Final Project Python Results S01B-01

Teammates: Kai Yang(Ben), Jiaheng Shao(Steve), Qianqian Xiao, Xiaoqi Zhong(Elly)

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats
import statsmodels.api as sm
import numpy.ma as ma
import datetime as dt
import pingouin
```

In [ ]:
```
data_1 = pd.read_excel("Regression 1.xlsx")
data_2 = pd.read_excel("Regression 2.xlsx")
data_3_1 = pd.read_excel("Regression 3-1.xlsx")
data_3_2 = pd.read_excel("Regression 3-2.xlsx")
```

In [ ]:
```
#Test 1
## Processing the data
factors1 = ["Market Average Spread", "Rating", "Issuing Amount", "Tenor", "G

X1 = sm.add_constant(data_1[factors1])
model1 = sm.OLS(data_1["Coupon"], X1).fit()
# Fit the model
prediction1 = model1.predict(X1)
# Print the parameters of the fitted model
model1.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Coupon	**R-squared:**	0.834
Model:	OLS	**Adj. R-squared:**	0.829
Method:	Least Squares	**F-statistic:**	146.1
Date:	Wed, 02 Nov 2022	**Prob (F-statistic):**	1.13e-75
Time:	12:06:02	**Log-Likelihood:**	-38.813
No. Observations:	211	**AIC:**	93.63
Df Residuals:	203	**BIC:**	120.4
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1715	0.184	0.932	0.352	-0.191	0.534
Market Average Spread	1.0509	0.065	16.193	0.000	0.923	1.179
Rating	-0.1385	0.031	-4.499	0.000	-0.199	-0.078
Issuing Amount	-0.0031	0.002	-1.533	0.127	-0.007	0.001
Tenor	0.0355	0.022	1.601	0.111	-0.008	0.079
Green Indicator	-0.1127	0.134	-0.841	0.401	-0.377	0.152
Russell ESG Score	0.0790	0.043	1.860	0.064	-0.005	0.163
interaction	0.0499	0.074	0.675	0.500	-0.096	0.196

Omnibus:	110.439	**Durbin-Watson:**	1.225
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1010.945
Skew:	1.792	**Prob(JB):**	2.99e-220
Kurtosis:	13.106	**Cond. No.**	198.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
## Obtaining the prediction and residual values
result1 = pd.concat([prediction1, model1.resid], axis =1)
result1 = result1.rename(columns = {0:'prediction', 1:'residual'})
```

In [ ]:

```
## Print the regression results
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
    print(result1)
```

	prediction	residual
0	3.135	0.535
1	3.028	0.442
2	2.824	0.386
3	2.969	0.211
4	2.984	0.366
5	2.978	0.122
6	2.981	0.349
7	3.109	0.091
8	2.866	0.384
9	2.818	0.152
10	2.630	0.220
11	2.318	0.362
12	2.580	0.290
13	2.603	0.177
14	2.728	0.202
15	2.605	0.275
16	2.555	0.195
17	2.253	-0.253
18	2.650	-0.320
19	2.461	-0.211
20	2.296	-0.156
21	2.437	-0.207
22	2.201	0.029
23	2.296	-0.126
24	2.409	0.041
25	2.460	0.040
26	2.358	0.142
27	2.762	-0.162
28	2.286	-0.456
29	1.456	0.424
30	1.465	0.035
31	1.742	-0.132
32	1.739	-0.129
33	1.742	-0.132
34	1.612	-0.112
35	1.744	0.046
36	1.520	-0.050
37	1.726	-0.136
38	1.510	-0.010
39	1.947	-0.027
40	1.528	-0.048
41	1.536	-0.286
42	1.714	-0.214
43	1.691	-0.191
44	1.709	0.001
45	1.883	-0.333
46	1.644	-0.144
47	1.648	-0.148
48	1.650	-0.150
49	1.735	0.005
50	1.707	-0.207
51	1.509	-0.049
52	1.675	-0.175
53	1.689	0.061
54	1.682	-0.102
55	1.392	0.078
56	1.663	0.587
57	1.631	0.019
58	1.543	0.087
59	1.673	0.327
60	1.431	0.039
61	1.661	-0.011
62	1.561	-0.051

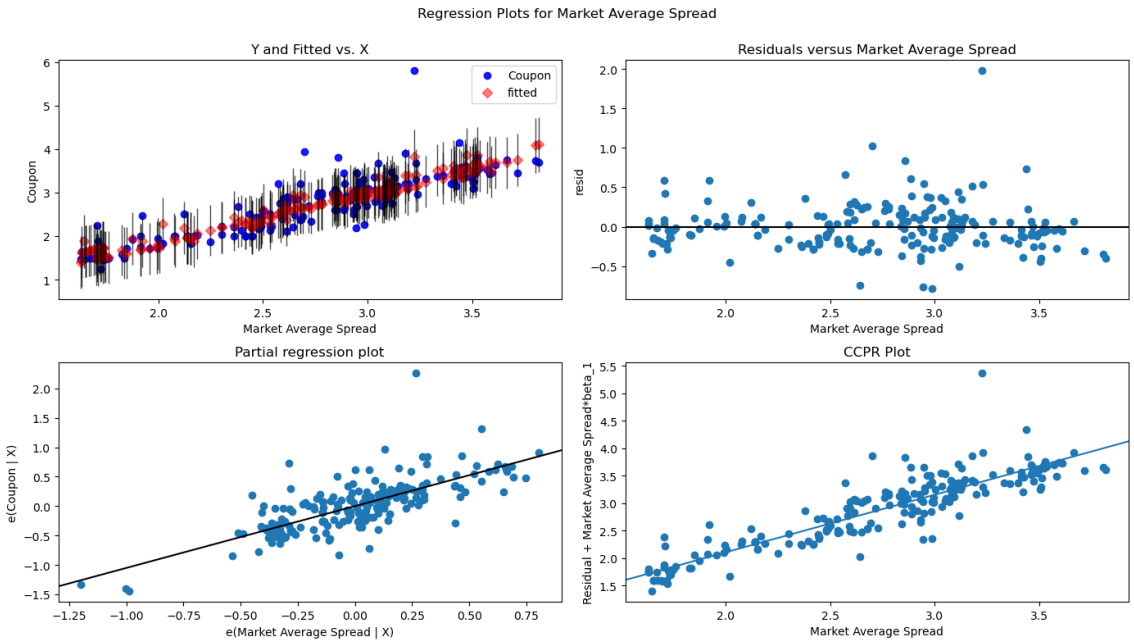
63	1.606	0.114
64	1.889	0.591
65	1.817	0.133
66	2.133	-0.273
67	1.878	0.122
68	1.940	-0.110
69	1.940	-0.110
70	2.193	0.307
71	1.935	0.065
72	1.884	0.116
73	2.054	-0.034
74	1.961	0.039
75	1.961	0.039
76	1.961	0.039
77	1.961	0.039
78	1.961	0.039
79	2.201	-0.161
80	2.432	-0.232
81	2.226	-0.226
82	2.195	-0.195
83	2.195	-0.195
84	2.611	-0.031
85	2.947	-0.747
86	2.384	-0.274
87	2.267	-0.097
88	2.339	-0.139
89	2.289	-0.289
90	2.289	-0.289
91	2.327	0.023
92	2.297	0.133
93	2.319	-0.089
94	2.605	0.245
95	2.895	0.005
96	3.516	-0.136
97	3.019	-0.069
98	3.615	-0.105
99	3.064	-0.114
100	3.624	-0.074
101	3.150	0.100
102	3.543	0.007
103	3.690	-0.060
104	2.663	0.087
105	2.761	0.039
106	2.831	0.239
107	3.526	-0.436
108	3.061	-0.261
109	2.982	-0.382
110	2.859	0.241
111	2.449	0.051
112	2.819	0.171
113	3.027	0.123
114	2.965	0.115
115	3.043	0.087
116	3.371	-0.121
117	2.844	-0.144
118	3.472	-0.072
119	2.995	0.055
120	2.962	-0.012
121	3.332	0.068
122	3.477	-0.027
123	2.655	0.045
124	3.521	-0.131
125	2.998	-0.048
126	2.835	0.065

127	3.249	0.071
128	3.614	-0.044
129	3.106	0.014
130	3.043	0.027
131	3.069	-0.009
132	2.553	-0.153
133	4.104	-0.404
134	3.003	-0.163
135	4.090	-0.350
136	3.059	0.011
137	3.064	-0.074
138	3.472	-0.122
139	3.755	-0.305
140	2.350	-0.210
141	2.653	-0.283
142	3.402	-0.202
143	3.570	-0.170
144	2.965	-0.365
145	3.251	-0.501
146	3.028	-0.198
147	2.804	-0.314
148	3.262	-0.132
149	3.407	-0.107
150	3.690	-0.240
151	2.726	-0.236
152	3.350	0.330
153	2.945	0.255
154	2.805	0.125
155	2.995	0.055
156	3.128	0.072
157	2.757	-0.107
158	2.898	-0.198
159	2.932	0.118
160	2.966	0.834
161	3.385	0.515
162	3.073	0.027
163	3.822	1.978
164	2.963	0.037
165	2.904	0.186
166	3.327	0.063
167	3.170	-0.210
168	3.353	0.227
169	2.903	0.547
170	2.838	0.612
171	3.419	0.731
172	2.927	0.373
173	2.912	1.028
174	2.669	-0.289
175	2.914	0.086
176	2.809	0.171
177	2.725	0.225
178	3.071	0.009
179	3.534	-0.014
180	3.687	0.073
181	2.535	0.665
182	2.581	0.319
183	2.707	0.293
184	3.711	-0.271
185	3.128	-0.038
186	2.911	-0.011
187	3.510	-0.110
188	3.580	-0.050
189	3.084	-0.284
190	2.837	-0.237

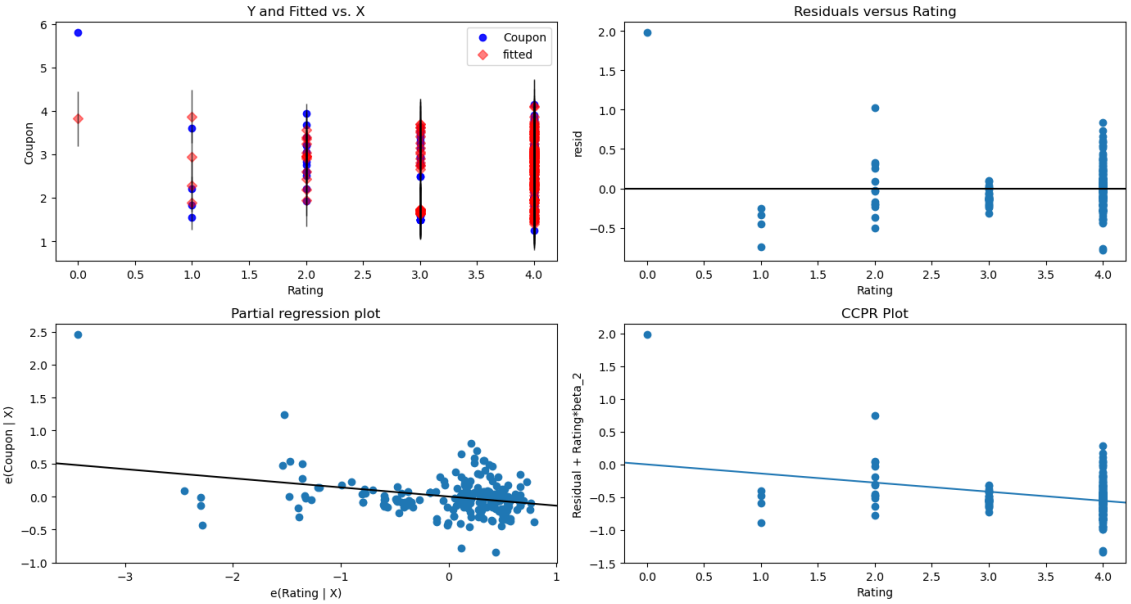
```
191        3.472    -0.032
192        3.022     0.098
193        3.478    -0.028
194        2.978    -0.138
195        3.868    -0.258
196        2.740     0.260
197        3.008     0.092
198        2.955    -0.275
199        2.647    -0.197
200        2.824    -0.024
201        3.069    -0.119
202        3.408    -0.208
203        2.935    -0.435
204        2.952    -0.762
205        3.043    -0.783
206        2.959     0.091
207        3.868    -0.068
208        3.543    -0.393
209        3.647    -0.397
210        3.656     0.054
```

In [ ]:
```
## Show the regression in charts
for x in range(len(factors1)):
    fig = plt.figure(figsize=(14, 8))
    fig = sm.graphics.plot_regress_exog(model1,
                                        factors1[x],
                                        fig=fig)
```

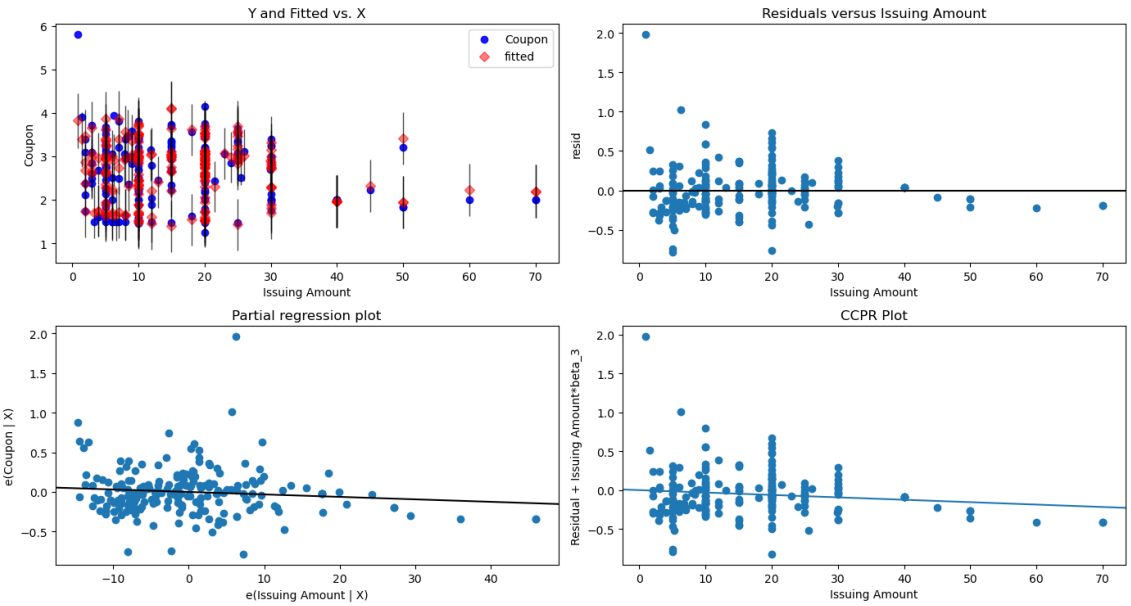
```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```



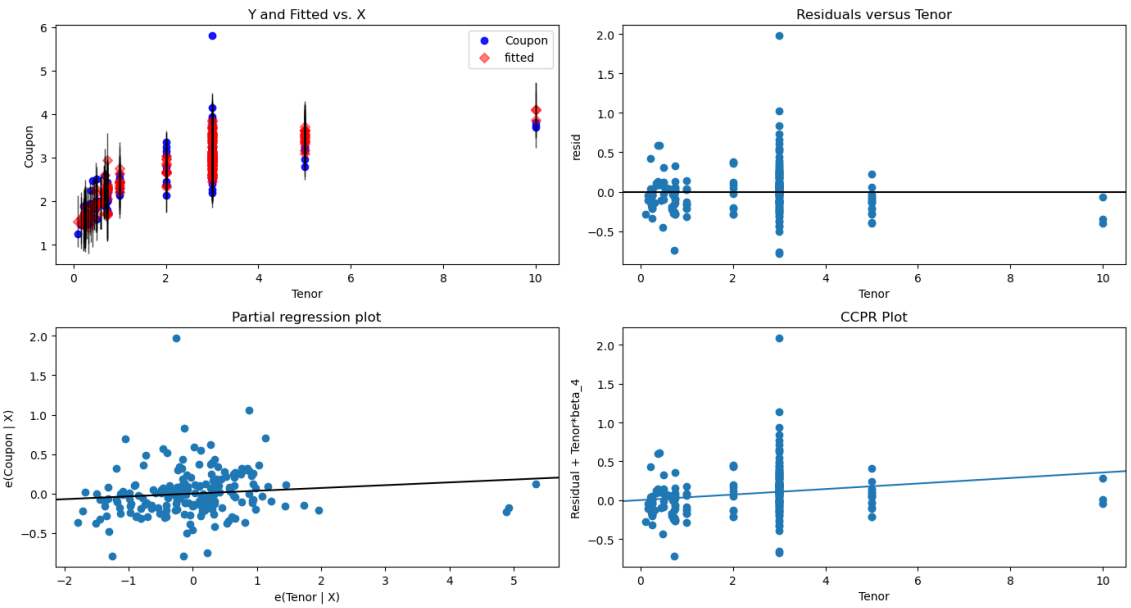
Regression Plots for Rating

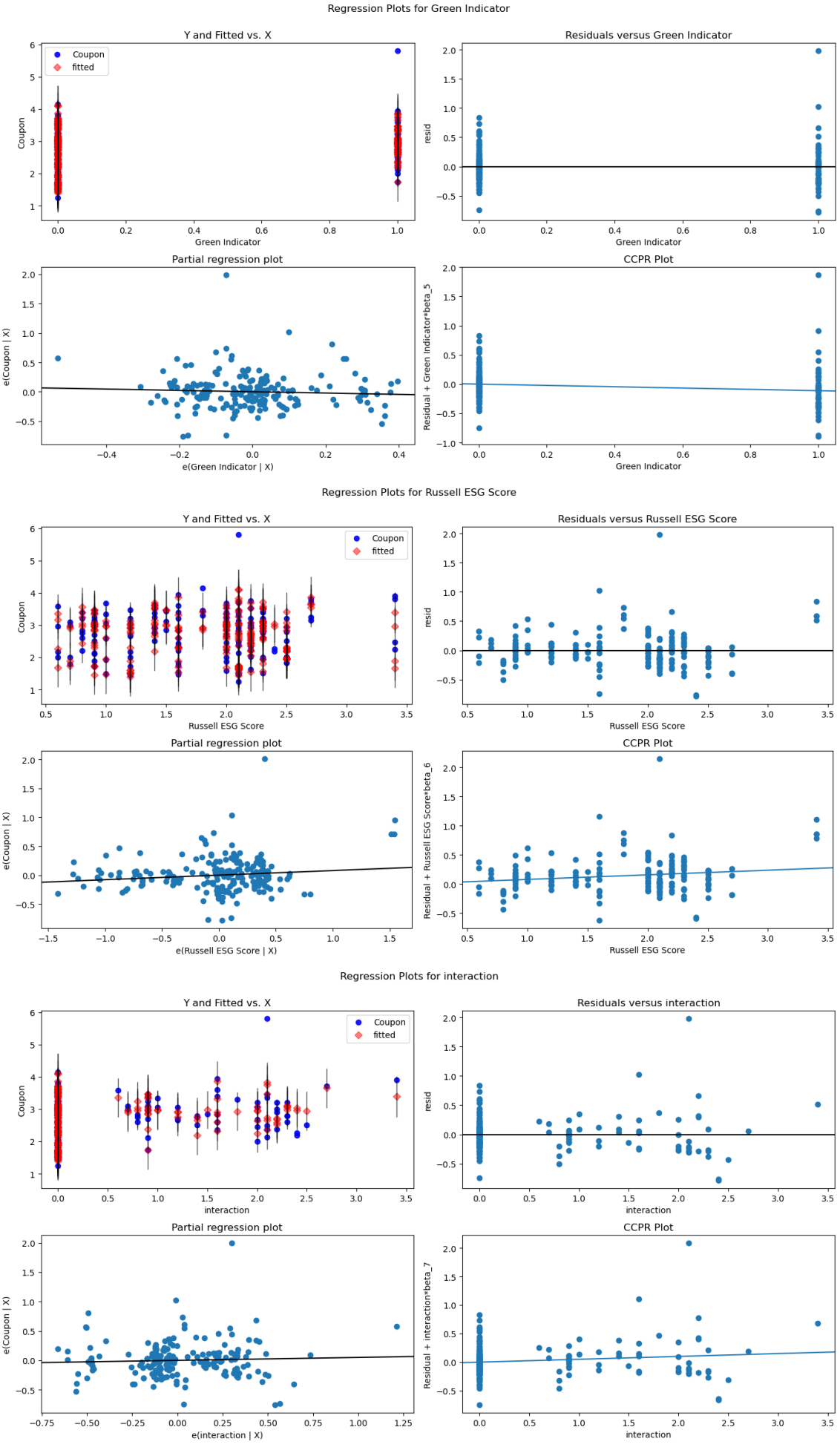


Regression Plots for Issuing Amount



Regression Plots for Tenor





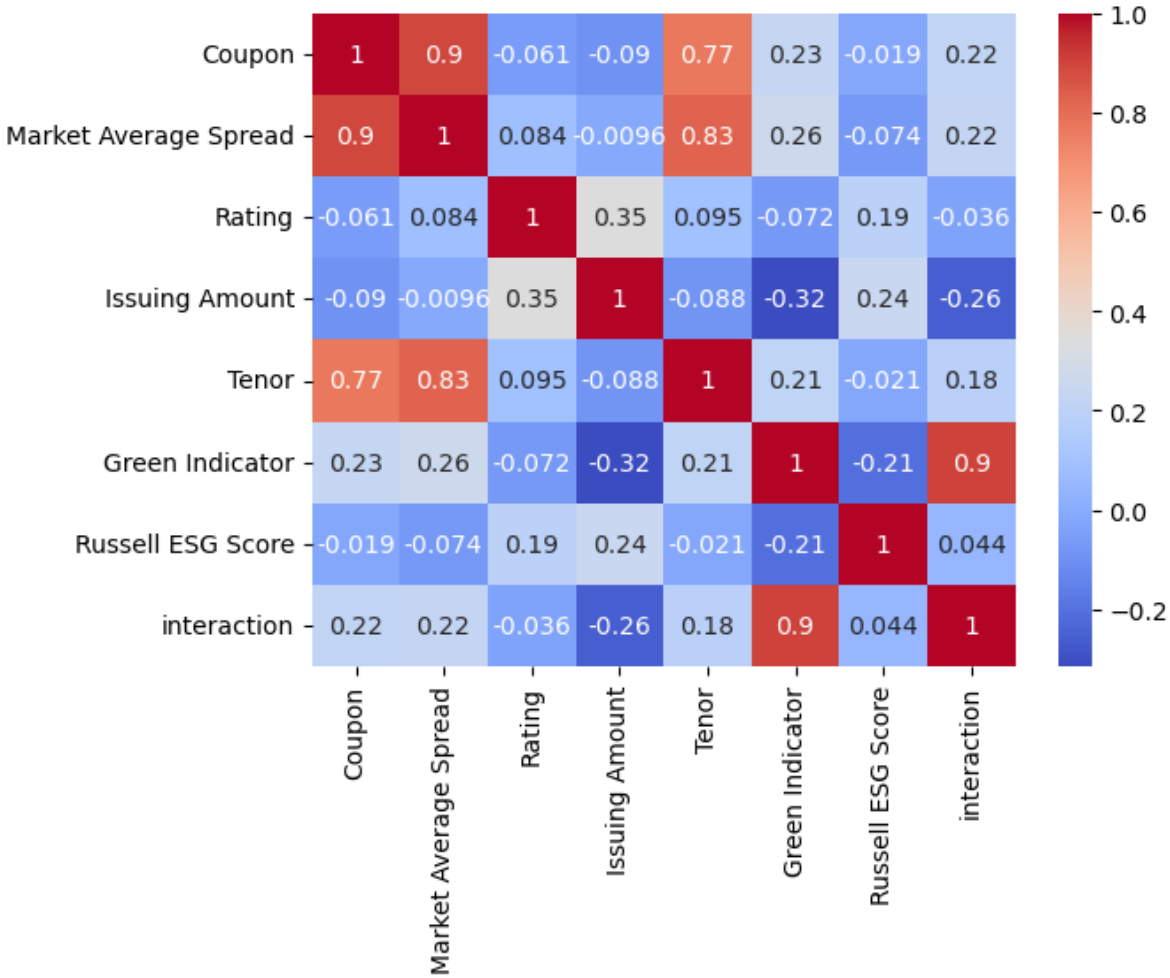
In [ ]:
```
## Anova test
pingouin.anova(data = data_1, dv = "Coupon", between = "Russell ESG Score")
```

Out[ ]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Russell ESG Score	17	193	2.850154	0.000255	0.200671

In [ ]:
```
## Heatmaps
sns.heatmap(data_1.corr(), cmap='coolwarm', annot=True)
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
# Test 2
## Processing the data
factors2 = ["overnight", "Refinitiv ESG Score", "rating", "Amount Issued (US

X2 = sm.add_constant(data_2[factors2])
model2 = sm.OLS(data_2["Coupon"], X2).fit()
# Fit the model
prediction2 = model2.predict(X2)
# Print the parameters of the fitted model
model2.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Coupon	**R-squared:**	0.561
Model:	OLS	**Adj. R-squared:**	0.553
Method:	Least Squares	**F-statistic:**	67.73
Date:	Wed, 02 Nov 2022	**Prob (F-statistic):**	5.76e-54
Time:	12:06:08	**Log-Likelihood:**	-465.11
No. Observations:	325	**AIC:**	944.2
Df Residuals:	318	**BIC:**	970.7
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.5240	0.346	10.193	0.000	2.844	4.204
overnight	1.5647	0.106	14.786	0.000	1.356	1.773
Refinitiv ESG Score	-0.0078	0.004	-1.764	0.079	-0.017	0.001
rating	-0.1565	0.028	-5.644	0.000	-0.211	-0.102
Amount Issued (USD)	3.087e-10	2.02e-10	1.530	0.127	-8.83e-11	7.06e-10
Year	0.0002	0.001	0.154	0.877	-0.002	0.002
interaction	-0.0042	0.008	-0.538	0.591	-0.020	0.011

Omnibus:	34.505	**Durbin-Watson:**	1.008
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	45.585
Skew:	0.754	**Prob(JB):**	1.26e-10
Kurtosis:	4.046	**Cond. No.**	4.32e+09

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.32e+09. This might indicate that there are strong multicollinearity or other numerical problems.

In [ ]:
```
## Obtaining the prediction and residual values
result2 = pd.concat([prediction2, model2.resid], axis =1)
result2 = result2.rename(columns = {0:'prediction', 1:'residual'})
```

In [ ]:
```
## Print the regression results
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
    print(result2)
```

	prediction	residual
0	3.254	1.121
1	1.959	-0.375
2	6.359	-1.234
3	3.372	1.378
4	6.107	-0.732
5	6.107	-0.732
6	6.020	-0.520
7	6.020	-0.520
8	3.203	1.547
9	3.194	0.306
10	2.916	0.209
11	3.118	0.595
12	3.264	1.611
13	2.669	0.456
14	2.760	0.115
15	3.367	1.133
16	5.712	0.541
17	5.712	0.541
18	3.094	0.031
19	2.653	3.347
20	2.176	1.074
21	4.721	0.404
22	4.734	0.641
23	5.337	-0.387
24	1.774	2.351
25	2.089	1.661
26	2.341	2.409
27	1.978	2.089
28	4.880	2.714
29	1.586	1.904
30	1.328	1.297
31	2.330	1.795
32	5.505	-0.755
33	6.601	-0.501
34	5.302	-1.152
35	6.111	-0.711
36	6.160	-0.560
37	1.993	0.882
38	4.728	1.103
39	4.728	1.103
40	4.700	1.836
41	4.700	1.836
42	1.102	3.398
43	4.169	-0.269
44	4.314	0.061
45	4.314	0.061
46	1.281	4.519
47	1.192	2.433
48	1.447	1.053
49	0.721	0.779
50	2.451	1.174
51	2.451	1.174
52	2.384	1.616
53	2.384	1.616
54	1.403	1.597
55	1.322	0.928
56	1.380	1.495
57	3.431	1.569
58	1.362	1.013
59	1.375	1.625
60	1.696	2.554
61	2.152	-1.528
62	1.369	0.881

63	1.305	1.195
64	1.612	0.513
65	1.187	2.063
66	0.674	1.451
67	1.622	1.128
68	2.095	-1.521
69	2.129	-1.485
70	1.540	0.585
71	1.147	1.103
72	3.385	1.240
73	3.385	1.240
74	0.918	1.332
75	1.208	0.917
76	2.979	1.271
77	0.792	0.958
78	1.052	0.867
79	0.414	0.649
80	2.014	-1.684
81	2.004	-1.430
82	0.702	0.923
83	3.071	0.829
84	1.352	0.648
85	1.657	1.093
86	1.495	-0.620
87	1.511	0.114
88	1.123	1.252
89	1.612	2.159
90	1.133	0.367
91	0.720	0.030
92	1.924	0.626
93	2.194	0.077
94	1.924	0.626
95	2.194	0.340
96	1.924	0.876
97	1.924	0.876
98	2.117	0.850
99	2.403	1.472
100	1.429	0.446
101	2.263	0.737
102	1.711	1.550
103	1.491	-0.116
104	1.233	-0.108
105	2.330	0.620
106	2.276	0.099
107	2.276	0.099
108	2.245	0.505
109	2.245	0.505
110	1.216	-0.466
111	1.623	-0.998
112	1.008	-0.133
113	2.481	-0.681
114	2.450	-0.200
115	2.999	0.571
116	2.999	0.571
117	2.152	0.420
118	2.152	0.420
119	2.511	0.339
120	1.704	-0.579
121	0.683	-0.308
122	1.536	-0.661
123	1.118	-0.368
124	1.780	0.692
125	3.305	0.445
126	3.305	0.445

127	3.305	0.445
128	1.741	-0.491
129	2.421	0.049
130	2.421	0.049
131	1.433	-1.183
132	1.264	-0.764
133	1.689	-0.689
134	1.566	-1.066
135	1.534	-0.534
136	1.489	-0.739
137	2.120	0.005
138	2.040	-0.140
139	0.416	-0.268
140	0.949	-0.199
141	1.739	-0.201
142	1.739	-0.201
143	3.437	-0.187
144	0.850	-0.600
145	2.086	-0.029
146	1.455	-0.455
147	2.716	-0.013
148	1.335	-0.960
149	1.353	-0.353
150	2.362	0.788
151	2.362	0.788
152	2.397	0.053
153	1.741	-1.116
154	1.754	-0.254
155	1.396	-0.271
156	1.402	1.913
157	1.463	-0.963
158	1.834	-0.034
159	2.269	1.231
160	2.269	1.231
161	2.565	-0.988
162	1.013	-0.888
163	2.133	-0.083
164	2.005	-0.130
165	2.080	-0.580
166	2.403	0.447
167	1.775	-0.400
168	1.569	-1.319
169	1.884	-0.509
170	2.514	-0.264
171	1.336	-0.961
172	1.797	-0.647
173	1.977	-0.852
174	1.960	-0.210
175	2.105	-0.755
176	2.358	-0.983
177	2.079	-0.704
178	2.079	-0.704
179	2.079	0.296
180	2.079	0.296
181	1.342	-0.542
182	1.285	-0.660
183	1.014	-0.139
184	2.104	0.796
185	1.341	-0.716
186	1.419	-1.044
187	1.749	-1.249
188	1.760	-0.510
189	1.526	-0.151
190	1.509	-0.884

191	1.442	-0.692
192	1.448	-0.573
193	2.177	-0.802
194	2.467	-0.967
195	1.419	-0.544
196	1.385	-0.260
197	1.413	-1.113
198	0.668	-0.418
199	1.222	-0.722
200	2.568	-0.068
201	3.230	1.020
202	1.447	-0.681
203	2.545	1.080
204	2.545	1.080
205	2.245	0.055
206	1.496	-0.121
207	2.190	-0.190
208	2.183	-1.183
209	2.183	-1.183
210	2.214	-0.464
211	2.214	-0.464
212	1.112	0.263
213	1.538	-0.938
214	1.125	-0.225
215	1.170	-1.124
216	1.862	-0.312
217	1.862	-0.312
218	1.934	0.441
219	1.186	-0.686
220	1.112	-0.612
221	1.815	-0.565
222	1.372	-0.622
223	1.156	-0.406
224	1.251	-0.876
225	0.825	-0.700
226	2.346	-1.221
227	1.758	-1.008
228	1.029	-1.019
229	2.331	-0.331
230	2.525	-0.025
231	0.416	-0.348
232	1.299	-1.299
233	1.656	-1.156
234	2.085	-0.210
235	1.464	-1.464
236	1.567	-1.317
237	2.529	-0.154
238	2.529	-0.154
239	1.826	-1.318
240	2.310	-0.410
241	1.350	-1.350
242	2.731	-1.356
243	2.731	-1.356
244	2.791	-0.341
245	2.791	-0.341
246	1.159	-1.149
247	1.512	-0.887
248	1.064	-1.064
249	2.437	-0.937
250	1.270	-1.145
251	1.661	-1.286
252	2.300	-0.600
253	1.631	-0.631
254	1.957	-1.007

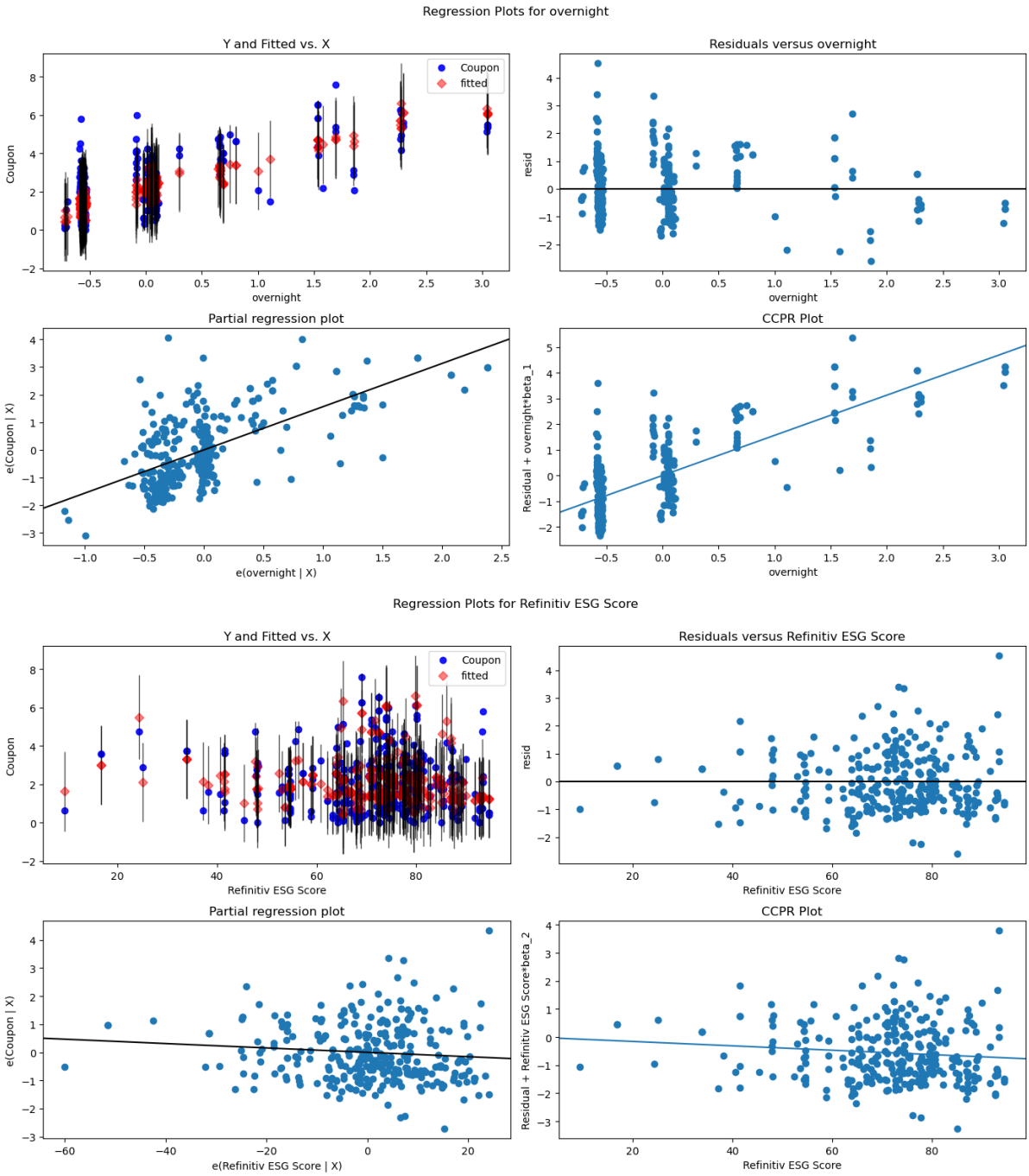
255	1.206	-1.081
256	1.220	-0.845
257	2.598	-1.598
258	0.774	-0.560
259	1.751	0.749
260	2.582	-1.032
261	2.531	0.419
262	1.806	0.319
263	1.613	1.262
264	1.445	-0.695
265	2.043	1.207
266	1.532	-0.657
267	2.465	-0.965
268	2.365	0.260
269	2.365	0.260
270	2.457	-0.707
271	1.162	-0.412
272	1.388	-0.388
273	2.229	-0.379
274	1.934	-0.309
275	1.843	-1.093
276	1.791	1.159
277	1.417	-0.667
278	1.358	1.071
279	2.258	0.442
280	2.474	-1.074
281	2.474	-1.074
282	1.917	-0.042
283	2.364	-0.414
284	2.364	-0.114
285	1.371	-0.246
286	1.757	-0.717
287	1.417	-0.667
288	1.405	-1.030
289	1.519	-0.394
290	1.519	-0.394
291	0.777	0.598
292	1.180	-0.930
293	2.055	0.304
294	1.494	-0.619
295	1.758	0.992
296	1.680	-0.055
297	2.362	0.838
298	1.986	0.964
299	1.396	0.354
300	2.044	0.581
301	1.406	0.719
302	3.698	-2.198
303	1.330	-1.080
304	1.725	-0.350
305	1.537	-0.912
306	2.119	-0.419
307	1.649	-1.274
308	1.250	-0.500
309	1.243	-0.743
310	1.700	0.175
311	1.309	0.691
312	0.494	-0.494
313	1.222	-0.347
314	0.509	-0.009
315	4.461	-2.261
316	1.753	-0.128
317	1.421	-0.921
318	4.649	-2.599

```
319        1.342     -0.967
320        4.951     -1.851
321        4.411     -1.536
322        1.044     -0.744
323        3.077     -1.000
324        1.406      0.718
```

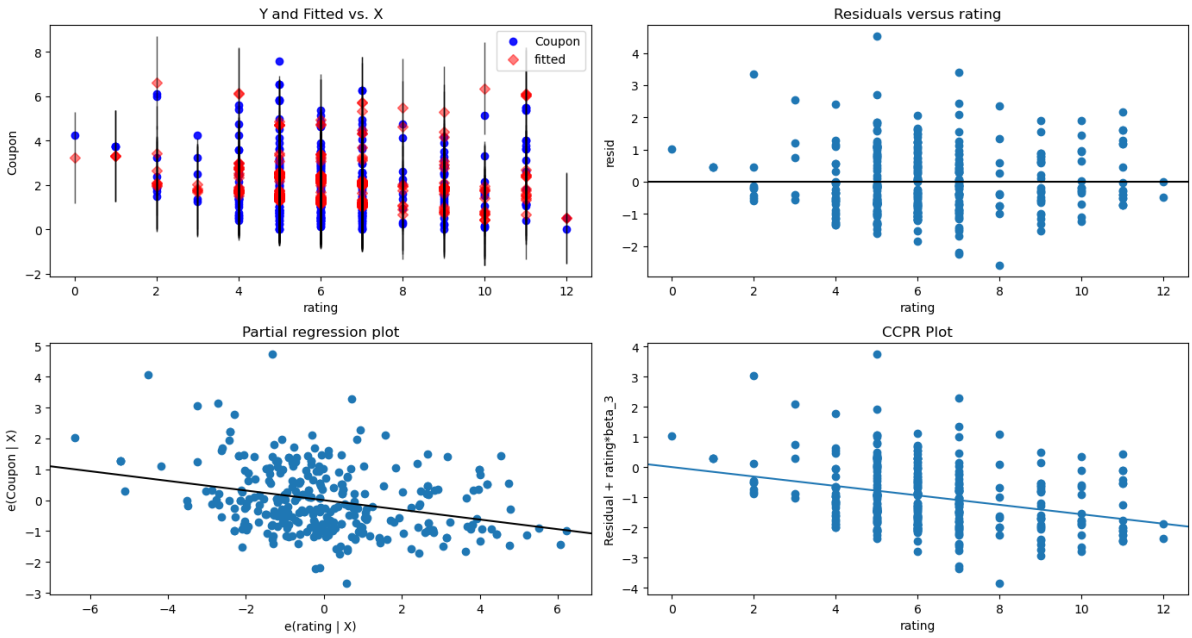
In [ ]:

```
## Show the regression in charts
for x in range(len(factors2)):
    fig = plt.figure(figsize=(14, 8))
    fig = sm.graphics.plot_regress_exog(model2,
                                        factors2[x],
                                        fig=fig)
```

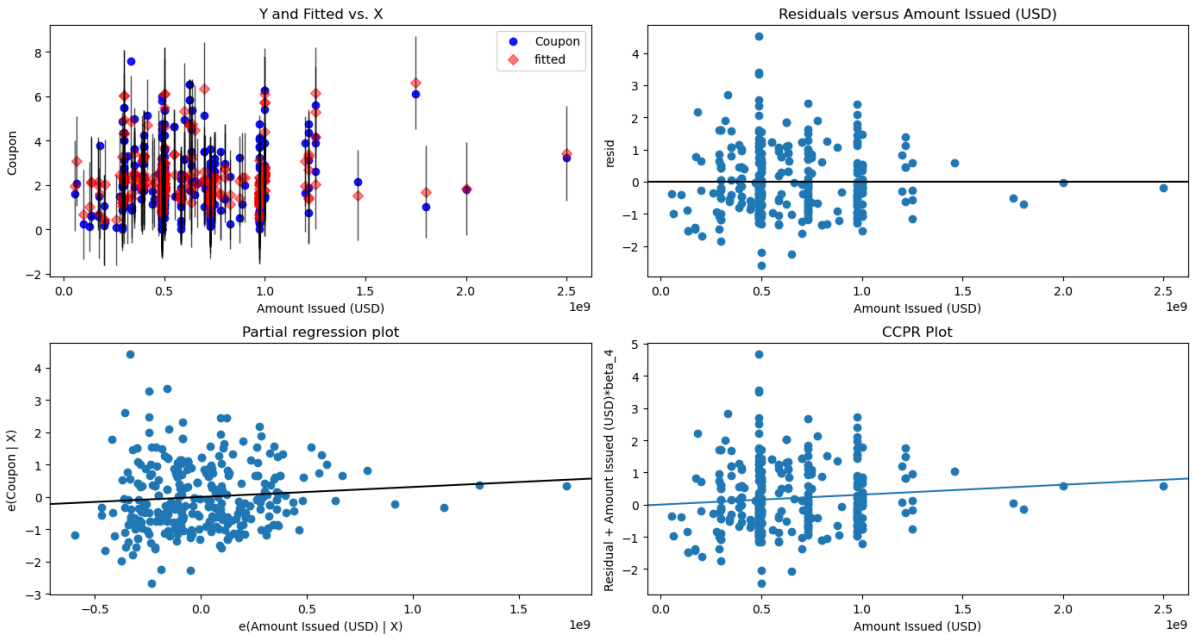
```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```



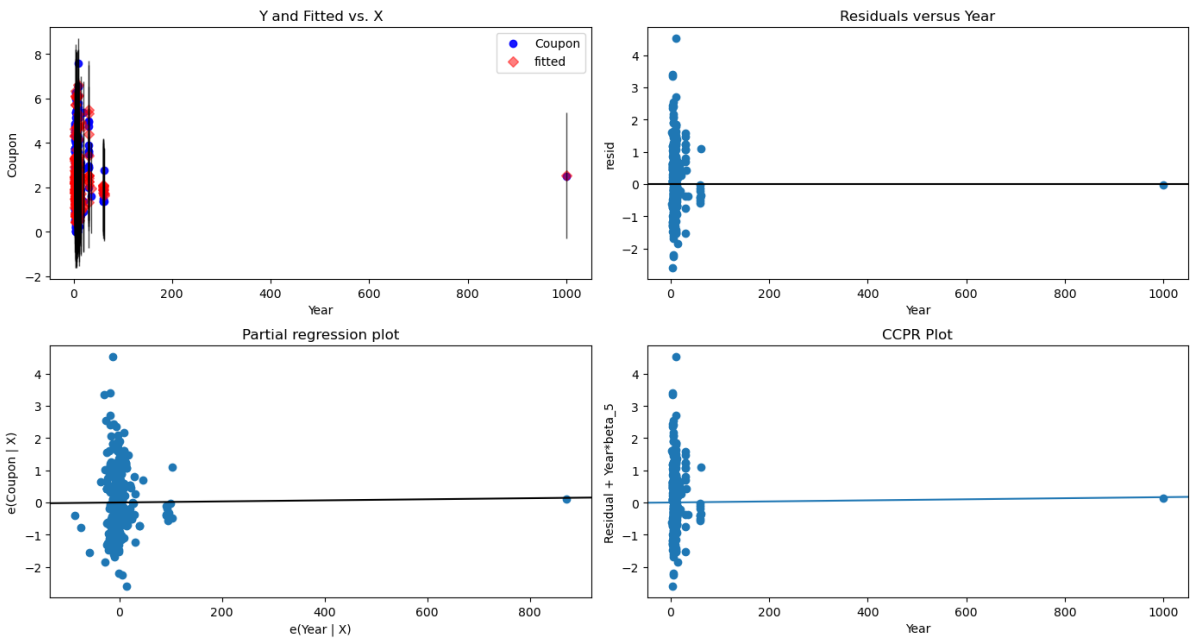
Regression Plots for rating

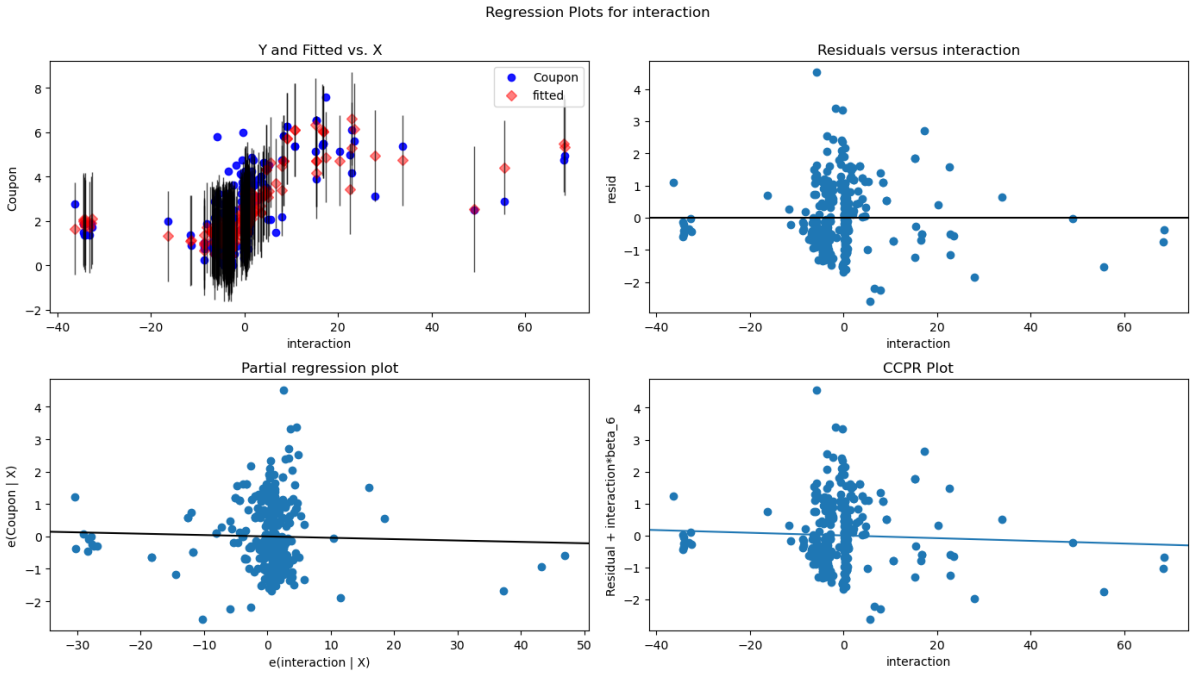


Regression Plots for Amount Issued (USD)



Regression Plots for Year





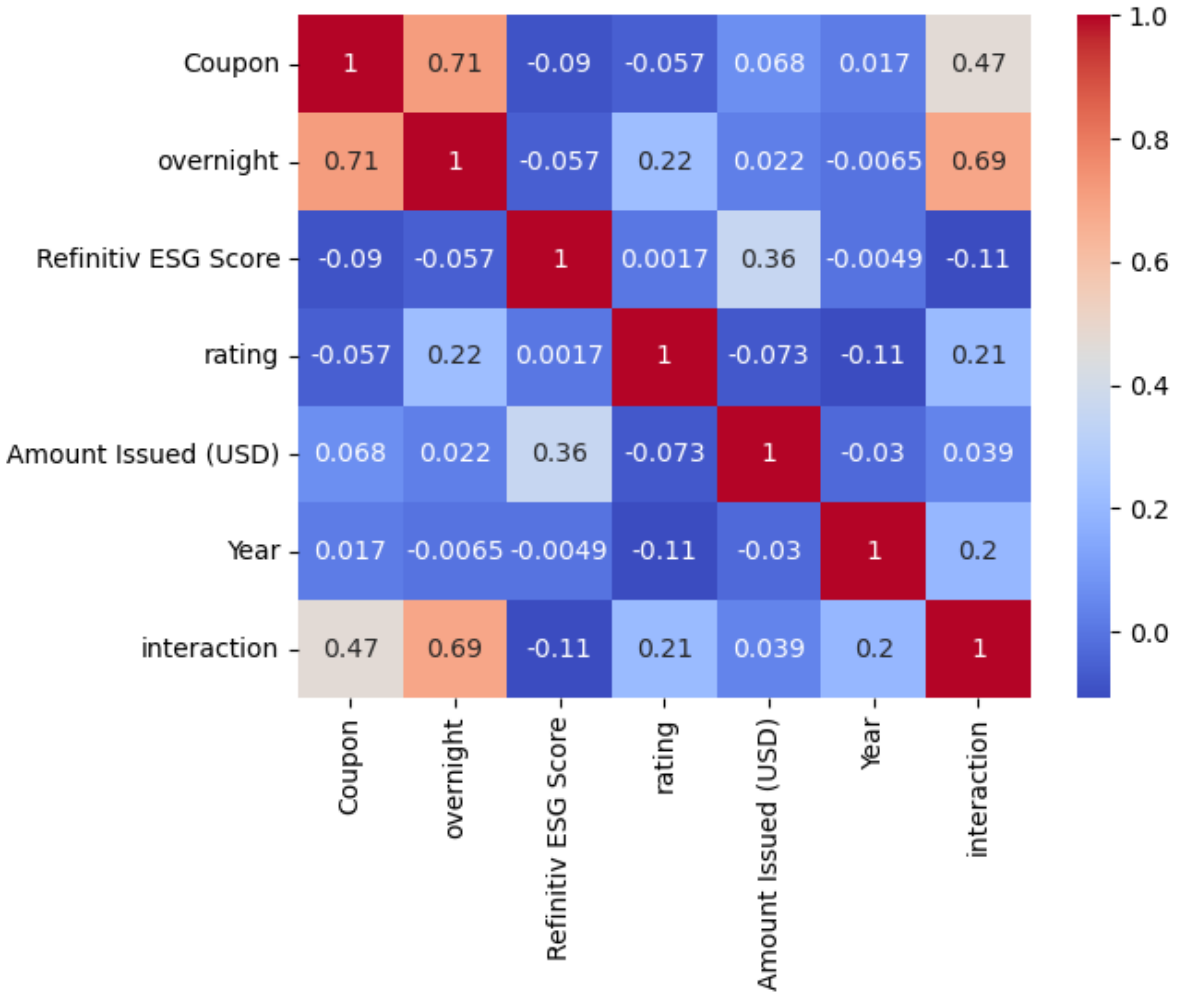
In [ ]:
```
## Anova test
pingouin.anova(data = data_2, dv = "Coupon", between = "Refinitiv ESG Score"
```

Out[ ]:

	Source	ddof1	ddof2	F	p-unc	np2
0	Refinitiv ESG Score	155	169	1.862813	0.000041	0.630792

In [ ]:
```
## Heatmap
sns.heatmap(data_2.corr(), cmap='coolwarm', annot=True)
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:

```
# Test 3_1
## Processing the data
factors3_1 = ["Bloomberg ESG Score", "Credit Rating", "Risk free rate with s

X3 = sm.add_constant(data_3_1[factors3_1])
model3_1 = sm.OLS(data_3_1["Coupon"], X3).fit()
# Fit the model
prediction3_1 = model3_1.predict(X3)
model3_1.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Coupon	**R-squared:**	0.780
Model:	OLS	**Adj. R-squared:**	0.747
Method:	Least Squares	**F-statistic:**	24.05
Date:	Wed, 02 Nov 2022	**Prob (F-statistic):**	2.83e-10
Time:	12:06:14	**Log-Likelihood:**	-45.347
No. Observations:	40	**AIC:**	102.7
Df Residuals:	34	**BIC:**	112.8
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.5290	0.865	1.767	0.086	-0.230	3.288
Bloomberg ESG Score	0.0071	0.015	0.487	0.630	-0.023	0.037
Credit Rating	-0.1530	0.066	-2.310	0.027	-0.288	-0.018
Risk free rate with same tenor	1.0608	0.146	7.247	0.000	0.763	1.358
Tenor	0.0300	0.021	1.403	0.170	-0.013	0.073
Option	-0.1341	0.414	-0.324	0.748	-0.975	0.706

Omnibus:	2.534	**Durbin-Watson:**	1.691
Prob(Omnibus):	0.282	**Jarque-Bera (JB):**	2.359
Skew:	0.558	**Prob(JB):**	0.307
Kurtosis:	2.587	**Cond. No.**	365.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```
## Obtaining the prediction and residual values
result3_1 = pd.concat([prediction3_1, model3_1.resid], axis =1)
result3_1 = result3_1.rename(columns = {0:'prediction', 1:'residual'})
```

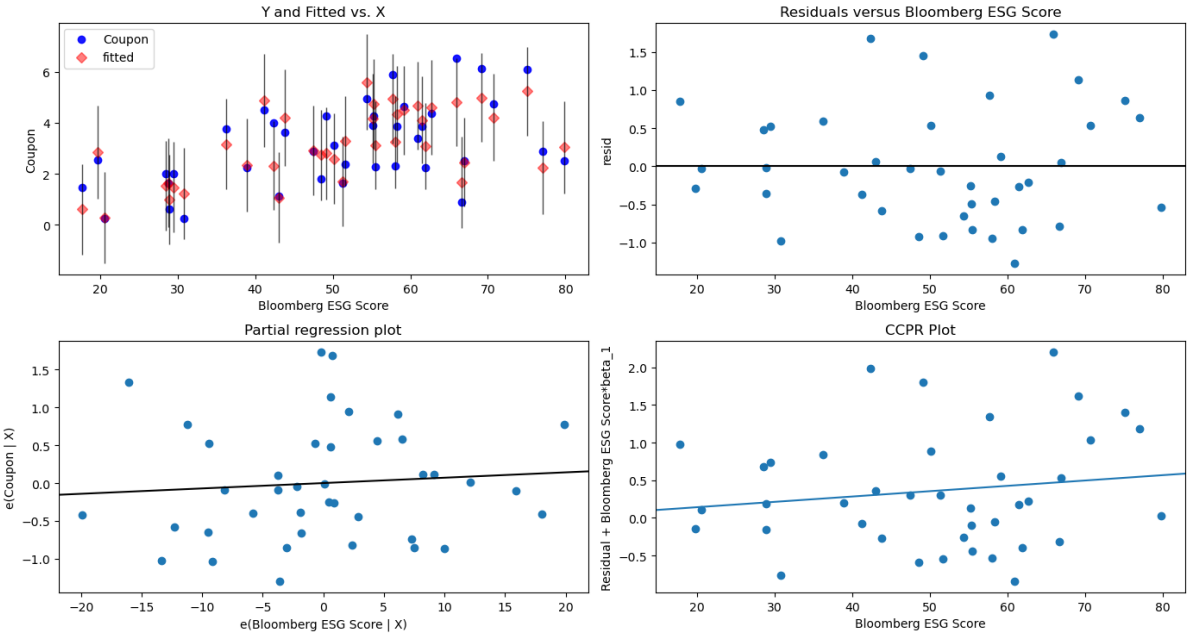
In [ ]:
```
## Print the regression results
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
    print(result3_1)
```

	prediction	residual
0	5.598	-0.648
1	5.236	0.864
2	4.806	1.730
3	4.156	-0.256
4	4.592	-0.217
5	4.945	0.930
6	4.872	-0.372
7	4.741	-0.491
8	4.493	0.132
9	4.992	1.133
10	2.835	-0.285
11	4.336	-0.461
12	2.916	-0.036
13	2.729	-0.929
14	3.283	-0.908
15	3.241	-0.941
16	4.673	-1.273
17	3.108	-0.837
18	4.114	-0.264
19	4.208	-0.583
20	2.326	-0.076
21	3.162	0.588
22	2.236	0.639
23	1.524	0.476
24	3.035	-0.535
25	3.087	-0.837
26	2.797	1.453
27	1.665	-0.790
28	4.215	0.535
29	1.690	-0.065
30	2.320	1.680
31	2.446	0.054
32	2.590	0.535
33	1.639	-0.014
34	0.601	0.852
35	0.285	-0.035
36	1.472	0.528
37	1.066	0.059
38	0.981	-0.356
39	1.226	-0.976

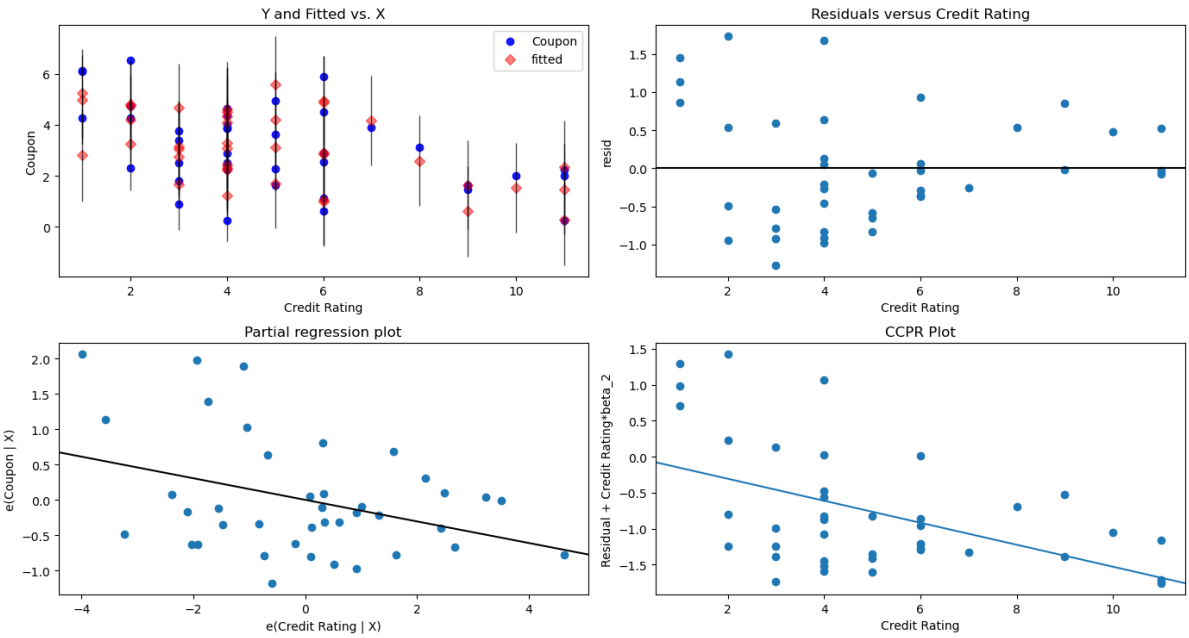
In [ ]:
```
## Show the regression in charts
for x in range(len(factors3_1)):
    fig = plt.figure(figsize=(14, 8))
    fig = sm.graphics.plot_regress_exog(model3_1,
                                        factors3_1[x],
                                        fig=fig)
```

```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```

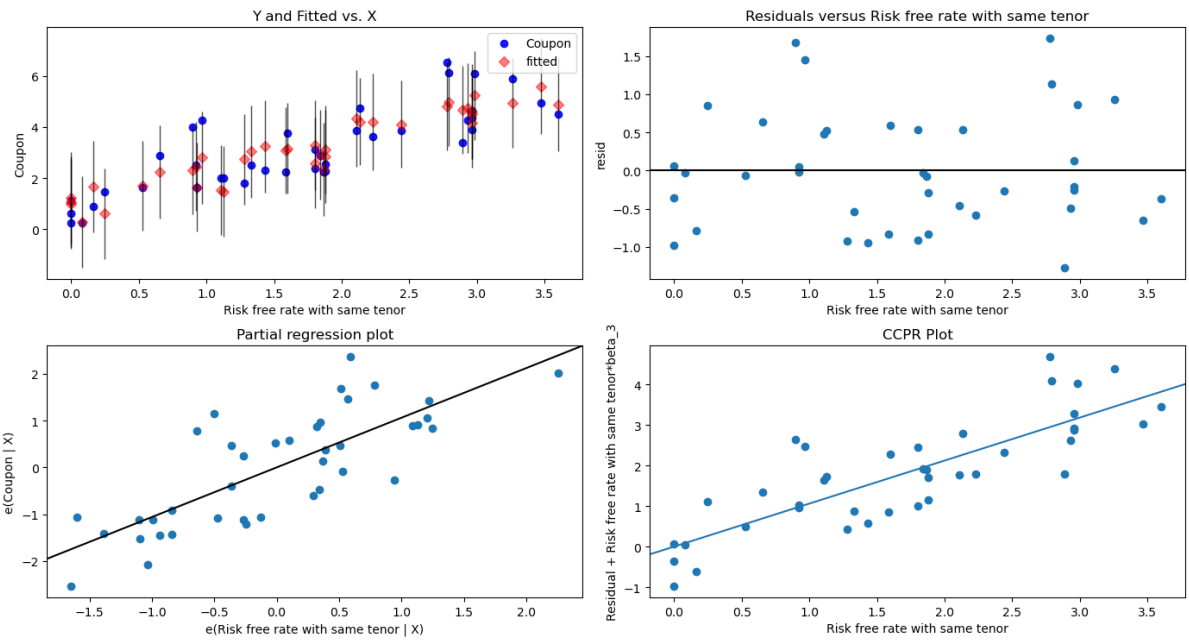
Regression Plots for Bloomberg ESG Score

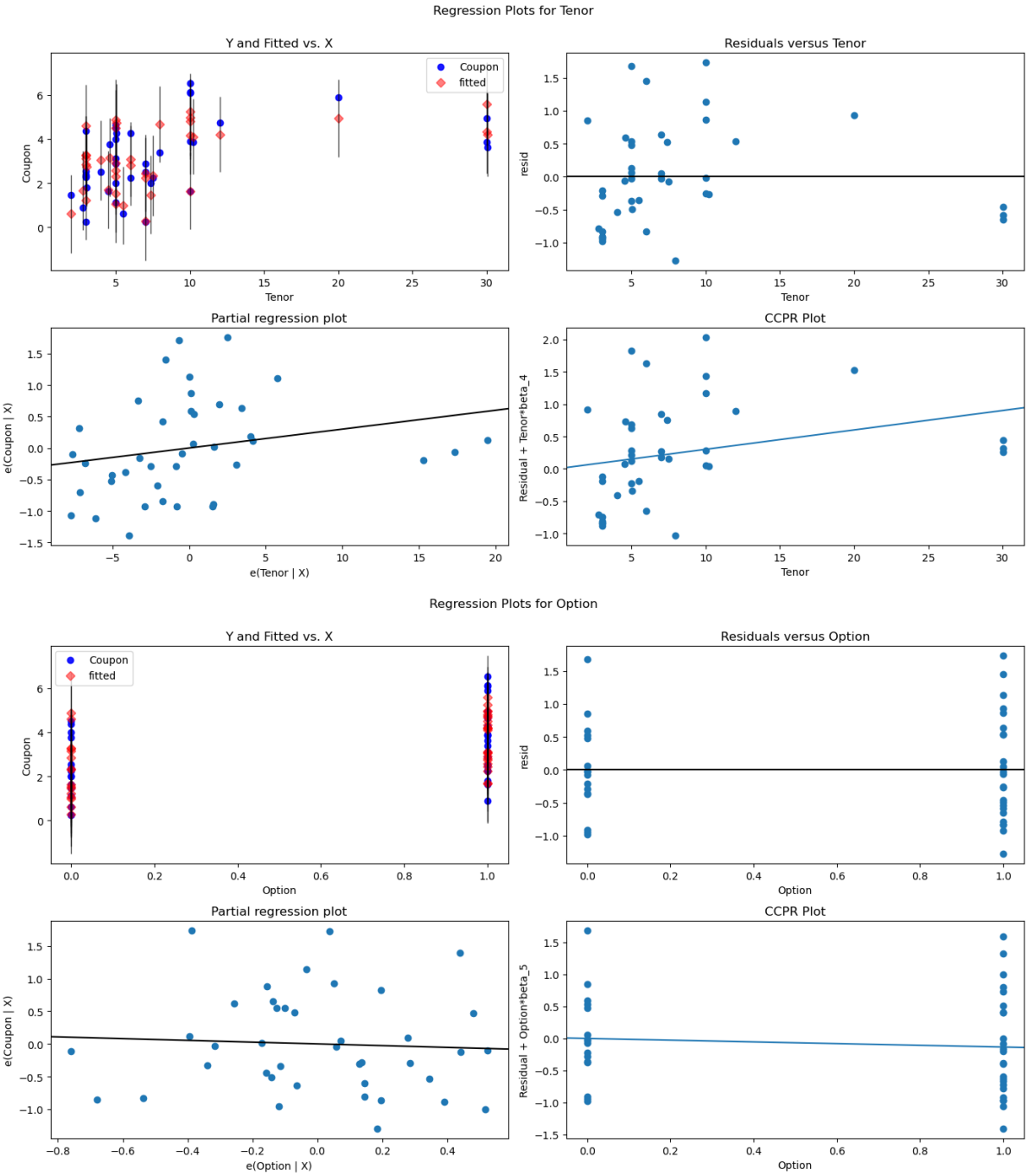


Regression Plots for Credit Rating



Regression Plots for Risk free rate with same tenor





In [ ]:
```
## Anova test
pingouin.anova(data = data_3_1, dv = "Coupon", between = "Bloomberg ESG Scor
```

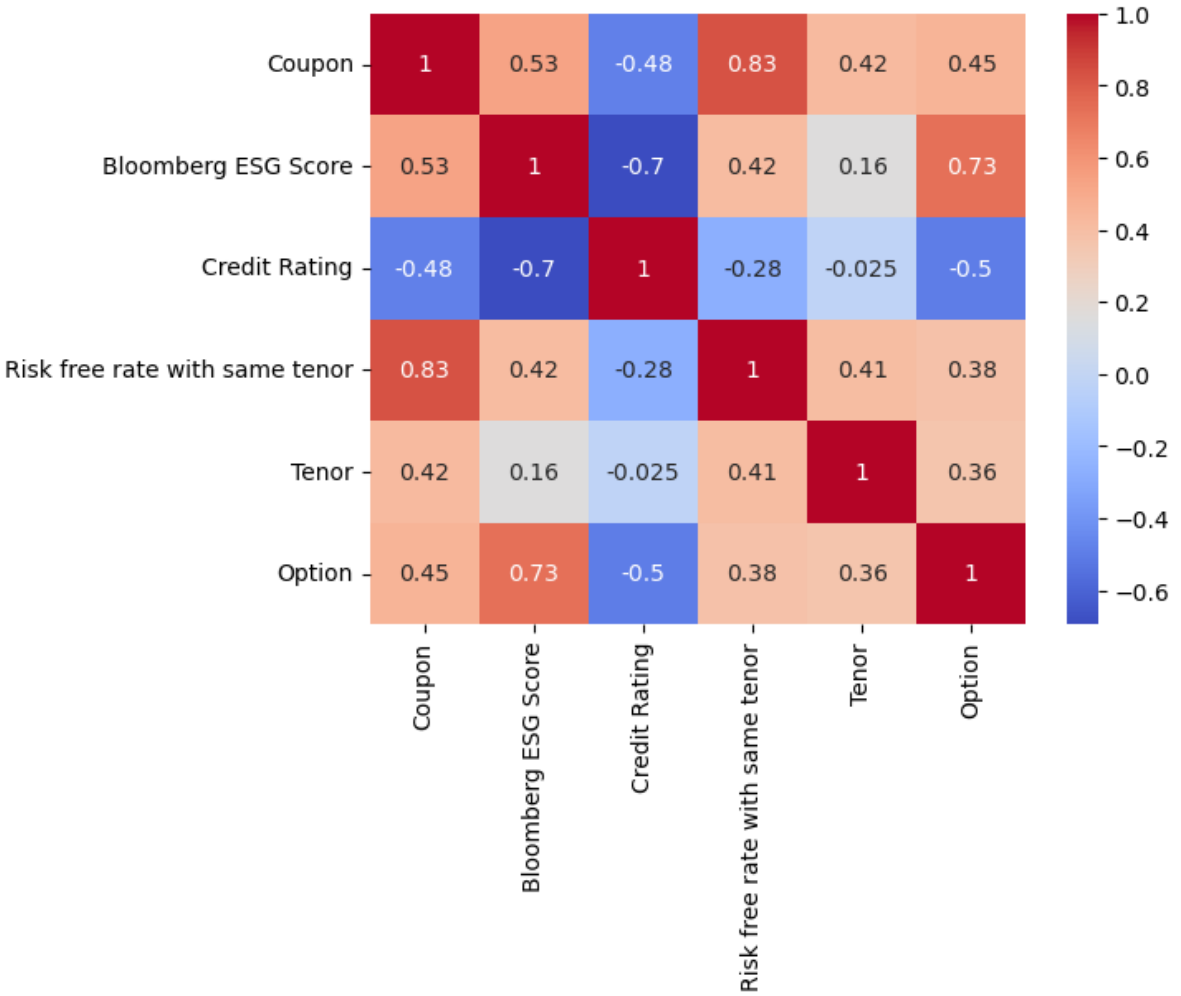
```
/opt/anaconda3/lib/python3.9/site-packages/pingouin/parametric.py:1000: Runt
imeWarning: invalid value encountered in double_scalars
  mserror = sserror / ddof2
```

Out[ ]:

	Source	ddof1	ddof2	np2
0	Bloomberg ESG Score	39	0	1.0

In [ ]:
```
## Heatmaps
sns.heatmap(data_3_1.corr(), cmap='coolwarm', annot=True)
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:

```
# Test 3_2
## Processing the data
factors3_2 = ["Market Average Spread", "Tenor", "Credit rating", "Issuer Typ

X3_2 = sm.add_constant(data_3_2[factors3_2])
model3_2 = sm.OLS(data_3_2["coupon"], X3_2).fit()
# Fit the model
prediction3_2 = model3_2.predict(X3_2)
# Print the parameters of the fitted model
model3_2.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	coupon	**R-squared:**	0.786
Model:	OLS	**Adj. R-squared:**	0.766
Method:	Least Squares	**F-statistic:**	37.57
Date:	Wed, 02 Nov 2022	**Prob (F-statistic):**	5.86e-16
Time:	12:06:18	**Log-Likelihood:**	-8.2004
No. Observations:	57	**AIC:**	28.40
Df Residuals:	51	**BIC:**	40.66
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.3724	0.470	0.792	0.432	-0.572	1.316
Market Average Spread	1.6874	0.218	7.729	0.000	1.249	2.126
Tenor	0.0571	0.057	1.006	0.319	-0.057	0.171
Credit rating	-0.2621	0.045	-5.829	0.000	-0.352	-0.172
Issuer Type	-0.2458	0.043	-5.773	0.000	-0.331	-0.160
Russeel ESG Score	0.0440	0.062	0.707	0.483	-0.081	0.169

Omnibus:	3.346	**Durbin-Watson:**	1.702
Prob(Omnibus):	0.188	**Jarque-Bera (JB):**	2.471
Skew:	0.332	**Prob(JB):**	0.291
Kurtosis:	3.774	**Cond. No.**	85.1

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```
## Obtaining the prediction and residual values
result3_2 = pd.concat([prediction3_2, model3_2.resid], axis =1)
result3_2 = result3_2.rename(columns = {0:'prediction', 1:'residual'})
```

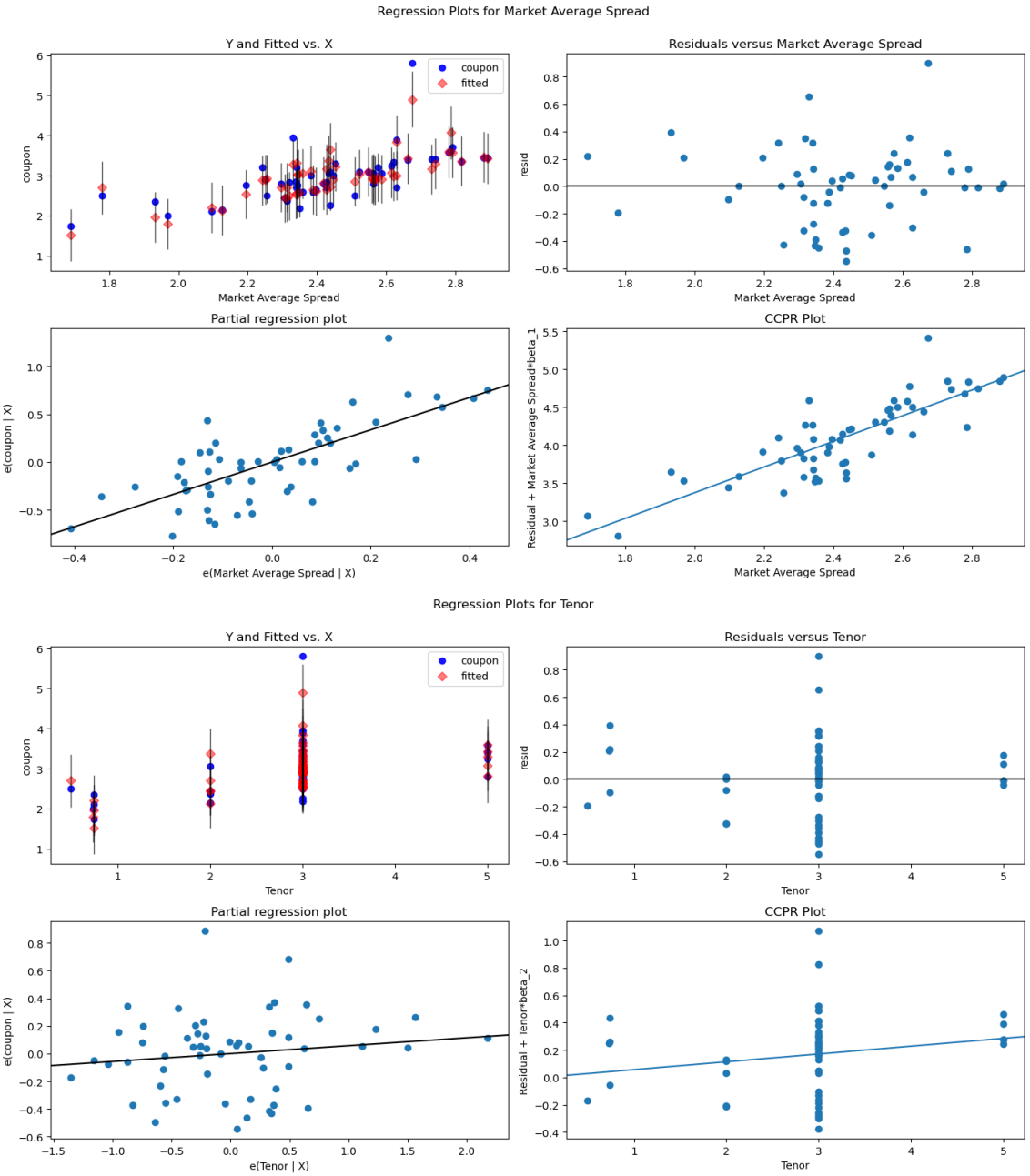
In [ ]:
```
## Print the regression results
with pd.option_context('display.max_rows', None,
                       'display.max_columns', None,
                       'display.precision', 3,
                       ):
    print(result3_2)
```

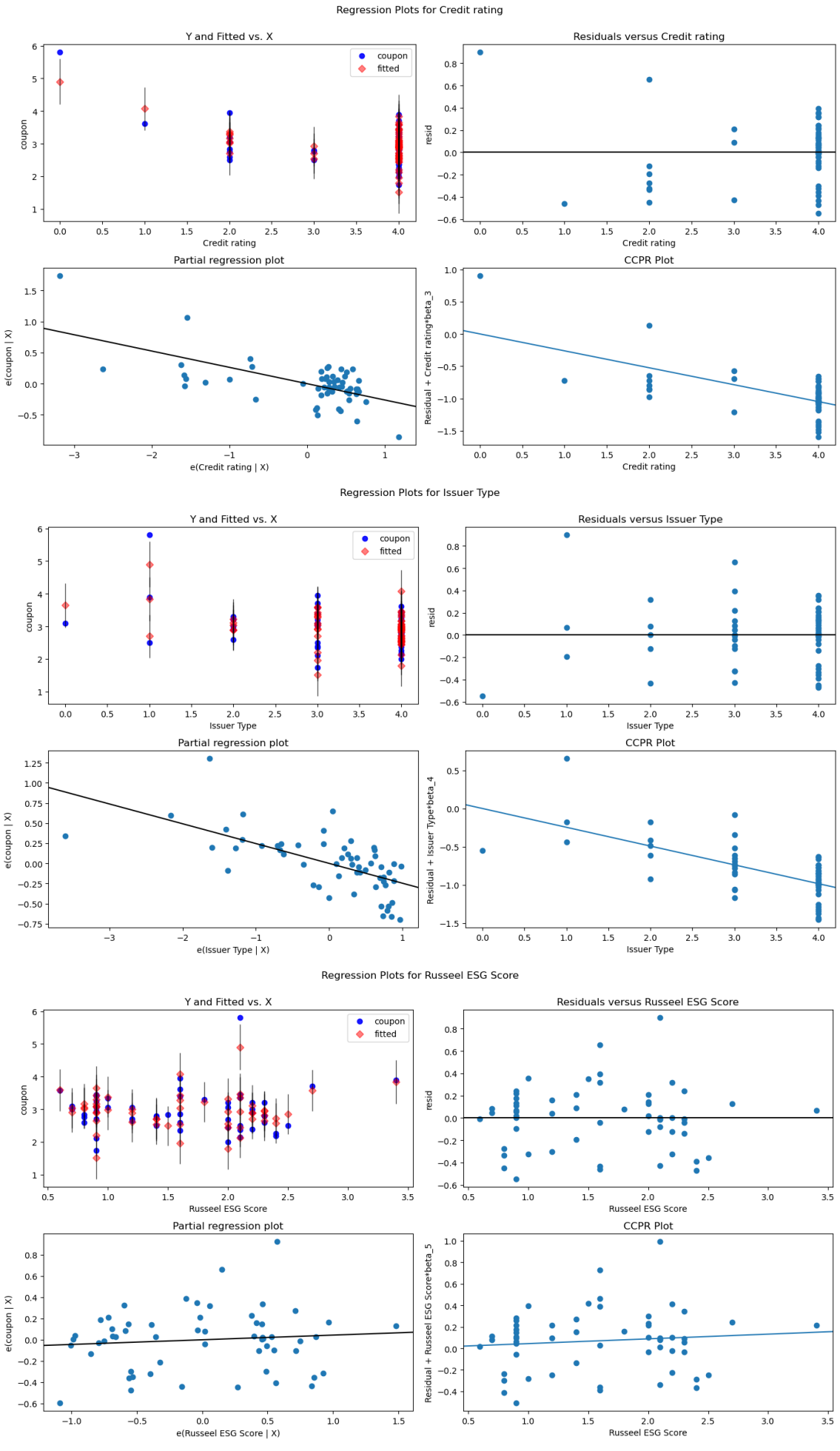
	prediction	residual
0	2.977	0.353
1	2.957	0.243
2	1.791	0.209
3	1.520	0.220
4	2.695	-0.195
5	2.205	-0.095
6	1.959	0.391
7	2.530	0.320
8	2.540	0.210
9	2.708	0.092
10	2.937	-0.137
11	2.642	-0.042
12	3.647	-0.547
13	3.075	0.175
14	2.645	0.055
15	3.292	0.108
16	2.915	0.135
17	2.883	0.067
18	3.159	0.241
19	3.430	0.020
20	2.917	0.143
21	3.358	-0.008
22	3.465	-0.015
23	2.135	0.005
24	2.451	-0.081
25	3.049	-0.449
26	3.023	-0.273
27	3.166	-0.336
28	2.918	-0.428
29	3.321	-0.121
30	2.608	0.042
31	3.001	-0.301
32	2.889	0.161
33	3.834	0.066
34	3.096	0.004
35	4.902	0.898
36	2.916	0.084
37	3.043	0.047
38	3.432	-0.042
39	3.590	-0.010
40	3.221	0.079
41	3.282	0.658
42	2.702	-0.322
43	2.883	0.317
44	2.897	0.003
45	3.120	-0.120
46	2.809	-0.009
47	3.032	-0.432
48	2.490	0.350
49	4.070	-0.460
50	2.551	0.129
51	2.434	0.016
52	2.857	-0.357
53	2.581	-0.391
54	2.729	-0.469
55	3.376	-0.326
56	3.585	0.125

In [ ]:
```
## Show the regression in charts
for x in range(len(factors3_2)):
    fig = plt.figure(figsize=(14, 8))
    fig = sm.graphics.plot_regress_exog(model3_2,
```

```
                                factors3_2[x],
                                fig=fig)
```

```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```





In [ ]:
```
## Anova test
print(pingouin.anova(data = data_3_2, dv = "coupon", between = "Russeel ESG
print(pingouin.anova(data = data_3_2, dv = "coupon", between = "Issuer Type"
```

```
              Source  ddof1  ddof2         F     p-unc       np2
0  Russeel ESG Score     17     39  0.733641  0.750581  0.242305
        Source  ddof1  ddof2         F     p-unc       np2
0  Issuer Type      4     52  3.218563  0.019578  0.198449
```

In [ ]:
```
## Heatmaps
sns.heatmap(data_3_2.corr(), cmap='coolwarm', annot=True)
```

Out[ ]:
```
<AxesSubplot:>
```