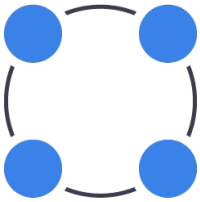

Detecting mutation in influenza virus A H3N2 caused an infection in vaccinated subject

Kazovskaia Anastasiia* @Chickypicky

Filippov Mikhail† @pseudocephalus

Abstract

In the presented case, sample from a vaccinated subject diagnosed with flu was taken, and hemagglutinin inhibition test showd that the isolated virus is an influenza A H3N2 subtype, which is covered by taken vaccine. For the purpose of investigating the cause of vaccine resistancy, deep sequencing of hemagglutinin (HA) gene was performed. Obtained data was analyzed with common bioinformatics tools to reveal rare variants in researched sample. As a result of the study, one missense mutation corresponding to the antibody recognition site of HA was reported.

Introduction

Quasispecies

Flu is an epidemic seasonal viral disease caused by RNA influenza viruses of family *Orthomyxoviridae*. There are many subtypes of influenza viruses including influenza viruses A, B, C and D. As there is no effective treatment for flu, vaccination is considered as one of the best options to manage the disease. However, effectiveness of vaccination is severely ceased by a remarkably high mutation rate in RNA viruses [Drake et al.; 1999]. Annual vaccine is developed by the prediction of next season's most expected variants of virus, but sometimes such prediction appears not to be correct. Moreover, there is a possibility that significant mutations will occur in a particular viral population of the subtype covered by the vaccine, which will allow it to overcome the immune barrier developed with vaccine intake. Such highly

*Saint Petersburg State University, Saint Petersburg
†Herzen University, Saint Petersburg

mutated populations of the species with relatively small genome, like RNA viruses, are called quasispecies [Domingo et al; 2021].

Vaccination in a nutshell

Vaccine itself is a weakened pathogenic organism or some part of it (protein, DNA, RNA or other), that, when injected into human tissues, can cause immune response and developing of immune memory, which allow vaccinated human to be prepared to the contact with real pathogen. Foreign substance recognised by immune system and causing antibody production by immunocompetent cells is called antigen (stands for ANTIbody GENerator). In case of flu infection, antigen presented in vaccine is a hemagglutinin - glycoprotein from the surface of virus that allows it to bind to the human cell and get into it.

Pathogen can only be neutralized by immunocytes when specific to that pathogen antibodies are presented in blood. If antibodies produced after vaccination can bind to real pathogen, vaccinated subject is under immune defense, but if pathogenic protein mutates, there is a chance that antibodies from vaccination can no longer bind to it, and before immune system will develop new antibodies, subject is defenseless as if he was not vaccinated at all.

Deep sequencing

There are several methods for detecting mutated viruses in sample, one of which is a deep sequencing, which refers sequencing genomic region multiple times. To detect extremely rare genetic variants in researched population (below 1% of all variants), we need to read sequence of interest many hundreds and thousand times. Such rare variants can belong to small amounts of cancer cells in tissue sample, particular (for example, resistant to antibiotics) bacterias in culture or, as in our case, small populations of viral quasispecies, which, however, can outbreak infection. This method is commonly used in oncology, microbiology, virology and other science fields to get a comprehensive picture of all mutations presented in sample [Goldman, Domschke; 2014].

When deep sequencing is performed, researcher faces the problem of differentiating real variants from sequencing and amplification errors. Errors can occur in both amplification of the fragment and sequencing steps, and in some cases frequency of errors approaches frequency of rare variants. That problem can be resolved by control sequencing, when sequence of the frag-

ment of interest (same as researched fragment) is already known. When we sequence control fragment with comparable to experiment's coverage, all of the appearing variants are errors, and we can estimate error rate. All of the variants in experiment with frequency below the one obtained from control run, are errors as well.

Methods

Data preparation

Hemagglutinin inhibition test was performed to find out which virus had caused the infection and then Illumina sequencing was performed. For control purpose, HA gene from isogenic H3N2 influenza virus was PCR amplified, subcloned into a plasmid and multiplied in bacterial culture. Cloned control HA was sequenced with the same Illumina machine. Sequencing data of the hemagglutinin gene of A/Hong Kong/4801/2014 (H3N2) influenza virus was obtained from SRA database [URL 1]. Control sequencing data is also presented in this database [URL 2, URL 3, URL 4]. Reference genome of the influenza A H3N2 virus is available in nuccore database [URL 5].

Alignment and piling up

Reads from subject were indexed and aligned to the reference using BWA 0.7.17-r1188 (BWA-mem) [Li; 2013]. Aligned reads were sorted, indexed and piled up using Samtools 1.13 [Li; 2009]. Depth flag for *samtools mpileup* is a number more than estimated coverage (20011). In our case it is 21000. Reads from control sequencing have estimated average coverage of 13775, and *samtools mpileup* was run with *-d* of 14000.

Variant calling

Variant analysis was performed with Varscan 2.4.4 with at least 0.1% variant support. [Koboldt et al; 2012]. Detected variants were manually inspected in Integrative Genomics Viewer [James et al; 2011].

Appendix

Exact average coverage for our sequencing was derived from .mpileup file created with depth equal to the total number of reads (-d 358265). Python 3 scripts for calculating the exact average coverage and for estimating error

rates in control sequencing data are presented in *Scripts* section. 3D models of HA protein was obtained from RCSB database [URL 6, URL 7] and visualized via PyMOL 2.0.

Results

Reference genome has length of 1665 bp (base pairs), and sequencing has 358265 reads with average length of 93 bp. Estimated alignment coverage is 20011. 361116 reads were aligned with BWA-MEM, which is 99.94% of total reads count. Estimated coverage was used as *depth* argument in *samtools mpileup*, and running VarScan resulted in 16 detected SNPs in viral genome. All there variants presented in table 3 of supplementary material. There are 5 SNPs with above 90% frequency and 11 SNPs with frequency below 1%. VarScan results for three control sequences are presented in Table 4, Table 5 and Table 6. Estimated error rate of different types of errors is shown in Table 1. Maximal average error rate is $0.2727\% \pm 0.0616$

Table 1: **Statistics of control samples frequencies.**

Number of SNP position occurrence[1]	**Mean, %**	**Standard deviation, %**	**Percentage of frequencies inside** $[\text{mean} - \text{std}; \text{mean} + \text{std};]$
1	0.2727	0.0616	73.3333%
2	0.2833	0.1061	90.4762%
3	0.2784	0.0492	70.5882%

[1] Count of SNPs in a certain position in different control samples.

In VarScan results there are 2 variants with frequency above 3 SD's (standard deviation) of average frequency of error (0.4575%), and such variants are presented in Table 2.

Table 2: **Significant SNPs.**

Position	Reference	Alternative	Frequency	Residue change
307	C	T	0.92%	P103S
1458	T	C	0.86%	Y484Y

Discussion

Variants significance

Both 5 variants with above 90% frequency and one of the significant rare variants in position 1458 appeared to be silent, but the other one in position 307 is a missense mutation, leading to replacing 103th Proline residue to Serine (P103S). 103th residue is considered to be the part of epitope D in influenza viruse's hemagglutinin protein [Munoz et al.; 2004]. That means that mutation in this position can prevent antibody from binding to the HA.

Mutation visualization

On the Figure 1A mutated residue is highlighted red on the 3D model of HA protein. 3D structure of antibodies binded to the HA is presented on the Figure 1B.

3D structure of antibodies binded to the HA protein shows that the position when mutation occured is not in touch with Fab fragment of antibody, but presumably changes whole epitope's structure in such a way that antibody can't bind to it anymore. Exact mechanism of how the mutation prevent antibody binding is an object for further research.

Estimating error rate

There are two kinds of error in sequencing procedure - amplification errors and errors of the sequencing per se. Errors of amplification are represented in primary structure of sequenced fragment itself, and as we run 3 control sequences, they must be presented in each of them. Errors of amplification are expected to not to have same position in each of 3 runs and occur in unique positions. Thus, to estimate error rate for amplification errors, we

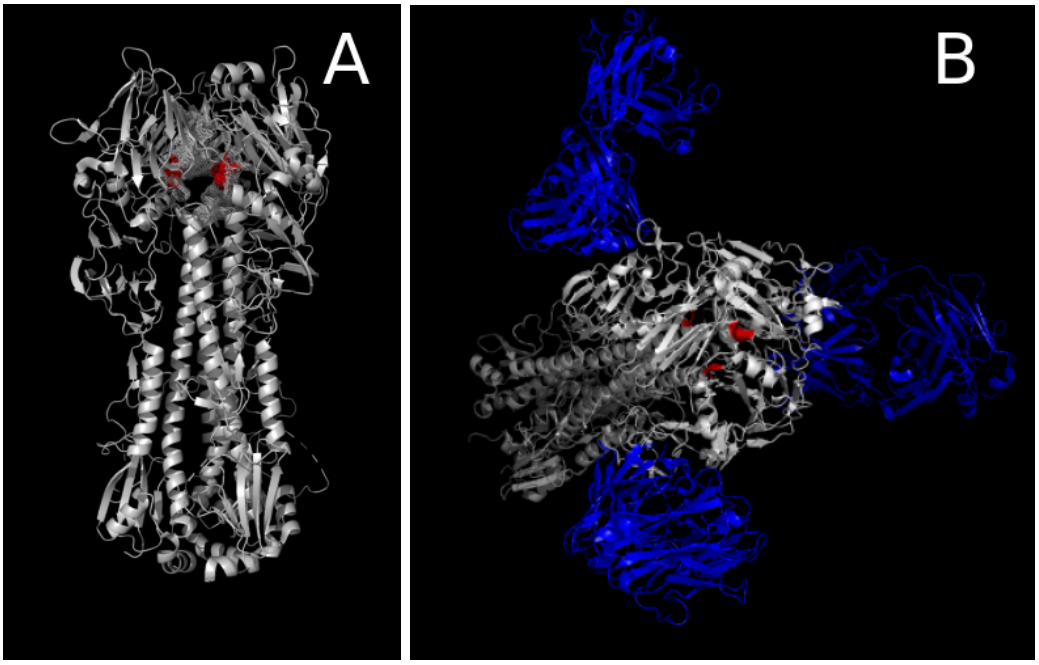

Figure 1: A: Influenza A H3N2 hemagglutinin with highlighted P103 residue (red) B: with binded antibodies (blue)

need to calculate the average frequency of all variants which occurred in the same position in each of 3 control runs, and to estimate error rate for sequencing errors we need to do the same for variants that have unique position. However, there are variants which has occurred in the same position in 2 of 3 runs, and it is unclear which type of mistake they belong to.

References

Drake JW, Holland JJ (November 1999). «Mutation rates among RNA viruse». Proceedings of the National Academy of Sciences of the United States of America. 96 (24): 13910–3. Bibcode:1999PNAS...9613910D. PMC 24164. PMID 10570172. https://doi.org/10.1073/pnas.96.24.13910

Domingo, Esteban; García-Crespo, Carlos; Perales, Celia (29 September 2021). «Historical Perspective on the Discovery of the Quasispecies Concept». Annual Review of Virology. 8 (1): 51–72. ISSN 2327-056X. PMID 34586874. https://doi.org/10.1146/annurev-virology-091919-105900

Goldman D, Domschke K. Making sense of deep sequencing. Int J Neuropsychopharmacol. 2014 Oct;17(10):1717-25. Epub 2014 Jun 13. PMID:

24925306; PMCID: PMC5895101.
https://doi.org/10.1017/S1461145714000789

James T. Robinson, Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S. Lander, Gad Getz, Jill P. Mesirov. Integrative Genomics Viewer. Nature Biotechnology 29, 24–26 (2011) https://doi.org/10.1038/nbt.1754

Koboldt, D., Zhang, Q., Larson, D., Shen, D., McLellan, M., Lin, L., Miller, C., Mardis, E., Ding, L., & Wilson, R. (2012). VarScan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing Genome Research DOI: https://doi.org/10.1101/gr.129684.111. URL: http://varscan.sourceforge.net

Li H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997v1 [q-bio.GN] https://doi.org/10.48550/arXiv.1303.3997

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, and 1000 Genome Project Data Processing Subgroup, The Sequence alignment/map (SAM) format and SAMtools, Bioinformatics (2009) 25(16) 2078-9 [19505943] https://doi.org/10.1093/bioinformatics/btp352

Muñoz ET, Deem MW. Epitope analysis for influenza vaccine design. Vaccine. 2005 Jan 19;23(9):1144-8. https://doi.org/10.1016/j.vaccine.2004.08.028. PMID: 15629357; PMCID: PMC4482133.

The PyMOL Molecular Graphics System, Version 2.0 Schrödinger, LLC.

URL 1: http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/001/SRR1705851/

URL 2: ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/008/SRR1705858/SRR1705858.fastq.gz

URL 3: ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/009/SRR1705859/SRR1705859.fastq.gz

URL 4: ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR170/000/SRR1705860/SRR1705860.fastq.gz

URL 5: https://www.ncbi.nlm.nih.gov/nuccore/KF848938.1?report=fasta

URL 6: https://www.rcsb.org/structure/7qa4

URL 7: https://www.rcsb.org/structure/1eo8

Supplementary materials

Tables

Table 3: **Subject's virus's SNPs.**

Position	**Reference**	**Alternative**	**Frequency**
72	A	G	99.95%
117	C	T	99.73%
165	T	C	0.2%
307	C	T	0.92%
389	T	C	0.22%
722	A	G	0.21%
774	T	C	99.96%
802	A	G	0.28%
913	T	C	0.2%
915	T	C	0.24%
999	C	T	99.86%
1086	A	G	0.24%
1213	A	G	0.26%
1260	A	C	99.93%
1280	T	C	0.2%
1458	T	C	0.86%

Table 4:
SNPs in SRR1705858 control sample.

Position	Reference	Alternative	Frequency
38	T	C	0.66%
54	T	C	0.34%
72	A	G	0.34%
117	C	T	0.33%
165	T	C	0.29%
183	A	G	0.34%
218	A	G	0.28%
222	T	C	0.22%
235	T	C	0.29%
254	A	G	0.23%
276	A	G	0.26%
297	T	C	0.22%
340	T	C	0.28%
389	T	C	0.23%
409	T	C	0.25%
414	T	C	0.22%
463	A	G	0.23%
595	G	T	0.35%
660	A	G	0.25%
670	A	G	0.22%
722	A	G	0.22%
744	A	G	0.24%
774	T	C	0.3%
793	A	G	0.24%
859	A	G	0.26%
898	A	G	0.22%
915	T	C	0.34%
1008	T	G	0.25%
1031	A	G	0.26%
1086	A	G	0.31%
1089	A	G	0.27%
1116	A	G	0.22%
1213	A	G	0.26%
1260	A	C	0.31%
1264	T	C	0.26%
1280	T	C	0.29%
1281	T	C	0.25%
1339	T	C	0.49%
1358	A	G	0.3%
1366	A	G	0.22%
1421	A	G	0.38%
1460	A	G	0.33%
1580	T	C	0.25%
1591	T	C	0.29%

Table 5:
SNPs in SRR1705859 control sample.

Position	Reference	Alternative	Frequency
44	T	C	0.47%
158	A	G	0.24%
165	T	C	0.28%
183	A	G	0.23%
193	A	G	0.23%
218	A	G	0.28%
222	T	A	0.21%
276	C	G	0.22%
319	T	C	0.22%
340	T	C	0.23%
356	A	G	0.22%
370	A	G	0.23%
398	A	G	0.25%
409	T	C	0.22%
463	A	G	0.23%
499	A	G	0.21%
516	A	G	0.22%
660	A	G	0.33%
722	A	G	0.26%
744	A	G	0.23%
793	A	G	0.24%
859	A	G	0.27%
898	A	G	0.25%
915	T	C	0.24%
1031	A	G	0.32%
1100	T	C	0.22%
1213	A	G	0.25%
1264	T	C	0.25%
1280	T	C	0.24%
1281	T	C	0.22%
1358	A	G	0.3%
1366	A	G	0.22%
1421	A	G	0.26%
1460	A	G	0.41%
1517	A	G	0.24%
1520	T	C	0.27%
1600	T	C	0.35%
1604	T	C	0.31%

Table 6:
SNPs in SRR1705860 control sample.

Position	Reference	Alternative	Frequency
38	T	C	0.7%
44	T	C	0.5%
105	A	G	0.3%
158	A	G	0.32%
165	T	C	0.29%
183	A	G	0.23%
193	A	C	0.22%
216	A	G	0.21%
235	T	C	0.29%
254	A	G	0.24%
271	A	G	0.23%
276	A	G	0.35%
297	T	C	0.3%
319	T	C	0.22%
340	T	C	0.25%
370	A	G	0.25%
414	T	C	0.25%
494	A	G	0.22%
566	A	G	0.29%
597	A	G	0.21%
660	A	G	0.27%
722	A	G	0.32%
759	T	C	0.23%
859	A	G	0.22%
915	T	C	0.31%
1031	A	G	0.26%
1086	A	G	0.31%
1089	A	G	0.24%
1105	A	G	0.25%
1209	A	G	0.32%
1213	A	G	0.27%
1264	T	C	0.32%
1280	T	C	0.25%
1281	T	C	0.22%
1358	A	G	0.32%
1366	A	G	0.26%
1398	T	C	0.27%
1421	A	G	0.44%
1460	A	G	0.3%
1580	T	C	0.27%
1604	T	C	0.3%

Scripts

Python script for significant SNPs extraction

Files *58.txt*, *59.txt*, *60.txt*, *patient.txt*, *patient_95.txt* contain some of the columns from **.vcf* files provided by VarScan for SRR1705858, SRR1705859, SRR1705860 control samples (with 0.01% threshold) and for initial reads (with 0.1% and 95% threshold), respectively.

```
import pandas as pd
import numpy as np
import re

data = pd.read_csv('58.txt', sep=" ", header=None)
data.columns = ["Strain", "Position", "Ref_nucleotide",
↪  "Nucleotide", "Frequency"]

percent =
↪  np.array([float(re.sub(r'(.\/1:)(\S+:)(\S+:)(\S+:)(\S+:)
↪  (\S+:)(\S+)(\%:\S+:)(\S+:)(\S+:)(\S+:)(\S+:)(\S+)',
↪  lambda x: x.group(7), text)) for text in
↪  data["Frequency"]])

mean58, std58 = percent.mean(), percent.std()

print('58.txt mean and std:', mean58, std58)

data = pd.read_csv('59.txt', sep=" ", header=None)
data.columns = ["Strain", "Position", "Ref_nucleotide",
"Nucleotide", "Frequency"]

percent =
↪  np.array([float(re.sub(r'(.\/1:)(\S+:)(\S+:)(\S+:)(\S+:)
↪  (\S+:)(\S+)(\%:\S+:)(\S+:)(\S+:)(\S+:)(\S+:)(\S+)',
↪  lambda x: x.group(7), text)) for text in
↪  data["Frequency"]])

mean59, std59 = percent.mean(), percent.std()

print('59.txt mean and std:', mean59, std59)

data = pd.read_csv('60.txt', sep=" ", header=None)
```

```
data.columns = ["Strain", "Position", "Ref_nucleotide",
↪  "Nucleotide", "Frequency"]

percent =
↪  np.array([float(re.sub(r'(.\/1:)(\S+:)(\S+:)(\S+:)(\S+:)
↪  (\S+:)(\S+)(\%:\S+:)(\S+:)(\S+:)(\S+:)(\S+:)(\S+)',
↪  lambda x: x.group(7), text)) for text in
↪  data["Frequency"]])

mean60, std60 = percent.mean(), percent.std()

print('60.txt mean and std:', mean60, std60)

data_patient = pd.read_csv('patient.txt', sep=" ",
↪  header=None)
data_patient.columns = ["Strain", "Position",
↪  "Ref_nucleotide", "Nucleotide", "Frequency"]

percent =
↪  np.array([float(re.sub(r'(.\/1:)(\S+:)(\S+:)(\S+:)(\S+:)
↪  (\S+:)(\S+)(\%:\S+:)(\S+:)(\S+:)(\S+:)(\S+:)(\S+)',
↪  lambda x: x.group(7), text)) for text in
↪  data_patient["Frequency"]])

away = list(np.array(data_patient["Position"]\
.astype(np.int32))[percent >= mean58 + 3 * std58])

for item in np.array(data_patient["Position"]\
↪  .astype(np.int32))[percent >= mean59 + 3 * std59]:
    if not item in away:
        away += [item]

for item in np.array(data_patient["Position"]\
    .astype(np.int32))[percent >= mean60 + 3 * std60]:
    if not item in away:
        away += [item]

data = pd.read_csv('patient_95.txt', sep=" ", header=None)
data.columns = ["Strain", "Position", "Ref_nucleotide",
↪  "Nucleotide"]
```

```
real = []

for item in away:
        if not item in np.array(data["Position"]\
            .astype(np.int32)):
            real += [item]
print(
↪  data_patient.loc[data_patient["Position"].isin(real)])
```

Python script for average coverage (depth) calculation

File *alignment_pile.mpileup* provided by *samtools mpileup* contains information on stacked up (or, in other words, piled up) initial reads.

```
import pandas as pd
import numpy as np

data = pd.read_csv('alignment_pile.mpileup', sep="\t",
↪  header=None)
print(np.mean(data[3].astype(np.float64)))
```

Python script for statistics of control samples calculation

Files *58.txt*, *59.txt*, *60.txt* contain some of the columns from **.vcf* files provided by VarScan for SRR1705858, SRR1705859, SRR1705860 control samples (with 0.01% threshold), respectively.

```
import pandas as pd
import numpy as np
import re
from collections import Counter

data = pd.read_csv('58.txt', sep=" ", header=None)
data.columns = ["Strain", "Position", "Ref_nucleotide",
↪  "Nucleotide", "Frequency"]
position58 = np.array(data["Position"].astype(np.int32))
freq58 =
↪  np.array([float(re.sub(r'(.\/1:)(\S+:)(\S+:)(\S+:)(\S+:)
↪  (\S+:)(\S+)(\%:\S+:)(\S+:)(\S+:)(\S+:)(\S+:)(\S+)',
↪  lambda x: x.group(7), text)) for text in
↪  data["Frequency"]])
```

```
data = pd.read_csv('59.txt', sep=" ", header=None)
data.columns = ["Strain", "Position", "Ref_nucleotide",
↪  "Nucleotide", "Frequency"]
position59 = np.array(data["Position"].astype(np.int32))
freq59 =
↪  np.array([float(re.sub(r'(.\/1:)(\S+:)(\S+:)(\S+:)(\S+:)
↪  (\S+:)(\S+)(\%:\S+:)(\S+:)(\S+:)(\S+:)(\S+:)(\S+)',
↪  lambda x: x.group(7), text)) for text in
↪  data["Frequency"]])

data = pd.read_csv('60.txt', sep=" ", header=None)
data.columns = ["Strain", "Position", "Ref_nucleotide",
↪  "Nucleotide", "Frequency"]
position60 = np.array(data["Position"].astype(np.int32))
freq60 =
↪  np.array([float(re.sub(r'(.\/1:)(\S+:)(\S+:)(\S+:)(\S+:)
↪  (\S+:)(\S+)(\%:\S+:)(\S+:)(\S+:)(\S+:)(\S+:)(\S+)',
↪  lambda x: x.group(7), text)) for text in
↪  data["Frequency"]])

cnt = Counter(np.hstack((position58, position59,
↪  position60)))
fr1, fr2, fr3 = [], [], []
count1, count2, count3 = 0, 0, 0

for k, it in cnt.items():
    if it == 1:
        if (position58 == k).any():
            fr1 += [freq58[position58 == k][0]]
            count1 += 1
        if (position59 == k).any():
            fr1 += [freq59[position59 == k][0]]
            count1 += 1
        if (position60 == k).any():
            fr1 += [freq60[position60 == k][0]]
            count1 += 1
    if it == 2:
        if (position58 == k).any():
            fr2 += [freq58[position58 == k][0]]
            count2 += 1
        if (position59 == k).any():
```

```
            fr2 += [freq59[position59 == k][0]]
            count2 += 1
        if (position60 == k).any():
            fr2 += [freq60[position60 == k][0]]
            count2 += 1
    if it == 3:
        if (position58 == k).any():
            fr3 += [freq58[position58 == k][0]]
            count3 += 1
        if (position59 == k).any():
            fr3 += [freq59[position59 == k][0]]
            count3 += 1
        if (position60 == k).any():
            fr3 += [freq60[position60 == k][0]]
            count3 += 1

mean1, std1 = np.mean(fr1), np.std(fr1)
mean2, std2 = np.mean(fr2), np.std(fr2)
mean3, std3 = np.mean(fr3), np.std(fr3)

part1, part2, part3 = 0, 0, 0
for k, it in cnt.items():
    if it == 1:
        if (position58 == k).any():
            part1 += float(freq58[position58 == k][0] >=
            ↪  mean1 - std1 and freq58[position58 == k][0]
            ↪  <= mean1 + std1)
        if (position59 == k).any():
            part1 += float(freq59[position59 == k][0] >=
            ↪  mean1 - std1 and freq59[position59 == k][0]
            ↪  <= mean1 + std1)
        if (position60 == k).any():
            part1 += float(freq60[position60 == k][0] >=
            ↪  mean1 - std1 and freq60[position60 == k][0]
            ↪  <= mean1 + std1)
    if it == 2:
        if (position58 == k).any():
            part2 += float(freq58[position58 == k][0] >=
            ↪  mean2 - std2 and freq58[position58 == k][0]
            ↪  <= mean2 + std2)
        if (position59 == k).any():
```

```
            part2 += float(freq59[position59 == k][0] >=
            ↪  mean2 - std2 and freq59[position59 == k][0]
            ↪  <= mean2 + std2)
        if (position60 == k).any():
            part2 += float(freq60[position60 == k][0] >=
            ↪  mean2 - std2 and freq60[position60 == k][0]
            ↪  <= mean2 + std2)
    if it == 3:
        if (position58 == k).any():
            part3 += float(freq58[position58 == k][0] >=
            ↪  mean3 - std3 and freq58[position58 == k][0]
            ↪  <= mean3 + std3)
        if (position59 == k).any():
            part3 += float(freq59[position59 == k][0] >=
            ↪  mean3 - std3 and freq59[position59 == k][0]
            ↪  <= mean3 + std3)
        if (position60 == k).any():
            part3 += float(freq60[position60 == k][0] >=
            ↪  mean3 - std3 and freq60[position60 == k][0]
            ↪  <= mean3 + std3)

print("Mean, std and part of close frequencies for count =
↪  1:", mean1, std1, part1 / count1)
print("Mean, std and part of close frequencies for count =
↪  2:", mean2, std2, part2 / count2)
print("Mean, std and part of close frequencies for count =
↪  3:", mean3, std3, part3 / count3)
```