You will have to use a computer in a university lab (e.g. Wells Hall B-Wing).

This *Mathematica* notebook contains a number of useful functions briefly described below. The first time you attempt to use one of these functions a panel will pop up asking "Do you want to evaluate all the initialization cells?" to which you must answer yes.

To enter a given command line you click on the screen whereupon a horizontal line should appear at the cursor. When right brackets are in view on the *Mathematica* panel you want to click at a place where a horizontal line will extend between two such brackets if you desire a new line. If you attempt to type multiple commands into a single bracketed location *Mathematica* will become confused.

Type the command you wish to execute then PRESS THE ENTER KEY ON THE NUMERIC KEYPAD. This is required because *Mathematica* wants to use the return or other enter key to move to the next line. You do nor want to move to a new line. You want to enter a command. That is why you must use the ENTER key on the numeric keypad. You may also use SHIFT and ENTER together instead of using ENTER on the numeric keypad.

To save your work select save from the pull down file menu, which saves it as a *Mathematica* .nb (notebook) file. If you wish to print your work at home, select print, then select the option of saving as a PDF. You will be unable to work with the .nb *Mathematica* file itself unless you have *Mathematica* installed (unlikely) but you can transport and print the .pdf file virtually anywhere.

Click the line below and press ENTER on the numeric keypad.

```
In[194]:= size[{4.5, 7.1, 7.8, 9.1}]

Out[194]= 4
```

Just above, I clicked to open a new line then typed
size[{4.5, 7.1, 7.8, 9.1}]
followed by a press of the numeric keypad ENTER key. Notice that off to the right of the entry there are nested brackets joining the command line and its output 4 = the number of data items in {4.5, 7.1, 7.8, 9.1}.

A complete list of the commands in this notebook and what they do.

size[{4.5, 7.1, 7.8, 9.1}] returns 4
mean[{4.5, 7.1, 7.8, 9.1}] returns the mean 7.125
median[{4.5, 7.1, 7.8, 9.1}] returns the median of the list {4.5, 7.1, 7.8, 9.1}
s[{4.5, 7.1, 7.8, 9.1}] returns the sample standard deviation s=1.93628
sd[{4.5, 7.1, 7.8, 9.1}] returns the n-divisor version of standard deviation s=1.67686
r[x, y] returns the sample correlation $r = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\sqrt{\overline{x^2} - \overline{x}^2}\,\sqrt{\overline{y^2} - \overline{y}^2}}$ for paired data.
sample[{4.5, 7.1, 7.8, 9.1}, 10] returns 10 samples from `{4.5, 7.1, 7.8, 9.1}`
ci[{4.5, 7.1, 7.8, 9.1}, 1.96] returns a 1.96 coefficient CI for the mean from given data
bootci[mean, {4.5, 7.1, 7.8, 9.1}, 10000, 0.95] returns 0.95 bootstrap ci for pop mean
smooth[{4.5, 7.1, 7.8, 9.1}, 0.2] returns the density for data at bandwidth 0.2
smooth2[{4.5, 7.1, 7.8, 9.1}, 0.2] returns the density for data at bandwidth 0.2
overlaid with normal densities having sd = 0.2 around each data value
smoothdistribution[{{1, 700},{4 ,300}}, 0.2] returns the density at bandwidth 0.2
for a list consisting of 700 ones and 300 fours.
popSALES is a file of 4000 sales amounts used for examples
entering $\boxed{\textbf{popSALES}}$ will spill 4000 numbers onto the screen. To prevent
that enter $\boxed{\textbf{popSALES;}}$ instead (the appended semi-colon suppresses output).
betahat[matrix x, data y] returns the least squares coefficients $\hat{\beta}$ for a fit of the model $y = x\beta + \epsilon$.
resid[matrix x, data y] returns the estimated errors $\hat{\epsilon} = y - x\hat{\beta}$ (see **betahat** above).
R[matrix x, data y] returns the **multiple correlation** between the fitted values $x\hat{\beta}$ and data y.
xquad[matrix x] returns the full quadratic extension of a design matrix with constant term
xcross[matrix x] returns the extension of x to include all products of differing columns.
betahatCOV[x matrix, data y] returns the estimated covariance matrix of the vector bet$\hat{a}$hat β.
normalprobabilityplot[data, dotsize] returns a normal probability plot for data (e.g. with dotsize .01).
t[df, conf] returns the t-score used in lieu pf z-score in a CI for confidence conf (t[Infinity, .95] ~ 1.96).
Tprob[t, df] returns P(|T| < t) for df (e.g. T[1.96, Infinity] ~ 0.95).

HW8 is due at the close of class next Wednesday, November 12.

BE SURE TO EVALUATE THIS NOTEBOOK BY SHIFT ENTER APPLIED TO THE LINE size[{...}] ABOVE. ANSWERE YES TO THE QUESTION "DO YOU WISH TO EVALUATE THIS NOTEBOOK" AND TAKE CARE NOT TO DELETE THE BRACKETS AT THE VERY TOP OF THIS FILE (THOSE BRACKETS CONTAIN ESSENTIAL FUNCTIONS THAT YOU NEED, SUCH AS s[ ]).

Part I is comprised of questions pertaining to the basics of regression, including the algebraic/calculus/formula relationships that are part of our treatment of LR (straight line linear regression), and MLR (multiple linear regression in which there may be several independent variables).

LR

1. You will be expected to be able to solve this type of problem without assistance on an exam. A sequence of number pairs (x_i, y_i), $i \le 100$ has

$\overline{x} = 8$ $\overline{x^2} = 80$

$\overline{y} = 10$ $\overline{y^2} = 136$ $\overline{xy} = 96$ (not -96)

Express in formula, evaluated formula, and reduced answer:

a. s_x (n-1 divisor sample sd of x scores)

b. $\hat{\sigma}_x$ (n-divisor sample sd of x scores)

c. $\hat{\sigma}_y$

d. r

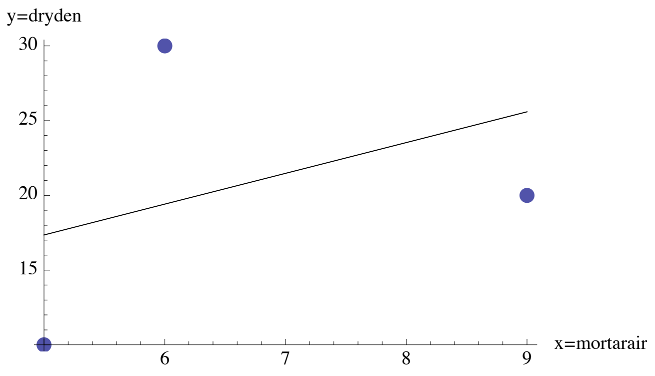
e. Slope $\hat{\beta}_1$ of estimated line of regression of y on x.

f. Sketch the line of regression identifying two points on that line with their formula and numerical values.

g. From (d), determine the fraction of $\overline{y^2}-\overline{y}^2$ explained by regression of y on x.

h. Express the multiple correlation r_{MLR} mathematically in terms of y_i values and fitted values $\hat{y}_i$, Determine its value from r.

2. For the scatterplot and its regression of y on x, given below, determine the requested quantities. You will need a calculator.



a. Determine $(\overline{x}, \overline{y})$ and identify it on the regression line.

b. Determine the slope of the regression line from formula calculation and observe that the picture is correct on that.

c. Determine the fitted value for the second point from the left.

d. Determine the correlation r between x and y scores.

e. Determine the correlation r_{MLR} between fitted scores and y scores and verify that indeed it is equal to | r | where r is the correlation from (d).

d. Determine the slope of regression.

e. Give the 3 by 2 design matrix xmortarair for a matrix formulation of this LR.

f. What regression quantities are produced by the *Mathematica* codes below?

PseudoInverse[xmortarair].dryden

Inverse[Transpose[xmortarair].xmortarair] $\frac{3-1}{3-2} s[\text{drydenresid}]^2$

MLR.

Using Little software7 routines solve the following exercises/examples from your textbook. Compare your solutions with those given in the book.

84 of page 498.

13.12 page 533.

47 of page 546.

52 of page 549.

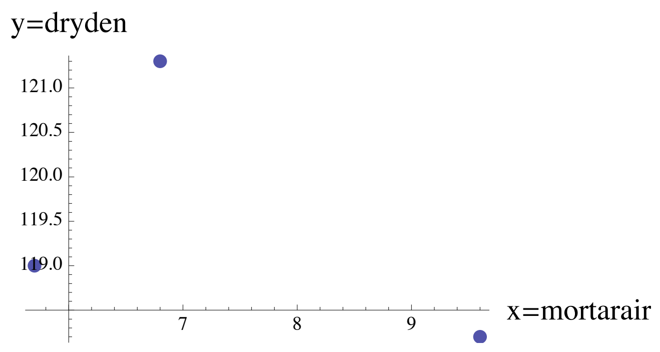
Some routines used in HW7 are copied below.

```
In[153]:= mortarair = {5.7, 6.8, 9.6};

In[154]:= dryden = {119, 121.3, 118.2};

In[155]:= Length[mortarair]

Out[155]= 3
```

You may wish to use a different point size in your plot for best legibility.

```
In[156]:= ListPlot[Table[{mortarair[[i]], dryden[[i]]}, {i, 1, 3}],
          AxesLabel → {"x=mortarair", "y=dryden"}, PlotStyle -> PointSize[0.03]]
```

Out[156]=



```
In[157]:= xmortarair = Table[{1, mortarair[[i]]}, {i, 1, 3}];

In[158]:= MatrixForm[xmortarair]

Out[158]//MatrixForm=
```

$$\begin{pmatrix} 1 & 5.7 \\ 1 & 6.8 \\ 1 & 9.6 \end{pmatrix}$$

PseudoInverse is a least squares solver applicable to systems of linear equations. It produces the unique solution of simultaneous linear equations in several variables (such as the normal equations of Least Squares) if there is one. If not, it produces a particular choice of a least squares solution known as the Moore - Penrose Inverse. In the present example, the matrix formulation of the equations of our linear model is

$$\begin{pmatrix} 119.0 \\ 121.3 \\ 118.2 \end{pmatrix} \sim \begin{pmatrix} 1 & 5.7 \\ 1 & 6.8 \\ 1 & 9.6 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$$

Because the points (x, y) do not fall on a line (see the plot above) there is no exact solution of these 3 equations in only 2 unknowns. The least squares solver uses a Pseudo-Inverse to find a Least Squares solution

$$\begin{pmatrix} 1 & 5.7 \\ 1 & 6.8 \\ 1 & 9.6 \end{pmatrix}^{-1} \begin{pmatrix} 119.0 \\ 121.3 \\ 118.2 \end{pmatrix} \sim \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \text{ (Careful! It is not a genuine inverse, only LS.)}$$

Observe the little dot in the code below. It denotes matrix product and is very important!

```
In[159]:= betahatmortar = PseudoInverse[xmortarair].dryden

Out[159]= {122.315, -0.38211}
```

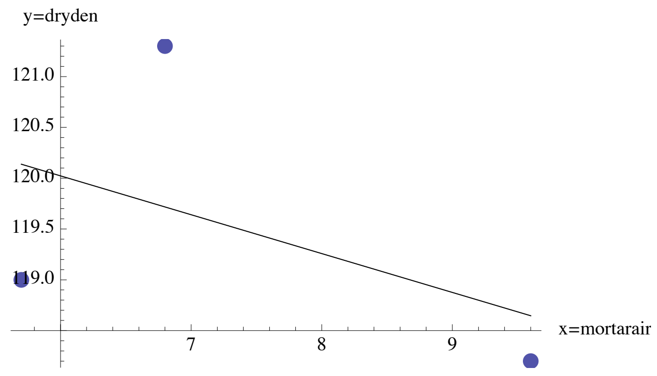
From the line above, the LS estimated slope and intercept are 122.315 and - 0.38211 respectively. Next, find the fitted values $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$.

```
In[160]:= drydenhat = xmortarair.betahatmortar

Out[160]= {120.137, 119.717, 118.647}
```

Next, overlay the LS line on the plot of (x, y). Notice that the code below asks that *Mathematica* join the line of LS fitted values, i.e. the line joins points $(x_i, \hat{y}_i)$. See that LS produces fitted values falling perfectly on a line (otherwise *Mathematica* would have plotted a broken zig-zag line).

```
In[161]:= Show[ListPlot[Table[{mortarair[[i]], dryden[[i]]}, {i, 1, 3}],
            AxesLabel → {"x=mortarair", "y=dryden"}, PlotStyle -> PointSize[0.03]],
          Graphics[Line[Table[{mortarair[[i]], drydenhat[[i]]}, {i, 1, 3}]]]]
```

Out[161]=



Examine the plot above. Check visually that the intercept is indeed $\hat{\beta}_0$ (calculated above) and the slope is indeed $\hat{\beta}_1$. Check visually that the heights of the regression line at the mortarair values {5.7, 6.8, 9.6} are indeed your previously calculated fitted values {120.137, 119.717, 118.647}. Also by eye, see that your residuals, calculated next, are indeed the signed vertical gaps between the points of the plot and the regression line.

Again, you will be working with *all* the data of 12.11.

```
In[162]:= drydenresid = dryden - drydenhat

Out[162]= {-1.13685, 1.58347, -0.44662}
```

Matrix Setup gives covariances of the estimates. Let x denote the matrix whose first column is all ones and whose second column holds the x-values {5.7, 6.8, 9.6}.

$$x = \begin{pmatrix} 1 & 5.7 \\ 1 & 6.8 \\ 1 & 9.6 \end{pmatrix}$$

This is called the ***design*** matrix. The probability model, stated in matrix form, is simply written $y = x.\beta + \epsilon$ where the errors ϵ_i are assumed to be statistically independent random variables having the same normal distribution with mean 0 and some unknown sd $\sigma > 0$.

$$\begin{array}{ccccccc} \begin{pmatrix} 119.0 \\ 121.3 \\ 118.2 \end{pmatrix} & = & \begin{pmatrix} 1 & 5.7 \\ 1 & 6.8 \\ 1 & 9.6 \end{pmatrix} & \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} & + & \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix} \\ y & = & x & \beta & + & \epsilon \end{array}$$

When writing this matrix model we drop the dot and just write $y = x\beta + \epsilon$. ***Mathematica* needs the dot however.**

A very nice thing happens. The variances and covariances of the estimators $\hat{\beta}_0$, $\hat{\beta}_1$ are just the entries of the matrix $(x^{tr}\, x)^{-1} \sigma^2$, provided it exists. This is always the case if the columns of x are not linearly dependent.

In *Mathematica,* $\boldsymbol{(x^{tr}\, x)^{-1}}$ is coded Inverse[Transpose[x].x].

The preferred estimator of σ^2 is $\hat{\sigma}^2 = \frac{n-1}{n-d} s^2_{\text{residuals}}$, where d is the number of columns of the design matrix x (for straight line regression above d = 2).

So the least squares estimators of intercept and slope have variances and covariance that are *estimated* by the entries of the following matrix:

```
In[163]:= Inverse[Transpose[xmortarair].xmortarair] (3 - 1)/(3 - 2) (s[drydenresid])^2

Out[163]= {{28.1713, -3.6432}, {-3.6432, 0.494552}}

In[164]:= MatrixForm[%]

Out[164]//MatrixForm=
 ( 28.1713   -3.6432  )
 ( -3.6432   0.494552 )
```

From the above, we estimate the variance of $\hat{\beta}_0$ to be 28.1713, the variance of $\hat{\beta}_1$ to be 0.494552, and the covariance of $\hat{\beta}_0$ with $\hat{\beta}_1$ (same as cov of $\hat{\beta}_1$ with $\hat{\beta}_0$) to be -3.6432.

So a 95% CI for β_0 (the true intercept absent errors of observation) is

$$122.315 \pm t\sqrt{28.1713}$$

and a 95% CI for β_1 (the true slope absent errors of observation) is

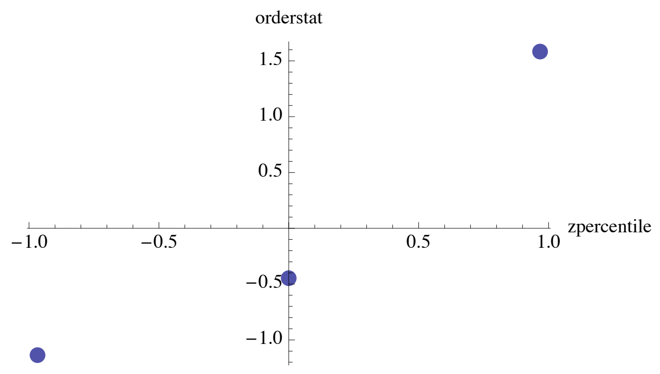
$$-0.38211 \pm t\sqrt{0.494552}$$

where t is for degrees of freedom d-2 and $\alpha = 0.025$ (for 95% confidence). This use of t results in an exact ci provided the measurements (x, y) are from a process under statistcal control.

For a ***partial*** check on the ***normal errors assumption*** of the probability model it is customary to perform a ***normal probability plot for the residuals*** to see if it departs very much from a straight line. Since n is only 3, not much can be learned from the plot. But when you do the same for the full data of Example 12.11 you can address the issue more confidently.

```
In[167]:= normalprobabilityplot[drydenresid, 0.03]
```

Out[167]=



Here is the correlation between the independent variable mortarair and the dependent variable dryden. Squaring it gives the coefficient of determination which is "the fraction of var y accounted for by regression on x." It is not very large in this tiny example.

```
In[168]:= r[mortarair, dryden]

Out[168]= -0.477429

In[169]:= %^2

Out[169]= 0.227938
```

Joe Meleca offered his data input for HW8.

```
In[170]:= 84.

Out[170]= 84.

In[171]:= inletTemp = { 7.68, 6.51, 6.43, 5.48, 6.57, 10.22, 15.69, 16.77, 17.13,
             17.63, 16.72, 15.45, 12.06, 11.44, 10.17, 9.64, 8.55, 7.57, 6.94, 8.32, 10.50,
             16.02, 17.83, 17.03, 16.18, 16.26, 14.44, 12.78, 12.25, 11.69, 11.34, 10.97 };

In[172]:=
          removalPerc = { 98.09, 98.25, 97.82, 97.82, 97.82, 97.93, 98.38, 98.89, 98.96, 98.90,
             98.68, 98.69, 98.51, 98.09, 98.25, 98.36, 98.27, 98.00, 98.09, 98.25, 98.41,
             98.51, 98.71, 98.79, 98.87, 98.76, 98.58, 98.73, 98.45, 98.37, 98.36, 98.45 };
```

13.12

```
In[173]:= force = { 30, 40, 30, 40, 30, 40, 30, 40, 30, 40, 30, 40,
             30, 40, 30, 40, 25, 45, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35 };

In[174]:= power = { 60, 60, 90, 90, 60, 60, 90, 90, 60, 60, 90, 90,
             60, 60, 90, 90, 75, 75, 45, 105, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75 };

In[175]:= temp = { 175, 175, 175, 175, 225, 225, 225, 225, 175, 175, 175, 175, 225, 225,
             225, 225, 200, 200, 200, 200, 150, 250, 200, 200, 200, 200, 200, 200, 200, 200 };

In[176]:= time = { 15, 15, 15, 15, 15, 15, 15, 15, 25, 25, 25, 25,
             25, 25, 25, 25, 20, 20, 20, 20, 20, 20, 10, 30, 20, 20, 20, 20, 20, 20 };


In[177]:= xxStats = Table[ {1, force[[i]], power[[i]], temp[[i]], time[[i]]}, {i, 1, 30}];

In[178]:= yyStrength =
           { 26.2, 26.3, 39.8, 39.7, 38.6, 35.5, 48.8, 37.8, 26.6, 23.4, 38.6, 52.1, 39.5, 32.3, 43.0,
             56.0, 35.2, 46.9, 22.7, 58.7, 34.5, 44.0, 35.7, 41.8, 36.5, 37.6, 40.3, 46.0, 27.8, 40.3 };

In[179]:= 47.

Out[179]= 47.

In[180]:= plastics = { 18.69, 19.43, 19.24, 22.64, 16.54, 21.44, 19.53, 23.97,
             21.45, 20.34, 17.03, 21.03, 20.49, 20.45, 18.81, 18.28, 21.41, 25.11, 21.04,
             17.99, 18.73, 18.49, 22.08, 14.28, 17.74, 20.54, 18.25, 19.09, 21.25, 21.62 };

In[181]:= paper =    { 15.65, 23.51, 24.23, 22.20, 23.56, 23.65, 24.45, 19.39,
             23.84, 26.50, 23.46, 26.99, 19.87, 23.03, 22.62, 21.87, 20.47, 22.59, 26.27,
             28.22, 29.39, 26.58, 24.88, 26.27, 23.61, 26.58, 13.77, 25.62, 20.63, 22.71 };

In[182]:= garbage =  { 45.01, 39.69, 43.16, 35.76, 41.20, 35.56, 40.18, 44.11,
             35.41, 34.21, 32.45, 38.19, 41.35, 43.59, 42.20, 41.50, 41.20, 37.02, 38.66,
             44.18, 34.77, 37.55, 37.07, 35.80, 37.36, 35.40, 51.32, 39.54, 40.72, 36.22 };
```

```
In[183]:= water =    { 58.21, 46.31, 46.63, 45.85, 55.14, 54.24, 47.20, 43.82,
             51.01, 49.06, 53.23, 51.78, 46.69, 53.57, 52.98, 47.44, 54.68, 48.74, 53.22,
             53.37, 51.06, 50.66, 50.72, 48.24, 49.92, 53.58, 51.38, 50.13, 48.67, 48.19 };

In[184]:= xxWaste = Table[ {1, plastics[[i]], paper[[i]], garbage[[i]], water[[i]]}, {i, 1, 30}];

In[185]:= yyEnergyContent =
            { 947, 1407, 1452, 1553, 989, 1162, 1466, 1656, 1254, 1336, 1097, 1266, 1401, 1223, 1216, 1334,
             1155, 1453, 1278, 1153, 1225, 1237, 1327, 1229, 1205, 1221, 1138, 1295, 1391, 1372 };

In[186]:= 52.

Out[186]= 52.

In[187]:= linoleic = { 30, 30, 30, 40, 30, 13.18, 20, 20, 40, 30, 30, 40, 40, 30, 30, 30, 30, 20, 20, 46.82 };

In[188]:= kerosene = { 30, 30, 30, 40, 30, 30, 40, 40, 20, 30, 30, 20, 40, 30, 46.82, 30, 13.18, 20, 20, 30 };

In[189]:= antiox =   { 10, 10, 18.41, 5, 10, 10, 5, 15, 5, 10, 1.59, 15, 15, 10, 10, 10, 10, 5, 15, 10 };

In[190]:= xxPredictors = Table[ {1, linoleic[[i]], kerosene[[i]], antiox[[i]]}, {i, 1, 20}];

In[191]:= yyBetacaro = { .7, .63, .013, .049, .7, .1, .04, .0065,
             .2020, .63, .04, .132, .150, .7, .346, .63, .397, .269, .0054, .064 };

In[192]:= xxPredictors = Table[ {1, linoleic[[i]], kerosene[[i]], antiox[[i]]}, {i, 1, 20}];

In[193]:= yyBetacaro = { .7, .63, .013, .049, .7, .1, .04, .0065,
             .2020, .63, .04, .132, .150, .7, .346, .63, .397, .269, .0054, .064 };

In[195]:= ? s

Info3435475849-3839578

Global`s

Info3435475849-3839578
```

$$s[x\_] := \sqrt{\frac{\text{Mean}\left[(x-\text{Mean}[x])^2\right] \text{Length}[x]}{\text{Length}[x]-1}} \; 1.$$