'].unique()}")
```

IPLTEEAEL ['RT' 'Pol']
APRGPHGGAASGL ['NY-ESO-1' 'NY-ESO']
HMTEVVRHC ['P53' 'TP53' 'p53']

Исправляем аннотацию

```
vdjdb_filtered.loc[vdjdb_filtered["Epitope gene"].str.contains("RT"),"Epitope gene"] = "Pol"
vdjdb_filtered.loc[vdjdb_filtered["Epitope gene"].str.contains("NY-ESO"),"Epitope gene"] = "NY-ESO-1"
vdjdb_filtered.loc[vdjdb_filtered["Epitope gene"].str.contains("P53"),"Epitope gene"] = "TP53"
vdjdb_filtered.loc[vdjdb_filtered["Epitope gene"].str.contains("p53"),"Epitope gene"] = "TP53"
```

Названия видов

```
print(vdjdb_filtered["Epitope species"].unique())
```

['HIV-1' 'CMV' 'SARS-CoV-2' 'HomoSapiens' 'EBV' 'HTLV-1' 'InfluenzaA'
 'SaccharomycesCerevisiae' 'HCV' 'synthetic' 'DENV1' 'DENV3/4'
 'M.tuberculosis' 'HIV1' 'Homo sapiens' 'YFV' 'HSV-2' 'DENV2' 'MCPyV'
 'HPV' 'StreptomycesKanamyceticus' 'HIV' 'HCoV-HKU1']

Исключим синтетические конструкты

```
vdjdb_filtered = vdjdb_filtered[vdjdb_filtered["Epitope species"] != "synthetic"]
vdjdb_filtered.shape
```

(77157, 17)

Исправляем аннотацию

```
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("HIV"),"Epitope species"] = "HIV-1"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("HSV-2"),"Epitope species"] = "HSV2"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("EBV"),"Epitope species"] = "Human gammaherpesvirus 4"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("CMV"),"Epitope species"] = "Human betaherpesvirus 5"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("HomoSapiens"),"Epitope species"] = "Homo sapiens"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("SaccharomycesCerevisiae"),"Epitope species"] = "Saccharo
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("StreptomycesKanamyceticus"),"Epitope species"] = "Strept
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("M.tuberculosis"),"Epitope species"] = "Mycobacterium tuber
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("HPV"),"Epitope species"] = "Human papillomavirus"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("DENV1"),"Epitope species"] = "Dengue virus 1"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("DENV2"),"Epitope species"] = "Dengue virus 2"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("InfluenzaA"),"Epitope species"] = "Influenza A virus"
vdjdb_filtered.loc[vdjdb_filtered["Epitope species"].str.contains("MCPyV"),"Epitope species"] = "Merkel cell polyomavirus"
```

```
vdjdb_filtered = vdjdb_filtered[vdjdb_filtered["Epitope species"] != "DENV3/4"]
```

```
print(vdjdb_filtered["Epitope species"].unique())
```

```
['HIV-1' 'Human betaherpesvirus 5' 'SARS-CoV-2' 'Homo sapiens'
 'Human gammaherpesvirus 4' 'HTLV-1' 'Influenza A virus'
 'Saccharomyces cerevisiae' 'HCV' 'Dengue virus 1'
 'Mycobacterium tuberculosis' 'YFV' 'HSV2' 'Dengue virus 2'
 'Merkel cell polyomavirus' 'Human papillomavirus'
 'Streptomyces kanamyceticus' 'HCoV-HKU1']
```

Описательная статистика

```
vdjdb_filtered["Epitope species"].value_counts().reset_index().loc[0:4]
```

	index	Epitope species
0	Human betaherpesvirus 5	37945
1	Human gammaherpesvirus 4	11291
2	Influenza A virus	10509
3	SARS-CoV-2	7135
4	Homo sapiens	4870

```
vdjdb_filtered["Species"].value_counts()
```

```
HomoSapiens    76978
Name: Species, dtype: int64
```

```
vdjdb_filtered["Epitope gene"].value_counts()
```

```
IE1        28142
M          10038
pp65        8941
EBNA4       5032
Spike       2896
           ...
TUBGCP2        1
KARS           1
CADPS2         1
CDKN2A         1
PHLPP1         1
```

Name: Epitope gene, Length: 168, dtype: int64

```
vdjdb_filtered["Score"].value_counts()
```

```
0    69400
1     4575
2     1584
3     1419
Name: Score, dtype: int64
```

```
print(max(vdjdb_filtered["Epitope"].str.len()))
```

```
20
```

```
epitopes = vdjdb_filtered["Epitope"].unique()
```

```
len(epitopes)
```

```
959
```

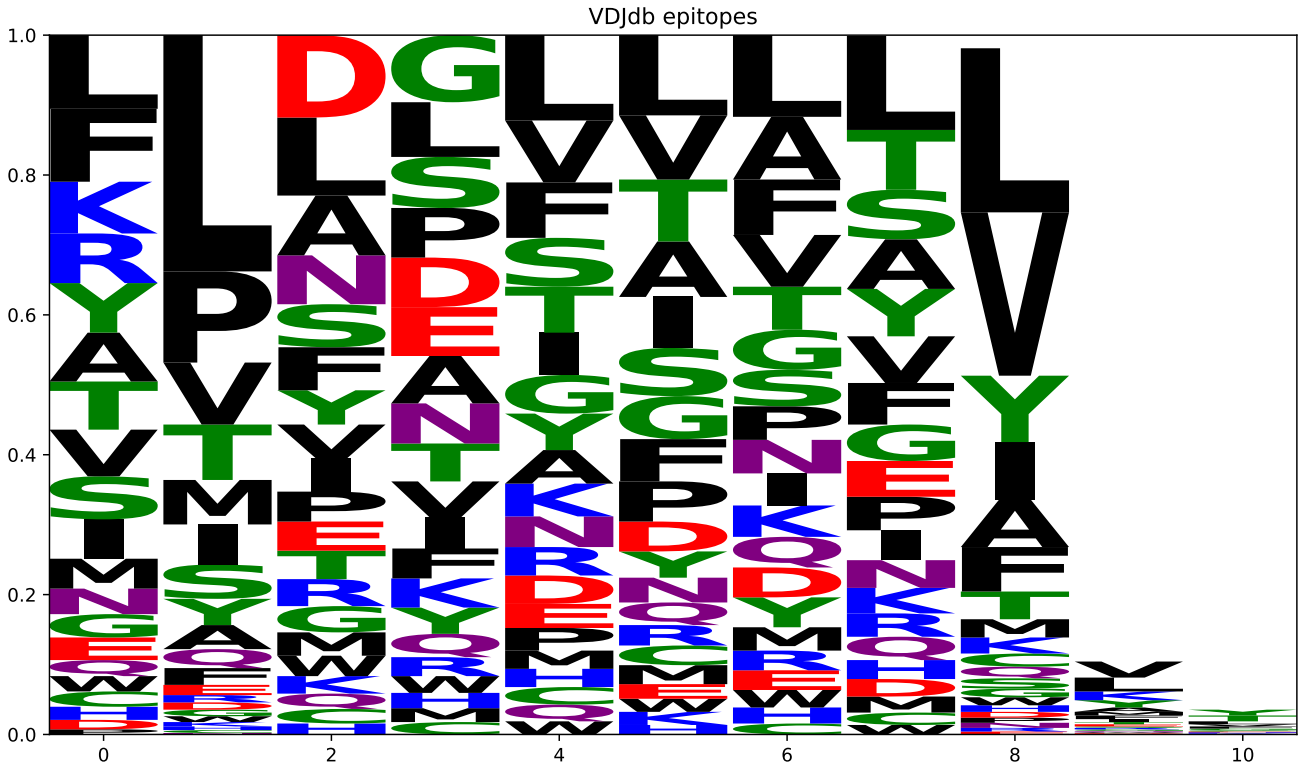
```
probs = pd.DataFrame(np.zeros_like(0,shape = (20,20)), index = list("ACDEFGHIKLMNPQRSTVWY"))
for i in epitopes:
    amk = list(i)
    for j,a in enumerate(amk):
        probs.loc[a,j] += 1
probs = probs.T / len(epitopes)
probs
```

	A	C	D	...	V	W	Y
0	0.069864	0.021898	0.013556	...	0.066736	0.021898	0.070907
1	0.034411	0.004171	0.010428	...	0.088634	0.008342	0.037539
2	0.085506	0.020855	0.117831	...	0.047967	0.029197	0.049009
3	0.067779	0.017727	0.070907	...	0.051095	0.023983	0.037539
4	0.047967	0.025026	0.040667	...	0.088634	0.018770	0.052138
5	0.079249	0.028154	0.042753	...	0.091762	0.018770	0.037539
6	0.089677	0.015641	0.042753	...	0.074035	0.025026	0.042753
7	0.070907	0.017727	0.025026	...	0.066736	0.013556	0.067779
8	0.067779	0.018770	0.008342	...	0.233577	0.011470	0.094891
9	0.006257	0.001043	0.001043	...	0.022941	0.001043	0.010428
10	0.001043	0.001043	0.000000	...	0.003128	0.000000	0.008342

```
11  0.009385  0.000000  0.000000  ...  0.002086  0.000000  0.002086
12  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.010428
13  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000
14  0.001043  0.000000  0.000000  ...  0.000000  0.000000  0.000000
15  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000
16  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000
17  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000
18  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000
19  0.001043  0.000000  0.000000  ...  0.000000  0.000000  0.000000

[20 rows x 20 columns]
```

```
#logo = logomaker.Logo(probs,color_scheme = "chemistry")
fig,ax = plt.subplots(figsize = (10,6))
ax.set_title("VDJdb epitopes")
logo = logomaker.Logo(probs.iloc[0:11,:], color_scheme = "chemistry", ax = ax)
fig.tight_layout()
plt.show()
```



```
vdjdb_filtered["Epitope"].str.len().median()
```

9.0

Ожидается, что большинство эпитопов имеют в качестве сайта разрезания лейцин или валин, так как они более представлены в датасете. Но в целом, наблюдается более менее равномерное распределение аминокислот на каждой позиции.

Поиск референсных последовательностей

Формирования множества запросов

```
uni_gene_spec = vdjdb_filtered[["Epitope gene","Epitope species"]].drop_duplicates()
uni_gene_spec.shape
```

(172, 2)

```
queries = set()
for i in uni_gene_spec.index:
    s = f"({uni_gene_spec.loc[i, 'Epitope gene']}) AND (\"{uni_gene_spec.loc[i, 'Epitope species']}\" [Organism])"
    queries.add(s)
print(len(queries))
```

172

Проверка на то, что запросы ищут все эпитопы

```
epi_ids = []
epi_num = 0
for k, q in enumerate(queries):
    gene = re.split("\((.*?)\)",q)[1]
    organism = re.split("\"",q)[1]
    epi = list(vdjdb_filtered.loc[(vdjdb_filtered["Epitope gene"] == gene) &
                                  (vdjdb_filtered["Epitope species"] == organism), "Epitope"].unique())
    epi_num += len(epi)
    for e in epi:
        #print(e)
        epi_id = e#f"{e}_{gene}_{organism}"
        if epi_id not in epi_ids:
            epi_ids.append(epi_id)
```

```
        else:
            print(epi_id)
print(f"ids {len(epi_ids)} epi_total {len(epitopes)} epi_num {epi_num}")
```

ids 959 epi_total 959 epi_num 959

Ищет и сохраняет последовательности на NCBI Protein, если в ней найден эпитоп. Код выполняется некоторое время. Готовый файл называется vdjdb_seqs.fasta в папке data. Не запускайте этот блок без необходимости.

```
seqs = {}
tries = 10
total = len(queries)
failed_queries = {}
epi_failed = 0
printProgressBar(0,total,length = 40, suffix = f"failed: 0 success: 0")
with open("../data/vdjdb_seqs_ref_new.fasta","w") as fasta:
    for k, q in enumerate(queries):
        gene = re.split("\((.*?)\)",q)[1]
        organism = re.split("\"",q)[1]
        epi = list(vdjdb_filtered.loc[(vdjdb_filtered["Epitope gene"] == gene) &
                                      (vdjdb_filtered["Epitope species"] == organism), "Epitope"].unique())
        search_res = Entrez.read(Entrez.esearch(db="protein", retmax=100, term=q))["IdList"]
        if organism == "SARS-CoV-2":
            # количество последовательностей для ковида очень велико
            search_res = Entrez.read(Entrez.esearch(db="protein", retmax=20000, term=q))["IdList"]
        handle = Entrez.efetch(id = ",".join(search_res), db = "protein", rettype="fasta", retmode="text")
        fasta_io = StringIO(handle.read())
        for record in SeqIO.parse(fasta_io, "fasta"):
            for e in epi:
                if e in record.seq:
                    epi_id = f"{e}_{gene}_{organism}"
                    record.id = epi_id
                    seqs[epi_id] = record.seq
                    SeqIO.write(record, fasta, "fasta")
                    epi.remove(e)
            if len(epi) == 0:
                break
```

```
    if len(epi) != 0:
        failed_queries[q] = {"epi":epi,"max_epi":f"{len(epi)}"}
        epi_failed += len(epi)
    printProgressBar(k,total, suffix = f"failed: {epi_failed} success:{len(seqs.values())}",length = 40)
```

Результаты

При повторном запуске блокнота, используйте этот код

```
seqs = {}
success_epitopes = []
with open("../data/vdjdb_seqs.fasta", "r") as fasta:
    for record in SeqIO.parse(fasta, "fasta"):
        seqs[record.id] = record.seq
        success_epitopes.append(re.split("_",record.id)[0])
print(len(seqs.keys()))
```

734

```
failed_epitopes = vdjdb_filtered[~vdjdb_filtered["Epitope"].isin(success_epitopes)]
failed_epitopes.shape
```

(6108, 17)

```
failed_stat = failed_epitopes[["Epitope","Epitope gene","Epitope species"]].drop_duplicates()
failed_stat.shape
```

(225, 3)

```
failed_stat.head()
```

	Epitope	Epitope gene	Epitope species
16	FLKETGGL	Nef	HIV-1
23	FLKEMGGL	Nef	HIV-1
433	ALSKGVHFV	ORF3	SARS-CoV-2
483	CLNEYHLFL	NDC1	Homo sapiens
493	AMFWSVPTV	TKT	Homo sapiens

Большинство ненайденных эпитопов - человеческие и нового короновируса. Возможно эпитопы являются довольно редкими вариантами.

```
failed_stat["Epitope species"].value_counts()
```

```
Homo sapiens                98
SARS-CoV-2                  91
HIV-1                       11
Human gammaherpesvirus 4    11
HTLV-1                       6
Human betaherpesvirus 5      3
HCV                          2
Influenza A virus            1
Mycobacterium tuberculosis   1
Streptomyces kanamyceticus   1
Name: Epitope species, dtype: int64
```

```
all_stat = vdjdb_filtered[["Epitope","Epitope gene","Epitope species"]].drop_duplicates()
all_stat["Epitope species"].value_counts()
```

```
SARS-CoV-2                  659
Homo sapiens                157
HIV-1                       49
Human gammaherpesvirus 4    30
Human betaherpesvirus 5     29
HTLV-1                       8
Influenza A virus            7
HCV                          7
Mycobacterium tuberculosis   3
Merkel cell polyomavirus     2
Dengue virus 2               1
Streptomyces kanamyceticus   1
Human papillomavirus         1
Dengue virus 1               1
HSV2                         1
YFV                          1
Saccharomyces cerevisiae     1
HCoV-HKU1                    1
Name: Epitope species, dtype: int64
```

```
combinations = {}
ff = {}
for key, seq in seqs.items():
    e = re.split("_",key)[0]
    e_start = seq.find(e)
    #print(f"{e_start} {len(seq)}")
    #print(seq[e_start + len(e)])
    if e_start + len(e) < len(seq):
        C_end = seq[e_start + len(e)]
    else:
        ff[e] = key
        continue
    combinations[e] = e[-1] + C_end
df_comb = pd.DataFrame.from_dict(combinations, orient = "index",columns = ["Comb"])
print(df_comb.shape)
```

(727, 1)

```
df_comb.head()
```

	Comb
ARMILMTHF	FF
KIFGSLAFL	LP
VLNGTVHPV	VF
MLWGYLQYV	VG
FRCPRRFCF	FS

Часть эпитопов равна по величине референсу

```
ff
```

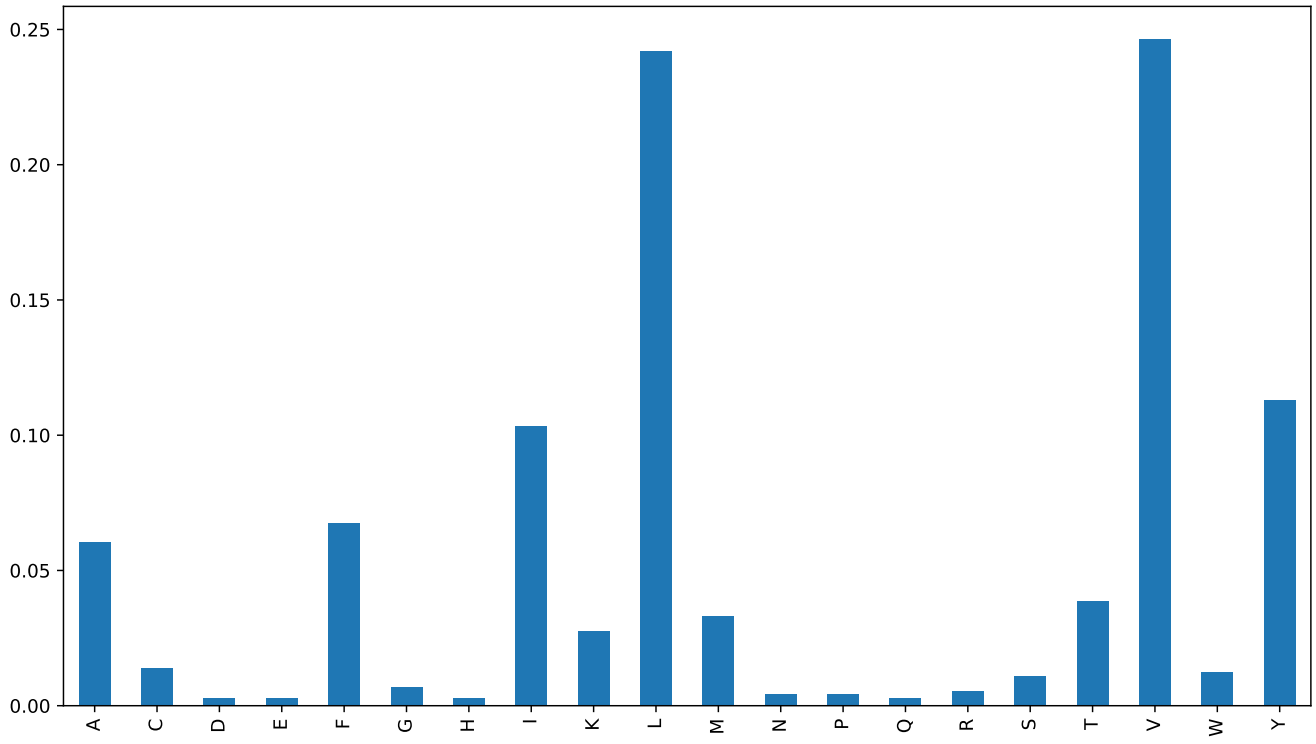
{'ILDQVPFSV': 'ILDQVPFSV_PMEL_Homo', 'IMDQVPFSV': 'IMDQVPFSV_PMEL_Homo', 'SLLMWITQV': 'SLLMWITQV_NY-E

```
success_stat = vdjdb_filtered.loc[vdjdb_filtered["Epitope"].isin(success_epitopes) & (~vdjdb_filtered["Epitope"].isin(ff.keys())
success_stat.shape
```

(727, 3)

Частоты встречаемости аминокислот в N-конце сайта разрезания

```
(success_stat["Epitope"].str[-1].value_counts().sort_index() / len(success_stat.index)).plot.bar()
```




```
df_comb.value_counts()
```

Comb

LL 20

VL 20

LV 14

VY 14

VD 13

..

FY 1

FH 1

FE 1

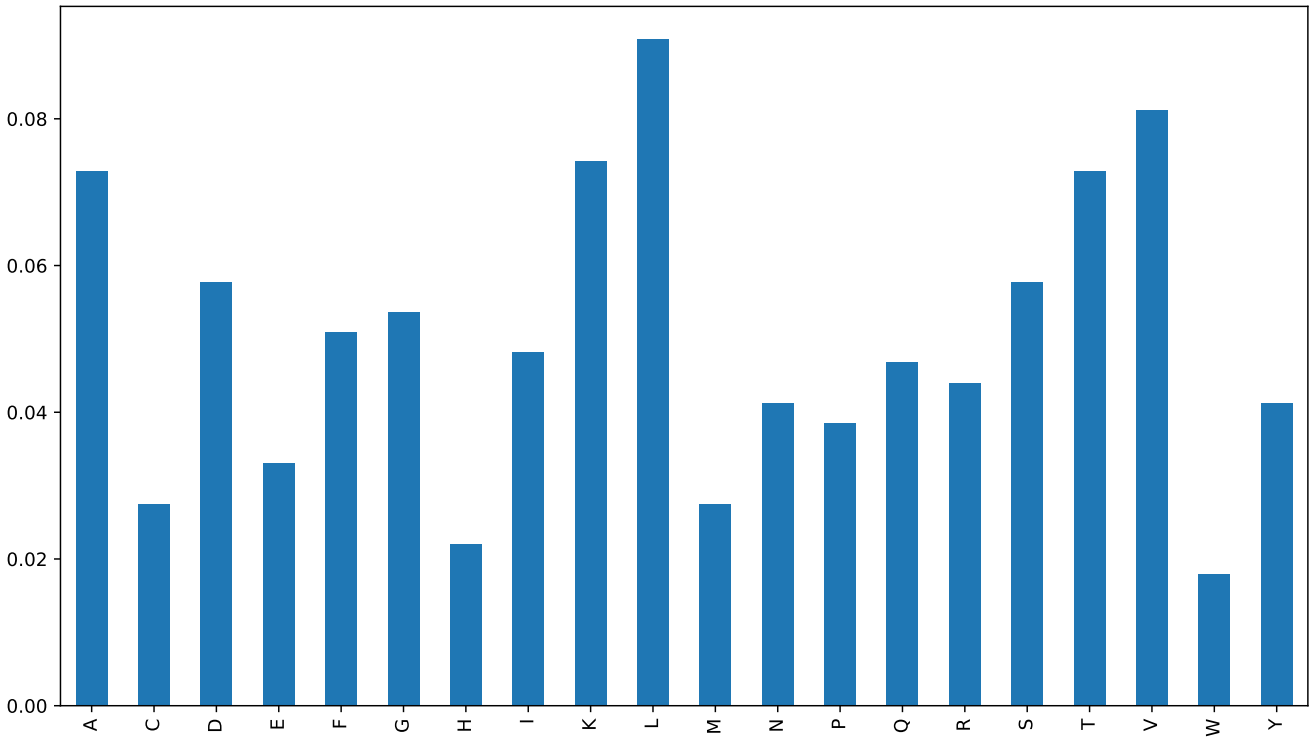
NK 1

AC 1

Length: 190, dtype: int64

Тот же график для С-конца сайта

```
(df_comb["Comb"].str[1].value_counts().sort_index() / len(df_comb.index)).plot.bar()
```



Можно сделать вывод, что не так важна аминокислота, идущая после сайта связывания, как перед ним.