# Wolfram Conference Demo

Based on the presentation at https://www.youtube.com/watch?v=JTz4wkD-mxU

## Central Limit Theorem

Note that the Pareto distribution is the most talked about in real life. Not the normal distribution. People are also excited about what is actually realised.

We first define a Pareto random variable.

In[∘]:=
```
w := RandomVariate[ParetoDistribution[1, 2]]
```

Here is the help menu for this.

In[∘]:=
```
?? RandomVariate
```

Out[∘]=

Symbol

RandomVariate[*dist*] gives a pseudorandom variate from the symbolic distribution *dist*.

RandomVariate[*dist*, *n*] gives a list of *n* pseudorandom variates from the symbolic distribution *dist*.

RandomVariate[*dist*, {$n_1$, $n_2$, …}] gives an $n_1 \times n_2 \times \ldots$

array of pseudorandom variates from the symbolic distribution *dist*.

| | |
|---|---|
| Documentation | Local » \| Web » |
| Options | {Method → Automatic, WorkingPrecision → Automatic} |
| Attributes | {Protected} |
| Full Name | System`RandomVariate |

A Pareto distribution is defined by its minimum value k and shape $\alpha$.

In[∘]:=
```
?? ParetoDistribution
```

Out[∘]=

Symbol

ParetoDistribution[$k$, $\alpha$] represents a Pareto

distribution with minimum value parameter $k$ and shape parameter $\alpha$.

ParetoDistribution[$k$, $\alpha$, $\mu$] represents a Pareto type II distribution with location parameter $\mu$.

ParetoDistribution[$k$, $\alpha$, $\gamma$, $\mu$] represents a Pareto type IV distribution with shape parameter $\gamma$.

| | |
|---|---|
| Documentation | Local » \| Web » |
| Attributes | {Protected, ReadProtected} |
| Full Name | System`ParetoDistribution |

This code adds up the two random variables.

In[∘]:=
```
{w, w} // Total
```
Out[∘]=
```
2.44889
```

The PostFix form allows you to run a function afterwards. Note that this represents the sum of all random variables in a list.

In[∘]:=
```
? //
```
Out[∘]=

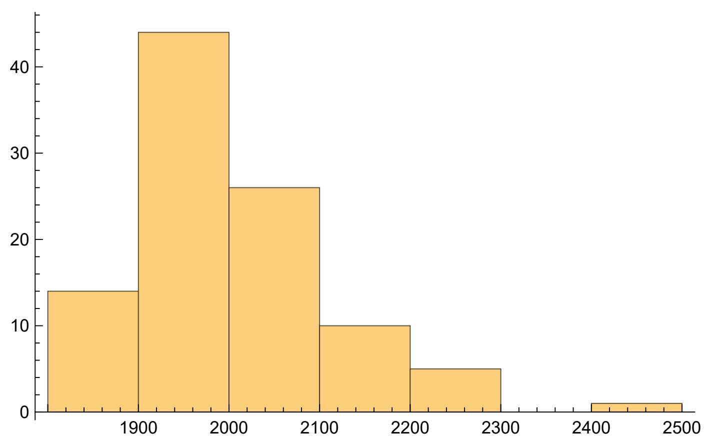
Symbol

Postfix[*f*[*expr*]] prints with *f*[*expr*] given in default postfix form: *expr* // *f*.

Postfix[*f*[*expr*], *h*] prints as *exprh*.

Create a list of 1000 Pareto random variables. Take the total. Run the simulation 100 times.

In[∘]:=
```
Histogram[Table[Table[w, {1000}] // Total, {100}]]
```
Out[∘]=




The observation for the Pareto random variable is this. It does not look remotely Gaussian after running the simulation 100 times. The question is: when will the law of large numbers apply? And also, is there a way to verify that it will eventually become a Gaussian using Mathematica as opposed to using an empirical heuristic?

In[∘]:=
```
y := RandomVariate[GammaDistribution[1, 1]]
```

We define a random variable that follows the gamma distribution.
Here is the help for this.

In[∘]:=
```
?? GammaDistribution
```
Out[∘]=

Symbol

GammaDistribution[$\alpha$, $\beta$] represents a gamma distribution with shape parameter $\alpha$ and scale parameter $\beta$.

GammaDistribution[$\alpha$, $\beta$, $\gamma$, $\mu$] represents a generalized gamma

distribution with shape parameters $\alpha$ and $\gamma$, scale parameter $\beta$, and location parameter $\mu$.

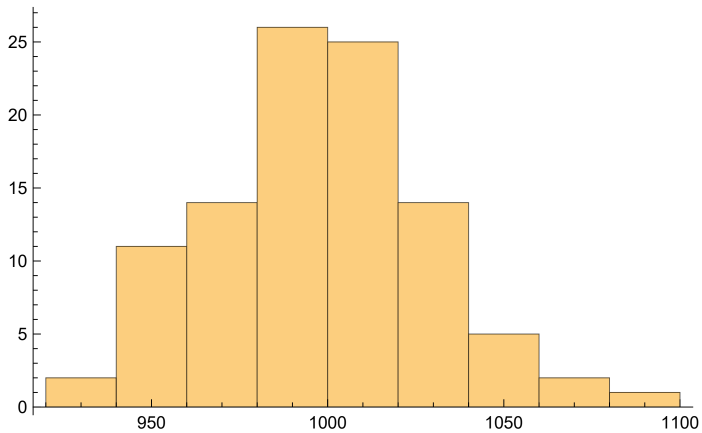
Documentation Local » | Web »

Attributes {Protected, ReadProtected}

Full Name System`GammaDistribution

We can plot a histogram of the totals of 1000 gamma distributed variables, and run this 100 times. Compile a table of these tables and plot a histogram.

In[∘]:=
```
Histogram[Table[Table[y, {1000}] // Total, {100}]]
```
Out[∘]=




This looks like a bell-curve shape or a normal distribution.

In[∘]:=
```
? Plot
```
Out[∘]=

Symbol

Plot[$f$, {$x$, $x_{min}$, $x_{max}$}] generates a plot of $f$ as a function of $x$ from $x_{min}$ to $x_{max}$.

Plot[{$f_1$, $f_2$, …}, {$x$, $x_{min}$, $x_{max}$}] plots several functions $f_i$.

Plot[{…, $w$[$f_i$], …}, …] plots $f_i$ with features defined by the symbolic wrapper $w$.

Plot[…, {$x$} ∈ $reg$] takes the variable $x$ to be in the geometric region $reg$.

In[ ]:= 
```
? PDF
```
Out[ ]=

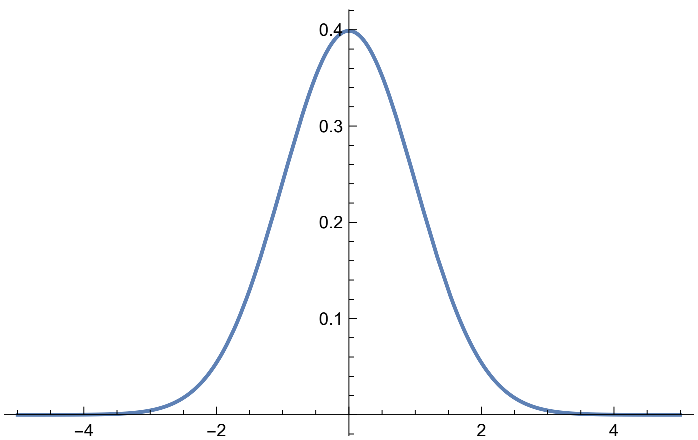
Symbol

PDF[*dist*, $x$] gives the probability density function for the distribution *dist* evaluated at $x$.

PDF[*dist*, $\{x_1, x_2, \ldots\}$] gives the multivariate probability density function for a distribution *dist* evaluated at $\{x_1, x_2, \ldots\}$.

PDF[*dist*] gives the PDF as a pure function.

In[ ]:= 
```
Plot[PDF[NormalDistribution[0, 1], x], {x, -5, 5}]
```
Out[ ]=




In[ ]:= 
```
? Table
```
Out[ ]=

Symbol

Table[*expr*, $n$] generates a list of $n$ copies of *expr*.

Table[*expr*, $\{i, i_{max}\}$] generates a list of the values of *expr* when $i$ runs from 1 to $i_{max}$.

Table[*expr*, $\{i, i_{min}, i_{max}\}$] starts with $i = i_{min}$.

Table[*expr*, $\{i, i_{min}, i_{max}, di\}$] uses steps $di$.

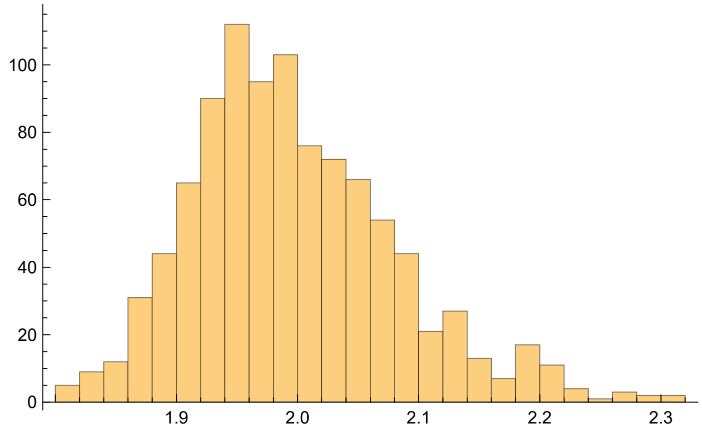
Table[*expr*, $\{i, \{i_1, i_2, \ldots\}\}$] uses the successive values $i_1, i_2, \ldots$.

Table[*expr*, $\{i, i_{min}, i_{max}\}, \{j, j_{min}, j_{max}\}, \ldots$] gives a nested list. The list associated with $i$ is outermost.

Make 30 copies of the Pareto random variable.
Calculate the mean. You will find that it is not stable.

In[ ]:=
```
ta = Table[Table[w, {1000}] // Mean, {1000}] // Histogram
```

Out[ ]=




This leads in to our next section on the law of large numbers.

## Law of Large Numbers

Recall that for a Pareto distribution there is a lower bound and a shape parameter.

In[ ]:=
```
? ParetoDistribution
```

Out[ ]=

Symbol

ParetoDistribution[$k$, $\alpha$] represents a Pareto distribution with minimum value parameter $k$ and shape parameter $\alpha$.

ParetoDistribution[$k$, $\alpha$, $\mu$] represents a Pareto type II distribution with location parameter $\mu$.

ParetoDistribution[$k$, $\alpha$, $\gamma$, $\mu$] represents a Pareto type IV distribution with shape parameter $\gamma$.

We can define a random variable as follows.

In[ ]:=
```
r := RandomVariate[ParetoDistribution[1, 1.14]]
```

In[∘]:=
```
ta = Table[r, {1000}];
DiscretePlot[Mean[ta⟦1 ;; i⟧], {i, 1, Length[ta]}, PlotStyle → Red, PlotRange → All]
```

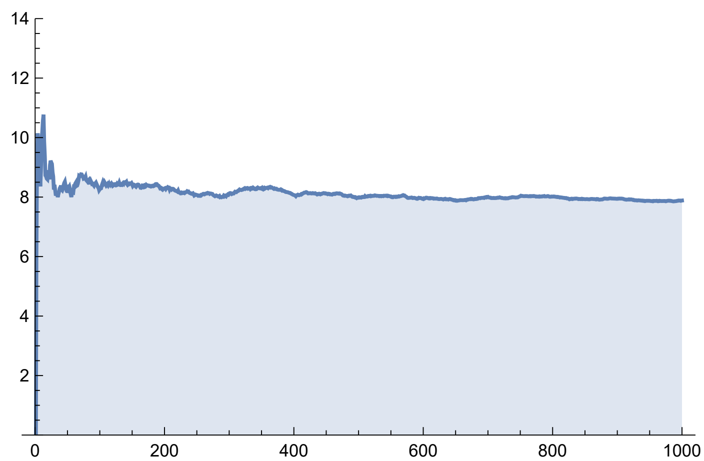
Out[∘]=


Compare this to a normal distribution.

In[∘]:=
```
k := RandomVariate[NormalDistribution[8, 5]]
```

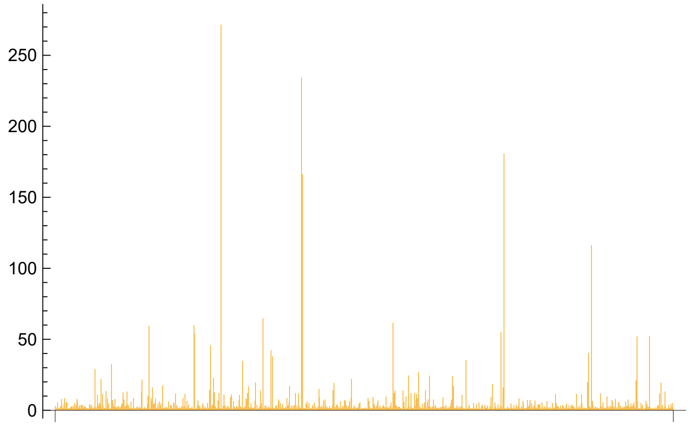
In[∘]:=
```
tak = Table[k, {1000}];
DiscretePlot[Mean[tak⟦1 ;; i⟧], {i, 1, Length[tak]}, PlotRange → {0, 14}]
```

Out[∘]=




Green lumber fallacy: the best trader in green lumber does not know green is freshly cuy.

```
In[•]:= BarChart[ta]
Out[•]=
```




Set up an aid to book to do analytical work on the side.

[Image]

```
In[•]:= Integrate[x^2, {x, 0, x}]
Out[•]=
```

$$\frac{x^3}{3}$$

## covfefe Markov Chain

```
In[•]:= states = {"Fail", "C", "CO", "COV", "COVF", "COVFE", "COVFEF", "COVFEFE"}
Out[•]=
{Fail, C, CO, COV, COVF, COVFE, COVFEF, COVFEFE}
```

One for starting state to fail straight away. One for each state except for failure. One for identity on "covfefe".

```
In[•]:= M = {{25 / 26, 1 / 26, 0, 0, 0, 0, 0, 0}, {24 / 26, 1 / 26, 1 / 26, 0, 0, 0, 0, 0},
   {24 / 26, 1 / 26, 0, 1 / 26, 0, 0, 0, 0}, {24 / 26, 1 / 26, 0, 0, 1 / 26, 0, 0, 0},
   {24 / 26, 1 / 26, 0, 0, 0, 1 / 26, 0, 0}, {24 / 26, 1 / 26, 0, 0, 0, 0, 1 / 26, 0},
   {24 / 26, 1 / 26, 0, 0, 0, 0, 0, 1 / 26}, {0, 0, 0, 0, 0, 0, 0, 1}};
```

In[∘]:= **TableForm[M, TableHeadings → {states, states}]**

Out[∘]//TableForm=

| | Fail | C | CO | COV | COVF | COVFE | COVFEF | COVFEFE |
|---|---|---|---|---|---|---|---|---|
| Fail | $\frac{25}{26}$ | $\frac{1}{26}$ | 0 | 0 | 0 | 0 | 0 | 0 |
| C | $\frac{12}{13}$ | $\frac{1}{26}$ | $\frac{1}{26}$ | 0 | 0 | 0 | 0 | 0 |
| CO | $\frac{12}{13}$ | $\frac{1}{26}$ | 0 | $\frac{1}{26}$ | 0 | 0 | 0 | 0 |
| COV | $\frac{12}{13}$ | $\frac{1}{26}$ | 0 | 0 | $\frac{1}{26}$ | 0 | 0 | 0 |
| COVF | $\frac{12}{13}$ | $\frac{1}{26}$ | 0 | 0 | 0 | $\frac{1}{26}$ | 0 | 0 |
| COVFE | $\frac{12}{13}$ | $\frac{1}{26}$ | 0 | 0 | 0 | 0 | $\frac{1}{26}$ | 0 |
| COVFEF | $\frac{12}{13}$ | $\frac{1}{26}$ | 0 | 0 | 0 | 0 | 0 | $\frac{1}{26}$ |
| COVFEFE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

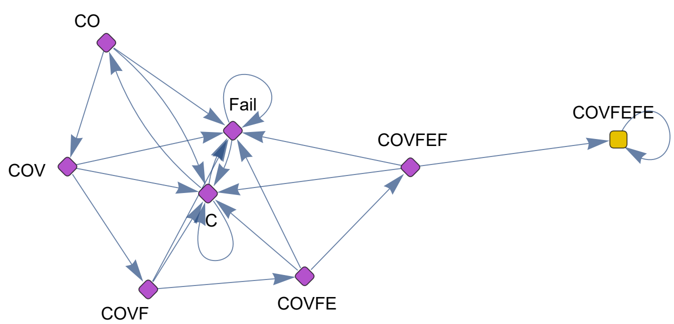
In[∘]:= **procCOVFEFE = DiscreteMarkovProcess[1, M]**

Out[∘]=

$$\text{DiscreteMarkovProcess}\left[1, \left\{\left\{\frac{25}{26}, \frac{1}{26}, 0, 0, 0, 0, 0, 0\right\}, \left\{\frac{12}{13}, \frac{1}{26}, \frac{1}{26}, 0, 0, 0, 0, 0\right\}, \left\{\frac{12}{13}, \frac{1}{26}, 0, \frac{1}{26}, 0, 0, 0, 0\right\}, \left\{\frac{12}{13}, \frac{1}{26}, 0, 0, \frac{1}{26}, 0, 0, 0\right\}, \left\{\frac{12}{13}, \frac{1}{26}, 0, 0, 0, \frac{1}{26}, 0, 0\right\}, \left\{\frac{12}{13}, \frac{1}{26}, 0, 0, 0, 0, \frac{1}{26}, 0\right\}, \left\{\frac{12}{13}, \frac{1}{26}, 0, 0, 0, 0, 0, \frac{1}{26}\right\}, \{0, 0, 0, 0, 0, 0, 0, 1\}\right\}\right]$$

In[∘]:= **Graph[procCOVFEFE, VertexLabels → Table[i → states⟦i⟧, {i, 1, 8}]]**

Out[∘]=

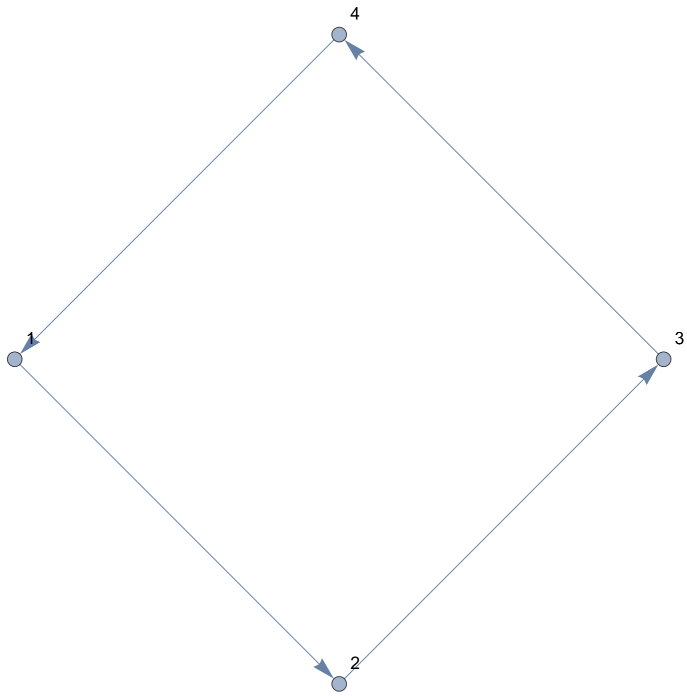



In[∘]:= **Export["graph.jpg", %]**

Out[∘]=

graph.jpg

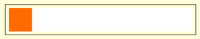
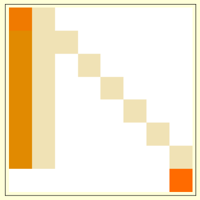
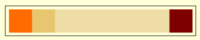
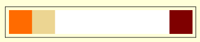
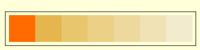
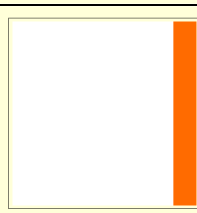
```
In[•]:= Graph[{1 → 2, 2 → 3, 3 → 4, 4 → 1}, VertexLabels → All]
Out[•]=
```




```
In[•]:= FirstPassageTimeDistribution[procCOVFEFE, 8] // Mean
Out[•]=
        8 031 810 176

In[•]:= MarkovProcessProperties[procCOVFEFE]
        Eigenvalues[M] // N
```

Out[ ]=

| | |
|---|---|
| **Basic Properties** | |
| InitialProbabilities |  |
| TransitionMatrix |  |
| HoldingTimeMean |  |
| HoldingTimeVariance |  |
| **Structural Properties** | |
| CommunicatingClasses | {8}, {1, …, 7} |
| RecurrentClasses | {8} |
| TransientClasses | {1, …, 7} |
| AbsorbingClasses | {8} |
| PeriodicClasses | None |
| Periods | {} |
| Irreducible | False |
| Aperiodic | True |
| Primitive | False |
| **Transient Properties** | |
| TransientVisitMean |  |
| TransientVisitVariance |  |
| TransientTotalVisitMean | 7 |
| **Limiting Properties** | |
| ReachabilityProbability |  |
| LimitTransitionMatrix |  |
| Reversible | False |

Out[•]=

```
{1., 1., 0.0224305, 0.0111299 + 0.0194242 i, 0.0111299 - 0.0194242 i,
 -0.0112132 + 0.0192801 i, -0.0112132 - 0.0192801 i, -0.022264}
```

## AiryAI Simplification

Mathematica enables you to basically copy Ramanujan and explore mathematics. Always use fractions on Mathematica to automatically find closed forms.

In[•]:=

```
PDF[StableDistribution[3 / 2, 1, 0, 1], x]
```

Out[•]=

$$-\frac{2^{1/3}\, e^{\frac{x^3}{27}} \left(3^{1/3}\, x\, \text{AiryAi}\left[\frac{x^2}{3\times 2^{2/3}\times 3^{1/3}}\right] + 3\times 2^{1/3}\, \text{AiryAiPrime}\left[\frac{x^2}{3\times 2^{2/3}\times 3^{1/3}}\right]\right)}{3\times 3^{2/3}}$$

Wolfram Engine does not work so well with Manipulate.

In[•]:=

```
Tuples[{a, b, c}, 3]
```

Out[•]=

```
{{a, a, a}, {a, a, b}, {a, a, c}, {a, b, a}, {a, b, b}, {a, b, c},
 {a, c, a}, {a, c, b}, {a, c, c}, {b, a, a}, {b, a, b}, {b, a, c}, {b, b, a},
 {b, b, b}, {b, b, c}, {b, c, a}, {b, c, b}, {b, c, c}, {c, a, a}, {c, a, b},
 {c, a, c}, {c, b, a}, {c, b, b}, {c, b, c}, {c, c, a}, {c, c, b}, {c, c, c}}
```