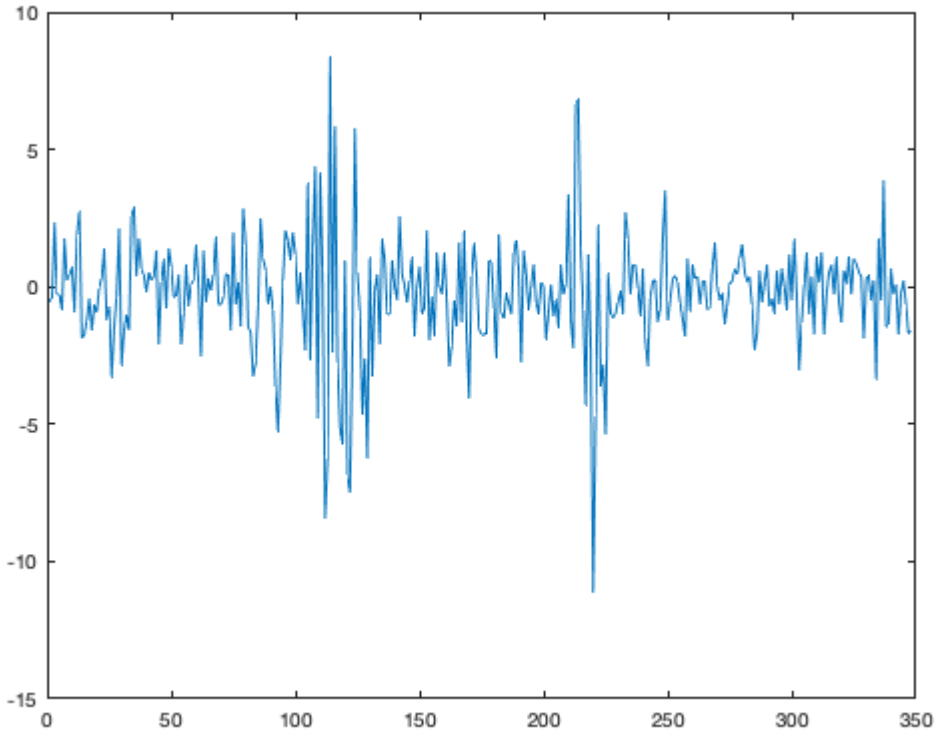
Macroeconomic uncertainty exposure and the factor zoo

The following Matlab live script investigates whether the exposure towards macroeconomic uncertainty contains a marginal contribution relative to existing factors. The data to estimate the factor exposure is the same as used in the portfolio sorting, the test portfolios are the ones from the three-pass estimator, while the confounding factors are from the global factor data website.

```
clear;
clc;

% load data needed to estimate MU exposure

load('ff5.mat')
load('crspDataMonthly2.mat')
load('LIQ.mat')
load('UMD.mat')
load('MU1.mat')
mu_exposure=nan(size(exret));

for i=60:size(exret,1)
   bv=[ones(60,1), MU1(i-59:i,1), ff5(i-59:i,:), UMD(i-59:i,1), LIQ(i-59:i,1)]\exret(i-
   mu_exposure(i,:)=bv(2,:);
end

%Construct the Factor
sizeMu6VW = bivariateSort(ret,meq,mu_exposure,50,[30 70],'Unconditional','VW',meq,'NYSE

% Constructing the momentum factor
muFactor = 0.5.*(sizeMu6VW.returns(:,3) + sizeMu6VW.returns(:,6)) ...
    - 0.5.*(sizeMu6VW.returns(:,1) + sizeMu6VW.returns(:,4));
muFactor=muFactor(61:end,:);

figure;
plot(muFactor);
```




```
% Benchmark alphas
res=nwRegress(muFactor, ff5(61:end,1:3),1, 6);
table(1,:)=[res.bv(1), res.tbv(1)];
res=nwRegress(muFactor, [ff5(61:end,1:3), UMD(61:end,:)],1, 6);
table(2,:)=[res.bv(1), res.tbv(1)];
res=nwRegress(muFactor, ff5(61:end,1:5),1, 6);
table(3,:)=[res.bv(1), res.tbv(1)];

table
```

```
table = 3×2
   -0.2467   -2.1850
   -0.3171   -2.3968
   -0.1447   -1.2462
```

... So the conclusion depends on the benchmark...

Macroeconomic uncertainty exposure relative to the factor zoo

Below, we examine whether exposure towards macroeconomic uncertainty contain a marginal contribution to explaining the cross-section of returns relative to 154 other factors. The factors are available from the website global factor data.

```
load('test_portfolios_giglio.mat');
load('US_factors.mat');

% First we need to align the time-dimension to fit 1991 to 2019:
```

```
Test_assets=Test_assets(end-419:end-12,:);
Test_assets=Test_assets(61:end,:);

vDates=factors(:,1);
factors=[factors(61:end,2:end), ff5(61:end,1)];
vFactors(end+1,:)={'Mkt'};


h_t=factors;
g_t=muFactor;

C_h=(Test_assets-nanmean(Test_assets,1))'*(h_t-nanmean(h_t,1))./size(factors,1);
C_h(isnan(C_h))=0;


beta = NaN(size(C_h));
for i = 1:size(h_t,2)
  beta(:,i) = C_h(:,i)/nanvar(h_t(i,:));
end
penalty = mean(beta.^2,1);
penalty = penalty./mean(penalty); % normalize the level

% Set the K-fold cross validation
k_fold=5;
seed_num=1;

C_g=(Test_assets-nanmean(Test_assets,1))'*(g_t-nanmean(g_t,1))./size(factors,1);
for i=1:seed_num
    rng(i);
    % first step
    [~,FitInfo] = lasso(C_h*diag(penalty),nanmean(Test_assets,1)','CV',k_fold, 'Alpha'
    lambda1(i)=FitInfo.Lambda1SE;

    % second step
    [~,FitInfo] = lasso(C_h*diag(penalty),C_g,'CV',k_fold, 'Alpha', 1, 'Standardize',fa
    lambda2(i)=FitInfo.Lambda1SE;

end

Lambda_opt1=mean(lambda1);
[B,FitInfo] = lasso(C_h*diag(penalty),nanmean(Test_assets,1)','Lambda', Lambda_opt1, '
coef= B;
vFactors(abs(coef)>0,:)
```

ans = 6×1 table

	name
1	'ami_126d'
2	'ebit_bev'
3	'ivol_ff3_21d'
4	'prc_highp...'
5	'rvol_21d'

	name
6	'Mkt'

```
Lambda_opt2=mean(lambda2);
[B,FitInfo] = lasso(C_h*diag(penalty),C_g,'Lambda', Lambda_opt2, 'Alpha', 1, 'Standard
coef2= B;
vFactors(abs(coef2)>0,:)
```

ans = 3×1 table

	name
1	'ami_126d'
2	'ivol_ff3_21d'
3	'Mkt'

```
% third step
I_selection=find(abs(coef)+abs(coef2)>0);
res=nwRegress(nanmean(Test_assets,1)', [C_h(:, I_selection), C_g], 1, 6);
```

Inference

We are now ready to test for whether exposure to macroeconomic uncertainty can explain the variation in expected returns. First we need to estimate the long-run covariance matrix. In the code below, we do not consider the HAC type as written in the paper. The code below corresponds to the version from the replication file of Feng et. al (2020).

```
lambda_all=res.bv(2:end);
v_t=[h_t(:, I_selection), g_t];

% Note, that matlab returns a warning when estimating Lasso without
% intercept. Furthermore, we have to standardize ourselves.
warning('off');
for i=1:seed_num
    rng(i);
    [~,FitInfo] = lasso((h_t-mean(h_t))./std(h_t),(g_t-mean(g_t))./std(g_t),'CV',k_fol
    lambda3(i)=FitInfo.Lambda1SE;
end
Lambda_opt3=mean(lambda3);

[B,FitInfo] = lasso((h_t-mean(h_t))./std(h_t),(g_t-mean(g_t))./std(g_t),'Lambda',Lambda
coef3= B;
I_tilde=find(abs(coef3)>0);
warning('on');

zt_hat=g_t-h_t(:, I_tilde)*inv(h_t(:, I_tilde)'*h_t(:, I_tilde))*h_t(:, I_tilde)'*g_t;
Sigma_z=mean(zt_hat'*zt_hat);

% Estimating the covariance matrix. Note that this is not the HAC version
PI=0;
for t=1:size(g_t,1)
```

```
    PI=PI+(1-v_t(t,:)*lambda_all).^2\Sigma_z*zt_hat(t,:)'*zt_hat(t,:)/(Sigma_z);
end
PI=PI/size(g_t,1);
% And we are left with the t-stat

t_stat=res.bv(end)./PI(end,end)
```

```
t_stat = -4.8629e-04
```

A relatively low t-stat....