Parse Time: Hard vs Soft : No-Load Environment

Author: Craig Shallahamer (craig@orapub.com), Version 1a, 13-July-2012

Background and Purpose

The purpose of this notepad is to simply compare the differences in parse times from hard and soft parsing based on the exec
tion number of a unique SQL statement... in a no-load enviroment

Experimental Data and Loading

Below is all the experimental data. The experiment was run on a Dell single six-core CPU, Oracle 11.2G. According to “c
/proc/version”: Linux version 2.6.32-300.3.1.el6uek.x86_64 (mockbuild@ca-build44.us.oracle.com) (gcc version 4.4.4 201007
(Red Hat 4.4.4-13) (GCC)) #1 SMP Fri Dec 9 18:57:35 EST 2011. There was a no outside load on the system; essentially
single user Linux system. For each simular yet unique 30 SQL statements the parse time was gatherd (based on a SQL tra
file) when it was run seven times. Details are presented in the associated blog posting in early July of 2012.

The order of sample data simply the parse time in either only CPU or the total elapsed time (which includes CPU and Oracle w
time).

In[1]:=

```
ssNum = 7;
ssCpu[1] = {22 996, 22 997, 22 996, 22 996, 22 997, 22 996, 22 996, 22 997,
   22 996, 22 996, 22 996, 21 997, 22 997, 22 997, 22 997, 21 996, 22 997, 21 997, 22 996,
   22 997, 22 996, 22 997, 22 997, 22 997, 22 997, 22 996, 22 996, 22 997, 21 996, 21 997};
ssCpu[2] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
ssCpu[3] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
ssCpu[4] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
ssCpu[5] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
ssCpu[6] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
ssCpu[7] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
ssElp[1] = {23 374, 23 539, 23 438, 23 391, 23 474, 23 376, 23 495, 23 445,
   23 401, 23 437, 23 462, 23 610, 23 469, 23 473, 23 516, 23 384, 23 572, 23 479, 23 434,
   23 619, 23 478, 23 497, 23 506, 23 515, 23 527, 23 530, 23 614, 23 509, 23 532, 23 625};
ssElp[2] = {126, 132, 128, 132, 127, 131, 126, 127, 126, 128, 136, 126, 127, 129, 133,
   130, 134, 129, 132, 132, 137, 132, 132, 129, 133, 129, 128, 161, 128, 128};
ssElp[3] = {25, 24, 24, 24, 24, 25, 25, 25, 25, 24, 25, 24, 38, 25, 39, 25,
   25, 41, 24, 25, 25, 28, 25, 25, 25, 25, 39, 25, 40, 25};
ssElp[4] = {18, 18, 18, 29, 17, 28, 17, 28, 18, 18, 28, 28, 17, 29, 17, 28,
   17, 22, 17, 19, 22, 18, 18, 17, 18, 18, 18, 29, 17, 18};
ssElp[5] = {17, 25, 29, 17, 22, 17, 18, 29, 17, 18, 18, 17, 32, 19, 17, 18,
   18, 29, 18, 18, 18, 31, 18, 19, 18, 17, 29, 18, 18, 29};
ssElp[6] = {17, 18, 18, 23, 17, 17, 29, 18, 17, 17, 17, 17, 18, 18, 17, 18,
   18, 18, 29, 18, 17, 18, 17, 29, 18, 17, 18, 17, 18, 18};
ssElp[7] = {17, 18, 18, 17, 18, 19, 18, 19, 17, 17, 18, 17, 18, 17, 17, 18,
   23, 17, 18, 18, 18, 18, 17, 18, 18, 18, 17, 18, 29, 18};
```

Basic Numeric Comparision

No comments.

```
In[16]:=
myData = Table[
   {
    ssidx,
    N[Mean[ssElp[ssidx] / 1000]], N[Round[Mean[ssElp[1]] / Mean[ssElp[ssidx]]]],
    N[StandardDeviation[ssElp[ssidx]] / 1000], DistributionFitTest[ssElp[ssidx]], Length[ssElp[ssidx]]
   }, {ssidx, 1, ssNum}
  ];
toGrid = Prepend[myData, {
    "Execution",
    "Elapsed Time\nAvg (ms)", "Elapsed Time\nX Times Faster",
    "Elapsed Time\nStdev (ms)", "Elapsed Time\nP-Value", "Elapsed Time\nSamples"
   }];
Grid[toGrid, Frame → All]
```

Out[18]=

Execution	Elapsed Time Avg (ms)	Elapsed Time X Times Faster	Elapsed Time Stdev (ms)	Elapsed Time P-Value	Elapsed Time Samples
1	23.4907	1.	0.0715822	0.766373	30
2	0.130933	179.	0.00641622	3.70683×10^{-6}	30
3	0.0272667	862.	0.00558281	0	30
4	0.0208	1129.	0.0048023	0	30
5	0.0209333	1122.	0.00520566	0	30
6	0.0188667	1245.	0.00360778	0	30
7	0.0182667	1286.	0.00231834	0	30

Sample Set Normality Tests

Before we can perform a standard t-test hypothesis tests on our data, we need to ensure it is normally distributed...because tl is one of the underlying assumptions and requirements for properly performing a t-test.

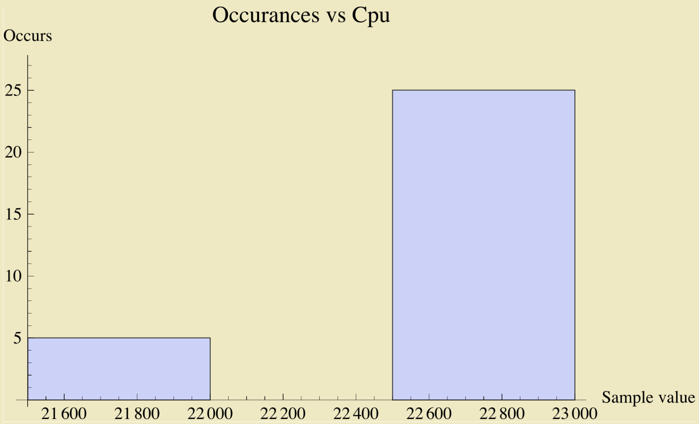
Statistical and vIsual normality test

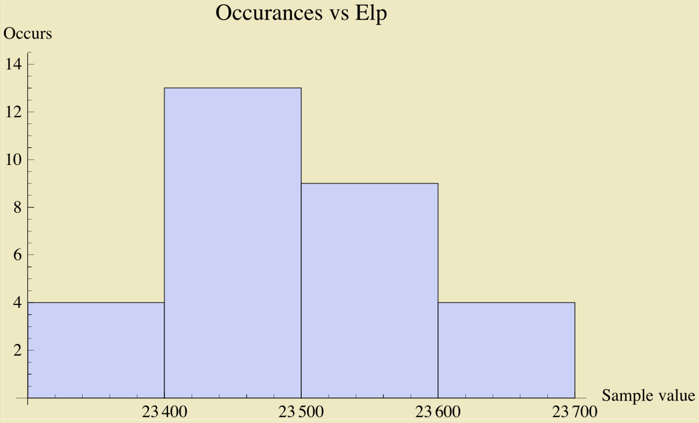
Our alpha will be 0.05, so if the distribution fit test results in a value greater than 0.05 then we can assume the data set is inde normally distributed.

The first test is just to double check to make sure my thinking is correct. Since I creating a normal distribution based on a me and standard deviation (just happens to be based on the my sample set data), I would expect a p-value (the result) to grea exceed 0.05. Notice that the more samples I have created (the final number), the closer the p-value approaches 1.0.

```
In[19]:=
check = DistributionFitTest[
   RandomVariate[NormalDistribution[Mean[ssCpu[1]], StandardDeviation[ssCpu[1]]], 10 000]];
Do[
  pValueCpu = DistributionFitTest[ssCpu[i]];
  pValueElp = DistributionFitTest[ssElp[i]];
  Print["Sample set ", i, " with ", Length[ssCpu[i]],
   " values. P-values: Cpu=", pValueCpu, " Elp=", pValueElp];
  cpu = Histogram[ssCpu[i], PlotLabel → "Occurances vs Cpu", AxesLabel → {"Sample value", "Occurs"}];
  elp = Histogram[ssElp[i], PlotLabel → "Occurances vs Elp", AxesLabel → {"Sample value", "Occurs"}];
  Print[cpu];
  Print[elp];
  Print["--------------------------------------------"];
  , {i, 1, ssNum}
 ];
Print["This number should be much greater than 0.05: ", check, " If not try again by re-evaluating."];
```

```
Sample set 1 with 30 values. P-values: Cpu=0 Elp=0.766373
```

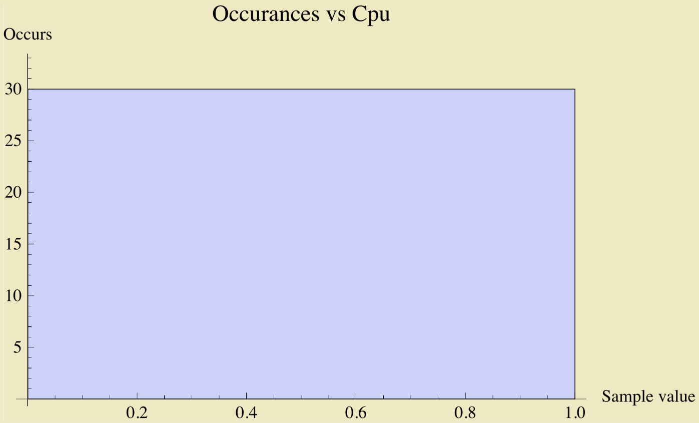


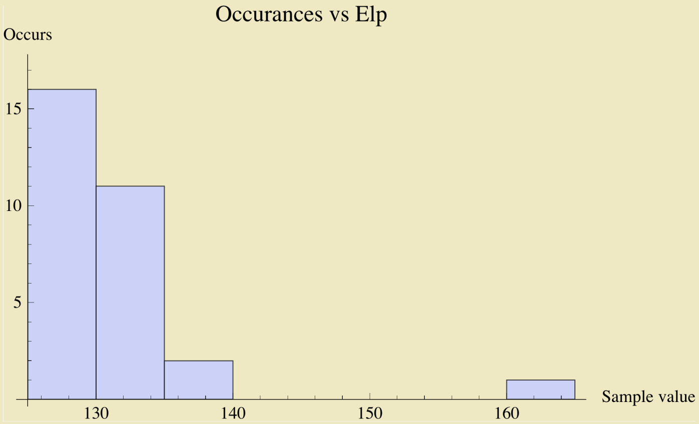


```
----------------------------------------
```

```
DistributionFitTest::rectuv :
   The argument {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} should be a vector or matrix
      of real numbers with positive variance. ≫
```

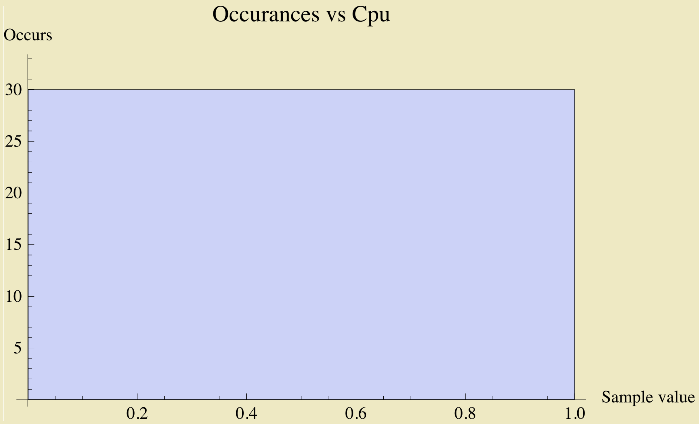
```
Sample set 2 with 30 values. P-values: Cpu=DistributionFitTest[
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}] Elp=3.70683 × 10^-6
```

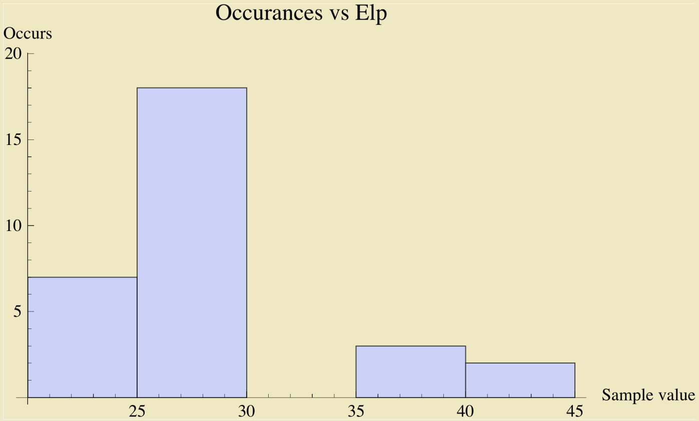




```
----------------------------------------

DistributionFitTest::rectuv :
   The argument {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} should be a vector or matrix
        of real numbers with positive variance. ≫
```

```
Sample set 3 with 30 values. P-values: Cpu=
 DistributionFitTest[{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}] Elp=0
```

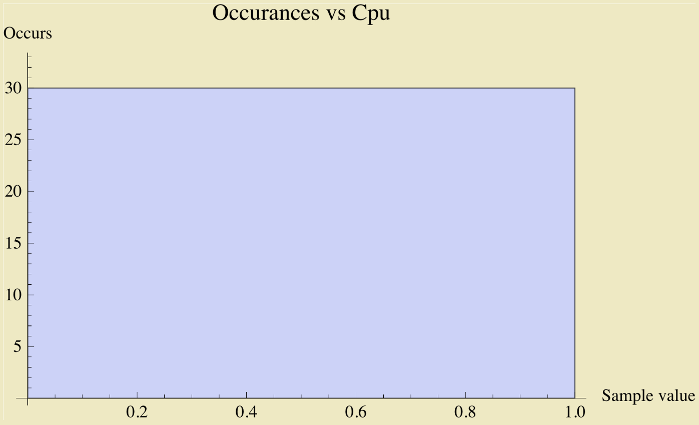


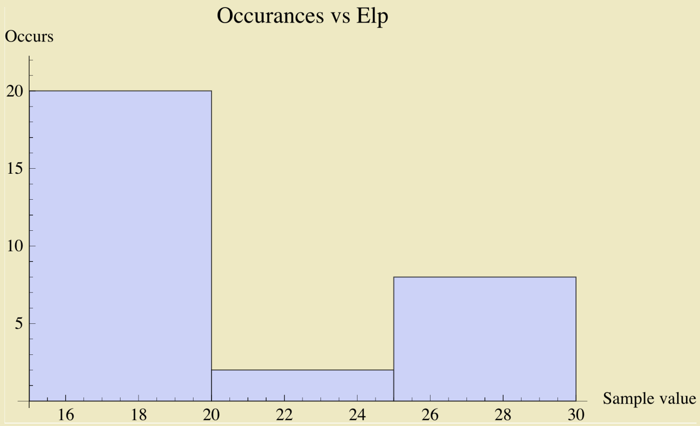


```
------------------------------------------
```

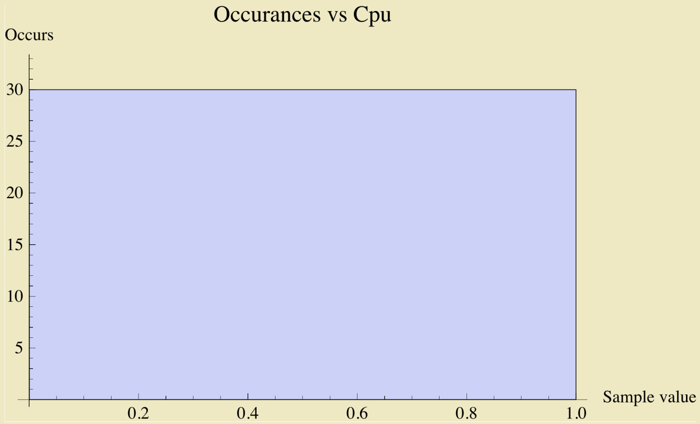
```
DistributionFitTest::rectuv :
   The argument {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} should be a vector or matrix
      of real numbers with positive variance. ≫

General::stop : Further output of DistributionFitTest::rectuv will be suppressed during this calculation. ≫
```

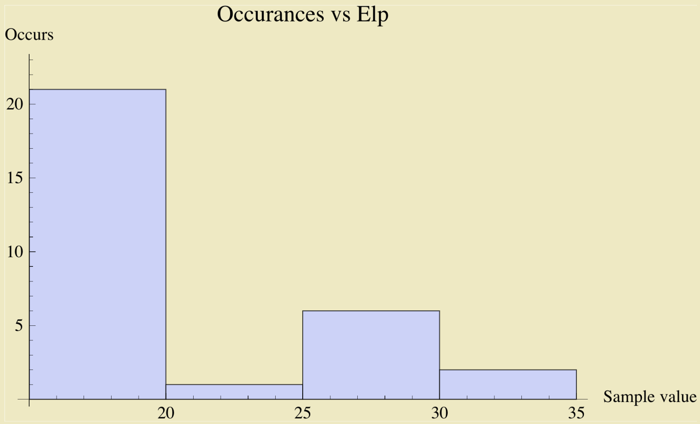
```
Sample set 4 with 30 values. P-values: Cpu=
 DistributionFitTest[{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}] Elp=0
```



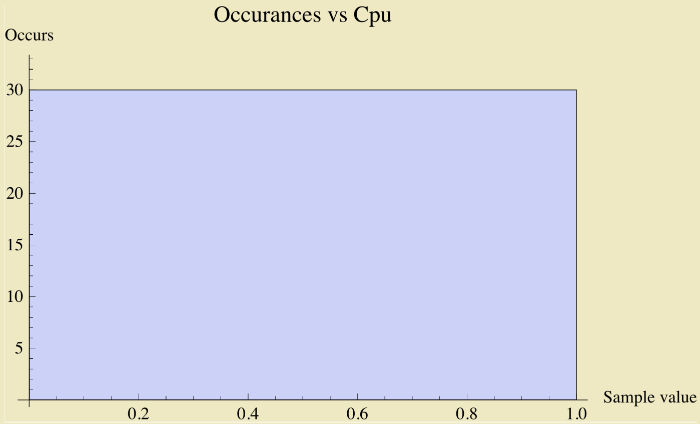


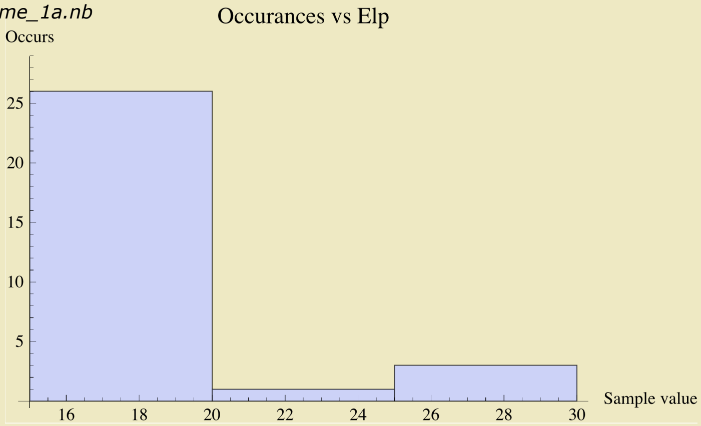
```
-------------------------------------------

Sample set 5 with 30 values. P-values: Cpu=
 DistributionFitTest[{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}] Elp=0
```



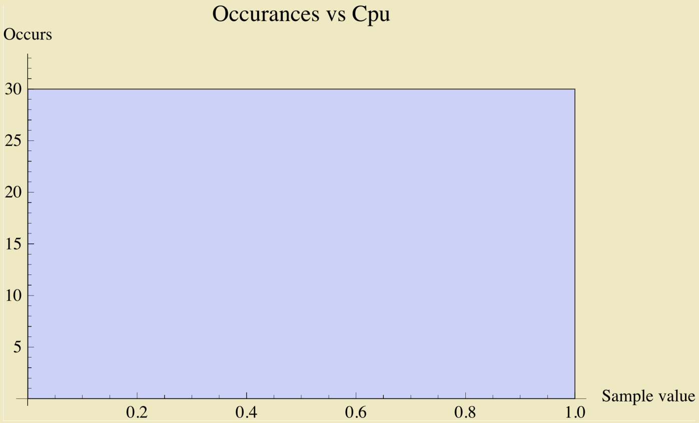


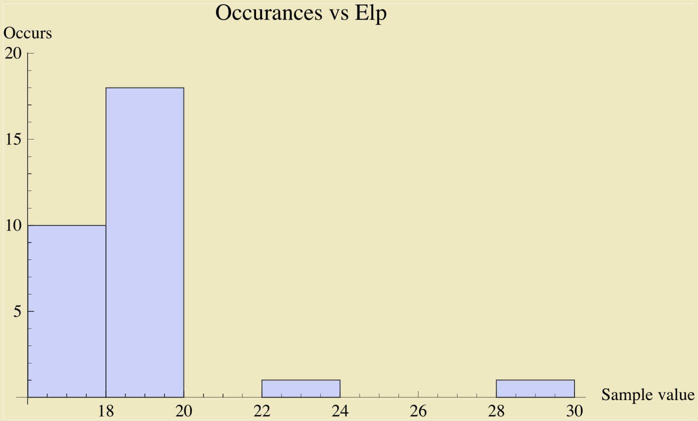
```
-------------------------------------------

Sample set 6 with 30 values. P-values: Cpu=
 DistributionFitTest[{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}] Elp=0
```





```
--------------------------------------------

Sample set 7 with 30 values. P-values: Cpu=
 DistributionFitTest[{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}] Elp=0
```





```
--------------------------------------------

This number should be much greater than 0.05: 0.219823 If not try again by re-evaluating.
```

Sample Comparison Tests (when normality exists)

Assuming our samples **are normally distributed**, now it’s time to see if they are significantly different. If so, then we kno changing the number of latches and chains indeed makes a significant performance difference...at least statistically.

The null hypothesis is; there is no real difference between our samples sets. We need to statistically prove that any difference the result of randomness; like we just happened to pick poor set of samples and it makes their difference look much worse thar really is.

A t-test will produce a statistic p. The p value is a probability, with a value ranging from zero to one. It is the answer to this ques tion: If the populations really have the same mean overall, what is the probability that random sampling would lead to a differen

For example, if the p value is 0.03 we can say a random sampling from identical populations would lead to a difference smal than you observed in 97% of the experiments and larger than you observed in 3% of the experiments.

Said another way, suppose I have a single sample set and I copy it, resultling in two identical sample sets. Now suppose w perform a significance test on these two identical sample sets. The resuting p-value will be 1.0 because they are exactly t same. We are essentially doing the same thing here except we have to different sample sets... but we still want to see if th “like” each other..and in our case we hope they are NOT the like each other, which means the p-value will low... below our cut value of 0.05.

For our analysis we choose alpha of 0.05. To accept that our two samples are statistically similar the p value would need to less than 0.05 (our alpha).

Good reference about the P-Value and significance testing: http://www.graphpad.com/articles/pvalue.htm

Here we go (assuming our samples are normally distributed):

1. Our P value threshold is 0.05, which is our alpha.
2. The null hypothesis is the two populations have the same mean. (Remember we have to sample sets, which not the popu tion.)
3. Do the statistical test to compute the P value.
4. Compare the result P value to our threshold alpha value. If the P value is less then our threshold, we will reject the null hypoth sis and say the difference between our samples is significant. However, if the P value is greater than the threshold, we cann reject the null hypothesis and any difference between our samples are not statistically significant.

In[22]:=

```
Do[
  pValueCpu = TTest[{ssCpu[i], ssCpu[i + 1]}];
  Print["Cpu: (", Length[ssCpu[i]],
   " values) pvalue between sample set ", i, " and ", i + 1, " is ", pValueCpu];
  pValueElp = TTest[{ssElp[i], ssElp[i + 1]}];
  Print["Elp: (", Length[ssElp[i]],
   " values) pvalue between sample set ", i, " and ", i + 1, " is ", pValueElp];
  ,
  {i, 1, ssNum - 1}
 ];
```

```
TTest::invltd : The argument
        {{22996, 22997, 22996, 22996, 22997, 22996, 22996, 22997, 22996, 22996, 22996, 21997, 22997, 22997, 22997, 21996, 22997,
           21997, 22996, 22997, 22996, 22997, 22997, 22997, 22997, 22996, 22996, 22997, 21996, 21997}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}} should be a vector of real numbers with positive variance, a real matrix
        with positive definite covariance and dimension less than length, or two such vectors or matricies of equal dimension. ≫
```

```
Cpu: (30 values) pvalue between sample set 1 and 2 is
 TTest[{{22 996, 22 997, 22 996, 22 996, 22 997, 22 996, 22 996, 22 997, 22 996, 22 996, 22 996, 21 997, 22 997, 22 997,
    22 997, 21 996, 22 997, 21 997, 22 996, 22 997, 22 996, 22 997, 22 997, 22 997, 22 997, 22 996, 22 996, 22 997,
    21 996, 21 997}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]
```

```
TTest::nortst : At least one of the p-values in {0.766373, 3.70683 × 10^-6}, resulting
        from a test for normality, is below 0.025`. The tests in {T} require that the data is normally distributed. ≫
```

```
Elp: (30 values) pvalue between sample set 1 and 2 is 1.07723 × 10^-75
```

```
TTest::invltd : The argument
        {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0}} should be a vector of real numbers with positive variance, a real matrix with
        positive definite covariance and dimension less than length, or two such vectors or matricies of equal dimension. ≫
```

```
Cpu: (30 values) pvalue between sample set 2 and 3 is
 TTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]
```

```
TTest::nortst : At least one of the p-values in {3.70683 × 10^-6, 0}, resulting from
        a test for normality, is below 0.025`. The tests in {T} require that the data is normally distributed. ≫
```

```
Elp: (30 values) pvalue between sample set 2 and 3 is 9.48079 × 10^-56
```

```
TTest::invltd : The argument
        {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
General::stop : Further output of TTest::invltd will be suppressed during this calculation. ≫

Cpu: (30 values) pvalue between sample set 3 and 4 is
 TTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]

TTest::nortst : At least one of the p-values in {0, 0}, resulting from a
     test for normality, is below 0.025`. The tests in {T} require that the data is normally distributed. ≫

General::stop : Further output of TTest::nortst will be suppressed during this calculation. ≫

Elp: (30 values) pvalue between sample set 3 and 4 is 0.0000111383

Cpu: (30 values) pvalue between sample set 4 and 5 is
 TTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]

Elp: (30 values) pvalue between sample set 4 and 5 is 0.918228

Cpu: (30 values) pvalue between sample set 5 and 6 is
 TTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]

Elp: (30 values) pvalue between sample set 5 and 6 is 0.0791274

Cpu: (30 values) pvalue between sample set 6 and 7 is
 TTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]

Elp: (30 values) pvalue between sample set 6 and 7 is 0.446592
```

If the above T-Test results (p value) are less then our threshold we can say there is a significant difference between the tw sample sets.

Sample Comparison Tests (when normality does NOT exist)

If our sample sets are **not normally distributed,** we can not perform a simple t-test. We can perform what are called locati tests. I did some research on significance testing when non-normal distributions exists. I found a very nice reference:

http://www.statsoft.com/textbook/nonparametric-statistics

The paragraph below (which is from the reference above) is a key reference to what we're doing here:

...the need is evident for statistical procedures that enable us to process data of "low quality," from small samples, on variabl about which nothing is known (concerning their distribution). Specifically, nonparametric methods were developed to be used cases when the researcher knows nothing about the parameters of the variable of interest in the population (hence the nar nonparametric). In more technical terms, nonparametric methods do not rely on the estimation of parameters (such as the me or the standard deviation) describing the distribution of the variable of interest in the population. Therefore, these methods a also sometimes (and more appropriately) called parameter-free methods or distribution-free methods.

Being that I'm not a statistician but still need to determine if these sample sets are significant different, I let *Mathematica* dete mine the appropriate test. Notice that one of the above mentioned tests will probably be the test *Mathematica* chooses.

Note: If we run our normally distributed data through this analysis (speically, the "LocationEquivalenceTest"), *Mathematica* shou detect this and use a more appropriate significant test, like a t-test.

Here we go with the hypothesis testing (assuming our sample sets are not normally distributed):

1. Our P value threshold is 0.05, which is our alpha.
2. The null hypotheses is the two populations have the same mean. (Remember we have to sample sets, which is not the popu tion.)
3. Do the statistical test to compute the P value.
4. Compare the result P value to our threshold alpha value. If the P value is less then our threshold, we will reject the null hypoth sis and say the difference between our samples is significant. (Which is what I'm hoping to see.) However, if the P value greater than the threshold, we cannot reject the null hypothesis and any difference between our samples are not statistica significant; randomness, picked the "wrong" samples, etc.

In[23]:=

```
myData = Table[
   {
    y,
    N[Round[MannWhitneyTest[{ssElp[1], ssElp[y]}], 1 / 1000]],
    N[Round[MannWhitneyTest[{ssElp[2], ssElp[y]}], 1 / 1000]],
    N[Round[MannWhitneyTest[{ssElp[3], ssElp[y]}], 1 / 1000]],
    N[Round[MannWhitneyTest[{ssElp[4], ssElp[y]}], 1 / 1000]],
    N[Round[MannWhitneyTest[{ssElp[5], ssElp[y]}], 1 / 1000]],
    N[Round[MannWhitneyTest[{ssElp[6], ssElp[y]}], 1 / 1000]],
    N[Round[MannWhitneyTest[{ssElp[7], ssElp[y]}], 1 / 1000]]
   }, {y, 1, ssNum}
  ];

toGrid = Prepend[myData, {
    "-", "1", "2", "3", "4", "5", "6", "7"}
  ];
Grid[toGrid, Frame → All]
```

Out[25]=

-	1	2	3	4	5	6	7
1	0.994	0.	0.	0.	0.	0.	0.
2	0.	0.994	0.	0.	0.	0.	0.
3	0.	0.	0.994	0.	0.	0.	0.
4	0.	0.	0.	0.994	0.72	0.112	0.114
5	0.	0.	0.	0.709	0.994	0.055	0.071
6	0.	0.	0.	0.115	0.056	0.994	0.757
7	0.	0.	0.	0.118	0.074	0.745	0.994

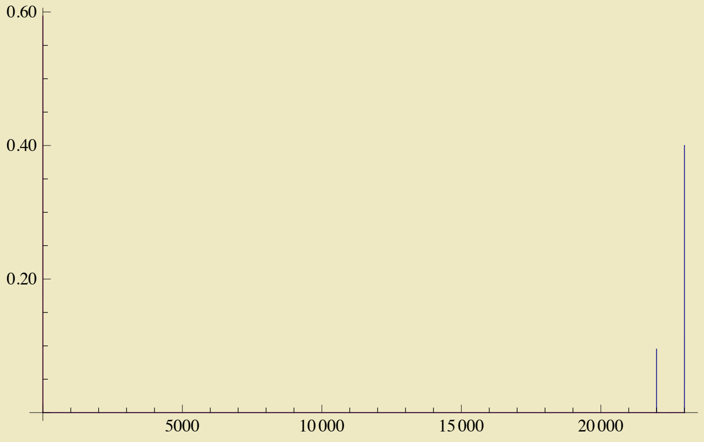
In[26]:=

```
Do[
  CpuHist = SmoothHistogram[{ssCpu[i], ssCpu[i + 1]}];
  CpuTest1 = MannWhitneyTest[{ssCpu[i], ssCpu[i + 1]}];
  CpuTest2 = LocationEquivalenceTest[{ssCpu[i], ssCpu[i + 1]}, {"TestDataTable", "AutomaticTest"}];
  Print["Cpu: (", Length[ssCpu[i]], " values) Between sample ",
   i, " and ", i + 1, ". Test1=", CpuTest1, " Test2=", CpuTest2];
  Print[CpuHist];
  Print["------------------------------------------------"];
  ElptHist = SmoothHistogram[{ssElp[i], ssElp[i + 1]}];
  ElpTest1 = MannWhitneyTest[{ssElp[i], ssElp[i + 1]}];
  ElpTest2 = LocationEquivalenceTest[{ssElp[i], ssElp[i + 1]}, {"TestDataTable", "AutomaticTest"}];
  Print["Elp: (", Length[ssElp[i]], " values) Between sample ",
   i, " and ", i + 1, ". Test1=", ElpTest1, " Test2=", ElpTest2];
  Print[ElptHist];
  Print[
   "---------------------------------------------------------------------------------------"
  , {i, 1, ssNum - 1}
 ];
```

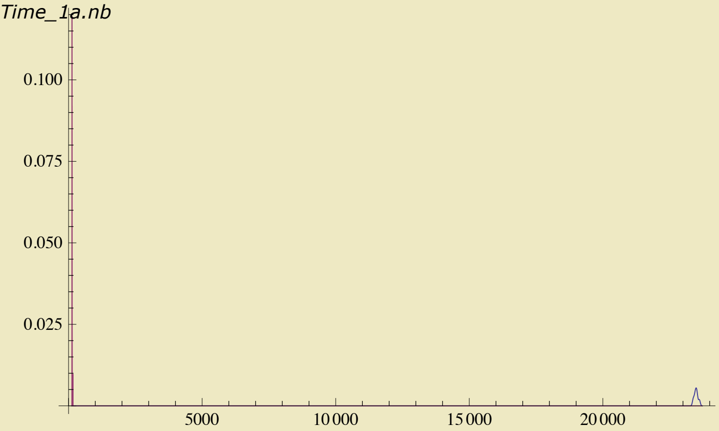
```
MannWhitneyTest::invltd : The argument
    {{22996, 22997, 22996, 22996, 22997, 22996, 22996, 22997, 22996, 22996, 22996, 21997, 22997, 22997, 22997, 21996, 22997,
       21997, 22996, 22997, 22996, 22997, 22997, 22997, 22997, 22996, 22996, 22997, 21996, 21997}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}} should be a vector of real numbers with positive variance, a real matrix
    with positive definite covariance and dimension less than length, or two such vectors or matricies of equal dimension. ≫

Table::iterb : Iterator
    {Statistics`LocationEquivalenceTestingDump`i, If[Min[0, DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}]] ≥ 0.025, {KSampleT}, {KruskalWallis}]}
    does not have appropriate bounds. ≫

Table::iterb : Iterator
    {Statistics`LocationEquivalenceTestingDump`i, If[Min[0, DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
              0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}]] ≥ 0.025, {KSampleT}, {KruskalWallis}]}
    does not have appropriate bounds. ≫

Table::iterb :
  Iterator {Statistics`LocationEquivalenceTestingDump`i, Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[
            Statistics`LocationEquivalenceTestingDump`i][HypothesisTestData[≪LocationEquivalenceTest≫]], {
          Statistics`LocationEquivalenceTestingDump`i, If[Min[0, DistributionFitTest[{≪30≫}]] ≥ 0.025, { …
              T}, {KruskalWallis}]}]} does not have appropriate bounds. ≫
```

```
Table::itform : Argument Statistics`LocationEquivalenceTestingDump`i at position 2 does not have the correct form for an iterator. ≫

Transpose::nmtx : The first two levels of the one−dimensional list
          {Table[If[Length[Statistics`LocationEquivalenceTestingDump`i] == 0, Statistics`LocationEquivalenceTestingDump`i,
                    Statistics`LocationEquivalenceTestingDump`i⟦1⟧], {Statistics`LocationEquivalenceTestingDump`i, Table[HypothesisTestData[≪
                              LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i]}], ≪1≫}
          cannot be transposed. ≫

Transpose::nmtx : The first two levels of the one−dimensional list
          {Statistics`LocationEquivalenceTestingDump`iFormatTestNames[If[Min[0, DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}]] ≥ 0.025, {KSampleT}, {KruskalWallis}]]}
          cannot be transposed. ≫

Table::itform : Argument Statistics`LocationEquivalenceTestingDump`i at position 2 does not have the correct form for an iterator. ≫

Transpose::nmtx : The first two levels of the one−dimensional list
          {Table[If[Length[Statistics`LocationEquivalenceTestingDump`i] == 0, Statistics`LocationEquivalenceTestingDump`i,
                    Statistics`LocationEquivalenceTestingDump`i⟦1⟧], {Statistics`LocationEquivalenceTestingDump`i, Table[HypothesisTestData[≪
                              LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i]}], ≪1≫}
          cannot be transposed. ≫

General::stop : Further output of Transpose::nmtx will be suppressed during this calculation. ≫

Join::headsd : Expression
          Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[If[Min[0, DistributionFitTest[{≪30≫}]] ≥ 0.025, {KSampleT},
                        {KruskalWallis}]]}] at position 1 is expected to have head Transpose for all subexpressions through level 2. ≫

Table::itform : Argument Statistics`LocationEquivalenceTestingDump`i at position 2 does not have the correct form for an iterator. ≫

General::stop : Further output of Table::itform will be suppressed during this calculation. ≫

Join::headsd : Expression
          Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[If[Min[0, DistributionFitTest[{≪30≫}]] ≥ 0.025, {KSampleT},
                        {KruskalWallis}]]}] at position 1 is expected to have head Transpose for all subexpressions through level 2. ≫

Join::heads : Heads List and Transpose at positions 1 and 2 are expected to be the same. ≫

Join::headsd : Expression
          Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[If[Min[0, DistributionFitTest[{≪30≫}]] ≥ 0.025, {KSampleT},
                        {KruskalWallis}]]}] at position 2 is expected to have head List for all subexpressions through level 2. ≫

General::stop : Further output of Join::headsd will be suppressed during this calculation. ≫
```

```
Cpu: (30 values) Between sample 1 and 2. Test1=
 MannWhitneyTest[{{22 996, 22 997, 22 996, 22 996, 22 997, 22 996, 22 996, 22 997,
    22 996, 22 996, 22 996, 21 997, 22 997, 22 997, 22 997, 21 996, 22 997, 21 997, 22 996,
    22 997, 22 996, 22 997, 22 997, 22 997, 22 997, 22 996, 22 996, 22 997, 21 996, 21 997},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}] Test2=
 {Grid[Join[{{, Statistic, P-Value}}, Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[If[Min[0, DistributionFitTest[{0., 0., 0., 0.,
                0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}]] ≥ 0.025, {KSampleT}, {KruskalWallis}]]}],
    Transpose[{Table[If[Length[Statistics`LocationEquivalenceTestingDump`i] == 0, Statistics`LocationEquivalenceTestingDump`i,
        Statistics`LocationEquivalenceTestingDump`i[[1]]], {Statistics`LocationEquivalenceTestingDump`i,
        Table[HypothesisTestData[≪LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i]}],
      If[Im[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
          Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
          Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][HypothesisTestData[
             ≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i, If[Min[0, DistributionFitTest[{0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}]] ≥ 0.025, {KSampleT}, {KruskalWallis}]]}]}]] <
         1
       ─────, Clip[Re[N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[
       10 000
           ≪LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
          Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][HypothesisTestData[
             ≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i, If[Min[0, DistributionFitTest[{0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}]] ≥ 0.025, {KSampleT}, {KruskalWallis}]]}]}],
        MachinePrecision]], {0, 1}], N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[
        HypothesisTestData[≪LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i],
       {Statistics`LocationEquivalenceTestingDump`i, Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[
           Statistics`LocationEquivalenceTestingDump`i][HypothesisTestData[≪LocationEquivalenceTest≫]],
         {Statistics`LocationEquivalenceTestingDump`i, If[Min[0, DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
               0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}]] ≥ 0.025, {KSampleT}, {KruskalWallis}]]}]}], MachinePrecision]]}], 2],
   Alignment → {Left, Automatic}, Dividers → {{2 → GrayLevel[0.7]}, {2 → GrayLevel[0.7]}}, Spacings →
    Automatic], Min[
    0,
    DistributionFitTest[
     {0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
      0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
      0., 0., 0., 0., 0., 0., 0., 0., 0.}]] ≥ 0.025}
```

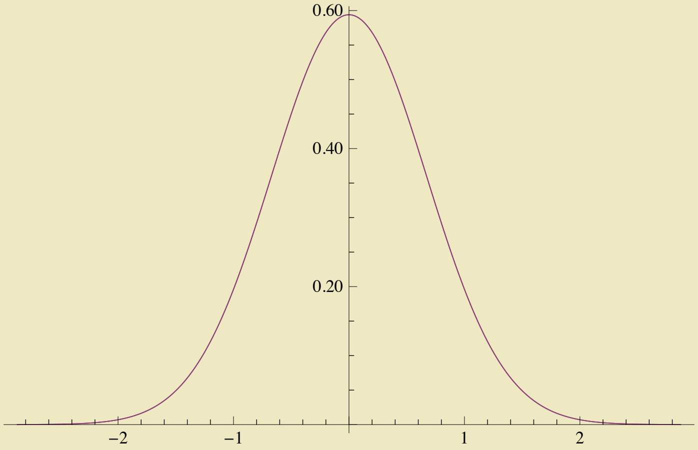


```
-------------------------------------------------------

Elp: (30 values) Between sample 1 and 2. Test1=
 2.87183 × 10^-11 Test2={                 Statistic  P-Value             , KruskalWallis}
                         Kruskal-Wallis   44.3597    3.34347 × 10^-19
```

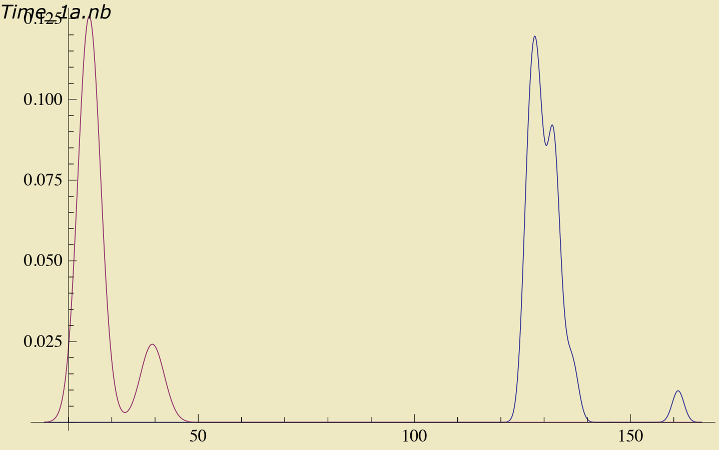


```
----------------------------------------------------------------------------------------------

MannWhitneyTest::invltd :  The argument
    {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0}} should be a vector of real numbers with positive variance, a real matrix with
    positive definite covariance and dimension less than length, or two such vectors or matricies of equal dimension. >>

Join::heads :  Heads List and Transpose at positions 1 and 2 are expected to be the same. >>

Cpu: (30 values) Between sample 2 and 3. Test1=
 MannWhitneyTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]
  Test2={Grid[Join[{{, Statistic, P-Value}}, Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[
         If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
           PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
               0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.},
             {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
               0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}],
       Transpose[{Table[If[Length[Statistics`LocationEquivalenceTestingDump`i] == 0, Statistics`LocationEquivalenceTestingDump`i,
           Statistics`LocationEquivalenceTestingDump`i[[1]]], {Statistics`LocationEquivalenceTestingDump`i,
           Table[HypothesisTestData[<<LocationEquivalenceTest>>], Statistics`LocationEquivalenceTestingDump`i]}],
         If[Im[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[<<LocationEquivalenceTest>>],
               Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
               Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                 HypothesisTestData[<<LocationEquivalenceTest>>]], {Statistics`LocationEquivalenceTestingDump`i,
                 If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥
                     0.025 && PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0.,
                       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                                                                                                                                              1
                       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]]] < ------,
                                                                                                                                            10 000
           Clip[Re[N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[<<LocationEquivalenceTest>>],
                 Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
                 Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                     HypothesisTestData[<<LocationEquivalenceTest>>]], {Statistics`LocationEquivalenceTestingDump`i, If[DistributionFitTest[
                         {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                       PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]], MachinePrecision]], {0, 1}],
           N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[<<LocationEquivalenceTest>>],
               Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
               Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                   HypothesisTestData[<<LocationEquivalenceTest>>]], {Statistics`LocationEquivalenceTestingDump`i,
                   If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                     PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]], MachinePrecision]]}],
       2], Alignment → {Left, Automatic}, Dividers → {{2 → GrayLevel[0.7]}, {2 → GrayLevel[0.7]}},
     Spacings →
       Automatic],
  DistributionFitTest[
     {0
```

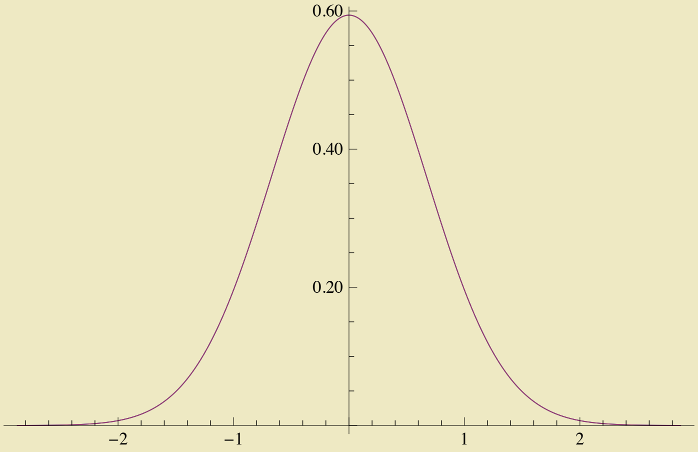
```
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.}] ≥ 0.025 &&
    PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0.,
        0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.},
      {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0.}] > 0.05}
```



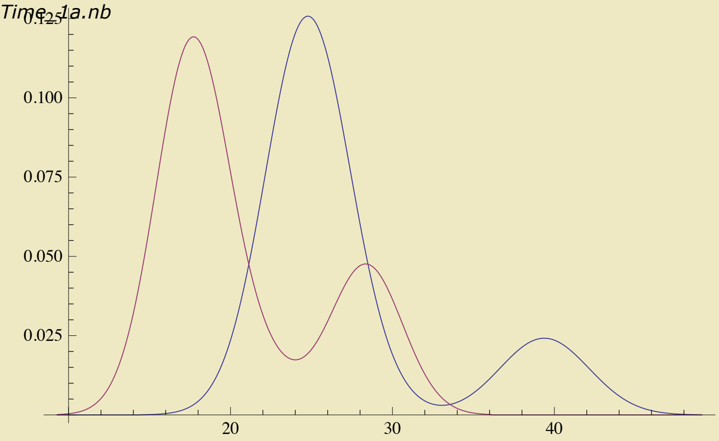
```
-----------------------------------------------------

Elp: (30 values) Between sample 2 and 3. Test1=
 1.63511 × 10^-11 Test2={                 | Statistic  P-Value
                         Kruskal-Wallis  | 45.4636    3.40156 × 10^-20 , KruskalWallis}
```



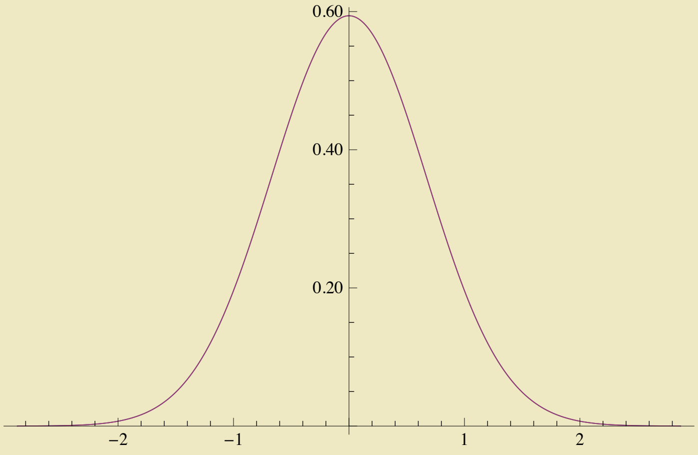
```
---------------------------------------------------------------------------------------------

MannWhitneyTest::invltd :  The argument
        {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0}} should be a vector of real numbers with positive variance, a real matrix with
        positive definite covariance and dimension less than length, or two such vectors or matricies of equal dimension. ≫

General::stop :  Further output of MannWhitneyTest::invltd will be suppressed during this calculation. ≫

Join::heads :  Heads List and Transpose at positions 1 and 2 are expected to be the same. ≫

General::stop :  Further output of Join::heads will be suppressed during this calculation. ≫

Cpu: (30 values) Between sample 3 and 4. Test1=
 MannWhitneyTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]
  Test2={Grid[Join[{{, Statistic, P-Value}}, Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[
            If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
               PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                   0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.},
                 {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                   0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}],
        Transpose[{Table[If[Length[Statistics`LocationEquivalenceTestingDump`i] == 0, Statistics`LocationEquivalenceTestingDump`i,
             Statistics`LocationEquivalenceTestingDump`i[[1]]], {Statistics`LocationEquivalenceTestingDump`i,
             Table[HypothesisTestData[≪LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i]}],
           If[Im[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
                 Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
                 Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                   HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i,
                   If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥
                       0.025 && PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0.,
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                                                                                                                                      1
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]]] < -----,
                                                                                                                                    10000
            Clip[Re[N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
                   Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
                   Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                     HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i, If[DistributionFitTest[
                         {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                       PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]], MachinePrecision]], {0, 1}],
            N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
                 Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
                 Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                   HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i,
                   If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                     PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]], MachinePrecision]}],
        2], Alignment → {Left, Automatic}, Dividers → {{2 → GrayLevel[0.7]}, {2 → GrayLevel[0.7]}},
     Automatic],
   DistributionFitTest[
      {0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.,
       0.}] ≥ 0.025 &&
    PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0.,
       0., 0., 0., 0.,
       0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0.},
      {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0.}] > 0.05}
```



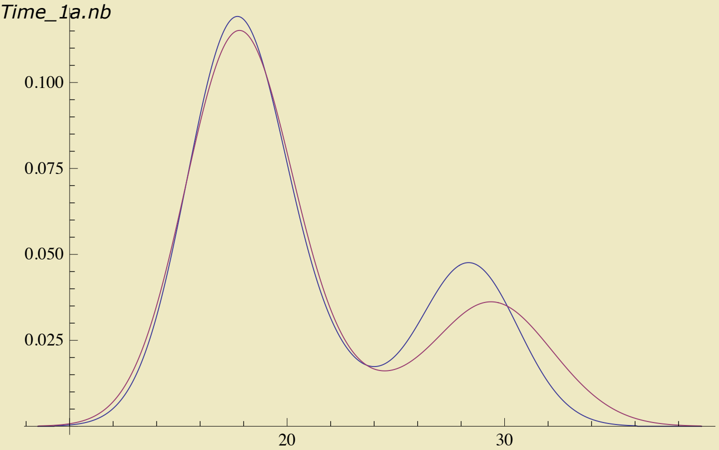
```
-----------------------------------------------------

Elp: (30 values) Between sample 3 and 4. Test1=
 0.000150955 Test2={                | Statistic  P-Value      , KruskalWallis}
                    ----------------+------------------------
                     Kruskal-Wallis | 14.4237    0.0000593948
```



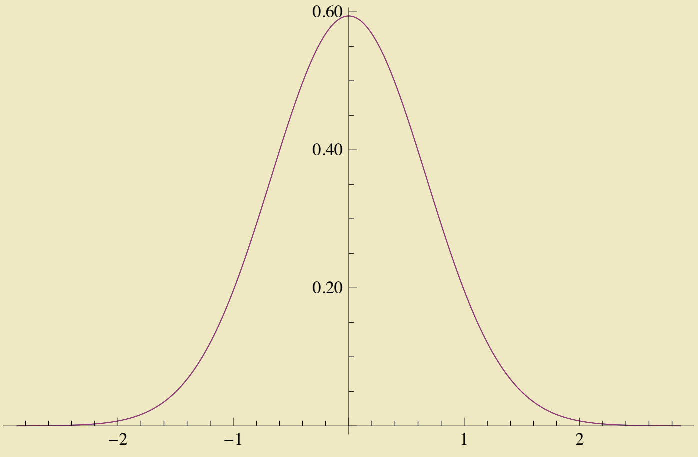
```
----------------------------------------------------------------------------------------------------

Cpu: (30 values) Between sample 4 and 5. Test1=
 MannWhitneyTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]
  Test2={Grid[Join[{{, Statistic, P-Value}}, Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[
        If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
          PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
             0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.},
            {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
             0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}],
    Transpose[{Table[If[Length[Statistics`LocationEquivalenceTestingDump`i] == 0, Statistics`LocationEquivalenceTestingDump`i,
        Statistics`LocationEquivalenceTestingDump`i[[1]]], {Statistics`LocationEquivalenceTestingDump`i,
        Table[HypothesisTestData[≪LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i]}],
      If[Im[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
           Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
           Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
             HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i,
             If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥
               0.025 && PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                                                                                                                            1
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]]] < ─────,
                                                                                                                          10 000
        Clip[Re[N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
             Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
             Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
               HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i, If[DistributionFitTest[
                 {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                  0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                  0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                  0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]]], MachinePrecision]], {0, 1}],
        N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
           Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
           Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
             HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i,
             If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
               PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}]]], MachinePrecision]]}],
    2], Alignment → {Left, Automatic}, Dividers → {{2 → GrayLevel[0.7]}, {2 → GrayLevel[0.7]}},
   Spacings →
    Automatic],
  DistributionFitTest[
     {0.,
      0.,
      0.,
      0.,
      0.,
      0.,
      0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.}] ≥ 0.025 &&
    PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0.,
        0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.},
       {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0.}] > 0.05}
```



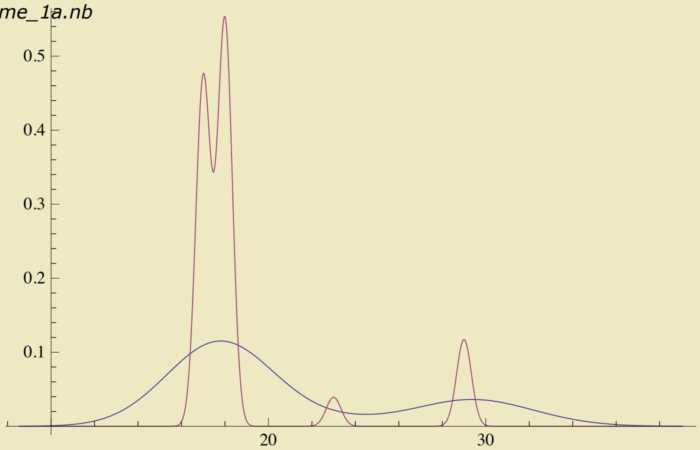
```
-----------------------------------------------------

Elp: (30 values) Between sample 4 and 5. Test1=0.708988 Test2={                 Statistic  P-Value   , KruskalWallis}
                                                               Kruskal-Wallis  0.141826   0.709884
```



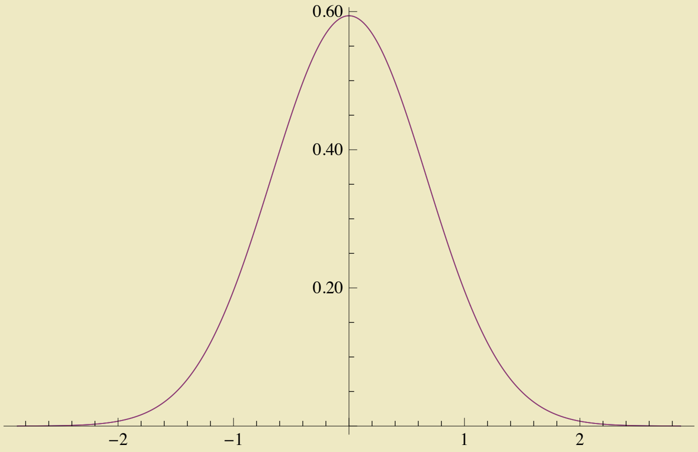
```
----------------------------------------------------------------------------------------------

Cpu: (30 values) Between sample 5 and 6. Test1=
 MannWhitneyTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]
   Test2={Grid[Join[{{, Statistic, P-Value}}, Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[
            If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
              PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.},
                {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}],
     Transpose[{Table[If[Length[Statistics`LocationEquivalenceTestingDump`i] == 0, Statistics`LocationEquivalenceTestingDump`i,
          Statistics`LocationEquivalenceTestingDump`i[[1]]], {Statistics`LocationEquivalenceTestingDump`i,
          Table[HypothesisTestData[≪LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i]}],
        If[Im[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
                Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
                Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                  HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i,
                  If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥
                     0.025 && PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0.,
                       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                                                                                                                                            1
                       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]}]}]] < ──────,
                                                                                                                                         10 000
          Clip[Re[N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
                 Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
                 Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                   HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i, If[DistributionFitTest[
                      {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                     PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]}]}], MachinePrecision]], {0, 1}],
          N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
              Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
              Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i,
                If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                  PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                     0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                     0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                     0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]}]}], MachinePrecision]]}],
     2], Alignment → {Left, Automatic}, Dividers → {{2 → GrayLevel[0.7]}, {2 → GrayLevel[0.7]}},
    Spacings →
     Automatic],
   DistributionFitTest[
      {0.,
       0.,
       0.,
       0.,
       0.,
       0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.,
        0.}] ≥ 0.025 &&
    PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0.,
        0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0.},
       {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0.}] > 0.05}
```



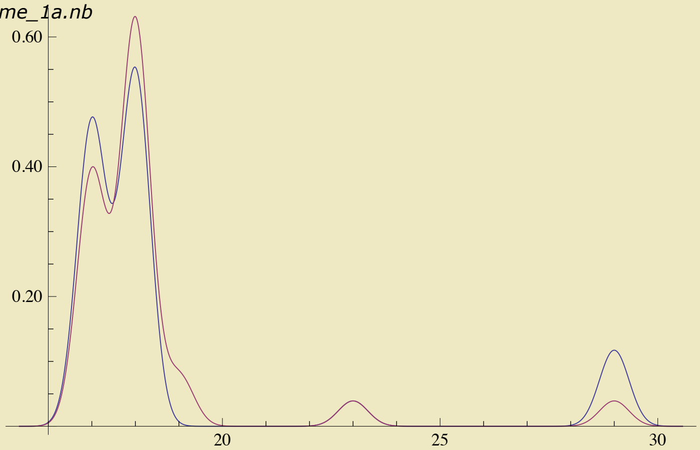
```
--------------------------------------------------------

                                                                                Statistic  P-Value
Elp: (30 values) Between sample 5 and 6. Test1=0.0564935 Test2={ Kruskal-Wallis 4.01628    0.0440634 , KruskalWallis}
```



```
------------------------------------------------------------------------------------------

Cpu: (30 values) Between sample 6 and 7. Test1=
 MannWhitneyTest[{{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
   {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}]
  Test2={Grid[Join[{{, Statistic, P-Value}}, Transpose[{Statistics`LocationEquivalenceTestingDump`iFormatTestNames[
          If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
            PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
               0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.},
              {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
               0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]]}],
    Transpose[{Table[If[Length[Statistics`LocationEquivalenceTestingDump`i] == 0, Statistics`LocationEquivalenceTestingDump`i,
         Statistics`LocationEquivalenceTestingDump`i[[1]]], {Statistics`LocationEquivalenceTestingDump`i,
         Table[HypothesisTestData[≪LocationEquivalenceTest≫], Statistics`LocationEquivalenceTestingDump`i]}],
       If[Im[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
              Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
              Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i,
                If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥
                  0.025 && PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0.,
                    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                                                                                                                                        1
                    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]}]}]] < ------,
                                                                                                                                      10000
          Clip[Re[N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
               Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
               Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                  HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i, If[DistributionFitTest[
                     {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                  PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                      0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                      0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                      0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]}]}], MachinePrecision]], {0, 1}],
          N[Table[Statistics`LocationEquivalenceTestingDump`iGetPValueForSpecificTest[HypothesisTestData[≪LocationEquivalenceTest≫],
             Statistics`LocationEquivalenceTestingDump`i], {Statistics`LocationEquivalenceTestingDump`i,
             Table[Statistics`LocationEquivalenceTestingDump`iTestNameParser[Statistics`LocationEquivalenceTestingDump`i][
                HypothesisTestData[≪LocationEquivalenceTest≫]], {Statistics`LocationEquivalenceTestingDump`i,
                If[DistributionFitTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] ≥ 0.025 &&
                  PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                     0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, {0., 0., 0., 0., 0., 0.,
                     0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                     0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}] > 0.05, {KSampleT}, {KruskalWallis}]}]}], MachinePrecision]}],
      2], Alignment → {Left, Automatic}, Dividers → {{2 → GrayLevel[0.7]}, {2 → GrayLevel[0.7]}},
     Spacings →
      Automatic],
  DistributionFitTest[
     {0.,
      0.,
      0.,
      0.,
      0.,
      0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.,
         0.}] ≥ 0.025 &&
      PearsonChiSquareTest[{0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0.,
         0., 0., 0., 0.,
         0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0.},
        {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0.}] > 0.05}
```



```
--------------------------------------------------

                                                                      Statistic  P-Value
Elp: (30 values) Between sample 6 and 7. Test1=0.74508 Test2={ Kruskal-Wallis  0.116753   0.735745 , KruskalWallis}
```



```
-----------------------------------------------------------------------------------------------
```

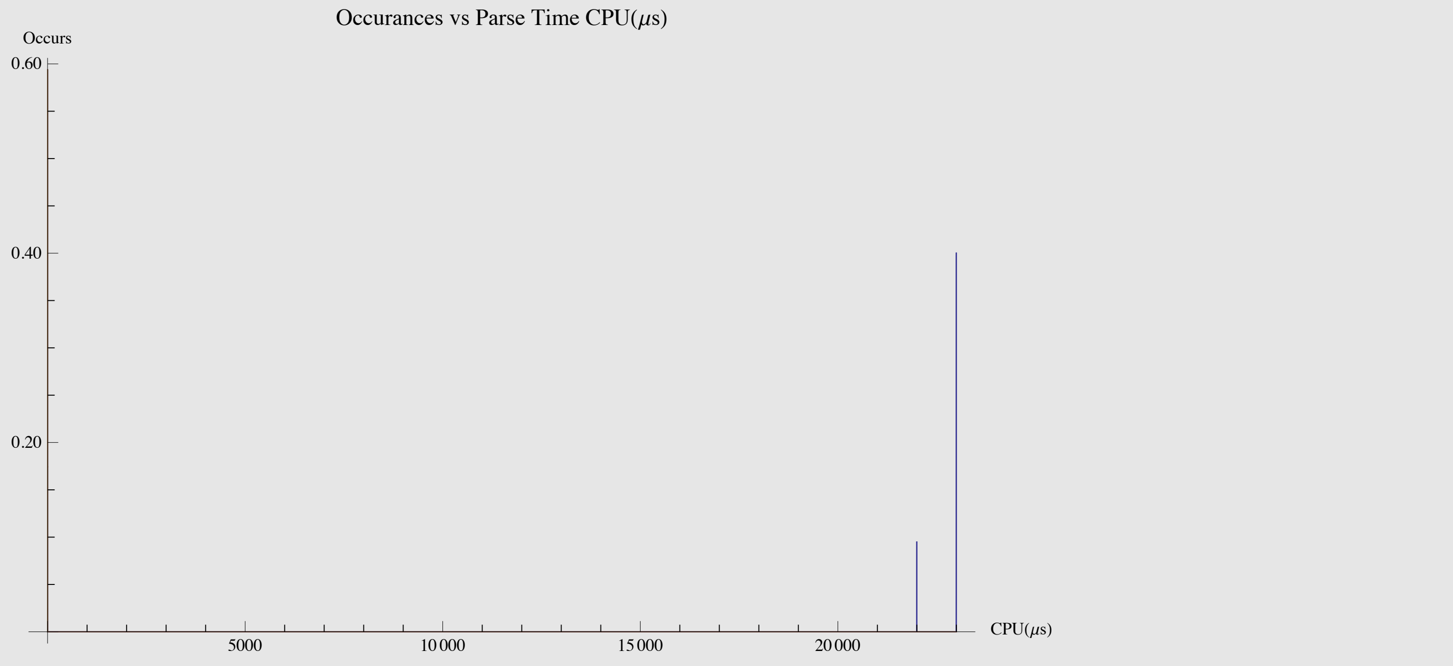
Visually Comparing Sample Sets

I also wanted to get a nice visual picture of my sample sets...together. Sometimes I include all the sample sets and sometime don't. It's just based on what I want to convey. Sometimes you get a more appropriate view if all the data is not included.

In[27]:=

```
SmoothHistogram[{ssCpu[1], ssCpu[2], ssCpu[3], ssCpu[4], ssCpu[5], ssCpu[6], ssCpu[7]},
 PlotLabel → "Occurances vs Parse Time CPU(μs)", AxesLabel → {"CPU(μs)", "Occurs"}]
```

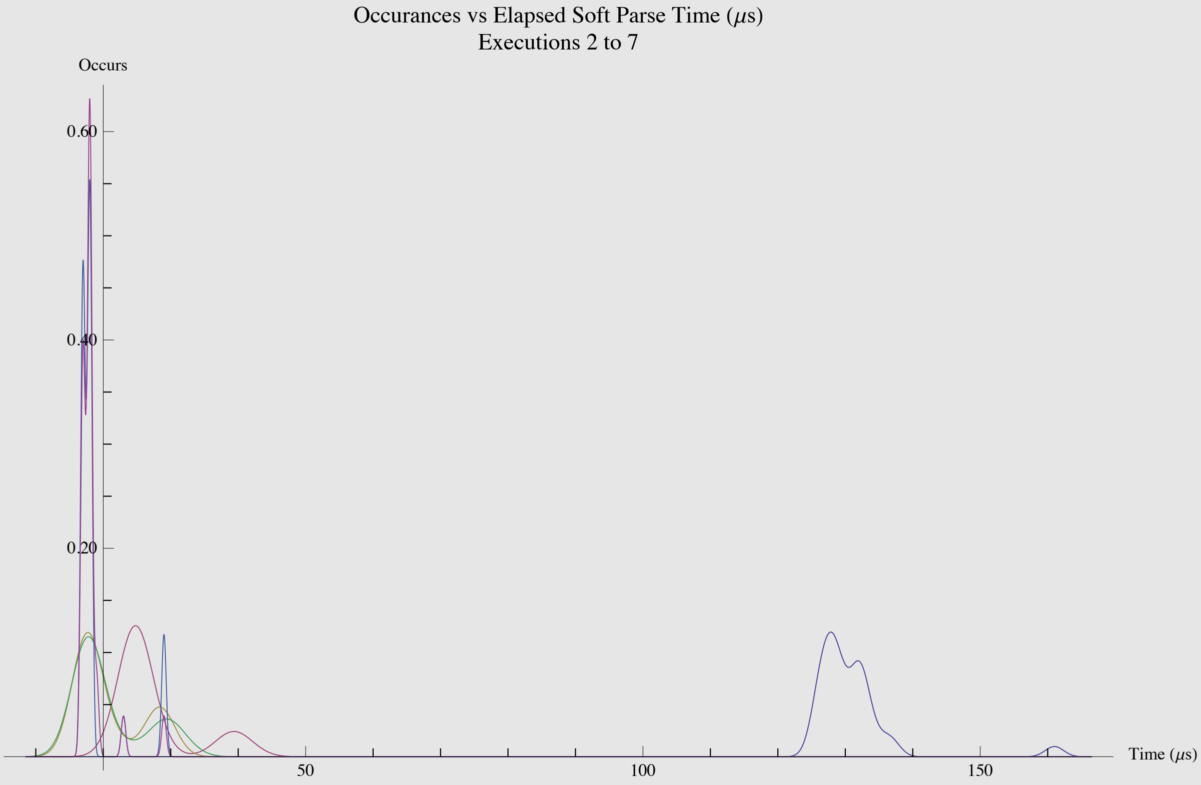
Out[27]=



In[28]:=

```
SmoothHistogram[{ssElp[2], ssElp[3], ssElp[4], ssElp[5], ssElp[6], ssElp[7]},
 PlotLabel → "Occurances vs Elapsed Soft Parse Time (μs)\nExecutions 2 to 7",
 AxesLabel → {"Time (μs)", "Occurs"}]
```

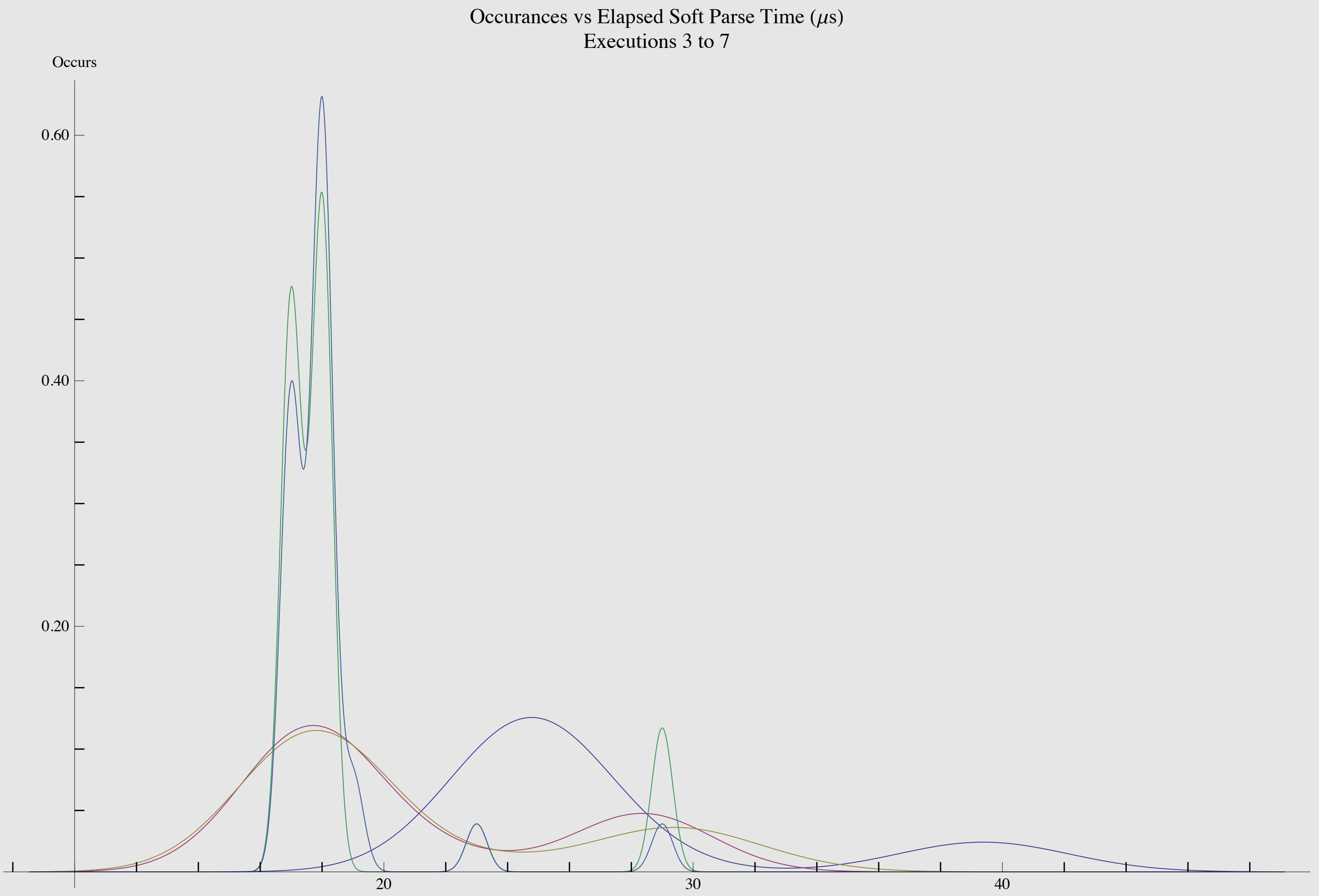
Out[28]=



In[29]:=

```
SmoothHistogram[{ssElp[3], ssElp[4], ssElp[5], ssElp[6], ssElp[7]},
 PlotLabel → "Occurances vs Elapsed Soft Parse Time (μs)\nExecutions 3 to 7",
 AxesLabel → {"Time (μs)", "Occurs"}]
```

Out[29]=


Occurances vs Elapsed Soft Parse Time (μs)
Executions 3 to 7
Occurs