```
(* Adam Beck *)
(* Problem 1*)


(* hosp-heart.nb data *)

(* {M,V} M = one year mortality rate,
percentage of patiuents that died within one year of the
 transplant operation,
V = average annual number of transplants at that center during the same 4 years *)
heart = {{17.9, 27}, {23.1, 4}, {40, 3}, {6.5, 35}, {14.9, 17}, {12.5, 4}, {15.7, 45},
   {9.8, 28}, {24, 6}, {5.0, 10}, {15.4, 13}, {4.8, 7}, {0, 1}, {19.1, 47}, {4.5, 6},
   {15, 56}, {12.5, 4}, {33.9, 8}, {10.7, 9}, {13, 14}, {28.3, 12}, {57.1, 2}, {6.3, 4},
   {10, 3}, {8.3, 12}, {17.5, 10}, {20, 3}, {29.3, 10}, {21.4, 7}, {27.3, 8}, {13.6, 6},
   {21.8, 30}, {36.4, 3}, {18.2, 11}, {33.3, 2}, {20, 4}, {38.5, 7}, {20.8, 18}, {12.2, 19},
   {22.2, 18}, {29, 8}, {0, 9}, {5.7, 9}, {50, 2}, {21.7, 15}, {66.7, 4}, {29.4, 17},
   {12.1, 27}, {10.7, 14}, {6.3, 4}, {16.2, 9}, {21.1, 5}, {17.4, 33}, {23.9, 17},
   {42.9, 2}, {40, 2}, {6.7, 15}, {44.4, 3}, {18.7, 34}, {14.7, 24}, {7.4, 7}, {12.6, 24},
   {9.7, 26}, {44.4, 2}, {16.7, 6}, {15.8, 14}, {83.3, 2}, {10.9, 22}, {13.3, 5},
   {11.1, 5}, {75, 2}, {19, 20}, {14, 13}, {60, 1}, {21.2, 8}, {9.7, 8}, {50, 2}, {25, 14},
   {18.6, 15}, {0.0, 1}, {35.3, 9}, {23.5, 85}, {15.6, 11}, {37.5, 2}, {14.3, 28},
   {14.3, 4}, {16.7, 6}, {20.0, 15}, {13.0, 17}, {9.6, 26}, {66.7, 3}, {30.8, 3},
   {14.0, 13}, {27.5, 10}, {37.5, 8}, {18.9, 13}, {0.0, 4}, {12.2, 44}, {57.1, 4},
   {21.4, 35}, {23.4, 16}, {10.9, 12}, {15.6, 8}, {16.7, 2}, {13.9, 9}, {18.2, 11},
   {11.5, 26}, {18.4, 13}, {16.7, 3}, {20.4, 14}, {40.0, 5}, {20.7, 56}, {19.6, 13},
   {13.5, 9}, {29.9, 36}, {8.4, 21}, {28.4, 24}, {7.7, 23}, {19.3, 29}, {0.0, 1},
   {22.2, 20}, {30.0, 5}, {7.0, 11}, {23.8, 7}, {18.8, 29}, {14.5, 16}, {17.0, 16},
   {20.0, 15}, {6.7, 15}, {11.4, 20}, {100.0, 1}, {31.4, 9}, {17.6, 26}, {19.6, 14}};

(* Split this M and V data into separate
 lists via Transpose[] in order to parse through *)
heartTranspose = Transpose[heart];
MData = heartTranspose[[1]];
VData = heartTranspose[[2]];

(* Define mean, median, quantile, and variance functions *)

mean[x_] := Sum[x[[i]], {i, 1, Length[x]}] / Length[x];
(* Sum elements, divide by length *)

In[38]:= median[x_] := (s = Sort[x]; s[[IntegerPart[.5 * Length[s]]]]);
(* Sort list, take element at index 1/2*length *)

In[39]:= quantile[x_, alpha_] := (s = Sort[x]; s[[IntegerPart[alpha * Length[s]]]])
 (* Sort list, take element at index alpha*length *)

In[40]:= variance[x_] := (m = mean[x]; Sum[(x[[i]] - m)^2, {i, 1, Length[x]}] / Length[x]);
(* difference of every element from mean, squared, times 1/length *)

In[41]:= (* Find the mean, median, q1 and q3, and variance *)
hospMeanM = mean[MData]

Out[41]= 21.9045
```

In[42]:=
```
hospMeanV = N[mean[VData]]
```
Out[42]= 13.8657

In[43]:=
```
hospMedianM = median[MData]
```
Out[43]= 18.2

In[44]:=
```
hospMedianV = median[VData]
```
Out[44]= 10

In[45]:=
```
hospQ1M = quantile[MData, .25]
```
Out[45]= 12.2

In[46]:=
```
hospQ1V = quantile[VData, .25]
```
Out[46]= 4

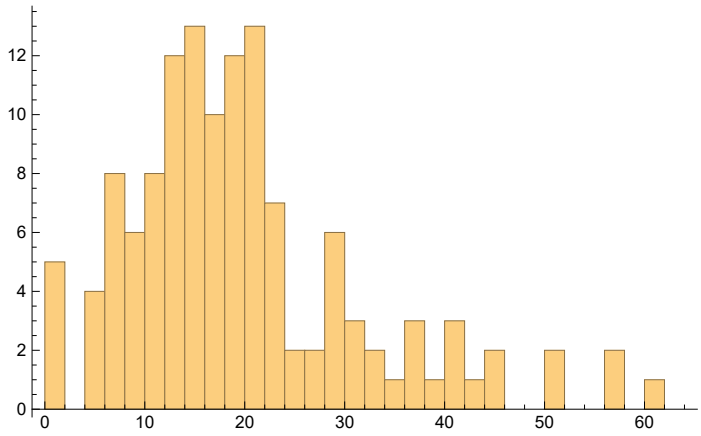
In[47]:=
```
hospQ3M = quantile[MData, .75]
```
Out[47]= 25

In[48]:=
```
hospQ3V = quantile[VData, .75]
```
Out[48]= 17

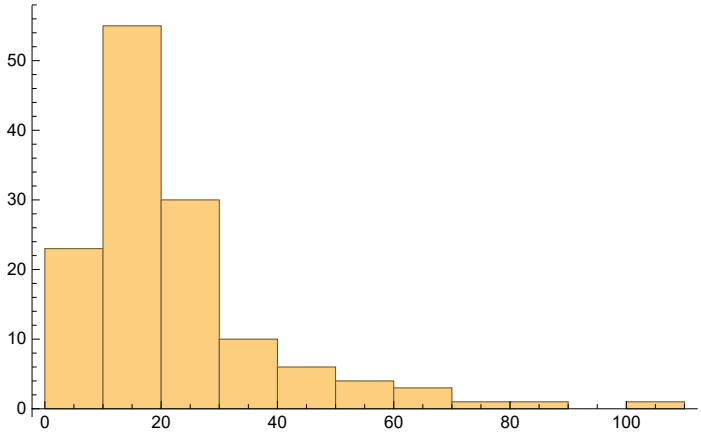
In[49]:=
```
hospVarianceM = variance[MData]
```
Out[49]= 268.634

In[50]:=
```
hospVarianceV = N[variance[VData]]
```
Out[50]= 166.46

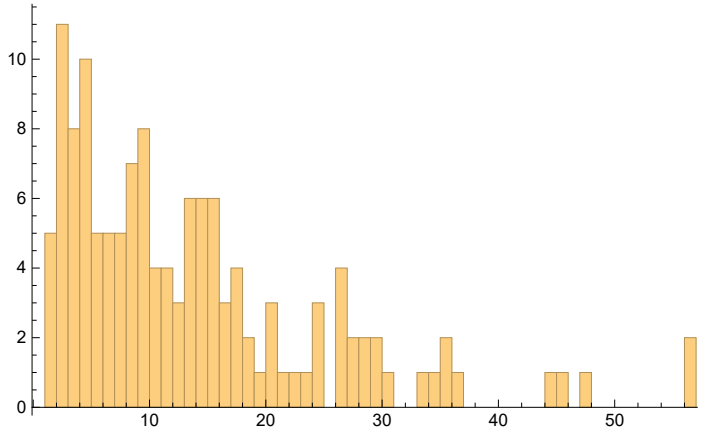
```
(* Histograms using two difference bin sizes *)
(* I will use a bin size Length/2 for a very large bin count,
and Length/10 for a smaller bin count *)

(* MData large bin count *)
Histogram[MData, IntegerPart[Length[MData] / 2]]
```



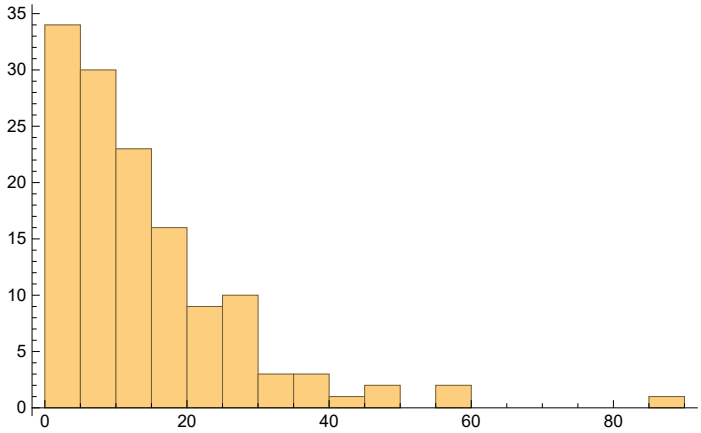
```
(* MData small bin count *)
Histogram[MData, IntegerPart[Length[MData] / 10]]
```




```
(* VData, large bin count *)
Histogram[VData, IntegerPart[Length[VData] / 2]]
```

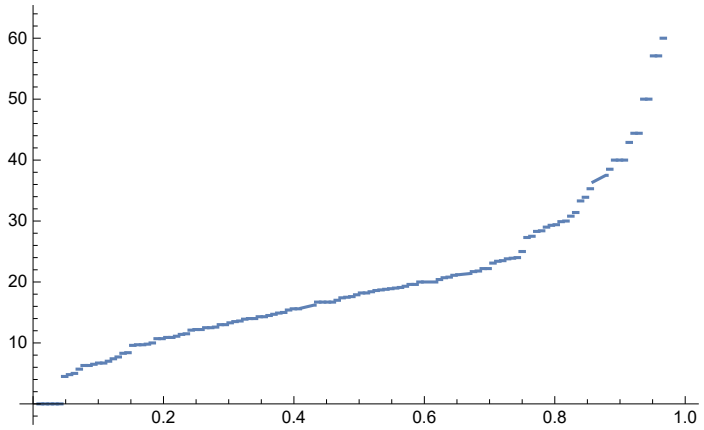



```
(* VData, small bin count*)
Histogram[VData, IntegerPart[Length[VData] / 10]]
```

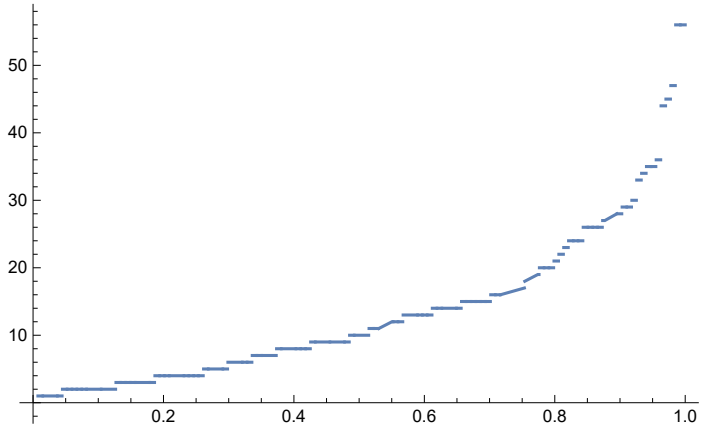


In[51]:=
```
(* Produce plots of quantile functions, moment functions, and CDFs *)

(* Define functions for moments and CDF *)

(* Sum elements raised to the kth power, divide by length *)
moment[x_, k_] := N[Sum[x[[i]]^k, {i, 1, Length[x]}] / Length[x]];
```

In[52]:=
```
cdf[x_, xi_] := N[Sum[If[x[[i]] <= xi, 1, 0], {i, 1, Length[x]}]];
(* Count that an element is less than or equal to a given element *)

(* Plot the quantile functions *)
(* MData *)
Plot[quantile[MData, i], {i, 0, 1}]
```



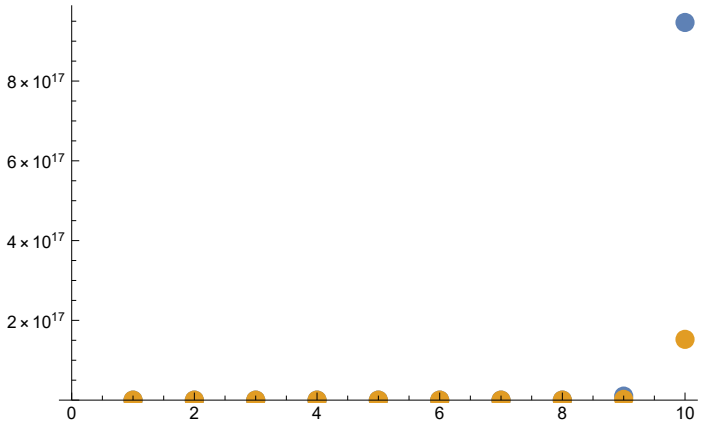

```
(* VData *)
Plot[quantile[VData, i], {i, 0, 1}]
```



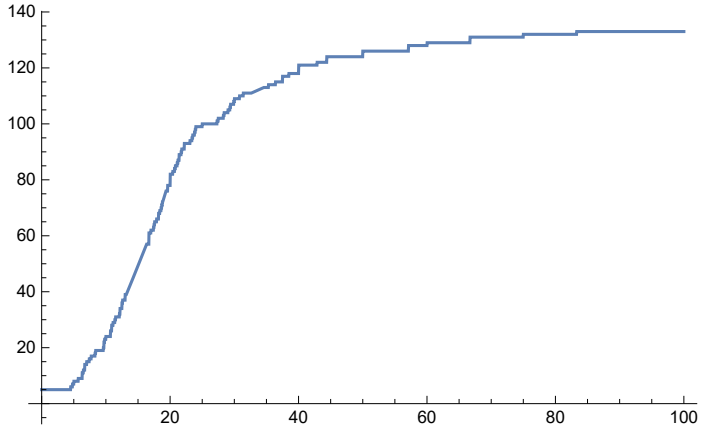
In[53]:=
```
(* Plot moment functions *)
(* Get the first 10 moments for MData and VData in a Table *)
momMData  = Table[moment[MData, i], {i, 1, 10}];
```

In[54]:=
```
momVData = Table[moment[VData, i], {i, 1, 10}];
```

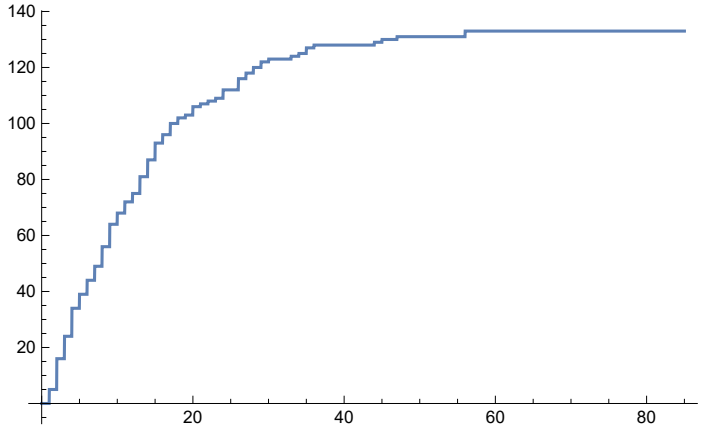
```
(* Plot the two moments *)
ListPlot[{momMData, momVData}, PlotStyle → PointSize[.03], PlotRange → {0, 9.9 * 10^17}]
(* Blue is MData, Orange is VData*)
```



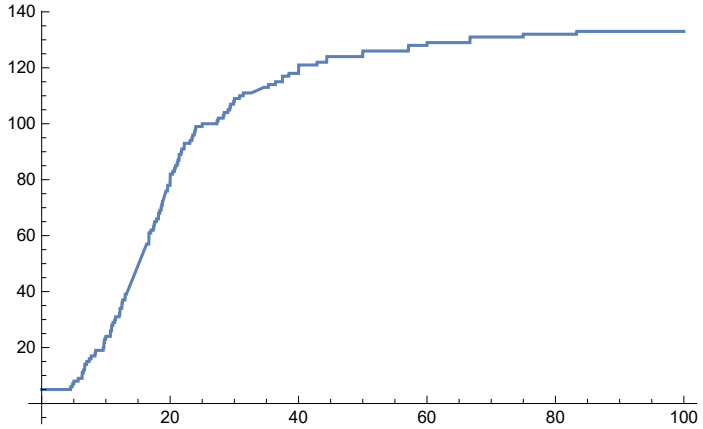

```
(* Plot CDF functions *)
(* MData from range 0 to maximum element in the data set *)
Plot[cdf[MData, i], {i, 0, Max[MData]}]
```



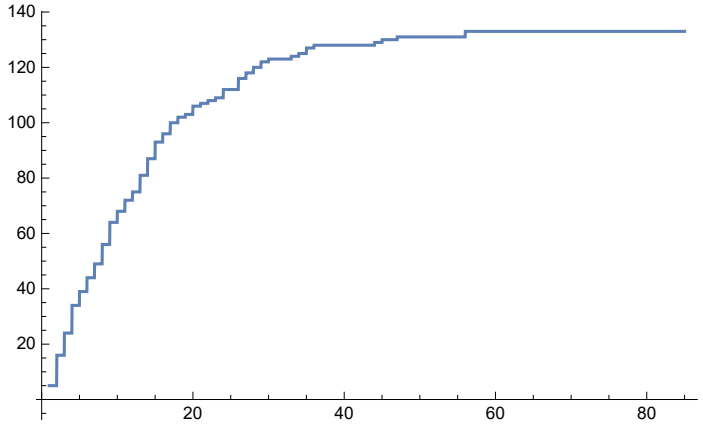

```
(* VData *)
(* VData from range 0 to maximum element in the data set *)
Plot[cdf[VData, i], {i, 0, Max[VData]}]
```



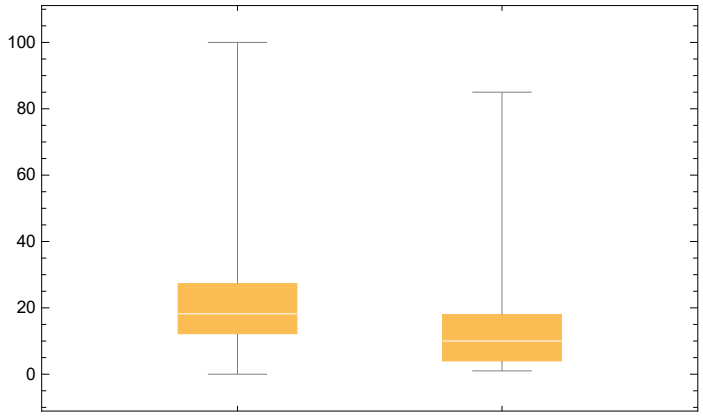
```
(* MData from range minimum element to maximum element in the data set *)
Plot[cdf[MData, i], {i, Min[MData], Max[MData]}]
```



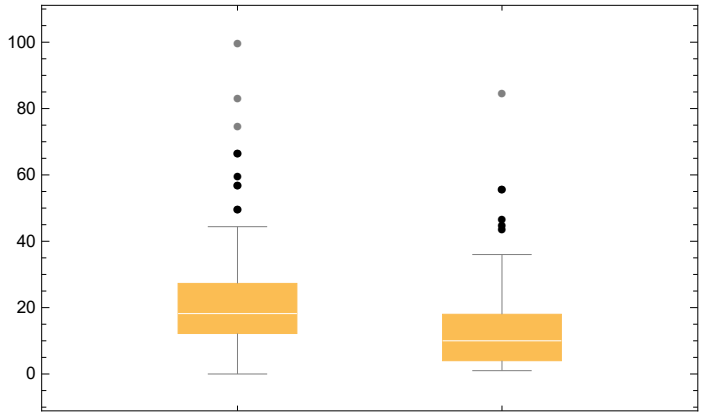

```
(* VData from range minimum element to maximum element in the data set *)
Plot[cdf[VData, i], {i, Min[VData], Max[VData]}]
```



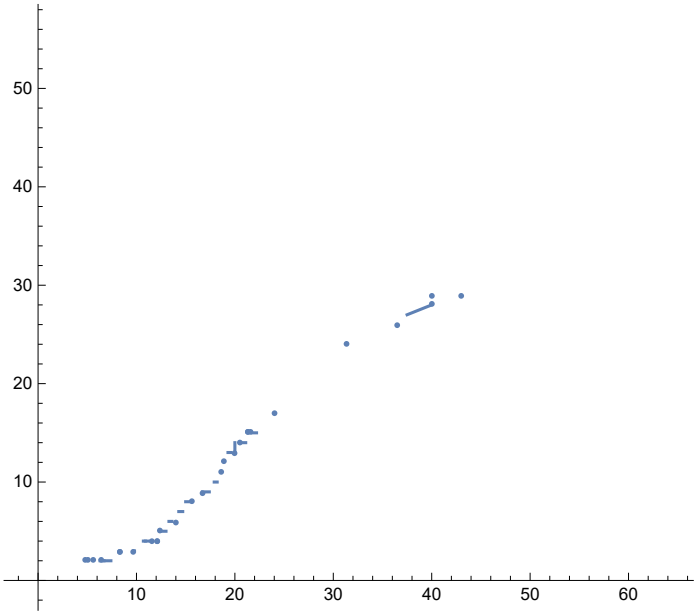
```
(* Although the instructions do not say to compare
 any box and whisker and QQ plots for this hospital data
 against other sets of data, I will product them anyways. QQ
 will be MData(x axis) against VDaya (y axis) *)

(* Box an whisker plots *)
(* A box and whisker plot takes a min, q1, q2 (median), q3, and max *)

(* Box and whisker, MData and VData, outliers not shown *)
BoxWhiskerChart[{MData, VData}]
```




```
(* Box and whisker, MData and VData, outliers shown *)
BoxWhiskerChart[{MData, VData}, "Outliers"]
```



```
(* Parametric Plots (QQ Plots), MData on x axis, VData on y axis *)
ParametricPlot[{quantile[MData, i], quantile[VData, i]}, {i, 0, 1}]
```



In[55]:=
```
(* If the two sets come from a population with the same distribution,
the points should fall approximately
 along a 45 degree reference line. As we can see,
the 2 batches do not appear to have come from populations with
 a common distribution, as they do not fit along a straight line. However,
I would consider it closer to a straight line
 than, say, a quadratic. So it suggests that the data
 sets came from population of fairly equal distributions. *)

(* resistor.nb data *)
(* The data represents a listing of the resistances
 (in ohms) of 200 resistors which are all rated at 10 kiloohms. *)
resistors = {9.97910927, 9.833997401, 10.48797923, 9.778286587, 10.4127049, 9.729651074,
   10.34005333, 9.894176108, 10.07983211, 9.933230947, 9.977783398, 10.13141411,
   10.1266421, 9.37852757, 10.26785423, 9.907086669, 9.744503691, 9.971603949,
   9.693939764, 9.620137112, 12.28072506, 10.0580338, 10.33764317, 9.757096213,
   9.593230848, 9.713741738, 9.432574293, 9.62099431, 9.802732952, 9.971484578,
   10.22548428, 10.3352728, 9.989841592, 10.29860424, 9.52298034, 10.08499861,
   9.394148142, 9.944944954, 10.21438162, 10.36193691, 10.02987499, 9.603449021,
   9.742946181, 9.875414084, 10.05078967, 10.12314509, 10.15281111, 5.870566193,
   9.484863417, 9.973958404, 9.94911044, 9.374762262, 9.788310356, 10.06500849,
   9.77439594, 10.03864565, 10.32397119, 9.916142963, 9.967350072, 10.09860352,
   9.987682395, 10.15563395, 9.537918791, 9.945042157, 10.02686399, 9.74540807,
   10.26915708, 9.696347652, 10.13930795, 9.51572712, 9.367227099, 9.831637831,
   10.1807235, 9.88921993, 9.923452458, 9.944225885, 9.779727284, 10.26538836, 10.2298635,
   10.2461264, 9.694717951, 9.771545526, 9.679096242, 10.15118993, 10.25894345,
   9.613968464, 10.14607857, 10.3809408, 10.00425765, 10.30422606, 9.938641588,
   10.14989447, 9.62901378, 6.613345698, 10.48706974, 10.10426569, 10.15476425,
   9.839152246, 9.74229305, 9.712882265, 10.09355753, 9.655283966, 10.01073951,
   10.23032052, 9.896222755, 9.646005983, 10.22741355, 9.916736976, 9.853518852,
   9.797304974, 9.542975581, 9.582644329, 10.06420074, 10.1110437, 9.09833499,
   9.694181349, 10.0837185, 9.990310834, 9.680224016, 9.544769559, 10.12220661,
   10.35625939, 9.68922915, 9.816272486, 9.838797828, 9.787675983, 10.01512384,
   9.672549018, 9.166747182, 9.839861368, 10.0490497, 9.9589975, 9.707653239, 9.642065029,
   10.14670044, 9.704657023, 9.851454583, 9.92931813, 10.05903936, 9.749898131,
   10.12904658, 9.776733909, 9.956306817, 10.10913774, 9.25291271, 9.823820724,
   9.581313056, 9.84027462, 9.738894951, 9.923279654, 9.815685862, 9.754906605,
   10.19531748, 9.718578829, 9.830784043, 9.860661512, 9.665515781, 9.956836598,
   10.06308718, 9.401201273, 10.10992616, 9.738494773, 9.991823154, 9.877411846,
   10.23755441, 10.04556889, 9.978626954, 10.06519891, 9.774786454, 10.26202664,
   10.10298671, 9.558598995, 9.352852535, 9.611078544, 9.807194024, 9.684415081,
   10.17326848, 9.683191811, 10.03918111, 9.891267714, 9.707087079, 9.68933829,
   10.10867702, 9.770431111, 9.697278747, 10.15024178, 10.17638293, 9.676198933,
   9.765484028, 9.952918381, 10.15444308, 10.03372073, 9.607316647, 9.856609145,
   9.805244863, 9.728007162, 9.951510938, 10.03217857, 10.19504918, 10.23059564};
```

In[56]:=
```
(* Find the mean, median, q1 and q3, and variance *)
(* functions were defined above for the hospital data *)
resistorMean = mean[resistors]
```

Out[56]= 9.87989

In[57]:=
```
resistorMedian = median[resistors]
```

Out[57]= 9.91674

In[58]:=
```
resistorQ1 = quantile[resistors, .25]
```

Out[58]= 9.71288

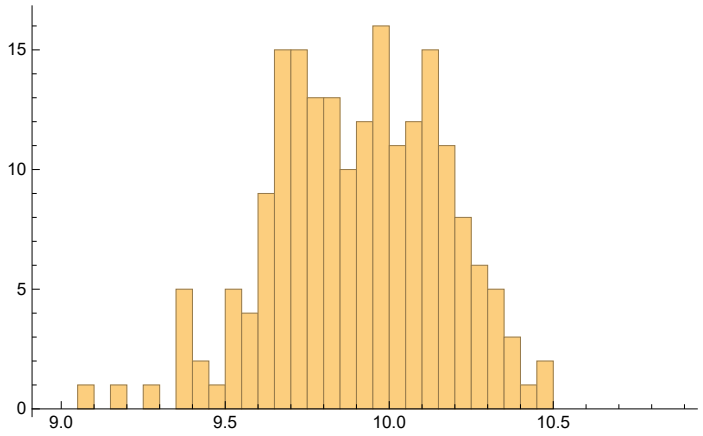
In[59]:=
```
resistorQ3 = quantile[resistors, .75]
```

Out[59]= 10.1043

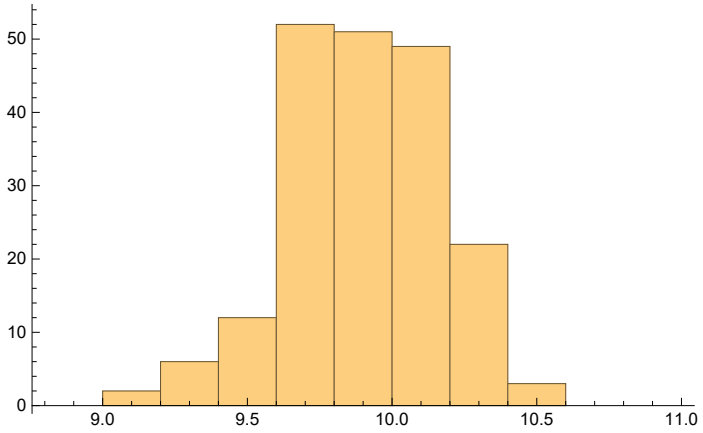
In[60]:=
```
resistorVariance = variance[resistors]
```

Out[60]= 0.229448

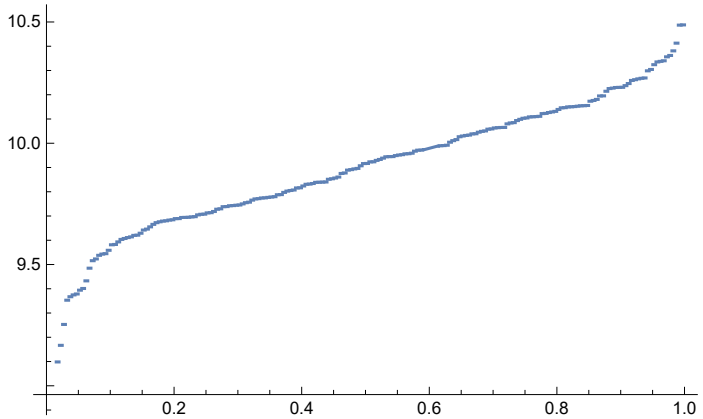
```
(* Histograms using two difference bin sizes *)
(* I will use a bin size Length/2 for a very large bin count,
and Length/10 for a smaller bin count*)

(* Large bin count*)
Histogram[resistors, IntegerPart[Length[resistors] / 2]]
```



```
(* Small bin count *)
Histogram[resistors, IntegerPart[Length[resistors] / 10]]
```

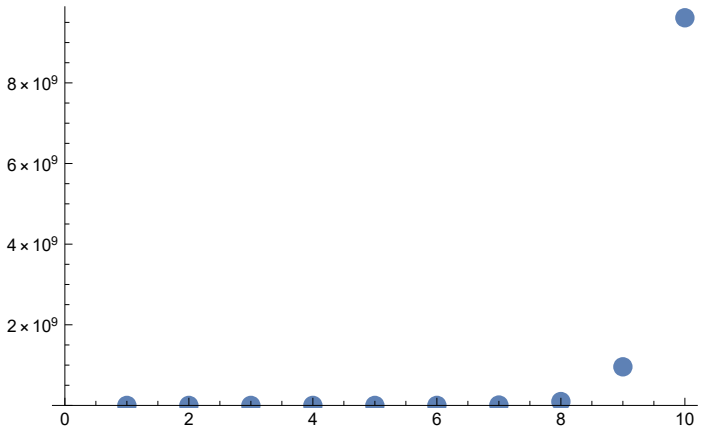



```
(* Produce plots of quantile functions, moment functions, and CDFS *)
(* functions were defined above for the hospital data *)

(* Plot the quantile functions *)
Plot[quantile[resistors, i], {i, 0, 1}]
```

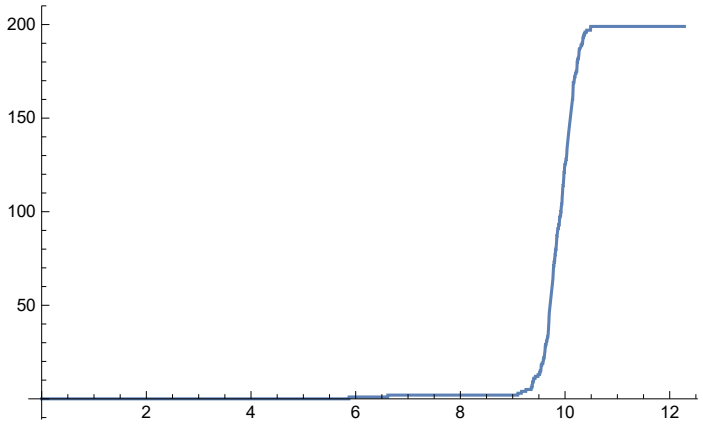


In[61]:=
```
(* Plot the moment functions *)
(* Get the first 10 moments for the resistor data in a table *)
momResistorData = Table[moment[resistors, i], {i, 1, 10}];

(* Plot the moment *)
ListPlot[momResistorData, PlotStyle → PointSize[.03], PlotRange → {0, 9.9 * 10^9}]
```

Out[62]=

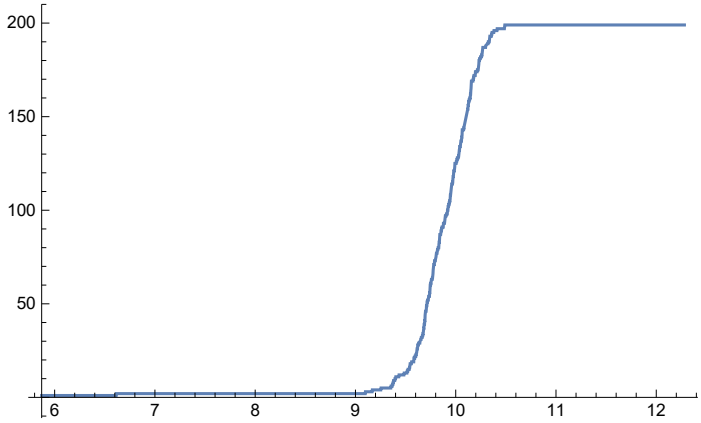



```
(* Plot the CDF functions *)
(* CDF for range 0 to the maximum element in the data set *)
Plot[cdf[resistors, i], {i, 0, Max[resistors]}]
```

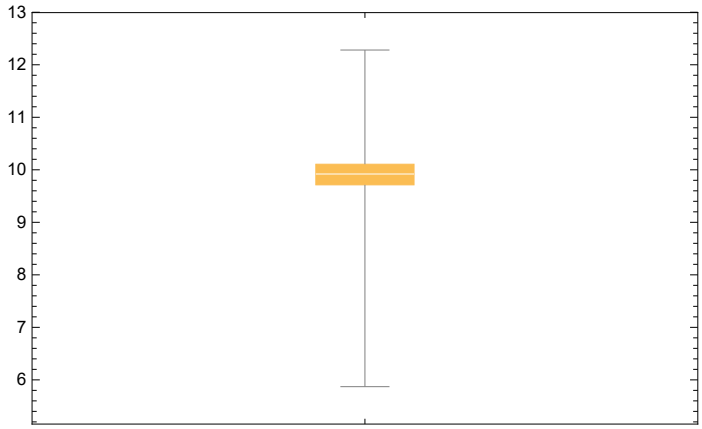



```
(* CDF for range minimum element to maximum element in the data set *)
```

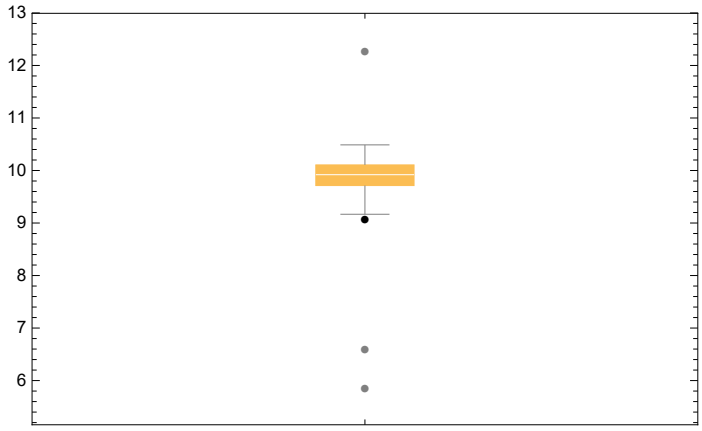
```
Plot[cdf[resistors, i], {i, Min[resistors], Max[resistors]}]
```




```
(* Box an whisker plots *)
(* A box and whisker plot takes a min, q1, q2 (median), q3, and max *)

(* Box and whisker, resistor data, outliers not shown *)
BoxWhiskerChart[resistors]
```




```
(* Box and whisker, resistor data, outliers shown *)

BoxWhiskerChart[resistors, "Outliers"]
```



In[63]:=
```
(* Parametric plot comparison with drips and resistor data will be later *)

(* drips-pcw.nb data *)

(* The set of data provided below represents the time intervals
 (in seconds) between consecutive water drips from a nozzle. *)
Drp = {0.18228360, 0.18623970, 0.13423350, 0.10354810, 0.15513900, 0.23274050, 0.20233310,
   0.12894790, 0.18684980, 0.22657810, 0.26112470, 0.19178580, 0.13767700, 0.14837620,
   0.22277630, 0.16055710, 0.13788350, 0.09521610, 0.24578210, 0.17383130, 0.25812850,
   0.18938570, 0.25420280, 0.22464200, 0.26155470, 0.10953020, 0.22034160, 0.10145990,
   0.19693630, 0.12816710, 0.13596500, 0.10053220, 0.25587460, 0.14042210, 0.25563470,
   0.13526800, 0.25192090, 0.25315870, 0.25650260, 0.07200480, 0.22221230, 0.10651970,
   0.23083320, 0.14309380, 0.12039200, 0.07573480, 0.28357910, 0.13409110, 0.25356630,
   0.13608820, 0.15960660, 0.20411740, 0.25443260, 0.10519020, 0.22459810, 0.10852200,
   0.23142460, 0.18760390, 0.14815410, 0.13764410, 0.22559380, 0.14294190, 0.21218650,
   0.12436990, 0.17052370, 0.26370070, 0.22443330, 0.13576010, 0.22104630, 0.14850460,
   0.20739950, 0.23946950, 0.09949950, 0.13500100, 0.22572200, 0.13560920, 0.26428920,
   0.16429840, 0.13426580, 0.21094650, 0.22839840, 0.13847070, 0.22873960, 0.09668840,
   0.21728640, 0.21903800, 0.14815920, 0.13720790, 0.22385060, 0.13849930, 0.25677440,
   0.14003500, 0.13574610, 0.21009840, 0.22764460, 0.13569010, 0.23362570, 0.10627020,
   0.20359420, 0.22037170, 0.14801240, 0.14313470, 0.22738740, 0.13893190, 0.26134080,
   0.15902480, 0.13821110, 0.25994540, 0.19984440, 0.13702970, 0.25571080, 0.10017000,
   0.24149470, 0.20942150, 0.09821050, 0.17724780, 0.22568230, 0.09987190, 0.25377520,
   0.13363160, 0.14934250, 0.25567830, 0.16646210, 0.13653430, 0.26081120, 0.11141880,
   0.22362140, 0.26196300, 0.25485620, 0.15926790, 0.22449670, 0.11083920, 0.22594470,
   0.14749760, 0.09495230, 0.13961280, 0.22764470, 0.14142230, 0.24807480, 0.13791740,
   0.24859080, 0.25935050, 0.26436380, 0.14695130, 0.23698890, 0.10994630, 0.24675460,
   0.19712410, 0.10504020, 0.11948800, 0.22092570, 0.13606610, 0.22484940, 0.12570890,
   0.16141710, 0.26257510, 0.14825630, 0.13100610, 0.20236200, 0.13893380, 0.26028230,
   0.23129920, 0.09723730, 0.24600470, 0.23302040, 0.14716670, 0.18991060, 0.09395180,
   0.14256570, 0.25940860, 0.13184190, 0.14779980, 0.26544610, 0.17659160, 0.25689390,
   0.22272640, 0.10381410, 0.22347600, 0.25889310, 0.09551590, 0.21917120, 0.07637890,
   0.14582530, 0.27091080, 0.13431580, 0.11580080, 0.26142020, 0.13817780, 0.25436520,
   0.19498570, 0.09923730, 0.22391340, 0.22166680, 0.09870400, 0.22214220, 0.14858030,
   0.17792490, 0.24428880, 0.14327470, 0.15278630, 0.25635740, 0.14497800, 0.18314970,
   0.23344810, 0.11068690, 0.26540170, 0.14857800, 0.10705320, 0.22829050, 0.18691300,
   0.13606300, 0.25154320, 0.10921350, 0.22934640, 0.22450430, 0.13692630, 0.23764460,
   0.18372960, 0.10921790, 0.19986190, 0.15007520, 0.13633160, 0.22506690, 0.11534350,
   0.17324380, 0.21237050, 0.14873530, 0.21643160, 0.22657440, 0.13686960, 0.23828540,
   0.318174320, 0.14837860, 0.21115580, 0.16123000, 0.13529380, 0.24016030, 0.12552070,
   0.15950540, 0.21113940, 0.14725600, 0.21805820, 0.22875220, 0.12800870, 0.20448610,
   0.21948410, 0.14692220, 0.24746870, 0.14248680, 0.13871530, 0.23005070, 0.16313400,
   0.13891430, 0.26708190, 0.11291010, 0.25614080, 0.24051200, 0.24548400, 0.23622310,
   0.23961130, 0.09302900, 0.21150500, 0.11970740, 0.12498680, 0.11555500, 0.13855050,
   0.12045270, 0.26194240, 0.14797380, 0.16864190, 0.22269730, 0.26032830, 0.25778280,
   0.22254430, 0.10020850, 0.26106870, 0.15855650, 0.14371780, 0.11136020, 0.12119620,
   0.14110830, 0.25684420, 0.09828760, 0.23301490, 0.20504970, 0.18349520, 0.21975910,
   0.24160800, 0.10200310, 0.21208000, 0.14020810, 0.15974330, 0.18873760, 0.14236020,
   0.12082240, 0.22755640, 0.14818550, 0.22244030, 0.17957990, 0.24217980, 0.20160190,
   0.22291860, 0.14020770, 0.25848130, 0.13665170, 0.17852080, 0.11437730, 0.15921670,
   0.13596970, 0.26868740, 0.10135420, 0.22488520, 0.20168690, 0.20958320, 0.22973200,
```

```
0.22528830, 0.10985200, 0.25246890, 0.14021050, 0.17207190, 0.14624990, 0.14421290,
0.13553570, 0.19721490, 0.11316860, 0.22365350, 0.19277550, 0.22177710, 0.22281810,
0.22673220, 0.18224740, 0.22948630, 0.13615970, 0.14343280, 0.13595380, 0.13332310,
0.13758030, 0.17235540, 0.13477980, 0.22750740, 0.27205230, 0.22723570, 0.22356740,
0.26976620, 0.16408940, 0.25041400, 0.13610470, 0.09729680, 0.10675350, 0.14689520,
0.10447090, 0.19264200, 0.11169570, 0.25101950, 0.24382670, 0.25675530, 0.23134120,
0.19872520, 0.15405980, 0.26169180, 0.12361070, 0.11594210, 0.13849870, 0.12892750,
0.15996210, 0.18149290, 0.10359830, 0.20306880, 0.22509590, 0.22510400, 0.22840200,
0.24895140, 0.19837270, 0.25859040, 0.18603360, 0.13769650, 0.13587430, 0.13107450,
0.12671680, 0.17744900, 0.10807120, 0.15458710, 0.22524160, 0.22834810, 0.26317470,
0.25623170, 0.14897020, 0.25305620, 0.19255750, 0.13608600, 0.13614730, 0.11293430,
0.13007420, 0.16590880, 0.10560190, 0.19604770, 0.22842020, 0.22890840, 0.19967150,
0.17581550, 0.21840620, 0.26584180, 0.18360270, 0.13641250, 0.15187060, 0.17108980};
```

In[64]:=
```
(* Find the mean, median, q1, q3, and variance *)
(* functions were defined above for the hospital data *)
dripMean = mean[Drp]
```
Out[64]= 0.182304

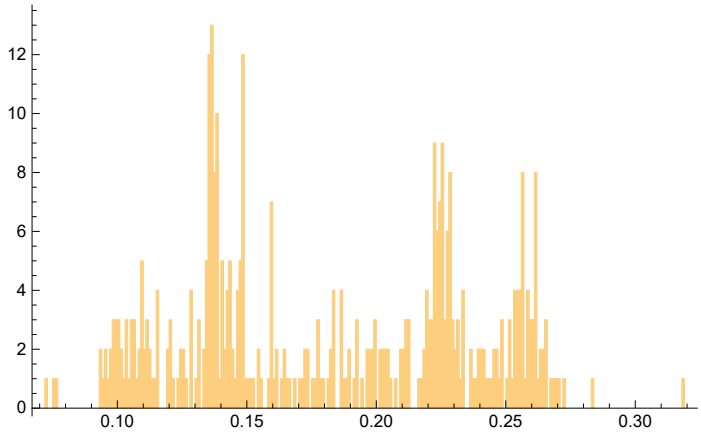
In[65]:=
```
dripMedian = median[Drp]
```
Out[65]= 0.181493

In[66]:=
```
dripQ1 = quantile[Drp, .25]
```
Out[66]= 0.136332

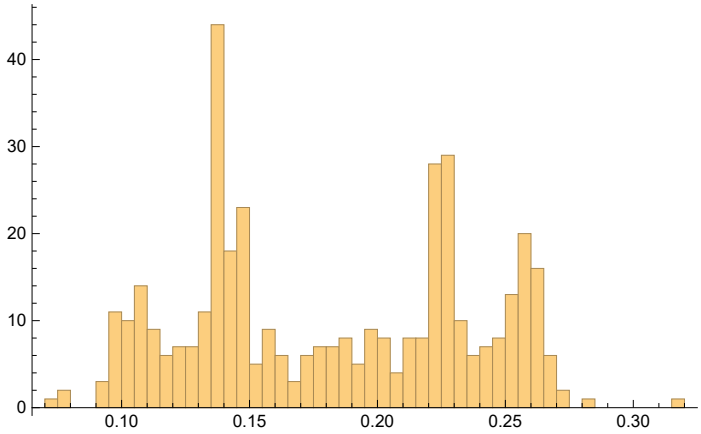
In[67]:=
```
dripQ3 = quantile[Drp, .75]
```
Out[67]= 0.227645

In[68]:=
```
dripVariance = variance[Drp]
```
Out[68]= 0.00291283

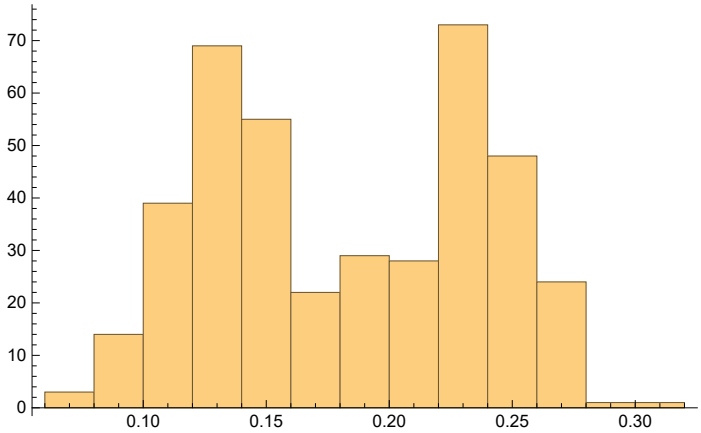
```
(* Histograms using two different bin sizes *)
(* I will use a bin size of Length/2 for a very large bin count,
and Length/10 for a smaller bin count *)

(* Large bin count *)
Histogram[Drp, IntegerPart[Length[Drp] / 2]]
```



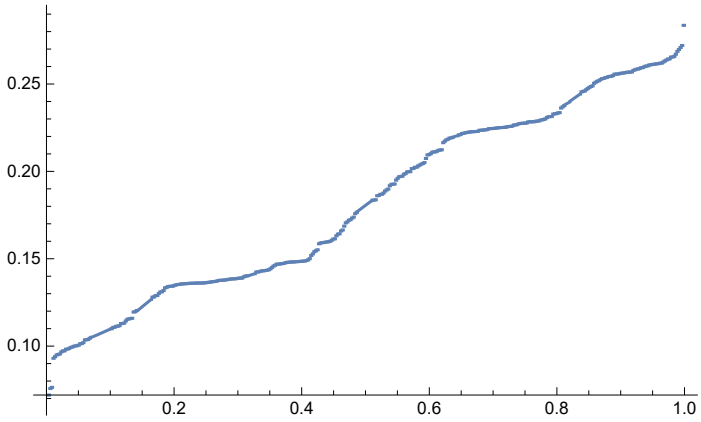

```
(* Small bin count *)
Histogram[Drp, IntegerPart[Length[Drp] / 10]]
```



```
(* As there is 200 data elements,
Igit  will use Length/30 for an even smaller bin count,
as it is appropriate for this data size, unlike the other data set sizes *)

Histogram[Drp, IntegerPart[Length[Drp] / 30]]
```

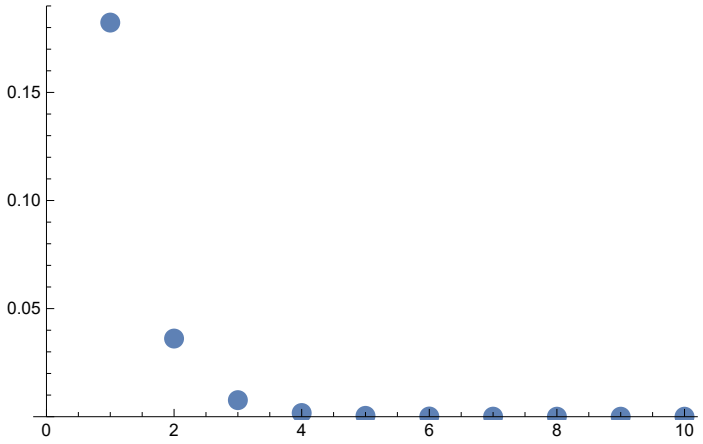



```
(* Product plots of quantile functions, moment functions, and CDFS *)
(* functions were defined above for the hospital data *)

(*Plot the quantile functions *)
Plot[quantile[Drp, i], {i, 0, 1}]
```

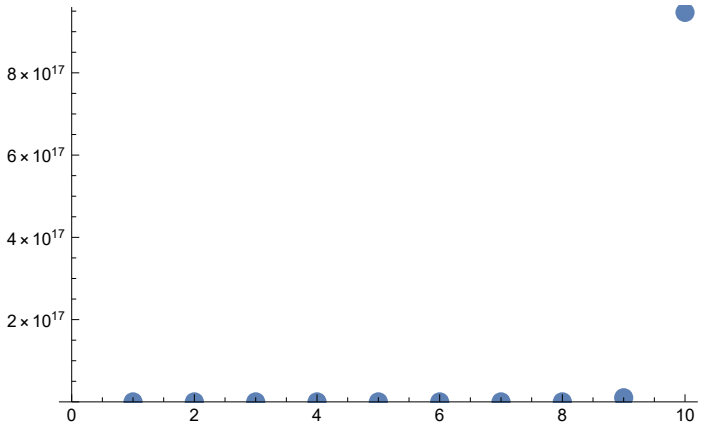


In[69]:=
```
(* Plot the moment functions *)
(* Get the first 10 moments for the resistor data in a table *)
momDripData = Table[moment[Drp, i], {i, 1, 10}];
```

```
(* Note that since the drip data's elements are all less than 1, raising these values
  to a power 1-10 be decreasing this value,
as opposed to the other data sets where raising their
  values to a power 1-10 will increase the value *)
ListPlot[momDripData, PlotStyle → PointSize[.03], PlotRange → {0, .19}]
```

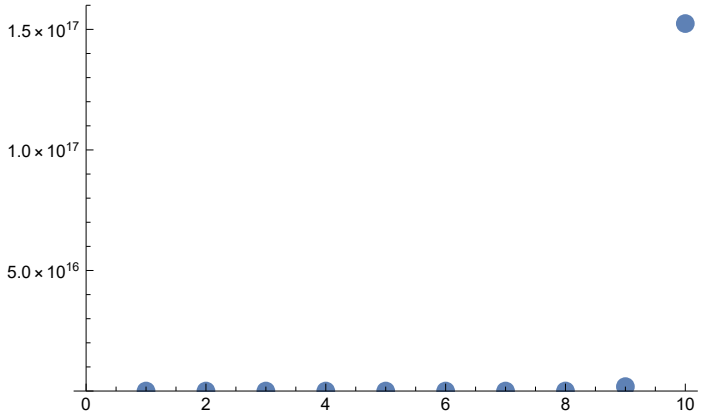



```
(* In comparison, here is what the MData and VData moments looked like *)
(* MData moment *)
ListPlot[momMData, PlotStyle → PointSize[.03], PlotRange → {0, 9.6 * 10^17}]
```

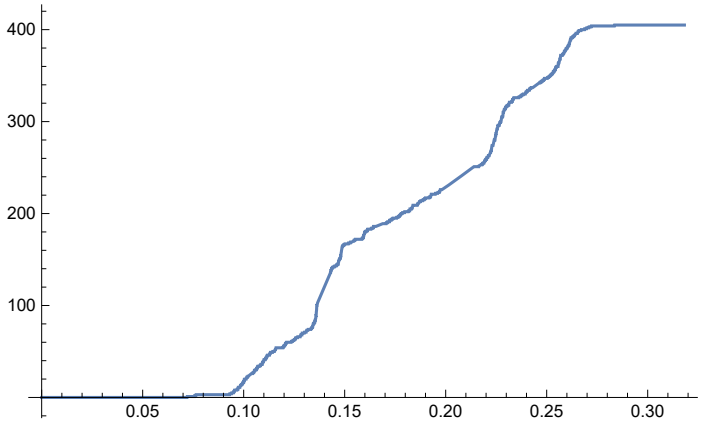



```
(* VData moment *)
```

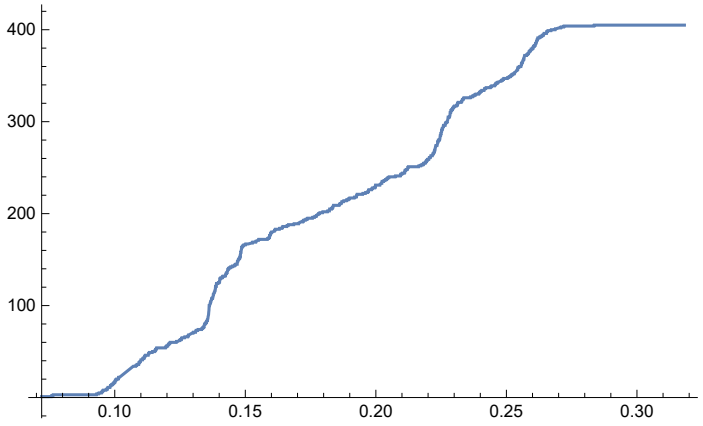
```
ListPlot[momVData, PlotStyle → PointSize[.03], PlotRange → {0, 1.6 * 10^17}]
```



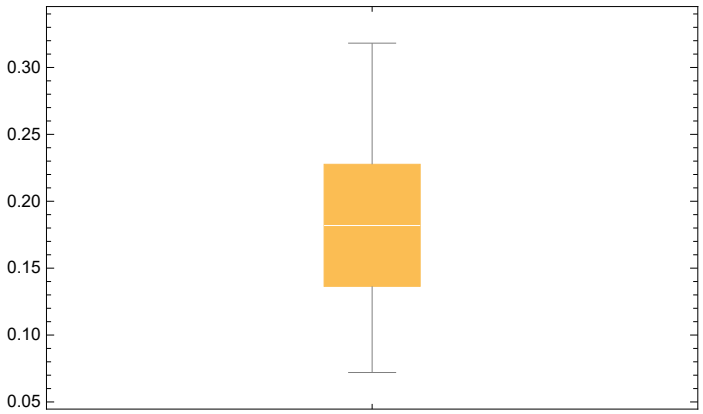

```
(* Plot the CDF functions *)
(* CDf for range 0 to the maximum element in the data set *)
Plot[cdf[Drp, i], {i, 0, Max[Drp]}]
```



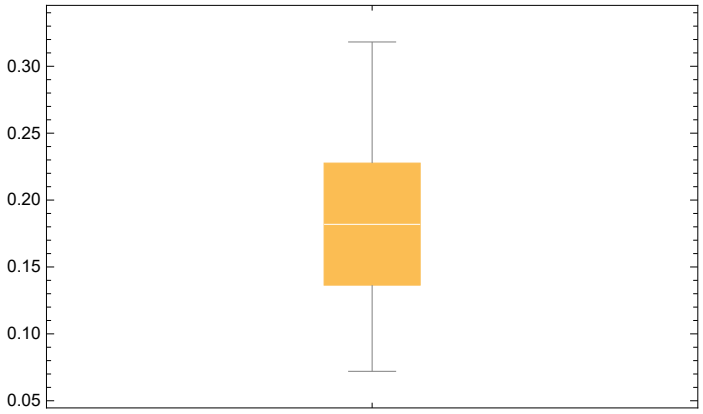

```
(* CDf for range minimum element to maximum element in the data set *)
Plot[cdf[Drp, i], {i, Min[Drp], Max[Drp]}]
```



```
(* Box and Whisker plots *)
(* A box and whisker plot takes a min, q1, q2 (median), q3, and max *)

(* Box and whisker, drip, data, outliers not shown *)
BoxWhiskerChart[Drp]
```




```
(* Box and whisker, drip, data, outliers shown *)
BoxWhiskerChart[Drp, "Outliers"]
```

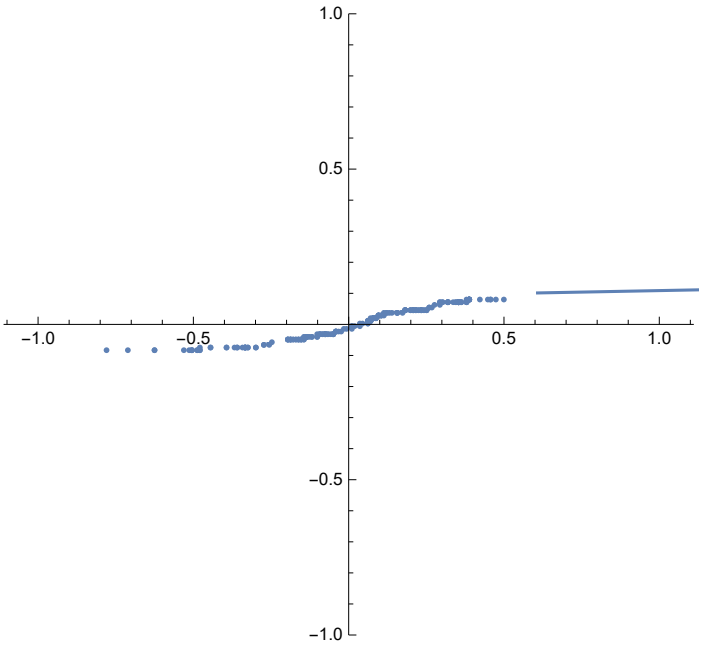


In[70]:=
```
(* QQ plot compairson for drip and resistor data *)
(* Data must be centered (subtract the means) for both data sets *)

(* "Centering simply means subtracting a constant from every value of a variable.
  What it does is redefine the 0 point for that predictor to be whatever
  value you subtracted.  It shifts the scale over, but retains the units." *)

(* Centering function *)
(* Subtract the mean off of every value in the data set *)
Centering[x_, meanValue_] :=
  (s = x;  For[i = 1, i <= Length[s], i++, s[[i]] = s[[i]] - meanValue]; Return[s]);
```

In[71]:=
```
(* Center the drip and resistor data *)
DripCentered = Centering[Drp, dripMean];
```

In[72]:=
```
ResistorCentered = Centering[resistors, resistorMean];
```

```
(* Parametric Plots (QQ Plots), resistor data on x axis, drip data on y axis *)
ParametricPlot[{quantile[ResistorCentered, i], quantile[DripCentered, i]},
 {i, 0, 1}, PlotRange → {-1, 1}]
```




```
(* If the two sets come from a population with the same distribution,
the points should fall approximately
 along a 45 degree reference line. As we can see,
the 2 batches do not appear to have come from populations with
 a common distribution, as they do not fit along a straight line. *)
(* The QQ plot looks logarithmic,
meaning that it dos not look like a straight line. This suggests that the two
 data sets came from populations with different distributions. *)
```

```
(* Problem 2 *)
(* Calculate the correlation coefficient between the volume V and mortality M
 of the heart transplants based on the data in hospheart.nb *)

(* hospheart.np data *)
(* {M,V} M = one year mortality rate,
percentage of patients that died within one year of the
 transplant operation,
V = average annual number of transplants at that center during the same 4 years *)
heart = {{17.9, 27}, {23.1, 4}, {40, 3}, {6.5, 35}, {14.9, 17}, {12.5, 4}, {15.7, 45},
   {9.8, 28}, {24, 6}, {5.0, 10}, {15.4, 13}, {4.8, 7}, {0, 1}, {19.1, 47}, {4.5, 6},
   {15, 56}, {12.5, 4}, {33.9, 8}, {10.7, 9}, {13, 14}, {28.3, 12}, {57.1, 2}, {6.3, 4},
   {10, 3}, {8.3, 12}, {17.5, 10}, {20, 3}, {29.3, 10}, {21.4, 7}, {27.3, 8}, {13.6, 6},
   {21.8, 30}, {36.4, 3}, {18.2, 11}, {33.3, 2}, {20, 4}, {38.5, 7}, {20.8, 18}, {12.2, 19},
   {22.2, 18}, {29, 8}, {0, 9}, {5.7, 9}, {50, 2}, {21.7, 15}, {66.7, 4}, {29.4, 17},
   {12.1, 27}, {10.7, 14}, {6.3, 4}, {16.2, 9}, {21.1, 5}, {17.4, 33}, {23.9, 17},
   {42.9, 2}, {40, 2}, {6.7, 15}, {44.4, 3}, {18.7, 34}, {14.7, 24}, {7.4, 7}, {12.6, 24},
   {9.7, 26}, {44.4, 2}, {16.7, 6}, {15.8, 14}, {83.3, 2}, {10.9, 22}, {13.3, 5},
   {11.1, 5}, {75, 2}, {19, 20}, {14, 13}, {60, 1}, {21.2, 8}, {9.7, 8}, {50, 2}, {25, 14},
   {18.6, 15}, {0.0, 1}, {35.3, 9}, {23.5, 85}, {15.6, 11}, {37.5, 2}, {14.3, 28},
   {14.3, 4}, {16.7, 6}, {20.0, 15}, {13.0, 17}, {9.6, 26}, {66.7, 3}, {30.8, 3},
   {14.0, 13}, {27.5, 10}, {37.5, 8}, {18.9, 13}, {0.0, 4}, {12.2, 44}, {57.1, 4},
   {21.4, 35}, {23.4, 16}, {10.9, 12}, {15.6, 8}, {16.7, 2}, {13.9, 9}, {18.2, 11},
   {11.5, 26}, {18.4, 13}, {16.7, 3}, {20.4, 14}, {40.0, 5}, {20.7, 56}, {19.6, 13},
   {13.5, 9}, {29.9, 36}, {8.4, 21}, {28.4, 24}, {7.7, 23}, {19.3, 29}, {0.0, 1},
   {22.2, 20}, {30.0, 5}, {7.0, 11}, {23.8, 7}, {18.8, 29}, {14.5, 16}, {17.0, 16},
   {20.0, 15}, {6.7, 15}, {11.4, 20}, {100.0, 1}, {31.4, 9}, {17.6, 26}, {19.6, 14}};

(* Split this M and V data into separate
 lists via Transpose[] in order to parse through *)
heartTranspose = Transpose[heart];
MData = heartTranspose[[1]];
VData = heartTranspose[[2]];

(* To calculate the correlation coefficient,
we need to define functions to find the means of a data set *)
mean[x_] := Sum[x[[i]], {i, 1, Length[x]}] / Length[x];
(* Sum elements, divide by length *)

meanM = mean[MData]
```

Out[15]= 21.9045

```
In[16]:= meanV = N[mean[VData]]
```

Out[16]= 13.8657

In[17]:=
```
(* Create a function to sum (m-meanM)*(v-meanV),
where m and v are elemnts of M and V respectively. *)
differenceMeanSum[m_, mBAR_, v_, vBAR_] :=
 Sum[ ((m[[i]] - mBAR) * (v[[i]] - vBAR)), {i, 1, Length[m]}]
(* Sum the product of (elementInM - meanOfM) (elementInV - meanOfV) *)
```

In[18]:=
```
(* find the sum of the mean difference as noted above,
this is the numerator of our correlation coefficient equation *)
```

In[19]:=
```
MVdifferenceMeanSum = differenceMeanSum[MData, meanM, VData, meanV]
```

Out[19]= −7238.42

In[20]:=
```
(* The denominator of the correlation coefficient equation is
 the square root of: sum of (x-xBAR)^2 times sum of (y-yBAR)^2 *)
(* Create a function to Sum a data sets elemnts, by taking an element,
subtracting the mean from it, and squating the value *)
```

In[21]:=
```
squaredSum[a_, aMean_] := Sum[(a[[i]] - aMean)^2, {i, 1, Length[a]}];
```

In[22]:=
```
(* Now find the the squared difference sum for the M and V data *)
squareDifferenceM = squaredSum[MData, meanM]
```

Out[22]= 35 996.9

In[23]:=
```
squareDifferenceV = squaredSum[VData, meanV]
```

Out[23]= 22 305.6

In[24]:=
```
(* Take the root of the product of these sums,
and that is the denominator of the correlation coefficient equation *)
rootOfSums = Sqrt[squareDifferenceM * squareDifferenceV]
```

Out[24]= 28 336.1

In[25]:=
```
(* Now take the numerator and denominator,
find the decimal of that fraction and we have the correlation coefficient *)
coefficient = MVdifferenceMeanSum / rootOfSums
```

Out[25]= −0.255449

In[26]:=
```
(* This is negative, so as x increases,
y decreases. As is is close to zero, it is not a strong correlation *)

(* Now, we need to make a scatter plot of this data *)

(* We also need the line of best fit,
which will be calculated manually. Assuming the V data is y axis, M data is x axis*)
(* The numerator for the slope of our
 best fit line is the variable MVdifferenceMeanSum *)
(* The denominator for the slope of our best fit line is the sum of x -
 xMean squared *)

(* This denominator is as follows *)
denom = Sum[(MData[[i]] - meanM)^2, {i, 1, Length[MData]}];
slope = MVdifferenceMeanSum / denom
```

Out[27]= -0.201084

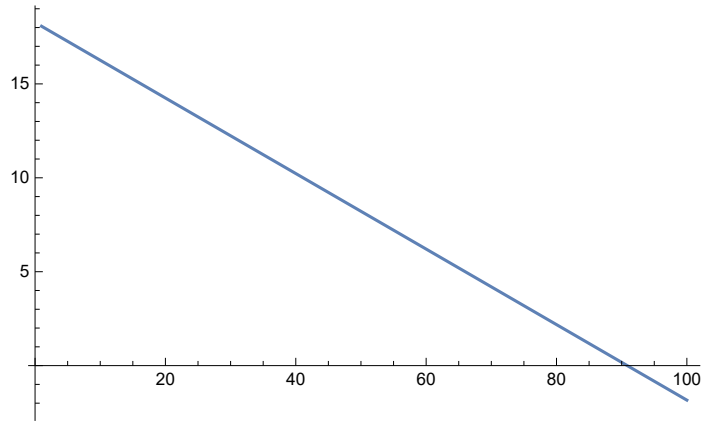
In[28]:=
```
(* The b value is calculated as the y mean, minus slope * x mean *)
b = meanV - slope * meanM
```

Out[28]= 18.2703

In[29]:=
```
(* The best fit equation is as follows *)
bestFit[x_] := slope * x + b;
```

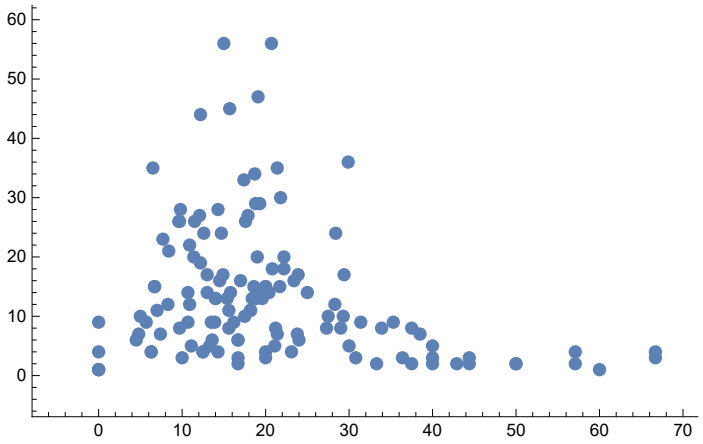
In[30]:=
```
(* Now, superimpose this regression line on the scatter plot *)
bestFitGraph = Plot[bestFit[x], {x, 1, 100}]
```

Out[30]=



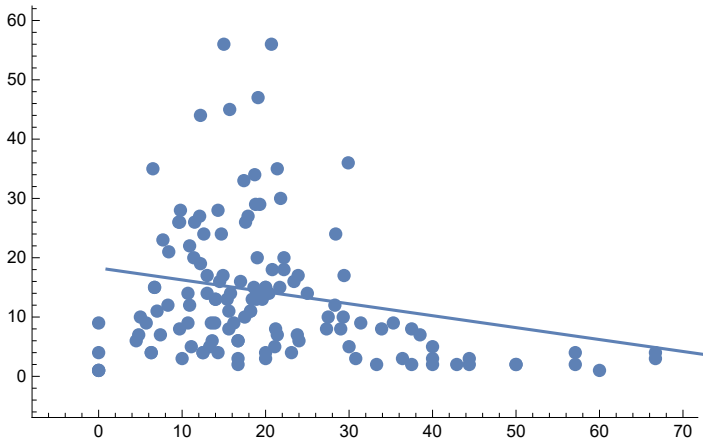
In[31]:=
```
scatterPlot = ListPlot[heart, PlotStyle → PointSize[.02],
  PlotRangePadding → Scaled[0.1], Axes → False, Frame → {True, True, False, False}]
(* Extra paramets on this scatter plot to best show the data *)
```

Out[31]=



In[32]:=
```
superimpose = Show[scatterPlot, bestFitGraph]
```

Out[32]=



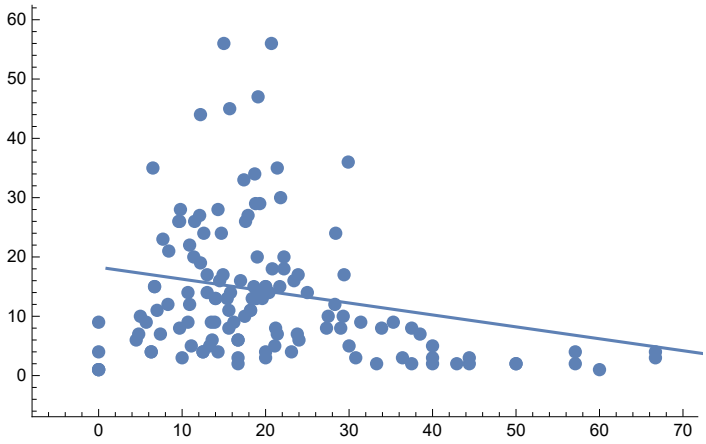
In[33]:=
```
(* To double check my calculations are correct,
I will let Mathematica generate a scatter plot and best fit line *)
```

In[34]:=
```
Fit[heart, {1, x}, {x}]
```

Out[34]= $18.2703 - 0.201084\,x$

In[35]:=
```
bestLine[x_] := 18.270320917115342` - 0.20108442453442926` x;
Show[ListPlot[heart, PlotStyle → PointSize[.02], PlotRangePadding → Scaled[0.1],
  Axes → False, Frame → {True, True, False, False}], Plot[bestLine[x], {x, 1, 100}]]
```

Out[36]=




```
(* Conclusions *)
(* This is negative, so as x increases,
y decreases. As is is close to zero, it is not a strong correlation *)
(* The one year mortality rate,
and average annual number of transplants at that center during the same 4 years, are
 not correlated strongly. The correlation is very weak *)
```

```
(* Problem 3 *)

(* Select long texts in three different languages. Calculate the entropy
 for each and compare them, draw your conclusions *)

(* I will first select an English text *)
```

In[73]:=
```
(* Sample text from The Adventures of Sherklock Holmes *)
text =
  "To Sherlock Holmes she is always the woman. I have seldom heard him mention her under
    any other name. In his eyes she eclipses and predominates the whole of her
    sex. It was not that he felt any emotion akin to love for Irene Adler. All
    emotions, and that one particularly, were abhorrent to his cold, precise but
    admirably balanced mind. He was, I take it, the most perfect reasoning and
    observing machine that the world has seen, but as a lover he would have placed
    himself in a false position. He never spoke of the softer passions, save with
    a gibe and a sneer. They were admirable things for the observer--excellent
    for drawing the veil from men's motives and actions. But for the trained
    reasoner to admit such intrusions into his own delicate and finely adjusted
    temperament was to introduce a distracting factor which might throw a doubt
    upon all his mental results. Grit in a sensitive instrument, or a crack in
    one of his own high-power lenses, would not be more disturbing than a strong
    emotion in a nature such as his. And yet there was but one woman to him,
    and that woman was the late Irene Adler, of dubious and questionable memory.

I had seen little of Holmes lately. My marriage had drifted us away from each other. My
    own complete happiness, and the home-centred interests which rise up around the
    man who first finds himself master of his own establishment, were sufficient
    to absorb all my attention, while Holmes, who loathed every form of society
    with his whole Bohemian soul, remained in our lodgings in Baker Street, buried
    among his old books, and alternating from week to week between cocaine and
    ambition, the drowsiness of the drug, and the fierce energy of his own keen
    nature. He was still, as ever, deeply attracted by the study of crime, and
    occupied his immense faculties and extraordinary powers of observation in
    following out those clues, and clearing up those mysteries which had been
    abandoned as hopeless by the official police. From time to time I heard some
    vague account of his doings: of his summons to Odessa in the case of the
    Trepoff murder, of his clearing up of the singular tragedy of the Atkinson
    brothers at Trincomalee, and finally of the mission which he had accomplished
    so delicately and successfully for the reigning family of Holland. Beyond
    these signs of his activity, however, which I merely shared with all the
    readers of the daily press, I knew little of my former friend and companion.

One night--it was on the twentieth of March, 1888--I was returning from a journey to a
    patient (for I had now returned to civil practice), when my way led me
    through Baker Street. As I passed the well-remembered door, which must always
    be associated in my mind with my wooing, and with the dark incidents of
    the Study in Scarlet, I was seized with a keen desire to see Holmes again,
    and to know how he was employing his extraordinary powers. His rooms were
    brilliantly lit, and, even as I looked up, I saw his tall, spare figure pass
    twice in a dark silhouette against the blind. He was pacing the room swiftly,
    eagerly, with his head sunk upon his chest and his hands clasped behind
    him. To me, who knew his every mood and habit, his attitude and manner told
    their own story. He was at work again. He had risen out of his drug-created
    dreams and was hot upon the scent of some new problem. I rang the bell
    and was shown up to the chamber which had formerly been in part my own.";
```

In[74]:=
```
lowerEnglishText = ToLowerCase[text]; (* Convert text to lowercase *)
```

In[75]:=
```
charcode = ToCharacterCode[lowerEnglishText];
(* Get a numerical character code for each letter in the string *)
```

In[76]:=
```
charentp = Table[{Length[Position[charcode, i]], FromCharacterCode[i]}, {i, 96, 122}]
 (* make a table with the counts for each character *)
```

Out[76]=
```
{{0, `}, {218, a}, {43, b}, {69, c}, {118, d}, {335, e}, {66, f}, {41, g}, {170, h},
 {211, i}, {2, j}, {21, k}, {110, l}, {93, m}, {196, n}, {208, o}, {42, p}, {1, q},
 {161, r}, {191, s}, {214, t}, {61, u}, {20, v}, {81, w}, {4, x}, {50, y}, {1, z}}
```

In[77]:=
```
charentp[[1]] = {Length[Position[charcode, 32]], "space"}
 (* Also, get the character count for spaces *)
```

Out[77]=
```
{611, space}
```

In[78]:=
```
TableForm[charentp] (* Put the counts and characters in a table to visualize easier *)
```

Out[78]//TableForm=

611	space
218	a
43	b
69	c
118	d
335	e
66	f
41	g
170	h
211	i
2	j
21	k
110	l
93	m
196	n
208	o
42	p
1	q
161	r
191	s
214	t
61	u
20	v
81	w
4	x
50	y
1	z

In[79]:=
```
sortcharentp = Reverse[Sort[charentp]]
 (* Sort by most common to least common frequencies *)
```

Out[79]=
```
{{611, space}, {335, e}, {218, a}, {214, t}, {211, i}, {208, o}, {196, n}, {191, s},
 {170, h}, {161, r}, {118, d}, {110, l}, {93, m}, {81, w}, {69, c}, {66, f}, {61, u},
 {50, y}, {43, b}, {42, p}, {41, g}, {21, k}, {20, v}, {4, x}, {2, j}, {1, z}, {1, q}}
```

In[80]:=
```
sumchars = Sum[sortcharentp[[i, 1]], {i, Length[sortcharentp]}]
 (* Get the total character count in the text *)
```

Out[80]=
```
3338
```

In[81]:=
```
charfreq = N[Table[{sortcharentp[[i, 1]] / sumchars, sortcharentp[[i, 2]]},
   {i, 1, Length[sortcharentp]}]] (* Get the frequency for each character in the text *)
```

Out[81]=
```
{{0.183044, space}, {0.100359, e}, {0.0653086, a}, {0.0641102, t},
 {0.0632115, i}, {0.0623128, o}, {0.0587178, n}, {0.0572199, s}, {0.0509287, h},
 {0.0482325, r}, {0.0353505, d}, {0.0329539, l}, {0.027861, m}, {0.024266, w},
 {0.0206711, c}, {0.0197723, f}, {0.0182744, u}, {0.014979, y}, {0.012882, b},
 {0.0125824, p}, {0.0122828, g}, {0.00629119, k}, {0.00599161, v},
 {0.00119832, x}, {0.000599161, j}, {0.000299581, z}, {0.000299581, q}}
```

In[82]:=
```
TableForm[charfreq] (* Put this frequency in a table to visualize easier *)
```

Out[82]//TableForm=

0.183044	space
0.100359	e
0.0653086	a
0.0641102	t
0.0632115	i
0.0623128	o
0.0587178	n
0.0572199	s
0.0509287	h
0.0482325	r
0.0353505	d
0.0329539	l
0.027861	m
0.024266	w
0.0206711	c
0.0197723	f
0.0182744	u
0.014979	y
0.012882	b
0.0125824	p
0.0122828	g
0.00629119	k
0.00599161	v
0.00119832	x
0.000599161	j
0.000299581	z
0.000299581	q

In[83]:=
```
entplang = 0;
(* Loop through the entire table, using the entropy formula,
calculate the entropy of the text given our frequencies *)
For[i = 1, i <= 27, i++,
            If[charfreq[[i, 1]] == 0, entplang,
                entplang = entplang - charfreq[[i, 1]] * Log[charfreq[[i, 1]]]
            ]
        ]
```

In[85]:=
```
(* This represents the average minimum number of bits needed to
 encode a string of symbols,based on the frequency of the symbols. *)
entplang
```

Out[85]=
```
2.83089
```

In[86]:=
```
(* Take this entropy, multiply it by the size of our text, and that gives the amount
 of bits required to optimally encode the string *)
optimalEnglishEncoding = entplang * sumchars
```

Out[86]= 9449.52

In[87]:=
```
(* Next, I will analyze a French text *)
(* Le tour du monde en quatre-vingts jours by Jules Verne
  is the French version of Around the World in Eighty Days *)
```

In[88]:=
```
frenchText = "
Mais si le rétablissement de la jeune Indienne ne fit pas question
dans l'esprit du brigadier général, celui-ci se montrait moins rassuré
pour l'avenir. Il n'hésita pas à dire à Phileas Fogg que si Mrs.
Aouda restait dans l'Inde, elle retomberait inévitablement entre les
mains de ses bourreaux. Ces énergumènes se tenaient dans toute la
péninsule, et certainement, malgré la police anglaise, ils sauraient
reprendre leur victime, fût-ce à Madras, à Bombay, à Calcutta. Et Sir
Francis Cromarty citait, à l'appui de ce dire, un fait de même nature
qui s'était passé récemment. A son avis, la jeune femme ne serait
véritablement en sûreté qu'après avoir quitté l'Inde.

Phileas Fogg répondit qu'il tiendrait compte de ces observations et
qu'il aviserait.

Vers dix heures, le guide annonçait la station d'Allahabad. Là
reprenait la voie interrompue du chemin de fer, dont les trains
franchissent, en moins d'un jour et d'une nuit, la distance qui sépare
Allahabad de Calcutta.

Phileas Fogg devait donc arriver à temps pour prendre un paquebot qui
ne partait que le lendemain seulement, 25 octobre, à midi, pour
Hong-Kong.

La jeune femme fut déposée dans une chambre de la gare. Passepartout
fut chargé d'aller acheter pour elle divers objets de toilette, robe,
châle, fourrures, etc., ce qu'il trouverait. Son maître lui ouvrait
un crédit illimité.

Passepartout partit aussitôt et courut les rues de la ville.
Allahabad, c'est la cité de Dieu, l'une des plus vénérées de l'Inde,
en raison de ce qu'elle est bâtie au confluent de deux fleuves sacrés,
le Gange et la Jumna, dont les eaux attirent les pèlerins de toute la
péninsule. On sait d'ailleurs que, suivant les légendes du Ramayana,
le Gange prend sa source dans le ciel, d'où, grâce à Brahma, il
descend sur la terre.";
```

In[89]:=
```
lowerFrenchText = ToLowerCase[frenchText]; (* Convert text to lowercase *)
```

In[90]:=
```
charcode = ToCharacterCode[lowerFrenchText];
(* Get a numerical character code for each letter in the string *)
```

In[91]:=
```
charentp = Table[{Length[Position[charcode, i]], FromCharacterCode[i]}, {i, 96, 122}];
(* make a table with the counts for each character *)
```

In[92]:=
```
charentp[[1]] = {Length[Position[charcode, 32]], "space"}
 (* Also, get the character count for spaces *)
```

Out[92]=
```
{262, space}
```

In[93]:=
```
TableForm[charentp] (* Put the counts and characters in a table to visualize easier *)
```

Out[93]//TableForm=

262	space
129	a
19	b
40	c
64	d
198	e
16	f
22	g
16	h
107	i
6	j
1	k
87	l
38	m
97	n
54	o
36	p
14	q
98	r
97	s
108	t
78	u
19	v
0	w
4	x
3	y
0	z

In[94]:=
```
sortcharentp = Reverse[Sort[charentp]]
 (* Sort by most common to least common frequencies *)
```

Out[94]=
```
{{262, space}, {198, e}, {129, a}, {108, t}, {107, i}, {98, r}, {97, s}, {97, n},
 {87, l}, {78, u}, {64, d}, {54, o}, {40, c}, {38, m}, {36, p}, {22, g}, {19, v},
 {19, b}, {16, h}, {16, f}, {14, q}, {6, j}, {4, x}, {3, y}, {1, k}, {0, z}, {0, w}}
```

In[95]:=
```
sumchars = Sum[sortcharentp[[i, 1]], {i, Length[sortcharentp]}]
 (* Get the total character count in the text *)
```

Out[95]=
```
1613
```

In[97]:=
```
(* Get the frequency for each character in the text *)
```

In[98]:=
```
charfreq = N[Table[{sortcharentp[[i, 1]] / sumchars, sortcharentp[[i, 2]]},
   {i, 1, Length[sortcharentp]}]]
```

Out[98]=
```
{{0.16243, space}, {0.122753, e}, {0.0799752, a}, {0.066956, t},
 {0.066336, i}, {0.0607564, r}, {0.0601364, s}, {0.0601364, n}, {0.0539368, l},
 {0.0483571, u}, {0.0396776, d}, {0.033478, o}, {0.0247985, c}, {0.0235586, m},
 {0.0223187, p}, {0.0136392, g}, {0.0117793, v}, {0.0117793, b},
 {0.0099194, h}, {0.0099194, f}, {0.00867948, q}, {0.00371978, j},
 {0.00247985, x}, {0.00185989, y}, {0.000619963, k}, {0., z}, {0., w}}
```

In[99]:=
```
TableForm[charfreq] (* Put this frequency in a table to visualize easier *)
```

Out[99]//TableForm=

0.16243	space
0.122753	e
0.0799752	a
0.066956	t
0.066336	i
0.0607564	r
0.0601364	s
0.0601364	n
0.0539368	l
0.0483571	u
0.0396776	d
0.033478	o
0.0247985	c
0.0235586	m
0.0223187	p
0.0136392	g
0.0117793	v
0.0117793	b
0.0099194	h
0.0099194	f
0.00867948	q
0.00371978	j
0.00247985	x
0.00185989	y
0.000619963	k
0.	z
0.	w

In[100]:=
```
entplang = 0;
(* Loop through the entire table, using the entropy formula,
calculate the entropy of the text given our frequencies *)
For[i = 1, i <= 27, i++,
            If[charfreq[[i, 1]] == 0, entplang,
                entplang = entplang - charfreq[[i, 1]] * Log[charfreq[[i, 1]]]
            ]
        ]
```

In[102]:=
```
(* This represents the average minimum number of bits needed to encode a string
 of symbols, based on the frequency of the symbols. *)
entplang
```

Out[102]=
```
2.78247
```

In[103]:=
```
(* Take this entropy, multiply it by the size of our text,
and that gives the amount of bits required to optimally encode the string *)
optimalFrenchEncoding = entplang * sumchars
```

Out[103]= 4488.12

In[104]:=
```
(* Next, I will analyze an Italian text *)
(* Orlando Furioso, and Italian epic poem by Ludovico Ariosto *)
```

In[105]:=
```
italianText = "Le donne, i cavallier, l'arme, gli amori,
  le cortesie, l'audaci imprese io canto,
  che furo al tempo che passaro i Mori
  d'Africa il mare, e in Francia nocquer tanto,
  seguendo l'ire e i giovenil furori
  d'Agramante lor re, che si diè vanto
  di vendicar la morte di Troiano
  sopra re Carlo imperator romano. Dirò d'Orlando in un medesmo tratto
  cosa non detta in prosa mai, né in rima:
  che per amor venne in furore e matto,
  d'uom che sì saggio era stimato prima;
  se da colei che tal quasi m'ha fatto,
  che 'l poco ingegno ad or ad or mi lima,
  me ne sarà però tanto concesso,
  che mi basti a finir quanto ho promesso. Piacciavi, generosa Erculea prole,
  ornamento e splendor del secol nostro,
  Ippolito, aggradir questo che vuole
  e darvi sol può l'umil servo vostro.
  Quel ch'io vi debbo, posso di parole
  pagare in parte e d'opera d'inchiostro;
  né che poco io vi dia da imputar sono,
  che quanto io posso dar, tutto vi dono.Voi sentirete fra i più degni eroi,
  che nominar con laude m'apparecchio,
  ricordar quel Ruggier, che fu di voi
  e de' vostri avi illustri il ceppo vecchio.
  L'alto valore e' chiari gesti suoi
  vi farò udir, se voi mi date orecchio,
  e vostri alti pensier cedino un poco,
  sì che tra lor miei versi abbiano loco. Orlando, che gran tempo innamorato
  fu de la bella Angelica, e per lei
  in India, in Media, in Tartaria lasciato
  avea infiniti ed immortal trofei,
  in Ponente con essa era tornato,
  dove sotto i gran monti Pirenei
  con la gente di Francia e de Lamagna
  re Carlo era attendato alla campagna,per far al re Marsilio e al re Agramante
  battersi ancor del folle ardir la guancia,
  d'aver condotto, l'un, d'Africa quante
  genti erano atte a portar spada e lancia;
  l'altro, d'aver spinta la Spagna inante
  a destruzion del bel regno di Francia.
  E così Orlando arrivò quivi a punto:
  ma tosto si pentì d'esservi giunto:
```

```
che vi fu tolta la sua donna poi:
  ecco il giudicio uman come spesso erra!
  Quella che dagli esperi ai liti eoi
  avea difesa con sì lunga guerra,
  or tolta gli è fra tanti amici suoi,
  senza spada adoprar, ne la sua terra.
  Il savio imperator, ch'estinguer volse
  un grave incendio, fu che gli la tolse.Nata pochi dì inanzi era una gara
  tra il conte Orlando e il suo cugin Rinaldo,
  che entrambi avean per la bellezza rara
  d'amoroso disio l'animo caldo.
  Carlo, che non avea tal lite cara,
  che gli rendea l'aiuto lor men saldo,
  questa donzella, che la causa n'era,
  tolse, e diè in mano al duca di Bavera;in premio promettendola a quel d'essi,
  ch'in quel conflitto, in quella gran giornata,
  degl'infideli più copia uccidessi,
  e di sua man prestasse opra più grata.
  Contrari ai voti poi furo i successi;
  ch'in fuga andò la gente battezzata,
  e con molti altri fu 'l duca prigione,
  e restò abbandonato il padiglione. ";
```

In[106]:=
```
lowerItalianText = ToLowerCase[italianText]; (* Convert text to lowercase *)
```

In[107]:=
```
charcode = ToCharacterCode[lowerItalianText];
(* Get a numerical character code for each letter in the string *)
```

In[108]:=
```
charentp = Table[{Length[Position[charcode, i]], FromCharacterCode[i]}, {i, 96, 122}]
 (* make a table with the counts for each character *)
```

Out[108]=
```
{{0, `}, {260, a}, {14, b}, {97, c}, {88, d}, {220, e}, {27, f}, {47, g}, {34, h},
 {215, i}, {0, j}, {0, k}, {121, l}, {56, m}, {146, n}, {204, o}, {62, p}, {14, q},
 {168, r}, {94, s}, {122, t}, {64, u}, {39, v}, {0, w}, {0, x}, {0, y}, {8, z}}
```

In[109]:=
```
charentp[[1]] = {Length[Position[charcode, 32]], "space"}
 (* Also, get the character count for spaces *)
```

Out[109]=
```
{550, space}
```

In[110]:=
```
TableForm[charentp] (* Put this frequency in a table to visualize easier *)
```

Out[110]//TableForm=

550	space
260	a
14	b
97	c
88	d
220	e
27	f
47	g
34	h
215	i
0	j
0	k
121	l
56	m
146	n
204	o
62	p
14	q
168	r
94	s
122	t
64	u
39	v
0	w
0	x
0	y
8	z

In[111]:=
```
sortcharentp = Reverse[Sort[charentp]]
 (* Sort by most common to least common frequencies *)
```

Out[111]=
```
{{550, space}, {260, a}, {220, e}, {215, i}, {204, o}, {168, r}, {146, n}, {122, t},
 {121, l}, {97, c}, {94, s}, {88, d}, {64, u}, {62, p}, {56, m}, {47, g}, {39, v},
 {34, h}, {27, f}, {14, q}, {14, b}, {8, z}, {0, y}, {0, x}, {0, w}, {0, k}, {0, j}}
```

In[112]:=
```
sumchars = Sum[sortcharentp[[i, 1]], {i, Length[sortcharentp]}]
 (* Get the total character count in the text *)
```

Out[112]=
```
2650
```

In[113]:=
```
(* Get the frequency for each character in the text *)
charfreq = N[Table[{sortcharentp[[i, 1]] / sumchars, sortcharentp[[i, 2]]},
   {i, 1, Length[sortcharentp]}]]
```

Out[113]=
```
{{0.207547, space}, {0.0981132, a}, {0.0830189, e}, {0.0811321, i}, {0.0769811, o},
 {0.0633962, r}, {0.0550943, n}, {0.0460377, t}, {0.0456604, l}, {0.0366038, c},
 {0.0354717, s}, {0.0332075, d}, {0.0241509, u}, {0.0233962, p}, {0.0211321, m},
 {0.0177358, g}, {0.014717, v}, {0.0128302, h}, {0.0101887, f}, {0.00528302, q},
 {0.00528302, b}, {0.00301887, z}, {0., y}, {0., x}, {0., w}, {0., k}, {0., j}}
```

In[114]:=
```
TableForm[charfreq] (* Put this frequency in a table to visualize easier *)
```

Out[114]//TableForm=

0.207547	space
0.0981132	a
0.0830189	e
0.0811321	i
0.0769811	o
0.0633962	r
0.0550943	n
0.0460377	t
0.0456604	l
0.0366038	c
0.0354717	s
0.0332075	d
0.0241509	u
0.0233962	p
0.0211321	m
0.0177358	g
0.014717	v
0.0128302	h
0.0101887	f
0.00528302	q
0.00528302	b
0.00301887	z
0.	y
0.	x
0.	w
0.	k
0.	j

In[115]:=
```
entplang = 0;
(* Loop through the entire table, using the entropy formula,
calculate the entropy of the text given our frequencies *)
For[i = 1, i <= 27, i++,
            If[charfreq[[i, 1]] == 0, entplang,
                entplang = entplang - charfreq[[i, 1]] * Log[charfreq[[i, 1]]]
            ]
        ]
```

In[117]:=
```
(* This represents the average minimum number of bits needed to
 encode a string of symbols, based on the frequency of the symbols .*)
entplang
```

Out[117]= 2.70014

In[118]:=
```
(* Take this entropy, multiply it by the size of our text, and that gives the amount
 of bits required to optimally encode the string *)
optimalItalianEncoding = entplang * sumchars
```

Out[118]= 7155.37

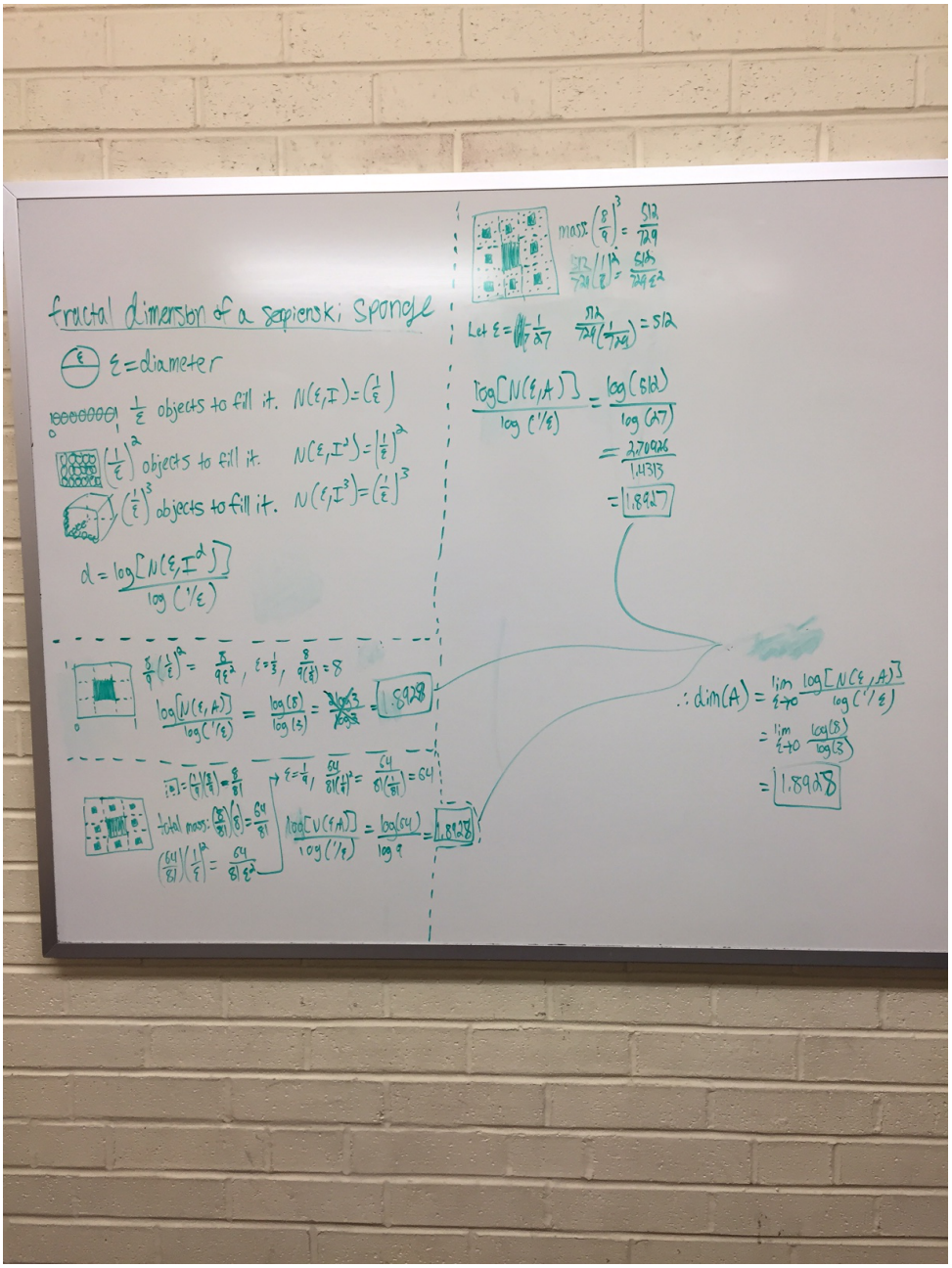
In[119]:=
```
(* Analysis on languages *)
(* English: 2.83089, French: 2.78246, Italian: 2.70014 *)
(* In general, all three of these languages have a very similar entropy. Analyzing even
 longer texts can yeild more accurate results due to the law of large numbers *)

(* Since English is a Germanic language, and Italian and French are Romance languages,
I expected English' entropy to be very different from French and Italian *)

(* The space, letter e, and a,
were all top 3 for these languages for relative frequency,
and were similar frequencies among
 the three languages. Since the top 3 were similar across all three languages,
it's not suprising
 that the entropies of the languages are relatively the same *)

(* From this test, I am concluding that there isn't a significant difference between the
 language's entropies. *)
```

In[120]:=

```
(* Problem 4 *)

(* Calculate the fractional dimension of the Sierpinski carpet and "sponge" *)

(* Sierpinski carpet *)
```

```
(* consider a 2D square. As discussed in class, we need (1/epsilon)^2 objects
 to "fill" the square, where epsilon is the diameter. Let E refer to epsilon. *)
(* dimension = log[N{E,A}] / log[1/E] *)
(* log[N{E,A}] is calculated by taking the mass of the current iteration of the carpet,
and plugging E into that mass equation. *)

(* For the first iteration, the mass of the carpet is 8/9, the next is 8/9 squared,
then cubed, and so on. These are all multiplied by (1/E) squared. A
 For loop can be used to iterate through a few iterations of the carpet,
and converge on an answer. Let epsilon be raised to the 1,
2nd, 3rd, etc. power for every iteration *)


(* Note, the title of this whiteboard says sponge,
but it should be carpet. This is the work shown for carpet equations  *)
```

In[126]:=

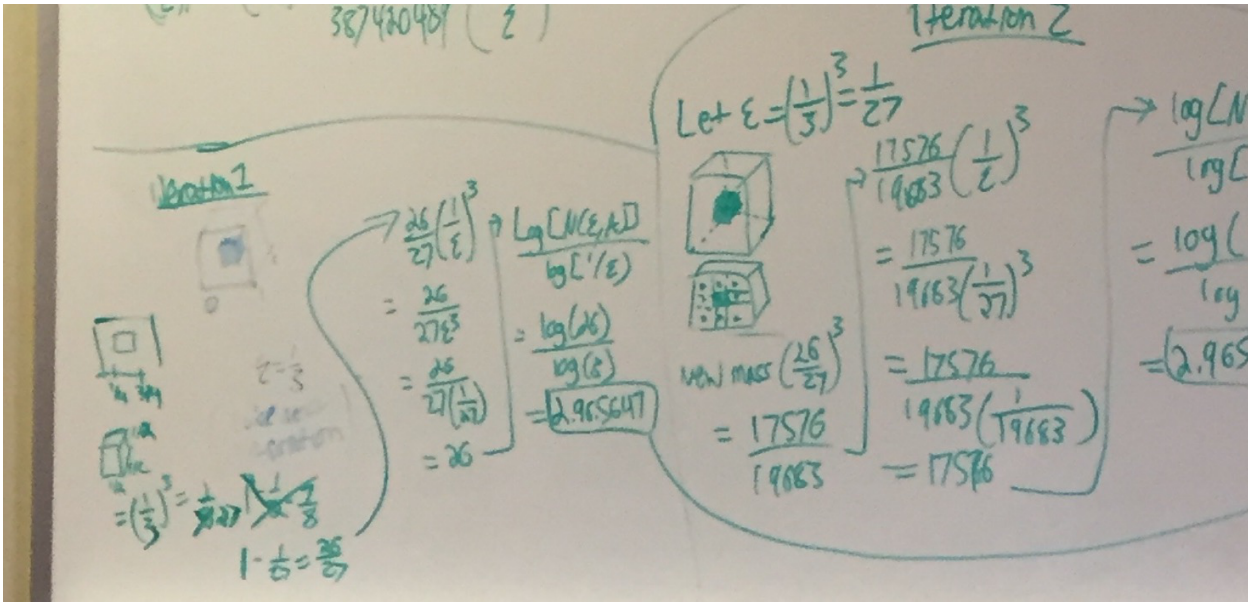



```
epsilon = 1 / 3;
mass = 8 / 9; (* The "mass" of the object at iteration 1 *)

(* Loop 4 times to get a result for the dimension,
using the formula for a dimension given in class *)
For[i = 1, i < 5, i++, (
    epsilonTemp = (epsilon ^ i) ^ 2; (* For every iteration, epsilon is raised to
     the power of that iteration, then squared to fill the whole carpet *)
    massTemp = mass ^ i; (* The mass is raised to the
     power of the iteration we are on *)
    massEquation = massTemp / epsilonTemp; (* Plug in the epsilon
```

```
    value into our equation for mass and solve *)
   Print[N[Log[massEquation] / Log[1 / (epsilon^i)]]]) (* Formula for dimensions *)
 ];
```

```
1.89279

1.89279

1.89279

1.89279
```

```
(* Therefore the dimension(A) =
 limit as E approahces infinity of Log[N{E,A}] / log(1/E) = 1.89279 *)

(* Sierpinski sponge *)

(* This is relatively the same procedure. The "filling" is now (1/E)^3,
and the mass starts off at 26/27 for iteration 1, and cubes for every next iteration *)


(* Work shown for sponge equations *)
```




```
epsilon = 1 / 3;
mass = 26 / 27; (* The "mass" of the object at iteration 1 *)
```

```
(* Loop 4 times to get a result for the dimension,
using the formula for a dimension given in class *)
For[i = 1, i < 5, i ++, (
   epsilonTemp = epsilon ^ (3^ (i - 1)); (* This calculates epsilon as 1/3, then (1/3)^3,
   then (1/3)^3^3, etc., for iterations 1,2,3, etc. Cubed each iteration *)
   massTemp = mass ^ (3 ^ (i - 1)); (* The mass starts
    off at 26/27 and is cubed each iteration *)
   massEquation = massTemp / ((epsilonTemp) ^3);
   (* Plug in the epsilon value into our equation for mass and solve *)
   Print[N[Log[massEquation] / Log[1 / (epsilonTemp)]]]) (* Formula for dimensions *)
 ];
```

```
2.96565

2.96565

2.96565

2.96565
```

```
(* Therefore the dimension(A) =
 limit as E approahces infinity of Log[N{E,A}] / log(1/E) = 2.96565 *)
```