| Name | ruid | netid |
| --- | --- | --- |
| Kush Aswani | 187001586 | kaa189 |
| Nishant Kumar | 186003764 | nak112 |
| Tejasva Tomar | 189007957 | tt425 |
| Dev Shah | | ds1560 |

# NCI60 Project

- **Download the ISLR package into R, construct Using the NCI60 data, construct simultaneous ttests and fdr for each class of cancer (check NCI60$labs)**

```
#importing ISLR library
library(ISLR)
#Finding unique cancer types
cancer.labels=unique(NCI60$labs)
#Finding no. cancer types
n=length(cancer.labels)
```

```
#Creating a loop to find overexpressed or underexpressed genes of a particular cancer type
for (x in 1:n){
    type <- match.arg(type)

    #Getting a based on argument type
    if(type=="overexpressed"){
        a=2
    }
    else{
        a=1
    }
    i=fdr_rate/100

    print('x')
    print(x)
    #Getting data for a particular cancer type
    data=NCI60$data[NCI60$labs==cancer.labels[x],]

    y=as.numeric(sum(NCI60$labs==cancer.labels[x]))
    z=1:y
    nl=length(z)

    #Code Block for checking
    #total=total+nl

    #For saving all the plots that are generated by the FDR function
    graphics.off()
    name=paste("rplot",cancer.labels[x],"-",type,"-p","-q",fdr_rate,".jpg",sep = "")

    #For discarding cancer types with only 1 row of data
    if (nl==1){
        data=t(data)
            print("Only 1 row of data")
    }

    #Saving the plots and genes generated by the FDR Function
    else{
        jpeg(name)
            #getting answer as per type(p or (1-p))
            #removing NA values
            answer[[cancer.labels[x]]]=na.omit(try(fdr(apply(data,2,mytfunc)[a,],i)))
            dev.off()
    }
```

We made a loop that does the ttests for each cancer type and saves the plot generated by FDR. Also it stores the values of genes given by FDR.

- **Find the genes which either over express or under express in the cancer**

  As shown in the above loop we store the values of overexpressed and under expressed genes.

- **Then for cancers which have over or under expressed genes find if there are any common genes**

```
#Only created function to compare 2 cancer types as there were no common genes between 3 cancer types
#Creating a function to find common genes between two different cancer types
common_genes=function(x,y){

    Reduce(intersect, list(answer[[x]],answer[[y]]))
}

#Loop for generating common genes between all pairs of cancer types
for(x in 1:14){
    first=cancer.labels[x]
    for (y in 1:14){
        second=cancer.labels[y]
        name=paste("Common genes for ",first," and ",second)
        print(name)
        if(first!=second){
           print(try(common_genes(first,second)))
        }
    }

}
```

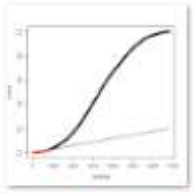
---

As shown above we find common genes between 2 types of cancer (every pair) through common genes function. For example: (Overexpressed genes with FDR 20%)

```
 "Common genes for  COLON  and  LEUKEMIA"
] 1868 1902 2130 1847 1697 1916 3296 2047 1871  392 3295 1728 1346 1841 1903 3294
] 1572 1915 1696 1904 1936 1968  232 1922 2282 1929 1694 2024  391 2385 2287 1021
] 2171 2423 2129 1887 2142 1901 1768 1777 1956 1894 1695 1918 2090 2038 3297 1864
] 1870 2151 2128 1691 2131 1769 1347 1519 2148 1815 2042 1762 3595 1538 2059 1563
] 2127 1835 1452 1849  468 2135 2008 1451 2283 1560 1313 1792 1914 3607 2943 1450
] 2147 1539 1453 1818 6774 2298 1850  739 1982 1301 2025 1927 2424 2091 1893 2049
] 2187 2245 3406
 "Common genes for  COLON  and  MELANOMA"
| 4550 3956 2228 1060 3957 3961 3304 4549  762 4057 3973 4245 4242 4635 4243 4595
| 3954  401 3286 3955 4058 3959 4634 4244 3977 4633 2895 4660
"Common genes for  COLON  and  RENAL"
  282 6535  252   98  152  354  355  100  151 5350  470  175 4251   11 5338
```

## FINDINGS:

- We understood the functioning of ttest and FDR.
- We saw how the numbers of elements increases as the rate of FDR increases.
- We automated the process of finding common underexpressed or overexpressed genes through R. For example: main_function has 2 arguments type and fdr_rate, main_function accepts “overexpressed” or “underexpressed” as argument type and FDR rate between [0,100]
- We are also attaching the plots generated by FDR function, the names of the plots indicate type of cancer, overexpression or underexpression of genes & the FDR rate. For example:



rplotCOLON-underexpressed-p-q20

(Colon cancer, underexpressed genes & FDR rate is 20%)

- We are also attaching a demo video which shows the working of the function which gives common genes.

```
> main_function("overexpressed",20)
[1] "x"
[1] 1
[1] "x"
[1] 2
[1] "x"
[1] 3
[1] "x"
[1] 4
[1] "x"
[1] 5
[1] "Only 1 row of data"
```

```
integer(0)
[1] "Common genes for  COLON  and  RENAL"
 [1]  282 6535  252   98  152  354  355  100  151 5350  470  175 4251   11 5338
[1] "Common genes for  COLON  and  BREAST"
integer(0)
[1] "Common genes for  COLON  and  NSCLC"
integer(0)
[1] "Common genes for  COLON  and  UNKNOWN"
NULL
[1] "Common genes for  COLON  and  OVARIAN"
integer(0)
[1] "Common genes for  COLON  and  MELANOMA"
 [1] 4550 3956 2228 1060 3957 3961 3304 4549  762 4057 3973 4245 4242 4635 4243 4595
[17] 3954  401 3286 3955 4058 3959 4634 4244 3977 4633 2895 4660
[1] "Common genes for  COLON  and  PROSTATE"
[1] 1121
[1] "Common genes for  COLON  and  LEUKEMIA"
 [1] 1868 1902 2130 1847 1697 1916 3296 2047 1871  392 3295 1728 1346 1841 1903 3294
[17] 1572 1915 1696 1904 1936 1968  232 1922 2282 1929 1694 2024  391 2385 2287 1021
[33] 2171 2423 2129 1887 2142 1901 1768 1777 1956 1894 1695 1918 2090 2038 3297 1864
[49] 1870 2151 2128 1691 2131 1769 1347 1519 2148 1815 2042 1762 3595 1538 2059 1563
[65] 2127 1835 1452 1849  468 2135 2008 1451 2283 1560 1313 1792 1914 3607 2943 1450
[81] 2147 1539 1453 1818 6774 2298 1850  739 1982 1301 2025 1927 2424 2091 1893 2049
[97] 2187 2245 3406
[1] "Common genes for  COLON  and  K562B-repro"
NULL
```

## UPDATED GOF:

We have updated the GOF function using quantiles.

```
> x=rbinom(100,5,0.5)
> gof_test(x,"binom")
[1] 0.594
> x=rbinom(100,1,0.5)
> gof_test(x,"bern")
[1] 0.921
> x=rbeta(100,5,3)
> gof_test(x,"beta")
[1] 0.104
> x=rchisq(100,5)
> gof_test(x,"chisq")
[1] 0.514
> x=rexp(100,5)
> gof_test(x,"exp")
[1] 0.222
> x=rgamma(100,10,5)
> gof_test(x,"gamma")
[1] 0.902
```

```
> x=rnorm(100,10,2)
> gof_test(x,"norm")
[1] 0.449
> x=rpois(100,6)
> gof_test(x,"pois")
[1] 0.474
> x=runif(100,6,20)
> gof_test(x,"unif")
[1] 0
>
```

```
for(i in 1:nboot) {

if (dist %in% c("geom", "pois", "exp", "chisq")) {
  xstar=rdist(n ,par)
  par <- mle(xstar,dist)
  y <- qdist(c(1:n)/(n+1) ,par)
  D.vec <- c(D.vec, kst(xstar, y))
  est_par <- c(est_par, c(par))
  input <- y
}
else {

  par <- mle(x,dist)


  # Only for Bernoulli distribution
  if (dist %in% c("bern")) {
    par[2] = par[1]
    par[1] = 1
    xstar=rdist(n ,par[1], par[2])
    y <- qdist(c(1:n)/(n+1) ,par[1], par[2])
    D.vec <- c(D.vec, kst(xstar, y))
    est_par <- c(est_par, c(par[1], par[2]))
    input <- y
  }

  if (dist %in% c("binom")) {
    par[1] = floor(par[1])
    xstar=rdist(n ,par[1], par[2])
    y <- qdist(c(1:n)/(n+1) ,par[1], par[2])
    D.vec <- c(D.vec, kst(xstar, y))
    est_par <- c(est_par, c(par[1], par[2]))
    input <- y
  }
  else{
  xstar=rdist(n ,par[1], par[2])
  y <- qdist(c(1:n)/(n+1) ,par[1], par[2])
  D.vec <- c(D.vec, kst(xstar, y))
  est_par <- c(est_par, c(par[1], par[2]))
  input <- y
  }
}
```