Analyzing SNPs Associated With Periodontal Disease in Head and Neck Cancer Irradiation Patients

Claire Menard, Jordan Connors, Lauren Roppolo Brazell, Valentina Talevi

Background:

Cancer of the head and neck (hereby abbreviated to HNC) encompasses a group of malignant tumors associated with this area of the body [25]. More than 63,000 new diagnoses of head and neck cancer of the oral cavity, pharynx, or larynx occur in the United States with 13,360 estimated deaths each year [1]. Like many other cancers, this disease is often treated with chemotherapy and radiation. Approximately 50-60 percent of patients with cancer will receive radiation therapy (hereby abbreviated to RT) at some point during the treatment of their illness [5]. Although effective, RT can have a harmful effect on the delicate tissues that surround the head and neck. Periodontitis, an oral disease, is known to arise in patients that undergo radiation treatment for cancer of the head and neck. Periodontitis is defined as an inflammatory disease of the supporting tissues of the teeth caused by specific microorganisms, resulting in progressive destruction of the periodontal ligament and alveolar bone with pocket formation, recession or both [20].

Periodontitis contributes to disability, increased costs, and a diminished quality of life in affected individuals [2]. Several oral toxicities can compromise delivery of optimal cancer therapy protocols [18]. For example, dose reduction or treatment schedule modifications may be necessary to allow for resolution of oral lesions. In cases of severe periodontitis, the patient may no longer be able to continue cancer therapy and the treatment with RT is then usually discontinued.

Some patients who undergo radiation therapy develop different levels of periodontitis, even though they may receive similar radiation in total dose and intensity. Because no two patients are exactly alike, certain variations in genetic makeup account for differences in how a patient has more genetic predisposition to developing a specific disease. Single nucleotide polymorphisms (hereby abbreviated to SNPs), indicate genetic differences and account for most of the known genetic variation between individuals, and nearly 300 genes have undergone a detailed analysis of SNP content to account for the genetic differences they reflect [24]. However, not all of these differences are advantageous for patients. Some SNPs are known to correlate directly with certain diseases. By looking into SNP makeup, it is possible to determine an individual's likelihood of a developing a disease, and further, predicting which individuals are most likely to develop complications resulting from RT [23]. Genes involved in DNA repair are among the most studied SNPs [19]. Based on previous research, it has been found that SNPs in genes Interleukin-1α and Interleukin-1β have an inflammatory role in periodontitis pathogenesis [22].

The Interleukin-1 family of genes is responsible for encoding proteins that mediate inflammatory responses in the body [20]. IL-1 stimulates bone resorption directly, and indirectly degrades the extracellular matrix by way of matrix metalloproteinases [18]. Though it is a disease that arises due to many factors, the biologic activities of IL-1 have been shown to coincide with the critical events of attachment loss and bone resorption that occur during periodontal disease [18]. Further evaluation shows that SNPs on the IL-1β and IL-1α gene have shown association with higher severities of periodontitis [18]. Specifically, IL-1α functions as a signaling mechanism to alert the nucleus of the cell that chromatin has been damaged, and produces an inflammatory response as a result of this [15], and in addition, IL-1β helps to propagate the inflammatory response at a local and systemic level [13].

The overall objective of this study is to identify genes and their polymorphisms that are significantly associated with periodontal disease in the HNC patients who undergo RT. A genome-wide association study (hereby abbreviated to GWAS) was chosen to employ the use of a control (those without disease of interest before radiation) and a case positive (those who developed disease of interest after radiation) to compare the genetic differences between the two groups. Using the GWAS, genomes of the patients of interest can be scanned to look for the aforementioned specific SNPs. These SNPs are important because they can provide further insight into whether any correlation exists between the genetic variants seen and the likelihood that a person has or will develop periodontitis.

Bioinformatics tools are important in genome-wide association studies because they allow for management and analysis of large quantities of data. In this project PyVCF, a variant call format reader for Python, will be used as a template for filtering VCF files based on properties of interest. This package is designed to manipulate, parse, and merge VCF files, a type of file used to store DNA polymorphism data. Accessible data from these file types include polymorphisms such as deletions, insertions, and SNPs. PyVCF will be used as a template for a Python script designed to manipulate and merge the patient VCF files in this project.

PLINK software will be used to perform logit regression analysis, allowing for identification of any SNPs present in different individuals that may be associated with the presence of periodontal disease. Finally, the Assocplots Python package will be used to summarize and display results on a Manhattan Plot and a quantile-quantile (QQ) plot.

Previous Work:

Previous studies have established that oral complications and diseases are a major problem in the aftermath of radiation therapy (RT) for head and neck cancer patients. A study published in 2017 assessed the significance of oral disease before radiation therapy in patients with HNC. This included examining caries (cavities),

reviewing dental recommendations, and diagnosing periodontal disease. Out of the 356 participants, 37.2% had caries and 49.5% of patients had treatment done before radiation therapy, with the most common form being extractions. These findings indicate that many individuals who undergo radiation for HNC have some sort of pre-existing ailment and are treated [4].

Another study evaluated the oral complications six-months post-radiation therapy, with these complications significantly affecting the overall quality of life for patients. Researchers found that maximal mouth opening dropped from 45.58 to 42.53 mm, 3.8% of patients had ulcers, and 8.1% had mucositis [16]. These findings are extremely troubling as even with the advancements in RT, complications still arise and affect the overall comfort and health of patients. Although these studies have assessed periodontal care before radiation therapy and the abundance of disease post-RT, no studies have analyzed the SNPs associated with periodontal disease in irradiation HNC patients.

The research approach requires a GWAS method to analyze the genetic variants within each patient to determine if any are associated with a particular susceptibility to post-RT oral disease. Thus, a GWAS pipeline is necessary to fully automate this process. The common disease/common variant (CD/CV) hypothesis pushed GWAS into being a common method of identifying SNPs associated with diseases. Although the CD/CV hypothesis does not hold true for all cases, it is worth testing for this study [5].

Furthermore, GWAS analyses have been conducted on patients suffering from chronic periodontitis, but no significant associations were found. For patients with aggressive periodontitis however, CDKN2BAS variants on chromosome 9 were shown to have significant correlation to the disease. Several other signals were identified for caries, but these findings do not have any specific correlation to post-RT [15].

A final step may be necessary to validate the findings of this study, as GWAS is not the only widely used procedure to identify genetic contributions to frequent diseases. A candidate gene study is another approach, but has a reputation for limited success. This method was the original way to evaluate the CD/CV hypothesis, but due to its common false positive associations and non-replication of results, it was soon overshadowed by GWAS. Studies that do both analyses and are able to produce the same results will have better credibility [6].

Methodology:

This study is based on a population of 65 HNC adult patients. All of these patients have undergone radiation therapy for a period of 24 months. At intervals of 6 months, the CAL index was measured for each patient. The CAL index is a representation of the clinical attachment loss and the distance from the cemento-enamel junction to the pocket base for periodontal disease. The population is divided into two groups based on the CAL index. Group 1 will represent a CAL increase of more

than 0.2 mm (periodontal disease positive) and group 0 will represent a CAL increase of less than 0.2 mm (periodontal disease negative) at any given time. These two groups are defined in order to have a binary outcome (case/control) to perform logit regression analysis.

For each patient, demographic data such as age, gender, and race was collected. This data was used in logit regression analysis as covariates. Saliva DNA was extracted at a baseline time point, before the radiation therapy was isolated. Whole exome sequencing for each of the 65 DNA samples was performed and VCF files were generated.

After receiving the VCF file, the first step included looking at file structure. In the case of this project, it was necessary to open the VCF file and understand file formatting in detail.

```
Cat ~/Desktop/Fall_2018/Programming/VCF_Files/Var_IonXpress_CMC-001.vcf | more
##INFO=<ID=OALT,Number=.,Type=String,Description="List of original variant bases">
##INFO=<ID=OMAPALT,Number=.,Type=String,Description="Maps OID,OPOS,OREF,OALT entries to specific ALT alleles">
##deamination_metric=0.151459629242
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  C-001A-Pellet
chr1    871334  .       G       T       2865.66 PASS    AF=1;AO=298;DP=298;FAO=298;FDP=298;FR=.;FRO=0;FSAF=168;FSAR=130;FSRF=0;FSRR=0;FWDB=0.029217
8;FXX=0;HRUN=2;LEN=1;MLLD=66.0345;QD=38.4652;RBI=0.0493028;REFB=0;REVB=0.0397125;RO=0;SAF=168;SAR=130;SRF=0;SRR=0;SSEN=0;SSEP=0;SSSB=-3.89064e-08;S
TB=0.5;STBP=1;TYPE=snp;VARB=0.00012453;OID=.;OPOS=871334;OREF=G;OALT=T;OMAPALT=T  GT:GQ:DP:FDP:RO:FRO:AO:FAO:AF:SAR:SAF:SRF:SRR:FSAR:FSAF:FSRF:FSRR
       1/1:99:298:298:0:0:298:298:1:130:168:0:0:130:168:0:0
```

Figure 1: VCF file format.

VCF is a text file format, and it can be thought of as having three sections: a VCF header, fix region, and a gt region. The VCF meta region is located at the top of the file and contains metadata describing the body of the file. Each CF meta line begins with a '##'. Below the metadata there is the header line that names the 8 fixed columns: CHROM, POS, ID, REF, ALT, QUAL, FILTER, and INFO [12]. CHROM, POS, and ID indicate respectively the chromosome, position and the identifier of each SNP. REF and ALT mean reference base and alternate base, and QUAL is the phred-scaled quality score. FILTER is the filter status, and usually indicates a "pass" if the position has passed all the filters, and INFO is for additional information. Beginning at column ten is a column for every sample. The organization of each cell containing genotype and associated information is specified in column nine, FORMAT [11].

From the exome sequencing, 72 VCF files were obtained (instead of 65 VCF files). For seven samples (CMC008, CMC022, CMC042, CMC043, CMC063, CMC075 and UC006) there are 2 VCF files, generated by 2 runs to achieve the target number of reads. The first VCF parsing that will be performed includes combining the 2 runs for the 7 samples, keeping the SNPs present in either the first or the second sample, and then keeping those SNPs in common that have the highest quality.

Figure 2: Visual representation of SNPs present in both sample runs after VCF parsing for patient CMC008.

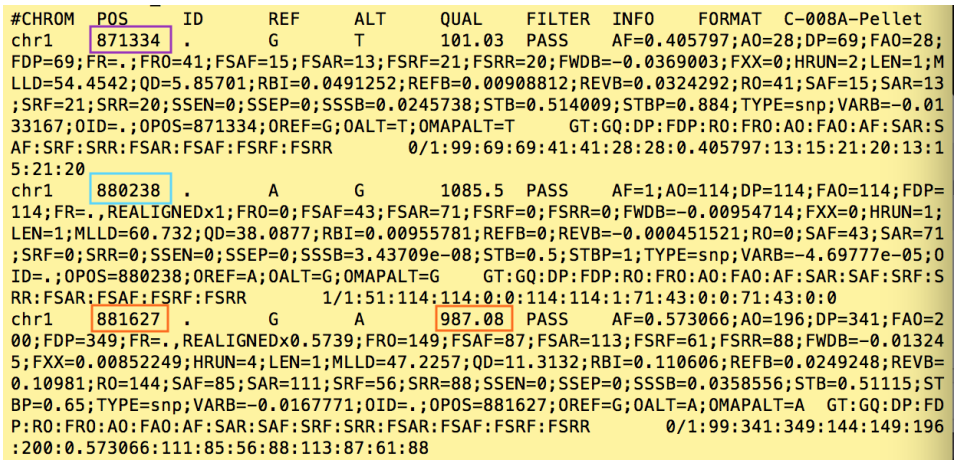
Before comparing the two VCF files for each of the 2 runs, the 98 line headers of both files were removed to make accessing the data quicker and easier.

```
#REMOVE HEADERS FUNCTION:
def remove_headers(input_file):
        lines = open (input_file).readlines()
        outfile = input_file[:-4]+'_noheader.vcf'
        open (outfile, 'w').writelines(lines[98:]
```

Figure 3: Python script for removing header lines.

Figure 4 shows a VCF file comparison of two runs for sample 8. The positions of interest are indicated in purple, and show the differing quality scores at that specific position. The quality scores at position 871334 for both runs are not identical, and require a method for keeping the unique SNPs within both runs, and for identical SNPs choosing the SNPs with the higher quality score.

$ cat Var_IonXpress_CMC-008.vcf | more



```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  C-008A-Pellet
chr1    871334  .       G       T       101.03  PASS    AF=0.405797;AO=28;DP=69;FAO=28;
FDP=69;FR=.;FRO=41;FSAF=15;FSAR=13;FSRF=21;FSRR=20;FWDB=-0.0369003;FXX=0;HRUN=2;LEN=1;M
LLD=54.4542;QD=5.85701;RBI=0.0491252;REFB=0.00908812;REVB=0.0324292;RO=41;SAF=15;SAR=13
;SRF=21;SRR=20;SSEN=0;SSEP=0;SSSB=0.0245738;STB=0.514009;STBP=0.884;TYPE=snp;VARB=-0.01
33167;OID=.;OPOS=871334;OREF=G;OALT=T;OMAPALT=T     GT:GQ:DP:FDP:RO:FRO:AO:FAO:AF:SAR:S
AF:SRF:SRR:FSAR:FSAF:FSRF:FSRR       0/1:99:69:69:41:41:28:28:0.405797:13:15:21:20:13:1
5:21:20
chr1    880238  .       A       G       1085.5  PASS    AF=1;AO=114;DP=114;FAO=114;FDP=
114;FR=.,REALIGNEDx1;FRO=0;FSAF=43;FSAR=71;FSRF=0;FSRR=0;FWDB=-0.00954714;FXX=0;HRUN=1;
LEN=1;MLLD=60.732;QD=38.0877;RBI=0.00955781;REFB=0;REVB=-0.000451521;RO=0;SAF=43;SAR=71
;SRF=0;SRR=0;SSEN=0;SSEP=0;SSSB=3.43709e-08;STB=0.5;STBP=1;TYPE=snp;VARB=-4.69777e-05;O
ID=.;OPOS=880238;OREF=A;OALT=G;OMAPALT=G    GT:GQ:DP:FDP:RO:FRO:AO:FAO:AF:SAR:SAF:SRF:S
RR:FSAR:FSAF:FSRF:FSRR       1/1:51:114:114:0:0:114:114:1:71:43:0:0:71:43:0:0
chr1    881627  .       G       A       987.08  PASS    AF=0.573066;AO=196;DP=341;FAO=2
00;FDP=349;FR=.,REALIGNEDx0.5739;FRO=149;FSAF=87;FSAR=113;FSRF=61;FSRR=88;FWDB=-0.01324
5;FXX=0.00852249;HRUN=4;LEN=1;MLLD=47.2257;QD=11.3132;RBI=0.110606;REFB=0.0249248;REVB=
0.10981;RO=144;SAF=85;SAR=111;SRF=56;SRR=88;SSEN=0;SSEP=0;SSSB=0.0358556;STB=0.51115;ST
BP=0.65;TYPE=snp;VARB=-0.0167771;OID=.;OPOS=881627;OREF=G;OALT=A;OMAPALT=A  GT:GQ:DP:FD
P:RO:FRO:AO:FAO:AF:SAR:SAF:SRF:SRR:FSAR:FSAF:FSRF:FSRR       0/1:99:341:349:144:149:196
:200:0.573066:111:85:56:88:113:87:61:88
```

$ cat Var_IonXpress_CMC-008_run2.vcf | more

```
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  C-008A-Pellet
chr1    871334  .       G       T       332.46  PASS    AF=0.461957;AO=85;DP=184;FAO=85;F
DP=184;FR=.;FRO=99;FSAF=46;FSAR=39;FSRF=54;FSRR=45;FWDB=0.0330374;FXX=0;HRUN=2;LEN=1;MLLD
=53.3846;QD=7.22746;RBI=0.0344683;REFB=-0.023978;REVB=0.00982836;RO=99;SAF=46;SAR=39;SRF=
54;SRR=45;SSEN=0;SSEP=0;SSSB=-0.00418488;STB=0.502318;STBP=0.952;TYPE=snp;VARB=0.0300192;
OID=.;OPOS=871334;OREF=G;OALT=T;OMAPALT=T   GT:GQ:DP:FDP:RO:FRO:AO:FAO:AF:SAR:SAF:SRF:SRR
:FSAR:FSAF:FSRF:FSRR       0/1:99:184:184:99:99:85:85:0.461957:39:46:54:45:39:46:54:45
chr1    881627  .       G       A       829.39  PASS    AF=0.496241;AO=246;DP=518;FAO=198
;FDP=399;FR=.,REALIGNEDx0.5025;FRO=201;FSAF=110;FSAR=88;FSRF=99;FSRR=102;FWDB=-0.022778;F
XX=0.00249994;HRUN=4;LEN=1;MLLD=50.996;QD=8.31472;RBI=0.0792249;REFB=0.0212707;REVB=0.075
8798;RO=271;SAF=138;SAR=108;SRF=137;SRR=134;SSEN=0;SSEP=0;SSSB=0.0532887;STB=0.531912;STB
P=0.187;TYPE=snp;VARB=-0.0217632;OID=.;OPOS=881627;OREF=G;OALT=A;OMAPALT=A  GT:GQ:DP:FDP:
RO:FRO:AO:FAO:AF:SAR:SAF:SRF:SRR:FSAR:FSAF:FSRF:FSRR       0/1:99:518:399:271:201:246:198
:0.496241:108:138:137:134:88:110:99:102
```

Figure 4: VCF file showing differing quality scores for two sample 8 runs.

In order to parse and compare the runs, a dictionary was created for both run 1 and run 2 of the seven patients. First, lines in the VCF file were read and stripped of white space. The dictionary has substrings with the variables CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, FORMAT, and PELLET. Then a key was created using CHROM and POS with the values being everything after. A second dictionary was created for run 2, to ensure proper comparison of each run. A list of keys was then generated using these dictionaries. The Python script for this step is listed below:

```
#CREATE A DICTIONARY FOR EACH RUN
#DICTIONARY FOR RUN1
run1_dic = {} #create an empty dictionary
CMC008_run1 = open('Var_IonXpress_CMC-008_noheader.vcf', 'r') #open a file in modality
"reading"
```

```
for line in CMC008_run1.readlines(): #read each line in the file
        line = line.strip('\n') #delete \n
        CHROM = line.split("\t")[0] #split each list in substring by tab and assign to CHROM variable
the first substring
        POS = line.split("\t")[1]
        ID = line.split("\t")[2]
        REF = line.split("\t")[3]
        ALT = line.split("\t")[4]
        QUAL = line.split("\t")[5]
        FILTER = line.split("\t")[6]
        INFO = line.split("\t")[7]
        FORMAT = line.split("\t")[8]
        PELLET = line.split("\t")[9]
        run1_dic[CHROM,POS] = ID,REF,ALT,QUAL,FILTER,INFO,FORMAT,PELLET#append
the key and values of the list in the dictionary "run1_dic"

#DICTIONARY FOR RUN2
run2_dic = {}
CMC008_run2 = open('Var_IonXpress_CMC-008_run2_noheader.vcf', 'r')
for line in CMC008_run2.readlines():
        line = line.strip('\n')
        CHROM = line.split("\t")[0]
        POS = line.split("\t")[1]
        ID = line.split("\t")[2]
        REF = line.split("\t")[3]
        ALT = line.split("\t")[4]
        QUAL = line.split("\t")[5]
        FILTER = line.split("\t")[6]
        INFO = line.split("\t")[7]
        FORMAT = line.split("\t")[8]
        PELLET = line.split("\t")[9]
        run2_dic[CHROM,POS] = ID,REF,ALT,QUAL,FILTER,INFO,FORMAT,PELLET
```

Figure 5: Python script to create dictionaries.

The lists of keys from the dictionaries were used for comparison of run 1 against run 2, using the *conditional execution* and a *for loop* in Python. *For loops* are used when the goal is to repeat a block of code a fixed number of times. The *if/else statement* evaluates the expression and will execute the *if* body only if the expression is true. If the condition is false, the *else* body will be executed.

This script design considers one key for dictionary 1 and checks if the key is present in the list of keys for dictionary 2. If the key is not present, the key and the

values it contains are kept, because it means that the SNP is present only in run 1. If the key from dictionary 1 is present in the list of keys for dictionary 2 (not unique), the quality (QUAL) values of both keys are compared and the key with the highest quality value is kept.

In order to keep the unique keys and the unique SNPs, in run 2 the same loop must be repeated with consideration of one key at a time for dictionary 2. Before writing any script in Python, a sketch of a workflow for this process was generated:

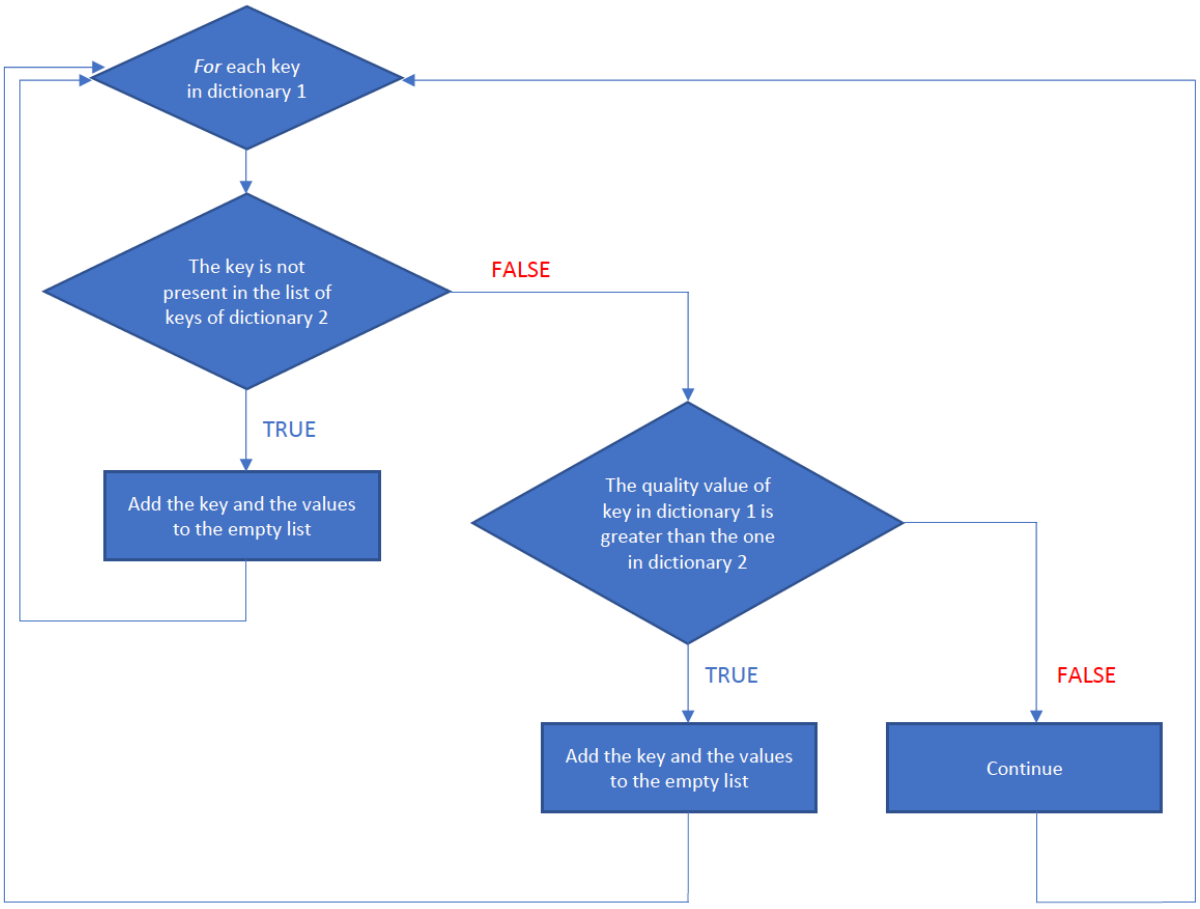


Figure 6: Flowchart of Python loops and conditional statements.

By following this flowchart, a Python script was generated for comparing the two dictionaries.

The following is the Python script:

```
#COMPARE THE RUNS AND WRITE THE RESULTS IN A LIST
CMC008_list = [] #create an empty list
for key in run1_dic.keys(): #for each key in the list of keys in run1_dict
```

```
        if key not in run2_dic.keys(): #if the key is not present in the list of keys in run2_dic
                CMC008_list.append(str(key) + " " + str(run1_dic[key])) #append as string the
key and the values for that key
        else: #if the key is present in the list of keys in run2_dic
                if run1_dic[key][3] >= run2_dic[key][3]: #check the quality column:if the value in
run1_dic(qual) is bigger than the value in run2_dic(qual)
                        CMC008_list.append(str(key) + " " + str(run1_dic[key])) #append as
string the key and the values for that key
                else:

                        continue
#repeat the same loop and condition by looking for keys in run2_dic in run1_dic
for key in run2_dic.keys():
        if key not in run1_dic.keys():
                CMC008_list.append(str(key) + " " + str(run2_dic[key]))
        else:
                if run2_dic[key][3] >= run1_dic[key][3]:
                        CMC008_list.append(str(key) + " " + str(run2_dic[key]))
                else:

                        Continue
```

Figure 7: Python script for comparing two dictionaries.

Once both loops were run through, the CMC008_list was written in a text file. For that, an empty text file was created and the list was copied to it. The following is the Python script:

```
#WRITE THE OUTPUT FILE
CMC008_run1.close()
CMC008_run2.close()
outfile= open("output.txt", "w")
outfile.write(str(CMC008_list))
outfile.close()
```

Figure 8: Python script for writing the output file.

In order to streamline the combining and parsing process for the remaining samples, *functions* were incorporated into the existing code. A *function* is a block of code that performs a specific task on a designated piece of code. Once the function has been defined, it can be called to perform its task on any part of the code in any place, at any time. It is reusable, thus functions were created within the code that removed headers from the files and combined the two runs as we would need them multiple

times for the different runs of the samples. Using the functions allowed the code to be executed in a more concise manner. Below is the new Python code, in its entirety, that was used after defining the functions:

```
#COMBINE RUNS:
def combine_runs(noheader_file_run1, noheader_file_run2):
	run1_dic = {}
	run1_file = open(noheader_file_run1, 'r')
	for line in run1_file.readlines():
		line = line.strip('\n') #delete \n
		CHROM = line.split("\t")[0]
		POS = line.split("\t")[1]
		ID = line.split("\t")[2]
		REF = line.split("\t")[3]
		ALT = line.split("\t")[4]
		QUAL = line.split("\t")[5]
		FILTER = line.split("\t")[6]
		INFO = line.split("\t")[7]
		FORMAT = line.split("\t")[8]
		PELLET = line.split("\t")[9]
		run1_dic[CHROM,POS] =
ID,REF,ALT,QUAL,FILTER,INFO,FORMAT,PELLET
	run2_dic = {}
	run2_file = open(noheader_file_run2, 'r')
	for line in run2_file.readlines():
		line = line.strip('\n') #delete \n
		CHROM = line.split("\t")[0]
		POS = line.split("\t")[1]
		ID = line.split("\t")[2]
		REF = line.split("\t")[3]
		ALT = line.split("\t")[4]
		QUAL = line.split("\t")[5]
		FILTER = line.split("\t")[6]
		INFO = line.split("\t")[7]
		FORMAT = line.split("\t")[8]
		PELLET = line.split("\t")[9]
		run2_dic[CHROM,POS] =
ID,REF,ALT,QUAL,FILTER,INFO,FORMAT,PELLET
	combine_list = []
	for key in run1_dic.keys():
		if key not in run2_dic.keys():
```

```
                    combine_list.append(str(key) + " " + str(run1_dic[key]))
            else:
                    if run1_dic[key][3] >= run2_dic[key][3]:
                            combine_list.append(str(key) + " " + str(run1_dic[key]))
                    else:
                            continue
    for key in run2_dic.keys():
            if key not in run1_dic.keys():
                    combine_list.append(str(key) + " " + str(run2_dic[key]))
            else:
                    if run2_dic[key][3] >= run1_dic[key][3]:
                            combine_list.append(str(key) + " " + str(run2_dic[key]))
                    else:
                            continue
    outfile = open("output.txt", "w")
    outfile.write(str(combine_list))
```

Figure 9: Python script for combining two runs of VCF.

The following script is an example of how the functions were run to combine run 1 and run 2 for patient CMC-022:

```
remove_headers ('Var_IonXpress_CMC-022_run2.vcf')
remove_headers('Var_IonXpress_CMC-022.vcf')
combine_runs('Var_IonXpress_CMC-022_noheader.vcf','Var_IonXpress_CMC-022_run2_noheader.vcf')
```

Figure 10: Python script for writing the output file.

The second step for this project is merging the 65 VCF files together and filtering based on the following criteria:

- Keep SNPs present in at least in 80% of the patients.
- Keep SNPs that are with a minor allele frequency larger than 5% (MAF > 0.05).
- Remove multi-allelic SNPs.

There are several Python packages such as VCFtools that can parse and manipulate VCF files. VCFtools is a program for file comparison, file merging, and variant filtering that will serve as a template for the research team's Python script.

The research team used VCFtools to merge the 65 VCF files by running the following command:

`vcf-merge *.vcf.gz.`

To make the filtering step easier with Python, only the genotype was kept for each patient by running the following step with bcftools:

```
bcftools annotate -x INFO,^FORMAT/GT small_vcf_65.vcf > small_vcf_65_GT
```

Then the quality control was performed by using python script. Quality control was based on 3 filters mentioned above.

The following is the script for removing multi allelic SNPs. First, a dictionary for the merged VCF was created using CHROM and POS as keys.

```
dic_VCF = {}
vcf_file = open('small_vcf_65_GT.vcf', 'r')
forline invcf_file.readlines():
   line = line.strip('\n')
   CHROM = line.split("\t")[0]
   POS = line.split("\t")[1]
   ID = line.split("\t")[2]
   REF = line.split("\t")[3]
   ALT = line.split("\t")[4]
   QUAL = line.split("\t")[5]
   FILTER = line.split("\t")[6]
   INFO = line.split("\t")[7]
   FORMAT = line.split("\t")[8]
   Pat1 = line.split("\t")[9]
   Pat2 = line.split("\t")[10]
   Pat3 = line.split("\t")[11]
   Pat4 = line.split("\t")[12]
   Pat5 = line.split("\t")[13]
   Pat6 = line.split("\t")[14]
   …
   Pat61 = line.split("\t")[69]
   Pat62 = line.split("\t")[70]
   Pat63 = line.split("\t")[71]
   Pat64 = line.split("\t")[72]
   Pat65 = line.split("\t")[73]
   dic_VCF[CHROM,POS]                                                    =
ID,REF,ALT,QUAL,FILTER,INFO,FORMAT,Pat1,Pat2,Pat3,Pat4,Pat5,Pat6,Pat7,Pat8,Pat9,Pat10,Pat11,Pat12,Pat13,Pat14,Pat15,Pat16,Pat17,Pat18,Pat19,Pat20,Pat21,Pat22,Pat23,Pat24,Pat25,Pat26,Pat27,Pat28,Pat29,Pat30,Pat31,Pat32,Pat33,Pat34,Pat35,Pat36,Pat37,Pat38,Pat39,Pat40,Pat41,Pat42,Pat43,Pat44,Pat45,Pat46,Pat47,Pat48,Pat49,Pat50,Pat51,Pat52,Pat53,Pat54,Pat55,Pat56,Pat57,Pat58,Pat59,Pat60,Pat61,Pat62,Pat63,Pat64,Pat65
```

Figure 11:Python script for identifying multi-allelic SNPs.

Then a condition statement was applied to the dictionary: if REF and ALT columns are different than a single base, append the corresponding key to the pop_list. In this way, the pop list contains all the keys for multi allelic SNPs. Then pop function was used to remove these keys from the original dictionary.

```
#DELETE MULTI_ALLELIC SNPS:
pop_list=[]
forkey indic_VCF.keys():
        if((dic_VCF[key][1] != 'A'anddic_VCF[key][1] != 'G'anddic_VCF[key][1] != 'T'anddic_VCF[key][1] != 'C') or(dic_VCF[key][2] != 'A'anddic_VCF[key][2] != 'T'and dic_VCF[key][2] != 'G'anddic_VCF[key][2] != 'C')):
                pop_list.append(key)
        else:
                continue
print(pop_list)

forx inpop_list:
        dic_VCF.pop(x)
print(dic_VCF)
```

Figure 12: Python script for removing multi allelic SNPs.

The second and third steps of filtering were performed using the Pandas package. This package is useful in data analysis of data structures. The following is the script for the second filter: keep the SNPs that are present in at least 80% of the patients or in other words, keep the SNPs that are not present in 20% in the patients or less. The second option (missing less than 20%) was used by the team, and counted the times that a single SNP is not present in the 65 patients. Dots are SNP's that are not present in a particular patient.

The column called 'sum_.' contains the sum of the dots for each SNP. Then a sub-table of the original vcf file was created by deleting all the SNPs that have 'sum_.' value smaller than 13 (20*100/65).

```
Import pandas as pd
VCF_table = pd.DataFrame.from_dict(dic_VCF, orient='index')
VCF_table['sum_.']=(VCF_table.iloc[:,7:] == 'sum_.').sum(axis=1)
#print(VCF_table)

VCF_table_20 = VCF_table[VCF_table['sum_.'] < 13]
print(VCF_table_20)
```

```
VCF_table2 = VCF_table_20.drop(columns=['sum_.'])
```

Figure 13: Python script for removing SNPs.

The third filter is keeping the SNPs with the minor allele frequency larger than 5%. The minor allele frequency equation that has been used is:

MAF = [(2*Alternate Allele)+Heterozygous Allele]/(Total Allele)

The sum of homozygous SNPs for the alternate allele (column 1/1) and the sum of the heterozygous SNPs (0/1) was generated. Then numpy package was used to perform numerical operations. The following is the script that we performed:

```
VCF_table2['0/1']=(VCF_table2.iloc[:,7:] == '0/1').sum(axis=1)
VCF_table2['1/1']=(VCF_table2.iloc[:,7:] == '1/1').sum(axis=1)
#print(VCF_table2)

importnumpy asnp
VCF_table2['sum'] = VCF_table2['1/1']*2+VCF_table2['0/1']
VCF_table2['MAF'] = VCF_table2['sum']/130
#print(VCF_table2)

VCF_table_filter3 = VCF_table2[VCF_table2['MAF'] > 0.05]
print(VCF_table_filter3)
VCF_table_final = VCF_table_filter3.drop(columns=['MAF'])
VCF_table_final = VCF_table_final.drop(columns=['sum'])
VCF_table_final = VCF_table_final.drop(columns=['0/1'])
VCF_table_final = VCF_table_final.drop(columns=['1/1'])
#print(VCF_table_final)
```

Figure 14: Python script for calculating MAF.

Then the final VCF output was written in a VCF file by running the following script:

```
header="""#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT Pat1 Pat2 Pat3 Pat4
Pat5 Pat6 Pat7 Pat8 Pat9 Pat10 Pat11 Pat12 Pat13 Pat14 Pat15 Pat16 Pat17 Pat18 Pat19
Pat20 Pat21 Pat22 Pat23 Pat24 Pat25 Pat26 Pat27 Pat28 Pat29 Pat30 Pat31 Pat32 Pat33
Pat34 Pat35 Pat36 Pat37 Pat38 Pat39 Pat40 Pat41 Pat42 Pat43 Pat44 Pat45 Pat46 Pat47
Pat48 Pat49 Pat50 Pat51 Pat52 Pat53 Pat54 Pat55 Pat56 Pat57 Pat58 Pat59 Pat60 Pat61
Pat62 Pat63 Pat64 Pat65"""

output_VCF = "VCF_with_filter.vcf"
withopen(output_VCF, 'w') asvcf:
        vcf.write(header)
VCF_table_final.to_csv(output_VCF, sep="\t", mode='a', index_label= False)
```

```
infile =  open("VCF_with_filter.vcf", 'r')
data = infile.read()
infile.close()
outfile= open("VCF_with_filter.vcf", "w")
data = data.replace("(", "")
data = data.replace(")", "")
data.replace(",", "")
data.replace("'", "")
outfile.write(data)
outfile.close
```

Figure15: Python script for merging and writing final VCF file.

Before performing any statistical analysis, the original merged VCF was filtered with VCFtools by using the same three types of filters, deleting multi allelic SNPs, keeping SNPs with MAF greater than 0.05 and present in at least 80% if the patients. This step was performed to ensure the reliability of our results. The following is the command ran with VCFtools:

```
vcftools --vcf CAL.vcf --maf 0.05 --max-missing 0.80 --remove-indels --min-alleles 2 --max-alleles 2 --recode --recode-INFO-all --out CAL_filtered --stdout | gzip -c > CAL_filtered.vcf.gz
```

The two output filtered files are different, in particular our output files contains more extra SNPs than VCFtools output file. Since it is important to have reproducible and reliable research in scientific field, the research team decided to do the statistical analysis using the VCFtools file.

Genome wide association studies (GWAS) are used to determine if the genetic variants (SNPs) among the sample of 65 patients are associated with periodontal disease development. Logit regression analysis is the statistical method used to analyze the merged VCF file in order to gain more insight into the relationship between the dichotomous characteristic of interest and the independent variable. This project employs the use of logistic regression analysis, instead of linear regression analysis because covariates (i.e. age, gender, race) can be added to the analysis and not only SNP genotypes.

Logistic regression is a statistical method for analyzing a dataset in which there are one (genotype) or more independent variables that determine the outcome. Logistic regression is a series of single locus statistic tests, examining each SNP independently for association to the phenotype, with a null hypothesis of no association with the phenotype. The result will be the probability of having case status given a genotype

class. Logistic regression generates the coefficients of the following formula to predict the probability of the presence of the characteristic of interest

Logit(p)= b0 + b1.ADD + b2.gender + b3.age + b4.gender + e,

where p is the probability that the patient has periodontal disease, b0 is the intercept of the line, b2, b3 and b4 are the coefficients of genetic and demographic factors and ADD is the genotype at a specific locus.

Within PLINK, the association between SNPs and the binary outcome (disease or no disease) can be tested with the options --assoc or --logistic. With the --logistic option, a logistic regression analysis will be performed which allows for the inclusion of covariates (age, gender and race).

Because PLINK works only with PED and MAP files, VCF files must be converted into these two formats. A PED file is a white-space (space or tab) delimited file; the first six columns are mandatory:

Family ID
Individual ID
Paternal ID
Maternal ID
Sex (1=male; 2=female; other=unknown)
Phenotype

In a MAP file, each line describes a single marker and must contain exactly 4 columns:

chromosome (1-22, X, Y or 0 if unplaced)
rs# or snp identifier
Genetic distance (morgans)
Base-pair position (bp units)

So the PED and FAM files were created by running the following script:

```
vcftools --vcf CAL_filtered.vcf --plink --out plink
```

Based on the demographic and clinical data, the 5th and the 6th columns of the PED file were modified by adding the gender and the phenotype (disease or not), respectively, for each patient.

Clinical Data	Gender	Disease
C-001A-Pellet	Male	Yes
C-003A-Pellet	Male	No
C-004A-Pellet	Female	Yes
C-005A-Pellet	Male	Yes
C-008A-Pellet	Male	No
C-010A-Pellet	Female	Yes
C-011A-Pellet	Male	Yes

Figure 16: Header of Demographic and Clinical Data

C-001A-Pellet	C-001A-Pellet	0	0	1	2	G	A	A
C-003A-Pellet	C-003A-Pellet	0	0	1	1	G	A	A
C-004A-Pellet	C-004A-Pellet	0	0	2	2	0	0	G
C-005A-Pellet	C-005A-Pellet	0	0	1	2	G	A	A
C-008A-Pellet	C-008A-Pellet	0	0	1	1	G	A	A
C-010A-Pellet	C-010A-Pellet	0	0	2	2	A	A	A
C-011A-Pellet	C-011A-Pellet	0	0	1	2	G	A	A

Figure 17: Header of Plink PED file

The following command was used to generate the binary PED file:

```
plink --file plinkCAL --make-bed
```

Once the generation of the PED file was completed, the command for running the logit regression analysis was run:

```
plink --bfile plinkCAL --logistic --ci 0.95 --out logistic.analysis
```

PLINK software generated the following file:

CHR	SNP	BP	A1	TEST	NMISS	OR	SE	L95	U95	STAT	P
1	chr1:881627	881627	G	ADD	54	2.769e-07	512	0	inf	-0.02949	0.9765
1	chr1:894573	894573	G	ADD	61	1.905	1.117	0.2131	17.02	0.5766	0.5642
1	chr1:909238	909238	G	ADD	56	3.529	0.7695	0.7811	15.95	1.639	0.1012
1	chr1:948921	948921	T	ADD	65	0.9	0.8628	0.1659	4.883	-0.1221	0.9028
1	chr1:949654	949654	A	ADD	64	0.4741	0.7782	0.1031	2.179	-0.9592	0.3375
1	chr1:977330	977330	T	ADD	61	0.5854	0.7706	0.1293	2.651	-0.6949	0.4871
1	chr1:982941	982941	T	ADD	59	0.4038	0.7277	0.097	1.681	-1.246	0.2128
1	chr1:982994	982994	T	ADD	51	0.4804	0.8053	0.09912	2.328	-0.9104	0.3626
1	chr1:985266	985266	C	ADD	46	0.6889	1.159	0.07105	6.679	-0.3215	0.7478
1	chr1:987200	987200	C	ADD	63	0.6447	0.7004	0.1634	2.544	-0.6266	0.5309
1	chr1:990280	990280	C	ADD	56	0.8769	0.6766	0.2328	3.303	-0.1941	0.8461
1	chr1:1007203	1007203	A	ADD	57	0.5143	0.727	0.1237	2.138	-0.9146	0.3604
1	chr1:1021415	1021415	A	ADD	59	2.145	0.8531	0.403	11.42	0.8947	0.371
1	chr1:1158631	1158631	A	ADD	64	NA	NA	NA	NA	NA	NA
1	chr1:1247494	1247494	T	ADD	57	4.065	1.109	0.4621	35.75	1.264	0.2062
1	chr1:1249187	1249187	G	ADD	59	2.145	0.8531	0.403	11.42	0.8947	0.371
1	chr1:1254841	1254841	C	ADD	63	0.6829	0.7641	0.1527	3.054	-0.4991	0.6177
1	chr1:1262966	1262966	C	ADD	64	1.047	0.857	0.1951	5.613	0.05305	0.9577

Figure 18: Header of PLINK logit regression output

The file contains 12 columns where:

- CHR: Chromosome
- SNP: SNP identifier
- BP: Physical position (base-pair)
- A1: Tested allele (minor allele by default)
- TEST: Code for the test (see below)
- NMISS: Number of non-missing individuals included in analysis
- BETA/OR: Regression coefficient (--linear) or odds ratio (--logistic)
- STAT: Coefficient t-statistic
- P: Asymptotic p-value for t-statistic

The last step was filtering for p-values smaller than 0.05, since it is necessary to keep just the SNPs that are potentially associated with the disease. R software was used to filter the file by running the following command:

```
logit_table <- read.delim('/Users/Valentina/plink_logistic.txt', header= TRUE, sep = "")
logit_05<-
subset(logit_table, logit_table$P < 0.05)
```

A p-value is the probability of seeing a test statistic equal or greater than the observed test statistic if the null hypothesis is true. Small p-values indicate that, if there is no association between a SNP and the disease, the chance of development of the disease is extremely small. Determining the correct p-value threshold for statistical significance is critical to control the number of false-positive associations. A Bonferroni correction could be used to determine the p-value threshold to use in an analysis, where the conventional p-value (0.05) is divided by the number of tested performed (total number of SNPs). This correction is the most conservative, as it assumes that each association test is independent of all other tests.

The results of this study was summarized and displayed on a Manhattan plot by using R software:

```
library("qqman",
lib.loc="/Library/Frameworks/R.framework/Versions/3.4/Resources/library")
manhattan(logit_table, chr = "CHR", bp = "BP", p = "P", snp = "SNP", col = c("red", "blue"), suggestiveline = -log10(0.01), genomewideline = -log10(0.05), ylim = c(0,4.5))
```

Results:

We were able to accomplish VCF parsing for the seven samples with two runs each. This included removing the headers from each file, creating a dictionary for each individual run, comparing individual runs for each sample and redirecting the results into a list. An output file was written to show the combined VCF files with the highest quality scores, generating seven total output files.

The following is an example of the head of the output file:

```
Cat output.txt | more
```

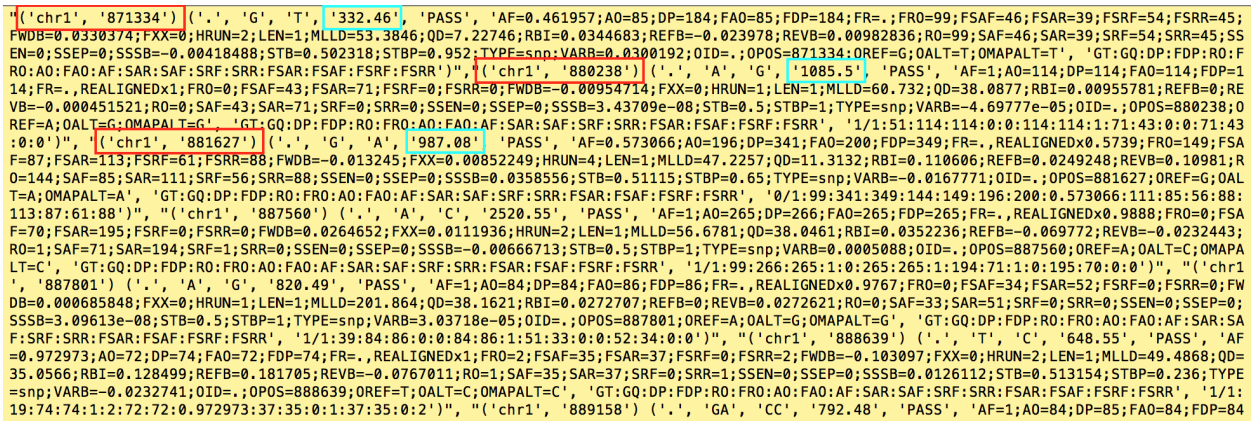


Figure 19: Head of the output file.

Figure 19 shows the output file containing the unique SNPs in run 1 (i.e. 880238 snp) for sample CMC-008, the SNPs present in both runs, with the highest quality (i.e. 871134 snp, 881627 snp) and the unique SNPs in run 2. The next step for this project was merging the 65 patient files, and filtering the merged patient file with python scripting. The criteria used for filtering the merged VCF file included keeping the SNPs present in at least in 80% of the patients, keeping the SNPs that had a minor allele frequency larger than 5% (MAF > 0.05), and removing all multi-allelic SNPs. Upon obtaining the SNP variant results from running our python script for filtering, the results were compared to those generated by VCFtools in order to check for accuracy and ensure quality control. The SNP variant results produced from our Python script generated more SNPs than VCFtools for the same data set, as shown in Figure 20.

#CHROM	POS	ID	RE	ALT	QUAL
'chr1'	'880238'	A	G	884.46	PASS
'chr1'	'881627'	G	A	1234.2	PASS
'chr1'	'883625'	A	G	533.88	PASS
'chr1'	'887560'	A	C	2049.5	PASS
'chr1'	'887801'	A	G	885.78	PASS
'chr1'	'888639'	T	C	935.61	PASS
'chr1'	'888659'	T	C	1055.51	PASS
'chr1'	'894573'	G	A	2547.03	PASS
'chr1'	'897325'	G	C	272.12	PASS
'chr1'	'909238'	G	C	356.61	PASS
'chr1'	'909768'	A	G	1259.58	PASS
'chr1'	'948921'	T	C	2293.6	PASS
'chr1'	'949654'	A	G	2372.49	PASS
'chr1'	'977330'	T	C	293.33	PASS
'chr1'	'982941'	T	C	613.03	PASS

#CHROM	POS	ID	REF	ALT	QUAL	FILTER
chr1	881627	.	G	A	1234.2	PASS
chr1	894573	.	G	A	2547.03	PASS
chr1	909238	.	G	C	356.61	PASS
chr1	948921	.	T	C	2293.6	PASS
chr1	949654	.	A	G	2372.49	PASS
chr1	977330	.	T	C	293.33	PASS
chr1	982941	.	T	C	613.03	PASS

Figure 20. Headers of SNP variant files for our python script (left) and VCFtools (right).

Due to VCFtools filtering more strictly than our script, our team decided to obtain the remainder of the project results with the files generated by VCFtools. By doing so, we were able to produce results of quality, and lower the risk of false positives.

PLINK software was utilized to identify the statistically significant SNPs within the patient samples. From PLINK analysis, we obtained the p-values for 16,637 SNPs. The SNPs with p-values < 0.05 were kept, and a total of 420 SNPs were found to be significantly associated with periodontal disease.

The results of this study were showcased on a Manhattan plot generated by R programming. The Manhattan plot is a scatter plot where genomic coordinates are displayed along the x-axis along with the negative logarithm of the association p-value for each SNP displayed on the y-axis. Each dot on the Manhattan plot signifies a SNP. The red line indicates a significance level of 0.05 and the blue line represents 0.01. Figure 21 shows the strongest associations have the smallest p-value and indicate the greatest negative logarithms.

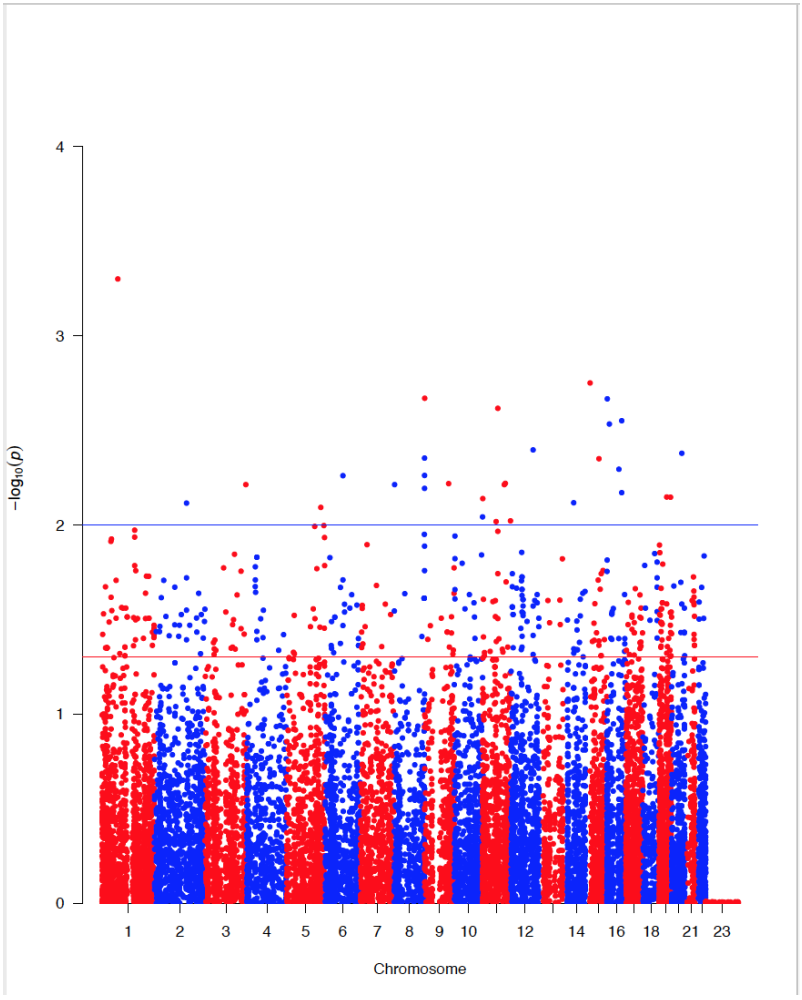


Figure 21. Manhattan plot of SNP variant results.

Discussion:

Our experimental design for this project allowed for removal of headers and creation of dictionaries for each of the 65 patient files using python scripting. With Python programming, we were able to combine the 2 runs generated for the 7 duplicate samples, and keep the SNPs present in both files with the higher quality score as well as the common SNPs that were present in the first and second run. We also were able to use Python to merge the 65 patient VCF files. Filtering the merged VCF file for biologically significant SNPs was accomplished using VCFtools. Statistical analysis was conducted with PLINK software, and generation of a Manhattan plot for visual representation was completed with R programming.

The Manhattan plot generated by R programming showed 16637 identified SNPs. Dots above the red line (significance 0.05) showed a significant correlation of the SNP and the presence of periodontal disease. Dots above the blue line (significance 0.01) were at a more restrictive significance level, and showcase an even stronger correlation with periodontal disease. A total of 420 SNPs were found to be significantly associated with periodontal disease. The corresponding chromosome was identified on the x-axis. The presence of these identified SNPs within the genome may be significantly associated with the development of Periodontal disease post radiation therapy. Due to the sample size being largely homogenous (23% female and 82.9% Caucasian), further statistical analysis is needed to identify false positive results, and we expect the total of 420 periodontal disease associated SNPs to decrease as a result. For further studies, if the SNP is found present in the patient's genome, this may confer more susceptibility to developing periodontal disease post radiation therapy.

Today's society is progressing into the era of personalized medicine in which an individual's genetic makeup will eventually determine how a doctor can tailor treatment strategies. Therefore, it is critical to understand the genetic basis of common diseases. How do genes predispose individuals to diseases? How do gene interactions affect risk of disease development? Moreover, how do rare genetic variants that are difficult to detect contribute to disease presence and prevalence?

Through GWAS analysis it was expected that statistically and biologically significant SNPs associated with periodontal disease post radiation therapy would be identified. The first steps toward identifying SNPs in which the allele frequencies vary systematically as a function of periodontal disease was made possible through use of Python programming, VCFtools and PLINK software. If a certain allele is more common among individuals with the disease than other healthy individuals, this is interpreted as evidence that this allele or perhaps another nearby variant may cause the disease or at least increase the risk of disease development. In this study we present the initial findings of an experimental design aimed at discovering correlations between SNPs and periodontal disease.

Since the VCF files have been generated from exome sequencing, it is expected that these identified SNPs will be found in the coding regions. SNPs can have functional consequences such as amino acid changes, changes to mRNA transcript stability, and changes to transcription factor binding affinity. Some of these SNPs could be potentially protective or confer higher susceptibility to periodontal disease. This study could potentially lead to the development of diagnostic biomarkers for risk assessment, early detection, and personalized treatment in the HNC patient population, but further investigation is necessary to produce valid and thorough conclusions [15].

The SNPs of focus in this study were polymorphisms present in the patients prior to radiation therapy and those that could confer susceptibility to periodontal disease. The radiation therapy was the element that determined the outcome of periodontal disease presence or absence. When a GWAS is performed, special attention should be given to those SNPs that directly influence a trait (direct association), and also to those SNPs that are not directly correlated to the phenotype, but are in high linkage disequilibrium with the tag SNP (indirect association). Linkage disequilibrium refers to the property of the alleles of two or more SNPs that are inherited together within a population. Further studies will need to be completed to determine the nature of the association between the 420 identified SNPs and disease presence.

In conclusion, the research team believes that GWAS was a useful method for identifying SNPs that are statistically associated with periodontal disease. In fact, genome-wide association studies have proved successful in identifying genetic associations with complex traits. Despite these advances, it is important to keep in mind that GWAS has its own limitations that could affect the results, including the presence of false positive SNPs, biases due to case and control selection, small population size, and genotyping errors. In order to account for this, commands within PLINK software allow for quality control when filtering out SNPs based on individual and SNP missingness, inconsistencies in assigned and genetic sex of subjects, minor allele frequency (MAF), deviations from Hardy–Weinberg equilibrium (HWE), heterozygosity rate, relatedness, and ethnic outliers [19]. For future direction, PLINK software could also be used to look further into the biological significance of each SNP, such as determining where specifically each SNP is located in the genome, and looking further into the IL-1 gene family. Moreover, utilizing a larger and more diverse sample size for future studies will aid in decreasing the margin for error by giving a more accurate view of the population as a whole. Further insight into the how this disease affects a diverse set of demographics will contribute to the idea of precision medicine and tailored treatment strategies for each unique patient. As for this study, we hope that these findings help pave the way toward such a goal.

References:

1. American Cancer Society. Cancer facts & figures 2017. Atlanta, GA: American Cancer Society; 2017: 4-18.
2. Ansari Moghadam, Somaye et al. "The Relationship Between Periodontal Disease and Public Health: A Population-Based Study" *Global journal of health science* vol. 8,7 110-5. 18 Nov. 2015, doi:10.5539/gjhs.v8n7p110
3. Aydil U. Recent Landmark Studies on Head and Neck Cancers: Evidence-Based Fundamentals of Modern Therapeutic Approaches. *Turk Arch Otorhinolaryngol*. 2015;53(1):26-31.
4. Brennan, M. T. *et al.* Dental disease before radiotherapy in patients with head and neck cancer. *The Journal of the American Dental Association* **148,** 868–877 (2017).
5. Bush WS, Moore JH (2012) Chapter 11: Genome-Wide Association Studies. PLoS Comput Biol 8(12): e1002822. https://doi.org/10.1371/journal.pcbi.1002822
6. Chang, C. Q. *et al.* A systematic review of cancer GWAS and candidate gene meta-analyses reveals limited overlap but similar effect sizes. *European Journal Of Human Genetics* **22,** 402 (2013).
7. Charlotte E. M. de Mooij, Mihai G. Netea, Walter J. F. M. van der Velden, Nicole M. A. Blijlevens. Targeting the interleukin-1 pathway in patients with hematological disorders. Blood Jun 2017, 129 (24) 3155-3164; **DOI:** 10.1182/blood-2016-12-754994
8. Cohen, Idan et al. "IL-1α is a DNA damage sensor linking genotoxic stress signaling to sterile inflammation and innate immunity" *Scientific reports* vol. 5 14756. 6 Oct. 2015, doi:10.1038/srep14756
9. Dinarello, Charles A. "Interleukin-1 in the pathogenesis and treatment of inflammatory diseases" *Blood* vol. 117,14 (2011): 3720-32.
10. Grigoriadou, Marianna E., and Spiridon-. "Interleukin-1 as a Genetic Marker for Periodontitis." *ResearchGate*, www.researchgate.net/.
11. https://grunwaldlab.github.io/Population_Genetics_in_R/reading_vcf.htm
12. https://samtools.github.io/hts-specs/VCFv4.2.pdf
13. http://zzz.bwh.harvard.edu/plink/data.shtml
14. Ian C. Gray, David A. Campbell, Nigel K. Spurr; Single nucleotide polymorphisms as tools in human genetics, *Human Molecular Genetics*, Volume 9, Issue 16, 1 October 2000, Pages 2403–2408, https://doi.org/10.1093/hmg/9.16.2403
15. Kornman, K. S. & Polverini, P. J. Clinical application of genetics to guide prevention and treatment of oral diseases. *Clinical Genetics* **86,** 44–49 (2014).
16. Lalla, R. *et al.* Oral complications at 6 months after radiation therapy for head and neck cancer. *Oral Diseases* **23,** 1134–1143 (2017).
17. Lazzari, Grazia et al. "Single nucleotide polymorphisms and unacceptable late toxicity in breast cancer adjuvant radiotherapy: a case report" *Breast cancer (Dove Medical Press)* vol. 9 401-406. 29 May. 2017, doi:10.2147/BCTT.S136048
18. López, N. J., Jara, L. and Valenzuela, C. Y. (2005), Association of Interleukin□1 Polymorphisms With Periodontal Disease. Journal of Periodontology, 76: 234-243. doi:10.1902/jop.2005.76.2.234
19. Marees AT, Kluiver HD, Stringer S, Vorspan F, Curis E, Marie-Claire C, Derks EM. A tutorial on conducting genome-wide association studies: Quality control and statistical analysis. *International Journal of Methods in Psychiatric Research* 27, 2018.
20. Newman MG, Carranza FA, Takei H, Klokkevold PR. Carranzas clinical Periodontology. 10th ed. Elsevier health sciences; 2006.
21. Rawlinson A, Grummitt JM, Walsh TF, Ian Douglas CW (2003) Interleukin 1 and receptor antagonist levels in gingival crevicular fluid in heavy smokers versus non-smokers. J Clin Periodontol, 42-48.
22. Rosenstein, Barry S. "Identification of SNPs associated with susceptibility for development of adverse reactions to radiotherapy" *Pharmacogenomics* vol. 12,2 (2011): 267-75.
23. Stefanov, Stefan, et al. "An Analysis Pipeline for Genome-Wide Association Studies." National Center for Biotechnology Information, U.S. National Library of Medicine, 2008, www.ncbi.nlm.nih.gov/pmc/articles/PMC2603547/.
24. Weiss, Cynthia. Rise in Head and Neck Cancer Spurs Innovations in Care. Mayo Clinic. https://newsnetwork.mayoclinic.org/discussion/rise-in-head-and-neck-cancer-spurs-innovations-in-care/ (2016)