In [1]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from scipy import stats
from matplotlib.backends.backend_pdf import PdfPages
import seaborn as sns
```

Seaborn and Correlation Matrix

Today we are going to load seaborn and make on of their plots. They have some nice plots and it is becoming mainstream. https://seaborn.pydata.org/examples/index.html

First see if you have it. Type import seaborn as sns

If not to install there are two ways. First I am going to try the new way.

- Open Anaconda Navigator
- goto environments
- where is says installed click not installed or all
- search for seaborn and check it.

If that doesn't work we can do it in the terminal.

- open a new terminal window
- googel conda install seaborn
- open the page
- copy the "conda install -c anaconda seaborn"
- paste into temrinal and run

If those all fail find me.

Read in our well data

In [2]:
```
df=pd.read_csv('well_data.csv')
```

List all the columns

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Well_ID', 'Lat', 'Lon', 'Depth', 'Drink', 'Si', 'P', 'S', 'Ca', 'Fe',
       'Ba', 'Na', 'Mg', 'K', 'Mn', 'As', 'Sr', 'F', 'Cl', 'SO4', 'Br'],
      dtype='object')
```

If you ever need to delete a column this is how you do it. (not needed)

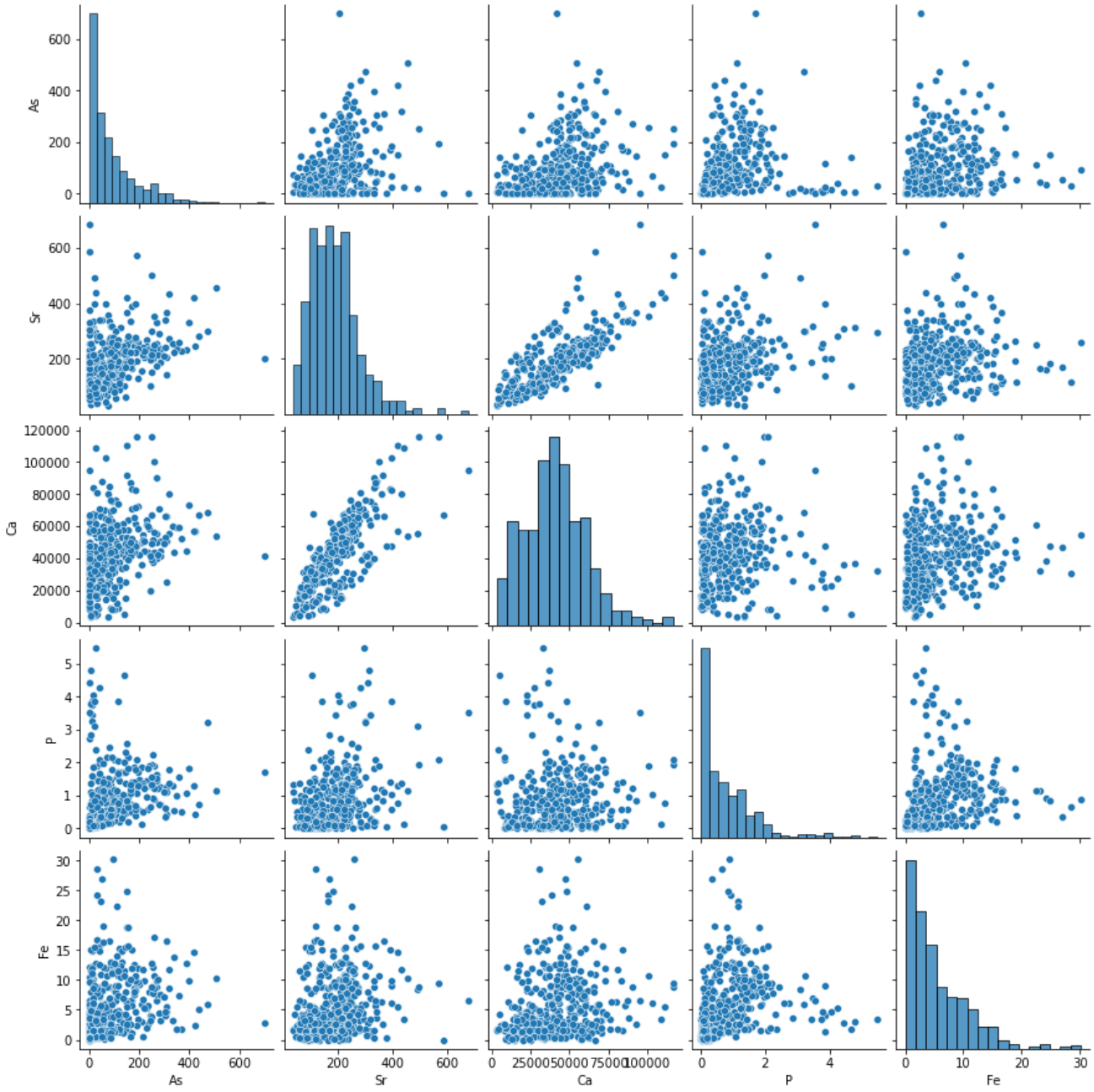
In [4]:
```
df = df.drop(['Well_ID','Drink'], axis=1)
```

Go back and find the columns you want to use and save it back to the dataframe.

In [6]:
```
df=df[['As', 'Sr', 'Ca', 'P', 'Fe']]
```

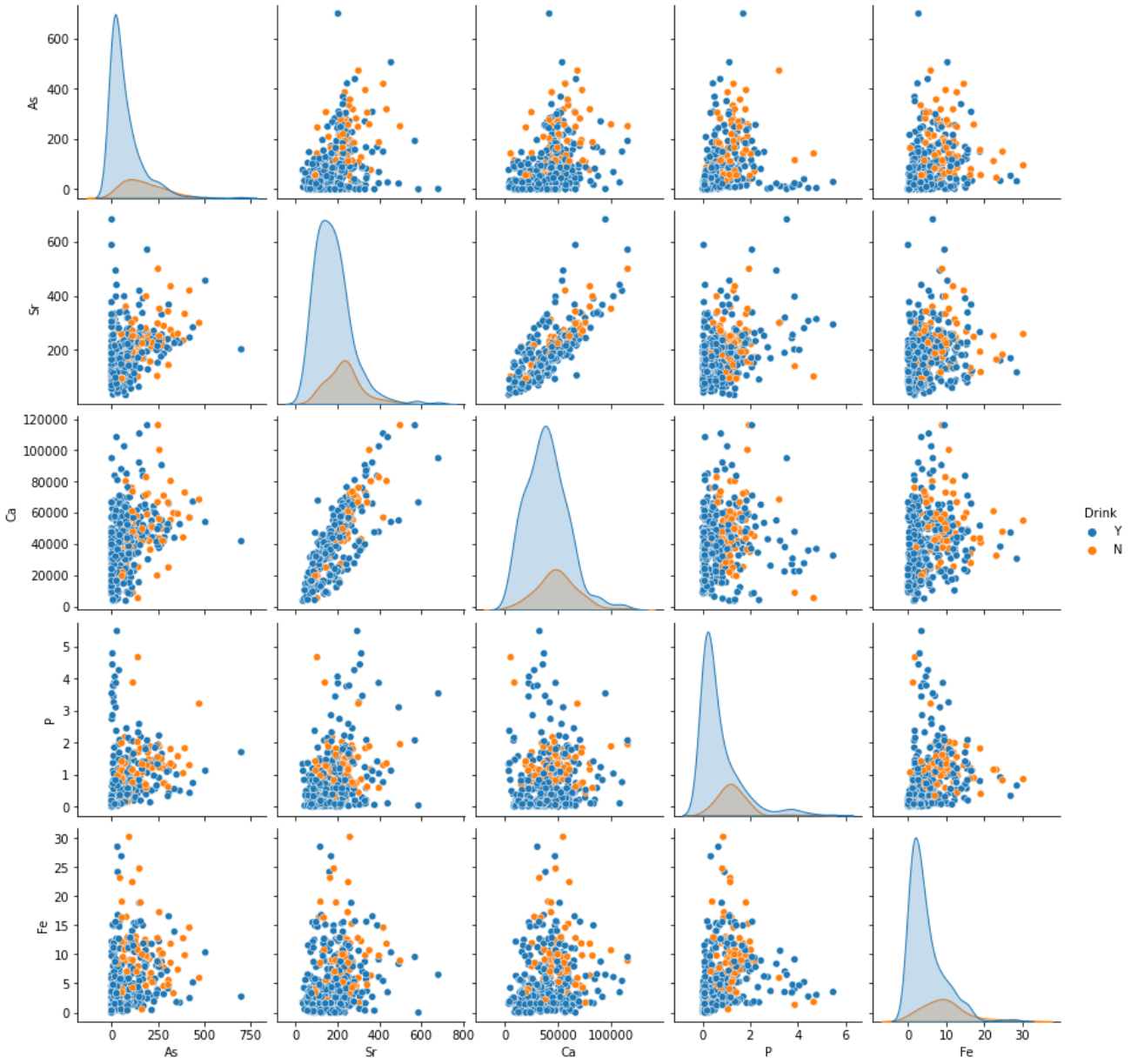
Run seaborn pairplot. https://seaborn.pydata.org/generated/seaborn.pairplot.html

In [7]:
```
sns.pairplot(df)
plt.savefig('test.jpg')
```



Also you can color by a categorical value. So add 'Drink' back in and use it to set hue

In [8]:
```
df=pd.read_csv('well_data.csv')
df=df[['As', 'Sr', 'Ca', 'P', 'Fe','Drink']]
sns.pairplot(df,hue='Drink')
plt.savefig('test.jpg')
```



If it is easier you could do it by column number instead

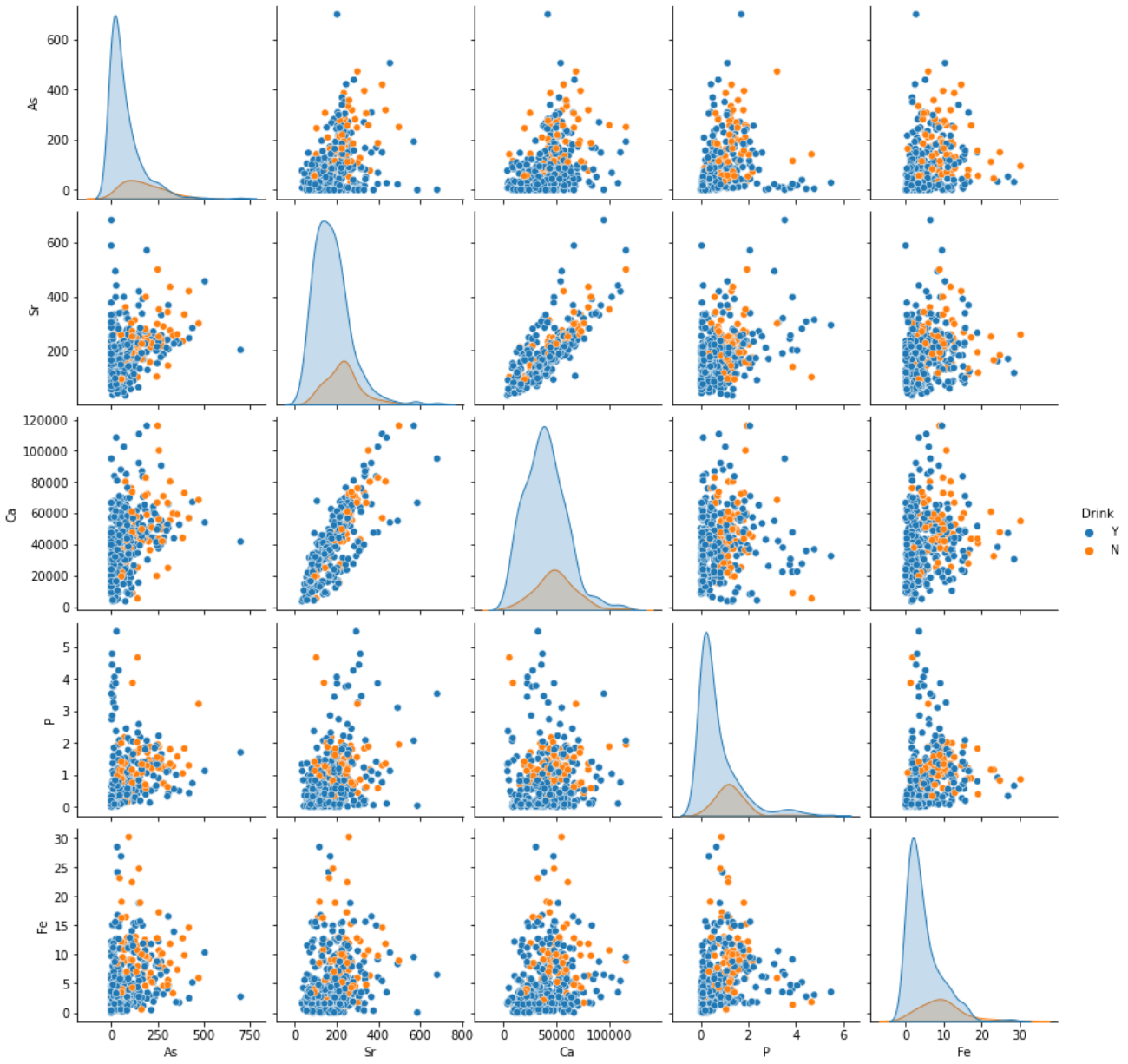
In [14]:
```
df=pd.read_csv('well_data.csv')
```

In [15]:
```
for count,col in enumerate(df):
    print(count,col)
```

```
0 Well_ID
1 Lat
2 Lon
3 Depth
4 Drink
5 Si
6 P
7 S
8 Ca
9 Fe
10 Ba
11 Na
12 Mg
13 K
14 Mn
15 As
16 Sr
17 F
18 Cl
```

```
19 SO4
20 Br
```

In [18]:
```
df=df.iloc[:,[15, 16, 8, 6, 9,4]]
sns.pairplot(df,hue='Drink')
plt.savefig('test.jpg')
```



In [ ]:

In [ ]:

In [ ]:

This is all extra material below that I found fun

But you don't get all the r-values or p-values. Can you make an excel file of all of them? I would

1. Just take all the numerical data. See first line for how I did it.
2. Make a double for loop to go through the data
3. Calculate the r2
4. print to your new dataframe
5. save to excel

In [16]:
```
df=df_well_data.select_dtypes(include=np.number)
df_r2=pd.DataFrame()

for i in df:
    for j in df:
        print (i,j)
        results=stats.linregress(df_well_data[[i,j]].dropna())
        df_r2.at[j,i]=results.rvalue**2

df_r2.to_excel('All_correlations.xlsx')
```

```
Well_ID Well_ID
Well_ID Lat
Well_ID Lon
Well_ID Depth
Well_ID Si
Well_ID P
Well_ID S
Well_ID Ca
Well_ID Fe
Well_ID Ba
Well_ID Na
Well_ID Mg
Well_ID K
Well_ID Mn
Well_ID As
Well_ID Sr
Well_ID F
Well_ID Cl
Well_ID SO4
Well_ID Br
Lat Well_ID
Lat Lat
Lat Lon
Lat Depth
Lat Si
Lat P
Lat S
Lat Ca
Lat Fe
Lat Ba
Lat Na
Lat Mg
Lat K
Lat Mn
Lat As
Lat Sr
Lat F
Lat Cl
Lat SO4
Lat Br
Lon Well_ID
Lon Lat
Lon Lon
Lon Depth
Lon Si
Lon P
Lon S
Lon Ca
Lon Fe
Lon Ba
Lon Na
Lon Mg
Lon K
Lon Mn
Lon As
Lon Sr
Lon F
Lon Cl
Lon SO4
Lon Br
Depth Well_ID
Depth Lat
Depth Lon
Depth Depth
```

```
Depth Si
Depth P
Depth S
Depth Ca
Depth Fe
Depth Ba
Depth Na
Depth Mg
Depth K
Depth Mn
Depth As
Depth Sr
Depth F
Depth Cl
Depth SO4
Depth Br
Si Well_ID
Si Lat
Si Lon
Si Depth
Si Si
Si P
Si S
Si Ca
Si Fe
Si Ba
Si Na
Si Mg
Si K
Si Mn
Si As
Si Sr
Si F
Si Cl
Si SO4
Si Br
P Well_ID
P Lat
P Lon
P Depth
P Si
P P
P S
P Ca
P Fe
P Ba
P Na
P Mg
P K
P Mn
P As
P Sr
P F
P Cl
P SO4
P Br
S Well_ID
S Lat
S Lon
S Depth
S Si
S P
S S
S Ca
S Fe
S Ba
S Na
S Mg
S K
S Mn
S As
S Sr
S F
S Cl
S SO4
S Br
Ca Well_ID
```

```
Ca Lat
Ca Lon
Ca Depth
Ca Si
Ca P
Ca S
Ca Ca
Ca Fe
Ca Ba
Ca Na
Ca Mg
Ca K
Ca Mn
Ca As
Ca Sr
Ca F
Ca Cl
Ca SO4
Ca Br
Fe Well_ID
Fe Lat
Fe Lon
Fe Depth
Fe Si
Fe P
Fe S
Fe Ca
Fe Fe
Fe Ba
Fe Na
Fe Mg
Fe K
Fe Mn
Fe As
Fe Sr
Fe F
Fe Cl
Fe SO4
Fe Br
Ba Well_ID
Ba Lat
Ba Lon
Ba Depth
Ba Si
Ba P
Ba S
Ba Ca
Ba Fe
Ba Ba
Ba Na
Ba Mg
Ba K
Ba Mn
Ba As
Ba Sr
Ba F
Ba Cl
Ba SO4
Ba Br
Na Well_ID
Na Lat
Na Lon
Na Depth
Na Si
Na P
Na S
Na Ca
Na Fe
Na Ba
Na Na
Na Mg
Na K
Na Mn
Na As
Na Sr
Na F
Na Cl
```

```
Na SO4
Na Br
Mg Well_ID
Mg Lat
Mg Lon
Mg Depth
Mg Si
Mg P
Mg S
Mg Ca
Mg Fe
Mg Ba
Mg Na
Mg Mg
Mg K
Mg Mn
Mg As
Mg Sr
Mg F
Mg Cl
Mg SO4
Mg Br
K Well_ID
K Lat
K Lon
K Depth
K Si
K P
K S
K Ca
K Fe
K Ba
K Na
K Mg
K K
K Mn
K As
K Sr
K F
K Cl
K SO4
K Br
Mn Well_ID
Mn Lat
Mn Lon
Mn Depth
Mn Si
Mn P
Mn S
Mn Ca
Mn Fe
Mn Ba
Mn Na
Mn Mg
Mn K
Mn Mn
Mn As
Mn Sr
Mn F
Mn Cl
Mn SO4
Mn Br
As Well_ID
As Lat
As Lon
As Depth
As Si
As P
As S
As Ca
As Fe
As Ba
As Na
As Mg
As K
As Mn
As As
```

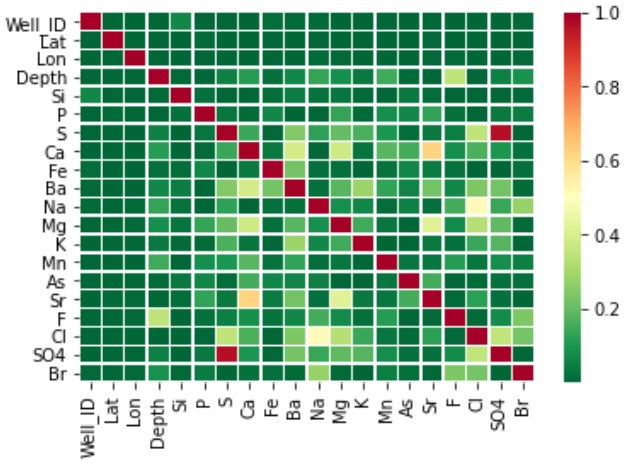
```
As Sr
As F
As Cl
As SO4
As Br
Sr Well_ID
Sr Lat
Sr Lon
Sr Depth
Sr Si
Sr P
Sr S
Sr Ca
Sr Fe
Sr Ba
Sr Na
Sr Mg
Sr K
Sr Mn
Sr As
Sr Sr
Sr F
Sr Cl
Sr SO4
Sr Br
F Well_ID
F Lat
F Lon
F Depth
F Si
F P
F S
F Ca
F Fe
F Ba
F Na
F Mg
F K
F Mn
F As
F Sr
F F
F Cl
F SO4
F Br
Cl Well_ID
Cl Lat
Cl Lon
Cl Depth
Cl Si
Cl P
Cl S
Cl Ca
Cl Fe
Cl Ba
Cl Na
Cl Mg
Cl K
Cl Mn
Cl As
Cl Sr
Cl F
Cl Cl
Cl SO4
Cl Br
SO4 Well_ID
SO4 Lat
SO4 Lon
SO4 Depth
SO4 Si
SO4 P
SO4 S
SO4 Ca
SO4 Fe
SO4 Ba
SO4 Na
SO4 Mg
```

```
SO4 K
SO4 Mn
SO4 As
SO4 Sr
SO4 F
SO4 Cl
SO4 SO4
SO4 Br
Br Well_ID
Br Lat
Br Lon
Br Depth
Br Si
Br P
Br S
Br Ca
Br Fe
Br Ba
Br Na
Br Mg
Br K
Br Mn
Br As
Br Sr
Br F
Br Cl
Br SO4
Br Br
```

You can open your excel file or visualize in sseaborn.

In [19]:
```
sns.heatmap(df_r2, cmap='RdYlGn_r', linewidths=0.5, annot=False)
```

Out[19]: <AxesSubplot:>



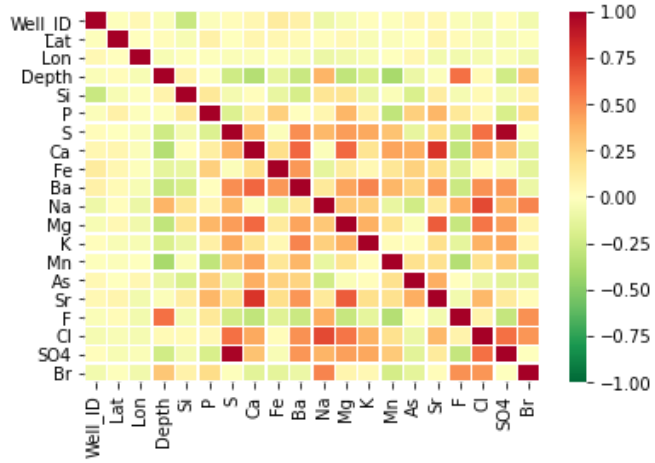
Try with r-vlaues

In [22]:
```
df=df_well_data.select_dtypes(include=np.number)
df_r2=pd.DataFrame()

for i in df:
    for j in df:
        results=stats.linregress(df_well_data[[i,j]].dropna())
        df_r2.at[j,i]=results.rvalue

sns.heatmap(df_r2, cmap='RdYlGn_r', linewidths=0.5, annot=False,vmin=-1,vmax=1)
```

Out[22]: <AxesSubplot:>

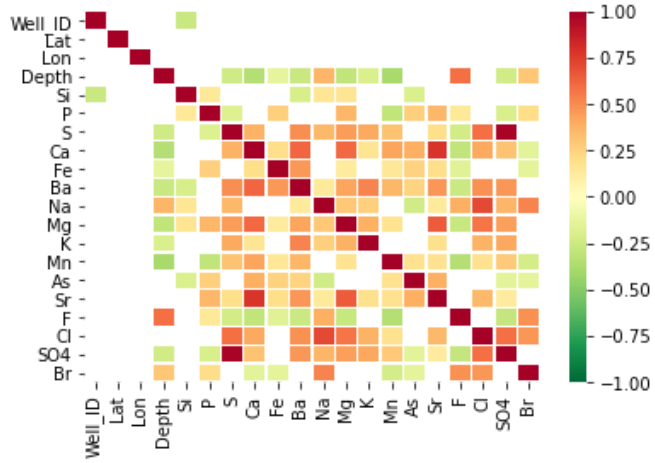


For excel I would do r-values and only save for when p-value is less than 0.01

In [25]:
```
df=df_well_data.select_dtypes(include=np.number)
df_r2=pd.DataFrame()

for i in df:
    for j in df:
        results=stats.linregress(df_well_data[[i,j]].dropna())
        if results.pvalue<0.01:
            df_r2.at[j,i]=results.rvalue
        else:
            df_r2.at[j,i]=np.nan
df_r2.to_excel('r-values.xlsx')
sns.heatmap(df_r2, cmap='RdYlGn_r', linewidths=0.5, annot=False,vmin=-1,vmax=1)
```

Out[25]: <AxesSubplot:>



If column names are really long remember you can use iloc to chosse a datafram by column number

In [29]:
```
df.iloc[:,3:7]
```

Out[29]:

	Depth	Si	P	S
0	45	NaN	NaN	NaN
1	60	NaN	NaN	NaN
2	60	NaN	NaN	NaN
3	50	48084.33842	0.936358	2085.570979
4	150	NaN	NaN	NaN
...	...	...	...	...
754	160	32379.64000	0.197380	3669.430000

	Depth	Si	P	S
755	60	25561.12000	0.090570	13771.370000
756	45	31319.48000	1.162550	38.300000
757	60	30605.53000	1.556120	4168.520000
758	50	NaN	NaN	NaN

759 rows × 4 columns

In [ ]: