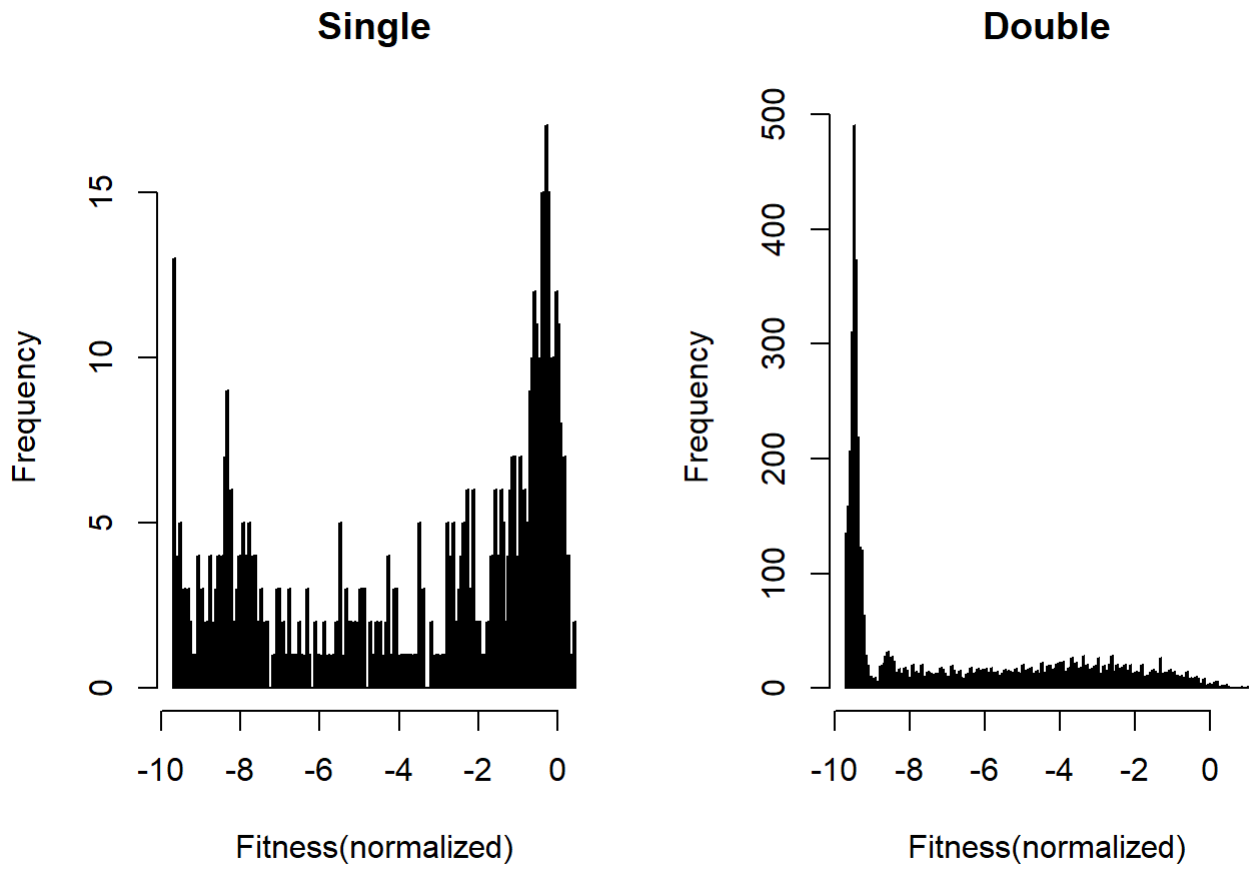
Error propagation

Get data

```
df.single.full<- read.table("lib1_normalized.txt", header = TRUE)
df.double.full<- read.table("lib6_normalized.txt", header = TRUE)

df.single<- df.single.full$Norm_5FOA
df.double<- df.double.full$Norm_5FOA
```

```
par(mfrow = c(1,2))
hist(df.single, breaks = 200, main = "Single", xlab = "Fitness(normalized)")
hist(df.double,breaks  = 200, main = "Double", xlab = "Fitness(normalized)")
```



Error propagation

Only for linear (addition and subtraction) Formula only:

Let's assume formula Y = a+b-c, and a,b,c have it's corresponding standard deviation $\sigma_a, \sigma_b, \sigma_c$

Then in general, $\sigma_Y = \sqrt{\sigma_a^2 + \sigma_b^2 + \sigma_c^2}$

Thus, $Y \pm \sigma_Y$

In your formula, Deviation = Fitness(AB)- (Fitness(A)+ Fitness(B)), D = F(AB) - (F(A)+F(B))

then, find the sd $\sigma_D^2 = \sigma_{F(AB)}^2 + \sigma_{F(A)}^2 + \sigma_{F(B)}^2$

Thus, we should have $D \pm \sigma_D$

The fitness data came from the median of 3 points. We propose to split the fitness into different groups by definition and then calculate the group variance.

Then for each data point, its variance will be the group variance.

Find σ:

We considered *two ways* to find σ, it is your choice to make which one you would like to use.

1. bootstrap

   (1). Directly calculate sd from group data

   (2). Bootstrap:

   - Using the data with n elements, we select n samples from data with replacement so that some elements can be selected more than once.
   - calculate the standard deviation based on selected data in (1)
   - repeat (1) & (2) for R = 1000 times; and now we have sd1, sd2, sd3, … , sd1000
   - calculate the mean of sampled standard deviations, providing a new sd.(sd_distribution = mean(sd1, sd2, sd3, … , sd1000))

In the end of this section I will show a table with both method's sd values.

Bootstrap sd function used to calculate sd in boot function

```
#bootstrap standard deviation function
bs.sd<- function(data, index){
  d.index<- data[index]
  return(sd(d.index))}
```

2. Standard Error:

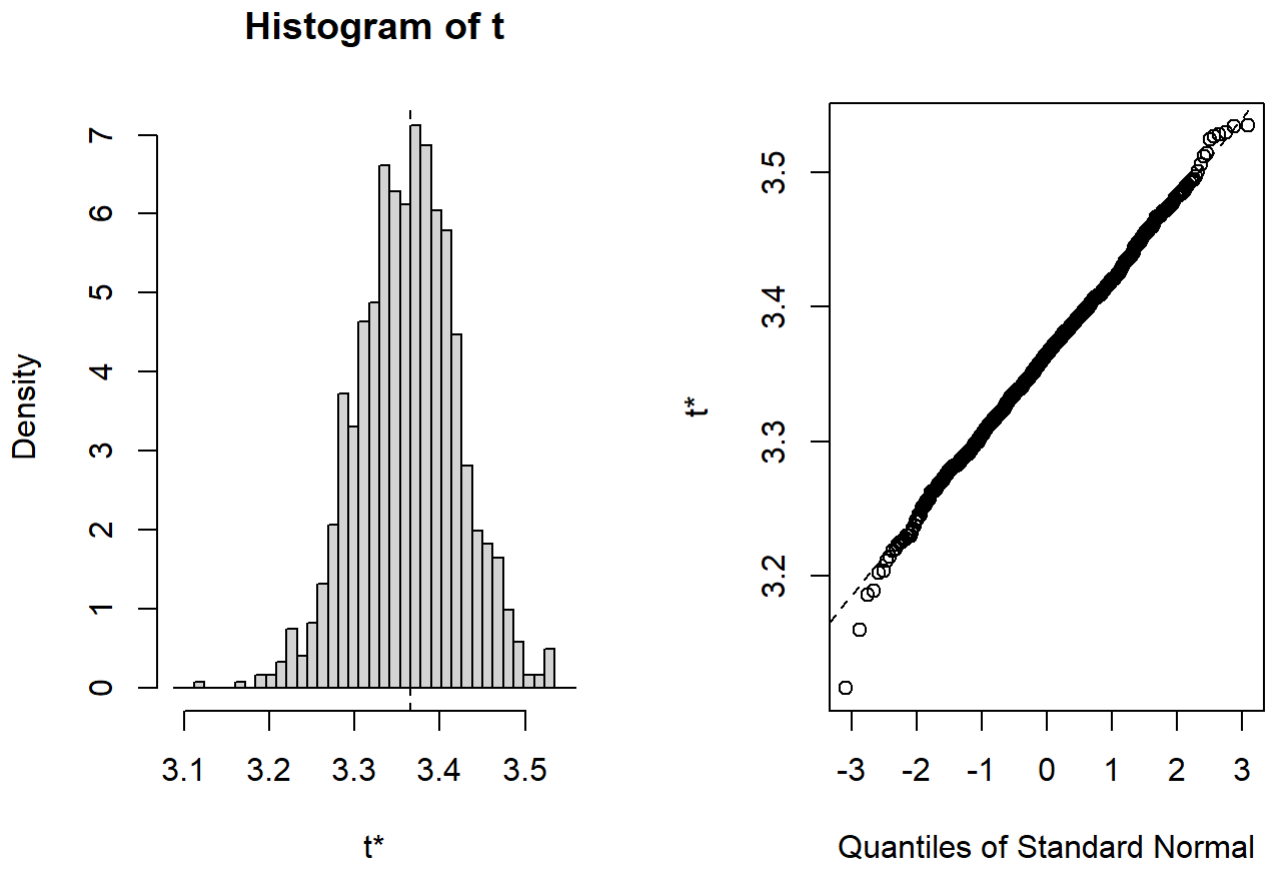
   - I could not find a good way to approximate the standard error for median when N is small (in your case N = 3)
   - Formula for standard error of median can be approximated by $\sigma_{median}^2 = \frac{\sigma^2}{N}\frac{pi}{2}$ when N is large, and $sigma$ is standard deviation of median.
   - I ranned a simulation a while ago, and it showed that when N is small the approximation is very inaccurate, if you are interested in the result please let me know and I could send you the code.
   - If we use the well-known se formula ($se = sd/\sqrt{N}$), this method should not be used, because it used the mean rather than median when calculating the standard error, but I have included the result in the end of the document anyway.

1. Bootstrap sd calculation

Single mutant

whole data

```
bot<- boot(df.single, bs.sd, R=1000)
bs.whole.single<- mean(bot$t)
calc.whole.single<- bot$t0
plot(bot)
```



Split data

This function generate:the standard deviation, bootstrap standard deviation, and bootstrap sd confidence interval.

```
Input, format: vector
- data we use to generate bootstrap
- threshold, group threshold, you dont need to worry about order
- group_name- the name of the group from healthy to sick

Output, format: table
- col1: bootstrap group sd
- col2: bootstrap group sd 95% lower ci
- col3: bootstrap group sd 95% upper ci
- col4: group sd direct calculation
```

```
sd.generator<- function(df, threshold, group_name){
  threshold<- sort(c(max(df),threshold, min(df)), decreasing = TRUE)
  bs.group<- calc.direct<-ci.l<- ci.u<-  c()
  for (i in 1:(length(threshold)-1)) {
    ### threshold bound
    upper<- threshold[i]
    lower<- threshold[i+1]

    ### locate the data
    loc<-  as.numeric(which((df>=lower) &(df<= upper)))
    dat<- df[loc]
    ### get bootstrap sd information
    bot<- boot(dat, bs.sd, R = 1000)

    #prepare print sd data
    bs.group<- c(bs.group,mean(bot$t) )
    quantile<- quantile(bot$t, c(0.025, 0.975))
    ci.l<- c(ci.l, quantile[1])
    ci.u<- c(ci.u, quantile[2])
    calc.direct<- c(calc.direct, bot$t0)
  }

  group.compare<- cbind(bs.group,ci.l, ci.u , calc.direct)
  colnames(group.compare)<- c("bs.group","bs.ci.l", "bs.ci.u" , "calc.direct")
  rownames(group.compare)<- group_name
  return(group.compare)
}
```

If you want to modify the threshold and groups just change the threshold, and add or change group name.

```
threshold<- c(-3,-6)
group_name<- c("health", "sick", "ultrasick")
```

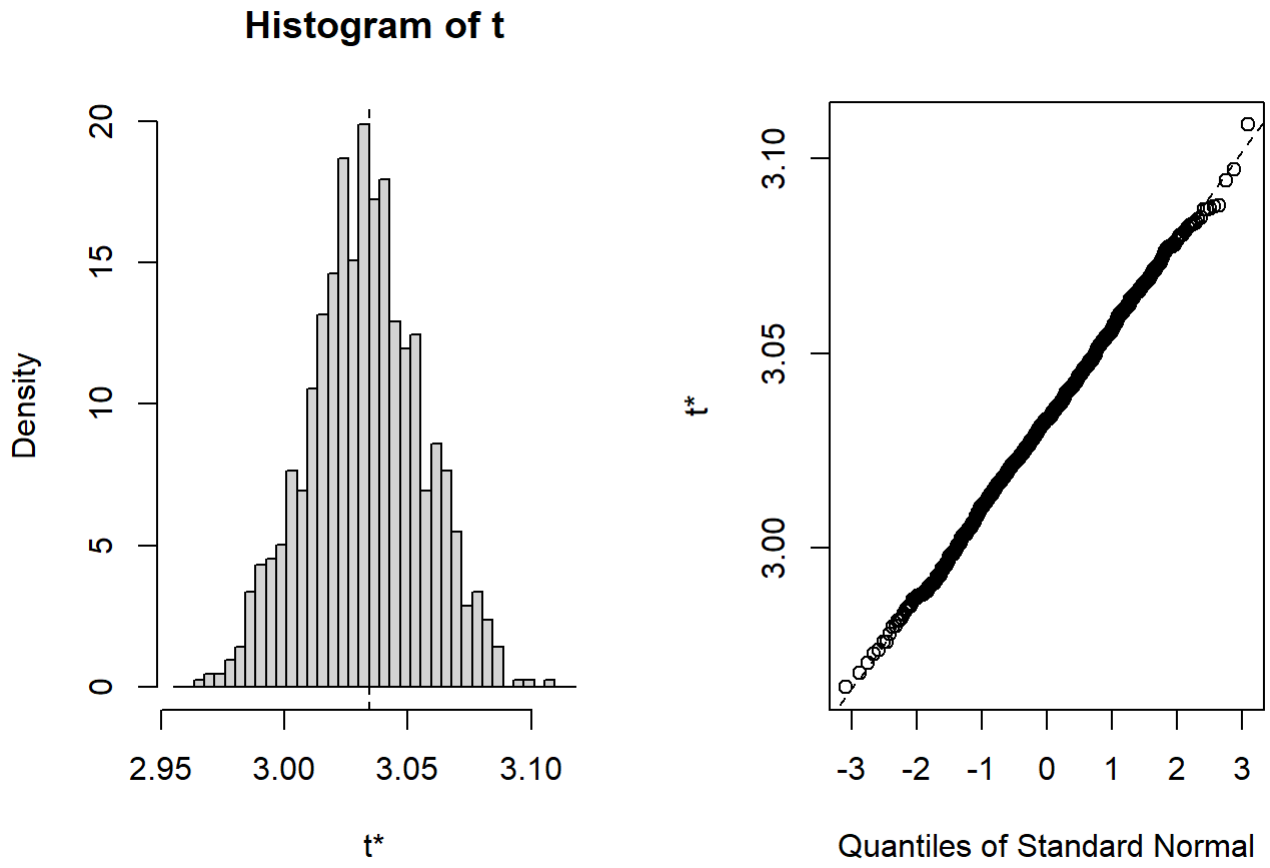
```
single<- sd.generator(df.single,threshold, group_name)
single
```

```
##            bs.group   bs.ci.l   bs.ci.u calc.direct
## health    0.8596125 0.8044618 0.9100003   0.8605537
## sick      0.8027496 0.7238139 0.8766393   0.8096627
## ultrasick 0.9654505 0.8740006 1.0435966   0.9710311
```

Double mutant

whole data

```
bot<- boot(df.double, bs.sd, R=1000)
bs.whole.double<- mean(bot$t)
calc.whole.double<- bot$t0
plot(bot)
```



Split data

If you want to modify the threshold and groups just change the threshold, and add or change group name

```
threshold<- c(-3,-6)
group_name<- c("health", "sick", "ultrasick")
```

```
double<- sd.generator(df.double,threshold, group_name)
double
```

```
##            bs.group   bs.ci.l   bs.ci.u calc.direct
## health    0.8649464 0.8309615 0.8990652   0.8656387
## sick      0.8526139 0.8266858 0.8778442   0.8529068
## ultrasick 0.9355253 0.9035090 0.9676081   0.9356116
```

1. Standard Error

lib1_growth_fitness_chemicalAA_replicates.txt file only contains the 3 replicate data, other wise this code will not work

```
single.data<- read.table("lib1_growth_fitness_chemicalAA_replicates.txt", header = TRUE)

main= single.data$Mutant
sd.single<- apply((single.data[,-1]), 1, sd)

se.single = (sd.single)/2
names(se.single)<- main

double.data<- read.table("lib6_F.txt", header = TRUE)
main= rownames(double.data)
sd.double<- apply((double.data[,-1]), 1, sd)

se.double = (sd.double)/2
names(se.double)<- main
```

Error propagation example

1. Bootstrap data

1st double data

```
AB<- (df.double.full)[1,]
AB
```

	Median <int>	**Norm_5FOA** <dbl>
[F1084I,V1089I]	6	-3.978534

1 row

find corresponding A and B

```
A.loc<- grep("F1084I", df.single.full$mutation)
B.loc<- grep("V1089I", df.single.full$mutation)
A<- df.single.full[A.loc,]
A
```

	mutation <chr>	Norm_5FOA <dbl>
370	[F1084I]	-2.624358

1 row

```
B<- df.single.full[B.loc,]
B
```

	mutation <chr>	Norm_5FOA <dbl>
121	[V1089I]	-0.4175079

1 row

Deviation = Fitness(AB)- (Fitness(A)+ Fitness(B)), D = F(AB) - (F(A)+F(B))

```
D = AB$Norm_5FOA - (A$Norm_5FOA + B$Norm_5FOA)
D
```

```
## [1] -0.9366681
```

Group:

AB: Sick

A: Healthy

B: Healthy

$D \pm \sigma_D$

$\sigma_D^2 = \sigma_{F(AB)}^2 + \sigma_{F(A)}^2 + \sigma_{F(B)}^2$

sig.d function: take group number 1,2,3 and return σ_d

1- health 2- sick 3- ultrasick

Thus the σ_D is:

```
sig.d<- function(AB.group, A.group, B.group){
  AB<- double[AB.group,][c(1,4)]
  A<- single[A.group,][c(1,4)]
  B<- single[B.group,][c(1,4)]
  D = sqrt(AB^2+A^2+B^2)
  return(D)
}

sd_d<- sig.d(2,1,1)
sd_d
```

```
##    bs.group calc.direct
##    1.484863    1.486121
```

and the error propagation give us D is in range:

```
names<- c("l", "u")
bs<- D+c(-1,1)*sd_d[1]
direct<- D+c(-1,1)*sd_d[2]
result<- rbind(bs,direct)
colnames(result)<- names
result
```

```
##                l         u
## bs     -2.421531 0.5481948
## direct -2.422789 0.5494528
```

2. Standard error

[F1084I,V1089I]

se for single

```
A.loc<- grep("F1084I", names(se.single))
B.loc<- grep("V1089I", names(se.single))
A<- se.single[A.loc]
A
```

```
##    F1084I
## 0.2386792
```

```
B<- se.single[B.loc]
B
```

```
##   V1089I
## 0.108468
```

se for double

```
loc<- grepl("[F1084I,V1089I]", names(se.double),fixed = TRUE)
c<- se.double[loc]
c
```

```
## [F1084I,V1089I]
##       0.6346192
```

```
sd_d = sqrt(A^2 + B^2 + c^2)
names(sd_d)<- NA
sd_d
```

```
##      <NA>
## 0.6866401
```

```
D
```

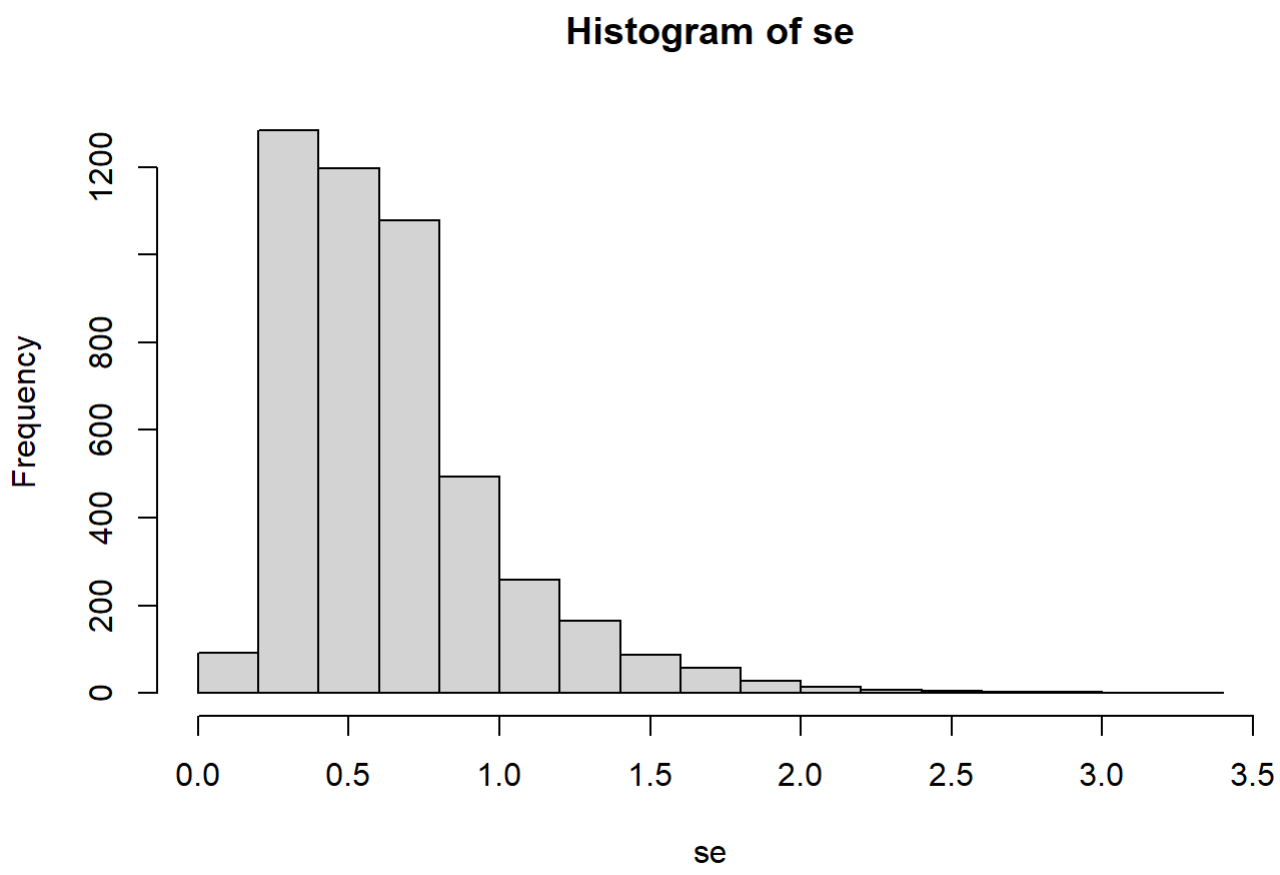
```
## [1] -0.9366681
```

```
bs<- D+c(-1,1)*sd_d
bs
```

```
## [1] -1.6233082 -0.2500279
```

```
se<- NULL
for (i in 1:length(se.double)) {
  string<- strsplit(names(se.double)[i], "\\[|\\]|,")[[1]][2:3]
  A.loc<- which(grepl(string[1], names(se.single)))
  B.loc<- which(grepl(string[2], names(se.single)))
  A<- se.single[A.loc]
  B<- se.single[B.loc]
  AB<- se.double[i]
  se<- c(se, sqrt(AB^2+A^2+B^2))}

hist(se)
```



Use this method it varies from small to large, but it is a good choice to use if we are using the average rather than median.

However, the median is the best choice in your case to select fitness, we should not use this method.