Data Scientist - Molecular Breeding -Interview

Part 1: Data Cleaning and Unify

By Zixiang Wen

The ultimate goal herein is to calculate the general combining ability (GCA) for three traits (yield, silk DAP and plant height) female parents across 4 years. BLUP of the three traits for each accession were calculated to replace means value in genomic selection analysis. As for genotypic data, SNPs with MAF<0.05 were discarded, genotypes with missing rate > 35% were discarded. Then genotypic data was converted to numerical data for genomic s selection analysis

1 Phenotypic data cleaning:

1.1 Work with 2017 hybrid data

1.2 Work with 2016 hybrid data

1.3 Work with 2015 hybrid data

1.4 Work with 2014 hybrid data

1.5 Combine all mean value for single line across all year and all location

1.6 Combine clean data for Best linear unbiased prediction (BLUP)

2 Genotypic data cleaning

2.1 Clean and unify genotype ID

2.2 SNP data cleaning based on MAF and missing rate

2.3 Transform SNP data to numerical data (0,1,2)

Datasets have been processed in this section:

g2f_2017_hybrid_data_clean.csv

g2f_2016_hybrid_data_clean.csv

g2f_2015_hybrid_data_clean.csv

g2f_2014_hybrid_data_clean.csv

g2f_2017_ZeaGBSv27_Imputed_AGPv4.h5

Data Scientist - Molecular Breeding -Interview-Part 1-Data Cleaning

November 26, 2019

0.0.1 1 Phenotypy data cleaning

(The ultimate goal herein is to caculate the general combining ability for female parents)

[2]:
```
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Setup Seaborn
sns.set_style("whitegrid")
sns.set_context("poster")
```

1.1 Work with 2017 hybrid_data

[2]:
```
p17 =pd.read_csv("g2f_2017_hybrid_data_clean.csv")
```

```
C:\Users\Ruijuan Tan\Anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3057: DtypeWarning: Columns (21,22)
have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

[3]:
```
p17.head()
```

[3]:

	Year	Field-Location	RecId	Source	\
0	2017	GAH1	4230119	LOCAL_CHECK_5	
1	2017	GAH1	3595936	15URN161-69:0760-0765	
2	2017	GAH1	3602424	16.2.19716.04555.0000000.M1	
3	2017	GAH1	3602498	16.2.19716.04188.0000000.M1	
4	2017	GAH2	4222112	S16 CR-0714	

	Pedigree	Replicate	Block	Plot	Range	Pass	...	\
0	DKC 67-88	1	1	100	NaN	NaN	...	
1	PHP38/PHN47	2	2	134	NaN	NaN	...	
2	2369/LH123HT	2	2	179	NaN	NaN	...	
3	NILASQ4G71I03S2/LH123HT	2	2	187	NaN	NaN	...	

```
4               CGR01/LH82          1      1     3    NaN   NaN  ...

  Root Lodging [plants]  Stalk Lodging [plants]  Grain Moisture [%]  \
0                   NaN                    NaN                14.1
1                   NaN                    NaN                13.8
2                   NaN                    NaN                14.1
3                   NaN                    NaN                13.9
4                  27.0                    NaN                17.5

   Test Weight [lbs/bu]  Plot Weight [lbs]  Grain Yield [bu/A]  \
0                  54.1               2.57               24.19
1                  46.5               2.17               20.50
2                  58.0               3.17               29.84
3                  52.7               2.53               23.87
4                  55.5               9.32               70.78

   Plot Discarded [enter "yes" or "blank"]  Comments  \
0                                      NaN       NaN
1                                      NaN       NaN
2                                      NaN       NaN
3                                      NaN       NaN
4                                      NaN       NaN

   Filler [enter "filler" or "blank"] [add additional measurements here]
0                                 NaN                                 NaN
1                                 NaN                                 NaN
2                                 NaN                                 NaN
3                                 NaN                                 NaN
4                                 NaN                                 NaN

[5 rows x 38 columns]
```

[4]:
```
## Select useful columns from the original data
p17c=p17[['Year','Field-Location','Pedigree','Replicate','Plant Height␣
 ↪[cm]','Silk DAP [days]','Grain Yield [bu/A]']]
```

[5]:
```
p17c.head()
```

[5]:
```
   Year Field-Location                Pedigree  Replicate  Plant Height [cm]  \
0  2017           GAH1               DKC 67-88          1              180.0
1  2017           GAH1             PHP38/PHN47          2              200.0
2  2017           GAH1            2369/LH123HT          2              190.0
3  2017           GAH1  NILASQ4G71I03S2/LH123HT          2              190.0
4  2017           GAH2              CGR01/LH82          1              201.0

   Silk DAP [days]  Grain Yield [bu/A]
0             60.0               24.19
1             60.0               20.50
```

```
2              63.0              29.84
3              60.0              23.87
4              51.0              70.78
```

[6]:
```
## chage columns names to clean ones
p17c.rename(columns={'Plant Height [cm]':'p_height','Silk DAP [days]':
 ↪'s_day','Grain Yield [bu/A]':'yield'},inplace=True)
```

```
C:\Users\Ruijuan Tan\Anaconda3\lib\site-packages\pandas\core\frame.py:4025:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  return super(DataFrame, self).rename(**kwargs)
```

[7]:
```
## Extract Female ID for GCA caculation
p17c[['Female','Male']] = p17c.Pedigree.str.split("/",expand=True)
```

```
C:\Users\Ruijuan Tan\Anaconda3\lib\site-packages\pandas\core\frame.py:3391:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  self[k1] = value[k2]
```

[8]:
```
p17_new=p17c
```

[31]:
```
## Calculate single value of GCA for each line grouped by␣
 ↪Field-Location,Replicate
cols=['Field-Location','Replicate','Female']
group_mean17=p17_new[['Year','Field-Location','Replicate','p_height','s_day','yield','Female']
 ↪groupby(cols).mean()
```

[32]:
```
group_mean17.reset_index()
```

[32]:

	Field-Location	Replicate	Female	Year	p_height	s_day	\
0	ARH1	1	2369	2017	209.875000	69.500000	
1	ARH1	1	2FACC	2017	164.500000	60.000000	
2	ARH1	1	4N506	2017	235.000000	65.000000	
3	ARH1	1	6F629	2017	194.300000	65.000000	
4	ARH1	1	78010	2017	162.100000	65.000000	
5	ARH1	1	A3G-3-3-1-313	2017	205.700000	65.000000	
6	ARH1	1	A632	2017	193.700000	65.000000	
7	ARH1	1	A634	2017	141.000000	65.000000	
8	ARH1	1	A635	2017	162.600000	65.000000	
9	ARH1	1	A679	2017	190.800000	65.000000	

10	ARH1	1	A680	2017	188.000000	65.000000
11	ARH1	1	B104	2017	198.800000	65.000000
12	ARH1	1	B105	2017	166.400000	65.000000
13	ARH1	1	B109	2017	195.600000	65.000000
14	ARH1	1	B110	2017	210.800000	65.000000
15	ARH1	1	B111	2017	NaN	NaN
16	ARH1	1	B119	2017	222.600000	65.000000
17	ARH1	1	B14A	2017	201.833333	65.000000
18	ARH1	1	B37	2017	210.800000	71.000000
19	ARH1	1	B73	2017	208.625000	69.500000
20	ARH1	1	B84	2017	215.900000	65.000000
21	ARH1	1	B97	2017	NaN	NaN
22	ARH1	1	BSSSC0_001	2017	188.000000	65.000000
23	ARH1	1	BSSSC0_002	2017	198.100000	65.000000
24	ARH1	1	BSSSC0_003	2017	NaN	NaN
25	ARH1	1	BSSSC0_005	2017	172.700000	65.000000
26	ARH1	1	BSSSC0_008	2017	193.000000	65.000000
27	ARH1	1	BSSSC0_009	2017	NaN	NaN
28	ARH1	1	BSSSC0_012	2017	213.400000	65.000000
29	ARH1	1	BSSSC0_015	2017	188.000000	65.000000
...	...	...	...	...	...	...
9388	WIH2	2	PHW03	2017	257.000000	70.000000
9389	WIH2	2	PHW52	2017	253.294118	76.470588
9390	WIH2	2	PHZ51	2017	280.000000	75.000000
9391	WIH2	2	R229	2017	NaN	NaN
9392	WIH2	2	S8324	2017	239.000000	70.285714
9393	WIH2	2	S8326	2017	260.000000	72.833333
9394	WIH2	2	SD101	2017	NaN	NaN
9395	WIH2	2	TR 9-1-1-6	2017	NaN	NaN
9396	WIH2	2	TX714	2017	235.000000	73.000000
9397	WIH2	2	VA35	2017	273.000000	79.000000
9398	WIH2	2	W10001_0022	2017	257.000000	73.000000
9399	WIH2	2	W10004_0007	2017	NaN	NaN
9400	WIH2	2	W10004_0026	2017	265.000000	73.000000
9401	WIH2	2	W10004_0041	2017	NaN	NaN
9402	WIH2	2	W10004_0045	2017	268.000000	73.000000
9403	WIH2	2	W10004_0062	2017	NaN	NaN
9404	WIH2	2	W10004_0072	2017	NaN	NaN
9405	WIH2	2	W10004_0095	2017	NaN	NaN
9406	WIH2	2	W10004_0119	2017	NaN	NaN
9407	WIH2	2	W10004_0121	2017	NaN	NaN
9408	WIH2	2	W10004_0148	2017	NaN	NaN
9409	WIH2	2	W10004_0208	2017	275.000000	73.000000
9410	WIH2	2	W10004_0212	2017	273.000000	75.000000
9411	WIH2	2	W10004_0216	2017	273.000000	78.000000
9412	WIH2	2	W10004_0225	2017	250.000000	73.000000
9413	WIH2	2	W10004_0258	2017	NaN	NaN

9414	WIH2	2	W10004_0292	2017	NaN	NaN
9415	WIH2	2	W10005_0107	2017	248.000000	75.000000
9416	WIH2	2	W37A	2017	257.000000	73.000000
9417	WIH2	2	WF9	2017	265.000000	78.000000

	yield
0	91.392500
1	103.570000
2	34.890000
3	71.380000
4	25.830000
5	72.940000
6	30.490000
7	92.820000
8	69.630000
9	72.960000
10	69.790000
11	74.830000
12	50.700000
13	112.270000
14	108.610000
15	NaN
16	143.870000
17	89.773333
18	121.670000
19	105.585000
20	96.640000
21	NaN
22	116.340000
23	116.740000
24	NaN
25	138.910000
26	96.650000
27	NaN
28	187.870000
29	71.710000
...	...
9388	158.608628
9389	205.613903
9390	99.710658
9391	NaN
9392	162.275430
9393	179.661959
9394	NaN
9395	NaN
9396	189.570355
9397	201.298550

```
9398  152.977846
9399         NaN
9400  207.675555
9401         NaN
9402  198.112139
9403         NaN
9404         NaN
9405         NaN
9406         NaN
9407         NaN
9408         NaN
9409  197.594125
9410  179.214604
9411  184.472399
9412  177.055182
9413         NaN
9414         NaN
9415   68.433018
9416  196.897945
9417  145.288316

[9418 rows x 7 columns]
```

[11]:
```
## Export a clean data for BLUP calculation
group_mean17.reset_index().to_csv('p17_clean.csv',index=False)
```

[13]:
```
single_line_mean17=group_mean17_n[['Year','p_height','s_day','yield','Female']].
 ↪groupby('Female').mean()
```

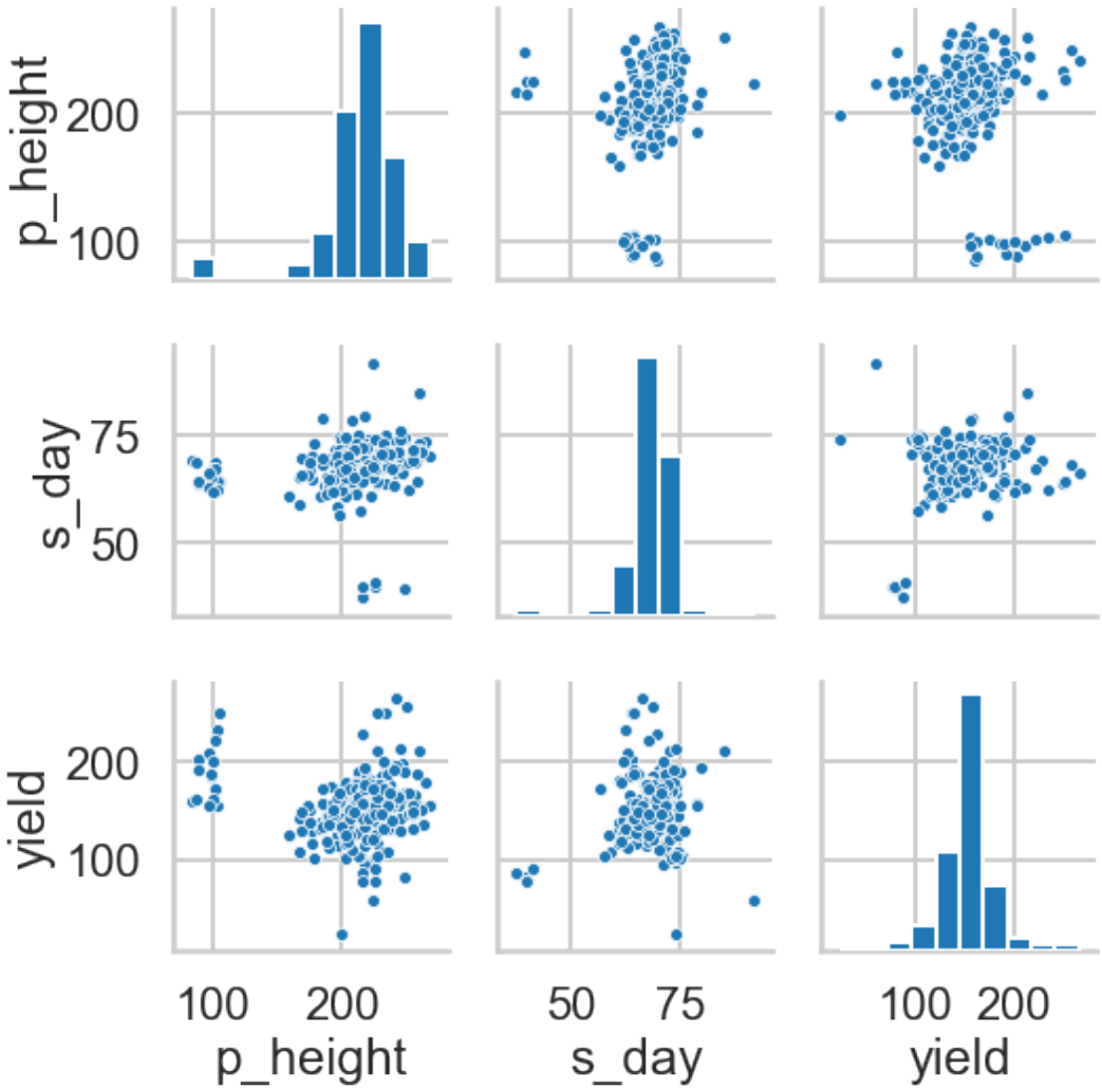
[14]:
```
single_line_mean17.head()
```

[14]:
```
        Year    p_height      s_day       yield
Female
2369    2017  222.467745  70.074208  153.581527
2FACC   2017  208.783333  67.000000  125.541667
31G66   2017  246.000000        NaN  199.180000
4N506   2017  263.600000  72.000000  136.514000
6F629   2017  193.908333  69.583333  158.786429
```

[15]:
```
single_line_mean17.to_csv('single_line_mean17.csv')
```

[90]:
```
sns.pairplot(single_line_mean17[['p_height','s_day','yield']],markers='.')
```

[90]:
```
<seaborn.axisgrid.PairGrid at 0xe5d00b8>
```



[13]:
```
group_mean17=group_mean.reset_index()
group_mean17
```

[13]:

	Field-Location	Replicate	Female	Year	p_height	s_day	yield
0	ARH1	1	2369	2017	209.875	69.5	91.392500
1	ARH1	1	2FACC	2017	164.500	60.0	103.570000
2	ARH1	1	4N506	2017	235.000	65.0	34.890000
3	ARH1	1	6F629	2017	194.300	65.0	71.380000
4	ARH1	1	78010	2017	162.100	65.0	25.830000
...	...	...	...	...	...	...	...
9413	WIH2	2	W10004_0258	2017	NaN	NaN	NaN
9414	WIH2	2	W10004_0292	2017	NaN	NaN	NaN
9415	WIH2	2	W10005_0107	2017	248.000	75.0	68.433018
9416	WIH2	2	W37A	2017	257.000	73.0	196.897945

```
9417          WIH2          2          WF9  2017   265.000   78.0  145.288316

[9418 rows x 7 columns]
```

[14]:
```
group_mean17[['p_height','s_day','yield']].describe()
```

[14]:
```
          p_height        s_day        yield
count  7368.000000  6000.000000  7628.000000
mean    216.127221    68.468101   152.424121
std      59.710377     8.357069    43.659151
min      71.000000    31.000000    10.880000
25%     194.000000    63.985294   124.942500
50%     229.000000    69.000000   156.625000
75%     255.000000    74.000000   183.252500
max     345.000000    96.000000   323.030000
```

1.2 Work with 2016 hybrid_data

[15]:
```
p16 =pd.read_csv("g2f_2016_hybrid_data_clean.csv")
```

[16]:
```
p16.head()
```

[16]:
```
   Year Field-Location    RecId                  Source       Pedigree  \
0  2016           ARH1  3094939              15SFG:2004     2369/PHZ51
1  2016           ARH1  3095402   15SJWE:G2F:11041/11020    LH195/PHN37
2  2016           ARH1  3093386  15URN161-69:0970 - 0975    PHHB9/PHM57
3  2016           ARH1  3093367  15URN161-69:0741 - 0746  PHW53/LH123HT
4  2016           ARH1  3085303            WISN15/50910     B119/3IIH6

   Replicate  Block  Plot  Range  Pass  ... Root Lodging [plants]  \
0          2      2     2    2.0   4.0  ...                   1.0
1          2      2    53    NaN   NaN  ...                   NaN
2          2      2   123   14.0   8.0  ...                   NaN
3          2      2   137   15.0   7.0  ...                   NaN
4          1      1   172    3.0   2.0  ...                   NaN

   Stalk Lodging [plants]  Grain Moisture [%]  Test Weight [lbs/bu]  \
0                     NaN                16.8                  61.2
1                     NaN                16.8                  61.5
2                     NaN                19.6                  59.6
3                     1.0                17.0                  61.4
4                     2.0                15.4                  62.0

   Plot Weight [lbs]  Grain Yield [bu/A]  \
0              23.12          111.836080
1              25.66          124.122570
2              18.10           84.606826
3              27.03          130.435233
4              24.25          119.275950
```

```
   Plot Discarded [enter "yes" or "blank"]  Comments  \
0                                      NaN       NaN
1                                      NaN       NaN
2                                      NaN       NaN
3                                      NaN       NaN
4                                      NaN       NaN

   Filler [enter "filler" or "blank"] [add additional measurements here]
0                                 NaN                                 NaN
1                                 NaN                                 NaN
2                                 NaN                                 NaN
3                                 NaN                                 NaN
4                                 NaN                                 NaN

[5 rows x 38 columns]
```

[17]:
```
p16c=p16[['Year','Field-Location','Pedigree','Replicate','Plant Height␣
 ↪[cm]','Silk DAP [days]','Grain Yield [bu/A]']]
```

[18]:
```
p16c.head()
```

[18]:
```
   Year Field-Location       Pedigree  Replicate  Plant Height [cm]  \
0  2016           ARH1     2369/PHZ51          2              188.0
1  2016           ARH1    LH195/PHN37          2              205.0
2  2016           ARH1    PHHB9/PHM57          2              170.0
3  2016           ARH1  PHW53/LH123HT          2              200.0
4  2016           ARH1     B119/3IIH6          1              175.0

   Silk DAP [days]  Grain Yield [bu/A]
0             73.0          111.836080
1             69.0          124.122570
2             75.0           84.606826
3             72.0          130.435233
4             69.0          119.275950
```

[19]:
```
## chage clean column name
p16c.rename(columns={'Plant Height [cm]':'p_height','Silk DAP [days]':
 ↪'s_day','Grain Yield [bu/A]':'yield'},inplace=True)
```

[20]:
```
## Extract Female ID for GCA caculation
p16c[['Female','Male']] = p16c.Pedigree.str.split("/",expand=True)
```

[21]:
```
p16c
```

[21]:
```
       Year Field-Location          Pedigree  Replicate  p_height  s_day  \
0      2016           ARH1        2369/PHZ51          2     188.0   73.0
1      2016           ARH1       LH195/PHN37          2     205.0   69.0
2      2016           ARH1       PHHB9/PHM57          2     170.0   75.0
3      2016           ARH1     PHW53/LH123HT          2     200.0   72.0
```

```
4      2016           ARH1         B119/3IIH6          1     175.0   69.0
...     ...            ...                ...        ...       ...     ...
16372  2016           WIH2   MBNIL B078/PHZ51          2       NaN    NaN
16373  2016           WIH2   MBNIL B131/PHZ51          2       NaN    NaN
16374  2016           WIH2  BGEM-0088-N/LH195          2       NaN    NaN
16375  2016           WIH2  BGEM-0120-N/LH195          2       NaN    NaN
16376  2016           WIH2         LE23/3IIH6          2       NaN    NaN

            yield       Female     Male
0      111.836080         2369    PHZ51
1      124.122570        LH195    PHN37
2       84.606826        PHHB9    PHM57
3      130.435233        PHW53  LH123HT
4      119.275950         B119    3IIH6
...           ...          ...      ...
16372         NaN   MBNIL B078    PHZ51
16373         NaN   MBNIL B131    PHZ51
16374         NaN  BGEM-0088-N    LH195
16375         NaN  BGEM-0120-N    LH195
16376         NaN         LE23    3IIH6

[16377 rows x 9 columns]
```

[22]:
```
## Calculate single value of GCA for each line grouped by␣
 ↪Field-Location,Replicate
cols=['Field-Location','Replicate','Female']
group_mean=p16c[['Year','Field-Location','Replicate','p_height','s_day','yield','Female']].
 ↪groupby(cols).mean()
```

[23]:
```
group_mean.reset_index().to_csv('p16_clean.csv',index=False)
```

[24]:
```
group_mean16=group_mean.reset_index()
group_mean16
```

[24]:
```
     Field-Location  Replicate       Female  Year  p_height  s_day       yield
0              ARH1          1         2369  2016     187.0   70.0  119.950639
1              ARH1          1        2FACC  2016     198.0   67.0  112.655439
2              ARH1          1        4N506  2016     195.0   69.0   90.671176
3              ARH1          1        6F629  2016     192.0   67.0  118.191476
4              ARH1          1         A632  2016     203.0   69.0   90.888617
...             ...        ...          ...   ...       ...    ...         ...
8569           WIH2          2  W10004_0258  2016     285.0   61.0  109.926403
8570           WIH2          2  W10004_0292  2016     248.0   60.0   90.957122
8571           WIH2          2  W10005_0107  2016     260.0   61.0  111.882553
8572           WIH2          2         W37A  2016     265.0   59.0  108.685207
8573           WIH2          2           WF9  2016     287.0   67.0  116.310355

[8574 rows x 7 columns]
```

[25]:
```
column_name_17=[item for item in group_mean17.Female]
```

[26]:
```
same_line=[item for item in group_mean16.Female if item in column_name_17]
```

[27]:
```
len(same_line)
```

[27]:
```
7254
```

[28]:
```
ee=pd.DataFrame(same_line).drop_duplicates()
ee.shape
```

[28]:
```
(251, 1)
```

0.0.2 251 line tested in both year (2016 and 2017)

[29]:
```
group_mean16[['p_height','s_day','yield']].describe()
```

[29]:

	p_height	s_day	yield
count	7549.000000	7219.000000	7752.000000
mean	225.971172	70.025685	123.958167
std	40.191176	9.413134	42.549029
min	89.000000	45.000000	7.973638
25%	200.000000	62.000000	97.992097
50%	229.000000	70.000000	125.661141
75%	256.000000	77.000000	154.732651
max	345.000000	95.000000	288.884087

[93]:
```
single_line_mean16=group_mean16[['Year','p_height','s_day','yield','Female']].
↪groupby('Female').mean()
```

[94]:
```
single_line_mean16.head()
```

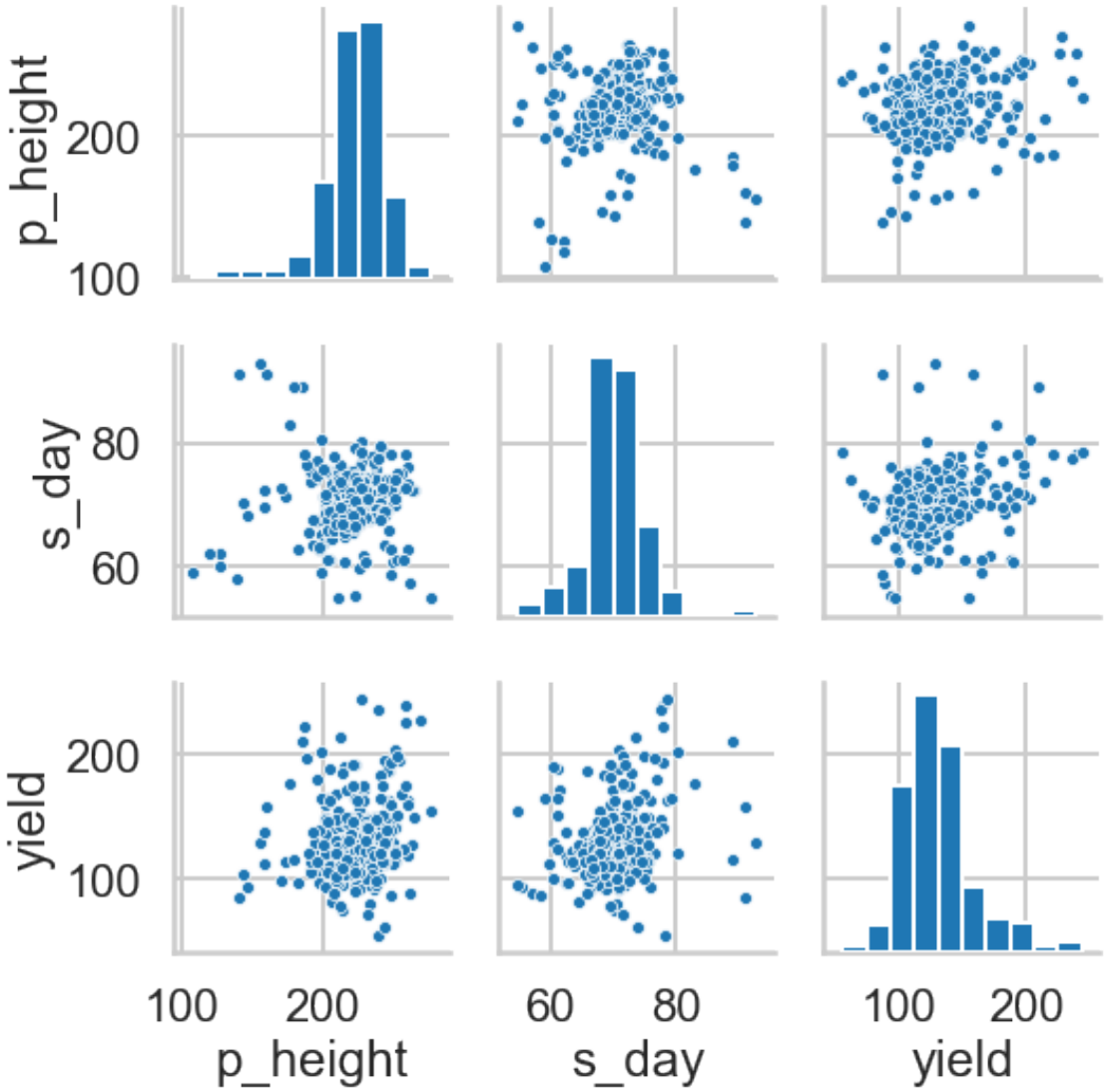
[94]:

Female	Year	p_height	s_day	yield
(CML442-B*CML343-B-B-B-B-B-B)-B-B-1-1-B-B-B-1-B...	2016	NaN	70.000000	155.633272
2369	2016	235.896259	70.851064	136.852867
2FACC	2016	223.774194	66.862069	139.054988
4N506	2016	240.935484	69.178571	132.508244
5618STXRIB	2016	243.500000	NaN	195.653993

[115]:
```
single_line_mean16.to_csv('single_line_mean16.csv')
```

[96]:
```
sns.pairplot(single_line_mean16[['p_height','s_day','yield']],markers='.')
```

[96]: <seaborn.axisgrid.PairGrid at 0x13936128>



1.3 Work with 2015 hybrid_data

[33]:
```
p15 =pd.read_csv("g2f_2015_hybrid_data_clean.csv",sep='\t')
```

[34]:
```
p15.head()
```

[34]:

	Year	Field-Location	RecId	Source	Pedigree	\
0	2015	DEH1	2662499	LOCAL_CHECK_2	DKC62-08 RIB GENSS	
1	2015	WIH2	2675715	WISN14/43969	LH82_PHG47-12/PHB47	
2	2015	TXH2	2674625	14SJWE:PHZ51:21032	GEMS-0142/PHZ51	
3	2015	DEH1	2662498	LOCAL_CHECK_2	DKC62-08 RIB GENSS	
4	2015	ONH1	2673236	13SAJL:NURSE:0045	F42/MO17	

```
   Replicate  Block  Plot  Range  Pass  ... Root Lodging [plants]  \
0          1     10   226     13    15  ...                   0.0
1          2      7   423     27    24  ...                   0.0
2          2      5   363     21    10  ...                   0.0
3          2      5   372     22    12  ...                   0.0
4          1      6   134      6    81  ...                   0.0

   Stalk Lodging [plants]  Grain Moisture [%]  Test Weight [lbs/bu]  \
0                     1.0                19.9                  52.9
1                     0.0                24.8                  72.0
2                     4.0                13.5                  58.0
3                     0.0                20.1                  52.9
4                     1.0                31.7                  56.8

   Plot Weight [lbs]  Grain Yield [bu/A]  \
0              30.71          294.079494
1              44.60          287.201940
2              43.13          286.191977
3              29.86          285.225922
4              43.80          282.444637

   Plot Discarded [enter "yes" or "blank"]  Comments  \
0                                      NaN       NaN
1                                      NaN       NaN
2                                      NaN       NaN
3                                      NaN       NaN
4                                      NaN       NaN

   Filler [enter "filler" or "blank"] [add additional measurements here]
0                                 NaN                                 NaN
1                                 NaN                                 NaN
2                                 NaN                                 NaN
3                                 NaN                                 NaN
4                                 NaN                                 NaN

[5 rows x 38 columns]
```

[35]:
```
p15c=p15[['Year','Field-Location','Pedigree','Replicate','Plant Height␣
↪[cm]','Silk DAP [days]','Grain Yield [bu/A]']]
```

[36]:
```
p15c.head()
```

[36]:
```
   Year Field-Location            Pedigree  Replicate  Plant Height [cm]  \
0  2015           DEH1   DKC62-08 RIB GENSS          1              267.0
1  2015           WIH2  LH82_PHG47-12/PHB47          2              237.0
2  2015           TXH2      GEMS-0142/PHZ51          2              258.0
3  2015           DEH1   DKC62-08 RIB GENSS          2              242.0
4  2015           ONH1             F42/MO17          1              240.0
```

```
   Silk DAP [days]  Grain Yield [bu/A]
0             62.0          294.079494
1              NaN          287.201940
2             77.0          286.191977
3             61.0          285.225922
4             87.0          282.444637
```

[37]:
```
## chage clean column name
p15c.rename(columns={'Plant Height [cm]':'p_height','Silk DAP [days]':
 ↪'s_day','Grain Yield [bu/A]':'yield'},inplace=True)
```

[38]:
```
## Extract Female ID for GCA caculation
p15c[['Female','Male']] = p15c.Pedigree.str.split("/",expand=True)
```

[39]:
```
p15c
```

[39]:
```
       Year Field-Location                                          Pedigree  \
0      2015           DEH1                                DKC62-08 RIB GENSS
1      2015           WIH2                              LH82_PHG47-12/PHB47
2      2015           TXH2                                  GEMS-0142/PHZ51
3      2015           DEH1                                DKC62-08 RIB GENSS
4      2015           ONH1                                         F42/MO17
...     ...            ...                                              ...
11978  2015           NYH2                                     CG105/CG102
11979  2015           NYH2                                      LH82/TX777
11980  2015           NYH2                                      CG60/PHB47
11981  2015           NYH2  (LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-...
11982  2015           NYH2                                     CG119/CG108

       Replicate  p_height  s_day       yield  \
0              1     267.0   62.0  294.079494
1              2     237.0    NaN  287.201940
2              2     258.0   77.0  286.191977
3              2     242.0   61.0  285.225922
4              1     240.0   87.0  282.444637
...          ...       ...    ...         ...
11978          2       NaN    NaN         NaN
11979          2       NaN    NaN         NaN
11980          2       NaN    NaN         NaN
11981          2       NaN    NaN         NaN
11982          2       NaN    NaN         NaN

                                      Female   Male
0                                DKC62-08 RIB GENSS   None
1                                     LH82_PHG47-12  PHB47
2                                         GEMS-0142  PHZ51
3                                DKC62-08 RIB GENSS   None
4                                              F42   MO17
```

```
...                                                          ...    ...
11978                                                      CG105  CG102
11979                                                       LH82  TX777
11980                                                       CG60  PHB47
11981  (LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-B16    LH195
11982                                                      CG119  CG108

[11983 rows x 9 columns]
```

[41]:
```
## Calculate single value of GCA for each line grouped by␣
 ↪Field-Location,Replicate

cols=['Field-Location','Replicate','Female']
group_mean15=p15c[['Year','Field-Location','Replicate','p_height','s_day','yield','Female']].
 ↪groupby(cols).mean()
```

[42]:
```
group_mean15.head()
```

[42]:
```
                                                                 Year
\
Field-Location Replicate Female
DEH1           1         (LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-B16  2015
                         2369                                               2015
                         A634                                               2015
                         B104                                               2015
                         B106                                               2015

p_height  \
Field-Location Replicate Female
DEH1           1         (LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-B16
228.0
                         2369
276.0
                         A634
250.0
                         B104
251.5
                         B106
266.5

s_day  \
Field-Location Replicate Female
DEH1           1         (LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-B16
71.0
                         2369
64.0
                         A634
64.0
```

```
                        B104
63.5
                        B106
62.0

 yield
Field-Location Replicate Female
DEH1           1         (LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-B16
246.982345
                        2369
222.500258
                        A634
192.531030
                        B104
171.507667
                        B106
197.816370
```

[44]:
```
group_mean15.reset_index().to_csv('p15_clean.csv',index=False)
```

[45]:
```
group_mean15=group_mean.reset_index()
group_mean15
```

[45]:
```
     Field-Location  Replicate  \
0              DEH1          1
1              DEH1          1
2              DEH1          1
3              DEH1          1
4              DEH1          1
...             ...        ...
8645           WIH2          2
8646           WIH2          2
8647           WIH2          2
8648           WIH2          2
8649           WIH2          2

                                                    Female  Year  p_height  \
0     (LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-B16  2015     228.0
1                                                   2369  2015     276.0
2                                                   A634  2015     250.0
3                                                   B104  2015     251.5
4                                                   B106  2015     266.5
...                                                  ...   ...       ...
8645                                           Z022E0073  2015     262.0
8646                                           Z022E0105  2015     273.0
8647                                           Z022E0123  2015     258.0
8648                                           Z022E0142  2015     223.0
8649                                           Z022E0149  2015     253.0
```

```
      s_day       yield
0      71.0  246.982345
1      64.0  222.500258
2      64.0  192.531030
3      63.5  171.507667
4      62.0  197.816370
...     ...         ...
8645    NaN  159.768239
8646    NaN  192.630268
8647    NaN  142.021921
8648    NaN  171.500641
8649    NaN  230.049617

[8650 rows x 7 columns]
```

[47]:
```
group_mean15[['p_height','s_day','yield']].describe()
```

[47]:
```
          p_height        s_day        yield
count  7500.000000  6500.000000  8579.000000
mean    223.972514    73.093337   133.815198
std      34.789771     8.924943    44.003825
min     116.000000    47.000000    10.801014
25%     202.000000    66.000000   104.151031
50%     229.083333    72.000000   134.013643
75%     250.000000    78.500000   163.788596
max     322.000000   102.000000   294.079494
```

[97]:
```
single_line_mean15=group_mean15[['Year','p_height','s_day','yield','Female']].
↪groupby('Female').mean()
```

[98]:
```
single_line_mean15.head()
```

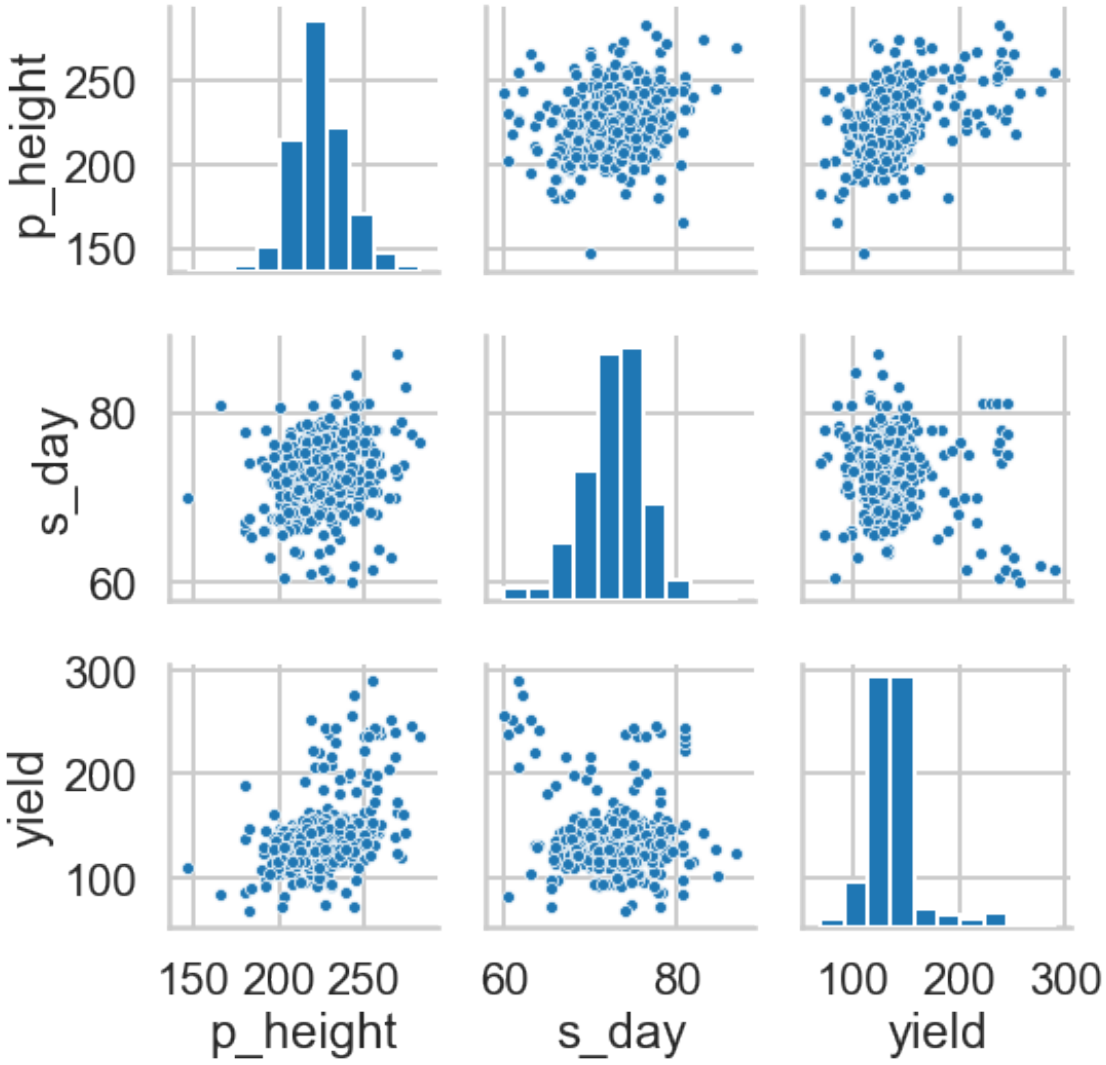
[98]:
```
                                                    Year    p_height  \
Female
(LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-B16  2015  219.658537
(TX739);LAMA2002-10-1-B-B-B-B3-B7ORANGE-B6         2015  216.375000
2369                                               2015  251.380952
31G66                                              2015  240.000000
31G71                                              2015  260.000000

                                                     s_day       yield
Female
(LAMA2002-35-2-B-B-B-B_CG44)-1-3-B-1-1-B24-B5-B16  80.764706  124.817108
(TX739);LAMA2002-10-1-B-B-B-B3-B7ORANGE-B6         77.875000  142.805183
2369                                               75.323529  149.288236
31G66                                                    NaN  196.016164
31G71                                                    NaN  238.653322
```

[114]:
```
single_line_mean15.to_csv('single_line_mean15.csv')
```

[100]:
```
sns.pairplot(single_line_mean15[['p_height','s_day','yield']],markers='.')
```

[100]:
```
<seaborn.axisgrid.PairGrid at 0x13ed6dd8>
```



1.4 Work with 2014 Hybrid_data

[57]:
```
p14 =pd.read_csv("g2f_2014_hybrid_data_clean.csv")
```

[58]:
```
p14.head()
```

[58]:

	Year	Field-Location	RecId	Source	\
0	2014	TXH1	2218825	LOCAL_CHECK	
1	2014	MNH1	2235804	13WJWE:CG102:1227	
2	2014	TXH1	2218560	WE13-80ISO-227-X-POL-80	
3	2014	TXH1	2218682	13SAJL:NURSE:0145	
4	2014	TXH1	2218600	WE13-195ISO-149-X-POL-195	

```
                     Pedigree  Replicate  Block  Plot  Range  Pass  ...  \
0                  DEKALB 64-69          2      5   113   14.0  25.0  ...
1         PHN11_LH145_0002/CG102          2      1    42    9.0  21.0  ...
2  MOG_NC230-043-1-1-1-1-B/PB80          1      5    98    6.0  30.0  ...
3                   PHG39/PHN82          2      3    50   12.0  32.0  ...
4                   TX303/LH195          1     10   258   11.0  25.0  ...

  Root Lodging [plants]  Stalk Lodging [plants]  Grain Moisture [%]  \
0                   NaN                    NaN                11.8
1                   0.0                    0.0                30.5
2                   NaN                    NaN                11.7
3                   NaN                    NaN                12.4
4                   NaN                    NaN                12.9

   Test Weight [lbs/bu]  Plot Weight [lbs]  Grain Yield [bu/A]  \
0                  58.9              32.54          251.616994
1                  52.5              30.51          177.450596
2                  60.3              16.28          126.028550
3                  57.5              24.63          189.156972
4                  57.6              26.91          205.487632

   Plot Discarded [enter "yes" or "blank"]  Comments  \
0                                      NaN       NaN
1                                      NaN       NaN
2                                      NaN       NaN
3                                      NaN       NaN
4                                      NaN       NaN

   Filler [enter "filler" or "blank"] [add additional measurements here]
0                                 NaN                                 NaN
1                                 NaN                                 NaN
2                                 NaN                                 NaN
3                                 NaN                                 NaN
4                                 NaN                                 NaN

[5 rows x 38 columns]
```

[59]:
```
## Rename columns with clean titles
p14c=p14[['Year','Field-Location','Pedigree','Replicate','Plant Height␣
 ↪[cm]','Silk DAP [days]','Grain Yield [bu/A]']]
```

[60]:
```
p14c.head()
```

[60]:
```
   Year Field-Location                     Pedigree  Replicate  \
0  2014           TXH1                  DEKALB 64-69          2
1  2014           MNH1         PHN11_LH145_0002/CG102          2
2  2014           TXH1  MOG_NC230-043-1-1-1-1-B/PB80          1
```

```
3  2014           TXH1                   PHG39/PHN82          2
4  2014           TXH1                   TX303/LH195          1

   Plant Height [cm]  Silk DAP [days]  Grain Yield [bu/A]
0              193.0             82.0          251.616994
1              190.0             68.0          177.450596
2              211.0             85.0          126.028550
3              196.0             82.0          189.156972
4              213.0             88.0          205.487632
```

[61]:
```
## chage clean column name
p14c.rename(columns={'Plant Height [cm]':'p_height','Silk DAP [days]':
 ↪'s_day','Grain Yield [bu/A]':'yield'},inplace=True)
```

[62]:
```
## Extract Female ID for GCA caculation
p14c[['Female','Male']] = p14c.Pedigree.str.split("/",expand=True)
```

[63]:
```
p14c
```

[63]:
```
       Year Field-Location                    Pedigree  Replicate  p_height  \
0      2014           TXH1                  DEKALB 64-69          2     193.0
1      2014           MNH1       PHN11_LH145_0002/CG102          2     190.0
2      2014           TXH1  MOG_NC230-043-1-1-1-1-B/PB80          1     211.0
3      2014           TXH1                   PHG39/PHN82          2     196.0
4      2014           TXH1                   TX303/LH195          1     213.0
...     ...            ...                           ...        ...       ...
12670  2014           ONH1                   M0087/CG102          2       NaN
12671  2014           ONH1                   M0027/CG102          2       NaN
12672  2014           ONH1                   M0262/CG102          2       NaN
12673  2014           ONH1                   CG108/CG102          2       NaN
12674  2014           ONH1                   CG105/CG102          2       NaN

       s_day       yield                  Female   Male
0       82.0  251.616994             DEKALB 64-69   None
1       68.0  177.450596         PHN11_LH145_0002  CG102
2       85.0  126.028550  MOG_NC230-043-1-1-1-1-B   PB80
3       82.0  189.156972                   PHG39  PHN82
4       88.0  205.487632                   TX303  LH195
...      ...         ...                     ...    ...
12670    NaN         NaN                  M0087  CG102
12671    NaN         NaN                  M0027  CG102
12672    NaN         NaN                  M0262  CG102
12673    NaN         NaN                  CG108  CG102
12674    NaN         NaN                  CG105  CG102

[12675 rows x 9 columns]
```

[65]:
```
## Calculate single value of GCA for each line grouped by␣
 ↪Field-Location,Replicate
```

```
cols=['Field-Location','Replicate','Female']
group_mean14=p14c[['Year','Field-Location','Replicate','p_height','s_day','yield','Female']].
 ↪groupby(cols).mean()
```

[71]:
```
group_mean14.head()
```

[71]:
```
                                                                 Year
\
Field-Location Replicate Female
DEH1           1         ARGNETINE FLINTY COMPOSITE-C(1)-37-B-B-B2-1-B25  2014
                         B37                                             2014
                         B73                                             2014
                         B73_NC230-041-1-1-1-1                           2014
                         B73_NC230-126-1-1-1-1                           2014

p_height  \
Field-Location Replicate Female
DEH1           1         ARGNETINE FLINTY COMPOSITE-C(1)-37-B-B-B2-1-B25
234.0
                         B37
258.5
                         B73
263.4
                         B73_NC230-041-1-1-1-1
227.0
                         B73_NC230-126-1-1-1-1
203.0

                                                                 s_day
\
Field-Location Replicate Female
DEH1           1         ARGNETINE FLINTY COMPOSITE-C(1)-37-B-B-B2-1-B25   70.0
                         B37                                              67.0
                         B73                                              67.2
                         B73_NC230-041-1-1-1-1                            70.0
                         B73_NC230-126-1-1-1-1                            70.0

yield
Field-Location Replicate Female
DEH1           1         ARGNETINE FLINTY COMPOSITE-C(1)-37-B-B-B2-1-B25
203.275391
                         B37
216.338835
                         B73
188.919799
                         B73_NC230-041-1-1-1-1
104.638647
                         B73_NC230-126-1-1-1-1
```

```
136.478859
```

[68]:
```
group_mean14.reset_index().to_csv('p14_clean.csv',index=False)
```

[107]:
```
group_14_n=group_mean14.reset_index()
```

[109]:
```
group_14_n[['p_height','s_day','yield']].describe()
```

[109]:
```
          p_height        s_day        yield
count  8516.000000  6316.000000  8467.000000
mean    215.828684    71.245310   145.316322
std      27.997686     7.799739    43.986529
min     102.000000    50.000000     9.217824
25%     196.000000    66.000000   117.348312
50%     217.000000    70.000000   149.567335
75%     235.500000    77.000000   176.347208
max     351.000000    95.000000   344.631088
```

[110]:
```
single_line_mean14=group_14_n[['Year','p_height','s_day','yield','Female']].
↪groupby('Female').mean()
```

[111]:
```
single_line_mean14.head()
```

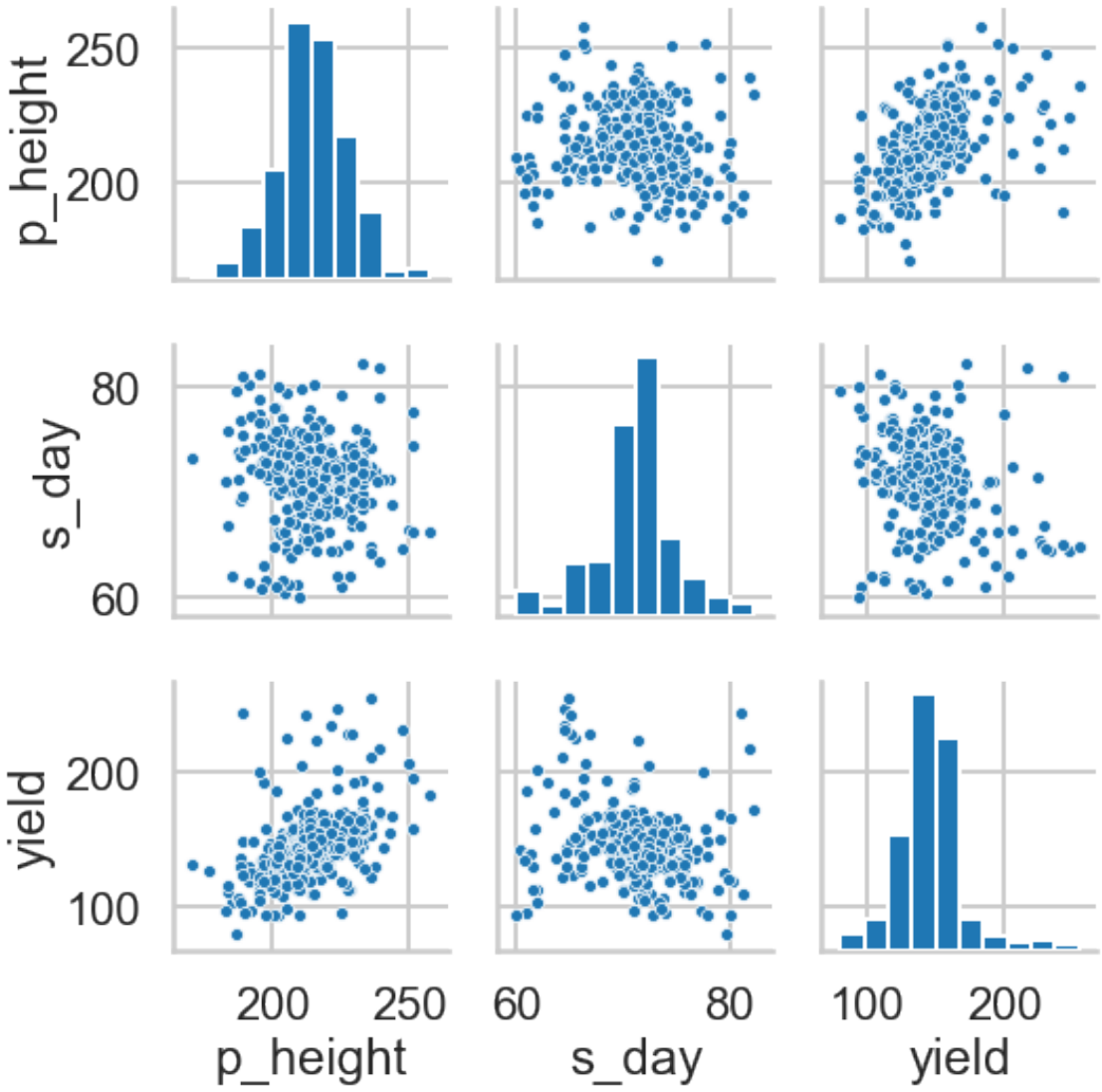
[111]:
```
                                                  Year    p_height      s_day  \
Female
(TX739);LAMA2002-10-1-B-B-B-B3-B7ORANGE-B6       2014  232.750000  82.333333
31G66                                            2014  217.400000        NaN
ARGNETINE FLINTY COMPOSITE-C(1)-37-B-B-B2-1-B25  2014  239.000000  79.000000
B14A                                             2014  231.390625  72.230769
B37                                              2014  233.182765  73.394608

                                                      yield
Female
(TX739);LAMA2002-10-1-B-B-B-B3-B7ORANGE-B6       171.651641
31G66                                            154.248454
ARGNETINE FLINTY COMPOSITE-C(1)-37-B-B-B2-1-B25  167.995240
B14A                                             132.563094
B37                                              147.925736
```

[112]:
```
single_line_mean14.to_csv('single_line_mean14.csv')
```

[113]:
```
sns.pairplot(single_line_mean14[['p_height','s_day','yield']],markers='.')
```

[113]:
```
<seaborn.axisgrid.PairGrid at 0x14461f98>
```



1. 5 Combine clean data for Best linear unbiased prediction (BLUP)

Data of 2014 and 2015 were put together to perform BLUP beacaue there are enough overlapping (252) of tested accessions

Data of 2016 and 2017 were put together to perform BLUP beacaue there are enough overlapping (251) of tested accessions

[3]:
```
dfb14=pd.read_csv('p14_clean.csv')
dfb15=pd.read_csv('p15_clean.csv')
dfb16=pd.read_csv('p16_clean.csv')
dfb17=pd.read_csv('p17_clean.csv')
```

[4]:
```
df_14_15=dfb14.append(dfb15,ignore_index=True)
```

[5]:
```
df_14_15.shape
```

[5]:
```
(17405, 7)
```

[6]:
```
## To find out outliers using 1.5 IQR
cols = ["p_height","s_day","yield"]
df_14_15[cols]
## creat a for loop to add IQR juddgement for each trait
for col in cols:
    col_IQR = col + '_1.5IQR'
    Q1 = df_14_15[col].quantile(0.25)
    Q3 = df_14_15[col].quantile(0.75)
    IQR = Q3 - Q1
    df_14_15[col_IQR] =(df_14_15[col]<(Q1 - 1.5 * IQR)) | (df_14_15[col] > (Q3␣
 ↪+ 1.5 * IQR))

df_14_15.head()
```

[6]:

	Field-Location	Replicate	Female	Year	p_height	s_day	yield	p_height_1.5IQR	s_day_1.5IQR	yield_1.5IQR
0	DEH1	1	ARGNETINE FLINTY COMPOSITE-C(1)-37-B-B-B2-1-B25	2014	234.0	70.0	203.275391	False	False	False
1	DEH1	1	B37	2014	258.5	67.0	216.338835	False	False	False
2	DEH1	1	B73	2014	263.4	67.2	188.919799	False	False	False
3	DEH1	1	B73_NC230-041-1-1-1-1	2014	227.0	70.0	104.638647	False	False	False
4	DEH1	1	B73_NC230-126-1-1-1-1	2014	203.0	70.0	136.478859	False	False	False

[7]:
```
from collections import Counter
```

[8]:
```
Counter(df_14_15['yield_1.5IQR'])
```

[8]:
```
Counter({False: 17335, True: 70})
```

[9]:
```
Counter(df_14_15['s_day_1.5IQR'])
```

[9]:
```
Counter({False: 17367, True: 38})
```

[10]:
```
Counter(df_14_15['p_height_1.5IQR'])
```

[10]:
```
Counter({False: 17341, True: 64})
```

[11]:
```
## Replace outliers as np.nan for height
for i in range(df_14_15.shape[0]):
    if df_14_15['p_height_1.5IQR'][i]==True:
       df_14_15['p_height'][i]=np.nan
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

[12]:
```
## Replace outliers as np.nan for s_day
for i in range(df_14_15.shape[0]):
    if df_14_15['s_day_1.5IQR'][i]==True:
       df_14_15['s_day'][i]=np.nan
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

[13]:
```
## Replace outliers as np.nan for yield
for i in range(df_14_15.shape[0]):
    if df_14_15['yield_1.5IQR'][i]==True:
       df_14_15['yield'][i]=np.nan
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

[15]:
```
df_14_15.to_csv('14_15_for_blup.csv')
```

[14]:
```
df_16_17=dfb16.append(dfb17,ignore_index=True)
```

[19]:
```
df_16_17.shape
```

[19]:
```
(17992, 7)
```

[20]:
```
## To find out outliers using 1.5 IQR
cols = ["p_height","s_day","yield"]
df_16_17[cols]
## creat a for loop to add IQR juddgement for each trait
for col in cols:
    col_IQR = col + '_1.5IQR'
    Q1 = df_16_17[col].quantile(0.25)
    Q3 = df_16_17[col].quantile(0.75)
    IQR = Q3 - Q1
    df_16_17[col_IQR] =(df_16_17[col]<(Q1 - 1.5 * IQR)) | (df_16_17[col] > (Q3␣
 ↪+ 1.5 * IQR))

df_16_17.head()
```

[20]:

	Field-Location	Replicate	Female	Year	p_height	s_day	yield	\
0	ARH1	1	2369	2016	187.0	70.0	119.950639	
1	ARH1	1	2FACC	2016	198.0	67.0	112.655439	
2	ARH1	1	4N506	2016	195.0	69.0	90.671176	
3	ARH1	1	6F629	2016	192.0	67.0	118.191476	
4	ARH1	1	A632	2016	203.0	69.0	90.888617	

	p_height_1.5IQR	s_day_1.5IQR	yield_1.5IQR
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

[21]:
```
Counter(df_16_17['yield_1.5IQR'])
```

[21]:
```
Counter({False: 17935, True: 57})
```

[22]:
```
Counter(df_16_17['s_day_1.5IQR'])
```

[22]:
```
Counter({False: 17882, True: 110})
```

[23]:
```
Counter(df_16_17['p_height_1.5IQR'])
```

[23]:
```
Counter({False: 16890, True: 1102})
```

[24]:
```
## Replace outliers as np.nan for height
for i in range(df_16_17.shape[0]):
    if df_16_17['p_height_1.5IQR'][i]==True:
       df_16_17['p_height'][i]=np.nan
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  after removing the cwd from sys.path.
```

[25]:
```
## Replace outliers as np.nan for s_day
for i in range(df_16_17.shape[0]):
    if df_16_17['s_day_1.5IQR'][i]==True:
       df_16_17['s_day'][i]=np.nan
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

[26]:
```
## Replace outliers as np.nan for yield
for i in range(df_16_17.shape[0]):
    if df_16_17['yield_1.5IQR'][i]==True:
       df_16_17['yield'][i]=np.nan
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

[27]:
```
df_16_17.to_csv('16_17_for_blup.csv')
```

0.0.3 The following BLUP calculation was done by R

R code for Blup analysis (p_height as an example): qualdat = read.csv("16_17_for_blup.csv", header=T)

##Check to ensure data imported correctly str(qualdat) head(qualdat) tail(qualdat)

##Attach dataset attach(qualdat)

##Examine distribution of brix data hist(p_height, col="gold") boxplot(p_height~Field.Location, xlab="Field.Location", ylab="Height", main="Degrees height by Location", col="pink")

#Rename variables for ease of use HEIGHT= as.numeric(p_height) LINE = as.factor(Female) LOC = as.factor(Field.Location) YEAR = as.factor(Year) REP = as.factor(Replicate)

##Calculate variance components #requires lme4 package library(lme4)

#Linear Model with random effects for variance components brixvarcomp = lmer(HEIGHT~ (1|LINE) + (1|LOC) + (1|YEAR) + (1|LINE:LOC) + (1|LINE:YEAR))
#Extract variance components summary(brixvarcomp)

##BLUPS #fit the model brixmodel = lmer(HEIGHT~ (1|LINE) + (1|LOC) + (1|YEAR) + (1|LINE:LOC) + (1|LINE:YEAR))

#estimate BLUPS brixblup = ranef(brixmodel) #look at output structure str(brixblup) #extract blup for line brixlineblup = brixblup$LINE #see the structure of the blup for each line str(brixlineblup) #save the brixlineblup output to a separate .csv file write.csv(brixlineblup, file="p_height_LineBLUPS.csv")

##Creating plots with the BLUPs #Create a numeric vector with the BLUP for each line LINEBLUP = brixlineblup[,1] #Create a histogram with the BLUP for each line hist(LINEBLUP, col="brown")

##Compare BLUP to line averages on a scatterplot lmean = tapply(HEIGHT, LINE, na.rm=T, mean) plot(LINEBLUP, lmean, col="blue")

[28]:
```
## for data blup from 2014 and 2015
y45=pd.read_csv('yblup45.csv')
s45=pd.read_csv('sblup45.csv')
h45=pd.read_csv('hblup45.csv')
```

[30]:
```
## for data blup from 2016 and 2017
y67=pd.read_csv('yblup67.csv')
s67=pd.read_csv('sblup67.csv')
h67=pd.read_csv('hblup67.csv')
```

[31]:
```
## combine all blup results from 2014 to 2017
y_blup=y45.append(y67,ignore_index=True)
```

[32]:
```
## export y_blup results
y_blup.groupby('id').mean().to_csv("y_blup_total.csv")
```

[33]:
```
## for s_day
```

[34]:
```
## combine all blup results from 2014 to 2017
s_blup=s45.append(s67,ignore_index=True)
```

[35]:
```
## export y_blup results
s_blup.groupby('id').mean().to_csv("s_blup_total.csv")
```

[36]:
```
## for p_height
```

[37]:
```
## combine all blup results from 2014 to 2017
h_blup=h45.append(h67,ignore_index=True)
```

[38]:
```
## export y_blup results
h_blup.groupby('id').mean().to_csv("h_blup_total.csv")
```

1. 6 Combine all mean value for GCB of single line across all year and all location

[47]:
```
Total_mean=df_14_15.append(df_16_17,ignore_index=True)
```

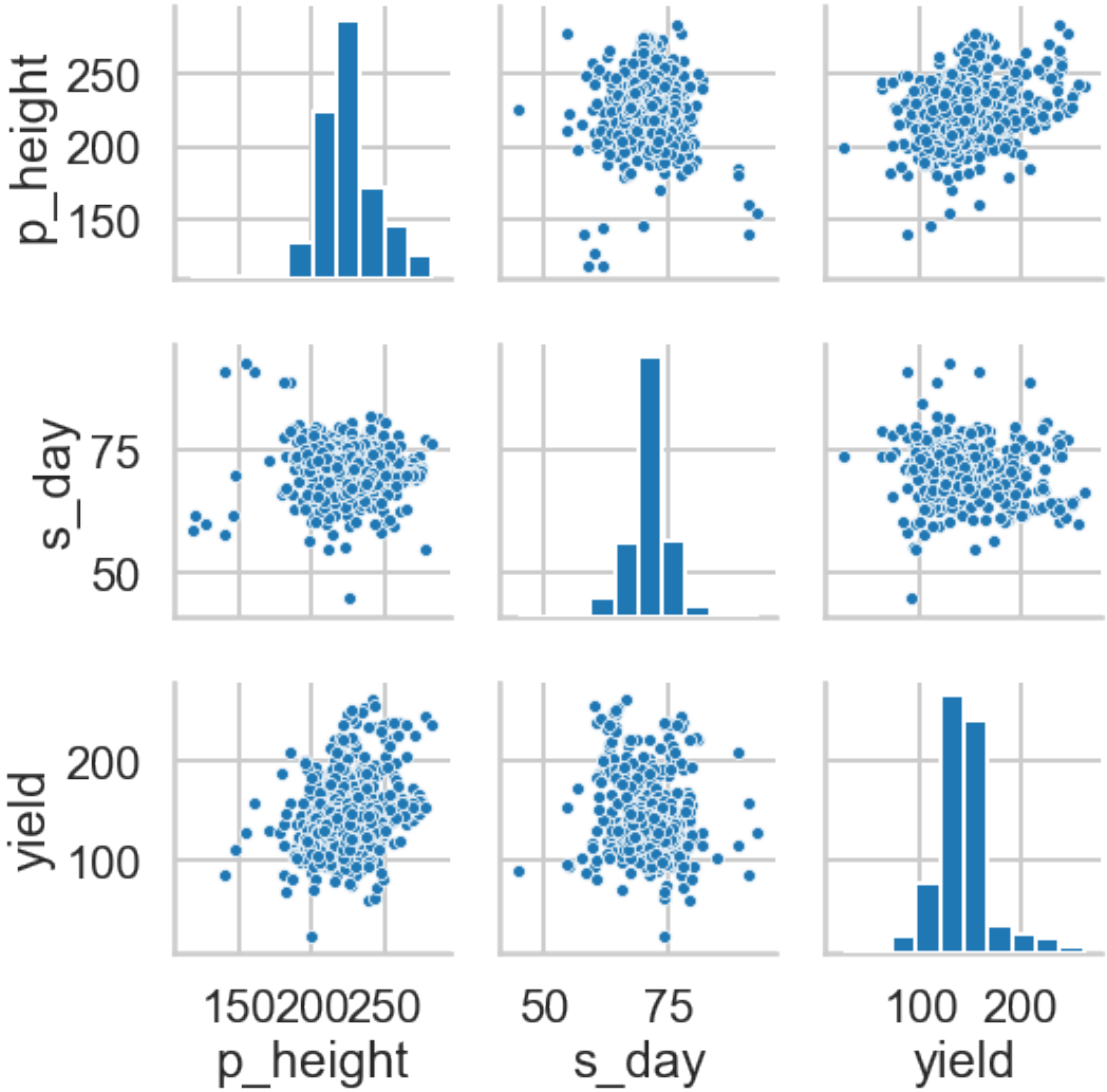
[48]:
```
Total_mean.shape
```

[48]:
```
(35397, 10)
```

[49]:
```
Total_mean=Total_mean[['p_height','s_day','yield','Female']].groupby('Female').
↪mean()
```

[53]:
```
sns.pairplot(Total_mean,markers='.')
```

[53]: <seaborn.axisgrid.PairGrid at 0x56e6208>



0.0.4 2 Genotype data cleaning

2.1 Clean and unify genotype ID Original h5 data," g2f_2017_ZeaGBSv27_Imputed_AGPv4.h5", was converted to hapmap file with following TASSEL commond line:

../TASSEL5/run_pipeline.pl -Xmx20G -h5 g2f_2017_ZeaGBSv27_Imputed_AGPv4.h5 -export g2f_h5.txt -exportType Hapmap

[153]:
```
df2=pd.read_csv('g2f_h5.txt',sep='\t')
```

[206]:
```
df2.iloc[:20,:20]
```

[206]:

	rs#	alleles	chrom	pos	strand	assembly#	center	protLSID	\
0	S1_10045	G/C	1	56073	+	NaN	NaN	NaN	

```
1    S1_10097     C/G      1    56125      +        NaN     NaN       NaN
2   S1_157465     T/C      1   210964      +        NaN     NaN       NaN
3   S1_222471     C/T      1   262738      +        NaN     NaN       NaN
4   S1_222473     T/C      1   262740      +        NaN     NaN       NaN
5   S1_222541     A/T      1   262808      +        NaN     NaN       NaN
6   S1_222588     T/C      1   262855      +        NaN     NaN       NaN
7   S1_222877     A/G      1   263144      +        NaN     NaN       NaN
8   S1_267769     G/A      1   334943      +        NaN     NaN       NaN
9   S1_264849     A/G      1   337863      +        NaN     NaN       NaN
10  S1_239225     C/G      1   369777      +        NaN     NaN       NaN
11  S1_237590     C/T      1   371412      +        NaN     NaN       NaN
12  S1_237571     C/G      1   371431      +        NaN     NaN       NaN
13  S1_515883     A/G      1   560842      +        NaN     NaN       NaN
14  S1_525692     T/A      1   572352      +        NaN     NaN       NaN
15  S1_644637     T/A      1   688873      +        NaN     NaN       NaN
16  S1_768337     C/T      1   798230      +        NaN     NaN       NaN
17  S1_982399     C/T      1  1012701      +        NaN     NaN       NaN
18  S1_982989     C/T      1  1013291      +        NaN     NaN       NaN
19  S1_985248     G/T      1  1015550      +        NaN     NaN       NaN

    assayLSID  panelLSID  QCcode BLANK:100000001 BLANK:100000002  \
0         NaN        NaN     NaN               N               G
1         NaN        NaN     NaN               N               C
2         NaN        NaN     NaN               N               T
3         NaN        NaN     NaN               N               C
4         NaN        NaN     NaN               N               T
5         NaN        NaN     NaN               N               A
6         NaN        NaN     NaN               N               T
7         NaN        NaN     NaN               N               A
8         NaN        NaN     NaN               N               N
9         NaN        NaN     NaN               N               N
10        NaN        NaN     NaN               N               G
11        NaN        NaN     NaN               N               C
12        NaN        NaN     NaN               N               C
13        NaN        NaN     NaN               N               N
14        NaN        NaN     NaN               N               T
15        NaN        NaN     NaN               N               T
16        NaN        NaN     NaN               N               N
17        NaN        NaN     NaN               N               C
18        NaN        NaN     NaN               N               C
19        NaN        NaN     NaN               N               T

   BLANK:100000003 PHN11_Oh43_0075:100000004 W10004_0248:100000005  \
0                N                         G                      G
1                N                         C                      C
2                N                         Y                      N
3                N                         C                      N
```

4	N	T	N
5	N	A	A
6	N	T	T
7	N	A	N
8	N	G	A
9	N	G	A
10	N	N	N
11	N	C	C
12	N	S	C
13	N	N	A
14	N	T	N
15	N	T	T
16	N	T	C
17	N	C	N
18	N	C	N
19	N	T	N

	AS6103:100000006	PHN11_LH145_0029:100000007	W10005_0107:100000008	\
0	G	G	G	
1	C	C	C	
2	T	T	T	
3	N	C	C	
4	N	T	T	
5	A	A	A	
6	T	T	T	
7	A	A	A	
8	A	A	A	
9	A	A	A	
10	C	C	C	
11	N	C	C	
12	N	C	C	
13	N	G	N	
14	T	T	T	
15	N	A	T	
16	C	C	C	
17	C	C	T	
18	C	C	C	
19	G	G	G	

	W10005_0032:100000009
0	G
1	C
2	T
3	C
4	T
5	A
6	T

```
7                     A
8                     N
9                     N
10                    C
11                    C
12                    C
13                    N
14                    T
15                    N
16                    C
17                    N
18                    C
19                    G
```

[160]:
```
df3=pd.concat([df2.iloc[:,0], df2.iloc[:,14:]],axis=1)
```

[161]:
```
id=[]
for item in df3.columns:
    id.append(item)
```

[162]:
```
id[0]="ID"
```

[163]:
```
ID_df=pd.DataFrame(id)
```

[164]:
```
ID_df.columns=['i']
```

[165]:
```
ID_df.head()
```

[165]:
```
                          i
0                        ID
1   PHN11_Oh43_0075:100000004
2       W10004_0248:100000005
3            AS6103:100000006
4  PHN11_LH145_0029:100000007
```

[166]:
```
ID_df[['id','num']] = ID_df.i.str.split(":",expand=True)
```

[167]:
```
df3.columns=ID_df['id']
```

[168]:
```
df3.head()
```

[168]:
```
id         ID PHN11_Oh43_0075 W10004_0248 AS6103 PHN11_LH145_0029 W10005_0107  \
0    S1_10045               G           G      G                G           G
1    S1_10097               C           C      C                C           C
2   S1_157465               Y           N      T                T           T
3   S1_222471               C           N      N                C           C
4   S1_222473               T           N      N                T           T

id W10005_0032 W10004_0082 PHN11_LH145_0028 W10005_0029  ... C103 Z013E0080  \
0            G           N                G           G  ...    C         G
1            C           N                C           C  ...    C         C
2            T           N                T           T  ...    T         C
```

```
3           C          N               C          N  ...    C        C
4           T          N               T          N  ...    C        T

id Z022E0009 Z022E0048 AH83 PI538011 Ames27138 Va35 Tx303 PI601773
0          G         G    G        G         G    G     G        G
1          C         C    C        C         C    C     C        C
2          T         T    T        N         T    T     T        T
3          C         C    C        N         C    C     T        C
4          T         T    T        N         T    T     T        T

[5 rows x 1575 columns]
```

2.2 SNP data cleaning based on MAF and missing rate

[170]:
```
df3.replace(to_replace='-', value=np.nan, inplace=True)
df3.replace(to_replace='N', value=np.nan, inplace=True)
df3.replace(to_replace='W', value='H', inplace=True)
df3.replace(to_replace='S', value='H', inplace=True)
df3.replace(to_replace='M', value='H', inplace=True)
df3.replace(to_replace='K', value='H', inplace=True)
df3.replace(to_replace='R', value='H', inplace=True)
df3.replace(to_replace='Y', value='H', inplace=True)
```

[173]:
```
cols = list(df3.columns)
```

[174]:
```
## Since inbred line is selfing , it is important to find out accessions with␣
 ↪high Heterozygous(H) genotype
Column_Total=[]
for col in cols:
    count_h=0
    for item in df3[col]:
        if item=='H':
         count_h+=1
    Column_Total.append(count_h/df3.shape[0])
df3.loc['Column_H_Total']=Column_Total
```

[175]:
```
Column_missing=[]
for col in cols:
    count_m=0
    for item in df3[col]:
        if str(item)=='nan':
         count_m+=1
    Column_missing.append(count_m/df3.shape[0])
df3.loc['Column_Missing_rate']=Column_missing
```

[176]:
```
df3.tail()
```

[176]:
```
id                            ID PHN11_Oh43_0075 W10004_0248      AS6103  \
303508  S10_150088202                       G           G           G
303509  S10_150107228                       A           A           A
```

```
303510               S10_150107267               A           A           A
Column_H_Total                   0        0.127584    0.123528  0.00121577
Column_Missing_rate              0         0.31666    0.337881   0.0843196

id                   PHN11_LH145_0029 W10005_0107 W10005_0032 W10004_0082  \
303508                              A           G           G           G
303509                              G           A           A           A
303510                              A           A           A           A
Column_H_Total            0.00152548   0.0302394   0.0012619   0.0928895
Column_Missing_rate        0.0988857    0.117564    0.100414    0.296344

id                   PHN11_LH145_0028 W10005_0029  ...        C103  Z013E0080  \
303508                              A           G  ...         NaN        NaN
303509                              G           A  ...           A          A
303510                              A           A  ...           A          A
Column_H_Total            0.00109386  0.00123554  ...  0.00545285  0.0136371
Column_Missing_rate         0.100793    0.107788  ...    0.145154  0.0890113

id                    Z022E0009  Z022E0048        AH83    PI538011   Ames27138  \
303508                        G          G           G           A         NaN
303509                        A          A           A           G           A
303510                        A          A           A           A           A
Column_H_Total       0.0214753   0.068752  0.00142993  0.00250403  0.00263582
Column_Missing_rate  0.0857759  0.0762046     0.13926    0.114081    0.140877

id                         Va35 Tx303   PI601773
303508                        A     G          A
303509                        G     A          G
303510                        A     A          A
Column_H_Total       0.00116635     0  0.0111858
Column_Missing_rate    0.137457     0   0.218275

[5 rows x 1575 columns]
```

[177]:
```
# Construct a Boolean Series to identify outliers for genotypic data based on␣
 ↪missing rate
outliers_m_def= df3.loc['Column_Missing_rate']<0.35
```

[178]:
```
df3_new=df3.loc[:,outliers_m_def]
```

[179]:
```
# Construct a Boolean Series to identify outliers for genotypic data based on␣
 ↪Hetozog rate
outliers_h_def=df3.loc['Column_H_Total']<0.15
```

[180]:
```
# Construct a Boolean Series to identify outliers for genotypic data based on␣
 ↪Hetozog rate
outliers_h_def=df3.loc['Column_H_Total']<0.10
```

[181]:
```
df3_new=df3.loc[:,outliers_h_def]
```

[264]:
```
df3_new.head().to_csv('clean_g_name.csv')
```

[189]:
```
test_last=df3_new.iloc[:,1:]
```

[190]:
```
## Minor Allele Frequency (MAF) refers to the frequency at which the least␣
 ↪common allele occurs in a
## given population. Low MAF SNPs tend to have poorly behaved test␣
 ↪statistics(violation of large sample assumption).
## The following function will help me finding out snps with low MAF and add a␣
 ↪new column(maf) to the data set.

def MAF(snps):
    print()
    counter = 0

    # Set up some constants for coding
    major = '0'
    minor = '2'
    hetero = '1'
    maf=[]
    for i in range(snps.shape[0]):
        # Progress updates
        counter += 1
        if counter % 1e2 == 0:
            print("Processed", counter, "of", snps.shape[0] - 1, "markers.",␣
 ↪end = '\r')

        # Get allele counts
        allele_counts = snps.iloc[i, :].value_counts()
        major_allele, minor_allele, het_allele = "N", "N", "N"
        major_count, minor_count, het_count = 0, 0, 0

        # Identify major and minor alleles
        for k, v in allele_counts.iteritems():
            if k in ['W', 'S', 'M', 'K', 'R', 'Y','H','0']:
                het_allele, het_count = k, v
            elif v > major_count:
                minor_count, major_count = major_count, v
                minor_allele, major_allele = major_allele, k
            else:
                minor_allele, minor_count = k, v
        MAF_r=(minor_count/snps.shape[1] + (het_count/snps.shape[1])*0.5)
        maf.append(MAF_r)
    snps.loc[:,'maf'] = maf
```

[191]:
```
MAF(test_last)
```

```
Processed 303500 of 303512 markers.arkers. markers.of 303512 markers. 9900 of
```

```
303512 markers.of 303512 markers.of 303512 markers. 303512 markers. 20800 of
303512 markers.markers. 26200 of 303512 markers.of 303512 markers. 28200 of
303512 markers.of 303512 markers.markers.of 303512 markers.of 303512 markers.of
303512 markers.of 303512 markers. 303512 markers. markers. 303512 markers. 72100
of 303512 markers. 303512 markers. markers.markers.of 303512 markers. 89700 of
303512 markers. of 303512 markers.101300 of 303512 markers.of 303512 markers.
116600 of 303512 markers. 303512 markers. 303512 markers. 139300 of 303512
markers.markers.of 303512 markers.of 303512 markers.markers.of 303512 markers.
of 303512 markers.of 303512 markers. 165700 of 303512 markers. 167600 of 303512
markers.170100 of 303512 markers. 171200 of 303512 markers. 177900 of 303512
markers. 179300 of 303512 markers.of 303512 markers. 181700 of 303512 markers.
markers. 192500 of 303512 markers. 303512 markers. 205100 of 303512 markers. of
303512 markers.markers. 214000 of 303512 markers.215400 of 303512 markers.
303512 markers.markers. markers. 251300 of 303512 markers. 259900 of 303512
markers.markers. 303512 markers.markers. 282200 of 303512 markers. markers. of
303512 markers. 288100 of 303512 markers.of 303512 markers. markers. markers. of
```

[192]:
```
# Construct a Boolean Series to identify outliers for genotypic data based on␣
 ↪MAF(minor allele frequency) rate
outliers_maf_def=test_last.maf>0.05
```

[193]:
```
geno_last=test_last.loc[outliers_maf_def]
```

[194]:
```
geno_last[:10]
```

[194]:
```
id AS6103 PHN11_LH145_0029 W10005_0107 W10005_0032 W10004_0082  \
3     NaN                C           C           C         NaN
8       A                A           A         NaN         NaN
13    NaN                G         NaN         NaN         NaN
15    NaN                A           T         NaN         NaN
16      C                C           C           C         NaN
17      C                C           T         NaN         NaN
19      G                G           G           G         NaN
20      G                G           G           G           G
21      A                A           G           G           A
22      A                A           A           A           A

id PHN11_LH145_0028 W10005_0029 W10004_0076 W10001_0018 PHW53 NyH-091  \
3                 C         NaN           C           C     C       C
8                 A           A           A         NaN     G       G
13              NaN         NaN         NaN         NaN   NaN     NaN
15                A           A           A         NaN     A       T
16                C           C           C         NaN     C       T
17                C           C           C           C     C       C
19                G           G           G         NaN     G       T
20                G           G           G           C     G       C
21                A           A           A           G     A       G
22                A           A           A           G     A       G
```

```
id W10006_0004 Mo44_PHW65_0175 Mo44_LH145_0081 Mo44_LH145_0011 MoG_Oh43_0057  \
3            C              NaN               T               T             C
8            G                G               A               A             G
13           A              NaN               A             NaN           NaN
15         NaN                A               A               A             T
16           C                C               C               C             T
17           C              NaN               C             NaN             C
19           G                G               G               G             T
20           C                C               C               C             C
21           G                G               G               G             G
22           A                A               A               A             G

id W10006_0007 W10005_0346 CI540       maf
3            C           C     C  0.105263
8          NaN         NaN   NaN  0.263158
13         NaN         NaN   NaN  0.052632
15           T           T   NaN  0.263158
16           C           C   NaN  0.105263
17           C         NaN   NaN  0.052632
19           G           G   NaN  0.105263
20         NaN           G   NaN  0.368421
21           G           G     G  0.368421
22           G           A     A  0.210526
```

[200]:
```
geno_last.to_csv('g2f_clean_s.txt')
```

2. 3 Transform SNP data to numerical data (0,1,2) with imputation

[55]:
```
import rpy2
import rpy2.robjects as robjects
from rpy2.robjects.packages import importr
import argparse
import sys
import textwrap
import timeit
import os
```

[59]:
```
## I connected the GAPIT.Numericalization and GAPIT.HapMap function. These R␣
 ↪codes will help convert the hapmap data to (1,0,2) data format
rstring="""
GAPIT.Numericalization <-
  function(x,bit=2,effect="Add",impute="Major", Create.indicator = FALSE, Major.
 ↪allele.zero = TRUE, byRow=TRUE){
    #Object: To convert character SNP genotpe to numerical
    #Output: Coresponding numerical value
    #Authors: Feng Tian and Zhiwu Zhang
    # Last update: May 30, 2011
```

```
  ␣
↪#########################################################################################
   if(bit==1)  {
     x[x=="X"]="N"
     x[x=="-"]="N"
     x[x=="+"]="N"
     x[x=="/"]="N"
     x[x=="H"]="Z" #K (for GT genotype)is replaced by Z to ensure heterozygose␣
↪has the largest value
   }

   if(bit==2)  {
     x[x=="XX"]="N"
     x[x=="--"]="N"
     x[x=="++"]="N"
     x[x=="//"]="N"
     x[x=="NN"]="N"
   }

   n=length(x)
   lev=levels(as.factor(x))
   lev=setdiff(lev,"N")
   #print(lev)
   len=length(lev)
   #print(lev)



   #Genotype counts
   count=1:len
   for(i in 1:len){
     count[i]=length(x[(x==lev[i])])
   }



   if(Major.allele.zero){
     if(len>1 & len<=3){
       #One bit: Make sure that the SNP with the major allele is on the top,␣
↪and the SNP with the minor allele is on the second position
       if(bit==1){
         count.temp = cbind(count, seq(1:len))
         if(len==3) count.temp = count.temp[-3,]
         count.temp <- count.temp[order(count.temp[,1], decreasing = TRUE),]
         if(len==3)order =  c(count.temp[,2],3)else order = count.temp[,2]
       }
```

```
      #Two bit: Make sure that the SNP with the major allele is on the top,␣
↪and the SNP with the minor allele is on the third position
      if(bit==2){
        count.temp = cbind(count, seq(1:len))
        if(len==3) count.temp = count.temp[-2,]
        count.temp <- count.temp[order(count.temp[,1], decreasing = TRUE),]
        if(len==3) order =  c(count.temp[1,2],2,count.temp[2,2])else order =␣
↪count.temp[,2]
      }

      count = count[order]
      lev = lev[order]

    }   #End  if(len<=1 | len> 3)
  } #End  if(Major.allele.zero)



  #make two  bit order genotype as AA,AT and TT, one bit as A(AA),T(TT) and␣
↪X(AT)
  if(bit==1 & len==3){
    temp=count[2]
    count[2]=count[3]
    count[3]=temp
  }
  position=order(count)


  #1status other than 2 or 3
  if(len<=1 | len> 3)x=0

  #2 status
  if(len==2)x=ifelse(x=="N",NA,ifelse(x==lev[1],0,2))

  #3 status
  if(bit==1){
    if(len==3)x=ifelse(x=="N",NA,ifelse(x==lev[1],0,ifelse(x==lev[3],1,2)))
  }else{
    if(len==3)x=ifelse(x=="N",NA,ifelse(x==lev[1],0,ifelse(x==lev[3],2,1)))
  }

  #print(paste(lev,len,sep=" "))
  #print(position)

  #missing data imputation
  if(impute=="Middle") {x[is.na(x)]=1 }
```

```
    if(len==3){
      if(impute=="Minor")  {x[is.na(x)]=position[1]  -1}
      if(impute=="Major")  {x[is.na(x)]=position[len]-1}

    }else{
      if(impute=="Minor")  {x[is.na(x)]=2*(position[1]  -1)}
      if(impute=="Major")  {x[is.na(x)]=2*(position[len]-1)}
    }

    #alternative genetic models
    if(effect=="Dom") x=ifelse(x==1,1,0)
    if(effect=="Left") x[x==1]=0
    if(effect=="Right") x[x==1]=2

    if(byRow) {
      result=matrix(x,n,1)
    }else{
      result=matrix(x,1,n)
    }

    return(result)
  }#end of GAPIT.Numericalization function

# Beginning of GAPIT.HapMap function


GAPIT.HapMap <-
  function(G,SNP.effect="Add",SNP.impute="Major",heading=TRUE, Create.indicator␣
 ↪= FALSE, Major.allele.zero = TRUE){
    #Object: To convert character SNP genotpe to numerical
    #Output: Coresponding numerical value
    #Authors: Feng Tian and Zhiwu Zhang
    # Last update: May 30, 2011
   ␣
 ↪##############################################################################################

    print(paste("Converting HapMap format to numerical under model of ", SNP.
 ↪impute,sep=""))
    #gc()
    #GAPIT.Memory.Object(name.of.trait="HapMap.Start")

    #GT=data.frame(G[1,-(1:5)])
    if(heading){
      GT= t(G[1,-(1:5)])
      GI= G[-1,c(1,3,4)]
    }else{
      GT=NULL
```

```
      GI= G[,c(1,3,4)]
    }


    #Set column names
    if(heading)colnames(GT)="taxa"
    colnames(GI)=c("SNP","Chromosome","Position")

    #Initial GD
    GD=NULL
    bit=nchar(as.character(G[2,12])) #to determine number of bits of genotype
    #print(paste("Number of bits for genotype: ", bit))

    print("Perform numericalization")

    if(heading){
      if(!Create.indicator) GD= apply(G[,-1],1,function(one) GAPIT.
↪Numericalization(one,bit=bit,effect=SNP.effect,impute=SNP.impute, Major.
↪allele.zero=Major.allele.zero))
      if(Create.indicator) GD= t(G[-1,-(1:5)])
    }else{
      if(!Create.indicator) GD= apply(G[  ,-(1:5)],1,function(one) GAPIT.
↪Numericalization(one,bit=bit,effect=SNP.effect,impute=SNP.impute, Major.
↪allele.zero=Major.allele.zero))
      if(Create.indicator) GD= t(G[ ,-(1:5)])
    }

    #set GT and GI to NULL in case of null GD
    if(is.null(GD)){
      GT=NULL
      GI=NULL
    }


    write.csv(GD, file ="Numeric data (0,1,2).csv")
    print("The dimension of GD is:")
    print(dim(GD))
    print ("Succesfuly finished converting HapMap to numeric data !")

    if(!Create.indicator) {print(paste("Succesfuly finished converting HapMap␣
↪which has bits of ", bit,sep="")) }
    return(list(GT=GT,GD=GD,GI=GI))
  }

"""
```

[60]:
```
readtable=robjects.r('read.csv')
```

[61]:
```
transfunc=robjects.r(rstring)
```

[63]:
```
r_g_data=readtable("g2f_clean_s.txt")
```

[64]:
```
r_df=transfunc(r_g_data)
```

##Numeric data (0,1,2) was exported from the above code, will be used in the following Genome Selection section

[ ]:
```
### This is end of this section
```