Hasan Haq

A Fresh Look at Comps

What are comps?

Comparable Company Analysis	Summary				P/E			EV/EBITDA		
	Price	Shares	Mkt Cap	EV	2010A	2011E	2012E	2010A	2011E	2012E
Company Comps										
Company A	$16.95	60.2	$1,032	$1,805	13.6x	11.7x	10.9x	4.4x	3.8x	3.7x
Company B	$5.26	1,730.2	$9,257	$19,074	15.8x	15.2x	14.5x	4.4x	4.6x	4.7x
Company C	$7.57	352.7	$2,635	$5,190	18.5x	13.8x	11.1x	5.2x	4.5x	4.2x
Company D	$11.07	456.6	$5,014	$10,033	12.8x	12.6x	13.5x	5.7x	5.8x	6.1x
Company E	$17.88	77.5	$1,398	$3,611	14.0x	13.2x	12.1x	5.9x	4.9x	4.3x
Company F	$15.03	30.2	$422	$417	10.4x	10.0x	8.6x	6.0x	5.4x	4.9x
Company G	$3.28	201.1	$640	$2,777	7.7x	7.5x	6.9x	6.0x	6.1x	6.1x
Company H	$1.49	63.1	$95	$78	17.5x	12.4x	9.9x	8.9x	8.0x	6.2x
Company I	$1.60	200.0	$318	$470	10.7x	8.9x	7.3x	7.9x	6.9x	6.5x
Median					**13.6x**	**12.4x**	**10.9x**	**5.9x**	**5.4x**	**4.9x**
Mean					**13.4x**	**11.7x**	**10.5x**	**6.1x**	**5.6x**	**5.2x**
Min					7.7x	7.5x	6.9x	4.4x	3.8x	3.7x
Max					18.5x	15.2x	14.5x	8.9x	8.0x	6.5x

Source: Company Filings, Bloomberg Consensus Estimates

Yeah…

The Gameplan…

- Target: EV / EBITDA Multiples
- Features: Every financial metric available
- The Tools:
  - BeautifulSoup / ~~Selenium~~
  - Statsmodels / Sklearn

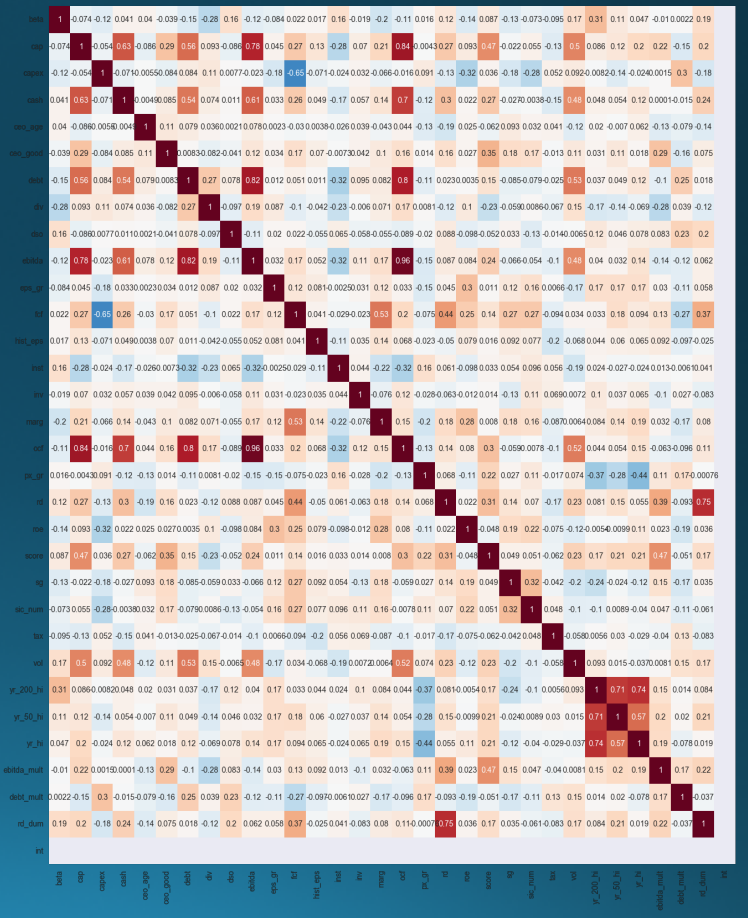
Scraping ain't easy…

- S&P 500
  - (and Russell 1000)
- WSJ Website
- SEC Website
- Reuters Website
- Morningstar API
- Yahoo Finance API

```
126     except:
127         inst = 0
128     return (beta, inst)
129
130 # Scrape WSJ using BeautifulSoup
131 def wsj(sym):
132     ''' Scrape for EBITDA, Cash Balance, and Debt '''
133     wsj_url_is = urlopen('http://quotes.wsj.com/' + sym +
134                          '/financials/annual/income-statement').read()
135     wsj_url_bs = urlopen('http://quotes.wsj.com/' + sym +
136                          '/financials/annual/balance-sheet').read()
137     soup = bs(wsj_url_is, 'lxml')
138     search_str = re.compile(r'EBITDA')
139     ebitda = intna(soup.find('td', text=search_str).find_next_sibling().string)
140
141     soup = bs(wsj_url_bs, 'lxml')
142     search_str = re.compile(r'Cash &')
143     cash = intna(soup.find('td', text=search_str).find_next_sibling().string)
144
145     search_str = re.compile(r'Long-Term Debt')
146     lt_debt = intna(soup.find('td', text=search_str).find_next_sibling().string)
147
148     search_str = re.compile(r'ST Debt')
149     st_debt =  intna(soup.find('td', text=search_str).find_next_sibling().string)
150     debt = (lt_debt + st_debt)
151     return (cash, debt, ebitda)
```

Correlation Contemplation

- 32 Features
- A few standout features
  - Analyst research "score"
  - R&D variable
  - "CEO Good
- Dummies not yet included



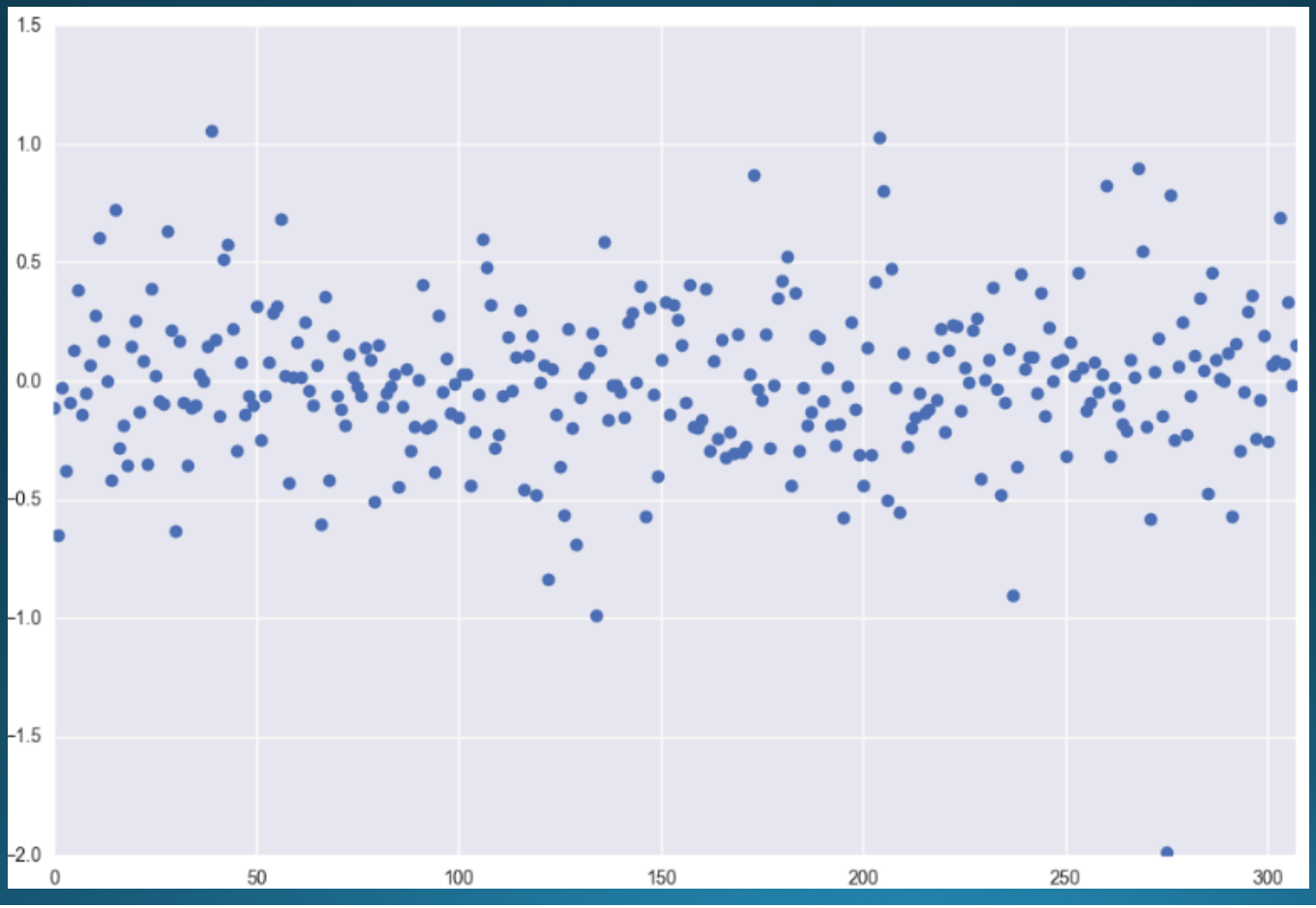
As Good As It's Gonna Get… (?)

Dep. Variable:	ebitda_mult	**R-squared:**	0.616
Model:	OLS	**Adj. R-squared:**	0.591
Method:	Least Squares	**F-statistic:**	24.30
Date:	Fri, 03 Feb 2017	**Prob (F-statistic):**	6.08e-49
Time:	08:03:20	**Log-Likelihood:**	-94.658
No. Observations:	308	**AIC:**	229.3
Df Residuals:	288	**BIC:**	303.9
Df Model:	19		
Covariance Type:	nonrobust		

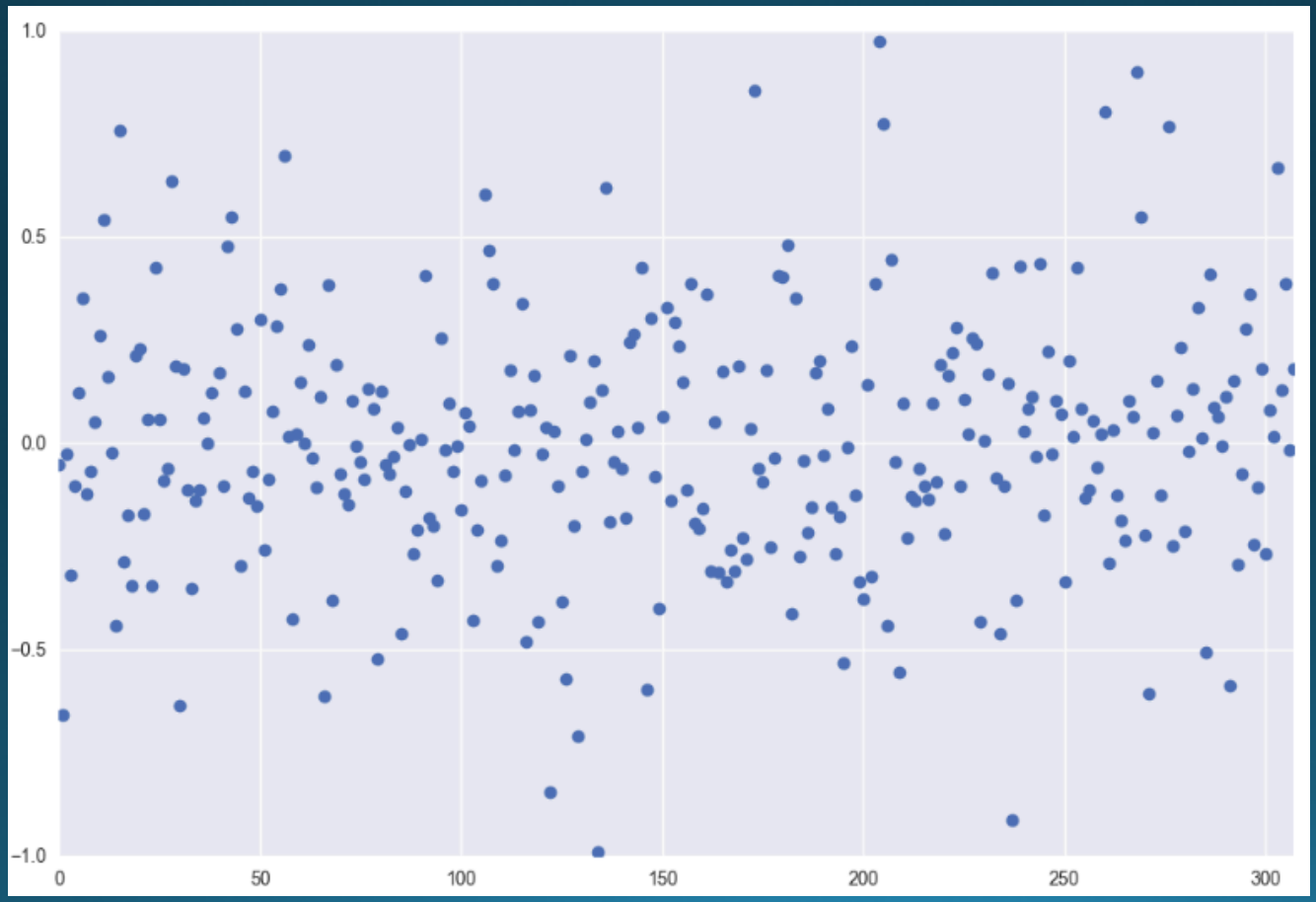
	coef	std err	t	P>|t|	[95.0% Conf. Int.]
beta	-0.2454	0.050	-4.922	0.000	-0.344 -0.147
ceo_good	0.2200	0.074	2.989	0.003	0.075 0.365
div	-0.0462	0.015	-3.059	0.002	-0.076 -0.016
marg	0.0044	0.002	2.094	0.037	0.000 0.009
rd	0.0220	0.006	3.766	0.000	0.011 0.034
score	0.0129	0.003	4.612	0.000	0.007 0.018
sg	0.0069	0.002	3.840	0.000	0.003 0.010
vol	-0.0115	0.004	-3.029	0.003	-0.019 -0.004
yr_hi	0.0124	0.002	6.508	0.000	0.009 0.016
debt_mult	0.1062	0.012	9.006	0.000	0.083 0.129
rd_dum	-0.0893	0.070	-1.268	0.206	-0.228 0.049
int	-0.8811	0.263	-3.355	0.001	-1.398 -0.364

Sectors (not shown) matter a lot!

Ummm...

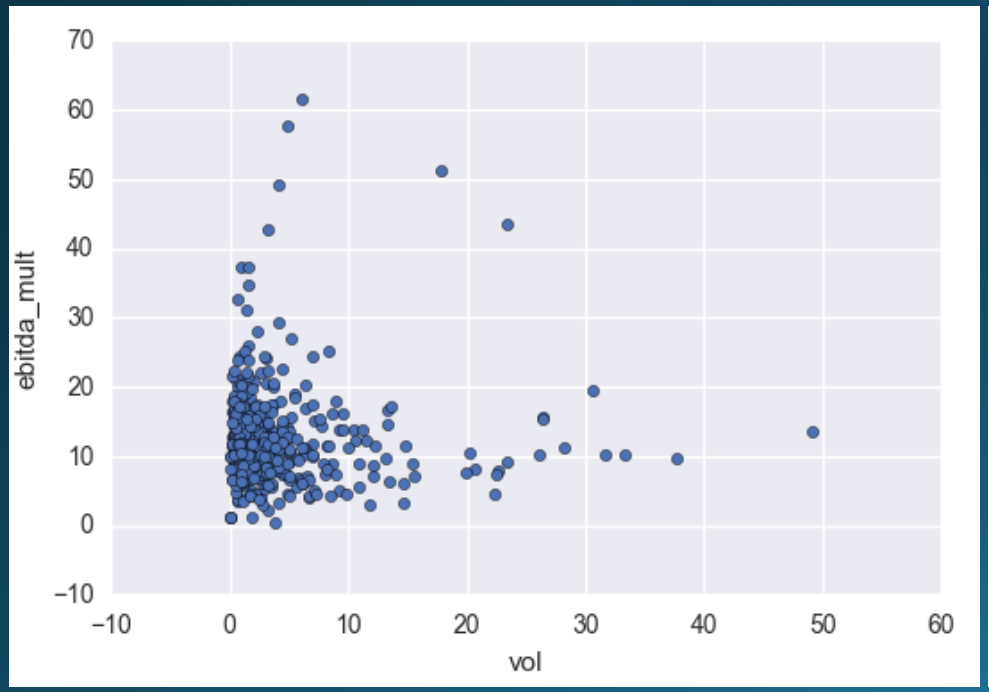


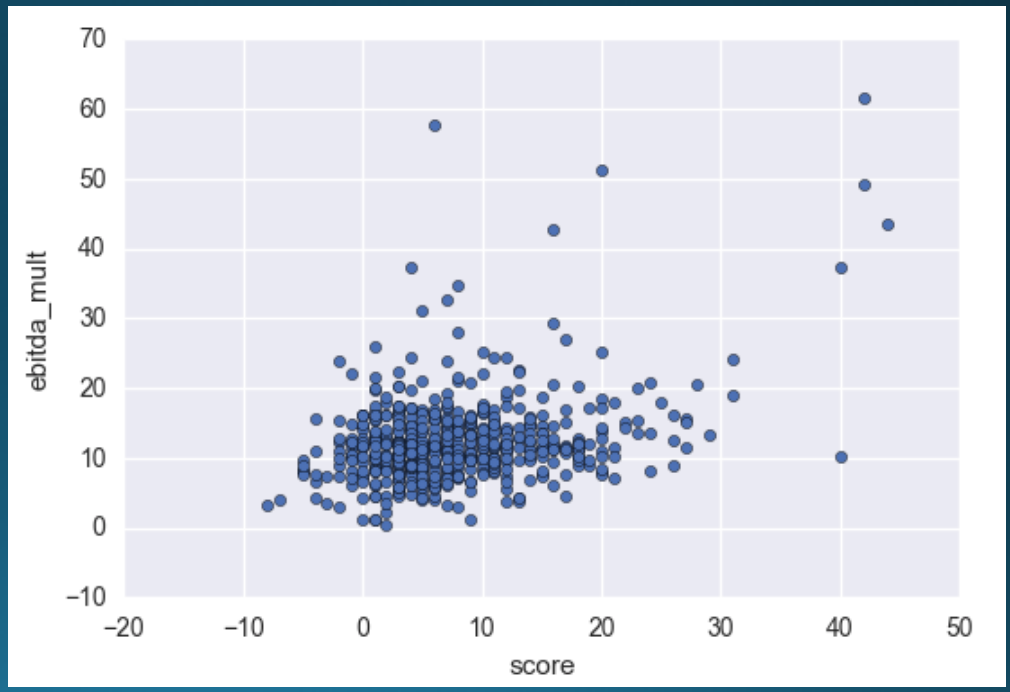
That's Better!



Pair Plots

- Very small improvement with polynomial features





Cross-Validation (Linear)

Cross-Validate with Sklearn (Linear)

In [216]:
```
lr1 = make_pipeline(PolynomialFeatures(1), LinearRegression())
lr1.fit(x, y)
lr1.score(x, y)
```

Out[216]:
```
0.63095702747853688
```

In [217]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
lr1.fit(X_train, y_train)
lr1.score(X_test, y_test)
```

Out[217]:
```
0.52334090234739428
```

In [209]:
```
reg = LinearRegression()
scores = cross_val_score(reg, x, y, cv=10, scoring='mean_squared_error')
print(-scores)
```

```
[ 0.83544012  0.49728468  0.06417079  0.10950318  0.12854397  0.11121793
  0.12784506  0.0854312   0.27630417  0.08385438]
```

LASSO!

Lasso Setup

In [222]:
```
Y=df_cleandum['ebitda_mult']
X=df_cleandum.drop(['cap', 'cash', 'debt', 'ebitda', 'ocf', 'ticker', 'ebitda_mult'],1)
```

In [224]:
```
las = make_pipeline(PolynomialFeatures(1), Lasso(alpha=1))
las.fit(X, Y)
las.score(X, Y)
```

Out[224]: 0.41658985870286475

In [225]:
```
las = make_pipeline(PolynomialFeatures(1), Lasso(alpha=.1))
las.fit(X, Y)
las.score(X, Y)
```

Out[225]: 0.49560066593747376

In [226]:
```
Ylog = np.log(Y)
las = make_pipeline(PolynomialFeatures(1), Lasso(alpha=1))
las.fit(X, Ylog)
las.score(X, Ylog)
```

Out[226]: 0.12200277880805098

In [227]:
```
las = make_pipeline(PolynomialFeatures(1), Lasso(alpha=.1))
las.fit(X, Ylog)
las.score(X, Ylog)
```

Out[227]: 0.31631275249073976

Takeaways

- The tried and true method is largely correct
- Should look incorporate analyst research
- Companies with R&D spending should be bucketed together
- Companies with notable CEOs/management should be bucketed together

For Future Versions…

- Scrub the Russell 1000 dataset
- Focus on one industry
- Evaluate more polynomial combinations
  - Look at more pairplots
- Lasso and Ridge