In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sns
from collections import Counter

import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
path = "/BiO/Preterm/raw_data/"
```

In [ ]:
```
#df = pd.read_csv(path+"case_control.csv",dtype = object, low_memory=Fal
se)
```

In [ ]:
```
df.shape
```

In [ ]:
```
df.head(2)
```

In [ ]:
```
#df.columns
pos = [75245928, 20326677, 130898247]
#df[df.POS.isin(pos)]
```

We take out 100 % SNP in Case Only

In [ ]:
```
df.columns
```

In [6]:
```
cas_col = ['1119_CS', '910_CS', '873_CS','880_CS','1282_CS','1584_CS','8
75_CS','1009_CS','1489_CS','989_CS']
con_col = ['1093_CNT', '1134_CNT', '1293_CNT', '1428_CNT', '1477_CNT', '
1481_CNT','1557_CNT', '1591_CNT', \
           '1316_CNT', '1389_CNT']
```

In [ ]:
```
case_control_100 = df[(df.loc[:,con_col].isnull().sum(axis =1)>9)
    &(df.loc[:,cas_col].isnull().sum(axis =1)<1)]
```

In [ ]:
```
case_control_100
case_control_100.to_csv(path+"case_control_100.csv",index = False)
```

Again Just Opposite 100% SNP in Control but not in Case

In [ ]:
```
control_case_100 = df[(df.loc[:,con_col].isnull().sum(axis =1)<1)
    &(df.loc[:,cas_col].isnull().sum(axis =1)>9)]
```

In [ ]:
```
control_case_100
#Save this for Future Fast Processing
#control_case_100.to_csv(path+"control_case_100.csv",index=False)
```

Since 100 % may not always be Found so lets Think For 90% with many Factors

In [ ]:
```
case_con_90= df[(df.loc[:,con_col].isnull().sum(axis =1)>9) #na in Contr
ol But Not in Case
    &(df.loc[:,cas_col].isnull().sum(axis =1)<2)]
```

In [ ]:
```
case_con_90["CHRM_POS"] = (case_con_90["CHROM"]+"_"+case_con_90.POS.asty
pe(str))
case_con_90 = case_con_90.rename(index= case_con_90["CHRM_POS"])#.drop(c
olumns=["POS","CHROM"])
#case_con.head(2)
```

In [ ]:
```
fig = plt.figure(figsize=(15,10))
plt.title("Heat Map REF and ALT")
plt.xlabel("Case(c) & Control(cn)")
plt.ylabel("Position of Sequence (SNP)")
sns.heatmap(case_con_90.loc[:,cas_col+con_col].isnull(),cmap=ListedColor
map(['red','green']),\
            cbar = False ,linewidth = 0.2,annot = True)
plt.savefig("../heatMeap_case_control_90_.png")
plt.show
#y Index is Not Proper Set Pos

#Cool Warm is Con & Case cmap="coolwarm"
```

In [ ]:
```
sns.countplot(case_con_90["CHROM"])
plt.xlabel("Chromosome Name")
plt.ylabel("Repeting Count")
plt.savefig("../chr_n_Rep_90.png")
```

Check For 80 %

In [ ]:
```
#We can see that the Case 2 People 1584 and 880 are exception
```

In [ ]:
```
case_con_80= df[(df.loc[:,con_col].isnull().sum(axis =1)>8) #na in Contr
ol But Not in Case
    &(df.loc[:,cas_col].isnull().sum(axis =1)<2)]
```

In [ ]:
```
case_con_80["CHRM_POS"] = (case_con_80["CHROM"]+"_"+case_con_80.POS.asty
pe(str))
case_con_80 = case_con_80.rename(index= case_con_80["CHRM_POS"])#.drop(c
olumns=["POS","CHROM"])
#case_con.head(2)
```

In [ ]:
```
case_con_80.shape
```

In [ ]:
```
fig = plt.figure(figsize=(20,15))
plt.title("Heat Map REF and ALT")
plt.xlabel("Case(c) & Control(cn)")
plt.ylabel("Position of Sequence (SNP)")
sns.heatmap(case_con_80.loc[:,cas_col+con_col].isnull(),cmap=ListedColor
map(['red','green']),\
            cbar = False ,linewidth = 0.2,annot = True)
plt.savefig("../heatMeap_case_control_80_.png")
plt.show
#y Index is Not Proper Set Pos

#Cool Warm is Con & Case cmap="coolwarm" cmap=ListedColormap(['red','gre
en'])
```

In [ ]:
```
sns.countplot(case_con_80["CHROM"])
plt.xlabel("Chromosome Name")
plt.ylabel("Repeting Count")
plt.savefig("../chr_n_Rep_80.png")
```

In [ ]:
```
#Check with 70%
```

In [ ]:
```
case_con_70= df[(df.loc[:,con_col].isnull().sum(axis =1)>7) #na in Contr
ol But Not in Case
    &(df.loc[:,cas_col].isnull().sum(axis =1)<3)]
```

let's see the Aggregate Figure

In [ ]:
```
#Save csv File for Later Fast Processing
case_control_90.to_csv(path+"case_control_90.csv",index= False)
```

In [ ]:
```
case_control_90.head(2)
```

In [ ]:
```
#case_control_90["CHRM_POS"] = (case_control_90["CHROM"]+"_"+case_contro
l_90.POS.astype(str))
#case_control_90 = case_control_90.rename(index= case_control_90["CHRM_P
OS"])#.drop(columns=["POS","CHROM"])

#case_con["CHRM_POS"] = (case_con["CHROM"]+"_"+case_con.POS.astype(str))
#case_con = case_con.rename(index= case_con["CHRM_POS"])#.drop(columns=
["POS","CHROM"])
#case_con.head(2)
```

In [ ]:
```
#case_control_90.head()
```

In [ ]:
```
#case_con.head()
```

In [ ]:
```

```

Just Testing with in 80 %

In [ ]:
```
con_case= df[(df.loc[:,con_col].isnull().sum(axis =1)<2) #na not in Con
but in Case
    &(df.loc[:,case_col].isnull().sum(axis =1)>8)] #Welcome NaN

#Doing at 90 %
```

In [ ]:
```
#case_con.to_csv(path +"con_on_case.csv",index=False)
#Read this File Directly From Here
```

In [ ]:
```

```

In [ ]:
```
case_con= df[(df.loc[:,con_col].isnull().sum(axis =1)>8) #na in Control
But Not in Case
    &(df.loc[:,cas_col].isnull().sum(axis =1)<2)]
```

In [ ]:
```
case_con.head()
```

In [ ]:
```
#case_col = ["1c","2c","3c","4c","5c","6c","7c","8c","9c","10c"]
#con_col = ["1cn","2cn","3cn","4cn","5cn","6cn","7cn","8cn",'9cn','10cn
']
```

In [ ]:
```
#case_con.to_csv(path +"../case_on_control_whole80.csv",index=False)
#con_case.to_csv(path+"../con_case_whole.csv",index=False)
#when need this file read

#case_con = pd.read_csv("/BiO/Preterm/case_on_control_whole.csv")
```

In [ ]:
```
#case_con_merge = pd.DataFrame(columns= list(case_con.columns))
#case_con_merge.head()
```

In [ ]:
```
#for easy of Use we merge both case & Control
#case_con_merge = case_con.merge(con_case,how="outer",on =list(case_con.
columns))
#case_con_merge.to_csv(path+"../con_case_merge.csv",index=False)
```

In [ ]:
```
#case_con_merge.columns
#case_con.shape
```

In [ ]:
```
case_con.head(2)
```

In [ ]:
```
case_con["CHRM_POS"] = (case_con["CHROM"]+"_"+case_con.POS.astype(str))
case_con = case_con.rename(index= case_con["CHRM_POS"])#.drop(columns=["
POS","CHROM"])
case_con.head(2)
```

In [ ]:
```
case_con.shape == case_control_90.shape
```

In [ ]:
```
fig = plt.figure(figsize=(10,10))
plt.title("Heat Map REF and ALT")
plt.xlabel("Case(c) & Control(cn)")
plt.ylabel("Position of Sequence (SNP)")
sns.heatmap(case_con.loc[:,cas_col+con_col].isnull(),cmap=ListedColormap
(['red','green']),\
            cbar = False ,linewidth = 0.2,annot = True)
plt.savefig("../heatMeap_case_control_70_.png")
plt.show
#y Index is Not Proper Set Pos

#Cool Warm is Con & Case cmap="coolwarm"
```

In [ ]:
```
case_con.shape
```

In [ ]:
```
plt.figure(figsize=(15,8))
sns.countplot(case_con["CHROM"],
              order = case_con.CHROM.value_counts().index)
plt.xlabel("Chromosome Name_POsition")
plt.ylabel("Repeting Count")
plt.savefig("../CHRM_count_80.png")
```

In [ ]:
```
con_case["CHRM_POS"] = (con_case["CHROM"]+"_"+con_case.POS.astype(str))
con_case= con_case.rename(index= con_case["CHRM_POS"])#drop(columns=["PO
S","CHROM"])
con_case.head()
```

In [ ]:
```
fig = plt.figure(figsize=(10,10))
plt.title("Heat Map REF and ALT")
plt.xlabel("Case(c) & Control(cn)")
plt.ylabel("Position of Sequence (SNP)")
sns.heatmap(con_case.loc[:,case_col+con_col].isnull(),cmap=ListedColorma
p(['red','green']),\
            cbar = False ,linewidth = 0.2,annot = True)
plt.savefig("../heatMap_control_case_80.png")
plt.show
#y Index is Not Proper Set Pos

#Cool Warm is Con & Case cmap="coolwarm"  cmap=ListedColormap(['red','gr
een'])
```

Part 2 Method Of Analysis

Method 2

In [ ]:
```
#Now We Analyse
```

In [ ]:
```
print(df.shape)
df =df[df.isna().sum(axis=1)<5]
df.shape
```

In [ ]:
```
df.head()
```

Read From Here Direct Fast

In [36]:
```
#df.head()
#df.to_csv(path+"case_control_na_moved.csv",index=False)
df = pd.read_csv(path + "case_control_na_moved.csv")
```

Adding all value in a single

In [37]:
```
df["Case"]= df[cas_col].apply(lambda row  :",".join(row.values.astype(st
r)),axis = 1)
df["Control"] = df[con_col].apply(lambda row : ",".join(row.values.astyp
e(str)),axis =1)
```

In [38]:
```
print(df.shape)
df.head(2)
```

```
(7468522, 25)
```

Out[38]:

	CHROM	POS	REF	1119_CS	910_CS	873_CS	880_CS	1282_CS	1584_CS	875_CS	...	1293_CNT
0	chr1	13656	CAG	CAG/C	CAG/C	CAG/C	CAG/C	CAG/C	CAG/C	NaN	...	CAG/C
1	chr1	15211	T	T/G	T/G	G/G	T/G	T/G	T/G	NaN	...	T/G

2 rows × 25 columns

Now We keep all datas (all Columns)

In [ ]:
```
#df = df.loc[:,["CHROM","POS","REF","case","Control"]]
#print(df.shape)
#df = df[df.loc[:,["CHROM","POS","REF","Control","Case"]]]
#df= df.loc[:,["CHROM","POS","REF","Case","Control"]]
```

In [9]:
```
#df =df[~(df.Control == df.Case)]
#df.shape
```

Out[9]:
```
(7083271, 25)
```

In [39]:
```
df["Control_Max"] = df.Control.apply(lambda x:(Counter(x.split(","))).mo
st_common(1))
df["Case_Max"] = df.Case.apply(lambda x:(Counter(x.split(","))).most_com
mon(1))
```

In [11]:
```
df.shape
```

Out[11]:
```
(7083271, 27)
```

In [45]:
```
#We Only Need to keep different data to Each Other
#If same Values are Repeating Highly in Both Case and Control then We Dr
op these kinds of Datas
#equal = (df.Control_Max.map(lambda x : x[0][0]) !=df.Case_Max.map(lambd
a x : x[0][0]))
no_equal = (df.Control_Max.map(lambda x : x[0][0]) !=df.Case_Max.map(lam
bda x : x[0][0]))
```

In [15]:
```
no_equal.head(2)
```

Out[15]:
```
0    False
1    False
dtype: bool
```

For Fast Processing We assign df to only No Eq datas

In [47]:
```
df = df[no_equal]
#no_equal.shape, df.shape
```

Till Now the data Size is Big so reduce to Greater than 8

Now We Do Percent wise Comparison

Take datas that are more than 8

In [ ]:
```
#Both method Give the same output
#df_80=df[(df.Control_Max.map(lambda x : x[0][1]) >8) & (df.Case_Max.map
(lambda x : x[0][1]) >8)]
```

In [56]:
```
fil_co = ["Control_Max","Case_Max"]
#First Condition is for First input Control Max and Second is for Case m
ax
Result= df[fil_co].apply(lambda row  : True if ((row.values[0][0][1]>8)&
(row.values[1][0][1]>8)) else False , axis=1)
```

In [64]:
```
#Save 80 % more different Datas for Fast Processing
#no_eq_80 = df[Result]
#no_eq_80.isin(df_80) , df_80.shape
```

Take Out data With 90 % Accuracy

In [65]:
```
#df_90=df[(df.Control_Max.map(lambda x : x[0][1]) >8) & (df.Case_Max.map
(lambda x : x[0][1]) >8)]

#Since we no need to chekc whole data because we already Reduced to 80 %

df_90=df_80[(df_80.Control_Max.map(lambda x : x[0][1]) >8) & (df_80.Case
_Max.map(lambda x : x[0][1]) >8)]
```

In [66]:
```
df_90
```

Out[66]:

	CHROM	POS	REF	1119_CS	910_CS	873_CS	880_CS	1282_CS	1584_CS	875_CS	...
82000	chr1	85980715	C	C/T	C/T	C/T	C/T	C/T	C/T	C/T	...
82256	chr1	85980715	C	C/T	C/T	C/T	C/T	C/T	C/T	C/T	...
82512	chr1	85980715	C	C/T	C/T	C/T	C/T	C/T	C/T	C/T	...
83024	chr1	85980715	C	C/T	C/T	C/T	C/T	C/T	C/T	C/T	...
84048	chr1	85980715	C	C/T	C/T	C/T	C/T	C/T	C/T	C/CAT	...
...	...	...	...	...	...	...	...	...	...	...	...
6761704	chr17	21521488	T	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	...
6761736	chr17	21521488	T	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	...
6761800	chr17	21521488	T	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	...

In [71]:
```
df_90.to_csv(path+"case_cont_diff_90.csv",index = False)
```

100 % Opposite SNP to each Other

In [70]:
```
#since Our data Set is 10 10 so max repeating is 10 i.e greater than 9
df_100=df_80[(df_80.Control_Max.map(lambda x : x[0][1]) >9) & (df_80.Cas
e_Max.map(lambda x : x[0][1]) >9)]
df_100.head()
```

Out[70]:

	CHROM	POS	REF	1119_CS	910_CS	873_CS	880_CS	1282_CS	1584_CS	875_(
1266226	chr2	91908521	T	T/A	T/A	T/A	T/A	T/A	T/A	T
2312755	chr2	91908521	T	T/TAA	T/TAA	T/TAA	T/TAA	T/TAA	T/TAA	T/T/
2890273	chr2	133106409	C	C/CAAAT	C/CAAAT	C/CAAAT	C/CAAAT	C/CAAAT	C/CAAAT	C/CAA/
5715143	chr17	21521488	T	T/A	T/A	T/A	T/A	T/A	T/A	T
6761672	chr17	21521488	T	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	T/TTC	T/T

5 rows × 27 columns

In [72]:
```
df_100.to_csv(path+"case_cont_diff_100.csv",index = False)
```

In [ ]:
```
plt.figure(figsize=(14,5))
sns.countplot(backup.CHROM,
              order =backup.CHROM.value_counts().index)
plt.xlabel("Chromosome Name_POsition")
plt.ylabel("Repeting Count")
#plt.savefig("../CHRM_Count_Opposite.png")
```