Project 4: Multi-factor Model

Instructions

Each problem consists of a function to implement and instructions on how to implement the function. The parts of the function that need to be implemented are marked with a `# TODO` comment. After implementing the function, run the cell to test it against the unit tests we've provided. For each problem, we provide one or more unit tests from our `project_tests` package. These unit tests won't tell you if your answer is correct, but will warn you of any major errors. Your code will be checked for the correct solution when you submit it to Udacity.

Packages

When you implement the functions, you'll only need to you use the packages you've used in the classroom, like Pandas and Numpy. These packages will be imported for you. We recommend you don't add any import statements, otherwise the grader might not be able to run your code.

The other packages that we're importing are `project_helper` and `project_tests`. These are custom packages built to help you solve the problems. The `project_helper` module contains utility functions and graph functions. The `project_tests` contains the unit tests for all the problems.

Install Packages

In [2]:
```
import sys
!{sys.executable} -m pip install -r requirements.txt
```

```
Collecting alphalens==0.3.2 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/a5/dc/2f9cd107d0d4c
f6223d37d81ddfbbdbf0d703d03669b83810fa6b97f32e5/alphalens-0.3.2.tar.gz (1
8.9MB)
    100% |████████████████████████████████| 18.9MB 24kB/s  eta 0:00:01
32% |██████████                          | 6.1MB 29.8MB/s eta 0:00:01    46%
|███████████████                         | 8.8MB 23.1MB/s eta 0:00:01    59% |███
████████████████                     | 11.2MB 23.2MB/s eta 0:00:01    73% |██████
██████████████████                  | 13.8MB 29.5MB/s eta 0:00:01    80% |█████████
█████████████████               | 15.2MB 28.4MB/s eta 0:00:01    87% |████████████
████████████████              | 16.5MB 24.7MB/s eta 0:00:01    94% |███████████████
███████████████               | 17.8MB 29.2MB/s eta 0:00:01
Requirement already satisfied: colour==0.1.5 in /opt/conda/lib/python3.6/
site-packages (from -r requirements.txt (line 2))
Collecting cvxpy==1.0.3 (from -r requirements.txt (line 3))
```

```
  Downloading https://files.pythonhosted.org/packages/a1/59/2613468ffbbe3
a818934d06b81b9f4877fe054afbf4f99d2f43f398a0b34/cvxpy-1.0.3.tar.gz (880k
B)
    100% |████████████████████████████████| 880kB 526kB/s eta 0:00:01
Requirement already satisfied: cycler==0.10.0 in /opt/conda/lib/python3.
6/site-packages/cycler-0.10.0-py3.6.egg (from -r requirements.txt (line
4))
Collecting numpy==1.13.3 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/57/a7/e3e6bd9d59512
5e1abbe162e323fd2d06f6f6683185294b79cd2cdb190d5/numpy-1.13.3-cp36-cp36m-m
anylinux1_x86_64.whl (17.0MB)
    100% |████████████████████████████████| 17.0MB 27kB/s  eta 0:00:01
14% |████▌                           | 2.4MB 27.1MB/s eta 0:00:01    21%
|███████                         | 3.7MB 24.5MB/s eta 0:00:01    64% |███
█████████████████▊           | 10.9MB 24.6MB/s eta 0:00:01    77% |██████
███████████████████       | 13.2MB 22.6MB/s eta 0:00:01    91% |█████████
████████████████████▎  | 15.6MB 25.6MB/s eta 0:00:01    98% |████████████
███████████████████▋| 16.8MB 25.5MB/s eta 0:00:01
Collecting pandas==0.18.1 (from -r requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/11/09/e66eb844daba8
680ddff26335d5b4fead77f60f957678243549a8dd4830d/pandas-0.18.1.tar.gz (7.3
MB)
    100% |████████████████████████████████| 7.3MB 62kB/s  eta 0:00:01
34% |███████████                     | 2.5MB 13.3MB/s eta 0:00:01    53%
|█████████████████               | 3.9MB 22.5MB/s eta 0:00:01
Collecting plotly==2.2.3 (from -r requirements.txt (line 7))
  Downloading https://files.pythonhosted.org/packages/99/a6/8214b6564bf4a
ce9bec8a26e7f89832792be582c042c47c912d3201328a0/plotly-2.2.3.tar.gz (1.1M
B)
    100% |████████████████████████████████| 1.1MB 424kB/s eta 0:00:01
12% |████                            | 133kB 23.7MB/s eta 0:00:01
Requirement already satisfied: pyparsing==2.2.0 in /opt/conda/lib/python
3.6/site-packages (from -r requirements.txt (line 8))
Requirement already satisfied: python-dateutil==2.6.1 in /opt/conda/lib/p
ython3.6/site-packages (from -r requirements.txt (line 9))
Requirement already satisfied: pytz==2017.3 in /opt/conda/lib/python3.6/s
ite-packages (from -r requirements.txt (line 10))
Requirement already satisfied: requests==2.18.4 in /opt/conda/lib/python
3.6/site-packages (from -r requirements.txt (line 11))
Collecting scipy==1.0.0 (from -r requirements.txt (line 12))
  Downloading https://files.pythonhosted.org/packages/d8/5e/caa01ba7be116
00b6a9d39265440d7b3be3d69206da887c42bef049521f2/scipy-1.0.0-cp36-cp36m-ma
nylinux1_x86_64.whl (50.0MB)
    100% |████████████████████████████████| 50.0MB 9.2kB/s eta 0:00:01
0% |▎                               | 460kB 23.4MB/s eta 0:00:03    5% |█
▋                              | 2.6MB 23.3MB/s eta 0:00:03    7% |██▍
| 3.7MB 19.8MB/s eta 0:00:03    17% |█████▋                          | 8.
8MB 18.8MB/s eta 0:00:03    21% |███████                         | 10.8MB
20.6MB/s eta 0:00:02    25% |████████                        | 12.8MB 20.
2MB/s eta 0:00:02    27% |█████████                       | 13.9MB 21.5M
B/s eta 0:00:02    31% |██████████                      | 15.7MB 24.3MB/s
eta 0:00:02    33% |██████████▋                     | 16.6MB 17.9MB/s eta
0:00:02    35% |███████████▎                    | 17.7MB 18.5MB/s eta 0:0
0:02    37% |████████████                    | 18.7MB 20.8MB/s eta 0:00:0
2    39% |████████████▌                   | 19.6MB 18.7MB/s eta 0:00:02
45% |██████████████▌                 | 22.6MB 23.8MB/s eta 0:00:02    47%
|███████████████                 | 23.7MB 20.6MB/s eta 0:00:02    49% |██
```

```
██████████████               | 24.7MB 26.5MB/s eta 0:00:01    51% |█████
████████████               | 25.8MB 24.4MB/s eta 0:00:01    53% |████████
█████████                | 26.8MB 20.9MB/s eta 0:00:02    57% |███████████
███████                | 28.7MB 20.2MB/s eta 0:00:02    59% |█████████████
█████                | 29.6MB 23.5MB/s eta 0:00:01    64% |█████████████████
████             | 32.5MB 20.6MB/s eta 0:00:01    67% |████████████████████
██            | 33.6MB 22.0MB/s eta 0:00:01    69% |██████████████████████
| 34.6MB 22.1MB/s eta 0:00:01    69% |██████████████████████          | 3
4.6MB 1.7MB/s eta 0:00:09    71% |███████████████████████          | 35.6M
B 22.6MB/s eta 0:00:01    78% |█████████████████████████        | 39.4MB 2
2.0MB/s eta 0:00:01    81% |██████████████████████████      | 40.5MB 23.6
MB/s eta 0:00:01    83% |██████████████████████████      | 41.7MB 28.6MB/
s eta 0:00:01    87% |████████████████████████████    | 43.6MB 24.1MB/s e
ta 0:00:01    89% |█████████████████████████████   | 44.8MB 25.5MB/s eta
0:00:01    91% |█████████████████████████████   | 46.0MB 22.9MB/s eta 0:0
0:01    95% |██████████████████████████████  | 48.0MB 22.2MB/s eta 0:00:0
1    98% |███████████████████████████████ | 49.1MB 22.6MB/s eta 0:00:01
Requirement already satisfied: scikit-learn==0.19.1 in /opt/conda/lib/pyt
hon3.6/site-packages (from -r requirements.txt (line 13))
Requirement already satisfied: six==1.11.0 in /opt/conda/lib/python3.6/si
te-packages (from -r requirements.txt (line 14))
Collecting tables==3.3.0 (from -r requirements.txt (line 15))
  Downloading https://files.pythonhosted.org/packages/09/e7/72ca83c7bd75d
b94c23fcaf58debe1be5e9842376c630793e23765cab44b/tables-3.3.0-cp36-cp36m-m
anylinux1_x86_64.whl (4.6MB)
    100% |████████████████████████████████| 4.6MB 102kB/s eta 0:00:01
16% |█████                           | 757kB 18.7MB/s eta 0:00:01    65%
|█████████████████████           | 3.0MB 27.4MB/s eta 0:00:01
Collecting tqdm==4.19.5 (from -r requirements.txt (line 16))
  Downloading https://files.pythonhosted.org/packages/71/3c/341b4fa23cb3a
bc335207dba057c790f3bb329f6757e1fcd5d347bcf8308/tqdm-4.19.5-py2.py3-none-
any.whl (51kB)
    100% |████████████████████████████████| 61kB 3.0MB/s ta 0:00:01
Collecting zipline===1.2.0 (from -r requirements.txt (line 17))
  Downloading https://files.pythonhosted.org/packages/15/d3/689f2a940478b
82ac57c751a40460598221fd82b0449a7a8f7eef47a3bcc/zipline-1.2.0.tar.gz (659
kB)
    100% |████████████████████████████████| 665kB 629kB/s eta 0:00:01
Requirement already satisfied: matplotlib>=1.4.0 in /opt/conda/lib/python
3.6/site-packages (from alphalens==0.3.2->-r requirements.txt (line 1))
Requirement already satisfied: seaborn>=0.6.0 in /opt/conda/lib/python3.
6/site-packages (from alphalens==0.3.2->-r requirements.txt (line 1))
Requirement already satisfied: statsmodels>=0.6.1 in /opt/conda/lib/pytho
n3.6/site-packages (from alphalens==0.3.2->-r requirements.txt (line 1))
Requirement already satisfied: IPython>=3.2.3 in /opt/conda/lib/python3.
6/site-packages (from alphalens==0.3.2->-r requirements.txt (line 1))
Collecting osqp (from cvxpy==1.0.3->-r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/c0/01/8becb29b0d38e
0c40eab9e3d54aa8138fa62a010d519caf65e9210021bd3/osqp-0.5.0-cp36-cp36m-man
ylinux1_x86_64.whl (147kB)
    100% |████████████████████████████████| 153kB 2.4MB/s eta 0:00:01
Collecting ecos>=2 (from cvxpy==1.0.3->-r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/55/ed/d131ff51f3a8f
73420eb1191345eb49f269f23cadef515172e356018cde3/ecos-2.0.7.post1-cp36-cp3
6m-manylinux1_x86_64.whl (147kB)
    100% |████████████████████████████████| 153kB 2.2MB/s ta 0:00:01   4
1% |█████████████                   | 61kB 9.0MB/s eta 0:00:01
```

```
Collecting scs>=1.1.3 (from cvxpy==1.0.3->-r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/07/a7/0b19c8f9476a4
762d296c6c5fa860f2fe580a4f579fa53aaa8515f4ca217/scs-2.1.0.tar.gz (154kB)
    100% |████████████████████████████████| 163kB 2.3MB/s eta 0:00:01
Collecting multiprocess (from cvxpy==1.0.3->-r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/31/60/6d74caa02b54c
a43092e745640c7d98f367f07160441682a01602ce00bc5/multiprocess-0.70.7.tar.g
z (1.4MB)
    100% |████████████████████████████████| 1.4MB 325kB/s eta 0:00:01
Requirement already satisfied: fastcache in /opt/conda/lib/python3.6/site
-packages (from cvxpy==1.0.3->-r requirements.txt (line 3))
Requirement already satisfied: toolz in /opt/conda/lib/python3.6/site-pac
kages (from cvxpy==1.0.3->-r requirements.txt (line 3))
Requirement already satisfied: decorator>=4.0.6 in /opt/conda/lib/python
3.6/site-packages (from plotly==2.2.3->-r requirements.txt (line 7))
Requirement already satisfied: nbformat>=4.2 in /opt/conda/lib/python3.6/
site-packages (from plotly==2.2.3->-r requirements.txt (line 7))
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/lib/py
thon3.6/site-packages (from requests==2.18.4->-r requirements.txt (line 1
1))
Requirement already satisfied: idna<2.7,>=2.5 in /opt/conda/lib/python3.
6/site-packages (from requests==2.18.4->-r requirements.txt (line 11))
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /opt/conda/lib/py
thon3.6/site-packages (from requests==2.18.4->-r requirements.txt (line 1
1))
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/pytho
n3.6/site-packages (from requests==2.18.4->-r requirements.txt (line 11))
Requirement already satisfied: numexpr>=2.5.2 in /opt/conda/lib/python3.
6/site-packages (from tables==3.3.0->-r requirements.txt (line 15))
Requirement already satisfied: pip>=7.1.0 in /opt/conda/lib/python3.6/sit
e-packages (from zipline===1.2.0->-r requirements.txt (line 17))
Requirement already satisfied: setuptools>18.0 in /opt/conda/lib/python3.
6/site-packages (from zipline===1.2.0->-r requirements.txt (line 17))
Collecting Logbook>=0.12.5 (from zipline===1.2.0->-r requirements.txt (li
ne 17))
  Downloading https://files.pythonhosted.org/packages/f6/83/20fc027061491
9cb799f76e32cf143a54c58ce2fa45c19fd38ac2e4f9977/Logbook-1.4.3.tar.gz (85k
B)
    100% |████████████████████████████████| 92kB 3.0MB/s eta 0:00:01
Collecting requests-file>=1.4.1 (from zipline===1.2.0->-r requirements.tx
t (line 17))
  Downloading https://files.pythonhosted.org/packages/23/9c/6e63c23c39e53
d3df41c77a3d05a49a42c4e1383a6d2a5e3233161b89dbf/requests_file-1.4.3-py2.p
y3-none-any.whl
Collecting pandas-datareader<0.6,>=0.2.1 (from zipline===1.2.0->-r requir
ements.txt (line 17))
  Downloading https://files.pythonhosted.org/packages/40/c5/cc720f531bbde
0efeab940de400d0fcc95e87770a3abcd7f90d6d52a3302/pandas_datareader-0.5.0-p
y2.py3-none-any.whl (74kB)
    100% |████████████████████████████████| 81kB 3.8MB/s eta 0:00:01
Requirement already satisfied: patsy>=0.4.0 in /opt/conda/lib/python3.6/s
ite-packages (from zipline===1.2.0->-r requirements.txt (line 17))
Requirement already satisfied: Cython>=0.25.2 in /opt/conda/lib/python3.
6/site-packages (from zipline===1.2.0->-r requirements.txt (line 17))
Collecting cyordereddict>=0.2.2 (from zipline===1.2.0->-r requirements.tx
t (line 17))
  Downloading https://files.pythonhosted.org/packages/d1/1a/364cbfd927be1
```

```
b743c7f0a985a7f1f7e8a51469619f9fefe4ee9240ba210/cyordereddict-1.0.0.tar.g
z (138kB)
    100% |████████████████████████████████| 143kB 2.6MB/s eta 0:00:01
Collecting bottleneck>=1.0.0 (from zipline===1.2.0->-r requirements.txt (
line 17))
  Downloading https://files.pythonhosted.org/packages/05/ae/cedf5323f398a
b4e4ff92d6c431a3e1c6a186f9b41ab3e8258dff786a290/Bottleneck-1.2.1.tar.gz (
105kB)
    100% |████████████████████████████████| 112kB 2.7MB/s ta 0:00:01   7
7% |█████████████████████████       | 81kB 10.0MB/s eta 0:00:01
Collecting contextlib2>=0.4.0 (from zipline===1.2.0->-r requirements.txt
(line 17))
  Downloading https://files.pythonhosted.org/packages/a2/71/8273a7eeed0af
f6a854237ab5453bc9aa67deb49df4832801c21f0ff3782/contextlib2-0.5.5-py2.py3
-none-any.whl
Requirement already satisfied: networkx<2.0,>=1.9.1 in /opt/conda/lib/pyt
hon3.6/site-packages (from zipline===1.2.0->-r requirements.txt (line 1
7))
Collecting bcolz<1,>=0.12.1 (from zipline===1.2.0->-r requirements.txt (l
ine 17))
  Downloading https://files.pythonhosted.org/packages/6c/8b/1ffa01f872cac
36173c5eb95b58c01040d8d25f1b242c48577f4104cd3ab/bcolz-0.12.1.tar.gz (622k
B)
    100% |████████████████████████████████| 624kB 717kB/s eta 0:00:01
Requirement already satisfied: click>=4.0.0 in /opt/conda/lib/python3.6/s
ite-packages (from zipline===1.2.0->-r requirements.txt (line 17))
Collecting multipledispatch>=0.4.8 (from zipline===1.2.0->-r requirement
s.txt (line 17))
  Downloading https://files.pythonhosted.org/packages/89/79/429ecef45fd5e
4504f7474d4c3c3c4668c267be3370e4c2fd33e61506833/multipledispatch-0.6.0-py
3-none-any.whl
Requirement already satisfied: MarkupSafe>=0.23 in /opt/conda/lib/python
3.6/site-packages (from zipline===1.2.0->-r requirements.txt (line 17))
Requirement already satisfied: Mako>=1.0.1 in /opt/conda/lib/python3.6/si
te-packages/Mako-1.0.7-py3.6.egg (from zipline===1.2.0->-r requirements.t
xt (line 17))
Requirement already satisfied: sqlalchemy>=1.0.8 in /opt/conda/lib/python
3.6/site-packages (from zipline===1.2.0->-r requirements.txt (line 17))
Collecting alembic>=0.7.7 (from zipline===1.2.0->-r requirements.txt (lin
e 17))
  Downloading https://files.pythonhosted.org/packages/d6/bb/ec1e21f2e3036
89ad2170eb47fc67df9ad4199ade6759a99474c4d3535c8/alembic-1.0.8.tar.gz (1.0
MB)
    100% |████████████████████████████████| 1.0MB 448kB/s eta 0:00:01
Collecting sortedcontainers>=1.4.4 (from zipline===1.2.0->-r requirement
s.txt (line 17))
  Downloading https://files.pythonhosted.org/packages/13/f3/cf85f7c3a2dbd
1a515d51e1f1676d971abe41bba6f4ab5443240d9a78e5b/sortedcontainers-2.1.0-py
2.py3-none-any.whl
Collecting intervaltree>=2.1.0 (from zipline===1.2.0->-r requirements.txt
(line 17))
  Downloading https://files.pythonhosted.org/packages/e8/f9/76237755b2020
cd74549e98667210b2dd54d3fb17c6f4a62631e61d31225/intervaltree-3.0.2.tar.gz
Collecting lru-dict>=1.1.4 (from zipline===1.2.0->-r requirements.txt (li
ne 17))
  Downloading https://files.pythonhosted.org/packages/00/a5/32ed6e10246cd
341ca8cc205acea5d208e4053f48a4dced2b1b31d45ba3f/lru-dict-1.1.6.tar.gz
```

```
Collecting empyrical>=0.4.2 (from zipline===1.2.0->-r requirements.txt (l
ine 17))
  Downloading https://files.pythonhosted.org/packages/7b/55/a01b05162b764
830dbbac868462f44cd847a5b6523a01ca9f955721819da/empyrical-0.5.0.tar.gz (4
9kB)
    100% |████████████████████████████████| 51kB 3.1MB/s ta 0:00:01
Requirement already satisfied: pygments in /opt/conda/lib/python3.6/site-
packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.txt (lin
e 1))
Requirement already satisfied: jedi>=0.10 in /opt/conda/lib/python3.6/sit
e-packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.txt (l
ine 1))
Requirement already satisfied: backcall in /opt/conda/lib/python3.6/site-
packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.txt (lin
e 1))
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.6/si
te-packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.txt (
line 1))
Requirement already satisfied: traitlets>=4.2 in /opt/conda/lib/python3.
6/site-packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirements.t
xt (line 1))
Requirement already satisfied: pexpect; sys_platform != "win32" in /opt/c
onda/lib/python3.6/site-packages (from IPython>=3.2.3->alphalens==0.3.2->
-r requirements.txt (line 1))
Requirement already satisfied: simplegeneric>0.8 in /opt/conda/lib/python
3.6/site-packages (from IPython>=3.2.3->alphalens==0.3.2->-r requirement
s.txt (line 1))
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.15 in /opt/cond
a/lib/python3.6/site-packages (from IPython>=3.2.3->alphalens==0.3.2->-r
requirements.txt (line 1))
Requirement already satisfied: future in /opt/conda/lib/python3.6/site-pa
ckages (from osqp->cvxpy==1.0.3->-r requirements.txt (line 3))
Collecting dill>=0.2.9 (from multiprocess->cvxpy==1.0.3->-r requirements.
txt (line 3))
  Downloading https://files.pythonhosted.org/packages/fe/42/bfe2e0857bc28
4cbe6a011d93f2a9ad58a22cb894461b199ae72cfef0f29/dill-0.2.9.tar.gz (150kB)
    100% |████████████████████████████████| 153kB 2.5MB/s eta 0:00:01
Requirement already satisfied: ipython-genutils in /opt/conda/lib/python
3.6/site-packages (from nbformat>=4.2->plotly==2.2.3->-r requirements.txt
(line 7))
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /opt/conda/lib/
python3.6/site-packages (from nbformat>=4.2->plotly==2.2.3->-r requiremen
ts.txt (line 7))
Requirement already satisfied: jupyter-core in /opt/conda/lib/python3.6/s
ite-packages (from nbformat>=4.2->plotly==2.2.3->-r requirements.txt (lin
e 7))
Collecting requests-ftp (from pandas-datareader<0.6,>=0.2.1->zipline===1.
2.0->-r requirements.txt (line 17))
  Downloading https://files.pythonhosted.org/packages/3d/ca/14b2ad1e93b51
95eeaf56b86b7ecfd5ea2d5754a68d17aeb1e5b9f95b3cf/requests-ftp-0.3.1.tar.gz
Collecting python-editor>=0.3 (from alembic>=0.7.7->zipline===1.2.0->-r r
equirements.txt (line 17))
  Downloading https://files.pythonhosted.org/packages/c6/d3/201fc3abe391b
bae6606e6f1d598c15d367033332bd54352b12f35513717/python_editor-1.0.4-py3-n
one-any.whl
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.
6/site-packages (from pexpect; sys_platform != "win32"->IPython>=3.2.3->a
```

```
lphalens==0.3.2->-r requirements.txt (line 1))
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.6/site-p
ackages (from prompt-toolkit<2.0.0,>=1.0.15->IPython>=3.2.3->alphalens==
0.3.2->-r requirements.txt (line 1))
Building wheels for collected packages: alphalens, cvxpy, pandas, plotly,
zipline, scs, multiprocess, Logbook, cyordereddict, bottleneck, bcolz, al
embic, intervaltree, lru-dict, empyrical, dill, requests-ftp
  Running setup.py bdist_wheel for alphalens ... done
  Stored in directory: /root/.cache/pip/wheels/77/1e/9a/223b4c94d7f564f25
d94b48ca5b9c53e3034016ece3fd8c8c1
  Running setup.py bdist_wheel for cvxpy ... done
  Stored in directory: /root/.cache/pip/wheels/2b/60/0b/0c2596528665e21d6
98d6f84a3406c52044c7b4ca6ac737cf3
  Running setup.py bdist_wheel for pandas ... done
  Stored in directory: /root/.cache/pip/wheels/a3/08/c3/8fdd52954d4b41562
4cff43c6dd32a22bac90306976a98f4af
  Running setup.py bdist_wheel for plotly ... done
  Stored in directory: /root/.cache/pip/wheels/98/54/81/dd92d5b0858fac680
cd7bdb8800eb26c001dd9f5dc8b1bc0ba
  Running setup.py bdist_wheel for zipline ... done
  Stored in directory: /root/.cache/pip/wheels/5d/20/7d/b48368c8634b1cb6c
c7232833b2780a265d4217c0ad2e3d24c
  Running setup.py bdist_wheel for scs ... done
  Stored in directory: /root/.cache/pip/wheels/94/e2/a6/64db723051c54017c
248ea5a26e7f1459c0242d735a496dd55
  Running setup.py bdist_wheel for multiprocess ... done
  Stored in directory: /root/.cache/pip/wheels/3a/ed/51/77c833462c3e757ce
50c4b2b68bdf53f5d1745542fe567d740
  Running setup.py bdist_wheel for Logbook ... done
  Stored in directory: /root/.cache/pip/wheels/a2/9f/6f/8c7a4ed6b9f6f3c98
b742dbb0fd41fff3c130119c507376301
  Running setup.py bdist_wheel for cyordereddict ... done
  Stored in directory: /root/.cache/pip/wheels/0b/9d/8b/5bf3e22c1edd59b50
f11bb19dec9dfcfe5a479fc7ace02b61f
  Running setup.py bdist_wheel for bottleneck ... done
  Stored in directory: /root/.cache/pip/wheels/f2/bf/ec/e0f39aa27001525ad
455139ee57ec7d0776fe074dfd78c97e4
  Running setup.py bdist_wheel for bcolz ... done
  Stored in directory: /root/.cache/pip/wheels/c5/cc/1b/2cf1f88959af5d7f4
d449b7fc6c9452d0ecbd86fd61a9ee376
  Running setup.py bdist_wheel for alembic ... done
  Stored in directory: /root/.cache/pip/wheels/a0/bc/74/834fa0c75c4ae6d67
18db5e65187d508623ee291dead032156
  Running setup.py bdist_wheel for intervaltree ... done
  Stored in directory: /root/.cache/pip/wheels/08/99/c0/5a5942f5b9567c59c
14aac76f95a70bf11dccc71240b91ebf5
  Running setup.py bdist_wheel for lru-dict ... done
  Stored in directory: /root/.cache/pip/wheels/b7/ef/06/fbdd555907a7d438f
b33e4c8675f771ff1cf41917284c51ebf
  Running setup.py bdist_wheel for empyrical ... done
  Stored in directory: /root/.cache/pip/wheels/83/14/73/34fb27552601518d2
8bd0813d75124be76d94ab29152c69112
  Running setup.py bdist_wheel for dill ... done
  Stored in directory: /root/.cache/pip/wheels/5b/d7/0f/e58eae695403de585
269f4e4a94e0cd6ca60ec0c202936fa4a
  Running setup.py bdist_wheel for requests-ftp ... done
  Stored in directory: /root/.cache/pip/wheels/2a/98/32/37195e45a3392a73d
```

```
9f65c488cbea30fe5bad76aaef4d6b020
Successfully built alphalens cvxpy pandas plotly zipline scs multiprocess
Logbook cyordereddict bottleneck bcolz alembic intervaltree lru-dict empy
rical dill requests-ftp
Installing collected packages: numpy, pandas, scipy, alphalens, osqp, eco
s, scs, dill, multiprocess, cvxpy, plotly, tables, tqdm, Logbook, request
s-file, requests-ftp, pandas-datareader, cyordereddict, bottleneck, conte
xtlib2, bcolz, multipledispatch, python-editor, alembic, sortedcontainer
s, intervaltree, lru-dict, empyrical, zipline
  Found existing installation: numpy 1.12.1
    Uninstalling numpy-1.12.1:
      Successfully uninstalled numpy-1.12.1
  Found existing installation: pandas 0.23.3
    Uninstalling pandas-0.23.3:
      Successfully uninstalled pandas-0.23.3
  Found existing installation: scipy 0.19.1
    Uninstalling scipy-0.19.1:
      Successfully uninstalled scipy-0.19.1
  Found existing installation: dill 0.2.7.1
    Uninstalling dill-0.2.7.1:
      Successfully uninstalled dill-0.2.7.1
  Found existing installation: plotly 2.0.15
    Uninstalling plotly-2.0.15:
      Successfully uninstalled plotly-2.0.15
  Found existing installation: tqdm 4.11.2
    Uninstalling tqdm-4.11.2:
      Successfully uninstalled tqdm-4.11.2
Successfully installed Logbook-1.4.3 alembic-1.0.8 alphalens-0.3.2 bcolz-
0.12.1 bottleneck-1.2.1 contextlib2-0.5.5 cvxpy-1.0.3 cyordereddict-1.0.0
dill-0.2.9 ecos-2.0.7.post1 empyrical-0.5.0 intervaltree-3.0.2 lru-dict-
1.1.6 multipledispatch-0.6.0 multiprocess-0.70.7 numpy-1.13.3 osqp-0.5.0
pandas-0.18.1 pandas-datareader-0.5.0 plotly-2.2.3 python-editor-1.0.4 re
quests-file-1.4.3 requests-ftp-0.3.1 scipy-1.0.0 scs-2.1.0 sortedcontaine
rs-2.1.0 tables-3.3.0 tqdm-4.19.5 zipline-1.2.0
You are using pip version 9.0.1, however version 19.0.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' comman
d.
```

Load Packages

In [4]:

```
import cvxpy as cvx
import numpy as np
import pandas as pd
import time
import project_tests
import project_helper

import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = (14, 8)
```

Data Bundle

We'll be using Zipline to handle our data. We've created a end of day data bundle for this project. Run the cell below to register this data bundle in zipline.

In [5]:
```
import os
import project_helper
from zipline.data import bundles

os.environ['ZIPLINE_ROOT'] = os.path.join(os.getcwd(), '..', '..', 'data'

ingest_func = bundles.csvdir.csvdir_equities(['daily'], project_helper.EO
bundles.register(project_helper.EOD_BUNDLE_NAME, ingest_func)

print('Data Registered')
```

```
Data Registered
```

Build Pipeline Engine

We'll be using Zipline's pipeline package to access our data for this project. To use it, we must build a pipeline engine. Run the cell below to build the engine.

In [6]:
```
from zipline.pipeline import Pipeline
from zipline.pipeline.factors import AverageDollarVolume
from zipline.utils.calendars import get_calendar


universe = AverageDollarVolume(window_length=120).top(500)
trading_calendar = get_calendar('NYSE')
bundle_data = bundles.load(project_helper.EOD_BUNDLE_NAME)
engine = project_helper.build_pipeline_engine(bundle_data, trading_calend
```

View Data

With the pipeline engine built, let's get the stocks at the end of the period in the universe we're using. We'll use these tickers to generate the returns data for the our risk model.

In [7]:
```
universe_end_date = pd.Timestamp('2016-01-05', tz='UTC')

universe_tickers = engine\
    .run_pipeline(
        Pipeline(screen=universe),
        universe_end_date,
        universe_end_date)\
    .index.get_level_values(1)\
    .values.tolist()

universe_tickers
```

Out[7]:
```
[Equity(0 [A]),
 Equity(1 [AAL]),
 Equity(2 [AAP]),
 Equity(3 [AAPL]),
 Equity(4 [ABBV]),
 Equity(5 [ABC]),
 Equity(6 [ABT]),
 Equity(7 [ACN]),
 Equity(8 [ADBE]),
 Equity(9 [ADI]),
 Equity(10 [ADM]),
 Equity(11 [ADP]),
 Equity(12 [ADS]),
 Equity(13 [ADSK]),
 Equity(14 [AEE]),
 Equity(15 [AEP]),
 Equity(16 [AES]),
 Equity(17 [AET]),
 Equity(18 [AFL]),
 Equity(19 [AGN]),
 Equity(20 [AIG]),
 Equity(21 [AIV]),
 Equity(22 [AIZ]),
 Equity(23 [AJG]),
 Equity(24 [AKAM]),
 Equity(25 [ALB]),
 Equity(26 [ALGN]),
 Equity(27 [ALK]),
 Equity(28 [ALL]),
 Equity(29 [ALLE]),
 Equity(30 [ALXN]),
 Equity(31 [AMAT]),
 Equity(32 [AMD]),
 Equity(33 [AME]),
 Equity(34 [AMG]),
 Equity(35 [AMGN]),
 Equity(36 [AMP]),
 Equity(37 [AMT]),
 Equity(38 [AMZN]),
 Equity(39 [ANDV]),
 Equity(40 [ANSS]),
 Equity(41 [ANTM]),
 Equity(42 [AON]),
 Equity(43 [AOS]),
```

```
                Equity(44 [APA]),
                Equity(45 [APC]),
                Equity(46 [APD]),
                Equity(47 [APH]),
                Equity(48 [ARE]),
                Equity(49 [ARNC]),
                Equity(50 [ATVI]),
                Equity(51 [AVB]),
                Equity(52 [AVGO]),
                Equity(53 [AVY]),
                Equity(54 [AWK]),
                Equity(55 [AXP]),
                Equity(56 [AYI]),
                Equity(57 [AZO]),
                Equity(58 [BA]),
                Equity(59 [BAC]),
                Equity(60 [BAX]),
                Equity(61 [BBT]),
                Equity(62 [BBY]),
                Equity(63 [BCR]),
                Equity(64 [BDX]),
                Equity(65 [BEN]),
                Equity(66 [BIIB]),
                Equity(67 [BK]),
                Equity(68 [BLK]),
                Equity(69 [BLL]),
                Equity(70 [BMY]),
                Equity(71 [BSX]),
                Equity(72 [BWA]),
                Equity(73 [BXP]),
                Equity(74 [C]),
                Equity(75 [CA]),
                Equity(76 [CAG]),
                Equity(77 [CAH]),
                Equity(78 [CAT]),
                Equity(79 [CB]),
                Equity(80 [CBG]),
                Equity(81 [CBOE]),
                Equity(82 [CBS]),
                Equity(83 [CCI]),
                Equity(84 [CCL]),
                Equity(85 [CELG]),
                Equity(86 [CERN]),
                Equity(87 [CF]),
                Equity(88 [CFG]),
                Equity(89 [CHD]),
                Equity(90 [CHK]),
                Equity(91 [CHRW]),
                Equity(92 [CHTR]),
                Equity(93 [CI]),
                Equity(94 [CINF]),
                Equity(95 [CL]),
                Equity(96 [CLX]),
                Equity(97 [CMA]),
                Equity(98 [CMCSA]),
                Equity(99 [CME]),
                Equity(100 [CMG]),
```

```
Equity(101 [CMI]),
Equity(102 [CMS]),
Equity(103 [CNC]),
Equity(104 [CNP]),
Equity(105 [COF]),
Equity(106 [COG]),
Equity(107 [COL]),
Equity(108 [COO]),
Equity(109 [COP]),
Equity(110 [COST]),
Equity(111 [COTY]),
Equity(112 [CPB]),
Equity(113 [CRM]),
Equity(114 [CSCO]),
Equity(115 [CSRA]),
Equity(116 [CSX]),
Equity(117 [CTAS]),
Equity(118 [CTL]),
Equity(119 [CTSH]),
Equity(120 [CTXS]),
Equity(121 [CVS]),
Equity(122 [CVX]),
Equity(123 [CXO]),
Equity(124 [D]),
Equity(125 [DAL]),
Equity(126 [DE]),
Equity(127 [DFS]),
Equity(128 [DG]),
Equity(129 [DGX]),
Equity(130 [DHI]),
Equity(131 [DHR]),
Equity(132 [DIS]),
Equity(133 [DISCA]),
Equity(134 [DISCK]),
Equity(135 [DISH]),
Equity(136 [DLR]),
Equity(137 [DLTR]),
Equity(138 [DOV]),
Equity(139 [DPS]),
Equity(140 [DRE]),
Equity(141 [DRI]),
Equity(142 [DTE]),
Equity(143 [DUK]),
Equity(144 [DVA]),
Equity(145 [DVN]),
Equity(146 [EA]),
Equity(147 [EBAY]),
Equity(148 [ECL]),
Equity(149 [ED]),
Equity(150 [EFX]),
Equity(151 [EIX]),
Equity(152 [EL]),
Equity(153 [EMN]),
Equity(154 [EMR]),
Equity(155 [EOG]),
Equity(156 [EQIX]),
Equity(157 [EQR]),
```

```
                Equity(158 [EQT]),
                Equity(159 [ES]),
                Equity(160 [ESRX]),
                Equity(161 [ESS]),
                Equity(162 [ETFC]),
                Equity(163 [ETN]),
                Equity(164 [ETR]),
                Equity(165 [EVHC]),
                Equity(166 [EW]),
                Equity(167 [EXC]),
                Equity(168 [EXPD]),
                Equity(169 [EXPE]),
                Equity(170 [EXR]),
                Equity(171 [F]),
                Equity(172 [FAST]),
                Equity(173 [FB]),
                Equity(174 [FBHS]),
                Equity(175 [FCX]),
                Equity(176 [FDX]),
                Equity(177 [FE]),
                Equity(178 [FFIV]),
                Equity(179 [FIS]),
                Equity(180 [FISV]),
                Equity(181 [FITB]),
                Equity(182 [FL]),
                Equity(183 [FLIR]),
                Equity(184 [FLR]),
                Equity(185 [FLS]),
                Equity(186 [FMC]),
                Equity(187 [FOX]),
                Equity(188 [FOXA]),
                Equity(189 [FRT]),
                Equity(190 [FTI]),
                Equity(191 [GD]),
                Equity(192 [GE]),
                Equity(193 [GGP]),
                Equity(194 [GILD]),
                Equity(195 [GIS]),
                Equity(196 [GLW]),
                Equity(197 [GM]),
                Equity(198 [GOOG]),
                Equity(199 [GOOGL]),
                Equity(200 [GPC]),
                Equity(201 [GPN]),
                Equity(202 [GPS]),
                Equity(203 [GRMN]),
                Equity(204 [GS]),
                Equity(205 [GT]),
                Equity(206 [GWW]),
                Equity(207 [HAL]),
                Equity(208 [HAS]),
                Equity(209 [HBAN]),
                Equity(210 [HBI]),
                Equity(211 [HCA]),
                Equity(212 [HCN]),
                Equity(213 [HCP]),
                Equity(214 [HD]),
```

```
Equity(215 [HES]),
Equity(216 [HIG]),
Equity(217 [HLT]),
Equity(218 [HOG]),
Equity(219 [HOLX]),
Equity(220 [HON]),
Equity(221 [HP]),
Equity(222 [HPE]),
Equity(223 [HPQ]),
Equity(224 [HRB]),
Equity(225 [HRL]),
Equity(226 [HRS]),
Equity(227 [HSIC]),
Equity(228 [HST]),
Equity(229 [HSY]),
Equity(230 [HUM]),
Equity(231 [IBM]),
Equity(232 [ICE]),
Equity(233 [IDXX]),
Equity(234 [IFF]),
Equity(235 [ILMN]),
Equity(236 [INCY]),
Equity(237 [INFO]),
Equity(238 [INTC]),
Equity(239 [INTU]),
Equity(240 [IP]),
Equity(241 [IPG]),
Equity(242 [IR]),
Equity(243 [IRM]),
Equity(244 [ISRG]),
Equity(245 [IT]),
Equity(246 [ITW]),
Equity(247 [IVZ]),
Equity(248 [JBHT]),
Equity(249 [JCI]),
Equity(250 [JEC]),
Equity(251 [JNJ]),
Equity(252 [JNPR]),
Equity(253 [JPM]),
Equity(254 [JWN]),
Equity(255 [K]),
Equity(256 [KEY]),
Equity(257 [KHC]),
Equity(258 [KIM]),
Equity(259 [KLAC]),
Equity(260 [KMB]),
Equity(261 [KMI]),
Equity(262 [KMX]),
Equity(263 [KO]),
Equity(264 [KORS]),
Equity(265 [KR]),
Equity(266 [KSS]),
Equity(267 [KSU]),
Equity(268 [L]),
Equity(269 [LB]),
Equity(270 [LEG]),
Equity(271 [LEN]),
```

```
Equity(272 [LH]),
Equity(273 [LKQ]),
Equity(274 [LLL]),
Equity(275 [LLY]),
Equity(276 [LMT]),
Equity(277 [LNC]),
Equity(278 [LNT]),
Equity(279 [LOW]),
Equity(280 [LRCX]),
Equity(281 [LUK]),
Equity(282 [LUV]),
Equity(283 [LVLT]),
Equity(284 [LYB]),
Equity(285 [M]),
Equity(286 [MA]),
Equity(287 [MAA]),
Equity(288 [MAC]),
Equity(289 [MAR]),
Equity(290 [MAS]),
Equity(291 [MAT]),
Equity(292 [MCD]),
Equity(293 [MCHP]),
Equity(294 [MCK]),
Equity(295 [MCO]),
Equity(296 [MDLZ]),
Equity(297 [MDT]),
Equity(298 [MET]),
Equity(299 [MGM]),
Equity(300 [MHK]),
Equity(301 [MKC]),
Equity(302 [MLM]),
Equity(303 [MMC]),
Equity(304 [MNST]),
Equity(305 [MO]),
Equity(306 [MON]),
Equity(307 [MOS]),
Equity(308 [MPC]),
Equity(309 [MRK]),
Equity(310 [MRO]),
Equity(311 [MS]),
Equity(312 [MSFT]),
Equity(313 [MSI]),
Equity(314 [MTB]),
Equity(315 [MTD]),
Equity(316 [MU]),
Equity(317 [MYL]),
Equity(318 [NAVI]),
Equity(319 [NBL]),
Equity(320 [NDAQ]),
Equity(321 [NEE]),
Equity(322 [NEM]),
Equity(323 [NFLX]),
Equity(324 [NFX]),
Equity(325 [NI]),
Equity(326 [NKE]),
Equity(327 [NLSN]),
Equity(328 [NOC]),
```

```
Equity(329 [NOV]),
Equity(330 [NRG]),
Equity(331 [NSC]),
Equity(332 [NTAP]),
Equity(333 [NTRS]),
Equity(334 [NUE]),
Equity(335 [NVDA]),
Equity(336 [NWL]),
Equity(337 [NWS]),
Equity(338 [NWSA]),
Equity(339 [O]),
Equity(340 [OKE]),
Equity(341 [OMC]),
Equity(342 [ORCL]),
Equity(343 [ORLY]),
Equity(344 [OXY]),
Equity(345 [PAYX]),
Equity(346 [PBCT]),
Equity(347 [PCAR]),
Equity(348 [PCG]),
Equity(349 [PDCO]),
Equity(350 [PEG]),
Equity(351 [PEP]),
Equity(352 [PFE]),
Equity(353 [PFG]),
Equity(354 [PG]),
Equity(355 [PGR]),
Equity(356 [PH]),
Equity(357 [PHM]),
Equity(358 [PKG]),
Equity(359 [PKI]),
Equity(360 [PLD]),
Equity(361 [PM]),
Equity(362 [PNC]),
Equity(363 [PNR]),
Equity(364 [PNW]),
Equity(365 [PPG]),
Equity(366 [PPL]),
Equity(367 [PRGO]),
Equity(368 [PRU]),
Equity(369 [PSA]),
Equity(370 [PSX]),
Equity(371 [PVH]),
Equity(372 [PWR]),
Equity(373 [PX]),
Equity(374 [PXD]),
Equity(375 [PYPL]),
Equity(376 [QCOM]),
Equity(377 [QRVO]),
Equity(378 [RCL]),
Equity(379 [RE]),
Equity(380 [REG]),
Equity(381 [REGN]),
Equity(382 [RF]),
Equity(383 [RHI]),
Equity(384 [RHT]),
Equity(385 [RJF]),
```

```
Equity(386 [RL]),
Equity(387 [RMD]),
Equity(388 [ROK]),
Equity(389 [ROP]),
Equity(390 [ROST]),
Equity(391 [RRC]),
Equity(392 [RSG]),
Equity(393 [RTN]),
Equity(394 [SBAC]),
Equity(395 [SBUX]),
Equity(396 [SCG]),
Equity(397 [SCHW]),
Equity(398 [SEE]),
Equity(399 [SHW]),
Equity(400 [SIG]),
Equity(401 [SJM]),
Equity(402 [SLB]),
Equity(403 [SLG]),
Equity(404 [SNA]),
Equity(405 [SNI]),
Equity(406 [SNPS]),
Equity(407 [SO]),
Equity(408 [SPG]),
Equity(409 [SPLS]),
Equity(410 [SRCL]),
Equity(411 [SRE]),
Equity(412 [STI]),
Equity(413 [STT]),
Equity(414 [STX]),
Equity(415 [STZ]),
Equity(416 [SWK]),
Equity(417 [SWKS]),
Equity(418 [SYF]),
Equity(419 [SYK]),
Equity(420 [SYMC]),
Equity(421 [SYY]),
Equity(422 [T]),
Equity(423 [TAP]),
Equity(424 [TDG]),
Equity(425 [TEL]),
Equity(426 [TGT]),
Equity(427 [TIF]),
Equity(428 [TJX]),
Equity(429 [TMK]),
Equity(430 [TMO]),
Equity(431 [TRIP]),
Equity(432 [TROW]),
Equity(433 [TRV]),
Equity(434 [TSCO]),
Equity(435 [TSN]),
Equity(436 [TSS]),
Equity(437 [TWX]),
Equity(438 [TXN]),
Equity(439 [TXT]),
Equity(440 [UAA]),
Equity(441 [UAL]),
Equity(442 [UDR]),
```

```
 Equity(443 [UHS]),
 Equity(444 [ULTA]),
 Equity(445 [UNH]),
 Equity(446 [UNM]),
 Equity(447 [UNP]),
 Equity(448 [UPS]),
 Equity(449 [URI]),
 Equity(450 [USB]),
 Equity(451 [UTX]),
 Equity(452 [V]),
 Equity(453 [VAR]),
 Equity(454 [VFC]),
 Equity(455 [VIAB]),
 Equity(456 [VLO]),
 Equity(457 [VMC]),
 Equity(458 [VNO]),
 Equity(459 [VRSK]),
 Equity(460 [VRSN]),
 Equity(461 [VRTX]),
 Equity(462 [VTR]),
 Equity(463 [VZ]),
 Equity(464 [WAT]),
 Equity(465 [WBA]),
 Equity(466 [WDC]),
 Equity(467 [WEC]),
 Equity(468 [WFC]),
 Equity(469 [WHR]),
 Equity(471 [WM]),
 Equity(472 [WMB]),
 Equity(473 [WMT]),
 Equity(474 [WRK]),
 Equity(475 [WU]),
 Equity(476 [WY]),
 Equity(477 [WYN]),
 Equity(478 [WYNN]),
 Equity(479 [XEC]),
 Equity(480 [XEL]),
 Equity(481 [XL]),
 Equity(482 [XLNX]),
 Equity(483 [XOM]),
 Equity(484 [XRAY]),
 Equity(485 [XRX]),
 Equity(486 [XYL]),
 Equity(487 [YUM]),
 Equity(488 [ZBH]),
 Equity(489 [ZION]),
 Equity(490 [ZTS])]
```

Get Returns

Not that we have our pipeline built, let's access the returns data. We'll start by building a data portal.

In [8]:
```
from zipline.data.data_portal import DataPortal


data_portal = DataPortal(
    bundle_data.asset_finder,
    trading_calendar=trading_calendar,
    first_trading_day=bundle_data.equity_daily_bar_reader.first_trading_d
    equity_minute_reader=None,
    equity_daily_reader=bundle_data.equity_daily_bar_reader,
    adjustment_reader=bundle_data.adjustment_reader)
```

To make the code easier to read, we've built the helper function `get_pricing` to get the pricing from the data portal.

In [9]:
```
def get_pricing(data_portal, trading_calendar, assets, start_date, end_da
    end_dt = pd.Timestamp(end_date.strftime('%Y-%m-%d'), tz='UTC', offset
    start_dt = pd.Timestamp(start_date.strftime('%Y-%m-%d'), tz='UTC', of

    end_loc = trading_calendar.closes.index.get_loc(end_dt)
    start_loc = trading_calendar.closes.index.get_loc(start_dt)

    return data_portal.get_history_window(
        assets=assets,
        end_dt=end_dt,
        bar_count=end_loc - start_loc,
        frequency='1d',
        field=field,
        data_frequency='daily')
```

View Data

Let's get returns data for our risk model using the `get_pricing` function. For this model, we'll be looking back to 5 years of data.

In [10]:
```
five_year_returns = \
    get_pricing(
        data_portal,
        trading_calendar,
        universe_tickers,
        universe_end_date - pd.DateOffset(years=5),
        universe_end_date)\
    .pct_change()[1:].fillna(0)

five_year_returns
```

Out[10]:

	Equity(0 [A])	Equity(1 [AAL])	Equity(2 [AAP])	Equity(3 [AAPL])	Equity(4 [ABBV])	
2011-01-07 00:00:00+00:00	0.00843652	0.01423027	0.02670202	0.00714639	0.00000000	0
2011-01-10 00:00:00+00:00	-0.00417428	0.00619534	0.00743543	0.01885158	0.00000000	-0

2011-01-11 00:00:00+00:00	-0.00188630	-0.04364361	-0.00592730	-0.00236744	0.00000000	0
2011-01-12 00:00:00+00:00	0.01725375	-0.00823708	0.01338721	0.00813289	0.00000000	-0
2011-01-13 00:00:00+00:00	-0.00455851	0.00095465	0.00303109	0.00365656	0.00000000	0
2011-01-14 00:00:00+00:00	0.00343886	-0.00915594	0.00302193	0.00810620	0.00000000	0
2011-01-18 00:00:00+00:00	0.03425353	-0.06208490	-0.00428562	-0.02247419	0.00000000	0
2011-01-19 00:00:00+00:00	-0.01022379	-0.00892857	0.00875376	-0.00531448	0.00000000	-0
2011-01-20 00:00:00+00:00	-0.00849568	0.02195299	-0.00473189	-0.01818900	0.00000000	0
2011-01-21 00:00:00+00:00	0.00787281	-0.04103759	0.00554409	-0.01791080	0.00000000	0
2011-01-24 00:00:00+00:00	0.01464622	0.02747253	-0.00110591	0.03283704	0.00000000	0
2011-01-25 00:00:00+00:00	-0.00673624	0.00298231	0.00914590	0.01170955	0.00000000	-0
2011-01-26 00:00:00+00:00	-0.03073582	0.06613350	0.00359340	0.00719342	0.00000000	0
2011-01-27 00:00:00+00:00	0.00772081	0.02317753	-0.00155262	-0.00187707	0.00000000	0
2011-01-28 00:00:00+00:00	-0.01884631	-0.08055268	-0.00093620	-0.02070958	0.00000000	-0
2011-01-31 00:00:00+00:00	0.00360809	-0.02361480	-0.00235062	0.00957845	0.00000000	-0
2011-02-01 00:00:00+00:00	0.01165376	-0.00104701	-0.00921769	0.01681783	0.00000000	0
2011-02-02 00:00:00+00:00	0.01011176	-0.03930406	-0.02747650	-0.00205320	0.00000000	-0
2011-02-03 00:00:00+00:00	-0.00028893	0.00730962	0.01412639	-0.00256035	0.00000000	-0
2011-02-04 00:00:00+00:00	0.00562724	-0.03649951	0.02401434	0.00891547	0.00000000	0
2011-02-07 00:00:00+00:00	0.00770895	0.05204586	0.00811404	0.01553804	0.00000000	-0
2011-02-08 00:00:00+00:00	0.01085425	0.01645475	0.00620226	0.00943966	0.00000000	0
2011-02-09 00:00:00+00:00	0.00466351	0.00000000	0.01695500	0.00833204	0.00000000	0
2011-02-10 00:00:00+00:00	0.00041298	-0.00304846	-0.01136679	-0.01010922	0.00000000	0

2011-02-11 00:00:00+00:00	-0.00714982	0.02836356	0.00076442	0.00650490	0.00000000	-0
2011-02-14 00:00:00+00:00	0.00166312	-0.01579001	-0.02327358	0.00652903	0.00000000	0
2011-02-15 00:00:00+00:00	-0.01190476	0.01104282	-0.00392614	0.00201613	0.00000000	-0
2011-02-16 00:00:00+00:00	0.01512325	0.00195775	0.01385974	0.00896684	0.00000000	-0
2011-02-17 00:00:00+00:00	-0.00331066	-0.01779103	-0.02484354	-0.01330906	0.00000000	0
2011-02-18 00:00:00+00:00	0.01107771	-0.02010261	-0.00669325	-0.02159490	0.00000000	0
...	...	...	...	...	...	
2015-11-20 00:00:00+00:00	0.00107212	-0.00237346	0.00276686	0.00438086	0.00925624	-0.
2015-11-23 00:00:00+00:00	-0.00709429	0.00237910	-0.00122838	-0.01298872	0.00064612	-0
2015-11-24 00:00:00+00:00	0.00208486	-0.02530879	0.00350426	0.00959410	-0.00032285	-0
2015-11-25 00:00:00+00:00	-0.00820649	0.00193813	0.00680544	-0.00715141	-0.01374361	0
2015-11-27 00:00:00+00:00	-0.00325586	0.00920070	0.00328491	-0.00186283	-0.00480271	0
2015-11-30 00:00:00+00:00	-0.01020870	-0.01032093	-0.01279784	0.00415919	-0.03083813	-0
2015-12-01 00:00:00+00:00	0.01574786	0.04849282	-0.00239635	-0.00811576	0.01495718	0
2015-12-02 00:00:00+00:00	-0.00528420	0.01293012	-0.02716607	-0.00902983	-0.02202152	-0
2015-12-03 00:00:00+00:00	-0.00952117	-0.01255465	-0.01994438	-0.00929219	-0.02772394	-0
2015-12-04 00:00:00+00:00	0.02019919	0.03930296	0.00677909	0.03324578	0.01889890	0
2015-12-07 00:00:00+00:00	-0.00033987	0.01801987	-0.02957813	-0.00629799	-0.01592051	0.
2015-12-08 00:00:00+00:00	-0.02330712	-0.02687583	-0.00886608	-0.00042489	0.00711446	0
2015-12-09 00:00:00+00:00	-0.00516964	-0.02021340	0.02222267	-0.02207699	-0.01130272	0
2015-12-10 00:00:00+00:00	0.01503860	0.01009224	-0.00946596	0.00476320	-0.00446315	0
2015-12-11 00:00:00+00:00	-0.01222670	-0.04535632	-0.01917906	-0.02573993	-0.03118548	0

2015-12-14 00:00:00+00:00	-0.01119741	-0.00761835	-0.00832534	-0.00618871	0.02589759	0.
2015-12-15 00:00:00+00:00	0.02468348	0.01976847	0.05799045	-0.01768577	0.01713257	-0
2015-12-16 00:00:00+00:00	0.01078027	0.01419020	0.02977102	0.00768495	0.02199833	-0
2015-12-17 00:00:00+00:00	-0.01721238	-0.01712199	-0.04819014	-0.02119576	-0.02169610	0
2015-12-18 00:00:00+00:00	-0.04108447	-0.03225884	-0.02286262	-0.02706364	-0.01134153	-0
2015-12-21 00:00:00+00:00	0.00945523	0.03186317	0.00053085	0.01225425	0.00824462	0
2015-12-22 00:00:00+00:00	0.01050225	0.01167033	-0.01175973	-0.00092672	0.02474629	0
2015-12-23 00:00:00+00:00	0.01180348	0.00921901	0.00832501	0.01286896	0.01717833	0
2015-12-24 00:00:00+00:00	-0.00368236	0.01202196	0.00046505	-0.00534053	-0.00204082	0.
2015-12-28 00:00:00+00:00	0.00704030	-0.01325882	0.00952567	-0.01120361	0.00495300	0
2015-12-29 00:00:00+00:00	0.01944279	0.00625637	0.01095726	0.01797599	0.01191076	0
2015-12-30 00:00:00+00:00	-0.00638405	-0.01608535	-0.00525423	-0.01305616	0.00590373	0.
2015-12-31 00:00:00+00:00	-0.01243206	-0.01053186	-0.00587919	-0.01919944	-0.00937219	-
2016-01-04 00:00:00+00:00	-0.02828157	-0.03398810	0.01149418	0.00085542	-0.02751240	-
2016-01-05 00:00:00+00:00	0.00405845	-0.00954098	-0.00683002	-0.02505441	-0.00416936	0

1256 rows × 490 columns

Statistical Risk Model

It's time to build the risk model. You'll be creating a statistical risk model using PCA. So, the first thing is building the PCA model.

Fit PCA

Implement `fit_pca` to fit a PCA model to the returns data

In [11]:
```
from sklearn.decomposition import PCA


def fit_pca(returns, num_factor_exposures, svd_solver):
    """
    Fit PCA model with returns.

    Parameters
    ----------
    returns : DataFrame
        Returns for each ticker and date
    num_factor_exposures : int
        Number of factors for PCA
    svd_solver: str
        The solver to use for the PCA model

    Returns
    -------
    pca : PCA
        Model fit to returns
    """
    #TODO: Implement function
    pca = PCA(n_components=num_factor_exposures, svd_solver=svd_solver)
    pca.fit(returns)

    return pca


project_tests.test_fit_pca(fit_pca)
```

```
Tests Passed
```

View Data

Let's see what the model looks like. First, we'll look at the PCA components.

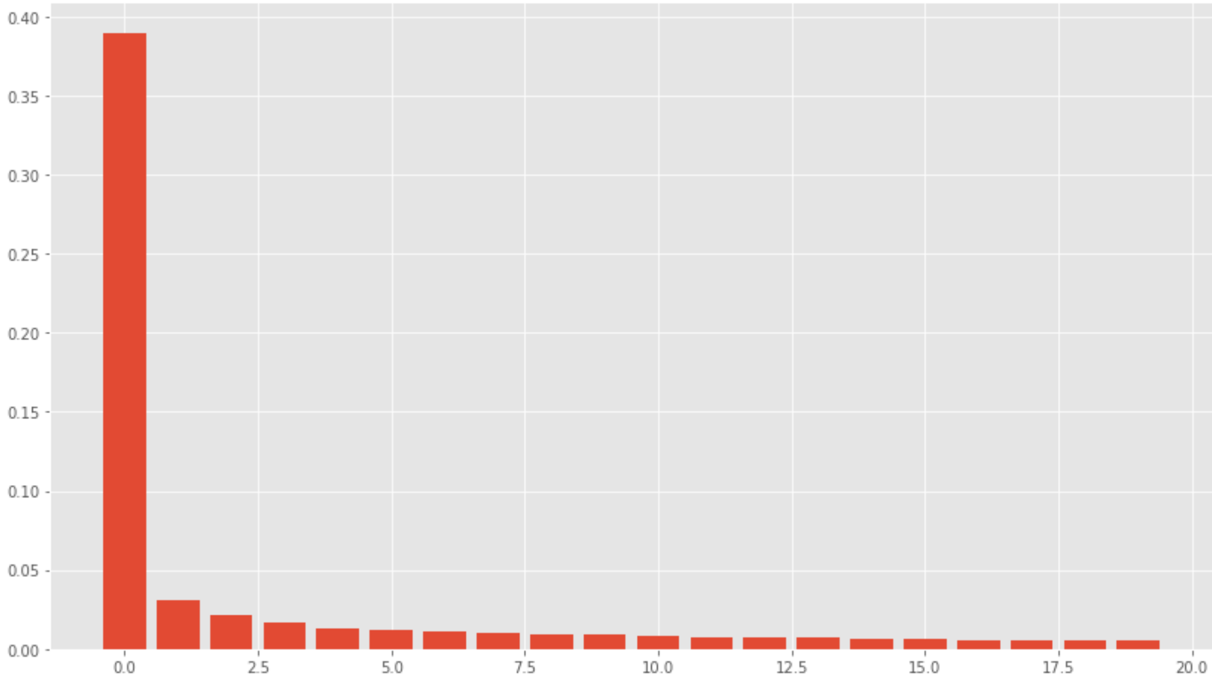
In [12]:
```
num_factor_exposures = 20
pca = fit_pca(five_year_returns, num_factor_exposures, 'full')

pca.components_
```

Out[12]:
```
array([[-0.04316847, -0.05874471, -0.03433256, ..., -0.03843904,
        -0.06092493, -0.01367163],
       [ 0.01955111,  0.19637679,  0.03451503, ...,  0.01749339,
        -0.01044197,  0.01892192],
       [-0.00993375,  0.07868756,  0.01133839, ..., -0.0157519 ,
         0.01261759,  0.01867875],
       ...,
       [-0.01174265,  0.01398085,  0.05143999, ...,  0.04125323,
         0.0035229 ,  0.03682367],
       [ 0.00526925, -0.04680674,  0.05716915, ...,  0.00671842,
        -0.02193923,  0.00833979],
       [-0.00535269, -0.01599057,  0.08414961, ..., -0.01540844,
         0.02188794,  0.01500221]])
```

Let's also look at the PCA's percent of variance explained by each factor

In [13]:
```
plt.bar(np.arange(num_factor_exposures), pca.explained_variance_ratio_)
```

Out[13]:
```
<Container object of 20 artists>
```



You will see that the first factor dominates. The precise definition of each factor in a latent model is unknown, however we can guess at the likely interpretation.

Factor Betas

Implement `factor_betas` to get the factor betas from the PCA model.

In [14]:
```
def factor_betas(pca, factor_beta_indices, factor_beta_columns):
    """
    Get the factor betas from the PCA model.

    Parameters
    ----------
    pca : PCA
        Model fit to returns
    factor_beta_indices : 1 dimensional Ndarray
        Factor beta indices
    factor_beta_columns : 1 dimensional Ndarray
        Factor beta columns

    Returns
    -------
    factor_betas : DataFrame
        Factor betas
    """
    assert len(factor_beta_indices.shape) == 1
    assert len(factor_beta_columns.shape) == 1

    #TODO: Implement function
    components = pca.components_.T
    the_factor_betas = pd.DataFrame(components, \
                                    index=factor_beta_indices, \
                                    columns=factor_beta_columns)

    return the_factor_betas


project_tests.test_factor_betas(factor_betas)
```

```
Tests Passed
```

View Data

Let's view the factor betas from this model.

In [15]:
```
risk_model = {}
risk_model['factor_betas'] = factor_betas(pca, five_year_returns.columns.

risk_model['factor_betas']
```

Out[15]:

	0	1	2	3	4	
Equity(0 [A])	-0.04316847	0.01955111	-0.00993375	0.01054038	-0.01819821	0.0107
Equity(1 [AAL])	-0.05874471	0.19637679	0.07868756	0.08209582	0.34847826	-0.1380
Equity(2 [AAP])	-0.03433256	0.03451503	0.01133839	-0.02543666	-0.00817211	-0.0131
Equity(3 [AAPL])	-0.03409988	-0.00139319	0.03946700	-0.01721303	-0.03046983	-0.0175

Equity(4 [ABBV])	-0.01803099	0.02568151	0.00435183	-0.07078179	0.01319937	0.0542
Equity(5 [ABC])	-0.02890016	0.03259161	-0.00742074	-0.03355183	-0.01152149	0.0264
Equity(6 [ABT])	-0.02905740	0.02977821	-0.02970871	-0.03574263	-0.01157351	0.0602
Equity(7 [ACN])	-0.04337745	0.00256907	0.00413229	-0.00349265	-0.05430743	0.0053
Equity(8 [ADBE])	-0.04730285	0.02661175	0.03057072	-0.02114690	-0.04838794	-0.0070
Equity(9 [ADI])	-0.04712287	-0.00381150	0.05600847	-0.01553775	-0.06946243	-0.0056
Equity(10 [ADM])	-0.04375945	-0.01130045	-0.03457005	0.00400541	-0.00645333	0.0174
Equity(11 [ADP])	-0.03648136	0.02528125	-0.01114832	-0.00940984	-0.03357614	0.0288
Equity(12 [ADS])	-0.04136654	0.01659887	0.01716323	-0.02418717	-0.00389740	0.0009
Equity(13 [ADSK])	-0.06028785	0.01995266	0.05164356	0.00105689	-0.06889175	-0.0489
Equity(14 [AEE])	-0.02646901	0.02222316	-0.10714999	-0.03824488	-0.00157895	-0.0231
Equity(15 [AEP])	-0.02263370	0.01880106	-0.10030310	-0.04210462	0.00205558	-0.0315
Equity(16 [AES])	-0.04557539	-0.01859287	-0.06960047	-0.02023102	0.01303429	-0.0174
Equity(17 [AET])	-0.04072498	0.02731997	-0.00507231	-0.02802199	-0.00180323	0.0638
Equity(18 [AFL])	-0.05336864	0.00134653	-0.02782496	0.06688998	-0.01355109	0.041
Equity(19 [AGN])	-0.03534102	0.04465091	0.02264084	-0.09556088	0.02867567	0.0651
Equity(20 [AIG])	-0.05982324	0.00588515	-0.01162873	0.06668318	0.01385338	0.0401
Equity(21 [AIV])	-0.04111735	0.03100207	-0.10211708	-0.00919327	0.01055402	-0.0473
Equity(22 [AIZ])	-0.04150874	0.01695771	-0.03297298	0.04770347	-0.00003273	0.0422
Equity(23 [AJG])	-0.03351812	0.01723113	-0.02538602	0.01303842	-0.00670008	0.0211
Equity(24 [AKAM])	-0.05587884	0.00798596	0.06670064	-0.00705542	-0.05149731	-0.0483
Equity(25 [ALB])	-0.05982022	-0.03657577	0.01798669	0.01363477	-0.01737160	-0.0307

Equity(26 [ALGN])	-0.05959262	0.01420686	0.04030645	-0.00223610	0.00381377	0.0355
Equity(27 [ALK])	-0.04887994	0.12134700	0.03971829	0.02356295	0.12648496	-0.0570
Equity(28 [ALL])	-0.03960459	0.01388579	-0.04298737	0.04705234	-0.00073028	0.0348
Equity(29 [ALLE])	-0.01215236	0.01485340	0.01783511	-0.02151478	0.00049741	0.0220
...	...	...	...	...	...	
Equity(460 [VRSN])	-0.03899324	0.01109333	0.04065347	-0.01351021	-0.03906872	0.0084
Equity(461 [VRTX])	-0.04909379	0.09821506	0.07490354	-0.22874272	0.18286011	0.2979
Equity(462 [VTR])	-0.03291326	0.03068907	-0.12174007	-0.03757317	-0.00091457	-0.0538
Equity(463 [VZ])	-0.07033190	0.00068427	-0.01610414	0.05981359	-0.01922444	0.0329
Equity(464 [WAT])	-0.04787963	0.01976102	0.00630830	-0.01309650	-0.02786813	0.0296
Equity(465 [WBA])	-0.03065220	0.03421533	-0.01027997	-0.04006665	-0.01725655	0.0203
Equity(466 [WDC])	-0.05340806	-0.00776436	0.06117296	-0.01397300	-0.02898379	-0.0515
Equity(467 [WEC])	-0.02279115	0.02557593	-0.10491696	-0.04691482	-0.00109145	-0.0288
Equity(468 [WFC])	-0.05131308	0.01109305	-0.03096455	0.08514607	0.00399898	0.0616
Equity(469 [WHR])	-0.05507142	0.03637282	-0.00100598	0.00590479	0.01092658	-0.0642
Equity(471 [WM])	-0.03151644	0.01496675	-0.03234348	-0.00111937	-0.02960532	0.0037
Equity(472 [WMB])	-0.05476156	-0.08924998	-0.02426238	-0.05723889	0.07886307	-0.0125
Equity(473 [WMT])	-0.01988671	0.03467415	-0.04017193	-0.00769646	-0.01460066	0.0072
Equity(474 [WRK])	-0.00563001	-0.00526226	0.00630657	-0.00994056	0.00615504	0.0108
Equity(475 [WU])	-0.04059879	0.00365919	0.01271940	0.00231537	-0.01726731	0.0314
Equity(476 [WY])	-0.04860757	0.01225382	-0.04997087	0.00551805	-0.00278578	-0.0303
Equity(477 [WYN])	-0.05535480	0.02642294	0.00304199	-0.00979668	-0.00616717	-0.0291

Equity(478 [WYNN])	-0.06224023	-0.03777009	0.05699233	-0.04984462	0.00328186	-0.0545
Equity(479 [XEC])	-0.06269690	-0.17159838	0.01127251	-0.03690692	0.07210881	0.0423
Equity(480 [XEL])	-0.02213358	0.02037290	-0.09965905	-0.04339570	-0.00356988	-0.0249
Equity(481 [XL])	-0.04533940	0.01070557	-0.03511721	0.04642140	-0.00583933	0.0117
Equity(482 [XLNX])	-0.04210479	-0.00382104	0.05760794	-0.01871999	-0.06972723	-0.0125
Equity(483 [XOM])	-0.03773468	-0.05378131	-0.03367517	-0.01036467	-0.00395082	0.0295
Equity(484 [XRAY])	-0.04417162	0.01778874	-0.00062230	0.00814409	-0.02933804	0.0185
Equity(485 [XRX])	-0.05418096	-0.00344402	0.01002127	0.02970052	-0.04632619	0.0141
Equity(486 [XYL])	-0.02818794	-0.01716654	0.03265037	-0.01947739	-0.00284445	0.0087
Equity(487 [YUM])	-0.03630261	0.02726148	0.00226076	-0.02614444	-0.01418528	0.0013
Equity(488 [ZBH])	-0.03843904	0.01749339	-0.01575190	-0.01540756	-0.00162086	0.0460
Equity(489 [ZION])	-0.06092493	-0.01044197	0.01261759	0.13419161	0.02396471	0.0923
Equity(490 [ZTS])	-0.01367163	0.01892192	0.01867875	-0.04878703	0.01263697	0.0483

490 rows × 20 columns

Factor Returns

Implement `factor_returns` to get the factor returns from the PCA model using the returns data.

In [16]:
```
def factor_returns(pca, returns, factor_return_indices, factor_return_col
    """
    Get the factor returns from the PCA model.

    Parameters
    ----------
    pca : PCA
        Model fit to returns
    returns : DataFrame
        Returns for each ticker and date
    factor_return_indices : 1 dimensional Ndarray
        Factor return indices
    factor_return_columns : 1 dimensional Ndarray
        Factor return columns

    Returns
    -------
    factor_returns : DataFrame
        Factor returns
    """
    assert len(factor_return_indices.shape) == 1
    assert len(factor_return_columns.shape) == 1

    #TODO: Implement function
    transform_returns = pca.transform(returns)
    the_factor_returns = pd.DataFrame(transform_returns, \
                                     index=factor_return_indices, \
                                     columns=factor_return_columns)

    return the_factor_returns


project_tests.test_factor_returns(factor_returns)
```

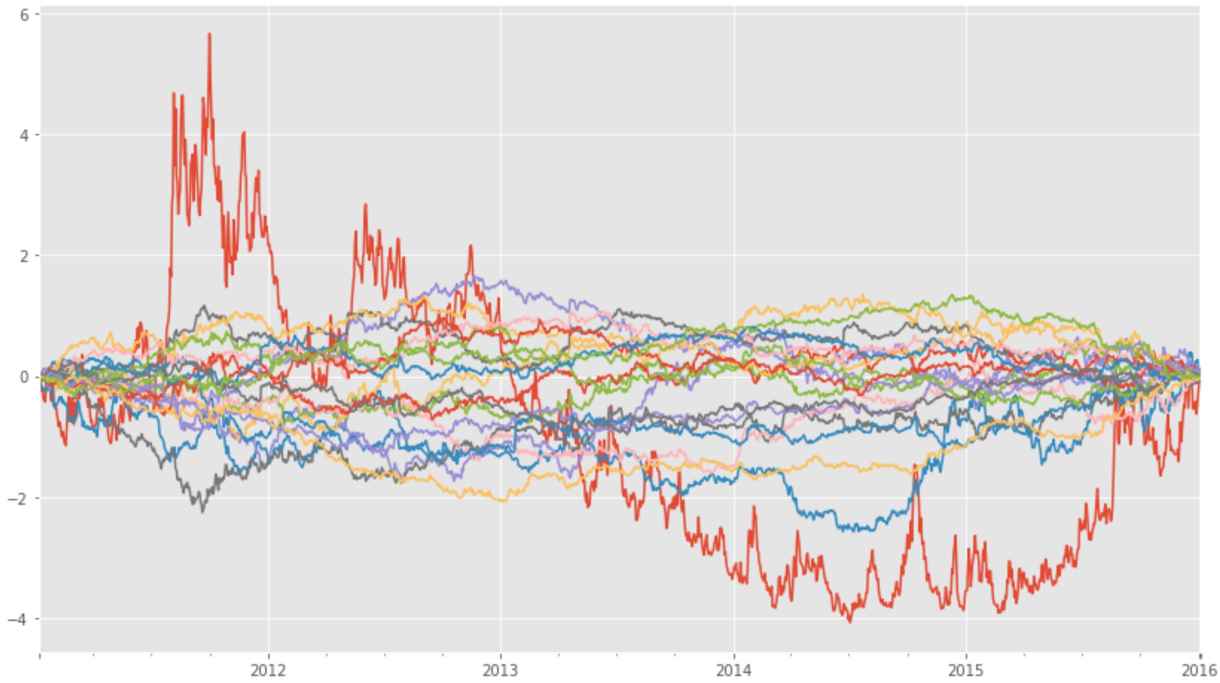
```
Tests Passed
```

View Data

Let's see what these factor returns looks like over time.

In [17]:
```
risk_model['factor_returns'] = factor_returns(
    pca,
    five_year_returns,
    five_year_returns.index,
    np.arange(num_factor_exposures))

risk_model['factor_returns'].cumsum().plot(legend=None)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4d20087ef0>
```



Factor Covariance Matrix

Implement `factor_cov_matrix` to get the factor covariance matrix.

In [18]:
```
def factor_cov_matrix(factor_returns, ann_factor):
    """
    Get the factor covariance matrix

    Parameters
    ----------
    factor_returns : DataFrame
        Factor returns
    ann_factor : int
        Annualization factor

    Returns
    -------
    factor_cov_matrix : 2 dimensional Ndarray
        Factor covariance matrix
    """

    #TODO: Implement function

    return ann_factor * np.diag(np.var(factor_returns, ddof=1))


project_tests.test_factor_cov_matrix(factor_cov_matrix)
```

```
Tests Passed
```

View Data

In [19]:
```
ann_factor = 252
risk_model['factor_cov_matrix'] = factor_cov_matrix(risk_model['factor_re

risk_model['factor_cov_matrix']
```

Out[19]:
```
array([[ 14.01830425,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   1.10591127,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.77099145,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.61798821,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ,
          0.47589087,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.43653315,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.3873247 ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.34930223,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.34350302,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.31674219,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ,
          0.        ,   0.        ,   0.        ,   0.        ],
       [  0.        ,   0.        ,   0.        ,   0.        ,
```

```
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.28186803,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.2762745 ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.26857691,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.24981278,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.23329965,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.21393011,
        0.        ,  0.        ,  0.        ,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.20845473,  0.        ,  0.        ,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.19480492,  0.        ,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.19126517,  0.        ],
      [ 0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.18725609]])
```

Idiosyncratic Variance Matrix

Implement `idiosyncratic_var_matrix` to get the idiosyncratic variance matrix.

In [20]:
```
def idiosyncratic_var_matrix(returns, factor_returns, factor_betas, ann_f
    """
    Get the idiosyncratic variance matrix

    Parameters
    ----------
    returns : DataFrame
        Returns for each ticker and date
    factor_returns : DataFrame
        Factor returns
    factor_betas : DataFrame
        Factor betas
    ann_factor : int
        Annualization factor

    Returns
    -------
    idiosyncratic_var_matrix : DataFrame
        Idiosyncratic variance matrix
    """

    #TODO: Implement function
    common_returns_ = pd.DataFrame(np.dot(factor_returns, factor_betas.T)
                                  index=returns.index, \
                                  columns=returns.columns)
    residuals_ = returns - common_returns_

    return pd.DataFrame(np.diag(np.var(residuals_))*ann_factor, \
                       index=returns.columns, \
                       columns=returns.columns)


project_tests.test_idiosyncratic_var_matrix(idiosyncratic_var_matrix)
```

```
Tests Passed
```

View Data

In [21]:
```
risk_model['idiosyncratic_var_matrix'] = idiosyncratic_var_matrix(five_ye

risk_model['idiosyncratic_var_matrix']
```

Out[21]:

	Equity(0 [A])	Equity(1 [AAL])	Equity(2 [AAP])	Equity(3 [AAPL])	Equity(4 [ABBV])	Equity(5 [ABC])
Equity(0 [A])	0.02272535	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(1 [AAL])	0.00000000	0.05190083	0.00000000	0.00000000	0.00000000	0.00000000
Equity(2 [AAP])	0.00000000	0.00000000	0.05431181	0.00000000	0.00000000	0.00000000
Equity(3 [AAPL])	0.00000000	0.00000000	0.00000000	0.04801884	0.00000000	0.00000000

Equity(4 [ABBV])	0.00000000	0.00000000	0.00000000	0.00000000	0.03040361	0.00000000
Equity(5 [ABC])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.01854504
Equity(6 [ABT])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(7 [ACN])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(8 [ADBE])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(9 [ADI])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(10 [ADM])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(11 [ADP])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(12 [ADS])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(13 [ADSK])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(14 [AEE])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(15 [AEP])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(16 [AES])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(17 [AET])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(18 [AFL])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(19 [AGN])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(20 [AIG])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(21 [AIV])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(22 [AIZ])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(23 [AJG])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(24 [AKAM])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(25 [ALB])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

Equity(26 [ALGN])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(27 [ALK])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(28 [ALL])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(29 [ALLE])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
...	...	...	...	...	...	...
Equity(460 [VRSN])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(461 [VRTX])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(462 [VTR])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(463 [VZ])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(464 [WAT])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(465 [WBA])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(466 [WDC])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(467 [WEC])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(468 [WFC])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(469 [WHR])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(471 [WM])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(472 [WMB])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(473 [WMT])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(474 [WRK])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(475 [WU])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(476 [WY])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(477 [WYN])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

Equity(478 [WYNN])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(479 [XEC])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(480 [XEL])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(481 [XL])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(482 [XLNX])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(483 [XOM])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(484 [XRAY])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(485 [XRX])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(486 [XYL])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(487 [YUM])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(488 [ZBH])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(489 [ZION])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Equity(490 [ZTS])	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

490 rows × 490 columns

Idiosyncratic Variance Vector

Implement `idiosyncratic_var_vector` to get the idiosyncratic variance Vector.

In [22]:
```
def idiosyncratic_var_vector(returns, idiosyncratic_var_matrix):
    """
    Get the idiosyncratic variance vector

    Parameters
    ----------
    returns : DataFrame
        Returns for each ticker and date
    idiosyncratic_var_matrix : DataFrame
        Idiosyncratic variance matrix

    Returns
    -------
    idiosyncratic_var_vector : DataFrame
        Idiosyncratic variance Vector
    """

    #TODO: Implement function
    # This Udacity Knowledge Article helped me to remind what the idiosyn
    # variance vector was ;-)   https://knowledge.udacity.com/questions/1
    return pd.DataFrame(np.diag(idiosyncratic_var_matrix), index=returns.

project_tests.test_idiosyncratic_var_vector(idiosyncratic_var_vector)
```

```
Tests Passed
```

View Data

In [23]:
```
risk_model['idiosyncratic_var_vector'] = idiosyncratic_var_vector(five_ye

risk_model['idiosyncratic_var_vector']
```

Out[23]:

	0
Equity(0 [A])	0.02272535
Equity(1 [AAL])	0.05190083
Equity(2 [AAP])	0.05431181
Equity(3 [AAPL])	0.04801884
Equity(4 [ABBV])	0.03040361
Equity(5 [ABC])	0.01854504
Equity(6 [ABT])	0.01481514
Equity(7 [ACN])	0.02177470
Equity(8 [ADBE])	0.03442125
Equity(9 [ADI])	0.01898404
Equity(10 [ADM])	0.02951444
Equity(11 [ADP])	0.00828126
Equity(12 [ADS])	0.02703428

Equity(13 [ADSK])	0.05224263
Equity(14 [AEE])	0.00961462
Equity(15 [AEP])	0.00800376
Equity(16 [AES])	0.03079578
Equity(17 [AET])	0.03579077
Equity(18 [AFL])	0.01894194
Equity(19 [AGN])	0.04211602
Equity(20 [AIG])	0.03317545
Equity(21 [AIV])	0.01468534
Equity(22 [AIZ])	0.02061951
Equity(23 [AJG])	0.01157320
Equity(24 [AKAM])	0.09690009
Equity(25 [ALB])	0.04106647
Equity(26 [ALGN])	0.10508661
Equity(27 [ALK])	0.03388559
Equity(28 [ALL])	0.01674196
Equity(29 [ALLE])	0.01456581
...	...
Equity(460 [VRSN])	0.04356330
Equity(461 [VRTX])	0.01133104
Equity(462 [VTR])	0.01239277
Equity(463 [VZ])	0.01826124
Equity(464 [WAT])	0.02519289
Equity(465 [WBA])	0.04833797
Equity(466 [WDC])	0.04271307
Equity(467 [WEC])	0.00660139
Equity(468 [WFC])	0.01138111
Equity(469 [WHR])	0.05778786
Equity(471 [WM])	0.01511105
Equity(472 [WMB])	0.05868578
Equity(473 [WMT])	0.01608075
Equity(474 [WRK])	0.00835928
Equity(475 [WU])	0.04770344
Equity(476 [WY])	0.02646036

Equity(477 [WYN])	0.02429392
Equity(478 [WYNN])	0.05814062
Equity(479 [XEC])	0.04861817
Equity(480 [XEL])	0.00534852
Equity(481 [XL])	0.02051926
Equity(482 [XLNX])	0.02684299
Equity(483 [XOM])	0.01059841
Equity(484 [XRAY])	0.01537171
Equity(485 [XRX])	0.03946866
Equity(486 [XYL])	0.03191603
Equity(487 [YUM])	0.04385518
Equity(488 [ZBH])	0.02233019
Equity(489 [ZION])	0.02337210
Equity(490 [ZTS])	0.02735075

490 rows × 1 columns

Predict using the Risk Model

Using the data we calculated in the risk model, implement `predict_portfolio_risk` to predict the portfolio risk using the formula $ \sqrt{X^{T}(BFB^{T} + S)X} $ where:

- $ X $ is the portfolio weights
- $ B $ is the factor betas
- $ F $ is the factor covariance matrix
- $ S $ is the idiosyncratic variance matrix

In [24]:
```
def predict_portfolio_risk(factor_betas, factor_cov_matrix, idiosyncratic_
    """
    Get the predicted portfolio risk

    Formula for predicted portfolio risk is sqrt(X.T(BFB.T + S)X) where:
      X is the portfolio weights
      B is the factor betas
      F is the factor covariance matrix
      S is the idiosyncratic variance matrix

    Parameters
    ----------
    factor_betas : DataFrame
        Factor betas
    factor_cov_matrix : 2 dimensional Ndarray
        Factor covariance matrix
    idiosyncratic_var_matrix : DataFrame
        Idiosyncratic variance matrix
    weights : DataFrame
        Portfolio weights

    Returns
    -------
    predicted_portfolio_risk : float
        Predicted portfolio risk
    """
    assert len(factor_cov_matrix.shape) == 2

    #TODO: Implement function
    returns = np.sqrt(np.dot(weights.T,\
                         np.dot(factor_betas, factor_cov_matrix)\
                         .dot(factor_betas.T)+idiosyncratic_var_matrix)\
                      .dot(weights))[0, 0]
    print(returns)
    return returns


project_tests.test_predict_portfolio_risk(predict_portfolio_risk)
```

```
0.0550369570517
Tests Passed
```

View Data

Let's see what the portfolio risk would be if we had even weights across all stocks.

In [25]:
```
all_weights = pd.DataFrame(np.repeat(1/len(universe_tickers), len(univers

predict_portfolio_risk(
    risk_model['factor_betas'],
    risk_model['factor_cov_matrix'],
    risk_model['idiosyncratic_var_matrix'],
    all_weights)
```

```
0.16094824687
```

Out[25]: 0.16094824687040468

Create Alpha Factors

With the profile risk calculated, it's time to start working on the alpha factors. In this project, we'll create the following factors:

- Momentum 1 Year Factor
- Mean Reversion 5 Day Sector Neutral Factor
- Mean Reversion 5 Day Sector Neutral Smoothed Factor
- Overnight Sentiment Factor
- Overnight Sentiment Smoothed Factor

Momentum 1 Year Factor

Each factor will have a hypothesis that goes with it. For this factor, it is "Higher past 12-month (252 days) returns are proportional to future return." Using that hypothesis, we've generated this code:

In [26]:
```
from zipline.pipeline.factors import Returns

def momentum_1yr(window_length, universe, sector):
    return Returns(window_length=window_length, mask=universe) \
        .demean(groupby=sector) \
        .rank() \
        .zscore()
```

Mean Reversion 5 Day Sector Neutral Factor

Now it's time for you to implement `mean_reversion_5day_sector_neutral` using the hypothesis "Short-term outperformers(underperformers) compared to their sector will revert." Use the returns data from `universe`, demean using the sector data to partition, rank, then converted to a zscore.

In [27]:
```
def mean_reversion_5day_sector_neutral(window_length, universe, sector):
    """
    Generate the mean reversion 5 day sector neutral factor

    Parameters
    ----------
    window_length : int
        Returns window length
    universe : Zipline Filter
        Universe of stocks filter
    sector : Zipline Classifier
        Sector classifier

    Returns
    -------
    factor : Zipline Factor
        Mean reversion 5 day sector neutral factor
    """

    #TODO: Implement function
    # This Udacity Knowledge Article helped me to understand the
    # meaning of the term 'revert'
    factor = -Returns(window_length=window_length,mask=universe) \
        .demean(groupby=sector) \
        .rank() \
        .zscore()

    return factor


project_tests.test_mean_reversion_5day_sector_neutral(mean_reversion_5day_
```

```
Running Integration Test on pipeline:
> start_dat = pd.Timestamp('2015-01-05', tz='utc')
> end_date = pd.Timestamp('2015-01-07', tz='utc')
> universe = AverageDollarVolume(window_length=2).top(4)
> factor = mean_reversion_5day_sector_neutral(
    window_length=3,
    universe=universe,
    sector=project_helper.Sector())
> pipeline.add(factor, 'Mean_Reversion_5Day_Sector_Neutral')
> engine.run_pipeline(pipeline, start_dat, end_date)

Tests Passed
```

View Data

Let's see what some of the factor data looks like. For calculating factors, we'll be looking back 2 years.

Note: *Going back 2 years falls on a day when the market is closed. Pipeline package doesn't handle start or end dates that don't fall on days when the market is open. To fix this, we went back 2 extra days to fall on the next day when the market is open.*

In [28]:
```
factor_start_date = universe_end_date - pd.DateOffset(years=2, days=2)
sector = project_helper.Sector()
window_length = 5

pipeline = Pipeline(screen=universe)
pipeline.add(
    mean_reversion_5day_sector_neutral(window_length, universe, sector),
    'Mean_Reversion_5Day_Sector_Neutral')
engine.run_pipeline(pipeline, factor_start_date, universe_end_date)
```

Out[28]:

		Mean_Reversion_5Day_Sector_Neutral
2014-01-03 00:00:00+00:00	Equity(0 [A])	0.85326482
	Equity(1 [AAL])	1.62630815
	Equity(2 [AAP])	0.64906469
	Equity(3 [AAPL])	1.40752230
	Equity(4 [ABBV])	1.45857233
	Equity(5 [ABC])	0.14585723
	Equity(6 [ABT])	-0.30630019
	Equity(7 [ACN])	0.88243626
	Equity(8 [ADBE])	-0.06563576
	Equity(9 [ADI])	1.67006532
	Equity(10 [ADM])	1.18144359
	Equity(11 [ADP])	0.24066444
	Equity(12 [ADS])	-1.69194391
	Equity(13 [ADSK])	-0.35735022
	Equity(14 [AEE])	0.34276450
	Equity(15 [AEP])	-0.45215742
	Equity(16 [AES])	0.50320746
	Equity(17 [AET])	0.79492192
	Equity(18 [AFL])	1.15227214
	Equity(19 [AGN])	-1.48045092
	Equity(20 [AIG])	-0.27712874
	Equity(21 [AIV])	-0.37922881
	Equity(22 [AIZ])	0.91890057
	Equity(23 [AJG])	-0.82409337
	Equity(24 [AKAM])	1.22520076
	Equity(25 [ALB])	0.69282186
	Equity(26 [ALGN])	1.08663639

	Equity(27 [ALK])	-1.30542224
	Equity(28 [ALL])	-0.48862173
	Equity(29 [ALLE])	1.42940089
...	**...**	...
2016-01-05 00:00:00+00:00	**Equity(460 [VRSN])**	1.48614551
	Equity(461 [VRTX])	-1.15826688
	Equity(462 [VTR])	-1.55742347
	Equity(463 [VZ])	-1.42912314
	Equity(464 [WAT])	0.47399845
	Equity(465 [WBA])	1.21528925
	Equity(466 [WDC])	-1.59306245
	Equity(467 [WEC])	-0.13899203
	Equity(468 [WFC])	0.35995371
	Equity(469 [WHR])	-1.15113908
	Equity(471 [WM])	0.38133710
	Equity(472 [WMB])	-1.69285160
	Equity(473 [WMT])	-1.52891228
	Equity(474 [WRK])	-1.32220619
	Equity(475 [WU])	0.54527641
	Equity(476 [WY])	0.08196966
	Equity(477 [WYN])	1.19390586
	Equity(478 [WYNN])	-1.49327330
	Equity(479 [XEC])	0.30293134
	Equity(480 [XEL])	0.02494729
	Equity(481 [XL])	1.06560553
	Equity(482 [XLNX])	1.34358958
	Equity(483 [XOM])	1.60019024
	Equity(484 [XRAY])	1.22241705
	Equity(485 [XRX])	0.33144252
	Equity(486 [XYL])	-0.13186423
	Equity(487 [YUM])	0.18175880

	Equity(488 [ZBH])	-1.40061195
	Equity(489 [ZION])	0.52389302
	Equity(490 [ZTS])	-0.93730520

244259 rows × 1 columns

Mean Reversion 5 Day Sector Neutral Smoothed Factor

Taking the output of the previous factor, let's create a smoothed version. Implement `mean_reversion_5day_sector_neutral_smoothed` to generate a mean reversion 5 fay sector neutral smoothed factor. Call the `mean_reversion_5day_sector_neutral` function to get the unsmoothed factor, then use `SimpleMovingAverage` function to smooth it. You'll have to apply rank and zscore again.

In [29]:
```
from zipline.pipeline.factors import SimpleMovingAverage

def mean_reversion_5day_sector_neutral_smoothed(window_length, universe,
    """
    Generate the mean reversion 5 day sector neutral smoothed factor

    Parameters
    ----------
    window_length : int
        Returns window length
    universe : Zipline Filter
        Universe of stocks filter
    sector : Zipline Classifier
        Sector classifier

    Returns
    -------
    factor : Zipline Factor
        Mean reversion 5 day sector neutral smoothed factor
    """

    #TODO: Implement function
    factor = SimpleMovingAverage(inputs=[mean_reversion_5day_sector_neutr
                                 window_length=window_length, \
                                 mask=universe) \
        .rank() \
        .zscore()

    return factor


project_tests.test_mean_reversion_5day_sector_neutral_smoothed(mean_rever
```

```
Running Integration Test on pipeline:
> start_dat = pd.Timestamp('2015-01-05', tz='utc')
> end_date = pd.Timestamp('2015-01-07', tz='utc')
> universe = AverageDollarVolume(window_length=2).top(4)
> factor = mean_reversion_5day_sector_neutral_smoothed(
    window_length=3,
    universe=universe,
    sector=project_helper.Sector())
> pipeline.add(factor, 'Mean_Reversion_5Day_Sector_Neutral_Smoothed')
> engine.run_pipeline(pipeline, start_dat, end_date)

Tests Passed
```

View Data

Let's see what some of the smoothed data looks like.

In [30]:
```
pipeline = Pipeline(screen=universe)
pipeline.add(
    mean_reversion_5day_sector_neutral_smoothed(5, universe, sector),
    'Mean_Reversion_5Day_Sector_Neutral_Smoothed')
engine.run_pipeline(pipeline, factor_start_date, universe_end_date)
```

Out[30]:

	Mean_Reversion_5Day_Sector_Neutral_Smoothed
Equity(0 [A])	1.11580784
Equity(1 [AAL])	1.72840822
Equity(2 [AAP])	1.34188655
Equity(3 [AAPL])	0.91160771
Equity(4 [ABBV])	0.96265774
Equity(5 [ABC])	0.77304334
Equity(6 [ABT])	0.48862173
Equity(7 [ACN])	-0.45945029
Equity(8 [ADBE])	0.81680051
Equity(9 [ADI])	0.94807202
Equity(10 [ADM])	0.73657903
Equity(11 [ADP])	0.32088591

	Equity(12 [ADS])	-1.59713671
	Equity(13 [ADSK])	0.08022148
	Equity(14 [AEE])	0.11668579
2014-01-03 00:00:00+00:00	**Equity(15 [AEP])**	0.21878585
	Equity(16 [AES])	-0.75116475
	Equity(17 [AET])	-1.09392925
	Equity(18 [AFL])	-0.07292862
	Equity(19 [AGN])	-0.37922881
	Equity(20 [AIG])	1.05017208
	Equity(21 [AIV])	-0.14585723
	Equity(22 [AIZ])	0.67094327
	Equity(23 [AJG])	-0.56155035
	Equity(24 [AKAM])	1.67735818
	Equity(25 [ALB])	1.35647227
	Equity(26 [ALGN])	1.32730082
	Equity(27 [ALK])	0.57613607
	Equity(28 [ALL])	0.02187859
	Equity(29 [ALLE])	1.65547960
...	**...**	...
	Equity(460 [VRSN])	1.12262790
	Equity(461 [VRTX])	-1.62157363
	Equity(462 [VTR])	-1.60731804

	Equity(463 [VZ])	-1.68572380
	Equity(464 [WAT])	-1.00858316
	Equity(465 [WBA])	0.98007198
	Equity(466 [WDC])	-1.22241705
	Equity(467 [WEC])	-0.78762148
	Equity(468 [WFC])	0.72347131
	Equity(469 [WHR])	0.37420930
	Equity(471 [WM])	-0.53814861
	Equity(472 [WMB])	-1.65721261
	Equity(473 [WMT])	-1.41486754
	Equity(474 [WRK])	-1.46476212
	Equity(475 [WU])	1.55742347
2016-01-05 00:00:00+00:00	Equity(476 [WY])	-0.41697608
	Equity(477 [WYN])	0.48112624
	Equity(478 [WYNN])	-1.36497297
	Equity(479 [XEC])	-0.47399845
	Equity(480 [XEL])	-0.95156079
	Equity(481 [XL])	-0.76623809
	Equity(482 [XLNX])	1.47188991
	Equity(483 [XOM])	1.08698892
	Equity(484 [XRAY])	0.10335304
	Equity(485 [XRX])	0.98719977

	Equity(486 [XYL])	0.66644894
	Equity(487 [YUM])	1.07273333
	Equity(488 [ZBH])	-0.13186423
	Equity(489 [ZION])	0.31718693
	Equity(490 [ZTS])	0.16750321

244259 rows × 1 columns

Overnight Sentiment Factor

For this factor, were using the hypothesis from the paper *Overnight Returns and Firm-Specific Investor Sentiment*.

In [31]:
```
from zipline.pipeline.data import USEquityPricing


class CTO(Returns):
    """
    Computes the overnight return, per hypothesis from
    https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2554010
    """
    inputs = [USEquityPricing.open, USEquityPricing.close]

    def compute(self, today, assets, out, opens, closes):
        """
        The opens and closes matrix is 2 rows x N assets, with the most r
        As such, opens[-1] is the most recent open, and closes[0] is the
        """
        out[:] = (opens[-1] - closes[0]) / closes[0]


class TrailingOvernightReturns(Returns):
    """
    Sum of trailing 1m O/N returns
    """
    window_safe = True

    def compute(self, today, asset_ids, out, cto):
        out[:] = np.nansum(cto, axis=0)


def overnight_sentiment(cto_window_length, trail_overnight_returns_window
    cto_out = CTO(mask=universe, window_length=cto_window_length)
    return TrailingOvernightReturns(inputs=[cto_out], window_length=trail
        .rank() \
        .zscore()
```

Overnight Sentiment Smoothed Factor

Just like the factor you implemented, we'll also smooth this factor.

In [32]:
```
def overnight_sentiment_smoothed(cto_window_length, trail_overnight_retur
    unsmoothed_factor = overnight_sentiment(cto_window_length, trail_over
    return SimpleMovingAverage(inputs=[unsmoothed_factor], window_length=
        .rank() \
        .zscore()
```

Combine the Factors to a single Pipeline

With all the factor implementations done, let's add them to a pipeline.

In [33]:
```
universe = AverageDollarVolume(window_length=120).top(500)
sector = project_helper.Sector()

pipeline = Pipeline(screen=universe)
pipeline.add(
    momentum_1yr(252, universe, sector),
    'Momentum_1YR')
pipeline.add(
    mean_reversion_5day_sector_neutral(5, universe, sector),
    'Mean_Reversion_5Day_Sector_Neutral')
pipeline.add(
    mean_reversion_5day_sector_neutral_smoothed(5, universe, sector),
    'Mean_Reversion_5Day_Sector_Neutral_Smoothed')
pipeline.add(
    overnight_sentiment(2, 5, universe),
    'Overnight_Sentiment')
pipeline.add(
    overnight_sentiment_smoothed(2, 5, universe),
    'Overnight_Sentiment_Smoothed')
all_factors = engine.run_pipeline(pipeline, factor_start_date, universe_e

all_factors.head()
```

Out[33]:

		Mean_Reversion_5Day_Sector_Neutral	Mean_Reversion_5Day_
2014-01-03 00:00:00+00:00	Equity(0 [A])	0.85326482	
	Equity(1 [AAL])	1.62630815	
	Equity(2 [AAP])	0.64906469	
	Equity(3 [AAPL])	1.40752230	
	Equity(4 [ABBV])	1.45857233	

Evaluate Alpha Factors

Note: We're evaluating the alpha factors using delay of 1

Get Pricing Data

In [34]:
```
import alphalens as al

assets = all_factors.index.levels[1].values.tolist()
pricing = get_pricing(
    data_portal,
    trading_calendar,
    assets,
    factor_start_date,
    universe_end_date)
```

Format alpha factors and pricing for Alphalens

In order to use a lot of the alphalens functions, we need to aligned the indices and convert the time to unix timestamp. In this next cell, we'll do just that.

In [35]:
```
clean_factor_data = {
    factor: al.utils.get_clean_factor_and_forward_returns(factor=factor_d
    for factor, factor_data in all_factors.iteritems()}

unixt_factor_data = {
    factor: factor_data.set_index(pd.MultiIndex.from_tuples(
        [(x.timestamp(), y) for x, y in factor_data.index.values],
        names=['date', 'asset']))
    for factor, factor_data in clean_factor_data.items()}
```

```
Dropped 1.2% entries from factor data: 1.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 1.2% entries from factor data: 1.2% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 2.3% entries from factor data: 2.3% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.4% entries from factor data: 0.4% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
Dropped 0.4% entries from factor data: 0.4% in forward returns computatio
n and 0.0% in binning phase (set max_loss=0 to see potentially suppressed
Exceptions).
max_loss is 35.0%, not exceeded: OK!
```

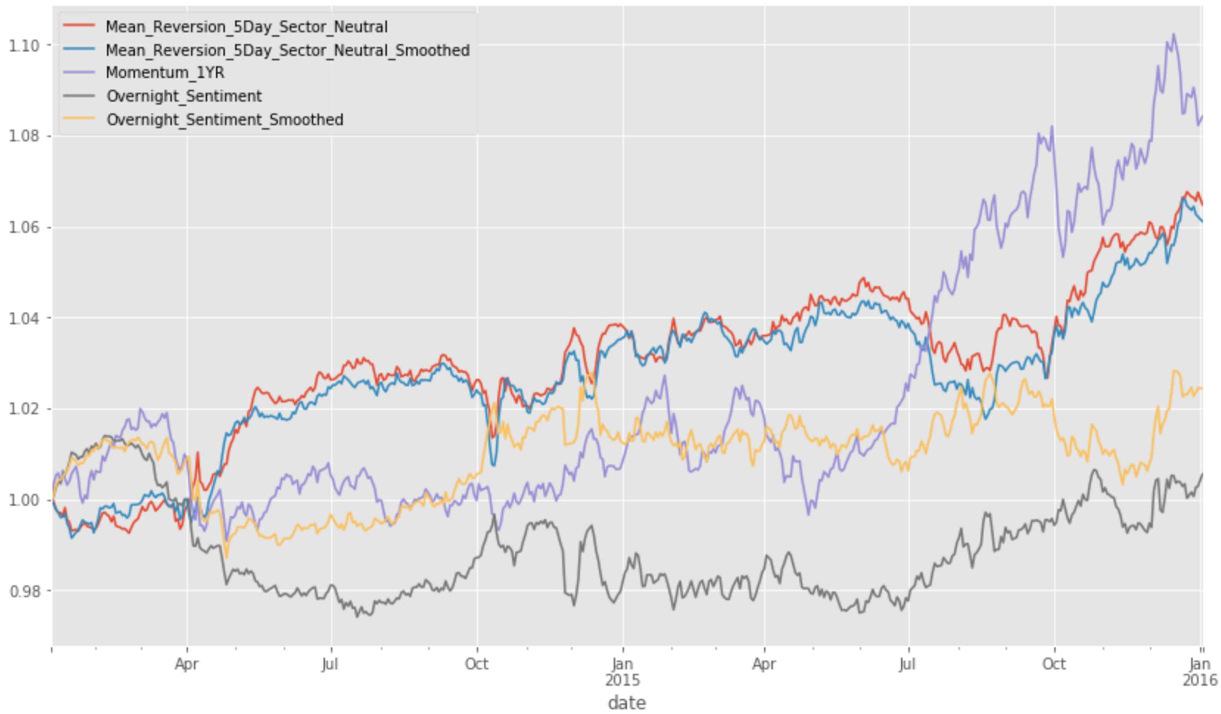
Quantile Analysis

Factor Returns

Let's view the factor returns over time. We should be seeing it generally move up and to the right.

In [36]:
```
ls_factor_returns = pd.DataFrame()

for factor, factor_data in clean_factor_data.items():
    ls_factor_returns[factor] = al.performance.factor_returns(factor_data

(1+ls_factor_returns).cumprod().plot()
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4d150914e0>

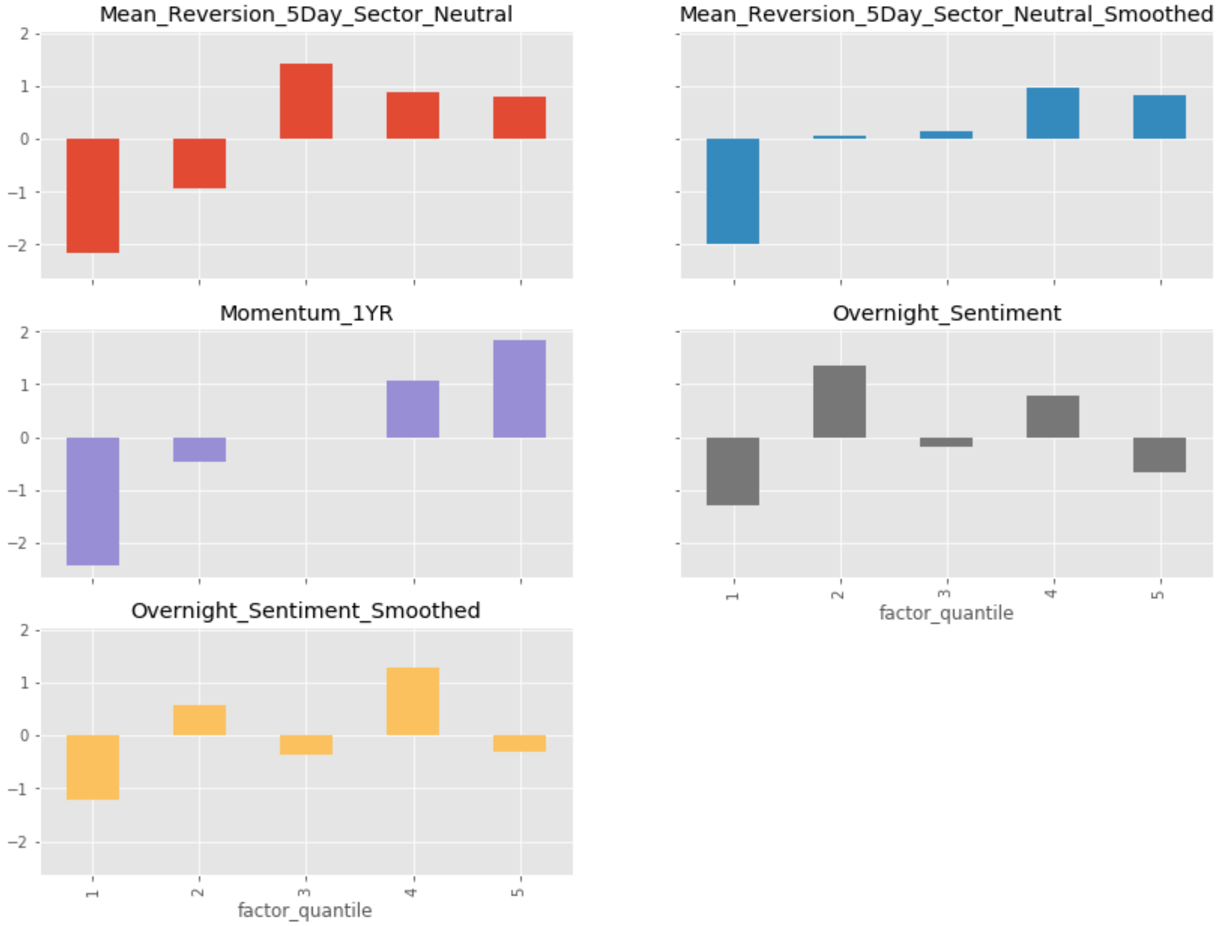


Basis Points Per Day per Quantile

It is not enough to look just at the factor weighted return. A good alpha is also monotonic in quantiles. Let's looks the basis points for the factor returns.

In [37]:
```
qr_factor_returns = pd.DataFrame()

for factor, factor_data in unixt_factor_data.items():
    qr_factor_returns[factor] = al.performance.mean_return_by_quantile(fa

(10000*qr_factor_returns).plot.bar(
    subplots=True,
    sharey=True,
    layout=(4,2),
    figsize=(14, 14),
    legend=False)
```

Out[37]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f4d10570748>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f4d10494160
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f4d10437f28>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f4d1046b898
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f4d0f0c89b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f4d0f0c8240
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f4d10352550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f4d10580160
>]], dtype=object)
```



What do you observe?

- None of these alphas are **strictly monotonic**; this should lead you to question why this is? Further research and refinement of the alphas needs to be done. What is it about these alphas that leads to the highest ranking stocks in all alphas except MR 5D smoothed to *not* perform the best.
- The majority of the return is coming from the **short side** in all these alphas. The negative return in quintile 1 is very large in all alphas. This could also a cause for concern becuase when you short stocks, you need to locate the short; shorts can be expensive or not available at all.
- If you look at the magnitude of the return spread (i.e., Q1 minus Q5), we are working with daily returns in the 0.03%, i.e., **3 basis points**, neighborhood *before all transaction costs, shorting costs, etc.*. Assuming 252 days in a year, that's 7.56% return annualized. Transaction costs may cut this in half. As such, it should be clear that these alphas can only survive in an institutional setting and that leverage will likely need to be applied in order to achieve an attractive return.

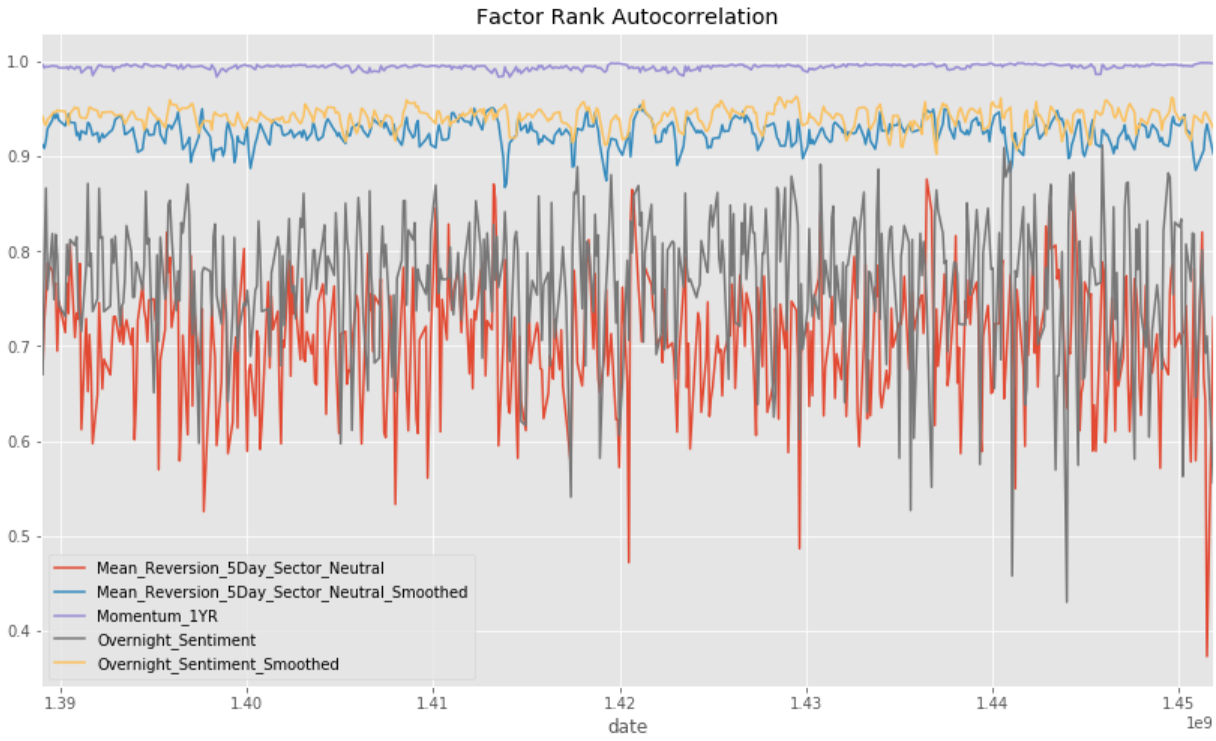
Turnover Analysis

Without doing a full and formal backtest, we can analyze how stable the alphas are over time. Stability in this sense means that from period to period, the alpha ranks do not change much. Since trading is costly, we always prefer, all other things being equal, that the ranks do not change significantly per period. We can measure this with the **factor rank autocorrelation (FRA)**.

alphalens.performance.factor_rank_autocorrelation

In [38]:
```
ls_FRA = pd.DataFrame()

for factor, factor_data in unixt_factor_data.items():
    ls_FRA[factor] = al.performance.factor_rank_autocorrelation(factor_da

ls_FRA.plot(title="Factor Rank Autocorrelation")
```

Out[38]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4d101b7630>
```



Sharpe Ratio of the Alphas

The last analysis we'll do on the factors will be sharpe ratio. Implement `sharpe_ratio` to calculate the sharpe ratio of factor returns.

In [39]:
```
def sharpe_ratio(factor_returns, annualization_factor):
    """
    Get the sharpe ratio for each factor for the entire period

    Parameters
    ----------
    factor_returns : DataFrame
        Factor returns for each factor and date
    annualization_factor: float
        Annualization Factor

    Returns
    -------
    sharpe_ratio : Pandas Series of floats
        Sharpe ratio
    """

    #TODO: Implement function
    #This solution did not work, why?
    #sharpe_ratio = np.mean(factor_returns)/np.std(factor_returns) * annu
    #This is the solution that works
    sharpe_ratio = annualization_factor*factor_returns.mean()/factor_retu

    return sharpe_ratio


project_tests.test_sharpe_ratio(sharpe_ratio)
```

```
Tests Passed
```

View Data

Let's see what the sharpe ratio for the factors are. Generally, a Sharpe Ratio of near 1.0 or higher is an acceptable single alpha for this universe.

In [40]:
```
daily_annualization_factor = np.sqrt(252)
sharpe_ratio(ls_factor_returns, daily_annualization_factor).round(2)
```

Out[40]:
```
Mean_Reversion_5Day_Sector_Neutral            1.37000000
Mean_Reversion_5Day_Sector_Neutral_Smoothed   1.27000000
Momentum_1YR                                  1.13000000
Overnight_Sentiment                           0.12000000
Overnight_Sentiment_Smoothed                  0.45000000
dtype: float64
```

Question: What do you think would happen if we smooth the momentum factor? Would the performance increase, decrease, or no major change? Why?

#TODO: Put Answer In this Cell

Answer: If we'd smooth the Momentum factor, there would be -- in sum -- no major change to the overall performance. This is due to the fact that the unsmoothed Momentum factor has already a nearly-ideal appearance: it firstly has a mean that is nearly equal to one; and it has secondly only few and very tiny deviations from that mean. So, smoothing this factor would only very minimally reduce its variations, and bring its mean only very minimally nearer to 1.0. In sum, smoothing the factor would not introduce any sensible change.

The Combined Alpha Vector

To use these alphas in a portfolio, we need to combine them somehow so we get a single score per stock. This is a area where machine learning can be very helpful. In this module, however, we will take the simplest approach of combination: simply averaging the scores from each alpha.

In [41]:
```
selected_factors = all_factors.columns[[1, 2, 4]]
print('Selected Factors: {}'.format(', '.join(selected_factors)))

all_factors['alpha_vector'] = all_factors[selected_factors].mean(axis=1)
alphas = all_factors[['alpha_vector']]
alpha_vector = alphas.loc[all_factors.index.get_level_values(0)[-1]]
alpha_vector.head()
```

```
Selected Factors: Mean_Reversion_5Day_Sector_Neutral_Smoothed, Momentum_1
YR, Overnight_Sentiment_Smoothed
```

Out[41]:

	alpha_vector
Equity(0 [A])	-0.58642457
Equity(1 [AAL])	-0.45333845
Equity(2 [AAP])	-0.69993898
Equity(3 [AAPL])	-0.06790952
Equity(4 [ABBV])	-1.21617871

Optimal Portfolio Constrained by Risk Model

You have an alpha model and a risk model. Let's find a portfolio that trades as close as possible to the alpha model but limiting risk as measured by the risk model. You'll be building thie optimizer for this portfolio. To help you out. we have provided you with an abstract class called `AbstractOptimalHoldings`.

In [42]:
```
from abc import ABC, abstractmethod


class AbstractOptimalHoldings(ABC):
    @abstractmethod
    def _get_obj(self, weights, alpha_vector):
        """
        Get the objective function

        Parameters
        ----------
        weights : CVXPY Variable
            Portfolio weights
        alpha_vector : DataFrame
            Alpha vector

        Returns
        -------
        objective : CVXPY Objective
            Objective function
        """

        raise NotImplementedError()

    @abstractmethod
    def _get_constraints(self, weights, factor_betas, risk):
        """
        Get the constraints

        Parameters
        ----------
        weights : CVXPY Variable
            Portfolio weights
        factor_betas : 2 dimensional Ndarray
            Factor betas
        risk: CVXPY Atom
            Predicted variance of the portfolio returns

        Returns
        -------
        constraints : List of CVXPY Constraint
            Constraints
        """

        raise NotImplementedError()

    def _get_risk(self, weights, factor_betas, alpha_vector_index, factor
        f = factor_betas.loc[alpha_vector_index].values.T * weights
        X = factor_cov_matrix
        S = np.diag(idiosyncratic_var_vector.loc[alpha_vector_index].valu

        return cvx.quad_form(f, X) + cvx.quad_form(weights, S)

    def find(self, alpha_vector, factor_betas, factor_cov_matrix, idiosyn
        weights = cvx.Variable(len(alpha_vector))
        risk = self._get_risk(weights, factor_betas, alpha_vector.index,
```

```
        obj = self._get_obj(weights, alpha_vector)
        constraints = self._get_constraints(weights, factor_betas.loc[alp

        prob = cvx.Problem(obj, constraints)
        prob.solve(max_iters=500)

        optimal_weights = np.asarray(weights.value).flatten()

        return pd.DataFrame(data=optimal_weights, index=alpha_vector.inde
```

Objective and Constraints

Using this class as a base class, you'll implement the `OptimalHoldings` class. There's two functions that need to be implemented in this class, the `_get_obj` and `_get_constraints` functions.

The `_get_obj` function should return an CVXPY objective function that maximizes $ \alpha^T * x \\ $, where $ x $ is the portfolio weights and $ \alpha $ is the alpha vector.

The `_get_constraints` function should return a list of the following constraints:

- $ r \leq risk_{\text{cap}}^2 \\ $
- $ B^T * x \preceq factor_{\text{max}} \\ $
- $ B^T * x \succeq factor_{\text{min}} \\ $
- $ x^T\mathbb{1} = 0 \\ $
- $ \|x\|_1 \leq 1 \\ $
- $ x \succeq weights_{\text{min}} \\ $
- $ x \preceq weights_{\text{max}} $

Where $ x $ is the portfolio weights, $ B $ is the factor betas, and $ r $ is the portfolio risk

The first constraint is that the predicted risk be less than some maximum limit. The second and third constraints are on the maximum and minimum portfolio factor exposures. The fourth constraint is the "market neutral constraint: the sum of the weights must be zero. The fifth constraint is the leverage constraint: the sum of the absolute value of the weights must be less than or equal to 1.0. The last are some minimum and maximum limits on individual holdings.

In [43]:
```
class OptimalHoldings(AbstractOptimalHoldings):
    def _get_obj(self, weights, alpha_vector):
        """
        Get the objective function

        Parameters
```

```
        ----------
        weights : CVXPY Variable
            Portfolio weights
        alpha_vector : DataFrame
            Alpha vector

        Returns
        -------
        objective : CVXPY Objective
            Objective function
        """
        assert(len(alpha_vector.columns) == 1)

        #TODO: Implement function
        #x = cvx.Variable(len(alpha_vector))
        objective = cvx.Minimize(alpha_vector.values.flatten()*weights*-1

        return objective

    def _get_constraints(self, weights, factor_betas, risk):
        """
        Get the constraints

        Parameters
        ----------
        weights : CVXPY Variable
            Portfolio weights
        factor_betas : 2 dimensional Ndarray
            Factor betas
        risk: CVXPY Atom
            Predicted variance of the portfolio returns

        Returns
        -------
        constraints : List of CVXPY Constraint
            Constraints
        """
        assert(len(factor_betas.shape) == 2)

        #TODO: Implement function
        constraints = []

        risk_constraint = risk <= (self.risk_cap**2)
        constraints.append(risk_constraint)

        # note that the max value applies to each of the elements in the
        factor_max_constraint = (factor_betas.T*weights) <= self.factor_m
        constraints.append(factor_max_constraint)

        # note that the min value applies to each of the elements in the
        factor_min_constraint = (factor_betas.T*weights) >= self.factor_m
        constraints.append(factor_min_constraint)

        market_neutral_constraint = sum(weights.T) == 0.0
        constraints.append(market_neutral_constraint)

        leverage_constraint = sum(cvx.abs(weights)) <= 1.0
```

```
        constraints.append(leverage_constraint)

        weights_min_constraint = weights >= self.weights_min
        constraints.append(weights_min_constraint)

        weights_max_constraint = weights <= self.weights_max
        constraints.append(weights_max_constraint)

        return constraints

    def __init__(self, risk_cap=0.05, factor_max=10.0, factor_min=-10.0,
        self.risk_cap=risk_cap
        self.factor_max=factor_max
        self.factor_min=factor_min
        self.weights_max=weights_max
        self.weights_min=weights_min


project_tests.test_optimal_holdings_get_obj(OptimalHoldings)
project_tests.test_optimal_holdings_get_constraints(OptimalHoldings)
```

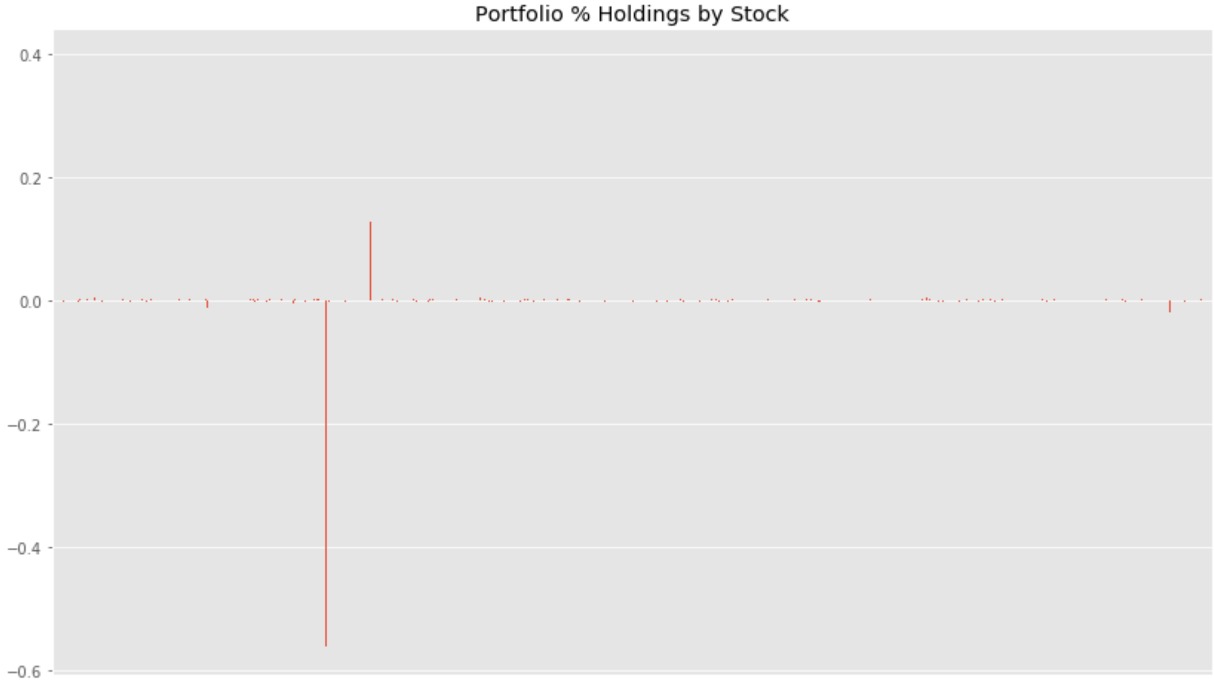
```
Running Integration Test on Problem.solve:
> constaints = [sum(weights) == 0.0, sum(cvx.abs(weights)) <= 1.0]
> obj = optimal_holdings._get_obj(weights, alpha_vector)
> prob = cvx.Problem(obj, constaints)
> prob.solve(max_iters=500)
> solution = np.asarray(weights.value).flatten()

Tests Passed

Running Integration Test on Problem.solve:
> x = np.diag(np.arange(3))
> s = np.diag(np.arange(4))
> factor_betas = np.arange(4 * 3).reshape([4, 3])
> risk = cvx.quad_form(weights * factor_betas, x) + cvx.quad_form(weight
s, s)
> constaints = optimal_holdings._get_constraints(weights, factor_betas, r
isk)
> obj = cvx.Maximize([0, 1, 5, -1] * weights)
> prob = cvx.Problem(obj, constaints)
> prob.solve(max_iters=500)
> solution = np.asarray(weights.value).flatten()

Tests Passed
```

View Data

With the `OptimalHoldings` class implemented, let's see the weights it generates.

In [45]:
```
optimal_weights = OptimalHoldings().find(alpha_vector, risk_model['factor

optimal_weights.plot.bar(legend=None, title='Portfolio % Holdings by Stoc
x_axis = plt.axes().get_xaxis()
x_axis.set_visible(False)
```

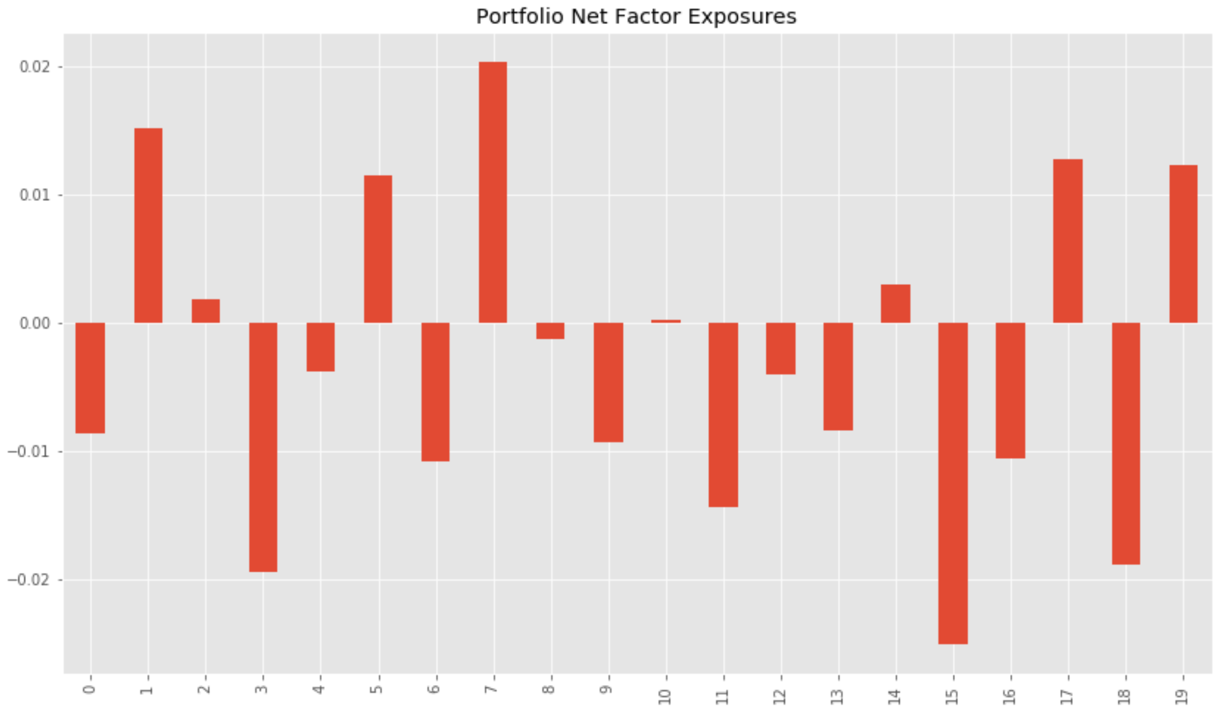
```
/opt/conda/lib/python3.6/site-packages/matplotlib/cbook/deprecation.py:10
6: MatplotlibDeprecationWarning: Adding an axes using the same arguments
as a previous axes currently reuses the earlier instance.  In a future ve
rsion, a new instance will always be created and returned.  Meanwhile, th
is warning can be suppressed, and the future behavior ensured, by passing
a unique label to each axes instance.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```



Yikes. It put most of the weight in a few stocks.

In [46]:
```
project_helper.get_factor_exposures(risk_model['factor_betas'], optimal_w
    title='Portfolio Net Factor Exposures',
    legend=False)
```

Out[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4d064788d0>
```



Optimize with a Regularization Parameter

In order to enforce diversification, we'll use regularization in the objective function. We'll create a new class called `OptimalHoldingsRegualization` which gets its constraints from the `OptimalHoldings` class. In this new class, implement the `_get_obj` function to return a CVXPY objective function that maximize $ \alpha^T * x + \lambda\|x\|_2\\ $, where $ x $ is the portfolio weights, $ \alpha $ is the alpha vector, and $ \lambda $ is the regularization parameter.

Note: *$ |lambda $ is located in* `self.lambda_reg`.

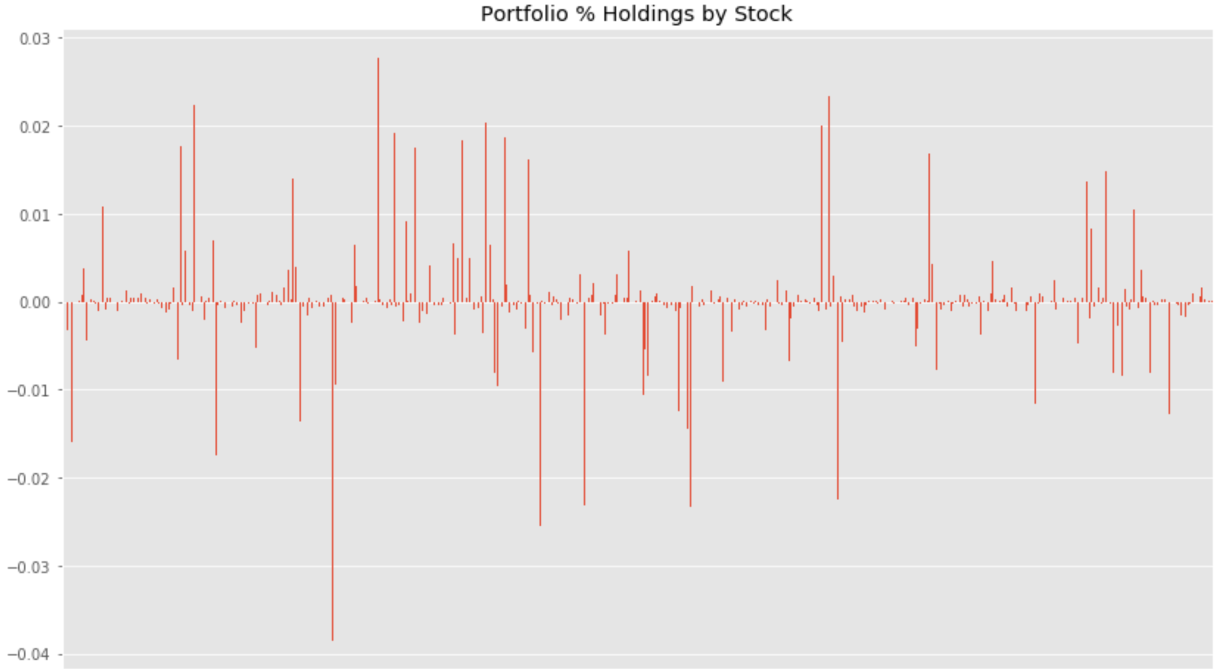
In [49]:
```
class OptimalHoldingsRegualization(OptimalHoldings):
    def _get_obj(self, weights, alpha_vector):
        """
        Get the objective function

        Parameters
        ----------
        weights : CVXPY Variable
            Portfolio weights
        alpha_vector : DataFrame
            Alpha vector

        Returns
        -------
        objective : CVXPY Objective
            Objective function
        """
        assert(len(alpha_vector.columns) == 1)

        #TODO: Implement function
        # Note that I chose to implement cvx function 'Minimize' as it wa
        # introduced in Lesson 29 Advanced Portfolio Optimization on Regu
        objective = cvx.Minimize((-alpha_vector.values.flatten()*weights)

        return objective

    def __init__(self, lambda_reg=0.5, risk_cap=0.05, factor_max=10.0, fa
        self.lambda_reg = lambda_reg
        self.risk_cap=risk_cap
        self.factor_max=factor_max
        self.factor_min=factor_min
        self.weights_max=weights_max
        self.weights_min=weights_min


project_tests.test_optimal_holdings_regualization_get_obj(OptimalHoldings
```

```
Running Integration Test on Problem.solve:
> constaints = [sum(weights) == 0.0, sum(cvx.abs(weights)) <= 1.0]
> obj = optimal_holdings_regualization._get_obj(weights, alpha_vector)
> prob = cvx.Problem(obj, constaints)
> prob.solve(max_iters=500)
> solution = np.asarray(weights.value).flatten()

Tests Passed
```

View Data

In [50]:
```
optimal_weights_1 = OptimalHoldingsRegualization(lambda_reg=5.0).find(alp

optimal_weights_1.plot.bar(legend=None, title='Portfolio % Holdings by St
x_axis = plt.axes().get_xaxis()
x_axis.set_visible(False)
```

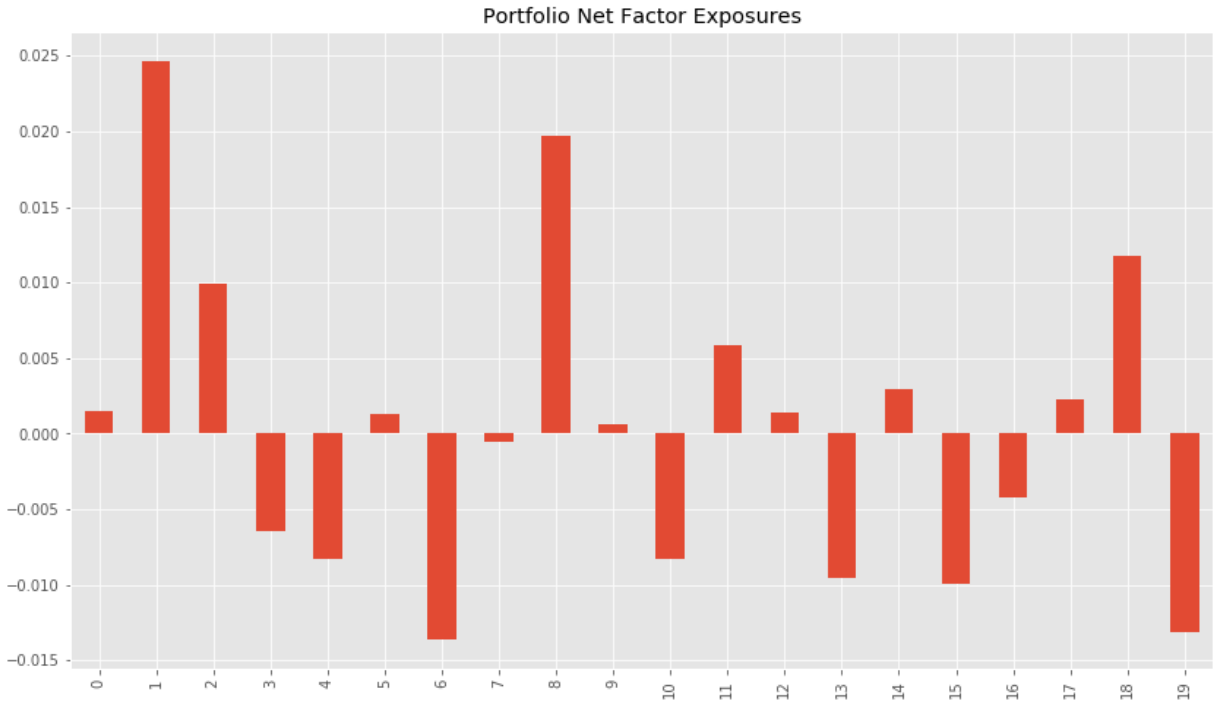
```
/opt/conda/lib/python3.6/site-packages/matplotlib/cbook/deprecation.py:10
6: MatplotlibDeprecationWarning: Adding an axes using the same arguments
as a previous axes currently reuses the earlier instance.  In a future ve
rsion, a new instance will always be created and returned.  Meanwhile, th
is warning can be suppressed, and the future behavior ensured, by passing
a unique label to each axes instance.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```



Nice. Well diversified.

In [51]:
```
project_helper.get_factor_exposures(risk_model['factor_betas'], optimal_w
    title='Portfolio Net Factor Exposures',
    legend=False)
```

Out[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4d05d84588>
```



Optimize with a Strict Factor Constraints and Target Weighting

Another common formulation is to take a predefined target weighting, x^* (e.g., a quantile portfolio), and solve to get as close to that portfolio while respecting portfolio-level constraints. For this next class, `OptimalHoldingsStrictFactor`, you'll implement the `_get_obj` function to minimize on on $ \|x - x^*\|_2 $, where $ x $ is the portfolio weights $ x^* $ is the target weighting.

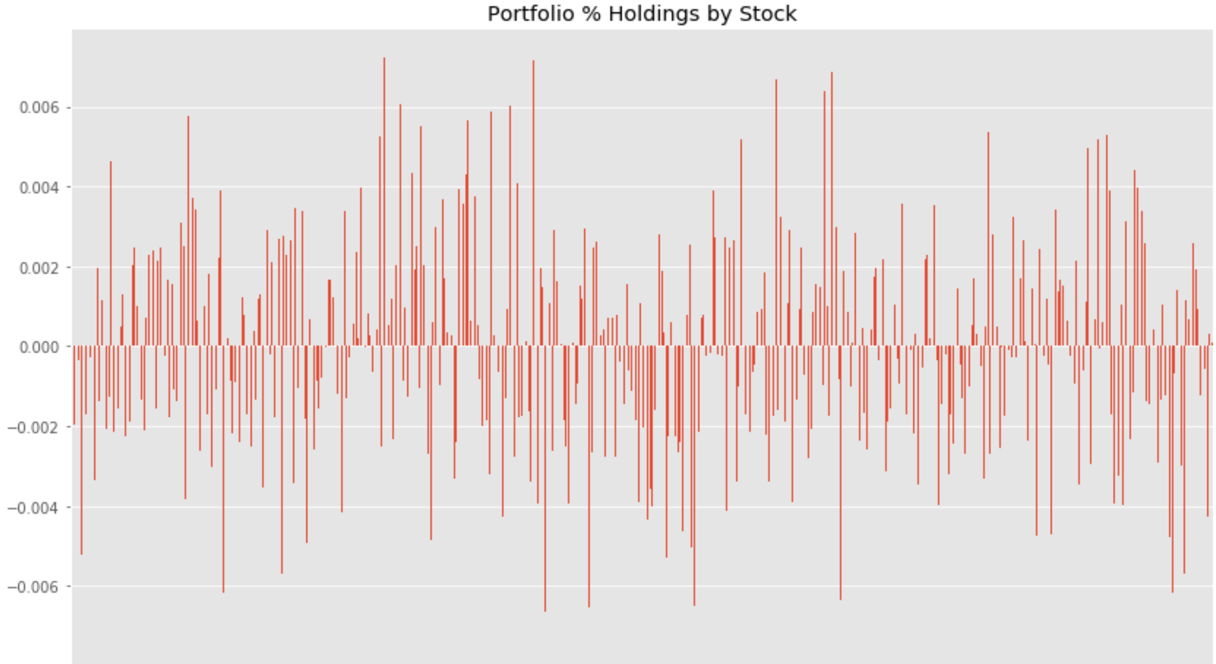
In [80]:
```
class OptimalHoldingsStrictFactor(OptimalHoldings):
    def _get_obj(self, weights, alpha_vector):
        """
        Get the objective function

        Parameters
        ----------
        weights : CVXPY Variable
            Portfolio weights
        alpha_vector : DataFrame
            Alpha vector

        Returns
        -------
        objective : CVXPY Objective
            Objective function
        """
        assert(len(alpha_vector.columns) == 1)

        #TODO: Implement function
        # I used this Udacity Knowledge Article https://knowledge.udacity
        # and the related text in Lesson 29 on Alternative Ways of Settin
        # to understand how to implement the objective

        # Also, it is not clear to me why this code did output wrong resu
        #target_weights = (alpha_vector-np.mean(alpha_vector)/np.sum(np.a

        # This solution works fine:
        target_weights = (alpha_vector-alpha_vector.mean())/alpha_vector.
        objective = cvx.Minimize(cvx.norm(weights - target_weights.values
        return objective


project_tests.test_optimal_holdings_strict_factor_get_obj(OptimalHoldings
```

```
Running Integration Test on Problem.solve:
> constaints = [sum(weights) == 0.0, sum(cvx.abs(weights)) <= 1.0]
> obj = optimal_holdings_strict_factor._get_obj(weights, alpha_vector)
> prob = cvx.Problem(obj, constaints)
> prob.solve(max_iters=500)
> solution = np.asarray(weights.value).flatten()

Tests Passed
```

View Data

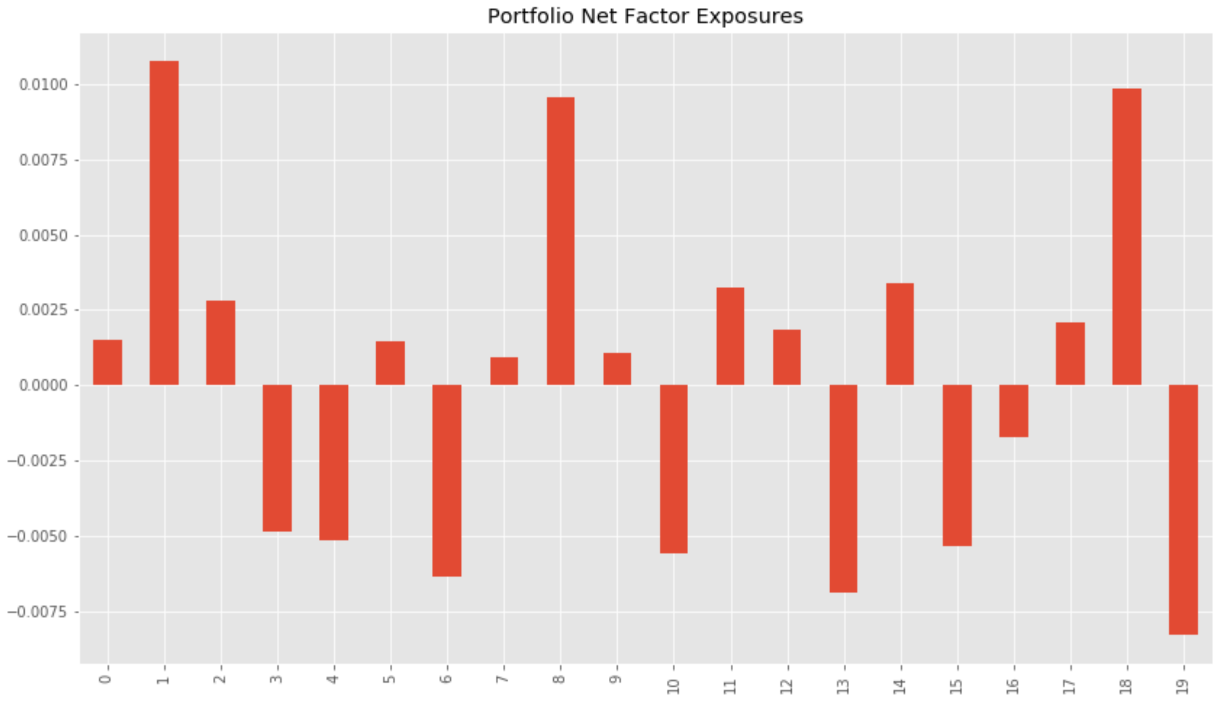
In [81]:
```
optimal_weights_2 = OptimalHoldingsStrictFactor(
    weights_max=0.02,
    weights_min=-0.02,
    risk_cap=0.0015,
    factor_max=0.015,
    factor_min=-0.015).find(alpha_vector, risk_model['factor_betas'], ris

optimal_weights_2.plot.bar(legend=None, title='Portfolio % Holdings by St
x_axis = plt.axes().get_xaxis()
x_axis.set_visible(False)
```

```
/opt/conda/lib/python3.6/site-packages/matplotlib/cbook/deprecation.py:10
6: MatplotlibDeprecationWarning: Adding an axes using the same arguments
as a previous axes currently reuses the earlier instance.  In a future ve
rsion, a new instance will always be created and returned.  Meanwhile, th
is warning can be suppressed, and the future behavior ensured, by passing
a unique label to each axes instance.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```



In [82]:
```
project_helper.get_factor_exposures(risk_model['factor_betas'], optimal_w
    title='Portfolio Net Factor Exposures',
    legend=False)
```

Out[82]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f4d05373f60>`



Submission

Now that you're done with the project, it's time to submit it. Click the submit button in the bottom right. One of our reviewers will give you feedback on your project with a pass or not passed grade. You can continue to the next section while you wait for feedback.