Project 2D

Contents

Introduction

Scientific Question

Can the latitudinal trends of rates of cutaneous squamous cell carcinoma be explained by mutations in the p53 gene caused by UV radiation?

Background

Cutaneous squamous cell carcinoma (cSCC) is the second most common form of skin cancer. It is characterized by accelerated growth of squamous cells, which are cells that are found in the tissue that forms the surface of the skin. More than 75% of cSCC and BCC in humans occur on sun-exposed skin. If untreated, cSCC can spread to other parts of the body, leading to serious complications.

p53, also called TP53, is a gene that encodes for a nuclear protein involved in controlling cell division and death. It is classified as a tumor suppressor gene, and mutations in p53 have been found in most tumor types.

UV ligth is a type of light with shorter wavelengths than visible light. Prolonged exposure to UV radiation from the sun (UVA and UVB radiation), is known to be a risk factor for sunburn, premature aging, eye damage, and all kinds of skin cancers.

A recent publication used UVA and UVB radiation to mutagenize the p53 in human cells, and documented the results and the kinds of mutations that arose. Thsese are called UVA and UVB fingerprint mutations because are only found in samples that have been exposed to UV radiation and are known to be caused by UV-induced DNA damage.

Data Used

First we will look at data published by Staples et al in their paper Non-melanoma skin cancer in Australia:the 2002 national survey and trends since 1985. They published data on the rates of cSCC by latitude, with

data split up by gender, age, skin type, and latitude. We will only be looking at the total rates by latitude, to observe what kinds of latitudinal trends exist of cSCC. Next we will use data published by pubmed called Exposure Data that is part of an excerpt of a book called Solar and Ultraviolet Radiation. We will chart this to see if the trend of UV radiation by latitude is similar to the trend of cSCC based on latitude. Then we will perform multiple sequence alignment. Our wild-type amino acid sequence of p53 was downloaded from NCBI gene. To encode the UVA and UVB signature mutations, we will be uploading a table published by Agar et al in the publication titled: The basal layer in human squamous tumors harbors more UVA than UVB fingerprint mutations: A role for UVA in human skin carcinogenesis. (PMID:15041750) We will use this table to write code that will introduce the mutations that this paper publshed occur as a result of UV exposure to the p53 gene. Then we will design sequences with mutations that are in the database cBioportal in three of their samples, Clin Cancer Res 2015 Mutations, MD Anderson, Clin Cancer Res 2014, and UCSF_NPJ_Genom_Med_2021. Van Kempen Et Al defined a hotspot mutation in sCCC as a mutation that was present in 22% of patients sampled, so we will use this value as a threshold of what should be considered a significant mutation (hotspot mutation).

Hypothesis

If UV radiation can cause mutations in p53 that result in cutaneous squamous cell carcinoma (cSCC), then we would expect to see places with lower latitudes experience lower levels of UV radiation, and we expect these places to have higher incidence of cSCC as well as UV fingerprint mutations in the P53 genes in cSCC tumor tissue samples.

Description of Analyses

First we will use ggPlots to visualize data on how different latitudes experience different levels of UV radiaiton. We will also plot data on how the rates of cSCC in the population differ based on latitude. Hopefully these reveal similar trends, which would support our hypothesis. Next we will perform multiple sequence alignments. First we will generate mutant sequences that contain all of the UVA and UVB fingerprint mutations, and we will align these with our wild-type sequence to visualize the UV generated mutaitons. Next we will generate sequences that have the cBioportal mutations that were found in real tumor samples of the p53 gene of individuals with cSCC. We will align these with out wild-type and uv-mutated sequences to see if any of the UV mutations appear in the real tumor samples. We will use the information from the mutations and perform p-values to determine if we can conclude that any of these mutations are hotspot mutaitons in cSCC.

Loading in Packages

```
#if (!requireNamespace("BiocManager", quietly=TRUE))
#install.packages("BiocManager")
#library(BiocManager)
```

BiocManager allows me to install the necessary packages from Bioconductor. If I didn't have BiocManager I wouldn't be able to install msa for my alignments.

```
#BiocManager::install("msa")
library(msa)
```

```
## Loading required package: Biostrings
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

## Loading required package: stats4

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

## Loading required package: XVector

## Loading required package: GenomeInfoDb

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit
```

msa is a package that has functions for performing multiple sequene alignment. I will be using this package to align my p53 sequences to observe the kinds of mutations that are found in the p53 gene.

```
#install.packages("ggplot2")
#This package is what I will use to display the UV data by latitude as well as the info for rates of cS
library(ggplot2)
```

ggplot2 is a package that allows you to use the ggplot() function for creating graphs from data. I will be using it to create graphs that show the rates of cSCC by latitude, as well as creating graphs that show how the levels of UVA and UVB radiation vary by latitude. There are lots of cool features in ggplot2 that allow you to customize your graphs.

```
#install.packages("seqinr")
library(seqinr)
```

```
##
## Attaching package: 'seqinr'
```

```
## The following object is masked from 'package:Biostrings':
##
##     translate
```

Seqinr has a lot of tools for loading in and analyzing biological data. I will be using the read.fasta() function to read in the AA sequence data that I will be using for my alignments. I will also be using the write.fasta() file to create individual fasta files for each mutated sequence, and then I will use fastaconc() to combine these individual files into a single fasta file that I will use for alignment.

```
#install.packages("EnvNJ")
library(EnvNJ)
```

I will be using the fastaconc() function from the EnvNJ package to create new fasta files that contian all of the sequences for my alignments. Since each alignment will be with the same gene, I will read in the wild-type amino acid sequence, and use this sequence as the basis for my mutated alignments. After adding the mutations, I will use fastaconc() to create new fasta files that have the sequences that I need to align so that I can just read back in these fasta files and use them in the msa function to align them.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:seqinr':
##
##     count
```

```
## The following objects are masked from 'package:Biostrings':
##
##     collapse, intersect, setdiff, setequal, union
```

```
## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect
```

```
## The following object is masked from 'package:XVector':
##
##     slice
```

```
## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union
```

```
## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union
```

```
## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

The dplyr package will allow me to use the select() function, which will help me a lot with my analysis. I will use this function when I need to select specific rows/columns from a dataframe.

```
#install.packages("pdftools")
library(pdftools)
```

```
## Using poppler version 20.12.1
```

My R studio has issues using the msaPrettyPrint function, and it will create a temporary pdf but will result in an error. To be able to display these alignments, I will use the pdf_convert() function to turn the pdfs that are generated by msaPrettyPrint into .png files so that I can show these in my final knitted html.

```
#install.packages("utils")
library(utils)
```

The utils package has the functions read.csv and read.delim that are helpful for reading data tables into the r notebook. I downloaded a couple of tables, for example the tsv files that have information on the kinds of mutations that are found in p53 genes from cSCC tumor samples.

```
#install.packages('knitr', dependencies = TRUE)
library(knitr)
```

The knitr package allows me to knit in the pngs that I make from hte msaPrettyPrint pdfs. Sometimes the output of the writing the pdfs into a png is not easily accesible by r, so using knitr to display the png images is the most reliable way to show them in the final html or PDF that is generated.

```
library(tinytex)
```

Making sure that tinytex is in the library helps at the end when I convert the R markdown into a PDF. Just as a backup I will upload a PDF to GitHub in addition to the html and .rmd, and without tinytex the conversion isn't possible.

Bioinformatics Methods

1. ggPlots

The first step is to prove that the part of the hypothesis that claims that there are latitudinal differences in the rates of cSCC. Our proposed mechanism states that a possible explanation for why there are different rates of cSCC by latitude could be because of different rates of UVA and UVB radiation by latitude.

But how do we know that there are latitudinal differences between rates of cSCC or in levels of UVA or UVB radiation?

1. Look at if there is a latitudinal difference between rates of cSCC. Using data from Non-melanoma skin cancer in Australia: the 2002 national survey and trends since 1985 by Staples et al (PMID: 26547141)

```
scc_rates_data<-read.csv("SCC_data.csv")
#Use the read.csv function to import the data from Staples et al
print(scc_rates_data)
```

```
##        Basal.cell.carcinoma Squamous.cell.carcinoma           X        X.1
## 1                                                Men       Women      Total
## 2                  Latitude
## 3     North region (< 29ºS)                    2145        1259       1662
## 4                (1778-2588)              (1004-1579) (1439-1921) (961-1601)
## 5       Central (29ºS-37ºS)                    1088         843        959
## 6                 (935-1265)               (718-990)  (859-1071)  (380-588)
## 7             South (> 37ºS)                     646         462        547
## 8                  (505-827)               (349-614)   (454-659)  (217-433)
## 9                  Skin type
## 10              Tans deeply                     685         484        585
```

```
## 11                (529-889)               (340-688)   (475-721)  (178-403)
## 12          Tans moderately                     935         558        722
## 13               (796-1098)               (458-679)   (637-818)  (321-514)
## 14             Does not tan                    1406        1217       1271
## 15               (1092-1811)             (993-1491) (1085-1489)  (339-629)
## 16          Region of birth
## 17                Australia                    1367         913       1113
## 18               (1218-1535)             (803-1038) (1021-1213)  (600-826)
## 19 All other than Australia                     614         462        541
## 20                (469-803)               (337-632)   (441-663)  (134-322)
## 21           United Kingdom                     758         648        703
## 22                (527-1091)              (434-966)   (537-920)
## 23          Southern Europe                     470         350        385
## 24                (152-1457)              (88-1401)   (160-925)
##                     X.2       X.3              X.4
## 1                   Men     Women            Total
## 2
## 3                  1240       429    794
## 4             (285-645) (639-985)
## 5                   473       400              432
## 6             (318-503) (368-506)
## 7                   306       171              232
## 8             (109-267) (177-306)
## 9
## 10                  268       132              215
## 11             (71-245) (153-303)
## 12                  406       252              319
## 13            (189-335) (266-382)
## 14                  965       462              611
## 15           (715-1301) (493-758)
## 16
## 17                  704       412              541
## 18            (340-498) (479-611)
## 19                  207        95              152
## 20             (50-183) (106-219)
## 21 Insufficient data *
## 22
## 23 Insufficient data *
## 24
```

```
#The table is pretty messy so we need to clean it up before we can start plotting.
clean_scc_rates_data<-scc_rates_data[, c(1, 7)]
#Select the first and last columns that have information on the latitude and totla number of cases of c
clean_scc_rates_data<-clean_scc_rates_data[c(3, 5, 7), ]
#We aren't looking at age or gender differences, so make a simple dataframe that just has info on the l
print(clean_scc_rates_data)
```

```
##    Basal.cell.carcinoma              X.4
## 3 North region (<29ºS)    794
## 5   Central (29ºS-37ºS)              432
## 7        South (>37ºS)              232
```

```
colnames(clean_scc_rates_data)<-c("Location", "Rate_of_cSCC")
#Add column names.
print(clean_scc_rates_data[1,2])
```

```
## [1] "   794   "
```

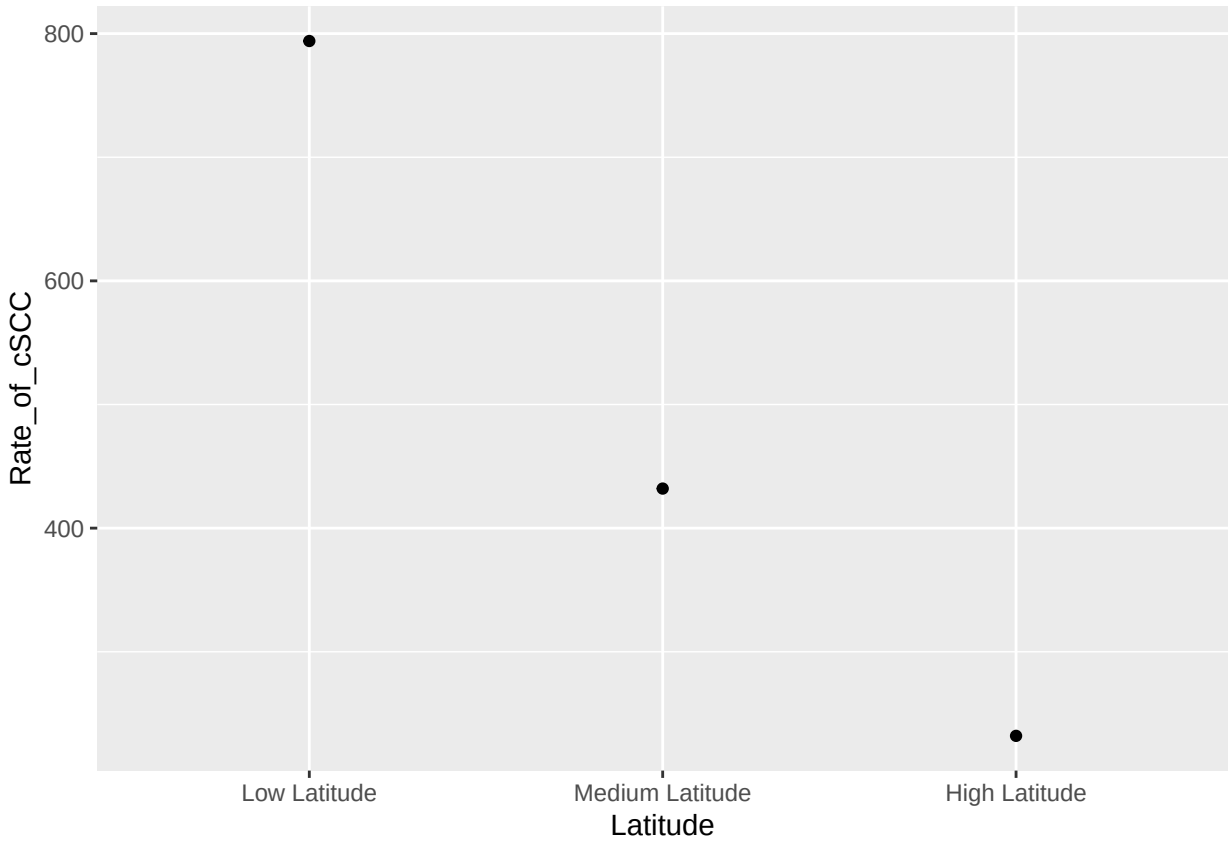
```
#Inspect one of the elements that has wonky formatting. Lots of extra spaces on either sides of the act
clean_scc_rates_data[1,2]<-gsub("[[:space:]]", "", clean_scc_rates_data[1,2])
#Remove the extra spaces and replace the entry with the same number but with proper formatting.
clean_scc_rates_data[,2]<-as.numeric(as.character(clean_scc_rates_data[,2]))
#Since we will be plotting the rates, we want them to be numeric so that ggplot doesn't think that they
clean_scc_rates_data$Location <- as.character(clean_scc_rates_data$Location)
clean_scc_rates_data$Location <- factor(clean_scc_rates_data$Location, levels=unique(clean_scc_rates_da
#Turn the location variable which will be the x axis into a character and back into a factor so ggplot
#print(clean_scc_rates_data)
#Make sure the data looks right and is ready to be plotted.
latitudes<-c("Low Latitude", "Medium Latitude", "High Latitude")
#Instead of doing north central and south, put the location into context of the hypothesis and say what
clean_scc_rates_data$Latitude<-latitudes
#Append the character vector we just wrote to the dataframe.
print(clean_scc_rates_data)
```

```
##                Location Rate_of_cSCC        Latitude
## 3 North region (< 29ºS)          794    Low Latitude
## 5   Central (29ºS-37ºS)          432 Medium Latitude
## 7         South (> 37ºS)          232   High Latitude
```

```
#Check to make sure it looks ok.

cSCC_rate_plot<-ggplot(clean_scc_rates_data, aes(x=Latitude, y=Rate_of_cSCC))+scale_x_discrete(limits=cl
#Use the ggplot2 package and the ggplot function to plot the data. Our x axis is the latitude, and the

print(cSCC_rate_plot)
```




```
#Print the plot to check
```

Based on the plot, we can see that there is a clear trend with lower latitudes having higher rates of cSCC.

Let's check if there is a similar trend in latitude versus amount of UV Radiation

This data was downloaded from an entry on pubmed called Solar and Ultraviolet Radiation: Exposure data (https://www.ncbi.nlm.nih.gov/books/NBK401584/)

```
uv_data_morning<-read.csv("uv_data_morning.csv")
uv_data_night<-read.csv("uv_data_night.csv" )
#clean up the data because the CSV files are little messy so start by making a global variable that is
clean_night_data<-uv_data_night[c(8, 9, 10, 11),]
clean_morning_data<-uv_data_morning[c(7, 8, 9, 10),]
print(clean_night_data)
```

```
##           X X.1 X.2 X.3 X.4
## 8  Latitude UVB UVA  NA  NA
## 9        20  78  73  NA  NA
## 10       40  75  68  NA  NA
## 11       60  69  60  NA  NA
```

```
clean_night_data<-clean_night_data[, c(1:3)]
clean_morning_data<-clean_morning_data[, c(1:3)]
#Get rid of any extra rows and columns that don't have data but that were added to our global variable.

print(clean_night_data)
```

```
##           X X.1 X.2
## 8  Latitude UVB UVA
```

```
## 9        20  78  73
## 10       40  75  68
## 11       60  69  60
```

```
print(clean_morning_data)
```

```
##           X X.1 X.2
## 7  Latitude UVB UVA
## 8        20  30  27
## 9        40  28  25
## 10       60  26  21
```

```
#print to inspect the data tables and make sure they look good.

colnames(clean_night_data)<-clean_night_data[1,]
clean_night_data<-clean_night_data[-c(1),]
colnames(clean_morning_data)<-clean_morning_data[1,]
clean_morning_data<-clean_morning_data[-c(1),]
print(clean_morning_data)
```

```
##    Latitude UVB UVA
## 8        20  30  27
## 9        40  28  25
## 10       60  26  21
```

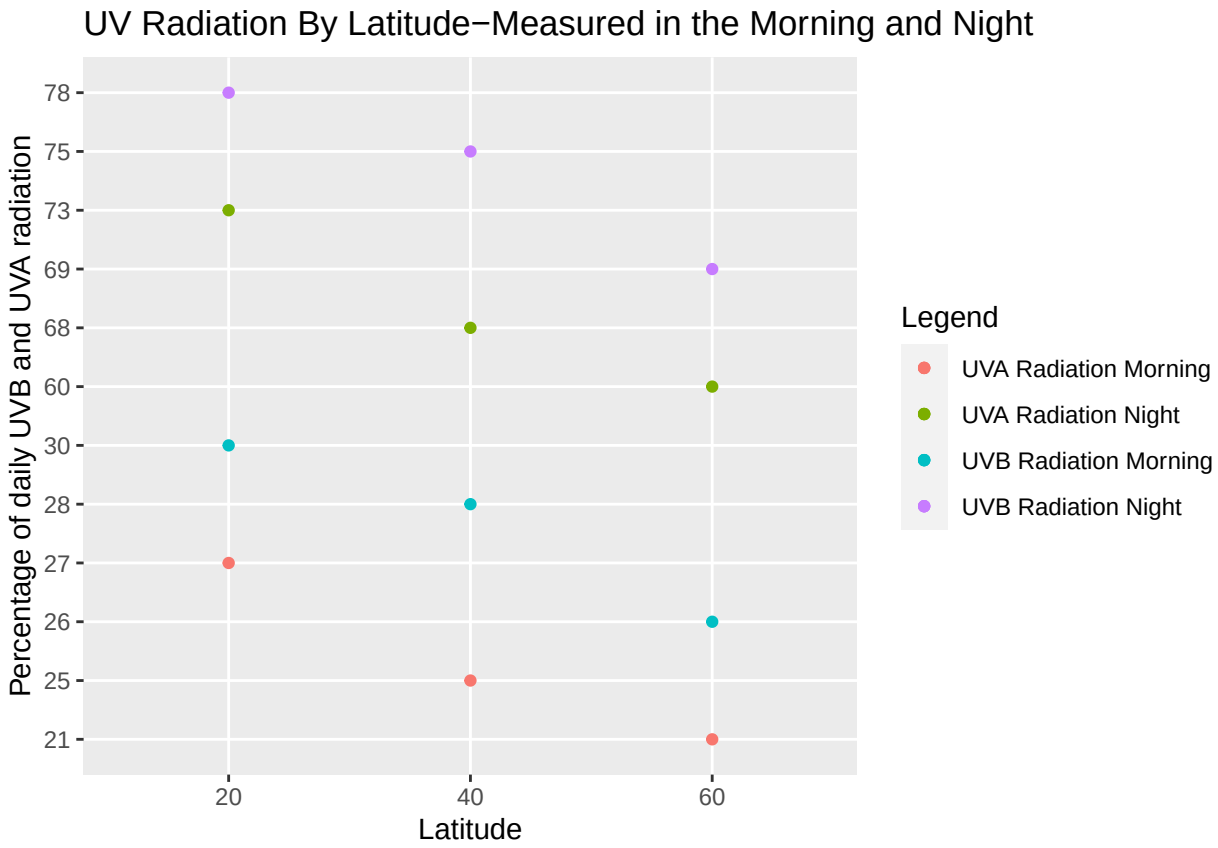
```
print(clean_night_data)
```

```
##    Latitude UVB UVA
## 9        20  78  73
## 10       40  75  68
## 11       60  69  60
```

```
#Add column names using the first row of each of the data frames. Then delete the first row since we no
#Print and inspect to make sure we didn't delete any necessary columns.

colors <- c("UVA Radiation Night" = "blue", "UVB Radiation Night" = "purple", "UVA Radiation Morning"="
#create a colors vector that assigns each condition a color so that our plots are easy to interpret.

ggplot() +
  geom_point(data = clean_morning_data, aes(x = Latitude, y = UVA, color="UVA Radiation Morning")) +
  geom_point(data = clean_morning_data, aes(x = Latitude, y = UVB, color="UVB Radiation Morning"))+
  geom_point(data=clean_night_data, aes(x=Latitude, y=UVA, color="UVA Radiation Night"))+
  geom_point(data=clean_night_data, aes(x=Latitude, y=UVB, color="UVB Radiation Night"))+ggtitle("UV Ra
```




```
#Use the ggplot function to create the dot plots. Start by calling the ggplot() and then add geom_point
```

Now we can see that there is also a nice trend showing that low latitudes have higher levels of UVA and UVB radiation.

We can see that the trends of latitude and rates of cSCC are very similar, indicating that there may be a correlation or a causitive factor behind this.

To see if our proposed explanation for why the two trends are similar, we will look at if any of the tumor p53 samples that are present in the cBioportal database show mutations that are known to come from UVA and UVB radiation.

2. Multiple Sequence Alignment

Purpose:To visualize if any of the tumor samples showed the UVA or UVB fingerprint mutations in p53. These mutations were found in tumor tissue samples of individuals with cSCC.

1. The first step is to load in the wild-type sequence of the p53 protein. This will be in all of the alignments and will be the basis of all of the mutated sequences that we generate. This was downloaded from NCBI's gene database (https://www.ncbi.nlm.nih.gov/gene/7157)

```
wt_aa_fasta<-read.fasta("protein.fa")
#print(wt_aa_fasta)
#This will load in the entire file that was downloaded from NCBI. To view the entire file, uncomment th
p53_wt_aa<-wt_aa_fasta$NP_000537.3
#print(p53_wt_aa)
#I only want the amino acid sequence of one transcript, so I will select for only one transcript and sa
p53_wt_aa<-p53_wt_aa[1:393]
print(p53_wt_aa)
```

```
##   [1] "m" "e" "e" "p" "q" "s" "d" "p" "s" "v" "e" "p" "p" "l" "s" "q" "e" "t"
##  [19] "f" "s" "d" "l" "w" "k" "l" "l" "p" "e" "n" "n" "v" "l" "s" "p" "l" "p"
##  [37] "s" "q" "a" "m" "d" "d" "l" "m" "l" "s" "p" "d" "d" "i" "e" "q" "w" "f"
##  [55] "t" "e" "d" "p" "g" "p" "d" "e" "a" "p" "r" "m" "p" "e" "a" "a" "p" "p"
##  [73] "v" "a" "p" "a" "p" "a" "a" "p" "t" "p" "a" "a" "p" "a" "p" "a" "p" "s"
##  [91] "w" "p" "l" "s" "s" "s" "v" "p" "s" "q" "k" "t" "y" "q" "g" "s" "y" "g"
## [109] "f" "r" "l" "g" "f" "l" "h" "s" "g" "t" "a" "k" "s" "v" "t" "c" "t" "y"
## [127] "s" "p" "a" "l" "n" "k" "m" "f" "c" "q" "l" "a" "k" "t" "c" "p" "v" "q"
## [145] "l" "w" "v" "d" "s" "t" "p" "p" "p" "g" "t" "r" "v" "r" "a" "m" "a" "i"
## [163] "y" "k" "q" "s" "q" "h" "m" "t" "e" "v" "v" "r" "r" "c" "p" "h" "h" "e"
## [181] "r" "c" "s" "d" "s" "d" "g" "l" "a" "p" "p" "q" "h" "l" "i" "r" "v" "e"
## [199] "g" "n" "l" "r" "v" "e" "y" "l" "d" "d" "r" "n" "t" "f" "r" "h" "s" "v"
## [217] "v" "v" "p" "y" "e" "p" "p" "e" "v" "g" "s" "d" "c" "t" "t" "i" "h" "y"
## [235] "n" "y" "m" "c" "n" "s" "s" "c" "m" "g" "g" "m" "n" "r" "r" "p" "i" "l"
## [253] "t" "i" "i" "t" "l" "e" "d" "s" "s" "g" "n" "l" "l" "g" "r" "n" "s" "f"
## [271] "e" "v" "r" "v" "c" "a" "c" "p" "g" "r" "d" "r" "r" "t" "e" "e" "e" "n"
## [289] "l" "r" "k" "k" "g" "e" "p" "h" "h" "e" "l" "p" "p" "g" "s" "t" "k" "r"
## [307] "a" "l" "p" "n" "n" "t" "s" "s" "s" "p" "q" "p" "k" "k" "k" "p" "l" "d"
## [325] "g" "e" "y" "f" "t" "l" "q" "i" "r" "g" "r" "e" "r" "f" "e" "m" "f" "r"
## [343] "e" "l" "n" "e" "a" "l" "e" "l" "k" "d" "a" "q" "a" "g" "k" "e" "p" "g"
## [361] "g" "s" "r" "a" "h" "s" "s" "h" "l" "k" "s" "k" "k" "g" "q" "s" "t" "s"
## [379] "r" "h" "k" "k" "l" "m" "f" "k" "t" "e" "g" "p" "d" "s" "d"
```

```
#Even after taking a subset of the entire file, there is still some extra info at the end of the sequen
```

2. The next step is to generate the p53 sequence with the UVA fingerprint mutations from the paper: The basal layer in human squamous tumors harbors more UVA than UVB fingerprint mutations: A role for UVA in human skin carcinogenesis by Agar et al. (PMID:15041750) Generate UVA Mutation AA Sequence

```
uv_mutations_file_table_3<-read.csv("Agar_et_al_table_3.csv")
#I downloaded figure 3 from the paper.
print(uv_mutations_file_table_3)
```

```
##                                                                                 Table.3.
## 1  Types of mutations detected in exons 5-9 p53 in the stratum granulosum of human SCC
## 2
## 3                                                                 Base change (total no.)
## 4                                                                        UVA fingerprints
## 5                                                                             AT _ CG (1)
## 6                                                                        UVB fingerprints
## 7                                                                             GC _ AT (9)
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15                                                                                ROS\xa6
## 16                                                                          GC _ TA (6)
## 17
## 18
## 19
## 20
```

```
## 21
## 22                                                                  GC _ CG (6)
## 23
## 24
## 25
## 26
## 27
## 28                                                                  GC _ AT (1)
## 29                                                                Other changes
## 30                                                                  AT _ GC (3)
## 31
## 32
## 33                                                                  AT _ TA (3)
## 34
## 35
## 36                                                                CpG sites (2)
## 37
##                X       X.1           X.2        X.3               X.4
## 1
## 2
## 3  Coding strand Codon p53      Sequence AA change* Lesion region\xa0
## 4
## 5          A _ C       311     caAca\xe0  Asn _ Thr            SCC4,1
## 6          C _ T       289      ctCc\xe0  Leu _ Leu            SCC1,1
## 7          C _ T       157      gtCc\xe0  Val _ Val            SCC1,6
## 8          C _ T       233 cCact\xe0\xa4  His _ Tyr            SCC1,3
## 9          C _ T       318  cCaa\xe0\xa4  Pro _ Leu            SCC4,1
## 10         C _ T       314      tCct\xe0  Ser _ Phe            SCC4,3
## 11         C _ T       296     tCacc\xe0  His _ Tyr            SCC4,6
## 12         C _ T       251      atCc\xe0  Ile _ Ile            SCC7,1
## 13         G _ A       197     aGtgg\xe0  Val _ Met            SCC3,1
## 14         G _ A       226      gGct\xe0  Gly _ Asp            SCC6,6
## 15
## 16         G _ T       290          cGca  Arg _ Leu            SCC4,6
## 17         C _ A       195      atCc\xe0  Ile _ Ile            SCC3,1
## 18         C _ A       208          gaCa  Asp _ Glu            SCC3,1
## 19         C _ A       312  acCa\xe0\xa4  Thr _ Thr            SCC4,1
## 20         C _ A       309  cCca\xe0\xa4  Pro _ His            SCC4,3
## 21         C _ A       313      agCt\xe0  Ser _ Arg            SCC4,3
## 22         G _ C       271     tGagg\xe0  Glu _ Gln            SCC1,3
## 23         G _ C       228     tGact\xe0  Asp _ His            SCC6,6
## 24         G _ C       326     aGaat\xe0  Glu _ Gln            SCC6,6
## 25         C _ G       190  cCtc\xe0\xa4  Pro _ Arg            SCC1,1
## 26         C _ G       252 cCtca\xe0\xa4  Leu _ Val            SCC7,1
## 27         C _ G       301     gCccc\xe0  Pro _ Ala            SCC1,3
## 28         C _ T       310          aaCa  Asn _ Asn            SCC4,1
## 29
## 30         T _ C       195      aTcc\xe0  Ile _ Thr            SCC1,1
## 31         A _ G       196      cgAg\xe0  Arg _ Arg            SCC3,1
## 32         A _ G       249     gAggc\xe0  Arg _ Gly            SCC1,1
## 33         T _ A       134     gTttt\xe0  Phe _ Ile            SCC1,6
## 34         A _ T       195         tAtcc  Ile _ Phe            SCC1,1
## 35         A _ T       310      aAca\xe0  Asn _ Ile            SCC4.3
## 36         C _ T       152  cCgc\xe0\xa4  Pro _ Leu            SCC1,6
```

```
## 37         C _ T       158 cCgcg\xe0\xa4  Arg _ Cys            SCC1,6
```

The table is pretty messy and hard to interpret, so the next steps are just reducing the table down to the information that I can use in my analysis.

```
library(dplyr)
uva_mutations_data_table_3<-select(uv_mutations_file_table_3, c('X.1', 'X.3'))
#print(uva_mutations_data_table_3, show="complete")
#At this point, uva_mutations_data_table_3 is still messy, but the previous command selects the columns
uva_mutations_final_data_table_3<-uva_mutations_data_table_3[5,]
#print(uva_mutations_final_data_table_3)
#In this table there is only one UVA mutation, and now I have that by itself with the number of the cod
#The variable uva_mutations_final_data_table_3 refers to the global variable for the table that I've ma
colnames(uva_mutations_final_data_table_3)<-c("Codon Number", "Amino Acid Change")
#print(uva_mutations_final_data_table_3)
#I am addding column names to the little mini-table that I've made from the original table, so that it
original_amino_acid<-c("N")
new_amino_acid<-c("T")
#To make my future analysis easier, I am adding the amino acid letter for the original and mutated amin
uva_mutations_final_data_table_3$Original_Amino_Acid<-original_amino_acid
#print(uva_mutations_final_data_table_3)
uva_mutations_final_data_table_3$New_Amino_Acid<-new_amino_acid
print(uva_mutations_final_data_table_3)
```

```
##   Codon Number Amino Acid Change Original_Amino_Acid New_Amino_Acid
## 5          311         Asn _ Thr                  N              T
```

Now I have cleaned up the UVA mutation data from table 3, I want to combine it with table #4 so that I can have all of the UVA mutations from the paper in one place.

```
uv_mutations_file_table_4<-read.csv("Agar_et_al_table_4.csv")
#This is table 4 from the same paper that I downloaded table 3 from in the previous chunk to do my anla
print(uv_mutations_file_table_4)
```

```
##
## 1                                                    Types of mutations detected in exons 5-
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16
## 17
## 18
## 19
## 20
## 21
```

```
## 22
## 23
## 24
## 25
## 26
## 27
## 28
## 29
## 30
## 31
## 32
## 33
## 34
## 35
## 36
## 37
## 38
## 39
## 40
## 41
## 42
## 43
## 44
## 45                                                                                                          Codon
## 46                                                                                                            *An
## 47                                                                                   \xa0Lesion number and region
## 48                             \xe0Base-pair substitutions occur at sites or runs on Py-Py/Py eit
## 49                                                                                     \xa45_G of 5_-GG-3_ on eit
## 50 \xa6Base changes potentially attributable to UVA indirectly through the production of ROS, not us
##              X       X.1           X.2        X.3               X.4
## 1
## 2
## 3  Coding strand Codon p53      Sequence AA change* Lesion/region\xa0
## 4
## 5          T _ G       238          tgTa  Cys _ Trp            SCC1,2
## 6          T _ G       257      cTgg\xe0  Leu _ Arg            SCC6,7
## 7          T _ G       220     cTatg\xe0  Tyr _ Asp            SCC8,2
## 8          T _ G       203          gTgg  Val _ Gly            SCC8,4
## 9          A _ C       320      Aaga\xe0  Lys _ Gln            SCC4,2
## 10         A _ C       331      cAgg\xe0  Gln _ Pro            SCC5,2
## 11         C _ T       315  cCtc\xe0\xa4  Ser _ Phe            SCC7,2
## 12         G _ A       293 aGggg\xe0\xa4  Gly _ Arg            SCC4,4
## 13
## 14         G _ T       305      aaGc\xe0  Lys _ Asn            SCC2,4
## 15         G _ T       283         cGcca  Arg _ Leu            SCC3,4
## 16         G _ T       317          caGc  Gln _ His            SCC4,2
## 17         C _ A       309  cCca\xe0\xa4  Pro _ His            SCC1,4
## 18         C _ A       289     tCtcc\xe0  Leu _ Met            SCC2,4
## 19         C _ A       314      tCct\xe0  Ser _ Tyr            SCC4,2
## 20         C _ A       316  ccCc\xe0\xa4  Pro _ Pro            SCC4,2
## 21         C _ A       228      gaCt\xe0  Asp _ Glu            SCC6,2
## 22         C _ A       289      ctCc\xe0  Leu _ Leu            SCC4,4
## 23         C _ A       308     aCtgc\xe0  Leu _ Met            SCC6,4
## 24         G _ C       215      aGtg\xe0  Ser _ Thr            SCC1,2
```

```
## 25         G _ C       317      caGc\xe0  Gln _ His            SCC6,4
## 26         G _ C       325      tgGa\xe0  Gly _ Ala            SCC7,2
## 27         C _ G       228      gaCt\xe0  Asp _ Glu            SCC2,4
## 28         C _ G       219  ccCt\xe0\xa4  Pro _ Pro            SCC8,2
## 29         C _ G       310          aaCa  Asn _ Lys            SCC6,4
## 30         C _ T       310          aaCa  Asn _ Asn            SCC1,4
## 31         C _ T       245          ggCa  Gly _ Gly            SCC6,2
## 32         C _ T       242          tgCa  Cys _ Cys            SCC6,7
## 33
## 34         A _ G       307          gcAc  Ala _ Ala            SCC1,4
## 35         A _ G       204      gAgt\xe0  Glu _ Gly            SCC8,4
## 36         T _ C       315     cTctc\xe0  Ser _ Pro            SCC5,2
## 37         T _ A       274      gtTt\xe0  Val _ Val            SCC5,4
## 38         T _ A       278      ccTg\xe0  Pro _ Pro            SCC4,4
## 39         A _ T       266      ggAc\xe0  Gly _ Gly            SCC2,4
## 40         A _ T       280        gAgag* Arg _ stop            SCC3,4
## 41         A _ T       281         gAcc*  Asp _ Val            SCC3,4
## 42         A _ T       311         cAaca*  Asn _ Tyr           SCC7,2
## 43         A _ T       211         cActt  Thr _ Ser            SCC8,2
## 44
## 45
## 46
## 47
## 48
## 49
## 50
```

Unfortunately, like table 3 this table is super messy and will be hard to use until it is cleaned up. I will follow the same process to generate a similar table as what I created for the UVA mutations from table 3.

```
uva_mutations_data_table_4<-select(uv_mutations_file_table_4, c('X.1', 'X.3'))
uva_mutations_data_table_4<-uva_mutations_data_table_4[(5:10),]
#print(uva_mutations_data_table_4)
#uva_mutations_data_table_4 is the global variable that I am assigning to the rough data. Using select
colnames(uva_mutations_data_table_4)<-c("Codon Number", "Amino Acid Change")
#print(uva_mutations_data_table_4)
#Adding column names to make easier to subset in the future.
original_amino_acid_uva_4<-c("C", "L", "Y", "V", "K", "Q")
new_amino_acid_uva_4<-c("W", "R", "D", "G", "Q", "P")
#I am creating global variables to store the amino acid letters for the original codon and the mutation
uva_mutations_data_table_4$Original_Amino_Acid<-original_amino_acid_uva_4
uva_mutations_data_table_4$New_Amino_Acid<-new_amino_acid_uva_4
print(uva_mutations_data_table_4)
```

```
##    Codon Number Amino Acid Change Original_Amino_Acid New_Amino_Acid
## 5           238         Cys _ Trp                  C              W
## 6           257         Leu _ Arg                  L              R
## 7           220         Tyr _ Asp                  Y              D
## 8           203         Val _ Gly                  V              G
## 9           320         Lys _ Gln                  K              Q
## 10          331         Gln _ Pro                  Q              P
```

```
#These commands append the amino acid variables to the table.
```

Now I have the UVA induced fingerprint mutations. To help generate the mutated amino acid sequence, I will combine these tables into one so that I have all of the data for the UVA mutations in a single dataframe.

```
uva_mutations_final <- rbind(uva_mutations_data_table_4, uva_mutations_final_data_table_3)
print(uva_mutations_final)
```

```
##    Codon Number Amino Acid Change Original_Amino_Acid New_Amino_Acid
## 5           238         Cys _ Trp                  C              W
## 6           257         Leu _ Arg                  L              R
## 7           220         Tyr _ Asp                  Y              D
## 8           203         Val _ Gly                  V              G
## 9           320         Lys _ Gln                  K              Q
## 10          331         Gln _ Pro                  Q              P
## 51          311         Asn _ Thr                  N              T
```

Using the rbind function, I was able to combine both of the uva mutation tables into a single table, which I am calling uva_mutations_final. This tabel has data for all 7 of the UVA fingerprint mutations. Next I need to create the amino acid sequence with the UVA mutations.

```
uva_mutations_codon<-uva_mutations_final$`Codon Number`
print(class(uva_mutations_codon))
```

```
## [1] "character"
```

```
#I am creating a global variable called uva_mutations_codon that will have the numbers of the codons wi
numeric_uva_mutations_codon<-as.numeric(uva_mutations_codon)
print(class(numeric_uva_mutations_codon))
```

```
## [1] "numeric"
```

```
print(uva_mutations_codon)
```

```
## [1] "238" "257" "220" "203" "320" "331" "311"
```

```
#To use the replace function, I need the codon numbers to be numeric and not characters. Since each num
uva_mutations_replacement<-uva_mutations_final$New_Amino_Acid
print((uva_mutations_replacement))
```

```
## [1] "W" "R" "D" "G" "Q" "P" "T"
```

```
#I am subsetting the uva_mutations_final table that I created to get the column that had the mutated am
```

```
uva_mutations_sequence<-replace(p53_wt_aa, numeric_uva_mutations_codon, uva_mutations_replacement)
```

I am using the replace function to create my sequence with the uva fingerprint mutations! It works by taking the input, in this case the wild-type amino acid sequence, and then uses a numeric vector to look for the number corresponding to the number of the input sequence that needs to be changed. Then, it uses the final argument to know what to change to. So in our case, it uses the numeric_uva_mutations_codon to find all of the numebrs of codons that need to be changed, and uses the letters in the uva_mutations_replacement as the replacement.

```
#print(uva_mutations_sequence)
#print(p53_wt_aa)
#Since there aren't very many mutations, printing out the wild type and mutated sequence isn't really g
length(uva_mutations_sequence)
```

```
## [1] 393
```

```
length(p53_wt_aa)
```

```
## [1] 393
```

```
#This means we can use the lenght of the original and the new mutated sequence to check to make sure ou
```

Since the lengths of the sequences are the same, it looks like our replace call worked!! Now we have a sequence with all of the fingerprint UVA mutations from the Agar et al paper, and now we will repeat these steps to generate a sequence with the UVB mutations.

Generate UVB Mutations Sequence

```
uv_mutations_file_table_3<-read.csv("Agar_et_al_table_3.csv")
#We already had this line of code in the section where we made the sequence with uva mutations, but I a
#print(uv_mutations_file_table_3)
#uv_mutations_file_table_3 is the
uvb_mutations_data_table_3<-uv_mutations_file_table_3[(6:14),]
#print(uvb_mutations_data_table_3)
final_uvb_mutations_data_table_3<-select(uvb_mutations_data_table_3, c('X.1', 'X.3'))
#print(final_uvb_mutations_data_table_3)
#These steps are to clean up the table so we can clearly see the information we are interested in. I cr
colnames(final_uvb_mutations_data_table_3)<-c("Codon Number", "Amino Acid Change")
#Adding column names so that it will be easier to subset in the future.
#print(final_uvb_mutations_data_table_3)
original_amino_acid_uvb_3<-c("L", "V", "H", "P", "S", "H", "I", "V", "G")
new_amino_acid_uvb_3<-c("L", "V", "Y", "L", "F", "Y", "I", "M", "D")
#Using the data in our final_uvb_mutations_data_table_3 to add the one amino acid letters that we will

final_uvb_mutations_data_table_3$Original_Amino_Acid<-original_amino_acid_uvb_3
final_uvb_mutations_data_table_3$New_Amino_Acid<-new_amino_acid_uvb_3
#Adding the vectors we made with the amino acid info to our dataframe.
print(final_uvb_mutations_data_table_3)
```

```
##    Codon Number Amino Acid Change Original_Amino_Acid New_Amino_Acid
## 6           289         Leu _ Leu                  L              L
## 7           157         Val _ Val                  V              V
## 8           233         His _ Tyr                  H              Y
## 9           318         Pro _ Leu                  P              L
## 10          314         Ser _ Phe                  S              F
## 11          296         His _ Tyr                  H              Y
## 12          251         Ile _ Ile                  I              I
## 13          197         Val _ Met                  V              M
## 14          226         Gly _ Asp                  G              D
```

```
#Check the final table.
```

Nice! Looks clean, now we have to do this one more time with the uvb mutations from the other table.

```
uvb_mutations_data_table_4<-uv_mutations_file_table_4[(11:12),]
final_uvb_mutations_data_table_4<-select(uvb_mutations_data_table_4, c('X.1', 'X.3'))
#print(final_uvb_mutations_data_table_4)
#We already created the uv_mutations_file_table_4 earlier when we read in the csv files.
#Clean up the data and save it as a new global variable that we will continue to modify until it matche

colnames(final_uvb_mutations_data_table_4)<-c("Codon Number", "Amino Acid Change")
#print(final_uvb_mutations_data_table_4)
#Add the proper column names.

original_amino_acid_uvb_4<-c("S", "G")
new_amino_acid_uvb_4<-c("F", "R")
```

```
final_uvb_mutations_data_table_4$Original_Amino_Acid<-original_amino_acid_uvb_4
final_uvb_mutations_data_table_4$New_Amino_Acid<-new_amino_acid_uvb_4
#Create the vectors with the amino acid info.
#Add the amino acid vectors we just made to the data frame so that we can use this in our replace funct
print(final_uvb_mutations_data_table_4)
```

```
##    Codon Number Amino Acid Change Original_Amino_Acid New_Amino_Acid
## 11          315         Ser _ Phe                   S              F
## 12          293         Gly _ Arg                   G              R
```

Now that we have the two tables that have the information from the uvb mutations from both tables, we are ready to combine them and use them in our replace function to generate the p53 amino acid sequence with the proper mutations.

```
uvb_mutations_final <- rbind(final_uvb_mutations_data_table_3, final_uvb_mutations_data_table_4)
print(uvb_mutations_final)
```

```
##     Codon Number Amino Acid Change Original_Amino_Acid New_Amino_Acid
## 6            289         Leu _ Leu                   L              L
## 7            157         Val _ Val                   V              V
## 8            233         His _ Tyr                   H              Y
## 9            318         Pro _ Leu                   P              L
## 10           314         Ser _ Phe                   S              F
## 11           296         His _ Tyr                   H              Y
## 12           251         Ile _ Ile                   I              I
## 13           197         Val _ Met                   V              M
## 14           226         Gly _ Asp                   G              D
## 111          315         Ser _ Phe                   S              F
## 121          293         Gly _ Arg                   G              R
```

```
#Combine our two uvb tables into one dataframe so we can have all of the info in the same place. This i
uvb_mutations_codon<-uvb_mutations_final$`Codon Number`
numeric_uvb_mutations_codon<-as.numeric(uvb_mutations_codon)
print(class(numeric_uvb_mutations_codon))
```

```
## [1] "numeric"
```

```
#We will use the new numeric_uvb_mutations_codon global variable in our replace function, but we need t
uvb_mutations_replacement<-uvb_mutations_final$New_Amino_Acid
#print((uvb_mutations_replacement))
#We need to subset our dataframe and create a new global variable uvb_mutations_replacement that has th

uvb_mutations_sequence<-replace(p53_wt_aa, numeric_uvb_mutations_codon, uvb_mutations_replacement)
length(uvb_mutations_sequence)
```

```
## [1] 393
```

```
length(p53_wt_aa)
```

```
## [1] 393
```

After using the replace function, we can check to make sure it worked and that it truly replaced and didn't add anything extra by checking the length of the sequences. Since the lenghts are the same we can move on to the actual alignmetns!! *Note: I tried to make a function that would perform the code for each subsection that we needed, but this was impossible. The formatting of the tables in the paper were super funky and since the mutations for each category were random and not in the same place it was just easier to do it this way.

3. Create Alignment with Wild-Type, UVA, and UVB mutations

```
write.fasta(sequences=p53_wt_aa, names="p53_wt_aa", file.out="p53_wt_aa.fasta")
#Generate fasta file with wild-type p53 sequence.
write.fasta(sequences=uva_mutations_sequence, names="uva_mutations", file.out="uva_mutations_sequence.fa
#Generate fasta file with p53 containing UVA signature mutations.
write.fasta(sequences=uvb_mutations_sequence, names="uvb_mutations", file.out="uvb_mutations_sequence.fa
#Generate fasta file with p53 containing UVB signature mutations.
```

In order to perform the alignments using the msa packages, all of the sequences need to be in the same file. In order to do this I am usign the write.fasta function to write all of them as individual fasta files that save to my working directory.

```
fastaconc(otus=c('p53_wt_aa', 'uva_mutations_sequence', 'uvb_mutations_sequence'), inputdir = ".", out.
```

```
## [1] "Work finished. Fasta file saved at ./wt_uva_uvb_clean.fasta"
```

In order to generate a single file with the sequences to align, I am using the fastaconc function to write a new file, wt_uva_uvb_clean.fasta that will be used for the alignment.

```
wt_uva_uvb<-readAAStringSet("wt_uva_uvb_clean.fasta")
#The input of the msa function has to be a AAStringSet, DNAStringSet, or RNAStringSet. I am using readA
wt_uva_uvb_alignment<-msa(wt_uva_uvb, order="input")
```

```
## use default substitution matrix
```

```
#This is the code that performs a basic alignment.
print(wt_uva_uvb_alignment, show="complete")
```

```
##
## MsaAAMultipleAlignment with 3 rows and 393 columns
##     aln (1..54)                                              names
## [1] MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQAMDDLMLSPDDIEQWF p53_wt_aa
## [2] MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQAMDDLMLSPDDIEQWF uva_mutations_seq...
## [3] MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQAMDDLMLSPDDIEQWF uvb_mutations_seq...
## Con MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQAMDDLMLSPDDIEQWF Consensus
##
##     aln (55..108)                                            names
## [1] TEDPGPDEAPRMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYG p53_wt_aa
## [2] TEDPGPDEAPRMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYG uva_mutations_seq...
## [3] TEDPGPDEAPRMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYG uvb_mutations_seq...
## Con TEDPGPDEAPRMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYG Consensus
##
##     aln (109..162)                                           names
## [1] FRLGFLHSGTAKSVTCTYSPALNKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAI p53_wt_aa
## [2] FRLGFLHSGTAKSVTCTYSPALNKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAI uva_mutations_seq...
## [3] FRLGFLHSGTAKSVTCTYSPALNKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAI uvb_mutations_seq...
## Con FRLGFLHSGTAKSVTCTYSPALNKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAI Consensus
##
##     aln (163..216)                                           names
## [1] YKQSQHMTEVVRRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFRHSV p53_wt_aa
## [2] YKQSQHMTEVVRRCPHHERCSDSDGLAPPQHLIRVEGNLRGEYLDDRNTFRHSV uva_mutations_seq...
## [3] YKQSQHMTEVVRRCPHHERCSDSDGLAPPQHLIRMEGNLRVEYLDDRNTFRHSV uvb_mutations_seq...
## Con YKQSQHMTEVVRRCPHHERCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFRHSV Consensus
##
##     aln (217..270)                                           names
## [1] VVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTIITLEDSSGNLLGRNSF p53_wt_aa
```

```
## [2] VVPDEPPEVGSDCTTIHYNYMWNSSCMGGMNRRPILTIITREDSSGNLLGRNSF uva_mutations_seq...
## [3] VVPYEPPEVDSDCTTIYYNYMCNSSCMGGMNRRPILTIITLEDSSGNLLGRNSF uvb_mutations_seq...
## Con VVPYEPPEVGSDCTTIHYNYMCNSSCMGGMNRRPILTIITLEDSSGNLLGRNSF Consensus
##
##     aln (271..324)                                          names
## [1] EVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKRALPNNTSSSPQPKKKPLD p53_wt_aa
## [2] EVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKRALPNTTSSSPQPKQKPLD uva_mutations_seq...
## [3] EVRVCACPGRDRRTEEENLRKKREPYHELPPGSTKRALPNNTSFFPQLKKKPLD uvb_mutations_seq...
## Con EVRVCACPGRDRRTEEENLRKKGEPHHELPPGSTKRALPNNTSSSPQPKKKPLD Consensus
##
##     aln (325..378)                                          names
## [1] GEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKSKKGQSTS p53_wt_aa
## [2] GEYFTLPIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKSKKGQSTS uva_mutations_seq...
## [3] GEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKSKKGQSTS uvb_mutations_seq...
## Con GEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPGGSRAHSSHLKSKKGQSTS Consensus
##
##     aln (379..393)  names
## [1] RHKKLMFKTEGPDSD p53_wt_aa
## [2] RHKKLMFKTEGPDSD uva_mutations_seq...
## [3] RHKKLMFKTEGPDSD uvb_mutations_seq...
## Con RHKKLMFKTEGPDSD Consensus
```

Our alignment worked, but it is hard to see which amino acids are different and overall it's not easy to draw any conclusions from.

```
#msaPrettyPrint(wt_uva_uvb_alignment, output="pdf", showNames="left", showLogo="none", showConsensus="n
```

This code uses the msaPrettyPrint that generates a much prettier alignment that is a lot easier to read. I didn't want there to be a consensus, which is why I had the argument showConsensus="none"

*Note:Running this code generates an error because my computer struggles to use LaTeX. It generates a PDF, but puts up an error so I am commenting it out and will be knitting in the pdf that was generated from this line to avoid the error.

```
pdf_convert("wt_uva_uvb_alignment.pdf", format = "png", pages = NULL, filenames = NULL, dpi = 300, opw =
```

```
## Converting page 1 to wt_uva_uvb_alignment_1.png... done!
```

```
## [1] "wt_uva_uvb_alignment_1.png"
```

I am using the pdf_convert to convert the pdf that was generated by the msaPrettyPrint function into a .png image.

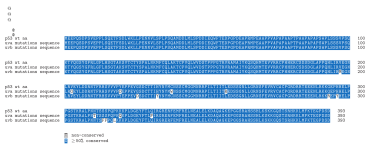


Figure 1: your caption

This chunk of code allows me to display the results of the msaPrettyPrint. It shows the .png that we generated in the chunk above.

This alignment shows the wild-type p53 protein sequence along with the UVA and UVB fingerprint mutations. I wanted to include this alignment to highlight the mutations that form in this gene as a result of UV radiation. As you can see there are 7 UVA mutations and 8 UVB mutations.

4.Create Mutant Sequences. The data for this section came from the cBioportal for Cancer Genomics (https://www.cbioportal.org/)

Generating 2015 Mutant Sequence

```
#DFCI, Clin Cancer Res 2015 Mutations
bioportal_2015_mutations<-read.delim("bioportal_2015_mutations.tsv", header=TRUE, sep="\t")
#print(bioportal_2015_mutations)
#I downlaoded the .tsv file directly from the bioportal website by selecting the 2015 samples and click

clean_2015_mutations<-bioportal_2015_mutations[,c(1, 5)]
#print(clean_2015_mutations)
#This line creates a new table using 2 columns from the original bioportal_2015_mutations.
raw_codon_data_2015<-c(clean_2015_mutations[,2])
print(raw_codon_data_2015)
```

```
##  [1] "E286K"       "E286K"       "R248W"       "G245D"       "R248Q"
##  [6] "R282W"       "R282W"       "P278S"       "S241F"       "S241F"
## [11] "R248L"       "V157F"       "G266R"       "V197E"       "R267P"
## [16] "R213*"       "E271K"       "Y205C"       "X187_splice" "V218G"
## [21] "X187_splice" "P152S"       "G279E"       "R342*"       "R196*"
## [26] "Q192*"       "L252P"       "W146*"       "W91*"        "W91*"
## [31] "R290C"       "R156P"       "W146*"       "S183*"       "P13S"
```

```
#This creates a new list of the information we need. In this table, the authors decided to give informa
#Original AA - Codon Number - Mutated AA


logical_non_splice_2015<-nchar(raw_codon_data_2015)==5
print(logical_non_splice_2015)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
## [25]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
```

```
#creates a list as a new global variable that says true for entries that are 5 characters and false for
index_non_splice_2015<-which(logical_non_splice_2015, arr.ind=FALSE)
print(index_non_splice_2015)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 20 22 23 24 25 26 27
## [26] 28 31 32 33 34
```

```
#The only kinds of mutations that were looking at are point mutations, we don't want to look at deletio

list_without_splice_2015<-raw_codon_data_2015[index_non_splice_2015]
print(list_without_splice_2015)
```

```
##  [1] "E286K" "E286K" "R248W" "G245D" "R248Q" "R282W" "R282W" "P278S" "S241F"
## [10] "S241F" "R248L" "V157F" "G266R" "V197E" "R267P" "R213*" "E271K" "Y205C"
## [19] "V218G" "P152S" "G279E" "R342*" "R196*" "Q192*" "L252P" "W146*" "R290C"
## [28] "R156P" "W146*" "S183*"
```

```
#This code uses the list that we made in the previous lines to select only the mutations that have 5 le
```

```
remove_asterisk_2015<-gsub("[[:punct:]]", "", list_without_splice_2015)
print(remove_asterisk_2015)
```

```
##  [1] "E286K" "E286K" "R248W" "G245D" "R248Q" "R282W" "R282W" "P278S" "S241F"
## [10] "S241F" "R248L" "V157F" "G266R" "V197E" "R267P" "R213"  "E271K" "Y205C"
## [19] "V218G" "P152S" "G279E" "R342"  "R196"  "Q192"  "L252P" "W146"  "R290C"
## [28] "R156P" "W146"  "S183"
```

```
#Mutations that result in deletions are written as AA-Codon Number-*. This line of code gets rid of any
logical_without_deletions_2015<-nchar(remove_asterisk_2015)==5
print(logical_without_deletions_2015)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [25]  TRUE FALSE  TRUE  TRUE FALSE FALSE
```

```
index_non_deletions_2015<-which(logical_without_deletions_2015, arr.ind=FALSE)
print(index_non_deletions_2015)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 17 18 19 20 21 25 27 28
```

```
#print(index_non_deletions_2015)
#Now we are creating another index that will say which of the entries in our list have5 letters, or are


list_mutations_2015<-list_without_splice_2015[index_non_deletions_2015]
print(list_mutations_2015)
```

```
##  [1] "E286K" "E286K" "R248W" "G245D" "R248Q" "R282W" "R282W" "P278S" "S241F"
## [10] "S241F" "R248L" "V157F" "G266R" "V197E" "R267P" "E271K" "Y205C" "V218G"
## [19] "P152S" "G279E" "L252P" "R290C" "R156P"
```

```
#Use the list that we generated with the which function to subset the list of mutations without the spl

codon_number_messy_2015<-substring(list_mutations_2015 ,2)
print(codon_number_messy_2015)
```

```
##  [1] "286K" "286K" "248W" "245D" "248Q" "282W" "282W" "278S" "241F" "241F"
## [11] "248L" "157F" "266R" "197E" "267P" "271K" "205C" "218G" "152S" "279E"
## [21] "252P" "290C" "156P"
```

```
#I want to generate a list of numbers of which codons were mutated, and this line creates a new global

clean_codon_number_2015<-substring(codon_number_messy_2015, 1, 3)
print(clean_codon_number_2015)
```

```
##  [1] "286" "286" "248" "245" "248" "282" "282" "278" "241" "241" "248" "157"
## [13] "266" "197" "267" "271" "205" "218" "152" "279" "252" "290" "156"
```

```
#This call of the substring function removes the final letter from each entry so that it is just a list
clean_codon_number_2015<-clean_codon_number_2015[-c(2, 5, 6, 10)]
#Removing duplicated mutations, since this will not work in our replace function.
print(clean_codon_number_2015)
```

```
##  [1] "286" "248" "245" "282" "278" "241" "248" "157" "266" "197" "267" "271"
## [13] "205" "218" "152" "279" "252" "290" "156"
```

```
#this is a list of all of the codons in this dataset that had point mutations.
#This tells us how many codons we will be mutating in our sequence.
```

```
class(clean_codon_number_2015)
```

```
## [1] "character"
```

```
numeric_codon_number_2015<-as.numeric(clean_codon_number_2015)
print(numeric_codon_number_2015)
```

```
##  [1] 286 248 245 282 278 241 248 157 266 197 267 271 205 218 152 279 252 290 156
```

```
#Like in previous sections, for the replace function we need all of the codon numbers to be numeric, so

mutated_aa_2015<-substring(list_mutations_2015, 5)
print(mutated_aa_2015)
```

```
##  [1] "K" "K" "W" "D" "Q" "W" "W" "S" "F" "F" "L" "F" "R" "E" "P" "K" "C" "G" "S"
## [20] "E" "P" "C" "P"
```

```
#Mutated_aa_2015 is a new global variable that is a list of all of the letters of the mutated amino aci
clean_mutated_aa_2015<-mutated_aa_2015[-c(2, 5, 6, 10)]
#Removing the replicates that we removed from the numbers.
print(mutated_aa_2015)
```

```
##  [1] "K" "K" "W" "D" "Q" "W" "W" "S" "F" "F" "L" "F" "R" "E" "P" "K" "C" "G" "S"
## [20] "E" "P" "C" "P"
```

```
length(clean_mutated_aa_2015)
```

```
## [1] 19
```

```
length(clean_codon_number_2015)
```

```
## [1] 19
```

```
#Checking the length helps us be sure that our cleanup was succesful and that these global variables ar

mutations_2015_sequence<-replace(p53_wt_aa, numeric_codon_number_2015, clean_mutated_aa_2015)
#Using the variables that we created earlier, we are once again using the replace function to generate

#print(mutations_2015_sequence)
#print(p53_wt_aa)
length(mutations_2015_sequence)
```

```
## [1] 393
```

```
length(p53_wt_aa)
```

```
## [1] 393
```

After checking the length we can see that our new sequence still has the same numebr of amino acids as the original, so our replace worked!!

The next two sections follow the same pipeline as the section above for designing the 2015 mutant sequence.
Generating 2014 Mutant Sequence

```
#MD Anderson, Clin Cancer Res 2014
bioportal_2014_mutations<-read.delim("Clin_Cancer_Res 2014.tsv", header=TRUE, sep="\t")
#print(bioportal_2014_mutations)
#I downlaoded the .tsv file directly from the bioportal website by selecting the 2014 samples and click
clean_2014_mutations<-bioportal_2014_mutations[,c(1, 5)]
print(clean_2014_mutations)
```

```
##                                                                 Study.of.Origin
## 1  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 2  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 3  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 4  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 5  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 6  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 7  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 8  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 9  Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 10 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 11 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 12 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 13 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 14 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 15 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 16 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 17 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 18 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 19 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 20 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 21 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 22 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 23 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 24 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 25 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 26 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 27 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 28 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 29 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 30 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 31 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 32 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 33 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 34 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 35 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 36 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 37 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 38 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 39 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 40 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 41 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 42 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 43 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 44 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 45 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 46 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 47 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 48 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 49 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 50 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 51 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 52 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 53 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
```

```
## 54 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 55 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 56 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 57 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
## 58 Cutaneous Squamous Cell Carcinoma (MD Anderson, Clin Cancer Res 2014)
##    Protein.Change
## 1           E286K
## 2           E286K
## 3           R248W
## 4           R248W
## 5           R248W
## 6           R280K
## 7           H179Y
## 8           P278S
## 9           S241F
## 10          R249S
## 11          R248W
## 12          R248W
## 13          R248W
## 14          H179Y
## 15          R248Q
## 16          R282W
## 17          A159V
## 18          G266R
## 19          G266R
## 20          A138V
## 21          R213*
## 22          R213*
## 23          P278T
## 24          V272G
## 25          Y236N
## 26          F270I
## 27    X225_splice
## 28    X187_splice
## 29     X33_splice
## 30          V216G
## 31          S127F
## 32          T125=
## 33    X187_splice
## 34    X187_splice
## 35    X125_splice
## 36          R110C
## 37          G279E
## 38          G279E
## 39      R209Kfs*6
## 40          R342*
## 41          R196*
## 42     R110Vfs*13
## 43          Q331*
## 44           W91*
## 45          Y107*
## 46          Q104*
## 47          Q104*
## 48          E298*
```

```
## 49          Q317*
## 50          W146*
## 51     P153Afs*28
## 52      P58Qfs*65
## 53     V157Pfs*23
## 54           W23*
## 55      S362Afs*8
## 56          P142S
## 57           E11K
## 58           P80S
```

```
#This line creates a new table using 2 columns from the original bioportal_2014_mutations since a lot o
raw_codon_data_2014<-c(clean_2014_mutations[,2])
print(raw_codon_data_2014)
```

```
##  [1] "E286K"       "E286K"       "R248W"       "R248W"       "R248W"
##  [6] "R280K"       "H179Y"       "P278S"       "S241F"       "R249S"
## [11] "R248W"       "R248W"       "R248W"       "H179Y"       "R248Q"
## [16] "R282W"       "A159V"       "G266R"       "G266R"       "A138V"
## [21] "R213*"       "R213*"       "P278T"       "V272G"       "Y236N"
## [26] "F270I"       "X225_splice" "X187_splice" "X33_splice"  "V216G"
## [31] "S127F"       "T125="       "X187_splice" "X187_splice" "X125_splice"
## [36] "R110C"       "G279E"       "G279E"       "R209Kfs*6"   "R342*"
## [41] "R196*"       "R110Vfs*13"  "Q331*"       "W91*"        "Y107*"
## [46] "Q104*"       "Q104*"       "E298*"       "Q317*"       "W146*"
## [51] "P153Afs*28"  "P58Qfs*65"   "V157Pfs*23"  "W23*"        "S362Afs*8"
## [56] "P142S"       "E11K"        "P80S"
```

```
#I am creating a new global variable from the Protein.Change colum of the table that will create a list
logical_non_splice_2014<-nchar(raw_codon_data_2014)==5
print(logical_non_splice_2015)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
## [25]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
```

```
#create a list as a new global variable that says true for entries that are 5 characters and false for
index_non_splice_2014<-which(logical_non_splice_2014, arr.ind=FALSE)
#print(index_non_splice_2014)
list_without_splice_2014<-raw_codon_data_2014[index_non_splice_2014]
print(list_without_splice_2014)
```

```
##  [1] "E286K" "E286K" "R248W" "R248W" "R248W" "R280K" "H179Y" "P278S" "S241F"
## [10] "R249S" "R248W" "R248W" "R248W" "H179Y" "R248Q" "R282W" "A159V" "G266R"
## [19] "G266R" "A138V" "R213*" "R213*" "P278T" "V272G" "Y236N" "F270I" "V216G"
## [28] "S127F" "T125=" "R110C" "G279E" "G279E" "R342*" "R196*" "Q331*" "Y107*"
## [37] "Q104*" "Q104*" "E298*" "Q317*" "W146*" "P142S"
```

```
#The only kinds of mutations that were looking at are point mutations, we don't want to look at deletio

remove_asterisk_2014<-gsub("[[:punct:]]", "", list_without_splice_2014)
#Mutations that result in deletions are written as AA-Codon Number-*. This line of code gets rid of any

logical_without_deletions_2014<-nchar(remove_asterisk_2014)==5
index_non_deletions_2014<-which(logical_without_deletions_2014, arr.ind=FALSE)
print(index_non_deletions_2014)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 23 24 25 26 27
## [26] 28 30 31 32 42
```

```
#Now we are creating another index that will say which of the entries in our list have5 letters, or are
list_mutations_2014<-list_without_splice_2014[index_non_deletions_2014]
print(list_mutations_2014)
```

```
##  [1] "E286K" "E286K" "R248W" "R248W" "R248W" "R280K" "H179Y" "P278S" "S241F"
## [10] "R249S" "R248W" "R248W" "R248W" "H179Y" "R248Q" "R282W" "A159V" "G266R"
## [19] "G266R" "A138V" "P278T" "V272G" "Y236N" "F270I" "V216G" "S127F" "R110C"
## [28] "G279E" "G279E" "P142S"
```

```
#Use the list that we generated with the which function to subset the list of mutations without the spl
codon_number_messy_2014<-substring(list_mutations_2014 ,2)
clean_codon_number_2014<-substring(codon_number_messy_2014, 1, 3)
print(codon_number_messy_2014)
```

```
##  [1] "286K" "286K" "248W" "248W" "248W" "280K" "179Y" "278S" "241F" "249S"
## [11] "248W" "248W" "248W" "179Y" "248Q" "282W" "159V" "266R" "266R" "138V"
## [21] "278T" "272G" "236N" "270I" "216G" "127F" "110C" "279E" "279E" "142S"
```

```
#I want to generate a list of numbers of which codons were mutated, and this line creates a new global
#I also want to get rid of the final letter so that I only have the codon number for my replace functio
clean_codon_number_2014<-clean_codon_number_2014[-c(2, 4, 5, 11, 12, 13, 15, 28)]
#print(clean_codon_number_2014)

#After printing the results of clean_codon_number_2014 I am removing any duplications that might otherw

class(clean_codon_number_2014)
```

```
## [1] "character"
```

```
numeric_codon_number_2014<-as.numeric(clean_codon_number_2014)
print(numeric_codon_number_2014)
```

```
##  [1] 286 248 280 179 278 241 249 179 282 159 266 266 138 278 272 236 270 216 127
## [20] 110 279 142
```

```
#To use the replace function the numbers that tell the function which amino acids to replace have to be

mutated_aa_2014<-substring(list_mutations_2014, 5)
#Taking only the 5th letter of the entries, which creates a list of only the amino acid that is present
clean_mutated_aa_2014<-mutated_aa_2014[-c(2, 4, 5, 11, 12, 13, 15, 28)]
#Remove  duplications that might otherwise confuse the replace function.

length(clean_codon_number_2014)
```

```
## [1] 22
```

```
length(clean_mutated_aa_2014)
```

```
## [1] 22
```

```
#Checking the length helps us be sure that our cleanup was succesful and that these global variables ar

mutations_2014_sequence<-replace(p53_wt_aa, numeric_codon_number_2014, clean_mutated_aa_2014)
#Using the variables that we created earlier, we are once again using the replace function to generate

#print(mutations_2014_sequence)
```

```
#print(p53_wt_aa)
length(mutations_2014_sequence)
```

```
## [1] 393
```

```
length(p53_wt_aa)
```

```
## [1] 393
```

After checking the length we can see that our new sequence still has the same numebr of amino acids as the original, so our replace worked!!

Generating 2021 Mutant Sequence (following the same steps as the two previous samples)

```
bioportal_2021_mutations<-read.delim("UCSF_NPJ_Genom_Med_2021.tsv" , header=TRUE, sep="\t")
#print(bioportal_2021_mutations)
#I downlaoded the .tsv file directly from the bioportal website by selecting the 2021 samples and click
clean_2021_mutations<-bioportal_2021_mutations[,c(1, 5)]
raw_codon_data_2021<-c(clean_2021_mutations[,2])
#print(raw_codon_data_2021)
#Cleaning up the data and generating a new global variable that has entries for each of the mutations f

logical_non_splice_2021<-nchar(raw_codon_data_2021)==5
print(logical_non_splice_2021)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [37] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [61]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
```

```
#create a list as a new global variable that says true for entries that are 5 characters and false for
index_non_splice_2021<-which(logical_non_splice_2021, arr.ind=FALSE)
#print(index_non_splice_2021)

list_without_splice_2021<-raw_codon_data_2021[index_non_splice_2021]
print(list_without_splice_2021)
```

```
##  [1] "H179Q" "E285K" "E285K" "R273H" "E286K" "E286K" "E286K" "R248W" "G245D"
## [10] "G245D" "C238Y" "C176F" "H179Y" "H179Y" "V173L" "R248Q" "R248Q" "R248Q"
## [19] "R248Q" "M237I" "P278S" "R248W" "H179Y" "R282W" "P151H" "G266R" "P151H"
## [28] "M133K" "V173M" "V173M" "P278L" "P278L" "H179N" "C238W" "C238W" "L194F"
## [37] "G266E" "R267G" "S127Y" "E258K" "H214Y" "E224E" "Y220H" "T125T" "F341S"
## [46] "G279E" "G279E" "P177S" "R196*" "R196*" "L145R" "Q104*" "E349*" "Y107*"
## [55] "Q317*" "Q317*" "Y220*" "P142F" "P47fs" "T329I"
```

```
#The only kinds of mutations that were looking at are point mutations, we don't want to look at deletio
remove_asterisk_2021<-gsub("[[:punct:]]", "", list_without_splice_2021)
#Mutations that result in deletions are written as AA-Codon Number-*. This line of code gets rid of any

logical_without_deletions_2021<-nchar(remove_asterisk_2021)==5
print(logical_without_deletions_2021)
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
## [49] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

```
#After removing the final asterisk in any deletions, create a list that says true for mutations that ar

index_non_deletions_2021<-which(logical_without_deletions_2021, arr.ind=FALSE)
print(index_non_deletions_2021)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 51 58
## [51] 59 60
```

```
#Now we are creating another index that will say which of the entries in our list have5 letters, or are
list_mutations_2021<-list_without_splice_2021[index_non_deletions_2021]
print(list_mutations_2021)
```

```
##  [1] "H179Q" "E285K" "E285K" "R273H" "E286K" "E286K" "E286K" "R248W" "G245D"
## [10] "G245D" "C238Y" "C176F" "H179Y" "H179Y" "V173L" "R248Q" "R248Q" "R248Q"
## [19] "R248Q" "M237I" "P278S" "R248W" "H179Y" "R282W" "P151H" "G266R" "P151H"
## [28] "M133K" "V173M" "V173M" "P278L" "P278L" "H179N" "C238W" "C238W" "L194F"
## [37] "G266E" "R267G" "S127Y" "E258K" "H214Y" "E224E" "Y220H" "T125T" "F341S"
## [46] "G279E" "G279E" "P177S" "L145R" "P142F" "P47fs" "T329I"
```

```
#Use the list that we generated with the which function to subset the list of mutations without the spl

codon_number_messy_2021<-substring(list_mutations_2021 ,2)
clean_codon_number_2021<-substring(codon_number_messy_2021, 1, 3)
#print(clean_codon_number_2021)
#I want to generate a list of numbers of which codons were mutated, and this line creates a new global
#I also want to get rid of the final letter so that I only have the codon number for my replace functio
clean_codon_number_2021<-clean_codon_number_2021[-c(2, 3, 4, 6, 11, 12, 13, 14, 15, 18, 51)]
#Remove any replications that will mess with our final replace call.
#print(clean_codon_number_2021)
class(clean_codon_number_2021)
```

```
## [1] "character"
```

```
numeric_codon_number_2021<-as.numeric(clean_codon_number_2021)
#print(numeric_codon_number_2021)
#To use the replace function the numbers that tell the function which amino acids to replace have to be

mutated_aa_2021<-substring(list_mutations_2021, 5)
clean_mutated_aa_2021<-mutated_aa_2021[-c(2, 3, 4, 6, 11, 12, 13, 14, 15, 18, 51)]
#Create a new global variable that is a list of the 5th letter of the entries, which is the letter of t

length(numeric_codon_number_2021)
```

```
## [1] 41
```

```
length(clean_mutated_aa_2021)
```

```
## [1] 41
```

```
#Checking the length helps us be sure that our cleanup was succesful and that these global variables ar


mutations_2021_sequence<-replace(p53_wt_aa, numeric_codon_number_2021, clean_mutated_aa_2021)
#Using the variables that we created earlier, we are once again using the replace function to generate
#print(mutations_2021_sequence)
```

```
#print(p53_wt_aa)
length(mutations_2021_sequence)
```

```
## [1] 393
```

```
length(p53_wt_aa)
```

```
## [1] 393
```

After checking the length we can see that our new sequence still has the same numebr of amino acids as the original, so our replace worked!!

5. Turn Sequences into Fasta Files and Concatenate

```
write.fasta(sequences=p53_wt_aa, names="p53_wt_aa", file.out="p53_wt_aa.fasta")
write.fasta(sequences=mutations_2015_sequence, names="mutations_2015_sequence", file.out="mutations_2015
write.fasta(sequences=mutations_2014_sequence, names="mutations_2014_sequence", file.out="mutations_2014
write.fasta(sequences=mutations_2021_sequence, names="mutations_2021_sequence", file.out="mutations_2021
```

Once again, the msa alignment needs all of the sequences in each alignment to be combined into a single file. To do this we are using the write.fasta function to turn all of the sequences that we just made into fasta files.

fastaconc(otus=c('p53_wt_aa', 'uva_mutations_sequence', 'uvb_mutations_sequence'), inputdir = ".", out.file = "./wt_uva_uvb.fasta")

fastaconc(otus=c('p53_wt_aa', 'uva_mutations_sequence', 'uvb_mutations_sequence', 'mutations_2015_sequence', 'mutations_2014_sequence', 'mutations_2021_sequence'), inputdir = ".", out.file = "./wt_uv_bioportal_mutations.fasta")

```
wt_uv_bioportal_mutations<-readAAStringSet("wt_uv_bioportal_mutations.fasta")
#Use the readAAStringSet to make a new global variable that containes the concatenated fasta file that
wt_uv_bioportal_mutations_alignment<-msa(wt_uv_bioportal_mutations, order="input")
```

```
## use default substitution matrix
```

```
#This code uses the msa multiple sequence alignment function to align the sequences.
print(wt_uv_bioportal_mutations, show="complete")
```

```
## <S4 Type Object>
## attr(,"elementType")
## [1] "AAString"
## attr(,"pool")
## SharedRaw_Pool of length 1
## 1: SharedRaw of length 2358 (data starting at address 0x7fdd5f1d2a30)
## attr(,"ranges")
##   group start  end width                  names
## 1     1     1  393   393               p53_wt_aa
## 2     1   394  786   393  uva_mutations_sequence
## 3     1   787 1179   393  uvb_mutations_sequence
## 4     1  1180 1572   393 mutations_2015_sequence
## 5     1  1573 1965   393 mutations_2014_sequence
## 6     1  1966 2358   393 mutations_2021_sequence
## attr(,"elementMetadata")
## `\001NULL\001`
## attr(,"metadata")
## list()
## attr(,"class")
## [1] "AAStringSet"
## attr(,"class")attr(,"package")
```

```
## [1] "Biostrings"
```

Once again this isn't super pretty or easy to interpret, so lets use msaPrettyPrint to clean it up and make it easier to read.

```
#msaPrettyPrint(wt_uv_bioportal_mutations_alignment, output="pdf", showNames="left", showLogo="none", s
```

This code generates a pdf called wt_uv_bioportal_mutations_alignment.pdf, but because my R studio has trouble communicating with LaTeX it results in an error, so I am commenting it out.

```
pdf_convert("wt_uv_bioportal_mutations_alignment.pdf", format = "png", pages = NULL, filenames = NULL, 
```

```
## Converting page 1 to wt_uv_bioportal_mutations_alignment_1.png... done!
```

```
## [1] "wt_uv_bioportal_mutations_alignment_1.png"
```

This code converts the pdf that was generated by the prettyprint into a png that can be knit into this R document so we can see the results of our alignment.

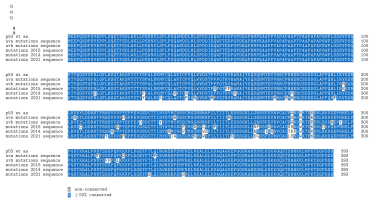


Figure 2: your caption

Using the knitr package, we can show the .png that we generated from the original pdf into this document.

After taking a look at the alignment, we can see that there aren't very many of the UV mutations that appear in the bioportal sequences.

P values

3. P-Value

Based on the results of the alignment, we can see that only one of the UV fingerprint mutations appeared in any of the p53 sequences from cBioportal. This mutation is a UVA fingerptint mutation that changes a C to a W, and on our alignment we see that at least one of the sequences in the 2021 sample have this mutation.

Let's first query the bioportal databases to see how many times this mutation appears.

```
print(uva_mutations_final)
```

```
##    Codon Number Amino Acid Change Original_Amino_Acid New_Amino_Acid
## 5           238         Cys _ Trp                  C              W
## 6           257         Leu _ Arg                  L              R
## 7           220         Tyr _ Asp                  Y              D
## 8           203         Val _ Gly                  V              G
## 9           320         Lys _ Gln                  K              Q
## 10          331         Gln _ Pro                  Q              P
## 51          311         Asn _ Thr                  N              T
```

```
#We see it in our UVA alignment
#The mutation of interest:
UVA_mutation<-"C238W"
```

```
#We are assigning a new global variable that will be used in our calculations going forward.

#print(clean_2014_mutations$Protein.Change)
#print(clean_2015_mutations$Protein.Change)
#print(clean_2021_mutations$Protein.Change)
#Uncommenting the prints above shows all of the lists of mutations.

#It is possible that our replace function may not have worked or introduced all of the mutations, so I

logical_MOI_2014<-clean_2014_mutations$Protein.Change==UVA_mutation
logical_MOI_2015<-clean_2015_mutations$Protein.Change==UVA_mutation
logical_MOI_2021<-clean_2021_mutations$Protein.Change==UVA_mutation
#print(logical_MOI_2014)
#print(logical_MOI_2015)
#print(logical_MOI_2021)
#These codes create lists that have output TRUE if entry matches the mutation, and false if the mutatio

index_MOI_2014<-which(logical_MOI_2014, arr.ind=TRUE)
index_MOI_2015<-which(logical_MOI_2015, arr.ind=TRUE)
index_MOI_2021<-which(logical_MOI_2021, arr.ind=TRUE)
#Now we can use the logical_MOI global variables that we created to make a new global variable, which i

number_of_MOI_2014<-length(index_MOI_2014)
number_of_MOI_2015<-length(index_MOI_2015)
number_of_MOI_2021<-length(index_MOI_2021)
print(number_of_MOI_2014)
```

```
## [1] 0
```

```
print(number_of_MOI_2015)
```

```
## [1] 0
```

```
print(number_of_MOI_2021)
```

```
## [1] 2
```

```
#Now we can use the length function to cound the index_MOI global variables, and assign the legnths to
```

This means that out of all of the samples in the 2021 dataset, 2 of them had a UV induced mutation.

Next we need to figure out the sample size by looking at the bioportal mutations tables that we created earlier.

```
#print(bioportal_2014_mutations)
samples_2014<-bioportal_2014_mutations$Sample.ID
samples_2015<-bioportal_2015_mutations$Sample.ID
samples_2021<-bioportal_2021_mutations$Sample.ID
#We want to make a new list out of the column in the original table that hase the information on the sa
unique_samples_2014<-unique(samples_2014)
unique_samples_2015<-unique(samples_2015)
unique_samples_2021<-unique(samples_2021)
#print(unique_samples_2014)
#print(unique_samples_2015)
#print(unique_samples_2021)
#Many of the samples had multiple p53 mutations, so we want to create a new global variable that has th
```

```
number_of_samples_2014<-length(unique_samples_2014)
number_of_samples_2015<-length(unique_samples_2015)
number_of_samples_2021<-length(unique_samples_2021)
print(number_of_samples_2014)
```

```
## [1] 37
```

```
print(number_of_samples_2015)
```

```
## [1] 23
```

```
print(number_of_samples_2021)
```

```
## [1] 55
```

```
#Using the length function, we can make new global variables that have the length of the unique samples
```

Averages from our samples

```
mean_MOI_2014<-number_of_MOI_2014/number_of_samples_2014
mean_MOI_2015<-number_of_MOI_2015/number_of_samples_2015
mean_MOI_2021<-number_of_MOI_2021/number_of_samples_2021
print(mean_MOI_2014)
```

```
## [1] 0
```

```
print(mean_MOI_2015)
```

```
## [1] 0
```

```
print(mean_MOI_2021)
```

```
## [1] 0.03636364
```

```
#We can make a new global variable that stores the mean of the frequency of the mutation. We caluclate

all_three_samples<-c(mean_MOI_2014, mean_MOI_2015, mean_MOI_2021)
print(all_three_samples)
```

```
## [1] 0.00000000 0.00000000 0.03636364
```

```
class(all_three_samples)
```

```
## [1] "numeric"
```

```
#For our p value calculation we need the mean from multiple samples, so we need to make a new global va
mean_samples<-mean(all_three_samples)
sd_samples<-sd(all_three_samples)
#Now we can use the mean() and sd() calculations on the all_three_samples_variable to find the average

sum_of_samples<-sum(number_of_samples_2014, number_of_samples_2015, number_of_samples_2021)
print(sum_of_samples)
```

```
## [1] 115
```

```
#To get the total sample size, we can make a new global variable that uses the sum() function to get th
```

Now we are ready to calculate a P-Value!

Van Kempen Et Al defined a hotspot mutation in sCCC as a mutation that was present in 22% of patients sampled. Our null hypotheis will be that 22% of the mutations should have this UV mutation for it to be

considered a hotspot mutation.

We will be performing a one-sided T test, and if we determine that the mean of the sample is significantly lower than .22 and we reject the null hypothesis, we will say that our findings are not indicitave that UV radiation could be causitive of cSCC.

```
xbar<-mean_samples
#When using p values, the xbar is the mean of the actual, studied sample. We are going to be using the
#print(xbar)
a=0.22
#The a value refers to our null hypothesis. We are using 0.22 because in the literature others have pub
std_dev=sd_samples
#The std_dev is the global variable that we wrote earlier when we took the standard deviation of the 3
n=sum_of_samples
#n is the sample size, and we will use the global variable that we wrote earlier that took the sum of a

p_value_function<-function(xbar_1, a_1, std_dev_1, n_1){
  #the arguments of this function will be xbar_1, or the standard mean, a_1 or the value of the null hy
 z_value<-(xbar_1-a_1)/(std_dev_1/sqrt(n_1))
 #the first step is to caluclate the z value with the xbar, a, standard deviation, and sample size
  p_value<-pnorm(-abs(z_value))
  #the p value is determined by inputting the negative absolute value of the z value into the pnorm fun
  return(p_value)
  #we want our function to return the p value that was calculated so we can interpret the final p value
}
p_value_function(xbar, a, std_dev, n)
```

```
## [1] 0
```

```
#Input our experimental values to get the p value.
```

The p value is 0. In the contex of our one-sided p-value anlaysis, this small of a p value indicates we can reject the null hypothesis. This means that in our sample the average rate of this mutation was significanlty smaller than what would be needed to be considered a hotspot mutation.

Results

Based on the results of our analysis, we cannot conclude that the latitudinal difference in rates of cSCC are due to higher frequencies of UV-induced mutations in the p53 gene. Our plotting showed that there might be a correlation between latitude and rate of cSCC becuase of the similar trend that rates of cSCC show when compared to UV radiation. It is possible that there may be another way that UV radiation is impacting DNA that results in higher rates of cSCC in places with high levels of UV radiation, but our results did not elucidate a pathway. We performed multiple sequence alignment to determine if any of the published UVA or UVB fingerprint mutations were common in p53 tumor samples that are published in the cBioportal database. This analysis revealed one UVA mutation that was present the actual cSCC tumor samples. Our P value revealed that the frequency of this mutation was signifcianlty lower than the threshold for being considered a hotspot mutation. A possible explanation could be that there are other mutations that are only induced my UV radiation but that have not yet been discovered. Another explanation could be that possibly the cBioportal samples were all taken from a high latitude or regions with overall lower rates of cSCC.