This is the perl documentation for MS3. Most of the code is the same, I will highlight anything that is new.

use LWP::Simple;  #allow access to pull the file off the web

#getstore('http://www.mdl.nws.noaa.gov/~naefs\_ekdmos/text/naefs\_tempcdf\_00.txt','AllStations.txt') or die 'unable to get page';

getstore('http://www.ems.psu.edu/~young/meteo473/Data/EKDMOS00.txt','AllStations.txt') or die 'unable to get page';

#gets the textfile from the webpage and stores it in a textfile

Gathers information from a specified web page and puts it into a textfile.

$file = "AllStations.txt";                    #textfile of stations  
open (INFILE, "< $file") or die "Can't open $file for read: $!"; #opens the infile handle

open (OUTFILE1, ">nws1.txt") or die "Can't open for write: $!"; #opens the outfile handle names the file  
open (OUTFILE2, ">nws2.txt") or die "Can't open for write: $!"; #opens the outfile handle names the file  
open (OUTFILE3, ">nws3.txt") or die "Can't open for write: $!"; #opens the outfile handle names the file for R code  
Opens our files so we can use them later.

print OUTFILE3 "Date ExpectedProfit\n";                #prints a header  
Prints the header to a textfile for the R code

print "What is the cost of your electricity?\n";        #query's user about cost of electricity  
$cost = <>;                            #input from the user  
print "Thank you.\n";

$c = 0;                                #sets the time counter  
Input for cost from the user, and setting a counter

while (<INFILE>) {                        #while we are reading through AllStations  
($date,$station,$c05,$c10,$! c20,$c30 ,$c40,$c50,$c60,$c70,$c80,$c90,$c95,$mean,$sd)=split(' ',$\_,15); #split into 15 parts

if ($date=='VALID')                        #if the first line is the word VALID  
{}                                #do nothing with that line  
elsif ($date>1){                        #all other lines  
$date=~/(\d\d\d\d)(\d\d)(\d\d)(\d\d)/;                #split date into year, month, day, hour  
While reading through the file, we split each line into its 15 parts

if ($c == 0)                            #for the first pass through   
     {  
      $file1 = $station."."."txt";                #renames the file for each station      
      open (OUTFILE1, "> $file1") or die "Can't open $file1 for write: $!" #opens output filehandle                  
     }  
If the time counter is equal to zero, we the put them into files named after the station models

$c00=$c10+(2\*($c05-$c10));                      #linear extrapolation for the 0 value  
$c1h=$c90+(2\*($c95-$c90));                    #linear extrapolation for the 100 value  
Interpolating for the first and last data point

    if ($n == 0)                        #I just need to do this once    
     {       
      $month[$c] = $2;                         #puts the month in a variable   
      $day[$c] = $3;                        #puts the day in a variable   
      $year[$c] = $1;                        #puts the year in a variable   
      $hour[$c] = $4;                        #puts the hour in a variable  
     }

The setup, I just need one set of date data. We use the date to keep track of what outputs to the textfile, and for easier understanding of the textfile.

$c = $c + 1;                            #increments time counter  
print OUTFILE1 "$station $c00 $c05 $c10 $c20 $c30 $c40 $c50 $c60 $c70 $c80 $c90 $c95 $c1h\n";  
 #prints what we need to the outfile  
 if ($c % 75 == 0) #if the time counter is evenly divisible by 75, do this  
     {   
      $n = $n + 1;  #station counter  
      $c = 0;   #reset time counter  
      close OUTFILE1 or die "Cannot close $file1: $!";        #closes the current OUTFILE1  
     } #closes the if

If the time counter $c is evenly divisible by 75: The station counter increments(so that the next station model is used as the file name, the time counter is reset to zero, and the current Output File Handle closes, to be reopened in the next iteration  
  } #the elsif  
} #closes the while  
close INFILE or die "Cannot close $file: $!"; #closes the INFILE since we are done with it

close OUTFILE1 or die "Cannot close $file1: $!";        #closes the last current OUTFILE1

The elsif and while loops close, and we close the Infile sice we are done with it

$x = 0;                                #set counter x equal to 0  
$cdf[0] = 0;                            #set the array cdf[0] equal to 0  
Set a counter and the first part of the array  
  
for ($z = 1; $z <= 13; $z++)                    #for loop for z, to keep track of the array   
{  
    if ($z <= 2 || $z >= 11)                #for the cdf values of 5, 10 and 95, 100       
    {  
        $cdf[$z] = $cdf[$x]+ 0.05;                #makes the above cdf values 0.05  
    $x++                            #x increments, one less than z  
    }  
    else                            #for all others  
    {  
    $cdf[$z] = $cdf[$x]+ 0.10; #make all the other cdf values 0.10  
    $x++;                            #x increments, one less than z  
    }  
} #closes the for loop, with the z counter  
The for loop opens and starts the set up of a variable array called cdf, which we use later to set the variable Ctop

$c = 0;                                #resets the date counter c  
Resets a counter. The “do” is new but I’ll talk about it later

do{  
print "What station model would you like to evaluate?\n";    #query's user about station model  
$stationu = <>;                            #input from the user  
chomp $stationu; #chomps the input  
print "Thank you.\n";  
Request from user regarding which station they would like to evaluate. The user is required to know which stations are available. The program does not print any output if the station does not exist and there is no error message.

open (FH, $stationu.".txt") ||  
    die "ERROR Unable to open Dates: $!\n";                    #opens up textfile and assigns it a file handle  
  
@array = <FH>;                            #puts the textfile into an array  
  
close FH;                            #closes the file handle  
Opens a file handle and the station model requested by the user.   Puts the entire station model into an array, then closes the file handle.

for($i = 0; $i <= 74; $i++) {                    #looping through the 75 values in each station      
  
    @Temp = split(" ", @array[$i]);                #split the file array into a Temperature array   
  
$n = ($Temp[12] - $Temp[1]) / 0.1;                #number of bins

$T = $Temp[1];                            #Sets T for incrementation  
$Ntemp[0] = $Temp[1];                        #Sets new temp for zero value  
$CDF[0] = 0;                            #sets the CDF  
$Tbot = $Temp[1];                         #sets the lower bound on temp  
$Cbot = $cdf[0];                        #sets the lower bound on cdf  
$j = 1;                                #counter for changing Ttop and Ctop  
$Ttop = $Temp[$j];                        #sets the upper bound on temp  
$Ctop = $cdf[$j];                        #sets the upper bound on cdf  
$k = 0;                                #counter for differnece, pdf and temp  
This loops through all 75 lines of the array, and all of the initial value assignments are made for the variables

for ($p=1; $p<=$n; $p++)                    #we are going to loop through n times, all of the bins  
{  
  $T = $T + 0.1;                        #increment temperature  
  
  if ($T > $Ttop)                    #if what we are incrementing equals the top of the bin  
    {  
      $j = $j + 1;                        #increment my changer  
      $Tbot = $Ttop;                        #change top to bottom for temp  
      $Ttop = $Temp[$j];                    #and create a new top for temp  
      $Cbot = $Ctop;                        #Change top to bottom for cdf  
      $Ctop = $cdf[$j];                        #and create a new top for cdf  
    }  
This is the inner for loop, it loops through all of the bins made. The if loop is only activated when then next iteration of T exceeds the top of the bin, Ttop.

$CDF[$p]=$Cbot +((($T-$Tbot)/($Ttop-$Tbot))\*($Ctop-$Cbot));    #this is my interpolation  
$Ntemp[$p] = $T;                        #creates the array for incremented temp  
  
$pdf[$p] = $CDF[$p] - $CDF[$k];                    #does the difference for the pdf  
$Mid[$p] = ($Ntemp[$p]+$Ntemp[$k])/2;                #calculate middle of the bin  
$k = $k + 1;                            #increment the difference counter  
  
$price[$p] = 1.36 \*(abs($Mid[$p]-55)) +20;            #price of the middle  
} #closes the inside for loop, with the p counter

All of the math associated with the integration, creating the middle of the bin, and the price. Then closes the inner for loop   
  
for ($x = 1; $x <= $n; $x++)

 {   
  $Pindicator = ($price[$x] - $cost);                #calculates the profit  
  
  $dailyp = sprintf("%.2f", $Pindicator \* $pdf[$x]);        #calculates the profit for each bin  
  $profit = sprintf("%.2f", $profit + $Pindicator \* $pdf[$x]);    #calculates profit for the whole week   

  $exprofit = sprintf("%.2f",$profit/$n);            #computes expected profit to 2 decimal places  
  print OUTFILE2 "\n \nThe expected profit is $exprofit \n";    #displays the expected profit  
  
  $std\_dev = sprintf("%.2f",sqrt(abs($exprofit)));        #computes standard deviation to 2 decimal places  
  print OUTFILE2 "The standard deviation of profit is $std\_dev\n"; #dispalys standard deviation  
Calulations: The difference in the price to the customer and the cost of running the power plant, daily profit and accumulated profit. Calculate and print expected profit and standard deviation.

if ($profit < 0)                        #when profit is less than 0  
  {                                #display today's loss in a textfile  
 print OUTFILE2 "You net a loss of $dailyp for $month[$c]-$day[$c]-$year[$c] at $hour[$c] Zulu \n";  
  }  
if ($profit > 0)                        #when profit is greater than 0  
  {                                #display today's profit in a textfile  
   print OUTFILE2 "You net a profit of $dailyp for $month[$c]-$day[$c]-$year[$c] at $hour[$c] Zulu \n";  
  }  
print OUTFILE2 "The accumulated profit is $profit from $month[0]-$day[0]-$year[0] at $hour[0] Zulu to $month[$c]-$day[$c]-$year[$c] at $hour[$c] Zulu\n";                             #output to a textfile  
 } #closes the inside for loop, with the x counter  
Print either net loss or net profit for that hour and the accumulated profit from the beginning of the run up to that hour. Close the inside for loop

print OUTFILE3 $month[$c];  
print OUTFILE3 $day[$c];  
print OUTFILE3 $year[$c];  
print OUTFILE3 " $exprofit\n";                    #prints data to textfile for histograms  
  
$c = $c + 1;                            #increment time counter  
} #closes the outside for loop, with the i counter

Sets up the OUTFILE for the R program to plot the Histogram, increments the time counter, and closes the outside for loop   
print "Did you want to evaluate another Station(y/n)?\n";    #Input from the user to evaluate another station  
$ans = <>;  
chomp $ans;  
}while ($ans eq 'y');                    #ends the Do-While loop  
This is the end of the DO-WHILE loop, we ask if the user wants to do another station. We do not run through the split up of the original textfile we just call the next station model. The only downside at the moment is that it overwrites the other data. Will try to see if we can open a different outfile for the future.

close OUTFILE2 or die "Can't close file: $!"; #closes the outfile  
close OUTFILE3 or die "Can't close file: $!"; #closes the outfile  
Closes the final 2 OUTFILE Handles used