For samples:

First transform vcf to bed files and lift from mm9 to mm10 to mm39. Then VEP annotation.

Start with: Exome-seq\_Sample\_13\_S15\_snvs\_filtered.vcf

vcf file Exome-seq\_Sample\_13\_S15\_snvs\_filtered.vcf

**vcf2bed < Exome-seq\_Sample\_13\_S15\_snvs\_filtered.vcf > Exome-seq\_Sample\_13\_S15\_snvs\_filtered.bed**

you get that: Exome-seq\_Sample\_13\_S15\_snvs\_filtered.bed

then run correct\_bed.py after you install pandas: pip install pandas

correct\_bed.py:

import pandas as pd

# Load the original BED file

original\_bed = pd.read\_csv('/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered.bed', sep='\t', header=None)

# Create a simplified BED file with only the necessary columns for liftover

simplified\_bed = original\_bed.iloc[:, [0, 1, 2, 3]]

simplified\_bed.columns = ['CHROM', 'START', 'END', 'ID']

# Save the simplified BED file

simplified\_bed.to\_csv('/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_simplified.bed', sep='\t', index=False, header=False)

you get this: Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_simplified.bed

**then lift the bed from mm9 to mm10 and mm39 in this website:** [**https://genome.ucsc.edu/cgi-bin/hgLiftOver?hgsid=2313004406\_Ql8wcYE3eOhUTakhHIeQ2CaHAtgr**](https://genome.ucsc.edu/cgi-bin/hgLiftOver?hgsid=2313004406_Ql8wcYE3eOhUTakhHIeQ2CaHAtgr)

then you get: **Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_simplified.bed**

and now you need to get all the extra information from the Exome-seq\_Sample\_13\_S15\_snvs\_filtered.bed and merge them with the new one. Merge.py:

you get: Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_with\_info.bed

then convert to vcf. bed\_to\_vcf1.py

import pandas as pd

# Paths to the BED files

original\_bed\_path = "/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered.bed"

lifted\_simplified\_bed\_path = "/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_simplified.bed"

output\_merged\_bed\_path = "/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_with\_info.bed"

# Load the original BED file

original\_bed = pd.read\_csv(original\_bed\_path, sep='\t', header=None)

original\_bed.columns = ['CHROM', 'START', 'END', 'ID', 'REF', 'ALT', 'QUAL', 'FILTER', 'INFO', 'FORMAT', 'NORMAL', 'TUMOR']

# Load the lifted simplified BED file

lifted\_bed = pd.read\_csv(lifted\_simplified\_bed\_path, sep='\t', header=None)

lifted\_bed.columns = ['CHROM', 'START', 'END', 'ID']

# Adjust positions: BED files are 0-based while VCF files are 1-based, so adjust the positions accordingly

lifted\_bed['START'] = lifted\_bed['START'] + 1

# Ensure both dataframes have the same length

if len(original\_bed) != len(lifted\_bed):

    raise ValueError("The original and lifted BED files do not have the same number of entries.")

# Merge the coordinates from the lifted BED into the original BED

merged\_bed = original\_bed.copy()

merged\_bed['CHROM'] = lifted\_bed['CHROM']

merged\_bed['START'] = lifted\_bed['START']

merged\_bed['END'] = lifted\_bed['END']

# Save the merged BED file with all the variant information

merged\_bed.to\_csv(output\_merged\_bed\_path, sep='\t', index=False, header=False)

print("Merged BED file has been created.")

import pandas as pd

# Load the merged BED file

merged\_bed\_path = "/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_with\_info.bed"

output\_vcf\_path = "/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted.vcf"

merged\_bed = pd.read\_csv(merged\_bed\_path, sep='\t', header=None)

merged\_bed.columns = ['CHROM', 'POS', 'END', 'ID', 'REF', 'ALT', 'QUAL', 'FILTER', 'INFO', 'FORMAT', 'NORMAL', 'TUMOR']

# Create a new DataFrame for the VCF

vcf\_df = merged\_bed[['CHROM', 'POS', 'ID', 'REF', 'ALT', 'QUAL', 'FILTER', 'INFO', 'FORMAT', 'NORMAL', 'TUMOR']]

# Load the original VCF header

original\_vcf\_path = "/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered.vcf"

with open(original\_vcf\_path, 'r') as file:

    vcf\_header = []

    for line in file:

        if line.startswith("#"):

            vcf\_header.append(line.strip())

        else:

            break

# Save the VCF file

with open(output\_vcf\_path, 'w') as file:

    # Write the VCF header

    for line in vcf\_header:

        file.write(line + '\n')

    # Write the VCF data

    vcf\_df.to\_csv(file, sep='\t', index=False, header=False)

print("VCF file has been created.")

then you get: Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted.vcf

then you run vep with frameshift, downstream and wildtype plugin:

**perl /mnt/c/Users/agsko/dev/pcm/ensembl-vep/vep -i /mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted.vcf -o /mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_annotated.vep.vcf --vcf --symbol --terms SO --plugin Wildtype --plugin Downstream --plugin Frameshift --tsl --dir\_plugins /mnt/c/Users/agsko/dev/pcm/ensembl-vep/Plugins --dir\_cache /mnt/c/Users/agsko/dev/pcm/ensembl-vep/cache --fasta /mnt/c/Users/agsko/dev/pcm/vep/fasta/Mus\_musculus.GRCm39.dna.primary\_assembly.fa --species mus\_musculus --cache –offline**

then install pvacseq – pip install pvactools, and pvactools –version

then install vatools: pip install vatools

then add a dummy sample using vcf-genotype-annotator: python genotype.py

import pandas as pd

# Paths to your files

vcf\_file = '/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_annotated.vep.vcf'

output\_vcf\_file = '/mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_annotated\_corrected.vep.vcf'

# Open and process the VCF file

with open(vcf\_file, 'r') as infile, open(output\_vcf\_file, 'w') as outfile:

    for line in infile:

        if line.startswith('#'):

            outfile.write(line)

        else:

            fields = line.strip().split('\t')

            if not fields[5].replace('.', '', 1).isdigit():

                fields[5] = '.'

            outfile.write('\t'.join(fields) + '\n')

print("VCF file has been corrected and saved.")

command: **vcf-genotype-annotator /mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_annotated\_corrected.vep.vcf DUMMY 0/1 -o /mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_annotated\_with\_dummy.vep.vcf**

then generate protein fasta using pvacseq:

**pvacseq generate\_protein\_fasta /mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_annotated\_with\_dummy.vep.vcf 17 /mnt/c/Users/agsko/dev/pcm/Sample\_13\_S15\_snvs\_output\_class1\_mutant\_peptides.txt --sample-name DUMMY**

**pvacseq generate\_protein\_fasta /mnt/c/Users/agsko/dev/pcm/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_annotated\_with\_dummy.vep.vcf 29 /mnt/c/Users/agsko/dev/pcm/Sample\_13\_S15\_snvs\_output\_class2\_mutant\_peptides.txt --sample-name DUMMY**

NOW WE HAVE NO PEPTIDES IN S12:

so I reviewed the vcf searching for IMPACT: HIGH, and the only variants with this impact are these: splice\_acceptor\_variant|HIGH, splice\_acceptor\_variant&NMD\_transcript\_variant|HIGH. would you expect those to give any peptides?

Splice acceptor variants, including those with "HIGH" impact such as splice\_acceptor\_variant and splice\_acceptor\_variant&NMD\_transcript\_variant, typically affect splicing of mRNA. These types of variants can lead to exon skipping or the inclusion of intronic sequences in the mRNA, potentially resulting in truncated or altered proteins. However, predicting specific peptide sequences from these variants can be challenging, as the exact changes to the protein sequence are dependent on the resulting mRNA splicing pattern. They may not always produce peptides suitable for neoantigen prediction tools like pVACseq, which typically expect missense or frameshift variants to generate predictable peptides.

To check if these variants result in meaningful peptides, you can explore further annotations or use specialized tools to assess their potential impact on protein sequences. You might also consider including variants with "MODERATE" impact, such as missense mutations, which are more likely to generate neoantigenic peptides.

MHC Alleles (documents about MHC haplotypes on PCM dissertation)

Based on the provided documents and your observations, it seems that the following MHC alleles are relevant for the A/J mouse strain:

**MHC Alleles for A/J Mouse Strain**

1. **Class I MHC Alleles**:
   * H2-Kk (or H-2Kk)
   * H2-Dd (or H-2Dd)
   * H2-Ld (or H-2Ld)
2. **Class II MHC Alleles**:
   * H2-IAk (or H-2IAk)
   * H2-IEk (or H-2IEk)

Let's refine the script to match each peptide to these specific MHC alleles.

MHC.py

import pandas as pd

# Define the MHC alleles for the A/J strain

class1\_alleles = ['H2-Kk', 'H2-Dd', 'H2-Ld']

class2\_alleles = ['H2-IAk', 'H2-IEk']

# Define the file paths for the peptides

class1\_peptides\_file = r"/mnt/c/Users/agsko/dev/pcm/samples\_peptides/Sample\_1\_S3/Sample\_1\_S3\_indels\_output\_class1\_mutant\_peptides.txt"

class2\_peptides\_file = r"/mnt/c/Users/agsko/dev/pcm/samples\_peptides/Sample\_1\_S3/Sample\_1\_S3\_indels\_output\_class2\_mutant\_peptides.txt"

# Load the peptides from the specified files

class1\_peptides = pd.read\_csv(class1\_peptides\_file, header=None, names=['Peptide'])

class2\_peptides = pd.read\_csv(class2\_peptides\_file, header=None, names=['Peptide'])

# Function to filter out header lines and match peptides to MHC alleles

def match\_peptides\_to\_hla(peptides, hla\_alleles):

    matched = []

    for peptide in peptides['Peptide']:

        if not peptide.startswith('>'):  # Filter out header lines

            for hla in hla\_alleles:

                matched.append({'HLA': hla, 'Peptide': peptide})

    return pd.DataFrame(matched)

# Match Class I peptides to Class I alleles

matched\_class1 = match\_peptides\_to\_hla(class1\_peptides, class1\_alleles)

# Match Class II peptides to Class II alleles

matched\_class2 = match\_peptides\_to\_hla(class2\_peptides, class2\_alleles)

# Combine Class I and Class II matches

combined\_matched = pd.concat([matched\_class1, matched\_class2])

# Save to a file in the format required by DeepNeo

output\_file = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/deepneo\_input.txt'

combined\_matched.to\_csv(output\_file, sep='\t', index=False, header=False)

print(f"DeepNeo input file saved as '{output\_file}'")

before you run the DeepNeo prediction you have to have a text with all potential 9mer peptides for class 1, run mer.py

import pandas as pd

# Define the MHC alleles for the A/J strain

class1\_alleles = ['H2-Kk', 'H2-Dd', 'H2-Ld']

# Define the file path for the long peptides

class1\_peptides\_file =  r"/mnt/c/Users/agsko/dev/pcm/samples\_peptides/Sample\_1\_S3/Sample\_1\_S3\_indels\_output\_class1\_mutant\_peptides.txt"

# Load the long peptides from the specified file

class1\_peptides = pd.read\_csv(class1\_peptides\_file, header=None, names=['Peptide'])

# Function to generate 9-mers from a longer peptide

def generate\_nine\_mers(peptides):

    nine\_mers = []

    for peptide in peptides['Peptide']:

        if not peptide.startswith('>'):  # Filter out header lines

            peptide\_length = len(peptide)

            for i in range(peptide\_length - 8):  # Slide window to generate 9-mers

                nine\_mer = peptide[i:i+9]

                nine\_mers.append(nine\_mer)

    return pd.DataFrame(nine\_mers, columns=['Peptide'])

# Generate 9-mers from the long peptides

nine\_mer\_peptides = generate\_nine\_mers(class1\_peptides)

# Function to match peptides to MHC alleles

def match\_peptides\_to\_hla(peptides, hla\_alleles):

    matched = []

    for peptide in peptides['Peptide']:

        for hla in hla\_alleles:

            matched.append({'HLA': hla, 'Peptide': peptide})

    return pd.DataFrame(matched)

# Match 9-mer peptides to Class I alleles

matched\_class1\_nine\_mers = match\_peptides\_to\_hla(nine\_mer\_peptides, class1\_alleles)

# Function to split DataFrame into multiple files if it has more than 1000 peptides

def split\_and\_save\_peptides(peptides, base\_filename, max\_peptides\_per\_file=1000):

total\_peptides = len(peptides)

num\_files = (total\_peptides // max\_peptides\_per\_file) + (1 if total\_peptides % max\_peptides\_per\_file > 0 else 0)

for i in range(num\_files):

start\_index = i \* max\_peptides\_per\_file

end\_index = start\_index + max\_peptides\_per\_file

subset = peptides[start\_index:end\_index]

output\_file = f"{base\_filename}\_part{i+1}.txt"

subset.to\_csv(output\_file, sep='\t', index=False, header=False)

# Save to multiple files if necessary

output\_base\_filename = '/mnt/c/Users/agsko/dev/pcm/samples\_peptides/deepneo\_input\_class1\_9mers\_S4'

split\_and\_save\_peptides(matched\_class1\_nine\_mers, output\_base\_filename)

# Output base file path

output\_base\_filename

colate.py

import pandas as pd

# Load the three TSV files

file1 = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/DeepNEO\_pred\_S3.tsv'

file2 = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/DeepNEO\_S3\_con.tsv'

file3 = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/DeepNEO\_S3\_con2.tsv'

# Read the TSV files into dataframes

df1 = pd.read\_csv(file1, sep='\t')

df2 = pd.read\_csv(file2, sep='\t')

df3 = pd.read\_csv(file3, sep='\t')

# Concatenate the dataframes

combined\_df = pd.concat([df1, df2, df3], ignore\_index=True)

# Save the combined dataframe to a new TSV file

output\_file\_combined = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/DeepNEO\_combined\_S3.tsv'

combined\_df.to\_csv(output\_file\_combined, sep='\t', index=False)

print(f"Combined file saved as '{output\_file\_combined}'")

then for class II, for all potential 15mer peptides

import pandas as pd

# Define the MHC alleles for the A/J strain specific to Class II

class2\_alleles = ['H2-IAk', 'H2-IEk']

# Define the file path for the long peptides

class2\_peptides\_file = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/Sample\_1\_S3/Sample\_1\_S3\_indels\_output\_class2\_mutant\_peptides.txt'

# Load the long peptides from the specified file

class2\_peptides = pd.read\_csv(class2\_peptides\_file, header=None, names=['Peptide'])

# Function to generate 15-mers from a longer peptide

def generate\_fifteen\_mers(peptides):

    fifteen\_mers = []

    for peptide in peptides['Peptide']:

        if not peptide.startswith('>'):  # Filter out header lines

            peptide\_length = len(peptide)

            for i in range(peptide\_length - 14):  # Slide window to generate 15-mers

                fifteen\_mer = peptide[i:i+15]

                fifteen\_mers.append(fifteen\_mer)

    return pd.DataFrame(fifteen\_mers, columns=['Peptide'])

# Generate 15-mers from the long peptides

fifteen\_mer\_peptides = generate\_fifteen\_mers(class2\_peptides)

# Function to match peptides to MHC alleles

def match\_peptides\_to\_hla(peptides, hla\_alleles):

    matched = []

    for peptide in peptides['Peptide']:

        for hla in hla\_alleles:

            matched.append({'HLA': hla, 'Peptide': peptide})

    return pd.DataFrame(matched)

# Match 15-mer peptides to Class II alleles

matched\_class2\_fifteen\_mers = match\_peptides\_to\_hla(fifteen\_mer\_peptides, class2\_alleles)

# Save to a file in the format required by DeepNeo

output\_file\_class2\_fifteen\_mers = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/deepneo\_input\_class2\_15mers.txt'

matched\_class2\_fifteen\_mers.to\_csv(output\_file\_class2\_fifteen\_mers, sep='\t', index=False, header=False)

# Output file path

output\_file\_class2\_fifteen\_mers

then run colate.py and then filter.py

**Human relevant peptides:**

Yes, there are several databases for human-relevant neoantigens that can be useful for research and clinical applications. Here are some of the notable ones:

**1. Cancer Antigenic Peptide Database (CAPD)**

The Cancer Antigenic Peptide Database provides information on antigenic peptides that have been identified in various cancers. It includes data on neoantigens, tumor-associated antigens, and their associated peptides.

* **URL**: CAPD

**2. IEDB - Immune Epitope Database**

The Immune Epitope Database (IEDB) contains a large collection of experimentally characterized immune epitopes from infectious diseases, allergens, autoimmune diseases, and transplantation.

* **URL**: [IEDB](https://www.iedb.org/)

**3. CancerPeptide Database**

CancerPeptide Database is a comprehensive resource for experimentally validated tumor antigen peptides. It includes information on MHC binding, T-cell assays, and cancer-specific peptides.

* **URL**: CancerPeptide Database

**4. TSNAdb - Tumor-Specific NeoAntigen Database**

TSNAdb is a database dedicated to tumor-specific neoantigens. It contains information on neoantigens identified from tumor samples, including their MHC binding affinity and potential immunogenicity.

* **URL**: TSNAdb

**5. NEOANTIGEN:**

This database provides information on neoantigens identified from various tumor samples, focusing on their relevance to personalized cancer immunotherapy.

* **URL**: NEOANTIGEN

**6. pVACtools**

Although not a database per se, pVACtools is a comprehensive computational pipeline for the identification of neoantigens from sequencing data. It can be used in conjunction with various databases for neoantigen prediction and validation.

* **URL**: [pVACtools](https://pvactools.readthedocs.io/en/latest/)

**7. dbPepNeo**

dbPepNeo is a dedicated database for neoantigen peptides. It provides comprehensive information on the sequences, MHC-binding properties, and immunogenicity of neoantigens.

* **URL**: dbPepNeo

**8. Tumor Neoantigen Selection Alliance (TESLA)**

TESLA provides resources and guidelines for the identification and validation of neoantigens in cancer. While it’s more of a collaborative effort and guideline than a database, it offers valuable information and tools for neoantigen research.

LENS: can’t use because I need DNA normal and RNA seq fastq data – NOT SUITABLE

Let’s see the prediction for HLA-A\*02:01 – using only 9mers from class 1 peptides.

import pandas as pd

# Define the human HLA allele for affinity checking

hla\_alleles = ["HLA-A\*02:01"]

# Define the file paths for the peptides

class1\_peptides\_file = r"/mnt/c/Users/agsko/dev/pcm/samples\_peptides/Sample\_1\_S3/Sample\_1\_S3\_indels\_output\_class1\_mutant\_peptides.txt"

# Load the peptides from the specified file

class1\_peptides = pd.read\_csv(class1\_peptides\_file, header=None, names=['Peptide'])

# Function to filter out header lines and match peptides to HLA alleles

def match\_peptides\_to\_hla(peptides, hla\_alleles):

    matched = []

    for peptide in peptides['Peptide']:

        if not peptide.startswith('>'):  # Filter out header lines

            for hla in hla\_alleles:

                matched.append({'HLA': hla, 'Peptide': peptide})

    return pd.DataFrame(matched)

# Match Class I peptides to the HLA allele

matched\_class1 = match\_peptides\_to\_hla(class1\_peptides, hla\_alleles)

# Save to a file in the format required by DeepNeo

output\_file = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/deepneo\_HLA\_input.txt'

matched\_class1.to\_csv(output\_file, sep='\t', index=False, header=False)

print(f"DeepNeo input file saved as '{output\_file}'")

# Function to generate 9-mers from a longer peptide

def generate\_nine\_mers(peptides):

    nine\_mers = []

    for peptide in peptides['Peptide']:

        if not peptide.startswith('>'):  # Filter out header lines

            peptide\_length = len(peptide)

            for i in range(peptide\_length - 8):  # Slide window to generate 9-mers

                nine\_mer = peptide[i:i+9]

                nine\_mers.append(nine\_mer)

    return pd.DataFrame(nine\_mers, columns=['Peptide'])

# Generate 9-mers from the long peptides

nine\_mer\_peptides = generate\_nine\_mers(class1\_peptides)

# Function to match peptides to MHC alleles

def match\_peptides\_to\_hla(peptides, hla\_alleles):

    matched = []

    for peptide in peptides['Peptide']:

        for hla in hla\_alleles:

            matched.append({'HLA': hla, 'Peptide': peptide})

    return pd.DataFrame(matched)

# Match 9-mer peptides to Class I alleles

matched\_HLA\_class1\_nine\_mers = match\_peptides\_to\_hla(nine\_mer\_peptides, hla\_alleles)

# Save to a file in the format required by DeepNeo

output\_file\_HLA\_class1\_nine\_mers = '/mnt/c/Users/agsko/dev/pcm/samples\_peptides/deepneo\_input\_HLA\_class1\_9mers.txt'

matched\_HLA\_class1\_nine\_mers.to\_csv(output\_file\_HLA\_class1\_nine\_mers, sep='\t', index=False, header=False)

# Output file path

output\_file\_HLA\_class1\_nine\_mers

import pandas as pd

# Load the combined TSV file

combined\_file = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/HLA\_S3.tsv'

combined\_df = pd.read\_csv(combined\_file, sep='\t')

# Print column names to confirm

print(combined\_df.columns)

# Define thresholds

score\_threshold = 0.5  # Threshold for both MHC binding score and TCR reactivity score

# Apply filters using the correct column names

filtered\_df = combined\_df[(combined\_df['MHC binding'] > score\_threshold) &

                          (combined\_df['TCR reactivity'] > score\_threshold)]

# Save the filtered dataframe to a new TSV file

filtered\_HLA\_file = r'/mnt/c/Users/agsko/dev/pcm/samples\_peptides/filtered\_HLA\_S3.tsv'

filtered\_df.to\_csv(filtered\_HLA\_file, sep='\t', index=False)

print(f"Filtered file saved as '{filtered\_HLA\_file}'")

if we are to run pvacseq we follow this command:

pvacseq run --iedb-install-directory /mnt/c/Users/agsko/dev/pcm/ \ /mnt/c/Users/agsko/dev/pcm/samples\_peptides/Sample\_13\_S15/Exome-seq\_Sample\_13\_S15\_snvs\_filtered\_lifted\_annotated\_with\_dummy.vep.vcf \ TUMOR \ HLA-A\*02:01 \ NetMHCpan \ /mnt/c/Users/agsko/dev/pcm/samples\_peptides/Sample\_13\_S15/pvacseq\_output

We can also run NetMHCpan directly from the website IEDB:

http://tools.iedb.org/mhci/

**Citations:**

If you use these predictions in a manuscript, please include the following in the method section:  
The MHCI binding predictions were made on 7/22/2024 using the IEDB analysis resource NetMHCpan (ver. 4.1) tool [1].

1. Birkir Reynisson, Bruno Alvarez, Sinu Paul, Bjoern Peters, Morten Nielsen. 2020. NetMHCpan-4.1 and NetMHCIIpan-4.0: improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data. Nucleic Acids Res. 48(W1):W449-W454. doi: 10.1093/nar/gkaa379.

Restrict for HLA-A\*02:01

Start with S3:

Use FASTA file: Sample\_1\_S3\_indels\_output\_class1\_mutant\_peptides.txt