Mähri Annagurban & Helene Seland
BI Norwegian Business School

# Active Share

Calculating active share using the formula by Cremers and Petajisto (2009) as Method 1 and Cremers (2017) as Method 2.

## Import fund data

```
xlsFiles = dir('*.xls');
numFiles = length(xlsFiles);
fundName = strings(numFiles,1);
reportDate = strings(numFiles,1);
weights = cell(numFiles,1);

for k = 1:numFiles
    [~,fundName(k)] = xlsread(xlsFiles(k).name,'','D5');
    [~,reportDate(k)] = xlsread(xlsFiles(k).name,'','D7');
    weights{k} = xlsread(xlsFiles(k).name);
end

% Change date format
reportDate = datetime(reportDate,'ConvertFrom','MM/dd/yyyy','Format','yyyy-MM-dd');

% Replace NaN with 0
value = 0;
weights = cellfun(@replace_nan, weights, repmat({value}, size(weights,1),...
    size(weights,2)), 'UniformOutput', 0);

% Check weights add up to 1
for k = 1:numFiles
    weights{k} = weights{k} / 100;
    sumWeights = round(sum(weights{k}));
    if sumWeights == 1
        % do nothing
    else
        error(strcat('Weights do not add up to 1 in:',xlsFiles(k).name))
    end
end
```

## Get fundShortName

```
fundShortName = strings(numFiles,1);

load('foundFunds.mat');

% Find fund name in foundFunds to get corresponding fundShortName
for k = 1:numFiles
    matchName = ismember(foundFunds.Bloomberg_Name,fundName(k));
    fundShortName(k) = foundFunds.fundShortName(matchName);
end
```

## Calculate active share (method 1)

Calculate the absolute weight differences in the holdings and take the sum. Divide the sum by 2 to get the active share for each report date.

```
sumAbsWeightDiff = zeros(numFiles,1);

for k = 1:numFiles
    sumAbsWeightDiff(k) = sum(abs(weights{k}(:,1) - weights{k}(:,2)));
    activeShareMethod1 = (sumAbsWeightDiff / 2);
end
```

## Calculate active share (method 2)

Find the minimum of a stock's weight in the fund and in the benchmark to get the overlapping weight for the stock. Calculate the sum of the overlapping weights between the fund and its benchmark. Subtract the sum from 1 to get the active share for each report date.

```
sumOverlap = zeros(numFiles,1);

for k = 1:numFiles
    sumOverlap(k) = sum(min(weights{k}(:,1), weights{k}(:,2)));
    activeShareMethod2 = (1 - sumOverlap);
end
```

## Save results

```
results = table2timetable(table(reportDate,fundShortName,fundName,...
    activeShareMethod1,activeShareMethod2));
results = struct(fundShortName(1),results);

save('activeShareResults.mat','-struct','results','-append');
```

## Compare activeShareMethod1 & activeShareMethod2

```
compare = round(activeShareMethod1 * 100) == round(activeShareMethod2 * 100);
nnz(~compare)
```

## Functions

Replace NaN with 0

```
function matrix = replace_nan(matrix, value)
  matrix(isnan(matrix)) = value;
end
```

# Tracking Error

## Import fund data

```
xlsFiles = dir('*.xls');
numFiles = length(xlsFiles);
fundName = strings(numFiles,1);
benchmark = strings(numFiles,1);
reportDate = strings(numFiles,1);

for k = 1:numFiles
    [~,fundName(k)] = xlsread(xlsFiles(k).name,'','D5');
    [~,benchmark(k)] = xlsread(xlsFiles(k).name,'','D6');
    [~,reportDate(k)] = xlsread(xlsFiles(k).name,'','D7');
end

% Change date format
reportDate = datetime(reportDate,'ConvertFrom','MM/dd/yyyy','Format','yyyy-MM-dd');
```

## Get fundShortName and ISIN

```
fundShortName = strings(numFiles,1);
ISIN = strings(numFiles,1);

load('foundFunds.mat');

% Find fund name in foundFunds to get corresponding fundShortName and ISIN
for k = 1:numFiles
    matchName = ismember(foundFunds.Bloomberg_Name,fundName(k));
    if nnz(matchName) ~= 0
        fundShortName(k) = foundFunds.fundShortName(matchName);
        ISIN(k) = foundFunds.ISIN(matchName);
    else
        error(strcat('Change Bloomberg_Name to ',fundName(k)));
    end
end
```

## Change fund name in foundFunds

foundFunds.Bloomberg_Name(43) = strrep(foundFunds.Bloomberg_Name(43),... foundFunds.Bloomberg_Name(43),fundName(1)); save('foundFunds.mat','foundFunds','-append');

```
% comment out and run loop above
```

## Assign ticker to benchmark

```
tickers = ["OBX";"OSEFX";"OSEBX";"OSESX"];
idx = zeros(4,1);
benchmarkTicker = strings(numFiles,1);

for k = 1:numFiles
```

```
    for i = 1:4
        idx(i) = contains(benchmark(k),tickers(i));
    end
    benchmarkTicker(k) = tickers(logical(idx));
end
```

## Calculate tracking error

Tracking error is the time series standard deviation of the difference between a fund return and its benchmark return. It is calculated from daily returns in the six months preceding a fund's holdings report date (Cremers and Petajisto, 2009).

```
load('dailyReturns.mat','dailyFundReturns2000to2018',...
    'dailyIndexReturns2000to2018');

% Check if all imported files have same fund name and benchmark
if all(fundShortName == fundShortName(1)) && ...
   all(benchmarkTicker == benchmarkTicker(1))
    % Get returns for benchmark and fund and synchronize based on date
    syncReturns = synchronize(dailyIndexReturns2000to2018.(benchmarkTicker(1)),...
        dailyFundReturns2000to2018.(fundShortName(1)),'intersection');
else
    error('Different fund/benchmark.');
end

trackingError = NaN(numFiles,1);

for k = 1:numFiles
    % For each reportDate, specifiy time range
    endTime = reportDate(k);
    startTime = endTime - calmonths(6); % reportDate minus 6 months
    timeRange = timerange(startTime,endTime,'closed');

    % Get returns within time range
    fundReturns = syncReturns(timeRange,'fundReturns');
    indexReturns = syncReturns(timeRange,'indexReturns');

    % Compute tracking error if there are at least 100 daily return data
    % within time range (to get accurate estimates of tracking error)
    if size(fundReturns,1) >= 100
        trackingError(k) = std(fundReturns.fundReturns - indexReturns.indexReturns);
    end
end
```

## Save results

```
results = table2timetable(table(reportDate,ISIN,fundShortName,fundName,...
    benchmarkTicker,trackingError));
results = struct(fundShortName(1),results);

save('trackingErrorResults.mat','-struct','results','-append');
whos('-file','trackingErrorResults.mat')
```

## Annualized tracking error

```
results = load('trackingErrorResults.mat');
fields = fieldnames(results);

for k = 1:length(fields)
    annualTrackingError = results.(fields{k}).trackingError * sqrt(252);
    results.(fields{k}).annualTrackingError = annualTrackingError;
end

save('trackingErrorResults.mat','-struct','results','-append');
```

# Benchmark-Adjusted Returns of Fund Portfolios

```
results = load('results.mat');
load('foundFunds.mat','nonIndexFunds');
fundShortName = nonIndexFunds.fundShortName;
```

## Select funds

A fund is included in a given month if it has reported its holdings in the previous 12 months.

```
t1 = datetime(2007,1,1);
t2 = datetime(2018,12,31);
t = t1:calmonths(1):t2;

for m = 1:length(t)
    endTime = t(m);
    startTime = endTime - calmonths(12);
    TR = timerange(startTime,endTime);

    funds = cell(1,38);
    for i = 1:length(fundShortName)
        fund = results.(fundShortName(i))(TR,:);
        if isempty(fund)
            % do nothing
        else % get last observation
            funds{i} = fund(end,:);
        end
    end
    % Funds included in a given month
    month{m} = vertcat(funds{:});

    % Remove funds with missing fund size (for fund size-active share sort)
    month{m} = rmmissing(month{m});
end
```

## Sort funds by active share and tracking error

```
for m = 1:length(t)
    funds = month{m};

    % Sort by active share
    [ASbin] = discretize(funds.activeShareMethod1,[0:0.33:1]);
    funds.ASbin = ASbin;

    % Sort further by tracking error
    for i = 1:3
        f = find(funds.ASbin == i);
        [TEbin] = discretize(funds.annualTrackingError(f),3);
        funds.TEbin(f) = TEbin;
    end

    % Get linear index in 2D distribution
    funds.idx = sub2ind([3 3],funds.ASbin,funds.TEbin);
    month{m} = funds;
end
```

## Sort funds by fund size and active share

```
for m = 1:length(t)
    funds = month{m};

    % Sort by fund size
    [FSbin] = discretize(funds.fundSize,3);
    funds.FSbin = FSbin;

    % Sort further by active share
    for i = 1:3
        f = find(funds.FSbin == i);
        [ASbin] = discretize(funds.activeShareMethod1(f),[0:0.33:1]);
        funds.ASbin(f) = ASbin;
    end

    % Get linear index in 2D distribution
    funds.idx = sub2ind([3 3],funds.ASbin,funds.FSbin);

    month{m} = funds;
end
```

## Median fund size

```
for m = 1:length(t)
    portfolio = {};
    for i = 1:3
        f = find(month{m}.FSbin == i);
        portfolio{i} = month{m}(f,:);
    end
    monthlyPortfolio{m} = portfolio;
end
```

```
for i = 1:3
    monthlyPortSize = [];
    for m = 1:length(t)
     monthlyPortSize{m} = monthlyPortfolio{m}{i}.fundSize;
    end
    medianSize(i) = round(median(vertcat(monthlyPortSize{:}))/1000000);
end
```

## Monthly benchmark-adjusted returns for each portfolio

```
benchAdjReturn = load('benchAdjReturns.mat');

for m = 1:length(t)
    % Monthly benchmark-adjusted return for each fund in a given month
    tEnd = dateshift(t(m),'end','month');
    funds = month{m}.fundShortName;
    for k = 1:length(funds)
        if ismember(tEnd,benchAdjReturn.(funds(k)).date)
            month{m}.benchAdjReturns(k) = benchAdjReturn.(funds(k)).benchAdjReturn(tEnd);
        else
            month{m}.benchAdjReturns(k) = NaN;
        end
    end

    % Equal-weighted return within each portfolio
    portfolioReturn = zeros(9,1);
    for i = 1:9
        f = find(month{m}.idx == i);
        returns = month{m}.benchAdjReturns(f);
        portfolioReturn(i) = nanmean(returns);
    end
    monthlyPortReturns(:,m) = portfolioReturn;

    % Equal-weighted return of funds sorted by active share
    ASportfolioReturn = zeros(3,1);
    for i = 1:3
        f = find(month{m}.ASbin == i);
        returns = month{m}.benchAdjReturns(f);
        ASportfolioReturn(i) = nanmean(returns);
    end
    monthlyASPortReturns(:,m) = ASportfolioReturn;

    % Equal-weighted return of funds sorted by tracking error
%     TEportfolioReturn = zeros(3,1);
%     for i = 1:3
%         f = find(month{m}.TEbin == i);
%         returns = month{m}.benchAdjReturns(f);
%         TEportfolioReturn(i) = nanmean(returns);
%     end
%     monthlyTEPortReturns(:,m) = TEportfolioReturn;

    % Equal-weighted return of funds sorted by fund size
    FSportfolioReturn = zeros(3,1);
    for i = 1:3
        f = find(month{m}.FSbin == i);
        returns = month{m}.benchAdjReturns(f);
        FSportfolioReturn(i) = nanmean(returns);
    end
    monthlyFSPortReturns(:,m) = FSportfolioReturn;

    % Equal-weighted return of all funds
    monthlyAllFunds(m) = nanmean(month{m}.benchAdjReturns);
end
```

## Portfolios for regression

```
date = transpose(dateshift(t,'end','month'));
date.Format = 'yyyy-MM-dd';
```

```
portNames = ["LowLow" ; "MedLow" ; "HighLow"
             "LowMed" ; "MedMed" ; "HighMed"
             "LowHigh" ; "MedHigh" ; "HighHigh"];

ASportNames = ["LowAS" ; "MedAS" ; "HighAS"];

TEportNames = ["LowTE" ; "MedTE" ; "HighTE"];

FSportNames = ["LowFS" ; "MedFS" ; "HighFS"];

for i = 1:9
    benchAdjReturn = transpose(monthlyPortReturns(i,:));
    tbl = table(date,benchAdjReturn);
    tbl.idx(:) = i;
    tbl.portName(:) = portNames(i);
    portfolioReturns.(portNames(i)) = table2timetable(tbl);
end

for i = 1:3
    benchAdjReturn = transpose(monthlyASPortReturns(i,:));
    tbl = table(date,benchAdjReturn);
    tbl.idx(:) = i;
    tbl.portName(:) = ASportNames(i);
    portfolioReturns.(ASportNames(i)) = table2timetable(tbl);
end

% for i = 1:3
%     benchAdjReturn = transpose(monthlyTEPortReturns(i,:));
%     tbl = table(date,benchAdjReturn);
%     tbl.idx(:) = i;
%     tbl.portName(:) = TEportNames(i);
%     portfolioReturns.(TEportNames(i)) = table2timetable(tbl);
% end

for i = 1:3
    benchAdjReturn = transpose(monthlyFSPortReturns(i,:));
    tbl = table(date,benchAdjReturn);
    tbl.idx(:) = i;
    tbl.portName(:) = FSportNames(i);
    portfolioReturns.(FSportNames(i)) = table2timetable(tbl);
end

benchAdjReturn = transpose(monthlyAllFunds);
tbl = table(date,benchAdjReturn);
portfolioReturns.All = table2timetable(tbl);

% save('benchAdjReturns.mat','-struct','portfolioReturns','-append');
save('benchAdjReturnsFS.mat','-struct','portfolioReturns');
```

## Average benchmark-adjusted return for each portfolio over sample period

```
T = length(month);
```

```matlab
% Average return for each portfolio
for i = 1:9
    meanPortReturn(i,1) = nanmean(monthlyPortReturns(i,:));
    stdPortReturn(i,1) = nanstd(monthlyPortReturns(i,:));
    tStatPortReturn(i,1) = meanPortReturn(i)/(stdPortReturn(i)/sqrt(T));
end

% Average return for portfolios sorted by active share
for i = 1:3
    meanASPortReturn(i,1) = nanmean(monthlyASPortReturns(i,:));
    stdASPortReturn(i,1) = nanstd(monthlyASPortReturns(i,:));
    tStatASPortReturn(i,1) = meanASPortReturn(i)/(stdASPortReturn(i)/sqrt(T));
end

% Average return for portfolios sorted by tracking error
for i = 1:3
    meanTEPortReturn(i,1) = nanmean(monthlyTEPortReturns(i,:));
    stdTEPortReturn(i,1) = nanstd(monthlyTEPortReturns(i,:));
    tStatTEPortReturn(i,1) = meanTEPortReturn(i)/(stdTEPortReturn(i)/sqrt(T));
end

% Average return for portfolio of all funds
meanAllFunds = mean(monthlyAllFunds);
stdAllFunds = std(monthlyAllFunds);
tStatAllFunds = meanAllFunds/(stdAllFunds/sqrt(T));

% HML
meanHMLcol(1,1) = meanPortReturn(7) - meanPortReturn(1);
meanHMLcol(2,1) = meanPortReturn(8) - meanPortReturn(2);
meanHMLcol(3,1) = meanPortReturn(9) - meanPortReturn(3);
meanHMLcol(4,1) = meanTEPortReturn(3) - meanTEPortReturn(1);

stdHMLcol(1,1) = stdPortReturn(7) - stdPortReturn(1);
stdHMLcol(2,1) = stdPortReturn(8) - stdPortReturn(2);
stdHMLcol(3,1) = stdPortReturn(9) - stdPortReturn(3);
stdHMLcol(4,1) = stdTEPortReturn(3) - stdTEPortReturn(1);

tStatHMLcol(1,1) = meanHMLcol(1)/(stdHMLcol(1)/sqrt(T));
tStatHMLcol(2,1) = meanHMLcol(2)/(stdHMLcol(2)/sqrt(T));
tStatHMLcol(3,1) = meanHMLcol(3)/(stdHMLcol(3)/sqrt(T));
tStatHMLcol(4,1) = meanHMLcol(4)/(stdHMLcol(4)/sqrt(T));

meanHMLrow(1,1) = meanPortReturn(3) - meanPortReturn(1);
meanHMLrow(2,1) = meanPortReturn(6) - meanPortReturn(4);
meanHMLrow(3,1) = meanPortReturn(9) - meanPortReturn(7);
meanHMLrow(4,1) = meanASPortReturn(3) - meanASPortReturn(1);

stdHMLrow(1,1) = stdPortReturn(3) - stdPortReturn(1);
stdHMLrow(2,1) = stdPortReturn(6) - stdPortReturn(4);
stdHMLrow(3,1) = stdPortReturn(9) - stdPortReturn(7);
stdHMLrow(4,1) = stdASPortReturn(3) - stdASPortReturn(1);

tStatHMLrow(1,1) = meanHMLrow(1)/(stdHMLrow(1)/sqrt(T));
tStatHMLrow(2,1) = meanHMLrow(2)/(stdHMLrow(2)/sqrt(T));
tStatHMLrow(3,1) = meanHMLrow(3)/(stdHMLrow(3)/sqrt(T));
tStatHMLrow(4,1) = meanHMLrow(4)/(stdHMLrow(4)/sqrt(T));
```

## Returns table

```
R = zeros(3,3);

for i = 1:9
    R(i) = meanPortReturn(i);
end

R(1:3,4) = meanASPortReturn;
R(4,1:3) = meanTEPortReturn;
R(4,4) = meanAllFunds;
R(1:4,5) = meanHMLcol;
R(5,1:4) = meanHMLrow;

R = R * 100;
```

## Tstat table

```
tstats = zeros(3,3);

for i = 1:9
    tstats(i) = tStatPortReturn(i);
end

tstats(1:3,4) = tStatASPortReturn;
tstats(4,1:3) = tStatTEPortReturn;
tstats(4,4) = tStatAllFunds;
tstats(1:4,5) = tStatHMLcol;
tstats(5,1:4) = tStatHMLrow;
```

## Export

```
X = [R(1,:);
     tstats(1,:);
     R(2,:);
     tstats(2,:);
     R(3,:);
     tstats(3,:);
     R(4,:);
     tstats(4,:);
     R(5,:);
     tstats(5,:)];

%writematrix(X,'FundPortfolios.xls');
```

## Benchmark-adjusted returns for individual funds (index funds included)

```
load('foundFunds.mat');
load('monthlyReturns.mat');
funds = load('results.mat');
fundShortName = foundFunds.fundShortName;
benchmarkTicker = foundFunds.benchmarkTicker;
fundName = foundFunds.Bloomberg_Name;
```

### Get monthly returns & calculate benchmark-adjusted returns

```
for i = 1:length(fundShortName)
    startTime = funds.(fundShortName(i)).reportDate(1);
    endTime = funds.(fundShortName(i)).reportDate(end);
    timeRange = timerange(startTime,endTime,'closed');

    % Get returns within time range
    fundReturns = monthlyFundReturns.(fundShortName(i))(timeRange,:);
    indexReturns = monthlyIndexReturns2000to2018.(benchmarkTicker(i))(timeRange,:);
    syncReturns = synchronize(fundReturns,indexReturns,'intersection');

    % Calculate benchmark-adjusted returns (R_fund minus R_index)
    benchAdjReturn = syncReturns.MonthlyReturn - syncReturns.returns;

    T = table(syncReturns.Date,'VariableNames',{'date'});
    T.fundShortName(:) = fundShortName(i);
    T.fundName(:) = fundName(i);
    T.benchmarkTicker(:) = benchmarkTicker(i);
    T.benchAdjReturn = benchAdjReturn;
    results = table2timetable(T);
    results = struct(fundShortName(i),results);
    save('benchAdjReturns.mat','-struct','results','-append');
end
```

# Four-Factor Alpha of Benchmark-Adjusted Return

```
benchAdjReturns = load('benchAdjReturns.mat');
% benchAdjReturns = load('benchAdjReturnsFS.mat');
load('4Factors.mat','monthlyPricingFactors');
factors = monthlyPricingFactors(:,[5 1 2 3]); % RMRF, SMB, HML, PR1YR
```

## Individual funds

```
load('foundFunds.mat','nonIndexFunds');
fundShortName = nonIndexFunds.fundShortName;
fundName = nonIndexFunds.OBI_Name;
benchmark = nonIndexFunds.benchmarkTicker;
expenseRatio = nonIndexFunds.expenseRatio;
```

## Fund portfolios

```
fundShortName = ["LowLow", "MedLow", "HighLow",...
                 "LowMed", "MedMed", "HighMed",...
                 "LowHigh", "MedHigh", "HighHigh",...
                 "LowAS", "MedAS", "HighAS",...
                 "LowTE", "MedTE", "HighTE",...
                 "All"];

% fundShortName = ["LowLow", "MedLow", "HighLow",...
%                  "LowMed", "MedMed", "HighMed",...
%                  "LowHigh", "MedHigh", "HighHigh",...
%                  "LowAS", "MedAS", "HighAS",...
%                  "LowFS", "MedFS", "HighFS",...
%                  "All"];

idx = [1:9 1:3 1:4];
```

## Estimate linear regression model

```
for i = 1:length(fundShortName)
    benchAdjReturn = benchAdjReturns.(fundShortName(i))(:,'benchAdjReturn');
    benchAdjReturn = rmmissing(benchAdjReturn);
    startTime = benchAdjReturn.date(1);
    endTime = benchAdjReturn.date(end);
    timeRange = timerange(startTime,endTime,'closed');
    myFactors = factors(timeRange,:);
    regData{i} = timetable2table(synchronize(myFactors,benchAdjReturn,'intersection'));

    mdl = fitlm(regData{i}, 'benchAdjReturn ~ RMRF + SMB + HML + PR1YR');
    mdlCoeficients.(fundShortName(i)) = mdl.Coefficients;
    estAlpha = mdl.Coefficients{1,1};
    SEalpha = mdl.Coefficients{1,2};
    testStat = mdl.Coefficients{1,3};
    pVal = mdl.Coefficients{1,4};
    T = mdl.NumObservations;
    k = mdl.NumEstimatedCoefficients;
    Rsqrd = mdl.Rsquared.Ordinary;
    adjRsqrd = mdl.Rsquared.Adjusted;
    residuals{i} = mdl.Residuals.Raw;

    meanBenchAdjRet = mean(benchAdjReturn.benchAdjReturn);
    stdBenchAdjRet = std(benchAdjReturn.benchAdjReturn);
    tStatBenchAdjRet = meanBenchAdjRet/(stdBenchAdjRet/sqrt(T));

    % Individual funds
%     tbl = table(fundShortName(i),fundName(i),benchmark(i),expenseRatio(i),...
%         startTime,endTime,meanBenchAdjRet,tStatBenchAdjRet,stdBenchAdjRet,...
%         T,k,estAlpha,SEalpha,testStat,pVal,Rsqrd,adjRsqrd);

    % Fund portfolios
    tbl = table(fundShortName(i),idx(i),...
        startTime,endTime,meanBenchAdjRet,tStatBenchAdjRet,stdBenchAdjRet,...
        T,k,estAlpha,SEalpha,testStat,pVal,Rsqrd,adjRsqrd);

    output{i} = tbl;
end

tbl = vertcat(output{:});
tbl.Properties.VariableNames(1) = {'fundShortName'};
tbl.Properties.VariableNames(2) = {'idx'};
% tbl.Properties.VariableNames(2) = {'fundName'};
% tbl.Properties.VariableNames(3) = {'benchmark'};
% tbl.Properties.VariableNames(4) = {'expenseRatio'};
```

## Statistical significance of alpha

H0: estAlpha = 0

```
alpha = 0.05;
for i = 1:height(tbl)
    T = tbl.T(i);
    k = tbl.k(i);
    testStat = tbl.testStat(i);
    tCrit = tinv(alpha/2,T-k);
    reject(i) = abs(testStat) > abs(tCrit);
    % 1: condition is true, i.e., reject H0
    % 0: condition is not true, i.e., do not reject H0
end

tbl.alphaSignificant = transpose(reject);
% 1: Alpha is statistically significant
% 0: Alpha is not statistically significant
```

## CLRM assumption 1: Mean of residuals is zero

H0: meanResiduals = 0

```
for i = 1:height(tbl)
    T = tbl.T(i);
    u = residuals{i};
    meanResiduals = mean(u);
    stdResiduals = std(u);
    testStat = meanResiduals/(stdResiduals/sqrt(T));
    tCrit = tinv(alpha/2,T-1);
    reject(i) = abs(testStat) > abs(tCrit);
end

tbl.A1violated = transpose(reject);
% 1: Assumption violated
% 0: Assumption not violated
```

## CLRM assumption 2: Variance of residuals is constant

White's test for heteroscedasticity H0: Auxiliary coefficients (excluding constant) are jointly zero

```
for i = 1:height(tbl)
regData2 = table(regData{i}.date,...
    regData{i}.RMRF, regData{i}.SMB, regData{i}.HML, regData{i}.PR1YR,...
    regData{i}.RMRF.^2, regData{i}.SMB.^2,regData{i}.HML.^2,...
    regData{i}.PR1YR.^2, residuals{i}.^2,...
    'VariableNames',...
    {'Date', 'RMRF', 'SMB', 'HML', 'PR1YR',...
    'RMRFsq', 'SMBsq', 'HMLsq', 'PR1YRsq', 'ResidualsSq'});

auxRegr = fitlm(regData2,...
    'ResidualsSq ~ RMRF + SMB + HML + PR1YR + RMRFsq + SMBsq + HMLsq + PR1YRsq');

% 8 out of 9 coefficients tested
c = zeros(9,1);
H = [0 0 0 0 0 0 0 0 0;
     0 1 0 0 0 0 0 0 0;
     0 0 1 0 0 0 0 0 0;
     0 0 0 1 0 0 0 0 0;
     0 0 0 0 1 0 0 0 0;
     0 0 0 0 0 1 0 0 0;
     0 0 0 0 0 0 1 0 0;
     0 0 0 0 0 0 0 1 0;
     0 0 0 0 0 0 0 0 1];
[p(i),F] = coefTest(auxRegr,H,c);
reject(i) = alpha > p(i);
end

tbl.A2violated = transpose(reject);
tbl.WhitesTestP = transpose(p);
```

## CLRM assumption 3: Covariance of residuals over time is zero

Breusch-Godfrey test with r = 5 lags H0: Autocorrelation = 0

```
for i = 1:height(tbl)
    regData{i}.u = residuals{i};
    regData{i}.u1 = lagmatrix(residuals{i},1);
    regData{i}.u2 = lagmatrix(residuals{i},2);
    regData{i}.u3 = lagmatrix(residuals{i},3);
    regData{i}.u4 = lagmatrix(residuals{i},4);
    regData{i}.u5 = lagmatrix(residuals{i},5);

    regData{i} = fillmissing(regData{i},'constant',0,'DataVariables',...
        {'u','u1','u2','u3','u4','u5'});

    auxRegr = fitlm(regData{i}, ['u ~ RMRF + SMB + HML + PR1YR' ...
    '+ u1 + u2 + u3 + u4 + u5']);

    T = tbl.T(i);
    r = 5;
    testStat = auxRegr.Rsquared.Ordinary * (T - r);
    critVal = chi2inv(1 - alpha,r);
    p(i) = 1 - cdf('Chisquare',testStat,r);
    reject(i) = testStat > critVal;
    % 1: condition is true, i.e., reject H0
    % 0: condition is not true, i.e., do not reject H0
end

tbl.A3violated = transpose(reject);
tbl.BGtestP = transpose(p);
```

## CLRM assumption 4: No relation between residuals and independent variables

H0: Corr(u_t,x_t) = 0

```
for i = 1:height(tbl)
    u = residuals{i};
    uCorrRMRF = corrcoef(u,regData{i}.RMRF);
    uCorrSMB = corrcoef(u,regData{i}.SMB);
    uCorrHML = corrcoef(u,regData{i}.HML);
    uCorrPR1YR = corrcoef(u,regData{i}.PR1YR);
    correlations = round([uCorrRMRF(1,2) uCorrSMB(1,2) uCorrHML(1,2) uCorrPR1YR(1,2)],5);
    reject(i) = any(correlations);
end

tbl.A4violated = transpose(reject);
```

## CLRM assumption 5: Residuals are normally distributed

Jarque-Bera test H0: Coefficients of skewness and excess kurtosis are jointly zero

```
for i = 1:height(tbl)
    u = residuals{i};
    [h,p(i),jbstat,critval] = jbtest(u);
    reject(i) = h;
end

tbl.A5violated = transpose(reject);
tbl.JBtestP = transpose(p);
```

```matlab
% Anderson-Darling test
% H0: Residuals are normally distributed
for i = 1:height(tbl)
    u = residuals{i};
    [h,p(i),adstat,critval] = adtest(u);
    reject(i) = h;
end

tbl.A5violated2 = transpose(reject);
tbl.ADtestP = transpose(p);
```

## Testing for multicollinearity

```matlab
for i = 1:height(tbl)
    x = [regData{i}.RMRF, regData{i}.SMB, regData{i}.HML, regData{i}.PR1YR];
    corrMatrix = corrcoef(x);
    rowNames = {'RMRF','SMB','HML','PR1YR'};
    varNames = {'RMRF','SMB','HML','PR1YR'};
    corrTbl{i} = array2table(corrMatrix,'RowNames',rowNames,'VariableNames',varNames);
end
```

## Estimate regression model with Newey-West's heteroscedasticity and autocorrelation consistent standard errors

Assumption: Estimated residuals are heteroscedastic and autocorrelated

```matlab
for i = 1:height(tbl)
    x = [regData{i}.RMRF, regData{i}.SMB, regData{i}.HML, regData{i}.PR1YR];
    y = regData{i}.benchAdjReturn;

    newey_west = table();
    T = tbl.T(i);
    const = 1;
    lags = 10;
    [beta,tstat,~,vcvm,R2,RBAR,~] = olsnw(y,x,const,lags);
    newey_west.beta = beta;
    newey_west.SE = sqrt(diag(vcvm));
    newey_west.tstat = tstat;
    newey_west_R2(i) = R2;
    newey_west_Rbar(i) = RBAR;
    newey_west.p = 2*(1-tcdf(abs(tstat),T-2));
    %newey_west.Properties.RowNames = {'(Intercept)', 'RMRF', 'SMB', 'HML', 'PR1YR'};

    output{i} = newey_west(1,:);
end

newey_west = vertcat(output{:});
newtbl = tbl;
newtbl.estAlpha = newey_west.beta;
newtbl.testStat = newey_west.tstat;
newtbl.Rsqrd = transpose(newey_west_R2);
newtbl.adjRsqrd = transpose(newey_west_Rbar);
```

```
% newtbl = tbl(:,[1:10 12 14 16:18]);
% newtbl.startTime.Format = 'yyyy-MM';
% newtbl.endTime.Format = 'yyyy-MM';

% save('portRegressionResults.mat','newtbl','corrTbl','-append');
save('FSportRegressionResults.mat','newtbl','corrTbl');
```

## HML alpha

```
HMLcol(1,1) = newtbl.estAlpha(7) - newtbl.estAlpha(1);
HMLcol(2,1) = newtbl.estAlpha(8) - newtbl.estAlpha(2);
HMLcol(3,1) = newtbl.estAlpha(9) - newtbl.estAlpha(3);
HMLcol(4,1) = newtbl.estAlpha(15) - newtbl.estAlpha(13);

seHMLcol(1,1) = newtbl.SEalpha(7) - newtbl.SEalpha(1);
seHMLcol(2,1) = newtbl.SEalpha(8) - newtbl.SEalpha(2);
seHMLcol(3,1) = newtbl.SEalpha(9) - newtbl.SEalpha(3);
seHMLcol(4,1) = newtbl.SEalpha(15) - newtbl.SEalpha(13);

tStatHMLcol(1,1) = HMLcol(1) / seHMLcol(1);
tStatHMLcol(2,1) = HMLcol(2) / seHMLcol(2);
tStatHMLcol(3,1) = HMLcol(3) / seHMLcol(3);
tStatHMLcol(4,1) = HMLcol(4) / seHMLcol(4);

HMLrow(1,1) = newtbl.estAlpha(3) - newtbl.estAlpha(1);
HMLrow(2,1) = newtbl.estAlpha(6) - newtbl.estAlpha(4);
HMLrow(3,1) = newtbl.estAlpha(9) - newtbl.estAlpha(7);
HMLrow(4,1) = newtbl.estAlpha(12) - newtbl.estAlpha(10);

seHMLrow(1,1) = newtbl.SEalpha(3) - newtbl.SEalpha(1);
seHMLrow(2,1) = newtbl.SEalpha(6) - newtbl.SEalpha(4);
seHMLrow(3,1) = newtbl.SEalpha(9) - newtbl.SEalpha(7);
seHMLrow(4,1) = newtbl.SEalpha(12) - newtbl.SEalpha(10);

tStatHMLrow(1,1) = HMLrow(1) / seHMLrow(1);
tStatHMLrow(2,1) = HMLrow(2) / seHMLrow(2);
tStatHMLrow(3,1) = HMLrow(3) / seHMLrow(3);
tStatHMLrow(4,1) = HMLrow(4) / seHMLrow(4);
```

## Output

```
S = newtbl(1:9,:);
R = zeros(3,3);
tstats = zeros(3,3);

% Alpha
for i = 1:9
    R(i) = S.estAlpha(i);
end

R(1:3,4) = newtbl.estAlpha(10:12);
R(4,1:3) = newtbl.estAlpha(13:15);
R(4,4) = newtbl.estAlpha(16);
```

```
R(1:4,5) = HMLcol;
R(5,1:4) = HMLrow;

R = R * 100;

% Alpha tstats
for i = 1:9
    tstats(i) = S.testStat(i);
end

tstats(1:3,4) = newtbl.testStat(10:12);
tstats(4,1:3) = newtbl.testStat(13:15);
tstats(4,4) = newtbl.testStat(16);
tstats(1:4,5) = tStatHMLcol;
tstats(5,1:4) = tStatHMLrow;

X = [R(1,:);
     tstats(1,:);
     R(2,:);
     tstats(2,:);
     R(3,:);
     tstats(3,:);
     R(4,:);
     tstats(4,:);
     R(5,:);
     tstats(5,:)];

writematrix(X,'FundPortfolios.xls','Sheet',2);
```

## Calculate average active share and tracking error (for individual funds)

```
results = load('results.mat');

for i = 1:length(fundShortName)
    meanActiveShare(i) = mean(results.(fundShortName(i)).activeShareMethod1);
    meanTrackingError(i) = mean(results.(fundShortName(i)).annualTrackingError);
end

newtbl.meanActiveShare = transpose(meanActiveShare);
newtbl.meanTrackingError = transpose(meanTrackingError);

newtbl.meanBenchAdjRet = newtbl.meanBenchAdjRet * 100;
newtbl.stdBenchAdjRet = newtbl.stdBenchAdjRet * 100;
newtbl.estAlpha = newtbl.estAlpha * 100;
newtbl.Rsqrd = newtbl.Rsqrd * 100;
newtbl.adjRsqrd = newtbl.adjRsqrd * 100;
newtbl.meanActiveShare = newtbl.meanActiveShare * 100;
newtbl.meanTrackingError = newtbl.meanTrackingError * 100;
newtbl = sortrows(newtbl,7);
```

save('regressionResults.mat','tbl','corrTbl','newtbl','-append'); export = newtbl(:,[2:9 11:12 16:17]); writetable(export,'FundRegResults.xls','Sheet',1);

# Results for 2018

```
results = load('results.mat');
fundSize = load('fundSize.mat');
f1 = string(fieldnames(results));
f2 = string(fieldnames(fundSize));
load('foundFunds.mat');
```

## Two-dimensional distribution of funds

```
startTime = '2018-01-01';
endTime = '2018-12-31';
TR = timerange(startTime,endTime,'closed');

Xedges = 0:10:100; % active share
Yedges = 0:2:100; % tracking error
```

## Number of mutual funds

```
N = zeros(size(Xedges,2)-1,size(Yedges,2)-1);
TN = zeros(size(Xedges,2)-1,size(Yedges,2)-1);
AC = zeros(1,size(Xedges,2)-1);
AR = zeros(1,size(Yedges,2)-1);

for i = 1:length(f1)
    % Extract 2018 results
    fund = results.(f1(i))(TR,:);
    % Convert to percentages
    fund.activeShareMethod1 = fund.activeShareMethod1 * 100;
    fund.annualTrackingError = fund.annualTrackingError * 100;
    % Sort into bins
    [N,~,~,binX,binY] = histcounts2(fund.activeShareMethod1,fund.annualTrackingError,...
        Xedges,Yedges);
    [C,~,row] = histcounts(fund.activeShareMethod1,Xedges);
    [R,~,column] = histcounts(fund.annualTrackingError,Yedges);
    % Add to previous count
    TN = TN + N;
    AC = AC + C; % "All" column
    AR = AR + R; % "All" row
    % Get linear index, index in last row, index in last column
    fund.idx = sub2ind(size(N),binX,binY);
    fund.row = row;
    fund.column = column;
    % Save
    output{i} = fund;
end

tbl = vertcat(output{:});
a = unique(tbl.fundShortName); % 39 funds

TN = [TN transpose(AC)]; % insert "All" column
TN = [TN ; AR sum(AR)]; % insert "All" row
```

```
% writematrix(TN,'2018Results.xls','Sheet',1);
```

## Equal-weighted expense ratio

```
TN = zeros(size(Xedges,2)-1,size(Yedges,2)-1);
AC = zeros(1,size(Xedges,2)-1);
AR = zeros(1,size(Yedges,2)-1);
All = zeros(1,1);
uniqueIdx = unique(tbl.idx);
uniqueRow = unique(tbl.row);
uniqueCol = unique(tbl.column);
uniqueAll = unique(tbl.fundShortName);

for i = 1:length(uniqueIdx)
    f = find(tbl.idx == uniqueIdx(i));
    funds = unique(tbl.fundShortName(f));
    expenseRatio = [];
    for k = 1:length(funds)
        m = ismember(foundFunds.fundShortName,funds(k));
        expenseRatio(k) = foundFunds.expenseRatio(m);
    end
    averageExpenseRatio = round(mean(expenseRatio),2);
    TN(uniqueIdx(i)) = averageExpenseRatio;
end

% Last column
for i = 1:length(uniqueRow)
    f = find(tbl.row == uniqueRow(i));
    funds = unique(tbl.fundShortName(f));
    expenseRatio = [];
    for k = 1:length(funds)
        m = ismember(foundFunds.fundShortName,funds(k));
        expenseRatio(k) = foundFunds.expenseRatio(m);
    end
    averageExpenseRatio = round(mean(expenseRatio),2);
    AC(uniqueRow(i)) = averageExpenseRatio;
end

% Last row
for i = 1:length(uniqueCol)
    f = find(tbl.column == uniqueCol(i));
    funds = unique(tbl.fundShortName(f));
    expenseRatio = [];
    for k = 1:length(funds)
        m = ismember(foundFunds.fundShortName,funds(k));
        expenseRatio(k) = foundFunds.expenseRatio(m);
    end
    averageExpenseRatio = round(mean(expenseRatio),2);
    AR(uniqueCol(i)) = averageExpenseRatio;
end

% "All" cell
```

```
expenseRatio = [];
for i = 1:length(uniqueAll)
    m = ismember(foundFunds.fundShortName,uniqueAll(i));
    expenseRatio(i) = foundFunds.expenseRatio(m);
end
averageExpenseRatio = round(mean(expenseRatio),2);
All = averageExpenseRatio;

TN = [TN transpose(AC)];
TN = [TN ; zeros(1,51)];
TN(end,:) = [AR All];

% writematrix(TN,'2018Results.xls','Sheet',2);
```

## Median market value (NOKm)

```
TN = zeros(size(Xedges,2)-1,size(Yedges,2)-1);
AC = zeros(1,size(Xedges,2)-1);
AR = zeros(1,size(Yedges,2)-1);
All = zeros(1,1);

for i = 1:length(f2)
    % Extract 2018 results
    fund = results.(f2(i))(TR,:);
    % Convert to percentages
    fund.activeShareMethod1 = fund.activeShareMethod1 * 100;
    fund.annualTrackingError = fund.annualTrackingError * 100;
    % Sort into bins
    [N,~,~,binX,binY] = histcounts2(fund.activeShareMethod1,fund.annualTrackingError,...
        Xedges,Yedges);
    [C,~,row] = histcounts(fund.activeShareMethod1,Xedges);
    [R,~,column] = histcounts(fund.annualTrackingError,Yedges);
    % Add to previous count
    TN = TN + N;
    AC = AC + C; % "All" column
    AR = AR + R; % "All" row
    % Get linear index, index in last row, index in last column
    fund.idx = sub2ind(size(N),binX,binY);
    fund.row = row;
    fund.column = column;
    % Save
    output{i} = fund;
end

tbl = vertcat(output{:});
a = unique(tbl.fundShortName); % 35 funds
```

## Calculate median market value

```
uniqueIdx = unique(tbl.idx);
uniqueRow = unique(tbl.row);
uniqueCol = unique(tbl.column);
uniqueAll = unique(tbl.fundShortName);
```

```
for i = 1:length(uniqueIdx)
    f = find(tbl.idx == uniqueIdx(i));
    values = tbl.fundSize(f);
    medianFundSize = median(values);
    medianFundSize = round(medianFundSize / 1000000);
    TN(uniqueIdx(i)) = medianFundSize;
end
```

## Last column

```
for i = 1:length(uniqueRow)
    f = find(tbl.row == uniqueRow(i));
    values = tbl.fundSize(f);
    medianFundSize = median(values);
    medianFundSize = round(medianFundSize / 1000000);
    AC(uniqueRow(i)) = medianFundSize;
end
```

## Last row

```
for i = 1:length(uniqueCol)
    f = find(tbl.column == uniqueCol(i));
    values = tbl.fundSize(f);
    medianFundSize = median(values);
    medianFundSize = round(medianFundSize / 1000000);
    AR(uniqueCol(i)) = medianFundSize;
end
```

## "All" cell

```
All = round(median(tbl.fundSize) / 1000000);

TN = [TN transpose(AC)];
TN = [TN ; zeros(1,51)];
TN(end,:) = [AR All];

% writematrix(TN,'2018Results.xls','Sheet',3);
```

## Scatter plot of active share vs fund size

Remove index funds

```
nonindex = foundFunds.fundShortName(foundFunds.indexFund == 0);
m = ismember(tbl.fundShortName,nonindex);
newtbl = tbl(m,:);

x = newtbl.fundSize / 1000000000;
y = newtbl.activeShareMethod1;
scatter(x,y,25,[0.4 0.4 0.4],'o','filled')
ax = gca;
ax.XAxis.Exponent = 0;
ax.YLim = [0 100];
ax.YLabel.String = 'Active Share (%)';
ax.XLabel.String = 'Fund Size (billion NOK)';
hold on
l = lsline ;
set(l,'LineWidth',1,'Color','k')
```

## Funds plotted along two dimensions

Selected funds

```
load('plotFunds.mat');

fields = string(fieldnames(plotFunds));

for i = 1:length(fields)
    % Extract 2018 results
    fund = plotFunds.(fields(i))(TR,:);
    if ~isempty(fund)
        % Extract last 2018 result
        fund = fund(end,:);
        % Convert to percentages
        fund.activeShareMethod1 = fund.activeShareMethod1 * 100;
        fund.annualTrackingError = fund.annualTrackingError * 100;
        % Save
        output{i} = fund;
    end
end

tbl = vertcat(output{:});

fundShortName = tbl.fundShortName;

for i = 1:length(fundShortName)
    idx = ismember(foundFunds.fundShortName,fundShortName(i));
    tbl.OBIname(i) = foundFunds.OBI_Name(idx);
end
```

```
x = tbl.annualTrackingError;
y = tbl.activeShareMethod1;
scatter(x,y,25,[0.4 0.4 0.4],'o','filled')
ax = gca;
ax.XTick = [0:2:16];
ax.YTick = [0:10:100];
ax.XLim = [0 14];
ax.YLim = [0 100];
ax.YLabel.String = 'Active Share (%)';
ax.XLabel.String = 'Tracking Error (%)';

name = tbl.OBIname;
dx = 0.2; dy = 0.2;
text(x+dx,y+dy,name,'FontSize',8);
```

```
set(gca,'XLim',[0 10],'YLim',[0 90]);
```