# Study on pedigree data

Importing necessary libraries and configurating plotting

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import plotly.io as pio
import plotly.express as px
from statistics import mode
from sklearn.cluster import KMeans
import scipy.interpolate
import plotly.figure_factory as ff

pio.renderers.default = 'browser'
plt.close('all')
```

Setting directory and uploading files.

short:Contains the ID's, phenotype, first year, last year alive, esimation of flowers and total number of fruits for all the plants.

pedigree:Contains the ID's,coordinates and phenotypes of kids and parents, the distance to the parents and the year of the seed, for the plants with parents identified.

In [2]:
```
dire='C:/Users/ferad/OneDrive/Escritorio/ISTA/dataFiles/'
short=pd.read_csv(dire+'short.csv', low_memory=False)
pedigree=pd.read_csv(dire+'pedigree.csv', low_memory=False)
```

Looks for plants without a first ID and adds it based on the ID2 column

In [3]:
```
misLabel=short.loc[short.X.notna() & short.ID2.notna() & short.ID.isnull() & short.FirstYear.notna()].copy() #plants with all data except id1, must be given id2
misLabel['ID']= misLabel['ID2'].str[0:5]

print('Number of flowers with missed ID on the first column: ', len(misLabel))
```

```
Number of flowers with missed ID on the first column: 9
```

Generate a data frame with the info of the plants presented in short but not in pedigree

In [4]:
```
short=pd.concat([short,misLabel],axis=0)
short=short[(short.X<420000) & (short.X>420000) & (short.Y<4680000) & (short.Y>4689000)] #Filter data from the center of the hybrid zone

del short['ID2'], misLabel
short=short.dropna(subset = ["ID"])
short=short.dropna(subset = ["X"])

missingParent=list(set(short['ID'].tolist()).difference(pedigree['Kid'].tolist())) #id list missing p
missP=short[short.ID.isin(missingParent)] #data frame from missingParent
```

Classify parent based on distance, logarithmic transformation

In [5]:
```
pedigree['closeP']=np.where(pedigree['Dist_pa']<pedigree['Dist_ma'],pedigree['Pa'],pedigree['Ma'])
pedigree['farP']=np.where(pedigree['Dist_pa']<pedigree['Dist_ma'],pedigree['Ma'],pedigree['Pa'])
pedigree['distClose']=np.where(pedigree['Dist_pa']<pedigree['Dist_ma'],pedigree['Dist_pa'],pedigree['Dist_ma'])
pedigree['distFar']=np.where(pedigree['Dist_pa']>pedigree['Dist_ma'],pedigree['Dist_ma'],pedigree['Dist_pa'])

del pedigree['Ma'], pedigree['Pa'],pedigree['Dist_pa'],pedigree['Dist_ma']

distC=pedigree['distClose'].tolist()
distC_log=np.log(np.array(distC)+1)
distF=pedigree['distFar'].tolist()
distF_log=np.log(np.array(distF)+1)
#plot
hist_data = [distC_log,distF_log]
group_labels = ['Distance Close', 'Distance far']
fig = ff.create_distplot(hist_data, group_labels, bin_size=10)
fig.show()
```

Figure 1) Distance from kid to parent with logarithmic conversion, dividing parents by distance in far and close

Change data frame such that there is an entry of every parent plant for every year that it was alive

In [6]:
```
parents=short[short.ID.isin(pedigree['closeP'].tolist()) | short.ID.isin(pedigree['farP'].tolist()) ]

year=2019
x=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), 'X'].tolist()
y=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), 'Y'].tolist()
ID=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), 'ID'].tolist()
nof1=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), ('TotalFlowers_'+str(year))].tolist()
nof=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), ('TotalFruit_'+str(year))].tolist()
pheno=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), 'pheno'].tolist()
years=[year]*len(x)
data={'ID':ID,'Year':years,'Pheno':pheno,'X':x,'Y':y,'noFlow':nof1,'noFr':nof} #add nof1 y nofr
parentsY=pd.DataFrame(data)

for i in range(int(min(parents['FirstYear'])),int(max(parents['LastAlive']))): #change from top to bottom
    year=int(i)
    x=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), 'X'].tolist()
    y=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), 'Y'].tolist()
    ID=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), 'ID'].tolist()
    #nof=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), ('TotalFlowers_'+str(year))].tolist()
    pheno=parents.loc[lambda parents: ((parents['FirstYear']<=year) & (parents['LastAlive']>=year)), 'pheno'].tolist()
    years=[year]*len(x)
    try:
        nof=parents.loc[lambda parents: ((parents['FirstYear']==year) & (parents['LastAlive']==year)), ('TotalFruit_'+str(year))].tolist()
    except:
        nof=(np.nan)*len(x)
        pass
    data={'ID':ID,'Year':years,'Pheno':pheno,'X':x,'Y':y,'noFr':nof#add nof1 y nofr
    parents0=pd.DataFrame(data)
    parentsY=pd.concat([parentsY,parents0])
```

After this change we can find a small amount of parents that were not recorded on the year of seed production

In [7]:
```
###Ids en number of total flowers
year=pedigree.groupby('year_seed')
closeP=year['closeP'].value_counts().to_frame()
farP=year['farP'].value_counts().to_frame()

farP.index.names=['year_seed','closeP']
a=closeP.add(farP, fill_value=0)
a=a.fillna(0)

p=a['closeP']+a['farP']
p=p.reset_index()
p.columns = ['Year','ID','noKids']
parents=pd.merge(parentsY,p,on=['ID', 'Year'],how='outer')

print("Percentage of parents that weren't found in the year of seed production:")
print(parents['X'].isna().sum()/len(parents)*100, " %")
notyear=parents[parents.X.isna()]['Year']
```

```
Percentage of parents that weren't found in the year of seed production:
7.307051673844883  %
```

Does the year of the plant influences the probability of not finding the parent in the year of seed production?

In [8]:
```
notyear=parents[parents.X.isna()]['Year']
##normalized
freqNot=notyear.value_counts()
freqP=short['FirstYear'].value_counts()
ratio=freqNot/freqP
ratio=ratio.fillna(0)

fig = px.histogram(notyear, x="Year",labels={'x':'year', 'y':'missing recording of parents'},title='Histogram of missing parents on year of seed production')
fig.show()

fig = px.bar(ratio,labels={'x':'year', 'y':'missing recording of parents normalized by total plants in short'},title='Normalized histogram of missing parents on year')
fig.show()
```

Figure 2) Histogram of occurrences on missing parents on year of seed production

Figure 3) normalized occurrences on missing parents on year of seed production

Graph of the parent plants, the size determines the number of kids and the color of the phenotype, the year scale is the year of the seed production

In [9]:
```
for i in ['Pheno','X','Y','noFlow','noFr']: #Filling values year not encontered
    ref=parents.groupby('ID')[i].first().to_frame()
    parents[i] = parents[i].fillna(parents['ID'].map(ref[i]))

parents['Pheno'].fillna('NotIdentified', inplace=True)
parents['Pheno'] = parents['Pheno'].astype(str)
parents['Pheno']=parents['Pheno'].apply(lambda x: mode(x.split(';')))

parents=parents.reset_index()
parents['noKids'].fillna(0, inplace=True)
parents=parents.sort_values(by=['Year'])

fig = px.scatter(parents, x="X", y="Y",animation_frame ='Year', animation_group ="ID",
           size="noKids", color="Pheno", hover_name="ID",
           size_max=50,
           color_discrete_sequence=["red","grey", "yellow", "darkorange", "salmon","sandybrown","white"] ) #range_x=[100,100000], range_y=[25,90],

fig['layout'].pop('updatemenus') # optional, drop animation buttons
fig.show()
```

Figure 4) Data points on all the parents in pedigree, color determining the phenotype, size the amount of kids and the scale of time in years the year of seed production

## Which phenotype appears more frequently as a parent in pedigree?

In [10]:
```
###pie proportion with kids and color
fig = px.pie(parents, values='noKids', names='Pheno', title='Color of parents in pedigree',
             color_discrete_sequence=["red","yellow","salmon","sandybrown","grey","darkorange","white"])
fig.show()
```

Figure 5) Pie graph of the phenotype of parents

In [11]:
```
##Normalized with total in short
short=short.drop(14182)
freqCP=parents['Pheno'].value_counts()
freqC=short['pheno'].value_counts()
ratio=freqCP/freqC

fig = px.pie(values=ratio.values, names=ratio.index, title='Color of parents in pedigree normalized',
             color_discrete_sequence=["salmon","red","white","darkorange","sandybrown","yellow"])
fig.show()
```

Figure 6) Normalized pie graph of the phenotype of parents

## Would plants that live for longer have more kids?

In [12]:
```
years=parents.groupby('ID', as_index=False).agg(alive=('ID','count'),
                                                noKids=('noKids','sum'))

fig = px.pie(years, values='noKids', names='alive', title='Years alive of parents in pedigree')
fig.show()

fig = px.box(years, x="alive", y="noKids", points="all",
             labels={'alive': 'Years alive','noKids':'Number of Kids'})

fig.show()
```

Figure 7) Pie graph of the number of parents based on how many years they live

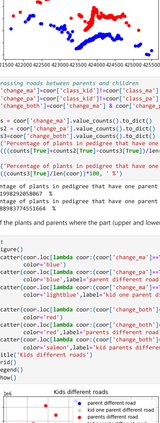
Figure 8) Box plot of the number of kids classified by how many years the parent live

## How common is for a plant to have parents from the other road?

In [13]:
```
###Line for crossing roads
coor=pedigree[['X_kid','Y_kid']].values
coor=coor=pedigree[['X_ma','Y_ma']].values
coor=coor=pedigree[['X_pa','Y_pa']].values

coor=pedigree.loc[:,['X_kid','Y_kid']]
coor.columns = ['X','Y']
ma=pedigree.loc[:,['X_ma','Y_ma']]
ma.columns = ['X','Y']
pa=pedigree.loc[:,['X_pa','Y_pa']]
pa.columns= ['X','Y']
coor=pd.concat([coor,ma],axis=0)
coor=pd.concat([coor,pa],axis=0)

coor=pedigree.drop(1046) #value too far

x_line=[421350,422220,423210,423630,424380,425820]
y_line=[4680250,4680520,4680320,4686110,4686200,4685910]

interpolations=scipy.interpolate.interp1d(x_line, y_line)
coor['class_kid']=coor.apply(lambda data: np.sign(data['Y_kid']-interpolations(data['X_kid'])),axis=1)
coor['class_ma']=coor.apply(lambda data: np.sign(data['Y_ma']-interpolations(data['X_ma'])),axis=1)
coor['class_pa']=coor.apply(lambda data: np.sign(data['Y_pa']-interpolations(data['X_pa'])),axis=1)
```

Clasifying plants based on which road they are (upper and lower)

In [14]:
```
upper=coor.loc[coor['class_kid']==1,['X_kid','Y_kid']]
upper=coor.loc[coor['class_pa']==1,['X_pa','Y_pa']]
upper=coor.loc[coor['class_ma']==1,['X_ma','Y_ma']]

lower=coor.loc[coor['class_kid']==-1,['X_kid','Y_kid']]
lower=coor.loc[coor['class_pa']==-1,['X_pa','Y_pa']]
lower=coor.loc[coor['class_ma']==-1,['X_ma','Y_ma']]

plt.figure()
plt.scatter(upper['X_ma'].tolist(),
            upper['Y_ma'].tolist(),color='red')
plt.scatter(lower['X_ma'].tolist(),
            lower['Y_ma'].tolist(),color='blue')
plt.show()
```




In [15]:
```
### crossing roads between parents and children
coor['change_ma']=coor['class_kid']!=coor['class_ma']
coor['change_pa']=coor['class_kid']!=coor['class_pa']
coor['change_both']=coor['change_ma'] & coor['change_pa']

counts = coor['change_ma'].value_counts().to_dict()
counts2 = coor['change_pa'].value_counts().to_dict()
counts3=coor['change_both'].value_counts().to_dict()
print('Percentage of plants in pedigree that have one parent in the other road ')
print((counts[True]+counts2[True]-counts3[True])/len(coor))*100, ' %')

print('Percentage of plants in pedigree that have one parent in the other road ')
print((counts3[True])/len(coor))*100, ' %')
```

```
Percentage of plants in pedigree that have one parent in the other road
2.9461998292058067  %
Percentage of plants in pedigree that have one parent in the other road
0.2988898377455164  %
```

Plot of the plants and parents where the part (upper and lower) of the road of the parent is different to the one of the children

In [16]:
```
##plot
plt.figure()
plt.scatter(coor.loc[lambda coor:(coor['change_ma']==True), 'X_ma'].tolist(),coor.loc[lambda coor:(coor['change_ma']==True), 'Y_ma'].tolist(),
            color='blue')
plt.scatter(coor.loc[lambda coor:(coor['change_pa']==True), 'X_pa'].tolist(),coor.loc[lambda coor:(coor['change_pa']==True), 'Y_pa'].tolist(),
            color='blue',label='parent different road')
plt.scatter(coor.loc[lambda coor:(coor['change_ma']==True), 'X_kid'].tolist(),coor.loc[lambda coor:(coor['change_ma']==True), 'Y_kid'].tolist(),
            color='lightblue',label='kid one parent different road')

plt.scatter(coor.loc[lambda coor:(coor['change_both']==True), 'X_ma'].tolist(),coor.loc[lambda coor:(coor['change_both']==True), 'Y_ma'].tolist(),
            color='red')
plt.scatter(coor.loc[lambda coor:(coor['change_both']==True), 'X_pa'].tolist(),coor.loc[lambda coor:(coor['change_both']==True), 'Y_pa'].tolist(),
            color='red',label='parents different road')
plt.scatter(coor.loc[lambda coor:(coor['change_both']==True), 'X_kid'].tolist(),coor.loc[lambda coor:(coor['change_both']==True), 'Y_kid'].tolist(),
            color='salmon',label='kid parents different road')
plt.title('Kids different roads')
plt.grid()
plt.legend()
plt.show()
```