This is a complement to the hands-on session for lecture 18 of BIMM-143. You can find the Rmarkdown document that generated this page here. In the following sections we walk through the analysis steps providing periodic output examples.

Q1. Identifing sites of mutation

We start by **1.** reading the provided sequences (lecture18_sequences.fa) into R, then **2.** aligning, **3.** looking for sites of cancer specific mutation (i.e. differences between the two sequences), and finally **4.** outputing all 9-mer contaning subsequences encompasing these mutant sites.

```
library(bio3d)
seqs <- read.fasta("lecture18_sequences.fa")
seqs

##                1        .         .         .         .         .         60
## P53_wt         MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQAMDDLMLSPDDIEQWFTEDPGP
## P53_mutant     MEEPQSDPSVEPPLSQETFSDLWKLLPENNVLSPLPSQAMLDLMLSPDDIEQWFTEDPGP
##                **************************************** *******************
##                1        .         .         .         .         .         60
##
##               61        .         .         .         .         .
120
## P53_wt         DEAPRMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAK
## P53_mutant     DEAPWMPEAAPPVAPAPAAPTPAAPAPAPSWPLSSSVPSQKTYQGSYGFRLGFLHSGTAK
##                **** *******************************************************
##               61        .         .         .         .         .
120
##
##              121        .         .         .         .         .
180
## P53_wt         SVTCTYSPALNKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHHE
## P53_mutant     SVTCTYSPALNKMFCQLAKTCPVQLWVDSTPPPGTRVRAMAIYKQSQHMTEVVRRCPHHE
##                ************************************************************
##              121        .         .         .         .         .
180
##
##              181        .         .         .         .         .
240
## P53_wt         RCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFRHSVVVPYEPPEVGSDCTTIHYNYMCNS
## P53_mutant     RCSDSDGLAPPQHLIRVEGNLRVEYLDDRNTFVHSVVVPYEPPEVGSDCTTIHYNYMCNS
##                ******************************** ***************************
##              181        .         .         .         .         .
240
##
##              241        .         .         .         .         .
300
## P53_wt         SCMGGMNRRPILTIITLEDSSGNLLGRNSFEVRVCACPGRDRRTEEENLRKKGEPHHELP
## P53_mutant     SCMGGMNRRPILTIITLEV-----------------------------------------
```

```
##               ******************
##             241         .         .         .         .         .
300
##
##             301         .         .         .         .         .
360
## P53_wt       PGSTKRALPNNTSSSPQPKKKPLDGEYFTLQIRGRERFEMFRELNEALELKDAQAGKEPG
## P53_mutant   ------------------------------------------------------------
##
##             301         .         .         .         .         .
360
##
##             361         .         .         .  393
## P53_wt       GSRAHSSHLKSKKGQSTSRHKKLMFKTEGPDSD
## P53_mutant   ---------------------------------
##
##             361         .         .         .  393
##
## Call:
##   read.fasta(file = "lecture18_sequences.fa")
##
## Class:
##   fasta
##
## Alignment dimensions:
##   2 sequence rows; 393 position columns (259 non-gap, 134 gap)
##
## + attr: id, ali, call
```

We can optionally align these sequences to make sure we have residue position correspondences correctly mapped between wt and mutant (incase of indels) with the following code. However, this appears to be unnecessary in this case as the provided sequences are already aligned.

```
#seqs <- seqaln(seqs)
```

Next we calculate identity per equivalent (i.e. aligned) position and then use this information to find non identical sites that do not contain gaps (i.e. indels).

```
## Calculate positional identity scores
ide <- conserv(seqs$ali, method="identity")
mutant.sites <- which(ide < 1)

## Exclude gap possitions from analysis
gaps <- gap.inspect(seqs)
mutant.sites <- mutant.sites[mutant.sites %in% gaps$f.inds]

mutant.sites

## [1]  41  65 213 259
```

We can use these indices in `mutant.sites` to extract subsequences as required for the hands-on session. First however we come up with suitable names for these subsequences based on the mutation. This will help us later to make sense and keep track of our results.

```
## Make a "names" label for our output sequences (one per mutant)
mutant.names <- paste0(seqs$ali["P53_wt",mutant.sites],
                       mutant.sites,
                       seqs$ali["P53_mutant",mutant.sites])

mutant.names

## [1] "D41L"  "R65W"  "R213V" "D259V"
```

Now lets extract all 9-mer mutant encompassing sequences for each mutant site. This is equivalent to finding the sequence region eight residues before and eight residues after our mutation sites and outputting this subsequence to a new FASTA file.

```
## Sequence positions surounding each mutant site
start.position <- mutant.sites - 8
end.position <-  mutant.sites + 8

# Blank matrix to store sub-sequences
store.seqs <- matrix("-", nrow=length(mutant.sites), ncol=17)
rownames(store.seqs) <- mutant.names

## Extract each sub-sequence
for(i in 1:length(mutant.sites)) {
  store.seqs[i,] <- seqs$ali["P53_mutant",start.position[i]:end.position[i]]
}

store.seqs

##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## D41L  "S"  "P"  "L"  "P"  "S"  "Q"  "A"  "M"  "L"  "D"   "L"   "M"   "L"  
## R65W  "D"  "P"  "G"  "P"  "D"  "E"  "A"  "P"  "W"  "M"   "P"   "E"   "A"  
## R213V "Y"  "L"  "D"  "D"  "R"  "N"  "T"  "F"  "V"  "H"   "S"   "V"   "V"  
## D259V "I"  "L"  "T"  "I"  "I"  "T"  "L"  "E"  "V"  "-"   "-"   "-"   "-"  
##       [,14] [,15] [,16] [,17]
## D41L  "S"   "P"   "D"   "D"  
## R65W  "A"   "P"   "P"   "V"  
## R213V "V"   "P"   "Y"   "E"  
## D259V "-"   "-"   "-"   "-"
```

Finally lets output all these sequences to a FASTA file for further analysis with the IEDB HLA binding prediction website http://tools.iedb.org/mhci/.

```
## Output a FASTA file for further analysis
write.fasta(seqs=store.seqs, ids=mutant.names, file="subsequences.fa")
```

Sidenote: Input sequence setup

For refernce only, here we use the UniProt KRas oncogene sequence (http://www.uniprot.org/uniprot/P01116) as an example input and make 4 substations at random positions. Students would not need to do this as they will be provided with the output wild-type (`wt`) and mutant (`mutant`) containing FASTA format sequence file. We could also use p53 or any other protein for this hands-on session.

```
library(bio3d)

## Read KRas oncogene sequence from UniProt
wt <- get.seq("P01116")
```

```
## Warning in get.seq("P01116"): Removing existing file: seqs.fasta
```

```
## Here we make four mutants namely: G12V, Q22N, T74S and A130V
mutant <- wt
mutant$ali[ c(12,22,74,130)] <- c("V", "N", "S", "V")

write.fasta( seqbind(wt, mutant), ids=c("wt","mutant"), file="kras-sequences.fa")
```

Session Info

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
## 
## Matrix products: default
## 
## locale:
## [1] LC_COLLATE=English_United States.1252 
## [2] LC_CTYPE=English_United States.1252   
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C                          
## [5] LC_TIME=English_United States.1252    
## 
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base     
## 
## other attached packages:
## [1] bio3d_2.3-4
## 
## loaded via a namespace (and not attached):
##  [1] compiler_3.5.2  magrittr_1.5    parallel_3.5.2  tools_3.5.2    
##  [5] htmltools_0.3.6 yaml_2.2.0      Rcpp_1.0.0      stringi_1.2.4  
##  [9] rmarkdown_1.11  grid_3.5.2      knitr_1.21      stringr_1.3.1  
## [13] xfun_0.4        digest_0.6.18   evaluate_0.12
```