Uterine Leiomyoma Beadchip Gene Expressions

Janis Corona

2/23/2020

This is to re-examine the UL and non-UL samples from the Gene Expression Omnibus online data repository (GEO) for genotypes in the ULs compared to those samples without tumor tissue in them. The accession IDs for the Series is GSE95101 and for the platform is GPL13376

Lets look at some of these copy number variants of one gene with seven copy number variants or CNVs and see where the changes in the nucleotide sequences occur. Copy number variations in nucleotides can have short repeats, jumps in sequence, or deletions of a gene. I have been calling these CNVs genotypes, which are the traits and alleles responsible for the physical traits or phenotypes of an organism. Some CNVs are responsible for diseases, and in tumors there are many different CNVs that are found to be responsible. A uterine leiomyoma or fibroid is a benign tumor. These samples were taken from uterus tissue with these uternine tumors and the same neighboring uterine tissue without uterine tumors.

```
library(dplyr)
library(tidyr)
library(e1071)
library(caret)
library(randomForest)
library(MASS)
library(gbm)

UL1a <- read.csv('UL1a.csv', sep=',',
                 header=T, na.strings=c('',' '))
UL1b <- read.csv('UL1b.csv', sep=',',
                 header=T, na.strings=c('',' '))
UL1c <- read.csv('UL1c.csv', sep=',',
                 header=T, na.strings=c('',' '))
UL1d <- read.csv('UL1d.csv', sep=',',
                 header=T, na.strings=c('',' '))

UL1 <- rbind(UL1a,UL1b,UL1c,UL1d)
rm(UL1a,UL1b,UL1c,UL1d)
str(UL1)

## 'data.frame':    48701 obs. of  51 variables:
##  $ X                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ID                  : Factor w/ 48701 levels "ILMN_1343289",..: 1 2 3
4 5 6 7 8 9 10 ...
##  $ Species             : Factor w/ 1 level "Homo sapiens": 1 1 1 1 1 1 1
```

```
1 1 1 ...
##  $ Source               : Factor w/ 3 levels "ILMN_Controls",..: 1 1 2 1 1
2 2 2 2 2 ...
##  $ Search_Key           : Factor w/ 46721 levels
"ILMN_10001","ILMN_10014",..: 11664 11664 11664 11664 11664 11664 11664 9848
11306 1417 ...
##  $ Transcript           : Factor w/ 46724 levels
"ILMN_10001","ILMN_10014",..: 2290 2291 1377 2292 2293 4903 1218 9858 11314
1421 ...
##  $ ILMN_Gene            : Factor w/ 43186 levels "1-Dec","1-Mar",..: 8904
10174 2074 10143 10156 88 2713 4872 9250 2029 ...
##  $ Source_Reference_ID  : Factor w/ 46721 levels "NM_000015.1",..: 776
4636 1225 1123 1827 1130 1438 9379 7158 10894 ...
##  $ RefSeq_ID            : Factor w/ 28570 levels "NM_000015.1",..: 776
4636 1225 1123 1827 1130 1438 9379 7158 10894 ...
##  $ Unigene_ID           : Factor w/ 18153 levels "NA","Hs.100554",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ Entrez_Gene_ID       : Factor w/ 16063 levels "10","10000","10001",..:
10744 10744 1487 10744 10744 5871 2331 6516 10718 5545 ...
##  $ GI                   : int  14141192 20149305 25453469 4507728 4507744
5016088 7669491 88954077 33469136 89070645 ...
##  $ Accession            : Factor w/ 46721 levels "NM_000015.1",..: 776
4636 1225 1123 1827 1130 1438 9379 7158 10894 ...
##  $ Symbol               : Factor w/ 25036 levels "1-Dec","1-Mar",..: 8904
10174 2074 10143 10156 88 2713 4872 9250 2029 ...
##  $ Protein_Product      : Factor w/ 28173 levels "NA","NP_000006.1",..: 1
1 1224 1 1 1129 1437 9300 7155 10815 ...
##  $ Probe_Id             : Factor w/ 48701 levels "ILMN_1343289",..: 1 2 3
4 5 6 7 8 9 10 ...
##  $ Array_Address_Id     : int  2140735 6550370 2690379 4590356 4260048
5860528 1770601 50270 3310274 7040079 ...
##  $ Probe_Type           : Factor w/ 3 levels "A","I","S": 3 3 3 3 3 3 3 3
3 2 ...
##  $ Probe_Start          : int  416 1856 1293 1408 72 1725 930 1 1103 2975
...
##  $ SEQUENCE             : Factor w/ 48701 levels
"AAAAAACAGGAATAGCTCTAGGAGTCCTTACACAGGTCCGAGGGACCAGC",..: 9453 9026 11512 4877
6702 9532 5532 1967 10077 11476 ...
##  $ Chromosome           : Factor w/ 56 levels "1","10","11",..: 27 27 22
27 27 23 4 14 1 27 ...
##  $ Probe_Chr_Orientation: Factor w/ 3 levels "-","+","NA": 3 3 1 3 3 1 2 2
1 3 ...
##  $ Probe_Coordinates    : Factor w/ 41351 levels "100000925-100000974",..:
10161 10161 8915 10161 10161 7449 8192 4231 2830 10161 ...
##  $ Cytoband             : Factor w/ 3676 levels "10p11.1d","10p11.21a",..:
2501 2501 2039 2501 2501 2165 288 1490 1036 2033 ...
##  $ Definition           : Factor w/ 46614 levels "Homo sapiens 1-
acylglycerol-3-phosphate O-acyltransferase 1 (lysophosphatidic acid
acyltransferase, alpha) (AGP"| __truncated__,..: 5692 7144 2221 7047 6692 98
2786 8195 6195 7895 ...
```

```
##  $ Ontology_Component   : Factor w/ 7849 levels "A 20S multiprotein
assembly of total mass about 1.2 MDa that activates dynein-based activity in
vivo. A large s"| __truncated__,..: 2614 1893 1304 1321 1266 2112 1468 1893
1763 269 ...
##  $ Ontology_Process     : Factor w/ 8950 levels "[goid 6069] [pmid
1755855] [evidence IDA]; A change in state or activity of a cell or an
organism (in terms of "| __truncated__,..: 2067 2257 3716 1085 547 417 1784
1091 1091 962 ...
##  $ Ontology_Function    : Factor w/ 9453 levels "[goid 16505] [pmid
10426319] [evidence NAS]",..: 3847 2486 1975 1911 2750 2514 95 3637 3637 828
...
##  $ Synonyms             : Factor w/ 16472 levels "0610037N12Rik; RPP20;
RPP2",..: 4591 4591 5325 4591 4591 5300 2402 4591 3649 4591 ...
##  $ Obsolete_Probe_Id    : Factor w/ 16878 levels "0610037N12Rik; RPP20;
RPP2",..: 4784 4784 645 6450 1251 5508 2490 4784 3785 4784 ...
##  $ GB_ACC               : Factor w/ 46717 levels "NA","NM_000015.1",..:
777 4635 1225 1123 1827 1130 1438 9377 7156 10892 ...
##  $ GSM2496185           : num  13942 23759 27434 3092 6857 ...
##  $ GSM2496186           : num  12934 15091 26473 4269 7799 ...
##  $ GSM2496187           : num  11909 22609 23964 3455 7954 ...
##  $ GSM2496188           : num  12147 18225 27823 4258 7380 ...
##  $ GSM2496189           : num  14142 20728 24486 3333 6445 ...
##  $ GSM2496190           : num  11650 19582 26225 4545 9215 ...
##  $ GSM2496191           : num  12786 19105 28200 3413 10031 ...
##  $ GSM2496192           : num  9383 10008 27997 3191 7428 ...
##  $ GSM2496193           : num  11481 10575 23172 3597 7712 ...
##  $ GSM2496203           : num  4136 1028 16324 4994 6466 ...
##  $ GSM2496204           : num  11458 17921 26664 3095 7471 ...
##  $ GSM2496205           : num  15445 18186 25687 3138 7047 ...
##  $ GSM2496206           : num  11098 8905 22094 2473 6307 ...
##  $ GSM2496207           : num  11510 9721 21161 4353 4826 ...
##  $ GSM2496208           : num  11446 11451 26427 3863 7069 ...
##  $ GSM2496209           : num  9945 16387 27837 3027 6462 ...
##  $ GSM2496217           : num  12707 18456 28792 3251 8407 ...
##  $ GSM2496218           : num  12261 19342 25018 2322 6925 ...
##  $ GSM2496219           : num  11087 9198 27179 4554 9100 ...
##  $ GSM2496220           : num  11746 21023 29030 4131 7771 ...
```

```
nonUL1a <- read.csv('nonUL1a.csv', sep=',',
                header=T, na.strings=c('',' '))
nonUL1b <- read.csv('nonUL1b.csv', sep=',',
                header=T, na.strings=c('',' '))
nonUL1c <- read.csv('nonUL1c.csv', sep=',',
                header=T, na.strings=c('',' '))
nonUL1d <- read.csv('nonUL1d.csv', sep=',',
                header=T, na.strings=c('',' '))

nonUL1 <- rbind(nonUL1a,nonUL1b,nonUL1c,nonUL1d)
```

```
rm(nonUL1a,nonUL1b,nonUL1c,nonUL1d)
str(nonUL1)

## 'data.frame':    48701 obs. of  49 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ ID                   : Factor w/ 48701 levels "ILMN_1343289",..: 1 2 3
4 5 6 7 8 9 10 ...
##  $ Species              : Factor w/ 1 level "Homo sapiens": 1 1 1 1 1 1 1
1 1 1 ...
##  $ Source               : Factor w/ 3 levels "ILMN_Controls",..: 1 1 2 1 1
2 2 2 2 2 ...
##  $ Search_Key           : Factor w/ 46721 levels
"ILMN_10001","ILMN_10014",..: 11664 11664 11664 11664 11664 11664 11664 9848
11306 1417 ...
##  $ Transcript           : Factor w/ 46724 levels
"ILMN_10001","ILMN_10014",..: 2290 2291 1377 2292 2293 4903 1218 9858 11314
1421 ...
##  $ ILMN_Gene            : Factor w/ 43186 levels "1-Dec","1-Mar",..: 8904
10174 2074 10143 10156 88 2713 4872 9250 2029 ...
##  $ Source_Reference_ID  : Factor w/ 46721 levels "NM_000015.1",..: 776
4636 1225 1123 1827 1130 1438 9379 7158 10894 ...
##  $ RefSeq_ID            : Factor w/ 28570 levels "NM_000015.1",..: 776
4636 1225 1123 1827 1130 1438 9379 7158 10894 ...
##  $ Unigene_ID           : Factor w/ 18153 levels "NA","Hs.100554",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ Entrez_Gene_ID       : Factor w/ 16063 levels "10","10000","10001",..:
10744 10744 1487 10744 10744 5871 2331 6516 10718 5545 ...
##  $ GI                   : int  14141192 20149305 25453469 4507728 4507744
5016088 7669491 88954077 33469136 89070645 ...
##  $ Accession            : Factor w/ 46721 levels "NM_000015.1",..: 776
4636 1225 1123 1827 1130 1438 9379 7158 10894 ...
##  $ Symbol               : Factor w/ 25036 levels "1-Dec","1-Mar",..: 8904
10174 2074 10143 10156 88 2713 4872 9250 2029 ...
##  $ Protein_Product      : Factor w/ 28173 levels "NA","NP_000006.1",..: 1
1 1224 1 1 1129 1437 9300 7155 10815 ...
##  $ Probe_Id             : Factor w/ 48701 levels "ILMN_1343289",..: 1 2 3
4 5 6 7 8 9 10 ...
##  $ Array_Address_Id     : int  2140735 6550370 2690379 4590356 4260048
5860528 1770601 50270 3310274 7040079 ...
##  $ Probe_Type           : Factor w/ 3 levels "A","I","S": 3 3 3 3 3 3 3 3
3 2 ...
##  $ Probe_Start          : int  416 1856 1293 1408 72 1725 930 1 1103 2975
...
##  $ SEQUENCE             : Factor w/ 48701 levels
"AAAAAACAGGAATAGCTCTAGGAGTCCTTACACAGGTCCGAGGGACCAGC",..: 9453 9026 11512 4877
6702 9532 5532 1967 10077 11476 ...
##  $ Chromosome           : Factor w/ 56 levels "1","10","11",..: 27 27 22
27 27 23 4 14 1 27 ...
##  $ Probe_Chr_Orientation: Factor w/ 3 levels "-","+","NA": 3 3 1 3 3 1 2 2
1 3 ...
```

```
##  $ Probe_Coordinates    : Factor w/ 41351 levels "100000925-100000974",..:
10161 10161 8915 10161 10161 7449 8192 4231 2830 10161 ...
##  $ Cytoband             : Factor w/ 3676 levels "10p11.1d","10p11.21a",..:
2501 2501 2039 2501 2501 2165 288 1490 1036 2033 ...
##  $ Definition           : Factor w/ 46614 levels "Homo sapiens 1-
acylglycerol-3-phosphate O-acyltransferase 1 (lysophosphatidic acid
acyltransferase, alpha) (AGP"| __truncated__,..: 5692 7144 2221 7047 6692 98
2786 8195 6195 7895 ...
##  $ Ontology_Component   : Factor w/ 7849 levels "A 20S multiprotein
assembly of total mass about 1.2 MDa that activates dynein-based activity in
vivo. A large s"| __truncated__,..: 2614 1893 1304 1321 1266 2112 1468 1893
1763 269 ...
##  $ Ontology_Process     : Factor w/ 8950 levels "[goid 6069] [pmid
1755855] [evidence IDA]; A change in state or activity of a cell or an
organism (in terms of "| __truncated__,..: 2067 2257 3716 1085 547 417 1784
1091 1091 962 ...
##  $ Ontology_Function    : Factor w/ 9453 levels "[goid 16505] [pmid
10426319] [evidence NAS]",..: 3847 2486 1975 1911 2750 2514 95 3637 3637 828
...
##  $ Synonyms             : Factor w/ 16472 levels "0610037N12Rik; RPP20;
RPP2",..: 4591 4591 5325 4591 4591 5300 2402 4591 3649 4591 ...
##  $ Obsolete_Probe_Id    : Factor w/ 16878 levels "0610037N12Rik; RPP20;
RPP2",..: 4784 4784 645 6450 1251 5508 2490 4784 3785 4784 ...
##  $ GB_ACC               : Factor w/ 46717 levels "NA","NM_000015.1",..:
777 4635 1225 1123 1827 1130 1438 9377 7156 10892 ...
##  $ GSM2496194           : num  9823 18157 27796 3428 7706 ...
##  $ GSM2496195           : num  11265 20893 24042 4279 9407 ...
##  $ GSM2496196           : num  13016 20943 24368 3049 9110 ...
##  $ GSM2496197           : num  11698 18242 24179 3574 7935 ...
##  $ GSM2496198           : num  11448 20998 25276 2178 7307 ...
##  $ GSM2496199           : num  11454 21756 26935 3768 8928 ...
##  $ GSM2496200           : num  11514 21849 26969 3170 9457 ...
##  $ GSM2496201           : num  10621 10200 24231 2292 7765 ...
##  $ GSM2496202           : num  11066 9349 22945 4513 8454 ...
##  $ GSM2496210           : num  10189 21816 29280 4816 8773 ...
##  $ GSM2496211           : num  9998 18435 26231 4683 8579 ...
##  $ GSM2496212           : num  11407 23942 27389 3589 7977 ...
##  $ GSM2496213           : num  9476 10440 23432 5444 8126 ...
##  $ GSM2496214           : num  11708 9478 22640 4533 8418 ...
##  $ GSM2496215           : num  11457 11803 24008 5839 8549 ...
##  $ GSM2496216           : num  12900 20375 28086 4044 7252 ...
##  $ GSM2496221           : num  10020 16842 25324 2469 7225 ...
##  $ GSM2496222           : num  13409 9030 22273 3315 7844 ...
```

```r
UL <- UL1[,-c(1:13,15:19,21,29:31)]
nonUL <- nonUL1[,-c(1:13,15:19,21,29:31)]

write.csv(UL,'UL.csv', row.names=FALSE)
write.csv(nonUL, 'nonUL.csv', row.names=FALSE)
```

```
fibroid <- read.csv('UL.csv', sep=',', header=T, na.strings=c('',' '))
nonFibroid <- read.csv('nonUL.csv', sep=',', header=T, na.strings=c('',' '))

fibroid_gene_n <- fibroid %>% group_by(Symbol) %>% count(n())
narm <- grep('^NA$',fibroid_gene_n$Symbol)

fibroid1 <- fibroid_gene_n[-narm,-2]
colnames(fibroid1)[2] <- 'gene_count'

NONfibroid_gene_n <- nonFibroid %>% group_by(Symbol) %>% count(n())
narm1 <- grep('^NA$',NONfibroid_gene_n$Symbol)

nonFibroid1 <- NONfibroid_gene_n[-narm1,-2]
colnames(nonFibroid1)[2] <- 'gene_count'

GeneCopyNumberVariants <-
fibroid1[order(fibroid1$gene_count,decreasing=TRUE)[1:10],]
GeneCopyNumberVariants

## # A tibble: 10 x 2
## # Groups:   Symbol [10]
##    Symbol    gene_count
##    <fct>          <int>
##  1 DDX12             10
##  2 KIAA0692           9
##  3 LOC23117           8
##  4 PLEC1              8
##  5 BDNF               7
##  6 CTNNB1             7
##  7 DMD                7
##  8 LOC202134          7
##  9 LOC339047          7
## 10 LOC653086          7
```

Combine the gene counts with the tables of samples for each type of UL or nonUL.

```
Fibroid_count <- merge(fibroid1, fibroid, by.x='Symbol', by.y='Symbol')
nonFibroid_count <- merge(nonFibroid1, nonFibroid, by.x='Symbol',
by.y='Symbol')
Fibroid_count[order(Fibroid_count$gene_count, decreasing=TRUE)[1:20],1:3]

##         Symbol gene_count
SEQUENCE
## 5646     DDX12         10
CCAGTCCCTGACTACAGAGGATTTCCCCAAAGTCCCTGGCTGTGAGGTTC
## 5647     DDX12         10
TTACTGGGGATGGTATTTAGGAGCCAGGAAAGCCGGTGCATTCCTAGTGA
## 5648     DDX12         10
TCTCCTGCCCCCTCCGGAAGCTTGGATGCCCCTCCACACCCTCTTGATCT
## 5649     DDX12         10
CAGACTTCTCGCTTCCTTTCTGCTGGGCCTCTGAGGGGTCATGGGGCCAT
```

```
## 5650     DDX12         10
ACATGTGCTGTCACTGGAACTTGCTCTTTTCACTCAGCAGCCAGAGGGTC
## 5651     DDX12         10
AAACGTTACAGTGTTCCGATGAGACACAGTAGGCAGTACTTGGGAGGGTC
## 5652     DDX12         10
CAGGGCAGGAACCACGTCTTTACAGTTTGATGTTCCCAGAGCTGACCCAG
## 5653     DDX12         10
GCAGGGGAGATTGGGTTTAGGGGCTTTCCTGGTCTGCATTCTGCTACAGC
## 5654     DDX12         10
CCGCCGGGCTGCTTTTTCCTTGGATGCCCATCAGGACGCCTCAGTTCTCT
## 5655     DDX12         10
CGTTGCTACAAGCTGTTTTTTGAATGTCTCTACACAGTCCAGGCAGGAAG
## 10983 KIAA0692          9
AAGTGGTGCCTGGCTGTCCCTATACTGTGCTGCTGGGTGTTCCAGCCTGT
## 10984 KIAA0692          9
TAAGTGCAGTGAGCTCTGGCGGAAACCACCCTCTGCCCCGTCTGTTGGAT
## 10985 KIAA0692          9
CATTGTAATGATAAGGAAATGTTGCGATCAAATAAGATTTAGACACACTT
## 10986 KIAA0692          9
GATCACAGGCACAGGGAAGCCACAAGGAGCTCTGTATGAGTTGTGTTTGC
## 10987 KIAA0692          9
CAGGCGACTGGGTAGCAGATGTGGAAGCTGATGGTTAGGCCCAGGGCATG
## 10988 KIAA0692          9
GTTGTTCTGGACGATCTTCGGGATCCTCTGGGGCACTGTGACACTCGGAG
## 10989 KIAA0692          9
GAGTGCTGGGAAGGTTAATGTTAAATGGGTTGTGTGTCGGGGAGGGTACA
## 10990 KIAA0692          9
AGCTCCACCTTGACCCAGCCTCACAACAAAAAGTTTGTGTATGACCAGGC
## 10991 KIAA0692          9
GCAAATGTAACTCAGGGGTTTGGGGCCAGAGGAAGAGGGAGAAGGTGGCC
## 12174 LOC23117          8
CTGGCCTTCCCTCATCAGCCGTAAATGATGATTTACTGCTGTTACCATCA
```

Add a mean, median, min, and max column to these tables.

```
Fibroid_count$Fibroid_Mean <- rowMeans(Fibroid_count[11:30])
nonFibroid_count$nonFibroid_Mean <- rowMeans(nonFibroid_count[11:28])
```

Use the tidyr package to group by sample ID by gathering those columns into one.

```
UL_3 <- gather(Fibroid_count, 'UL_Sample_ID','Value',11:30)
nonUL_3 <- gather(nonFibroid_count, 'nonUL_Sample_ID', 'Value',11:28)
```

Create the stat tables then combine for the UL and nonUL sample sets using the dplyr package.

```
UL_median <- UL_3 %>% group_by(SEQUENCE) %>% summarise_at(vars(Value),
median)
colnames(UL_median)[2] <- 'Fibroid_Median'

nonUL_median <- nonUL_3 %>% group_by(SEQUENCE) %>% summarise_at(vars(Value),
```

```
median)
colnames(nonUL_median)[2] <- 'nonFibroid_Median'

UL_max <- UL_3 %>% group_by(SEQUENCE) %>% summarise_at(vars(Value), max)
colnames(UL_max)[2] <- 'Fibroid_max'

nonUL_max <- nonUL_3 %>% group_by(SEQUENCE) %>% summarise_at(vars(Value),
max)
colnames(nonUL_max)[2] <- 'nonFibroid_max'

UL_min <- UL_3 %>% group_by(SEQUENCE) %>% summarise_at(vars(Value), min)
colnames(UL_min)[2] <- 'Fibroid_min'

nonUL_min <- nonUL_3 %>% group_by(SEQUENCE) %>% summarise_at(vars(Value),
min)
colnames(nonUL_min)[2] <- 'nonFibroid_min'

UL_sd <- UL_3 %>% group_by(SEQUENCE) %>% summarise_at(vars(Value), sd)
colnames(UL_sd)[2] <- 'Fibroid_stdError'

nonUL_sd <- nonUL_3 %>% group_by(SEQUENCE) %>% summarise_at(vars(Value), sd)
colnames(nonUL_sd)[2] <- 'nonFibroid_stdError'
```

Combine these four tables together.

```
Fibroid_stats <- merge(UL_median, UL_max, by.x='SEQUENCE', by.y='SEQUENCE')
Fibroid_stats1 <- merge(Fibroid_stats, UL_min, by.x='SEQUENCE',
by.y='SEQUENCE')
Fibroid_stats2 <- merge(Fibroid_count, Fibroid_stats1, by.x='SEQUENCE',
by.y='SEQUENCE')
Fibroid_stats3 <- merge(Fibroid_stats2, UL_sd, by.x='SEQUENCE',
by.y='SEQUENCE')
colnames(Fibroid_stats3)[11:30] <- paste('UL_',
colnames(Fibroid_stats3)[11:30], sep='')

nonFibroid_stats <- merge(nonUL_median, nonUL_max, by.x='SEQUENCE',
by.y='SEQUENCE')
nonFibroid_stats1 <- merge(nonFibroid_stats, nonUL_min, by.x='SEQUENCE',
by.y='SEQUENCE')
nonFibroid_stats2 <- merge(nonFibroid_count, nonFibroid_stats1,
by.x='SEQUENCE', by.y='SEQUENCE')
nonFibroid_stats3 <- merge(nonFibroid_stats2, nonUL_sd, by.x='SEQUENCE',
by.y='SEQUENCE')
colnames(nonFibroid_stats3)[11:28] <- paste('nonUL_',
colnames(nonFibroid_stats3)[11:28], sep='')


nonfibroid <- nonFibroid_stats3[,c(1,11:33)]
```

```
all <- merge(Fibroid_stats3, nonfibroid, by.x='SEQUENCE', by.y='SEQUENCE')
str(all)

## 'data.frame':    30549 obs. of  58 variables:
##  $ SEQUENCE             : Factor w/ 48701 levels
"AAAAAACAAAACCGCGCAGCGGAGAACCGGTGCCTGAGTCTCCCAGGGAC",..: 1 4 5 7 8 10 12 13
15 18 ...
##  $ Symbol               : Factor w/ 25036 levels "1-Dec","1-Mar",..: 12031
11383 13002 14397 15611 12721 12474 10953 24822 18000 ...
##  $ gene_count           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Probe_Chr_Orientation: Factor w/ 3 levels "-","+","NA": 2 1 2 3 3 2 2 3
1 2 ...
##  $ Probe_Coordinates    : Factor w/ 41351 levels "100000925-100000974",..:
33600 3552 19430 41351 41351 36542 28872 41351 31503 41119 ...
##  $ Cytoband             : Factor w/ 3676 levels "10p11.1d","10p11.21a",..:
3472 3472 3472 3472 3472 293 3472 3472 1261 2180 ...
##  $ Definition           : Factor w/ 46614 levels "{3 region, probe S2}
[human, 76N, mammary epithelial cells, mRNA Partial, 339 nt]",..: 33743 33388
39229 37744 38664 34103 33968 35730 29449 21454 ...
##  $ Ontology_Component   : Factor w/ 7849 levels "A 20S multiprotein
assembly of total mass about 1.2 MDa that activates dynein-based activity in
vivo. A large s"| __truncated__,..: 4150 4150 4150 4150 4150 4150 4150 4150
5434 6267 ...
##  $ Ontology_Process     : Factor w/ 8950 levels "[goid 19642] [evidence
IEA]; The chemical reactions and pathways involving carbohydrates, any of a
group of org"| __truncated__,..: 2492 2492 2492 2492 2492 2492 2492 2492 8502
3316 ...
##  $ Ontology_Function    : Factor w/ 9453 levels "[goid 15280] [evidence
IEA]; Interacting selectively with sodium ions (Na+) [goid 31402] [evidence
IEA]",..: 8111 8111 8111 8111 8111 8111 8111 8111 7071 2705 ...
##  $ UL_GSM2496185        : num  49.5 51.2 59.9 312.3 52.3 ...
##  $ UL_GSM2496186        : num  51.5 52.3 64.5 333.3 53.7 ...
##  $ UL_GSM2496187        : num  46.9 49 50.3 331.8 56.4 ...
##  $ UL_GSM2496188        : num  50.1 48.4 52 360.9 51.8 ...
##  $ UL_GSM2496189        : num  54 49.8 52.8 339 52.6 ...
##  $ UL_GSM2496190        : num  50.1 48.6 52 411.7 51.9 ...
##  $ UL_GSM2496191        : num  49.5 46 54.7 424.3 54.8 ...
##  $ UL_GSM2496192        : num  46.5 50.6 50.2 517.3 53.6 ...
##  $ UL_GSM2496193        : num  48 46.7 52.9 631.7 54.1 ...
##  $ UL_GSM2496203        : num  48.1 51.5 55.1 576.7 55 ...
##  $ UL_GSM2496204        : num  51.7 50.8 51.6 300.4 52.3 ...
##  $ UL_GSM2496205        : num  46.9 47.5 52.3 348.9 53.2 ...
##  $ UL_GSM2496206        : num  50.5 43.9 54.9 722.6 56.1 ...
##  $ UL_GSM2496207        : num  44 46.5 56.5 1218 55.9 ...
##  $ UL_GSM2496208        : num  51.1 46 51.9 535.5 55.5 ...
##  $ UL_GSM2496209        : num  55.2 46.9 50.6 669 53.3 ...
##  $ UL_GSM2496217        : num  48.9 45.8 49.6 361.9 49.8 ...
##  $ UL_GSM2496218        : num  50 49.9 53.9 377.2 53.6 ...
##  $ UL_GSM2496219        : num  48.3 49.5 50.4 544.1 58.1 ...
##  $ UL_GSM2496220        : num  47.7 46.7 51.7 406.1 56.3 ...
```

```
##  $ Fibroid_Mean         : num  49.4 48.4 53.4 486.1 54 ...
##  $ Fibroid_Median       : num  49.5 48.5 52.1 408.9 53.7 ...
##  $ Fibroid_max          : num  55.2 52.3 64.5 1218 58.1 ...
##  $ Fibroid_min          : num  44 43.9 49.6 300.4 49.8 ...
##  $ Fibroid_stdError     : num  2.59 2.3 3.61 214.26 2 ...
##  $ nonUL_GSM2496194     : num  42.7 42.7 56.6 442 57 ...
##  $ nonUL_GSM2496195     : num  49.1 45.3 56 505.4 55.3 ...
##  $ nonUL_GSM2496196     : num  51.1 45.3 55.1 416.5 57.4 ...
##  $ nonUL_GSM2496197     : num  46 52.6 56.2 475.3 54.6 ...
##  $ nonUL_GSM2496198     : num  49.1 54.1 53.7 408.1 50.8 ...
##  $ nonUL_GSM2496199     : num  52.6 47.9 51.1 510.7 52.5 ...
##  $ nonUL_GSM2496200     : num  46 47.1 52.5 681.9 50.3 ...
##  $ nonUL_GSM2496201     : num  45.9 46.8 48.3 689.3 56.7 ...
##  $ nonUL_GSM2496202     : num  50.6 45.3 56.5 735.2 48.7 ...
##  $ nonUL_GSM2496210     : num  48.3 49.3 56.3 426.9 52.5 ...
##  $ nonUL_GSM2496211     : num  46.8 46 54.6 567.3 47.4 ...
##  $ nonUL_GSM2496212     : num  50.6 48.7 54.3 493.8 51.5 ...
##  $ nonUL_GSM2496213     : num  49.1 49.5 56.7 792.8 48.8 ...
##  $ nonUL_GSM2496214     : num  48.8 49.9 54 710.5 50.6 ...
##  $ nonUL_GSM2496215     : num  48 47.1 52.9 795.4 53.1 ...
##  $ nonUL_GSM2496216     : num  47.9 49.2 48 574.5 51.1 ...
##  $ nonUL_GSM2496221     : num  48.8 48.2 50.6 507.4 54.7 ...
##  $ nonUL_GSM2496222     : num  44.6 49.5 50.4 590 56 ...
##  $ nonFibroid_Mean      : num  48.1 48 53.5 573.5 52.7 ...
##  $ nonFibroid_Median    : num  48.5 48.1 54.1 539 52.5 ...
##  $ nonFibroid_max       : num  52.6 54.1 56.7 795.4 57.4 ...
##  $ nonFibroid_min       : num  42.7 42.7 48 408.1 47.4 ...
##  $ nonFibroid_stdError  : num  2.46 2.75 2.85 130.03 3.07 ...
```

Lets change the 'fibroid' in the column names to 'UL' for uterine leiomyoma.

```
colnames(all) <- gsub('Fibroid', 'UL', colnames(all))
```

Reorder the table so that the stats are at the end of the columns.

```
All <- all[,c(1:10,11:30, 36:53,31:35,54:58)]
str(All)
```

```
## 'data.frame':    30549 obs. of  58 variables:
##  $ SEQUENCE             : Factor w/ 48701 levels
"AAAAAACAAAACCGCGCAGCGGAGAACCGGTGCCTGAGTCTCCCAGGGAC",..: 1 4 5 7 8 10 12 13
15 18 ...
##  $ Symbol               : Factor w/ 25036 levels "1-Dec","1-Mar",..: 12031
11383 13002 14397 15611 12721 12474 10953 24822 18000 ...
##  $ gene_count           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Probe_Chr_Orientation: Factor w/ 3 levels "-","+","NA": 2 1 2 3 3 2 2 3
1 2 ...
##  $ Probe_Coordinates    : Factor w/ 41351 levels "100000925-100000974",..:
33600 3552 19430 41351 41351 36542 28872 41351 31503 41119 ...
##  $ Cytoband             : Factor w/ 3676 levels "10p11.1d","10p11.21a",..:
3472 3472 3472 3472 3472 293 3472 3472 1261 2180 ...
```

```
##  $ Definition           : Factor w/ 46614 levels "{3 region, probe S2}
[human, 76N, mammary epithelial cells, mRNA Partial, 339 nt]",..: 33743 33388
39229 37744 38664 34103 33968 35730 29449 21454 ...
##  $ Ontology_Component   : Factor w/ 7849 levels "A 20S multiprotein
assembly of total mass about 1.2 MDa that activates dynein-based activity in
vivo. A large s"| __truncated__,..: 4150 4150 4150 4150 4150 4150 4150 4150
5434 6267 ...
##  $ Ontology_Process     : Factor w/ 8950 levels "[goid 19642] [evidence
IEA]; The chemical reactions and pathways involving carbohydrates, any of a
group of org"| __truncated__,..: 2492 2492 2492 2492 2492 2492 2492 2492 8502
3316 ...
##  $ Ontology_Function    : Factor w/ 9453 levels "[goid 15280] [evidence
IEA]; Interacting selectively with sodium ions (Na+) [goid 31402] [evidence
IEA]",..: 8111 8111 8111 8111 8111 8111 8111 8111 7071 2705 ...
##  $ UL_GSM2496185        : num  49.5 51.2 59.9 312.3 52.3 ...
##  $ UL_GSM2496186        : num  51.5 52.3 64.5 333.3 53.7 ...
##  $ UL_GSM2496187        : num  46.9 49 50.3 331.8 56.4 ...
##  $ UL_GSM2496188        : num  50.1 48.4 52 360.9 51.8 ...
##  $ UL_GSM2496189        : num  54 49.8 52.8 339 52.6 ...
##  $ UL_GSM2496190        : num  50.1 48.6 52 411.7 51.9 ...
##  $ UL_GSM2496191        : num  49.5 46 54.7 424.3 54.8 ...
##  $ UL_GSM2496192        : num  46.5 50.6 50.2 517.3 53.6 ...
##  $ UL_GSM2496193        : num  48 46.7 52.9 631.7 54.1 ...
##  $ UL_GSM2496203        : num  48.1 51.5 55.1 576.7 55 ...
##  $ UL_GSM2496204        : num  51.7 50.8 51.6 300.4 52.3 ...
##  $ UL_GSM2496205        : num  46.9 47.5 52.3 348.9 53.2 ...
##  $ UL_GSM2496206        : num  50.5 43.9 54.9 722.6 56.1 ...
##  $ UL_GSM2496207        : num  44 46.5 56.5 1218 55.9 ...
##  $ UL_GSM2496208        : num  51.1 46 51.9 535.5 55.5 ...
##  $ UL_GSM2496209        : num  55.2 46.9 50.6 669 53.3 ...
##  $ UL_GSM2496217        : num  48.9 45.8 49.6 361.9 49.8 ...
##  $ UL_GSM2496218        : num  50 49.9 53.9 377.2 53.6 ...
##  $ UL_GSM2496219        : num  48.3 49.5 50.4 544.1 58.1 ...
##  $ UL_GSM2496220        : num  47.7 46.7 51.7 406.1 56.3 ...
##  $ nonUL_GSM2496194     : num  42.7 42.7 56.6 442 57 ...
##  $ nonUL_GSM2496195     : num  49.1 45.3 56 505.4 55.3 ...
##  $ nonUL_GSM2496196     : num  51.1 45.3 55.1 416.5 57.4 ...
##  $ nonUL_GSM2496197     : num  46 52.6 56.2 475.3 54.6 ...
##  $ nonUL_GSM2496198     : num  49.1 54.1 53.7 408.1 50.8 ...
##  $ nonUL_GSM2496199     : num  52.6 47.9 51.1 510.7 52.5 ...
##  $ nonUL_GSM2496200     : num  46 47.1 52.5 681.9 50.3 ...
##  $ nonUL_GSM2496201     : num  45.9 46.8 48.3 689.3 56.7 ...
##  $ nonUL_GSM2496202     : num  50.6 45.3 56.5 735.2 48.7 ...
##  $ nonUL_GSM2496210     : num  48.3 49.3 56.3 426.9 52.5 ...
##  $ nonUL_GSM2496211     : num  46.8 46 54.6 567.3 47.4 ...
##  $ nonUL_GSM2496212     : num  50.6 48.7 54.3 493.8 51.5 ...
##  $ nonUL_GSM2496213     : num  49.1 49.5 56.7 792.8 48.8 ...
##  $ nonUL_GSM2496214     : num  48.8 49.9 54 710.5 50.6 ...
##  $ nonUL_GSM2496215     : num  48 47.1 52.9 795.4 53.1 ...
##  $ nonUL_GSM2496216     : num  47.9 49.2 48 574.5 51.1 ...
```

```
##  $ nonUL_GSM2496221     : num  48.8 48.2 50.6 507.4 54.7 ...
##  $ nonUL_GSM2496222     : num  44.6 49.5 50.4 590 56 ...
##  $ UL_Mean              : num  49.4 48.4 53.4 486.1 54 ...
##  $ UL_Median            : num  49.5 48.5 52.1 408.9 53.7 ...
##  $ UL_max               : num  55.2 52.3 64.5 1218 58.1 ...
##  $ UL_min               : num  44 43.9 49.6 300.4 49.8 ...
##  $ UL_stdError          : num  2.59 2.3 3.61 214.26 2 ...
##  $ nonUL_Mean           : num  48.1 48 53.5 573.5 52.7 ...
##  $ nonUL_Median         : num  48.5 48.1 54.1 539 52.5 ...
##  $ nonUL_max            : num  52.6 54.1 56.7 795.4 57.4 ...
##  $ nonUL_min            : num  42.7 42.7 48 408.1 47.4 ...
##  $ nonUL_stdError       : num  2.46 2.75 2.85 130.03 3.07 ...

All_stats_only <- All[,c(1,2,3,49:58)]
stats_all <- All_stats_only[!duplicated(All_stats_only),]

stats_all$foldChangeMean_UL_to_nonUL <-
stats_all$UL_Mean/stats_all$nonUL_Mean

FoldChangeGenes <- stats_all[order(stats_all$foldChangeMean_UL_to_nonUL,
decreasing=TRUE)[c(1:5,30545:30549)],]

FoldChangeGenes

##                                                  SEQUENCE   Symbol
gene_count
## 16709 GCAACGCTCCTCTGAAATGCTTGTCTTTTTTCTGTTGCCGAAATAGCTGG KIAA1199
1
## 17948 GCCCCAGCAAGCCTCCCTCCATCCTCCAGTGGGAAACTGTTGATGGTGTT     PENK
1
## 22790 GGTATTGCTGATCGTATGCAGAAGGAAATCACTGCTCTGGCTCCTAGCAC     ACTC
1
## 28162 TGCGAGACCTGGGTGTCCAACCTGCGCTACAACCACATGCTGCGGAAGAA     DLK1
2
## 3569  AGGCCCTGGAGGCTGCAACATACCTCAATCCTGTCCCAGGCCGGATCCTC    MMP11
1
## 17565 GCCAACCTCCTCTCACAGCCTCTGTATCTCTGCAGGCCATACTGGTTCCA    ABCA8
1
## 6582  CAGATGTTTTCCCTTGTGGCAGTCTTCAGCCTCCTCTACCCTACATGATC    ADH1A
1
## 8958  CCCAGTGACACTTCAGAGAGCTGGTAGTTAGTAGCATGTTGAGCCAGGCC      FOS
1
## 27542 TGACTGTCCCTGCCAATGCTCCAGCTGTCGTCTGACTCTGGGTTCGTTGG     FOSB
1
## 6881  CAGCTGGGCGATGTGCGAGCTGATAGTGAGCGGCAGAATCAGGAGTACCA    KRT19
1
##         UL_Mean UL_Median     UL_max   UL_min UL_stdError nonUL_Mean
## 16709 3029.4710 2566.7096  6548.8160 156.8375  1948.88193  167.75283
## 17948 1798.4359  237.7790 26134.7105  48.2000  5783.33114  120.28719
## 22790 4693.5182 4880.4535 13992.2647 110.2923  3328.78611  392.30767
```

```
## 28162  946.4365  249.4440  5799.7170  54.7250  1542.06289   87.41913
## 3569  4767.6862 3652.1430 15286.3475 235.9674  4029.71487  442.19991
## 17565  143.2650   99.0957   628.2257  56.0500   126.17477  724.33959
## 6582   608.7258  545.4995  1947.6415  54.3500   476.16852 3102.90066
## 8958  1287.1220  805.1551  7335.1691 220.6810  1573.12379 6916.73077
## 27542  205.9075   70.8273  2290.1261  53.1000   493.97933 1142.67872
## 6881   130.4870   85.6123   311.2020  45.6000    90.60451  815.61118
##       nonUL_Median  nonUL_max nonUL_min nonUL_stdError
## 16709    116.83410   514.9222   63.1000      127.90943
## 17948     71.88335   431.7050   54.2000       98.67055
## 22790    332.26735   830.2335  169.4130      197.47324
## 28162     52.70835   669.4027   47.3000      145.29680
## 3569     308.24915  1683.5592   75.9750      430.96362
## 17565    643.88430  1493.2380  125.9636      317.38002
## 6582    2961.70150  7727.8238  275.2883     1833.54639
## 8958    6722.92010 17362.0317 2406.0706     3676.04993
## 27542    595.09105  7604.0221   92.4250     1864.52510
## 6881     738.20585  2049.6529   57.4000      498.12489
##       foldChangeMean_UL_to_nonUL
## 16709                 18.0591346
## 17948                 14.9511840
## 22790                 11.9638706
## 28162                 10.8264232
## 3569                  10.7817440
## 17565                  0.1977871
## 6582                   0.1961796
## 8958                   0.1860882
## 27542                  0.1801972
## 6881                   0.1599868
```

```
str(stats_all)
```

```
## 'data.frame':    30549 obs. of  14 variables:
##  $ SEQUENCE                  : Factor w/ 48701 levels
"AAAAAACAAAACCGCGCAGCGGAGAACCGGTGCCTGAGTCTCCCAGGGAC",..: 1 4 5 7 8 10 12 13
15 18 ...
##  $ Symbol                    : Factor w/ 25036 levels "1-Dec","1-Mar",..:
12031 11383 13002 14397 15611 12721 12474 10953 24822 18000 ...
##  $ gene_count                : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ UL_Mean                   : num  49.4 48.4 53.4 486.1 54 ...
##  $ UL_Median                 : num  49.5 48.5 52.1 408.9 53.7 ...
##  $ UL_max                    : num  55.2 52.3 64.5 1218 58.1 ...
##  $ UL_min                    : num  44 43.9 49.6 300.4 49.8 ...
##  $ UL_stdError               : num  2.59 2.3 3.61 214.26 2 ...
##  $ nonUL_Mean                : num  48.1 48 53.5 573.5 52.7 ...
##  $ nonUL_Median              : num  48.5 48.1 54.1 539 52.5 ...
##  $ nonUL_max                 : num  52.6 54.1 56.7 795.4 57.4 ...
##  $ nonUL_min                 : num  42.7 42.7 48 408.1 47.4 ...
##  $ nonUL_stdError            : num  2.46 2.75 2.85 130.03 3.07 ...
##  $ foldChangeMean_UL_to_nonUL: num  1.027 1.007 0.997 0.848 1.024 ...
```

```
write.csv(stats_all, 'stats_only_UL_nonUL.csv', row.names=FALSE)
```

Combine the table of top and bottom five genes in fold change values of the ratio of UL to non-UL sample means, FoldChangeGenes, with the table of the ten genes having the highest number of copy number variations or genotypes, GeneCopyNumberVariants.

```
ontology <- nonFibroid[,c(1,6:9)]
gnc <- as.data.frame(GeneCopyNumberVariants)[1]

keyGenes1 <- merge(gnc, stats_all, by.x='Symbol', by.y='Symbol')

keyGenes1a <- merge(keyGenes1, ontology, by.x='Symbol', by.y='Symbol')

keyGenes2 <- merge(FoldChangeGenes, ontology, by.x='Symbol', by.y='Symbol')
keyGenes2a <- keyGenes2[,c(1:3,15:18,4:14)]
keyGenes1b <- keyGenes1a[,c(1:3,15:18,4:14)]

KeyGenes <- rbind(keyGenes2a, keyGenes1b)
KG <- KeyGenes[!duplicated(KeyGenes$SEQUENCE),]
KG1 <- KG[order(KG$foldChangeMean_UL_to_nonUL, decreasing=TRUE),]
KG1[,c(1:3,18)]
```

```
##        Symbol                                                SEQUENCE
gene_count
## 8    KIAA1199 GCAACGCTCCTCTGAAATGCTTGTCTTTTTTCTGTTGCCGAAATAGCTGG
1
## 11       PENK GCCCCAGCAAGCCTCCCTCCATCCTCCAGTGGGAAACTGTTGATGGTGTT
1
## 2        ACTC GGTATTGCTGATCGTATGCAGAAGGAAATCACTGCTCTGGCTCCTAGCAC
1
## 4        DLK1 TGCGAGACCTGGGTGTCCAACCTGCGCTACAACCACATGCTGCGGAAGAA
2
## 10      MMP11 AGGCCCTGGAGGCTGCAACATACCTCAATCCTGTCCCAGGCCGGATCCTC
1
## 68     CTNNB1 AGCTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAG
7
## 89     CTNNB1 CTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAGCT
7
## 82     CTNNB1 AGTCTCTCGTAGTGTTAAGTTATAGTGAATACTGCTACAGCAATTTCTAA
7
## 231       DMD CAGTGTTGGGATCACTCACTTTCCCCCTACAGGACTCAGATCTGGGAGGC
7
## 210       DMD CTCCTCTCAGCTGAACACCCTCCTTTCACTCCCAAATGCAAACAGTCTCT
7
## 96     CTNNB1 GCCTCTTGCACTCTGAATTGGGAATGTTTGCACCACAGTGGGGGGCTTGC
7
## 140     DDX12 CCGCCGGGCTGCTTTTTCCTTGGATGCCCATCAGGACGCCTCAGTTCTCT
10
## 397  LOC23117 TCAACCACATCCTTCAAAAGGACTATGCCTGTTTATAAGCCCAGCTGTTT
8
```

```
## 361 LOC202134 GCCAAAGGAATGGGCTCCAGACACCCCCTCTTCCAGAGCAAGGATGAAGG
7
## 286  KIAA0692 TAAGTGCAGTGAGCTCTGGCGGAAACCACCCTCTGCCCCGTCTGTTGGAT
9
## 61     CTNNB1 CAAACTTTACAGAGGAGAATGCCCTGTTTGTTAACCATGTTTCTTTTGGC
7
## 516 LOC653086 TGATGTGTCACGCCACTGTACTCCAGCCTGACGGCAGAGCGAGACTCCAT
7
## 33       BDNF CCCTCCACCTCCTGCTCGGGGGGCTTTAATGAGACACCCACCGCTGCTGT
7
## 551     PLEC1 CCCGACGAGCAGGACTTCATCCAGGCCTACGAGGAGGTGCGCGAGAAGTA
8
## 445  LOC23117 TGTCGTTTCCTCCATTCTTCACCAAAACATCAGCGTACATAGGCACATGG
8
## 474 LOC339047 ACTGCCTGTGTGGCTCCTTGAGTGCGCGGAGGCCAAAGCTGAGATGACTT
7
## 331  KIAA0692 CATTGTAATGATAAGGAAATGTTGCGATCAAATAAGATTTAGACACACTT
9
## 559     PLEC1 CCCTCGGGCAGCCTGTTTCCCTCCCTGGTGGTTGTGGGTCACGTTGTCAC
8
## 530 LOC653086 GGTGTGCTCTGGTATGTAATGACAATATGTGAACAAACCTGTGGAATTAA
7
## 259  KIAA0692 GAGTGCTGGGAAGGTTAATGTTAAATGGGTTGTGTGTCGGGGAGGGTACA
9
## 429  LOC23117 GGCTCCTCTTTGGGCTCCTACTGGAATTTATCAGCCATCAGTGCATCTCT
8
## 252       DMD TCTATCAACAGAGCTGAATGAGTGCCAGGAAGCTGCGAAATCTGTCTTAC
7
## 268  KIAA0692 GCAAATGTAACTCAGGGGTTTGGGGCCAGAGGAAGAGGGAGAAGGTGGCC
9
## 19       BDNF AATAATAGAGTGTGGGAGTTTTGGGGCCGAAGTCTTTCCCGGAGCAGCTG
7
## 340 LOC202134 CAAACCCTTGAAGACATTTCAGGGCCATGCTCACTTGGGAGGGTTTGAGG
7
## 405  LOC23117 AAAGCAGTGGTTTTCAGCTGCCAGAGGCCTGAGAGAGTTTGGGCATACTC
8
## 245       DMD CCATTCAGAAGAATGATAAATGCCACAAGCATTTGGAAACAGGCTTCCCT
7
## 322  KIAA0692 AGCTCCACCTTGACCCAGCCTCACAACAAAAAGTTTGTGTATGACCAGGC
9
## 375 LOC202134 AGTGGGCAGAATGATGAGGGAAGTGGGCACGTGCCCATGTTCTTCTTGGC
7
## 382 LOC202134 TTCATCCAGGCCTGCGCCGGTGTTCACAGTGGTCCTCATCTAAGCCAGCC
7
## 591     PLEC1 CCGGGCCTTCTCGTGGTACCCTGCCTGCTGCCTTTGCCCCCGCACTGACT
8
## 47       BDNF GCTCGCTGAAGTTGGCTTCCTAGCGGTGTAGGCTGGAATAGACTCTTGGC
7
## 575     PLEC1 GGCGCAGACATGGACCCCTCGCGAGCCATCCAGAACGAGATCAGCTCCCT
8
```

```
## 313  KIAA0692 GATCACAGGCACAGGGAAGCCACAAGGAGCTCTGTATGAGTTGTGTTTGC
9
## 495 LOC339047 TTTCAGGCCCATGGCAGAGGGTGGGCTCAGGAGGGCCATCGTGGGTGTCC
7
## 170     DDX12 GCAGGGGAGATTGGGTTTAGGGGCTTTCCTGGTCTGCATTCTGCTACAGC
10
## 460 LOC339047 AGTGCCCACATCACACAGCATCTAGCACGTAACTGCACCCCGGGAGTCGT
7
## 437  LOC23117 GGCTCTGTTGGAATCCGCATAGTGTGGAAATGAGTTTGCCCTGGAAAGGG
8
## 502 LOC653086 AGTGTTGGGACTACAGGTGTGTGTTACTGCTCCCAGCTGGGAGGCAGGCT
7
## 509 LOC653086 GTGAGCCTGTTTCATCATCTGTAAACTTTGAATAATGATACCTACCCCGC
7
## 467 LOC339047 CGCCCTGAAAGGACCAGGACATGCGGGTGCGGTGGCTGCTCTTTTGGCTC
7
## 150     DDX12 CAGGGCAGGAACCACGTCTTTACAGTTTGATGTTCCCAGAGCTGACCCAG
10
## 75     CTNNB1 CAGGAATCTAGTCTGGATGACTGCTTCTGGAGCCTGGATGCAGTACCATT
7
## 295  KIAA0692 GTTGTTCTGGACGATCTTCGGGATCCTCTGGGGCACTGTGACACTCGGAG
9
## 180     DDX12 TCTCCTGCCCCCTCCGGAAGCTTGGATGCCCCTCCACACCCTCTTGATCT
10
## 389  LOC23117 CTGGCCTTCCCTCATCAGCCGTAAATGATGATTTACTGCTGTTACCATCA
8
## 103    CTNNB1 GCAATTTGCCAAGTTTCTTTAGCATTTGGCCCTGGATTACGCTGGACCCC
7
## 413  LOC23117 CCCTTCCTACATTCTTGTTTTCATTTTTTCGGAGGAAGAGGAGTTGCTAG
8
## 523 LOC653086 AGCAGCACATCGTCATTTTACAATTGAGAAACATGGAGACTCCAAATGGA
7
## 421  LOC23117 GGGAAGTACATGGGGCAGATGGAAGAACCTGAGATAATCGCAAGGATGGC
8
## 40       BDNF TCAGACCCCTCAGGCCACTGCTGTTCCTGTCACACATTCCTGCAAAGGAC
7
## 607     PLEC1 CTCCGTCTGCCCCGTGGGCTCCTGCCACCGTCCCCGATGAAGATCGTGCC
8
## 453 LOC339047 TTTCCTGAAATGGAGCTTTGCTCTTGTTGCCCAGGCCGTAGTGCAATGGC
7
## 599     PLEC1 GCCTTTGCCTCGCCGAGGGAGGTCTTGCTGGAGCGGCCGTGCTGGCTGGA
8
## 12       BDNF ATGTACGTGGGGGATTCTTGACTCGGGTTAGTCTCTGGGGATGCAGAGCC
7
## 583     PLEC1 CAGCCCTGGGGACACACTGCCCTGGAACCTTGGGAAAACGCAGCGGAGCC
8
## 190     DDX12 CGTTGCTACAAGCTGTTTTTTGAATGTCTCTACACAGTCCAGGCAGGAAG
10
## 481 LOC339047 TCTGTATGGACCCTGCCAAGCTCTGCCCCTCTGCCCCTGCATTGGGGCGC
7
```

```
## 54       BDNF TGGGGAGACGAGATTTTAAGACACTTGAGTCTCCAGGACAGCAAAGGCAC
7
## 160     DDX12 CAGACTTCTCGCTTCCTTTCTGCTGGGCCTCTGAGGGGTCATGGGGCCAT
10
## 304  KIAA0692 AAGTGGTGCCTGGCTGTCCCTATACTGTGCTGCTGGGTGTTCCAGCCTGT
9
## 224       DMD GCAGCCAACTTATTGGCATGATGGAGTGACAGGAAAAACAGCTGGCATGG
7
## 200     DDX12 TTACTGGGGATGGTATTTAGGAGCCAGGAAAGCCGGTGCATTCCTAGTGA
10
## 544 LOC653086 TACCTGGCCTATCTTTCATAGGTTATATAAATTCCTTGGTTCCCAGTTTT
7
## 217       DMD GGGTTTTCTCAGGATTGCTATGCAACAGGATCAGTGCTGTAGTGCCCGGT
7
## 120     DDX12 ACATGTGCTGTCACTGGAACTTGCTCTTTTCACTCAGCAGCCAGAGGGTC
10
## 110     DDX12 CCAGTCCCTGACTACAGAGGATTTCCCCAAAGTCCCTGGCTGTGAGGTTC
10
## 130     DDX12 AAACGTTACAGTGTTCCGATGAGACACAGTAGGCAGTACTTGGGAGGGTC
10
## 368 LOC202134 GACCAAAGCAGGACAATTGCTTGATCCCAGGAGTTTAAGACCAGCCGGGG
7
## 537 LOC653086 AAGGACTCAGATGCAGGGTCTTCTCTGCTCCCCGTCACACAGAGGGTGGC
7
## 277  KIAA0692 CAGGCGACTGGGTAGCAGATGTGGAAGCTGATGGTTAGGCCCAGGGCATG
9
## 488 LOC339047 GACCTGTAGCTAAACCTTCCACCAGCGCTTGAGAACTTAATTTGAACCGG
7
## 238       DMD GCACTCCGACTACATCAGGAGAAGATGTTCGAGACTTTGCCAAGGTACTA
7
## 354 LOC202134 CCACGCCGGCAAAGAAATTGGAAGACTCCACCATTACAGGCAGCCACCAG
7
## 26       BDNF CTTGCTGTGGTCTCTTTGTGGCAGAAGTGTTTCATGCATGGCAGCAGGCC
7
## 567     PLEC1 AGCCTCTGTTCCCCTAGTAAGTGCCTTCCATGTCGGCCTCTAACCCCAGG
8
## 347 LOC202134 CCTGTTTGGATCACATGGTCTTGTCCTGATAACTTGGAAGAGGTTGCTTC
7
## 1       ABCA8 GCCAACCTCCTCTCACAGCCTCTGTATCTCTGCAGGCCATACTGGTTCCA
1
## 3       ADH1A CAGATGTTTTCCCTTGTGGCAGTCTTCAGCCTCCTCTACCCTACATGATC
1
## 6         FOS CCCAGTGACACTTCAGAGAGCTGGTAGTTAGTAGCATGTTGAGCCAGGCC
1
## 7        FOSB TGACTGTCCCTGCCAATGCTCCAGCTGTCGTCTGACTCTGGGTTCGTTGG
1
## 9       KRT19 CAGCTGGGCGATGTGCGAGCTGATAGTGAGCGGCAGAATCAGGAGTACCA
1
##     foldChangeMean_UL_to_nonUL
## 8                   18.0591346
```

```
## 11                  14.9511840
## 2                   11.9638706
## 4                   10.8264232
## 10                  10.7817440
## 68                   1.5689829
## 89                   1.5248323
## 82                   1.2503004
## 231                  1.1977828
## 210                  1.1332249
## 96                   1.1098758
## 140                  1.1076674
## 397                  1.0971923
## 361                  1.0775532
## 286                  1.0762308
## 61                   1.0740152
## 516                  1.0680820
## 33                   1.0451334
## 551                  1.0433765
## 445                  1.0317830
## 474                  1.0291369
## 331                  1.0250419
## 559                  1.0243757
## 530                  1.0237370
## 259                  1.0212595
## 429                  1.0204791
## 252                  1.0202573
## 268                  1.0150804
## 19                   1.0137555
## 340                  1.0132932
## 405                  1.0108892
## 245                  1.0100160
## 322                  1.0098454
## 375                  1.0093062
## 382                  1.0091169
## 591                  1.0071980
## 47                   1.0067778
## 575                  1.0042547
## 313                  1.0038554
## 495                  1.0036545
## 170                  1.0019589
## 460                  1.0005056
## 437                  0.9998782
## 502                  0.9991520
## 509                  0.9982094
## 467                  0.9981707
## 150                  0.9980314
## 75                   0.9977081
## 295                  0.9976800
## 180                  0.9964084
## 389                  0.9937542
```

```
## 103                  0.9930219
## 413                  0.9927896
## 523                  0.9924044
## 421                  0.9921585
## 40                   0.9911345
## 607                  0.9868791
## 453                  0.9867059
## 599                  0.9860355
## 12                   0.9821732
## 583                  0.9816619
## 190                  0.9793547
## 481                  0.9791535
## 54                   0.9767836
## 160                  0.9762911
## 304                  0.9737579
## 224                  0.9735481
## 200                  0.9725752
## 544                  0.9708972
## 217                  0.9699886
## 120                  0.9694418
## 110                  0.9672238
## 130                  0.9664926
## 368                  0.9545894
## 537                  0.9529309
## 277                  0.9406746
## 488                  0.9392117
## 238                  0.9047898
## 354                  0.8995475
## 26                   0.8520654
## 567                  0.7581050
## 347                  0.4746867
## 1                    0.1977871
## 3                    0.1961796
## 6                    0.1860882
## 7                    0.1801972
## 9                    0.1599868

write.csv(KG1,'keyGenes_UL_FCs_CNVs.csv', row.names=FALSE)
```

Order by gene count, then by fold change.

```
KG2 <- KG1[with(KG1, order(gene_count, foldChangeMean_UL_to_nonUL,
decreasing=TRUE)),]
```

Lets add in a fold change of the median value ratios of UL to non-UL samples to compare.

```
colnames(KG2)[18] <- 'foldChange_Mean'
KG2$foldChange_Median <- KG2$UL_Median/KG2$nonUL_Median
```

Lets look at some of these copy number variants of one gene with seven copy number variants or CNVs and see where the changes in the nucleotide sequences occur. Copy

number variations in nucleotides can have short repeats, jumps in sequence, insertions, or deletions of a gene. I have been calling these CNVs genotypes, which are the traits and alleles responsible for the physical traits or phenotypes of an organism. Some CNVs are responsible for diseases, and in tumors there are many different CNVs that are found to be responsible. A uterine leiomyoma or fibroid is a benign tumor. These samples were taken from uterus tissue with these uternine tumors and the same neighboring uterine tissue without uterine tumors.

```
CTNNB1 <- subset(KG2, KG2$Symbol=='CTNNB1')
CTNNB1_seq <- CTNNB1[,1:2]
```

Add in a column to describe the length of the nucleotides in each copy number variant nucleotide strand.

```
CTNNB1_seq$SEQUENCE <- as.character(CTNNB1$SEQUENCE)

CTNNB1_seq$nChar <- nchar(CTNNB1_seq$SEQUENCE)
```

Lets look at the CNVs of the CTNNB1 gene.

```
CTNNB1_seq
```

```
##     Symbol                                                SEQUENCE nChar
## 68  CTNNB1 AGCTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAG    50
## 89  CTNNB1 CTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAGCT    50
## 82  CTNNB1 AGTCTCTCGTAGTGTTAAGTTATAGTGAATACTGCTACAGCAATTTCTAA    50
## 96  CTNNB1 GCCTCTTGCACTCTGAATTGGGAATGTTTGCACCACAGTGGGGGGCTTGC    50
## 61  CTNNB1 CAAACTTTACAGAGGAGAATGCCCTGTTTGTTAACCATGTTTCTTTTGGC    50
## 75  CTNNB1 CAGGAATCTAGTCTGGATGACTGCTTCTGGAGCCTGGATGCAGTACCATT    50
## 103 CTNNB1 GCAATTTGCCAAGTTTCTTTAGCATTTGGCCCTGGATTACGCTGGACCCC    50
```

From the above, some of the CNVs make you wonder if they are even the same gene. The first two have the same pattern of 'CTGCAGGG' then some variations. Then its not obvious what the other sequence alignments are. We could go back to the cytoband location and where the gene starts to see if there is more information.

Lets get the SEQUENCE, protein product, and cytoband columns from the original UL1 table.

```
cytoband <- UL1[,c(15,20,24)]
```

Now combine with the CTNNB1_seq and the KG2 table.

```
CTNNB1_cyto <- merge(cytoband, CTNNB1_seq, by.x='SEQUENCE', by.y='SEQUENCE')
KG2_cyto <- merge(cytoband, KG2, by.x='SEQUENCE', by.y='SEQUENCE')
```

Now lets look at the KG2_cyto table to see where these CNVs are located within the cytoband of each gene location.

```
KG3 <- KG2_cyto[with(KG2_cyto, order(gene_count,foldChange_Mean, decreasing =
TRUE)),]
KG3[,1:5]
```

```
##                                                  SEQUENCE Protein_Product
## 37 CCGCCGGGCTGCTTTTTCCTTGGATGCCCATCAGGACGCCTCAGTTCTCT     XP_936926.1
## 56 GCAGGGGAGATTGGGTTTAGGGGCTTTCCTGGTCTGCATTCTGCTACAGC     XP_937020.1
## 26 CAGGGCAGGAACCACGTCTTTACAGTTTGATGTTCCCAGAGCTGACCCAG     XP_936919.1
## 77 TCTCCTGCCCCCTCCGGAAGCTTGGATGCCCCTCCACACCCTCTTGATCT     XP_936947.1
## 41 CGTTGCTACAAGCTGTTTTTTGAATGTCTCTACACAGTCCAGGCAGGAAG     XP_937000.1
## 20 CAGACTTCTCGCTTCCTTTCTGCTGGGCCTCTGAGGGGTCATGGGGCCAT     XP_936988.1
## 84 TTACTGGGGATGGTATTTAGGAGCCAGGAAAGCCGGTGCATTCCTAGTGA     XP_936932.1
## 6  ACATGTGCTGTCACTGGAACTTGCTCTTTTCACTCAGCAGCCAGAGGGTC     XP_936976.1
## 30 CCAGTCCCTGACTACAGAGGATTTCCCCAAAGTCCCTGGCTGTGAGGTTC     XP_936952.1
## 1  AAACGTTACAGTGTTCCGATGAGACACAGTAGGCAGTACTTGGGAGGGTC     XP_936980.1
## 72 TAAGTGCAGTGAGCTCTGGCGGAAACCACCCTCTGCCCCGTCTGTTGGAT     XP_935983.1
## 28 CATTGTAATGATAAGGAAATGTTGCGATCAAATAAGATTTAGACACACTT     XP_935991.1
## 49 GAGTGCTGGGAAGGTTAATGTTAAATGGGTTGTGTGTCGGGGAGGGTACA     XP_935974.1
## 51 GCAAATGTAACTCAGGGGTTTGGGGCCAGAGGAAGAGGGAGAAGGTGGCC     XP_935936.1
## 10 AGCTCCACCTTGACCCAGCCTCACAACAAAAAGTTTGTGTATGACCAGGC     XP_935967.1
## 50 GATCACAGGCACAGGGAAGCCACAAGGAGCTCTGTATGAGTTGTGTTTGC     XP_935893.1
## 71 GTTGTTCTGGACGATCTTCGGGATCCTCTGGGGCACTGTGACACTCGGAG     XP_936004.1
## 4  AAGTGGTGCCTGGCTGTCCCTATACTGTGCTGCTGGGTGTTCCAGCCTGT     XP_935903.1
## 25 CAGGCGACTGGGTAGCAGATGTGGAAGCTGATGGTTAGGCCCAGGGCATG     XP_935881.1
## 74 TCAACCACATCCTTCAAAAGGACTATGCCTGTTTATAAGCCCAGCTGTTT     XP_938957.1
## 33 CCCGACGAGCAGGACTTCATCCAGGCCTACGAGGAGGTGCGCGAGAAGTA     NP_958780.1
## 83 TGTCGTTTCCTCCATTCTTCACCAAAACATCAGCGTACATAGGCACATGG     XP_938806.1
## 35 CCCTCGGGCAGCCTGTTTCCCTCCCTGGTGGTTGTGGGTCACGTTGTCAC     NP_958784.1
## 64 GGCTCCTCTTTGGGCTCCTACTGGAATTTATCAGCCATCAGTGCATCTCT     XP_938917.1
## 2  AAAGCAGTGGTTTTCAGCTGCCAGAGGCCTGAGAGAGTTTGGGCATACTC     XP_938927.1
## 38 CCGGGCCTTCTCGTGGTACCCTGCCTGCTGCCTTTGCCCCCGCACTGACT     NP_958782.1
## 63 GGCGCAGACATGGACCCCTCGCGAGCCATCCAGAACGAGATCAGCTCCCT     NP_958781.1
## 65 GGCTCTGTTGGAATCCGCATAGTGTGGAAATGAGTTTGCCCTGGAAAGGG     XP_938916.1
## 45 CTGGCCTTCCCTCATCAGCCGTAAATGATGATTTACTGCTGTTACCATCA     XP_939002.1
## 36 CCCTTCCTACATTCTTGTTTTCATTTTTTCGGAGGAAGAGGAGTTGCTAG     XP_938960.1
## 66 GGGAAGTACATGGGGCAGATGGAAGAACCTGAGATAATCGCAAGGATGGC     XP_938807.1
## 42 CTCCGTCTGCCCCGTGGGCTCCTGCCACCGTCCCCGATGAAGATCGTGCC     NP_958783.1
## 61 GCCTTTGCCTCGCCGAGGGAGGTCTTGCTGGAGCGGCCGTGCTGGCTGGA     NP_958785.1
## 22 CAGCCCTGGGGACACACTGCCCTGGAACCTTGGGAAAACGCAGCGGAGCC     NP_000436.2
## 9  AGCCTCTGTTCCCCTAGTAAGTGCCTTCCATGTCGGCCTCTAACCCCAGG     NP_958786.1
## 11 AGCTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAG     XP_950743.1
## 44 CTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAGCT     NP_001895.1
## 13 AGTCTCTCGTAGTGTTAAGTTATAGTGAATACTGCTACAGCAATTTCTAA     NP_001895.1
## 27 CAGTGTTGGGATCACTCACTTTCCCCCTACAGGACTCAGATCTGGGAGGC     NP_003997.1
## 43 CTCCTCTCAGCTGAACACCCTCCTTTCACTCCCAAATGCAAACAGTCTCT     NP_004010.1
## 60 GCCTCTTGCACTCTGAATTGGGAATGTTTGCACCACAGTGGGGGGCTTGC     XP_950747.1
## 57 GCCAAAGGAATGGGCTCCAGACACCCCCTCTTCCAGAGCAAGGATGAAGG     XP_937236.1
## 19 CAAACTTTACAGAGGAGAATGCCCTGTTTGTTAACCATGTTTCTTTTGGC     XP_950746.1
## 80 TGATGTGTCACGCCACTGTACTCCAGCCTGACGGCAGAGCGAGACTCCAT     XP_936088.1
## 34 CCCTCCACCTCCTGCTCGGGGGGCTTTAATGAGACACCCACCGCTGCTGT     NP_001700.2
```

```
## 7  ACTGCCTGTGTGGCTCCTTGAGTGCGCGGAGGCCAAAGCTGAGATGACTT     XP_937640.1
## 69 GGTGTGCTCTGGTATGTAATGACAATATGTGAACAAACCTGTGGAATTAA     XP_936056.1
## 76 TCTATCAACAGAGCTGAATGAGTGCCAGGAAGCTGCGAAATCTGTCTTAC     NP_004003.1
## 5  AATAATAGAGTGTGGGAGTTTTGGGGCCGAAGTCTTTCCCGGAGCAGCTG     NP_733929.1
## 18 CAAACCCTTGAAGACATTTCAGGGCCATGCTCACTTGGGAGGGTTTGAGG     XP_937228.1
## 31 CCATTCAGAAGAATGATAAATGCCACAAGCATTTGGAAACAGGCTTCCCT     NP_004001.1
## 15 AGTGGGCAGAATGATGAGGGAAGTGGGCACGTGCCCATGTTCTTCTTGGC     XP_937222.1
## 85 TTCATCCAGGCCTGCGCCGGTGTTCACAGTGGTCCTCATCTAAGCCAGCC     XP_937214.1
## 62 GCTCGCTGAAGTTGGCTTCCTAGCGGTGTAGGCTGGAATAGACTCTTGGC     NP_733928.1
## 86 TTTCAGGCCCATGGCAGAGGGTGGGCTCAGGAGGGCCATCGTGGGTGTCC     XP_937694.1
## 14 AGTGCCCACATCACACAGCATCTAGCACGTAACTGCACCCCGGGAGTCGT     XP_937456.1
## 16 AGTGTTGGGACTACAGGTGTGTGTTACTGCTCCCAGCTGGGAGGCAGGCT     XP_936104.1
## 70 GTGAGCCTGTTTCATCATCTGTAAACTTTGAATAATGATACCTACCCCGC     XP_936038.1
## 40 CGCCCTGAAAGGACCAGGACATGCGGGTGCGGTGGCTGCTCTTTTGGCTC     XP_937724.1
## 24 CAGGAATCTAGTCTGGATGACTGCTTCTGGAGCCTGGATGCAGTACCATT     XP_950748.1
## 53 GCAATTTGCCAAGTTTCTTTAGCATTTGGCCCTGGATTACGCTGGACCCC     XP_947138.1
## 8  AGCAGCACATCGTCATTTTACAATTGAGAAACATGGAGACTCCAAATGGA     XP_936080.1
## 75 TCAGACCCCTCAGGCCACTGCTGTTCCTGTCACACATTCCTGCAAAGGAC     NP_733931.1
## 87 TTTCCTGAAATGGAGCTTTGCTCTTGTTGCCCAGGCCGTAGTGCAATGGC     XP_937537.1
## 17 ATGTACGTGGGGGATTCTTGACTCGGGTTAGTCTCTGGGGATGCAGAGCC     NP_733930.1
## 78 TCTGTATGGACCCTGCCAAGCTCTGCCCCTCTGCCCCTGCATTGGGGCGC     XP_937505.1
## 82 TGGGGAGACGAGATTTTAAGACACTTGAGTCTCCAGGACAGCAAAGGCAC     NP_733927.1
## 55 GCAGCCAACTTATTGGCATGATGGAGTGACAGGAAAAACAGCTGGCATGG     NP_000100.2
## 73 TACCTGGCCTATCTTTCATAGGTTATATAAATTCCTTGGTTCCCAGTTTT     XP_936049.1
## 67 GGGTTTTCTCAGGATTGCTATGCAACAGGATCAGTGCTGTAGTGCCCGGT     NP_004005.1
## 47 GACCAAAGCAGGACAATTGCTTGATCCCAGGAGTTTAAGACCAGCCGGGG     XP_932593.1
## 3  AAGGACTCAGATGCAGGGTCTTCTCTGCTCCCCGTCACACAGAGGGTGGC     XP_936046.1
## 48 GACCTGTAGCTAAACCTTCCACCAGCGCTTGAGAACTTAATTTGAACCGG     XP_937490.1
## 54 GCACTCCGACTACATCAGGAGAAGATGTTCGAGACTTTGCCAAGGTACTA     NP_004010.1
## 29 CCACGCCGGCAAAGAAATTGGAAGACTCCACCATTACAGGCAGCCACCAG     XP_937214.1
## 46 CTTGCTGTGGTCTCTTTGTGGCAGAAGTGTTTCATGCATGGCAGCAGGCC     NP_001700.2
## 39 CCTGTTTGGATCACATGGTCTTGTCCTGATAACTTGGAAGAGGTTGCTTC     XP_371783.3
## 81 TGCGAGACCTGGGTGTCCAACCTGCGCTACAACCACATGCTGCGGAAGAA     NP_003827.3
## 52 GCAACGCTCCTCTGAAATGCTTGTCTTTTTTCTGTTGCCGAAATAGCTGG     NP_061159.1
## 59 GCCCCAGCAAGCCTCCCTCCATCCTCCAGTGGGAAACTGTTGATGGTGTT     NP_006202.1
## 68 GGTATTGCTGATCGTATGCAGAAGGAAATCACTGCTCTGGCTCCTAGCAC     NP_005150.1
## 12 AGGCCCTGGAGGCTGCAACATACCTCAATCCTGTCCCAGGCCGGATCCTC     NP_005931.2
## 58 GCCAACCTCCTCTCACAGCCTCTGTATCTCTGCAGGCCATACTGGTTCCA     NP_009099.1
## 21 CAGATGTTTTCCCTTGTGGCAGTCTTCAGCCTCCTCTACCCTACATGATC     NP_000658.1
## 32 CCCAGTGACACTTCAGAGAGCTGGTAGTTAGTAGCATGTTGAGCCAGGCC     NP_005243.1
## 79 TGACTGTCCCTGCCAATGCTCCAGCTGTCGTCTGACTCTGGGTTCGTTGG     NP_006723.1
## 23 CAGCTGGGCGATGTGCGAGCTGATAGTGAGCGGCAGAATCAGGAGTACCA     NP_002267.2
##          Cytoband    Symbol gene_count
## 37      12p13.31a     DDX12         10
## 56      12p13.31a     DDX12         10
## 26      12p13.31a     DDX12         10
## 77      12p13.31a     DDX12         10
## 41      12p13.31a     DDX12         10
## 20      12p13.31a     DDX12         10
## 84      12p13.31a     DDX12         10
```

```
## 6       12p13.31a     DDX12         10
## 30      12p13.31a     DDX12         10
## 1       12p13.31a     DDX12         10
## 72      12q24.33d  KIAA0692          9
## 28      12q24.33d  KIAA0692          9
## 49      12q24.33d  KIAA0692          9
## 51      12q24.33d  KIAA0692          9
## 10      12q24.33d  KIAA0692          9
## 50      12q24.33d  KIAA0692          9
## 71      12q24.33d  KIAA0692          9
## 4       12q24.33d  KIAA0692          9
## 25      12q24.33d  KIAA0692          9
## 74       16p12.2a  LOC23117          8
## 33        8q24.3g     PLEC1          8
## 83       16p12.2a  LOC23117          8
## 35        8q24.3g     PLEC1          8
## 64       16p12.2a  LOC23117          8
## 2        16p12.2a  LOC23117          8
## 38        8q24.3g     PLEC1          8
## 63        8q24.3g     PLEC1          8
## 65       16p12.2a  LOC23117          8
## 45       16p12.2a  LOC23117          8
## 36       16p12.2a  LOC23117          8
## 66       16p12.2a  LOC23117          8
## 42        8q24.3g     PLEC1          8
## 61        8q24.3g     PLEC1          8
## 22        8q24.3g     PLEC1          8
## 9         8q24.3g     PLEC1          8
## 11        3p22.1b    CTNNB1          7
## 44        3p22.1b    CTNNB1          7
## 13        3p22.1b    CTNNB1          7
## 27 Xp21.2a-p21.1d       DMD          7
## 43 Xp21.2a-p21.1d       DMD          7
## 60        3p22.1b    CTNNB1          7
## 57        5q35.2d LOC202134          7
## 19        3p22.1b    CTNNB1          7
## 80             NA LOC653086          7
## 34       11p14.1d      BDNF          7
## 7       16p13.11b LOC339047          7
## 69             NA LOC653086          7
## 76 Xp21.2a-p21.1d       DMD          7
## 5        11p14.1d      BDNF          7
## 18        5q35.2d LOC202134          7
## 31 Xp21.2a-p21.1d       DMD          7
## 15        5q35.2d LOC202134          7
## 85        5q35.2d LOC202134          7
## 62       11p14.1d      BDNF          7
## 86      16p13.11b LOC339047          7
## 14      16p13.11b LOC339047          7
## 16             NA LOC653086          7
```

```
## 70             NA LOC653086          7
## 40      16p13.11b LOC339047          7
## 24        3p22.1b    CTNNB1          7
## 53        3p22.1b    CTNNB1          7
## 8              NA LOC653086          7
## 75       11p14.1d      BDNF          7
## 87      16p13.11b LOC339047          7
## 17       11p14.1d      BDNF          7
## 78      16p13.11b LOC339047          7
## 82       11p14.1d      BDNF          7
## 55 Xp21.2a-p21.1d       DMD          7
## 73             NA LOC653086          7
## 67 Xp21.2a-p21.1d       DMD          7
## 47        5q35.2d LOC202134          7
## 3              NA LOC653086          7
## 48      16p13.11b LOC339047          7
## 54 Xp21.2a-p21.1d       DMD          7
## 29        5q35.2d LOC202134          7
## 46       11p14.1d      BDNF          7
## 39        5q35.2d LOC202134          7
## 81       14q32.2b      DLK1          2
## 52       15q25.1b  KIAA1199          1
## 59        8q12.1b      PENK          1
## 68         15q14a      ACTC          1
## 12      22q11.23a     MMP11          1
## 58       17q24.2c     ABCA8          1
## 21          4q23b     ADH1A          1
## 32       14q24.3b       FOS          1
## 79      19q13.32a      FOSB          1
## 23       17q21.2b     KRT19          1
```

```
CTNNB1_b <- subset(KG3, KG3$Symbol=='CTNNB1')
CTNNB1_b[,c(1:5,20:21)]
```

```
##                                                SEQUENCE Protein_Product
Cytoband
## 11 AGCTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAG     XP_950743.1
3p22.1b
## 44 CTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAGCT     NP_001895.1
3p22.1b
## 13 AGTCTCTCGTAGTGTTAAGTTATAGTGAATACTGCTACAGCAATTTCTAA     NP_001895.1
3p22.1b
## 60 GCCTCTTGCACTCTGAATTGGGAATGTTTGCACCACAGTGGGGGGCTTGC     XP_950747.1
3p22.1b
## 19 CAAACTTTACAGAGGAGAATGCCCTGTTTGTTAACCATGTTTCTTTTGGC     XP_950746.1
3p22.1b
## 24 CAGGAATCTAGTCTGGATGACTGCTTCTGGAGCCTGGATGCAGTACCATT     XP_950748.1
3p22.1b
## 53 GCAATTTGCCAAGTTTCTTTAGCATTTGGCCCTGGATTACGCTGGACCCC     XP_947138.1
3p22.1b
```

```
##    Symbol gene_count foldChange_Mean foldChange_Median
## 11 CTNNB1          7       1.5689829         1.5744314
## 44 CTNNB1          7       1.5248323         1.4655024
## 13 CTNNB1          7       1.2503004         1.1602376
## 60 CTNNB1          7       1.1098758         1.0850642
## 19 CTNNB1          7       1.0740152         1.0471422
## 24 CTNNB1          7       0.9977081         0.9924948
## 53 CTNNB1          7       0.9930219         0.9818097
```

The cytoband location of each of these CNVs for CTNNB1 is the same location on chromosome 3 on the p strand/direction along 22.1b. Also, the fold change for the mean and median values for the first listed CNVs changed by 16-57 percent more in UL compared to non-UL samples. This could mean that these four CNVs of the gene CTNNB1 offer some clues as to what mutations or changes impact risk in developing uterine leiomyomas for some females.

Lets order the key genes by fold change median then by CNVs.

```
KG4 <- KG3[with(KG3, order(foldChange_Median, gene_count, decreasing =
TRUE)),]
KG4[,c(1:5,21)]
```

```
##                                                SEQUENCE Protein_Product
## 52 GCAACGCTCCTCTGAAATGCTTGTCTTTTTTCTGTTGCCGAAATAGCTGG     NP_061159.1
## 68 GGTATTGCTGATCGTATGCAGAAGGAAATCACTGCTCTGGCTCCTAGCAC     NP_005150.1
## 12 AGGCCCTGGAGGCTGCAACATACCTCAATCCTGTCCCAGGCCGGATCCTC     NP_005931.2
## 81 TGCGAGACCTGGGTGTCCAACCTGCGCTACAACCACATGCTGCGGAAGAA     NP_003827.3
## 59 GCCCCAGCAAGCCTCCCTCCATCCTCCAGTGGGAAACTGTTGATGGTGTT     NP_006202.1
## 11 AGCTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAG     XP_950743.1
## 44 CTGCAGGGGTCCTCTGTGAACTTGCTCAGGACAAGGAAGCTGCAGAAGCT     NP_001895.1
## 43 CTCCTCTCAGCTGAACACCCTCCTTTCACTCCCAAATGCAAACAGTCTCT     NP_004010.1
## 13 AGTCTCTCGTAGTGTTAAGTTATAGTGAATACTGCTACAGCAATTTCTAA     NP_001895.1
## 27 CAGTGTTGGGATCACTCACTTTCCCCCTACAGGACTCAGATCTGGGAGGC     NP_003997.1
## 60 GCCTCTTGCACTCTGAATTGGGAATGTTTGCACCACAGTGGGGGGCTTGC     XP_950747.1
## 74 TCAACCACATCCTTCAAAAGGACTATGCCTGTTTATAAGCCCAGCTGTTT     XP_938957.1
## 10 AGCTCCACCTTGACCCAGCCTCACAACAAAAAGTTTGTGTATGACCAGGC     XP_935967.1
## 19 CAAACTTTACAGAGGAGAATGCCCTGTTTGTTAACCATGTTTCTTTTGGC     XP_950746.1
## 33 CCCGACGAGCAGGACTTCATCCAGGCCTACGAGGAGGTGCGCGAGAAGTA     NP_958780.1
## 80 TGATGTGTCACGCCACTGTACTCCAGCCTGACGGCAGAGCGAGACTCCAT     XP_936088.1
## 31 CCATTCAGAAGAATGATAAATGCCACAAGCATTTGGAAACAGGCTTCCCT     NP_004001.1
## 64 GGCTCCTCTTTGGGCTCCTACTGGAATTTATCAGCCATCAGTGCATCTCT     XP_938917.1
## 69 GGTGTGCTCTGGTATGTAATGACAATATGTGAACAAACCTGTGGAATTAA     XP_936056.1
## 28 CATTGTAATGATAAGGAAATGTTGCGATCAAATAAGATTTAGACACACTT     XP_935991.1
## 76 TCTATCAACAGAGCTGAATGAGTGCCAGGAAGCTGCGAAATCTGTCTTAC     NP_004003.1
## 16 AGTGTTGGGACTACAGGTGTGTGTTACTGCTCCCAGCTGGGAGGCAGGCT     XP_936104.1
## 37 CCGCCGGGCTGCTTTTTCCTTGGATGCCCATCAGGACGCCTCAGTTCTCT     XP_936926.1
## 86 TTTCAGGCCCATGGCAGAGGGTGGGCTCAGGAGGGCCATCGTGGGTGTCC     XP_937694.1
## 49 GAGTGCTGGGAAGGTTAATGTTAAATGGGTTGTGTGTCGGGGAGGGTACA     XP_935974.1
## 83 TGTCGTTTCCTCCATTCTTCACCAAAACATCAGCGTACATAGGCACATGG     XP_938806.1
## 35 CCCTCGGGCAGCCTGTTTCCCTCCCTGGTGGTTGTGGGTCACGTTGTCAC     NP_958784.1
```

```
## 57 GCCAAAGGAATGGGCTCCAGACACCCCCTCTTCCAGAGCAAGGATGAAGG     XP_937236.1
## 14 AGTGCCCACATCACACAGCATCTAGCACGTAACTGCACCCCGGGAGTCGT     XP_937456.1
## 50 GATCACAGGCACAGGGAAGCCACAAGGAGCTCTGTATGAGTTGTGTTTGC     XP_935893.1
## 70 GTGAGCCTGTTTCATCATCTGTAAACTTTGAATAATGATACCTACCCCGC     XP_936038.1
## 38 CCGGGCCTTCTCGTGGTACCCTGCCTGCTGCCTTTGCCCCCGCACTGACT     NP_958782.1
## 65 GGCTCTGTTGGAATCCGCATAGTGTGGAAATGAGTTTGCCCTGGAAAGGG     XP_938916.1
## 78 TCTGTATGGACCCTGCCAAGCTCTGCCCCTCTGCCCCTGCATTGGGGCGC     XP_937505.1
## 85 TTCATCCAGGCCTGCGCCGGTGTTCACAGTGGTCCTCATCTAAGCCAGCC     XP_937214.1
## 7  ACTGCCTGTGTGGCTCCTTGAGTGCGCGGAGGCCAAAGCTGAGATGACTT     XP_937640.1
## 5  AATAATAGAGTGTGGGAGTTTTGGGGCCGAAGTCTTTCCCGGAGCAGCTG     NP_733929.1
## 18 CAAACCCTTGAAGACATTTCAGGGCCATGCTCACTTGGGAGGGTTTGAGG     XP_937228.1
## 62 GCTCGCTGAAGTTGGCTTCCTAGCGGTGTAGGCTGGAATAGACTCTTGGC     NP_733928.1
## 66 GGGAAGTACATGGGGCAGATGGAAGAACCTGAGATAATCGCAAGGATGGC     XP_938807.1
## 41 CGTTGCTACAAGCTGTTTTTTGAATGTCTCTACACAGTCCAGGCAGGAAG     XP_937000.1
## 45 CTGGCCTTCCCTCATCAGCCGTAAATGATGATTTACTGCTGTTACCATCA     XP_939002.1
## 51 GCAAATGTAACTCAGGGGTTTGGGGCCAGAGGAAGAGGGAGAAGGTGGCC     XP_935936.1
## 40 CGCCCTGAAAGGACCAGGACATGCGGGTGCGGTGGCTGCTCTTTTGGCTC     XP_937724.1
## 84 TTACTGGGGATGGTATTTAGGAGCCAGGAAAGCCGGTGCATTCCTAGTGA     XP_936932.1
## 17 ATGTACGTGGGGGATTCTTGACTCGGGTTAGTCTCTGGGGATGCAGAGCC     NP_733930.1
## 20 CAGACTTCTCGCTTCCTTTCTGCTGGGCCTCTGAGGGGTCATGGGGCCAT     XP_936988.1
## 24 CAGGAATCTAGTCTGGATGACTGCTTCTGGAGCCTGGATGCAGTACCATT     XP_950748.1
## 36 CCCTTCCTACATTCTTGTTTTCATTTTTTCGGAGGAAGAGGAGTTGCTAG     XP_938960.1
## 15 AGTGGGCAGAATGATGAGGGAAGTGGGCACGTGCCCATGTTCTTCTTGGC     XP_937222.1
## 26 CAGGGCAGGAACCACGTCTTTACAGTTTGATGTTCCCAGAGCTGACCCAG     XP_936919.1
## 2  AAAGCAGTGGTTTTCAGCTGCCAGAGGCCTGAGAGAGTTTGGGCATACTC     XP_938927.1
## 75 TCAGACCCCTCAGGCCACTGCTGTTCCTGTCACACATTCCTGCAAAGGAC     NP_733931.1
## 56 GCAGGGGAGATTGGGTTTAGGGGCTTTCCTGGTCTGCATTCTGCTACAGC     XP_937020.1
## 55 GCAGCCAACTTATTGGCATGATGGAGTGACAGGAAAAACAGCTGGCATGG     NP_000100.2
## 8  AGCAGCACATCGTCATTTTACAATTGAGAAACATGGAGACTCCAAATGGA     XP_936080.1
## 42 CTCCGTCTGCCCCGTGGGCTCCTGCCACCGTCCCCGATGAAGATCGTGCC     NP_958783.1
## 53 GCAATTTGCCAAGTTTCTTTAGCATTTGGCCCTGGATTACGCTGGACCCC     XP_947138.1
## 61 GCCTTTGCCTCGCCGAGGGAGGTCTTGCTGGAGCGGCCGTGCTGGCTGGA     NP_958785.1
## 72 TAAGTGCAGTGAGCTCTGGCGGAAACCACCCTCTGCCCCGTCTGTTGGAT     XP_935983.1
## 71 GTTGTTCTGGACGATCTTCGGGATCCTCTGGGGCACTGTGACACTCGGAG     XP_936004.1
## 22 CAGCCCTGGGGACACACTGCCCTGGAACCTTGGGAAAACGCAGCGGAGCC     NP_000436.2
## 73 TACCTGGCCTATCTTTCATAGGTTATATAAATTCCTTGGTTCCCAGTTTT     XP_936049.1
## 77 TCTCCTGCCCCCTCCGGAAGCTTGGATGCCCCTCCACACCCTCTTGATCT     XP_936947.1
## 63 GGCGCAGACATGGACCCCTCGCGAGCCATCCAGAACGAGATCAGCTCCCT     NP_958781.1
## 67 GGGTTTTCTCAGGATTGCTATGCAACAGGATCAGTGCTGTAGTGCCCGGT     NP_004005.1
## 82 TGGGGAGACGAGATTTTAAGACACTTGAGTCTCCAGGACAGCAAAGGCAC     NP_733927.1
## 4  AAGTGGTGCCTGGCTGTCCCTATACTGTGCTGCTGGGTGTTCCAGCCTGT     XP_935903.1
## 48 GACCTGTAGCTAAACCTTCCACCAGCGCTTGAGAACTTAATTTGAACCGG     XP_937490.1
## 87 TTTCCTGAAATGGAGCTTTGCTCTTGTTGCCCAGGCCGTAGTGCAATGGC     XP_937537.1
## 6  ACATGTGCTGTCACTGGAACTTGCTCTTTTCACTCAGCAGCCAGAGGGTC     XP_936976.1
## 34 CCCTCCACCTCCTGCTCGGGGGGCTTTAATGAGACACCCACCGCTGCTGT     NP_001700.2
## 30 CCAGTCCCTGACTACAGAGGATTTCCCCAAAGTCCCTGGCTGTGAGGTTC     XP_936952.1
## 1  AAACGTTACAGTGTTCCGATGAGACACAGTAGGCAGTACTTGGGAGGGTC     XP_936980.1
## 3  AAGGACTCAGATGCAGGGTCTTCTCTGCTCCCCGTCACACAGAGGGTGGC     XP_936046.1
## 47 GACCAAAGCAGGACAATTGCTTGATCCCAGGAGTTTAAGACCAGCCGGGG     XP_932593.1
## 46 CTTGCTGTGGTCTCTTTGTGGCAGAAGTGTTTCATGCATGGCAGCAGGCC     NP_001700.2
```

```
## 29 CCACGCCGGCAAAGAAATTGGAAGACTCCACCATTACAGGCAGCCACCAG     XP_937214.1
## 25 CAGGCGACTGGGTAGCAGATGTGGAAGCTGATGGTTAGGCCCAGGGCATG     XP_935881.1
## 54 GCACTCCGACTACATCAGGAGAAGATGTTCGAGACTTTGCCAAGGTACTA     NP_004010.1
## 9  AGCCTCTGTTCCCCTAGTAAGTGCCTTCCATGTCGGCCTCTAACCCCAGG     NP_958786.1
## 39 CCTGTTTGGATCACATGGTCTTGTCCTGATAACTTGGAAGAGGTTGCTTC     XP_371783.3
## 21 CAGATGTTTTCCCTTGTGGCAGTCTTCAGCCTCCTCTACCCTACATGATC     NP_000658.1
## 58 GCCAACCTCCTCTCACAGCCTCTGTATCTCTGCAGGCCATACTGGTTCCA     NP_009099.1
## 32 CCCAGTGACACTTCAGAGAGCTGGTAGTTAGTAGCATGTTGAGCCAGGCC     NP_005243.1
## 79 TGACTGTCCCTGCCAATGCTCCAGCTGTCGTCTGACTCTGGGTTCGTTGG     NP_006723.1
## 23 CAGCTGGGCGATGTGCGAGCTGATAGTGAGCGGCAGAATCAGGAGTACCA     NP_002267.2
##          Cytoband    Symbol gene_count foldChange_Median
## 52       15q25.1b  KIAA1199          1        21.9688396
## 68         15q14a      ACTC          1        14.6883331
## 12      22q11.23a     MMP11          1        11.8480229
## 81       14q32.2b      DLK1          2         4.7325319
## 59        8q12.1b      PENK          1         3.3078453
## 11        3p22.1b    CTNNB1          7         1.5744314
## 44        3p22.1b    CTNNB1          7         1.4655024
## 43 Xp21.2a-p21.1d       DMD          7         1.2286298
## 13        3p22.1b    CTNNB1          7         1.1602376
## 27 Xp21.2a-p21.1d       DMD          7         1.1029701
## 60        3p22.1b    CTNNB1          7         1.0850642
## 74       16p12.2a  LOC23117          8         1.0667330
## 10      12q24.33d  KIAA0692          9         1.0473309
## 19        3p22.1b    CTNNB1          7         1.0471422
## 33        8q24.3g     PLEC1          8         1.0462839
## 80             NA LOC653086          7         1.0414456
## 31 Xp21.2a-p21.1d       DMD          7         1.0377049
## 64       16p12.2a  LOC23117          8         1.0286458
## 69             NA LOC653086          7         1.0279570
## 28      12q24.33d  KIAA0692          9         1.0247769
## 76 Xp21.2a-p21.1d       DMD          7         1.0233573
## 16             NA LOC653086          7         1.0191080
## 37      12p13.31a     DDX12         10         1.0179040
## 86      16p13.11b LOC339047          7         1.0175470
## 49      12q24.33d  KIAA0692          9         1.0151057
## 83       16p12.2a  LOC23117          8         1.0146392
## 35        8q24.3g     PLEC1          8         1.0138161
## 57        5q35.2d LOC202134          7         1.0137122
## 14      16p13.11b LOC339047          7         1.0120598
## 50      12q24.33d  KIAA0692          9         1.0118272
## 70             NA LOC653086          7         1.0087172
## 38        8q24.3g     PLEC1          8         1.0081202
## 65       16p12.2a  LOC23117          8         1.0071909
## 78      16p13.11b LOC339047          7         1.0071241
## 85        5q35.2d LOC202134          7         1.0066428
## 7       16p13.11b LOC339047          7         1.0047835
## 5        11p14.1d      BDNF          7         1.0046246
## 18        5q35.2d LOC202134          7         1.0037413
## 62       11p14.1d      BDNF          7         1.0016629
```

```
## 66       16p12.2a  LOC23117          8         1.0009319
## 41      12p13.31a     DDX12         10         0.9972581
## 45       16p12.2a  LOC23117          8         0.9970746
## 51      12q24.33d  KIAA0692          9         0.9962676
## 40      16p13.11b LOC339047          7         0.9950549
## 84      12p13.31a     DDX12         10         0.9945794
## 17       11p14.1d      BDNF          7         0.9937282
## 20      12p13.31a     DDX12         10         0.9930535
## 24        3p22.1b    CTNNB1          7         0.9924948
## 36       16p12.2a  LOC23117          8         0.9922770
## 15        5q35.2d LOC202134          7         0.9917228
## 26      12p13.31a     DDX12         10         0.9915254
## 2        16p12.2a  LOC23117          8         0.9908288
## 75       11p14.1d      BDNF          7         0.9900315
## 56      12p13.31a     DDX12         10         0.9899665
## 55 Xp21.2a-p21.1d       DMD          7         0.9894561
## 8               NA LOC653086          7         0.9847075
## 42        8q24.3g     PLEC1          8         0.9844314
## 53        3p22.1b    CTNNB1          7         0.9818097
## 61        8q24.3g     PLEC1          8         0.9800936
## 72      12q24.33d  KIAA0692          9         0.9784906
## 71      12q24.33d  KIAA0692          9         0.9774402
## 22        8q24.3g     PLEC1          8         0.9773820
## 73              NA LOC653086          7         0.9770270
## 77      12p13.31a     DDX12         10         0.9757264
## 63        8q24.3g     PLEC1          8         0.9754829
## 67 Xp21.2a-p21.1d       DMD          7         0.9733874
## 82       11p14.1d      BDNF          7         0.9728164
## 4       12q24.33d  KIAA0692          9         0.9714923
## 48      16p13.11b LOC339047          7         0.9696381
## 87      16p13.11b LOC339047          7         0.9692033
## 6       12p13.31a     DDX12         10         0.9685507
## 34       11p14.1d      BDNF          7         0.9681280
## 30      12p13.31a     DDX12         10         0.9657570
## 1       12p13.31a     DDX12         10         0.9617268
## 3               NA LOC653086          7         0.9575042
## 47        5q35.2d LOC202134          7         0.9568036
## 46       11p14.1d      BDNF          7         0.9289947
## 29        5q35.2d LOC202134          7         0.9285395
## 25      12q24.33d  KIAA0692          9         0.8823950
## 54 Xp21.2a-p21.1d       DMD          7         0.7956211
## 9         8q24.3g     PLEC1          8         0.7882088
## 39        5q35.2d LOC202134          7         0.3998123
## 21          4q23b     ADH1A          1         0.1841845
## 58       17q24.2c     ABCA8          1         0.1539030
## 32       14q24.3b       FOS          1         0.1197627
## 79      19q13.32a      FOSB          1         0.1190193
## 23       17q21.2b     KRT19          1         0.1159735
```

The above table gives the protein products, the ontology function, the fold change of the median values of UL/nonUL, gene symbol, sequence of CNV, and cytoband location. The protein products can be found at genecards.org by entering the ID for the protein product into the search bar. A quick scan of a few of the protein products in genecards.org gave the following descriptions. The first listed protein NP_061159.1 says it is a colon cancer secreted protein. Many of the above CNVs are listed as proteins involved in the extracellular matrix like DMD and CTNNB1. There are also various neurological and synapses diseases associated with those proteins.

The site genecards.org has very useful properties in analyzing gene expression data from this research. If you are a member, you can download the network genes involved in diseases you query and compare to how the genes in certain tissues compare to those genes. Three out of the seven CNVs for CTNNB1 are in the top fold change median values in the ratio of UL/nonUL samples.

```
write.csv(KG4, 'keyGenes_topMedFCs.csv', row.names=FALSE)
```

Lets make a machine learning data set to test various algorithms on predicting if the sample is a UL or not. We will use the samples in this set, plus add in some microarray samples that have been studied by me elsewhere using this set of genes and sequences if available in any of the microarray studies.

Lets isolate those genes that are in our key genes of top picks for UL targets and combine the UL and nonUL sample information to those genes and sequences without the stats.

```
keyTargets <- KG4[,c(1,4)]

ULs <- UL[,c(2,10:29)]
colnames(ULs)[2:21] <- paste('UL', colnames(ULs)[2:21], sep='_')

nonULs <- nonUL[,c(2,10:27)]
colnames(nonULs)[2:19] <- paste('nonUL', colnames(nonULs)[2:19], sep='_')

keyULs <- merge(keyTargets, ULs, by.x='SEQUENCE', by.y='SEQUENCE')
keys <- merge(keyULs, nonULs, by.x='SEQUENCE', by.y='SEQUENCE')

write.csv(keys,'keyGeneTargetsCNVs.csv',row.names=FALSE)
```

Lets create the matrix for machine learning.

```
keysNames <- paste(keys$Symbol,keys$SEQUENCE, sep='_')
keys0 <- keys[,-(1:2)]
keys_ml <- as.data.frame(t(keys0))
colnames(keys_ml) <- keysNames
keys_ml$Type <-
as.factor(c(rep('UL',length(grep('^UL_',row.names(keys_ml)))),
                    rep('nonUL',length(grep('^nonUL_',row.names(keys_ml))))))
keys_ml0 <- keys_ml[,c(88,1:87)]
```

```
write.csv(keys_ml0, 'ml_ready_UL_classes.csv',row.names=TRUE)
```

Now, lets pull in the other data sets that are from the microarray samples and see if we can get the genes and sequences that correspond to our key genes above in identifying a sample as UL or not with predictive analytics.

There is one study of the other studies that has Sequence, Gene symbol and a few UL and nonUL microarray samples to compare to this above beadchip UL and nonUL set. The GEO series ID is GSE68295 with the GEO platform of GPL6480. The files are 27 MB each in file size.

```
setwd('./microArray UL')

non <- read.csv('nonUL_GSE68295_GPL6480_table.csv', sep=',',
                header=T, na.strings=c('',' '))
uls <- read.csv('UL_GSE68295_GPL6480_table.csv', sep=',',
                header=T, na.strings=c('',' '))
setwd('../')
```

Keep only the needed columns.

```
uls_array <- uls[,c(8,18:21)]
colnames(uls_array)[3:5] <- paste('UL', colnames(uls_array)[3:5], sep='_')

non_array <- non[,c(8,18:21)]
colnames(non_array)[3:5] <- paste('nonUL', colnames(non_array)[3:5], sep='_')
```

The sequences don't align or match any in the microarrays with the beadchip UL samples.

```
uls_array0 <- merge(keyTargets, uls_array, by.x='SEQUENCE', by.y='SEQUENCE')
ulsArray0 <- uls_array0[,-2]
```

Match by gene symbol between the microarray and beadchip UL samples.

```
uls_array1 <- merge(keyTargets, uls_array, by.x='Symbol', by.y='GENE_SYMBOL')
ulsArray <- uls_array1[,-2]
```

The sequences don't align between the arrays and beadchip samples for nonULs.

```
non_array0 <- merge(keyTargets, non_array, by.x='SEQUENCE', by.y='SEQUENCE')
nonArray0 <- non_array0[,-(1:2)]
```

Match by Gene symbol between the microarray and beadchip samples of nonULs.

```
non_array1 <- merge(keyTargets, non_array, by.x='Symbol', by.y='GENE_SYMBOL')
nonArray <- non_array1[,-(1:2)]
```

Combine the UL and nonUL samples of the microarrays into one dataset.

```
microarrays <- merge(ulsArray, nonArray, by.x='SEQUENCE.y',
by.y='SEQUENCE.y')
Marrays <- microarrays[!duplicated(microarrays$SEQUENCE),]
```

Since these two expression types can't be compared by sequence, they should be compared by gene. Lets combine them into a study by gene expression values.

```
keys1 <- keys[,-1]
keys2 <- keys1 %>% group_by(Symbol) %>%
  summarise_at(vars(as.vector(colnames(keys1)[2:39])), mean)

Marrays1 <- Marrays[,-1]
Marrays2 <- Marrays1 %>% group_by(Symbol) %>%
  summarise_at(vars(as.vector(colnames(Marrays1)[2:7])), mean)

beadArrays <- merge(keys2, Marrays2, by.x='Symbol', by.y='Symbol')
```

There are only 12 genes in common among these combined samples of microarray and beadchip UL and nonUL samples.

```
names <- (beadArrays$Symbol)
beadArrays1 <- beadArrays[,-1]
beadArrays_ML <- as.data.frame(t(beadArrays1))
colnames(beadArrays_ML) <- names

beadArrays_ML$Type <- as.factor(c(rep('UL_bead',20), rep('nonUL_bead',18),
                          rep('UL_array',3), rep('nonUL_array',3)))
beadArrays_ML2 <- beadArrays_ML[,c(13,1:12)]

UL_seq_ML <- keys_ml0
UL_gene_ML <- beadArrays_ML2
```

There are two datasets to use for machine learning. The first is our beadchip samples of 88 sequences and 38 samples of 20 UL and 18 nonUL in the **UL_seq_ML** data set. The second dataset for machine learning is the mixed microarray and beadchip samples of UL and nonUL by gene in the **UL_gene_ML** data set, because there were no common sequence or copy number variants of the gene sequences between the beadchip and microarray sets of UL and nonUL samples.

The libraries were installed earlier.

```
set.seed(12356789)
```

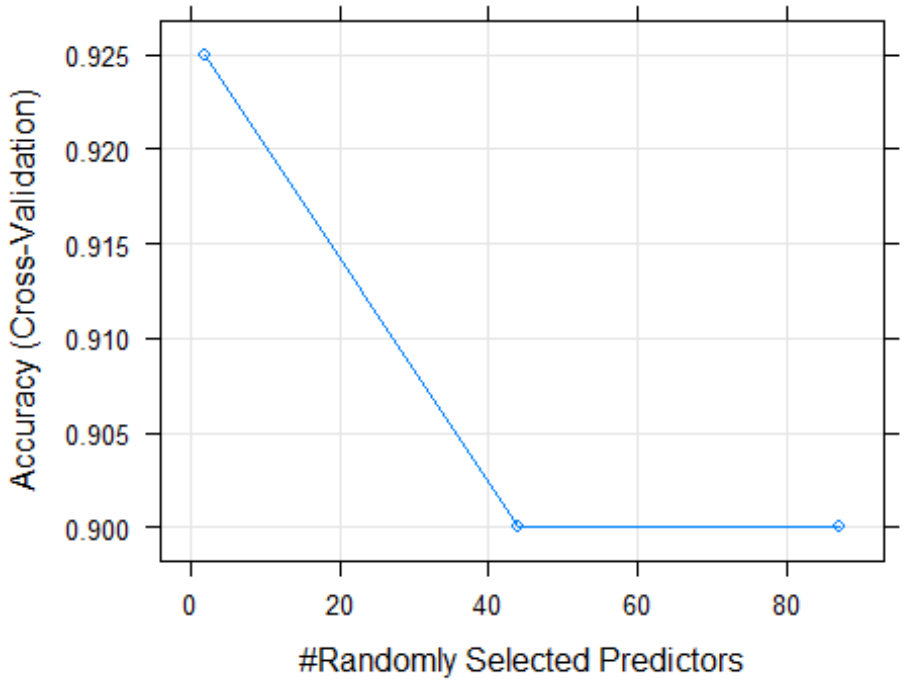
Create a partition of the data with a 70/30 split into training/testing sets of the first data set with two classes of UL or nonUL and 88 features of genes with their CNVs.

```
inTrain <- createDataPartition(y=UL_seq_ML$Type, p=0.7, list=FALSE)
```

```
trainingSet <- UL_seq_ML[inTrain,]
testingSet <- UL_seq_ML[-inTrain,]
```

RandomForest, cross-validation (cv) = 5

```
rfMod <- train(Type~., method='rf', data=(trainingSet),
               trControl=trainControl(method='cv'), number=5)
plot(rfMod)
```



Run predictions on the testing set

```
predRF <- predict(rfMod, testingSet)

predDF <- data.frame(predRF, type=testingSet$Type)
predDF

##    predRF  type
## 1      UL    UL
## 2      UL    UL
## 3      UL    UL
## 4      UL    UL
## 5      UL    UL
## 6      UL    UL
## 7   nonUL nonUL
## 8   nonUL nonUL
## 9   nonUL nonUL
## 10  nonUL nonUL
## 11  nonUL nonUL
```

```
sum <- sum(predRF==testingSet$Type)
length <- length(testingSet$Type)
accuracy_rfMod <- (sum/length)
accuracy_rfMod

## [1] 1

results <- c(round(accuracy_rfMod,2), round(100,2))
results <- as.factor(results)
results <- t(data.frame(results))

colnames(results) <- colnames(predDF)
Results <- rbind(predDF, results)
Results

##         predRF  type
## 1           UL    UL
## 2           UL    UL
## 3           UL    UL
## 4           UL    UL
## 5           UL    UL
## 6           UL    UL
## 7        nonUL nonUL
## 8        nonUL nonUL
## 9        nonUL nonUL
## 10       nonUL nonUL
## 11       nonUL nonUL
## results      1   100
```

The above shows that using the genes and the CNV of each gene totalling 88 features, makes a perfect data set of results with only 27 observations to train and 11 to test on 2 classes of UL or non-UL. Using only the random forest algorithm it classified each sample 100% accurately trained on 70% of the samples.

What if we used random forest to only predict by gene in the first beadchip type data set? We can use the transpose of the keys2 data set made earlier when combining to make the 2nd ML dataset.

```
names <- keys2$Symbol
keys_2 <- keys2[,-1]
keys_t <- as.data.frame(t(keys_2))
colnames(keys_t) <- names
keys_t$Type <- keys_ml$Type
keys_ML <- keys_t[,c(21,1:20)]
```

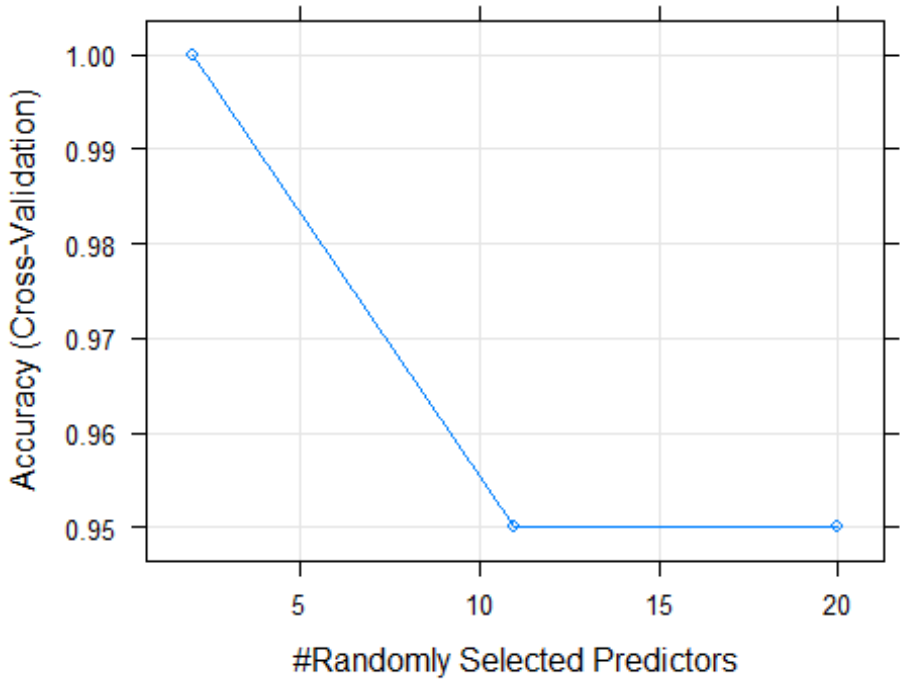
Now we will use our new data set based on the beadchip genes and not the CNVs of those genes, in the keys_ML data set to predict with RandomForest.

```
inTrain <- createDataPartition(y=keys_ML$Type, p=0.7, list=FALSE)
```

```
trainingSet <- keys_ML[inTrain,]
testingSet <- keys_ML[-inTrain,]
```

RandomForest, cross-validation (cv) = 5

```
rfMod <- train(Type~., method='rf', data=(trainingSet),
               trControl=trainControl(method='cv'), number=5)
plot(rfMod)
```



Run predictions on the testing set

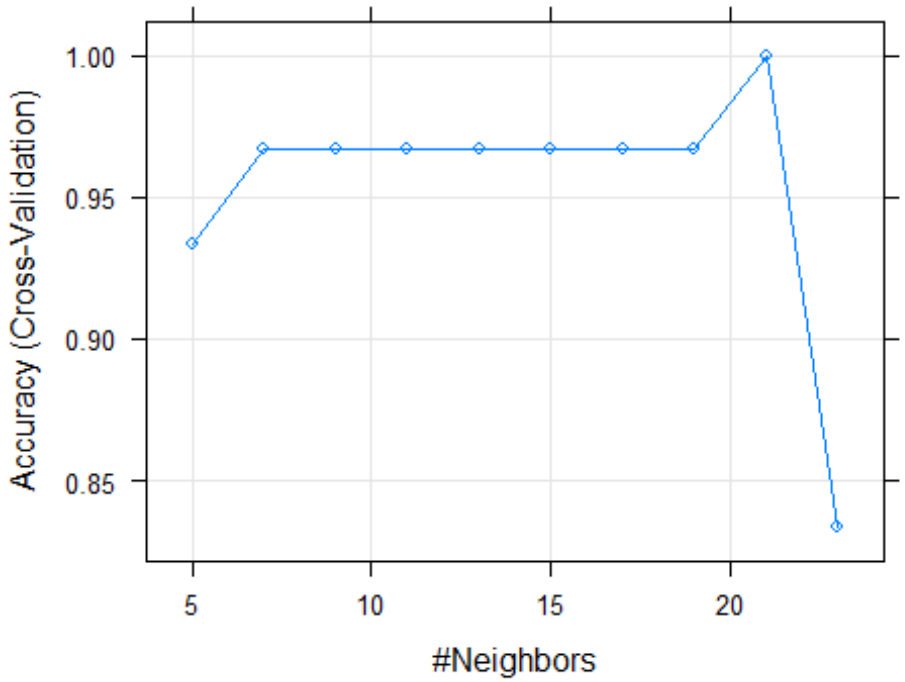
```
predRF <- predict(rfMod, testingSet)

predDF <- data.frame(predRF, type=testingSet$Type)
predDF

##    predRF  type
## 1      UL    UL
## 2      UL    UL
## 3      UL    UL
## 4      UL    UL
## 5      UL    UL
## 6   nonUL    UL
## 7   nonUL nonUL
## 8   nonUL nonUL
## 9   nonUL nonUL
## 10  nonUL nonUL
## 11  nonUL nonUL
```

```
sum <- sum(predRF==testingSet$Type)
length <- length(testingSet$Type)
accuracy_rfMod <- (sum/length)
accuracy_rfMod

## [1] 0.9090909

results <- c(round(accuracy_rfMod,2), round(100,2))
results <- as.factor(results)
results <- t(data.frame(results))

colnames(results) <- colnames(predDF)
Results <- rbind(predDF, results)
Results

##         predRF  type
## 1           UL    UL
## 2           UL    UL
## 3           UL    UL
## 4           UL    UL
## 5           UL    UL
## 6        nonUL    UL
## 7        nonUL nonUL
## 8        nonUL nonUL
## 9        nonUL nonUL
## 10       nonUL nonUL
## 11       nonUL nonUL
## results   0.91   100
```

From the above table, the random forest algorithm only misclassified one sample as nonUL, when it was really a UL sample. This data set used the mean values of the genes and not the copy number variants of each gene as the previous data set we just used was built on. So, a true positive was misclassified as a negative. This means its a false negative or Type II error. If it had misclassified a nonUL as UL then it would be a Type I error for false positive. There is a good article to review material you never really use, until time to write a theoretical research paper. The values are good to know for precision and recall, and sensitivity and specificity. Depending on what your overarching goal is in sampling outcomes, you want to improve one more than the other.

How about with the KNN algorithm.

```
knnMod <- train(Type ~ .,
                method='knn', preProcess=c('center','scale'),
                tuneLength=10, trControl=trainControl(method='cv'),
data=trainingSet)
plot(knnMod)
```




```
rpartMod <- train(Type ~ ., method='rpart', tuneLength=7, data=trainingSet)

glmMod <- train(Type ~ .,
                method='glm', data=trainingSet)

predKNN <- predict(knnMod, testingSet)
predRPART <- predict(rpartMod, testingSet)
predGLM <- predict(glmMod, testingSet)

length=length(testingSet$Type)

sumKNN <- sum(predKNN==testingSet$Type)
sumRPart <- sum(predRPART==testingSet$Type)
sumGLM <- sum(predGLM==testingSet$Type)

accuracy_KNN <- sumKNN/length
accuracy_RPART <- sumRPart/length
accuracy_GLM <- sumGLM/length

predDF2 <- data.frame(predRF,predKNN,predRPART,predGLM,
                      TYPE=testingSet$Type)
colnames(predDF2) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')

results <- c(round(accuracy_rfMod,2),
             round(accuracy_KNN,2),
             round(accuracy_RPART,2),
             round(accuracy_GLM,2),
```

```
            round(100,2))

results <- as.factor(results)
results <- t(data.frame(results))
colnames(results) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')
Results <- rbind(predDF2, results)
Results
```

```
##         RandomForest   KNN Rpart   GLM TrueValue
## 1                 UL    UL nonUL    UL        UL
## 2                 UL    UL    UL    UL        UL
## 3                 UL    UL nonUL    UL        UL
## 4                 UL    UL    UL    UL        UL
## 5                 UL    UL    UL    UL        UL
## 6              nonUL nonUL nonUL nonUL        UL
## 7              nonUL nonUL nonUL nonUL     nonUL
## 8              nonUL nonUL nonUL nonUL     nonUL
## 9              nonUL nonUL nonUL nonUL     nonUL
## 10             nonUL nonUL nonUL nonUL     nonUL
## 11             nonUL nonUL    UL nonUL     nonUL
## results         0.91  0.91  0.64  0.91       100
```

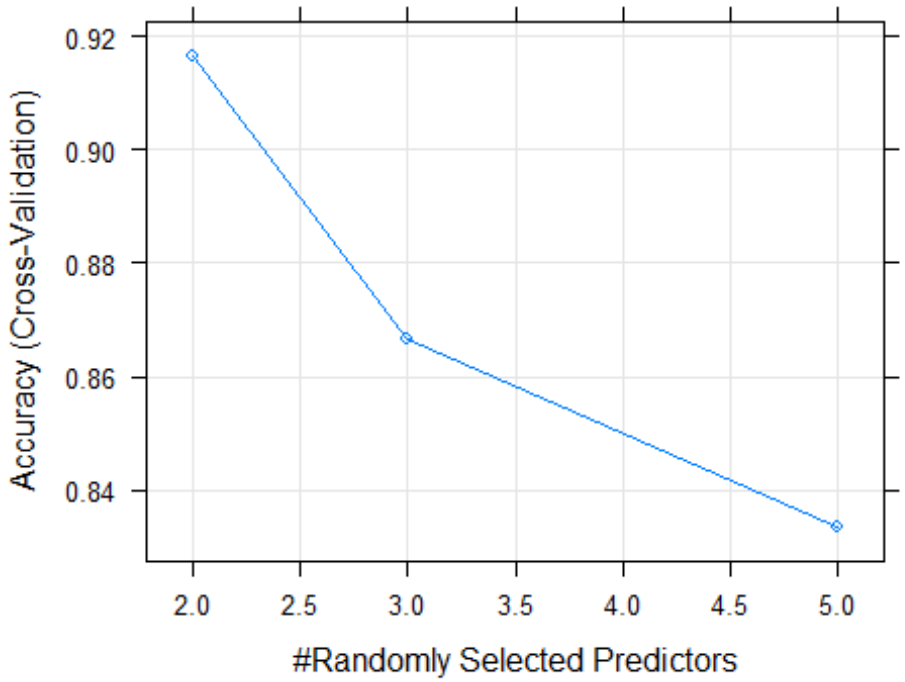
As far as the algorithms used above go, the prediction accuracy is great for Random Forest, GLM, and K-Nearest Neighbor with 91% accuracy. Make sure to remove any fields before transposing such as the symbol field when keeping the samples as numeric. Because the numeric sample values will be factors, and throw off the algorithms. Or you could manually change each of the class types of the above genes above. Rpart, or recursive partitioning trees did the worst with 64% accuracy.

This next data set uses the three most expressed and two lease expressed genes by fold change in UL to nonUL sample means.

```
keys_ML_b <- keys_ML[,c(1,3,10,11,13,19)]

inTrain <- createDataPartition(y=keys_ML_b$Type, p=0.7, list=FALSE)

trainingSet <- keys_ML_b[inTrain,]
testingSet <- keys_ML_b[-inTrain,]
```

RandomForest, cross-validation (cv) = 5

```
rfMod <- train(Type~., method='rf', data=(trainingSet),
               trControl=trainControl(method='cv'), number=5)
plot(rfMod)
```



Run predictions on the testing set

```
predRF <- predict(rfMod, testingSet)

predDF <- data.frame(predRF, type=testingSet$Type)
predDF
```

```
##    predRF  type
## 1      UL    UL
## 2      UL    UL
## 3      UL    UL
## 4      UL    UL
## 5      UL    UL
## 6      UL    UL
## 7   nonUL nonUL
## 8   nonUL nonUL
## 9   nonUL nonUL
## 10  nonUL nonUL
## 11  nonUL nonUL
```

```
sum <- sum(predRF==testingSet$Type)
length <- length(testingSet$Type)
accuracy_rfMod <- (sum/length)
accuracy_rfMod
```

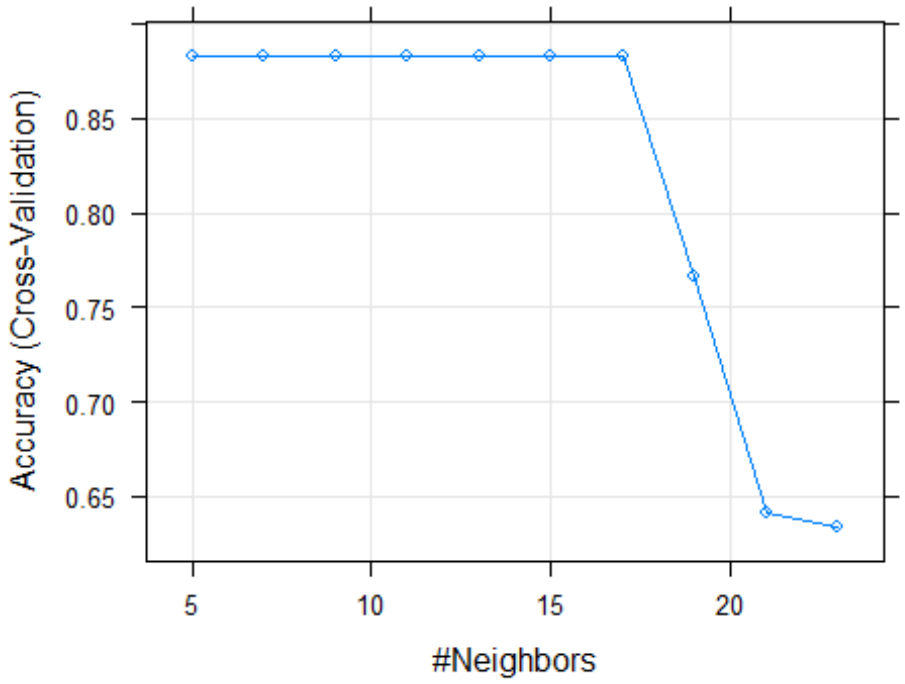
```
## [1] 1
```

```
results <- c(round(accuracy_rfMod,2), round(100,2))
results <- as.factor(results)
results <- t(data.frame(results))

colnames(results) <- colnames(predDF)
Results <- rbind(predDF, results)
Results
```

```
##         predRF  type
## 1           UL    UL
## 2           UL    UL
## 3           UL    UL
## 4           UL    UL
## 5           UL    UL
## 6           UL    UL
## 7        nonUL nonUL
## 8        nonUL nonUL
## 9        nonUL nonUL
## 10       nonUL nonUL
## 11       nonUL nonUL
## results      1   100
```

How about with the KNN algorithm.

```
knnMod <- train(Type ~ .,
                method='knn', preProcess=c('center','scale'),
                tuneLength=10, trControl=trainControl(method='cv'),
data=trainingSet)
plot(knnMod)
```



The accuracy seems to be better between 5 and 17 neighbors for classification from what the above plot is displaying.

```
rpartMod <- train(Type ~ ., method='rpart', tuneLength=7, data=trainingSet)

glmMod <- train(Type ~ .,
                method='glm', data=trainingSet)

predKNN <- predict(knnMod, testingSet)
predRPART <- predict(rpartMod, testingSet)
predGLM <- predict(glmMod, testingSet)

length=length(testingSet$Type)

sumKNN <- sum(predKNN==testingSet$Type)
sumRPart <- sum(predRPART==testingSet$Type)
sumGLM <- sum(predGLM==testingSet$Type)

accuracy_KNN <- sumKNN/length
accuracy_RPART <- sumRPart/length
accuracy_GLM <- sumGLM/length

predDF2 <- data.frame(predRF,predKNN,predRPART,predGLM,
                      TYPE=testingSet$Type)
colnames(predDF2) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')

results <- c(round(accuracy_rfMod,2),
```

```
              round(accuracy_KNN,2),
              round(accuracy_RPART,2),
              round(accuracy_GLM,2),
              round(100,2))

results <- as.factor(results)
results <- t(data.frame(results))
colnames(results) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')
Results <- rbind(predDF2, results)
Results
```

```
##         RandomForest   KNN Rpart   GLM TrueValue
## 1                 UL    UL    UL    UL        UL
## 2                 UL    UL    UL    UL        UL
## 3                 UL    UL    UL    UL        UL
## 4                 UL    UL    UL    UL        UL
## 5                 UL    UL    UL    UL        UL
## 6                 UL nonUL nonUL    UL        UL
## 7              nonUL nonUL nonUL nonUL     nonUL
## 8              nonUL nonUL nonUL nonUL     nonUL
## 9              nonUL nonUL nonUL nonUL     nonUL
## 10             nonUL nonUL nonUL nonUL     nonUL
## 11             nonUL nonUL nonUL nonUL     nonUL
## results            1  0.91  0.91     1       100
```

The above data shows that using the three highest fold change and lowest fold change genes to predict the sample as a UL or not scored 100% accuracy for GLM and Random Forest. And the KNN and Rpart scored 91% accuracy. Make sure to set your seed to the same value so you don't get different results when re-running the algorithms above, because the first seed was set by me and I got different results where KNN scored 100% and GLM and Random Forest scored 82% accuracy. ***

Lets go back to the data that placed the fold change values by mean of the sequence values.

```
fib <- fibroid[,c(1,10:29)]
fib_mean <- fib %>% group_by(Symbol) %>%
  summarise_at(vars(as.vector(colnames(fib)[2:21])), mean, na.rm=TRUE)
fib_mean$UL_gene_mean <- rowMeans(fib_mean[2:21])
colnames(fib_mean)[2:21] <- paste('UL', colnames(fib_mean)[2:21], sep='_')

nfib <- nonFibroid[,c(1,10:27)]
nfib_mean <- nfib %>% group_by(Symbol) %>%
  summarise_at(vars(as.vector(colnames(nfib)[2:19])), mean, na.rm=TRUE)
nfib_mean$nonUL_gene_mean <- rowMeans(nfib_mean[2:19])
colnames(nfib_mean)[2:19] <- paste('nonUL', colnames(nfib_mean)[2:19],
sep='_')

fib_nonfib <- merge(fib_mean,nfib_mean, by.x='Symbol', by.y='Symbol')
fib_nonfib$FC_UL2non <- fib_nonfib$UL_gene_mean/fib_nonfib$nonUL_gene_mean
Fib_Non <- fib_nonfib[,c(1,22,41,42,2:21,23:40)]
```

```
FC_genes <- Fib_Non[order(Fib_Non$FC_UL2non, decreasing=TRUE)[1:5],]
FCs <- FC_genes[,c(1,5:42)]
FCs_t <- as.data.frame(t(FCs))
colnames(FCs_t) <- FCs$Symbol
FCs_ML <- FCs_t[-1,]
FCs_ML$Type <- as.factor(c(rep('UL',20),rep('nonUL',18)))
FCs_ML1 <- FCs_ML[,c(6,1:5)]
head(FCs_ML1)
```

```
##                Type  KIAA1199      PENK      ACTC     MMP11      DLK1
## UL_GSM2496185    UL     861.2     442.0    1452.7   13478.5     151.6
## UL_GSM2496186    UL 3915.9192  219.7620 4341.5140 8300.2949  221.1859
## UL_GSM2496187    UL 2292.2627  300.5855 1460.5967 6096.7978  750.0491
## UL_GSM2496188    UL 4272.8288  699.5680 8327.8549 8858.7194  297.5915
## UL_GSM2496189    UL 2212.2382   99.1353 5434.4145 3590.9747  105.1468
## UL_GSM2496190    UL 6548.8160  255.7960 6894.4949 4141.8642  246.1572
```

Now lets try these algorithms again, to see if they provide better results on the top five genes with the highest fold change values in each gene.

```
FCs_ML1$KIAA1199 <- as.numeric(FCs_ML1$KIAA1199)
FCs_ML1$PENK <- as.numeric(FCs_ML1$PENK)
FCs_ML1$ACTC <- as.numeric(FCs_ML1$ACTC)
FCs_ML1$MMP11 <- as.numeric(FCs_ML1$MMP11)
FCs_ML1$DLK1 <- as.numeric(FCs_ML1$DLK1)

set.seed(123789)
inTrain <- createDataPartition(y=FCs_ML1$Type, p=0.7, list=FALSE)

trainingSet <- FCs_ML1[inTrain,]
testingSet <- FCs_ML1[-inTrain,]
```

RandomForest, cross-validation (cv) = 5

```
rfMod <- train(Type~., method='rf', data=(trainingSet),
               trControl=trainControl(method='cv'), number=5)
```

Run predictions on the testing set

```
predRF <- predict(rfMod, testingSet)

predDF <- data.frame(predRF, type=testingSet$Type)
predDF
```

```
##    predRF  type
## 1      UL    UL
## 2      UL    UL
## 3      UL    UL
## 4      UL    UL
## 5      UL    UL
## 6      UL    UL
## 7   nonUL nonUL
```

```
## 8   nonUL nonUL
## 9   nonUL nonUL
## 10  nonUL nonUL
## 11  nonUL nonUL
```

```
sum <- sum(predRF==testingSet$Type)
length <- length(testingSet$Type)
accuracy_rfMod <- (sum/length)
accuracy_rfMod
```

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

The above table shows that using the set of five genes that had the highest fold change values scored 100% accuracy using the random forest algorithm to predict a sample as being a UL or not.

```
results <- c(round(accuracy_rfMod,2), round(100,2))
results <- as.factor(results)
results <- t(data.frame(results))

colnames(results) <- colnames(predDF)
Results <- rbind(predDF, results)
Results
```

```
##         predRF  type
## 1           UL    UL
## 2           UL    UL
## 3           UL    UL
## 4           UL    UL
## 5           UL    UL
## 6           UL    UL
## 7        nonUL nonUL
## 8        nonUL nonUL
## 9        nonUL nonUL
## 10       nonUL nonUL
## 11       nonUL nonUL
## results      1   100
```

How about with the KNN algorithm.

```
knnMod <- train(Type ~ .,
                method='knn', preProcess=c('center','scale'),
                tuneLength=10, trControl=trainControl(method='cv'),
data=trainingSet)

rpartMod <- train(Type ~ ., method='rpart', tuneLength=7, data=trainingSet)

glmMod <- train(Type ~ .,
                method='glm', data=trainingSet)
```

```
predKNN <- predict(knnMod, testingSet)
predRPART <- predict(rpartMod, testingSet)
predGLM <- predict(glmMod, testingSet)

length=length(testingSet$Type)

sumKNN <- sum(predKNN==testingSet$Type)
sumRPart <- sum(predRPART==testingSet$Type)
sumGLM <- sum(predGLM==testingSet$Type)

accuracy_KNN <- sumKNN/length
accuracy_RPART <- sumRPart/length
accuracy_GLM <- sumGLM/length

predDF2 <- data.frame(predRF,predKNN,predRPART,predGLM,
                      TYPE=testingSet$Type)
colnames(predDF2) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')

results <- c(round(accuracy_rfMod,2),
             round(accuracy_KNN,2),
             round(accuracy_RPART,2),
             round(accuracy_GLM,2),
             round(100,2))

results <- as.factor(results)
results <- t(data.frame(results))
colnames(results) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')
Results <- rbind(predDF2, results)
Results
```

```
##         RandomForest   KNN Rpart   GLM TrueValue
## 1                 UL nonUL    UL    UL        UL
## 2                 UL    UL nonUL nonUL        UL
## 3                 UL    UL    UL    UL        UL
## 4                 UL    UL nonUL    UL        UL
## 5                 UL    UL    UL    UL        UL
## 6                 UL    UL    UL    UL        UL
## 7              nonUL nonUL nonUL nonUL     nonUL
## 8              nonUL nonUL nonUL nonUL     nonUL
## 9              nonUL    UL nonUL nonUL     nonUL
## 10             nonUL nonUL nonUL nonUL     nonUL
## 11             nonUL nonUL nonUL    UL     nonUL
## results            1  0.82  0.82  0.82       100
```

The random forest classifier scored 100% accuracy, while KNN, Rpart, and GLM algorithms all scored 82% accuracy in predicting a sample as uL or not.

Lets use the data set with four classes to predict in the mixed beadchip and microarray samples as either UL or nonUL in their respective medium.

```
set.seed(123789)
inTrain <- createDataPartition(y=UL_gene_ML$Type, p=0.7, list=FALSE)

trainingSet <- UL_gene_ML[inTrain,]
testingSet <- UL_gene_ML[-inTrain,]
```

RandomForest, cross-validation (cv) = 5

```
rfMod <- train(Type~., method='rf', data=(trainingSet),
               trControl=trainControl(method='cv'), number=5)
```

Run predictions on the testing set

```
predRF <- predict(rfMod, testingSet)

predDF <- data.frame(predRF, type=testingSet$Type)
predDF
```

```
##         predRF       type
## 1     UL_bead    UL_bead
## 2     UL_bead    UL_bead
## 3     UL_bead    UL_bead
## 4     UL_bead    UL_bead
## 5     UL_bead    UL_bead
## 6     UL_bead    UL_bead
## 7  nonUL_bead nonUL_bead
## 8  nonUL_bead nonUL_bead
## 9  nonUL_bead nonUL_bead
## 10 nonUL_bead nonUL_bead
## 11 nonUL_bead nonUL_bead
```

```
sum <- sum(predRF==testingSet$Type)
length <- length(testingSet$Type)
accuracy_rfMod <- (sum/length)
accuracy_rfMod
```

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

```
results <- c(round(accuracy_rfMod,2), round(100,2))
results <- as.factor(results)
results <- t(data.frame(results))

colnames(results) <- colnames(predDF)
Results <- rbind(predDF, results)
Results
```

```
##              predRF       type
## 1          UL_bead    UL_bead
## 2          UL_bead    UL_bead
```

```
## 3          UL_bead    UL_bead
## 4          UL_bead    UL_bead
## 5          UL_bead    UL_bead
## 6          UL_bead    UL_bead
## 7       nonUL_bead nonUL_bead
## 8       nonUL_bead nonUL_bead
## 9       nonUL_bead nonUL_bead
## 10      nonUL_bead nonUL_bead
## 11      nonUL_bead nonUL_bead
## results          1        100
```

The above table shows that the random forest algorithm scored 100% accuracy on the four class predictions of our testing set.

How about with the KNN algorithm.

```
knnMod <- train(Type ~ .,
                method='knn', preProcess=c('center','scale'),
                tuneLength=10, trControl=trainControl(method='cv'),
data=trainingSet)

rpartMod <- train(Type ~ ., method='rpart', tuneLength=7, data=trainingSet)

#glmMod <- train(Type ~ .,
 #                method='glm', data=trainingSet)
glmMod <- glm(Type ~ .,family = binomial(), data=trainingSet,
method='glm.fit',)
GLMpred <- predict.glm(glmMod, type = "response")
```

The above GLM model is not liking this type of data, predicting the class by the numeric probabilities of the features provided. It seemed to do fine with the other data sets that had two classes and also used numeric data to predict each class factor.

```
predKNN <- predict(knnMod, testingSet)
predRPART <- predict(rpartMod, testingSet)
predGLM <- GLMpred

length=length(testingSet$Type)

sumKNN <- sum(predKNN==testingSet$Type)
sumRPart <- sum(predRPART==testingSet$Type)
sumGLM <- sum(predGLM==testingSet$Type)

accuracy_KNN <- sumKNN/length
accuracy_RPART <- sumRPart/length
accuracy_GLM <- sumGLM/length

predDF2 <- data.frame(predRF,predKNN,predRPART,predGLM,
                      TYPE=testingSet$Type)
colnames(predDF2) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')
```

```
results <- c(round(accuracy_rfMod,2),
             round(accuracy_KNN,2),
             round(accuracy_RPART,2),
             round(accuracy_GLM,2),
             round(100,2))

results <- as.factor(results)
results <- t(data.frame(results))
colnames(results) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')
Results <- rbind(predDF2, results)
Results
```

```
##                   RandomForest        KNN      Rpart                    GLM
## UL_GSM2496185          UL_bead    UL_bead    UL_bead                      1
## UL_GSM2496187          UL_bead    UL_bead    UL_bead                      1
## UL_GSM2496189          UL_bead    UL_bead    UL_bead                      1
## UL_GSM2496190          UL_bead    UL_bead    UL_bead                      1
## UL_GSM2496191          UL_bead    UL_bead nonUL_bead                      1
## UL_GSM2496192          UL_bead    UL_bead    UL_bead                      1
## UL_GSM2496193       nonUL_bead nonUL_bead nonUL_bead                      1
## UL_GSM2496203       nonUL_bead nonUL_bead nonUL_bead                      1
## UL_GSM2496204       nonUL_bead nonUL_bead nonUL_bead                      1
## UL_GSM2496206       nonUL_bead nonUL_bead nonUL_bead                      1
## UL_GSM2496207       nonUL_bead nonUL_bead nonUL_bead                      1
## UL_GSM2496209          UL_bead    UL_bead    UL_bead                      1
## UL_GSM2496217          UL_bead    UL_bead    UL_bead                      1
## UL_GSM2496220          UL_bead    UL_bead    UL_bead                      1
## nonUL_GSM2496194       UL_bead    UL_bead    UL_bead                      1
## nonUL_GSM2496196       UL_bead    UL_bead nonUL_bead                      1
## nonUL_GSM2496197       UL_bead    UL_bead    UL_bead                      1
## nonUL_GSM2496198    nonUL_bead nonUL_bead nonUL_bead                      1
## nonUL_GSM2496200    nonUL_bead nonUL_bead nonUL_bead                      1
## nonUL_GSM2496201    nonUL_bead nonUL_bead nonUL_bead                      1
## nonUL_GSM2496210    nonUL_bead nonUL_bead nonUL_bead                      1
## nonUL_GSM2496212    nonUL_bead nonUL_bead nonUL_bead                      1
## nonUL_GSM2496213       UL_bead    UL_bead    UL_bead                      1
## nonUL_GSM2496215       UL_bead    UL_bead    UL_bead                      1
## nonUL_GSM2496216       UL_bead    UL_bead    UL_bead                      1
## nonUL_GSM2496221       UL_bead    UL_bead    UL_bead                      1
## nonUL_GSM2496222       UL_bead    UL_bead nonUL_bead                      1
## UL_GSM1667147          UL_bead    UL_bead    UL_bead 2.22044604925031e-16
## UL_GSM1667148       nonUL_bead nonUL_bead nonUL_bead 2.22044604925031e-16
## UL_GSM1667149       nonUL_bead nonUL_bead nonUL_bead 2.22044604925031e-16
## nonUL_GSM1667144    nonUL_bead nonUL_bead nonUL_bead 2.22044604925031e-16
## nonUL_GSM1667145    nonUL_bead nonUL_bead nonUL_bead 2.22044604925031e-16
## nonUL_GSM1667146    nonUL_bead nonUL_bead nonUL_bead 2.22044604925031e-16
## results                      1          1       0.91                      0
##                    TrueValue
## UL_GSM2496185        UL_bead
## UL_GSM2496187        UL_bead
```

```
## UL_GSM2496189       UL_bead
## UL_GSM2496190       UL_bead
## UL_GSM2496191       UL_bead
## UL_GSM2496192       UL_bead
## UL_GSM2496193    nonUL_bead
## UL_GSM2496203    nonUL_bead
## UL_GSM2496204    nonUL_bead
## UL_GSM2496206    nonUL_bead
## UL_GSM2496207    nonUL_bead
## UL_GSM2496209       UL_bead
## UL_GSM2496217       UL_bead
## UL_GSM2496220       UL_bead
## nonUL_GSM2496194    UL_bead
## nonUL_GSM2496196    UL_bead
## nonUL_GSM2496197    UL_bead
## nonUL_GSM2496198 nonUL_bead
## nonUL_GSM2496200 nonUL_bead
## nonUL_GSM2496201 nonUL_bead
## nonUL_GSM2496210 nonUL_bead
## nonUL_GSM2496212 nonUL_bead
## nonUL_GSM2496213    UL_bead
## nonUL_GSM2496215    UL_bead
## nonUL_GSM2496216    UL_bead
## nonUL_GSM2496221    UL_bead
## nonUL_GSM2496222    UL_bead
## UL_GSM1667147       UL_bead
## UL_GSM1667148    nonUL_bead
## UL_GSM1667149    nonUL_bead
## nonUL_GSM1667144 nonUL_bead
## nonUL_GSM1667145 nonUL_bead
## nonUL_GSM1667146 nonUL_bead
## results                 100
```

The GLM or generalized linear model is used to regress actual predicted numeric values using linear regression and naive bayes type linear models. This could be why its values were numeric the first run and now unable to complete training so it was excluded from results as there were no results for the GLM model. The Random Forest and KNN scored 100% accuracy, and the Rpart scored 91% accuracy.

Lets add in the gene that had the lowest fold change value and use it as an outcome variable to predict the value based on these high fold change values. We will remove the Type field.

```
FC_genes1 <- Fib_Non[order(Fib_Non$FC_UL2non)[c(1:2,25032:25036)],]
row.names(FC_genes1) <- FC_genes1$Symbol
FC_genes2 <- FC_genes1[-2,]
FC_genes2_ML <- as.data.frame(t(FC_genes2))
FCs_ML_4 <- FC_genes2_ML[-c(1:4),]
write.csv(FCs_ML_4,'ML_highFCs_lowFC.csv', row.names=TRUE)
```

```
set.seed(123789)
ML4 <- FCs_ML_4
ML4$KRT19 <- as.numeric(ML4$KRT19)
ML4$DLK1 <- as.numeric(ML4$DLK1)
ML4$MMP11 <- as.numeric(ML4$MMP11)
ML4$ACTC <- as.numeric(ML4$ACTC)
ML4$PENK <- as.numeric(ML4$PENK)
ML4$KIAA1199 <- as.numeric(ML4$KIAA1199)


inTrain <- createDataPartition(y=ML4$KRT19, p=0.7, list=FALSE)

trainingSet <- ML4[inTrain,]
testingSet <- ML4[-inTrain,]
```

RandomForest, cross-validation (cv) = 5

```
rfMod <- train(KRT19~., method='rf', data=(trainingSet),
               trControl=trainControl(method='cv'), number=5)
```

Run predictions on the testing set, altered so that the predicted value is within one standard deviations of the mean.

```
predRF <- round(predict(rfMod, testingSet),0)

predDF <- data.frame(predRF, KRT19_Value=testingSet$KRT19)
predDF
```

```
##                   predRF KRT19_Value
## UL_GSM2496185         16           3
## UL_GSM2496191         21           2
## UL_GSM2496204         16          11
## UL_GSM2496208         15           9
## UL_GSM2496219         24          18
## nonUL_GSM2496199      33          34
## nonUL_GSM2496210      33          35
## nonUL_GSM2496212      22          28
## nonUL_GSM2496215      32          39
## nonUL_GSM2496222      27          23
```

```
mu <- mean(testingSet$KRT19)
sde <- sd(testingSet$KRT19)

sum <- sum(predRF < (mu+sde))

length <- length(testingSet$KRT19)
accuracy_rfMod <- (sum/length)
accuracy_rfMod
```

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

All predicted values are within one standard deviation of the mean.

```
results <- c(round(accuracy_rfMod,2), round(100,2))
results <- as.factor(results)
results <- t(data.frame(results))

colnames(results) <- colnames(predDF)
Results <- rbind(predDF, results)
Results
```

```
##                   predRF KRT19_Value
## UL_GSM2496185         16           3
## UL_GSM2496191         21           2
## UL_GSM2496204         16          11
## UL_GSM2496208         15           9
## UL_GSM2496219         24          18
## nonUL_GSM2496199      33          34
## nonUL_GSM2496210      33          35
## nonUL_GSM2496212      22          28
## nonUL_GSM2496215      32          39
## nonUL_GSM2496222      27          23
## results                1         100
```

The above didn't get any of the predicted results exactly equal to the true value of the lowest expressed gene in fold change of UL/nonUL, but it did get every predicted value within one standard deviation of the sample mean.

How about with the KNN algorithm.

```
knnMod <- train(KRT19 ~ .,
                method='knn', preProcess=c('center','scale'),
                tuneLength=10, trControl=trainControl(method='cv'),
data=trainingSet)
```

The accuracy seems to be better between 8 and 9 neighbors for classification from what the above plot is displaying.

```
rpartMod <- train(KRT19 ~ ., method='rpart', tuneLength=9, data=trainingSet)

glmMod <- train(KRT19 ~ .,
                method='glm', data=trainingSet)

predKNN <- predict(knnMod, testingSet)
predRPART <- predict(rpartMod, testingSet)
predGLM <- predict(glmMod, testingSet)

length=length(testingSet$KRT19)

sumKNN <- sum(predKNN==testingSet$KRT19)
sumRPart <- sum(predRPART==testingSet$KRT19)
sumGLM <- sum(predGLM==testingSet$KRT19)
```

```
accuracy_KNN <- sumKNN/length
accuracy_RPART <- sumRPart/length
accuracy_GLM <- sumGLM/length

predDF2 <- data.frame(predRF,predKNN,predRPART,predGLM,
                      KRT19=testingSet$KRT19)
colnames(predDF2) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')

results <- c(round(accuracy_rfMod,2),
             round(accuracy_KNN,2),
             round(accuracy_RPART,2),
             round(accuracy_GLM,2),
             round(100,2))

results <- as.factor(results)
results <- t(data.frame(results))
colnames(results) <- c('RandomForest','KNN','Rpart','GLM','TrueValue')
Results <- rbind(predDF2, results)
Results
```

```
##                  RandomForest  KNN            Rpart              GLM
TrueValue
## UL_GSM2496185              16 21.8          15.3125 17.7925113372005
3
## UL_GSM2496191              21 30.8 28.9166666666667 41.9876957302681
2
## UL_GSM2496204              16   16          15.3125 11.2416036435578
11
## UL_GSM2496208              15 18.8          15.3125 10.8507470045519
9
## UL_GSM2496219              24 27.8 28.9166666666667 34.1668248778496
18
## nonUL_GSM2496199           33 35.6 28.9166666666667 21.3231243095602
34
## nonUL_GSM2496210           33 36.2 28.9166666666667 33.9084163681288
35
## nonUL_GSM2496212           22 21.8          15.3125 21.5506884597176
28
## nonUL_GSM2496215           32   27 28.9166666666667 24.3442067799312
39
## nonUL_GSM2496222           27 30.2 28.9166666666667 23.9479962823136
23
## results                     1    0               0                0
100
```

When it comes to regression on numeric values and using the five highest expressed genes to predict the value of the lowest expressed gene in each sample of UL or nonUL, the results were far from useful. Every algorithm scored 0% but the random forest which was modified to gain accuracty if the prediction is within 1 standard deviation of the sample

mean in which case it scored 100%. But would have also scored 0% as the other algorithms have.