```
dm 'log;clear';
dm 'output;clear';
dm 'odsresults;clear';
proc datasets library = work kill; quit;

* SAS 1: Make a data request in Beta Suite by WRDS
to get the daily market betas for US common stocks;
* Download and rename the output data;
* Move the data file to my_lib folder;
* SAS 2: Add the beta measure to the monthly stock data set *******************************;

* Keep only the last observation of each month to get monthly betas;
data daily_beta_data;
      set my_lib.daily_beta_data;
      t = intnx('month',date,1,'end');
      format t yymmddn8.;
run;
proc sort data = daily_beta_data; by permno t date; run;
data monthly_beta_data (keep = permno t b_mkt);
      set daily_beta_data;
      by permno t date;
      if last.t;
run;

* Generate the month-end date of each monthly stock observation;
data monthly_stock_data1;
      set my_lib.assignment1_data;
      t = intnx('month',date,0,'end');
      format t yymmddn8.;
run;
proc sort data = monthly_stock_data1; by permno t; run;

* Add beta information to the monthly stock data set;
data monthly_stock_data2;
      merge monthly_stock_data1 (in = a) monthly_beta_data (in = b);
      by permno t;
      if a and b and missing(b_mkt) = 0;
run;
```

```
* SAS 3: Generate Mktcap_CPI and Size variables ******************************************;

* Load CPI data;
proc import out = CPI (rename = (cpiaucsl = cpi observation_date = cpi_date))
     datafile = "&my_directory\CPIAUCSL.xls"
     dbms = xls replace;
     sheet = "Sheet1";
     namerow = 11;
     datarow = 12;
run;

* CPI in June of each year;
data CPI_Jun;
     set CPI (where = (month(cpi_date) = 6));
     t = year(cpi_date);
     keep t cpi;
run;
* CPI in Dec, 2012;
%let CPI_2012 = 231.221;

* Calculate Mktcap_CPI, Size, and log_BM variables;
data monthly_stock_data2;
     set monthly_stock_data2;
     if month(date)>6 then t = year(date);
     else t = year(date)-1;
run;
proc sort data = monthly_stock_data2; by t date permno; run;
data monthly_stock_data3;
     merge monthly_stock_data2 (in = a) CPI_Jun (in = b);
     by t;
     if a;

     ME_Jun_CPI = (ME_Jun/cpi)*&CPI_2012;
     size = log(ME_Jun);
     size_CPI = log(ME_Jun_CPI);
     log_BM = log(BM);
     keep permno date year exchcd siccd retadj eretadj altprc_lag1 ME_lag1
          b_mkt size size_CPI BM log_BM;
run;
```

```
* SAS 4: Winsorize stock characteristic variables ****************************************;

* Rename characteristic variables;
data monthly_stock_data3;
      set monthly_stock_data3;
      rename b_mkt = b_mkt_o size = size_o size_CPI = size_CPI_o BM = BM_o log_BM = log_BM_o;
run;

* Calculate 0.5% and 99.5% level of each characteristic variable
on a monthly basis;
proc sort data = monthly_stock_data3; by date; run;
proc univariate data = monthly_stock_data3 noprint;
      by date;
      var b_mkt_o size_o size_CPI_o BM_o log_BM_o;
      output out = bounds pctlpts = 0.5 99.5 pctlpre = b_mkt_ size_ size_CPI_ BM_ log_BM_;
run;

* Merge the bounds with the monthly stock data
and winsorize characteristic variables;
data monthly_stock_data4;
      merge monthly_stock_data3 bounds;
      by date;

      array original(5) b_mkt_o size_o size_CPI_o BM_o log_BM_o;
      array winsorized(5) b_mkt size size_CPI BM log_BM;
      array l_bound(5) b_mkt_0_5 size_0_5 size_CPI_0_5 BM_0_5 log_BM_0_5;
      array u_bound(5) b_mkt_99_5 size_99_5 size_CPI_99_5 BM_99_5 log_BM_99_5;

      do ii = 1 to 5;
           if original(ii)<l_bound(ii) then winsorized(ii) = l_bound(ii);
           else if original(ii)>u_bound(ii) then winsorized(ii) = u_bound(ii);
           else winsorized(ii) = original(ii);
           end;

      drop b_mkt_0_5--log_BM_99_5 ii b_mkt_o size_o size_CPI_o BM_o log_BM_o;
run;
```

```
* SAS 5: Calculate summary statistics;

%let varlist = b_mkt size size_CPI BM log_BM;

ods exclude all; * suppress ods output;
* ods:  ODS stands for output delivery system.
It is mostly used to format the output data of a SAS program
to nice reports which are good to look at and understand.;

* Calculate summary statistics of variables in the "varlist" across stocks in each month,
and stack the results in stats_by_year data set;
proc sort data = monthly_stock_data4; by date permno; run;
proc means data = monthly_stock_data4
                           mean std skew kurt min p5 p25 median p75 max n
                           stackodsoutput nolabels;
     by date;
     var &varlist;
     ods output summary = stats_by_month;
     * proc means results are stored in the ods table called "summary";
run;

ods exclude none;

* Calculate the time-series-means of the summary statistics for the variables in the "varlist";
proc sort data = stats_by_month; by variable date; run;
proc means data = stats_by_month mean nolabels noprint;
     by variable;
     var mean stddev skew kurt min p25 median p75 max n;
     output out = stats (drop = _TYPE_ _FREQ_)
          mean(mean stddev skew kurt min p25 median p75 max n)
               = mean stddev skew kurt min p25 median p75 max n;
run;

* Reorder the variables;
data stats;
     set stats;
     if variable = "b_mkt" then row_num = 1;
     else if variable = "size" then row_num = 2;
     else if variable = "size_CPI" then row_num = 3;
     else if variable = "BM" then row_num = 4;
     else if variable = "log_BM" then row_num = 5;
run;
proc sort data = stats; by row_num; run;
```

```
* SAS 6: Calculate correlations ***********************************************************;

* Calculate correlations in each month;
proc corr data = monthly_stock_data4 outp = pcorr_by_month (where = (_TYPE_ = "CORR")) noprint;
     by date;
     var &varlist;
run;

* Calculate the time-series-means of the correlations for variables in the "varlist";
proc sort data = pcorr_by_month; by _name_ date; run;
proc means data = pcorr_by_month mean nolabels noprint;
     by _name_;
     var &varlist;
     output out = pcorr (keep = _NAME_ &varlist) mean(&varlist) = &varlist;
run;

* Reorder the variables;
data pcorr;
     set pcorr;
     if _NAME_ = "b_mkt" then row_num = 1;
     else if _NAME_ = "size" then row_num = 2;
     else if _NAME_ = "size_CPI" then row_num = 3;
     else if _NAME_ = "BM" then row_num = 4;
     else if _NAME_ = "log_BM" then row_num = 5;
run;
proc sort data = pcorr; by row_num; run;
```

```
* SAS 7: Dependent-sort stocks into 25 portfolios based on size and BM *******************************;

* Check if we have many stocks with the same value of size or BM in any month;
data test1;
      set monthly_stock_data3 (keep = date size_o);
run;
proc sort data = test1; by date size_o; run;
data test1;
      set test1;
      by date size_o;

      retain N;
      if first.size_o then N = 1;
      else N = N+1;

      if last.size_o then output;
run;
proc sort data = test1; by descending N; run;
data test1;
      set test1;
      if N>1;
run;

data test2;
      set monthly_stock_data3 (keep = date BM_o);
run;
proc sort data = test2; by date BM_o; run;
data test2;
      set test2;
      by date BM_o;

      retain N;
      if first.BM_o then N = 1;
      else N = N+1;

      if last.BM_o then output;
run;
proc sort data = test2; by descending N; run;
data test2;
      set test2;
      if N>1;
run;
```

```
* Since only up to three stocks have the same size_o value
* and only up to two stocks have the same BM value in a certain month,
* we will define the i-th portfolio as the set of stocks with B(i-1)<=X<B(i)
* instead of B(i-1)<=X<=B(i) as in BEM;

* Calculate size breakpoints as 20th, 40th, 60th, and 80th size percentiles
among NYSE stocks in each month;
proc univariate data = monthly_stock_data4 (where = (exchcd in (1,31))) noprint;
     by date;
     var size;
     output out = size_breakpoints pctlpts = 20 40 60 80 pctlpre = size_;
run;

* Merge the size breakpoints with the monthly stock data
and define size sorted portfolios;
data monthly_stock_data5;
     merge monthly_stock_data4 size_breakpoints;
     by date;

     if size < size_20 then p1 = 1;
     else if size < size_40 then p1 = 2;
     else if size < size_60 then p1 = 3;
     else if size < size_80 then p1 = 4;
     else p1 = 5;
run;
```

```
* Calculate BM breakpoints as 20th, 40th, 60th, and 80th BM percentiles
* among all stocks in each size sorted portfolio in each month;
proc sort data = monthly_stock_data5; by date p1; run;
proc univariate data = monthly_stock_data5 noprint;
      by date p1;
      var BM;
      output out = BM_breakpoints pctlpts = 20 40 60 80 pctlpre = BM_;
run;

* Merge the BM breakpoints with the monthly stock data
and define BM sorted portfolios in each size sorted portfolio;
data monthly_stock_data6;
      merge monthly_stock_data5 BM_breakpoints;
      by date p1;

      if BM < BM_20 then p2 = 1;
      else if BM < BM_40 then p2 = 2;
      else if BM < BM_60 then p2 = 3;
      else if BM < BM_80 then p2 = 4;
      else p2 = 5;
run;

* Save the final data set in a local folder;
data my_lib.assignment2_data;
      set monthly_stock_data6;
run;
proc sort data = my_lib.assignment2_data; by date permno; run;
```

```
* Calculate the time-series average number of stocks in each portfolio;
proc sort data = monthly_stock_data6; by date p1 p2; run;
proc means data = monthly_stock_data6 n nolabels noprint;
     by date p1 p2;
     var permno;
     output out = nstocks_per_p n = nstocks;
run;
proc sort data = nstocks_per_p; by p1 p2; run;
proc means data = nstocks_per_p mean nolabels noprint;
     by p1 p2;
     var nstocks;
     output out = nstocks_per_p (drop = _TYPE_) mean = ave_nstocks;
run;
proc transpose data = nstocks_per_p out = nstocks_per_p (drop = _NAME_ _LABEL_) prefix = p2_;
     by p1;
     id p2;
     var ave_nstocks;
run;
```