Descriptive Statistics

Interpreting MathCad Worksheets:

For classes such as this, where it is useful to make documents with math symbols, graphs, etc, I find the software MathCad to be quite useful. This program makes available an extensive library of mathematics functions allowing import, export, and manipulation of data in real-time. It also allows me to document what I have done using familiar mathematics symbols directly comparable to that seen in the text, and lots of words in the worksheet itself. For the purpose of prototyping statistical procedures, and keeping track of what I have done, I find the combination ideal.

Although I recommend MathCad, or somewhat similar Matlab or R, as a way to take the next mathematical step beyond a spreadsheet such as Microsoft Excel, I do not require that you buy MathCad for this course. Instead, I will make all worksheets available to you in both MathCad (*.mcd) and in Adobe Acrobat (*.pdf) formats. Using the examples presented in class, difference between them will appear slight. However, MathCad documents are "dynamic" in the sense that all calculations are dependent on assigning specific values for variables and performing calculations on them. As a result, MathCad documents often depend on an associated datafile called by a read statement somewhere in the document - usually near the top. You will need to download both. By contrast, Acrobat documents are written by MathCad as "static"printed document based on the *appearance* of the "dynamic" document. These need no associated datafiles.

To get started, this worksheet is designed to provide an overview of what you might expect to see in lecture worksheets from now on.

$\text{ORIGIN} \equiv 1$ **< I normally put this in all my worksheets to standardize use of index variables across all my worksheets. It is an example of "global assignment" (keyed by using the symbol ~ on the keyboard). Global assignment means that the value of 1 in this case is assigned to the variable ORIGIN (a special variable available in MathCad for indexing variables).**

^ OK, so now you see how I label things...

Calculations:

$2 + 2 = 4$ **< Calculations are done in the normal way using familiar symbols.**

$\frac{35}{5} = 7$

$6 \cdot 5 = 30$

$\pi = 3.142$ **< Some common mathematical values are built into the program...**

Assignment versus Evaluation:

< Variables may be named at any time. Note, however, that there are two distinct meanings here for what we normally term "equals"!

$\text{Var} := 78$ $\text{Var} = 78$

^ assignment **^ evaluation**

Assignment (using : on the keyboard and shown on the worksheet as :=) means "put the numerical value 79 into a variable I now name Var".

$\text{var2} := 6 \cdot \frac{7}{9}$ $\text{var2} = 4.667$

Evaluation (using = on the keyboard) means "tell me what value is placed in the already named variable Var".

$\text{var3} := \frac{(5 + 3)}{\sqrt{\pi}}$ $\text{var3} = 4.514$

This distinction is important and common to most programming languages... including Statistical Programming languague 'R'.

Data Input:

iris := READPRN("c:/DATA/Biostatistics/iris.txt")

^ Importing data is easy using MathCad's built-in READPRN() function for simple text format data. When prototyping, using an existing dynamic worksheet, I can read in different data files and calculate things in exactly the same way. A worksheet showing how to do a specific statistical test, for instance, is critical for evaluating output from canned programs that might otherwise appear to be a "BLACK BOX".

iris =

	1	2	3	4	5
1	1	5.1	3.5	1.4	0.2
2	2	4.9	3	1.4	0.2
3	3	4.7	3.2	1.3	0.2
4	4	4.6	3.1	1.5	0.2
5	5	5	3.6	1.4	0.2
6	6	5.4	3.9	1.7	0.4
7	7	4.6	3.4	1.4	0.3
8	8	5	3.4	1.5	0.2
9	9	4.4	2.9	1.4	0.2
10	10	4.9	3.1	1.5	0.1
11	11	5.4	3.7	1.5	0.2
12	12	4.8	3.4	1.6	0.2
13	13	4.8	3	1.4	0.1
14	14	4.3	3	1.1	0.1
15	15	5.8	4	1.2	0.2
16	16	5.7	4.4	1.5	0.4

< Evaluation of variable iris.

Note that the display is often a *partial* list that may be scrolled in the normal manner, like a spreadsheet, in a MathCad "dynamic" document. However, it will only appear as the partial list in an Adobe "static" document The variable might also be displayed in matrix form...

When working with a datachart, the shaded row and column numbers in MathCad provide the index for each datum. Note that columns start their numbering with '1'. This is the result of my ORIGIN assignment above. The first column of numbers is merely the row number, so I usually find it convenient to index variables starting at zero.

Species names in column 6 didn't import here as MathCad interpreted the data to be numeric... Statistics programs such as R will do a better job than this. However, we won't worry about it for our purposes here.

Scrolling down one can see that there are 150 rows.

ORIGIN := 0 **< a new global assignment for built-in variable ORIGIN**

iris =

	0	1	2	3	4
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5	3.6	1.4	0.2
5	6	5.4	3.9	1.7	0.4
6	7	4.6	3.4	1.4	0.3
7	8	5	3.4	1.5	0.2
8	9	4.4	2.9	1.4	0.2
9	10	4.9	3.1	1.5	0.1
10	11	5.4	3.7	1.5	0.2
11	12	4.8	3.4	1.6	0.2
12	13	4.8	3	1.4	0.1
13	14	4.3	3	1.1	0.1
14	15	5.8	4	1.2	0.2
15	16	5.7	4.4	1.5	0.4

< Evaluation of variable iris.

Note that MathCad's dynamic logic is *linear* - moving from top to bottom of the page, and also from left to right. Most procedural programming languagues work in a similar way. Scripts written in the statistical language R works this way also.

$SL := iris^{\langle 1 \rangle}$

$SW := iris^{\langle 2 \rangle}$

$PL := iris^{\langle 3 \rangle}$

$PW := iris^{\langle 4 \rangle}$

< New variables are now named and assigned to the values in different columns of the dataset iris using the built-in column function <x>.

Evaluation of variable SL >

SL =

	0
0	5.1
1	4.9
2	4.7
3	4.6
4	5
5	5.4
6	4.6
7	5
8	4.4
9	4.9
10	5.4
11	4.8
12	4.8
13	4.3
14	5.8
15	5.7

Descriptive Statistics:

Vector lengths:

length(SL) = 150 **<- using built-in function length() to evaluate the number of rows (i.e., objects = flowers) inside our assigned variable SL. This is a useful general technique in programming languages including 'R'.**

length(SW) = 150

length(PL) = 150

length(PW) = 150

<- Evaluating the other variables. The result is hardly a surprise, but a useful check anyway in case something went wrong in using function READPRN() above.

mean:

n := length(SL) n = 150

$i := 0..n-1$ **< sets up a list of 150 numbers indexed from 0 to 149.**

$SL_2 = 4.7$ **< indexed value - scroll on the evaluation of vecgtor SL above to verify!**

^ Note: this is the value in the row corresponding to index 2 of variable SL, called by specifying the index (using '[' left bracket on the keyboard in MathCad).

$$X_{bar} := \frac{1}{n}\cdot\sum_i \left(SL_i\right)$$

< X.bar is the name of the variable as typed on the keyboard. The bar part is shown as a subscript in MathCad...

^ Sum values of SL over all rows and divided by n. Note that Σ() is a built in function in MathCad that sums all elements in variable SL (indexed one at a time by values of index variable i).

i =

0
1
2
3
4
5
6
7
8
9
10
11

prototype for mean:

$X_{bar} = 5.843$ **< Evaluation of X_{bar}**

mean(SL) = 5.843 **< compared with MathCad's built-in function mean(). Thus, our explicit calcuation of mean matches the output of MathCad's built-in function mean().** ***This strategy of prototyping methods is critically important.***

median:

$SL_{sort} := sort(SL)$ **< using MathCad's sort() function to rearrange the values of SL in order.**

$midpoint := \frac{n}{2}$ midpoint = 75 **< figuring the midpoint (i.e., halfway) index**

$$medianSL := \frac{1}{2}\left(SL_{sort_{midpoint}} + SL_{sort_{midpoint+1}}\right)$$ **< variable SL.sort indexed by midpoints**

^ When the number of values in a variable are even, the definition of median requires that we average the two closest points.

SL_{sort} =

	0
0	4.3
1	4.4
2	4.4
3	4.4
4	4.5
5	4.6
6	4.6
7	4.6
8	4.6
9	4.7
10	4.7
11	4.8
12	4.8
13	4.8
14	4.8
15	4.8

prototype for median:

medianSL = 5.8

median(SL) = 5.8

< Our explicit calculation matches MathCad's built-in function median(). We now have confidence that we know what MathCad's function median() in fact does!

$\text{mean}(SL) = 5.843$	$\text{median}(SL) = 5.8$
$\text{mean}(SW) = 3.057$	$\text{median}(SW) = 3$
$\text{mean}(PL) = 3.758$	$\text{median}(PL) = 4.35$
$\text{mean}(PW) = 1.199$	$\text{median}(PW) = 1.3$

<- Having prototyped one use each of mean() and median(), we now have confidence that we know how to calculate ALL of these!

geometric mean:

$$\text{Gmean}_{SL} := \sqrt[n]{\prod_i SL_i} \qquad \text{Gmean}_{SL} = 5.786 \qquad e^{\left[\sum_i \frac{\ln\left(\left(SL_i\right)\right)}{n}\right]} = 5.786 \qquad e^{\left(\ln\left(\frac{\sum_i SL_i}{n}\right)\right)} = 5.843$$

^ calculated directly Zar Eq. 3.6 **^ Evaluation** **^ calculated using logs/antilogs Zar Eq. 3.7. Note: sum of logs** **^ wrong way! log of sums...**

prototype for geometric mean:

$\text{gmean}(SL) = 5.786$ **< MathCad's function gmean() confirmed.**

harmonic mean:

$$\text{Hmean}_{SL} := \frac{n}{\sum_i \frac{1}{SL_i}} \qquad \text{Hmean}_{SL} = 5.729$$

^ calculated directly Zar Eq. 3.8 **^ Evaluation**

prototype for harmonic mean:

$\text{hmean}(SL) = 5.729$ **< MathCad's function hmean() confirmed.**

Descriptive Statistics: Variability/Dispersion

sample variance and sample standard deviation:

$$\text{var}_{SL} := \frac{1}{n-1} \cdot \left[\sum_i \left[\left(SL_i - \text{mean}(SL)\right)^2\right]\right]$$

< applying standard formula for sample variance. Variable SL is indexed by previously defined index i with mean(SL) as prototyped above for MathCad's funtion mean().

$\text{standdev}_{SL} := \sqrt{\text{var}_{SL}}$ **< Standard deviation is the square root of variance**

prototype:

$\text{var}_{SL} = 0.686$ $\quad \text{var}(SL) = 0.681$ **< Note the difference! MathCad's built-in function var() must be calculating population, not sample, variance!** ***However, we need to verify this to be certain...***

$\frac{n}{n-1} \cdot \text{var}(SL) = 0.686$ **< This converts population variance into sample variance. Matches our calculation and confirms what MathCad is doing.** ***Note how this prototype allows us to avoid making silly but devastating calculation errors using automated statistical software...*** **Statistical programs often utilize different assumptions, making prototyping *critically* important.**

$\text{standdev}_{SL} = 0.828$ $\text{stdev}(SL) = 0.825$ **<- Again, doesn't match for same reason.**

$\sqrt{\frac{n}{n-1} \cdot \text{var}(SL)} = 0.828$ **<- Converting to sample standard deviation and verifying.**

This section displays the value and power of making prototypes! In statistical analysis, it is *very important* to understand *exactly* what you are doing using a computer-based statistical program. For minor reasons like here, a program may be doing something subtly different different than you expect. Without making a prototype the first time you use a procedure, you might end up reporting, and perhaps trying to publish, an ERROR... THIS CAN BE VERY EMBARRASSING!

Linear Transformations of Mean and Variance:

Often, one has a choice in the units employed in measuring or counting a property. For instance, one might decide to measure temperature in either degrees Celsius or Fahrenheit. Conversion from one measurement to the other typically involves translation (adding or subtracting a constant) and scaling (multiplying a measurement by a constant). Translation and scaling together may be summarized by the following fomula, where x is the original measurment and y is a measurement "transformed" by translation and scaling.

$$y := c \cdot x + b$$

where c is the multiplication constant in scaling, and b is the translation constant.
(Don't worry about the color of b above. This is MathCad's way of telling me that variable b hasn't been assigned a value yet...)

You might recognize the above formula as the equation for a line. As a result, transformations of this kind are often called linear transformations. Zar also refers to this as data "coding".

In statistics, we are interested in what happens to means and variance when original measurements are modified in this way.

translation:

$b := 5$ **< now assigning a value to b**

$\text{translated}_{SL} := SL + b$

^ translated.SL is a new variable "coded" by translation y = x + 5 >

SL =

	0
0	5.1
1	4.9
2	4.7
3	4.6
4	5
5	5.4
6	4.6
7	5
8	4.4
9	4.9
10	5.4
11	4.8
12	4.8
13	4.3
14	5.8
15	5.7

translated_{SL} =

	0
0	10.1
1	9.9
2	9.7
3	9.6
4	10
5	10.4
6	9.6
7	10
8	9.4
9	9.9
10	10.4
11	9.8
12	9.8
13	9.3
14	10.8
15	10.7

$\text{mean}(SL) = 5.843$

$\text{mean}(\text{translated}_{SL}) = 10.843$

$\text{mean}(SL) + b = 10.843$

^ translation shifts the mean value by b

$\text{var}_{SL} = 0.686$ **< evaluation of sample variance calculated above**

$\text{var}_{SLtranslated} := \frac{n}{(n-1)} \cdot \text{var}(\text{translated}_{SL})$ **< sample variance**

$\text{var}_{SLtranslated} = 0.686$

^ translation does nothing to variance. Since standard deviation is the square root of variance, translation does nothing to standard deviation as well.

scaling:

$c := 3$

$\text{scaled}_{SL} := c \cdot SL$

^ scaled$_{SL}$ is a new variable "coded" by scaling factor y = 3x

$\text{mean}(SL) = 5.843$

$\text{mean}(\text{scaled}_{SL}) = 17.53$

$\text{mean}(SL) \cdot c = 17.53$

^ scaling multiplies the mean by the same factor c as each of the values in SL

$\text{var}_{SL} = 0.686$ **< sample variance**

$\text{var}_{SLscaled} := \frac{n}{(n-1)} \cdot \text{var}(\text{scaled}_{SL})$ **< scaled variance**

$\text{var}_{SLscaled} = 6.171$ $c^2 \cdot \text{var}_{SL} = 6.171$ **<- scaling multiplies sample variance by factor c^2.**

$\text{standdev}_{SLscaled} := \sqrt{\text{var}_{SLscaled}}$ $\text{standdev}_{SLscaled} = 2.484$ **<- scaling multiplies sample standard deviation by factor c.**

$c \cdot \text{standdev}_{SL} = 2.484$

SL =

	0
0	5.1
1	4.9
2	4.7
3	4.6
4	5
5	5.4
6	4.6
7	5
8	4.4
9	4.9
10	5.4
11	4.8
12	4.8
13	4.3
14	5.8
15	5.7

scaled$_{SL}$ =

	0
0	15.3
1	14.7
2	14.1
3	13.8
4	15
5	16.2
6	13.8
7	15
8	13.2
9	14.7
10	16.2
11	14.4
12	14.4
13	12.9
14	17.4
15	17.1

linear transformation:

$c := 1.8$ $b := 32$ **<- note that these values convert a degree measurement in Celsius in to the equivalent value on the Fahrenheit scale.**

$\text{transformed}_{SL} := c \cdot SL + b$

^ transformed.SL is a new variable "coded" by linear tranformation y = 1.8x + 32 >

$\text{mean}(SL) = 5.843$

$\text{mean}(\text{transformed}_{SL}) = 42.518$

$c \cdot \text{mean}(SL) + b = 42.518$

^ scaling multiplies the mean by the same factor c as each of the values in SL and adds factor b

$\text{var}_{SL} = 0.686$ **< sample variance**

$\text{var}_{SLtransformed} := \frac{n}{(n-1)} \cdot \text{var}(\text{transformed}_{SL})$ **< scaled variance**

$\text{var}_{SLtransformed} = 2.222$ $c^2 \cdot \text{var}_{SL} = 2.222$

^ scaling multiplies variance by factor c^2 and translation in b has no effect.

SL =

	0
0	5.1
1	4.9
2	4.7
3	4.6
4	5
5	5.4
6	4.6
7	5
8	4.4
9	4.9
10	5.4
11	4.8
12	4.8
13	4.3
14	5.8
15	5.7

transformed$_{SL}$ =

	0
0	41.18
1	40.82
2	40.46
3	40.28
4	41
5	41.72
6	40.28
7	41
8	39.92
9	40.82
10	41.72
11	40.64
12	40.64
13	39.74
14	42.44
15	42.26

$\text{standdev}_{SLtransformed} := \sqrt{\text{var}_{SLtransformed}}$

$\text{standdev}_{SLtransformed} = 1.491$ $c \cdot \text{standdev}_{SL} = 1.491$ **<- scaling multiplies observed standard deviation by factor c and translation in b has no effect.**

Prototype in R:

```
#BIOSTATISTICS 020
#DESCRIPTIVE STATISTICS:

#CALCULATIONS:
2+2
35/5
6*5
pi

#ASSIGNMENT:
var=78
var2=6*(7/9)
var3=(5+3)/(sqrt(pi))

#EVALUATION:
var
var2
var3

#INPUT DATA:
I1=read.table(file.choose())

I1 #EVALUATION OF I1

I2=read.table("c:/DATA/Biostatistics/iris.txt")
I2 #EVALUATION OF I2

#VARIABLE ASSIGNMENTS:
Iris=I2
SL=Iris[,1]
SL=Iris$Sepal.Length

SW=Iris$Sepal.Width
PL=Iris$Petal.Length
PW=Iris$Petal.Width
Species=Iris[,5]

#EVALUATION:
Out=cbind(SL,SW,PL,PW,Species)
Out

#DESCRIPTIVE STATISTICS:

length(SL)
length(Iris$Sepal.Width)
length(Iris[,3])
length(PW)

n=length(SL)
n
SL[2]
SL[3]     #NOTE INDEX RANGE HERE!
Iris[3,1]
```

```
Xbar= (1/n)*sum(SL)
Xbar
mean(SL)

median(SL)

geom_meanSL = prod(SL)^(1/length(SL))
geom_meanSL

harm_meanSL = n/(sum(1/SL))
harm_meanSL

#VARIANCE:
var(SL)     #SAMPLE VARIANCE
Sample_VarSL=((n-1)/n)*var(SL)
Sample_VarSL  #POPULATION VARIANCE

#LINEAR TRANSFORMATION OF MEAN AND
VARIANCE
b=32
c=1.8

SLt=c*SL+b
mean(SLt)
var(SLt)
```